Modern Web Development
From Angle Brackets to Web Sockets

Pete Snyder <psnyde2@uic.edu>
Outline (or, what am i going to be going on about…)

1. What is the Web?

2. Why the web matters

3. What’s unique about web development?

4. What goes into a modern web project?

5. Making it fun and getting it done

6. Web development challenges
1. What is the Web?
(or, let's define terms)
1. What is the Web? (1/2)

- Markup: (X)HTML, JSON, YAML, XML…
- Protocols: HTTP(S), WebSockets, SPDY…
- WebServices: SOAP, REST, XML-RPC…
- Browsers: Moasic … Chrome…
- Design strategies: Response, Accessible…
- Languages: Javascript, Python, PHP, Ruby…
- Databases: MySQL, MSSQL, MongoDB, CouchDB…
- Scaling: Reverse Proxies, load balancers, CDNs…
- Servers: Tomcat, Apache, Nginx, Node.js…
- Frameworks: Ruby on Rails, Django, Drupal, Symphony…
- Standards: RSS, ATOM, RDFa…
1. What is the Web? (2/2)
2. Why the Web Matters
(or, why this isn’t all a waste of time)
2. Why the Web Matters

• Most popular application platform (by far)

• Everything talks “the web”

• Building for other platforms requires the web
3. What’s Unique About Web Development?
(or, bad news / good news)
What’s Unique About Web Development? (1/2)

Bad Bits

1. Unplanned Environment
2. Everything Needs to Work Everywhere
3. Many Parts / Languages / Environments
What’s Unique About Web Development? (2/2)

Good Bits

1. Heavily OpenSource

2. High Level Languages / Abstractions

3. Popular!
4. What Goes Into a Modern Web Project?
(or, how the sausage is made)
4. What Goes Into a Modern Web Project?

1. Static Web Pages
4.1 Static Web Pages: HTTP

- Application layer protocol
- Usually sent over TCP
- Usually port 80
- Plain text
4.1 Static Web Pages: More HTTP

GET /index.html HTTP/1.1
Host: www.example.com

HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Accept-Ranges: none
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8

{Body}
4.1 Static Web Pages: HTML

- Plain text markup
- Angle brackets
- Well defined tags and attributes
  - `<p>`: Paragraph
  - `<div>`: Division / Section
  - `<ul>`: Unordered List
  - `<ol>`: Ordered List
  - `<h[1-5]>`: Section Headers
  - `<img>`: An image
  - etc...
- “id” and “class” for overloading
4.1 Static Web Pages: The Dom

```html
<html>
  <head>
    <title>Todo Page</title>
  </head>
  <body>
    <h1>My TODOs</h1>
    <p>Boy, its a long list.</p>
    <img src="http://localhost/img/lots.jpg">
    <ul id="todo-list">
      <li>First Thing</li>
      <li class="important">Second Thing</li>
      <li class="important">Third Thing</li>
    </ul>
  </body>
</html>
```

DOM:

```
- HEAD
  - TITLE[text="Todo Page"]
- BODY
  - H1[text="Things I gotta Do Today"]
  - P[text="This is Quite Exciting"]
  - IMG[src="http://localhost/img/lots.jpg"]
  - UL[id="todo-list"]
    - LI[text="First Thing"]
    - LI[class="important-item", text="Second Thing"]
    - LI[class="important-item", text="Third Thing"]
```
4.1 Static Web Pages: Display

DOM

- HEAD
  - TITLE[text="Todo Page"]
- BODY
  - H1[text="Things I gotta Do Today"]
  - P[text="This is Quite Exciting"]
  - IMG[src="http://localhost/img/lots.jpg"]
  - UL[id="todo-list"]
    - LI[text="First Thing"]
    - LI[class="important-item", text="Second Thing"]
    - LI[class="important-item", text="Third Thing"]

Render

My TODOs

Boy, its a long list.

- First Thing
- Second Thing
- Third Thing
4.1 Static Web Pages: Putting It All Together

- Browser makes HTTP request
- Server responds with HTML (or other asset)
- Client Parses HTML to DOM
- Client Renders DOM

GET /index.html

Parse

Display

Monday, November 5, 12
4. What Goes Into a Modern Web Project?

1. Static Web Pages

2. Presentation
4.2 Presentation: CSS

- CSS: Cascading Style Sheets
- Plain text
- Included in document
- Separates content and presentation
- Pairs of “selectors” and “rules”
- Controls positioning, colors, fonts, animations, etc
4.2 Presentation: CSS

```html
<html>
<head>
  <title>Todo Page</title>
  <link type="text/css" rel="stylesheet" href="http://localhost/styles.4-2.css">
</head>
<body>
  <h1>My TODOs</h1>
  <p>Boy, its a long list.</p>
  <img src="http://localhost/img/lots.jpg">
  <ul id="todo-list">
    <li>First Thing</li>
    <li class="important">Second Thing</li>
    <li class="important">Third Thing</li>
  </ul>
</body>
</html>
```
4.2 Presentation: CSS

```
body {
  background: lightgrey;
}

img {
  border: 2px solid black;
}

#todo-list {
  background-color: yellow;
}

li.important {
  color: red;
  font-weight: bold;
}
```
4. What Goes Into a Modern Web Project?

1. Static Web Pages
2. Presentation
3. Dynamic Page Generation
3. Dynamic Page Generation

- Need a way to send input from browser to server
- Need a way to process that information
3. Dynamic Page Generation: Forms

HTML Form

```
<form>
  <p>
    <label for="statement">Say this:</label>
    <input name="statement">
  </p>
  <p>
    <label for="count">This many times:</label>
    <select name="count">
      <option value="5">5</option>
      <option value="10">10</option>
      <option value="15">15</option>
    </select>
  </p>
  <button type="submit">Speak it</button>
</form>
```

Rendering

Say this: 

This many times:  

Speak it
3. Dynamic Page Generation: HTTP

Form

HTTP Request

GET

GET /index.4-3.html
→ ?statement=Oh+Hey&count=5 HTTP/1.1
Host: localhost

POST

POST /index.4-3.html HTTP/1.1
Host: localhost
statement=Oh+Hey&count=5
3. Dynamic Page Generation: HTTP

**HTTP Request**

```
GET /speaker.php → ?statement=Oh+Hey&count=5 HTTP/1.1
Host: localhost
```

**Server Side Program**

```php
<?php

$num_times = $_GET['count'];
$phrase = $_GET['statement'];

echo "<ol>
";

for ($i = 0; $i < $num_times; $i++) {
    echo "<li>" . $phrase . "</li>";
}

echo "</ol>";

?>
```
3. Dynamic Page Generation: HTTP

Server Side Program

```php
<?php

$num_times = $_GET['count'];
$phrase = $_GET['statement'];

echo "<ol>\n";

for ($i = 0; $i < $num_times; $i++) {
    echo "<li>" . $phrase . "</li>";
}

echo "</ol>\n";

```

Dynamic Output

```html
<ol>
  <li>Oh Hey</li>
  <li>Oh Hey</li>
  <li>Oh Hey</li>
  <li>Oh Hey</li>
</ol>
```
3. Dynamic Page Generation: HTTP

Dynamic Output

Browser Rendering
3. Dynamic Page Generation: Complete

GET /index.html

<head><body>…

Parse

Display

GET /index.html

<head><body>…

Generation
4. What Goes Into a Modern Web Project?

1. Static Web Pages

2. Presentation

3. Dynamic Page Generation

4. Persistence
4.4 Persistence: Cookies and Sessions

- Values that browsers automatically include
- Allow sites to distinguish users across requests
- Along with server side persistent stores, allows for “sessions”

DELETE COOKIES ?!
4.4 Persistence: Cookies and Sessions

- GET /index.php
  - Login Request HTML
  - POST /login.php
    - user / pass
  - HTML
    - set-cookie: <uuid>
  - Parse / Record Cookie
  - Display

- GET /index.php
  - Login Request HTML
  - POST /login.php
    - user / pass
  - HTML
    - set-cookie: <uuid>
  - Checks authentication
  - Checks request
4.4 Persistence: Cookies and Sessions

GET /index.php
Cookie: <uuid>

Restricted Access HTML

Parse

Display

GET /index.php
Cookie: <uuid>

Checks request

Restricted Access HTML
4. What Goes Into a Modern Web Project?

1. Static Web Pages
2. Presentation
3. Dynamic Page Generation
4. Persistence
5. Client Side Logic
4.5 Client Side Programming: Javascript (1/6)

- Javascript
- Client / Browser side programming
- History: Originally developed by Netscape in 1995 by Brendan Eich in 10 days!
- AKA ECMAscript, Livescript and JScript
- Not at all related to Java
4.5 Client Side Programming: Looks Like C… (2/6)

- Curly braces
- for / while / do-while loops
- Semi-colon terminated statements
- Functions look C-“ish”
- Trips lots of people up

```javascript
function doubler (a) {
    return a + a;
}

var i = 0,
    value = 1;

for (i = 0; i < 10; i += 1) {
    value = doubler(value);
}

// Will print: 'Value' now equals 1024
alert("'Value' now equals " + value);

do {
    i -= 1;
    value = value / 2;
} while (i > 0);

// Will print: 'Value' now equals 1
alert("'Value' now equals " + value);
```
4.5 Client Side Programming: Not Like C! (3/6)

(function () {
    var cat = {
        voice: "meow",
        speak: function () {
            alert(this.voice);
        }
    }

    var dog = Object.create(cat);
    dog.voice = "woof";
    cat.speak(); // Alerts "meow"
    dog.speak(); // Alerts "woof"

    var adder = (function () {
        var initial = 0;
        return function (value) {
            initial + value;
            return initial;
        }
    })();

    // Error because initial is undefined
    alert("initial = " + initial);
})();

- Dynamically typed
- First class functions
- Lambda / anonymous functions
- Prototypal inheritance
- Function Scope
4.5 Client Side Programming: The DOM API (4/6)

- API for interacting with DOM

- Buggy, unintuitive and inconsistent

- Events / effects / styles, etc.
4.5 Client Side Programming: Example (5/6)

```html
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <h1>Blitzkrieg Bop</h1>
    <p>Hey Ho, Lets Go!</p>
    <p>A song by <span id="band-name">Blondie</span>.</p>
  </body>
</html>
```
4.5 Client Side Programming: Example (6/6)

```html
<html>
<head>
<title>Example</title>
<script src="http://localhost/script.js"></script>
</head>
<body>
<h1>Blitzkrieg Bop</h1>
<p>Hey Ho, Lets Go!</p>
<p>A song by Blondie</p>
</body>
</html>
```

```javascript
var band_name = document.getElementById("band-name");
title_tag = document.getElementsByTagName("H1");
paragraphs = document.getElementsByTagName("P");

band_name.innerHTML = "The Ramones";
title_tag[0].style.color = "red";
title_tag[0].style.textDecoration = "underline";
paragraphs[0].parentNode.removeChild(paragraphs[0]);
```
4. What Goes Into a Modern Web Project?

1. Static Web Pages
2. Presentation
3. Dynamic Page Generation
4. Persistence
5. Client Side Logic
6. Background Client/Server Interaction (AJAX)
4.6 Background Interaction: AJAX (1/3)

- Asynchronous Javascript and XML
- Allows for the browser to do background requests
- Javascript API
- Ends page response model
- Talk HTML / JSON / XML / YAML / etc.
4.6 Background Interaction: Flow (2/3)
4.6 Background Interaction: Formats (3/3)

JSON

```json
{
  "topic": "Web Development",
  "languages": ["Javascript", "Python", "Ruby", "PHP"],
  "ongoing": true
}
```

XML

```xml
<Document>
  <topic>Web Development</topic>
  <languages>
    <language>Javascript</language>
    <language>Python</language>
    <language>Ruby</language>
    <language>PHP</language>
  </languages>
  <ongoing>true</ongoing>
</Document>
```

YAML

```yaml
---

topic: "Web Development"
languages:
  - Javascript
  - Python
  - Ruby
  - PHP
ongoing: true
```
4. What Goes Into a Modern Web Project?

1. Static Web Pages
2. Presentation
3. Dynamic Page Generation
4. Persistence
5. Client Side Logic
6. Background Client/Server Interaction (AJAX)
7. Two Way Communication
4.7 Two Way Communication: Old Way (1/4)

- AJAX Polling
- Client checks periodically for updates
- Wasteful but reliable
4.7 Two Way Communication: Polling Flow (2/4)

GET /index.php

HTML

Parse

Display

AJAX Request (anything new?)

Response (no)

AJAX Request (anything new?)

Response (no)

AJAX Request (anything new?)

Response (Yes!) - JSON
4.7 Two Way Communication: WebSockets (3/4)

- Allows servers to push updates to clients
- Starts on HTTP, updates to lighter protocol
- Available in all latest browsers
- Javascript API talks to implementing server
4.7 Two Way Communication: WebSockets (4/4)

HTTP

GET /index.php

HTML

Parse

Display

Web Sockets

Server -> Client Message

Server -> Client Message

Client -> Server Message

Server -> Client Message
4. What Goes Into a Modern Web Project?

1. Static Web Pages
2. Presentation
3. Dynamic Page Generation
4. Persistence
5. Client Side Logic
6. Background Client/Server Interaction (AJAX)
7. Two Way Communication
8. HTML5 (The Kitchen Sink)
4.8 Looking Forward: HTML5

- LocalStorage (javascript API for structured DB storage in the browser)
- Canvas API
- WebGL
- Failure condition specs
- New CSS
- `<audio>` / `<video>`
- New Markup (`<article>`, `<section>`)
5. Making Web Development Fun
(or, at least easing the pain)
5. Making This All Fun

- Use client side (CSS/JS) frameworks
  - Bootstrap / Blueprint
  - jQuery / YUI

- Use compile down languages
  - LESS
  - CoffeeScript

- Use Server Side Frameworks
  - PHP: Symphony, Drupal, CodeIgniter, Zend
  - Ruby: Rails
  - Python: Django, web2py, Twisted, Tornado
6. Web Development Challenges
(or, everything is broken)
6. Web Development Challenges

- Following best practices is tedious!
- HTTP is not a good protocol for all this!
- Scaling this is really hard!
- Web security is really hard!
Thanks Very Much!

Pete Snyder <psnyde2@uic.edu>