HTML Complete Course

By:

PASS Education System

What is HTML?

History

HTML (Hypertext Markup Language) is the standard markup language for creating web pages and applications. It provides the structure and format for displaying content on the internet. The history of HTML dates back to the early days of the World Wide Web. Here's a brief overview of its evolution:

HTML 1.0: The first version of HTML was published by Tim Berners-Lee in 1991. It was a simple language with limited tags, primarily used to structure documents and create links between them.

HTML 2.0: This version, released in 1995, introduced new features like tables, image support, and form elements. It helped in advancing the web's capabilities and layout possibilities.

HTML 3.2: HTML 3.2, released in 1997, brought additional improvements, including support for style sheets, client-side image maps, and better table layouts.

HTML 4.01: This version, released in 1999, added more features and introduced a stricter standard. It included support for frames, scripting, and multimedia elements. HTML 4.01 had three variations: Strict, Transitional, and Frameset.

XHTML: XHTML (Extensible HTML) was introduced as an XML-based version of HTML. It followed stricter syntax rules and was designed to work with XML-based technologies. XHTML 1.0 was released in 2000, followed by XHTML 1.1 in 2001.

HTML5: HTML5 is a major revision of HTML that was first proposed in 2004 and became a W3C recommendation in 2014. It introduced many new features and improvements to enhance the web experience. Some notable additions include native support for video and audio playback, canvas for drawing graphics, new semantic elements, and improved form controls.

HTML5.1, HTML5.2, and Beyond: After HTML5, the development of HTML shifted to a more incremental update model. HTML5.1 was released in 2016, followed by HTML5.2 in 2017. The HTML working group continues to work on future versions, introducing new features, improving existing ones, and maintaining compatibility.

It's important to note that HTML is often accompanied by CSS (Cascading Style Sheets) and JavaScript to enhance the appearance and functionality of web pages. Together, these technologies form the foundation for building modern web applications.

What is HTML (FEATURES) ?

HTML (Hypertext Markup Language) provides a wide range of features for creating and structuring web pages. Here are some key features of HTML:

Document Structure: HTML allows you to define the structure of a web page using elements such as <html>, <head>, and <body>. These elements provide the foundation for organizing content.

Text Formatting: HTML offers various tags for formatting text, including headings (<h1> to <h6>), paragraphs (<p>), bold (<b>), italic (<i>), underline (<u>), and more. These tags help in visually enhancing and styling text.

Links: HTML enables the creation of hyperlinks using the <a> tag. Links allow users to navigate between different web pages or sections within a page.

Images and Media: HTML provides tags like <img> for embedding images and <video>, <audio> for including multimedia content on web pages. These tags allow you to display images and play audio or video files directly within the page.

Lists: HTML supports ordered lists (<ol>), unordered lists (<ul>), and definition lists (<dl>) for organizing content in a list format.

Tables: HTML includes the <table> element for creating tabular data. You can define rows (<tr>), table headers (<th>), and table cells (<td>) to structure and present data in a tabular form.

Forms: HTML provides form elements (<form>, <input>, <select>, <textarea>, etc.) for collecting user input. Forms allow users to enter data, make selections, and submit information to a server for processing.

Semantic Elements: HTML5 introduced a set of semantic elements (<header>, <footer>, <nav>, <article>, <section>, <aside>, etc.) that give meaning and structure to different sections of a web page. These elements help search engines and assistive technologies understand the content better.

Meta Information: HTML allows you to specify metadata about a web page using elements like <title>, <meta>, and <link>. These elements provide information such as the page title, character encoding, stylesheets, and more.

Embedding External Content: HTML supports the inclusion of external content within web pages through tags like <iframe>, which allows embedding external websites or media.

Applications of HTML?

HTML (Hypertext Markup Language) is widely used in various applications and industries. Here are some common applications of HTML:

Web Development: HTML is the backbone of web development. It is used to create the structure and content of web pages, allowing developers to define the layout, text, images, links, forms, and other elements of a website.

Website Design: HTML is essential for designing the visual appearance of websites. It provides the foundation for applying styles and layouts using CSS (Cascading Style Sheets). HTML works in conjunction with CSS to control the presentation and formatting of web pages.

Mobile App Development: HTML is a key component of hybrid mobile app development frameworks like Apache Cordova, PhoneGap, and Ionic. These frameworks utilize HTML, CSS, and JavaScript to create mobile applications that can run on multiple platforms.

Email Templates: HTML is commonly used for creating email templates. Email marketing campaigns, newsletters, and automated emails often rely on HTML to structure and style the content. HTML allows for the inclusion of images, links, and other interactive elements in emails.

E-learning and Online Education: HTML plays a significant role in creating e-learning platforms and online educational content. It enables the creation of interactive course materials, quizzes, multimedia presentations, and other learning resources delivered through web browsers.

Content Management Systems (CMS): Many CMS platforms, such as WordPress and Joomla, utilize HTML as the underlying markup language. HTML is used to define the structure and layout of web pages within the CMS, enabling users to create and manage content easily.

Web-based Applications: HTML is used in the development of web applications. It provides the structure for user interfaces, form input, and data presentation. HTML works together with JavaScript and server-side programming languages to create dynamic and interactive web applications.

Digital Publishing: HTML is employed in digital publishing to create web-based magazines, e-books, and interactive documents. HTML allows for the inclusion of multimedia elements, navigation menus, and interactive features, enhancing the reading experience.

Online Forms and Surveys: HTML forms are widely used for collecting data and user input on websites. HTML provides form elements such as text fields, checkboxes, radio buttons, dropdown menus, and submit buttons. These forms can be used for various purposes, including contact forms, registration forms, and surveys.

Web Scraping: HTML is essential for web scraping, which involves extracting data from websites. Web scraping tools and scripts analyze the HTML structure of web pages to extract specific information, such as prices, product details, news articles, and more.

Basics of HTML? / Requirements for HTML / How to Start HTML?

To start learning and using HTML, you can follow these steps:

Set Up Your Development Environment: Choose a text editor or an integrated development environment (IDE) to write your HTML code. Popular options include Visual Studio Code, Sublime Text, Atom, or Notepad++. Install your chosen editor/IDE and ensure it is ready for use.

Create an HTML File: Open a new file in your text editor or IDE and save it with an .html extension. This file will contain your HTML code.

Understand the Basic Structure: Every HTML file follows a basic structure. Start by adding the following lines of code to your HTML file:

```
<!DOCTYPE html>
<html>
<head>
    <title>My First HTML Page</title>
</head>
<body>


</body>
</html>
```

This code defines the document type, creates an HTML root element, and contains the head and body sections of your page.

Add Content to Your Web Page: Within the body tags, you can start adding content to your web page. For example, you can create headings, paragraphs, lists, and images. Here's an example of adding a heading and a paragraph:

```
<body>
  <h1>Welcome to PASS Education System</h1>
  <p>This is my first HTML page.</p>
</body>
```

Use HTML Tags: HTML uses tags to define elements and their attributes. Tags are enclosed in angle brackets (<>). Some common tags include <h1> to <h6> for headings, <p> for paragraphs, <a> for links,

<img> for images, and <ul> and <li> for lists. You can find a comprehensive list of HTML tags and their usage in HTML documentation or online tutorials.

Save and View Your Web Page: Save your HTML file and open it in a web browser. Double-click on the file or drag it onto the browser window to see how your page looks. You can make changes to your HTML code, save it, and refresh the browser to see the updated page.

Learn and Practice: HTML offers many features and tags to explore. Continue learning by referring to online tutorials, documentation, and resources. Practice by creating different elements, structuring content, applying styles with CSS, and adding interactivity with JavaScript.

Combine HTML with CSS and JavaScript: To enhance the appearance and functionality of your web page, learn how to use CSS for styling and JavaScript for interactivity. CSS allows you to control colors, fonts, layouts, and more, while JavaScript enables dynamic behavior and interaction.

Remember to validate your HTML code using tools like the W3C Markup Validation Service to ensure your code follows the standards.

Starting with simple examples and gradually progressing to more complex projects will help you gain proficiency in HTML. Practice, experiment, and explore the vast possibilities HTML offers for creating web pages and applications.

Basics Operators for HTML?

In HTML, there are no operators like those found in programming languages. HTML is a markup language used for structuring and presenting content on the web. It focuses on defining the structure and layout of elements rather than performing computations or operations. However, HTML does provide attributes and properties that allow you to control and manipulate elements in certain ways. Here are some common attributes and properties that are often used in HTML:

id Attribute: The id attribute allows you to uniquely identify an HTML element. It is typically used to target elements with CSS or JavaScript.

```
<div id="myElement">This is a div element with an id.</div>
```

class Attribute: The class attribute is used to assign one or more class names to an HTML element. It is commonly used for styling and selecting elements with CSS or JavaScript.

```
<p class="highlight">This is a paragraph element with a class.</p>
```

style Attribute: The style attribute allows you to apply inline CSS styles to an HTML element. You can use CSS property-value pairs within the style attribute to define the element's appearance.

```
<span style="color: red; font-size: 18px;">This is a span element with inline styles.</span>
```

src Attribute: The src attribute specifies the source URL or path for an external resource, such as an image or video.

```
<img src="image.jpg" alt="Image">
```

href Attribute: The href attribute is used in anchor (<a>) tags to define the target URL or destination of a hyperlink.

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

alt Attribute: The alt attribute provides alternative text for an image. It is displayed if the image fails to load or for accessibility purposes.

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

disabled Attribute: The disabled attribute is used to disable user interaction with an input or button element.

```
<button disabled>Click me</button>
```

These are just a few examples of attributes and properties used in HTML to control and manipulate elements. HTML's primary focus is on defining the structure and content of web pages, while CSS and JavaScript are used for more advanced styling, interactivity, and computations.

Basics Variables for HTML?

In HTML, there is no concept of variables like in programming languages. HTML (Hypertext Markup Language) is a markup language used for structuring and presenting content on the web. It focuses on defining the structure of elements and their relationships, rather than storing and manipulating data.

However, HTML does provide attributes and properties that can be used to associate values with elements. These attributes and properties allow you to provide additional information or configuration for elements. Here are some common attributes and properties that can be considered as representing values:

value Attribute: This attribute is often used with form elements such as <input> and <button>. It represents the current value of an input field or the displayed text on a button.

```
<input type="text" value="Initial value">
<button value="Submit">Submit</button>
```

data-* Attributes: The data-* attributes are custom attributes that allow you to store custom data with an element. These attributes can be named according to your requirements and used to associate specific values or information.

```
<div data-user-id="123" data-role="admin">...</div>
```

alt Attribute: The alt attribute, commonly used with the <img> tag, represents alternative text for an image. It can be considered as a value associated with the image, providing a textual description.

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

Content Text: The text content placed between opening and closing tags represents the value or data associated with an element. For example, the text content within a paragraph (<p>) element represents the content or value of that paragraph.

```
<p>This is the content of the paragraph.</p>
```

href Attribute: The href attribute, used with the <a> (anchor) tag, represents the URL or destination associated with a hyperlink.

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

It's important to note that HTML is primarily a structural language and doesn't provide mechanisms for storing and manipulating data like programming languages do. For dynamic and interactive functionality, HTML is often combined with CSS (Cascading Style Sheets) for styling and JavaScript for programming logic and data manipulation.

Basics Functions for HTML?

HTML (Hypertext Markup Language) is primarily a markup language and doesn't provide built-in functions like programming languages do. However, HTML does offer a variety of elements and attributes that enable certain functionalities. These functionalities can be considered as "basic functions" of HTML. Here are some examples:

Hyperlinks: HTML provides the <a> (anchor) tag for creating hyperlinks. It allows you to link to other web pages, sections within the same page, email addresses, or specific actions.

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

Lists: HTML supports ordered lists (<ol>), unordered lists (<ul>), and definition lists (<dl>) for creating lists of items.

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

Forms: HTML provides form elements (<form>, <input>, <select>, <textarea>, etc.) for creating interactive forms to collect user input.

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

Images and Multimedia: HTML allows you to embed images (<img>) and multimedia elements such as audio (<audio>) and video (<video>) in web pages.

```
<img src="image.jpg" alt="Image">
<audio src="audio.mp3" controls></audio>
<video src="video.mp4" controls></video>
```

Semantic Elements: HTML5 introduced semantic elements (<header>, <footer>, <nav>, <article>, <section>, <aside>, etc.) that provide meaningful structure to different sections of a web page. These elements aid in better organization and understanding of content.

```
<header>
```

```
  <h1>Website Header</h1>
</header>

<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>

<section>
  <h2>About Us</h2>
  <p>Content goes here...</p>
</section>
```

Tables: HTML allows the creation of tables (<table>) to organize and present tabular data.

```
<table>
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>John</td>
    <td>25</td>
  </tr>
  <tr>
    <td>Jane</td>
```

```
    <td>30</td>
  </tr>
</table>
```

These are some of the basic functionalities provided by HTML elements and attributes. It's important to note that HTML is often combined with CSS (Cascading Style Sheets) for styling and layout, as well as JavaScript for dynamic and interactive behaviors. Together, these technologies allow for more advanced functionality and interactivity on the web.

How to Run HTML Code?

To run HTML code, you need a web browser. Here's a step-by-step guide on how to run your HTML code:

Create an HTML file: Use a text editor (e.g., Notepad, Visual Studio Code, Sublime Text) to create a new file and save it with a .html extension. For example, you can save the file as "index.html".

Write your HTML code: Open the HTML file in your text editor and start writing your HTML code within the file. You can include HTML elements, tags, attributes, and content as needed to structure and present your web page.

```
<!DOCTYPE html>
<html>
<head>
    <title>My First HTML Page</title>
</head>
<body>
    <h1>Welcome to My Website</h1>
```

```
    <p>This is my first HTML page.</p>
</body>
</html>
```

Save the HTML file: After writing your HTML code, save the file.

Open the HTML file in a web browser: Double-click on the HTML file or right-click and select "Open with" to choose a web browser to open the file. The web browser will render and display your HTML page.

Alternatively, you can open a web browser and use the "File" or "Open" option from the browser's menu to browse and select the HTML file to view.

View and test your HTML page: The web browser will display your HTML page based on the code you wrote. You can interact with the page, follow links, submit forms, and see how your content is rendered.

Make changes and refresh the browser: If you make changes to your HTML file, save the file and then refresh the web browser to see the updated version of your page.

By following these steps, you can run and view your HTML code in a web browser. This allows you to see how your HTML elements and content are displayed and interact with your web page.

HTML Projects :

- Simple Calculator

- Snake Game

- Make Your Own for Practice

Project: Simple Calculator

Here's an example of a simple calculator using HTML. This calculator can perform basic arithmetic operations such as addition, subtraction, multiplication, and division.

```
<!DOCTYPE html>
<html>
<head>
    <title>Simple Calculator</title>
    <style>
        .container {
            width: 300px;
            padding: 20px;
            background-color: #f2f2f2;
            border-radius: 5px;
        }

        .form-group {
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Simple Calculator</h1>
        <form>
            <div class="form-group">
                <label for="num1">Number 1:</label>
                <input type="number" id="num1" required>
            </div>
            <div class="form-group">
                <label for="num2">Number 2:</label>
                <input type="number" id="num2" required>
```

```
            </div>
            <div class="form-group">
                <label for="operator">Operator:</label>
                <select id="operator">
                    <option value="+">+</option>
                    <option value="-">-</option>
                    <option value="*">*</option>
                    <option value="/">/</option>
                </select>
            </div>
            <div class="form-group">
                <button type="button"
onclick="calculate()">Calculate</button>
            </div>
            <div class="form-group">
                <label for="result">Result:</label>
                <input type="text" id="result" readonly>
            </div>
        </form>
    </div>

    <script>
        function calculate() {
            var num1 = parseFloat(document.getElementById('num1').value);
            var num2 = parseFloat(document.getElementById('num2').value);
            var operator = document.getElementById('operator').value;
            var result;

            if (operator === '+') {
                result = num1 + num2;
```

```
            } else if (operator === '-') {
                result = num1 - num2;
            } else if (operator === '*') {
                result = num1 * num2;
            } else if (operator === '/') {
                result = num1 / num2;
            }

            document.getElementById('result').value = result;
        }
    </script>
</body>
</html>
```

Let's go through the code and its explanation:

HTML structure: The HTML code is enclosed within the <html>, <head>, and <body> tags. Inside the <head> section, we set the title of the web page and define some CSS styles for styling the calculator.

Calculator form: We create a element with the class "container" to hold the calculator form. The form contains two number input fields (<input type="number">) for entering the operands (number1 and number2). We also have a select dropdown (<select>) for selecting the operator and a calculate button (<button>) to trigger the calculation.

JavaScript function: We define a JavaScript function called calculate() that is invoked when the Calculate button is clicked. Inside the function, we retrieve the values of the operands and the selected operator from the respective elements using document.getElementById().

Calculation: Based on the selected operator, we perform the corresponding arithmetic operation using if-else statements. The result is stored in the result variable.

Displaying the result: Finally, we set the value of the result in the <input> field with the id "result" using document.getElementById().

By using this HTML code, you can create a simple calculator that performs basic arithmetic operations based on user input.

Project: Snake Game

Creating a full-fledged snake game using HTML alone is not practical since HTML is primarily a markup language for structuring content. However, you can create a basic version of the snake game using HTML, CSS, and JavaScript. The game will run within the HTML page and utilize CSS for styling and JavaScript for game logic.

```
<!DOCTYPE html>
<html>
<head>
    <title>Snake Game</title>
    <style>
        #game-board {
            width: 400px;
            height: 400px;
            border: 1px solid #000;
            position: relative;
        }

        .snake {
            width: 20px;
            height: 20px;
            background-color: green;
            position: absolute;
        }
```

```
        .food {
            width: 20px;
            height: 20px;
            background-color: red;
            position: absolute;
        }
    </style>
</head>
<body>
    <div id="game-board">
        <div id="snake" class="snake"></div>
        <div id="food" class="food"></div>
    </div>

    <script>
        // Game variables
        var snake = [{ x: 200, y: 200 }];
        var food = { x: 0, y: 0 };
        var direction = "right";
        var interval;

        // Function to generate random position for food
        function generateFoodPosition() {
            var x = Math.floor(Math.random() * 20) * 20;
            var y = Math.floor(Math.random() * 20) * 20;
            return { x: x, y: y };
        }

        // Function to update the game state
```

```
        function update() {
            // Update snake position
            var head = { x: snake[0].x, y: snake[0].y };
            if (direction === "right") head.x += 20;
            if (direction === "left") head.x -= 20;
            if (direction === "up") head.y -= 20;
            if (direction === "down") head.y += 20;
            snake.unshift(head);

            // Check if snake eats the food
            if (head.x === food.x && head.y === food.y) {
                food = generateFoodPosition();
            } else {
                snake.pop();
            }

            // Check for game over conditions
            if (
                head.x < 0 ||
                head.x >= 400 ||
                head.y < 0 ||
                head.y >= 400 ||
                checkCollision(head)
            ) {
                clearInterval(interval);
                alert("Game Over!");
                return;
            }

            // Update game board
```

```
            var gameBoard = document.getElementById("game-board");
            gameBoard.innerHTML = "";
            snake.forEach(function (segment) {
                var snakeElement = document.createElement("div");
                snakeElement.className = "snake";
                snakeElement.style.left = segment.x + "px";
                snakeElement.style.top = segment.y + "px";
                gameBoard.appendChild(snakeElement);
            });

            var foodElement = document.createElement("div");
            foodElement.className = "food";
            foodElement.style.left = food.x + "px";
            foodElement.style.top = food.y + "px";
            gameBoard.appendChild(foodElement);
        }

        // Function to check collision with the snake's body
        function checkCollision(head) {
            return snake.slice(1).some(function (segment) {
                return segment.x === head.x && segment.y === head.y;
            });
        }

        // Function to handle key events for changing direction
        function handleKeyPress(event) {
            if (event.key === "ArrowUp" && direction !== "down") {
                direction = "up";
            } else if (event.key === "ArrowDown" && direction !== "up") {
                direction = "down";
```

```
            } else if (event.key === "ArrowLeft" && direction !== "right") {
                direction = "left";
            } else if (event.key === "ArrowRight" && direction !== "left") {
                direction = "right";
            }
        }

        // Start the game
        function startGame() {
            food = generateFoodPosition();
            interval = setInterval(update, 200);
        }

        // Event listener for key presses
        document.addEventListener("keydown", handleKeyPress);

        // Start the game when the page loads
        document.addEventListener("DOMContentLoaded", startGame);
    </script>
</body>
</html>
```

Now let's go through the code and its explanation:

HTML structure: The HTML code contains a element with the id "game-board" that represents the game area. It also contains a snake element ( ) and a food element ( ) within the game board.

CSS styling: The CSS styles define the appearance of the game board, snake, and food elements. The game board has a fixed width and height and is bordered. The snake and food elements are square blocks with different background colors.

JavaScript game logic: The JavaScript code contains the game logic. It includes variables to store the snake's position, food position, and the current direction of the snake. The generateFoodPosition() function generates a random position for the food within the game board. The update() function updates the game state, moves the snake, checks for collisions, and updates the game board on each iteration. The checkCollision() function checks if the snake collides with itself. The handleKeyPress() function handles the key events for changing the direction of the snake. The startGame() function initializes the game by generating the initial food position and starting the game loop. Event listeners are added to listen for key presses and start the game when the page loads.

By using this HTML, CSS, and JavaScript code, you can create a basic version of the snake game where the snake moves, eats food, and grows. The game loop updates the game state, checks for collisions, and renders the game board accordingly. Note that this implementation is a simplified version and does not include features such as scorekeeping or advanced gameplay mechanics.

Best Of Luck

Don't Stop Until You Have Done Completely and Truly.