

2.1 Introduction to Javascript and Web Development

R E A D Y

The Relationship among HTML, CSS, and JavaScript

The most important three languages for creating web frontends are certainly HTML, CSS, and JavaScript. Each of these languages serves its own purpose.



Use HTML elements to specify the structure of a web page and the meaning (semantics) of individual components on a web page. For example, they describe which area on the web page is the main content and which area is used for navigation, and they define components such as forms, lists, buttons, input fields, or tables, as



CSS uses special CSS rules to determine how the individual components that you have previously defined in HTML should be displayed; this is used to define the design and layout of a web page. For example, you can define text color, text size, borders, background colors, color gradients, and so on

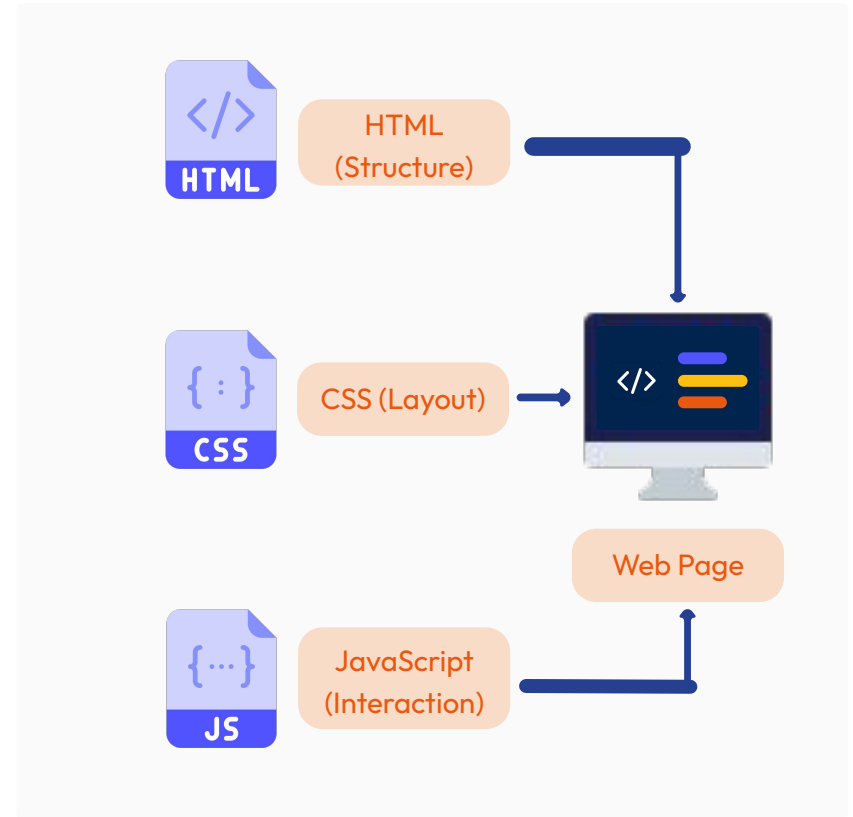


JavaScript is used to add dynamic behavior to the web page (or to the components on a web page) or to provide more interactivity on the web page. Examples of this are sorting and filtering the table data

Structure of a web page

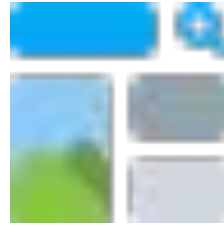
Usually, a of HTML, CSS, and JavaScript Is Used within a Web Page

You can create functional web pages entirely without JavaScript or even CSS, that means, however, that the web page is less fancy (without CSS) and less interactive and user-friendly (without JavaScript)

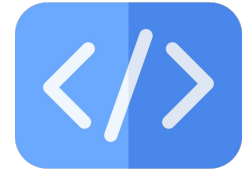


Phases of Website Development

When developing professional websites, there are several stages preceding the development step



Before development even begins, prototypes are designed in concept and design phases (either digitally or quite classically with pen and paper)



The step-by-step approach just described (first HTML, then CSS, then JavaScript) thus only refers to development

Text Editors

When writing JavaScript you should acquire a good text editor that is specifically designed for developing JavaScript programs

For example, Sublime Text (www.sublimetext.com) is available for Windows, macOS, and Linux, are popular editors in the developer community

Editor supports highlighting the source text in color, relieving you of writing recurring source text modules, recognizing errors in the source text, and much more

Integrated Development Environments

Software developers switching from languages like Java or C++ to JavaScript are in most cases used to integrated development environments (IDEs).

An IDE as a very powerful editor that provides various additional features compared to a "normal" editor, such as synchronization with a source control system, running automatic builds, or integrating test frameworks

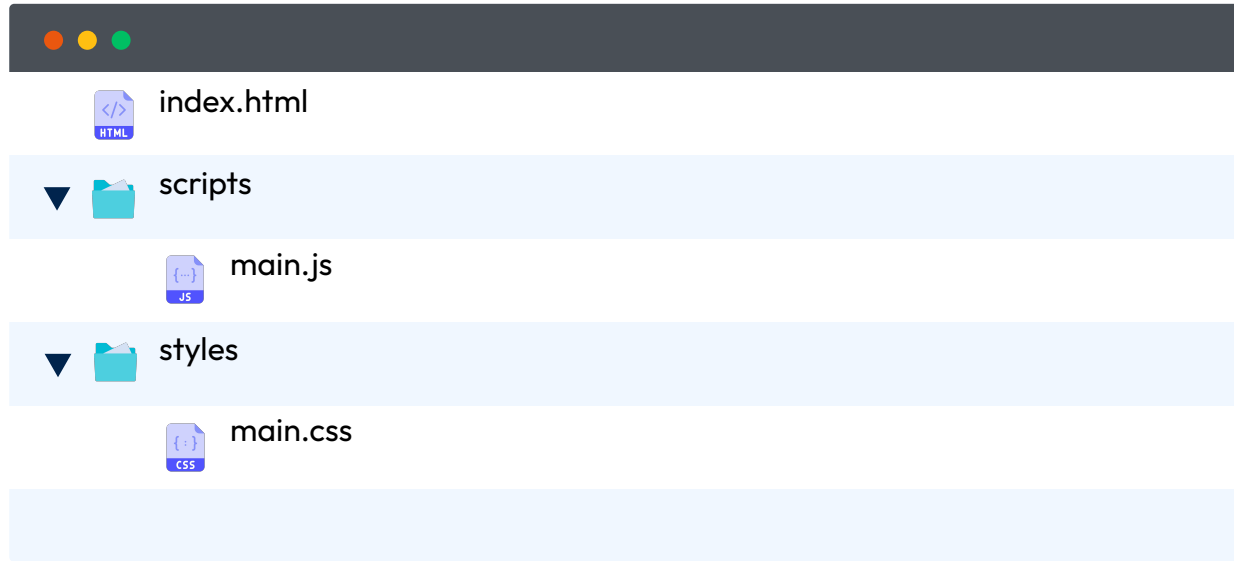
WebStorm by IntelliJ (www.jetbrains.com/webstorm/) and Visual Studio Code by Microsoft (<https://code.visualstudio.com>) are very popular and also very good development environments

2.2 Integrating JavaScript into a Web Page

R E A D Y

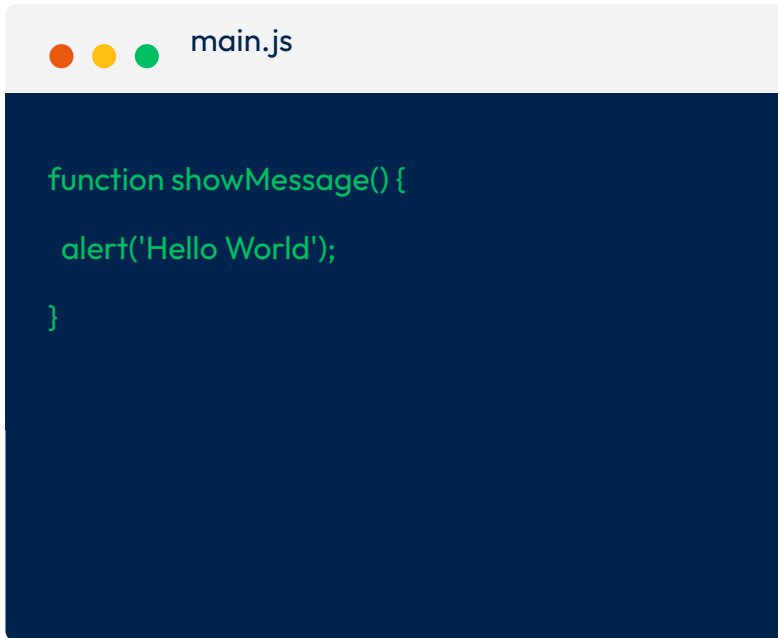
Folder Structure

For getting started and working through the following examples, it is a good idea to create different folders for the CSS and JavaScript files and leave the HTML file at the top level



Folder Structure

Create a new file, enter the lines of source code, and then save the file under the name "main.js" All JavaScript files have the extension .js



```
function showMessage() {  
    alert('Hello World');  
}
```

The code in main.js defines a function with the name showMessage, which in turn calls another function (with the name alert) and passes it the message "Hello World"

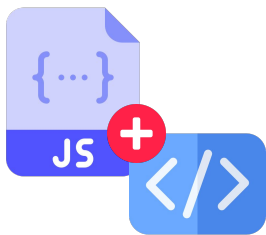
The alert function is a JavaScript standard function, which we will briefly discuss later in this chapter

Embedding JavaScript

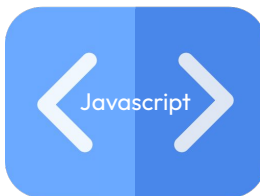
To use the JavaScript source code within a web page, you need to link the JavaScript file to the web page or embed the JavaScript file in the HTML file

JavaScript is added to an HTML file via the HTML element named `<script>`

This element can be used in two different ways:



External JavaScript files can be included in a `<script>` tag



Writing JavaScript source code directly inside the `<script>` tag

Embedding an External JavaScript File


Embed an external JavaScript file by writing the path to the file in the `src` attribute for the `script` tag

```
index.html

<!DOCTYPE html>
<html>
  <head lang="en">
    <meta charset="UTF-8">
    <title>Example</title>
    <link rel="stylesheet" href="styles/main.css" type="text/css">
  </head>
  <!-- Here the JavaScript file will be included -->
  <body>
    <script src="scripts/main.js"></script>

  </body>
</html>
```

Run the JavaScript File



```
function showMessage() {  
    alert('Hello World');  
}  
showMessage();
```

If you open HTML file in the browser, nothing will happen yet because the function `main.js` is not yet called at any point
Add the `showMessage()` call at the end of the JavaScript file and reload the web page in the appropriate browser

Defining JavaScript Directly within a `<script>` tag

index.html

```
<!DOCTYPE html>
<html>
<head lang="en">...</head>
<body>
<!-- Here the JavaScript will be directly included -->
<script>
  function showMessage() {
    alert('Hello World'); }
  showMessage();
</script>
</html>
```

To define JavaScript directly within an HTML file, write the relevant JavaScript code inside the `<script>` element instead of linking it via the `src` attribute

This is usually not advisable because it means mixing HTML and JavaScript code in one file

The <noscript> Element

You can use the <noscript> element to define an HTML section that is displayed when JavaScript is not supported in the browser or has been disabled by the user

(If JavaScript is supported or enabled, the content of the <noscript> element will not be shown)

index.html

```
<noscript>  
  JavaScript is not available or is disabled. <br />  
  Please use a browser that supports JavaScript,  
  or enable JavaScript in your browser.  
</noscript>
```

Placement of the <script> Elements



```
index.html

<!DOCTYPE html>
<html>
  <head lang="en">
    ... css elements ...
  </head>
  <body>
    <!-- The JavaScript files should be included here -->
    <script src="scripts/main.js"></script>
  </body>
</html>
```

While CSS files are still placed in the <head> area, JavaScript files should be included before the closing </body> tag instead

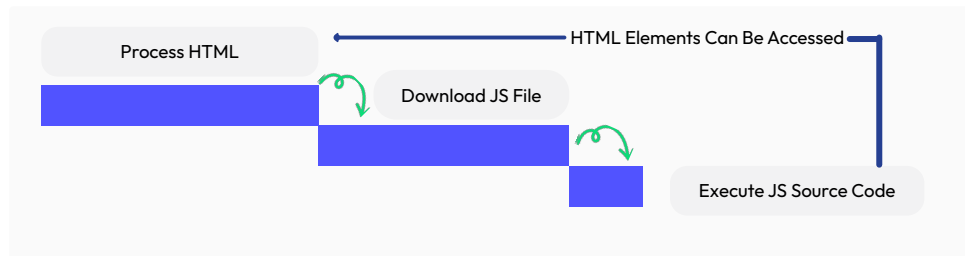
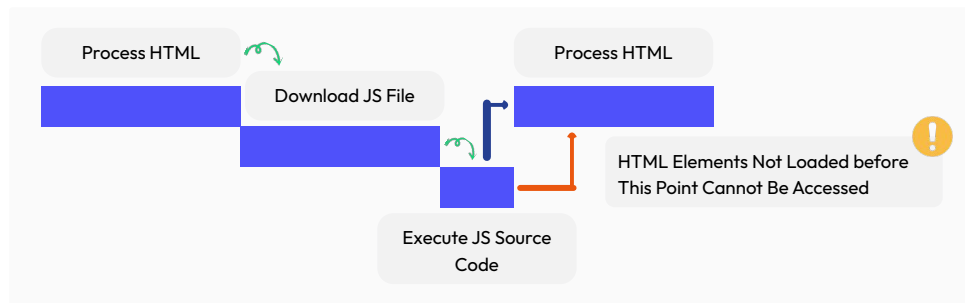
Execution of the `<script>` Elements

By default, HTML processing stops when the browser encounters a `<script>` element

If JavaScript is executed before HTML elements have been processed, you'll get an access error

If the `<script>` element is before the closing `</body>` tag, all elements will have been loaded

When the browser encounters a `<script>` element, it downloads the corresponding JavaScript source code entirely and immediately starts processing the code. This pauses all other processing, which leads to the user impression that it takes longer to build the web page



Displaying the Source Code

All browsers usually provide a way to view the source code of a web page

- Chrome: View → Developer → View Source
- Firefox: Tools → Browser Tools → Page Source
- Safari, Develop → Show Page Source
- Opera, Developer → View Source
- Microsoft Edge, Tools → Developer → View Source

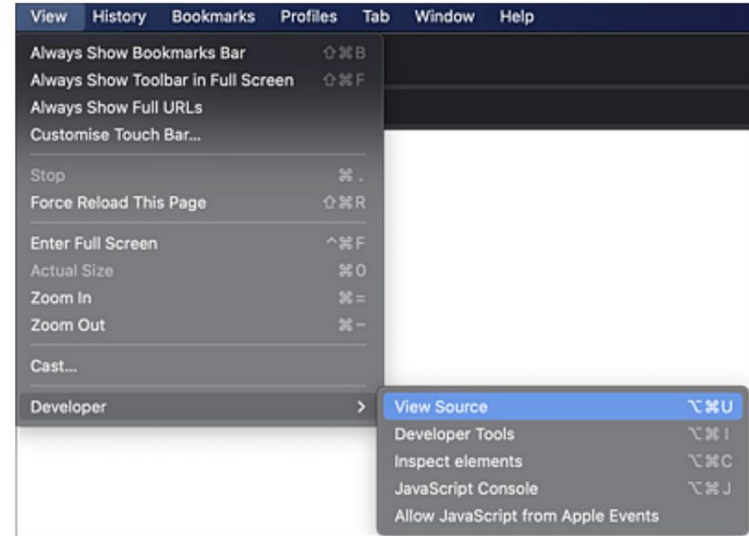


Figure 2.18 Show Source Code in Chrome

2.3 Creating Output

R E A D Y

Dialog Windows

The JavaScript language provides multiple standard functions for displaying dialog boxes to a user

Method	Description
alert()	Displays a hint dialog
confirm()	Displays confirmation dialogs - that is, yes/no decisions
prompt()	Opens an input dialog where users can enter text

```
main.js  
  
alert("This is an alert box");  
  
confirm("This is a confirmation box");  
  
prompt("This is a prompt box");
```

Open the Console

Open your console depending on your browser:

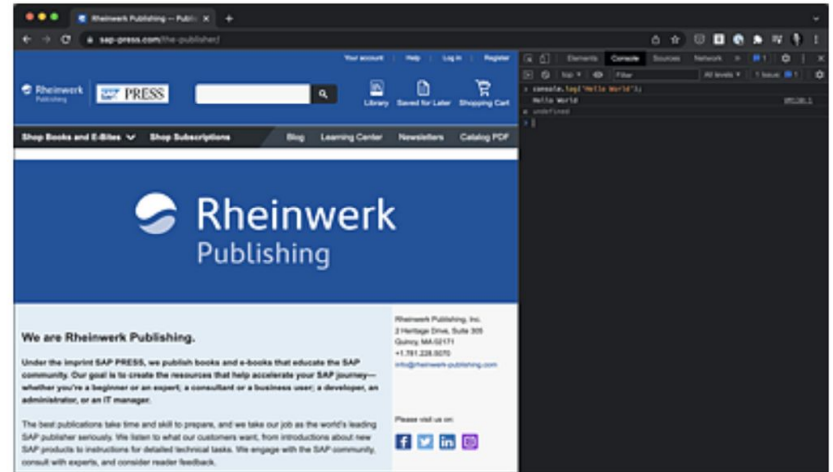
Chrome: View → Developer → JavaScript Console

Firefox: Tools → Browser Tools → Browser Console

Safari: Develop → Show JavaScript Console

Opera: Developer → Developer Tools → Console tab

Microsoft Edge: Tools → Developer → JavaScript Console



Writing Output to the Console

For writing to the console, browsers provide the console object, a generally supported method is the `log()` method, which can be used to generate simple console output

```
console.log("Message");
```



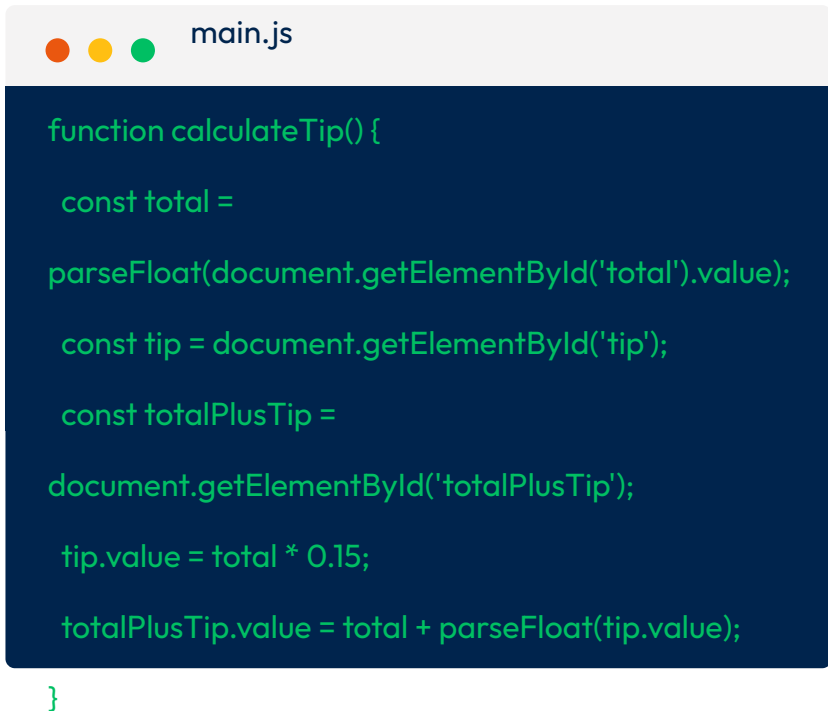
```
function showMessage() {  
  console.log('Hello World');  
}  
showMessage();
```

Console Methods

In addition to the `log()` method, console provides several other methods

Method	Description
<code>clear()</code>	Clears the console
<code>debug()</code>	Used to output a message intended for debugging (or troubleshooting). (You may first need to set the appropriate developer tools to return this type of output.)
<code>error()</code>	Used to output an error message. Some browsers display an error icon next to the output message within the console
<code>info()</code>	This will display an info message in the console. Some browsers—Chrome, for example—also output an info icon
<code>log()</code>	Probably the most commonly used method of console. Generates normal output to the console
<code>trace()</code>	Outputs the stack trace—that is, the method call stack (see also Chapter 3) to the console
<code>warn()</code>	Used to issue a warning to the console. Again, most browsers will display a corresponding icon next to the message

Using Existing UI Components



```
function calculateTip() {  
    const total =  
    parseFloat(document.getElementById('total').value);  
    const tip = document.getElementById('tip');  
    const totalPlusTip =  
    document.getElementById('totalPlusTip');  
    tip.value = total * 0.15;  
    totalPlusTip.value = total + parseFloat(tip.value);  
}
```

You can write the output of a program into existing UI components

Existing UI components can also be used to input data or run JavaScript functions

The example on the next slides shows how JavaScript can be used to interact with the existing HTML UI components to product a dynamic web page

Using Existing UI Components

```
index.html

<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
</head>
<body>
  Bill total: <input id="total" type="input"><br/>
  <button onclick="calculateTip()">Calculate
  tip</button><br/><br/>
  Tip: <input id="tip" type="input"><br/>
  Total + Tip: <input id="totalPlusTip"
  type="input"><br/>
  <script src="scripts/main.js"></script>
</body>
</html>
```

Web Page

Bill total:

Tip:

Total + Tip: