

2023

Easy Steps in Web Tech

SAJAN KAFLE

HTML Introduction

HTML stands for Hypertext Markup Language, and it is a standard markup language used for creating web pages and applications. HTML is the foundation of all websites and web applications, and it provides the structure and content of a web page.

HTML is a markup language, meaning that it doesn't have any programming logic, but rather it is used to structure content and present it in a meaningful way. HTML is often combined with other technologies such as CSS (Cascading Style Sheets) and JavaScript to create more dynamic and interactive web pages.

To understand "HTML" from front to back, let's look at each word that makes up the abbreviation:

Hypertext: text (often with embeds such as images, too) that is organized in order to connect related items

Markup: a style guide for typesetting anything to be printed in hardcopy or soft copy format

Language: a language that a computer system understands and uses to interpret commands.

HTML History

HTML (Hypertext Markup Language) is a markup language used to create web pages. It was created by Sir Tim Berners-Lee in 1991 while he was working at CERN, the European Organization for Nuclear Research, as a way to share and access scientific documents over the internet.

Since its inception, HTML has gone through several versions, each one adding new features and capabilities. Here is a brief history of the major versions of HTML:

HTML 1.0 (1993): This was the first version of HTML to be standardized, and it included only a few basic tags for defining headings, paragraphs, and lists.

HTML 2.0 (1995): This version added new tags for images, tables, and forms, as well as support for colors and backgrounds.

HTML 3.2 (1997): This version introduced new features such as support for style sheets, frames, and tables with colored borders.

HTML 4.0 (1997): This version added new tags for scripting and multimedia, as well as support for cascading style sheets (CSS) and Unicode.

XHTML (2000): XHTML was a reformulation of HTML 4.0 as an XML application, which meant that HTML documents could be parsed by XML tools.

HTML5 (2014): HTML5 is the latest version of HTML, and it includes new features for audio and video playback, canvas drawing, and offline web applications. It also includes support for responsive web design and semantic markup.

As the web has evolved, HTML has remained a fundamental technology for creating web pages, and it continues to be an essential tool for web developers today.

HTML Tags

HTML tags are used to define the structure and content of a web page. HTML tags are enclosed in angle brackets and consist of a tag name and attributes, if any.

In HTML, there are two main types of tags: container tags and empty tags.

Container tags are used to define an element that has both a start tag and an end tag, and the content is placed between the two tags. Here are some examples of container tags:

- **<html>...</html>:** Defines the root element of an HTML document.
- **<head>...</head>:** Contains metadata about the document, such as the title, links to stylesheets, and scripts.
- **<body>...</body>:** Contains the content of the document, such as headings, paragraphs, images, and other elements.
- **<div>...</div>:** Defines a container element for grouping other elements.
- **<p>...</p>:** Defines a paragraph.
- **..., ..., ...:** Defines an unordered or ordered list and list items.

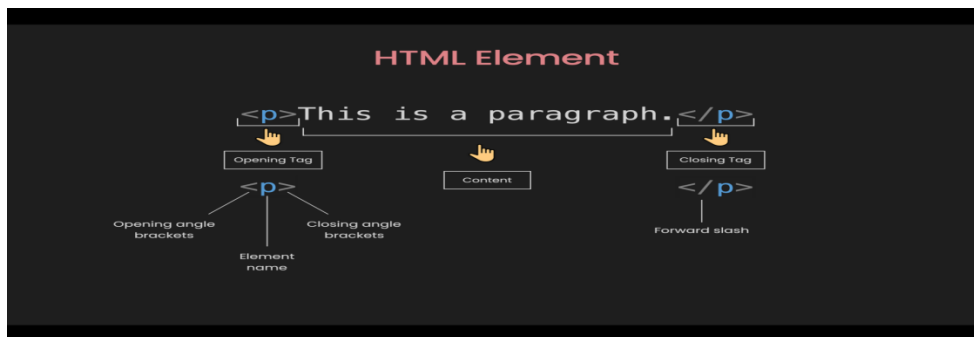
Empty tags, also called self-closing tags, are used to define an element that doesn't have any content or an end tag. Here are some examples of empty tags:

- ``: Displays an image.
- `
`: Inserts a line break.
- `<hr>`: Inserts a horizontal rule.
- `<input>`: Defines an input field.
- `<meta>`: Defines metadata about the document, such as the character encoding and keywords.

It's important to use the correct type of tag for the element you want to define to ensure proper rendering and accessibility of web pages.

HTML Elements

HTML elements are the building blocks of a web page. An HTML element is defined by a start tag, content, and an end tag (or a self-closing tag in some cases). The content can be text, images, videos, other elements, or a combination of these.



HTML Attributes

HTML stands for Hypertext Markup Language, and it is a standard markup language used for creating web pages and applications. HTML is the foundation of all websites and web applications, and it provides the structure and content of a web page.

HTML uses a series of tags, which are enclosed in angle brackets, to define the structure and content of a web page. Tags can be used to create headings, paragraphs, lists, links, images, and other elements of a web page.

HTML is a markup language, meaning that it doesn't have any programming logic, but rather it is used to structure content and present it in a meaningful way. HTML is often combined with other technologies such as CSS (Cascading Style Sheets) and JavaScript to create more dynamic and interactive web pages.

HTML5 is the latest version of HTML, which includes new features and enhancements, such as support for multimedia, new form elements, and semantic tags that help search engines understand the content of a web page.

HTML Basic structure

The basic structure of an HTML document is as follows:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>Heading</h1>
    <p>Paragraph</p>
  </body>
</html>
```

Let's break it down:

- **<!DOCTYPE html>:** This declaration specifies the version of HTML that is being used. In this case, it's HTML5.
- **<html>:** This tag indicates the start of an HTML document.
- **<head>:** This section contains information about the document, such as the title of the page, the author, and links to external resources.
- **<title>:** This tag defines the title of the document, which is displayed in the browser's title bar.
- **<body>:** This section contains the content of the page, such as headings, paragraphs, images, and other elements.
- **<h1>:** This tag defines a heading level 1.

- **<p>**: This tag defines a paragraph.

It's important to note that every HTML document should have a DOCTYPE declaration, an HTML tag, a head tag, and a body tag. The content within the head tag is not visible on the page, while the content within the body tag is what is displayed in the browser.

Creating a Basic Page

To create a basic HTML page, follow these steps:

1. Open a text editor, such as Notepad, Sublime Text, or Visual Studio Code.
2. Start with an empty file and save it with a .html file extension, such as "index.html".
3. Begin your HTML document by adding the <!DOCTYPE html> declaration at the top of the file. This declaration tells the browser that you're using HTML5, the latest version of HTML.
4. Add the opening and closing <html> tags to create the root element of your HTML document.
5. Inside the <html> tags, add the <head> and <body> tags. The <head> section contains metadata about the document, such as the title and links to stylesheets and scripts. The <body> section contains the content of your document, such as text, images, and other elements.
6. Inside the <head> tags, add a <title> tag to specify the title of your document, which will appear in the browser's title bar and search engine results.
7. Inside the <body> tags, add the elements you want to display on your page, such as headings, paragraphs, images, and links.

Here's an example of a basic HTML 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>
  
  <a href="http://www.example.com">Visit Example.com</a>
```

```
</body>  
</html>
```

Save the file and open it in a web browser to see your basic HTML page in action!

HTML Headings

HTML headings are used to create a hierarchy of titles and sub-titles on a web page. They are defined using the <h1> to <h6> tags, where <h1> represents the most important heading and <h6> represents the least important. Here's an example of how to use headings in HTML:

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>My Heading Example</title>  
  </head>  
  <body>  
    <h1>Heading 1</h1>  
    <h2>Heading 2</h2>  
    <h3>Heading 3</h3>  
    <h4>Heading 4</h4>  
    <h5>Heading 5</h5>  
    <h6>Heading 6</h6>  
  </body>  
</html>
```

When you view this code in a web browser, you'll see that the headings appear in a hierarchy, with the <h1> is the largest and most important and the <h6> being the smallest and least important.

It's important to use headings properly on your web pages, as they provide structure and hierarchy to your content, and can help with search engine optimization and accessibility. Best practices suggest using only one <h1> tag per page, and using the other headings in a logical hierarchy.

Semantic HTML

Semantic HTML refers to the use of HTML elements that provide more meaningful information about the content of a web page, rather than just using elements for their visual appearance. By using semantic HTML, web developers can make their pages more accessible to users with disabilities, improve search engine optimization, and create a more organized and readable codebase.

Some examples of semantic HTML elements include:

1. **<header>**: This element is used to define the header section of a web page or a section within it.
2. **<nav>**: This element is used to define a navigation bar or menu on a web page.
3. **<section>**: This element is used to define a section of a web page that groups related content together.
4. **<article>**: This element is used to define an individual piece of content, such as a blog post or news article.
5. **<aside>**: This element is used to define a section of a web page that is related to the main content, such as a sidebar or callout box.
6. **<footer>**: This element is used to define the footer section of a web page or a section within it.

By using these semantic HTML elements, web developers can provide more meaningful information about the content of a web page to assistive technologies like screen readers, improve the organization and readability of their code, and signal to search engines the importance and context of different sections of the page.

Here's an example of how to use some of these semantic HTML elements:

```
<!DOCTYPE html>
<html>
<head>
  <title>Semantic HTML Example</title>
</head>
<body>
  <header>
    <h1>My Website</h1>
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
  </header>
  <section>
```



```
<article>
  <h2>My First Blog Post</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed sit amet dolor id
massa malesuada fermentum. Integer sodales, metus vel hendrerit dictum, tellus massa fringilla magna,
vitae blandit enim quam a velit.</p>
</article>
<aside>
  <h3>Related Posts</h3>
  <ul>
    <li><a href="#">My Second Blog Post</a></li>
    <li><a href="#">My Third Blog Post</a></li>
  </ul>
</aside>
</section>
<footer>
  <p>&copy; 2023 My Website</p>
</footer>
</body>

</html>
```

When you view this code in a web browser, you'll see that the elements used provide more meaningful information about the content of the web page, and can be used to improve accessibility, organization, and search engine optimization.

HTML Text Formatting

HTML provides a variety of formatting elements that can be used to modify the appearance of text and other content on a web page. Some of the most commonly used formatting elements include:

1. ****: This element is used to make text bold.
2. **<i>**: This element is used to make text italic.
3. **<u>**: This element is used to underline text.
4. ****: This element is used to create a strikethrough effect on text.
5. ****: This element is used to emphasize text, typically by italicizing it.
6. ****: This element is used to emphasize text, typically by making it bold.
7. **<sup>**: This element is used to create superscript text.
8. **<sub>**: This element is used to create subscript text.
9. **<code>**: This element is used to display computer code or other text in a monospace font.

10. <pre>: This element is used to display preformatted text, typically in a fixed-width font.
11. <s> - This element is used to strike through text.
12. <mark> - This element is used to highlight text by changing its background color.
13. <small> - This element is used to make text smaller.
14. <big> - This element is used to make text bigger.
15. <blockquote> - Blockquote text

HTML Comments

HTML comments are used to add notes or explanations to HTML code, without affecting the output of the web page. They can be helpful for other developers who are working on the same code or for future reference.

To add a comment in HTML, you can use the following syntax:

```
<!-- This is a comment -->
```

Anything between <!-- and --> will be ignored by the browser and will not be displayed on the web page.

When you view this code in a web browser, you'll see that the comment is not displayed on the web page. However, it can still be viewed in the HTML code when you inspect the page. In the example above, there is also a commented out paragraph of text, which can be used to temporarily disable a section of code without deleting it.

HTML Lists

HTML 5 provides three types of lists: unordered lists, ordered lists, and definition lists.

1. **Unordered Lists:** Unordered lists are created using the tag. Each item in the list is created using the tag. Here's an example of how to create an unordered list:

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

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

This will create an unordered list with three items: "Item 1", "Item 2", and "Item 3". By default, each item in an unordered list is displayed with a bullet point.

2. **Ordered Lists:** Ordered lists are created using the `` tag. Each item in the list is created using the `` tag. Here's an example of how to create an ordered list:

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

This will create an ordered list with three items: "Item 1", "Item 2", and "Item 3". By default, each item in an ordered list is displayed with a number.

3. **Definition Lists:** Definition lists are created using the `<dl>` tag. Each item in the list is created using the `<dt>` tag for the term being defined, and the `<dd>` tag for the definition. Here's an example of how to create a definition list:

```
<dl>
<dt>Term 1</dt>
<dd>Definition 1</dd>
<dt>Term 2</dt>
<dd>Definition 2</dd>
<dt>Term 3</dt>
<dd>Definition 3</dd>
</dl>
```

This will create a definition list with three items: "Term 1" with "Definition 1", "Term 2" with "Definition 2", and "Term 3" with "Definition 3".

Here are some commonly used attributes for unordered lists in HTML:

1. **type:** This attribute is used with the `` tag to specify the type of bullet used in an unordered list. The most common values for this attribute are "disc" (for a solid bullet), "circle" (for an empty circle), and "square" (for an empty square).

```
<ul type="square">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

This will create an unordered list with square bullet points.

2. **start:** This attribute is used with the tag to specify the starting number for an unordered list. However, unlike the start attribute for ordered lists, this attribute is not supported by all browsers, so it is generally not recommended to use it.
3. **compact:** This attribute is used with the tag to create a more compact list by removing the extra spacing between list items. However, like the start attribute, this attribute is also not supported by all browsers and is generally not recommended to use.

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

HTML Tables

HTML tables are used to display tabular data in a grid of rows and columns.

Table Cells: Each table cell is defined by a <td> and a </td> tag.

Table Rows : Each table row starts with a <tr> and ends with a </tr> tag.

Table Borders:

HTML anchor

HTML links are used to connect one webpage to another webpage or resource, allowing users to navigate between different pages or access external resources such as images or documents. Here's an example of how to create an HTML link:

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

In this example, the `<a>` element is used to create the link, and the `href` attribute is used to specify the URL of the webpage or resource that the link points to. The text "Visit Example Website" will be displayed as a clickable link in the browser.

You can also use the `target` attribute to specify where the linked resource should be displayed. For example, you can open the linked page in a new browser window or tab by setting the `target` attribute to `_blank`:

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

There are also other attributes that you can use with links, such as the `title` attribute to provide a tooltip text when the user hovers over the link, or the `rel` attribute to indicate the relationship between the linked page and the current page. For example:

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

```
<a href="https://www.example.com" target="_blank" rel="noopener">Example</a>
```

In the second example, the `rel` attribute is set to `"noopener"` to indicate that the linked page should not be able to access the current page's window object, which can help prevent certain types of security vulnerabilities.

The `rel` attribute in an HTML link is used to specify the relationship between the linked document and the current document. The value of the `rel` attribute is a space-separated list of relationship values. Here are some common values for the `rel` attribute:

alternate: Indicates that the linked document is an alternate version of the current document, such as a different language or format.

stylesheet: Indicates that the linked document is a CSS stylesheet.

nofollow: Indicates that search engines should not follow the link.

noopener: Prevents the new page from being able to access the `window.opener` property of the previous page.

noreferrer: Prevents the browser from sending the `Referer` header to the linked page.

Here's an example of how to use the rel attribute in an HTML link:

```
<a href="https://www.example.com" rel="noopener noreferrer">Click here to visit  
Example website</a>
```

In this example, the rel attribute is set to `noopener noreferrer` to prevent the new page from being able to access the `window.opener` property of the previous page and to prevent the browser from sending the Referer header to the linked page.

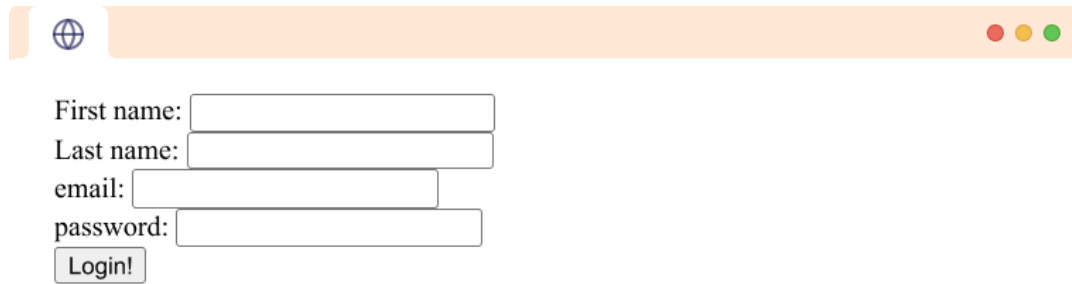
HTML Forms

An HTML Form is a section of the document that collects input from the user. The input from the user is generally sent to a server (Web servers, Mail clients, etc). We use the HTML **<form>** element to create forms in HTML.

The HTML **<form>** element is used to create HTML forms. For example,

```
<form>  
  <label for="firstname">First name: </label>  
  <input type="text" name="firstname" required>  
  <br>  
  <label for="lastname">Last name: </label>  
  <input type="text" name="lastname" required>  
  <br>  
  <label for="email">email: </label>  
  <input type="email" name="email" required>  
  <br>  
  <label for="password">password: </label>  
  <input type="password" name="password" required>  
  <br>  
  <input type="submit" value="Login!">  
</form>
```

Browser Output





HTML Form Elements

A form contains special interactive elements that users use to send the input. They are text inputs, textarea fields, checkboxes, dropdowns, and much more. For example,

```
<form>
  <label for="name">Name:</label>
  <input type="text" name="name"><br><br>
  <label for="sex">Sex:</label>
  <input type="radio" name="sex" id="male" value="male">
  <label for="male">Male</label>
  <input type="radio" name="sex" id="female" value="female">
  <label for="female">Female</label> <br><br>
  <label for="country">Country: </label>
  <select name="country" id="country">
    <option>Select an option</option>
    <option value="nepal">Nepal</option>
    <option value="usa">USA</option>
    <option value="australia">Australia</option>
  </select><br><br>
  <label for="message">Message:</label><br>
  <textarea name="message" id="message" cols="30" rows="4"></textarea><br><br>
  <input type="checkbox" name="newsletter" id="newsletter">
  <label for="newsletter">Subscribe?</label><br><br>
  <input type="submit" value="Submit">
</form>
```

Browser Output



Name:

Sex: ☐ Male ☐ Female

Country:

Message:

☐ Subscribe?

Form Attributes

The HTML `<form>` element contains several attributes for controlling data submission. They are as follows:

action

The action attributes define the action to be performed when the form is submitted. It is usually the url for the server where the form data is to be sent.

```
<form action="/login">
  <label for="email">Email:</label>
  <input type="email" name="email"><br><br>
  <label for="password">Password:</label>
  <input type="password" name="password"><br><br>
  <input type="submit" value="Submit">
</form>
```


In the above example, when the form is submitted, the data from the form is sent to */login*.

method

The method attribute defines the HTTP method to be used when the form is submitted. There are **3** possible values for the *method* attribute:

post - It is used to send data to a server to update a resource.

```
<form method = "post">  
...  
</form>
```

get: It is used to request data from a specified resource.

```
<form method = "get">  
...  
</form>
```

target

It specifies where to display the response received after the form is submitted. Similar to the *target* attribute in `<a>` tags, the *target* attribute has four possible values.

_self (default): Load the response into the same browser tab.

```
<form target="_self">  
  <label for="firstname">Enter your first name:</label>  
  <input type="text" name="firstname">  
</form>
```

_blank: Load the response into a new browser tab.

```
<form target="_blank">  
  <label for="firstname">Enter your first name:</label>  
  <input type="text" name="firstname">  
</form>
```

_parent: Load into the parent frame of the current one. If no parent is available, it loads the response into the same tab.

```
<form target="_parent">
  <label for="firstname">Enter your first name:</label>
  <input type="text" name="firstname">
</form>
```

_top: Load the response into the top-level frame. If no parent is available, it loads the response into the same tab.

```
<form target="_top">
  <label for="firstname">Enter your first name:</label>
  <input type="text" name="firstname">
</form>
```

enctype

It specifies how the form data should be encoded for the request. It is only applicable if we use the POST method.

```
<form method="post" enctype="application/x-www-form-urlencoded"></form>
```

In the above example, data from the form will be encoded in the x-www-form-urlencoded format (which is the default encoding format).

name

It specifies the name of the form. The name is used in Javascript to reference or access this form.

```
<form name="login_form">
  <label for="email">Email:</label>
  <input type="email" name="email"><br><br>
  <label for="password">Password:</label>
  <input type="password" name="password"><br><br>
  <input type="submit" value="Submit">
</form>
```

The above form can be accessed in javascript as:

```
document.forms['login_form']
```

Although it is possible to use name to access form elements in javascript, it is recommended to use id to access the form elements.

novalidate

If the *novalidate* attribute is set, all validations in the form elements are skipped.

```
<form novalidate>
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email"><br>
  <input type="submit">
</form>
```

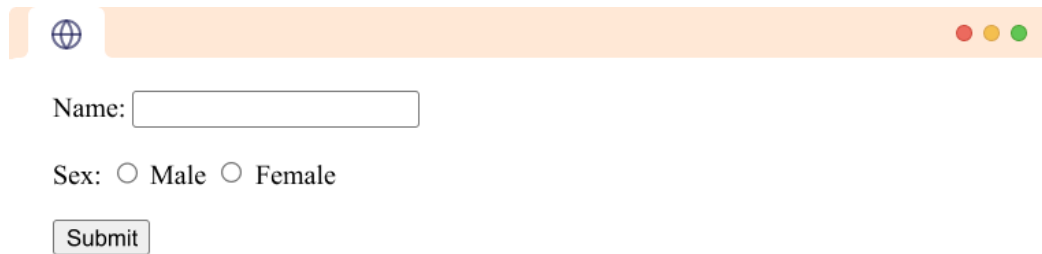
In the above example, the form will be submitted even if we enter some invalid value to the email field, such as *Hi*.

HTML Form Elements

HTML form elements are used to store user input. There are different types of form elements such as the text box, check box, drop-down, submit button, etc. For example,

```
<form>
  <label for="name">Name:</label>
  <input type="text" name="name"><br><br>
  <label for="sex">Sex:</label>
  <input type="radio" name="sex" id="male" value="male">
  <label for="male">Male</label>
  <input type="radio" name="sex" id="female" value="female">
  <label for="female">Female</label> <br><br>
  <input type="submit" value="Submit">
</form>
```

Browser Output

A screenshot of a web browser window with an orange header bar. Inside the browser, there is a form with a "Name:" label followed by a text input field. Below that is a "Sex:" label followed by two radio buttons labeled "Male" and "Female". At the bottom of the form is a "Submit" button.

Name:

Sex: ☐ Male ☐ Female

Here, `<input>` and `<label>` are all HTML Form Elements.

HTML Form Elements

There are various HTML form elements for various input types. The form elements available in HTML5 are as follows:

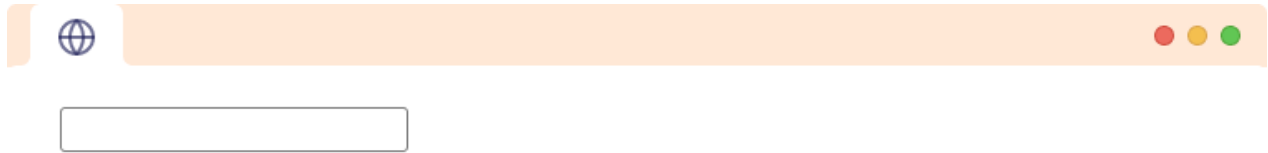
- HTML `<input>` tag
- HTML `<label>` tag
- HTML `<button>` tag
- HTML `<select>`, `<option>` and `<optgroup>` tags
- HTML `<textarea>` tag
- HTML `<fieldset>` tag
- HTML `<legend>` tag
- HTML `<datalist>` tag
- HTML `<output>` tag

HTML `<input>` tag

The HTML `<input>` tag defines the field where the user can enter data. For example,

```
<input type="text" name="firstname">
```

Browser Output



Here,

- *type* - determines the type of input the `<input>` tag takes
- *name* - specifies the name of the input.

HTML `<label>` tag

The HTML label tag is used to create a caption for a form element. The text inside the `<label>` tag is shown to the user.

```
<label for="firstname">First Name</label>  
<input type="text" name="firstname" id="firstname">
```

Browser Output



The `for` attribute is used to associate a label with the form element. The value for the `for` attribute is set as the `id` of the form element which is `firstname` in the above example.

HTML Label is mainly used for accessibility as screen-readers read out the label associated with the field and it also improves user experience as clicking on the label also focuses on the input field.

This is also greatly helpful in small screens as it makes it easier to perform actions like focusing on input, selecting a checkbox, selecting a radio box, etc.

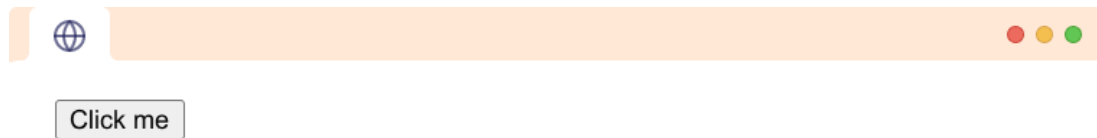
HTML <button> tag

The HTML <button> element is an interactive element that is activated by a user with a mouse, keyboard, finger, voice command, or other assistive technology.

It performs a programmable action, such as submitting a form or opening a dialog when clicked. For example,

```
<button type="button">Click me</button>
```

Browser Output



The type attribute determines the action performed by clicking the button. It has **3** possible values:

Submit

If the value of *type* is submit, the button click action submits the form. For example,

```
<form>
  <label for="name">Name:</label>
  <input type="text" name="name"><br><br>
  <button type="submit">Submit</button>
</form>
```

Browser Output

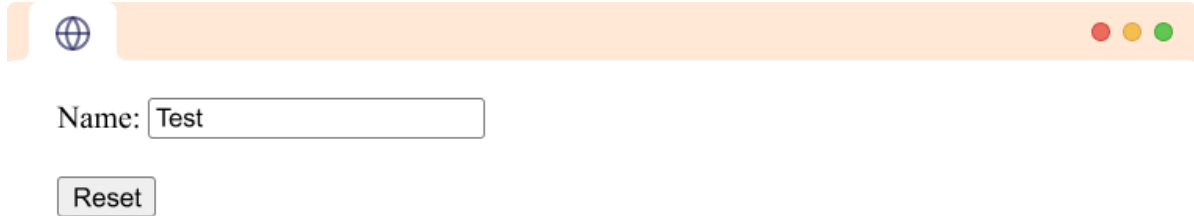


Reset

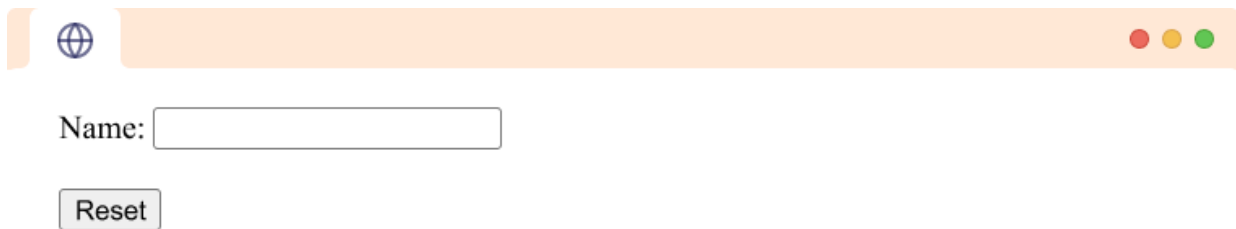
If the value of *type* is reset, the button click action resets the value of all form elements to their initial value. For example,

```
<form>
  <label for="name">Name:</label>
  <input type="text" name="name"><br><br>
  <button type="reset">Reset</button>
</form>
```

Browser Output (before reset)

A screenshot of a web browser window. The address bar is empty. The page content shows a form with a label "Name:" followed by a text input field containing the text "Test". Below the input field is a button labeled "Reset".

Browser Output (after reset)

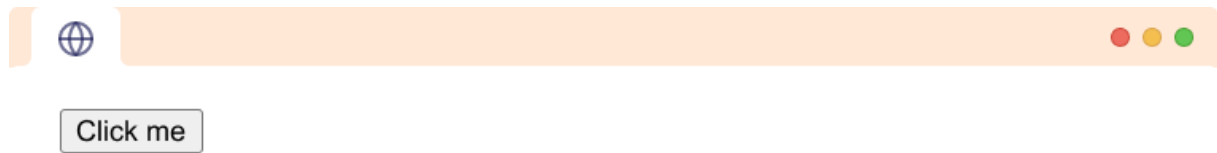
A screenshot of a web browser window, identical to the one above, but the text input field is now empty, demonstrating the effect of the reset button.

button

If the value of *type* is button, the button click action does not have a default function. Generally, javascript is used to add functionality to such buttons. For example,

```
<button type="button">Click me</button>
```

Browser Output



HTML <select>, <option> and <optgroup> tags

The HTML <select> tag is used to create a menu of options. Each of the options is represented by the <option> tag. For example,

```
<label for="pets">Pets:</label>
<select id="pets">
  <option value="dog">Dog</option>
  <option value="cat">Cat</option>
</select>
```

Browser Output



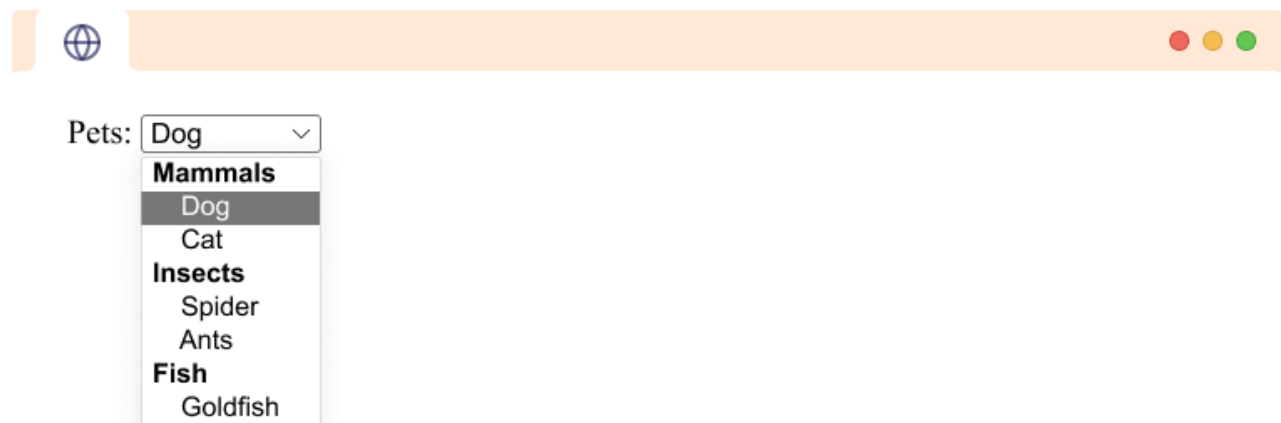
Browser Output



Additionally, we can also group option elements inside the <optgroup> tag to create a group of options. For example,

```
<label for="pets">Pets:</label>
<select id="pets">
  <optgroup label="Mammals">
    <option value="dog">Dog</option>
    <option value="cat">Cat</option>
  </optgroup>
  <optgroup label="Insects">
    <option value="spider">Spider</option>
    <option value="ants">Ants</option>
  </optgroup>
  <optgroup label="Fish">
    <option value="goldfish">Goldfish</option>
  </optgroup>
</select>
```

Browser Output



HTML <textarea> tag

The HTML <textarea> tag is used to define a customizable multiline text input field. For example,

```
<textarea rows="10" cols="30"> Type something...</textarea>
```

Browser Output

A screenshot of a web browser window. The window has a light orange header bar with a globe icon on the left and three colored window control buttons (red, yellow, green) on the right. Below the header is a large, empty text input field with a thin gray border. The placeholder text "Type something..." is visible in the top-left corner of the input field.

Here, the *rows* and *cols* attributes represent the rows and columns of the text field.

HTML <fieldset> tag

The HTML <fieldset> tag is used to group similar form elements. It is a presentational tag and does not affect the data in the form. For example,

```
<form >
  <fieldset>
    <label for="firstname">First name:</label><br>
    <input type="text" id="firstname" name="fname"><br>
    <label for="lastname">Last name:</label><br>
    <input type="text" id="lastname" name="lname"><br>
  </fieldset>
</form>
```

Browser Output




Here, the border is from the `<fieldset>` element.

HTML `<legend>` tag

The HTML `<legend>` tag is another presentational tag used to give a caption to a `<fieldset>` element. It acts similarly to an HTML `<label>` tag. For example,

```
<form>
  <fieldset>
    <legend>Name</legend>
    <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname"><br>
    <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname"><br>
  </fieldset>
</form>
```

Browser Output



HTML <datalist> tag

The <datalist> tag defines a list of pre-defined options for an <input> element. It is used to provide autocomplete options to the form elements that show up as recommended options when the user fills in the form. For example,

```
<label for="country-choice">Choose a country:</label>
<input list="country-options" id="country-choice" name="country-choice">
<datalist id="country-options">
  <option value="Australia">
  <option value="Austria">
  <option value="America">
  <option value="Nepal">
</datalist>
```

Browser Output



Here, when the user types *au*, the browser suggests options with the letters to the user as a recommendation.

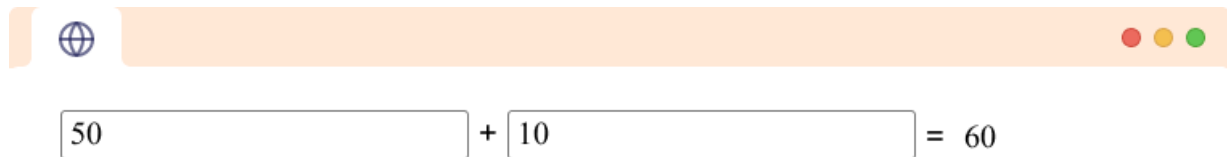
HTML <output> tag

The HTML <output> tag is a container element that is used to store the output of a calculation usually performed using javascript. For example,

```
<form>
  <input type="number" id="b" name="b" value="50" /> +
  <input type="number" id="a" name="a" value="10" /> =
```

```
<output name="result" for="a b"></output>
</form>
<script>
  const output = document.getElementsByName("result")[0]
  const inputA = document.getElementById('a')
  const inputB = document.getElementById('b')
  output.innerText = Number(inputA.value) + Number(inputB.value)
</script>
```

Browser Output



50 + 10 = 60

The `for` attribute of the `<output>` tag accepts a space-separated value of all the inputs used in the calculation. You can notice we have used *a b* inside the `for`. This is to denote that the output inside the `<output>` tag was generated using inputs *a* and *b*.

The value inside the `<output>` is generally generated from Javascript and filled inside the element. Here, we have calculated the sum of both inputs and inserted it inside the `<output>` element.

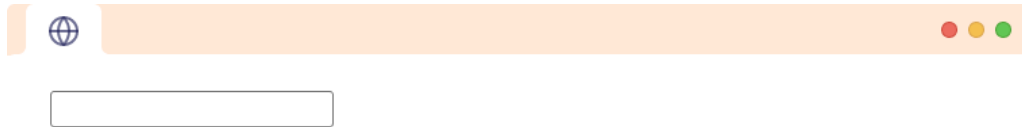
HTML Input Tag

In this tutorial, we will learn about the input tag and its types in HTML with the help of examples.

The HTML `<input>` tag defines the field where the user can enter data. The `type` attribute determines what type of user input is taken.

```
<input type="text" name="firstname">
```

Browser Output



Here,

- *type* - determines the type of input the `<input>` tag takes
- *name* - specifies the name of the input which is what the data is named as when submitting the data to a server.

Different Input Types

The various types of input tags available in HTML5 are:

1. **text** - creates a single-line text fields (default)
 2. **button** - creates a button with no default functionality
 3. **checkbox** - creates a checkbox
 4. **color** - creates a color picker
 5. **date** - creates a date picker
 6. **datetime-local** - creates a date and time picker
 7. **email** - creates an input field that allows the user to input a valid email address
 8. **file** - creates an input field that lets the user upload a file or multiple files
 9. **hidden** - creates an invisible input field
 10. **image** - creates a button using an image
 11. **month** - creates an input field that lets the user enter month and year
 12. **password** - creates an input field that lets the user enter information securely
 13. **radio** - creates a radio button
 14. **range** - creates a range picker from which the user can select the value
 15. **reset** - creates the button which clears all the form values to their default value
 16. **search** - allows user to enter their search queries in the text fields
 17. **submit** - allows user to submit form to the server
 18. **tel** - defines the field to enter a telephone number
 19. **time** - creates an input field that accepts time value
 20. **url** - lets the user enter and edit a URL
 21. **week** - lets the user pick a week and a year from a calendar
-

1. Input Type text

The input type text is used to create single-line text fields. It is the default input type.

```
<label for="name">Search: </label>  
<input type="text" id="name">
```

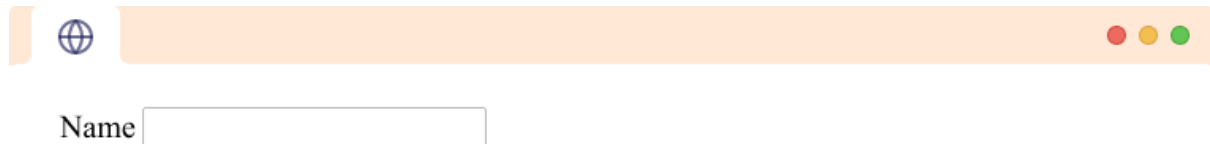
Browser Output



The input type text can also contain *minlength*, *maxlength*, and *size* attributes. For example,

```
<label for="name">Name</label>  
<input type="text" id="name" minlength="4" maxlength="8">
```

Browser Output



In the above example, values are only allowed between the length of **4** to **8** characters.

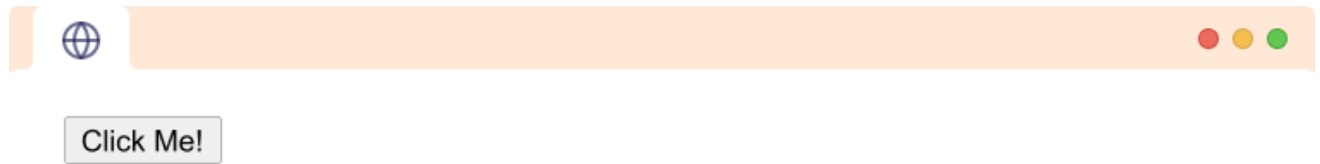
Note: If the *type* attribute is not provided, the tag becomes the type of text.

2. Input Type button

The input type button is used to create a button with no default functionality. For example,

```
<input type="button" value="Click Me!">
```

Browser Output



The text inside the value attribute is displayed in the button.

Note: Generally, javascript is used to add functionality to such buttons.

3. Input Type checkbox

The input type checkbox is used to create a checkbox input. For example,

```
<input type="checkbox" id="subscribe" value="subscribe">  
<label for="subscribe">Subscribe to newsletter!</label><br>
```

Browser Output (checkbox unselected)



The checkbox can be toggled between selected and not selected.

Browser Output (checkbox selected)



☒ Subscribe to newsletter!

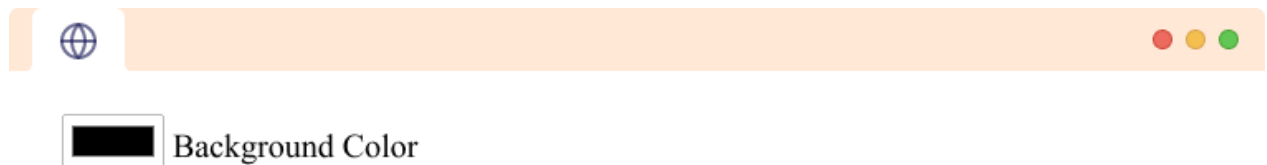
The value of the checkbox is included in the form data only if the checkbox is selected and is omitted if the checkbox is not selected.

4. Input Type color

The input type color is used to create an input field that lets the user pick a color. For example,

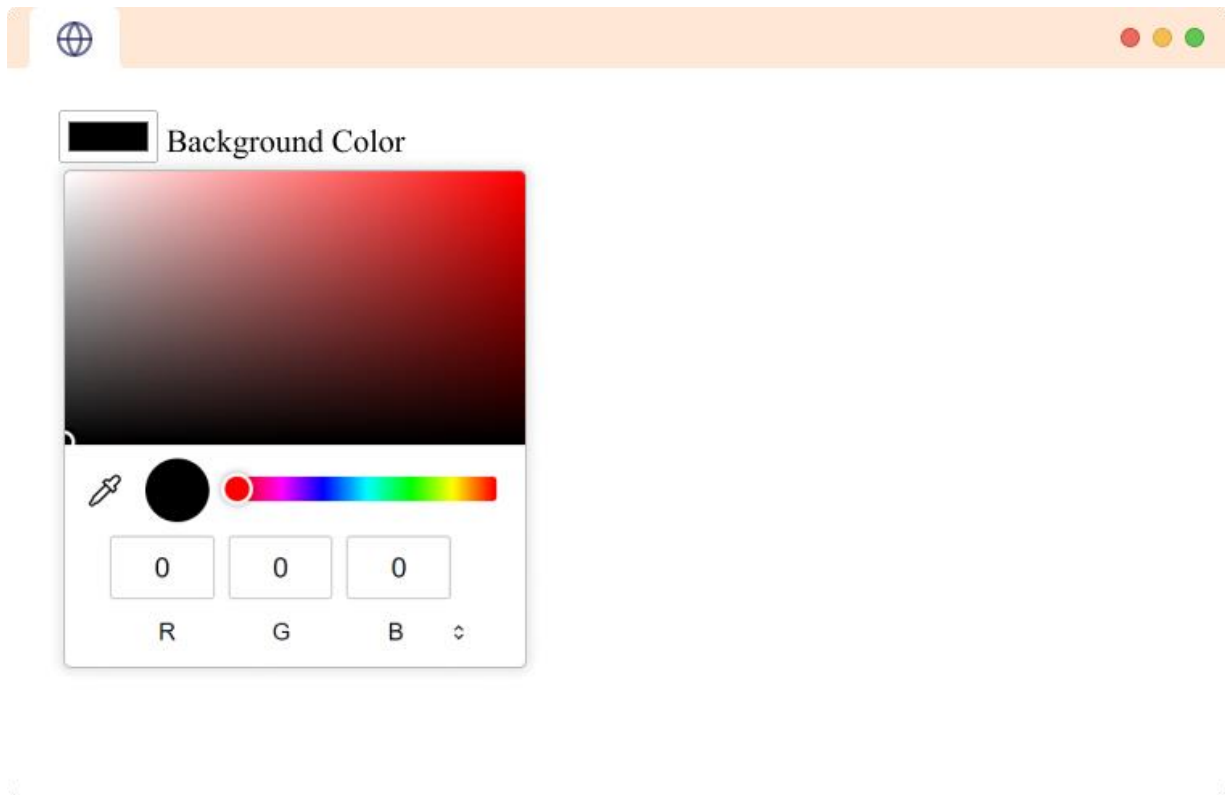
```
<input type="color" id="background">  
<label for="background">Background Color</label>
```

Browser Output (before expanding)



Background Color

Browser Output (after expanding)



The color picker is inbuilt into the browser. Users can also manually enter the hex code of the color instead. The UI for the color picker differs from browser to browser.

5. Input Type date

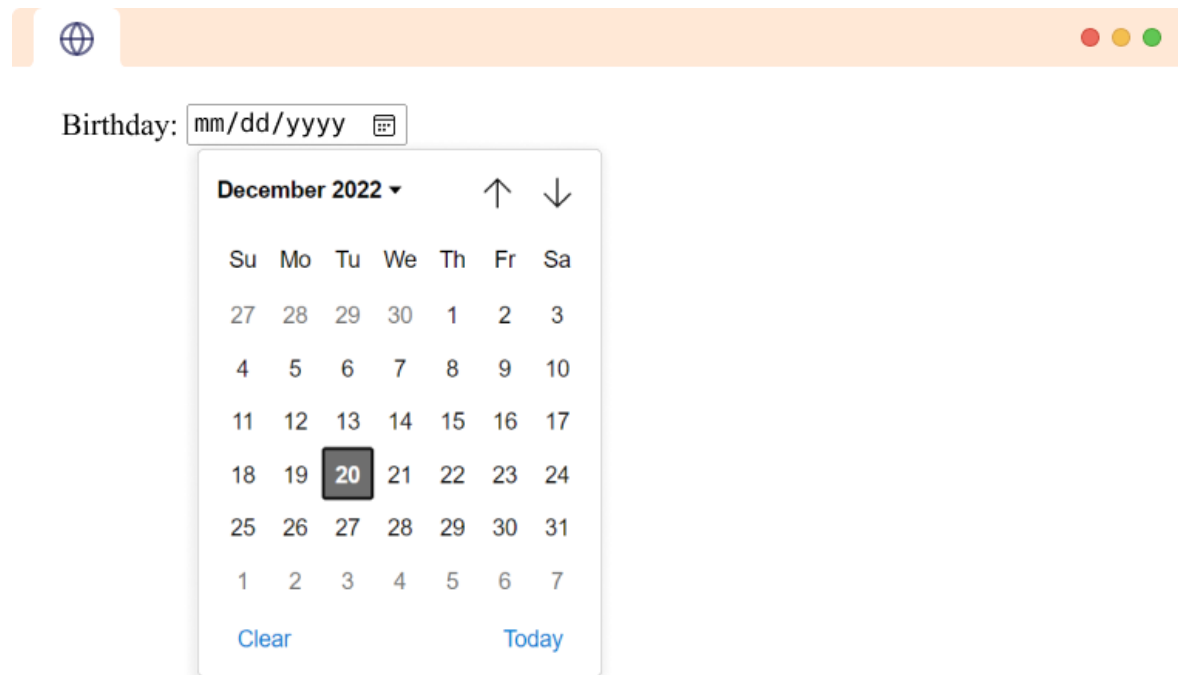
The input type date is used to create an input field that lets the user input a date using the date picker interface from the browser. For example,

```
<label for="birthday">Birthday:</label>  
<input type="date" id="birthday">
```

Browser Output (before expanding)



Browser Output (after expanding)



Birthday:

December 2022

| Su | Mo | Tu | We | Th | Fr | Sa |
|----|----|----|----|----|----|----|
| 27 | 28 | 29 | 30 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Clear Today

6. Input Type datetime-local

The input type `datetime-local` is used to create an input field that lets the user pick a date and time using a date-time-picker from the browser. The time selected from the input does not include information about the timezone. For example,

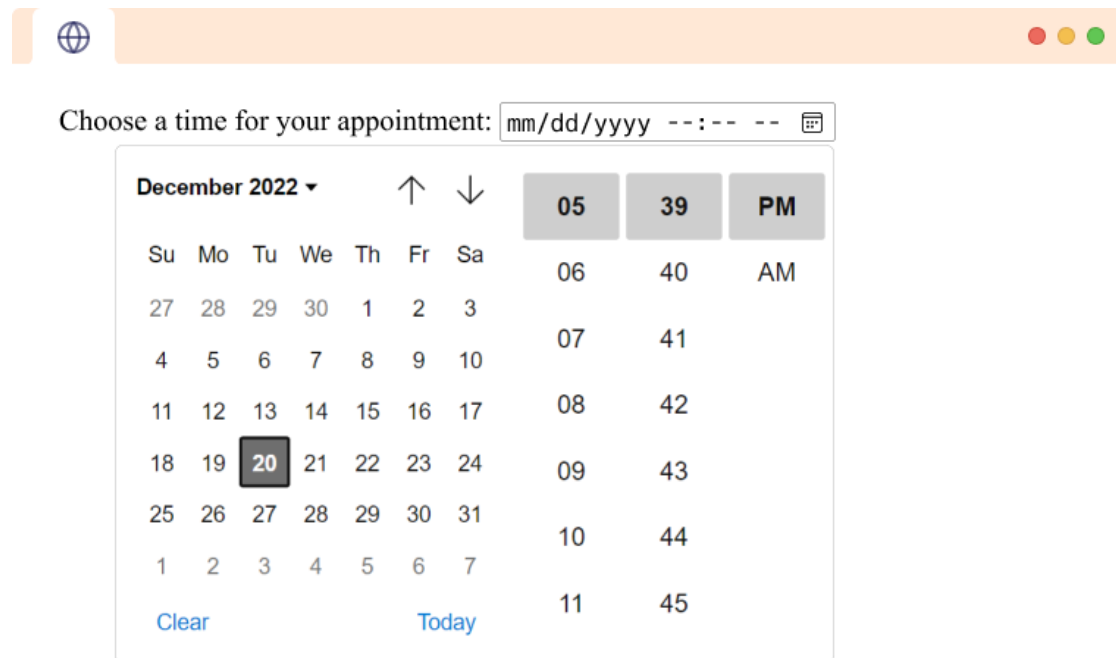
```
<label for="meeting-time">Choose a time for your appointment:</label>  
<input type="datetime-local" id="meeting-time" >
```

Browser Output (before expanding)



Choose a time for your appointment:

Browser Output (after expanding)



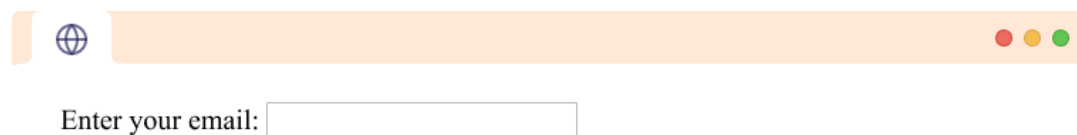
The screenshot shows a web browser window with a light orange header bar. Below the header, there is a text input field with the placeholder text "Choose a time for your appointment:" followed by a date and time picker. The date picker is expanded, showing a calendar for December 2022. The calendar has a grid of days from Sunday to Saturday. The 20th is highlighted. To the right of the calendar, there are three buttons: "05", "39", and "PM". Below these buttons, there are two columns of numbers: "06", "07", "08", "09", "10", "11" and "40", "41", "42", "43", "44", "45". At the bottom of the calendar, there are two buttons: "Clear" and "Today".

7. Input Type email

The input type email is used to create an input field that allows the user to input a valid email address.

```
<label for="email">Enter your email:</label>
<input type="email" id="email">
```

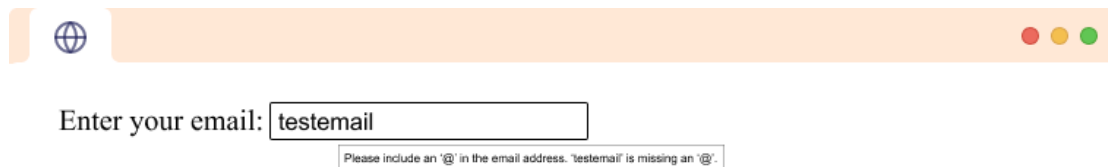
Browser Output



The screenshot shows a web browser window with a light orange header bar. Below the header, there is a text input field with the placeholder text "Enter your email:". The input field is empty.

This input field throws an error if the value provided is not a valid email. For example,

Browser Output



8. Input Type file

The input type file is used to create an input field that lets the user upload a file or multiple files from their device. For example,

```
<input type="file" name="file">
```

Browser Output



9. Input Type hidden

The input type hidden is used to create an invisible input field. This is used to supply additional value to the server from the form that cannot be seen or modified by the user. For example,

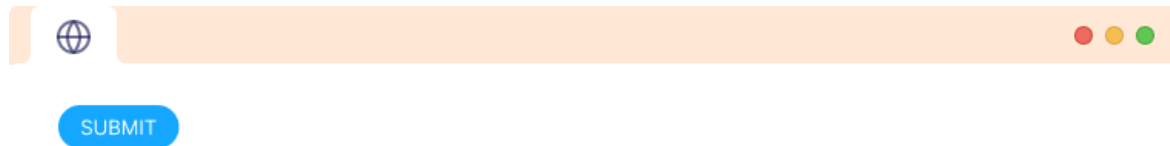
```
<input type="hidden" name="id" value="123" >
```

10. Input Type image

The input type image is used to create a button using an image.

```
<input type="image" src="/submit.png" alt="submit" >
```

Browser Output



Let's see an example of how we can use it in a form.

```
<form>
  <label for="firstname">First name: </label>
  <input type="text" id="firstname" name="firstname"><br><br>
  <label for="lastname">Last name: </label>
  <input type="text" id="lastname" name="lastname"><br><br>
  <input type="image" src="/submit.png" alt="submit" >
</form>
```

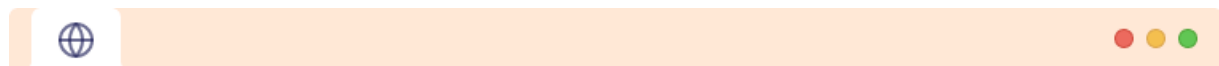
Browser Output

11. Input Type month

The input type month is used to create an input field that lets the user enter month and year using a visual interface from the browser. For example,

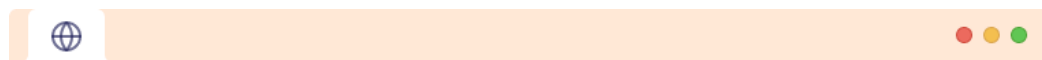
```
<label for="start">Select Month:</label>
<input type="month" id="start" >
```

Browser Output (before expanding)



Select Month: 

Browser Output (after expanding)



Select Month: 

2022

| | | | |
|-----|-----|-----|------------|
| Jan | Feb | Mar | Apr |
| May | Jun | Jul | Aug |
| Sep | Oct | Nov | Dec |

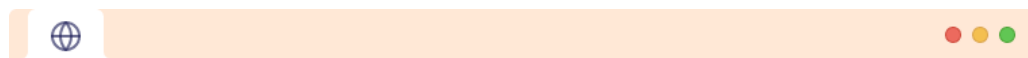
[Clear](#)[This month](#)

12. Input Type password

The input type password is used to create an input field that lets the user enter information securely. For example,

```
<label for="password">Password:</label>  
<input type="password" id="password">
```

Browser Output



Password:

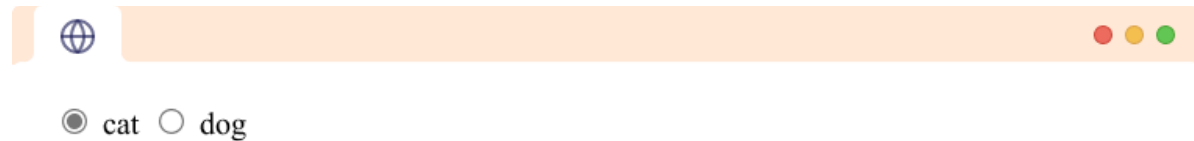
The browser displays all the characters the user types using an asterisk (*).

13. Input Type radio

The input type radio is used to define a radio button. Radio buttons are defined in a group. Radio buttons let users pick one option out of a group of options.

```
<form>
  <input type="radio" id="cat" name="animal" value="cat">
  <label for="cat">cat</label>
  <input type="radio" id="dog" name="animal" value="dog">
  <label for="dog">dog</label>
</form>
```

Browser Output



From the above example, we can see that all the radio buttons share the same name attribute. It allows the user to select exactly one option from the group of radio buttons.

When submitting the form data, the key for the input will be the name attribute, and the value will be the radio button selected.

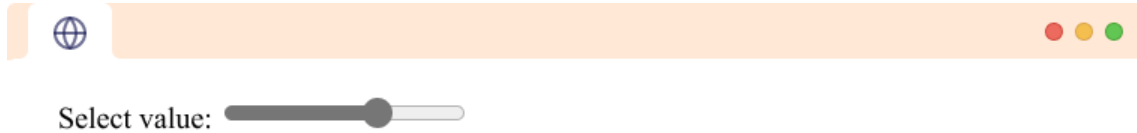
Note: The name attribute is used as the key for the data when submitting the form.

14. Input Type range

The input type range is used to create a range picker from which the user can select the value. User can select a value from the range given. It has a default range from **0** to **100**. For example,


```
<label for="range">Select value: </label>
<input type="range" id="range" value="90">
```

Browser Output



15. Input Type reset

The input type reset defines the button which clears all the form values to their default value. For example,

```
<form>
  <label for="name">Name:</label>
  <input id="name" type="text" /><br />
  <input type="reset" value="Reset" />
</form>
```

Browser Output



Browser Output (after reset)



A browser window mockup with an orange header bar containing a globe icon on the left and three colored window control buttons (red, yellow, green) on the right. Below the header, the text "Name:" is followed by an empty text input field. Below the input field is a button labeled "Reset".

16. Input Type search

The input type search allows user to enter their search queries in the text fields. It is similar to input type text. For example,

```
<label for="search">Search: </label><input type="search" id="search" >
```

Browser Output



A browser window mockup with an orange header bar containing a globe icon on the left and three colored window control buttons (red, yellow, green) on the right. Below the header, the text "Search:" is followed by a search input field containing the text "Test".

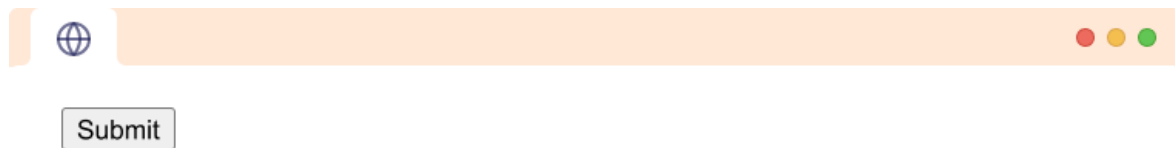
Note: The search input does not work as a search unless we use some JavaScript to do the search calculation.

17. Input Type submit

The input type submit is used when the user submits the form to the server. It is rendered as a button. For example,

```
<input type="submit" value="submit">
```

Browser Output



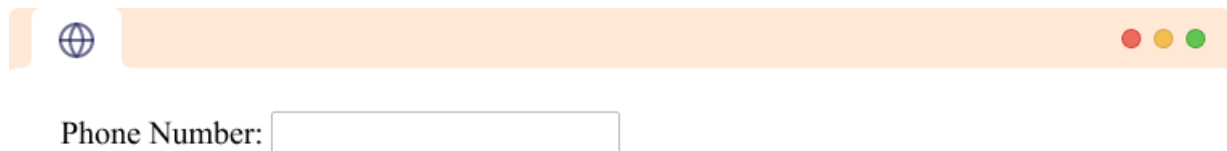
Here, The text provided in the *value* attribute of the input is shown in the button.

18. Input Type tel

The input type tel is used to define the field to enter a telephone number. The telephone number is not automatically validated to a particular format as telephone formats differ across the world. It has an attribute named *pattern* used to validate the entered value. For example,

```
<label for="phone">Phone Number:</label>
<input type="tel" id="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
```

Browser Output



The pattern in the above example allows numbers in the format XXX-XX-XXX where X is a number between **0** and **9**.

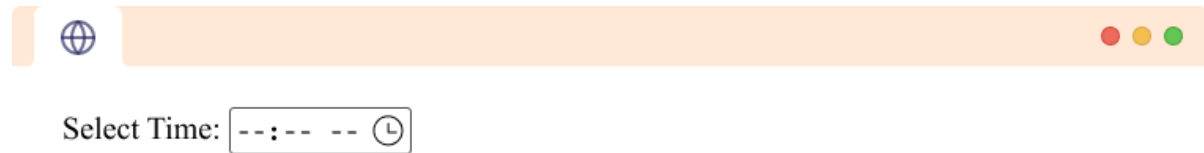
19. Input Type time

The input type time attribute creates an input field that accepts time value. It allows users to add time in hours, minutes, and seconds easily. For example,

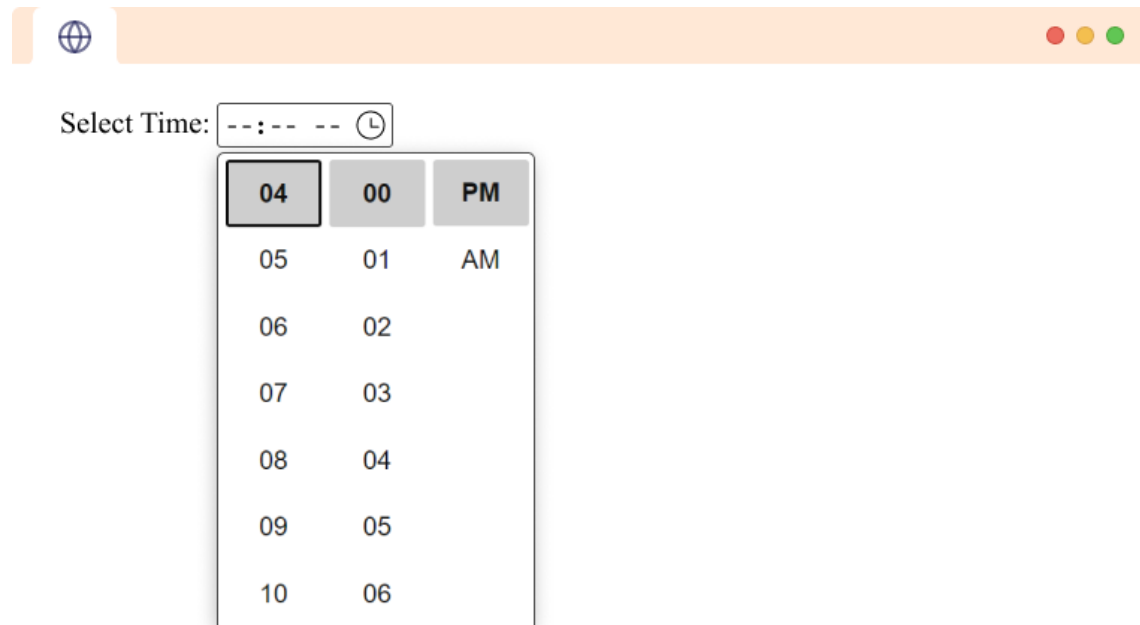
```
<label for="time">Select Time:</label>
```

```
<input type="time" id="time">
```

Browser Output (before expanding)



Browser Output (after expanding)



20. Input Type url

The input type url is used to let the user enter and edit a URL. For example,

```
<label for="url">URL:</label>
```

```
<input type="url" id="url" placeholder="https://example.com" pattern="https://.*">
```

Browser Output



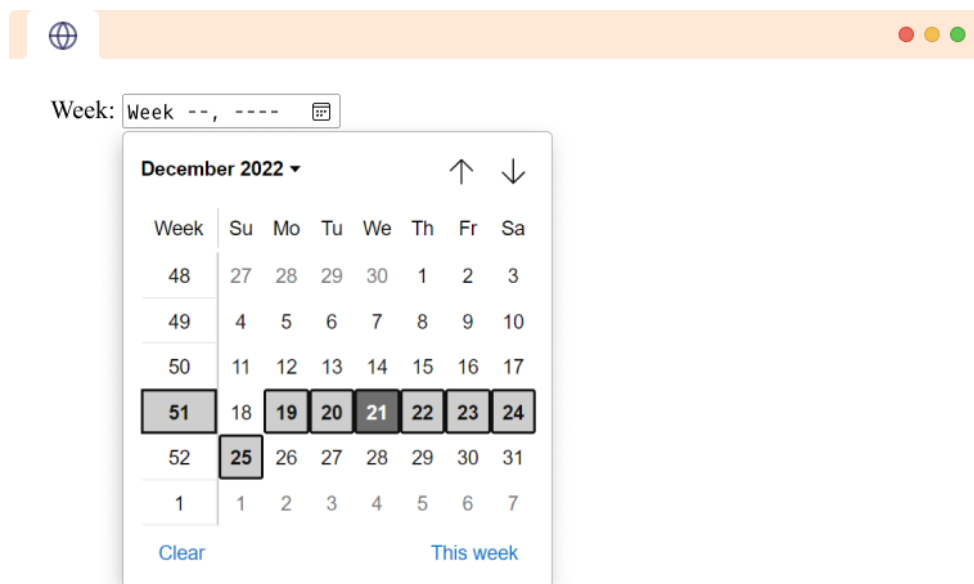
Here, placeholder is a hint that specifies the expected value of an input field, and the pattern defines what type of value is accepted. The above pattern means that only text beginning with *https://* will be valid.

21. Input Type week

The input type week lets the user pick a week and a year from a calendar. For example,

```
<label for="week">Week</label>  
<input type="week" id="week" >
```

Browser Output



<https://www.programiz.com/html/form-elements>

HTML Form Action: POST and GET

In this tutorial, we will learn about POST and GET methods in HTML with the help of examples.

The method attribute in the <form> element specifies how the data is sent to the server.

HTTP methods declare what action is to be performed on the data that is submitted to the server. HTTP Protocol provides several methods, and the HTML Form element is able to use two methods to send user data:

- **GET method** - used to request data from a specified resource
- **POST method** - used to send data to a server to update a resource

The GET Method

The HTML GET method is used to get a resource from the server. For example,

```
<form method="get" action="www.programiz.com/search">
  <input type="search" name="location" placeholder="Search.." />
  <input type="submit" value="Go" />
</form>
```

When we submit the above form by entering *California* in the input field, the request sent to the server will be `www.programiz.com/search/?location=California`.

The HTTP GET method adds a query string at the end of the URL to send data to the server. The query string is in the form of key-value pair followed by ? symbol.

From the URL, the server can parse the user-submitted value where:

- key - *location*
- value - *California*

Note: If there is more than one query, the query string will be separated by a & symbol.

The POST method

The HTTP POST method is used to send data to the server for further processing. For example,

```
<form method="post" action="www.programiz.com/user">
  <label for="firstname">First name:</label>
  <input type="text" name="firstname" /><br />
  <label for="lastname">Last name:</label>
  <input type="text" name="lastname" /><br />
  <input type="submit" />
</form>
```

When we submit the form, it will add the user input data to the body of the request sent to the server. The request would look like

```
POST /user HTTP/2.0
Host: www.programiz.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 33
```

firstname=Robin&lastname=Williams

The data sent is not easily visible to the user. However, we can check the sent data using special tools like the browsers' dev tools.

GET vs POST

| GET | POST |
|--|--|
| Data sent with the GET method is visible in the URL. | Data sent with the POST method is not visible. |
| GET requests can be bookmarked. | POST requests can't be bookmarked. |
| GET requests can be cached. | POST requests can't be cached. |
| GET requests have a character limit of 2048 characters. | POST requests do not have a limit. |

Only ASCII characters are allowed in GET requests.

All data is allowed in POST request

HTML Graphics & Media

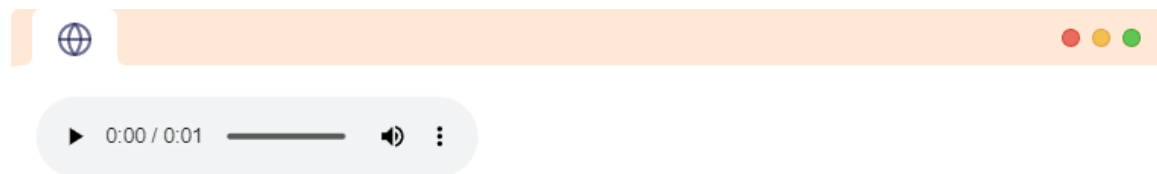
HTML Audio

In this tutorial, we will learn about the audio tag in HTML with the help of examples.

The HTML `<audio>` tag is used to embed a media player which supports audio playback into the HTML page. We use the HTML `<audio>` tag along with the `<source>` tag to add the audio player. For example,

```
<audio controls>
  <source src="audio.mp3" type="audio/mp3">
</audio>
```

Browser output



In the above code:

- *audio.mp3* - path to the audio we want to display
- *audio/mp3* - the type of resource we want to display.

The *audio.mp3* file in the above example is located in the same directory as the HTML file.

HTML <audio> with Multiple <source> Tag

We can have more than one video <source> inside the <audio> tag. For example,

```
<audio controls>
  <source src="audio.ogg" type="audio/ogg">
  <source src="audio.mp3" type="audio/mpeg">
</audio>
```

The <source> tag is used to provide several URLs of alternate formats for resources for multimedia tags like <audio> tag. The browser chooses the first <source> tag whose resource is supported. In the above case, if the browser does not support the ogg format, it will move to the next <source> tag and play the mp3 file.

Attributes of HTML <audio> tag

Let us look at the attributes supported by the HTML <audio> tag.

- autoplay
- controls
- loop
- muted
- src
- preload

We will learn about each of these in detail.

autoplay

The autoplay attribute automatically plays the audio. For example,

```
<audio controls autoplay>
  <source src="audio.ogg" type="audio/ogg">
  <source src="audio.mp3" type="audio/mpeg">
</audio>
```

Browser Output



This will cause the audio file to begin playing as soon as the page loads.

Note: The autoplay attribute is generally considered a bad user experience, as it can be annoying for users.

controls

The control attribute allows the user to control audio playback including volume, seeking, and pause/resume playback. For example,

```
<audio controls>
  <source src="audio.ogg" type="audio/ogg">
  <source src="audio.mp3" type="audio/mpeg">
</audio>
```

Browser Output



This will add the default audio controls to the element, allowing the user to play, pause, adjust the volume, and seek through the audio file.

You can customize the audio controls using JavaScript and the HTMLMediaElement API. This allows you to build your own audio player with custom design and functionality.

loop

The loop attribute specifies the audio to start from the beginning once it ends. For example,

```
<audio controls loop>
  <source src="audio.ogg" type="audio/ogg">
  <source src="audio.mp3" type="audio/mpeg">
</audio>
```

Browser Output



This will cause the audio file to start over from the beginning when it reaches the end.

muted

The muted attribute sets the volume of the audio to **0**. For example,

```
<audio controls muted>
  <source src="audio.ogg" type="audio/ogg">
  <source src="audio.mp3" type="audio/mpeg">
</audio>
```

Browser output



In the above example, the audio file will start with the volume set to zero

src

The src attribute specifies the location of the audio file that should be played in the audio player. For example,

```
<audio src="/audios/sample.mp3" controls>
</audio>
```

Here, the audio element will create an audio player that will play the audio file located at /audios/sample.mp3

Note: At least one src attribute or <source> tag is required for HTML video.

preload

The preload attribute specifies how the audio file should be loaded after the page loads for a better user experience. It may have one of the following values:

1. **none:** Indicates that the audio should not be preloaded. For example,

```
<audio src="song.mp3" preload="auto"></audio>
```

2. **metadata:** Indicates that only audio metadata is fetched. For example,

```
<audio src="song.mp3" preload="metadata"></audio>
```

3. **audio**: Indicates that the entire audio file will be loaded when the page loads. For example,

```
<audio src="song.mp3" preload="auto"></audio>
```

HTML SVG

In this tutorial, we will learn about the `svg` tag in HTML with the help of examples.

SVG (Scalable Vector Graphics) is used to create 2D diagrams such as shapes, logos, charts, etc.

The HTML `<svg>` tag is used to embed SVG graphics in a web page. For example,

```
<svg width="100" height="100" style="border:1px solid black;">
  <circle cx="50" cy="50" r="30" fill="blue" />
</svg>
```

Browser Output



In the above example, the code creates an SVG element with a width of **100** pixels and a height of **100** pixels. The SVG element has a solid black border.

Inside the `<svg>` element we have created a circle using `<circle>` element. The `cx` and `cy` attributes define the **x** and **y** coordinates of the center of the circle, while the `r` attribute defines the **radius**. The `fill` attribute determines the color of the circle, which is set to blue in this example.

Together, these elements create a blue circle with a black border that is centered within the <svg> container.

Attributes of SVG

There are various attributes of SVG. They are as follows:

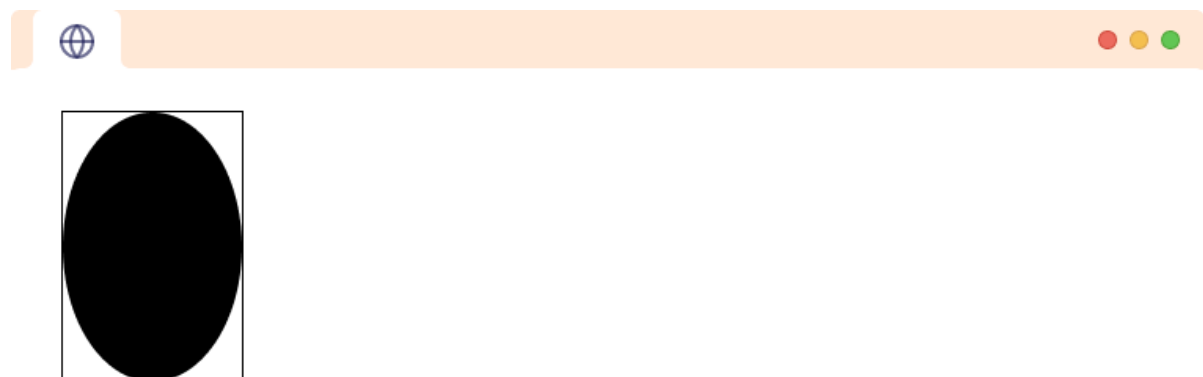
- preserveAspectRatio
- viewport and viewBox

preserveAspectRatio

The preserveAspectRatio attribute is used in SVG elements to specify how an element should be scaled and aligned within the viewport. This attribute determines how the aspect ratio (ratio of width to height) of the element is preserved when the element is resized or stretched. For example,

```
<svg width="100" height="150" viewBox="0 0 100 100" preserveAspectRatio="none" style="border: 1px solid black">  
  <circle cx="50" cy="50" r="50" />  
</svg>
```

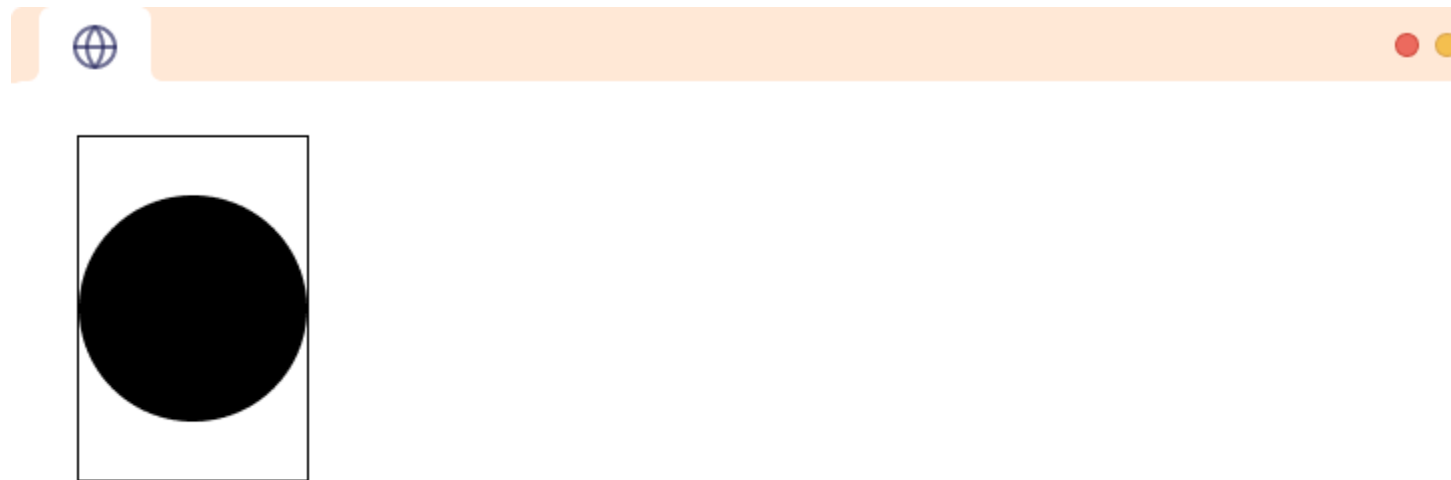
Browser Output



In this example, the `preserveAspectRatio` attribute is set to `none`, so the circle will be stretched to fit the dimensions of the viewport. As a result, the circle appears distorted. Let's look at another example.

```
<svg width="100" height="150" viewBox="0 0 100 100" preserveAspectRatio="meet" style="border: 1px solid black">
  <circle cx="50" cy="50" r="50" />
</svg>
```

Browser Output



In this example, the `preserveAspectRatio` attribute is set to `meet`, so the circle will be scaled down (if necessary) to fit within the viewport. As a result, there are empty spaces around the circle.

viewport and viewBox

SVG viewport

The viewport is the visible area of the SVG. We use `width` and `height` attributes to define the viewport of an SVG. For example,

```
<svg width="100" height="100" style="border: 1px solid black">
```

```
<circle cx="50" cy="50" r="50" fill="blue" />
</svg>
```

Browser Output



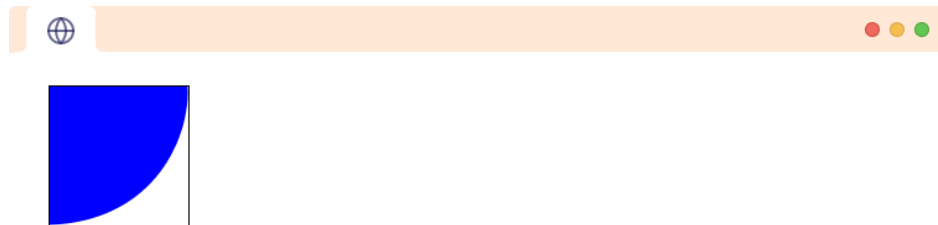
In the above example, we have created an SVG with height and width **100** px.

SVG viewBox

The viewBox attribute defines the position and dimension, in user space, of an SVG viewport. Think of it like a telescope that is used to view the element inside the SVG. We can move the viewbox left and right and also zoom in and zoom out. Let's see an example,

```
<svg width="100" height="100" viewBox = "50 50 50 50" style="border: 1px solid black">
  <circle cx="50" cy="50" r="50" fill="blue" />
</svg>
```

Browser Output



Here, the first two parameters of viewBox are min-x and min-y respectively. They define the top left corner of the viewBox.

The value of min-x is moving our viewBox to the right by **50** pixels. Similarly, the value of min-y is moving our viewBox to the bottom by **50** pixels. That is why we can only see bottom-right of the circle.

The third and fourth parameters represent the width and height of the viewBox respectively. They can also be used for zooming in and zooming out.

Zoom out using SVG viewBox

If the size of viewBox is larger than the viewport the circle will zoom out. Let's see an example,

```
<svg width="100" height="100" viewBox = "0 0 1000 1000" style="border: 1px solid black">  
  <circle cx="50" cy="50" r="50" fill="blue" />  
</svg>
```

Browser Output



In the above example, the value of height and width of the viewBox is larger than the viewport size so the circle has zoomed out.

Zoom in using SVG viewBox

If the size of viewBox is smaller than the viewport the circle will zoom in. For example,

```
<svg width="100" height="100" viewBox = "0 0 25 25" style="border: 1px solid black">
```

```
<circle cx="50" cy="50" r="50" fill="blue" />
</svg>
```

Browser Output



In the above example, the value of height and width of the viewbox is smaller than the viewport size so the circle has zoomed in.

Why SVG?

The advantages of using SVGs are:

- SVGs maintain their quality and remain lightweight even when resized.
- They have a consistent file size and can be easily created and edited with simple code.
- The SVG code is human-readable and doesn't require any specialized software.

HTML Canvas

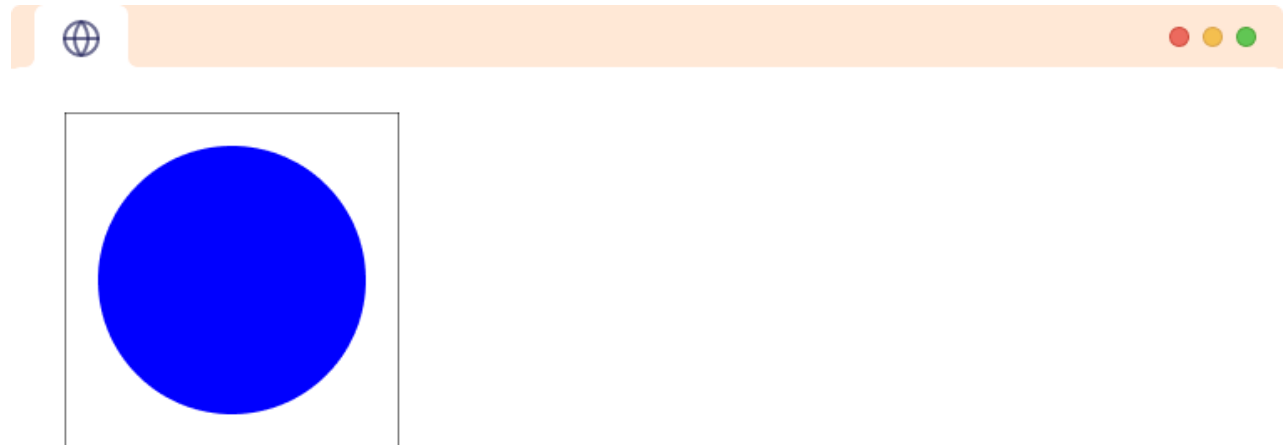
In this tutorial, we will learn about div tags in HTML with the help of examples.

HTML `<canvas>` is used to create graphics in HTML. We create the graphics inside the `<canvas>` using JavaScript. For example,

```
<canvas id="circle-canvas" height="200" width="200" style="border: 1px solid;"></canvas>
<script>
  let canvas = document.getElementById("circle-canvas");
  let ctx = canvas.getContext("2d");
```

```
ctx.beginPath();
ctx.arc(100, 100, 80, 0, 2 * Math.PI, false);
ctx.fillStyle = 'blue';
ctx.fill();
</script>
```

Browser Output



Here, we are using JavaScript along with the Canvas API to create a circle inside the canvas. This code creates a simple drawing using the canvas element in an HTML page.

The canvas has an id *circle-canvas* and dimensions of **200x200** pixels. This creates a surface on which we can draw using JavaScript.

We have used the `getContext()` method of the canvas element and stored it in the *ctx* variable. It allows us to access the drawing context and draw on the canvas.

In the above example,

- `beginPath()` - creates a new path, any subsequent path commands will start from this point
- `arc()` - draws a circle where,
 - the circle is centered at *(100, 100)*
 - the circle has a radius of *80* pixels
 - the circle is drawn to cover the entire circumference in a counter-clockwise direction, by starting at **0** radians and ending at *2*Math.PI* radians
- `fillStyle` - sets the fill color of the circle to blue

- fill()- fills the circle with the blue color

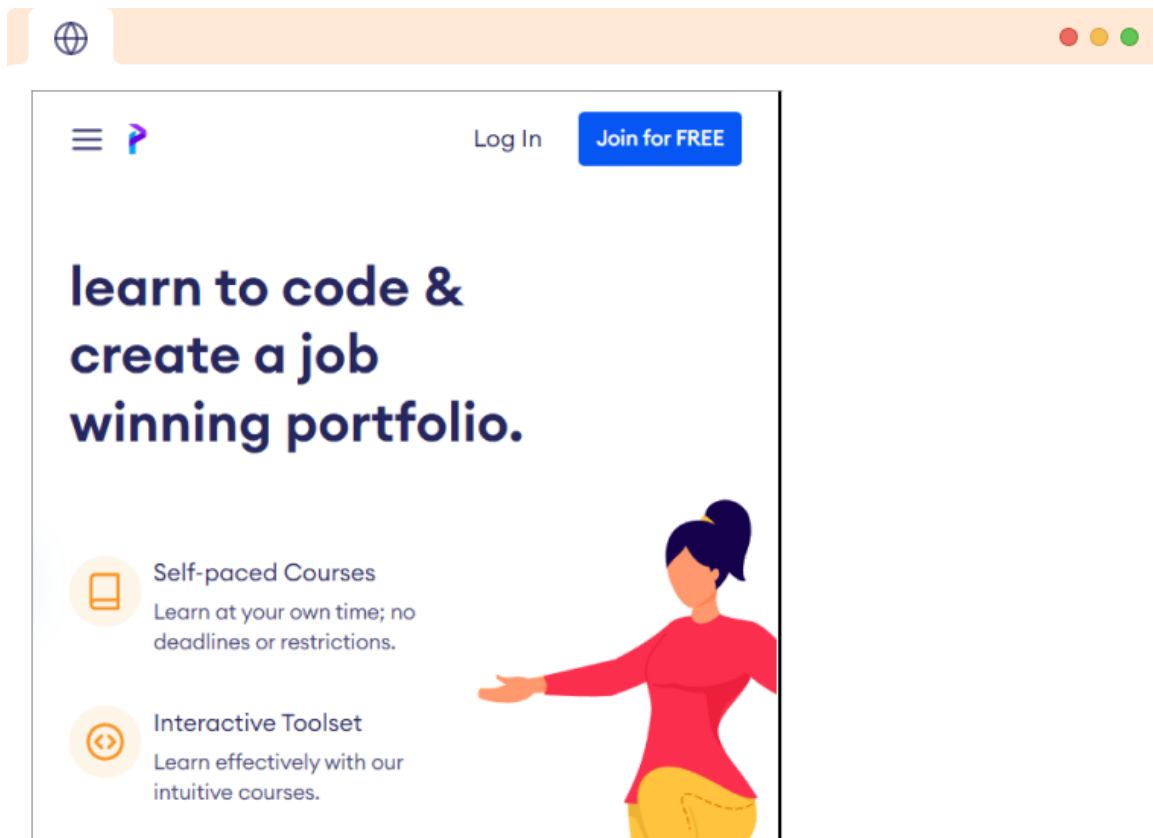
HTML Iframes

In this tutorial, we will learn about div iframes in HTML with the help of examples.

The HTML <iframe> tag is used to embed a webpage within a webpage. It is also called an inline frame. For example,

```
<iframe src="https://programiz.pro" title="programiz pro website" height="500" width="500" ></iframe>
```

Browser Output



Here,

- **src:** It is used to specify the URL of the website to be loaded.
- **title:** It is good practice to include a *title* attribute so that screen readers can read the title to users.

Other Attributes for <iframe>

There are some important attributes for <iframe>. They are:

- height and width
- name
- srcdoc

We will explore each of them in detail.

height and width

We can set the height and width of the <iframe> element with the height or width attribute. For example,

```
<iframe src="https://programiz.pro" height="200" width="300"></iframe>
```

We can also use CSS to set the width and height of the <iframe> using the style attribute. For example,

```
<iframe src="https://programiz.pro" style="height:200px;width:300px"></iframe>
```

It is important to add height and width to allocate space on the webpage for the iframe. It prevents content from moving when the iframe loads.

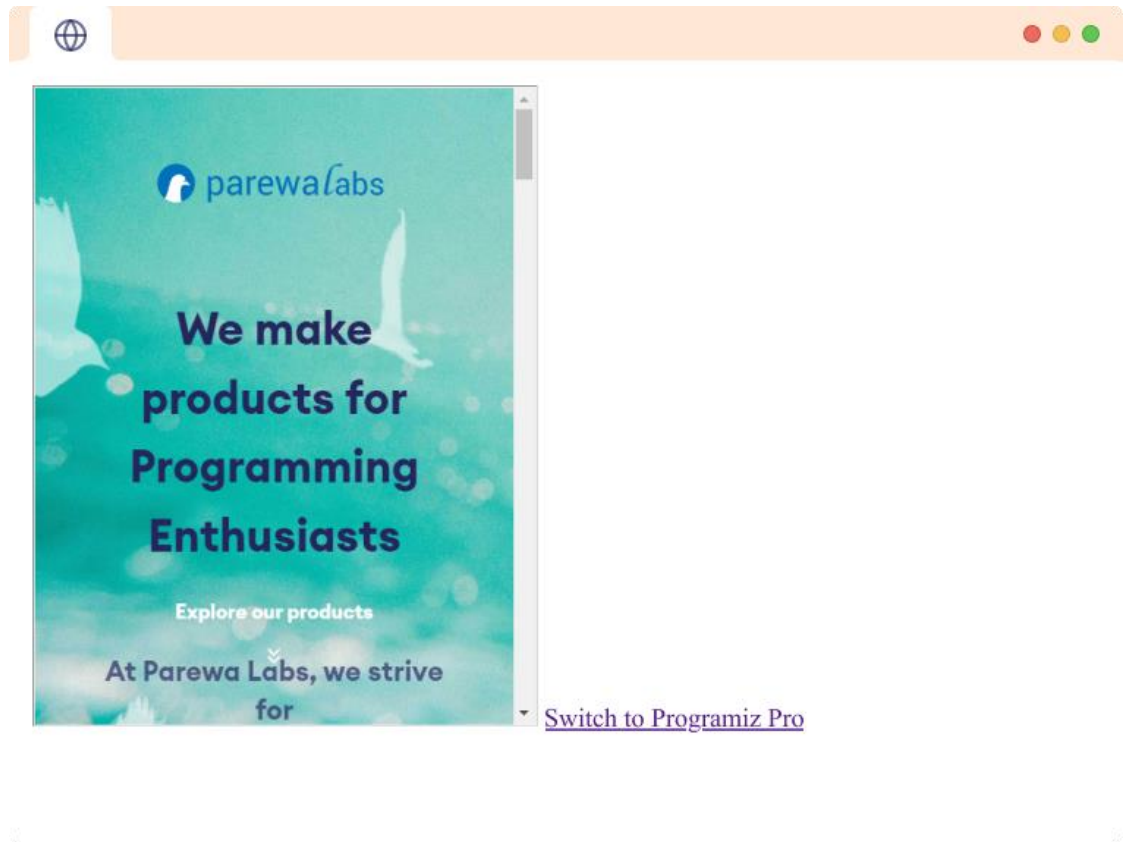
name

The name attribute is used to specify the name for an iframe. It can be used as a target for other HTML elements like the <a> tag. For example,

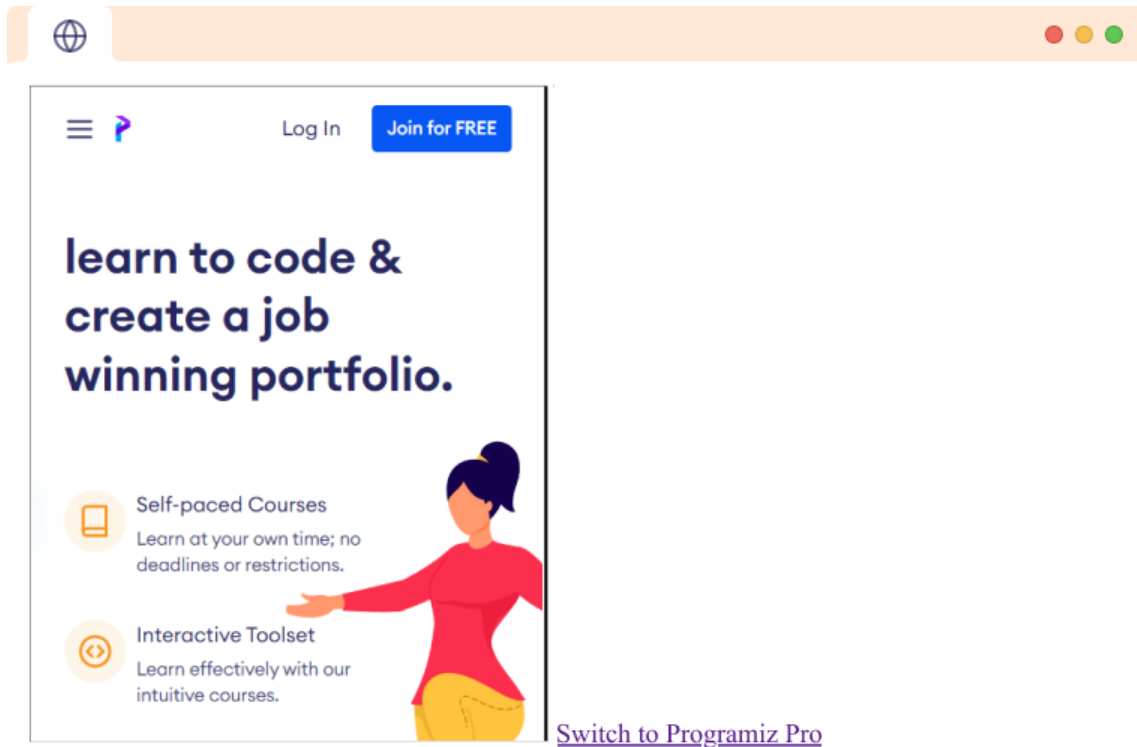
```
<iframe src="https://parewalabs.com" name="iframe_target" height="500" width="400"></iframe>
```

```
<a href="https://www.programiz.pro" target="iframe_target">Switch to Programiz Pro</a>
```

Browser Output (Before Clicking the link)



Browser Output (After clicking the link)



Here, clicking the `<a>` tag changes the URL of the target instead of the current window.

srcdoc

Instead of a website URL, we can send HTML directly to the `iframe`, which will be displayed instead of another website. For example,

```
<iframe srcdoc="<h1>Learn to code</h1>"></iframe>
```

Browser Output



HTML Entities

In this tutorial, we will learn about entities in HTML with the help of examples.

The HTML entities are used to display reserved characters (characters that are used in HTML code), special characters, or invisible characters. For example,

`<p>This is a <p> tag.</p>`

Browser Output



Here, `<` and `>` are the HTML entities used to display `<` and `>` respectively.

If we were to use the `<p>` tag instead of `<p>`, the browser would read it as a tag. Hence to display `<p>` we need to use `<` and `>` instead of `<` and `>` respectively.

HTML Entities Syntax

HTML entities are represented by either their name or their number.

- **Entity names**

We use `&entity_name;` to add reserved characters using Entity names. For example, `¢` would be displayed as ¢

- **Entity number**

We use `&#entity_number;` to add reserved characters using Entity number. For example, `¢` would also be displayed as ¢

All entity names and numbers start with an `&` and end with `;`

Note: Not all entities have names, only characters that are commonly used have entity names.

Special Characters

Special characters are characters that are not available on a general keyboard like ®, ©, ¢, etc.

We use HTML entities to add special characters to HTML documents. For example

```
<footer> &copy; 2022 Programiz. All rights reserved. </footer>
```

Browser Output



Invisible Characters

An invisible character is a character that is not visible when rendered in a document or text field. These characters can be used for various purposes, such as adding white space or formatting a document. For example,

<p>Invi‌sible Cha‌racte
rs</p>

Browser Output

Here, ‌ and are invisible characters. ‌ is a zero-width character whereas is a space character.

Some examples of invisible characters are – space, tab, zero-width space, etc.

Some important HTML entities

| Character(s) | Literal(s) | Alphanumeric value(s) | Unicode value(s) |
|----------------------|------------|-----------------------|------------------|
| Cent (currency) | ¢ | ¢ | ¢ |
| Pound (currency) | £ | £ | £ |
| Section | § | § | § |
| Copyright | © | © | © |
| Guillemets | « » | « » | « » |
| Registered trademark | ® | ® | ® |
| Degree(s) | ° | ° | ° |
| Plus/minus | ± | ± | ± |
| Pilcrow (paragraph) | ¶ | ¶ | ¶ |
| Middle dot | · | · | · |
| Fractional half | ½ | ½ | ¼ |
| En dash | – | – | – |
| Em (long) dash | — | — | — |

| | | | |
|-------------------------------|-----|-------------------|-----------------|
| Single quotes | ' ' | ‘ ’ | ‘ ’ |
| Single low quote | , | ‚ | ‚ |
| Double quotes | " " | “ ” | “ ” |
| Double low quote | „ | „ | „ |
| Single & double daggers | † ‡ | † ‡ | † ‡ |
| Bullet | • | • | • |
| Ellipsis | ... | … | … |
| Prime & double prime | ' " | ′ ″ | ′ ″ |
| Euro sign | € | € | € |
| Trademark | ™ | ™ | ™ |
| Almost equal to | ≈ | ≈ | ≈ |
| Not equal to | ≠ | ≠ | ≠ |
| Less/greater than or equal to | ≤ ≥ | ≤ ≥ | ≤ ≥ |
| Less/greater than | < > | < > | > < |

HTML Quotation

In this tutorial, we will learn about quotations in HTML with the help of examples.

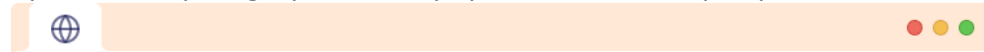
The HTML quotation elements are used to insert the quoted text in the web page that is different from the standard text on the website. In this tutorial, we will learn about the following HTML Quotation elements.

- <q>
- <blockquote>

<q> tag

The <q> tag is used for short quotations that do not require paragraph breaks. For example,

```
<p>This is a paragraph with <q>quotation text</q></p>
```



This is a paragraph with "quotation text"

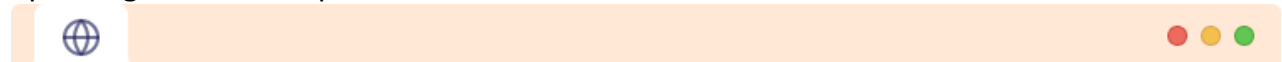
Here, the quotation marks are coming from the <q> tag.

<blockquote> tag

The <blockquote> tag defines the section of text that is quoted from another source. Another source is any place other than the current website. It is used to represent block of quoted text. For example,

```
<blockquote cite="https://www.programiz.com/about-us"> Our mission is to help users learn programming more easily.</blockquote>
```

```
<p>-Programiz.com</p>
```



Our mission is to help users learn programming more easily.

-Programiz.com

In the above example, the <blockquote> tag displays the quoted text as a block with an indentation at the beginning that indicates that it is a quotation.

The cite attribute is an optional attribute used to add a URL that designates a source document for the information quoted. In this example, we can see that the above quote is quoted from the Programiz site.

HTML File Paths

In this tutorial, we will learn about HTML file paths with the help of examples.

An HTML file path is the address of a particular file. It is used to link external files like javascript, CSS, images, and other webpages in the HTML document. For example,

```
<a href="path/to/file.html">Link to file</a>
```

In this example, *path/to/file.html* is the file path to the HTML file. This file path is relative to the current HTML document.

Types of File Path

There are two types of file paths:

- Absolute File Path
- Relative File Path

We will learn about both file paths in detail.

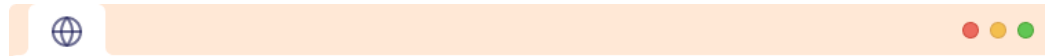
Absolute File Path

The absolute file path is a full URL (address) of the file path to access an internet file. For example,

```

```

Browser Output



In the above example,

https://cdn.programiz.com/sites/tutorial2program/files/pc_logo.svg is the complete location of the image.

Relative File Path

The relative file path describes the path of the file relative to the location of the current web page. For example,

```


```

File Path

✓ SERVER

✓ pages

✓ images

 programiz.png

 index.html

In the above example, *images/programiz.png* is the relative path. We are able to access the *images* folder because the *images* directory is located in the same folder as our HTML file (*index.html*).

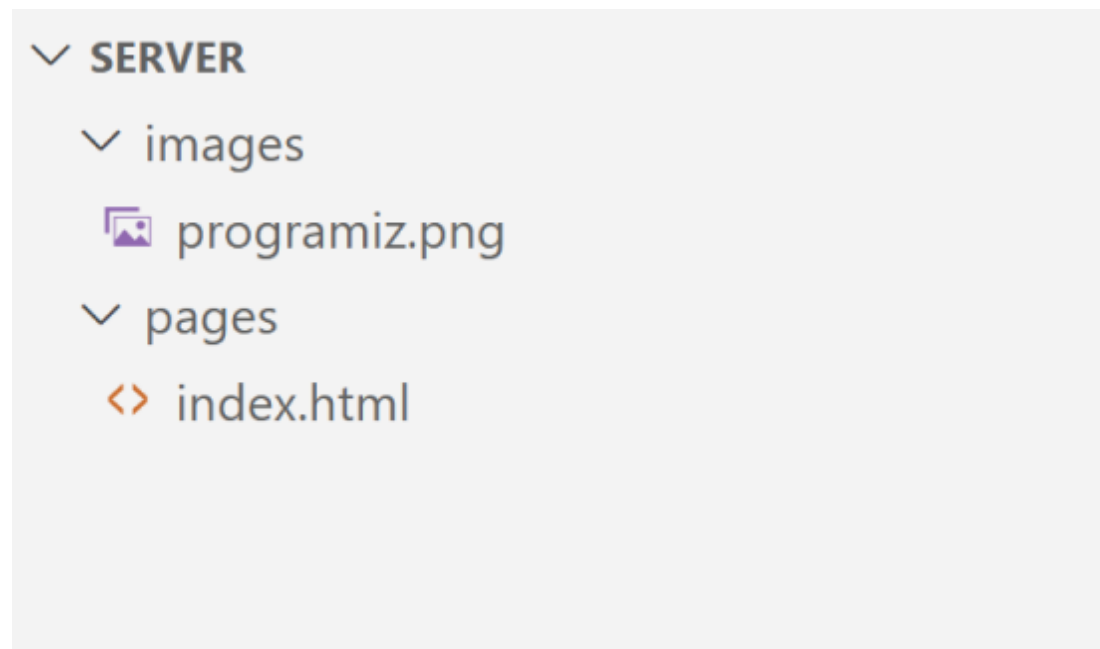
Access File Located Inside The Root Directory

Now, let's see how we access the folder that is present in the root directory. The root directory is the topmost directory that contains all related files and folders.

```

```

File Path



In the above example, *SERVER* is our root directory. The forward slash (/) represents the root directory. So, to access the *images* folder inside the root directory we use */images*. And to access the *programiz.png*, we use */images/programiz.png*.

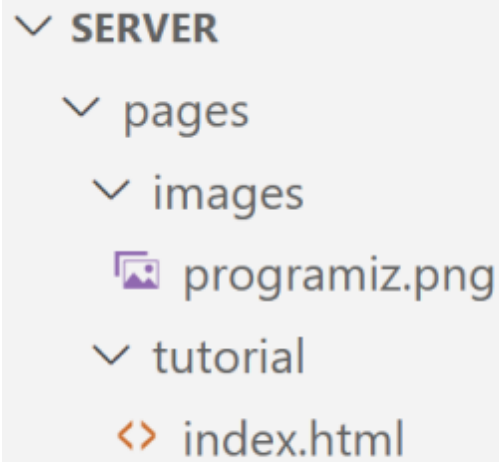
File located one level up

Now, let's see how we access the folder that is located one level up.

```

```

File Path



The `../` part of the path indicates that the file is located one level up in the directory hierarchy from the current location of the HTML file. In other words, it specifies the parent directory of the current directory.

So, to access the *images* folder inside the *pages* folder (the parent directory of the *tutorial* directory) we use `../images`. And to access the *programiz.png*, we use `../images/programiz.png`.

CSS Introduction

CSS (Cascading Style Sheets) is a language used for describing the visual presentation of web pages written in HTML and XML. CSS separates the presentation from the content of a web page, allowing developers to control the layout, typography, colors, and other visual aspects of a website.

CSS was first proposed by Håkon Wium Lie in 1994 and was later developed by a group of developers at the World Wide Web Consortium (W3C). Here are some key features and benefits of CSS:

1. Separation of presentation and content: With CSS, web developers can separate the presentation of a web page from its content. This allows for greater flexibility and easier maintenance of web pages.
2. Consistent styling: CSS provides a way to apply consistent styling to all pages of a website, making it easier to maintain and update the website as a whole.
3. Reusability: CSS styles can be applied to multiple elements throughout a website, reducing the amount of code required and making it easier to make changes.
4. Responsive design: CSS provides tools for creating responsive designs that adapt to different screen sizes and devices.
5. Accessibility: CSS can be used to create more accessible websites by controlling the presentation of content and making it easier for users with disabilities to navigate and understand.

CSS has gone through several versions since its inception, with CSS3 being the latest version. CSS3 introduces new features such as gradients, animations, and transformations, making it even more powerful for creating dynamic and visually engaging web pages.

CSS History

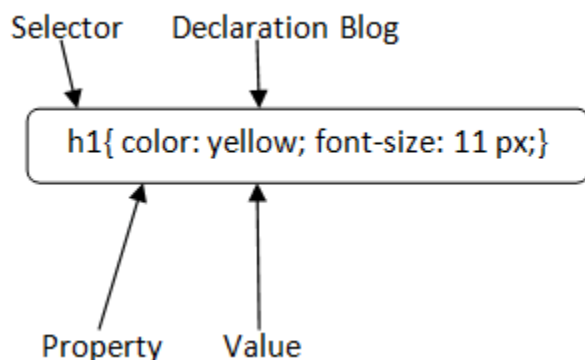
CSS has gone through several versions and updates, each one adding new features and capabilities. Here is a brief history of CSS:

1. **CSS1 (1996):** This was the first version of CSS to be standardized by the W3C. It provided basic styling capabilities such as font, color, and background properties.

2. **CSS2 (1998)**: This version added more advanced styling capabilities, such as positioning, floating, and layout properties. It also introduced support for media types, allowing developers to create different stylesheets for different devices and screen sizes.
3. **CSS2.1 (2004)**: This version was a revision of CSS2, adding new features and clarifying some of the language in the specification.
4. **CSS3 (2001-2017)**: CSS3 is not a single specification, but rather a collection of modules that add new features to CSS. Some of the key modules include selectors, backgrounds and borders, text effects, 2D/3D transformations, animations, and media queries. The development of CSS3 spanned over a decade, with new modules being added over time.
5. **CSS4 (proposed)**: There is currently no official CSS4 specification, but there are ongoing discussions and proposals for new features and capabilities that could be included in a future version of CSS.

CSS has become an essential tool for web developers, allowing them to create visually appealing and responsive websites. With new features and updates being added over time, CSS continues to evolve and adapt to the changing needs of the web.

CSS Syntax



1. **Selector**: CSS selectors are used to target specific HTML elements to apply styles. Selectors can be based on element type, class, ID, attributes, and more. For example, the selector for all h1 elements is written as h1.
2. **Declaration Block**: The declaration block contains one or more declarations, separated by semicolons. Each declaration consists of a property and a value,

separated by a colon. For example, the declaration block for styling h1 elements with a red font color is written as {color: red ;}.

3. **Property:** CSS properties are used to specify the visual style of an element, such as font size, color, and background. For example, the color property is used to set the font color.
4. **Value:** The value of a property specifies the specific style to apply to the element. For example, the value for the color property can be red, blue, or any other color.

How to add CSS

There are three ways to add CSS to an HTML document:

1. **Inline CSS:** Inline CSS is added directly to an HTML element using the style attribute. For example:

```
<h1 style="color: red;">Hello, World!</h1>
```

In this example, the style attribute is used to set the color of the h1 element to red.

2. **Internal CSS:** Internal CSS is added to the head section of an HTML document using the style element. For example:

```
<!DOCTYPE html>
<html>
<head>
<style>
  h1 {
    color: red;
  }
</style>
</head>
<body>
  <h1>Hello, World!</h1>
</body>
</html>
```

In this example, the style element is used to set the color of all h1 elements to red.

3. **External CSS:** External CSS is added to an HTML document as a separate file, which is linked to the HTML document using the link element. For example:

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

In this example, the link element is used to link to an external CSS file called style.css.

Using external CSS is the most common and recommended way to add CSS to an HTML document, as it allows for greater organization, reuse, and maintainability of styles across multiple pages.

CSS Comments

SS comments are used to add notes or descriptions to a CSS file, without affecting the styles applied to the HTML elements. There are two ways to add comments in CSS:

1. **Single-line comments:** Single-line comments start with `//` and end at the end of the line. For example:

```
/* This is a single-line comment */
h1 {
  color: red; /* This is another single-line comment */
}
```

In this example, the first comment applies to the entire line, while the second comment only applies to the color property.

Multi-line comments: Multi-line comments start with `/*` and end with `*/`. They can span multiple lines and can be used to add longer descriptions or notes to the CSS file. For example:

```
/*  
  This is a multi-line comment that can span multiple lines  
  and be used to add longer descriptions or notes to the CSS file.  
*/  
h1 {  
  color: red;  
}
```

In this example, the multi-line comment applies to all the lines between `/*` and `*/`.

CSS comments are useful for documenting the purpose of CSS rules, explaining the reasoning behind certain design choices, or temporarily disabling or testing styles without deleting them from the file. It is good practice to use comments sparingly and only when necessary, to keep the code clean and readable

CSS Background

In CSS, the background property is used to set the background color, image, and other properties of an element. Here is an overview of the different aspects of the background property:

1. **Background color:** The `background-color` property sets the background color of an element. It can be set to a named color, a hexadecimal color value, or an RGB color value. For example:

```
div {  
  background-color: red;  
}
```

2. **Background image:** The `background-image` property sets the background image of an element. It can be set to a URL pointing to an image file, or to a linear gradient, radial gradient, or repeating pattern. For example:

```
div {  
  background-image: url("bg-image.jpg");  
}
```

3. **Background repeat:** The background-repeat property sets whether and how the background image should repeat within the element. It can be set to repeat (default), no-repeat, repeat-x, or repeat-y. For example:

```
div {  
  background-repeat: no-repeat;  
}
```

4. **Background position:** The background-position property sets the position of the background image within the element. Position sets the position of the background image. You can use values like top, right, bottom, left, center, or use specific coordinates like 50% 50%.. For example:

```
div {  
  background-position: center;  
}
```

5. **Background size:** The background-size property sets the size of the background image. It can be set to a keyword value such as cover or contain, or to a percentage or pixel value. For example:

```
div {  
  
  background-size: cover;  
}
```

5. **background-attachment:** sets whether the background image should scroll with the rest of the page or remain fixed in place. You can use values like scroll, fixed, and local.

Background shorthand: The background property can be used as a shorthand to set all of the above properties at once, in the order of background-color, background-image, background-repeat, background-position, and background-size. For example:

background: [background-color] [background-image] [background-repeat] [background-attachment] [background-position];

```
div {  
  background: url("bg-image.jpg") no-repeat center/cover;  
}
```

In this example, the background color is not specified, so it will default to transparent.

The background property is a versatile and powerful tool for styling the background of an element, and can be used to create a wide range of effects, from simple solid colors to complex layered backgrounds with multiple images and patterns.

CSS Border

In CSS, the border property is used to create a border around an element. It has three sub-properties: border-width, border-style, and border-color.

The border-width sub-property controls the thickness of the border. You can specify a single value to create a border of equal thickness on all four sides of the element, or you can specify individual values for each side. For example:

```
border-width: 1px; /* creates a border with 1-pixel thickness on all sides */
```

```
border-width: 1px 2px 3px 4px; /* creates a border with 1-pixel thickness on the top, 2-  
pixel thickness on the right, 3-pixel thickness on the bottom, and 4-pixel thickness on  
the left */
```

The border-style sub-property controls the style of the border. You can use values such as solid, dashed, dotted, double, groove, ridge, inset, and outset. For example:

```
border-style: solid; /* creates a solid border */
```

```
border-style: dotted dashed; /* creates a dotted border on the top and bottom and a  
dashed border on the left and right */
```

The border-color sub-property controls the color of the border. You can use any valid CSS color value. For example:

```
border-color: red; /* creates a red border */
```

```
border-color: #333333; /* creates a border with a dark gray color */
```

```
border-color: red green blue yellow; /* creates a border with different colors on each  
side */
```

You can combine all three sub-properties into a single border property. Here's an example:

```
css
border: 1px solid red; /* creates a 1-pixel solid red border */
```

You can also specify each sub-property individually:

```
border-width: 1px;
border-style: solid;
border-color: red;
```

It's important to note that you can set the border property separately for each side of the element using the following sub-properties:

border-top, border-right, border-bottom, border-left

For example:

```
border-top: 1px solid red; /* creates a 1-pixel solid red border on the top side */
```

CSS Margin

In CSS, the margin property is used to create space around an element, outside of any defined borders. It is used to control the gap or distance between the element and the adjacent elements.

The margin property has four values: margin-top, margin-right, margin-bottom, and margin-left. You can specify any combination of these values to create different margins on each side of an element.

Here's an example of how to set a margin of 20 pixels on all sides of an element:

```
margin: 20px;
```


You can also specify different margin values for each side of the element by listing the values in the following order: margin-top, margin-right, margin-bottom, and margin-left. For example, to set a margin of 10 pixels on the top and bottom and 20 pixels on the left and right, you could use the following:

```
margin: 10px 20px;
```

Or you can also specify margin values individually for each side like this:

```
css
margin-top: 10px;
margin-right: 20px;
margin-bottom: 10px;
margin-left: 20px;
```

You can also use negative values for the margin property to pull an element closer to its neighbors or to overlap with them. However, using negative margins can also cause unintended effects, so use them with caution.

It's important to note that the margin property does not affect the size or position of the element itself, only the space around it. If you want to control the position of an element, you should use the position property instead.

CSS Padding

In CSS, the padding property is used to create space between the content of an element and its borders. It is used to control the space between the element's content and its border.

The padding property also has four values: padding-top, padding-right, padding-bottom, and padding-left. You can specify any combination of these values to create different padding on each side of an element.

Here's an example of how to set a padding of 20 pixels on all sides of an element:

```
css
padding: 20px;
```

You can also specify different padding values for each side of the element by listing the values in the following order: padding-top, padding-right, padding-bottom, and padding-

left. For example, to set a padding of 10 pixels on the top and bottom and 20 pixels on the left and right, you could use the following:

```
css
padding: 10px 20px;
```

Or you can also specify padding values individually for each side like this:

```
css
padding-top: 10px;
padding-right: 20px;
padding-bottom: 10px;
padding-left: 20px;
```

It's important to note that the padding property affects the size and position of the element's content, not its borders or margins. If you want to control the space between an element and its neighbors, you should use the margin property instead.

Using the padding property can help you create visually pleasing layouts by creating space around an element's content, separating it from other elements, and improving the readability of your web page.

CSS Text Formatting

In CSS, there are several properties that you can use to format text. Here are some of the most commonly used properties:

1. font-family: specifies the font family for the text.

```
css
font-family: Arial, sans-serif;
```

2. font-size: specifies the font size for the text.

```
css
font-size: 16px;
```

3. font-weight: specifies the boldness of the text.

```
css
font-weight: bold;
```

4. text-align: specifies the alignment of the text.

CSS

text-align: center;

5. text-decoration: specifies the decoration of the text, such as underline or strikethrough.

CSS

text-decoration: underline;

6. text-transform: specifies the capitalization of the text.

CSS

text-transform: uppercase;

7. line-height: specifies the height of each line of text.

Arduino

line-height: 1.5;

8. letter-spacing: specifies the spacing between each letter.

CSS

letter-spacing: 1px;

9. word-spacing: specifies the spacing between each word.

CSS

word-spacing: 2px;

10. color: specifies the color of the text.

CSS

color: #333333;

You can combine these properties to create different styles for your text. For example:

CSS

font-family: Arial, sans-serif;

font-size: 18px;

font-weight: bold;

```
text-align: center;
text-decoration: underline;
text-transform: uppercase;
line-height: 1.5;
letter-spacing: 1px;
word-spacing: 2px;
color: #333333;
```

This would create a bold, underlined, uppercase heading with centered text, using the Arial font with a size of 18 pixels and a dark gray color. The line height would be 1.5 times the font size, and there would be 1 pixel of space between each letter and 2 pixels of space between each word.

CSS Position

In CSS, the position property is used to specify the type of positioning used for an element. There are four possible values for this property: static, relative, absolute, and fixed.

1. static: This is the default value for the position property. It means that the element is positioned according to the normal flow of the document. The top, right, bottom, and left properties have no effect on static elements.

```
arduino
position: static;
```

2. relative: This value positions an element relative to its normal position in the document flow. The element can be moved with the top, right, bottom, and left properties.

```
css
position: relative;
top: 10px;
left: 20px;
```

3. absolute: This value positions an element relative to its closest positioned ancestor (an ancestor element with a position value other than static). If no positioned ancestor exists, the element is positioned relative to the initial containing block (usually the body element). The element can be moved with the top, right, bottom, and left properties.

```
css
position: absolute;
top: 50px;
right: 0;
```

4. **fixed:** This value positions an element relative to the viewport (the browser window). The element remains in the same position even if the page is scrolled. The element can be moved with the top, right, bottom, and left properties.

```
css
position: fixed;
top: 0;
right: 0;
```

The position property can be used to create complex layouts and effects, such as sticky headers, tooltips, and pop-up windows. It's important to use it carefully and understand its behavior, as it can affect the layout and positioning of other elements on the page.

CSS Selectors

In CSS, selectors are used to target specific HTML elements and apply styles to them. There are several types of selectors that you can use:

- We can divide CSS selectors into five categories:
- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

Simple selectors css

In CSS, Simple Selectors are the most basic type of selector that target elements based on their element type, class, or ID. There are three types of simple selectors:

1. **Element Type Selector:** Targets all elements of a specific type.

```
css
p {
```

```
color: blue;
}
```

2. Class Selector: Targets elements with a specific class attribute.

```
css
.my-class {
  font-weight: bold;
}
```

3. ID Selector: Targets elements with a specific ID attribute.

```
arduino
#my-id {
  background-color: yellow;
}
```

You can also use multiple simple selectors to create more specific selectors. For example, you can target a p element with a class of my-class using the following selector:

```
css
p.my-class {
  font-size: 18px;
}
```

This selector targets all p elements with a class of my-class and applies a font size of 18 pixels. Simple selectors are the foundation of CSS selectors and provide a powerful way to style HTML elements

Combinator selectors css

In CSS, combinator selectors are used to combine multiple simple selectors and create more specific selectors. There are four types of combinator selectors:

1. Descendant Selector (space): Targets elements that are descendants of another element.

```
css
div p {
```

```
color: red;
}
```

This selector targets all p elements that are descendants of a div element and applies a red color.

2. Child Selector (>): Targets elements that are direct children of another element.

```
css
ul > li {
  list-style: none;
}
```

This selector targets all li elements that are direct children of a ul element and removes the list-style.

3. Adjacent Sibling Selector (+): Targets the element that immediately follows another element.

```
css
h2 + p {
  font-size: 18px;
}
```

This selector targets the p element that immediately follows an h2 element and sets the font size to 18 pixels.

4. General Sibling Selector (~): Targets all elements that are siblings of another element.

```
css
h2 ~ p {
  font-size: 16px;
}
```

This selector targets all p elements that are siblings of an h2 element and sets the font size to 16 pixels.

Combinator selectors allow you to create more specific selectors and apply styles to specific elements on the page.

Pseudo-class selectors

Pseudo-class selectors are special selectors in CSS that allow you to select and style elements based on their state or relationship to other elements. Here is a list of some commonly used pseudo-class selectors:

1. `:hover` - selects an element when the user hovers over it with their cursor
2. `:active` - selects an element while it is being clicked on or activated
3. `:focus` - selects an element when it has focus (such as when it is selected using the keyboard)
4. `:visited` - selects links that have already been visited by the user
5. `:link` - selects links that have not yet been visited by the user
6. `:first-child` - selects the first child element of its parent element
7. `:last-child` - selects the last child element of its parent element
8. `:nth-child(n)` - selects the nth child element of its parent element (where n is a number)
9. `:nth-of-type(n)` - selects the nth element of a specific type within its parent element (where n is a number)
10. `:not(selector)` - selects all elements that do not match the specified selector
11. `:empty` - selects elements that have no children or content
12. `:checked` - selects radio buttons or checkboxes that have been checked by the user
13. `:disabled` - selects form elements that are currently disabled
14. `:enabled` - selects form elements that are currently enabled

Pseudo-Element selectors

Here are some commonly used pseudo-element selectors in CSS:

1. `::before` - creates a new element before the content of the selected element
2. `::after` - creates a new element after the content of the selected element
3. `::first-line` - selects the first line of text within the selected element
4. `::first-letter` - selects the first letter of the first line of text within the selected element
5. `::selection` - selects the portion of text that has been highlighted by the user
6. `::placeholder` - selects the placeholder text within an input or textarea element
7. `::marker` - selects the marker of a list item

These selectors can be used to apply styles to specific parts of an element's content or to add new content to an element. For example, you can use the `::before` selector to add an icon or a decorative element before a heading, or use the `::first-letter` selector to style the first letter of a paragraph differently from the rest of the text. The `::selection` selector can be used to style the text that a user selects on the page.

Attribute selectors css

Pseudo-class selectors list

3 / 3

Here are some commonly used pseudo-class selectors in CSS:

1. `:hover` - selects an element when the user hovers over it with their cursor
2. `:active` - selects an element while it is being clicked on or activated
3. `:focus` - selects an element when it has focus (such as when it is selected using the keyboard)
4. `:visited` - selects links that have already been visited by the user
5. `:link` - selects links that have not yet been visited by the user
6. `:first-child` - selects the first child element of its parent element
7. `:last-child` - selects the last child element of its parent element
8. `:nth-child(n)` - selects the nth child element of its parent element (where n is a number)
9. `:nth-of-type(n)` - selects the nth element of a specific type within its parent element (where n is a number)
10. `:not(selector)` - selects all elements that do not match the specified selector
11. `:empty` - selects elements that have no children or content
12. `:checked` - selects radio buttons or checkboxes that have been checked by the user
13. `:disabled` - selects form elements that are currently disabled
14. `:enabled` - selects form elements that are currently enabled

These selectors can be used to apply different styles to different states of an element, or to select specific elements based on their position or properties.

Pseudo-Element selectors

Here are some commonly used pseudo-element selectors in CSS:

1. `::before` - creates a new element before the content of the selected element

2. `::after` - creates a new element after the content of the selected element
3. `::first-line` - selects the first line of text within the selected element
4. `::first-letter` - selects the first letter of the first line of text within the selected element
5. `::selection` - selects the portion of text that has been highlighted by the user
6. `::placeholder` - selects the placeholder text within an input or textarea element
7. `::marker` - selects the marker of a list item

These selectors can be used to apply styles to specific parts of an element's content or to add new content to an element. For example, you can use the `::before` selector to add an icon or a decorative element before a heading, or use the `::first-letter` selector to style the first letter of a paragraph differently from the rest of the text. The `::selection` selector can be used to style the text that a user selects on the page.

Attribute selectors css

Attribute selectors are CSS selectors that select elements based on the presence or value of a specific attribute. Here are some commonly used attribute selectors:

1. `[attribute]` - selects elements that have the specified attribute, regardless of its value
2. `[attribute=value]` - selects elements that have the specified attribute with a value that exactly matches the given value
3. `[attribute^=value]` - selects elements that have the specified attribute with a value that starts with the given value
4. `[attribute$=value]` - selects elements that have the specified attribute with a value that ends with the given value
5. `[attribute*=value]` - selects elements that have the specified attribute with a value that contains the given value

For example, the following selector would select all `<a>` elements that have a href attribute with a value of "<https://www.example.com>":

```
a[href="https://www.example.com"]
```

The following selector would select all `` elements that have an alt attribute:

```
img[alt]
```

Attribute selectors can be very useful for selecting specific elements based on their attributes and values. They are often used in combination with other selectors to create more complex selection rules.

Flex

Flexbox is a CSS layout module that makes it easy to create flexible and responsive layouts. The flexbox model uses two main components: flex containers and flex items.

A flex container is an element that contains one or more flex items. To create a flex container, you can use the `display: flex` property on the parent element. Once an element becomes a flex container, its child elements become flex items.

Here are some common properties used on flex containers:

- `display: flex;` - sets the element as a flex container
- `flex-direction: row | row-reverse | column | column-reverse;` - sets the direction in which the flex items are laid out
- `justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right;` - aligns the flex items along the main axis of the flex container
- `align-items: stretch | flex-start | flex-end | center | baseline;` - aligns the flex items along the cross axis of the flex container
- `flex-wrap: nowrap | wrap | wrap-reverse;` - sets whether the flex items should wrap to a new line when there is not enough space on the current line

A flex item is an element that is a child of a flex container. Here are some common properties used on flex items:

- `flex-grow: <number>;` - sets the amount that the flex item should grow relative to the other flex items
- `flex-shrink: <number>;` - sets the amount that the flex item should shrink relative to the other flex items
- `flex-basis: <length> | auto;` - sets the initial size of the flex item before any remaining space is distributed
- `order: <number>;` - sets the order in which the flex item appears in the flex container

Flexbox provides a powerful and flexible way to create layouts that adapt to different screen sizes and devices. By using the properties above, you can create complex and responsive layouts that adjust based on the available space and content.

Responsive Css

Responsive CSS is a design approach in which web pages are designed and coded to provide an optimal viewing experience across different devices and screen sizes. The goal of responsive design is to make sure that web pages are easy to read and navigate on desktop computers, laptops, tablets, and smartphones.

To create a responsive design, web developers typically use a combination of HTML, CSS, and JavaScript. Here are some common techniques used in responsive CSS:

1. Media queries - Media queries allow you to apply different styles based on the width of the viewport, or the device that the page is being viewed on. For example, you might use a media query to adjust the font size or layout of a page for smaller screens.
2. Fluid grids - Fluid grids allow you to create layouts that adjust to different screen sizes by using relative units like percentages instead of fixed pixel values.
3. Responsive images - Responsive images use techniques like srcset and sizes attributes to serve different image sizes based on the viewport size and device type.
4. Flexbox - Flexbox is a layout module that makes it easier to create flexible and responsive layouts without using floats or positioning.
5. CSS frameworks - CSS frameworks like Bootstrap and Foundation provide pre-built CSS and HTML templates that are designed to be responsive out of the box.

By using these techniques, web developers can create websites that look great and function well across a wide range of devices and screen sizes, from large desktop monitors to small smartphone screens. Responsive design is an essential part of modern web development, as more and more people access the internet on mobile devices.

