



TEKsystems Education Services

presents

HTML5 for Developers: Beyond the Basics

Copyright

This subject matter contained herein is covered by a copyright owned by:

Copyright © 2014 Robert Gance, LLC

This document contains information that may be proprietary. The contents of this document may not be duplicated by any means without the written permission of TEKsystems.

TEKsystems, Inc. is an Allegis Group, Inc. company. Certain names, products, and services listed in this document are trademarks, registered trademarks, or service marks of their respective companies.

All rights reserved

20750 Civic Center Drive
Suite 400, Oakland Commons II
Southfield, MI 48076
800.294.9360

TA440-SG / 9.24.14





HTML5

The HTML5 Ecosystem

Course Objectives

Provide an overview of the **HTML5 ecosystem**
including **CSS3** and **ECMAScript 5**

Examine **browser** capabilities and
recommended best practices

Provide hands-on exposure to **HTML5 markup**
and **JavaScript APIs**

Course Agenda

Day 1

- HTML5 and Enterprise JavaScript Overview
- Cross-Browser Development
- HTML5 Markup
 - **Recommended Practices**

Course Agenda

Day 2

- CSS Layout
- CSS 3
- JavaScript Libraries, jQuery

Course Agenda

Day 3

- HTML5 and Related JavaScript APIs
 - Selector API
 - Local Storage
 - Cross-Origin Requests
 - Geolocation APIs
 - Web Workers
 - Web Sockets
 - Server-sent Events
 - Application Cache
- Introducing ECMAScript 5

Introductions

- Name
- Where you're from
- What you work on
- Reason for attending



Course Logistics

Typical Daily Schedule *

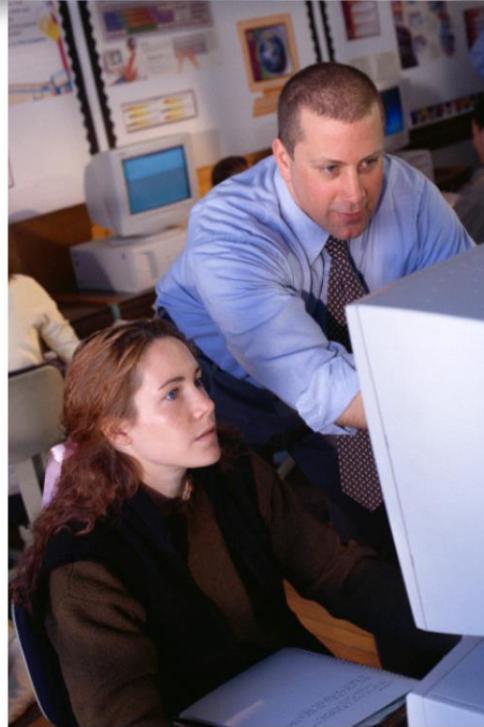
8:30	Start Day
9:45	Morning Break
11:30 – 12:30	Lunch
1:45	Afternoon Break 1
3:15	Afternoon Break 2
4:30	End of Day

* your schedule may vary

Get the Most...

...from your experience

Ask Questions!





A photograph of a person from the side, wearing a black headset with a microphone. They are looking down at a dark computer keyboard. The background is blurred with blue and white streaks, suggesting motion or digital data.

Introducing Node.js

Overview

node.js overview

Setup, install, testing

Creating an app server

node.js overview

- node.js is a *server-side* JavaScript framework
 - Contains the Google V8 JS Engine
- Created by Ryan Dahl in 2009 and sponsored by Joyent
- Used in this course as our server!

Obtain and Install

- Visit nodejs.org and click on the install option
 - Download the .msi for windows and run it!



For Windows-based installation, simply click the install button shown, run the .msi file and allow it to set your PATH variable to the node.exe and npm.exe files.

Note: for Eclipse users, there is a plugin that may be installed to facilitate working with it called Nodeclipse: <http://www.nodeclipse.org/>
It can be installed in about 7-10 minutes.

An Initial App

This example creates an HTTP server in a few lines of code

require() can import various modules used

```
var http = require("http"),  
  
    handleRequest = function(request, response) {  
        var resp = 'Thank you for your response.';  
        response.writeHead(200, {"Content-Type": "text/plain"});  
        response.write(resp);  
        console.log(resp);  
        response.end();  
    },  
  
    server = http.createServer(handleRequest);  
  
server.listen(8005);
```

Builds a response

Wait for requests from the browser

node/example01.js

Test your installation by running **node example01.js** and then browsing to localhost:8005

Reading files: Node is Asynchronous

```
var http      = require("http"),
    fs        = require('fs'),
    respData = '',
    handleRequest = function(request, response) {
        fs.readFile('sample.txt', 'utf-8', function(err, data) {
            if (err) respData = err;
            else respData = data;

            response.writeHead(200, {"Content-Type": "text/plain"});
            response.write(respData);
            console.log(respData);
            response.end();
        });
    },
    server    = http.createServer(handleRequest);

server.listen(8005);
```

This example attempts to read data from a file and send it to the browser.

This version properly displays the data found in sample.txt

nodejs/example02.js

Reading files from the file system is asynchronous and when finished will invoke the callback function (the anonymous function in the fs.readFile() call).

To process the data read from the file, you must do it in the callback function.

The example works because the callback function is properly invoked after file contents have been read.

Installing 3rd Party Modules

- To install 3rd party modules, use the *node package manager* (npm):

```
npm install <module_name>
```

- A list of 3rd party packages can be found at the npm repository:



The node package manager is similar to Red Hat's rpm utility or Python's pip tool or Ubuntu's apt-get command.

Creating Custom Node Modules

example03.js

```
var http      = require('http'),
    ex04      = require('./example04'), ←
    server     = http.createServer(ex04.handleRequest);

server.listen(8000);
```

example04.js

```
var fs        = require('fs'),
    respData = '',
    handleRequest = function(request, response) {
        fs.readFile('sample.txt', 'utf-8', function(err, data){
            if (err) respData = err;
            else respData = data;
            response.writeHead(200, {"Content-Type": "text/plain"});
            response.write(respData);
            console.log(respData);
            response.end();
        });
    };

exports.handleRequest = handleRequest;
```

Here, example03.js tries to read the server module found in the other module, example04.js.

Any items not exported will be private to that module.

Order in Which Modules are Loaded

- Internal modules provided by node (found in <nodejs>/lib) have the highest priority
 - Examples: 'http', 'fs'
- Modules in paths using /<module> or ./<module> or ../<module>
- Modules found in the *node_modules* directory (*underneath where the current module exists*)
- Modules in the *node_modules* directory of the parent directories of the current module

Modules are loaded in the order specified here. Built-in modules have the highest priority, followed by modules of a specified path. Modules within a *node_modules* sub-directory where the current module is currently being executed will be searched followed by the parent directories (looking for a *node_modules* named directory) of the current module.

Debugging Node Apps

- Use node-inspector within Chrome to debug nodejs apps

```
$ npm install -g node-inspector
```

- Start the node in debug mode:

```
node --debug index.js
```

- Debug from within Chrome browser

For more on node-inspector, visit: <https://github.com/dannycoates/node-inspector>

The Jetbrains WebStorm IDE also has integrated node debugging support.

Eclipse and Node



Nodeclipse

<http://www.nodeclipse.org>

- Eclipse also supports the installation of a Node.js plugin
- To install, simply drag the install icon onto Eclipse's toolbar



- Easily run/debug node apps using familiar Eclipse-style controls

Nodeclipse plugin for Eclipse installs in about 3-4 minutes.

Using Express

- expressjs is a 3rd party add-on module for nodejs
 - Provides simple mappings for request / controller logic

```
var express = require('express');
var app = express();

app.get('/', function (req, resp) {
    resp.setHeader('Content-Type', 'text/plain');
    resp.end(body);
});

app.listen(8005);
```

Great for building REST-style apps

Node, Express, and REST

```

var express = require('express'),
empdata = require('./empdata.js'),
emps     = empdata.empdata.emps,
app      = express(), port = 8005;

app.use(express.directory("public"));
app.use(express.static("public"));

app.get("/employee/:empid", function (req, resp) {
    var empid = req.params.empid,
        results = {"error" : "Not found."};

    resp.setHeader("Access-Control-Allow-Origin", "*");

    emps.forEach(function(emp){
        if (emp.empid == empid)
            results = emp;
    });
    resp.json(results);
    resp.end();
});

app.listen(port);

```

Expressjs can support
RESTful URLs

To run: run server.js in the
nodejs/example05 folder.
Test in browser at:localhost:8005/index.html

This example can be found in server.js in the example05 folder of the nodejs directory. It loads employee data (empdata.js) and receives GET requests from the user on port 8005. It will then allow ajax requests to be sent using the /employee/:empid syntax which allows a variable to be specified and encoded into the request object.



Before we get started!...

...let's set up our environments

Perform Lab 0 (Exercise 0)

Environment Setup

Locate the instructions for this exercise in the Student Exercises Workbook in the back of the manual



HTML5

Overview and Architecture

Overview

HTML 5 Overview

Front-End Architectures

Cross-Browser Development

A Need for HTML5

- The front-end has lacked major updates for over a decade



HTML 4.01 has remained largely the same for over a decade. During this time, the markup has been used and misused in so many ways that it is evident a new markup is needed to support the current demands of page development.

CSS 2.1 has gone from a working draft to candidate recommendation within the W3C, and back to working draft over the last six years. Nevertheless, CSS 2.1 has merely served as an update fixing minor errors within the CSS 2 specification. The major features of CSS have largely remained unchanged since 1998.

ECMAScript underwent 3 revisions from 1997 until 1999. After that, little occurred in the specification. Other features, such as the XMLHttpRequest, have appeared since, but these are a result of widespread adoption by numerous browser vendors and not as a result of updates to any specifications. Disagreements as to how ECMAScript 4 should look led to the formation of ECMAScript Harmony and the now abandoned ECMAScript 4. ECMAScript 5 was released in December of 2009. It is beginning to pick up support within the current generation of browsers.

Why Move to HTML5?

- "Paving the cowpaths"
- Improved semantics
 - <div> not needed to lay out entire page
- Native browser support for better performance

HTML5 tries to "pave the cowpaths", meaning it attempts to formalize the approaches developers have taken over the years to create well-organized, layered architectures. In their attempts, patterns such as <div class="header">, etc.

The new HTML grammar will provide better semantics, meaning a <div> doesn't have to be used to describe the layout of an entire page now.

It is expected that the incorporation of HTML5 and related JavaScript APIs into the browser will release developers from having to write their own code to perform simple tasks (such as Geolocation or DOM path searches, etc.). Developers will be able to rely more on the native features provided by the browsers, which should provide better performance results.

What is HTML5?

- HTML5 is the standard for next generation front-end development
- CSS3 falls outside the HTML5 standard
 - Considered part of HTML5 "ecosystem"
- Many JavaScript APIs are also being lumped into this "ecosystem"

HTML5 ≈ HTML5 Tags + CSS3 + New JS APIs

While CSS3 technically existed even before HTML5 was on the horizon, today many consider CSS3 part of the larger group of specifications that are sometimes dubbed the HTML5 "ecosystem".

The HTML5 "Ecosystem"

- HTML5 markup
 - New Semantic HTML Tags
 - New tags for laying out pages
 - New forms capabilities
 - Input types, validation
- CSS3
 - Powerful transformations, animations, effects
- JS APIs
 - Geolocation, local storage, web workers, web sockets, other APIs...

The HTML5 Ecosystem includes items that are not officially part of the standard. However, the inclusion of CSS3 and many different JavaScript APIs have created the "illusion" of a larger approaching overall standard.

The Evolving Standard



- Currently in **W3C Proposed Recommendation Status (16 Sep 2014)**
- Currently in Draft Standard Status in **WHATWG**



The HTML5 standard will require many years to formalize. The fact that two standards bodies have undertaken the HTML5 standardization process has caused much political in-fighting and has created a process-inhibiting slowing of progress.

The W3C in 2013 decided to "freeze" the HTML5 standard while the WHATWG wanted it to continue as a living document standard. The result is that now HTML5.1 continues to evolve while HTML5.0 was frozen.

Shown at right is the logo introduced by the W3C for HTML5.

Browser Support for HTML5

- Browsers currently implement **varying** levels of HTML5



There are a number of sites that provide matrices, summaries, or charts of browser support. Some of these sites include:

html5demos.com

html5readiness.com

findmebyip.com/litmus

en.wikipedia.org/wiki/Comparison_of_web_browsers

The one shown here provides information current through the latest release candidate of IE11.

HTML5 Status Update

- May 2013 - **HTML 5.1 draft released**
- **HTML5 features largely "frozen"** while new features have been added to HTML5.1
- **Example 5.1 considerations include:**
 - **<hgroup> dropped**
 - **<time> dateTime attribute should be spelled with 'T' in the DOM**
 - **New preferences for <h1> usage**
 - **New Canvas API methods**
 - **Consideration for <picture> and <main> elements**

Recommendation
Status expected
2016

A few notes about the standards bodies: WHATWG looks at HTML5 as a "living standard" that may evolve while the W3C wants concrete specs. So, HTML5 was more or less "baselined" or feature frozen. HTML5.1 was introduced to "align" with the WHATWG version of the HTML5 standard.

<h1> usage is discussed later in this chapter.

Visit here for more on the plans for formalization of HTML5:

<http://dev.w3.org/html5/decision-policy/html5-2014-plan.html>

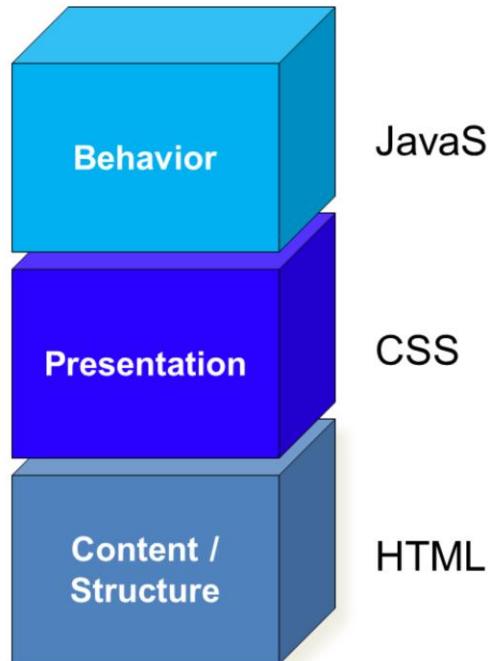
Overview

HTML 5 Overview

Front-End Architectures

Cross Browser Development

A Layered Architecture



An architecture makes things more manageable when developing complex applications - analogous to MVC on the server side

Why Consider a Layered Architecture?

- Attempts to support all user-agents (browsers)
 - Uses a universally understood markup document
 - Layers additional stylesheets and scripts as supported by the browser
- Allows modern devices to take advantage of newer features
- Provides structure for future design strategies

Structural Content: Semantic HTML

- Applies **meaningful HTML** tags to the content
 - The most appropriate HTML tag should be chosen to encapsulate the content



```
<table>  
  <tr id="head"><td>...</td></tr>  
  <tr id="body"><td>...</td></tr>  
  <tr id="footer"><td>...</td></tr>  
</table>
```



```
<div>  
  <div class="hd">...</div>  
  <div class="bd">...</div>  
  <div class="ft">...</div>  
</div>
```

- Elements are only used if they provide **meaningful structural context** to the content
- Elements shouldn't be used for presentation

Why Semantic HTML?

- Use of semantic HTML improves:
 - SEO rankings
 - Ease of maintenance (no tag soup)
 - Page size/load time
 - Accessibility
- Semantic HTML forms the basis of
“Progressive Enhancement”

A semantically-driven approach to HTML increases useful life-span of that HTML

“structural semantic markup provides lean, meaningful, accessible pages” – natekoechley

Semantic HTML Elements

Tag	Description
<div>	Divides page into sections
, , 	List of items including menus
<h1-h6>	Places importance on text-based content
<p>	Paragraphs of text
<abbr>, <acronym>, <cite>, , , , <code>	For inline text-use. Use depends on content.
<table>, <thead>, <tbody>, <tfoot>, <tr>, <td>, <th>, <caption>	For tabular display of content

Don't Use These Tags

Presentational Elements

b
br
big
hr
i
small
tt

Deprecated Elements

basefont
center
dir
font
isindex
menu
s
strike
u

Hopefully, you are already avoiding most of these tags. Note that some of these tags (like br, hr, small, and b) are being "re-introduced" in HTML5. More on this later. For now, understand that under HTML 4.01, these elements are all considered poor semantic choices.

Don't use presentational attributes either.

Div's & Span's

<div>

“division” - provides a way of dividing a document into **meaningful block-level** areas

a generic mechanism for adding **in-line** structure to documents

Block vs. Inline Elements

- HTML Elements are rendered using box model properties depending on whether an element is a **block** element or an **inline** element

Block Elements

- div
- ul
- li
- form
- h1-h6
- p

Block elements fill the width of their parent, next one renders beneath last

Inline Elements

- span
- input
- a
- img

div

h1

p

span

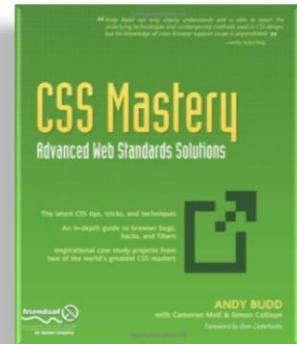
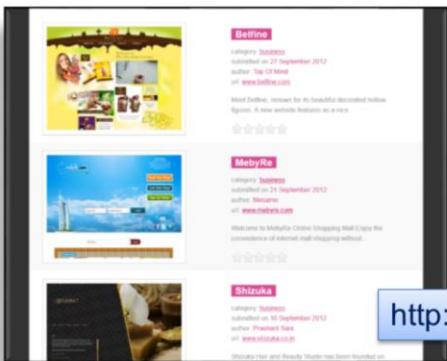
em

span

Inline elements render side-by-side, size to their content, and cannot control vertical properties such as **height** and **margin-top/bottom**

The Presentation Layer

- *Cascading Style Sheets* define the look-and-feel of an application
 - Separation of content and presentation is instrumental in ensuring responsive, maintainable designs



<http://www.css-showcase.com/en/business.html>

CSS Zen Garden

<http://www.csszengarden.com/>

- Great example illustrating
 - Semantic naming/structure
 - Separation of content & presentation
- When creating a layered architecture, ask yourself:

"How would this solution hold up on CSS Zen Garden?"



CSS Zen Garden
'Under the Sea' Theme

The Behavioral Layer

- Enterprise JavaScript source should be
 - unobtrusive
 - standards-based
 - cross-browser/platform/device capable
- Modern solutions also incorporate:

A JavaScript Library
(jQuery, Dojo, YUI,
ExtJS, Prototype, ...)

A Dependency Management Library (requirejs, Bootstrap, StealJS, LabJS, ...)

An MV* Framework (Backbone, JVMC, Angular, Knockout, Sammy, ...)

A Templating System (Handlebars, Mustache, Underscore Templates, EJS, jquery-tmpl, ...)

Inter-module Communication frameworks (jquery-pubsub)

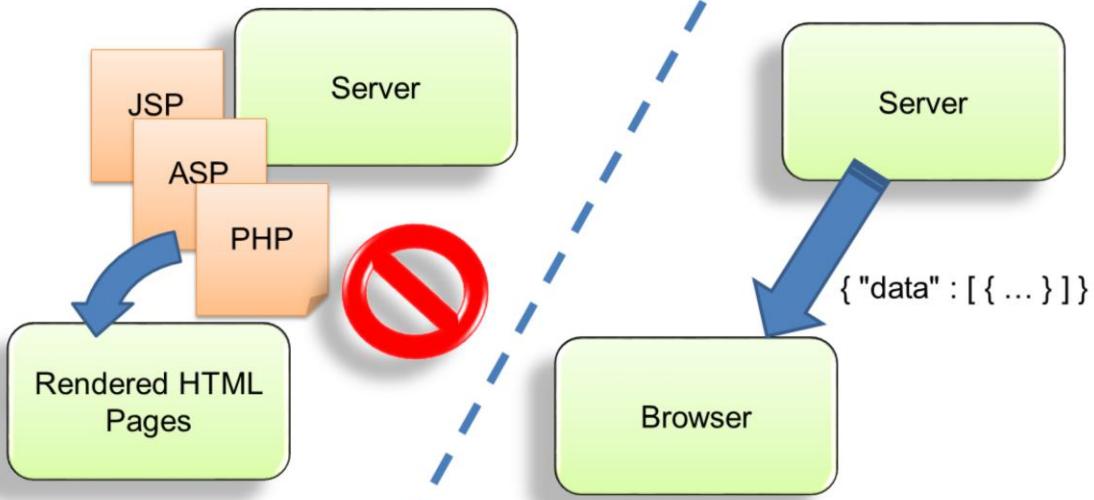
Testing Environments (FuncUnit, qUnit, Selenium, JSUnit, Jasmine, SinonJS, DOH)

Concatenation, Minification and Build Systems Tools (Bootstrap, steal.build, steal.clean)

Enterprise JavaScript is so named because it involves the use of frameworks that may not normally come up in smaller development environments. Where larger teams become involved in application development, testing, builds, and development environments become more important.

Modern Thin Clients

- Modern browser-based architectures are heavily *service-driven* rather than page-oriented



Modern day thin-client solutions, especially those that exhibit the single-page, desktop-style application model, heavily emphasize service-oriented architectures. The data exchange format is often via JSON (JavaScript Object Notation) or perhaps JavaScript data or even proprietary data formats. While XML is not commonly used for most browser-based implementations, it can be selected as a data communications format.

Browser Rendering Modes

- IE6 engineers wanted a browser that was more standards compliant but also backwards compatible
- Built-in two distinct “browser modes”:
 - **Standards** - renders according to spec
 - **Quirks** - for rendering legacy code built for older versions of IE
- Others followed
 - Mozilla-based browsers in “quirks mode” emulate Netscape 4 rendering style

Source: meyerweb.com

Doctype Switch

- Allows developers to control browser's rendering mode
- It's a hack!—
 - But it works
 - And is formally accepted as a means of controlling the browser
- Doctype/Browser Mode chart:
 - <http://hsivonen.iki.fi/doctype/>

Which Mode Am I In?

Which mode is your browser currently rendering in?

- Type the following into the address bar:

```
javascript:alert(document.compatMode)
```

- **BackCompat** = Quirks Mode
- **CSS1Compat** = Standards Mode

Try it for yourself!

Note: in FF6+, the address bar cannot
be used to execute JavaScript. Use the
scratchpad (Shift-F4) for this

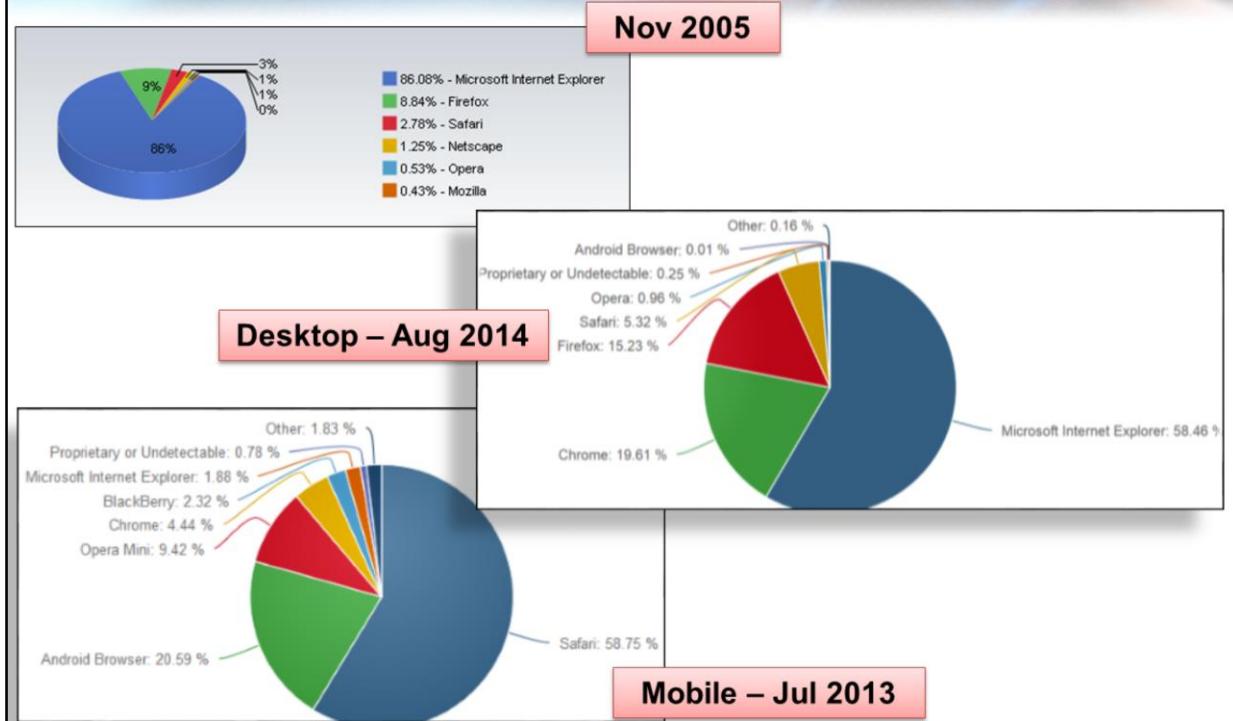
Overview

HTML 5 Overview

Front-End Architectures

Cross-Browser Development

Cross-Browser Design



Source: marketshare.hitslink.com

Note the changes in desktop browser market share over the last 6 years. No longer should solutions be written for single-browsers. Even in situations where applications are targeted for internal clients whose browsers are strictly controlled, careful consideration should be given to targeting that single browser.

Cross-Browser Designing

- Based on concept of Progressive Enhancement
- Three grades of support:
 - **C-grade**
 - Provides core content and functionality
 - Browsers are “incapable, antiquated, or rare”
 - **A-grade**
 - Provides advanced functionality and visual fidelity
 - Browsers are “modern, generally adhere to standards”
 - **X-grade**
 - Provides support for unknown, rare or fringe browsers
 - Browsers “assumed to be capable”
- <http://developer.yahoo.com/yui/articles/gbs>

Progressive Enhancement

- *Creating the best possible solution for the widest number of users*
- Approach:
 - Develop a base level solution that puts content at the center of the design
 - Layer enhancements on top by adding presentational and behavioral features
- Maintains some (limited) functionality for older browsers
- Newer browsers see an enhanced environment

There are numerous examples of progressive enhancement in action. A YUI datatable widget is a commonly cited one because it transforms a basic HTML table (seen by all browsers) into a stylized, interactive one for browsers that support it.

Other examples of progressive enhancement include the use of CSS rounded corners, where IE browsers (8 and earlier) would see squared corner solutions while other browsers could see the more visually appealing rounded corner versions (assumes use of CSS rounded corner properties and not use of background image techniques).

Progressive Enhancement Example



www.yahoo.com

Secret to keeping guacamole green

Leaving the pit in only keeps part of the avocado from browning, and using lemon juice doesn't do much, either. ['Genius' trick »](#)

1 – 5 of 90



www.yahoo.com in IE6-8

Secret to keeping guacamole green

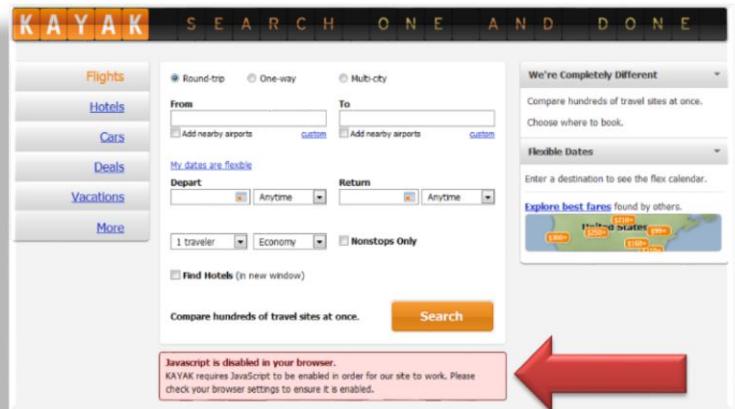
Leaving the pit in only keeps part of the avocado from browning, and using lemon juice doesn't do much, either. ['Genius' trick »](#)

1 – 5 of 90

Web 2.0 and Points of Failure

- The Web 2.0 era has spawned many poor practices:
 - Focusing on a single browser for development
 - Requiring JavaScript (ex: Kayak.com)
 - Try *nike.com* with JavaScript disabled

- Reliance on plug-ins
- Reliance on CSS to display content
 - How does it work on Kindle?



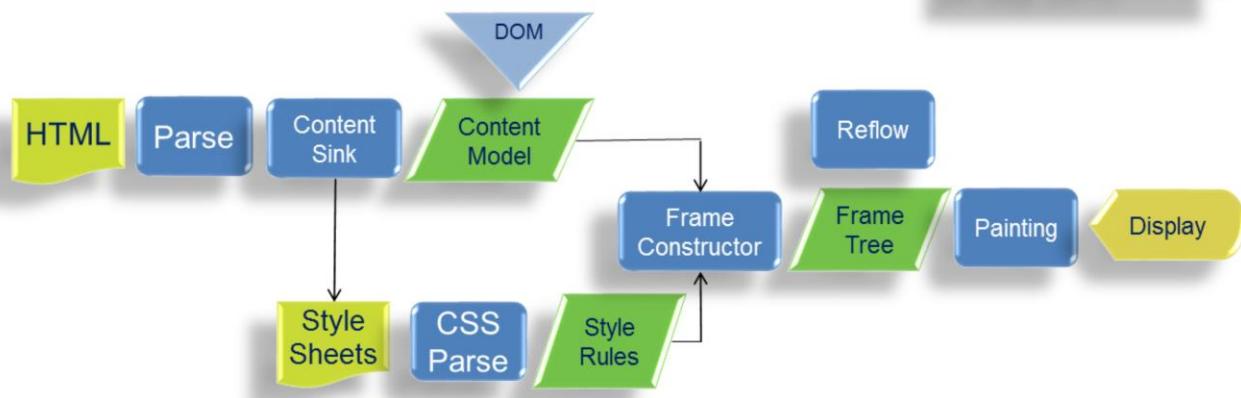
The pie chart shown earlier provides evidence that focusing on a single browser for dev purposes will lead to failure in production. Popular public sites, like Kayak.com require JavaScript to be turned on. Reliance of plugins such as Flash or Silverlight will most certainly cause problems in the mobile world. Assuming that devices have every plugin available and that they will most certainly have YOUR plugin available is a bad assumption. Relying on CSS to ensure content can be read will lead some devices not to work or display properly. Kindle? Nook?

Firefox



- Firefox moved to a 16-week release cycle (early 2011)
 - Spurred by rapid changes in HTML5/CSS3/ECMAScript 5 specs

Version releases:
4 Mar 2011
8 Nov 2011
10 Jan 2012
12 Apr 2012
16 Oct 2012
24 Sep 2013



www.getfirefox.com

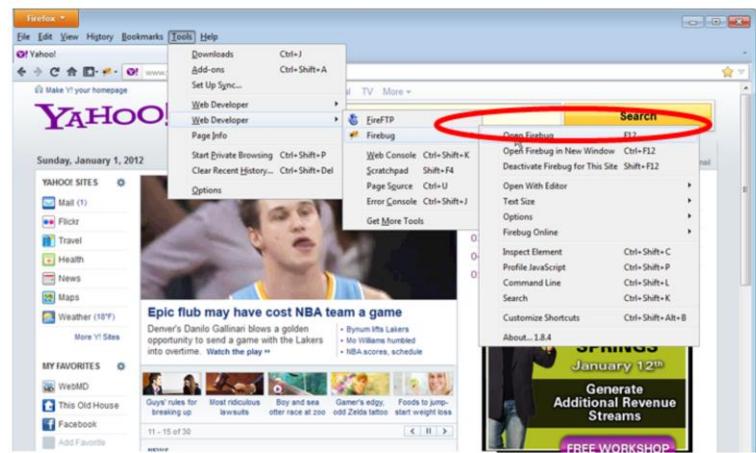
The diagram shows the operations that take place when a document is loaded by Firefox. The version shown is for 3.6. Reference: <http://www-archive.mozilla.org/newlayout/doc/gecko-overview.htm> (slide 9).

A detailed description of how browsers work can be found here:
<http://taligarsiel.com/Projects/howbrowserswork1.htm>.

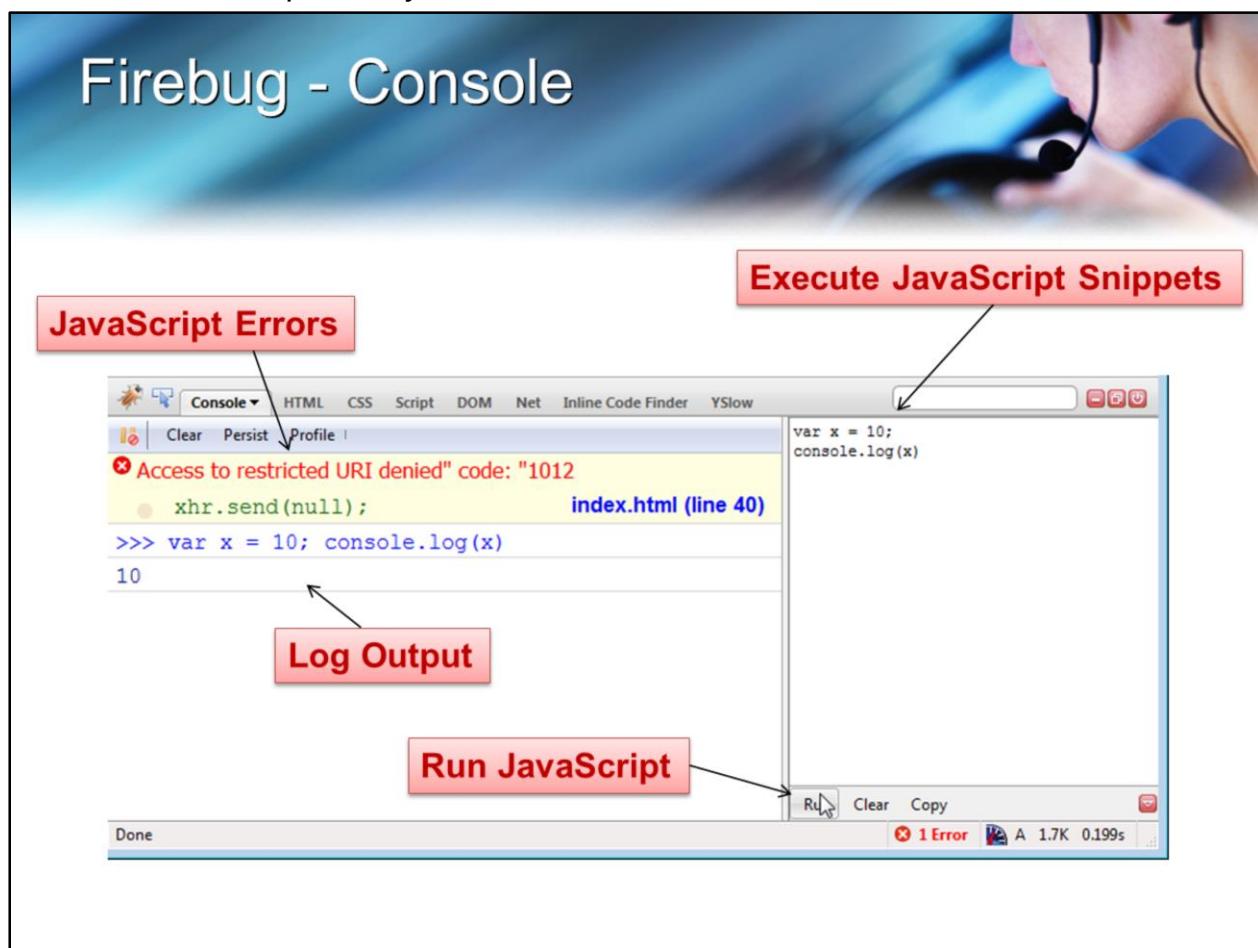
Firebug

- Firefox plug-in that provides
 - A Debugger
 - DOM Inspection Features
 - Box Model & CSS Property Info
 - Event Notifications
 - Profiling Information
 - XHR Monitoring
- F12 to launch, or
Tools > Web Developer >
Firebug > Open Firebug...

Install from
<http://getfirebug.com>

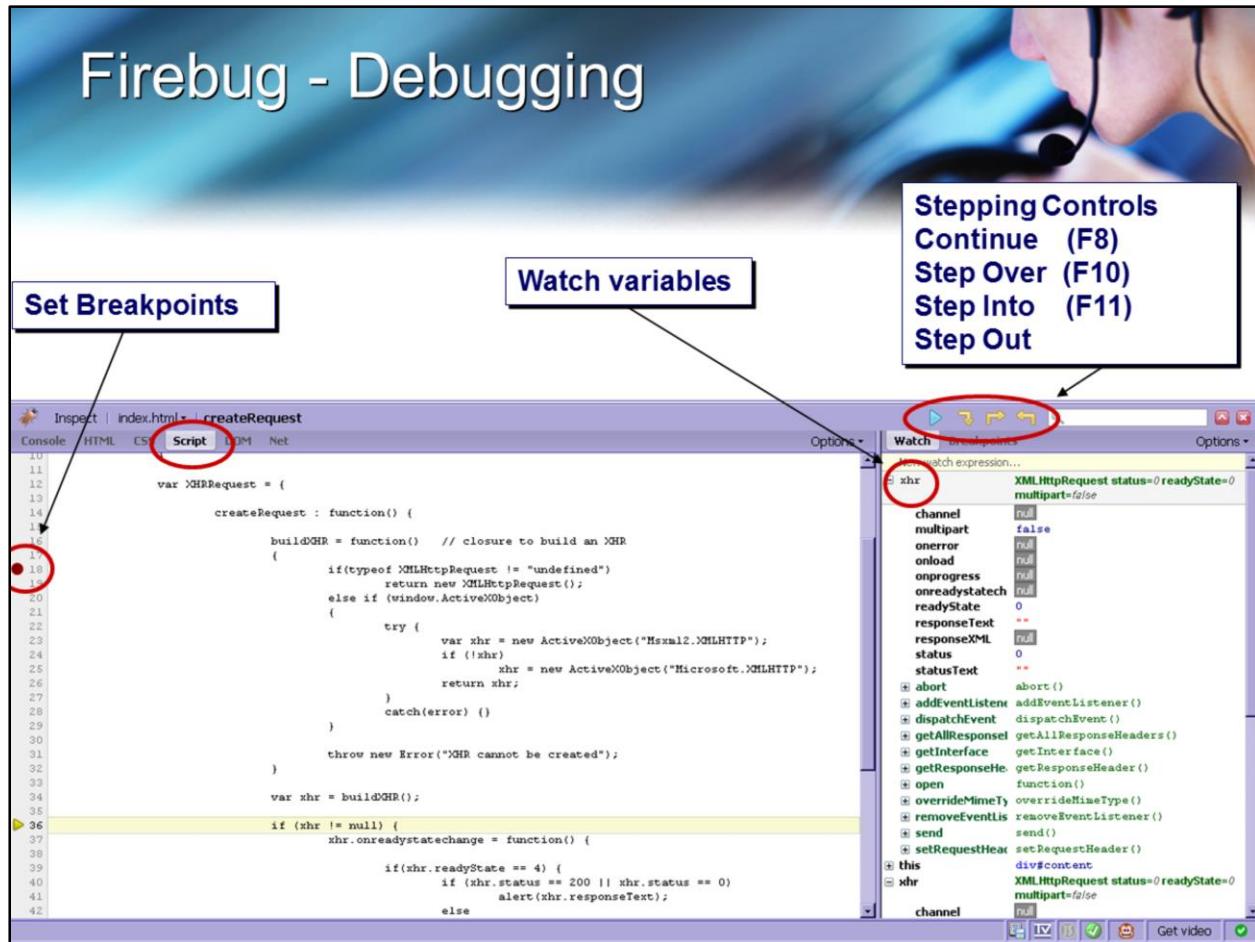


Test to see if Firebug is installed by pressing F12.



The Firebug console is useful for viewing errors, warnings, and log output. It can also be useful to test JavaScript Snippets.

Firebug - Debugging



Firebug provides a JavaScript debugger that allows developers to step through code line by line or run to a breakpoint and view variables.

To use this debugger, first load up a page in your browser and turn on Firebug. Next, in Firebug, switch to the Script tab. Directly beneath this tab is a drop-down option that allows you to select the JavaScript file to edit. Select the proper JavaScript code and locate the line to place a breakpoint on. Set a breakpoint in the gutter of the code as shown in the slide. Run the code up to this breakpoint by either refreshing the page or causing an event to hit it. Check your variables on the right pane when the breakpoint is hit.

Firebug - DOM Inspection

Allows for viewing DOM nodes and their properties

<div id="myDiv">...</div>

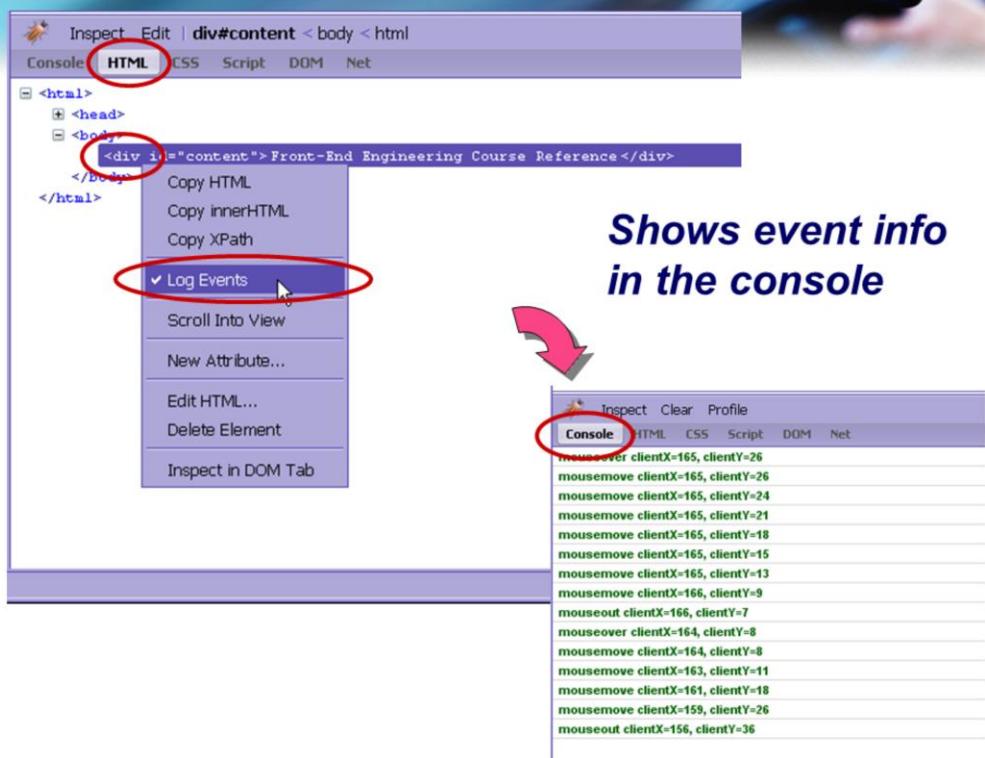
Select and then
mouse over DOM
Elements

View
Properties



The DOM tab can show DOM information about any node in the HTML tree. Because DOM information can be acquired from the HTML tab, this one isn't commonly used.

Firebug - Event Information



Events related to a single DOM node can be monitored in the Firebug console by first selecting on the HTML Tab, locating the desired DOM node in the HTML tree, right-clicking on the node, and choosing Log Events. Then switch to the console and watch for events to occur.

Firebug - Net

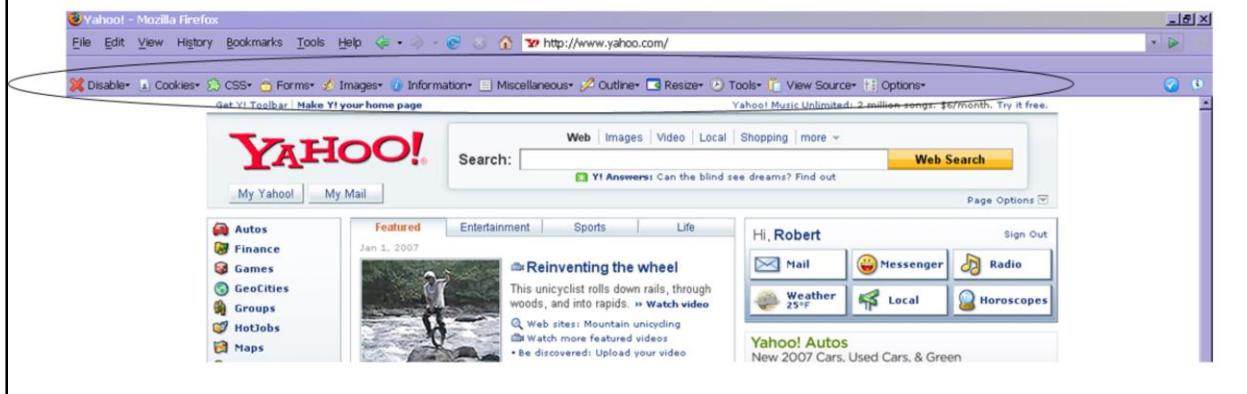
Shows data transfer time for various resources including Ajax requests

Request	Source	Size	Duration
my.yahoo.com	my.yahoo.com	34 KB	16ms
ym_a_63.css	us.js2.yimg.com	4 KB	15ms
ult_ylc_2.js	my.yahoo.com	3 KB	63ms
yad_20060816.js	ads.yimg.com	661 b	375ms
http://us.my1.yimg.com/i/feed/destspot/feb2007/my_91ec70139d409794.jpg	gmetab.com	?	453ms
favicon.ico (404)	gmetab.com	?	469ms
my_416c26547477c07b.jpg	us.my1.yimg.com	3 KB	282ms
freemyfeaturecue.js	eur.li.yimg.com	29 KB	94ms
ym_bot_a_63.css	us.js2.yimg.com	643 b	140ms
20070112_80278_1_728x90_-	ads.yimg.com	31 KB	62ms
ymy_ppe_utils_1.1.3.js	ads.yimg.com	13 KB	94ms
ymy_base_68.js	us.js2.yimg.com	14 KB	141ms
bc_2.0.3.js	us.js2.yimg.com	906 b	31ms
b	us.bc.yahoo.com	43 b	235ms
b.gif	us.lrd.yahoo.com	43 b	250ms
15 requests		140 KB (10 KB from cache)	1.46s

The Net tab within firebug provides lots of valuable information related to the loading of resources. It can give how long resources take to load, when they start to load, whether or not they were cached, what the content was that was loaded, and much more.

Firefox Web Developer Toolbar

- 1** Browse to <http://addons.mozilla.org>
- 2** Under recommended addons,
find 'Web Developer' by Chris Pederick
- 3** Download and install the plug-in, restart the browser





Chrome

- ▶ Google Chrome features separate processes for each tab
 - ▶ Version releases:
 - ▶ 1 Dec 2008
 - ▶ 12 Jun 2011
 - ▶ 16 Dec 2011
 - ▶ 29 Sep 2013
 - ▶ [webkit](#) for page layout up through Chrome 27
 - ▶ Embedded [SQLite](#)
 - ▶ Mozilla's [Portable Runtime](#) (for platform independence)
 - ▶ [V8](#) JavaScript VM, commonly benchmarks fastest
 - ▶ No major security vulnerabilities

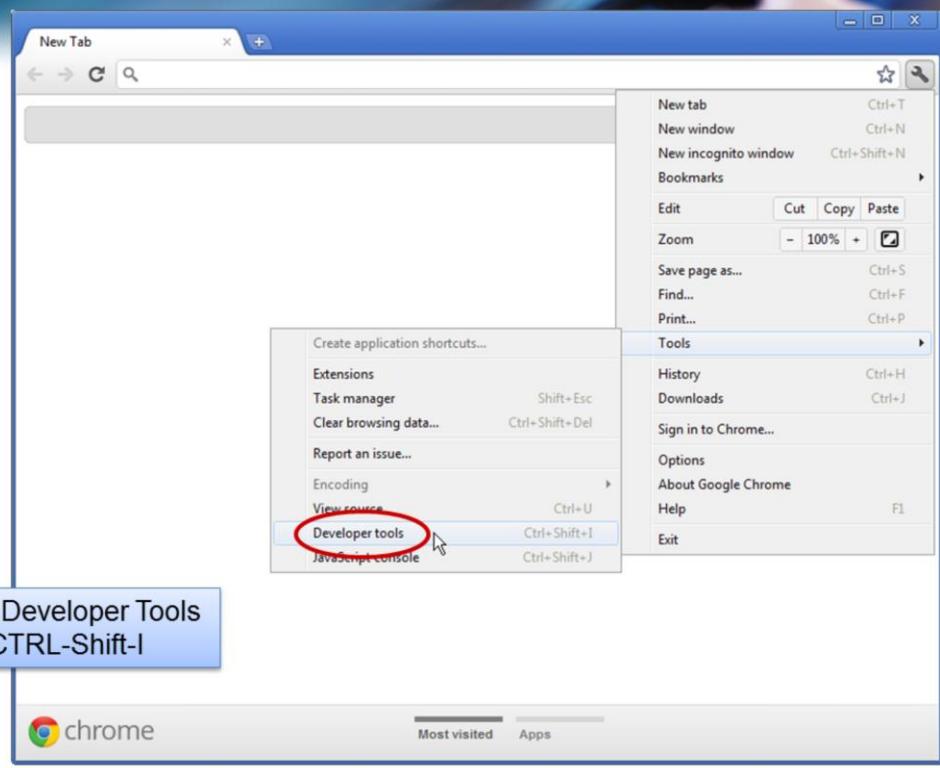
Chrome now uses the [Blink](#) layout engine (a fork of webkit)

Chromium is the open-source version and base project of Chrome. It typically features a faster update schedule, no flash integration, and a lighter (less restrictive) license.

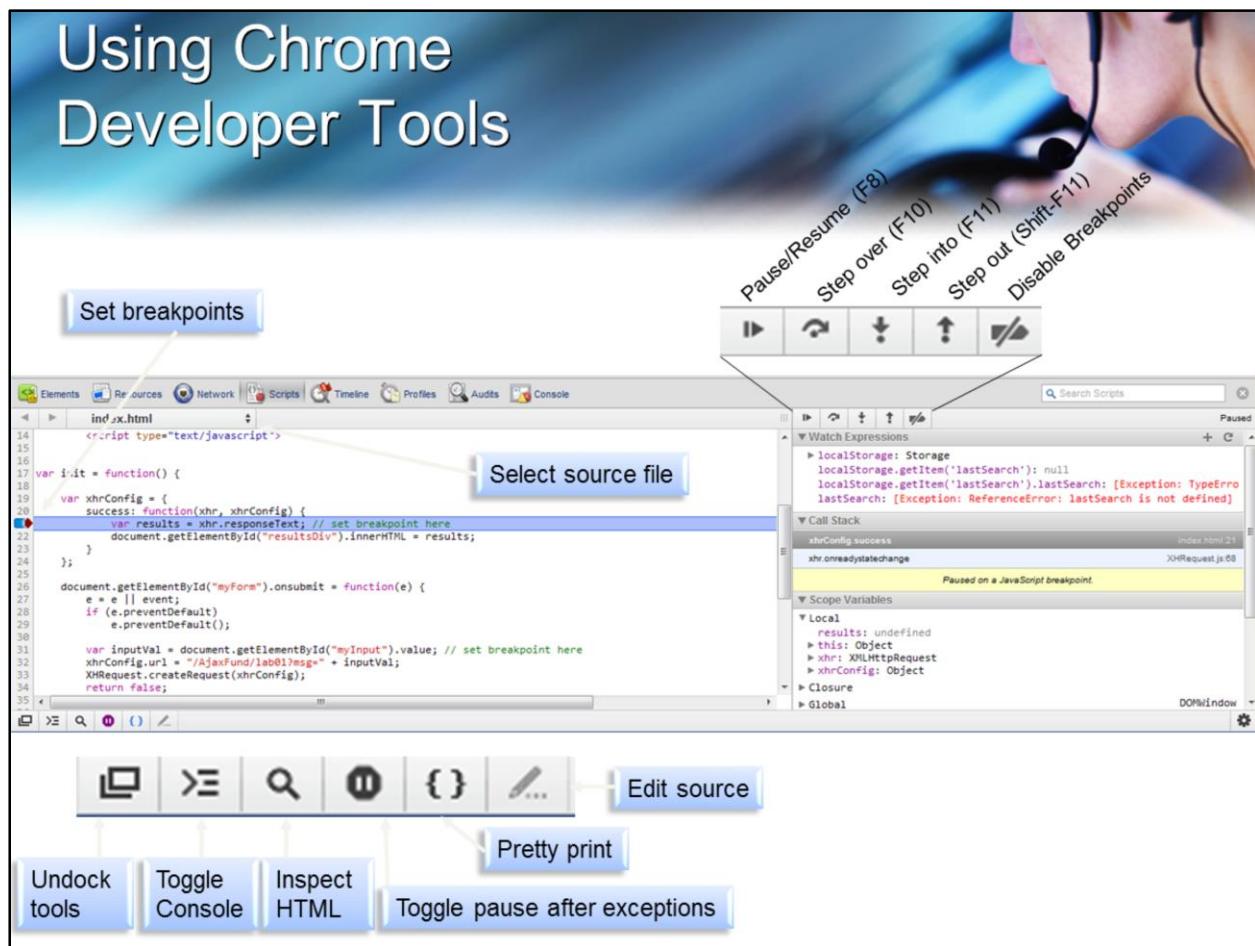
Chrome: www.google.com/chrome

Chromium: www.google.com/chromium

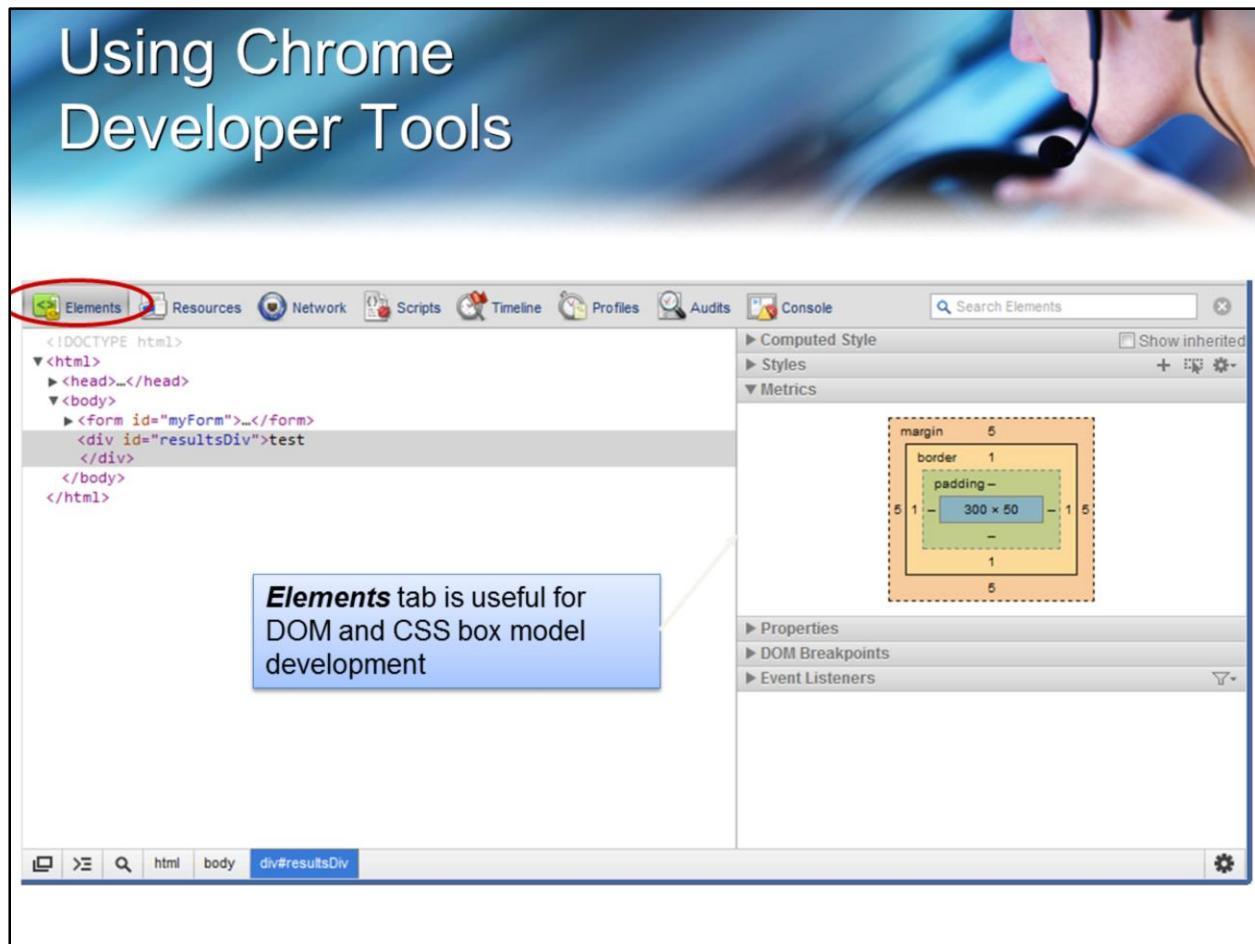
Chrome Developer Tools



Using Chrome Developer Tools



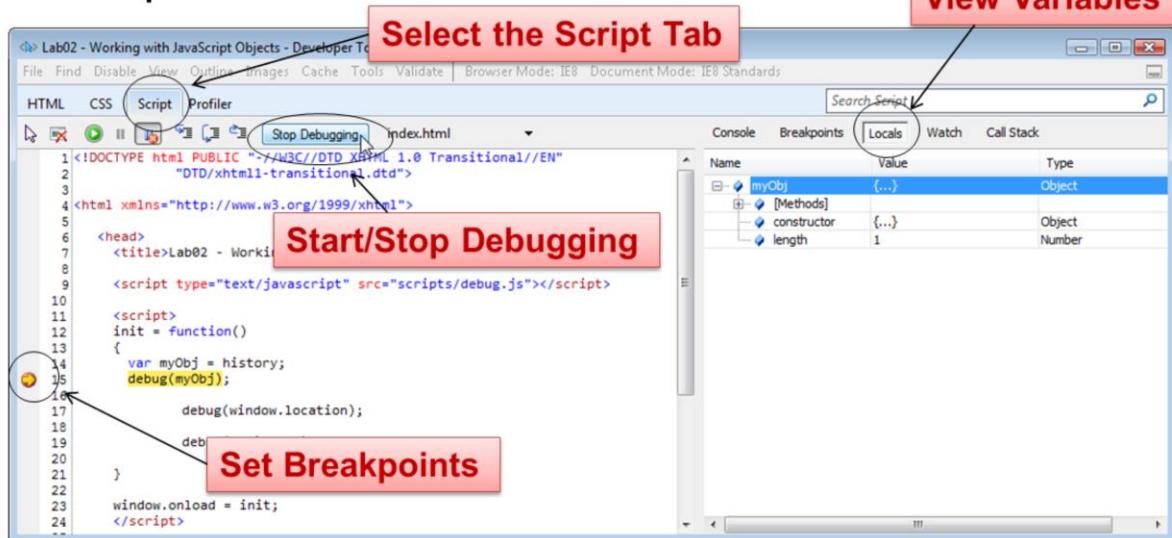
Using Chrome Developer Tools



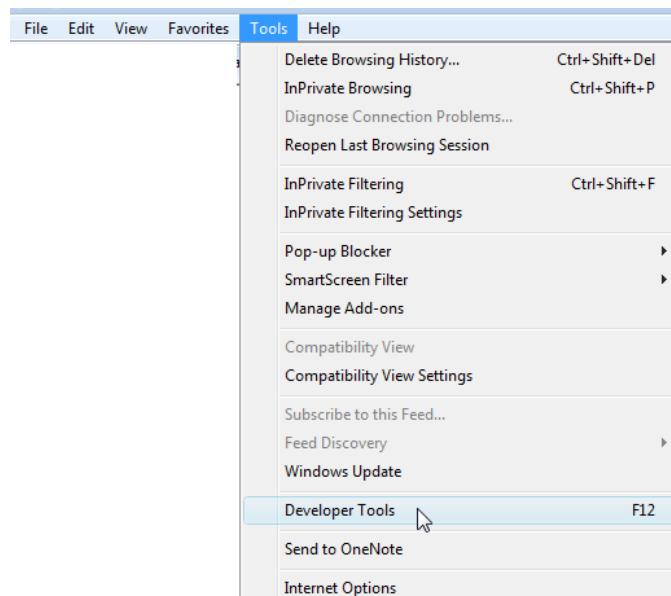
Elements tab is useful for
DOM and CSS box model
development

IE 9 Developer Tools

- IE 8+ provides a built-in debugger and DOM inspector



To open the developer tools, perform the following:



IE 11 Developer Tools

- Use IE11 has improved its developer tools
 - Similar to Firebug in many ways



IE11 now provides developer tools on par with Firebug/Chrome Dev Tools. Selecting Browser/Document Modes can be done on the Emulation tab. (Bottom icon on the left).

Lab 01a: Testing Your Environment

- Test the following sites in an HTML5 compliant browser:

<http://sitepoint-examples.s3.amazonaws.com/responsive/index.html>

- Examine the HTML source
- Compare with versions in IE (6 – 9), FF, Webkit

Indian Chickpea and Pumpkin Soup



Winter squash soups are a healthy and economical way to comfort and nourish the body and soul during the long cold months ahead. Chickpeas add depth and protein to this hearty and delicious autumn pumpkin soup, enhanced with a zesty Indian seed and spice tempering that will warm and delight your guests. Serve with a whole grain for a complete and wholesome vegetarian meal.

Soup:

- ½ cup dried chickpeas
- 4 cups water
- 1 cinnamon stick
- 1 tablespoon olive oil
- 4 dried whole red chilies
- 1 tomato, chopped
- 1 cup cooked pumpkin
- ½ teaspoon sea salt, or to taste

Tempering:

- 1 tablespoon olive oil
- 1 teaspoon cumin seed
- 1 teaspoon black mustard seeds
- 1 teaspoon caraway seeds
- 1 teaspoon ground coriander
- ½ teaspoon fenugreek seeds
- ½ teaspoon fennel seeds

Garnish:

- 1 cup fresh cilantro leaves

Related Recipes

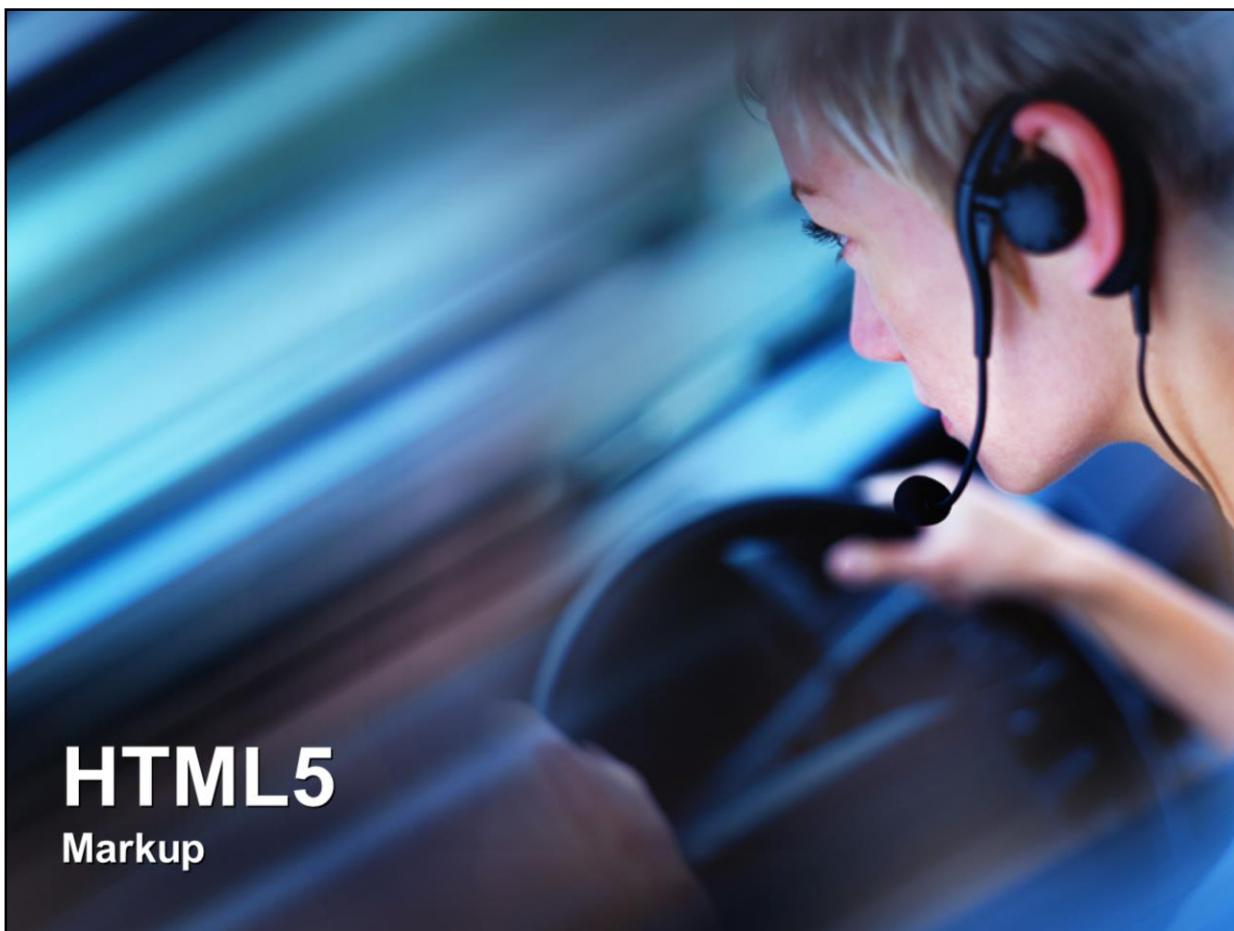
- If you liked this recipe you may also enjoy:
- [Four Our Pumpkin Soup](#)
 - [Spicy Peanut Pumpkin Soup](#)
 - [Indian Style Cream of Cauliflower Soup](#)

This page listed here implements HTML5. Examine the markup, CSS, and JavaScript used by viewing the page's source in an editor or browser.

Lab 01b: Debugging in the Browser

- In this exercise you will **debug** an Ajax interaction
 - Test the debugger using a [Chrome](#) browser

Locate the instructions for this exercise in the Student Exercises Workbook in the back of the manual



HTML5
Markup

Overview

Major Page Changes (doctype, root, head, etc.)

New Sectional Elements

Other New Elements/Attributes

Addendum: Ajax and jQuery

New Form Capabilities

Embedded Content (<audio>, <video>, <canvas>)

Major Page Changes

- New DOCTYPE
- Changes to <meta>, charset etc.
- Deprecations
- Default document types
- Validation

The New DOCTYPE

- HTML5 supports a new DOCTYPE

<!doctype html>

This declaration
is case insensitive
for HTML:

<!DOCTYPE HTML>

For XHTML, (and the preferred version) use:

<!DOCTYPE html>

Compare to HTML 4.01 Strict:

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">**

While the `<!doctype html>` declaration is case insensitive for HTML, for XHTML, you must use the all-caps version (`<!DOCTYPE html>`). For this reason, and also because some validating text editors do not properly recognize the all-lowercase version, you should use `<!DOCTYPE html>`.

Can I Use the New DOCTYPE?

- Most likely
 - If currently using a strict DOCTYPE, you may move to the new DOCTYPE without suffering visual effects
- If you use deprecated HTML 4.01 tags, your page will not validate
 - Do you care?

Triggers standards mode in all modern browsers (even IE6)

This doctype is safe to serve to all user-agents as it triggers standards mode in all browsers back to Internet Explorer 6. Additionally, using the HTML5 doctype triggers better error recovery in newer browsers and allows the use of embedded SVG.

Deprecated Tags

- HTML5 has deprecated some tags from HTML 4.01*

Tag	Replacement
applet	embed or object
acronym	abbr
dir	ul
frame, frameset, noframes	iframe
basefont, big, blink, center, font, marquee, multicol, nobr, spacer, strike	Use CSS as appropriate
tt	kbd, var, code, or samp
u	em, b, mark

Deprecated elements relate to document validity. User agents will still be capable of rendering *frames*, for example, even if using an HTML5 DOCTYPE

Despite the fact that HTML5 has declared these tags to be deprecated, using the HTML5 doctype will not cause a browser to ignore them or crash when they are encountered. The doctype will merely indicate to a validating tool that these tags should no longer be used and the tool will report this to you. If validation is critical to you and you are still using these tags, then you may not be able to use the HTML5 doctype.

<meta>

- <meta> tag syntax for specifying charset has been simplified

HTML 4.01

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

HTML5

```
<meta charset="utf-8">
```

It should be noted that in general, usually the server should be responsible for establishing the character set via an HTTP header. The charset attribute within <meta> should be limited to circumstances where the developer is unable to change the default server setting and doesn't have access to the server's configuration.

Using Default Types

- Default types are now assumed

`<link rel="stylesheet" href="main.css">` assumes type="text/css"

`<script> ... </script>` assumes type="text/javascript"

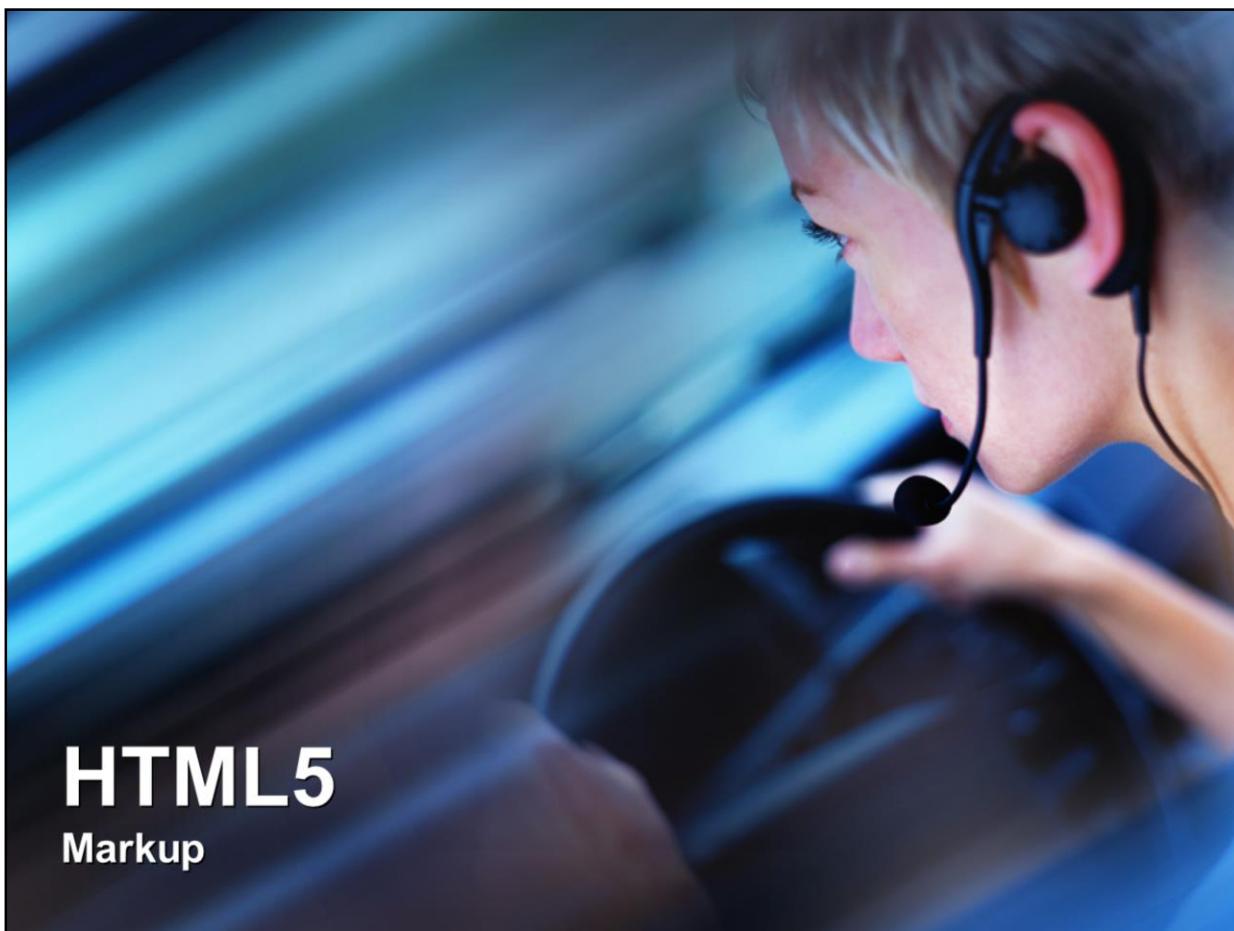
While you may certainly override the default values, browsers will be able to accept default values for the type used in `<link>` and `<script>` tags.

This behaviour is already supported cross-browser and is only formalized by HTML5.

HTML5 Validators

- The following HTML5 validators exist
 - <http://html5.validator.nu/>
 - <http://validator.w3.org/>

Validating HTML5 this early might not provide the most accurate or up to date information simply because the standard has fully stabilized yet.



HTML5

Markup

Overview

Major Page Changes (doctype, root, head, etc.)

New Sectional Elements

Other New Elements/Attributes

Addendum: Ajax and jQuery

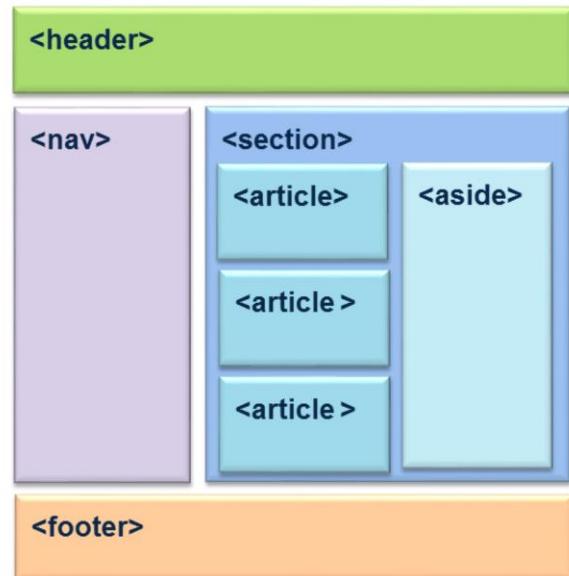
New Form Capabilities

Embedded Content (<audio>, <video>, <canvas>)

HTML5 Brings Better Semantics

- HTML5 provides new structural block-level elements:

- **<header>**
- **<section>**
- **<footer>**
- **<article>**
- **<nav>**
- **<aside>**



The following provides a high-level overview of the uses of some of the structural HTML5 block elements. More detailed explanations and use cases follow.

Issues with Pre-HTML5 Browsers

- Internet Explorer (even IE8) does not natively understand HTML5 elements
- Other browsers understand HTML5 elements, but may not render them properly (as block elements)
 - Some browsers (such as early FF3 browsers) require CSS `display:block` property to be set

```
<style>
    article,aside,details,figcaption,figure,footer,
    header,hgroup,menu,nav,section {
        display: block;
    }
</style>
```

This issue at hand becomes, how do we as developers maintain a leading-edge perspective by utilizing the "cool" new features of HTML5, while at the same time providing solutions that will work even in browsers that pre-date the HTML5 specification?

An HTML Shim (Shiv) for IE

- IE8 and earlier can be "tricked" into understanding the new markup (even IE6)

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <style>
      time { font-style: italic; color: red; }
    </style>
    <script>document.createElement('time');</script>
  </head>
  <body>
    Born:<time datetime="1950-03-15">
      March 15, 1950</time>
    </body>
  </html>
```

This solution has one drawback:
it requires JavaScript!

Several solutions have emerged to try to solve this dilemma of making older browsers recognize the HTML5 tags. One approach has been to utilize what has been dubbed a "shiv". The shiv essentially uses DOM to create an element within the browser via JavaScript. This element will essentially be a new element, however, it turns out that IE can then recognize the element and even utilize it within CSS source for styling.

There are a few limitations to the JavaScript shiv approach one of which is that there are some issues with DOM manipulation of these objects. Another issue, and an even bigger one, is that in order to get a browser to recognize the new tags, JavaScript will be required.

Can You Require JavaScript?

- In order for IE 6-8 to recognize HTML5, JavaScript must be turned on
 - *For internal apps, controlled environments, this can be required*
 - Can you require your users to turn on JavaScript?
 - If your answer is no, then you must come up with a solution that works without JavaScript
 - One exists, though it is not at all ideal...

On the slide that follows, you must keep in perspective that this is only an alternative that attempts to present a non-JavaScript approach toward using HTML5 in browsers that do not support it. This means IE6 through IE8. Also keep in mind that as of this writing, IE8 is currently the world's most popular browser.

<header>

- Headers are specialized forms of sections
 - Indicate introductory content in a section
 - Used for universal page header, leading story content (title, author, etc.), TOC, by-lines, etc.
 - Formalizes: <div class="hd"> pattern
 - Should not exist within another <header> or within <footer>



Because a <header> should not contain a header or footer element, a complex page header might require a section instead.

<header> Example

```
<div id="header" class="hd">
  <h3>BMW USA</h3>
  <h4 class="tagline">The ultimate Driving
    Machine</h4>
</div>
```



```
<header>
  <h3>BMW USA</h3>
  <h4 class="tagline">The ultimate Driving
    Machine</h4>
</header>
```

The HTML5 `<header>` element replaces the common pattern of using a `<div>` as a header element within an HTML4 page (this is a perfect example of "paving the cowpaths").

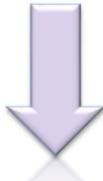
<footer>

- Footers are also specialized forms of sections
 - Indicate concluding content in a section
 - May be used to contain copyright info, author info, links to additional references, etc.
 - Formalizes: <div id="footer"> <div class="ft"> patterns



<footer> Example

```
<div class="ft">
  <h4>More links</h4>
  <ul><li>...</li></ul>
</div>
```



```
<footer>
  <h1>More links</h1>
  <ul><li>...</li></ul>
</footer>
```

This example shows the h4 changed to h1, which is correct for HTML5 syntax, but not in current practice
(more shortly....)
(more shortly....)

Notice, once again, the replacement of the <div> element previously used as a footer. Now, with the better semantic choices, the footer element can be used instead.

Note: The above example (and the one shown previously with the <header> element) assume the client uses JavaScript and an IE Shiv can be employed (or IE is not a concern at all). If JavaScript cannot be ensured in the client, but HTML5 is desired, then the nested <div> technique would have to be used. It would look like the following:

```
<footer>
  <div class="footer ft">
    <h1>More links</h1>
    <ul><li>...</li></ul>
  </div>
</footer>
```

<section>

- A *generic section* of a document,
 - Should contain a heading (h1 - h6) and often a <header> and/or <footer>
- Not a generic container...use <div>
 - A <div> is appropriate for holding generic content
 - Sections are relevant if they would be listed in a document's outline
 - Think chapters in a book, verses in a poem
- <section> uses the heading elements (e.g., h1-h6) relatively *(more on this later)*

A <section> is not a <div>. <div> elements are used to divide up a page, or a component. <section> elements should be used to create sections within a document in a textual sense. Like chapters in a book, or verses in a poem.

Sections "reset" the value of h1 internally, so that two different sections can define their own <h1> elements. This presents a problem right now since most browsers don't handle the "relative header tags" very well yet.

<section> Examples

```
<section>
  <header>...</header>
  <article>...</article>
  ...
  <article>... </article>
  <footer>... </footer>
</section>
```

```
<article>
  <header>...</header>
  <section>...</section>
  <footer>... </footer>
</article>
```

Sections may contain additional sub-headers, sub-footers, and sub-sections.

<article>

- <*article*> elements are specialized sections
 - Typically articles will contain syndicated content such as feeds, blogs, news stories, forums, reviews, commentaries, etc.
- A <*section*> may always be used in place of an article (or header or footer), but it is less semantic
- <*article*> elements may contain <*section*> elements

Sections may contain articles just like an article may be broken into sections. Earlier rules about ensuring a <div> (or <p> or etc.) directly enclose the elements still applies to <articles> as well.

<article> Example

```
<div class="entry">
  <h3>
    <a href="#" rel="bookmark" title="link to post">
      Travel day
    </a>
  </h3>
  <p class="post-date">October 22, 2009</p>
  <div>article content...</div>
</div>
```

This might represent a "classic" HTML4-style markup for a blog entry.

<article> Example (as HTML5)

```
<article class="entry">
  <header>
    <h1>
      <a href="#" rel="bookmark" title="link to post">
        Travel day
      </a>
    </h1>
    <time class="post-date">
      October 22, 2009
    </time>
  </header>
  <section>
    ...Article Content...
  </section>
</article>
```

The HTML5 version of the example from the previous slide obviously contains more markup. The growth in the markup size is expected for now.

Also, notice the change in structure from an `<h3>` to an `<h1>`. This is a shift in the way HTML5 uses the h1-h6 header tags and it differs from HTML 4 in that the h1-h6 header tags under HTML 4 were relative to the entire document.

<article> Example (with HTML 4-style headings)

```
<article class="entry">
  <header>
    <h3>
      <a href="#" rel="bookmark" title="link to post">
        Travel day
      </a>
    </h3>
    <time class="post-date">
      October 22, 2009
    </time>
  </header>
  <section>
    ...Article Content...
  </section>
</article>
```

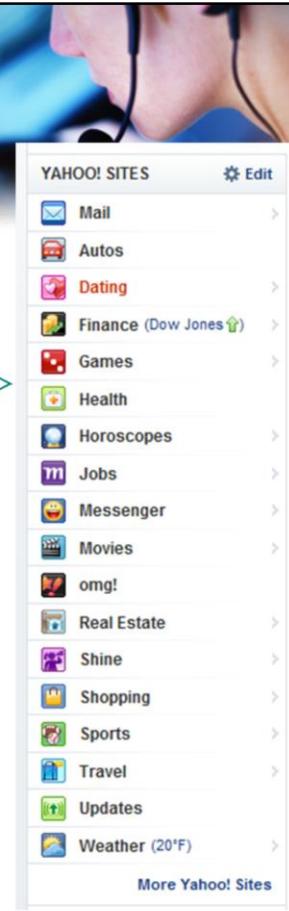
Under HTML4, only one
<h1> should have
existed on a page

This difference between this and the previous slide is that we are maintaining the h1-h6 structure throughout the document. The option not to use h4 in place of the p was that the h4 might have been used differently elsewhere on the page. Not using an h4 in place of the p requires us to remove the hgroup tag.

Example from yahoo.com

```
<div class="y y-1n-1" id="u_2588582-y">  
  <div class="hd y-1n-1 y-fp-grad clearfix">...</div>  
  <div class="bd">...</div>  
  <div class="ft y-fp-grad-controls clearfix">...</div>  
</div>
```

Actual source for their nav bar

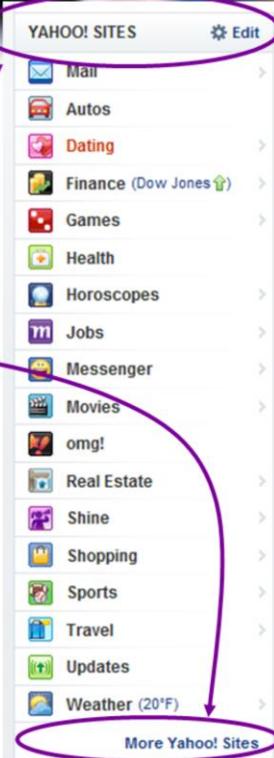


This is the actual HTML for the sidebar as it currently exists on yahoo.com.

yahoo.com using HTML5 Markup

```
<nav class="y y-ln-1 nav" id="u_2588582-y">
  <header class="hd y-ln-1 y-fp-pg-grad clearfix">
    ...
  </header>
  <section class="bd">...</section>
  <footer class="y-fp-pg-controls clearfix">
    ...
  </footer>
</nav>
```

Possible HTML5 version



In the example, the common Yahoo "hd", "bd", "ft" microformat has been replaced with `<header>`, `<section>`, and `<footer>` elements.

<aside>

- <aside> is another specialized form of section
 - Provides tangentially related content
- May be used for
 - advertisements, pull quotes, or sidebars
 - groups of related links (nav)
 - background / additional helpful information
- Should appear as a child inside the element it supports
 - If supporting an article, it should appear within the article

<aside> Examples

```
<article>
  <header>...</header>
  <section>...</section>
  <footer>...</footer>
  <aside>...</aside>
</article>
```

```
<article>
  <h1>Blog 2</h1>
  <p>I discovered a new planet while...</p>
  <aside>
    <p>The number of potential planets
       that can support life are...</p>
  </aside>
</article>
```



Usage Test
Can the <aside> be removed without losing the meaning of the main section?

<nav>

- A specialized section that contains lists (groups) of links to other pages (or to locations within the page)
- <nav> supplements the classic <ul id="nav"> pattern
- <nav> has no relation to page position



The <nav> element doesn't imply any presentation or document location. It can appear anywhere in the document and it can "look" like anything. Semantically the nav element should contain a list of links, that's all.

<nav> Example

```
<div>
  <ul id="nav">
    <li><a href="#">home</a></li>
    <li><a href="#">blog</a></li>
    <li><a href="#">gallery</a></li>
    <li><a href="#">about</a></li>
  </ul>
</div>

<nav>
  <ul id="nav">
    <li><a href="#">home</a></li>
    <li><a href="#">blog</a></li>
    <li><a href="#">gallery</a></li>
    <li><a href="#">about</a></li>
  </ul>
</nav>
```



Typically the will be contained within the <nav>. A <nav> is simply a container (technically just a specialized section) holding the list of links. It doesn't replace the .

Document Outline

- HTML5 allows for the "resetting" of the heading (h1-h6) elements in each section

```
<article>
  <heading>
    <h1>My Title</h1>
    <h2>My Subtitle</h2>
  </heading>
  <section>
    <h1>Section Title</h1>
    <p>Article content</p>
    <section>
      <h1>Sub-content title</h1>
      <p>Article sub-content</p>
    </section>
  </section>
</article>
```

In HTML5, h1 is relative to its containing section, not the document

An option exists to continue using document-relative, h1-h6 tags, therefore no changes are required to current use of heading tag

Only articles and sections participate in the document outlining (i.e. have the effect of resetting the heading elements).

Document Outline Problems

- The outline concept of headings now means that no h1-h6 tags should be skipped
 - You should not begin an article with an h5 or h6 element, for example, but rather h1 then h2
- This will eventually improve semantics, but currently many user agents and search engines will not interpret the heading tags when used this way
 - So, for now, **continue to use the heading tags as you have before...HTML 4.01-style**

Check out the following reference:

The Truth about multiple <h1> tags.

<http://webdesign.tutsplus.com/articles/the-truth-about-multiple-h1-tags-in-the-html5-era-webdesign-16824>

Should You Use HTML5 Document Outlining?

- There are a few scenarios where the HTML5 document outline algorithm will have an effect:
 - Browsers (user-agents)
 - Screen Readers
 - Search Engines
- Chrome / Firefox now support HTML5 document outlining
- Screen readers prior to 2012 do not support it
- Search engines are beginning to allow for it
(see footnotes on Google comments)

From Google Webmaster Central:

Our general strategy is to wait to see how content is marked up on the web in practice and to adapt to that. If we find that more and more content uses HTML5 markup, that this markup can give us additional information, and that it doesn't cause problems if webmasters incorrectly use it (which is always a problem in the beginning), then over time we'll attempt to work that into our algorithms....

HTML5 is still very much a work in progress, so it's great to see bleeding-edge sites making use of the new possibilities

Document Outline – HTML 4 Style

- HTML5 permits the heading tags to be used as they always have: with respect to the entire document

```
<article>
  <header>
    <h1>My Title</h1>
    <h2>My Subtitle</h2>
  </header>
  <section>
    <h3>Section Title</h3>
    <p>Article content</p>
    <section>
      <h4>Sub-content title</h4>
      <p>Article sub-content</p>
    </section>
  </section>
</article>
```

A blue callout box with a white border and a shadow is positioned to the right of the code. Two blue arrows point from the text 'Use the heading tags as if they were *document* relative' towards the opening and closing tags of the first `h1` element in the code.

Use the heading tags as if they were *document* relative

Your rule of thumb (best practice) will be to use the heading tags exactly as you always have in the past. Generally this means with respect to the entire document.

Exercise: Convert the News App

- Yahoo! News Page
- An archived page of news.yahoo.com has been provided for you in your student files:

lab02/starter/news.html

The screenshot shows the Yahoo! News homepage. At the top, there's a navigation bar with links for Home, U.S., Business, World, Entertainment, Sports, Tech, Politics, Elections, Science, Health, and Most Popular. Below the navigation is a search bar. The main content area features several news stories with images and headlines:

- Japan's prime minister says he'll resign** (with a photo of Shinzo Abe)
- Quake triggers tsunami in Indonesia** (with a photo of a person in a disaster zone)
- Nancy Pelosi and the Iraq Report** (with a photo of Nancy Pelosi)
- Putin names surprise nominee for PM** (with a photo of Vladimir Putin)
- 60 MINUTES ON YAHOO! NEWS** (with a photo of Jerry Seinfeld)
- YOU WITNESS NEWS** (with a photo of a person holding a sign)

On the right side, there's a sidebar titled "PEOPLE OF THE WEB" featuring "People of the Web". Below that is an "ADVERTISEMENT" section for "ONLINE PROGRAMS" at "University of Phoenix".

Lab 02 – HTML5 Document Structure

- In the student files lab02/starter directory open **news.html**
 - Convert this page to use HTML5
- Use the proper doctype and sectional elements
- Only modify the HTML not the CSS





HTML5

Markup

HTML5 Markup

Major Page Changes (doctype, root, head, etc.)

New Sectional Elements

Other New Elements/Attributes

Addendum: Ajax and jQuery

New Form Capabilities

Embedded Content (<audio>, <video>, <canvas>)

Other Notable Elements / Attributes

- <figure>, <figcaption>
- <mark>
- <time>, <meter>
- <details> & <summary>
- , <i>, <small> revisited
- Custom attributes
- Global attributes

At this current time, these elements are provided for informational purposes. Due to their overall lack of support in pre-HTML5 browsers and due to the fact that their appearance/behavior is still to be established in cross-browser design, you should avoid using them in practical implementations.

<figure> & <figcaption>

- <figure> is a block element that marks up illustrations, photos, diagrams, code listing
 - <figcaption> provides a description for a figure

```
<figure>
  
  <figcaption>
    This car exploded as it rounded the
    last turn of lap 37--
    <a href="http://www.flickr.com/photos/fake">
      speedcar
    </a>
  </figcaption>
</figure>
```

Figure is a block-level element.

<figure> & <figcaption> Example

In [listing 4](#14) we see the primary core interface API declaration.

```
<figure id="14">
  <figcaption>
    Listing 4. The primary core interface API declaration.
  </figcaption>
  <pre><code>interface PrimaryCore {
    boolean verifyDataLine();
    void sendData(in sequence<byte> data);
    void initSelfDestruct();
  }</code></pre>
</figure>
```

The API is designed to use UTF-8.

<figcaption> should be a child of <figure>

Figures can encapsulate images and even code examples, as shown above. The figcaption does not have to be placed first or last within a <figure> only as a child somewhere.

<mark>

- Mark allows a text to be highlighted for use
 - Example: Results page from a search may "mark" the original search string

```
<p>I also have some <mark>kitten</mark>s who are visiting  
me these days. They're really cute. I think they like my  
garden! Maybe I should adopt a <mark>kitten</mark>.</p>
```

- Mark simply provides a slightly better semantic choice than a currently does
 - A is a generic container
 - A <mark> is designed to "mark" something for styling

A <mark> element is an inline element to be used for the purpose of "marking" some code for the purpose of styling (or manipulation).

A mark may also be used for bringing attention to a quotation.

Date and Time with <time>

- Dates and times may be marked with a **<time>** element

```
<time datetime="1950-03-15">  
    March 15, 1950  
</time>
```

datetime should be used to represent "machine-formatted", or parsable, times when the content value is not in a valid format

The specification defines the rules for datetime formatting

The HTML5 specs provide detailed information on formatting the value for the datetime element.

<meter> and <progress>

- Meter denotes a scalar measurement
 - A range should be defined (min/max, low/high)

```
<span>2.5</span>
```



```
<meter min="0" max="5" value="2.5">
  <span class="meter">2.5</span>
</meter>
```

- Use <progress> to indicate a progress bar

```
<progress max="1000" value="356" />
```

Since older browsers will not understand the value attribute or min/max values used within <meter>, these may not be used for visual purposes. They can be used to provide progressively enhanced content to browsers supporting the element, but it should degrade such that all browsers retain the core information provided by the element.

 and <i> - they're back!

- While controversial and perhaps not final, and <i> have returned to HTML5
- <i> implies a **voice inflection** or language or tone change
- implies a **stylistic offset** such as keywords in a document or any item that might require a hook for styling
- * and <i> should be used only as a last resort*

To avoid confusion, and <i> should generally be avoided in HTML5 unless absolutely necessary.

<small> and <hr>

- **<small>** used to indicate "small points" or side comments

```
<p>In an effort to dissuade investors from making a serious  
mistake by buying Falcomm Inc. shares  
<small>(full disclosure: This company has no  
vested interests in Falcomm)  
</small>, leading to speculation about a third quarter  
our recommendations are...  
</p>
```

- **<hr>** used to indicate a break in thought usually separating paragraph text elements

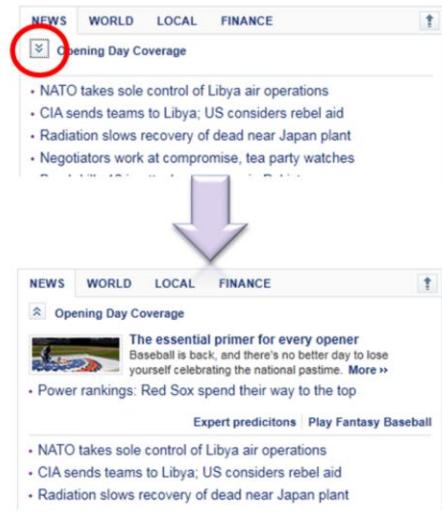
```
<p>At first it seemed we would never arrive... </p>  
<hr>  
<p>On another day, there was this incident...</p>
```

<section> and <h1> elements have implied breaks in them so an <hr> between those types of elements is unnecessary.

<details> & <summary>

- Represents a disclosure widget from which the user can obtain additional information or controls

```
<details>
  <summary>
    Opening Day Coverage
  </summary>
  <article>
    <h1>The essential...
    <p> Baseball is back...
    <ul><li>...</li></ul>
    <footer>...</footer>
  </article>
</details>
```



Global Attributes

- A number of attributes may be used on all HTML elements
 - contenteditable
 - contextmenu
 - onXXX (e.g. onclick)
 - draggable
 - hidden
 - role
 - aria-XXX
 - spellcheck="true|false"
 - Custom attributes: data-XXX

There are a number of elements that may now appear in any HTML element. Some of these are discussed on successive pages.

Custom Attributes

- It is possible to specify custom attributes on any element now

- Must be prefixed with **data-**

```
<input type="email" data-allowedDomains="yahoo google"/>
```

- This can be used in all browsers currently including pre-HTML5 browsers

- Data can be retrieved using

```
el.getAttribute(attrName)
```

This works in pre-HTML5 browsers as well.

ContentEditable

- Allows content to be directly edited by a user

```
<article contenteditable="true">
  <h2>Blog 1</h2>
  <p>Today I saw a bear...</p>
</article>
```

- Use only if a fallback, such as a textarea or equivalent, is provided for pre-HTML5 browsers

To achieve this effect in older browsers, it may be necessary to implement onclick event handling on the `<div>` element and "simulate" this behavior using JavaScript.



HTML5

Addendum: jQuery

HTML5 Markup

Major Page Changes (doctype, root, head, etc.)

New Sectional Elements

Other New Elements/Attributes

Addendum: Ajax and jQuery

New Form Capabilities

Embedded Content (<audio>, <video>, <canvas>)

Client-Side JavaScript Libraries

- Promote unobtrusive JS development



Others:
HTML5 Shim
Modernizr
EnhanceJS
Passive Localization



jquery.com

prototypejs.org

sencha.com

developer.yahoo.com/yui

dojotoolkit.com

www.wakanda.org

labs.adobe.com/technologies/spry/

qooxdoo.org

mootools.net

jqueryui.com

A Bit of jQuery to Help Us...

- jQuery is a fast, pluggable, widely used, clean API
- To incorporate jQuery, simply provide a single reference to its JS file
 - A hosted CDN version may also be used

```
<script type="text/javascript" src=
  "https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.js">
</script>
<script type="text/javascript">
  window.jQuery || document.write('<script type="text/javascript"
src="path/to/your/jquery"></script>');
</script>
```

This works because <script> elements are synchronous before the `window.onload` fires

To incorporate many of the leading libraries, you may use a CDN such as Google's JS API. To use it for your favorite library, visit the following URL and follow the instructions for providing the proper hosted URL on your pages:

<https://developers.google.com/speed/libraries/devguide>

After the DOM Builds...

- When `<script>` elements are encountered by the browser, they can begin executing
 - This may occur before the DOM has fully completed building
- To remedy this issue:
 1. Place `<script>` elements at the bottom of the body, or
 2. Wait until the `window.onload` has fired
 3. Wait until the document is ready (jQuery)

```
<html>
  <head></head>
  <body>
    ...
    <script></script>
  </body>
</html>
```



`examples/jquery/02_executing_scripts.html`

jQuery: Document Ready

- In order to ensure the DOM has been constructed, before executing JavaScript, invoke either:
 - **jQuery's ready() method**

```
$ (document) .ready(function() {  
    // work safely with DOM  
});
```

- **Use the \$() function**

```
$(function() {  
    // work safely with DOM  
});
```

In order to safely reference the DOM nodes created from your HTML document, you must ensure the browser has "read" and loaded them all. There are several ways to accomplish this. One is to wait until window.onload fires, which is after everything loads, includes stylesheets and images. Another approach is to use jQuery's ready() function. Use this in a <script> tag within the head and it will not fire until the DOM has been fully constructed. A last approach might be to place JavaScript code at the bottom of the page (within the body tag).

Wrapped Nodes in jQuery

- When working with DOM nodes, jQuery wraps them providing new (additional) functionality

myDiv is wrapped in a jQuery object

```
$("#myDiv").click(function(e) {
```

```
    console.log(this.innerHTML);
```

```
    console.log(e.target.innerHTML);
```

```
});
```

Both of these display "Welcome!"

```
<div id="myDiv">Welcome!</div>
```

examples/jquery/03_jquery_wrapper.html

The jQuery Object

- The jQuery object is "overloaded"

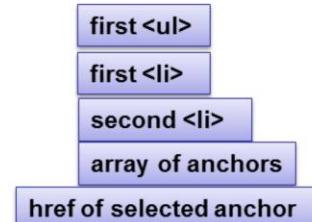
<code>jQuery(selector, startingPoint)</code>	<code>jQuery("li", "#topMenu")</code>
<code>jQuery(htmlElement)</code>	<code>jQuery(document.getElementById('myDiv'))</code>
<code>jQuery(this)</code>	wraps a DOM node as above example
<code>jQuery(htmlArray)</code>	wraps an array of HTML nodes
<code>jQuery(jQueryObj)</code>	<code>jQuery(\$('myDiv'))</code> clones it
<code>jQuery(document)</code>	wraps the document object
<code>jQuery()</code>	an empty set, wraps no nodes
<code>jQuery('div', xhr.responseXML)</code>	finds div's in an Ajax XML response

The \$ and jQuery objects are the same.

jQuery DOM Traversal Methods

- Use the following methods to perform DOM traversal in jQuery (*using the HTML shown on the next slide*)

```
console.log($('#topmenu')  
           .find('ul:first')  
           .children(0)  
           .next()  
           .children()  
           .attr('href'));
```



- *Other methods include:*
 - `closest(selector)` matches first ancestor up the tree
 - `parent(selector)` matches parent element
 - `siblings(selector)` matches all siblings based on selector

In general, while DOM traversal is discouraged and should generally be avoided, jQuery makes this task easier by normalizing differences (such as how whitespace is handled) between browsers.

Selecting Elements in jQuery

- `\$()` can select nodes using CSS3 selector syntax:

```
$ (document).ready(function() {
    $("li ul").hide();
    $("li ul:first").show();
    $("body > ul > li > a").click(function() {
        $("ul > li > ul:visible").slideUp("slow");
        $(this).next().slideDown("slow");
        return false;
    });
});
```

The screenshot shows a user interface element where a menu item has been expanded. The expanded menu contains three items: "jQuery", "jQuery UI", and "Prototype". These items are part of a larger navigation structure.

```
<ul id="topmenu">
    <li><a href="#">Libraries 1</a>
        <ul>
            <li><a href="http://jquery.com">jQuery</a></li>
            <li><a href="http://ui.jquery.com">jQuery UI</a></li>
            <li><a href="http://prototypejs.org">Prototype</a></li>
        </ul>
    </li>
</ul>
```

examples/jquery/04_jquery_traversal.html

jQuery Events

- There are methods for most browser events
 - Syntax: `someElement.evType(handlerFunction)`
- ```
$('.lastInput').mouseover(function() { ... });
```

|            |            |           |                |                    |
|------------|------------|-----------|----------------|--------------------|
| click()    | keyup()    | ready()   | ready()        | mouseup/down()     |
| dblclick() | keydown()  | unbind()  | change()       | mouseenter/leave() |
| focus()    | keypress() | toggle()  | select()       | mouseover()        |
| blur()     | scroll()   | trigger() | submit()       | mousemove()        |
| resize()   | live()     | on()      | hover()        | load/unload()      |
| one()      | die()      | off()     | jQuery.proxy() | bind/unbind()      |

*bind()* is a generic event handler that is used in the form of `someElement.bind('eventType', function(evt){});`

`unbind('eventType')` removes this binding

`one()` can be used to attach a function that runs only once on an element and is then removed

`live()` and `die()` are illustrated on the next slide

`toggle()` can be used to define two event handlers and toggles between them when a particular element's event occurs

`ready()` can be triggered on any DOM element, not just the document. It fires when the element becomes first available after page construction

`hover()` binds between two functions that fire alternately as the hover occurs or ends

`jQuery.proxy()` is used to takes a function

# The jQuery Event Object

- The following illustrate methods/properties from the jQuery library related to event handling:

event.preventDefault()  
event.stopPropagation()  
event.target  
event.delegateTarget  
event.pageX, event.pageY  
event.type  
event.result  
event.which  
event.timeStamp  
event.data

Prevent default action from occurring  
Prevent event bubbling  
The event source where bubbling began  
The element where handling was placed  
The mouse coords., works in all browsers  
The event type (e.g. 'click', 'mouseover')  
Value returned by prev. event handler in a chain  
Determines which mouse or key button was pressed  
Time since handler established, time since page load  
Object that may be passed to handler. Defined when handler is first bound to element

[examples/jquery/07\\_jquery\\_event\\_result\\_and\\_data.html](examples/jquery/07_jquery_event_result_and_data.html)

With stop propagation, this can affect the live() method as it uses propagation on the document and assumes events will eventually arrive at the document level.

event.which normalizes between browsers which key or mouse button was pressed.

event.timeStamp reports the time that has elapsed since the event handler was first established. This is usually since the page loaded.

event.data may pass in an optional data structure each time a handler is invoked. See the above referenced example.

# Appending Text & Attributes

- To set the text or HTML of an element:
  - Use `.html("content")` if it contains HTML
  - Use `.text("content")` to ignore (escape) HTML:

```
$("#myDiv").text("This text is now in the div.");
$("#myDiv").html("This is html within the div.");
```

- Use `.text()` or `.html()` without params to retrieve the contents of an element

- To create an element and text:

```
$('#myDiv').append('<div>Nested element</div>');
```

Other useful manipulation methods include `appendTo()`, `prepend()`, `prependTo()`, `after()`, `before()`, `insertAfter()`, `insertBefore()`, `wrap()`, `wrapInner()`, `wrapAll()`

# Adding & Removing Classes

- Classes are common in front end solutions
  - jQuery provides methods to work with classes if the browser doesn't support it
  - Use `addClass()`, `hasClass()`, `removeClass()`, and `toggleClass()` to manipulate class behavior

`removeClass()` without parameters removes all classes

*examples/jquery/08\_jquery\_classes.html*

# Setting CSS Styles

- To set or get an individual CSS property, use the `css()` method:

```
$("#topmenu").css('backgroundColor', 'red');
```

- This can be applied to many elements at once:

```
$("#topmenu li").css('backgroundColor', 'red');
```

Generally, setting styles can break down a nicely established layered architecture. In general, it is best to use classes.

# Removing & Replacing Nodes

- Use remove() to remove the selected node(s):

```
$('#myDiv').remove();
```

- Use replaceAll() and replaceWith() methods

```
$('<div>Some Content</div>').replaceAll('.myClass');
```

Creates the div and replaces every element  
with the *myClass* class with the div

- replaceWith() performs the opposite:

```
$('.myClass').replaceWith('<div>Some Content</div>');
```

# Effects

- There are a number of easy-to-use effects supported by jQuery
  - `animate()`
  - `fadeIn()`
  - `fadeOut()`
  - `hide(), show()`
  - `slideDown(), slideUp()`
  - `toggle()`

# A Few jQuery Coding Best Practices

- Create Layered Architectures
  - Avoid inlining CSS and JavaScript

```
...
```

- When using selectors:
  - Prefer id selectors, followed by HTML selectors
  - Class selectors are the slowest performing

```
$('div.myclass') is better than $('.myclass')
```

# jQuery Coding Best Practices

- Don't repeat selectors

```
$("li ul").hide();
$("li ul").slideUp();
$("li ul").show();
```



- Instead use:

*or*    var ul = \$("li ul").hide();
ul.slideUp();
ul.show();

- Use a context for searches:

```
$('li a.myClass', '#myDiv')
```

is better than

```
$('a.myClass')
```

# jQuery Coding Best Practices

- Allow events to bubble
- Use delegation (handle them at a higher level)
  - Reduces and simplifies redundant code

```
$('#topMenu').bind('click', function(e) {
 var target = e.target;
 var $target = $(target);
 if (target.nodeName === 'li')
 $target.addClass('clicked');
});
```

- Opt to use **\$varName = \$(varName)** to indicate a jquery-wrapped obj (shown with target above)

# Ajax and jQuery

```
var makeRequest = function() {

 var xhrConfig = {
 url : "ajaxdemo.txt",
 type: 'GET',
 data: { 'address' : '123 Main St.' },
 headers: { 'Accept': 'text/html' },
 dataType: 'html',// can be json, script, xml, html
 success: function(results) {
 $("#resultsDiv").text(results);
 }
 };

 $.ajax(xhrConfig);
};

$('#clickDiv').click(makeRequest);
```

Create settings obj with needed params

Update a <div>

Make Ajax request

Set up event handling

*examples/jquery/11\_jquery\_ajax.html*

# jQuery Cheat Sheet

**JQUERY 1.7  
VISUAL CHEAT SHEET**

○ ○ ○ ○ ○ ● ○ ○

| jQuery.fx.interval                                          |  | N |
|-------------------------------------------------------------|--|---|
| <i>The rate (in milliseconds) at which animations fire.</i> |  |   |

| jQuery.fx.off                           |  | O-1 |
|-----------------------------------------|--|-----|
| <i>Globally disable all animations.</i> |  |     |

| .hide()                                                                                                                   |  | JQ |
|---------------------------------------------------------------------------------------------------------------------------|--|----|
| <i>Hide the matched elements.</i><br><code>Hide duration [ callback ]<br/>Hide([duration] [, easing] [, callback])</code> |  |    |

| .queue()                                                                                                                                                                                |  | O |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|---|
| <i>Show the queue of functions to be executed on the matched elements.</i><br><code>queue([queueName])<br/>queue([queueName], newQueue)<br/>queue([queueName], callback[ next ])</code> |  |   |

| .show()                                                                                                                      |  | JQ |
|------------------------------------------------------------------------------------------------------------------------------|--|----|
| <i>Display the matched elements.</i><br><code>Show duration [ callback ]<br/>Show([duration] [, easing] [, callback])</code> |  |    |

| .slideDown()                                                                                                                                                  |  | JQ |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------|--|----|
| <i>Display the matched elements with a sliding motion.</i><br><code>slideDown(duration [, callback])<br/>slideDown([duration] [, easing] [, callback])</code> |  |    |

| .slideToggle()                                                                                                                                                            |  | JQ |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|----|
| <i>Display or hide the matched elements with a sliding motion.</i><br><code>slideToggle(duration [, callback])<br/>slideToggle([duration] [, easing] [, callback])</code> |  |    |

| jQuery.ajax()                                                                                                               |  | jqXHR |
|-----------------------------------------------------------------------------------------------------------------------------|--|-------|
| <i>Perform an asynchronous HTTP (Ajax) request.</i><br><code>jQuery.ajax(url [, settings])<br/>jQuery.ajax(settings)</code> |  |       |

| .ajaxComplete()                                                                                                                                                          |  | JQ |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|----|
| <i>Register a handler to be called when Ajax requests complete. This is an Ajax Event.</i><br><code>.ajaxComplete(handlerEvent, jqXHR, ajaxSettings, ajaxOptions)</code> |  |    |

| .ajaxError()                                                                                                                                                                       |  | JQ |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|----|
| <i>Register a handler to be called when Ajax requests complete with an error. This is an Ajax Event.</i><br><code>.ajaxError(handlerEvent, jqXHR, ajaxSettings, textStatus)</code> |  |    |

| jQuery.ajaxPrefilter()                                                                                                                                                                                                      |  | N |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|---|
| <i>Handle custom Ajax options or modify existing options before each request is sent and before they are processed by \$ ajax().</i><br><code>jQuery.ajaxPrefilter(dataType, handlerOptions, originalOptions, jqXHR)</code> |  |   |

| .ajaxSend()                                                                                                                                                |  | JQ |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|--|----|
| <i>Attach a function to be executed before an Ajax request is sent. This is an Ajax Event.</i><br><code>.ajaxSend(handlerEvent, jqXHR, ajaxOptions)</code> |  |    |

| jQuery.ajaxSetup( options )                                                                   |  |  |
|-----------------------------------------------------------------------------------------------|--|--|
| <i>Set default values for future Ajax requests.</i><br><code>jQuery.ajaxSetup(options)</code> |  |  |

| .ajaxStart( handler ) |  |  |
|-----------------------|--|--|
| <i></i>               |  |  |

| jQuery.getJSON()                                                                                                                                       |  | jqXHR |
|--------------------------------------------------------------------------------------------------------------------------------------------------------|--|-------|
| <i>Load JSON-encoded data from the server using a GET HTTP request.</i><br><code>jQuery.getJSON(url [, data], success(data, textStatus, jqXHR))</code> |  |       |

| jQuery.getScript()                                                                                                                                           |  | jqXHR |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|--|-------|
| <i>Load a JavaScript file from the server using a GET HTTP request, then execute it.</i><br><code>jQuery.getScript(url [, success(data, textStatus)])</code> |  |       |

| .load()                                                                                                                                                                                 |  | JQ |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|----|
| <i>Load data from the server and place the returned HTML into the matched element.</i><br><code>.load(url [, data], complete(responseText, textStatus, jqXHR), callbackFunction)</code> |  |    |

| jQuery.param()                                                                                                                                 |  |  |
|------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| <i>Create a serialized representation of an array object, suitable for use in a URL.</i><br><code>jQuery.param(obj)<br/>jQuery.param(o)</code> |  |  |

| jQuery.pq()                                                 |  |  |
|-------------------------------------------------------------|--|--|
| <i>Load data / response.</i><br><code>jQuery.pq(url)</code> |  |  |

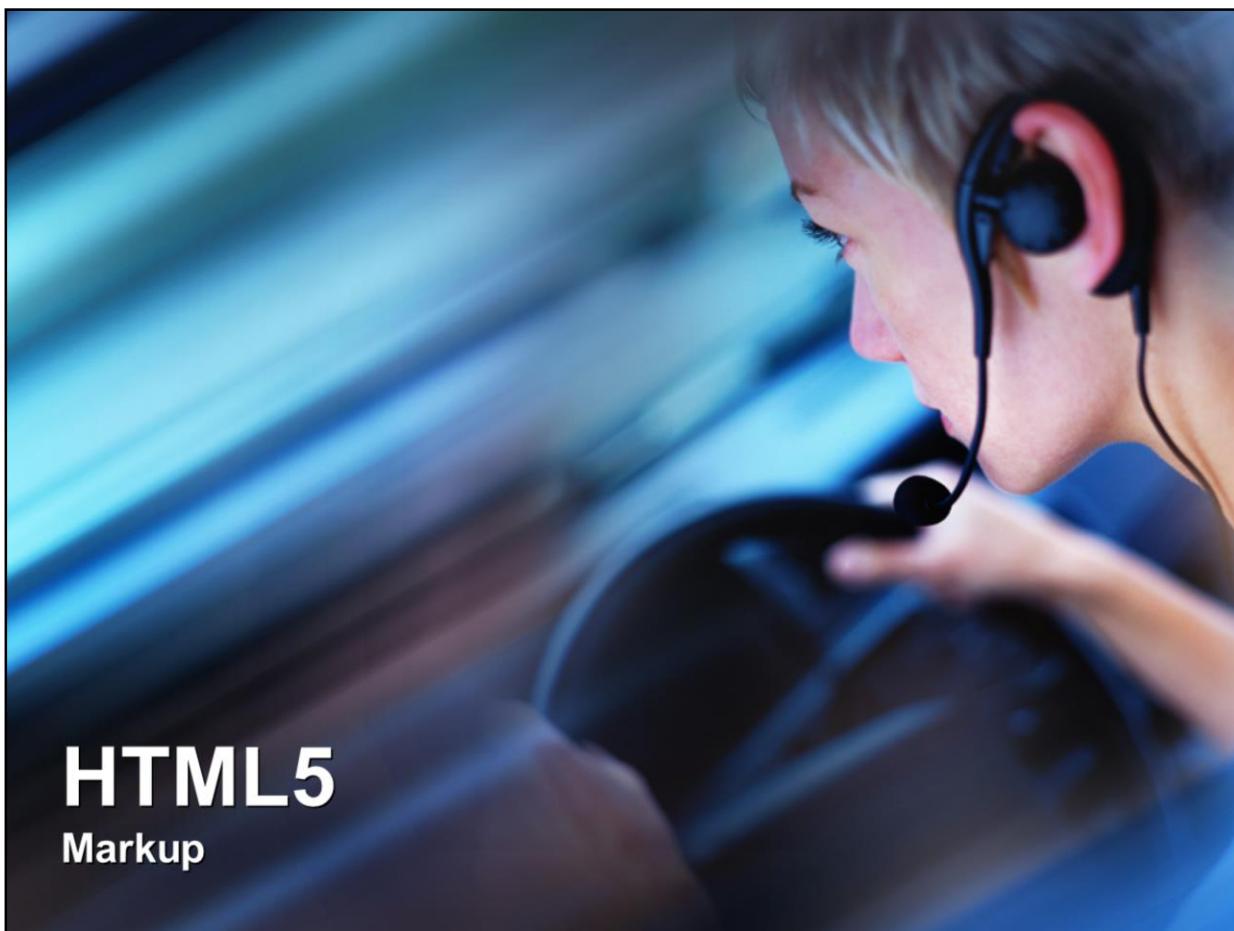
<http://woorkup.com/wp-content/uploads/2011/12/jQuery-17-Visual-Cheat-Sheet1.pdf>

# Summary

- Use DOM APIs to manipulate the look and feel of a rendered page
  - Do this by:
    - changing node styles
    - Adding/removing nodes and content
- Cross-browser event handling can best be handled by using a JavaScript library
- Timers provide an ability to perform what appears to be multi-threaded capabilities

## Lab 3 – HTML5 Markup and jQuery

- Create the initial client to the application using HTML5 Markup, support for the HTML5 Shim, and jQuery
  - *Presentation (CSS) is not a concern at this point*
  - *You will begin with the creation of index.html and modify it over the next few exercises*
  - *Follow the instructions at the back of your manual.*



# HTML5

## Markup

# HTML5 Markup

Major Page Changes (doctype, root, head, etc.)

New Sectional Elements

Other New Elements/Attributes

Addendum: Ajax and jQuery

**New Form Capabilities**

Embedded Content (<audio>, <video>, <canvas>)

## Changes To Forms

- Many New Features within HTML forms
  - New input types
  - Use of required fields, placeholders, and autofocus
  - Validation capabilities
  - Form elements outside of the <form> tag

# Placeholders

- Identifies hint text within the form element

Search:

```
Search: <input type="search" name="term"
placeholder="Enter search term here">
```

Email:

```
<input type="email"
placeholder="Enter your email">
</input>
```

# Placeholders

- To ensure cross-browser support for non-conforming browsers:
  - Either use it as a progressive-enhancement feature for browsers that support it
  - Or use JavaScript to detect the feature and then set the placeholder value

```
if (!("placeholder" in document.createElement("input"))){
 //use fallback JavaScript
}
```

Placeholder property detection code

# Autofocus

- Sets an element to be the default starting element in the form (can only be used once in the form)

```
<input type="text" name="twitter-status" autofocus>
```

- Autofocus may be used as an enhanced or with JavaScript for Yahoo in pre-HTML5 browsers

```
if (!("autofocus" in document.createElement("input"))){
 //use fallback JavaScript
 Y.one('#sf input').focus();
}
```

autofocus property detection code

## Required Fields

- Makes an input a required field

```
<input type="password" name="password" required>
```

- The attribute is ignored by pre-HTML5 browsers
- JavaScript validation will still be necessary when form validation is needed

# New <input> Types

```
<form>
 <input type="email"></input>
 <input type="number" min="0" max="10"></input>
 <input type="month" min="2010-01" max="2010-12"/>
</form>
```

These degrade to type="text" for  
browsers that don't recognize the new  
type attribute

# New <input> Types

- <input> has new type attribute values:

– tel	These may be used, recognizing that some browsers add additional styling such as spinners to fields
– number	
– url	
– email	
– search	← May be used with caveats (see footnotes)
– range	
– color	
– datetime, datetime-local, date, month, week, time	

For the input types that perform validation, visual feedback should be provided as to the validity of the data by using the :valid and :invalid CSS pseudo-classes

The other input types (the lower half of the ones described above) should generally be avoided at this time due to specification maturity or lack of fallback capabilities.

All browsers add a small "x" that can be used to clear the value for the search input type. Use CSS styling to ensure a consistent look and feel. On Safari (MAC OSX), use the following to help remove the browser's default styling:

```
input[type="search"] {
-webkit-appearance: textfield;
}
```

## New <input> Constraints

- New constraint attributes have been added to help support validation criteria

- **autocomplete**

- **min**

- **max**

- ```
<input type="month" min="2010-01"
       max="2010-12" required />
```

- **multiple**

- **pattern**

- ```
<input type="text" name="email"
 autocomplete="off">
```

- **step**

# The Pattern Attribute

- **pattern** provides ability to validate (client-side) using a regex before submission occurs
  - **title** provides a message to be displayed

```
<input type="tel" name="phone"
 pattern="^(\d{3})\d{3}-\d{4}"
 title="U.S. phone including zip" />
```

- If invalid, **:invalid pseudo-class** is applied to the input
- If value doesn't match form will not submit

# Implementing the Pattern Attribute

- Ignored by pre-HTML5 browsers, fallback required:
  - Use JavaScript to validate against this property
  - Use `:invalid` CSS pseudo-class when field is invalid

pattern property detection code

```
if (!("pattern" in document.createElement("input"))){
 //use fallback (next slide)
}
```

# CSS3 Pseudo-classes and Forms

- The pattern input constraint doesn't match, the input will be *invalid*
- CSS3 provides a number of pseudo-classes that may capture styles when elements exist in particular states

```
:required { }
:optional { }
:valid { }
:invalid { }
:default { }
:in-range { }
:out-of-range { }
```

These pseudo classes will be automatically applied to input elements when they exist in a particular state. They are similar to :hover, but for inputs instead.

# Browser Default Appearances

- New inputs have varying appearances in browsers
  - Use CSS to remove these effects

```
<input type="number" min="0" max="10" />
```



asdf

FF4 allows invalid values to be typed, adds a red border for invalid conditions. Also displays a message upon submitting.



2

Chrome provides a spinner, allows only numbers to be typed in

```
input[type=number]::-webkit-inner-spin-button {
 -webkit-appearance: none;
}

removes spinners
```

Other useful values include:

```
/* removes the 'x' from a type='search' input */

input[type="search"]::-webkit-search-cancel-button {
 -webkit-appearance: none;
}
```

```
/* removes rounded corners from a search input box as well */

input[type="search"] {
 -webkit-appearance: textfield;
 -webkit-box-sizing: border-box;

}
```

## Lab 03b: New Form Capabilities

- Working from your previous solution, convert the contact search form to use the new input types:
  - Convert it to type="search"
  - Include **placeholder**, **required** and **autofocus** capabilities
  - Use **pattern** for restricting its use to alpha-numeric values only
  - Add styling to present **:invalid** and **:valid** conditions on the input and form elements

What do you think of this behavior?

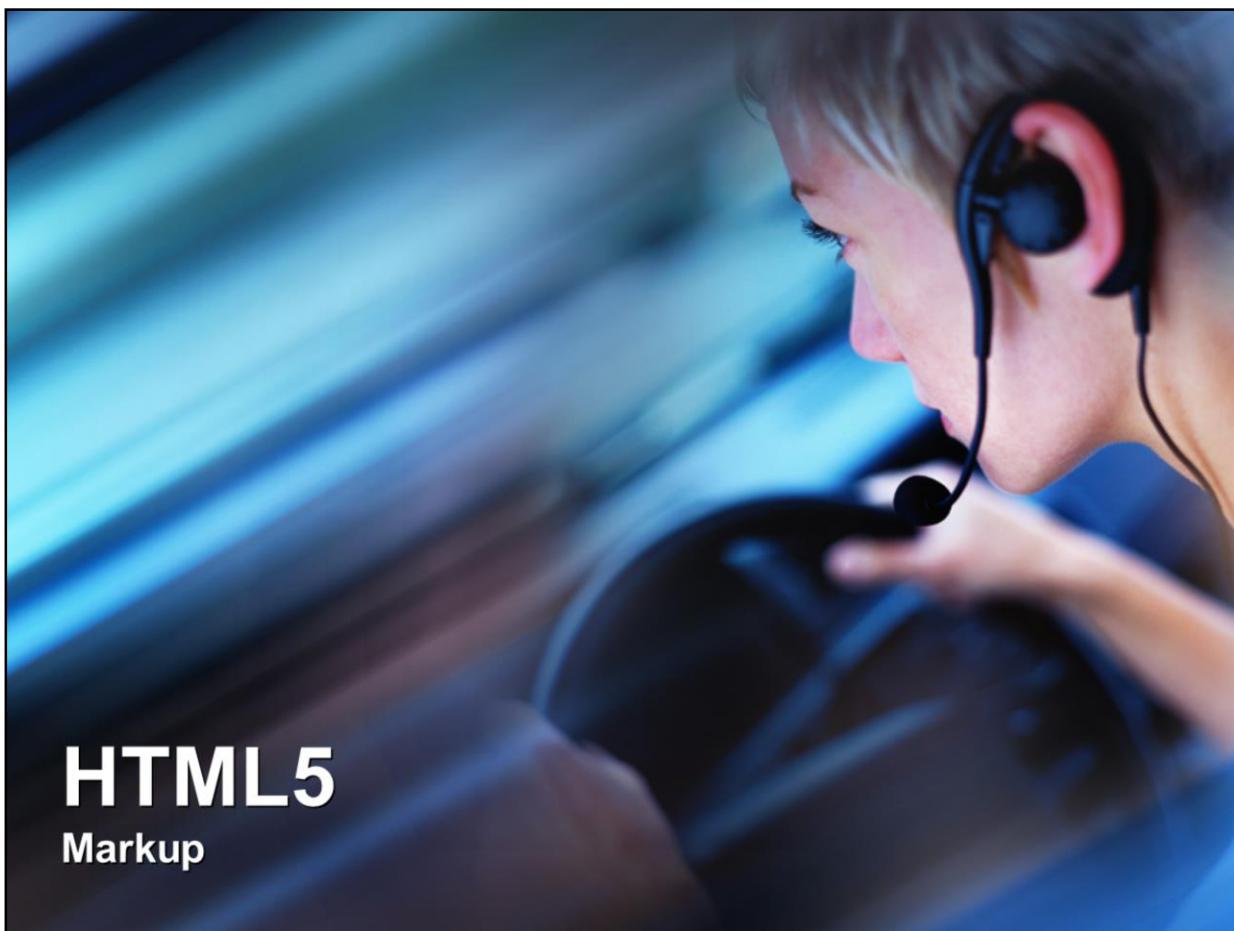
Note that using type="search" causes Chrome to behave a little differently. You will likely need to use CSS to manage its look and feel by forcing it to act like a text input. Do this with the following CSS:

```
input[type="search"] {
 -webkit-appearance: textfield;
 -webkit-box-sizing: border-box;
}
```

The following will remove the "x" from the search element if desired:

```
input[type="search"]::-webkit-search-cancel-button {
 -webkit-appearance: none;
}
```





# HTML5

## Markup

# HTML5 Markup

Major Page Changes (doctype, root, head, etc.)

New Sectional Elements

Other New Elements/Attributes

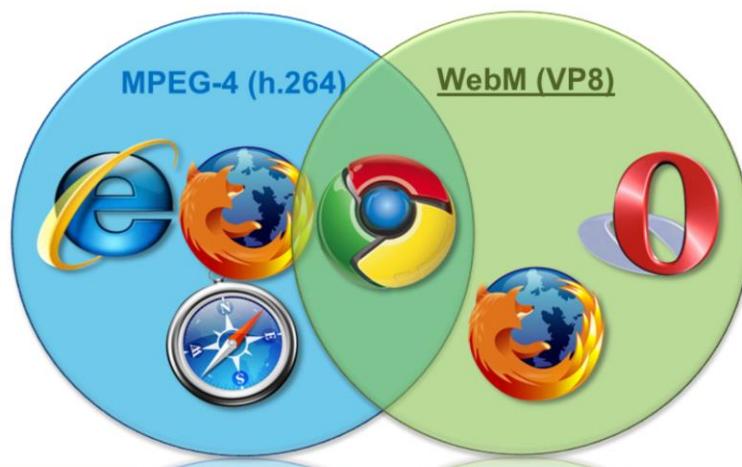
Addendum: Ajax and jQuery

New Form Capabilities

Embedded Content (<audio>, <video>, <canvas>)

## <video>

- Two video standards have arisen:



Flash supports mostly h.264,  
doesn't support WebM

As of Firefox 21, it will support h.264  
using your OS native capabilities

Chrome has recently dropped support for h.264 (as of January 2011).

## <video>

- A <source> must be specified in a format supported by that browser
- Multiple <source> elements may be specified and the browser will try them **first to last**
- iOS only supports h.264 and only supports a **single** <source> element, so it needs to be specified first
  - Omit the type on the h.264 source as Android devices will mistakenly interpret it
  - All browsers understand the type based on filename extension

Widest support is with the MPEG-4 format due to its support within IE and Flash.

# <video> Cross Browser

- The following works in the widest implementations

```
<video id="movie" width="320" height="240" controls>
 <source src="pr6.mp4">
 <source src="pr6.webm" type='video/webm; codecs="vp8,
 vorbis"'>
 <object width="320" height="240"
 type="application/x-shockwave-flash"
 data="flowplayer-3.2.1.swf">
 <param name="movie" value="flowplayer-3.2.1.swf">
 <param name="allowfullscreen" value="true">
 <param name="flashvars" value='config={"clip": {"url":
 "http://wearehugh.com/dih5/good/bbb_480 }}'>
 <p>Download video as MP4 or
 WebM.</p>
 </object>
</video>
```

The controls attribute specifies that the browser should render the default controls for video playback. This attribute should be used unless custom video controls are implemented.

## <video> and Firefox

- FF < 4
  - Does not support WebM or MPEG-4
  - It does not properly fall back to the flash version
  - JavaScript is required for inserting video:

```
var video = document.createElement("video");
if (video.canPlayType && (video.canPlayType("video/mp4") ||
 video.canPlayType("video/webm"))){
 //insert <video> code
} else {
 //insert Flash player
}
```

- However, this solution doesn't degrade for non-JavaScript FF users

# <video> with FF Support

- To avoid the JavaScript implementation, FF3.5 supports the OGG format:

```
<video id="movie" width="320" height="240" controls>
 <source src="pr6.mp4">
 <source src="pr6.webm" type='video/webm; codecs="vp8, vorbis"'>
 <source src="pr6.ogv" type='video/ogg; codecs="theora, vorbis"'>
 <object width="320" height="240"
 type="application/x-shockwave-flash"
 data="flowplayer-3.2.1.swf">
 <param name="movie" value="flowplayer-3.2.1.swf">
 <param name="allowfullscreen" value="true">
 <param name="flashvars" value='config={"clip": {"url":
 "http://wearehugh.com/dih5/good/bbb_480" }}'>
 <p>Download video as MP4 or
 WebM.</p>
 </object>
</video>
```

The final version, shown here, now provides support for all major browsers. However, this solution would require providing video in 3 formats. This is not common. At the least, the MP4 format should be supported for the widest support.

# <video> Test Your Browser's Support

- youtube.com/html5



By joining HTML5 beta on youtube.com/html5 and by using a browser that supports HTML5 video, you can be served non-flash-based content (in other words content via the <video> tag!).

## <audio>

- Audio is supported in much the same way as <video>
- Browsers support WAV, MP3, Ogg Vorbis
- Prefer MP3 format if only one choice exists as it is supported by IE and Flash
- Best to provide <audio> tag and fallback to Flash if not supported

```
<audio controls>
 <source src="intro.mp3">
 <source src="intro.ogg">
</audio>
```

## <canvas>

- Specifies a region on a page to render images, 2d chart, games, etc., using a JavaScript API

```
<canvas id="canvasApp" width="200" height="200">
This text is displayed if your browser does not
support HTML5 Canvas.
</canvas>
```



```
<script>
 var node = document.getElementById('canvasApp');
 var context = node.getContext('2d');
 context.fillStyle = "rgb(0, 255, 0)";
 context.fillRect(150, 30, 250, 50);
</script>
```

For IE 7/8, ExplorerCanvas shim may be used in a limited way for 2D operations.

## <svg>

- SVG (Scalable Vector Graphics) is a vector-based markup language and relies on the DOM
  - Not supported in IE6-8, nor truly a part of HTML5

```
<svg xmlns:svg="http://www.w3.org/2000/svg"
 xmlns="http://www.w3.org/2000/svg" viewBox="0 0 200 100"
 width="200px" height="100px">

 <circle cx="50" cy="50" r="30"
 style="stroke:none; fill:#ff0000;" />

 <rect x="60" y="20" height="60" width="60"
 style="stroke:none; fill:#00ff00;
 fill-opacity: 0.5;" />
</svg>
```

## Summary

- There are many new HTML5 elements available for use right now
- Ensure proper use of doctype, tag semantics, and above all: ensure solutions continue to work across all A-grade browsers



# HTML5

## Presentation Layer

# CSS 3

## CSS Key Features

CSS Box Model

CSS3 Overview

New Features

# How CSS "Hooks" into HTML

- CSS uses selectors to bind to HTML elements
  - They do this in several ways:

**ID selectors: #quote1**

```
<p id="quote1">No great thing is created suddenly</p>
<p class="author">Epictetus</p>
```

**Class  
selectors:  
.author**

```
→ <div id="quote2">
 <p class="proverb">Inches make champions
 <cite>Vince Lombardi</cite>
 </p>
</div>
```

**HTML selectors: div cite**

**Combinations of these selectors:  
#quote2 .proverb cite**

# CSS Rule Placement

## 1. **Inline** (within the HTML tag)

```
<p style="padding: 5px">This is...</p>
```

Don't use this as it violates  
a layered architecture

# CSS Rule Placement

## 2. Embedded

```
<html>
 <head>
 <style type="text/css">
 p {padding: 5px; }
 </style>
 </head>
 <body>
 <p>This is...</p>
```

# CSS Rule Placement

## 3. External (using link tag)

```
<html>
 <head>
 <link
 href="styles.css"
 rel="stylesheet"
 type="text/css"
 />
 </head>
```

The diagram shows the XML structure of an external CSS link tag. Annotations with arrows point to specific attributes:

- An arrow points to the `href="styles.css"` attribute with the text "absolute or relative path".
- An arrow points to the `rel="stylesheet"` attribute with the text "Relationship between external document and the HTML source".
- An arrow points to the `type="text/css"` attribute with the text "content type".

External CSS files are cached by browser – which can help with performance when you reuse the same CSS rules on multiple pages

# Using Shorthands

The following three rules express the same thing:

```
body {
 margin-top: 5px;
 margin-right: 5px;
 margin-bottom: 5px;
 margin-left: 5px;
}

body { margin: 5px 5px 5px 5px; }

body { margin: 5px; }
```

The order of the middle example is “top, right, bottom, left” (clockwise from top)

# Shorthands

- Set or Skip

```
body {
 padding-right: 5px;
 padding-bottom: 5px;
}
```

Browser's default (or inherited value) will apply to missing styles

- Symmetry Shorthand

```
body { padding: 5px 5px; }
– top/bottom, left/right
```

- Side to Side Symmetry

```
body { margin: 5px 5px 5px; }
– top, right/left, bottom
```

# Shorthand - Properties

```
div {
 border-width: 5px;
 border-style: solid;
 border-color: #CCC;
}
```

```
div {
 border-top-width: 5px;
 border-right-width: 5px;
 border-bottom-width: 5px;
 border-left-width: 5px;
 border-top-style: solid;
 border-right-style: solid;
 border-bottom-style: solid;
 border-left-style: solid;
 border-top-color: #CCC;
 border-right-color: #CCC;
 border-bottom-color: #CCC;
 border-left-color: #CCC;
}
```

```
div { border: 5px solid #CCC; }
```

Order of values in the shorthand version can make a difference in some older browsers

# Most Common CSS Properties

## ***Text***

### ***Formatting***

color  
letter-spacing  
text-align  
text-decoration  
text-indent  
text-transform  
line-height

## ***Font***

### ***Related***

font-size  
font-family  
font-weight  
font-style  
font-variant  
font  
white-space

## ***Color and***

### ***Background***

background-color  
background-image  
background-position  
background-repeat  
background-attachment  
background-color

## ***Box Model***

width  
height  
border  
margin  
padding  
overflow  
float  
clear  
position  
top, left,  
bottom, right  
display  
z-index  
visibility

## ***Lists***

list-style-image  
list-style-position  
list-style-type  
list-style

## ***Links***

a:link, a:visited, a:hover,  
a:active

# Font Properties

- Some common font properties:

```
font-family: [family name/generic family]
font-style: [normal|italic|oblique]
font-size: [keyword or unit]
font-weight: [keyword]
font-variant: [normal|small-caps]
```

- Can be combined into shorthand:

```
font: italic bold 12px arial sans-serif;
```

font-style  
font-weight  
font-size  
font-family

Not recommended to use font shorthand: if you don't specify the font-weight, font-style, or font-variant then these values will automatically default to a value of "normal" (which may override any normally inherited values!)

# Text Properties

- Some common text properties:

```
letter-spacing: [normal|+ or - value]
word-spacing: [normal|+ or - value]
text-indent: [+ or - length or % value]
text-align: [left|right|center|justify]
text-decoration: [none|line-through|underline]
text-transform: [lowercase|uppercase|capitalize]
white-space: [normal|nowrap|pre]
vertical-align: [super|top|middle|bottom|sub]
line-height: [normal|any unit]
```

Text decoration can have multiple values at the same time

“pre” will retain preformatted text white-spacing

“line-height” defines the spacing between the lines (eg, single space, double space etc...)

# HTML (Type) Selector

```
p { margin: 5px }
```

```
<body>
 <div>
 <p>Lorem ipsum dolor...</p>
 </div>

<p class="intro">This is a paragraph</p>
```

Targets *all* instances of a particular element

## Descendant Selector

```
ul em { text-decoration: underline; }
```

```
<body> targets this

 first item
 second item

<p>This is a great paragraph</p>
```

↑  
not this

Targets any element that is a descendent

# Universal Selector

```
* { margin: 5px; }
```

- Can be combined with other selectors

```
ul * { margin: 5px; }
```

```
div > * { margin: 5px; }
```

Targets *all* elements

# Class Selectors

- A form of selector, using the “class” attribute
- Classes can be applied to multiple elements on a page, allowing re-use of declarations
- Uses dot notation within CSS

```
.warning { color: red; }
```

```
<div class="warning"><p>Florida continues to get battered by
hurricanes.</p></div>
```

```
<p>Beware of the hurricanes.</p>
```

Notice we're using the same class “warning” on different elements

# Class Selectors

- Can be referenced by a particular type of element

```
.warning { color: red; }
span.warning { font-weight: bold; }
```

```
<div class="warning"><p>Florida continues to get
battered by hurricanes.</p></div>
```

```
<p>Beware of the hurricanes.</p>
```

**span.warning** and **span .warning** are different!

# Class Selectors

- Elements can declare multiple classes

```
<div class="warning first"><p>Florida continues to
get battered by hurricanes.</p></div>
```

- UI Libraries use classes extensively

```
<div role="presentation" id="widget_searchInput"
 class="dijit dijitReset dijitInline dijitLeft dijitTextBox" widgetid="searchInput">

 <div class="dijitReset dijitInputField dijitInputContainer">
 <input type="text" autocomplete="off" data-dojo-attach-point="textbox,focusNode"
 class="dijitReset dijitInputInner" tabindex="0" id="searchInput" value="">
 </div>
</div>
```

# ID Selectors

- Similar to class selectors, but each ID attribute must be unique on a page
- Use hash (#) notation

```
#mainNav { margin: 5px; }
#mainNav li { float: left; }
[...]

<ul id="mainNav">
 Home
 About

```

# Pseudo-Classes

- A set of rules that target the different states of an element (usually an anchor):

```
a:link {color: #00F;}
a:visited {color: #0F0;}
a:hover {color: #C0F;}
a:active {color: #F00;}
```

- You may combine these pseudo-classes with other selectors to be more specific:

```
a.external:link {color: #93F;}
a:link img {color: #93F;}
```

The browser already understands that these states exist, no classes necessary

Order is important because several of these states can exist at the same time (e.g. a visited link that you're also hovering over). Order determines which rule takes effect.

## Before and After Pseudo Elements

- Can be used to insert generated content before or after an element's *content* (not before or after the element itself)

```
p.caution:before { content: "Beware: " }
p.caution:after { content: "!!!" }
[...]
```

```
<p class="caution">Don't talk to strangers</p>
```

Renders:

“**Beware:** Don’t talk to strangers!!!”

You should not use :before and :after for general use, but it can be very useful as a hack (presented in the clearfix discussion at the end of the next chapter).

# Using Class Names vs. ID's

- Rules for IDs:
  - ID's must be unique on a page
  - ID's are useful for targeting an element in JavaScript
  - ID's for main sections on a page  
(e.g. mainNav, footer, etc.)
- Rules for classes:
  - Use classes for different instances (e.g. "first", "last", "headline", "alternate"..)
  - Elements may have both an ID and class name(s)

# Choosing Selectors

- There are a multitude of options for how to construct a given selector

```
<div class="mod">
 <ul id="mainNav">
 home
 about
 careers

</div>
```

What color will this anchor be?

all refer to

```
a { color: white; }
#aboutLink { color: blue; }
a#aboutLink { color: green; }
#mainNav li #aboutLink { color: yellow; }
div.mod ul#mainNav li a#aboutLink { color: red; }
.mod #aboutLink { color: black; }
```

The answer is red. The reason is explained on the next slide.

# Selector Weight (Specificity)

- Calculated as follows:
  - Count the ID attributes ( = a )
  - Count the Class or other attributes ( = b )
  - Count the elements ( = c )
- Concatenating a+b+c gives you the selector weight

```
* {} /* a=0 b=0 c=0 -> specificity = 0 */
li {} /* a=0 b=0 c=1 -> specificity = 1 */
ul li {} /* a=0 b=0 c=2 -> specificity = 2 */
ul ol+li {} /* a=0 b=0 c=3 -> specificity = 3 */
h1 + *[REL=up] {} /* a=0 b=1 c=1 -> specificity = 11 */
ul ol li.first {} /* a=0 b=1 c=3 -> specificity = 13 */
li.first.level {} /* a=0 b=2 c=1 -> specificity = 21 */
#x34y {} /* a=1 b=0 c=0 -> specificity = 100 */
```

Specificity is a misleading term: longer selectors can appear to be more specific, but can actually have lower weight (last, and next to last examples above)

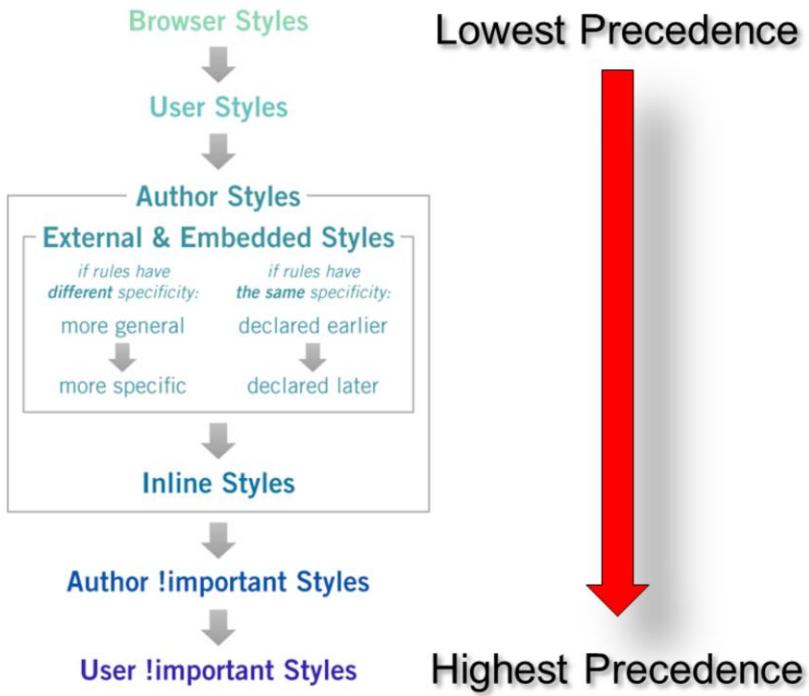
# !important

- Takes precedence over all other rules

```
div {
 font-size: 1.5em;
 color: #000 !important;
 padding: 0.2em;
}
```

- Generally, use it primarily for debugging!

# Cascade Precedence



<http://www.communitymx.com/content/article.cfm?page=2&cid=2795D>

# Normalizing Browser Styles

- Browsers yield different default style values
- It's imperative to set a level playing field
- **reset.css**

<http://yui.yahooapis.com/combo?2.7.0/build/reset/reset.css>

```
body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,form,fieldset,
input,textarea,p,blockquote,th,td {
 margin:0;
 padding:0;
}
table {
 border-collapse:collapse;
 border-spacing:0;
}
[...]
```

# HTML5 Resets

- With the arrival of HTML5, new values will need to be placed into a reset
  - HTML5 Doctor based theirs on Eric Meyer's HTML4 reset
- <http://html5resetcss.googlecode.com/files/html5reset-1.6.1.css>
- Others exist as well

<http://html5reset.org/>

<http://html5boilerplate.com/>

# HTML5Doctor Reset CSS

```
html, body, div, span, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre,
abbr, address, cite, code, del, dfn, em, img, ins, kbd, q, samp, small, strong, sub,
sup, var, b, i, dl, dt, dd, ol, ul, li, fieldset, form, label, legend, table, caption, tbody,
tfoot, thead, tr, th, td, article, aside, canvas, details, figcaption, figure, footer,
header, hgroup, menu, nav, section, summary, time, mark, audio, video {
 margin:0; padding:0; border:0; outline:0; font-size:100%;
 vertical-align:baseline; background:transparent;
}

article,aside,details,figcaption,figure,footer,header,hgroup,menu,nav,section {
 display:block;
}

a {
 margin:0; padding:0; font-size:100%;
 vertical-align:baseline; background:transparent;
}
```

This sample is only a partial  
listing of the full reset

[html5doctor.com](http://html5doctor.com) Reset Stylesheet

# Background Properties

```
background-color: [keyword|hex|rgb]
background-image: [url(path)]
background-repeat: [no-repeat|repeat-x|repeat-y]
background-position: [x-pos y-pos]
background-attachment: [scroll|fixed]
```

## Shorthand:

```
background: pink url(smiley.jpg) no-repeat 10px 50%;
```

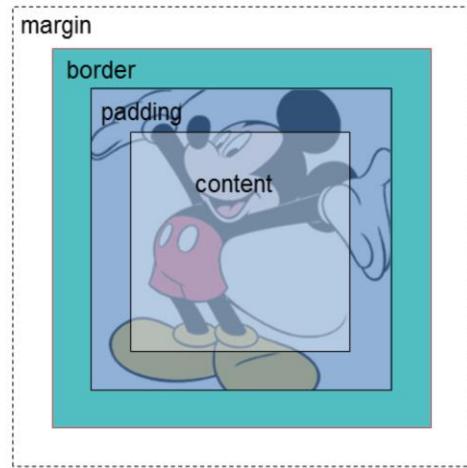


Good time to address using background images re: content vs. presentation

# Background Properties

- Any background properties (image, color etc..) will apply to the **content** and **padding** areas.

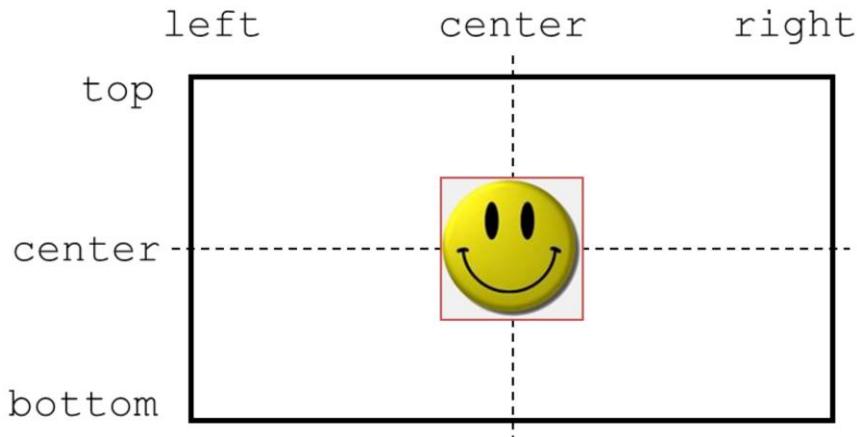
```
{
height: 150px;
width: 150px;
padding: 10px;
border-width: 1px;
margin: 5px;
background: url(mickey.jpg)
no-repeat;
}
```



Margin is transparent: background colors or images will not show in that area.

# Background Positioning

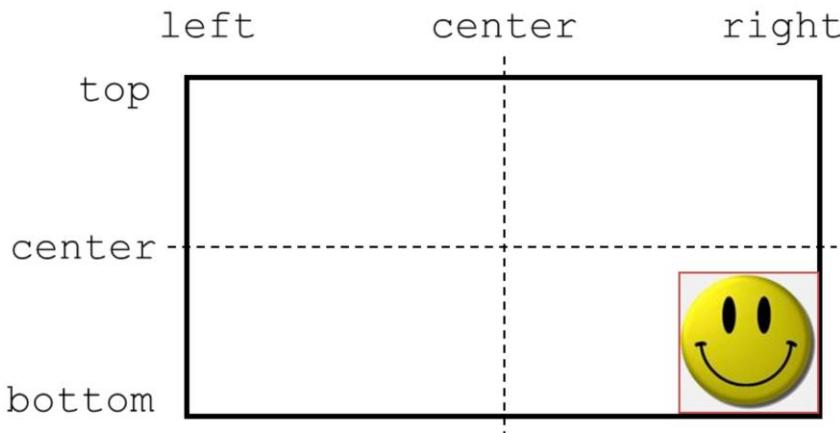
- By keyword



```
background: url(smiley.jpg) no-repeat center center;
```

# Background Positioning

- By keyword

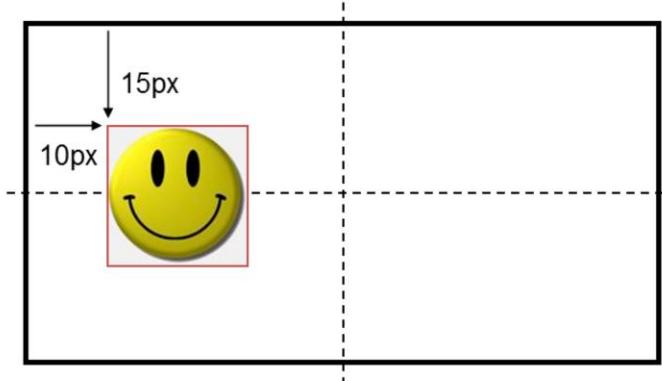


```
background: url(smiley.jpg) no-repeat right bottom;
```

Keyword here sets the right of the image with the right of the container, and the bottom with the bottom

# Background Positioning

- By **fixed unit**

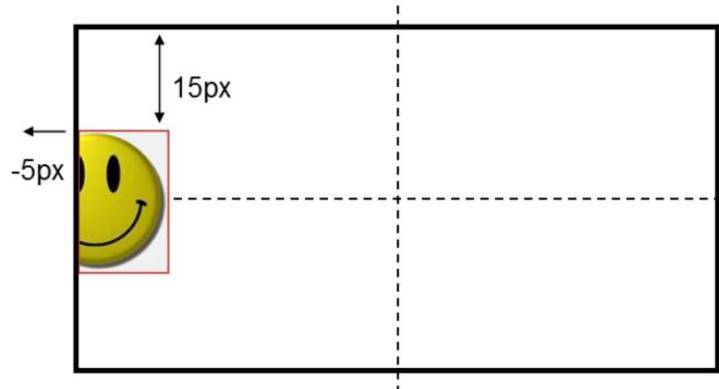


```
background: url(smiley.jpg) no-repeat 10px 15px;
```

Sets the start position (top/left) of the background image within the container element

# Background Positioning

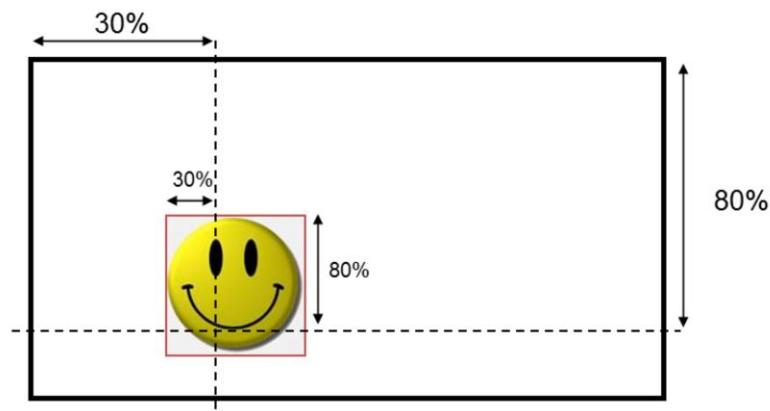
- By **fixed unit**



```
background: url(smiley.jpg) no-repeat -5px 15px;
```

# Background Positioning

- By percentage



```
background: url(smiley.jpg) no-repeat 30% 80%;
```

Matches the x-axis percentage of the container with the same x-axis percentage of the background image

Same with y-axis

Very hard to predict results – usually avoided.

# Image Sprites

- Collection of (typically) smaller background images concatenated onto one image file:



- Reduces HTTP requests
- Improves performance





A photograph of a person wearing a black headset with a microphone, looking down at a computer monitor. The monitor displays a dark interface with some blurred text or graphics. The background is a soft-focus blue and white.

# HTML5 CSS Box Model

# CSS 3

CSS Key Features

**CSS Box Model**

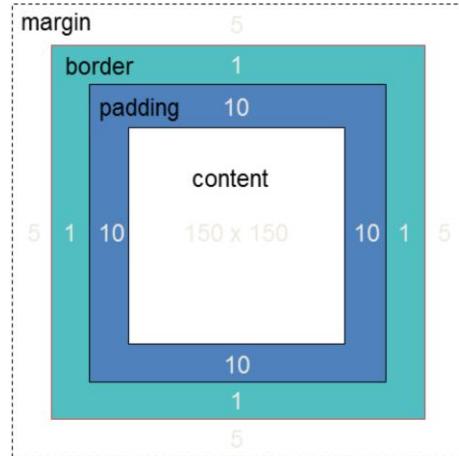
CSS3 Overview

New Features

# Box Dimensions

- Each box has a **content** area (e.g., text, an image, etc.) and optional surrounding **padding**, **border**, and **margin** areas

```
{
 height: 150px;
 width: 150px;
 padding: 10px;
 border-width: 1px;
 margin: 5px;
}
```



Margin is transparent: background colors or images will not show in that area.

# Positioning Schemes

## 1. Normal Flow

- Content flows from top to bottom, and left to right
- Block-level elements generate line breaks

## 2. Absolute Positioning

- Content removed entirely from normal flow and positioned with respect to an ancestor element

## 3. Float

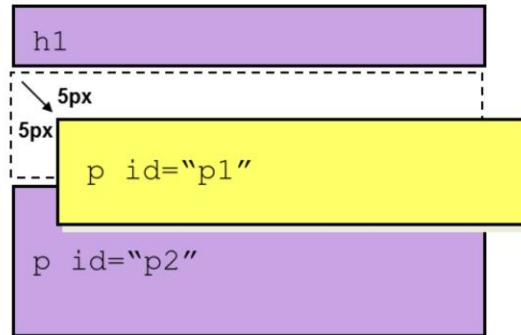
- Content first laid out in normal flow, then taken out of flow and shifted as far right or left as possible

We use the “position” and “float” properties to assign a scheme for each element

# Relative Positioning

- Part of the normal flow scheme
- Should really be called “offset positioning”
- Content is laid out according to normal flow, and then offset from its original position

```
<h1></h1>
<p id="p1"></p>
<p id="p2"></p>
[...]
p#p1 {
 position: relative;
 top: 5px;
 left: 5px;
}
```



The space that the element would have occupied in normal flow is maintained. Element now layers over neighboring elements.

Properties can be top|right|bottom|left

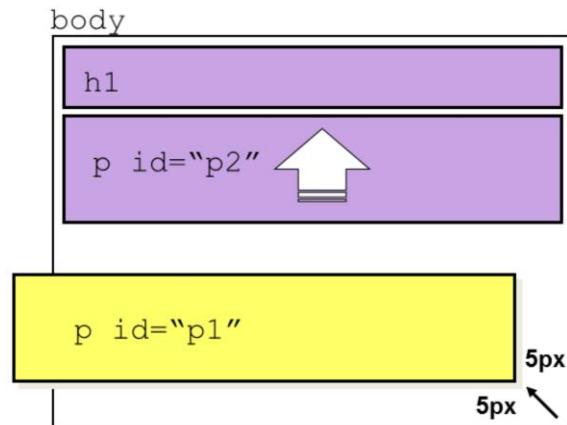
Values can be positive or negative

# Absolute Positioning

- Removed from normal flow scheme

```
<body>
 <h1></h1>
 <p id="p1"></p>
 <p id="p2"></p>
</body>
[...]
```

```
p#p1 {
 position: absolute;
 bottom: 5px;
 right: 5px;
}
```

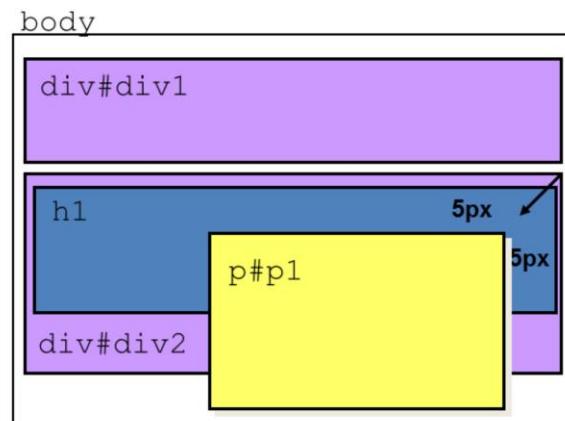


# Absolute Positioning

- Positioned with respect to the body or next highest positioned ancestor
  - Travel up the document tree until encountering an element with position: relative|absolute

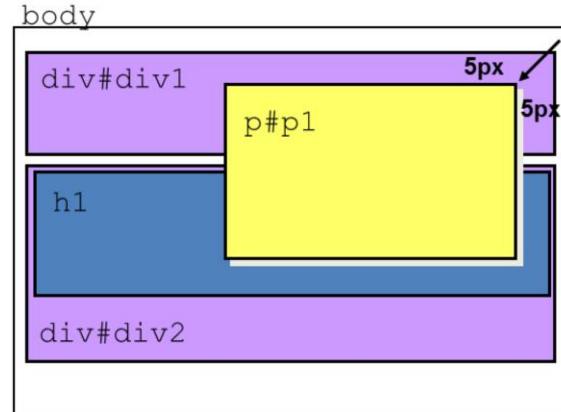
```
<div id="div1"></div>
<div id="div2">
 <h1></h1>
 <p id="p1"></p>
</div>
[...]

p#p1 {
 position: absolute;
 top: 5px;
 right: 5px;
 width: 50%;
}
div#div2 { position: relative; }
```



# Absolute Positioning

```
<div id="div1"></div>
<div id="div2">
 <h1></h1>
 <p id="p1"></p>
</div>
[...]
p#p1 {
 position: absolute;
 top: 5px;
 right: 5px;
 width: 50%;
```

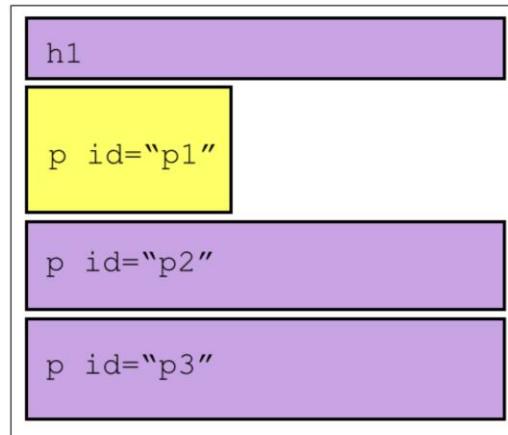


Showcase what happens when positioning is removed from ancestor div (now in relation to the body)

# Floated Elements

- First, content is laid out according to normal flow

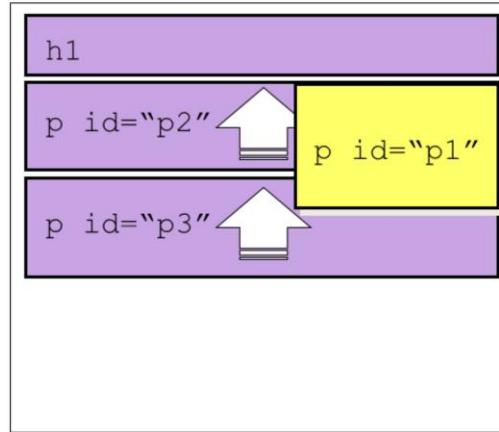
```
<h1></h1>
<p id="p1"></p>
<p id="p2"></p>
<p id="p3"></p>
[...]
p#p1 {
 width: 40%;
```



# Floated Elements

- Then shifted left or right as far as possible
  - Subsequent block elements move up behind floated element

```
<h1></h1>
<p id="p1"></p>
<p id="p2"></p>
<p id="p3"></p>
[...]
p#p1 {
 width: 40%;
 float: right;
}
```



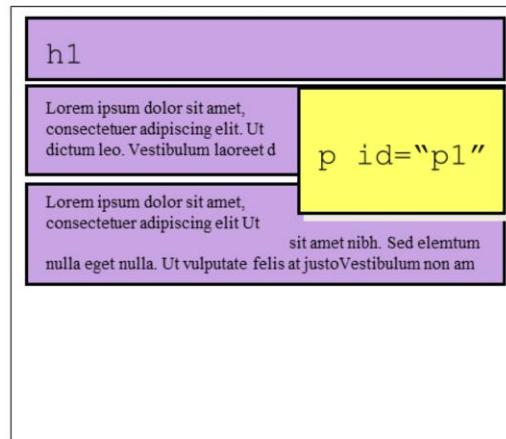
FLOATS ARE IN RELATION TO BLOCK LEVEL ELEMENTS, REMOVED FROM THE FLOW.

# Floated Elements

- Inline content (i.e. text) flows *around* floated elements

```
<h1></h1>
<p id="p1">Lorem...</p>
<p id="p2">Lorem...</p>
<p id="p3"></p>
[...]

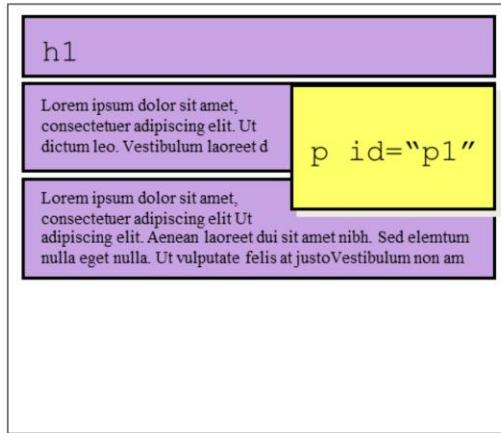
p#p1 {
 width: 40%;
 float: right;
}
```



# Floated Elements

- Floats are block-level elements
  - Will override value of “display” property
  - However, must have an explicit width set, else shrink-to-fit

```
p#p1 {
 width: 40%;
 float: right;
}
```



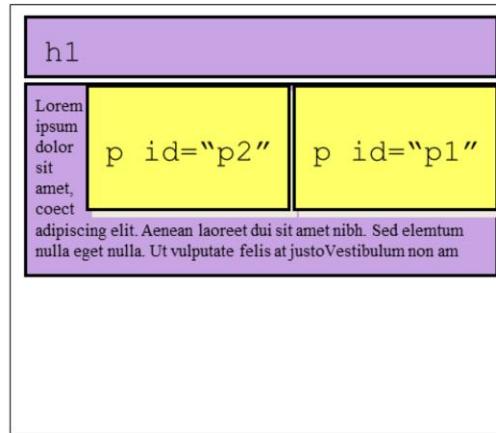
(So, it's not entirely removed from the normal flow) Key difference from absolutely positioned elements.

# Floated Elements

- Floats are shifted left or right until their outer edge touches the containing block edge *or the outer edge of another float*

```
<h1></h1>
<p id="p1">/p>
<p id="p2"></p>
<p id="p3">Lorem...</p>
[...]

p#p1,
p#p2 {
 width: 40%;
 float: right;
}
```



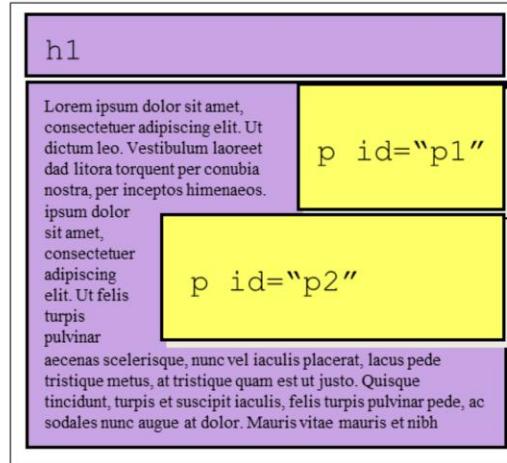
All floated objects interact with each other on the same “pane” or “context”. (vs. Absolutely Pos. elements will stack, each on their own layer.)

# Floated Elements

- If there isn't enough horizontal room on the current line for the float, it is shifted downward, line by line, until a line has room for it.

```
<h1></h1>
<p id="p1">/p>
<p id="p2"></p>
<p id="p3">Lorem...</p>
[...]
```

```
p#p1 {
 width: 40%;
 float: right;
}
p#p2 {
 width: 70%;
 float: right;
}
```



# Floating for Layout

- Nearly every web page uses floating to control layout:

Header	Col1	Col3	Col2
	Col1 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut.	Col3 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut.	Col2 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut.
Footer			

```
<div id="wrapper">
 <div id="header"></div>
 <div id="container">
 <div id="col1"></div>
 <div id="col2"></div>
 <div id="col3"></div>
 </div>
 <div id="footer"></div>
</div>
```

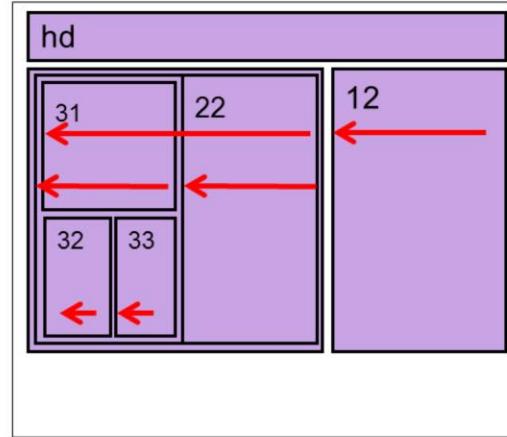
```
#header { background-color: #ffc; }
#col1 { background-color: #fcf;
 float: left;
 width: 25%;
 }
#col2 { background-color: #fcf;
 float: right;
 width: 25%;
 }
#col3 { background-color: #acb; }
#footer { background-color: #abc; }
```

See examples in examples/css/floating0.html

# Two-Columns

- Many Layouts are decomposed into two columns

```
<div id="wrapper">
 <div id="hd"></div>
 <div id="col11">
 <div id="col21">
 <div id="col31"></div>
 <div id="col32"></div>
 <div id="col33"></div>
 </div>
 <div id="col22"></div>
 </div>
 <div id="col12"></div>
</div>
```

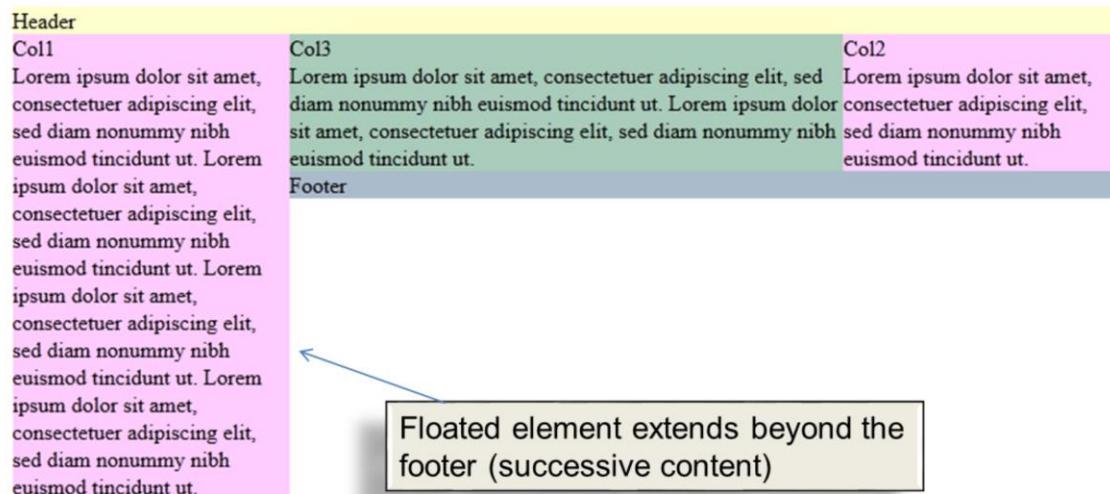


```
#col11, #col21, #col32 { width: 55%; float: left; }
#col12, #col22, #col33 { width: 45%; float: left; }
```

See examples/floating1.html

# Dilemma with Floating

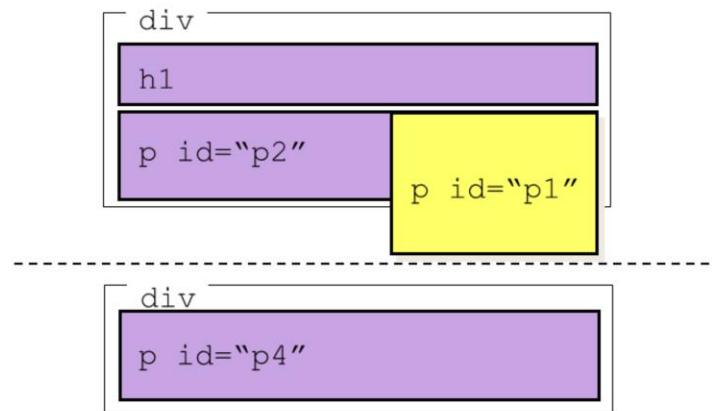
- What happens if floated content gets too large?



See examples in examples/css/floating2.html

# Solving the Dilemma: Clearing

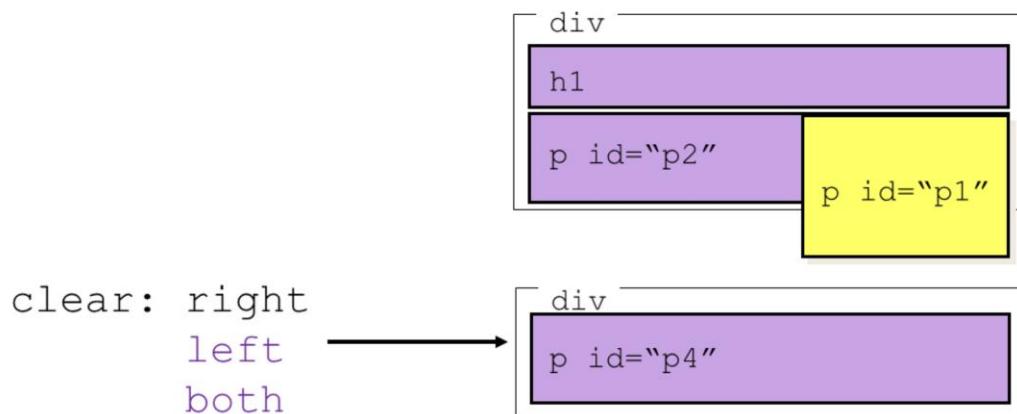
- Often, you want to define a break point where content stops sliding up behind a float



*4 ways to do this...*

# Tactic #1: Clearing Floats

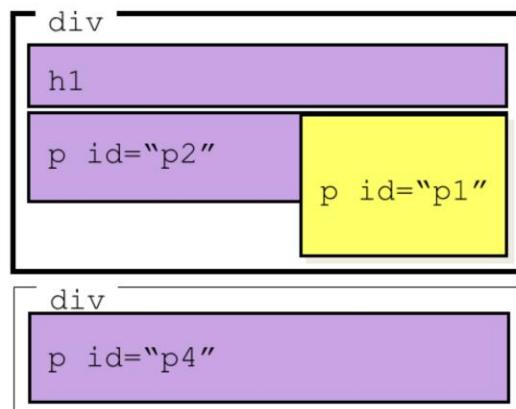
- Use the “clear” property on the subsequent element



The disadvantage to this approach is that the subsequent div must "know" to clear the content above it. There is a coupling or dependency created.

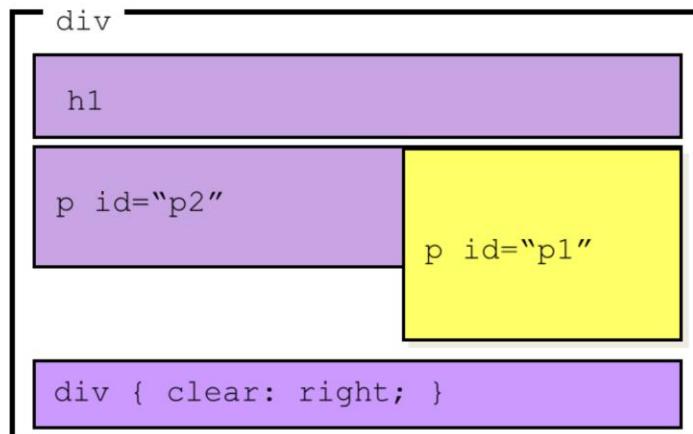
## Tactic #2: Clearing Floats

- Force the parent to enclose the float;  
"Float nearly everything"



This approach leads to the task of needing to float everything.

## Tactic #3: Self-clearing The Non-Semantic Way



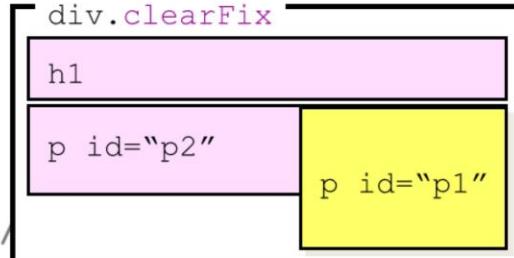
This approach introduces a non-semantic element, usually a div placed after the floated content which then clears. This works nicely, but the div serves no purpose but to aid in presentation.

## Tactic #4 : The Best Technique

- The clearfix technique

```
.clearfix:after {
 content: "";
 display: block;
 clear: both;
}

.clearfix {
 zoom: 1; /* IE hack */
}
```



What is this hack with IE?

Read on...

# clearfix Now Added

- clearfix has been added to the containing element now



Header

Col1

Col3

Col2

Footer

HTML code:

```
#container:after {
 content: '';
 display: none;
 clear: both;
}

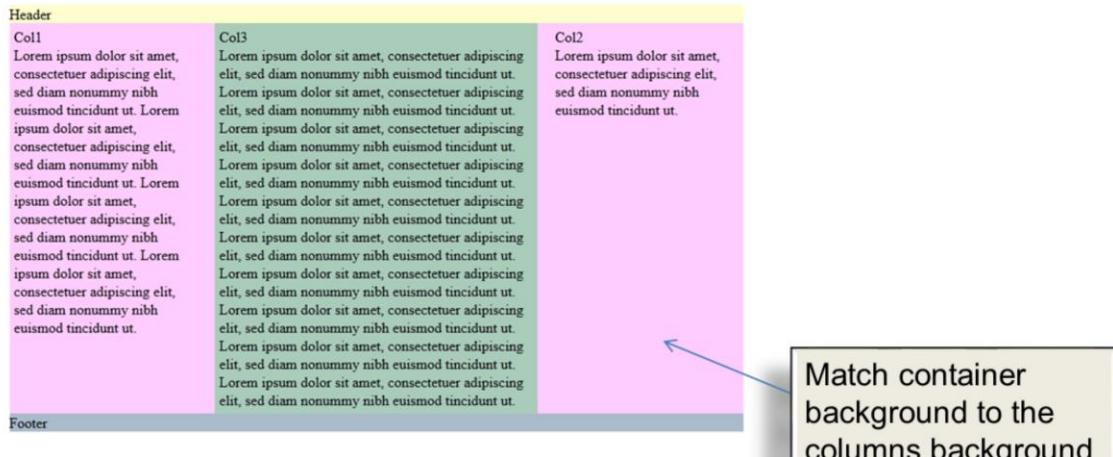
#container { zoom: 1; }
```

Detailed description: This block contains a screenshot of a web page layout. At the top is a yellow header bar labeled 'Header'. Below it are three columns: 'Col1' (purple), 'Col3' (light blue), and 'Col2' (pink). Each column contains placeholder text ('Lorem ipsum...'). A grey footer bar at the bottom is labeled 'Footer'. To the right of the screenshot is the CSS code for the clearfix technique.

See examples in examples/css/floating3.html

## But My Columns Don't Have Same Height!

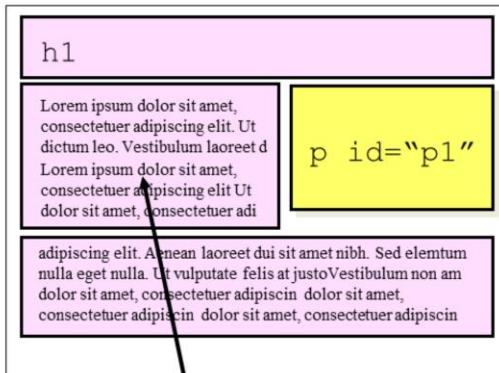
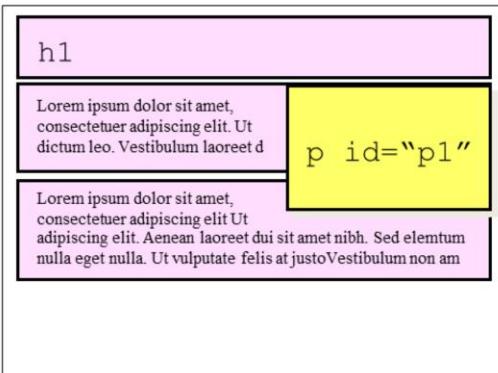
- They shouldn't
  - Floated Columns are "shrink-to-fit"
  - A little CSS trickery can fix this however:



See examples in examples/css/floating4.html

For this trick to work, the middle column should be the longest one. But generally this requirement is not a problem.

# Why zoom: 1 for IE?



"hasLayout" answers many of the reasons for IE rendering bugs

In IE, a block element with "hasLayout" will not run behind the floated element as the spec dictates. Instead, the entire block will run alongside the float.

hasLayout is an important feature to understand how IE behaves when content renders. For more on this hidden property, read this very well written article:

<http://www.satzansatz.de/cssd/onhavinglayout.html>

hasLayout is used to either "trigger" layout if set or NOT trigger layout if NOT set. hasLayout, however, is a readonly property and therefore it cannot be directly set. Instead, a number of properties can be used to "trigger" hasLayout in IE. (refer to the above cited article for these). This also explains why zoom:1 on the previous page happens to work for the clearfix on IE.

## Lab 04 – Creating a Multi-Column Layout

- Use CSS floating and clearing to create a 3-column layout out of the current index.html solution
- Follow the instructions at the end of the manual





A photograph of a person wearing a black headset with a microphone, looking down at a computer monitor. The monitor displays a dark interface with some blurred text or graphics. The background is a soft-focus blue and white.

# HTML5 CSS3 Overview

# CSS 3

CSS Key Features

CSS Box Model

CSS3 Overview

New Features

# Overview

- selectors
- colors
- rounded corners
- transitions and animations
- background enhancements
- shadows and reflections
- data URLs
- border images
- transforms
- media queries
- columns
- @font-face

## The CSS 3 Standard

- CSS 3 standard managed by W3C is so large it has been broken into 30+ modules
- The **selectors** module has been given the highest priority and is a *Proposed Recommendation*
- Most other modules are in working draft status and are subject to change, including:
  - **animations, transformations, transitions**
- Parts of the standard have been around for years

Good reference on CSS3 standards status: <http://www.css3.info/modules/>

# CSS 3 Feature X-Browser Support Table

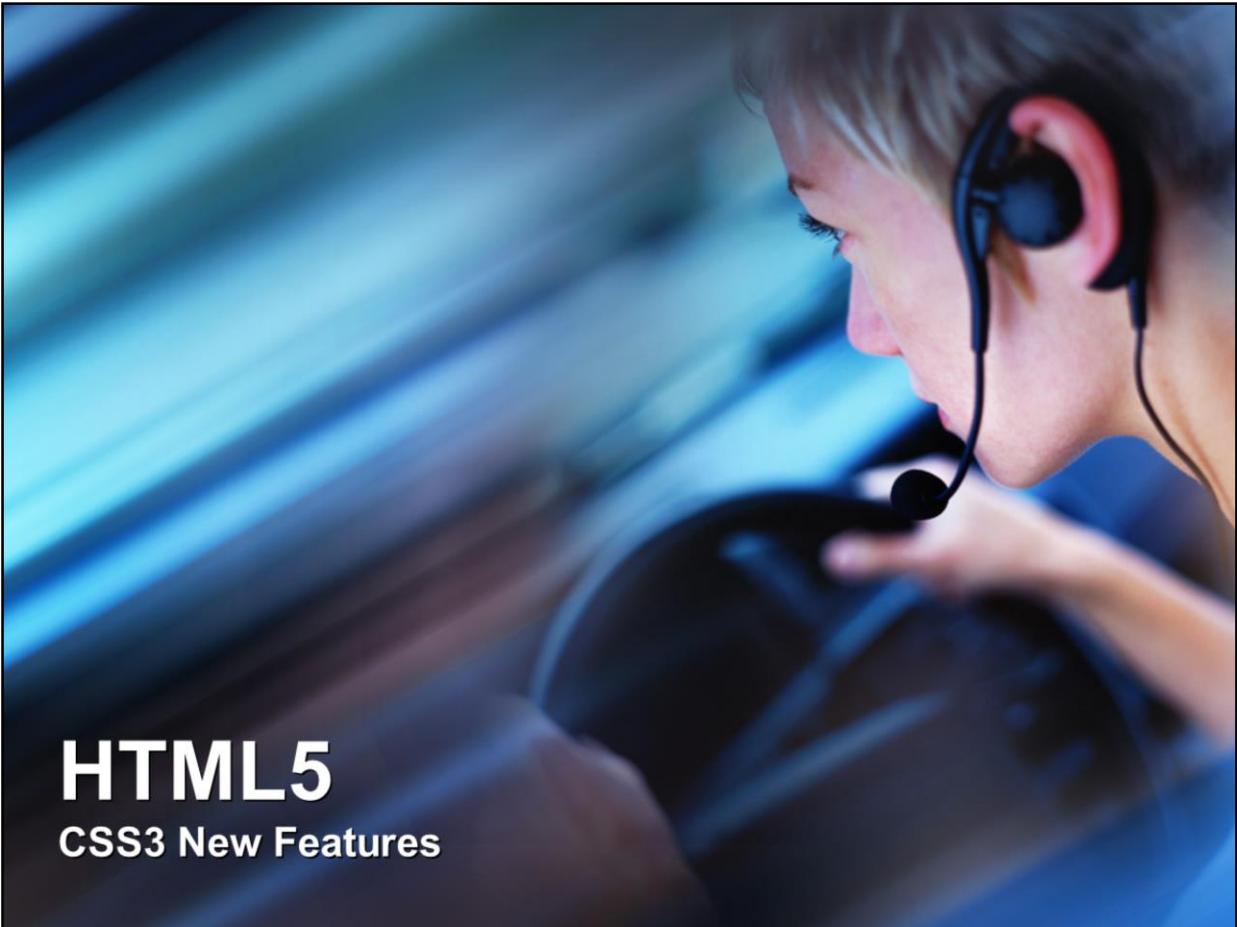
Feature	IE 6	IE 7	IE8	IE9	FF 3	FF 3.5	FF 3.6	FF 4+	SF 4	SF 5+	C h	Op 10	Op 10. 5
Hover Effects	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Rounded Corners	N	N	N	Y	Y	Y	Y	Y	N	Y	Y	N	Y
Drop Shadows	N	N	Y	Y	N	Y	Y	Y	Y	Y	Y	N	Y
Gradients	N	N	Y	Y	N	N	Y	Y	Y	Y	Y	N	N
Transparency (RGBA)	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Transparency (Opacity)	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Transitions	N	N	N	N	N	N	N	N	Y	Y	Y	Y	Y
2D Transforms	N	N	N	Y	N	Y	Y	Y	Y	Y	Y	N	Y
Data URIs	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Note that using filters for functionality in IE 6/7 is not recommended due to memory issues.

IE9 and earlier do not support 3D transforms, border-images, animations, transitions, multiple column layouts, and text-shadow.

# CSS Performance Improvements

- Fewer images (due to rounded-corner/gradient CSS) require fewer network requests
- Images may be encoded into the stylesheet requests
- Use of gradients via CSS statements is more efficient than use of images
- Transitions, animations, transformations accomplished by browser native code rather than via JavaScript



A photograph of a person wearing a black headset with a microphone, looking down at a computer monitor. The monitor displays a dark, abstract background with blue and white streaks. The person is wearing a blue shirt.

# HTML5

## CSS3 New Features

# CSS 3

CSS Key Features

CSS Box Model

CSS3 Overview

New Features

## Improved Selector API

- Direct child
- Direct adjacent siblings
- Indirect adjacent siblings
- Attribute selectors
- :first-child, :last-child
- :first-of-type, :last-of-type
- New form validation selectors
- :nth-child
- ::selection
- :target

The improved selector API includes many new ways of selecting DOM nodes within a page. These new selectors may be used in CSS code, JavaScript native APIs (discussed later), or even within JavaScript libraries such as jQuery, YUI, Dojo, and others.

## Direct Child

- Selects a specified direct child of a parent
  - Doesn't work in IE6

```
p > em { text-decoration: underline; }
```

```
<body>
```

target this

```
<p>This is a great paragraph</p>
```

```
<p>This is not a great
paragraph</p>
```

not this

# Direct Adjacent Siblings

- Selects sibling immediately following a specified node

```
h2 + h3 { margin-top: 10px; }
```

```
<div>
 <h2>Economy Recovers!</h2>
 <h3>Fed is Optimistic</h3>
 <p>Lorem ipsum dolor...</p>
</div>
```

target this

```
<div>
 <h2>Economy Recovers!</h2>
 <p>Lorem ipsum dolor...</p>
 <h3>Fed is Optimistic</h3>
</div>
```

not this

This too doesn't work in IE6.

# Indirect Adjacent Siblings

- Selects a sibling of a specified node

```
h2 ~ h3 { margin-top: 10px; }
```

```
<body>

<div>
 <h2>Economy Recovers!</h2>
 <p>Lorem ipsum dolor...</p>
 <h3>Fed is optimistic</h3> ← target's this
</div>

<h3>Stocks Rebound overnight</h3> ← not this
```

Once again, this won't work for IE6.

# Attribute Selectors

- Selects elements based on presence or values of attributes

```
img[title] {border: 2px solid #000;}
```

Styles any <img> with a title attribute.

```
img[title="Figure"] {border: 2px solid #000;}
```

Styles any <img> with the title attribute "Figure", symbol = signifies an exact match.

```
img[title~="Figure"] {border: 2px solid #000;}
```

Note the additional “~”. Styles any <img> with a title attribute that includes "Figure" in a space-separated list of words.

```
*[lang|="en"] {color: #CCC;}
```

Styles any element with a lang attribute that begins with "en" in a hyphen-separated list.

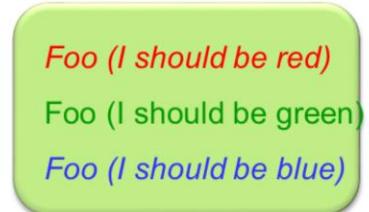
This too doesn't work in IE6.

## :first-child, :last-child, nth-child

- Styles either the first, last child, or nth-child of an element

```
<div>
 <p>Foo (I should be red)</p>
 <p>Foo (I should be green)</p>
 <p>Foo (I should be blue)</p>
</div>

p:first-child {color: red;}
p:last-child {color: blue;}
p:nth-child(2) {color: green;}
p:nth-child('odd') { font-style: italic; }
```



Foo (I should be red)  
Foo (I should be green)  
Foo (I should be blue)

Foo (I should be blue)

- For browsers not supporting these, the fallback is to attach class names manually to HTML elements

Some of the selectors presented here are actually from CSS2.1.

## :first-of-type, :last-of-type

- Selects the first (or last) of a specified type
- Equivalent to nth-of-type(1) or nth-last-of-type(1)

```
<div>
 <div>(span) First child</div>
 <p>(p) First p (I should be red)</p>
 <p>(p) Last p (I should be blue)</p>
 <div>(span) Last child</div>
</div>

p:first-of-type { color: red; }
p:last-of-type { color: blue; }
```

(span) First child  
(p) First p (I should be red)  
(p) Last p (I should be blue)  
(span) Last child

basic class (uses)

## input:valid input:invalid

- `input:valid` and `input:invalid` style elements that either do or do not meet validation criteria (e.g., required, pattern, or another constraint)



```
input:invalid {
 border-color: red;
}
input:valid {
 border-color: green;
}
```

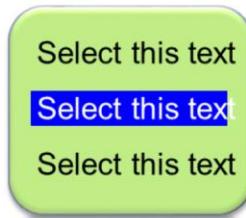
```
<input required pattern="\d{3,5}" name="zipcode"
value="" title="A valid local zipcode"/>
```

When an invalid value is specified (according to the pattern attribute described in the input element), the CSS styles that use the `:invalid` pseudo-class will be applied. Similarly, when the values are valid, the `:valid` pseudo-class is rendered.

## ::selection

- Styles the content selected by the user

```
<p id="webkit">Select this text</p>
<p id="moz">Select this text</p>
<p id="normal">Select this text</p>
```



```
#webkit::-webkit-selection { background-color: red; }
#moz::-moz-selection { background-color: blue; }
#normal::selection { background-color: yellow; }
```

The ::selection pseudo-element can determine how to style content. Currently this implementation is browser vendor specific and requires the specific browser prefixes.

## :target

- Styles the element with the id equivalent to the hash in the address of the browser

```
<p>skip to section 1</p>
<p>skip to section 2</p>
<p>skip to section 3</p>
<p>skip to section 4</p>
<p>skip to section 5</p>
```

```
<p id="sec1">Section 1</p>
<p id="sec2">Section 2</p>
<p id="sec3">Section 3</p>
<p id="sec4">Section 4</p>
<p id="sec5">Section 5</p>
```

click, styles this!

```
:target { background-color: black; color: white; }
```

## Lab 05 – Using Selectors

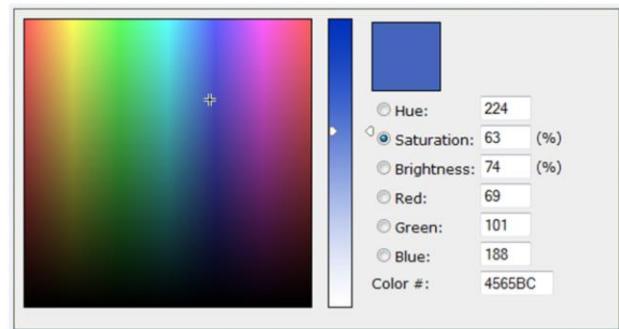
- Open *news.html* in the Lab05 starter folder
- At the bottom is a set of Y.one() statements that require various CSS selectors
- Insert the appropriate selector for each item
- Example:

```
// select the 'More Stories' <h4> element using
// 'Direct Child' selector notation
ans: #more-stories > header > h4
```

Note: this exercise does not require knowledge of YUI or JavaScript. It merely allows you to test some of the selectors previously discussed. Also note that although there are "better" selector choices, these exercises are designed to give you some exposure to the new selectors discussed in this chapter.

# New CSS Color Options

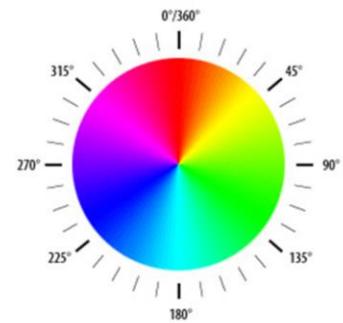
- CSS3 provides additional ways of specifying colors
  - Classic approaches (hex values, RGB percentages/intensities still valid)
  - Now may specify:
    - RGBA
    - HSL (Hue/Saturation/ Lightness)
    - HSLA



<http://www.w3.org/TR/css3-color/>

# How HSL Works

- HSL (Hue / Saturation / Lightness) works as follows:
  - Hue is defined as a degree on a color wheel
    - 0 (360) is red
    - 240 is blue
    - 120 is green
  - Saturation is a percentage of the full color (0% – 100%)
  - Lightness (0% dark – 100% white)



# Alpha Transparency

- Both RGB and HSL provide for a fourth alpha transparency value
  - 0 is transparent
  - 1 is opaque
- RGBA and does not have an equivalent hexadecimal color notation

# New Color Examples

```
div {
 color: rgb(0,0,0);
 background-color: rgb(255,255,255);
}
div {
 color: rgba(0,0,0,0.6); /* semitransparent black */
 background-color: rgba(255,255,255,0.6); /* semitrans. wht */
}

div {
 color: hsl(0,0%,0%); /* black */
 background-color: hsl(0,0%,100%); /* white */
}

div {
 color: hsla(0,0%,0%,0.6); /* semitransparent white */
 background-color: hsla(0,0%,100%,0.6); /* semitrans black */
}
```

## New Color Improvements

- Though IE6/7 support transparency, they also have memory issues supporting it
- Use a solid background color or -ms-filter for IE8 as a fallback:

```
selector {
 background-color: #f00; /*fallback */
 background-color: rgba(255,0,0, 0.5);
}

selector {
 opacity: 0.5;
 -ms-filter:
 "progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";
}
```

Note that the -ms-filter opacity (and the standard opacity) are slightly different in meaning than the background-color alpha transparency value. The later will only make the background-color transparent, while the -ms-filter (or opacity) option will make the entire element transparent including foreground text color, border, padding, etc.

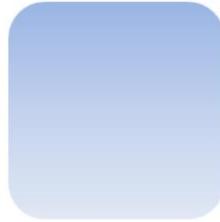
# Graceful Degradation Property Order

- Browsers will drop the remaining properties in a rule if they do not recognize one
  - Place more generalized property first to capture all browsers
  - Place the more specialized property last

```
selector {
 color: #f00; /* fallback */
 color: rgba(255, 0, 0, 0.5);
}
```

# Rounded Corners

- Rounded corners can be achieved using CSS only
  - No longer requires use of images or markup
  - IE6-8 would see non-rounded (square) corners



```
div {
 -moz-border-radius: 5px; /* FF3.x */
 -webkit-border-radius: 5px; /* SF3-4 */
 border-radius: 5px; /* Opera 10.5, IE 9,
 SF5, Chrome, FF4 */
}
```

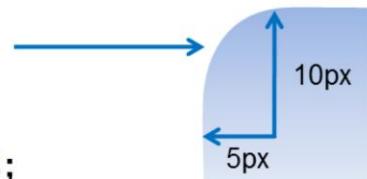
std. version should come last

<http://www.w3.org/TR/css3-background/>

# Rounded Corners Syntax

- Specific properties can define each corner

```
border-top-left-radius: 5px 10px;
border-top-right-radius: 10px;
border-bottom-left-radius: 10px;
border-bottom-right-radius: 10% 5%;
```



- Shorthand syntax:

```
border-radius: [<length> | <percentage>]{1,2,4}
[/ [<length> | <percentage>]{1,2,4}]?

border-radius: 5px 10px 5px 10px / 10px 5px 10px 5px;
border-radius: 5px;
border-radius: 5px 10px / 10px;
```

Note that in FF3.6, it uses a different syntax for top-left, top-right, bottom-left, and bottom-right:

- moz-border-radius-topright
- moz-border-radius-topleft
- moz-border-radius-bottomright
- moz-border-radius-bottomleft

This has been corrected in FF4.0.

There are several ways to specify rounded-corner values. Either as one or two values for each corner or via the use of shorthands (shown at the bottom). Use the / operator to separate the x-values from the y-values.

# Gradients

- Gradients use the *background-image* property
- For non-supporting browsers:
  - Use solid background color, or
  - Use 1px repeating image for IE6/7
- Opera supports SVG-based gradients
- IE6-9 support gradients (-filter, -ms-filter)

Neither are recommended

Example of progressive enhancement:

View [finance.yahoo.com](http://finance.yahoo.com) in Firefox and MS IE6-9. Compare the page headers.

Gradients receive various levels of support in browsers. Because of this a determination has to be made as to the need for gradients within non-supporting browsers. If the gradient is important to the design of the UI or presentation, then an appropriate solution must be achieved for all browsers. If it is not important, an option to simply degrade to a solid color background can be made.

Repeating images can still be used.

# Gradients Example



```
.com-gradient {
background-color: #b82cb8;
background-image: -webkit-gradient(linear, left top, left bottom, from(#b82cb8), to(#2b29ba));
background-image: -webkit-linear-gradient(top, #b82cb8, #2b29ba);
background-image: -moz-linear-gradient(top, #b82cb8, #2b29ba);
background-image: -o-linear-gradient(top, #b82cb8, #2b29ba);
background-image: linear-gradient(to bottom, #b82cb8, #2b29ba); /* Ch26+, FF16+, IE10+ */
}
```

Gradient has been simplified recently, see discussion on the next slide!

Our rule of thumb continues, specify the generic "all-supported" value first, in this case the standard color value. Then "enhance it" with the browser-specific extensions, and finally specify the standards-based version last.

# Why CSS3Please?

```

/*
CSS3, Please! The Cross-Browser CSS3 Rule Generator

You can edit the underlined values in this css file,
but don't worry about making sure the corresponding
values match, that's all done automagically for you.

Whenever you want, you can copy the whole or part of
this page and paste it into your own stylesheet.
----- */

/* [to clipboard] [toggle rule off] */
.box_round {
 -webkit-border-radius: 12px; /* Saf3-4, iOS 1-3.2, Android 3.1-6 */
 border-radius: 12px; /* Opera 10.5, IE9, Saf5, Chrome, FF4+, 10 */
}

/* useful if you don't want a bg color from leaking outside the border; */ /* [toggle styling] */
-moz-background-clip: padding; -webkit-background-clip: padding-box; background-clip: padding-box;
}

/* [to clipboard] [toggle rule off] */
.box_shadow {
 -webkit-box-shadow: 0px 0px 4px 0px #fffff; /* Saf3-4, 10 */
 box-shadow: 0px 0px 4px 0px #fffff; /* Opera 10 */
}

```

## CSS3, please!

This element will receive instant changes as you edit the CSS rules on the left.

Enjoy!

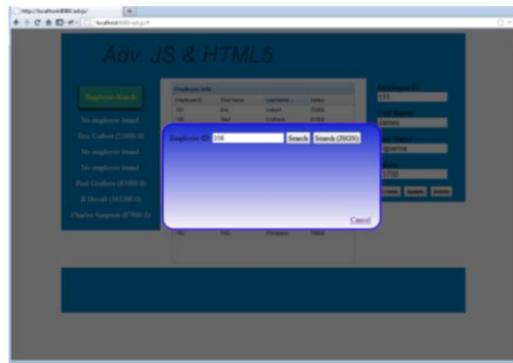
### Provides:

- 1- Interactive interface
- 2- Rules are in proper fallback order
- 3- Supported Browsers are listed
- 4- Easy copy-and-paste

CSS3Please.com provides a good way to remember the proper order of elements in CSS. The page shows the supported browser version and the correct order in which the rules need to be placed.

## Lab 06 – CSS3 Search Popup

- Use CSS3 gradients and rounded corners in conjunction with CSS3Please and jQuery to create a dialog box that popups in the center of the screen



- Follow the instructions at the end of the manual

# Multiple Background Images

- CSS3 allows a single element to support multiple background images

```
div.article {
 background-image: url(kitten1.jpg), url(kitten2.jpg);
 background-position: top left, top right;
 background-repeat: no-repeat, no-repeat;
}
```

shorthand  
↓



```
div.article {
 background: url(kitten1.jpg) top left no-repeat;
 url(kitten2.jpg) top right no-repeat;
}
```

It may seem strange to add this support to CSS for this, but it actually comes in handy for solutions that implemented the sliding doors technique and required an extra (sometimes non-semantic) element in the HTML to hook an image into.

# Border Images

- Images may be used within borders
- border-image is a shorthand



```
border-image: url(border.png) 27 27 27 27 repeat repeat;
```

border-image-slice

border-image-repeat

- Other properties include:
  - border-image-source, border-image-slice, border-image-width, border-image-outset, border-image-repeat

<http://www.w3.org/TR/css3-background/#the-border-image>

The border-image property is somewhat complex. Visit the above reference for more detail on its usage.

Not supported in IE9 (or earlier), fallback: use images.

# Text and Box Shadows

- Use to create shadows on either text or box elements
  - Fallback: older browsers would show no shadow

box-shadow: offset-x offset-y blur-radius? spread-radius? color?

Text shadow!



```
.textshadow {
 text-shadow: 3px 3px 3px #888; /* FF3.5+, Opera 9+, all SF/Ch */
}

.boxshadow {
 -moz-box-shadow: 5px 5px 5px #808080; /* FF3.5+ */
 -webkit-box-shadow: 5px 5px 5px #808080; /* SF, Chrome */
 box-shadow: 5px 5px 5px #808080; /* Opera 10.5, IE9 */
}
```

Syntax is box-shadow: offset-x offset-y blur-radius? spread-radius? color?

# Transforms

- Transformations such as **skew**, **rotate**, and **scaling**, **matrix**, **translate** can be performed on elements
  - May be used in solutions as long as the element remains viewable in non-conforming browsers
- Use the standard and vendor-specific versions

```
.scale {
 left: 25px;
 -webkit-transform: scale(2.5);
 -moz-transform: scale(2.5);
 -o-transform: scale(2.5);
 -ms-transform: scale(2.5);
 transform: scale(2.5);
}
```

The transform property allows elements to be skewed, rotated, scaled, and translated. Implementations may use this technique as long as the non-transformed browser versions are fully usable without being transformed.

# Transforms

- Valid functions include:

– matrix	applies a transformation matrix
– rotate(deg)	
– scale(factor)	scales by a factor
– scaleX(factor)	
– scaleY(factor)	
– skew(angle)	skews on both axes
– skewX(angle)	
– skewY(angle)	
– translate(amount)	translates along x and y axis
– translateX(amount)	
– translateY(amount)	

The following provides a list of the transform functions available to supporting browsers.  
Be sure to indicate the proper CSS property prefix for specific browsers.

# Transitions (Hover Effects)

- Transitions allow animation-type effects using CSS
  - Typically transitions must occur using a `:hover` to trigger the effect or a class set by JavaScript
- Transition may only be used if it enhances an effect on a page, but is not central or important to the UI
  - In other words, lack of a transition should not affect the user's ability to interact with the page

<http://www.w3.org/TR/css3-transitions/>

# Transitions Example

```
.item {
 position: absolute;
 left: 25px;
 -webkit-transition: left 1s ease;
 -moz-transition: left 1s ease;
 -o-transition: left 1s ease;
 transition: left 1s ease;
}

.item:hover {
 left: 100px;
}

<div class="item" ></div>
```



The div will animate by sliding to the right when the mouse hovers over it and left when it exits

See hover.html and hover2.html in the student files example folder.

# onTransitionEnd JS Event

- This event can be captured to signal when a CSS transition has ended
  - transitionend (FF)
  - webKitTransitionEnd (Safari/Chrome)
  - oTransitionEnd (Opera)



At this stage, the onTransitionEnd event is not consistent across browsers. To safely use it, you would have to detect whether or not a browser supports it. In each of the cases above, detecting support for this event is different per browser and not existent in IE. If you wish to use this event, ensure it is used in a manner that progressively enhances the application and that an appropriate fallback exists for browsers that do not support it.

# Animations

- Animations allow for keyframe-style movements
  - Animations are currently only supported in webkit browsers

```
.myanimation {
 -webkit-animation-name: bounce;
 -webkit-animation-duration: 3s;
 -webkit-animation-iteration-count: 5;
 -webkit-animation-direction: alternate;
 position: relative;
 left: 0px;
}

@-webkit-keyframes bounce {
 from { left: 0px; }
 to { left: 400px; }
}
```

<div class="myAnimation">  
 People...  
</div>

# Using Custom Fonts

- Fonts can be downloaded as defined by CSS
  - No longer reliant upon fonts installed in all browsers
  - Incurs an additional download

```
@font-face {
 font-family: 'Tangerine';
 font-style: normal;
 font-weight: normal;
 src: local('Tangerine'),
 url('http://themes.googleusercontent.com/font?
 kit=DTPeM3IROhnkz7aYG2a9sA') format('truetype');
}

body {
 font-family: 'Tangerine', arial, serif;
}
```

Use with caution due to  
performance issues and flash  
of unstyled text (fout) issue

The technique incurs an additional download for the font, and is susceptible to the "flash of unstyled text" effect before the font is downloaded. (<http://paulirish.com/2009/fighting-the-font-face-fout/>)

# Using data-URIs

- Data-URIs allow images to be embedded into CSS to save on additional requests
  - Faster than both individual images and sprite techniques
- Use on small background images that are not critical in appearance on all browsers
  - IE8 has a 32kb size limitation

```
#demo {
 width: 16px;
 height: 14px;
 background-image:url(data:image/gif;base64,
 R0lhEAAOALMA...);
}
```

<div id="demo"></div>


Additional info:

<http://www.nczonline.net/blog/2009/10/27/data-uris-explained/>

For the demo div above, the full CSS is:

```
#demo {
 width: 16px;
 height: 14px;
 background-image:
url(data:image/gif;base64,R0lGODlhEAAOALMAOazToeHh0t
LS/7LZv/0jvb29t/f3//Ub//ge8WSLf/rhf/3kdbW1mxsbP//mf//_
/yH5BAAAAAAALAAAAAQAA4AAARe8L1Ekyky67QZ1hLnjM5UUde0E
CwLJoExKcppV0aCcGCmTIHEIUEqjgaORCMxIC6e0CcgwWw6aFjsVM
kkIr7g77ZKPJjPZqIyd7sJAgVGvEGv2xsBxqNgYPj/gAwXEQA7) ;
```

## Summary

- Using CSS3, many new presentation features become available including:

new selectors	embedded fonts & images
transitions	animations
transforms	rounded corners
multi-background imgs	border-images
data-URIs	flex-boxes
...much more...	

## Lab 7 – A Polaroid Effect

- To one of the article images on the news page, add a **box shadow**

- **Rotate** it by 9 degrees

- On the main article image, create the following effect:

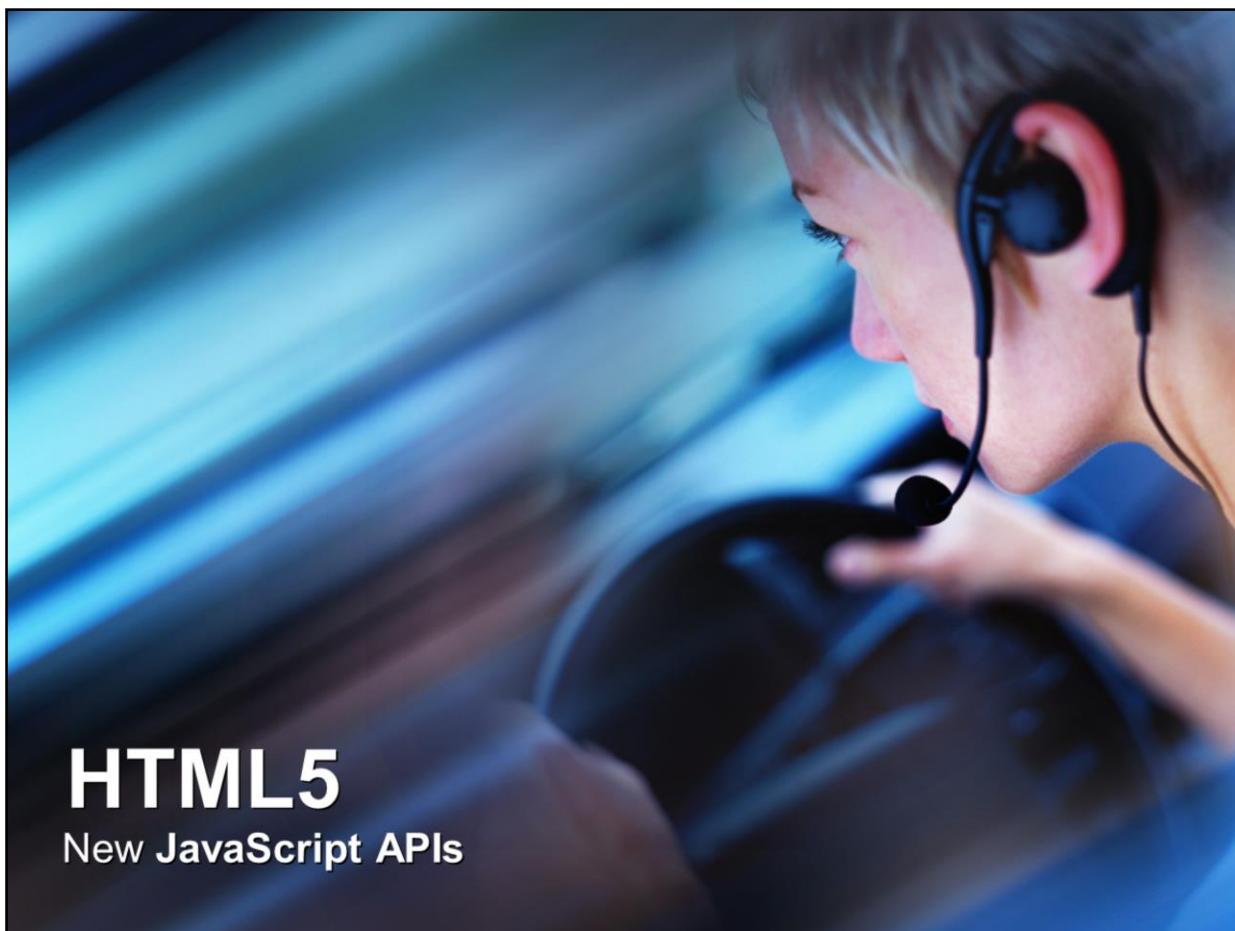


- When hovering over the image, cause it to turn upright

**Quake triggers tsunami in Indonesia**

AP - 24 minutes ago  
JAKARTA, Indonesia -  
A powerful earthquake shook Indonesia on Wednesday, killing seven people, injuring 100 and triggering a small tsunami that hit one city on the island of Sumatra, authorities said. Tsunami warnings were issued for much of the Indian Ocean region.

[Slideshow: Powerful earthquake hits Indonesia](#)  
 [Video: Tsunami warnings after strong Indonesia quake AP](#)



# JavaScript APIs Overview

- A number of additional APIs have emerged as either part of the HTML5 spec or associated with HTML5
- These additional APIs include:
  - Selector API
  - Web Storage
  - Geocoding
  - Web Workers
  - Drag-Drop
  - Web Sockets
  - History
  - File
  - Touch
  - Server-Sent
  - Cross-Origin Resource Sharing

# New JS API Support Table

Feature	IE6	IE7	IE8	IE9	FF 3.6	IE 10	FF 23	SF 6	Ch 28	Op 16
Local Storage	N	N	Y	Y	Y	Y	Y	Y	Y	Y
WebSQL Database	N	N	N	N	N	N	N	Y	Y	Y
IndexedDB	N	N	N	N	N	Y	Y	N	Y	Y
Cross-Origin Resource Sharing	N	N	Y	Y	Y	Y	Y	Y	Y	Y
Server-Sent Events	Y*	Y*	Y*	Y*	Y*	Y*	Y	Y	Y	Y
Web Sockets	N	N	N	N	N	Y	Y	Y	Y	Y
Web Workers	N	N	N	N	Y	Y	Y	Y	Y	Y
History API	Y*	Y*	Y*	Y*	Y*	Y	Y	N*	Y	Y
Geolocation API	Y*	Y*	Y*	Y	Y	Y	Y	Y	Y	Y
File API	N	N	N	N	Y	Y	Y	Y	Y	N

Y\* implies help from a JavaScript Library.

N\* implies support is buggy.

# Selecting Nodes



- W3C Selector API defines a syntax for selecting DOM nodes using CSS selectors
- Defines the methods:
  - `getElementsByClassName(classA ClassB)`
  - `querySelector()` – returns first node found
  - `querySelectorAll()` – returns Nodelist of all nodes found

<http://www.w3.org/TR/selectors-api/>

Both `querySelector()` and `querySelectorAll()` may be executed from a Node or from the document.

## Examples of Using Selectors

```
elems = document.getElementsByClassName('c1 c2');
elem = document.querySelector('ul > li:last-child');
elems = document.querySelectorAll('ul > li:nth-child(odd)');
```

While you may use these methods  
freely, most often, you will use them via  
a JavaScript Library

The CSS3 selector syntax becomes available to you when you use the YUI 2 node Selector utility. The example shown above will select all elements with the class name of 'first' that occur within the element with the id of 'panel'.

# Local Storage (Web Storage) APIs



- API for storing client-side structured data
  - Differs from cookies: Cookies more a server-side tool
    - Local storage is for client-side
- Session storage API –
  - Store data during a user's *single session*
  - Works within single tabs for lifetime of tab only
- Local storage API –
  - Store data that *persists after a browser shuts down*
  - Also works with multi-tabbed environments

```
<input type="checkbox" onchange="sessionStorage.username = checked">
```

<http://www.w3.org/TR/webstorage/>

What's the difference between local storage and cookies? Cookies are intended for server-side processing. Data stored locally is sent to the server for processing. Local storage is read and utilized on the client side.

Currently, size limitations recommended by the spec for session or local storage is approximately 5Mb.

## Storage API

- The API (both session or local) define the following methods:
  - `key(n)` - objects are ordered
  - `getItem(key)` - clone of object or null
  - `setItem(key, value)` - saves a clone under key
  - `removeItem(key)` - remove key/value or do nothing
  - `clear()` - remove all key/values
  - `length` - number of items stored

A `storage` event fires when the storage domain changes

# Modifying Local Storage

Add miscellaneous item to storage

Key  Value

```
<form id="storageForm">
 <input type="text" id="key" size="10">
 <input type="text" id="miscData">
 <input type="submit" value="Add Item">
</form>
```

```
storageForm.onsubmit = function(evt){
 if (key.value && miscData.value)
 localStorage[key.value] = miscData.value;

 displayProps();
};
```

examples/localstorage.html

# Accessing Local Storage

```
<section id="localItems"></section>

var localItems = document.getElementById('localItems');

function displayProps() {
 localItems.innerHTML = '';
 for (var prop in localStorage) {
 item = document.createElement('p');
 item.innerHTML = (prop + ' - ' + localStorage[prop]);
 localItems.appendChild(item);
 }
}

displayProps();
```

examples/localstorage.html

# Privacy

- To prevent 3<sup>rd</sup> parties from tracking user data, user-agents may:
  - delete data after a period of time
  - white/black-list sites to allow/disallow local storage
- Security: User-specific data should only be stored on the client in an encrypted format
  - Review to determine what type and how long data will remain on client

Other considerations:

Sites existing on hosts who provide multiple vendors (geocities.com, github.com, etc.) should avoid using local storage as the storage data would be shared. Storage data is organized according to hostname.

## Lab 8 – Using Local Storage

- In the Contact application, implement local storage
- Restore the last contact search completed, even after restarting the browser

# Web Database

- Use of SQLite memory-based database
  - Provides full SQL-syntax for storage and retrieval
  - Abandoned due to lack of differing implementations



Beware. This specification is no longer in active maintenance and the Web Applications Working Group does not intend to maintain it further.

<http://www.w3.org/TR/webdatabase/>

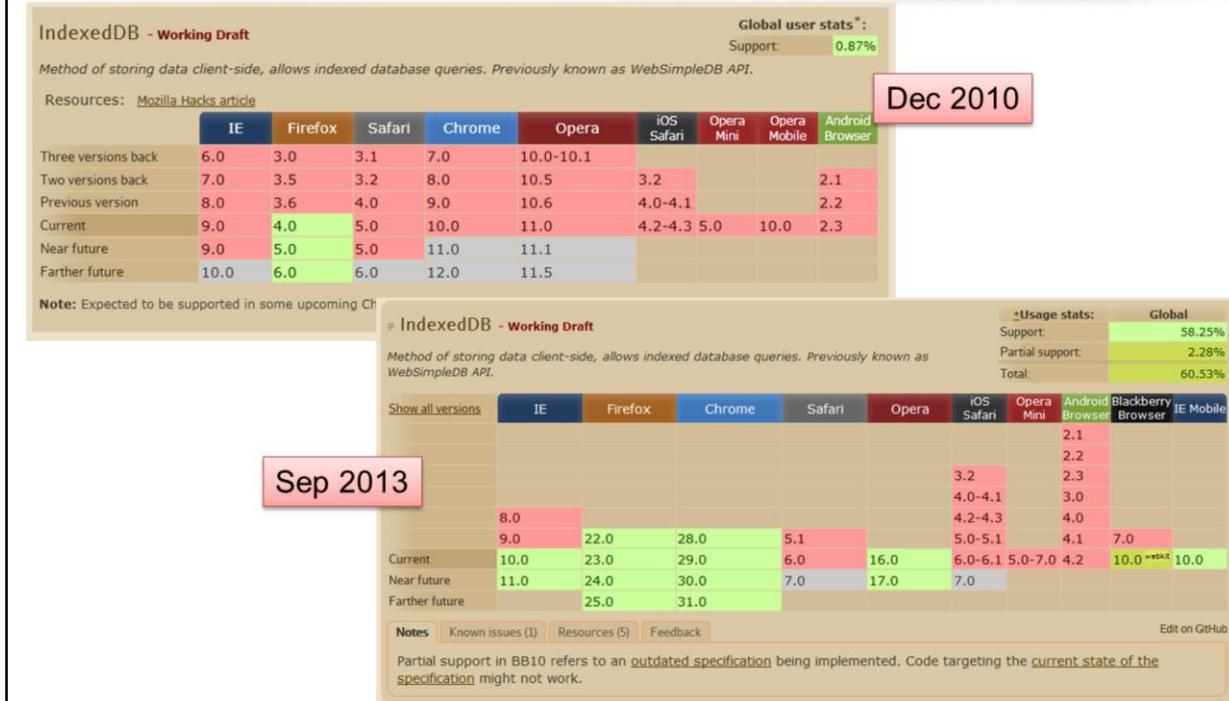
# IndexedDB

Candidate Recommendation  
July 4, 2013

- **IndexedDB** will likely replace Web SQL Database
  - Originally proposed by Oracle in 2009 (*called WebSimpleDB*)
  - Values stored as JavaScript objects
  - Indexing system facilitates data lookups (queries)
  - Defines synchronous and asynchronous APIs
    - Only asynchronous API implemented in FF4+
- Primary use case
  - store large amounts of data for offline usage
  - may also be used for caching application data

<http://www.w3.org/TR/IndexedDB/>

# IndexedDB Browser Support



Support for IndexedDB is increasing, however, it is still not supported in Safari 6.0 or any mobile browsers

## Browser Detecting IndexedDB

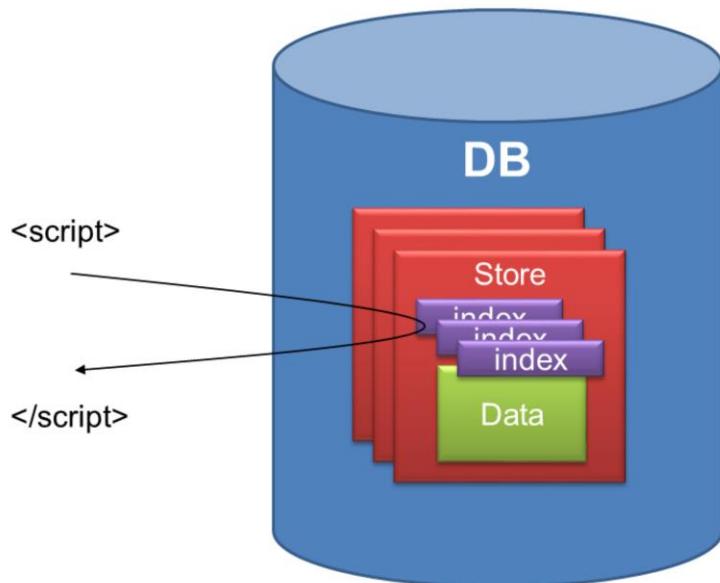
- Vendor support is improving, but still faces differences

```
var indexedDB = window.indexedDB ||
 window.mozIndexedDB ||
 window.webkitIndexedDB ||
 window.msIndexedDB;

var request = indexedDB.open('contacts', 1);
```

Because the API is still in its infancy, the vendors provide specific properties, rather than standard properties, to the database. FF4 uses mozIndexedDB, while Chrome specifies webkitIndexedDB.

# How It Works



Communication is event based

**Databases** define one or more stores

**Stores** define the properties every object must contain

Stores use key-value pairs to maintain objects

Stores contain objects (data) and may have one or more **Indexes**

## Local Storage vs. IndexedDB

- Local Storage advantages:
  - Widely supported
  - Simple API
- IndexedDB advantages:
  - better data search capabilities because of indexing

# File API

Working Draft Status  
Oct 2012

- Defines an API for user agents to manipulate files and raw data that may be uploaded
  - Create a FileReader, invoke:  
`readAsText()`,  
`readAsBinaryString()`,  
`readAsArrayBuffer()`,  
`readAsDataURL()`
  - Handle events such as `onload`,  
`onprogress`, `onloadstart`, `onabort`,  
`onerror`, `onloadend`

Drag images onto the input field  amsterdam\_cam.jpg



<http://www.w3.org/TR/FileAPI/>

See examples/file\_api.html for this example.

## File API Example (1 of 2)

```
(function() {
 function startRead() {
 var file = document.getElementById('file').files[0];
 if(file){
 var reader = new FileReader();
 reader.readAsText(readFile, "UTF-16");
 reader.onprogress = updateProgress;
 reader.onload = loaded;
 reader.onerror = errorHandler;
 }
 }

 function updateProgress(evt) {
 if (evt.lengthComputable) {
 var loaded = (evt.loaded / evt.total);
 if (loaded < 1) {
 // update the progress information (progress bar)
 }
 }
 }
})
```

## File API Example (2 of 2)

```
function loaded(evt) {
 var fileString = evt.target.result;
 // process the loaded file
}

function errorHandler(evt) {
 if(evt.target.error.code ==
 evt.target.error.NOT_READABLE_ERR) {
 // The file could not be read
 }
}
});
```

# Offline Web Applications (Application Cache)

- Defines a means for users to interact with applications even when "offline"
  - Specify assets to be cached: JS Files, Images, CSS
  - Useful for where limited connectivity is an issue
  - Create a manifest file, cache.manifest
    - All of your pages need to contain the manifest attribute and point to the correct path

// In your HTML:  
`<html manifest="/cache.manifest">`

CACHE MANIFEST  
index.html  
page.css  
news.js  
NETWORK:  
tracking.cgi

<http://dev.w3.org/html5/spec/offline.html>

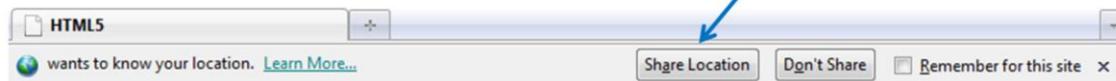
# Summary of HTML5 Data Access Capabilities

API	Description	Recommendation
<b>Local Storage (Web Storage)</b>	Cookie-style local, session storage	OK to use, IE6/7 not supported, may need workaround
<b>Web SQL Database</b>	Abandoned	Don't use
<b>IndexedDB</b>	New proposal, asynchronous operations	Too early at this stage to implement cross-browser
<b>File API</b>	Access to load files from file system	Use requires provision of implementation for older browsers
<b>Offline Storage (Application cache)</b>	Ability to specify which resources to cache locally via a manifest	Useful for browsers supporting it. Won't affect older browsers as they will ignore the manifest and will request documents in a normal fashion

# Geolocation APIs

- W3C Geolocation API defines an API that can provide a user's location
  - Candidate Recommendation status already
  - Location is determined based on IP address, wireless network ID, and/or GPS (mobile)

Most browsers prompt before allowing your location information to be acquired



<http://www.w3.org/TR/geolocation-API/>

See the examples/geolocation.html file for this example.

# Using Geolocation

- `navigator.geolocation` object
  - `getCurrentPosition(success, failure, options)`
  - `watchPosition()`
  - `clearWatch()`
- Not supported in IE 6-8

```
if(navigator.geolocation) {
 navigator.geolocation.getCurrentPosition(function(position) {
 do_something(position.coords.latitude,
 position.coords.longitude);
 }, failure, {maximumAge: 60000});
}
```

A callback

The third parameter to `getCurrentPosition()` defines a `maximumAge` indicates how long a cached position should be considered valid (specified in milliseconds). Default is zero, which means don't cache a position, but instead re-acquire one.

A timeout value may be specified as part of the options, also indicated in milliseconds.

# Position and Coordinates

- The position object defines two properties:
  - coords (Coordinates)
  - timestamp (DOMTimeStamp)
- The Coordinates object contains:
  - latitude (double)
  - longitude (double)
  - altitude
  - accuracy
  - altitude accuracy
  - heading
  - speed

# Geolocation and Google Maps v3

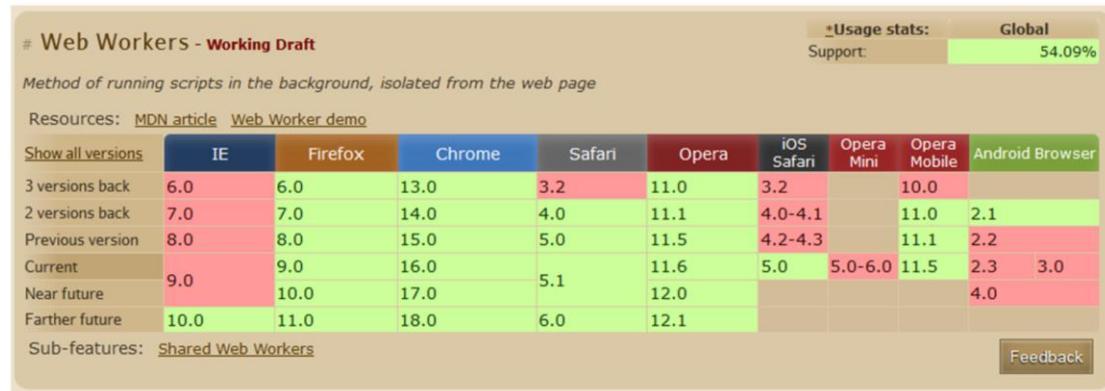
- As an example, here is the Geolocation API used with Google Maps API

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false"></script>

function success(position) {
 var lat = position.coords.latitude;
 var lon = position.coords.longitude;
 var location = new google.maps.LatLng(lat, lon);
 var options = {zoom: 12, center: location,
 mapTypeId: google.maps.MapTypeId.ROADMAP };
 var map = new google.maps.Map(document.getElementById("map"),
 options);
 map.setCenter(location);
 var marker = new google.maps.Marker({ position: location,
 map: map, title:"You are here!" });
}
```

# Web Workers

- The WHATWG Web Workers API defines a way for browsers to run scripts as background jobs
  - Living Standard Status
  - Messages are passed to-from the workers to synchronize communication



<http://www.whatwg.org/specs/web-workers/current-work/>

# Web Workers API

- Web workers can be used to perform computationally expensive tasks or network-based calls without interrupting the user interface

```
var worker = new Worker('worker.js');

worker.onmessage = function (event) {
 document.getElementById('result').textContent = event.data;
};
```

- Use **postMessage(obj)** to communicate between main page and worker...

# Web Workers Example

```
(function() {
 var webworker = new Worker('work.js');
 webworker.onmessage = function(evt) {
 document.body.innerHTML +=
 "<div>Message from worker thread: " +
 evt.data + "</div>";
 };
})();

var end=1e8,tmp=1;
postMessage('start');
while(end) {
 end -= 1;
 tmp += end;
 if (end === 5e7) {
 postMessage('halfway there, tmp is now ' +
tmp);
 }
postMessage('done');
```

work.js

# Shared Workers

- Shared Workers are a special type of Web Worker shared by all instances of a particular page
  - Ex: if the user has two tabs open with a shared worker is a single worker shared by both tabs
  - Shared Workers are preferred to avoid spawning multiple processes performing the same task



Shared workers are not enjoyed in many browsers yet.

# Client - Server Communication

- Currently 4 ways to retrieve messages from server:

- Short polling

*Repeated requests every few seconds checking for messages*

- Long polling

*Client makes request, leaves connection open until server responds, when server responds, client immediately opens new request*

- Event Streaming

*Client makes single request, connection remains open, server makes as many responses as needed when needed*

- Web Sockets

*Client makes connection, both client and server may exchange data bi-directionally*

Short polling is the classic approach of making Ajax requests, getting a response and then making a new request a few moments later.

Long polling is often called Comet and involves making a request, but the server doesn't immediately respond. So, the connection remains open for a long time (hence the name long polling), then the server responds. The client then immediately makes a new request that sits unanswered again.

Candidate Rec  
Sep 2012

# Web Sockets

- W3C Web Sockets API defines two-way communication with browser and server
- Why use sockets?
  - Avoid continuous / long polling techniques
- Uses:
  - Message-based application
  - Chat clients
  - Social, news, sports sites
  - Multiplayer games

<http://www.w3.org/TR/websockets/>

# Web Sockets

- Events include:

Event	Event Handler	Description
open	Socket.onopen	This event occurs when socket connection is established.
message	Socket.onmessage	This event occurs when client receives data from server.
error	Socket.onerror	This event occurs when there is any error in communication.
close	Socket.onclose	This event occurs when connection is closed.

- Sample:

```
var myWS = new WebSocket("ws://localhost:8007");

myWS.onopen = function(evt) { myWS.send('info'); };

myWS.onmessage = function(evt) { console.log(evt.data); };

myWS.onclose = function(evt) { ... };
```

# Server-Sent Events

- **Server-Sent Events (SSE)** is a W3C Working Draft for Comet-style interactions with the server
  - Designed to standardize how data is streamed from the server to the client
  - Useful for *streaming data* (e.g. stock quotes, etc.)

<http://dev.w3.org/html5/eventsource/>

# Server-Sent Events

## Browser Support

# Server-sent DOM events - Candidate Recommendation

*Method of continuously sending data from a server to the browser, rather than repeatedly requesting it (EventSource interface, used to fall under HTML5)*

**Usage stats:**

Support:	63.14%
Partial support:	0.06%
Total:	63.2%

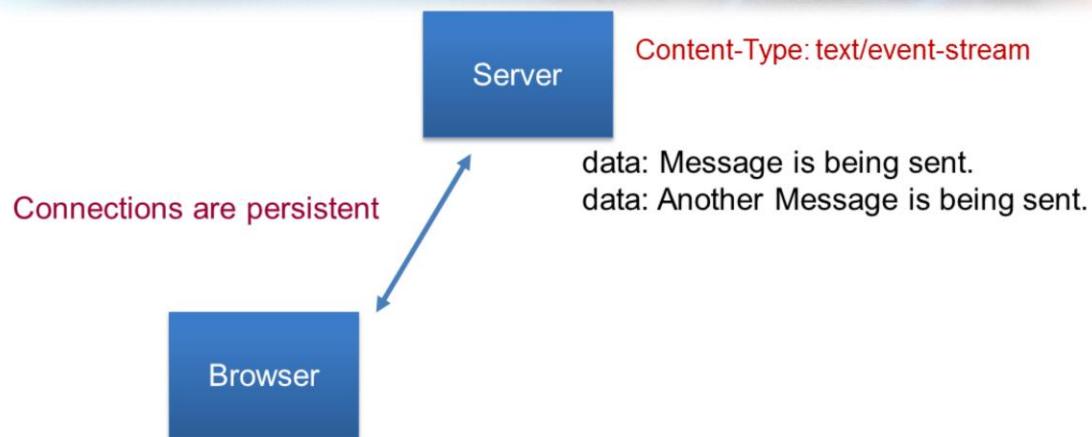
**Global**

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
										2.1
										2.2
						3.2	2.3			
						4.0-4.1	3.0			
						4.2-4.3	4.0			
8.0						5.0-5.1	4.1	7.0		
9.0	22.0	28.0	5.1		16.0	6.0-6.1	5.0-7.0	4.2	10.0	10.0
Current	10.0	23.0	29.0	6.0	16.0	6.0-6.1	5.0-7.0	4.2	10.0	10.0
Near future	11.0	24.0	30.0	7.0	17.0	7.0				
Farther future	25.0	31.0								

**Notes** Known issues (1) Resources (4) Feedback Edit on GitHub

No notes

# Server-Sent Events: How It Works



<http://yuilibrary.com/gallery/show/eventsouce>

# Cross-Origin Resource Sharing

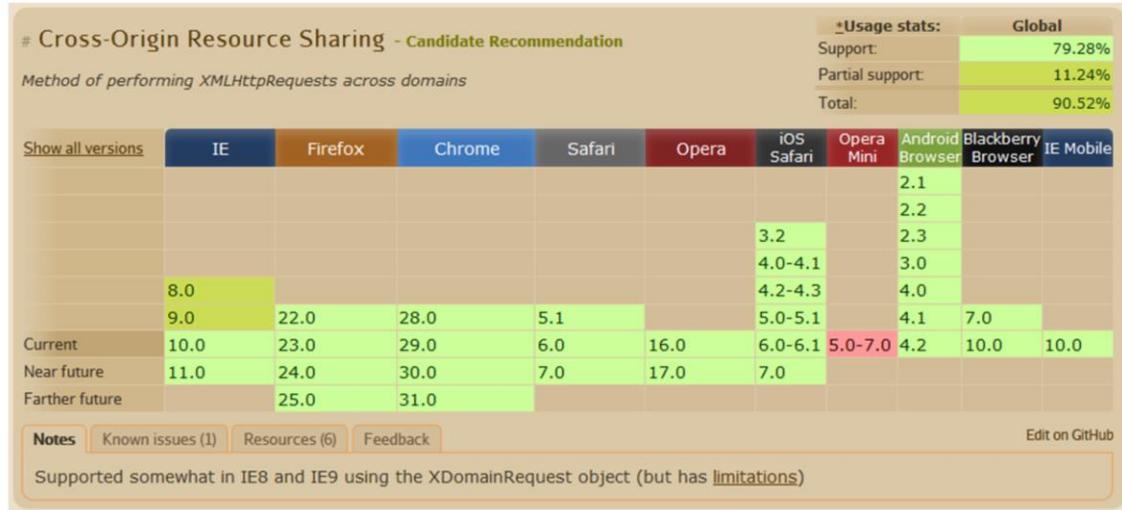
- **Cross-Origin Resource Sharing (CORS)** is a protocol which allows browsers to communicate with a server at a different domain than the original
  - Browsers restrict cross-domain Ajax requests
  - Server sends a header allowing browser to make a cross-domain request
  - CORS works in IE8 but not IE6/7

<http://www.w3.org/TR/cors/>

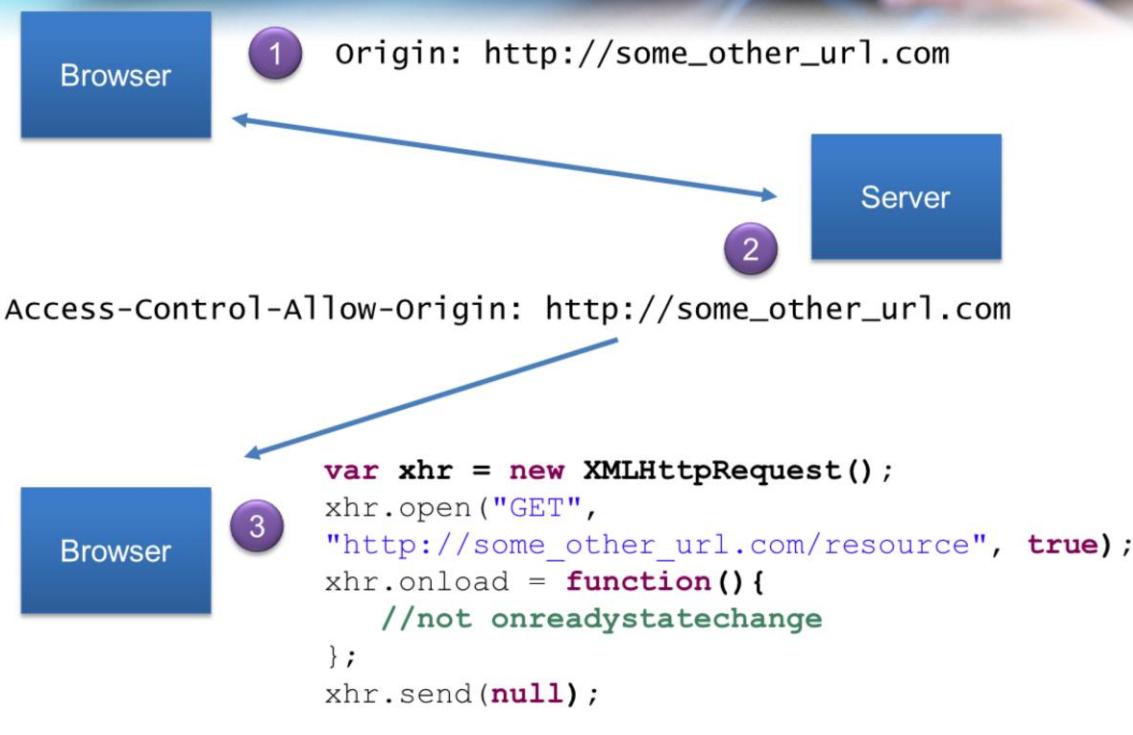
CORS supports GET and POST using basic headers, but can also support PUT, DELETE and requests using different content types (IE8 cannot do this however). To support this, requests must be "preflighted". Preflighting will entail using additional headers sent from the browser and returned by the server.

# Cross-Origin Resource Sharing

## Browser Support



# CORS – How It Works



Since, IE8 supports CORS, but via an object called XDomainRequest(), a more cross-browser friendly implementation can be written for CORS. Checking for the "withCredentials" attribute suggests the browser favors the XHR approach over the XDR approach.

```

function createCORSRequest(method, url) {
 var xhr = new XMLHttpRequest();
 if ("withCredentials" in xhr) {
 xhr.open(method, url, true);
 } else if (typeof XDomainRequest != "undefined") {
 xhr = new XDomainRequest();
 xhr.open(method, url);
 } else {
 xhr = null;
 }
 return xhr;
}
var request = createCORSRequest("get", "http://some_other_resource.com/");
if (request) {
 request.onload = function() {
 //do something with request.responseText
 };
 request.send();
}

```

## DragDrop Capabilities

- An HTML5 API that defines native event-based drag-and-drop mechanisms within the browser
  - Not yet in first draft proposal status, API may still change
- Add the `draggable` attribute to an element to allow it to be dragged
- Defines new events `ondragenter`, `ondragover`, `ondrag`, `ondragstart`, `ondragstop`, `ondragleave`, `ondragend`

<http://www.whatwg.org/specs/web-apps/current-work/multipage/dnd.html#dnd>

## Summary

- A number of new APIs will slowly become available to us over the coming year
- Our use cases and design patterns will change over time with respect to these APIs
- The newer APIs will require special consideration to keep functionality cross-browser
- The older browsers will need to leverage YUI as much as possible

## Lab 9 – CORS

- In the Contact application, implement CORS
- Afterwards, your index.html page should open and successfully work through the file system without having to be loaded through a browser...
- Follow the instructions in the back of the manual





## Course Summary

# What's the Big Deal?

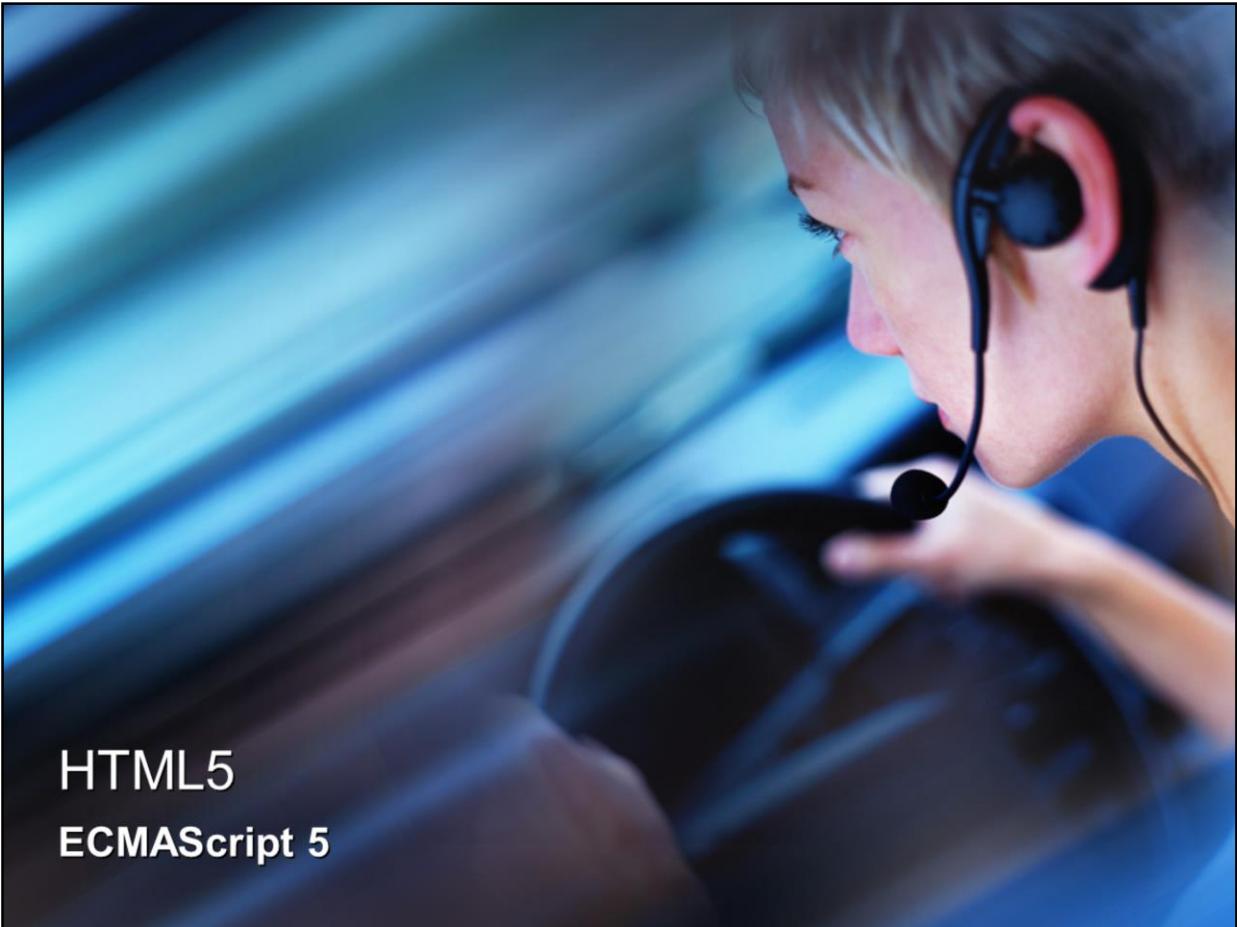
- New structural tags: section, article, header, footer, aside, nav
  - Can be used in all browsers with proper shim technique
  - New form capabilities:
    - New types: **search, email, number, url, tel, color, range**
    - New form attributes: **placeholder, autofocus, required, pattern**
  - Other new markup elements: **time, figure, details/summary, b, i, small, hr** (controversial, use sparingly)

# What's the Big Deal?

- More advanced CSS selectors (direct-child, attribute selectors, new pseudo-classes like `:invalid, :valid`)
- Transitions, Transformations, Animations
  - Very vendor specific, use as progressively enhanced feature
- HSLA/RGBA colors, prefer `rgba`, use more generic one first then the more specific one
- Gradients, rounded corners may now be used
  - Specify the vendor version (`-moz/-webkit`) first
  - Then specify the spec version

# What's the Big Deal?

- **ECMAScript 5** – it's early to use, but time to be aware of it
- **IndexedDB** – lacks a fallback for non-conforming browsers
- **LocalStorage** – can be used now, careful as IE6/7 don't work
- **Geolocation** – available in all current browsers
- **Web Workers** – tough to implement for non-conforming browsers, use for progressive enhancement
- **Web Sockets** – avoid at this time
- **History API** – Only valid in Webkit, Mozilla
- **Drag-Drop** – good cross browser support now
- **Server-Sent Events** – Works cross-browser, IE8 must use long-polling
- **CORS** – works in all latest browsers, not in IE6/7



**HTML5  
ECMAScript 5**

## What is ECMAScript 5?

- ECMAScript 5 is beginning to appear in the new round of browsers
  - ES4 proposal was too complex and required too much change to existing JavaScript code
  - ES5 is a less aggressive standard and will now be the follow-on to ES3 (some call it ES3.1)
- Maintains backward compatibility with ES3

# ES5 Browser Support

- Browser support for ES5 is largely mixed
  - IE9, Ch10, SF5, FF4 have good ES5 support

THIS BROWSER	IE 7	IE 8	IE 9	FF 3	FF 3.5, 3.6	FF 4	SF 3.2	SF 4	SF 5	WebKit	CH 5	CH 6	CH 7-10	OP 10.1	OP 10.5, 10.6, 10.7, 11	Konq 4.3	<b>BESEN</b>	Rhino 1.7
Object.create	No	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Object.defineProperty	Yes	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes
Object.defineProperties	No	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes
Object.getPrototypeOf	No	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes
Object.keys	No	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes
Object.seal	No	No	Yes	No	No	Yes	No	No	No	No	No	Yes	Yes	No	No	No	Yes	Yes
Object.freeze	No	No	Yes	No	No	Yes	No	No	No	No	No	Yes	Yes	No	No	No	Yes	Yes
Object.preventExtensions	No	No	Yes	No	No	Yes	No	No	No	No	No	Yes	Yes	No	No	No	Yes	Yes
Object.isSealed	No	No	Yes	No	No	Yes	No	No	No	No	No	Yes	Yes	No	No	No	Yes	Yes
Object.isFrozen	No	No	Yes	No	No	Yes	No	No	No	No	No	Yes	Yes	No	No	No	Yes	Yes
Object.isExtensible	No	No	Yes	No	No	Yes	No	No	No	No	No	Yes	Yes	No	No	No	Yes	Yes
Object.getOwnPropertyDescriptor	Yes	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes
Object.getOwnPropertyNames	No	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes
Date.prototype.toISOString	No	No	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes
Date.now	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Array.isArray	No	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes
JSON	Yes	No	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes
Function.prototype.bind	No	No	Yes	No	No	Yes	No	No	No	No	No	No	No	Yes	No	No	Yes	Yes
String.prototype.trim	No	No	Yes	No	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Array.prototype.indexOf	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.lastIndexOf	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.every	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.some	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.forEach	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.map	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.filter	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Array.prototype.reduce	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Array.prototype.reduceRight	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes

This chart can be found at <http://kangax.github.com/es5-compat-table/>

It provides accurate information about the support provided by the latest browsers for ECMAScript 5.

# Introducing Strict Mode

- **ECMAScript 5 strict mode** provides a stricter set of rules for coding
- Changes attempt to tighten or remove error-prone features by:
  - Changing rules for **variable assignments**
  - Adding rules for **declaring and invoking functions**
  - Changing how **eval** and **arguments** behave
  - Tightening rules to prevent JavaScript "tampering"

References:

<http://kangax.github.com/es5-compat-table/strict-mode/>

[https://developer.mozilla.org/en/JavaScript/Strict\\_mode](https://developer.mozilla.org/en/JavaScript/Strict_mode)

# Establishing Strict Mode

- Currently, establish strict mode using a string literal

```
"use strict";

(function(){
 "use strict";
 ...
})();
```

Turns on strict mode (script-wide) when used before any other statements

Uses strict mode at the functional level (place before any other statements)

Only FF4 fully supports strict mode right now

# Strict Mode: Variable Assignments

- Turns on stricter rule interpretation:

```
x = 10;
```

will cause a ReferenceError if  
x has not been declared yet

- 'with' is prevented due to its potential ambiguities

```
var foo = 3;
var myObj = { foo : 5 };

with (myObj) {
 console.log(foo);
 delete myObj.foo;
 console.log(foo);
}
```

SyntaxError: can't use 'with'  
in strict mode

Octal is also prevented as a numeric value. This is generally not used much any more and only leads to errors in coding practices.

# Strict Mode: Variable Assignments

- Can't assign duplicate object properties

```
var foo = {'hello' : 'world',
 'hello' : 'there'};
```

SyntaxError: Attempting to  
create a property twice can't  
occur in strict mode

- Assignments to non-writable properties or NaN throw an error

```
NaN = 15; // nothing happens normally
"use strict";
NaN = 15 // throws a TypeError
```

You may also not delete properties from objects that contain undeletable properties (e.g. prototype).

## Strict Mode: eval()

- eval() can only create variables in the code being eval'd, not in the surrounding namespace

```
var x = 10;

eval('use strict'; var x = 33);

console.log(x); // outputs 10
```

- eval cannot be modified

```
"use strict";
eval = 'hello'; // generates error
```

# Functions and Strict Mode

- arguments can not be redefined

```
(function(){
 "use strict";
 arguments = [] // generates error
})();
```

- argumentscallee, argumentscaller can not be used

```
"use strict";
function myFunc() {
 return argumentscallee;
};

myFunc(); // generates a TypeError
```

Simply replace the name of the function with argumentscallee. In some browsers this allowed code to potentially access arguments to other functions.

# Functions and Strict Mode

- arguments will store the original function arguments and will not change if a named parameter changes

```
"use strict";
function foo(a) {
 a = 10;
 return [arguments[0], a];
}

var results = foo(3); // should be [3, 10]
console.log(results);
```

Argument names may not be duplicated in strict mode. function f(a, a) {...}

Under normal practices, this would not be desirable, but would be allowable, so in strict mode this is attempting to catch typos.

## Strict Mode and "Tampering"

- *this* cannot be "coerced" in function calls:

```
(function(){
 console.debug(this); // window object normally
}).call(null);

"use strict";

(function(){
 console.debug(this); // this would be null
}).call(null);
```

## Using Strict Mode

- Strict Mode intentionally breaks normal coding semantics
  - Enforces better coding practices and can be "adhered to" even before browsers support it
  - Uses a non-invasive "use strict" syntax that won't hurt older browsers

# JSON

- The JSON Spec defines how objects can be converted to JSON strings and back to objects
  - Supported by all current browsers
  - If using a JavaScript library, prefer it



Most JavaScript libraries will make use of the native JSON binding capabilities if they exist, otherwise they often provide a scripted approach when not natively supported. For this reason, when using a JavaScript library, use its API for JSON conversions.

# JSON Objects

- Native support for JSON Objects:
  - `JSON.parse()`
  - `JSON.stringify()`
- Use a JS Library to support all JSON interaction
  - These use native browser support when available

```
var pStr = '{ "name" : "Rob", "age" : 25 }';
var pObj = null;
```

```
pObj = $.parseJSON(pStr);
```

jQuery

```
YUI().use('json-parse', function (Y) {
 pObj = Y.JSON.parse(pStr)
});
```

YUI

```
pObj = dojo.toJson(pStr);
```

Dojo

Using ExtJS:      `pObj = Ext.decode(pStr);`

## New Object Methods

- `Object.create(parent, donor)`
- `Object.defineProperty(obj, str, descriptor)`
- `Object.defineProperties(obj, props)`
- `Object.preventExtensions(obj)`
- `Object.keys(obj)`
- `Object.isSealed(obj)`
- `Object.freeze(obj), Object.seal(obj)`
- `Object.isFrozen(obj)`
- `Object.isExtensible(obj)`
- `Object.getOwnPropertyDescriptor(obj, str)`
- `Object.getOwnPropertyNames(obj)`
- `Object.getPrototypeOf(obj)`

# Property Descriptors

- ES5 property descriptors:
  - `writable` (true=can change, false=can't be changed)
  - `enumerable` (true=can be iterated over)
  - `configurable` (ability to change/delete attributes)
  - `value` (the value of the property)
- Together, these form the property descriptor
  - All default to true, all are optional
- Use `Object.defineProperty(obj, "prop", descriptor)` to define a property
- Use `Object.getOwnPropertyDescriptor(obj, "prop")` to work with a property's descriptor

`Object.keys(obj)` returns an array of properties within the object that may be enumerated.

# Viewing a Property's Descriptor

- Use `Object.getOwnPropertyDescriptor()`

```
var obj = { "myProp": 55 };
var descriptor =
Object.getOwnPropertyDescriptor(obj, "myProp");
console.log(JSON.stringify(descriptor));
```

```
{"value":55,"writable":true,"enumerable":true,"configurable":true}
```

This example shows how a property's descriptor can be accessed (and converted into a string in this case).

# Adding / Manipulating Descriptors

- `defineProperty()` allows properties to be added to objects with a customized descriptor

```
var obj = { "myProp": 55 };
```

```
Object.defineProperty(obj, "age", {value: 33, writable:
 false, enumerable: false,
 configurable: true});
```

```
console.log(obj.age); 33
obj.age = 44;
console.log(obj.age); 33
```

```
for (var prop in obj)
 console.log(prop + " = " + obj[prop]); myProp = 55
```

This example adds an age property to the obj object. It also disallows it from being modified.

In the example, when iterating over obj, the "age" property is not enumerated.

`Object.defineProperties(obj, props)` provides a means for adding multiple properties into an object in one call.

# Preventing Object Extension

- Use `preventExtensions()` to prevent an object from gaining new properties:

```
var obj = { "myProp": 55 };

Object.preventExtensions(obj);
obj.foo = 'hello';
Object.defineProperty(obj, "age", {value: 33,
 writable: true,
 enumerable: true,
 configurable: true
});

obj.age = 44;

for (var prop in obj)
 console.log(prop + " = " + obj[prop])
```

✖ ► Uncaught TypeError: Can't add property foo, object is not extensible

In this example, the `obj` is prevented from adding new properties.

So, when either the `foo` or `age` property are added, a `TypeError` is thrown in Chrome.

Use `Object.isExtensible()` to determine if an object may be extended.

# Object.create()

- Allows for creating objects using the new syntax of property descriptors:

```
var o = Object.create(proto, {prop: {descriptor}})
```

- proto becomes the object's prototype, second parameter become properties of the object itself

```
var o = Object.create(Object.prototype, {foo: {'value': 15 }});
var o = new Object();
o.foo = 15;
```

these are almost equivalent

```
var o = Object.create({getFoo: function(){ return this.foo; } },
 {foo: {'value': 15 } });
console.log(o.getFoo());
```

It says nearly equivalent because there is actually a subtle but important difference between the new Object() created and the o object created using Object.create(). The difference is that the former (Object.create) has all of its properties property descriptors set to false, while the new Object() object will have its property descriptors set to true each.

In the examples, the first parameter of Object.create() is an object that will become a part of the prototype. The second is a parameter that will become properties of the object instance. The property is further enhanced by a property descriptor.

Object.create() simply formalizes an inheritance pattern created by Douglas Crockford.  
(<http://javascript.crockford.com/prototypal.html>)

# Freezing and Sealing Objects

- Sealing prevents properties from being added or removed and marks descriptors as non-configurable
  - Sealing allows properties to be modified
- Freezing is similar except properties can't be edited

```
var obj = {foo: 'hello'};
Object.freeze(obj);
obj.foo = 'goodbye';
console.log(obj.foo); hello

var desc = Object.getOwnPropertyDescriptor(obj, 'foo');
desc.writable = true;
desc.value = 15;
Object.defineProperty(obj, 'foo', { value: 33 });

obj.foo = 44;
console.log(obj.foo);
```

Generates a TypeError  
because defineProperty  
cannot redefine a property

## New Array Methods

- `Array.prototype.indexOf`
- `Array.prototype.lastIndexOf`
- `Array.prototype.forEach`
- `Array.prototype.every`
- `Array.prototype.some`
- `Array.prototype.map`
- `Array.prototype.filter`
- `Array.prototype.reduce`
- `Array.prototype.reduceRight`
- `Array.isArray`

Methods related  
to functional  
programming  
style

A number of new methods have been added to the Array's prototype including a helpful `forEach()` and method to supplement the functional programming style.

# Some Array Method Examples

- **forEach**

```
var ar = [5, 9, 12];

if (Array.prototype.forEach) {
 ar.forEach(function(val, idx, array) {
 console.log(idx + " - " + val*val)
 });
}
```

- **Some**

```
var ar = [77, 58, 91];
if (Array.prototype.some) {
 if(ar.some(function(val, idx, array){
 return (val - 60) < 0 }))
 console.log('failed at least one exam!');
}
```

Most browsers won't support the newest Array methods. In all cases, these methods can be avoided by writing fully implemented code equivalents. Here, the arrays are iterated over automatically. The functions are executed for each element in the array. In the case of **some**, it will return true if just one execution of the supplied function returns true for any element in the array.

# Other New Methods and Features

- String.prototype.trim()
- Date.prototype.toJSON()
- Date.prototype.toISOString()
- Function.prototype.bind(thisObj)
  - binds a function to thisObj

```
var x = 5;
var obj = {
 x : 10,
 show: function() {
 console.log(this.x);
 }
};

obj.show(); // 10
var f = obj.show;
f(); // 5
f = obj.show.bind(obj);
f(); // 10
```

## Apdx A – Exploring ECMAScript 5

- Use `Object.create()` to create an object called `myCar` that contains `year`, `make`, and `model` properties
  - The `Car` prototype should contain a `start()` and `stop()` method
  - Make the `year` property  
*not writable*
- Attempt to change the `year` property
- Freeze the object, change the `make`, and view the results







## **Appendix B:** **Responsive Design** **and CSS Preprocessing**

## Overview

- What is Responsive Design?
- Design Considerations
- Achieving Device Independence
- Localization and Internationalization
- CSS Pre-processing Frameworks

# What is Responsive Design?

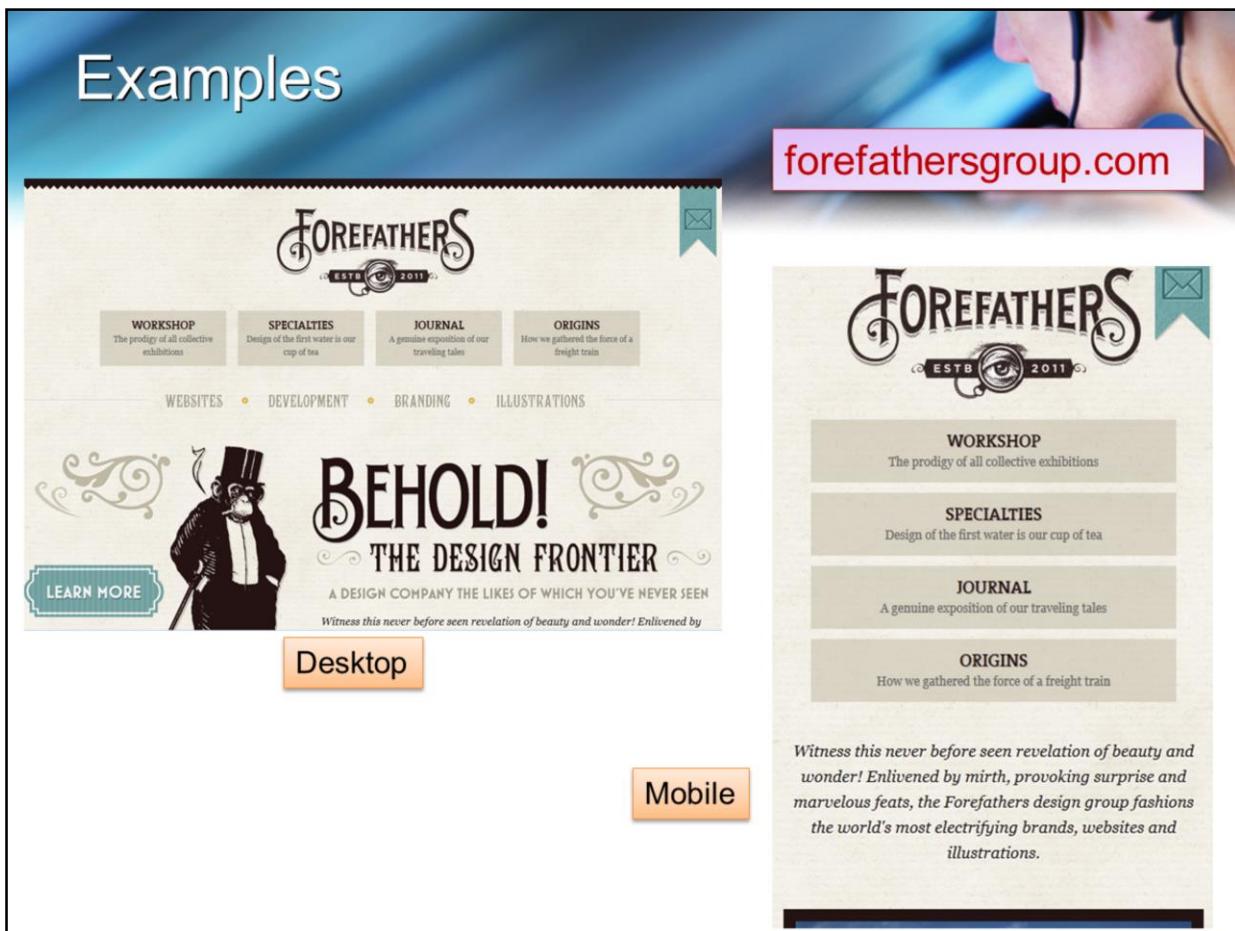


**Responsive design** is the concept of developing web applications such that they are laid out according to the user's screen width and height

# Responsive Design Considerations

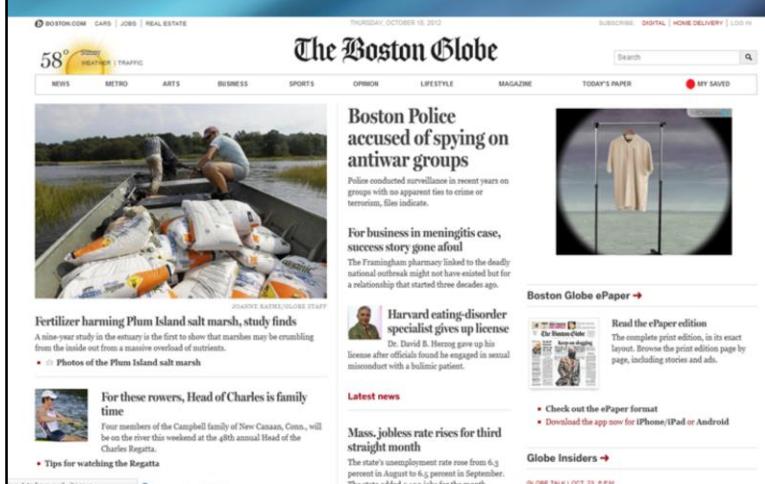
- There are a number of issues that must be addressed when considering responsive solutions:
  - **Tables** – fine on desktops, too wide on mobile devices
  - **Columns** – 3-columns on desktops, 1-column on mobile  
2 or 3 columns on tablets
  - **Images** – may require smaller sizes / resolution
  - **Content** – may need to be omitted on smaller devices

The 3 cornerstones to responsive solutions are: fluid grids (adjustable not fixed width columns), media queries, and flexible images (images that use CSS to resize them by specifying max-width: 100%, which causes them to fill only the size of the container).

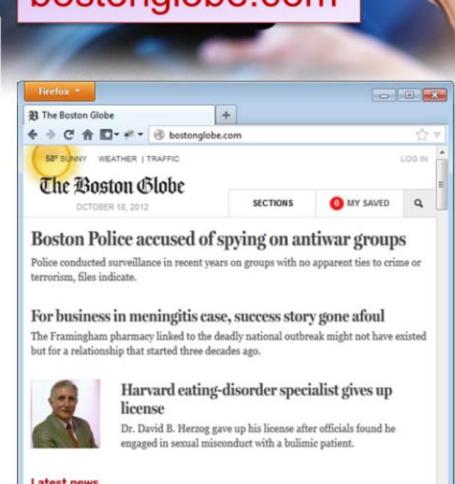


forefathersgroup removes unessential images when content is squeezed down.

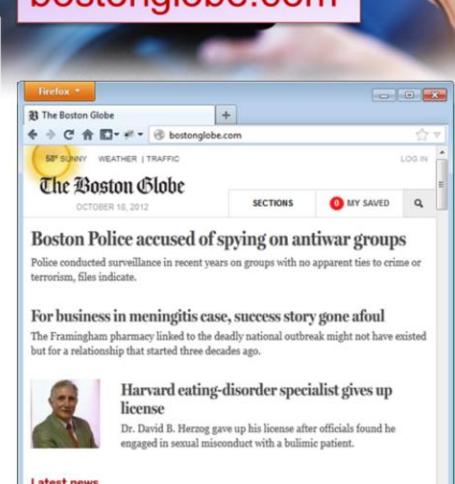
# Examples



**Desktop**



**Mobile**



bostonglobe.com

Notice the dynamic layout of columns based on screen resolution.

# Creating Responsive Solutions

```
<div id="col-1" class="col"> The 2008 Summer Olympic Games,...</div>
<div id="col-2" class="col">The Olympic games were ...</div>
<div id="col-3" class="col"> The Chinese government has ...</div>
```

```
<style type="text/css" media="screen">
 body { width: 100%; }
 div.col{ width: 33%; float: left; }
 #col-1 { background: #ffcccc; }
 #col-2 { background: #ccffcc; }
 #col-3 { background: #ccccff; }
 body.narrow div.col{ width: 100%; float: none; }
 body.medium div.col{ width: 50%; }
 body.medium div#col-2{ float: right; }
 body.medium div#col-3{ float: right }
</style>
```

Refer to examples/responsive/responsive\_design\_static\_content.html

# Creating Responsive Solutions

```
require(['dojo/query', 'dojo/on', 'dojo/window',
'dojo/dom-class'], function(query, on, win, domClass){

 var lastWidth;

 function doLayout() {
 var width = win.getBox().w;
 if (width !== lastWidth) {
 var bodyClass = 'narrow';
 if (width > 800)
 bodyClass = 'wide';
 else if (width >= 600 && width <= 800)
 bodyClass = 'medium';

 domClass.remove(query('body')[0]);
 domClass.add(query('body')[0], bodyClass);
 lastWidth = width;
 }
 }

 on(window, 'resize', doLayout);
 on(window, 'load', doLayout);
});
```

**Response.js** or **Adapt.js** are popular JavaScript libraries designed to assist in creating responsive solutions

This code simply invokes `doLayout()` on any resizing that occurs and then applies a class to the body that alters the layout.

# CSS 2.1 Media Types

- CSS 2.1 defined media types

all  
braille  
embossed  
handheld  
print  
projection  
screen  
speech  
tty  
tv

**CSS3 Media Queries**  
is the follow-on to CSS 2.1  
media types

```
<link rel="stylesheet" type="text/css" href="main.css" media="screen" />
```

```
<link rel="stylesheet" type="text/css" href="print.css" media="print" />
```

<http://www.w3.org/TR/CSS21/media.html>

# Using Media Queries

- **Media Queries** are a CSS3 feature that allow for targeting classes of devices and obtaining device characteristics

```
<link rel="stylesheet" type="text/css"
 media="screen and (max-device-width: 480px)" href="inner.css" />
```

- Queries can be applied to CSS elements directly

```
@media screen and (max-width: 400px) {
 .lastChild { padding: 5%; width: 42%; }
}
```

In the media query above, the statement checks the device's max width, if it is less than 480px, load the inner.css file. The second example applies the .lastChild class to HTML elements only if the max-width is no more than 400px (it must also be a screen-based device).

# Media Query Syntax

```
<link rel="stylesheet" type="text/css"
 media="media_type and (expression)" href="external.css" />
```

or

```
@import url(external.css) media_type and (expression);
```

```
@media media_type and (expression) {
 /* define embedded rules this way
}
```

Multiple expressions may be applied by using another 'and (expression)' after the previous one. The first and second forms above apply external css files when the media\_type and expressions are met. The third syntax above assumes rules will be embedded within the @media{} braces.

# Media Query Expressions Properties

- All of these are valid expression variables for media queries:

width	color
height	color-index
device-width	monochrome
device-height	resolution
orientation	scan
device-aspect-ratio	aspect-ratio
	grid

```
@media screen and (min-width: 180px) and (max-width: 480px)
 and (color) {
 .secondary_col { display: none; }
}
```

width, height, device-width, device-height, color, resolution all support a min- and max-prefix as well.

orientation can be 'portrait' or 'landscape'

color reports whether or not it supports color or how many bits depth color is supported.  
(e.g. @media all and (color) {....} )

resolution supports min-resolution, max-resolution. Units for this can be dpi (dots per inch)

# Media Query Device Support



Media queries are supported by all modern devices. Only legacy IE browsers will not. However, this is generally okay as IE legacy browsers are usually run within standard desktop/laptop devices which are often the default implementations that do not require a media query-based solution.

# Guidelines for Responsive Design

- Steps to consider when designing for all devices:

## 1. Determine "groups" of device widths

- Small: < 180px
- Medium: 180 – 480px
- Large: 480 – 1024px
- Desktop > 1024px

These are arbitrary, you may choose different groups. Check out <http://deviceatlas.com/resourcecentre> for specs on all devices

## 2. Define your primary target (primary group)

- Are most of your users desktop? Make this your target design size

It is generally better to design your "break" points (the points at which your media queries will alter the look and feel) based on your application and not based on popular or known devices.

# Guidelines for Responsive Design

## 3. Adapt your target solution to the other groups

- Use media queries to help define CSS rule changes
- Adapt images, headers, footers, detailed content

## 4. Use **flexible images** and **fluid grids**

- Generally widths should be specified in percentages
- Use max-widths for images

# Grid Systems

- Grid systems lend themselves nicely to responsive designs



<http://960.gs/>

The screenshot shows the homepage of the Fluid 960 Grid System. It features a large logo on the left and a navigation bar at the top with links for Fluid 12-column, Fluid 16-column, Fixed 12-column, Fixed 16-column, Download / Forum, and The 960 Grid System. Below the navigation is a section titled "Templates for Rapid Interactive Prototyping" with four columns: "DESIGN PROCESS", "DESIGN INFLUENCES", "INSPIRATION", and "CONTRIBUTION". Under "DESIGN PROCESS", there is a paragraph about the history of design. Under "DESIGN INFLUENCES", it lists Jesse Bennett-Chamberlain, Douglas Bowman, Allen Chang, Andy Clarke, Jon Hicks, Shaun Inman, Cameron Moll, Venkateshwaran Sankaranarayanan, Shee Ryan Sims, Nathan Smith, and Jeffrey Zeldman. Under "INSPIRATION", it says "I have been inspired by the work of many who have pioneered advances in Web Standards, including Jesse Bennett-Chamberlain, Douglas Bowman, Allen Chang, Andy Clarke, Jon Hicks, Shaun Inman, Cameron Moll, Venkateshwaran Sankaranarayanan, Shee Ryan Sims, Nathan Smith, and Jeffrey Zeldman. To name a few. Thank you for inspiring me to give something back." Under "CONTRIBUTION", it says "The Fluid 960 Grid System templates have been built upon the work of Nathan Smith and his 960 Grid System using effects from the MooTools and jQuery JavaScript libraries. The idea for building these templates was inspired by Andy Clarke, author of Transforming CSS, who advocated for modular and reusable web-based interactive prototyping, crediting Jason Santa Maria with the grey box method." Below this is a "16-COLUMN GRID" section with a "MOOTTOOLS FX ELEMENTS" example showing a 4x4 grid of boxes labeled One, Two, Three, and Four. At the bottom are examples for "PARAGRAPHS", "ACCORDION", and "SEARCH".

<http://www.designinfluences.com/fluid960gs/>

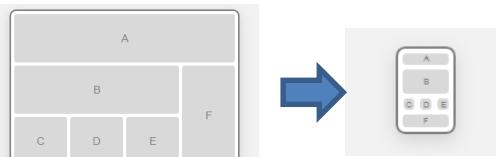
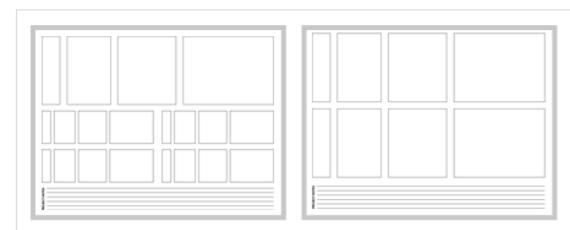
We have removed the "design" portion of the responsive design discussion here. It is assumed that a designer has already provided a desktop and narrow view.

Tools can assist in the design process, including wire framing and sketch sheets.

For more on this, see:

<http://jeremypalford.com/arch-journal/responsive-web-design-sketch-sheets>

<http://www.thismanslife.co.uk/projects/lab/responsivewireframes/>



The most popular CSS grid system is 960 GS. Once the pattern system is understood, it is fairly easy to design from.

## Additional Resources

<http://www.alistapart.com/articles/responsive-web-design/>

<http://designmodo.com/responsive-design-examples/>

<http://coding.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/>



Ethan Marcotte is sometimes credited for first discussing and coining the phrase 'Responsive Web Design'.

# CSS Pre-processing Frameworks

- Limitations in dynamic CSS have spawned new (CSS Pre-processing) frameworks to become popular
- Some of these include:
  - **SASS** – most popular, see next slides
  - **Compass** (addon supporting many pre-written CSS3 templates)
  - **LESS** – similar to SASS syntax, similar features, easy to translate between SASS and LESS
  - **Stylus** – looser CSS syntax, least popular

All of these frameworks are very similar in their syntax with only minor differences. If no preference, choose SASS.

# Syntactically Awesome Stylesheets

**Homepage:** <http://sass-lang.com>

- Requires Ruby to be installed
  - After installing Ruby, issue the command:

`gem install sass`



- SASS Stylesheets are written in 2 formats
  - SCSS – "Sassy CSS" (.scss files) newer more commonly used format
  - SASS – (.sass files) older format, uses indentation, less preferred now

SASS requires Ruby to be installed first. Afterwards, simply executing the `gem install sass` command will cause SASS to be installed.

SASS stylesheets can be written using the classic (original) format, which uses files proper indentation instead of curly braces to indicate ownership. SCSS file formats support standard CSS3 syntax.

To convert .sass files to .scss files use the conversion tool `sass-convert myfile.sass myfile.scss`

## SASS can "Watch" and Convert



- Issue the following command to have SASS automatically update your .css files when changes to .scss files occur:

```
sass --watch myfile.scss:myfile.css
```

SASS runs in the background. It actively monitors the files (directories) you specify. For example, in slide, issuing the described command will cause sass to "watch" for changes to your .scss file. When you save it, sass automatically rebuilds to CSS file.

SASS can also monitor entire directories and convert .scss files within the directory as well.

# SASS Nesting (DRY)

```
#empDetails {
 width: 80%;
 margin: 0 auto;
 font: normal 16px Arial;
}

#empDetails label, #empDetails input[type="text"] {
 display: block;
}

#empDetails input[type="text"] {
 margin-bottom: 15px;
}
```



Our final advjs app  
had the following CSS

SASS (like Ruby) encourages the DRY principle, don't repeat yourself. To this means, it uses the concept of "Nesting" rules.

## SASS Nesting (*continued*)

```
#empDetails {
 width: 80%;
 margin: 0 auto;
 font: normal 16px Arial;
 label, input[type="text"] {
 display: block;
 }
 input[type="text"] {
 margin-bottom: 15px;
 }
}
```



...the SASS way

SASS (like Ruby) encourage the DRY principle, don't repeat yourself. To this means, it uses the concept of "Nesting" rules.

# SASS Variables

```
$width: 80%;

#wrapper {
width: $width;
min-width: 800px;
margin: 10px auto;
}
```

*emp.scss*



**Sass.**

{style with attitude}



```
#wrapper {
width: 80%;
min-width: 800px;
margin: 10px auto;
}
```

*emp.css*

# SASS Mixins

```
@ mixin gradient-mixin {
background-color: #1d2fcf;
background-image: -webkit-gradient(linear, left top, left bottom, from(#1d2fcf), to(#faf5fa));
background-image: -webkit-linear-gradient(top, #1d2fcf, #faf5fa);
background-image: -moz-linear-gradient(top, #1d2fcf, #faf5fa);
background-image: -ms-linear-gradient(top, #1d2fcf, #faf5fa);
background-image: -o-linear-gradient(top, #1d2fcf, #faf5fa);
background-image: linear-gradient(to bottom, #1d2fcf, #faf5fa);
}

#empSearch {
-webkit-box-shadow: 0px 0px 4px 0px #c23cc2;
box-shadow: 0px 0px 4px 0px #c23cc2;

@include gradient-mixin;

a {
position: absolute;
bottom: 10px;
right: 20px;
}
}
```



Mixins allow for groups of properties to be "mixed in" or reused in various parts of CSS code. Declare a mixin using @mixin and then use it with an @include declaration.

An additional reference for more you can do with SASS can be found here:

<http://sass-lang.com/tutorial.html>

## Summary

- While fairly new, responsive design is a growing and accepted practice for laying out front-end solutions
- Designs are created independent of devices and rendering is commonly determined on the server-side



# Advanced JavaScript and HTML5 Exercises

# Exercise 0 – Environment and Student Files Setup



## Overview

In this exercise, you will ensure you have properly obtained, set up, and tested your student files and server.

## Objectives

At the conclusion of this exercise, you should:

Have all necessary environment elements installed and configured to properly perform the exercises in this course



## Step 1: Obtain and Install node.js

NodeJS is used to serve up content for a number of exercises during this course. You will need to download node.js from <http://nodejs.org/download>

Code in this course should successfully run on Windows, Mac, or even Linux environments simply by downloading the correct version of nodeJS.

Window users, you should download and run the .msi file:

A screenshot of the official Node.js download page. The page features the Node.js logo at the top. On the left is a sidebar with links: HOME, DOWNLOAD (which is highlighted in green), ABOUT, NPM REGISTRY, DOCS, BLOG, COMMUNITY, LOGOS, and JOBS. Below the sidebar is a social media link to @nodejs. The main content area has a dark background. It displays the text "Download the Node.js source code or a pre-built installer for your platform, and start developing today." followed by "Current version: v0.10.7". There are three download options: "Windows Installer (.msi)" (highlighted with a pink oval), "Macintosh Installer (.pkg)", and "Source Code (node-v0.10.7.tar.gz)". A table below shows the available architectures for each type of installer. The table has three columns: "Windows Installer (.msi)", "32-bit", and "64-bit".

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.exe)	32-bit	64-bit
Mac OS X Installer (.pkg)	Universal	
Mac OS X Binaries (.tar.gz)	32-bit	64-bit
Linux Binaries (.tar.gz)	32-bit	64-bit
SunOS Binaries (.tar.gz)	32-bit	64-bit
Source Code	node-v0.10.7.tar.gz	

NodeJS should be ready to run after installation.

## ***Step 2: Test Out Node***

Once installed, you can test out nodeJS. Do this by opening a command window and browsing to the location of your student files. Note: your location may vary as the student files may reside anywhere on your hard disk.

Change to your <student\_files>\labs\nodejs directory.  
Type the command:

**`node lab0_server.js`**

You should see a message that your server is running on port 8005.

Next, in your browser, type the following command:

**`localhost:8005/test_page.html`**

Send a test message and verify that it echoes.

What happens if you send <input> as a value?

## ***Step 3: Debugging Node (optional)***

Node has a built-in (simple) debugger. To use it, run the following program with the debug switch (flag). First add a **debugger;** statement to the following line in the **labs/lab0/lab0.js** file as shown:

```
var x = 10, y = 20

function f() {
 debugger; ←
 console.log(x,y);
 var x = 5;
 console.log(x,y);

}

f();

console.log(x, y);
```

**cd** to the **labs/lab0** directory and type the following:

**node debug lab0.js**

Once it starts, type 'c' to continue to your breakpoint.

Type 'repl' and then x and then y to view the x and y variables. Hit CTRL-C.

Type 'n' to advance to the next line. Type 'n' two more times and then view your variables by typing 'repl' and then 'x' and then 'y' again.

Your results should be similar to the screen shot:

```

C:\workspace_adv_js_fedex\advjs\labs\lab0>node debug lab0.js
< debugger listening on port 5858
connecting... ok
break in C:\workspace_adv_js_fedex\advjs\labs\lab0\lab0.js:1
 1 var x = 10, y = 20
 2
 3 function f() {
debug> c
break in C:\workspace_adv_js_fedex\advjs\labs\lab0\lab0.js:4
 2
 3 function f() {
 4 debugger;
 5 console.log(x,y);
 6

debug> repl
Press Ctrl + C to leave debug repl
> x
> y
20
debug> n
break in C:\workspace_adv_js_fedex\advjs\labs\lab0\lab0.js:5
 3 function f() {
 4 debugger;
 5 console.log(x,y);
 6
 7 var x = 5;
debug> n
< undefined 20
break in C:\workspace_adv_js_fedex\advjs\labs\lab0\lab0.js:7
 5 console.log(x,y);
 6
 7 var x = 5;
 8
 9 console.log(x,y);
debug> n
break in C:\workspace_adv_js_fedex\advjs\labs\lab0\lab0.js:9
 7 var x = 5;
 8
 9 console.log(x,y);
 10
 11 >
debug> repl
Press Ctrl + C to leave debug repl
> x
5
> y
20
> =

```

#### ***Step 4: (optional) Using Eclipse and Node***

Eclipse supports a plugin for nodejs called Nodeclipse. To install this plugin, visit: <http://www.nodeclipse.org/>

Once there, drag the install symbol onto the Eclipse titlebar as shown. It will install momentarily after accepting the license.

Note: on some machines, Nodeclipse fails to install. If this happens, merely start the node server



To run the debugger, open lab0.js within Eclipse (you may need to create a project and point it to your student files. Once done, you can set a breakpoint and run up to the breakpoint using the standard Eclipse debugger.

```
(function (exports, require, module, __filename, __dirname) { var x = 10, y = 20
function f() {
 console.log(x,y);
 var x = 5;
 console.log(x,y);
}
f();
console.log(x, y);
});
```

Note: you may have to set the path to your nodejs executable in the Winodw > Preferences > Nodecplise > Node path field. If you use Eclipse, try debugging lab 0 and determine what x and y are at the highlighted line shown above.

This concludes exercise 0 – setup.

# Exercise 1b – Debugging



## *Overview*

In this exercise, you will debug an application using the Chrome Browser and the built-in Chrome Developer Tools.

## *Objectives*

At the conclusion of this exercise, you should be able to:

Utilize one or more browsers for its debugging capabilities

Get exposure to Chrome Developer Tools

Follow an Ajax request

Launch Chrome from a command-line using special switch values



## ***Step 1: Open and Test the lab0\_server.js node app***

Start the node server by browsing from the command line to <student\_files>/labs/nodejs and typing  
`node lab0_server.js`.

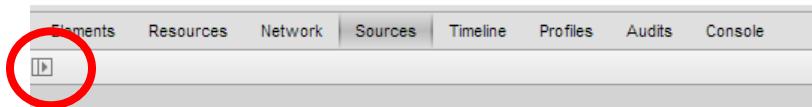
Note: Eclipse users with Nodeclipse may right-click on the lab0\_server.js file and choose Run As... > Node Application if you have been given instructions to install and set up Eclipse.

Using the Chrome browser, open a tab in your browser to `localhost:8005/test_page.html` in your browser.

Click the Submit Button. A message should be displayed in the page below. If not, your server didn't start properly.

## **Step 2: Use the Debugger**

Press F12 to bring up the Developer Tools. Select the Scripts tab (now called Sources). Then select the navigator (circled below). Choose test\_page.html.



Set a breakpoint at the first line within the submit() event callback function. (Should be line 28).

Submit the form. Your breakpoint should be hit.

## **Step 3: View Variables**

On the right-hand pane, you should now see some variables—use the **Scope Variables** pane. Examine the **this** reference. What does it currently represent?

Notice the value for testval. It should be undefined. Resume the debugger. Now type something into the form input on the HTML page. Place a second breakpoint within submit() function on line 37. This should be the \$.ajax() statement. Also, place a third breakpoint in the Ajax callback function, handleJson. (Should be line 44).

Type a message into the form this time. Type a message and click to submit the form again. After hitting your initial breakpoint, click to resume and it should hit your second breakpoint. View the change to testval. It should match what you typed into the input field.

To edit testval, double-click its value in the Scope Variables pane. Change its value—remember to put " " around the new value!

Next, run up to your 3rd breakpoint on the line:  
`$("#resultsDiv").append(...);`

Did the result change to your new value? Why, or why not?

In the right-hand side there should be a panel called 'Watch Expressions'. Expand it and click the '+' plus sign to add a variable. Type 'results'. Repeat this step adding a variable called '\$', a variable called 'options', and finally a variable called 'arguments'.

What type of variable is arguments? What is the first value within arguments?

#### ***Step 4: View Variables***

On the Network tab of the browser you should be able to see the requested URL (you can also see it listed on the console tab). View the contents of this file by selecting it (should be msg.dat) and verify it is what you see in the yellow div on the screen.

Next, switch to the Elements tab. Use the "Inspector" tool (at the bottom, looks like a magnifying glass) to select the html form in the page. Determine the "Computed Style" value for the "height" of the form. What is it?

#### ***Step 5: Try Another Browser***

While not specifically a part of this exercise, if you finish early, locate the same items within Firebug and IE Developer Tools to get a feel for these tools. Both tools are launched with F12 from within the browser.

This concludes the debugging exercise!

# Exercise 3 – HTML5 Markup and Ajax



## Overview

In this exercise, you will use the jQuery JavaScript library to communicate with the server-side component that you created earlier. This code will evolve over several additional exercises. For this exercise, you will make Ajax queries searching for contact data.

## Objectives

At the conclusion of this exercise, you should be able to:

Incorporate jQuery into solutions

Manipulate the DOM to make Ajax requests

Create an HTML 5 document from scratch

Dynamically handle form submissions and deal with default  
form actions



## Step 1: Create the Main Page

Begin by creating *index.html* within the `labs/nodejs/public` folder. Do this by using the file manager and creating the file or from within Eclipse (if you are using Eclipse) you can right-click the `labs/nodejs/public` folder and select New > File.

Name it *index.html*.

Add an HTML5 doctype.

Add an html, head, and body element.

In the `<head>` section, create a link to jquery as follows:

```
<script src="jquery.js" type="text/javascript"></script>
```

## ***Step 2: Add the Shim (Shiv)***

Add the following conditional comment to the <head> section of your HTML file.

```
<!--[if lt IE 9]>
 <script src="html5shiv.js"></script>
<![endif]-->
```

## ***Step 3: Create an HTML5 Structure and Form***

Let's create an HTML5 layout using markup. Some of this markup we won't use until later. Create a <form> nested in a <div>. Add a text input, a submit button (for XML-based requests) and a button (for JSON-based requests). Use labels semantically to describe the input field.

```
<div id="wrapper">
 <header>
 <h1>HTML5</h1>
 </header>
 <div id="content">
 <div id="primary">
 <div id="results"></div>
 </div>
 <div id="secondary">
 <div id="contactDetails"></div>
 </div>
 <section id="main">
 <div id="gridMain"></div>
 </section>
 </div>
 <footer></footer>
</div>
<div id="contactSearch">
 <form id="contactSearchForm">
 <label for="contactid">Contact ID</label>
 <input type="text" id="contactid">
 <input type="submit" value="Search">
 <input type="button" value="Search (XML response)"
 id="xmlSearch">
 </form>
</div>
```

#### ***Step 4: Create an external CSS File***

Create an external stylesheet to be linked into the main page. Do this by right-clicking the nodejs/public folder and selecting New > File (if using Eclipse). Name it **contacts.css**. At this stage we will only add a minor amount of styling. Add the following rules to contacts.css:

```
body { color: #000; }

#contactSearch {
 padding: 15px;
 width: 500px;
 height: 200px;
 border: 3px solid #00f;
}
```

Link it into the main document by placing the following statement in the head section:

```
<link href="contacts.css" rel="stylesheet" type="text/css">
```

#### ***Step 5: Attach jQuery, Execute on Ready***

Following the results <div>, place a <script> tag (essentially at the bottom of the body). Within the <script> tag, create a function that executes when the document is ready using jQuery:

```
<script type="text/javascript">

$(function() {

});

</script>
```

## Step 6: Submit the Form

Write a jQuery function to capture submit events. Submit events should be handled by capturing the event off of the form not the button!

```
$(function() {

 $("#contactSearchForm").submit(function(evt){

 evt.preventDefault();
 });
});
```

In jQuery, return false (used in IE event handling) is equivalent to evt.preventDefault()

## Step 7: Make a JSON Ajax Request

When the Send button is clicked, and Ajax request will be made. Add this code now, which will obtain the contact ID and then make the Ajax request. The Ajax request **type** will be a "GET" request. We need to set **headers** (one called *format*) to the value 'json'. The **url** will be the one we tested in the previous exercise, except the contactid will come from the form input element. Finally, the **dataType** attribute tells jQuery what kind of content will be coming back from the server. The **success** function indicates a function to be called when the Ajax response comes back. Let's call ours *handleJson*.

```
$("#contactSearchForm").submit(function(evt){

 var contactid = $('#contactid').val();
 var options = {
 url : "/contact/" + contactid,
 type: "GET",
 headers: {'format' : 'json'},
 dataType: "json",
 success: handleJson
 };
});
```

```

 if(contactid)
 $.ajax(options);

 return false;
});

```

### ***Step 8: Create a JSON Ajax Callback***

The `handleJson` function will convert the returned JSON object into a string for viewing on the page. (Later we will work with templates to do this more efficiently.) jQuery provides the JSON object to us automatically as an argument to our function. Write the `handleJson()` callback as follows after the submit event handler:

```

function handleJson(results) {
 var contactid = results.contactid, name, address, phones, email,
 company, position, contactInfo = 'No contact found.'

 if(contactid) {
 name = results.name,
 address = results.address,
 phones = results.phones,
 email = results.email,
 company = results.company,
 position = results.position;

 contactInfo = name + ' ' + address + ' Ph: ' +
 phones[0].number + ' ' + phones[0].type;
 }

 $('#results').append('<p>' + contactInfo + '</p>');
}

```

### ***Step 9: Test it out!***

Save the file, restart and test your server (CTRL-C, node server.js). Visit

`http://localhost:8005/index.html`

### **Step 10: Want to Try It with XML? (optional)**

As an optional task, we'll send back XML and extract it on the client using jquery. On the client, add a button to submit data now in XML format:

```
<form id="contactSearchForm">
 <label for="contactid">Contact ID:</label>
 <input type="text" id="contactid" autofocus>
 <input type="submit" value="Search" />
 <input type="button" value="Search (XML response)" id="xmlSearch" />
</form>
```

Create an event handler for the button. The Ajax request is similar as before, but this time we will be receiving XML-based data in return, so dataType reflects this:

```
$('#xmlSearch').click(function(evt){

 var contactid = $('#contactid').val(),
 options = {
 url : "/contact/" + contactid,
 type : "GET",
 headers : {'format' : 'xml'},
 dataType: "xml",
 success : handleXml
 };

 if (contactid)
 $.ajax(options);

 return false;
});
```

Finally, create the handleXml Ajax callback function:

```
function handleXml(result) {
 var doc = $(result.documentElement), name, address, phones, email, company,
 position, contactInfo = 'No contact found.'

 if(doc) {
 contactid = doc.find('contactid').text(),
 name = doc.find('name').text(),
 address = doc.find('address').text(),
 ph_num = doc.find('phones').find('number').text(),
 ph_type = doc.find('phones').find('type').text(),
 email = doc.find('email').text(),
 company = doc.find('company').text(),
 position = doc.find('position').text(),

 contactInfo = name + ' ' + address + ' Ph: ' + ph_num + ' ' + ph_type;
 }

 $('#results').append('<p>' + contactInfo + '</p>');
}
```

Test it out by refreshing the HTML page.

# Exercise 4 – Multi-Column Layout



## *Overview*

In this exercise, you will add some structure to the HTML and create a multi-column layout. The solution will not contain much content (such as the header and footer), therefore an artificial height may be given for those sections if needed.

## *Objectives*

At the conclusion of this exercise, you should be able to:

- Apply floating CSS mechanisms
- Create columned look-and-feels



## *Step 1: Style the wrapper <div> and add reset.css*

Within contacts.css, add content to center and size the wrapper <div>:

```
#wrapper {
 width: 80%;
 min-width: 850px;
 margin: 10px auto;
}
```

Also, add a default font-size to the body element:

```
body {
 color: #000;
 font: 13px/1.231 arial,helvetica,clean,sans-serif;
}
```

In the <head> section, before the other CSS references, add a <link> to the reset-min.css file provided in your starter folder. Move this file into the nodejs/public folder.

```
<head>
 <link href="reset-min.css" rel="stylesheet" type="text/css">
 <link href="contacts.css" rel="stylesheet" type="text/css">
```

## ***Step 2: Style the Header & Footer***

Similarly, add CSS into the contact.css file to size the header and footer (artificially):

```
header, footer {
 background-color: rgb(0, 125, 195);
 height: 100px;
 position: relative;
}

header h1 {
 font: italic 48px Arial;
 padding-top: 10px;
 text-align: center;
 position: absolute;
 height: 60px;
 width: 600px;
 bottom: 10px;
 left: 30px;
}
```

## ***Step 3: Float and Size Columns***

Float the primary column div to the left and the secondary column div to the right. When floating, you will need to set a fixed width. We'll also ensure the main <div> doesn't "slide" underneath the floated divs.

```
#primary {
 float:left;
```

```

width:200px;
padding: 10px;
background-color: rgb(0, 125, 195);
}

#secondary {
 float:right;
 width:200px;
 padding: 10px;
 background-color: rgb(0, 125, 195);
}

#content { background-color: rgb(0, 125, 195); }

#main { margin: 0 220px; padding: 10px 0;
 background-color: rgb(255, 255, 255);
}

```

#### ***Step 4: Implement ClearFix and Test It!***

To ensure the menu and secondary floated columns never overlap the footer, we must apply clearfix to the content div (the one wrapping the menu and secondary divs). Add the following to the CSS and the class to the content div:

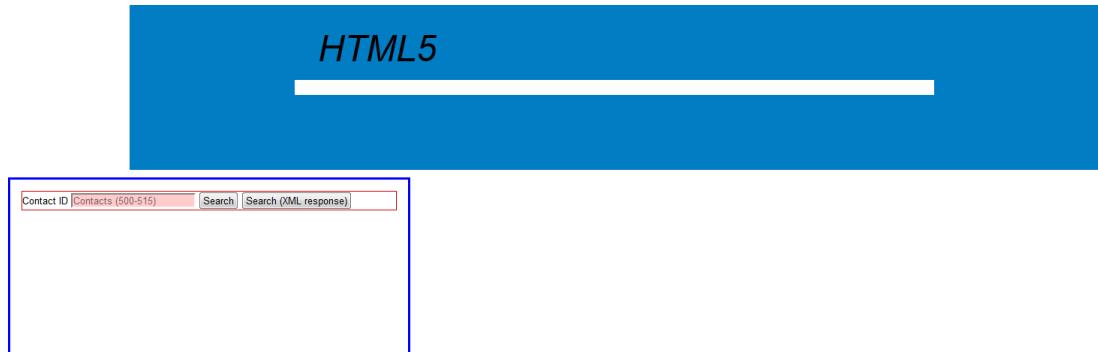
```

.clearfix { zoom: 1; }

.clearfix:after {
 content: "";
 display: block;
 clear: both;
}

<div id="content" class="clearfix">
```

Test your solution in Firefox/Chrome and use Firebug/Chrome Developer Tools to troubleshoot it as necessary.



### **Step 5: Add Some Data to the Grid, Test it Again!**

The following step uses a 3<sup>rd</sup> party templating system, called EJS. While not specifically a part of our discussions, we've used it here to illustrate the ease at which a table can be dynamically rendered. More can be learned about EJS from [embeddedjs.com](http://embeddedjs.com).

After your jQuery script tag in the <head>, add an EJS script tag:

```
<script type="text/javascript" src="ejs.js"></script>
```

Add the following code at the beginning of your JavaScript inside the jQuery ready function which renders a table of contacts:

```
$(function() {

 // load the table...
 $.ajax({ 'url': '/contact',
 success: function(data) {
 $('#main').append(new EJS(
 {url: '/tmpl/contacts.ejs'}).render(data));
 }
 });

 $('#contactSearchForm').submit(function(evt) {
 evt.preventDefault();
 var contactId = $('#contactId').val();
 var contacts = $('#contacts').val();
 var search = $('#search').val();
 var searchXml = $('#searchXml').val();
 });
});
```

Add the following CSS to render the table:

```

.contactdata {
 border: 1px solid black;
 margin: 20px;
}
.contactdata th { background: #cc8; font: 14px bold helvetica; }
.contactdata tr {
 padding: 3px 0;
 font: 14px normal helvetica;
 background: #ffe;
}
.contactdata tr:nth-child(odd) { background: #eee; }
.contactdata td { width: 60px; }
.contactdata th.addr { width: 250px; }
.contactdata th.name { width: 180px; }
.contactdata th.company { width: 200px; }

```

Your results should now look as follows:

**HTML5**

Contact Id	Name	Address	Phone Num	Phone Type	Email	Company	Job Title
500	Red Vectors	4517 Elm St. Riverside NJ 08075	(301)356-8921	home	fthompson@yahoo.com	ABC Inc.	President
501	Bob Green	2101 Eucalyptus Ave. Philadelphia PA 09119	(202)901-2121	home	dbreal@hotmail.com	Tupelo Industries	Product Manager
502	John Brown	331 Birch Cir. Black Hills SD 82101	(719)421-8875	home	jb6711@gmail.com	Last Rites LLC	Undertaker
503	Tina O. Range	82 Pine Dr. Lakewood CA 90713	(212)432-0944	home	tormut@besttip.com	Best Holistic Practices	Customer Service Representative
504	Berry Blumenthal	3012 Mahogany Ln. Denver CO 80101	(202)685-2323	home	bleubry@yahoo.com	Roller Heights Packaging	Owner
505	Jim Pinkade	2113 Redwood Blvd. Long Beach CA 90314	(204)740-9000	work	jpinkman@rocketmail.com	Hollywood West Novelties	Distribution Manager
506	Alicia Grey	415 Poplar Ct. St. Louis MO 72210	(211)870-6780	home	aigrey@blanksystems.com	Blank Systems Inc.	Lead Technical Engineer
507	Violet Waters	821 Ash Way Seattle WA 92230	(302)390-1181	home	waters@medcare.com	Home Medical Services	Regional Account Representative
508	Sandy White	906 Hickory Rd. Phoenix, AZ 83010	(213)221-4143	home	swhite@bricks.com	Bricks and More	Product Sales
509	Kay Black	1241 Maple Pl. Plano TX 72110	(401)322-8728	home	ksb2101@yahoo.com	Certified Signing Authorities	Graphics Artist

Contact ID

## ***Step 6: Style One Contact***

As a last change, let's use the templating system one last time to display the results of ONE contact. Add the following CSS to contacts.css:

```
.contact { background: #fcf;
 border-radius: 5px;
 border: 1px solid black;
 zoom: 1;
 margin: 20px 0;
 box-shadow: 5px 5px 5px 0px #777;
 }

.contact:after { clear: both; content: ""; display: block; } /* clearfix */
.contact li { font: 16px bold helvetica; float: left; width: 125px;
 list-style-type: none; padding: 2px; line-height: 18px; vertical-align: middle;
 }
```

## ***Step 7: Modify handleJson() to render a template***

Modify the handleJson() function as follows (we won't bother with the XML version):

```
function handleJson(results) {
 $('#results').append(
 new EJS({url: '/tmpl/contact.ejs'}).render(results));
}
```

Test it again!

# Exercise 6 – CSS3 Search Popup



## *Overview*

In this exercise, you will use CSS3 features to implement the Contact search box as a popup-style dialog that centers on the screen.

## *Objectives*

At the conclusion of this exercise, you should be able to:

- Incorporate CSS3 features such as rounded corners and gradients
- Utilize positioning, display, and event handling



## *Step 1: Hide the Contact Search <div>*

It is time to present the Contact Search box that has remained on the bottom of our page. Modify it by making it disappear from our view

```
#contactSearch {
 padding: 15px;
 width: 500px;
 height: 200px;
 border: 3px solid #00f;
 position: absolute;
 left: -1000px;
}
```

## **Step 2: Borrow from CSS3Please.com**

Let's add some rounded corners and a gradient to the Search Dialog Box. This, of course, would become a progressive enhancement feature in IE 6-8 (and 9 for the gradient).

You may add the gradient shown below, or you may visit CSS3Please.com and create your own gradient (preferred)

If you desire, you may also add a box shadow effect:

```
#contactSearch {
 padding: 15px;
 width: 500px;
 height: 200px;
 border: 3px solid #00f;
 border-radius: 25px;
 position: absolute;
 left: -1000px;

 -webkit-box-shadow: 0px 0px 4px 0px #c23cc2;
 box-shadow: 0px 0px 4px 0px #c23cc2;

 background-color: #1d2fcf;
 background-image: -webkit-gradient(linear, left top, left bottom, from(#1d2fcf),
 to(#faf5fa));
 background-image: -webkit-linear-gradient(top, #1d2fcf, #faf5fa);
 background-image: -moz-linear-gradient(top, #1d2fcf, #faf5fa);
 background-image: -ms-linear-gradient(top, #1d2fcf, #faf5fa);
 background-image: -o-linear-gradient(top, #1d2fcf, #faf5fa);
 background-image: linear-gradient(to bottom, #1d2fcf, #faf5fa);
}
```

## **Step 3: Make it Appear and Disappear (Centered)**

Within the <div id="primary"> but before the results div, add an anchor which will "popup" the new dialog. Style it as shown or change it to fit your liking:

```
<div id="primary">
 Contact Search
 <div id="results"></div>
</div>
```

Add the following to main.css:

```
#contactSearchLink {
 display: block;
 text-align: center;
 padding-top: 15px;
 text-decoration:none;
 font: 16px Arial italic;
 margin: 5px auto;
 width: 140px;
 height: 35px;
 border: 1px solid hsl(140, 10%, 10%);
 background-color: hsla(140, 50%, 50%, 0.5);
 color: #ffff00;
 -webkit-box-shadow: 0px 3px 3px 0px #000;
 -moz-box-shadow: 0px 3px 3px 0px #000;
 box-shadow: 0px 3px 3px 0px #000;
}
```

Test your solution again. You should see the search dialog is gone while the new link has appeared in the first column:

## ***Step 4: Show the Dialog***

The dialog is not really a OS-style dialog, but rather a <div> that we will make appear centered on the screen. To do this, add the following code after the handleXml() function:

```
$('#contactSearchLink').click(function(){
 var bodyWidth = $('body').width(),
 bodyHeight = $('body').height(),
 dialogWidth = $('#contactSearch').width(),
 dialogHeight = $('#contactSearch').height(),
 left = (bodyWidth - dialogWidth) / 2,
 top = (bodyHeight - dialogHeight) / 2;

 $('#contactSearch').css('left', left).css('top', top).show();

 return false;
});
```

It should work at this point. Test it out!

## ***Step 5: Now Make It Go Away!***

At this point the Dialog box works but doesn't go away. This can be implemented with two features: 1) add a Cancel option in the dialog. 2) Make it disappear after completing a request!

Both can be implemented easily.

First, add an anchor to the contactSearchForm such that you can cancel a search:

```
<div id="contactSearch">
 <form id="contactSearchForm">
 <label for="contactid">Contact ID:</label>
 <input type="text" id="contactid" autofocus>
 <input type="submit" value="Search">
 Cancel
 </form>
</div>
```

Position it in the lower right by using CSS:

```
#contactSearch a {
 position: absolute;
 bottom: 10px;
 right: 20px;
}
```

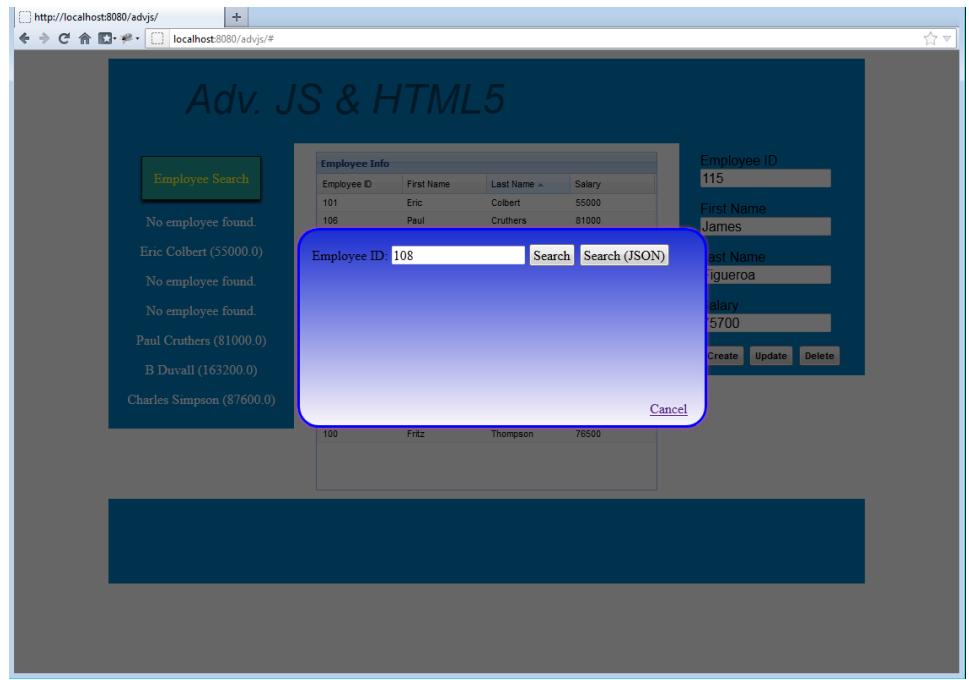
Add a close() method to your ContactSearch controller and hook up an event to the cancel link:

```
$('#cancelSearch').click(function() { close(); });
function close() { $('#contactSearch').hide(); }
```

To make the dialog go away when a search is performed, you will need to call close() after a search is performed. Place a call to close() at the end of handleJson() and handleXml().

## Step 6: Add an Overlay

An overlay will prevent someone from clicking on items in the background while the dialog box is open. See the screen shot below (notice the gray background).



Let's add this overlay as follows...

To add an overlay, first let's add a new `<div>` to our HTML as shown:

```
<div id="contactSearch ">
 <form id="contactSearchForm">
 <label for="contactid">Contact ID:</label>
 <input type="text" id="contactid" autofocus>
 <input type="submit" value="Search">
 Cancel
 </form>
</div>
<div id="overlay" class="overlay"></div>

<script type="text/javascript">
 ...
</script>
```

Next, we'll create a class and style the <div>.

```
.overlay {
 height: 100%;
 width: 100%;
 background-color: #777;
 background-color: hsla(0,0%, 0%, 0.6);
 position: absolute;
 top: 0;
 left: 0;
 z-index: 1000;
 display: none;
}
```

Notice the use of positioning. Also, notice we used width/height 100% to fill the screen. Finally, the background color is optional. If you want a transparent overlay, don't put a background color on it. We gave it a z-index value to put the overlay on top of all other HTML elements on the page.

Now we need to show and hide the overlay <div> at the same time that we show and hide the dialog box <div>. Two lines of JavaScript help us with this:

To show the overlay, add the following to #contactSearchLink onclick method:

```
$('#contactSearchLink').click(function(evt){
 evt.preventDefault();
 var bodyWidth = $('body').width(),
 bodyHeight = $('body').height(),
 dialogWidth = $('#contactSearch').width(),
 dialogHeight = $('#contactSearch').height(),
 left = (bodyWidth - dialogWidth) / 2,
 top = (bodyHeight - dialogHeight) / 2;
 $('#contactSearch').css('left', left).css('top', top).show();
 $('#overlay').show();
});
```

To hide the overlay, add the following to your close() function:

```
$('#cancelSearch').click(function(){ close(); });
function close() { $('#contactSearch').hide(); $('#overlay').hide(); }
```

The last step will ensure the dialog always appears on top of the overlay. Put a z-index property on the dialog to make it appear on top:

```
#contactSearch {
 padding: 15px;
 width: 450px;
 height: 200px;
 border: 3px solid #00f;
 position: absolute;
 display: none;
 -webkit-box-shadow: 0px 0px 4px 0px #c23cc2;
 border-radius: 25px;
 box-shadow: 0px 0px 4px 0px #c23cc2;
 background-color: #1d2fcf;
 background-image: -webkit-gradient(linear, left top, left bottom, from(#1d2fcf), to(#faf5fa));
 background-image: -webkit-linear-gradient(top, #1d2fcf, #faf5fa);
 background-image: -moz-linear-gradient(top, #1d2fcf, #faf5fa);
 background-image: -ms-linear-gradient(top, #1d2fcf, #faf5fa);
 background-image: -o-linear-gradient(top, #1d2fcf, #faf5fa);
 background-image: linear-gradient(to bottom, #1d2fcf, #faf5fa);
z-index: 1001;
}
```

That's it! Test it out!—you're done!

# Exercise 9 – Implementing CORS



## *Overview*

In this exercise, you will implement the Cross-Origin Resource Sharing (CORS) feature of modern browsers. This feature isn't supported in IE6, 7 and version 8 has slightly varying support for it. To support IE & CORS, see the slides.

To accomplish this task, we'll simulate a cross-domain request by starting a second nodejs server that runs on port 8006. You will serve up the main page (index.html) from port 8006, but retrieve data from port 8005.



## *Objectives*

At the conclusion of this exercise, you should be able to:

- Understand how to implement CORS on the server and client
- Utilize jQuery to implement CORS
- Make use of cross-domain request techniques from HTML5

## *Step 1: Create the Cross-Domain Environment*

Copy the server\_lab09.js file from the lab09/starter directory into the nodejs folder. Open a command window and start this server issuing the command node server\_lab09.js.

You should see a message as follows:

**Lab09 server running on port 8006.**

**Use this server to serve up the index.html page only.**

Open localhost:8006/index.html and verify that the page does not properly load.

### ***Step 2: Modify index.html***

Edit index.html in 3 places. Each of these places should be where an ajax request is made (don't modify the calls to load the templates). You should specify FULL URLs to the 8005 server. One example has been shown for you:

```
$('#contactSearchForm').submit(function(evt) {

 var contactid = $('#contactid').val();
 var options = {
 url : "http://localhost:8005/contact/" + contactid,
 type : "GET",
 headers : {'format' : 'json'},
 dataType: "json",
 success : handleJson
 };

 if (contactid)
 $.ajax(options);

 return false;
});
```

Repeat this for the other two ajax calls (one for the submit button, one for the XML button).

### ***Step 3: Test it again!***

Use **Firefox** and perform the same test as you did in Step 3. Hopefully with better results!

# Supplemental Exercise – Support for RESTful Services



## *Overview*

In this exercise, you will continue evolving the contact exercise started in earlier labs. You will add support for making RESTful services.

## *Objectives*

At the conclusion of this exercise, you should be able to:

- Make RESTful calls to the server
- Incorporate Forms



## *Step 1: Create an HTML Contact Details Form*

Add the following within the "secondary" div:

```
<div id="contactDetails">
 <form id="contactDetailsForm">
 <label for="contactidDetails">Contact ID</label>
 <input type="text" id="contactidDetails" readonly="readonly">
 <label for="nameDetails">Name</label>
 <input type="text" id="nameDetails">
 <label for="addressDetails">Address</label>
 <input type="text" id="addressDetails">
 <label for="phonenumDetails">Primary Phone</label>
 <input type="text" id="phonenumDetails">
 <label for="phonetypeDetails">Type</label>
 <input type="text" id="phonetypeDetails">
 <label for="emailDetails">Email</label>
 <input type="text" id="emailDetails">
 <label for="companyDetails">Company</label>
 <input type="text" id="companyDetails">
 <label for="positionDetails">Job Title</label>
 <input type="text" id="positionDetails">
```

```

<input type="button" value="Create" id="createBtn">
<input type="button" value="Update" id="updateBtn">
<input type="button" value="Delete" id="deleteBtn">
</form>
</div>

```

## ***Step 2: Populate the Form from the Table***

The following is a jQuery event handler that selects rows and populates the form with appropriate details.

```

$(document).on('click', '#contactgrid tr', function(evt) {
});

```

Complete the event handling function now as follows:

```

$(document).on('click', '#contactgrid tr', function(evt) {
 var data = $(this).children('td');
 $('#contactidDetails').val(data.eq(0).text());
 $('#nameDetails').val(data.eq(1).text());
 $('#addressDetails').val(data.eq(2).text());
 $('#phonenumDetails').val(data.eq(3).text());
 $('#phonetypeDetails').val(data.eq(4).text());
 $('#emailDetails').val(data.eq(5).text());
 $('#companyDetails').val(data.eq(6).text());
 $('#positionDetails').val(data.eq(7).text());
});

```

### **Step 3: Enable the Insert, Update, Delete Buttons**

These buttons are part of the details form functionality.

To delete a record, we must add event handling functionality to the Delete button. The handler must obtain the contactidDetails from the Contact Details form and invoke a DELETE-style Ajax call:

```
$('#deleteBtn').click(function(evt){
 evt.preventDefault();
 var contactid = $('#contactidDetails').val();
 if (contactid.length > 0)
 $.ajax ({url: '/contact/'+contactid,
 type: 'DELETE',
 success: function(){ renderTable(); }
 });
});

function renderTable() {
 $.ajax({url: '/contact',
 success: function(data) {
 $('#main').html(new EJS(
 {url: '/tmpl/contacts.ejs'}).render(data));
 }
 });
}
```

The second function above re-renders the table after a row has been deleted.

Next, we'll add the ability to update a row:

```

$('#updateBtn').click(function(evt){
 evt.preventDefault();
 var contactid = $('#contactidDetails').val(),
 name = $('#nameDetails').val(),
 address = $('#addressDetails').val(),
 phone_num = $('#phonenumDetails').val(),
 phone_type = $('#phonetypeDetails').val(),
 email = $('#emailDetails').val(),
 company = $('#companyDetails').val(),
 position = $('#positionDetails').val(),

 params = {name: name, address: address, phone_num:
 phone_num, phone_type: phone_type, email:email,
 company:company, position:position};

 $.ajax({ url: '/contact/' + contactid , type: 'PUT', data : params,
 success: function() { renderTable(); } }
);
});

```

The above function will update a contact on the server. It saves the object back on the server side and re-renders the data grid.

The last function is to add the *create* button functionality.

```
$('#createBtn').click(function(evt){
 evt.preventDefault();
 var contactid = $('#contactidDetails').val(),
 name = $('#nameDetails').val(),
 address = $('#addressDetails').val(),
 phone_num = $('#phonenumDetails').val(),
 phone_type = $('#phonetypeDetails').val(),
 email = $('#emailDetails').val(),
 company = $('#companyDetails').val(),
 position = $('#positionDetails').val(),

 params = {name: name, address: address, phone_num: phone_num,
 phone_type: phone_type, email:email, company:company,
 position:position};

 $.ajax({ url: '/contact/' , type: 'POST', data : params,
 success: function() { renderTable(); }
 });
});
```

#### ***Step 4: Style It Your Way***

Add CSS to style your form your way. As a hint, here is a suggested set of CSS rules:

```

#contactDetails {
 width: 85%;
 margin: 0 auto;
 font: normal 16px Arial;
}

#contactDetails label, #contactDetails input[type="text"] {
 display: block;
 width: 90%;
}

#contactDetails input[type="text"] {
 margin-bottom: 15px;
}

#contactDetails input[type="button"] {
 font: bold 11px Arial;
 padding: 2px;
}

```

### ***Step 5: Test it Again!***

Save your files, ensure your server is running and test it once again. Here's how it might look when finished:

HTML5

Contact ID	Name	Address	Phone Num	Phone Type	Email	Company	Job Title
500	Red Victoria	4517 Elm St. Riveride NJ 08075	(301)556-8921	home	fttomprix@yahoo.com	ABC Inc.	President
501	Bob Green	2101 Eucalyptus Ave. Philadelphia PA 09119	(202)901-2121	home	dreal@hotmail.com	Tupelo Industries	Product Manager
502	John Brown	331 Birch Cir. Black Hills SD 82101	(719)421-8875	home	jbt12@gmail.com	Last Rate LLC	Undertaker
504	Berry Blumenthal	3012 Mahogany Ln. Denver CO 80101	(202)685-2323	home	bleubry@yahoo.com	Roller Heights Packaging	asdfasfd
506	Alicia Grey	415 Poplar Ct. St. Louis MO 72210	(211)870-6780	home	algrey@blanksystems.com	Blank Systems Inc.	Last Name
507	Violet Waters	821 Ash Way Seattle WA 92230	(302)390-1181	home	waters@medicare.com	Home Medical Services	Technical Engineer
508	Sandy White	906 Hickory Rd. Phoenix, AZ 83010	(213)221-4143	mobile	swhite@bricka.com	Bricks and More	Regional Account Representative
509	Kay Black	1241 Maple Pl. Plano TX 72110	(401)332-8728	home	ksb2101@yahoo.com	Certified Signing Authority	Product Sales
511	Tina O. Range	82 Pine Dr. Lakewood CA 90713	(212)432-0944	home	tornut@besthp.com	Best Holistic Practices	Graphics Artist
512	Tina O. Range	82 Pine Dr. Lakewood CA 90713	(212)432-0944	home	tornut@besthp.com	Best Holistic Practices	Customer Service Representative

Name: Berry Blumenthal

Address: 3012 Mahogany Ln. Denver CO

Primary Phone: (202)685-2323

Type: home

Email: bleubry@yahoo.com

Company: Roller Heights Packaging

Job Title: asdfasfd