Getting started with jQuery

Gill Cleeren
@gillcleeren
Hi, I’m Gill!

Gill Cleeren
MVP and Regional Director
.NET Architect @ Ordina
Trainer & speaker

@gillcleeren

gill@snowball.be
I’m a Pluralsight author!

• Courses on Windows 8, social and HTML5
• http://gicl.me/mypscourses
What we’ll be looking at...

- Hello jQuery!!
- The 3 jQuery fundamentals
- Creating and manipulating elements
- Working with events
- Built-in animations and effects
- Talking to the server with Ajax
- Working with WebForms and MVC
- jQuery UI
- jQuery plugins
- Using the CDN

I love jQuery!
Throughout the session...

• You’ll see some I♥ jQuery

• Goal: show a particular place where jQuery really stands out
HELLO JQUERY!
Hello jQuery!

- jQuery is
  - Most popular, cross-browser JavaScript library
  - Focusing on making client-side scripting of HTML simpler
    - Easy navigating the DOM
    - Handling events
    - Working with Ajax
  - Open-source, first released in 2006
  - Current release is 1.11 and 2.1
    - Same API
    - 2.X branch doesn’t support IE 6, 7 and 8
      - Recommended to use 1.X for public sites
Why jQuery?

• Many JavaScript frameworks try bending the language out of its natural form
• jQuery aims at leveraging CSS, HTML and JavaScript
• Advantages
  – Lightweight
  – Easy to learn using familiar CSS syntax and intuitive
  
  ```javascript
  $('#something').hide().css('background', 'red').fadeIn();
  ```
  
  – Many plugins available
  – Easy to extend and compatible
  – Support from Microsoft
  – Rich community
You are not alone!

Many LARGE companies use jQuery for their sites, including:

- Netflix
- EA
- NBC
- Dell
- Amazon
- Twitter
- Match.com
- Visual Studio
Microsoft and jQuery

• Included with Visual Studio
  – MVC
  – WebForms

• Microsoft is/was contributor to jQuery
  – Created templating, data linking and globalization (2010)
  – Not actively maintained now though

• CDN from Microsoft
Script, don’t get in my way!

- jQuery helps us writing *Unobtrusive JavaScript* code
- You don’t want to mingle style with HTML
- Why would you want to mingle *behavior* with HTML?

```javascript
<script type="text/javascript">
    window.onload = function() {
        document.getElementById('testButton').onclick = function() {
            document.getElementById('xyz').style.color = 'red';
        };
    };
</script>
```

- This will become a heavy job without jQuery!

This code is downloaded every time
Fundamentals #1: $

• $ function (aka jQuery() function) returns
  – A JavaScript object containing an array of DOM elements
  – In the order they were found in the document
  – Matching a specified selector (for example a CSS selector)
  – Known to mankind as a wrapper or wrapped set

• It returns the same group of elements, can be chained

```javascript
$(`div.someClass`).show();
```

```javascript
$(`div.someClass`).show().addClass(`SomeOtherClass`);```

Finds all DIVs with class someClass and sets them visible.
To the same set, this adds another class.
Fundamental #2: the ready handler

• Script execution should wait until DOM elements are ready
  – You say: window.onload?
  – Sadly, this waits for everything to be loaded, including images etc
  – Script execution is too late

• Instead, we need to wait only until the DOM tree is created
  – Can be difficult in cross-browser situations
  – Easy-peasy with jQuery

```javascript
$(document).ready(function() {
  $('div.someClass').show();
});
```

```javascript
(function() {
  $('div.someClass').show();
})();
```
Fundamental #3: selectors

• At the core of jQuery lies its **selector** engine
• $( ) is heavily overloaded
  – Making a selection
  – Creating new HTML elements
Fundamental #3: selectors

- Most basic: CSS selectors
  - Can be combined
    - Child selector
    - Attribute selector

\[
\text{
$\$(\text{"p a.someClass"})$
}\]

\[
\text{
$\$(\text{"p a.someClass, div"})$
}\]

\[
\text{
$\$(\text{"ul.someList > li > a"})$
}\]

\[
\text{
$\$(\text{"a[href*='http://www.snowball.be']"})$
}\]

\[
\text{
$\$(\text{"span[class^='some'"]")$
}\]

\[
\text{
$\$(\text{"span[class"]")$
}\]
Fundamental #3: selectors

- Position

  - \$\("a:first\)"
  - \$\("div:even\)"
  - \$\("table#someTable td:first-child\)"

- Pseudo-classes (CSS filter selectors & custom selectors)

  - \$\("input:checked\)"
  - \$\(":password\)"
  - \$\("input:not(:checked)\)"

Selects first A we can find on the page
Selects the “even” DIVs on a page
Selects the first cells within a table named someTable
Selects checked inputs (including the ones that weren’t checked initially)
Selects all INPUTs of type password
Selects all ‘not-checked’ inputs
More selectors

- Full list at [http://www.w3.org/TR/css3-selectors/](http://www.w3.org/TR/css3-selectors/)

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>any element</td>
</tr>
<tr>
<td>E</td>
<td>an element of type E</td>
</tr>
<tr>
<td>E[foo]</td>
<td>an E element with a &quot;foo&quot; attribute</td>
</tr>
<tr>
<td>E[foo^=&quot;bar&quot;]</td>
<td>an E element whose &quot;foo&quot; attribute value begins exactly with the string &quot;bar&quot;</td>
</tr>
<tr>
<td>E:nth-child(n)</td>
<td>an E element, the n-th child of its parent</td>
</tr>
<tr>
<td>E:first-child</td>
<td>an E element, first child of its parent</td>
</tr>
<tr>
<td>E:empty</td>
<td>an E element that has no children (including text nodes)</td>
</tr>
<tr>
<td>E:link</td>
<td>an E element being the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited)</td>
</tr>
<tr>
<td>E:visited</td>
<td>an E element being the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited)</td>
</tr>
<tr>
<td>E &gt; F</td>
<td>an F element child of an E element</td>
</tr>
<tr>
<td>E + F</td>
<td>an F element immediately preceded by an E element</td>
</tr>
</tbody>
</table>
Selecting elements using selectors

DEMO
Fundamental #3.1: creating elements

- `$('...')` selects an element <> `$('<li>')` creates an element

```javascript
$(function(){
    $('<div>I’m off</div>')
    .appendTo('body');
})

$(function(){
    $('<img>', {
        src: 'someImage.jpg',
        alt: 'Some nice image'
    })
    .appendTo('body');
})
```
Creating elements using $
CREATING AND MANIPULATING ELEMENTS
Working with the result of $\$

• Once we have a wrapped set, we can go wild with it!
  – Handle the set as a whole
  – Work with individual elements
Working with the result of $ $

- A wrapped set is like an array of elements, normal “array operations” can be done on it
  - Check the size
    ```javascript
    $('a').size();
    ```
  - Access an individual element
    ```javascript
    $('a')[0];
    $('a').get(0);
    ```
  - Loop over the elements
    ```javascript
    $('img').each(function(n){
      this.alt='image['+n+']';
    });
    ```
Working with the result of $\$

• Set operations (continued)
  – Add and remove elements

\[$(\"a[\text{class}]\").\text{add}(\"a[\text{href}]\")\;

  \]

  – Filter elements

\[$(\"a\").\text{filter}(\"[\text{href}^=\text{http://}]\")\;

  \]

• Remember that we are always returning the set
  – Chaining is always possible!

\[$(\"a[\text{class}]\")
 .\text{add}(\"a[\text{href}]\")
 .\text{filter}(\"[\text{href}^=\text{http://}]\")
 ...;\$
Working with the set

DEMO
Attributes

• When we want to change how an element looks, we can change its attributes

• jQuery provides the attr() method
  – 2 variations based on number and types of parameters
    • Read a specified property from first element in wrapped set
      $$\text{
      \$\left('\#myImage\right).\text{attr\left('alt'\right);}
      $$
    • Set a property on all elements in the wrapped set (0 or more)
      $$\text{
      \$\left('\#myImage\right).\text{attr\left('alt', 'Me in Paris'\right);}
      $$
Attributes (2)

• jQuery makes it easy to apply and remove CSS classes
  – addClass(), removeClass(), toggleClass() and hasClass()

• Changing individual CSS elements is supported
  – css() can be used to get or set CSS on an element

  ```javascript
  $('#mydiv').css('background-color', 'yellow');
  ```
Working with elements

• `html()` can be used to get or set the content of an element

  
  ```
  $('#mydiv').html();
  ```

  Retrieves the HTML content of `mydiv`

  – `text()` can retrieve combined textual content of all elements, including their children

• If the elements are form elements, we need to use `val()`

  ```
  $('input:checkbox:checked').val();
  ```

  Retrieves the value from a checked checkbox
Working with attributes

DEMO
Events: A bit of history

• Once upon a time, a browser called Netscape introduced an event model: DOM Level 0 Event Model
  – Creates event handlers as references to a function on a property
  – Not what we need if we want to create Unobtrusive JavaScript
  – Only one event handler per element for specific event
• Only got standardized until DOM Level 2 Event Model
  – Based on a system of event listeners (addEventListener)
  – IE decided to go its own way (attachEvent)
• Using event was a real mess because of browser dependencies
• jQuery comes to the rescue  I♥jQuery
jQuery events

• on() is where it all starts
  – Binds a function to any event on any DOM element
  – off() can be used to unbind a function from an event
  – Replaces the bind() and unbind()

```javascript
$('#myimg').on('click',
    function(event){alert('Hello World!');}
);
```

  – Works in any browser, jQuery hides the details for us
  – Possible to bind more than one event handler for an event on one element

• one() removes itself after event handler executed
Events

DEMO
BUILT-IN ANIMATIONS AND EFFECTS
Animations and effects

• Core jQuery has some basic effects
  – More are available in jQuery UI
  – Should be used with caution!

• Most basic ‘animation’ is hiding/showing an element
  – hide(): sets display:none on the element
  – show(): sets display to inline/block
  – toggle(): sets visible is hidden and vice-versa

• Methods are overloaded, accepting
  – Speed
  – Callback
Animations and effects (2)

• Elements can also be gradually added/removed
  – slideDown() and slideUp()
• Fading in is supported as well
  – fadeIn() and fadeOut()
• animate() is mother of all animations
  – Using ‘target values’ for style properties, jQuery will animate the transition
    $$\$('.'\text{someClass}')$.animate({opacity:0.25},'slow');$$
TALKING TO THE SERVER WITH AJAX
Ajax in the past

- When we were all young (in 1998), Microsoft introduced the ability to perform asynchronous requests from script (ActiveX)
- Later, other browsers implemented a standard, the XMLHttpRequest
  - IE6 uses an ActiveX object
- Result is that we need to do checks

```javascript
if(window.ActiveXObject) {
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
} else if (window.XMLHttpRequest) {
    xhr = new XMLHttpRequest();
}
```

- Again... jQuery to the rescue! I♥ jQuery
Ajax with jQuery

• Basic functionality to load content from a server-side resource:
  – load()
    • url
    • parameters: data to be passed (string, object...)
      – If provided, a POST is executed, otherwise a GET
    • callback (optional)

```
$( '#someDiv' ).load( 'test.html',
    function() {
        alert( 'Load was performed.' );
    } );
```

• Next to load, we can also use $.get()/$.getJSON() or $.post()
Basic Ajax request with load()
Ajax with jQuery

• If we need all control over the Ajax request we can get:
  – $.ajax()
    • options: defines an object containing all the properties for the Ajax request

• List of options is huge!
  – $.ajaxSetup
    • options: defines an object containing all the properties for the Ajax request, becoming the default for Ajax requests

```
$.ajaxSetup({
  type: 'POST',
  timeout: 5000,
  dataType: 'html'
});
```
Ajax with jQuery

• Throughout the Ajax request, we can get feedback
  – Local events from the $.ajax() call (callbacks)
  – Global events
    • Are broadcast to every element within the DOM, can be
      attached on any element
      – ajaxStart
      – ajaxSend
      – ajaxSuccess
      – ajaxError
      – ajaxComplete
More control with ajax()

DEMO
WORKING WITH WEBFORMS AND MVC
jQuery Ajax, ASP.NET MVC and WebForms

- jQuery can work in harmony with ASP.NET MVC and WebForms
  - Sample `ajax()` call for WebForms

```javascript
$.ajax({
    type: "post",
    contentType: "application/json; charset=utf-8",
    dataType: "json",
    url: "/Default.aspx/AddTask",
    data: JSON.stringify(dto)
});
```
ASP.NET WebForms with jQuery

DEMO
ASP.NET MVC with jQuery

DEMO
JQUERY UI
jQuery UI

• Huge extension of jQuery, providing more UI capabilities
• Contains number of UI features we’d logically need
• Includes
  – Effects: more advanced than core effects
  – Interactions: drag and drop
  – Widgets (aka controls): date picker...
  – All can be themed
• Code included in jquery-ui.js
jQueryUI Themes

• Themes come with the download
  – It’s *never* going to be OK for the marketing guys!
  – Options
    • Use it anyway
    • Use the ThemeRoller
    • Tweak a default or custom-created one
    • Create one yourself (Warning: the CSS is quite large)
Effects

• jQuery core contains some basic effects
• Based on the `effect(type, options, speed, callback)` method
  – Has several animation types such as puff, highlight and shake (even explode exists)
  – Also allows to do animations with colors (not possible with animate())
    • backgroundColor, color...
• Visibility methods (show()...) are extended
• Class methods (addClass()...) are extended
• position() method is added for advanced positioning

```javascript
$('#someElement').position({
  my: 'top center',
  at: 'bottom right',
  of: '#someOtherElement'
});
```
Effects

DEMO
Interactions

- Interactions focus on allowing users to directly interact with elements, which isn’t possible with standard HTML controls
  - They add advanced behaviors to our pages related to mouse interactions
- Available interactions:
  - Dragging
  - Dropping
  - Sorting
  - Resizing
  - Selecting
Dragging

• Easy-peasy (again) with jQuery
• draggable() is your friend (heavily overloaded once again)
  – Allows making elements draggable, possible with options (opacity...)

```javascript
$('#someDiv').draggable();
```
Dragging and drop

DEMO
Widgets: controls on steroids

- New controls (based on existing ones)
- Contents
  - Buttons
  - Sliders
  - Progress bars
  - Autocompletion
  - Date picker
  - Tabs
  - Accordion
  - Dialog box
Date picker

• Have you noticed that entering dates is a difficult thing for end users? Some will always get it wrong!

• jQuery UI’s DatePicker can help
  – datepicker() creates the control for you
  – Has numerous options, mostly defaults will do
Widgets in action

DEMO
JQUERY PLUGINS
Something missing in jQuery?

• 2 options:
  – Use an existing plugin
    • Google code (code.google.com): going to be retired soon!
    • GitHub
    • jQuery plugin (not active anymore)
  – Write a plugin yourself
    • Custom utility function
    • Create wrapper functions
Using a plugin

DEMO
Writing your own plugins

- Write a plugin to add it yourself!
  - Possible to write your own utility functions and wrapper methods
- Creating new wrapper methods:
  - Add the method as a property on the fn object in the $ namespace

```javascript
$.fn.wrapperFunctionName = function(params){function-body};
```

```javascript
(function($){
    $.fn.setToRed = function() {
        return this.css('color','red');
    };
})(jQuery);
```
Writing a plugin

DEMO
USING THE CDN
Where to get your stuff?

• Use a CDN?
  – Microsoft
  – Google

• Put scripts locally as well with a fallback mechanism

```
<script type="text/javascript"
    src="http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.min.js"/>
</script>
<script type="text/javascript">
if (typeof jQuery == 'undefined') {
    document.write(unescape("%3Cscript src="/Scripts/jquery-1.4.2.min.js"
        type='text/javascript'%3E%3C/script%3E"));
}
</script>
```
Summary

• Where does all the (l) for jQuery come from?
  – Light-weight library that uses JavaScript as JavaScript, relying on CSS
  – Cross-browser compatible, hides the details (ready handler)
  – Easy eventing model
  – Can work with MVC & WebForms
  – Easily extensible to fit your needs, tons of plugins already available
So I hope you now say too...
THANKS!

THANK YOU AND HAVE FUN

THANKS!
Getting started with jQuery