FIREBRAND

Microsoft

MCSD: 70-480: Programming in HTML5 with JavaScript and CSS3

MHTML5 Courseware

Version 1.3

Module 0
Introduction

Programming in HTML5 with JavaScript and CSS3

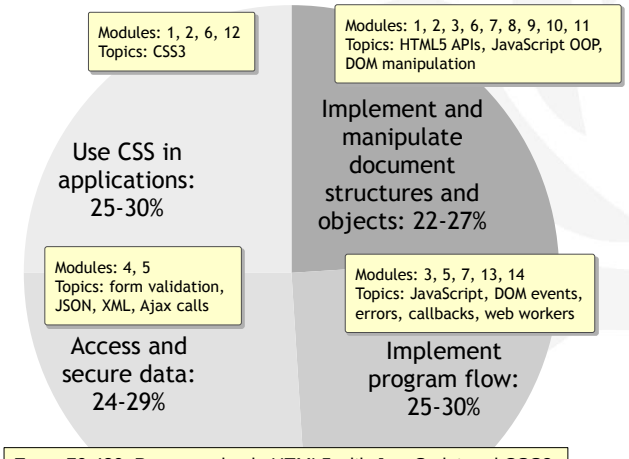
Updated 15th August 2015

Overview

- Why is this course called *Programming in HTML5* rather than *Programming in JavaScript*?
- JavaScript (25% of the course)
  - Language and native objects like RegExp, Date, and so on
  - Common libraries like jQuery
- HTML5 (75%): all of this could be used with VBScript!
  - HTML5, Document Object Model
  - Styling with CSS3
  - Calling services with XMLHttpRequest
  - Web Storage, Cache, Multimedia, Drag and Drop, Geolocation
  - Graphics, Animation
  - Web Sockets, Web Workers

Programming in HTML5 with JavaScript and CSS3
70-480 Exam Guide

0.3



Exam 70-480: Programming in HTML5 with JavaScript and CSS3
http://www.microsoft.com/learning/en-us/exam-70-480.aspx

The SharePoint Viking: Bjoern H Rapp
http://www.sharepointviking.com/70-480-exam-preparation-guide/

June 2013
39 questions in 130 minutes

September 2013
58 questions in 120 minutes

December 2013
45 questions in 120 minutes

November 2014
49 questions
120 minutes

Since March 2015
60 questions
120 minutes
No case study

Microsoft
Specialist

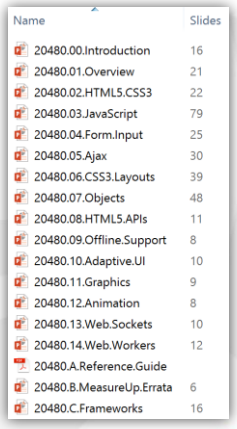
Programming in HTML5 with JavaScript and CSS3 Specialist

Programming in HTML5 with JavaScript and CSS3
Estimate of Number of Exam Questions per Module

0.4

Module	Qs
1: Overview of HTML and CSS	3
2: Creating and Styling HTML Pages	4
3: Introduction to JavaScript	7
4: Creating Forms to Collect and Validate User Input	6
5: Communicating with a Remote Server	7
6: Styling HTML5 by Using CSS3	8
7: Creating Objects and Methods by Using JavaScript	4
8: Creating Interactive Pages by Using HTML5 APIs	3
9: Adding Offline Support to Web Applications	3
10: Implementing an Adaptive User Interface	3
11: Creating Advanced Graphics	4
12: Animating the User Interface	1
13: Implementing Real-time Communication by Using Web Sockets	2
14: Performing Background Processing by Using Web Workers	5
Total questions in exam	**60**

Firebrand extra slides to summarize and “fill the gaps”



Name	Slides
20480.00.Introduction	16
20480.01.Overview	21
20480.02.HTML5.CSS3	22
20480.03.JavaScript	79
20480.04.Form.Input	25
20480.05.Ajax	30
20480.06.CSS3.Layouts	39
20480.07.Objects	48
20480.08.HTML5.APIs	11
20480.09.Offline.Support	8
20480.10.Adaptive.UI	10
20480.11.Graphics	9
20480.12.Animation	8
20480.13.Web.Sockets	10
20480.14.Web.Workers	12
20480.A.Reference.Guide	
20480.B.MeasureUp.Errata	6
20480.C.Frameworks	16

Suggested Extra Materials
Succinctly Series: The essentials in about 100 pages

0.5

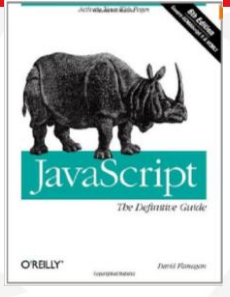
- Useful for 20480
  - HTTP Succinctly
  - JavaScript Succinctly
  - jQuery Succinctly
  - Regular Expressions Succinctly
  - Twitter Bootstrap Succinctly

About the Succinctly series
http://www.syncfusion.com/resources/techportal

Further Study
JavaScript

0.6

- JavaScript: The Definitive Guide
  - David Flanagan, 10 May 2011
  - Edition: 6, Paperback: 1100 pages
- JavaScript: The Good Parts
  - Douglas Crockford, 15 May 2008
  - Paperback: 172 pages



- Two hour video lecture from Douglas Crockford:

Douglas Crockford: The JavaScript Programming Language
https://www.youtube.com/watch?v=v2ifWcnQs6M

Further Study
Microsoft Virtual Academy

0.7

- JavaScript Fundamentals for Absolute Beginners

http://www.microsoftvirtualacademy.com/training-courses/javascript-fundamentals-for-absolute-beginners

- HTML5 & CSS3 Fundamentals: Development for Absolute Beginners

http://www.microsoftvirtualacademy.com/training-courses/html5-css3-fundamentals-development-for-absolute-beginners

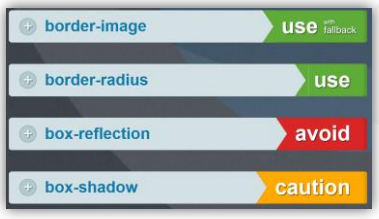
- Developing in HTML5 with JavaScript and CSS3 Jump Start

http://www.microsoftvirtualacademy.com/training-courses/learn-html5-with-javascript-css3-jumpstart-training

Further Study
HTML5 Rocks and HTML5 Please

0.8

- Lots of great tutorials, articles, and advice

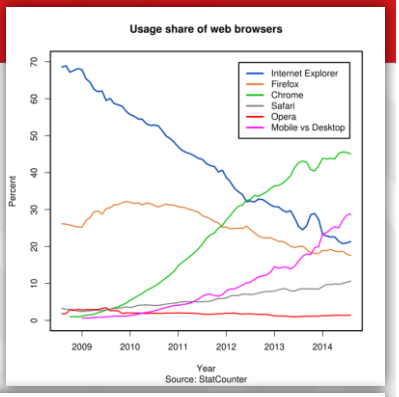


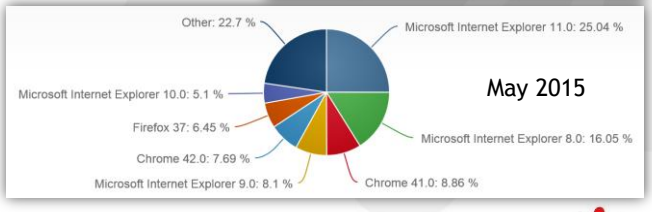
HTML5 Rocks
http://www.html5rocks.com/en/

HTML5 Please
http://html5please.com/

Browsers
Microsoft Browsers

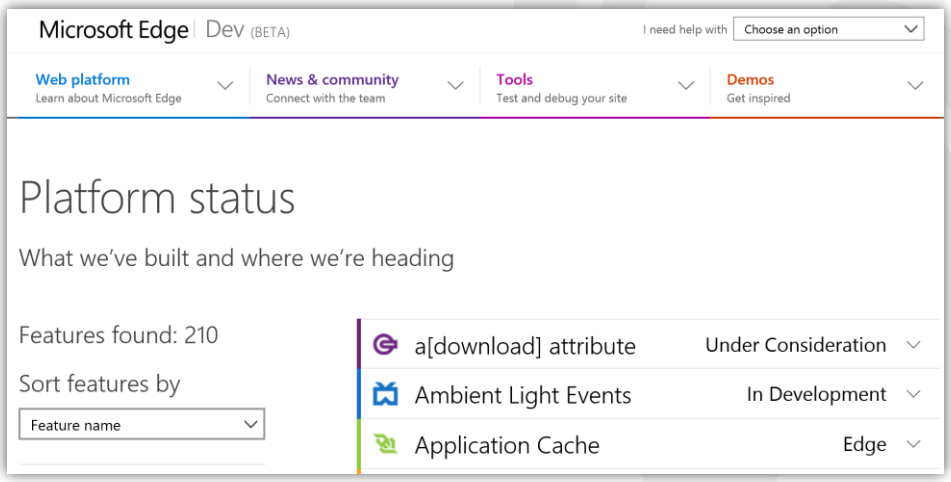
- Microsoft Internet Explorer
  - 6.0 August 2001
  - 7.0 October 2006
  - 8.0 March 2009
  - 9.0 March 2011
  - 10.0 August 2012
  - 11.0 October 2013
- Microsoft Edge
  - July 2015





Desktop Browser Version Market Share
https://www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=0

Browsers
Microsoft Edge



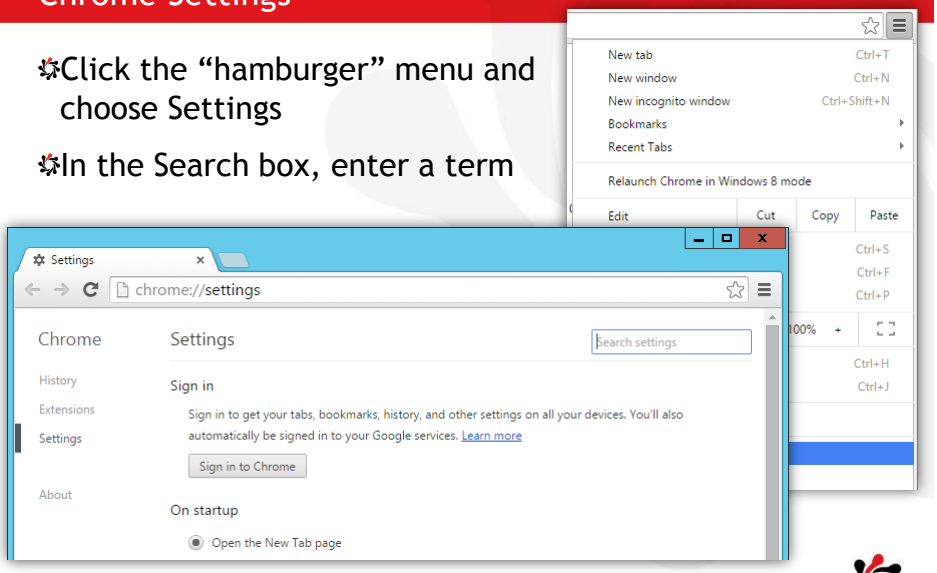
Microsoft Edge – Platform Status
https://dev.modern.ie/platform/status//

Browsers

Chrome Settings

0.11

- Click the “hamburger” menu and choose Settings
- In the Search box, enter a term

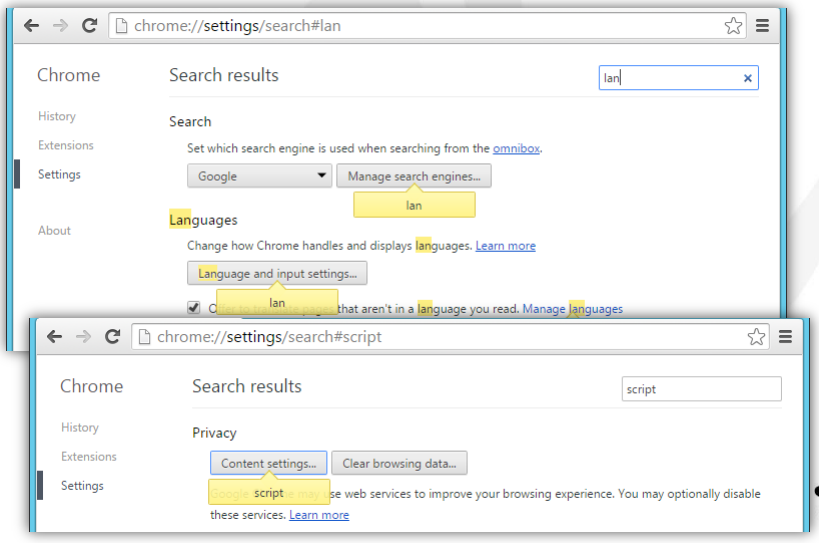


Browsers

Chrome Settings Search

0.12

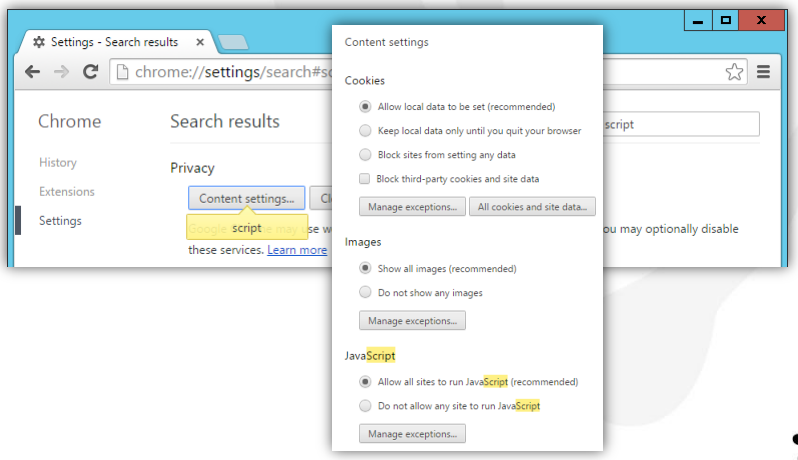
- Search highlights matches, even in sub-sections



Browsers

Chrome Settings – Disabling JavaScript

- To disable JavaScript, search for “script” and click Content settings…



Versions

JavaScript and ECMAScript

- JavaScript is officially named ECMAScript
  - “JavaScript” is a trademark of Oracle Corporation
  - Microsoft’s version of it is known as JScript
- Developed under the name Mocha, the language was officially called LiveScript when it first shipped
  - Originally developed by Brendan Eich
- In June 1997, Ecma International published the first edition of the ECMA-262 specification
  - The previous edition was 5, release in 2009
  - The current edition is 6, released in June 2015

ECMAScript 2015 - Let’s talk about ES6
https://medium.com/ecmascript-2015

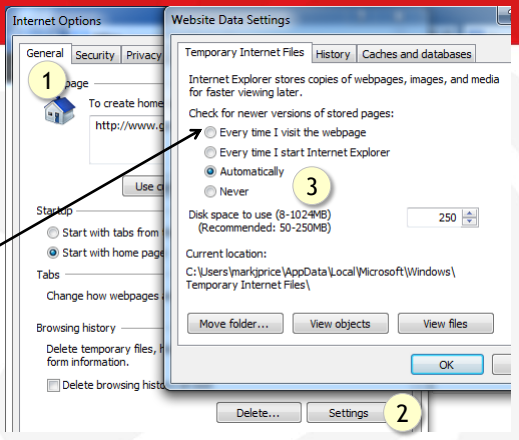
ECMAScript® 2015 Language Specification (ECMA-262, 6th Edition / June 2015)
http://www.ecma-international.org/ecma-262/6.0/index.html

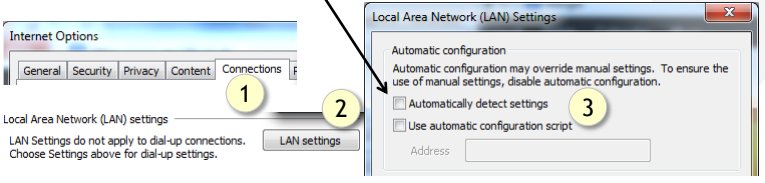
20480B Virtual Machine
Web Browser Issues

0.15

- Switch *off* compatibility view

- Ensure your browser is *not* caching old pages
  - Ctrl+F5 to force a refresh
- Switch *off* "Automatically detect settings" for LAN

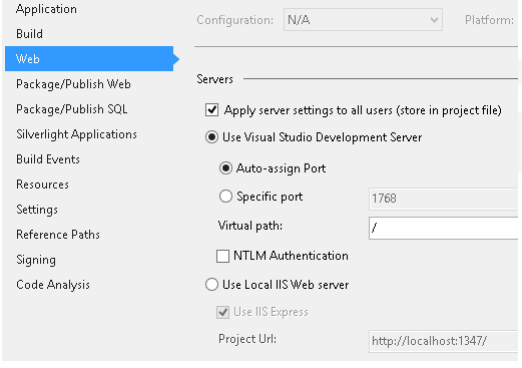




20480B Virtual Machine
Web Server Issues

0.16

- All the 20480B Starter and Solution projects share the same port numbers hosted in IIS Express
  - Change your project properties to use the Visual Studio Development Server on an different specific port



Module 1
Overview of HTML and CSS

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Overview of HTML and CSS

Contents

Exam Topic: Structure a CSS file by using CSS selectors
- ❑ Reference elements correctly
- ❑ Implement inheritance
- ❑ Override inheritance by using !important

MOC Errata

- Position 3-2682, alt is NOT a valid attribute for <a>

CSS
http://msdn.microsoft.com/library/ie/hh673536(v=vs.85).aspx

HTML
Common Elements and Their Common Attributes

1.3

Element	Attributes*	Description
html, body		Root and visible part of a web page
head, title, meta		Meta-data about web page and its title
h1, h2, h3, h4, h5, h6		Heading levels
p, br, hr		Paragraph, line break, horizontal rule
img	src, alt	Image, URL of picture, alternative text
a	href, target	Anchor for hyperlinks
b, i, u		HTML4: bold, italic, underline HTML5: keywords, alternative voice
strong, em		Important text, stress emphasis
table, tr, th, td		Table, row, header, and datum (column)
div, span		"Block" and "inline" content
form	method, action	Form for gathering input from user
input	type	Input control (e.g. text box)
ol, ul, li		Ordered list, unordered list, list item

* All elements can have an id, a style, and a class attribute

HTML/Elements/b - Example of bad usage
https://www.w3.org/wiki/HTML/Elements/b

HTML
Languages and Serializations

1.4

- HTML, XML, and XHTML are defined in terms of a *language* and a *serialization*
  - Language defines the vocabulary (nouns, verbs)
  - Serialization defines rules for the format (grammar)
  - HTML 4.01 specification defines both the HTML language and the HTML serialization
  - XML 1.0 specification defines a serialization but leaves the language to be defined by other specifications
  - XHTML 1.0 and 1.1 specifications use the same language as HTML 4.01 but use a different serialization, one that is compatible with the XML 1.0 specification (stricter rules)
  - HTML5 specification describes a new language for both HTML and XHTML intended to be backward compatible

HTML 4, HTML 5, XHTML, MIME types - the definitive resource
http://stackoverflow.com/questions/2662508/html-4-html-5-xhtml-mime-types-the-definitive-resource

HTML

XHTML Rules

1.5

- You MUST follow these rules for valid XHTML
  - XHTML DOCTYPE is mandatory
  - The XML namespace attribute in <html> is mandatory
  - <html>, <head>, <title>, and <body> is mandatory
  - XHTML elements must be properly nested
  - XHTML elements must always be closed (inc. empty elements)
  - XHTML elements must be in lowercase
  - XHTML documents must have one root element
  - Attribute names must be in lower case
  - Attribute values must be quoted
  - Attribute minimization is forbidden, for example:

```
<input disabled="disabled" /> <!--not minimized valid XHTML5-->
```

```
<input disabled> <!--minimized HTML5-->
```

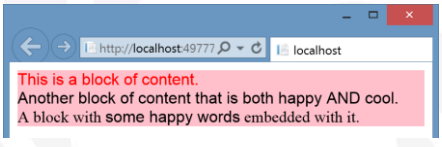
HTML and XHTML
http://www.w3schools.com/html/html_xhtml.asp

HTML

div, span, id, and class

1.6

- Marking areas of content
  - Use div for *block* content
  - Use span for *inline* content
- Naming areas of content
  - Use id to identify a single element
  - Use class to identify multiple elements




```
div {
  background-color: pink;
}
#alpha {
  color: red;
}
.happy {
  font-family: Arial;
}
```

```
<div id="alpha" class="happy">
  This is a block of content.
</div>
<div id="beta" class="happy cool">
  Another block of content that is both happy AND cool.
</div>
<div>
  A block with <span class="happy">some happy words</span>
embedded with it.
</div>
```

HTML
Hyperlinks

1.7

- The a (anchor) tag creates clickable hyperlinks
  - href: Specifies the URL of the resource the link goes to
- A linked page is normally displayed in the current browser window, unless you specify another target
  - _blank: Opens the linked document in a new window or tab

```
<a href="customers/edit/2" target="_blank">Edit Bob in New Window</a>
```

  - _self (default), _parent, _top, framename: Opens the linked document in the same, parent, top, or named frame
  - In HTML 4 <a> could be a hyperlink or a named anchor like <a name="bookmark" /> but in HTML5 the <a> tag cannot be named so use any element with an ID instead

```
<a href="#bookmark">Jump to bookmark</a>
```

```
<span id="bookmark" />
```

HTML <a> Tag
http://www.w3schools.com/tags/tag_a.asp

HTML
address element

1.8

- <address> is for contacting the author of a web page; it is NOT for postal addresses!
  - "The address element provides contact information for a document or part of a document. Information provided by address may include the names of the document's maintainers, links to the maintainers' Web pages, e-mail addresses for feedback, postal addresses, phone numbers, and so on. The address element is not appropriate for all postal and e-mail addresses; it should be reserved for providing such information about the contact people for the document."

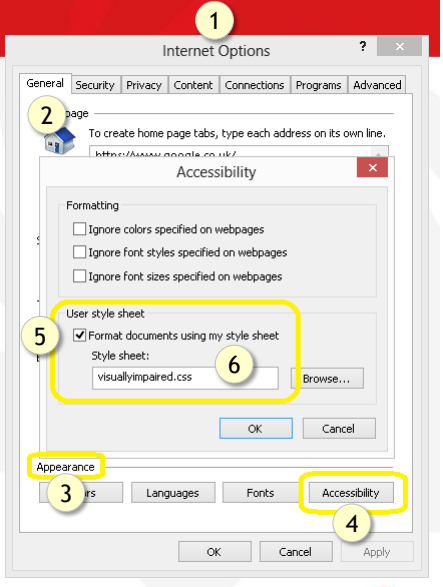
The Address Element
http://html5doctor.com/the-address-element/

Origins

- Style sheets may have three different origins:
  - **Author**: specifies style sheets for a source document according to the conventions of the document language
  - **User**: may be able to specify style information for a particular document, for example, for visual impairment
  - **User agent**: Conforming user agents must apply a default style sheet (e.g., for visual browsers, the EM element in HTML is presented using an italic font)
- By default, rules in author style sheets have higher precedence than rules in user style sheets which have higher precedence than rules in user agent style sheets
  - author > user > user agent, except…
  - …precedence is reversed for **!important** rules, so: user agent > user > author

Setting a User Style Sheet

- For Internet Explorer
  1. Internet Options
  2. On the General tab
  3. Appearance
  4. Accessibility
  5. Select “Format documents using my style sheet”
  6. Type or browse for a .css
- Other browsers don’t make it so easy



CSS
Rule Precedence

1.11

- Sort according to importance (normal or important), animations, and origin (author, user, or user agent)
- In descending order of precedence:
  1. user agent !important declarations (highest precedence)
  2. user !important declarations
  3. author !important declarations
  4. animations
  5. author declarations
  6. user declarations
  7. user agent declarations (lowest precedence)

6.2 Important Declarations: the '!important' annotation
http://www.w3.org/TR/css3-cascade/
(PPT bug with this: http://www.w3.org/TR/css3-cascade/#important)

You must manually look for 6.2 in the Table of Contents because PowerPoint cannot use # in URLs!

CSS
Selector Specificity

1.12

- A selector's specificity is calculated as follows
  - a = number of ID selectors
  - b = number of class, attribute, and pseudo-classes selectors
  - c = number of type and pseudo-element selectors
- Order descending a, b, c
- Which has the highest specificity?

```
*                /* a=0 b=0 c=0 */
li               /* a=0 b=0 c=1 */
ul li            /* a=0 b=0 c=2 */
ul ol+li         /* a=0 b=0 c=3 */
h1 + *[rel=up]   /* a=0 b=1 c=1 */
ul ol li.red     /* a=0 b=1 c=3 */
li.red.level     /* a=0 b=2 c=1 */
#x34y            /* a=1 b=0 c=0 */
#s12:not(foo)    /* a=1 b=0 c=1 */
```

```
ul li ol.bar li ol li.foo ol li > ol li span::first   /* a=0 b=2 c=12 */
body section ~ a:hover.cool#bob                      /* a=1 b=2 c=3 */
```

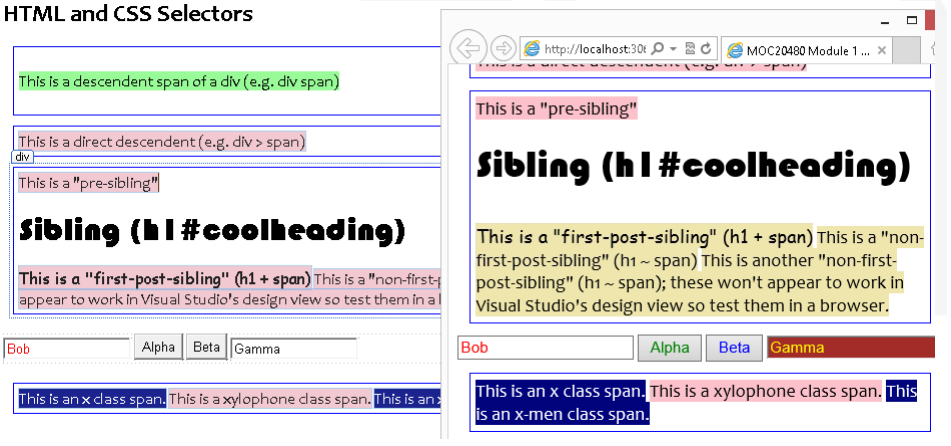
9 Calculating a selector's specificity
http://www.w3.org/TR/selectors/
http://www.w3.org/TR/selectors/#specificity

You must manually look for 9 in the Table of Contents because PowerPoint cannot use # in URLs!

Visual Studio Design View

To enable Design view in Visual Studio 2013: Right-click an HTML file in Solution Explorer, choose **Open With...**, then choose **HTML (Web Forms) Editor**

- Do NOT trust Design view to show styles
  - It does not correctly apply all CSS rules, for example, ~ selector and some attribute selectors, so always test in browser too



Lab and Demos

- The labs use Microsoft-only rules for the CSS Flexible Box model to layout the "tabs" at the top of the pages
  - The file **styles/nav.css** they use:

```
nav.page-nav .container {
  display: -ms-flexbox;
  display: flexbox;
}
```

  - To work in Chrome and Firefox as well, you need to add:

```
nav.page-nav .container {
  display: -moz-box;
  display: -webkit-box;
  display: -ms-flexbox;
  display: flexbox;
}
```

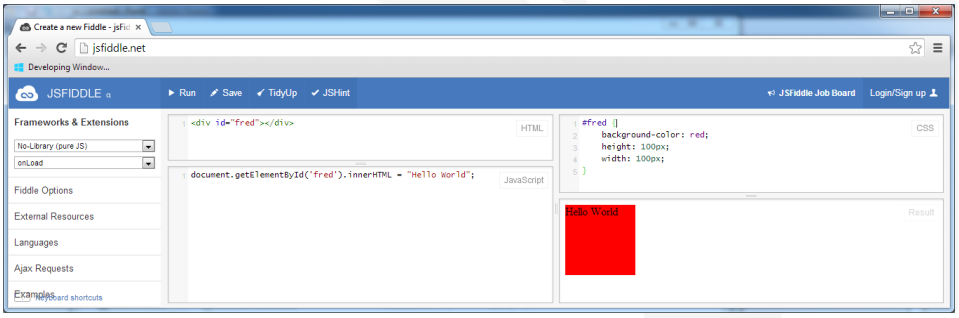
- Notes
  - Labs have "code signature blocks"—just ignore them
  - Demos in Module 1 enable NTLM Authentication; switch this off in Project Properties—Start Options

1.15

Further Study

Experimenting with HTML5, CSS3, and JavaScript

- jsfiddle.net is great for experimenting on any device
  - BUT only use it when you don't have Visual Studio!
  - Can import common libraries easily



- jsperf.com is great for test cases for benchmarking

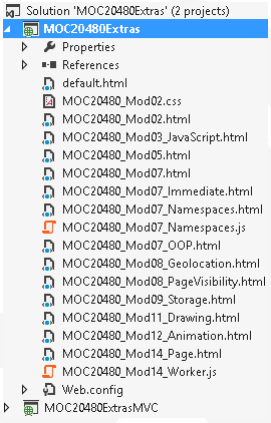
jsfiddle	jsPerf — JavaScript performance playground
http://jsfiddle.net/	http://jsperf.com/

1.16

Further Study

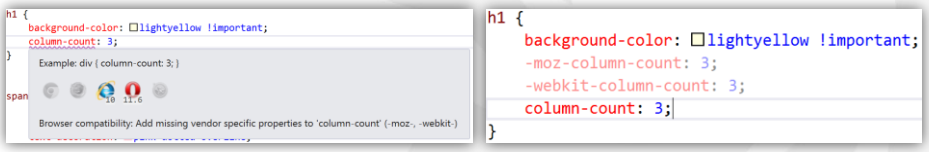
Extra Example Code

- Download extra code for examples
- Solution has two projects
  - ASP.NET Empty Web Application named **MOC20480Extras**
    - Contains HTML, CSS, and JavaScript files
    - default.html is the home page from which there are links to all other pages
  - ASP.NET MVC 4 Application named **MOC20480ExtrasMvc**
    - Contains an MVC Web API "REST service" and HTML "clients"
    - Make sure this application is running by right-clicking the project and choosing View in Browser



Further Study
Tools

- Web Essentials for 2012 and 2013
  - Extends Visual Studio with new features: colour preview, missing vendor prefix warning with auto-fix and sync, design-time minification, JSHint, and so on



- Fiddler and Wireshark
  - HTTP and other protocol monitoring
- Chrome and Firefox for testing multiple web browsers
- ILSpy or Redgate Reflector
  - For reverse-engineering other developers' .NET assemblies

Further Study
Zen Coding with Web Essentials

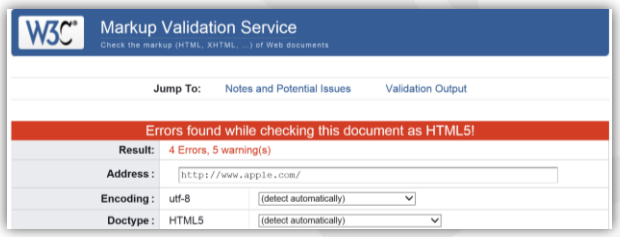
- Zen Coding is a set of shortcuts that will generate HTML code for you and save you a lot of typing

Enter this, press Tab	Generates
div	
ul>li	
table>tr>td	<table><tr><td></td></tr></table>
ul>li*3	
lorem	30 Latin words
lorem1	One Latin word
lorem3	Three Latin words
pix-300x200-animals	An image element referencing a random picture hosted by lorempixel.com

Zen Coding in Visual Studio 2012
http://www.johnpapa.net/zen-coding-in-visual-studio-2012/

Further Study
W3C and HTML5

- HTML5 was officially recommended on 28 October 2014
  - Almost no page on the Web is 100% “valid”, for example, Apple’s home page currently has 4 errors and 5 warnings



http://validator.w3.org/check?uri=apple.com&charset=%28detect+automatically%29&doctype=Inline&group=0

HTML5
A vocabulary and associated APIs for HTML and XHTML
W3C Recommendation 28 October 2014
http://www.w3.org/TR/html5/

The Group That Rules the Web
http://www.newyorker.com/tech/elements/group-rules-web

Further Study
WebKit Inspectors

- WebKit is a layout engine software component for rendering web pages in web browsers
  - It powers Apple’s Safari web browser, and a fork of the project is used by Google’s Chrome web browser
  - By September 2013, WebKit browser market share was larger than that of both the Trident engine used by Internet Explorer and the Gecko engine used by Firefox
- The link below is a tutorial showing how to use the WebKit Inspector developer tools in Chrome

THE WEBKIT INSPECTOR
http://jtaby.com/blog/2012/04/23/modern-web-development-part-1

Lab Alternative

- Download the Zen Garden HTML page
- Create a CSS file and link it to the HTML page that
  - Makes all the h1s red
  - Makes the first paragraph after an h3 underlined
  - Makes all paragraphs after h3s have a light green background
  - Removes underlines from links until the mouse hovers over them and changes the colour to something subtle
  - Alternates list items between bold and italic
  - Links in a footer must be smaller font size
- Create a user CSS file that overrides the h1 colour to green instead of red

CSS Zen Garden
http://www.csszengarden.com/

Module 2 Creating and Styling HTML Pages

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Contents

Exam Topic: Create the document structure
- Structure the UI by using semantic markup, including for search engines and screen readers (section, article, nav, header, footer, and aside)

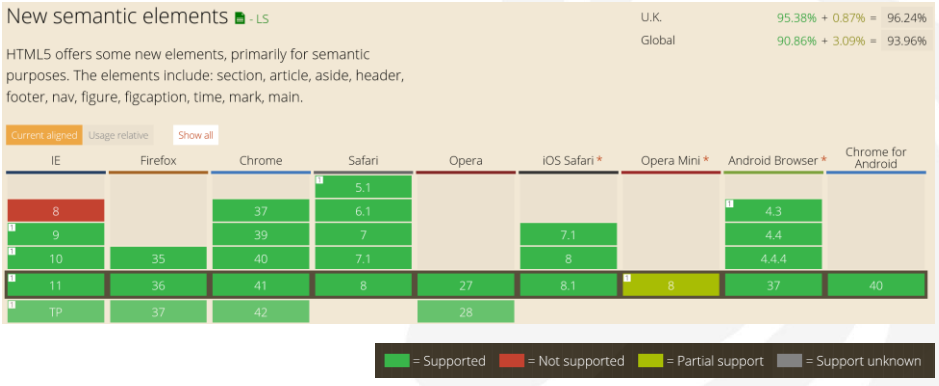
Exam Topic: Find elements by using CSS selectors and jQuery
- Choose the correct selector to reference an element
- Define element, style, and attribute selectors

Lab 2, Task 2, Step 4. (Position 4, 6960)

- The MOC says to set the margin to **1em 0 0,25em 0**
- It should have used dot (.) like this: **1em 0 0.25em 0**

2.3

Can I use…



Can I Use?
http://caniuse.com/
(PPT bug with this: http://caniuse.com/#feat=html5semantic)

You must manually search for "semantic" because PowerPoint cannot use # in URLs!

2.4

HTML
Semantic HTML5 Elements

Tag	Description
article	Defines independent, self-contained content like a single blog entry. An article should make sense on its own and it should be possible to distribute it independently from the rest of the page.
aside	Defines some content aside from the content it is placed in. The aside content should be tangentially related to the surrounding content.
div	Defines a division or grouped block of content in an HTML page.
nav	Defines a set of navigation links. Not all links of a page must be in a <nav> element. The <nav> element is intended only for a major block of navigation links. Browsers, such as screen readers for visually-impaired users, can use this element to determine whether to omit the initial rendering of this content.
section	Defines sections such as chapters of the page. Usually larger than divs and they have special abilities like shrinking heading levels.
span	Used to group inline-elements in a document. The tag provides no visual change by itself.

HTML Reference - (HTML5 Compliant)
http://www.w3schools.com/tags/default.asp

HTML
section, article, aside

2.5

- section
  - Used for grouping together thematically-related content
- article
  - Authors are encouraged to use the article element instead of the section element when it would make sense to syndicate the contents of the element
- aside
  - Used for tangentially related content
  - Ask yourself if the content within the aside can be removed without reducing the meaning of the main content

HTML5 section, aside, header, nav, footer elements – not as obvious as they sound
http://www.anthonycalzadilla.com/2010/08/html5-section-aside-header-nav-footer-elements-not-as-obvious-as-they-sound/

The article element
http://html5doctor.com/the-article-element/

HTML5, headings and sections
http://mattryall.net/blog/2008/10/html-5-headings-and-sections

HTML
section and Headings

2.6

- If you have h1 elements in nested sections they get smaller and smaller, like h2s and h3s, which would not happen with divs

Heading 1

Heading 2

Heading 3

Heading 1

Heading 1

Heading 1

```
<div>
    <h1>Heading 1</h1>
    <h2>Heading 2</h2>
    <h3>Heading 3</h3>
</div>
<div>
    <h1>Heading 1</h1>
    <section>
        <h1>Heading 1</h1>
        <section>
            <h1>Heading 1</h1>
        </section>
    </section>
</div>
```

- Note: sections also define new blocks for nth-child selectors

HTML
Boolean Attributes

2.7

- A number of attributes are boolean attributes
  - The *presence* of a boolean attribute on an element represents the true value, and the *absence* of the attribute represents the false value

```
<!-- enabling with HTML attribute -->
<input type=email required />
<input type=email required='' />
<input type=email required="" />
<input type=email required=required />
<input type=email required='required' />
<input type=email required="required" />
```

```
// enabling with JavaScript
control.required = true;
```

  - "true" and "false" are NOT allowed on boolean HTML attributes
  - To represent a false value the attribute *has* to be omitted

HTML
iframe

2.8

Some students have had multiple questions about iframe even though it is NOT on the exam objectives!

- Specifies an inline frame
  - An inline frame is used to embed another document within the current HTML document

Enables all sandbox restrictions

Must use a full end tag or you only see one iframe per page

```
<iframe sandbox="" src="http://www.w3schools.com"></iframe>
```

  - sandbox enables security so the contents of the iframe cannot use plugins, forms, scripts, and links to other browsing contexts
  - To enable features use space-separated: allow-forms allow-same-origin allow-scripts allow-top-navigation allow-popups

```
<iframe src="http://www.w3schools.com" seamless="seamless"
        sandbox="allow-scripts allow-forms"></iframe>
```

Loosens the sandbox

Removes border and scroll bar, enables style inheritance and links (but not well supported)

How to Safeguard your Site with HTML5 Sandbox
http://msdn.microsoft.com/en-us/hh563496.aspx

HTML <iframe> Tag
http://www.w3schools.com/tags/tag_iframe.asp

You've Been Framed!
http://www.debug.is/2015/04/15/youve-been-framed/

HTML

Obsolete Features in HTML5

2.9

- Obsolete but conforming features
  - **border** attribute on an **img** element
  - **language** attribute on a **script** element
  - **name** attribute on **a** elements
  - **maxlength** and **size** attributes on input elements whose type attributes are in the **Number** state
- Non-conforming elements
  - applet, acronym, bgsound, dir, frame, frameset, noframes, hgroup, isindex, listing, nextid, noembed, plaintext, strike, xmp, basefont, big, blink, center, font, marquee, multicol, nobr, spacer, tt
- Many attributes are non-conforming (see link below)

11 Obsolete features
http://www.w3.org/TR/2014/REC-html5-20141028/obsolete.html

HTML

Custom Data Attributes

2.10

- In HTML5 you have the ability to embed custom data attributes on all HTML elements
- The attribute must be at least one character long and must be prefixed with data-
  - It must not contain any uppercase letters

```
<span data-age="32" data-profession="Web Designer">Alice Jones</span>
```

HTML5 Custom Data Attributes (data-*)
http://html5doctor.com/html5-custom-data-attributes/

CSS
border

2.11

- Allows you to specify the style, size, and colour of an element's border
  - border-style: **none** (default), **dashed**, **dotted**, **solid**, and so on
  - border-width: pixels or **thin**, **medium** (default), **thick**
  - border-color: a colour value or **inherit**, **currentColor** (default)
- You can set individual border sides
  - border-top-style: dashed;
  - border-bottom-style: double;
- You can set multiple border styles
  - border-style: dotted solid double dashed; (TRouBLe)

CSS Border
http://www.w3schools.com/css/css_border.asp

CSS
nth-child selector

2.12

- nth-child can accept numbers, special keywords such as odd and even, and even formulae (*n* starts at 0)

```
ul li:nth-child(2) {
  color: red;
}
```

- Aaa
- Bbb
- Ccc
- Ddd
- Eee
- Fff
- Ggg
- Hhh

```
ul li:nth-child(odd) {
  color: red;
}
```

- Aaa
- Bbb
- Ccc
- Ddd
- Eee
- Fff
- Ggg
- Hhh

```
ul li:nth-child(3n + 2) {
  color: red;
}
```

- Aaa
- Bbb
- Ccc
- Ddd
- Eee
- Fff
- Ggg
- Hhh

```
<ul>
    <li>Aaa</li>
    <li>Bbb</li>
    <li>Ccc</li>
    <li>Ddd</li>
    <li>Eee</li>
    <li>Fff</li>
    <li>Ggg</li>
    <li>Hhh</li>
</ul>
```

- Note: jQuery supports all CSS selectors, including nth-child

How nth-child Works
http://css-tricks.com/how-nth-child-works/

CSS Techniques

Fallbacks to Support Old Browsers

2.13

Exam Topic: none

- Many features of CSS, for example the RGBA color value notation, are well-supported in modern browsers
  - For anything that might not be supported set an older supported property value *first*, then override it with the modern equivalent, for example, it's a good idea to specify solid fallback colors just in case


```
div {
  background-color: #000080; /* Fallback: navy blue in hexadecimal notation */
  color: #ffffff; /* Fallback: white in hexadecimal notation */
  background-color: rgba(0, 0, 128, 0.5); /* navy blue with 50% alpha */
  color: rgba(255, 255, 255, 0.3); /* white with 30% alpha (opacity) */
}
```

CSS Techniques

Normalizing Styles

2.14

Exam Topic: none

- The goal of a reset stylesheet is to reduce browser inconsistencies in things like default line heights, margins and font sizes of headings, and so on
- Normalize.css preserves useful defaults rather than "unstyling" everything
- Make it your first CSS link to "reset" or "normalize" any browser styles before applying your own styles

About normalize.css
http://nicolasgallagher.com/about-normalize-css/

What is the difference between Normalize.css and Reset CSS?
http://stackoverflow.com/questions/6887336/what-is-the-difference-between-normalize-css-and-reset-css

CSS Techniques
Multi-Level Numbering using Counters

2.15

Exam Topic: none

```
<ol>
    <li>Apples</li>
    <li>
        Bananas
        <ol>
            <li>Green</li>
            <li>Yellow</li>
            <li>
                Pink
                <ol>
                    <li>Tasty</li>
                    <li>Yum</li>
                </ol>
            </li>
        </ol>
    </li>
    <li>Cherries</li>
    <li>Elderberries</li>
</ol>
```

1. Apples
2. Bananas
    1. Green
    2. Yellow
    3. Pink
        1. Tasty
        2. Yum
3. Cherries
4. Elderberries

```
ol {
  counter-reset: item;
}
li {
  display: block;
}
  li:before {
    content: counters(item, ".") ". ";
    counter-increment: item;
  }
```

1. Apples
2. Bananas
    2.1. Green
    2.2. Yellow
    2.3. Pink
        2.3.1. Tasty
        2.3.2. Yum
3. Cherries
4. Elderberries

Counters
http://stackoverflow.com/questions/4098195/can-ordered-list-produce-result-that-looks-like-1-1-1-2-1-3-instead-of-just-1

CSS Techniques
Responsive Custom Ordered Lists

2.16

Exam Topic: none

```
<h1>Clarke’s Three Laws</h1>
<ol id="clarke-laws">
    <li>When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.
    <li>The only way of discovering the limits of the possible is to venture a little way past them into the impossible.
    <li>Any sufficiently advanced technology is indistinguishable from magic.
</ol>
```

Clarke’s Three Laws

1. When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.
2. The only way of discovering the limits of the possible is to venture a little way past them into the impossible.
3. Any sufficiently advanced technology is indistinguishable from magic.

Clarke’s Three Laws

1. When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.
2. The only way of discovering the limits of the possible is to venture a little way past them into the impossible.

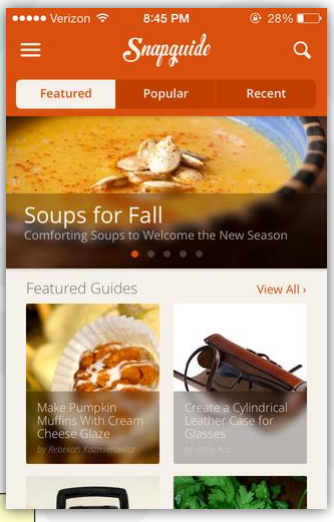
Creating Decorated Ordered Lists With CSS
http://demosthenes.info/blog/855/Creating-Decorated-Ordered-Lists-With-CSS

CSS Techniques
UI Design

2.17

Exam Topic: none

- A primer in digital aesthetics
- The Rules
  - Light comes from the sky
  - Black and white first
  - Double your whitespace
  - Learn the methods of overlaying text on images
  - Make text pop— and un-pop
  - Only use good fonts
  - Steal like an artist



7 Rules for Creating Gorgeous UI (Part 1)
https://medium.com/@erikdkennedy/7-rules-for-creating-gorgeous-ui-part-1-559d4e805cda

7 Rules for Creating Gorgeous UI (Part 2)
https://medium.com/@erikdkennedy/7-rules-for-creating-gorgeous-ui-part-2-430de537ba96

CSS Techniques
HTML5 Templates and Images

2.18

Exam Topic: none

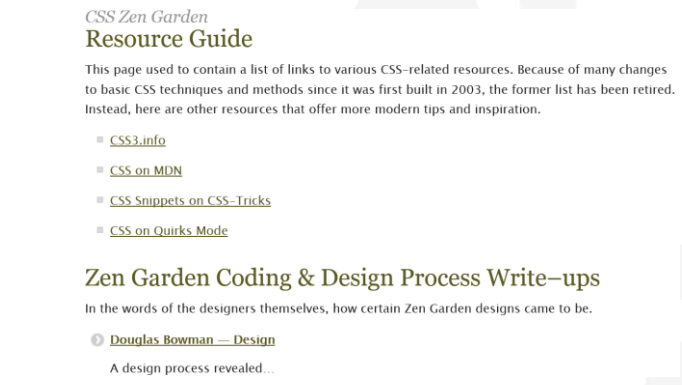
- Makes spiffy HTML5 site templates that are
  - Fully responsive
  - Built on intelligent HTML5 and CSS3
  - Super customizable
  - 100% free under the Creative Commons
- How to find public domain images
  - Follow the Harvard guide to learn how to find free images for use in your projects

HTML5 Up!
http://html5up.net

Images - Finding Public Domain & Creative Commons Media - Research Guides at Harvard Library
http://guides.library.harvard.edu/c.php?g=310751&p=2072816

Further Study
CSS Zen Garden

- To learn techniques, try CSS Zen Garden and CSS-Tricks



Resource Guide
http://www.mezzoblue.com/zengarden/resources/

CSS Zen Garden
http://www.csszengarden.com/

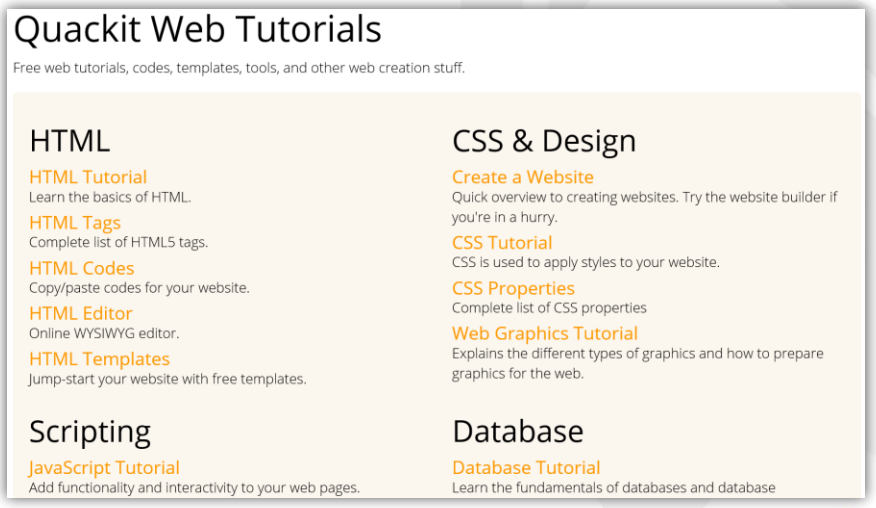
CSS-Tricks
https://css-tricks.com/almanac/

Further Study
CSS Diner

- CSS Diner
  - "Where we feast on CSS Selectors!"

CSS Diner
http://flukeout.github.io/

Further Study
Quackit Web Tutorials



Quackit Web Tutorials
http://www.quackit.com

Further Study
How to Code in HTML5 and CSS3

- "How to Code in HTML5 and CSS3" is a free e-book about making websites in HTML5 and CSS for absolute beginners
- It doesn't require any experience in IT to start
- The aim of this book is to show the art of making websites using a plain language which is full of practical analogies
- After reading over 100 pages you will get to know basic concepts and techniques of web development and be able to build your first website ever!

How to Code in HTML5 and CSS3
http://howtocodeinhtml.com

Module 3
Introduction to JavaScript

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Introduction to JavaScript
Contents

Controlling Program Flow (JavaScript)
http://msdn.microsoft.com/library/ie/kw1tezhk(v=vs.94).aspx

Contents

Exam Topic: Implement program flow
- ❑ Iterate across collections and array items
- ❑ Manage program decisions by using switch statements, if/then, and operators
- ❑ Evaluate expressions

Exam Topic: Raise and handle an event
- ❑ Handle common events exposed by DOM (OnBlur, OnFocus, OnClick)
- ❑ Declare and handle bubbled events
- ❑ Handle an event by using an anonymous function
- ❑ Use the "this" keyword to reference an object that fired an event

Exam Topic: Implement exception handling
- ❑ Set and respond to error codes
- ❑ Throw an exception
- ❑ Request for null checks
- ❑ Implement try-catch-finally blocks

Exam Topic: Write code that interacts with UI controls
- ❑ Programmatically add and modify HTML elements

MOC Errata

- Page 3-16, the 4th code block (position 5, 5641)
  - The MOC says

```
list.removeAttribute(list.attributes[0]);
```

  - It should have said

```
list.removeAttributeNode(list.attributes[0]);
```

- Page 3-24, on the slide (position 5, 8592)
  - The MOC says

```
$(#Warning")
```

```
$("input[type=text").val();
```

  - It should have said

```
$("#Warning")
```

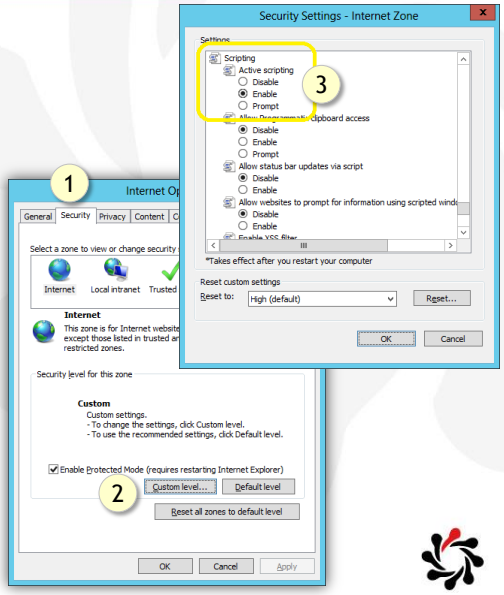
```
$("input[type=text]").val();
```

Basics

Ensuring JavaScript is Enabled

3.5

- The user can control if JavaScript is enabled
- To disable or enable JavaScript in IE
  - Internet Options
  - (1) Security Tab
  - (2) Custom level…
  - Scroll down to the (3) Scripting section and choose an option for Active scripting



Basics

Adding Script Statically to a Page

3.6

- In-page

```
<script>
  function doSomething() {
    // ...
  }
</script>
```

```
<script type="text/javascript">
  // type is optional because
  // it is now the default
```

- External file

Must use a full end tag for old versions of Internet Explorer

```
<script src="extLib.js"></script>
```

- For browsers with scripting disabled or is unsupported

```
<noscript>
  Warning! Your browser does not support
  JavaScript or you don't have it enabled.
</noscript>
```

Chrome will HTML encode any tags; Internet Explorer will not

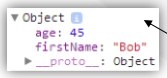
HTML <noscript> Tag
http://www.w3schools.com/tags/tag_noscript.asp

Basics
console

3.7

- *Most* browsers have a console that you can write to by using the console.log() method
  - View the console in Internet Explorer with F12 Developer Tools and then click the Console tab

```
console.log(bob);
```



Chrome outputs more details

```
[object Object]
```

Internet Explorer calls toString() on the object

- Some browsers do not have a console so you should check for its existence before using it

```
if(console !== undefined) {
```

```
if(console) {
```

console.log(object [, object, ...])
https://developers.google.com/chrome-developer-tools/docs/console-api#consolelogobject_object

Basics
Host Objects

3.8

- JavaScript executes in a host environment that can provide objects to access features of the host
  - navigator: represents the browser

```
console.log(navigator.appName); // => Microsoft Internet Explorer
```

  - window: represents the main window

```
if(window.confirm('Do you like JavaScript?')) window.alert('Yay!');
```

  - document: represents the page (DOM)

```
console.log(document.activeElement.tagName);
console.log(document.fileCreatedDate);
console.log(document.charset);
```

  - IE only: ScriptEngine() returns the language you are using, e.g., JScript or VBScript

The Navigator Object
http://www.w3schools.com/jsref/obj_navigator.asp

ScriptEngine Function (JavaScript)
http://msdn.microsoft.com/en-us/library/ie/efy5bay1(v=vs.94).aspx

Basics

Single or Double Quotes

3.9

- There isn't a preferred method, you can use either
  - However if you are using one form of quote in the string, you might want to use the other as the literal

```
console.log('Say "Hello"');
console.log("Say 'Hello'");
```

  - You can also use escaped characters with a backslash

```
console.log("It's \"game\" time.");
console.log('It\'s "game" time.');
```

When to Use Double or Single Quotes in JavaScript
http://stackoverflow.com/questions/242813/when-to-use-double-or-single-quotes-in-javascript

Operators

Equality and Truthiness

3.10

- What do these expressions evaluate to?

```
console.log("5" == 5);
console.log(0 == "");
console.log(null == "");
console.log(0 == false);
console.log(undefined == false);
console.log(undefined != true);
console.log("\r\n" == 0);
```

```
// => true
// => true
// => false
// => true
// => false
// => true
// => true
```

- Use === to check equality of value and type

```
if (5 === 5) {   // => true
if ('5' === 5) { // => false
if (0 === '') {  // => false
```

  - Always use === and !== unless you are sure you only want to check for the "truthiness" of an expression

Operators
Floating Point Arithmetic

3.11

- What gets output?

```
if (1 + 2 === 3) console.log("A"); else console.log("B");
if (0.1 + 0.2 === 0.3) console.log("A"); else console.log("B");
```

```
// => A
// => B
```

```
console.log(0.1 + 0.2);
```

```
// => 0.30000000000000004
```

- Almost every language has a floating-point datatype
  - Unlike *integral* (whole) numbers, squeezing *real* numbers into a finite number of bits can require an approximate representation

32	16	8	4	2	1	½	¼	1/8	1/16	1/32
0	0	0	0	1	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1

Decimal	Binary
3	000011
20	010100
2.5	000010.10000
0.1	0.0001100110011…

What Every Computer Scientist Should Know About Floating-Point Arithmetic
http://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html

Operators
How to Deal with Floating Point Numbers

3.12

- Floating point math is based on the IEEE 754 standard
  - JavaScript uses 64-bit floating point representation, which is the same as C#'s double (and .NET's System.Double)
- You must never compare floats and double with equals
  - Instead compare the absolute value of their differences and make sure that this difference is as small as possible

```
var x = 0.1 + 0.2;
if (x === 0.3) // never use this with real numbers!
{
    console.log("Bad code!");
}
```

```
var x = 0.1 + 0.2;
var difference = Math.abs(x - 0.3);
if (difference < 0.00000001)
{
    console.log("Better code!");
}
```

MIM-104 Patriot, Failure at Dhahran
http://en.wikipedia.org/wiki/MIM-104_Patriot#Failure_at_Dhahran

Operators
Dividing by Zero

3.13

- Dividing by zero does not throw an error
- Instead it returns a special value: Infinity

```
var a = 5;
var b = 0;
var c = a / b;
console.log(c); // => Infinity
if (c === Infinity) {
    console.log("The infinity of the Universe is limited.");
}
```

- There is also a value of -Infinity!

Operators
Rounding Numbers

3.14

- What does the following output in C#?

```
ListBox1.Items.Add("Math.Round(-0.5) = " + Math.Round(-0.5));
ListBox1.Items.Add("Math.Round(0.5) = " + Math.Round(0.5));
ListBox1.Items.Add("Math.Round(2.5) = " + Math.Round(2.5));
ListBox1.Items.Add("Math.Round(3.5) = " + Math.Round(3.5));
```

```
Math.Round(-0.5) = 0
Math.Round(0.5) = 0
Math.Round(2.5) = 2
Math.Round(3.5) = 4
```

  - This is called “Bankers’ Rounding” which produces better statistical results because it doesn’t have a bias

- What does the same code output in JavaScript?

```
console.log("Math.round(-0.5): " + Math.round(-0.5));
console.log("Math.round(0.5): " + Math.round(0.5));
console.log("Math.round(2.5): " + Math.round(2.5));
console.log("Math.round(3.5): " + Math.round(3.5));
```

```
Math.round(-0.5): 0
Math.round(0.5): 1
Math.round(2.5): 3
Math.round(3.5): 4
```

Gaussian/Banker's Rounding in Javascript
http://stackoverflow.com/questions/3108986/gaussian-bankers-rounding-in-javascript

What's the strangest corner case you've seen in C# or .NET?
http://stackoverflow.com/questions/194484/whats-the-strangest-corner-case-youve-seen-in-c-sharp-or-net

Operators
Null-Coalescing Operator

3.15

- Often you need to check the validity of an object before using it, commonly using an if statement

```
if (x) {        // if x is usable, i.e. NOT undefined,
  answer = x;   // null, false, etc., then use it
} else {        // otherwise, use some default value
  answer = someDefault;
}
```

- A simpler way is to use the || operator which is equivalent to the ?? operator in C#

```
answer = x || someDefault;
```

- Note: the || operator is useful for creating a namespace if it doesn't already exist (see Module 7)

Operators
Bitwise Operators

3.16

- Bitwise operators work on the *bits* of a 32-bit number
  - & (and), | (or), ~ (not), ^ (xor)

```
var expr1 = 9;                 //  9 is 00001001
var expr2 = 5;                 //  5 is 00000101
var andResult = expr1 & expr2; //  1 is 00000001
var orResult =  expr1 | expr2; // 13 is 00001101
var xorResult = expr1 ^ expr2; // 12 is 00001100
```

  - << (left shift), >> (right shift), >>> (unsigned right shift)

```
var expr1 = 9 << 2;
var expr2 = 9 >> 2;
console.log(expr1); // => 36
console.log(expr2); // => 2
```

```
//      128 64 32 16  8  4  2  1
// 9 =    0  0  0  0  1  0  0  1
// 36 =   0  0  1  0  0  1  0  0
// 2  =   0  0  0  0  0  0  1  0
```

Bitwise Left Shift Operator (<<) (JavaScript)
http://msdn.microsoft.com/en-us/library/ie/t7f48wx9(v=vs.94).aspx

Statements
if

3.17

- Although braces { } are not required for single line **if** and **for** statements, it is strongly recommended that you always use them to avoid serious bugs such as the notorious “goto fail” bug

```
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen) {
          OSStatus        err;
          ...
          if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
                    goto fail;
          if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
                    goto fail;
                    goto fail;
          if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
                    goto fail;
          ...
fail:
          SSLFreeBuffer(&signedHashes);
          SSLFreeBuffer(&hashCtx);
          return err;
}
```

Apple's SSL/TLS bug (22 Feb 2014)
https://www.imperialviolet.org/2014/02/22/applebug.html

Statements
for loops

3.18

- for (uses an initializer, condition, and incrementer)

```
for(initializer ; condition ; incrementer)
  statement; or { block }
```

- Equivalent to while like this

```
initializer;    // happens once
while(condition) {  // checked every time round the loop
  statement
  incrementer;   // executed every time round the loop
}
```

- for…in (uses a “key” variable)
  - Only shows *enumerable* properties

```
for (var variable in object)
  statement; or { block }
```

```
for (var p in o) {   // assign property names of o to variable p
  console.log(o[p]); // print the value of each property
}
```

for…in is NOT like foreach in C#; it doesn’t use IEnumerable

for loop with multiple counters

- Some-times multiple variables change with each iteration of the loop
  - This is the only place that the comma operator is commonly used in JavaScript
  - It provides a way to combine multiple initialization and increment expressions into a single expression suitable for use in a for loop

```
var i, j, sum = 0;
for (i = 0, j = 10 ; i < 10 ; i++, j--) {
  sum += i * j;
}
console.log(sum); // => 165
```

Built-In Types

- JavaScript only has six “types” (ECMAScript 5)
  - **undefined**, **null**, **boolean**, **number**, **string**, **object**
  - Variables do NOT have types but the *values* in them do
  - Use **typeof** operator allows you to detect the type

```
var i = 42;
console.log(typeof i); // => number
var o = { x: 1 };
console.log(typeof o); // => object
```

```
var n = null;
console.log(typeof n); // => object (bug!)
console.log((n === null) ? "null" : typeof n); // => null
```

There is a bug in JavaScript that makes the **typeof** operator return “object” instead of “null” so you must use the conditional operator to check explicitly for nulls!

Types
Built-In Sub-Types

3.21

- Function, Array, Date, window, RegExp, and your own custom objects are “sub-types”
  - You must write your own helper function to detect the sub-type

```
function getSubtype(o) {
  if (o === null) return "Null";
  if (o === undefined) return "Undefined";
  return Object.prototype.toString.call(o).slice(8, -1);
}
```

`[object subtype]`

```
console.log(getSubtype(null));       // => Null
console.log(getSubtype(1));          // => Number
console.log(getSubtype(""));         // => String
console.log(getSubtype(false));      // => Boolean
console.log(getSubtype({}));         // => Object
console.log(getSubtype([]));         // => Array
console.log(getSubtype(/./));        // => Regexp
console.log(getSubtype(new Date())); // => Date
console.log(getSubtype(window));     // => Window
```

Types
Duck Typing

3.22

- Objects in JavaScript are duck types
  - Objects have properties and are dynamically typed (loose-type), unlike .NET which is statically typed (strong-type at compile)
- JavaScript is called a “duck typing” language

A “duck” object

Property	Value
waddle	function
quack	function

Is this a “duck”?

Property	Value
account	“Bob”
waddle	function
balance	£546.87
quack	function
…	

“When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.”

Duck typing
http://en.wikipedia.org/wiki/Duck_typing

Types
Arrays

- To create an array

```
var muppets = new Array(); // create an empty array
muppets[0] = "Kermit";     // resize array and assign value
muppets[1] = "Fozzie";
muppets[4] = "Miss Piggy"; // [2], [3] will be undefined
```

```
var muppets = [ "Kermit", "Fozzie", "Miss Piggy" ]; // array literal
```

- To combine two arrays into a new array

```
var group1 = ["Cecilie", "Lone"];
var group2 = ["Emil", "Tobias", "Linus"];
var combined = group1.concat(group2);
```

- To combine array items into a comma-separated string

```
console.log(combined.join()); // => "Cecilie,Lone,Emil,Tobias,Linus"
```

JavaScript Array Object
http://www.w3schools.com/jsref/jsref_obj_array.asp

Types
Enumerating Arrays

- Arrays have a foreach method

```
// Define the callback function.
function ShowResults(value, index, ar) {
    document.write("value: " + value);
    document.write(" index: " + index);
    document.write("<br />");
}
```

```
// Create an array.
var letters = ['ab', 'cd', 'ef'];

// Call the ShowResults callback function for each
// array element.
letters.forEach(ShowResults);

// Output:
//  value: ab index: 0
//  value: cd index: 1
//  value: ef index: 2
```

forEach Method (Array) (JavaScript)
https://msdn.microsoft.com/library/ff679980(v=vs.94).aspx

Types
Arrays as Stacks and Queues

3.25

- To make an array act like a stack or a queue

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var newLength = fruits.push("Kiwi");  // => 5 (adds to top/end)
console.log(fruits.join());           // => "Banana,Orange,Apple,Mango,Kiwi"
var lastFruitAdded = fruits.pop();    // => Kiwi (removes from top/end)
fruits.reverse();
var firstFruit = fruits.shift();      // => Mango (removes from start/bottom)
newLength = fruits.unshift("Cherry"); // => 4 (adds to start/bottom)
fruits.sort();
console.log(fruits.join());           // => Apple,Banana,Cherry,Orange
```

JavaScript	C# equivalent
Array.push	Stack.Push/Queue.Enqueue
Array.pop	Stack.Pop
Array.shift	Queue.Dequeue
Array.unshift	No equivalent
Array[Array.length-1]	Stack.Peek
Array[0]	Queue.Peek

Functions
Basics

3.26

- Named functions with and without arguments

```
function sum(a, b) {
  console.log(a + b);
}
sum(10, 20); // => 30
```

```
function shout() {
  console.log("boo!");
}
shout(); // => boo!
```

- Anonymous functions do not have names
  - Useful in situations like event handlers where the function will not be used anywhere else

```
element.addEventListener('click', function (e) {
  console.log("You clicked " + e.target.id);
}, false);
```

- For any function you must use
  - `function` keyword and braces `{}` around code statements
  - Parentheses `()` around parameters, even if there aren't any

Functions
Arguments

3.27

- Any function can have any number of arguments and can have any number of parameters passed to it

```
function alpha(name, age) {
  console.log(arguments.length);
  console.log(arguments[0]);
  console.log(name);
}
```

```
alpha();
// => 0
// => undefined
// => undefined
```

```
alpha("Bob");
// => 1
// => Bob
// => Bob
```

```
function beta() {
  console.log(arguments.length);
  console.log(arguments[0]);
  console.log(name);
  console.log(arguments[1]);
}
```

```
beta("Bob", 23);
// => 2
// => Bob
// => undefined
// => 23
```

- The length property of a function tells us the *arity* (number of expected arguments) of a function

```
console.log(alpha.length); // => 2
```

```
console.log(beta.length); // => 0
```

Functions
Overloading

3.28

- JavaScript functions don't have typed signatures which means they can't do overloading

```
function sayMessage(message) {
    console.log(message);
}
function sayMessage() {
    console.log("Default message");
}
```

```
// what happens?
sayMessage("Hello!");
```

```
// => Default message
```

  - When you define multiple functions with the same name, the one that appears last in your code replaces any previous ones

- Check for named arguments to simulate overloading

```
function sayMessage(message) {
    if (message === undefined) console.log("Default message");
    else console.log(message);
}
sayMessage("Hello!"); // => Hello
sayMessage(); // => Default message
```

Referencing and Executing

- You will often need to pass a *reference* to a function
  - When assigning an event handler
  - When assigning a callback

```
function f() { return "test"; }
```

- For a named function, pass its name

```
// assigns a reference to the function
element.onclick = f;
element.addEventListener("click", f, false);
$.get("/api/value", f);
```

- If you use parentheses, you are *executing* the function and assigning whatever it returns as the reference

```
// assigns the return value of the function
element.onclick = f();
element.addEventListener("click", f(), false);
$.get("/api/value", f());
```

Returning Values

- If a function does NOT return a value, it will assign **undefined** to an existing variable

```
function doesNotReturnAnything() {

}

var x = 10;
x = doesNotReturnAnything();
console.log(x); // => undefined
```

Dynamically Defining Functions

- Functions are usually defined with function keyword

```
var f = function (x, y) { return x * y; };
```

- Functions can also be defined with Function constructor (they will be anonymous)
  - Expects any number of string arguments
  - The last argument is the function body, others are arguments

```
var f = new Function("x", "y", "return x * y;");
```

- Always defined as if they were top-level functions i.e. with global scope

Don't Be Eval!

- The eval() method evaluates JavaScript code represented as a string…

```
var x = 10;
var y = 20;
var a = eval("x * y"); // => 200
```

- …but avoid it because
  - Improper use of eval opens up your code for injection attacks
  - Debugging can be more challenging (no line numbers, etc.)
  - eval'd code executes more slowly (no opportunity to compile/cache eval'd code)

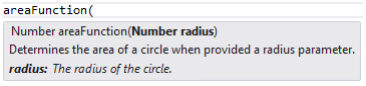
eval()
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval

Why is using the JavaScript eval function a bad idea?
http://stackoverflow.com/questions/86513/why-is-using-the-javascript-eval-function-a-bad-idea

Functions
Documenting for IntelliSense

- Visual Studio looks for special XML comments to provide improved IntelliSense when using libraries

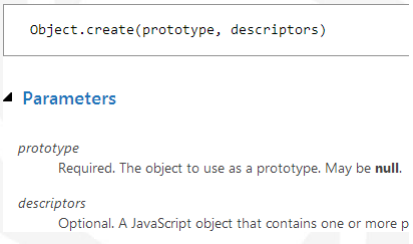
```
function areaFunction(radius) {
  /// <summary>Determines the area of a circle when provided a
  /// radius parameter.</summary>
  /// <param name="radius" type="Number">
  /// The radius of the circle.</param>
  /// <returns type="Number">The area.</returns>
  var areaVal;
  areaVal = Math.PI * radiusParam * radiusParam;
  return areaVal;
}
```



XML Documentation Comments (JavaScript)
http://msdn.microsoft.com/en-us/library/vstudio/hh524453.aspx

Functions
Understanding Parameter Documentation

- Here is some documentation for a JavaScript function
  - Note that because JavaScript is very flexible with passing arguments, the documentation describes which parameters are required or optional



- Another example
  - Note that some of the arguments are optional (those in brackets for example **[, data]**), even in the middle of an argument list!

Functions
Understanding Parameter Documentation

- In this call, the second parameter is the data

```
$.get("api/product", dataToSend, function (result) { });
```

- In this call, the second parameter is the success callback function

```
$.get("api/product", function (result) { });
```

  - It all depends how smart the function is when checking the parameter types

jQuery.get(url [, data] [, success(data, textStatus, jqXHR)] [, dataType])

url
Type: String
A string containing the URL to which the request is sent.

data
Type: PlainObject or String
A plain object or string that is sent to the server with the request.

success(data, textStatus, jqXHR)
Type: Function()
A callback function that is executed if the request succeeds.

Functions
Passing Parameters by Value and by Reference

- Numbers and Strings are passed by value
- Objects and Arrays are passed by reference

```
function processString(input) {
  input = "2";
}

function processNumber(input) {
  input = 2;
}

function processObject(input) {
  input.p = 2;
}

function processObject(input) {
  input = 2; // breaks link
}
```

```
var s = "1";
processString(s); // byval
console.log(s);   // => 1

var n = 1;
processNumber(n); // byval
console.log(n);   // => 1

var o = { p: 1 };
processObject(o); // byref
console.log(o.p); // => 2

var o = { p: 1 };
processObject(o); // byref
console.log(o.p); // => 1
```

this is the Invocation Context

- If you define a variable or function in the global scope in a browser, `this` is the window
  - You do not have to use the `this` prefix unless you need to differentiate with function-scope variables and parameters

```
// this is the window object
function thisExamples(parameter) {
    this.number = 1;
    console.log(this.number);    // => 1
    number = 2; // same as this.number
    console.log(this.number);    // => 2
    console.log(parameter);      // => 4
    this.parameter = 3; // different from parameter
    console.log(this.parameter); // => 3
    console.log(parameter);      // => 4
}
thisExamples(4);
console.log(this.parameter); // => 3
console.log(this.number);    // => 2
```

Controlling the Invocation Context

- Execute a function once with an object bound to `this`
  - **apply**: array of arguments, or **call**: comma-separated arguments
- Execute a function multiple times: **bind**

```
function showPerson(color, size) {
    console.log(this.firstName + " is " + this.age);
    console.log(color + " - " + size);
}
showPerson();
var bob = {
    firstName: "Bob",
    age: 42
};
showPerson.apply(bob, [ "Red", 2.5 ]);
showPerson.call(bob, "Red", 2.5);
var f = showPerson.bind(bob);
f("Red", 2.5);
```

```
undefined is undefined
undefined - undefined
Bob is 42
Red - 2.5
Bob is 42
Red - 2.5
Bob is 42
Red - 2.5
```

Binding a Method to *this*

- How to bind `this` to a method in another object

```
var originalObject = {
  minimum: 50, // some properties
  maximum: 100,
  checkNumericRange: function (numberToCheck) { // a method
    if (typeof numberToCheck !== 'number')
      return false;
    else
      return numberToCheck >= this.minimum &&
             numberToCheck <= this.maximum;
  }
}
var result1 = originalObject.checkNumericRange(15);
console.log(result1); // => false
```

```
var newObject = { minimum: 10, maximum: 20 }; // “duck” type
var newFunction = originalObject.checkNumericRange.bind(newObject);
var result2 = newFunction(15); // does not affect original method
console.log(result2); // => true
```

Referencing an Element

- To reference an element give it a unique **id**
  - If it is *outside* a form, use *id*
  - If it is *inside* a form, use *formid.id*, or use document.getElementById("*id*")

```
<input id="outside" value="A" />
<form id="frm"><input id="inside" value="B" /></form>
```

```
console.log(outside.value);
console.log(frm.inside.value);
console.log(document.getElementById("inside").value);
```

- Or use one of these methods of **document**
  - Returns first match: querySelector(“*cssselector*”)
  - Returns all matches: querySelectorAll(“*cssselector*”), getElementsByClassName(“*class1 class2*”), getElementsByName(“*name*”), getElementsByTagName(“tag*name*”)

Child Nodes

- How many child nodes does *list* have?

```
<ol id="list">
  <li>Apples</li>
  <li>Bananas</li>
</ol>
```

```
5
  [object Text]
  [object HTMLLIElement]
  [object Text]
  [object HTMLLIElement]
  [object Text]
```

- Because of whitespace!

```
console.log(list.childNodes.length);
for (var i = 0; i < list.childNodes.length; i++) {
    console.log("  " + list.childNodes[i]);
}
```

- Or use children instead ;)

```
for (var i = 0; i < list.children.length; i++) {
    console.log("  " + list.children[i]);
}
```

Getting and Setting an Attribute

- To get an attribute

```
<input id="firstname" value="Alice" data-age="32" required="required" />
```

```
console.log(firstname.value); // => Alice
console.log(firstname.getAttribute("value")); // => Alice
console.log(firstname.required); // => true
console.log(firstname.getAttribute("required")); // => required
console.log(firstname.getAttribute("data-age")); // => 32
```

- To set an attribute

```
firstname.value = "Bob";
firstname.setAttribute("value", "Bob");
firstname.required = true;
firstname.setAttribute("required", "required");
firstname.setAttribute("data-age", "33");
```

classList

- classList returns a token list of the class attribute of the element
  - classList is a convenient alternative to accessing an element's list of classes as a space-delimited string via element.className

```
<div id="kermit" class="foo bar"></div>
```

```
kermit.classList.remove("foo");
kermit.classList.add("anotherclass");
```

```
// if visible is set, then remove it, otherwise add it
kermit.classList.toggle("visible");
alert(kermit.classList.contains("foo"));
kermit.classList.add("foo", "bar"); // add multiple classes
```

element.classList
https://developer.mozilla.org/en-US/docs/Web/API/element.classList

Adding and Removing Event Handlers

- To add an event handler programmatically
  - Internet Explorer 8 and earlier

```
element.attachEvent("onclick", function () { ... });
```

  - Every other browser

```
element.addEventListener("click", function () { ... }, false);
```

  - All browsers

```
element.onclick = function () { ... };
```

- To remove an event handler, use a named function

```
function doStuff() { ... }
```

```
element.addEventListener("click", doStuff, false);
```

```
element.removeEventListener("click", doStuff, false);
```

Multicast Events

- To add multiple handlers for an event you must use addEventListener, not set the on*event* property

```
<input type="button" id="bob" value="Bob" />
```

```
<script>
    bob.addEventListener('click', A); // assigns a handler
    bob.addEventListener('click', B); // assigns another handler

    function A() {
        alert('A');
    }

    function B() {
        alert('B');
    }
</script>
```

```
bob.onclick = A; // assigns a handler
bob.onclick = B; // re-assigns the handler!
```

document and window Events

- Which event happens earliest?
  - document's **DOMContentLoaded** event fires earliest, when parsing of the current page is complete; use it to hookup UI functionality to complex web pages
  - window's **load** event fires later, when all files have finished loading from all resources, including ads and images

```
// a custom function to hide IE8 and earlier differences
function myEventAdder(element, eventName, listener) {
  if (element.addEventListener) {
    element.addEventListener(eventName, listener, false);
  } else { // IE 8 and earlier
    element.attachEvent("on" + eventName, listener);
  }
}
```

```
myEventAdder(document, "DOMContentLoaded", finishedDCL);
myEventAdder(window, "load", finishedLoad);
```

DOMContentLoaded
http://ie.microsoft.com/testdrive/HTML5/DOMContentLoaded/Default.html

Event Order (Bubbling and Capturing)

- If an element and one of its ancestors have an event handler for the same event, which one should fire first?
  - Bubbling means B, then A (default)
  - Capturing means A, then B
  - W3C supports both: capturing happens first

```
<div onclick="alert('A');">
  <div onclick="alert('B');">
    Click Me
  </div>
</div>
```

- addEventListener allows you to control which to use
  - 3rd parameter: true means capturing, false means bubbling
  - Warning! Old Internet Explorer versions only support bubbling which is why jQuery doesn't support capturing either!

```
elem.addEventListener('eventName', eventHandler, true);  // use capturing
elem.addEventListener('eventName', eventHandler, false); // use bubbling
elem.addEventListener('eventName', eventHandler);        // use bubbling
```

Event order
http://www.quirksmode.org/js/events_order.html

Event Object

- When an event occurs the first parameter of the event handler function is a reference to the *event object*

```
function eventHandler(e) {
    console.log(e.currentTarget);
```

- Alternatively, use the **event** keyword

```
function eventHandler() {
    console.log(event.currentTarget);
```

- The event object has some useful properties
  - eventPhase: one of the event phase constants:
    - CAPTURING_PHASE (1), AT_TARGET (2), BUBBLING_PHASE (3)
  - currentTarget: the element that *handled* the event
  - target: the element that *triggered* the event

stopPropagation() and preventDefault()

preventDefault()

- Cancels any default action normally taken, for example, navigating to another page when clicking on a link

```
msLink.onclick = function (e) {
    e.preventDefault();
    alert('Microsoft is ace!')
};
```

Note: returning false from within a jQuery event handler is effectively the same as calling both e.preventDefault and e.stopPropagation on the passed jQuery.Event object

```
<a id="msLink" href="http://www.microsoft.com/">Microsoft</a>
```

Note: event.returnValue has been deprecated but if set to false would also prevent the default action

stopPropagation()

- Cancels further capturing/bubbling by other event handlers

event.preventDefault() vs. return false
http://stackoverflow.com/questions/1357118/event-preventdefault-vs-return-false

returnValue property
https://msdn.microsoft.com/en-us/library/ie/ms534372(v=vs.85).aspx

event.returnValue is deprecated. Please use the standard event.preventDefault() instead
http://stackoverflow.com/questions/20045162/event-returnvalue-is-deprecated-please-use-the-standard-event-preventdefault

Event Object for Mouse and Keyboard Events

Additional properties are available on the event object

- Keyboard-related
  - keyIdentifier: the identifier of a key
  - keyLocation: the location of the key on the device
  - altKey: was the ALT key was pressed
  - ctrlKey: was the CTRL key pressed
  - shiftKey: was the SHIFT key pressed
  - metaKey: was the META key pressed
- Mouse-related
  - button: which mouse button was clicked
  - clientX, clientY: the horizontal and vertical coordinates of the mouse pointer, relative to the current window
  - screenX, screenY: the horizontal and vertical coordinates of the mouse pointer, relative to the screen

this in Event Handlers

- `this` in JavaScript is very special and powerful—it can mean just about anything
  - For event handlers `this` refers to the DOM element that's the subject (owner) of the function being called

```
$("div").click(function () {
  // this will be the DOM element for the div that was clicked,
  // so you could (for instance) set its foreground color:
  this.style.color = "red";
});
```

this in Event Handlers

- What does `this` refer to in doSomething?

```
function doSomething() {
  alert('clicked: ' + this);
}
```

  - In JavaScript `this` always refers to the "owner" of the function, usually the object that a function is a method of
    - In the example above, *window* is the owner of doSomething()
  - An onclick property, though, is owned by the HTML element it belongs to, so if we do this, `this` now refers to the element

```
element.onclick = doSomething; // reassigns 'this' to the element
```

  - BUT inline event handling doesn't have the same effect!

```
<h2 onclick="doSomething();">
```

```
<!-- does nothing! -->
<h2 onclick="doSomething">
```

The this keyword
http://www.quirksmode.org/js/this.html

DOM and Events
this in Event Handlers

3.53

```
function doSomething() {
  alert('clicked: ' + this);
}
```

- Examples when reassignment of `this` happens

```
element.onclick = doSomething;
element.addEventListener('click', doSomething, false);
element.onclick = function () { this.style.color = '#cc0000'; }
```

```
<element onclick="this.style.color='#cc0000';">
```

- Examples when reassignment does NOT happen

```
element.onclick = function () { doSomething(); }
element.attachEvent('onclick', doSomething); // IE8 and earlier
```

```
<element onclick="doSomething();">
```

- Avoid the problem by explicitly passing this

```
function doSomething2(element) {
  // element now refers to the HTML element
  element.style.color = '#cc0000';
}
```

```
<h2 onclick="doSomething2(this);">Click Me</h2>
```

```
<h2 onclick="doSomething.call(this);">
    Click Me</h2>
```

Errors
Three Common Ways of Indicating an Error

3.54

- Some objects raise "error" events
  - Write a handler and check the event parameter's properties
- Some functions return objects with error properties
  - For example, when making an async call, the returned object may have properties that indicate standard HTTP status codes:
    - 200 OK, 404 Not Found, 500 Internal Server Error, and so on
- Sometimes an Error object is thrown
  - Use a try/catch block

```
"use strict";
try {
  x = 10; // undefined variable in strict mode
}
catch (error) {
  alert("Error " + error.number + ": " + error.message);
}
```

The number property is Microsoft-only!

Errors
Catching and Throwing Errors

```
try {
  try {
    console.log("Nested try running...");
    // to throw a custom error pass unique number and string (IE only!)
    throw new Error(301, "an error message from IE");
    // to throw a custom error pass string (all other browsers)
    throw new Error("an error message from other browsers");
  }
  catch (error) { // number, description, message properties
    if (error.number === 301) console.log("my error");
    console.log("Nested catch " + error.message);
    throw error; // re-throw
  }
}
catch (e) { // required
  console.log("Outer catch " + e.message);
}
finally { // optional
  console.log("Outer finally running");
}
```

The description property provides backward compatibility; the message property complies with the ECMA standard

try...catch...finally Statement (JavaScript)
http://msdn.microsoft.com/en-us/library/ie/4yahc5d8(v=vs.94).aspx

Errors
Error Types

- Besides the Error constructor, there are other Error-derived constructors in JavaScript
  - EvalError, InternalError, RangeError, ReferenceError, SyntaxError, TypeError, URIError
- Instead of error numbers, non-Microsoft browsers use the above types or name property to determine the type of error that has occurred

```
catch (error) {
  if (error instanceof RangeError) {
    alert("Number out of range!");
  } else {
    alert(error.name + ": " + error.message);
  }
}
```

Error
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Error

JSON
Comparing OLN and JSON

3.57

- Object Literal Notation (OLN)
  - Putting property names in quotes does NOT mean you are using JSON
  - MOC is wrong on page 3-12 (position 5, 3997)

```
var point = { x: 5, y: 12 };
var book = {
  "main title": "Exam 70-480",
  subtitle: "Programming with HTML5",
  "for": "Beginners"
};
```

- JavaScript Object Notation (JSON) is when a JavaScript object is serialized into a string using JSON.stringify()

```
o = { x: 1, y: { z: [false, null, ""] } };
s = JSON.stringify(o); // s is '{"x":1,"y":{"z":[false,null,""]}}'
p = JSON.parse(s);     // p is a deep copy of o
```

  - If you parse an object it throws error: “Invalid character”
  - An optional second argument can be used to customize the stringify and parse processes

JSON
Limitations

3.58

- JSON is a subset of OLN
  - Objects, arrays, strings, finite numbers, true, false, and null are fully supported so they can be serialized *and* restored
- Limitations
  - Only the enumerable own properties of an object are serialized
  - NaN, Infinity, and -Infinity are serialized to null
  - Date objects are serialized to ISO-formatted date strings, but JSON.parse() leaves it as a string rather than restoring the Date
  - Function, RegExp, and Error objects and the undefined value cannot be serialized; if a property value cannot be serialized, that property is simply omitted from the stringified output

Dates and Times

How to Get Today's Date

- Although **Date** has a **now**() method it returns the number of milliseconds since 1 January 1970 which isn't very useful!

```
console.log(Date.now()); // => 1439381488570
```

- Create a new instance of **Date** to get an object representing the current date and time

```
var today = new Date();
console.log(today); // => Wed Aug 12 2015 13:09:44 GMT+0100 (GMT Summer Time)
console.log(today.getYear()); // => 115
console.log(today.getFullYear()); // => 2015
console.log(today.getMonth()); // => 7 (7 = August because 0 = January!)
console.log(today.getHours()); // => 13
console.log(today.getMinutes()); // => 9
console.log(today.getSeconds()); // => 44
```

How to get current date in JavaScript?
http://stackoverflow.com/questions/1531093/how-to-get-current-date-in-javascript

jQuery

What is jQuery?

- Created by John Resig, January 2006
  - Light-weight (32kB), CSS3 compliant, cross-browser, feature-rich (DOM manipulation, eventing, Ajax, animation) library
- jQuery function is aliased to use dollar sign

`jQuery` or `$`

- Two major version branches
  - jQuery 1.x (version 1.11.3 on 16th July 2015)
  - jQuery 2.x (version 2.1.4 on 16th July 2015)
  - 2.x has the same API, but does not support Microsoft Internet Explorer 6, 7, or 8 so use the 1.x version unless you are certain IE 6/7/8 users are not visiting your site

jQuery
http://jquery.com/

jQuery
…and Visual Studio 2012 RTM

3.61

- 70-480 covers jQuery 1.7.1
  - Because that is the version included with Visual Studio 2012 RTM →
  - The exam only covers the core of jQuery not any of its extensions such as jQuery UI
  - To install a specific version of jQuery use Tools-NuGet Package Manager-Package Manager Console

```
install-package jquery –version 1.7.1
```

- …or use a Content Delivery Network (CDN) instead

```
http://code.jquery.com/jquery-latest.js
```

Warning! Some 1.7.1 APIs are deprecated but might still be in the exam

jQuery
Enabling IntelliSense

3.62

- When writing JavaScript in a separate file it is useful to manually add a reference to libraries such as jQuery to get IntelliSense
  - Drag and drop the jQuery library to the top of a .js file

```
/// <reference path="jquery-1.10.2.js" />
```

  - Or add a _references.js file to the Scripts folder

```
/// <autosync enabled="true" />
/// <reference path="modernizr-2.6.2.js" />
/// <reference path="jquery-1.10.2.js" />
/// <reference path="bootstrap.js" />
/// <reference path="respond.js" />
/// <reference path="jquery.validate.js" />
/// <reference path="jquery.validate.unobtrusive.js" />
/// <reference path="site.js" />
```

jQuery
Basic Selectors and jQuery Extensions

- jQuery selectors are the same as CSS selectors
  - $('tag'), $('#id'), $('.class'), $('#id tag.class tag')

```
$('#peopleTable tr.cool td')
```

- ...and jQuery adds some useful extra selectors
  - For example, to make all h1, h2, h3, and so on green

```
$(':header').css({ color: 'green' });
```

  - To make all buttons red

```
$(':button').css({ color: 'red' }); // easy way
```

```
$("button, input[type='button']").css({ color: 'red' });
```

Basic Selectors
http://api.jquery.com/category/selectors/basic-css-selectors/

jQuery Extensions
http://api.jquery.com/category/selectors/jquery-selector-extensions/

jQuery
Attribute Selectors

- Search characters in HTML attributes with [...]
  - *= searches for the term in all text

```
$('a[href*=firebrand]')
```

  - ~= searches for the *word* (term delimited by spaces) in all text
  - ^= searches for the term at the beginning
  - $= searches for the term at the end

```
$('a[href$=.com]')
```

  - != searches for non-matches
- Logical operations (AND, OR)
  - [criteria][criteria] multiple attributes must match
  - [criteria],[criteria] either attribute must match

Attribute Selectors
http://api.jquery.com/category/selectors/attribute-selectors/

jQuery
Basic, Form, and Child Filters

3.65

- Basic
  - :odd, :even, :first, :last, :not(...), contains(...), and so on
  - Note: tr counting starts from 1, so odd would apply to the first row

```
$('span:first')
```

```
$('tr:odd')
```

- Form
  - :button, :textbox, :selected, :checked, and so on

```
$(':checked')
```

- Child
  - :first-child, :last-child, :nth-child(n), and so on

```
$("ul li:nth-child(4)")
```

```
$("ul li:nth-child(3n + 2)")
```

Basic Filter
http://api.jquery.com/category/selectors/basic-filter-selectors/

Form
http://api.jquery.com/category/selectors/form-selectors/

Child Filter
http://api.jquery.com/category/selectors/child-filter-selectors/

jQuery
Hierarchical

3.66

- Descendant selector (“ancestor descendant”)
  - All elements that are descendants of a given ancestor

```
$("div p")
```

- Child selector (“parent > child”)
  - Only direct child elements

```
$("div > p")
```

- Next Adjacent selector (“prev + next”)
  - All elements that are *immediately preceded* by a sibling “prev”

```
<label>Name:</label>
<input name="name" />
<fieldset>
  <label>Newsletter:</label>
  <input name="newsletter" />
  <span>Active:</span>
  <input name="active" />
```

```
$("label + input").css("color", "blue");
```

Hierarchy
http://api.jquery.com/category/selectors/hierarchy-selectors/

jQuery
Hierarchical

3.67

- Next Siblings (“prev ~ siblings”)
  - All sibling elements that follow after the “prev” element, have the same parent, and match the filtering “siblings” selector

```
$("label ~ input").css("color", "blue");
```

```
<label>Name:</label>
<input name="name" />
<fieldset>
  <label>Newsletter:</label>
  <input name="newsletter" />
  <input name="active" />
```

jQuery
Traversing

3.68

- .children(['selector'])

```
$("div").children() // => an array
```

  - All children
  - All children that match selector

```
$("div").children(".cool")
```

- .closest('selector')

```
$("td.cool").closest("tr")
```

  - gets parent elements
- .find('selector')

```
$("div").first()    // => an array
$("div").first()[0] // => 1st item
```

  - gets child elements
- .first(), .last(), .next(), .prev(), .parent(), .siblings()
  - Warning! Most jQuery functions return arrays, but some of them only contain one item, for example, first(), last() and so on

Traversing
http://api.jquery.com/category/traversing/

jQuery
Applying Actions and Styles

- To apply a css style or set an attribute value
  - .css('attribute', 'value'), .attr('attribute', 'value')

```
$('#a1').css('color', 'blue').attr('target', '_blank');
```

- To apply action to each item in returned results
  - .each(function(index)): index starts from 0 (zero)

```
$("h2").each(function (index) {
  this.innerHTML = "Section " + index;
});
```

- To get an indexed item or count of items in selection

```
$('span').get(0)
$('span')[0]
```

```
var items = $('h2').size(); // deprecated since 1.8
items = $('h2').length;
```

Utilities
http://api.jquery.com/category/utilities/

CSS
http://api.jquery.com/category/css/

jQuery
Event Handlers

- To wait until the DOM is fully loaded before executing your JavaScript code

```
$(document).ready(function () { /* ... */ });
```

```
$(function () { /* ... */ }); // newer syntax
```

- To handle events [deprecated since 1.7]
  - .bind('event', function) and .unbind()
  - .click(function), .dblclick(function), .blur(function), .hover(function, .keydown(function), .keyup(function), .mouseover(function), .mouseout(function), and so on
- To toggle between handlers on each click
  - .toggle(function, function, ...)

Events
http://api.jquery.com/category/events/

.ready()
http://api.jquery.com/ready/

jQuery
on and off

3.71

- Attaches event handlers to the selected elements
  - As of jQuery 1.7, the .on() method provides all functionality required for attaching event handlers

```
$("#bob").on("click", function () {
    alert($(this).text());
});
```

```
function notify() {
    alert("clicked");
}
$("#bob").on("click", notify);
```

  - For help in converting from older jQuery event methods, see .bind(), .delegate(), and .live() (all now deprecated)
- To attach an event that runs only once and then removes itself, see .one()
- .off() removes event handlers previously attached with .on()

```
$("#bob").off("click", notify);
```

```
$("#bob").off("click");
```

.on()
http://api.jquery.com/on/

.one()
http://api.jquery.com/one/

.off()
http://api.jquery.com/off/

jQuery
Adding Elements and Content

3.72

```
<div id="mydiv">
    <span>Alpha</span>
</div>
```

- To add new elements as children to an existing element

```
$('#mydiv').append('<span>Beta</span>');
```

```
<div id="mydiv">
    <span>Alpha</span>
    <span>Beta</span>
</div>
```

- To *replace* an existing element

```
$('#mydiv').replaceWith('<span>Beta</span>');
```

```
<span>Beta</span>
```

- To replace the inner *HTML* inside an element

```
$('#mydiv').html('<span>Beta</span>');
```

```
<div id="mydiv">
    <span>Beta</span>
</div>
```

.replaceWith()
http://api.jquery.com/replaceWith/

.html()
http://api.jquery.com/html/

jQuery
Wrapping DOM Elements

3.73

- Often you have an ordinary element and you want to use a jQuery feature

```
$(element).hide();
```

Misc
What Is RequireJS?

3.74

Exam Topic: none

- RequireJS is a JavaScript file and module loader
  - Using a modular script loader like RequireJS will improve the speed and quality of your code
- Download and copy require.js into your scripts folder
- Add the following <script> block

```
<!-- data-main attribute tells require.js to load
          scripts/main.js after require.js loads. -->
<script data-main="scripts/main" src="scripts/require.js"></script>
```

- Inside main.js use requirejs() to load any other scripts you need to run

```
requirejs(["helper/util"], function (util) {
```

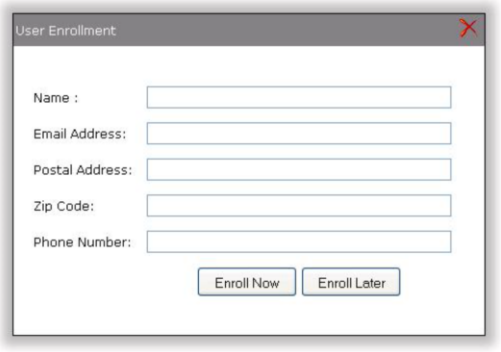
RequireJS
http://requirejs.org/

Misc
Modal Popups

3.75

Exam Topic: none

- There are many JQuery based Modal Popup Windows
  - This developer decided to create his own Modal Popup window without using JQuery which is easy to use and flexible



JavaScript Modal Popup Window
http://www.codeproject.com/Tips/589445/JavaScript-Modal-Popup-Window

Misc
Polyfills

3.76

Exam Topic: none

- A piece of code (or plugin) that provides the technology that the developer expects the browser to have
  - For example, classList is a relatively new feature which may not be available in older browsers and Internet Explorer <= 9 but you can add the same ability to older browsers using classList.js

eligrey/classList.js on GitHub
https://github.com/eligrey/classList.js

What is a Polyfill?
https://remysharp.com/2010/10/08/what-is-a-polyfill

HTML5 Cross Browser Polyfills
https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills

Polyfills for HTML5 Features
http://html5please.com/#polyfill

Misc

JavaScript Unit Testing

3.77

Exam Topic: none

- One of the problems with unit test JavaScript is that your code is often mixed with HTML and appears on both the client and server
- You should start by refactoring your code as much as possible and use libraries that support “unobtrusive” JavaScript
- Read this article for more details…

Introduction To JavaScript Unit Testing
http://coding.smashingmagazine.com/2012/06/27/introduction-to-javascript-unit-testing/

Misc

JavaScript Unit Test Tools for TDD

3.78

Exam Topic: none

- There are many test tools for Test-Driven Development with JavaScript
- JsUnit seems to be the best option, but it is not perfect because
  - It does not provide a simple and integrated way to run JavaScript unit test
  - It forces you to write the unit tests in a html file instead of a .js file
  - It forces you to have a local installation of the JsUnit framework in order to avoid absolute hard coded path to reference js unit files
- Read this StackOverflow discussion for more details…

JavaScript unit test tools for TDD
http://stackoverflow.com/questions/300855/javascript-unit-test-tools-for-tdd

Further Study
Superhero.js

3.79

Exam Topic: none

- A lot of great articles on creating, testing and maintaining large JavaScript applications

UNDERSTANDING JAVASCRIPT

1. Understanding JavaScript Function Invocation and "this" — Yehuda Katz, yehudakatz.com
2. Common JavaScript "Gotchas" — James Fuller, jblotus.com
3. Preparing Yourself for Modern JavaScript Development — Justin Etheredge, codethinked.com
4. Prototypes and Inheritance in JavaScript — Scott Allen, msdn.microsoft.com
5. Style Guide: A mostly reasonable approach to JavaScript — Airbnb, github.com/airbnb
6. COURSE Code School: The JavaScript Path — Code School
7. BOOK Eloquent JavaScript — Marijn Haverbeke
8. BOOK Effective JavaScript — David Herman

Superhero.js
http://superherojs.com/

Module 4
Creating Forms to Collect and Validate User Input

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Creating Forms to Collect and Validate User Input

Contents

Exam Topic: Validate user input by using HTML5 elements
- ❑ Choose the appropriate controls based on requirements
- ❑ Implement HTML input types and content attributes to collect user input

Exam Topic: Validate user input by using JavaScript
- ❑ Evaluate a regular expression to validate the input format
- ❑ Validate that you are getting the right kind of data type by using built-in functions
- ❑ Prevent code injection

Can I use...

= Supported = Not supported = Partially supported = Support unknown

Number input type - **Working Draft**

Form field type for numbers.

*Usage stats: Global

Support: 38.94%
Partial support: 22.25%
Total: 61.19%

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		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
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							

Date/time input types - **Working Draft**

Form field widget to easily allow users to enter dates and/or times, generally by using a calendar widget.

*Usage stats: Global

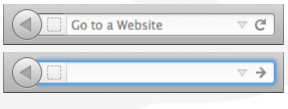
Support: 9.56%
Partial support: 33.31%
Total: 42.87%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
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							

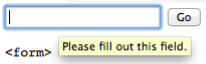
HTML5 Input Types
http://caniuse.com/#search=input

input Element
What Is It?

- <input> elements are used within a <form> element to declare input controls that allow users to input data
  - An input field can vary in many ways, depending on the type attribute
  - Note: The <input> element is empty, it contains attributes only
  - Use the <label> element to define labels for <input> elements
- Useful new attributes
  - placeholder=*text*
  - autocomplete=on|off
  - autofocus=autofocus
  - pattern=*regular_expression*
  - required=required



pattern is not checked if required is not set and the user inputs nothing

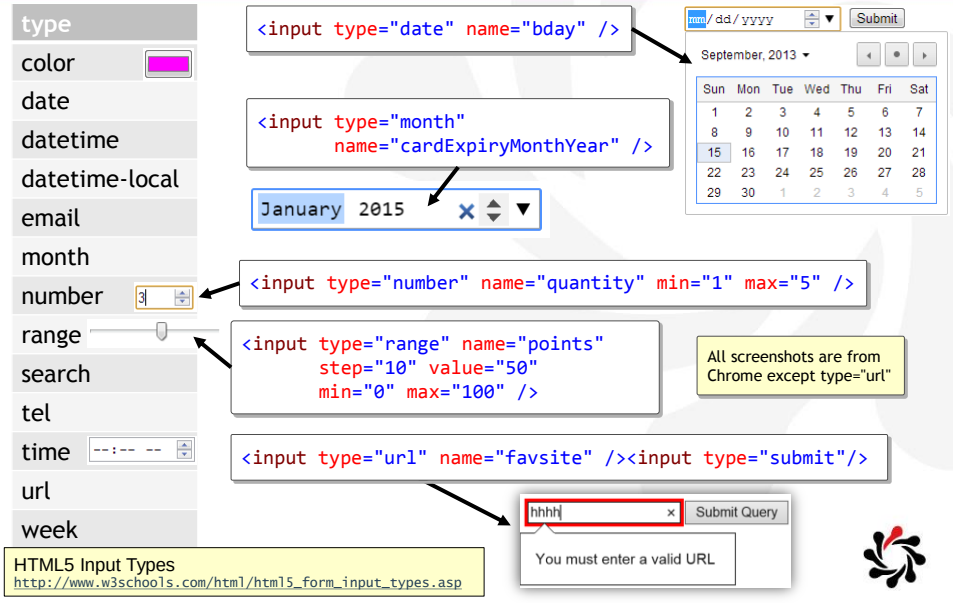



```
<form>
  <input name="q" required>
  <input type="submit" value="Go">
</form>
```

HTML <input> Tag
http://www.w3schools.com/tags/tag_input.asp

input Element
type attribute

4.5



input Element
Detecting HTML5 Input Support

4.6

- Create a dummy <input> element

```
var i = document.createElement("input");
```

- Set the type to the input type you want to detect

```
i.setAttribute("type", "color");
```

- If your browser doesn't support that type it will ignore the value and revert to "text"

```
if(i.type !== "text") {
  // browser supports color!
```

- Modernizr does this for you (and more efficiently)

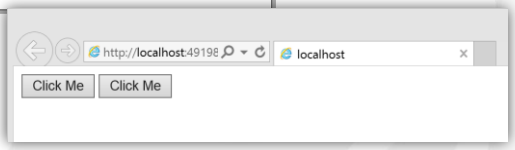
Modernizr
http://modernizr.com/

Other Elements
Buttons

4.7

- Clickable buttons can be created using input or button elements

```
<input type="button" id="button1" value="Click Me" />
<button type="button" id="button2">Click Me</button>
```



- The coolest thing about the <button> tag is that you can put useful HTML elements inside them, like images

Rediscovering the Button Element
http://web.archive.org/web/20110719013452/http://particletree.com/features/rediscovering-the-button-element/

Button tag in bloody Internet Explorer
http://www.peterbe.com/plog/button-tag-in-IE

Other Elements
progress and meter

4.8

- The <progress> tag represents the progress of a task
  - Use the <progress> tag in conjunction with JavaScript to display the progress of a task

```
<progress value="50" max="100" />
```

- The <meter> tag represents a gauge (like temperature)
  - The <progress> tag is not suitable for representing a gauge (e.g. disk space usage or relevance of a query result)

```
<meter min="-20" max="50" value="10" />
```

HTML <progress> Tag
http://www.w3schools.com/tags/tag_progress.asp

Other Elements
output

4.9

- The <output> tag represents the result of a calculation (like one performed by a script)
  - for: Specifies the relationship between the result of the calculation, and the elements used in the calculation
  - form: Specifies one or more forms the output element belongs to
  - name: Specifies a name for the output element

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">
  0<input type="range" id="a" value="75">100
  +<input type="number" id="b" value="25">
  =<output id="x" for="a b"></output>
</form>
```

Result:

0 100 + 25 =100

Note: The output tag is not supported in Internet Explorer.

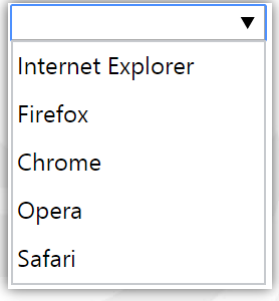
HTML <output> Tag
http://www.w3schools.com/tags/tag_output.asp

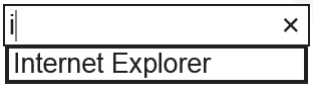
Other Elements
datalist

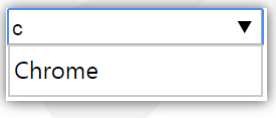
4.10

- Provides an <input> element with a list of pre-defined values

```
<input list="browsers" />
<datalist id="browsers">
    <option value="Internet Explorer" />
    <option value="Firefox" />
    <option value="Chrome" />
    <option value="Opera" />
    <option value="Safari" />
</datalist>
```







HTML <datalist> Tag
http://www.w3schools.com/tags/tag_datalist.asp

Other Elements
Access Keys and Labels

4.11

- Elements can have access keys for short cuts
  - To allow the user to press Alt+F to set the focus to this text box

```
<input id="firstName" accesskey="f" />
```

- To allow the user to press Alt+H to toggle the check box

```
<input id="horiCheckBox" type="checkbox" />
<label for="horiCheckBox" accesskey="h">Horizontal</label>
```

Validation
Common Events

4.12

- Mouse and keyboard events
  - click, dblclick, mousedown, mousemove, mouseover, mouseout, mouseup, keydown, keypress, keyup
  - input event is the best choice for validation because the key*something* events only happen when you use the keyboard so what if the user right-clicks and chooses Paste from the menu?
- Other events
  - blur: when an element loses focus
  - change: when the content of an element, the selection, or the checked state have changed
  - focus: when an element gets focus
  - reset/submit: when a form is reset or submitted
  - select: when a user selects some text

HTML DOM Events
http://www.w3schools.com/jsref/dom_obj_event.asp

Validation
Submit Event for a Form

- The handler for the **submit** event of a form can return **false** to prevent the form being posted to the server
  - Returning true *or not returning anything* allows the submit

```
<form onsubmit="alert('stop submit'); return false;">
```

- Or use a named function

```
function toSubmit() {
    alert('I will not submit');
    return false;
}
```

The statements for onsubmit must "return" a value if you need to prevent submitting

```
<form onsubmit="return toSubmit();">
```

- Warning!
  - When programmatically calling the form.submit() method the onsubmit event handler will NOT be called

How to prevent form from being submitted?
http://stackoverflow.com/questions/3350247/how-to-prevent-form-from-being-submitted

Validation
novalidate

- Indicate that the form is not to be validated on submit
  - This form MUST have a valid email entered before it can be submitted

```
<form action="demo_form.asp">
    E-mail:
    <input type="email" name="user_email" required="required" />
    <input type="submit" />
</form>
```

  - Adding novalidate disables the checks

```
<form action="demo_form.asp" novalidate="novalidate">
    E-mail:
    <input type="email" name="user_email" required="required" />
    <input type="submit" />
</form>
```

HTML <form> novalidate Attribute
http://www.w3schools.com/tags/att_form_novalidate.asp

Validation
Parsing Numbers and Dates

4.15

- To parse a string into a Number

```
var i = parseInt('42');
var f = parseFloat('4.2');
```

- To parse a string into a Date
  - Date.parse method is completely implementation dependent and new Date(string) is equivalent to Date.parse(string), so the recommendation is to write your own function

```
function parseDate(input) {
  var parts = input.split('-');
  // new Date(year, month[, day[, hour[, minute[, second[, ms]]]]])
  return new Date(parts[0], parts[1] - 1, parts[2]); // months are 0-based
}
```

```
var d = parseDate('2012-01-27');
```

- Better to use a library like Moment.js or Datejs

Moment.js
http://momentjs.com/docs/#/parsing/

Datejs
http://www.datejs.com/

Validation
Using jQuery

4.16

- .val()
  - Get the current value of the first element in the set of matched elements or set the value of every matched element
  - Primarily used to get the values of form elements such as input, select and textarea

```
$('select.foo option:selected').val(); // from a multi-select
$('select.foo').val();                 // from a dropdown select
$('#productcode').val();               // from a input type="text"
```

- .text()
  - Get the combined text contents of each element in the set of matched elements, including their descendants, or set the text contents of the matched elements
  - *It cannot be used on form inputs or scripts*

.val()
http://api.jquery.com/val/

.text()
http://api.jquery.com/text/

Regular Expressions
What Are They?

4.17

- Regular expressions are patterns used to match character combinations in strings
  - *RegExp*.**exec**: returns first match value
  - *RegExp*.**test**: returns true if a match
  - *String*.**match**: returns one (or more) matches (if global)
  - *String*.**replace**/**search**/**split**
- You construct a regular expression in one of two ways
  - Constructor (uses a string)

```
var re = new RegExp("ab+c", "g"); // g finds multiple matches
```

  - Literal (compiled for better performance but cannot change)

```
var re = /ab+c/;
```

```
var re = /ab+c/g; // g makes it global
```

Regular Expressions
https://developer.mozilla.org/en/docs/Web/JavaScript/Guide/Regular_Expressions

Regular Expressions
Basics

4.18

- Regular expressions can validate and process text
  - When validating input, include the leading caret and trailing dollar to avoid security vulnerabilities
  - **\d{4}** means four digits, but would also match **DROP table;1234**
  - **^\d{4}$** means only four digits
- {…} is a quantifier, […] is a set (or range) of characters

```
var r1 = /b{2}/g;              // bb (g makes it find multiple matches)
var r2 = /^[abc]{3}$/;         // aaa, aab, abb, bbc, and so on
var r3 = /^[a-e]{2}$/;         // ae, bd, ee, and so on
var r4 = /^[a-zA-Z0-9]{1,5}$/; // one to five alphanumeric chars
```

```
if(r1.test("abbc")) { // => true
```

```
var matches = "abbcbbd".match(r1) { // => array of string matches
```

Regular Expression Library
http://www.regexlib.com/

Regular Expression Basic Syntax Reference
http://www.regular-expressions.info/reference.html

Regular Expressions
Common Characters

a	The letter ‘a’ once	a?	The letter ‘a’ zero or once
a*	The letter ‘a’ zero, one or more times	a+	The letter ‘a’ one or more times
\t, \n	Tab, new line	x|y	Either x or y
.	Any single character	[^aeiou]	*Not* in set of characters
[xyz]	In set of characters	[a-z]	In range
\d \D	Digit / Non-digit	\w \W	Word character / non-word character
\s \S	White space / non-white space	\b \B	Word boundary / non-boundary
\040	ASCII as octal	\u0020	Unicode as hex
(*expr*)	Parentheses create an expression	(\d{2})?	Two digits or nothing

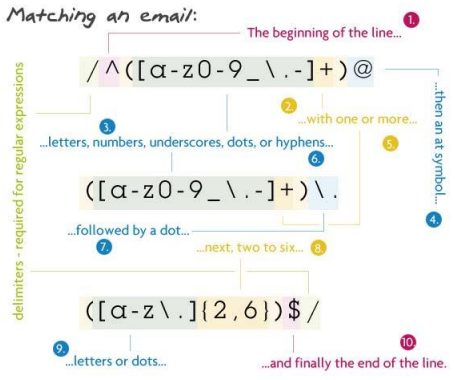
Regular Expressions Quick Reference
http://www.regular-expressions.info/refquick.html

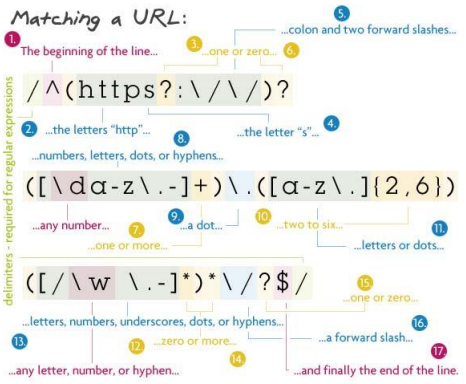
Regular Expressions 101
https://www.regex101.com/

Regular Expressions
Word Boundaries

- Input
  - This island is beautiful.
- Regular expression: **is**
- Matches
  - This island is beautiful.
- Regular expression: **\bis\b**
- Matches
  - This island is beautiful.

Regular Expressions
Common Examples






```
/^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{1,6})$/
```

α was the first single char top-level domain!

```
/^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{1,6})([\/\w \.-]*)*\/?$/
```

8 Regular Expressions You Should Know
http://net.tutsplus.com/tutorials/other/8-regular-expressions-you-should-know/

Regular Expressions
Email

- Email addresses are defined by RFC 5322 and could be represented by this regular expression

```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*
  |  "(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]
      |  \\[\x01-\x09\x0b\x0c\x0e-\x7f])*")
@ (?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?
  |  \[(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}
       (?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:
          (?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]
          |  \\[\x01-\x09\x0b\x0c\x0e-\x7f])+)
     \])
```

  - The reason you shouldn't use this is that it only checks the basic syntax of email addresses, for example, john@aol.com.nospam would be considered a valid email address according to RFC 5322

How to Find or Validate an Email Address
http://www.regular-expressions.info/email.html

Regular Expressions
UK Bank Sort Code

- It should allow either hyphens or not (12-34-56 or 123456)

Attempt 1

\d{2} = two digits

-? = optional hyphen

```
var sortcode1 = /^\d{2}-?\d{2}-?\d{2}$/;
console.log(sortcode1.test('12-34-56')); // => true
console.log(sortcode1.test('123456'));   // => true
console.log(sortcode1.test('000000'));   // => true
console.log(sortcode1.test('ab-cd-ef')); // => false
console.log(sortcode1.test('1234-56'));  // => true
```

- It should not allow all zeros or mixed hyphens or not

Attempt 2

Not allow zeros

Allow six digits only

Allow digits with hyphens

```
var sortcode = /^(?!(?:0{6}|00-00-00))(?:\d{6}|\d\d-\d\d-\d\d)$/;
console.log(sortcode.test('12-34-56')); // => true
console.log(sortcode.test('123456'));   // => true
console.log(sortcode.test('000000'));   // => false
console.log(sortcode.test('ab-cd-ef')); // => false
console.log(sortcode.test('1234-56'));  // => false
```

Lab Alternative

Form input

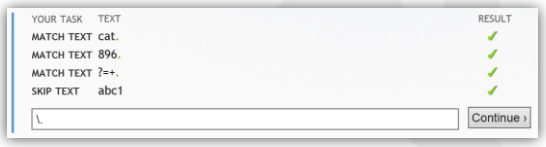
- Create a page that uses some of the new input types (url, email, date, number, range, color, and so on) and compare the experience in different browsers
- Create a page that uses some of the new input attributes (pattern, placeholder, required, autofocus, and so on) and compare the experience in different browsers

Regular expressions

- Create a page with text boxes into which a user may enter a regular expression and a value to test
- Provide a list box of common examples which when clicked will populate the text box

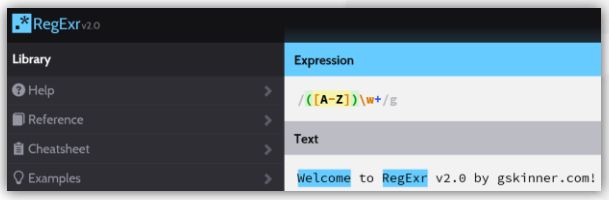
Further Study

- Learn regular expressions with interactive game



Learn regular expressions with simple, interactive examples
http://regexone.com/

- Learn regular expressions with interactive tool



RegExr is an online tool to learn, build, & test Regular Expressions
http://regexr.com/

Module 5
Communicating with a Remote Server

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Communicating with a Remote Server

Contents

Exam Topic: Consume data
- ❑ Consume JSON and XML data
- ❑ Retrieve data by using web services
- ❑ Load data or get data from other sources by using XMLHTTPRequest

Exam Topic: Serialize, deserialize, and transmit data
- ❑ Binary data
- ❑ Text data (JSON, XML)
- ❑ Implement the jQuery serialize method
- ❑ Form.Submit
- ❑ Parse data
- ❑ Send data by using XMLHTTPRequest
- ❑ Sanitize input by using URI/form encoding

Exam Topic: Implement a callback
- ❑ Use jQuery to make an AJAX call
- ❑ Wire up an event
- ❑ Implement a callback by using anonymous functions
- ❑ Handle the “this” pointer

XMLHttpRequest object
http://msdn.microsoft.com/library/ie/ms535874(v=vs.85).aspx

MOC Errata

5.3

- Page 5-6
  - The MOC says `var type = request.getResponseHeader();`
  - It should have said

```
var type = request.getResponseHeader("Content-Type");
```

- Page 5-10
  - In last code block, the MOC says

```
data: {
  ('#myForm').serializeArray();
}
```

  - It should have said

```
data: $('#myForm').serializeArray()
```

- Also, in the slide, it missed the close brace } for data:

Sanitizing
Encoding and Decoding

5.4

- For concatenating together and splitting apart text strings in URI parts
  - encodeURI takes something that's nearly a URI, but has invalid characters such as spaces in it, and turns it into a real URI
  - encodeURI and decodeURI are intended for use on the full URI
  - encodeURIComponent and decodeURIComponent are intended to be used on URI *components* i.e. any part that lies between separators (; / ? : @ & = + $, #)

```
js> s = "http://www.example.com/string with + and ? and & and spaces";
js> encodeURI(s)
http://www.example.com/string%20with%20+%20and%20?%20and%20&%20and%20spaces
js> encodeURIComponent(s)
http%3A%2F%2Fwww.example.com%2Fstring%20with%20%2B%20and%20%3F%20and%20%26%20and%20spaces
```

Spaces are encoded/decoded with both functions
+ ? & are encoded/decoded with URIComponent but NOT with URI

& becomes %26

What is the difference between decodeURIComponent and decodeURI?
http://stackoverflow.com/questions/747641/what-is-the-difference-between-decodeuricomponent-and-decodeuri

Sanitizing
Encoding and Decoding

5.5

Characters never encoded	( ) ! ~ _ - *
encodeURI and encodeURIComponent	[spaces] % £ others
encodeURIComponent	$ % @ = + &

	encodeURI	encodeURIComponent
$	$	%24
=	=	%3D
%	%25	%25
+	+	%2B
	%20	%20
@	@	%40
&	&	%26

```
var s = " %$=";
console.log(encodeURI(s));          // %20%25$=
console.log(encodeURIComponent(s)); // %20%25%24%3D
```

```
var s2 = "%20%25%24%3D";
console.log(decodeURI(s2));          //  %%24%3D
console.log(decodeURIComponent(s2)); //  %$=
```

XMLHttpRequest
Requests with Credentials

5.6

- Some calls may require user's credentials to be passed
  - To explicitly specify the username and password

```
var xhr = new XMLHttpRequest();
var url = 'http://www.firebrand.co.uk/protectedContent/';
xhr.open('GET', url, true, 'username', 'password'); // configuration
xhr.onreadystatechange = xhr_readystatechange; // set up handler
xhr.send(); // make the asynchronous request
```

  - To pass the currently logged on user credentials

```
xhr.withCredentials = true;
```

- Some calls may require credentials to be passed in the header before calling send()

```
xhr.setRequestHeader("secretKey", "Pa$$w0rd");
```

open()
https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest#open()

XMLHttpRequest
Cookies

5.7

- An AJAX request is an HTTP request so cookies get sent back and forth in the AJAX request-response headers
- To send a cookie, set it on the document before making an AJAX request

```
document.cookie =
    "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";
```

- To protect your data, AJAX calls only send cookies if the URL you're calling is on the same domain as your calling script

JavaScript Cookies
http://www.w3schools.com/js/js_cookies.asp

XMLHttpRequest
Events

5.8

- In ancient times the event to wait for the response is readystatechange when the readyState === 4
  - Warning! readystatechange with readyState === 4 also gets triggered for errors so load is better to avoid multiple events being triggered
- You can now also use simpler events
  - loadstart, progress, abort, error, load, timeout, loadend
  - Warning! Some mobile browsers don't support these yet

XMLHttpRequest - Living Standard
https://xhr.spec.whatwg.org/#event-handlers

Is onload equal to readyState==4 in XMLHttpRequest?
http://stackoverflow.com/questions/9181090/is-onload-equal-to-readystate-4-in-xmlhttprequest

jQuery AJAX Features
Overview

5.9

Method	Description
load	Loads data (typically partial HTML) from a server and puts it into the selected element
serialize	Encodes a set of form elements as an x-www-form-urlencoded string
serializeArray	Encodes a set of form elements as an array of names and values (which can then be JSON.stringify-ed)
param	Creates a serialized representation of an array or object (can be used as URL query string for AJAX requests)
get	Loads data from a server using an AJAX HTTP GET request
getJSON	Loads JSON-encoded data from a server using a HTTP GET request
post	Loads data from a server using an AJAX HTTP POST request
ajax	Performs an async AJAX request using a complex custom settings object
ajaxSetup	Sets the default values for future AJAX requests

jQuery AJAX Methods
http://www.w3schools.com/jquery/jquery_ref_ajax.asp

jQuery AJAX Features
$.get function

5.10

- jQuery.get(url [, data] [, success(data, textStatus, jqXHR)] [, dataType])
  - The success callback function is passed the returned data, which will be an XML root element, text string, JavaScript file, or JSON object, depending on the MIME type of the response
  - It is also passed the text status of the response

```
$.get("api/values", { 'choices': ["Jon", "Susan"] },
  function (data, textStatus, xhr) {
    $('#result').html(data);
    alert('Status: ' + textStatus);
});
```

jQuery.get()
http://api.jquery.com/jQuery.get/

$.ajax function caching

- If cache is set to false, it will force requested pages not to be cached by the browser
  - Note: Setting cache to false will only work correctly with HEAD and GET requests
  - It works by appending "_={timestamp}" to the GET parameters
  - The parameter is not needed for other types of requests, except in IE8 when a POST is made to a URL that has already been requested by a GET
- To force to retrieve the latest version of an HTML page

```
$.ajax({
  url: "test.html",
  cache: false
}).done(function (html) {
  $("#results").append(html);
});
```

jQuery.ajax()
http://api.jquery.com/jQuery.ajax/

$.ajax settings object properties[1] (1/3)

Name	Description	Default
accepts	What kind of response it will accept	Depends on dataType
async[2]	If you need synchronous requests, set this option to false (cross-domain requests and dataType: "jsonp" requests do not support synchronous operation)	true
cache	If set to false, it will force requested pages not to be cached by the browser. Setting cache to false will only work correctly with HEAD and GET requests	true, except for datatype 'script' and 'jsonp'
contentType	When sending data to the server, use this content type	'application/x-www-form-urlencoded; charset=UTF-8'
context	This object will be made the context of all Ajax-related callbacks	$.ajaxSettings merged with settings passed to $.ajax
data	Data to be sent to the server, converted to a query string, if not already a string (appended to the url for GET-requests)	

[1]Not all of them! [2]Deprecated

jQuery AJAX Features

$.ajax settings object properties[1] (2/3)

Name	Description	Default
crossDomain	If you wish to force a crossDomain request (such as JSONP) on the same domain, set the value to true	false for same-domain requests, true otherwise
dataFilter	A function to be used to sanitize the raw response	None
dataType*	The type of data that you're expecting back from the server	Intelligent guess
headers	An object of additional header key/value pairs to send	{}
jsonp	Override the callback function name in a jsonp request	
username, password	A username and password to be used with XMLHttpRequest in response to an HTTP access authentication request	""

[1]Not all of them! [2]Deprecated *If dataType is jsonp, type must be GET

How to use type: "POST" in jsonp ajax call
http://stackoverflow.com/questions/4508198/how-to-use-type-post-in-jsonp-ajax-call

jQuery AJAX Features

$.ajax settings object properties[1] (3/3)

Name	Description	Default
statusCode	An object of numeric HTTP codes and functions to be called when the response has the corresponding code e.g. statusCode: { 404: function() { ... } }	{}
complete[2]	Function called if request finishes (after error and success)	None
error[2]	Function called if request fails	None
success[2]	Function called if request succeeds	None
timeout	Set a timeout (in milliseconds)	
type	The type of request to make ("POST")	'GET'
url	The URL to which the request is sent	The current page

[1]Not all of them! [2]Deprecated: use done, fail, always instead

Should I Use success or Done?

- There are two ways to continue after an AJAX call
  - **success** has been the traditional name of the success callback in jQuery, defined as an option in the ajax call

```
$.ajax({ //...
  success: function(data) { /* ... */ }
});
```

  - Since the implementation of $.Deferreds, **done** is the preferred way to implement success callbacks

```
$.ajax({ /* ... */ })
  .done(function (data) {
    /* ... */
  });
```

Old	New
success	done
error	fail
complete	always

jQuery.ajax handling continue responses: “success:” vs “.done”?
http://stackoverflow.com/questions/8840257/jquery-ajax-handling-continue-responses-success-vs-done

What is *this* Inside jQuery Ajax Calls?

- If you use the (old) callback functions, what is the invocation context (this)? The combined settings object

```
$.ajax({
  // ...
  success: function (data) {
    // what does “this” refer to in here?
```

- For control, either create a temporary variable, or set the context property explicitly

```
var tempThis = this;
$.ajax({
  // ...
  success: function (data) {
    $(tempThis).addClass("cool");
  }
});
```

```
$.ajax({
  // ...
  context: anObject,
  success: function (data) {
    $(this).addClass("cool");
  }
});
```

$(this) inside of AJAX success not working
http://stackoverflow.com/questions/6394812/this-inside-of-ajax-success-not-working

ajax method returns jqXHR

- The jQuery XMLHttpRequest (jqXHR) object returned by $.ajax() as of jQuery 1.5 is a superset of the browser's native XMLHttpRequest object
  - So it has useful functions like getResponseHeader(*string*) as well as all the standard jQuery methods

```
var jqxhr = $.ajax( /* ... */);
var contType = jqxhr.getResponseHeader("Content-Type");
var contLength = jqxhr.getResponseHeader("Content-Length");
var lastMod = jqxhr.getResponseHeader("Last-Modified");
```

The jqXHR Object
http://api.jquery.com/jQuery.ajax/#jqXHR

getResponseHeader method (XMLHttpRequest)
http://help.dottoro.com/ljxsrgqe.php

Global Ajax Event Handlers

Method	Description
ajaxStart	When an Ajax operation begins and none are already active
ajaxSend	When an Ajax operation begins
ajaxComplete	When an Ajax operation finishes
ajaxSuccess	When an Ajax operation finishes successfully
ajaxError	When an Ajax operation has an error
ajaxStop	When all Ajax operations have finished

```
$(document).ajaxStart(function () {
    $(".log").text("Triggered ajaxStart handler.");
});
```

```
$(document).ajaxStop(function () {
    $("#loading").hide();
});
```

Ajax
http://api.jquery.com/category/ajax/

Serializing Forms

serialize() method

- Encode a set of form elements as a x-www-form-urlencoded string suitable for submission to a server (using name NOT id)

```
<form id="personForm">
    First Name: <input name="firstName" id="fnameID" /><br/>
    Age: <input name="age" id="ageID" /><br/>
    <input type="button" id="serializeButton" value="Serialize" /><br/>
    <textarea id="output"></textarea>
</form>
```

```
personForm.serializeButton.onclick = function () {
    personForm.output.value = $('#personForm').serialize();
};
```

serializeArray() method

```
personForm.output.value = JSON.stringify(
    $('#personForm').serializeArray());
```

```
[{"name":"firstName","value":"Bob"},
{"name":"age","value":"40"}]
```

- Returns a JavaScript array of objects that can be encoded with JSON.stringify()

.serialize()
http://api.jquery.com/serialize/

.serializeArray()
http://api.jquery.com/serializeArray/

Parsing XML

Sometimes a service will return XML

Browsers have a built-in XML parser

- An XML parser converts an XML document into an XML DOM object - which can then be manipulated with JavaScript

```
var x = "<!-- lots of XML to process -->";
if (window.DOMParser) {
  parser = new DOMParser();
  xmlDoc = parser.parseFromString(x, "text/xml");
} else { // Internet Explorer
  xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
  xmlDoc.async = false;
  xmlDoc.loadXML(x);
}
```

```
// or using jQuery
xmlDoc = $.parseXML("<xml ...");
```

XML Parser
http://www.w3schools.com/xml/xml_parser.asp

Starting with IE11 the window.ActiveXObject property is hidden from the DOM
http://msdn.microsoft.com/en-us/library/ie/dn423948%28v=vs.85%29.aspx

XML DOM

- How to retrieve data from an XML document
  - Retrieve the text value of the first <title> element:

```
txt = xmlDoc.getElementsByTagName("title")[0]
  .childNodes[0].nodeValue;
```

  - Retrieve the text value of the "lang" attribute of the first <title> element:

```
txt = xmlDoc.getElementsByTagName("title")[0]
  .getAttribute("lang");
```

```
<?xml version="1.0" encoding="utf-8" ?>
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <price>30.00</price>
  </book>
```

XML DOM Advanced
http://www.w3schools.com/xml/xml_dom_advanced.asp

XPath Basics

- Use / to define a path to elements and attributes
  - Name of all author elements

```
bookstore/author/name
```

  - All period attributes

```
bookstore/author/@period
```

- Use * as a wildcard
  - Name of all authors and books

```
bookstore/*/name
```

  - Name and nationality of all authors

```
bookstore/author/*
```

```
<bookstore>
  <author>
    <name>Victor Hugo</name>
    <nationality>French</nationality>
  </author>
  <author period="classical">
    <name>Sophocles</name>
    <nationality>Greek</nationality>
  </author>
  <author>
    <name>Leo Tolstoy</name>
    <nationality>Russian</nationality>
  </author>
  <author>
    <name>Alexander Pushkin</name>
    <nationality>Russian</nationality>
  </author>
  <book period="classical">
    <name>Plato's Works</name>
    <price>24.95</price>
  </book>
</bookstore>
```

XML
XPath Context and Filtering

- Context is very important for XPath statements
  - Author at root of document

```
/author
```

  - Authors anywhere in document

```
//author
```

- Use [ ] to define criteria to select nodes
  - Names of authors with Russian nationality

```
//author[nationality='Russian']/name
```

  - Nationalities of authors not from the classical period

```
//author[@period!='Classical']/nationality
```

Summary
Serialization Technologies

To do this…	Use this…
Serialize *from* a JavaScript object *to* a JSON string	`var string = JSON.stringify(object)`
Deserialize *from* a JSON string *to* a JavaScript object	`var object = JSON.parse(string);`
Serialize *from* a form *to* a x-www-formurlencoded string	`// requires jQuery` `var string = $(form).serialize();`
Serialize *from* a form *to* a JavaScript array	`// requires jQuery` `var array = $(form).serializeArray();`
Deserialize *from* an XML string *to* an XML DOM	`var xmldom = (new DOMParser())` `  .parseFromString(string, "text/xml");`

Cross Domain Requests

Understanding the Same-Origin Policy Problem

- Two pages have the same origin if the protocol, port (if one is specified), and host are the same for both pages
  - If the origin is: `http://store.company.com/dir/page.html`
  - Succeeds: `http://store.company.com/dir2/other.html`
  - Fails: `https://store.company.com/secure.html`
  - Fails: `http://news.company.com/dir/other.html`
- The same-origin policy controls interactions between two different origins, such as when you use XMLHttpRequest
- Use JSONP or CORS to allow cross-origin access

Cross Domain Requests

JSONP is "JSON with Padding"

- Requests for JSONP retrieve JavaScript code which is not blocked by same origin policy as JSON would be
  - Evaluated by the JavaScript interpreter, not by a JSON parser
- JSON payload (data) would be blocked

```
{ "Name": "Foo", "Id": 1234, "Rank": 7 }
```

- Equivalent JSONP payload (JavaScript) is let through

```
parseResponse({ "Name": "Foo", "Id": 1234, "Rank": 7 });
```

Cross Domain Requests

How JSONP Works Under the Covers

- The browser and server have to work together
  - By convention, the browser provides the name of the callback function as a named query parameter value, typically using the name **jsonp** or **callback** as the query parameter name

```
<script src="http://server2.example.com/Users/1234?jsonp=parseResponse">
</script>
```

  - The server responds with JSONP instead of JSON or XML

```
parseResponse({ "Name": "Foo", "Id": 1234, "Rank": 7 });
```

- For each new JSONP request, the browser must add a new <script> element, or reuse an existing one
  - JSONP can be said to allow browser pages to work around the same origin policy via “script element injection” and function name negotiation with the server

Cross Domain Requests

Using JSONP with jQuery

- With **ajax** function, specify **jsonp** as the dataType

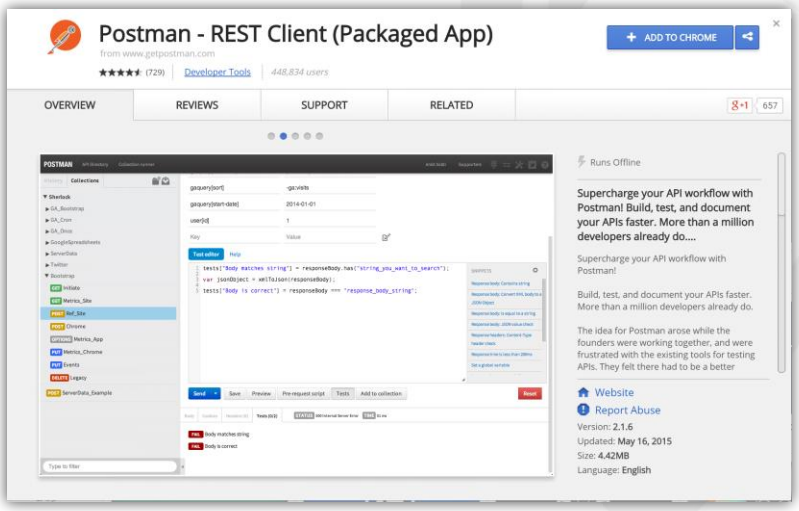
```
$.ajax({
  url: "person/update",
  dataType: 'jsonp'
})
.done(function (person) { alert("Got: " + person.name); })
.fail(function () { alert("Error"); });
```

- With **getJSON**, specify **callback=?** in the query string

```
$.getJSON("person/update?callback=?",
  function (person) { alert("Got: " + person.name); });
```

  - Note: if the URL includes the string "callback=?" (or similar, as defined by the server-side API), the request is treated as JSONP instead of JSON automatically if the server is configured to expect it

Further Study
Postman REST Client



Postman - REST Client
https://chrome.google.com/webstore/detail/postman-rest-client-packa/fhbjgbiflinjbdggehcddcbncdddomop/related?hl=en

Lab Alternative

- Create a ASP.NET MVC Web API project
  - Add an ADO.NET Entity Data Model (.edmx) for Northwind and then build the project
  - Add a Web API controller for the Supplier class
  - Add some form elements and script to the Home controller's Index view and use jQuery to call each of the action methods on the Supplier Web API controller to:
    - Get a list of all suppliers
    - Get a specific supplier based on ID
    - Insert a new supplier
    - Update an existing supplier
    - Delete a supplier

Module 6
Styling HTML5 by Using CSS3

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Styling HTML5 by Using CSS3

Contents

Exam Topic: Style HTML box properties
- ❑ Apply styles to alter appearance attributes (size, border and rounding border corners, outline, padding, margin)
- ❑ Apply styles to alter graphic effects (transparency, opacity, background image, gradients, shadow, clipping)
- ❑ Apply styles to establish and change an element's position (static, relative, absolute, fixed)

Exam Topic: Create a flexible content layout
- ❑ Implement a layout using:
  - ❑ a flexible box model
  - ❑ multi-column
  - ❑ position floating and exclusions
  - ❑ grid alignment
  - ❑ regions, grouping, and nesting

Exam Topic: Style HTML text properties
- ❑ Apply styles for:
  - ❑ text appearance (color, bold, italics)
  - ❑ text font (WOFF and @font-face, size)
  - ❑ text alignment, spacing, and indentation
  - ❑ text hyphenation & text drop shadow

Exam Topic: Create the document structure
- ❑ Create a layout container in HTML

Exam Topic: Structure a CSS file by using CSS selectors
- ❑ Style an element based on pseudo-elements and pseudo-classes (for example, :before, :first-line, :first-letter, :target, :lang, :checked, :first-child)

Exam Topic: Find elements by using CSS selectors and jQuery
- ❑ Find elements by using pseudo-elements and pseudo-classes

Can I use...

= Supported = Not supported = Partially supported = Support unknown

CSS Grid Layout - **Working Draft**

*Usage stats: Global
Support: 9.65%

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for lay out into columns and rows using a set of predictable sizing behaviors

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
Current	10.0 -ms-	23.0	29.0	6.0	16.0	6.0-6.1	5.0-7.0	4.2	10.0	10.0 -ms-
Near future	11.0 -ms-	24.0	30.0	7.0	17.0	7.0				
Farther future		25.0	31.0							

Flexible Box Layout Module - **Candidate Recommendation**

*Usage stats: Global
Support: 35.03%
Partial support: 44.29%
Total: 79.32%

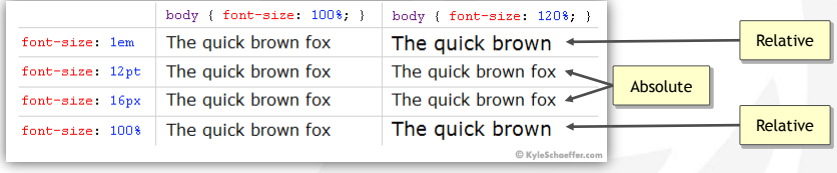
Method of positioning elements in horizontal or vertical stacks.

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1 -webkit-		
								2.2 -webkit-		
						3.2 -webkit-		2.3 -webkit-		
						4.0-4.1 -webkit-		3.0 -webkit-		
	8.0					4.2-4.3 -webkit-		4.0 -webkit-		
	9.0	22.0	28.0 -webkit-	5.1 -webkit-		5.0-5.1 -webkit-		4.1 -webkit-	7.0 -webkit-	
Current	10.0 -ms-	23.0	29.0	6.0 -webkit-	16.0 -webkit-	6.0-6.1 -webkit-	5.0-7.0	4.2 -webkit-	10.0 -webkit-	10.0 -ms-
Near future	11.0	24.0	30.0	7.0 -webkit-	17.0 -webkit-	7.0 -webkit-				
Farther future		25.0	31.0							

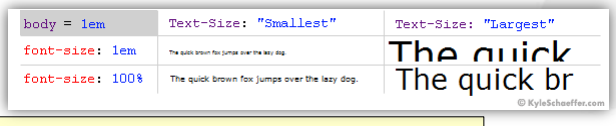
Text

Font-Size is Complicated

- Generally, 1em = 12pt = 16px = 100%
  - When using these font-sizes, when you increase the base font size (using the body CSS selector) from 100% to 120%



  - When 1em is our body font-size and the client alters the Text Size setting of their browser; on “Smallest”, ems are much smaller than percent, but on “Largest” setting, it’s opposite



CSS Font-Size: em vs. px vs. pt vs. percent
http://kyleschaeffer.com/development/css-font-size-em-vs-px-vs-pt-vs/

Text
Font-Size with rem

6.5

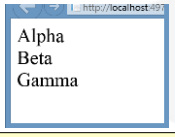
- Sizing with px
  - Pixels provide reliability and consistency, but IE will not allow adjustments to font size (although “zoom” gets round this)
- Sizing with em
  - Measurements are relative to parent element, so they compound with nested elements

```
body { font-size: 10pt; }
div  { font-size: 1.5em; }
```


```
<body>
  <div>
    Alpha
    <div>
      Beta
      <div>
        Gamma
```

- Sizing with rem
  - “Root em”: measurements are relative to the root element

```
div  { font-size: 1.5rem; }
```



Font Sizing With rem
http://snook.ca/archives/html_and_css/font-size-with-rem

CSS Fonts Module Level 3
http://www.w3.org/TR/css3-fonts/

Text
@font-face

6.6

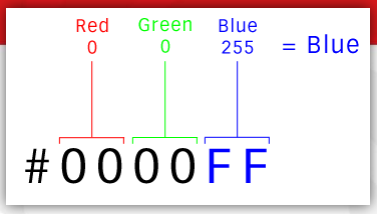
- @font-face is a css rule which allows you to download a particular font from your server to render a webpage if the user hasn’t got that font installed
  - This means that web designers will no longer have to adhere to a particular set of “web safe” fonts that the user has pre-installed on their computer
  - “Free” font faces tend to require attribution on your site

@font-face
http://www.font-face.com/

35+ awesome and free @font-face kits
http://www.webdesignerdepot.com/2012/09/35-awesome-and-free-font-face-kits/

Colours
Specifying Colours

6.7



- In hex colour notation, two consecutive characters that are identical can be shortened to just one character
  - That's why CSS hex colour notation can also be written with just 3 characters so #DDCC11 is the same as #DC1
- currentColor
  - A convenience keyword that just means the color being declared is equal to the CSS color property value:

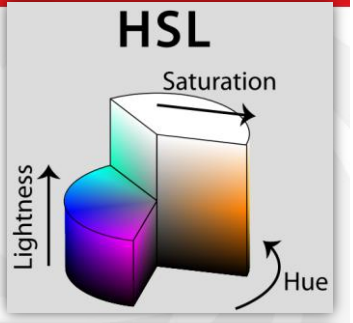
```
div {
    color: blue;
    background-color: currentColor; /* will also be blue */
}
```

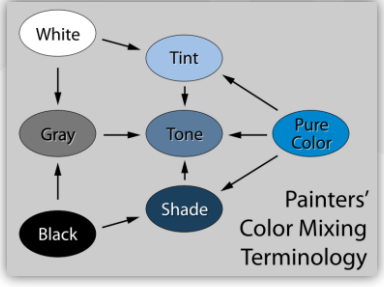
Introduction to CSS Colors
http://sixrevisions.com/css/css-colors/

Colours
Specifying Colours with HSL

6.8

- HSL allows cylindrical-coordinate colour representation
- Hue (0-360 degrees)
  - The *colour* e.g. red, blue, yellow
- Saturation (0-1)
  - Colourfulness e.g. "grey" to *colour*
- Lightness (0-1)
  - From black to *colour* to white
- Alpha (0-1): opacity





HSL Color Picker
http://hslpicker.com/

HSL and HSV
https://en.wikipedia.org/wiki/HSL_and_HSV

Graphical Effects
border-radius

6.9

- The border of elements can be given a radius to create rounded corners

```
#radiusDemo {
    height: 100px;
    width: 140px;
    background-color: pink;
    border-radius: 10px 20px 30px 40px;
}
```

```
<div id="radiusDemo"></div>
```


Layout
position

6.10

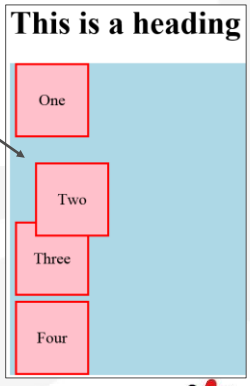
- position has a default value of static

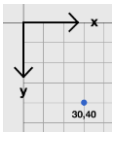
```
position: static;
```

  - Other choices: relative, absolute, fixed*, initial, inherit
  - top, right, bottom, and left can be set when not static

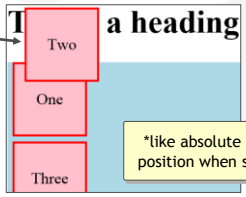
```
<div id="ancester">
  <div id="one">One</div>
  <div id="two">Two</div>
  <div id="three">Three</div>
  <div id="four">Four</div>
</div>
```

```
#two {
  position: relative;
  top: 20px;
  left: 20px;
}
```






```
#two {
  position: absolute;
  top: 20px;
  left: 20px;
}
```



CSS Positioning 101
http://alistapart.com/article/css-positioning-101

*like absolute but stays in its fixed position when scrolling down a page

Layout
position

6.11

position	Description
static	Default value. Elements render in order, as they appear in the document flow
relative	The element is positioned relative to its normal position, so "left:20" adds 20 pixels to the element's LEFT position
absolute	The element is positioned relative to its first positioned (not static) ancestor element
fixed	The element is positioned relative to the browser window
initial	Sets this property to its default value (not supported by IE)
inherit	Inherits this property from its parent element

```
<div>
    <h1>Will this be black or red?</h1>
</div>
```

```
div {
    color: red;
}
h1 {
    color: initial;
}
```

CSS initial Keyword
http://www.w3schools.com/cssref/css_initial.asp

Layout
Block Styles

6.12

```
/* CSS */
display: inline;
```

```
// JavaScript
element.style.display = "inline";
```

display

- **block**: formatted down the page one after another and respect padding, border, and margin values
- **inline**: formatted one after another based on the baseline of their text content until they break down onto another line, but they *ignore* height and width values
- **inline-block**: formatted one after another based on the baseline of their text content, but they *keep* height and width values
- **table**: enables you to identify blocks on the page as tables, rows, columns, and cells. Blocks are aligned by their edges rather than their content, and sized to fit the computed table
- **flex/-ms-flexbox**: choose in which direction boxes are laid out and how boxes are sized, depending on how any excess whitespace around blocks should be handled

CSS - The Box Model, Floats, and Positioning
http://cfg.good.is/lessons/css-the-box-model-floats-and-positioning

Layout
float

6.13

- Specifies whether or not an element should float
  - left, right, none, inherit
- Elements float horizontally, meaning that an element can only be floated left or right, not up or down
  - A floated element will move as far to the left or right as it can within its containing element
  - The elements after the floating element will flow around it
  - The elements before the floating element will not be affected
  - Note: you must specify a width for block elements otherwise they will continue to take up the full width of their parent

CSS Float
http://www.w3schools.com/css/css_float.asp

CSS Float – Try It
http://www.w3schools.com/css/tryit.asp?filename=trycss_float_elements

Layout
float Example

6.14

- Without float

```
<div>AAAAA Tincidunt integer eu augue ...</div>
<aside>BBBBB Placerat suscipit, orci nisl ...</aside>
<div>CCCCC Fusce non varius purus aenean nec ...</div>
<div id="D">DDDDD Sem nulla eu ultricies ...</div>
```

```
div, aside {
    margin: 0.5em;
    padding: 0.5em;
}
```

AAAAA Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque euismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac, aliquam sit amet justo nunc tempor, metus vel.

BBBBB Placerat suscipit, orci nisl iaculis eros, a tincidunt nisi odio eget lorem nulla condimentum tempor mattis ut vitae feugiat augue eras ut metus a risus iaculis scelerisque eu ac ante.

CCCCC Fusce non varius purus aenean nec magna felis fusce vestibulum velit mollis odio sollicitudin lacinia aliquam posuere, sapien elementum lobortis tincidunt, turpis dui ornare nisl, sollicitudin interdum turpis nunc eget.

- With float *and width* set on the aside

AAAAA Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque euismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac, aliquam sit amet justo nunc tempor, metus vel.

CCCCC Fusce non varius purus aenean nec magna felis fusce vestibulum velit mollis odio sollicitudin lacinia aliquam posuere, sapien elementum lobortis tincidunt, turpis dui ornare nisl, sollicitudin interdum turpis nunc eget.

BBBBB Placerat suscipit, orci nisl iaculis eros, a tincidunt nisi odio eget lorem nulla condimentum tempor mattis ut vitae feugiat augue eras ut metus a risus iaculis scelerisque eu ac ante.

DDDDD Sem nulla eu ultricies orci praesent id augue nec lorem pretium congue sit amet ac nunc fusce iaculis lorem eu diam hendrerit at mattis purus dignissim vivamus mauris tellus, fringilla.

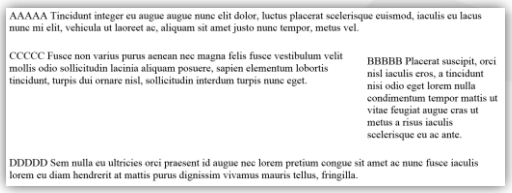
EEEEE Vel dapibus a, blandit quis erat vivamus elementum aliquam luctus etiam fringilla pretium sem vitae sodales mauris id nulla est praesent laoreet, metus vel auctor aliquam, eros purus vulputate leo.

```
aside {
    float: right;
    width: 200px;
}
```

Clearing Floats

- Elements after the floating element will flow around it unless you use the clear property
  - The clear property specifies which sides of an element other floating elements are not allowed
- Turning off float by using clear
  - left, right, both, none, inherit

```
div#D {
    clear: right;
}
```



CSS Float with Clear – Try It
http://www.w3schools.com/css/tryit.asp?filename=trycss_float_clear

Using Floats for Layouts (1 of 2)

- Two-column layout with header and footer, no CSS

```
<article>
    <header>
        <h1>Welcome</h1>
    </header>
    <section>
        <div>Tincidunt integer eu ...</div>
        <div>Placerat suscipit, ...</div>
        <div>Fusce non varius ...</div>
        <div>Sem nulla eu ...</div>
        <div>Vel dapibus a, ...</div>
    </section>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">Sales</a></li>
            <li><a href="#">Contact</a></li>
            <li><a href="#">About</a></li>
        </ul>
    </nav>
    <footer>
        Copyright &copy; 2015 Firebrand Training Ltd
    </footer>
</article>
```

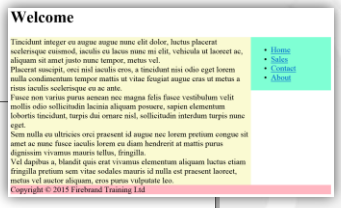
Layout

Using Floats for Layouts (2 of 2)

- Two-column layout with header and footer, with CSS

```
<article>
    <header>
        <h1>Welcome</h1>
    </header>
    <section>
        <div>Tincidunt integer eu ...</div>
        <div>Placerat suscipit, ...</div>
        <div>Fusce non varius ...</div>
        <div>Sem nulla eu ...</div>
        <div>Vel dapibus a, ...</div>
    </section>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">Sales</a></li>
            <li><a href="#">Contact</a></li>
            <li><a href="#">About</a></li>
        </ul>
    </nav>
    <footer>
        Copyright &copy; 2015 Firebrand Train
    </footer>
</article>
```

```
article {
    margin: 0 auto;
    padding: 10px;
    width: 640px;
}
section {
    width: 480px;
    background-color: lightgoldenrodyellow;
    float: left;
}
nav {
    background-color: aquamarine;
    width: 160px;
    float: right;
}
footer {
    background-color: lightpink;
    clear: both;
}
```



Without the **clear** the **footer** would appear directly under the **nav**

CSS Floats 101
http://alistapart.com/article/css-floats-101

Flex Layout

Enabling Flex Layout (apply to container)

- The main idea behind the flex layout
  - Give the container the ability to alter its items' width, height and order to best fill the available space (mostly to accommodate to all kind of display devices and screen sizes)
  - It expands items to fill available free space, or shrinks them to prevent overflow
- Flex layout involves setting properties on both the "flex container" and the "flex items"
- To enable flex layout on a container

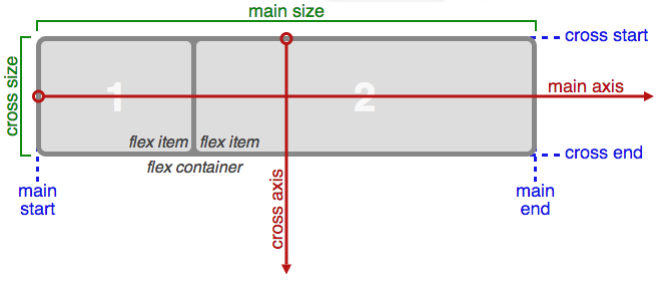
```
display:  flex | <others>;
/* vendor prefixes */
display: -webkit-box;
display: -moz-box;
display: -ms-flexbox;
display: -webkit-flex;
```

A Complete Guide to Flexbox
http://css-tricks.com/snippets/css/a-guide-to-flexbox/

Using CSS flexible boxes
https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Flexible_boxes

Flex Layout
flex-direction (apply to container)

- To establish the main-axis, thus defining the direction flex items are placed in the flex container
  - Items will be laid out following either the main axis (from main-start to main-end) or the cross axis (cross-start to cross-end)




```
flex-direction: row | row-reverse | column | column-reverse;
-ms-flex-direction: row | row-reverse | column | column-reverse;
```

  - "column" means top to bottom

Flex Layout
flex-wrap (apply to container)

- Defines whether the flex container is single-line or multi-line

```
flex-wrap: nowrap | wrap | wrap-reverse;
-ms-flex-wrap: nowrap | wrap | wrap-reverse;
```

- flex-flow is a shorthand that combines the settings for flex-direction and flex-wrap
  - Default is **row nowrap**

```
flex-flow: <flex-direction> || <flex-wrap>;
-ms-flex-flow: <flex-direction> || <flex-wrap>;
```

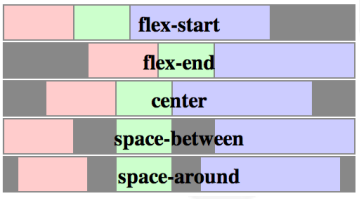
Flex Layout
6.21

justify-content (apply to container)

- Defines the alignment along the main axis

```
justify-content: flex-start | flex-end | center |
                 space-between | space-around;
-ms-flex-pack: start | end | center | justify | distribute;
```

- flex-start / start: items are packed toward the start line (default)
- flex-end / end: items are packed toward to end line
- center: items are centered along the line
- space-between / justify
- space-around / distribute



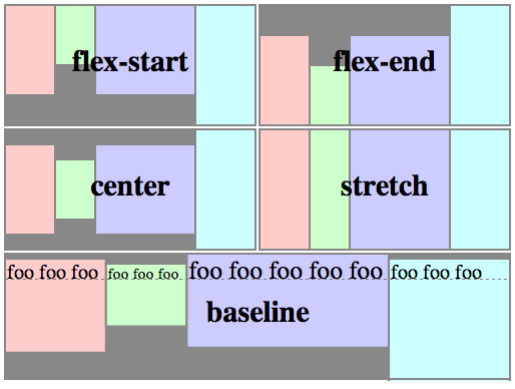
Flex Layout
6.22

align-items (apply to container)

- Sort of the justify-content version for the cross-axis

```
align-items: flex-start | flex-end | center | baseline | stretch;
-ms-flex-align: start | end | center | baseline | stretch;
```

- stretch (default): items stretch to fill the container

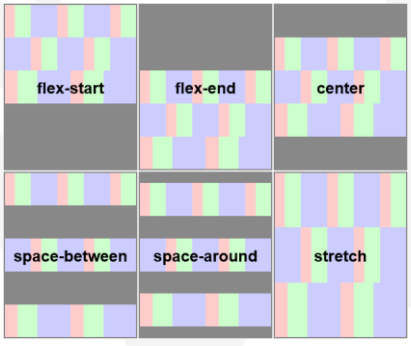


Flex Layout

align-content (apply to container)

- Aligns the flexible container's items when the items do not use all available space on the cross-axis (vertically)
  - There must be multiple lines of items for this property to have any effect!

```
align-content: stretch | center |
flex-start | flex-end | space-between
| space-around | initial | inherit;
```



align-content
http://css-tricks.com/almanac/properties/a/align-content/

Flex Layout

order and flex (apply to flex item)

- By default, flex items are laid out in the source order
  - Order controls the order in which they appear in their container

```
order: <integer>;
-ms-flex-order: <integer>;
```

- flex allows a flex item to grow if necessary

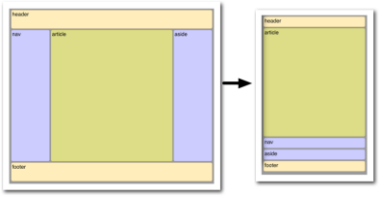
```
flex: none | auto | <integer>;
-ms-flex: none | auto | <integer>;
```

  - It accepts a unitless value that serves as a proportion and dictates what amount of the available space inside the flex container the item should take up
  - If all items have flex-grow set to 1, every item will use an equal size inside the container; if you were to give one of the children a value of 2, that child would take up twice as much space

Flex Layout

CSS3 Flexible Box Example for Internet Explorer 10

```
<header>...</header>
<div id='main'>
    <article>...</article>
    <nav>...</nav>
    <aside>...</aside>
</div>
<footer>...</footer>
```



```
#main { display: -ms-flexbox; }
#main > article { -ms-flex-order: 2; min-width: 12em; -ms-flex: 1; }
#main > nav { -ms-flex-order: 1; width: 200px; }
#main > aside { -ms-flex-order: 3; width: 200px; }
```

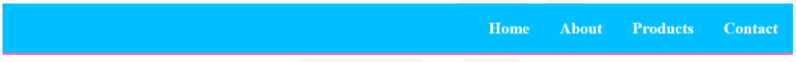
```
@media all and (max-width: 600px) {
  #main { -ms-flex-flow: column; }
  #main > article, #main > nav, #main > aside {
      -ms-flex-order: 0; width: auto; }
}
```

Use a media query to switch to an all-vertical layout for narrow screens

CSS Flexible Box Layout Module Level 1, Editor's Draft, 11 September 2013
http://dev.w3.org/csswg/css-flexbox/

Flex Layout

CSS3 Flexible Box Navigation Example (1 of 2)




```
.navigation {
  list-style: none;
  display: -ms-flexbox;
  background: deepskyblue;
  border-bottom: 2px solid hotpink;
  -ms-flex-pack: end;
}
.navigation a {
  text-decoration: none;
  display: block;
  padding: 15px;
  color: white;
  font-weight: bold;
}
.navigation a:hover {
  background: darken(deepskyblue, 2%);
}
```

```
<ul class="navigation">
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Products</a></li>
    <li><a href="#">Contact</a></li>
</ul>
```

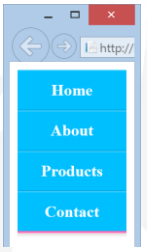
Flex Layout

CSS3 Flexible Box Navigation Example (2 of 2)

```
@media all and (max-width: 800px) {
    .navigation {
        -ms-flex-pack : distribute;
    }
}
```


```
@media all and (max-width: 500px) {
    .navigation {
        -ms-flex-flow: column wrap;
        padding: 0;
    }

        .navigation a {
            text-align: center;
            padding: 10px;
            border-top: 1px solid
rgba(255,255,255,0.3);
            border-bottom: 1px solid rgba(0,0,0,0.1);
        }
}
```



-ms-grid

Grid Block Style

- Grid enables you to align elements into columns and rows but has no content structure

```
<div id="grid">
    <div id="item1">abc</div>
    ...
</div>
```

```
#grid {
    display: -ms-grid;
    -ms-grid-columns: auto 100px 1fr 2fr;
    -ms-grid-rows: 50px 5em auto;
}
```

- -ms-grid-columns, -ms-grid-rows
  - auto: fitted to the content
  - fr: fractional unit of remaining space (like * in <table>)

Grid layout
http://msdn.microsoft.com/en-us/library/ie/hh673533(v=vs.85).aspx

CSS Grid Layout, W3C Working Draft, 2 April 2013
http://www.w3.org/TR/css3-grid-layout/

-ms-grid
Repeating Syntax

- If there are large number of columns or rows that are the same or exhibit a recurring pattern, a repeat syntax can be applied using brackets

```
#grid { /* the long way */
    display: -ms-grid;
    -ms-grid-columns: 10px 250px 10px 250px 10px 250px 10px;
```

```
#grid { /* the shorter way */
    display: -ms-grid;
    -ms-grid-columns: 10px (250px 10px)[3];
```

- Must use () even with only one value to repeat

```
#anotherGrid {
    display: -ms-grid;
    -ms-grid-columns: 10px (80px)[6] 10px;
```

-ms-grid
Grid Items

- Child elements of the Grid are called Grid items
  - Positioned using -ms-grid-row and -ms-grid-column (1-based)

```
#item1 {
    background: red;
    border: orange solid 1px;
    -ms-grid-row: 1; /* 1 means first row */
    -ms-grid-column: 1;
}

#item2 {
    background: green;
    border: red solid 1px;
    -ms-grid-row: 2;
    -ms-grid-column: 4;
}
```

```
<div id="grid">
    <div id="item1">abc</div>
    <div id="item2">def</div>
    ...
</div>
```

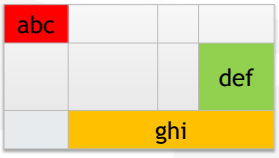
Grid layout reference
http://msdn.microsoft.com/en-us/library/ie/hh772052(v=vs.85).aspx

-ms-grid
Spanning

- You can make grid items span multiple columns or rows using –ms-grid-column-span or –ms-grid-row-span

```
#item3 {
    background: orange;
    border: maroon solid 1px;
    -ms-grid-row: 3;
    -ms-grid-column: 2;
    -ms-grid-column-span: 3;
    text-align: center;
}
```

```
<div id="grid">
    ...
    <div id="item3">ghi</div>
    ...
</div>
```



- Note: if a spanned item overlaps another item then the order of the divs determines which appears on top

How to create an adaptive layout with CSS Grid
http://msdn.microsoft.com/en-us/library/ie/jj553856(v=vs.85).aspx

Pseudo-Elements and Pseudo-Classes
What Are They?

- Pseudo-elements (part of an element) and pseudo-classes (an element in a particular state) create abstractions about the document tree

```
a:hover { color: orange; } /* pseudo-class for the link's hover state */
p::after { content: "*"; } /* pseudo-element for the space after a p */
```

- Pseudo-element :: notation was introduced in order to establish a discrimination from pseudo-classes :
- For compatibility, user agents must also accept the previous one-colon notation introduced in CSS levels 1 and 2 (and IE8 only understands a single colon!)
- Only one pseudo-*thingy* may appear per selector so you cannot combine :hover and ::after, for example

7. Pseudo-elements
http://www.w3.org/TR/css3-selectors/#pseudo-elements

Link Styles

- If you define CSS rules that match against more than one of these pseudo-classes, it is important that you specify these pseudo-classes in the following order:
  - a:link
  - a:visited
  - a:focus
  - a:hover
  - a:active
  - Louis
  - Vuitton
  - For
  - Hells
  - Angels
- a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!
- a:active MUST come after a:hover in the CSS definition in order to be effective!

CSS Pseudo-classes
http://www.w3schools.com/css/css_pseudo_classes.asp

nth-child vs nth-of-type

- nth-child is commonly used although nth-of-type is usually better

```
<div>
    <p>Apples</p>
    <p>Bananas</p>
</div>
<div>
    <h1>Heading</h1>
    <p>Apples</p>
    <p>Bananas</p>
</div>
<div>
    <h1>Heading</h1>
    <h2>Sub</h2>
    <p>Apples</p>
    <p>Bananas</p>
</div>
```

Apples
Bananas
Heading
Apples
Bananas
Heading
Sub
Apples
Bananas

```
p:nth-child(2) {
    color: red;
}
p:nth-of-type(2) {
    background-color: yellow;
}
```

The Difference Between :nth-child and :nth-of-type
http://css-tricks.com/the-difference-between-nth-child-and-nth-of-type/

Pseudo-Elements and Pseudo-Classes
::after and ::before with inputs

- Pseudo elements can only be defined (or better said are only supported) on *container* elements
  - Because the way they are rendered is within the container itself as a child element; input or img cannot contain other elements hence they're not supported (inputs are *replaced* elements)
  - Wrap the input in a container element such as a or

```
<div class="field_with_errors showstar">
    <input type="text" />
</div>
```

```
.field_with_errors {
    display: inline;
    color: red;
}
.showstar::after {
    content: "*";
}
```

  - If using jQuery you could use

```
$(".mystyle").after("add your smiley here");
```

.after()
http://api.jquery.com/after/

CSS :after pseudo element on INPUT field
http://stackoverflow.com/questions/2587669/css-after-pseudo-element-on-input-field

Pseudo-Elements and Pseudo-Classes
::first-letter

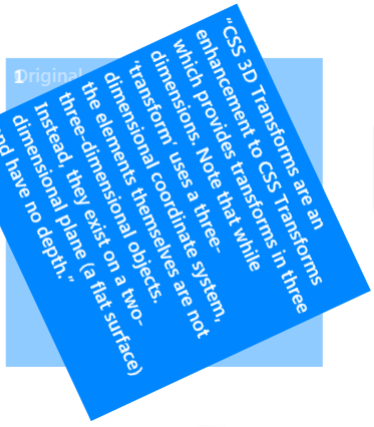
- The CSS ::first-letter selector only works with block-level elements like div, not inline elements like span

```
p::first-letter {
    font-size: 200%;
    color: #8A2BE2;
}
```

CSS ::first-letter Selector
http://www.w3schools.com/cssref/sel_firstletter.asp

CSS Transforms
Rotate

- rotateZ is equivalent to rotate because it specifies a clockwise rotation by the given angle about the z-axis




```
div {
    -moz-transform: rotateZ(45deg);
    -ms-transform: rotateZ(45deg);
    -o-transform: rotateZ(45deg);
    -webkit-transform: rotateZ(45deg);
    transform: rotateZ(45deg);
}
```

rotateZ() function
http://msdn.microsoft.com/en-us/library/ie/jj200275(v=vs.85).aspx

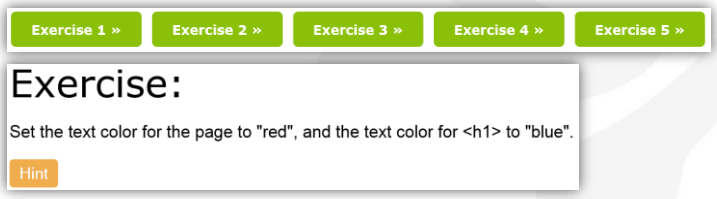
CSS Columns
Supporting Multiple Browsers

- To support columns in Chrome use vendor prefix

```
div.newspaper {
    -moz-column-count: 3;
    -webkit-column-count: 3; /* Chrome */
    column-count: 3;         /* IE 10 */
}
```

Lab Alternative

- Experiment with different layout models
- Review Bootstrap in Appendix and see how it works
- w3schools.com
  - Learn HTML and Learn CSS
  - Do exercises at bottom of each page



THE WORLD'S LARGEST WEB DEVELOPER SITE
http://www.w3schools.com/

CSS3 Please! The Cross-Browser CSS3 Rule Generator
http://css3please.com/

Module 7
Creating Objects and Methods by Using JavaScript

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Creating Objects and Methods by Using JavaScript
Contents

Exam Topic: Create and implement objects and methods
- Implement native objects
- Create custom objects and custom properties for native objects using prototypes and functions
- Inherit from an object
- Implement native methods and create custom methods

Exam Topic: Establish the scope of objects and variables
- Define the lifetime of variables
- Keep objects out of the global namespace
- Scope variables locally and globally

Variable Scope (JavaScript)
http://msdn.microsoft.com/library/bzt2dkta(v=vs.94).aspx

isPrototypeOf Method (Object) (JavaScript)
http://msdn.microsoft.com/library/bch72c9e(v=vs.94).aspx

ECMAScript Versions

ECMAScript 5 Compatibility Table

ECMAScript 5 compatibility table — Also see compatibility tables for ES6

Please note that *some of these tests* represent **existence**, not functionality or full conformance. I hope to test conformance some

	THIS BROWSER	IE 7	IE 8	IE 9	IE 10, 11	FF 3	FF 3.5, 3.6	FF 4+
Object.create	Yes	No	No	Yes	Yes	No	No	Yes
Object.defineProperty	Yes	No	Yes[1]	Yes	Yes	No	No	Yes
Object.defineProperties	Yes	No	No	Yes	Yes	No	No	Yes
Object.getPrototypeOf	Yes	No	No	Yes	Yes	No	Yes	Yes
Object.keys	Yes	No	No	Yes	Yes	No	No	Yes
Object.seal	Yes	No	No	Yes	Yes	No	No	Yes
Object.freeze	Yes	No	No	Yes	Yes	No	No	Yes
Object.preventExtensions	Yes	No	No	Yes	Yes	No	No	Yes
Object.isSealed	Yes	No	No	Yes	Yes	No	No	Yes
Object.isFrozen	Yes	No	No	Yes	Yes	No	No	Yes
Object.isExtensible	Yes	No	No	Yes	Yes	No	No	Yes
Object.getOwnPropertyDescriptor	Yes	No	Yes[1]	Yes	Yes	No	No	Yes
Object.getOwnPropertyNames	Yes	No	No	Yes	Yes	No	No	Yes

ECMAScript 5 compatibility table
http://kangax.github.io/es5-compat-table/

Scope

Block Scope using let

ECMAScript 6.0 only!

- let allows you to declare variables, limiting its scope to the block, statement, or expression on which it is used
- This is unlike var keyword, which defines a variable globally, or locally to an entire function regardless of block scope

```
function doStuff() {
  var a = 5;
  var b = 10;
  if (a === 5) {
    let a = 4; // a new variable scoped inside the if-block
    var b = 1; // replaces the existing 'b'
    console.log(a);  // 4
    console.log(b);  // 1
  }
  console.log(a); // 5
  console.log(b); // 1
}
```

let
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let

Scope
Hoisting Variables

7.5

- Function declarations and variable declarations are always moved ("hoisted") invisibly to the top of their containing scope by the JavaScript interpreter
  - So this…

```
function foo() {
    if (false) {
        var x = 1;
    }
    return;
    var y = 1;
}
```

…is actually interpreted as this

```
function foo() {
    var x, y;
    if (false) {
        x = 1;
    }
    return;
    y = 1;
}
```

- Best practice
  - Each function should have single var at the top for all variables

```
function foo(a, b, c) {
    var x = 1,
        bar,
        baz = "something";
```

Scope
Hoisting Functions

7.6

- Hoisting with functions differs depending on if you are using a named or anonymous function

```
function test() {
    foo(); // TypeError "foo is not a function"
    bar(); // "this will run!"
    var foo = function () {
        alert("this won't run!");
    }
    function bar() {
        alert("this will run!");
    }
}
```

```
test();
```

…is actually interpreted as this →

```
function test() {
    var foo;
    function bar() {
        alert("this will run!");
    }
    foo(); // TypeError "foo is not ...
    bar(); // "this will run!"
    foo = function () {
        alert("this won't run!");
    }
}
```

Scope
"Iffy" jQuery

7.7

- You sometimes see this:

```
(function ($) {
    // some JavaScript code here that uses jQuery
})(jQuery);
```

  - This is an example of the "JavaScript Module" pattern using immediately invoked function execution (IIFE)
  - The purpose is to provide modularity, privacy and encapsulation
  - Passing jQuery to the parenthesis gives local scoping to the global variable that helps reduce the amount of overhead of looking up the $ variable
  - It is very different from an event handler for DOMReady:

```
$(function () {
    // some JavaScript code that executes when the DOM is ready
});
```

Why define anonymous function and pass jQuery as the argument?
http://stackoverflow.com/questions/10371539/why-define-anonymous-function-and-pass-jquery-as-the-argument/10372429#10372429

Objects and Prototypes
Terms

7.8

- Native object
  - Defined by ECMAScript specification
  - e.g. arrays, functions, dates, and regular expressions
- Host object
  - Defined by host environment (typically the web browser)
  - e.g. DOM, window
- User-defined object
  - Any object defined by the execution of code
- "Own" and "inherited" properties
  - An own property is defined directly on an object
  - An inherited property is defined by an object's prototype chain

What Is An Object?

- It is a dynamic unordered dictionary of properties
  - Each property has a **name** and a **value**: the name can be any string but each must be unique, the value can be any type, including functions
- Each property has three other hidden attributes
  - **writable**, **enumerable** (included with a for-in loop), **configurable** (can it be deleted or modified)
- Each object has five hidden attributes
  - **prototype**: an object from which more properties are inherited
  - **subtype**: a string that categorizes the object
  - **extensible**, **sealed**, and **frozen**: Can properties be added? Are they configurable? Are they read-only?

Example of an Object

- **bob** has its "own" properties
- **bob** has "inherited" properties from its prototype

```
var bob = {
  firstName: "Bob",
  age: 42
};
console.log(bob.age);
var s = bob.toString();
```

bob

prototype	Object.prototype
subtype	"Object"
extensible	true
sealed	false
frozen	false

own properties

Property	Value	Writable	Enumerable	Configurable
"firstName"	"Bob"	true	true	true
"age"	42	true	true	true

Object.prototype

prototype	null
subtype	"Object"
extensible	true
sealed	false
frozen	false

Inherited properties

Property	Value	Writable	Enumerable	Configurable
"toString"	function	true	false	true
...	...	...	...	...

Comparing Object and Object.prototype

- **Object** is a `function` that can be called with `new` to construct an empty object whose prototype is **Object.prototype** meaning **Object.prototypes'** properties will be inherited by new object
- **Object** has properties to manage objects

```
var bob = new Object();
var s = bob.toString();
Object.defineProperty(...
```

Object

prototype	Object.prototype
subtype	"Function"
extensible	true
sealed	false
frozen	false

own properties

Property	Value	Writable	Enumerable	Configurable
"create"	function	true	true	true
"defineProperty"	function	true	true	true
...	...	...	...	...

Object.prototype

prototype	null
subtype	"Object"
extensible	true
sealed	false
frozen	false

Property	Value	Writable	Enumerable	Configurable
"toString"	function	true	false	true
...	...	...	...	...

Object

Object has "methods" that are *not* inherited

Member	Description
create	Creates a new object with the specified prototype object and properties
defineProperty, defineProperties	Adds the named property(ies) described by given descriptor(s) to an object
getOwnPropertyDescriptor	Returns a property descriptor for a named property on an object
getOwnPropertyNames	Returns an array containing the names of all of the given object's own enumerable and non-enumerable properties
freeze, seal, preventExtensions	"Lock" an object (see slide 7.6)
isFrozen, isSealed, isExtensible	Check if an object is "locked"
getPrototypeOf, setPrototypeOf*	Returns the prototype of the specified object

*Experimental and very slow!

Inherited from Function: apply, call, toSource, toString

Object.prototype

- (Almost) all objects have a prototype chain back to **Object.prototype** and so "inherit" these "methods"

Member	Description
hasOwnProperty	Returns a boolean indicating whether an object contains the specified property as a direct property of that object and not inherited through the prototype chain
isPrototypeOf	Returns a boolean indication whether the specified object is in the prototype chain of the object this method is called upon
propertyIsEnumerable	Returns a boolean indicating if the internal ECMAScript DontEnum attribute is set
toLocaleString	Returns an object converted to a string based on the current locale
toString	Returns a string representation of the object
valueOf	Returns the primitive value of the specified object

Getting the subtype

- You need to write your own function to read a subtype

```
function getSubtype(o) {
  if (o === null) return "Null";
  if (o === undefined) return "Undefined";
  var subtype = Object.prototype.toString.call(o).slice(8, -1);
  // for your custom objects, give them a property named subtype
  return ((subtype === "Object") && (o.subtype)) ? o.subtype : subtype;
}
```

`[object subtype]`

Note: Custom objects, even created with a constructor, will return "Object", so add a property called subtype (or whatever) to the prototype of your constructors

```
console.log(getSubtype(null));       // => Null
console.log(getSubtype(1));          // => Number
console.log(getSubtype(""));         // => String
console.log(getSubtype(false));      // => Boolean
console.log(getSubtype({}));         // => Object
console.log(getSubtype([]));         // => Array
console.log(getSubtype(/./));        // => Regexp
console.log(getSubtype(new Date())); // => Date
console.log(getSubtype(window));     // => Window
```

```
var bob = new Firebrand.Person('Bob', 'Smith', 45);
console.log(getSubtype(bob)); // => Firebrand.Person
```

Extensibility

- Specifies whether new properties can be added to an object (or not)

```
var o = { x: 4 };
console.log(Object.isExtensible(o)); // => true
Object.preventExtensions(o);
console.log(Object.isExtensible(o)); // => false
```

  - Object.isSealed(object)
  - Object.seal(object): makes existing properties nonconfigurable
  - Object.isFrozen(object)
  - Object.freeze(object): makes existing properties read-only
- These only affect the object you apply method to
  - To completely lock an object you would need to call method on all prototypes in the chain

Warning! preventExtensions(), seal(), and freeze() cannot be undone

Helper Method to Read Special Values

```
function showObjectDetails(obj, name) {
    console.log("*** " + name);
    console.log("Prototype: " + Object.getPrototypeOf(obj));
    console.log("Subtype: " + getSubtype(obj));
    console.log("Extensible: " + Object.isExtensible(obj));
    console.log("Sealed: " + Object.isSealed(obj));
    console.log("Frozen: " + Object.isFrozen(obj));
    console.log("Own and inherited enumerable properties:")
    for (var p in obj) {
        console.log("  " + p + ": " + obj[p]);
    }
    var ownPropertyNames = Object.getOwnPropertyNames(obj);
    console.log("Own properties (with hidden attributes):")
    for (var i = 0; i < ownPropertyNames.length; i++) {
        var descriptor = Object.getOwnPropertyDescriptor(
            obj, ownPropertyNames[i]);
        console.log("  " + ownPropertyNames[i]
            + ", writable=" + descriptor.writable
            + ", enumerable=" + descriptor.enumerable
            + ", configurable=" + descriptor.configurable
            + ", value=" + descriptor.value);
    }
}
```

```
var bob = {
    firstName: "Bob",
    age: 42
};
showObjectDetails(
    bob, "bob");
```

Three Ways to Create an Object

- Object Literal Notation (prototype is Object.prototype)

```
var emptyObject = {};
var coord = { x: 5, y: 12 };
```

```
var book = { // quotes with spaces...
  "main title": "Exam 70-480",
  subtitle: "Programming with HTML5",
  "for": "Beginners",
  "!": "Special Characters"
}; // ... or reserved keywords & chars
```

- new *Constructor*

```
var o = new Object();          // prototype is Object.prototype
var d = new Date();            // prototype is Date.prototype
var a = new Array();           // prototype is Array.prototype
var r = new RegExp("[dh]og"); // prototype is RegExp.prototype
```

- Object.create(prototype)

```
var o1 = Object.create(coord);            // inherits x and y
var o2 = Object.create(null);             // no inherited properties
var o3 = Object.create(Object.prototype); // like new Object or {}
```

Encapsulating Properties

- To define an accessor property

```
function BankAccount(name, balance) {
  // “private” data property
  var _balance = 0;
  // “public” accessor property
  Object.defineProperty(this, "balance", {
    configurable: true,
    enumerable: true,
    get: function() { return _balance; },
    set: function(value) { _balance = value; }
  });
  // use the accessor property to set the initial balance
  this.balance = balance;
```

We cannot use this._balance because that would create a public property!

```
var baSmith = new BankAccount("Mr. Smith", 300);
console.log(baSmith);         // => "Mr. Smith: 300" (overridden earlier)
baSmith.balance = 400;        // calls the setter
console.log(baSmith.balance); // calls the getter
```

Object Literal Syntax Quirks

- Trailing commas are allowed in the specification, which makes it easier when re-ordering properties or adding and removing properties

```
var point = {
    x: 5,
    y: 12, // <= this comma SHOULD be allowed in all browsers
};
```

  - …but some browsers will complain
    (bow your head in shame, Internet Explorer 8!)

prototype Property

- To get the prototype of an object
  - For a non-function

```
var prototypeOfpoint = Object.getPrototypeOf(point);
```

  - For a function

```
var prototypeOfcalc = calc.prototype;
```

- Prototypes form “chains” of inheritance

```
var r = new RegExp("[dh]og");
```

Object.prototype	RegExp.prototype	r.prototype
hasOwnProperty	exec	
isPrototypeOf	test	
toString	toString	
valueOf		

An “overridden” method

Querying and Setting Properties

- To get, create, or set a property, use the dot (.) or square bracket ([]) operators

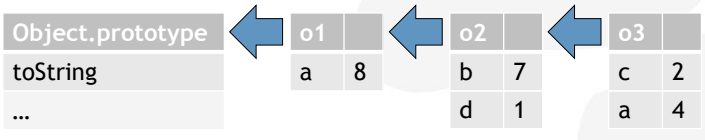
```
// must not have spaces or be a reserved word
console.log(object.property);
console.log(object["property"]); // must be a string
```

```
var title = book["main title"];
var subtitle = book.subtitle;
book["main title"] = "changed title";
book["for"] = "Advanced Students";
// creates a new property if it doesn't already exist
book.edition = 2;
```

- Getting a property searches the prototype chain until it finds the first match

Overriding Inheritance

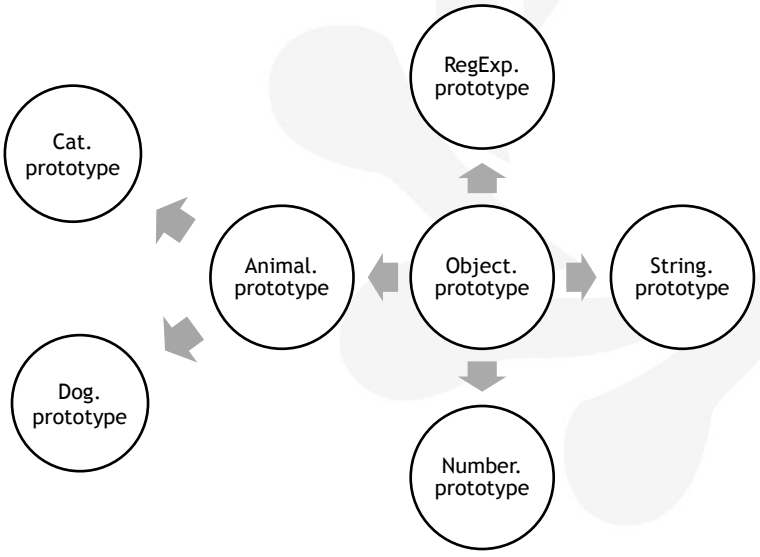
```
var o1 = {};                    // o1 inherits from Object.prototype
o1.a = 8;                       // adds property to o1
var o2 = Object.create(o1);     // o2 inherits from o1
o2.b = 7;                       // adds property to o2
var o3 = Object.create(o2);     // o3 inherits from o2
o3.c = 2;                       // adds property to o3
var s = o3.toString();          // inherited from Object.prototype
var answer = o3.a + o3.b + o3.c; // => 17
o3.a = 4;                       // overrides a by adding property to o3
var newAns = o3.a + o3.b + o3.c; // => 13
```




```
o2.d = 1;
alert(o3.d); // => 1
```

Inheritance happens when *getting* properties but overriding happens when *setting* them!

Inheritance Hierarchy



Determining the Prototype

- Prototype-base inheritance
  - If two objects inherit properties from the same prototype object, then we say that they are instances of the same class
  - They were probably created by the same factory or constructor
- To determine the prototype (and chains)

```
var ptype1 = Object.getPrototypeOf(o); // ECMAScript 5
var ptype2 = o.constructor.prototype;  // not reliable
```

```
var p = { x: 1 };
var o = Object.create(p);
console.log(p.isPrototypeOf(o));                // => true
console.log(Object.prototype.isPrototypeOf(o)); // => true
```

Deleting Properties

- The delete operator removes an own property from an object

```
delete book.subtitle;
delete book["main title"];
```

  - Calling delete on a property that does not exist silently does nothing
  - Calling delete on an inherited property silently does nothing
  - Calling delete on a property with a configurable attribute of false silently does nothing

- If you enable strict mode, modern browsers will throw an error to warn you

```
"use strict";
```

Testing for Properties

```
var o = { x: 1, y: 2 };
```


- Use **in** operator to check for own and inherited

```
console.log("x" in o);        // => true (own)
console.log("z" in o);        // => false
console.log("toString" in o); // => true (inherited)
```

- Use **hasOwnProperty()** to check for own

```
console.log(o.hasOwnProperty("x"));        // => true (own)
console.log(o.hasOwnProperty("z"));        // => false
console.log(o.hasOwnProperty("toString")); // => false (inherited)
```

- Check if it has enumerable properties

```
console.log(o.propertyIsEnumerable("toString")); // => false
console.log(o.propertyIsEnumerable("x"));        // => true
```

- Iterate through all enumerable properties using for…in

```
for (var p in o)
  console.log(p); // prints x and y but not toString
```

Objects and Prototypes

Property Descriptors and Defining Properties

```
var o = { x: 1 };
var desc = Object.getOwnPropertyDescriptor(o, "x");
console.log(desc.value);        // => 1
console.log(desc.writable);     // => true
console.log(desc.enumerable);   // => true
console.log(desc.configurable); // => true
```

```
Object.defineProperty(o, "y", { value: 2,
  writable: true, enumerable: false, configurable: true
});
```

```
console.log(o.y); // => 2
```

- If you don't specify an attribute when defining a new property it assumes false; when redefining a property you don't need to specify all the options

```
Object.defineProperty(o, "y", { writeable: false });
o.y = 3; // silently fails unless "use strict"
console.log(o.y); // => 2
```

```
Object.defineProperty(o, "y", { value: 3 });
console.log(o.y); // => 3
```

Functions and Methods

What Are They?

- In addition to its arguments, each invocation of a function has another value—its invocation context
- If a function is assigned to a property of an object it is known as a *method*
  - When a function is invoked on or through an object, that object is the invocation context (aka `this`) for the function
  - Methods cannot be overloaded in JavaScript
- Functions designed to initialize a newly created object are called *constructors*
- Functions are objects
  - Can have properties (and therefore methods!)
  - Can be passed to other functions

Functions and Methods
Invoking Them

- As functions

```
function f() { /* ... */ };
f(); // invoke function
```

- As methods

```
o.m = f;
o.m(); // invoke method
```

- As constructors (using new keyword)

```
var o = new Object(); // invoke constructor
var o2 = new Object;  // allowed if no parameters
```

- Through their call() and apply() methods
  - Both allow you to explicit specify the *this* value, and optional arguments as a list (using call) or an array (using apply)

```
f.call(o, 1, 2);    // pass parameters comma-separated
f.apply(o, [1, 2]); // pass parameters as single array
```

Call uses **C**ommas, **a**pply uses **a**rray

Constructors
Inheritance with Factories (not recommended)

- Example of using a factory function
  - Function defines instance members (usually only properties)
  - The function has a child object containing static members (usually methods but sometimes properties too)

```
function createBankAccount(name, balance) {
  // initialize object with shared "static" members
  var ba = Object.create(staticMembers);
  // define and initialize "instance" members
  ba.name = name;
  ba.balance = balance;
  return ba;
}
var staticMembers = {
  // define "static" members
  interestRate: 3.5,
  toString: function() {
    return this.name + ": " + this.balance;
  }, // and so on
```

```
// create a bank account using the factory
var baSmith = createBankAccount("Mr. Smith", 300);
console.log(baSmith); // => "Mr. Smith: 300"
```

Constructors
Inheritance with Constructors

- Constructors are invoked with the `new` keyword
  - Auto-creates the new object, sets its prototype, and returns it

```
function BankAccount(name, balance) {
  // initialize object with shared “static” members
  // var this = Object.create(BankAccount.prototype);
  // define and initialize “instance” members
  this.name = name;
  this.balance = balance;
  // return this;
}
BankAccount.prototype = {
  // since we are overwriting the existing prototype,
  // we should re-define the constructor property
  constructor: BankAccount,
  // define “static” members
  interestRate: 3.5,
  toString: function() {
    return this.name + ": " + this.balance;
  }, // and so on
```

You MUST use **new** or **this** will be the Global (window) object

```
// create a bankAccount using the constructor
var baSmith = new BankAccount("Mr. Smith", 300);
console.log(baSmith); // => "Mr. Smith: 300"
```

Constructors
Inheritance Checks

- Use the `instanceof` operator when testing for membership of a class

```
// create a bankAccount using the constructor
var baSmith = new BankAccount("Mr. Smith", 300);
console.log(baSmith); // => "Mr. Smith: 300"
// if baSmith.prototype refers to BankAccount.prototype
console.log("Is baSmith a BankAccount? " +
  (baSmith instanceof BankAccount)); // => true
```

- Note: `instanceof` operator just tests the prototype chain, so even if the object was NOT created with a constructor, as long as it has its prototype pointing to the constructor’s prototype then `instanceof` returns true

```
var baJones = Object.create(BankAccount.prototype);
baJones.name = "Mr. Smith";
// if baJones.prototype refers to BankAccount.prototype
console.log("Is baJones a BankAccount? " +
  (baJones instanceof BankAccount)); // => true
```

Constructors
Ensuring They Are Called Properly

- There is nothing special about the function itself
  - We should ensure the function is being called correctly by checking if this has the constructor as its prototype

```
function BankAccount(name, initialBalance) {
    if (this instanceof BankAccount) {
        this.accountName = name;
        this.balance = initialBalance;
    } else {
        var myCustomErrorNumber = 1000;
        throw new Error(myCustomErrorNumber, "The BankAccount function \
            should be treated as a constructor and therefore must always \
            be called with 'new'");
    }
}
```

Check that **this** has BankAccount as its prototype...

...or throw an error if not

```
try {
    var ba1 = new BankAccount("Alice", 2300); // works
    alert(ba1.toString());
    var ba2 = BankAccount("Bob", 34000); // throws error
}
catch (error) {
    alert(error.message);
}
```

Constructors
constructor Property

- By default, every prototype has one non-enumerable property called constructor, the value of which is the function used to create it
  - If you overwrite the prototype you must re-define the constructor property

```
BankAccount.prototype = {
  constructor: BankAccount, // re-define the constructor
  toString: function() {
    return this.name + ": " + this.balance;
  },
  interestRate: 3.5, // and so on
```

  - Or add static members one at a time to the prototype

```
BankAccount.prototype.toString = function () {
  return this.name + ": " + this.balance;
};
BankAccount.prototype.interestRate = 3.5; // and so on
```

Constructors
constructor Property

- Why bother setting the constructor?
  - It allows you to create a new object from an existing object

```
var p1 = new Firebrand.Person("Alice", "Smith", 23);
// create a new object using the same constructor
// that created p1
var p2 = new p1.constructor("Bob", "Jones", 28);
```

  - It allows easy access to extending the constructors prototype (and therefore all objects created using it)

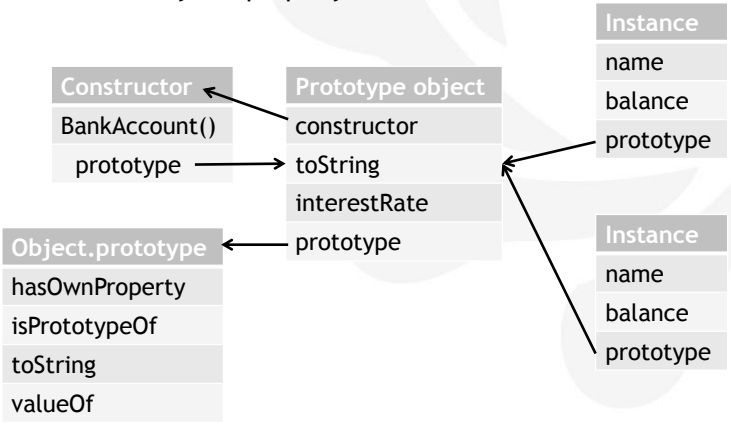
```
p1.constructor.prototype.getMarried = function(spouse) { };
p2.getMarried(p1);
```

  - Of course, if you have the original function then you could do this too

```
Firebrand.Person.prototype.getMarried = function(spouse) { };
```

Constructors
constructor Property diagram

- A constructor function, its prototype, and instances
  - Note the constructor and its prototype object point to each other so they are properly linked



Advanced Inheritance
Operators for Objects

- JavaScript does not allow operators to be defined

```
function Vector(x, y, z) {
  this.x = x;
  this.y = y;
  this.z = z;
}
```

```
var a = new Vector(2, 4, 1);
var b = new Vector(1, 3, 2);
var c = a + b; // what should + do?
if(a === b) { // what should === do?
```

- Instead, create methods for the prototype

```
Vector.prototype.add = function (v2) {
  return new Vector(this.x + v2.x,
                    this.y + v2.y,
                    this.z + v2.z);
}
```

```
var c = a.add(b);
```

```
if(a.equals(b)) {
```

```
Vector.prototype.equals = function (v2) {
  return this.x === v2.x && this.y === v2.y && this.z === v2.z;
}
```

Can I define custom operator overloads in Javascript?
http://stackoverflow.com/questions/4700085/can-i-define-custom-operator-overloads-in-javascript

Advanced Inheritance
Extending Classes

- An object inherits properties from its prototype, even if the prototype changes after the object is created
  - Therefore we can extend objects by adding new methods to their prototype objects

```
BankAccount.prototype.deposit = function (amount) {
  return this.balance += amount;
};
```

- Native JavaScript classes like Number can be augmented too

```
var n = 3;
Number.prototype.square = function () { return this * this; };
console.log(n.square()); // => 9
```

```
Number.prototype.toString = function () { return this + 1; };
var n = 9;
console.log("n is " + n.toString()); // => n is 10
```

Subclasses

- Class B can *derive from* or *subclass* another class A
  - Class B can inherit or override class A's properties and methods
  - By "class" we mean a constructor function and its prototype
- The key to subclasses in JavaScript is proper initialization of the prototype object
  - If B extends A, B.prototype must be an heir of A.prototype and B.prototype should have a property that refs the constructor

```
B.prototype = new A();
// equivalent to: B.prototype = Object.create(A.prototype);
B.prototype.constructor = B;
```

  - Without these two lines of code, the prototype object will be an ordinary object (one that inherits from Object.prototype)

```
// I also recommend adding a string to indicate the "class"
B.prototype.class = "B";
```

Chaining Constructors and Methods

- When you define a new constructor for a new class, it needs to perform all the work to initialize the object
  - Save effort by calling the superclass' constructor

```
function Person(name) { // constructor for Person
  this.name = name;
```

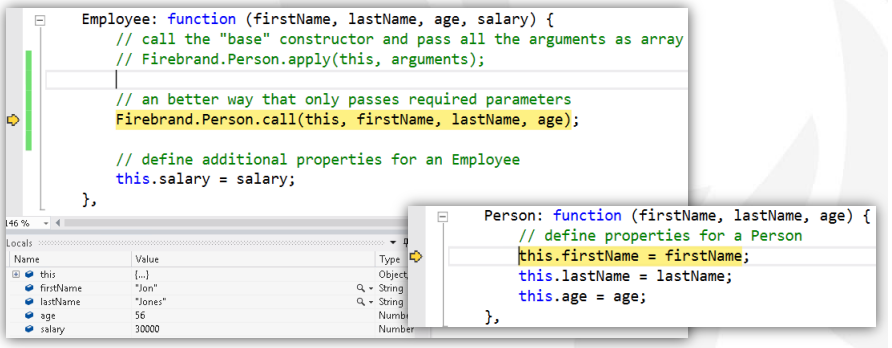
```
function Student(name, subject) {  // constructor for Student
    Person.apply(this, arguments); // call base constructor
    // or Person.call(this, name);
    this.subject = subject;
}
Student.prototype = Object.create(Person.prototype);
Student.prototype.constructor = Student;
```

```
Student.prototype.toString = function () {
    // call the base class' method
    return Person.prototype.toString.apply(
        this, arguments) + " (BSc. Hons.)";
};
```

Advanced Inheritance

Chaining Constructors Example

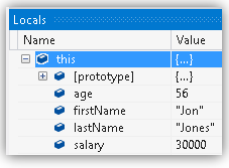
- Stepping through constructor for Employee that need to call constructor from Person

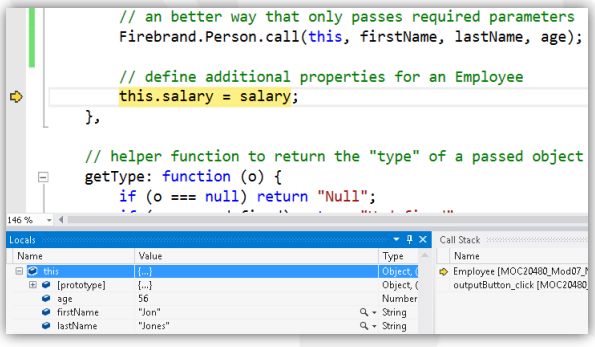


Advanced Inheritance

Chaining Constructors Example

- **this** object (the new Employee) now has the three properties created by the Person constructor (see the Locals window)





- After executing the salary line, it now has salary too

Advanced Inheritance

Namespaces and Files

- Although namespaces (actually just objects) can be used to logically group our code it is good to use separate files for each “class” of object even if they are within the same namespace
  - At the top of each file, check for the existence of the namespace or create a blank one if it’s missing

```
var Firebrand = Firebrand || { };
```

- Then add properties to the namespace, one by one, to avoid overwriting any existing ones

```
Firebrand.Person = function (...
```

```
Firebrand.Person.prototype.class = "Firebrand.Person";
```

```
Firebrand.Person.prototype.haveBirthday = function (...
```

Advanced Inheritance

JSON Deserialization and Re-associating prototype

- Object.createObject accepts a second argument
  - propertiesObject: If specified and not undefined, an object whose enumerable own properties specify property descriptors to be added to the newly-created object, with the corresponding property names

Exam Topic: none

- Therefore when deserializing we have to re-associate the correct prototype manually

```
function generatePropertyDescriptorsFor(obj) { // helper method
  var propertyDescriptors = {};
  for (var p in obj)
    propertyDescriptors[p] = Object.getOwnPropertyDescriptor(obj, p);
  return propertyDescriptors;
};
```

```
var personFromService = JSON.parse(" ... ");
var personWithPrototype = Object.create(Person.prototype,
  generatePropertyDescriptorsFor(personFromService));
```

Advanced Inheritance
Custom Events

- Events can be created with the Event constructor

```
var event = new Event('build');
// Listen for the event
elem.addEventListener('build', function (e) { /* ... */ }, false);
// Dispatch the event
elem.dispatchEvent(event);
```

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari (WebKit)
`Event()` constructor	15	11	Not supported	11.60	6

Creating and triggering events
https://developer.mozilla.org/en-US/docs/Web/Guide/Events/Creating_and_triggering_events

Advanced Inheritance
What Is a Closure?

- It incorporates a function and the parameters and local variables that existed when the closure was created
  - Normally, the local variables within a function only exist for the duration of that function's execution

```
function makeAdder(x) {
    return function (y) {
        return x + y;
    };
}
```

```
var add5 = makeAdder(5);
var add10 = makeAdder(10);
console.log(add5(2));  // 7
console.log(add10(2)); // 12
```

  - makeAdder is effectively a function factory — it creates functions which can add a specific value to their argument
  - We used it to create two new functions ("closures") — one that adds 5 to its argument, and one that adds 10; they share the same function body definition, but store different environments

Closures
https://developer.mozilla.org/en/docs/Web/JavaScript/Guide/Closures

Practice
What Is *this*?

7.47

```
<input type="button" id="myButton" value="Click Me"
       onclick="Animal();" />
<script>
    this.number = 1; // a) what is this?

    myButton.onclick = function (e) {
        this.number = 2; // b) what is this?
    };

    function Animal(name) {
        this.number = 3; // c) what is this?
        this.name = name;
        this.speak = function () {
            this.number = 4; // d) what is this?
            console.log("Hello " + this.name);
        };
    }

    var dog = new Animal("Fido");
    dog.speak();
</script>
```

Global object (Window)

myButton

The new instance of an object referenced by **dog** when called using **new**, but the **window** when called from onclick attribute!

The same as c)

Practice
Lab Alternative

7.48

- Create an object hierarchy
  - Person
    - Student
    - Employee
  - Animal
    - Dog
    - Cat
- Experiment with inheritance, overriding, and so on
- Note about the lab
  - The *inherit* method used in the lab is a custom created function for convenience, NOT a standard part of JavaScript, and only works properly with Internet Explorer!

Module 8
Creating Interactive Web Pages by Using HTML5 APIs

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Creating Interactive Web Pages by Using HTML5 APIs
Contents

Exam Topic: Implement HTML5 APIs
- ❑ Implement Geolocation API

Exam Topic: Write code that interacts with UI controls
- ❑ Implement media controls

MOC Errata, page 8-3, the 2nd slide

- The MOC says:

```
<input type="field"
```

It should have said:

```
<input type="file"
```

Geolocation API Specification
http://dev.w3.org/geo/api/spec-source.html#api_description

HTML5 Video and History: Features Users Can Really See
https://msdn.microsoft.com/en-us/magazine/dn423697.aspx

8.3

Can I use...

= Supported = Not supported = Partially supported = Support unknown

Video element - **Working Draft**

Method of playing videos on webpages (without requiring a plug-in)

*Usage stats:	Global
Support:	83.48%
Partial support:	0.22%
Total:	83.7%

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		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
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							

Geolocation - **Candidate Recommendation**

Method of informing a website of the user's geographical location

*Usage stats:	Global
Support:	83.43%
Partial support:	0.02%
Total:	83.45%

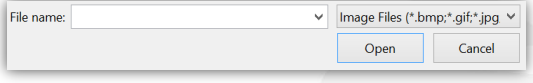
Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
								2.2		
						3.2		2.3		
						4.0-4.1		3.0		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
Current	10.0	23.0	29.0	6.0	16.0	6.0-6.1	5.0-7.0	4.2	10.0	10.0
Near future	11.0	24.0	30.0	7.0	17.0	7.0				
Farther future		25.0	31.0							

8.4

File I/O
File Pickers

- file inputs in IE10+ can have an accept attribute to control what types of file the user can pick
  - For example, to only pick image files

```
<input type="file" id="picker" accept="image/*" />
```



HTML <input> accept Attribute
http://www.w3schools.com/tags/att_input_accept.asp

File I/O
FileReader methods

8.5

- All of the read methods start reading the contents of the specified Blob or File asynchronously
  - The only difference is what the **result** attribute contains after it's finished reading

Method	result contains
readAsArrayBuffer()	ArrayBuffer representing the file's data
readAsBinaryString()	Raw binary data from the file as a string
readAsDataURL()	URL representing the file's data that can be used to set the **src** of a **img** or **video** or other element
readAsText()	Contents of the file as a text string

FileReader
https://developer.mozilla.org/en/docs/Web/API/FileReader

File I/O
URL.createObjectURL() static method

8.6

- Creates a DOMString containing an URL representing the object given in parameter
  - Represents the specified File object or Blob object

```
var img = document.createElement("img");
img.src = window.URL.createObjectURL(picker.files[0]);
img.onload = function () {
    window.URL.revokeObjectURL(this.src); // or img.src
}
```

  - Set up the image's load event handler to release the object URL, since it's no longer needed once the image is loaded

```
<input type="file" id="picker" accept="image/*" />
```

URL.createObjectURL
https://developer.mozilla.org/en-US/docs/Web/API/URL.createObjectURL

Example: Using object URLs to display images
https://developer.mozilla.org/en-US/docs/Using_files_from_web_applications#Example.3A_Using_object_URLs_to_display_images

How to Get Your Location

- To get your location once

```
navigator.geolocation.getCurrentPosition(
  successCallback, errorCallback,
  { enableHighAccuracy: true, timeout: 2000 });
```

```
function successCallback(e) {
  // e.coords.latitude
  // e.coords.longitude
  // e.coords.accuracy (meters)
```

- To get your location at a regular interval

```
var id = navigator.geolocation.watchPosition(
  successCallback, errorCallback,
  { enableHighAccuracy: true, maximumAge: 5000 });
```

IE10 uses 65 seconds for maximumAge; Android is smarter and only triggers event if position changes

```
navigator.geolocation.clearWatch(id); // to stop receiving events
```

- error.message, error.code
  - error.PERMISSION_DENIED
  - error.POSITION_UNAVAILABLE
  - error.TIMEOUT

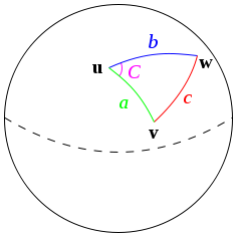
```
function errorCallback(error) {
  if(error.code === error.TIMEOUT) {
    alert(error.message);
```

Haversine formula

- An equation important in navigation, giving great-circle distances between two points on a sphere from their longitudes and latitudes

$$\operatorname{haversin}\left(\frac{d}{r}\right) = \operatorname{haversin}(\phi_2 - \phi_1) + \cos(\phi_1)\cos(\phi_2)\operatorname{haversin}(\lambda_2 - \lambda_1)$$

$$\operatorname{haversin}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2}$$



Haversine formula
https://en.wikipedia.org/wiki/Haversine_formula

Multimedia
Video Sources

8.9

- When setting the video source
  - Specify alternative sources with different formats
  - Embed Flash or Silverlight using the <object> tag
  - Embed text or a hyperlink to download the video for later

```
<video controls="controls" autoplay="autoplay">
  <source src="small.mp4" type="video/mp4" />
  <source src="small.ogv" type="video/ogg" />
  <!-- embed Flash via the object tag and set parameters -->
  <object type="application/x-shockwave-flash"
          width="0" height="0" data="small.swf">
    <param name="movie" value="small.swf" />
    <param name="quality" value="high" />
  </object>
  <!-- if browser doesn't support Flash either -->
  <a href="small.mp4">Download</a> the video as MP4.
</video>
```

Supports Mozilla and Opera

Supports Internet Explorer

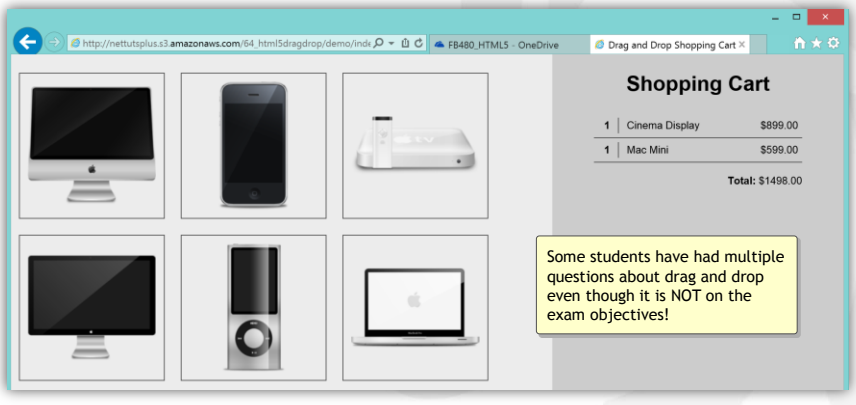
HTML5 Video
http://www.w3schools.com/html/html5_video.asp

Video for Everybody!
http://camendesign.com/code/video_for_everybody

Valid (X)HTML while embedding SWF Flash objects
https://yoast.com/articles/valid-flash-embedding/

Drag and Drop
Example

8.10



Drag and Drop Shopping Cart
http://nettutsplus.s3.amazonaws.com/64_html5dragdrop/demo/index.html

Warning! Many online drag and drop examples do not work with Internet Explorer (read this link)

Internet Explorer 9 Drag and Drop (DnD)
http://stackoverflow.com/questions/5500615/internet-explorer-9-drag-and-drop-dnd

Drag and Drop
Example (does not work with IE)

- To make an element draggable

```
el.setAttribute('draggable', 'true');
```

- To handle the dragstart event

```
addEvent(el, 'dragstart', function (e) {
  // only dropEffect='copy' will be dropable
  e.dataTransfer.effectAllowed = 'copy';
  // required otherwise doesn't work
  e.dataTransfer.setData('Text', this.id);
});
```

- To handle the dragdrop event

```
addEvent(bin, 'drop', function (e) {
  // stops the browser from redirecting...why???
  if (e.stopPropagation) e.stopPropagation();
  var el = document.getElementById(e.dataTransfer.getData('Text'));
```

Drag and drop
http://html5demos.com/drag

Module 9
Adding Offline Support to Web Applications

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Adding Offline Support to Web Applications
Contents

Exam Topic: Implement HTML5 APIs
- ❑ Implement:
  - ❑ Storage APIs
  - ❑ AppCache API

Offline web applications - **Working Draft**

Method of defining web page files to be cached using a cache manifest file, allowing them to work offline on subsequent visits to the page

*Usage stats:	Global
Support:	82.09%
Partial support:	0.07%
Total:	82.16%

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		
	8.0		31.0			4.2-4.3		4.0		
	9.0		32.0			5.0-5.1		4.1		
	10.0	28.0	33.0	5.1		6.0-6.1		4.2-4.3	7.0	
Current	11.0	29.0	34.0	7.0	20.0	7.0	5.0-7.0	4.4	10.0	10.0

lawnchair: simple json storage
http://brian.io/lawnchair/

Introduction to Web Storage
http://msdn.microsoft.com/library/cc197062(v=vs.85).aspx

9.3

Can I use...

= Supported = Not supported = Partially supported = Support unknown

Web Storage - name/value pairs - **Recommendation**

Method of storing data locally like cookies, but for larger amounts of data (sessionStorage and localStorage, used to fall under HTML5).

*Usage stats: Global

Support: 90.6%
Partial support: 0.11%
Total: 90.71%

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		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
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							

IndexedDB - **Working Draft**

Method of storing data client-side, allows indexed database queries. Previously known as WebSimpleDB API.

*Usage stats: Global

Support: 58.25%
Partial support: 2.28%
Total: 60.53%

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		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
Current	10.0	23.0	29.0	6.0	16.0	6.0-6.1	5.0-7.0	4.2	10.0 webkit	10.0
Near future	11.0	24.0	30.0	7.0	17.0	7.0				
Farther future		25.0	31.0							

9.4

Choosing Between Storage Options

- *localStorage* and *sessionStorage* both extend Storage
  - There is virtually no difference between them except for the intended “non-persistence” of sessionStorage
  - Each Storage object provides access to a list of key/value pairs, which are sometimes called items
  - Keys are strings so any string (including the empty string) is a valid key
  - Values are similarly strings
- sessionStorage represents the set of storage areas specific to the current *top-level browsing context*
- localStorage provides a Storage object for an *origin*

4.1 The Storage interface
http://www.w3.org/TR/webstorage/#the-storage-interface

9.5 ApplicationCache

- Any page the user navigates to that includes a manifest will be implicitly added to the application cache
- You can see the urls that are controlled by the application cache by visiting
  - chrome://appcache-internals/ in Chrome
- If the manifest itself returns a 404 or 410, the cache is deleted
- If the manifest or a resource specified in it fails to download, the entire cache update process fails

A Beginner's Guide to Using the Application Cache
http://www.html5rocks.com/en/tutorials/appcache/beginner/

Application Cache is a Douchebag
http://alistapart.com/article/application-cache-is-a-douchebag

9.6 Example

```
CACHE MANIFEST
# 2010-06-18:v2  Note: CACHE MANIFEST label is required

# Explicitly cached 'master entries'.
# Note: CACHE: label is optional.
CACHE:
/favicon.ico
index.html
stylesheet.css
images/logo.png
scripts/main.js

# Resources that require the user to be online.
NETWORK:
*

# static.html will be served if main.py is inaccessible
# offline.jpg will be served in place of all images in images/large/
# offline.html will be served in place of all other .html files
FALLBACK:
/main.py /static.html
images/large/ images/offline.jpg
/ /offline.html
```

9.7

Updating the Cache

- Once an application is offline it remains cached until one of the following happens:
  - The user clears their browser's data storage for your site
  - The manifest file is modified
- To programmatically check for updates to the manifest, first call applicationCache.update()
  - This will attempt to update the user's cache (which requires the manifest file to have changed)
  - When the applicationCache.status is in its UPDATEREADY state, calling applicationCache.swapCache() will swap the old cache for the new one

9.8

applicationCache Events

Event	Description
cached	Fired after the first cache of the manifest
checking	Checking for an update
downloading	An update was found; the browser is fetching resources
error	The manifest returns 404 or 410, the download failed, or the manifest changed while the download was in progress
noupdate	Fired after the first download of the manifest
obsolete	Fired if the manifest file returns a 404 or 410 which results in the application cache being deleted
progress	Fired for each resource listed in the manifest as it is being fetched
updateready	Fired when the manifest resources have been newly redownloaded, so you can now call swapCache()

Module 10 Implementing an Adaptive User Interface

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Implementing an Adaptive User Interface
Contents

Exam Topic: Create an animated and adaptive UI
- ❑ Adjust UI based on media queries (device adaptations for output formats, displays, and representations)
- ❑ Hide or disable controls

Exam Topic: Apply styling to HTML elements programmatically
- ❑ Change the location of an element
- ❑ Show and hide elements

- Page 10.8, position 12-3195
  - Code example has three CSS selectors for ".article" which should be "article" because it needs to select an element with tag name of article, NOT a class name of article
  - The media queries should use **min-width** instead of **max-width**
- Page 10.20, position 12-4222
  - Code should use ::before and ::after
  - Single-colon (:) is supported for backwards compatibility only

Can I use...

= Supported = Not supported = Partially supported = Support unknown

CSS3 Media Queries - **Recommendation**

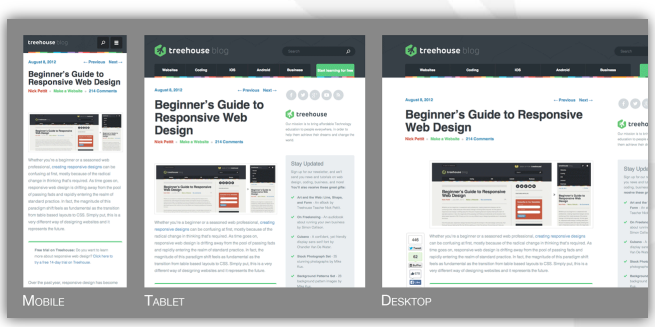
Method of applying styles based on media information. Includes things like page and device dimensions

*Usage stats:	Global
Support:	86.55%
Partial support:	0.01%
Total:	86.56%

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		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
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							

Responsive Design
What Is It?

- Responsive web design is a technique for building websites that work on mobile devices, tablets, and desktop screens



The 2014 Guide to Responsive Web Design
http://blog.teamtreehouse.com/modern-field-guide-responsive-web-design

Dynamic HTML
Add/Remove from Layout, Hide/Show Elements

10.5

HTML

```
<div class="removeMe">Hello</div>
```

```
<!-- to disable a control in HTML -->
<input type="button" disabled />
```

```
<div class="hideMe">Hello</div>
```

CSS

```
/* to remove from layout */
.removeMe {
    display: none;
}
```

```
/* to hide */
.hideMe {
    visibility: hidden;
}
```

Note: disabled cannot be set in CSS

JavaScript

```
// to disable a control
element.disabled = true;
// to enable an control
element.disabled = false;
```

```
// to remove an element from layout
element.style.display = "none";
// to add an element back to layout
element.style.display = "block"; // or others
// to hide an element
element.style.visibility = "hidden";
// to show an element
element.style.visibility = "visible";
```

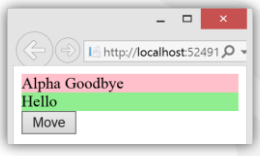
Dynamic HTML
Move and Copy Elements

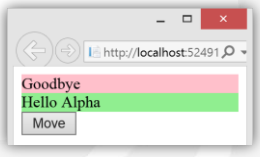
10.6

Use the DOM to move an element

```
<div style="background-color:pink">
  <span id="A">Alpha</span>
  <span>Goodbye</span>
</div>
<div style="background-color:lightgreen" id="B">
  <span>Hello</span>
</div>
```

```
B.appendChild(A);
```





To copy an element, clone it and append the clone

```
var copy = A.cloneNode(true); // include descendants
B.appendChild(copy);
```

HTML DOM cloneNode() Method
http://www.w3schools.com/jsref/met_node_clonenode.asp

CSS
Printing

10.7

- Style and link elements support the **media** attribute, which defines the output device for the style sheet
  - Values are screen (default), print, all, and many others
  - The print value specifies that the style sheet is used when the page is printed or print-previewed and does not affect how the document will be displayed onscreen

```
<style type="text/css" media="print">
    div.page {
        page-break-before: always;
    }
</style>
```

Printing and Style Sheets
https://msdn.microsoft.com/en-us/library/ms533037%28v=vs.85%29.aspx

CSS
Media Queries

10.8

- Different style sheets for different scenarios

```
<link rel='stylesheet' media='only screen and (max-width: 700px)'
      href='css/narrow.css' />
```

CSS Specification: "The keyword 'only' can also be used to hide style sheets from older user agents. User agents must process media queries starting with 'only' as if the 'only' keyword was not present."

```
<link rel='stylesheet'
      media='only screen and (min-width: 701px) and (max-width: 900px)'
      href='css/medium.css' />
```

- Although media queries support the keywords "and" and "not", they do not support the "or" keyword
  - Use a comma-separated list (MOC is wrong: position 12, 2870)

```
@media screen and (max-width: 995px), screen and (max-height: 700px) {
   /* rules for either media query */
```

CSS Media Queries & Using Available Space
http://css-tricks.com/css-media-queries/

CSS
Media Queries and Specificity

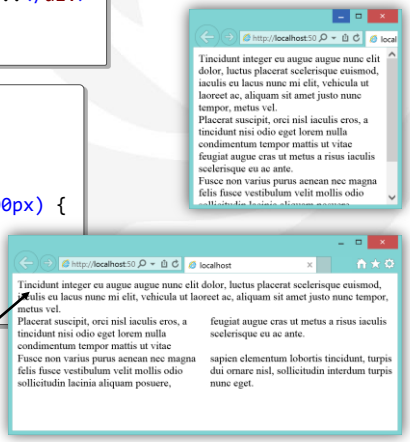
10.9

- If a selector in a media query has a lower specificity than a default applied selector then it won't be applied

```
<div class="cool">Tincidunt eu ...</div>
<div>Placerat, orci ...</div>
<div>Fusce non varius ...</div>
```

```
div.cool {
    column-count: 1;
}

@media screen and (min-width: 500px) {
    div {
        column-count: 2;
    }
}
```



First paragraph still has one column because its selector is more specific than the one in the media query

Conditional Comments
Not Supported in Internet Explorer 10 and later

10.10

- Support for conditional comments has been removed in Internet Explorer 10 standards and quirks modes for improved interoperability and compliance with HTML5
  - …but you may opt into Internet Explorer 9 behaviour

```
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE9">
```

- Or use feature detection (see link below)

```
<!--[if !IE]-->
IE ignores this
<!--[endif]-->
```

- MOC is wrong on position 12, 3689
  - Cannot use MOCs syntax because non-IE browsers will not recognize conditional comments! Should have been this:

```
<!--[if !IE]--><link href="..." rel="stylesheet" /><!--[endif]-->
```

How to Detect Features Instead of Browsers
http://msdn.microsoft.com/en-US/library/hh273397.aspx

Conditional comments are no longer supported
http://msdn.microsoft.com/en-us/library/ie/hh801214(v=vs.85).aspx

Module 11
Creating Advanced Graphics

Programming in HTML5 with
JavaScript and CSS3

Updated 15th August 2015

Creating Advanced Graphics
Contents

Exam Topic: Write code that interacts with UI controls
- ❑ Implement HTML5 canvas and SVG graphics

SVG

- No z-order in SVG so you must re-arrange elements in DOM

Can I use...

= Supported = Not supported = Partially supported = Support unknown

SVG (basic support) - Recommendation

Method of displaying basic Vector Graphics features using the embed or object elements

*Usage stats:	Global
Support:	84.48%
Partial support:	0.02%
Total:	84.5%

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		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
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							

Canvas (basic support) - Candidate Recommendation

Method of generating fast, dynamic graphics using JavaScript

*Usage stats:	Global
Support:	83.87%
Partial support:	2.81%
Total:	86.68%

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		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
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							

Image Maps

What Are They?

- A technology for layering clickable regions onto a static image
  - img element references the map
  - map element defines clickable regions with hrefs

```
<img src="planets.gif" alt="Planets" usemap="#planetmap">
<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun">
  <area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury">
  <area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus">
</map>
```

HTML <map> Tag
http://www.w3schools.com/tags/tag_map.asp

canvas
Basics

11.5

- Get canvas and its 2D context, set fill and stroke styles

```
<canvas width="300" height="300" id="mycanvas"></canvas>
```

```
var context = mycanvas.getContext('2d');
context.fillStyle = 'yellow'; // can be gradients and patterns too
context.strokeStyle = 'blue'; // createLinearGradient, createPattern
```

- To define a path and then draw the fill/stroke

```
context.rect(50, 50, 150, 100); // define a path (not visible yet)
context.fill();   // fill the path
context.stroke(); // draw stroke for path
```

- To define a path and fill/stroke in one method call

```
context.fillRect(50, 50, 150, 100);
context.strokeRect(50, 50, 150, 100);
```

Drawing squares on a canvas
http://falcon80.com/HTMLCanvas/BasicShapes/Square.html

canvas
Complex Paths

11.6

- Define paths by using
  - rect(), arc()
  - beginPath() moveTo(), lineTo(), arcTo(), bezierCurveTo(), quadraticCurveTo, closePath(), and so on

```
// define a complex path
context.beginPath();
context.moveTo(50, 50);
context.lineTo(200, 50);
context.lineTo(200, 150);
context.lineto(50, 150);
context.closePath();
```

- You can define multiple parts of a path and then draw them all at once

```
context.rect(50, 50, 150, 100); // define a path
context.rect(100, 20, 100, 80); // made up of
context.rect(70, 70, 40, 60);   // three rects
context.stroke(); // draw stroke for complete path
```

- To check if a point is inside a path

```
if (context.isPointInPath(80, 70)) {
```

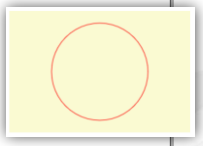
HTML Canvas Reference
http://www.w3schools.com/tags/ref_canvas.asp

canvas
Height and Width

11.7

- The **height** and **width** attributes of the canvas are the *logical* height and width and are different from the *style* **height** and **width** that can be used to scale

```
<canvas id="mycanvas" height="200" width="300"
    style="background-color: lightgoldenrodyellow;"></canvas>
<script>
    var context = mycanvas.getContext("2d");
    // x, y, radius, startAngle, endAngle
    context.arc(150, 100, 80, 0, 360);
    context.strokeStyle = "red";
    context.stroke();
</script>
```



```
<canvas id="mycanvas" height="200" width="300"
    style="background-color: lightgoldenrodyellow; width:100px; height:100px;">
</canvas>
```

Canvas width and height in HTML5
http://stackoverflow.com/questions/4938346/canvas-width-and-height-in-html5

canvas
Transformations

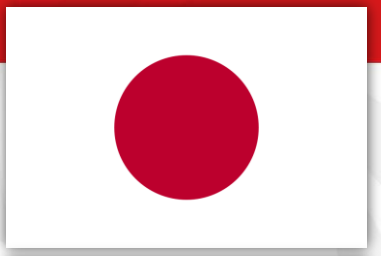
11.8

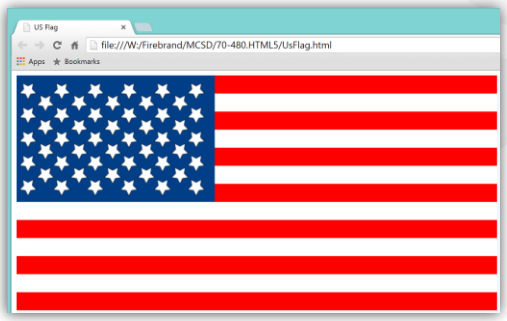
	Syntax
To shrink or grow	*context*.scale(*scalewidth*, *scaleheight*);
To rotate	*context*.rotate(*angle*);
To move	*context*.translate(*x*, *y*);
To transform using a matrix	*context*.transform(scaleX, skewX, skewY, scaleY, translateX, translateY);

Transformations
https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Transformations

Lab Alternative

- Use canvas and SVG to draw the flags for
  - Modern Japan
  - Arab Revolt of 1916
  - US Flag





Module 12
Animating the User Interface

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Animating the User Interface

Contents

Exam Topic: Apply styling to HTML elements programmatically
- ❑ Apply a transform

Exam Topic: Create an animated and adaptive UI
- ❑ Animate objects by applying CSS transitions
- ❑ Apply 3-D and 2-D transformations

MOC: the 3D cube needs the following for Chrome

```
#container {
    /* other stuff */
    -webkit-transition: -webkit-transform 5s;
}

    #container:hover {
        /* other stuff */
        -webkit-transform: rotate(90deg);
    }
```

Can I use...

= Supported = Not supported = Partially supported = Support unknown

CSS3 Transitions - **Working Draft**

*Usage stats: Global
Support: 79.04%

Simple method of animating certain properties of an element

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1 -webkit-		
								2.2 -webkit-		
						3.2 -webkit-		2.3 -webkit-		
						4.0-4.1 -webkit-		3.0 -webkit-		
	8.0					4.2-4.3 -webkit-		4.0 -webkit-		
	9.0	22.0	28.0	5.1 -webkit-		5.0-5.1 -webkit-		4.1 -webkit-	7.0 -webkit-	
Current	10.0	23.0	29.0	6.0 -webkit-	16.0	6.0-6.1 -webkit-	5.0-7.0	4.2 -webkit-	10.0 -webkit-	10.0
Near future	11.0	24.0	30.0	7.0	17.0	7.0				
Farther future		25.0	31.0							

Using transitions to make JavaScript functionality smooth
http://jsfiddle.net/RwtHn/5/

Transforms

Transforming Elements

- Applies a 2D or 3D transformation to an element
  - Allows you to rotate, scale, move, skew, etc., elements

```
div {
    -ms-transform: rotate(7deg); /* IE 9 */
    -webkit-transform: rotate(7deg); /* Chrome, Safari, Opera */
    transform: rotate(7deg);
}
```

CSS3 transform Property
http://www.w3schools.com/cssref/css3_pr_transform.asp

Transitions
Example

12.5

- A blue box that doubles in size, rotates 180°, and changes to red when the mouse hovers over it

```
<div id="box"></div>
```

```
#box {
  border-style: solid;
  border-width: 1px;
  width: 100px;
  height: 100px;
  background-color: #0000FF;
  -moz-transition: width 2s, height 2s, background-color 2s,
    -moz-transform 2s;
  -webkit-transition: width 2s, height 2s, background-color 2s,
    -webkit-transform 2s;
  transition: width 2s, height 2s, background-color 2s, transform 2s;
}
```

```
#box:hover {
  background-color: #FFCCCC;
  width: 200px;
  height: 200px;
  -moz-transform: rotate(180deg);
  -webkit-transform: rotate(180deg);
  transform: rotate(180deg);
}
```

Using CSS transitions
https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_CSS_transitions

Transitions
Forwards and Backwards

12.6

- Forwards-only transition

```
#box {
  /* switch back to red instantly */
  background-color: red;
```

```
#box:hover {
  /* transition to blue in 2s */
  background-color: blue;
  transition: background-color 2s;
```

- Forwards and backwards using same transition

```
#box {
  /* transition back to red in 2s */
  background-color: red;
  transition: background-color 2s;
```

```
#box:hover {
  /* transition to blue in 2s */
  background-color: blue;
```

- Forwards and backwards using different transitions

```
#box {
  /* transition back to red in 2s */
  background-color: red;
  transition: background-color 2s;
```

```
#box:hover {
  /* transition to blue in 4s */
  background-color: blue;
  transition: background-color 4s;
```

Defining Keyframes

- To make h1s slide in from right-to-left
  - Optionally, add a keyframe so that three quarters of the way through the animation it trebles the size of the font
  - Add an iteration count and direction to repeat the animation

```
h1 {
    animation-duration: 3s;
    animation-name: slidein;
}
@keyframes slidein {
    from {
        margin-left: 100%;
    }

    to {
        margin-left: 0%;
    }
}
```

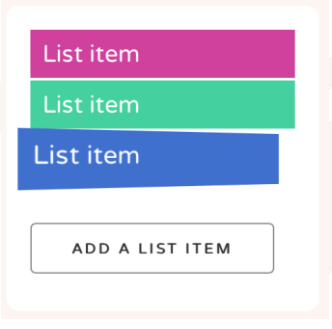
```
animation-iteration-count: infinite;
animation-direction: alternate;
```

```
75% {
    font-size: 300%;
    margin-left: 25%;
}
```

Using CSS animations
https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_CSS_animations

Animating List Items

- When parts of a web page change, adding some animation is a good way to help your viewers understand what's going on
  - Animations can announce the arrival of new content, or draw attention to content that's in the process of being removed



Animating List Items
https://cssanimation.rocks/list-items/

Module 13
Implementing Real-time Communication by Using Web Sockets

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Implementing Real-time Communication by Using Web Sockets
Contents

Exam Topic: Implement a callback
- ❑ Receive messages from the HTML5 WebSocket API

Web Sockets - Candidate Recommendation

Bidirectional communication technology for web apps

*Usage stats:	Global
Support:	69.72%
Partial support:	3.25%
Total:	72.97%

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		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
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							

= Supported = Not supported = Partially supported = Support unknown

The WebSocket API
http://msdn.microsoft.com/library/ie/hh673567(v=vs.85).aspx

13.3

Web Sockets
When To Use Web Sockets

- Achieving zero-lag connectivity between Web clients and servers requires going beyond the HTTP protocol
  - The new WebSocket Protocol aims to overcome a structural limitation of the HTTP protocol that makes it inefficient for Web applications hosted in browsers to stay connected to the server over a persistent connection
- Great for *real-time* communication and updates

Understanding the Power of WebSockets
http://msdn.microsoft.com/en-us/magazine/hh975342.aspx

13.4

Web Sockets
On the Server

Exam Topic: none

- Add the Microsoft WebSockets NuGet package

```
public class ChatController : ApiController {
  public HttpResponseMessage Get(string username) {
    HttpContext.Current.AcceptWebSocketRequest(new ChatWebSocketHandler());
    return Request.CreateResponse(HttpStatusCode.SwitchingProtocols);
  }
}
```

```
using Microsoft.Web.WebSockets;
```

```
public class ChatWebSocketHandler : WebSocketHandler {
    public ChatWebSocketHandler() {
      // constructor
    }
    public override void OnOpen() {
      // triggered when a connection is opened
    }
    public override void OnClose() {
      // triggered when a connection is closed
    }
    public override void OnMessage(string message) {
      // triggered when client sends a message
    }
}
```

System.Web.WebSockets Namespace
http://msdn.microsoft.com/en-us/library/system.web.websockets(v=vs.110).aspx

Web Sockets
On the Browser

13.5

- Create and use a WebSocket object

```
$(function () {
    websocket = new WebSocket('ws://localhost/api/ChatController');
    websocket.onopen = function () {
        // when a new connection is opened
    };
    websocket.onclose = function (e) {
        // e.reason, e.code, e.wasClean
    };
    websocket.onmessage = function (e) {
        // e.type, e.data
    };
});
```

```
$('#sendMessageButton').on('click', function () {
    websocket.send('Hello from Client');
});
```

```
$('#closeButton').on('click', function () {
    websocket.close();
});
```

Web Sockets
SignalR

13.6

- ASP.NET SignalR is a library for ASP.NET developers that makes it incredibly simple to add real-time web functionality to your applications
  - SignalR will use WebSockets under the covers when it's available
  - SignalR will fallback to other techniques and technologies when it isn't
  - Your application code stays the same

70-480 Exam Topic: none

70-486 and 70-487 Exam Topic: since April 2014

Learn About ASP.NET SignalR
http://www.asp.net/signalr

SignalR

What Is SignalR 2.0?

13.7

- Incredibly simple real-time web for .NET
  - Ability to have your server-side code push content to the connected clients as it happens, in real-time
  - SignalR will use WebSockets under the covers when it's available, and gracefully fallback to other technologies when it isn't, while your application code stays the same
- Install it with NuGet

```
Install-Package Microsoft.AspNet.SignalR
```

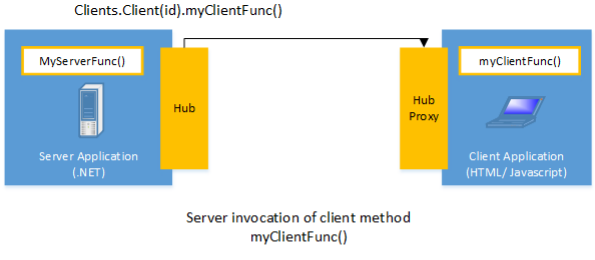
- Install a sample application

```
Install-Package Microsoft.AspNet.SignalR.Sample
```

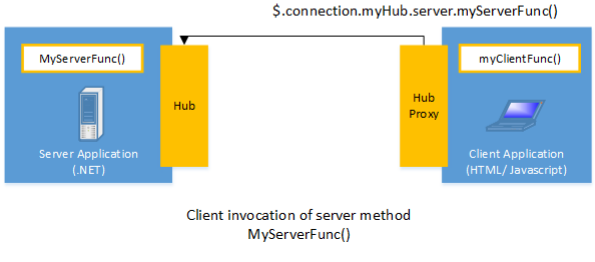
SignalR

Communication

13.8



Server invocation of client method
myClientFunc()



Client invocation of server method
MyServerFunc()

SignalR provides a simple API for creating server-to-client remote procedure calls (RPC) that call JavaScript functions in client browsers from server-side .NET code

SignalR
Transport Selection Process

13.9

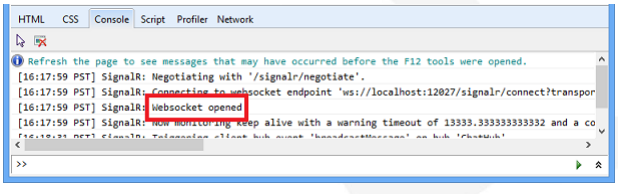
- Steps that SignalR uses to decide which transport to use
  - If the browser is IE8 or earlier, Long Polling is used
  - If JSONP is configured (that is, the jsonp parameter is set to true when the connection is started), Long Polling is used
  - If a cross-domain connection is being made then WebSocket will be used if the client supports CORS and both support WebSocket
  - If JSONP is not configured and the connection is not cross-domain, WebSocket will be used if both the client and server support it
  - If either the client or server do not support WebSocket, Server Sent Events is used if it is available
  - If Server Sent Events is not available, Forever Frame is attempted
  - If Forever Frame fails, Long Polling is used

SignalR
Monitoring Transports

13.10

- You can determine what transport your application is using by enabling logging on your hub

```
$.connection.hub.logging = true;
```



- You can request transport preferences

```
connection.start({ transport: ['webSockets','longPolling'] });
```

Tutorial: Getting Started with SignalR 2.0 and MVC 5
http://www.asp.net/signalr/overview/signalr-20/getting-started-with-signalr-20/tutorial-getting-started-with-signalr-20-and-mvc-5

Module 14
Performing Background Processing by Using Web Workers

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

Performing Background Processing by Using Web Workers
Contents

Exam Topic: Create a web worker process

- ❑ Start and stop a web worker
- ❑ Pass data to a web worker
- ❑ Configure timeouts and intervals on the web worker
- ❑ Register an event listener for the web worker
- ❑ Limitations of a web worker

Web Workers
http://msdn.microsoft.com/library/ie/hh673568(v=vs.85).aspx

Can I use...

= Supported = Not supported = Partially supported = Support unknown

Web Workers - Candidate Recommendation

*Usage stats: Global Support: 73.39%

Method of running scripts in the background, isolated from the web page

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		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
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							

Shared Web Workers - Candidate Recommendation

*Usage stats: Global Support: 46.68%

Method of allowing multiple scripts to communicate with a single web worker.

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		
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
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							

Web Workers

What Can You Use Inside a Web Worker?

- Worker has a Global scope separate from the page
- Worker can use
  - Most JavaScript APIs: navigator.location, XMLHttpRequest, etc.
  - External script libraries: importScripts(*url*)
- Major limitations
  - Worker code cannot access the DOM or window
  - Worker has a single thread, so if it could have multiple messages posted to it simultaneously, it should process and return as quickly as possible, otherwise the messages will be queued up (or the worker could spawn additional workers)

Functions available to workers
https://developer.mozilla.org/en-US/docs/Web/Guide/Needs_categorization/Functions_available_to_workers

Web Workers
Message Passing

14.5

- In both the page and the worker
  - Handle the *message* (aka *onmessage*) event to receive messages
  - Call the *postMessage()* method to send messages
- In the page: create Worker instances, post objects to them, and handle received objects

```
if (Worker) {
  var theWorker = new Worker("worker.js");
  theWorker.addEventListener("message", theWorker_message, false);
}
```

```
function theWorker_message(e) {
  // process e.data
```

```
function squareANumberButton_click() {
  theWorker.postMessage({ task: "SQUARE", value: 5 });
} // pass an object with custom properties
```

```
function stopWorkerButton_click() {
  theWorker.postMessage({ task: "STOP" });
}
```

```
function killButton_click() {
  theWorker.terminate();
} // not recommended
```

Web Workers
The Web Worker

14.6

- Handle receiving messages from the page

```
self.addEventListener("message", self_message, false);
// or use: self.onmessage = self_message;
// or use: onmessage = self_message;
```

Best: use **self** to refer to the worker
Alternative: use **this** or nothing at all

```
function self_message(e) {
  switch (e.data.task) { // data is an object posted
    case "SQUARE":       // with custom properties
        var number = e.data.value * e.data.value;
        self.postMessage({ answer: number });
        break;
    case "ERROR":
        throw new Error("Throwing a simulated error as requested");
        break;
    case "STOP":      // to safely stop the worker
        self.close(); // use self.close() inside a worker
        break;        // but terminate() outside
    default:
        self.postMessage({ answer: 0 });
  }
}
```

```
importScripts("jquery-1.7.1.js");
```

Web Workers
Handling Errors

14.7

- Handle receiving errors throw in the worker

```
// on the page
worker.addEventListener("error", function (e) {
    messages.innerHTML = "<div>Error on line " + e.lineno
        + ": " + e.message + " in file " + e.filename + "</div>";
}
```

- Note: the worker does NOT terminate when an error is thrown

Web Workers
Message Serialization

14.8

- The object passed is duplicated using serialization (typically by using the structured clone algorithm) that has less limits than JSON
  - As well as being able to serialize everything that JSON can, it also serializes RegExp, Blob, File, FileList, ImageData
  - It can correctly serialize cyclic graphs of references
- Things that cannot be passed to a web worker
  - Function, Error, and DOM Node instances
  - Property descriptors are reset to defaults
  - Prototype chain is broken

Web Workers
Timeouts and Intervals

- Workers can use timeouts and intervals just like a page can
- Timeouts execute once after *n* milliseconds
  - To create a 5 second timeout and then post a message

```
var timeoutID = setTimeout(self.postMessage, 5000,
  { message: "Finished long operation." });
```

  - To cancel a timeout

```
clearTimeout(timeoutID);
```

- Intervals execute repeatedly every *n* milliseconds
  - id = setInterval(function, milliseconds, object)
  - clearInterval(id)

Web Workers
Image Processing

- Image processing can be done by a web worker and an HTML5 canvas

```
var tempContext = canvas.getContext("2d");
```

```
var canvasData = tempContext.getImageData(
    0, blockSize * index, canvas.width, blockSize);
```

```
worker.postMessage(
    { data: canvasData, index: index, length: segmentLength });
```

Using Web Workers to improve performance of image manipulation
http://www.htmlgoodies.com/html5/client/using-web-workers-to-improve-performance-of-image-manipulation.html#fbid=eqyeRapQM71

Promises and Deferred
What Are They?

14.11

Exam Topic: none

- "Promises" represent the next great paradigm in JavaScript programming
  - A promise represents the result of an action, which may or may not have completed (similar to a Task<T> in C#)
  - A promise has a function called *then*, which can be given callbacks to be called when the promise is fulfilled or has failed

```
// old way using callbacks
entity.save({ key: value }, {
  success: function (result) {
    // the object was saved
  },
  error: function (error) {
    // saving the object failed
  }
});
```

```
// new way using promises
entity.save({ key: value })
  .then(function (result) {
    // the object was saved
  })
  .fail(function (error) {
    // saving the object failed
  });
```

What's so great about JavaScript Promises?
http://blog.parse.com/2013/01/29/whats-so-great-about-javascript-promises/

Promises and Deferred
What's the Big Deal?

14.12

Exam Topic: none

- The real power of promises comes from chaining many of them together
  - Calling promise.then(func) returns a new promise, which is not fulfilled until func has completed

```
Parse.User.logIn("user", "pass")
  .then(function (user) {
    return query.find(user);
  })
  .then(function (results) {
    return results[0].save({ key: value });
  })
  .then(function (result) {
    // the object was saved
  });
```

  - jQuery's $.Deferred (done, fail, always, then, and so on)
  - Microsoft's WinJS.Promise

jQuery, HTML5, CSS3 Reference Guide

March 17

2013

The document covers some of the main concepts of jQuery, HTML5, and CSS3 properties, elements, selectors, and methods. It also includes the main syntax selectors of Regular Expressions.

Appendix A

Table of Contents

Regular Expressions

A regular expression is a pattern describing a certain amount of text. Their name comes from the mathematical theory on which they are based.

Characters

Character	Description	Example
Any character except `[\^$.|?*+()`	All characters except the listed special characters match a single instance of themselves. { and } are literal characters, unless they're part of a valid regular expression token (e.g. the `{n}` quantifier).	`a` matches `a`
\ (backslash) followed by any of `[\^$.|?*+(){}`	A backslash escapes special characters to suppress their special meaning.	`\+` matches `+`
`\Q...\E`	Matches the characters between `\Q` and `\E` literally, suppressing the meaning of special characters.	`\Q+-*/\E` matches `+-*/`
`\xFF` where FF are 2 hexadecimal digits	Matches the character with the specified ASCII/ANSI value, which depends on the code page used. Can be used in character classes.	`\xA9` matches © when using the Latin-1 code page.
`\n`, `\r` and `\t`	Match an LF character, CR character and a tab character respectively. Can be used in character classes.	`\r\n` matches a DOS/Windows CRLF line break.
`\a`, `\e`, `\f` and `\v`	Match a bell character (`\x07`), escape character (`\x1B`), form feed (`\x0C`) and vertical tab (`\x0B`) respectively. Can be used in character classes.	
`\cA` through `\cZ`	Match an ASCII character Control+A through Control+Z, equivalent to `\x01` through `\x1A`. Can be used in character classes.	`\cM\cJ` matches a DOS/Windows CRLF line break.

Character Sets

Character	Description	Example
`[` (opening square bracket)	Starts a character class. A character class matches a single character out of all the possibilities offered by the character class. Inside a character class, different rules apply. The rules in this section are only valid inside character classes. The rules outside this section are not valid in character classes, except for a few character escapes that are indicated with "can be used inside character classes".	
Any character except `^-]\`	All characters except the listed special characters.	`[abc]` matches `a`, `b` or `c`
`\` (backslash) followed by any of `^-]\`	A backslash escapes special characters to suppress their special meaning.	`[\^\]]` matches `^` or `]`
`-` (hyphen) except immediately after the opening `[`	Specifies a range of characters. (Specifies a hyphen if placed immediately after the opening `[`)	`[a-zA-Z0-9]` matches any letter or digit
`^` (caret) immediately after the opening `[`	Negates the character class, causing it to match a single character *not* listed in the character class. (Specifies a caret if placed anywhere except after the opening `[`)	`[^a-d]` matches `x` (any character except a, b, c or d)
`\d`, `\w` and `\s`	Shorthand character classes matching digits, word characters (letters, digits, and underscores), and whitespace (spaces, tabs, and line breaks). Can be used inside and outside character classes.	`[\d\s]` matches a character that is a digit or whitespace
`\D`, `\W` and `\S`	Negated versions of the above. Should be used only outside character classes. (Can be used inside, but that is confusing.)	`\D` matches a character that is not a digit
`[\b]`	Inside a character class, `\b` is a backspace character.	`[\b\t]` matches a backspace or tab character

Dots

Character	Description	Example
. (dot)	Matches any single character except line break characters \r and \n. Most regex flavors have an option to make the dot match line break characters too.	`.` matches `x` or (almost) any other character

Anchors

Character	Description	Example
^ (caret)	Matches at the start of the string the regex pattern is applied to. Matches a position rather than a character. Most regex flavors have an option to make the caret match after line breaks (i.e. at the start of a line in a file) as well.	`^.` matches `a` in `abc\ndef`. Also matches `d` in "multi-line" mode.
$ (dollar)	Matches at the end of the string the regex pattern is applied to. Matches a position rather than a character. Most regex flavors have an option to make the dollar match before line breaks (i.e. at the end of a line in a file) as well. Also matches before the very last line break if the string ends with a line break.	`.$` matches `f` in `abc\ndef`. Also matches `c` in "multi-line" mode.
\A	Matches at the start of the string the regex pattern is applied to. Matches a position rather than a character. Never matches after line breaks.	`\A.` matches `a` in `abc`
\Z	Matches at the end of the string the regex pattern is applied to. Matches a position rather than a character. Never matches before line breaks, except for the very last line break if the string ends with a line break.	`.\Z` matches `f` in `abc\ndef`
\z	Matches at the end of the string the regex pattern is applied to. Matches a position rather than a character. Never matches before line breaks.	`.\z` matches `f` in `abc\ndef`

Word Boundaries

Character	Description	Example
\b	Matches at the position between a word character (anything matched by \w) and a non-word character (anything matched by [^\w] or \w) as well as at the start and/or end of the string if the first and/or last characters in the string are word characters.	.\b matches c in abc
\B	Matches at the position between two word characters (i.e the position between \w\w) as well as at the position between two non-word characters (i.e. \W\W).	\B.\B matches b in abc

Alternation

Character	Description	Example
| (pipe)	Causes the regex engine to match either the part on the left side, or the part on the right side. Can be strung together into a series of options.	abc|def|xyz matches abc, def or xyz
| (pipe)	The pipe has the lowest precedence of all operators. Use grouping to alternate only part of the regular expression.	abc(def|xyz) matches abcdef or abcxyz

Quantifiers

Character	Description	Example
? (question mark)	Makes the preceding item optional. Greedy, so the optional item is included in the match if possible.	`abc?` matches `ab` or `abc`
??	Makes the preceding item optional. Lazy, so the optional item is excluded in the match if possible. This construct is often excluded from documentation because of its limited use.	`abc??` matches `ab` or `abc`
* (star)	Repeats the previous item zero or more times. Greedy, so as many items as possible will be matched before trying permutations with less matches of the preceding item, up to the point where the preceding item is not matched at all.	`".*"` matches `"def" "ghi"` in `abc "def" "ghi" jkl`
*? (lazy star)	Repeats the previous item zero or more times. Lazy, so the engine first attempts to skip the previous item, before trying permutations with ever increasing matches of the preceding item.	`".*?"` matches `"def"` in `abc "def" "ghi" jkl`
+ (plus)	Repeats the previous item once or more. Greedy, so as many items as possible will be matched before trying permutations with less matches of the preceding item, up to the point where the preceding item is matched only once.	`".+"` matches `"def" "ghi"` in `abc "def" "ghi" jkl`
+? (lazy plus)	Repeats the previous item once or more. Lazy, so the engine first matches the previous item only once, before trying permutations with ever increasing matches of the preceding item.	`".+?"` matches `"def"` in `abc "def" "ghi" jkl`
`{n}` where n is an integer >= 1	Repeats the previous item exactly n times.	`a{3}` matches `aaa`
`{n,m}` where n >= 0 and m >= n	Repeats the previous item between n and m times. Greedy, so repeating m times is tried before reducing the repetition to n times.	`a{2,4}` matches `aaaa`, `aaa` or `aa`
`{n,m}?` where n >= 0 and m >= n	Repeats the previous item between n and m times. Lazy, so repeating n times is tried before increasing the repetition to m times.	`a{2,4}?` matches `aa`, `aaa` or `aaaa`

`{n,}` where n >= 0	Repeats the previous item at least n times. Greedy, so as many items as possible will be matched before trying permutations with less matches of the preceding item, up to the point where the preceding item is matched only n times.	`a{2,}` matches `aaaaa` in `aaaaa`
`{n,}?` where n >= 0	Repeats the previous item n or more times. Lazy, so the engine first matches the previous item n times, before trying permutations with ever increasing matches of the preceding item.	`a{2,}?` matches `aa` in `aaaaa`

Regular Expressions Examples

Grabbing HTML Tags

<TAG\b[^>]*>(.*?)</TAG>

Matches the opening and closing pair of a specific HTML tag.

Trimming Whitespace

^[\t]+|[\t]+$

Trims unnecessary white space from the beginning and end of a line.

Matching IP Addresses

\b(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b (single line)

Matches any IP address and restricts all 4 numbers to between 0..255.

Numeric Ranges

[0-255]

The expression does not match any number between 0 and 255. Since it is a character class with 3 elements, it will match only single character values that fit between 0-2, 5, and 5 again. The result is: 0, 1, 2, or 5.

[1-9][0-9]

Matches values between 10 and 99.

\b([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\b

Will in fact match the range between 0 and 255 by using alternation operators surrounded by round brackets to group the alternatives together due to their “lowest precedence” property.

Email Addresses

\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b

Matches most commonly used email addresses. It is NOT case sensitive and is delimited with word boundaries.

^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$

Checks for valid case insensitive email addresses without word boundaries.

Valid Dates

^(19|20)\d\d[- /.](0[1-9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])$

Matches date values in the yyyy-mm-dd format between 1900-01-01 and 2099-12-31.

JavaScript and jQuery

Topic: Data Type Conversion

The variable's **data type** is the JavaScript scripting engine's interpretation of the type of data that variable is currently holding. A string variable holds a string; a number variable holds a number value, and so on. However, unlike many other languages, in JavaScript, the same variable can hold different types of data, all within the same application.

This is a concept known by the terms loose typing and dynamic typing, both of which mean that a JavaScript variable can hold different data types at different times depending on context.

With a loosely typed language, you don't have to declare ahead of time that a variable will be a string or a number or a boolean, as the data type is actually determined while the application is being processed. If you start out with a string variable and then want to use it as a number, that's perfectly fine, as long as the string actually contains something that resembles a number and not something such as an email address. If you later want to treat it as a string again, that's fine, too.

Topic: Exception Handling

try…catch…finally Statement

Sets up blocks of code in which errors that are thrown in one block are handled in another. Errors that are thrown inside the try block are caught in the catch block.

```
try {
    tryStatements
}
catch(exception){
    catchStatements
}
finally {
    finallyStatements
}
```

tryStatements – Required. Statements where an error can occur.
exception – Required. Any variable name. The initial value of exception is the value of the thrown error.
catchStatements – Optional. Statements to handle errors occurring in the associated *tryStatements*.
finallyStatements – Optional. Statements that are unconditionally executed after all other error processing has occurred.

Topic: *this* object

In JavaScript, *this* normally refers to the object which 'owns' the method, but it depends on how a function is called.

If there's no current object, *this* refers to the global object. In a web browser, that's 'window' — the top-level object which represents the document, location, history and a few other useful properties and methods.

When calling an object constructor or any of its methods, *this* refers to the instance of the object.

jQuery DOM Insertion, Inside

.append() and .aapendTo()

The .append() method inserts the specified content as the last child of each element in the jQuery collection (To insert it as the *first* child, use .prepend()).

The .append() and .appendTo() methods perform the same task. The major difference is in the syntax-specifically, in the placement of the content and target. With .append(), the selector expression preceding the method is the container into which the content is inserted. With .appendTo(), on the other hand, the content precedes the method, either as a selector expression or as markup created on the fly, and it is inserted into the target container.

.html()

Get the HTML contents of the first element in the set of matched elements or set the HTML contents of every matched element (cannot be used with XML elements).

In an HTML document, .html() can be used to get the contents of any element. If the selector expression matches more than one element, only the first match will have its HTML content returned.

.prepend() and .prependTo()

The .prepend() method inserts the specified content as the first child of each element in the jQuery collection (To insert it as the last child, use .append()).

The .prepend() and .prependTo() methods perform the same task. The major difference is in the syntax—specifically, in the placement of the content and target. With .prepend(), the selector expression preceding the method is the container into which the content is inserted. With .prependTo(), on the other hand, the content precedes the method, either as a selector expression or as markup created on the fly, and it is inserted into the target container.

.text()

Get the combined text contents of each element in the set of matched elements, including their descendants, or set the text contents of the matched elements. (can be used in XML and HTML elements)

jQuery: DOM Insertion, Outside

.after()
Insert content, specified by the parameter, after each element in the set of matched elements.

.before()
Insert content, specified by the parameter, before each element in the set of matched elements.

.insertAfter()
Insert every element in the set of matched elements after the target.

.insertBefore()
Insert every element in the set of matched elements before the target.

jQuery DOM Insertion, Around

.unwrap()
Remove the parents of the set of matched elements from the DOM, leaving the matched elements in their place.

.wrap()
Wrap an HTML structure around each element in the set of matched elements.

.wrapAll()
Wrap an HTML structure around all elements in the set of matched elements.

.wrapInner()
Wrap an HTML structure around the content of each element in the set of matched elements.

jQuery General Attribute

These methods get and set DOM attributes of elements

.attr()

Get the value of an attribute for the first element in the set of matched elements or set one or more attributes for every matched element.

.prop()

Get the value of a property for the first element in the set of matched elements or set one or more properties for every matched element.

.removeAttr()

Remove an attribute from each element in the set of matched elements.

.removeProp()

Remove a property for the set of matched elements.

.val()

Get the current value of the first element in the set of matched elements or set the value of every matched element.

jQuery Helper Functions

These functions assist with common idioms encountered when performing AJAX tasks.

.param()

Create a serialized representation of an array or object, suitable for use in a URL query string or Ajax request.

.serialize()

Encode a set of form elements as a string for submission.

.serializeArray()

Encode a set of form elements as an array of names and values.

jQuery Events

.bind()
Attach a handler to an event for the elements.

.blur()
Bind an event handler to the “blur” JavaScript event, or trigger that event on an element.

.change()
Bind an event handler to the “change” JavaScript event, or trigger that event on an element.

.click()
Bind an event handler to the “click” JavaScript event, or trigger that event on an element.

.dblclick()
Bind an event handler to the “dblclick” JavaScript event, or trigger that event on an element.

.delegate()
Attach a handler to one or more events for all elements that match the selector, now or in the future, based on a specific set of root elements.

.die()
Remove event handlers previously attached using .live() from the elements.

.error()
Bind an event handler to the “error” JavaScript event.

event.currentTarget
The current DOM element within the event bubbling phase.

event.data
An optional object of data passed to an event method when the current executing handler is bound.

event.delegateTarget
The element where the currently-called jQuery event handler was attached.

event.isDefaultPrevented()
Returns whether event.preventDefault() was ever called on this event object.

event.isImmediatePropagationStopped()
Returns whether event.stopImmediatePropagation() was ever called on this event object.

event.isPropagationStopped()
Returns whether event.stopPropagation() was ever called on this event object.

event.metaKey
Indicates whether the META key was pressed when the event fired.

event.namespace
The namespace specified when the event was triggered.

event.pageX
The mouse position relative to the left edge of the document.

event.pageY
The mouse position relative to the top edge of the document.

event.preventDefault()
If this method is called, the default action of the event will not be triggered.

event.relatedTarget
The other DOM element involved in the event, if any.

event.result
The last value returned by an event handler that was triggered by this event, unless the value was undefined.

event.stopImmediatePropagation()
Keeps the rest of the handlers from being executed and prevents the event from bubbling up the DOM tree.

event.stopPropagation()
Prevents the event from bubbling up the DOM tree, preventing any parent handlers from being notified of the event.

event.target
The DOM element that initiated the event.

event.timeStamp
The difference in milliseconds between the time the browser created the event and January 1, 1970.

event.type
Describes the nature of the event.

event.which
For key or mouse events, this property indicates the specific key or button that was pressed.

.focus()
Bind an event handler to the “focus” JavaScript event, or trigger that event on an element.

.focusin()
Bind an event handler to the “focusin” event.

.focusout()
Bind an event handler to the “focusout” JavaScript event.

.hover()
Bind one or two handlers to the matched elements, to be executed when the mouse pointer enters and leaves the elements.

jQuery.proxy()
Takes a function and returns a new one that will always have a particular context.

.keydown()
Bind an event handler to the “keydown” JavaScript event, or trigger that event on an element.

.keypress()
Bind an event handler to the “keypress” JavaScript event, or trigger that event on an element.

.keyup()
Bind an event handler to the “keyup” JavaScript event, or trigger that event on an element.

.live()
Attach an event handler for all elements which match the current selector, now and in the future.

.load()
Bind an event handler to the “load” JavaScript event.

.mousedown()
Bind an event handler to the “mousedown” JavaScript event, or trigger that event on an element.

.mouseenter()
Bind an event handler to be fired when the mouse enters an element, or trigger that handler on an element.

.mouseleave()
Bind an event handler to be fired when the mouse leaves an element, or trigger that handler on an element.

.mousemove()
Bind an event handler to the “mousemove” JavaScript event, or trigger that event on an element.

.mouseout()
Bind an event handler to the “mouseout” JavaScript event, or trigger that event on an element.

.mouseover()
Bind an event handler to the “mouseover” JavaScript event, or trigger that event on an element.

.mouseup()
Bind an event handler to the “mouseup” JavaScript event, or trigger that event on an element.

.off()
Remove an event handler.

.on()
Attach an event handler function for one or more events to the selected elements.

.one()
Attach a handler to an event for the elements. The handler is executed at most once per element.

.ready()
Specify a function to execute when the DOM is fully loaded.

.resize()
Bind an event handler to the “resize” JavaScript event, or trigger that event on an element.

.scroll()
Bind an event handler to the “scroll” JavaScript event, or trigger that event on an element.

.select()
Bind an event handler to the “select” JavaScript event, or trigger that event on an element.

.submit()
Bind an event handler to the “submit” JavaScript event, or trigger that event on an element.

.toggle()
Bind two or more handlers to the matched elements, to be executed on alternate clicks.

.trigger()
Execute all handlers and behaviors attached to the matched elements for the given event type.

.triggerHandler()
Execute all handlers attached to an element for an event.

.unbind()
Remove a previously-attached event handler from the elements.

.undelegate()
Remove a handler from the event for all elements which match the current selector, based upon a specific set of root elements.

.unload()
Bind an event handler to the “unload” JavaScript event.

jQuery Ajax Methods

The jQuery library has a full suite of AJAX capabilities. The functions and methods therein allow us to load data from the server without a browser page refresh.

.ajaxComplete()

Register a handler to be called when Ajax requests complete. This is an AjaxEvent.

.ajaxError()

Register a handler to be called when Ajax requests complete with an error. This is an Ajax Event.

.ajaxSend()

Attach a function to be executed before an Ajax request is sent. This is an Ajax Event.

.ajaxStart()

Register a handler to be called when the first Ajax request begins. This is an Ajax Event.

.ajaxStop()

Register a handler to be called when all Ajax requests have completed. This is an Ajax Event.

.ajaxSuccess()

Attach a function to be executed whenever an Ajax request completes successfully. This is an Ajax Event.

.ajaxPrefilter()

Handle custom Ajax options or modify existing options before each request is sent and before they are processed by $.ajax().

.ajaxSetup()

Set default values for future Ajax requests.

.ajaxTransport()

Creates an object that handles the actual transmission of Ajax data.

.getJSON()

Load JSON-encoded data from the server using a GET HTTP request.

.getScript()

Load a JavaScript file from the server using a GET HTTP request, then execute it.

.ajax()

Perform an asynchronous HTTP (Ajax) request.

.get()

Load data from the server using a HTTP GET request.

.post()

Load data from the server using a HTTP POST request.

.load()

Load data from the server and place the returned HTML into the matched element.

jQuery Prototype Methods

The prototype property allows you to add properties and methods to an object.

Example:

```
<script>
 function employee(name,jobtitle,born) {
  this.name=name;
  this.jobtitle=jobtitle;
  this.born=born;
 }
 …
 var fred=new employee("Fred Flintstone","Caveman",1970);
 employee.prototype.salary=null;
 fred.salary=20000;

 document.write(fred.salary);
 …
</script>
```

.call()

The Function.prototype.call() method calls a function with a given this value and arguments provided individually.

Syntax: fun.call(thisArg[, arg1[, arg2[, ...]]])

You can assign a different *this* object when calling an existing function. *this* refers to the current object, the calling object. With .call(), you can write a method once and then inherit it in another object, without having to rewrite the method for the new object.

.call can be used to chain constructors for an object and call a method on behalf of another object.

.apply()

The Function.prototype.apply() method calls a function with a given this value and arguments provided as an array (or an array like object).

Syntax: fun.apply(thisArg[, argsArray])

You can assign a different *this* object when calling an existing function. *this* refers to the current object, the calling object. With apply, you can write a method once and then inherit it in another object, without having to rewrite the method for the new object.

.apply() is very similar to .call(), except for the type of arguments it supports. You can use an *arguments* array instead of a named set of parameters. With .apply(), you can use an array literal, for example, fun.apply(this, ['eat', 'bananas']), or an Array object, for example, fun.apply(this, new Array('eat', 'bananas')).

You can also use *arguments* for the *argsArray* parameter. *arguments* is a local variable of a function. It can be used for all unspecified arguments of the called object. Thus, you do not have to know the arguments of the called object when you use the .apply() method. You can use *arguments* to pass all the arguments to the called object. The called object is then responsible for handling the arguments.

jQuery Selectors

nth-child selector

Description: *Selects all elements that are the nth-child of their parent.*

index: The index of each child to match, starting with 1, the string even or odd, or an equation (eg. :nth-child(even), :nth-child(4n))

Because jQuery's implementation of :nth- selectors is strictly derived from the CSS specification, the value of n is "1-indexed", meaning that the counting starts at 1. For other selector expressions such as :eq() or :even jQuery follows JavaScript's "0-indexed" counting. Given a single <ul> containing two s, $('li:nth-child(1)') selects the first <li> while $('li:eq(1)') selects the second.

The :nth-child(n) pseudo-class is easily confused with :eq(n), even though the two can result in dramatically different matched elements. With :nth-child(n), all children are counted, regardless of what they are, and the specified element is selected only if it matches the selector attached to the pseudo-class. With :eq(n) only the selector attached to the pseudo-class is counted, not limited to children of any other element, and the (n+1)th one (n is 0-based) is selected.

Example

Find the second li in each matched ul.

```
<script>$("ul li:nth-child(2)").append("<span> - 2nd!</span>");
```

jQuery Node Methods

addEventListener("eventType", listenerFunction)

Binds an event handler function to the current node so that the function executes when an event of a particular type arrives at the node either as event target or during event propagation. The node listens for the event type either during event capture or event bubbling propagation, depending upon the setting of the Boolean third parameter. You may invoke this method multiple times for the same node but with different parameter values to assign as many event handling behaviors as you like, but only one listener function may be invoked for the same event and propagation type. If the event listener is added on a temporary basis, it may be removed via the removeEventListener( ) method.

removeEventListener ("eventType", listenerFunction)

Removes the event handler.

CSS3

@media

Sets the media types for a set of rules in a styleSheet object.

Examples

In the following example, the **@media** rule is used to specify the **font-size** attribute of the **body** element for two media types.

```
// For computer screens, the font size is 12pt.
@media screen {
   BODY {font-size:12pt;}
}

// When printed, the font size is 8pt.
@media print {
   BODY {font-size:8pt;}
}
```

The following declaration is a typical media query. In this case, `screen` indicates the target media type, and `max-width` is the target media property. The declaration states that the specified rules (no border on **div** elements) are only to be applied when the page is displayed on a screen in a browser window with a width of at most 400 pixels.

```
@media screen and (max-width:400px) {
   div {border:none;}
}
```

You can use media properties together to create even more specific queries, such as the following. This declaration applies the specified rules when the medium is a screen and the browser window has a width of no more than 400 pixels *and* a height of no more than 600 pixels.

```
@media screen and (max-width:400px) and (max-height:600px) {
   ...
}
```

transition property

The *transition* property specifies one or more sets of space-delimited transition properties for a set of corresponding object properties. The transition property values must be set in the following order:

- *transition-property*
- *transition-duration*
- *transition-timing-function*
- *transition-delay*

If you have more than one set of the four transition property values, you must separate each set using a comma.

Example

If the following style was applied to a element,

```
transition: opacity 5s linear 1s, background-color 2s ease;
```

and subsequently if a <div:hover> specified a new background color value,

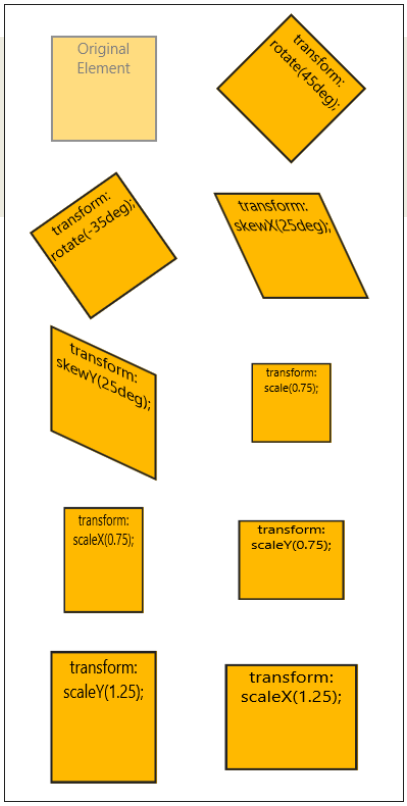
the transition property will cause the color to change on hover using a two seconds and the ease timing function. There is no delay value, so the transition begins immediately.

2-D Transform functions

Function	Description
matrix(*a*, *b*, *c*, *d*, *e*, *f*)	Specifies a 2-D transformation in the form of a transformation matrix of six values.
rotate(*angle*)	Specifies a 2-D rotation by the angle specified in the parameter about the origin of the element.
scale(*sx*,*sy*)	Specifies a 2-D scale operation by the [*sx*,*sy*] scaling vector that is described by the two parameters.
scaleX(*sx*)	Specifies a scale operation by using the [*sx*,1] scaling vector, where *sx* is given as the parameter.
scaleY(*sy*)	Specifies a scale operation by using the [1,*sy*] scaling vector, where *sy* is given as the parameter.
skew(*angleX*,*angleY*)	Specifies a skew transformation along the *x*- and *y*-axes. The first angle parameter specifies the skew on the *x*-axis. The second angle parameter specifies the skew on the *y*-axis.
skewX(*angle*)	Specifies a skew transformation along the *x*-axis by the given angle.
skewY(*angle*)	Specifies a skew transformation along the *y*-axis by the given angle.
translate(*tx*,*ty*)	Specifies a 2-D translation by the vector [*tx*,*ty*], where *tx* is the first translation-value parameter and *ty* is the optional second translation-value parameter.
translateX(*tx*)	Specifies a translation by the given amount in the *x* direction.
translateY(*ty*)	Specifies a translation by the given amount in the *y* direction.

The following declarations ensure support in Windows Internet Explorer 9 ("-ms-"), Chrome and Safari ("-webkit-"), Firefox ("-moz-"), Opera ("-o-"), and browsers that don't require a prefix, such as Internet Explorer 10:

```
-ms-transform: translateX(400px);
-webkit-transform: translateX(400px);
-moz-transform: translateX(400px);
-o-transform: translateX(400px);
transform: translateX(400px);
```

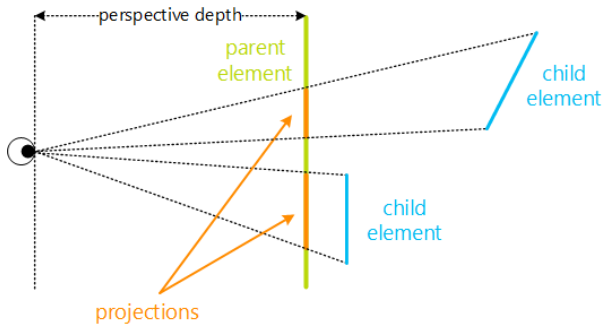


3-D Transform functions

3-D transforms are applied in the same way as 2-D transforms (by adding a transform property to the element's style). The available transform functions that support 3-D are:

Function	Description
rotate3d(*x*, *y*, *z*, *angle*)	Specifies a clockwise 3-D rotation.
rotateX(*angle*)	Specifies a clockwise rotation by the given angle about the *x*-axis.
rotateY(*angle*)	Specifies a clockwise rotation by the given angle about the *y*-axis.
rotateZ(*angle*)	Specifies a clockwise rotation by the given angle about the *z*-axis.
scale3d(*sx*,*sy*,*sz*)	Specifies a 3-D scale operation by the [*sx*,*sy*,*sz*] scaling vector described by the three parameters.
scaleZ(*sz*)	Specifies a scale operation using the [1,1,*sz*] scaling vector, where *sz* is given as the parameter.
translate3d(*tx*, *tytz*)	Specifies a 3-D translation by the vector [*tx*,*ty*,*tz*], where *tx*, *ty*, and *tz* are the first, second, and third translation-value parameters respectively.
translateZ(*tz*)	Specifies a translation by a given amount in the *z*-direction.
matrix3d(*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*, *i*, *j*, *k*, *l*, *m*, *n*, *o*, *p*)	Specifies a 3-D transformation as a 4×4 homogeneous matrix of sixteen values in column-major order.

Adding perspective to 3-D transforms The perspective property adds the illusion of depth to CSS transforms. It specifies a length value that represents the perspective from which all child elements of the object are viewed.



CSS Keyframes Animations

Cascading Style Sheets (CSS) animations enable you to do more than just smoothly change CSS properties over time (CSS transitions already do this). They also offer you the ability to design complex animations using keyframes, as well as more fine-grained control via scripting.

@keyframe rule

It allows you to specify the values a CSS property needs to have at different points during the animation. For example:

```
@keyframes fadeOut {
  from {
    opacity: 1;
  }
  to {
    opacity: 0;
  }
}

.TransformDemoDivFadeOut:hover {
  animation-duration: 2s;
  animation-name: fadeOut;
}
```

HTML5

HTML5 Input Types

HTML5 has several new input types for forms. These new features allow better input control and validation.

Value	Description
button	Defines a clickable button (mostly used with a JavaScript to activate a script)
checkbox	Defines a checkbox
color New	Defines a color picker
date New	Defines a date control (year, month and day (no time))
datetime New	Defines a date and time control (year, month, day, hour, minute, second, and fraction of a second, based on UTC time zone)
datetime-local New	Defines a date and time control (year, month, day, hour, minute, second, and fraction of a second (no time zone)
email New	Defines a field for an e-mail address
file hidden	Defines a file-select field and a "Browse..." button (for file uploads)
image	Defines a hidden input field
month	Defines an image as the submit button
number New	Defines a month and year control (no time zone)
password	Defines a field for entering a number
radio	Defines a password field (characters are masked)
range	Defines a radio button
	Defines a control for entering a number whose exact value is not important (like a slider control)
reset	Defines a reset button (resets all form values to default values)
search	Defines a text field for entering a search string
submit	Defines a submit button
tel	Defines a field for entering a telephone number
text	Default. Defines a single-line text field (default width is 20 characters)
time	Defines a control for entering a time (no time zone)
url ew	Defines a field for entering a URL

week New	Defines a week and year control (no time zone)

required: HTML <input> required Attribute

The required attribute is a boolean attribute.

When present, it specifies that an input field must be filled out before submitting the form.

Note: The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

HTML Tags

HTML <input> tag

The <input> tag specifies an input field where the user can enter data. <input> elements are used within a <form> element to declare input controls that allow users to input data. An input field can vary in many ways, depending on the type attribute.

Differences Between HTML 4.01 and HTML5

- In HTML 4.01, the "align" attribute is deprecated, and it is not supported in HTML5. Use CSS to align <input> elements.
- In HTML5, the <input> tag has several new attributes, and the type attribute has several new values.

HTML <nav> Tag

New in HTML5. The <nav> tag defines a section of navigation links. Not all links of a document must be in a <nav> element. The <nav> element is intended only for major block of navigation links.

Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content.

The <nav> tag is **new** in HTML5.

HTML <figure> Tag

New in HTML5. The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

While the content of the <figure> element is related to the main flow, its position is independent of the main flow, and if removed it should not affect the flow of the document.

Example:

```
<figure>
  <figcaption>FirebrandTraining</figcaption>
  <img src="firebrandtraining.jpg" alt="Firebrand Training Logo">
</figure>
```

HTML5 Web Storage

With HTML5, web pages can store data locally within the user's browser.

Earlier, this was done with cookies. However, Web Storage is more secure and faster. The data is not included with every server request, but used ONLY when asked for. It is also possible to store large amounts of data, without affecting the website's performance.

The data is stored in key/value pairs, and a web page can only access data stored by itself.

HTMLStorage Object

Represents the list of key/value pairs that have been assigned to a single storage area.

clear() Method

Removes all key/value pairs from the Web Storage area.

getItem() Method

Retrieves the current value associated with the Web Storage key.

initStorageEvent() Method

Initializes a new Document Object Model (DOM) storage event that the createEvent method created.

Key() Method

Retrieves the key at the specified index in the collection.

removeItem() and setItem Methods

Delete and set a key/value pair from the Web Storage collection respectively.

key Property

Gets the key that is updated.

length Property

Retrieves the length of the key/value list.

localStorage Property

Retrieves the Web Storage area specific to the current document.

newValue Property

Gets the new value of the key.

oldValue Property

Gets the previous value of the key.

remainingSpace Property

Retrieves the remaining memory space, in bytes, for the storage object.

sessionStorage Property

Retrieves the Web Storage area for the session.

storageArea Property

Gets the Storage object of the affected document.

url Property

Gets the address of the document that the update affects.

Canvas

The HTML5 <canvas> element is used to draw graphics using JavaScript. The element is only a container for graphics. The graphics are actually drawn via the script.

IE8 and previous versions of IE do not support the <canvas> element. IE9 and other major web browsers support it.

Syntax

```
<canvas id="myCanvas"
        width="200"
        height="100"
        style="border:1px solid #000000;>

</canvas>
```

To draw the graphic on the canvas, use a JavaScript tag:

```
<script>
  var c=document.getElementById("myCanvas");
  var ctx=c.getContext("2d");
  ctx.fillStyle="#FF8866";
  ctx.fillRect(24,11,150,75);
</script>
```

Canvas Paths

To draw shapes on a canvas you can use various methods for different types of paths

To draw a straight line, use the following methods:

moveTo(x,y) – defines the starting point of the line

lineTo(x,y) – defines the ending point of the line

To draw a circle, use the following method:

arc(x,y,r,start,stop

To draw Text, use the following methods: font –

defines the font properties for text fillText(text,x,y) –

Draws "filled" text on the canvas strokeText(text,x,y) –

Draws text on the canvas (no fill)

Canvas Gradients

Gradients can be used to fill rectangles, circles, lines, text, etc. with gradient colors.

There are 2 types of gradients: createLinearGradient(x,y,x1,y1) –

Creates a linear gradient createRadialGradient(x,y,r,x1,y1,r1) – Creates

a radial/circular gradient **Gradients require 2 or more color stops:**

addColorStop() – method specifies the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1.

To use the gradient set either of the following 2 methods:

fillStyle or strokeStyle

Example (linear gradient)

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");

// Create gradient
var grd=ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle=grd;
ctx.fillRect(10,10,150,80);
```

Example (radial gradient)

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");

// Create gradient
var grd=ctx.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");

// Fill with gradient
ctx.fillStyle=grd;
ctx.fillRect(10,10,150,80);
```

Canvas Images

To draw an image on a canvas, we will use the following method:

drawImage(image,x,y)

SVG

SVG is a W3C standard and stands for Scalable Vector Graphics. It sis used to define vector-based graphics for the web in xml format. With SVG, graphics do not loose quality if they are zoomed or resized. Every element and every attribute in SVG files can be animated.

SVG can be embedded directly into HTML pages.

The following html code segment,

```
<html>
<body>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="190">
    <polygon
        points="100,10 40,180 190,60 10,60 160,180"
        style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;">
  </svg>
</body>
</html>
```

will generate the following graphic:

Differences between SVG and Canvas

Canvas	SVG
• Resolution dependent • No support for event handlers • Poor text rendering capabilities • You can save the resulting image as .png or .jpg • Well suited for graphic-intensive games	• Resolution independent • Support for event handlers • Best suited for applications with large rendering areas (Google Maps) • Slow rendering if complex (anything that uses the DOM a lot will be slow) • Not suited for game applications

The WebSocket API

References: W3C

The API enables Web applications to maintain bidirectional communications with server-side processes.

Dependencies

HTML and WebIDL (Web Interface Definition Language)

The WebSocket Interface

```
IDL[Constructor(DOMString url, optional (DOMString or DOMString[]) protocols)]
interface WebSocket : EventTarget {
  readonly attribute DOMString url;

  // ready state
  const unsigned short CONNECTING = 0;
  const unsigned short OPEN = 1;
  const unsigned short CLOSING = 2;
  const unsigned short CLOSED = 3;
  readonly attribute unsigned short readyState;
  readonly attribute unsigned long bufferedAmount;

  // networking
           attribute EventHandler onopen;
           attribute EventHandler onerror;
           attribute EventHandler onclose;
  readonly attribute DOMString extensions;
  readonly attribute DOMString protocol;
  void close([Clamp] optional unsigned short code, optional DOMString reason);

  // messaging
           attribute EventHandler onmessage;
           attribute DOMString binaryType;
  void send(DOMString data);
  void send(Blob data);
  void send(ArrayBuffer data);
  void send(ArrayBufferView data);
};
```

url: The first argument, *url*, specifies the URL to which to connect.

protocols: The second, *protocols*, optional array of strings. Each string in the array is a subprotocol name. The connection will only be established if the server reports that it has selected one of these subprotocols. The subprotocol names must all be strings that match the requirements for elements that comprise the value of Sec-WebSocket-Protocol header fields as defined by the WebSocket protocol specification.

When the WebSocket() constructor is invoked, the user agent must run these steps:

1. Parse a WebSocket URL's components from the url argument, to obtain *host*, *port*, *resource name*, and *secure*. If this fails, throw a SyntaxError exception and abort these steps.
2. If *secure* is false but the origin of the entry script has a scheme component that is itself a secure protocol, then throw a SecurityError exception.
3. If *port* is a port to which the user agent is configured to block access, then throw a SecurityError exception.
4. If *protocols* is absent, let *protocols* be an empty array. Otherwise, if *protocols* is present and a string, let *protocols* instead be an array consisting of just that string.
5. If any of the values in *protocols* occur more than once or otherwise fail to match the requirements for elements that comprise the value of Sec-WebSocket-Protocol header fields as defined by the WebSocket protocol specification, then throw a SyntaxError exception and abort these steps.
6. Let *origin* be the ASCII serialization of the origin of the entry script, converted to ASCII lowercase.
7. Return a new WebSocket object, and continue these steps in the background (without blocking scripts).
8. Establish a WebSocket connection given the set (*host*, *port*, *resource* name, *secure*), along with the *protocols* list, an empty list for the extensions, and *origin*. The headers to send appropriate cookies must be a Cookie header whose value is the cookie-string computed from the user's cookie store and the URL *url*; for these purposes this is not a "non-HTTP" API.
   When the user agent validates the server's response during the "establish a WebSocket connection" algorithm, if the status code received from the server is not 101 (e.g. it is a redirect), the user agent must fail the WebSocket connection.

The ***readyState*** attribute represents the state of the connection. It can have the following values:

- *CONNECTING* (numeric value 0)
  The connection has not yet been established.
- *OPEN* (numeric value 1)
  The WebSocket connection is established and communication is possible.
- *CLOSING* (numeric value 2)
  The connection is going through the closing handshake, or the close() method has been invoked.
- *CLOSED* (numeric value 3)
  The connection has been closed or could not be opened.

When the object is created its *readyState* must be set to CONNECTING (0).

The ***extensions*** attribute must initially return the empty string. After the WebSocket connection is established, its value might change, as defined below.

The ***protocol*** attribute must initially return the empty string. After the WebSocket connection is established, its value might change, as defined below.

The ***close()*** method must run the following steps:

1. If the method's first argument is present but is not an integer equal to 1000 or in the range 3000 to 4999, throw an InvalidAccessError exception and abort these steps.
2. If the method's second argument is present, then run these substeps:
    1. Let raw reason be the method's second argument.
    2. Let Unicode reason be the result of converting raw reason to a sequence of Unicode characters.
    3. Let reason be the result of encoding Unicode reason as UTF-8.
    4. If reason is longer than 123 bytes, then throw a SyntaxError exception and abort these steps. [RFC3629]
3. Run the first matching steps from the following list:
    a. If the *readyState* attribute is in the CLOSING (2) or CLOSED (3) state:
    Do nothing.
    b. If the WebSocket connection is not yet established:
    Fail the WebSocket connection and set the readyState attribute's value to CLOSING (2).
    c. If the WebSocket closing handshake has not yet been started:
    Start the WebSocket closing handshake and set the readyState attribute's value to CLOSING (2).
    If the first argument is present, then the status code to use in the WebSocket Close message must be the integer given by the first argument.
    If the second argument is also present, then reason must be provided in the Close message after the status code.
    d. Otherwise:
    Set the *readyState* attribute's value to CLOSING (2).

The ***bufferedAmount*** attribute must return the number of bytes of application data (UTF-8 text and binary data) that have been queued using ***send()*** but that, as of the last time the event loop started executing a task, had not yet been transmitted to the network.

The ***send(data)*** method transmits data using the connection. If the *readyState* attribute is CONNECTING, it must throw an *InvalidStateError* exception. Otherwise, the user agent must run the appropriate set of steps from the following list:

- **If the argument is a string**

Let data be the result of converting the data argument to a sequence of Unicode characters. If the WebSocket connection is established and the WebSocket closing handshake has not yet started, then the user agent must send a WebSocket Message comprised of data using a text frame opcode; if the data cannot be sent, e.g. because it would need to be buffered but the buffer is full, the user agent must close the WebSocket connection with prejudice. Any invocation of this method with a string argument that does not throw an exception must increase the bufferedAmount attribute by the number of bytes needed to express the argument as UTF-8.

- **If the argument is a Blob object**

If the WebSocket connection is established, and the WebSocket closing handshake has not yet started, then the user agent must send a WebSocket Message comprised of data using a binary frame opcode; if the data cannot be sent, e.g. because it would need to be buffered but the buffer is full, the user agent must close the WebSocket connection with prejudice. The data to be sent is the raw data represented by the Blob object. Any invocation of this method with a Blob argument that does not throw an exception must increase the bufferedAmount attribute by the size of the Blob object's raw data, in bytes.

- **If the argument is an ArrayBuffer object**

If the WebSocket connection is established, and the WebSocket closing handshake has not yet started, then the user agent must send a WebSocket Message comprised of data using a binary frame opcode; if the data cannot be sent, e.g. because it would need to be buffered but the buffer is full, the user agent must close the WebSocket connection with prejudice. The data to be sent is the data stored in the buffer described by the ArrayBuffer object. Any invocation of this method with an ArrayBuffer argument that does not throw an exception must increase the bufferedAmount attribute by the length of the ArrayBuffer in bytes.

- **If the argument is an ArrayBufferView object**

If the WebSocket connection is established, and the WebSocket closing handshake has not yet started, then the user agent must send a WebSocket Message comprised of data using a binary frame opcode; if the data cannot be sent, e.g. because it would need to be buffered but the buffer is full, the user agent must close the WebSocket connection with prejudice. The data to be sent is the data stored in the section of the buffer described by the ArrayBuffer object that the ArrayBufferView object references. Any invocation of this method with an ArrayBufferView argument that does not throw an exception must increase the bufferedAmount attribute by the length of the ArrayBufferView in bytes.

Feedback from the Protocol

When the WebSocket connection is established, the user agent must queue a task to run these steps:

1. Change the *readyState* attribute's value to OPEN (1).
2. Change the extensions attribute's value to the extensions in use, if is not the null value.
3. Change the protocol attribute's value to the subprotocol in use, if is not the null value.
4. Act as if the user agent had received a set-cookie-string consisting of the cookies set during the server's opening handshake, for the URL url given to the WebSocket() constructor.
5. Fire a simple event named open at the WebSocket object.

When a WebSocket message has been received with type type and data data, the user agent must queue a task to follow these steps:

1. If the *readyState* attribute's value is not OPEN (1), then abort these steps.
2. Let event be an event that uses the MessageEvent interface, with the event type message, which does not bubble, is not cancelable, and has no default action.

3. Initialize event's origin attribute to the Unicode serialization of the origin of the URL that was passed to the WebSocket object's constructor.
4. If type indicates that the data is Text, then initialize event's data attribute to data.
   If type indicates that the data is Binary, and binaryType is set to "blob", then initialize event's data attribute to a new Blob object that represents data as its raw data.
   If type indicates that the data is Binary, and binaryType is set to "arraybuffer", then initialize event's data attribute to a new read-only ArrayBuffer object whose contents are data.
5. Dispatch event at the WebSocket object.

Parsing WebSocket URLs

The steps to parse a WebSocket URL's components from a string *url* are as follows. These steps return either a *host*, a *port*, a *resource* name, and a *secure* flag, or they fail.

1. If the *url* string is not an absolute URL, then fail this algorithm.
2. Resolve the *url* string, with the URL character encoding set to UTF-8.
3. If *url* does not have a <scheme> component whose value, when converted to ASCII lowercase, is either "ws" or "wss", then fail this algorithm.
4. If *url* has a <fragment> component, then fail this algorithm.
5. If the <scheme> component of *url* is "ws", set secure to false; otherwise, the <scheme> component is "wss", set *secure* to true.
6. Let host be the value of the <host> component of *url*, converted to ASCII lowercase.
7. If *url* has a <port> component, then let *port* be that component's value; otherwise, there is no explicit port.
8. If there is no explicit *port*, then: if secure is false, let port be 80, otherwise let port be 443.
9. Let *resource* name be the value of the <path> component (which might be empty) of *url*.
10. If *resource* name is the empty string, set it to a single character U+002F SOLIDUS (/).
11. If *url* has a <query> component, then append a single U+003F QUESTION MARK character (?) to *resource* name, followed by the value of the <query> component.
12. Return *host*, *port*, *resource* name, and *secure*.

Event definitions

```
[Constructor(DOMString type, optional CloseEventInit eventInitDict)]
interface CloseEvent : Event {
  readonly attribute boolean wasClean;
  readonly attribute unsigned short code;
  readonly attribute DOMString reason;
};

dictionary CloseEventInit : EventInit {
  boolean wasClean;
  unsigned short code;
  DOMString reason;
};
```

The ***wasClean*** attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to false. It represents whether the connection closed cleanly or not.

The ***code*** attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to zero. It represents the WebSocket connection close code provided by the server.

The ***reason*** attribute must return the value it was initialized to. When the object is created, this attribute must be initialized to empty string. It represents the WebSocket connection close reason provided by the server.

WebWorker

The Web Workers API defines a way to run scripts in the background. Traditionally, browsers have been single-threaded, forcing all the script in your application to run together in a single UI thread. Although you can create the illusion of several things happening at the same time by using DOM events and the setTimeout API, it takes only one computationally intensive task to bring the user experience to a screeching halt.

The Web Worker API provides a way for web application authors to spawn background scripts that run in parallel with the main page. You can spawn several threads at a time to use for long-running tasks. A new worker object requires a .js file, which is included via an asynchronous request to the server.

```
var myWorker = new Worker('worker.js');
```

All communication to and from the worker thread is managed through messages. Both the host worker and the worker script can send messages by using postMessage and listen for a response by using the onmessage event. The content of the message is sent as the data property of the event.

The following example creates a worker thread and listens for a message.

```
var hello = new Worker('hello.js');
hello.onmessage = function(e) {
  alert(e.data);
};
```

The worker thread sends the message to be displayed.

```
postMessage('Hello world!');
```

Two-way communication with Web Workers

To set up two-way communication, both the main page and the worker thread listen for the onmessage event. In the following example, the worker thread returns the message after a specified delay.

First, the script creates the worker thread.

```
var echo = new Worker('echo.js');
echo.onmessage = function(e) {
  alert(e.data);
}
```

The message text and timeout values are specified in a form. When the user clicks the submit button, the script passes two pieces of information to the worker in a JavaScript object literal. To prevent the page from submitting the form values in a new HTTP request, the event handler also calls preventDefault on the event object. Note that you cannot send references to DOM objects to a worker thread. Web Workers are limited in what data they can access. Only JavaScript primitives such as Object or String values are allowed.

```
<script>
window.onload = function() {
  var echoForm = document.getElementById('echoForm');
  echoForm.addEventListener('submit', function(e) {
    echo.postMessage({
      message : e.target.message.value,
      timeout : e.target.timeout.value
    });
    e.preventDefault();
  }, false);
}
</script>
<form id="echoForm">
  <p>Echo the following message after a delay.</p>
  <input type="text" name="message" value="Input message here."/><br/>
  <input type="number" name="timeout" max="10" value="2"/> seconds.<br/>
  <button type="submit">Send Message</button>
</form>
```

Finally, the worker listens for the message and returns it after the specified timeout interval.

```
onmessage = function(e)
{
  setTimeout(function()
  {
    postMessage(e.data.message);
  },
  e.data.timeout * 1000);
}
```

Web Worker API

In Internet Explorer 10 and Windows Store apps using JavaScript, the Web Workers API supports the following methods, properties, and events:

Method	Description
void**close();**	Terminates the worker thread.
void**importScripts(**inDOMString**... *urls*);**	Imports a comma-separated list of additional JavaScript files.
void**postMessage(in any data);**	Sends a message to or from the worker thread.

Attribute	Type	Description
location	WorkerLocation	Represents an absolute URL, including **protocol**, **host**, **port**, **hostname**, **pathname**, **search**, and **hash** components.
navigator	WorkerNavigator	Represents the identity and **onLine** state of the user agent client.
self	WorkerGlobalScope	The worker scope, which includes the WorkerLocation and WorkerNavigator objects.

Event	Description
onerror	A runtime error occurred.
onmessage	Message data received.

B.1

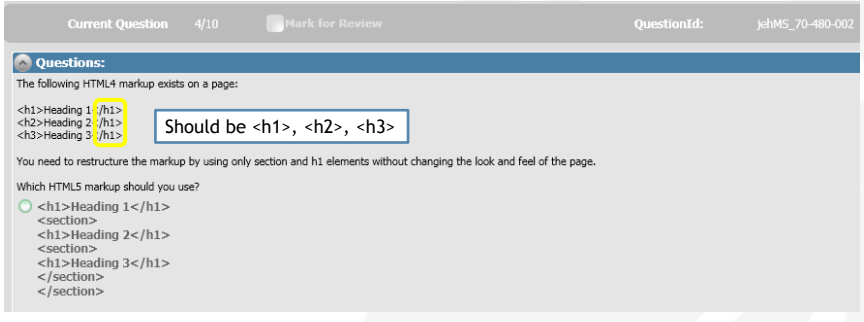
Appendix B
MeasureUp Errata

Programming in HTML5 with JavaScript and CSS3

Updated 15th August 2015

B.2

MeasureUp Errata
Question 70-480-002

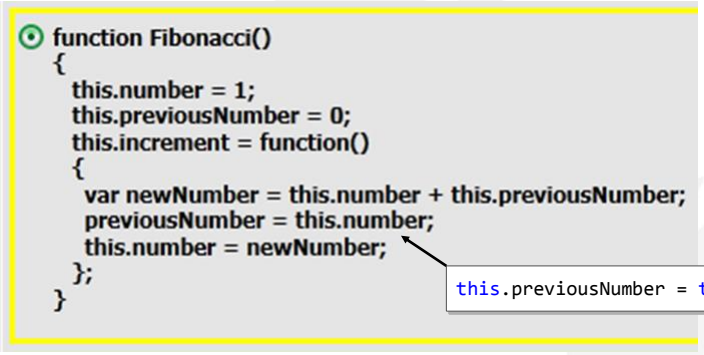


B.3

MeasureUp Errata

Question 70-480-015

- They say this is the correct answer but it is wrong because of a missing **this**



- See MUP_70-480-15.html for an example

B.4

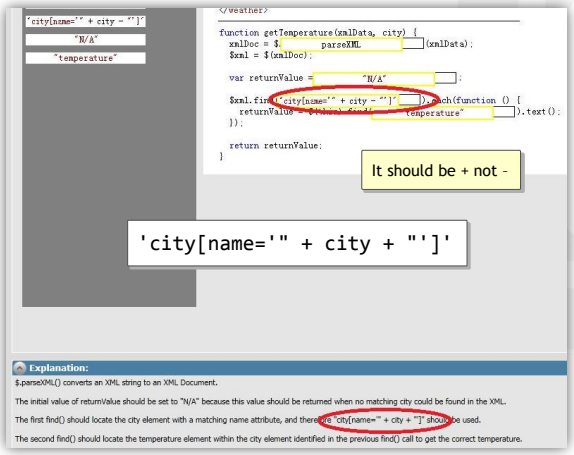
MeasureUp Errata

Question 70-480-035 and 70-480-046

- Question 70-480-035
  - A question about animation and it marks your answer as wrong when it is actually correct!
- Question 70-480-046
  - You must select two answers. The question says that each part is a complete answer—they are wrong! You need both parts to make a complete answer.

MeasureUp Errata
Unknown Question

B.5



MeasureUp Errata
Unknown Question

B.6

A web page contains the following markup:

```
<table>
<tr>
<td>Name:</td>
<td><input id="name" type="text"/></td>
</tr>
<tr>
<td>Age</td>
<td><input id="age" type="number"/></td>
</tr>
<tr>
<td>Salary</td>
<td><input id="salary" type="number"/></td>
</tr>
</table>
```

When the mouse pointer enters an input element, the background color of the row must turn green.

You need to write code to accomplish your goal.

Which jQuery code segment should you use?

- $(this).focus(function() { $("table td input").parent().parent().css("background-color", "green"); });
- $("td > input").focus(function() { $("input").parent().parent().css("background-color", "green"); });
- $("td").focus(function() { $("table td input").parent().parent().css("background-color", "green"); });
- $("table td input").focus(function() { $(this).parent().parent().css("background-color", "green"); });

All the answers use the **focus** event when they should use the **mouseover** or **mouseenter** events

Appendix C Frameworks

Programming in HTML5 with JavaScript and CSS3

Exam Topic: none

Updated 15th August 2015

Contents

- There are many client-side frameworks you can use to improve productive and maintenance of your HTML5 client-side applications
  - Bootstrap 3: layout and theming
  - Angular, Knockout, and Aurelia: MVVM frameworks
  - D3: bring data to life using HTML, SVG, and CSS

C.3

Bootstrap 3
What Is It?

- Visual Studio 2013 project templates use Bootstrap 3, a layout and theming framework created by Twitter
  - Bootstrap uses CSS3 to provide responsive design, which means layouts can dynamically adapt to different browser window sizes
- You can also use Bootstrap's theming feature to easily effect a change in the application's look and feel
  - In your browser, go to http://Bootswatch.com, choose a theme, and then click Download

Bootstrap
http://getbootstrap.com/

Free themes for Bootstrap
http://bootswatch.com

C.4

Bootstrap 3
Start With The Basic Template

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first; other content must come *after* -->
    <title>Bootstrap 101 Template</title>
    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <h1>Hello, Bootstrap!</h1>
    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

Basic template
http://getbootstrap.com/getting-started/#template

Bootstrap Grid System
Container and Rows

C.5

- The grid system is enabled by creating an element with a class of container and at least one row element
  - This will create a 1140/940/720 pixel centred container
  - Divide this into rows using elements with the class of row

```
<div class="container">
    <div class="row">
```

- Bootstrap logically divides each row into 12 equally spaced columns
  - You can make an element pan multiple columns by applying class of col-xs-*nn* where *nn* is a number between 1 and 12
  - For example, content that spans all 12 columns, or two columns

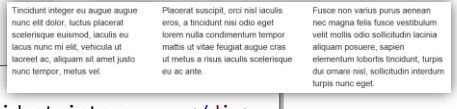
```
<div class="col-xs-12">
    <h1>Hello Bootstrap</h1>
```

```
<div class="col-xs-6"><h1>Hello</h1></div>
<div class="col-xs-6"><h1>Bootstrap</h1></div>
```

Bootstrap Grid System
Column Spanning

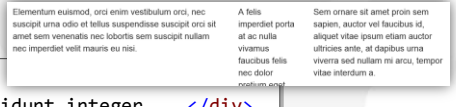
C.6

- A row with three columns that span equal widths
  - 4 + 4 + 4 = 12 in total




```
<div class="row">
    <div class="col-xs-4">Tincidunt integer ...</div>
    <div class="col-xs-4">Placerat suscipit, ...</div>
    <div class="col-xs-4">Fusce non varius ...</div>
```

- A row with three columns that span varying widths
  - 6 + 2 + 4 = 12 in total

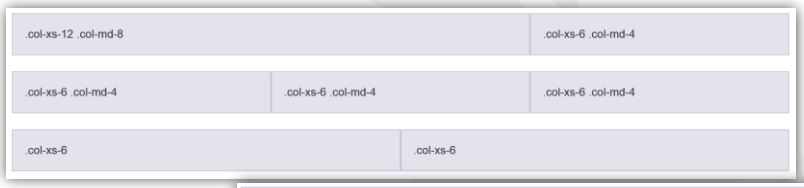



```
<div class="row">
    <div class="col-xs-6">Tincidunt integer ...</div>
    <div class="col-xs-2">Placerat suscipit, ...</div>
    <div class="col-xs-4">Fusce non varius ...</div>
```

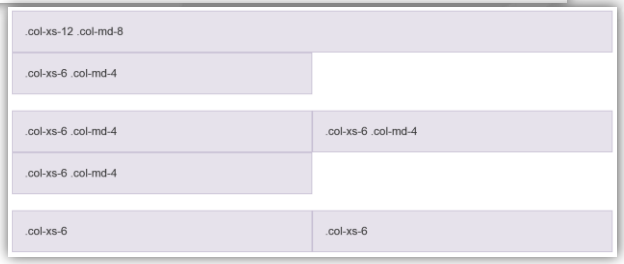
C.7

Bootstrap Grid System
Tiers of Classes

- The grid system has four tiers of classes: xs (phones), sm (tablets), md (desktops), and lg (larger desktops)
  - Use nearly any combination to create more flexible layouts



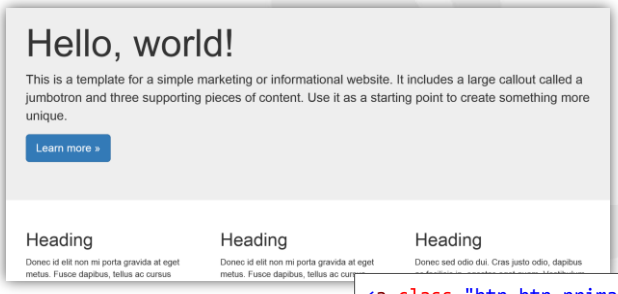
If the view port is wide enough the **md** values are used, otherwise the **xs** values are used



C.8

Bootstrap Jumbotron
What Is It?

- A large callout called a jumbotron and three supporting pieces of content




```
<a class="btn btn-primary btn-lg" href="#"
   role="button">Learn more &raquo;</a></p>
```

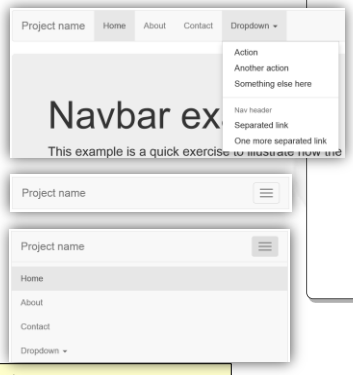
```
<div class="jumbotron">
    <div class="container">
        <h1>Hello, world!</h1>
        <p>This is a template for a simple marketing or ...</p>
```

Jumbotron template
http://getbootstrap.com/examples/jumbotron/

C.9

Bootstrap Navbar
Responsive Navigation Bar

- On desktop
- On mobile (collapsed)
- On mobile (expanded)




```
<nav class="navbar navbar-default">
    <div class="container-fluid">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed"
                    data-toggle="collapse" data-target="#navbar"
                    aria-expanded="false" aria-controls="navbar">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">Project name</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
              <li class="active"><a href="#">Home</a></li>
              <li><a href="#">About</a></li>
              <li><a href="#">Contact</a></li>
              <li class="dropdown">
                <a href="#" class="dropdown-toggle"
                   data-toggle="dropdown" role="button"
                   aria-expanded="false">
                    Dropdown <span class="caret"></span></a>
                <ul class="dropdown-menu" role="menu">
                  <li><a href="#">Action</a></li>
                  <li class="divider"></li>
                  <li class="dropdown-header">Nav header</li>
                  <li><a href="#">Separated link</a></li>
```

Navbar template
http://getbootstrap.com/examples/navbar/

C.10

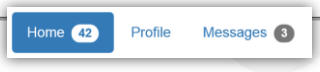
Bootstrap Themes
What Are They?

- Styles for buttons, links, labels, badges

btn-lg: large
n/a: normal
btn-sm: small
btn-xs: extra small

```
<p>
    <button type="button" class="btn btn-lg btn-default">Default</button>
    <button type="button" class="btn btn-lg btn-primary">Primary</button>
    <button type="button" class="btn btn-lg btn-success">Success</button>
    <button type="button" class="btn btn-lg btn-info">Info</button>
    <button type="button" class="btn btn-lg btn-warning">Warning</button>
    <button type="button" class="btn btn-lg btn-danger">Danger</button>
    <button type="button" class="btn btn-lg btn-link">Link</button>
</p>
```

```
<ul class="nav nav-pills" role="tablist">
    <li role="presentation" class="active"><a href="#">Home <span class="badge">42</span></a></li>
    <li role="presentation"><a href="#">Profile</a></li>
    <li role="presentation"><a href="#">Messages <span class="badge">3</span></a></li>
</ul>
```



Theme template
http://getbootstrap.com/examples/theme/

C.11

AngularJS
What Is It?

- Software engineering concepts like testable code, separation of concerns, MVC (Model-View-Controller) (or rather, MVVM), and so on applied to JavaScript
- Model-View-Controller
  - Model is the data behind the application, usually fetched from the server
  - View is the UI that the user interacts with
  - Controller is business logic and presentation layer
- AngularJS is declarative and easy to test

AngularJS — Superheroic JavaScript MVW Framework
https://angularjs.org/

C.12

AngularJS
How to Enable It

- Install the NuGet package for AngularJS Core or download from the official site
- Reference the script library and enable it using the **ng-app** directive on a root element

```
<body ng-app>
    <h1>Hello {{1+2}}</h1>
...
    <script src="Scripts/angular.js"></script>
</body>
```

- To enable better IntelliSense in Visual Studio 2013, download the file in the link below into
  - C:\Program Files (x86)\Microsoft Visual Studio 12.0\JavaScript\References

AngularJS IntelliSense File
https://raw.githubusercontent.com/jmbledsoe/angularjs-visualstudio-intellisense/master/src/Scripts/angular.intellisense.js

AngularJS
Data Binding

- Data binding in Angular looks something like this

```
Hello <span>{{name}}</span>
```

```
<input ng-model="name" />
```

- Now all we have to do is set the value of the name in JavaScript and Angular deals with the one-way binding
- Angular also supports two-way binding
  - Note: the ng- attributes are known as Angular *directives*

```
<form name="myForm" ng-submit="ctrl.submitData()">
    <input type="text" ng-model="user.name" />
    <input type="text" ng-model="user.email" />
</form>
```

- Angular directives extend the vocabulary of HTML to allow it to do new tasks

AngularJS
Back End

- Angular has no set requirements for a kind of back end
- Preferably pick a back end that can respond to asynchronous HTTP or Web Socket calls using a REST-style API and JSON

C.15

Aurelia
What Is It?

- Aurelia is a next generation JavaScript client framework that leverages simple conventions to empower your creativity, written with ES6 and ES7

Aurelia
http://aurelia.io/

C.16

D3
What Is It?

- D3 is a JavaScript library for manipulating documents based on data
  - D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation
  - For example, you can use D3 to generate an HTML table from an array of numbers, or use the same data to create an interactive SVG bar chart with smooth transitions and interaction

D3.js
http://d3js.org/