Javascript eco-system

* Javascript has become the target for a large number of proprietary and open-source libraries and add-ins
  * e.g. Adobe Flash is based on Javascript
  * e.g. Firebug is a browser-based debugger for Javascript code
* Some of these are now considered " industry standard" , in that they are widely used by many practitioners
  * Dojo, Yui (Yahoo! User-Interface Library), Mootools, jsBox etc.
* jQuery is probably the most widely used Javascript add-in
  * It is open source
  * It is well documented (including many independent books)
  * It has become a separate eco-system, with many add-ins of its own
    * E.g. jQuery mobile – a library for Smartphone app development, jQuery UI for desktop user-interfaces etc.

jQuery

- What is it?
  - A code library that covers a number of important Javascript ' issues' , such as:
    - Working around browser differences so that the same code works anywhere
    - Reducing the wordiness of common Javascript patterns – for example:

```
document.getElementById("x");
```

becomes

```
$("#x");
```

    - Provides easy access to CSS styles in a document e.g.

```
$("#disclaimer" ).addClass('highlight');
```

    - Animations to promote interactivity
    - AJAX (Asynchronous Javascript & XML) – a mechanism for updating a page more efficiently than a refresh
    - Simplifies common Javascript tasks – iterating, manipulating arrays etc.

jQuery in Action

Plain Javascript Code

```
window.onload = function(){
  subject = document.getElementById(" subject" );
  description = document.getElementById(" description" );
  date = document.getElementById(" duedate" );
  time = document.getElementById(" duetime" );
  fillTimeList(time);
  ok = document.getElementById(" ok" );
  ok.onclick = add;
  cancel = document.getElementById(" cancel" );
  cancel.onclick = cancel;
  setInterval(checkDueAlerts, 60000);
};
```

jQuery powered code

```
$(document).ready( function(){
  var subject = $(' #subject' ),
      description = $(' #description' ),
      date = $(' #duedate' ),
      time = $(' #duetime' );
  fillTimeList(time);
  $(' #ok' ).bind(' click' , add);
  $(' #cancel' ).bind(' click' , cancel);
  setInterval(checkDueAlerts, 60000);
});
```

Anonymous (lambda) functions

- This is something that came from ' functional' programming languages (Lisp, Haskell, F#) that jQuery uses a lot
- Principle is simple: wherever you would refer to a named function without executing it, you can replace the name with the actual function definition
  - This produces code that is more ' interesting' (read that as awkward) looking
  - However, it reduces an effect that is called " namespace pollution" – i.e. every name you use in a program is one more possibility of that name clashing with the same name in someone else' s code
- The trick is to use anonymous functions only where they provide a benefit to the ***understandability*** of the code

Lambda & non-lambda

- Assigning a function to an event by name is clear and generally easy
- Lambda functions can drop you into "parenthesis hell"
  - Look at the organization of brackets in the second listing
- However, for straightforward uses (attaching a single event handler), anonymous functions represent a saving in time and effort without a noticeable effect on the readability of the code

```
var doSomething = function() {
  // Code in here does some job.
};

window.onload = function() {
  var button = document.getElementById(" button" );
  button.onclick = doSomething;
};
```

```
$(document).ready( function() {
  $(' #button' ).bind(' click' , function() {
      // Code in here does some job.
  });
});
```

jQuery Selectors

- Selectors – a powerful set of tools for accessing elements in HTML documents
  - By ID
    - $(' #id') replaces document.getElementById(' id');
  - By class
    - $(' .class') replaces document.getElementsByClassName(' class');
  - By tag name
    - $(' p') replaces document.getElementsByTagName(' p');
  - Everything
    - $(' *') selects every element in a document
  - Filters
    - $(" p:first"), $(" p:last"), $(" tr:even"), $(" :checkbox") etc.
    - First <p>, last <p>, all even numbered <tr>, all checkbox elements etc.
- Note – the jQuery versions return ***jQuery Objects*** (simple wrappers around the actual objects from the DOM) . This can simplify coding (or not). See last example, next slide.
- See www.w3schools.com/jquery/jquery_ref_selectors.asp

jQuery use cases

- jQuery covers a wide range of capabilities for manipulating, updating, and animating pages
- See www.w3schools.com/jquery
- Try the tutorials on this site

- Creating Animations, e.g.

```
$(" #button" ).click(function() {
  $(" #data_div" ).animate({height: " 300px" });
});
```

- Simplifying Events, e.g.

```
$(" #button" ).mousedown(function() {
  $(" p" ).slideToggle();
});
```

- Accessing/Manipulating HTML, e.g.

```
$(" #result" ).val( doCalculation() );
```

- Asynchronous updates - AJAX, e.g.

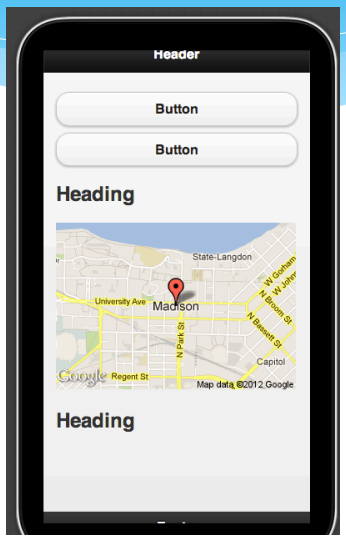
```
$(" button" ).click(function() {
  $.ajax({url: " datafile.txt" , success: function(result) {
    $(" #target_div" ).html(result);
  }});
});
```

- Function chaining, e.g.

```
$(" #mydiv" ).find(' TABLE .firstCol' ).removeClass(' .firstCol' )
             .css(' background' : ' red' )
             .append(' <span>This cell is now red</span>' );
```

jQuery Mobile

* jQuery Mobile is a jQuery add-in for styling and structuring web pages for mobile clients
  * Main advantage – pages styled to work on most popular smartphones
  * Significant features
    * Page structure is simplified
    * Automatic styling of buttons, lists, UI-elements
    * Style-sheets applied dynamically to suit device' s native styles



Google Maps

* This may not seem to be a specifically Javascript add-in
  * Available for Java, .NET, Android etc.
* For web-apps, Gmaps is made available as a <script> heading which provides a maps object hierarchy

* Access g-maps using script at: http://maps.googleapis.com/maps/api/js
* NO developer key needed unless you generate a lot of traffic
  * Free use for first 25,000 map requests per 24 hours for 90 days in a row
* Use an API key to get feedback from Google (e.g. app problems etc.)
  * https://developers.google.com/maps/documentation/javascript/get-api-key
  * Guidance at https://developers.google.com/maps/documentation/javascript/tutorial
* Part of a much larger eco-system:
  * Geo-coding (get GPS from Address)
  * Travel distances on road networks
  * Street View images
  * Places Library

RequireJS

- This library is for defining modules of JS code (defining application structure)
  - Meets AMD (Asynchronous Module Definition) specification
  - Works out dependencies (i.e. which module uses which module etc., to any depth)
  - Simplifies creating interdependent Javascript libraries for re-use

File is " counter.js"

```
define(
  function() {
    function Counter(upTo, f) {
      this.limit = upTo;
      this.count = 0;
      this.callback = f;
    };
    Counter.prototype.addOne = function() {
      this.count += 1;
      if(this.count >= this.limit) {
        this.callback();
      }
    };

    return Counter;
});
```

This is a normal JS type/class definition

Defines a new type : Counter

```
require( [" counter" ], function(Counter) {
  var c = new Counter(10, function(){ alert(" Reached 10" );}
  $(' #button' ).click( function() { c.addOne(); }
});
```

This code uses the new Counter type

Jasmine

- Behaviour-Driven-Development (BDD)
  - Build programs by defining the behaviour required in concrete terms
  - Jasmine is a BDD framework for Javascript
    - Describe a test/scenario that you expect a specific result from
    - Test for the result
    - Now write the software that must pass the test
- This removes possibly the biggest road-blocks to writing good software
  - What are its requirements?
  - When is it completed?
- To use Jasmine:
  - Include jasmine.js by adding it in a <script> tag
  - Write tests in JS code
  - Run them in the Jasmine Test Runner

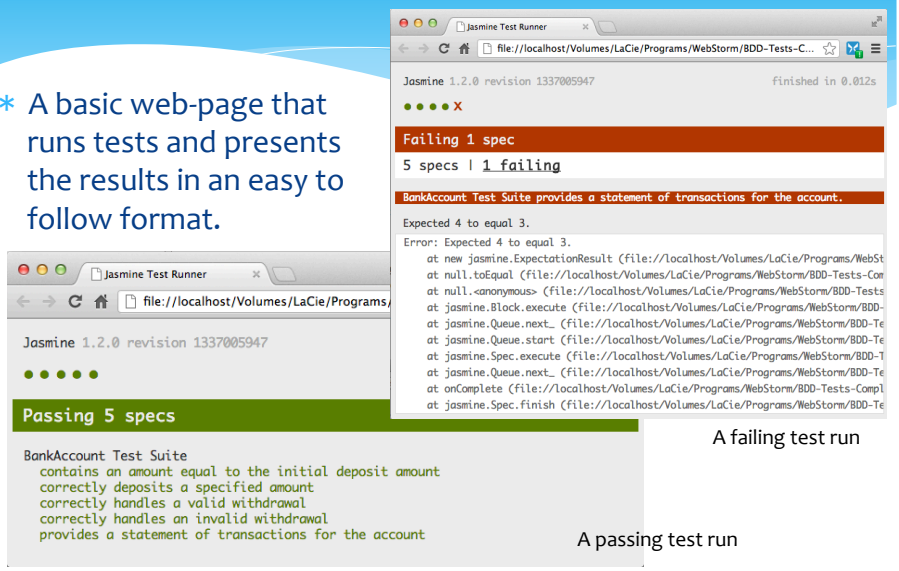
```
describe(" Tests" , function() {
  it(" has an expectation" , function() {
    expect(someCondition).toBe(true);
  });
});
```

- For Example:

```
describe(" addition" , function() {
  it(" Adds two numbers" , function() {
    expect(add_func(1, 1)).toEqual(2);
  });
});
```

Jasmine Test Runner

* A basic web-page that runs tests and presents the results in an easy to follow format.



A failing test run

A passing test run

jsDoc

* Program code is not easy to read
  * Programmers in all languages understand the need to document programs – particularly code libraries that others may use
  * Javascript contains internal documentation features

    // An inline comment

    /* A multi-line comment for more info. */
* JDoc is a Java utility for generating structured documentation
* jsDoc follows this pattern for Javascript code
* WebStorm includes jsDoc features
  * Search for jsDoc in Help.

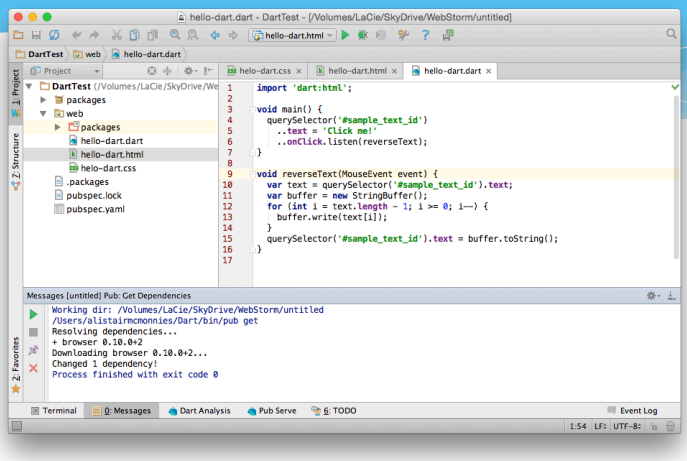
Javascript Alternatives

- Since Javascript is the only language that all browsers agree on, it looks like the only possibility for web clients, but…
  - Never underestimate
    1. the ingenuity of developers
    2. the level of desperation browser manufacturers will exhibit when trying to lock you into their tech (and escape Apple' s patents)
- For example
  - Google Dart
  - Microsoft TypeScript
- See https://jaxenter.com/angular-typescript-dart-115426.html for the current state of the showdown
- An interesting point – both Google Dart and TypeScript follow a pattern created by CoffeeScript – an open-source language that compiles to JS and has been available since 2009.

Google Dart

- This is a " clean break" from Javascript
  - Code runs natively in " Dartium" – a special version of Chromium that comes with the SDK
  - A tool is provided to " compile" dart language into very efficient Javascript (i.e. you are not " locked-in")
- The language is far better than Javascript
  - All of the serious problems – types, scope limitations, modular coding etc. – are ' solved' in Dart
  - Proper Object-Orientation (i.e. class based)
  - Less code to do a lot more
  - Still a strong basis in Javascript architecture – e.g. arrays and maps are the only data structures, closures still available
  - Dart IDE is based on Eclipse (a well-known and robust development environment)
- Dart (like TypeScript) is free
  - Google wants you to use it (that's not reason enough to do it, but it is a good choice of programming language)

WebStorm knows Dart.



See: https://www.jetbrains.com/webstorm/help/dart-support.html

TypeScript has a similar eco-system

* See https://www.jetbrains.com/webstorm/help/typescript-support.html
* Handles transpiling (converting .ts files to Javascript)
* Runs TypeScript projects in browser
* Debugs TypeScript
* Includes TypeScript code quality tools (TSLint – like jsLint)

Javascript – where from here?

- Javascript is not going away any time soon
  - Too much code already on the web for that, meaning…
  - too many vested interests
- The current status is that it is a well supported language (better than it deserves)
  - Good libraries, tools, publications, online support etc.
- A fair bet is that no-one apart from Google will go for Dart
  - However, no-brainer to continue to use it because of the Dart -> JS compilation feature
  - Similar languages (e.g. CoffeeScript) have been used to ' improve' Javascript for a long time, and are widely used
  - Google probably have the muscle (and the deep pockets) to continue supporting Dart for a many years yet
  - Even if they decide to give up on it, Dart -> JS is not going to disappear
- Anguar.js (developed by a Google team) supports ***TypeScript (whaaa!!)***
  - Google & Microsoft are best buddies these days
  - See https://www.reddit.com/r/dartlang/comments/2yhhyr/dart_20_vs_typescript_20/
  - See http://developer.telerik.com/featured/the-rise-of-typescript/