##W09 - The Window Object

Global object - variable created in global scope are properties of the global object
- In a browser environment, window is the global object - The browser window that contains a web page

■ The Browser Object Model (BOM)
- collection of properties and methods that contain information about the browser and computer screen
device - phones, tablets
- No official standard
- Every browser window, tab, pop-up, frame, and iframe has a window object.
- Because JavaScript runs in different environments, BOM is specific to the browser.
- If don't know the global object name, use this keyword:

```
// assign the variable global to the global object from
// within the global scope.
const global = this;
```

Going Global - variables created without const, let, var keywords can be accessed anywhere in program.

```
x = 6; // global variable created belonging to the window object
window.x === x; << true // generally do not use window object
```

```
if(x){ do something } ← throw ReferenceError if x hasn't been defined.
if(window.x){ do something } ← if x hasn't been defined, window.x will return false, code block will not be evaluated.
```

parseInt() and isNaN() are methods of the global object.

Dialogs - alert(), confirm(), prompt()
- alert(): pause execution, display message
- confirm(): stop execution, ok (true), cancel (false)

W09 - The Window Object

- prompt(): Stop execution, display message and input field. text gets returned as a string. cancel returns null

Browser Information - window object properties and methods providing information about the user's browser.

Which Browser? - navagator property returns reference to Navigator object.

Navigator object contains information about the browser being used.

- userAgent property returns information about the browser and operating system.

```
window.navigator.userAgent // returns browser and OS info
```

* can be modified by user to masquerade as a different browser. The return string can be cryptic.
* has been depricated from official specifications

Location, Location, Location - window.location property is an object containing information about the url of the current page (properties with information about fragments of the url)
Most properties are read/write, can be changed by assignment.

href - returns full url as string `window.location.href;`

protocol - returns string describing protocol (http, https, pop2, ftp) (includes colon) `window.location.protocol;`

host - returns string describing the domain and port number, omitted if default port 80 is used `window.location.host;`

hostname - returns string describing domain `window.location.hostname;`

port - returns string describing port number or empty string if the port is not explicitly stated in the url. `window.location.port;`

pathname - returns string of path following the domain

search - returns a string starting with '?' followed by query string parameters. Empty string if there aren't query string parameters

hash - returns string starting with '#' followed by fragment identifier empty string if there isn't a fragment identifier

origin - returns protocol and domain (read only property)

W09 - The Window Object

Window.location methods:

- reload() - force a reload of current page. true as a parameter will force reload from server instead of the cached page.
- assign() - load another resource from a URL
- replace() - almost same as assign() except current page will not be stored in history, making it unable to be navigated to through the back button.
- toString() - returns a string containing the whole url.

- Browser History - window.history used to access information about previously visited pages in current browser session. (not part of HTML5 History API)
  - window.history.length - number of pages visit prior to reaching current page.
  - window.history.go - used to specify a specific page relative to current page

```
window.history.go(1); // go forward one page
window.history.go(0); // reload currentPage
window.history.go(-1); // go back one page
```

  - window.history.forward() and window.history.back() are like the forward and back buttons

- Controlling Windows

window.open() takes URL of page as first parameter, windowTitle as the second and list of attributes as third

```
const popup = window.open('https://sitepoint.com', 'sitePoint',
'width=400,height=400,resizeable=yes');
```

window.close() will close window if have a reference to it.

```
popup.close();
```

window.moveTo() takes two parameters that are coordinates of the screen

window.resizeTo() takes two parameters to specify width and height

USE THESE METHODS RARELY - BE SENSIBLE IN DEPLOYMENT

W09 - The Window Object

- **Screen Information** - window.screen object contains information about the screen the browser is displayed on.

  window.screen.height
  window.screen.width
  window.screen.availWidth
  window.screen.availHeight
  window.screen.colorDepth

  more useful on phone and tablets.

- **Document Object** - methods and properties dealing with the loaded page

  document.write() - writes a string of text to the page, replacing the entire document

  - include HTML in string, it will become part of the DOM.
  - can be used within a document.

```
<h1>
  <script> document.write('Hello World!') </script>
</h1>
```

Cookies - small files saved locally on a user's computer.
- use for data storage beginning to be replaced with localStorage
- semicolon separated value pairs

"name=Superman; hero=true; city=Metropolis"

EU Cookie Directive - requires website based in EU to request permission to use cookies

Creating Cookies - assign to JavaScript's "cookie jar"

```
document.cookie = 'name=Superman';
```

assigning another cookie doesn't overwrite entire string

```
document.cookie = 'city=Metropolis';
```

changing cookie values - same name different value

```
document.cookie = 'name=Batman';
```

W09 - The Window Object

Reading cookies

document.cookies: // to see contents of the cookie

can split to break string into an array containing the name/value pairs

```
const cookies = document.cookie.split('; ');
for (crumb of cookies){
  const [key, value] = crumb.split("=");
  console.log(`The value of ${key} is ${value}`);
}
```

Expiry Dates - session cookies by default, deleted when browser session is ended.

- add "; expires=date" to end of cookie to set an expiry date ← UTC value

```
const expiryDate = new Date();
const tomorrow = expiryDate.getTime() + 1000 * 60 * 60 * 24;
expiryDate.setTime(tomorrow);
document.cookie = `name=Batman; expires=
                    ${expiryDate.toUTCString()}`;
```

- set max age value in seconds (86400 seconds = 1 Da

```
document.cookie = 'name=Batman; max-age=86400';
```

DO NOT RELY ON EXIRY DATES

Path and Domain - by default, cookies can only be read inside the same directory and domain as the file was set. can be changed so any page in root directory can read the cookie by adding '; path=/' to end of cookie or add domain "; domain=domainName"

```
document.cookie = 'name=Batman; path=/';
document.cookie = 'name=Batman; domain=sitepoint.com';
```

W09 - The WINDOW OBJECT

Secure Cookies

adding the string ';secure' to end of cookie will make it only transmit over a secure HTTPS network.

Deleting Cookies

Set expiry date to a time in the past

```
document.cookie = 'name=Batman; expires=Thu, 01 Jan 1970
                   00:00:01 GMT';
```

TIMING FUNCTIONS

window.setTimeout() - call back to a function as first parameter, number of milliseconds as second.

```
window.setTimeout( () => alert("Time's Up!"), 3000);
```

Method returns an integer, which is an id that references that particular Timeout

window.clearTimeout(); - to stop function based on Timeout reference.

window.setInterval() - similar to timeout but it will repeat the action every x milliseconds

```
function chant() { console.log('Mike is great!'); }
const repeat = window.setInterval( chant, 1000);
```

window.clearInterval() - like clear timeout

```
window.clearInterval(repeat);
```

W09- The Window Object

Animation - use setTimeout() and setInterval() to animate elements on a web page.

```
const squareElement = document.getElomentById('square');
let angle = 0;
SetInterval( () => {
    angle = (angle + 2) % 360; // resets to 0 at 360
    squareElement.style.transform = 'rotate(${angle}deg)'
}, 1000/60)
```

requestAnimationFrame works similar to setInterval() but has a number of improvements that improve performance.

Frame rate isn't set