edureka!
a Veranda Enterprise

Full Stack Web Development Program

edureka!
a Veranda Enterprise

Day 17 – JavaScript Essentials

Titles

- Introduction to JavaScript
- Benefits of JavaScript
- Adding JavaScript to HTML file
- Adding <script> tag in <head> section
- Adding <script> tag in <body> section
- Including External JavaScript File
- Life without JavaScript
- Variables
- Scope of Variables
- Data Types in JavaScript
- Hoisting in JavaScript

Learning Objectives

By the end of this module, you will be able to:

- Recall the concepts of JavaScript, including its benefits, adding JavaScript to HTML files, the different ways to include the <script> tag, and understanding life without JavaScript.
- Comprehend the concept of variables, the scope of variables, and the different data types in JavaScript.

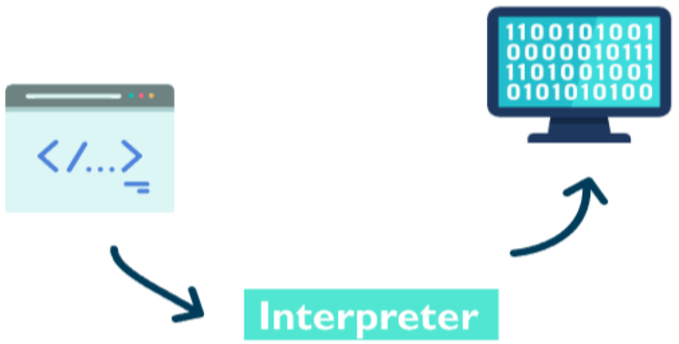
The Transformative Power of JavaScript

edureka!

Introduction to JavaScript

JavaScript is a high-level, interpreted programming language that makes web pages more interactive.

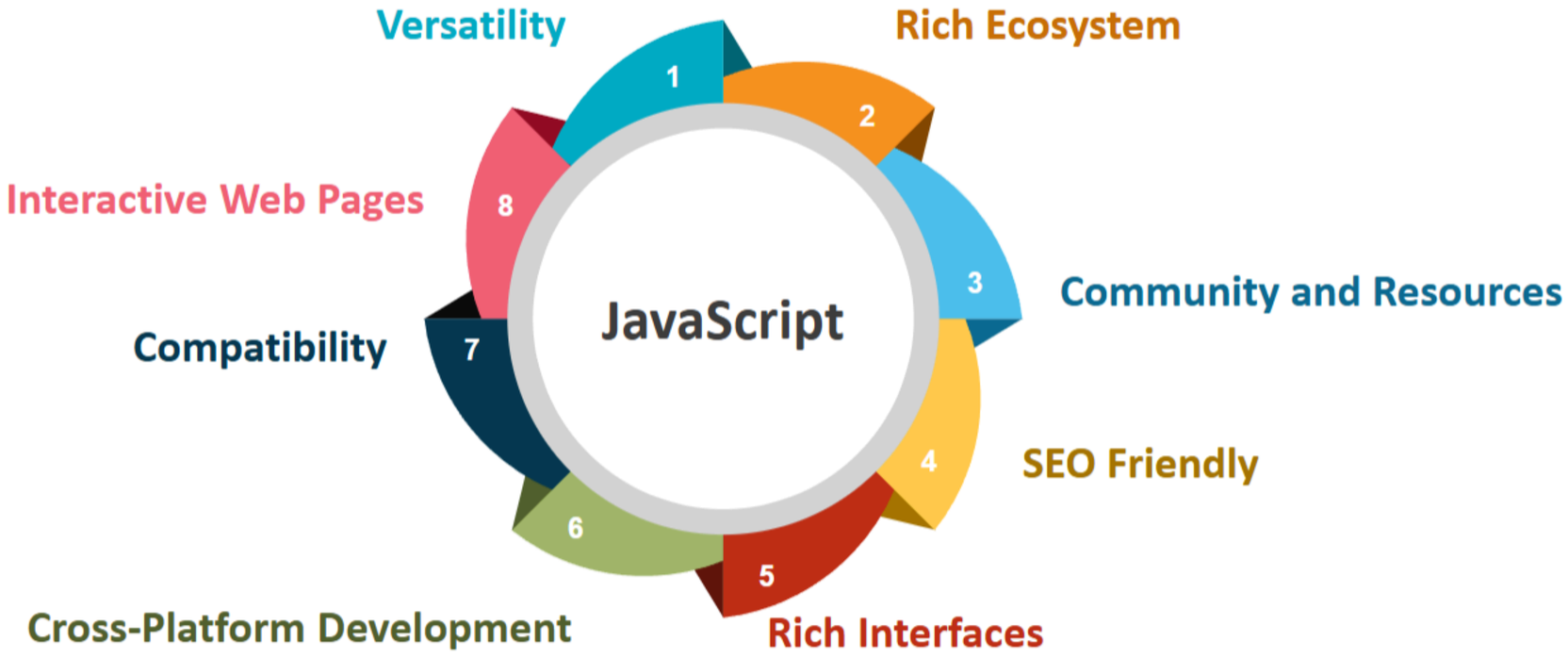
Interactive Web Pages



Interpreted Language

Runs on the Client's System

Benefits of JavaScript



Adding JavaScript to HTML file

JavaScript provides flexibility to place the code anywhere in an HTML document.

However, some of the most preferred ways to include JavaScript in an HTML file are:

- Script in **<head>..</head>** section
- Script in **<body>..</body>** section
- Script in **<body>..</body>** and **<head>..</head>** sections
- Script written in external file included in **<head>..</head>** section

Adding <script> tag in <head> section

- If you want to run the script on the **occurrence of some event** (suppose when a user clicks on a button), then you will place the script in the head as follows:

```
<html>
      <head>
        <script
            type="text/javascript">
            function printHelloWorld()
            {
                alert("Hello World! Welcome to JavaScript")
             }
        </script>
      </head>
      <body>
         <input type="button" onclick= “printHelloWorld()” value="Print Hello" />
      </body>
    </html>
```

Adding <script> tag in <body> section

- If you want a script to run on page load, then it should be included in the **<body>** section of the document.

```
<html>
  <head>
  </head>
  <body>
   <script type="text/javascript">
       document.write("Welcome user!")
   </script>

   <p> This is the webpage body </p>
 </body>
</html>
```

Including External JavaScript File

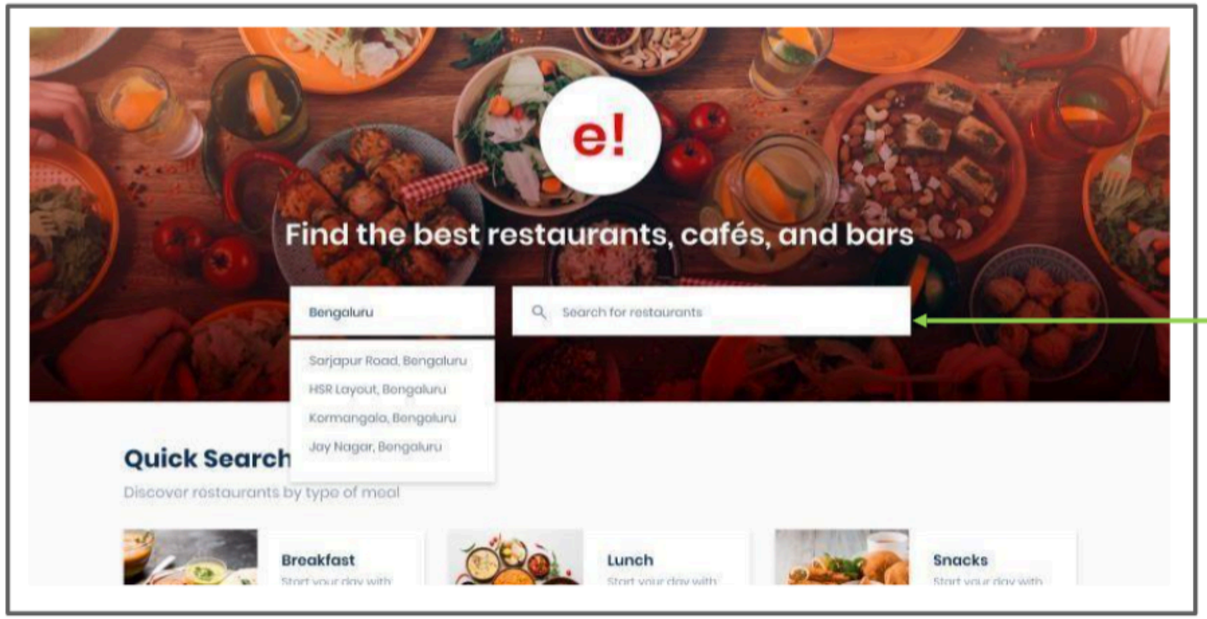
- A common practice to separate JavaScript code from HTML markup, enhancing maintainability, readability, and reusability. This is achieved by using the **<script>** tag in HTML with the **src** attribute set to the URL or path of the JavaScript file.

```
<html>
  <head>
    <script type = "text/javascript" src = "file_name.js">
    </script>

  </head>
  <body>
    <p> This is the body </p>
  </body>
</html>
```

Life without JavaScript

- Until now, you have made a static webpage with only HTML elements. The images, dropdowns, and buttons are unresponsive, and no action is performed on clicking or hovering over the different elements.



There is no response on clicking this element

Life without JavaScript (contd.)

What are the drawbacks/ restrictions of the webpage created till now?

- It is a static unresponsive HTML with images, text, buttons, and other UI features.
- There is no use of such web pages from the perspective of a business or the customers.

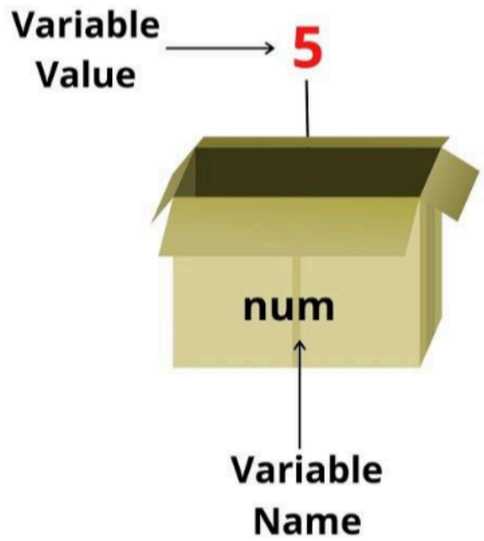
This is where JavaScript comes into the picture. It provides the following advantages:

- It is very fast because any code can run immediately instead of having to contact the server.
- It allows you to create highly responsive interfaces to improve the user experience.
- JavaScript has no compilation step. An interpreter in the browser reads and executes the JavaScript code line by line.
- It provides dynamic functionality, allowing updates on the page without waiting for the server to respond and load a new page.

Variables

A variable is a name given to a memory location that acts as a container for storing data temporarily. They are reserved memory locations to store values.

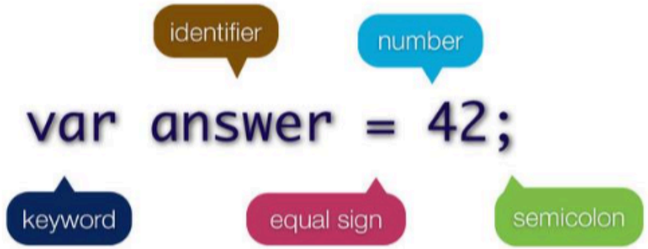
```
var num = 5;
```



Explanation of variable in pictorial format

Variables (contd.)

- In JavaScript, each variable must be uniquely identified by its name, known as an **Identifier**.
- When declaring a variable in JavaScript, certain fundamental rules apply:
  - They are case-sensitive.
  - A variable name must start with either a letter, an underscore ("_"), or the "**$**" character.
  - The name can include letters, digits, underscores, or the "**$**" symbol.
  - It's important to avoid using reserved keywords of JavaScript as variable names.



Variables (contd.)

- JavaScript is a language with dynamic typing, meaning the types of variables are determined during execution.
- Hence, there's no requirement to specify the variable type beforehand.
- Variables in JavaScript can be declared using one of three methods:
  - Using the **`var`** keyword in JavaScript
  - Using the **`let`** keyword in JavaScript
  - Using the **`const`** keyword in JavaScript

```
var greet= "Hello Learner"        // Declaration using var
let _variable = "Welcome"                 // Declaration using let
const $msg = “to Edureka"       // Declaration using const
```

Variables (contd.)

Using var Keyword

- **var** is the oldest keyword used for variable declarations in JavaScript.
- It has function scope when declared within a function and global scope when declared outside a function.
- Variables declared with **var** are also hoisted, meaning their declaration is moved to the top of their scope.

Example:

```
function exampleFunction() {
    if (true) {
        var x = 5; // Function scope
    }
    console.log(x); // Outputs 5,
because 'x' is accessible within the
entire function
}
exampleFunction();
console.log(x); // Error, 'x' is not
defined outside the function
```

Variables (contd.)

Using let Keyword

- Introduced in **ES6 (ECMAScript 2015)**, **`let`** allows you to declare variables with block scope, which is limited to the block, statement, or expression where it's used.

- Unlike **`var`**, **`let`** does not hoist the variable.

Example:

```
function exampleFunction() {
    if (true) {
        let y = 10; // Block scope
        console.log(y); // Outputs 10
    }
    console.log(y); // Error, 'y' is not
accessible outside the block
}
exampleFunction();
```

Variables (contd.)

Using const Keyword

- Also introduced in ES6, **`const`** is used to declare constants.
- A constant in JavaScript means that the variable's identifier cannot be reassigned.
- **`const`** has block scope like **`let`**.

Example:

```
const z = 15;
console.log(z); // Outputs 15
z = 20; // Error, cannot reassign a constant
```