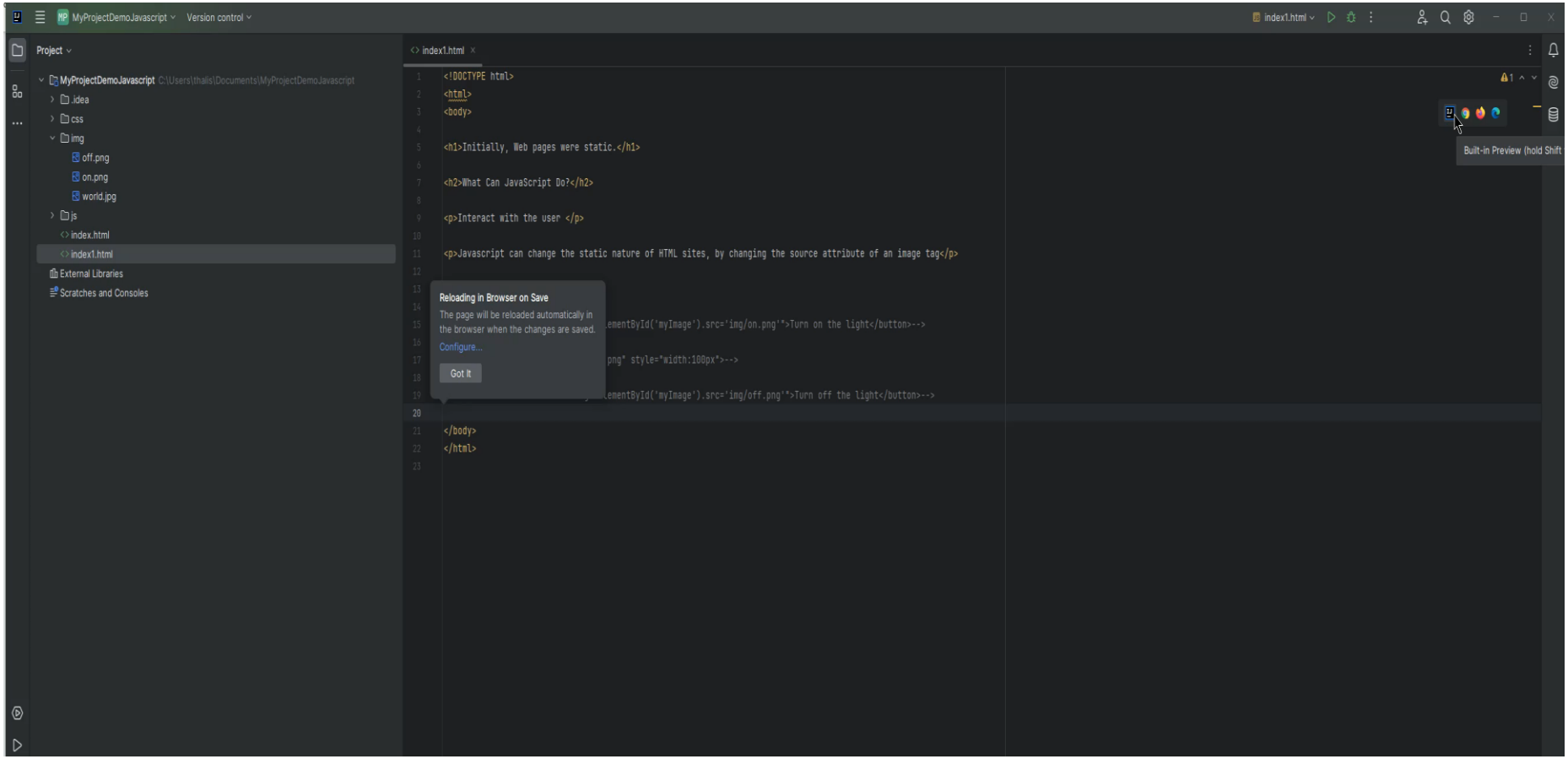
JavaScript

An introduction to JavaScript Programming

Why we need JavaScript?



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

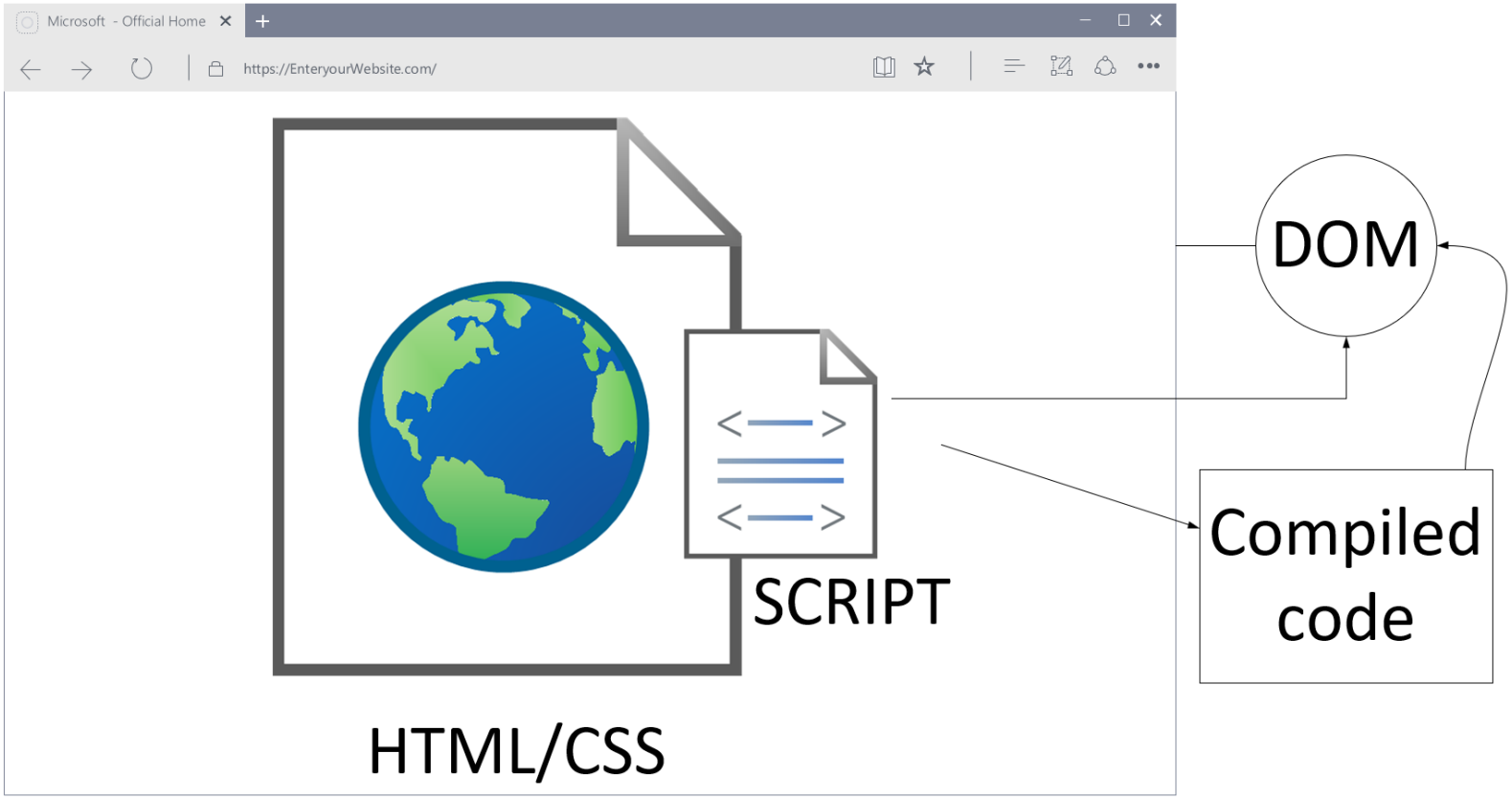
defer: the script is downloaded in parallel while parsing the page

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

Inline JavaScript

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

Just-in-time compiler (JIT)



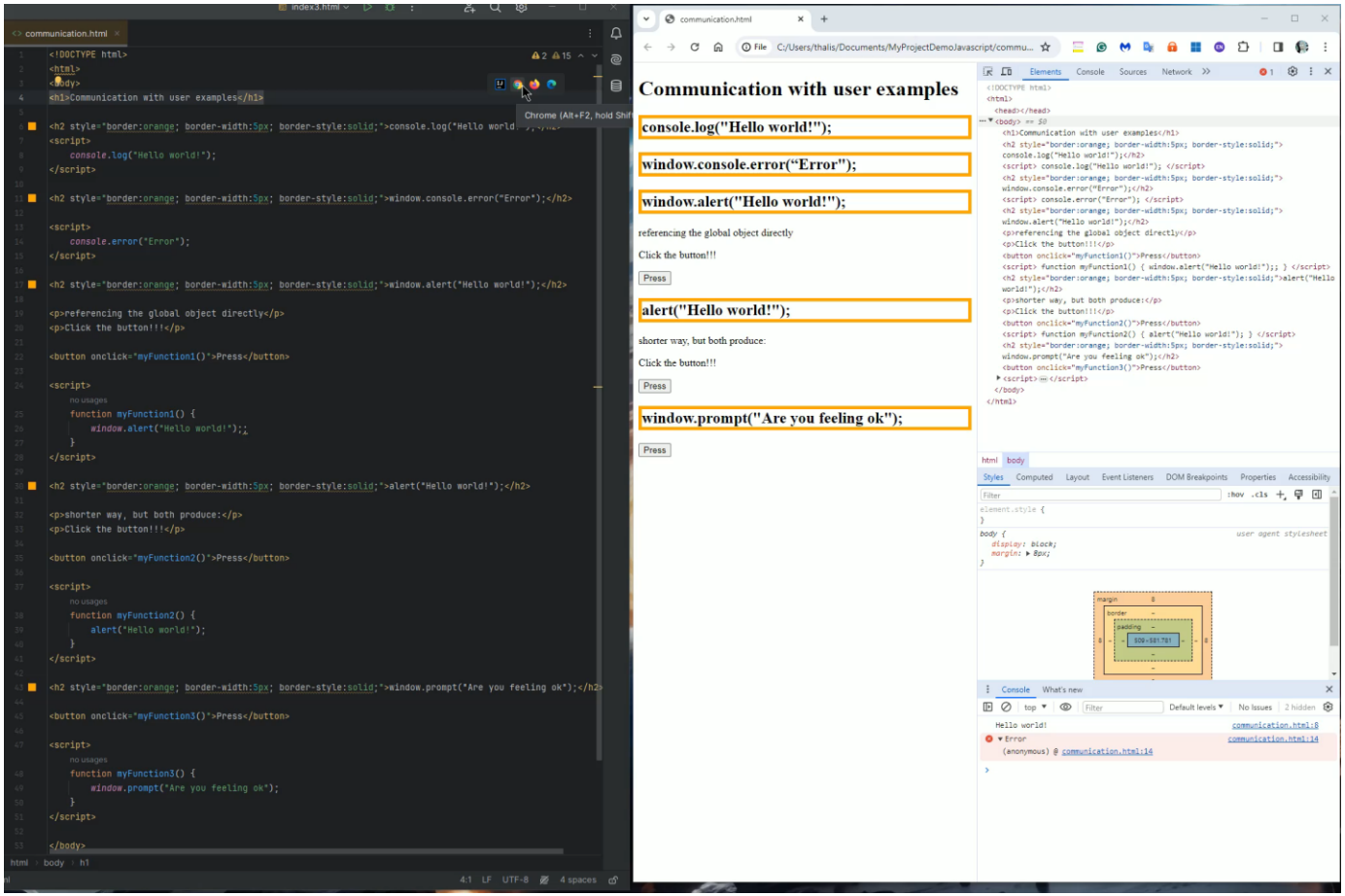
JavaScript communication with user examples

- 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



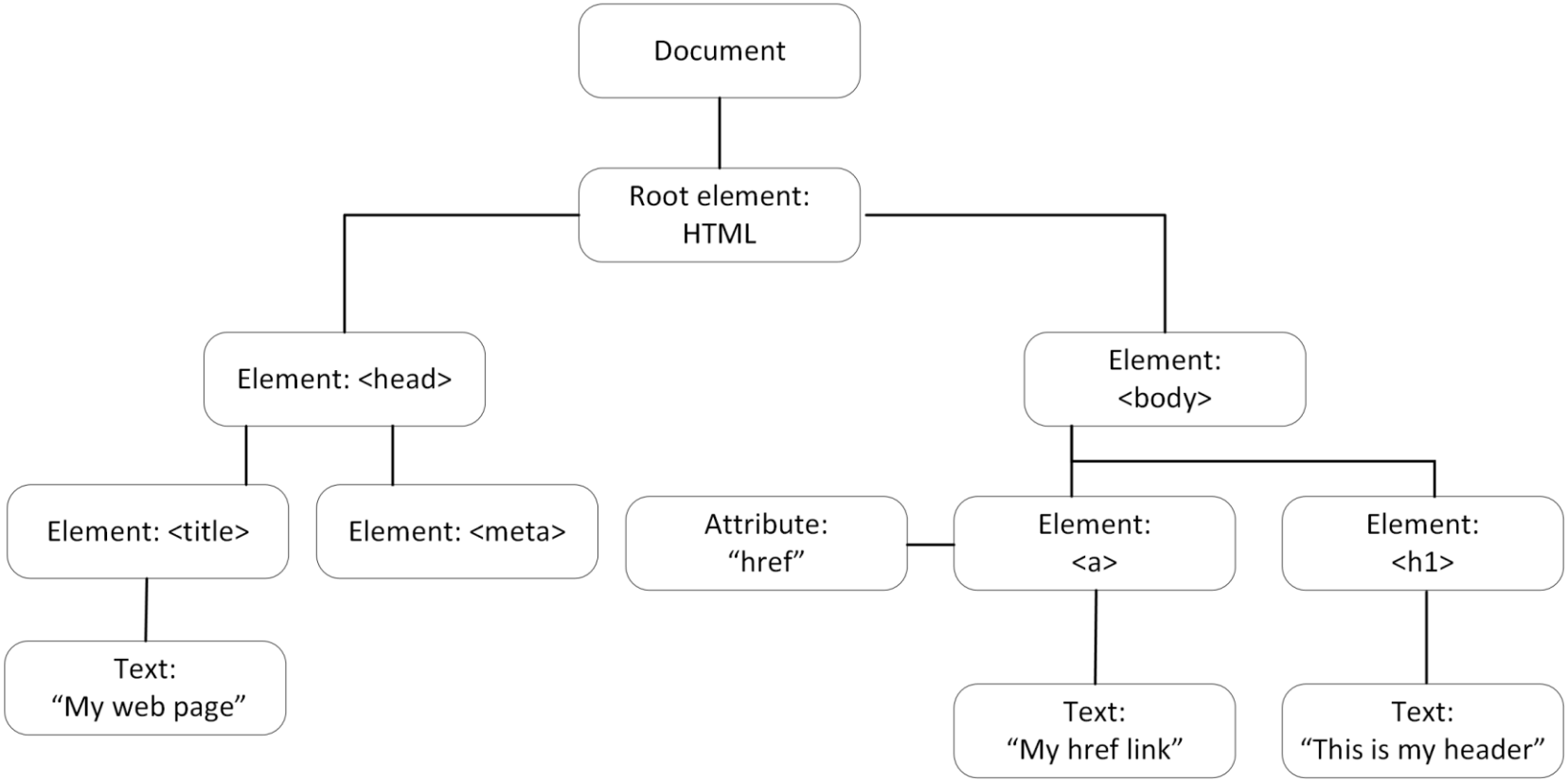
JavaScript engines

Every browser starts two processes for every web page

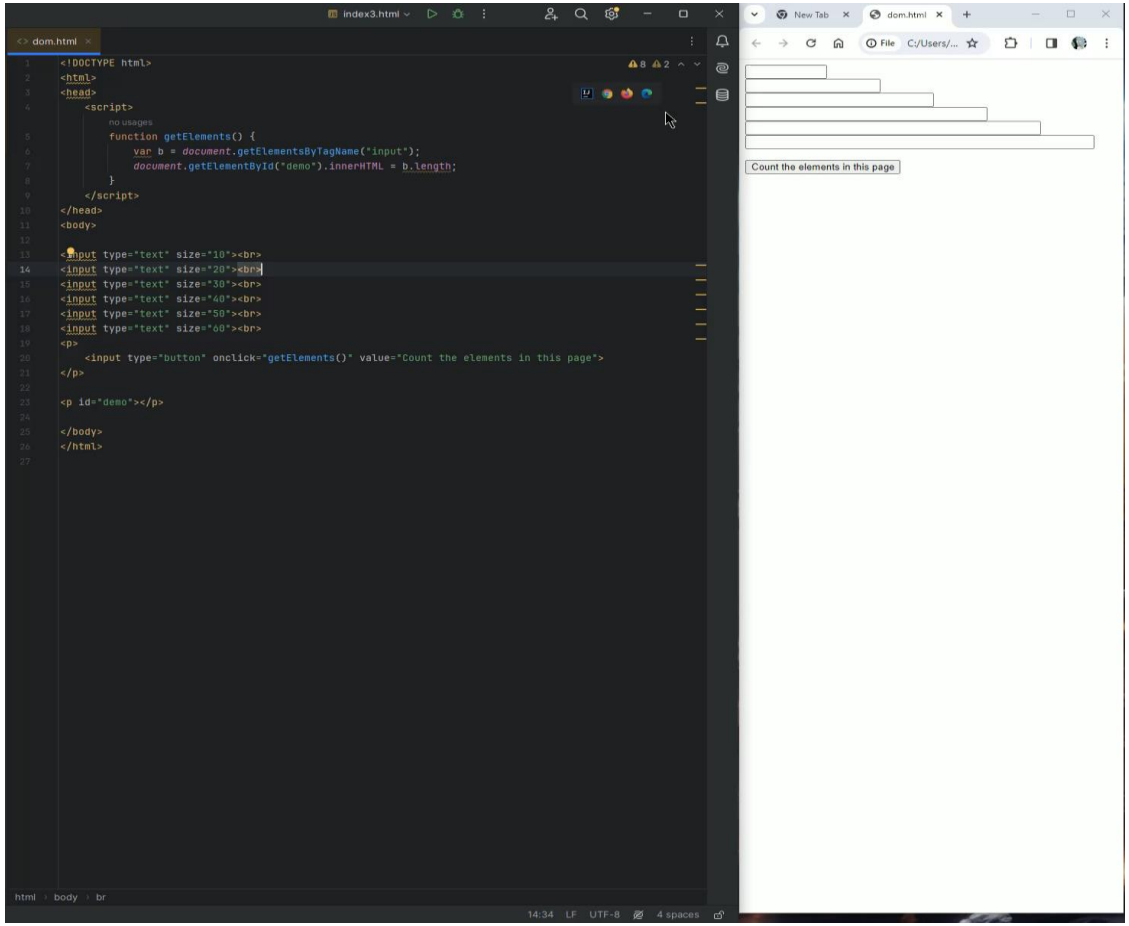
- **Rendering engine:** it is responsible to display the web page. Parses the HTML and CSS and displays the content on the screen
- **JavaScript engine:** This is where JavaScript code gets executed

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?



DOM Manipulation in JavaScript

Microsoft - Official Home

https://EnteryourWebsite.com/

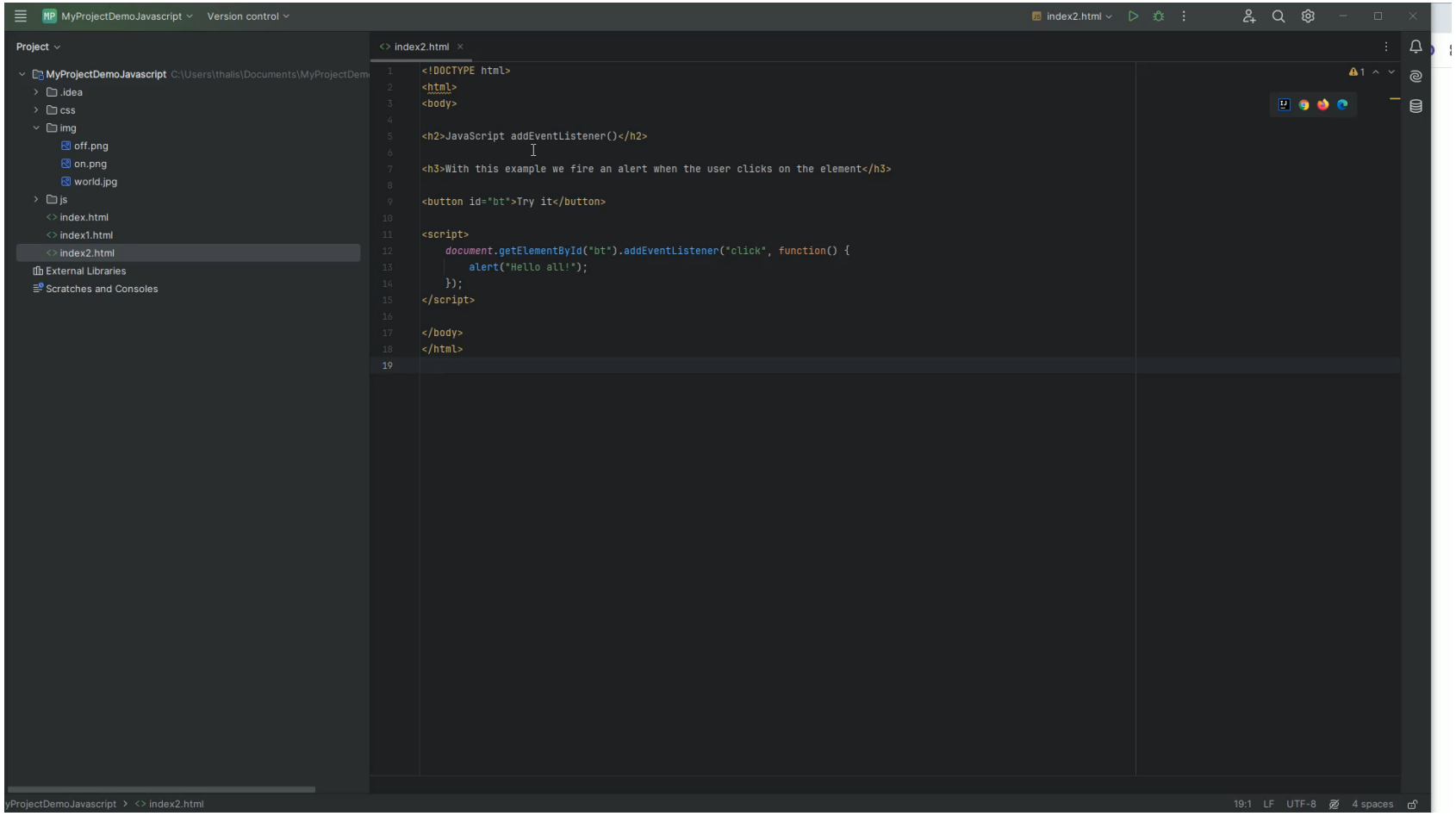
```
<!DOCTYPE html>
<html>
  <head>
  <title>JavaScript Demo page</title>
  </head>
<body>

<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

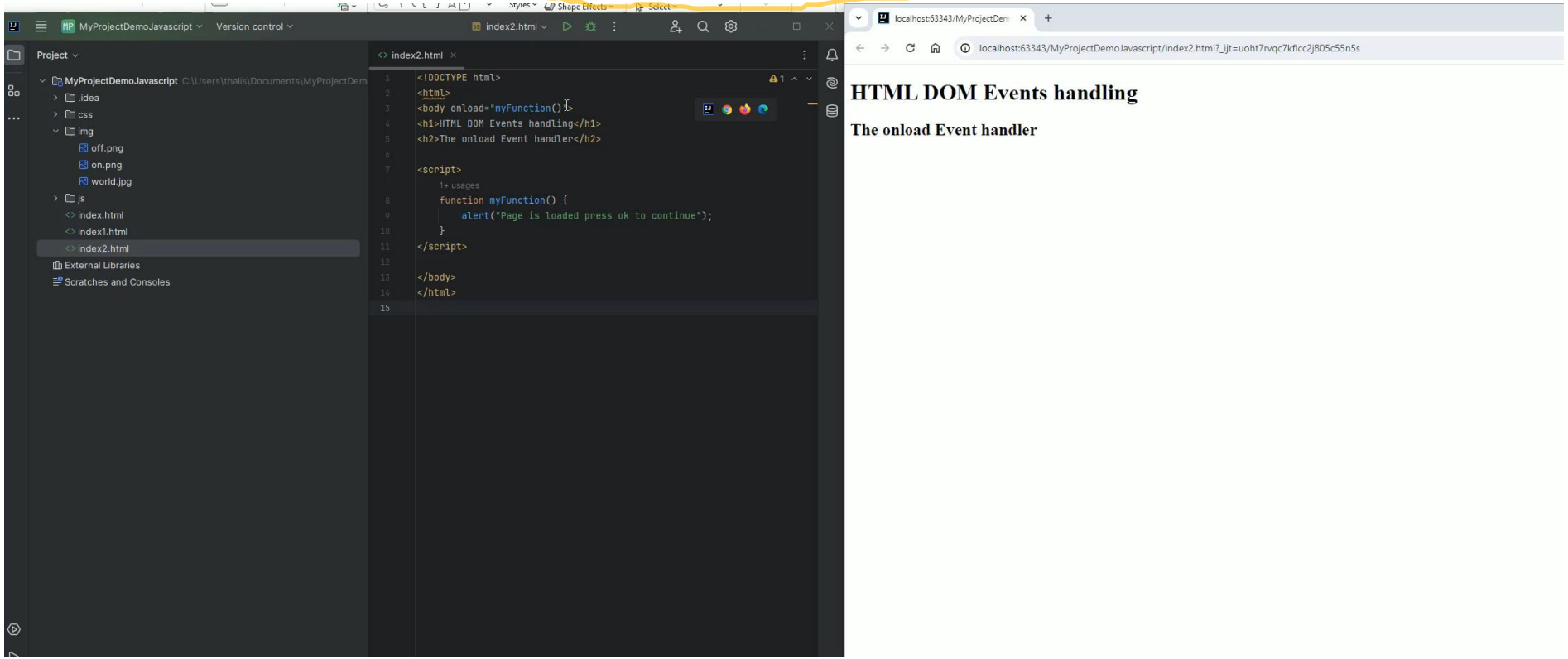
Main document
(DOM, HTML)

JavaScript HTML DOM EventListener

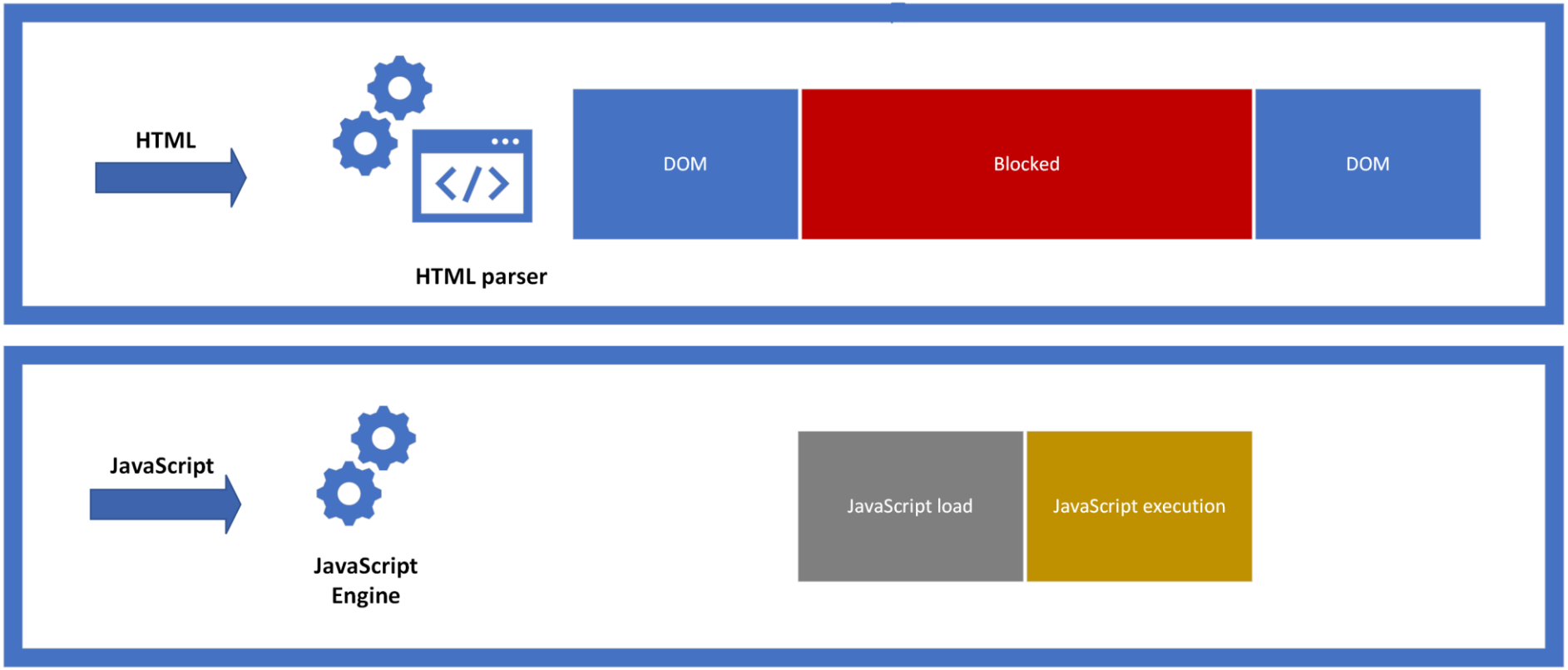


How to Make JavaScript Execute After HTML Load onload Event

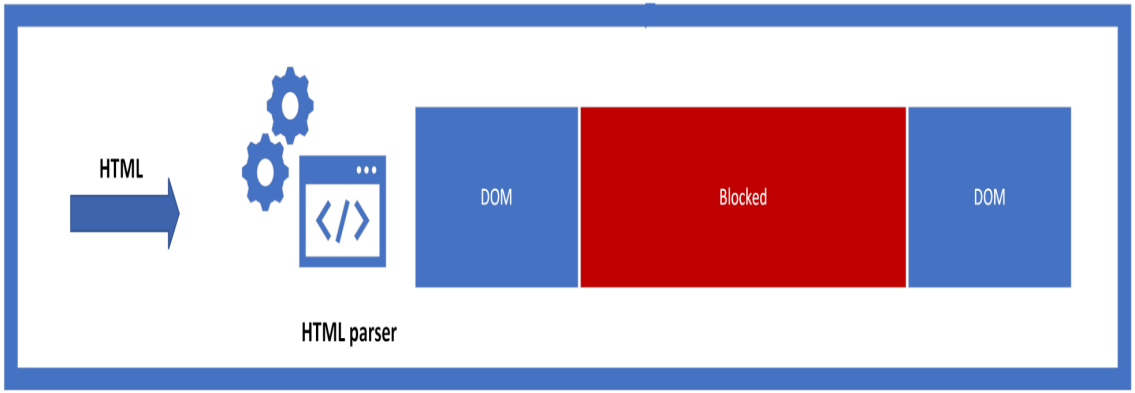
If we use JavaScript to handle a Document Object Model (DOM) then our code executes after HTML (blocked code until OK!)

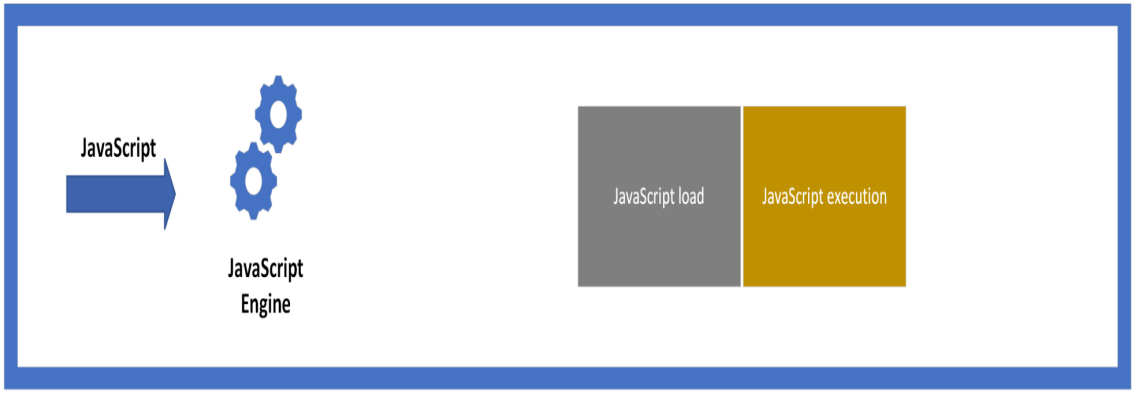


Execute JavaScript: parser-blocking scripts (synchronous)



Execute JavaScript: parser-blocking scripts (synchronous)



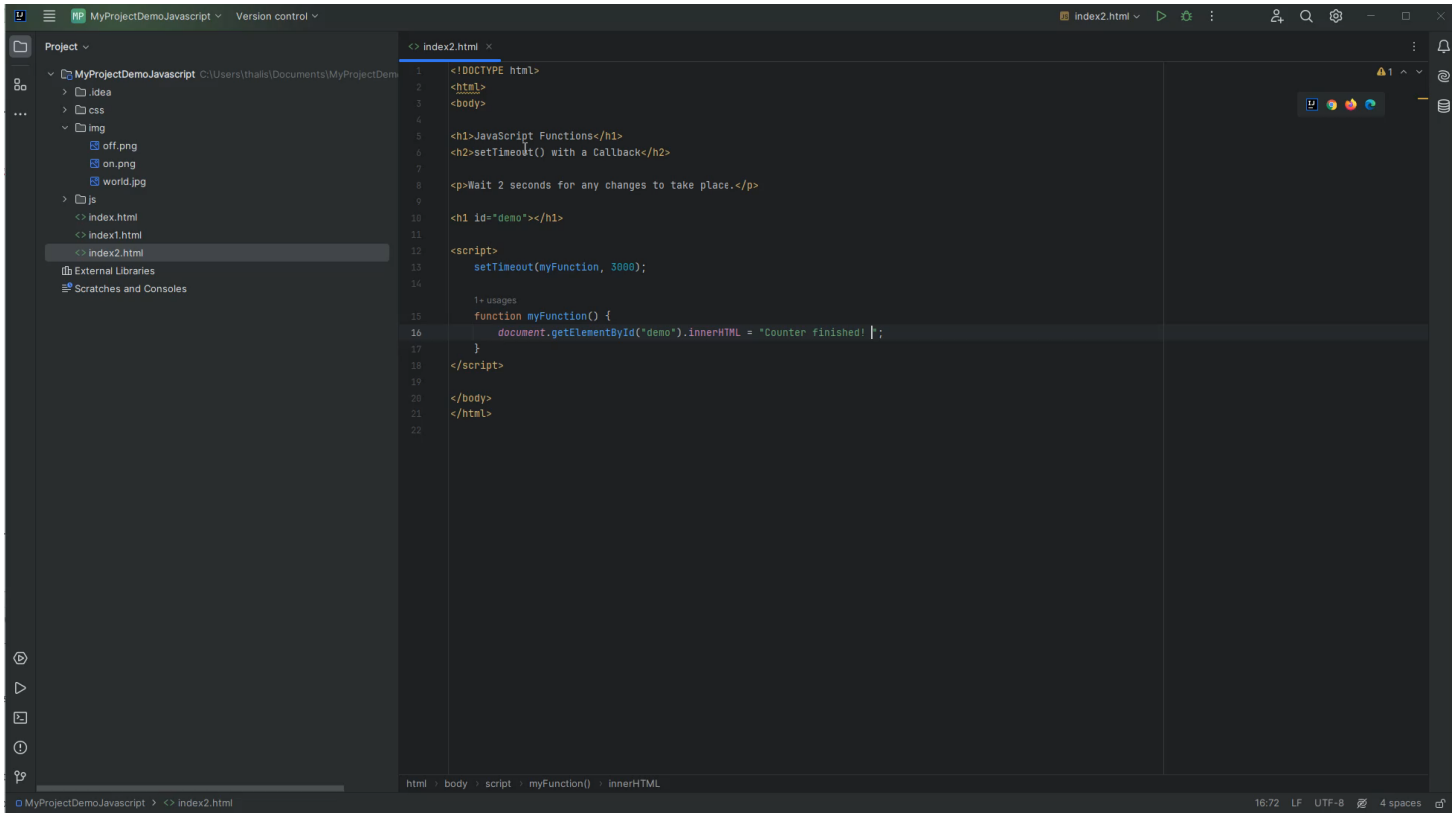


Asynchronous JavaScript

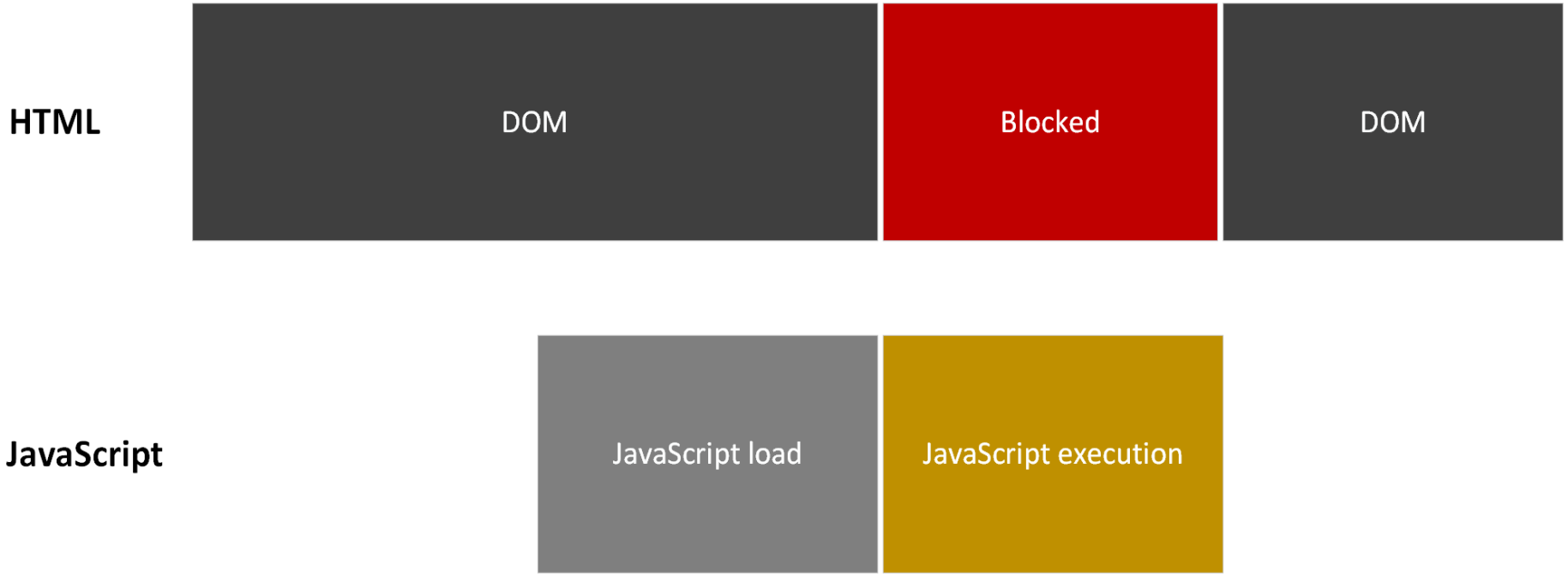
异步与同步编码：

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



Asynchronous JavaScript



Variable Names

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

```
validVariable
_privateVar
$money
myVariable123
```

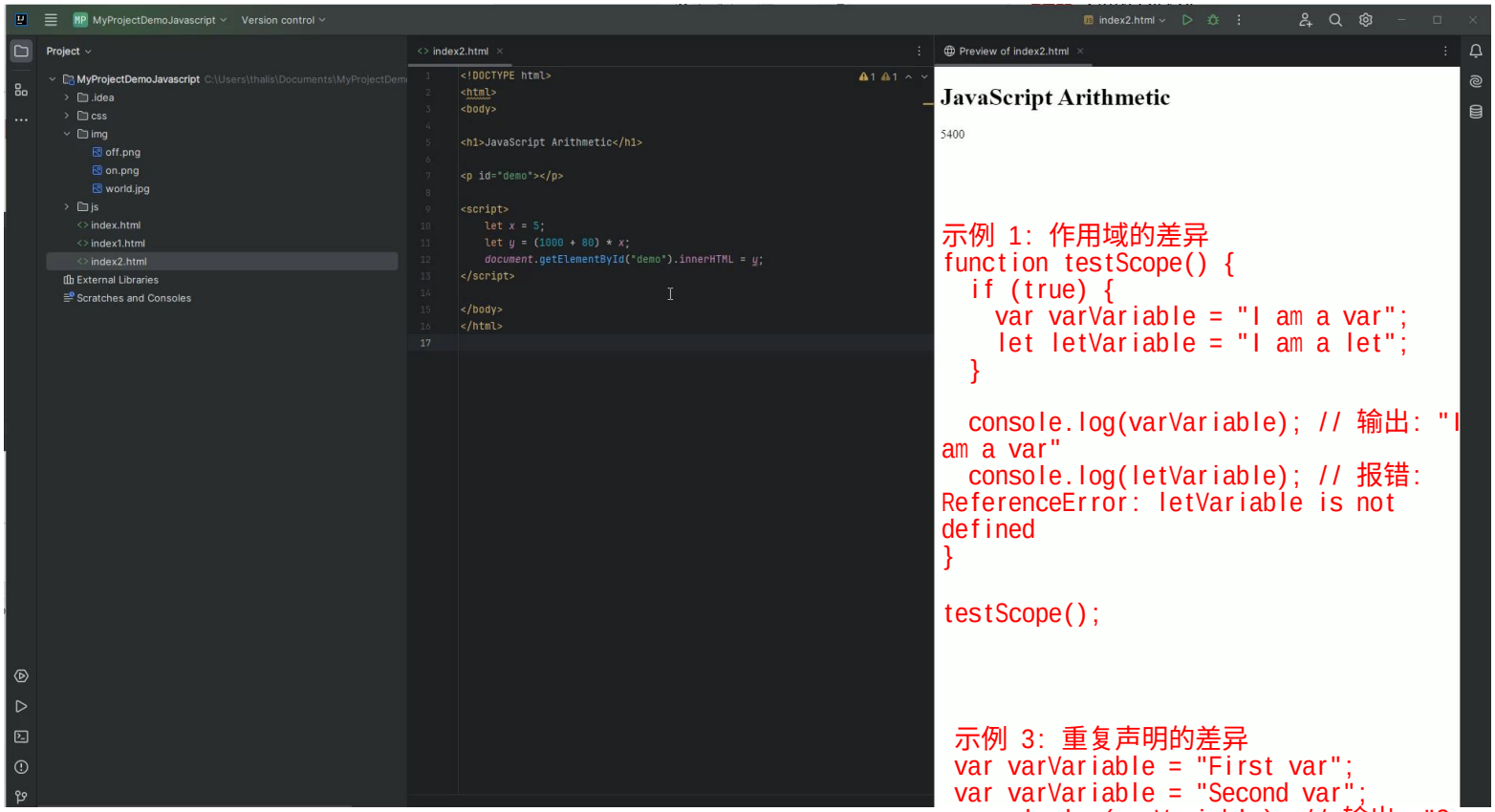
```
let $4 = 1;      → Block
let _4 = 10;
var $_$ = "money"; → Function
var I_AM_HUNGRY = true;
```

`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




```
示例 1: 作用域的差异
function testScope() {
  if (true) {
    var varVariable = "I am a var";
    let letVariable = "I am a let";
  }

  console.log(varVariable); // 输出: "I
am a var"
  console.log(letVariable); // 报错:
ReferenceError: letVariable is not
defined
}

testScope();
```

```
示例 3: 重复声明的差异
var varVariable = "First var";
var varVariable = "Second var";
console.log(varVariable); // 输出: "Second var"

let letVariable = "First let";
let letVariable = "Second let"; // 报错:
SyntaxError: Identifier 'letVariable' has
already been declared
```

String in JavaScript

作用域（Scope）：
var 声明的变量具有函数作用域，或者如果在函数外部声明，则为全局作用域。这意味着如果 var 在一个函数内部声明，它只能在该函数内部访问，但如果在函数外声明，它可以在全局范围内访问。
let 声明的变量具有块级作用域（block scope），即只能在包含它们的代码块（例如：for循环、if语句等）中访问。

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 ] ]
```

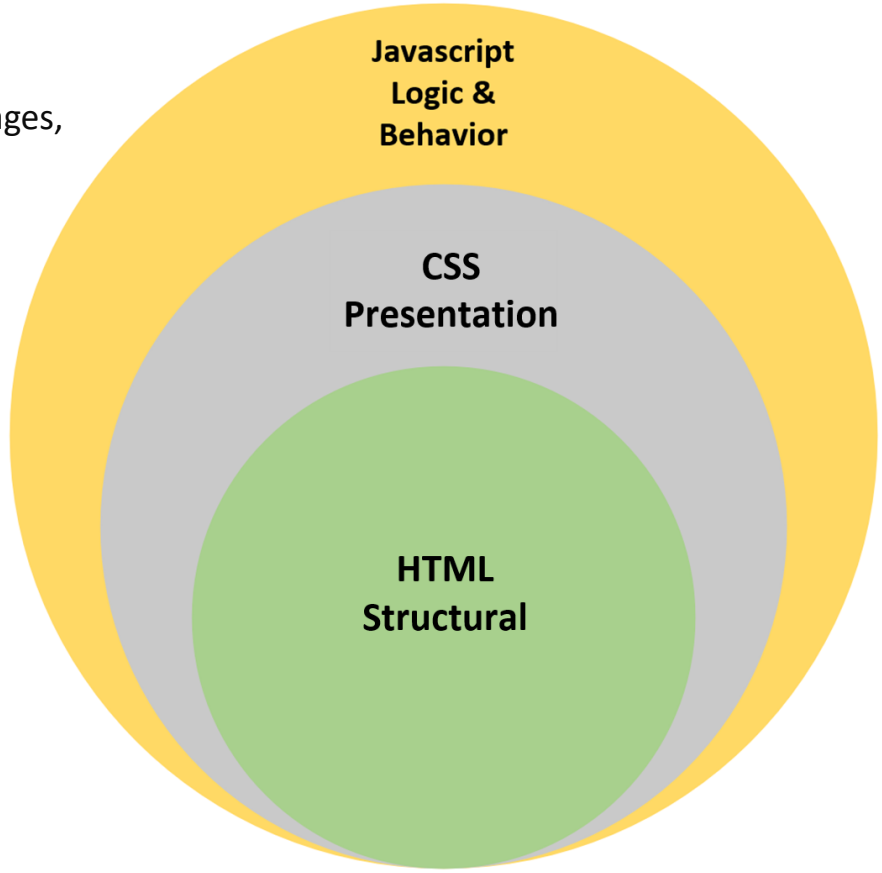
在交互式JavaScript环境中，当你输入一个语句时，它会执行该语句并返回其结果。在这种情况下，当你输入 let numbers = ['one', 'two', 'three']; 时，它会声明一个名为 numbers 的变量，并将一个包含三个字符串元素的数组赋值给它。因为这个语句本身没有返回值，所以会显示 undefined。实际上，let 声明的结果通常是 undefined，表示该变量已被成功声明，但尚未赋值。

JavaScript and html

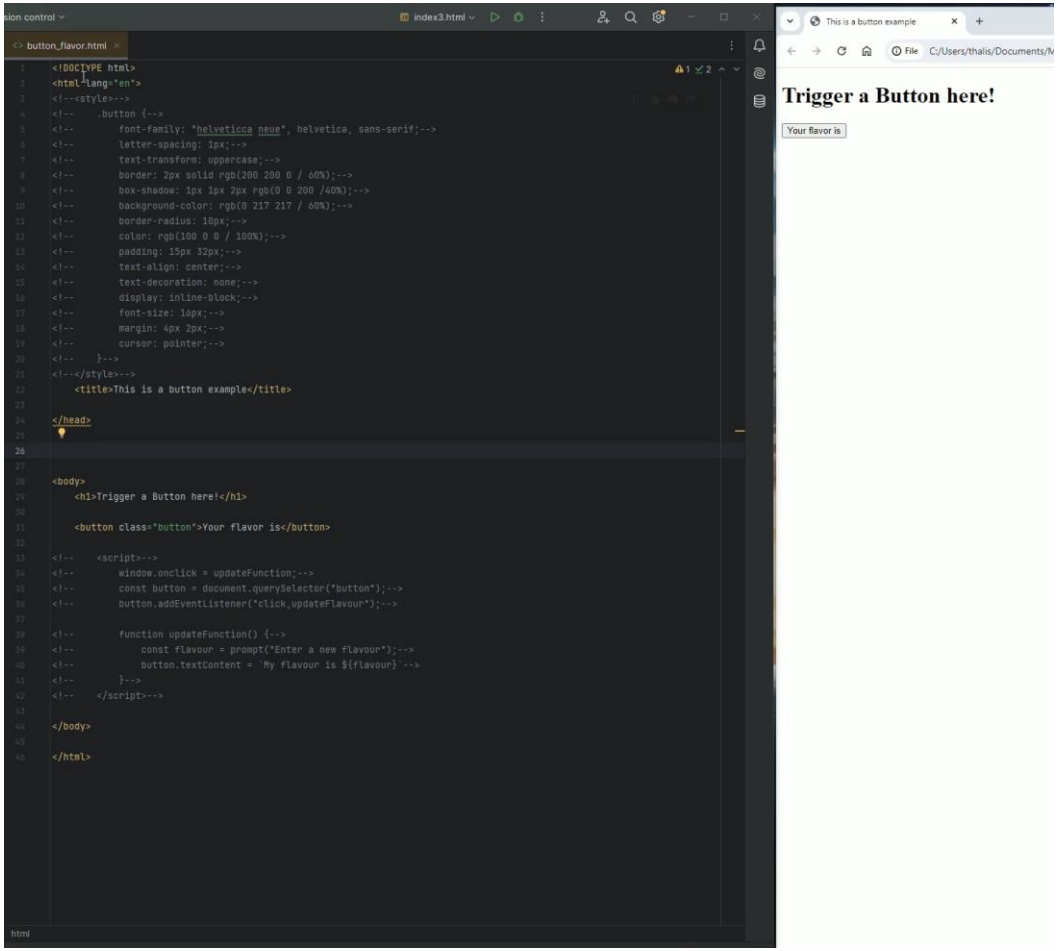
JavaScript is a scripting language utilized within an HTML document, to add functionality to the document or if you prefer to make a document dynamic.

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



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

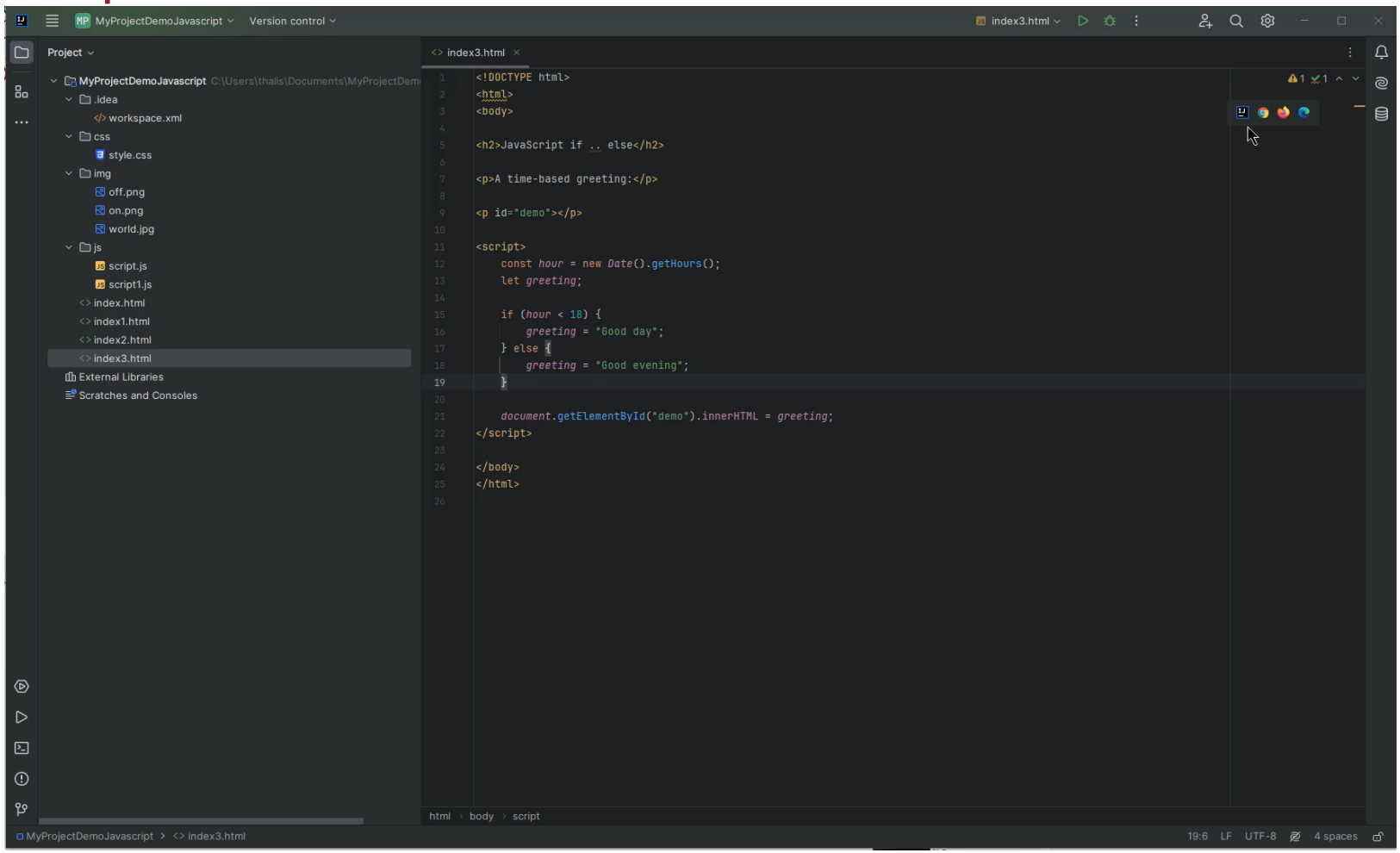
General syntax:

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

Example:

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

JavaScript Conditions



JavaScript Loops

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

Example:

JavaScript + No-Library (pure JS) ▼

```
var i;
for (i=0; i<5; i++)
{
  document.write("Number is: " + i);
  document.write("<br />");
}
```

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) ▼

```
var i=0;
while (i<5)
{
  document.write("Number is: " + i);
  document.write("<br />");
  i=i+1;
}
```

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:

```
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=/";
}
```

Cookies store user/visitor information

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

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

```html
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript OOP</h1>

<p>Creating JS objects from a class</p>

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

<script>
    class Names {
        constructor(firstName, lastName) {
            this.firstName = firstName;
            this.lastName = lastName;
        }
    }

    const myName = new Names("Manolis", "Samanis");

    document.getElementById("demo").innerHTML =
        myName.firstName + " " + myName.lastName ;
</script>

</body>
</html>
```

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

```javascript
let products = {};

products.apples = 1.5;
products.bananas = 2.5;
products.oranges = 1.1;
products.pears = 1.9;

for(key in products) {
    console.log(key, products[key]);
}
```

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

The date object is used for managing dates and times

Example:

```html
<div id = "timer"></div>
<script>
  var d = new Date();
  var t = document.getElementById("timer");
  t.innerHTML = d.getHours();
</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);**

```
var btn = document.getElementById("myButton");
btn.addEventListener("click", getvalue);

function getvalue() {
  ...
}
```

JavaScript Custom Objects with functions

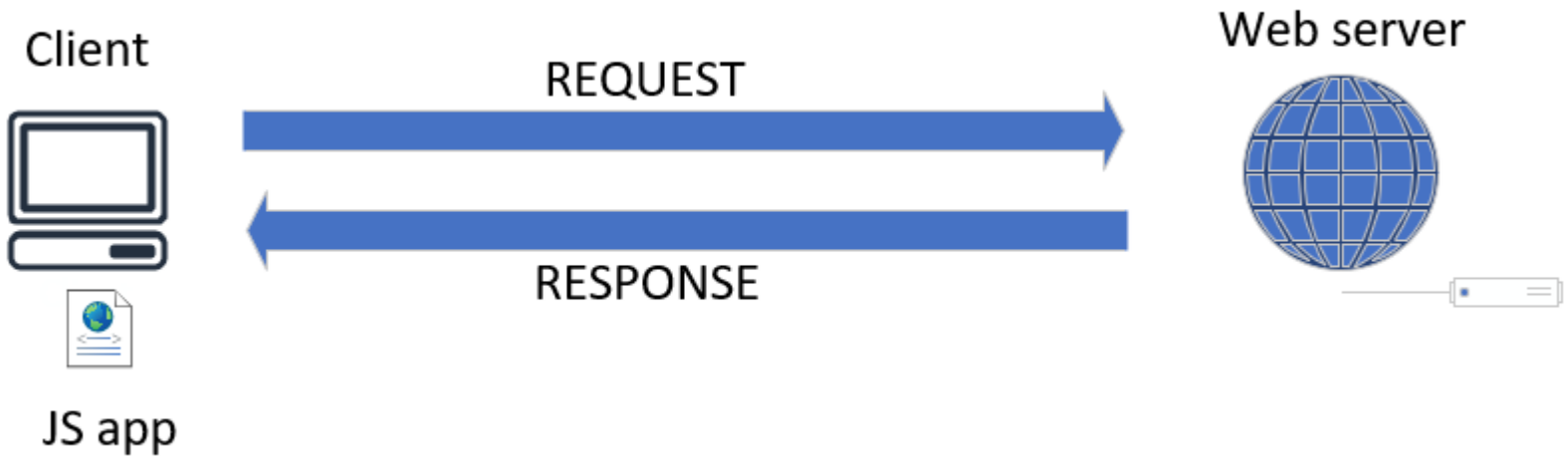
car_custom_objects.html

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Custom Objects with functions</h2>

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

<button id="displayButton">What are my car details?</button>

<div id="output"></div>

<script>
    var myObject = {
        make: "Corsa",
        year: "2015",
        price: 5000,
        color: "blue"
    };

    no usages
    function displayObjectValue() {
        var outputDiv = document.getElementById("output");
        outputDiv.innerHTML = "make: " + myObject.make + "<br>" + "year: " + myObject.year + "<br>" +
            "price: " + myObject.price + "<br>" + "color: " + myObject.color;

    }
    document.getElementById("displayButton").addEventListener("click", displayObjectValue);
</script>



</body>
</html>
```

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

AJAX.html

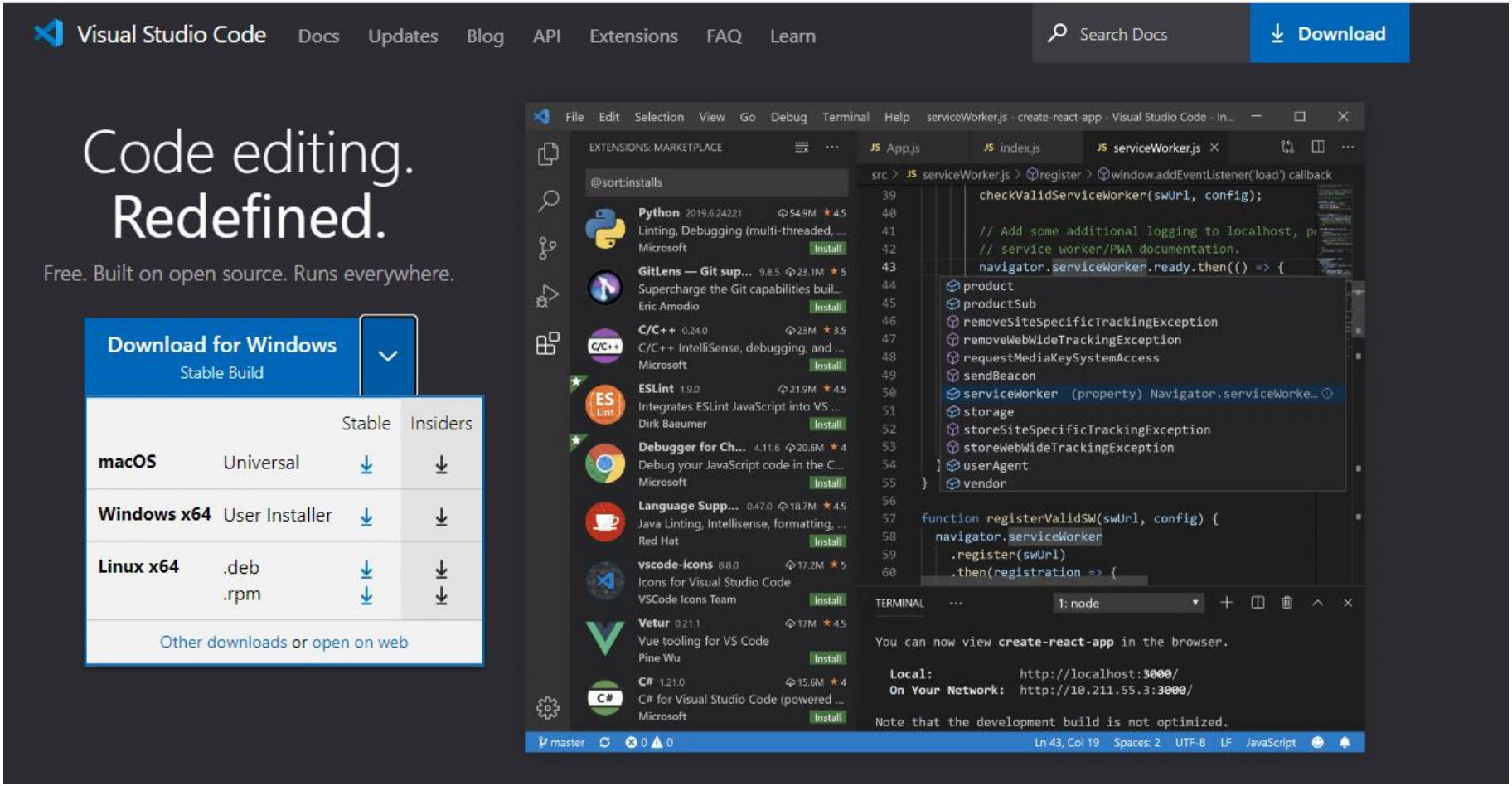
```html
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8"…
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>A simple AJAX Request Example</title>
</head>

<body>

<h2>AJAX request Demo</h2>

<button id="loadData">Load external Data</button>

<div id="output"></div>

<script>
    document.getElementById('loadData').addEventListener('click',function() {
        var xhr = new XMLHttpRequest(); // we create here the XMLHttpRequest object
        xhr.open('GET', 'https://jsonplaceholder.typicode.com/posts/1', true); // we specify the request type and the URL we need
        xhr.onreadystatechange = function () { // here define the callback function to handle the response
            if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) { // check for the request if successful
                document.getElementById('output').innerHTML = xhr.responseText; // we update the content of the output div with the response text
            }
        };
        xhr.send(); // here we send the request
    });
</script>


</body>
</html>
```

Suggested resources for further reading

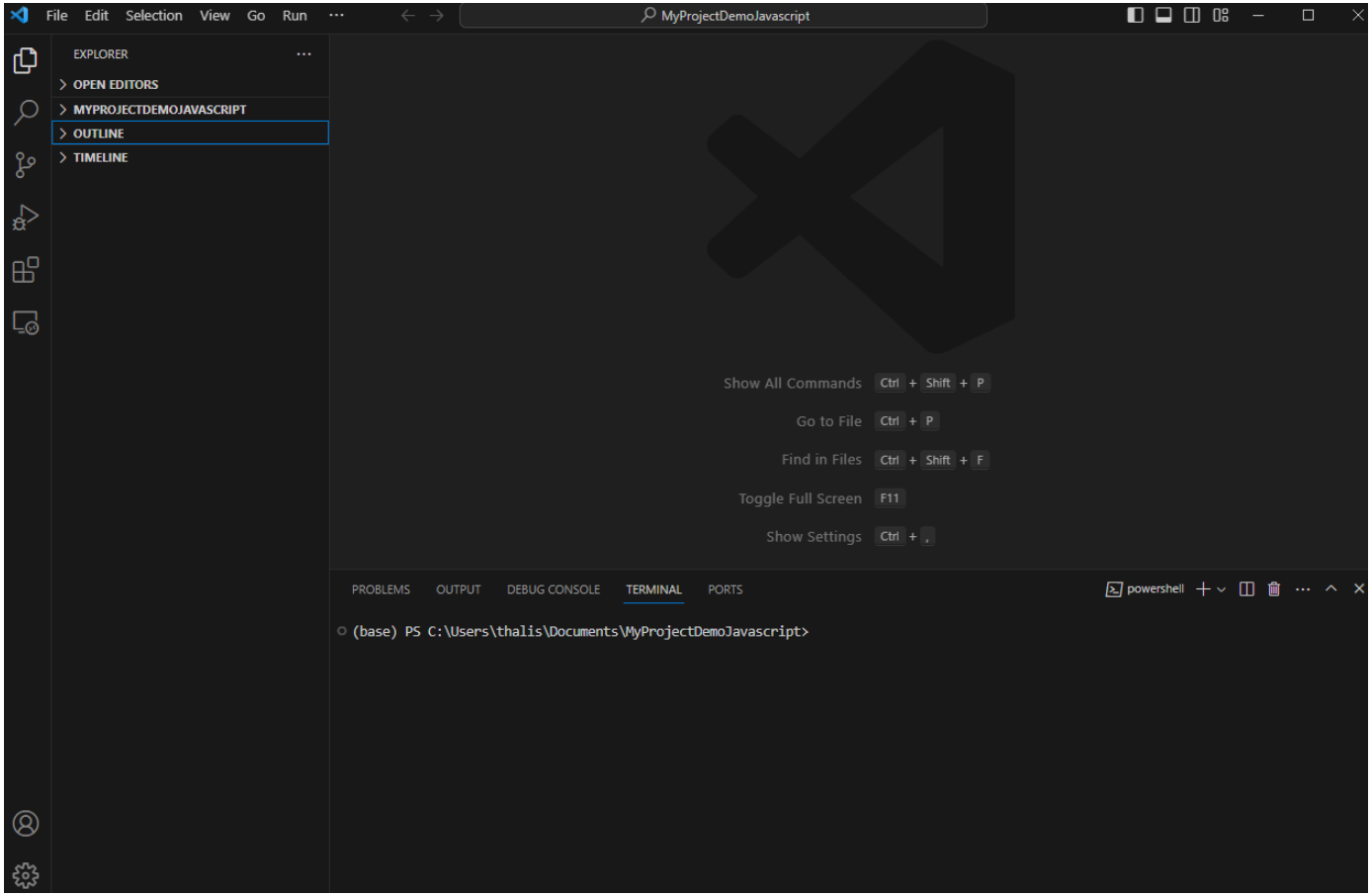
https://developer.mozilla.org/en-US/docs/Learn/JavaScript

JavaScript: The Definitive Guide, Author: David Flanagan

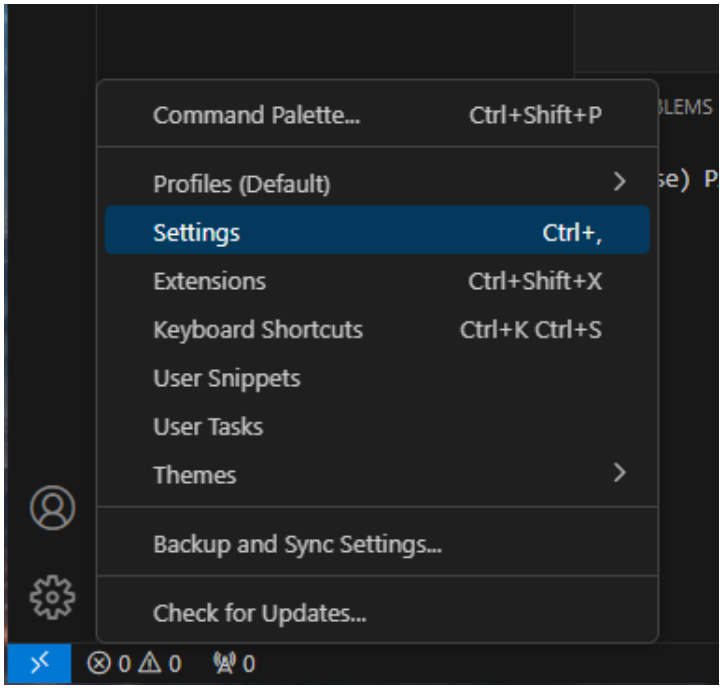
JS editor VS code

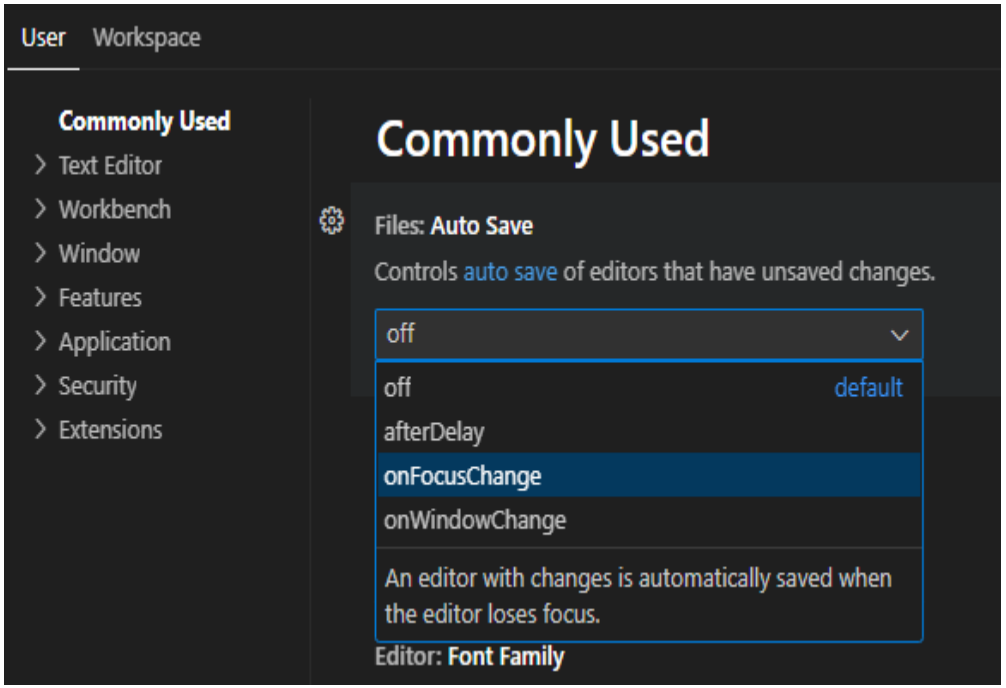


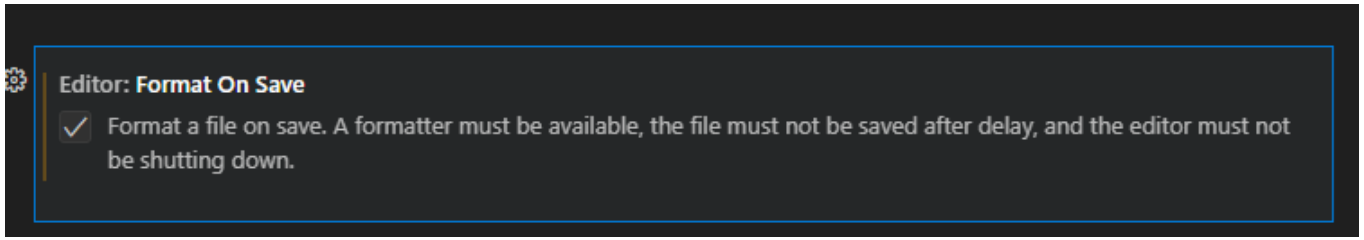
JS editor VS code



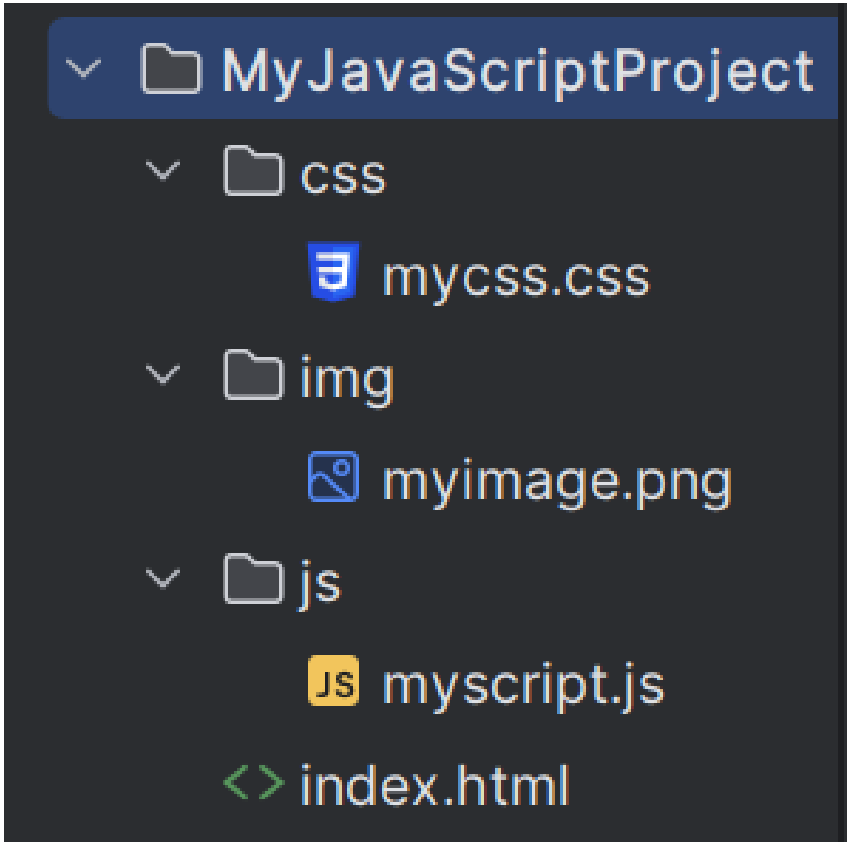
JS editor VS code







How to organize your project folder and file structure



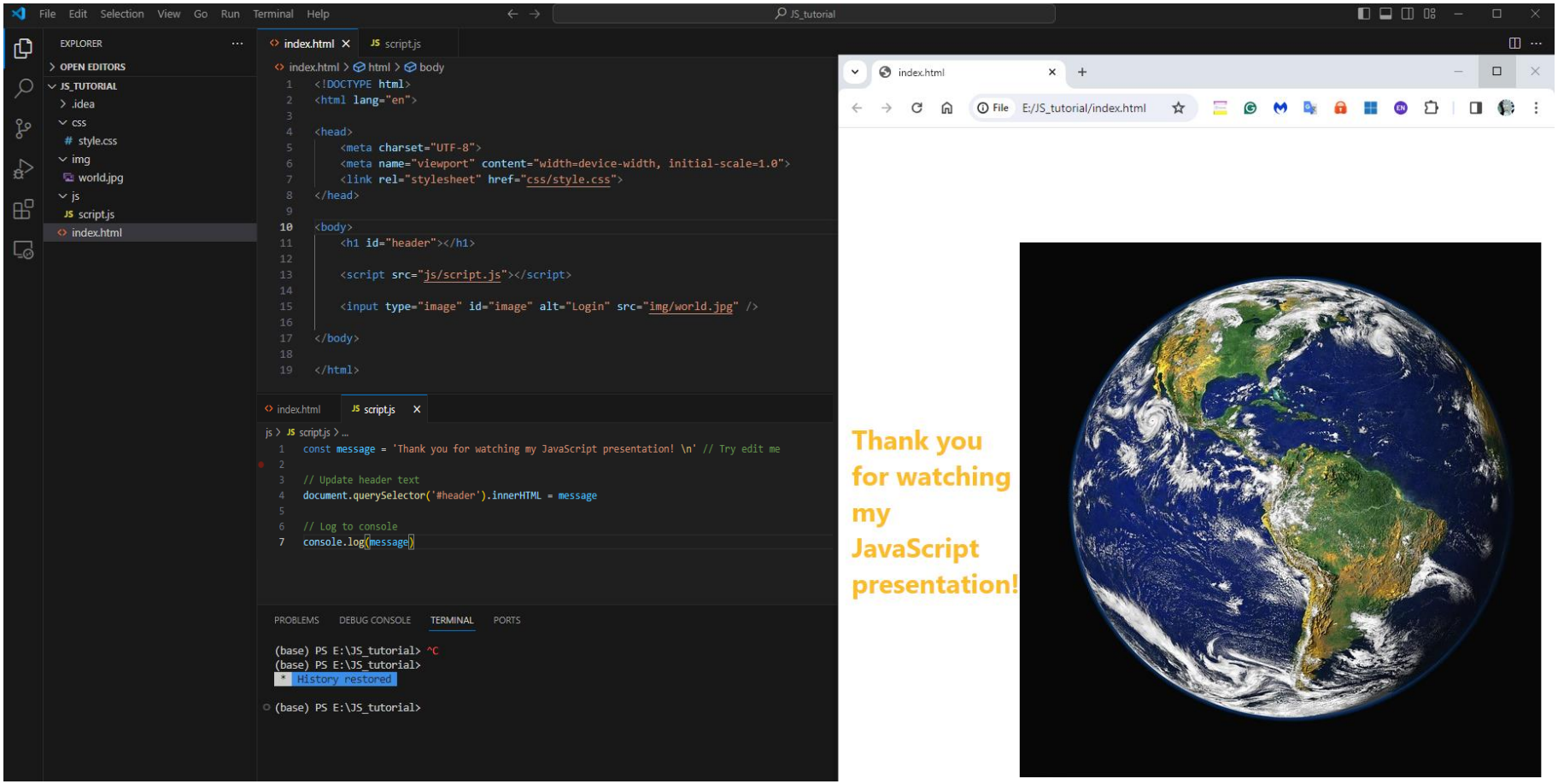

```
<link rel="stylesheet" type="text/css"
href="css/mycss.css">
```

```
<img src="img/myimage.jpg"></img>
```

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

Thank you for watching my JavaScript presentation

JS editor VS code



Have any questions?