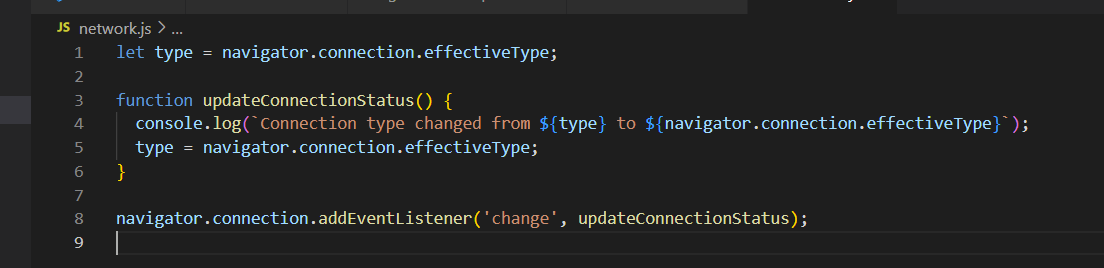
**Network Information API**

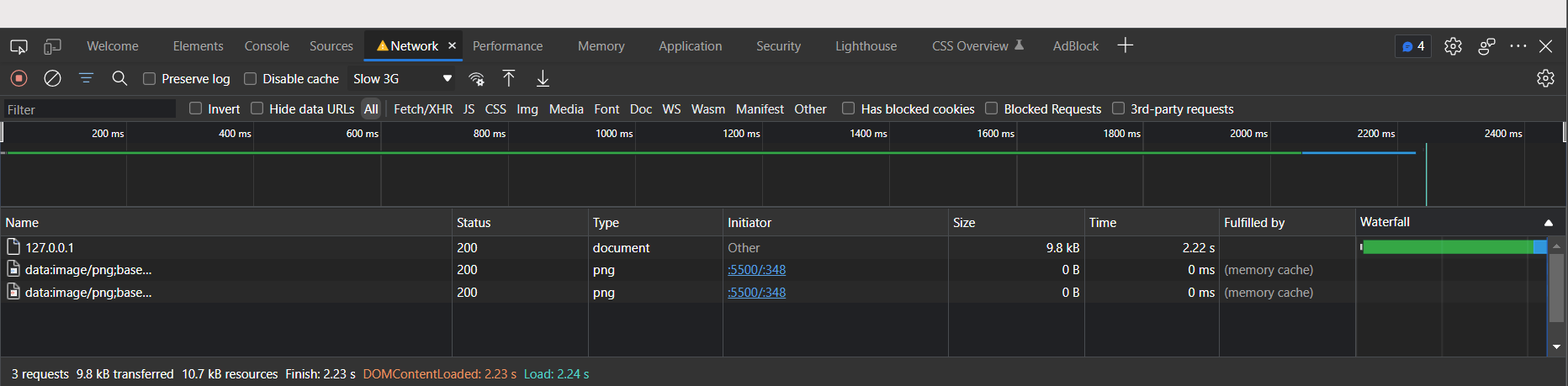
Knowing the connection state is essential with the emergence of mobile devices like phones and tablets because it might change at any time and impact the user experience. We also need to be mindful of various internet connections because their speeds vary greatly.

Thankfully, the Network Information API found in browsers allows us to check the status of an internet connection. Both worker contexts and browser contexts can use this API.

Detect Connection Changes

This example watches for changes to the user's connection.



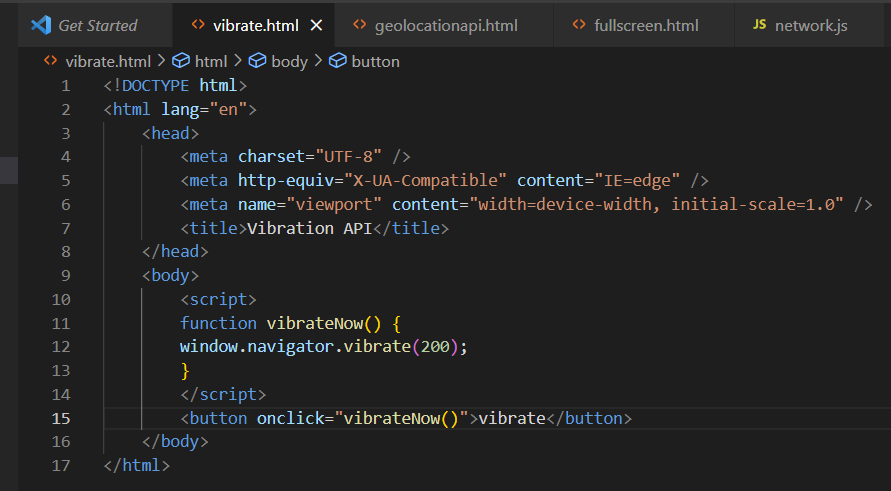


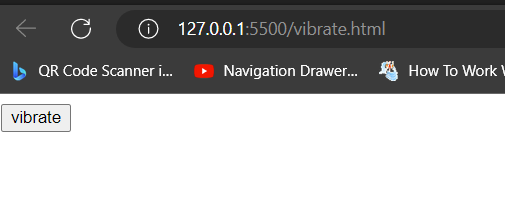
We can then test the connection type changes by going to the Chrome developer console by pressing F12. Then go to the Network tab, and then on the top right, there’s a dropdown to change the connection type.

**Vibration API**

The supported device's vibration mechanism is accessible through the API. The host device can be vibrated or a vibrational pattern can be carried out using this API. The vibrational motor can be accessed via this API, and if it doesn't exist, nothing happens. Except for Safari, all of the major web browsers support the API.

The vibration API provides a vibrate method that takes a single parameter. It can be an integer or an array of integers. If it is the former, the device will vibrate for milliseconds passed as the parameter. But if you have given an array of integer values, it will generate an alternating pattern in which the device is vibrating and not vibrating.





**Clipboard API**

The Clipboard API provides the ability to respond to clipboard commands (cut, copy, and paste) as well as to asynchronously read from and write to the system clipboard.

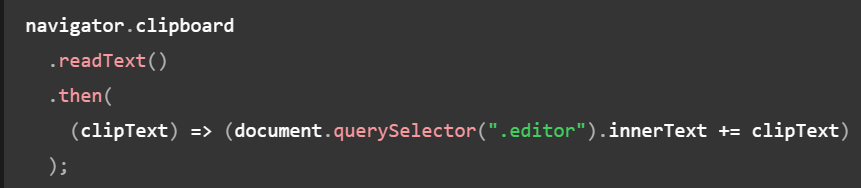
This API is designed to supersede accessing the clipboard using [document.execCommand ()](https://developer.mozilla.org/en-US/docs/Web/API/Document/execCommand).

The Clipboard API introduces two new objects:

Clipboard, which is accessible through navigator.clipboard and contains methods for reading from and writing to the system clipboard.

ClipboardItem, which represents a single item on the system clipboard that may have multiple representations. Currently, when reading and writing data, WebKit supports four MIME type representations: "text/plain", "text/html", "text/uri-list", and "image/png".

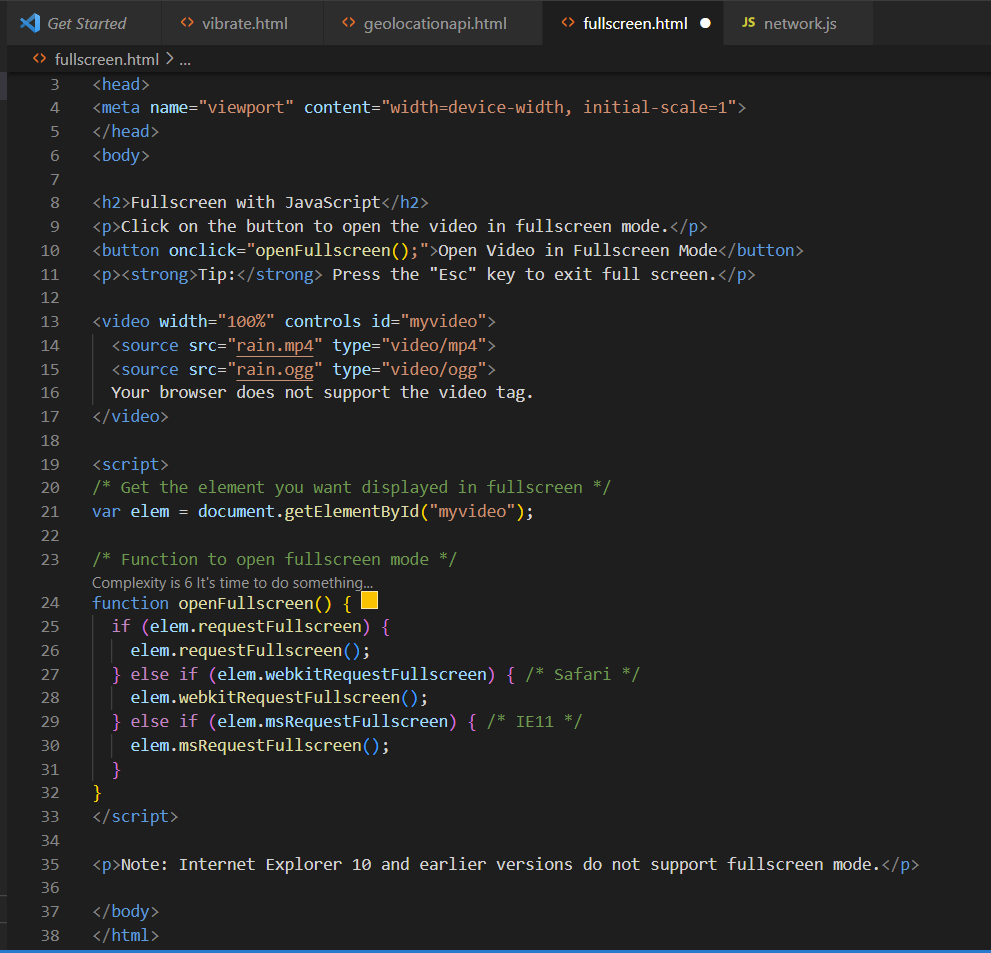
Instead of creating a Clipboard object through instantiation, you access the system clipboard through the [Navigator.clipboard](https://developer.mozilla.org/en-US/docs/Web/API/Navigator/clipboard) global:

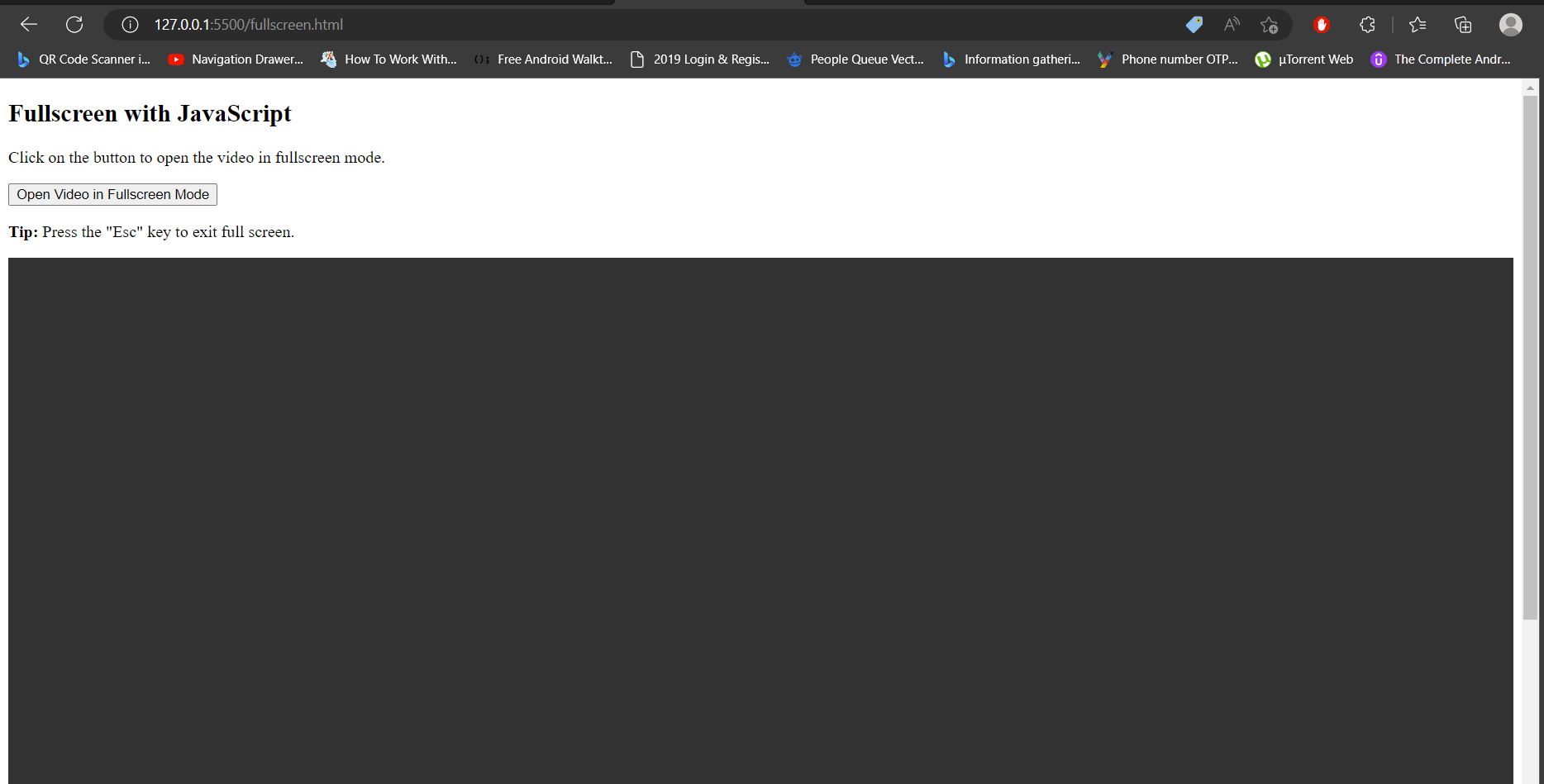


This snippet fetches the text from the clipboard and appends it to the first element found with the class editor. Since [readText()](https://developer.mozilla.org/en-US/docs/Web/API/Clipboard/readText" \o "readText()) (and [read()](https://developer.mozilla.org/en-US/docs/Web/API/Clipboard/read), for that matter) returns an empty string if the clipboard isn't text, this code is safe.

**FullScreen API**

The Fullscreen API adds methods to present a specific [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element) (and its descendants) in fullscreen mode, and to exit fullscreen mode once it is no longer needed. This makes it possible to present desired content—such as an online game—using the user's entire screen, removing all browser user interface elements and other applications from the screen until fullscreen mode is shut off.



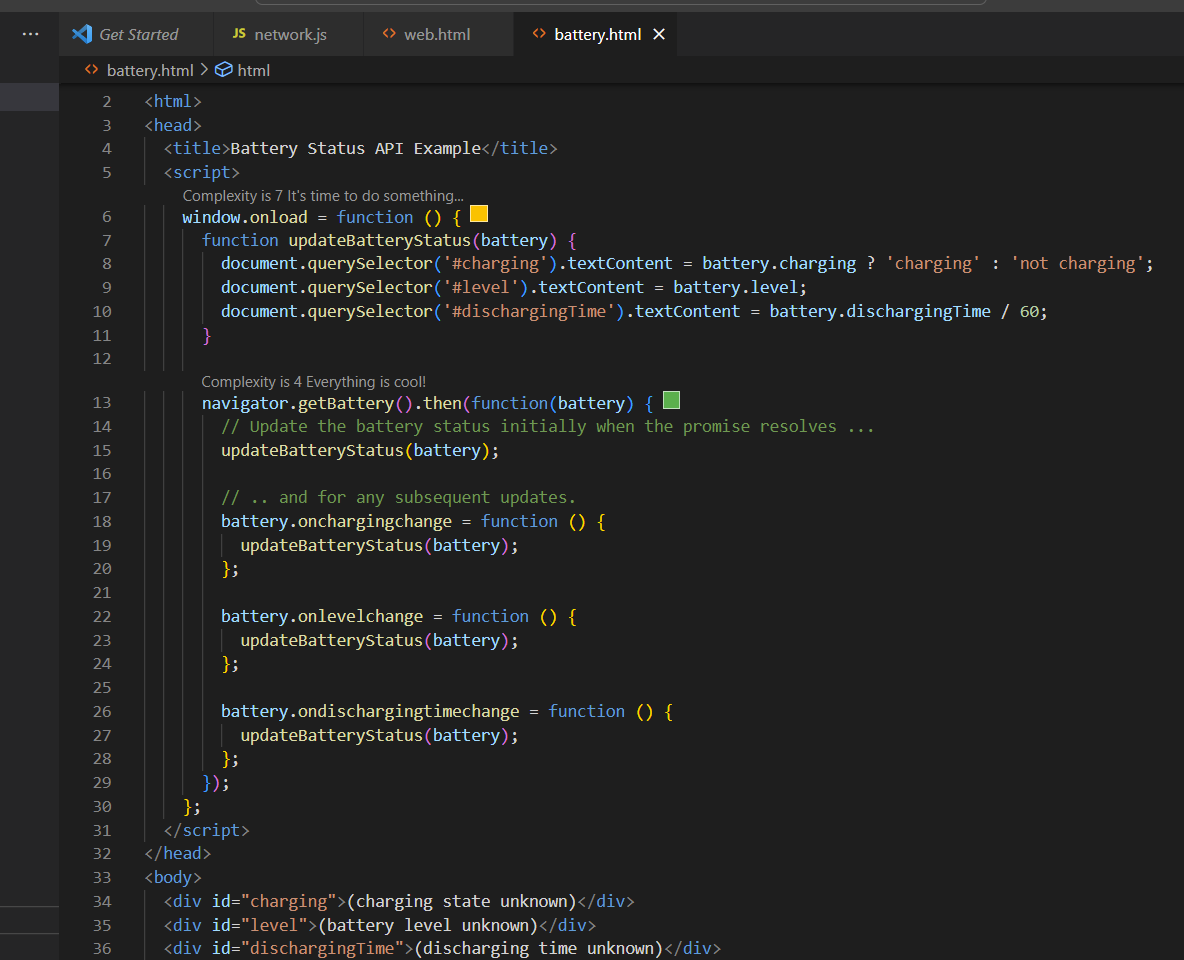


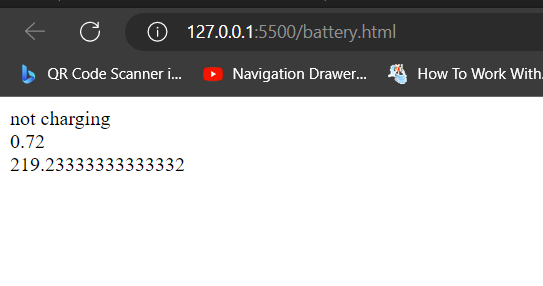
**Battery Status API**

The Battery Status API sometimes referred to as the Battery API provides information about the battery of the computer. It provides the remaining battery level and charging status of the battery. It also gives an estimate of the remaining time to completely charge the battery or discharge the battery if it is charging or discharging.

[**Example**](https://developer.mozilla.org/en-US/docs/Web/API/Battery_Status_API#example)

The following example updates the indicators to show the charging state, level and time remaining in minutes:





**Offline API**

The Offline Data Access API allows you to store data onto the device store persistently (data is made available even after application or device restart). You can access this data even when the device is offline, i.e. the device is not connected to any wireless network (cellular or WiFi).

The Local Storage API Library contains the kony.store namespace and related functions:

[kony.store.clear](https://docs.kony.com/konylibrary/visualizer/viz_api_dev_guide/content/kony.store_functions.htm)

This API allows you to empty the database by clearing all the key-value pairs. If there are no key-value pairs, then the API does not do anything.

**Syntax**

kony.store.clear();

**Input Parameters**

None

**Example**

try {

kony.store.clear();

alert("store is cleared");

} catch (err) {

alert("error occurred in clear() and the error is :" + err);

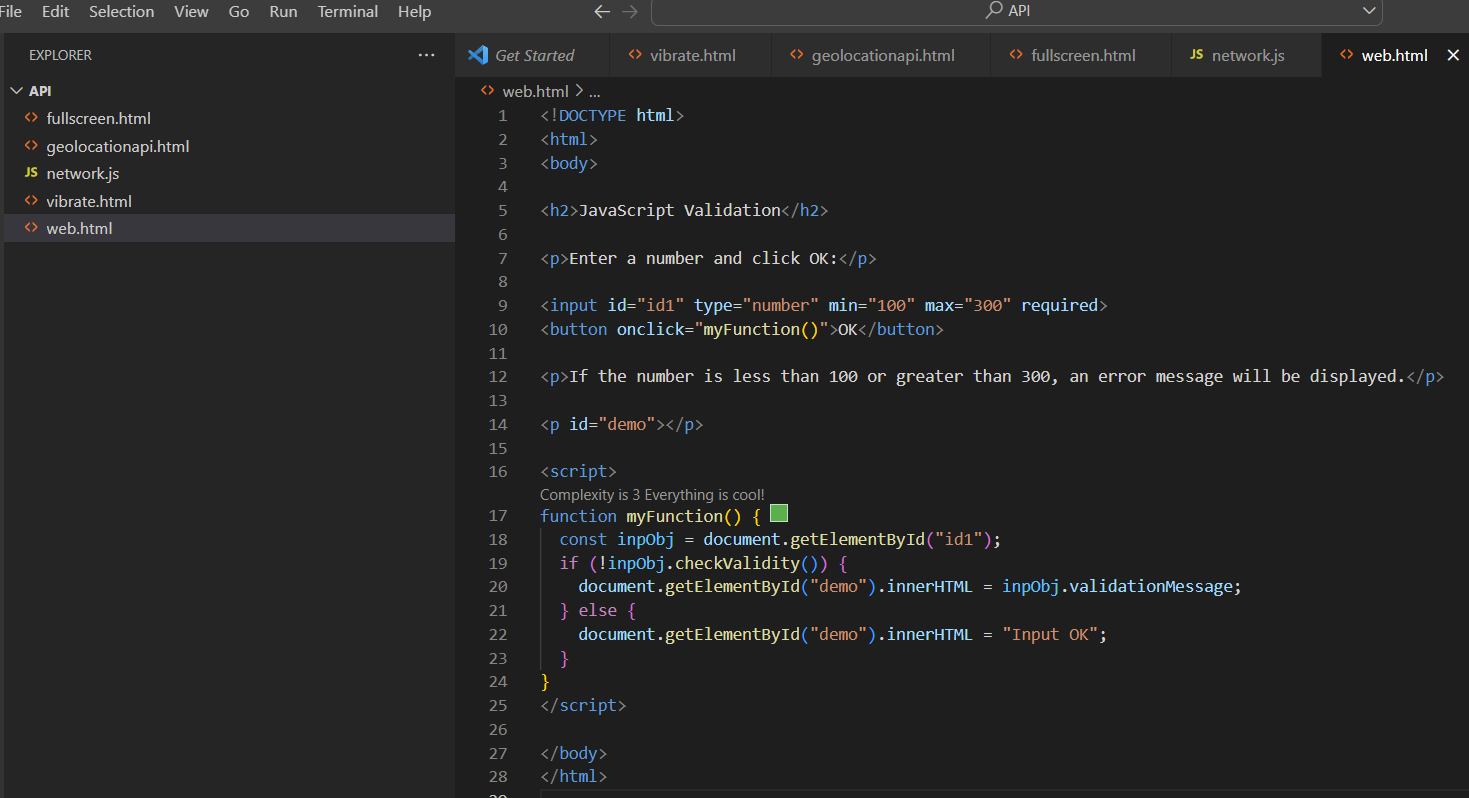
}

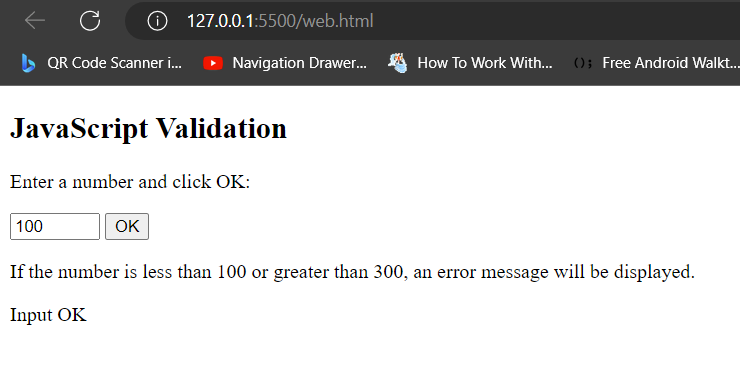
**Return value**

none

**Web API /Online API**

Web API is an API as the name suggests, it can be accessed over the web using the HTTP protocol. It is a framework that helps you to create and develop HTTP based RESTFUL services. The web API can be developed by using different technologies such as java, ASP.NET, etc. Web API is used in either a web server or a web browser. Basically Web API is a web development concept. It is limited to Web Application’s client-side and also it does not include a web server or web browser details. If an application is to be used on a distributed system and to provide services on different devices like laptops, mobiles, etc then web API services are used. Web API is the enhanced form of the web application.





**Geolocation API**

The Geolocation API is used to get the geographical position of a user.

Since this can compromise privacy, the position is not available unless the user approves it. The Geolocation API is supported in all browsers.

