Topics

• Basics: HTTP Get/Post
• Scripting
• Ajax
• HTML5, SVG
• Toolkits
• Server push (Comet)
• Going Browserless
Topics

• Basics: HTTP Get/Post
• Scripting
• Ajax
• HTML5, SVG
• Toolkits
• Server push (Comet)
• Going Browserless
HTML + HTTP

Classic web application model

Jesse James Garrett / adaptivepath.com
HTTP Get and Post

Both retrieve a web page

Get:
• Sends a URL (only) to the server
• Client data encoded as arguments in the URL
• **Idempotent (no change to server state)**

Post:
• Sends a “form” to the server at a specified URL
• Client and server have to agree on the format of this data
• **Causes change to server state (DB update)**
• Or to send large data blocks
HTTP Get


https://www.google.com/search?hl=en&q=http+get

HTTP Post

You can post data in any format, but the server and client need to know what it is.

Common examples:
• A HTML document (a “form” with fields filled in).
• An XML document.
• A URL-encoded string (like Get).
• JSON data.

POST /path/script.cgi HTTP/1.0
From: joe@joeblow.com
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
JavaScript Object Notation: A format for exchanging data with JavaScript engines.

A more compact, simpler alternative to XML (no Schema).

```json
{
  "employees": [
    { "firstName":"John" , "lastName":"Doe" },
    { "firstName":"Anna" , "lastName":"Smith" },
    { "firstName":"Peter" , "lastName":"Jones" }
  ]
}
```
JSON in Javascript

<!DOCTYPE html>
<html>
<body>
<h2>JSON Object Creation in JavaScript</h2>
<p>Name: <span id="jname"></span><br />
Age: <span id="jage"></span><br />
Address: <span id="jstreet"></span><br />
Phone: <span id="jphone"></span><br />
</p>
<script type="text/javascript">
var JSONObject = {
"name":"John Johnson",
"street":"Oslo West 555",
"age":33,
"phone":"555 1234567"};
document.getElementById("jname").innerHTML=JSONObject.name
document.getElementById("jage").innerHTML=JSONObject.age
document.getElementById("jstreet").innerHTML=JSONObject.street
document.getElementById("jphone").innerHTML=JSONObject.phone
</script>
Topics

- Basics: HTTP Get/Post
- Scripting
- Ajax
- HTML5, SVG
- Toolkits
- Server push (Comet)
- Going Browserless
1995: Javascript

- This supported a variety of richer client interaction (than clicking links) but server data still required page refresh and GET/POST.

Telerobotic camera control using client-side scripting
Javascript and Dom

<html>
<head>
  <title>The title</title>
</head>
<body>
The body
</body>
</html>

document.getElementById("body").innerHTML="Some new text"
Javascript Example

See [http://www.w3schools.com/js/default.asp](http://www.w3schools.com/js/default.asp)

```html
<!DOCTYPE html>
<html>
  <head>
    <script>
      function myFunction() {
        document.getElementById("demo").innerHTML="A JavaScript Function";
      }
    </script>
  </head>
  <body>
    <h1>My Web Page</h1>
    <p id="demo">A Paragraph</p>
    <button type="button" onclick="myFunction()">Try it</button>
  </body>
</html>
```
Javascript Limitations

- No widgets
- Browser-specific code!!

```javascript
function getXHR () {
    var request = null;
    if (window.XMLHttpRequest) {
        try {
            request = new XMLHttpRequest();
        } catch(e) {
            request = null;
        }
    } else if (window.ActiveXObject) {
        try {
            request = new ActiveXObject("Msxml2.XMLHTTP");
            // Try the older ActiveX object for older versions of IE
        } catch(e) {
            try {
                request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch(e) {
                request = null;
            }
        }
    }
    return request;
};
```
jQuery (2005)

http://jquery.com Major overlay API on Javascript. Perhaps 70% of Javascript sites use jQuery?

• Follows the HTML/CSS paradigm for web page structure.
• Provides browser-independent operations.
• Full Ajax support*.
• Animations.
• JQuery UI widgets: http://jquery.com/demos
Browser Runtimes

Almost all other Web Toolkits (later) use Javascript as the output script. Other common runtimes include:

- Adobe Flash
- Java
- Microsoft Silverlight
- Curl

But these require a plugin.

Once upon a time there was a large performance gap between Java and JS, but with intensive effort on JS JIT compilers, it has substantially closed. See:

Topics

• Basics: HTTP Get/Post
• Scripting
• Ajax
• HTML5, SVG
• Toolkits
• Server push (Comet)
• Going Browserless
Ajax
XMLHttpRequest (2000)

- Originally designed for Outlook Remote Access by Microsoft.
- XMLHttpRequest is an API that supports GET and POST requests to a web server, and makes the result available to a script.
- i.e. XMLHttpRequest allows general message-passing between client and server.

- XMLHttpRequest was the key technology of what became known as “Ajax” later.
XMLHttpRequest examples

http://www.w3schools.com/dom/tryit.asp?filename=try_dom_xmlhttprequest_first

http://www.w3schools.com/dom/tryit.asp?filename=try_dom_xmlhttprequest_suggest
Ajax in jQuery

Much cleaner thanks to encapsulation of browser-specific code in jQuery.

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_ajax1

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_ajax2
Topics

• Basics: HTTP Get/Post
• Scripting
• Ajax
• HTML5, SVG
• Toolkits
• Server push (Comet)
• Going Browserless
HTML5

Taxonomy & Status (December 2011)
- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated W3C APIs

By Sergey Mavrody 2011 | CC Attribution-ShareAlike 3.0
HTML5

Some important components:

• CSS3
• Canvas for drawing, SVG for animation
• Media, audio/video
• Web Storage
• WebGL, for 3D
CSS3 capabilities

http://www.w3schools.com/css3/tryit.asp?filename=trycss3_border-radius

http://www.w3schools.com/css3/tryit.asp?filename=trycss3_text-shadow_tut

http://www.w3schools.com/css3/tryit.asp?filename=trycss3_transition2

http://www.w3schools.com/css3/tryit.asp?filename=trycss3_animation3
HTML5 drawing

**HTML5 Canvas:**
- Draw directly on the screen.
- No “objects,” only pixels.
- Animation OK, but requires complete redraw.

**or SVG:**
- Is an XML format that you can edit via its DOM.
- Vector graphic form – display independent.
- Very rich animation possibilities.

SVG should be the better option for non-trivial animation.

**Performance?:** generally canvas drawing is faster, but depends on many factors.
```javascript
<script>
var c=document.getElementById("myCanvas");
{
    var ctx=c.getContext("2d");
    ctx.fillStyle="#FF0000";
    ctx.fillRect(20,20,100,50);

    var grd=ctx.createLinearGradient(140,20,240,70);
    grd.addColorStop(0,"black");
    grd.addColorStop(1,"white");
    ctx.fillStyle=grd;
    ctx.fillRect(140,20,100,50);

    var grd2=ctx.createLinearGradient(20,90,120,90);
    grd2.addColorStop(0,"black");
    grd2.addColorStop("0.3","magenta");
    grd2.addColorStop("0.6","blue");
    grd2.addColorStop("0.8","green");
    grd2.addColorStop(1,"red");
    ctx.strokeStyle=grd2;
    ctx.strokeText("Smile!",140,120);
}
</script>
```

```javascript
var grd3=ctx.createLinearGradient(140,20,240,90);
grd3.addColorStop(0,"black");
grd3.addColorStop("0.3","magenta");
grd3.addColorStop("0.6","blue");
grd3.addColorStop("0.8","green");
grd3.addColorStop(1,"red");
ctx.strokeStyle=grd3;
ctx.strokeText("Smile!",140,120);
```

```javascript
var grd2=ctx.createLinearGradient(20,90,120,90);
grd2.addColorStop(0,"black");
grd2.addColorStop("0.3","magenta");
grd2.addColorStop("0.5","blue");
grd2.addColorStop("0.6","green");
grd2.addColorStop("0.8","yellow");
grd2.addColorStop(1,"red");
ctx.fillStyle=grd2;
ctx.fillRect(20,90,100,50);
```
SVG

Inline XML that describes graphic elements

```html
<html>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" fill="red" />
</svg>
</body>
</html>
```
HTML5 graphics

Some advanced examples:

http://www.jswidget.com/ipaint.html

https://tinkercad.com/
Topics

• Basics: HTTP Get/Post
• Scripting
• Ajax
• HTML5, SVG
• Toolkits
• Server push (Comet)
• Going Browserless
Toolkits: Gware

**Closure Toolkit:** an alternative Javascript toolkit to jQuery
- Includes an optimizing “compiler” to javascript.
- Somewhat richer widget library than jQuery UI.
- Templates for common designs.

**GWT:** Google Web Toolkit:
- Develop web applications in Java, APIs + UI widgets.
- Compiles into Javascript for the client.
- Eclipse support, GWT designer.
- Jetty server, concurrent client/server debugging.
GWT examples
Toolkits: Python

**Pyjamas**: A framework for web app. Development entirely in Python. Includes:

- Python-based API, with compilation into Javascript.
- Ajax library.
- GUI widget set.
- Pyjamas Desktop: a library that allows Pyjamas apps to run browserless.

Pyjamas is a port of GWT to Python.
Toolkits - Dojo

- A component-oriented Javascript framework.
- Rich, high-level widget library.
- Eclipse support, and an open-source WYSIWYG editor: Wavemaker Visual Ajax Studio.
- Lucid desktop supports desktop apps.
- CometD web server.
- Development tools: Marquetta, General Interface: http://dojofoundation.org/projects/
  - http://demos.dojotoolkit.org/demos/
Toolkits

Other tools you know about? Pros and Cons?
Topics

• Basics: HTTP Get/Post
• Scripting
• Ajax
• HTML5, SVG
• Toolkits
• Server push (Comet)
• Going Browserless
Comet: Server Push

• In standard HTTP and Ajax, all interactions are initiated by the server via GET, POST, etc.

• They don’t support live updates from the server:
  – Instant messaging
  – Shared calendars
  – Email updates

• Comet is the (relatively new) “push” part of Ajax.

• It was adopted widely in 2006.
CometD (web server) is an implementation of Comet in Java, and should be good for testing push applications.

Its part of the Dojo project.
Topics

- Basics: HTTP Get/Post
- Scripting
- Ajax
- HTML5, SVG
- Toolkits
- Server push (Comet)
- Going Browserless
Browserless Runtimes

- Adobe AIR (Webkit)
- Microsoft Silverlight 3+
- Google Chrome OS (runs on non-Chromebooks as well)
- Apple OS X Lion Automator
- JavaFX
- Lucid Desktop, Pyjamas Desktop,…

What they do:
- Allow apps to run outside a browser
- Provide shortcuts and access through normal application menus
- Open up access to the local filesystem, local devices,…
Adobe AIR

Adobe Integrated Runtime (AIR), released in 2008:

- Supports Adobe Flash, Apache Flex, and HTML/Ajax (via Webkit) applications on the desktop or in a browser.
- App still needs to be “installed” locally, and signed.
- Most standard web apps will run, but custom code is needed to:
  - Access the local file system
  - Use taskbar/dock features
  - Access sensors like GPS, accelerometer, touchscreen
- Some extra security restrictions, but works with jQuery, Dojo, and MooTools.
- Adobe claims > 100 million installations (compulsory install with Adobe Reader 9).
Balsamiq Mockups for Desktop (AIR)
Perspective

• Web applications have expanded enormously from hypertext link-following in 1994.

• Scripting allowed progressively richer experiences in the browser, rivaling native interaction.

• Server innovations have allowed the application “back end” to migrate comfortably into the network.

• HTML5 and its component standards has great potential to become a true standard for content design (WebKit helps).
Perspective

• No standard of complexity comparable to HTML 5 has ever succeeded (Flash/Flex lives!).

• There remains and “Uncanny valley” between desktop (or browser) web applications and native applications.

• Cloud-based services are great for providers, but for consumers?