jQuery and Friends

Agenda

- jQuery
  - Intro and overview
- Selectors
- Manipulation
- HTML Forms
- Events
- Other JavaScript Libraries

What Is jQuery?

- **A JavaScript library**
  - **designed to simplify the client-side scripting of HTML**
- It's the most popular JavaScript library
  - Used by over 60% of the 10,000 most visited websites
- It was released in January 2006 at BarCamp NYC by John Resig
  - It is currently developed by a team of open source developers
- jQuery is free, open source software
  - licensed under the MIT License

Why Use jQuery?

- jQuery really is the "**write less, do more**" JavaScript Library!
- jQuery's syntax is designed to make it easier to:
  - navigate a document
  - select DOM elements
  - create animations
  - handle events
  - develop Ajax applications

- **Allows us to avoid common headaches associated with cross-browser development**
- Provides a large pool of plugins
- Tested on 50 browsers, 11 platforms and mobile (v1.x)

jQuery 1.x vs. 2.x vs. 3.x

- jQuery 2.x leaves behind the older Internet Explorer 6, 7, and 8 browsers
  - In return it is smaller, faster, and can be used in JavaScript environments where the code needed for old-IE compatibility often causes problems of its own
- **But don't worry, the jQuery team still supports the 1.x branch which *does* run on IE 6/7/8**
- You can (and should) continue to use jQuery 1.x on web sites that need to accommodate older browsers
  - 1.10 ~ 2.0
  - 1.10.2 ~ 2.0.3
  - 1.12.3 ~ 2.2.3 (newest)
  - . . .

Only one function!

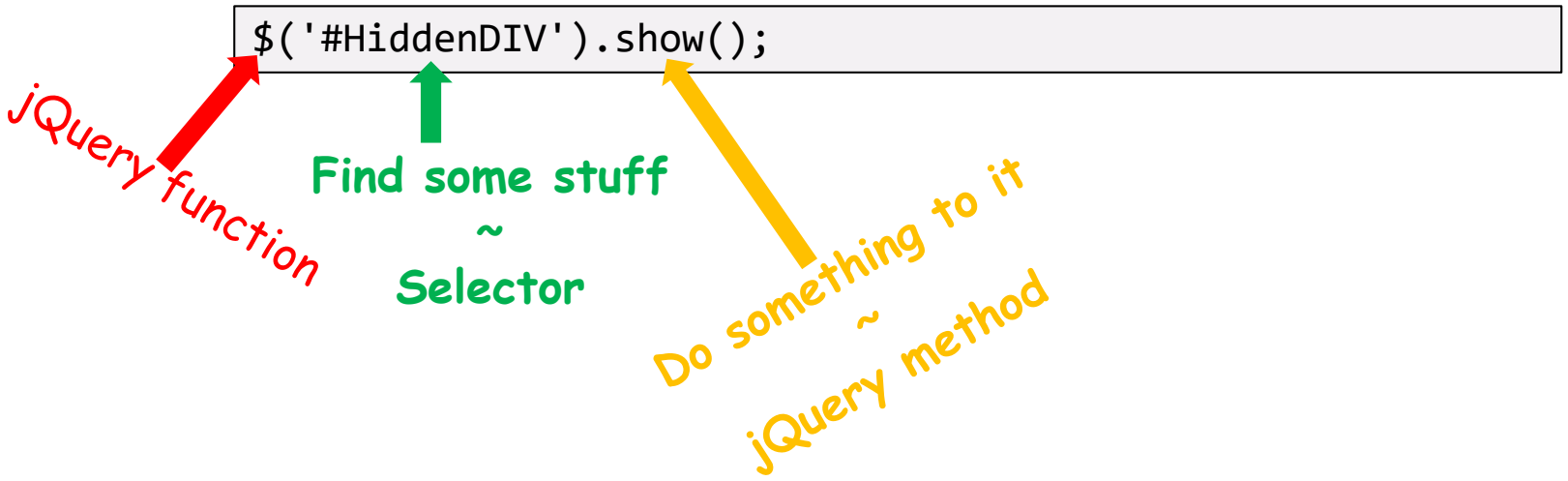
- Everything starts with a call to the jQuery() function
- Since it's called so often, the **$** variable is set up as an alias to jQuery

```
// Expose jQuery to the global object
window.jQuery = window.$ = jQuery;
```

- If you're also using another library you can revert to the previous $ function with jQuery.noConflict();

jQuery Philosophy

- Focus on the interaction between JavaScript and HTML
- Almost every operation boils down to:
  - **Find some stuff**
  - **Do something to it**



How to use jQuery on a Page?

- You just add a reference to the jQuery script file after all references to css-files but before your own javaScript code

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Hello World - jQuery Style</title>
    <link href="MyStyles.css" rel="stylesheet" />
</head>
<body>
    <div id="first"></div>

    <script src="Scripts/jquery-2.1.1.js"></script>
    <script>
        $(document).ready(function () {
            $('#first').html('<h1>Hello World!</h1>');
        });
    </script>
</body>
</html>
```

Use of CDN (Content Delivery Networks)

- Instead of deploying and serving your own jquery-2.0.min.js use a centrally-hosted CDN location for common client-side libraries like jQuery
- **Why?**
  - User's browser will get the library code from an optimized well-known location provided by a giant like Google, Microsoft and others
  - In fact, the browser will likely already have a downloaded + cached copy!

- Common CDN's:
  - Google: http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js
  - Microsoft: http://ajax.aspnetcdn.com/ajax/jquery/jquery-2.0.0.min.js

- You may provide a fallback from CDN to local jQuery:

```
<script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-2.0.0.min.js">
</script>
<script>window.jQuery || document.write(
'<script src="scripts/jquery-2.0.0.min.js">\x3C/script>')</script>
```

The ready Event

- jQuery fires a custom event named **ready** when the DOM is loaded and available for manipulation
  - Code that manipulates the DOM can run in a handler for this event

```
<!DOCTYPE html>
<html lang="en">
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
<script>
// Standard.
jQuery(document).ready(function () { alert('DOM is ready!'); });

// Shortcut, but same thing as above.
jQuery(function () { alert('No really, the DOM is ready!'); });
</script>
</head>
<body></body>
</html>
```

  - **Or you can place your DOM manipulating code just before the closing body tag </body>**

Obtrusive JavaScript

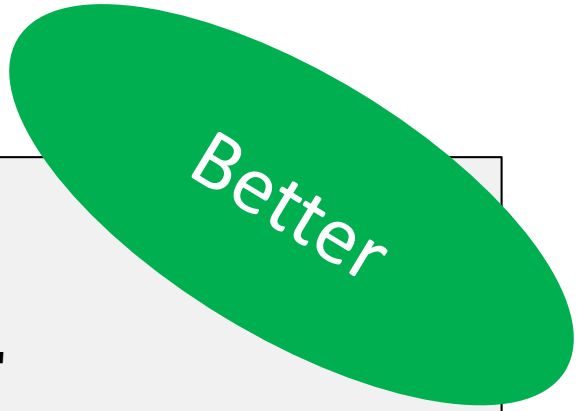
- Event subscriptions are embedded in the markup

```
...
<script type="text/javascript">
  function sayHello(){
    alert('Hello!');
  }
</script>
</head>

<body>
<input name="btn" id="btn" type="button" value="Hello!"
      onclick="sayHello();" />
</body>
</html>
```

Unobtrusive JavaScript

- Event subscriptions are made at run time




```
...
<body>
  <!-- button has no event wire up code -->
  <input name="btn" id="btn" type="button"
         value="Hello from jQuery!" />

  <script src="Scripts/jquery-2.0.0.js"></script>
  <script>
      $('#btn').click(function(){
          alert('Hello from jQuery!');
      });
  </script>

</body>
```

Unobtrusive JavaScript

- Event subscriptions are made at run time


```
...
<body>
  <!-- button has no event wire up code -->
  <input name="btn" id="btn" type="button"
         value="Hello from jQuery!" />

  <script src="Scripts/jquery-2.0.0.js"></script>
  <script>
      $('#btn').on('click',function(){
          alert('Hello from jQuery!');
      });
  </script>

</body>
```

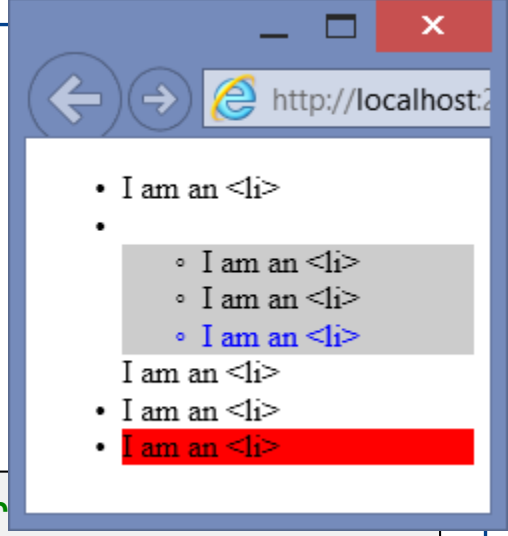
jQuery Chaining

- Once you have selected something using the jQuery function and created a wrapper set, you can actually chain jQuery methods to the DOM elements contained inside the set
- jQuery methods continue the chain by always returning the jQuery wrapper set, which can then be used by the next jQuery method in the chain
  - Note: Not all jQuery methods are chainable
    - And some functions alter the wrapper set (know as destructive methods)

```
$('a').text('jQuery')
      .attr('href', 'http://www.jquery.com/')
      .addClass('jQuery');
```

Exiting Chaining Using end()

- Chaining does not have to stop once the original wrapped set is altered
- Using the **end()** method, you can back out of any destructive changes made to the original wrapper set




```
<body>
  <ul id="list">
    <li></li>
    <li>
      <ul>
        <li></li>
        <li></li>
        <li></li>
      </ul>
    </li>
    <li></li>
    <li></li>
  </ul>
</body>
```

```
$('#list') // Original wrapper
.find('> li') // Destructive method.
.filter(':last') // Destructive method.
.addClass('last') // style background: #900;
.end() // End .filter(':last').
.find('ul') // Destructive method.
.css('background', '#ccc')
.find('li:last') // Destructive method.
.css('color', '#0000ff')
.end() // End .find('li:last')
.end() // End .find('ul')
.end() // End .find('> li')
.find('li') // Back to the orginal $('#list')
.append('I am an &lt;li&gt;');
```

Use of this in jQuery

- When attaching events to DOM elements contained in a wrapper set, the keyword **this** can be used to refer to the current DOM element invoking the event

```
<body>
<a id="link1">jQuery.com</a>
<a id="link2">jQuery.com</a>
<a id="link3">jQuery.com</a>
</body>
```

```
$('a').mouseenter(function () {
   alert(this.id);
});
```

each()

- Iterating over a set of elements by use of the jQuery **each()** method

```
<body>
<a id="link1">jQuery.com</a>
<a id="link2">jQuery.com</a>
<a id="link3">jQuery.com</a>
</body>
```

```
$('a').each(function(){
    alert($(this).attr('id'));
});
```

"this" refers to the current element in the loop

Extracting elements from a wrapper set

- You can always extract an element from the wrapper set and operate on the element via native JavaScript:
  - jQuery provides the handy **get()** method for accessing DOM elements at a specific index in the wrapper set
  - Or you can use the square bracket array notation on the jQuery object itself
  - Or you can call the get() method without passing it an index parameter, the method will then return all of the DOM elements in the wrapper set in a native JavaScript array

```
// Using DOM node properties to set the title attribute.
$('a').get(0).title = 'jQuery.com';
// Manipulation of DOM element using jQuery methods.
$('a').get(0).attr('href', 'http://www.jquery.com');

// Using the square bracket array notation.
$('a')[0].title = 'jQuery.com';
// Create native array from wrapper set.
var arrayOfAnchors = $('a').get();
```

SELECTORS

Simple Selection

- **Id**

  get element with id = myId

  ```
  $("#myId")
  ```

- **Class**

  get all elements with class=myClass

  ```
  $(".myClass")
  ```

- **Element**

  get all elements

  ```
  $("div")
  ```

Multiple Selectors

- You can string selectors together to selects the combined results of all the specified selectors

```
$("#myId, .myClass, div")
```

Filters

- Filter used alone

```
$(':hidden');
```

  Selects all hidden elements

- Select and filter combined

```
$('div:hidden')
```

  Selects all div elements, then selects only hidden elements

- Select and then filter

```
$('div').filter(':hidden');
```

  Selects all div elements, then selects only hidden elements
  - The **filter()** method is used to filter the current set of elements contained within the wrapper set

The :hidden and :visible Filter

- The custom jQuery selector filters **:hidden** and **:visible** do not take into account the CSS visibility property!
- The way jQuery determines if an element is hidden or visible is if the element consumes any space in the document
- That way, an element that might have a CSS display value of block contained in an element with a display value of none would accurately report that it is not visible

```
<body>
  <div id="parentDiv" style="display: none;">
    <div id="childDiv" style="display: block;"></div>
  </div>
</body>
```

```
$('#childDiv').is(':hidden'); // true
```

The is() Method

- You can check the current set against an expression/filter
- The check will return **true** if the set contains at least one element that is selected by the given expression/filter

```
<body>
  <div id="i0">jQuery</div>
  <div id="i1">jQuery</div>
</body>
```

```
alert($('div').is('#i1')); // Returns true.
alert($('div').is('#i2')); // Returns false.
alert($('div').is(':hidden')); // Returns false.
```

Filtering by Numeric Order

- jQuery provides filters for filtering a wrapper set by an element's numerical context within the set:
  - :first
  - :last
  - :even
  - :odd
  - :eq(index)
  - :gt(index)
  - :lt(index)

- For example **:eq(0)** and **:first** access the first element in the set

Filtering by relationships within the DOM

- jQuery provides several selectors to select elements that have unique relationships with other elements within the DOM

Relation	Examples
ancestor descendant	$("form input") , $("form fieldset input")
parent **>** *child*	$("ul.topnav > li")
prev **+** *next*	$("label + input") *Note: selects only one sibling*
prev **~** *siblings*	$("#prev ~ div") *Note: selects all following siblings*
:nth-child(*selector*)	$("ul li:nth-child(2)") *Note: first index is 1.*
:first-child	$("div span:first-child")
:last-child	$("div span:last-child")
:only-child	$("div button:only-child")

This is only an extract!

Filtering by Content

Content	Examples
:contains(*text*)	$("div:contains('John')")
:empty	$("td:empty")
:has(*selector*)	$("div:has(p)")
:parent	$("td:parent")

Check for Empty Wrapper Set

- Before you begin to operate on a wrapper set, it is logical to check that you have selected something
- The simplest solution is to use an **if** statement to check if the wrapper set contains any DOM elements

```
if (jQuery('a').get(0)) { // Is there an element in the set?
    jQuery('a').attr('href', 'http://www.jquery.com');
}
if (jQuery('a').length) { // Check the length of the set.
    jQuery('a').attr('title', 'jQuery');
}
```

Nesting Selector Filters

- Selector filters can be nested:

```
// Select all div's, remove all div's that have a child element
// with class="jQuery"
$('div:not(:has(.jQuery))')

// Select all div's, remove all div's that are odd in the set
// (count starts at 0)
$('div:not(:odd)')

// You can also nest and stack filters
$('p').filter(':not(:first):not(:last)')
```

MANIPULATION

Creating, operating, and adding HTML on the fly

DOM Insertion, Inside

- These methods allow us to insert new content inside an existing element
  - .append()
  - .appendTo()
  - .html()
  - .prepend()
  - .prependTo()
  - .text()

.append() And .appendTo()

- Same task, but different syntax:

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

```
$( ".inner" ).append( "<p>Test</p>" );
```

```
$( "<p>Test</p>" ).appendTo( ".inner" );
```

```
<h2>Greetings</h2>
<div class="container">
  <div class="inner">
    Hello
    <p>Test</p>
  </div>
  <div class="inner">
    Goodbye
    <p>Test</p>
  </div>
</div>
```

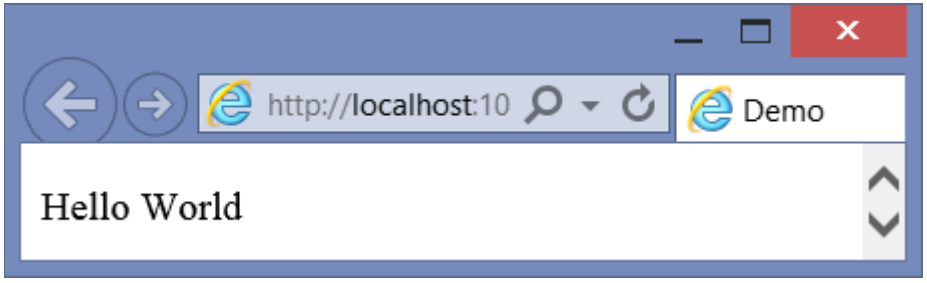
.html()

- Get the HTML contents of the first element in the set of matched elements
- Or set the HTML contents of every matched element

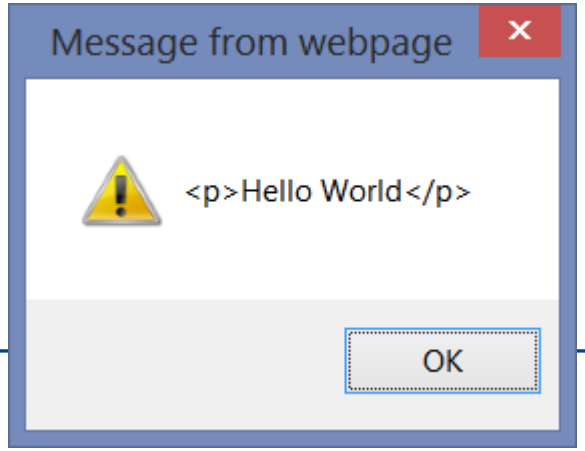
```
<div id="here"></div>
```

```
$('#here').html("<p>Hello World</p>");
```

```
<div id="here">
  <p>Hello World</p>
</div>
```




```
alert($('#here').html());
```



HTML FORMS

Val()

- The **val()** method can be used to set and get the attribute value of an input element

```
<body>
  <input type="button" />
  <input type="checkbox" />
  <input type="hidden" />
</body>
```

```
$('input:button').val('I am a button');
$('input:checkbox').val('I am a check box');
$('input:hidden').val('I am a hidden input');

alert($('input:button').val());
alert($('input:checkbox').val());
alert($('input:hidden').val());
```

Disable/Enable Form Elements

- You can easily disable form elements by setting the disabled attribute value of a form element to disabled

```
<body>
  <input name="targetButton" type="button" id="target" value="Click Me"/>
  <input name="disButton" type="button" id="disButton" value="Disable"/>
  <input name="enButton" type="button" id="enButton" value="Enable"/>
</body>
```

```
$('#disButton').click(function () {
    $('#target').attr('disabled', 'disabled');
});
$('#enButton').click(function () {
    $('#target').removeAttr('disabled');
});
```

Selecting/Clearing a Check Box or Radio Button

- You can select a radio button input or check box by setting its checked attribute to true using the attr()
- To clear a radio button input or check box, simply remove the checked attribute using the removeAttr() method or set the checked attribute value to an empty string

```
<body>
  <input name="" value="" type="checkbox" />
  <input name="" value="" type="radio" />
</body>
```

```
$('input:checkbox,input:radio').attr('checked', 'checked');
```

Selecting/Clearing Multiple Check Boxes or Radio Button

- You can use jQuery's val() on multiple check-box inputs or radio-button inputs to set the inputs to checked

```
<body>
  <input type="radio" value="radio1">
  <input type="radio" value="radio2">
  <input type="checkbox" value="checkbox1">
  <input type="checkbox" value="checkbox2">
</body>
```

```
$('input:radio,input:checkbox').val(['radio1', 'radio2',
    'checkbox1','checkbox2']);
```

- Use explicit iteration to clear:

```
$('input:radio,input:checkbox').removeAttr('checked');
```

Determining if a check box or radio button is selected

- You can determine if a check box input or radio button input is selected or cleared by using the :checked filter

```
<body>
  <input name="" value="" type="checkbox" />
  <input name="" value="" type="radio" />
</body>
```

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

Or

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

EVENTS

How to connect eventhandler to event

- You can use the **on()** method to add event handlers to the appropriate DOM elements
- Events:
  - blur
  - focus
  - load
  - resize
  - scroll
  - unload
  - beforeunload
  - click
  - dblclick
  - mousedown
  - mouseup
  - mousemove
  - mouseover
  - mouseout
  - change
  - select
  - submit
  - keydown
  - keypress
  - keyup
  - error

Old style

```
// Bind events
$('input').bind('click', function () {
    alert('You clicked me!');
});
$('input').bind('focus', function () {
    $('#log').html('You focused this input!');
});

// Unbind events
$('button').click(function () {
    $('input').unbind('click');
    $('input').unbind('focus');
    // Or, unbind all events
    // $('button').unbind();
});
```

Still ok

New style

```
$('input').on('click',function(){
    alert('You clicked me!');
});
```

Shortcuts to the **on()** Method

- jQuery provides several shortcuts to the **on()** method for use with all standard DOM events
  - simply substitute the event's name as the method name

```
<body>
  <a>Say Hi</a>
  <a>Say Hi</a>
</body>
```

```
$('a').click(function () { alert('hi') });
```

Equals

```
$('a').on('click', function () { alert('hi') });
```

Programmatically invoke a specific handler

- The shortcut syntax—e.g. **.click()**—for binding an event handler to a DOM element can also be used to invoke handlers programmatically
- To do this, simply use the shortcut event method without passing it a function

```
$('a').click(function () { alert('hi') }).click();
```

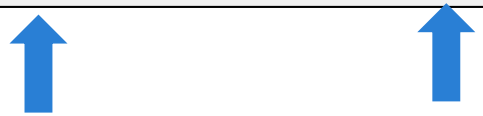
Invoke handler

- It is also possible to use the event **trigger()** method to invoke specific handlers

The Event Object

- To access the normalized jQuery event object, simply pass the anonymous function, passed to a jQuery event method, a parameter named "event" (or whatever you want to call it)
- Then, inside of the anonymous callback function, use the parameter to access the event object

```
$(window).load(function (event) { alert(event.type); });
```



Event Delegation

- Event delegation relies on event propagation (a.k.a. bubbling) When you click an <a> inside of a <li>, which is inside of a <ul>, the click event bubbles up the DOM from the <a> to the <li> to the <ul> and so on, until each ancestor element with a function assigned to an event handler fires

```
<body>
  <ul>
    <li><a href="#">remove</a></li>
    <li><a href="#">remove</a></li>
    <li><a href="#">remove</a></li>
    <li><a href="#">remove</a></li>
    <li><a href="#">remove</a></li>
    <li><a href="#">remove</a></li>
</ul></body>
```

```
$('ul').click(function (event) {
    $(event.target).parent('li').remove();
    e.preventDefault();  // Cancel default browser behavior
    e.stopPropagation(); // stop propagation.
});
```

OTHER JAVASCRIPT LIBRARIES

jQuery UI

http://jqueryui.com/

- Provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets
- **Widgets**
  All of jQuery UI's widgets are fully themeable using a consolidated, coordinated theme mechanism
  - **Accordion** – Accordion containers
  - **Autocomplete** – Auto-complete boxes based on what the user types
  - **Button** – Enhanced button appearance, turn radio buttons and checkboxes into pushbuttons
  - **Datepicker** – Advanced date-picker
  - **Dialog** – Show dialog boxes on top of other content, easily and robustly
  - **Menu** – Show a Menu
  - **Progressbar** – Progress bars, both animated and not
  - **Slider** – Fully customizable sliders with various features
  - **Spinner** – Show a Number Spinner
  - **Tabs** – Tabbed user interface handling, with both inline and demand-loaded content
  - **Tooltip** – Show a Tooltip

jQuery Mobile

http://jquerymobile.com/

- A touch-optimized web framework for a wide variety of smartphones and tablet computers
- Features
  - Theming framework that allows creation of custom themes
  - Limited dependencies and lightweight to optimize speed
  - The same underlying codebase will automatically scale to any screen
  - HTML5-driven configuration for laying out pages with minimal scripting
  - Ajax-powered navigation with animated page transitions that provides ability to clean URLs through pushState
  - UI widgets that are touch-optimized and platform-agnostic

JQuery Alternative

https://github.com/kenwheeler/cash

- Cash

  An absurdly small jQuery alternative for modern browsers (IE9+)

  - Provides jQuery style syntax to wrap modern Vanilla JS features
  - 100% feature parity with jQuery isn't a goal, but cash comes helpfully close

- Size Comparison

Library	jQuery 1.12.2	jQuery 2.2.2	Cash
Uncompressed	287K	253K	20K
Minified	95K	76K	9.5K
Minified & Gzipped	**34K**	**30K**	**3.4K**

- Example

```
$(function(){
  $('html').addClass('dom-loaded');
  $('<footer>Appended with cash</footer>').appendTo(
      document.body);
});
```

Common javascript libraries

- Globalize https://github.com/globalizejs/globalize
  A JavaScript library for internationalization and localization
- Lodash https://lodash.com/
  Utility library that provides functions for functional programming but without extending any of the built-in JavaScript objects
- Modernizr
  HTML5/CSS feature detector
- Require.js
  A JavaScript file and module loader
- history.js
  History State/APIs
- A list of useful JavaScript Libraries and jQuery Plugins (By Smashing Editorial)
  http://coding.smashingmagazine.com/2012/09/23/useful-javascript-libraries-jquery-plugins-web-developers/

Javascript MVC/MVP frameworks

- Angular
- React.js
- Vue.js
- Ember.js
- KnockoutJS
- Backbone.js

References and Links

- http://jquery.com/
  - http://learn.jquery.com/
  - http://api.jquery.com/

- jQuery Succinctly (free book by Cody Lindley) http://www.syncfusion.com/resources/techportal/ebooks/jquery
- Useful jQuery Function Demos http://coding.smashingmagazine.com/2012/05/31/50-jquery-function-demos-for-aspiring-web-developers/
- Writing Better jQuery Code http://flippinawesome.org/2013/11/25/writing-better-jquery-code/
- Simplest Way to Use JQuery Date Picker and Date Time Picker in ASP.NET MVC https://www.codeproject.com/Articles/1136464/Simplest-Way-to-Use-JQuery-Date-Picker-and-Date-Ti