|  |
| --- |
| Atos |
| Workshop Javascript |
| jQuery, jQueryUI, KnockoutJS |



|  |
| --- |
| Marco Franssen, Jan Saris, Sander van de Velde  23-9-2011 |



Inhoudsopgave

[1. Basic jQuery 3](#_Toc305055977)

[1.1 Create a MVC3 project 3](#_Toc305055978)

[1.2 How to get DOM elements 18](#_Toc305055979)

[1.3 How to get values from DOM elements 18](#_Toc305055980)

[1.4 How to set values on DOM elements 18](#_Toc305055981)

[2. Ajax 19](#_Toc305055982)

[2.1 What is JSON 20](#_Toc305055983)

[2.2 A Method with a Json result 20](#_Toc305055984)

[2.3 Make an Ajax call 20](#_Toc305055985)

[2.4 Passing parameters 20](#_Toc305055986)

[2.5 Update the page with the Json result 20](#_Toc305055987)

[2.6 Use fiddler to test your call 21](#_Toc305055988)

[3. jQuery UI 22](#_Toc305055989)

[3.1 Dialog 22](#_Toc305055990)

[3.2 Datepicker 22](#_Toc305055991)

[3.3 Buttons 22](#_Toc305055992)

[3.3.1 Checkbox buttons 22](#_Toc305055993)

[3.3.2 Radio buttons 22](#_Toc305055994)

[3.3.3 Group buttons 22](#_Toc305055995)

[3.4 Animations / effects 22](#_Toc305055996)

[3.5 Create control sets 22](#_Toc305055997)

[3.6 Switching themes 22](#_Toc305055998)

[4. Advanced Javascript 23](#_Toc305055999)

[4.1 Create your own methods 23](#_Toc305056000)

[4.2 Create a javascript object 24](#_Toc305056001)

[4.3 Communicate between objects with events 24](#_Toc305056002)

[5. MVVM met Knockout JS 25](#_Toc305056003)

[5.1 Simple bindings 25](#_Toc305056004)

[5.2 Advanced bindings 25](#_Toc305056005)

[5.3 Cascading dropdown 25](#_Toc305056006)

[5.4 Animations with jQueryUI 25](#_Toc305056007)

# Basic jQuery

Welkom bij deze introductie op jQuery. Dit hoofdstuk zal in het tekenstaan van browser DOM manipulatie. Voor diegene die niet weet war de browser DOM is, dis is een afkorting voor Document Object Model, oftewel een boomstructuur van alle html elementen op het scherm. Deze elementen kunnen genest zijn.

De Leerdoelen van deze jQuery Basis zijn:

* 1. Een kaal MVC3 project aanmaken met daarin eenvoudige HTML code
  2. Hoe jquery op te nemen in je pagina (Shared/\_Layout.cshtml)
  3. Een apart javascript bestand toevoegen
     1. jQuery binnen VS2010 (aparte js bestanden voor debuggen; vsdoc)
  4. De .ready (tevens kennismaking met functie notatie)
  5. Reageren op events: click
  6. Selectors om enkele objecten te veranderen
  7. Lezen van attributen, schrijven van attibuten
  8. CSS wijzigingen

## Create a MVC3 project

Open een Kaal mvc3 project genaamd JqueryWorkShopMvcApplication aan in VS2010 (met razor en HTML5 ondersteuning)

We kunnen jQuery met de hand ophalen van de website (<http://docs.jquery.com/Downloading_jQuery>) maar we kunnen ook Nuget gebruiken. Maar standaard is er al een (oudere) versie van jQuery aanwezig in een MVC3 project. Die passen we toe.

Controleer dus dat de \_Layout.cshtml (de masterpage equivalent in Razor) (in \Views\Shared) met de juiste referenties gevuld is.

|  |
| --- |
| <!DOCTYPE html> <html> <head>     <meta charset="utf-8" />     <title>@ViewBag.Title</title>     <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />     <script src="@Url.Content("~/Scripts/jquery-1.5.1.min.js")" type="text/javascript"></script> </head>   <body>     @RenderBody() </body> </html> |

**Tip1**: de standaard aanwezige modernzr javascript bibliotheek is weggelaten (buiten scope voor onze workshop)

**Tip2**: Eventueel kan /Scripts/jquery-1.5.1.js gerefereerd worden. Dit bestand met exact dezelfde logica is niet geminified. Dwz. De minified versie is geoptimaliseerd voor het internet door deze zo klein mogelijk te maken (commentaar verwijderd, functienamen verkort, etc.) De niet-minified versie is prettiger bij javascript debug werkzaamheden.

Voeg een HomeController toe (rechtermuisclick op de de controllers map)

Noem deze HomeController



Zie dat er een Index methode aanwezigi is. Dit is een MVC action.

Maar een View aan voor deze action (rechtermuisclick op de methode)



Gebruik de standaard instellingen:

De View is aangemaakt:

|  |
| --- |
| @{     ViewBag.Title = "Index"; }   <h2>Index</h2> |

Maak een home.index.js aan in de scripts map via add New item. Hier gaan we de jQuery statements in schrijven, deze code is namelijk te debuggen via breakpoints in de ‘gutter’.





Het javaScript is nu geopend in de Editor. Sleep nu de jquery-1.5.1-vsdoc.js in de text editor . Er komt regel ‘commentaar’ beschikbaar:

|  |
| --- |
| /// <reference path="jquery-1.5.1-vsdoc.js" /> |

Dit maakt het voor VS2010 mogelijk om code inside aan te bieden voor jQuery. Erg handig.

Voeg een lege functie toe:

|  |
| --- |
| /// <reference path="jquery-1.5.1-vsdoc.js" />   function ExecuteOnStartup() {   // Add your script here } |

Refereer nu vanuit de index.cshtml naar het javascript bestand:

|  |
| --- |
| @{     ViewBag.Title = "Index";             <script src="@Url.Content("~/Scripts/home.index.js")" type="text/javascript"></script> }   <h2>Index</h2> |

Alles staat nu klaar om met de workshop te beginnen. Compileer even om te controleren of alles werkt ;-)

Dit moet de volgende pagina geven:



De gegenereerde html is eenvoudige maar duidelijke html 5 opmaak.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <title>Index</title>  <link href="/Content/Site.css" rel="stylesheet" type="text/css" />  <script src="/Scripts/jquery-1.5.1.js" type="text/javascript"></script>  </head>    <body>  <script src="/Scripts/home.index.js" type="text/javascript"></script>    <h2>Index</h2>    </body>  </html> |

Deze is gegenereerd door de combinatie van \_Layout.cshtml en de index.cshtml

Voeg nu een aantal html objecten en een stukje javascript toe aan de index.cshtml:

|  |
| --- |
| @{     ViewBag.Title = "Index";             <script src="@Url.Content("~/Scripts/home.index.js")" type="text/javascript"></script> }   <h2>Index</h2>   <a id="link1" href="">Link1</a>  <br />  <a id="link2" href="">Link2</a>   <script type="text/javascript">     $(document).ready(function () {         ExecuteOnStartup();     }); </script> |

We zien hier twee links zonder echte logica (er zal niks gebeuren als je er op klikt).

Het stukje script zal uitgevoerd worden als deze pagina in de browser laadt. De opdracht daarin is niks anders dan een stukje jQuery code wat zegt:

“Zodra de pagina geheel in de browser geladen is en de gebruiker bijna de eerste controls kan gaan aanklikken, dan moet de ExecuteOnStartup methode uitgevoerd worden”.

Dit is essentieel! De .ready(); gaat pas af op dat moment af en dit kan per merk browser verschillen! jQuery lost specifieke browser afhankelijkheden voor ons op.

We passen de ExecuteOnStartup() iets aan:

|  |
| --- |
| function ExecuteOnStartup() {       // add to all links (the a elements) a click function     $("a").click(function () {         alert("Hello jQuery workshop (from " + this.id + ")");     }); } |

Hier zien we het eerste voorbeeld van een jQuery selector. Voor alle elementen die voldoen aan de eis dat ze simpelweg een link zijn, wordt aan hun click event een functie toegevoegd. Hierin wordt een dialoogje getoond met daarin een tekst en de id van het element waarop geclicked is.

Dit ziet er dan zo uit:



We klikken op ieder element:

|  |  |
| --- | --- |
| De H2 | Er gebeurt niks |
| De eerste link |  |
| De tweede link |  |

Iedere link voerst dus zijn eigen functie uit. Een selector kan dus 0, 1 of meerdere elementen opleveren waar iets mee gedaan moet worden.

jQuery bevat vele varianten op selectors. Er staat een uitputtende lijst op <http://api.jquery.com/category/selectors/>

We kunnen ook functies aan andere DOM element events hangen. Zie ook <http://api.jquery.com/category/events/>

Hier gaan we dieper op in. We passen dus de ExecuteOnStartup weer iets aan:

|  |
| --- |
| function ExecuteOnStartup() {       // add to all links (the a elements) a click function     $("#link1").mouseenter(function () {         $("#link2").hide(500);     });       $("#link1").mouseleave(function () {         $("#link2").show(1000);     }); } |

Wat we nu zien is data an het specifieke element link1 (en alleen deze; dat komt door het # wat jQuery vertelt alleen naar elementen met deze id te zoeken) maar liefst twee functies te koppelen.

Als de muis naar link1 gaat, dan wordt specifiek link2 onzichtbaar gemaakt. Dit wordt uitgevoerd in 500 milliseconden en dit geeft dus een animatie. Als de muis link1 weer verlaat, dan moet link2 weer zichtbaar gemaakt worden in 1000 milliseconden. Ook dit geeft dus een animatie.

Probeer het maar eens:



Zie dat de tweede link langzamer terugkomt dan verdwijnt.

We zien hier twee belangrijke aanwinsten met jQuery. De selectors maken het veel eenvoudiger om specifieke elementen te vinden, zelfs als deze genest zijn zoals rijen in een tabel. En jQuery beidt simpele animaties wat een prettige beleving van de pagina voor de gebruiker is.

Er zijn flink wat effecten mogelijk. Kijk voor een uitputtende lijst op <http://api.jquery.com/category/effects/>

Overigens, wie een breakpoint in de [home.index.js](http://home.index.js) heeft opgenomen, ziet dat VS2010 netjes stopt op deze breakpoint:



Met bv. het Immeadiate scherm is de inhoud van de javascript elementen te acherhalen.

Laten we eens wat meer proberen met de selectors. Voeg eerst een aantal paragraph elementen toe achter de a elementen:

|  |
| --- |
| ...  <p>Hello</p>  <p>and</p>  <p>Goodbye</p>  ... |

En we maken een stylesheet bestand aan in de Content map:



Noem deze home.index.css



En voeg deze toe aan de index.cshtml:

|  |
| --- |
| @{     ViewBag.Title = "Index";             <script src="@Url.Content("~/Scripts/home.index.js")" type="text/javascript"></script>     <link href="@Url.Content("~/Content/home.index.css")" rel="stylesheet" type="text/css" /> }   <h2>Index</h2>   <a id="link1" href="">Link1</a> <br /> <a id="link2" href="">Link2</a>   <p>Hello</p>  <p>and</p>  <p>Goodbye</p>    <script type="text/javascript">     $(document).ready(function () {         ExecuteOnStartup();     }); </script> |

In de css moeten de volgende ‘styles’ toegevoegd worden:

|  |
| --- |
| p { margin: 8px; font-size:16px; }       .selected { color:blue; } |

En aan de ExecuteOnStartup voegen we de volgende regel toe:

|  |
| --- |
| …  var lastParagraph = $("p:last"); alert("the text within the last paragraph was:" + lastParagraph.text()); lastParagraph.addClass("selected").text("BYE BYE");  … |

Voer dit eens uit. Eerst volgt er een alert:



Dan volgt de volgende weergave in de browser:



Wat is er gebeurd? Deze volgende situatie is ontstaan:

* Alle laatste paragrafen (eentje dus) worden door een variabele gerepresenteerd.
* We geven een alert af met de inhoud van de laatste paragraaf
* Alle drie de paragrafen hebben een andere marge en lettergrootte gekregen (dit is standaard css gedrag)
* En we hebben gecodeerd dat de LAATSTE paragaaf de extra selected stijl heeft gekregen
* En we hebben de tekst in de paragraaf overschreven

Tip: Zag je dat we in de laatste regel dezelfde paragraaf twee maal manipuleerden? Alle functies (zoals de .addClass(“…”) en de .text(“…”)) manipuleren de aangeboden lijst van objecten en geven die daarna weer terug. De elementen die dus door de .addClass zijn gemanipuleerd, worden dus ook door de .text gemanipuleerd. Dit noemen we een *Fluent* notatie en dat zijn we ook terug bij bv. LinQ en EF 4.1 Code First.

De .ready() is was dus ook al een selector, wat grappig.

We kunnen dus met jQuery ook de attributen van (meerdere of afzonderlijke) elementen uitlezen en aanpassen. En we kunnen de CSS stijlen uitlezen en aanpassen.

Het is zelfs mogelijk op te selecteren op het wel of niet hebben van een bepaalde CSS stijl. Dit is erg krachting. Zo kunnen heel snel afzondelijke elementen een nieuwe CSS look krijgen.

Laten we nog iets meer met selectors spelen. Voeg aan de index.cshtml een tabel toe met een kolomnaam en view regels

|  |
| --- |
| <table border="1">  <thead>     <tr><td>Header</td></tr>  </thead> <tbody>     <tr><td>Row with Index #0</td></tr>      <tr><td>Row with Index #1</td></tr>      <tr><td>Row with Index #2</td></tr>      <tr><td>Row with Index #3</td></tr>  </tbody> </table> |

Voeg wederom aan de ExecuteOnStartup wat javascript toe:

|  |
| --- |
| …  $("table thead tr:even").css("background-color", "#bb00ff");   $("table tbody tr:even").css("background-color", "#bbbbff");  … |

En we hebben nog wat extra styling nodig voor de table in de home.index.css :

|  |
| --- |
| …  table {    background:#eeeeee;  }  … |

Dit geeft de volgende tabel invulling:



Wat is er gebeurd:

* De totale tabel heeft een achter achtergrond kleurtje gekregen (standaard css gedrag)
* De even regels in de tabel head krijgen een ander kleurtje
* De even regels in de tabel body krijgen een ander kleurtje

We kunnen dus bij de selector ook specifiek genest gaan zoeken. Dit heeft de voorkeur bij grote documenten met veel DOM elementen waar slechts enkele objecten moeten worden aangepast. Het zoeken naar elementen is gewoon het aflopen van eigenlijk alle elementen en dit kost tijd en energie (het verbruikt tenslotte stroom van je PDA en het is toch jammer als merkbaar minder kunt bellen na het bezoek van een website).

Overigens zal de header row NIET gemanipuleerd worden als de de oneven regels kleuren in de header:

|  |
| --- |
| $("thead tr:odd").css("background-color", "#bb00ff"); |

Dit geeft:



De regel met daarin “Header” is de nulde regel, dus een even regel, niet een oneven regel…

Conclusie met slechts enkele regels jQuery code is het heeft eenvoudig geworden om een aantrekkelijke (..) pagina te creeeren. Had jij dit ook met standaard javascript kunnen schrijven en had die code ook in alle browsers gewerkt?

Overigens is De DOM, dus de boomstructuur aan elementen zoals de browser die zeit, ook in te zien in moderne browsers of via plugins zoals Firebug:



Wil je meer weten over bovenstaande DOM manipulatie? Kijk dan eens op <http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery>

Bonus opdracht:

|  |
| --- |
| Voeg een checkbox toe aan de pagina en als ik die van waarde verander, toon of verberg dan de tabel. (hints: input, type , checkbox, .change() .toggle()) |

## How to get DOM elements

Uitleg over de $(‘#id’) en $(‘.class’) etc.

## How to get values from DOM elements

## How to set values on DOM elements

# Ajax

DE JSONRESULT (todo lijstje meegeven)

public class HomeController : Controller

{

public ActionResult Index()

{

return View();

}

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult DoSomething(string name, string phone,

int zip, string email)

{

var myJsonResult = new MyJsonResult();

try

{

// Do something

Thread.Sleep(2000);

// throw new ArgumentException("boo");

myJsonResult.Succeeded = true;

}

catch (Exception)

{

myJsonResult.Succeeded = false;

}

return new JsonResult { Data = myJsonResult };

}

}

public class MyJsonResult

{

public bool Succeeded { get; set; }

}

// ajax call

// Attach an event when starting screen

$('#btnStartCallback').bind(clickEvent, function (e) {

var name = $('#name').val() + "";

var phone = $('#phone').val() + "";

var zip = $('#zip').val() + 0;

var email = $('#email').val() + "";

var dataToSend = { 'name': name, 'phone': phone,

'zip': zip, 'email': email };

var url = "@Url.Action("DoSomething", "Home")";

$.ajax({

url: url,

data: dataToSend,

dataType: 'json',

type: 'POST',

beforeSend: function () {

// before callback, show the waiting animation

jqt.goTo('#waiting', 'dissolve');

},

success: function (result) {

if (result.Succeeded) {

// after a successfull callback

// TODO vul tabel $('#errorInfo').text("Invalid input");

}

else {

// after an unsuccessful callback

$('#errorInfo').text("Invalid input");

}

},

error: function (xhr, err) {

// after an error

$('#errorInfo').text("Status: " + xhr.status);

}

});

Extra uitleg: zie hoe via "@Url.Action("DoSomething", "Home")"; de url netjes wordt uitgeschreven naar een volledig pad. Dit blijft ook werken als de code op de server (lees: andere machine) wordt uitgerold.

Extra uitleg: er zijn verkorte schrijfwijzen bv. .get en .post maar bovenstaande legt goed uit wat er gebeurd.

Extra opdracht doe ook eens een GET

## What is JSON

## A Method with a Json result

## Make an Ajax call

## Passing parameters

## Update the page with the Json result

## Use fiddler to test your call

# jQuery UI

We weten inmiddels hoe de basis van jQuery werkt. We kunnen elementen aanpassen en al wat gegevens ophalen van een server. Nu wordt het tijd voor het echte werk.

We gaan controls gebruiken die op die in de browser draaien. We willen tenslotte het gevoel krijgen alsof we met een echte windows applicatie aan het werken zijn.

We beginnen ook hier weer met de basis. Wat is een dedicated Windows applicatie zonder Error box???

## Scripts

jQuery heeft al een heleboel werk voor je gedaan en biedt daarom al verschillende controls aan. Hiervoor moeten we natuurlijk wel eerst een javascript file en wat css inlezen.

Laad het script en css in door binnen @section Head de volgende regels op te nemen:

|  |
| --- |
| <link href="@Url.Content("~/Content/themes/base/jquery.ui.all.css")"rel="stylesheet" type="text/css" />  <script src="@Url.Content("~/Scripts/jquery-ui-1.8.16.min.js")" type="text/javascript"></script> |

We hebben nu de jQuery UI library en de bijbehorende CSS ingelezen. Nu kunnen we echt aan de slag!

In de code zag je al dat we het javascript “jQueryUI.js” al hadden toegevoegd. Hier gaan wij zo meteen in werken. Bekijk eens de eerste paar regels, waar we het starten van het script iets verkort opschrijven.

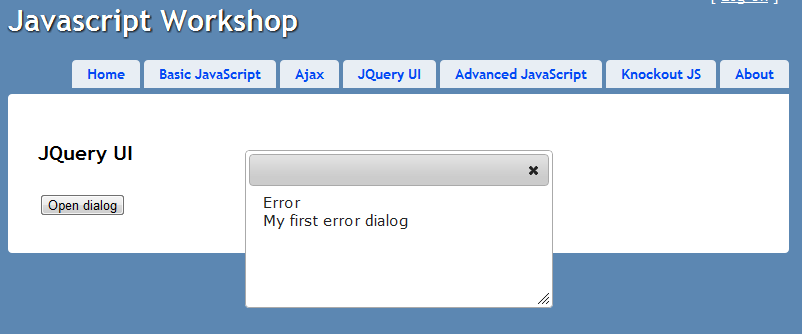
## Dialog

Ok, laten we nu eens onze pagina interactief en mooi gaan maken!  
We gaan een <div> element omtoveren tot een dialog-box die je op je scherm gaat zien.

De html is al voorbereid zoals je had gezien.   
Ga nu naar het javascript bestand (jQueryUI.js) en voeg aan de functie: ExecuteOnStartup de volgende regel toe:

|  |
| --- |
| $('#myDialog').dialog(); |

  
**Start de applicatie en bekijk het resultaat:**



Dankzij de UI library van jQuery hebben we nu een standaard dialog plugin tot onze beschikking, die van een element op je pagina een dialog maakt. Dit alles door alleen de dialog() functie aan te roepen.

Maar je zag al dat het dialog erg kaal was. Zo had het dialog geen titel en misten we de “Ok” knop voor onze foutmelding te sluiten. En het dialog ging automatisch open, dit willen we natuurlijk zelf doen.

Met andere woorden, we willen onze dialog configureren.  
Maak een config object aan en geef deze mee aan de initialisatie van het dialog.  
Dit doe je door de volgende code boven de intialisatie van het dialog te zetten:

|  |
| --- |
| var config = {          autoOpen: false,          resizable: false,          draggable: false,          modal: true,  height: 500,        title: 'My dialog title',          buttons: { Ok: function () { $(this).dialog("close"); } }      }; |

  
We hebben nu een config object aangemaakt met een aantal properties die allemaal redelijk voor zichzelf spreken. De property “buttons” bevat weer een nieuw object met properties. Als je gaat debuggen zul je dit ook zien. In het geval van de “ok” button hebben we er geen primitieve waarde maar een hele functie aan toegekend. Deze wordt uitgevoerd zodra op de “ok” button wordt geklikt.  
  
We gaan nu eerst de config meegeven aan de initialisatie van het dialog en we koppelen het openen van het dialog aan de button click van de button die op het scherm staat.

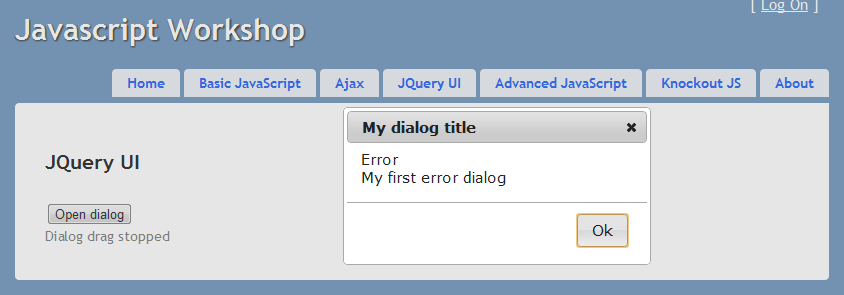
|  |
| --- |
| $('#myDialog').dialog(config); $('#openDialogButton').click(function () { $('#myDialog').dialog('open'); }); |

  
Een dialog genereert ook verschillende events. Hier kun je je op abonneren en daar je voordeel uit halen. We gaan in dit geval de status van het dialog laten zien op de pagina net onder de button.

Voeg nu de volgende regels code toe om de events af te vangen:

|  |
| --- |
| $('#myDialog').bind('dialogopen', function (){ $('#dialogState').html('Dialog opened'); });  $('#myDialog').bind('dialogdragstart',function() { $('#dialogState').html('Dialog drag started'); });  $('#myDialog').bind('dialogdragstop', function() { $('#dialogState').html('Dialog drag stopped'); });  $('#myDialog').bind('dialogclose', function () { $('#dialogState').html('Dialog closed'); }); |

  
 **Start de applicatie en bekijk het resultaat:**

  
  
**Opschoon tips:**  
Nu abonneren we ons na de initialisatie pas op de events, maar zoals je bij de ok button al had gezien kun je het ook gewoon in de config opnemen. De configuratie zou er dan als volgt uit komen te zien:

|  |
| --- |
| var config = {          autoOpen: false,          resizable: false,          draggable: true,          modal: true,          title: 'My dialog title',          buttons: { Ok: function () { $(this).dialog("close"); } },          open: function () { $('#dialogState').html('Dialog opened'); },          close: function () { $('#dialogState').html('Dialog closed'); },          dragStart: function () { $('#dialogState').html('Dialog drag started'); },          dragStop: function () { $('#dialogState').html('Dialog drag stopped'); }      }; |

**Extra info:**  
Voor een mooi overzicht van alle functionaliteit zie: <http://jqueryui.com/demos/dialog/>

## Datepicker

Nu we hebben gezien hoe eenvoudig het is om een dialog te tonen gaan we nog eens kijken wat jQuery UI nog meer voor ons in petto heeft?

We gaan nu aan ons dialog een Datepicker toevoegen. Hiervoor moeten we een stukje html toevoegen en een extra regel javascript.

Maak de inhoud van de <div> met id:”myDialog” leeg.  
Zet daar een input element van het type tekst in met het id: “myDate”

|  |
| --- |
| <input type="text" id="myDate" /> |

Voeg onderaan de ExecuteOnStartup de initialisatie van de Datepicker toe:

|  |
| --- |
| $('#myDate').datepicker(); |

  
**Start de applicatie, open de dialog en zie het resultaat:**



Dus door letterlijk 1 regel code toe te voegen kunnen we van een tekstvak een Datepicker maken. Ook hier geldt hetzelfde als voor het dialog. We kunnen de Datepicker helemaal configureren door options mee te geven of ons te abonneren op een event.

Als je een datum selecteert zie je dat hij niet de standaard Nederlandse notatie gebruikt. Verder zagen we dat de kalender standaard op zondag begint. Dit gaan we her-configureren.

We gaan dus nu de firstDay en de datum notatie wijzigen. Om onnodige regels code te gebruiken gaan we nu het object inline aanmaken en vullen. We kunnen dan de initialisatie die we zojuist hadden toegevoegd aan het javascript bestand vervangen met de volgende code:

|  |
| --- |
| $('#myDate').datepicker({ firstDay: 1, dateFormat: 'dd-mm-yy' }); |

  
**Start de applicatie en bekijk het resultaat:**



Ook voor de Datepicker geldt dat het veel meer mogelijkheden heeft. Daarvoor kun je alle demo’s bekijken en technische documentatie raadplegen op: <http://jqueryui.com/demos/datepicker/>

## Buttons

Inleiding tot buttons

### Checkbox buttons

### Radio buttons

### Group buttons

## Animations / effects

## Create control sets

## Switching themes

# Advanced Javascript

## Create your own methods

Van <http://sandervandevelde.wordpress.com/2011/06/15/schrijf-eens-je-eigen-jquery-functies/>

Zie een pagina met html:

// this code contains a bug

<html>

<head>

<script type="text/javascript"

src="http://ajax.googleapis.com/ajax

/libs/jquery/1.6.1/jquery.min.js"></script>

</head>

<body>

<a id="nupuntnl"

href="http://www.nu.nl">Nu</a>

<script type="text/javascript">

$(document).ready(function() {

$('#nuputnl').text('Public link');

});

</script>

</body>

</html>

De .text manipuleert geen enkel object.

Dus we schrijven een functie die piept als nul of meerdere objecten gevonden zijn:

(function( $ ){

$.fn.onlyOne = function(mode) {

var mode = mode || 'off';

if (mode != 'off') {

if (this.length == 1) {

if (mode != 'on') {

alert('Information: one object found.' );

}

}

else {

alert('Warning: not one but '

+ this.length + ' objects expected.');

}

}

// do not break the normal flow;

return $(this);

};

})( jQuery );

<script type="text/javascript">

$(document).ready(function() {

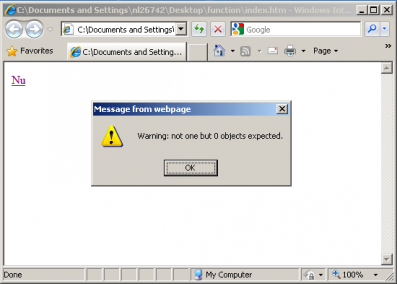
$( nuputnl').<strong>onlyOne('on')</strong>

.text('Public link');

});

</script>

Dit geeft dan namelijk :



Extra opdracht: bouw een functie die verwacht dat minstens één object wordt gevonden.

## Create a javascript object

<http://marcofranssen.nl/2011/09/17/jquery-events-contributes-to-clean-javascript/>

## Communicate between objects with events

<http://marcofranssen.nl/2011/09/17/jquery-events-contributes-to-clean-javascript/>

# MVVM met Knockout JS

Combineer

<http://sandervandevelde.wordpress.com/2011/09/23/integratie-van-knockout-in-asp-net-mvc3/>

en

<http://sandervandevelde.wordpress.com/2011/09/26/schrijf-je-in-voor-events-in-knockoutjs/>

extra opdracht:

## Simple bindings

## Advanced bindings

## Cascading dropdown

<http://marcofranssen.nl/2011/09/13/knockout-that-cascading-dropdown/>

project is al ver uitgewerkt. De controller acties zijn voorbereid. Wij gaan alleen het javascript schrijven.

## Animations with jQueryUI