**UNIT- III**

**HTML**

**What is Programming?**

Programming, in the context of computer science and software development, refers to the process of designing, writing, testing, and maintaining computer programs. A computer program is a set of instructions that a computer can execute to perform a specific task or solve a particular problem. These instructions are written in a programming language, which is a formal and structured way of communicating with a computer.

**Programming involves several key activities:**

1. **Problem Solving:** Programming begins with identifying a problem or a task that can be solved or automated through the use of a computer program.
2. **Algorithm Design:** Programmers design algorithms, which are step-by-step procedures for solving a problem. These algorithms serve as the blueprint for the computer program.
3. **Writing Code:** The programmer writes the code, which is a textual representation of the algorithm using a programming language. This code is a set of instructions that the computer can understand and execute.
4. **Testing and Debugging:** After writing the code, it's essential to test it to ensure it works correctly. If errors or bugs are found, they need to be identified and fixed through a process called debugging.
5. **Optimization:** Programmers may optimize their code to make it more efficient in terms of performance, memory usage, or other factors.
6. **Documentation:** Proper documentation is crucial for understanding and maintaining the code. This includes comments within the code to explain its functionality and overall documentation that describes how the program works.
7. **Maintenance and Updates:** Software programs require ongoing maintenance to fix bugs, add new features, or adapt to changing requirements.

Programming can be done using a wide variety of programming languages, each with its own syntax and capabilities. Some popular programming languages include Python, Java, C++, JavaScript, and many others. The choice of programming language depends on the specific task, platform, and developer's preferences.

Programming is a fundamental skill in the field of computer science and is used to create software for a wide range of applications, from web development and mobile apps to scientific research, artificial intelligence, and system administration. It plays a crucial role in driving technological advancements and solving complex problems in various domains.

**What is Web Based Programming? Which Programming Languages are used to develop Web Based Applications?**

Web-based programming, often referred to as web development or web programming, is the process of creating applications or websites that are accessible through web browsers. It involves using various technologies, programming languages, and tools to build web applications that can be hosted on web servers and accessed by users via the internet. Web-based applications encompass a wide range of services, from simple websites to complex web applications and online services.

Some of the key programming languages and technologies commonly used in web-based programming include:

1. **HTML (Hypertext Markup Language):** HTML is the standard markup language used to create the structure and content of web pages. It defines the layout and presentation of text, images, links, and other elements on a webpage.
2. **CSS (Cascading Style Sheets):** CSS is used for styling and formatting web content, including fonts, colors, layout, and responsive design. It complements HTML by defining the visual presentation of web pages.
3. **JavaScript:** JavaScript is a versatile and widely used scripting language for adding interactivity and functionality to web pages. It allows developers to create dynamic and responsive web applications, enabling features like client-side form validation, animations, and real-time updates.
4. **PHP:** PHP is a server-side scripting language often used for web development. It is commonly used to build dynamic websites and web applications, interact with databases, and perform server-side processing.
5. **Python:** Python can be used for web development through frameworks like Django and Flask. These frameworks simplify the process of building web applications and are known for their readability and maintainability.
6. **Ruby:** Ruby on Rails is a web development framework that uses the Ruby programming language. It follows the principles of convention over configuration (CoC) and don't repeat yourself (DRY), making it efficient for building web applications.
7. **Java:** Java is used for server-side web development, particularly in enterprise-level applications. Technologies like JavaServer Pages (JSP) and Java Servlets are commonly used for building web applications.
8. **ASP.NET:** Developed by Microsoft, ASP.NET is a framework for building web applications using languages like C# or Visual Basic. It is primarily used on the Windows platform.
9. **Node.js:** Node.js is a runtime environment that allows developers to run JavaScript on the server side. It's particularly well-suited for building real-time and scalable web applications.
10. **SQL (Structured Query Language):** SQL is essential for database management in web applications. It's used to create, read, update, and delete data in relational databases.
11. **Front-End Frameworks:** Various front-end frameworks and libraries, such as React, Angular, and Vue.js, are used to simplify the development of interactive and responsive user interfaces.
12. **Back-End Frameworks:** Frameworks like Ruby on Rails, Django, Laravel (for PHP), Express.js (for Node.js), and many others provide a structured approach to building the server-side components of web applications.

The choice of programming languages and technologies depends on the specific requirements of the web application, the development team's expertise, and other factors. Modern web development often involves a combination of front-end and back-end technologies, as well as databases and other infrastructure components to deliver a complete web-based solution.

**What is HTML?**

HTML, or Hypertext Markup Language, is a standard markup language used for creating and structuring content on the World Wide Web. It is the backbone of web pages and is responsible for defining the structure and elements of a web document, including text, images, links, forms, and more. HTML uses a system of tags to mark up and format content, allowing web browsers to render it in a visually pleasing and interactive manner.

Here are some key characteristics and aspects of HTML:

1. **Markup Language:** HTML is a markup language, not a programming language. It doesn't have the logic or computational capabilities of programming languages. Instead, it defines the structure and presentation of content on a web page.
2. **Tags:** HTML content is enclosed within various HTML tags, which are enclosed in angle brackets (< >). Tags are used to define elements such as headings, paragraphs, images, links, lists, tables, and more.
3. **Hierarchy:** HTML documents are organized hierarchically, with a root **<html>** element containing two main sections: the **<head>** section for metadata and the **<body>** section for the visible content.
4. **Attributes:** Many HTML elements can have attributes that provide additional information about the element. For example, the **<a>** (anchor) element has an "href" attribute to specify the URL a link points to.
5. **Semantic Elements:** HTML5 introduced semantic elements like **<header>**, **<nav>**, **<article>**, and **<footer>**, which provide meaning to the structure of the document and make it more accessible and SEO-friendly.
6. **Cross-Browser Compatibility:** HTML is designed to be rendered by web browsers consistently across different platforms and devices, ensuring a standardized user experience.
7. **Versioning:** HTML has evolved over the years, with HTML5 being the latest and most widely used version. HTML5 introduced many new features and elements for multimedia, forms, and improved semantics.
8. **Integration with Other Technologies:** HTML is often used in conjunction with CSS (Cascading Style Sheets) for styling web content and JavaScript for adding interactivity and functionality to web pages.

Here's a simple example of HTML code that creates a basic web page structure:

*<!DOCTYPE html>*

*<html>*

*<head> <title>My Web Page</title> </head>*

*<body> <h1>Welcome to My Web Page</h1>*

*<p>This is a simple example of HTML content.</p>*

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

*</body>*

*</html>*

This HTML code defines a web page with a title, a heading, a paragraph, and a hyperlink. When rendered by a web browser, it would display the content as described in the HTML markup. HTML serves as the foundation for creating web pages and web applications on the internet.

**What are HTML Tags?**

HTML tags are a fundamental part of the Hypertext Markup Language (HTML). They are used to define and structure the content within an HTML document, making it possible to create web pages with text, images, links, forms, and more. HTML tags are enclosed in angle brackets ("<" and ">"), and most come in pairs: an opening tag and a closing tag, which helps define the beginning and end of an element. The closing tag typically has a forward slash ("/") before the tag name.

Here are some commonly used HTML tags:

1. **<html>**: The root element that encapsulates the entire HTML document.
2. **<head>**: Contains metadata about the web page, such as the title, character set, and links to external resources like stylesheets and scripts.
3. **<title>**: Sets the title of the web page, which is displayed in the browser's title bar or tab.
4. **<body>**: Contains the visible content of the web page, including text, images, links, and more.
5. **<h1>, <h2>, <h3>, <h4>, <h5>, <h6>**: Headings of varying levels of importance, with **<h1>** being the highest.
6. **<p>**: Defines a paragraph of text.
7. **<a>**: Creates hyperlinks, allowing users to navigate to other web pages or resources.
8. **<img>**: Embeds images in a web page.
9. **<ul>**: Defines an unordered list, typically used for creating bullet-point lists.
10. **<ol>**: Defines an ordered list, often used for numbered lists.
11. **<li>**: Represents a list item within **<ul>** or **<ol>** elements.
12. **<br>**: Inserts a line break, typically used to break text or content to a new line.
13. **<hr>**: Inserts a horizontal rule (a horizontal line) to separate content.
14. **<div>**: A generic container for grouping and styling content.
15. **<span>**: A generic inline container for styling or scripting individual pieces of text or content.
16. **<form>**: Defines a form for collecting user input.
17. **<input>**: Represents form input elements like text fields, radio buttons, checkboxes, etc.
18. **<button>**: Creates a clickable button within a form or web page.
19. **<table>**: Defines an HTML table for organizing data in rows and columns.
20. **<tr>**: Defines a table row within a **<table>** element.
21. **<td>**: Defines a table cell within a **<tr>** element.

These are just a few examples of HTML tags. HTML provides a wide range of tags to structure content and define its behavior on the web. The proper use of HTML tags is essential for creating well-structured and semantically meaningful web pages that are both readable by browsers and accessible to users.

**What are empty Tags?**

Empty tags, also known as self-closing tags or void elements, are a subset of HTML tags that do not have a closing tag. They are used to insert content or elements that don't contain any nested content or text. Instead, their attributes define their behavior or appearance. Empty tags are typically written with a forward slash before the closing angle bracket. Some common empty tags include:

1. **<br>**: The line break tag is used to insert a line break, forcing content that follows it to appear on a new line. It does not have a closing tag.

Example:

<p>This is some text.<br>This is on a new line.</p>

1. **<hr>**: The horizontal rule tag is used to insert a horizontal line or thematic break. Like **<br>**, it doesn't have a closing tag.

Example:

<p>Some content above<hr>Some content below.</p>

1. **<img>**: The image tag is used to insert images into web pages. It does not have a closing tag but includes attributes like **src** to specify the image source and **alt** for alternative text.

Example:

<img src="image.jpg" alt="An example image">

1. **<input>**: The input tag is used within forms to create various types of form fields, such as text inputs, radio buttons, checkboxes, etc. It does not have a closing tag.

Example:

<input type="text" name="username">

1. **<meta>**: The meta tag is used to provide metadata about the HTML document, such as character encoding or page description. It doesn't have a closing tag.

Example:

<meta charset="UTF-8">

1. **<link>**: The link tag is used to include external resources like stylesheets. It does not have a closing tag.

Example:

<link rel="stylesheet" type="text/css" href="styles.css">

These empty tags are essential for various purposes in web development, allowing you to control the layout, styling, and behavior of web content. Their lack of a closing tag simplifies the structure of HTML and makes it easier to create and manage web documents.

**What is an Element in HTML?**

In HTML, an element is a fundamental building block that defines the structure and content of a webpage. An element is created using HTML tags and consists of various parts, including the opening tag, the closing tag (in some cases), and the content or data that is enclosed between these tags. Elements can be used to describe the structure, semantics, and presentation of the content on a webpage.

Here's a basic breakdown of the parts of an HTML element:

1. **Opening Tag**: The opening tag is the part that marks the beginning of an element and is enclosed in angle brackets **< >**. It specifies the name of the element and may contain attributes that provide additional information about the element.

Example of an opening tag:

<p>

1. **Closing Tag**: Many HTML elements have a corresponding closing tag that marks the end of the element. The closing tag is also enclosed in angle brackets and is preceded by a forward slash **/**.

Example of a closing tag for a paragraph element:

</p>

1. **Content**: The content is the actual information or data that is placed between the opening and closing tags. The content can be text, other elements, images, links, or any other type of data.

Example of content within a paragraph element:

<p>This is a paragraph of text.</p>

HTML elements can vary widely in their purpose and appearance, ranging from structural elements like headings (**<h1>**, **<h2>**, etc.) and paragraphs (**<p>**) to multimedia elements like images (**<img>**) and links (**<a>**), and even interactive elements like forms (**<form>**) and buttons (**<button>**).

Elements are used to define the structure of a webpage and provide semantic meaning to the content. They are the basic building blocks that web browsers use to render and display web pages, and they can be styled and manipulated using CSS and JavaScript to achieve the desired appearance and functionality.

**What are attributes in HTML?**

In HTML, attributes are additional pieces of information that can be added to an HTML element to provide extra context, configuration, or properties to that element. Attributes are defined within the opening tag of an element and are typically written as name-value pairs, with the attribute name followed by an equal sign and the attribute value enclosed in quotation marks. Here's a breakdown of how attributes work in HTML:

1. **Attribute Name**: This is the name of the attribute and indicates what type of information it provides. Attribute names are not case-sensitive, so they can be written in uppercase or lowercase letters (e.g., "class" and "CLASS" are equivalent). However, it's a common convention to use lowercase for attribute names in HTML.
2. **Attribute Value**: This is the value associated with the attribute and is enclosed in either single or double quotation marks. The attribute value provides the specific information or data associated with the attribute. The attribute value should be enclosed in quotes, but the quotes themselves can be single or double, as long as they match (e.g., **value="example"** or **value='example'**).

Here's an example of an HTML element with attributes:

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

In this example:

* **<a>** is the HTML element (an anchor or link element).
* **href** is an attribute that specifies the destination URL.
* **"https://www.example.com"** is the value of the **href** attribute.
* **title** is another attribute that provides additional information (a tooltip) when a user hovers over the link.
* **"Visit Example Website"** is the value of the **title** attribute.

Commonly used attributes in HTML include **class**, **id**, **src**, **alt**, **width**, **height**, **href**, **title**, and many more, depending on the type of element and its purpose.

Attributes play a crucial role in defining the behavior, appearance, and functionality of HTML elements, and they are often used in combination with CSS and JavaScript to create interactive and visually appealing web pages.

**Explain different attributes available in HTML?**

HTML provides a wide range of attributes that can be used with various HTML elements to define their behavior, appearance, and functionality. Here are some commonly used attributes in HTML, categorized by their primary purposes:

**1. Global Attributes**:

* These attributes can be used with most HTML elements and have general applications.
  + **id**: Provides a unique identifier for an element.
  + **class**: Assigns one or more class names to an element, enabling CSS styling and JavaScript selection.
  + **style**: Specifies inline CSS styles for an element.
  + **title**: Provides additional information about an element (often used for tooltips).
  + **data-\***: Allows custom data attributes for scripting and styling.

**2. Attributes for Hypertext Links**:

* These attributes are used with anchor **<a>** elements for creating hyperlinks.
  + **href**: Specifies the URL of the linked resource.
  + **target**: Controls where the linked content opens (e.g., in a new tab or window).
  + **rel**: Defines the relationship between the current document and the linked resource (e.g., "stylesheet," "nofollow").

**3. Attributes for Images**:

* These attributes are used with the **<img>** element for displaying images.
  + **src**: Specifies the image source URL.
  + **alt**: Provides alternative text for the image (for accessibility).
  + **width** and **height**: Define the dimensions of the image.

**4. Attributes for Forms**:

* These attributes are used with form-related elements (**<form>**, **<input>**, **<select>**, **<textarea>**) to create and handle forms.
  + **action**: Specifies the URL to which the form data is sent when submitted.
  + **method**: Defines the HTTP method used for form submission (e.g., "GET" or "POST").
  + **name**: Assigns a name to the form control.
  + **type**: Specifies the type of input element (e.g., "text," "checkbox," "submit").
  + **value**: Sets the initial or default value for a form control.
  + **placeholder**: Provides a short hint for the expected value in form controls.
  + **required**: Indicates that the input field must be filled out.

**5. Attributes for Tables**:

* These attributes are used with the **<table>** element to structure and format tables.
  + **border**: Sets the border width around the table (deprecated in HTML5).
  + **cellpadding** and **cellspacing**: Define spacing inside and between table cells.
  + **colspan** and **rowspan**: Merge table cells horizontally or vertically.

**6. Media Attributes**:

* These attributes are used with media elements like **<audio>** and **<video**.
  + **controls**: Adds playback controls to the media element.
  + **autoplay**: Automatically starts media playback.
  + **loop**: Specifies whether the media should loop.
  + **poster**: Sets an image as a poster frame for video elements.

**7. Form Control Attributes**:

* These are specific to form control elements (e.g., **<input>**, **<select>**, **<textarea>)** and include attributes like **disabled**, **readonly**, **min**, **max**, and more for customizing form behavior.

This list is not exhaustive, and there are many more attributes available in HTML for various elements. The choice of attributes depends on the specific requirements and the element being used. You can refer to the official HTML documentation for a complete list of attributes and their descriptions.

Top of Form

**Difference between Name, ID and Class attribute in HTML**

In HTML, the **name**, **id**, and **class** attributes are used to provide additional information to HTML elements, and they serve different purposes:

1. **Name Attribute**:
   * The **name** attribute is primarily used with form elements such as **<input>**, **<select>**, and **<textarea>**.
   * It defines the name of the input element, which is sent to the server when the form is submitted.
   * In the context of form elements, the **name** attribute is used to identify the form field and associate it with a value that the user enters or selects. This is how data is typically sent to the server when a form is submitted.

Example:

<input type="text" name="username" />

1. **ID Attribute**:
   * The **id** attribute is used to give a unique identifier to an HTML element within a webpage.
   * It is used for scripting purposes, to select and manipulate specific elements using JavaScript or CSS.
   * Each element on a page should have a unique **id** attribute value. This uniqueness is important to ensure that scripting and styling can target specific elements.

Example:

<div id="header">This is the header</div>

1. **Class Attribute**:
   * The **class** attribute is used to define one or more class names for an HTML element.
   * It is used for styling purposes with CSS and for grouping multiple elements that share common styling or behavior.
   * Multiple elements can share the same class name, and multiple class names can be applied to a single element by separating them with spaces.

Example:

<p class="important note">This is an important note.</p>

In summary, the **name** attribute is primarily used with form elements for data submission, the **id** attribute is used to uniquely identify elements for scripting and styling, and the **class** attribute is used for grouping elements and applying common styles to multiple elements. It's important to use these attributes appropriately and understand their distinct purposes in web development.

Explain important tags with its important attributes/elements

1. **<!DOCTYPE>**:
   * This is not a tag but a declaration used at the very beginning of an HTML document to specify the document type and version. It is crucial for defining the version of HTML you are using.
2. **<html>**:
   * The root element of an HTML document.
   * Common attributes: **lang** for specifying the language of the document.
3. **<head>**:
   * Contains metadata about the document.
   * Common elements: **<title>** for setting the title of the web page, **<meta>** for defining character encoding and other metadata, **<link>** for linking to external resources like stylesheets, and **<script>** for including scripts.
4. **<meta>**:
   * Provides metadata about the HTML document.
   * Important attributes: **charset** to define the character encoding, **name** and **content** for various metadata, such as **name="viewport"** for responsive design.
5. **<link>**:
   * Used to link external resources to the HTML document, often used for stylesheets.
   * Important attributes: **rel** specifies the relationship of the linked resource, **href** specifies the URL of the resource, and **type** defines the resource's MIME type.
6. **<style>**:
   * Contains CSS rules for styling the document.
   * Common attribute: **type** to specify the style language (usually "text/css").
7. **<script>**:
   * Used to include JavaScript code in the HTML document.
   * Common attributes: **src** for specifying an external script file and **type** for indicating the script language.
8. **<body>**:
   * Contains the visible content of the web page.
   * Common attributes: **bgcolor** (deprecated) or **style** for setting the background color or background image.
9. **<h1>, <h2>, <h3>, ... <h6>**:
   * Heading tags for creating hierarchical headings.
   * No specific attributes, but can be styled with CSS.
10. **<p>**:
    * Represents a paragraph of text.
    * Common attributes: **class** and **id** for styling and scripting.
11. **<a>**:
    * Creates hyperlinks to other web pages or resources.
    * Important attributes: **href** for specifying the destination URL, and **target** to control how the link is opened (e.g., in a new window/tab).
12. **<img>**:
    * Embeds images in the document.
    * Key attributes: **src** for the image source URL, **alt** for alternative text (important for accessibility), **width** and **height** for specifying dimensions.
13. **<ul>, <ol>, <li>**:
    * Used for creating unordered lists, ordered lists, and list items.
    * No specific attributes for **<ul>** and **<ol**, but **<li>** can have attributes like **value**.
14. **<table>, <tr>, <th>, <td>**:
    * Elements for creating tables.
    * Common attributes include **border**, **cellspacing**, **cellpadding** for table styling.
15. **<form>**, **<input>**, **<button>**, **<select>**, **<textarea>**:
    * Elements for creating forms and form controls.
    * Attributes such as **action**, **method**, **name**, **type**, **value**, and others are essential for defining form behavior and handling user input.
16. **<div>**:
    * A generic container for grouping and styling content.
    * Common attributes: **class** and **id** for styling and scripting.
17. **<span>**:
    * A generic inline container for styling specific text or elements.
    * Common attributes: **class** and **id** for styling and scripting.
18. **<hr>**:
    * Represents a horizontal rule or thematic break in content.
    * Common attribute: **color** (deprecated) for specifying the line color, and **size** (deprecated) for line thickness.
19. **<br>**:
    * Inserts a line break or new line within text.
    * No specific attributes.
20. **<b>, <strong>**:
    * Used to make text bold.
    * No specific attributes, but they have semantic differences (**<strong>** represents strong importance).
21. **<i>, <em>**:
    * Used to italicize text.
    * No specific attributes, but they have semantic differences (**<em>** represents emphasized text).
22. **<sup>, <sub>**:
    * For superscript and subscript text, respectively.
    * No specific attributes.
23. **<iframe>**:
    * Embeds an external webpage or content within the current page.
    * Important attributes: **src** for the source URL, **width**, **height**, and **allowfullscreen** for controlling iframe dimensions and behavior.
24. **<audio>, <video>**:
    * Used to embed audio and video content.
    * Important attributes: **src** for the media source, **controls** for adding playback controls, **autoplay**, **loop**, **poster**, and more.
25. **<nav>**:
    * Represents a section of the page containing navigation links.
    * Common attributes: **role** (for accessibility) and **aria-label** (to provide an accessible label).
26. **<aside>**:
    * Contains content that is tangentially related to the surrounding content.
    * Common attributes: **class** and **id**.

Explain form tag with input tag and its elements

The **<form>** element in HTML is used to create a form that allows users to input data, which can then be submitted to a web server for processing. Within the **<form>** element, you typically use various form controls, including the **<input>** element, to collect data from users. Here's an explanation of the **<form>** element along with the **<input>** element and its important attributes and elements:

**<form> Element**:

* The **<form>** element is used to define the structure and behavior of a form on a webpage.
* Important attributes:
  + **action**: Specifies the URL to which the form data is sent when it's submitted.
  + **method**: Defines the HTTP method used for form submission, usually "GET" or "POST."
* Other common attributes include **name**, **enctype**, and **target** for controlling form behavior.

Example of a simple **<form>**:

<form action="/submit" method="post"> <!-- Form controls go here --> </form>

**<input> Element**:

* The **<input>** element is a versatile form control that allows users to input various types of data.
* Important attributes:
  + **type**: Specifies the type of input (e.g., "text" for text input, "checkbox" for checkboxes, "radio" for radio buttons, etc.).
  + **name**: Assigns a name to the input element, which is used to identify the input data when the form is submitted.
  + **value**: Sets the initial or default value of the input.
  + **placeholder**: Provides a short hint for the expected value.
  + **required**: Indicates that the input is mandatory and must be filled out.
  + **readonly**: Makes the input field read-only, preventing user editing.
  + **disabled**: Disables the input, making it non-interactive.
  + **max** and **min**: Define the maximum and minimum values for numeric input types.
  + **pattern**: Specifies a regular expression pattern for validation.

1. **Text: Example of an <input> for text input:**

<input type="text" name="username" placeholder="Enter your username" required>

1. **Check Box : Example of an <input> for a checkbox:**

<input type="checkbox" name="subscribe" value="yes" id="subscribe"> <label for="subscribe">Subscribe to our newsletter</label>

Additional elements and attributes can be used in combination with the **<form>** and **<input>** elements to create more complex and interactive forms. Other form elements include **<textarea>**, **<select>**, and **<button>**, each with their own set of attributes for customizing their behavior.

Forms are essential for gathering user input, conducting surveys, and facilitating user interactions on websites. They can be processed on the server side with a scripting language like PHP or by using JavaScript for client-side validation and interaction.

1. **. Radio Buttons (<input type="radio">):**

* Radio buttons allow users to select one option from a list of choices.

Important attributes:

* + **type="radio"** to specify the input type.
  + **name** to group related radio buttons.
  + **value** to define the value of the selected option.

Example:

<input type="radio" name="gender" value="male"> Male

<input type="radio" name="gender" value="female"> Female

**4. Password Input (<input type="password">):**

* Password inputs mask the entered text, providing security for sensitive data.

Important attributes:

* + **type="password"** to specify the input type.
  + **name** for identifying the input field.

Example:

<input type="password" name="password" placeholder="Password">

**5. Date (<input type="date">) and Date-Time Local (<input type="datetime-local">):**

* These input types allow users to select dates and date-time values.

Important attributes:

* + **type="date"** or **type="datetime-local"** to specify the input type.
  + **name** for identifying the input field.

Example:

<input type="date" name="birthdate"> <input type="datetime-local" name="eventTime">

**6. Email (<input type="email">):**

* The email input type is used for collecting email addresses.

Important attributes:

* + **type="email"** to specify the input type.
  + **name** for identifying the input field.

Example:

<input type="email" name="userEmail" placeholder="Email">

**7. Reset Button (<input type="reset">):**

* A reset button resets form fields to their default values.

Important attributes:

* + **type="reset"** to create a reset button.

Example:

<input type="reset" value="Reset Form">

**8. Select (<select>) and Option (<option>):**

* The **<select>** element creates dropdown lists, and **<option>** elements define individual options within the list.

Important attributes:

* + **name** for identifying the input field.

Example:

<select name="country">

<option value="usa">USA</option>

<option value="canada">Canada</option>

<option value="uk">UK</option>

</select>

**8. Textarea (<textarea>):**

* Textareas allow users to input multiple lines of text.

Important attributes:

* + **name** for identifying the input field.
  + **rows** and **cols** for specifying the visible dimensions of the textarea.

Example:

<textarea name="comments" rows="4" cols="50"></textarea>

**9. Number Input (<input type="number">):**

* Number inputs allow users to input numeric values.

Important attributes:

* + **type="number"** to specify the input type.
  + **name** for identifying the input field.
  + **min** and **max** to set value ranges.

Example:

<input type="number" name="quantity" min="1" max="10" step="1">

These are some of the commonly used input types and form elements in HTML, each serving specific purposes for collecting user input or displaying options within a form. Additional attributes and elements can be used to further customize and control their behavior.

**What is CSS?**

CSS stands for Cascading Style Sheets. It is **a style sheet language which is used to describe the look and formatting of a document written in markup language**. It provides an additional feature to HTML.

 It is generally used with HTML to change the style of web pages and user interfaces.  CSS (Cascading Style Sheets) is used to style and layout web pages — for example, to alter the font, color, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features.

CSS describes how elements should be rendered on screen, on paper, in speech, or on other media. It is designed to enable the separation of content and presentation, including layout, colors, and fonts.

**In other Words,**

**C**ascading **S**tyle **S**heets, fondly referred to as **CSS**, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independently of the HTML that makes up each web page. It describes how a webpage should look: it prescribes colours, fonts, spacing, and much more. In short, you can make your website look however you want. CSS lets developers and designers define how it behaves, including how elements are positioned in the browser.

While HTML uses tags, CSS uses rulesets. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.

## CSS Key Features

* CSS specifies how HTML elements should be displayed on screens.
* The Major key feature of CSS includes styling rules which are interpreted by the client browser and applied to various elements.
* Using CSS, you can control the color text, the style of fonts, spacing between elements, how columns are sized, variation in display for different devices and screen size as well as a variety of other effects.

**Why CSS?**

* **CSS saves time:**You can write CSS once and reuse the same sheet in multiple HTML pages.
* **Easy Maintenance:**To make a global change simply change the style, and all elements in all the webpages will be updated automatically.
* **Search Engines:**CSS is considered a clean coding technique, which means search engines won’t have to struggle to “read” its content.
* **Superior styles to HTML:**CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
* **Offline Browsing:**CSS can store web applications locally with the help of an offline cache. Using this we can view offline websites.

**CSS Syntax:**

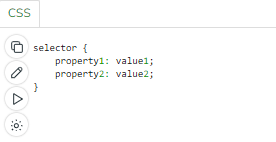
CSS comprises style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule set consists of a selector and declaration block.

1. **Selector:** A selector in CSS is used to target and select specific HTML elements to apply styles to.
2. **Declaration:**A declaration in CSS is a combination of a property and its corresponding value.

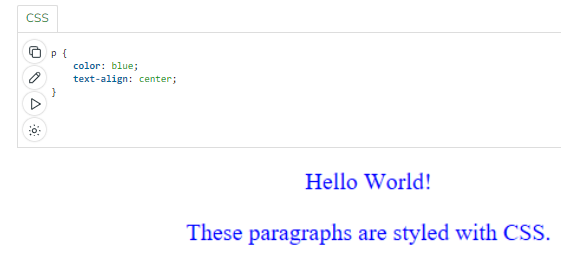
Selector -- h1  
Declaration -- {color:blue;font size:12px;}

* The selector points to the HTML element you want to style.
* The declaration block contains one or more declarations separated by semicolons.
* Each declaration includes a CSS property name and a value, separated by a colon.

**For example :**



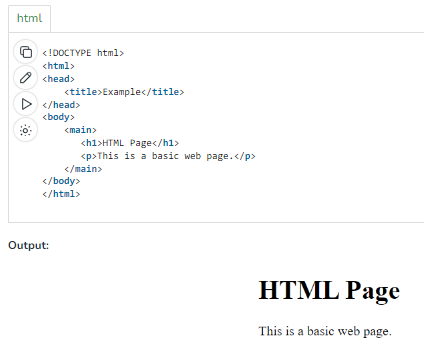
CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.



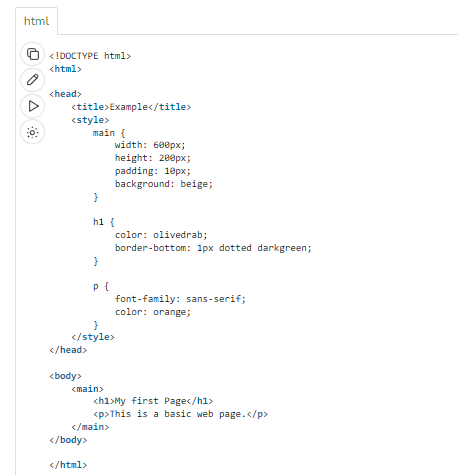
**Here is a more specific example:** In the following example all p elements will be centre-aligned, with a blue text colour:

**Let’s see how our page looks with & without CSS :**

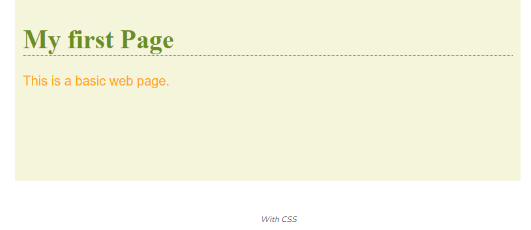
**Before CSS:** In this example, we have not added any CSS.



**After CSS:** In this example, we added some CSS styling inside the HTML code only to show how CSS makes a dull HTML page look beautiful.



**Output:**



**What are the advantages and disadvantages of using CSS?**

**Advantages of CSS:**

* CSS plays an important role, by using CSS you simply got to specify a repeated style for element once & use it multiple times as because CSS will automatically apply the required styles.
* The main advantage of CSS is that style is applied consistently across variety of sites. One instruction can control several areas which is advantageous.
* Web designers needs to use few lines of programming for every page improving site speed.
* Cascading sheet not only simplifies website development, but also simplifies the maintenance as a change of one line of code affects the whole web site and maintenance time.
* It is less complex therefore the effort are significantly reduced.
* It helps to form spontaneous and consistent changes.
* CSS changes are device friendly. With people employing a batch of various range of smart devices to access websites over the web, there’s a requirement for responsive web design.
* It has the power for re-positioning. It helps us to determine the changes within the position of web elements who are there on the page.
* These bandwidth savings are substantial figures of insignificant tags that are indistinct from a mess of pages.
* Easy for the user to customize the online page
* It reduces the file transfer size.

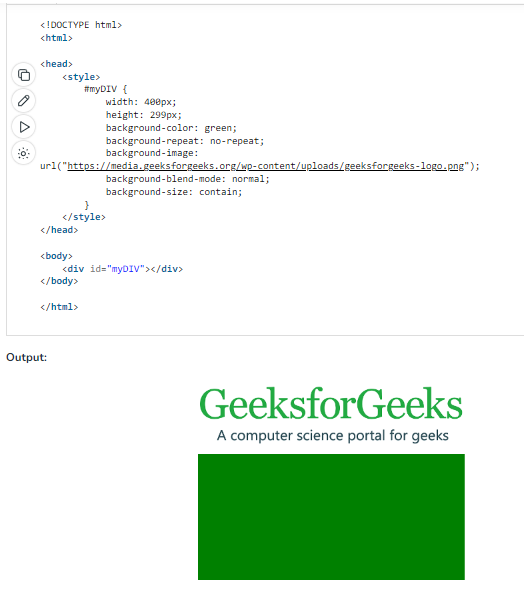
**Disadvantages of CSS:**

* CSS, CSS 1 up to CSS3, result in creating of confusion among  web browsers.
* With CSS, what works with one browser might not always work with another. The web developers need to test for compatibility, running the program across multiple browsers.
* There exists a scarcity of security.
* After making the changes we need to confirm the compatibility if they appear. The similar change affects on all the browsers.
* The programming language world is complicated for non-developers and beginners. Different levels of CSS i.e. CSS, CSS 2, CSS 3 are often quite confusing.
* Browser compatibility (some styles sheet are supported and some are not).
* CSS works differently on different browsers. IE and Opera supports CSS as different logic.
* There might be cross-browser issues while using CSS.
* There are multiple levels which creates confusion for non-developers and beginners.

# CSS Properties

CSS property is used to set the style or assign behavior of HTML elements. The CSS property contains two parts, property\_name, and property\_value. The property\_value is enclosed within double quotes (” “).

**Example:** In this example, we will see the use of many properties usage, all the keywords mentioned inside of { and } braces are properties.



**Note: Students are requested to visit below mentioned webpage to see all properties and their description with example in css.**

<https://www.geeksforgeeks.org/css-properties-complete-reference/?ref=shm>

**Types of CSS (Cascading Style Sheet)**

Cascading Style Sheet ([CSS](https://www.geeksforgeeks.org/css/)) is used to set the style in web pages that contain [HTML](https://www.geeksforgeeks.org/html/) elements. It sets the background color, font-size, font-family, color, … etc. properties of elements on a web page.

There are three types of [CSS](https://www.geeksforgeeks.org/css/) which are given below:

* Inline CSS
* Internal or Embedded CSS
* External CSS

**Inline CSS:**

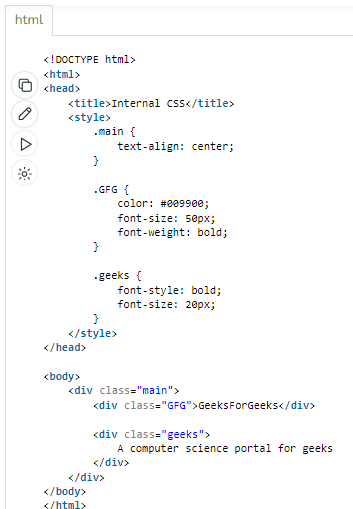
Inline CSS contains the [CSS](https://www.geeksforgeeks.org/css/) property in the body section attached to the element is known as inline CSS. This kind of style is specified within an HTML tag using the style attribute.

**Example:** This example shows the application of inline-css.



**Internal or Embedded CSS:**

This can be used when a single HTML document must be styled uniquely. The [CSS](https://www.geeksforgeeks.org/css/) rule set should be within the [HTML](https://www.geeksforgeeks.org/html/) file in the head section i.e. the [CSS](https://www.geeksforgeeks.org/css/) is embedded within the <style> tag inside the head section of the [HTML](https://www.geeksforgeeks.org/html/) file.

Example:This example shows the application of internal-css.



**External CSS:**

External CSS contains separate [CSS](https://www.geeksforgeeks.org/css/) files that contain only style properties with the help of tag attributes (For example class, id, heading, … etc). [CSS](https://www.geeksforgeeks.org/css/) property is written in a separate file with a .css extension and should be linked to the [HTML](https://www.geeksforgeeks.org/html/) document using a **link** tag. It means that, for each element, style can be set only once and will be applied across web pages.

**Example:** The file given below contains [CSS](https://www.geeksforgeeks.org/css/) property. This file saves with .css extension. For

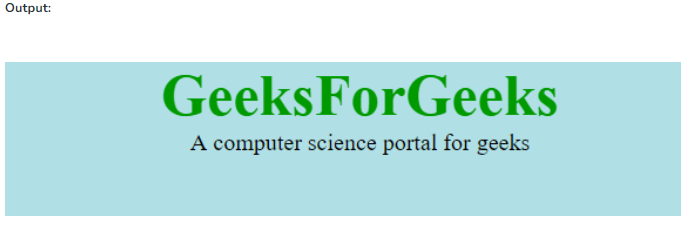
Ex: **geeks.css**



Below is the [HTML](https://www.geeksforgeeks.org/html/) file that is making use of the created external style sheet.

* **link** tag is used to link the external style sheet with the html webpage.
* **href** attribute is used to specify the location of the external style sheet file.





**Priorities of**[**CSS**](https://www.geeksforgeeks.org/css/)**:**

Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority. Multiple style sheets can be defined on one page.

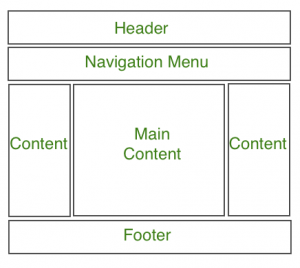
For an [HTML](https://www.geeksforgeeks.org/html/) tag, styles can be defined in multiple style types and follow the below order.

* As Inline has the highest priority, any styles that are defined in the internal and external style sheets are overridden by Inline styles.
* Internal or Embedded stands second in the priority list and overrides the styles in the external style sheet.
* External style sheets have the least priority. If there are no styles defined either in inline or internal style sheet then external style sheet rules are applied for the [HTML](https://www.geeksforgeeks.org/html/) tags.

[CSS](https://www.geeksforgeeks.org/css/) is the foundation of webpages and is used for webpage development by styling websites and web apps.

# CSS Website Layout

A website can be divided into various sections comprising of header, menus, content, and footer based on which there are many different layout designs available for developers. Different layouts can be created by using a div tag and using CSS property to style it.   
The most common structure of website layout is given below:



**Notice:**

 Header section contains a website logo, a search bar and profile of user. The navigation menu contains link to various categories of articles available and content section is divided into 3 parts(columns) with left and right sidebar containing links to other articles and advertisements whereas the main content section is the one containing this article, then at the bottom there is a footer section which contains address, links, contacts etc.

**List of important properties in css**

1. **color:** Sets the text color of an element.
2. **background-color:** Sets the background color of an element.
3. **font-family:** Specifies the typeface or font family for text content.
4. **font-size:** Defines the size of the font.
5. **font-weight:** Determines the thickness or boldness of the font.
6. **text-align:** Aligns text content horizontally within an element (e.g., left, right, center).
7. **text-decoration:** Controls text decoration (e.g., underline, overline, line-through).
8. **line-height:** Sets the vertical spacing between lines of text.
9. **letter-spacing:** Adjusts the space between characters in text.
10. **word-spacing:** Adjusts the space between words in text.
11. **margin:** Defines the spacing outside the border of an element.
12. **padding:** Sets the spacing between the content and the border of an element.
13. **border:** Defines the border properties (width, style, and color) of an element.
14. **border-radius:** Rounds the corners of elements, creating a border with curved edges.
15. **width:** Specifies the width of an element.
16. **height:** Specifies the height of an element.
17. **display:** Controls how an element is displayed (e.g., **block**, **inline**, **inline-block**, **none**, etc.).
18. **position:** Defines the positioning method for an element (e.g., **static**, **relative**, **absolute**, **fixed**).
19. **top, right, bottom, left:** Used in conjunction with the **position** property to precisely position elements.
20. **float:** Positions an element to the left or right, allowing other content to flow around it.
21. **clear:** Specifies how elements should behave in relation to floating elements.
22. **overflow:** Determines how content that overflows the element's box should be handled (e.g., **hidden**, **scroll**, **auto**).
23. **text-transform:** Controls the capitalization of text (e.g., **uppercase**, **lowercase**, **capitalize**).
24. **background-image:** Sets an image as the background of an element.
25. **background-size:** Defines the size of the background image.
26. **background-position:** Specifies the position of the background image.
27. **background-repeat:** Controls how the background image repeats (e.g., **repeat**, **no-repeat**).
28. **box-shadow:** Adds a shadow effect to an element's box.
29. **text-shadow:** Adds a shadow effect to text.
30. **z-index:** Controls the stacking order of elements on the page.
31. **cursor:** Specifies the type of cursor to be displayed when hovering over an element.
32. **opacity:** Adjusts the transparency of an element.
33. **transform:** Applies 2D or 3D transformations to an element (e.g., rotation, scaling).
34. **transition:** Defines transition effects for CSS properties (e.g., smooth color changes).
35. **animation:** Creates animations within CSS, allowing for keyframes and timing.
36. **box-sizing:** Determines how the sizing of an element is calculated (e.g., **content-box**, **border-box**).
37. **list-style:** Combines **list-style-type**, **list-style-image**, and **list-style-position** to set the style, image, and position of list markers (e.g., bullets or numbers) for lists.
38. **text-indent:** Sets the indentation of the first line of text within an element, useful for creating hanging indents or text blocks.
39. **text-overflow:** Controls how text that overflows its container is indicated or truncated (e.g., **ellipsis** for an ellipsis symbol).
40. **white-space:** Defines how white space and line breaks within text content are handled (e.g., **nowrap**, **pre**, **pre-line**).
41. **text-justify:** Specifies the justification method for text within a container (e.g., **auto**, **inter-word**, **inter-ideograph**).
42. **font-style:** Sets the style of the font (e.g., **italic**, **oblique**, **normal**).
43. **font-variant:** Controls the usage of small-caps or other font variants.
44. **font-stretch:** Adjusts the width of a font (e.g., **ultra-condensed**, **expanded**).
45. **font-variant-caps:** Defines how to handle capitalization in fonts (e.g., **small-caps**, **all-small-caps**).
46. **box-align:** Specifies the alignment of an element within its container when using CSS Flexible Box Layout.
47. **box-pack:** Determines how multiple elements should be aligned within a flex container.
48. **column-count:** Sets the number of columns an element should be divided into for multi-column layouts.
49. **column-gap:** Specifies the gap between columns in a multi-column layout.
50. **column-rule:** Combines **column-rule-width**, **column-rule-style**, and **column-rule-color** to set the rule between columns.
51. **border-collapse:** Defines how table borders should be collapsed or separated (e.g., **collapse**, **separate**).
52. **caption-side:** Specifies the placement of a table caption (e.g., **top**, **bottom**).
53. **text-align-last:** Controls the alignment of the last line of text in a block element.
54. **text-emphasis:** Combines **text-emphasis-style** and **text-emphasis-color** to add emphasis marks to text.
55. **outline:** Sets the outline properties (color, style, and width) for an element.
56. **outline-offset:** Defines the space between the outline and the border of an element.
57. **column-width:** Sets the width of columns in a multi-column layout.
58. **column-span:** Determines whether a specific element should span multiple columns.
59. **content:** Inserts generated content before or after an element, often used with pseudo-elements like **::before** and **::after**.
60. **counter-reset, counter-increment:** Used for creating and manipulating counters, which are often used with lists and generated content.
61. **display:** Determines how an element is displayed (e.g., **block**, **inline**, **none**, **inline-block**, **flex**, **grid**).
62. **visibility:** Controls the visibility of an element (e.g., **visible**, **hidden**, **collapse**).
63. **opacity:** Sets the transparency of an element (values between 0 and 1).
64. **position:** Defines the positioning method for an element (e.g., **static**, **relative**, **absolute**, **fixed**, **sticky**).
65. **top, right, bottom, left:** Used with the **position** property to position elements.
66. **float:** Positions an element to the left or right, allowing text and other elements to flow around it.
67. **clear:** Specifies how elements should behave in relation to floating elements.
68. **overflow:** Controls how content that overflows an element's box is handled (e.g., **hidden**, **scroll**, **auto**).
69. **box-sizing:** Determines how the sizing of an element is calculated (e.g., **content-box**, **border-box**).
70. **margin:** Sets the margin space on all four sides of an element.
71. **padding:** Defines the padding space between the content and the border of an element.
72. **border:** Sets the width, style, and color of an element's border.
73. **width, height:** Specifies the width and height of an element.
74. **max-width, max-height:** Defines the maximum allowable width and height of an element.
75. **min-width, min-height:** Sets the minimum allowable width and height of an element.
76. **background-color:** Sets the background color of an element.
77. **background-image:** Specifies an image as the background of an element.
78. **background-size:** Defines the size of the background image.
79. **background-position:** Specifies the position of the background image.
80. **background-repeat:** Controls how the background image repeats (e.g., **repeat**, **no-repeat**).
81. **list-style-type:** Sets the type of marker for list items (e.g., **disc**, **decimal**, **square**).
82. **list-style-image:** Sets an image as the marker for list items.
83. **list-style-position:** Specifies the position of the marker in relation to the list item content.
84. **font-family:** Specifies the typeface or font family for text content.
85. **font-size:** Defines the size of the font.
86. **font-weight:** Determines the thickness or boldness of the font.
87. **text-align:** Aligns text content horizontally within an element (e.g., **left**, **right**, **center**, **justify**).
88. **text-decoration:** Controls text decoration (e.g., **underline**, **overline**, **line-through**, **none**).
89. **color:** Sets the text color of an element.
90. **line-height:** Sets the vertical spacing between lines of text.

These properties cover a wide range of styles and layout aspects and are commonly used in web development for creating responsive and visually appealing web pages.