```
/*!
* jQuery Mobile 1.4.2
* Git HEAD hash: 9d9a42a27d0c693e8b5569c3a10d771916af5045 <> Date: Fri Feb 28 2014 17:32:01 UTC
* http://jquerymobile.com
*
* Copyright 2010, 2014 jQuery Foundation, Inc. and other contributors
* Released under the MIT license.
* http://jquery.org/license
*
*/


(function ( root, doc, factory ) {
    if ( typeof define === "function" && define.amd ) {
        // AMD. Register as an anonymous module.
        define( [ "jquery" ], function ( $ ) {
            factory( $, root, doc );
            return $.mobile;
        });
    } else {
        // Browser globals
        factory( root.jQuery, root, doc );
    }
}( this, document, function ( jQuery, window, document, undefined ) {
(function( $ ) {
    $.mobile = {};
}( jQuery ));

(function( $, window, undefined ) {
    $.extend( $.mobile, {

        // Version of the jQuery Mobile Framework
        version: "1.4.2",

        // Deprecated and no longer used in 1.4 remove in 1.5
        // Define the url parameter used for referencing widget-generated sub-pages.
        // Translates to example.html&ui-page=subpageIdentifier
        // hash segment before &ui-page= is used to make Ajax request
        subPageUrlKey: "ui-page",

        hideUrlBar: true,

        // Keepnative Selector
        keepNative: ":jqmData(role='none'), :jqmData(role='nojs')",

        // Deprecated in 1.4 remove in 1.5
        // Class assigned to page currently in view, and during transitions
        activePageClass: "ui-page-active",

        // Deprecated in 1.4 remove in 1.5
        // Class used for "active" button state, from CSS framework
        activeBtnClass: "ui-btn-active",

        // Deprecated in 1.4 remove in 1.5
```

```
        // Class used for "focus" form element state, from CSS framework
        focusClass: "ui-focus",

        // Automatically handle clicks and form submissions through Ajax, when same-domain
        ajaxEnabled: true,

        // Automatically load and show pages based on location.hash
        hashListeningEnabled: true,

        // disable to prevent jquery from bothering with links
        linkBindingEnabled: true,

        // Set default page transition - 'none' for no transitions
        defaultPageTransition: "fade",

        // Set maximum window width for transitions to apply - 'false' for no limit
        maxTransitionWidth: false,

        // Minimum scroll distance that will be remembered when returning to a page
        // Deprecated remove in 1.5
        minScrollBack: 0,

        // Set default dialog transition - 'none' for no transitions
        defaultDialogTransition: "pop",

        // Error response message - appears when an Ajax page request fails
        pageLoadErrorMessage: "Error Loading Page",

        // For error messages, which theme does the box uses?
        pageLoadErrorMessageTheme: "a",

        // replace calls to window.history.back with phonegaps navigation helper
        // where it is provided on the window object
        phonegapNavigationEnabled: false,

        //automatically initialize the DOM when it's ready
        autoInitializePage: true,

        pushStateEnabled: true,

        // allows users to opt in to ignoring content by marking a parent element as
        // data-ignored
        ignoreContentEnabled: false,

        buttonMarkup: {
            hoverDelay: 200
        },

        // disable the alteration of the dynamic base tag or links in the case
        // that a dynamic base tag isn't supported
        dynamicBaseEnabled: true,

        // default the property to remove dependency on assignment in init module
        pageContainer: $(),
```

```
        //enable cross-domain page support
        allowCrossDomainPages: false,

        dialogHashKey: "&ui-state=dialog"
    });
})( jQuery, this );

(function( $, window, undefined ) {
    var nsNormalizeDict = {},
        oldFind = $.find,
        rbrace = /(?:\{[\s\S]*\}|\[[\s\S]*\])$/,
        jqmDataRE = /:jqmData\(([^)]*)\)/g;

    $.extend( $.mobile, {

        // Namespace used framework-wide for data-attrs. Default is no namespace

        ns: "",

        // Retrieve an attribute from an element and perform some massaging of the value

        getAttribute: function( element, key ) {
            var data;

            element = element.jquery ? element[0] : element;

            if ( element && element.getAttribute ) {
                data = element.getAttribute( "data-" + $.mobile.ns + key );
            }

            // Copied from core's src/data.js:dataAttr()
            // Convert from a string to a proper data type
            try {
                data = data === "true" ? true :
                    data === "false" ? false :
                    data === "null" ? null :
                    // Only convert to a number if it doesn't change the string
                    +data + "" === data ? +data :
                    rbrace.test( data ) ? JSON.parse( data ) :
                    data;
            } catch( err ) {}

            return data;
        },

        // Expose our cache for testing purposes.
        nsNormalizeDict: nsNormalizeDict,

        // Take a data attribute property, prepend the namespace
        // and then camel case the attribute string. Add the result
        // to our nsNormalizeDict so we don't have to do this again.
        nsNormalize: function( prop ) {
            return nsNormalizeDict[ prop ] ||
```

```
            ( nsNormalizeDict[ prop ] = $.camelCase( $.mobile.ns + prop ) );
        },

        // Find the closest javascript page element to gather settings data jsperf test
        // http://jsperf.com/single-complex-selector-vs-many-complex-selectors/edit
        // possibly naive, but it shows that the parsing overhead for *just* the page selector vs
        // the page and dialog selector is negligable. This could probably be speed up by
        // doing a similar parent node traversal to the one found in the inherited theme code
        above
        closestPageData: function( $target ) {
            return $target
                .closest( ":jqmData(role='page'), :jqmData(role='dialog')" )
                .data( "mobile-page" );
        }

    });

    // Mobile version of data and removeData and hasData methods
    // ensures all data is set and retrieved using jQuery Mobile's data namespace
    $.fn.jqmData = function( prop, value ) {
        var result;
        if ( typeof prop !== "undefined" ) {
            if ( prop ) {
                prop = $.mobile.nsNormalize( prop );
            }

            // undefined is permitted as an explicit input for the second param
            // in this case it returns the value and does not set it to undefined
            if ( arguments.length < 2 || value === undefined ) {
                result = this.data( prop );
            } else {
                result = this.data( prop, value );
            }
        }
        return result;
    };

    $.jqmData = function( elem, prop, value ) {
        var result;
        if ( typeof prop !== "undefined" ) {
            result = $.data( elem, prop ? $.mobile.nsNormalize( prop ) : prop, value );
        }
        return result;
    };

    $.fn.jqmRemoveData = function( prop ) {
        return this.removeData( $.mobile.nsNormalize( prop ) );
    };

    $.jqmRemoveData = function( elem, prop ) {
        return $.removeData( elem, $.mobile.nsNormalize( prop ) );
    };

    $.find = function( selector, context, ret, extra ) {
```

```
        if ( selector.indexOf( ":jqmData" ) > -1 ) {
            selector = selector.replace( jqmDataRE, "[data-" + ( $.mobile.ns || "" ) + "$1]" );
        }

        return oldFind.call( this, selector, context, ret, extra );
    };

    $.extend( $.find, oldFind );

})( jQuery, this );

/*!
 * jQuery UI Core c0ab71056b936627e8a7821f03c044aec6280a40
 * http://jqueryui.com
 *
 * Copyright 2013 jQuery Foundation and other contributors
 * Released under the MIT license.
 * http://jquery.org/license
 *
 * http://api.jqueryui.com/category/ui-core/
 */
(function( $, undefined ) {

var uuid = 0,
    runiqueId = /^ui-id-\d+$/;

// $.ui might exist from components with no dependencies, e.g., $.ui.position
$.ui = $.ui || {};

$.extend( $.ui, {
    version: "c0ab71056b936627e8a7821f03c044aec6280a40",

    keyCode: {
        BACKSPACE: 8,
        COMMA: 188,
        DELETE: 46,
        DOWN: 40,
        END: 35,
        ENTER: 13,
        ESCAPE: 27,
        HOME: 36,
        LEFT: 37,
        PAGE_DOWN: 34,
        PAGE_UP: 33,
        PERIOD: 190,
        RIGHT: 39,
        SPACE: 32,
        TAB: 9,
        UP: 38
    }
});

// plugins
$.fn.extend({
```

```
    focus: (function( orig ) {
        return function( delay, fn ) {
            return typeof delay === "number" ?
                this.each(function() {
                    var elem = this;
                    setTimeout(function() {
                        $( elem ).focus();
                        if ( fn ) {
                            fn.call( elem );
                        }
                    }, delay );
                }) :
                orig.apply( this, arguments );
        };
    })( $.fn.focus ),

    scrollParent: function() {
        var scrollParent;
        if (($.ui.ie && (/(static|relative)/).test(this.css("position"))) || (/absolute/).test(
        this.css("position"))) {
            scrollParent = this.parents().filter(function() {
                return (/(relative|absolute|fixed)/).test($.css(this,"position")) && (
                /(auto|scroll)/).test($.css(this,"overflow")+$.css(this,"overflow-y")+$.css(this
                ,"overflow-x"));
            }).eq(0);
        } else {
            scrollParent = this.parents().filter(function() {
                return (/(auto|scroll)/).test($.css(this,"overflow")+$.css(this,"overflow-y")+$.
                css(this,"overflow-x"));
            }).eq(0);
        }

        return ( /fixed/ ).test( this.css( "position") ) || !scrollParent.length ? $( this[ 0 ].
        ownerDocument || document ) : scrollParent;
    },

    uniqueId: function() {
        return this.each(function() {
            if ( !this.id ) {
                this.id = "ui-id-" + (++uuid);
            }
        });
    },

    removeUniqueId: function() {
        return this.each(function() {
            if ( runiqueId.test( this.id ) ) {
                $( this ).removeAttr( "id" );
            }
        });
    }
});

// selectors
```

```
function focusable( element, isTabIndexNotNaN ) {
    var map, mapName, img,
        nodeName = element.nodeName.toLowerCase();
    if ( "area" === nodeName ) {
        map = element.parentNode;
        mapName = map.name;
        if ( !element.href || !mapName || map.nodeName.toLowerCase() !== "map" ) {
            return false;
        }
        img = $( "img[usemap=#" + mapName + "]" )[0];
        return !!img && visible( img );
    }
    return ( /input|select|textarea|button|object/.test( nodeName ) ?
        !element.disabled :
        "a" === nodeName ?
            element.href || isTabIndexNotNaN :
            isTabIndexNotNaN) &&
        // the element and all of its ancestors must be visible
        visible( element );
}

function visible( element ) {
    return $.expr.filters.visible( element ) &&
        !$( element ).parents().addBack().filter(function() {
            return $.css( this, "visibility" ) === "hidden";
        }).length;
}

$.extend( $.expr[ ":" ], {
    data: $.expr.createPseudo ?
        $.expr.createPseudo(function( dataName ) {
            return function( elem ) {
                return !!$.data( elem, dataName );
            };
        }) :
        // support: jQuery <1.8
        function( elem, i, match ) {
            return !!$.data( elem, match[ 3 ] );
        },

    focusable: function( element ) {
        return focusable( element, !isNaN( $.attr( element, "tabindex" ) ) );
    },

    tabbable: function( element ) {
        var tabIndex = $.attr( element, "tabindex" ),
            isTabIndexNaN = isNaN( tabIndex );
        return ( isTabIndexNaN || tabIndex >= 0 ) && focusable( element, !isTabIndexNaN );
    }
});

// support: jQuery <1.8
if ( !$( "<a>" ).outerWidth( 1 ).jquery ) {
    $.each( [ "Width", "Height" ], function( i, name ) {
```

```
        var side = name === "Width" ? [ "Left", "Right" ] : [ "Top", "Bottom" ],
            type = name.toLowerCase(),
            orig = {
                innerWidth: $.fn.innerWidth,
                innerHeight: $.fn.innerHeight,
                outerWidth: $.fn.outerWidth,
                outerHeight: $.fn.outerHeight
            };

        function reduce( elem, size, border, margin ) {
            $.each( side, function() {
                size -= parseFloat( $.css( elem, "padding" + this ) ) || 0;
                if ( border ) {
                    size -= parseFloat( $.css( elem, "border" + this + "Width" ) ) || 0;
                }
                if ( margin ) {
                    size -= parseFloat( $.css( elem, "margin" + this ) ) || 0;
                }
            });
            return size;
        }

        $.fn[ "inner" + name ] = function( size ) {
            if ( size === undefined ) {
                return orig[ "inner" + name ].call( this );
            }

            return this.each(function() {
                $( this ).css( type, reduce( this, size ) + "px" );
            });
        };

        $.fn[ "outer" + name] = function( size, margin ) {
            if ( typeof size !== "number" ) {
                return orig[ "outer" + name ].call( this, size );
            }

            return this.each(function() {
                $( this).css( type, reduce( this, size, true, margin ) + "px" );
            });
        };
    });
}

// support: jQuery <1.8
if ( !$.fn.addBack ) {
    $.fn.addBack = function( selector ) {
        return this.add( selector == null ?
            this.prevObject : this.prevObject.filter( selector )
        );
    };
}

// support: jQuery 1.6.1, 1.6.2 (http://bugs.jquery.com/ticket/9413)
```

```
if ( $( "<a>" ).data( "a-b", "a" ).removeData( "a-b" ).data( "a-b" ) ) {
    $.fn.removeData = (function( removeData ) {
        return function( key ) {
            if ( arguments.length ) {
                return removeData.call( this, $.camelCase( key ) );
            } else {
                return removeData.call( this );
            }
        };
    })( $.fn.removeData );
}




// deprecated
$.ui.ie = !!/msie [\w.]+/.exec( navigator.userAgent.toLowerCase() );

$.support.selectstart = "onselectstart" in document.createElement( "div" );
$.fn.extend({
    disableSelection: function() {
        return this.bind( ( $.support.selectstart ? "selectstart" : "mousedown" ) +
            ".ui-disableSelection", function( event ) {
                event.preventDefault();
            });
    },

    enableSelection: function() {
        return this.unbind( ".ui-disableSelection" );
    },

    zIndex: function( zIndex ) {
        if ( zIndex !== undefined ) {
            return this.css( "zIndex", zIndex );
        }

        if ( this.length ) {
            var elem = $( this[ 0 ] ), position, value;
            while ( elem.length && elem[ 0 ] !== document ) {
                // Ignore z-index if position is set to a value where z-index is ignored by the
                browser
                // This makes behavior of this function consistent across browsers
                // WebKit always returns auto if the element is positioned
                position = elem.css( "position" );
                if ( position === "absolute" || position === "relative" || position === "fixed"
                ) {
                    // IE returns 0 when zIndex is not specified
                    // other browsers return a string
                    // we ignore the case of nested elements with an explicit value of 0
                    // <div style="z-index: -10;"><div style="z-index: 0;"></div></div>
                    value = parseInt( elem.css( "zIndex" ), 10 );
                    if ( !isNaN( value ) && value !== 0 ) {
                        return value;
```

```
                    }
                }
                elem = elem.parent();
            }
        }

        return 0;
    }
});

// $.ui.plugin is deprecated. Use $.widget() extensions instead.
$.ui.plugin = {
    add: function( module, option, set ) {
        var i,
            proto = $.ui[ module ].prototype;
        for ( i in set ) {
            proto.plugins[ i ] = proto.plugins[ i ] || [];
            proto.plugins[ i ].push( [ option, set[ i ] ] );
        }
    },
    call: function( instance, name, args, allowDisconnected ) {
        var i,
            set = instance.plugins[ name ];

        if ( !set ) {
            return;
        }

        if ( !allowDisconnected && ( !instance.element[ 0 ].parentNode || instance.element[ 0 ].
        parentNode.nodeType === 11 ) ) {
            return;
        }

        for ( i = 0; i < set.length; i++ ) {
            if ( instance.options[ set[ i ][ 0 ] ] ) {
                set[ i ][ 1 ].apply( instance.element, args );
            }
        }
    }
};

})( jQuery );

(function( $, window, undefined ) {

    // Subtract the height of external toolbars from the page height, if the page does not have
    // internal toolbars of the same type
    var compensateToolbars = function( page, desiredHeight ) {
        var pageParent = page.parent(),
            toolbarsAffectingHeight = [],
            externalHeaders = pageParent.children( ":jqmData(role='header')" ),
            internalHeaders = page.children( ":jqmData(role='header')" ),
            externalFooters = pageParent.children( ":jqmData(role='footer')" ),
            internalFooters = page.children( ":jqmData(role='footer')" );
```

```
        // If we have no internal headers, but we do have external headers, then their height
        // reduces the page height
        if ( internalHeaders.length === 0 && externalHeaders.length > 0 ) {
            toolbarsAffectingHeight = toolbarsAffectingHeight.concat( externalHeaders.toArray()
            );
        }

        // If we have no internal footers, but we do have external footers, then their height
        // reduces the page height
        if ( internalFooters.length === 0 && externalFooters.length > 0 ) {
            toolbarsAffectingHeight = toolbarsAffectingHeight.concat( externalFooters.toArray()
            );
        }

        $.each( toolbarsAffectingHeight, function( index, value ) {
            desiredHeight -= $( value ).outerHeight();
        });

        // Height must be at least zero
        return Math.max( 0, desiredHeight );
    };

    $.extend( $.mobile, {
        // define the window and the document objects
        window: $( window ),
        document: $( document ),

        // TODO: Remove and use $.ui.keyCode directly
        keyCode: $.ui.keyCode,

        // Place to store various widget extensions
        behaviors: {},

        // Scroll page vertically: scroll to 0 to hide iOS address bar, or pass a Y value
        silentScroll: function( ypos ) {
            if ( $.type( ypos ) !== "number" ) {
                ypos = $.mobile.defaultHomeScroll;
            }

            // prevent scrollstart and scrollstop events
            $.event.special.scrollstart.enabled = false;

            setTimeout(function() {
                window.scrollTo( 0, ypos );
                $.mobile.document.trigger( "silentscroll", { x: 0, y: ypos });
            }, 20 );

            setTimeout(function() {
                $.event.special.scrollstart.enabled = true;
            }, 150 );
        },

        getClosestBaseUrl: function( ele )  {
```

```
            // Find the closest page and extract out its url.
            var url = $( ele ).closest( ".ui-page" ).jqmData( "url" ),
                base = $.mobile.path.documentBase.hrefNoHash;

            if ( !$.mobile.dynamicBaseEnabled || !url || !$.mobile.path.isPath( url ) ) {
                url = base;
            }

            return $.mobile.path.makeUrlAbsolute( url, base );
        },
        removeActiveLinkClass: function( forceRemoval ) {
            if ( !!$.mobile.activeClickedLink &&
                ( !$.mobile.activeClickedLink.closest( "." + $.mobile.activePageClass ).length ||
                    forceRemoval ) ) {

                $.mobile.activeClickedLink.removeClass( $.mobile.activeBtnClass );
            }
            $.mobile.activeClickedLink = null;
        },

        // DEPRECATED in 1.4
        // Find the closest parent with a theme class on it. Note that
        // we are not using $.fn.closest() on purpose here because this
        // method gets called quite a bit and we need it to be as fast
        // as possible.
        getInheritedTheme: function( el, defaultTheme ) {
            var e = el[ 0 ],
                ltr = "",
                re = /ui-(bar|body|overlay)-([a-z])\b/,
                c, m;
            while ( e ) {
                c = e.className || "";
                if ( c && ( m = re.exec( c ) ) && ( ltr = m[ 2 ] ) ) {
                    // We found a parent with a theme class
                    // on it so bail from this loop.
                    break;
                }

                e = e.parentNode;
            }
            // Return the theme letter we found, if none, return the
            // specified default.
            return ltr || defaultTheme || "a";
        },

        enhanceable: function( elements ) {
            return this.haveParents( elements, "enhance" );
        },

        hijackable: function( elements ) {
            return this.haveParents( elements, "ajax" );
        },

        haveParents: function( elements, attr ) {
```

```
            if ( !$.mobile.ignoreContentEnabled ) {
                return elements;
            }

            var count = elements.length,
                $newSet = $(),
                e, $element, excluded,
                i, c;

            for ( i = 0; i < count; i++ ) {
                $element = elements.eq( i );
                excluded = false;
                e = elements[ i ];

                while ( e ) {
                    c = e.getAttribute ? e.getAttribute( "data-" + $.mobile.ns + attr ) : "";

                    if ( c === "false" ) {
                        excluded = true;
                        break;
                    }

                    e = e.parentNode;
                }

                if ( !excluded ) {
                    $newSet = $newSet.add( $element );
                }
            }

            return $newSet;
        },

        getScreenHeight: function() {
            // Native innerHeight returns more accurate value for this across platforms,
            // jQuery version is here as a normalized fallback for platforms like Symbian
            return window.innerHeight || $.mobile.window.height();
        },

        //simply set the active page's minimum height to screen height, depending on orientation
        resetActivePageHeight: function( height ) {
            var page = $( "." + $.mobile.activePageClass ),
                pageHeight = page.height(),
                pageOuterHeight = page.outerHeight( true );

            height = compensateToolbars( page,
                ( typeof height === "number" ) ? height : $.mobile.getScreenHeight() );

            page.css( "min-height", height - ( pageOuterHeight - pageHeight ) );
        },

        loading: function() {
            // If this is the first call to this function, instantiate a loader widget
            var loader = this.loading._widget || $( $.mobile.loader.prototype.defaultHtml ).
```

```
            loader(),

                // Call the appropriate method on the loader
                returnValue = loader.loader.apply( loader, arguments );

            // Make sure the loader is retained for future calls to this function.
            this.loading._widget = loader;

            return returnValue;
        }
    });

    $.addDependents = function( elem, newDependents ) {
        var $elem = $( elem ),
            dependents = $elem.jqmData( "dependents" ) || $();

        $elem.jqmData( "dependents", $( dependents ).add( newDependents ) );
    };

    // plugins
    $.fn.extend({
        removeWithDependents: function() {
            $.removeWithDependents( this );
        },

        // Enhance child elements
        enhanceWithin: function() {
            var index,
                widgetElements = {},
                keepNative = $.mobile.page.prototype.keepNativeSelector(),
                that = this;

            // Add no js class to elements
            if ( $.mobile.nojs ) {
                $.mobile.nojs( this );
            }

            // Bind links for ajax nav
            if ( $.mobile.links ) {
                $.mobile.links( this );
            }

            // Degrade inputs for styleing
            if ( $.mobile.degradeInputsWithin ) {
                $.mobile.degradeInputsWithin( this );
            }

            // Run buttonmarkup
            if ( $.fn.buttonMarkup ) {
                this.find( $.fn.buttonMarkup.initSelector ).not( keepNative )
                .jqmEnhanceable().buttonMarkup();
            }

            // Add classes for fieldContain
```

```
            if ( $.fn.fieldcontain ) {
                this.find( ":jqmData(role='fieldcontain')" ).not( keepNative )
                .jqmEnhanceable().fieldcontain();
            }

            // Enhance widgets
            $.each( $.mobile.widgets, function( name, constructor ) {

                // If initSelector not false find elements
                if ( constructor.initSelector ) {

                    // Filter elements that should not be enhanced based on parents
                    var elements = $.mobile.enhanceable( that.find( constructor.initSelector ) );

                    // If any matching elements remain filter ones with keepNativeSelector
                    if ( elements.length > 0 ) {

                        // $.mobile.page.prototype.keepNativeSelector is deprecated this is
                        just for backcompat
                        // Switch to $.mobile.keepNative in 1.5 which is just a value not a
                        function
                        elements = elements.not( keepNative );
                    }

                    // Enhance whatever is left
                    if ( elements.length > 0 ) {
                        widgetElements[ constructor.prototype.widgetName ] = elements;
                    }
                }
            });

            for ( index in widgetElements ) {
                widgetElements[ index ][ index ]();
            }

            return this;
        },

        addDependents: function( newDependents ) {
            $.addDependents( this, newDependents );
        },

        // note that this helper doesn't attempt to handle the callback
        // or setting of an html element's text, its only purpose is
        // to return the html encoded version of the text in all cases. (thus the name)
        getEncodedText: function() {
            return $( "<a>" ).text( this.text() ).html();
        },

        // fluent helper function for the mobile namespaced equivalent
        jqmEnhanceable: function() {
            return $.mobile.enhanceable( this );
        },
```

```
        jqmHijackable: function() {
            return $.mobile.hijackable( this );
        }
    });

    $.removeWithDependents = function( nativeElement ) {
        var element = $( nativeElement );

        ( element.jqmData( "dependents" ) || $() ).remove();
        element.remove();
    };
    $.addDependents = function( nativeElement, newDependents ) {
        var element = $( nativeElement ),
            dependents = element.jqmData( "dependents" ) || $();

        element.jqmData( "dependents", $( dependents ).add( newDependents ) );
    };

    $.find.matches = function( expr, set ) {
        return $.find( expr, null, null, set );
    };

    $.find.matchesSelector = function( node, expr ) {
        return $.find( expr, null, null, [ node ] ).length > 0;
    };

})( jQuery, this );


/*!
 * jQuery UI Widget c0ab71056b936627e8a7821f03c044aec6280a40
 * http://jqueryui.com
 *
 * Copyright 2013 jQuery Foundation and other contributors
 * Released under the MIT license.
 * http://jquery.org/license
 *
 * http://api.jqueryui.com/jQuery.widget/
 */
(function( $, undefined ) {

var uuid = 0,
    slice = Array.prototype.slice,
    _cleanData = $.cleanData;
$.cleanData = function( elems ) {
    for ( var i = 0, elem; (elem = elems[i]) != null; i++ ) {
        try {
            $( elem ).triggerHandler( "remove" );
        // http://bugs.jquery.com/ticket/8235
        } catch( e ) {}
    }
    _cleanData( elems );
};
```

```
$.widget = function( name, base, prototype ) {
    var fullName, existingConstructor, constructor, basePrototype,
        // proxiedPrototype allows the provided prototype to remain unmodified
        // so that it can be used as a mixin for multiple widgets (#8876)
        proxiedPrototype = {},
        namespace = name.split( "." )[ 0 ];

    name = name.split( "." )[ 1 ];
    fullName = namespace + "-" + name;

    if ( !prototype ) {
        prototype = base;
        base = $.Widget;
    }

    // create selector for plugin
    $.expr[ ":" ][ fullName.toLowerCase() ] = function( elem ) {
        return !!$.data( elem, fullName );
    };

    $[ namespace ] = $[ namespace ] || {};
    existingConstructor = $[ namespace ][ name ];
    constructor = $[ namespace ][ name ] = function( options, element ) {
        // allow instantiation without "new" keyword
        if ( !this._createWidget ) {
            return new constructor( options, element );
        }

        // allow instantiation without initializing for simple inheritance
        // must use "new" keyword (the code above always passes args)
        if ( arguments.length ) {
            this._createWidget( options, element );
        }
    };
    // extend with the existing constructor to carry over any static properties
    $.extend( constructor, existingConstructor, {
        version: prototype.version,
        // copy the object used to create the prototype in case we need to
        // redefine the widget later
        _proto: $.extend( {}, prototype ),
        // track widgets that inherit from this widget in case this widget is
        // redefined after a widget inherits from it
        _childConstructors: []
    });

    basePrototype = new base();
    // we need to make the options hash a property directly on the new instance
    // otherwise we'll modify the options hash on the prototype that we're
    // inheriting from
    basePrototype.options = $.widget.extend( {}, basePrototype.options );
    $.each( prototype, function( prop, value ) {
        if ( !$.isFunction( value ) ) {
            proxiedPrototype[ prop ] = value;
            return;
```

```
        }
        proxiedPrototype[ prop ] = (function() {
            var _super = function() {
                    return base.prototype[ prop ].apply( this, arguments );
                },
                _superApply = function( args ) {
                    return base.prototype[ prop ].apply( this, args );
                };
            return function() {
                var __super = this._super,
                    __superApply = this._superApply,
                    returnValue;

                this._super = _super;
                this._superApply = _superApply;

                returnValue = value.apply( this, arguments );

                this._super = __super;
                this._superApply = __superApply;

                return returnValue;
            };
        })();
    });
    constructor.prototype = $.widget.extend( basePrototype, {
        // TODO: remove support for widgetEventPrefix
        // always use the name + a colon as the prefix, e.g., draggable:start
        // don't prefix for widgets that aren't DOM-based
        widgetEventPrefix: existingConstructor ? (basePrototype.widgetEventPrefix || name) : name
    }, proxiedPrototype, {
        constructor: constructor,
        namespace: namespace,
        widgetName: name,
        widgetFullName: fullName
    });

    // If this widget is being redefined then we need to find all widgets that
    // are inheriting from it and redefine all of them so that they inherit from
    // the new version of this widget. We're essentially trying to replace one
    // level in the prototype chain.
    if ( existingConstructor ) {
        $.each( existingConstructor._childConstructors, function( i, child ) {
            var childPrototype = child.prototype;

            // redefine the child widget using the same prototype that was
            // originally used, but inherit from the new version of the base
            $.widget( childPrototype.namespace + "." + childPrototype.widgetName, constructor,
            child._proto );
        });
        // remove the list of existing child constructors from the old constructor
        // so the old child constructors can be garbage collected
        delete existingConstructor._childConstructors;
    } else {
```

```
        base._childConstructors.push( constructor );
    }

    $.widget.bridge( name, constructor );

    return constructor;
};

$.widget.extend = function( target ) {
    var input = slice.call( arguments, 1 ),
        inputIndex = 0,
        inputLength = input.length,
        key,
        value;
    for ( ; inputIndex < inputLength; inputIndex++ ) {
        for ( key in input[ inputIndex ] ) {
            value = input[ inputIndex ][ key ];
            if ( input[ inputIndex ].hasOwnProperty( key ) && value !== undefined ) {
                // Clone objects
                if ( $.isPlainObject( value ) ) {
                    target[ key ] = $.isPlainObject( target[ key ] ) ?
                        $.widget.extend( {}, target[ key ], value ) :
                        // Don't extend strings, arrays, etc. with objects
                        $.widget.extend( {}, value );
                // Copy everything else by reference
                } else {
                    target[ key ] = value;
                }
            }
        }
    }
    return target;
};

$.widget.bridge = function( name, object ) {
    var fullName = object.prototype.widgetFullName || name;
    $.fn[ name ] = function( options ) {
        var isMethodCall = typeof options === "string",
            args = slice.call( arguments, 1 ),
            returnValue = this;

        // allow multiple hashes to be passed on init
        options = !isMethodCall && args.length ?
            $.widget.extend.apply( null, [ options ].concat(args) ) :
            options;

        if ( isMethodCall ) {
            this.each(function() {
                var methodValue,
                    instance = $.data( this, fullName );
                if ( options === "instance" ) {
                    returnValue = instance;
                    return false;
                }
```

```
                if ( !instance ) {
                    return $.error( "cannot call methods on " + name + " prior to
                    initialization; " +
                        "attempted to call method '" + options + "'" );
                }
                if ( !$.isFunction( instance[options] ) || options.charAt( 0 ) === "_" ) {
                    return $.error( "no such method '" + options + "' for " + name + " widget
                    instance" );
                }
                methodValue = instance[ options ].apply( instance, args );
                if ( methodValue !== instance && methodValue !== undefined ) {
                    returnValue = methodValue && methodValue.jquery ?
                        returnValue.pushStack( methodValue.get() ) :
                        methodValue;
                    return false;
                }
            });
        } else {
            this.each(function() {
                var instance = $.data( this, fullName );
                if ( instance ) {
                    instance.option( options || {} )._init();
                } else {
                    $.data( this, fullName, new object( options, this ) );
                }
            });
        }

        return returnValue;
    };
};

$.Widget = function( /* options, element */ ) {};
$.Widget._childConstructors = [];

$.Widget.prototype = {
    widgetName: "widget",
    widgetEventPrefix: "",
    defaultElement: "<div>",
    options: {
        disabled: false,

        // callbacks
        create: null
    },
    _createWidget: function( options, element ) {
        element = $( element || this.defaultElement || this )[ 0 ];
        this.element = $( element );
        this.uuid = uuid++;
        this.eventNamespace = "." + this.widgetName + this.uuid;
        this.options = $.widget.extend( {},
            this.options,
            this._getCreateOptions(),
            options );
```

```
        this.bindings = $();
        this.hoverable = $();
        this.focusable = $();

        if ( element !== this ) {
            $.data( element, this.widgetFullName, this );
            this._on( true, this.element, {
                remove: function( event ) {
                    if ( event.target === element ) {
                        this.destroy();
                    }
                }
            });
            this.document = $( element.style ?
                // element within the document
                element.ownerDocument :
                // element is window or document
                element.document || element );
            this.window = $( this.document[0].defaultView || this.document[0].parentWindow );
        }

        this._create();
        this._trigger( "create", null, this._getCreateEventData() );
        this._init();
    },
    _getCreateOptions: $.noop,
    _getCreateEventData: $.noop,
    _create: $.noop,
    _init: $.noop,

    destroy: function() {
        this._destroy();
        // we can probably remove the unbind calls in 2.0
        // all event bindings should go through this._on()
        this.element
            .unbind( this.eventNamespace )
            .removeData( this.widgetFullName )
            // support: jquery <1.6.3
            // http://bugs.jquery.com/ticket/9413
            .removeData( $.camelCase( this.widgetFullName ) );
        this.widget()
            .unbind( this.eventNamespace )
            .removeAttr( "aria-disabled" )
            .removeClass(
                this.widgetFullName + "-disabled " +
                "ui-state-disabled" );

        // clean up events and states
        this.bindings.unbind( this.eventNamespace );
        this.hoverable.removeClass( "ui-state-hover" );
        this.focusable.removeClass( "ui-state-focus" );
    },
    _destroy: $.noop,
```

```
	widget: function() {
		return this.element;
	},

	option: function( key, value ) {
		var options = key,
			parts,
			curOption,
			i;

		if ( arguments.length === 0 ) {
			// don't return a reference to the internal hash
			return $.widget.extend( {}, this.options );
		}

		if ( typeof key === "string" ) {
			// handle nested keys, e.g., "foo.bar" => { foo: { bar: ___ } }
			options = {};
			parts = key.split( "." );
			key = parts.shift();
			if ( parts.length ) {
				curOption = options[ key ] = $.widget.extend( {}, this.options[ key ] );
				for ( i = 0; i < parts.length - 1; i++ ) {
					curOption[ parts[ i ] ] = curOption[ parts[ i ] ] || {};
					curOption = curOption[ parts[ i ] ];
				}
				key = parts.pop();
				if ( value === undefined ) {
					return curOption[ key ] === undefined ? null : curOption[ key ];
				}
				curOption[ key ] = value;
			} else {
				if ( value === undefined ) {
					return this.options[ key ] === undefined ? null : this.options[ key ];
				}
				options[ key ] = value;
			}
		}

		this._setOptions( options );

		return this;
	},
	_setOptions: function( options ) {
		var key;

		for ( key in options ) {
			this._setOption( key, options[ key ] );
		}

		return this;
	},
	_setOption: function( key, value ) {
```

```
        this.options[ key ] = value;

        if ( key === "disabled" ) {
            this.widget()
                .toggleClass( this.widgetFullName + "-disabled", !!value );
            this.hoverable.removeClass( "ui-state-hover" );
            this.focusable.removeClass( "ui-state-focus" );
        }

        return this;
    },

    enable: function() {
        return this._setOptions({ disabled: false });
    },
    disable: function() {
        return this._setOptions({ disabled: true });
    },

    _on: function( suppressDisabledCheck, element, handlers ) {
        var delegateElement,
            instance = this;

        // no suppressDisabledCheck flag, shuffle arguments
        if ( typeof suppressDisabledCheck !== "boolean" ) {
            handlers = element;
            element = suppressDisabledCheck;
            suppressDisabledCheck = false;
        }

        // no element argument, shuffle and use this.element
        if ( !handlers ) {
            handlers = element;
            element = this.element;
            delegateElement = this.widget();
        } else {
            // accept selectors, DOM elements
            element = delegateElement = $( element );
            this.bindings = this.bindings.add( element );
        }

        $.each( handlers, function( event, handler ) {
            function handlerProxy() {
                // allow widgets to customize the disabled handling
                // - disabled as an array instead of boolean
                // - disabled class as method for disabling individual parts
                if ( !suppressDisabledCheck &&
                        ( instance.options.disabled === true ||
                            $( this ).hasClass( "ui-state-disabled" ) ) ) {
                    return;
                }
                return ( typeof handler === "string" ? instance[ handler ] : handler )
                    .apply( instance, arguments );
            }
```

```
            // copy the guid so direct unbinding works
            if ( typeof handler !== "string" ) {
                handlerProxy.guid = handler.guid =
                    handler.guid || handlerProxy.guid || $.guid++;
            }

            var match = event.match( /^(\w+)\s*(.*)$/ ),
                eventName = match[1] + instance.eventNamespace,
                selector = match[2];
            if ( selector ) {
                delegateElement.delegate( selector, eventName, handlerProxy );
            } else {
                element.bind( eventName, handlerProxy );
            }
        });
    },

    _off: function( element, eventName ) {
        eventName = (eventName || "").split( " " ).join( this.eventNamespace + " " ) + this.
        eventNamespace;
        element.unbind( eventName ).undelegate( eventName );
    },

    _delay: function( handler, delay ) {
        function handlerProxy() {
            return ( typeof handler === "string" ? instance[ handler ] : handler )
                .apply( instance, arguments );
        }
        var instance = this;
        return setTimeout( handlerProxy, delay || 0 );
    },

    _hoverable: function( element ) {
        this.hoverable = this.hoverable.add( element );
        this._on( element, {
            mouseenter: function( event ) {
                $( event.currentTarget ).addClass( "ui-state-hover" );
            },
            mouseleave: function( event ) {
                $( event.currentTarget ).removeClass( "ui-state-hover" );
            }
        });
    },

    _focusable: function( element ) {
        this.focusable = this.focusable.add( element );
        this._on( element, {
            focusin: function( event ) {
                $( event.currentTarget ).addClass( "ui-state-focus" );
            },
            focusout: function( event ) {
                $( event.currentTarget ).removeClass( "ui-state-focus" );
            }
```

```
        });
    },

    _trigger: function( type, event, data ) {
        var prop, orig,
            callback = this.options[ type ];

        data = data || {};
        event = $.Event( event );
        event.type = ( type === this.widgetEventPrefix ?
            type :
            this.widgetEventPrefix + type ).toLowerCase();
        // the original event may come from any element
        // so we need to reset the target on the new event
        event.target = this.element[ 0 ];

        // copy original event properties over to the new event
        orig = event.originalEvent;
        if ( orig ) {
            for ( prop in orig ) {
                if ( !( prop in event ) ) {
                    event[ prop ] = orig[ prop ];
                }
            }
        }

        this.element.trigger( event, data );
        return !( $.isFunction( callback ) &&
            callback.apply( this.element[0], [ event ].concat( data ) ) === false ||
            event.isDefaultPrevented() );
    }
};

$.each( { show: "fadeIn", hide: "fadeOut" }, function( method, defaultEffect ) {
    $.Widget.prototype[ "_" + method ] = function( element, options, callback ) {
        if ( typeof options === "string" ) {
            options = { effect: options };
        }
        var hasOptions,
            effectName = !options ?
                method :
                options === true || typeof options === "number" ?
                    defaultEffect :
                    options.effect || defaultEffect;
        options = options || {};
        if ( typeof options === "number" ) {
            options = { duration: options };
        }
        hasOptions = !$.isEmptyObject( options );
        options.complete = callback;
        if ( options.delay ) {
            element.delay( options.delay );
        }
        if ( hasOptions && $.effects && $.effects.effect[ effectName ] ) {
```

```
            element[ method ]( options );
        } else if ( effectName !== method && element[ effectName ] ) {
            element[ effectName ]( options.duration, options.easing, callback );
        } else {
            element.queue(function( next ) {
                $( this )[ method ]();
                if ( callback ) {
                    callback.call( element[ 0 ] );
                }
                next();
            });
        }
    };
});

})( jQuery );

(function( $, undefined ) {

var rcapitals = /[A-Z]/g,
    replaceFunction = function( c ) {
        return "-" + c.toLowerCase();
    };

$.extend( $.Widget.prototype, {
    _getCreateOptions: function() {
        var option, value,
            elem = this.element[ 0 ],
            options = {};

        //
        if ( !$.mobile.getAttribute( elem, "defaults" ) ) {
            for ( option in this.options ) {
                value = $.mobile.getAttribute( elem, option.replace( rcapitals, replaceFunction
                ) );

                if ( value != null ) {
                    options[ option ] = value;
                }
            }
        }

        return options;
    }
});

//TODO: Remove in 1.5 for backcompat only
$.mobile.widget = $.Widget;

})( jQuery );


(function( $ ) {
    // TODO move loader class down into the widget settings
```

```
    var loaderClass = "ui-loader", $html = $( "html" );

    $.widget( "mobile.loader", {
        // NOTE if the global config settings are defined they will override these
        //      options
        options: {
            // the theme for the loading message
            theme: "a",

            // whether the text in the loading message is shown
            textVisible: false,

            // custom html for the inner content of the loading message
            html: "",

            // the text to be displayed when the popup is shown
            text: "loading"
        },

        defaultHtml: "<div class='" + loaderClass + "'>" +
            "<span class='ui-icon-loading'></span>" +
            "<h1></h1>" +
            "</div>",

        // For non-fixed supportin browsers. Position at y center (if scrollTop supported),
        above the activeBtn (if defined), or just 100px from top
        fakeFixLoader: function() {
            var activeBtn = $( "." + $.mobile.activeBtnClass ).first();

            this.element
                .css({
                    top: $.support.scrollTop && this.window.scrollTop() + this.window.height() /
                     2 ||
                        activeBtn.length && activeBtn.offset().top || 100
                });
        },

        // check position of loader to see if it appears to be "fixed" to center
        // if not, use abs positioning
        checkLoaderPosition: function() {
            var offset = this.element.offset(),
                scrollTop = this.window.scrollTop(),
                screenHeight = $.mobile.getScreenHeight();

            if ( offset.top < scrollTop || ( offset.top - scrollTop ) > screenHeight ) {
                this.element.addClass( "ui-loader-fakefix" );
                this.fakeFixLoader();
                this.window
                    .unbind( "scroll", this.checkLoaderPosition )
                    .bind( "scroll", $.proxy( this.fakeFixLoader, this ) );
            }
        },

        resetHtml: function() {
```

```
            this.element.html( $( this.defaultHtml ).html() );
        },

        // Turn on/off page loading message. Theme doubles as an object argument
        // with the following shape: { theme: '', text: '', html: '', textVisible: '' }
        // NOTE that the $.mobile.loading* settings and params past the first are deprecated
        // TODO sweet jesus we need to break some of this out
        show: function( theme, msgText, textonly ) {
            var textVisible, message, loadSettings;

            this.resetHtml();

            // use the prototype options so that people can set them globally at
            // mobile init. Consistency, it's what's for dinner
            if ( $.type( theme ) === "object" ) {
                loadSettings = $.extend( {}, this.options, theme );

                theme = loadSettings.theme;
            } else {
                loadSettings = this.options;

                // here we prefer the theme value passed as a string argument, then
                // we prefer the global option because we can't use undefined default
                // prototype options, then the prototype option
                theme = theme || loadSettings.theme;
            }

            // set the message text, prefer the param, then the settings object
            // then loading message
            message = msgText || ( loadSettings.text === false ? "" : loadSettings.text );

            // prepare the dom
            $html.addClass( "ui-loading" );

            textVisible = loadSettings.textVisible;

            // add the proper css given the options (theme, text, etc)
            // Force text visibility if the second argument was supplied, or
            // if the text was explicitly set in the object args
            this.element.attr("class", loaderClass +
                " ui-corner-all ui-body-" + theme +
                " ui-loader-" + ( textVisible || msgText || theme.text ? "verbose" : "default" )
                +
                ( loadSettings.textonly || textonly ? " ui-loader-textonly" : "" ) );

            // TODO verify that jquery.fn.html is ok to use in both cases here
            //      this might be overly defensive in preventing unknowing xss
            // if the html attribute is defined on the loading settings, use that
            // otherwise use the fallbacks from above
            if ( loadSettings.html ) {
                this.element.html( loadSettings.html );
            } else {
                this.element.find( "h1" ).text( message );
            }
```

```
            // attach the loader to the DOM
            this.element.appendTo( $.mobile.pageContainer );

            // check that the loader is visible
            this.checkLoaderPosition();

            // on scroll check the loader position
            this.window.bind( "scroll", $.proxy( this.checkLoaderPosition, this ) );
        },

        hide: function() {
            $html.removeClass( "ui-loading" );

            if ( this.options.text ) {
                this.element.removeClass( "ui-loader-fakefix" );
            }

            $.mobile.window.unbind( "scroll", this.fakeFixLoader );
            $.mobile.window.unbind( "scroll", this.checkLoaderPosition );
        }
    });

})(jQuery, this);


/*!
 * jQuery hashchange event - v1.3 - 7/21/2010
 * http://benalman.com/projects/jquery-hashchange-plugin/
 *
 * Copyright (c) 2010 "Cowboy" Ben Alman
 * Dual licensed under the MIT and GPL licenses.
 * http://benalman.com/about/license/
 */

// Script: jQuery hashchange event
//
// *Version: 1.3, Last updated: 7/21/2010*
//
// Project Home - http://benalman.com/projects/jquery-hashchange-plugin/
// GitHub       - http://github.com/cowboy/jquery-hashchange/
// Source       - http://github.com/cowboy/jquery-hashchange/raw/master/jquery.ba-hashchange.js
// (Minified)   -
http://github.com/cowboy/jquery-hashchange/raw/master/jquery.ba-hashchange.min.js (0.8kb gzipped)
//
// About: License
//
// Copyright (c) 2010 "Cowboy" Ben Alman,
// Dual licensed under the MIT and GPL licenses.
// http://benalman.com/about/license/
//
// About: Examples
//
// These working examples, complete with fully commented code, illustrate a few
```

```
// ways in which this plugin can be used.
//
// hashchange event - http://benalman.com/code/projects/jquery-hashchange/examples/hashchange/
// document.domain -
http://benalman.com/code/projects/jquery-hashchange/examples/document_domain/
//
// About: Support and Testing
//
// Information about what version or versions of jQuery this plugin has been
// tested with, what browsers it has been tested in, and where the unit tests
// reside (so you can test it yourself).
//
// jQuery Versions - 1.2.6, 1.3.2, 1.4.1, 1.4.2
// Browsers Tested - Internet Explorer 6-8, Firefox 2-4, Chrome 5-6, Safari 3.2-5,
//                   Opera 9.6-10.60, iPhone 3.1, Android 1.6-2.2, BlackBerry 4.6-5.
// Unit Tests      - http://benalman.com/code/projects/jquery-hashchange/unit/
//
// About: Known issues
//
// While this jQuery hashchange event implementation is quite stable and
// robust, there are a few unfortunate browser bugs surrounding expected
// hashchange event-based behaviors, independent of any JavaScript
// window.onhashchange abstraction. See the following examples for more
// information:
//
// Chrome: Back Button -
http://benalman.com/code/projects/jquery-hashchange/examples/bug-chrome-back-button/
// Firefox: Remote XMLHttpRequest -
http://benalman.com/code/projects/jquery-hashchange/examples/bug-firefox-remote-xhr/
// WebKit: Back Button in an Iframe -
http://benalman.com/code/projects/jquery-hashchange/examples/bug-webkit-hash-iframe/
// Safari: Back Button from a different domain -
http://benalman.com/code/projects/jquery-hashchange/examples/bug-safari-back-from-diff-domain/
//
// Also note that should a browser natively support the window.onhashchange
// event, but not report that it does, the fallback polling loop will be used.
//
// About: Release History
//
// 1.3   - (7/21/2010) Reorganized IE6/7 Iframe code to make it more
//         "removable" for mobile-only development. Added IE6/7 document.title
//         support. Attempted to make Iframe as hidden as possible by using
//         techniques from http://www.paciellogroup.com/blog/?p=604. Added
//         support for the "shortcut" format $(window).hashchange( fn ) and
//         $(window).hashchange() like jQuery provides for built-in events.
//         Renamed jQuery.hashchangeDelay to <jQuery.fn.hashchange.delay> and
//         lowered its default value to 50. Added <jQuery.fn.hashchange.domain>
//         and <jQuery.fn.hashchange.src> properties plus document-domain.html
//         file to address access denied issues when setting document.domain in
//         IE6/7.
// 1.2   - (2/11/2010) Fixed a bug where coming back to a page using this plugin
//         from a page on another domain would cause an error in Safari 4. Also,
//         IE6/7 Iframe is now inserted after the body (this actually works),
//         which prevents the page from scrolling when the event is first bound.
```

```
//         Event can also now be bound before DOM ready, but it won't be usable
//         before then in IE6/7.
// 1.1   - (1/21/2010) Incorporated document.documentMode test to fix IE8 bug
//         where browser version is incorrectly reported as 8.0, despite
//         inclusion of the X-UA-Compatible IE=EmulateIE7 meta tag.
// 1.0   - (1/9/2010) Initial Release. Broke out the jQuery BBQ event.special
//         window.onhashchange functionality into a separate plugin for users
//         who want just the basic event & back button support, without all the
//         extra awesomeness that BBQ provides. This plugin will be included as
//         part of jQuery BBQ, but also be available separately.

(function($,window,undefined){
  '$:nomunge'; // Used by YUI compressor.

  // Reused string.
  var str_hashchange = 'hashchange',

    // Method / object references.
    doc = document,
    fake_onhashchange,
    special = $.event.special,

    // Does the browser support window.onhashchange? Note that IE8 running in
    // IE7 compatibility mode reports true for 'onhashchange' in window, even
    // though the event isn't supported, so also test document.documentMode.
    doc_mode = doc.documentMode,
    supports_onhashchange = 'on' + str_hashchange in window && ( doc_mode === undefined ||
    doc_mode > 7 );

  // Get location.hash (or what you'd expect location.hash to be) sans any
  // leading #. Thanks for making this necessary, Firefox!
  function get_fragment( url ) {
    url = url || location.href;
    return '#' + url.replace( /^[^#]*#?(.*)$/, '$1' );
  };

  // Method: jQuery.fn.hashchange
  //
  // Bind a handler to the window.onhashchange event or trigger all bound
  // window.onhashchange event handlers. This behavior is consistent with
  // jQuery's built-in event handlers.
  //
  // Usage:
  //
  // > jQuery(window).hashchange( [ handler ] );
  //
  // Arguments:
  //
  //  handler - (Function) Optional handler to be bound to the hashchange
  //    event. This is a "shortcut" for the more verbose form:
  //    jQuery(window).bind( 'hashchange', handler ). If handler is omitted,
  //    all bound window.onhashchange event handlers will be triggered. This
  //    is a shortcut for the more verbose
  //    jQuery(window).trigger( 'hashchange' ). These forms are described in
```

```
//    the <hashchange event> section.
//
// Returns:
//
//  (jQuery) The initial jQuery collection of elements.

// Allow the "shortcut" format $(elem).hashchange( fn ) for binding and
// $(elem).hashchange() for triggering, like jQuery does for built-in events.
$.fn[ str_hashchange ] = function( fn ) {
  return fn ? this.bind( str_hashchange, fn ) : this.trigger( str_hashchange );
};

// Property: jQuery.fn.hashchange.delay
//
// The numeric interval (in milliseconds) at which the <hashchange event>
// polling loop executes. Defaults to 50.

// Property: jQuery.fn.hashchange.domain
//
// If you're setting document.domain in your JavaScript, and you want hash
// history to work in IE6/7, not only must this property be set, but you must
// also set document.domain BEFORE jQuery is loaded into the page. This
// property is only applicable if you are supporting IE6/7 (or IE8 operating
// in "IE7 compatibility" mode).
//
// In addition, the <jQuery.fn.hashchange.src> property must be set to the
// path of the included "document-domain.html" file, which can be renamed or
// modified if necessary (note that the document.domain specified must be the
// same in both your main JavaScript as well as in this file).
//
// Usage:
//
// jQuery.fn.hashchange.domain = document.domain;

// Property: jQuery.fn.hashchange.src
//
// If, for some reason, you need to specify an Iframe src file (for example,
// when setting document.domain as in <jQuery.fn.hashchange.domain>), you can
// do so using this property. Note that when using this property, history
// won't be recorded in IE6/7 until the Iframe src file loads. This property
// is only applicable if you are supporting IE6/7 (or IE8 operating in "IE7
// compatibility" mode).
//
// Usage:
//
// jQuery.fn.hashchange.src = 'path/to/file.html';

$.fn[ str_hashchange ].delay = 50;
/*
$.fn[ str_hashchange ].domain = null;
$.fn[ str_hashchange ].src = null;
*/

// Event: hashchange event
```

```
//
// Fired when location.hash changes. In browsers that support it, the native
// HTML5 window.onhashchange event is used, otherwise a polling loop is
// initialized, running every <jQuery.fn.hashchange.delay> milliseconds to
// see if the hash has changed. In IE6/7 (and IE8 operating in "IE7
// compatibility" mode), a hidden Iframe is created to allow the back button
// and hash-based history to work.
//
// Usage as described in <jQuery.fn.hashchange>:
//
// > // Bind an event handler.
// > jQuery(window).hashchange( function(e) {
// >   var hash = location.hash;
// >   ...
// > });
// >
// > // Manually trigger the event handler.
// > jQuery(window).hashchange();
//
// A more verbose usage that allows for event namespacing:
//
// > // Bind an event handler.
// > jQuery(window).bind( 'hashchange', function(e) {
// >   var hash = location.hash;
// >   ...
// > });
// >
// > // Manually trigger the event handler.
// > jQuery(window).trigger( 'hashchange' );
//
// Additional Notes:
//
// * The polling loop and Iframe are not created until at least one handler
//   is actually bound to the 'hashchange' event.
// * If you need the bound handler(s) to execute immediately, in cases where
//   a location.hash exists on page load, via bookmark or page refresh for
//   example, use jQuery(window).hashchange() or the more verbose
//   jQuery(window).trigger( 'hashchange' ).
// * The event can be bound before DOM ready, but since it won't be usable
//   before then in IE6/7 (due to the necessary Iframe), recommended usage is
//   to bind it inside a DOM ready handler.

// Override existing $.event.special.hashchange methods (allowing this plugin
// to be defined after jQuery BBQ in BBQ's source code).
special[ str_hashchange ] = $.extend( special[ str_hashchange ], {

  // Called only when the first 'hashchange' event is bound to window.
  setup: function() {
    // If window.onhashchange is supported natively, there's nothing to do..
    if ( supports_onhashchange ) { return false; }

    // Otherwise, we need to create our own. And we don't want to call this
    // until the user binds to the event, just in case they never do, since it
    // will create a polling loop and possibly even a hidden Iframe.
```

```
    $( fake_onhashchange.start );
  },

  // Called only when the last 'hashchange' event is unbound from window.
  teardown: function() {
    // If window.onhashchange is supported natively, there's nothing to do..
    if ( supports_onhashchange ) { return false; }

    // Otherwise, we need to stop ours (if possible).
    $( fake_onhashchange.stop );
  }

});

// fake_onhashchange does all the work of triggering the window.onhashchange
// event for browsers that don't natively support it, including creating a
// polling loop to watch for hash changes and in IE 6/7 creating a hidden
// Iframe to enable back and forward.
fake_onhashchange = (function(){
  var self = {},
    timeout_id,

    // Remember the initial hash so it doesn't get triggered immediately.
    last_hash = get_fragment(),

    fn_retval = function(val){ return val; },
    history_set = fn_retval,
    history_get = fn_retval;

  // Start the polling loop.
  self.start = function() {
    timeout_id || poll();
  };

  // Stop the polling loop.
  self.stop = function() {
    timeout_id && clearTimeout( timeout_id );
    timeout_id = undefined;
  };

  // This polling loop checks every $.fn.hashchange.delay milliseconds to see
  // if location.hash has changed, and triggers the 'hashchange' event on
  // window when necessary.
  function poll() {
    var hash = get_fragment(),
      history_hash = history_get( last_hash );

    if ( hash !== last_hash ) {
      history_set( last_hash = hash, history_hash );

      $(window).trigger( str_hashchange );

    } else if ( history_hash !== last_hash ) {
      location.href = location.href.replace( /#.*/, '' ) + history_hash;
```

```
    }

    timeout_id = setTimeout( poll, $.fn[ str_hashchange ].delay );
  };

  // vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
  // vvvvvvvvvvvvvvvvvv REMOVE IF NOT SUPPORTING IE6/7/8 vvvvvvvvvvvvvvvvvv
  // vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
  window.attachEvent && !window.addEventListener && !supports_onhashchange && (function(){
    // Not only do IE6/7 need the "magical" Iframe treatment, but so does IE8
    // when running in "IE7 compatibility" mode.

    var iframe,
      iframe_src;

    // When the event is bound and polling starts in IE 6/7, create a hidden
    // Iframe for history handling.
    self.start = function(){
      if ( !iframe ) {
        iframe_src = $.fn[ str_hashchange ].src;
        iframe_src = iframe_src && iframe_src + get_fragment();

        // Create hidden Iframe. Attempt to make Iframe as hidden as possible
        // by using techniques from http://www.paciellogroup.com/blog/?p=604.
        iframe = $('<iframe tabindex="-1" title="empty"/>').hide()

          // When Iframe has completely loaded, initialize the history and
          // start polling.
          .one( 'load', function(){
            iframe_src || history_set( get_fragment() );
            poll();
          })

          // Load Iframe src if specified, otherwise nothing.
          .attr( 'src', iframe_src || 'javascript:0' )

          // Append Iframe after the end of the body to prevent unnecessary
          // initial page scrolling (yes, this works).
          .insertAfter( 'body' )[0].contentWindow;

        // Whenever `document.title` changes, update the Iframe's title to
        // prettify the back/next history menu entries. Since IE sometimes
        // errors with "Unspecified error" the very first time this is set
        // (yes, very useful) wrap this with a try/catch block.
        doc.onpropertychange = function(){
          try {
            if ( event.propertyName === 'title' ) {
              iframe.document.title = doc.title;
            }
          } catch(e) {}
        };

      }
    };
```

```
      // Override the "stop" method since an IE6/7 Iframe was created. Even
      // if there are no longer any bound event handlers, the polling loop
      // is still necessary for back/next to work at all!
      self.stop = fn_retval;

      // Get history by looking at the hidden Iframe's location.hash.
      history_get = function() {
        return get_fragment( iframe.location.href );
      };

      // Set a new history item by opening and then closing the Iframe
      // document, *then* setting its location.hash. If document.domain has
      // been set, update that as well.
      history_set = function( hash, history_hash ) {
        var iframe_doc = iframe.document,
          domain = $.fn[ str_hashchange ].domain;

        if ( hash !== history_hash ) {
          // Update Iframe with any initial `document.title` that might be set.
          iframe_doc.title = doc.title;

          // Opening the Iframe's document after it has been closed is what
          // actually adds a history entry.
          iframe_doc.open();

          // Set document.domain for the Iframe document as well, if necessary.
          domain && iframe_doc.write( '<script>document.domain="' + domain + '"</script>' );

          iframe_doc.close();

          // Update the Iframe's hash, for great justice.
          iframe.location.hash = hash;
        }
      };

    })();
    // ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    // ^^^^^^^^^^^^^^^^^^^ REMOVE IF NOT SUPPORTING IE6/7/8 ^^^^^^^^^^^^^^^^^^^
    // ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

    return self;
  })();

})(jQuery,this);

(function( $, undefined ) {

    /*! matchMedia() polyfill - Test a CSS media type/query in JS. Authors & copyright (c)
    2012: Scott Jehl, Paul Irish, Nicholas Zakas. Dual MIT/BSD license */
    window.matchMedia = window.matchMedia || (function( doc, undefined ) {

        var bool,
            docElem = doc.documentElement,
```

```
            refNode = docElem.firstElementChild || docElem.firstChild,
            // fakeBody required for <FF4 when executed in <head>
            fakeBody = doc.createElement( "body" ),
            div = doc.createElement( "div" );

        div.id = "mq-test-1";
        div.style.cssText = "position:absolute;top:-100em";
        fakeBody.style.background = "none";
        fakeBody.appendChild(div);

        return function(q){

            div.innerHTML = "&shy;<style media=\"" + q + "\"> #mq-test-1 { width: 42px;
            }</style>";

            docElem.insertBefore( fakeBody, refNode );
            bool = div.offsetWidth === 42;
            docElem.removeChild( fakeBody );

            return {
                matches: bool,
                media: q
            };

        };

    }( document ));

    // $.mobile.media uses matchMedia to return a boolean.
    $.mobile.media = function( q ) {
        return window.matchMedia( q ).matches;
    };

})(jQuery);

    (function( $, undefined ) {
        var support = {
            touch: "ontouchend" in document
        };

        $.mobile.support = $.mobile.support || {};
        $.extend( $.support, support );
        $.extend( $.mobile.support, support );
    }( jQuery ));

    (function( $, undefined ) {
        $.extend( $.support, {
            orientation: "orientation" in window && "onorientationchange" in window
        });
    }( jQuery ));

(function( $, undefined ) {

// thx Modernizr
```

```
function propExists( prop ) {
    var uc_prop = prop.charAt( 0 ).toUpperCase() + prop.substr( 1 ),
        props = ( prop + " " + vendors.join( uc_prop + " " ) + uc_prop ).split( " " ),
        v;

    for ( v in props ) {
        if ( fbCSS[ props[ v ] ] !== undefined ) {
            return true;
        }
    }
}

var fakeBody = $( "<body>" ).prependTo( "html" ),
    fbCSS = fakeBody[ 0 ].style,
    vendors = [ "Webkit", "Moz", "O" ],
    webos = "palmGetResource" in window, //only used to rule out scrollTop
    operamini = window.operamini && ({}).toString.call( window.operamini ) === "[object
    OperaMini]",
    bb = window.blackberry && !propExists( "-webkit-transform" ), //only used to rule out box
    shadow, as it's filled opaque on BB 5 and lower
    nokiaLTE7_3;

// inline SVG support test
function inlineSVG() {
    // Thanks Modernizr & Erik Dahlstrom
    var w = window,
        svg = !!w.document.createElementNS && !!w.document.createElementNS(
        "http://www.w3.org/2000/svg", "svg" ).createSVGRect && !( w.opera && navigator.userAgent
        .indexOf( "Chrome" ) === -1 ),
        support = function( data ) {
            if ( !( data && svg ) ) {
                $( "html" ).addClass( "ui-nosvg" );
            }
        },
        img = new w.Image();

    img.onerror = function() {
        support( false );
    };
    img.onload = function() {
        support( img.width === 1 && img.height === 1 );
    };
    img.src = "";
}

function transform3dTest() {
    var mqProp = "transform-3d",
        // Because the `translate3d` test below throws false positives in Android:
        ret = $.mobile.media( "(-" + vendors.join( "-" + mqProp + "),(-" ) + "-" + mqProp +
        "),(" + mqProp + ")" ),
        el, transforms, t;

    if ( ret ) {
        return !!ret;
```

```
    }

    el = document.createElement( "div" );
    transforms = {
        // We're omitting Opera for the time being; MS uses unprefixed.
        "MozTransform": "-moz-transform",
        "transform": "transform"
    };

    fakeBody.append( el );

    for ( t in transforms ) {
        if ( el.style[ t ] !== undefined ) {
            el.style[ t ] = "translate3d( 100px, 1px, 1px )";
            ret = window.getComputedStyle( el ).getPropertyValue( transforms[ t ] );
        }
    }
    return ( !!ret && ret !== "none" );
}

// Test for dynamic-updating base tag support ( allows us to avoid href,src attr rewriting )
function baseTagTest() {
    var fauxBase = location.protocol + "//" + location.host + location.pathname + "ui-dir/",
        base = $( "head base" ),
        fauxEle = null,
        href = "",
        link, rebase;

    if ( !base.length ) {
        base = fauxEle = $( "<base>", { "href": fauxBase }).appendTo( "head" );
    } else {
        href = base.attr( "href" );
    }

    link = $( "<a href='testurl' />" ).prependTo( fakeBody );
    rebase = link[ 0 ].href;
    base[ 0 ].href = href || location.pathname;

    if ( fauxEle ) {
        fauxEle.remove();
    }
    return rebase.indexOf( fauxBase ) === 0;
}

// Thanks Modernizr
function cssPointerEventsTest() {
    var element = document.createElement( "x" ),
        documentElement = document.documentElement,
        getComputedStyle = window.getComputedStyle,
        supports;

    if ( !( "pointerEvents" in element.style ) ) {
        return false;
    }
```

```
    element.style.pointerEvents = "auto";
    element.style.pointerEvents = "x";
    documentElement.appendChild( element );
    supports = getComputedStyle &&
    getComputedStyle( element, "" ).pointerEvents === "auto";
    documentElement.removeChild( element );
    return !!supports;
}

function boundingRect() {
    var div = document.createElement( "div" );
    return typeof div.getBoundingClientRect !== "undefined";
}

// non-UA-based IE version check by James Padolsey, modified by jdalton - from
http://gist.github.com/527683
// allows for inclusion of IE 6+, including Windows Mobile 7
$.extend( $.mobile, { browser: {} } );
$.mobile.browser.oldIE = (function() {
    var v = 3,
        div = document.createElement( "div" ),
        a = div.all || [];

    do {
        div.innerHTML = "<!--[if gt IE " + ( ++v ) + "]><br><![endif]-->";
    } while( a[0] );

    return v > 4 ? v : !v;
})();

function fixedPosition() {
    var w = window,
        ua = navigator.userAgent,
        platform = navigator.platform,
        // Rendering engine is Webkit, and capture major version
        wkmatch = ua.match( /AppleWebKit\/([0-9]+)/ ),
        wkversion = !!wkmatch && wkmatch[ 1 ],
        ffmatch = ua.match( /Fennec\/([0-9]+)/ ),
        ffversion = !!ffmatch && ffmatch[ 1 ],
        operammobilematch = ua.match( /Opera Mobi\/([0-9]+)/ ),
        omversion = !!operammobilematch && operammobilematch[ 1 ];

    if (
        // iOS 4.3 and older : Platform is iPhone/Pad/Touch and Webkit version is less than 534
        (ios5)
        ( ( platform.indexOf( "iPhone" ) > -1 || platform.indexOf( "iPad" ) > -1  || platform.
        indexOf( "iPod" ) > -1 ) && wkversion && wkversion < 534 ) ||
        // Opera Mini
        ( w.operamini && ({}).toString.call( w.operamini ) === "[object OperaMini]" ) ||
        ( operammobilematch && omversion < 7458 )   ||
        //Android lte 2.1: Platform is Android and Webkit version is less than 533 (Android 2.2)
        ( ua.indexOf( "Android" ) > -1 && wkversion && wkversion < 533 ) ||
        // Firefox Mobile before 6.0 -
```

```
        ( ffversion && ffversion < 6 ) ||
        // WebOS less than 3
        ( "palmGetResource" in window && wkversion && wkversion < 534 ) ||
        // MeeGo
        ( ua.indexOf( "MeeGo" ) > -1 && ua.indexOf( "NokiaBrowser/8.5.0" ) > -1 ) ) {
        return false;
    }

    return true;
}

$.extend( $.support, {
    // Note, Chrome for iOS has an extremely quirky implementation of popstate.
    // We've chosen to take the shortest path to a bug fix here for issue #5426
    // See the following link for information about the regex chosen
    // https://developers.google.com/chrome/mobile/docs/user-agent#chrome_for_ios_user-agent
    pushState: "pushState" in history &&
        "replaceState" in history &&
        // When running inside a FF iframe, calling replaceState causes an error
        !( window.navigator.userAgent.indexOf( "Firefox" ) >= 0 && window.top !== window ) &&
        ( window.navigator.userAgent.search(/CriOS/) === -1 ),

    mediaquery: $.mobile.media( "only all" ),
    cssPseudoElement: !!propExists( "content" ),
    touchOverflow: !!propExists( "overflowScrolling" ),
    cssTransform3d: transform3dTest(),
    boxShadow: !!propExists( "boxShadow" ) && !bb,
    fixedPosition: fixedPosition(),
    scrollTop: ("pageXOffset" in window ||
        "scrollTop" in document.documentElement ||
        "scrollTop" in fakeBody[ 0 ]) && !webos && !operamini,

    dynamicBaseTag: baseTagTest(),
    cssPointerEvents: cssPointerEventsTest(),
    boundingRect: boundingRect(),
    inlineSVG: inlineSVG
});

fakeBody.remove();

// $.mobile.ajaxBlacklist is used to override ajaxEnabled on platforms that have known
conflicts with hash history updates (BB5, Symbian)
// or that generally work better browsing in regular http for full page refreshes (Opera Mini)
// Note: This detection below is used as a last resort.
// We recommend only using these detection methods when all other more reliable/forward-looking
approaches are not possible
nokiaLTE7_3 = (function() {

    var ua = window.navigator.userAgent;

    //The following is an attempt to match Nokia browsers that are running Symbian/s60, with
    webkit, version 7.3 or older
    return ua.indexOf( "Nokia" ) > -1 &&
            ( ua.indexOf( "Symbian/3" ) > -1 || ua.indexOf( "Series60/5" ) > -1 ) &&
```

```
            ua.indexOf( "AppleWebKit" ) > -1 &&
            ua.match( /(BrowserNG|NokiaBrowser)\/7\.[0-3]/ );
})();

// Support conditions that must be met in order to proceed
// default enhanced qualifications are media query support OR IE 7+

$.mobile.gradeA = function() {
    return ( ( $.support.mediaquery && $.support.cssPseudoElement ) || $.mobile.browser.oldIE &&
     $.mobile.browser.oldIE >= 8 ) && ( $.support.boundingRect || $.fn.jquery.match(
    /1\.[0-7+]\.[0-9+]?/) !== null );
};

$.mobile.ajaxBlacklist =
            // BlackBerry browsers, pre-webkit
            window.blackberry && !window.WebKitPoint ||
            // Opera Mini
            operamini ||
            // Symbian webkits pre 7.3
            nokiaLTE7_3;

// Lastly, this workaround is the only way we've found so far to get pre 7.3 Symbian webkit
devices
// to render the stylesheets when they're referenced before this script, as we'd recommend doing.
// This simply reappends the CSS in place, which for some reason makes it apply
if ( nokiaLTE7_3 ) {
    $(function() {
        $( "head link[rel='stylesheet']" ).attr( "rel", "alternate stylesheet" ).attr( "rel",
        "stylesheet" );
    });
}

// For ruling out shadows via css
if ( !$.support.boxShadow ) {
    $( "html" ).addClass( "ui-noboxshadow" );
}

})( jQuery );


(function( $, undefined ) {
    var $win = $.mobile.window, self,
        dummyFnToInitNavigate = function() {
        };

    $.event.special.beforenavigate = {
        setup: function() {
            $win.on( "navigate", dummyFnToInitNavigate );
        },

        teardown: function() {
            $win.off( "navigate", dummyFnToInitNavigate );
        }
    };
```

```
$.event.special.navigate = self = {
    bound: false,

    pushStateEnabled: true,

    originalEventName: undefined,

    // If pushstate support is present and push state support is defined to
    // be true on the mobile namespace.
    isPushStateEnabled: function() {
        return $.support.pushState &&
            $.mobile.pushStateEnabled === true &&
            this.isHashChangeEnabled();
    },

    // !! assumes mobile namespace is present
    isHashChangeEnabled: function() {
        return $.mobile.hashListeningEnabled === true;
    },

    // TODO a lot of duplication between popstate and hashchange
    popstate: function( event ) {
        var newEvent = new $.Event( "navigate" ),
            beforeNavigate = new $.Event( "beforenavigate" ),
            state = event.originalEvent.state || {};

        beforeNavigate.originalEvent = event;
        $win.trigger( beforeNavigate );

        if ( beforeNavigate.isDefaultPrevented() ) {
            return;
        }

        if ( event.historyState ) {
            $.extend(state, event.historyState);
        }

        // Make sure the original event is tracked for the end
        // user to inspect incase they want to do something special
        newEvent.originalEvent = event;

        // NOTE we let the current stack unwind because any assignment to
        //      location.hash will stop the world and run this event handler. By
        //      doing this we create a similar behavior to hashchange on hash
        //      assignment
        setTimeout(function() {
            $win.trigger( newEvent, {
                state: state
            });
        }, 0);
    },

    hashchange: function( event /*, data */ ) {
```

```
            var newEvent = new $.Event( "navigate" ),
                beforeNavigate = new $.Event( "beforenavigate" );

            beforeNavigate.originalEvent = event;
            $win.trigger( beforeNavigate );

            if ( beforeNavigate.isDefaultPrevented() ) {
                return;
            }

            // Make sure the original event is tracked for the end
            // user to inspect incase they want to do something special
            newEvent.originalEvent = event;

            // Trigger the hashchange with state provided by the user
            // that altered the hash
            $win.trigger( newEvent, {
                // Users that want to fully normalize the two events
                // will need to do history management down the stack and
                // add the state to the event before this binding is fired
                // TODO consider allowing for the explicit addition of callbacks
                //      to be fired before this value is set to avoid event timing issues
                state: event.hashchangeState || {}
            });
        },

        // TODO We really only want to set this up once
        //      but I'm not clear if there's a beter way to achieve
        //      this with the jQuery special event structure
        setup: function( /* data, namespaces */ ) {
            if ( self.bound ) {
                return;
            }

            self.bound = true;

            if ( self.isPushStateEnabled() ) {
                self.originalEventName = "popstate";
                $win.bind( "popstate.navigate", self.popstate );
            } else if ( self.isHashChangeEnabled() ) {
                self.originalEventName = "hashchange";
                $win.bind( "hashchange.navigate", self.hashchange );
            }
        }
    };
})( jQuery );



(function( $, undefined ) {
        var path, $base, dialogHashKey = "&ui-state=dialog";

        $.mobile.path = path = {
            uiStateKey: "&ui-state",
```

```
        // This scary looking regular expression parses an absolute URL or its relative
        // variants (protocol, site, document, query, and hash), into the various
        // components (protocol, host, path, query, fragment, etc that make up the
        // URL as well as some other commonly used sub-parts. When used with RegExp.exec()
        // or String.match, it parses the URL into a results array that looks like this:
        //
        //     [0]:
        http://jblas:password@mycompany.com:8080/mail/inbox?msg=1234&type=unread#msg-content
        //     [1]: http://jblas:password@mycompany.com:8080/mail/inbox?msg=1234&type=unread
        //     [2]: http://jblas:password@mycompany.com:8080/mail/inbox
        //     [3]: http://jblas:password@mycompany.com:8080
        //     [4]: http:
        //     [5]: //
        //     [6]: jblas:password@mycompany.com:8080
        //     [7]: jblas:password
        //     [8]: jblas
        //     [9]: password
        //    [10]: mycompany.com:8080
        //    [11]: mycompany.com
        //    [12]: 8080
        //    [13]: /mail/inbox
        //    [14]: /mail/
        //    [15]: inbox
        //    [16]: ?msg=1234&type=unread
        //    [17]: #msg-content
        //
        urlParseRE:
        /^\s*(((([^:\/#\?]+:)?(?:(\/\/)((?:(([^:@\/#\?]+)(?:\:([^:@\/#\?]+))?)@)?(([^:\/#\?\]
        \[]+|\[[^\/\]@#?]+\])(?:\:([0-9]+))?))?)?)?((\/?(?:[^\/\?#]+\/+)*)([^\?#]*)))?(\?[^#]
        +)?)(#.*)?/,

        // Abstraction to address xss (Issue #4787) by removing the authority in
        // browsers that auto   decode it. All references to location.href should be
        // replaced with a call to this method so that it can be dealt with properly here
        getLocation: function( url ) {
            var uri = url ? this.parseUrl( url ) : location,
                hash = this.parseUrl( url || location.href ).hash;

            // mimic the browser with an empty string when the hash is empty
            hash = hash === "#" ? "" : hash;

            // Make sure to parse the url or the location object for the hash because using
            location.hash
            // is autodecoded in firefox, the rest of the url should be from the object
            (location unless
            // we're testing) to avoid the inclusion of the authority
            return uri.protocol + "//" + uri.host + uri.pathname + uri.search + hash;
        },

        //return the original document url
        getDocumentUrl: function( asParsedObject ) {
            return asParsedObject ? $.extend( {}, path.documentUrl ) : path.documentUrl.href;
        },
```

```
        parseLocation: function() {
            return this.parseUrl( this.getLocation() );
        },

        //Parse a URL into a structure that allows easy access to
        //all of the URL components by name.
        parseUrl: function( url ) {
            // If we're passed an object, we'll assume that it is
            // a parsed url object and just return it back to the caller.
            if ( $.type( url ) === "object" ) {
                return url;
            }

            var matches = path.urlParseRE.exec( url || "" ) || [];

                // Create an object that allows the caller to access the sub-matches
                // by name. Note that IE returns an empty string instead of undefined,
                // like all other browsers do, so we normalize everything so its consistent
                // no matter what browser we're running on.
                return {
                    href:         matches[  0 ] || "",
                    hrefNoHash:   matches[  1 ] || "",
                    hrefNoSearch: matches[  2 ] || "",
                    domain:       matches[  3 ] || "",
                    protocol:     matches[  4 ] || "",
                    doubleSlash:  matches[  5 ] || "",
                    authority:    matches[  6 ] || "",
                    username:     matches[  8 ] || "",
                    password:     matches[  9 ] || "",
                    host:         matches[ 10 ] || "",
                    hostname:     matches[ 11 ] || "",
                    port:         matches[ 12 ] || "",
                    pathname:     matches[ 13 ] || "",
                    directory:    matches[ 14 ] || "",
                    filename:     matches[ 15 ] || "",
                    search:       matches[ 16 ] || "",
                    hash:         matches[ 17 ] || ""
                };
        },

        //Turn relPath into an asbolute path. absPath is
        //an optional absolute path which describes what
        //relPath is relative to.
        makePathAbsolute: function( relPath, absPath ) {
            var absStack,
                relStack,
                i, d;

            if ( relPath && relPath.charAt( 0 ) === "/" ) {
                return relPath;
            }

            relPath = relPath || "";
```

```
            absPath = absPath ? absPath.replace( /^\/|(\/[^\/]*|[^\/]+)$/g, "" ) : "";

            absStack = absPath ? absPath.split( "/" ) : [];
            relStack = relPath.split( "/" );

            for ( i = 0; i < relStack.length; i++ ) {
                d = relStack[ i ];
                switch ( d ) {
                    case ".":
                        break;
                    case "..":
                        if ( absStack.length ) {
                            absStack.pop();
                        }
                        break;
                    default:
                        absStack.push( d );
                        break;
                }
            }
            return "/" + absStack.join( "/" );
        },

        //Returns true if both urls have the same domain.
        isSameDomain: function( absUrl1, absUrl2 ) {
            return path.parseUrl( absUrl1 ).domain === path.parseUrl( absUrl2 ).domain;
        },

        //Returns true for any relative variant.
        isRelativeUrl: function( url ) {
            // All relative Url variants have one thing in common, no protocol.
            return path.parseUrl( url ).protocol === "";
        },

        //Returns true for an absolute url.
        isAbsoluteUrl: function( url ) {
            return path.parseUrl( url ).protocol !== "";
        },

        //Turn the specified realtive URL into an absolute one. This function
        //can handle all relative variants (protocol, site, document, query, fragment).
        makeUrlAbsolute: function( relUrl, absUrl ) {
            if ( !path.isRelativeUrl( relUrl ) ) {
                return relUrl;
            }

            if ( absUrl === undefined ) {
                absUrl = this.documentBase;
            }

            var relObj = path.parseUrl( relUrl ),
                absObj = path.parseUrl( absUrl ),
                protocol = relObj.protocol || absObj.protocol,
                doubleSlash = relObj.protocol ? relObj.doubleSlash : ( relObj.doubleSlash ||
```

```
                absObj.doubleSlash ),
            authority = relObj.authority || absObj.authority,
            hasPath = relObj.pathname !== "",
            pathname = path.makePathAbsolute( relObj.pathname || absObj.filename, absObj
            .pathname ),
            search = relObj.search || ( !hasPath && absObj.search ) || "",
            hash = relObj.hash;

        return protocol + doubleSlash + authority + pathname + search + hash;
    },

    //Add search (aka query) params to the specified url.
    addSearchParams: function( url, params ) {
        var u = path.parseUrl( url ),
            p = ( typeof params === "object" ) ? $.param( params ) : params,
            s = u.search || "?";
        return u.hrefNoSearch + s + ( s.charAt( s.length - 1 ) !== "?" ? "&" : "" ) + p
        + ( u.hash || "" );
    },

    convertUrlToDataUrl: function( absUrl ) {
        var u = path.parseUrl( absUrl );
        if ( path.isEmbeddedPage( u ) ) {
            // For embedded pages, remove the dialog hash key as in getFilePath(),
            // and remove otherwise the Data Url won't match the id of the embedded Page.
            return u.hash
                .split( dialogHashKey )[0]
                .replace( /^#/, "" )
                .replace( /\?.*$/, "" );
        } else if ( path.isSameDomain( u, this.documentBase ) ) {
            return u.hrefNoHash.replace( this.documentBase.domain, "" ).split(
            dialogHashKey )[0];
        }

        return window.decodeURIComponent(absUrl);
    },

    //get path from current hash, or from a file path
    get: function( newPath ) {
        if ( newPath === undefined ) {
            newPath = path.parseLocation().hash;
        }
        return path.stripHash( newPath ).replace( /[^\/]*\.[^\/*]+$/, "" );
    },

    //set location hash to path
    set: function( path ) {
        location.hash = path;
    },

    //test if a given url (string) is a path
    //NOTE might be exceptionally naive
    isPath: function( url ) {
        return ( /\// ).test( url );
```

```
        },

        //return a url path with the window's location protocol/hostname/pathname removed
        clean: function( url ) {
            return url.replace( this.documentBase.domain, "" );
        },

        //just return the url without an initial #
        stripHash: function( url ) {
            return url.replace( /^#/, "" );
        },

        stripQueryParams: function( url ) {
            return url.replace( /\?.*$/, "" );
        },

        //remove the preceding hash, any query params, and dialog notations
        cleanHash: function( hash ) {
            return path.stripHash( hash.replace( /\?.*$/, "" ).replace( dialogHashKey, "" )
            );
        },

        isHashValid: function( hash ) {
            return ( /^#[^#]+$/ ).test( hash );
        },

        //check whether a url is referencing the same domain, or an external domain or
        different protocol
        //could be mailto, etc
        isExternal: function( url ) {
            var u = path.parseUrl( url );
            return u.protocol && u.domain !== this.documentUrl.domain ? true : false;
        },

        hasProtocol: function( url ) {
            return ( /^(:?\w+:)/ ).test( url );
        },

        isEmbeddedPage: function( url ) {
            var u = path.parseUrl( url );

            //if the path is absolute, then we need to compare the url against
            //both the this.documentUrl and the documentBase. The main reason for this
            //is that links embedded within external documents will refer to the
            //application document, whereas links embedded within the application
            //document will be resolved against the document base.
            if ( u.protocol !== "" ) {
                return ( !this.isPath(u.hash) && u.hash && ( u.hrefNoHash === this.
                documentUrl.hrefNoHash || ( this.documentBaseDiffers && u.hrefNoHash ===
                this.documentBase.hrefNoHash ) ) );
            }
            return ( /^#/ ).test( u.href );
        },
```

```
            squash: function( url, resolutionUrl ) {
                var href, cleanedUrl, search, stateIndex,
                    isPath = this.isPath( url ),
                    uri = this.parseUrl( url ),
                    preservedHash = uri.hash,
                    uiState = "";

                // produce a url against which we can resole the provided path
                resolutionUrl = resolutionUrl || (path.isPath(url) ? path.getLocation() : path.
                getDocumentUrl());

                // If the url is anything but a simple string, remove any preceding hash
                // eg #foo/bar -> foo/bar
                //    #foo -> #foo
                cleanedUrl = isPath ? path.stripHash( url ) : url;

                // If the url is a full url with a hash check if the parsed hash is a path
                // if it is, strip the #, and use it otherwise continue without change
                cleanedUrl = path.isPath( uri.hash ) ? path.stripHash( uri.hash ) : cleanedUrl;

                // Split the UI State keys off the href
                stateIndex = cleanedUrl.indexOf( this.uiStateKey );

                // store the ui state keys for use
                if ( stateIndex > -1 ) {
                    uiState = cleanedUrl.slice( stateIndex );
                    cleanedUrl = cleanedUrl.slice( 0, stateIndex );
                }

                // make the cleanedUrl absolute relative to the resolution url
                href = path.makeUrlAbsolute( cleanedUrl, resolutionUrl );

                // grab the search from the resolved url since parsing from
                // the passed url may not yield the correct result
                search = this.parseUrl( href ).search;

                // TODO all this crap is terrible, clean it up
                if ( isPath ) {
                    // reject the hash if it's a path or it's just a dialog key
                    if ( path.isPath( preservedHash ) || preservedHash.replace("#", "").indexOf(
                     this.uiStateKey ) === 0) {
                        preservedHash = "";
                    }

                    // Append the UI State keys where it exists and it's been removed
                    // from the url
                    if ( uiState && preservedHash.indexOf( this.uiStateKey ) === -1) {
                        preservedHash += uiState;
                    }

                    // make sure that pound is on the front of the hash
                    if ( preservedHash.indexOf( "#" ) === -1 && preservedHash !== "" ) {
                        preservedHash = "#" + preservedHash;
                    }
```

```
                // reconstruct each of the pieces with the new search string and hash
                href = path.parseUrl( href );
                href = href.protocol + "//" + href.host + href.pathname + search +
                preservedHash;
            } else {
                href += href.indexOf( "#" ) > -1 ? uiState : "#" + uiState;
            }

            return href;
        },

        isPreservableHash: function( hash ) {
            return hash.replace( "#", "" ).indexOf( this.uiStateKey ) === 0;
        },

        // Escape weird characters in the hash if it is to be used as a selector
        hashToSelector: function( hash ) {
            var hasHash = ( hash.substring( 0, 1 ) === "#" );
            if ( hasHash ) {
                hash = hash.substring( 1 );
            }
            return ( hasHash ? "#" : "" ) + hash.replace(
            /([!"#$%&'()*+,./:;<=>?@[\]^`{|}~])/g, "\\$1" );
        },

        // return the substring of a filepath before the sub-page key, for making
        // a server request
        getFilePath: function( path ) {
            var splitkey = "&" + $.mobile.subPageUrlKey;
            return path && path.split( splitkey )[0].split( dialogHashKey )[0];
        },

        // check if the specified url refers to the first page in the main
        // application document.
        isFirstPageUrl: function( url ) {
            // We only deal with absolute paths.
            var u = path.parseUrl( path.makeUrlAbsolute( url, this.documentBase ) ),

                // Does the url have the same path as the document?
                samePath = u.hrefNoHash === this.documentUrl.hrefNoHash ||
                    ( this.documentBaseDiffers &&
                        u.hrefNoHash === this.documentBase.hrefNoHash ),

                // Get the first page element.
                fp = $.mobile.firstPage,

                // Get the id of the first page element if it has one.
                fpId = fp && fp[0] ? fp[0].id : undefined;

            // The url refers to the first page if the path matches the document and
            // it either has no hash value, or the hash is exactly equal to the id
            // of the first page element.
            return samePath &&
```

```
                    ( !u.hash ||
                        u.hash === "#" ||
                        ( fpId && u.hash.replace( /^#/, "" ) === fpId ) );
            },

            // Some embedded browsers, like the web view in Phone Gap, allow
            // cross-domain XHR requests if the document doing the request was loaded
            // via the file:// protocol. This is usually to allow the application to
            // "phone home" and fetch app specific data. We normally let the browser
            // handle external/cross-domain urls, but if the allowCrossDomainPages
            // option is true, we will allow cross-domain http/https requests to go
            // through our page loading logic.
            isPermittedCrossDomainRequest: function( docUrl, reqUrl ) {
                return $.mobile.allowCrossDomainPages &&
                    (docUrl.protocol === "file:" || docUrl.protocol === "content:") &&
                    reqUrl.search( /^https?:/ ) !== -1;
            }
        };

        path.documentUrl = path.parseLocation();

        $base = $( "head" ).find( "base" );

        path.documentBase = $base.length ?
            path.parseUrl( path.makeUrlAbsolute( $base.attr( "href" ), path.documentUrl.href ) )
             :
            path.documentUrl;

        path.documentBaseDiffers = (path.documentUrl.hrefNoHash !== path.documentBase.hrefNoHash
        );

        //return the original document base url
        path.getDocumentBase = function( asParsedObject ) {
            return asParsedObject ? $.extend( {}, path.documentBase ) : path.documentBase.href;
        };

        // DEPRECATED as of 1.4.0 - remove in 1.5.0
        $.extend( $.mobile, {

            //return the original document url
            getDocumentUrl: path.getDocumentUrl,

            //return the original document base url
            getDocumentBase: path.getDocumentBase
        });
})( jQuery );



(function( $, undefined ) {
    $.mobile.History = function( stack, index ) {
        this.stack = stack || [];
        this.activeIndex = index || 0;
    };
```

```
$.extend($.mobile.History.prototype, {
    getActive: function() {
        return this.stack[ this.activeIndex ];
    },

    getLast: function() {
        return this.stack[ this.previousIndex ];
    },

    getNext: function() {
        return this.stack[ this.activeIndex + 1 ];
    },

    getPrev: function() {
        return this.stack[ this.activeIndex - 1 ];
    },

    // addNew is used whenever a new page is added
    add: function( url, data ) {
        data = data || {};

        //if there's forward history, wipe it
        if ( this.getNext() ) {
            this.clearForward();
        }

        // if the hash is included in the data make sure the shape
        // is consistent for comparison
        if ( data.hash && data.hash.indexOf( "#" ) === -1) {
            data.hash = "#" + data.hash;
        }

        data.url = url;
        this.stack.push( data );
        this.activeIndex = this.stack.length - 1;
    },

    //wipe urls ahead of active index
    clearForward: function() {
        this.stack = this.stack.slice( 0, this.activeIndex + 1 );
    },

    find: function( url, stack, earlyReturn ) {
        stack = stack || this.stack;

        var entry, i, length = stack.length, index;

        for ( i = 0; i < length; i++ ) {
            entry = stack[i];

            if ( decodeURIComponent(url) === decodeURIComponent(entry.url) ||
                decodeURIComponent(url) === decodeURIComponent(entry.hash) ) {
                index = i;
```

```
                if ( earlyReturn ) {
                    return index;
                }
            }
        }

        return index;
    },

    closest: function( url ) {
        var closest, a = this.activeIndex;

        // First, take the slice of the history stack before the current index and search
        // for a url match. If one is found, we'll avoid avoid looking through forward
        history
        // NOTE the preference for backward history movement is driven by the fact that
        //      most mobile browsers only have a dedicated back button, and users rarely use
        //      the forward button in desktop browser anyhow
        closest = this.find( url, this.stack.slice(0, a) );

        // If nothing was found in backward history check forward. The `true`
        // value passed as the third parameter causes the find method to break
        // on the first match in the forward history slice. The starting index
        // of the slice must then be added to the result to get the element index
        // in the original history stack :( :(
        //
        // TODO this is hyper confusing and should be cleaned up (ugh so bad)
        if ( closest === undefined ) {
            closest = this.find( url, this.stack.slice(a), true );
            closest = closest === undefined ? closest : closest + a;
        }

        return closest;
    },

    direct: function( opts ) {
        var newActiveIndex = this.closest( opts.url ), a = this.activeIndex;

        // save new page index, null check to prevent falsey 0 result
        // record the previous index for reference
        if ( newActiveIndex !== undefined ) {
            this.activeIndex = newActiveIndex;
            this.previousIndex = a;
        }

        // invoke callbacks where appropriate
        //
        // TODO this is also convoluted and confusing
        if ( newActiveIndex < a ) {
            ( opts.present || opts.back || $.noop )( this.getActive(), "back" );
        } else if ( newActiveIndex > a ) {
            ( opts.present || opts.forward || $.noop )( this.getActive(), "forward" );
        } else if ( newActiveIndex === undefined && opts.missing ) {
```

```
                opts.missing( this.getActive() );
            }
        }
    });
})( jQuery );



(function( $, undefined ) {
    var path = $.mobile.path,
        initialHref = location.href;

    $.mobile.Navigator = function( history ) {
        this.history = history;
        this.ignoreInitialHashChange = true;

        $.mobile.window.bind({
            "popstate.history": $.proxy( this.popstate, this ),
            "hashchange.history": $.proxy( this.hashchange, this )
        });
    };

    $.extend($.mobile.Navigator.prototype, {
        squash: function( url, data ) {
            var state, href, hash = path.isPath(url) ? path.stripHash(url) : url;

            href = path.squash( url );

            // make sure to provide this information when it isn't explicitly set in the
            // data object that was passed to the squash method
            state = $.extend({
                hash: hash,
                url: href
            }, data);

            // replace the current url with the new href and store the state
            // Note that in some cases we might be replacing an url with the
            // same url. We do this anyways because we need to make sure that
            // all of our history entries have a state object associated with
            // them. This allows us to work around the case where $.mobile.back()
            // is called to transition from an external page to an embedded page.
            // In that particular case, a hashchange event is *NOT* generated by the browser.
            // Ensuring each history entry has a state object means that onPopState()
            // will always trigger our hashchange callback even when a hashchange event
            // is not fired.
            window.history.replaceState( state, state.title || document.title, href );

            return state;
        },

        hash: function( url, href ) {
            var parsed, loc, hash, resolved;

            // Grab the hash for recording. If the passed url is a path
```

```
            // we used the parsed version of the squashed url to reconstruct,
            // otherwise we assume it's a hash and store it directly
            parsed = path.parseUrl( url );
            loc = path.parseLocation();

            if ( loc.pathname + loc.search === parsed.pathname + parsed.search ) {
                // If the pathname and search of the passed url is identical to the current loc
                // then we must use the hash. Otherwise there will be no event
                // eg, url = "/foo/bar?baz#bang", location.href =
                "http://example.com/foo/bar?baz"
                hash = parsed.hash ? parsed.hash : parsed.pathname + parsed.search;
            } else if ( path.isPath(url) ) {
                resolved = path.parseUrl( href );
                // If the passed url is a path, make it domain relative and remove any trailing
                hash
                hash = resolved.pathname + resolved.search + (path.isPreservableHash( resolved.
                hash )? resolved.hash.replace( "#", "" ) : "");
            } else {
                hash = url;
            }

            return hash;
        },

        // TODO reconsider name
        go: function( url, data, noEvents ) {
            var state, href, hash, popstateEvent,
                isPopStateEvent = $.event.special.navigate.isPushStateEnabled();

            // Get the url as it would look squashed on to the current resolution url
            href = path.squash( url );

            // sort out what the hash sould be from the url
            hash = this.hash( url, href );

            // Here we prevent the next hash change or popstate event from doing any
            // history management. In the case of hashchange we don't swallow it
            // if there will be no hashchange fired (since that won't reset the value)
            // and will swallow the following hashchange
            if ( noEvents && hash !== path.stripHash(path.parseLocation().hash) ) {
                this.preventNextHashChange = noEvents;
            }

            // IMPORTANT in the case where popstate is supported the event will be triggered
            //      directly, stopping further execution - ie, interupting the flow of this
            //      method call to fire bindings at this expression. Below the navigate method
            //      there is a binding to catch this event and stop its propagation.
            //
            //      We then trigger a new popstate event on the window with a null state
            //      so that the navigate events can conclude their work properly
            //
            // if the url is a path we want to preserve the query params that are available on
            // the current url.
            this.preventHashAssignPopState = true;
```

```
            window.location.hash = hash;

            // If popstate is enabled and the browser triggers `popstate` events when the hash
            // is set (this often happens immediately in browsers like Chrome), then the
            // this flag will be set to false already. If it's a browser that does not trigger
            // a `popstate` on hash assignement or `replaceState` then we need avoid the branch
            // that swallows the event created by the popstate generated by the hash assignment
            // At the time of this writing this happens with Opera 12 and some version of IE
            this.preventHashAssignPopState = false;

            state = $.extend({
                url: href,
                hash: hash,
                title: document.title
            }, data);

            if ( isPopStateEvent ) {
                popstateEvent = new $.Event( "popstate" );
                popstateEvent.originalEvent = {
                    type: "popstate",
                    state: null
                };

                this.squash( url, state );

                // Trigger a new faux popstate event to replace the one that we
                // caught that was triggered by the hash setting above.
                if ( !noEvents ) {
                    this.ignorePopState = true;
                    $.mobile.window.trigger( popstateEvent );
                }
            }

            // record the history entry so that the information can be included
            // in hashchange event driven navigate events in a similar fashion to
            // the state that's provided by popstate
            this.history.add( state.url, state );
        },

        // This binding is intended to catch the popstate events that are fired
        // when execution of the `$.navigate` method stops at window.location.hash = url;
        // and completely prevent them from propagating. The popstate event will then be
        // retriggered after execution resumes
        //
        // TODO grab the original event here and use it for the synthetic event in the
        //      second half of the navigate execution that will follow this binding
        popstate: function( event ) {
            var hash, state;

            // Partly to support our test suite which manually alters the support
            // value to test hashchange. Partly to prevent all around weirdness
            if ( !$.event.special.navigate.isPushStateEnabled() ) {
                return;
            }
```

```
        // If this is the popstate triggered by the actual alteration of the hash
        // prevent it completely. History is tracked manually
        if ( this.preventHashAssignPopState ) {
            this.preventHashAssignPopState = false;
            event.stopImmediatePropagation();
            return;
        }

        // if this is the popstate triggered after the `replaceState` call in the go
        // method, then simply ignore it. The history entry has already been captured
        if ( this.ignorePopState ) {
            this.ignorePopState = false;
            return;
        }

        // If there is no state, and the history stack length is one were
        // probably getting the page load popstate fired by browsers like chrome
        // avoid it and set the one time flag to false.
        // TODO: Do we really need all these conditions? Comparing location hrefs
        // should be sufficient.
        if ( !event.originalEvent.state &&
            this.history.stack.length === 1 &&
            this.ignoreInitialHashChange ) {
            this.ignoreInitialHashChange = false;

            if ( location.href === initialHref ) {
                event.preventDefault();
                return;
            }
        }

        // account for direct manipulation of the hash. That is, we will receive a popstate
        // when the hash is changed by assignment, and it won't have a state associated. We
        // then need to squash the hash. See below for handling of hash assignment that
        // matches an existing history entry
        // TODO it might be better to only add to the history stack
        //      when the hash is adjacent to the active history entry
        hash = path.parseLocation().hash;
        if ( !event.originalEvent.state && hash ) {
            // squash the hash that's been assigned on the URL with replaceState
            // also grab the resulting state object for storage
            state = this.squash( hash );

            // record the new hash as an additional history entry
            // to match the browser's treatment of hash assignment
            this.history.add( state.url, state );

            // pass the newly created state information
            // along with the event
            event.historyState = state;

            // do not alter history, we've added a new history entry
            // so we know where we are
```

```
            return;
        }

        // If all else fails this is a popstate that comes from the back or forward buttons
        // make sure to set the state of our history stack properly, and record the
        directionality
        this.history.direct({
            url: (event.originalEvent.state || {}).url || hash,

            // When the url is either forward or backward in history include the entry
            // as data on the event object for merging as data in the navigate event
            present: function( historyEntry, direction ) {
                // make sure to create a new object to pass down as the navigate event data
                event.historyState = $.extend({}, historyEntry);
                event.historyState.direction = direction;
            }
        });
    },

    // NOTE must bind before `navigate` special event hashchange binding otherwise the
    //      navigation data won't be attached to the hashchange event in time for those
    //      bindings to attach it to the `navigate` special event
    // TODO add a check here that `hashchange.navigate` is bound already otherwise it's
    //      broken (exception?)
    hashchange: function( event ) {
        var history, hash;

        // If hashchange listening is explicitly disabled or pushstate is supported
        // avoid making use of the hashchange handler.
        if (!$.event.special.navigate.isHashChangeEnabled() ||
            $.event.special.navigate.isPushStateEnabled() ) {
            return;
        }

        // On occasion explicitly want to prevent the next hash from propogating because we
        only
        // with to alter the url to represent the new state do so here
        if ( this.preventNextHashChange ) {
            this.preventNextHashChange = false;
            event.stopImmediatePropagation();
            return;
        }

        history = this.history;
        hash = path.parseLocation().hash;

        // If this is a hashchange caused by the back or forward button
        // make sure to set the state of our history stack properly
        this.history.direct({
            url: hash,

            // When the url is either forward or backward in history include the entry
            // as data on the event object for merging as data in the navigate event
            present: function( historyEntry, direction ) {
```

```
                    // make sure to create a new object to pass down as the navigate event data
                    event.hashchangeState = $.extend({}, historyEntry);
                    event.hashchangeState.direction = direction;
                },

                // When we don't find a hash in our history clearly we're aiming to go there
                // record the entry as new for future traversal
                //
                // NOTE it's not entirely clear that this is the right thing to do given that we
                //      can't know the users intention. It might be better to explicitly _not_
                //      support location.hash assignment in preference to $.navigate calls
                // TODO first arg to add should be the href, but it causes issues in identifying
                //      embeded pages
                missing: function() {
                    history.add( hash, {
                        hash: hash,
                        title: document.title
                    });
                }
            });
        }
    });
})( jQuery );



(function( $, undefined ) {
    // TODO consider queueing navigation activity until previous activities have completed
    //      so that end users don't have to think about it. Punting for now
    // TODO !! move the event bindings into callbacks on the navigate event
    $.mobile.navigate = function( url, data, noEvents ) {
        $.mobile.navigate.navigator.go( url, data, noEvents );
    };

    // expose the history on the navigate method in anticipation of full integration with
    // existing navigation functionalty that is tightly coupled to the history information
    $.mobile.navigate.history = new $.mobile.History();

    // instantiate an instance of the navigator for use within the $.navigate method
    $.mobile.navigate.navigator = new $.mobile.Navigator( $.mobile.navigate.history );

    var loc = $.mobile.path.parseLocation();
    $.mobile.navigate.history.add( loc.href, {hash: loc.hash} );
})( jQuery );


(function( $, undefined ) {
    var props = {
            "animation": {},
            "transition": {}
        },
        testElement = document.createElement( "a" ),
        vendorPrefixes = [ "", "webkit-", "moz-", "o-" ];
```

```
    $.each( [ "animation", "transition" ], function( i, test ) {

        // Get correct name for test
        var testName = ( i === 0 ) ? test + "-" + "name" : test;

        $.each( vendorPrefixes, function( j, prefix ) {
            if ( testElement.style[ $.camelCase( prefix + testName ) ] !== undefined ) {
                 props[ test ][ "prefix" ] = prefix;
                return false;
            }
        });

        // Set event and duration names for later use
        props[ test ][ "duration" ] =
            $.camelCase( props[ test ][ "prefix" ] + test + "-" + "duration" );
        props[ test ][ "event" ] =
            $.camelCase( props[ test ][ "prefix" ] + test + "-" + "end" );

        // All lower case if not a vendor prop
        if ( props[ test ][ "prefix" ] === "" ) {
            props[ test ][ "event" ] = props[ test ][ "event" ].toLowerCase();
        }
    });

    // If a valid prefix was found then the it is supported by the browser
    $.support.cssTransitions = ( props[ "transition" ][ "prefix" ] !== undefined );
    $.support.cssAnimations = ( props[ "animation" ][ "prefix" ] !== undefined );

    // Remove the testElement
    $( testElement ).remove();

    // Animation complete callback
    $.fn.animationComplete = function( callback, type, fallbackTime ) {
        var timer, duration,
            that = this,
            animationType = ( !type || type === "animation" ) ? "animation" : "transition";

        // Make sure selected type is supported by browser
        if ( ( $.support.cssTransitions && animationType === "transition" ) ||
            ( $.support.cssAnimations && animationType === "animation" ) ) {

            // If a fallback time was not passed set one
            if ( fallbackTime === undefined ) {

                // Make sure the was not bound to document before checking .css
                if ( $( this ).context !== document ) {

                    // Parse the durration since its in second multiple by 1000 for milliseconds
                    // Multiply by 3 to make sure we give the animation plenty of time.
                    duration = parseFloat(
                        $( this ).css( props[ animationType ].duration )
                    ) * 3000;
                }
```

```
                // If we could not read a duration use the default
                if ( duration === 0 || duration === undefined || isNaN( duration ) ) {
                    duration = $.fn.animationComplete.defaultDuration;
                }
            }

            // Sets up the fallback if event never comes
            timer = setTimeout( function() {
                $( that ).off( props[ animationType ].event );
                callback.apply( that );
            }, duration );

            // Bind the event
            return $( this ).one( props[ animationType ].event, function() {

                // Clear the timer so we dont call callback twice
                clearTimeout( timer );
                callback.call( this, arguments );
            });
        } else {

            // CSS animation / transitions not supported
            // Defer execution for consistency between webkit/non webkit
            setTimeout( $.proxy( callback, this ), 0 );
            return $( this );
        }
    };

    // Allow default callback to be configured on mobileInit
    $.fn.animationComplete.defaultDuration = 1000;
})( jQuery );

// This plugin is an experiment for abstracting away the touch and mouse
// events so that developers don't have to worry about which method of input
// the device their document is loaded on supports.
//
// The idea here is to allow the developer to register listeners for the
// basic mouse events, such as mousedown, mousemove, mouseup, and click,
// and the plugin will take care of registering the correct listeners
// behind the scenes to invoke the listener at the fastest possible time
// for that device, while still retaining the order of event firing in
// the traditional mouse environment, should multiple handlers be registered
// on the same element for different events.
//
// The current version exposes the following virtual events to jQuery bind methods:
// "vmouseover vmousedown vmousemove vmouseup vclick vmouseout vmousecancel"

(function( $, window, document, undefined ) {

var dataPropertyName = "virtualMouseBindings",
    touchTargetPropertyName = "virtualTouchID",
    virtualEventNames = "vmouseover vmousedown vmousemove vmouseup vclick vmouseout
    vmousecancel".split( " " ),
    touchEventProps = "clientX clientY pageX pageY screenX screenY".split( " " ),
```

```
    mouseHookProps = $.event.mouseHooks ? $.event.mouseHooks.props : [],
    mouseEventProps = $.event.props.concat( mouseHookProps ),
    activeDocHandlers = {},
    resetTimerID = 0,
    startX = 0,
    startY = 0,
    didScroll = false,
    clickBlockList = [],
    blockMouseTriggers = false,
    blockTouchTriggers = false,
    eventCaptureSupported = "addEventListener" in document,
    $document = $( document ),
    nextTouchID = 1,
    lastTouchID = 0, threshold,
    i;

$.vmouse = {
    moveDistanceThreshold: 10,
    clickDistanceThreshold: 10,
    resetTimerDuration: 1500
};

function getNativeEvent( event ) {

    while ( event && typeof event.originalEvent !== "undefined" ) {
        event = event.originalEvent;
    }
    return event;
}

function createVirtualEvent( event, eventType ) {

    var t = event.type,
        oe, props, ne, prop, ct, touch, i, j, len;

    event = $.Event( event );
    event.type = eventType;

    oe = event.originalEvent;
    props = $.event.props;

    // addresses separation of $.event.props in to $.event.mouseHook.props and Issue 3280
    // https://github.com/jquery/jquery-mobile/issues/3280
    if ( t.search( /^(mouse|click)/ ) > -1 ) {
        props = mouseEventProps;
    }

    // copy original event properties over to the new event
    // this would happen if we could call $.event.fix instead of $.Event
    // but we don't have a way to force an event to be fixed multiple times
    if ( oe ) {
        for ( i = props.length, prop; i; ) {
            prop = props[ --i ];
            event[ prop ] = oe[ prop ];
```

```
        }
    }

    // make sure that if the mouse and click virtual events are generated
    // without a .which one is defined
    if ( t.search(/mouse(down|up)|click/) > -1 && !event.which ) {
        event.which = 1;
    }

    if ( t.search(/^touch/) !== -1 ) {
        ne = getNativeEvent( oe );
        t = ne.touches;
        ct = ne.changedTouches;
        touch = ( t && t.length ) ? t[0] : ( ( ct && ct.length ) ? ct[ 0 ] : undefined );

        if ( touch ) {
            for ( j = 0, len = touchEventProps.length; j < len; j++) {
                prop = touchEventProps[ j ];
                event[ prop ] = touch[ prop ];
            }
        }
    }

    return event;
}

function getVirtualBindingFlags( element ) {

    var flags = {},
        b, k;

    while ( element ) {

        b = $.data( element, dataPropertyName );

        for (  k in b ) {
            if ( b[ k ] ) {
                flags[ k ] = flags.hasVirtualBinding = true;
            }
        }
        element = element.parentNode;
    }
    return flags;
}

function getClosestElementWithVirtualBinding( element, eventType ) {
    var b;
    while ( element ) {

        b = $.data( element, dataPropertyName );

        if ( b && ( !eventType || b[ eventType ] ) ) {
            return element;
        }
```

```
        element = element.parentNode;
    }
    return null;
}

function enableTouchBindings() {
    blockTouchTriggers = false;
}

function disableTouchBindings() {
    blockTouchTriggers = true;
}

function enableMouseBindings() {
    lastTouchID = 0;
    clickBlockList.length = 0;
    blockMouseTriggers = false;

    // When mouse bindings are enabled, our
    // touch bindings are disabled.
    disableTouchBindings();
}

function disableMouseBindings() {
    // When mouse bindings are disabled, our
    // touch bindings are enabled.
    enableTouchBindings();
}

function startResetTimer() {
    clearResetTimer();
    resetTimerID = setTimeout( function() {
        resetTimerID = 0;
        enableMouseBindings();
    }, $.vmouse.resetTimerDuration );
}

function clearResetTimer() {
    if ( resetTimerID ) {
        clearTimeout( resetTimerID );
        resetTimerID = 0;
    }
}

function triggerVirtualEvent( eventType, event, flags ) {
    var ve;

    if ( ( flags && flags[ eventType ] ) ||
                ( !flags && getClosestElementWithVirtualBinding( event.target, eventType ) ) ) {

        ve = createVirtualEvent( event, eventType );

        $( event.target).trigger( ve );
    }
```

```
    return ve;
}

function mouseEventCallback( event ) {
    var touchID = $.data( event.target, touchTargetPropertyName ),
        ve;

    if ( !blockMouseTriggers && ( !lastTouchID || lastTouchID !== touchID ) ) {
        ve = triggerVirtualEvent( "v" + event.type, event );
        if ( ve ) {
            if ( ve.isDefaultPrevented() ) {
                event.preventDefault();
            }
            if ( ve.isPropagationStopped() ) {
                event.stopPropagation();
            }
            if ( ve.isImmediatePropagationStopped() ) {
                event.stopImmediatePropagation();
            }
        }
    }
}

function handleTouchStart( event ) {

    var touches = getNativeEvent( event ).touches,
        target, flags, t;

    if ( touches && touches.length === 1 ) {

        target = event.target;
        flags = getVirtualBindingFlags( target );

        if ( flags.hasVirtualBinding ) {

            lastTouchID = nextTouchID++;
            $.data( target, touchTargetPropertyName, lastTouchID );

            clearResetTimer();

            disableMouseBindings();
            didScroll = false;

            t = getNativeEvent( event ).touches[ 0 ];
            startX = t.pageX;
            startY = t.pageY;

            triggerVirtualEvent( "vmouseover", event, flags );
            triggerVirtualEvent( "vmousedown", event, flags );
        }
    }
}
```

```
function handleScroll( event ) {
    if ( blockTouchTriggers ) {
        return;
    }

    if ( !didScroll ) {
        triggerVirtualEvent( "vmousecancel", event, getVirtualBindingFlags( event.target ) );
    }

    didScroll = true;
    startResetTimer();
}

function handleTouchMove( event ) {
    if ( blockTouchTriggers ) {
        return;
    }

    var t = getNativeEvent( event ).touches[ 0 ],
        didCancel = didScroll,
        moveThreshold = $.vmouse.moveDistanceThreshold,
        flags = getVirtualBindingFlags( event.target );

        didScroll = didScroll ||
            ( Math.abs( t.pageX - startX ) > moveThreshold ||
                Math.abs( t.pageY - startY ) > moveThreshold );

    if ( didScroll && !didCancel ) {
        triggerVirtualEvent( "vmousecancel", event, flags );
    }

    triggerVirtualEvent( "vmousemove", event, flags );
    startResetTimer();
}

function handleTouchEnd( event ) {
    if ( blockTouchTriggers ) {
        return;
    }

    disableTouchBindings();

    var flags = getVirtualBindingFlags( event.target ),
        ve, t;
    triggerVirtualEvent( "vmouseup", event, flags );

    if ( !didScroll ) {
        ve = triggerVirtualEvent( "vclick", event, flags );
        if ( ve && ve.isDefaultPrevented() ) {
            // The target of the mouse events that follow the touchend
            // event don't necessarily match the target used during the
            // touch. This means we need to rely on coordinates for blocking
            // any click that is generated.
            t = getNativeEvent( event ).changedTouches[ 0 ];
```

```
            clickBlockList.push({
                touchID: lastTouchID,
                x: t.clientX,
                y: t.clientY
            });

            // Prevent any mouse events that follow from triggering
            // virtual event notifications.
            blockMouseTriggers = true;
        }
    }
    triggerVirtualEvent( "vmouseout", event, flags);
    didScroll = false;

    startResetTimer();
}

function hasVirtualBindings( ele ) {
    var bindings = $.data( ele, dataPropertyName ),
        k;

    if ( bindings ) {
        for ( k in bindings ) {
            if ( bindings[ k ] ) {
                return true;
            }
        }
    }
    return false;
}

function dummyMouseHandler() {}

function getSpecialEventObject( eventType ) {
    var realType = eventType.substr( 1 );

    return {
        setup: function(/* data, namespace */) {
            // If this is the first virtual mouse binding for this element,
            // add a bindings object to its data.

            if ( !hasVirtualBindings( this ) ) {
                $.data( this, dataPropertyName, {} );
            }

            // If setup is called, we know it is the first binding for this
            // eventType, so initialize the count for the eventType to zero.
            var bindings = $.data( this, dataPropertyName );
            bindings[ eventType ] = true;

            // If this is the first virtual mouse event for this type,
            // register a global handler on the document.

            activeDocHandlers[ eventType ] = ( activeDocHandlers[ eventType ] || 0 ) + 1;
```

```
            if ( activeDocHandlers[ eventType ] === 1 ) {
                $document.bind( realType, mouseEventCallback );
            }

            // Some browsers, like Opera Mini, won't dispatch mouse/click events
            // for elements unless they actually have handlers registered on them.
            // To get around this, we register dummy handlers on the elements.

            $( this ).bind( realType, dummyMouseHandler );

            // For now, if event capture is not supported, we rely on mouse handlers.
            if ( eventCaptureSupported ) {
                // If this is the first virtual mouse binding for the document,
                // register our touchstart handler on the document.

                activeDocHandlers[ "touchstart" ] = ( activeDocHandlers[ "touchstart" ] || 0) +
                1;

                if ( activeDocHandlers[ "touchstart" ] === 1 ) {
                    $document.bind( "touchstart", handleTouchStart )
                        .bind( "touchend", handleTouchEnd )

                        // On touch platforms, touching the screen and then dragging your finger
                        // causes the window content to scroll after some distance threshold is
                        // exceeded. On these platforms, a scroll prevents a click event from
                        being
                        // dispatched, and on some platforms, even the touchend is suppressed. To
                        // mimic the suppression of the click event, we need to watch for a
                        scroll
                        // event. Unfortunately, some platforms like iOS don't dispatch scroll
                        // events until *AFTER* the user lifts their finger (touchend). This
                        means
                        // we need to watch both scroll and touchmove events to figure out
                        whether
                        // or not a scroll happenens before the touchend event is fired.

                        .bind( "touchmove", handleTouchMove )
                        .bind( "scroll", handleScroll );
                }
            }
        },

        teardown: function(/* data, namespace */) {
            // If this is the last virtual binding for this eventType,
            // remove its global handler from the document.

            --activeDocHandlers[ eventType ];

            if ( !activeDocHandlers[ eventType ] ) {
                $document.unbind( realType, mouseEventCallback );
            }

            if ( eventCaptureSupported ) {
```

```
                // If this is the last virtual mouse binding in existence,
                // remove our document touchstart listener.

                --activeDocHandlers[ "touchstart" ];

                if ( !activeDocHandlers[ "touchstart" ] ) {
                    $document.unbind( "touchstart", handleTouchStart )
                        .unbind( "touchmove", handleTouchMove )
                        .unbind( "touchend", handleTouchEnd )
                        .unbind( "scroll", handleScroll );
                }
            }

            var $this = $( this ),
                bindings = $.data( this, dataPropertyName );

            // teardown may be called when an element was
            // removed from the DOM. If this is the case,
            // jQuery core may have already stripped the element
            // of any data bindings so we need to check it before
            // using it.
            if ( bindings ) {
                bindings[ eventType ] = false;
            }

            // Unregister the dummy event handler.

            $this.unbind( realType, dummyMouseHandler );

            // If this is the last virtual mouse binding on the
            // element, remove the binding data from the element.

            if ( !hasVirtualBindings( this ) ) {
                $this.removeData( dataPropertyName );
            }
        }
    };
}

// Expose our custom events to the jQuery bind/unbind mechanism.

for ( i = 0; i < virtualEventNames.length; i++ ) {
    $.event.special[ virtualEventNames[ i ] ] = getSpecialEventObject( virtualEventNames[ i ] );
}

// Add a capture click handler to block clicks.
// Note that we require event capture support for this so if the device
// doesn't support it, we punt for now and rely solely on mouse events.
if ( eventCaptureSupported ) {
    document.addEventListener( "click", function( e ) {
        var cnt = clickBlockList.length,
            target = e.target,
            x, y, ele, i, o, touchID;
```

```
        if ( cnt ) {
            x = e.clientX;
            y = e.clientY;
            threshold = $.vmouse.clickDistanceThreshold;

            // The idea here is to run through the clickBlockList to see if
            // the current click event is in the proximity of one of our
            // vclick events that had preventDefault() called on it. If we find
            // one, then we block the click.
            //
            // Why do we have to rely on proximity?
            //
            // Because the target of the touch event that triggered the vclick
            // can be different from the target of the click event synthesized
            // by the browser. The target of a mouse/click event that is synthesized
            // from a touch event seems to be implementation specific. For example,
            // some browsers will fire mouse/click events for a link that is near
            // a touch event, even though the target of the touchstart/touchend event
            // says the user touched outside the link. Also, it seems that with most
            // browsers, the target of the mouse/click event is not calculated until the
            // time it is dispatched, so if you replace an element that you touched
            // with another element, the target of the mouse/click will be the new
            // element underneath that point.
            //
            // Aside from proximity, we also check to see if the target and any
            // of its ancestors were the ones that blocked a click. This is necessary
            // because of the strange mouse/click target calculation done in the
            // Android 2.1 browser, where if you click on an element, and there is a
            // mouse/click handler on one of its ancestors, the target will be the
            // innermost child of the touched element, even if that child is no where
            // near the point of touch.

            ele = target;

            while ( ele ) {
                for ( i = 0; i < cnt; i++ ) {
                    o = clickBlockList[ i ];
                    touchID = 0;

                    if ( ( ele === target && Math.abs( o.x - x ) < threshold && Math.abs( o.y -
                    y ) < threshold ) ||
                                $.data( ele, touchTargetPropertyName ) === o.touchID ) {
                        // XXX: We may want to consider removing matches from the block list
                        //      instead of waiting for the reset timer to fire.
                        e.preventDefault();
                        e.stopPropagation();
                        return;
                    }
                }
                ele = ele.parentNode;
            }
        }
    }, true);
}
```

```
})( jQuery, window, document );


(function( $, window, undefined ) {
    var $document = $( document ),
        supportTouch = $.mobile.support.touch,
        scrollEvent = "touchmove scroll",
        touchStartEvent = supportTouch ? "touchstart" : "mousedown",
        touchStopEvent = supportTouch ? "touchend" : "mouseup",
        touchMoveEvent = supportTouch ? "touchmove" : "mousemove";

    // setup new event shortcuts
    $.each( ( "touchstart touchmove touchend " +
        "tap taphold " +
        "swipe swipeleft swiperight " +
        "scrollstart scrollstop" ).split( " " ), function( i, name ) {

        $.fn[ name ] = function( fn ) {
            return fn ? this.bind( name, fn ) : this.trigger( name );
        };

        // jQuery < 1.8
        if ( $.attrFn ) {
            $.attrFn[ name ] = true;
        }
    });

    function triggerCustomEvent( obj, eventType, event, bubble ) {
        var originalType = event.type;
        event.type = eventType;
        if ( bubble ) {
            $.event.trigger( event, undefined, obj );
        } else {
            $.event.dispatch.call( obj, event );
        }
        event.type = originalType;
    }

    // also handles scrollstop
    $.event.special.scrollstart = {

        enabled: true,
        setup: function() {

            var thisObject = this,
                $this = $( thisObject ),
                scrolling,
                timer;

            function trigger( event, state ) {
                scrolling = state;
                triggerCustomEvent( thisObject, scrolling ? "scrollstart" : "scrollstop", event
                );
            }
```

```
        // iPhone triggers scroll after a small delay; use touchmove instead
        $this.bind( scrollEvent, function( event ) {

            if ( !$.event.special.scrollstart.enabled ) {
                return;
            }

            if ( !scrolling ) {
                trigger( event, true );
            }

            clearTimeout( timer );
            timer = setTimeout( function() {
                trigger( event, false );
            }, 50 );
        });
    },
    teardown: function() {
        $( this ).unbind( scrollEvent );
    }
};

// also handles taphold
$.event.special.tap = {
    tapholdThreshold: 750,
    emitTapOnTaphold: true,
    setup: function() {
        var thisObject = this,
            $this = $( thisObject ),
            isTaphold = false;

        $this.bind( "vmousedown", function( event ) {
            isTaphold = false;
            if ( event.which && event.which !== 1 ) {
                return false;
            }

            var origTarget = event.target,
                timer;

            function clearTapTimer() {
                clearTimeout( timer );
            }

            function clearTapHandlers() {
                clearTapTimer();

                $this.unbind( "vclick", clickHandler )
                    .unbind( "vmouseup", clearTapTimer );
                $document.unbind( "vmousecancel", clearTapHandlers );
            }

            function clickHandler( event ) {
```

```
                clearTapHandlers();

                // ONLY trigger a 'tap' event if the start target is
                // the same as the stop target.
                if ( !isTaphold && origTarget === event.target ) {
                    triggerCustomEvent( thisObject, "tap", event );
                } else if ( isTaphold ) {
                    event.stopPropagation();
                }
            }

            $this.bind( "vmouseup", clearTapTimer )
                .bind( "vclick", clickHandler );
            $document.bind( "vmousecancel", clearTapHandlers );

            timer = setTimeout( function() {
                if ( !$.event.special.tap.emitTapOnTaphold ) {
                    isTaphold = true;
                }
                triggerCustomEvent( thisObject, "taphold", $.Event( "taphold", { target:
                origTarget } ) );
            }, $.event.special.tap.tapholdThreshold );
        });
    },
    teardown: function() {
        $( this ).unbind( "vmousedown" ).unbind( "vclick" ).unbind( "vmouseup" );
        $document.unbind( "vmousecancel" );
    }
};

// Also handles swipeleft, swiperight
$.event.special.swipe = {

    // More than this horizontal displacement, and we will suppress scrolling.
    scrollSupressionThreshold: 30,

    // More time than this, and it isn't a swipe.
    durationThreshold: 1000,

    // Swipe horizontal displacement must be more than this.
    horizontalDistanceThreshold: 30,

    // Swipe vertical displacement must be less than this.
    verticalDistanceThreshold: 30,

    getLocation: function ( event ) {
        var winPageX = window.pageXOffset,
            winPageY = window.pageYOffset,
            x = event.clientX,
            y = event.clientY;

        if ( event.pageY === 0 && Math.floor( y ) > Math.floor( event.pageY ) ||
            event.pageX === 0 && Math.floor( x ) > Math.floor( event.pageX ) ) {
```

```
            // iOS4 clientX/clientY have the value that should have been
            // in pageX/pageY. While pageX/page/ have the value 0
            x = x - winPageX;
            y = y - winPageY;
        } else if ( y < ( event.pageY - winPageY) || x < ( event.pageX - winPageX ) ) {

            // Some Android browsers have totally bogus values for clientX/Y
            // when scrolling/zooming a page. Detectable since clientX/clientY
            // should never be smaller than pageX/pageY minus page scroll
            x = event.pageX - winPageX;
            y = event.pageY - winPageY;
        }

        return {
            x: x,
            y: y
        };
    },

    start: function( event ) {
        var data = event.originalEvent.touches ?
                event.originalEvent.touches[ 0 ] : event,
            location = $.event.special.swipe.getLocation( data );
        return {
                    time: ( new Date() ).getTime(),
                    coords: [ location.x, location.y ],
                    origin: $( event.target )
                };
    },

    stop: function( event ) {
        var data = event.originalEvent.touches ?
                event.originalEvent.touches[ 0 ] : event,
            location = $.event.special.swipe.getLocation( data );
        return {
                    time: ( new Date() ).getTime(),
                    coords: [ location.x, location.y ]
                };
    },

    handleSwipe: function( start, stop, thisObject, origTarget ) {
        if ( stop.time - start.time < $.event.special.swipe.durationThreshold &&
            Math.abs( start.coords[ 0 ] - stop.coords[ 0 ] ) > $.event.special.swipe.
            horizontalDistanceThreshold &&
            Math.abs( start.coords[ 1 ] - stop.coords[ 1 ] ) < $.event.special.swipe.
            verticalDistanceThreshold ) {
            var direction = start.coords[0] > stop.coords[ 0 ] ? "swipeleft" : "swiperight";

            triggerCustomEvent( thisObject, "swipe", $.Event( "swipe", { target: origTarget,
             swipestart: start, swipestop: stop }), true );
            triggerCustomEvent( thisObject, direction,$.Event( direction, { target:
            origTarget, swipestart: start, swipestop: stop } ), true );
            return true;
        }
```

```
            return false;

        },

        // This serves as a flag to ensure that at most one swipe event event is
        // in work at any given time
        eventInProgress: false,

        setup: function() {
            var events,
                thisObject = this,
                $this = $( thisObject ),
                context = {};

            // Retrieve the events data for this element and add the swipe context
            events = $.data( this, "mobile-events" );
            if ( !events ) {
                events = { length: 0 };
                $.data( this, "mobile-events", events );
            }
            events.length++;
            events.swipe = context;

            context.start = function( event ) {

                // Bail if we're already working on a swipe event
                if ( $.event.special.swipe.eventInProgress ) {
                    return;
                }
                $.event.special.swipe.eventInProgress = true;

                var stop,
                    start = $.event.special.swipe.start( event ),
                    origTarget = event.target,
                    emitted = false;

                context.move = function( event ) {
                    if ( !start ) {
                        return;
                    }

                    stop = $.event.special.swipe.stop( event );
                    if ( !emitted ) {
                        emitted = $.event.special.swipe.handleSwipe( start, stop, thisObject,
                        origTarget );
                        if ( emitted ) {

                            // Reset the context to make way for the next swipe event
                            $.event.special.swipe.eventInProgress = false;
                        }
                    }
                    // prevent scrolling
                    if ( Math.abs( start.coords[ 0 ] - stop.coords[ 0 ] ) > $.event.special.
                    swipe.scrollSupressionThreshold ) {
```

```
                    event.preventDefault();
                }
            };

            context.stop = function() {
                    emitted = true;

                    // Reset the context to make way for the next swipe event
                    $.event.special.swipe.eventInProgress = false;
                    $document.off( touchMoveEvent, context.move );
                    context.move = null;
            };

            $document.on( touchMoveEvent, context.move )
                .one( touchStopEvent, context.stop );
        };
        $this.on( touchStartEvent, context.start );
    },

    teardown: function() {
        var events, context;

        events = $.data( this, "mobile-events" );
        if ( events ) {
            context = events.swipe;
            delete events.swipe;
            events.length--;
            if ( events.length === 0 ) {
                $.removeData( this, "mobile-events" );
            }
        }

        if ( context ) {
            if ( context.start ) {
                $( this ).off( touchStartEvent, context.start );
            }
            if ( context.move ) {
                $document.off( touchMoveEvent, context.move );
            }
            if ( context.stop ) {
                $document.off( touchStopEvent, context.stop );
            }
        }
    }
};
$.each({
    scrollstop: "scrollstart",
    taphold: "tap",
    swipeleft: "swipe",
    swiperight: "swipe"
}, function( event, sourceEvent ) {

    $.event.special[ event ] = {
        setup: function() {
```

```
                $( this ).bind( sourceEvent, $.noop );
            },
            teardown: function() {
                $( this ).unbind( sourceEvent );
            }
        };
    });

})( jQuery, this );


    // throttled resize event
    (function( $ ) {
        $.event.special.throttledresize = {
            setup: function() {
                $( this ).bind( "resize", handler );
            },
            teardown: function() {
                $( this ).unbind( "resize", handler );
            }
        };

        var throttle = 250,
            handler = function() {
                curr = ( new Date() ).getTime();
                diff = curr - lastCall;

                if ( diff >= throttle ) {

                    lastCall = curr;
                    $( this ).trigger( "throttledresize" );

                } else {

                    if ( heldCall ) {
                        clearTimeout( heldCall );
                    }

                    // Promise a held call will still execute
                    heldCall = setTimeout( handler, throttle - diff );
                }
            },
            lastCall = 0,
            heldCall,
            curr,
            diff;
    })( jQuery );


(function( $, window ) {
    var win = $( window ),
        event_name = "orientationchange",
        get_orientation,
        last_orientation,
```

```
        initial_orientation_is_landscape,
        initial_orientation_is_default,
        portrait_map = { "0": true, "180": true },
        ww, wh, landscape_threshold;

    // It seems that some device/browser vendors use window.orientation values 0 and 180 to
    // denote the "default" orientation. For iOS devices, and most other smart-phones tested,
    // the default orientation is always "portrait", but in some Android and RIM based tablets,
    // the default orientation is "landscape". The following code attempts to use the window
    // dimensions to figure out what the current orientation is, and then makes adjustments
    // to the to the portrait_map if necessary, so that we can properly decode the
    // window.orientation value whenever get_orientation() is called.
    //
    // Note that we used to use a media query to figure out what the orientation the browser
    // thinks it is in:
    //
    //     initial_orientation_is_landscape = $.mobile.media("all and (orientation: landscape)");
    //
    // but there was an iPhone/iPod Touch bug beginning with iOS 4.2, up through iOS 5.1,
    // where the browser *ALWAYS* applied the landscape media query. This bug does not
    // happen on iPad.

    if ( $.support.orientation ) {

        // Check the window width and height to figure out what the current orientation
        // of the device is at this moment. Note that we've initialized the portrait map
        // values to 0 and 180, *AND* we purposely check for landscape so that if we guess
        // wrong, , we default to the assumption that portrait is the default orientation.
        // We use a threshold check below because on some platforms like iOS, the iPhone
        // form-factor can report a larger width than height if the user turns on the
        // developer console. The actual threshold value is somewhat arbitrary, we just
        // need to make sure it is large enough to exclude the developer console case.

        ww = window.innerWidth || win.width();
        wh = window.innerHeight || win.height();
        landscape_threshold = 50;

        initial_orientation_is_landscape = ww > wh && ( ww - wh ) > landscape_threshold;

        // Now check to see if the current window.orientation is 0 or 180.
        initial_orientation_is_default = portrait_map[ window.orientation ];

        // If the initial orientation is landscape, but window.orientation reports 0 or 180, *OR*
        // if the initial orientation is portrait, but window.orientation reports 90 or -90, we
        // need to flip our portrait_map values because landscape is the default orientation for
        // this device/browser.
        if ( ( initial_orientation_is_landscape && initial_orientation_is_default ) || ( !
        initial_orientation_is_landscape && !initial_orientation_is_default ) ) {
            portrait_map = { "-90": true, "90": true };
        }
    }

    $.event.special.orientationchange = $.extend( {}, $.event.special.orientationchange, {
        setup: function() {
```

```
            // If the event is supported natively, return false so that jQuery
            // will bind to the event using DOM methods.
            if ( $.support.orientation && !$.event.special.orientationchange.disabled ) {
                return false;
            }

            // Get the current orientation to avoid initial double-triggering.
            last_orientation = get_orientation();

            // Because the orientationchange event doesn't exist, simulate the
            // event by testing window dimensions on resize.
            win.bind( "throttledresize", handler );
        },
        teardown: function() {
            // If the event is not supported natively, return false so that
            // jQuery will unbind the event using DOM methods.
            if ( $.support.orientation && !$.event.special.orientationchange.disabled ) {
                return false;
            }

            // Because the orientationchange event doesn't exist, unbind the
            // resize event handler.
            win.unbind( "throttledresize", handler );
        },
        add: function( handleObj ) {
            // Save a reference to the bound event handler.
            var old_handler = handleObj.handler;

            handleObj.handler = function( event ) {
                // Modify event object, adding the .orientation property.
                event.orientation = get_orientation();

                // Call the originally-bound event handler and return its result.
                return old_handler.apply( this, arguments );
            };
        }
    });

    // If the event is not supported natively, this handler will be bound to
    // the window resize event to simulate the orientationchange event.
    function handler() {
        // Get the current orientation.
        var orientation = get_orientation();

        if ( orientation !== last_orientation ) {
            // The orientation has changed, so trigger the orientationchange event.
            last_orientation = orientation;
            win.trigger( event_name );
        }
    }

    // Get the current page orientation. This method is exposed publicly, should it
    // be needed, as jQuery.event.special.orientationchange.orientation()
    $.event.special.orientationchange.orientation = get_orientation = function() {
```

```
        var isPortrait = true, elem = document.documentElement;

        // prefer window orientation to the calculation based on screensize as
        // the actual screen resize takes place before or after the orientation change event
        // has been fired depending on implementation (eg android 2.3 is before, iphone after).
        // More testing is required to determine if a more reliable method of determining the
        new screensize
        // is possible when orientationchange is fired. (eg, use media queries + element +
        opacity)
        if ( $.support.orientation ) {
            // if the window orientation registers as 0 or 180 degrees report
            // portrait, otherwise landscape
            isPortrait = portrait_map[ window.orientation ];
        } else {
            isPortrait = elem && elem.clientWidth / elem.clientHeight < 1.1;
        }

        return isPortrait ? "portrait" : "landscape";
    };

    $.fn[ event_name ] = function( fn ) {
        return fn ? this.bind( event_name, fn ) : this.trigger( event_name );
    };

    // jQuery < 1.8
    if ( $.attrFn ) {
        $.attrFn[ event_name ] = true;
    }

}( jQuery, this ));




(function( $, undefined ) {

    // existing base tag?
    var baseElement = $( "head" ).children( "base" ),

    // base element management, defined depending on dynamic base tag support
    // TODO move to external widget
    base = {

        // define base element, for use in routing asset urls that are referenced
        // in Ajax-requested markup
        element: ( baseElement.length ? baseElement :
            $( "<base>", { href: $.mobile.path.documentBase.hrefNoHash } ).prependTo( $( "head"
            ) ) ),

        linkSelector: "[src], link[href], a[rel='external'], :jqmData(ajax='false'), a[target]",

        // set the generated BASE element's href to a new page's base path
        set: function( href ) {
```

```
            // we should do nothing if the user wants to manage their url base
            // manually
            if ( !$.mobile.dynamicBaseEnabled ) {
                return;
            }

            // we should use the base tag if we can manipulate it dynamically
            if ( $.support.dynamicBaseTag ) {
                base.element.attr( "href",
                    $.mobile.path.makeUrlAbsolute( href, $.mobile.path.documentBase ) );
            }
        },

        rewrite: function( href, page ) {
            var newPath = $.mobile.path.get( href );

            page.find( base.linkSelector ).each(function( i, link ) {
                var thisAttr = $( link ).is( "[href]" ) ? "href" :
                    $( link ).is( "[src]" ) ? "src" : "action",
                thisUrl = $( link ).attr( thisAttr );

                // XXX_jblas: We need to fix this so that it removes the document
                //            base URL, and then prepends with the new page URL.
                // if full path exists and is same, chop it - helps IE out
                thisUrl = thisUrl.replace( location.protocol + "//" +
                    location.host + location.pathname, "" );

                if ( !/^(\w+:|#|\/)/.test( thisUrl ) ) {
                    $( link ).attr( thisAttr, newPath + thisUrl );
                }
            });
        },

        // set the generated BASE element's href to a new page's base path
        reset: function(/* href */) {
            base.element.attr( "href", $.mobile.path.documentBase.hrefNoSearch );
        }
    };

    $.mobile.base = base;

})( jQuery );


(function( $, undefined ) {
$.mobile.widgets = {};

var originalWidget = $.widget,

    // Record the original, non-mobileinit-modified version of $.mobile.keepNative
    // so we can later determine whether someone has modified $.mobile.keepNative
    keepNativeFactoryDefault = $.mobile.keepNative;

$.widget = (function( orig ) {
```

```
    return function() {
        var constructor = orig.apply( this, arguments ),
            name = constructor.prototype.widgetName;

        constructor.initSelector = ( ( constructor.prototype.initSelector !== undefined ) ?
            constructor.prototype.initSelector : ":jqmData(role='" + name + "')" );

        $.mobile.widgets[ name ] = constructor;

        return constructor;
    };
})( $.widget );

// Make sure $.widget still has bridge and extend methods
$.extend( $.widget, originalWidget );

// For backcompat remove in 1.5
$.mobile.document.on( "create", function( event ) {
    $( event.target ).enhanceWithin();
});

$.widget( "mobile.page", {
    options: {
        theme: "a",
        domCache: false,

        // Deprecated in 1.4 remove in 1.5
        keepNativeDefault: $.mobile.keepNative,

        // Deprecated in 1.4 remove in 1.5
        contentTheme: null,
        enhanced: false
    },

    // DEPRECATED for > 1.4
    // TODO remove at 1.5
    _createWidget: function() {
        $.Widget.prototype._createWidget.apply( this, arguments );
        this._trigger( "init" );
    },

    _create: function() {
        // If false is returned by the callbacks do not create the page
        if ( this._trigger( "beforecreate" ) === false ) {
            return false;
        }

        if ( !this.options.enhanced ) {
            this._enhance();
        }

        this._on( this.element, {
            pagebeforehide: "removeContainerBackground",
            pagebeforeshow: "_handlePageBeforeShow"
```

```
        });

        this.element.enhanceWithin();
        // Dialog widget is deprecated in 1.4 remove this in 1.5
        if ( $.mobile.getAttribute( this.element[0], "role" ) === "dialog" && $.mobile.dialog ) {
            this.element.dialog();
        }
    },

    _enhance: function () {
        var attrPrefix = "data-" + $.mobile.ns,
            self = this;

        if ( this.options.role ) {
            this.element.attr( "data-" + $.mobile.ns + "role", this.options.role );
        }

        this.element
            .attr( "tabindex", "0" )
            .addClass( "ui-page ui-page-theme-" + this.options.theme );

        // Manipulation of content os Deprecated as of 1.4 remove in 1.5
        this.element.find( "[" + attrPrefix + "role='content']" ).each( function() {
            var $this = $( this ),
                theme = this.getAttribute( attrPrefix + "theme" ) || undefined;
                self.options.contentTheme = theme || self.options.contentTheme || ( self.options
                .dialog && self.options.theme ) || ( self.element.jqmData("role") === "dialog"
                &&  self.options.theme );
                $this.addClass( "ui-content" );
                if ( self.options.contentTheme ) {
                    $this.addClass( "ui-body-" + ( self.options.contentTheme ) );
                }
                // Add ARIA role
                $this.attr( "role", "main" ).addClass( "ui-content" );
        });
    },

    bindRemove: function( callback ) {
        var page = this.element;

        // when dom caching is not enabled or the page is embedded bind to remove the page on
        hide
        if ( !page.data( "mobile-page" ).options.domCache &&
            page.is( ":jqmData(external-page='true')" ) ) {

            // TODO use _on - that is, sort out why it doesn't work in this case
            page.bind( "pagehide.remove", callback || function( e, data ) {

                //check if this is a same page transition and if so don't remove the page
                if( !data.samePage ){
                    var $this = $( this ),
                        prEvent = new $.Event( "pageremove" );

                    $this.trigger( prEvent );
```

```
                if ( !prEvent.isDefaultPrevented() ) {
                    $this.removeWithDependents();
                }
            }
        });
    }
},

_setOptions: function( o ) {
    if ( o.theme !== undefined ) {
        this.element.removeClass( "ui-page-theme-" + this.options.theme ).addClass(
        "ui-page-theme-" + o.theme );
    }

    if ( o.contentTheme !== undefined ) {
        this.element.find( "[data-" + $.mobile.ns + "='content']" ).removeClass( "ui-body-"
        + this.options.contentTheme )
            .addClass( "ui-body-" + o.contentTheme );
    }
},

_handlePageBeforeShow: function(/* e */) {
    this.setContainerBackground();
},
// Deprecated in 1.4 remove in 1.5
removeContainerBackground: function() {
    this.element.closest( ":mobile-pagecontainer" ).pagecontainer({ "theme": "none" });
},
// Deprecated in 1.4 remove in 1.5
// set the page container background to the page theme
setContainerBackground: function( theme ) {
    this.element.parent().pagecontainer( { "theme": theme || this.options.theme } );
},
// Deprecated in 1.4 remove in 1.5
keepNativeSelector: function() {
    var options = this.options,
        keepNative = $.trim( options.keepNative || "" ),
        globalValue = $.trim( $.mobile.keepNative ),
        optionValue = $.trim( options.keepNativeDefault ),

        // Check if $.mobile.keepNative has changed from the factory default
        newDefault = ( keepNativeFactoryDefault === globalValue ?
            "" : globalValue ),

        // If $.mobile.keepNative has not changed, use options.keepNativeDefault
        oldDefault = ( newDefault === "" ? optionValue : "" );

    // Concatenate keepNative selectors from all sources where the value has
    // changed or, if nothing has changed, return the default
    return ( ( keepNative ? [ keepNative ] : [] )
        .concat( newDefault ? [ newDefault ] : [] )
        .concat( oldDefault ? [ oldDefault ] : [] )
        .join( ", " ) );
```

```
    }
});
})( jQuery );

(function( $, undefined ) {

    $.widget( "mobile.pagecontainer", {
        options: {
            theme: "a"
        },

        initSelector: false,

        _create: function() {
            this.setLastScrollEnabled = true;

            this._on( this.window, {
                // disable an scroll setting when a hashchange has been fired,
                // this only works because the recording of the scroll position
                // is delayed for 100ms after the browser might have changed the
                // position because of the hashchange
                navigate: "_disableRecordScroll",

                // bind to scrollstop for the first page, "pagechange" won't be
                // fired in that case
                scrollstop: "_delayedRecordScroll"
            });

            // TODO consider moving the navigation handler OUT of widget into
            //      some other object as glue between the navigate event and the
            //      content widget load and change methods
            this._on( this.window, { navigate: "_filterNavigateEvents" });

            // TODO move from page* events to content* events
            this._on({ pagechange: "_afterContentChange" });

            // handle initial hashchange from chrome :(
            this.window.one( "navigate", $.proxy(function() {
                this.setLastScrollEnabled = true;
            }, this));
        },

        _setOptions: function( options ) {
            if ( options.theme !== undefined && options.theme !== "none" ) {
                this.element.removeClass( "ui-overlay-" + this.options.theme )
                    .addClass( "ui-overlay-" + options.theme );
            } else if ( options.theme !== undefined ) {
                this.element.removeClass( "ui-overlay-" + this.options.theme );
            }

            this._super( options );
        },

        _disableRecordScroll: function() {
```

```
            this.setLastScrollEnabled = false;
        },

        _enableRecordScroll: function() {
            this.setLastScrollEnabled = true;
        },

        // TODO consider the name here, since it's purpose specific
        _afterContentChange: function() {
            // once the page has changed, re-enable the scroll recording
            this.setLastScrollEnabled = true;

            // remove any binding that previously existed on the get scroll
            // which may or may not be different than the scroll element
            // determined for this page previously
            this._off( this.window, "scrollstop" );

            // determine and bind to the current scoll element which may be the
            // window or in the case of touch overflow the element touch overflow
            this._on( this.window, { scrollstop: "_delayedRecordScroll" });
        },

        _recordScroll: function() {
            // this barrier prevents setting the scroll value based on
            // the browser scrolling the window based on a hashchange
            if ( !this.setLastScrollEnabled ) {
                return;
            }

            var active = this._getActiveHistory(),
                currentScroll, minScroll, defaultScroll;

            if ( active ) {
                currentScroll = this._getScroll();
                minScroll = this._getMinScroll();
                defaultScroll = this._getDefaultScroll();

                // Set active page's lastScroll prop. If the location we're
                // scrolling to is less than minScrollBack, let it go.
                active.lastScroll = currentScroll < minScroll ? defaultScroll : currentScroll;
            }
        },

        _delayedRecordScroll: function() {
            setTimeout( $.proxy(this, "_recordScroll"), 100 );
        },

        _getScroll: function() {
            return this.window.scrollTop();
        },

        _getMinScroll: function() {
            return $.mobile.minScrollBack;
        },
```

```
        _getDefaultScroll: function() {
            return $.mobile.defaultHomeScroll;
        },

        _filterNavigateEvents: function( e, data ) {
            var url;

            if ( e.originalEvent && e.originalEvent.isDefaultPrevented() ) {
                return;
            }

            url = e.originalEvent.type.indexOf( "hashchange" ) > -1 ? data.state.hash : data.
            state.url;

            if ( !url ) {
                url = this._getHash();
            }

            if ( !url || url === "#" || url.indexOf( "#" + $.mobile.path.uiStateKey ) === 0 ) {
                url = location.href;
            }

            this._handleNavigate( url, data.state );
        },

        _getHash: function() {
            return $.mobile.path.parseLocation().hash;
        },

        // TODO active page should be managed by the container (ie, it should be a property)
        getActivePage: function() {
            return this.activePage;
        },

        // TODO the first page should be a property set during _create using the logic
        //      that currently resides in init
        _getInitialContent: function() {
            return $.mobile.firstPage;
        },

        // TODO each content container should have a history object
        _getHistory: function() {
            return $.mobile.navigate.history;
        },

        // TODO use _getHistory
        _getActiveHistory: function() {
            return $.mobile.navigate.history.getActive();
        },

        // TODO the document base should be determined at creation
        _getDocumentBase: function() {
            return $.mobile.path.documentBase;
```

```
        },

        back: function() {
            this.go( -1 );
        },

        forward: function() {
            this.go( 1 );
        },

        go: function( steps ) {

            //if hashlistening is enabled use native history method
            if ( $.mobile.hashListeningEnabled ) {
                window.history.go( steps );
            } else {

                //we are not listening to the hash so handle history internally
                var activeIndex = $.mobile.navigate.history.activeIndex,
                    index = activeIndex + parseInt( steps, 10 ),
                    url = $.mobile.navigate.history.stack[ index ].url,
                    direction = ( steps >= 1 )? "forward" : "back";

                //update the history object
                $.mobile.navigate.history.activeIndex = index;
                $.mobile.navigate.history.previousIndex = activeIndex;

                //change to the new page
                this.change( url, { direction: direction, changeHash: false, fromHashChange:
                true } );
            }
        },

        // TODO rename _handleDestination
        _handleDestination: function( to ) {
            var history;

            // clean the hash for comparison if it's a url
            if ( $.type(to) === "string" ) {
                to = $.mobile.path.stripHash( to );
            }

            if ( to ) {
                history = this._getHistory();

                // At this point, 'to' can be one of 3 things, a cached page
                // element from a history stack entry, an id, or site-relative /
                // absolute URL. If 'to' is an id, we need to resolve it against
                // the documentBase, not the location.href, since the hashchange
                // could've been the result of a forward/backward navigation
                // that crosses from an external page/dialog to an internal
                // page/dialog.
                //
                // TODO move check to history object or path object?
```

```
                to = !$.mobile.path.isPath( to ) ? ( $.mobile.path.makeUrlAbsolute( "#" + to,
                this._getDocumentBase() ) ) : to;

                // If we're about to go to an initial URL that contains a
                // reference to a non-existent internal page, go to the first
                // page instead. We know that the initial hash refers to a
                // non-existent page, because the initial hash did not end
                // up in the initial history entry
                // TODO move check to history object?
                if ( to === $.mobile.path.makeUrlAbsolute( "#" + history.initialDst, this.
                _getDocumentBase() ) &&
                    history.stack.length &&
                    history.stack[0].url !== history.initialDst.replace( $.mobile.dialogHashKey,
                     "" ) ) {
                    to = this._getInitialContent();
                }
            }
            return to || this._getInitialContent();
        },

        _handleDialog: function( changePageOptions, data ) {
            var to, active, activeContent = this.getActivePage();

            // If current active page is not a dialog skip the dialog and continue
            // in the same direction
            if ( activeContent && !activeContent.hasClass( "ui-dialog" ) ) {
                // determine if we're heading forward or backward and continue
                // accordingly past the current dialog
                if ( data.direction === "back" ) {
                    this.back();
                } else {
                    this.forward();
                }

                // prevent changePage call
                return false;
            } else {
                // if the current active page is a dialog and we're navigating
                // to a dialog use the dialog objected saved in the stack
                to = data.pageUrl;
                active = this._getActiveHistory();

                // make sure to set the role, transition and reversal
                // as most of this is lost by the domCache cleaning
                $.extend( changePageOptions, {
                    role: active.role,
                    transition: active.transition,
                    reverse: data.direction === "back"
                });
            }

            return to;
        },
```

```
        _handleNavigate: function( url, data ) {
            //find first page via hash
            // TODO stripping the hash twice with handleUrl
            var to = $.mobile.path.stripHash( url ), history = this._getHistory(),

                // transition is false if it's the first page, undefined
                // otherwise (and may be overridden by default)
                transition = history.stack.length === 0 ? "none" : undefined,

                // default options for the changPage calls made after examining
                // the current state of the page and the hash, NOTE that the
                // transition is derived from the previous history entry
                changePageOptions = {
                    changeHash: false,
                    fromHashChange: true,
                    reverse: data.direction === "back"
                };

            $.extend( changePageOptions, data, {
                transition: ( history.getLast() || {} ).transition || transition
            });

            // TODO move to _handleDestination ?
            // If this isn't the first page, if the current url is a dialog hash
            // key, and the initial destination isn't equal to the current target
            // page, use the special dialog handling
            if ( history.activeIndex > 0 &&
                to.indexOf( $.mobile.dialogHashKey ) > -1 &&
                history.initialDst !== to ) {

                to = this._handleDialog( changePageOptions, data );

                if ( to === false ) {
                    return;
                }
            }

            this._changeContent( this._handleDestination( to ), changePageOptions );
        },

        _changeContent: function( to, opts ) {
            $.mobile.changePage( to, opts );
        },

        _getBase: function() {
            return $.mobile.base;
        },

        _getNs: function() {
            return $.mobile.ns;
        },

        _enhance: function( content, role ) {
            // TODO consider supporting a custom callback, and passing in
```

```
            // the settings which includes the role
            return content.page({ role: role });
        },

        _include: function( page, settings ) {
            // append to page and enhance
            page.appendTo( this.element );

            // use the page widget to enhance
            this._enhance( page, settings.role );

            // remove page on hide
            page.page( "bindRemove" );
        },

        _find: function( absUrl ) {
            // TODO consider supporting a custom callback
            var fileUrl = this._createFileUrl( absUrl ),
                dataUrl = this._createDataUrl( absUrl ),
                page, initialContent = this._getInitialContent();

            // Check to see if the page already exists in the DOM.
            // NOTE do _not_ use the :jqmData pseudo selector because parenthesis
            //      are a valid url char and it breaks on the first occurence
            page = this.element
                .children( "[data-" + this._getNs() +"url='" + dataUrl + "']" );

            // If we failed to find the page, check to see if the url is a
            // reference to an embedded page. If so, it may have been dynamically
            // injected by a developer, in which case it would be lacking a
            // data-url attribute and in need of enhancement.
            if ( page.length === 0 && dataUrl && !$.mobile.path.isPath( dataUrl ) ) {
                page = this.element.children( $.mobile.path.hashToSelector("#" + dataUrl) )
                    .attr( "data-" + this._getNs() + "url", dataUrl )
                    .jqmData( "url", dataUrl );
            }

            // If we failed to find a page in the DOM, check the URL to see if it
            // refers to the first page in the application. Also check to make sure
            // our cached-first-page is actually in the DOM. Some user deployed
            // apps are pruning the first page from the DOM for various reasons.
            // We check for this case here because we don't want a first-page with
            // an id falling through to the non-existent embedded page error case.
            if ( page.length === 0 &&
                $.mobile.path.isFirstPageUrl( fileUrl ) &&
                initialContent &&
                initialContent.parent().length ) {
                page = $( initialContent );
            }

            return page;
        },

        _getLoader: function() {
```

```
            return $.mobile.loading();
        },

        _showLoading: function( delay, theme, msg, textonly ) {
            // This configurable timeout allows cached pages a brief
            // delay to load without showing a message
            if ( this._loadMsg ) {
                return;
            }

            this._loadMsg = setTimeout($.proxy(function() {
                this._getLoader().loader( "show", theme, msg, textonly );
                this._loadMsg = 0;
            }, this), delay );
        },

        _hideLoading: function() {
            // Stop message show timer
            clearTimeout( this._loadMsg );
            this._loadMsg = 0;

            // Hide loading message
            this._getLoader().loader( "hide" );
        },

        _showError: function() {
            // make sure to remove the current loading message
            this._hideLoading();

            // show the error message
            this._showLoading( 0, $.mobile.pageLoadErrorMessageTheme, $.mobile.
            pageLoadErrorMessage, true );

            // hide the error message after a delay
            // TODO configuration
            setTimeout( $.proxy(this, "_hideLoading"), 1500 );
        },

        _parse: function( html, fileUrl ) {
            // TODO consider allowing customization of this method. It's very JQM specific
            var page, all = $( "<div></div>" );

            //workaround to allow scripts to execute when included in page divs
            all.get( 0 ).innerHTML = html;

            page = all.find( ":jqmData(role='page'), :jqmData(role='dialog')" ).first();

            //if page elem couldn't be found, create one and insert the body element's contents
            if ( !page.length ) {
                page = $( "<div data-" + this._getNs() + "role='page'>" +
                    ( html.split( /<\/?body[^>]*>/gmi )[1] || "" ) +
                    "</div>" );
            }
```

```
            // TODO tagging a page with external to make sure that embedded pages aren't
            // removed by the various page handling code is bad. Having page handling code
            // in many places is bad. Solutions post 1.0
            page.attr( "data-" + this._getNs() + "url", $.mobile.path.convertUrlToDataUrl(
            fileUrl) )
                .attr( "data-" + this._getNs() + "external-page", true );

            return page;
        },

        _setLoadedTitle: function( page, html ) {
            //page title regexp
            var newPageTitle = html.match( /<title[^>]*>([^<]*)/ ) && RegExp.$1;

            if ( newPageTitle && !page.jqmData("title") ) {
                newPageTitle = $( "<div>" + newPageTitle + "</div>" ).text();
                page.jqmData( "title", newPageTitle );
            }
        },

        _isRewritableBaseTag: function() {
            return $.mobile.dynamicBaseEnabled && !$.support.dynamicBaseTag;
        },

        _createDataUrl: function( absoluteUrl ) {
            return $.mobile.path.convertUrlToDataUrl( absoluteUrl );
        },

        _createFileUrl: function( absoluteUrl ) {
            return $.mobile.path.getFilePath( absoluteUrl );
        },

        _triggerWithDeprecated: function( name, data, page ) {
            var deprecatedEvent = $.Event( "page" + name ),
                newEvent = $.Event( this.widgetName + name );

            // DEPRECATED
            // trigger the old deprecated event on the page if it's provided
            ( page || this.element ).trigger( deprecatedEvent, data );

            // use the widget trigger method for the new content* event
            this.element.trigger( newEvent, data );

            return {
                deprecatedEvent: deprecatedEvent,
                event: newEvent
            };
        },

        // TODO it would be nice to split this up more but everything appears to be "one off"
        //      or require ordering such that other bits are sprinkled in between parts that
        //      could be abstracted out as a group
        _loadSuccess: function( absUrl, triggerData, settings, deferred ) {
            var fileUrl = this._createFileUrl( absUrl ),
```

```
            dataUrl = this._createDataUrl( absUrl );

        return $.proxy(function( html, textStatus, xhr ) {
            //pre-parse html to check for a data-url,
            //use it as the new fileUrl, base path, etc
            var content,

                // TODO handle dialogs again
                pageElemRegex = new RegExp( "(<[^>]+\\bdata-" + this._getNs() +
                "role=[\"']?page[\"']?[^>]*>)" ),

                dataUrlRegex = new RegExp( "\\bdata-" + this._getNs() +
                "url=[\"']?([^\"'>]*)[\"']?" );

            // data-url must be provided for the base tag so resource requests
            // can be directed to the correct url. loading into a temprorary
            // element makes these requests immediately
            if ( pageElemRegex.test( html ) &&
                RegExp.$1 &&
                dataUrlRegex.test( RegExp.$1 ) &&
                RegExp.$1 ) {
                fileUrl = $.mobile.path.getFilePath( $("<div>" + RegExp.$1 + "</div>").text
                () );
            }

            //dont update the base tag if we are prefetching
            if ( settings.prefetch === undefined ) {
                this._getBase().set( fileUrl );
            }

            content = this._parse( html, fileUrl );

            this._setLoadedTitle( content, html );

            // Add the content reference and xhr to our triggerData.
            triggerData.xhr = xhr;
            triggerData.textStatus = textStatus;

            // DEPRECATED
            triggerData.page = content;

            triggerData.content = content;

            // If the default behavior is prevented, stop here!
            // Note that it is the responsibility of the listener/handler
            // that called preventDefault(), to resolve/reject the
            // deferred object within the triggerData.
            if ( !this._trigger( "load", undefined, triggerData ) ) {
                return;
            }

            // rewrite src and href attrs to use a base url if the base tag won't work
            if ( this._isRewritableBaseTag() && content ) {
                this._getBase().rewrite( fileUrl, content );
```

```
            }

            this._include( content, settings );

            // Enhancing the content may result in new dialogs/sub content being inserted
            // into the DOM. If the original absUrl refers to a sub-content, that is the
            // real content we are interested in.
            if ( absUrl.indexOf( "&" + $.mobile.subPageUrlKey ) > -1 ) {
                content = this.element.children( "[data-" + this._getNs() +"url='" + dataUrl
                + "']" );
            }

            // Remove loading message.
            if ( settings.showLoadMsg ) {
                this._hideLoading();
            }

            // BEGIN DEPRECATED ----------------------------------------------------
            // Let listeners know the content loaded successfully.
            this.element.trigger( "pageload" );
            // END DEPRECATED ------------------------------------------------------

            deferred.resolve( absUrl, settings, content );
        }, this);
    },

    _loadDefaults: {
        type: "get",
        data: undefined,

        // DEPRECATED
        reloadPage: false,

        reload: false,

        // By default we rely on the role defined by the @data-role attribute.
        role: undefined,

        showLoadMsg: false,

        // This delay allows loads that pull from browser cache to
        // occur without showing the loading message.
        loadMsgDelay: 50
    },

    load: function( url, options ) {
        // This function uses deferred notifications to let callers
        // know when the content is done loading, or if an error has occurred.
        var deferred = ( options && options.deferred ) || $.Deferred(),

            // The default load options with overrides specified by the caller.
            settings = $.extend( {}, this._loadDefaults, options ),

            // The DOM element for the content after it has been loaded.
```

```
            content = null,

            // The absolute version of the URL passed into the function. This
            // version of the URL may contain dialog/subcontent params in it.
            absUrl = $.mobile.path.makeUrlAbsolute( url, this._findBaseWithDefault() ),
            fileUrl, dataUrl, pblEvent, triggerData;

        // DEPRECATED reloadPage
        settings.reload = settings.reloadPage;

        // If the caller provided data, and we're using "get" request,
        // append the data to the URL.
        if ( settings.data && settings.type === "get" ) {
            absUrl = $.mobile.path.addSearchParams( absUrl, settings.data );
            settings.data = undefined;
        }

        // If the caller is using a "post" request, reload must be true
        if ( settings.data && settings.type === "post" ) {
            settings.reload = true;
        }

        // The absolute version of the URL minus any dialog/subcontent params.
        // In otherwords the real URL of the content to be loaded.
        fileUrl = this._createFileUrl( absUrl );

        // The version of the Url actually stored in the data-url attribute of
        // the content. For embedded content, it is just the id of the page. For
        // content within the same domain as the document base, it is the site
        // relative path. For cross-domain content (Phone Gap only) the entire
        // absolute Url is used to load the content.
        dataUrl = this._createDataUrl( absUrl );

        content = this._find( absUrl );

        // If it isn't a reference to the first content and refers to missing
        // embedded content reject the deferred and return
        if ( content.length === 0 &&
            $.mobile.path.isEmbeddedPage(fileUrl) &&
            !$.mobile.path.isFirstPageUrl(fileUrl) ) {
            deferred.reject( absUrl, settings );
            return;
        }

        // Reset base to the default document base
        // TODO figure out why we doe this
        this._getBase().reset();

        // If the content we are interested in is already in the DOM,
        // and the caller did not indicate that we should force a
        // reload of the file, we are done. Resolve the deferrred so that
        // users can bind to .done on the promise
        if ( content.length && !settings.reload ) {
            this._enhance( content, settings.role );
```

```
                deferred.resolve( absUrl, settings, content );

                //if we are reloading the content make sure we update
                // the base if its not a prefetch
                if ( !settings.prefetch ) {
                    this._getBase().set(url);
                }

                return;
            }

            triggerData = {
                url: url,
                absUrl: absUrl,
                dataUrl: dataUrl,
                deferred: deferred,
                options: settings
            };

            // Let listeners know we're about to load content.
            pblEvent = this._triggerWithDeprecated( "beforeload", triggerData );

            // If the default behavior is prevented, stop here!
            if ( pblEvent.deprecatedEvent.isDefaultPrevented() ||
                pblEvent.event.isDefaultPrevented() ) {
                return;
            }

            if ( settings.showLoadMsg ) {
                this._showLoading( settings.loadMsgDelay );
            }

            // Reset base to the default document base.
            // only reset if we are not prefetching
            if ( settings.prefetch === undefined ) {
                this._getBase().reset();
            }

            if ( !( $.mobile.allowCrossDomainPages ||
                $.mobile.path.isSameDomain($.mobile.path.documentUrl, absUrl ) ) ) {
                deferred.reject( absUrl, settings );
                return;
            }

            // Load the new content.
            $.ajax({
                url: fileUrl,
                type: settings.type,
                data: settings.data,
                contentType: settings.contentType,
                dataType: "html",
                success: this._loadSuccess( absUrl, triggerData, settings, deferred ),
                error: this._loadError( absUrl, triggerData, settings, deferred )
            });
```

```
        },

        _loadError: function( absUrl, triggerData, settings, deferred ) {
            return $.proxy(function( xhr, textStatus, errorThrown ) {
                //set base back to current path
                this._getBase().set( $.mobile.path.get() );

                // Add error info to our triggerData.
                triggerData.xhr = xhr;
                triggerData.textStatus = textStatus;
                triggerData.errorThrown = errorThrown;

                // Let listeners know the page load failed.
                var plfEvent = this._triggerWithDeprecated( "loadfailed", triggerData );

                // If the default behavior is prevented, stop here!
                // Note that it is the responsibility of the listener/handler
                // that called preventDefault(), to resolve/reject the
                // deferred object within the triggerData.
                if ( plfEvent.deprecatedEvent.isDefaultPrevented() ||
                    plfEvent.event.isDefaultPrevented() ) {
                    return;
                }

                // Remove loading message.
                if ( settings.showLoadMsg ) {
                    this._showError();
                }

                deferred.reject( absUrl, settings );
            }, this);
        },

        _getTransitionHandler: function( transition ) {
            transition = $.mobile._maybeDegradeTransition( transition );

            //find the transition handler for the specified transition. If there
            //isn't one in our transitionHandlers dictionary, use the default one.
            //call the handler immediately to kick-off the transition.
            return $.mobile.transitionHandlers[ transition ] || $.mobile.
            defaultTransitionHandler;
        },

        // TODO move into transition handlers?
        _triggerCssTransitionEvents: function( to, from, prefix ) {
            var samePage = false;

            prefix = prefix || "";

            // TODO decide if these events should in fact be triggered on the container
            if ( from ) {

                //Check if this is a same page transition and tell the handler in page
                if( to[0] === from[0] ){
```

```
                samePage = true;
            }

            //trigger before show/hide events
            // TODO deprecate nextPage in favor of next
            this._triggerWithDeprecated( prefix + "hide", { nextPage: to, samePage: samePage
             }, from );
        }

        // TODO deprecate prevPage in favor of previous
        this._triggerWithDeprecated( prefix + "show", { prevPage: from || $( "" ) }, to );
    },

    // TODO make private once change has been defined in the widget
    _cssTransition: function( to, from, options ) {
        var transition = options.transition,
            reverse = options.reverse,
            deferred = options.deferred,
            TransitionHandler,
            promise;

        this._triggerCssTransitionEvents( to, from, "before" );

        // TODO put this in a binding to events *outside* the widget
        this._hideLoading();

        TransitionHandler = this._getTransitionHandler( transition );

        promise = ( new TransitionHandler( transition, reverse, to, from ) ).transition();

        // TODO temporary accomodation of argument deferred
        promise.done(function() {
            deferred.resolve.apply( deferred, arguments );
        });

        promise.done($.proxy(function() {
            this._triggerCssTransitionEvents( to, from );
        }, this));
    },

    _releaseTransitionLock: function() {
        //release transition lock so navigation is free again
        isPageTransitioning = false;
        if ( pageTransitionQueue.length > 0 ) {
            $.mobile.changePage.apply( null, pageTransitionQueue.pop() );
        }
    },

    _removeActiveLinkClass: function( force ) {
        //clear out the active button state
        $.mobile.removeActiveLinkClass( force );
    },

    _loadUrl: function( to, triggerData, settings ) {
```

```
            // preserve the original target as the dataUrl value will be
            // simplified eg, removing ui-state, and removing query params
            // from the hash this is so that users who want to use query
            // params have access to them in the event bindings for the page
            // life cycle See issue #5085
            settings.target = to;
            settings.deferred = $.Deferred();

            this.load( to, settings );

            settings.deferred.done($.proxy(function( url, options, content ) {
                isPageTransitioning = false;

                // store the original absolute url so that it can be provided
                // to events in the triggerData of the subsequent changePage call
                options.absUrl = triggerData.absUrl;

                this.transition( content, triggerData, options );
            }, this));

            settings.deferred.fail($.proxy(function(/* url, options */) {
                this._removeActiveLinkClass( true );
                this._releaseTransitionLock();
                this._triggerWithDeprecated( "changefailed", triggerData );
            }, this));
        },

        _triggerPageBeforeChange: function( to, triggerData, settings ) {
            var pbcEvent = new $.Event( "pagebeforechange" );

            $.extend(triggerData, { toPage: to, options: settings });

            // NOTE: preserve the original target as the dataUrl value will be
            // simplified eg, removing ui-state, and removing query params from
            // the hash this is so that users who want to use query params have
            // access to them in the event bindings for the page life cycle
            // See issue #5085
            if ( $.type(to) === "string" ) {
                // if the toPage is a string simply convert it
                triggerData.absUrl = $.mobile.path.makeUrlAbsolute( to, this.
                    _findBaseWithDefault() );
            } else {
                // if the toPage is a jQuery object grab the absolute url stored
                // in the loadPage callback where it exists
                triggerData.absUrl = settings.absUrl;
            }

            // Let listeners know we're about to change the current page.
            this.element.trigger( pbcEvent, triggerData );

            // If the default behavior is prevented, stop here!
            if ( pbcEvent.isDefaultPrevented() ) {
                return false;
            }
```

```
        return true;
    },

    change: function( to, options ) {
        // If we are in the midst of a transition, queue the current request.
        // We'll call changePage() once we're done with the current transition
        // to service the request.
        if ( isPageTransitioning ) {
            pageTransitionQueue.unshift( arguments );
            return;
        }

        var settings = $.extend( {}, $.mobile.changePage.defaults, options ),
            triggerData = {};

        // Make sure we have a fromPage.
        settings.fromPage = settings.fromPage || this.activePage;

        // if the page beforechange default is prevented return early
        if ( !this._triggerPageBeforeChange(to, triggerData, settings) ) {
            return;
        }

        // We allow "pagebeforechange" observers to modify the to in
        // the trigger data to allow for redirects. Make sure our to is
        // updated. We also need to re-evaluate whether it is a string,
        // because an object can also be replaced by a string
        to = triggerData.toPage;

        // If the caller passed us a url, call loadPage()
        // to make sure it is loaded into the DOM. We'll listen
        // to the promise object it returns so we know when
        // it is done loading or if an error ocurred.
        if ( $.type(to) === "string" ) {
            // Set the isPageTransitioning flag to prevent any requests from
            // entering this method while we are in the midst of loading a page
            // or transitioning.
            isPageTransitioning = true;

            this._loadUrl( to, triggerData, settings );
        } else {
            this.transition( to, triggerData, settings );
        }
    },

    transition: function( toPage, triggerData, settings ) {
        var fromPage, url, pageUrl, fileUrl,
            active, activeIsInitialPage,
            historyDir, pageTitle, isDialog,
            alreadyThere, newPageTitle,
            params, cssTransitionDeferred,
            beforeTransition;
```

```
            // If we are in the midst of a transition, queue the current request.
            // We'll call changePage() once we're done with the current transition
            // to service the request.
            if ( isPageTransitioning ) {
                // make sure to only queue the to and settings values so the arguments
                // work with a call to the change method
                pageTransitionQueue.unshift( [toPage, settings] );
                return;
            }

            // DEPRECATED - this call only, in favor of the before transition
            // if the page beforechange default is prevented return early
            if ( !this._triggerPageBeforeChange(toPage, triggerData, settings) ) {
                return;
            }

            // if the (content|page)beforetransition default is prevented return early
            // Note, we have to check for both the deprecated and new events
            beforeTransition = this._triggerWithDeprecated( "beforetransition", triggerData );
            if (beforeTransition.deprecatedEvent.isDefaultPrevented() ||
                beforeTransition.event.isDefaultPrevented() ) {
                return;
            }

            // Set the isPageTransitioning flag to prevent any requests from
            // entering this method while we are in the midst of loading a page
            // or transitioning.
            isPageTransitioning = true;

            // If we are going to the first-page of the application, we need to make
            // sure settings.dataUrl is set to the application document url. This allows
            // us to avoid generating a document url with an id hash in the case where the
            // first-page of the document has an id attribute specified.
            if ( toPage[ 0 ] === $.mobile.firstPage[ 0 ] && !settings.dataUrl ) {
                settings.dataUrl = $.mobile.path.documentUrl.hrefNoHash;
            }

            // The caller passed us a real page DOM element. Update our
            // internal state and then trigger a transition to the page.
            fromPage = settings.fromPage;
            url = ( settings.dataUrl && $.mobile.path.convertUrlToDataUrl(settings.dataUrl) ) ||
                toPage.jqmData( "url" );

            // The pageUrl var is usually the same as url, except when url is obscured
            // as a dialog url. pageUrl always contains the file path
            pageUrl = url;
            fileUrl = $.mobile.path.getFilePath( url );
            active = $.mobile.navigate.history.getActive();
            activeIsInitialPage = $.mobile.navigate.history.activeIndex === 0;
            historyDir = 0;
            pageTitle = document.title;
            isDialog = ( settings.role === "dialog" ||
                toPage.jqmData( "role" ) === "dialog" ) &&
                toPage.jqmData( "dialog" ) !== true;
```

```
        // By default, we prevent changePage requests when the fromPage and toPage
        // are the same element, but folks that generate content
        // manually/dynamically and reuse pages want to be able to transition to
        // the same page. To allow this, they will need to change the default
        // value of allowSamePageTransition to true, *OR*, pass it in as an
        // option when they manually call changePage(). It should be noted that
        // our default transition animations assume that the formPage and toPage
        // are different elements, so they may behave unexpectedly. It is up to
        // the developer that turns on the allowSamePageTransitiona option to
        // either turn off transition animations, or make sure that an appropriate
        // animation transition is used.
        if ( fromPage && fromPage[0] === toPage[0] &&
            !settings.allowSamePageTransition ) {

            isPageTransitioning = false;
            this._triggerWithDeprecated( "transition", triggerData );
            this.element.trigger( "pagechange", triggerData );

            // Even if there is no page change to be done, we should keep the
            // urlHistory in sync with the hash changes
            if ( settings.fromHashChange ) {
                $.mobile.navigate.history.direct({ url: url });
            }

            return;
        }

        // We need to make sure the page we are given has already been enhanced.
        toPage.page({ role: settings.role });

        // If the changePage request was sent from a hashChange event, check to
        // see if the page is already within the urlHistory stack. If so, we'll
        // assume the user hit the forward/back button and will try to match the
        // transition accordingly.
        if ( settings.fromHashChange ) {
            historyDir = settings.direction === "back" ? -1 : 1;
        }

        // Kill the keyboard.
        // XXX_jblas: We need to stop crawling the entire document to kill focus.
        //            Instead, we should be tracking focus with a delegate()
        //            handler so we already have the element in hand at this
        //            point.
        // Wrap this in a try/catch block since IE9 throw "Unspecified error" if
        // document.activeElement is undefined when we are in an IFrame.
        try {
            if ( document.activeElement &&
                document.activeElement.nodeName.toLowerCase() !== "body" ) {

                $( document.activeElement ).blur();
            } else {
                $( "input:focus, textarea:focus, select:focus" ).blur();
            }
```

```
            } catch( e ) {}

            // Record whether we are at a place in history where a dialog used to be -
            // if so, do not add a new history entry and do not change the hash either
            alreadyThere = false;

            // If we're displaying the page as a dialog, we don't want the url
            // for the dialog content to be used in the hash. Instead, we want
            // to append the dialogHashKey to the url of the current page.
            if ( isDialog && active ) {
                // on the initial page load active.url is undefined and in that case
                // should be an empty string. Moving the undefined -> empty string back
                // into urlHistory.addNew seemed imprudent given undefined better
                // represents the url state

                // If we are at a place in history that once belonged to a dialog, reuse
                // this state without adding to urlHistory and without modifying the
                // hash. However, if a dialog is already displayed at this point, and
                // we're about to display another dialog, then we must add another hash
                // and history entry on top so that one may navigate back to the
                // original dialog
                if ( active.url &&
                    active.url.indexOf( $.mobile.dialogHashKey ) > -1 &&
                    this.activePage &&
                    !this.activePage.hasClass( "ui-dialog" ) &&
                    $.mobile.navigate.history.activeIndex > 0 ) {

                    settings.changeHash = false;
                    alreadyThere = true;
                }

                // Normally, we tack on a dialog hash key, but if this is the location
                // of a stale dialog, we reuse the URL from the entry
                url = ( active.url || "" );

                // account for absolute urls instead of just relative urls use as hashes
                if ( !alreadyThere && url.indexOf("#") > -1 ) {
                    url += $.mobile.dialogHashKey;
                } else {
                    url += "#" + $.mobile.dialogHashKey;
                }

                // tack on another dialogHashKey if this is the same as the initial hash
                // this makes sure that a history entry is created for this dialog
                if ( $.mobile.navigate.history.activeIndex === 0 && url === $.mobile.navigate.
                history.initialDst ) {
                    url += $.mobile.dialogHashKey;
                }
            }

            // if title element wasn't found, try the page div data attr too
            // If this is a deep-link or a reload ( active === undefined ) then just
            // use pageTitle
            newPageTitle = ( !active ) ? pageTitle : toPage.jqmData( "title" ) ||
```

```
            toPage.children( ":jqmData(role='header')" ).find( ".ui-title" ).text();
        if ( !!newPageTitle && pageTitle === document.title ) {
            pageTitle = newPageTitle;
        }
        if ( !toPage.jqmData( "title" ) ) {
            toPage.jqmData( "title", pageTitle );
        }

        // Make sure we have a transition defined.
        settings.transition = settings.transition ||
            ( ( historyDir && !activeIsInitialPage ) ? active.transition : undefined ) ||
            ( isDialog ? $.mobile.defaultDialogTransition : $.mobile.defaultPageTransition );

        //add page to history stack if it's not back or forward
        if ( !historyDir && alreadyThere ) {
            $.mobile.navigate.history.getActive().pageUrl = pageUrl;
        }

        // Set the location hash.
        if ( url && !settings.fromHashChange ) {

            // rebuilding the hash here since we loose it earlier on
            // TODO preserve the originally passed in path
            if ( !$.mobile.path.isPath( url ) && url.indexOf( "#" ) < 0 ) {
                url = "#" + url;
            }

            // TODO the property names here are just silly
            params = {
                transition: settings.transition,
                title: pageTitle,
                pageUrl: pageUrl,
                role: settings.role
            };

            if ( settings.changeHash !== false && $.mobile.hashListeningEnabled ) {
                $.mobile.navigate( url, params, true);
            } else if ( toPage[ 0 ] !== $.mobile.firstPage[ 0 ] ) {
                $.mobile.navigate.history.add( url, params );
            }
        }

        //set page title
        document.title = pageTitle;

        //set "toPage" as activePage deprecated in 1.4 remove in 1.5
        $.mobile.activePage = toPage;

        //new way to handle activePage
        this.activePage = toPage;

        // If we're navigating back in the URL history, set reverse accordingly.
        settings.reverse = settings.reverse || historyDir < 0;
```

```
            cssTransitionDeferred = $.Deferred();

            this._cssTransition(toPage, fromPage, {
                transition: settings.transition,
                reverse: settings.reverse,
                deferred: cssTransitionDeferred
            });

            cssTransitionDeferred.done($.proxy(function( name, reverse, $to, $from,
            alreadyFocused ) {
                $.mobile.removeActiveLinkClass();

                //if there's a duplicateCachedPage, remove it from the DOM now that it's hidden
                if ( settings.duplicateCachedPage ) {
                    settings.duplicateCachedPage.remove();
                }

                // despite visibility: hidden addresses issue #2965
                // https://github.com/jquery/jquery-mobile/issues/2965
                if ( !alreadyFocused ) {
                    $.mobile.focusPage( toPage );
                }

                this._releaseTransitionLock();
                this.element.trigger( "pagechange", triggerData );
                this._triggerWithDeprecated( "transition", triggerData );
            }, this));
        },

        // determine the current base url
        _findBaseWithDefault: function() {
            var closestBase = ( this.activePage &&
            $.mobile.getClosestBaseUrl( this.activePage ) );
        return closestBase || $.mobile.path.documentBase.hrefNoHash;
        }
    });

    // The following handlers should be bound after mobileinit has been triggered
    // the following deferred is resolved in the init file
    $.mobile.navreadyDeferred = $.Deferred();

    //these variables make all page containers use the same queue and only navigate one at a time
    // queue to hold simultanious page transitions
    var pageTransitionQueue = [],

        // indicates whether or not page is in process of transitioning
        isPageTransitioning = false;

})( jQuery );

(function( $, undefined ) {

        // resolved on domready
    var domreadyDeferred = $.Deferred(),
```

```
        // resolved and nulled on window.load()
        loadDeferred = $.Deferred(),
        documentUrl = $.mobile.path.documentUrl,

        // used to track last vclicked element to make sure its value is added to form data
        $lastVClicked = null;

    /* Event Bindings - hashchange, submit, and click */
    function findClosestLink( ele ) {
        while ( ele ) {
            // Look for the closest element with a nodeName of "a".
            // Note that we are checking if we have a valid nodeName
            // before attempting to access it. This is because the
            // node we get called with could have originated from within
            // an embedded SVG document where some symbol instance elements
            // don't have nodeName defined on them, or strings are of type
            // SVGAnimatedString.
            if ( ( typeof ele.nodeName === "string" ) && ele.nodeName.toLowerCase() === "a" ) {
                break;
            }
            ele = ele.parentNode;
        }
        return ele;
    }

    $.mobile.loadPage = function( url, opts ) {
        var container;

        opts = opts || {};
        container = ( opts.pageContainer || $.mobile.pageContainer );

        // create the deferred that will be supplied to loadPage callers
        // and resolved by the content widget's load method
        opts.deferred = $.Deferred();

        // Preferring to allow exceptions for uninitialized opts.pageContainer
        // widgets so we know if we need to force init here for users
        container.pagecontainer( "load", url, opts );

        // provide the deferred
        return opts.deferred.promise();
    };

    //define vars for interal use

    /* internal utility functions */

    // NOTE Issue #4950 Android phonegap doesn't navigate back properly
    //      when a full page refresh has taken place. It appears that hashchange
    //      and replacestate history alterations work fine but we need to support
    //      both forms of history traversal in our code that uses backward history
    //      movement
    $.mobile.back = function() {
```

```
        var nav = window.navigator;

        // if the setting is on and the navigator object is
        // available use the phonegap navigation capability
        if ( this.phonegapNavigationEnabled &&
            nav &&
            nav.app &&
            nav.app.backHistory ) {
            nav.app.backHistory();
        } else {
            $.mobile.pageContainer.pagecontainer( "back" );
        }
    };

    // Direct focus to the page title, or otherwise first focusable element
    $.mobile.focusPage = function ( page ) {
        var autofocus = page.find( "[autofocus]" ),
            pageTitle = page.find( ".ui-title:eq(0)" );

        if ( autofocus.length ) {
            autofocus.focus();
            return;
        }

        if ( pageTitle.length ) {
            pageTitle.focus();
        } else{
            page.focus();
        }
    };

    // No-op implementation of transition degradation
    $.mobile._maybeDegradeTransition = $.mobile._maybeDegradeTransition || function( transition
    ) {
        return transition;
    };

    // Exposed $.mobile methods

    $.mobile.changePage = function( to, options ) {
        $.mobile.pageContainer.pagecontainer( "change", to, options );
    };

    $.mobile.changePage.defaults = {
        transition: undefined,
        reverse: false,
        changeHash: true,
        fromHashChange: false,
        role: undefined, // By default we rely on the role defined by the @data-role attribute.
        duplicateCachedPage: undefined,
        pageContainer: undefined,
        showLoadMsg: true, //loading message shows by default when pages are being fetched
        during changePage
        dataUrl: undefined,
```

```
        fromPage: undefined,
        allowSamePageTransition: false
    };

    $.mobile._registerInternalEvents = function() {
        var getAjaxFormData = function( $form, calculateOnly ) {
            var url, ret = true, formData, vclickedName, method;
            if ( !$.mobile.ajaxEnabled ||
                    // test that the form is, itself, ajax false
                    $form.is( ":jqmData(ajax='false')" ) ||
                    // test that $.mobile.ignoreContentEnabled is set and
                    // the form or one of it's parents is ajax=false
                    !$form.jqmHijackable().length ||
                    $form.attr( "target" ) ) {
                return false;
            }

            url = ( $lastVClicked && $lastVClicked.attr( "formaction" ) ) ||
                $form.attr( "action" );
            method = ( $form.attr( "method" ) || "get" ).toLowerCase();

            // If no action is specified, browsers default to using the
            // URL of the document containing the form. Since we dynamically
            // pull in pages from external documents, the form should submit
            // to the URL for the source document of the page containing
            // the form.
            if ( !url ) {
                // Get the @data-url for the page containing the form.
                url = $.mobile.getClosestBaseUrl( $form );

                // NOTE: If the method is "get", we need to strip off the query string
                // because it will get replaced with the new form data. See issue #5710.
                if ( method === "get" ) {
                    url = $.mobile.path.parseUrl( url ).hrefNoSearch;
                }

                if ( url === $.mobile.path.documentBase.hrefNoHash ) {
                    // The url we got back matches the document base,
                    // which means the page must be an internal/embedded page,
                    // so default to using the actual document url as a browser
                    // would.
                    url = documentUrl.hrefNoSearch;
                }
            }

            url = $.mobile.path.makeUrlAbsolute(  url, $.mobile.getClosestBaseUrl( $form ) );

            if ( ( $.mobile.path.isExternal( url ) && !$.mobile.path.
            isPermittedCrossDomainRequest( documentUrl, url ) ) ) {
                return false;
            }

            if ( !calculateOnly ) {
                formData = $form.serializeArray();
```

```
            if ( $lastVClicked && $lastVClicked[ 0 ].form === $form[ 0 ] ) {
                vclickedName = $lastVClicked.attr( "name" );
                if ( vclickedName ) {
                    // Make sure the last clicked element is included in the form
                    $.each( formData, function( key, value ) {
                        if ( value.name === vclickedName ) {
                            // Unset vclickedName - we've found it in the serialized data
                            already
                            vclickedName = "";
                            return false;
                        }
                    });
                    if ( vclickedName ) {
                        formData.push( { name: vclickedName, value: $lastVClicked.attr(
                        "value" ) } );
                    }
                }
            }

            ret = {
                url: url,
                options: {
                    type:       method,
                    data:       $.param( formData ),
                    transition: $form.jqmData( "transition" ),
                    reverse:    $form.jqmData( "direction" ) === "reverse",
                    reloadPage: true
                }
            };
        }

        return ret;
    };

    //bind to form submit events, handle with Ajax
    $.mobile.document.delegate( "form", "submit", function( event ) {
        var formData;

        if ( !event.isDefaultPrevented() ) {
            formData = getAjaxFormData( $( this ) );
            if ( formData ) {
                $.mobile.changePage( formData.url, formData.options );
                event.preventDefault();
            }
        }
    });

    //add active state on vclick
    $.mobile.document.bind( "vclick", function( event ) {
        var $btn, btnEls, target = event.target, needClosest = false;
        // if this isn't a left click we don't care. Its important to note
        // that when the virtual event is generated it will create the which attr
        if ( event.which > 1 || !$.mobile.linkBindingEnabled ) {
```

```
            return;
        }

        // Record that this element was clicked, in case we need it for correct
        // form submission during the "submit" handler above
        $lastVClicked = $( target );

        // Try to find a target element to which the active class will be applied
        if ( $.data( target, "mobile-button" ) ) {
            // If the form will not be submitted via AJAX, do not add active class
            if ( !getAjaxFormData( $( target ).closest( "form" ), true ) ) {
                return;
            }
            // We will apply the active state to this button widget - the parent
            // of the input that was clicked will have the associated data
            if ( target.parentNode ) {
                target = target.parentNode;
            }
        } else {
            target = findClosestLink( target );
            if ( !( target && $.mobile.path.parseUrl( target.getAttribute( "href" ) || "#"
            ).hash !== "#" ) ) {
                return;
            }

            // TODO teach $.mobile.hijackable to operate on raw dom elements so the
            // link wrapping can be avoided
            if ( !$( target ).jqmHijackable().length ) {
                return;
            }
        }

        // Avoid calling .closest by using the data set during .buttonMarkup()
        // List items have the button data in the parent of the element clicked
        if ( !!~target.className.indexOf( "ui-link-inherit" ) ) {
            if ( target.parentNode ) {
                btnEls = $.data( target.parentNode, "buttonElements" );
            }
        // Otherwise, look for the data on the target itself
        } else {
            btnEls = $.data( target, "buttonElements" );
        }
        // If found, grab the button's outer element
        if ( btnEls ) {
            target = btnEls.outer;
        } else {
            needClosest = true;
        }

        $btn = $( target );
        // If the outer element wasn't found by the our heuristics, use .closest()
        if ( needClosest ) {
            $btn = $btn.closest( ".ui-btn" );
        }
```

```
        if ( $btn.length > 0 &&
            !( $btn.hasClass( "ui-state-disabled" ||

                // DEPRECATED as of 1.4.0 - remove after 1.4.0 release
                // only ui-state-disabled should be present thereafter
                $btn.hasClass( "ui-disabled" ) ) ) ) {
            $.mobile.removeActiveLinkClass( true );
            $.mobile.activeClickedLink = $btn;
            $.mobile.activeClickedLink.addClass( $.mobile.activeBtnClass );
        }
    });

    // click routing - direct to HTTP or Ajax, accordingly
    $.mobile.document.bind( "click", function( event ) {
        if ( !$.mobile.linkBindingEnabled || event.isDefaultPrevented() ) {
            return;
        }

        var link = findClosestLink( event.target ),
            $link = $( link ),

            //remove active link class if external (then it won't be there if you come back)
            httpCleanup = function() {
                window.setTimeout(function() { $.mobile.removeActiveLinkClass( true ); },
                200 );
            },
            baseUrl, href,
            useDefaultUrlHandling, isExternal,
            transition, reverse, role;

        // If a button was clicked, clean up the active class added by vclick above
        if ( $.mobile.activeClickedLink &&
            $.mobile.activeClickedLink[ 0 ] === event.target.parentNode ) {
            httpCleanup();
        }

        // If there is no link associated with the click or its not a left
        // click we want to ignore the click
        // TODO teach $.mobile.hijackable to operate on raw dom elements so the link wrapping
        // can be avoided
        if ( !link || event.which > 1 || !$link.jqmHijackable().length ) {
            return;
        }

        //if there's a data-rel=back attr, go back in history
        if ( $link.is( ":jqmData(rel='back')" ) ) {
            $.mobile.back();
            return false;
        }

        baseUrl = $.mobile.getClosestBaseUrl( $link );

        //get href, if defined, otherwise default to empty hash
```

```
            href = $.mobile.path.makeUrlAbsolute( $link.attr( "href" ) || "#", baseUrl );

            //if ajax is disabled, exit early
            if ( !$.mobile.ajaxEnabled && !$.mobile.path.isEmbeddedPage( href ) ) {
                httpCleanup();
                //use default click handling
                return;
            }

            // XXX_jblas: Ideally links to application pages should be specified as
            //            an url to the application document with a hash that is either
            //            the site relative path or id to the page. But some of the
            //            internal code that dynamically generates sub-pages for nested
            //            lists and select dialogs, just write a hash in the link they
            //            create. This means the actual URL path is based on whatever
            //            the current value of the base tag is at the time this code
            //            is called. For now we are just assuming that any url with a
            //            hash in it is an application page reference.
            if ( href.search( "#" ) !== -1 ) {
                href = href.replace( /[^#]*#/, "" );
                if ( !href ) {
                    //link was an empty hash meant purely
                    //for interaction, so we ignore it.
                    event.preventDefault();
                    return;
                } else if ( $.mobile.path.isPath( href ) ) {
                    //we have apath so make it the href we want to load.
                    href = $.mobile.path.makeUrlAbsolute( href, baseUrl );
                } else {
                    //we have a simple id so use the documentUrl as its base.
                    href = $.mobile.path.makeUrlAbsolute( "#" + href, documentUrl.hrefNoHash );
                }
            }

            // Should we handle this link, or let the browser deal with it?
            useDefaultUrlHandling = $link.is( "[rel='external']" ) || $link.is(
            ":jqmData(ajax='false')" ) || $link.is( "[target]" );

            // Some embedded browsers, like the web view in Phone Gap, allow cross-domain XHR
            // requests if the document doing the request was loaded via the file:// protocol.
            // This is usually to allow the application to "phone home" and fetch app specific
            // data. We normally let the browser handle external/cross-domain urls, but if the
            // allowCrossDomainPages option is true, we will allow cross-domain http/https
            // requests to go through our page loading logic.

            //check for protocol or rel and its not an embedded page
            //TODO overlap in logic from isExternal, rel=external check should be
            //     moved into more comprehensive isExternalLink
            isExternal = useDefaultUrlHandling || ( $.mobile.path.isExternal( href ) && !$.
            mobile.path.isPermittedCrossDomainRequest( documentUrl, href ) );

            if ( isExternal ) {
                httpCleanup();
                //use default click handling
```

```
                return;
            }

            //use ajax
            transition = $link.jqmData( "transition" );
            reverse = $link.jqmData( "direction" ) === "reverse" ||
                        // deprecated - remove by 1.0
                        $link.jqmData( "back" );

            //this may need to be more specific as we use data-rel more
            role = $link.attr( "data-" + $.mobile.ns + "rel" ) || undefined;

            $.mobile.changePage( href, { transition: transition, reverse: reverse, role: role,
            link: $link } );
            event.preventDefault();
        });

        //prefetch pages when anchors with data-prefetch are encountered
        $.mobile.document.delegate( ".ui-page", "pageshow.prefetch", function() {
            var urls = [];
            $( this ).find( "a:jqmData(prefetch)" ).each(function() {
                var $link = $( this ),
                    url = $link.attr( "href" );

                if ( url && $.inArray( url, urls ) === -1 ) {
                    urls.push( url );

                    $.mobile.loadPage( url, { role: $link.attr( "data-" + $.mobile.ns + "rel" ),
                    prefetch: true } );
                }
            });
        });

        // TODO ensure that the navigate binding in the content widget happens at the right time
        $.mobile.pageContainer.pagecontainer();

        //set page min-heights to be device specific
        $.mobile.document.bind( "pageshow", function() {

            // We need to wait for window.load to make sure that styles have already been
            rendered,
            // otherwise heights of external toolbars will have the wrong value
            if ( loadDeferred ) {
                loadDeferred.done( $.mobile.resetActivePageHeight );
            } else {
                $.mobile.resetActivePageHeight();
            }
        });
        $.mobile.window.bind( "throttledresize", $.mobile.resetActivePageHeight );

    };//navreadyDeferred done callback

    $( function() { domreadyDeferred.resolve(); } );
```

```
    $.mobile.window.load( function() {

        // Resolve and null the deferred
        loadDeferred.resolve();
        loadDeferred = null;
    });

    $.when( domreadyDeferred, $.mobile.navreadyDeferred ).done( function() { $.mobile.
    _registerInternalEvents(); } );
})( jQuery );


(function( $, window, undefined ) {

    // TODO remove direct references to $.mobile and properties, we should
    //      favor injection with params to the constructor
    $.mobile.Transition = function() {
        this.init.apply( this, arguments );
    };

    $.extend($.mobile.Transition.prototype, {
        toPreClass: " ui-page-pre-in",

        init: function( name, reverse, $to, $from ) {
            $.extend(this, {
                name: name,
                reverse: reverse,
                $to: $to,
                $from: $from,
                deferred: new $.Deferred()
            });
        },

        cleanFrom: function() {
            this.$from
                .removeClass( $.mobile.activePageClass + " out in reverse " + this.name )
                .height( "" );
        },

        // NOTE overridden by child object prototypes, noop'd here as defaults
        beforeDoneIn: function() {},
        beforeDoneOut: function() {},
        beforeStartOut: function() {},

        doneIn: function() {
            this.beforeDoneIn();

            this.$to.removeClass( "out in reverse " + this.name ).height( "" );

            this.toggleViewportClass();

            // In some browsers (iOS5), 3D transitions block the ability to scroll to the
            desired location during transition
            // This ensures we jump to that spot after the fact, if we aren't there already.
```

```
            if ( $.mobile.window.scrollTop() !== this.toScroll ) {
                this.scrollPage();
            }
            if ( !this.sequential ) {
                this.$to.addClass( $.mobile.activePageClass );
            }
            this.deferred.resolve( this.name, this.reverse, this.$to, this.$from, true );
        },

        doneOut: function( screenHeight, reverseClass, none, preventFocus ) {
            this.beforeDoneOut();
            this.startIn( screenHeight, reverseClass, none, preventFocus );
        },

        hideIn: function( callback ) {
            // Prevent flickering in phonegap container: see comments at #4024 regarding iOS
            this.$to.css( "z-index", -10 );
            callback.call( this );
            this.$to.css( "z-index", "" );
        },

        scrollPage: function() {
            // By using scrollTo instead of silentScroll, we can keep things better in order
            // Just to be precautios, disable scrollstart listening like silentScroll would
            $.event.special.scrollstart.enabled = false;
            //if we are hiding the url bar or the page was previously scrolled scroll to hide
            or return to position
            if ( $.mobile.hideUrlBar || this.toScroll !== $.mobile.defaultHomeScroll ) {
                window.scrollTo( 0, this.toScroll );
            }

            // reenable scrollstart listening like silentScroll would
            setTimeout( function() {
                $.event.special.scrollstart.enabled = true;
            }, 150 );
        },

        startIn: function( screenHeight, reverseClass, none, preventFocus ) {
            this.hideIn(function() {
                this.$to.addClass( $.mobile.activePageClass + this.toPreClass );

                // Send focus to page as it is now display: block
                if ( !preventFocus ) {
                    $.mobile.focusPage( this.$to );
                }

                // Set to page height
                this.$to.height( screenHeight + this.toScroll );

                if ( !none ) {
                    this.scrollPage();
                }
            });
```

```
            this.$to
                .removeClass( this.toPreClass )
                .addClass( this.name + " in " + reverseClass );

            if ( !none ) {
                this.$to.animationComplete( $.proxy(function() {
                    this.doneIn();
                }, this ));
            } else {
                this.doneIn();
            }

        },

        startOut: function( screenHeight, reverseClass, none ) {
            this.beforeStartOut( screenHeight, reverseClass, none );

            // Set the from page's height and start it transitioning out
            // Note: setting an explicit height helps eliminate tiling in the transitions
            this.$from
                .height( screenHeight + $.mobile.window.scrollTop() )
                .addClass( this.name + " out" + reverseClass );
        },

        toggleViewportClass: function() {
            $.mobile.pageContainer.toggleClass( "ui-mobile-viewport-transitioning viewport-" +
            this.name );
        },

        transition: function() {
            // NOTE many of these could be calculated/recorded in the constructor, it's my
            //      opinion that binding them as late as possible has value with regards to
            //      better transitions with fewer bugs. Ie, it's not guaranteed that the
            //      object will be created and transition will be run immediately after as
            //      it is today. So we wait until transition is invoked to gather the following
            var none,
                reverseClass = this.reverse ? " reverse" : "",
                screenHeight = $.mobile.getScreenHeight(),
                maxTransitionOverride = $.mobile.maxTransitionWidth !== false &&
                    $.mobile.window.width() > $.mobile.maxTransitionWidth;

            this.toScroll = $.mobile.navigate.history.getActive().lastScroll || $.mobile.
            defaultHomeScroll;

            none = !$.support.cssTransitions || !$.support.cssAnimations ||
                maxTransitionOverride || !this.name || this.name === "none" ||
                Math.max( $.mobile.window.scrollTop(), this.toScroll ) >
                    $.mobile.getMaxScrollForTransition();

            this.toggleViewportClass();

            if ( this.$from && !none ) {
                this.startOut( screenHeight, reverseClass, none );
            } else {
```

```
                this.doneOut( screenHeight, reverseClass, none, true );
            }

            return this.deferred.promise();
        }
    });
})( jQuery, this );


(function( $ ) {

    $.mobile.SerialTransition = function() {
        this.init.apply(this, arguments);
    };

    $.extend($.mobile.SerialTransition.prototype, $.mobile.Transition.prototype, {
        sequential: true,

        beforeDoneOut: function() {
            if ( this.$from ) {
                this.cleanFrom();
            }
        },

        beforeStartOut: function( screenHeight, reverseClass, none ) {
            this.$from.animationComplete($.proxy(function() {
                this.doneOut( screenHeight, reverseClass, none );
            }, this ));
        }
    });

})( jQuery );


(function( $ ) {

    $.mobile.ConcurrentTransition = function() {
        this.init.apply(this, arguments);
    };

    $.extend($.mobile.ConcurrentTransition.prototype, $.mobile.Transition.prototype, {
        sequential: false,

        beforeDoneIn: function() {
            if ( this.$from ) {
                this.cleanFrom();
            }
        },

        beforeStartOut: function( screenHeight, reverseClass, none ) {
            this.doneOut( screenHeight, reverseClass, none );
        }
    });
```

```
})( jQuery );


(function( $ ) {

    // generate the handlers from the above
    var defaultGetMaxScrollForTransition = function() {
        return $.mobile.getScreenHeight() * 3;
    };

    //transition handler dictionary for 3rd party transitions
    $.mobile.transitionHandlers = {
        "sequential": $.mobile.SerialTransition,
        "simultaneous": $.mobile.ConcurrentTransition
    };

    // Make our transition handler the public default.
    $.mobile.defaultTransitionHandler = $.mobile.transitionHandlers.sequential;

    $.mobile.transitionFallbacks = {};

    // If transition is defined, check if css 3D transforms are supported, and if not, if a
    fallback is specified
    $.mobile._maybeDegradeTransition = function( transition ) {
        if ( transition && !$.support.cssTransform3d && $.mobile.transitionFallbacks[ transition
         ] ) {
            transition = $.mobile.transitionFallbacks[ transition ];
        }

        return transition;
    };

    // Set the getMaxScrollForTransition to default if no implementation was set by user
    $.mobile.getMaxScrollForTransition = $.mobile.getMaxScrollForTransition ||
    defaultGetMaxScrollForTransition;

})( jQuery );

/*
* fallback transition for flip in non-3D supporting browsers (which tend to handle complex
transitions poorly in general
*/

(function( $, window, undefined ) {

$.mobile.transitionFallbacks.flip = "fade";

})( jQuery, this );

/*
* fallback transition for flow in non-3D supporting browsers (which tend to handle complex
transitions poorly in general
*/
```

```
(function( $, window, undefined ) {

$.mobile.transitionFallbacks.flow = "fade";

})( jQuery, this );

/*
* fallback transition for pop in non-3D supporting browsers (which tend to handle complex
transitions poorly in general
*/

(function( $, window, undefined ) {

$.mobile.transitionFallbacks.pop = "fade";

})( jQuery, this );

/*
* fallback transition for slide in non-3D supporting browsers (which tend to handle complex
transitions poorly in general
*/

(function( $, window, undefined ) {

// Use the simultaneous transitions handler for slide transitions
$.mobile.transitionHandlers.slide = $.mobile.transitionHandlers.simultaneous;

// Set the slide transitions's fallback to "fade"
$.mobile.transitionFallbacks.slide = "fade";

})( jQuery, this );

/*
* fallback transition for slidedown in non-3D supporting browsers (which tend to handle complex
transitions poorly in general
*/

(function( $, window, undefined ) {

$.mobile.transitionFallbacks.slidedown = "fade";

})( jQuery, this );

/*
* fallback transition for slidefade in non-3D supporting browsers (which tend to handle complex
transitions poorly in general
*/

(function( $, window, undefined ) {

// Set the slide transitions's fallback to "fade"
$.mobile.transitionFallbacks.slidefade = "fade";

})( jQuery, this );
```

```
/*
* fallback transition for slideup in non-3D supporting browsers (which tend to handle complex
transitions poorly in general
*/

(function( $, window, undefined ) {

$.mobile.transitionFallbacks.slideup = "fade";

})( jQuery, this );

/*
* fallback transition for turn in non-3D supporting browsers (which tend to handle complex
transitions poorly in general
*/

(function( $, window, undefined ) {

$.mobile.transitionFallbacks.turn = "fade";

})( jQuery, this );


(function( $, undefined ) {

$.mobile.degradeInputs = {
    color: false,
    date: false,
    datetime: false,
    "datetime-local": false,
    email: false,
    month: false,
    number: false,
    range: "number",
    search: "text",
    tel: false,
    time: false,
    url: false,
    week: false
};
// Backcompat remove in 1.5
$.mobile.page.prototype.options.degradeInputs = $.mobile.degradeInputs;

// Auto self-init widgets
$.mobile.degradeInputsWithin = function( target ) {

    target = $( target );

    // Degrade inputs to avoid poorly implemented native functionality
    target.find( "input" ).not( $.mobile.page.prototype.keepNativeSelector() ).each(function() {
        var element = $( this ),
            type = this.getAttribute( "type" ),
            optType = $.mobile.degradeInputs[ type ] || "text",
```

```
            html, hasType, findstr, repstr;

        if ( $.mobile.degradeInputs[ type ] ) {
            html = $( "<div>" ).html( element.clone() ).html();
            // In IE browsers, the type sometimes doesn't exist in the cloned markup, so we
            replace the closing tag instead
            hasType = html.indexOf( " type=" ) > -1;
            findstr = hasType ? /\s+type=["']?\w+['"]?/ : /\/?>/;
            repstr = " type=\"" + optType + "\" data-" + $.mobile.ns + "type=\"" + type + "\"" +
             ( hasType ? "" : ">" );

            element.replaceWith( html.replace( findstr, repstr ) );
        }
    });

};

})( jQuery );

(function( $, window, undefined ) {

$.widget( "mobile.page", $.mobile.page, {
    options: {

        // Accepts left, right and none
        closeBtn: "left",
        closeBtnText: "Close",
        overlayTheme: "a",
        corners: true,
        dialog: false
    },

    _create: function() {
        this._super();
        if ( this.options.dialog ) {

            $.extend( this, {
                _inner: this.element.children(),
                _headerCloseButton: null
            });

            if ( !this.options.enhanced ) {
                this._setCloseBtn( this.options.closeBtn );
            }
        }
    },

    _enhance: function() {
        this._super();

        // Class the markup for dialog styling and wrap interior
        if ( this.options.dialog ) {
            this.element.addClass( "ui-dialog" )
                .wrapInner( $( "<div/>", {
```

```
                // ARIA role
                "role" : "dialog",
                "class" : "ui-dialog-contain ui-overlay-shadow" +
                    ( this.options.corners ? " ui-corner-all" : "" )
            }));
        }
    },

    _setOptions: function( options ) {
        var closeButtonLocation, closeButtonText,
            currentOpts = this.options;

        if ( options.corners !== undefined ) {
            this._inner.toggleClass( "ui-corner-all", !!options.corners );
        }

        if ( options.overlayTheme !== undefined ) {
            if ( $.mobile.activePage[ 0 ] === this.element[ 0 ] ) {
                currentOpts.overlayTheme = options.overlayTheme;
                this._handlePageBeforeShow();
            }
        }

        if ( options.closeBtnText !== undefined ) {
            closeButtonLocation = currentOpts.closeBtn;
            closeButtonText = options.closeBtnText;
        }

        if ( options.closeBtn !== undefined ) {
            closeButtonLocation = options.closeBtn;
        }

        if ( closeButtonLocation ) {
            this._setCloseBtn( closeButtonLocation, closeButtonText );
        }

        this._super( options );
    },

    _handlePageBeforeShow: function () {
        if ( this.options.overlayTheme && this.options.dialog ) {
            this.removeContainerBackground();
            this.setContainerBackground( this.options.overlayTheme );
        } else {
            this._super();
        }
    },

    _setCloseBtn: function( location, text ) {
        var dst,
            btn = this._headerCloseButton;

        // Sanitize value
```

```
        location = "left" === location ? "left" : "right" === location ? "right" : "none";

        if ( "none" === location ) {
            if ( btn ) {
                btn.remove();
                btn = null;
            }
        } else if ( btn ) {
            btn.removeClass( "ui-btn-left ui-btn-right" ).addClass( "ui-btn-" + location );
            if ( text ) {
                btn.text( text );
            }
        } else {
            dst = this._inner.find( ":jqmData(role='header')" ).first();
            btn = $( "<a></a>", {
                    "href": "#",
                    "class": "ui-btn ui-corner-all ui-icon-delete ui-btn-icon-notext ui-btn-" +
                    location
                })
                .attr( "data-" + $.mobile.ns + "rel", "back" )
                .text( text || this.options.closeBtnText || "" )
                .prependTo( dst );
        }

        this._headerCloseButton = btn;
    }
});

})( jQuery, this );

(function( $, window, undefined ) {

$.widget( "mobile.dialog", {
    options: {

        // Accepts left, right and none
        closeBtn: "left",
        closeBtnText: "Close",
        overlayTheme: "a",
        corners: true
    },

    // Override the theme set by the page plugin on pageshow
    _handlePageBeforeShow: function() {
        this._isCloseable = true;
        if ( this.options.overlayTheme ) {
            this.element
                .page( "removeContainerBackground" )
                .page( "setContainerBackground", this.options.overlayTheme );
        }
    },

    _handlePageBeforeHide: function() {
        this._isCloseable = false;
```

```
    },

    // click and submit events:
    // - clicks and submits should use the closing transition that the dialog
    //   opened with unless a data-transition is specified on the link/form
    // - if the click was on the close button, or the link has a data-rel="back"
    //   it'll go back in history naturally
    _handleVClickSubmit: function( event ) {
        var attrs,
            $target = $( event.target ).closest( event.type === "vclick" ? "a" : "form" );

        if ( $target.length && !$target.jqmData( "transition" ) ) {
            attrs = {};
            attrs[ "data-" + $.mobile.ns + "transition" ] =
                ( $.mobile.navigate.history.getActive() || {} )[ "transition" ] ||
                $.mobile.defaultDialogTransition;
            attrs[ "data-" + $.mobile.ns + "direction" ] = "reverse";
            $target.attr( attrs );
        }
    },

    _create: function() {
        var elem = this.element,
            opts = this.options;

        // Class the markup for dialog styling and wrap interior
        elem.addClass( "ui-dialog" )
            .wrapInner( $( "<div/>", {

                // ARIA role
                "role" : "dialog",
                "class" : "ui-dialog-contain ui-overlay-shadow" +
                    ( !!opts.corners ? " ui-corner-all" : "" )
            }));

        $.extend( this, {
            _isCloseable: false,
            _inner: elem.children(),
            _headerCloseButton: null
        });

        this._on( elem, {
            vclick: "_handleVClickSubmit",
            submit: "_handleVClickSubmit",
            pagebeforeshow: "_handlePageBeforeShow",
            pagebeforehide: "_handlePageBeforeHide"
        });

        this._setCloseBtn( opts.closeBtn );
    },

    _setOptions: function( options ) {
        var closeButtonLocation, closeButtonText,
            currentOpts = this.options;
```

```
        if ( options.corners !== undefined ) {
            this._inner.toggleClass( "ui-corner-all", !!options.corners );
        }

        if ( options.overlayTheme !== undefined ) {
            if ( $.mobile.activePage[ 0 ] === this.element[ 0 ] ) {
                currentOpts.overlayTheme = options.overlayTheme;
                this._handlePageBeforeShow();
            }
        }

        if ( options.closeBtnText !== undefined ) {
            closeButtonLocation = currentOpts.closeBtn;
            closeButtonText = options.closeBtnText;
        }

        if ( options.closeBtn !== undefined ) {
            closeButtonLocation = options.closeBtn;
        }

        if ( closeButtonLocation ) {
            this._setCloseBtn( closeButtonLocation, closeButtonText );
        }

        this._super( options );
    },

    _setCloseBtn: function( location, text ) {
        var dst,
            btn = this._headerCloseButton;

        // Sanitize value
        location = "left" === location ? "left" : "right" === location ? "right" : "none";

        if ( "none" === location ) {
            if ( btn ) {
                btn.remove();
                btn = null;
            }
        } else if ( btn ) {
            btn.removeClass( "ui-btn-left ui-btn-right" ).addClass( "ui-btn-" + location );
            if ( text ) {
                btn.text( text );
            }
        } else {
            dst = this._inner.find( ":jqmData(role='header')" ).first();
            btn = $( "<a></a>", {
                    "role": "button",
                    "href": "#",
                    "class": "ui-btn ui-corner-all ui-icon-delete ui-btn-icon-notext ui-btn-" +
                    location
                })
                .text( text || this.options.closeBtnText || "" )
```

```
                .prependTo( dst );
            this._on( btn, { click: "close" } );
        }

        this._headerCloseButton = btn;
    },

    // Close method goes back in history
    close: function() {
        var hist = $.mobile.navigate.history;

        if ( this._isCloseable ) {
            this._isCloseable = false;
            // If the hash listening is enabled and there is at least one preceding history
            // entry it's ok to go back. Initial pages with the dialog hash state are an example
            // where the stack check is necessary
            if ( $.mobile.hashListeningEnabled && hist.activeIndex > 0 ) {
                $.mobile.back();
            } else {
                $.mobile.pageContainer.pagecontainer( "back" );
            }
        }
    }
});

})( jQuery, this );

(function( $, undefined ) {

var rInitialLetter = /([A-Z])/g,

    // Construct iconpos class from iconpos value
    iconposClass = function( iconpos ) {
        return ( "ui-btn-icon-" + ( iconpos === null ? "left" : iconpos ) );
    };

$.widget( "mobile.collapsible", {
    options: {
        enhanced: false,
        expandCueText: null,
        collapseCueText: null,
        collapsed: true,
        heading: "h1,h2,h3,h4,h5,h6,legend",
        collapsedIcon: null,
        expandedIcon: null,
        iconpos: null,
        theme: null,
        contentTheme: null,
        inset: null,
        corners: null,
        mini: null
    },

    _create: function() {
```

```
        var elem = this.element,
            ui = {
                accordion: elem
                    .closest( ":jqmData(role='collapsible-set')," +
                        ":jqmData(role='collapsibleset')" +
                        ( $.mobile.collapsibleset ? ", :mobile-collapsibleset" :
                            "" ) )
                    .addClass( "ui-collapsible-set" )
            };

        this._ui = ui;
        this._renderedOptions = this._getOptions( this.options );

        if ( this.options.enhanced ) {
            ui.heading = $( ".ui-collapsible-heading", this.element[ 0 ] );
            ui.content = ui.heading.next();
            ui.anchor = $( "a", ui.heading[ 0 ] ).first();
            ui.status = ui.anchor.children( ".ui-collapsible-heading-status" );
        } else {
            this._enhance( elem, ui );
        }

        this._on( ui.heading, {
            "tap": function() {
                ui.heading.find( "a" ).first().addClass( $.mobile.activeBtnClass );
            },

            "click": function( event ) {
                this._handleExpandCollapse( !ui.heading.hasClass(
                "ui-collapsible-heading-collapsed" ) );
                event.preventDefault();
                event.stopPropagation();
            }
        });
    },

    // Adjust the keys inside options for inherited values
    _getOptions: function( options ) {
        var key,
            accordion = this._ui.accordion,
            accordionWidget = this._ui.accordionWidget;

        // Copy options
        options = $.extend( {}, options );

        if ( accordion.length && !accordionWidget ) {
            this._ui.accordionWidget =
            accordionWidget = accordion.data( "mobile-collapsibleset" );
        }

        for ( key in options ) {

            // Retrieve the option value first from the options object passed in and, if
            // null, from the parent accordion or, if that's null too, or if there's no
```

```
            // parent accordion, then from the defaults.
            options[ key ] =
                ( options[ key ] != null ) ? options[ key ] :
                ( accordionWidget ) ? accordionWidget.options[ key ] :
                accordion.length ? $.mobile.getAttribute( accordion[ 0 ],
                    key.replace( rInitialLetter, "-$1" ).toLowerCase() ):
                null;

            if ( null == options[ key ] ) {
                options[ key ] = $.mobile.collapsible.defaults[ key ];
            }
        }

        return options;
    },

    _themeClassFromOption: function( prefix, value ) {
        return ( value ? ( value === "none" ? "" : prefix + value ) : "" );
    },

    _enhance: function( elem, ui ) {
        var iconclass,
            opts = this._renderedOptions,
            contentThemeClass = this._themeClassFromOption( "ui-body-", opts.contentTheme );

        elem.addClass( "ui-collapsible " +
            ( opts.inset ? "ui-collapsible-inset " : "" ) +
            ( opts.inset && opts.corners ? "ui-corner-all " : "" ) +
            ( contentThemeClass ? "ui-collapsible-themed-content " : "" ) );
        ui.originalHeading = elem.children( this.options.heading ).first(),
        ui.content = elem
            .wrapInner( "<div " +
                "class='ui-collapsible-content " +
                contentThemeClass + "'></div>" )
            .children( ".ui-collapsible-content" ),
        ui.heading = ui.originalHeading;

        // Replace collapsibleHeading if it's a legend
        if ( ui.heading.is( "legend" ) ) {
            ui.heading = $( "<div role='heading'>"+ ui.heading.html() +"</div>" );
            ui.placeholder = $( "<div><!-- placeholder for legend --></div>" ).insertBefore( ui.
            originalHeading );
            ui.originalHeading.remove();
        }

        iconclass = ( opts.collapsed ? ( opts.collapsedIcon ? "ui-icon-" + opts.collapsedIcon :
        "" ):
            ( opts.expandedIcon ? "ui-icon-" + opts.expandedIcon : "" ) );

        ui.status = $( "<span class='ui-collapsible-heading-status'></span>" );
        ui.anchor = ui.heading
            .detach()
            //modify markup & attributes
            .addClass( "ui-collapsible-heading" )
```

```
            .append( ui.status )
            .wrapInner( "<a href='#' class='ui-collapsible-heading-toggle'></a>" )
            .find( "a" )
                .first()
                .addClass( "ui-btn " +
                    ( iconclass ? iconclass + " " : "" ) +
                    ( iconclass ? iconposClass( opts.iconpos ) +
                        " " : "" ) +
                    this._themeClassFromOption( "ui-btn-", opts.theme ) + " " +
                    ( opts.mini ? "ui-mini " : "" ) );

        //drop heading in before content
        ui.heading.insertBefore( ui.content );

        this._handleExpandCollapse( this.options.collapsed );

        return ui;
    },

    refresh: function() {
        this._applyOptions( this.options );
        this._renderedOptions = this._getOptions( this.options );
    },

    _applyOptions: function( options ) {
        var isCollapsed, newTheme, oldTheme, hasCorners, hasIcon,
            elem = this.element,
            currentOpts = this._renderedOptions,
            ui = this._ui,
            anchor = ui.anchor,
            status = ui.status,
            opts = this._getOptions( options );

        // First and foremost we need to make sure the collapsible is in the proper
        // state, in case somebody decided to change the collapsed option at the
        // same time as another option
        if ( options.collapsed !== undefined ) {
            this._handleExpandCollapse( options.collapsed );
        }

        isCollapsed = elem.hasClass( "ui-collapsible-collapsed" );

        // We only need to apply the cue text for the current state right away.
        // The cue text for the alternate state will be stored in the options
        // and applied the next time the collapsible's state is toggled
        if ( isCollapsed ) {
            if ( opts.expandCueText !== undefined ) {
                status.text( opts.expandCueText );
            }
        } else {
            if ( opts.collapseCueText !== undefined ) {
                status.text( opts.collapseCueText );
            }
        }
```

```
        // Update icon

        // Is it supposed to have an icon?
        hasIcon =

            // If the collapsedIcon is being set, consult that
            ( opts.collapsedIcon !== undefined ? opts.collapsedIcon !== false :

                // Otherwise consult the existing option value
                currentOpts.collapsedIcon !== false );


        // If any icon-related options have changed, make sure the new icon
        // state is reflected by first removing all icon-related classes
        // reflecting the current state and then adding all icon-related
        // classes for the new state
        if ( !( opts.iconpos === undefined &&
            opts.collapsedIcon === undefined &&
            opts.expandedIcon === undefined ) ) {

            // Remove all current icon-related classes
            anchor.removeClass( [ iconposClass( currentOpts.iconpos ) ]
                .concat( ( currentOpts.expandedIcon ?
                    [ "ui-icon-" + currentOpts.expandedIcon ] : [] ) )
                .concat( ( currentOpts.collapsedIcon ?
                    [ "ui-icon-" + currentOpts.collapsedIcon ] : [] ) )
                .join( " " ) );

            // Add new classes if an icon is supposed to be present
            if ( hasIcon ) {
                anchor.addClass(
                    [ iconposClass( opts.iconpos !== undefined ?
                        opts.iconpos : currentOpts.iconpos ) ]
                        .concat( isCollapsed ?
                            [ "ui-icon-" + ( opts.collapsedIcon !== undefined ?
                                opts.collapsedIcon :
                                currentOpts.collapsedIcon ) ] :
                            [ "ui-icon-" + ( opts.expandedIcon !== undefined ?
                                opts.expandedIcon :
                                currentOpts.expandedIcon ) ] )
                        .join( " " ) );
            }
        }

        if ( opts.theme !== undefined ) {
            oldTheme = this._themeClassFromOption( "ui-btn-", currentOpts.theme );
            newTheme = this._themeClassFromOption( "ui-btn-", opts.theme );
            anchor.removeClass( oldTheme ).addClass( newTheme );
        }

        if ( opts.contentTheme !== undefined ) {
            oldTheme = this._themeClassFromOption( "ui-body-",
                currentOpts.contentTheme );
```

```
            newTheme = this._themeClassFromOption( "ui-body-",
                opts.contentTheme );
            ui.content.removeClass( oldTheme ).addClass( newTheme );
        }

        if ( opts.inset !== undefined ) {
            elem.toggleClass( "ui-collapsible-inset", opts.inset );
            hasCorners = !!( opts.inset && ( opts.corners || currentOpts.corners ) );
        }

        if ( opts.corners !== undefined ) {
            hasCorners = !!( opts.corners && ( opts.inset || currentOpts.inset ) );
        }

        if ( hasCorners !== undefined ) {
            elem.toggleClass( "ui-corner-all", hasCorners );
        }

        if ( opts.mini !== undefined ) {
            anchor.toggleClass( "ui-mini", opts.mini );
        }
    },

    _setOptions: function( options ) {
        this._applyOptions( options );
        this._super( options );
        this._renderedOptions = this._getOptions( this.options );
    },

    _handleExpandCollapse: function( isCollapse ) {
        var opts = this._renderedOptions,
            ui = this._ui;

        ui.status.text( isCollapse ? opts.expandCueText : opts.collapseCueText );
        ui.heading
            .toggleClass( "ui-collapsible-heading-collapsed", isCollapse )
            .find( "a" ).first()
            .toggleClass( "ui-icon-" + opts.expandedIcon, !isCollapse )

            // logic or cause same icon for expanded/collapsed state would remove the
            ui-icon-class
            .toggleClass( "ui-icon-" + opts.collapsedIcon, ( isCollapse || opts.expandedIcon ===
             opts.collapsedIcon ) )
            .removeClass( $.mobile.activeBtnClass );

        this.element.toggleClass( "ui-collapsible-collapsed", isCollapse );
        ui.content
            .toggleClass( "ui-collapsible-content-collapsed", isCollapse )
            .attr( "aria-hidden", isCollapse )
            .trigger( "updatelayout" );
        this.options.collapsed = isCollapse;
        this._trigger( isCollapse ? "collapse" : "expand" );
    },
```

```
    expand: function() {
        this._handleExpandCollapse( false );
    },

    collapse: function() {
        this._handleExpandCollapse( true );
    },

    _destroy: function() {
        var ui = this._ui,
            opts = this.options;

        if ( opts.enhanced ) {
            return;
        }

        if ( ui.placeholder ) {
            ui.originalHeading.insertBefore( ui.placeholder );
            ui.placeholder.remove();
            ui.heading.remove();
        } else {
            ui.status.remove();
            ui.heading
                .removeClass( "ui-collapsible-heading ui-collapsible-heading-collapsed" )
                .children()
                    .contents()
                        .unwrap();
        }

        ui.anchor.contents().unwrap();
        ui.content.contents().unwrap();
        this.element
            .removeClass( "ui-collapsible ui-collapsible-collapsed " +
                "ui-collapsible-themed-content ui-collapsible-inset ui-corner-all" );
    }
});

// Defaults to be used by all instances of collapsible if per-instance values
// are unset or if nothing is specified by way of inheritance from an accordion.
// Note that this hash does not contain options "collapsed" or "heading",
// because those are not inheritable.
$.mobile.collapsible.defaults = {
    expandCueText: " click to expand contents",
    collapseCueText: " click to collapse contents",
    collapsedIcon: "plus",
    contentTheme: "inherit",
    expandedIcon: "minus",
    iconpos: "left",
    inset: true,
    corners: true,
    theme: "inherit",
    mini: false
};
```

```
})( jQuery );

(function( $, undefined ) {

$.mobile.behaviors.addFirstLastClasses = {
    _getVisibles: function( $els, create ) {
        var visibles;

        if ( create ) {
            visibles = $els.not( ".ui-screen-hidden" );
        } else {
            visibles = $els.filter( ":visible" );
            if ( visibles.length === 0 ) {
                visibles = $els.not( ".ui-screen-hidden" );
            }
        }

        return visibles;
    },

    _addFirstLastClasses: function( $els, $visibles, create ) {
        $els.removeClass( "ui-first-child ui-last-child" );
        $visibles.eq( 0 ).addClass( "ui-first-child" ).end().last().addClass( "ui-last-child" );
        if ( !create ) {
            this.element.trigger( "updatelayout" );
        }
    },

    _removeFirstLastClasses: function( $els ) {
        $els.removeClass( "ui-first-child ui-last-child" );
    }
};

})( jQuery );

(function( $, undefined ) {

var childCollapsiblesSelector = ":mobile-collapsible, " + $.mobile.collapsible.initSelector;

$.widget( "mobile.collapsibleset", $.extend( {

    // The initSelector is deprecated as of 1.4.0. In 1.5.0 we will use
    // :jqmData(role='collapsibleset') which will allow us to get rid of the line
    // below altogether, because the autoinit will generate such an initSelector
    initSelector: ":jqmData(role='collapsible-set'),:jqmData(role='collapsibleset')",

    options: $.extend( {
        enhanced: false
    }, $.mobile.collapsible.defaults ),

    _handleCollapsibleExpand: function( event ) {
        var closestCollapsible = $( event.target ).closest( ".ui-collapsible" );

        if ( closestCollapsible.parent().is( ":mobile-collapsibleset,
```

```
        :jqmData(role='collapsible-set')" ) ) {
            closestCollapsible
                .siblings( ".ui-collapsible:not(.ui-collapsible-collapsed)" )
                .collapsible( "collapse" );
        }
    },

    _create: function() {
        var elem = this.element,
            opts = this.options;

        $.extend( this, {
            _classes: ""
        });

        if ( !opts.enhanced ) {
            elem.addClass( "ui-collapsible-set " +
                this._themeClassFromOption( "ui-group-theme-", opts.theme ) + " " +
                ( opts.corners && opts.inset ? "ui-corner-all " : "" ) );
            this.element.find( $.mobile.collapsible.initSelector ).collapsible();
        }

        this._on( elem, { collapsibleexpand: "_handleCollapsibleExpand" } );
    },

    _themeClassFromOption: function( prefix, value ) {
        return ( value ? ( value === "none" ? "" : prefix + value ) : "" );
    },

    _init: function() {
        this._refresh( true );

        // Because the corners are handled by the collapsible itself and the default state is
        collapsed
        // That was causing https://github.com/jquery/jquery-mobile/issues/4116
        this.element
            .children( childCollapsiblesSelector )
            .filter( ":jqmData(collapsed='false')" )
            .collapsible( "expand" );
    },

    _setOptions: function( options ) {
        var ret, hasCorners,
            elem = this.element,
            themeClass = this._themeClassFromOption( "ui-group-theme-", options.theme );

        if ( themeClass ) {
            elem
                .removeClass( this._themeClassFromOption( "ui-group-theme-", this.options.theme
                ) )
                .addClass( themeClass );
        }

        if ( options.inset !== undefined ) {
```

```
            hasCorners = !!( options.inset && ( options.corners || this.options.corners ) );
        }

        if ( options.corners !== undefined ) {
            hasCorners = !!( options.corners && ( options.inset || this.options.inset ) );
        }

        if ( hasCorners !== undefined ) {
            elem.toggleClass( "ui-corner-all", hasCorners );
        }

        ret = this._super( options );
        this.element.children( ":mobile-collapsible" ).collapsible( "refresh" );
        return ret;
    },

    _destroy: function() {
        var el = this.element;

        this._removeFirstLastClasses( el.children( childCollapsiblesSelector ) );
        el
            .removeClass( "ui-collapsible-set ui-corner-all " +
                this._themeClassFromOption( "ui-group-theme-", this.options.theme ) )
            .children( ":mobile-collapsible" )
            .collapsible( "destroy" );
    },

    _refresh: function( create ) {
        var collapsiblesInSet = this.element.children( childCollapsiblesSelector );

        this.element.find( $.mobile.collapsible.initSelector ).not( ".ui-collapsible" ).
        collapsible();

        this._addFirstLastClasses( collapsiblesInSet, this._getVisibles( collapsiblesInSet,
        create ), create );
    },

    refresh: function() {
        this._refresh( false );
    }
}, $.mobile.behaviors.addFirstLastClasses ) );

})( jQuery );

(function( $, undefined ) {

// Deprecated in 1.4
$.fn.fieldcontain = function(/* options */) {
    return this.addClass( "ui-field-contain" );
};

})( jQuery );

(function( $, undefined ) {
```

```
$.fn.grid = function( options ) {
    return this.each(function() {

        var $this = $( this ),
            o = $.extend({
                grid: null
            }, options ),
            $kids = $this.children(),
            gridCols = { solo:1, a:2, b:3, c:4, d:5 },
            grid = o.grid,
            iterator,
            letter;

            if ( !grid ) {
                if ( $kids.length <= 5 ) {
                    for ( letter in gridCols ) {
                        if ( gridCols[ letter ] === $kids.length ) {
                            grid = letter;
                        }
                    }
                } else {
                    grid = "a";
                    $this.addClass( "ui-grid-duo" );
                }
            }
            iterator = gridCols[grid];

        $this.addClass( "ui-grid-" + grid );

        $kids.filter( ":nth-child(" + iterator + "n+1)" ).addClass( "ui-block-a" );

        if ( iterator > 1 ) {
            $kids.filter( ":nth-child(" + iterator + "n+2)" ).addClass( "ui-block-b" );
        }
        if ( iterator > 2 ) {
            $kids.filter( ":nth-child(" + iterator + "n+3)" ).addClass( "ui-block-c" );
        }
        if ( iterator > 3 ) {
            $kids.filter( ":nth-child(" + iterator + "n+4)" ).addClass( "ui-block-d" );
        }
        if ( iterator > 4 ) {
            $kids.filter( ":nth-child(" + iterator + "n+5)" ).addClass( "ui-block-e" );
        }
    });
};
})( jQuery );

(function( $, undefined ) {

$.widget( "mobile.navbar", {
    options: {
        iconpos: "top",
        grid: null
```

```
    },

    _create: function() {

        var $navbar = this.element,
            $navbtns = $navbar.find( "a" ),
            iconpos = $navbtns.filter( ":jqmData(icon)" ).length ? this.options.iconpos :
            undefined;

        $navbar.addClass( "ui-navbar" )
            .attr( "role", "navigation" )
            .find( "ul" )
            .jqmEnhanceable()
            .grid({ grid: this.options.grid });

        $navbtns
            .each( function() {
                var icon = $.mobile.getAttribute( this, "icon" ),
                    theme = $.mobile.getAttribute( this, "theme" ),
                    classes = "ui-btn";

                if ( theme ) {
                    classes += " ui-btn-" + theme;
                }
                if ( icon ) {
                    classes += " ui-icon-" + icon + " ui-btn-icon-" + iconpos;
                }
                $( this ).addClass( classes );
            });

        $navbar.delegate( "a", "vclick", function( /* event */ ) {
            var activeBtn = $( this );

            if ( !( activeBtn.hasClass( "ui-state-disabled" ) ||

                // DEPRECATED as of 1.4.0 - remove after 1.4.0 release
                // only ui-state-disabled should be present thereafter
                activeBtn.hasClass( "ui-disabled" ) ||
                activeBtn.hasClass( $.mobile.activeBtnClass ) ) ) {

                $navbtns.removeClass( $.mobile.activeBtnClass );
                activeBtn.addClass( $.mobile.activeBtnClass );

                // The code below is a workaround to fix #1181
                $( document ).one( "pagehide", function() {
                    activeBtn.removeClass( $.mobile.activeBtnClass );
                });
            }
        });

        // Buttons in the navbar with ui-state-persist class should regain their active state
        before page show
        $navbar.closest( ".ui-page" ).bind( "pagebeforeshow", function() {
            $navbtns.filter( ".ui-state-persist" ).addClass( $.mobile.activeBtnClass );
```

```
        });
    }
});

})( jQuery );

(function( $, undefined ) {

var getAttr = $.mobile.getAttribute;

$.widget( "mobile.listview", $.extend( {

    options: {
        theme: null,
        countTheme: null, /* Deprecated in 1.4 */
        dividerTheme: null,
        icon: "carat-r",
        splitIcon: "carat-r",
        splitTheme: null,
        corners: true,
        shadow: true,
        inset: false
    },

    _create: function() {
        var t = this,
            listviewClasses = "";

        listviewClasses += t.options.inset ? " ui-listview-inset" : "";

        if ( !!t.options.inset ) {
            listviewClasses += t.options.corners ? " ui-corner-all" : "";
            listviewClasses += t.options.shadow ? " ui-shadow" : "";
        }

        // create listview markup
        t.element.addClass( " ui-listview" + listviewClasses );

        t.refresh( true );
    },

    // TODO: Remove in 1.5
    _findFirstElementByTagName: function( ele, nextProp, lcName, ucName ) {
        var dict = {};
        dict[ lcName ] = dict[ ucName ] = true;
        while ( ele ) {
            if ( dict[ ele.nodeName ] ) {
                return ele;
            }
            ele = ele[ nextProp ];
        }
        return null;
    },
    // TODO: Remove in 1.5
```

```
    _addThumbClasses: function( containers ) {
        var i, img, len = containers.length;
        for ( i = 0; i < len; i++ ) {
            img = $( this._findFirstElementByTagName( containers[ i ].firstChild, "nextSibling",
             "img", "IMG" ) );
            if ( img.length ) {
                $( this._findFirstElementByTagName( img[ 0 ].parentNode, "parentNode", "li",
                "LI" ) ).addClass( img.hasClass( "ui-li-icon" ) ? "ui-li-has-icon" :
                "ui-li-has-thumb" );
            }
        }
    },

    _getChildrenByTagName: function( ele, lcName, ucName ) {
        var results = [],
            dict = {};
        dict[ lcName ] = dict[ ucName ] = true;
        ele = ele.firstChild;
        while ( ele ) {
            if ( dict[ ele.nodeName ] ) {
                results.push( ele );
            }
            ele = ele.nextSibling;
        }
        return $( results );
    },

    _beforeListviewRefresh: $.noop,
    _afterListviewRefresh: $.noop,

    refresh: function( create ) {
        var buttonClass, pos, numli, item, itemClass, itemTheme, itemIcon, icon, a,
            isDivider, startCount, newStartCount, value, last, splittheme, splitThemeClass,
            spliticon,
            altButtonClass, dividerTheme, li,
            o = this.options,
            $list = this.element,
            ol = !!$.nodeName( $list[ 0 ], "ol" ),
            start = $list.attr( "start" ),
            itemClassDict = {},
            countBubbles = $list.find( ".ui-li-count" ),
            countTheme = getAttr( $list[ 0 ], "counttheme" ) || this.options.countTheme,
            countThemeClass = countTheme ? "ui-body-" + countTheme : "ui-body-inherit";

        if ( o.theme ) {
            $list.addClass( "ui-group-theme-" + o.theme );
        }

        // Check if a start attribute has been set while taking a value of 0 into account
        if ( ol && ( start || start === 0 ) ) {
            startCount = parseInt( start, 10 ) - 1;
            $list.css( "counter-reset", "listnumbering " + startCount );
        }
```

```
        this._beforeListviewRefresh();

        li = this._getChildrenByTagName( $list[ 0 ], "li", "LI" );

        for ( pos = 0, numli = li.length; pos < numli; pos++ ) {
            item = li.eq( pos );
            itemClass = "";

            if ( create || item[ 0 ].className.search( /\bui-li-static\b|\bui-li-divider\b/ ) <
            0 ) {
                a = this._getChildrenByTagName( item[ 0 ], "a", "A" );
                isDivider = ( getAttr( item[ 0 ], "role" ) === "list-divider" );
                value = item.attr( "value" );
                itemTheme = getAttr( item[ 0 ], "theme" );

                if ( a.length && a[ 0 ].className.search( /\bui-btn\b/ ) < 0 && !isDivider ) {
                    itemIcon = getAttr( item[ 0 ], "icon" );
                    icon = ( itemIcon === false ) ? false : ( itemIcon || o.icon );

                    // TODO: Remove in 1.5 together with links.js (links.js / .ui-link
                    deprecated in 1.4)
                    a.removeClass( "ui-link" );

                    buttonClass = "ui-btn";

                    if ( itemTheme ) {
                        buttonClass += " ui-btn-" + itemTheme;
                    }

                    if ( a.length > 1 ) {
                        itemClass = "ui-li-has-alt";

                        last = a.last();
                        splittheme = getAttr( last[ 0 ], "theme" ) || o.splitTheme || getAttr(
                        item[ 0 ], "theme", true );
                        splitThemeClass = splittheme ? " ui-btn-" + splittheme : "";
                        spliticon = getAttr( last[ 0 ], "icon" ) || getAttr( item[ 0 ], "icon" )
                         || o.splitIcon;
                        altButtonClass = "ui-btn ui-btn-icon-notext ui-icon-" + spliticon +
                        splitThemeClass;

                        last
                            .attr( "title", $.trim( last.getEncodedText() ) )
                            .addClass( altButtonClass )
                            .empty();
                    } else if ( icon ) {
                        buttonClass += " ui-btn-icon-right ui-icon-" + icon;
                    }

                    a.first().addClass( buttonClass );
                } else if ( isDivider ) {
                    dividerTheme = ( getAttr( item[ 0 ], "theme" ) || o.dividerTheme || o.theme
                    );
```

```
                    itemClass = "ui-li-divider ui-bar-" + ( dividerTheme ? dividerTheme :
                    "inherit" );

                    item.attr( "role", "heading" );
                } else if ( a.length <= 0 ) {
                    itemClass = "ui-li-static ui-body-" + ( itemTheme ? itemTheme : "inherit" );
                }
                if ( ol && value ) {
                    newStartCount = parseInt( value , 10 ) - 1;

                    item.css( "counter-reset", "listnumbering " + newStartCount );
                }
            }

            // Instead of setting item class directly on the list item
            // at this point in time, push the item into a dictionary
            // that tells us what class to set on it so we can do this after this
            // processing loop is finished.

            if ( !itemClassDict[ itemClass ] ) {
                itemClassDict[ itemClass ] = [];
            }

            itemClassDict[ itemClass ].push( item[ 0 ] );
        }

        // Set the appropriate listview item classes on each list item.
        // The main reason we didn't do this
        // in the for-loop above is because we can eliminate per-item function overhead
        // by calling addClass() and children() once or twice afterwards. This
        // can give us a significant boost on platforms like WP7.5.

        for ( itemClass in itemClassDict ) {
            $( itemClassDict[ itemClass ] ).addClass( itemClass );
        }

        countBubbles.each( function() {
            $( this ).closest( "li" ).addClass( "ui-li-has-count" );
        });
        if ( countThemeClass ) {
            countBubbles.addClass( countThemeClass );
        }

        // Deprecated in 1.4. From 1.5 you have to add class ui-li-has-thumb or ui-li-has-icon
        to the LI.
        this._addThumbClasses( li );
        this._addThumbClasses( li.find( ".ui-btn" ) );

        this._afterListviewRefresh();

        this._addFirstLastClasses( li, this._getVisibles( li, create ), create );
    }
}, $.mobile.behaviors.addFirstLastClasses ) );
```

```
})( jQuery );

(function( $, undefined ) {

function defaultAutodividersSelector( elt ) {
    // look for the text in the given element
    var text = $.trim( elt.text() ) || null;

    if ( !text ) {
        return null;
    }

    // create the text for the divider (first uppercased letter)
    text = text.slice( 0, 1 ).toUpperCase();

    return text;
}

$.widget( "mobile.listview", $.mobile.listview, {
    options: {
        autodividers: false,
        autodividersSelector: defaultAutodividersSelector
    },

    _beforeListviewRefresh: function() {
        if ( this.options.autodividers ) {
            this._replaceDividers();
            this._superApply( arguments );
        }
    },

    _replaceDividers: function() {
        var i, lis, li, dividerText,
            lastDividerText = null,
            list = this.element,
            divider;

        list.children( "li:jqmData(role='list-divider')" ).remove();

        lis = list.children( "li" );

        for ( i = 0; i < lis.length ; i++ ) {
            li = lis[ i ];
            dividerText = this.options.autodividersSelector( $( li ) );

            if ( dividerText && lastDividerText !== dividerText ) {
                divider = document.createElement( "li" );
                divider.appendChild( document.createTextNode( dividerText ) );
                divider.setAttribute( "data-" + $.mobile.ns + "role", "list-divider" );
                li.parentNode.insertBefore( divider, li );
            }

            lastDividerText = dividerText;
        }
```

```
    }
});

})( jQuery );

(function( $, undefined ) {

var rdivider = /(^|\s)ui-li-divider($|\s)/,
    rhidden = /(^|\s)ui-screen-hidden($|\s)/;

$.widget( "mobile.listview", $.mobile.listview, {
    options: {
        hideDividers: false
    },

    _afterListviewRefresh: function() {
        var items, idx, item, hideDivider = true;

        this._superApply( arguments );

        if ( this.options.hideDividers ) {
            items = this._getChildrenByTagName( this.element[ 0 ], "li", "LI" );
            for ( idx = items.length - 1 ; idx > -1 ; idx-- ) {
                item = items[ idx ];
                if ( item.className.match( rdivider ) ) {
                    if ( hideDivider ) {
                        item.className = item.className + " ui-screen-hidden";
                    }
                    hideDivider = true;
                } else {
                    if ( !item.className.match( rhidden ) ) {
                        hideDivider = false;
                    }
                }
            }
        }
    }
});

})( jQuery );

(function( $, undefined ) {

$.mobile.nojs = function( target ) {
    $( ":jqmData(role='nojs')", target ).addClass( "ui-nojs" );
};

})( jQuery );

(function( $, undefined ) {

$.mobile.behaviors.formReset = {
    _handleFormReset: function() {
        this._on( this.element.closest( "form" ), {
```

```
            reset: function() {
                this._delay( "_reset" );
            }
        });
    }
};

})( jQuery );

/*
* "checkboxradio" plugin
*/

(function( $, undefined ) {

var escapeId = $.mobile.path.hashToSelector;

$.widget( "mobile.checkboxradio", $.extend( {

    initSelector: "input:not( :jqmData(role='flipswitch' )
    )[type='checkbox'],input[type='radio']:not( :jqmData(role='flipswitch' ))",

    options: {
        theme: "inherit",
        mini: false,
        wrapperClass: null,
        enhanced: false,
        iconpos: "left"

    },
    _create: function() {
        var input = this.element,
            o = this.options,
            inheritAttr = function( input, dataAttr ) {
                return input.jqmData( dataAttr ) ||
                    input.closest( "form, fieldset" ).jqmData( dataAttr );
            },
            // NOTE: Windows Phone could not find the label through a selector
            // filter works though.
            parentLabel = input.closest( "label" ),
            label = parentLabel.length ? parentLabel :
                input
                    .closest( "form, fieldset, :jqmData(role='page'), :jqmData(role='dialog')" )
                    .find( "label" )
                    .filter( "[for='" + escapeId( input[0].id ) + "']" )
                    .first(),
            inputtype = input[0].type,
            checkedClass = "ui-" + inputtype + "-on",
            uncheckedClass = "ui-" + inputtype + "-off";

        if ( inputtype !== "checkbox" && inputtype !== "radio" ) {
            return;
        }
```

```
        if ( this.element[0].disabled ) {
            this.options.disabled = true;
        }

        o.iconpos = inheritAttr( input, "iconpos" ) || label.attr( "data-" + $.mobile.ns +
        "iconpos" ) || o.iconpos,

        // Establish options
        o.mini = inheritAttr( input, "mini" ) || o.mini;

        // Expose for other methods
        $.extend( this, {
            input: input,
            label: label,
            parentLabel: parentLabel,
            inputtype: inputtype,
            checkedClass: checkedClass,
            uncheckedClass: uncheckedClass
        });

        if ( !this.options.enhanced ) {
            this._enhance();
        }

        this._on( label, {
            vmouseover: "_handleLabelVMouseOver",
            vclick: "_handleLabelVClick"
        });

        this._on( input, {
            vmousedown: "_cacheVals",
            vclick: "_handleInputVClick",
            focus: "_handleInputFocus",
            blur: "_handleInputBlur"
        });

        this._handleFormReset();
        this.refresh();
    },

    _enhance: function() {
        this.label.addClass( "ui-btn ui-corner-all");

        if ( this.parentLabel.length > 0 ) {
            this.input.add( this.label ).wrapAll( this._wrapper() );
        } else {
            //this.element.replaceWith( this.input.add( this.label ).wrapAll( this._wrapper() )
            );
            this.element.wrap( this._wrapper() );
            this.element.parent().prepend( this.label );
        }

        // Wrap the input + label in a div
```

```
        this._setOptions({
            "theme": this.options.theme,
            "iconpos": this.options.iconpos,
            "mini": this.options.mini
        });

    },

    _wrapper: function() {
        return $( "<div class='"  +
            ( this.options.wrapperClass ? this.options.wrapperClass : "" ) +
            " ui-" + this.inputtype +
            ( this.options.disabled ? " ui-state-disabled" : "" ) + "' ></div>" );
    },

    _handleInputFocus: function() {
        this.label.addClass( $.mobile.focusClass );
    },

    _handleInputBlur: function() {
        this.label.removeClass( $.mobile.focusClass );
    },

    _handleInputVClick: function() {
        // Adds checked attribute to checked input when keyboard is used
        this.element.prop( "checked", this.element.is( ":checked" ) );
        this._getInputSet().not( this.element ).prop( "checked", false );
        this._updateAll();
    },

    _handleLabelVMouseOver: function( event ) {
        if ( this.label.parent().hasClass( "ui-state-disabled" ) ) {
            event.stopPropagation();
        }
    },

    _handleLabelVClick: function( event ) {
        var input = this.element;

        if ( input.is( ":disabled" ) ) {
            event.preventDefault();
            return;
        }

        this._cacheVals();

        input.prop( "checked", this.inputtype === "radio" && true || !input.prop( "checked" ) );

        // trigger click handler's bound directly to the input as a substitute for
        // how label clicks behave normally in the browsers
        // TODO: it would be nice to let the browser's handle the clicks and pass them
        //       through to the associate input. we can swallow that click at the parent
        //       wrapper element level
        input.triggerHandler( "click" );
```

```
        // Input set for common radio buttons will contain all the radio
        // buttons, but will not for checkboxes. clearing the checked status
        // of other radios ensures the active button state is applied properly
        this._getInputSet().not( input ).prop( "checked", false );

        this._updateAll();
        return false;
    },

    _cacheVals: function() {
        this._getInputSet().each( function() {
            $( this ).attr("data-" + $.mobile.ns + "cacheVal", this.checked );
        });
    },

    // Returns those radio buttons that are supposed to be in the same group as
    // this radio button. In the case of a checkbox or a radio lacking a name
    // attribute, it returns this.element.
    _getInputSet: function() {
        var selector, formId,
            radio = this.element[ 0 ],
            name = radio.name,
            form = radio.form,
            doc = this.element.parents().last().get( 0 ),

            // A radio is always a member of its own group
            radios = this.element;

        // Only start running selectors if this is an attached radio button with a name
        if ( name && this.inputtype === "radio" && doc ) {
            selector = "input[type='radio'][name='" + escapeId( name ) + "']";

            // If we're inside a form
            if ( form ) {
                formId = form.id;

                // If the form has an ID, collect radios scattered throught the document which
                // nevertheless are part of the form by way of the value of their form attribute
                if ( formId ) {
                    radios = $( selector + "[form='" + escapeId( formId ) + "']", doc );
                }

                // Also add to those the radios in the form itself
                radios = $( form ).find( selector ).filter( function() {

                    // Some radios inside the form may belong to some other form by virtue of
                    // having a form attribute defined on them, so we must filter them out here
                    return ( this.form === form );
                }).add( radios );

            // If we're outside a form
            } else {
```

```
                // Collect all those radios which are also outside of a form and match our name
                radios = $( selector, doc ).filter( function() {
                    return !this.form;
                });
            }
        }
        return radios;
    },

    _updateAll: function() {
        var self = this;

        this._getInputSet().each( function() {
            var $this = $( this );

            if ( this.checked || self.inputtype === "checkbox" ) {
                $this.trigger( "change" );
            }
        })
        .checkboxradio( "refresh" );
    },

    _reset: function() {
        this.refresh();
    },

    // Is the widget supposed to display an icon?
    _hasIcon: function() {
        var controlgroup, controlgroupWidget,
            controlgroupConstructor = $.mobile.controlgroup;

        // If the controlgroup widget is defined ...
        if ( controlgroupConstructor ) {
            controlgroup = this.element.closest(
                ":mobile-controlgroup," +
                controlgroupConstructor.prototype.initSelector );

            // ... and the checkbox is in a controlgroup ...
            if ( controlgroup.length > 0 ) {

                // ... look for a controlgroup widget instance, and ...
                controlgroupWidget = $.data( controlgroup[ 0 ], "mobile-controlgroup" );

                // ... if found, decide based on the option value, ...
                return ( ( controlgroupWidget ? controlgroupWidget.options.type :

                    // ... otherwise decide based on the "type" data attribute.
                    controlgroup.attr( "data-" + $.mobile.ns + "type" ) ) !== "horizontal" );
            }
        }

        // Normally, the widget displays an icon.
        return true;
    },
```

```
    refresh: function() {
        var hasIcon = this._hasIcon(),
            isChecked = this.element[ 0 ].checked,
            active = $.mobile.activeBtnClass,
            iconposClass = "ui-btn-icon-" + this.options.iconpos,
            addClasses = [],
            removeClasses = [];

        if ( hasIcon ) {
            removeClasses.push( active );
            addClasses.push( iconposClass );
        } else {
            removeClasses.push( iconposClass );
            ( isChecked ? addClasses : removeClasses ).push( active );
        }

        if ( isChecked ) {
            addClasses.push( this.checkedClass );
            removeClasses.push( this.uncheckedClass );
        } else {
            addClasses.push( this.uncheckedClass );
            removeClasses.push( this.checkedClass );
        }

        this.label
            .addClass( addClasses.join( " " ) )
            .removeClass( removeClasses.join( " " ) );
    },

    widget: function() {
        return this.label.parent();
    },

    _setOptions: function( options ) {
        var label = this.label,
            currentOptions = this.options,
            outer = this.widget(),
            hasIcon = this._hasIcon();

        if ( options.disabled !== undefined ) {
            this.input.prop( "disabled", !!options.disabled );
            outer.toggleClass( "ui-state-disabled", !!options.disabled );
        }
        if ( options.mini !== undefined ) {
            outer.toggleClass( "ui-mini", !!options.mini );
        }
        if ( options.theme !== undefined ) {
            label
                .removeClass( "ui-btn-" + currentOptions.theme )
                .addClass( "ui-btn-" + options.theme );
        }
        if ( options.wrapperClass !== undefined ) {
            outer
```

```
                .removeClass( currentOptions.wrapperClass )
                .addClass( options.wrapperClass );
        }
        if ( options.iconpos !== undefined && hasIcon ) {
            label.removeClass( "ui-btn-icon-" + currentOptions.iconpos ).addClass(
            "ui-btn-icon-" + options.iconpos );
        } else if ( !hasIcon ) {
            label.removeClass( "ui-btn-icon-" + currentOptions.iconpos );
        }
        this._super( options );
    }

}, $.mobile.behaviors.formReset ) );

})( jQuery );

(function( $, undefined ) {

$.widget( "mobile.button", {

    initSelector: "input[type='button'], input[type='submit'], input[type='reset']",

    options: {
        theme: null,
        icon: null,
        iconpos: "left",
        iconshadow: false, /* TODO: Deprecated in 1.4, remove in 1.5. */
        corners: true,
        shadow: true,
        inline: null,
        mini: null,
        wrapperClass: null,
        enhanced: false
    },

    _create: function() {

        if ( this.element.is( ":disabled" ) ) {
            this.options.disabled = true;
        }

        if ( !this.options.enhanced ) {
            this._enhance();
        }

        $.extend( this, {
            wrapper: this.element.parent()
        });

        this._on( {
            focus: function() {
                this.widget().addClass( $.mobile.focusClass );
            },
```

```
            blur: function() {
                this.widget().removeClass( $.mobile.focusClass );
            }
        });

        this.refresh( true );
    },

    _enhance: function() {
        this.element.wrap( this._button() );
    },

    _button: function() {
        var options = this.options,
            iconClasses = this._getIconClasses( this.options );

        return $("<div class='ui-btn ui-input-btn" +
            ( options.wrapperClass ? " " + options.wrapperClass : "" ) +
            ( options.theme ? " ui-btn-" + options.theme : "" ) +
            ( options.corners ? " ui-corner-all" : "" ) +
            ( options.shadow ? " ui-shadow" : "" ) +
            ( options.inline ? " ui-btn-inline" : "" ) +
            ( options.mini ? " ui-mini" : "" ) +
            ( options.disabled ? " ui-state-disabled" : "" ) +
            ( iconClasses ? ( " " + iconClasses ) : "" ) +
            "' >" + this.element.val() + "</div>" );
    },

    widget: function() {
        return this.wrapper;
    },

    _destroy: function() {
            this.element.insertBefore( this.button );
            this.button.remove();
    },

    _getIconClasses: function( options ) {
        return ( options.icon ? ( "ui-icon-" + options.icon +
            ( options.iconshadow ? " ui-shadow-icon" : "" ) + /* TODO: Deprecated in 1.4,
            remove in 1.5. */
            " ui-btn-icon-" + options.iconpos ) : "" );
    },

    _setOptions: function( options ) {
        var outer = this.widget();

        if ( options.theme !== undefined ) {
            outer
                .removeClass( this.options.theme )
                .addClass( "ui-btn-" + options.theme );
        }
        if ( options.corners !== undefined ) {
            outer.toggleClass( "ui-corner-all", options.corners );
```

```
        }
        if ( options.shadow !== undefined ) {
            outer.toggleClass( "ui-shadow", options.shadow );
        }
        if ( options.inline !== undefined ) {
            outer.toggleClass( "ui-btn-inline", options.inline );
        }
        if ( options.mini !== undefined ) {
            outer.toggleClass( "ui-mini", options.mini );
        }
        if ( options.disabled !== undefined ) {
            this.element.prop( "disabled", options.disabled );
            outer.toggleClass( "ui-state-disabled", options.disabled );
        }

        if ( options.icon !== undefined ||
                options.iconshadow !== undefined || /* TODO: Deprecated in 1.4, remove in 1.5. */
                options.iconpos !== undefined ) {
            outer
                .removeClass( this._getIconClasses( this.options ) )
                .addClass( this._getIconClasses(
                    $.extend( {}, this.options, options ) ) );
        }

        this._super( options );
    },

    refresh: function( create ) {
        var originalElement,
            isDisabled = this.element.prop( "disabled" );

        if ( this.options.icon && this.options.iconpos === "notext" && this.element.attr(
        "title" ) ) {
            this.element.attr( "title", this.element.val() );
        }
        if ( !create ) {
            originalElement = this.element.detach();
            $( this.wrapper ).text( this.element.val() ).append( originalElement );
        }
        if ( this.options.disabled !== isDisabled ) {
            this._setOptions({ disabled: isDisabled });
        }
    }
});

})( jQuery );

(function( $ ) {
    var meta = $( "meta[name=viewport]" ),
        initialContent = meta.attr( "content" ),
        disabledZoom = initialContent + ",maximum-scale=1, user-scalable=no",
        enabledZoom = initialContent + ",maximum-scale=10, user-scalable=yes",
        disabledInitially = /(user-scalable[\s]*=[\s]*no)|(maximum-scale[\s]*=[\s]*1)[$,\s]/.
        test( initialContent );
```

```
    $.mobile.zoom = $.extend( {}, {
        enabled: !disabledInitially,
        locked: false,
        disable: function( lock ) {
            if ( !disabledInitially && !$.mobile.zoom.locked ) {
                meta.attr( "content", disabledZoom );
                $.mobile.zoom.enabled = false;
                $.mobile.zoom.locked = lock || false;
            }
        },
        enable: function( unlock ) {
            if ( !disabledInitially && ( !$.mobile.zoom.locked || unlock === true ) ) {
                meta.attr( "content", enabledZoom );
                $.mobile.zoom.enabled = true;
                $.mobile.zoom.locked = false;
            }
        },
        restore: function() {
            if ( !disabledInitially ) {
                meta.attr( "content", initialContent );
                $.mobile.zoom.enabled = true;
            }
        }
    });

}( jQuery ));

(function( $, undefined ) {

$.widget( "mobile.textinput", {
    initSelector: "input[type='text']," +
        "input[type='search']," +
        ":jqmData(type='search')," +
        "input[type='number']," +
        ":jqmData(type='number')," +
        "input[type='password']," +
        "input[type='email']," +
        "input[type='url']," +
        "input[type='tel']," +
        "textarea," +
        "input[type='time']," +
        "input[type='date']," +
        "input[type='month']," +
        "input[type='week']," +
        "input[type='datetime']," +
        "input[type='datetime-local']," +
        "input[type='color']," +
        "input:not([type])," +
        "input[type='file']",

    options: {
        theme: null,
        corners: true,
```

```
        mini: false,
        // This option defaults to true on iOS devices.
        preventFocusZoom: /iPhone|iPad|iPod/.test( navigator.platform ) && navigator.userAgent.
        indexOf( "AppleWebKit" ) > -1,
        wrapperClass: "",
        enhanced: false
    },

    _create: function() {

        var options = this.options,
            isSearch = this.element.is( "[type='search'], :jqmData(type='search')" ),
            isTextarea = this.element[ 0 ].tagName === "TEXTAREA",
            isRange = this.element.is( "[data-" + ( $.mobile.ns || "" ) + "type='range']" ),
            inputNeedsWrap = ( (this.element.is( "input" ) ||
                this.element.is( "[data-" + ( $.mobile.ns || "" ) + "type='search']" ) ) &&
                    !isRange );

        if ( this.element.prop( "disabled" ) ) {
            options.disabled = true;
        }

        $.extend( this, {
            classes: this._classesFromOptions(),
            isSearch: isSearch,
            isTextarea: isTextarea,
            isRange: isRange,
            inputNeedsWrap: inputNeedsWrap
        });

        this._autoCorrect();

        if ( !options.enhanced ) {
            this._enhance();
        }

        this._on( {
            "focus": "_handleFocus",
            "blur": "_handleBlur"
        });

    },

    refresh: function() {
        this.setOptions({
            "disabled" : this.element.is( ":disabled" )
        });
    },

    _enhance: function() {
        var elementClasses = [];

        if ( this.isTextarea ) {
            elementClasses.push( "ui-input-text" );
```

```
        }

        if ( this.isTextarea || this.isRange ) {
            elementClasses.push( "ui-shadow-inset" );
        }

        //"search" and "text" input widgets
        if ( this.inputNeedsWrap ) {
            this.element.wrap( this._wrap() );
        } else {
            elementClasses = elementClasses.concat( this.classes );
        }

        this.element.addClass( elementClasses.join( " " ) );
    },

    widget: function() {
        return ( this.inputNeedsWrap ) ? this.element.parent() : this.element;
    },

    _classesFromOptions: function() {
        var options = this.options,
            classes = [];

        classes.push( "ui-body-" + ( ( options.theme === null ) ? "inherit" : options.theme ) );
        if ( options.corners ) {
            classes.push( "ui-corner-all" );
        }
        if ( options.mini ) {
            classes.push( "ui-mini" );
        }
        if ( options.disabled ) {
            classes.push( "ui-state-disabled" );
        }
        if ( options.wrapperClass ) {
            classes.push( options.wrapperClass );
        }

        return classes;
    },

    _wrap: function() {
        return $( "<div class='" +
            ( this.isSearch ? "ui-input-search " : "ui-input-text " ) +
            this.classes.join( " " ) + " " +
            "ui-shadow-inset'></div>" );
    },

    _autoCorrect: function() {
        // XXX: Temporary workaround for issue 785 (Apple bug 8910589).
        //      Turn off autocorrect and autocomplete on non-iOS 5 devices
        //      since the popup they use can't be dismissed by the user. Note
        //      that we test for the presence of the feature by looking for
        //      the autocorrect property on the input element. We currently
```

```
        //      have no test for iOS 5 or newer so we're temporarily using
        //      the touchOverflow support flag for jQM 1.0. Yes, I feel dirty.
        //      - jblas
        if ( typeof this.element[0].autocorrect !== "undefined" &&
            !$.support.touchOverflow ) {

            // Set the attribute instead of the property just in case there
            // is code that attempts to make modifications via HTML.
            this.element[0].setAttribute( "autocorrect", "off" );
            this.element[0].setAttribute( "autocomplete", "off" );
        }
    },

    _handleBlur: function() {
        this.widget().removeClass( $.mobile.focusClass );
        if ( this.options.preventFocusZoom ) {
            $.mobile.zoom.enable( true );
        }
    },

    _handleFocus: function() {
        // In many situations, iOS will zoom into the input upon tap, this
        // prevents that from happening
        if ( this.options.preventFocusZoom ) {
            $.mobile.zoom.disable( true );
        }
        this.widget().addClass( $.mobile.focusClass );
    },

    _setOptions: function ( options ) {
        var outer = this.widget();

        this._super( options );

        if ( !( options.disabled === undefined &&
            options.mini === undefined &&
            options.corners === undefined &&
            options.theme === undefined &&
            options.wrapperClass === undefined ) ) {

            outer.removeClass( this.classes.join( " " ) );
            this.classes = this._classesFromOptions();
            outer.addClass( this.classes.join( " " ) );
        }

        if ( options.disabled !== undefined ) {
            this.element.prop( "disabled", !!options.disabled );
        }
    },

    _destroy: function() {
        if ( this.options.enhanced ) {
            return;
        }
```

```
        if ( this.inputNeedsWrap ) {
            this.element.unwrap();
        }
        this.element.removeClass( "ui-input-text " + this.classes.join( " " ) );
    }
});

})( jQuery );

(function( $, undefined ) {

$.widget( "mobile.slider", $.extend( {
    initSelector: "input[type='range'], :jqmData(type='range'), :jqmData(role='slider')",

    widgetEventPrefix: "slide",

    options: {
        theme: null,
        trackTheme: null,
        corners: true,
        mini: false,
        highlight: false
    },

    _create: function() {

        // TODO: Each of these should have comments explain what they're for
        var self = this,
            control = this.element,
            trackTheme = this.options.trackTheme || $.mobile.getAttribute( control[ 0 ], "theme"
            ),
            trackThemeClass = trackTheme ? " ui-bar-" + trackTheme : " ui-bar-inherit",
            cornerClass = ( this.options.corners || control.jqmData( "corners" ) ) ? "
            ui-corner-all" : "",
            miniClass = ( this.options.mini || control.jqmData( "mini" ) ) ? " ui-mini" : "",
            cType = control[ 0 ].nodeName.toLowerCase(),
            isToggleSwitch = ( cType === "select" ),
            isRangeslider = control.parent().is( ":jqmData(role='rangeslider')" ),
            selectClass = ( isToggleSwitch ) ? "ui-slider-switch" : "",
            controlID = control.attr( "id" ),
            $label = $( "[for='" + controlID + "']" ),
            labelID = $label.attr( "id" ) || controlID + "-label",
            min = !isToggleSwitch ? parseFloat( control.attr( "min" ) ) : 0,
            max =  !isToggleSwitch ? parseFloat( control.attr( "max" ) ) : control.find(
            "option" ).length-1,
            step = window.parseFloat( control.attr( "step" ) || 1 ),
            domHandle = document.createElement( "a" ),
            handle = $( domHandle ),
            domSlider = document.createElement( "div" ),
            slider = $( domSlider ),
            valuebg = this.options.highlight && !isToggleSwitch ? (function() {
                var bg = document.createElement( "div" );
                bg.className = "ui-slider-bg " + $.mobile.activeBtnClass;
                return $( bg ).prependTo( slider );
```

```
            })() : false,
            options,
            wrapper,
            j, length,
            i, optionsCount, origTabIndex,
            side, activeClass, sliderImg;

        $label.attr( "id", labelID );
        this.isToggleSwitch = isToggleSwitch;

        domHandle.setAttribute( "href", "#" );
        domSlider.setAttribute( "role", "application" );
        domSlider.className = [ this.isToggleSwitch ? "ui-slider ui-slider-track
        ui-shadow-inset " : "ui-slider-track ui-shadow-inset ", selectClass, trackThemeClass,
        cornerClass, miniClass ].join( "" );
        domHandle.className = "ui-slider-handle";
        domSlider.appendChild( domHandle );

        handle.attr({
            "role": "slider",
            "aria-valuemin": min,
            "aria-valuemax": max,
            "aria-valuenow": this._value(),
            "aria-valuetext": this._value(),
            "title": this._value(),
            "aria-labelledby": labelID
        });

        $.extend( this, {
            slider: slider,
            handle: handle,
            control: control,
            type: cType,
            step: step,
            max: max,
            min: min,
            valuebg: valuebg,
            isRangeslider: isRangeslider,
            dragging: false,
            beforeStart: null,
            userModified: false,
            mouseMoved: false
        });

        if ( isToggleSwitch ) {
            // TODO: restore original tabindex (if any) in a destroy method
            origTabIndex = control.attr( "tabindex" );
            if ( origTabIndex ) {
                handle.attr( "tabindex", origTabIndex );
            }
            control.attr( "tabindex", "-1" ).focus(function() {
                $( this ).blur();
                handle.focus();
            });
```

```
            wrapper = document.createElement( "div" );
            wrapper.className = "ui-slider-inneroffset";

            for ( j = 0, length = domSlider.childNodes.length; j < length; j++ ) {
                wrapper.appendChild( domSlider.childNodes[j] );
            }

            domSlider.appendChild( wrapper );

            // slider.wrapInner( "<div class='ui-slider-inneroffset'></div>" );

            // make the handle move with a smooth transition
            handle.addClass( "ui-slider-handle-snapping" );

            options = control.find( "option" );

            for ( i = 0, optionsCount = options.length; i < optionsCount; i++ ) {
                side = !i ? "b" : "a";
                activeClass = !i ? "" : " " + $.mobile.activeBtnClass;
                sliderImg = document.createElement( "span" );

                sliderImg.className = [ "ui-slider-label ui-slider-label-", side, activeClass ].
                join( "" );
                sliderImg.setAttribute( "role", "img" );
                sliderImg.appendChild( document.createTextNode( options[i].innerHTML ) );
                $( sliderImg ).prependTo( slider );
            }

            self._labels = $( ".ui-slider-label", slider );

        }

        // monitor the input for updated values
        control.addClass( isToggleSwitch ? "ui-slider-switch" : "ui-slider-input" );

        this._on( control, {
            "change": "_controlChange",
            "keyup": "_controlKeyup",
            "blur": "_controlBlur",
            "vmouseup": "_controlVMouseUp"
        });

        slider.bind( "vmousedown", $.proxy( this._sliderVMouseDown, this ) )
            .bind( "vclick", false );

        // We have to instantiate a new function object for the unbind to work properly
        // since the method itself is defined in the prototype (causing it to unbind everything)
        this._on( document, { "vmousemove": "_preventDocumentDrag" });
        this._on( slider.add( document ), { "vmouseup": "_sliderVMouseUp" });

        slider.insertAfter( control );

        // wrap in a div for styling purposes
```

```
        if ( !isToggleSwitch && !isRangeslider ) {
            wrapper = this.options.mini ? "<div class='ui-slider ui-mini'>" : "<div
            class='ui-slider'>";

            control.add( slider ).wrapAll( wrapper );
        }

        // bind the handle event callbacks and set the context to the widget instance
        this._on( this.handle, {
            "vmousedown": "_handleVMouseDown",
            "keydown": "_handleKeydown",
            "keyup": "_handleKeyup"
        });

        this.handle.bind( "vclick", false );

        this._handleFormReset();

        this.refresh( undefined, undefined, true );
    },

    _setOptions: function( options ) {
        if ( options.theme !== undefined ) {
            this._setTheme( options.theme );
        }

        if ( options.trackTheme !== undefined ) {
            this._setTrackTheme( options.trackTheme );
        }

        if ( options.corners !== undefined ) {
            this._setCorners( options.corners );
        }

        if ( options.mini !== undefined ) {
            this._setMini( options.mini );
        }

        if ( options.highlight !== undefined ) {
            this._setHighlight( options.highlight );
        }

        if ( options.disabled !== undefined ) {
            this._setDisabled( options.disabled );
        }
        this._super( options );
    },

    _controlChange: function( event ) {
        // if the user dragged the handle, the "change" event was triggered from inside
        refresh(); don't call refresh() again
        if ( this._trigger( "controlchange", event ) === false ) {
            return false;
        }
```

```
        if ( !this.mouseMoved ) {
            this.refresh( this._value(), true );
        }
    },

    _controlKeyup: function(/* event */) { // necessary?
        this.refresh( this._value(), true, true );
    },

    _controlBlur: function(/* event */) {
        this.refresh( this._value(), true );
    },

    // it appears the clicking the up and down buttons in chrome on
    // range/number inputs doesn't trigger a change until the field is
    // blurred. Here we check thif the value has changed and refresh
    _controlVMouseUp: function(/* event */) {
        this._checkedRefresh();
    },

    // NOTE force focus on handle
    _handleVMouseDown: function(/* event */) {
        this.handle.focus();
    },

    _handleKeydown: function( event ) {
        var index = this._value();
        if ( this.options.disabled ) {
            return;
        }

        // In all cases prevent the default and mark the handle as active
        switch ( event.keyCode ) {
            case $.mobile.keyCode.HOME:
            case $.mobile.keyCode.END:
            case $.mobile.keyCode.PAGE_UP:
            case $.mobile.keyCode.PAGE_DOWN:
            case $.mobile.keyCode.UP:
            case $.mobile.keyCode.RIGHT:
            case $.mobile.keyCode.DOWN:
            case $.mobile.keyCode.LEFT:
                event.preventDefault();

                if ( !this._keySliding ) {
                    this._keySliding = true;
                    this.handle.addClass( "ui-state-active" ); /* TODO: We don't use this class
                    for styling. Do we need to add it? */
                }

                break;
        }

        // move the slider according to the keypress
        switch ( event.keyCode ) {
```

```
            case $.mobile.keyCode.HOME:
                this.refresh( this.min );
                break;
            case $.mobile.keyCode.END:
                this.refresh( this.max );
                break;
            case $.mobile.keyCode.PAGE_UP:
            case $.mobile.keyCode.UP:
            case $.mobile.keyCode.RIGHT:
                this.refresh( index + this.step );
                break;
            case $.mobile.keyCode.PAGE_DOWN:
            case $.mobile.keyCode.DOWN:
            case $.mobile.keyCode.LEFT:
                this.refresh( index - this.step );
                break;
        }
    }, // remove active mark

    _handleKeyup: function(/* event */) {
        if ( this._keySliding ) {
            this._keySliding = false;
            this.handle.removeClass( "ui-state-active" ); /* See comment above. */
        }
    },

    _sliderVMouseDown: function( event ) {
        // NOTE: we don't do this in refresh because we still want to
        //       support programmatic alteration of disabled inputs
        if ( this.options.disabled || !( event.which === 1 || event.which === 0 || event.which
        === undefined ) ) {
            return false;
        }
        if ( this._trigger( "beforestart", event ) === false ) {
            return false;
        }
        this.dragging = true;
        this.userModified = false;
        this.mouseMoved = false;

        if ( this.isToggleSwitch ) {
            this.beforeStart = this.element[0].selectedIndex;
        }

        this.refresh( event );
        this._trigger( "start" );
        return false;
    },

    _sliderVMouseUp: function() {
        if ( this.dragging ) {
            this.dragging = false;

            if ( this.isToggleSwitch ) {
```

```
                // make the handle move with a smooth transition
                this.handle.addClass( "ui-slider-handle-snapping" );

                if ( this.mouseMoved ) {
                    // this is a drag, change the value only if user dragged enough
                    if ( this.userModified ) {
                        this.refresh( this.beforeStart === 0 ? 1 : 0 );
                    } else {
                        this.refresh( this.beforeStart );
                    }
                } else {
                    // this is just a click, change the value
                    this.refresh( this.beforeStart === 0 ? 1 : 0 );
                }
            }

            this.mouseMoved = false;
            this._trigger( "stop" );
            return false;
        }
    },

    _preventDocumentDrag: function( event ) {
            // NOTE: we don't do this in refresh because we still want to
            //       support programmatic alteration of disabled inputs
            if ( this._trigger( "drag", event ) === false) {
                return false;
            }
            if ( this.dragging && !this.options.disabled ) {

                // this.mouseMoved must be updated before refresh() because it will be used in
                the control "change" event
                this.mouseMoved = true;

                if ( this.isToggleSwitch ) {
                    // make the handle move in sync with the mouse
                    this.handle.removeClass( "ui-slider-handle-snapping" );
                }

                this.refresh( event );

                // only after refresh() you can calculate this.userModified
                this.userModified = this.beforeStart !== this.element[0].selectedIndex;
                return false;
            }
        },

    _checkedRefresh: function() {
        if ( this.value !== this._value() ) {
            this.refresh( this._value() );
        }
    },

    _value: function() {
```

```
        return  this.isToggleSwitch ? this.element[0].selectedIndex : parseFloat( this.element.
        val() ) ;
    },

    _reset: function() {
        this.refresh( undefined, false, true );
    },

    refresh: function( val, isfromControl, preventInputUpdate ) {
        // NOTE: we don't return here because we want to support programmatic
        //       alteration of the input value, which should still update the slider

        var self = this,
            parentTheme = $.mobile.getAttribute( this.element[ 0 ], "theme" ),
            theme = this.options.theme || parentTheme,
            themeClass =  theme ? " ui-btn-" + theme : "",
            trackTheme = this.options.trackTheme || parentTheme,
            trackThemeClass = trackTheme ? " ui-bar-" + trackTheme : " ui-bar-inherit",
            cornerClass = this.options.corners ? " ui-corner-all" : "",
            miniClass = this.options.mini ? " ui-mini" : "",
            left, width, data, tol,
            pxStep, percent,
            control, isInput, optionElements, min, max, step,
            newval, valModStep, alignValue, percentPerStep,
            handlePercent, aPercent, bPercent,
            valueChanged;

        self.slider[0].className = [ this.isToggleSwitch ? "ui-slider ui-slider-switch
        ui-slider-track ui-shadow-inset" : "ui-slider-track ui-shadow-inset", trackThemeClass,
        cornerClass, miniClass ].join( "" );
        if ( this.options.disabled || this.element.prop( "disabled" ) ) {
            this.disable();
        }

        // set the stored value for comparison later
        this.value = this._value();
        if ( this.options.highlight && !this.isToggleSwitch && this.slider.find( ".ui-slider-bg"
         ).length === 0 ) {
            this.valuebg = (function() {
                var bg = document.createElement( "div" );
                bg.className = "ui-slider-bg " + $.mobile.activeBtnClass;
                return $( bg ).prependTo( self.slider );
            })();
        }
        this.handle.addClass( "ui-btn" + themeClass + " ui-shadow" );

        control = this.element;
        isInput = !this.isToggleSwitch;
        optionElements = isInput ? [] : control.find( "option" );
        min =  isInput ? parseFloat( control.attr( "min" ) ) : 0;
        max = isInput ? parseFloat( control.attr( "max" ) ) : optionElements.length - 1;
        step = ( isInput && parseFloat( control.attr( "step" ) ) > 0 ) ? parseFloat( control.
        attr( "step" ) ) : 1;
```

```
        if ( typeof val === "object" ) {
            data = val;
            // a slight tolerance helped get to the ends of the slider
            tol = 8;

            left = this.slider.offset().left;
            width = this.slider.width();
            pxStep = width/((max-min)/step);
            if ( !this.dragging ||
                    data.pageX < left - tol ||
                    data.pageX > left + width + tol ) {
                return;
            }
            if ( pxStep > 1 ) {
                percent = ( ( data.pageX - left ) / width ) * 100;
            } else {
                percent = Math.round( ( ( data.pageX - left ) / width ) * 100 );
            }
        } else {
            if ( val == null ) {
                val = isInput ? parseFloat( control.val() || 0 ) : control[0].selectedIndex;
            }
            percent = ( parseFloat( val ) - min ) / ( max - min ) * 100;
        }

        if ( isNaN( percent ) ) {
            return;
        }

        newval = ( percent / 100 ) * ( max - min ) + min;

        //from jQuery UI slider, the following source will round to the nearest step
        valModStep = ( newval - min ) % step;
        alignValue = newval - valModStep;

        if ( Math.abs( valModStep ) * 2 >= step ) {
            alignValue += ( valModStep > 0 ) ? step : ( -step );
        }

        percentPerStep = 100/((max-min)/step);
        // Since JavaScript has problems with large floats, round
        // the final value to 5 digits after the decimal point (see jQueryUI: #4124)
        newval = parseFloat( alignValue.toFixed(5) );

        if ( typeof pxStep === "undefined" ) {
            pxStep = width / ( (max-min) / step );
        }
        if ( pxStep > 1 && isInput ) {
            percent = ( newval - min ) * percentPerStep * ( 1 / step );
        }
        if ( percent < 0 ) {
            percent = 0;
        }
```

```
        if ( percent > 100 ) {
            percent = 100;
        }

        if ( newval < min ) {
            newval = min;
        }

        if ( newval > max ) {
            newval = max;
        }

        this.handle.css( "left", percent + "%" );

        this.handle[0].setAttribute( "aria-valuenow", isInput ? newval : optionElements.eq(
        newval ).attr( "value" ) );

        this.handle[0].setAttribute( "aria-valuetext", isInput ? newval : optionElements.eq(
        newval ).getEncodedText() );

        this.handle[0].setAttribute( "title", isInput ? newval : optionElements.eq( newval ).
        getEncodedText() );

        if ( this.valuebg ) {
            this.valuebg.css( "width", percent + "%" );
        }

        // drag the label widths
        if ( this._labels ) {
            handlePercent = this.handle.width() / this.slider.width() * 100;
            aPercent = percent && handlePercent + ( 100 - handlePercent ) * percent / 100;
            bPercent = percent === 100 ? 0 : Math.min( handlePercent + 100 - aPercent, 100 );

            this._labels.each(function() {
                var ab = $( this ).hasClass( "ui-slider-label-a" );
                $( this ).width( ( ab ? aPercent : bPercent  ) + "%" );
            });
        }

        if ( !preventInputUpdate ) {
            valueChanged = false;

            // update control"s value
            if ( isInput ) {
                valueChanged = control.val() !== newval;
                control.val( newval );
            } else {
                valueChanged = control[ 0 ].selectedIndex !== newval;
                control[ 0 ].selectedIndex = newval;
            }
            if ( this._trigger( "beforechange", val ) === false) {
                    return false;
            }
            if ( !isfromControl && valueChanged ) {
```

```
                control.trigger( "change" );
            }
        }
    },

    _setHighlight: function( value ) {
        value = !!value;
        if ( value ) {
            this.options.highlight = !!value;
            this.refresh();
        } else if ( this.valuebg ) {
            this.valuebg.remove();
            this.valuebg = false;
        }
    },

    _setTheme: function( value ) {
        this.handle
            .removeClass( "ui-btn-" + this.options.theme )
            .addClass( "ui-btn-" + value );

        var currentTheme = this.options.theme ? this.options.theme : "inherit",
            newTheme = value ? value : "inherit";

        this.control
            .removeClass( "ui-body-" + currentTheme )
            .addClass( "ui-body-" + newTheme );
    },

    _setTrackTheme: function( value ) {
        var currentTrackTheme = this.options.trackTheme ? this.options.trackTheme : "inherit",
            newTrackTheme = value ? value : "inherit";

        this.slider
            .removeClass( "ui-body-" + currentTrackTheme )
            .addClass( "ui-body-" + newTrackTheme );
    },

    _setMini: function( value ) {
        value = !!value;
        if ( !this.isToggleSwitch && !this.isRangeslider ) {
            this.slider.parent().toggleClass( "ui-mini", value );
            this.element.toggleClass( "ui-mini", value );
        }
        this.slider.toggleClass( "ui-mini", value );
    },

    _setCorners: function( value ) {
        this.slider.toggleClass( "ui-corner-all", value );

        if ( !this.isToggleSwitch ) {
            this.control.toggleClass( "ui-corner-all", value );
        }
    },
```

```
    _setDisabled: function( value ) {
        value = !!value;
        this.element.prop( "disabled", value );
        this.slider
            .toggleClass( "ui-state-disabled", value )
            .attr( "aria-disabled", value );
    }

}, $.mobile.behaviors.formReset ) );

})( jQuery );

(function( $, undefined ) {

var popup;

function getPopup() {
    if ( !popup ) {
        popup = $( "<div></div>", {
            "class": "ui-slider-popup ui-shadow ui-corner-all"
        });
    }
    return popup.clone();
}

$.widget( "mobile.slider", $.mobile.slider, {
    options: {
        popupEnabled: false,
        showValue: false
    },

    _create: function() {
        this._super();

        $.extend( this, {
            _currentValue: null,
            _popup: null,
            _popupVisible: false
        });

        this._setOption( "popupEnabled", this.options.popupEnabled );

        this._on( this.handle, { "vmousedown" : "_showPopup" } );
        this._on( this.slider.add( this.document ), { "vmouseup" : "_hidePopup" } );
        this._refresh();
    },

    // position the popup centered 5px above the handle
    _positionPopup: function() {
        var dstOffset = this.handle.offset();

        this._popup.offset( {
            left: dstOffset.left + ( this.handle.width() - this._popup.width() ) / 2,
```

```
            top: dstOffset.top - this._popup.outerHeight() - 5
        });
    },

    _setOption: function( key, value ) {
        this._super( key, value );

        if ( key === "showValue" ) {
            this.handle.html( value && !this.options.mini ? this._value() : "" );
        } else if ( key === "popupEnabled" ) {
            if ( value && !this._popup ) {
                this._popup = getPopup()
                    .addClass( "ui-body-" + ( this.options.theme || "a" ) )
                    .hide()
                    .insertBefore( this.element );
            }
        }
    },

    // show value on the handle and in popup
    refresh: function() {
        this._super.apply( this, arguments );
        this._refresh();
    },

    _refresh: function() {
        var o = this.options, newValue;

        if ( o.popupEnabled ) {
            // remove the title attribute from the handle (which is
            // responsible for the annoying tooltip); NB we have
            // to do it here as the jqm slider sets it every time
            // the slider's value changes :(
            this.handle.removeAttr( "title" );
        }

        newValue = this._value();
        if ( newValue === this._currentValue ) {
            return;
        }
        this._currentValue = newValue;

        if ( o.popupEnabled && this._popup ) {
            this._positionPopup();
            this._popup.html( newValue );
        } else if ( o.showValue && !this.options.mini ) {
            this.handle.html( newValue );
        }
    },

    _showPopup: function() {
        if ( this.options.popupEnabled && !this._popupVisible ) {
            this.handle.html( "" );
            this._popup.show();
```

```
            this._positionPopup();
            this._popupVisible = true;
        }
    },

    _hidePopup: function() {
        var o = this.options;

        if ( o.popupEnabled && this._popupVisible ) {
            if ( o.showValue && !o.mini ) {
                this.handle.html( this._value() );
            }
            this._popup.hide();
            this._popupVisible = false;
        }
    }
});

})( jQuery );

(function( $, undefined ) {

$.widget( "mobile.flipswitch", $.extend({

    options: {
        onText: "On",
        offText: "Off",
        theme: null,
        enhanced: false,
        wrapperClass: null,
        corners: true,
        mini: false
    },

    _create: function() {
            if ( !this.options.enhanced ) {
                this._enhance();
            } else {
                $.extend( this, {
                    flipswitch: this.element.parent(),
                    on: this.element.find( ".ui-flipswitch-on" ).eq( 0 ),
                    off: this.element.find( ".ui-flipswitch-off" ).eq(0),
                    type: this.element.get( 0 ).tagName
                });
            }

            this._handleFormReset();

            // Transfer tabindex to "on" element and make input unfocusable
            this._originalTabIndex = this.element.attr( "tabindex" );
            if ( this._originalTabIndex != null ) {
                this.on.attr( "tabindex", this._originalTabIndex );
            }
            this.element.attr( "tabindex", "-1" );
```

```
            this._on({
                "focus" : "_handleInputFocus"
            });

            if ( this.element.is( ":disabled" ) ) {
                this._setOptions({
                    "disabled": true
                });
            }

            this._on( this.flipswitch, {
                "click": "_toggle",
                "swipeleft": "_left",
                "swiperight": "_right"
            });

            this._on( this.on, {
                "keydown": "_keydown"
            });

            this._on( {
                "change": "refresh"
            });
    },

    _handleInputFocus: function() {
        this.on.focus();
    },

    widget: function() {
        return this.flipswitch;
    },

    _left: function() {
        this.flipswitch.removeClass( "ui-flipswitch-active" );
        if ( this.type === "SELECT" ) {
            this.element.get( 0 ).selectedIndex = 0;
        } else {
            this.element.prop( "checked", false );
        }
        this.element.trigger( "change" );
    },

    _right: function() {
        this.flipswitch.addClass( "ui-flipswitch-active" );
        if ( this.type === "SELECT" ) {
            this.element.get( 0 ).selectedIndex = 1;
        } else {
            this.element.prop( "checked", true );
        }
        this.element.trigger( "change" );
    },

    _enhance: function() {
```

```
        var flipswitch = $( "<div>" ),
            options = this.options,
            element = this.element,
            theme = options.theme ? options.theme : "inherit",

            // The "on" button is an anchor so it's focusable
            on = $( "<a></a>", {
                "href": "#"
            }),
            off = $( "<span></span>" ),
            type = element.get( 0 ).tagName,
            onText = ( type === "INPUT" ) ?
                options.onText : element.find( "option" ).eq( 1 ).text(),
            offText = ( type === "INPUT" ) ?
                options.offText : element.find( "option" ).eq( 0 ).text();

            on
                .addClass( "ui-flipswitch-on ui-btn ui-shadow ui-btn-inherit" )
                .text( onText );
            off
                .addClass( "ui-flipswitch-off" )
                .text( offText );

            flipswitch
                .addClass( "ui-flipswitch ui-shadow-inset " +
                    "ui-bar-" + theme + " " +
                    ( options.wrapperClass ? options.wrapperClass : "" ) + " " +
                    ( ( element.is( ":checked" ) ||
                        element
                            .find( "option" )
                            .eq( 1 )
                            .is( ":selected" ) ) ? "ui-flipswitch-active" : "" ) +
                    ( element.is(":disabled") ? " ui-state-disabled": "") +
                    ( options.corners ? " ui-corner-all": "" ) +
                    ( options.mini ? " ui-mini": "" ) )
                .append( on, off );

            element
                .addClass( "ui-flipswitch-input" )
                .after( flipswitch )
                .appendTo( flipswitch );

        $.extend( this, {
            flipswitch: flipswitch,
            on: on,
            off: off,
            type: type
        });
    },

    _reset: function() {
        this.refresh();
    },
```

```
    refresh: function() {
        var direction,
            existingDirection = this.flipswitch.hasClass( "ui-flipswitch-active" ) ? "_right" :
            "_left";

        if ( this.type === "SELECT" ) {
            direction = ( this.element.get( 0 ).selectedIndex > 0 ) ? "_right": "_left";
        } else {
            direction = this.element.prop( "checked" ) ? "_right": "_left";
        }

        if ( direction !== existingDirection ) {
            this[ direction ]();
        }
    },

    _toggle: function() {
        var direction = this.flipswitch.hasClass( "ui-flipswitch-active" ) ? "_left" : "_right";

        this[ direction ]();
    },

    _keydown: function( e ) {
        if ( e.which === $.mobile.keyCode.LEFT ) {
            this._left();
        } else if ( e.which === $.mobile.keyCode.RIGHT ) {
            this._right();
        } else if ( e.which === $.mobile.keyCode.SPACE ) {
            this._toggle();
            e.preventDefault();
        }
    },

    _setOptions: function( options ) {
        if ( options.theme !== undefined ) {
            var currentTheme = options.theme ? options.theme : "inherit",
                newTheme = options.theme ? options.theme : "inherit";

            this.widget()
                .removeClass( "ui-bar-" + currentTheme )
                .addClass( "ui-bar-" + newTheme );
        }
        if ( options.onText !== undefined ) {
            this.on.text( options.onText );
        }
        if ( options.offText !== undefined ) {
            this.off.text( options.offText );
        }
        if ( options.disabled !== undefined ) {
            this.widget().toggleClass( "ui-state-disabled", options.disabled );
        }
        if ( options.mini !== undefined ) {
            this.widget().toggleClass( "ui-mini", options.mini );
        }
```

```
        if ( options.corners !== undefined ) {
            this.widget().toggleClass( "ui-corner-all", options.corners );
        }

        this._super( options );
    },

    _destroy: function() {
        if ( this.options.enhanced ) {
            return;
        }
        if ( this._originalTabIndex != null ) {
            this.element.attr( "tabindex", this._originalTabIndex );
        } else {
            this.element.removeAttr( "tabindex" );
        }
        this.on.remove();
        this.off.remove();
        this.element.unwrap();
        this.flipswitch.remove();
        this.removeClass( "ui-flipswitch-input" );
    }

}, $.mobile.behaviors.formReset ) );

})( jQuery );

(function( $, undefined ) {
    $.widget( "mobile.rangeslider", $.extend( {

        options: {
            theme: null,
            trackTheme: null,
            corners: true,
            mini: false,
            highlight: true
        },

        _create: function() {
            var $el = this.element,
            elClass = this.options.mini ? "ui-rangeslider ui-mini" : "ui-rangeslider",
            _inputFirst = $el.find( "input" ).first(),
            _inputLast = $el.find( "input" ).last(),
            _label = $el.find( "label" ).first(),
            _sliderWidgetFirst = $.data( _inputFirst.get( 0 ), "mobile-slider" ) ||
                $.data( _inputFirst.slider().get( 0 ), "mobile-slider" ),
            _sliderWidgetLast = $.data( _inputLast.get(0), "mobile-slider" ) ||
                $.data( _inputLast.slider().get( 0 ), "mobile-slider" ),
            _sliderFirst = _sliderWidgetFirst.slider,
            _sliderLast = _sliderWidgetLast.slider,
            firstHandle = _sliderWidgetFirst.handle,
            _sliders = $( "<div class='ui-rangeslider-sliders' />" ).appendTo( $el );

            _inputFirst.addClass( "ui-rangeslider-first" );
```

```
            _inputLast.addClass( "ui-rangeslider-last" );
            $el.addClass( elClass );

            _sliderFirst.appendTo( _sliders );
            _sliderLast.appendTo( _sliders );
            _label.insertBefore( $el );
            firstHandle.prependTo( _sliderLast );

            $.extend( this, {
                _inputFirst: _inputFirst,
                _inputLast: _inputLast,
                _sliderFirst: _sliderFirst,
                _sliderLast: _sliderLast,
                _label: _label,
                _targetVal: null,
                _sliderTarget: false,
                _sliders: _sliders,
                _proxy: false
            });

            this.refresh();
            this._on( this.element.find( "input.ui-slider-input" ), {
                "slidebeforestart": "_slidebeforestart",
                "slidestop": "_slidestop",
                "slidedrag": "_slidedrag",
                "slidebeforechange": "_change",
                "blur": "_change",
                "keyup": "_change"
            });
            this._on({
                "mousedown":"_change"
            });
            this._on( this.element.closest( "form" ), {
                "reset":"_handleReset"
            });
            this._on( firstHandle, {
                "vmousedown": "_dragFirstHandle"
            });
        },
        _handleReset: function() {
            var self = this;
            //we must wait for the stack to unwind before updateing other wise sliders will not
            have updated yet
            setTimeout( function() {
                self._updateHighlight();
            },0);
        },

        _dragFirstHandle: function( event ) {
            //if the first handle is dragged send the event to the first slider
            $.data( this._inputFirst.get(0), "mobile-slider" ).dragging = true;
            $.data( this._inputFirst.get(0), "mobile-slider" ).refresh( event );
            return false;
        },
```

```
        _slidedrag: function( event ) {
            var first = $( event.target ).is( this._inputFirst ),
                otherSlider = ( first ) ? this._inputLast : this._inputFirst;

            this._sliderTarget = false;
            //if the drag was initiated on an extreme and the other handle is focused send the
            events to
            //the closest handle
            if ( ( this._proxy === "first" && first ) || ( this._proxy === "last" && !first ) ) {
                $.data( otherSlider.get(0), "mobile-slider" ).dragging = true;
                $.data( otherSlider.get(0), "mobile-slider" ).refresh( event );
                return false;
            }
        },

        _slidestop: function( event ) {
            var first = $( event.target ).is( this._inputFirst );

            this._proxy = false;
            //this stops dragging of the handle and brings the active track to the front
            //this makes clicks on the track go the the last handle used
            this.element.find( "input" ).trigger( "vmouseup" );
            this._sliderFirst.css( "z-index", first ? 1 : "" );
        },

        _slidebeforestart: function( event ) {
            this._sliderTarget = false;
            //if the track is the target remember this and the original value
            if ( $( event.originalEvent.target ).hasClass( "ui-slider-track" ) ) {
                this._sliderTarget = true;
                this._targetVal = $( event.target ).val();
            }
        },

        _setOptions: function( options ) {
            if ( options.theme !== undefined ) {
                this._setTheme( options.theme );
            }

            if ( options.trackTheme !== undefined ) {
                this._setTrackTheme( options.trackTheme );
            }

            if ( options.mini !== undefined ) {
                this._setMini( options.mini );
            }

            if ( options.highlight !== undefined ) {
                this._setHighlight( options.highlight );
            }
            this._super( options );
            this.refresh();
        },
```

```
        refresh: function() {
            var $el = this.element,
                o = this.options;

            if ( this._inputFirst.is( ":disabled" ) || this._inputLast.is( ":disabled" ) ) {
                this.options.disabled = true;
            }

            $el.find( "input" ).slider({
                theme: o.theme,
                trackTheme: o.trackTheme,
                disabled: o.disabled,
                corners: o.corners,
                mini: o.mini,
                highlight: o.highlight
            }).slider( "refresh" );
            this._updateHighlight();
        },

        _change: function( event ) {
            if ( event.type === "keyup" ) {
                this._updateHighlight();
                return false;
            }

            var self = this,
                min = parseFloat( this._inputFirst.val(), 10 ),
                max = parseFloat( this._inputLast.val(), 10 ),
                first = $( event.target ).hasClass( "ui-rangeslider-first" ),
                thisSlider = first ? this._inputFirst : this._inputLast,
                otherSlider = first ? this._inputLast : this._inputFirst;

            if ( ( this._inputFirst.val() > this._inputLast.val() && event.type === "mousedown"
            && !$(event.target).hasClass("ui-slider-handle")) ) {
                thisSlider.blur();
            } else if ( event.type === "mousedown" ) {
                return;
            }
            if ( min > max && !this._sliderTarget ) {
                //this prevents min from being greater then max
                thisSlider.val( first ? max: min ).slider( "refresh" );
                this._trigger( "normalize" );
            } else if ( min > max ) {
                //this makes it so clicks on the target on either extreme go to the closest
                handle
                thisSlider.val( this._targetVal ).slider( "refresh" );

                //You must wait for the stack to unwind so first slider is updated before
                updating second
                setTimeout( function() {
                    otherSlider.val( first ? min: max ).slider( "refresh" );
                    $.data( otherSlider.get(0), "mobile-slider" ).handle.focus();
                    self._sliderFirst.css( "z-index", first ? "" : 1 );
```

```
                self._trigger( "normalize" );
            }, 0 );
            this._proxy = ( first ) ? "first" : "last";
        }
        //fixes issue where when both _sliders are at min they cannot be adjusted
        if ( min === max ) {
            $.data( thisSlider.get(0), "mobile-slider" ).handle.css( "z-index", 1 );
            $.data( otherSlider.get(0), "mobile-slider" ).handle.css( "z-index", 0 );
        } else {
            $.data( otherSlider.get(0), "mobile-slider" ).handle.css( "z-index", "" );
            $.data( thisSlider.get(0), "mobile-slider" ).handle.css( "z-index", "" );
        }

        this._updateHighlight();

        if ( min >= max ) {
            return false;
        }
    },

    _updateHighlight: function() {
        var min = parseInt( $.data( this._inputFirst.get(0), "mobile-slider" ).handle.get(0
        ).style.left, 10 ),
            max = parseInt( $.data( this._inputLast.get(0), "mobile-slider" ).handle.get(0).
            style.left, 10 ),
            width = (max - min);

        this.element.find( ".ui-slider-bg" ).css({
            "margin-left": min + "%",
            "width": width + "%"
        });
    },

    _setTheme: function( value ) {
        this._inputFirst.slider( "option", "theme", value );
        this._inputLast.slider( "option", "theme", value );
    },

    _setTrackTheme: function( value ) {
        this._inputFirst.slider( "option", "trackTheme", value );
        this._inputLast.slider( "option", "trackTheme", value );
    },

    _setMini: function( value ) {
        this._inputFirst.slider( "option", "mini", value );
        this._inputLast.slider( "option", "mini", value );
        this.element.toggleClass( "ui-mini", !!value );
    },

    _setHighlight: function( value ) {
        this._inputFirst.slider( "option", "highlight", value );
        this._inputLast.slider( "option", "highlight", value );
    },
```

```
        _destroy: function() {
            this._label.prependTo( this.element );
            this.element.removeClass( "ui-rangeslider ui-mini" );
            this._inputFirst.after( this._sliderFirst );
            this._inputLast.after( this._sliderLast );
            this._sliders.remove();
            this.element.find( "input" ).removeClass( "ui-rangeslider-first ui-rangeslider-last"
             ).slider( "destroy" );
        }

    }, $.mobile.behaviors.formReset ) );

})( jQuery );

(function( $, undefined ) {

    $.widget( "mobile.textinput", $.mobile.textinput, {
        options: {
            clearBtn: false,
            clearBtnText: "Clear text"
        },

        _create: function() {
            this._super();

            if ( !!this.options.clearBtn || this.isSearch ) {
                this._addClearBtn();
            }
        },

        clearButton: function() {

            return $( "<a href='#' class='ui-input-clear ui-btn ui-icon-delete
            ui-btn-icon-notext ui-corner-all" +
    "' title='" + this.options.clearBtnText + "'>" + this.options.clearBtnText + "</a>" );

        },

        _clearBtnClick: function( event ) {
            this.element.val( "" )
                    .focus()
                    .trigger( "change" );

            this._clearBtn.addClass( "ui-input-clear-hidden" );
            event.preventDefault();
        },

        _addClearBtn: function() {

            if ( !this.options.enhanced ) {
                this._enhanceClear();
            }

            $.extend( this, {
```

```
            _clearBtn: this.widget().find("a.ui-input-clear")
        });

        this._bindClearEvents();

        this._toggleClear();

    },

    _enhanceClear: function() {

        this.clearButton().appendTo( this.widget() );
        this.widget().addClass( "ui-input-has-clear" );

    },

    _bindClearEvents: function() {

        this._on( this._clearBtn, {
            "click": "_clearBtnClick"
        });

        this._on({
            "keyup": "_toggleClear",
            "change": "_toggleClear",
            "input": "_toggleClear",
            "focus": "_toggleClear",
            "blur": "_toggleClear",
            "cut": "_toggleClear",
            "paste": "_toggleClear"

        });

    },

    _unbindClear: function() {
        this._off( this._clearBtn, "click");
        this._off( this.element, "keyup change input focus blur cut paste" );
    },

    _setOptions: function( options ) {
        this._super( options );

        if ( options.clearBtn !== undefined &&
            !this.element.is( "textarea, :jqmData(type='range')" ) ) {
            if ( options.clearBtn ) {
                this._addClearBtn();
            } else {
                this._destroyClear();
            }
        }

        if ( options.clearBtnText !== undefined && this._clearBtn !== undefined ) {
            this._clearBtn.text( options.clearBtnText )
```

```
                    .attr("title", options.clearBtnText);
            }
        },

        _toggleClear: function() {
            this._delay( "_toggleClearClass", 0 );
        },

        _toggleClearClass: function() {
            this._clearBtn.toggleClass( "ui-input-clear-hidden", !this.element.val() );
        },

        _destroyClear: function() {
            this.widget().removeClass( "ui-input-has-clear" );
            this._unbindClear();
            this._clearBtn.remove();
        },

        _destroy: function() {
            this._super();
            this._destroyClear();
        }

    });

})( jQuery );

(function( $, undefined ) {

    $.widget( "mobile.textinput", $.mobile.textinput, {
        options: {
            autogrow:true,
            keyupTimeoutBuffer: 100
        },

        _create: function() {
            this._super();

            if ( this.options.autogrow && this.isTextarea ) {
                this._autogrow();
            }
        },

        _autogrow: function() {
            this.element.addClass( "ui-textinput-autogrow" );

            this._on({
                "keyup": "_timeout",
                "change": "_timeout",
                "input": "_timeout",
                "paste": "_timeout"
            });

            // Attach to the various you-have-become-visible notifications that the
```

```
            // various framework elements emit.
            // TODO: Remove all but the updatelayout handler once #6426 is fixed.
            this._on( true, this.document, {

                // TODO: Move to non-deprecated event
                "pageshow": "_handleShow",
                "popupbeforeposition": "_handleShow",
                "updatelayout": "_handleShow",
                "panelopen": "_handleShow"
            });
        },

        // Synchronously fix the widget height if this widget's parents are such
        // that they show/hide content at runtime. We still need to check whether
        // the widget is actually visible in case it is contained inside multiple
        // such containers. For example: panel contains collapsible contains
        // autogrow textinput. The panel may emit "panelopen" indicating that its
        // content has become visible, but the collapsible is still collapsed, so
        // the autogrow textarea is still not visible.
        _handleShow: function( event ) {
            if ( $.contains( event.target, this.element[ 0 ] ) &&
                this.element.is( ":visible" ) ) {

                if ( event.type !== "popupbeforeposition" ) {
                    this.element
                        .addClass( "ui-textinput-autogrow-resize" )
                        .animationComplete(
                            $.proxy( function() {
                                this.element.removeClass( "ui-textinput-autogrow-resize" );
                            }, this ),
                        "transition" );
                }
                this._timeout();
            }
        },

        _unbindAutogrow: function() {
            this.element.removeClass( "ui-textinput-autogrow" );
            this._off( this.element, "keyup change input paste" );
            this._off( this.document,
                "pageshow popupbeforeposition updatelayout panelopen" );
        },

        keyupTimeout: null,

        _prepareHeightUpdate: function( delay ) {
            if ( this.keyupTimeout ) {
                clearTimeout( this.keyupTimeout );
            }
            if ( delay === undefined ) {
                this._updateHeight();
            } else {
                this.keyupTimeout = this._delay( "_updateHeight", delay );
            }
```

```
    },

    _timeout: function() {
        this._prepareHeightUpdate( this.options.keyupTimeoutBuffer );
    },

    _updateHeight: function() {
        var paddingTop, paddingBottom, paddingHeight, scrollHeight, clientHeight,
            borderTop, borderBottom, borderHeight, height,
            scrollTop = this.window.scrollTop();
        this.keyupTimeout = 0;

        // IE8 textareas have the onpage property - others do not
        if ( !( "onpage" in this.element[ 0 ] ) ) {
            this.element.css({
                "height": 0,
                "min-height": 0,
                "max-height": 0
            });
        }

        scrollHeight = this.element[ 0 ].scrollHeight;
        clientHeight = this.element[ 0 ].clientHeight;
        borderTop = parseFloat( this.element.css( "border-top-width" ) );
        borderBottom = parseFloat( this.element.css( "border-bottom-width" ) );
        borderHeight = borderTop + borderBottom;
        height = scrollHeight + borderHeight + 15;

        // Issue 6179: Padding is not included in scrollHeight and
        // clientHeight by Firefox if no scrollbar is visible. Because
        // textareas use the border-box box-sizing model, padding should be
        // included in the new (assigned) height. Because the height is set
        // to 0, clientHeight == 0 in Firefox. Therefore, we can use this to
        // check if padding must be added.
        if ( clientHeight === 0 ) {
            paddingTop = parseFloat( this.element.css( "padding-top" ) );
            paddingBottom = parseFloat( this.element.css( "padding-bottom" ) );
            paddingHeight = paddingTop + paddingBottom;

            height += paddingHeight;
        }

        this.element.css({
            "height": height,
            "min-height": "",
            "max-height": ""
        });

        this.window.scrollTop( scrollTop );
    },

    refresh: function() {
        if ( this.options.autogrow && this.isTextarea ) {
            this._updateHeight();
```

```
            }
        },

        _setOptions: function( options ) {

            this._super( options );

            if ( options.autogrow !== undefined && this.isTextarea ) {
                if ( options.autogrow ) {
                    this._autogrow();
                } else {
                    this._unbindAutogrow();
                }
            }
        }

    });
})( jQuery );

(function( $, undefined ) {

$.widget( "mobile.selectmenu", $.extend( {
    initSelector: "select:not( :jqmData(role='slider')):not( :jqmData(role='flipswitch') )",

    options: {
        theme: null,
        icon: "carat-d",
        iconpos: "right",
        inline: false,
        corners: true,
        shadow: true,
        iconshadow: false, /* TODO: Deprecated in 1.4, remove in 1.5. */
        overlayTheme: null,
        dividerTheme: null,
        hidePlaceholderMenuItems: true,
        closeText: "Close",
        nativeMenu: true,
        // This option defaults to true on iOS devices.
        preventFocusZoom: /iPhone|iPad|iPod/.test( navigator.platform ) && navigator.userAgent.
        indexOf( "AppleWebKit" ) > -1,
        mini: false
    },

    _button: function() {
        return $( "<div/>" );
    },

    _setDisabled: function( value ) {
        this.element.attr( "disabled", value );
        this.button.attr( "aria-disabled", value );
        return this._setOption( "disabled", value );
    },

    _focusButton : function() {
```

```
        var self = this;

        setTimeout( function() {
            self.button.focus();
        }, 40);
    },

    _selectOptions: function() {
        return this.select.find( "option" );
    },

    // setup items that are generally necessary for select menu extension
    _preExtension: function() {
        var inline = this.options.inline || this.element.jqmData( "inline" ),
            mini = this.options.mini || this.element.jqmData( "mini" ),
            classes = "";
        // TODO: Post 1.1--once we have time to test thoroughly--any classes manually applied
        to the original element should be carried over to the enhanced element, with an
        `-enhanced` suffix. See https://github.com/jquery/jquery-mobile/issues/3577
        /* if ( $el[0].className.length ) {
            classes = $el[0].className;
        } */
        if ( !!~this.element[0].className.indexOf( "ui-btn-left" ) ) {
            classes = " ui-btn-left";
        }

        if (  !!~this.element[0].className.indexOf( "ui-btn-right" ) ) {
            classes = " ui-btn-right";
        }

        if ( inline ) {
            classes += " ui-btn-inline";
        }
        if ( mini ) {
            classes += " ui-mini";
        }

        this.select = this.element.removeClass( "ui-btn-left ui-btn-right" ).wrap( "<div
        class='ui-select" + classes + "'>" );
        this.selectId  = this.select.attr( "id" ) || ( "select-" + this.uuid );
        this.buttonId = this.selectId + "-button";
        this.label = $( "label[for='"+ this.selectId +"']" );
        this.isMultiple = this.select[ 0 ].multiple;
    },

    _destroy: function() {
        var wrapper = this.element.parents( ".ui-select" );
        if ( wrapper.length > 0 ) {
            if ( wrapper.is( ".ui-btn-left, .ui-btn-right" ) ) {
                this.element.addClass( wrapper.hasClass( "ui-btn-left" ) ? "ui-btn-left" :
                "ui-btn-right" );
            }
            this.element.insertAfter( wrapper );
            wrapper.remove();
```

```
        }
    },

    _create: function() {
        this._preExtension();

        this.button = this._button();

        var self = this,

            options = this.options,

            iconpos = options.icon ? ( options.iconpos || this.select.jqmData( "iconpos" ) ) :
            false,

            button = this.button
                .insertBefore( this.select )
                .attr( "id", this.buttonId )
                .addClass( "ui-btn" +
                    ( options.icon ? ( " ui-icon-" + options.icon + " ui-btn-icon-" + iconpos +
                    ( options.iconshadow ? " ui-shadow-icon" : "" ) ) : "" ) + /* TODO: Remove
                    in 1.5. */
                    ( options.theme ? " ui-btn-" + options.theme : "" ) +
                    ( options.corners ? " ui-corner-all" : "" ) +
                    ( options.shadow ? " ui-shadow" : "" ) );

        this.setButtonText();

        // Opera does not properly support opacity on select elements
        // In Mini, it hides the element, but not its text
        // On the desktop,it seems to do the opposite
        // for these reasons, using the nativeMenu option results in a full native select in
        Opera
        if ( options.nativeMenu && window.opera && window.opera.version ) {
            button.addClass( "ui-select-nativeonly" );
        }

        // Add counter for multi selects
        if ( this.isMultiple ) {
            this.buttonCount = $( "<span>" )
                .addClass( "ui-li-count ui-body-inherit" )
                .hide()
                .appendTo( button.addClass( "ui-li-has-count" ) );
        }

        // Disable if specified
        if ( options.disabled || this.element.attr( "disabled" )) {
            this.disable();
        }

        // Events on native select
        this.select.change(function() {
            self.refresh();
```

```
            if ( !!options.nativeMenu ) {
                this.blur();
            }
        });

        this._handleFormReset();

        this._on( this.button, {
            keydown: "_handleKeydown"
        });

        this.build();
    },

    build: function() {
        var self = this;

        this.select
            .appendTo( self.button )
            .bind( "vmousedown", function() {
                // Add active class to button
                self.button.addClass( $.mobile.activeBtnClass );
            })
            .bind( "focus", function() {
                self.button.addClass( $.mobile.focusClass );
            })
            .bind( "blur", function() {
                self.button.removeClass( $.mobile.focusClass );
            })
            .bind( "focus vmouseover", function() {
                self.button.trigger( "vmouseover" );
            })
            .bind( "vmousemove", function() {
                // Remove active class on scroll/touchmove
                self.button.removeClass( $.mobile.activeBtnClass );
            })
            .bind( "change blur vmouseout", function() {
                self.button.trigger( "vmouseout" )
                    .removeClass( $.mobile.activeBtnClass );
            });

        // In many situations, iOS will zoom into the select upon tap, this prevents that from
        happening
        self.button.bind( "vmousedown", function() {
            if ( self.options.preventFocusZoom ) {
                    $.mobile.zoom.disable( true );
            }
        });
        self.label.bind( "click focus", function() {
            if ( self.options.preventFocusZoom ) {
                    $.mobile.zoom.disable( true );
            }
        });
        self.select.bind( "focus", function() {
```

```
            if ( self.options.preventFocusZoom ) {
                    $.mobile.zoom.disable( true );
            }
        });
        self.button.bind( "mouseup", function() {
            if ( self.options.preventFocusZoom ) {
                setTimeout(function() {
                    $.mobile.zoom.enable( true );
                }, 0 );
            }
        });
        self.select.bind( "blur", function() {
            if ( self.options.preventFocusZoom ) {
                $.mobile.zoom.enable( true );
            }
        });

    },

    selected: function() {
        return this._selectOptions().filter( ":selected" );
    },

    selectedIndices: function() {
        var self = this;

        return this.selected().map(function() {
            return self._selectOptions().index( this );
        }).get();
    },

    setButtonText: function() {
        var self = this,
            selected = this.selected(),
            text = this.placeholder,
            span = $( document.createElement( "span" ) );

        this.button.children( "span" ).not( ".ui-li-count" ).remove().end().end().prepend( (
        function() {
            if ( selected.length ) {
                text = selected.map(function() {
                    return $( this ).text();
                }).get().join( ", " );
            } else {
                text = self.placeholder;
            }

            if ( text ) {
                span.text( text );
            } else {

                // Set the contents to   which we write as   to be XHTML compliant -
                see gh-6699
                span.html( " " );
```

```
            }

            // TODO possibly aggregate multiple select option classes
            return span
                .addClass( self.select.attr( "class" ) )
                .addClass( selected.attr( "class" ) )
                .removeClass( "ui-screen-hidden" );
        })());
    },

    setButtonCount: function() {
        var selected = this.selected();

        // multiple count inside button
        if ( this.isMultiple ) {
            this.buttonCount[ selected.length > 1 ? "show" : "hide" ]().text( selected.length );
        }
    },

    _handleKeydown: function( /* event */ ) {
        this._delay( "_refreshButton" );
    },

    _reset: function() {
        this.refresh();
    },

    _refreshButton: function() {
        this.setButtonText();
        this.setButtonCount();
    },

    refresh: function() {
        this._refreshButton();
    },

    // open and close preserved in native selects
    // to simplify users code when looping over selects
    open: $.noop,
    close: $.noop,

    disable: function() {
        this._setDisabled( true );
        this.button.addClass( "ui-state-disabled" );
    },

    enable: function() {
        this._setDisabled( false );
        this.button.removeClass( "ui-state-disabled" );
    }
}, $.mobile.behaviors.formReset ) );

})( jQuery );
```

```
(function( $, undefined ) {

$.mobile.links = function( target ) {

    //links within content areas, tests included with page
    $( target )
        .find( "a" )
        .jqmEnhanceable()
        .filter( ":jqmData(rel='popup')[href][href!='']" )
        .each( function() {
            // Accessibility info for popups
            var element = this,
                idref = element.getAttribute( "href" ).substring( 1 );

            if ( idref ) {
                element.setAttribute( "aria-haspopup", true );
                element.setAttribute( "aria-owns", idref );
                element.setAttribute( "aria-expanded", false );
            }
        })
        .end()
        .not( ".ui-btn, :jqmData(role='none'), :jqmData(role='nojs')" )
        .addClass( "ui-link" );

};

})( jQuery );


(function( $, undefined ) {

function fitSegmentInsideSegment( windowSize, segmentSize, offset, desired ) {
    var returnValue = desired;

    if ( windowSize < segmentSize ) {
        // Center segment if it's bigger than the window
        returnValue = offset + ( windowSize - segmentSize ) / 2;
    } else {
        // Otherwise center it at the desired coordinate while keeping it completely inside the
        window
        returnValue = Math.min( Math.max( offset, desired - segmentSize / 2 ), offset +
        windowSize - segmentSize );
    }

    return returnValue;
}

function getWindowCoordinates( theWindow ) {
    return {
        x: theWindow.scrollLeft(),
        y: theWindow.scrollTop(),
        cx: ( theWindow[ 0 ].innerWidth || theWindow.width() ),
        cy: ( theWindow[ 0 ].innerHeight || theWindow.height() )
    };
```

```
}

$.widget( "mobile.popup", {
    options: {
        wrapperClass: null,
        theme: null,
        overlayTheme: null,
        shadow: true,
        corners: true,
        transition: "none",
        positionTo: "origin",
        tolerance: null,
        closeLinkSelector: "a:jqmData(rel='back')",
        closeLinkEvents: "click.popup",
        navigateEvents: "navigate.popup",
        closeEvents: "navigate.popup pagebeforechange.popup",
        dismissible: true,
        enhanced: false,

        // NOTE Windows Phone 7 has a scroll position caching issue that
        //      requires us to disable popup history management by default
        //      https://github.com/jquery/jquery-mobile/issues/4784
        //
        // NOTE this option is modified in _create!
        history: !$.mobile.browser.oldIE
    },

    _create: function() {
        var theElement = this.element,
            myId = theElement.attr( "id" ),
            currentOptions = this.options;

        // We need to adjust the history option to be false if there's no AJAX nav.
        // We can't do it in the option declarations because those are run before
        // it is determined whether there shall be AJAX nav.
        currentOptions.history = currentOptions.history && $.mobile.ajaxEnabled && $.mobile.
        hashListeningEnabled;

        // Define instance variables
        $.extend( this, {
            _scrollTop: 0,
            _page: theElement.closest( ".ui-page" ),
            _ui: null,
            _fallbackTransition: "",
            _currentTransition: false,
            _prerequisites: null,
            _isOpen: false,
            _tolerance: null,
            _resizeData: null,
            _ignoreResizeTo: 0,
            _orientationchangeInProgress: false
        });

        if ( this._page.length === 0 ) {
```

```
            this._page = $( "body" );
        }

        if ( currentOptions.enhanced ) {
            this._ui = {
                container: theElement.parent(),
                screen: theElement.parent().prev(),
                placeholder: $( this.document[ 0 ].getElementById( myId + "-placeholder" ) )
            };
        } else {
            this._ui = this._enhance( theElement, myId );
            this._applyTransition( currentOptions.transition );
        }
        this
            ._setTolerance( currentOptions.tolerance )
            ._ui.focusElement = this._ui.container;

        // Event handlers
        this._on( this._ui.screen, { "vclick": "_eatEventAndClose" } );
        this._on( this.window, {
            orientationchange: $.proxy( this, "_handleWindowOrientationchange" ),
            resize: $.proxy( this, "_handleWindowResize" ),
            keyup: $.proxy( this, "_handleWindowKeyUp" )
        });
        this._on( this.document, { "focusin": "_handleDocumentFocusIn" } );
    },

    _enhance: function( theElement, myId ) {
        var currentOptions = this.options,
            wrapperClass = currentOptions.wrapperClass,
            ui = {
                screen: $( "<div class='ui-screen-hidden ui-popup-screen " +
                this._themeClassFromOption( "ui-overlay-", currentOptions.overlayTheme ) +
                "'></div>" ),
                placeholder: $( "<div style='display: none;'><!-- placeholder --></div>" ),
                container: $( "<div class='ui-popup-container ui-popup-hidden ui-popup-truncate"
                 +
                    ( wrapperClass ? ( " " + wrapperClass ) : "" ) + "'></div>" )
            },
            fragment = this.document[ 0 ].createDocumentFragment();

        fragment.appendChild( ui.screen[ 0 ] );
        fragment.appendChild( ui.container[ 0 ] );

        if ( myId ) {
            ui.screen.attr( "id", myId + "-screen" );
            ui.container.attr( "id", myId + "-popup" );
            ui.placeholder
                .attr( "id", myId + "-placeholder" )
                .html( "<!-- placeholder for " + myId + " -->" );
        }

        // Apply the proto
        this._page[ 0 ].appendChild( fragment );
```

```
        // Leave a placeholder where the element used to be
        ui.placeholder.insertAfter( theElement );
        theElement
            .detach()
            .addClass( "ui-popup " +
                this._themeClassFromOption( "ui-body-", currentOptions.theme ) + " " +
                ( currentOptions.shadow ? "ui-overlay-shadow " : "" ) +
                ( currentOptions.corners ? "ui-corner-all " : "" ) )
            .appendTo( ui.container );

        return ui;
    },

    _eatEventAndClose: function( theEvent ) {
        theEvent.preventDefault();
        theEvent.stopImmediatePropagation();
        if ( this.options.dismissible ) {
            this.close();
        }
        return false;
    },

    // Make sure the screen covers the entire document - CSS is sometimes not
    // enough to accomplish this.
    _resizeScreen: function() {
        var screen = this._ui.screen,
            popupHeight = this._ui.container.outerHeight( true ),
            screenHeight = screen.removeAttr( "style" ).height(),

            // Subtracting 1 here is necessary for an obscure Andrdoid 4.0 bug where
            // the browser hangs if the screen covers the entire document :/
            documentHeight = this.document.height() - 1;

        if ( screenHeight < documentHeight ) {
            screen.height( documentHeight );
        } else if ( popupHeight > screenHeight ) {
            screen.height( popupHeight );
        }
    },

    _handleWindowKeyUp: function( theEvent ) {
        if ( this._isOpen && theEvent.keyCode === $.mobile.keyCode.ESCAPE ) {
            return this._eatEventAndClose( theEvent );
        }
    },

    _expectResizeEvent: function() {
        var windowCoordinates = getWindowCoordinates( this.window );

        if ( this._resizeData ) {
            if ( windowCoordinates.x === this._resizeData.windowCoordinates.x &&
                windowCoordinates.y === this._resizeData.windowCoordinates.y &&
                windowCoordinates.cx === this._resizeData.windowCoordinates.cx &&
                windowCoordinates.cy === this._resizeData.windowCoordinates.cy ) {
```

```
                // timeout not refreshed
                return false;
            } else {
                // clear existing timeout - it will be refreshed below
                clearTimeout( this._resizeData.timeoutId );
            }
        }

        this._resizeData = {
            timeoutId: this._delay( "_resizeTimeout", 200 ),
            windowCoordinates: windowCoordinates
        };

        return true;
    },

    _resizeTimeout: function() {
        if ( this._isOpen ) {
            if ( !this._expectResizeEvent() ) {
                if ( this._ui.container.hasClass( "ui-popup-hidden" ) ) {
                    // effectively rapid-open the popup while leaving the screen intact
                    this._ui.container.removeClass( "ui-popup-hidden ui-popup-truncate" );
                    this.reposition( { positionTo: "window" } );
                    this._ignoreResizeEvents();
                }

                this._resizeScreen();
                this._resizeData = null;
                this._orientationchangeInProgress = false;
            }
        } else {
            this._resizeData = null;
            this._orientationchangeInProgress = false;
        }
    },

    _stopIgnoringResizeEvents: function() {
        this._ignoreResizeTo = 0;
    },

    _ignoreResizeEvents: function() {
        if ( this._ignoreResizeTo ) {
            clearTimeout( this._ignoreResizeTo );
        }
        this._ignoreResizeTo = this._delay( "_stopIgnoringResizeEvents", 1000 );
    },

    _handleWindowResize: function(/* theEvent */) {
        if ( this._isOpen && this._ignoreResizeTo === 0 ) {
            if ( ( this._expectResizeEvent() || this._orientationchangeInProgress ) &&
                !this._ui.container.hasClass( "ui-popup-hidden" ) ) {
                // effectively rapid-close the popup while leaving the screen intact
                this._ui.container
                    .addClass( "ui-popup-hidden ui-popup-truncate" )
```

```
                .removeAttr( "style" );
            }
        }
    },

    _handleWindowOrientationchange: function(/* theEvent */) {
        if ( !this._orientationchangeInProgress && this._isOpen && this._ignoreResizeTo === 0 ) {
            this._expectResizeEvent();
            this._orientationchangeInProgress = true;
        }
    },

    // When the popup is open, attempting to focus on an element that is not a
    // child of the popup will redirect focus to the popup
    _handleDocumentFocusIn: function( theEvent ) {
        var target,
            targetElement = theEvent.target,
            ui = this._ui;

        if ( !this._isOpen ) {
            return;
        }

        if ( targetElement !== ui.container[ 0 ] ) {
            target = $( targetElement );
            if ( 0 === target.parents().filter( ui.container[ 0 ] ).length ) {
                $( this.document[ 0 ].activeElement ).one( "focus", function(/* theEvent */) {
                    target.blur();
                });
                ui.focusElement.focus();
                theEvent.preventDefault();
                theEvent.stopImmediatePropagation();
                return false;
            } else if ( ui.focusElement[ 0 ] === ui.container[ 0 ] ) {
                ui.focusElement = target;
            }
        }

        this._ignoreResizeEvents();
    },

    _themeClassFromOption: function( prefix, value ) {
        return ( value ? ( value === "none" ? "" : ( prefix + value ) ) : ( prefix + "inherit" )
         );
    },

    _applyTransition: function( value ) {
        if ( value ) {
            this._ui.container.removeClass( this._fallbackTransition );
            if ( value !== "none" ) {
                this._fallbackTransition = $.mobile._maybeDegradeTransition( value );
                if ( this._fallbackTransition === "none" ) {
                    this._fallbackTransition = "";
                }
```

```
                this._ui.container.addClass( this._fallbackTransition );
            }
        }

        return this;
    },

    _setOptions: function( newOptions ) {
        var currentOptions = this.options,
            theElement = this.element,
            screen = this._ui.screen;

        if ( newOptions.wrapperClass !== undefined ) {
            this._ui.container
                .removeClass( currentOptions.wrapperClass )
                .addClass( newOptions.wrapperClass );
        }

        if ( newOptions.theme !== undefined ) {
            theElement
                .removeClass( this._themeClassFromOption( "ui-body-", currentOptions.theme ) )
                .addClass( this._themeClassFromOption( "ui-body-", newOptions.theme ) );
        }

        if ( newOptions.overlayTheme !== undefined ) {
            screen
                .removeClass( this._themeClassFromOption( "ui-overlay-", currentOptions.
                overlayTheme ) )
                .addClass( this._themeClassFromOption( "ui-overlay-", newOptions.overlayTheme )
                );

            if ( this._isOpen ) {
                screen.addClass( "in" );
            }
        }

        if ( newOptions.shadow !== undefined ) {
            theElement.toggleClass( "ui-overlay-shadow", newOptions.shadow );
        }

        if ( newOptions.corners !== undefined ) {
            theElement.toggleClass( "ui-corner-all", newOptions.corners );
        }

        if ( newOptions.transition !== undefined ) {
            if ( !this._currentTransition ) {
                this._applyTransition( newOptions.transition );
            }
        }

        if ( newOptions.tolerance !== undefined ) {
            this._setTolerance( newOptions.tolerance );
        }
```

```
        if ( newOptions.disabled !== undefined ) {
            if ( newOptions.disabled ) {
                this.close();
            }
        }

        return this._super( newOptions );
    },

    _setTolerance: function( value ) {
        var tol = { t: 30, r: 15, b: 30, l: 15 },
            ar;

        if ( value !== undefined ) {
            ar = String( value ).split( "," );

            $.each( ar, function( idx, val ) { ar[ idx ] = parseInt( val, 10 ); } );

            switch( ar.length ) {
                // All values are to be the same
                case 1:
                    if ( !isNaN( ar[ 0 ] ) ) {
                        tol.t = tol.r = tol.b = tol.l = ar[ 0 ];
                    }
                    break;

                // The first value denotes top/bottom tolerance, and the second value denotes
                left/right tolerance
                case 2:
                    if ( !isNaN( ar[ 0 ] ) ) {
                        tol.t = tol.b = ar[ 0 ];
                    }
                    if ( !isNaN( ar[ 1 ] ) ) {
                        tol.l = tol.r = ar[ 1 ];
                    }
                    break;

                // The array contains values in the order top, right, bottom, left
                case 4:
                    if ( !isNaN( ar[ 0 ] ) ) {
                        tol.t = ar[ 0 ];
                    }
                    if ( !isNaN( ar[ 1 ] ) ) {
                        tol.r = ar[ 1 ];
                    }
                    if ( !isNaN( ar[ 2 ] ) ) {
                        tol.b = ar[ 2 ];
                    }
                    if ( !isNaN( ar[ 3 ] ) ) {
                        tol.l = ar[ 3 ];
                    }
                    break;

                default:
```

```
                break;
            }
        }

        this._tolerance = tol;
        return this;
    },

    _clampPopupWidth: function( infoOnly ) {
        var menuSize,
            windowCoordinates = getWindowCoordinates( this.window ),
            // rectangle within which the popup must fit
            rectangle = {
                x: this._tolerance.l,
                y: windowCoordinates.y + this._tolerance.t,
                cx: windowCoordinates.cx - this._tolerance.l - this._tolerance.r,
                cy: windowCoordinates.cy - this._tolerance.t - this._tolerance.b
            };

        if ( !infoOnly ) {
            // Clamp the width of the menu before grabbing its size
            this._ui.container.css( "max-width", rectangle.cx );
        }

        menuSize = {
            cx: this._ui.container.outerWidth( true ),
            cy: this._ui.container.outerHeight( true )
        };

        return { rc: rectangle, menuSize: menuSize };
    },

    _calculateFinalLocation: function( desired, clampInfo ) {
        var returnValue,
            rectangle = clampInfo.rc,
            menuSize = clampInfo.menuSize;

        // Center the menu over the desired coordinates, while not going outside
        // the window tolerances. This will center wrt. the window if the popup is
        // too large.
        returnValue = {
            left: fitSegmentInsideSegment( rectangle.cx, menuSize.cx, rectangle.x, desired.x ),
            top: fitSegmentInsideSegment( rectangle.cy, menuSize.cy, rectangle.y, desired.y )
        };

        // Make sure the top of the menu is visible
        returnValue.top = Math.max( 0, returnValue.top );

        // If the height of the menu is smaller than the height of the document
        // align the bottom with the bottom of the document

        returnValue.top -= Math.min( returnValue.top,
            Math.max( 0, returnValue.top + menuSize.cy - this.document.height() ) );
```

```
        return returnValue;
    },

    // Try and center the overlay over the given coordinates
    _placementCoords: function( desired ) {
        return this._calculateFinalLocation( desired, this._clampPopupWidth() );
    },

    _createPrerequisites: function( screenPrerequisite, containerPrerequisite, whenDone ) {
        var prerequisites,
            self = this;

        // It is important to maintain both the local variable prerequisites and
        // self._prerequisites. The local variable remains in the closure of the
        // functions which call the callbacks passed in. The comparison between the
        // local variable and self._prerequisites is necessary, because once a
        // function has been passed to .animationComplete() it will be called next
        // time an animation completes, even if that's not the animation whose end
        // the function was supposed to catch (for example, if an abort happens
        // during the opening animation, the .animationComplete handler is not
        // called for that animation anymore, but the handler remains attached, so
        // it is called the next time the popup is opened - making it stale.
        // Comparing the local variable prerequisites to the widget-level variable
        // self._prerequisites ensures that callbacks triggered by a stale
        // .animationComplete will be ignored.

        prerequisites = {
            screen: $.Deferred(),
            container: $.Deferred()
        };

        prerequisites.screen.then( function() {
            if ( prerequisites === self._prerequisites ) {
                screenPrerequisite();
            }
        });

        prerequisites.container.then( function() {
            if ( prerequisites === self._prerequisites ) {
                containerPrerequisite();
            }
        });

        $.when( prerequisites.screen, prerequisites.container ).done( function() {
            if ( prerequisites === self._prerequisites ) {
                self._prerequisites = null;
                whenDone();
            }
        });

        self._prerequisites = prerequisites;
    },

    _animate: function( args ) {
```

```
        // NOTE before removing the default animation of the screen
        //      this had an animate callback that would resolve the deferred
        //      now the deferred is resolved immediately
        // TODO remove the dependency on the screen deferred
        this._ui.screen
            .removeClass( args.classToRemove )
            .addClass( args.screenClassToAdd );

        args.prerequisites.screen.resolve();

        if ( args.transition && args.transition !== "none" ) {
            if ( args.applyTransition ) {
                this._applyTransition( args.transition );
            }
            if ( this._fallbackTransition ) {
                this._ui.container
                    .addClass( args.containerClassToAdd )
                    .removeClass( args.classToRemove )
                    .animationComplete( $.proxy( args.prerequisites.container, "resolve" ) );
                return;
            }
        }
        this._ui.container.removeClass( args.classToRemove );
        args.prerequisites.container.resolve();
    },

    // The desired coordinates passed in will be returned untouched if no reference element can
    be identified via
    // desiredPosition.positionTo. Nevertheless, this function ensures that its return value
    always contains valid
    // x and y coordinates by specifying the center middle of the window if the coordinates are
    absent.
    // options: { x: coordinate, y: coordinate, positionTo: string: "origin", "window", or
    jQuery selector
    _desiredCoords: function( openOptions ) {
        var offset,
            dst = null,
            windowCoordinates = getWindowCoordinates( this.window ),
            x = openOptions.x,
            y = openOptions.y,
            pTo = openOptions.positionTo;

        // Establish which element will serve as the reference
        if ( pTo && pTo !== "origin" ) {
            if ( pTo === "window" ) {
                x = windowCoordinates.cx / 2 + windowCoordinates.x;
                y = windowCoordinates.cy / 2 + windowCoordinates.y;
            } else {
                try {
                    dst = $( pTo );
                } catch( err ) {
                    dst = null;
                }
                if ( dst ) {
```

```
                dst.filter( ":visible" );
                if ( dst.length === 0 ) {
                    dst = null;
                }
            }
        }
    }

    // If an element was found, center over it
    if ( dst ) {
        offset = dst.offset();
        x = offset.left + dst.outerWidth() / 2;
        y = offset.top + dst.outerHeight() / 2;
    }

    // Make sure x and y are valid numbers - center over the window
    if ( $.type( x ) !== "number" || isNaN( x ) ) {
        x = windowCoordinates.cx / 2 + windowCoordinates.x;
    }
    if ( $.type( y ) !== "number" || isNaN( y ) ) {
        y = windowCoordinates.cy / 2 + windowCoordinates.y;
    }

    return { x: x, y: y };
},

_reposition: function( openOptions ) {
    // We only care about position-related parameters for repositioning
    openOptions = {
        x: openOptions.x,
        y: openOptions.y,
        positionTo: openOptions.positionTo
    };
    this._trigger( "beforeposition", undefined, openOptions );
    this._ui.container.offset( this._placementCoords( this._desiredCoords( openOptions ) ) );
},

reposition: function( openOptions ) {
    if ( this._isOpen ) {
        this._reposition( openOptions );
    }
},

_openPrerequisitesComplete: function() {
    var id = this.element.attr( "id" );

    this._ui.container.addClass( "ui-popup-active" );
    this._isOpen = true;
    this._resizeScreen();
    this._ui.container.attr( "tabindex", "0" ).focus();
    this._ignoreResizeEvents();
    if ( id ) {
        this.document.find( "[aria-haspopup='true'][aria-owns='" +  id + "']" ).attr(
        "aria-expanded", true );
```

```
        }
        this._trigger( "afteropen" );
    },

    _open: function( options ) {
        var openOptions = $.extend( {}, this.options, options ),
            // TODO move blacklist to private method
            androidBlacklist = ( function() {
                var ua = navigator.userAgent,
                    // Rendering engine is Webkit, and capture major version
                    wkmatch = ua.match( /AppleWebKit\/([0-9\.]+)/ ),
                    wkversion = !!wkmatch && wkmatch[ 1 ],
                    androidmatch = ua.match( /Android (\d+(?:\.\d+))/ ),
                    andversion = !!androidmatch && androidmatch[ 1 ],
                    chromematch = ua.indexOf( "Chrome" ) > -1;

                // Platform is Android, WebKit version is greater than 534.13 ( Android 3.2.1 )
                and not Chrome.
                if ( androidmatch !== null && andversion === "4.0" && wkversion && wkversion >
                534.13 && !chromematch ) {
                    return true;
                }
                return false;
            }());

        // Count down to triggering "popupafteropen" - we have two prerequisites:
        // 1. The popup window animation completes (container())
        // 2. The screen opacity animation completes (screen())
        this._createPrerequisites(
            $.noop,
            $.noop,
            $.proxy( this, "_openPrerequisitesComplete" ) );

        this._currentTransition = openOptions.transition;
        this._applyTransition( openOptions.transition );

        this._ui.screen.removeClass( "ui-screen-hidden" );
        this._ui.container.removeClass( "ui-popup-truncate" );

        // Give applications a chance to modify the contents of the container before it appears
        this._reposition( openOptions );

        this._ui.container.removeClass( "ui-popup-hidden" );

        if ( this.options.overlayTheme && androidBlacklist ) {
            /* TODO: The native browser on Android 4.0.X ("Ice Cream Sandwich") suffers from an
            issue where the popup overlay appears to be z-indexed above the popup itself when
            certain other styles exist on the same page -- namely, any element set to
            `position: fixed` and certain types of input. These issues are reminiscent of
            previously uncovered bugs in older versions of Android's native browser:
            https://github.com/scottjehl/Device-Bugs/issues/3
            This fix closes the following bugs ( I use "closes" with reluctance, and stress
            that this issue should be revisited as soon as possible ):
            https://github.com/jquery/jquery-mobile/issues/4816
```

```
            https://github.com/jquery/jquery-mobile/issues/4844
            https://github.com/jquery/jquery-mobile/issues/4874
            */

            // TODO sort out why this._page isn't working
            this.element.closest( ".ui-page" ).addClass( "ui-popup-open" );
        }
        this._animate({
            additionalCondition: true,
            transition: openOptions.transition,
            classToRemove: "",
            screenClassToAdd: "in",
            containerClassToAdd: "in",
            applyTransition: false,
            prerequisites: this._prerequisites
        });
    },

    _closePrerequisiteScreen: function() {
        this._ui.screen
            .removeClass( "out" )
            .addClass( "ui-screen-hidden" );
    },

    _closePrerequisiteContainer: function() {
        this._ui.container
            .removeClass( "reverse out" )
            .addClass( "ui-popup-hidden ui-popup-truncate" )
            .removeAttr( "style" );
    },

    _closePrerequisitesDone: function() {
        var container = this._ui.container,
            id = this.element.attr( "id" );

        container.removeAttr( "tabindex" );

        // remove the global mutex for popups
        $.mobile.popup.active = undefined;

        // Blur elements inside the container, including the container
        $( ":focus", container[ 0 ] ).add( container[ 0 ] ).blur();

        if ( id ) {
            this.document.find( "[aria-haspopup='true'][aria-owns='" +  id + "']" ).attr(
            "aria-expanded", false );
        }

        // alert users that the popup is closed
        this._trigger( "afterclose" );
    },

    _close: function( immediate ) {
        this._ui.container.removeClass( "ui-popup-active" );
```

```
        this._page.removeClass( "ui-popup-open" );

        this._isOpen = false;

        // Count down to triggering "popupafterclose" - we have two prerequisites:
        // 1. The popup window reverse animation completes (container())
        // 2. The screen opacity animation completes (screen())
        this._createPrerequisites(
            $.proxy( this, "_closePrerequisiteScreen" ),
            $.proxy( this, "_closePrerequisiteContainer" ),
            $.proxy( this, "_closePrerequisitesDone" ) );

        this._animate( {
            additionalCondition: this._ui.screen.hasClass( "in" ),
            transition: ( immediate ? "none" : ( this._currentTransition ) ),
            classToRemove: "in",
            screenClassToAdd: "out",
            containerClassToAdd: "reverse out",
            applyTransition: true,
            prerequisites: this._prerequisites
        });
    },

    _unenhance: function() {
        if ( this.options.enhanced ) {
            return;
        }

        // Put the element back to where the placeholder was and remove the "ui-popup" class
        this._setOptions( { theme: $.mobile.popup.prototype.options.theme } );
        this.element
            // Cannot directly insertAfter() - we need to detach() first, because
            // insertAfter() will do nothing if the payload div was not attached
            // to the DOM at the time the widget was created, and so the payload
            // will remain inside the container even after we call insertAfter().
            // If that happens and we remove the container a few lines below, we
            // will cause an infinite recursion - #5244
            .detach()
            .insertAfter( this._ui.placeholder )
            .removeClass( "ui-popup ui-overlay-shadow ui-corner-all ui-body-inherit" );
        this._ui.screen.remove();
        this._ui.container.remove();
        this._ui.placeholder.remove();
    },

    _destroy: function() {
        if ( $.mobile.popup.active === this ) {
            this.element.one( "popupafterclose", $.proxy( this, "_unenhance" ) );
            this.close();
        } else {
            this._unenhance();
        }

        return this;
```

```
    },

    _closePopup: function( theEvent, data ) {
        var parsedDst, toUrl,
            currentOptions = this.options,
            immediate = false;

        if ( ( theEvent && theEvent.isDefaultPrevented() ) || $.mobile.popup.active !== this ) {
            return;
        }

        // restore location on screen
        window.scrollTo( 0, this._scrollTop );

        if ( theEvent && theEvent.type === "pagebeforechange" && data ) {
            // Determine whether we need to rapid-close the popup, or whether we can
            // take the time to run the closing transition
            if ( typeof data.toPage === "string" ) {
                parsedDst = data.toPage;
            } else {
                parsedDst = data.toPage.jqmData( "url" );
            }
            parsedDst = $.mobile.path.parseUrl( parsedDst );
            toUrl = parsedDst.pathname + parsedDst.search + parsedDst.hash;

            if ( this._myUrl !== $.mobile.path.makeUrlAbsolute( toUrl ) ) {
                // Going to a different page - close immediately
                immediate = true;
            } else {
                theEvent.preventDefault();
            }
        }

        // remove nav bindings
        this.window.off( currentOptions.closeEvents );
        // unbind click handlers added when history is disabled
        this.element.undelegate( currentOptions.closeLinkSelector, currentOptions.
        closeLinkEvents );

        this._close( immediate );
    },

    // any navigation event after a popup is opened should close the popup
    // NOTE the pagebeforechange is bound to catch navigation events that don't
    //      alter the url (eg, dialogs from popups)
    _bindContainerClose: function() {
        this.window
            .on( this.options.closeEvents, $.proxy( this, "_closePopup" ) );
    },

    widget: function() {
        return this._ui.container;
    },
```

```
    // TODO no clear deliniation of what should be here and
    // what should be in _open. Seems to be "visual" vs "history" for now
    open: function( options ) {
        var url, hashkey, activePage, currentIsDialog, hasHash, urlHistory,
            self = this,
            currentOptions = this.options;

        // make sure open is idempotent
        if ( $.mobile.popup.active || currentOptions.disabled ) {
            return this;
        }

        // set the global popup mutex
        $.mobile.popup.active = this;
        this._scrollTop = this.window.scrollTop();

        // if history alteration is disabled close on navigate events
        // and leave the url as is
        if ( !( currentOptions.history ) ) {
            self._open( options );
            self._bindContainerClose();

            // When histoy is disabled we have to grab the data-rel
            // back link clicks so we can close the popup instead of
            // relying on history to do it for us
            self.element
                .delegate( currentOptions.closeLinkSelector, currentOptions.closeLinkEvents,
                function( theEvent ) {
                    self.close();
                    theEvent.preventDefault();
                });

            return this;
        }

        // cache some values for min/readability
        urlHistory = $.mobile.navigate.history;
        hashkey = $.mobile.dialogHashKey;
        activePage = $.mobile.activePage;
        currentIsDialog = ( activePage ? activePage.hasClass( "ui-dialog" ) : false );
        this._myUrl = url = urlHistory.getActive().url;
        hasHash = ( url.indexOf( hashkey ) > -1 ) && !currentIsDialog && ( urlHistory.
        activeIndex > 0 );

        if ( hasHash ) {
            self._open( options );
            self._bindContainerClose();
            return this;
        }

        // if the current url has no dialog hash key proceed as normal
        // otherwise, if the page is a dialog simply tack on the hash key
        if ( url.indexOf( hashkey ) === -1 && !currentIsDialog ) {
            url = url + (url.indexOf( "#" ) > -1 ? hashkey : "#" + hashkey);
```

```
        } else {
            url = $.mobile.path.parseLocation().hash + hashkey;
        }

        // Tack on an extra hashkey if this is the first page and we've just reconstructed the
        initial hash
        if ( urlHistory.activeIndex === 0 && url === urlHistory.initialDst ) {
            url += hashkey;
        }

        // swallow the the initial navigation event, and bind for the next
        this.window.one( "beforenavigate", function( theEvent ) {
            theEvent.preventDefault();
            self._open( options );
            self._bindContainerClose();
        });

        this.urlAltered = true;
        $.mobile.navigate( url, { role: "dialog" } );

        return this;
    },

    close: function() {
        // make sure close is idempotent
        if ( $.mobile.popup.active !== this ) {
            return this;
        }

        this._scrollTop = this.window.scrollTop();

        if ( this.options.history && this.urlAltered ) {
            $.mobile.back();
            this.urlAltered = false;
        } else {
            // simulate the nav bindings having fired
            this._closePopup();
        }

        return this;
    }
});

// TODO this can be moved inside the widget
$.mobile.popup.handleLink = function( $link ) {
    var offset,
        path = $.mobile.path,

        // NOTE make sure to get only the hash from the href because ie7 (wp7)
        //      returns the absolute href in this case ruining the element selection
        popup = $( path.hashToSelector( path.parseUrl( $link.attr( "href" ) ).hash ) ).first();

    if ( popup.length > 0 && popup.data( "mobile-popup" ) ) {
        offset = $link.offset();
```

```
        popup.popup( "open", {
            x: offset.left + $link.outerWidth() / 2,
            y: offset.top + $link.outerHeight() / 2,
            transition: $link.jqmData( "transition" ),
            positionTo: $link.jqmData( "position-to" )
        });
    }

    //remove after delay
    setTimeout( function() {
        $link.removeClass( $.mobile.activeBtnClass );
    }, 300 );
};

// TODO move inside _create
$.mobile.document.on( "pagebeforechange", function( theEvent, data ) {
    if ( data.options.role === "popup" ) {
        $.mobile.popup.handleLink( data.options.link );
        theEvent.preventDefault();
    }
});

})( jQuery );

/*
* custom "selectmenu" plugin
*/

(function( $, undefined ) {

var unfocusableItemSelector =
".ui-disabled,.ui-state-disabled,.ui-li-divider,.ui-screen-hidden,:jqmData(role='placeholder')",
    goToAdjacentItem = function( item, target, direction ) {
        var adjacent = item[ direction + "All" ]()
            .not( unfocusableItemSelector )
            .first();

        // if there's a previous option, focus it
        if ( adjacent.length ) {
            target
                .blur()
                .attr( "tabindex", "-1" );

            adjacent.find( "a" ).first().focus();
        }
    };

$.widget( "mobile.selectmenu", $.mobile.selectmenu, {
    _create: function() {
        var o = this.options;

        // Custom selects cannot exist inside popups, so revert the "nativeMenu"
        // option to true if a parent is a popup
        o.nativeMenu = o.nativeMenu || ( this.element.parents(
```

```
        ":jqmData(role='popup'),:mobile-popup" ).length > 0 );

        return this._super();
    },

    _handleSelectFocus: function() {
        this.element.blur();
        this.button.focus();
    },

    _handleKeydown: function( event ) {
        this._super( event );
        this._handleButtonVclickKeydown( event );
    },

    _handleButtonVclickKeydown: function( event ) {
        if ( this.options.disabled || this.isOpen ) {
            return;
        }

        if (event.type === "vclick" ||
                event.keyCode && (event.keyCode === $.mobile.keyCode.ENTER || event.keyCode ===
                $.mobile.keyCode.SPACE)) {

            this._decideFormat();
            if ( this.menuType === "overlay" ) {
                this.button.attr( "href", "#" + this.popupId ).attr( "data-" + ( $.mobile.ns ||
                "" ) + "rel", "popup" );
            } else {
                this.button.attr( "href", "#" + this.dialogId ).attr( "data-" + ( $.mobile.ns ||
                 "" ) + "rel", "dialog" );
            }
            this.isOpen = true;
            // Do not prevent default, so the navigation may have a chance to actually open the
            chosen format
        }
    },

    _handleListFocus: function( e ) {
        var params = ( e.type === "focusin" ) ?
            { tabindex: "0", event: "vmouseover" }:
            { tabindex: "-1", event: "vmouseout" };

        $( e.target )
            .attr( "tabindex", params.tabindex )
            .trigger( params.event );
    },

    _handleListKeydown: function( event ) {
        var target = $( event.target ),
            li = target.closest( "li" );

        // switch logic based on which key was pressed
        switch ( event.keyCode ) {
```

```
            // up or left arrow keys
        case 38:
            goToAdjacentItem( li, target, "prev" );
            return false;
            // down or right arrow keys
        case 40:
            goToAdjacentItem( li, target, "next" );
            return false;
            // If enter or space is pressed, trigger click
        case 13:
        case 32:
            target.trigger( "click" );
            return false;
        }
    },

    _handleMenuPageHide: function() {
        // TODO centralize page removal binding / handling in the page plugin.
        // Suggestion from @jblas to do refcounting
        //
        // TODO extremely confusing dependency on the open method where the pagehide.remove
        // bindings are stripped to prevent the parent page from disappearing. The way
        // we're keeping pages in the DOM right now sucks
        //
        // rebind the page remove that was unbound in the open function
        // to allow for the parent page removal from actions other than the use
        // of a dialog sized custom select
        //
        // doing this here provides for the back button on the custom select dialog
        this.thisPage.page( "bindRemove" );
    },

    _handleHeaderCloseClick: function() {
        if ( this.menuType === "overlay" ) {
            this.close();
            return false;
        }
    },

    build: function() {
        var selectId, popupId, dialogId, label, thisPage, isMultiple, menuId,
            themeAttr, overlayTheme, overlayThemeAttr, dividerThemeAttr,
            menuPage, listbox, list, header, headerTitle, menuPageContent,
            menuPageClose, headerClose, self,
            o = this.options;

        if ( o.nativeMenu ) {
            return this._super();
        }

        self = this;
        selectId = this.selectId;
        popupId = selectId + "-listbox";
        dialogId = selectId + "-dialog";
```

```
        label = this.label;
        thisPage = this.element.closest( ".ui-page" );
        isMultiple = this.element[ 0 ].multiple;
        menuId = selectId + "-menu";
        themeAttr = o.theme ? ( " data-" + $.mobile.ns + "theme='" + o.theme + "'" ) : "";
        overlayTheme = o.overlayTheme || o.theme || null;
        overlayThemeAttr = overlayTheme ? ( " data-" + $.mobile.ns +
            "overlay-theme='" + overlayTheme + "'" ) : "";
        dividerThemeAttr = ( o.dividerTheme && isMultiple ) ? ( " data-" + $.mobile.ns +
        "divider-theme='" + o.dividerTheme + "'" ) : "";
        menuPage = $( "<div data-" + $.mobile.ns + "role='dialog' class='ui-selectmenu' id='" +
        dialogId + "'" + themeAttr + overlayThemeAttr + ">" +
            "<div data-" + $.mobile.ns + "role='header'>" +
            "<div class='ui-title'>" + label.getEncodedText() + "</div>"+
            "</div>"+
            "<div data-" + $.mobile.ns + "role='content'></div>"+
            "</div>" );
        listbox = $( "<div id='" + popupId + "' class='ui-selectmenu'></div>" ).insertAfter(
        this.select ).popup({ theme: o.overlayTheme });
        list = $( "<ul class='ui-selectmenu-list' id='" + menuId + "' role='listbox'
        aria-labelledby='" + this.buttonId + "'" + themeAttr + dividerThemeAttr + "></ul>" ).
        appendTo( listbox );
        header = $( "<div class='ui-header ui-bar-" + ( o.theme ? o.theme : "inherit" ) +
        "'></div>" ).prependTo( listbox );
        headerTitle = $( "<h1 class='ui-title'></h1>" ).appendTo( header );

        if ( this.isMultiple ) {
            headerClose = $( "<a>", {
                "role": "button",
                "text": o.closeText,
                "href": "#",
                "class": "ui-btn ui-corner-all ui-btn-left ui-btn-icon-notext ui-icon-delete"
            }).appendTo( header );
        }

        $.extend( this, {
            selectId: selectId,
            menuId: menuId,
            popupId: popupId,
            dialogId: dialogId,
            thisPage: thisPage,
            menuPage: menuPage,
            label: label,
            isMultiple: isMultiple,
            theme: o.theme,
            listbox: listbox,
            list: list,
            header: header,
            headerTitle: headerTitle,
            headerClose: headerClose,
            menuPageContent: menuPageContent,
            menuPageClose: menuPageClose,
            placeholder: ""
        });
```

```
        // Create list from select, update state
        this.refresh();

        if ( this._origTabIndex === undefined ) {
            // Map undefined to false, because this._origTabIndex === undefined
            // indicates that we have not yet checked whether the select has
            // originally had a tabindex attribute, whereas false indicates that
            // we have checked the select for such an attribute, and have found
            // none present.
            this._origTabIndex = ( this.select[ 0 ].getAttribute( "tabindex" ) === null ) ?
            false : this.select.attr( "tabindex" );
        }
        this.select.attr( "tabindex", "-1" );
        this._on( this.select, { focus : "_handleSelectFocus" } );

        // Button events
        this._on( this.button, {
            vclick: "_handleButtonVclickKeydown"
        });

        // Events for list items
        this.list.attr( "role", "listbox" );
        this._on( this.list, {
            focusin : "_handleListFocus",
            focusout : "_handleListFocus",
            keydown: "_handleListKeydown"
        });
        this.list
            .delegate( "li:not(.ui-disabled,.ui-state-disabled,.ui-li-divider)", "click",
            function( event ) {

                // index of option tag to be selected
                var oldIndex = self.select[ 0 ].selectedIndex,
                    newIndex = $.mobile.getAttribute( this, "option-index" ),
                    option = self._selectOptions().eq( newIndex )[ 0 ];

                // toggle selected status on the tag for multi selects
                option.selected = self.isMultiple ? !option.selected : true;

                // toggle checkbox class for multiple selects
                if ( self.isMultiple ) {
                    $( this ).find( "a" )
                        .toggleClass( "ui-checkbox-on", option.selected )
                        .toggleClass( "ui-checkbox-off", !option.selected );
                }

                // trigger change if value changed
                if ( self.isMultiple || oldIndex !== newIndex ) {
                    self.select.trigger( "change" );
                }

                // hide custom select for single selects only - otherwise focus clicked item
                // We need to grab the clicked item the hard way, because the list may have
```

```
                been rebuilt
                if ( self.isMultiple ) {
                    self.list.find( "li:not(.ui-li-divider)" ).eq( newIndex )
                        .find( "a" ).first().focus();
                }
                else {
                    self.close();
                }

                event.preventDefault();
            });

        // button refocus ensures proper height calculation
        // by removing the inline style and ensuring page inclusion
        this._on( this.menuPage, { pagehide: "_handleMenuPageHide" } );

        // Events on the popup
        this._on( this.listbox, { popupafterclose: "close" } );

        // Close button on small overlays
        if ( this.isMultiple ) {
            this._on( this.headerClose, { click: "_handleHeaderCloseClick" } );
        }

        return this;
    },

    _isRebuildRequired: function() {
        var list = this.list.find( "li" ),
            options = this._selectOptions().not( ".ui-screen-hidden" );

        // TODO exceedingly naive method to determine difference
        // ignores value changes etc in favor of a forcedRebuild
        // from the user in the refresh method
        return options.text() !== list.text();
    },

    selected: function() {
        return this._selectOptions().filter( ":selected:not( :jqmData(placeholder='true') )" );
    },

    refresh: function( force ) {
        var self, indices;

        if ( this.options.nativeMenu ) {
            return this._super( force );
        }

        self = this;
        if ( force || this._isRebuildRequired() ) {
            self._buildList();
        }

        indices = this.selectedIndices();
```

```
        self.setButtonText();
        self.setButtonCount();

        self.list.find( "li:not(.ui-li-divider)" )
            .find( "a" ).removeClass( $.mobile.activeBtnClass ).end()
            .attr( "aria-selected", false )
            .each(function( i ) {

                if ( $.inArray( i, indices ) > -1 ) {
                    var item = $( this );

                    // Aria selected attr
                    item.attr( "aria-selected", true );

                    // Multiple selects: add the "on" checkbox state to the icon
                    if ( self.isMultiple ) {
                        item.find( "a" ).removeClass( "ui-checkbox-off" ).addClass(
                        "ui-checkbox-on" );
                    } else {
                        if ( item.hasClass( "ui-screen-hidden" ) ) {
                            item.next().find( "a" ).addClass( $.mobile.activeBtnClass );
                        } else {
                            item.find( "a" ).addClass( $.mobile.activeBtnClass );
                        }
                    }
                }
            });
    },

    close: function() {
        if ( this.options.disabled || !this.isOpen ) {
            return;
        }

        var self = this;

        if ( self.menuType === "page" ) {
            self.menuPage.dialog( "close" );
            self.list.appendTo( self.listbox );
        } else {
            self.listbox.popup( "close" );
        }

        self._focusButton();
        // allow the dialog to be closed again
        self.isOpen = false;
    },

    open: function() {
        this.button.click();
    },

    _focusMenuItem: function() {
```

```
        var selector = this.list.find( "a." + $.mobile.activeBtnClass );
        if ( selector.length === 0 ) {
            selector = this.list.find( "li:not(" + unfocusableItemSelector + ") a.ui-btn" );
        }
        selector.first().focus();
    },

    _decideFormat: function() {
        var self = this,
            $window = this.window,
            selfListParent = self.list.parent(),
            menuHeight = selfListParent.outerHeight(),
            scrollTop = $window.scrollTop(),
            btnOffset = self.button.offset().top,
            screenHeight = $window.height();

        if ( menuHeight > screenHeight - 80 || !$.support.scrollTop ) {

            self.menuPage.appendTo( $.mobile.pageContainer ).page();
            self.menuPageContent = self.menuPage.find( ".ui-content" );
            self.menuPageClose = self.menuPage.find( ".ui-header a" );

            // prevent the parent page from being removed from the DOM,
            // otherwise the results of selecting a list item in the dialog
            // fall into a black hole
            self.thisPage.unbind( "pagehide.remove" );

            //for WebOS/Opera Mini (set lastscroll using button offset)
            if ( scrollTop === 0 && btnOffset > screenHeight ) {
                self.thisPage.one( "pagehide", function() {
                    $( this ).jqmData( "lastScroll", btnOffset );
                });
            }

            self.menuPage.one( {
                pageshow: $.proxy( this, "_focusMenuItem" ),
                pagehide: $.proxy( this, "close" )
            });

            self.menuType = "page";
            self.menuPageContent.append( self.list );
            self.menuPage.find( "div .ui-title" ).text( self.label.text() );
        } else {
            self.menuType = "overlay";

            self.listbox.one( { popupafteropen: $.proxy( this, "_focusMenuItem" ) } );
        }
    },

    _buildList: function() {
        var self = this,
            o = this.options,
            placeholder = this.placeholder,
            needPlaceholder = true,
```

```
            dataIcon = "false",
            $options, numOptions, select,
            dataPrefix = "data-" + $.mobile.ns,
            dataIndexAttr = dataPrefix + "option-index",
            dataIconAttr = dataPrefix + "icon",
            dataRoleAttr = dataPrefix + "role",
            dataPlaceholderAttr = dataPrefix + "placeholder",
            fragment = document.createDocumentFragment(),
            isPlaceholderItem = false,
            optGroup,
            i,
            option, $option, parent, text, anchor, classes,
            optLabel, divider, item;

        self.list.empty().filter( ".ui-listview" ).listview( "destroy" );
        $options = this._selectOptions();
        numOptions = $options.length;
        select = this.select[ 0 ];

        for ( i = 0; i < numOptions;i++, isPlaceholderItem = false) {
            option = $options[i];
            $option = $( option );

            // Do not create options based on ui-screen-hidden select options
            if ( $option.hasClass( "ui-screen-hidden" ) ) {
                continue;
            }

            parent = option.parentNode;
            text = $option.text();
            anchor  = document.createElement( "a" );
            classes = [];

            anchor.setAttribute( "href", "#" );
            anchor.appendChild( document.createTextNode( text ) );

            // Are we inside an optgroup?
            if ( parent !== select && parent.nodeName.toLowerCase() === "optgroup" ) {
                optLabel = parent.getAttribute( "label" );
                if ( optLabel !== optGroup ) {
                    divider = document.createElement( "li" );
                    divider.setAttribute( dataRoleAttr, "list-divider" );
                    divider.setAttribute( "role", "option" );
                    divider.setAttribute( "tabindex", "-1" );
                    divider.appendChild( document.createTextNode( optLabel ) );
                    fragment.appendChild( divider );
                    optGroup = optLabel;
                }
            }

            if ( needPlaceholder && ( !option.getAttribute( "value" ) || text.length === 0 ||
            $option.jqmData( "placeholder" ) ) ) {
                needPlaceholder = false;
                isPlaceholderItem = true;
```

```
            // If we have identified a placeholder, record the fact that it was
            // us who have added the placeholder to the option and mark it
            // retroactively in the select as well
            if ( null === option.getAttribute( dataPlaceholderAttr ) ) {
                this._removePlaceholderAttr = true;
            }
            option.setAttribute( dataPlaceholderAttr, true );
            if ( o.hidePlaceholderMenuItems ) {
                classes.push( "ui-screen-hidden" );
            }
            if ( placeholder !== text ) {
                placeholder = self.placeholder = text;
            }
        }

        item = document.createElement( "li" );
        if ( option.disabled ) {
            classes.push( "ui-state-disabled" );
            item.setAttribute( "aria-disabled", true );
        }
        item.setAttribute( dataIndexAttr, i );
        item.setAttribute( dataIconAttr, dataIcon );
        if ( isPlaceholderItem ) {
            item.setAttribute( dataPlaceholderAttr, true );
        }
        item.className = classes.join( " " );
        item.setAttribute( "role", "option" );
        anchor.setAttribute( "tabindex", "-1" );
        if ( this.isMultiple ) {
            $( anchor ).addClass( "ui-btn ui-checkbox-off ui-btn-icon-right" );
        }

        item.appendChild( anchor );
        fragment.appendChild( item );
    }

    self.list[0].appendChild( fragment );

    // Hide header if it's not a multiselect and there's no placeholder
    if ( !this.isMultiple && !placeholder.length ) {
        this.header.addClass( "ui-screen-hidden" );
    } else {
        this.headerTitle.text( this.placeholder );
    }

    // Now populated, create listview
    self.list.listview();
},

_button: function() {
    return this.options.nativeMenu ?
        this._super() :
        $( "<a>", {
```

```
                "href": "#",
                "role": "button",
                // TODO value is undefined at creation
                "id": this.buttonId,
                "aria-haspopup": "true",

                // TODO value is undefined at creation
                "aria-owns": this.menuId
            });
    },

    _destroy: function() {

        if ( !this.options.nativeMenu ) {
            this.close();

            // Restore the tabindex attribute to its original value
            if ( this._origTabIndex !== undefined ) {
                if ( this._origTabIndex !== false ) {
                    this.select.attr( "tabindex", this._origTabIndex );
                } else {
                    this.select.removeAttr( "tabindex" );
                }
            }

            // Remove the placeholder attribute if we were the ones to add it
            if ( this._removePlaceholderAttr ) {
                this._selectOptions().removeAttr( "data-" + $.mobile.ns + "placeholder" );
            }

            // Remove the popup
            this.listbox.remove();

            // Remove the dialog
            this.menuPage.remove();
        }

        // Chain up
        this._super();
    }
});

})( jQuery );


// buttonMarkup is deprecated as of 1.4.0 and will be removed in 1.5.0.

(function( $, undefined ) {

// General policy: Do not access data-* attributes except during enhancement.
// In all other cases we determine the state of the button exclusively from its
// className. That's why optionsToClasses expects a full complement of options,
// and the jQuery plugin completes the set of options from the default values.
```

```
// Map classes to buttonMarkup boolean options - used in classNameToOptions()
var reverseBoolOptionMap = {
        "ui-shadow" : "shadow",
        "ui-corner-all" : "corners",
        "ui-btn-inline" : "inline",
        "ui-shadow-icon" : "iconshadow", /* TODO: Remove in 1.5 */
        "ui-mini" : "mini"
    },
    getAttrFixed = function() {
        var ret = $.mobile.getAttribute.apply( this, arguments );

        return ( ret == null ? undefined : ret );
    },
    capitalLettersRE = /[A-Z]/g;

// optionsToClasses:
// @options: A complete set of options to convert to class names.
// @existingClasses: extra classes to add to the result
//
// Converts @options to buttonMarkup classes and returns the result as an array
// that can be converted to an element's className with .join( " " ). All
// possible options must be set inside @options. Use $.fn.buttonMarkup.defaults
// to get a complete set and use $.extend to override your choice of options
// from that set.
function optionsToClasses( options, existingClasses ) {
    var classes = existingClasses ? existingClasses : [];

    // Add classes to the array - first ui-btn
    classes.push( "ui-btn" );

    // If there is a theme
    if ( options.theme ) {
        classes.push( "ui-btn-" + options.theme );
    }

    // If there's an icon, add the icon-related classes
    if ( options.icon ) {
        classes = classes.concat([
            "ui-icon-" + options.icon,
            "ui-btn-icon-" + options.iconpos
        ]);
        if ( options.iconshadow ) {
            classes.push( "ui-shadow-icon" ); /* TODO: Remove in 1.5 */
        }
    }

    // Add the appropriate class for each boolean option
    if ( options.inline ) {
        classes.push( "ui-btn-inline" );
    }
    if ( options.shadow ) {
        classes.push( "ui-shadow" );
    }
    if ( options.corners ) {
```

```
        classes.push( "ui-corner-all" );
    }
    if ( options.mini ) {
        classes.push( "ui-mini" );
    }

    // Create a string from the array and return it
    return classes;
}

// classNameToOptions:
// @classes: A string containing a .className-style space-separated class list
//
// Loops over @classes and calculates an options object based on the
// buttonMarkup-related classes it finds. It records unrecognized classes in an
// array.
//
// Returns: An object containing the following items:
//
// "options": buttonMarkup options found to be present because of the
// presence/absence of corresponding classes
//
// "unknownClasses": a string containing all the non-buttonMarkup-related
// classes found in @classes
//
// "alreadyEnhanced": A boolean indicating whether the ui-btn class was among
// those found to be present
function classNameToOptions( classes ) {
    var idx, map, unknownClass,
        alreadyEnhanced = false,
        noIcon = true,
        o = {
            icon: "",
            inline: false,
            shadow: false,
            corners: false,
            iconshadow: false,
            mini: false
        },
        unknownClasses = [];

    classes = classes.split( " " );

    // Loop over the classes
    for ( idx = 0 ; idx < classes.length ; idx++ ) {

        // Assume it's an unrecognized class
        unknownClass = true;

        // Recognize boolean options from the presence of classes
        map = reverseBoolOptionMap[ classes[ idx ] ];
        if ( map !== undefined ) {
            unknownClass = false;
            o[ map ] = true;
```

```
        // Recognize the presence of an icon and establish the icon position
        } else if ( classes[ idx ].indexOf( "ui-btn-icon-" ) === 0 ) {
            unknownClass = false;
            noIcon = false;
            o.iconpos = classes[ idx ].substring( 12 );

        // Establish which icon is present
        } else if ( classes[ idx ].indexOf( "ui-icon-" ) === 0 ) {
            unknownClass = false;
            o.icon = classes[ idx ].substring( 8 );

        // Establish the theme - this recognizes one-letter theme swatch names
        } else if ( classes[ idx ].indexOf( "ui-btn-" ) === 0 && classes[ idx ].length === 8 ) {
            unknownClass = false;
            o.theme = classes[ idx ].substring( 7 );

        // Recognize that this element has already been buttonMarkup-enhanced
        } else if ( classes[ idx ] === "ui-btn" ) {
            unknownClass = false;
            alreadyEnhanced = true;
        }

        // If this class has not been recognized, add it to the list
        if ( unknownClass ) {
            unknownClasses.push( classes[ idx ] );
        }
    }

    // If a "ui-btn-icon-*" icon position class is absent there cannot be an icon
    if ( noIcon ) {
        o.icon = "";
    }

    return {
        options: o,
        unknownClasses: unknownClasses,
        alreadyEnhanced: alreadyEnhanced
    };
}

function camelCase2Hyphenated( c ) {
    return "-" + c.toLowerCase();
}

// $.fn.buttonMarkup:
// DOM: gets/sets .className
//
// @options: options to apply to the elements in the jQuery object
// @overwriteClasses: boolean indicating whether to honour existing classes
//
// Calculates the classes to apply to the elements in the jQuery object based on
// the options passed in. If @overwriteClasses is true, it sets the className
// property of each element in the jQuery object to the buttonMarkup classes
```

```
// it calculates based on the options passed in.
//
// If you wish to preserve any classes that are already present on the elements
// inside the jQuery object, including buttonMarkup-related classes that were
// added by a previous call to $.fn.buttonMarkup() or during page enhancement
// then you should omit @overwriteClasses or set it to false.
$.fn.buttonMarkup = function( options, overwriteClasses ) {
    var idx, data, el, retrievedOptions, optionKey,
        defaults = $.fn.buttonMarkup.defaults;

    for ( idx = 0 ; idx < this.length ; idx++ ) {
        el = this[ idx ];
        data = overwriteClasses ?

            // Assume this element is not enhanced and ignore its classes
            { alreadyEnhanced: false, unknownClasses: [] } :

            // Otherwise analyze existing classes to establish existing options and
            // classes
            classNameToOptions( el.className );

        retrievedOptions = $.extend( {},

            // If the element already has the class ui-btn, then we assume that
            // it has passed through buttonMarkup before - otherwise, the options
            // returned by classNameToOptions do not correctly reflect the state of
            // the element
            ( data.alreadyEnhanced ? data.options : {} ),

            // Finally, apply the options passed in
            options );

        // If this is the first call on this element, retrieve remaining options
        // from the data-attributes
        if ( !data.alreadyEnhanced ) {
            for ( optionKey in defaults ) {
                if ( retrievedOptions[ optionKey ] === undefined ) {
                    retrievedOptions[ optionKey ] = getAttrFixed( el,
                        optionKey.replace( capitalLettersRE, camelCase2Hyphenated )
                    );
                }
            }
        }

        el.className = optionsToClasses(

            // Merge all the options and apply them as classes
            $.extend( {},

                // The defaults form the basis
                defaults,

                // Add the computed options
                retrievedOptions
```

```
            ),

            // ... and re-apply any unrecognized classes that were found
            data.unknownClasses ).join( " " );
        if ( el.tagName.toLowerCase() !== "button" ) {
            el.setAttribute( "role", "button" );
        }
    }

    return this;
};

// buttonMarkup defaults. This must be a complete set, i.e., a value must be
// given here for all recognized options
$.fn.buttonMarkup.defaults = {
    icon: "",
    iconpos: "left",
    theme: null,
    inline: false,
    shadow: true,
    corners: true,
    iconshadow: false, /* TODO: Remove in 1.5. Option deprecated in 1.4. */
    mini: false
};

$.extend( $.fn.buttonMarkup, {
    initSelector: "a:jqmData(role='button'), .ui-bar > a, .ui-bar >
    :jqmData(role='controlgroup') > a, button"
});

})( jQuery );


(function( $, undefined ) {

$.widget( "mobile.controlgroup", $.extend( {
    options: {
        enhanced: false,
        theme: null,
        shadow: false,
        corners: true,
        excludeInvisible: true,
        type: "vertical",
        mini: false
    },

    _create: function() {
        var elem = this.element,
            opts = this.options;

        // Run buttonmarkup
        if ( $.fn.buttonMarkup ) {
            this.element.find( $.fn.buttonMarkup.initSelector ).buttonMarkup();
        }
```

```
        // Enhance child widgets
        $.each( this._childWidgets, $.proxy( function( number, widgetName ) {
            if ( $.mobile[ widgetName ] ) {
                this.element.find( $.mobile[ widgetName ].initSelector ).not( $.mobile.page.
                prototype.keepNativeSelector() )[ widgetName ]();
            }
        }, this ));

        $.extend( this, {
            _ui: null,
            _initialRefresh: true
        });

        if ( opts.enhanced ) {
            this._ui = {
                groupLegend: elem.children( ".ui-controlgroup-label" ).children(),
                childWrapper: elem.children( ".ui-controlgroup-controls" )
            };
        } else {
            this._ui = this._enhance();
        }

    },

    _childWidgets: [ "checkboxradio", "selectmenu", "button" ],

    _themeClassFromOption: function( value ) {
        return ( value ? ( value === "none" ? "" : "ui-group-theme-" + value ) : "" );
    },

    _enhance: function() {
        var elem = this.element,
            opts = this.options,
            ui = {
                groupLegend: elem.children( "legend" ),
                childWrapper: elem
                    .addClass( "ui-controlgroup " +
                        "ui-controlgroup-" +
                            ( opts.type === "horizontal" ? "horizontal" : "vertical" ) + " " +
                        this._themeClassFromOption( opts.theme ) + " " +
                        ( opts.corners ? "ui-corner-all " : "" ) +
                        ( opts.mini ? "ui-mini " : "" ) )
                    .wrapInner( "<div " +
                        "class='ui-controlgroup-controls " +
                            ( opts.shadow === true ? "ui-shadow" : "" ) + "'></div>" )
                    .children()
            };

        if ( ui.groupLegend.length > 0 ) {
            $( "<div role='heading' class='ui-controlgroup-label'></div>" )
                .append( ui.groupLegend )
                .prependTo( elem );
        }
```

```
        return ui;
    },

    _init: function() {
        this.refresh();
    },

    _setOptions: function( options ) {
        var callRefresh, returnValue,
            elem = this.element;

        // Must have one of horizontal or vertical
        if ( options.type !== undefined ) {
            elem
                .removeClass( "ui-controlgroup-horizontal ui-controlgroup-vertical" )
                .addClass( "ui-controlgroup-" + ( options.type === "horizontal" ? "horizontal" :
                 "vertical" ) );
            callRefresh = true;
        }

        if ( options.theme !== undefined ) {
            elem
                .removeClass( this._themeClassFromOption( this.options.theme ) )
                .addClass( this._themeClassFromOption( options.theme ) );
        }

        if ( options.corners !== undefined ) {
            elem.toggleClass( "ui-corner-all", options.corners );
        }

        if ( options.mini !== undefined ) {
            elem.toggleClass( "ui-mini", options.mini );
        }

        if ( options.shadow !== undefined ) {
            this._ui.childWrapper.toggleClass( "ui-shadow", options.shadow );
        }

        if ( options.excludeInvisible !== undefined ) {
            this.options.excludeInvisible = options.excludeInvisible;
            callRefresh = true;
        }

        returnValue = this._super( options );

        if ( callRefresh ) {
            this.refresh();
        }

        return returnValue;
    },

    container: function() {
        return this._ui.childWrapper;
```

```
    },

    refresh: function() {
        var $el = this.container(),
            els = $el.find( ".ui-btn" ).not( ".ui-slider-handle" ),
            create = this._initialRefresh;
        if ( $.mobile.checkboxradio ) {
            $el.find( ":mobile-checkboxradio" ).checkboxradio( "refresh" );
        }
        this._addFirstLastClasses( els,
            this.options.excludeInvisible ? this._getVisibles( els, create ) : els,
            create );
        this._initialRefresh = false;
    },

    // Caveat: If the legend is not the first child of the controlgroup at enhance
    // time, it will be after _destroy().
    _destroy: function() {
        var ui, buttons,
            opts = this.options;

        if ( opts.enhanced ) {
            return this;
        }

        ui = this._ui;
        buttons = this.element
            .removeClass( "ui-controlgroup " +
                "ui-controlgroup-horizontal ui-controlgroup-vertical ui-corner-all ui-mini " +
                this._themeClassFromOption( opts.theme ) )
            .find( ".ui-btn" )
            .not( ".ui-slider-handle" );

        this._removeFirstLastClasses( buttons );

        ui.groupLegend.unwrap();
        ui.childWrapper.children().unwrap();
    }
}, $.mobile.behaviors.addFirstLastClasses ) );

})(jQuery);

(function( $, undefined ) {

    $.widget( "mobile.toolbar", {
        initSelector: ":jqmData(role='footer'), :jqmData(role='header')",

        options: {
            theme: null,
            addBackBtn: false,
            backBtnTheme: null,
            backBtnText: "Back"
        },
```

```
        _create: function() {
            var leftbtn, rightbtn,
                role =  this.element.is( ":jqmData(role='header')" ) ? "header" : "footer",
                page = this.element.closest( ".ui-page" );
            if ( page.length === 0 ) {
                page = false;
                this._on( this.document, {
                    "pageshow": "refresh"
                });
            }
            $.extend( this, {
                role: role,
                page: page,
                leftbtn: leftbtn,
                rightbtn: rightbtn
            });
            this.element.attr( "role", role === "header" ? "banner" : "contentinfo" ).addClass(
            "ui-" + role );
            this.refresh();
            this._setOptions( this.options );
        },
        _setOptions: function( o ) {
            if ( o.addBackBtn !== undefined ) {
                if ( this.options.addBackBtn &&
                    this.role === "header" &&
                    $( ".ui-page" ).length > 1 &&
                    this.page[ 0 ].getAttribute( "data-" + $.mobile.ns + "url" ) !== $.mobile.
                    path.stripHash( location.hash ) &&
                    !this.leftbtn ) {
                        this._addBackButton();
                } else {
                    this.element.find( ".ui-toolbar-back-btn" ).remove();
                }
            }
            if ( o.backBtnTheme != null ) {
                this.element
                    .find( ".ui-toolbar-back-btn" )
                    .addClass( "ui-btn ui-btn-" + o.backBtnTheme );
            }
            if ( o.backBtnText !== undefined ) {
                this.element.find( ".ui-toolbar-back-btn .ui-btn-text" ).text( o.backBtnText );
            }
            if ( o.theme !== undefined ) {
                var currentTheme = this.options.theme ? this.options.theme : "inherit",
                    newTheme = o.theme ? o.theme : "inherit";

                this.element.removeClass( "ui-bar-" + currentTheme ).addClass( "ui-bar-" +
                newTheme );
            }

            this._super( o );
        },
        refresh: function() {
            if ( this.role === "header" ) {
```

```
            this._addHeaderButtonClasses();
        }
        if ( !this.page ) {
            this._setRelative();
            if ( this.role === "footer" ) {
                this.element.appendTo( "body" );
            }
        }
        this._addHeadingClasses();
        this._btnMarkup();
    },

    //we only want this to run on non fixed toolbars so make it easy to override
    _setRelative: function() {
        $( "[data-"+ $.mobile.ns + "role='page']" ).css({ "position": "relative" });
    },

    // Deprecated in 1.4. As from 1.5 button classes have to be present in the markup.
    _btnMarkup: function() {
        this.element
            .children( "a" )
            .filter( ":not([data-" + $.mobile.ns + "role='none'])" )
            .attr( "data-" + $.mobile.ns + "role", "button" );
        this.element.trigger( "create" );
    },
    // Deprecated in 1.4. As from 1.5 ui-btn-left/right classes have to be present in the
    markup.
    _addHeaderButtonClasses: function() {
        var $headeranchors = this.element.children( "a, button" );
        this.leftbtn = $headeranchors.hasClass( "ui-btn-left" );
        this.rightbtn = $headeranchors.hasClass( "ui-btn-right" );

        this.leftbtn = this.leftbtn || $headeranchors.eq( 0 ).not( ".ui-btn-right" ).
        addClass( "ui-btn-left" ).length;

        this.rightbtn = this.rightbtn || $headeranchors.eq( 1 ).addClass( "ui-btn-right" ).
        length;

    },
    _addBackButton: function() {
        var options = this.options,
            theme = options.backBtnTheme || options.theme;

        $( "<a role='button' href='javascript:void(0);' " +
            "class='ui-btn ui-corner-all ui-shadow ui-btn-left " +
                ( theme ? "ui-btn-" + theme + " " : "" ) +
                "ui-toolbar-back-btn ui-icon-carat-l ui-btn-icon-left' " +
            "data-" + $.mobile.ns + "rel='back'>" + options.backBtnText + "</a>" )
                .prependTo( this.element );
    },
    _addHeadingClasses: function() {
        this.element.children( "h1, h2, h3, h4, h5, h6" )
            .addClass( "ui-title" )
            // Regardless of h element number in src, it becomes h1 for the enhanced page
```

```
                .attr({
                    "role": "heading",
                    "aria-level": "1"
                });
        }
    });

})( jQuery );

(function( $, undefined ) {

    $.widget( "mobile.toolbar", $.mobile.toolbar, {
        options: {
            position:null,
            visibleOnPageShow: true,
            disablePageZoom: true,
            transition: "slide", //can be none, fade, slide (slide maps to slideup or slidedown)
            fullscreen: false,
            tapToggle: true,
            tapToggleBlacklist: "a, button, input, select, textarea, .ui-header-fixed,
            .ui-footer-fixed, .ui-flipswitch, .ui-popup, .ui-panel, .ui-panel-dismiss-open",
            hideDuringFocus: "input, textarea, select",
            updatePagePadding: true,
            trackPersistentToolbars: true,

            // Browser detection! Weeee, here we go...
            // Unfortunately, position:fixed is costly, not to mention probably impossible, to
            feature-detect accurately.
            // Some tests exist, but they currently return false results in critical devices
            and browsers, which could lead to a broken experience.
            // Testing fixed positioning is also pretty obtrusive to page load, requiring
            injected elements and scrolling the window
            // The following function serves to rule out some popular browsers with known
            fixed-positioning issues
            // This is a plugin option like any other, so feel free to improve or overwrite it
            supportBlacklist: function() {
                return !$.support.fixedPosition;
            }
        },

        _create: function() {
            this._super();
            if ( this.options.position === "fixed" && !this.options.supportBlacklist() ) {
                this._makeFixed();
            }
        },

        _makeFixed: function() {
            this.element.addClass( "ui-"+ this.role +"-fixed" );
            this.updatePagePadding();
            this._addTransitionClass();
            this._bindPageEvents();
            this._bindToggleHandlers();
        },
```

```
        _setOptions: function( o ) {
            if ( o.position === "fixed" && this.options.position !== "fixed" ) {
                this._makeFixed();
            }
            if ( this.options.position === "fixed" && !this.options.supportBlacklist() ) {
                var $page = ( !!this.page )? this.page: ( $(".ui-page-active").length > 0 )? $(
                ".ui-page-active"): $(".ui-page").eq(0);

                if ( o.fullscreen !== undefined) {
                    if ( o.fullscreen ) {
                        this.element.addClass( "ui-"+ this.role +"-fullscreen" );
                        $page.addClass( "ui-page-" + this.role + "-fullscreen" );
                    }
                    // If not fullscreen, add class to page to set top or bottom padding
                    else {
                        this.element.removeClass( "ui-"+ this.role +"-fullscreen" );
                        $page.removeClass( "ui-page-" + this.role + "-fullscreen" ).addClass(
                        "ui-page-" + this.role+ "-fixed" );
                    }
                }
            }
            this._super(o);
        },

        _addTransitionClass: function() {
            var tclass = this.options.transition;

            if ( tclass && tclass !== "none" ) {
                // use appropriate slide for header or footer
                if ( tclass === "slide" ) {
                    tclass = this.element.hasClass( "ui-header" ) ? "slidedown" : "slideup";
                }

                this.element.addClass( tclass );
            }
        },

        _bindPageEvents: function() {
            var page = ( !!this.page )? this.element.closest( ".ui-page" ): this.document;
            //page event bindings
            // Fixed toolbars require page zoom to be disabled, otherwise usability issues crop
            up
            // This method is meant to disable zoom while a fixed-positioned toolbar page is
            visible
            this._on( page , {
                "pagebeforeshow": "_handlePageBeforeShow",
                "webkitAnimationStart":"_handleAnimationStart",
                "animationstart":"_handleAnimationStart",
                "updatelayout": "_handleAnimationStart",
                "pageshow": "_handlePageShow",
                "pagebeforehide": "_handlePageBeforeHide"
            });
        },
```

```
        _handlePageBeforeShow: function( ) {
            var o = this.options;
            if ( o.disablePageZoom ) {
                $.mobile.zoom.disable( true );
            }
            if ( !o.visibleOnPageShow ) {
                this.hide( true );
            }
        },

        _handleAnimationStart: function() {
            if ( this.options.updatePagePadding ) {
                this.updatePagePadding( ( !!this.page )? this.page: ".ui-page-active" );
            }
        },

        _handlePageShow: function() {
            this.updatePagePadding( ( !!this.page )? this.page: ".ui-page-active" );
            if ( this.options.updatePagePadding ) {
                this._on( this.window, { "throttledresize": "updatePagePadding" } );
            }
        },

        _handlePageBeforeHide: function( e, ui ) {
            var o = this.options,
                thisFooter, thisHeader, nextFooter, nextHeader;

            if ( o.disablePageZoom ) {
                $.mobile.zoom.enable( true );
            }
            if ( o.updatePagePadding ) {
                this._off( this.window, "throttledresize" );
            }

            if ( o.trackPersistentToolbars ) {
                thisFooter = $( ".ui-footer-fixed:jqmData(id)", this.page );
                thisHeader = $( ".ui-header-fixed:jqmData(id)", this.page );
                nextFooter = thisFooter.length && ui.nextPage && $(
                ".ui-footer-fixed:jqmData(id='" + thisFooter.jqmData( "id" ) + "')", ui.nextPage
                 ) || $();
                nextHeader = thisHeader.length && ui.nextPage && $(
                ".ui-header-fixed:jqmData(id='" + thisHeader.jqmData( "id" ) + "')", ui.nextPage
                 ) || $();

                if ( nextFooter.length || nextHeader.length ) {

                    nextFooter.add( nextHeader ).appendTo( $.mobile.pageContainer );

                    ui.nextPage.one( "pageshow", function() {
                        nextHeader.prependTo( this );
                        nextFooter.appendTo( this );
                    });
                }
```

```
            }
        },

        _visible: true,

        // This will set the content element's top or bottom padding equal to the toolbar's
        height
        updatePagePadding: function( tbPage ) {
            var $el = this.element,
                header = ( this.role ==="header" ),
                pos = parseFloat( $el.css( header ? "top" : "bottom" ) );

            // This behavior only applies to "fixed", not "fullscreen"
            if ( this.options.fullscreen ) { return; }
            // tbPage argument can be a Page object or an event, if coming from throttled resize.
            tbPage = ( tbPage && tbPage.type === undefined && tbPage ) || this.page || $el.
            closest( ".ui-page" );
            tbPage = ( !!this.page )? this.page: ".ui-page-active";
            $( tbPage ).css( "padding-" + ( header ? "top" : "bottom" ), $el.outerHeight() + pos
             );
        },

        _useTransition: function( notransition ) {
            var $win = this.window,
                $el = this.element,
                scroll = $win.scrollTop(),
                elHeight = $el.height(),
                pHeight = ( !!this.page )? $el.closest( ".ui-page" ).height():$(
                ".ui-page-active").height(),
                viewportHeight = $.mobile.getScreenHeight();

            return !notransition &&
                ( this.options.transition && this.options.transition !== "none" &&
                (
                    ( this.role === "header" && !this.options.fullscreen && scroll > elHeight )
                    ||
                    ( this.role === "footer" && !this.options.fullscreen && scroll +
                    viewportHeight < pHeight - elHeight )
                ) || this.options.fullscreen
                );
        },

        show: function( notransition ) {
            var hideClass = "ui-fixed-hidden",
                $el = this.element;

            if ( this._useTransition( notransition ) ) {
                $el
                    .removeClass( "out " + hideClass )
                    .addClass( "in" )
                    .animationComplete(function () {
                        $el.removeClass( "in" );
                    });
            }
```

```
            else {
                $el.removeClass( hideClass );
            }
            this._visible = true;
        },

        hide: function( notransition ) {
            var hideClass = "ui-fixed-hidden",
                $el = this.element,
                // if it's a slide transition, our new transitions need the reverse class as
                well to slide outward
                outclass = "out" + ( this.options.transition === "slide" ? " reverse" : "" );

            if ( this._useTransition( notransition ) ) {
                $el
                    .addClass( outclass )
                    .removeClass( "in" )
                    .animationComplete(function() {
                        $el.addClass( hideClass ).removeClass( outclass );
                    });
            }
            else {
                $el.addClass( hideClass ).removeClass( outclass );
            }
            this._visible = false;
        },

        toggle: function() {
            this[ this._visible ? "hide" : "show" ]();
        },

        _bindToggleHandlers: function() {
            var self = this,
                o = self.options,
                delayShow, delayHide,
                isVisible = true,
                page = ( !!this.page )? this.page: $(".ui-page");

            // tap toggle
            page
                .bind( "vclick", function( e ) {
                    if ( o.tapToggle && !$( e.target ).closest( o.tapToggleBlacklist ).length ) {
                        self.toggle();
                    }
                })
                .bind( "focusin focusout", function( e ) {
                    //this hides the toolbars on a keyboard pop to give more screen room and
                    prevent ios bug which
                    //positions fixed toolbars in the middle of the screen on pop if the input
                    is near the top or
                    //bottom of the screen addresses issues #4410 Footer navbar moves up when
                    clicking on a textbox in an Android environment
                    //and issue #4113 Header and footer change their position after keyboard
                    popup - iOS
```

```
                    //and issue #4410 Footer navbar moves up when clicking on a textbox in an
                    Android environment
                    if ( screen.width < 1025 && $( e.target ).is( o.hideDuringFocus ) && !$( e.
                    target ).closest( ".ui-header-fixed, .ui-footer-fixed" ).length ) {
                        //Fix for issue #4724 Moving through form in Mobile Safari with "Next"
                        and "Previous" system
                        //controls causes fixed position, tap-toggle false Header to reveal
                        itself
                        // isVisible instead of self._visible because the focusin and focusout
                        events fire twice at the same time
                        // Also use a delay for hiding the toolbars because on Android native
                        browser focusin is direclty followed
                        // by a focusout when a native selects opens and the other way around
                        when it closes.
                        if ( e.type === "focusout" && !isVisible ) {
                            isVisible = true;
                            //wait for the stack to unwind and see if we have jumped to another
                            input
                            clearTimeout( delayHide );
                            delayShow = setTimeout( function() {
                                self.show();
                            }, 0 );
                        } else if ( e.type === "focusin" && !!isVisible ) {
                            //if we have jumped to another input clear the time out to cancel
                            the show.
                            clearTimeout( delayShow );
                            isVisible = false;
                            delayHide = setTimeout( function() {
                                self.hide();
                            }, 0 );
                        }
                    }
                });
        },

        _setRelative: function() {
            if( this.options.position !== "fixed" ){
                $( "[data-"+ $.mobile.ns + "role='page']" ).css({ "position": "relative" });
            }
        },

        _destroy: function() {
            var $el = this.element,
                header = $el.hasClass( "ui-header" );

            $el.closest( ".ui-page" ).css( "padding-" + ( header ? "top" : "bottom" ), "" );
            $el.removeClass( "ui-header-fixed ui-footer-fixed ui-header-fullscreen
            ui-footer-fullscreen in out fade slidedown slideup ui-fixed-hidden" );
            $el.closest( ".ui-page" ).removeClass( "ui-page-header-fixed ui-page-footer-fixed
            ui-page-header-fullscreen ui-page-footer-fullscreen" );
        }

    });
})( jQuery );
```

```
(function( $, undefined ) {
    $.widget( "mobile.toolbar", $.mobile.toolbar, {

        _makeFixed: function() {
            this._super();
            this._workarounds();
        },

        //check the browser and version and run needed workarounds
        _workarounds: function() {
            var ua = navigator.userAgent,
            platform = navigator.platform,
            // Rendering engine is Webkit, and capture major version
            wkmatch = ua.match( /AppleWebKit\/([0-9]+)/ ),
            wkversion = !!wkmatch && wkmatch[ 1 ],
            os = null,
            self = this;
            //set the os we are working in if it dosent match one with workarounds return
            if ( platform.indexOf( "iPhone" ) > -1 || platform.indexOf( "iPad" ) > -1  ||
            platform.indexOf( "iPod" ) > -1 ) {
                os = "ios";
            } else if ( ua.indexOf( "Android" ) > -1 ) {
                os = "android";
            } else {
                return;
            }
            //check os version if it dosent match one with workarounds return
            if ( os === "ios" ) {
                //iOS  workarounds
                self._bindScrollWorkaround();
            } else if ( os === "android" && wkversion && wkversion < 534 ) {
                //Android 2.3 run all Android 2.3 workaround
                self._bindScrollWorkaround();
                self._bindListThumbWorkaround();
            } else {
                return;
            }
        },

        //Utility class for checking header and footer positions relative to viewport
        _viewportOffset: function() {
            var $el = this.element,
                header = $el.hasClass( "ui-header" ),
                offset = Math.abs( $el.offset().top - this.window.scrollTop() );
            if ( !header ) {
                offset = Math.round( offset - this.window.height() + $el.outerHeight() ) - 60;
            }
            return offset;
        },

        //bind events for _triggerRedraw() function
        _bindScrollWorkaround: function() {
            var self = this;
```

```
            //bind to scrollstop and check if the toolbars are correctly positioned
            this._on( this.window, { scrollstop: function() {
                var viewportOffset = self._viewportOffset();
                //check if the header is visible and if its in the right place
                if ( viewportOffset > 2 && self._visible ) {
                    self._triggerRedraw();
                }
            }});
        },

        //this addresses issue #4250 Persistent footer instability in v1.1 with long select
        lists in Android 2.3.3
        //and issue #3748 Android 2.x: Page transitions broken when fixed toolbars used
        //the absolutely positioned thumbnail in a list view causes problems with fixed
        position buttons above in a nav bar
        //setting the li's to -webkit-transform:translate3d(0,0,0); solves this problem to
        avoide potential issues in other
        //platforms we scope this with the class ui-android-2x-fix
        _bindListThumbWorkaround: function() {
            this.element.closest( ".ui-page" ).addClass( "ui-android-2x-fixed" );
        },
        //this addresses issues #4337 Fixed header problem after scrolling content on iOS and
        Android
        //and device bugs project issue #1 Form elements can lose click hit area in position:
        fixed containers.
        //this also addresses not on fixed toolbars page in docs
        //adding 1px of padding to the bottom then removing it causes a "redraw"
        //which positions the toolbars correctly (they will always be visually correct)
        _triggerRedraw: function() {
            var paddingBottom = parseFloat( $( ".ui-page-active" ).css( "padding-bottom" ) );
            //trigger page redraw to fix incorrectly positioned fixed elements
            $( ".ui-page-active" ).css( "padding-bottom", ( paddingBottom + 1 ) + "px" );
            //if the padding is reset with out a timeout the reposition will not occure.
            //this is independant of JQM the browser seems to need the time to react.
            setTimeout( function() {
                $( ".ui-page-active" ).css( "padding-bottom", paddingBottom + "px" );
            }, 0 );
        },

        destroy: function() {
            this._super();
            //Remove the class we added to the page previously in android 2.x
            this.element.closest( ".ui-page-active" ).removeClass( "ui-android-2x-fix" );
        }
    });

})( jQuery );


( function( $, undefined ) {

var ieHack = ( $.mobile.browser.oldIE && $.mobile.browser.oldIE <= 8 ),
    uiTemplate = $(
        "<div class='ui-popup-arrow-guide'></div>" +
```

```
        "<div class='ui-popup-arrow-container" + ( ieHack ? " ie" : "" ) + "'>" +
            "<div class='ui-popup-arrow'></div>" +
        "</div>"
    );

function getArrow() {
    var clone = uiTemplate.clone(),
        gd = clone.eq( 0 ),
        ct = clone.eq( 1 ),
        ar = ct.children();

    return { arEls: ct.add( gd ), gd: gd, ct: ct, ar: ar };
}

$.widget( "mobile.popup", $.mobile.popup, {
    options: {

        arrow: ""
    },

    _create: function() {
        var ar,
            ret = this._super();

        if ( this.options.arrow ) {
            this._ui.arrow = ar = this._addArrow();
        }

        return ret;
    },

    _addArrow: function() {
        var theme,
            opts = this.options,
            ar = getArrow();

        theme = this._themeClassFromOption( "ui-body-", opts.theme );
        ar.ar.addClass( theme + ( opts.shadow ? " ui-overlay-shadow" : "" ) );
        ar.arEls.hide().appendTo( this.element );

        return ar;
    },

    _unenhance: function() {
        var ar = this._ui.arrow;

        if ( ar ) {
            ar.arEls.remove();
        }

        return this._super();
    },

    // Pretend to show an arrow described by @p and @dir and calculate the
```

```
    // distance from the desired point. If a best-distance is passed in, return
    // the minimum of the one passed in and the one calculated.
    _tryAnArrow: function( p, dir, desired, s, best ) {
        var result, r, diff, desiredForArrow = {}, tip = {};

        // If the arrow has no wiggle room along the edge of the popup, it cannot
        // be displayed along the requested edge without it sticking out.
        if ( s.arFull[ p.dimKey ] > s.guideDims[ p.dimKey ] ) {
            return best;
        }

        desiredForArrow[ p.fst ] = desired[ p.fst ] +
            ( s.arHalf[ p.oDimKey ] + s.menuHalf[ p.oDimKey ] ) * p.offsetFactor -
            s.contentBox[ p.fst ] + ( s.clampInfo.menuSize[ p.oDimKey ] - s.contentBox[ p.
            oDimKey ] ) * p.arrowOffsetFactor;
        desiredForArrow[ p.snd ] = desired[ p.snd ];

        result = s.result || this._calculateFinalLocation( desiredForArrow, s.clampInfo );
        r = { x: result.left, y: result.top };

        tip[ p.fst ] = r[ p.fst ] + s.contentBox[ p.fst ] + p.tipOffset;
        tip[ p.snd ] = Math.max( result[ p.prop ] + s.guideOffset[ p.prop ] + s.arHalf[ p.dimKey
         ],
            Math.min( result[ p.prop ] + s.guideOffset[ p.prop ] + s.guideDims[ p.dimKey ] - s.
            arHalf[ p.dimKey ],
                desired[ p.snd ] ) );

        diff = Math.abs( desired.x - tip.x ) + Math.abs( desired.y - tip.y );
        if ( !best || diff < best.diff ) {
            // Convert tip offset to coordinates inside the popup
            tip[ p.snd ] -= s.arHalf[ p.dimKey ] + result[ p.prop ] + s.contentBox[ p.snd ];
            best = { dir: dir, diff: diff, result: result, posProp: p.prop, posVal: tip[ p.snd ]
             };
        }

        return best;
    },

    _getPlacementState: function( clamp ) {
        var offset, gdOffset,
            ar = this._ui.arrow,
            state = {
                clampInfo: this._clampPopupWidth( !clamp ),
                arFull: { cx: ar.ct.width(), cy: ar.ct.height() },
                guideDims: { cx: ar.gd.width(), cy: ar.gd.height() },
                guideOffset: ar.gd.offset()
            };

        offset = this.element.offset();

        ar.gd.css( { left: 0, top: 0, right: 0, bottom: 0 } );
        gdOffset = ar.gd.offset();
        state.contentBox = {
            x: gdOffset.left - offset.left,
```

```
            y: gdOffset.top - offset.top,
            cx: ar.gd.width(),
            cy: ar.gd.height()
        };
        ar.gd.removeAttr( "style" );

        // The arrow box moves between guideOffset and guideOffset + guideDims - arFull
        state.guideOffset = { left: state.guideOffset.left - offset.left, top: state.guideOffset
        .top - offset.top };
        state.arHalf = { cx: state.arFull.cx / 2, cy: state.arFull.cy / 2 };
        state.menuHalf = { cx: state.clampInfo.menuSize.cx / 2, cy: state.clampInfo.menuSize.cy
        / 2 };

        return state;
    },

    _placementCoords: function( desired ) {
        var state, best, params, elOffset, bgRef,
            optionValue = this.options.arrow,
            ar = this._ui.arrow;

        if ( !ar ) {
            return this._super( desired );
        }

        ar.arEls.show();

        bgRef = {};
        state = this._getPlacementState( true );
        params = {
            "l": { fst: "x", snd: "y", prop: "top", dimKey: "cy", oDimKey: "cx", offsetFactor: 1
            , tipOffset:  -state.arHalf.cx, arrowOffsetFactor: 0 },
            "r": { fst: "x", snd: "y", prop: "top", dimKey: "cy", oDimKey: "cx", offsetFactor: -
            1, tipOffset: state.arHalf.cx + state.contentBox.cx, arrowOffsetFactor: 1 },
            "b": { fst: "y", snd: "x", prop: "left", dimKey: "cx", oDimKey: "cy", offsetFactor:
            -1, tipOffset: state.arHalf.cy + state.contentBox.cy, arrowOffsetFactor: 1 },
            "t": { fst: "y", snd: "x", prop: "left", dimKey: "cx", oDimKey: "cy", offsetFactor:
            1, tipOffset: -state.arHalf.cy, arrowOffsetFactor: 0 }
        };

        // Try each side specified in the options to see on which one the arrow
        // should be placed such that the distance between the tip of the arrow and
        // the desired coordinates is the shortest.
        $.each( ( optionValue === true ? "l,t,r,b" : optionValue ).split( "," ),
            $.proxy( function( key, value ) {
                best = this._tryAnArrow( params[ value ], value, desired, state, best );
            }, this ) );

        // Could not place the arrow along any of the edges - behave as if showing
        // the arrow was turned off.
        if ( !best ) {
            ar.arEls.hide();
            return this._super( desired );
        }
```

```
        // Move the arrow into place
        ar.ct
            .removeClass( "ui-popup-arrow-l ui-popup-arrow-t ui-popup-arrow-r ui-popup-arrow-b" )
            .addClass( "ui-popup-arrow-" + best.dir )
            .removeAttr( "style" ).css( best.posProp, best.posVal )
            .show();

        // Do not move/size the background div on IE, because we use the arrow div for
        background as well.
        if ( !ieHack ) {
            elOffset = this.element.offset();
            bgRef[ params[ best.dir ].fst ] = ar.ct.offset();
            bgRef[ params[ best.dir ].snd ] = {
                left: elOffset.left + state.contentBox.x,
                top: elOffset.top + state.contentBox.y
            };
        }

        return best.result;
    },

    _setOptions: function( opts ) {
        var newTheme,
            oldTheme = this.options.theme,
            ar = this._ui.arrow,
            ret = this._super( opts );

        if ( opts.arrow !== undefined ) {
            if ( !ar && opts.arrow ) {
                this._ui.arrow = this._addArrow();

                // Important to return here so we don't set the same options all over
                // again below.
                return;
            } else if ( ar && !opts.arrow ) {
                ar.arEls.remove();
                this._ui.arrow = null;
            }
        }

        // Reassign with potentially new arrow
        ar = this._ui.arrow;

        if ( ar ) {
            if ( opts.theme !== undefined ) {
                oldTheme = this._themeClassFromOption( "ui-body-", oldTheme );
                newTheme = this._themeClassFromOption( "ui-body-", opts.theme );
                ar.ar.removeClass( oldTheme ).addClass( newTheme );
            }

            if ( opts.shadow !== undefined ) {
                ar.ar.toggleClass( "ui-overlay-shadow", opts.shadow );
            }
```

```
        }

        return ret;
    },

    _destroy: function() {
        var ar = this._ui.arrow;

        if ( ar ) {
            ar.arEls.remove();
        }

        return this._super();
    }
});

})( jQuery );


(function( $, undefined ) {

$.widget( "mobile.panel", {
    options: {
        classes: {
            panel: "ui-panel",
            panelOpen: "ui-panel-open",
            panelClosed: "ui-panel-closed",
            panelFixed: "ui-panel-fixed",
            panelInner: "ui-panel-inner",
            modal: "ui-panel-dismiss",
            modalOpen: "ui-panel-dismiss-open",
            pageContainer: "ui-panel-page-container",
            pageWrapper: "ui-panel-wrapper",
            pageFixedToolbar: "ui-panel-fixed-toolbar",
            pageContentPrefix: "ui-panel-page-content", /* Used for wrapper and fixed toolbars
            position, display and open classes. */
            animate: "ui-panel-animate"
        },
        animate: true,
        theme: null,
        position: "left",
        dismissible: true,
        display: "reveal", //accepts reveal, push, overlay
        swipeClose: true,
        positionFixed: false
    },

    _closeLink: null,
    _parentPage: null,
    _page: null,
    _modal: null,
    _panelInner: null,
    _wrapper: null,
    _fixedToolbars: null,
```

```
    _create: function() {
        var el = this.element,
            parentPage = el.closest( ".ui-page, :jqmData(role='page')" );

        // expose some private props to other methods
        $.extend( this, {
            _closeLink: el.find( ":jqmData(rel='close')" ),
            _parentPage: ( parentPage.length > 0 ) ? parentPage : false,
            _openedPage: null,
            _page: this._getPage,
            _panelInner: this._getPanelInner(),
            _fixedToolbars: this._getFixedToolbars
        });
        if ( this.options.display !== "overlay" ){
            this._getWrapper();
        }
        this._addPanelClasses();

        // if animating, add the class to do so
        if ( $.support.cssTransform3d && !!this.options.animate ) {
            this.element.addClass( this.options.classes.animate );
        }

        this._bindUpdateLayout();
        this._bindCloseEvents();
        this._bindLinkListeners();
        this._bindPageEvents();

        if ( !!this.options.dismissible ) {
            this._createModal();
        }

        this._bindSwipeEvents();
    },

    _getPanelInner: function() {
        var panelInner = this.element.find( "." + this.options.classes.panelInner );

        if ( panelInner.length === 0 ) {
            panelInner = this.element.children().wrapAll( "<div class='" + this.options.classes.
            panelInner + "' />" ).parent();
        }

        return panelInner;
    },

    _createModal: function() {
        var self = this,
            target = self._parentPage ? self._parentPage.parent() : self.element.parent();

        self._modal = $( "<div class='" + self.options.classes.modal + "'></div>" )
            .on( "mousedown", function() {
                self.close();
```

```
            })
            .appendTo( target );
    },

    _getPage: function() {
        var page = this._openedPage || this._parentPage || $( "." + $.mobile.activePageClass );

        return page;
    },

    _getWrapper: function() {
        var wrapper = this._page().find( "." + this.options.classes.pageWrapper );
        if ( wrapper.length === 0 ) {
            wrapper = this._page().children( ".ui-header:not(.ui-header-fixed),
            .ui-content:not(.ui-popup), .ui-footer:not(.ui-footer-fixed)" )
                .wrapAll( "<div class='" + this.options.classes.pageWrapper + "'></div>" )
                .parent();
        }

        this._wrapper = wrapper;
    },

    _getFixedToolbars: function() {
        var extFixedToolbars = $( "body" ).children( ".ui-header-fixed, .ui-footer-fixed" ),
            intFixedToolbars = this._page().find( ".ui-header-fixed, .ui-footer-fixed" ),
            fixedToolbars = extFixedToolbars.add( intFixedToolbars ).addClass( this.options.
            classes.pageFixedToolbar );

        return fixedToolbars;
    },

    _getPosDisplayClasses: function( prefix ) {
        return prefix + "-position-" + this.options.position + " " + prefix + "-display-" + this
        .options.display;
    },

    _getPanelClasses: function() {
        var panelClasses = this.options.classes.panel +
            " " + this._getPosDisplayClasses( this.options.classes.panel ) +
            " " + this.options.classes.panelClosed +
            " " + "ui-body-" + ( this.options.theme ? this.options.theme : "inherit" );

        if ( !!this.options.positionFixed ) {
            panelClasses += " " + this.options.classes.panelFixed;
        }

        return panelClasses;
    },

    _addPanelClasses: function() {
        this.element.addClass( this._getPanelClasses() );
    },

    _handleCloseClickAndEatEvent: function( event ) {
```

```
        if ( !event.isDefaultPrevented() ) {
            event.preventDefault();
            this.close();
            return false;
        }
    },

    _handleCloseClick: function( event ) {
        if ( !event.isDefaultPrevented() ) {
            this.close();
        }
    },

    _bindCloseEvents: function() {
        this._on( this._closeLink, {
            "click": "_handleCloseClick"
        });

        this._on({
            "click a:jqmData(ajax='false')": "_handleCloseClick"
        });
    },

    _positionPanel: function( scrollToTop ) {
        var self = this,
            panelInnerHeight = self._panelInner.outerHeight(),
            expand = panelInnerHeight > $.mobile.getScreenHeight();

        if ( expand || !self.options.positionFixed ) {
            if ( expand ) {
                self._unfixPanel();
                $.mobile.resetActivePageHeight( panelInnerHeight );
            }
            if ( scrollToTop ) {
                this.window[ 0 ].scrollTo( 0, $.mobile.defaultHomeScroll );
            }
        } else {
            self._fixPanel();
        }
    },

    _bindFixListener: function() {
        this._on( $( window ), { "throttledresize": "_positionPanel" });
    },

    _unbindFixListener: function() {
        this._off( $( window ), "throttledresize" );
    },

    _unfixPanel: function() {
        if ( !!this.options.positionFixed && $.support.fixedPosition ) {
            this.element.removeClass( this.options.classes.panelFixed );
        }
    },
```

```
    _fixPanel: function() {
        if ( !!this.options.positionFixed && $.support.fixedPosition ) {
            this.element.addClass( this.options.classes.panelFixed );
        }
    },

    _bindUpdateLayout: function() {
        var self = this;

        self.element.on( "updatelayout", function(/* e */) {
            if ( self._open ) {
                self._positionPanel();
            }
        });
    },

    _bindLinkListeners: function() {
        this._on( "body", {
            "click a": "_handleClick"
        });

    },

    _handleClick: function( e ) {
        var link,
            panelId = this.element.attr( "id" );

        if ( e.currentTarget.href.split( "#" )[ 1 ] === panelId && panelId !== undefined ) {

            e.preventDefault();
            link = $( e.target );
            if ( link.hasClass( "ui-btn" ) ) {
                link.addClass( $.mobile.activeBtnClass );
                this.element.one( "panelopen panelclose", function() {
                    link.removeClass( $.mobile.activeBtnClass );
                });
            }
            this.toggle();
            return false;
        }
    },

    _bindSwipeEvents: function() {
        var self = this,
            area = self._modal ? self.element.add( self._modal ) : self.element;

        // on swipe, close the panel
        if ( !!self.options.swipeClose ) {
            if ( self.options.position === "left" ) {
                area.on( "swipeleft.panel", function(/* e */) {
                    self.close();
                });
            } else {
```

```
            area.on( "swiperight.panel", function(/* e */) {
                self.close();
            });
        }
    }
},

_bindPageEvents: function() {
    var self = this;

    this.document
        // Close the panel if another panel on the page opens
        .on( "panelbeforeopen", function( e ) {
            if ( self._open && e.target !== self.element[ 0 ] ) {
                self.close();
            }
        })
        // On escape, close? might need to have a target check too...
        .on( "keyup.panel", function( e ) {
            if ( e.keyCode === 27 && self._open ) {
                self.close();
            }
        });
    if ( !this._parentPage && this.options.display !== "overlay" ) {
        this._on( this.document, {
            "pageshow": "_getWrapper"
        });
    }
    // Clean up open panels after page hide
    if ( self._parentPage ) {
        this.document.on( "pagehide", ":jqmData(role='page')", function() {
            if ( self._open ) {
                self.close( true );
            }
        });
    } else {
        this.document.on( "pagebeforehide", function() {
            if ( self._open ) {
                self.close( true );
            }
        });
    }
},

// state storage of open or closed
_open: false,
_pageContentOpenClasses: null,
_modalOpenClasses: null,

open: function( immediate ) {
    if ( !this._open ) {
        var self = this,
            o = self.options,
```

```
            _openPanel = function() {
                self.document.off( "panelclose" );
                self._page().jqmData( "panel", "open" );

                if ( $.support.cssTransform3d && !!o.animate && o.display !== "overlay" ) {
                    self._wrapper.addClass( o.classes.animate );
                    self._fixedToolbars().addClass( o.classes.animate );
                }

                if ( !immediate && $.support.cssTransform3d && !!o.animate ) {
                    self.element.animationComplete( complete, "transition" );
                } else {
                    setTimeout( complete, 0 );
                }

                if ( o.theme && o.display !== "overlay" ) {
                    self._page().parent()
                        .addClass( o.classes.pageContainer + "-themed " + o.classes.
                        pageContainer + "-" + o.theme );
                }

                self.element
                    .removeClass( o.classes.panelClosed )
                    .addClass( o.classes.panelOpen );

                self._positionPanel( true );

                self._pageContentOpenClasses = self._getPosDisplayClasses( o.classes.
                pageContentPrefix );

                if ( o.display !== "overlay" ) {
                    self._page().parent().addClass( o.classes.pageContainer );
                    self._wrapper.addClass( self._pageContentOpenClasses );
                    self._fixedToolbars().addClass( self._pageContentOpenClasses );
                }

                self._modalOpenClasses = self._getPosDisplayClasses( o.classes.modal ) + " "
                 + o.classes.modalOpen;
                if ( self._modal ) {
                    self._modal
                        .addClass( self._modalOpenClasses )
                        .height( Math.max( self._modal.height(), self.document.height() ) );
                }
            },
            complete = function() {

                if ( o.display !== "overlay" ) {
                    self._wrapper.addClass( o.classes.pageContentPrefix + "-open" );
                    self._fixedToolbars().addClass( o.classes.pageContentPrefix + "-open" );
                }

                self._bindFixListener();

                self._trigger( "open" );
```

```
                self._openedPage = self._page();
            };

        self._trigger( "beforeopen" );

        if ( self._page().jqmData( "panel" ) === "open" ) {
            self.document.on( "panelclose", function() {
                _openPanel();
            });
        } else {
            _openPanel();
        }

        self._open = true;
    }
},

close: function( immediate ) {
    if ( this._open ) {
        var self = this,
            o = this.options,

            _closePanel = function() {

                self.element.removeClass( o.classes.panelOpen );

                if ( o.display !== "overlay" ) {
                    self._wrapper.removeClass( self._pageContentOpenClasses );
                    self._fixedToolbars().removeClass( self._pageContentOpenClasses );
                }

                if ( !immediate && $.support.cssTransform3d && !!o.animate ) {
                    self.element.animationComplete( complete, "transition" );
                } else {
                    setTimeout( complete, 0 );
                }

                if ( self._modal ) {
                    self._modal.removeClass( self._modalOpenClasses );
                }
            },
            complete = function() {
                if ( o.theme && o.display !== "overlay" ) {
                    self._page().parent().removeClass( o.classes.pageContainer + "-themed "
                    + o.classes.pageContainer + "-" + o.theme );
                }

                self.element.addClass( o.classes.panelClosed );

                if ( o.display !== "overlay" ) {
                    self._page().parent().removeClass( o.classes.pageContainer );
                    self._wrapper.removeClass( o.classes.pageContentPrefix + "-open" );
                    self._fixedToolbars().removeClass( o.classes.pageContentPrefix + "-open"
```

```
                    );
                }

                if ( $.support.cssTransform3d && !!o.animate && o.display !== "overlay" ) {
                    self._wrapper.removeClass( o.classes.animate );
                    self._fixedToolbars().removeClass( o.classes.animate );
                }

                self._fixPanel();
                self._unbindFixListener();
                $.mobile.resetActivePageHeight();

                self._page().jqmRemoveData( "panel" );

                self._trigger( "close" );

                self._openedPage = null;
            };

        self._trigger( "beforeclose" );

        _closePanel();

        self._open = false;
    }
},

toggle: function() {
    this[ this._open ? "close" : "open" ]();
},

_destroy: function() {
    var otherPanels,
    o = this.options,
    multiplePanels = ( $( "body > :mobile-panel" ).length + $.mobile.activePage.find(
    ":mobile-panel" ).length ) > 1;

    if ( o.display !== "overlay" ) {

        //  remove the wrapper if not in use by another panel
        otherPanels = $( "body > :mobile-panel" ).add( $.mobile.activePage.find(
        ":mobile-panel" ) );
        if ( otherPanels.not( ".ui-panel-display-overlay" ).not( this.element ).length === 0
         ) {
            this._wrapper.children().unwrap();
        }

        if ( this._open ) {

            this._fixedToolbars().removeClass( o.classes.pageContentPrefix + "-open" );

            if ( $.support.cssTransform3d && !!o.animate ) {
                this._fixedToolbars().removeClass( o.classes.animate );
            }
```

```
                this._page().parent().removeClass( o.classes.pageContainer );

                if ( o.theme ) {
                    this._page().parent().removeClass( o.classes.pageContainer + "-themed " + o.
                    classes.pageContainer + "-" + o.theme );
                }
            }
        }

        if ( !multiplePanels ) {

            this.document.off( "panelopen panelclose" );

        }

        if ( this._open ) {
            this._page().jqmRemoveData( "panel" );
        }

        this._panelInner.children().unwrap();

        this.element
            .removeClass( [ this._getPanelClasses(), o.classes.panelOpen, o.classes.animate ].
            join( " " ) )
            .off( "swipeleft.panel swiperight.panel" )
            .off( "panelbeforeopen" )
            .off( "panelhide" )
            .off( "keyup.panel" )
            .off( "updatelayout" );

        if ( this._modal ) {
            this._modal.remove();
        }
    }
});

})( jQuery );

(function( $, undefined ) {

$.widget( "mobile.table", {
    options: {
        classes: {
            table: "ui-table"
        },
        enhanced: false
    },

    _create: function() {
        if ( !this.options.enhanced ) {
            this.element.addClass( this.options.classes.table );
        }
```

```
        // extend here, assign on refresh > _setHeaders
        $.extend( this, {

            // Expose headers and allHeaders properties on the widget
            // headers references the THs within the first TR in the table
            headers: undefined,

            // allHeaders references headers, plus all THs in the thead, which may
            // include several rows, or not
            allHeaders: undefined
        });

        this._refresh( true );
    },

    _setHeaders: function() {
        var trs = this.element.find( "thead tr" );

        this.headers = this.element.find( "tr:eq(0)" ).children();
        this.allHeaders = this.headers.add( trs.children() );
    },

    refresh: function() {
        this._refresh();
    },

    rebuild: $.noop,

    _refresh: function( /* create */ ) {
        var table = this.element,
            trs = table.find( "thead tr" );

        // updating headers on refresh (fixes #5880)
        this._setHeaders();

        // Iterate over the trs
        trs.each( function() {
            var columnCount = 0;

            // Iterate over the children of the tr
            $( this ).children().each( function() {
                var span = parseInt( this.getAttribute( "colspan" ), 10 ),
                    selector = ":nth-child(" + ( columnCount + 1 ) + ")",
                    j;

                this.setAttribute( "data-" + $.mobile.ns + "colstart", columnCount + 1 );

                if ( span ) {
                    for( j = 0; j < span - 1; j++ ) {
                        columnCount++;
                        selector += ", :nth-child(" + ( columnCount + 1 ) + ")";
                    }
                }
```

```
                // Store "cells" data on header as a reference to all cells in the
                // same column as this TH
                $( this ).jqmData( "cells", table.find( "tr" ).not( trs.eq( 0 ) ).not( this ).
                children( selector ) );

                columnCount++;
            });
        });
    }
});

})( jQuery );


(function( $, undefined ) {

$.widget( "mobile.table", $.mobile.table, {
    options: {
        mode: "columntoggle",
        columnBtnTheme: null,
        columnPopupTheme: null,
        columnBtnText: "Columns...",
        classes: $.extend( $.mobile.table.prototype.options.classes, {
            popup: "ui-table-columntoggle-popup",
            columnBtn: "ui-table-columntoggle-btn",
            priorityPrefix: "ui-table-priority-",
            columnToggleTable: "ui-table-columntoggle"
        })
    },

    _create: function() {
        this._super();

        if ( this.options.mode !== "columntoggle" ) {
            return;
        }

        $.extend( this, {
            _menu: null
        });

        if ( this.options.enhanced ) {
            this._menu = $( this.document[ 0 ].getElementById( this._id() + "-popup" ) ).
            children().first();
            this._addToggles( this._menu, true );
        } else {
            this._menu = this._enhanceColToggle();
            this.element.addClass( this.options.classes.columnToggleTable );
        }

        this._setupEvents();

        this._setToggleState();
    },
```

```
	_id: function() {
		return ( this.element.attr( "id" ) || ( this.widgetName + this.uuid ) );
	},

	_setupEvents: function() {
		//NOTE: inputs are bound in bindToggles,
		// so it can be called on refresh, too

		// update column toggles on resize
		this._on( this.window, {
			throttledresize: "_setToggleState"
		});
		this._on( this._menu, {
			"change input": "_menuInputChange"
		});
	},

	_addToggles: function( menu, keep ) {
		var inputs,
			checkboxIndex = 0,
			opts = this.options,
			container = menu.controlgroup( "container" );

		// allow update of menu on refresh (fixes #5880)
		if ( keep ) {
			inputs = menu.find( "input" );
		} else {
			container.empty();
		}

		// create the hide/show toggles
		this.headers.not( "td" ).each( function() {
			var header = $( this ),
				priority = $.mobile.getAttribute( this, "priority" ),
				cells = header.add( header.jqmData( "cells" ) );

			if ( priority ) {
				cells.addClass( opts.classes.priorityPrefix + priority );

				( keep ? inputs.eq( checkboxIndex++ ) :
					$("<label><input type='checkbox' checked />" +
						( header.children( "abbr" ).first().attr( "title" ) ||
							header.text() ) +
						"</label>" )
						.appendTo( container )
						.children( 0 )
						.checkboxradio( {
							theme: opts.columnPopupTheme
						}) )
					.jqmData( "cells", cells );
			}
		});
```

```
        // set bindings here
        if ( !keep ) {
            menu.controlgroup( "refresh" );
        }
    },

    _menuInputChange: function( evt ) {
        var input = $( evt.target ),
            checked = input[ 0 ].checked;

        input.jqmData( "cells" )
            .toggleClass( "ui-table-cell-hidden", !checked )
            .toggleClass( "ui-table-cell-visible", checked );

        if ( input[ 0 ].getAttribute( "locked" ) ) {
            input.removeAttr( "locked" );

            this._unlockCells( input.jqmData( "cells" ) );
        } else {
            input.attr( "locked", true );
        }
    },

    _unlockCells: function( cells ) {
        // allow hide/show via CSS only = remove all toggle-locks
        cells.removeClass( "ui-table-cell-hidden ui-table-cell-visible");
    },

    _enhanceColToggle: function() {
        var id , menuButton, popup, menu,
            table = this.element,
            opts = this.options,
            ns = $.mobile.ns,
            fragment = this.document[ 0 ].createDocumentFragment();

        id = this._id() + "-popup";
        menuButton = $( "<a href='#" + id + "' " +
            "class='" + opts.classes.columnBtn + " ui-btn " +
            "ui-btn-" + ( opts.columnBtnTheme || "a" ) +
            " ui-corner-all ui-shadow ui-mini' " +
            "data-" + ns + "rel='popup'>" + opts.columnBtnText + "</a>" );
        popup = $( "<div class='" + opts.classes.popup + "' id='" + id + "'></div>" );
        menu = $( "<fieldset></fieldset>" ).controlgroup();

        // set extension here, send "false" to trigger build/rebuild
        this._addToggles( menu, false );

        menu.appendTo( popup );

        fragment.appendChild( popup[ 0 ] );
        fragment.appendChild( menuButton[ 0 ] );
        table.before( fragment );

        popup.popup();
```

```
        return menu;
    },

    rebuild: function() {
        this._super();

        if ( this.options.mode === "columntoggle" ) {
            // NOTE: rebuild passes "false", while refresh passes "undefined"
            // both refresh the table, but inside addToggles, !false will be true,
            // so a rebuild call can be indentified
            this._refresh( false );
        }
    },

    _refresh: function( create ) {
        this._super( create );

        if ( !create && this.options.mode === "columntoggle" ) {
            // columns not being replaced must be cleared from input toggle-locks
            this._unlockCells( this.element.find( ".ui-table-cell-hidden, " +
                ".ui-table-cell-visible" ) );

            // update columntoggles and cells
            this._addToggles( this._menu, create );

            // check/uncheck
            this._setToggleState();
        }
    },

    _setToggleState: function() {
        this._menu.find( "input" ).each( function() {
            var checkbox = $( this );

            this.checked = checkbox.jqmData( "cells" ).eq( 0 ).css( "display" ) === "table-cell";
            checkbox.checkboxradio( "refresh" );
        });
    },

    _destroy: function() {
        this._super();
    }
});

})( jQuery );

(function( $, undefined ) {

$.widget( "mobile.table", $.mobile.table, {
    options: {
        mode: "reflow",
        classes: $.extend( $.mobile.table.prototype.options.classes, {
            reflowTable: "ui-table-reflow",
```

```
            cellLabels: "ui-table-cell-label"
        })
    },

    _create: function() {
        this._super();

        // If it's not reflow mode, return here.
        if ( this.options.mode !== "reflow" ) {
            return;
        }

        if ( !this.options.enhanced ) {
            this.element.addClass( this.options.classes.reflowTable );

            this._updateReflow();
        }
    },

    rebuild: function() {
        this._super();

        if ( this.options.mode === "reflow" ) {
            this._refresh( false );
        }
    },

    _refresh: function( create ) {
        this._super( create );
        if ( !create && this.options.mode === "reflow" ) {
            this._updateReflow( );
        }
    },

    _updateReflow: function() {
        var table = this,
            opts = this.options;

        // get headers in reverse order so that top-level headers are appended last
        $( table.allHeaders.get().reverse() ).each( function() {
            var cells = $( this ).jqmData( "cells" ),
                colstart = $.mobile.getAttribute( this, "colstart" ),
                hierarchyClass = cells.not( this ).filter( "thead th" ).length && "
                ui-table-cell-label-top",
                text = $( this ).text(),
                iteration, filter;

                if ( text !== ""  ) {

                    if ( hierarchyClass ) {
                        iteration = parseInt( this.getAttribute( "colspan" ), 10 );
                        filter = "";

                        if ( iteration ) {
```

```
                            filter = "td:nth-child("+ iteration +"n + " + ( colstart ) +")";
                        }

                        table._addLabels( cells.filter( filter ), opts.classes.cellLabels +
                        hierarchyClass, text );
                    } else {
                        table._addLabels( cells, opts.classes.cellLabels, text );
                    }

                }
        });
    },

    _addLabels: function( cells, label, text ) {
        // .not fixes #6006
        cells.not( ":has(b." + label + ")" ).prepend( "<b class='" + label + "'>" + text +
        "</b>"  );
    }
});

})( jQuery );

(function( $, undefined ) {

// TODO rename filterCallback/deprecate and default to the item itself as the first argument
var defaultFilterCallback = function( index, searchValue ) {
    return ( ( "" + ( $.mobile.getAttribute( this, "filtertext" ) || $( this ).text() ) )
        .toLowerCase().indexOf( searchValue ) === -1 );
};

$.widget( "mobile.filterable", {

    initSelector: ":jqmData(filter='true')",

    options: {
        filterReveal: false,
        filterCallback: defaultFilterCallback,
        enhanced: false,
        input: null,
        children: "> li, > option, > optgroup option, > tbody tr, > .ui-controlgroup-controls >
        .ui-btn, > .ui-controlgroup-controls > .ui-checkbox, > .ui-controlgroup-controls >
        .ui-radio"
    },

    _create: function() {
        var opts = this.options;

        $.extend( this, {
            _search: null,
            _timer: 0
        });

        this._setInput( opts.input );
        if ( !opts.enhanced ) {
```

```
            this._filterItems( ( ( this._search && this._search.val() ) || "" ).toLowerCase() );
        }
    },

    _onKeyUp: function() {
        var val, lastval,
            search = this._search;

        if ( search ) {
            val = search.val().toLowerCase(),
            lastval = $.mobile.getAttribute( search[ 0 ], "lastval" ) + "";

            if ( lastval && lastval === val ) {
                // Execute the handler only once per value change
                return;
            }

            if ( this._timer ) {
                window.clearTimeout( this._timer );
                this._timer = 0;
            }

            this._timer = this._delay( function() {
                this._trigger( "beforefilter", null, { input: search } );

                // Change val as lastval for next execution
                search[ 0 ].setAttribute( "data-" + $.mobile.ns + "lastval", val );

                this._filterItems( val );
                this._timer = 0;
            }, 250 );
        }
    },

    _getFilterableItems: function() {
        var elem = this.element,
            children = this.options.children,
            items = !children ? { length: 0 }:
                $.isFunction( children ) ? children():
                children.nodeName ? $( children ):
                children.jquery ? children:
                this.element.find( children );

        if ( items.length === 0 ) {
            items = elem.children();
        }

        return items;
    },

    _filterItems: function( val ) {
        var idx, callback, length, dst,
            show = [],
            hide = [],
```

```
            opts = this.options,
            filterItems = this._getFilterableItems();

        if ( val != null ) {
            callback = opts.filterCallback || defaultFilterCallback;
            length = filterItems.length;

            // Partition the items into those to be hidden and those to be shown
            for ( idx = 0 ; idx < length ; idx++ ) {
                dst = ( callback.call( filterItems[ idx ], idx, val ) ) ? hide : show;
                dst.push( filterItems[ idx ] );
            }
        }

        // If nothing is hidden, then the decision whether to hide or show the items
        // is based on the "filterReveal" option.
        if ( hide.length === 0 ) {
            filterItems[ opts.filterReveal ? "addClass" : "removeClass" ]( "ui-screen-hidden" );
        } else {
            $( hide ).addClass( "ui-screen-hidden" );
            $( show ).removeClass( "ui-screen-hidden" );
        }

        this._refreshChildWidget();

        this._trigger( "filter", null, {
            items: filterItems
        });
    },

    // The Default implementation of _refreshChildWidget attempts to call
    // refresh on collapsibleset, controlgroup, selectmenu, or listview
    _refreshChildWidget: function() {
        var widget, idx,
            recognizedWidgets = [ "collapsibleset", "selectmenu", "controlgroup", "listview" ];

        for ( idx = recognizedWidgets.length - 1 ; idx > -1 ; idx-- ) {
            widget = recognizedWidgets[ idx ];
            if ( $.mobile[ widget ] ) {
                widget = this.element.data( "mobile-" + widget );
                if ( widget && $.isFunction( widget.refresh ) ) {
                    widget.refresh();
                }
            }
        }
    },

    // TODO: When the input is not internal, do not even store it in this._search
    _setInput: function ( selector ) {
        var search = this._search;

        // Stop a pending filter operation
        if ( this._timer ) {
            window.clearTimeout( this._timer );
```

```
            this._timer = 0;
        }

        if ( search ) {
            this._off( search, "keyup change input" );
            search = null;
        }

        if ( selector ) {
            search = selector.jquery ? selector:
                selector.nodeName ? $( selector ):
                this.document.find( selector );

            this._on( search, {
                keyup: "_onKeyUp",
                change: "_onKeyUp",
                input: "_onKeyUp"
            });
        }

        this._search = search;
    },

    _setOptions: function( options ) {
        var refilter = !( ( options.filterReveal === undefined ) &&
                ( options.filterCallback === undefined ) &&
                ( options.children === undefined ) );

        this._super( options );

        if ( options.input !== undefined ) {
            this._setInput( options.input );
            refilter = true;
        }

        if ( refilter ) {
            this.refresh();
        }
    },

    _destroy: function() {
        var opts = this.options,
            items = this._getFilterableItems();

        if ( opts.enhanced ) {
            items.toggleClass( "ui-screen-hidden", opts.filterReveal );
        } else {
            items.removeClass( "ui-screen-hidden" );
        }
    },

    refresh: function() {
        if ( this._timer ) {
            window.clearTimeout( this._timer );
```

```
            this._timer = 0;
        }
        this._filterItems( ( ( this._search && this._search.val() ) || "" ).toLowerCase() );
    }
});

})( jQuery );

(function( $, undefined ) {

// Create a function that will replace the _setOptions function of a widget,
// and will pass the options on to the input of the filterable.
var replaceSetOptions = function( self, orig ) {
        return function( options ) {
            orig.call( this, options );
            self._syncTextInputOptions( options );
        };
    },
    rDividerListItem = /(^|\s)ui-li-divider(\s|$)/,
    origDefaultFilterCallback = $.mobile.filterable.prototype.options.filterCallback;

// Override the default filter callback with one that does not hide list dividers
$.mobile.filterable.prototype.options.filterCallback = function( index, searchValue ) {
    return !this.className.match( rDividerListItem ) &&
        origDefaultFilterCallback.call( this, index, searchValue );
};

$.widget( "mobile.filterable", $.mobile.filterable, {
    options: {
        filterPlaceholder: "Filter items...",
        filterTheme: null
    },

    _create: function() {
        var idx, widgetName,
            elem = this.element,
            recognizedWidgets = [ "collapsibleset", "selectmenu", "controlgroup", "listview" ],
            createHandlers = {};

        this._super();

        $.extend( this, {
            _widget: null
        });

        for ( idx = recognizedWidgets.length - 1 ; idx > -1 ; idx-- ) {
            widgetName = recognizedWidgets[ idx ];
            if ( $.mobile[ widgetName ] ) {
                if ( this._setWidget( elem.data( "mobile-" + widgetName ) ) ) {
                    break;
                } else {
                    createHandlers[ widgetName + "create" ] = "_handleCreate";
                }
            }
```

```
        }

        if ( !this._widget ) {
            this._on( elem, createHandlers );
        }
    },

    _handleCreate: function( evt ) {
        this._setWidget( this.element.data( "mobile-" + evt.type.substring( 0, evt.type.length -
         6 ) ) );
    },

    _trigger: function( type, event, data ) {
        if ( this._widget && this._widget.widgetFullName === "mobile-listview" &&
            type === "beforefilter" ) {

            // Also trigger listviewbeforefilter if this widget is also a listview
            this._widget._trigger( "beforefilter", event, data );
        }
        this._super( type, event, data );
    },

    _setWidget: function( widget ) {
        if ( !this._widget && widget ) {
            this._widget = widget;
            this._widget._setOptions = replaceSetOptions( this, this._widget._setOptions );
        }

        if ( !!this._widget ) {
            this._syncTextInputOptions( this._widget.options );
            if ( this._widget.widgetName === "listview" ) {
                this._widget.options.hideDividers = true;
                this._widget.element.listview( "refresh" );
            }
        }

        return !!this._widget;
    },

    _isSearchInternal: function() {
        return ( this._search && this._search.jqmData( "ui-filterable-" + this.uuid +
        "-internal" ) );
    },

    _setInput: function( selector ) {
        var opts = this.options,
            updatePlaceholder = true,
            textinputOpts = {};

        if ( !selector ) {
            if ( this._isSearchInternal() ) {

                // Ignore the call to set a new input if the selector goes to falsy and
                // the current textinput is already of the internally generated variety.
```

```
                return;
            } else {

                // Generating a new textinput widget. No need to set the placeholder
                // further down the function.
                updatePlaceholder = false;
                selector = $( "<input " +
                    "data-" + $.mobile.ns + "type='search' " +
                    "placeholder='" + opts.filterPlaceholder + "'></input>" )
                    .jqmData( "ui-filterable-" + this.uuid + "-internal", true );
                $( "<form class='ui-filterable'></form>" )
                    .append( selector )
                    .submit( function( evt ) {
                        evt.preventDefault();
                        selector.blur();
                    })
                    .insertBefore( this.element );
                if ( $.mobile.textinput ) {
                    if ( this.options.filterTheme != null ) {
                        textinputOpts[ "theme" ] = opts.filterTheme;
                    }

                    selector.textinput( textinputOpts );
                }
            }
        }

        this._super( selector );

        if ( this._isSearchInternal() && updatePlaceholder ) {
            this._search.attr( "placeholder", this.options.filterPlaceholder );
        }
    },

    _setOptions: function( options ) {
        var ret = this._super( options );

        // Need to set the filterPlaceholder after having established the search input
        if ( options.filterPlaceholder !== undefined ) {
            if ( this._isSearchInternal() ) {
                this._search.attr( "placeholder", options.filterPlaceholder );
            }
        }

        if ( options.filterTheme !== undefined && this._search && $.mobile.textinput ) {
            this._search.textinput( "option", "theme", options.filterTheme );
        }

        return ret;
    },

    _destroy: function() {
        if ( this._isSearchInternal() ) {
            this._search.remove();
```

```
        }
        this._super();
    },

    _syncTextInputOptions: function( options ) {
        var idx,
            textinputOptions = {};

        // We only sync options if the filterable's textinput is of the internally
        // generated variety, rather than one specified by the user.
        if ( this._isSearchInternal() && $.mobile.textinput ) {

            // Apply only the options understood by textinput
            for ( idx in $.mobile.textinput.prototype.options ) {
                if ( options[ idx ] !== undefined ) {
                    if ( idx === "theme" && this.options.filterTheme != null ) {
                        textinputOptions[ idx ] = this.options.filterTheme;
                    } else {
                        textinputOptions[ idx ] = options[ idx ];
                    }
                }
            }
            this._search.textinput( "option", textinputOptions );
        }
    }
});

})( jQuery );

/*!
 * jQuery UI Tabs fadf2b312a05040436451c64bbfaf4814bc62c56
 * http://jqueryui.com
 *
 * Copyright 2013 jQuery Foundation and other contributors
 * Released under the MIT license.
 * http://jquery.org/license
 *
 * http://api.jqueryui.com/tabs/
 *
 * Depends:
 *  jquery.ui.core.js
 *  jquery.ui.widget.js
 */
(function( $, undefined ) {

var tabId = 0,
    rhash = /#.*$/;

function getNextTabId() {
    return ++tabId;
}

function isLocal( anchor ) {
    return anchor.hash.length > 1 &&
```

```
        decodeURIComponent( anchor.href.replace( rhash, "" ) ) ===
            decodeURIComponent( location.href.replace( rhash, "" ) );
}

$.widget( "ui.tabs", {
    version: "fadf2b312a05040436451c64bbfaf4814bc62c56",
    delay: 300,
    options: {
        active: null,
        collapsible: false,
        event: "click",
        heightStyle: "content",
        hide: null,
        show: null,

        // callbacks
        activate: null,
        beforeActivate: null,
        beforeLoad: null,
        load: null
    },

    _create: function() {
        var that = this,
            options = this.options;

        this.running = false;

        this.element
            .addClass( "ui-tabs ui-widget ui-widget-content ui-corner-all" )
            .toggleClass( "ui-tabs-collapsible", options.collapsible )
            // Prevent users from focusing disabled tabs via click
            .delegate( ".ui-tabs-nav > li", "mousedown" + this.eventNamespace, function( event ) {
                if ( $( this ).is( ".ui-state-disabled" ) ) {
                    event.preventDefault();
                }
            })
            // support: IE <9
            // Preventing the default action in mousedown doesn't prevent IE
            // from focusing the element, so if the anchor gets focused, blur.
            // We don't have to worry about focusing the previously focused
            // element since clicking on a non-focusable element should focus
            // the body anyway.
            .delegate( ".ui-tabs-anchor", "focus" + this.eventNamespace, function() {
                if ( $( this ).closest( "li" ).is( ".ui-state-disabled" ) ) {
                    this.blur();
                }
            });

        this._processTabs();
        options.active = this._initialActive();

        // Take disabling tabs via class attribute from HTML
```

```
        // into account and update option properly.
        if ( $.isArray( options.disabled ) ) {
            options.disabled = $.unique( options.disabled.concat(
                $.map( this.tabs.filter( ".ui-state-disabled" ), function( li ) {
                    return that.tabs.index( li );
                })
            ) ).sort();
        }

        // check for length avoids error when initializing empty list
        if ( this.options.active !== false && this.anchors.length ) {
            this.active = this._findActive( options.active );
        } else {
            this.active = $();
        }

        this._refresh();

        if ( this.active.length ) {
            this.load( options.active );
        }
    },

    _initialActive: function() {
        var active = this.options.active,
            collapsible = this.options.collapsible,
            locationHash = location.hash.substring( 1 );

        if ( active === null ) {
            // check the fragment identifier in the URL
            if ( locationHash ) {
                this.tabs.each(function( i, tab ) {
                    if ( $( tab ).attr( "aria-controls" ) === locationHash ) {
                        active = i;
                        return false;
                    }
                });
            }

            // check for a tab marked active via a class
            if ( active === null ) {
                active = this.tabs.index( this.tabs.filter( ".ui-tabs-active" ) );
            }

            // no active tab, set to false
            if ( active === null || active === -1 ) {
                active = this.tabs.length ? 0 : false;
            }
        }

        // handle numbers: negative, out of range
        if ( active !== false ) {
            active = this.tabs.index( this.tabs.eq( active ) );
            if ( active === -1 ) {
```

```
                active = collapsible ? false : 0;
            }
        }

        // don't allow collapsible: false and active: false
        if ( !collapsible && active === false && this.anchors.length ) {
            active = 0;
        }

        return active;
    },

    _getCreateEventData: function() {
        return {
            tab: this.active,
            panel: !this.active.length ? $() : this._getPanelForTab( this.active )
        };
    },

    _tabKeydown: function( event ) {
        var focusedTab = $( this.document[0].activeElement ).closest( "li" ),
            selectedIndex = this.tabs.index( focusedTab ),
            goingForward = true;

        if ( this._handlePageNav( event ) ) {
            return;
        }

        switch ( event.keyCode ) {
            case $.ui.keyCode.RIGHT:
            case $.ui.keyCode.DOWN:
                selectedIndex++;
                break;
            case $.ui.keyCode.UP:
            case $.ui.keyCode.LEFT:
                goingForward = false;
                selectedIndex--;
                break;
            case $.ui.keyCode.END:
                selectedIndex = this.anchors.length - 1;
                break;
            case $.ui.keyCode.HOME:
                selectedIndex = 0;
                break;
            case $.ui.keyCode.SPACE:
                // Activate only, no collapsing
                event.preventDefault();
                clearTimeout( this.activating );
                this._activate( selectedIndex );
                return;
            case $.ui.keyCode.ENTER:
                // Toggle (cancel delayed activation, allow collapsing)
                event.preventDefault();
                clearTimeout( this.activating );
```

```
                // Determine if we should collapse or activate
                this._activate( selectedIndex === this.options.active ? false : selectedIndex );
                return;
            default:
                return;
        }

        // Focus the appropriate tab, based on which key was pressed
        event.preventDefault();
        clearTimeout( this.activating );
        selectedIndex = this._focusNextTab( selectedIndex, goingForward );

        // Navigating with control key will prevent automatic activation
        if ( !event.ctrlKey ) {
            // Update aria-selected immediately so that AT think the tab is already selected.
            // Otherwise AT may confuse the user by stating that they need to activate the tab,
            // but the tab will already be activated by the time the announcement finishes.
            focusedTab.attr( "aria-selected", "false" );
            this.tabs.eq( selectedIndex ).attr( "aria-selected", "true" );

            this.activating = this._delay(function() {
                this.option( "active", selectedIndex );
            }, this.delay );
        }
    },

    _panelKeydown: function( event ) {
        if ( this._handlePageNav( event ) ) {
            return;
        }

        // Ctrl+up moves focus to the current tab
        if ( event.ctrlKey && event.keyCode === $.ui.keyCode.UP ) {
            event.preventDefault();
            this.active.focus();
        }
    },

    // Alt+page up/down moves focus to the previous/next tab (and activates)
    _handlePageNav: function( event ) {
        if ( event.altKey && event.keyCode === $.ui.keyCode.PAGE_UP ) {
            this._activate( this._focusNextTab( this.options.active - 1, false ) );
            return true;
        }
        if ( event.altKey && event.keyCode === $.ui.keyCode.PAGE_DOWN ) {
            this._activate( this._focusNextTab( this.options.active + 1, true ) );
            return true;
        }
    },

    _findNextTab: function( index, goingForward ) {
        var lastTabIndex = this.tabs.length - 1;

        function constrain() {
```

```
            if ( index > lastTabIndex ) {
                index = 0;
            }
            if ( index < 0 ) {
                index = lastTabIndex;
            }
            return index;
        }

        while ( $.inArray( constrain(), this.options.disabled ) !== -1 ) {
            index = goingForward ? index + 1 : index - 1;
        }

        return index;
    },

    _focusNextTab: function( index, goingForward ) {
        index = this._findNextTab( index, goingForward );
        this.tabs.eq( index ).focus();
        return index;
    },

    _setOption: function( key, value ) {
        if ( key === "active" ) {
            // _activate() will handle invalid values and update this.options
            this._activate( value );
            return;
        }

        if ( key === "disabled" ) {
            // don't use the widget factory's disabled handling
            this._setupDisabled( value );
            return;
        }

        this._super( key, value);

        if ( key === "collapsible" ) {
            this.element.toggleClass( "ui-tabs-collapsible", value );
            // Setting collapsible: false while collapsed; open first panel
            if ( !value && this.options.active === false ) {
                this._activate( 0 );
            }
        }

        if ( key === "event" ) {
            this._setupEvents( value );
        }

        if ( key === "heightStyle" ) {
            this._setupHeightStyle( value );
        }
    },
```

```
    _tabId: function( tab ) {
        return tab.attr( "aria-controls" ) || "ui-tabs-" + getNextTabId();
    },

    _sanitizeSelector: function( hash ) {
        return hash ? hash.replace( /[!"$%&'()*+,.\/:;<=>?@\[\]\^`{|}~]/g, "\\$&" ) : "";
    },

    refresh: function() {
        var options = this.options,
            lis = this.tablist.children( ":has(a[href])" );

        // get disabled tabs from class attribute from HTML
        // this will get converted to a boolean if needed in _refresh()
        options.disabled = $.map( lis.filter( ".ui-state-disabled" ), function( tab ) {
            return lis.index( tab );
        });

        this._processTabs();

        // was collapsed or no tabs
        if ( options.active === false || !this.anchors.length ) {
            options.active = false;
            this.active = $();
        // was active, but active tab is gone
        } else if ( this.active.length && !$.contains( this.tablist[ 0 ], this.active[ 0 ] ) ) {
            // all remaining tabs are disabled
            if ( this.tabs.length === options.disabled.length ) {
                options.active = false;
                this.active = $();
            // activate previous tab
            } else {
                this._activate( this._findNextTab( Math.max( 0, options.active - 1 ), false ) );
            }
        // was active, active tab still exists
        } else {
            // make sure active index is correct
            options.active = this.tabs.index( this.active );
        }

        this._refresh();
    },

    _refresh: function() {
        this._setupDisabled( this.options.disabled );
        this._setupEvents( this.options.event );
        this._setupHeightStyle( this.options.heightStyle );

        this.tabs.not( this.active ).attr({
            "aria-selected": "false",
            tabIndex: -1
        });
        this.panels.not( this._getPanelForTab( this.active ) )
            .hide()
```

```
            .attr({
                "aria-expanded": "false",
                "aria-hidden": "true"
            });

        // Make sure one tab is in the tab order
        if ( !this.active.length ) {
            this.tabs.eq( 0 ).attr( "tabIndex", 0 );
        } else {
            this.active
                .addClass( "ui-tabs-active ui-state-active" )
                .attr({
                    "aria-selected": "true",
                    tabIndex: 0
                });
            this._getPanelForTab( this.active )
                .show()
                .attr({
                    "aria-expanded": "true",
                    "aria-hidden": "false"
                });
        }
    },

    _processTabs: function() {
        var that = this;

        this.tablist = this._getList()
            .addClass( "ui-tabs-nav ui-helper-reset ui-helper-clearfix ui-widget-header
            ui-corner-all" )
            .attr( "role", "tablist" );

        this.tabs = this.tablist.find( "> li:has(a[href])" )
            .addClass( "ui-state-default ui-corner-top" )
            .attr({
                role: "tab",
                tabIndex: -1
            });

        this.anchors = this.tabs.map(function() {
                return $( "a", this )[ 0 ];
            })
            .addClass( "ui-tabs-anchor" )
            .attr({
                role: "presentation",
                tabIndex: -1
            });

        this.panels = $();

        this.anchors.each(function( i, anchor ) {
            var selector, panel, panelId,
                anchorId = $( anchor ).uniqueId().attr( "id" ),
                tab = $( anchor ).closest( "li" ),
```

```
                originalAriaControls = tab.attr( "aria-controls" );

            // inline tab
            if ( isLocal( anchor ) ) {
                selector = anchor.hash;
                panel = that.element.find( that._sanitizeSelector( selector ) );
            // remote tab
            } else {
                panelId = that._tabId( tab );
                selector = "#" + panelId;
                panel = that.element.find( selector );
                if ( !panel.length ) {
                    panel = that._createPanel( panelId );
                    panel.insertAfter( that.panels[ i - 1 ] || that.tablist );
                }
                panel.attr( "aria-live", "polite" );
            }

            if ( panel.length) {
                that.panels = that.panels.add( panel );
            }
            if ( originalAriaControls ) {
                tab.data( "ui-tabs-aria-controls", originalAriaControls );
            }
            tab.attr({
                "aria-controls": selector.substring( 1 ),
                "aria-labelledby": anchorId
            });
            panel.attr( "aria-labelledby", anchorId );
        });

        this.panels
            .addClass( "ui-tabs-panel ui-widget-content ui-corner-bottom" )
            .attr( "role", "tabpanel" );
    },

    // allow overriding how to find the list for rare usage scenarios (#7715)
    _getList: function() {
        return this.element.find( "ol,ul" ).eq( 0 );
    },

    _createPanel: function( id ) {
        return $( "<div>" )
            .attr( "id", id )
            .addClass( "ui-tabs-panel ui-widget-content ui-corner-bottom" )
            .data( "ui-tabs-destroy", true );
    },

    _setupDisabled: function( disabled ) {
        if ( $.isArray( disabled ) ) {
            if ( !disabled.length ) {
                disabled = false;
            } else if ( disabled.length === this.anchors.length ) {
                disabled = true;
```

```
            }
        }

        // disable tabs
        for ( var i = 0, li; ( li = this.tabs[ i ] ); i++ ) {
            if ( disabled === true || $.inArray( i, disabled ) !== -1 ) {
                $( li )
                    .addClass( "ui-state-disabled" )
                    .attr( "aria-disabled", "true" );
            } else {
                $( li )
                    .removeClass( "ui-state-disabled" )
                    .removeAttr( "aria-disabled" );
            }
        }

        this.options.disabled = disabled;
    },

    _setupEvents: function( event ) {
        var events = {
            click: function( event ) {
                event.preventDefault();
            }
        };
        if ( event ) {
            $.each( event.split(" "), function( index, eventName ) {
                events[ eventName ] = "_eventHandler";
            });
        }

        this._off( this.anchors.add( this.tabs ).add( this.panels ) );
        this._on( this.anchors, events );
        this._on( this.tabs, { keydown: "_tabKeydown" } );
        this._on( this.panels, { keydown: "_panelKeydown" } );

        this._focusable( this.tabs );
        this._hoverable( this.tabs );
    },

    _setupHeightStyle: function( heightStyle ) {
        var maxHeight,
            parent = this.element.parent();

        if ( heightStyle === "fill" ) {
            maxHeight = parent.height();
            maxHeight -= this.element.outerHeight() - this.element.height();

            this.element.siblings( ":visible" ).each(function() {
                var elem = $( this ),
                    position = elem.css( "position" );

                if ( position === "absolute" || position === "fixed" ) {
                    return;
```

```
            }
            maxHeight -= elem.outerHeight( true );
        });

        this.element.children().not( this.panels ).each(function() {
            maxHeight -= $( this ).outerHeight( true );
        });

        this.panels.each(function() {
            $( this ).height( Math.max( 0, maxHeight -
                $( this ).innerHeight() + $( this ).height() ) );
        })
        .css( "overflow", "auto" );
    } else if ( heightStyle === "auto" ) {
        maxHeight = 0;
        this.panels.each(function() {
            maxHeight = Math.max( maxHeight, $( this ).height( "" ).height() );
        }).height( maxHeight );
    }
},

_eventHandler: function( event ) {
    var options = this.options,
        active = this.active,
        anchor = $( event.currentTarget ),
        tab = anchor.closest( "li" ),
        clickedIsActive = tab[ 0 ] === active[ 0 ],
        collapsing = clickedIsActive && options.collapsible,
        toShow = collapsing ? $() : this._getPanelForTab( tab ),
        toHide = !active.length ? $() : this._getPanelForTab( active ),
        eventData = {
            oldTab: active,
            oldPanel: toHide,
            newTab: collapsing ? $() : tab,
            newPanel: toShow
        };

    event.preventDefault();

    if ( tab.hasClass( "ui-state-disabled" ) ||
            // tab is already loading
            tab.hasClass( "ui-tabs-loading" ) ||
            // can't switch durning an animation
            this.running ||
            // click on active header, but not collapsible
            ( clickedIsActive && !options.collapsible ) ||
            // allow canceling activation
            ( this._trigger( "beforeActivate", event, eventData ) === false ) ) {
        return;
    }

    options.active = collapsing ? false : this.tabs.index( tab );

    this.active = clickedIsActive ? $() : tab;
```

```
        if ( this.xhr ) {
            this.xhr.abort();
        }

        if ( !toHide.length && !toShow.length ) {
            $.error( "jQuery UI Tabs: Mismatching fragment identifier." );
        }

        if ( toShow.length ) {
            this.load( this.tabs.index( tab ), event );
        }
        this._toggle( event, eventData );
    },

    // handles show/hide for selecting tabs
    _toggle: function( event, eventData ) {
        var that = this,
            toShow = eventData.newPanel,
            toHide = eventData.oldPanel;

        this.running = true;

        function complete() {
            that.running = false;
            that._trigger( "activate", event, eventData );
        }

        function show() {
            eventData.newTab.closest( "li" ).addClass( "ui-tabs-active ui-state-active" );

            if ( toShow.length && that.options.show ) {
                that._show( toShow, that.options.show, complete );
            } else {
                toShow.show();
                complete();
            }
        }

        // start out by hiding, then showing, then completing
        if ( toHide.length && this.options.hide ) {
            this._hide( toHide, this.options.hide, function() {
                eventData.oldTab.closest( "li" ).removeClass( "ui-tabs-active ui-state-active" );
                show();
            });
        } else {
            eventData.oldTab.closest( "li" ).removeClass( "ui-tabs-active ui-state-active" );
            toHide.hide();
            show();
        }

        toHide.attr({
            "aria-expanded": "false",
            "aria-hidden": "true"
        });
```

```
        eventData.oldTab.attr( "aria-selected", "false" );
        // If we're switching tabs, remove the old tab from the tab order.
        // If we're opening from collapsed state, remove the previous tab from the tab order.
        // If we're collapsing, then keep the collapsing tab in the tab order.
        if ( toShow.length && toHide.length ) {
            eventData.oldTab.attr( "tabIndex", -1 );
        } else if ( toShow.length ) {
            this.tabs.filter(function() {
                return $( this ).attr( "tabIndex" ) === 0;
            })
            .attr( "tabIndex", -1 );
        }

        toShow.attr({
            "aria-expanded": "true",
            "aria-hidden": "false"
        });
        eventData.newTab.attr({
            "aria-selected": "true",
            tabIndex: 0
        });
    },

    _activate: function( index ) {
        var anchor,
            active = this._findActive( index );

        // trying to activate the already active panel
        if ( active[ 0 ] === this.active[ 0 ] ) {
            return;
        }

        // trying to collapse, simulate a click on the current active header
        if ( !active.length ) {
            active = this.active;
        }

        anchor = active.find( ".ui-tabs-anchor" )[ 0 ];
        this._eventHandler({
            target: anchor,
            currentTarget: anchor,
            preventDefault: $.noop
        });
    },

    _findActive: function( index ) {
        return index === false ? $() : this.tabs.eq( index );
    },

    _getIndex: function( index ) {
        // meta-function to give users option to provide a href string instead of a numerical
        index.
        if ( typeof index === "string" ) {
            index = this.anchors.index( this.anchors.filter( "[href$='" + index + "']" ) );
```

```
        }

        return index;
    },

    _destroy: function() {
        if ( this.xhr ) {
            this.xhr.abort();
        }

        this.element.removeClass( "ui-tabs ui-widget ui-widget-content ui-corner-all
        ui-tabs-collapsible" );

        this.tablist
            .removeClass( "ui-tabs-nav ui-helper-reset ui-helper-clearfix ui-widget-header
            ui-corner-all" )
            .removeAttr( "role" );

        this.anchors
            .removeClass( "ui-tabs-anchor" )
            .removeAttr( "role" )
            .removeAttr( "tabIndex" )
            .removeUniqueId();

        this.tabs.add( this.panels ).each(function() {
            if ( $.data( this, "ui-tabs-destroy" ) ) {
                $( this ).remove();
            } else {
                $( this )
                    .removeClass( "ui-state-default ui-state-active ui-state-disabled " +
                        "ui-corner-top ui-corner-bottom ui-widget-content ui-tabs-active
                        ui-tabs-panel" )
                    .removeAttr( "tabIndex" )
                    .removeAttr( "aria-live" )
                    .removeAttr( "aria-busy" )
                    .removeAttr( "aria-selected" )
                    .removeAttr( "aria-labelledby" )
                    .removeAttr( "aria-hidden" )
                    .removeAttr( "aria-expanded" )
                    .removeAttr( "role" );
            }
        });

        this.tabs.each(function() {
            var li = $( this ),
                prev = li.data( "ui-tabs-aria-controls" );
            if ( prev ) {
                li
                    .attr( "aria-controls", prev )
                    .removeData( "ui-tabs-aria-controls" );
            } else {
                li.removeAttr( "aria-controls" );
            }
        });
```

```
        this.panels.show();

        if ( this.options.heightStyle !== "content" ) {
            this.panels.css( "height", "" );
        }
    },

    enable: function( index ) {
        var disabled = this.options.disabled;
        if ( disabled === false ) {
            return;
        }

        if ( index === undefined ) {
            disabled = false;
        } else {
            index = this._getIndex( index );
            if ( $.isArray( disabled ) ) {
                disabled = $.map( disabled, function( num ) {
                    return num !== index ? num : null;
                });
            } else {
                disabled = $.map( this.tabs, function( li, num ) {
                    return num !== index ? num : null;
                });
            }
        }
        this._setupDisabled( disabled );
    },

    disable: function( index ) {
        var disabled = this.options.disabled;
        if ( disabled === true ) {
            return;
        }

        if ( index === undefined ) {
            disabled = true;
        } else {
            index = this._getIndex( index );
            if ( $.inArray( index, disabled ) !== -1 ) {
                return;
            }
            if ( $.isArray( disabled ) ) {
                disabled = $.merge( [ index ], disabled ).sort();
            } else {
                disabled = [ index ];
            }
        }
        this._setupDisabled( disabled );
    },

    load: function( index, event ) {
```

```
        index = this._getIndex( index );
        var that = this,
            tab = this.tabs.eq( index ),
            anchor = tab.find( ".ui-tabs-anchor" ),
            panel = this._getPanelForTab( tab ),
            eventData = {
                tab: tab,
                panel: panel
            };

        // not remote
        if ( isLocal( anchor[ 0 ] ) ) {
            return;
        }

        this.xhr = $.ajax( this._ajaxSettings( anchor, event, eventData ) );

        // support: jQuery <1.8
        // jQuery <1.8 returns false if the request is canceled in beforeSend,
        // but as of 1.8, $.ajax() always returns a jqXHR object.
        if ( this.xhr && this.xhr.statusText !== "canceled" ) {
            tab.addClass( "ui-tabs-loading" );
            panel.attr( "aria-busy", "true" );

            this.xhr
                .success(function( response ) {
                    // support: jQuery <1.8
                    // http://bugs.jquery.com/ticket/11778
                    setTimeout(function() {
                        panel.html( response );
                        that._trigger( "load", event, eventData );
                    }, 1 );
                })
                .complete(function( jqXHR, status ) {
                    // support: jQuery <1.8
                    // http://bugs.jquery.com/ticket/11778
                    setTimeout(function() {
                        if ( status === "abort" ) {
                            that.panels.stop( false, true );
                        }

                        tab.removeClass( "ui-tabs-loading" );
                        panel.removeAttr( "aria-busy" );

                        if ( jqXHR === that.xhr ) {
                            delete that.xhr;
                        }
                    }, 1 );
                });
        }
    },

    _ajaxSettings: function( anchor, event, eventData ) {
        var that = this;
```

```
        return {
            url: anchor.attr( "href" ),
            beforeSend: function( jqXHR, settings ) {
                return that._trigger( "beforeLoad", event,
                    $.extend( { jqXHR : jqXHR, ajaxSettings: settings }, eventData ) );
            }
        };
    },

    _getPanelForTab: function( tab ) {
        var id = $( tab ).attr( "aria-controls" );
        return this.element.find( this._sanitizeSelector( "#" + id ) );
    }
});

})( jQuery );

(function( $, undefined ) {

})( jQuery );

(function( $, window ) {

    $.mobile.iosorientationfixEnabled = true;

    // This fix addresses an iOS bug, so return early if the UA claims it's something else.
    var ua = navigator.userAgent,
        zoom,
        evt, x, y, z, aig;
    if ( !( /iPhone|iPad|iPod/.test( navigator.platform ) && /OS [1-5]_[0-9_]* like Mac OS X/i.
    test( ua ) && ua.indexOf( "AppleWebKit" ) > -1 ) ) {
        $.mobile.iosorientationfixEnabled = false;
        return;
    }

    zoom = $.mobile.zoom;

    function checkTilt( e ) {
        evt = e.originalEvent;
        aig = evt.accelerationIncludingGravity;

        x = Math.abs( aig.x );
        y = Math.abs( aig.y );
        z = Math.abs( aig.z );

        // If portrait orientation and in one of the danger zones
        if ( !window.orientation && ( x > 7 || ( ( z > 6 && y < 8 || z < 8 && y > 6 ) && x > 5 )
         ) ) {
                if ( zoom.enabled ) {
                    zoom.disable();
                }
        }   else if ( !zoom.enabled ) {
                zoom.enable();
        }
```

```
    }

    $.mobile.document.on( "mobileinit", function() {
        if ( $.mobile.iosorientationfixEnabled ) {
            $.mobile.window
                .bind( "orientationchange.iosorientationfix", zoom.enable )
                .bind( "devicemotion.iosorientationfix", checkTilt );
        }
    });

}( jQuery, this ));

(function( $, window, undefined ) {
    var $html = $( "html" ),
        $window = $.mobile.window;

    //remove initial build class (only present on first pageshow)
    function hideRenderingClass() {
        $html.removeClass( "ui-mobile-rendering" );
    }

    // trigger mobileinit event - useful hook for configuring $.mobile settings before they're
    used
    $( window.document ).trigger( "mobileinit" );

    // support conditions
    // if device support condition(s) aren't met, leave things as they are -> a basic, usable
    experience,
    // otherwise, proceed with the enhancements
    if ( !$.mobile.gradeA() ) {
        return;
    }

    // override ajaxEnabled on platforms that have known conflicts with hash history updates
    // or generally work better browsing in regular http for full page refreshes (BB5, Opera
    Mini)
    if ( $.mobile.ajaxBlacklist ) {
        $.mobile.ajaxEnabled = false;
    }

    // Add mobile, initial load "rendering" classes to docEl
    $html.addClass( "ui-mobile ui-mobile-rendering" );

    // This is a fallback. If anything goes wrong (JS errors, etc), or events don't fire,
    // this ensures the rendering class is removed after 5 seconds, so content is visible and
    accessible
    setTimeout( hideRenderingClass, 5000 );

    $.extend( $.mobile, {
        // find and enhance the pages in the dom and transition to the first page.
        initializePage: function() {
            // find present pages
            var path = $.mobile.path,
                $pages = $( ":jqmData(role='page'), :jqmData(role='dialog')" ),
```

```
            hash = path.stripHash( path.stripQueryParams(path.parseLocation().hash) ),
            hashPage = document.getElementById( hash );

        // if no pages are found, create one with body's inner html
        if ( !$pages.length ) {
            $pages = $( "body" ).wrapInner( "<div data-" + $.mobile.ns +
            "role='page'></div>" ).children( 0 );
        }

        // add dialogs, set data-url attrs
        $pages.each(function() {
            var $this = $( this );

            // unless the data url is already set set it to the pathname
            if ( !$this[ 0 ].getAttribute( "data-" + $.mobile.ns + "url" ) ) {
                $this.attr( "data-" + $.mobile.ns + "url", $this.attr( "id" ) || location.
                pathname + location.search );
            }
        });

        // define first page in dom case one backs out to the directory root (not always
        the first page visited, but defined as fallback)
        $.mobile.firstPage = $pages.first();

        // define page container
        $.mobile.pageContainer = $.mobile.firstPage
            .parent()
            .addClass( "ui-mobile-viewport" )
            .pagecontainer();

        // initialize navigation events now, after mobileinit has occurred and the page
        container
        // has been created but before the rest of the library is alerted to that fact
        $.mobile.navreadyDeferred.resolve();

        // alert listeners that the pagecontainer has been determined for binding
        // to events triggered on it
        $window.trigger( "pagecontainercreate" );

        // cue page loading message
        $.mobile.loading( "show" );

        //remove initial build class (only present on first pageshow)
        hideRenderingClass();

        // if hashchange listening is disabled, there's no hash deeplink,
        // the hash is not valid (contains more than one # or does not start with #)
        // or there is no page with that hash, change to the first page in the DOM
        // Remember, however, that the hash can also be a path!
        if ( ! ( $.mobile.hashListeningEnabled &&
            $.mobile.path.isHashValid( location.hash ) &&
            ( $( hashPage ).is( ":jqmData(role='page')" ) ||
                $.mobile.path.isPath( hash ) ||
                hash === $.mobile.dialogHashKey ) ) ) {
```

```
            // Store the initial destination
            if ( $.mobile.path.isHashValid( location.hash ) ) {
                $.mobile.navigate.history.initialDst = hash.replace( "#", "" );
            }

            // make sure to set initial popstate state if it exists
            // so that navigation back to the initial page works properly
            if ( $.event.special.navigate.isPushStateEnabled() ) {
                $.mobile.navigate.navigator.squash( path.parseLocation().href );
            }

            $.mobile.changePage( $.mobile.firstPage, {
                transition: "none",
                reverse: true,
                changeHash: false,
                fromHashChange: true
            });
        } else {
            // trigger hashchange or navigate to squash and record the correct
            // history entry for an initial hash path
            if ( !$.event.special.navigate.isPushStateEnabled() ) {
                $window.trigger( "hashchange", [true] );
            } else {
                // TODO figure out how to simplify this interaction with the initial
                history entry
                // at the bottom js/navigate/navigate.js
                $.mobile.navigate.history.stack = [];
                $.mobile.navigate( $.mobile.path.isPath( location.hash ) ? location.hash :
                location.href );
            }
        }
    }
});

$(function() {
    //Run inlineSVG support test
    $.support.inlineSVG();

    // check which scrollTop value should be used by scrolling to 1 immediately at domready
    // then check what the scroll top is. Android will report 0... others 1
    // note that this initial scroll won't hide the address bar. It's just for the check.

    // hide iOS browser chrome on load if hideUrlBar is true this is to try and do it as
    soon as possible
    if ( $.mobile.hideUrlBar ) {
        window.scrollTo( 0, 1 );
    }

    // if defaultHomeScroll hasn't been set yet, see if scrollTop is 1
    // it should be 1 in most browsers, but android treats 1 as 0 (for hiding addr bar)
    // so if it's 1, use 0 from now on
    $.mobile.defaultHomeScroll = ( !$.support.scrollTop || $.mobile.window.scrollTop() === 1
     ) ? 0 : 1;
```

```
        //dom-ready inits
        if ( $.mobile.autoInitializePage ) {
            $.mobile.initializePage();
        }

        // window load event
        // hide iOS browser chrome on load if hideUrlBar is true this is as fall back incase we
        were too early before
        if ( $.mobile.hideUrlBar ) {
            $window.load( $.mobile.silentScroll );
        }

        if ( !$.support.cssPointerEvents ) {
            // IE and Opera don't support CSS pointer-events: none that we use to disable
            link-based buttons
            // by adding the 'ui-disabled' class to them. Using a JavaScript workaround for
            those browser.
            // https://github.com/jquery/jquery-mobile/issues/3558

            // DEPRECATED as of 1.4.0 - remove ui-disabled after 1.4.0 release
            // only ui-state-disabled should be present thereafter
            $.mobile.document.delegate( ".ui-state-disabled,.ui-disabled", "vclick",
                function( e ) {
                    e.preventDefault();
                    e.stopImmediatePropagation();
                }
            );
        }
    });
}( jQuery, this ));


}));
```