summer school

Web 2.0 with XML

24 September 2009

Building Dynamic Web Apps with XML

Norman Walsh

Mark Logic Corporation

Learning Objectives

Understand the fundamentals of dynamic web applications. Learn how to apply web technologies to introduce dynamic behavior to your web pages.

- HTML
- CSS
- JavaScript
- XML

summer school

Contents

Lecture

1. Web 1.0 vs. Web 2.0
2. Web standards
3. Asynchronous communication
4. JavaScript frameworks

Tutorial Session

1. Incorporating simple dynamic behavior

summer school

Web 1.0 vs. Web 2.0

Web 1.0

- Very limited designs
  - No images at first
  - No CSS
- Completely static pages

Web 1.0 “applications”

- The introduction of forms allowed for some interactivity:
  - Initial page provides a form
  - Client data sent to server
  - Server does some processing
  - Results sent back to client
- Very heavyweight:
  - No client-side interaction
  - Result is always a complete refresh of the whole page
- Still very limited designs

Web 2.0 applications (1)

- Rich designs
  - Images, icons, colors, gradients, rounded corners, drop shadows, …
  - They (can) look much more like desktop applications

summer school

Web 2.0 applications (2)

- Fully interactive
  - Client side: fade-in, fade-out, drag-an-drop, …
  - Server side: type-ahead, spelling suggestions, forms processing…

Web 2.0 applications (3)

- Much more lightweight
  - Client (perhaps in concert with or server side processing) can update different parts of the page without updating the whole page

summer school

Web standards

Web 2.0 technologies

- Web standards
  - HTML
  - CSS
  - JavaScript
  - XML
- Proprietary technologies
  - ~~Flash~~
  - ~~Silverlight~~

Browser features

- Web 2.0 applications require specific client features
  - JavaScript
  - AJAX (XMLHttpRequest object)
  - CSS

Supported browsers

- Supported by all the major browsers
  - Firefox and "Gecko"-derived browsers
  - Safari and "Webkit"-derived browsers
  - Internet Explorer (to varying degrees, with varying bugs)
  - Opera
- Not universally supported
  - Links and other "text only" browsers
  - Screen readers
  - Search engines

What, how, and why?

- What are web pages made of?
- How are they made?
- How do they work?

HTML

- "Standard" HTML
  - Universally supported vocabulary
    - div, span, h1, h2, table, …
  - Supports a degree of "semantic" markup
    - Most presentational decisions can be made elsewhere
- HTML vs. XHTML
- HTML5 vs. HTML vs. XHTML vs. …

CSS

- Applies presentation to HTML
  - borders, margins
  - colors
  - fonts
  - graphics
- Sufficiently powerful for most web applications
- Continues to grow in sophistication
  - Browsers lag by varying amounts

JavaScript

- Web browser scripting language
  - Quite modern and advanced
  - Defined by ECMA
- JavaScript runs in the browser
  - Extends the browser in application-specific ways
  - Responds to user interaction
  - Communicates with the server
  - Updates underlying HTML and CSS

XML

- Declarative markup language
  - Arbitrary semantic complexity
    - div, span, book, spreadsheet, widget-description, …
  - Can be parsed without knowing the tag set
- Nearly ubiquitous support

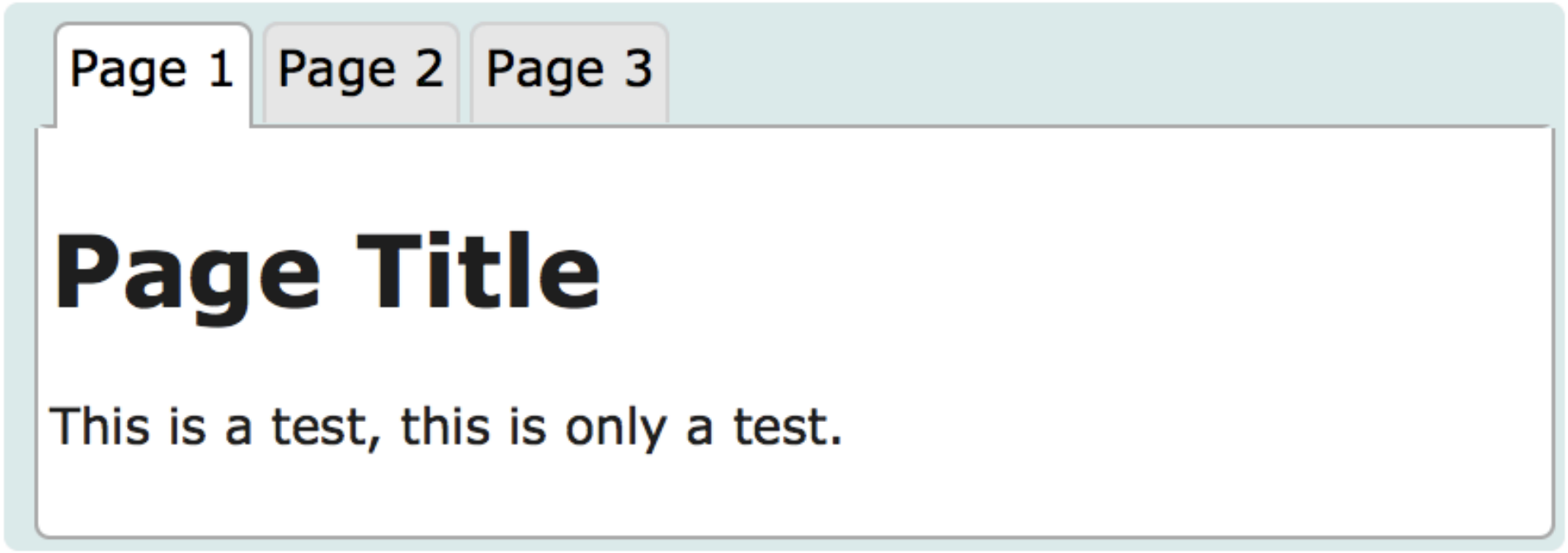
```
<book>
  <title>Acme Widget Reference</title>
  <price>13.95</price>
  <cover>http://example.com/cover.jpg</cover>
  <abstract>A reference to the Acme Widgets, what else?
    </abstract>
</book>
```

JSON: JavaScript Object Notation

- Native JavaScript notation
  - Nested name/value pairs
- Ubiquitous support in JavaScript clients

```
{
  "title" : "Acme Widget Reference",
  "price" : "13.95",
  "cover" : "http://example.com/cover.jpg",
  "abstract": "A reference to the Acme Widgets, what else?"
}
```

What is a web page?



What is a web page?

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Page Title</title>




</head>
<body>
  <div class="navbar">
    <ul>
      <li>Page 1</li><li>Page 2</li><li>Page 3</li>
    </ul>
  </div>
  <div class="body">
    <h1>Page Title</h1>
    <p>This is a test, this is only a test.</p>
  </div>
</body>
</html>
```

What is a web page?

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Page Title</title>




</head>
<body>
  <div class="navbar">
    <ul>
      <li>Page 1</li><li>Page 2</li><li>Page 3</li>
    </ul>
  </div>
  <div class="body">
    <h1>Page Title</h1>
    <p>This is a test, this is only a test.</p>
  </div>
</body>
</html>
```

An initial model

What is a web page?

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Page Title</title>


  <script type="text/javascript" src="/js/library.js"></script>
  <script type="text/javascript" src="/js/page.js"></script>
  <script type="text/javascript" src="/js/extra.js"></script>
</head>
<body>
  <div class="navbar">
    <ul>
      <li>Page 1</li><li>Page 2</li><li>Page 3</li>
    </ul>
  </div>
  <div class="body">
    <h1>Page Title</h1>
    <p>This is a test, this is only a test.</p>
  </div>
</body>
</html>
```

Scripts

What is a web page?

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Page Title</title>
  <link rel="stylesheet" type="text/css" href="/cs/base-style.css" />
  <link rel="stylesheet" type="text/css" href="/cs/page-style.css" />
  <script type="text/javascript" src="/js/library.js"></script>
  <script type="text/javascript" src="/js/page.js"></script>
  <script type="text/javascript" src="/js/extra.js"></script>
</head>
<body>
  <div class="navbar">
    <ul>
      <li>Page 1</li><li>Page 2</li><li>Page 3</li>
    </ul>
  </div>
  <div class="body">
    <h1>Page Title</h1>
    <p>This is a test, this is only a test.</p>
  </div>
</body>
</html>
```

Stylesheets

What is a web page?

summer school

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Page Title</title>
  <link rel="stylesheet" type="text/css" href="/cs/base-style.css" />
  <link rel="stylesheet" type="text/css" href="/cs/page-style.css" />
  <script type="text/javascript" src="/js/library.js"></script>
  <script type="text/javascript" src="/js/page.js"></script>
  <script type="text/javascript" src="/js/extra.js"></script>
</head>
<body>
  <div class="navbar">
    <ul>
      <li>Page 1</li><li>Page 2</li><li>Page 3</li>
    </ul>
  </div>
  <div class="body">
    <h1>Page Title</h1>
    <p>This is a test, this is only a test.</p>
  </div>
</body>
</html>
```

Stylesheets

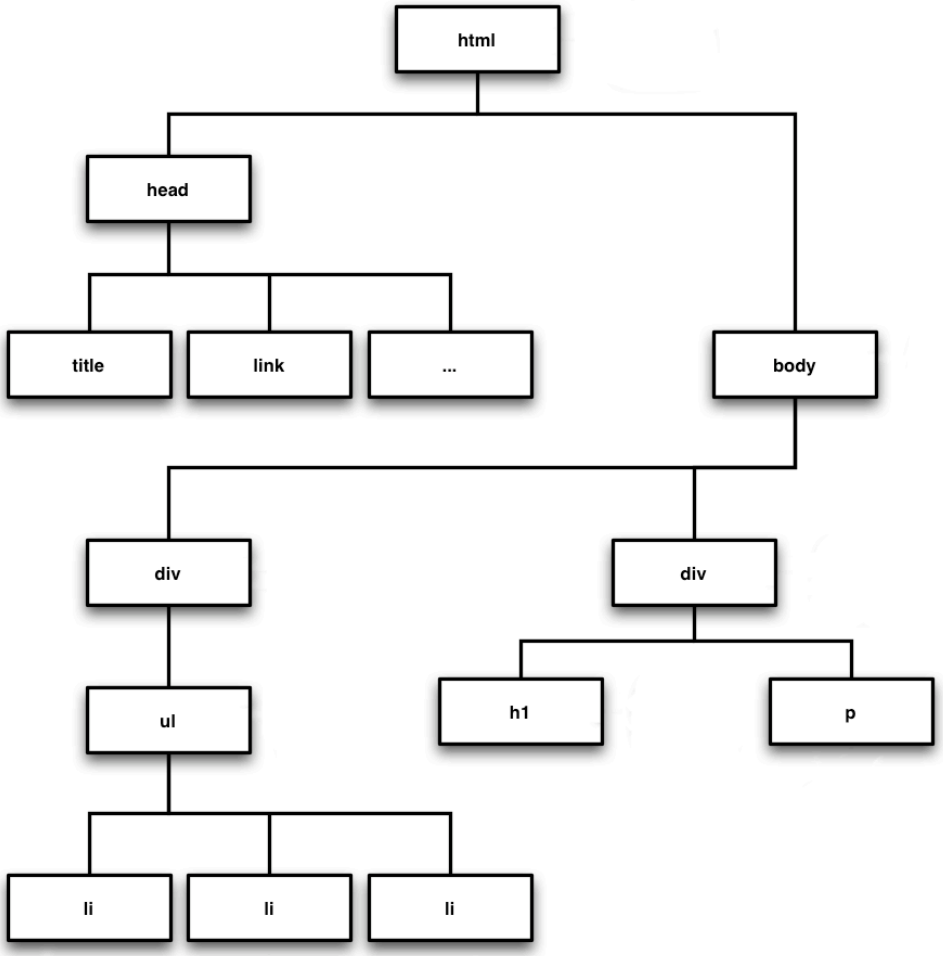
Scripts

An initial model

Why do Web 2.0 pages work?

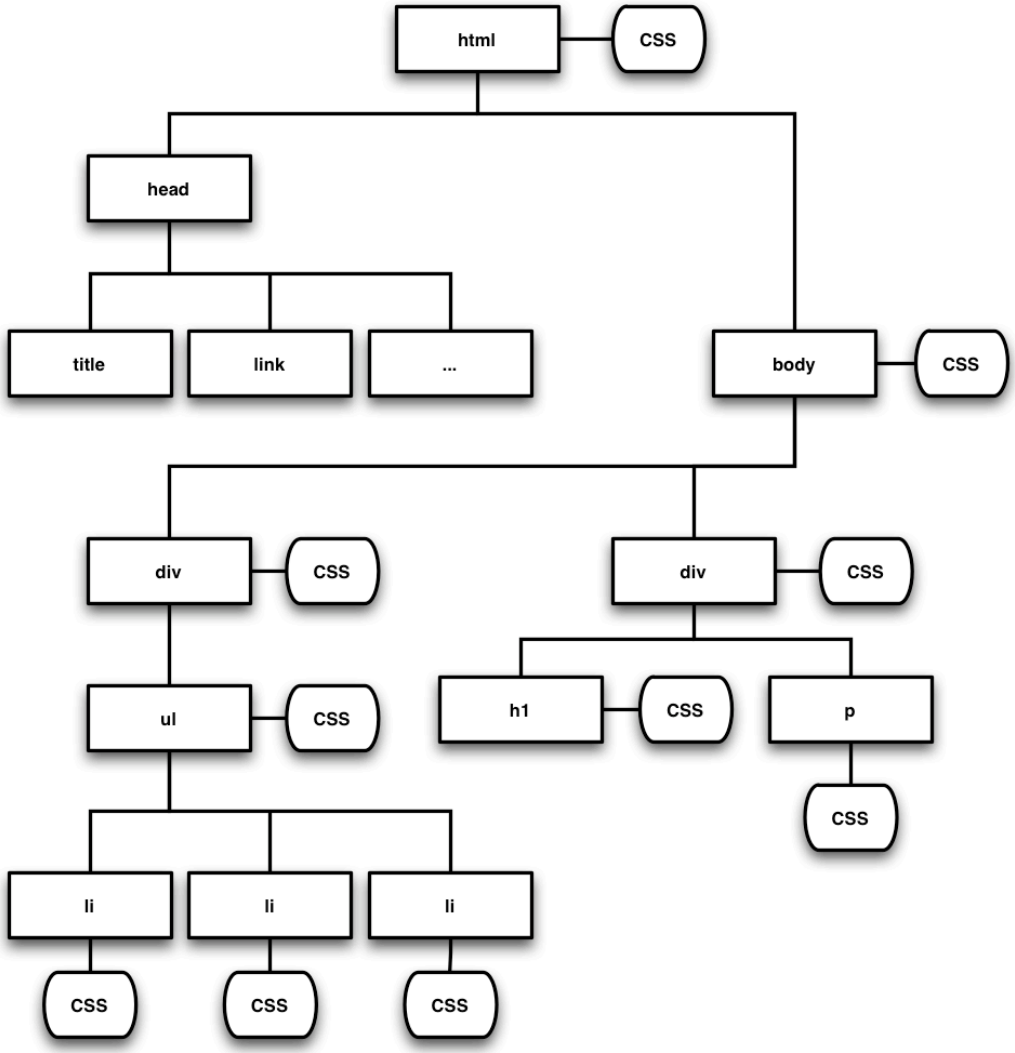
- Standardization
  - HTML
  - CSS
  - JavaScript
  - XML and/or JSON
- More-or-less universal browser support
  - Firefox
  - Safari
  - Opera
  - Internet Explorer

How do Web 2.0 pages work?



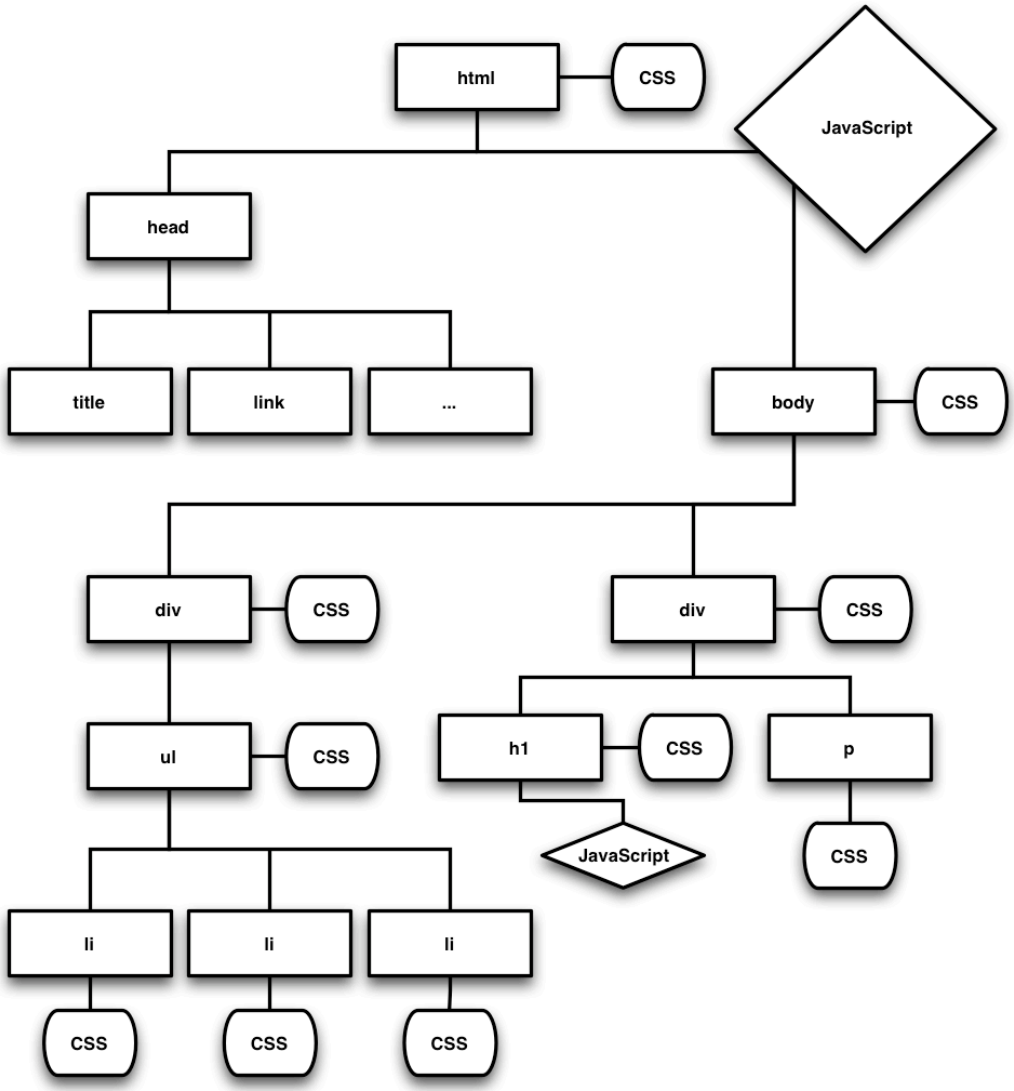
- Parse (X)HTML
- Build a DOM

How do Web 2.0 pages work?



- Parse (X)HTML
- Build a DOM
- Load CSS
- Attach styles

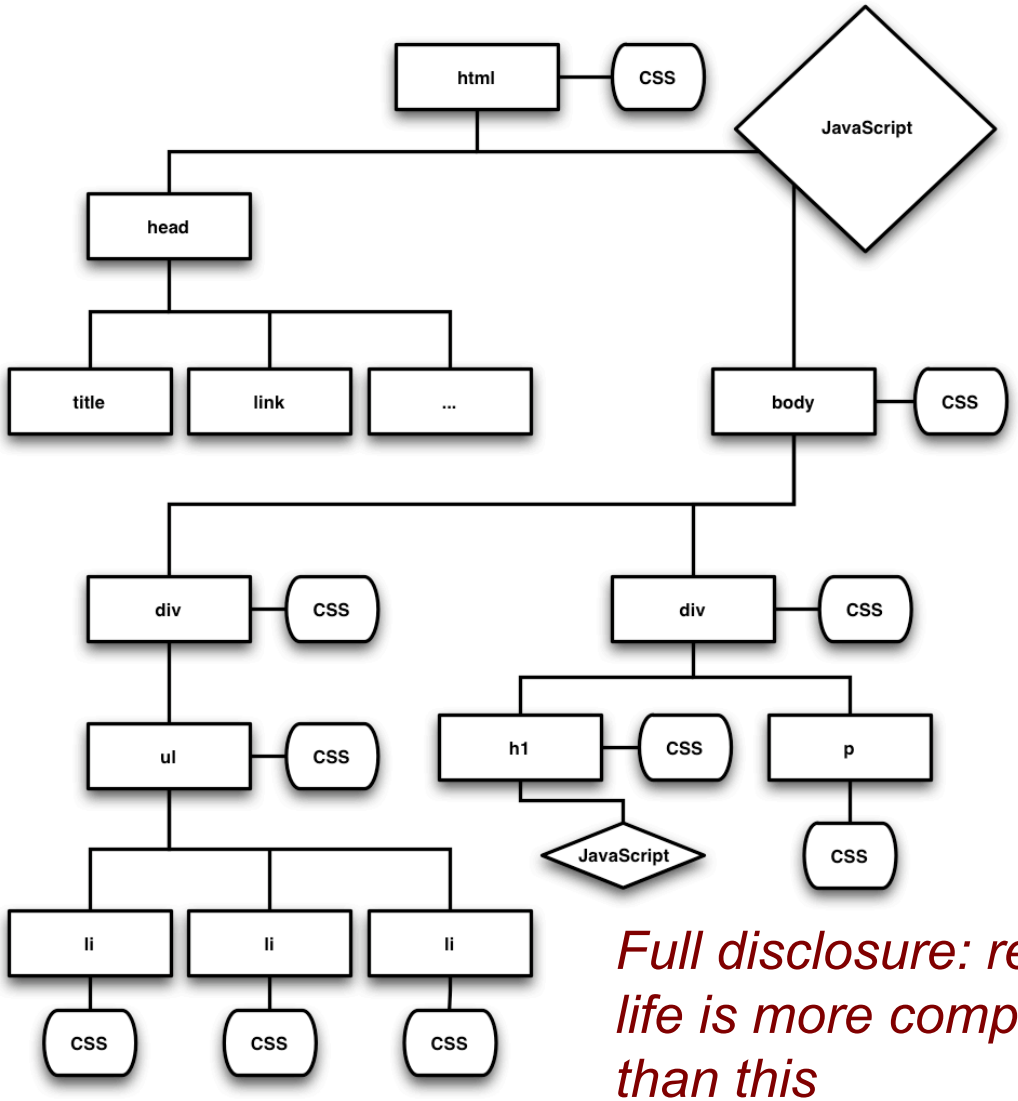
How do Web 2.0 pages work?



- Parse (X)HTML
- Build a DOM
- Load CSS
- Attach styles
- Load JavaScript
- Run scripts
- Attach event handlers

summer school

How do Web 2.0 pages work?



Full disclosure: real life is more complicated than this

- Parse (X)HTML
- Build a DOM
- Load CSS
- Attach styles
- Load JavaScript
- Run scripts
- React to events
  - Change DOM
  - Change Style
  - Attach new events
  - Repeat

How do Web 2.0 pages work?

- JavaScript gives the programmer almost complete freedom
  - Add or remove HTML elements
  - Add, remove, or change CSS properties
  - Move the input focus, change the cursor
  - Create new windows
  - Setup functions to run automatically in response to user events
    - Change of focus
    - Clicking or typing
  - Communicate with *the* server

summer school

How do Web 2.0 pages work?

- JavaScript gives the programmer almost complete freedom
- For security purposes, there are some limits:
  - The same origin policy
  - Browsers restrict or forbid “cross-site scripting” (XSS)

summer school

Doing two things at once

- The browser is “single threaded”; it can only do one thing at time
- How can it respond to multiple situations simultaneously?
- It can’t, but the JavaScript interpreter fakes it!

Doing two things at once

- The interpreter fakes it with:
- Events
  - Changes to the environment detected by the interpreter
  - "Handlers" are called automatically
  - When the handler finishes, the interpreter resumes whatever it was doing before
- And callback functions
  - Events by another name
  - Giving the interpreter a handler to call when some request is complete

summer school

Browser limitations

- Clients can do *a lot*, but
- Clients are ultimately limited
  - Little memory
  - Little horsepower
  - Slow connections to the network
- Don’t think so?
  - Think mobile phone, not desktop
- Servers are much less limited
  - I can upgrade the server
  - I can’t upgrade your desktop

summer school

Overcoming browser limitations

- The goal:
  - Allow the server to support my web application
  - Augment (not interfere) with the user experience

summer school

Asynchronous communication

Asynchronous communication (1)

- XMLHttpRequest
  - The magic behind the curtain
- Allows the client to construct an HTTP request
  - http://example.com/get-stock-quote?ticker=GOOG
- Asks the browser to perform the request
  - GET
- Provides a callback or event handler to process the response
  - updateStockTicker()

Asynchronous communication (2)

- Control returns to the browser while the request is being processed, so the application continues to function normally
- The callback function can insert the response into the users experience with the application in any appropriate way

XMLHttpRequest The Hard Way

- Create an array to hold requests, and build the infrastructure to initialize it.

```
var xmlReqs = new Array();
function ReqObj(type, xmlhttp) { ... }
function getReqObj() {
    var req = false;
    if (window.XMLHttpRequest) { // native branch
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) { // IE branch
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
    var reqobj = new ReqObj("busy", req);
    xmlReqs.push(reqobj);
    return reqobj;
}
```

XMLHttpRequest The Hard Way

- Write a function to submit the request

```
function loadXMLDoc(url) {
    var req = getReqObj();
    if (window.XMLHttpRequest) { // native branch
        req.xmlhttp.onreadystatechange = procReqChange;
        req.xmlhttp.open("GET", url, true);
        req.xmlhttp.send(null);
    } else if (window.ActiveXObject) { // IE branch
        req.xmlhttp.onreadystatechange = procReqChange;
        req.xmlhttp.open("GET", url, true);
        req.xmlhttp.send();
    }
}
```

XMLHttpRequest The Hard Way

- Write the callback handler and the function to process the values returned by the call

```
function processReqChange() {
    for (var i = 0; i < xmlReqs.length; i++) {
        if (xmlReqs[i].type == "busy"
            && xmlReqs[i].xmlhttp.readyState == 4) {
            handleResponse(xmlReqs[i].xmlhttp);
            xmlReqs[i].type = "ready";
}}}

function handleResponse(req) {
    if (req.status == 200) {
       var elem = document.getElementById(“someid”)
       // ...processing statements go here...
    } else { ... }
}
```

XMLHttpRequest The Easy Way

- Include a JavaScript framework, like jQuery
- Use it

```
$("#someid").get(url,
                 { param1: value1,
                   param2: value2 },
                 handleResponse);

function handleResponse(text,status,reqobj) {
  ...
}
```

summer school

JavaScript frameworks

JavaScript frameworks

There are **lots** to choose from:

- script.aculo.us
- Prototype
- jQuery
- MochiKit
- Railto
- Dojo Toolkit
- SproutCore
- midori

- My choice, having tried a few but by no means all of them, is jQuery

Why I like jQuery

- Robust
- Concise
- Active developer community
- Large inventory of extensions

CSS frameworks

- There are lots of CSS frameworks as well:
  - Blueprint
  - Baseline CSS
  - YUI Grid CSS
  - Elements CSS Frameworks
  - CleverCSS
  - Tripoli CSS Framework

summer school

Tutorial Session

Adding interactivity

Imagine that you've written a web service which accepts a pair of lat/long coordinates and returns a list of photographs contained within that rectangle

Write a client application that

- Allows users to enter coordinates
- Validates the coordinates
- Updates the page with a list of matching photographs

summer school

Any questions?

Thank you for your attention