FIREBRAND

Microsoft

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

MHTML5 Courseware

Version 1.2

www.firebrandtraining.com

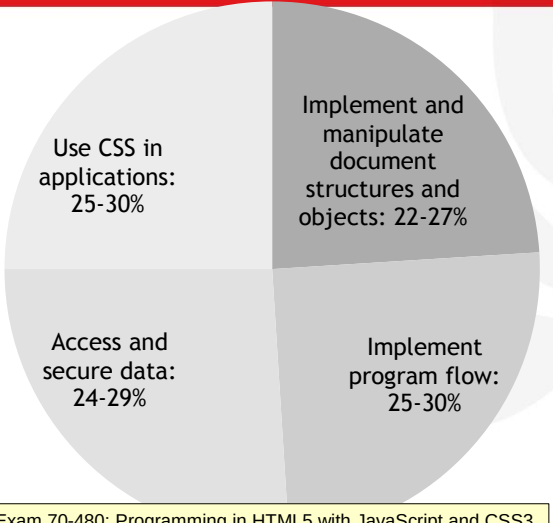
Module 1
Overview of HTML and CSS

Programming in HTML5 with JavaScript and CSS3

Updated 11th April 2014

Programming in HTML5 with JavaScript and CSS3

70-480 Exam Guide to Ratio of Questions



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://spviking.com/2012/12/08/70-480-exam-preparation-guide/

June 2013
39 questions
130 minutes

September 2013
58 questions
120 minutes

December 2013
45 questions
120 minutes
No case study
Time can be an issue!

Microsoft
Specialist

Programming in HTML5 with JavaScript and CSS3 Specialist

Programming in HTML5 with JavaScript and CSS3 1.3

Estimate of Number of Exam Questions per Module

Module	Qs
1: Overview of HTML and CSS	1
2: Creating and Styling HTML Pages	4
3: Introduction to JavaScript	6
4: Creating Forms to Collect and Validate User Input	4
5: Communicating with a Remote Server	5
6: Styling HTML5 by Using CSS3	6
7: Creating Objects and Methods by Using JavaScript	5
8: Creating Interactive Pages by Using HTML5 APIs	2
9: Adding Offline Support to Web Applications	2
10: Implementing an Adaptive User Interface	2
11: Creating Advanced Graphics	1
12: Animating the User Interface	1
13: Implementing Real-time Communication by Using Web Sockets	2
14: Performing Background Processing by Using Web Workers	4
Total questions in exam	**45**

Day 1 55%

Day 2 45%

Programming in HTML5 with JavaScript and CSS3 1.4

Extra Firebrand Materials – Review Slides

Name	Slides
20480.01.Overview	14
20480.02.HTML5.CSS3	9
20480.03.JavaScript	52
20480.04.Form.Input	18
20480.05.Ajax	16
20480.06.CSS3.Layouts	26
20480.07.Objects	34
20480.08.HTML5.APIs	7
20480.09.Offline.Support	8
20480.10.Adaptive.UI	7
20480.11.Graphics	4
20480.12.Animation	5
20480.13.Web.Sockets	6
20480.14.Web.Workers	9
20480.A.Reference.Guide	
20480.B.Unit.Testing.JavaScript	3
20480.C.Cross.Domain.Requests	12

Day 1 55%

Day 2 45%

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

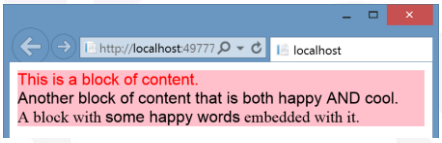
CSS
http://msdn.microsoft.com/library/ie/hh673536(v=vs.85).aspx

HTML

div, span, id, and class

1.5

- 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

1.6

- 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 4.01 <a> could be a hyperlink or an anchor…
  - …but in HTML5 the <a> tag is always a hyperlink

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

```
<a name="bookmark" />
```

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

CSS
Origins

1.7

- 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 more weight than rules in user style sheets
  - Precedence is reversed, however, for **!important** rules
  - All user and author rules have more weight than rules in the user agent's default style sheet

Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, 6.4 The cascade
http://www.w3.org/TR/2011/REC-CSS2-20110607/cascade.html#cascade

CSS
Rule Precedence

1.8

- Sort according to importance (normal or important), animations, and origin (author, user, or user agent)
- In descending order of precedence:
  1. user !important declarations (highest precedence)
  2. author !important declarations
  3. CSS animations
  4. author declarations
  5. user declarations
  6. user agent !important declarations
  7. user agent declarations (lowest precedence)
  - In case of equality, the specificity of a value is considered to choose one or the other

Cascade
https://developer.mozilla.org/en-US/docs/Web/CSS/Cascade

CSS
Selector Specificity

1.9

- CSS selectors by *increasing* specificity
  - Universal selectors (*)
  - Type selectors (h1)
  - Class selectors (.cool)
  - Attributes selectors (input[type='text'])
  - Pseudo-classes (a:hover)
  - ID selectors (#fname)
  - Inline style

```
<div id="test">
    <span>Text</span>
</div>
```

```
div#test span {
    color: green;
}
span {
    color: red;
}
div span {
    color: blue;
}
```

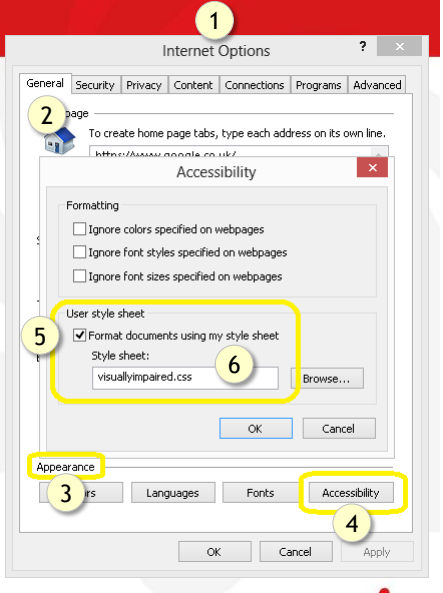
- No matter what the order, the text will be green because that rule is most specific
  - Also, the rule for blue overwrites the rule for red, no matter the order of the rules

Specificity
https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity

CSS
Setting a User Style Sheet

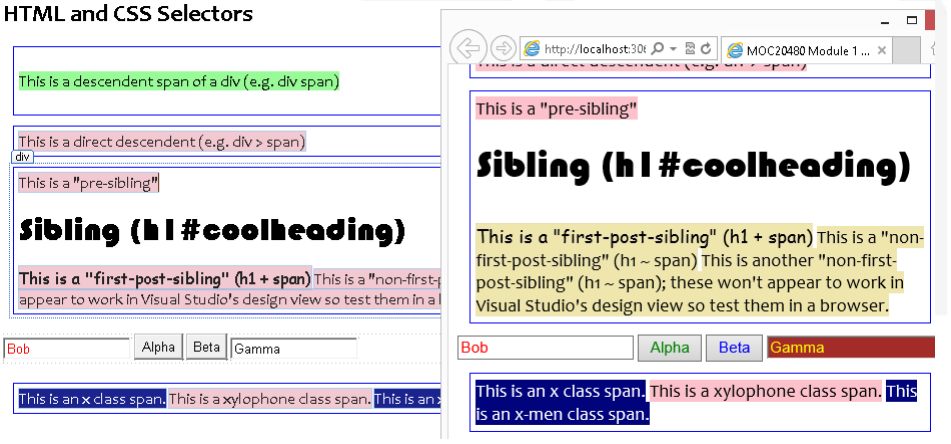
1.10

- 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



Visual Studio Design View

- 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;
}
```

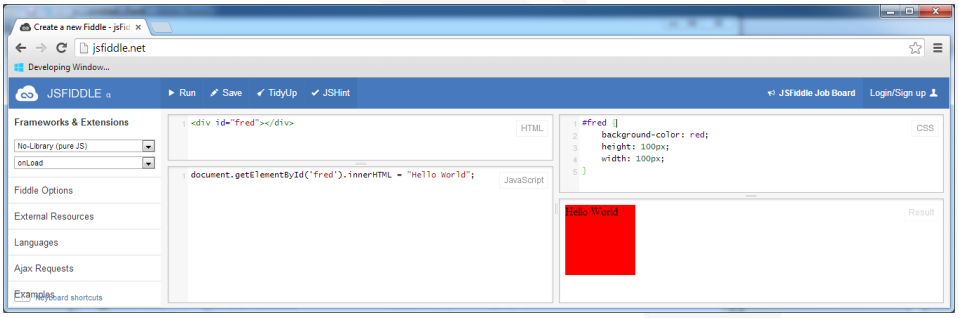
- Notes
  - Labs have “code signature blocks”—just ignore them
  - Demos in Module 1 enable NTLM Authentication; switch this off in Project Properties—Start Options

1.13

Further Study

Experimenting with HTML5, CSS3, and JavaScript

- jsfiddle.net is great for experimenting on any device
  - Blocks for HTML, CSS, JavaScript, and output
  - Can import common libraries easily



- jsperf.com is great for test cases for benchmarking

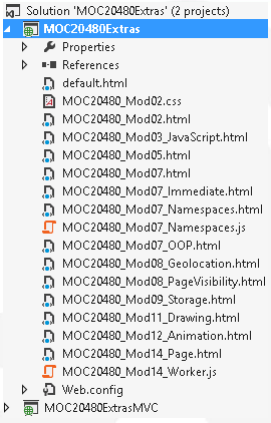
jsfiddle	jsPerf — JavaScript performance playground
http://jsfiddle.net/	http://jsperf.com/

1.14

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



Module 2
Creating and Styling HTML Pages

Programming in HTML5 with JavaScript and CSS3

Updated 11th April 2014

Creating and Styling HTML Pages

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)

- Page 2-23, 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**

Get Started Using HTML5
http://www.microsoft.com/web/post/get-started-using-html5?sf1284466=1

2.3

Can I use...

= Supported = Not supported = Partially supported = Support unknown

New semantic elements - **Working Draft**

HTML5 offers some new elements, primarily for semantic purposes. The elements include: section, article, aside, header, footer, nav, figure, figcaption, time, mark.

*Usage stats:	Global
Support:	83.12%
Partial support:	3.54%
Total:	86.66%

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							

2.4

HTML

Semantic HTML5 Elements

Tag	Description
article	Defines independent, self-contained content. An article should make sense on its own and it should be possible to distribute it independently from the rest of the site.
aside	Defines some content aside from the content it is placed in. The aside content should be related to the surrounding content.
div	Defines a division or a section in an HTML document. The tag is used to group block-elements to format them with CSS.
nav	Defines a set 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.
section	Defines sections such as chapters, headers, footers of the document.
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 h2 and h3

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

Heading 1

Heading 2

Heading 3

Heading 1

Heading 2

Heading 3

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 so to represent a false value the attribute has to be omitted

HTML
iframe

2.8

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

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

- New attributes in HTML5
  - sandbox = allow-forms, allow-same-origin, allow-scripts, allow-top-navigation
  - seamless=seamless

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

CSS
nth-child selector

2.9

- 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/

Further Study

2.10

- To learn techniques, try CSS Zen Garden


CSS Zen Garden
Resource Guide

This page used to contain a list of links to various CSS-related resources. Because of many changes to basic CSS techniques and methods since it was first built in 2003, the former list has been retired. Instead, here are other resources that offer more modern tips and inspiration.

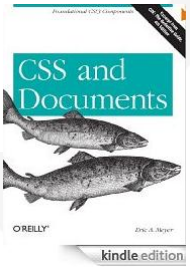
- CSS3.info
- CSS on MDN
- CSS Snippets on CSS-Tricks
- CSS on Quirks Mode

Zen Garden Coding & Design Process Write-ups

In the words of the designers themselves, how certain Zen Garden designs came to be.

- Douglas Bowman — Design

  A design process revealed...



Kindle Price: £0.00

CSS and Documents [Kindle Edition] by Eric A. Meyer (Author)
http://www.amazon.co.uk/CSS-Documents-Eric-A-Meyer-ebook/dp/B008RBQHTW/

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

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

Module 3
Introduction to JavaScript

Programming in HTML5 with JavaScript and CSS3

Updated 11th April 2014

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

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

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

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

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

3.3 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();
```

3.4 JavaScript console

- *Most* browsers have a console that you can output to using console.log method

```
console.log("Hello World");
```

- View the console in Internet Explorer with F12 Developer Tools and click the Console tab
- Some browsers do not have a console so you should check for its existence before using it

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

- Or

```
if(console) {
```

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

JavaScript
Adding Script Statically to a Page

3.5

- 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>
  <h2>Warning</h2>
  <p>Your browser does not support JavaScript
     or you don't have it enabled.</p>
</noscript>
```

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

JavaScript
Equality and Truthiness

3.6

- What do these expressions evaluate to?

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

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

JavaScript
Floating Point Arithmetic

3.7

- 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	11
20	10100
2.5	10.1
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

JavaScript
How to Deal with Floating Point Numbers

3.8

- 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 ==
  - 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 == 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

JavaScript
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

JavaScript
Host Objects

3.10

- 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

```
window.alert('message in a dialog box');
```

- document: represents the page (DOM)

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

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

JavaScript
Duck Typing

3.11

- Everything in JavaScript is a duck type object
  - 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

JavaScript
for loops

3.12

- 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
}
```

JavaScript
for loop with multiple counters

3.13

- 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
```

JavaScript
Arrays

3.14

- To create an array

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

```
var muppets = [ "Kermit", "Fozzie", "Miss Piggy" ]; // simpler syntax
```

- 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

JavaScript
Arrays as Stacks and Queues

3.15

- 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#
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

JavaScript
Functions

3.16

- 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

JavaScript
Function Arguments

3.17

- 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
```

JavaScript
Function Overloading

3.18

- 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
```

JavaScript
Referencing and Executing Functions

- 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());
```

JavaScript
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

JavaScript
Documenting for IntelliSense

3.21

- 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;
}
```

```
areaFunction(
  Number areaFunction(Number radius)
  Determines the area of a circle when provided a radius parameter.
  radius: The radius of the circle.
```

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

JavaScript
Understanding Function Parameter Documentation

3.22

- 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

```
Object.create(prototype, descriptors)
```

Parameters

prototype
Required. The object to use as a prototype. May be **null**.

descriptors
Optional. A JavaScript object that contains one or more p

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

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.

JavaScript
Understanding Function Parameter Documentation
3.23

- 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.

JavaScript
Passing Parameters by Value and by Reference
3.24

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

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

function processNumber(input) {
  input = 2;
}

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

```
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
```

JavaScript
document and window Events

3.25

- 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 IE weirdness
function addEvent(obj, eventName, listener) {
  if (obj.addEventListener) {
    obj.addEventListener(eventName, listener, false);
  } else { // IE 8 and earlier
    obj.attachEvent("on" + eventName, listener);
  }
}
```

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

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

JavaScript
DOM Event Handling

3.26

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

```
<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! Internet Explorer only supports bubbling

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

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

JavaScript
Event Object

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

```
function eventHandler(eventObject) {
```

- Some useful properties
  - currentTarget: the element whose listeners caught the event
  - target: the element that triggered the event
  - cancelable: can an event have its default action prevented
  - eventPhase: one of the event phase constants:
    - CAPTURING_PHASE (1), AT_TARGET (2), BUBBLING_PHASE (3)
- Some useful methods
  - preventDefault(): any default action normally taken
  - stopPropagation(): prevent further capturing/bubbling

JavaScript
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

Three Common Ways of Indicating an Error

- Some objects raise "error" events
  - Write an event handler for the event and check the event parameter's properties
- Some functions return objects that contain properties that you should check
  - 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/finally block around the code that could throw an error (see the next slide for examples of catching, throwing, and nesting)

Catching and Throwing Errors

```
try {
  try {
    console.log("Nested try running...");
    // to throw a custom error pass unique number and string
    throw new Error(301, "an error message");
  }
  catch (error) { // number, description, message properties
    if (error.number === 301) console.log("my error");
      console.log("Nested catch " + error.message);
    throw error; // rethrow
  }
  finally { /* clean up */ }
}
catch (e) {
  console.log("Outer catch " + e.message);
}
finally {
  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

JavaScript
Null-Coalescing Operator

- 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;
```

JavaScript
this in functions

- 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
```

JavaScript
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";
});
```

JavaScript
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

JavaScript
this in Event Handlers

3.35

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

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

JavaScript
OLN and JSON

3.36

- 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
```

  - An optional second argument can be used to customize the stringify and parse processes
  - JSON cannot serialize every part of an object (see next slide)

Limitations of JSON

- 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

Binding a Function to *this*

- You can manually bind `this` to 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
```

JavaScript
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>
```

```
var div = document.getElementById("kermit");
div.classList.remove("foo");
div.classList.add("anotherclass");
```

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

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

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 1.11.0 on 24th March 2014)
  - jQuery 2.x (version 2.1.0 on 24th March 2014)
  - 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 no IE 6/7/8 users are visiting your site

jQuery
http://jquery.com/

jQuery
…and Visual Studio 2012 RTM

3.41

- Note: the Microsoft exams cover jQuery 1.7.1
  - That is the version included with Visual Studio 2012 RTM
  - Warning! Some APIs are deprecated but might still be in the exam
  - The exams only cover the core of jQuery, not any of its extensions such as jQuery UI
- To use a Content Delivery Network (CDN) instead of Scripts


```
http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.min.js
```

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

jQuery
Basic Selectors and jQuery Extensions

3.42

- 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' });
```

```
// You'll frequently see $(elem) used to wrap a jQuery object
// around the element, because jQuery makes lots of things a
// lot simpler. You might hide the element, for example:
$(element).hide();
```

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

- 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/

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

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

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

- 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

3.47

- .children(['selector'])
  - All children
  - All children that match selector

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

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

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

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

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

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

- .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

3.48

- 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

3.49

- 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

3.50

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

```
$("#dataTable tbody tr").on("click", function (e) {
  alert($(this).text());
});
```

```
function notify() { alert("clicked"); }
$("button").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()

```
$("button").off("click", notify);
```

  - Remove all event handlers previously attached with .on()

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

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

Adding Elements and Content

- To add new elements as children to an existing element

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

- To *replace* an existing element

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

- To insert *inside* an element

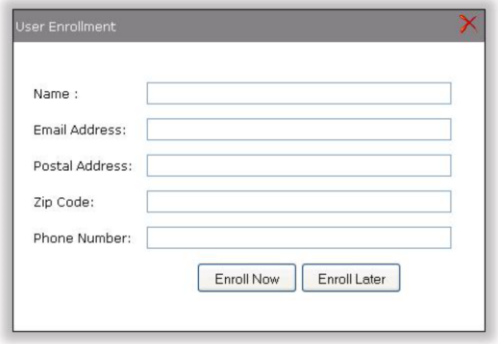
```
$('#mydiv').html('<span>...</span>');
```

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

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

Modal Popups

- 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

Unit Testing JavaScript

Introduction To JavaScript Unit Testing

- 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/

Unit Testing JavaScript

JavaScript unit test tools for TDD

- There are many test tools for TDD 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

Module 4
Creating Forms to Collect and Validate User Input

Programming in HTML5 with JavaScript and CSS3

Updated 11th April 2014

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

4.3

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%

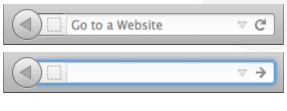
Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
	8.0					4.2-4.3		4.0		
	9.0	22.0	28.0	5.1		5.0-5.1		4.1	7.0	
Current	10.0	23.0	29.0	6.0	16.0	6.0-6.1	5.0-7.0	4.2	10.0	10.0
Near future	11.0	24.0	30.0	7.0	17.0	7.0				
Farther future		25.0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

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

4.4

HTML input element

- <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
  - autocomplete=on|off
  - autofocus=autofocus
  - pattern=*regular_expression*
  - placeholder=*text*
  - required=required



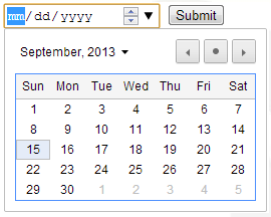
HTML <input> Tag
http://www.w3schools.com/tags/tag_input.asp

4.5 input type attribute

type
color
date
datetime
datetime-local
email
month
number
range
search
tel
time
url
week

```
<input type="date" name="bday" />
```

```
document.getElementById('bday')
  .addEventListener("click",
  function () {
    alert(this.value);
  }, false);
```




```
<input type="number" name="quantity" min="1" max="5" />
```

```
<input type="range" name="points"
       step="10" value="50"
       min="0" max="100" />
```

Control screenshots are from Chrome

HTML5 Input Types
http://www.w3schools.com/html/html5_form_input_types.asp

4.6 progress and meter

- 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="0" 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 value="70" max="100" />
```

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

4.7 output

- 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

4.8 Common Events

- 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?
- Form events
  - blur: when a form element loses focus
  - change: when the content of a form 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

4.9

Detecting HTML5 Input Support

- 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/

4.10

JavaScript
Parsing Numbers and Dates

- 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

```
// parse a date in 'yyyy-mm-dd' format
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');
```

JavaScript
Validating Input

4.11

- How to check input against an array of valid entries
  - Define an array of lookup values (could come from service)

```
var arr = new Array();
arr["JavaScript"] = false;
arr["C#"] = true;
arr["Java"] = false;
arr["Visual Basic"] = true;
```

```
<input id="MSLang" />
<div id="Message"></div>
```

  - Lookup the value in the array from the users' input

```
var lang = document.getElementById("MSLang").value;
// lookup in the array using language as a dictionary key
if(!arr[lang]) {
  var msg = "Not a valid language. ";
  msg += "Valid choices include: ";
  for (var item in arr) // build a list of valid choices
    if(arr[item]) txt += item + " ";
  document.getElementById("Message").innerText = msg;
}
```

Validation
Getting Data for Validation using jQuery

4.12

- .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
Basics

4.13

- Regular expressions can validate and process text
- When validating input, include the leading caret and trailing dollar to avoid security vulnerabilities
  - ^ means start of input; $ means end of input
  - \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}$/;              // bb
var r2 = /^[abc]{3}$/;          // 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("abc")) {
```

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

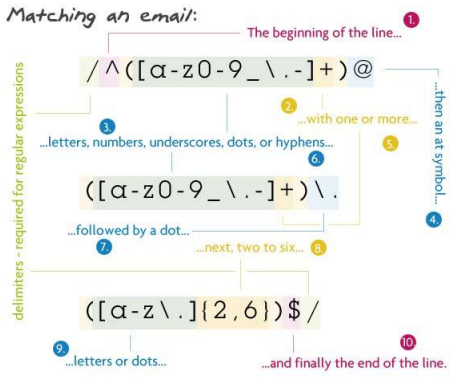
Regular Expression Basic Syntax Reference
http://www.regular-expressions.info/reference.html

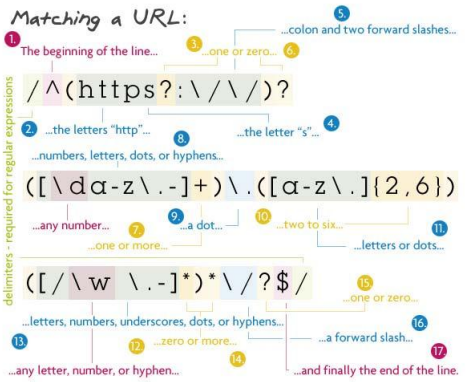
Regular Expressions
Common Characters

4.14

^	Start of line/string	$	End of line/string
\t	Tab	\n	New line
\b	Boundary of word	\B	Non-boundary
*	Zero or more times	+	One or more times
?	Zero or one time	x|y	Either x or y
.	Single character	[^aeiou]	*Not* in set of chars
[xyz]	Any of the enclosed characters	[a-z]	A range of characters
\d \D	Digit Non-digit	\w \W	Word character non-word character
\s \S	White space / non-white space	\G	Match at point previous match ended
\040	ASCII as octal	\u0020	Unicode as hex

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
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

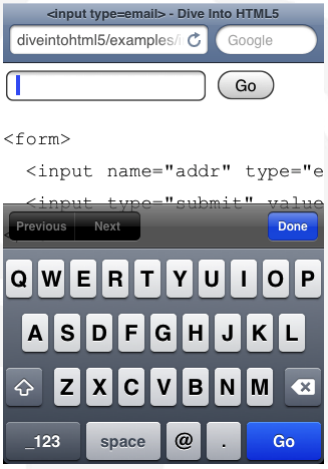
Allow six digits only

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
```

Further Study
HTML5 Forms

- How to use the new HTML5 form features with graceful fallback, for example, email
  - "To sum up: there's no downside to converting all your email address form fields to type="email" immediately. Virtually no one will even notice, except iPhone users, who probably won't notice either. But the ones who do notice will smile quietly and thank you for making their web experience just a little easier."



A FORM OF MADNESS
http://diveintohtml5.info/forms.html

Lab Alternative

- Form input
  - Create a page that uses some of the new input types 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

Module 5
Communicating with a Remote Server

Programming in HTML5 with JavaScript and CSS3

Updated 11th April 2014

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

5.3

MOC Errata

- 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:

5.4

Sanitizing

Encoding and Decoding

- 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

jQuery
$.get function

5.5

- jQuery.get(url [, data] [, success(data, textStatus, jqXHR)] [, dataType])
- success(data, textStatus, jqXHR)
  - 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) {
    $('.result').html(data);
    alert('Status: ' + textStatus);
});
```

The name of a property can be any string, so choices[] is valid. We're using it to indicate that we expect the value of the property to be an array. It is not a special syntax. ☺

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

jQuery
$.ajax function caching

5.6

- 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/

jQuery

$.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"[3] 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 [3]See Appendix C: Cross Domain Requests

jQuery

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

Name	Description	Default
crossDomain[3]	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[3]	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 [3]See Appendix C: Cross Domain Requests

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

*If dataType is jsonp, type must be GET

jQuery

$.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

jQuery

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

jQuery

What is *this* Inside jQuery Ajax Calls?

5.11

- If you want to refer to `this` in an Ajax callback

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

- Or set the context property

```
$.ajax({
  //...
  context: this,
  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

jQuery

ajax method returns jqXHR

5.12

- 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*)

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

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

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

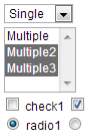
jQuery
Event Handlers for Ajax Operations

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

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

jQuery
Serializing Forms

- serialize() method
  - Returns a string in standard URL-encoded notation
  - Encode a set of form elements as a string for submission




```
var str = $("form").serialize();
```

```
single=Single&multiple=Multiple2&multiple=Multiple3&check=check2&radio=radio1
```

```
var str = $("form").serializeArray();
```

```
[
  {
    name: "single",
    value: "Single"
  },
  {
    name: "multiple",
    value: "Multiple2"
  },
```

- Returns a JavaScript array of objects, ready to be encoded JSON.stringify()

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

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

XML
Parsing XML

5.15

- If you're lucky, the services you call will return JSON
  - If not, you might have to parse XML ☹
- All modern 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);
}
```

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

XML
XML DOM

5.16

- 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

Module 6
Styling HTML5 by Using CSS3

Programming in HTML5 with JavaScript and CSS3

Updated 11th April 2014

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%

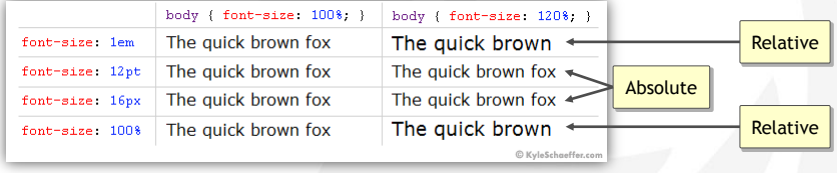
Method of positioning elements in horizontal or vertical stacks.

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							

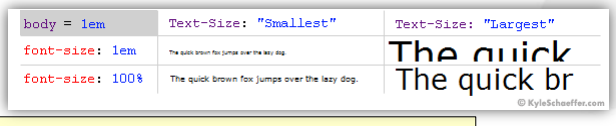
Text

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

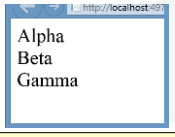
- Sizing with px
  - Pixels provide reliability and consistency, but IE will not allow adjustments to font size (although “zoom” gets round this)
- Sizing with em
  - Measurements are relative to parent element, so they compound with nested elements

```
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

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

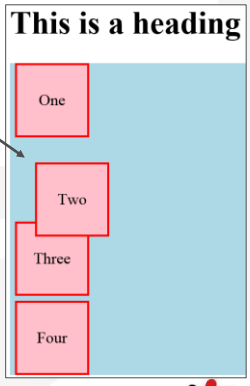
35+ awesome and free @font-face kits
http://www.webdesignerdepot.com/2012/09/35-awesome-and-free-font-face-kits/

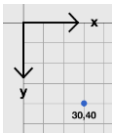
```
position: static;
```

- position has an initial value of static
  - Other choices: relative, absolute, or fixed*
  - top, right, bottom, and left can be set when not static

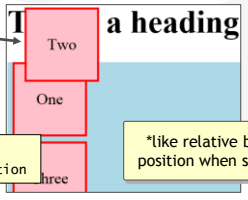
```
<div id="ancester">
  <div id="one">One</div>
  <div id="two">Two</div>
  <div id="three">Three</div>
  <div id="four">Four</div>
</div>
```

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






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



position
https://developer.mozilla.org/en-US/docs/Web/CSS/position

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


```
/* 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

6.9

- The float property specifies whether or not a box (an element) should float
  - left, right, none, inherit

```
#three { float: right; }
```

- 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
  - Usually this means all the way to the left or right of the containing element
  - The elements after the floating element will flow around it
  - The elements before the floating element will not be affected

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
clear

6.10

- Turning off float by using clear
  - left, right, both, none, inherit

```
#four { clear: both; }
```

- 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

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

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;
```

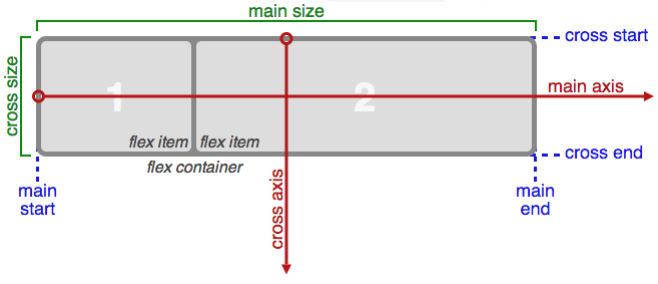
A Complete Guide to Flexbox
http://css-tricks.com/snippets/css/a-guide-to-flexbox/

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)

6.13

- 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>;
```

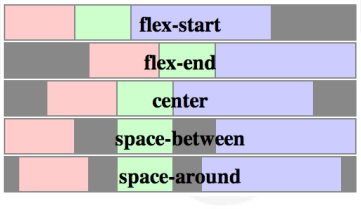
Flex Layout
justify-content (apply to container)

6.14

- 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



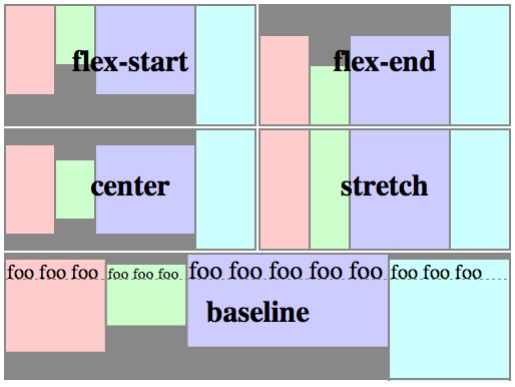
Flex Layout
align-items (apply to container)

6.15

- 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
order and flex (apply to flex item)

6.16

- 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>;
```

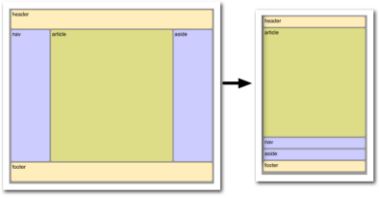
- flex allows a flex item to grow if necessary

```
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

CSS3 Flexible Box Example for Internet Explorer 10


```
<header>...</header>
<div id='main'>
    <article>...</article>
    <nav>...</nav>
    <aside>...</aside>
</div>
<footer>...</footer>
```



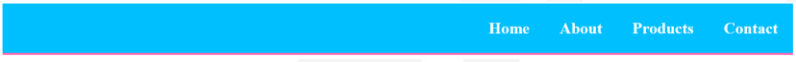
```
#main { display: -ms-flexbox; }
#main > article { -ms-flex-order: 2; min-width: 12em; -ms-flex: 1; }
#main > nav { -ms-flex-order: 1; width: 200px; }
#main > aside { -ms-flex-order: 3; width: 200px; }
```

```
@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/

CSS3 Flexible Box Navigation Example (1 of 2)



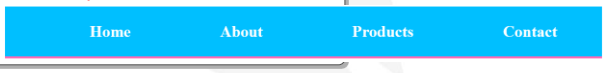

```
.navigation {
  list-style: none;
  display: -ms-flexbox;
  background: deepskyblue;
  border-bottom: 2px solid hotpink;
  -ms-flex-pack: end;
}
.navigation a {
  text-decoration: none;
  display: block;
  padding: 15px;
  color: white;
  font-weight: bold;
}
.navigation a:hover {
  background: darken(deepskyblue, 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

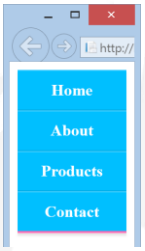
CSS3 Flexible Box Navigation Example (2 of 2)

```
@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

6.21

- 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

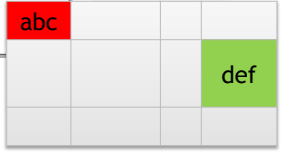
6.22

- 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>
```



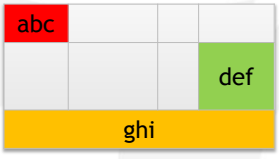
Grid layout reference
http://msdn.microsoft.com/en-us/library/ie/hh772052(v=vs.85).aspx

-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: 1;
    -ms-grid-column-span: 4;
    text-align: center;
}
```

```
<div id="grid">
    ...
    <div id="item3">ghi</div>
    ...
</div>
```



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 and pseudo-classes create abstractions about the document tree

```
a:hover { color: orange; } /* pseudo-class */
p::after { color: pink; } /* pseudo-element */
```

- A pseudo-element is made of two colons (::) followed by the name of the pseudo-element
  - This :: notation is introduced in order to establish a discrimination between pseudo-classes and pseudo-elements
  - For compatibility, user agents must also accept the previous one-colon notation introduced in CSS levels 1 and 2
  - 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

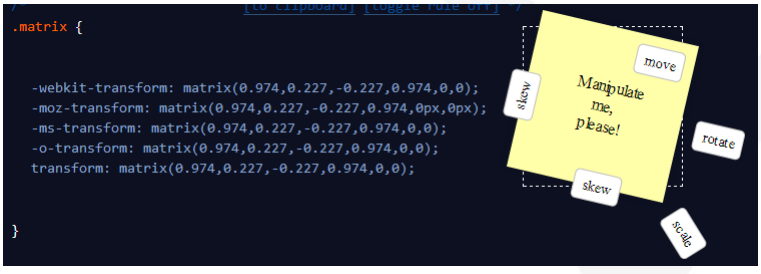
Link Styles

- 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

Lab Alternative

- Experiment with different layout models
- CSS3 Please!
  - Experiment with CSS3



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 11th April 2014

Creating Objects and Methods by Using JavaScript
Contents

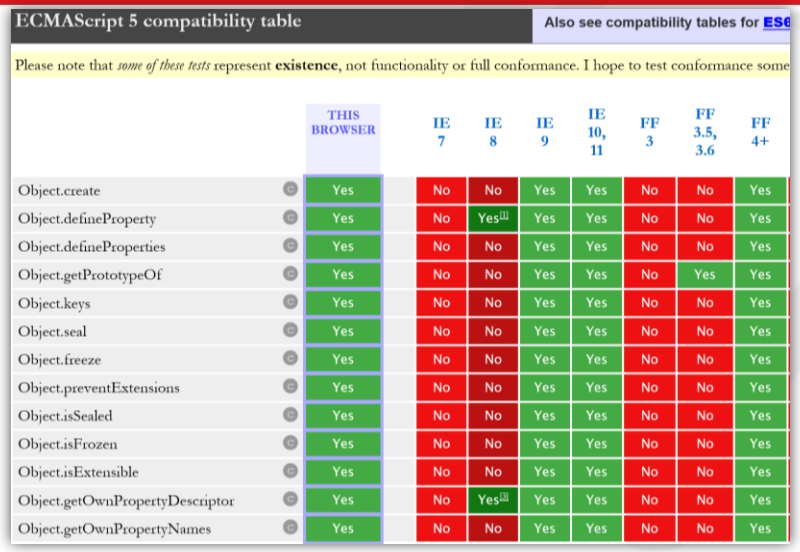
Exam Topic: Create and implement objects and methods
- ❑ Implement native objects
- ❑ Create custom objects and custom properties for native objects using prototypes and functions
- ❑ Inherit from an object
- ❑ Implement native methods and create custom methods

Exam Topic: Establish the scope of objects and variables
- ❑ Define the lifetime of variables
- ❑ Keep objects out of the global namespace
- ❑ Use the "this" keyword to reference an object that fired an event
- ❑ 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

Objects
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/

Scoping
Block Scope using let

- JavaScript 1.7 and later allow block scope using let
  - 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

```
var a = 5;
var b = 10;
if (a === 5) {
    let a = 4; // The scope is inside the if-block
    var b = 1; // The scope is inside the function
    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

Scoping
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";
```

Scoping
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!");
    }
}
```

Objects
Terms

- 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

Objects
What Is An Object?

- An object is a dynamic unordered list of properties
  - Each property has a name and a value
  - A property name can be any string, including empty, but each property name in an object must be unique, like a dictionary
  - A property value can be any value, including functions
- Properties have attributes
  - writable: can it be set? enumerable: is it returned in a for…in loop? configurable: can it be deleted or modified?
- As well as properties, objects have attributes, none of which are directly accessible
  - prototype: an object from which properties are inherited
  - class: a string that categorizes the object (see slide 7.5)
  - extensible: can new properties be added? (see slide 7.6)

Objects
class

- Need to write our own method to read the class

```
function getType(o) {
  if (o === null) return "Null";
  if (o === undefined) return "Undefined";
  var className = Object.prototype.toString.call(o).slice(8, -1);
  // for your custom objects, give them a property named type
  if ((className === "Object") && (o.type)) return o.type;
  return className;
}
```

[object *className*]

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

Note: custom objects, even created with a constructor, will return “Object”, so add a property called type (or similar) to the prototype of your constructors

```
var bob = new Firebrand.Person('Bob', 'Smith', 45);
console.log(getType(bob)); // => Firebrand.Person
```

Objects
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
```

Warning!
preventExtensions(), seal(), and freeze() cannot be undone

- 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

Three Ways to Create an Object

- Object Literal Notation (prototype is Object.prototype)

```
var emptyObject = {};
var point = { x: 5, y: 12 };
```

```
var book = { // for names with spaces
  "main title": "Exam 70-480",
  subtitle: "Programming with HTML5",
  "for": "Beginners"
}; // or reserved keywords
```

- 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(point);            // inherits x and y
var o2 = Object.create(null);             // no inherited properties
var o3 = Object.create(Object.prototype); // like new Object or {}
```

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!)

Objects
prototype Property

- To get the prototype of an object
  - For a non-function

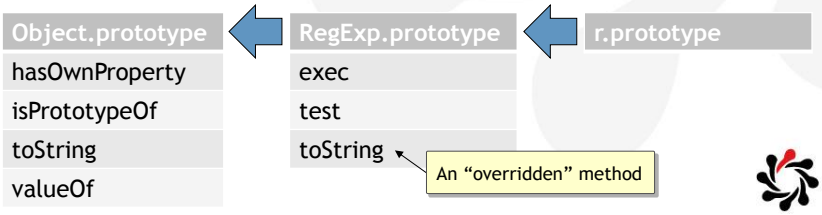
```
var prototypeOfpoint = Object.getPrototypeOf(point);
```

  - For a function

```
var prototypeOfcalc = calc.prototype;
```

- Prototypes form “chains” of inheritance

```
var r = new RegExp("[dh]og");
```



Objects
Object.prototype

- (Almost) all objects chain back to Object.prototype so share these inherited 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

Objects
Object

- Object also 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

Inherited from Function: apply, call, toSource, toString

*Experimental and very slow!

Objects
Querying and Setting Properties

- To get, create, or set a property, use the dot (.) or square bracket ([]) operators

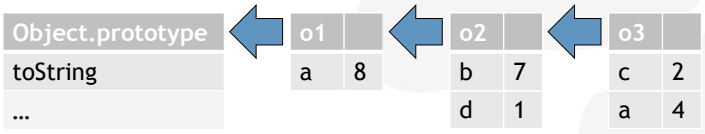
```
// must not have spaces or be a reserved word
console.log(object.property);
console.log(object["property"]); // must be a string
```

```
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

Objects
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!

Objects
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";
```

Objects
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, y, and z, but not toString
```

Objects
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
```

Objects
Functions

- 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

Objects
Invoking Functions

- 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

Objects
To Determine 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
```

Objects
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 bankAccount(name, balance) {
  // initialize object with shared "static" members
  var ba = Object.create(bankAccount.staticMembers);
  // define and initialize "instance" members
  ba.name = name;
  ba.balance = balance;
  return ba;
}
bankAccount.staticMembers = {
  // define "static" members
  interestRate: 3.5,
  toString: function() {
    return this.name + ": " + this.balance;
  }, // and so on
```

```
// create a bankAccount using the factory
var baSmith = bankAccount("Mr. Smith", 300);
console.log(baSmith); // => "Mr. Smith: 300"
```

Objects
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
```

```
// create a bankAccount using the constructor
var baSmith = new BankAccount("Mr. Smith", 300);
console.log(baSmith); // => "Mr. Smith: 300"
```

Objects
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
```

Objects
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
```

Objects
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)

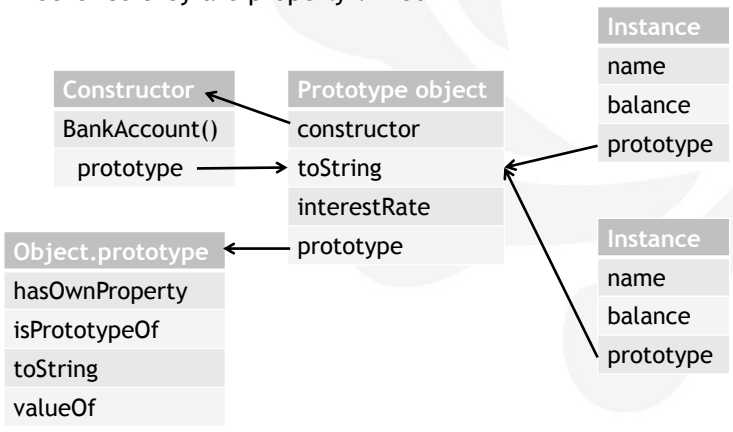
```
p1.constructor.prototype.getMarried = function(spouse) { };
p2.getMarried(p1);
```

  - Of course, if you have the original function then you could do this too

```
Firebrand.Person.prototype.getMarried = function(spouse) { };
```

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



Encapsulation

- 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 this._balance; },
    set: function(value) { this._balance = value; }
  });
  // use the accessor property to set the initial balance
  this.balance = balance;
```

```
var baSmith = new BankAccount("Mr. Smith", 300);
console.log(baSmith);          // => "Mr. Smith: 300"
baSmith.balance = 400;         // calls the setter
console.log(baSmith.balance); // calls the getter
```

Objects
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

```
Number.prototype.times = function (f, context) {
  var n = Number(this);
  for (var i = 1; i <= n; i++) f.call(context, i);
};
```

```
var n = 3;
n.times(function (n) { console.log(n + " hello"); });
```

```
1 hello
2 hello
3 hello
```

Objects
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

Objects
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 "type"
B.prototype.type = "B";
```

Objects
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

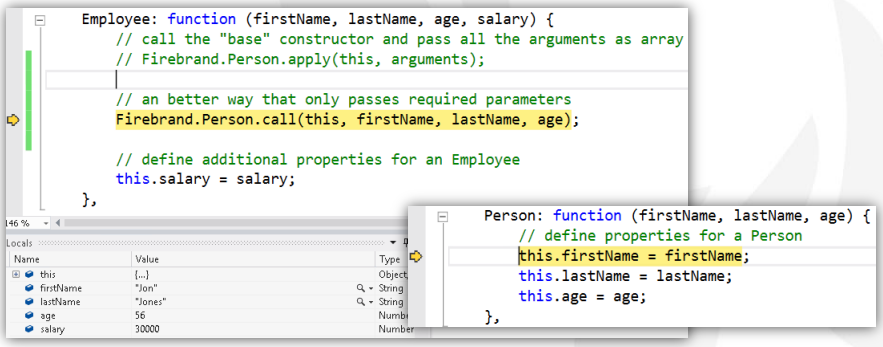
```
function Person(name) { // constructor for Person
  this.name = name;
```

```
function Student(name, subject) {  // constructor for Student
    Person.apply(this, arguments); // call base constructor
    this.subject = subject;
}
Student.prototype = Object.create(Person.prototype);
Student.prototype.constructor = Student;
```

```
Student.prototype.toString = function () {
  return Person.prototype.toString.apply(this, arguments) + " (BSc. Hons.)";
};
```

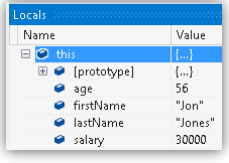
Objects
Chaining Constructors Example

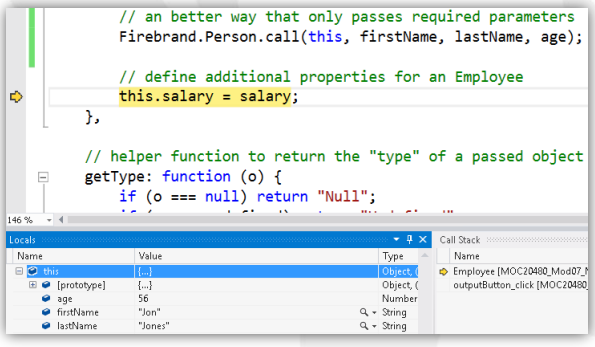
- Stepping through constructor for Employee that need to call constructor from Person



Objects
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

Objects

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
- 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;
};
```

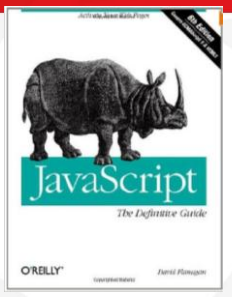
```
var personFromService = JSON.parse(" ... ");
var personWithPrototype = Object.createObject(Person.prototype,
  generatePropertyDescriptorsFor(personFromService));
```

Lab Alternative

- Note: the *inherit* method used in the lab is a custom created function for convenience, NOT a standard part of JavaScript, and does not work well with non-IE!
- Create an object hierarchy
  - Person
    - Student
    - Employee
  - Animal
    - Dog
    - Cat
- Experiment with inheritance, overriding, and so on

Further Study
JavaScript

- JavaScript: The Definitive Guide
  - David Flanagan, 10 May 2011
  - ISBN-10: 0596805527
  - ISBN-13: 978-0596805524
  - Edition: 6, Paperback: 1100 pages
- JavaScript: The Good Parts
  - Douglas Crockford, 15 May 2008
  - ISBN-10: 0596517742
  - ISBN-13: 978-0596517748
  - Paperback: 172 pages



Further Study
Useful Links

- Although JavaScript is not a true OOP language, we can approximate some OOP features
  - OOP in JS, Part 1 : Public/Private Variables and Methods
    http://phrogz.net/js/classes/OOPinJS.html
  - OOP in JS, Part 2 : Inheritance
    http://phrogz.net/js/classes/OOPinJS2.html
  - Extending JavaScript Objects and Classes
    http://phrogz.net/js/classes/ExtendingJavaScriptObjectsAndClasses.html

Module 8
Creating Interactive Web Pages by Using HTML5 APIs

Programming in HTML5 with JavaScript and CSS3

Updated 11th April 2014

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

Geolocation API Specification
http://dev.w3.org/geo/api/spec-source.html#api_description

8.3

MOC Errata

- Page 8-3, the 2nd slide
  - The MOC says

```
<input type="field"
```

  - It should have said

```
<input type="file"
```

8.4

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							

Geolocation
How to Get Your Location

8.5

- To get your location once

```
navigator.geolocation.getCurrentPosition(
  successCallback, errorCallback,
  { enableHighAccuracy: true, timeout: 2000 });
```

```
function successCallback(e) {
  // e.coords.latitude
  // e.coords.longitude
```

- To get your location at a regular interval

```
var id = navigator.geolocation.watchPosition(
  successCallback, errorCallback,
  { enableHighAccuracy: true, maximumAge: 5000 });
```

```
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);
```

Multimedia
Video Sources

8.6

- You can specify a list of alternative sources for browsers that do not understand some video formats

```
<video controls="controls" autoplay="autoplay">
  <source src="Videos\small.mp4" type="video/mp4" />
  <source src="Videos\small.ogv" type="video/ogg" />
  <!-- embed Flash via the object tag and set parameters -->
  <object type="application/x-shockwave-flash" data="...">
    <param name="movie" value="..." />
  </object>
</video>
```

Drag and Drop
Example

- 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 11th April 2014

Adding Offline Support to Web Applications

Contents

Exam Topic: Implement HTML5 APIs
- Implement:
  - Storage APIs
  - AppCache API

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

# Explicitly cached 'master entries'.
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 11th April 2014

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

- 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

Page 10.20

- Code should use ::before and ::after
- Single-colon (:) is supported for backwards compatibility only

10.3

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							

10.4

Remove, Hide, Show Elements

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 DOM */
.removeMe {
  display: none;
}
```

```
/* to hide */
.hideMe {
  visibility: hidden;
}
```

JavaScript

```
// to disable a control
elem.disabled = true;
// to enable an control
elem.disabled = false;
```

```
// to remove an element from layout
elem.style.display = "none";
// to add an element back
elem.style.display = "block"; // or others
// to hide an element
elem.style.visibility = "hidden";
// to show an element
elem.style.visibility = "visible";
```

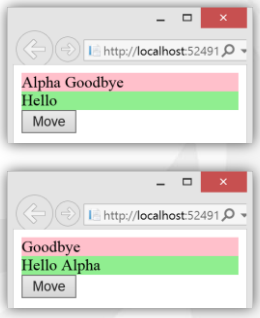
Move Elements

10.5

- 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>
```

```
var elementToMove = document.getElementById('A');
var elementToMoveInto = document.getElementById('B');
elementToMoveInto.appendChild(elementToMove);
```



CSS Printing

10.6

- style and link elements support the MEDIA attribute, which defines the output device for the style sheet
  - Values for MEDIA are screen (default), print and all
  - The print value specifies that the style sheet is used when the page is printed; this value 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
http://msdn.microsoft.com/en-us/library/ms530709(v=vs.85).aspx

CSS
Media Queries

10.7

- 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/

Conditional Comments
Not Supported in Internet Explorer 10 and later

10.8

- Support for conditional comments has been removed in Internet Explorer 10 standards and quirks modes for improved interoperability and compliance with HTML5
- You may opt into Internet Explorer 9 behaviour

```
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE9">
```

- Or use feature detection (see link below)
- 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]-->
IE ignores this
<!--[endif]-->
```

```
<!--[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 11th April 2014

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

Image Maps

- A technology for layering clickable regions onto a static image

```
<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

11.3

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							

11.4

```
<canvas width="300" height="300" id="myCanvas"></canvas>
```

- Get canvas and its 2D context, set fill and stroke styles

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');
context.fillStyle = 'yellow';
context.strokeStyle = 'blue';
```

- Next, either define a path by using rect() or by using moveTo(), lineTo(), arcTo(), bezierCurveTo() and so on

```
context.rect(50, 50, 150, 100); // define a path
```

```
context.fill();   // fill the path
context.stroke(); // draw stroke for path
```

```
// define a complex path
context.beginPath();
context.moveTo(50, 50);
context.lineTo(200, 50);
context.lineTo(200, 150);
context.lineto(50, 150);
context.closePath();
```

- Or define and fill in one method

```
context.fillRect(50, 50, 150, 100);
```

Drawing squares on a canvas
http://falcon80.com/HTMLCanvas/BasicShapes/Square.html

HTML Canvas Reference
http://www.w3schools.com/tags/ref_canvas.asp

Module 12
Animating the User Interface

Programming in HTML5 with
JavaScript and CSS3

Updated 11th April 2014

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

12.3

Can I use...

= Supported = Not supported = Partially supported = Support unknown

CSS3 Transitions - **Working Draft**

Simple method of animating certain properties of an element

*Usage stats: Global
Support: 79.04%

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/

12.4

Transition Example

- A blue box that doubles in size, rotates 180°, and changes to red when the mouse hovers over it

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

```
#box:hover {
  background-color: #FFCCCC;
  width: 200px;
  height: 200px;
  -moz-transform: rotate(180deg);
  -webkit-transform: rotate(180deg);
  transform: rotate(180deg);
}
```

```
#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;
}
```

Using CSS transitions
https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_CSS_transitions

12.5 Transitions Backwards and Forwards

- If you want the transition to use the same duration forwards and backwards, just specify it on the original

```
#box {
  /* transition back to red in 2s */
  background-color: red;
  transition: background-color 2s;
```

```
#box:hover {
  /* transition to blue in 2s */
  background-color: blue;
```

- If you want the transition to use different durations forwards and backwards, specify it on original state and new state

```
#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;
```

12.6 Animation with 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
  - Optionally, 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

Module 13
Implementing Real-time Communication by Using Web Sockets

Programming in HTML5 with JavaScript and CSS3

Updated 11th April 2014

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

Web Sockets
When To Use Web Sockets

13.3

- 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

Web Sockets
On the Server

13.4

- 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);
  }
  class ChatWebSocketHandler : WebSocketHandler {
    public ChatWebSocketHandler() {
    }
    public override void OnOpen() {
    }
    public override void OnMessage(string message) {
    }
  }
}
```

```
using Microsoft.Web.WebSockets;
```

Web Sockets
On the Browser

13.5

- Create and use a WebSocket object

```
$(document).ready(function () {
    websocket = new WebSocket('ws://localhost/api/ChatController');
    websocket.onopen = function () {
    };
    websocket.onerror = function (event) {
    };
    websocket.onmessage = function (event) {
    };
    websocket.send('Hello');
    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

Learn About ASP.NET SignalR
http://www.asp.net/signalr

Module 14
Performing Background Processing by Using Web Workers

Programming in HTML5 with JavaScript and CSS3

Updated 11th April 2014

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);
```

```
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({ error: false, answer: number });
      break;
    case "Stop":    // to safely stop the worker
      self.close(); // use self.close() inside a worker
      break;        // but terminate() outside
    default:
      self.postMessage({ error: true, answer: 0 });
  }
}
```

- To import script libraries

```
importScripts("jquery-1.7.1.js");
```

Web Workers
Timeouts and Intervals

14.7

- 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)

Promises
What Are They?

14.8

- "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/

What's the Big Deal?

- 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,

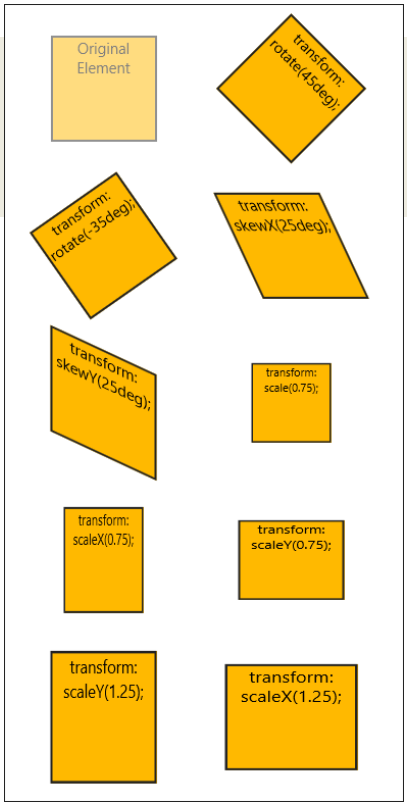
the transition property will cause the color to change on hover using a two seconds and the ease timing function. There is no delay value, so the transition begins immediately.

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);
```

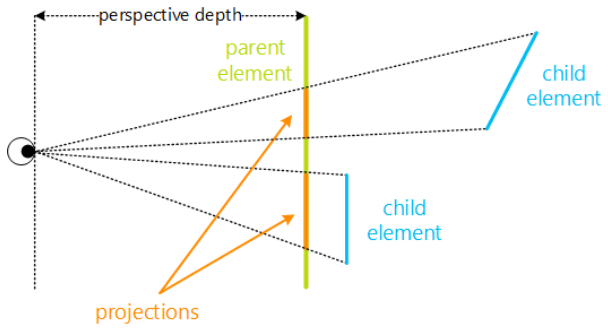


3-D Transform functions

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.

Appendix B
Unit Testing JavaScript

Programming in HTML5 with JavaScript and CSS3

Unit Testing JavaScript

Introduction To JavaScript Unit Testing

- 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/

JavaScript unit test tools for TDD

- There are many test tools for TDD 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

Appendix C
Cross Domain Requests

Programming in HTML5 with JavaScript and CSS3

Updated 11th April 2014

Overview

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-Origin Resource Sharing
http://www.w3.org/TR/cors/

B.3

Overview

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 });
```

B.4

Overview

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

B.5

Client-Side
Using JSONP with jQuery

Exam Topic: 70-480

- 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

B.6

Server-Side
Using JSONP with ASP.NET Web API

Exam Topic: 70-487

- JSONP is not supported by default so we need to add a media type formatter manually
  - Insert a media type formatter to return JSONP when requested

```
var config = GlobalConfiguration.Configuration;
config.Formatters.Insert(0, new JsonpMediaTypeFormatter());
```

  - Make a call requesting JSONP, for example, using jQuery

```
$.ajax({
  type: 'GET',
  url: 'person/update',
  dataType: 'jsonp'
}).done(function (person) {
  alert(person.name);
});
```

JSONP with ASP.NET Web API
http://stackoverflow.com/questions/9421312/jsonp-with-asp-net-web-api

Server-Side

Using JSONP with WCF Services

B.7

Exam Topic: 70-487

- To enable JSONP support for WCF services

```
<system.serviceModel>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true" />
  <standardEndpoints>
    <webScriptEndpoint>
      <standardEndpoint crossDomainScriptAccessEnabled="true" />
```

- To make a jQuery call to your operation

```
$.ajax({
  type: "PUT",
  url: "person/update",
  dataType: "jsonp",
  crossDomain: true,
  data: { name: "John", location: "Boston" }
});
```

Only if you need to force a crossDomain request (such as JSONP) on the same domain

JSONP
http://msdn.microsoft.com/en-us/library/ee834511.aspx

Server-Side

Cross-Origin Resource Sharing (CORS)

B.8

- CORS works by adding new HTTP headers that allow servers to describe the set of origins that are permitted to read that information
- A simple cross-site request is one that
  - Only uses GET, HEAD or POST with no custom headers

```
GET /resources/public-data/ HTTP/1.1
...
Origin: http://foo.example
```

How does Access-Control-Allow-Origin header work?
http://stackoverflow.com/questions/10636611/how-does-access-control-allow-origin-header-work

Server-Side
Configuring IIS to Enable CORS

B.9

Exam Topic: 70-487

- To set custom headers in IIS configuration

```
<system.webServer>
  <httpProtocol>
    <customHeaders>
      <add name="Access-Control-Allow-Origin"  value="*" />
      <add name="Access-Control-Allow-Headers" value="Content-Type" />
      <add name="Access-Control-Allow-Methods" value="GET,POST,PUT,DELETE" />
    </customHeaders>
  </httpProtocol>
</system.webServer>
```

- Typical response from the server if it allows *any* client

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
```

- ...or if it only allow calls from specific domain

```
Access-Control-Allow-Origin: http://foo.example
```

Server-Side
Preflight Requests

B.10

- "Preflighted" requests first send an HTTP OPTIONS request header in order to determine whether the actual request is safe to send

```
OPTIONS /resources/post-here/ HTTP/1.1
...
Origin: http://foo.example
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER
```

- Response from server (max-age in seconds)

```
HTTP/1.1 200 OK
...
Access-Control-Allow-Origin: http://foo.example
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGOTHER
Access-Control-Max-Age: 1728000
```

HTTP access control (CORS)
https://developer.mozilla.org/en-US/docs/HTTP/Access_control_CORS

B.11

Server-Side
Requests with Credentials

Exam Topic: 70-487

- Requests that are cognizant of HTTP Cookies and HTTP Authentication information

```
var xhr = new XMLHttpRequest();
var url = 'http://bar.other/resources/credentialed-content/';
function callOtherDomain() {
  if(xhr) {
    xhr.open('GET', url, true);
    xhr.withCredentials = true;
    xhr.onreadystatechange = handler;
    xhr.send();
  }
```

```
HTTP/1.1 200 OK
...
Access-Control-Allow-Origin: http://foo.example
Access-Control-Allow-Credentials: true
...
Set-Cookie: pageAccess=3; expires=Wed, 31-Dec-2008 01:34:53 GMT
```

B.12

Server-Side
Support in ASP.NET Web API 2

Permission	Request Header	Response Header	Example
Origin	Origin	Access-Control-Allow-Origin	*
HTTP method	Access-Control-Request-Method	Access-Control-Allow-Method	
Request headers	Access-Control-Request-Headers	Access-Control-Allow-Headers	
Response headers		Access-Control-Expose-Headers	
Credentials		Access-Control-Allow-Credentials	
Cache preflight response		Access-Control-Allow-Max-Age	

CORS Support in ASP.NET Web API 2
http://msdn.microsoft.com/en-us/magazine/dn532203.aspx