Week 09 Reading

**Chapter 9, The Window Object**

Windows Object: Every JavaScript environment has a global object. Any variables that are created in the global scope are actually properties of this object, and any functions are methods of it. In a browser environment the global object is the window object, which represents the browser window that contains a web page.

The Browser Object Model (or BOM for short) is a collection of properties and methods that contain information about the browser and computer screen

*// from within the global scope*

*const* global = *this*;

**Dialogs**

window.alert('Hello');

<< *undefined*

Graphical user interface, website

Description automatically generated

Confirmation

window.confirm('Do you wish to continue?');

<< *undefined*

Input box

window.prompt('Please enter your name:');

**Browser Information**

The window object has a number of properties and methods that provide information about the user’s browser.

* window.navigator.userAgent
* << "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_12\_3) AppleWebKit/602.4.8 (KHTML, like Gecko) Version/10.0.3 Safari/602.4.8"

**The Browser History**

window.history.go(1); *// goes forward 1 page*

window.history.go(0); *// reloads the current page*

window.history.go(-1); *// goes back 1 page*

**Controlling Windows**

Opening new window

*const* popup = window.open('https://sitepoint.com','SitePoint','width=400,height=400,resizable=yes');

### Cookies

Cookies are small files that are saved locally on a user’s computer. They were invented by Netscape as a way of getting round HTTP being a stateless protocol. This means that a browser does not remember anything from one request to another. So every time a user visits a page, nothing about any previous visits is remembered. Cookies can be used to sidestep this problem by storing information that can then be retrieved between requests.

**Creating Cookies**

document.cookie = 'name=Superman';

<< "name=Superman"

**Secure Cookies**

Adding the string ; secure to the end of a cookie will ensure it’s only transmitted over a secure HTTPS network:

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

**Deleting Cookies**

To remove a cookie, you need to set it to expire at a time in the past:

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

**Timing Functions**

The window object provides some useful methods for scheduling the execution of a function, and for repeatedly executing functions at regular intervals.

The window.setTimeout() method accepts a callback to a function as its first parameter and a number of milliseconds as its second parameter. Try entering the following example into a console. It should show an alert dialog after three seconds (that’s 3000 milliseconds):

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

<< 4

**Chapter 14, HTML5 APIs**

HTML5 is the latest version of the Hypertext Markup Language used to create web pages.

**The data- Attribute**

Start with data-

data-powers = 'flight superSpeed'

data-rating = '5'

data-dropdown data-user = 'DAZ'

data-max-length = '32'

**HTML5 APIs**

The HTML5 specification contains a number of APIs that help to gain access to hardware, such as cameras, batteries, geolocation, and the graphics card

### HTML5 Web Storage

The Web Storage API provides a key-value store on the client’s computer that is similar to using cookies but has fewer restrictions, more storage capacity, and is generally easier to use.

Example:

localStorage.setItem('name', 'Walter White');

localStorage.getItem('name');

<< "Walter White"

Example:

localStorage.name = 'Heisenberg';

console.log(localStorage.name);

<< "Heisenberg";

To remove an entry from local storage, use the removeItem method:

localStorage.removeItem('name');

Alternatively, this can be done using the delete operator:

*delete* localStorage.name;

To completely remove everything stored in local storage, use the clear() method:

localStorage.clear();

**Geolocation**

The Geolocation API is used to obtain the geographical position of the device. This means it can be used to find the user’s exact location, then link to nearby places or measure the speed at which the user is moving

*function* youAreHere(position) {

console.log(`Latitude: ${position.coords.latitude}, Longitude: ${position.coords.longitude}`); }

Web worker

**A Factorizing Example**

Html

<!doctype html>

<html lang='en'>

<head>

<meta charset='utf-8'>

<title>Factorizor</title>

</head>

<body>

<button id='rainbow'>Change Color</button>

<form>

<label *for*='number'>Enter a Number to Factorize:</label> <input id='number' type='number' name='number' min=1 value='20'> <button type='submit'>Submit</button>

</form> <div id='output'>

</div>

<script src='main.js'></script>

</body>

</html>

main.js

*const* btn = document.getElementById('rainbow');

*const* rainbow = ['red','orange','yellow','green','blue','rebeccapurple','violet'];

*function* change() {

document.body.style.background = rainbow[Math.floor(7\* Math.random())];

}

btn.addEventListener('click', change);

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

**Notifications:**

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

**Multimedia**

<audio src='/song.mp3' controls>

Your browser does not support the audio element.

</audio>

<video src='http://movie.mp4' controls>

Your browser does not support the video element.

</video>

A screenshot of a child

Description automatically generated with low confidence

The play() method will start the clip playing from its current position:

video.play();

The pause() method will pause the clip at its current position:

video.pause();

The volume property is a number that can be used to set the audio volume:

video.volume = 0.9;

The muted property is a boolean value that can be used to mute the audio:

video.muted = true;

The currentTime property is a number value that can be used to jump to another part of the clip:

video.currentTime += 10; *// jumps forward 10 seconds*

The playbackRate property is used to fast-forward or rewind the clip by changing its value. A value of 1 is playback at normal speed:

video.playbackRate = 8; *// fast-forward at 8 times as fast*

The loop property is a boolean value that can be set to true to make the clip repeat in a loop:

video.loop = true;

The duration property can be used to see how long the clip lasts:

video.duration;<< 52.209