jQuery Advanced Module 3

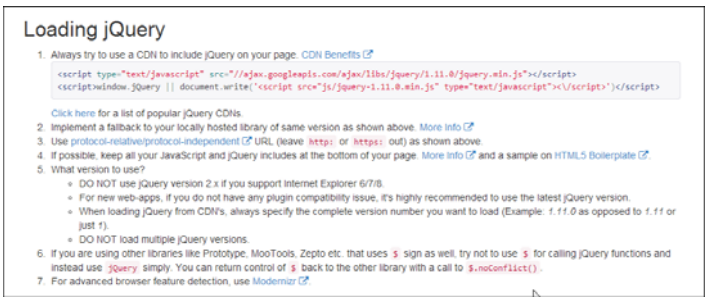
jQuery Coding Standards & Best Practices

Peter Kassenaar – info@kassenaar.com

jQuery Best Practices & coding standards

Credits: http://lab.abhinayrathore.com/jquery-standards/

1. JQuery Laden
   - Gebruik een CDN (geldt *niet* voor mobile/standalone apps)
   - Schrijf lokaal een fallback-optie.
   - Kies de goede jQuery branch: 1.x vs. 2.x



Loading jQuery

1. Always try to use a CDN to include jQuery on your page. CDN Benefits

```
<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
<script>window.jQuery || document.write('<script src="js/jquery-1.11.0.min.js" type="text/javascript"><\/script>')</script>
```

Click here for a list of popular jQuery CDNs.

2. Implement a fallback to your locally hosted library of same version as shown above. More Info
3. Use protocol-relative/protocol-independent URL (leave `http:` or `https:` out) as shown above.
4. If possible, keep all your JavaScript and jQuery includes at the bottom of your page. More Info and a sample on HTML5 Boilerplate
5. What version to use?
   - DO NOT use jQuery version 2.x if you support Internet Explorer 6/7/8.
   - For new web-apps, if you do not have any plugin compatibility issue, it's highly recommended to use the latest jQuery version.
   - When loading jQuery from CDN's, always specify the complete version number you want to load (Example: *1.11.0* as opposed to *1.11* or just *1*).
   - DO NOT load multiple jQuery versions.
6. If you are using other libraries like Prototype, MooTools, Zepto etc. that uses `$` sign as well, try not to use `$` for calling jQuery functions and instead use `jQuery` simply. You can return control of `$` back to the other library with a call to `$.noConflict()`.
7. For advanced browser feature detection, use Modernizr.

2. Variabelen in jQuery scripts

- Een variabele die gewrapt is in een jQuery object begint met $
- Cache je variabelen voor betere performance bij hergebruik
- Gebruik camelCasing

jQuery Variables

1. All variables that are used to store/cache jQuery objects should have a name prefixed with a `$`.
2. Always cache your jQuery selector returned objects in variables for reuse.

```
var $myDiv = $("#myDiv");
$myDiv.click(function(){...});
```

3. Use camel case for naming variables.

5HART-IT

3. Selectors

- Gebruik ID-selectors voor snelheid, single classes voor class-selectie, zonder element of ID-specificiteit
- Gebruik .find() voor subselecties
- Geen overbodige specificiteit
- Gebruik waar mogelijk context selector

Selectors

1. Use ID selector whenever possible. It is faster because they are handled using `document.getElementById()`.
2. When using class selectors, don't use the element type in your selector. Performance Comparison

```
var $products = $("div.products"); // SLOW
var $products = $(".products"); // FAST
```

3. Use find for *Id->Child* nested selectors. The `.find()` approach is faster because the first selection is handled without going through the Sizzle selector engine. More Info

```
// BAD, a nested query for Sizzle selector engine
var $productIds = $("#products div.id");

// GOOD, #products is already selected by document.getElementById() so only div.id needs to go through Sizzle selector engine
var $productIds = $("#products").find("div.id");
```

5HART-IT

4. Be specific on the right-hand side of your selector, and less specific on the left. More Info

```
// Unoptimized
$("div.data .gonzalez");

// Optimized
$(".data td.gonzalez");
```

5. Avoid Excessive Specificity. More Info, Performance Comparison

```
$(".data table.attendees td.gonzalez");

// Better: Drop the middle if possible.
$(".data td.gonzalez");
```

6. Give your Selectors a Context.

```
// SLOWER because it has to traverse the whole DOM for .class
$('.class');

// FASTER because now it only looks under class-container.
$('.class', '#class-container');
```

7. Avoid Universal Selectors. More Info

```
$('div.container > *'); // BAD
$('div.container').children(); // BETTER
```

8. Avoid Implied Universal Selectors. When you leave off the selector, the universal selector (*) is still implied. More Info

```
$('div.someclass :radio'); // BAD
$('div.someclass input:radio'); // GOOD
```

9. Don't Descend Multiple IDs or nest when selecting an ID. ID-only selections are handled using `document.getElementById()` so don't mix them with other selectors.

```
$('#outer #inner'); // BAD
$('div#inner'); // BAD
$('.outer-container #inner'); // BAD
$('#inner'); // GOOD, only calls document.getElementById()
```

5HART-IT

4. DOM Manipulation

- Gebruik `.detach()` als je (extreem) veel handelingen moet verrichten met een element. Voeg het daarna weer toe met `.append()` of `.appendTo()`.
- Test eerst of een element bestaat voordat je animatie gebruikt.

1. Always detach any existing element before manipulation and attach it back after manipulating it. More Info

```
var $myList = $("#list-container > ul").detach();
//...a lot of complicated things on $myList
$myList.appendTo("#list-container");
```

Niet (meer) van toepassing: `Array.join()`. Maar test dit in specifieke situatie

5HART-IT

5. Events

- Gebruik één `document.ready()` per pagina.
- Gebruik benoemde functies (in plaats van anonymous functions)
- Vermijd inline event handlers (`onclick="…"`) (==oldschool)

1. Use only one Document Ready handler per page. It makes it easier to debug and keep track of the behavior flow.
2. DO NOT use anonymous functions to attach events. Anonymous functions are difficult to debug, maintain, test, or reuse. More Info

```
$("#myLink").on("click", function(){...}); // BAD

// GOOD
function myLinkClickHandler(){...}
$("#myLink").on("click", myLinkClickHandler);
```

3. Document ready event handler should not be an anonymous function. Once again, anonymous functions are difficult to debug, maintain, test, or reuse.

```
$(function(){ ... }); // BAD: You can never reuse or write a test for this function.

// GOOD
$(initPage); // or $(document).ready(initPage);
function initPage(){
    // Page load event where you can initialize values and call other initializers.
}
```

5HART-IT

6. Ajax

- Gebruik `$.ajax()` in plaats van de shorthand methods
- Wees specifiek met betrekking tot `dataType`.
- Gebruik de `data:`-parameter in plaats van inline querystring
- Gebruik deferred objects/promises in plaats van `success:`- en `error:`- callback

4. Try to specify the *dataType* setting so it's easier to know what kind of data you are working with. (See Ajax Template example below)
5. Use Delegated event handlers for attaching events to content loaded using Ajax. Delegated events have the advantage that they can process events from descendant elements that are added to the document at a later time (example Ajax). More Info

```
$("#parent-container").on("click", "a", delegatedClickHandlerForAjax);
```

6. Use Promise interface: More Examples

```
$.ajax({ ... }).then(successHandler, failureHandler);

// OR
var jqxhr = $.ajax({ ... });
jqxhr.done(successHandler);
jqxhr.fail(failureHandler);
```

5HART-IT

7. Overig

- Effecten en animaties
- Plug-ins
- Chaining en object literals
- Vermijd mixen van CSS en JavaScript/jQuery

Miscellaneous

1. Use Object literals for parameters

```
$myLink.attr("href", "#").attr("title", "my link").attr("rel", "external"); // BAD, 3 calls to attr()
// GOOD, only 1 call to attr()
$myLink.attr({
    href: "#",
    title: "my link",
    rel: "external"
});
```

2. Do not mix CSS with jQuery

```
$("#mydiv").css({'color':red, 'font-weight':'bold'}); // BAD
```

```
.error { color: red; font-weight: bold; } /* GOOD */
```


John Resig over jQuery

De meester aan het woord

http://ejohn.org/apps/workshop/adv-talk/#0

Enigszins verouderd, maar alleszins de moeite waard!

Improved Creation

In jQuery 1.4 we provided a new way to create an element and then modify it.

All the setting is put into a single object.

```
$("<li/>", {
  click: function(){},
  id: "test", // mix ids and jQuery methods
  addClass: "clickable"
});
```

Custom Selectors

You can create custom selectors quite easily.

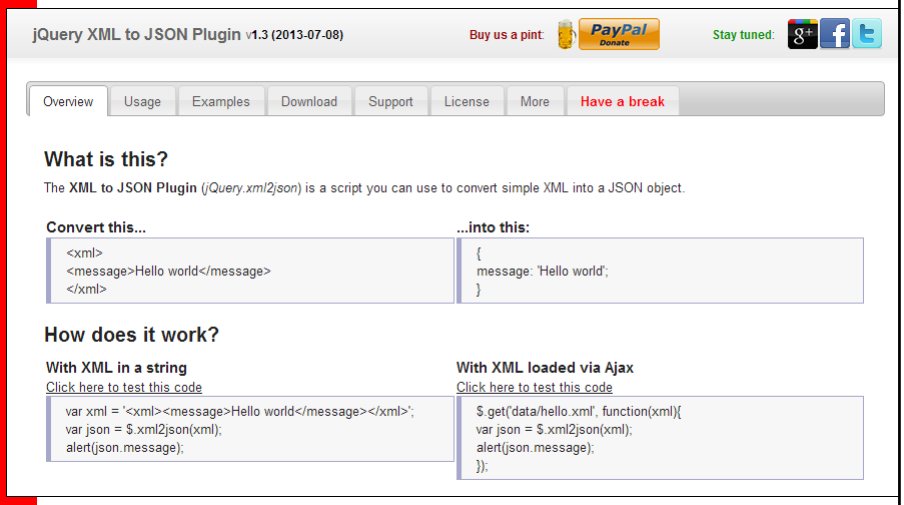
Add custom selectors to your plugins for easy querying.

```
jQuery.expr[":"].myplugin = function(elem) {
  return !!jQuery.data("myplugin");
};
```

Leestips

- documentsFragments zijn niet echt doorgebroken.
- Verouderd: opmerkingen over `.bind()`, `.live()` en `.delegate()`.
  - Deze zijn in nieuwe versies van jQuery gebundeld in `.on()`
- Veel issues die genoemd worden als ‘nieuw’ zijn inmiddels gerealiseerd.

XML to JSON plugin



http://www.fyneworks.com/jquery/xml-to-json/