Course Outline

Week 1: HTML Basics

1. **Introduction to HTML**
   - Structure of an HTML document
   - Basic tags and attributes
   - Document structure (`<html>`, `<head>`, `<body>`)
2. **HTML Text and Formatting**
   - Headings, paragraphs, and text formatting
   - Lists (ordered and unordered)
   - Links and images
3. **HTML Forms and Inputs**
   - Creating forms
   - Form elements: text fields, checkboxes, radio buttons, etc.
   - Form attributes and validation

Week 2: CSS Basics

4. **Introduction to CSS**
   - Inline, internal, and external CSS
   - CSS selectors and properties
   - Basic styling (colors, fonts, margins, paddings)
5. **CSS Layout Techniques**
   - Box model
   - Positioning (static, relative, absolute, fixed)
   - Flexbox basics
6. **Responsive Design with Media Queries**
   - Media queries syntax and usage
   - Creating responsive layouts
   - Mobile-first design principles

Week 3: Bootstrap Basics

7. **Introduction to Bootstrap**
   - Overview of Bootstrap framework
   - Adding Bootstrap to a project
   - Basic Bootstrap components
8. **Bootstrap Grid System**
   - Understanding the grid system
   - Creating responsive layouts with rows and columns
   - Grid offsets and nesting
9. **Bootstrap Components**
   - Navbar, buttons, forms, cards, modals
   - Customizing Bootstrap components
   - Using Bootstrap utilities

Week 4: JavaScript Basics

10. **Introduction to JavaScript**
    - JavaScript syntax and basic operations
    - Variables, data types, and operators
    - Functions and scope
11. **DOM Manipulation**
    - Selecting and modifying DOM elements
    - Event handling
    - Dynamic content updates
12. **JavaScript Control Structures**
    - Conditional statements (`if`, `else if`, `switch`)
    - Loops (`for`, `while`, `do-while`)
    - Error handling with `try-catch`

Week 5: Advanced JavaScript

13. **JavaScript Objects and Arrays**
    - Object creation and manipulation
    - Array methods and iteration
    - JSON and parsing
14. **Asynchronous JavaScript**
    - Understanding callbacks
    - Promises and `async/await`
    - Fetch API for HTTP requests
15. **JavaScript ES6+ Features**
    - Arrow functions
    - Template literals
    - Destructuring and spread/rest operators

Week 6: Introduction to PHP

16. **PHP Basics**
    - PHP syntax and variables
    - Data types and operators
    - Basic PHP functions
17. **PHP Control Structures**
    - Conditional statements (`if`, `else`, `switch`)
    - Loops (`for`, `while`, `do-while`)
    - Error handling
18. **Working with Forms in PHP**
    - Handling form submissions
    - Validating and sanitizing input
    - Sessions and cookies

Week 7: PHP and Databases

19. **Introduction to MySQL**
    - Basics of MySQL and SQL
    - Connecting to a MySQL database
    - Performing CRUD operations (Create, Read, Update, Delete)
20. **PHP and MySQL Integration**
    - Executing SQL queries in PHP
    - Fetching and displaying data
    - Using prepared statements for security
21. **PHP File Handling**
    - Reading from and writing to files
    - Uploading files via PHP
    - File manipulation

Week 8: Laravel Basics

22. **Introduction to Laravel**
    - Setting up Laravel environment
    - Basic Laravel structure and MVC architecture
    - Routing and controllers
23. **Laravel Views and Blade Templating**
    - Working with Blade templates
    - Passing data to views
    - Layouts and components
24. **Laravel Eloquent ORM**
    - Introduction to Eloquent ORM
    - Defining models and relationships
    - Performing CRUD operations with Eloquent

Week 9: Laravel CRUD Operations

25. **Creating and Storing Data**
    - Building forms for data entry
    - Validating input
    - Storing data in the database
26. **Reading and Displaying Data**
    - Retrieving data from the database
    - Displaying data in views
    - Pagination
27. **Updating and Deleting Data**
    - Building forms for data updates
    - Updating records in the database
    - Deleting records

Week 10: Server Administration

28. **Introduction to Web Servers**

    - Overview of web server concepts
    - Apache vs. Nginx
    - Basic server configuration
29. **Deploying a Web Application**
    - Deploying Laravel applications to a server
    - Managing environment variables
    - Setting up database connections
30. **Server Security and Maintenance**
    - Basic security practices (firewalls, SSL/TLS)
    - Monitoring server performance
    - Regular maintenance tasks

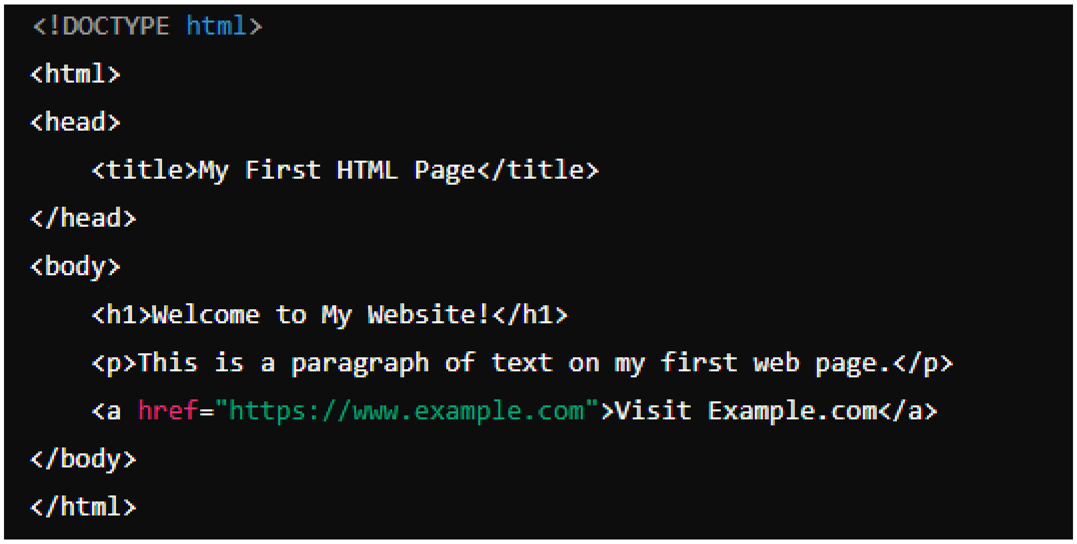
This outline provides a structured approach to learning web development and server management, with practical applications and hands-on projects throughout the course.

Class 1: Introduction to HTML

HTML stands for HyperText Markup Language. It’s the standard language used to create and design web pages. HTML structures the content on the web by using a system of tags and elements to format text, images, links, and other media.

Basic Structure of an HTML Document

Here’s a simple example of a basic HTML document:



```html
<!DOCTYPE html>
<html>
<head>
    <title>My First HTML Page</title>
</head>
<body>
    <h1>Welcome to My Website!</h1>
    <p>This is a paragraph of text on my first web page.</p>
    <a href="https://www.example.com">Visit Example.com</a>
</body>
</html>
```

Explanation of the Example:

1. **`<!DOCTYPE html>`**: This declaration defines the document type and version of HTML (HTML5 in this case).

2. **`<html>`**: This is the root element of an HTML document. All other elements are nested inside it.
3. **`<head>`**: Contains meta-information about the HTML document, such as the title and links to stylesheets. It's not displayed directly on the page.
4. **`<title>`**: Sets the title of the web page, which appears in the browser's title bar or tab.
5. **`<body>`**: Contains the content of the web page that is displayed to users.
6. **`<h1>`**: Defines a top-level heading. It's used to indicate the most important heading on the page. There are six levels of headings (`<h1>` to `<h6>`), with `<h1>` being the highest priority.
7. **`<p>`**: Represents a paragraph of text. This tag automatically adds space before and after the paragraph.
8. **`<a href="https://www.example.com">`**: Defines a hyperlink. The `href` attribute specifies the URL that the link points to. Clicking the link will take the user to "https://www.example.com".

This basic structure is the foundation of most web pages. By learning HTML, you can begin to create your own web content and structure it in a way that browsers can understand and display.

Basic HTML Tags

Headings: Used to define headings in a document. There are six levels of headings, from `<h1>` to `<h6>`, with `<h1>` being the most important and `<h6>` the least.

```
<h1>Main Heading</h1>
<h2>Subheading</h2>
<h3>Sub-subheading</h3>
```

Paragraphs: The `<p>` tag defines a paragraph of text.

<p>This is a paragraph of text. It is used to structure content into readable sections.</p>

Links: The `<a>` tag defines a hyperlink. The `href` attribute specifies the URL of the page the link goes to.

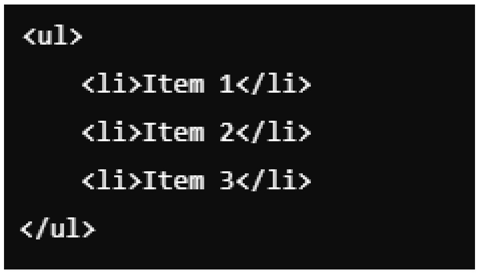
```
<a href="https://www.example.com">Visit Example.com</a>
```

Images: The `<img>` tag embeds an image. The `src` attribute specifies the path to the image file, and the `alt` attribute provides alternative text if the image cannot be displayed.

```
<img src="image.jpg" alt="Description of the image">
```

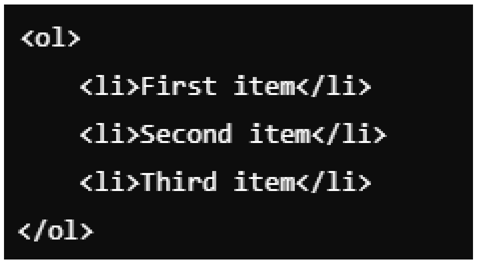
Lists: Lists can be either ordered (`<ol>`) or unordered (`<ul>`). List items are defined using the `<li>` tag.

Unordered List



```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
```

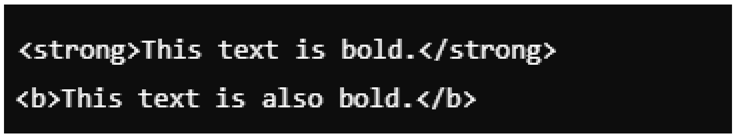
Ordered List



```
<ol>
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
</ol>
```

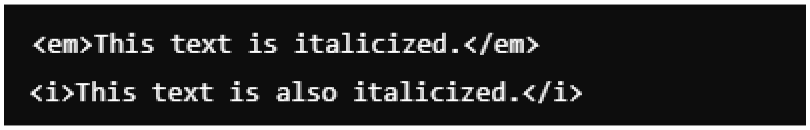
Bold and Italics: Used to emphasize text.

Bold: `<strong>` or `<b>`



```
<strong>This text is bold.</strong>
<b>This text is also bold.</b>
```

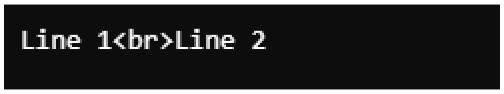
Italics: `<em>` or `<i>`



```
 <em>This text is italicized.</em>
 <i>This text is also italicized.</i>
```

Breaks and Horizontal Rules:

Line Break: `<br>` inserts a line break:



```
Line 1<br>Line 2
```

Horizontal Rule: `<hr>` creates a horizontal line.

```
<hr>
```

Basic HTML Attributes

Attributes provide additional information about HTML elements. They are always specified in the opening tag and are written as name-value pairs. Here are some common attributes:

1. **`href`**: Specifies the URL for a hyperlink.

```
<a href="https://www.example.com">Visit Example.com</a>
```

2.**`src`**: Defines the source path of an image.

```
<img src="image.jpg" alt="Description of the image">
```

3.**`alt`**: Provides alternative text for images.

```
<img src="image.jpg" alt="A beautiful landscape">
```

4. **`title`**: Adds additional information that usually appears as a tooltip when the user hovers over the element.

```
<a href="https://www.example.com" title="Go to Example.com">Visit Example.com</a>
```

5. **`id`**: Assigns a unique identifier to an element, which can be used for styling or scripting.

```
<p id="uniqueParagraph">This is a paragraph with a unique ID.</p>
```

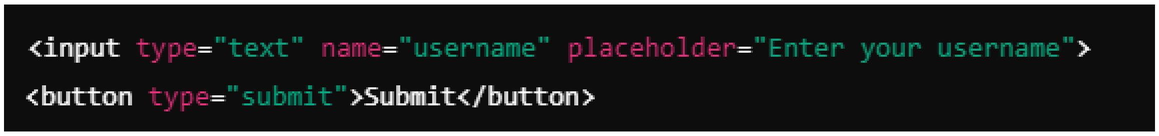
6. **`class`**: Assigns one or more class names to an element for styling purposes.

```
<p class="text-muted">This is a paragraph with a class.</p>
```

7. **`style`**: Applies inline CSS styles directly to an element.

```
<p style="color: red; font-size: 18px;">This text is red and 18px in size.</p>
```

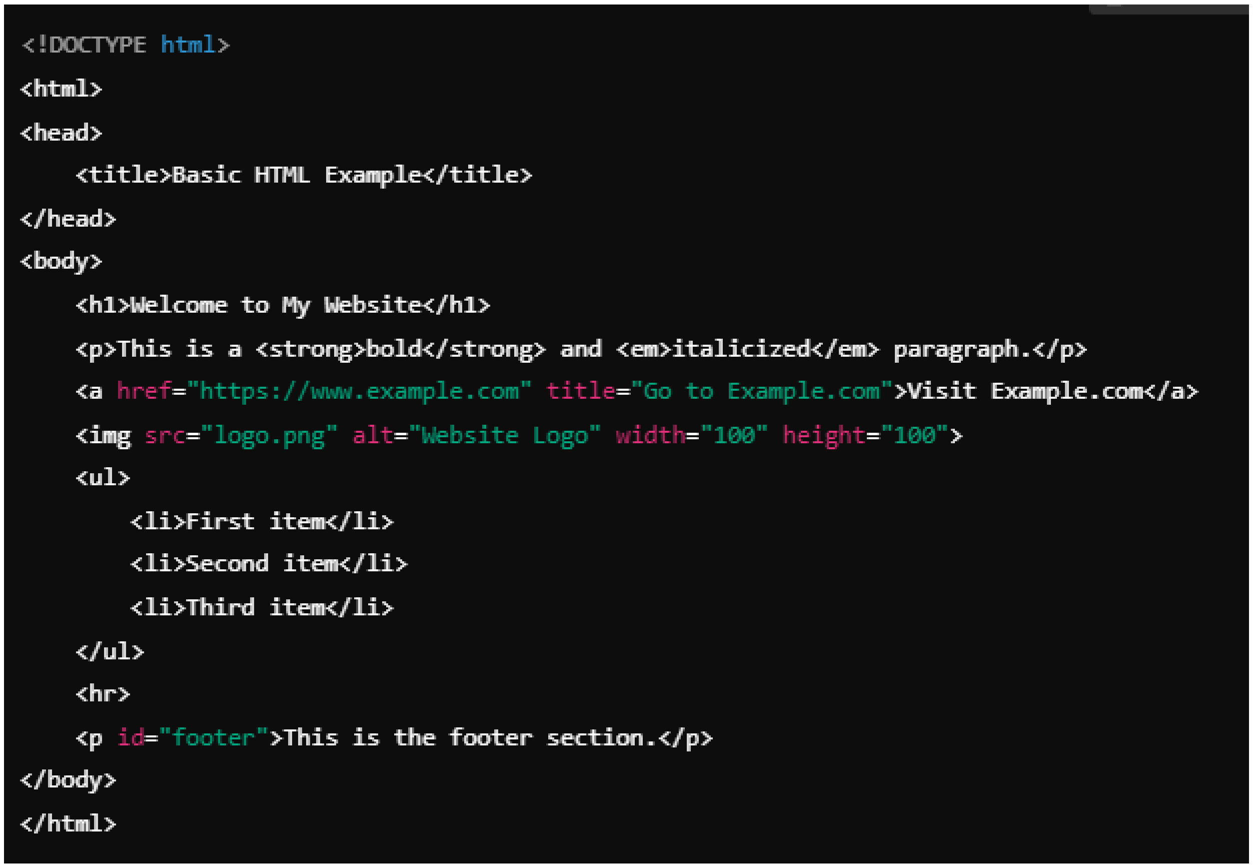
8. **`type`**: Specifies the type of input or button, among other things. Often used in forms.



```html
<input type="text" name="username" placeholder="Enter your username">
<button type="submit">Submit</button>
```

Example Combining Tags and Attributes

Here's a simple HTML snippet that combines various tags and attributes:



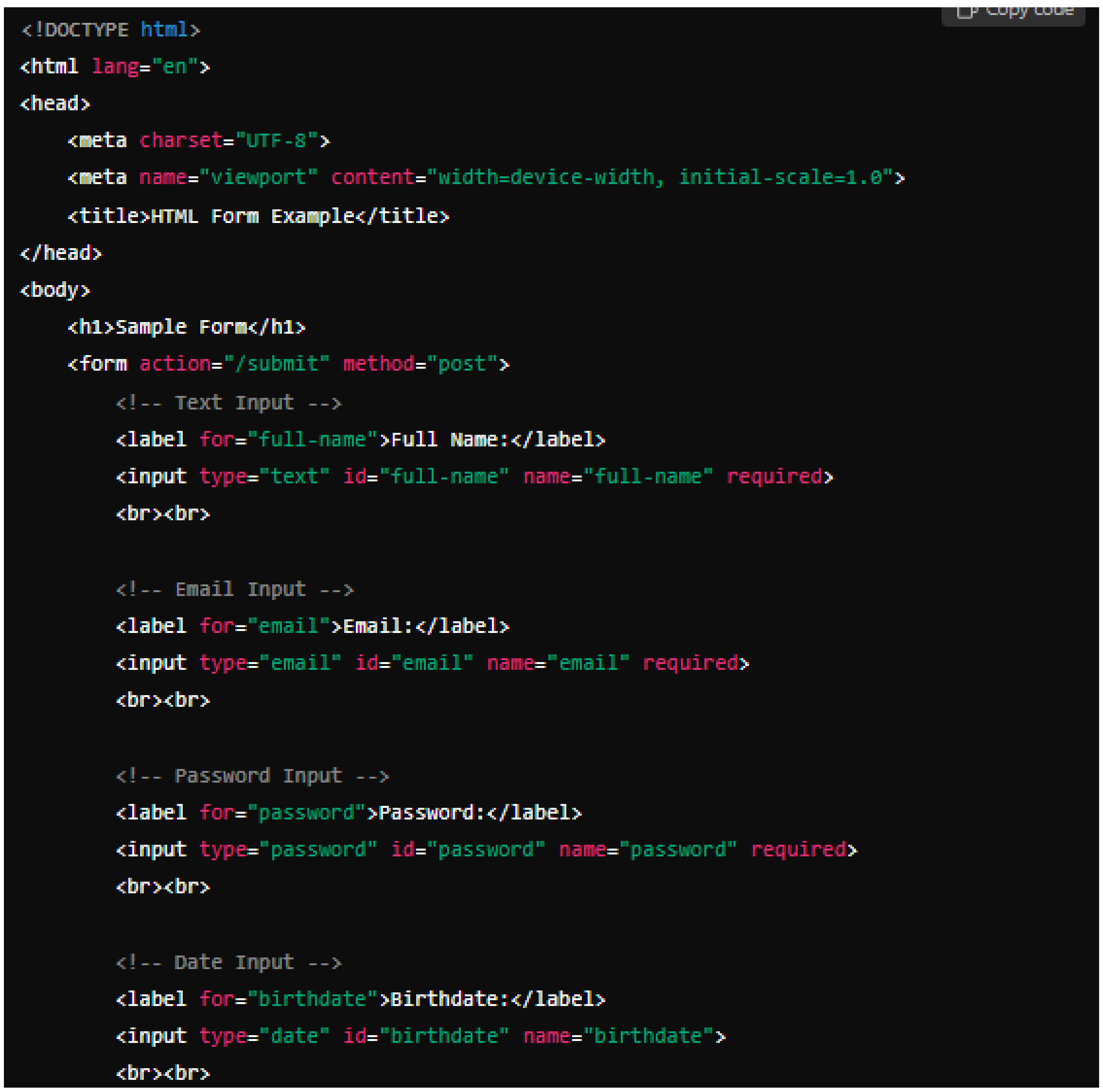
```html
<!DOCTYPE html>
<html>
<head>
    <title>Basic HTML Example</title>
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>This is a <strong>bold</strong> and <em>italicized</em> paragraph.</p>
    <a href="https://www.example.com" title="Go to Example.com">Visit Example.com</a>
    <img src="logo.png" alt="Website Logo" width="100" height="100">
    <ul>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
    </ul>
    <hr>
    <p id="footer">This is the footer section.</p>
</body>
</html>
```

In this example, you can see the use of different tags like `<h1>`, `<p>`, `<a>`, `<img>`, and `<ul>`, along with various attributes such as `href`, `src`, `alt`, `title`, and `id`. This demonstrates how HTML tags and attributes work together to create and format web content.

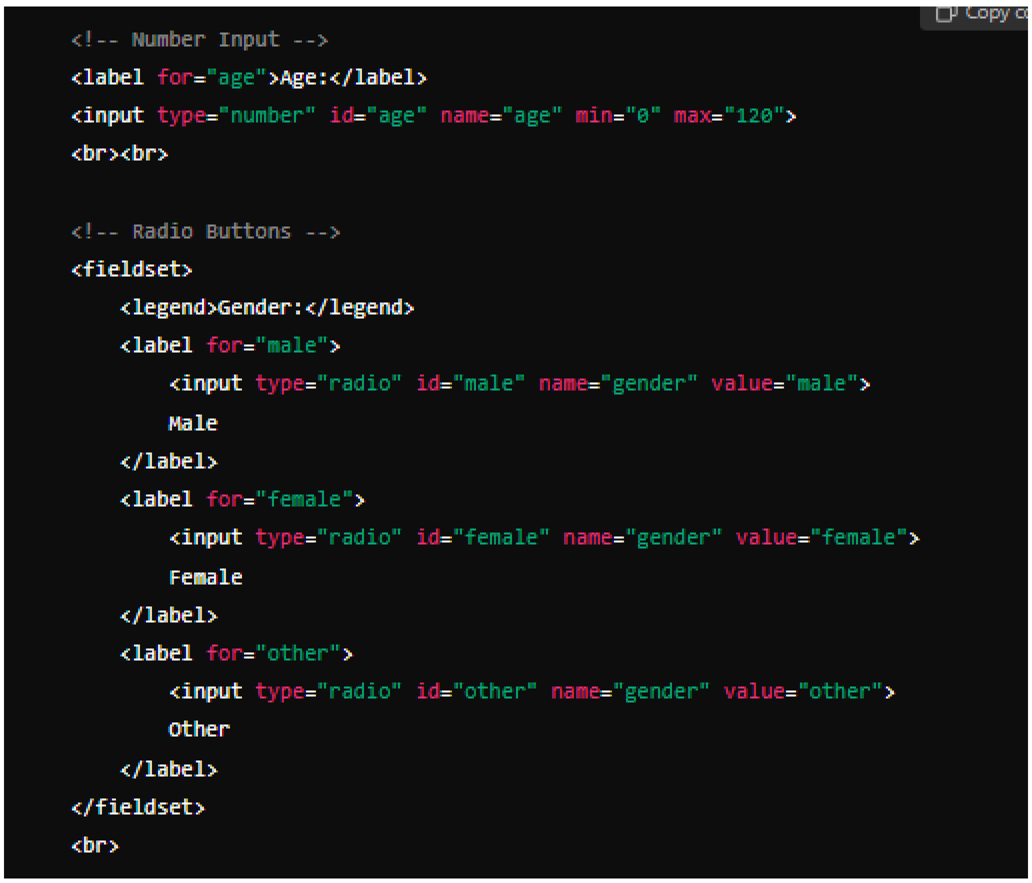
Class 2:HTML Forms

Creating a form in HTML involves using various `<input>` elements to collect user data. Below is an example of an HTML form that includes a range of input types. Each input type has its own use case, such as text fields for names, email fields for emails, and more specialized inputs for dates and numbers.

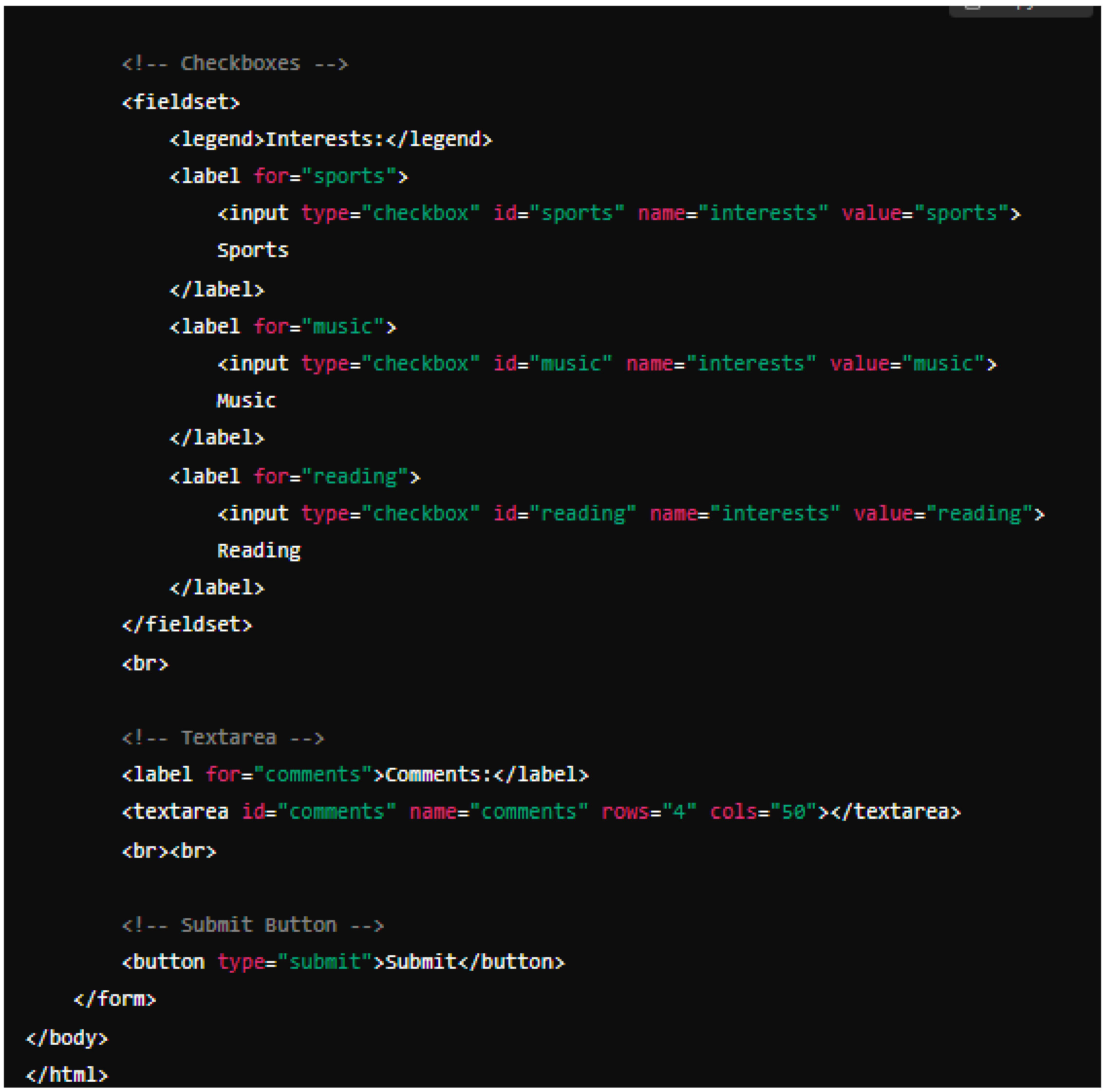
Here's a basic HTML form example covering a variety of input types:



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML Form Example</title>
</head>
<body>
    <h1>Sample Form</h1>
    <form action="/submit" method="post">
        <!-- Text Input -->
        <label for="full-name">Full Name:</label>
        <input type="text" id="full-name" name="full-name" required>
        <br><br>

        <!-- Email Input -->
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <br><br>

        <!-- Password Input -->
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <br><br>

        <!-- Date Input -->
        <label for="birthdate">Birthdate:</label>
        <input type="date" id="birthdate" name="birthdate">
        <br><br>
```



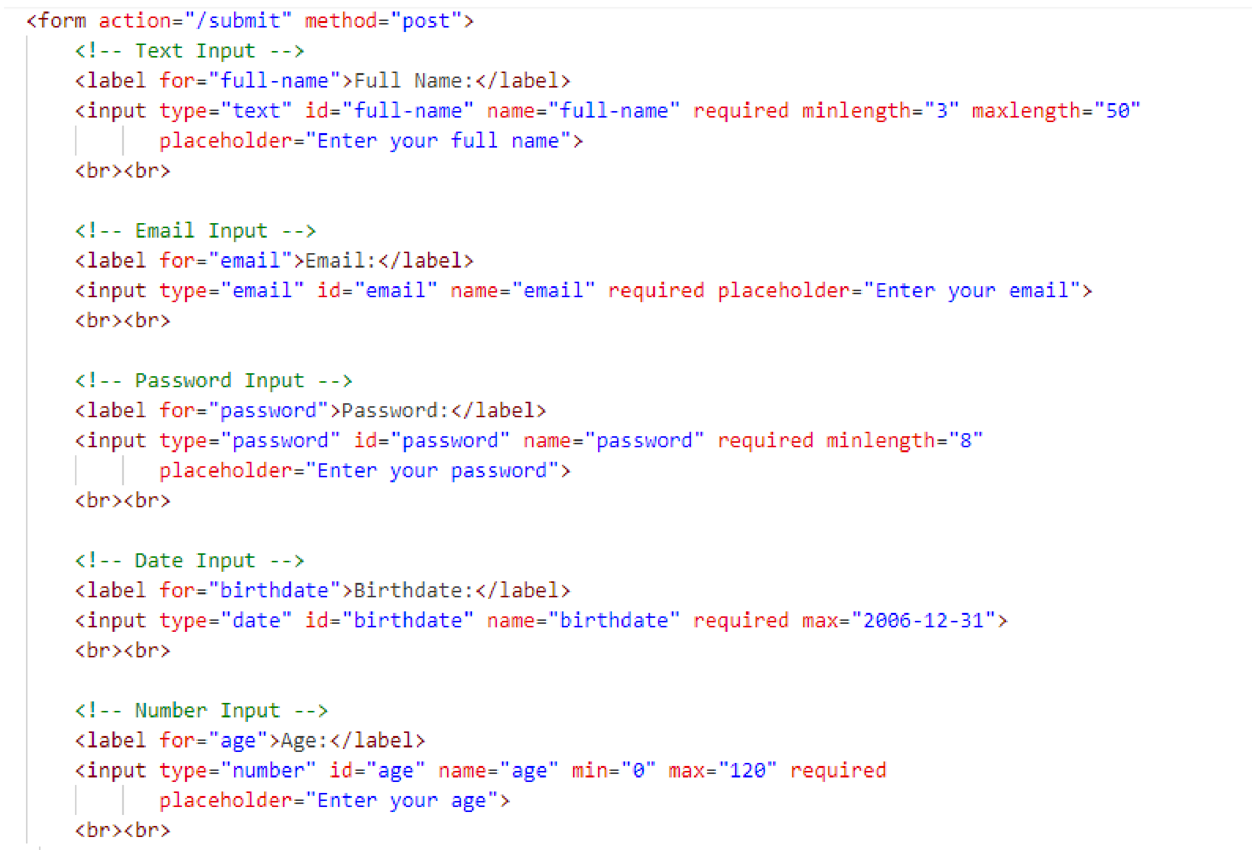
```
<!-- Number Input -->
<label for="age">Age:</label>
<input type="number" id="age" name="age" min="0" max="120">
<br><br>

<!-- Radio Buttons -->
<fieldset>
    <legend>Gender:</legend>
    <label for="male">
        <input type="radio" id="male" name="gender" value="male">
        Male
    </label>
    <label for="female">
        <input type="radio" id="female" name="gender" value="female">
        Female
    </label>
    <label for="other">
        <input type="radio" id="other" name="gender" value="other">
        Other
    </label>
</fieldset>
<br>
```



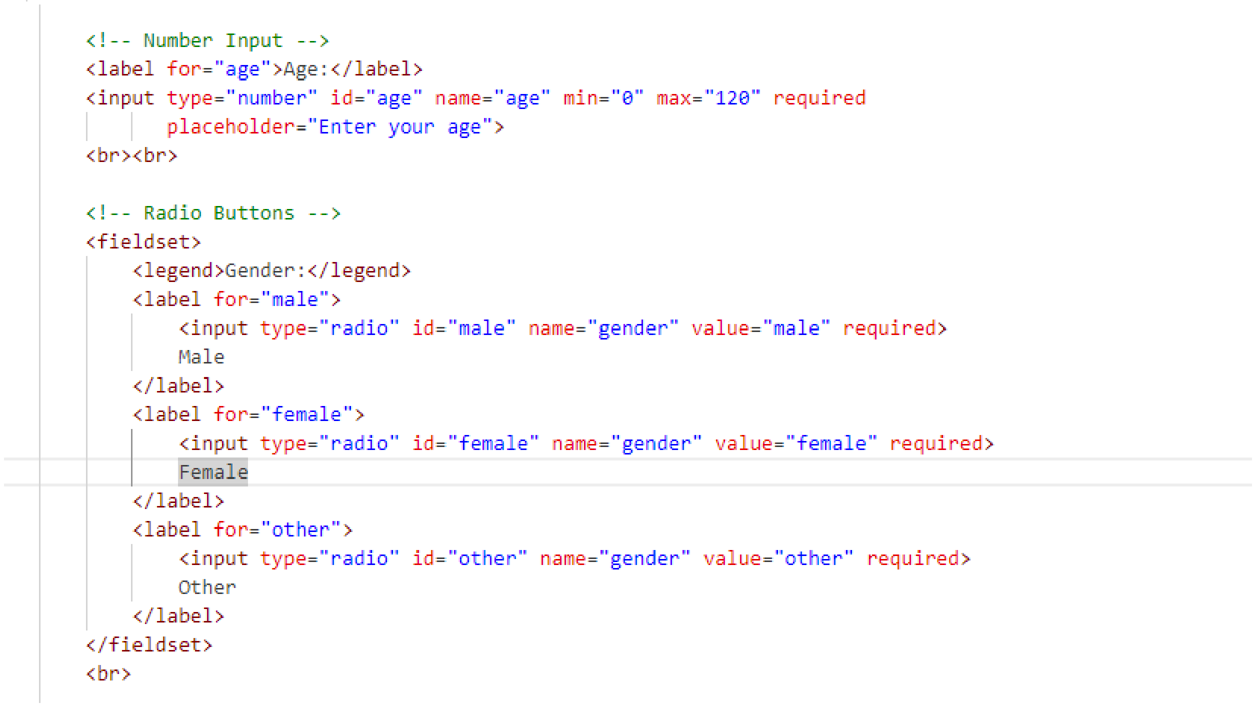
```html
        <!-- Checkboxes -->
        <fieldset>
            <legend>Interests:</legend>
            <label for="sports">
                <input type="checkbox" id="sports" name="interests" value="sports">
                Sports
            </label>
            <label for="music">
                <input type="checkbox" id="music" name="interests" value="music">
                Music
            </label>
            <label for="reading">
                <input type="checkbox" id="reading" name="interests" value="reading">
                Reading
            </label>
        </fieldset>
        <br>

        <!-- Textarea -->
        <label for="comments">Comments:</label>
        <textarea id="comments" name="comments" rows="4" cols="50"></textarea>
        <br><br>

        <!-- Submit Button -->
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

Explanation:

1. **Text Input**: Collects single-line text. Example: Full Name.
2. **Email Input**: Ensures the entered value is in email format. Example: Email.
3. **Password Input**: Hides the text for secure entry. Example: Password.
4. **Date Input**: Provides a date picker. Example: Birthdate.
5. **Number Input**: Restricts input to numbers, with optional min/max values. Example: Age.
6. **Radio Buttons**: Allows selection of a single option from a set. Example: Gender.
7. **Checkboxes**: Allows selection of multiple options. Example: Interests.
8. **Textarea**: Provides a multi-line text input. Example: Comments.
9. **Submit Button**: Submits the form data to the server.



```html
<form action="/submit" method="post">
    <!-- Text Input -->
    <label for="full-name">Full Name:</label>
    <input type="text" id="full-name" name="full-name" required minlength="3" maxlength="50"
           placeholder="Enter your full name">
    <br><br>

    <!-- Email Input -->
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required placeholder="Enter your email">
    <br><br>

    <!-- Password Input -->
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required minlength="8"
           placeholder="Enter your password">
    <br><br>

    <!-- Date Input -->
    <label for="birthdate">Birthdate:</label>
    <input type="date" id="birthdate" name="birthdate" required max="2006-12-31">
    <br><br>

    <!-- Number Input -->
    <label for="age">Age:</label>
    <input type="number" id="age" name="age" min="0" max="120" required
           placeholder="Enter your age">
    <br><br>
```



```html
    <!-- Number Input -->
    <label for="age">Age:</label>
    <input type="number" id="age" name="age" min="0" max="120" required
           placeholder="Enter your age">
    <br><br>

    <!-- Radio Buttons -->
    <fieldset>
        <legend>Gender:</legend>
        <label for="male">
            <input type="radio" id="male" name="gender" value="male" required>
            Male
        </label>
        <label for="female">
            <input type="radio" id="female" name="gender" value="female" required>
            Female
        </label>
        <label for="other">
            <input type="radio" id="other" name="gender" value="other" required>
            Other
        </label>
    </fieldset>
    <br>
```

```html
        <!-- Checkboxes -->
        <fieldset>
            <legend>Interests:</legend>
            <label for="sports">
                <input type="checkbox" id="sports" name="interests" value="sports">
                Sports
            </label>
            <label for="music">
                <input type="checkbox" id="music" name="interests" value="music">
                Music
            </label>
            <label for="reading">
                <input type="checkbox" id="reading" name="interests" value="reading">
                Reading
            </label>
        </fieldset>
        <br>

        <!-- Textarea -->
        <label for="comments">Comments:</label>
        <textarea id="comments" name="comments" rows="4" cols="50" maxlength="500"
                  placeholder="Enter your comments"></textarea>
        <br><br>

        <!-- Submit Button -->
        <button type="submit">Submit</button>
    </form>
```

Explanation of Validation Attributes Used

1. `required`: Makes the field mandatory. The form will not submit unless the field is filled out.

```html
<input type="text" id="full-name" name="full-name" required>
```

2. `minlength` and `maxlength`: Specify the minimum and maximum number of characters allowed.

```html
<input type="text" id="full-name" name="full-name" required minlength="3" maxlength="50"
```

3. `pattern`: Defines a regular expression for custom validation. (Not used in this example, but useful for more complex validations.)

4. `type="email"`: Validates that the input conforms to the format of an email address.

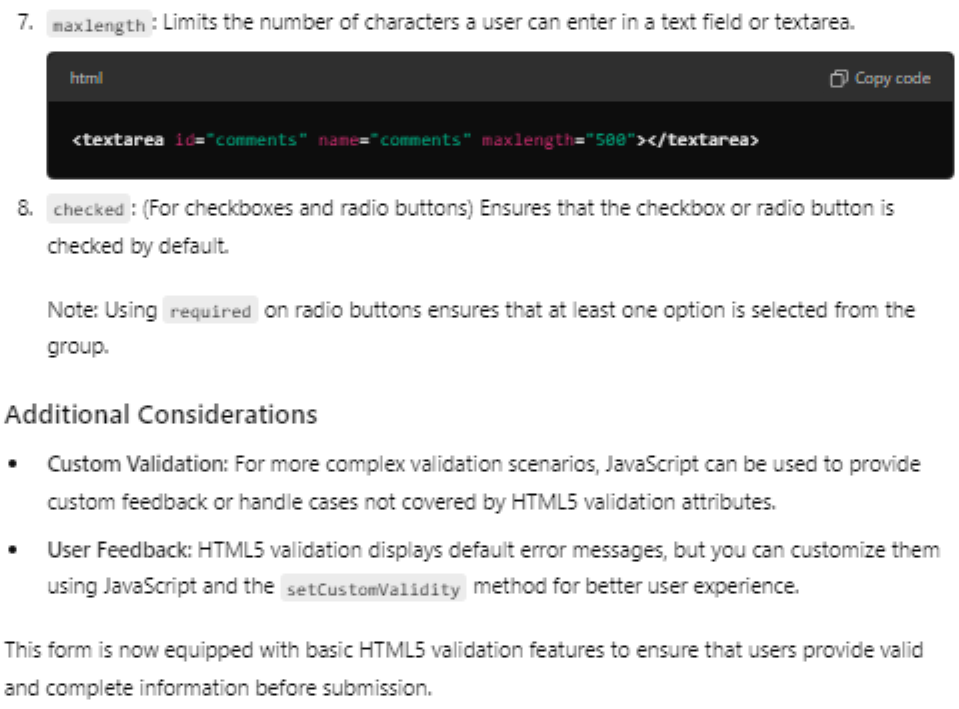
```html
<input type="email" id="email" name="email" required>
```

5. `min` and `max`: Define minimum and maximum values for numeric and date inputs.

```html
<input type="number" id="age" name="age" min="0" max="120" required>
<input type="date" id="birthdate" name="birthdate" required max="2006-12-31">
```

6. `placeholder`: Provides a hint about what the field is for, though it's not a validation attribute.

```html
<input type="text" id="full-name" name="full-name" placeholder="Enter your full name">
```



7. `maxlength`: Limits the number of characters a user can enter in a text field or textarea.

```html
<textarea id="comments" name="comments" maxlength="500"></textarea>
```

8. `checked`: (For checkboxes and radio buttons) Ensures that the checkbox or radio button is checked by default.

   Note: Using `required` on radio buttons ensures that at least one option is selected from the group.

Additional Considerations

- Custom Validation: For more complex validation scenarios, JavaScript can be used to provide custom feedback or handle cases not covered by HTML5 validation attributes.
- User Feedback: HTML5 validation displays default error messages, but you can customize them using JavaScript and the `setCustomValidity` method for better user experience.

This form is now equipped with basic HTML5 validation features to ensure that users provide valid and complete information before submission.

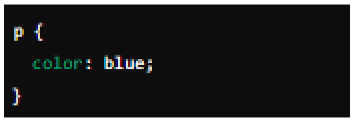
<u>**Class 3: CSS Basics**</u>

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a document written in HTML or XML. It defines how HTML elements should be displayed on screen, paper, or in other media. CSS is used to control the layout of multiple web pages all at once, allowing for separation of content (HTML) from design (CSS).

Key Concepts in CSS:

1. **Selectors**: Define which HTML elements the CSS rules apply to.

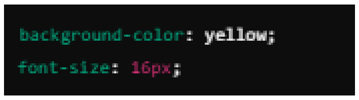
   Example:



```css
p {
  color: blue;
}
```

This targets all `<p>` elements and makes their text color blue.

2. **Properties and Values**: CSS applies styles through properties, and each property has a set of values.
   Example:



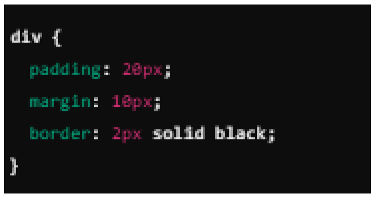
```css
background-color: yellow;
font-size: 16px;
```

3. **Cascading and Specificity**:

- CSS allows multiple rules to apply to the same element. The term "cascading" means that the order of rules can affect how they apply. More specific selectors (like IDs) take precedence over less specific ones (like classes or elements).

4. **Box Model**: This model refers to the space an element takes up on a page, which includes:
   - Content
   - Padding (space between content and border)
   - Border (around padding)
   - Margin (space outside the border)

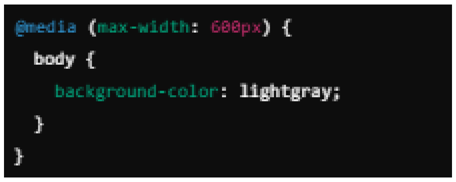
   Example:



```
div {
  padding: 20px;
  margin: 10px;
  border: 2px solid black;
}
```

5. **Layout Techniques**:
   - Flexbox: A layout model for designing responsive layouts.
   - Grid: A two-dimensional layout system for creating complex web designs.
6. **Responsive Design**:

   CSS can make web pages adapt to different screen sizes using media queries.



```
@media (max-width: 600px) {
  body {
    background-color: lightgray;
  }
}
```

External, Internal, and Inline CSS:

7. External: CSS is written in a separate file and linked to HTML.
   - Internal: CSS is placed within a <style> tag inside the HTML <head>.
   - Inline: CSS is applied directly to an HTML element using the style attribute.
8. CSS Frameworks:

   - Popular CSS frameworks, such as Bootstrap or Tailwind CSS, provide pre-designed components and layout options for faster development.

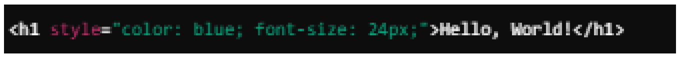
CSS is a powerful tool that enhances the visual presentation and user experience of a web page, enabling developers to create visually appealing and responsive designs.

its own use case depending on the scope and complexity of the styles you're working with.

1. Inline CSS

- **Definition**: Inline CSS is applied directly within the HTML element using the `style` attribute. This method only affects the specific element it's applied to.
- **Use Case**: Best for applying quick, one-off styles to a specific element.

  Example:

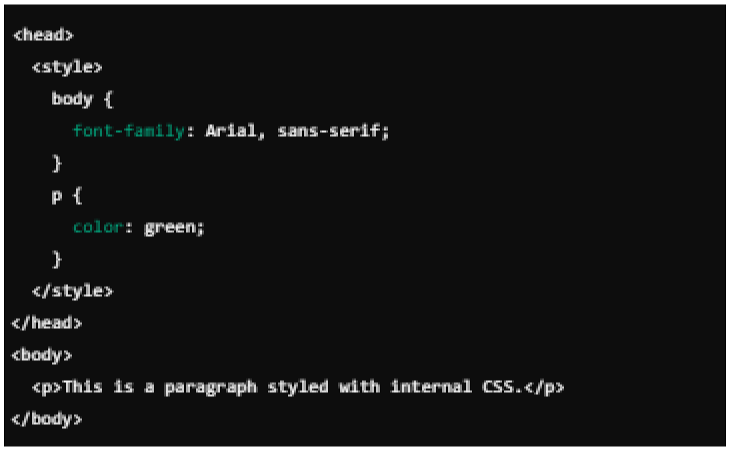
```
<h1 style="color: blue; font-size: 24px;">Hello, World!</h1>
```

  - **Pros**:
    - Quick and easy for small styling tasks.
    - Overrides both internal and external styles due to its higher specificity.
  - **Cons**:
    - Makes HTML harder to read and maintain.
    - Repetitive if multiple elements require the same style.

2. Internal CSS

- **Definition**: Internal CSS is placed inside a `<style>` tag in the `<head>` section of the HTML document. It applies styles to the entire document, but only that specific HTML file.
- **Use Case**: Useful when you want to apply styles to a single page without affecting other pages.

  Example:

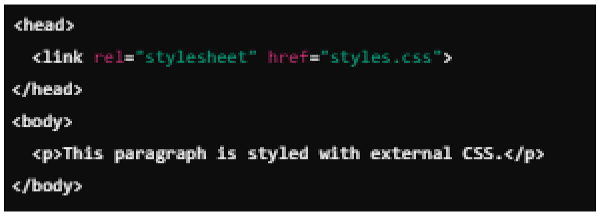
```
<head>
  <style>
    body {
      font-family: Arial, sans-serif;
    }
    p {
      color: green;
    }
  </style>
</head>
<body>
  <p>This is a paragraph styled with internal CSS.</p>
</body>
```

  - **Pros**:
    - Keeps style centralized for the page, improving maintainability compared to inline CSS.
    - Easy to manage for single-page designs.

- **Cons**:
  - Cannot be reused across multiple pages.
  - Increases page size since the styles are embedded within the HTML file.

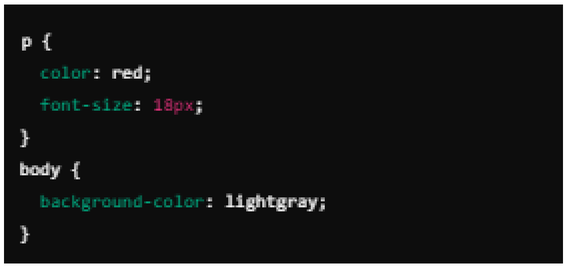
3. External CSS

- **Definition**: External CSS is written in a separate `.css` file and linked to the HTML document using the `<link>` tag in the `<head>` section. The styles can be applied to multiple HTML pages, promoting reusability.
- **Use Case**: Best for larger websites where you need consistent styling across multiple pages.
- **Example**:
  - In your HTML file:



```html
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <p>This paragraph is styled with external CSS.</p>
</body>
```

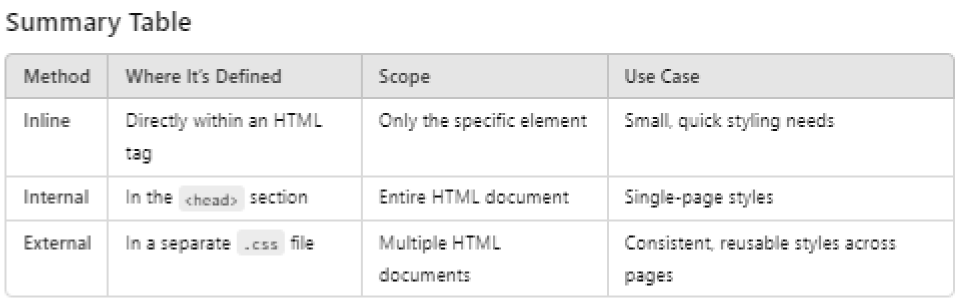
In the `styles.css` file:



```css
p {
  color: red;
  font-size: 18px;
}
body {
  background-color: lightgray;
}
```

- **Pros**:
  - Styles can be reused across multiple pages.
  - Keeps the HTML document clean and more manageable.
  - Enhances performance, as browsers can cache external CSS files.

- **Cons**:
  - Requires an extra HTTP request to load the external CSS file (though this can be minimized with caching).
  - Changes require updates to the external file, so debugging may sometimes be more complex.



Summary Table

Method	Where It's Defined	Scope	Use Case
Inline	Directly within an HTML tag	Only the specific element	Small, quick styling needs
Internal	In the `<head>` section	Entire HTML document	Single-page styles
External	In a separate `.css` file	Multiple HTML documents	Consistent, reusable styles across pages

Each method of applying CSS has its specific strengths, and often you'll use a combination of these depending on your project needs. For large-scale projects, **external CSS** is usually preferred for better maintainability and reusability.

Class 4: Bootstrap

Bootstrap is a popular open-source front-end framework for building responsive, mobile-first websites. It includes HTML, CSS, and JavaScript components, which make it easier to design web pages without writing a lot of custom code. It is used to create responsive and consistent designs across different devices like desktops, tablets, and phones.

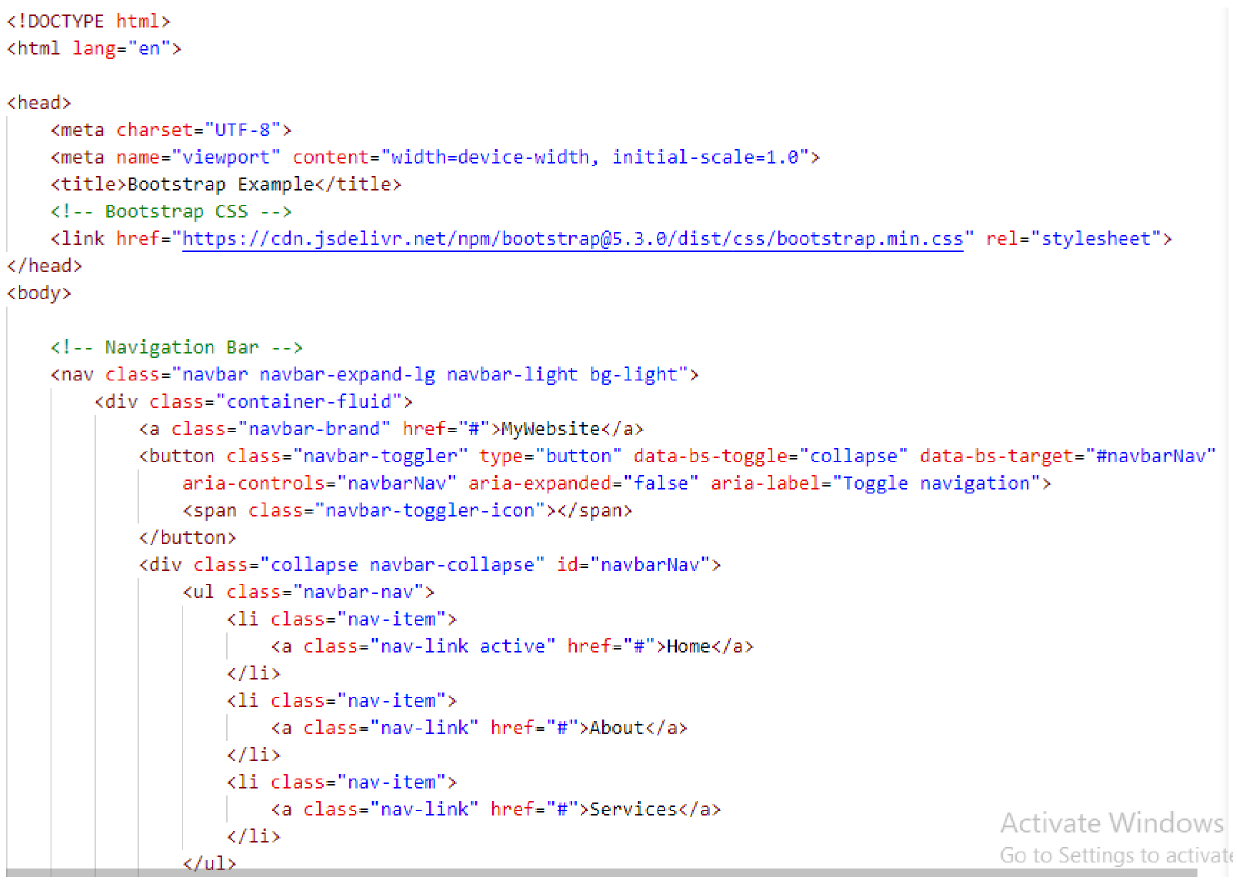
Key Features of Bootstrap:

1. **Responsive Grid System**: A 12-column grid system to easily arrange your content.
2. **Components**: Pre-designed components like navigation bars, modals, buttons, alerts, etc.
3. **Utilities**: Helper classes for margin, padding, text alignment, etc.
4. **JavaScript Plugins**: JavaScript components for modals, tooltips, carousels, etc.
5. **Customizable**: You can customize Bootstrap by overriding its default styles.

Example of a Simple Bootstrap Webpage

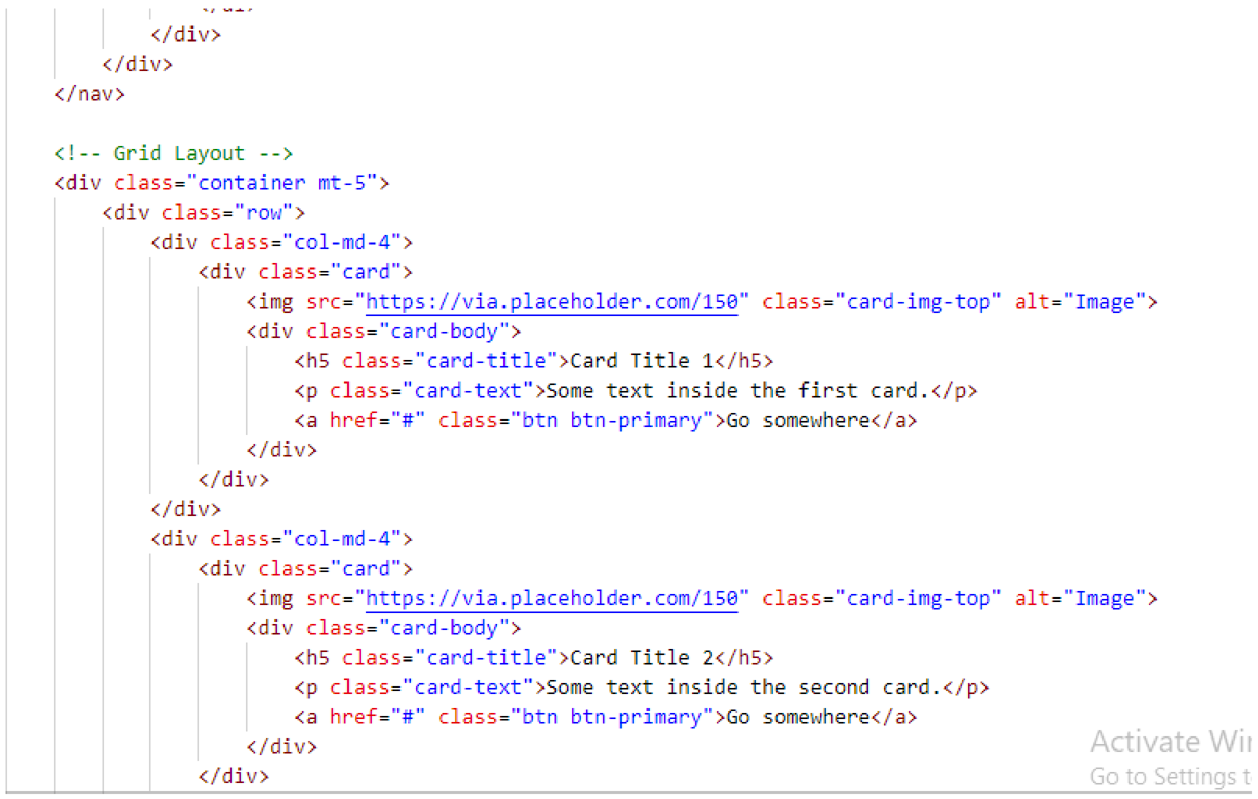
Explanation:

1. **Navbar**: The `<nav>` tag creates a responsive navigation bar.
2. **Grid Layout**: The `.container`, `.row`, and `.col-md-4` classes implement a 3-column layout for medium devices and above.
3. **Cards**: The `.card` class is used to display content (image, title, and text) in a card format.



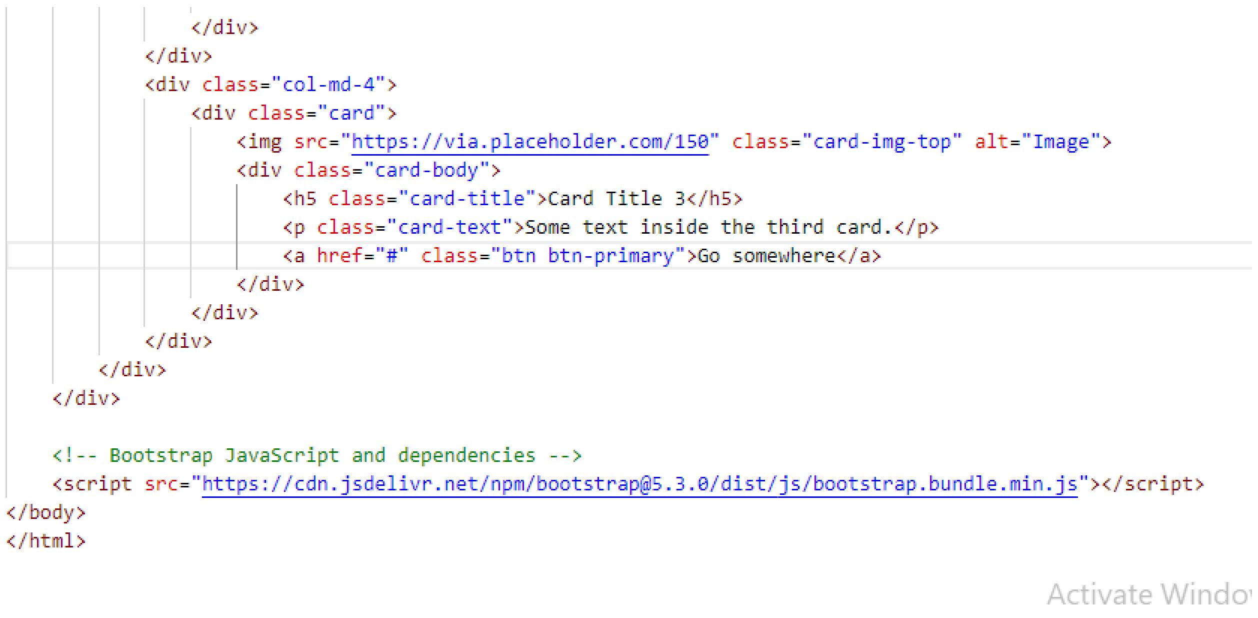
```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bootstrap Example</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>

    <!-- Navigation Bar -->
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">MyWebsite</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
                aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                    <li class="nav-item">
                        <a class="nav-link active" href="#">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#">About</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#">Services</a>
                    </li>
                </ul>
```

Activate Windows
Go to Settings to activate



```html
            </div>
        </div>
    </nav>

    <!-- Grid Layout -->
    <div class="container mt-5">
        <div class="row">
            <div class="col-md-4">
                <div class="card">
                    <img src="https://via.placeholder.com/150" class="card-img-top" alt="Image">
                    <div class="card-body">
                        <h5 class="card-title">Card Title 1</h5>
                        <p class="card-text">Some text inside the first card.</p>
                        <a href="#" class="btn btn-primary">Go somewhere</a>
                    </div>
                </div>
            </div>
            <div class="col-md-4">
                <div class="card">
                    <img src="https://via.placeholder.com/150" class="card-img-top" alt="Image">
                    <div class="card-body">
                        <h5 class="card-title">Card Title 2</h5>
                        <p class="card-text">Some text inside the second card.</p>
                        <a href="#" class="btn btn-primary">Go somewhere</a>
                    </div>
                </div>
```

Activate Wi
Go to Settings t



```
                </div>
            </div>
            <div class="col-md-4">
                <div class="card">
                    <img src="https://via.placeholder.com/150" class="card-img-top" alt="Image">
                    <div class="card-body">
                        <h5 class="card-title">Card Title 3</h5>
                        <p class="card-text">Some text inside the third card.</p>
                        <a href="#" class="btn btn-primary">Go somewhere</a>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Bootstrap JavaScript and dependencies -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Activate Windo

Class 5: Javascript

JavaScript Syntax

JavaScript syntax is the set of rules that define a correctly structured JavaScript program. Here are some key components:

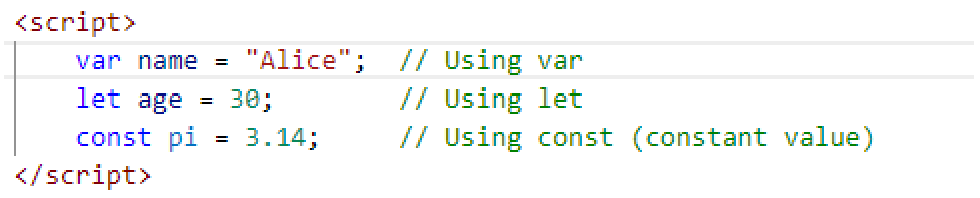
1. **Comments:**
   - Single-line comments use `//`.
   - Multi-line comments use `/* ... */`
2. **Statements:**

- A statement is a piece of code that performs an action. Most statements end with a semicolon (`;`), though it's often optional.

  let x = 5; // This is a statement

3. **Variables:**

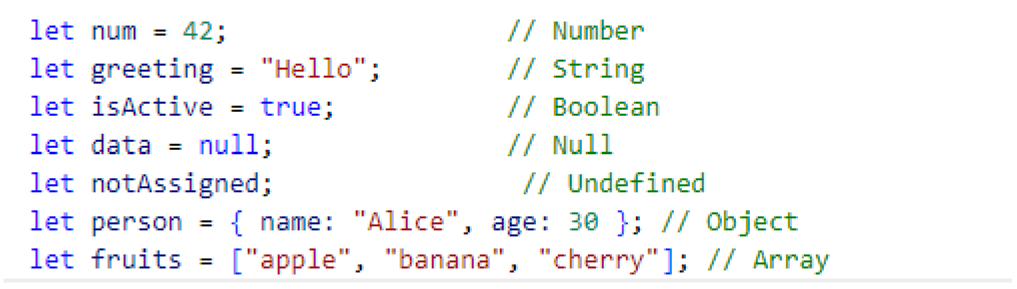
- Variables can be declared using `var`, `let`, or `const`.



```
<script>
    var name = "Alice";  // Using var
    let age = 30;        // Using let
    const pi = 3.14;     // Using const (constant value)
</script>
```

4. **Data Types:**

- JavaScript has several data types, including:
  - **Number:** Numeric values.
  - **String:** Text values enclosed in quotes.
  - **Boolean:** `true` or `false`.
  - **Undefined:** A variable that has been declared but not assigned a value.
  - **Null:** A variable that has no value.
  - **Object:** A collection of key-value pairs.
  - **Array:** A special type of object used for storing ordered collections.



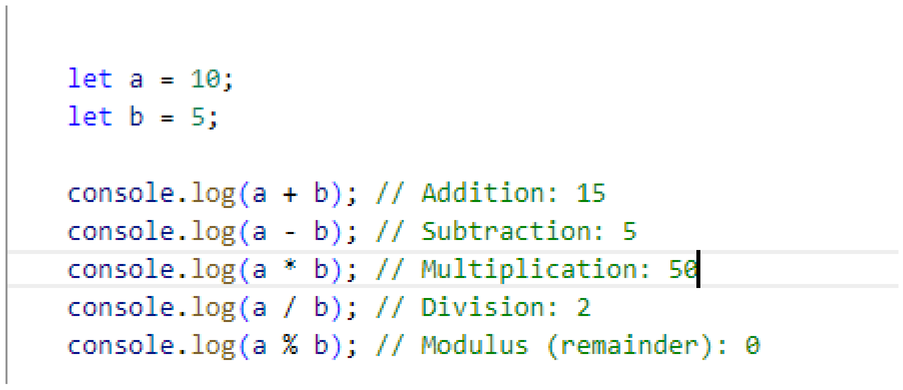
```
let num = 42;                    // Number
let greeting = "Hello";          // String
let isActive = true;             // Boolean
let data = null;                 // Null
let notAssigned;                  // Undefined
let person = { name: "Alice", age: 30 }; // Object
let fruits = ["apple", "banana", "cherry"]; // Array
```

Basic Operations

JavaScript supports several types of operations, including arithmetic, assignment, comparison, and logical operations.

1. Arithmetic Operations

JavaScript provides basic arithmetic operations like addition, subtraction, multiplication, and division.



```
let a = 10;
let b = 5;

console.log(a + b); // Addition: 15
console.log(a - b); // Subtraction: 5
console.log(a * b); // Multiplication: 50
console.log(a / b); // Division: 2
console.log(a % b); // Modulus (remainder): 0
```

2. Assignment Operators

Assignment operators assign values to variables.

```
let c = 10;
c += 5; // c = c + 5, c is now 15
c -= 3; // c = c - 3, c is now 12
c *= 2; // c = c * 2, c is now 24
c /= 4; // c = c / 4, c is now 6
```

3. Comparison Operators

Comparison operators compare two values and return a Boolean (`true` or `false`).

```
console.log(5 == "5");  // Loose equality: true
console.log(5 === "5"); // Strict equality: false
console.log(5 != "5");  // Loose inequality: false
console.log(5 !== "5"); // Strict inequality: true
console.log(5 > 3);     // Greater than: true
console.log(5 < 10);    // Less than: true
console.log(5 >= 5);    // Greater than or equal: true
console.log(5 <= 4);    // Less than or equal: false
```

4. Logical Operators

Logical operators combine Boolean values.

```
let x = true;
let y = false;

console.log(x && y); // AND: false
console.log(x || y); // OR: true
console.log(!x);     // NOT: false
```

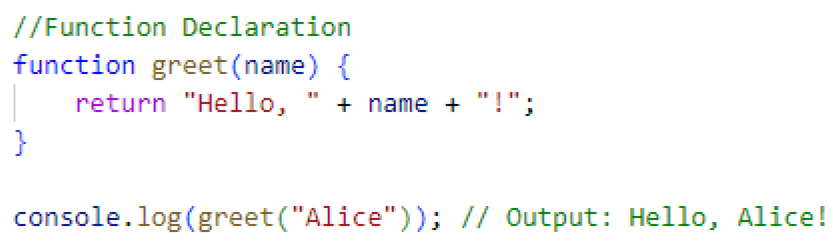
Functions in JavaScript

A **function** is a block of code designed to perform a specific task. Functions can take inputs (parameters) and can return an output.

Defining Functions

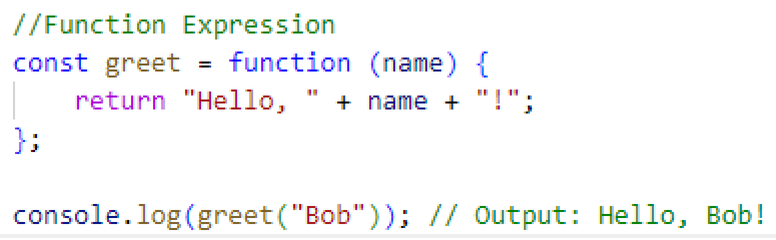
There are several ways to define a function in JavaScript:

1. **Function Declaration**



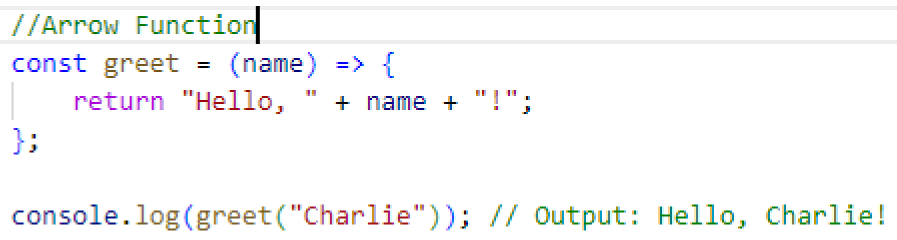
```javascript
//Function Declaration
function greet(name) {
    return "Hello, " + name + "!";
}

console.log(greet("Alice")); // Output: Hello, Alice!
```

2. Function Expression



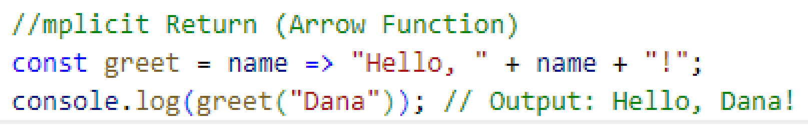
```javascript
//Function Expression
const greet = function (name) {
    return "Hello, " + name + "!";
};

console.log(greet("Bob")); // Output: Hello, Bob!
```

3. Arrow Function



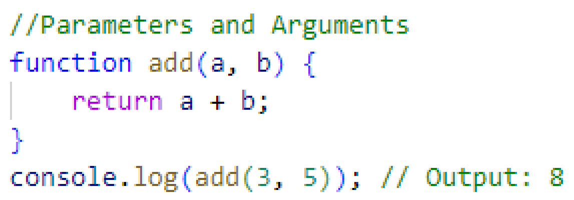
```javascript
//Arrow Function
const greet = (name) => {
    return "Hello, " + name + "!";
};

console.log(greet("Charlie")); // Output: Hello, Charlie!
```

4. **Implicit Return (Arrow Function)** If the function body is a single expression, you can omit the braces and the `return` statement:



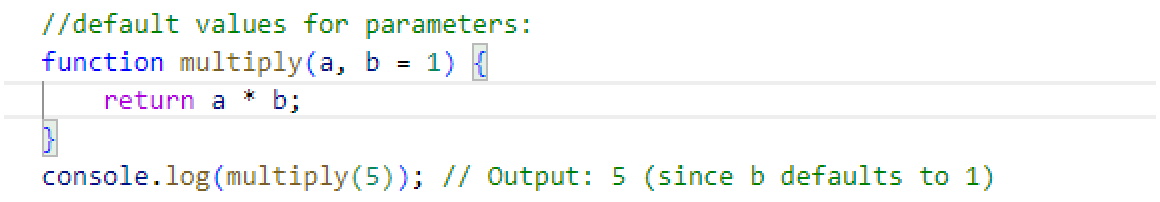
```javascript
//mplicit Return (Arrow Function)
const greet = name => "Hello, " + name + "!";
console.log(greet("Dana")); // Output: Hello, Dana!
```

Parameters and Arguments
Functions can have parameters, which are placeholders for the values you pass when you call the function.



```
//Parameters and Arguments
function add(a, b) {
    return a + b;
}
console.log(add(3, 5)); // Output: 8
```

You can also set default values for parameters:

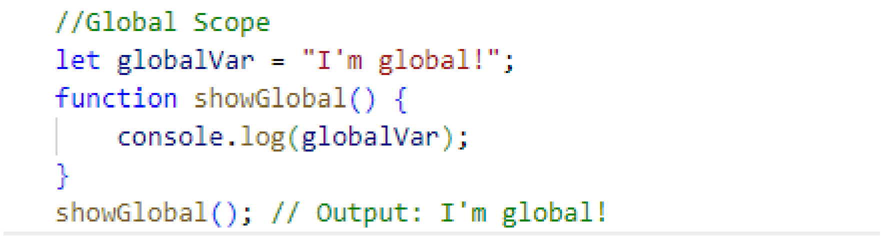


```
//default values for parameters:
function multiply(a, b = 1) {
    return a * b;
}
console.log(multiply(5)); // Output: 5 (since b defaults to 1)
```

Scope in JavaScript

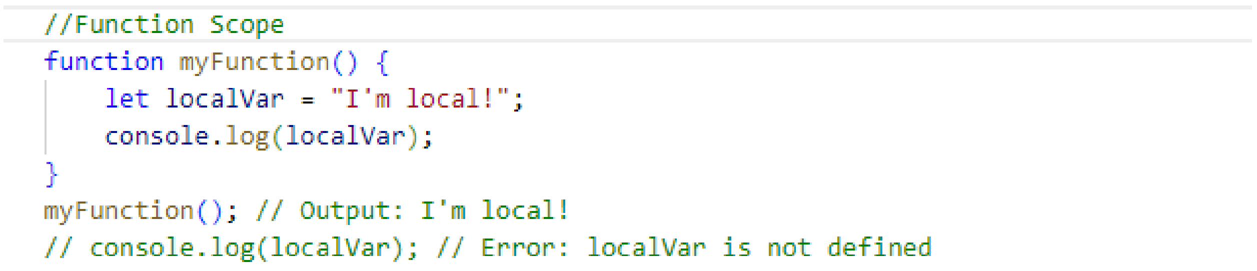
Scope refers to the visibility or accessibility of variables in your code. JavaScript has several types of scope:

1. **Global Scope** Variables declared outside any function are in the global scope. They can be accessed from anywhere in the code.



```
//Global Scope
let globalVar = "I'm global!";
function showGlobal() {
    console.log(globalVar);
}
showGlobal(); // Output: I'm global!
```

2. **Function Scope** Variables declared inside a function are in function scope and can only be accessed within that function.



```
//Function Scope
function myFunction() {
    let localVar = "I'm local!";
    console.log(localVar);
}
myFunction(); // Output: I'm local!
// console.log(localVar); // Error: localVar is not defined
```

3. **Block Scope** Variables declared with `let` or `const` inside a block (enclosed by `{}`) are in block scope and cannot be accessed outside that block.

```javascript
//Block Scope
if (true) {
    let blockVar = "I'm in a block!";
    console.log(blockVar); // Output: I'm in a block!
}
// console.log(blockVar); // Error: blockVar is not defined
```