

70-480 EXAM NOTES

STUDY NOTES - DOMINIKGORECKI.COM

Topics covered in these notes:

- Implement and Manipulate Document Structures and Objects
 - Create the Document Structure
 - Write Code that Interacts with UI controls
 - Apply Styling to HTML elements using JavaScript
 - Implement HTML5 APIs
 - Establish the scope of objects and variables
 - Create and implement objects and methods
- Implement program flow
 - JavaScript flow
 - Raise\handle an event
 - Implement exception handling
 - Implement a callback
 - Create a web worker process
- Access and Secure Data
 - Validate user input using HTML5
 - Validate user input by using JavaScript
 - Consume data
 - Serialize, deserialize, and transmit data
- Use CSS3 in Applications
 - Style HTML text properties
 - Style HTML box properties
 - Create flexible content layout
 - Create an animated and adaptive UI
 - Find elements by using CSS selectors and jQuery
 - Structure a CSS file by using CSS selectors

CONTENTS

Implement and Manipulate Document Structures and Objects	3
Block Versus In-line Level Elements.....	3
Main Content Categories and their Tags	3
Meta Content	3
Sectioning Content	4
Heading Content.....	4
Phrasing Content	4
Embedded Content.....	5
Interactive Content.....	5
Form-associated Content	5
Table Specific Tags.....	5
HTML Outline.....	6
Code that Interacts with UI controls.....	7
Add and Modify Html Elements.....	7
HTML5 Canvas	9
SVG – Scalable Vector Graphics.....	9
HTML5 AUDIO and VIDEO.....	9
Styling HTML Elements Programmatically	10
CSS Positioning	10
CSS Display.....	10
Modifying HTML DOM Style Object.....	10
CSS3 Transforms – 2D.....	10
CSS3 Transforms – 3d	11
CSS3 Tansition.....	11
Implementing HTML5 APIs.....	12
HTML5 Geolocation	12
HTML5 Web Storage	12
Html5 App Cache	13
Establish the scope of Objects	13
Javascript Closures	13
Namespaces in JavaScript	13
Create and Implement Objects and Methods.....	14
Javascript Native Objects	14
Classes in JavaScript.....	15
Pseudo-Classical Inheritance in JavaScript	15
Implement Program Flow	16
Html DOM Events	16
Event Bubbling	16
Implementing Callbacks.....	17
WebSockets API	17
CSS Selectors	18
References	19
Exam Guide-Links.....	19

IMPLEMENT AND MANIPULATE DOCUMENT STRUCTURES AND OBJECTS

May include: Semantic markup including for search engines and screen readers

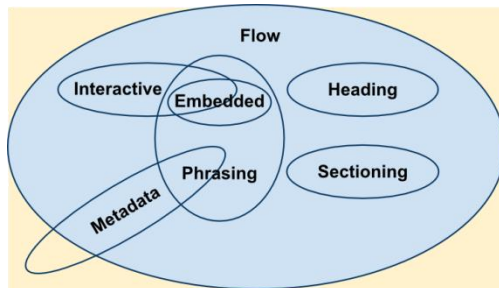
BLOCK VERSUS IN-LINE LEVEL ELEMENTS¹

Block-level elements usually begin with a line break before and create the larger structures of a page versus inline-elements. Block level elements may contain both block and inline elements; however, in-line elements should not contain block-level elements.

The HTML5 standard provides a more detailed set of distinctions.

HTML4.01	HTML5
Block	Flow Content
In-line	Phrasing Content

MAIN CONTENT CATEGORIES AND THEIR TAGS²



¹ https://developer.mozilla.org/en-US/docs/HTML/Block-level_elements
https://developer.mozilla.org/en-US/docs/HTML/Inline_elements

² https://developer.mozilla.org/en-US/docs/HTML/Content_categories

META CONTENT

<base> | <command>* | <link> | <meta> | <noscript> | <script> | <style> | <title>

These elements modify the presentation or behavior of the rest of the page, set up link to other documents, or convey out of band info.

Tags	Explanation\Example
<link>	Mostly used for stylesheets, the link tag sets up a relationship between the page and an external resource <pre><link rel="stylesheet" type="text/css" href="main.css" ></pre>
<meta>	Data about data. Not displayed on page. Used generally to specify page description, keywords, author, last modified... This tag always goes in the <head> tag. <pre><meta name="keywords" content="programming, Microsoft stack"></pre>
<noscript>	Used to provide an alternate content for users that do not have script available on their browser
<script>	Define client-side script (JS). It either contain in-line code or links to a external js file using the src attribute. <pre><script src="main.js" type="text/javascript"></script></pre>
<style>	Used to define in-line style for the doc; unless scoped is used (new to HTML5), this tag should only be used in the <head> tag. <pre><head><style type="text/css"> ... </style></head></pre>
<title>	Required. Defines title of doc.

SECTIONING CONTENT

<section> | <article> | <aside> | <nav>

Used to help create an outline of the document.

Tags	Explanation\Example
<section>*	<p>Defines section in a document. Can be chapters and can contain child sections.</p> <pre><h1>Html5 Outline</h1> <section> <h1>First child</h1> </section> <h1>Firs child of first child. Grandchild of Html5 Outline</h1> </section> </section></pre>
<article>*	<p>Defines an article—-independent, self-contained content. Examples: Forum post, blog post, News story</p> <pre><article>[Contents of Article]</article></pre>
<aside>*	<p>Defines content aside from the content it is placed in, but should be related to content; can be placed as a sidebar in an article.</p> <pre><aside> [content of aside] </aside></pre>
<nav>*	<p>Defines a section of navigation links—intended for major navigation links. Example of possible uses: main navigation, table of contents, prev and next buttons, search form, breadcrumbs.³</p>

* new to HTML5

³ <http://html5doctor.com/nav-element/>

HEADING CONTENT

<h1> ... <h6> | <hgroup>

Defines the title of the section.

PHRASING CONTENT

<abbr>, <audio>, , <bdo>,
, <button>, <canvas>, <cite>, <code>, <command>, <datalist>, <dfn>, , <embed>, <i>, <iframe>, , <input>, <kbd>, <keygen>, <label>, <mark>, <math>, <meter>, <noscript>, <object>, <output>, <progress>, <q>, <ruby>, <samp>, <script>, <select>, <small>, , , <sub>, <sup>, <svg>, <textarea>, <time>, <var>, <video>, <wbr> and plain text (not only consisting of white spaces characters).

Tags	Explanation\Example
<abbr>	<p>Defines an abbreviation</p> <pre>The <abbr title="Unite Nations">UN</abbr> Security Council did not approve the aggression.</pre>
<area>	<p>Defines an area inside an image map</p>
<audio>*	<p>Defines sound audio streams in either MP3, Wav, or Ogg</p> <pre><audio controls> <source src="welcome_message.ogg" type="audio/ogg"> <source src="welcome_message.wav" type="audio/wav"> <source src=" welcome_message.mp3" type="audio/mpeg"> Your browser does not support the audio tag. </audio></pre>
<bdo>	<p>“Bi-Directional Override” to override the current text direction. <bdo dir=[rtl ltr]></p> <pre><bdo dir="rtl">This will render from right-to-left</bdo></pre>
<blockquote>	<p>Specifies section that is quoted from another source.</p> <pre><blockquote cite="www.asdf.com"> . . . </blockquote></pre>

<button>

Clickable button. Element can handle Image and text inside as opposed to <input> cannot.

```
<button type="[button | reset | submit]" > . . .</button>
```

<canvas>

Used to draw graphics on the fly via scripting (js). The canvas tag is only the container.

<cite>

Defines the title of a work

EMBEDDED CONTENT

Imports another resource or content from another mark-up language or namespace

```
<audio> | <canvas> | <embed> | <iframe> | <img> | <math> | <objects> | <svg> | <video>
```

INTERACTIVE CONTENT

Elements designed for user interaction

```
<a> | <button> | <details> | <embed> | <iframe> | <keygen> | <label> | <select> | <textarea> | <audio [with controls]> | <video [with controls]> ...
```

- , if the usemap attribute is present
- <input>, if the type attribute is not in the hidden state
- <menu>, if the type attribute is in the toolbar state
- <object>, if the usemap attribute is present

FORM-ASSOCIATED CONTENT⁴

<BUTTON>

Attributes:

- autofocus* <button ... autofocus>
- disabled <button ... disabled>
- form <button ... form="[form_id]">

⁴ <http://www.html5rocks.com/en/tutorials/forms/html5forms/>

- formaction (overrides <form>'s action) <button ... formaction="URL">
- formmethod (overrides <form>'s method) <button ... formmethod=[get | post]>
- type <button ... type=[button | reset | submit]>

<INPUT>

Input field where the user can enter data used within <form> element. The type of input is specified by the type attribute.

Attributes:

- autocomplete=[off | on]; for [text, search, url, tel, email, password, datepickers, range, and color]
- autofocus
- disabled
- form
- max=[number | date] and min=[number | date]; for [number, range, date, datetime, datetime-local, month, time, and week]
- maxlength="int" (max number of characters)
- name="name_of_input" -- name of the input element
- pattern=[[regex]] -- for text, search, url, tel, email, and password; use "title" to describe pattern to help user
- readonly -- cannot modify
- required -- for text, search, url, tel, email, password, date pickers, number, checkbox, radio and file
- size -- specify size (in chars) of an input element
- step="number" -- legal number of intervals for number, range, date, datetime, datetime-local, month, time and week
- value
- type

Types

TABLE SPECIFIC TAGS

<table> Represents data in 2d or more.

<caption> Defines a table caption and must be inserted

immediately after the <table> tag. Only once caption per table.

```
<table>  
  <caption>Quarterly Sales</caption>  
  <tr><th>...  
  ...  
</table>
```

General Flow

<address> **Defines contact information for the author/owner of a document (when in <body>) or article (when in <article> element)**

```
<address>Page by ... </address>
```

HTML OUTLINE

Resource: <http://www.w3schools.com/tags/default.asp>

<section> -

<area>

<article>

<header>

<aside>

CODE THAT INTERACTS WITH UI CONTROLS

ADD AND MODIFY HTML ELEMENTS

The HTML DOM⁵ is a standard object model and interface for HTML. It defines all of objects and properties of all HTML elements and the methods to access them⁶.

- Everything in an HTML doc is a node:
Entire doc | Every HTML element | text inside each element | every HTML attribute | comments are comment nodes

Finding Elements in the DOM

Modifying html elements on a page requires first finding them in the DOM. The methods used to get a desired HTML element is using are:

`getElementById`, `getElementsByTagName`, `getElementsByName`, `getElementsByClassName`, `querySelector`, and `querySelectorAll`⁷.

Method

`getElementById ([Id])` Query by id of element. Null is returned if no element found.

```
<body>
  <div id="info">Info</div>
  <script>
    var div = document.getElementById('info')
    alert( div.innerHTML )
  </script>
</body>
```

- `getElementsByTagName ([Element Tag])`**
- Searches given tag name (name of html element) and returns a `NodeList` of elements that match.
 - Resulting list is live. It updates itself with the DOM automatically
 - "*" returns all elements

```
var article = document.getElementById("main-
```

⁵ <http://www.w3.org/TR/REC-DOM-Level-1/level-one-core.html#method-getAttributeNode>

⁶ http://www.w3schools.com/html/dom_intro.asp

⁷ <http://javascript.info/tutorial/searching-elements-dom>

```
article");
var paragraphs = article.getElementsByTagName("p");
```

`getElementsByName ([Element Name])` Returns a list (`HTMLCollection` of elements) of elements with a given name attribute. Not often used.

`getElementsByClassName ([class name])`

- Returns a live `NodeList` of all the elements with class name(s).

Get all elements that have rock and star in their class:

```
document.getElementsByClassName("rock star");
```

Get all of the child elements with a class star of the parent that has a class named rock:

```
document.getElementsByClassName("rock").getElementsByClassName("star");
```

`querySelector ([selector(s)])` Returns the first element within the document matching specified selector.

```
document.querySelector(".rock .star");
```

`querySelectorAll ([selector(s)])` Returns a non-live `NodeList` of elements matching thing the specified group of selectors.

Example-- get all of the child elements with a class star of the parent that has a class named rock:

```
document.querySelectorAll(".rock > .star");
```

Manipulating Nodes of a DOM

Method	Description\Example
--------	---------------------

<code>appendChild ([node])</code>	Adds a child node to the specified element (at last position)
--	---

<code>removeChild ([node])</code>	<ul style="list-style-type: none">• Removes a child node.
--	---

- Returns the removed node on success.
- Null on failure.

```
replaceChild([new_node], [old_node])
```

- Replaces child node with another.
- Returns the removed node on success.
- Null on failure.

```
insertBefore([new_node], [existing_node])
```

- Inserts new child node before an existing node.

```
createAttribute([attribute_name])
```

- Creates an attribute with the specified name
- Returns the attribute as an object that can be manipulated

```
createElement([element_name])
```

- Creates an element node
- Returns element object

```
createTextNode([text])
```

Html elements usually consist of both an element node and a text node. For example, a <p> element node with some text inside it (the text node).

- Creates a text node that can be appended to an HTML element node

```
getAttribute([attribute_name])
```

- Returns the value of the attribute with the specified name

```
setAttribute([attribute_name], [value])
```

- Adds a new attribute or changes the value of the existing one

```
hasAttribute([attribute_name])
```

- Returns true if attr exists; false if not

```
removeAttribute([attribute_name])
```

- Removes attribute from a specified element
- If attribute does not exist, no exception is

raised.

HTML DOM Properties

Property

innerHTML

- Getting or setting the content of HTML elements

nodeName

- Same as tag name for element (uppercase), attribute name for attribute
- **Read only**

nodeValue

- Undefined for element
- Inner text for text node
- Attribute value for attr
- **Setable**

nodeType

- Returns type of Node as an integer code
 - Element = 1
 - Attribute = 2
 - Text = 3
 - Comment = 8
 - Document = 9

Changing HTML content

Content of an HTML element can be changed by using the innerHTML property, which is settable.

Creating New HTML Elements

First create the element node (manipulate it as desired) and then append it to an existing node. You can use either appendChild(node) or insertBefore(existing_node, new_node).

Remove HTML Elements

Use `parent.removeChild(node)` to remove a node (HTML Element).

HTML5 CANVAS⁸

- Draw graphics with JS on the fly: graphs, compositions, animations...
- Only has two attributes: height (150 default), width (300 default)

Example:

```
<canvas id="test-canvas" width="300" height="150">
  [fallback content]
</canvas>
```

The Rendering Context

Canvas creates a drawing surface that exposes one or more rendering contexts used to create/manipulate content shown.

```
[canvas].getContext("2d");
```

Checking for support in JS: `If (canvas.getContext)`

SVG – SCALABLE VECTOR GRAPHICS

- Defines graphics in XML format.
- Elements can be animated, available as part of DOM in HTML pages

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

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

- DTD: Document Type Definition is a set of markup declarations and describes which elements and references can appear and where in the document
- Xmlns defines svg namespace

SVG IN HTML

<embed>

```
<embed src="picture.svg" type="image/svg+xml" />
```

⁸ https://developer.mozilla.org/en-US/docs/HTML/Canvas/Tutorial?redirectlocale=en-US&redirectslug=Canvas_tutorial

Supported in all major browsers, allows scripting but deprecated in HTML4 (allowed in HTML5).

<object>

```
<object data="picture.svg" type="image/svg+xml"></object>
```

Supported in all moajor browsers and conforms to HTML4, but scripting not allowed.

<iframe>

```
<iframe src="picture.svg"></iframe>
```

Supported by all major browsers and allows scripting but does not conform to strict HTML4/XHTML DTD.

<SVG> (in markup)

```
<html>
<body>

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

</body>
</html>
```

Not supported by IE8 or earlier, cannot load asynchronously (ux issue for large svgs).

PREDEFINED SVG SHAPES

<rect> | <circle> | <ellipse> | <line> | <polyline> | <polygon> | <path>

HTML5 AUDIO AND VIDEO

STYLING HTML ELEMENTS PROGRAMMATICALLY

CSS POSITIONING

```
position: [static | relative | absolute | fixed | inherit ]
```

- **static** – Normal behavior and “left,right,top,bottom” do not apply
- **relative** – lay out element as if it were static and then adjusts elements position (without changing layout)
- **absolute** - position it at a specified position relative to it's closest positioned ancestor or to the containing block (no room left for element)
- **fixed** – position element relative to the screen's viewpoint—does not move when scrolled (for print, displays on every page)

CSS DISPLAY

```
display: [ none | inline | block | list-item | inline-block | inline-table | table | table-caption | table-cell | table-column | table-column-group | table-row | table-row-group | flex | inline-flex ]
```

- **none** – turns off display with no effect on layout. ie document rendered as if element did not exist
- **inline** – generates inline box like span
- **block** – block level display like paragraph or div
- **list-item** -- block element box with a list-item inline box (bullet)
- **inline-table** – behaves like <table> but as an inline box with a block-level box inside
- **table** – block-level, behaves like <table>
- **flex** – behaves like a block element and lays out its content according to the flexbox model
- **inline-flex** – in-line element but lays out its contents according to the flexbox model

MODIFYING HTML DOM STYLE OBJECT⁹

Example of editing a property of style:

```
document.getElementById("id").style.property = "value";
```

The style object of an HTML DOM exposes each style property as a property of the object. This includes background, border, list, margin\padding, positioning, layout, and text properties (amongst others).

Some examples:

- `style.background` – sets or gets the background properties in one declaration.
- `style.backgroundColor` – sets or gets the css background-color styling
- `style.border` – sets or ges the border css styling as one line declaration.
- `style.margin`, `style.padding`
- `cssText` – style declaration as a string
- `style.display`, `style.visibility`

CSS3 TRANSFORMS – 2D

```
transform: matrix(a,c,b,d,tx,ty) | rotate(angle) | scale(sx[, sy]) | scaleX(sx) | skew(ax[, ay]) | translate(tx[, typ]) | translateX(tx) | translateY(ty)
```

Effect that lets you change shape, size, and position in 2d or 3d. Using it elements can be translated, rotated, and skewed.

- `matrix(a,c,b,d,tx,ty)` - $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ - transform matrix, and `tx`, `ty` for translate
- `rotate(angle)` – rotates element clockwise around transform-origin property by specified angle (20deg)
- `scale(sx,sy)`, `scaleX(sx)`, `scaleY(sy)` – 2d scaling operation with a unitless number from where 1 is 1:1 scale
- `skew(ax,ay)`, `skewX(ax)`, `skewY(ay)` – Non-standard, removed from latest draft of CSS3. Avoid using. Same effect can be created with `matrix(1, tan(ay), tan(ax), 1, 0, 0)`;
- `translate(tx, ty)`, `translateX(tx)`, `translateY(ty)` – 2d translation

⁹ http://www.w3schools.com/jsref/dom_obj_style.asp

CSS3 TRANSFORMS – 3D^{10 11}

```
perspective: [number in pixels]
perspective-origin: [percent, percent]

transform: translate3d(tx,ty,tz) | scale3d(sx,sy,sz) | rotate3d(rx, ry,
rz, angle) | matrix3d(n,n,n,n,n,n,n,n)
```

To activate 3d space, add element using the `perspective` css property or `transform: perspective(number);`. When using the `perspective` function on the `transform` property, each element will have its own vanishing point—use the `perspective` property on the parent element of the 3d objects so they line up correctly.

The greater the size of the perspective, the less subtle the 3d effect and vice versa. You can adjust the vanishing point of an object with the `perspective-origin` property which is in the center by default.

- `translate3d(tx,ty,tz)` - x,y same as 2d, z is front and back
- `scale3d(sx,sy,sz)` – scale along corresponding axis
- `rotate(rx,ry,rz, angle)` – rx,ry,rz set the vector for which to rotate on, and angle is the intensity of the rotation. Example of rotating about the x-axis `rotate(1,0,0,45deg)`
- `matrix3d(n,n,n,n,n,n,n,n)` – translate according to the transformation matrix for 3d.

CSS3 TRANSITION

```
transition: <transition property> | <transition-duration> | <transition-
timing-function> | <transition-delay>
```

Provides a way to create an animation when changing certain properties of an element by making the changes to take place over a period of time. These are *implicit transition* because it involves an animation between two implicitly defined states.

- `transition-property: [none | all | IDENT]` – used to specify which properties to transition. Eg. `transition-property: background;`. List of transition-able properties: [link](#).

- `transition-duration: time`: How long transition should take place in second (s) or millisecond (ms). Default is 0. A list of times can be provided to provide a different duration for different properties specified in the `transition-property` property.
- `transition-timing-function: [ease | ease-in | ease-out | ease-in-out | linear | cubic-bezier(n,n,n,n) | step-start | step-end | steps(n, start | end)]`—lets you change how the speed of the animation is calculated over time.^{12 13} A list can be provided that will correspond to different properties specified in the `transition-property` property.
- `transition-delay: time` – amount of time to wait to before beginning transition of the changing property; default is 0s, which means it begins right away. Negative numbers begin transition right away but skip a portion of the transition in the beginning. A list can be provided that will correspond to different properties specified in the `transition-property` property

Detecting the Completion of a transition: event is fired called `transitioned`.

Eg. `el.addEventListener("transitioned",updateTransition, true);`

¹⁰ <http://desandro.github.com/3dtransforms/docs/perspective.html>

¹¹ <http://www.eleqtriq.com/2010/05/understanding-css-3d-transforms/>

¹² <https://developer.mozilla.org/en-US/docs/CSS/transition-timing-function>

¹³ <https://developer.mozilla.org/en-US/docs/CSS/timing-function>

IMPLEMENTING HTML5 APIS

HTML5 GEOLOCATION¹⁴

Allows user to provide their location to the page, but is prompted by the browser for privacy considerations.

API is published through geolocation child object within the navigator so you can test for support using:

```
If("geolocation" in navigator) { // available }  
If(navigator.geolocation) { // available }
```

- `getCurrentLocation(position_function[, error_function, options])` – initiates async request, when found `position_function` is executed. If an error is encountered, the `error_function` is executed.
- `watchPosition(position_function[, error_function, position_object])` – the `position_function` is executed multiple times either when the device changes location or a more accurate location is found using a different means (geoip vs. gps). The `error_function` is only called once if the position callback will never be run and no valid results ever returned.
 - `position_object` – { `enableHighAccuracy`: boolean, `maximumAge`: milliseconds(number), `timeout`: milliseconds(number) }
- `Position` Object is returned on success and contains the `timestamp` property, and the `coords` object.
 - `Timestamp` – `DOMTimeStamp` type indicates at the time the reading was taken.
 - `coords.latitude` – of type double, returns the lat of location in degrees
 - `coords.longitude` – of type double, returns the long of a location in degrees
 - `coords.altitude` – of type double, returns the altitude in meters; returns 0 if not supported by device
 - `coords.accuracy` – of type double, returns the position accuracy in meters
 - `coords.accuracyAltitude` – double, returns accuracy of altitude information in meters; 0 is returned if not supported.

- `coords.heading` – double, the heading in which the user is moving in degrees
- `coords.speed` – double, the speed in m/s of user

Handling Errors

The error callback structure: `function errorCallback(positionError) { ... }`

```
positionError.code = error.UNKNOWN_ERROR | error.PERMISSION_DENIED |  
                    error.POSITION_UNAVAILABLE | error.TIMEOUT
```

- `error.UNKNOWN_ERROR` (0) – unknown error occurred, could not get position.
- `error.PERMISSION_DENIED` (1) – permission unavailable at origin
- `error.POSITION_UNAVAILABLE` (2) – could not determine location b/c of device
- `error.TIMEOUT` (3) – callback timed out before could obtain position

HTML5 WEB STORAGE

Alternative to cookies, provides a larger, more secure, and easier-to-use alternatives to storing information on the client side. String/key value pairs are used.

- `sessionStorage.setItem(key,value)`, `sessionStorage.getItem(key,value)` – global object that maintains data for the duration of the session per page. Most useful for persisting temporary data just in case page is refreshed.
- `localStorage.setItem(key,value)`, `localStorage.getItem(key,value)` – no expiry date, and not lost when browser or sessions is closed.

¹⁴ https://developer.mozilla.org/en-US/docs/Using_geolocation

HTML5 APP CACHE^{15 16 17}

Allows web-based apps to run offline by specifying which resources need to be cached and make available offline. Used to browse offline, speed, and offloading server load.

- The manifest attribute must be set on the <html> tag: `<html manifest="example.appcache">`
- The manifest file needs to be sent with the MIME type `text/cache-manifest`

THE MANIFEST FILE

- Begins with line "CACHE MANIFEST" and has 3 sections: cache, network, fallback
- **CACHE:** The default. Files listed under "CACHE:" or immediately after "CACHE MANIFEST" are explicitly cached after being downloaded
- **NETWORK:** white-listed resources that need a connection—all requests to resources specified bypass the cache. Wild cards may be used.
- **FALLBACK:** specifies fallback pages that should be loaded if the resource is not available. Each entry lists two URIs: the resource and the fallback. Both URIs must be relative and from the same origin. Wildcards are allowed.

UPDATING THE CACHE

Cache is updated when: 1) user clears browser's cache, 2) manifest is modified, and 3) the application cache is programmatically update

Note: never cache the manifest

ESTABLISH THE SCOPE OF OBJECTS

JAVASCRIPT CLOSURES

¹⁵ https://developer.mozilla.org/en-US/docs/HTML/Using_the_application_cache

¹⁶ <http://www.html5rocks.com/en/tutorials/appcache/beginner/>

¹⁷ <http://alistapart.com/article/application-cache-is-a-douchebag>

In JS closures, the inner function of a function is returned. The inner function will close over all the inner variables of the outer function; new variables can be passed in, but the ones instantiated on the inside of the outer function cannot be affected.

In the example below, once the `var` in the global namespace is created (`var writeToResult = writeToId("result-block")`) the `elementId` is closed-over. Meaning, we don't have access to it anymore from the outside. The returned function can still modify the variable inside the function, however.

Example:

```
function writeToId(elementId) {
    var elementToEdit = document.getElementById(elementId);
    return function(innerHTMLInput)
    {
        elementToEdit.innerHTML += innerHTMLInput;
    }
}

var writeToResult = writeToId("result-block");
var writeToAside = writeToId("aside");

writeToResult("Hello! This is writing t the result block");
writeToResult("<br/>Since I already instantiated this variable, I don't have access to the elementId property anymore and I cannot change it. The only thing I can affect is what's exposted in the return function.");
writeToAside("This is an aside.");
```

NAMESPACES IN JAVASCRIPT

You can create name space like objects to hold the functions and objects that you create so as to not pollute the global namespace.

- Good way to implement: `var myNameSpace = myNamespaces || {};`
By having the previous declaration, you create a new "myNamespaces" object if one does not already exist.
- `var myNamespaces.myObjects = {};`

CREATE AND IMPLEMENT OBJECTS AND METHODS

JAVASCRIPT NATIVE OBJECTS

Data types: String, number, date, regex

Collection Types: Array, object

Array Sort

```
[array].sort(function(first,second)
{
  // A = 0 if first==second
  //A < 0 (negative) if right order (first < second)
  // A > 0 (positive) if wrong order (first > second)

  return A;
});
```

String :

- `String.length()` for how many characters in the string.
- `String.charAt(n)` returns at character at n
- `String.charCodeAt(n)` returns the unicode of character at n
- `String.fromCharCode(n1[, n2, n3...])` is a static method that returns the characters converted from Unicode.
- `.indexOf(string)`, `.lastIndexOf(string)` used for searching of the “string” occurrence in the string object it is run on. Returns the character position in that string. `lastIndexOf` starts from the back when searching.
- `String.substring(start,end)` - doesn't modify string it is run on, but returns a part of the string from the `start` index of the character to the `end` index provided. Note: the end character is NOT included, it is cut off.
- `String.substr(start,length)` – similar to above, but the second property is the length of characters to include in the returned text.
- `.toLowerCase()` and `.toUpperCase()` – returned a string with a change to the case of a string

Math Object

- `Math.abs(number)` – Absolute of a number
- `Math.floor(number)` – rounds down to the next smallest int
- `Math.ceil(number)` – rounds up to the next largest int

- `Math.pow(A,B)` – base to the exponent power (A^B)
- `Math.random()` – pseudo random number between 0 and 1

Number

- `[number].toFixed(int)` – returns. Rounds to the nearest `int`. Doesn't affect object

Array Object

- `[array].length` – how many elements in array. Returns int.
- `[array].concat(array)` – joins two arrays
- `[array].slice(first, last)` – copying part of an array. Last position is not included in the returned result. The original array is not affected.
- `[array].join(some_string)` – joins elements together, returns them as a string with `some_string` added in between each element.
- `[array].sort()` – puts array into ascending order. Note: this affects the actual array—it's not just a return function.

```
[array].sort(function(first,second)
{
  // A = 0 if first==second
  //A < 0 (negative) if right order (first < second)
  // A > 0 (positive) if wrong order (first > second)
  return A;
});
```
- `[array].reverse()` –puts in reverse order and affects the object it is run on.

Date

- No literal type
- Month is 0 based
 - `new Date(2010,10,25);`
 - `new Date()` – current
 - `new Date(year, month, day, hours, minutes, seconds, milliseconds)`
- `date.toUTCString()` – Coordinated universal time
- `[date].getDate()` – day of the month
- `[date].getDay()` – day of the week. Sunday as 0.
- `[date].getMonth()` – month fo the year. Jan is 0;
- `[date].getFullYear()` – year in 4 digit number

eval

- Interprets strings of js code

- Slow, insecure, unnecessary
- Do not use! :P

isNaN

- Check to see if is number

parseFloat

- Converts string to number
- `parseFloat("533");`

CLASSES IN JAVASCRIPT

Defining a Class

```
myExamples.simpleClass = function simpleClass(name,age) {
  this.name = name;
  this.age = age;

  // OR
  // this.setName(name)...
};
```

Getter\Setter Methods

```
myExamples.simpleClass.prototype.getName = function()
{
  return this.name;
};

myExamples.simpleClass.prototype.setName = function(name)
// OR myExamples.simpleClass.method('setName', function() {...});
{
  this.name = name;
};
```

Creating a new Object Based on the Class

```
var exampleSimpleClass = new this.simpleClass("RockStar",18);
```

- We want to cast references of similar classes

```
myExamples.Vehicle = function vehicle(model, type, totalWheels) {
  this.setModel(model);
  this.setType(type);
  this.setTotalWheels(totalWheels);
};

myExamples.Vehicle.prototype.setModel = function (model) {
  this.model = model;
};

myExamples.Vehicle.prototype.setType = function (type) {
  this.type = type;
};

myExamples.Vehicle.prototype.setTotalWheels = function (wheels) {
  this.totalWheels = wheels;
};

myExamples.Vehicle.prototype.getVehicle = function () {
  return this.model + " " + this.type + " with " + this.totalWheels
  + " wheels";
};
```

PSEUDO-CLASSICAL INHERITANCE IN JAVASCRIPT^{18 19}

¹⁸ <http://stackoverflow.com/questions/2107556/how-to-inherit-from-a-class-in-javascript>

¹⁹ <http://phrogz.net/JS/classes/OOPinJS2.html>

IMPLEMENT PROGRAM FLOW

[Note: this section is abridged because many of these topics are elementary]

HTML DOM EVENTS²⁰

Mouse Events

- `onclick` – triggered when an element is clicked
- `ondblclick` – triggered when an element is double clicked
- `onmousedown` –
- `onmousemove`
- `onmouseover`
- `onmouseout`
- `onmouseup` – triggered when the mouse button is released over an element

Keyboard Events

- `onkeydown` – triggered for any keys in all browsers (including CTRL, ALT, SHIFT)
- `onkeypress` – not fired for all keys in all browsers (may not fire for CTRL, ALT, SHIFT...)
- `onkeyup`

Frame/Object Events

- `onabort` – triggered when object is stopped from loading before completely loaded
- `onload` – triggered when a document, frameset, or `<object>` has been loaded
- `onresize` – triggered when a document view is resized.
- `onscroll` – triggered when a document view is scrolled
- `onunload` – triggered when a page has unloaded

²⁰ http://www.w3schools.com/jsref/dom_obj_event.asp

Form Events

- `onblur` – when form element loses focus
- `onchange` – when the content, selection, or the checked state have changed on a form element (`<input>`, `<select>` and `<textarea>`)
- `onfocus` – when element gets focus (`<label>`, `<input>`, `<select>`, `<textarea>`, and `<button>`)
- `onreset` – when form is reset
- `onsubmit` – when form is submitted

EVENT BUBBLING²¹

Event bubbling deals with situations where a single event may trigger two or more event handlers defined at different levels of the DOM hierarchy. The event bubbles up from the lowest level triggering events set at each one.

Example of event bubbling: <http://jsbin.com/ipahid/4>

```
var div4 = document.getElementById("div4");
var div3 = document.getElementById("div3");
var div2 = document.getElementById("div2");

div4.addEventListener('click', function () {
  this.style.background = '#FFFF00';
  alert("Div4 Event Triggered");
});

div3.addEventListener('click', function () {
  this.style.background = '#FFFF00';
  alert("Div3 Event Triggered");
});

div2.addEventListener('click', function () {
  this.style.background = '#FFFF00';
  alert("Div2 Event Triggered");
});

div1.addEventListener('click', function () {
```

²¹ <http://www.javascripter.net/faq/eventbubbling.htm>


```

    this.style.background = '#FFFF00';
    alert("Div1 Event Triggered");
});

```

TO DO: Attaching event handlers methods: `addEventListener`, `<element onclick|onmouseover... = "function", $.bind`^{22 23}

TO DO: creating and triggering events:²⁴

```

// create the event
var evt = document.createEvent('Event');
// define that the event name is `build`
evt.initEvent('build', true, true);

// elem is any element
elem.dispatchEvent(evt);

// later on.. binding to that event
// we'll bind to the document for the event delegation style.
document.addEventListener('build', function (e) {
    // e.target matches the elem from above
}, false);

```

IMPLEMENTING CALLBACKS

WEBSOCKETS API^{25 26 27}

WebSockets are well suited for low-latency web application because do not carry overhead of HTTP by establishing a “socket” between the browser and the server.

Example from HTML5 rocks:

²² <http://api.jquery.com/bind/>

²³ <https://developer.mozilla.org/en-US/docs/DOM/element.addEventListener>

²⁴ https://developer.mozilla.org/en-US/docs/DOM/Creating_and_triggering_events

²⁵ <http://www.websocket.org/echo.html>

²⁶ <http://www.html5rocks.com/en/tutorials/websockets/basics/>

²⁷ https://developer.mozilla.org/en-US/docs/WebSockets/Writing_WebSocket_client_applications

```

var connection = new
WebSocket('ws://html5rocks.websocket.org/echo', ['soap',
'xmpp']);

```

```

// When the connection is open, send some data to the
server
connection.onopen = function () {
    connection.send('Ping'); // Send the message 'Ping' to
the server
};

// Log errors
connection.onerror = function (error) {
    console.log('WebSocket Error ' + error);
};

// Log messages from the server
connection.onmessage = function (e) {
    console.log('Server: ' + e.data);
};

```

TO DO XMLHttpRequest

<http://www.w3.org/TR/CSS21/cascade.html#cascading-order>

<http://www.html5rocks.com/en/tutorials/flexbox/quick/>

CSS SELECTORS^{28 29}

.class

Selector	Example	Description
<code>.class</code>	<code>.rock</code>	Selects all the elements with the class "rock"
<code>#id</code>	<code>#star</code>	Selects the div with id "star"
<code>*</code>	<code>*</code>	Selects all the elements
<code>element</code>	<code>div</code>	Selects all the <div> elements
<code>element,element</code>	<code>div,span</code>	Select all <div> and all elements
<code>element element</code>	<code>div span</code>	Select all elements inside div.
<code>element>element</code>	<code>div>span</code>	Selects all elements where the parent is a div
<code>element+element</code>	<code>div+span</code>	Selects all elements placed immediately after <div> elements
<code>[attribute]</code>	<code>id</code>	Selects all elements with an id attribute
<code>[attribute=value]</code>	<code>name=rock-star</code>	Selects all elements where the

²⁸ http://www.w3schools.com/cssref/css_selectors.asp

²⁹ <http://www.webdirections.org/blog/html5-selectors-api-its-like-a-swiss-army-knife-for-the-dom/>

		name attribute is rock-star
<code>[attribute~=value]</code>	<code>name~=rock</code>	Selects all elements where the name attribute has the word rock in it
<code>[attribute =value]</code>	<code>[name =rock]</code>	Select all elements where the name attributes begins with rock
<code>:link</code>		
<code>:visited</code>		
<code>:active</code>		
<code>:hover</code>		
<code>:focus</code>		
<code>:first-letter</code>	<code>div:first-letter</code>	Selects first letter of every <div>

REFERENCES

Microsoft, Programming in HTML5 with JavaScript and CSS3:

<http://www.microsoft.com/learning/en/us/exam.aspx?ID=70-480#tab2>

W3Schools, HTML Reference:

<http://www.w3schools.com/tags/default.asp>

html5 Doctor, Semantic navigation with the nav element

<http://html5doctor.com/nav-element/>

html5 Doctor, Document Outlines

<http://html5doctor.com/outlines/>

Mozilla Developer Network, Section and Outlines of an HTML5 Document

https://developer.mozilla.org/en-US/docs/HTML/Sections_and_Outlines_of_an_HTML5_document?redirectlocale=en-US&redirectslug=Sections_and_Outlines_of_an_HTML5_document

Mozilla Developer Network, JavascriptGuide

<https://developer.mozilla.org/en-US/docs/JavaScript/Guide>

EXAM GUIDE-LINKS

<http://www.microsoft.com/learning/en/us/mcsd-web-apps-certification.aspx>

<http://blog.beckybertram.com/Lists/Exam%2070480%20Study%20Guide/AllItems.aspx>

<http://www.bloggedbychris.com/2012/09/19/microsoft-exam-70-480-study-guide/>

<http://geekswithblogs.net/WTFNext/archive/2012/10/08/exam-70-480-study-material-programming-in-html5-with-javascript-and.aspx>

http://www.microsoftvirtualacademy.com/tracks/developing-html5-apps-jump-start?WT.mc_id=MSLS_HTML5OfferMVA