



Introduction To Javascript For Beginners

Building Interactivity On The Web

INDEX

- ▶ **What is JavaScript?**
- ▶ **How JavaScript Works**
- ▶ **Basic Syntax**
- ▶ **Control Flow**
- ▶ **Functions**
- ▶ **DOM Manipulation**
- ▶ **Events**
- ▶ **Example: Simple JavaScript Program**
- ▶ **Resources for Learning JavaScript**

What is JavaScript?

Definition of JavaScript:

► Definition of JavaScript:

JavaScript is a high-level, interpreted programming language primarily used for adding interactivity and dynamic behavior to web pages. It runs on the client-side, meaning it's executed in the user's web browser rather than on a remote server.

Importance Of Javascript In Web Development

- ▶ Javascript Is An Essential Component Of Modern Web Development. It Allows Developers To Create Interactive Elements, Handle User Input, Manipulate Web Page Content, And Communicate With Web Servers Asynchronously, Enhancing The Overall User Experience.

How JavaScript Works

► Explanation of Client-Side Scripting:

JavaScript is commonly referred to as a client-side scripting language because it's executed on the client's computer (in the browser). This allows for immediate feedback and interaction with the user without needing to communicate with the server for every action.

► Interaction with HTML and CSS:

JavaScript interacts with HTML (the structure) and CSS (the presentation) of a web page, enabling developers to dynamically modify the content and appearance of web pages based on user actions and other events.

Basic Syntax

Variables and Data Types:

► Data Types in JavaScript:

► Primitive Data Types:-

- **Number:-** Represents numeric values (e.g., integers, floats).
- **String:** Represents textual data enclosed in single or double quotes.
- **Boolean:** Represents true or false values.
- **Null:** Represents the intentional absence of any value.
- **Undefined:** Represents a variable that has been declared but not assigned a value.

► Variables:

❑ Var Keyword:

- Traditional variable declaration.
- Function or globally scoped.
- Allows redeclaration and updating.

❑ Let Keyword:

- Introduced in ES6.
- Block-scoped.
- Allows reassignment within block scope.

❑ Const Keyword:

- Also introduced in ES6.
- Block-scoped.
- Constants cannot be reassigned but can be modified for objects and arrays.

Control Flow

► Conditional Statements (if-else):

- Conditional statements allow developers to execute different blocks of code based on specified conditions. The if-else statement is commonly used to control the flow of execution in JavaScript.

► Loops (for, while):

- Loops are used to repeat a block of code multiple times. JavaScript provides different types of loops, including for, while, and do-while, which allow developers to iterate over arrays, perform repetitive tasks, and handle iterative operations.

Functions

► Definition of Functions:

- Functions in JavaScript are reusable blocks of code that perform a specific task. They help in organizing code, improving code readability, and reducing redundancy. Functions can accept input parameters (arguments) and return output values. Parameters and Return Values: Parameters are placeholders for values that are passed to a function when it's called. Return values are the result of executing a function, which can be used elsewhere in the code.

DOM Manipulation

► Introduction to the Document Object Model (DOM):

- The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of an HTML document as a hierarchical tree of objects, allowing JavaScript to access and manipulate elements, attributes, and content dynamically.

► Modifying HTML Elements with JavaScript:

- JavaScript can modify the content, attributes, and styling of HTML elements on a web page by accessing and manipulating the DOM. This enables developers to create dynamic and interactive user interfaces.

Events

► Handling User Interactions (Clicks, Input):

- Events are actions or occurrences that happen in the browser, such as clicks, mouse movements, keypresses, or form submissions. JavaScript allows developers to handle these events and execute specific code in response, enabling interactivity and user engagement.

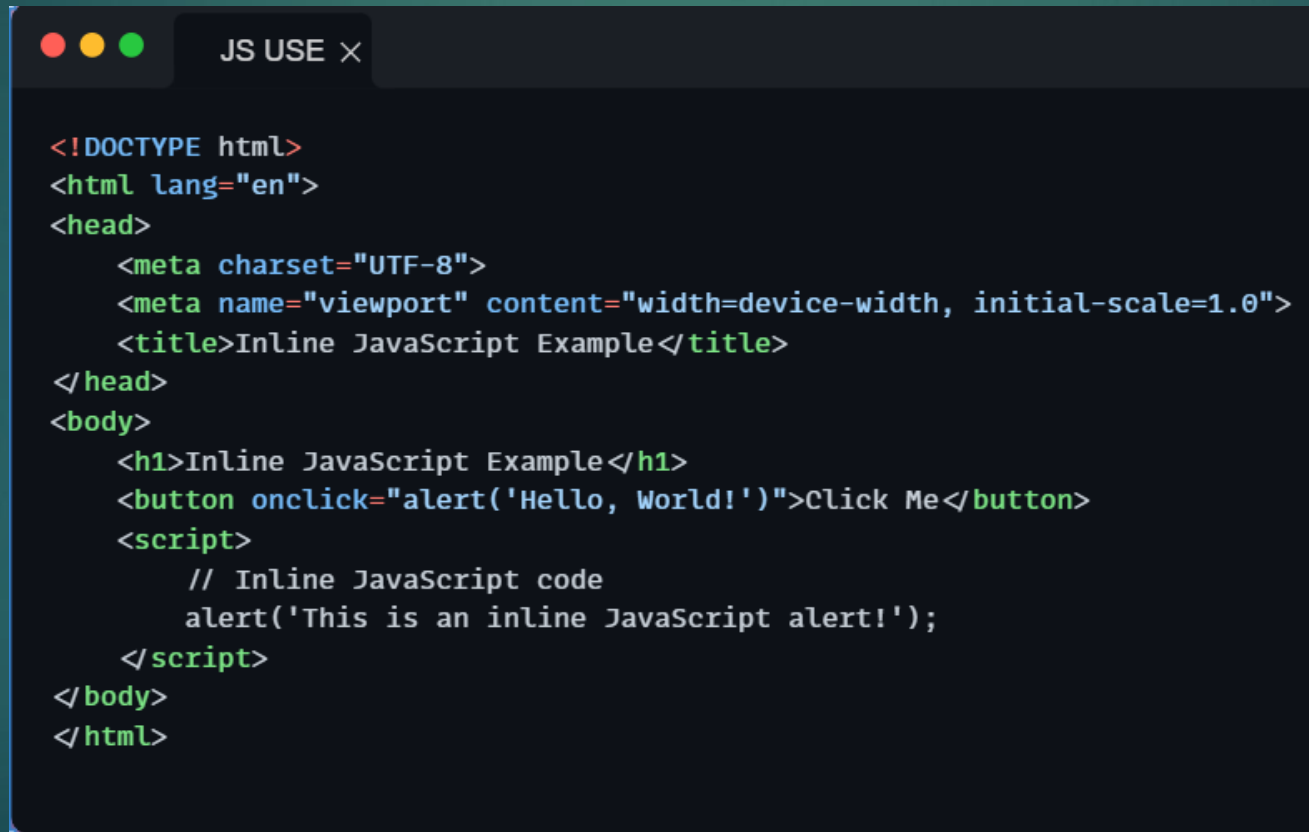
► Event Listeners and Event Handling:

- Event listeners are JavaScript functions that listen for specific events on HTML elements and trigger corresponding actions or behaviors. Event handling involves attaching event listeners to elements and defining callback functions to handle events when they occur.

Javascript using types

► Inline JavaScript:

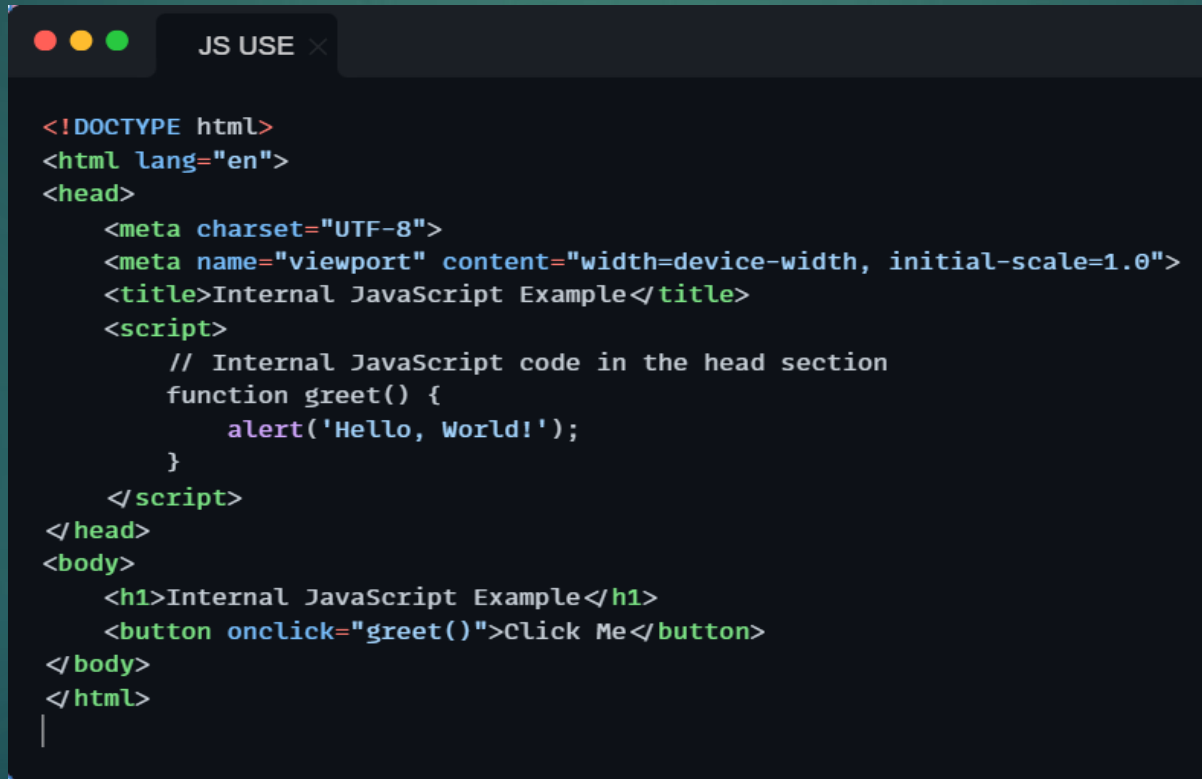
- JavaScript code can be directly embedded within HTML documents using the `<script>` element. This method is useful for small scripts or quick tasks.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Inline JavaScript Example</title>
</head>
<body>
  <h1>Inline JavaScript Example</h1>
  <button onclick="alert('Hello, World!')">Click Me</button>
  <script>
    // Inline JavaScript code
    alert('This is an inline JavaScript alert!');
  </script>
</body>
</html>
```

- ▶ Internal JavaScript:

- JavaScript code can be placed within the `<script>` tags in the `<head>` or `<body>` sections of the HTML document. This method allows you to separate HTML and JavaScript code within the same file. For example:



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Internal JavaScript Example</title>
  <script>
    // Internal JavaScript code in the head section
    function greet() {
      alert('Hello, World!');
    }
  </script>
</head>
<body>
  <h1>Internal JavaScript Example</h1>
  <button onclick="greet()">Click Me</button>
</body>
</html>
```

- External JavaScript:

- ▶ JavaScript code can be placed in an external .js file and linked to the HTML document using the `<script>` tag's `src` attribute. This method allows for better code organization and reusability. For example:

```
JS USE X
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>External JavaScript Example</title>
  <script src="script.js"></script>
</head>
<body>
  <h1>External JavaScript Example</h1>
  <button onclick="greet()">Click Me</button>
</body>
</html>
|
```

```
JS USE X
// External JavaScript code in script.js
function greet() {
  alert('Hello, World!');
}
|
```



THANK YOU

.