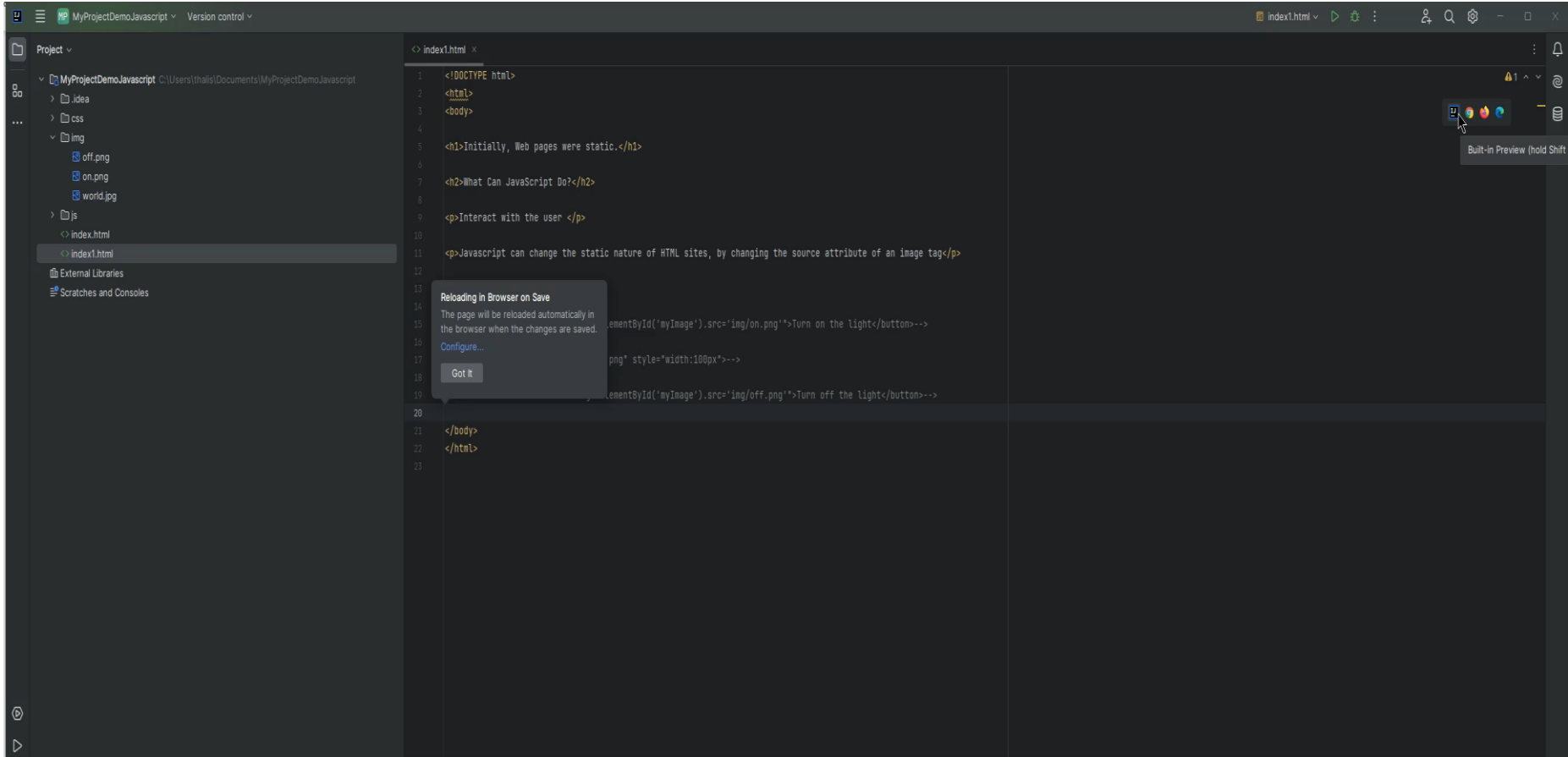


JavaScript

An introduction to JavaScript Programming

Why we need JavaScript?



The screenshot shows an IDE window with a project named 'MyProjectDemo.Javascript'. The file 'index.html' is open, displaying the following HTML code:

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>Initially, Web pages were static.</h1>
6
7 <h2>What Can JavaScript Do?</h2>
8
9 <p>Interact with the user </p>
10
11 <p>Javascript can change the static nature of HTML sites, by changing the source attribute of an image tag</p>
12
13
14
15 
16
17 
18
19
20
21 </body>
22 </html>
23
```

A tooltip titled 'Reloading in Browser on Save' is visible, stating: 'The page will be reloaded automatically in the browser when the changes are saved. Configure... Got it'.

Why we need JavaScript?

- JavaScript is a high-level language like Python and Java
- JavaScript, HTML and CSS are the primary languages used to build a website's front-end applications
- All browsers include a JavaScript engine to execute code
- JavaScript can even be used for server-side operations with `node.js`

Main attributes and properties

- JavaScript is a dynamically typed language
- Is a weakly typed language and prototype-based (objects)
- It is a multi-paradigm language, supporting functional and event-driven behavior
- It has an application programming interface (API), for handling text, arrays and HTML Document Object Model (DOM)
- JavaScript does not include input/output (I/O) module for networking, storage or any graphics interface

How to Add JavaScript into an HTML Document

Internal JavaScript

placed in head area of a document

```
<script>  
    // code block  
</script>
```

External JavaScript

async: scripts with the async attribute are executed asynchronously

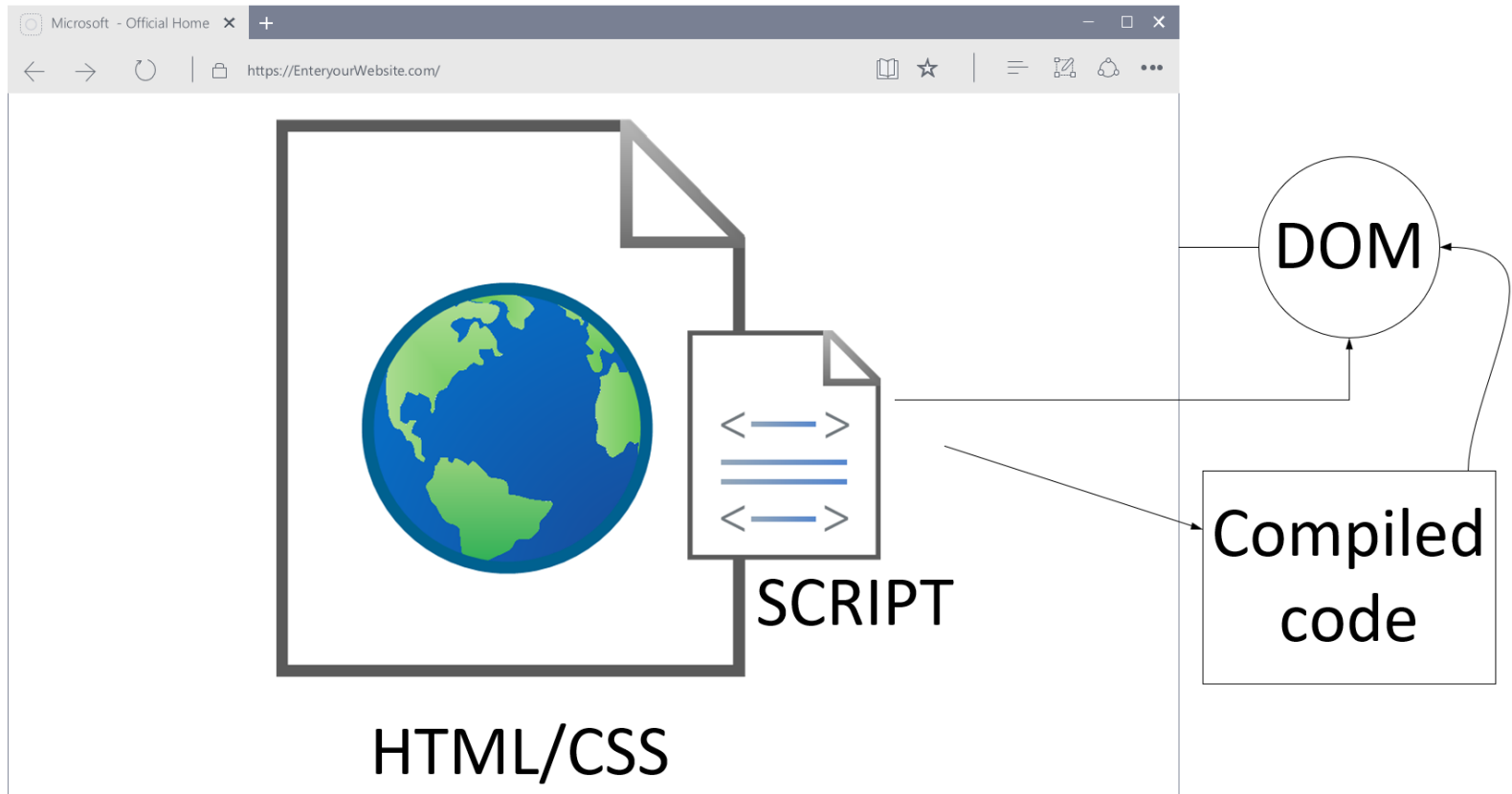
defer: the script is downloaded in parallel while parsing the page

```
<script src = "scriptfile.js"  
async>  
</script>
```

```
<button  
onclick="createParagraph()  
"CLick!!</button>
```

Inline JavaScript

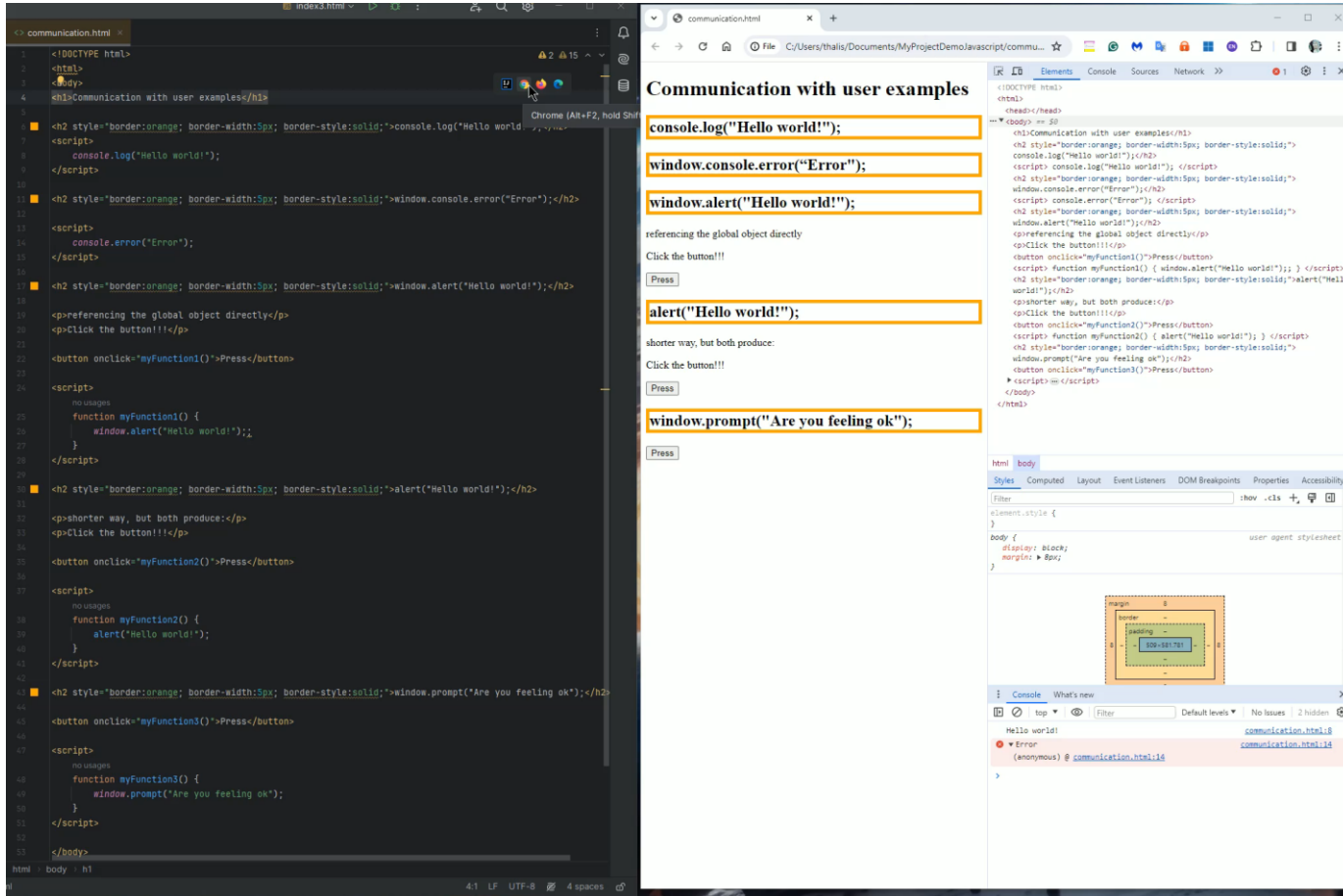
Just-in-time compiler (JIT)



JavaScript communication with user examples

- `console.log("Hello world!");`
- `window.console.error("Error");`
- `window.alert("Hello world!");`
`//referencing the global object directly`
- `alert("Hello world!");`
`//shorthand way`
- `window.prompt("Are you feeling ok");`

JavaScript communication with user examples



The image shows a code editor on the left and a browser window on the right. The code editor contains the following HTML and JavaScript code:

```

1 <!DOCTYPE html>
2 <html>
3 <body>
4 <h1>Communication with user examples</h1>
5
6 <h2 style="border:orange; border-width:5px; border-style:solid;">console.log("Hello world")
7 <script>
8   console.log("Hello world!");
9 </script>
10
11 <h2 style="border:orange; border-width:5px; border-style:solid;">window.console.error("Error");</h2>
12 <script>
13   console.error("Error");
14 </script>
15
16 <h2 style="border:orange; border-width:5px; border-style:solid;">window.alert("Hello world!");</h2>
17
18 <p>referencing the global object directly</p>
19 <p>Click the button!!</p>
20 <button onclick="myFunction1()">Press</button>
21
22 <script>
23   no images
24   function myFunction1() {
25     window.alert("Hello world!");
26   }
27 </script>
28
29 <h2 style="border:orange; border-width:5px; border-style:solid;">alert("Hello world!");</h2>
30
31 <p>shorter way, but both produce:</p>
32 <p>Click the button!!</p>
33 <button onclick="myFunction2()">Press</button>
34
35 <script>
36   no images
37   function myFunction2() {
38     alert("Hello world!");
39   }
40 </script>
41
42 <h2 style="border:orange; border-width:5px; border-style:solid;">window.prompt("Are you feeling ok");</h2>
43
44 <button onclick="myFunction3()">Press</button>
45
46 <script>
47   no images
48   function myFunction3() {
49     window.prompt("Are you feeling ok");
50   }
51 </script>
52
53 </body>
54 </html>

```

The browser window on the right shows the rendered page with the following text highlighted in yellow:

- console.log("Hello world!");
- window.console.error("Error!");
- window.alert("Hello world!");
- alert("Hello world!");
- window.prompt("Are you feeling ok!");

The browser's developer tools show the console with the following output:

```

Hello world!
Error
Hello world!
Are you feeling ok?

```

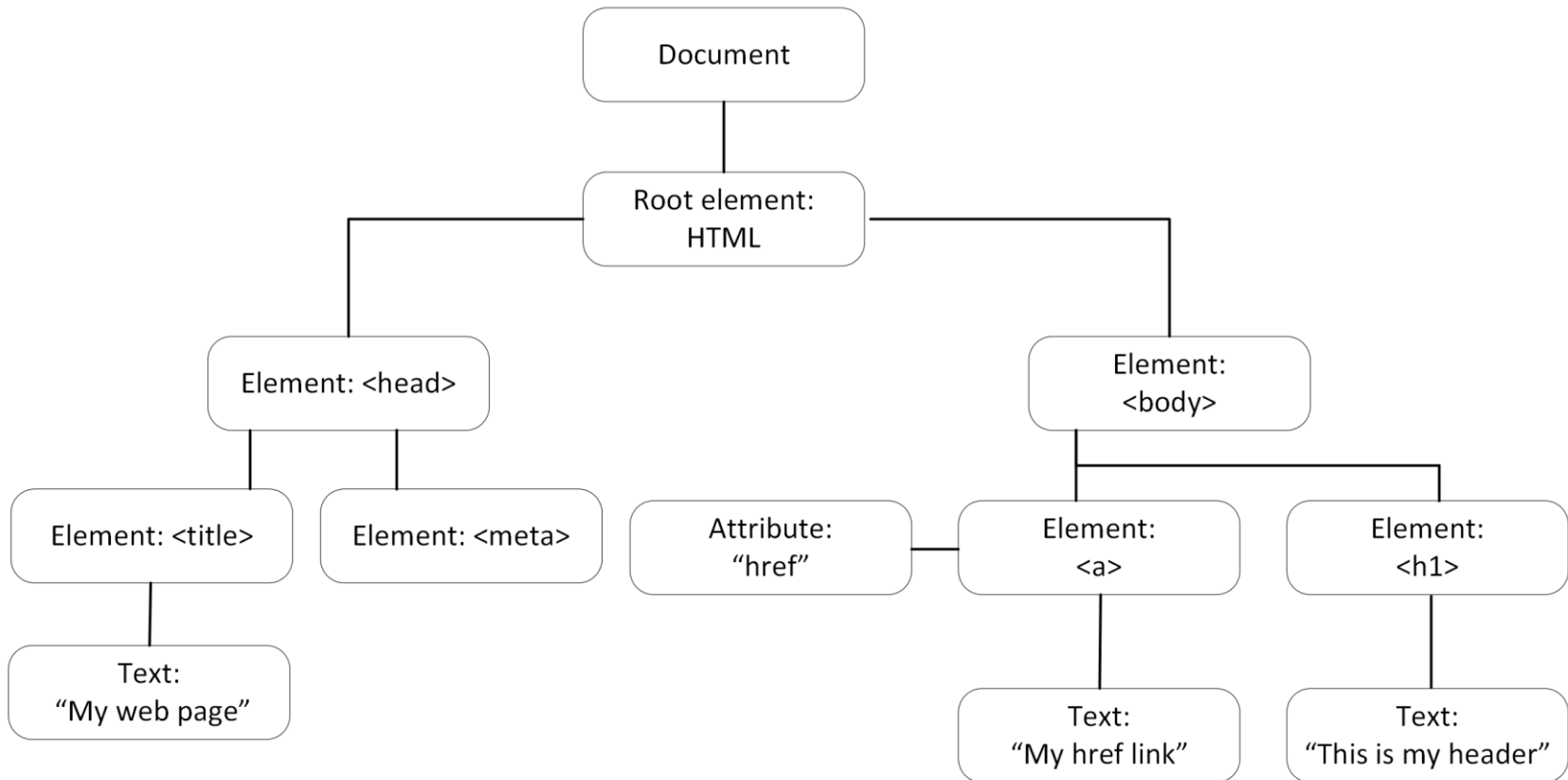

JavaScript engines

Every browser starts two processes for every web page

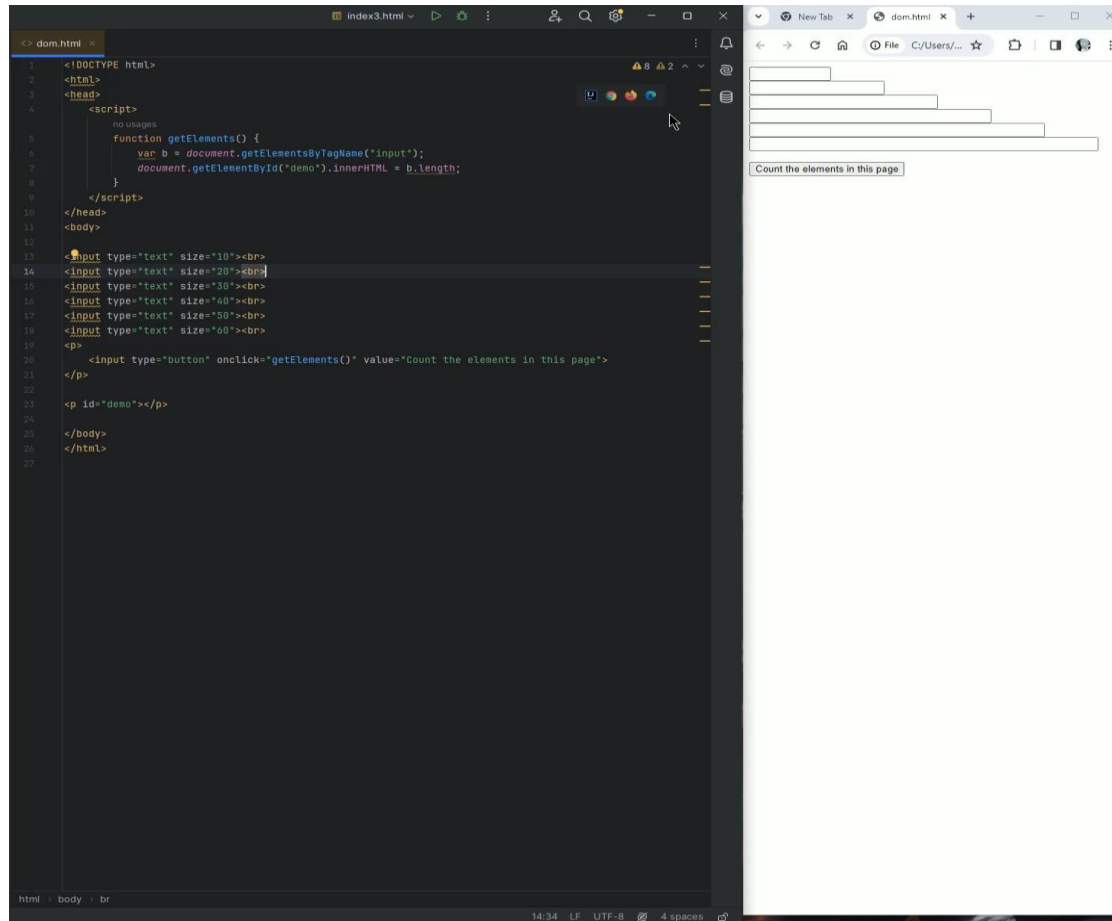
- **Rendering engine:** it is responsible to display the web page. Parses the HTML and CSS and displays the content on the screen
- **JavaScript engine:** This is where JavaScript code gets executed

JavaScript is interpreted, not a compiled language. Modern browsers use a technology called Just-In-Time (JIT) compilation that compiles the code to an executable bytecode.

Document Object Model (DOM) tree structure



Can you count the elements in your structure?

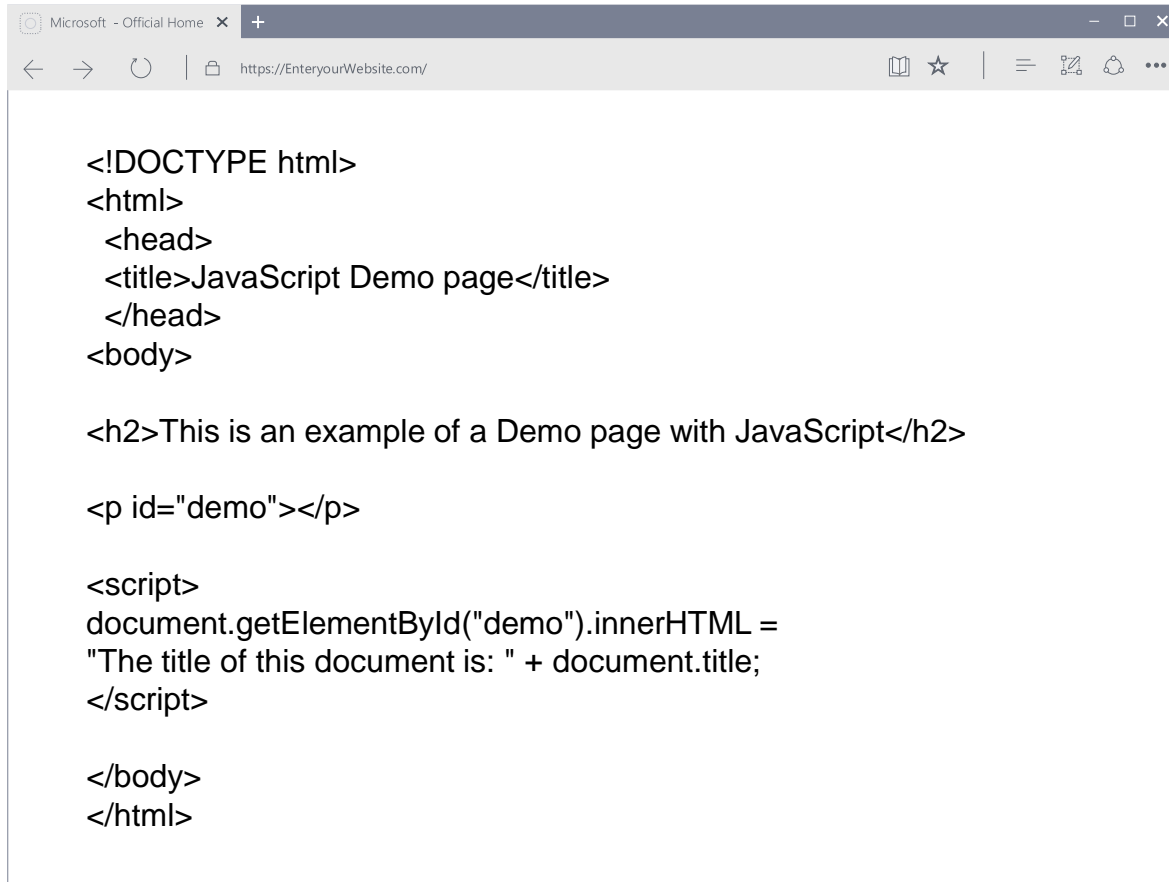


The image shows a code editor on the left and a browser window on the right. The code editor displays the following HTML and JavaScript code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <script>
5     no usage
6     function getElements() {
7       var b = document.getElementsByTagName("input");
8       document.getElementById("demo").innerHTML = b.length;
9     }
10  </script>
11 </head>
12 <body>
13   <input type="text" size="10"><br>
14   <input type="text" size="20"><br>
15   <input type="text" size="30"><br>
16   <input type="text" size="40"><br>
17   <input type="text" size="50"><br>
18   <input type="text" size="60"><br>
19   <p>
20     <input type="button" onClick="getElements()" value="Count the elements in this page">
21   </p>
22   <p id="demo"></p>
23 </body>
24 </html>
```

The browser window shows the rendered page with a button labeled "Count the elements in this page". The page content is currently empty, but the button is intended to trigger the JavaScript function to count the number of input elements and display the result in the `demo` paragraph.

DOM Manipulation in JavaScript



```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript Demo page</title>
  </head>
  <body>

  <h2>This is an example of a Demo page with JavaScript</h2>

  <p id="demo"></p>

  <script>
document.getElementById("demo").innerHTML =
"The title of this document is: " + document.title;
</script>

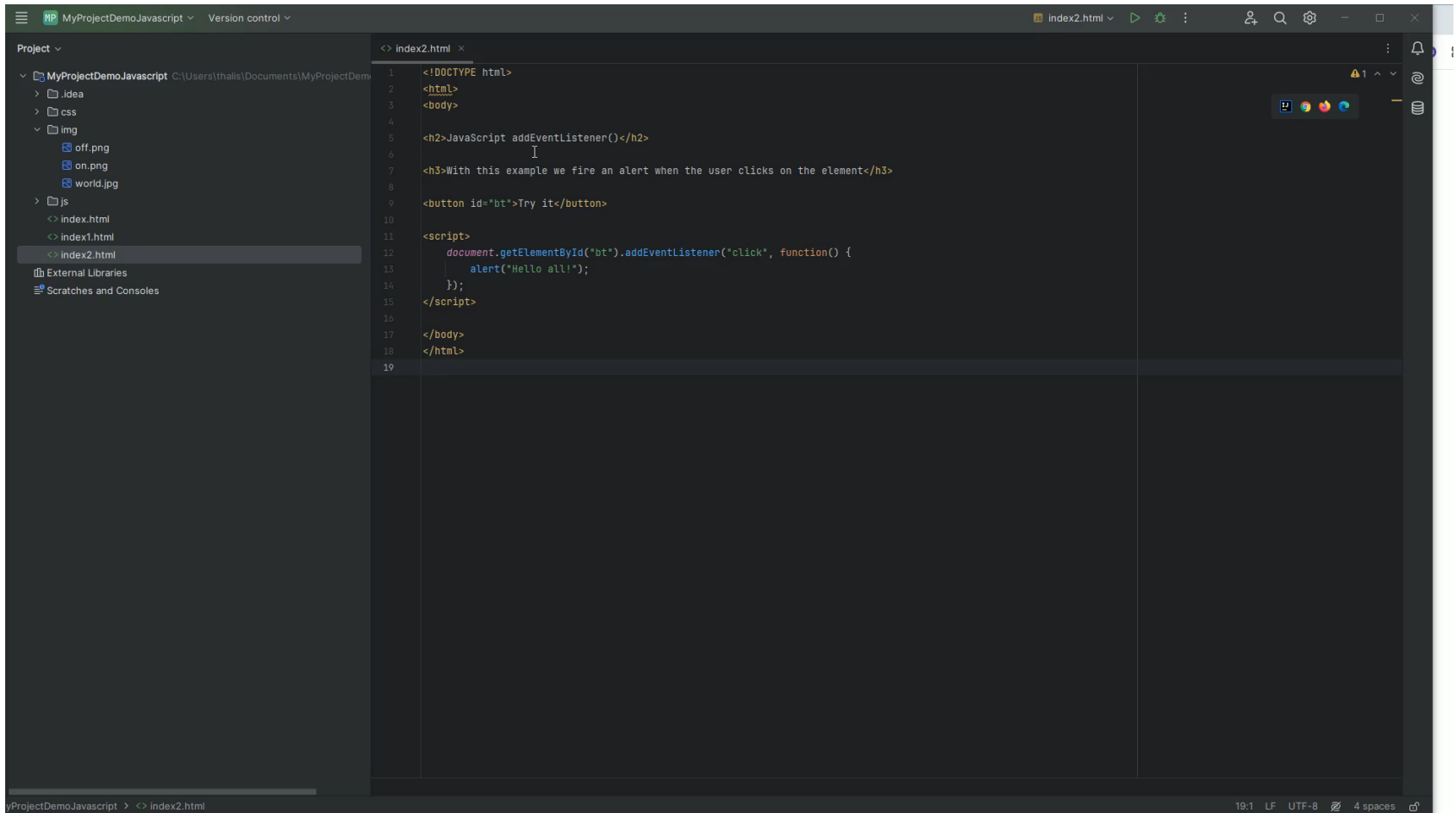
</body>
</html>
```

Navigator
(user agent)

Window
(tabs,
Window.innerWidth
Window.innerHeight)

Main document
(DOM, HTML)

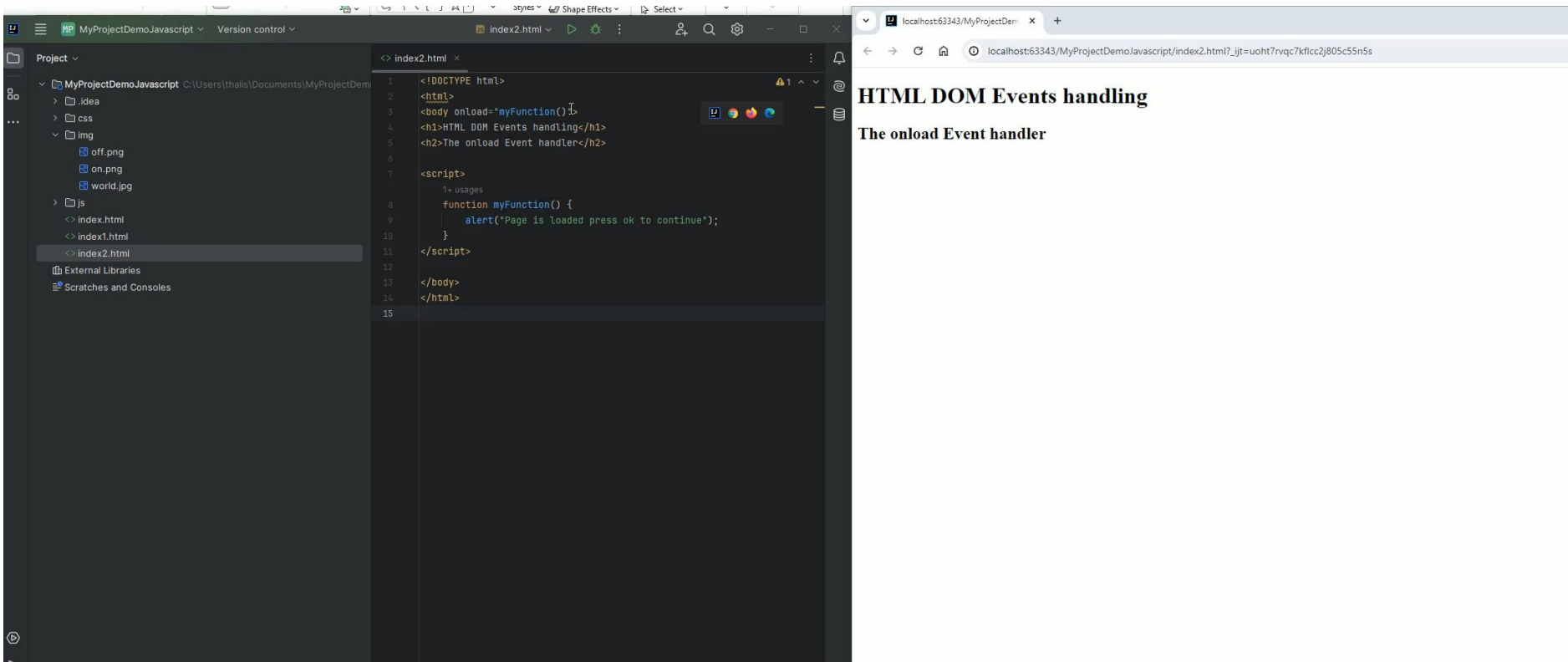
JavaScript HTML DOM EventListener



```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h2>JavaScript addEventListener()</h2>
6
7 <h3>With this example we fire an alert when the user clicks on the element</h3>
8
9 <button id="bt">Try it</button>
10
11 <script>
12     document.getElementById("bt").addEventListener("click", function() {
13         alert("Hello all!");
14     });
15 </script>
16
17 </body>
18 </html>
19
```

How to Make JavaScript Execute After HTML Load onload Event

If we use JavaScript to handle a Document Object Model (DOM) then our code executes after HTML (blocked code until OK!)

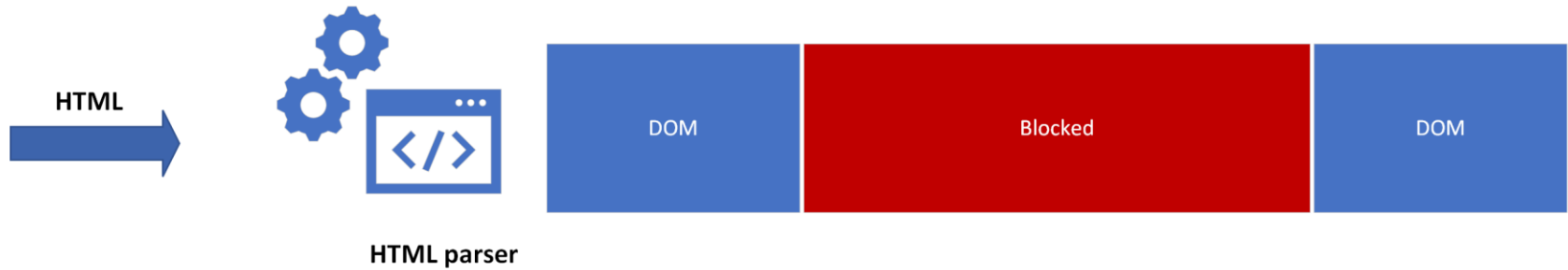


The screenshot shows a development environment with an IDE on the left and a browser on the right. The IDE displays the following HTML code in `index2.html`:

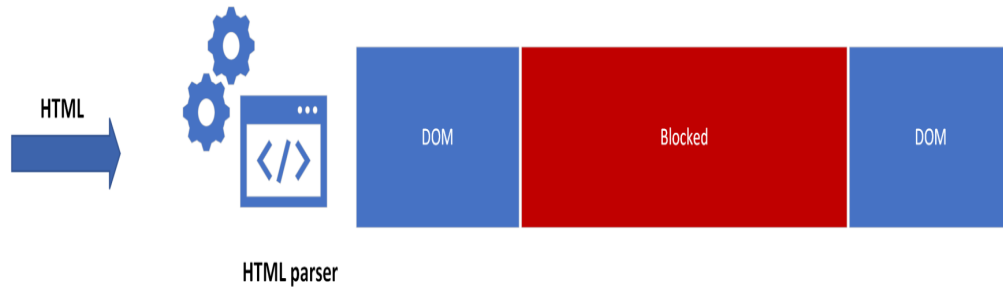
```
1 <!DOCTYPE html>
2 <html>
3 <body onload="myFunction()">
4 <h1>HTML DOM Events handling</h1>
5 <h2>The onload Event handler</h2>
6
7 <script>
8     1+ usages
9     function myFunction() {
10         alert("Page is loaded press ok to continue");
11     }
12 </script>
13 </body>
14 </html>
15
```

The browser on the right shows the rendered page with the title "HTML DOM Events handling" and the heading "The onload Event handler". The browser's address bar shows the URL: `localhost:63343/MyProjectDemoJavaScript/index2.html?_ijt=uoht7rvq7kflcc2j805c55n5s`.

Execute JavaScript: parser-blocking scripts (synchronous)



Execute JavaScript: parser-blocking scripts (synchronous)



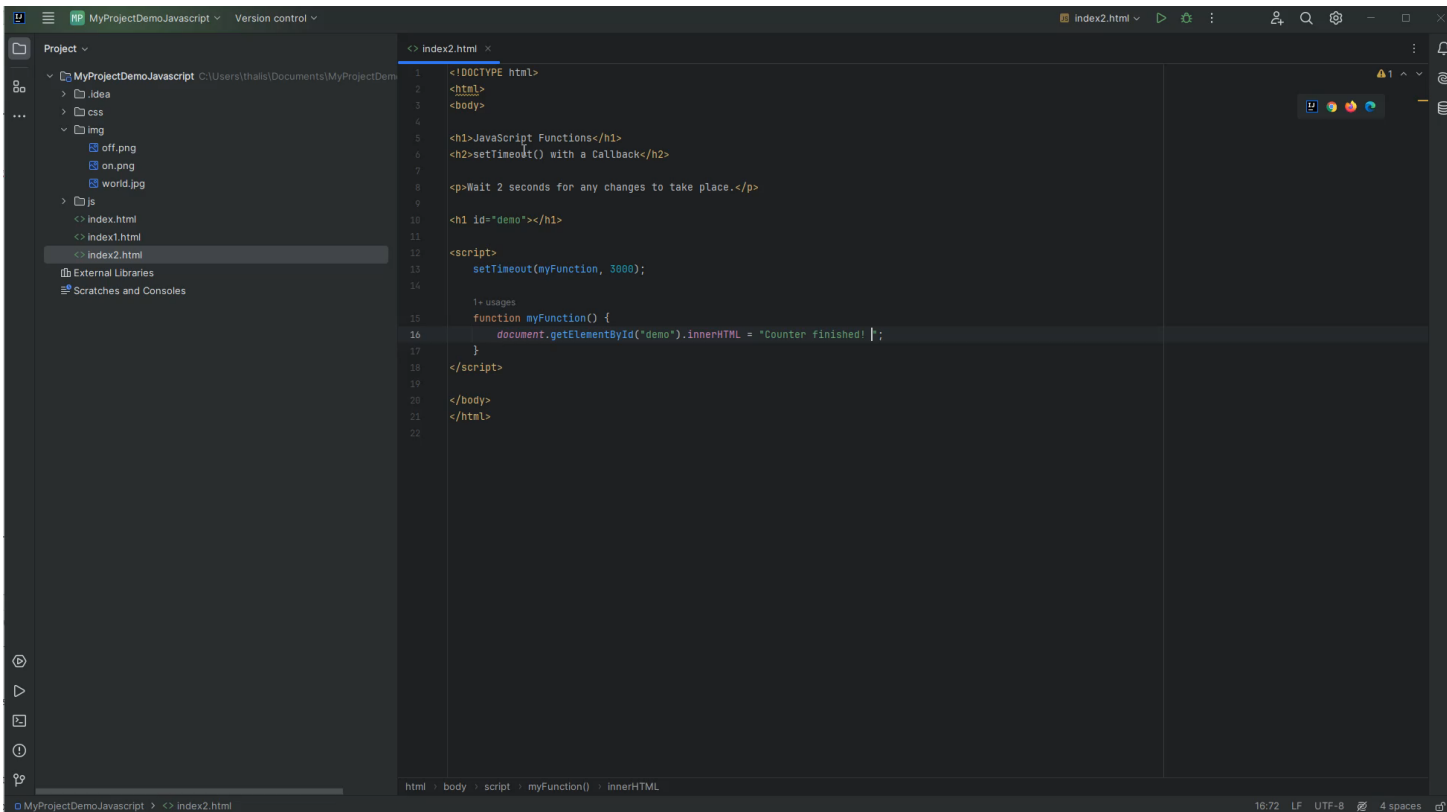
Asynchronous JavaScript

We can run function in parallel Asynchronously vs Synchronous coding:

is executed line by line (JavaScript has single thread execution)

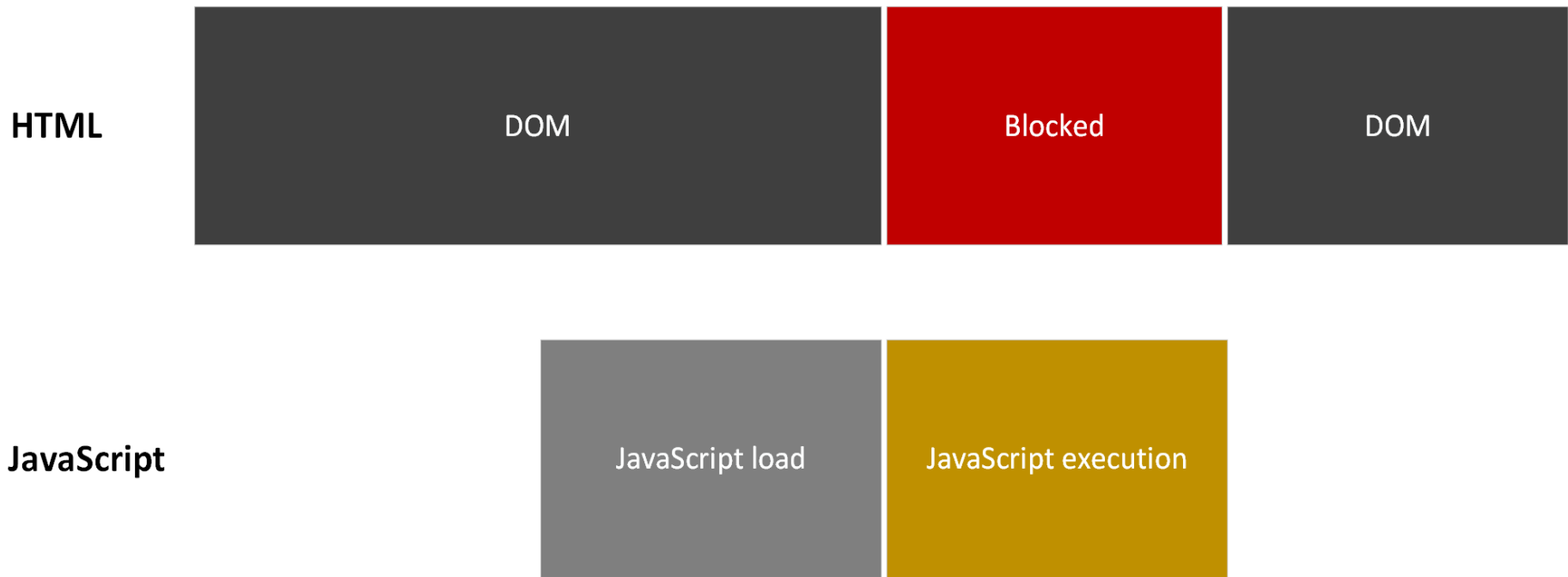
each line waits for previous line to finish

long time operation



```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>JavaScript Functions</h1>
6 <h2>setTimeout() with a Callback</h2>
7
8 <p>Wait 2 seconds for any changes to take place.</p>
9
10 <h1 id="demo"></h1>
11
12 <script>
13     setTimeout(myFunction, 3000);
14
15     1 ← usages
16     function myFunction() {
17         document.getElementById("demo").innerHTML = "Counter finished!";
18     }
19 </script>
20 </body>
21 </html>
22
```

Asynchronous JavaScript



Variable Names

- JavaScript variable names cannot include any Unicode character
- First character can be any of a-z, A-Z or (_) or \$

`let $4 = 1; → Block`

`let _4 = 10;`

`var $_$ = "money"; → Function`

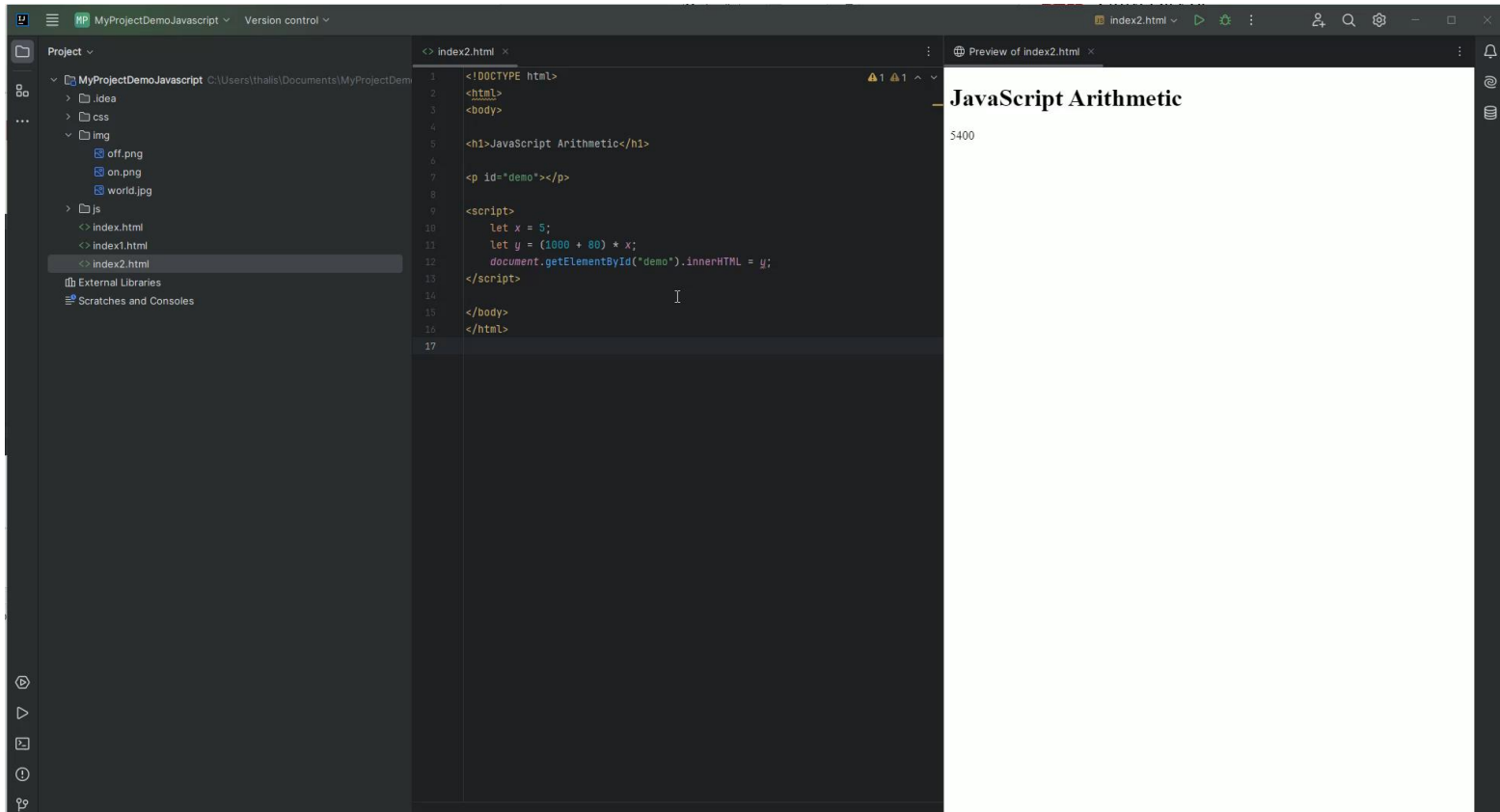
`var I_AM_HUNGRY = true;`

typeof Operator

We can use the **typeof** operator to identify the data type of a Javascript variable:

```
let x = 10;  
console.log(typeof x);  
>number
```

JavaScript Arithmetic Operators Demo



```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>JavaScript Arithmetic</h1>
6
7 <p id="demo"></p>
8
9 <script>
10   let x = 5;
11   let y = (1000 + 80) * x;
12   document.getElementById("demo").innerHTML = y;
13 </script>
14
15 </body>
16 </html>
17
```

JavaScript Arithmetic

5400

String in JavaScript

Declare:

```
let abc = 'hi  
all'; → The  
same
```

```
let def = "hi  
all";
```

\ → escape point

```
let "I\'m feeling  
lucky";
```

concatenate them using +

```
let one =  
"Hello";
```

```
let two =  
"world!!!";
```

```
let concatenation  
= one + two
```

String methods in JavaScript

```
let myName = 'Manolis'  
> myName.length;  
7  
> myName[0];  
'M'  
> myName[myName.length-1];  
's'  
> myName.slice(0,3);  
'Man'
```

```
> myName.indexOf('lis');  
4  
> myName.slice(3);  
'olis'  
> myName.toLowerCase();  
'manolis'  
> myName.toUpperCase();  
'MANOLIS'  
>  
myName.replace('lis','s');  
'Manos'
```

Arrays in JavaScript

Arrays are objects which can hold multiple values:

```
> let numbers = ['one', 'two', 'three'];
```

```
undefined
```

```
> numbers
```

```
[ 'one', 'two', 'three' ]
```

```
> let values = [1, 2, 3, 4, 5, 6, 10];
```

```
undefined
```

```
> let variety = ['one', 7896, [0, 1, 2]];
```

```
undefined
```

```
> variety
```

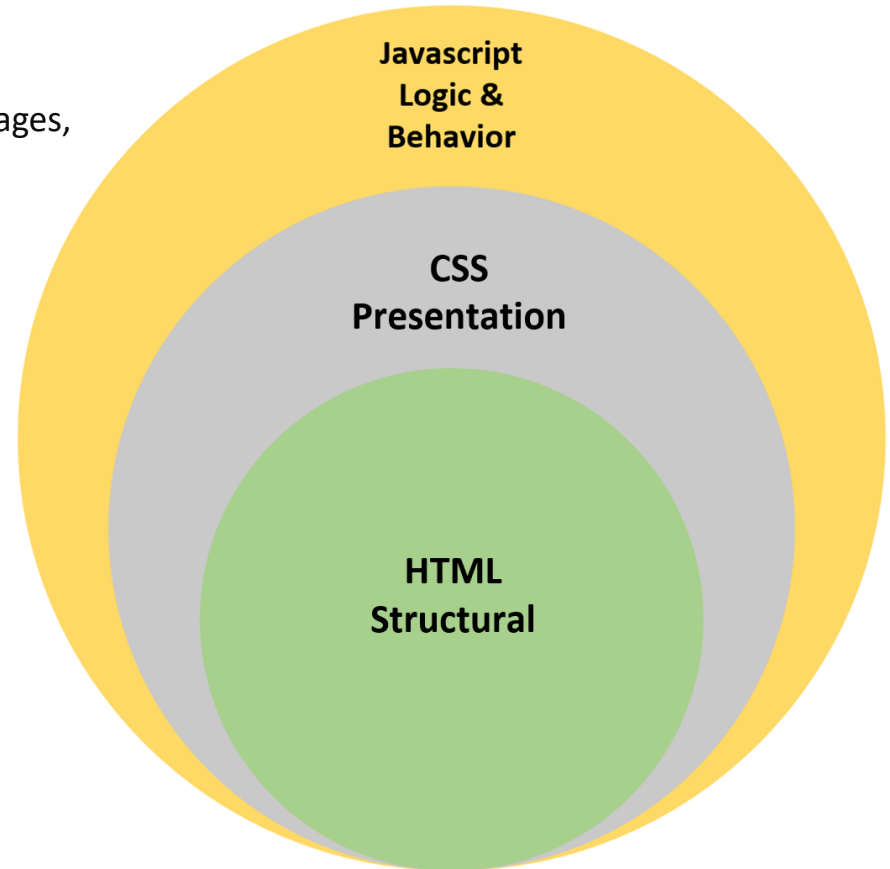
```
[ 'one', 7896, [ 0, 1, 2 ] ]
```


JavaScript and html

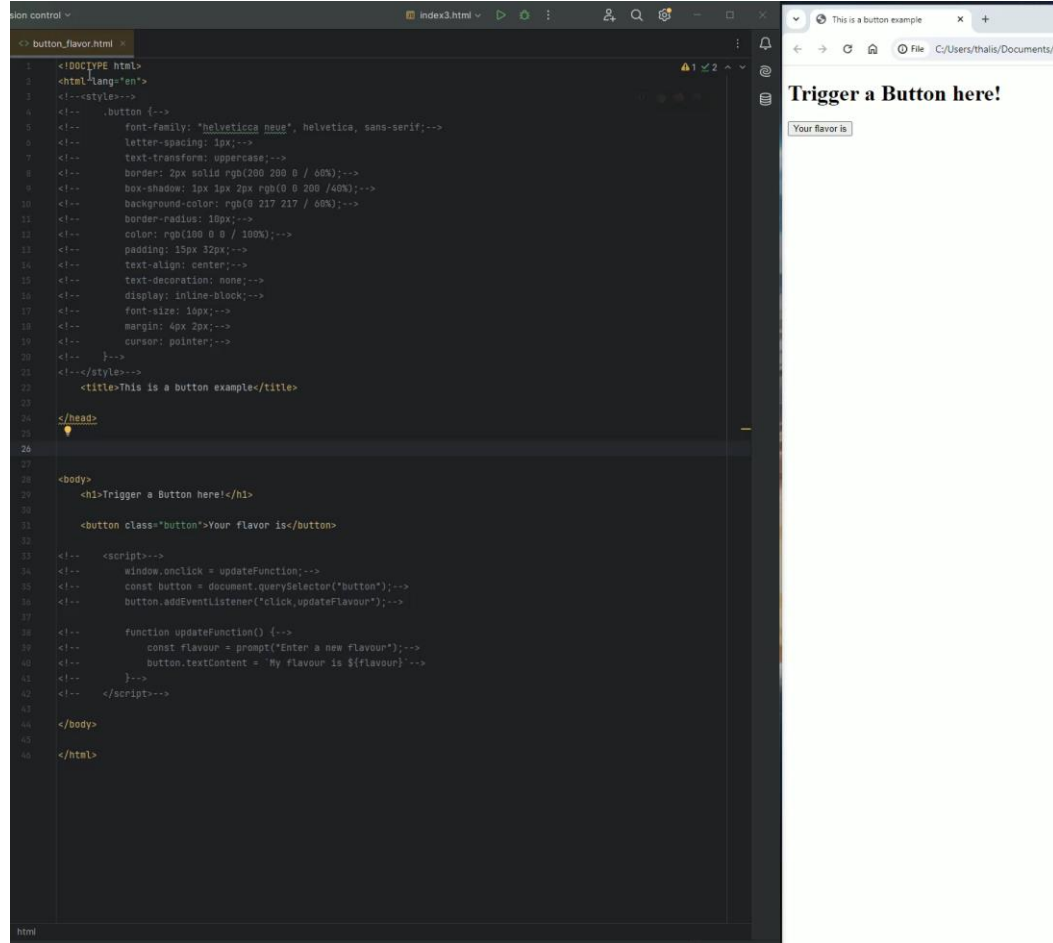
JavaScript is a scripting language utilized within an HTML document, to add functionality to the document or if you prefer to make a document dynamic.

With JavaScript, what we add complex features on web pages, specifically, we can:

- **Dynamically update html page content**
- **Control multimedia\images content of a page**
- **Control, change text**



Trigger a click event JavaScript– Button example HTML



The image shows a code editor on the left and a browser window on the right. The code editor displays the following HTML and JavaScript code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <!--style-->
4 <!-- .button {-->
5 <!--   font-family: "helvetica neue", helvetica, sans-serif;-->
6 <!--   letter-spacing: 1px;-->
7 <!--   text-transform: uppercase;-->
8 <!--   border: 2px solid rgb(200 200 0 / 40%);-->
9 <!--   box-shadow: 1px 1px 2px rgb(0 0 200 / 40%);-->
10 <!--   background-color: rgb(0 217 217 / 60%);-->
11 <!--   border-radius: 10px;-->
12 <!--   color: rgb(100 0 0 / 100%);-->
13 <!--   padding: 15px 32px;-->
14 <!--   text-align: center;-->
15 <!--   text-decoration: none;-->
16 <!--   display: inline-block;-->
17 <!--   font-size: 16px;-->
18 <!--   margin: 4px 2px;-->
19 <!--   cursor: pointer;-->
20 <!-- }-->
21 <!--/style-->
22 <title>This is a button example</title>
23
24 </head>
25
26
27
28 <body>
29 <h1>Trigger a Button here!</h1>
30
31 <button class="button">Your flavor is</button>
32
33 <!-- <script-->
34 <!--   window.onclick = updateFunction;-->
35 <!--   const button = document.querySelector("button");-->
36 <!--   button.addEventListener("click,updateFlavour");-->
37
38 <!--   function updateFunction() {-->
39 <!--     const flavour = prompt("Enter a new Flavour");-->
40 <!--     button.textContent = "My flavour is ${flavour}";-->
41 <!--   }-->
42 <!-- </script-->
43
44 </body>
45
46 </html>
```

The browser window on the right shows the rendered page with the heading "Trigger a Button here!" and a button labeled "Your flavor is".

JavaScript Functions

Every function contains a set of instructions in a logical sequence so that a specific result is always achieved. For function in JavaScript:

- Every function must have a specific and unique name (e.g., myFunction)
- Same rules for naming variables apply to function names
- Word "function" must precede the name of a function (e.g., function myFunction).
- Each function name is followed by a pair of parentheses (e.g., function myFunction()).
- The set of instructions for each function is contained within curly braces { and } (e.g., function myFunction() { . . . })
- The function parameters may be included within the parentheses, and there may be 0, 1, or more parameters
- Each function is called by its name (e.g., myFunction())
- Instructions within a function are executed when the function is called by its name
- Each function could be called multiple times within a script
- An HTML document may contain more than one function
- All the functions are contained within the <script> and </script> tags or wherever else code is inserted

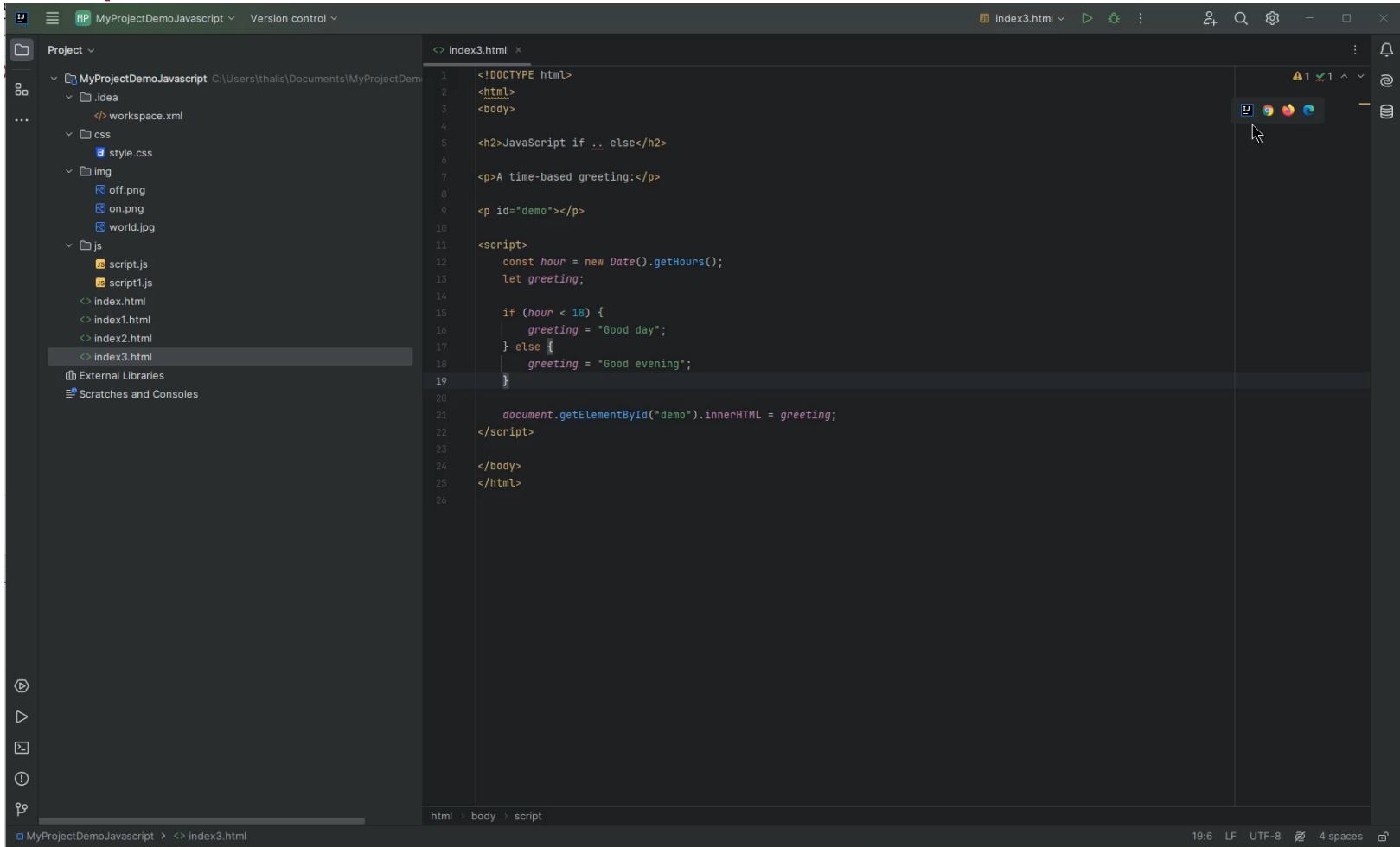
General syntax:

```
function FunctionName([param[, param[, ... param]]])  
{  
...  
}
```

Example:

```
<script>  
    function Hello() {  
        alert("Welcome!");  
    }  
</script>
```

JavaScript Conditions



```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h2>JavaScript if .. else</h2>
6
7 <p>A time-based greeting:</p>
8
9 <p id="demo"></p>
10
11 <script>
12   const hour = new Date().getHours();
13   let greeting;
14
15   if (hour < 18) {
16     greeting = "Good day";
17   } else {
18     greeting = "Good evening";
19   }
20
21   document.getElementById("demo").innerHTML = greeting;
22 </script>
23
24 </body>
25 </html>
26
```

JavaScript Loops

for (Initial counter value; Condition; Counting step) {
 // code block to be executed
}

Example:

```
JavaScript + No-Library (pure JS) ▼
1  var i;
2  for (i=0; i<5; i++)
3  {
4      document.write("Number is: " + i);
5      document.write("<br />");
6  }
```

Number is: 0
Number is: 1
Number is: 2
Number is: 3
Number is: 4

while (condition) {
 // code block to be executed
}

Example:

```
JavaScript + No-Library (pure JS) ▼
1  var i=0;
2  while (i<5)
3  {
4      document.write("Number is: " + i);
5      document.write("<br />");
6      i=i+1;
7  }
```

Number is: 0
Number is: 1
Number is: 2
Number is: 3
Number is: 4

JavaScript Cookies

Cookies are small text files that are stored on the computer and contain data in the form of name=value pairs. For example: visitor=vist123.

To write (or store) and read cookies on the machine, the command used is `document.cookie`.

`document.cookie = "cookieName=cookieValue";`

Overall, the cookie is stored as a string, and any additional parameters you can pass are separated by a question mark (`;`).

Example: `document.cookie = "userid=Fe80gRCCijyH4mgdO; expires=Sun, 13 Jun 2021 20:31:59 GMT; path=/"`

This is a JavaScript function to create a cookie:

Cookies store user/visitor information

When a browser requests a web page from the server, page cookies are added to that request



```
function setCookie(name, value, days) {  
    var expires = "";  
    if (days) {  
        var date = new Date();  
        date.setTime(date.getTime() + (days * 24 * 60 * 60 * 1000));  
        expires = "; expires=" + date.toUTCString();  
    }  
    document.cookie = name + "=" + value + expires + "; path=/";  
}
```



Privacy concerns

- Tracking
- Profiling
- Data Collection
- Third-Party vs First-Party Cookies

How to protect

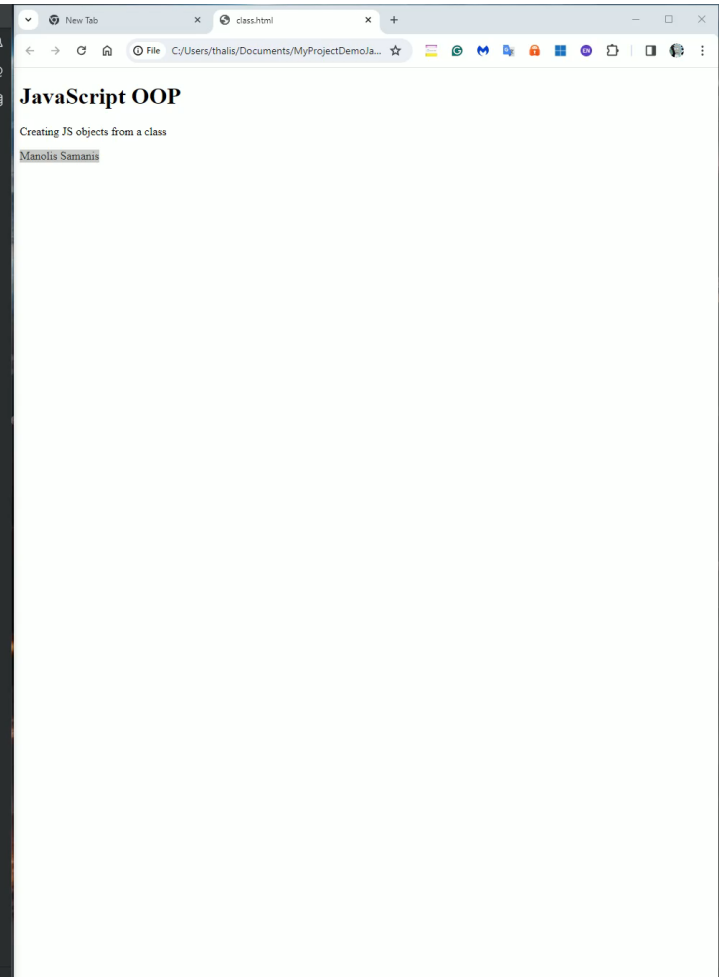
- Browser Privacy Settings: private browsing, cookie blocking, and third-party cookie restrictions
- Cookie Blockers, extensions or browsers
- Anonymization with Tor (not 100% safe)
- Use of Tails, a portable operating system that protects against surveillance and hides your identity

Object-Oriented Programming With JavaScript (OOP)

The JavaScript language has been designed on an object-oriented basis. With OOP a developer can:

- Create their own objects and organize better the code by making it more flexible and maintainable
- Use the pre-made built-in objects provided by JavaScript (JSON, Date, Math)
- We use OOP to model or better describe real-world examples
- The objects can contain data or methods. With objects we incorporate the data and their behavior into a block of code
- Objects can interact with one another
- We can use an API methods to access and communicate with the object


```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <h1>JavaScript OOP</h1>
5
6 <p>Creating JS objects from a class</p>
7
8 <p id="demo"></p>
9
10 <script>
11     no usages
12     class Names {
13         no usages
14         constructor(firstName, lastName) {
15             this.firstName = firstName;
16             this.lastName = lastName;
17         }
18     }
19
20     const myName = new Names("Manolis", "Samanis");
21
22     document.getElementById("demo").innerHTML =
23         myName.firstName + " " + myName.lastName ;
24 </script>
25 </body>
26 </html>
```



New Tab class.html

File C:/Users/thalis/Documents/MyProjectDemo...

JavaScript OOP

Creating JS objects from a class

[Manolis Samanis](#)

JavaScript Built-in objects

JSON (JavaScript Object Notation) is used for storing and transmitting data, primarily in web applications.

- JSON objects share many similarities with Array objects.
- Data is recorded in the format key:value.

Example:

```
JavaScript + No-Library (pure JS) ▼ Tidy
1 let products = {};
2
3 products.apples = 1.5;
4 products.bananas = 2.5;
5 products.oranges = 1.1;
6 products.pears = 1.9;
7
8 for (key in products) {
9   console.log(key, products[key]);
10 }
```

"apples", 1.5
"bananas", 2.5
"oranges", 1.1
"pears", 1.9
>_

The date object is used for managing dates and times

Example:

```
1 <div id = "timer"></div>
2 <script>
3   var d = new Date();
4   var t = document.getElementById("timer");
5   t.innerHTML = d.getHours();
6 </script>
```

20

Objects and Instances

In JavaScript, anything stored in a variable is an object. To use an object, we first need to create an instance of that object.

Creating an object is done using the new operator -> **var d = new Date();**

Each instance (or object, as we will call it) has:

- Properties or characteristics (features, properties, or fields)
- Functions or methods
- Ability to handle events

An object can have sub objects (child objects) -> **window.document**

To add an event handler, we use the method -> **addEventListener("eventname",
functionname);**

```
1  var btn = document.getElementById("myButton");
2  btn.addEventListener("click", getValue);
3
4  function getValue() {
5      ...
6  }
```

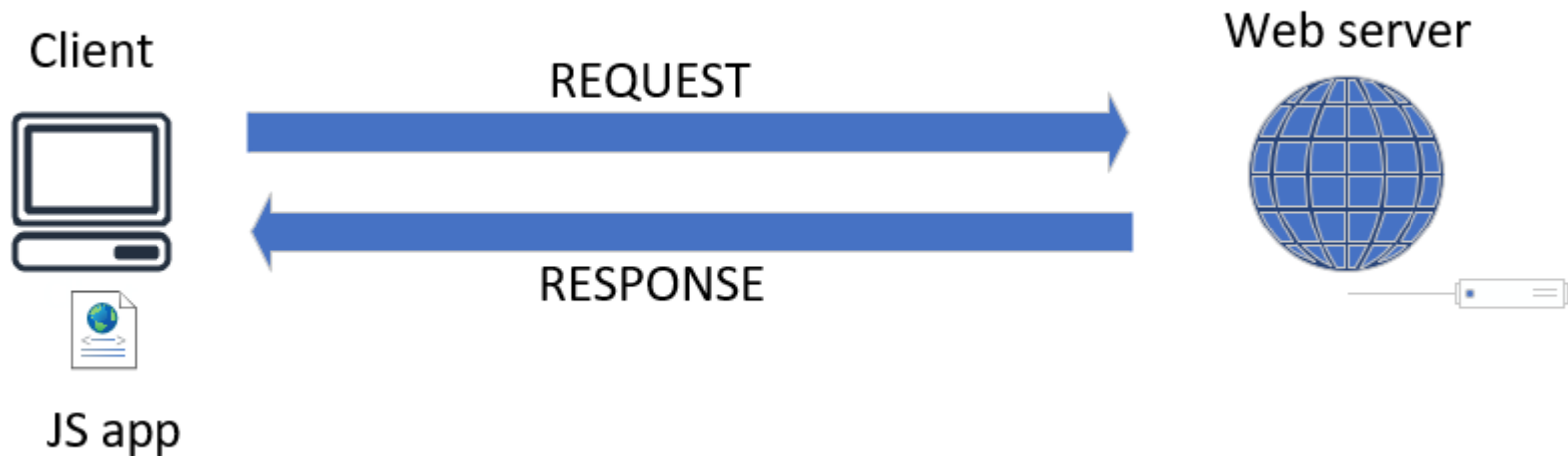
JavaScript Custom Objects with functions

```
index3.html
car_custom_objects.html
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h2>JavaScript Custom Objects with functions</h2>
6
7 <p id="demo"></p>
8
9 <button id="displayButton">What are my car details?</button>
10
11 <div id="output"></div>
12
13 <script>
14     var myObject = {
15         make: "Corssa",
16         year: "2015",
17         price: 5000,
18         color: "blue"
19     };
20
21     no spaces
22     function displayObjectValue() {
23         var outputDiv = document.getElementById("output");
24         outputDiv.innerHTML = "make: " + myObject.make + "<br>" + "year: " + myObject.year + "<br>" +
25             "price: " + myObject.price + "<br>" + "color: " + myObject.color;
26     }
27     document.getElementById("displayButton").addEventListener("click", displayObjectValue);
28 </script>
29
30
31
32 </body>
33 </html>
34
```

Asynchronous JavaScript and AJAX

AJAX (Asynchronous JavaScript And XML):

- It can communicate with remote web servers in an asynchronous manner
- It requests data from web servers dynamically



Asynchronous JavaScript and AJAX

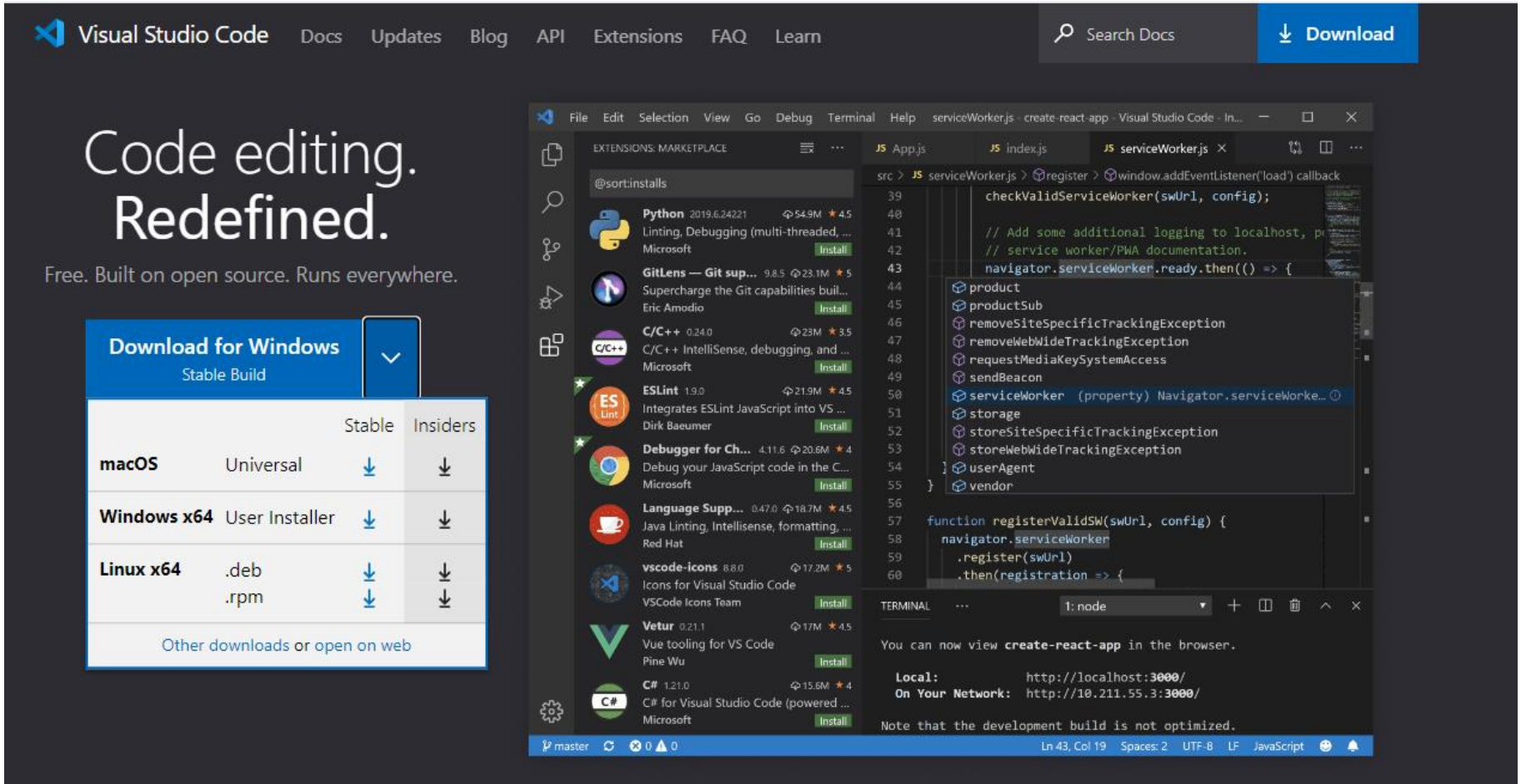
```
index3.html
AJAX.html
1 <!DOCTYPE html>
2 <html lang="en">
3
4
5 <head>
6 <meta charset="UTF-8">
7 <meta name="viewport" content="width=device-width, initial-scale=1.0">
8 <title>A simple AJAX Request Example</title>
9 </head>
10
11 <body>
12
13 <h2>AJAX request Demo</h2>
14
15 <button id="loadData">Load external Data</button>
16
17 <div id="output"></div>
18
19 <script>
20     document.getElementById('loadData').addEventListener('click', function() {
21         var xhr = new XMLHttpRequest(); // we create here the XMLHttpRequest object
22         xhr.open('GET', 'https://jsonplaceholder.typicode.com/posts/1', true); // we specify the request type and the URL we need
23         xhr.onreadystatechange = function () { // here define the callback function to handle the response
24             if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) { // check for the request if successful
25                 document.getElementById('output').innerHTML = xhr.responseText; // we update the content of the output div with the response text
26             }
27         };
28         xhr.send(); // here we send the request
29     });
30 </script>
31
32
33 </body>
34 </html>
35
```

Suggested resources for further reading

<https://developer.mozilla.org/en-US/docs/Learn/JavaScript>

JavaScript: The Definitive Guide, Author: David Flanagan

JS editor VS code



Visual Studio Code Docs Updates Blog API Extensions FAQ Learn

Search Docs Download

Code editing. Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows
Stable Build

		Stable	Insiders
macOS	Universal	↓	↓
Windows x64	User Installer	↓	↓
Linux x64	.deb .rpm	↓ ↓	↓ ↓

[Other downloads or open on web](#)

EXTENSIONS: MARKETPLACE

- Python** 2019.6.24221 54.9M ★ 4.5
Linting, Debugging (multi-threaded, ...
Microsoft [Install](#)
- GitLens — Git sup...** 9.8.5 23.1M ★ 5
Supercharge the Git capabilities built...
Eric Amodio [Install](#)
- C/C++** 0.24.0 23M ★ 3.5
C/C++ IntelliSense, debugging, and ...
Microsoft [Install](#)
- ESLint** 1.9.0 21.9M ★ 4.5
Integrates ESLint JavaScript into VS ...
Dirk Baeumer [Install](#)
- Debugger for Ch...** 4.11.6 20.6M ★ 4
Debug your JavaScript code in the C...
Microsoft [Install](#)
- Language Supp...** 0.47.0 18.7M ★ 4.5
Java Linting, Intellisense, formatting, ...
Red Hat [Install](#)
- vscode-icons** 8.8.0 17.2M ★ 5
Icons for Visual Studio Code
VSCode Icons Team [Install](#)
- Vetur** 0.21.1 17M ★ 4.5
Vue tooling for VS Code
Pine Wu [Install](#)
- C#** 1.21.0 15.6M ★ 4
C# for Visual Studio Code (powered ...
Microsoft [Install](#)

```

src > JS serviceWorker.js > register > window.addEventListener('load') callback
39
40
41   checkValidServiceWorker(swUrl, config);
42
43   // Add some additional logging to localhost, p
44   // service worker/PWA documentation.
45   navigator.serviceWorker.ready.then(() => {
46     product
47     productSub
48     removeSiteSpecificTrackingException
49     removeWebWideTrackingException
50     requestMediaKeySystemAccess
51     sendBeacon
52     serviceWorker (property) Navigator.serviceWorke...
53     storage
54     storeSiteSpecificTrackingException
55     storeWebWideTrackingException
56     userAgent
57     vendor
58
59   function registerValidSW(swUrl, config) {
60     navigator.serviceWorker
61       .register(swUrl)
62       .then(registration => {

```

TERMINAL ... | node

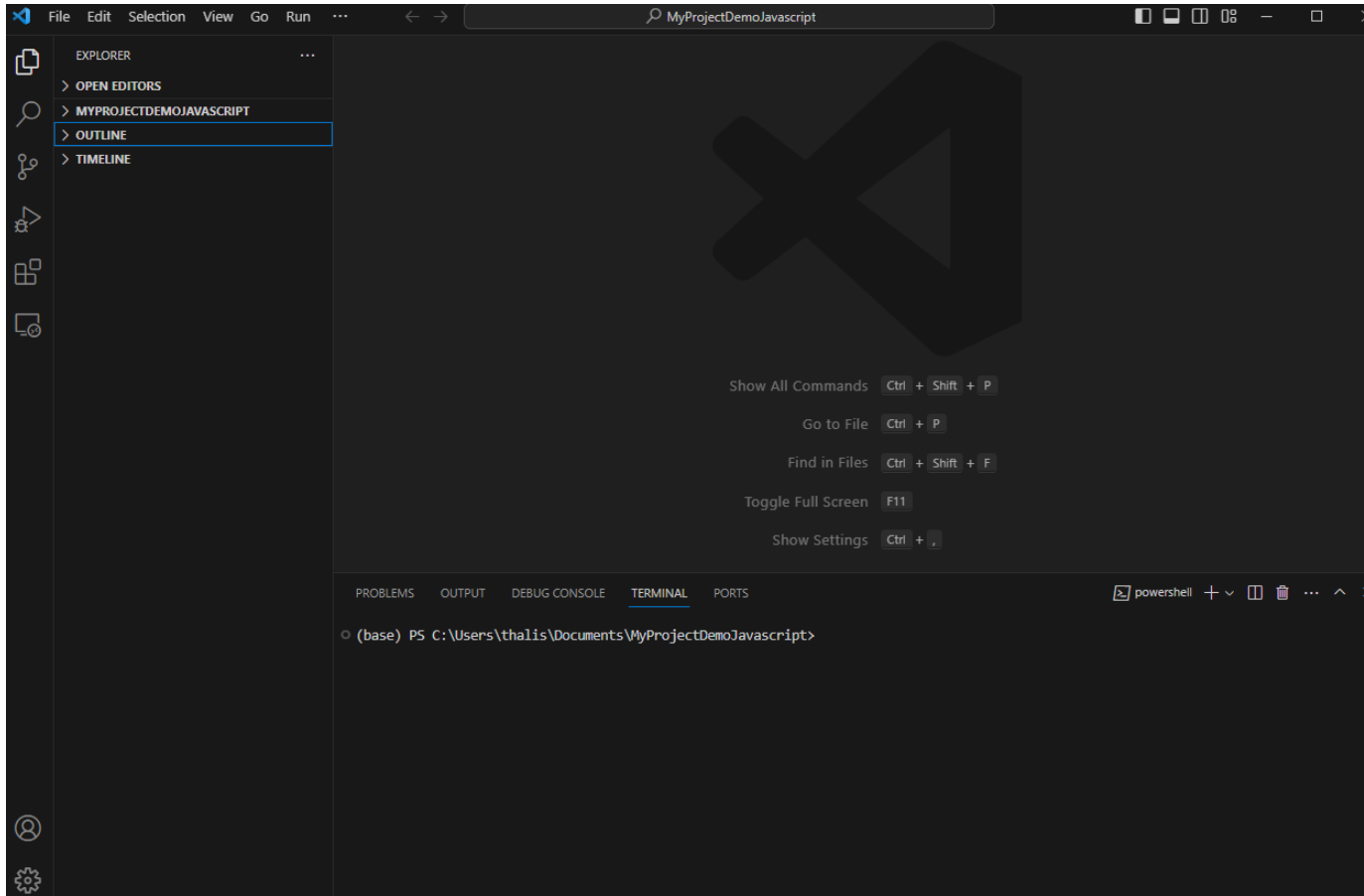
You can now view **create-react-app** in the browser.

Local: http://localhost:3000/
On Your Network: http://10.211.55.3:3000/

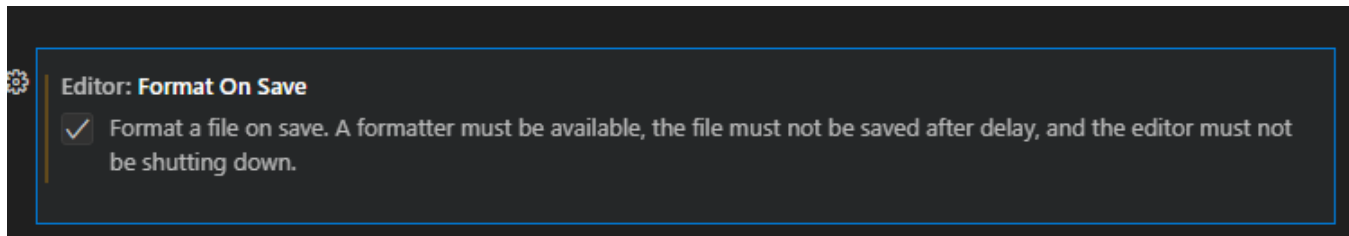
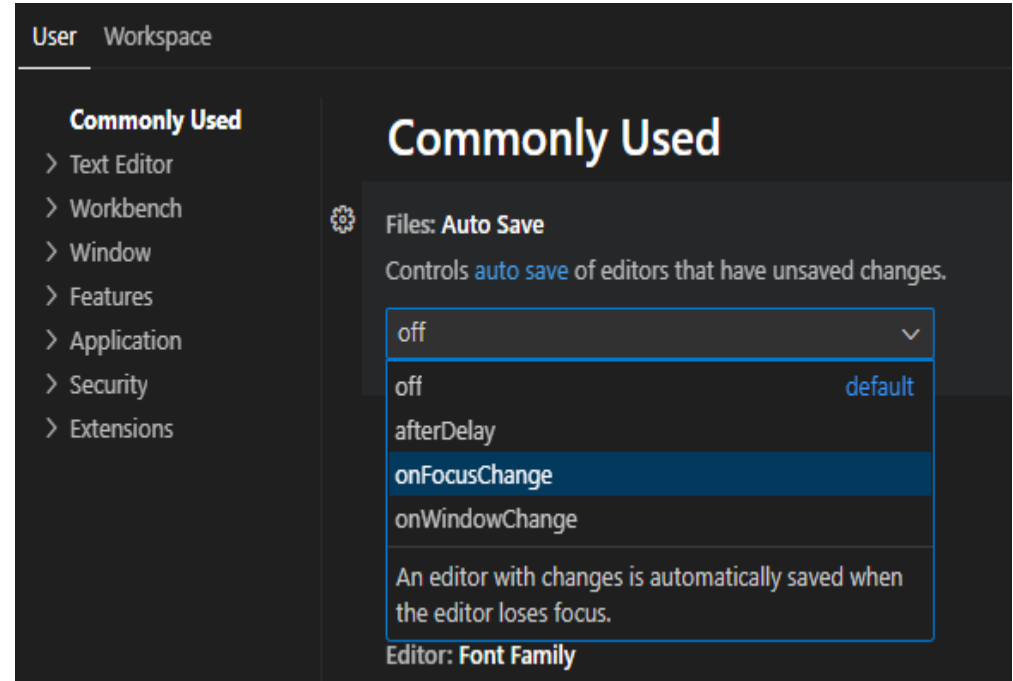
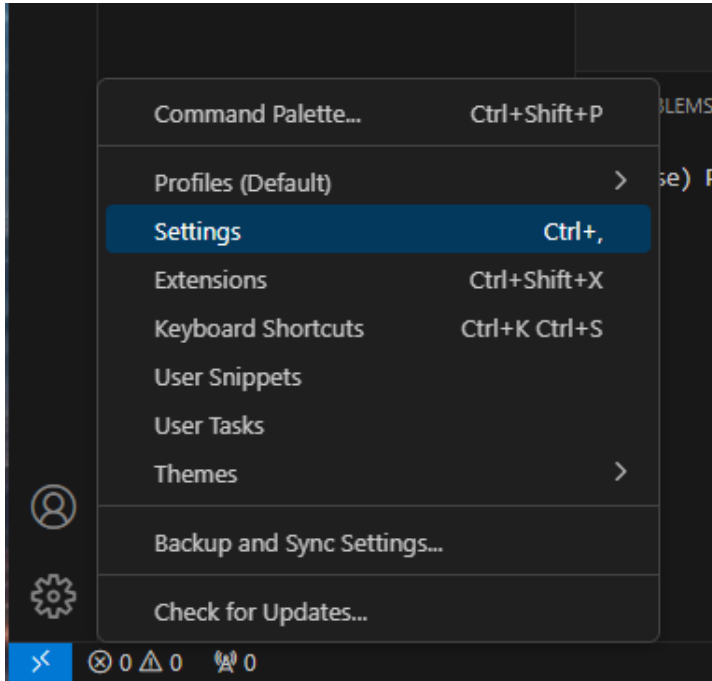
Note that the development build is not optimized.

Ln 43, Col 19 Spaces: 2 UTF-8 LF JavaScript

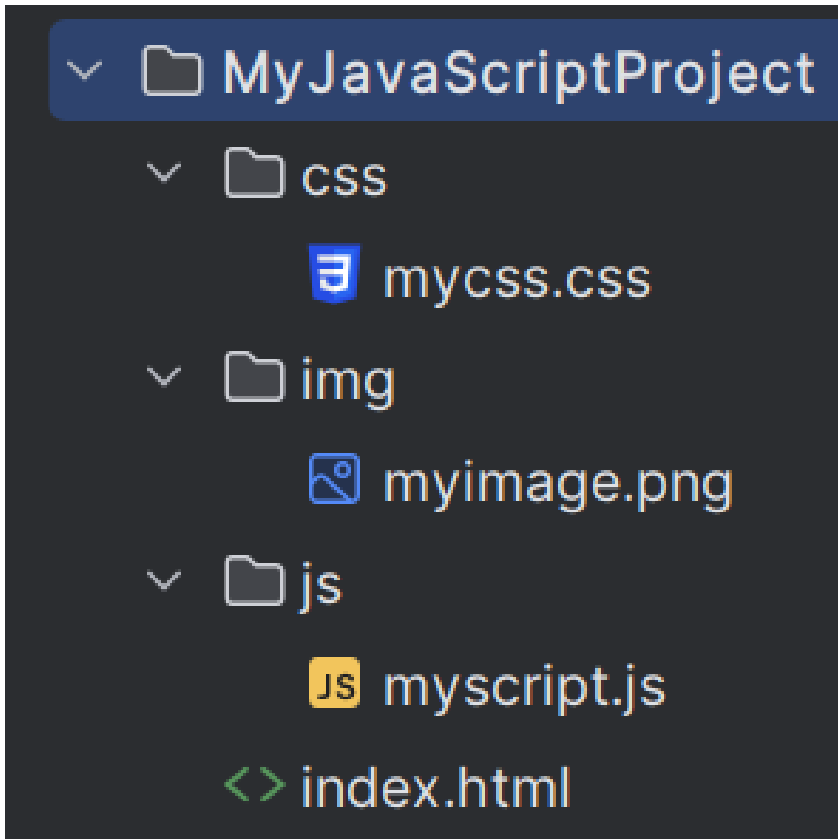
JS editor VS code



JS editor VS code



How to organize your project folder and file structure

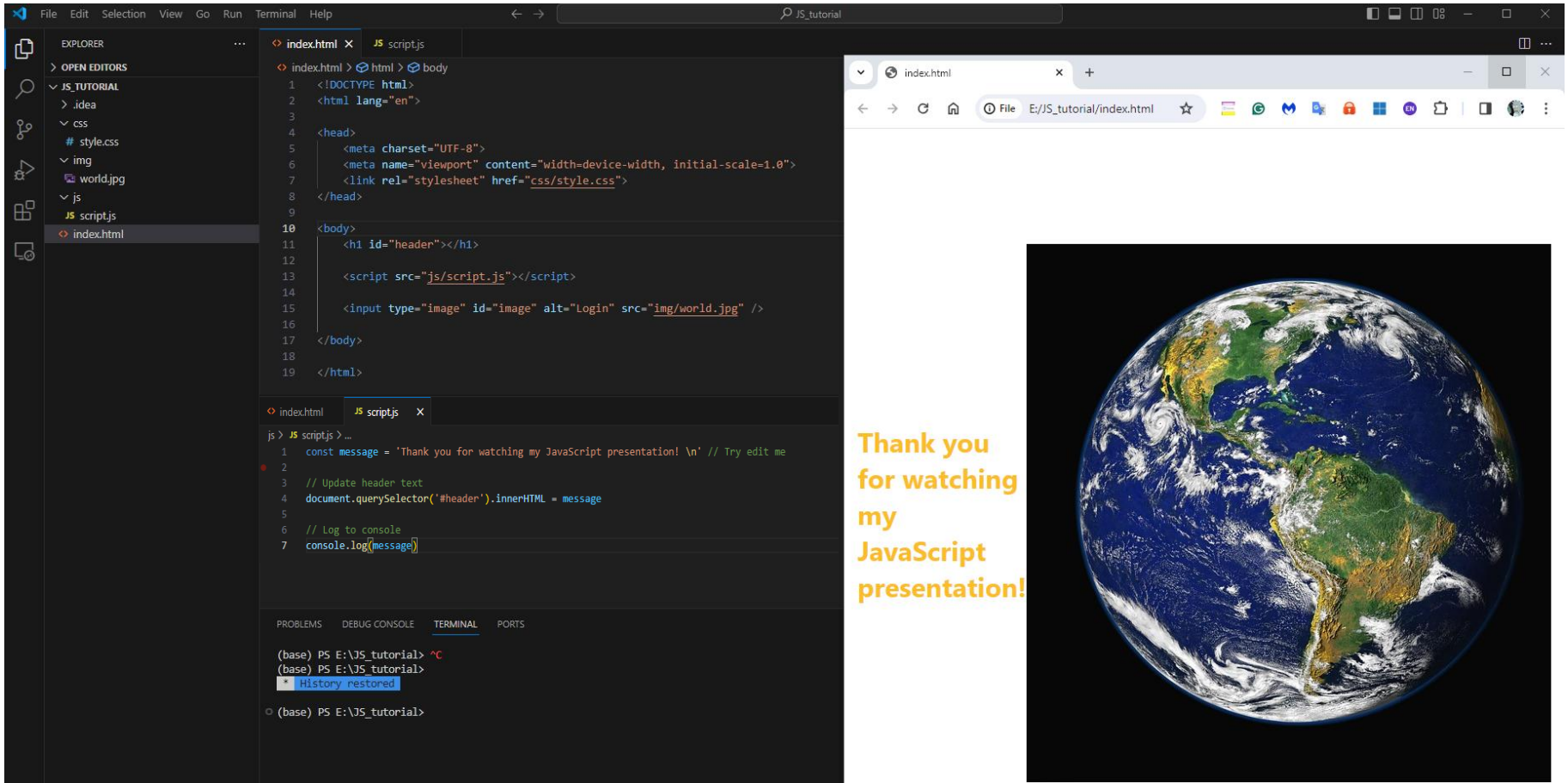


`<link rel="stylesheet" type="text/css" href="css/mycss.css">`

``

`<script src="js/myscript.js"></script>`

JS editor VS code



The image shows a VS Code editor window with two files open: `index.html` and `script.js`. The `index.html` file contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <link rel="stylesheet" href="css/style.css">
8 </head>
9
10 <body>
11   <h1 id="header"></h1>
12
13   <script src="js/script.js"></script>
14
15   <input type="image" id="image" alt="Login" src="img/world.jpg" />
16
17 </body>
18
19 </html>
```

The `script.js` file contains the following JavaScript code:

```
1 const message = 'Thank you for watching my JavaScript presentation! \n' // Try edit me
2
3 // Update header text
4 document.querySelector("#header").innerHTML = message
5
6 // Log to console
7 console.log(message)
```

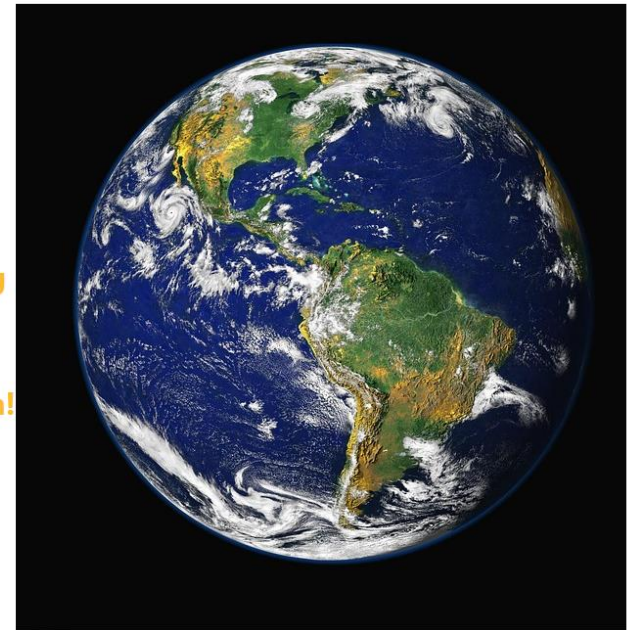
The browser window shows the rendered page with the following HTML structure:

```
<h1 id="header"></h1>
<script src="js/script.js"></script>
<input type="image" id="image" alt="Login" src="img/world.jpg" />
```

The browser window also displays a globe image, which is the content of the `img/world.jpg` file. The terminal window shows the following output:

```
(base) PS E:\JS_tutorial> ^C
(base) PS E:\JS_tutorial>
* History restored
(base) PS E:\JS_tutorial>
```

Thank you
for watching
my
JavaScript
presentation!



Have any questions?

