



## jQuery Advanced Module 2



jQuery Plug-ins  
Gebruiken en zelf maken

Peter Kassenaar – info@kassenaar.com



## Plug-ins

- Doel: standaardgedrag uitbreiden
- Herhaalde gedragingen automatiseren
- Doeleinden:
  - Photo gallery
  - Slideshow
  - Validatie
  - tabellen
  - Drag/Drop
  - ...



SHART-IT



## Algemene werkwijze

1. jQuery Core invoegen via `<script>`
2. plug-in toevoegen via `<script>`
3. HTML/CSS/JavaScript schrijven
4. plugin activeren via `$('#selectie').plugin();`
5. eventueel: configuratieparameters.

Lees *altijd* de documentatie op de site!

SHART-IT



## Voorbeeld – de plug-in Cycle gebruiken

- <http://jquery.malsup.com/cycle/>

```

5 <title>jQuery voorbeeld 01</title>
6 <script src="scripts/jquery-1.7.1.min.js"></script>
7 <script src="scripts/jquery.cycle.all.js"></script>
8 </head>
9
10 <body>
11 <div id="fotoContainer">
12 
13 
14 
15 
16 
17 
18 </div>
19 <script>
20 $(document).ready(function() {
21   $('#fotoContainer').cycle();
22 });
23 </script>
24 </body>

```



SHART-IT



## Meer plug-ins – categorieën

- Animaties
- Formulieren
- Responsive
- Validaties

Algemeen: de kwaliteit is wisselend. Afhankelijk van documentatie van de makers. Bekijk ook:

- Activiteit op github
- Ondersteunend forum, discussiegroep, StackOverflow?
- Sites die de plug-in gebruiken

SHART-IT



## Zelf plug-ins maken

Demo: chainable plug-in maken in 4 stappen

Algemene template voor elke plug-in:

```
(function ($) {  
    // algemeen template  
    // jouw plugin code...  
})(jQuery);
```



SHART-IT



## Best Practices bij plug-ins

```
(function($){
  $.fn.colorize = function(){
    // alleen de onderdelen van de collectie die zijn voorzien
    return this.each(function(index, el){
      // code voor elk van de onderdelen van de selectie
    });
  };
})(jQuery);
```

1. Object \$.fn uitbreiden
2. Return `this.each(function(){...});`

SHART-IT



## Basisstructuur plug-in

```
01. //You need an anonymous function to wrap around your function to
    //avoid conflict
02. (function($){
03.
04.     //Attach this new method to jQuery
05.     $.fn.extend({
06.
07.         //This is where you write your plugin's name
08.         pluginname: function() {
09.
10.             //Iterate over the current set of matched elements
11.             return this.each(function() {
12.
13.                 //code to be inserted here
14.
15.             });
16.         }
17.     });
18.
19. //pass jQuery to the function,
20. //So that we will be able to use any valid Javascript variable name
21. //to replace "$" SIGN. But, we'll stick to $ (I like dollar sign:
    // )
22. })(jQuery);
```

SHART-IT



## Advanced plug-in concepts

- <http://learn.jquery.com/plugins/advanced-plugin-concepts/>
  - Default plug-in settings
  - Definieer publieke functies in de plugin-API
  - Gebruik closures om private functies af te schermen
  - Bied callback-faciliteiten

Posted in: [Plugins](#)

### Advanced Plugin Concepts

**Provide Public Access to Default Plugin Settings**

An improvement we can, and should, make to the code above is to expose the default plugin settings. This is important because it makes it very easy for plugin users to override/customize the plugin with minimal code. And this is where we begin to take advantage of the function object.

```

1 // Plugin definition.
2 $.fn.hilight = function( options ) {
3
4     // Extend our default options with those provided.
5     // Note that the first argument to extend is an empty
6     // object - this is to keep from overwriting our "defaults" object.
7     var opts = $.extend( {}, $.fn.hilight.defaults, options );
8
9     // Our plugin implementation code goes here.
10
11 };
12
13 // Plugin defaults - added as a property on our plugin function.
14 $.fn.hilight.defaults = {
15     foreground: "red",
16     background: "yellow"
17 };
  
```

SHART-IT



## Een plug-in analyseren

- <http://rubentd.com/toasty/>

**toasty**

Show Dan Forden's Toasty from Mortal Kombat as an Easter Egg for your website

[Show toasty](#) [Advanced plug-in concepts](#)

### Instructions


Include the necessary files, add the plugin and then call the 'pop' event inside the event that will trigger the easter egg.  
Combine with the Kenami Code for awesome results.

### Example code

```

1 <button id="toasty-button" style="padding: 5px;">Show toasty</button>
2
3 <link rel="stylesheet" href="toasty.css">
4 <script type="text/javascript" src="jquery-2.1.0.min.js"></script>
5 <script type="text/javascript" src="jquery.toasty.js"></script>
6 <script>
7     $(document).ready( function(){
8
9         $("body").toasty();
10
11         $("#toasty-button").click( function(){
  
```

SHART-IT




# jQuery

## Performance tips

Snellere code door betere syntaxis

---

SHART-IT



## Optimization is nuttig, MAAR....

- Schrijf je code begrijpelijk
  - Als je er een half uur naar zit te staren voordat je begrijpt wat een stukje code doet, heb je alle milliseconden tijdswinst al teniet gedaan.
  - Er moeten waarschijnlijk meer developers met jouw code aan de slag!
- Huidige computers, telefoons en browsers zijn *SNEL*....
  - "Mijn iPhone van nu is sneller dan mijn iMac uit 2009"
  - "Een DOM-tree van 2000 elementen wordt in minder dan 50ms samengesteld en gerenderd."

Vertraging in de praktijk: omvang & duur van Ajax-requests, afbeeldingen en (gebrek aan) caching

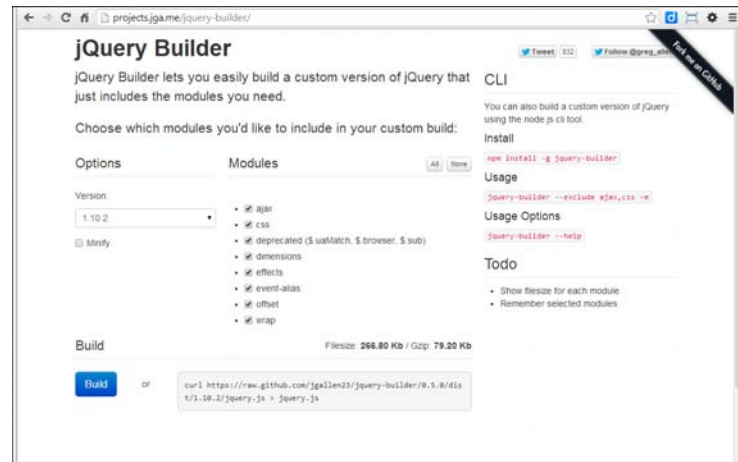
---

SHART-IT



## 1. Include alleen benodigde modules

<http://projects.jga.me/jquery-builder/>



SHART-IT



## 2. Optimize selectors

- Gebruik ID-based selectors
- Gebruik .find() om binnen context te selecteren

<http://learn.jquery.com/performance/optimize-selectors/>

Posted in: [Performance](#)

Beta

### Optimize Selectors

Selector optimization is less important than it used to be, as more browsers implement `document.querySelectorAll()` and the burden of selection shifts from jQuery to the browser. However, there are still some tips to keep in mind.

#### ID-Based Selectors

Beginning your selector with an ID is always best.

```

1 // Fast:
2 $( "#container div.robotarm" );
3
4 // Super-fast:
5 $( "#container" ).find( "div.robotarm" );

```

The `.find()` approach is faster because the first selection is handled without going through the Sizzle selector engine – ID-only selections are handled using `document.getElementById()`, which is extremely fast because it is native to the browser.

SHART-IT



### 3. Gebruik de nieuwste jQuery-versie

- 1.x-branch: 1.10 voor legacy browsers
- 2.x-branch: 2.0+ voor Moderne browsers en mobiele apparaten
  - Optimalisaties
  - Veel oude code verwijderd & rest geoptimaliseerd
- Gebruik een CDN indien mogelijk
- Gebruik een minifier om scripts te verkleinen

SHART-IT



### 4. Gebruik native functies waar mogelijk

- Een native JavaScript for-lus is tientallen malen sneller dan een \$.each(lus).

```

<!-- custom scripts -->
<script>
  // Performance testing: vul een array met 1 miljoen items en loop over die array
  // 1. Native
  var array = new Array();
  for (var i = 0; i < 1000000; i++) {
    array[i] = 0;
  }

  console.time('native');
  var l = array.length;
  for (var i = 0; i < l; i++) {
    array[i] = i;
  }
  console.timeEnd('native');

  // 2. jQuery
  console.time('jquery');
  $.each(array, function (i) {
    array[i] = i;
  });
  console.timeEnd('jquery');

```

SHART-IT





## 5. Selectie via ID's in plaats van Classes

- Zie ook #2. Selectie via ID's verloopt 5-7x sneller

```
// Of: voor meer complexe items en selecties; gebruik ID's in plaats van classes.
// Een lijst maken, deze vullen met items en daarna elk item selecteren:

// 3. Via classes.
console.time('class');
var list = $('#list');
var items = '<ul>';

for (i = 0; i < 1000; i++) {
    items += '<li class="item" + i + ">item</li>';
}

items += '</ul>';
list.html(items);

for (i = 0; i < 1000; i++) {
    var s = $('#.item' + i);
}
console.timeEnd('class');

// 4. Via ID's
console.time('id');
var list = $('#list');
```

SHART-IT



## 6. Caching van selecties

Gebruik chaining om een element te selecteren en daarna te bewerken. EN:

Maak een selectie één keer. Sla deze op in een variabele. Gebruik daarna de variabele voor bewerkingen.

```
<script>
// Gebruik chaining om elementen te selecteren en daarna te bewerken.

// 1. FOUT:
$('#divResult').css('color', 'green');
$('#divResult').html('hello');
$('#divResult').css('background-color', 'ffffff');

// 2. BETTER:
$('#divResult')
    .css('color', 'green')
    .html('Hello World')
    .css('background-color', 'ffffff');

// 3. NOG BETTER (vooral als er meer dan 1 element in de selectie zit en je er overheen loopt)
var item = $('#divResult');
item.css('color', 'green');
item.html('Hello World');
item.css('background-color', 'ffffff');
```

SHART-IT



## Test cases: gebruik jsperf.com

- Bijvoorbeeld: <http://jsperf.com/ns-jq-cached/3>

Done. Ready to run again. Run again

Testing in Chrome 33.0.1750.146 32-bit on Windows NT 6.3 64-bit

	Test	Ops/sec
<b>Cached elements</b>	<pre>cached.css("color", "#ff0000"); cached.css("text-decoration", "underline"); cached.css("background-color", "#ffffff");</pre>	21,470 ±2.04% fastest
<b>No cache</b>	<pre>jQuery("#top").find("p.mytext").css("color", "#ff0000"); jQuery("#top").find("p.mytext").css("text-decoration", "underline"); jQuery("#top").find("p.mytext").css("background-color", "#ffffff");</pre>	8,786 ±0.87% 59% slower
<b>Cache2</b>	<pre>cached.css("color", "#ff0000").css("text-decoration", "underline").css("background-color", "#ffffff");</pre>	21,066 ±3.62% fastest
<b>Cach3</b>	<pre>cached.css([   "color": "#ff0000",   "text-decoration": "underline",   "background-color": "#ffffff" ]);</pre>	18,589 ±0.67% 12% slower

Compare results of other browsers

Chart type: [bar](#), [column](#), [line](#), [pie](#), [table](#)

SHART-IT



## 7. Vermijd DOM-manipulatie

- Vermijd DOM-manipulatie zoveel mogelijk. In ieder geval in lussen.

```
<!-- custom scripts -->
<script>
  // 1. Vermijd DOM-manipulatie in lussen
  console.time('DOM-manipulation inside loop');
  for (var i = 0; i < 100000; i++) {
    $('#divResult').append(' hello', i);
  }
  console.timeEnd('DOM-manipulation inside loop');

  // 2. Een betere manier, in de lus een string samenstellen en deze buiten
  console.time('DOM-manipulation outside loop');
  var text = '';
  for (var i = 0; i < 100000; i++) {
    text += ' hello' + i;
  }
  $('#divResult').append(i);
  console.timeEnd('DOM-manipulation outside loop');
```

SHART-IT



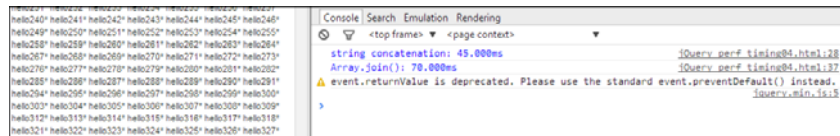
## 8. Achterhaald: “gebruik Array.join() in plaats van stringconcatenatie”

Vaak lees je nog: “Array.join() is vele malen sneller dan stringconcatenatie”

```
string = oldString + newString
```

Dit is overigens algemeen JavaScript – niet specifiek voor jQuery.

Achterhaald in moderne browsers!



<http://www.sitepen.com/blog/2008/05/09/string-performance-an-analysis/>

SHART-IT




## Meer performance tips – Addy Osmani



<http://vimeo.com/18846584>

SHART-IT



## Maak je keuze

App-design & architectuur	Data binding options
"jQuery free JavaScript"	Mobile strategies & Frameworks

SHART-IT