Modern Web Application Development

Sam Hogarth
Some History
Early Web Applications

- Server-side scripting only
  - e.g. PHP/ASP
- Basic client-side scripts
  - JavaScript/JScript/VBScript
  - Major differences in browser implementations
    - Different scripting engines
    - Inconsistent Document Object Model (DOM) for manipulating HTML elements
    - Only used for small tasks

2005: The Facebook
Server-side PHP with MySQL database

2006: Twitter
Ruby client, MySQL database
2002: Java Applet for real-time chat
Internet relay chat (IRC) protocol

2005: Flash component for video
YouTube

2007: Pure HTML/JavaScript chat
Mibbit using AJAX to communicate

Rich in-browser content
Traditional Architecture

Client-side (browser)

Page

- HTML
- CSS
- (Small amount of) JS

HTTP Request/Response

Server-side

Web Server

- PHP
- ASP(.NET)
- JSP
- or Perl, Python, etc…

Other systems (databases, APIs, etc.)
Mobile

2005: BBC News WAP
Early mobile internet

2005: BBC News WAP
Early mobile internet

2011: Google Maps
Website with access to device GPS and compass through new web APIs

2012: BBC News goes responsive
Single website for desktop/mobile

2015: Facebook Mobile
Mobile-optimised website
Trends

- Increasing availability and usage of mobile devices
- Speed and performance improvements to browser’s JavaScript engines
- Release and further development of HTML5 Specification

Meaning: **Doing more with JavaScript**
Now

- Web applications built with JavaScript, HTML, CSS
- Servers provide services and capabilities
- “Responsive” behaviour
- Feature-rich experiences natively in the browser
Now

Google Docs
Real-time updates and collaboration

Coinbase Exchange
Live updating Bitcoin trading platform

Outlook.com
Single page application (SPA) email client
Modern web application architecture

Client-side (browser)

- Single-Page Application (SPA)
  - HTML(5)
  - CSS
  - JavaScript

  - HTTP Request/Response
  - WebSocket Streaming

Server-side

- REST Services
  - Java
  - C#
  - JavaScript (Node.js) or Go, Scala, etc...

  - Other systems (databases, APIs, etc.)
Writing Large Web Apps
Main Concerns

- Displaying data on the page
- Handling user events
- Managing navigation (routing)
- Co-ordinating network activity
- Security
- Code complexity
Frameworks

Reusable software that provides a basic structure and functionality

To compare, see TodoMVC
MVC

Maps user actions to model updates
Manages the flow of data

Controller

View
- Displays the data
- Directs input to the controller

Model
- Holds data and application state
- Responds to data queries
- Handles the ‘business logic’
Unidirectional (Flux/Redux) Architecture

All changes to the system enter via a single point, whether:
- UI
- Network
- Server

Update application state in response to change

UI redraws based on new application state
Reusable components

Libraries
- Charting (e.g. D3)
- UI components (e.g. Chosen, jQuery UI)
- Utility functions (e.g. lodash)

Front-end frameworks
- Layouts (e.g. Bootstrap)
Transpilation

ES6 (or something else)

\[ [1,2,3].map(n => n + 1); \]

ES5

\[ [1,2,3].map(function(n) { return n + 1; }); \]
Development Process
The Team

Product Owner (& Users!)
Business Analyst
Project manager
User experience (UX) designer
Technical architect
Developer(s)
Tester(s)
Scrum

- Daily Scrum
- 24 Hours
- 2-4 Weeks
- Potentially Shippable Product Increment

Product Backlog

Sprint Backlog
Source Control Tooling

1. Manages source code storage
2. Branches for developing features
3. Tags for releases
4. Code review tools
Consistent Styles

Lint tool (e.g. JSLint, JSHint, ESLint, JSCS) checks for unintended syntax, bad practice and conformance to style guide

- e.g. Google JS Style Guide, Airbnb JS Style Guide

```
jslint:x2j.js:21:10:Expected exactly one space between 'function' and '('.
jslint:x2j.js:24:5:Missing 'use strict' statement.
jslint:x2j.js:26:9:Expected 'isIE' at column 13, not column 9.
jslint:x2j.js:26:23:Expected exactly one space between 'function' and '('.
jslint:x2j.js:26:25:Expected exactly one space between ')' and '{'.
jslint:x2j.js:26:25:Missing space between ')' and '{'.
```
Testing (Unit)

Ensures each component functions as expected.

Run in a real browser, or "headless" browser (Phantom)

```javascript
// Calculator object
var calc = {
    sum: function( a, b ) {
        return a + b;
    }
};

// Test `sum` method
test( Test the calculator can add numbers correctly, 2, function() {
    equal( calc.sum( 1, 2 ), 3 );
    equal( calc.sum( 5, 5 ), 10 );
});
```
Continuous Integration

1. Watch source code repository for new code changes
2. Build: Check styling, Run Tests, Minify and combine code
Deployment

Very specific to an individual application

DEV
- Server
- Database

QA
- Server
- Database

UAT
- Load Balancer
- Server 1
- Server 2
- Cloned Database

PROD
- Load Balancer
- Server 1
- Server 2
- Real Database
Testing (End-to-end)

Simulates using the whole application
The Future?
More Developer Tools, More Browser Capabilities

- Web Components and Custom Elements
- Front-end programming languages: Elm, PureScript
- Service Workers
- WebVR  
  - Web Speech API
  - Web Audio API