jQuery Advanced

Complexe webapps met jQuery

Peter Kassenaar – info@kassenaar.com

Tevens besproken:

- `.on()` - event handling, met *contextual selector.* Zie boek p. 112 en verder.
- `var` gebruiken als keyword om functions scoping mee te geven en niet op globale namespace te plaatsen.
- `===` gebruiken in plaats van `==` om type safe checking te doen.
- Function hoisting: functies worden door de JavaScript-compiler verplaatst naar de *top van de outer most function.* Terwijl variabelen (`var`) just-in-time worden gecompileerd.
- Let op de regels voor ASI: best practice: *altijd* zelf je semicolon plaatsen.
- Regels voor CSS3-selectors: [] met + en > . Zie ook http://www.w3schools.com/cssref/css_selectors.asp

Tevens besproken (2)

- jQuery Validation Plugin: alles wat je wilt over validatie. Is vaak handiger (en in ieder geval sneller) dan zelf schrijven: http://jqueryvalidation.org/
- jQuery performance tips & tricks.

Introductie

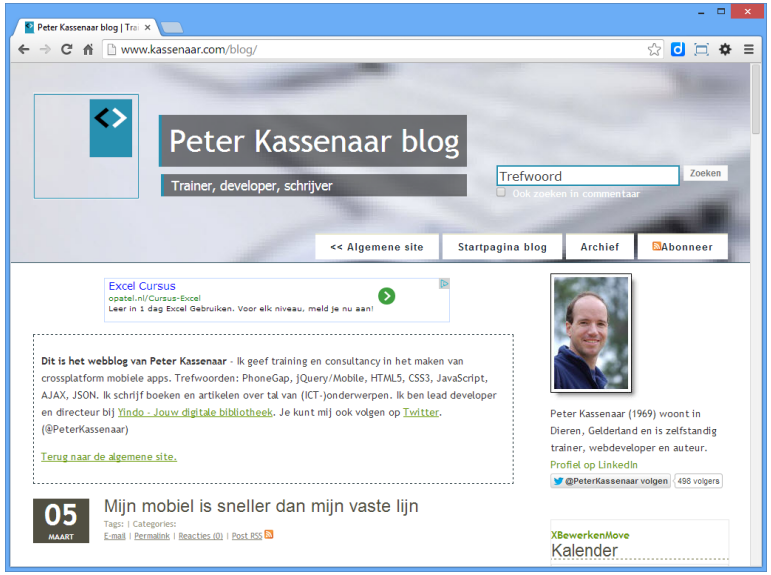
Peter Kassenaar;

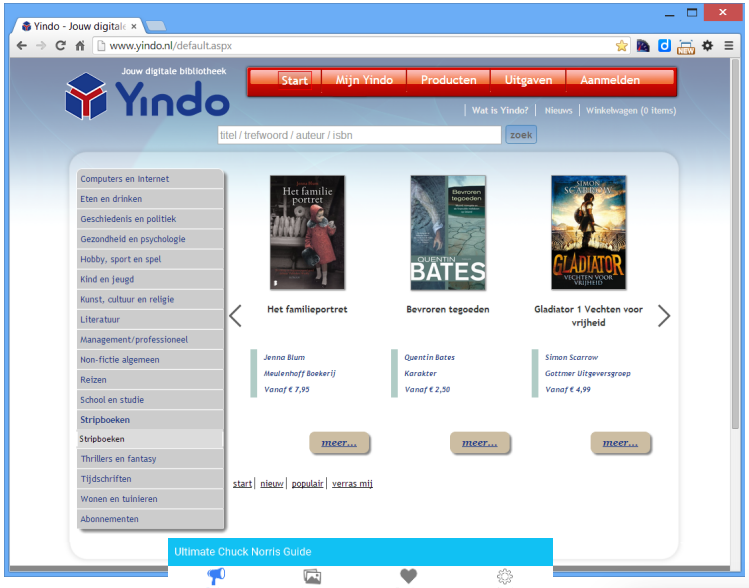
- Auteur, developer, docent
- Eigen bedrijf (1996)

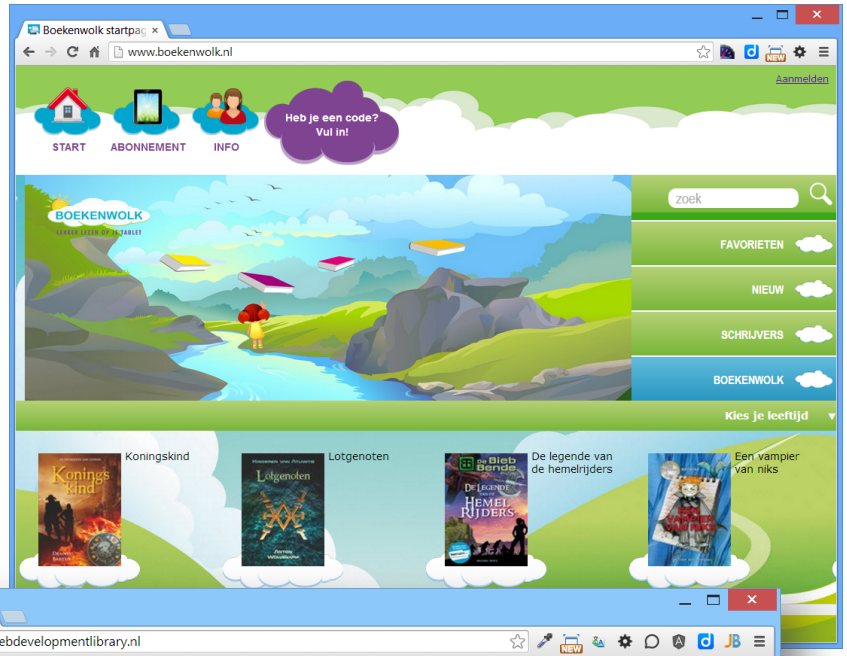
www.kassenaar.com/blog

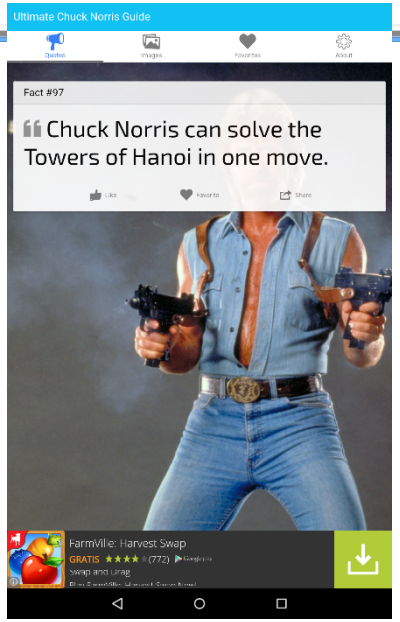
info@kassenaar.com

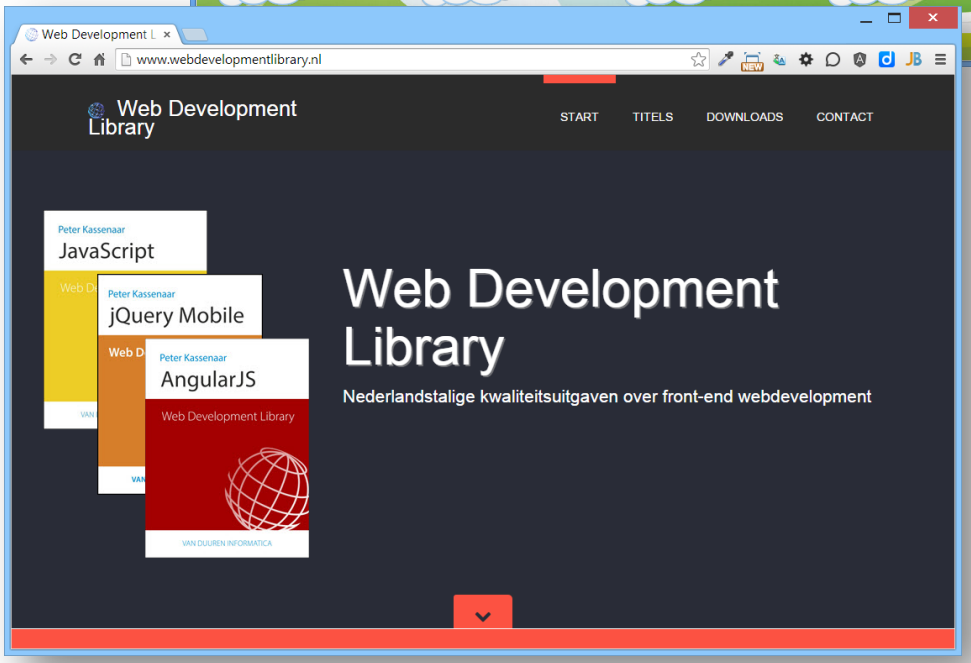
Twitter: @PeterKassenaar




Yindo - Jouw digitale
www.yindo.nl/default.aspx
Jouw digitale bibliotheek
Yindo
Start
Mijn Yindo
Producten
Uitgaven
Aanmelden
Wat is Yindo?
Nieuws
Winkelwagen (0 items)
titel / trefwoord / auteur / isbn
zoek
Computers en Internet
Eten en drinken
Geschiedenis en politiek
Gezondheid en psychologie
Hobby, sport en spel
Kind en jeugd
Kunst, cultuur en religie
Literatuur
Management/professioneel
Non-fictie algemeen
Reizen
School en studie
Stripboeken
Thrillers en fantasy
Tijdschriften
Wonen en tuinieren
Abonnementen
Het familieportret
Bevroren tegoeden
Gladiator 1 Vechten voor vrijheid
Jenna Blum
Meulenhoff Boekerij
Vanaf € 7,95
Quentin Bates
Karakter
Vanaf € 2,50
Simon Scarrow
Gottmer Uitgeversgroep
Vanaf € 4,99
meer...
meer...
meer...
start | nieuw | populair | verras mij


Boekenwolk startpag
www.boekenwolk.nl
Aanmelden
START
ABONNEMENT
INFO
Heb je een code? Vul in!
BOEKENWOLK
zoek
FAVORIETEN
NIEUW
SCHRIJVERS
BOEKENWOLK
Kies je leeftijd
Koningskind
Lotgenoten
De legende van de hemelrijders
Een vampier van niks


Ultimate Chuck Norris Guide
Fact #97
Chuck Norris can solve the Towers of Hanoi in one move.
FarmVille: Harvest Swap
GRATIS
Swap and Urag


Web Development L
www.webdevelopmentlibrary.nl
Web Development Library
START
TITELS
DOWNLOADS
CONTACT
Peter Kassenaar
JavaScript
Peter Kassenaar
jQuery Mobile
Peter Kassenaar
AngularJS
Web Development Library
VAN DUUREN INFORMATICA
Web Development Library
Nederlandstalige kwaliteitsuitgaven over front-end webdevelopment

Over jullie...

Voorkennis webdesign, (mobile) apps?

Voorkennis andere (web-)talen?

Verwachtingen van de cursus?

Concrete projecten?

Materialen

Software

Handouts

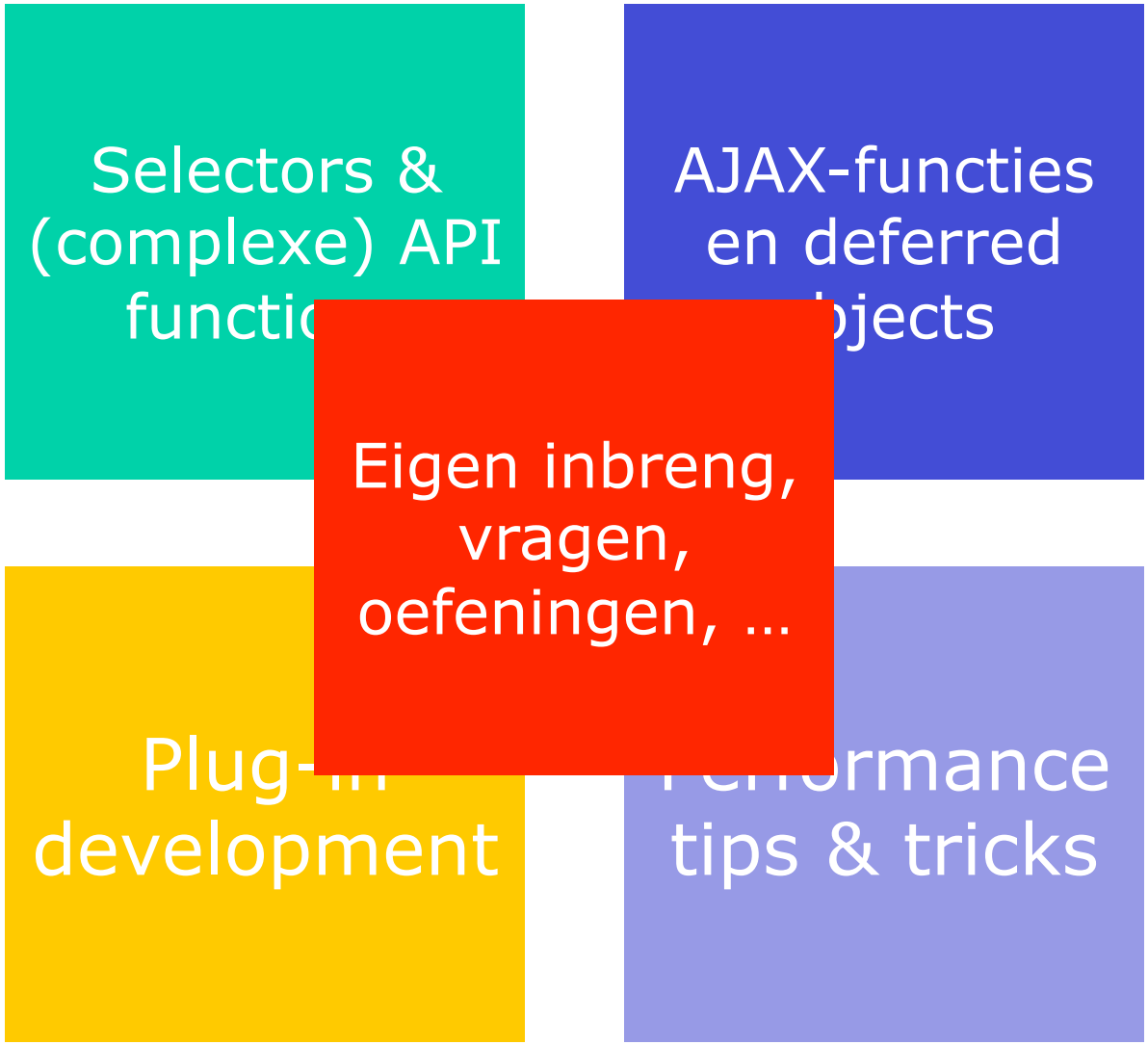
Oefeningen

Websites

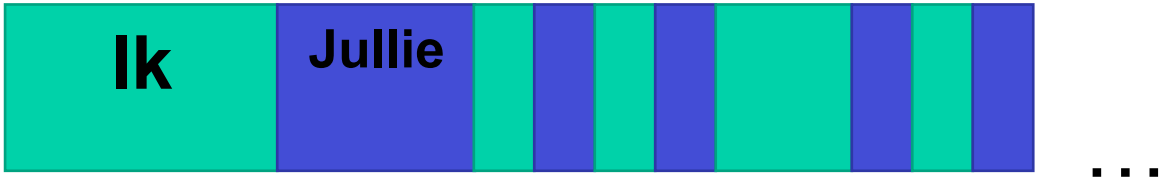
Indeling

- Geschiedenis - plaatsbepaling – context
- Webapps met jQuery
- In detail:
  - Selectors, .each(), .data(), .extend() en meer.
  - Werken met Ajax
  - Deferred objects
  - Plug-ins ontwikkelen
  - …

In een notendop



Werkwijze



Vragen?

"jQuery makes JavaScript suck less…"

"jQuery is what the HTML DOM should have been!"

UI

Mobile

API

What is jQuery?

- *Javascript HTML DOM Manipulation library*
- Light weight (~80k compressed, all modules)
- Powerful and extremely practical functionality
- Easy to learn and intuitive to use
- *CSS 3 based* selector syntax
  - ID's, Classes, Elements
- Custom Event handling
- Easily to extend through plug-in API

Decompose jQuery statements

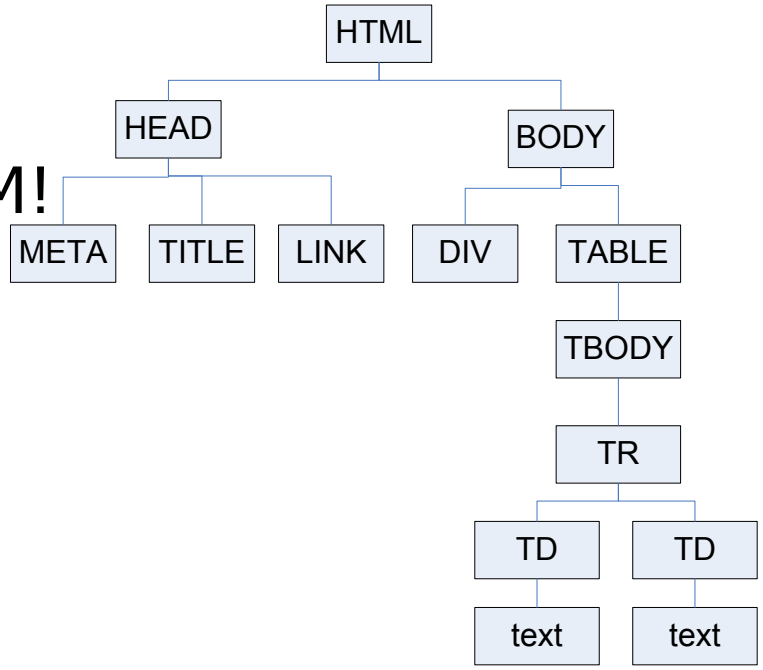
selector	action	parameters
`jQuery('p')`	`.css`	`('color', 'blue');`
`$('p')`	`.css`	`('color', 'blue');`

- *selector* is always `jQuery`, or it's alias `$`
- *Action* is one of the jQuery API commands
- *parameters* a list of 0, 1, or more arguments
  - Comma-separated
  - JSON-notation

The Document Object Model

What to select?

- Elements in the DOM!



Use CSS-syntax

to select elements

- elements/tags
- classes
- id’s

Always make sure the Page is ready

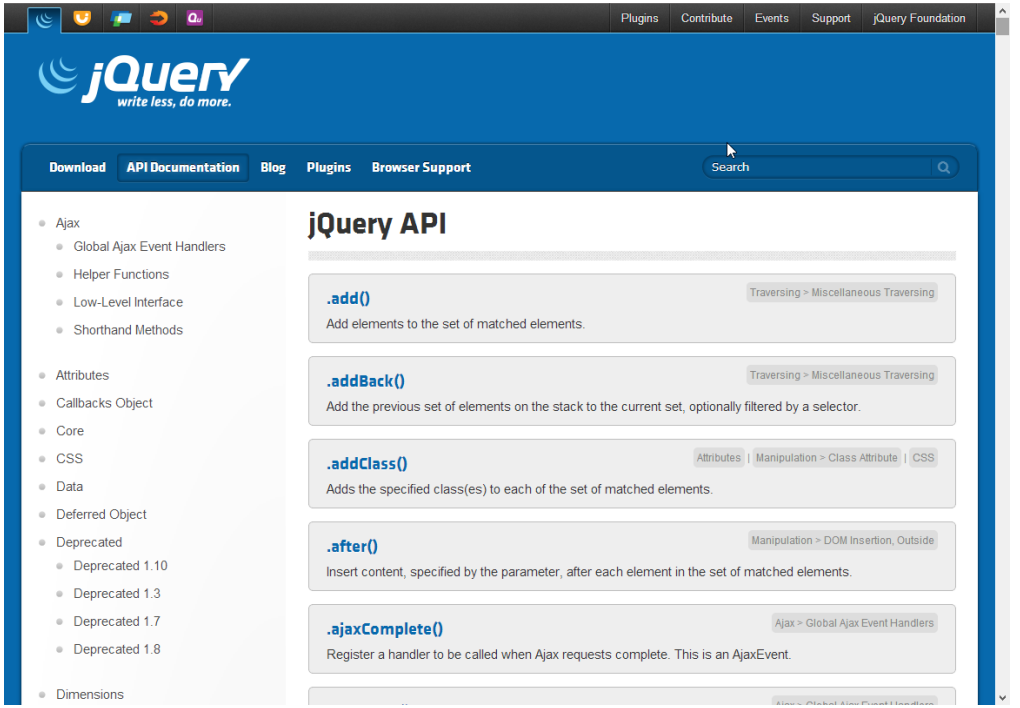
- All elements needs to be loaded before we can manipulate them
- Always use a document-ready function

```
$(document).ready(function() {
      alert('Document is ready…');
      // other functions, hooks, etc.
});
```

In plain JavaScript: `window.onload=function(){…}`

The jQuery API

- All options for `.action()` are described internet the *jQuery API*
- Learn and use!



http://api.jquery.com/

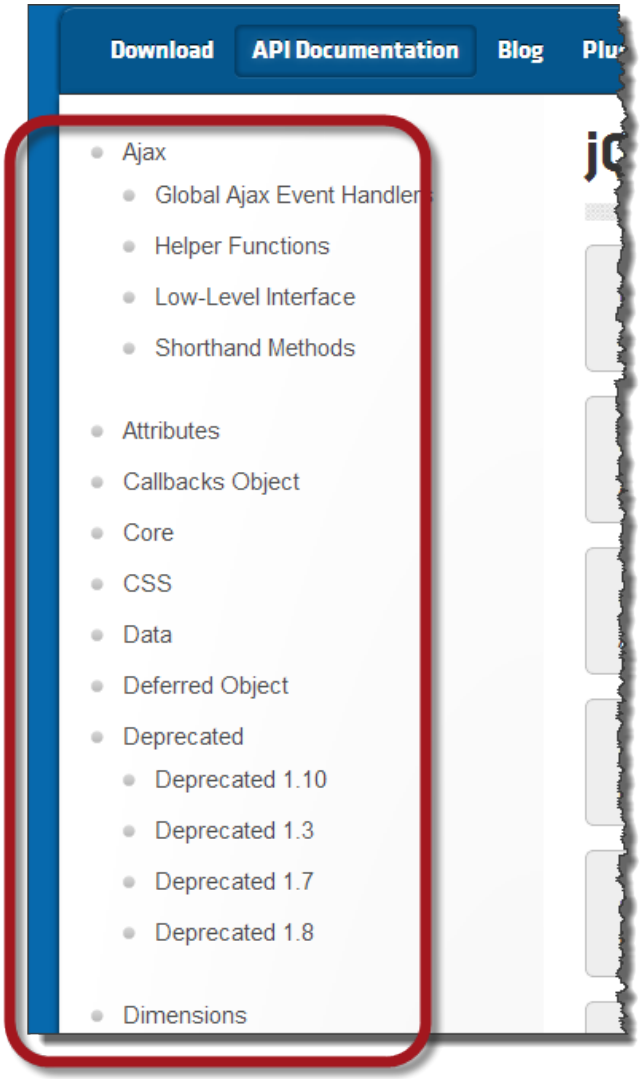
API categories

Most used:

- Ajax
- Attributes
- CSS
- Data
- DOM
- Effects
- Events
- Forms

Supportive

- Utilities
- Dimensions
- Offset
- …



Enkele complexe jQuery methods

Complexe selectie, .data(), .each() en .extend()

Complex selection

Select on attributes:

- `$("[href='#']")` →alle tags met een href-attribuut dat gelijk is aan #
- `$("[src$='.jpg']")` →alle tags met een src-attribuut eindigt op .jpg

Select on position within an element:

- `$("ul li:first")` →alle eerste li-tags in elke lijst
- `$("tr td:last")` →elke laatste cel in tabelrij
- `$("tr td:eq(2)")` →elke tweede cel binnen tabelrij
- `$("tr:even")` →elke even tabelrij
- `$("ul li:odd")` →elke oneven li-tag binnen een ul

Select form elements

- `$(":text")` →alle input elementen van het type tekst
- `$(":password")` →alle input elementen van het type password
- `$(':selected')` → alle elementen die geselecteerd zijn
- `$(':checked')` → alle selectievakjes die geselecteerd zijn
- `$(':visible')` → alle zichtbare elementen

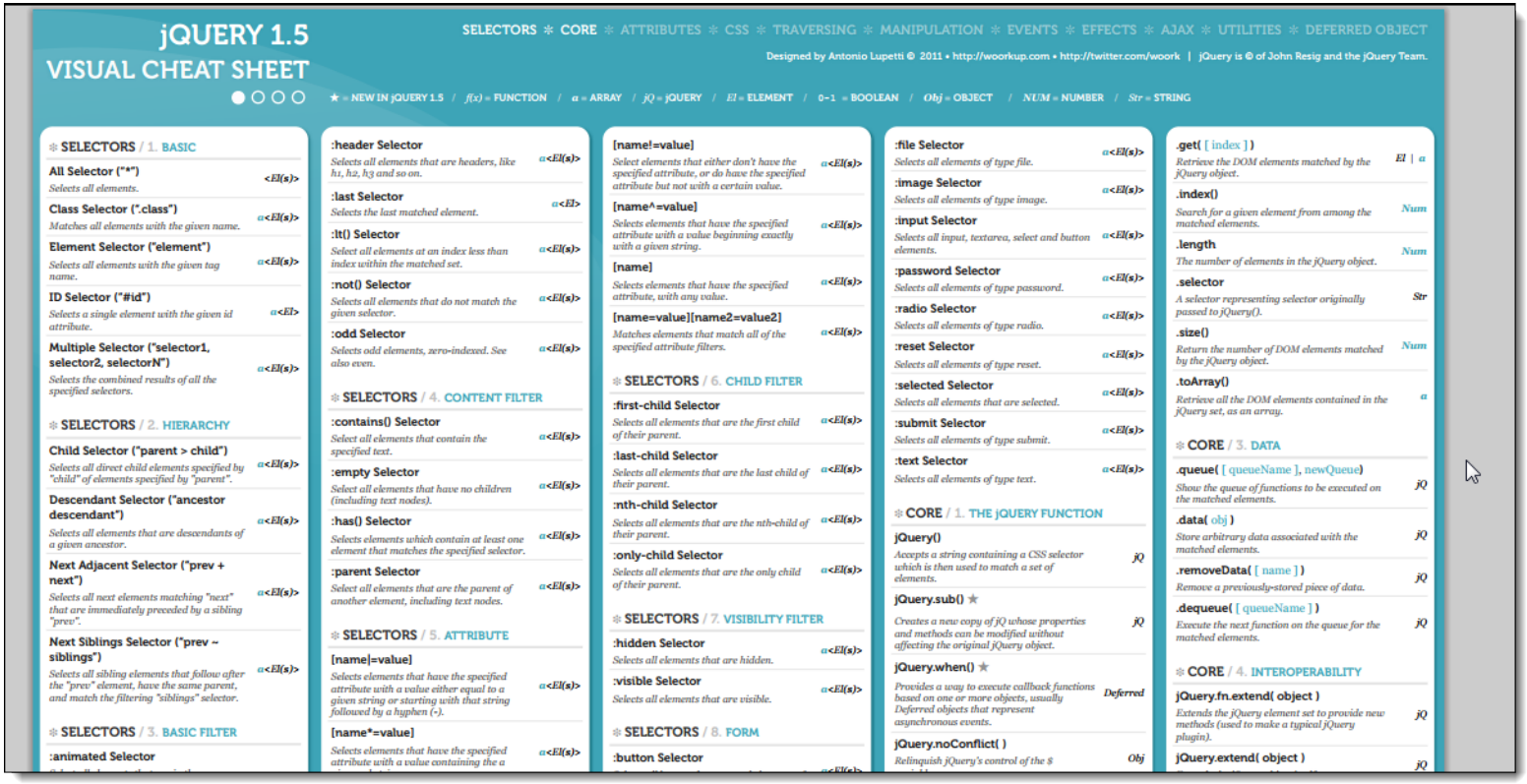
Selecting specific elements

- Child elements only
  - `$('#nav li > a');`
  - `$('#nav li').children();`
  - `$('#nav li').children('a'); // filter specific type of child`
- Terminologie:
  - `Descendant`: een compleet element, inclusief child elements van een parent element
  - `Child` : alleen het eerste child-element

```
<ul id="nav">
    <li><a href="#">Item 1</a></li>
    <li><a href="#"><span>Item 2</span></a></li>
    <li><a href="#">Item 3</a></li>
    <li><a href="#">Item 4</a></li>
    <li><a href="#">Item 5</a></li>
</ul>
```

Reference: cheat sheets

- zoek op trefwoord ‘jQuery Cheat Sheet’



jQUERY 1.5
VISUAL CHEAT SHEET

SELECTORS ✻ CORE ✻ ATTRIBUTES ✻ CSS ✻ TRAVERSING ✻ MANIPULATION ✻ EVENTS ✻ EFFECTS ✻ AJAX ✻ UTILITIES ✻ DEFERRED OBJECT

Designed by Antonio Lupetti © 2011 • http://woorkup.com • http://twitter.com/woork | jQuery is © of John Resig and the jQuery Team.

★ = NEW IN jQUERY 1.5 / f(x) = FUNCTION / a = ARRAY / jQ = jQUERY / El = ELEMENT / 0-1 = BOOLEAN / Obj = OBJECT / NUM = NUMBER / Str = STRING

✻ SELECTORS / 1. BASIC

All Selector ("*")
Selects all elements. <El(s)>

Class Selector (".class")
Matches all elements with the given name. a<El(s)>

Element Selector ("element")
Selects all elements with the given tag name. a<El(s)>

ID Selector ("#id")
Selects a single element with the given id attribute. a<El>

Multiple Selector ("selector1, selector2, selectorN")
Selects the combined results of all the specified selectors. a<El(s)>

✻ SELECTORS / 2. HIERARCHY

Child Selector ("parent > child")
Selects all direct child elements specified by "child" of elements specified by "parent". a<El(s)>

Descendant Selector ("ancestor descendant")
Selects all elements that are descendants of a given ancestor. a<El(s)>

Next Adjacent Selector ("prev + next")
Selects all next elements matching "next" that are immediately preceded by a sibling "prev". a<El(s)>

Next Siblings Selector ("prev ~ siblings")
Selects all sibling elements that follow after the "prev" element, have the same parent, and match the filtering "siblings" selector. a<El(s)>

✻ SELECTORS / 3. BASIC FILTER

:animated Selector

:header Selector
Selects all elements that are headers, like h1, h2, h3 and so on. a<El(s)>

:last Selector
Selects the last matched element. a<El>

:lt() Selector
Select all elements at an index less than index within the matched set. a<El(s)>

:not() Selector
Selects all elements that do not match the given selector. a<El(s)>

:odd Selector
Selects odd elements, zero-indexed. See also even. a<El(s)>

✻ SELECTORS / 4. CONTENT FILTER

:contains() Selector
Select all elements that contain the specified text. a<El(s)>

:empty Selector
Select all elements that have no children (including text nodes). a<El(s)>

:has() Selector
Selects elements which contain at least one element that matches the specified selector. a<El(s)>

:parent Selector
Select all elements that are the parent of another element, including text nodes. a<El(s)>

✻ SELECTORS / 5. ATTRIBUTE

[name|=value]
Selects elements that have the specified attribute with a value either equal to a given string or starting with that string followed by a hyphen (-). a<El(s)>

[name*=value]
Selects elements that have the specified attribute with a value containing the a a<El(s)>

[name!=value]
Select elements that either don't have the specified attribute, or do have the specified attribute but not with a certain value. a<El(s)>

[name^=value]
Selects elements that have the specified attribute with a value beginning exactly with a given string. a<El(s)>

[name]
Selects elements that have the specified attribute, with any value. a<El(s)>

[name=value][name2=value2]
Matches elements that match all of the specified attribute filters. a<El(s)>

✻ SELECTORS / 6. CHILD FILTER

:first-child Selector
Selects all elements that are the first child of their parent. a<El(s)>

:last-child Selector
Selects all elements that are the last child of their parent. a<El(s)>

:nth-child Selector
Selects all elements that are the nth-child of their parent. a<El(s)>

:only-child Selector
Selects all elements that are the only child of their parent. a<El(s)>

✻ SELECTORS / 7. VISIBILITY FILTER

:hidden Selector
Selects all elements that are hidden. a<El(s)>

:visible Selector
Selects all elements that are visible. a<El(s)>

✻ SELECTORS / 8. FORM

:button Selector

:file Selector
Selects all elements of type file. a<El(s)>

:image Selector
Selects all elements of type image. a<El(s)>

:input Selector
Selects all input, textarea, select and button elements. a<El(s)>

:password Selector
Selects all elements of type password. a<El(s)>

:radio Selector
Selects all elements of type radio. a<El(s)>

:reset Selector
Selects all elements of type reset. a<El(s)>

:selected Selector
Selects all elements that are selected. a<El(s)>

:submit Selector
Selects all elements of type submit. a<El(s)>

:text Selector
Selects all elements of type text. a<El(s)>

✻ CORE / 1. THE jQUERY FUNCTION

jQuery()
Accepts a string containing a CSS selector which is then used to match a set of elements. jQ

jQuery.sub() ★
Creates a new copy of jQ whose properties and methods can be modified without affecting the original jQuery object. jQ

jQuery.when() ★
Provides a way to execute callback functions based on one or more objects, usually Deferred objects that represent asynchronous events. Deferred

jQuery.noConflict( )
Relinquish jQuery's control of the $ Obj

.get([index])
Retrieve the DOM elements matched by the jQuery object. El | a

.index()
Search for a given element from among the matched elements. Num

.length
The number of elements in the jQuery object. Num

.selector
A selector representing selector originally passed to jQuery(). Str

.size()
Return the number of DOM elements matched by the jQuery object. Num

.toArray()
Retrieve all the DOM elements contained in the jQuery set, as an array. a

✻ CORE / 3. DATA

.queue([queueName], newQueue)
Show the queue of functions to be executed on the matched elements. jQ

.data(obj)
Store arbitrary data associated with the matched elements. jQ

.removeData([name])
Remove a previously-stored piece of data. jQ

.dequeue([queueName])
Execute the next function on the queue for the matched elements. jQ

✻ CORE / 4. INTEROPERABILITY

jQuery.fn.extend(object)
Extends the jQuery element set to provide new methods (used to make a typical jQuery plugin). jQ

jQuery.extend(object) jQ

De functie $().each()

- Een functie uitvoeren voor elk van de elementen in de selectie
- Nodig: als je *meerdere* opdrachten wilt gebruiken voor elementen binnen een set
- Notatie: `$('#selectie').each(function(i, el){…});`
- Parameters
  - `i`: index (nummer) van het element
  - `el`, het betreffende element waarop de `each` van toepassing is.
  - Dus: `el === this`, binnen de functie.

Bijvoorbeeld: hyperlinks selecteren → tooltip instellen → click()-event instellen → *en* speciale opmaak toekennen

```
// state van de accordion met categorieen herstellen
var categoryID = localStorage.getItem('categoryID');
if (categoryID) {
    $('#accordion h2').each(function () {
        if ($(this).data('categoryid') == categoryID) {
            $(this).click();
        }
    });
}
```

http://api.jquery.com/each/

De functie .data()

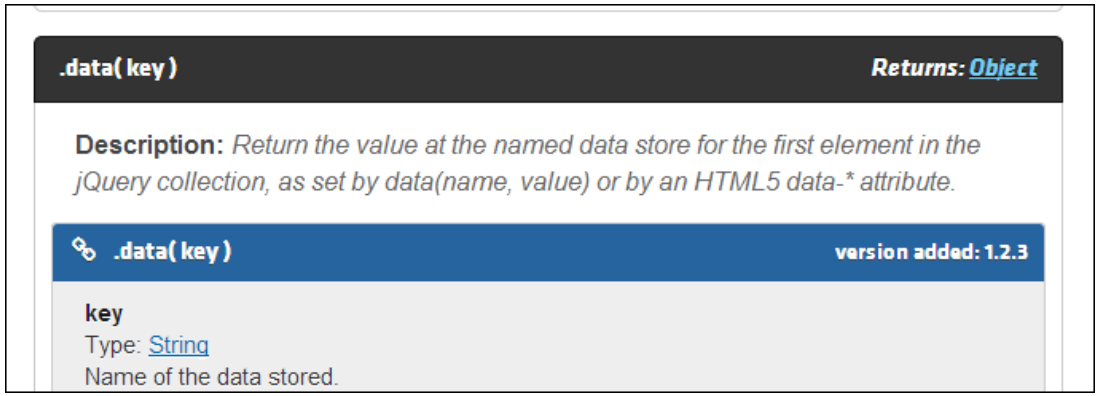
- Werkt samen met HTML5-attribuut `data="…"`
- Zowel getter als setter

- HTML:

```
<div data-customid="5">Data-div met ID 5</div>
```

Script

```
$('div').data('customid')
```



http://api.jquery.com/data/

De functie .extend()

- Inhoud van twee of meer objecten samenvoegen in het eerste object
- In de praktijk: vooral bij plug-ins, meegegeven configuratieobject verwerken

```
var fruit = {
    appel: 5,
    banaan: 10,
    peer: 3
}
var meerFruit = {
    kiwi: 13,
    mango: 8
}
```

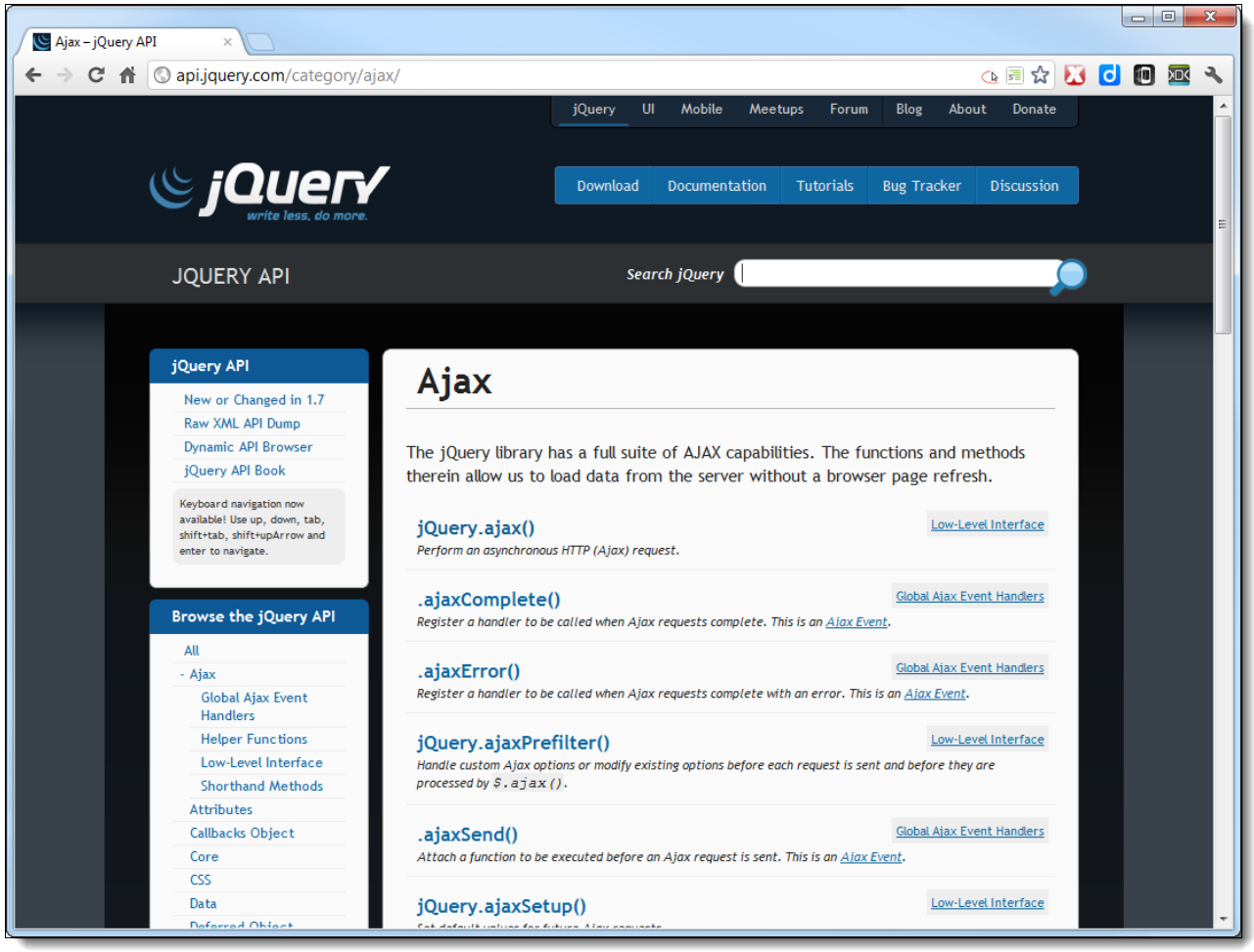
```
$.extend(fruit, meerFruit));
```

jQuery AJAX

AJAX-functies en deferred objects

AJAX-opties

http://api.jquery.com/category/ajax/

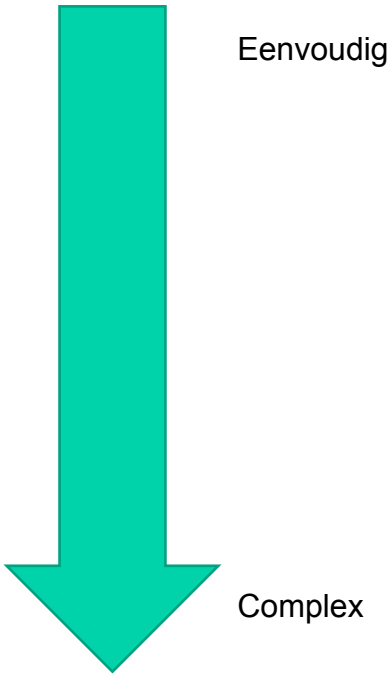


Ajax-methods in jQuery

- `.load()`
- Denk aan standaard jQuery-werkwijze:
  - Selecteer element met `$(…)`
  - roep method `.load()` aan
  - gebruik success callback-functie

Meer AJAX-functies

- `$.getJSON()`
- `$.get()`
- `$.post()`
- `$.ajax()`



$.getJSON()

Shorthand method voor ophalen van JSON-gegevens

Parameters: `url`, `data`, `callbackfunction`

```
// de klik op de knop omzetten in een Ajax-request
function handleClick(event) {
    // 1. variabelen instellen
    $.getJSON('ajaxData.json',  // dit is de URL waar de call naar toe gaat
            {},                 // Het object dat wordt meegestuurd (in dit geval leeg)
            function (data) {
                // Callback functie bij succes.
                // Over het object loopen en tonen in de UI
                console.log(data)
                for (var i in data) {
                    $('#myDiv').append('<p>' + i + ': <strong>' + data[i] + '</strong></p>');
                }
            });
}// einde function handleClick
```

$.get()

- Voert een standaard GET-request uit naar de server

```
// de klik op de knop omzetten in een Ajax-request
function handleClick(event){
    // 1. $.get() aanroepen. URL als parameter meegeven, callbackfunctie voor resultaat
    $.get('simple.html', function (result) {
        $('#myDiv').html(result);
    })
}
```

$.post()

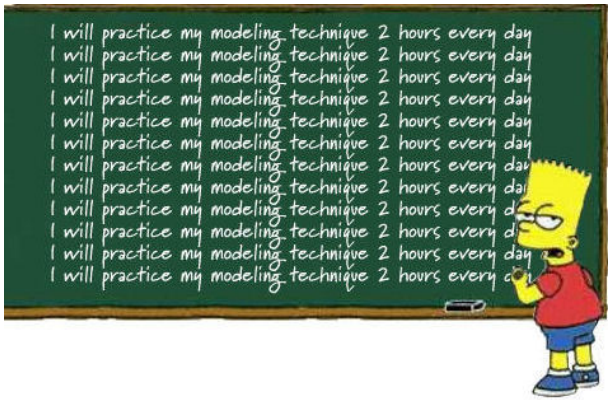
- Voert een standaard POST-request uit
- Vaak gebruikt voor meesturen gegevens

```
// de klik op de knop omzetten in een Ajax-request
function handleClick(event) {
    // 1. data-object maken van tekstveld
    var data = {};
    data.name = $('#txtName').val();
    // 2. $.post() aanroepen en data meegeven. Op de server moet iets v
    $.post('simple.html', data, function (result) {
        $('#myDiv').html(result);
    })
}
```

Kijk in de Developer tools voor details Network Request
(zoals `request-header` en `Form Data`)

Oefening

- Schrijf zelf jQuery `$.get()` en `$.post()`-opdrachten
- Retourneer eenvoudige tekst, of haal 'echte' data op.
- Kijk ook naar `$.getJSON()`
- Documentatie op http://api.jquery.com



Generic workhorse: $.ajax()

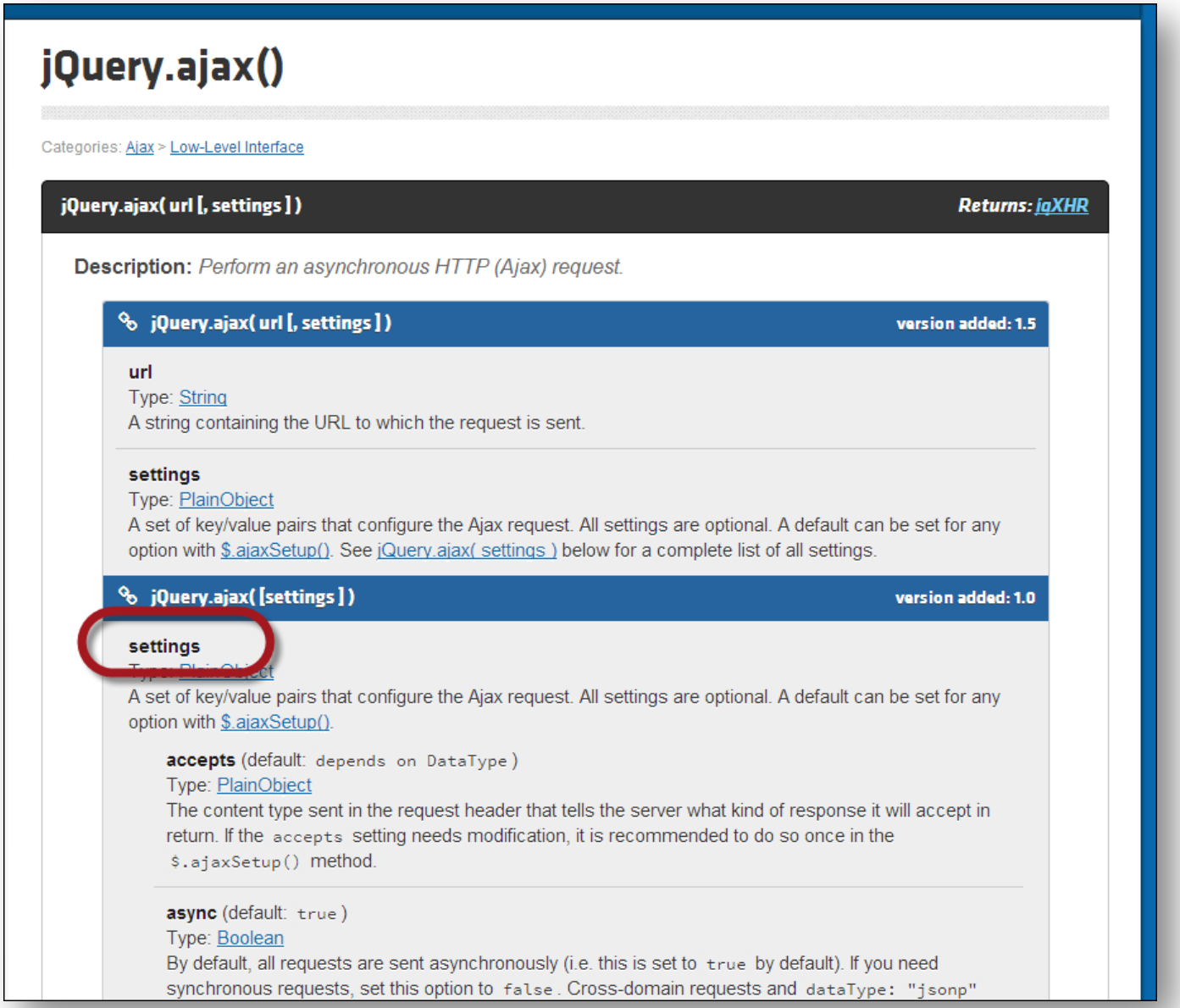
- Alle parameters zelf instellen
- Meeste flexibiliteit
- Configuratieobject meegeven

```
13  function handleClick(event){
14      // 1. variabelen instellen
15      $.ajax({
16          dataType: 'json',
17          contentType: 'application/json',
18          url : 'ajaxData.json',
19          success: handleData,    // success callbackfunctie
20          error : handleError     // error callbackfunctie
21      });
22  }
```

Parameters voor $.ajax()

`url` :	Adres waar call naar toe gaat
`type`:	GET- of POST-request
`contentType` :	Gegevenstype voor verzenden
`succes`:	Callback bij succes
`error`:	Callback bij optreden van fout
`beforesend` :	Functie voorafgaand aan verzenden
`data`:	De gegevens die worden meegezonden

Gebruikelijk: meezenden in configuratie-object {..}



jQuery.ajax()

Categories: Ajax > Low-Level Interface

jQuery.ajax(url [, settings]) *Returns: jqXHR*

Description: *Perform an asynchronous HTTP (Ajax) request.*

jQuery.ajax(url [, settings]) version added: 1.5

url
Type: String
A string containing the URL to which the request is sent.

settings
Type: PlainObject
A set of key/value pairs that configure the Ajax request. All settings are optional. A default can be set for any option with $.ajaxSetup(). See jQuery.ajax(settings) below for a complete list of all settings.

jQuery.ajax([settings]) version added: 1.0

settings
Type: PlainObject
A set of key/value pairs that configure the Ajax request. All settings are optional. A default can be set for any option with $.ajaxSetup().

accepts (default: `depends on DataType`)
Type: PlainObject
The content type sent in the request header that tells the server what kind of response it will accept in return. If the `accepts` setting needs modification, it is recommended to do so once in the `$.ajaxSetup()` method.

async (default: `true`)
Type: Boolean
By default, all requests are sent asynchronously (i.e. this is set to `true` by default). If you need synchronous requests, set this option to `false`. Cross-domain requests and `dataType: "jsonp"`

http://api.jquery.com/jQuery.ajax/

Ajax-instellingen instellen

Standaard Ajax-instellingen:

- `$.ajaxSetup();`

Basisinstellingen voor :

- `type (GET, POST)`
- `contentType`
- `error-callback`
- …

```
1  // Generieke instellingen voor jQuery- AJAX-calls
2  $.ajaxSetup({
3      type: 'POST',
4      contentType: 'application/json',
5      error: function (xhr, status) {
6          alert('Er is een jQuery/AJAX-fout opgetreden : ' + xhr.responseText);
7      }
8  });
9
```

http://api.jquery.com/jQuery.ajaxSetup/

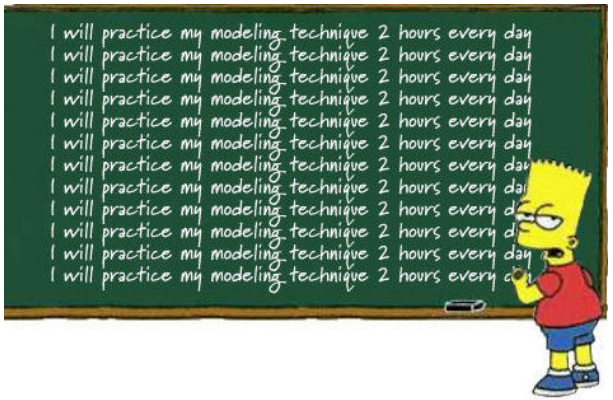
Jsonp via $.ajax()

- Voor Cross-domain calls
- Parameter: `dataType : 'jsonp'`
- Achter de schermen regelt jQuery juiste callback, etc.

```
12  // de klik op de knop omzetten in een Ajax-request
13  function handleClick(event){
14      // 1. variabelen instellen
15      $.ajax({
16          dataType: 'jsonp',
17          contentType: 'application/json',
18          url : 'http://api.yindo.com/api/214f032c-de61-4b86-87d2-8b1f7f5a7ca9/book/new/10/1/10',
19          success: handleData,    // success callbackfunctie
20          error : handleError     // error callbackfunctie
21      });
22  }
```

Oefening

- Schrijf zelf jQuery `$.ajax()`-opdrachten
- Retourneer eenvoudige tekst, of haal 'echte' data op.
- Gebruik het configuratie-object `url`, `dataType`, `success`, etc.

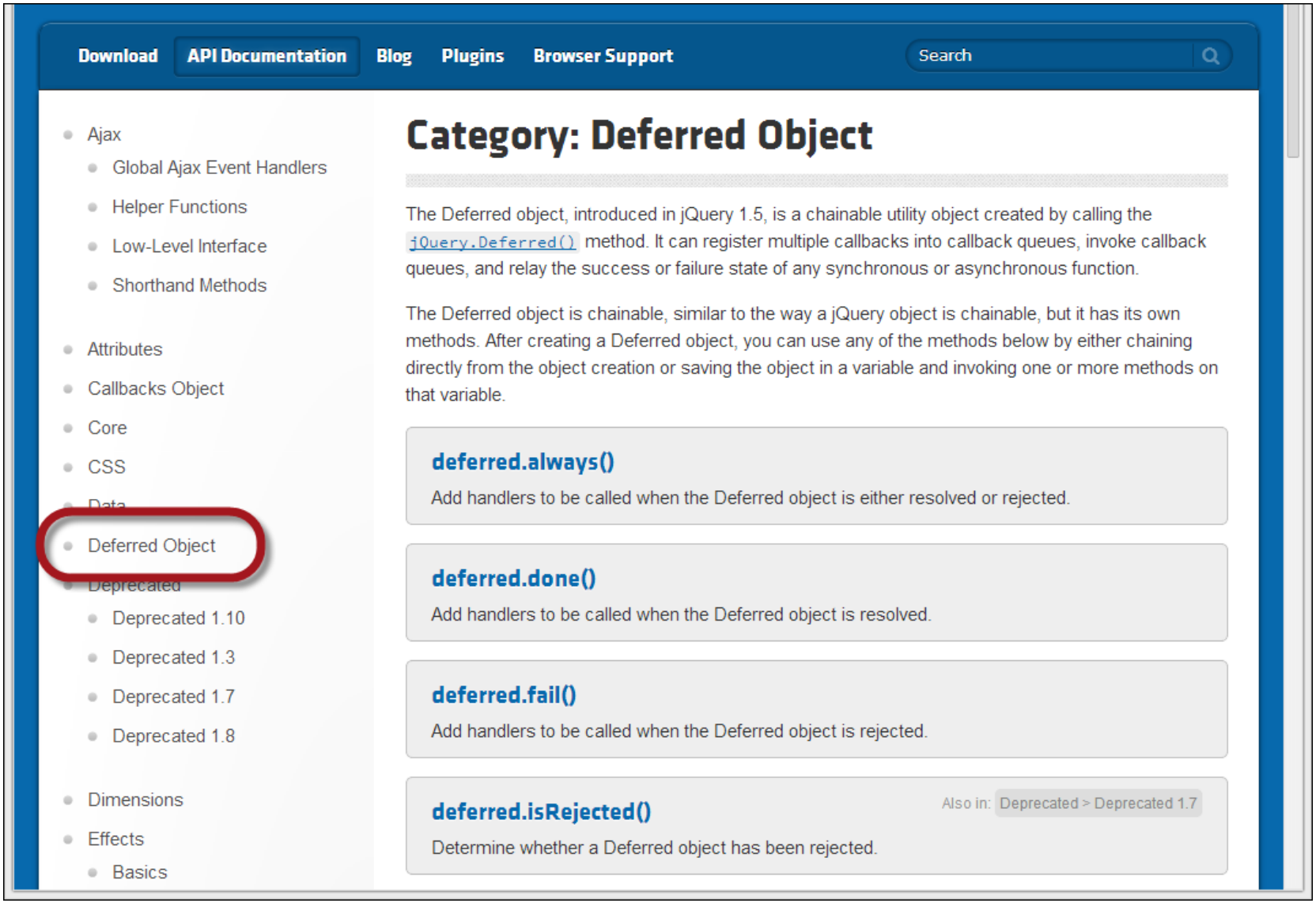


Defferred object

$.when() en $.then()

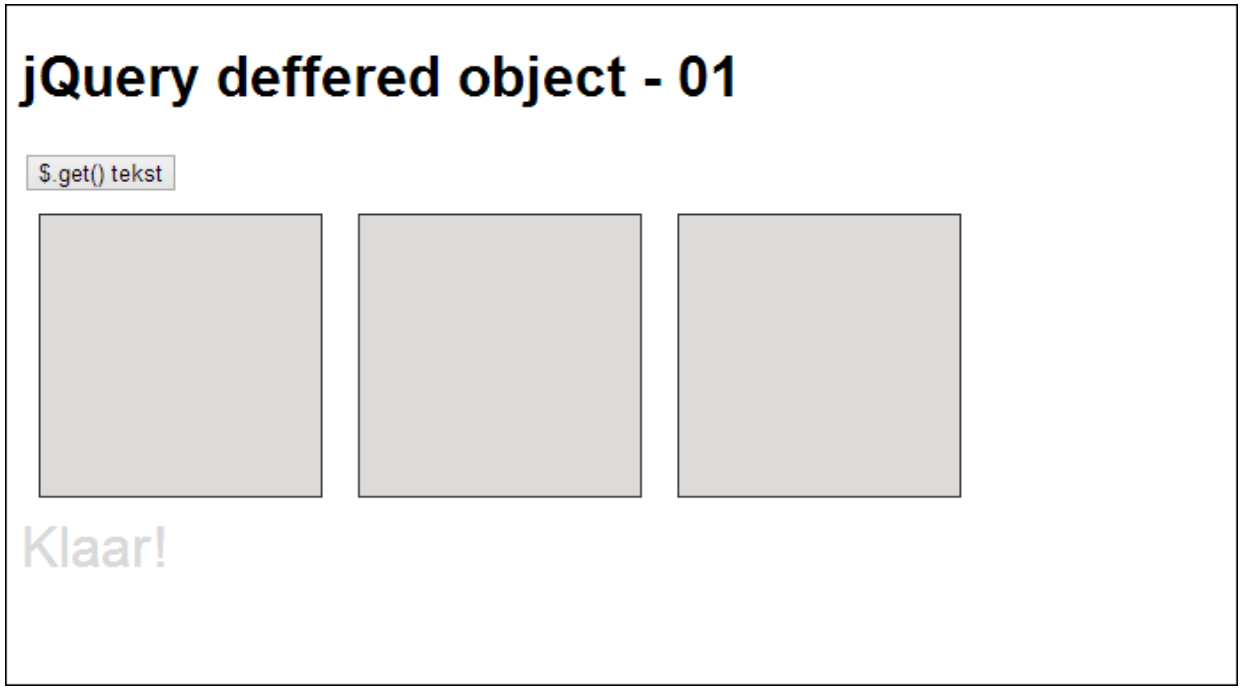
Deffered object

- Als een resultaat van meerdere ajax-calls afhankelijk is.
- Of: als meerdere ajax-calls in chain moeten worden uitgevoerd
- Doel: betere leesbaarheid, meer modulaire opbouw, betere foutafhandeling.
- Gebruik jQuery `$.when()` en `$.then()`
- jQuery 1.8+



http://api.jquery.com/category/deferred-object/

Case – 3 ajax calls



```
// Event handler
function handleClick(event) {
    // 1. $.get() aanroepen. URL als parameter meegeven, callbackfunctie voor resultaat
    $.get('simpleText.txt', function (result) {
        $('#myDiv1').html(result);
    });
    // 2. maar wat nu als een van de calls heel lang gaat duren... (simuleer met setTimeOut())
    setTimeout(function () {
        $.get('simpleText.txt', function (result) {
            $('#myDiv2').html(result);
        })
    }, 2500);
    // 3. derde call + updaten UI
    $.get('simpleText.txt', function (result) {
        $('#myDiv3').html(result);
        $(
    })
}
```

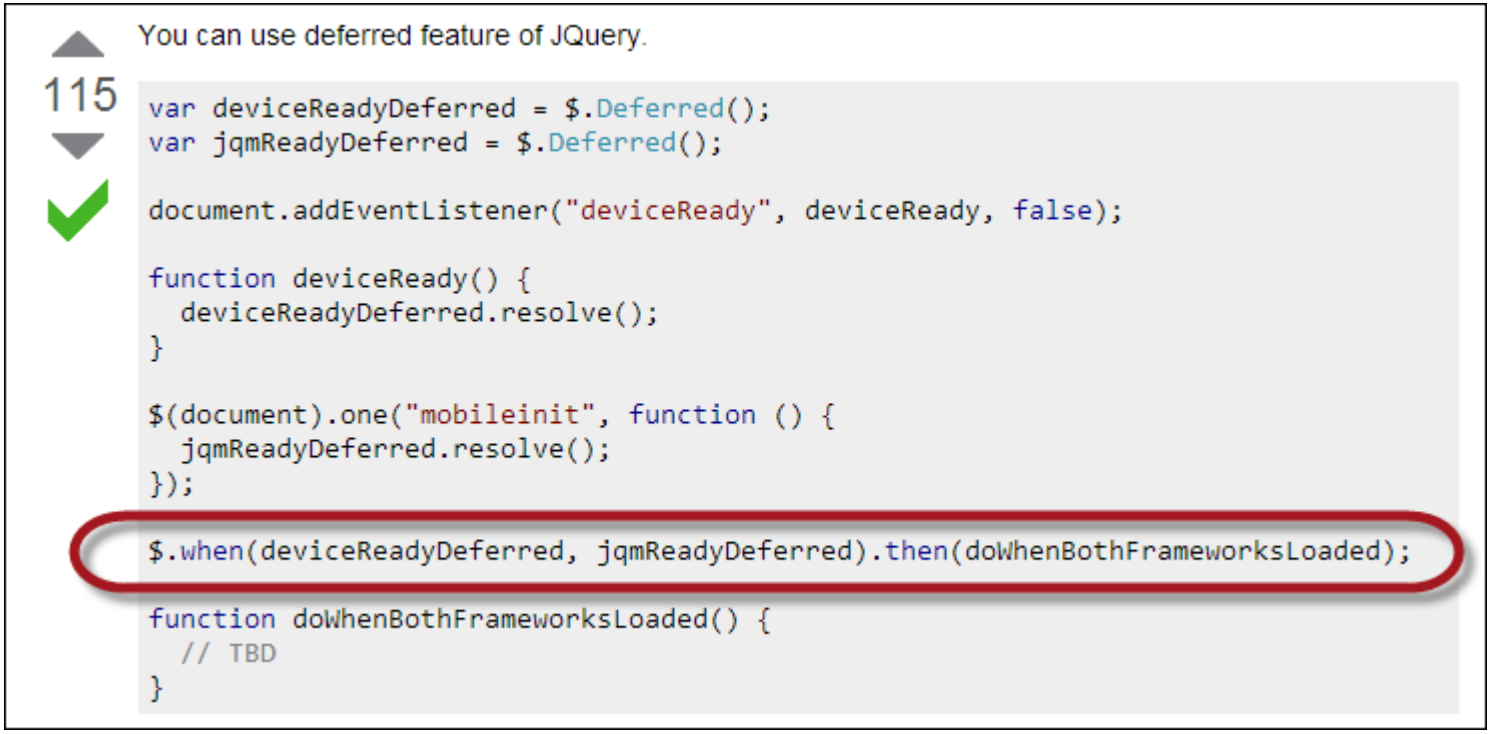
```
// Event handler
function handleClick(event) {
    // 1. $.get() aanroepen. URL als parameter meegeven, callbackfunctie voor resultaat
    $.get('simpleText.txt', function (result) {
        $('#myDiv1').html(result);
        // 2. twede call (traagheid gesimuleerd met setTimeOut())
        setTimeout(function () {
            $.get('simpleText.txt', function (result) {
                $('#myDiv2').html(result);
                // 3. derde call + updaten UI
                $.get('simpleText.txt', function (result) {
                    $('#myDiv3').html(result);
                    $('.result').addClass('klaar');
                })
            })
        }, 2500);
    });
}
```

Deferred Methods

- **Some methods available on deferred objects**
  - Return deferred object for a process flow
    - when
    - promise
  - Attach handler functions to deferred object
    - then
    - done
    - fail
    - progress
    - always
  - Change state of deferred object
    - resolve
    - reject

Ander voorbeeld van deferreds

Case – afhankelijkheid van meerdere events
http://stackoverflow.com/questions/10945643/correct-way-of-using-jquery-mobile-phonegap-together



You can use deferred feature of JQuery.

115

```
var deviceReadyDeferred = $.Deferred();
var jqmReadyDeferred = $.Deferred();

document.addEventListener("deviceReady", deviceReady, false);

function deviceReady() {
  deviceReadyDeferred.resolve();
}

$(document).one("mobileinit", function () {
  jqmReadyDeferred.resolve();
});

$.when(deviceReadyDeferred, jqmReadyDeferred).then(doWhenBothFrameworksLoaded);

function doWhenBothFrameworksLoaded() {
  // TBD
}
```

Oefening

- Leer werken met het deferred object
- Schrijf een chain ajax-calls en resolve deze met `.then()`
- Maak foutafhandeling met `.fail()`
- Documentatie:
  http://api.jquery.com/category/deferred-object/