**70-480 Exam Reference Notes**

**Chapter 1 – Implement and manipulate document structures and objects**

**Using HTML5 semantic markup**

|  |  |
| --- | --- |
| HTML5 Element | Description |
| *<article>* | Defines self-contained areas on a page |
| *<aside>* | Defines smaller content areas outside the flow of a webpage |
| *<figcaption>* | Defines the caption of a figure element |
| *<figure>* | Defines content that contains a figure, such as an image, chart, or picture |
| *<footer>* | Defines the bottom of a section or page |
| *<header>* | Defines the top of a section or page |
| *<hgroup>* | Defines a group of headings (H1–H6 elements) |
| *<mark>* | Defines text that should be highlighted |
| *<nav>* | Defines navigation to other pages in the site |
| *<progress>* | Defines the progress of the task |
| *<section>* | Defines the distinct content of a document |

**DOM Element Properties**

|  |  |
| --- | --- |
| Property | Description |
| childNodes | A collection of all child nodes of the parent element |
| firstChild | A reference to the very first child node in the list of nodes of the parent node. |
| lastChild | A reference to the very last child node in the list of the child nodes of the parent node |
| hasChildNodes | *True* if the parent element has any child nodes. *False* if it has none. Good to use before firstChild or lastChild. |

**JavaScript**

|  |  |
| --- | --- |
| Function | Action |
| Alert | Pop up dialog in browser, accesses the DOM |
| getElementById | Access a specific DOM element based on its CSS ID selector |
| getElementsByTagName | Access all elements of a specific type |
| getElementsByClassName | Get all elements of the same CSS class |
| removeChild | Takes an element and removes its first child |
| createElement | Makes a new element based on passed in parameter type |
| innerText | Sets the content for inside an element |
| appendChild | Adds a child element onto passed in element (after). Returns a reference to the new element appended to the child elements. |
| insertBefore | Parameters: new elements, node before which you want to append the new element. Inserts the new element before the node. |
| removeNode | Takes one Boolean parameter, if *True*, method does a deep removal (all children are also removed) |
| replaceNode, replaceChild | Same effect as removeNode & removeChild but they allow you to replace the target element with a new one.  Ex. Convert all <p> tags to <a> tags, use a for loop. |

Good to keep a unique ID reference for all of your DOM objects, especially if you remove an element and want to use it elsewhere later. This way you won’t lose references to your elements completely if the JavaScript variable goes out of scope.

**Implementing media controls**

<video> element

<video src=”thevideofile.mp4” autoplay> </video>

<video controls height="400" width="600" poster="picture.jpg">

<source src="samplevideo.ogv" type="video/ogg"/>

<source src="samplevideo.mp4" type="audio/mp4"/>

<object>

<p>Video is not supported by this browser.</p>

</object>

</video>

|  |  |
| --- | --- |
| Attribute | Description |
| Src | Specifies video to play (local or a URL). Not all browsers support the same video formats, MP4 is a good standard though. You can have multiple src’s in a video control tag to assist with this, and an object to notify the user if the browser doesn’t support the file. Browser goes through the source elements from top to bottom and plays the first one that it supports. Browsers that don’t support the video element will not show it at all. |
| Autoplay | Tells browser to start playing video as soon as it loads. If not included, video only plays when told to through player control or through JS. |
| Controls | Tells browser to include or hide built-in video controls |
| Height/width | Amount of space the video will occupy, without a definition the video will be played at its native size. |
| Loop | Tells browser if it should continuously play the video after it is finished playing the first time. |
| Poster | Specifies an image to show in the place for the video until the user starts to play the video. |

<audio> element

Essentially the same as the video element but for audio files. Both provide a standard way of displaying audio and video content for more straightforward markup.

**Implementing graphics with HTML5 <canvas> and SVG**

<canvas> provides a blank canvas area that can be drawn on dynamically.

No default visibility (like a div) on a blank HTML page.

Need to specify a size for the canvas.

Should have an onLoad JS event to cause graphics to render when the page is loaded.

Fixed (x, y) coordinate system, top left corner of the canvas is (0, 0). Width and height determine the x and y axis lengths.

Context is currently only 2d but 3d will come in the future.

|  |  |
| --- | --- |
| Method | Description |
| beginPath | Rests/begins a new drawing path |
| moveTo | Moves the context to the point set in the beginPath method |
| lineTo | Sets the destination end point for the line |
| Stroke | Strokes the line, which makes the line visible |
| lineWidth | Line thickness property |
| strokeStyle | Color of the line, HEX or string |

Methods for drawing curves

|  |  |
| --- | --- |
| Method | Description |
| Arc | A standard arc based on a string and ending angle and a defined radius |
| QuadradicCurveTo | More complex arc that allows you to control the steepness of the curve, has one control point. |
| bezierCurveTo | Another complex arc that you can skew, has two control points. |

Each drawing method has styles applicable to it (lineWidth, strokeStyle, lineCap).

**Using path methods**

Always need a start and the end point when using the context object to draw.

Ending point for one stroke can be the starting point of another stroke.

Use the *beginPath* method for this (creates a new line), then draw all your lines, then the *closePath* method to close a line.

**Using the rect method**

Draws rectangles.

|  |  |
| --- | --- |
| Parameter | Description |
| X, Y | Starting point of the rectangle (top left corner) |
| Width | Rectangle’s width |
| Height | Height of the rectangle |

Center a rectangle by dividing your X and Y (width) parameters by 2 for the starting position of the rectangle.

**Using the fill method**

Fills shapes with colours, gradients, or patterns.

Sett the *fillStyle* property to a colour and call the *fill* method (before calling the *stroke* method).

*fillRect* is best for filling rectangles.

Logic is completely handled by the browser.

Creating a gradient involves using a new *CanvasGradient* object. First call the *createLinearGradient* methodto get a *CanvasGradient* object. On the *CG* object you define the color stops that you want to blend to create the gradient effect. Then you assign your *CG* object to the *fillStyle* property of the context.

To repeat a pattern, use an *onLoad* call to ensure that the image is repeated only after it is loaded by the browser. *createPattern(img, “repeat”);*

**Drawing images**

Use the *drawImage* method of the context object to draw an image on a canvas.

Methods takes an Image object and X & Y coordinates to define where the image should be drawn. Default IMG size if it’s resolution, but it is resizable (just mathematically alter the X & Y dimensions in the parameter calls).

**Drawing text**

strokeText(“string here”, x width, y height) – called on context object.

Always do after windows loads (window.onLoad).

Change font with .font = “…”

Use strokeStyle = “colour” to change the colour of the text.

textAlign =”center” for aligning text in the center. *Note:* Need to pass in X/2, Y/2 to center text in the center of the canvas.

**Scalable Vector Graphics (SVG)**

XML-based language for creating 2D graphics.

<svg> element.

Don’t lose quality when a user zooms in or out because they are vectors.

Can be accessed via the DOM.

Have attributes, styles, and event handlers.

SVG element provides a container to render graphics in.

Renders inline with the page’s layout.

In some cases SVG graphics are simpler than using the *canvas* element.

Performance can be a concern with SVG.

SVG’s can be rendered from existing graphics (image href).