JavaScript Libraries - jQuery Mobile Best Practices Guide

Version: 2013.11.24

Best practices for jQuery Mobile practitioners.

Checklist

❏ Always use namespacing when writing modules �

❏ Override jQuery Mobile defaults as needed �

❏ Create DOM nodes with jQuery API instead of string concat �

❏ Using the proper events �

❏ Use `.on` instead of `.live` �

❏ Write for Better Performance �

Device-specific considerations

❏ Use media query for loading css per orientation �

❏ Disabling long-touch events �

Common Problems and Workarounds

❏ Avoid shake problems while page transitioning �

❏ Avoid keyboard flicker problems �

❏ Avoid double click problem on Android 4.2 �

❑ Performance problem when loading large data �

Discussion

jQuery core is a DOM query mechanism based on CSS3 selectors to access and manipulate elements. It also provides a consistent API for HTTP communication, simple effects, and event handling. There are several extensions to jQuery that provide extra features, such as:

- jQuery UI
- jQuery Mobile
- Several hundred community sourced plug-ins

Source for the following information is from Zhang Xiaoming, IBM.

Clean coding

Always use namespacing when writing modules

Namespacing is a good practice because you have less chance of having conflicting names with other scripts. you can create a namespace for the whole application to hold all the global variables, functions, constants etc and create namespaces for each module to export some public interfaces for each module. Below is a utility function for creating namespaces.

```
(function($) {
    $.namespace = function() {
        var a = arguments, o = null, i, j, d;
        for (i = 0; i < a.length; i = i + 1) {
            d = a[i].split(".");
            o = window;
            for (j = 0; j < d.length; j = j + 1) {
                o[d[j]] = o[d[j]] || {};
                o = o[d[j]];
            }
        }
        return o;
    };
})(jQuery);
```

And you may use it as below:

```
(function($) {
    $.namespace('ft.flight');
    ...
    // public API for this module
    $.extend(ft.flight, {
        setDepartDate: function() { ... },
        getDepartDate: function() { ... }
    });
})(jQuery);
```

References:

- Unobtrusive JavaScript

Override jQuery Mobile defaults as needed

Generally, you may want to override some default settings of jQuery Mobile, like the transition effect, enable/disable page DOM cache and so on. You can create a separate javascript file (e.g. customize.js) for this. There is a event called ‘mobileinit’ to which you can listen and execute the customize code, the content of customize.js may like below:

```
(function($) {
    $(document).bind('mobileinit', function() {
        $.extend($.mobile, {
            defaultPageTransition: 'slide'
        });
        $.mobile.page.prototype.options.domCache = true;
    });
})(jQuery);
```

Also, you need to put the file after jQuery and ahead of jQuery Mobile:

```
<script src="js/jquery-1.8.2.js"></script>
<script src="js/customize.js"></script>
<script src="js/jquery.mobile-1.3.2.js"></script>
```

Create DOM nodes with jQuery API instead of string concat

It is very common to create dom nodes dynamically on mobile web development. A simple way is just append a html string to a dom node, it’s easy and common, but may cause a bad readability, you can consider to use jQuery API instead. For example, you want to append a html fragment:

```
<div id="a" style="width:100%;height:100%;">
  <span>test</span>
</div>
```

to document body, you can do it with jQuery:

```
$('<div>').attr('id', 'a')
          .css({ width: '100%', height: '100%' })
          .append($('<span>')
          .text('test'))
          .appendTo(document.body);
```

This improves the readability of your code and reduce the error probability greatly. What's more, you can create a template and use the 'clone' method to generate a listview:

```
<ul id="airportList" data-role="listview">
  <li class="template airport" style="display: none;">
    <a href="#" class="airportName"></a>
  </li>
</ul>

$.each(airports, function(i, ap) {
  var $itemAirport = $('#airportList.template.airport')
                       .clone();
  $('.airportName', $itemAirport).text(ap.name);
  $itemAirport.removeClass('template')
              .appendTo('#airportList').show();
});
```

Using the proper events

- **pageinit** : The 'pageinit' event handlers is to do some initialize work when loading the page for the first time. For example, you can put the binding event handlers related code here:

```
$(document).on('pageinit', '#mainPage', function() {
    $('#btnId').on('tap', function(e) {
        ...
    });
});
```

- **pagebeforeshow** : The 'pagebeforeshow' event handlers is suitable for change some kind of dynamically content of the page or loading data from remote server.

```
$(document).on("pagebeforeshow", "#selectOutboundFlightPage",
function() {
    $('.depart-airport .departAP', $(this)
).text(window.ft.departAP);
    $('.dest-airport .destAP', $(this)).text(window.ft.destAP);
});
```

References:

- [jQuery Events]()

Use .on instead of .live

The .live method is deprecated in higher version of jQuery, you should use .on to attach event handlers as a uniform way. The reason is jQuery will attempt to retrieve the elements specified by the selector before calling these kind of binding methods

References

- [JQuery live]()

Write for Better Performance

By default, there will be about 300ms delay for the click event on webview, which results in a very slow response/performance when click on a button. You can try to override the click event with the ‘tap’ event in jQuery Mobile to remove the delay:

```
$('btnId').on('tap', function(e) {
    e.stopImmediatePropagation();
    e.preventDefault();
    ...
});
```

And you can also do some preload and cache DOM nodes to make the transition smoothly:

```
$(document).on('mobileinit', function() {
    $.mobile.page.prototype.options.domCache = true; // cache the
dom nodes of loaded pages
    $.mobile.loadPage('targetPage.html'); // preload the target
page
    ...
});
```

The default duration of transition speed is about 300ms, which may be a little slow, you can change it with css:

```
.slide.in,.slide.out {
    -webkit-animation-timing-function: ease-in-out;
    -webkit-animation-duration: 200ms;
}

.slidefade.in,.slidefade.out {
    -webkit-animation-timing-function: ease-in-out;
    -webkit-animation-duration: 150ms;
}
```

Device-specific considerations

Using media query for loading css per orientation

If you want to have different UI for portrait and landscape mode, you can write separate css for portrait and landscape and dynamically load them with css media query as below:

```
<link rel="stylesheet" media="all and (orientation:landscape)"
href="css/landscape.css">
<link rel="stylesheet" media="all and (orientation:portrait)"
href="css/portrait.css">
```

Disabling long-touch events

You can add the following css to disable user selection/action in webview.

```
* {
    -webkit-touch-callout: none;
    -webkit-user-select: none;
}
```

Common problems and workarounds

Avoid shake problems while page transitioning

By default, jQuery mobile will scroll to the top of the page (to hide the address bar on of web browser), you can add the following code to disable this behavior in ‘pageinit’ event handler of your first page.

```
$("#loginPage").on("pageinit", function() {
    $.mobile.defaultHomeScroll = 0;
});
```

Avoid keyboard flicker problems

On some Android devices (most of os version is lower then 4), when tap on an input box, the keyboard will cover the input box. This problem also occurs on iOS platform if you are using an scroll plugin like iScroll. A workaround for this is to create a div to jack up the page manually. Below is a piece of script code to show how to fix the issue on iOS when using iScroll plugin.

```
var $currScroller = $(el).parents('.iscroll-wrapper');
var _id = jq(this).attr('id');
window._focusElements = [];
$(el).on('vmousedown', function() {
  window._focusElements.push(_id);

  // the keyboard_holder is a div node to jack up the page
  if($currScroller.find('.keyboard_holder').is(':visible')) {
    $currScroller.scrollTo(0, -scrollHeight, 0);
    $(el).focus();
  } else {
    if(!$currScroller.find('.keyboard_holder').length) {
      $currScroller.find('.iscroll-content')
        .append($('<div class="keyboard_holder"
 style="height:500px;display:none;">'));
    }
    $currScroller.find('.keyboard_holder').show();
  }
  return false;
}).focusin(function() {
  if($.inArray(_id, window._focusElements) < 0) {
    window._focusElements.push(_id);
    $currScroller.scrollTo(0, -scrollHeight, 0);
  }
}).focusout(function() {
  window._focusElements.pop();

  setTimeout(function() {
    if(!window._focusElements.length) {
      $currScroller.find('.keyboard_holder').hide();
      $currScroller.refresh();
    }
  }, 0);
});
```

Avoid double click problem on Android 4.2

Applicable to jQuery mobile 1.3.1 + iScroll4. Since Android 4.2 changed it's default WebView core, the event handler is not the same as previous Android version. On Android 4.2, when you tap on an item, it will trigger 'tap' and 'click' event on the same element, when the 'tap' event bubbling to iScroll, iScroll will trigger the 'click' event. However, on Android 2.3, it seems only 'tap' event occurs on the element bubbling to iScroll. Not quite sure what jQueryMobile/iScroll did to prevent the 'click' event. When iScroll captures the 'click' event, it will dispatch again and then element will capture the 'click' event to expend/close items.

To fix this problem, we can do the following:

1. Update jQuery to 1.9.1 and jQuery Mobile to 1.3.2.
2. Set iScroll handleClick option to true (we need iScroll to dispatch event on low version Android)
3. Stop 'tap' event bubbling for Android 4.2, by this the 'tap' event not occurs on the element and not dispatch to iScroll, so iScroll not dispatch 'click' event. In the meanwhile, the element will get 'click' event on itself, so it can do the action successfully.
4. When application initialization done, we can check the device version and bind jQuery event on the elements that were included in iScroll and stop propagation for 'tap' event.

Performance problem when loading large data

In some apps like booking flight tickets, a common requirement is to display all the city list in a page, which will cause a performance problem on low-powered devices. To solve this problem, we need to limit the size of the list, and change the content of the list dynamically. Please refer to the sample project for more details here.

Other tips

Set value for checkbox/radio button group programmatically

Sometimes you may want to change the value of a radio button group programmatically, but there isn't any api from jquery mobile widget to do this. So you need to find the radio button and check it manually:

```
$('#radio').attr('checked', true);
```

Do not forget to refresh all the button group after this:

```
$('#radioButtonGroup input[type=radio]')
  .checkboxradio('refresh');
```

Reference Links

- Is jQuery The Best Option For Large-Scale App Development?
- jquery-week-calendar/weekcalendar_json.html at master · themouette/jquery-week-calendar
- jQuery Learning Center
- IgniteUI Controls - Samples of IgniteUI Controls - IgniteUI Featured Controls
- Slide in menu for mobile webkit
- Rotating wheel for your iPhone webapps
- Spinning wheel on webkit for iPhone/iPod touch
- On jQuery & Large Applications - rmurphey
- jQuery File Upload Demo
- 16 jQuery Menus with Beautiful Animations for Developers | DJDESIGNERLAB
- advanced-jquery-boilerplate/advanced-jquery-boilerplate.js at master · elclanrs/advanced-jquery-boilerplate