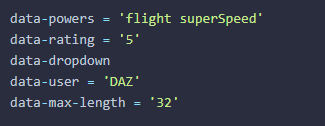
**HTML5**

HTML 5.1 has already become the latest standard, and HTML 5.2 is in development.

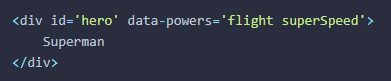
**The data- Attribute**

The data- attribute is a way of embedding data in a web page using custom attributes that are ignored by the browser.



The information contained in the attributes can be used to identify particular elements. For example, all the elements with an attribute of data-dropdown could be identified as dropdown menu.

Each element has a dataset property that can be used to access any data- attributes it contains. Here’s an example of some markup:



The **data-powers** attribute can be accessed using the following code:

Text

Description automatically generated

**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.

If a browser supports the Web Storage API, the window object will have a property called **localStorage**, which is a native object with a number of properties and methods used to store data.





Text

Description automatically generated

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



Alternatively, this can be done using the **delete** operator:



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



Every time a value is saved to local storage, a storage event is fired.

addEventListener('storage', (event) => {

console.log(`The ${event.key} was updated from ${event.oldValue} to ${event.newValue} and saved in

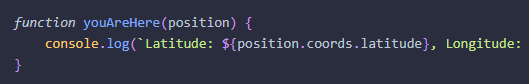
${event.storageArea}`) }, false);

**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.

If geolocation is available, it will be a property of the navigator object.



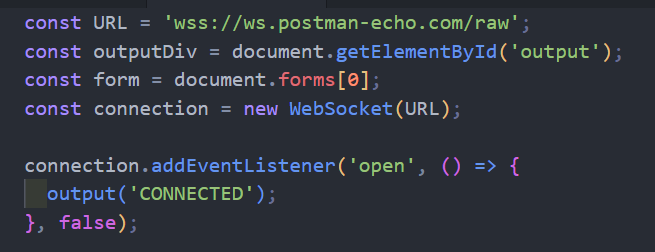


Text

Description automatically generated

**Websockets**

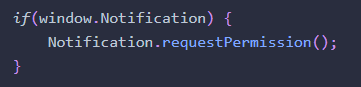
Websocket is a new protocol that allows two-way communication with a server – also known as push messaging. This means that a connection is kept open and responses are ‘**pushed’** to the client as soon as they are received.



**Notifications**

The Notification API allows you to show messages using the system's notifications. This is usually a popup in the corner of the screen, but it changes depending on the operating system.

Before you can send notifications, you need to get permission granted by the user. This can be achieved using the requestPermission( ) method of a Notification global object.



Text

Description automatically generated

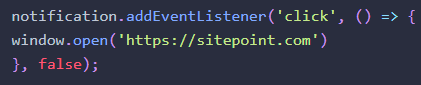
Text

Description automatically generated

Depending on your browser and operating system, some notifications close automatically after a short period of time, and some will stay on the screen until the user clicks on them. You can close the notification programmatically using the close() method:



For example, you could open a new window when the user clicked on the notification using the following code:



**Shims and Polyfills**

The terms shim and polyfill are often used interchangeably. The main difference between them is that a shim is a piece of code that adds some missing functionality to a browser, although the implementation method may differ slightly from the standard API. A polyfill is a shim that achieves the same functionality, while also using the API commands that would be used if the feature was supported natively.

This means that your code can use the APIs as normal and it should work as expected in older browsers.