JQUERY & HTML 5

Replacing legacy in line JavaScript

VANILLA JAVASCRIPT

What's wrong with it?

Synchronous Execution

- External resources are loaded as soon as the browser hits the <script> tag
- Page loading hangs until the referenced JavaScript file can be loaded by the browser
- If the script is hosted on an external server, your site is at the mercy of the external domain's performance

```
<script src="http://a.com/widget.js"></script>
<script>
    function loadComplete() { alert('test'); };
    runWidget();  // defined in external widget.js file
</script>
```

Global Namespace

- Functions and variables defined in-line typically pollute the global namespace
- Global variables are a source of unreliability and insecurity

Cross-browser Compatibility

- Programming is complicated enough, let a the framework deal with browser inconsistencies
- Example: Creating an AJAX request

```
function asyncRequest() {
        var req = false;
        if (window.XMLHttpRequest) {
                req = new XMLHttpRequest();
        } else if (window.ActiveXObject) { // IE
                try {
                        req = new ActiveXObject("Msxml2.XMLHTTP");
                } catch (e) {
                        try { req = new ActiveXObject("Microsoft.XMLHTTP"); }
                        catch (e) {}
                }
        }
        return req;
}
```

Etc.

- Selecting DOM elements can be complicated
- Manipulating the DOM can be complicated
- Only one script can override window.onload()
- SEO penalty – in line JavaScript is considered poor form; all JavaScript tags should be placed just before the closing </body> tag
- Readability – JavaScript intermingled with HTML complicates code maintenance
- Reusability – Embedded JavaScript is only usable on the template it's a part of. This often results in duplicate code being embedded on multiple similar pages.

JQUERY

Introduction

What is jQuery?

- jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

Launching Code on Document Ready

```
window.onload = function() {
  // Your code here
};
```

```
$(document).ready(function() {
  // Your code here
});
```

Click Handlers

```
<script>
  function doSomething() {
        alert('link clicked');
  }
</script>
<a onclick="doSomething()">
  Hello
</a>
```

```
<a>Hello</a>
<script>
 $('a').click(function(event) {
        alert('link clicked');
 });
</script>
```

HTML5

Document type declaration & important additions

Declaring an HTML5 Document

- Simplified doc type: <!DOCTYPE html>
- A valid HTML 5 document:

```
<!DOCTYPE html>
<html>
        <head>
                <title></title>
        </head>
        <body>
        </body>
</html>
```

Custom Data Attributes (data-*)

- Valid HTML attributes in HTML 4 were very limited (id, style, class, title, etc)
- HTML 5 adds unlimited custom data attributes to all elements in the format data-*, where * is any lowercase string you choose (e.g., data-role="section")
- Sample HTML with data attributes:

```
<ul>
    <li data-db-id="01">Red</li>
    <li data-db-id="02">Blue</li>
    <li data-db-id="03">Green</li>
</ul>
```

Why Custom Data Attributes?

- Custom data attributes are intended to store custom data private to the page or application, for which there are no more appropriate attributes or elements.
- These attributes are not intended for use by software that is independent of the site that uses the attributes.
- Every HTML element may have any number of custom data attributes specified, with any value.