HTML Media

- Image
- Audio
- Video
- YouTube, etc.

Image

<img> element is used for displaying images on a webpage.

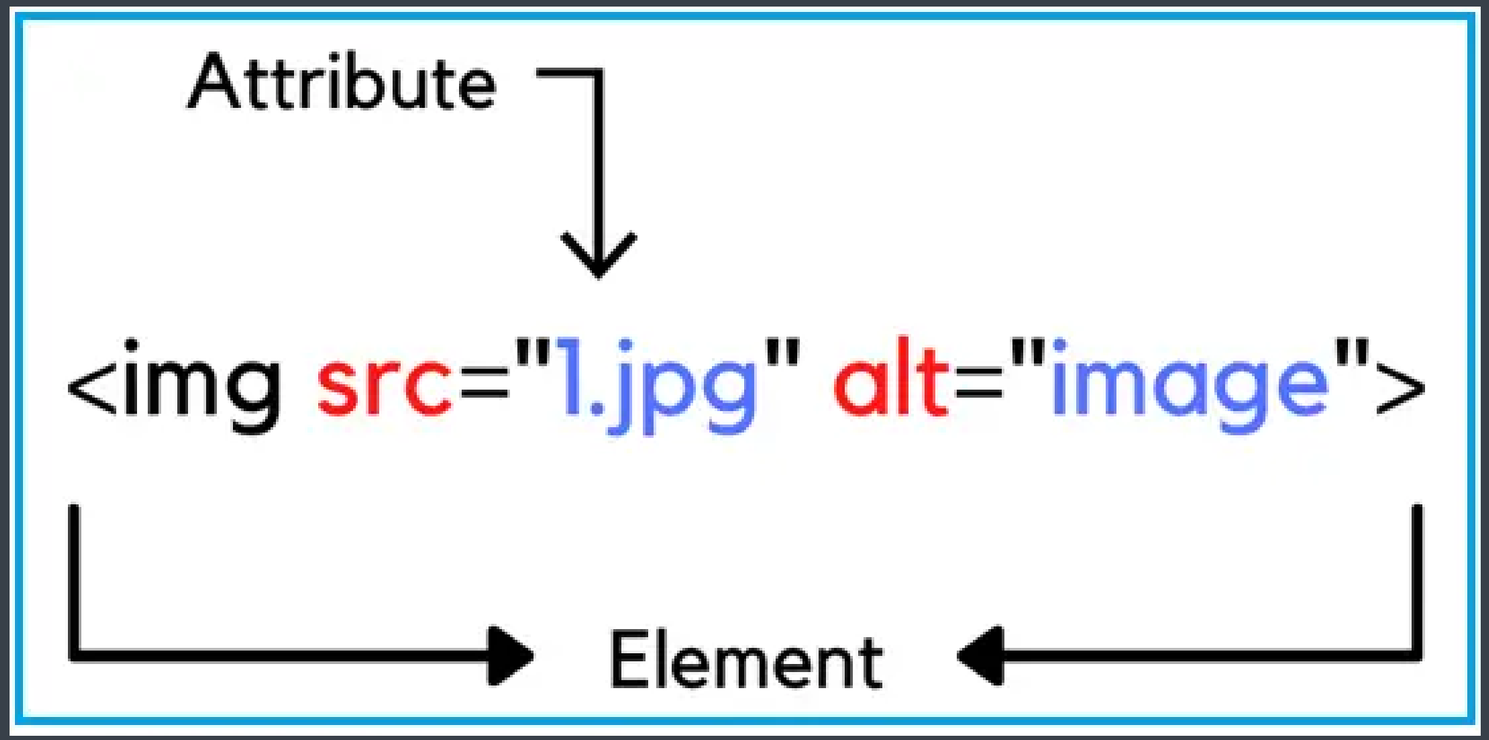


Image Attributes

- **src:** Specifies the source URL of the image.
- **alt:** Provides alternative text for the image (important for accessibility).

Audio

Used for embedding audio files.

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

Audio Attributes

- **src:** Specifies the source URL of the audio file.
- **controls:** Adds playback controls (play, pause, volume, etc.).

Video

Used for embedding video files.

```
<video src="video.mp4" controls width="480" height="360"></video>
```

Video Attributes

- **src:** Specifies the source URL of the video file.
- **controls:** Adds video playback controls.
- **width and height:** Set the dimensions of the video.

Embedding Other Media

use <iframe> tag for embedding external media like YouTube

```
<iframe width="560" height="315" src="" frameborder="0" allowfullscreen></iframe>
```

COMMENTS in HTML

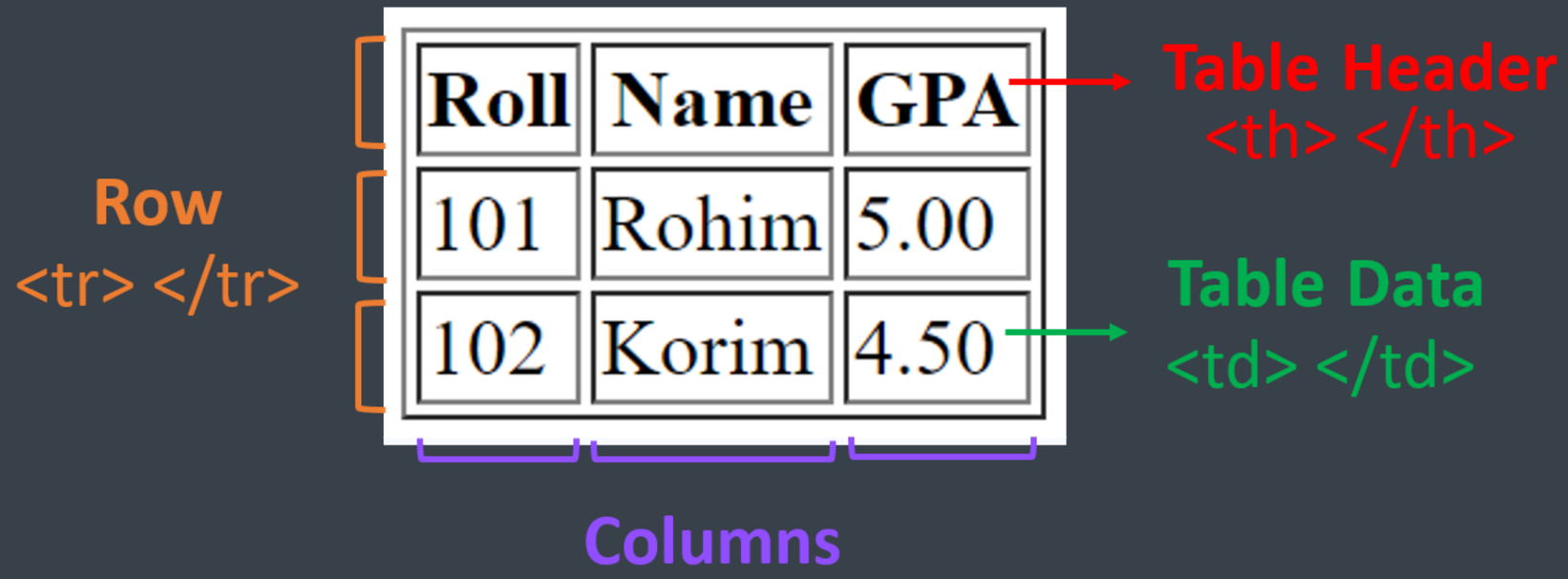
```
<!--
    This is a
    multi-line
    comment
-->
```

HTML Tables

Basic Table Structure:

- <table> (table container)
- <th> (table header)
- <td> (table data)
- <tr> (table row)

Basic Table Structure:



<table> in HTML

Start with the <table> element, which serves as the container for you

```
<table>
  <!-- Table content goes here -->
</table>
```

<tr> and <th>

Within each <tr>, you can use the <th> (table header) element to create header cells for the table.

```
<table>
  <tr>
    <th>Header 1</th>
    <th>Header 2</th>
  </tr>
</table>
```

<tr> and <td>

After the header row, you can use the <td> (table data) element to create cells containing data for each row

```
<tr>
  <td>Data 1</td>
  <td>Data 2</td>
</tr>
```

Example

```
<table border="1" cellpadding="5" cellspacing="3">
        <tr>
            <th>Firstname</th>
            <th>Lastname</th>
            <th>Age</th>
        </tr>
        <tr>
            <td>Aryan</td>
            <td>Gupta</td>
            <td>23</td>
        </tr>
        <tr>
            <td>John</td>
            <td>Reece</td>
            <td>32</td>
        </tr>

    </table>
```

Output

Firstname	Lastname	Age
Aryan	Gupta	23
John	Reece	32

Table Code

```
<table>
  <tr>
    <th>Header 1</th>
    <th>Header 2</th>
  </tr>
  <tr>
    <td>Data 1</td>
    <td>Data 2</td>
  </tr>
</table>
```

Spanning Rows and Columns

- The colspan attribute is used to join two or more columns into a single column.
- Similarly, you will use the rowspan if you want to combine two or more rows.

Colspan

```
<table border="1">
  <tr>
    <th>Header 1</th>
    <th colspan="2">Header 2</th>
  </tr>
  <tr>
    <td>Data 1</td>
    <td>Data 2</td>
    <td>Data 3</td>
  </tr>
</table>
```

Rowspan

```
<table border="1">
  <tr>
    <th rowspan="2">Header 1</th>
    <th>Header 2</th>
    <th>Header 3</th>
  </tr>
  <tr>
    <td>Data 1</td>
    <td>Data 2</td>
  </tr>
</table>
```

Combining colspan and rowspan

```
<table border="1">
  <tr>
    <th rowspan="2">Header 1</th>
    <th>Header 2</th>
    <th colspan="2">Header 3</th>
  </tr>
  <tr>
    <td>Data 1</td>
    <td>Data 2</td>
  </tr>
</table>
```

Forms

Introduction HTML Forms

- Forms are interactive elements used to collect user input
- Allows users to input data, which can be sent to a server for processing or used within the web page itself.
- Forms are used for various purposes like user registration, login, search boxes, and more.

HTML Form Elements:

1. **Form**
2. **Input**
3. **Label**
4. **Textarea**
5. **Button**
6. **Select**

- **Tag**
- **Attributes**
- **Types**

Form tag

- The <form> element is used to create a container for all form elements.

```
<form action="/submit" method="POST">
    <!-- Form elements go here -->
</form>
```

Form attributes

- action: This attribute specifies the URL where the form data will be sent.
- method: This attribute defines how the data is sent to the server (e.g., "GET" or "POST").

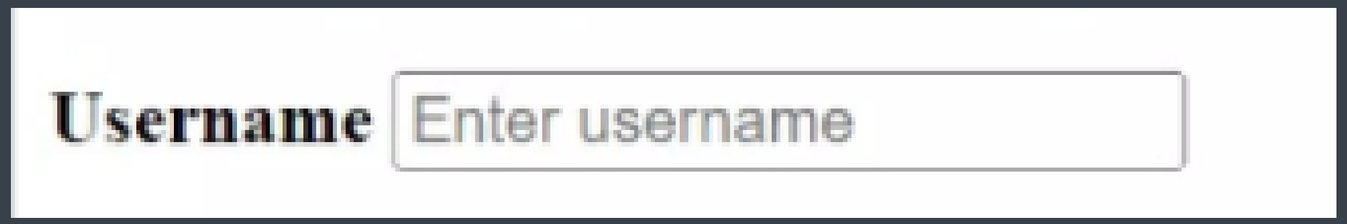
Input & Label tag

- The <input> element is used for various types of user input, such as text, checkboxes, radio buttons, and more.
- The <label> element contains the text "Username:" which serves as the label for the input field.

Input & Label tag

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

Output



Label Attributes

- for: Associates a label with an input element by referencing the input's id attribute.

Input Attributes

- Type: Determines the input's format (e.g., text, password, email).
- Name: Gives the input a unique server-side identifier.
- ID: Provides a unique label for the input.
- Value: Sets the initial input content.
- Placeholder: Offers a hint for the input.
- Required: Ensures input completion before form submission.
- Size & Maxlength: Control text input appearance and length.
- Min & Max: Define numerical input limits.
- Disabled: Prevents input interaction.
- Readonly: Makes the input unEditable.

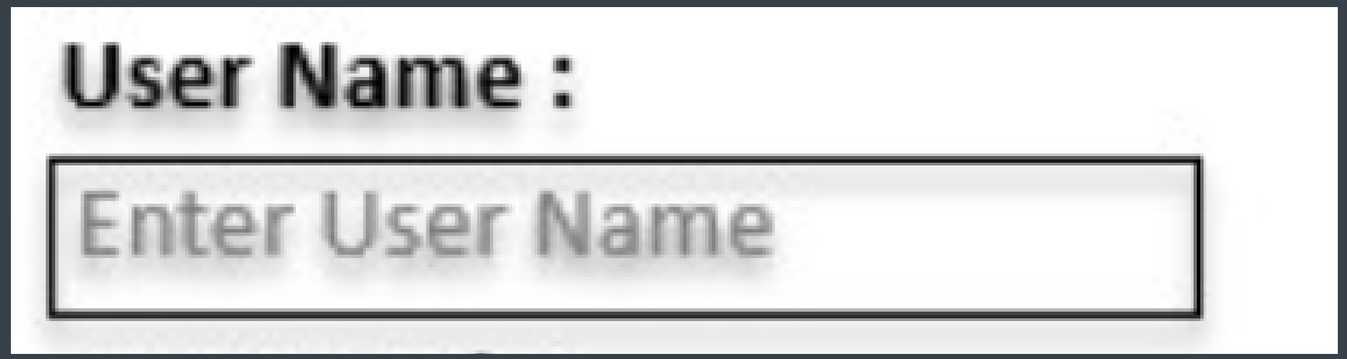
(HTML Form) Input types

1. **Text**
2. **Password**
3. **Radio**
4. **Checkbox**

Text

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

Output



Password

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

Output

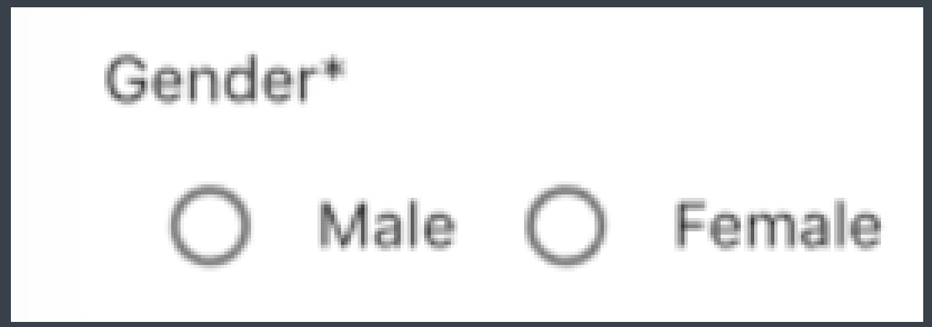
Radio

```
<label>Gender:</label>
<input type="radio" id="male" name="gender" value="male">
<label for="male">Male</label>

<input type="radio" id="female" name="gender" value="female">
<label for="female">Female</label>
```

Radio

Output



Checkbox

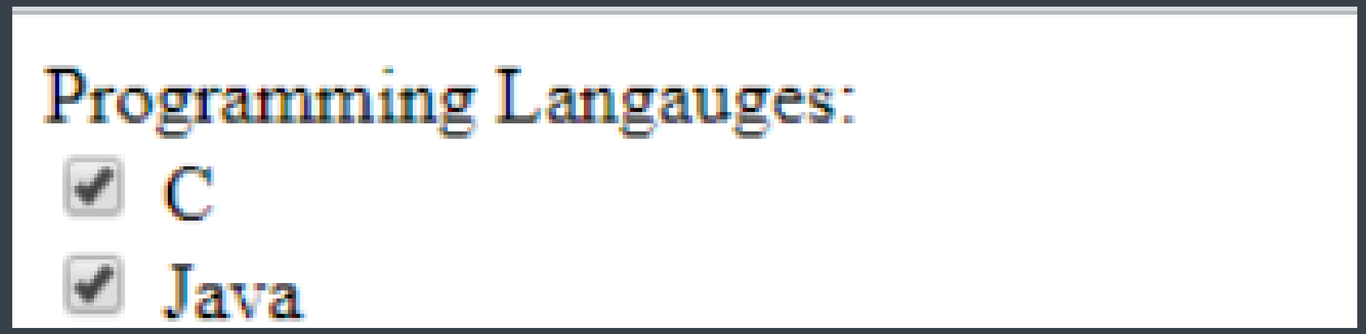
```
<label for="programming-languages">Programming Languages:</label>

<input type="checkbox" id="c" name="languages" value="C">
<label for="c">C</label>

<input type="checkbox" id="java" name="languages" value="Java">
<label for="java">Java</label>
```

Checkbox

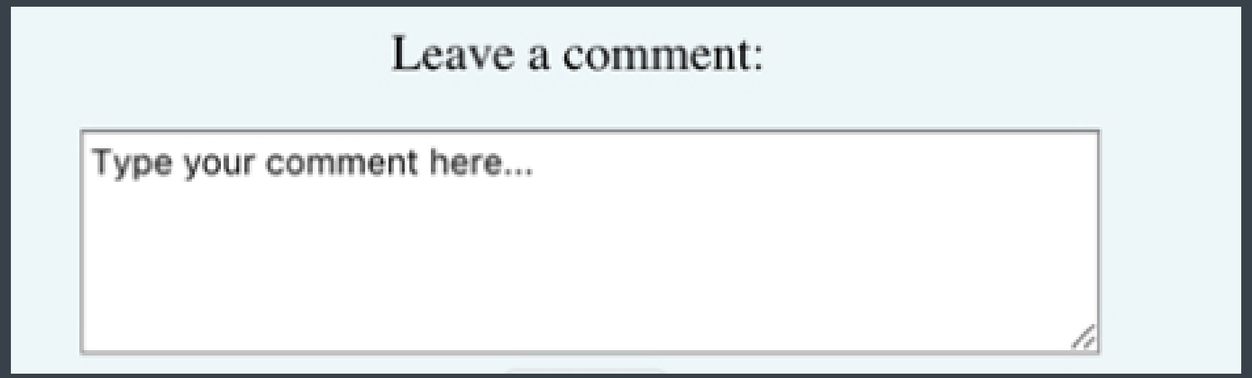
Output



Textarea tag

```
<label for="comment">Leave a comment:</label>
<textarea id="comment" name="comment" rows="4" cols="50"
          placeholder="Type your comment here..."></textarea>
```

Output



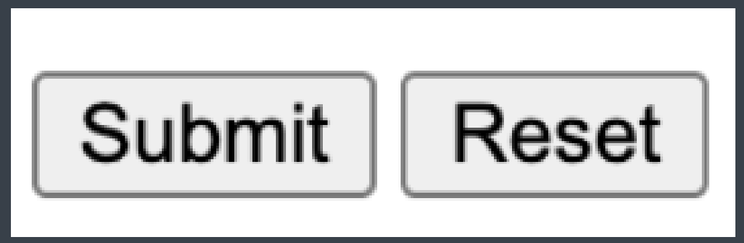
Textarea Attributes

- **rows and cols:** Define the visible number of rows and columns in a <textarea>.

Button tag

```
<button type="submit">Submit</button>
<button type="reset">Reset</button>
```

Output



Button Attributes

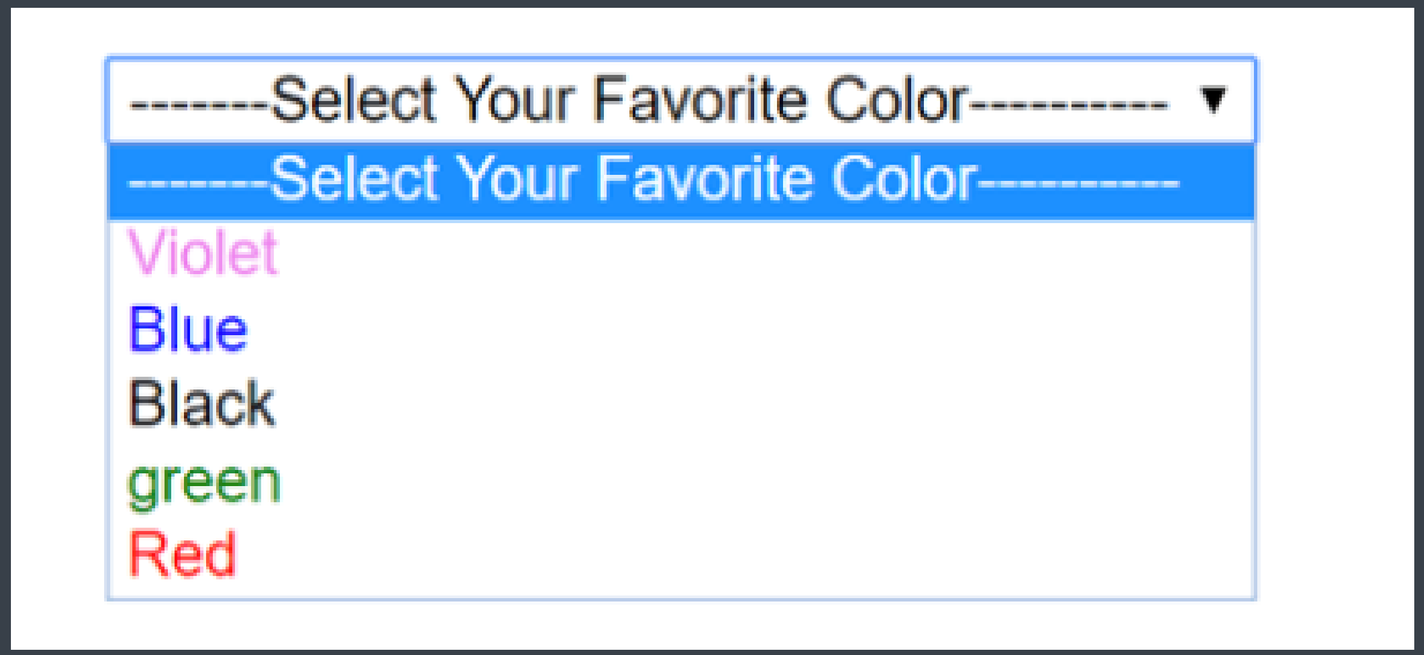
- **type:** Specifies the type of button (e.g., submit, reset, or button).

Select tag

```
<label for="color">Select your favorite color:</label>
<select id="color" name="color">
    <option value="violet">Violet</option>
    <option value="blue">Blue</option>
    <option value="black">Black</option>
    <option value="green">Green</option>
    <option value="red">Red</option>
</select>
```

Select tag

Output



HTML Form Validation

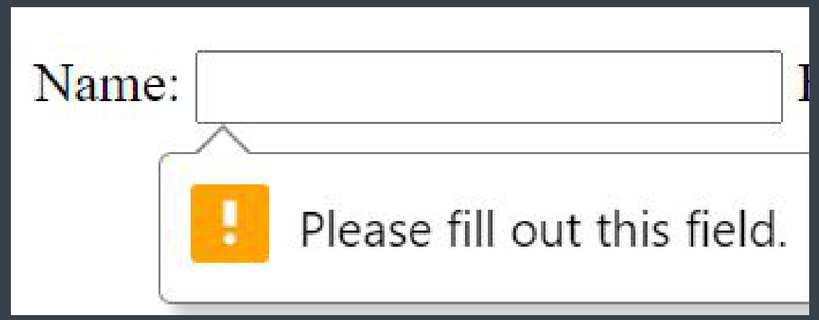
1. **Required**
2. **email**
3. **Number**
4. **Url**

Required

Required: The required attribute ensure that users provide input in those fields before submitting the form.

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

Output

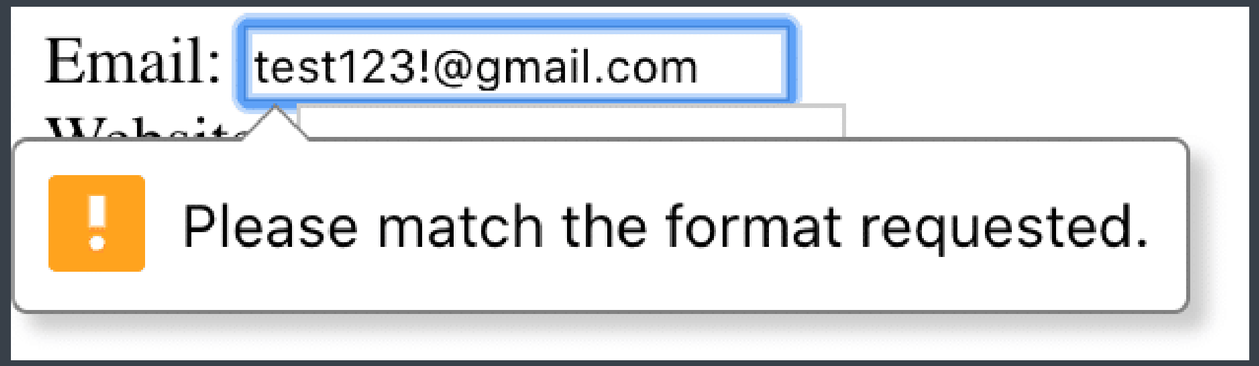


Email

This input type guarantees an email address is formatted correctly.

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

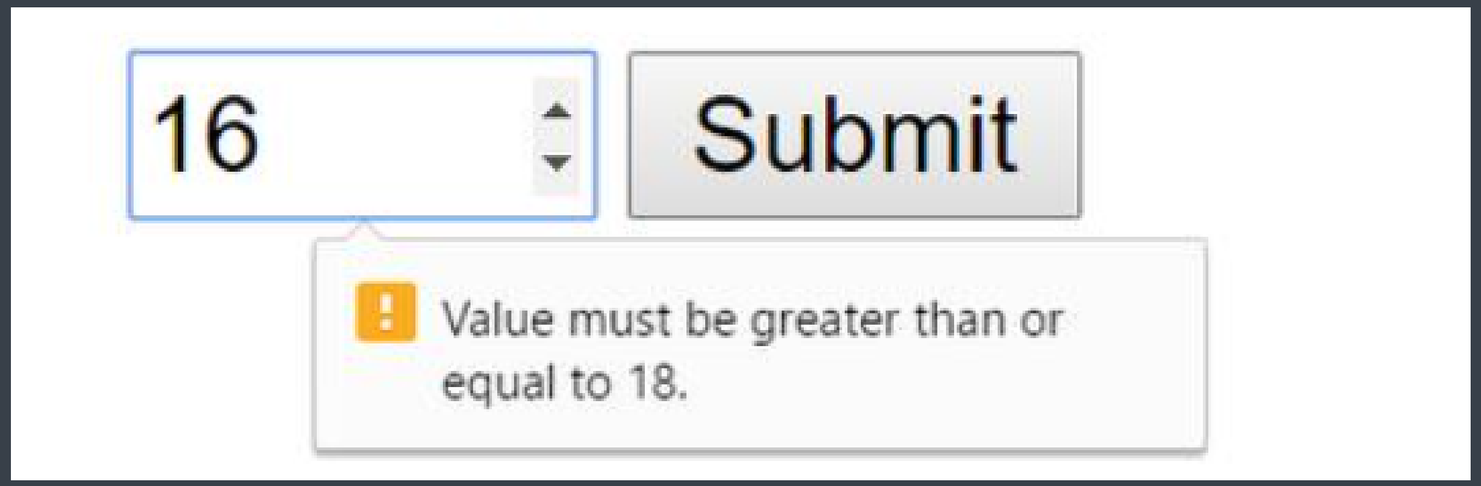
Output



Number

```
<input type="number" name="age" min="18" max="100">
```

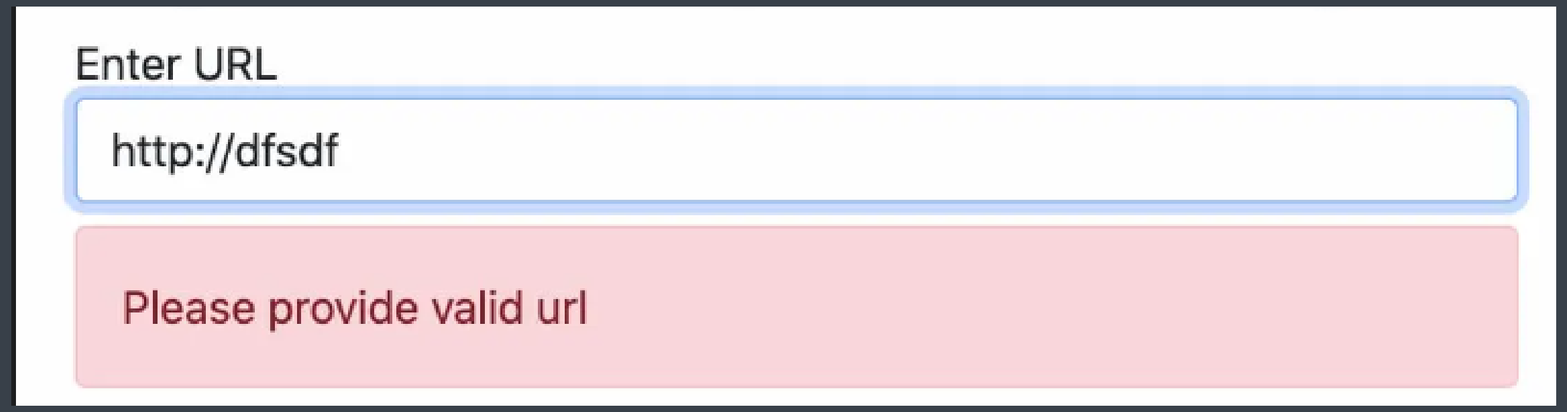
Output



Url

```
<input type="url" name="website">
```

Output



Introduction to Web Development

What you will Learn ?

1. Basics of Web Development
2. Introduction to HTML
3. Setting Up Your Development Environment

Web Development

1. **HTML (Hypertext Markup Language):**
    - It structures web content using tags.
    - Tags define headings, paragraphs, links, and more.
2. **CSS (Cascading Style Sheets):**
    - It styles web content, changing colors, fonts, and layouts.

Web Development

3. **JavaScript:**

- **It adds interactivity and dynamic features to websites.**

Intro to HTML

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content

Structure of HTML Document

```
<!DOCTYPE html>
    <html>

        <head>
            <title> Title here </title>
        </head>

        <body>
            Web page content goes here.
        </body>

    </html>
```

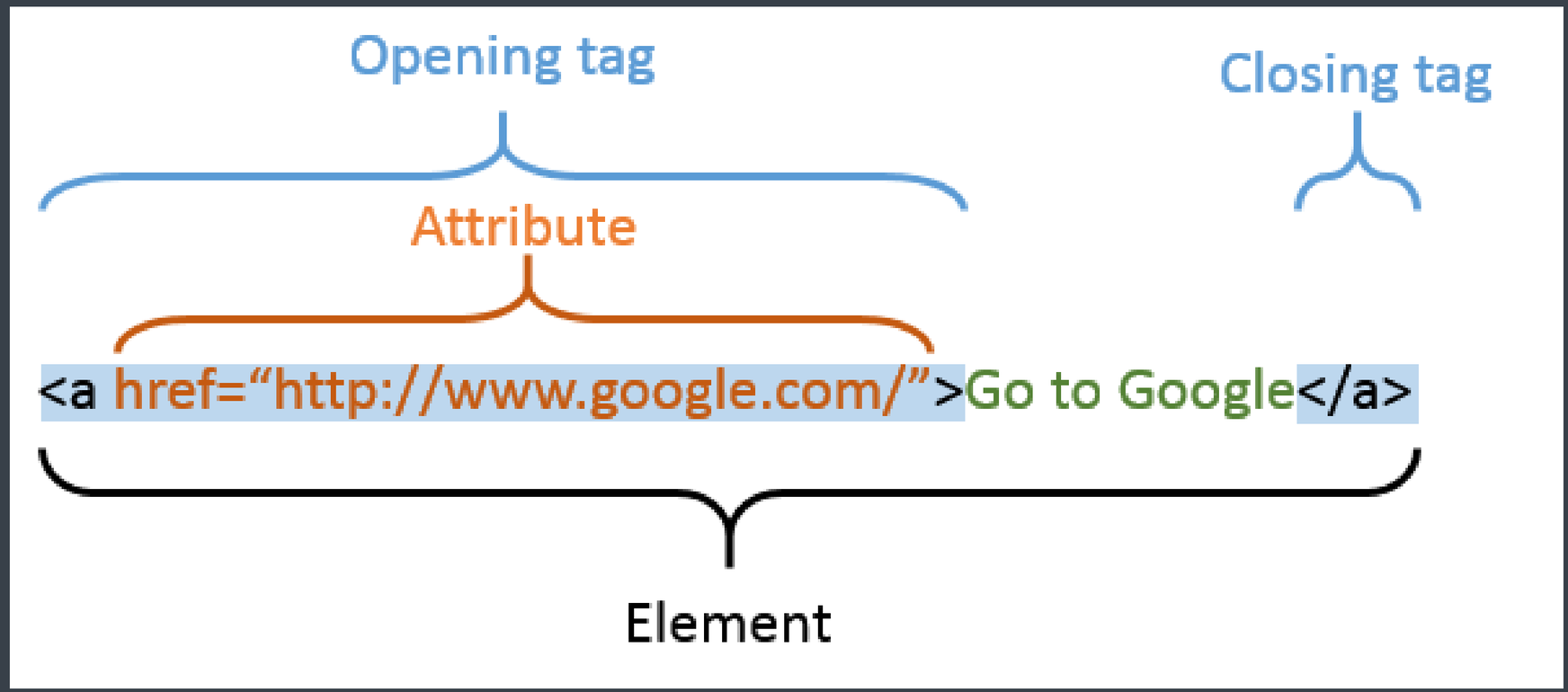
Setting up of environment

HTML Basics

Building Blocks of HTML

1. Tags
2. Attributes
3. Elements

Building Blocks of HTML



Tags

- Tags are like instructions for the browser and are enclosed in angle brackets, like <tagname>.
- Tags tell the browser how to display content on the web page.

Attributes

- HTML attributes are used to provide additional information about HTML elements.
- They are always specified in the opening tag of an element and typically consist of a name-value pair.

Elements

- Elements are made up of tags, with an opening tag <tagname> and a closing tag </tagname>.
- The opening tag marks the beginning of an element, and the closing tag marks the end.
- For example, is the opening tag for a paragraph, and is the closing tag.

HTML Document Structure

An HTML document's basic structure consists of 5 elements:

- <!DOCTYPE>
- <html>
- <head>
- <title>
- <body>

HTML Document Structure

```
<!DOCTYPE html>
<title>HTML tag</title>
<html>
    <body>
        <h1>Welcome</h1>
    </body>
</html>
```

HTML Tags & Elements

<!DOCTYPE html>: This declaration defines the document type and version (HTML5).

<html>: The root element that contains all other HTML elements on the page.

HTML Tags & Elements

<head>: Contains metadata about the document, such as the page title, character encoding, and linked stylesheets.

<title>: Sets the title of the web page, which appears in the browser's title bar or tab.

HTML Tags & Elements

<body>: Contains the visible content of the web page, including text, images, and more.

Text Formatting

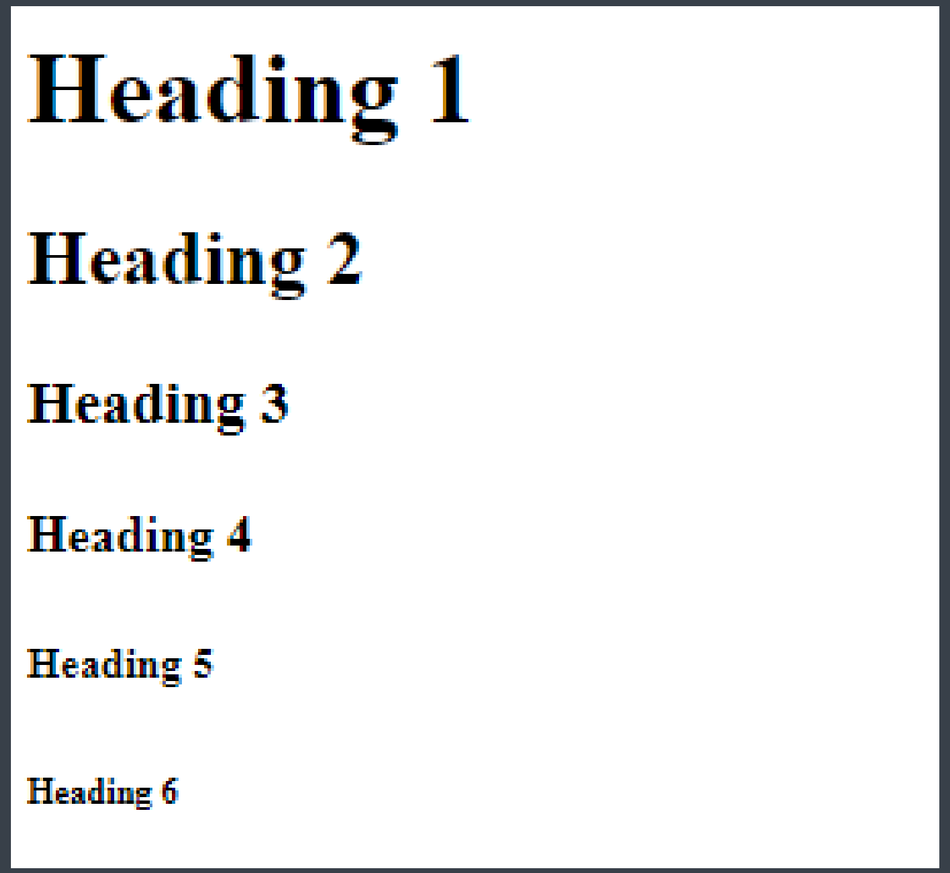
1. **Heading (<h1>)**
2. **Paragraph**
3. **Strong**
4. **Emphasize**
5. **Break**
6. **Horizontal line**

Text Formatting

Heading

```
<h1>This is a Heading 1</h1>
<h2>This is a Heading 2</h2>
<h3>This is a Heading 3</h3>
<h4>This is a Heading 4</h4>
<h5>This is a Heading 5</h5>
<h6>This is a Heading 6</h6>
```

Output



Text Formatting

Paragraph

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

Output:

This is a paragraph of text.

Text Formatting

Strong

```
<p>This is <strong>important</strong> text.</p>
```

Output:

This is **important** text.

The word "important" will be displayed in a bold font within the paragraph

Text Formatting

Emphasize

```
<p>This is <em>emphasized</em> text.</p>
```

Output:

This is emphasized text.

Text Formatting

Break

```
<p>This is the first line.<br>This is the second line.</p>
```

Output:

This is the first line.
This is the second line.

Text Formatting

Horizontal line

```
<p>Text above the line</p>
<hr>
<p>Text below the line</p>
```

Output:

```
Text above the line

----------------------
Text below the line
```

Creating Lists

1. **List (<li>)**
2. **Unordered (Bulleted) List (<ul>)**
3. **Ordered (Numbered) List (<ol>)**

Creating Lists

Using <ul> and

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

Output:

- Item 1
- Item 2
- Item 3

Creating Lists

Using <ol> and

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

Output:

1. First item
2. Second item
3. Third item

Hyperlinks

Hyperlinks using the <a> (anchor) tag
The href attribute specifies the URL (web address) that the link will point to.

Hyperlinks

anchor tag:

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

```
<a href="/about.html">About Us</a>
```

Hyperlinks

Output:

When a user clicks on the link, they will be directed to the "https://www.example.com" website.

HTML
Page Layout

- Semantic tags
- Non Semantic Tags
- Understanding Block & In Line Block
- class & id Attributes

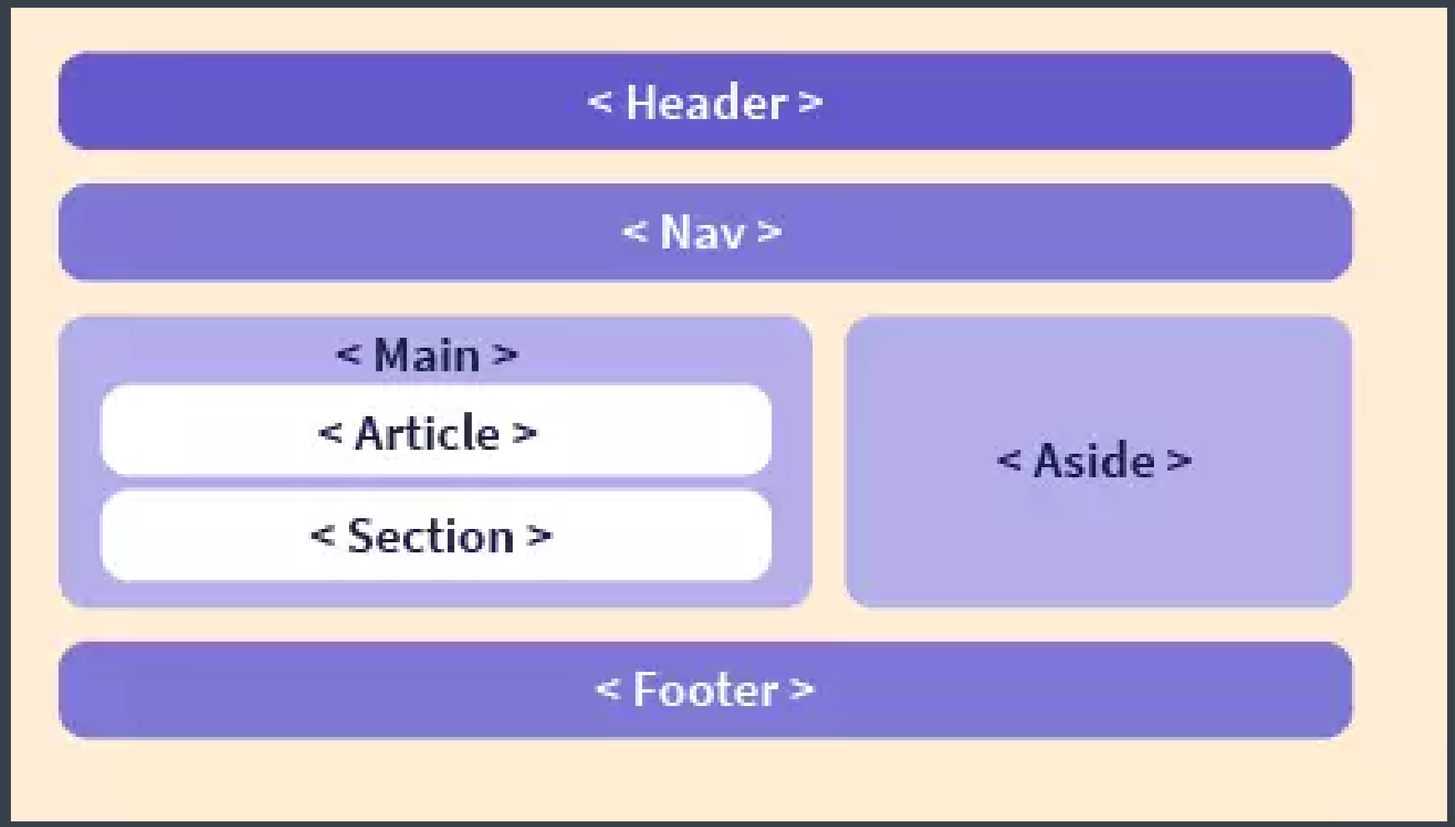
Semantic Tags

- <header>
- <main>
- <footer>

Non-Semantic Tags

-
-

Semantic Tags



Header

It often contains things like page titles, logos, navigation menus, and other top-of-the-page content.

Footer

It often contains information like copyright notices, contact details, and links to related pages or resources.

Main

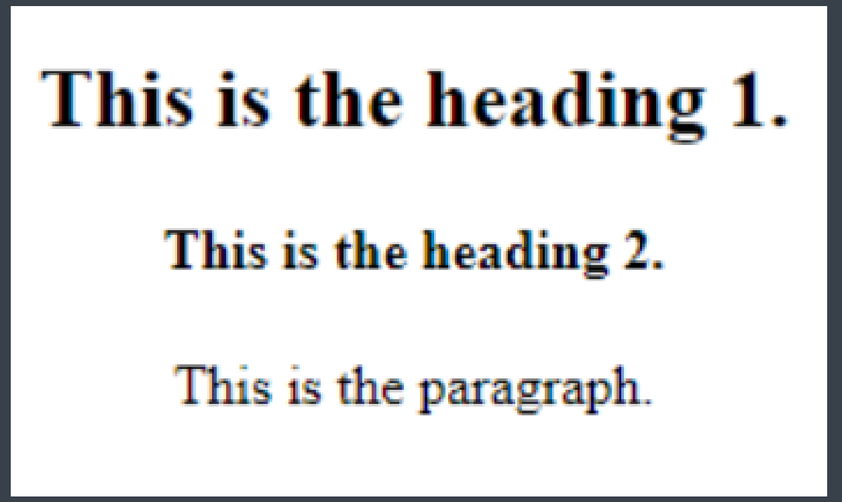
It helps screen readers and search engines understand the primary content of your page.

<header> Tag

- Using <nav> within the <header>

<header> Tag

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Header Tag</title>
    </head>
    <body>
        <article>
            <header>
                <h1>This is the heading 1.</h1>
                <h4>This is the heading 2.</h4>
                <p>This is the paragraph.</p>
            </header>
        </article>
    </body>
</html>
```



<main> Tag

- Section Tag: <section>
  For a section on your page

- Article Tag: <article>
  For an article on your page

- Aside Tag: <aside>
  For content aside main content(ads)

<main> Tag

```
<!DOCTYPE html>
<html>
<head>
    <title>My Website</title>
</head>
<body>
    <main>
        <h2>Welcome to My Website</h2>
        <p>This is the main content of the webpage.</p>
    </main>
</body>
</html>
```

<footer> Tag

<footer> Tag

```
<!DOCTYPE html>
<html>
<head>
    <title>My Website</title>
</head>
<body>
    <main>
        <h2>Welcome to My Website</h2>
        <p>This is the main content of the webpage.</p>
    </main>

    <footer>
        &copy; 2023 My Website
    </footer>
</body>
</html>
```

Non Semantic Tags

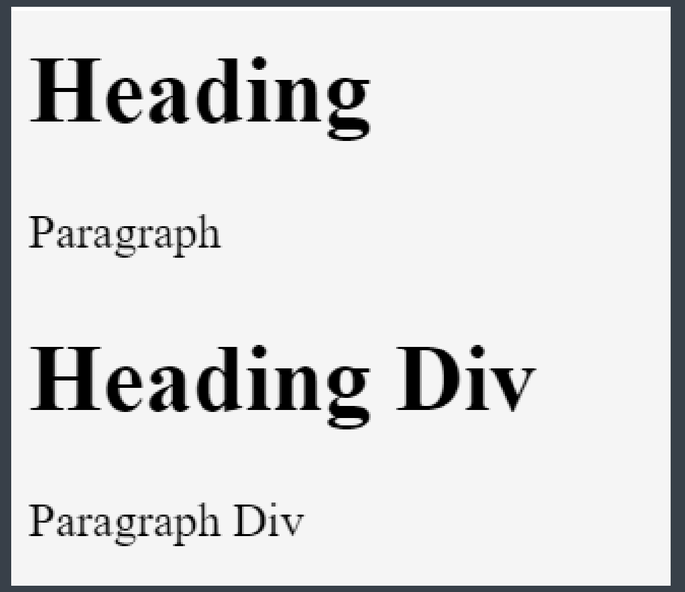
-
-

Tag

- Div is a container used for other HTML elements
- It is a Block Element
- Block Element: (takes full width)

Tag

```
<html>
<head>
</head>
    <body>
        <h1>Heading</h1>
        <p>Paragraph</p>
        <div>
            <h1>Heading Div</h1>
            <p>Paragraph Div</p>
        </div>
    </body>
</html>
```



Block Element Tags

- <address>
- <article>
- <aside>
- <blockquote>
- <canvas>
- <dd>
-
- <dl>
- <dt>
- <fieldset>
- <fiecaption>
- <figure>
- <footer>
- <form>
- <h1> to <h6>
- <header>
- <hr>
- <i>
- <main>
- <nav>
-
- <p>
- <pre>
- <section>
- <table>
- <tfoot>
- <video>

Tag

- Span is also a container used for other HTML elements
- It is a Inline Element
- Inline Element: (takes full width)

Tag

```
<body>
    <div>
        <h1> what is span tag</h1>
        <span>This is a span tag</span>
    </div>
</body>
```

Inline Element Tags

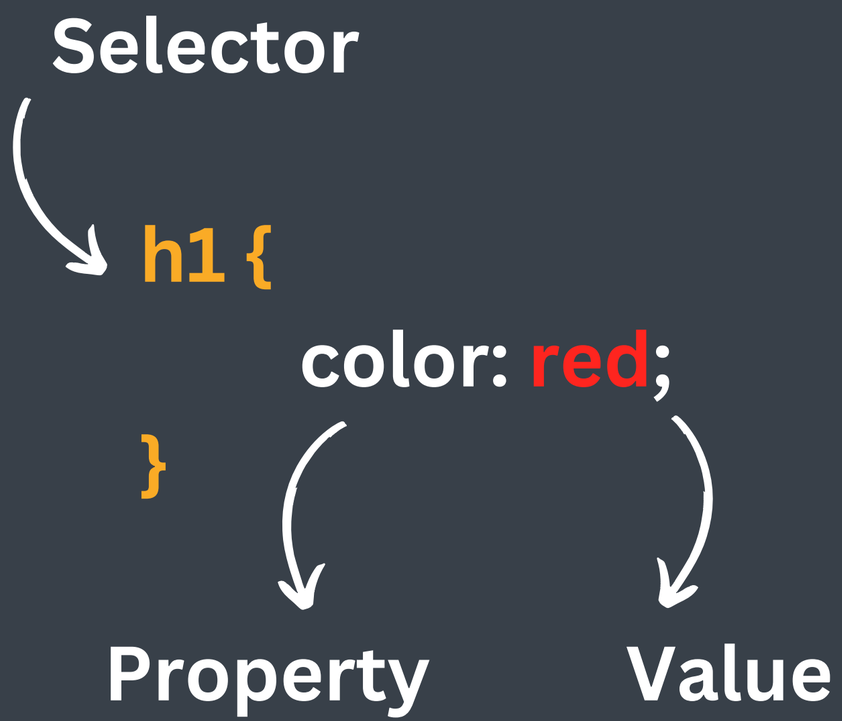
- <a>
- <abbr>
- <acronym>
- <b>
- <bdo>
- <big>
-
- <button>
- <cite>
- <object>
- <code>
- <dfn>
-
- <i>
- <img>
- <input>
- <kbd>
- <label>
- <map>
- <var>
- <tt>
- <output>
- <a>
- <samp>
- <script>
- <select>
- <small>
-
-
- <sub>
- <sup>
- <textarea>
- <time>

Introduction to CSS

Basics of CSS

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

Basic Syntax:

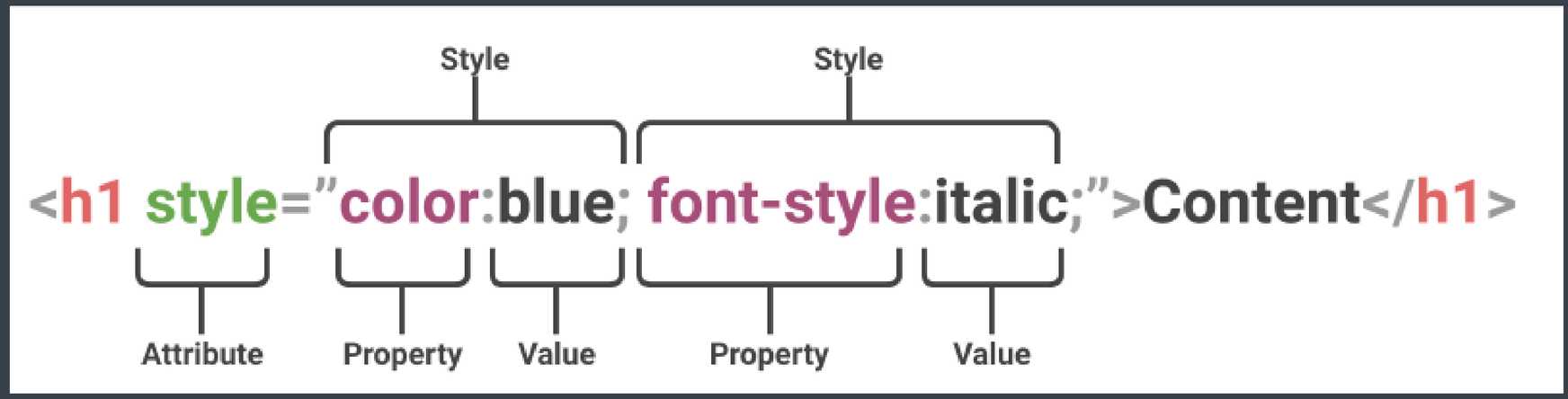


Types of CSS

- Inline CSS
- Internal or Embedded CSS
- External CSS

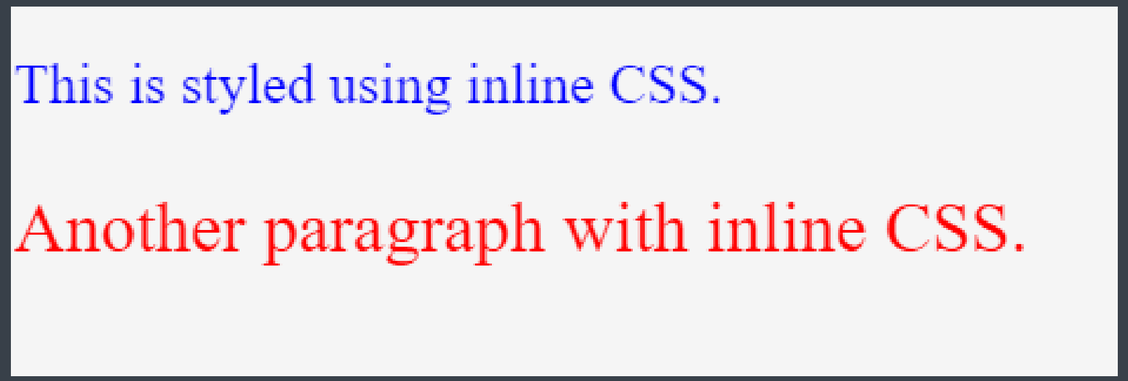
Inline CSS

Inline CSS is applied directly to individual HTML elements using the style attribute.



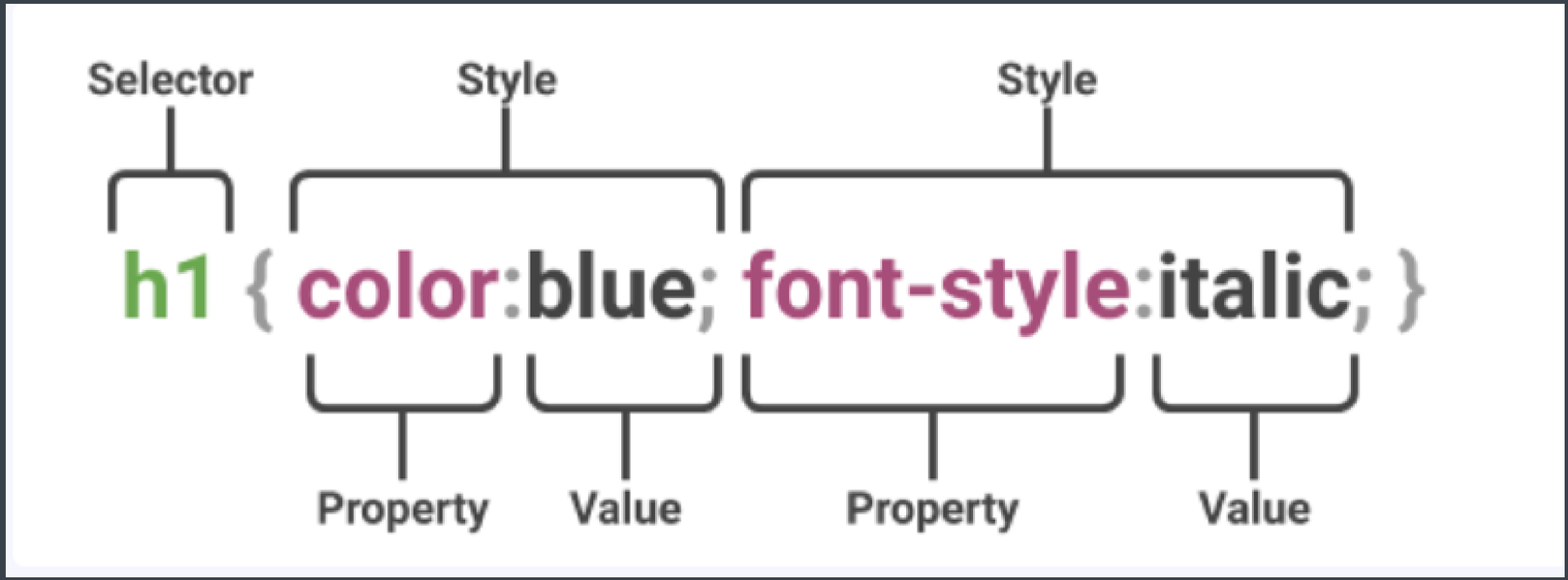
Inline CSS

```
<!DOCTYPE html>
<html>
<head>
    <title>Inline CSS Example</title>
</head>
<body>
    <p style="color: blue; font-size: 16px;">This is styled using inline CSS.</p>
    <p style="color: red; font-size: 20px;">Another paragraph with inline CSS.</p>
</body>
</html>
```



Internal or Embedded CSS

Embedded CSS is placed within the <style> element in the <head> section of an HTML document.



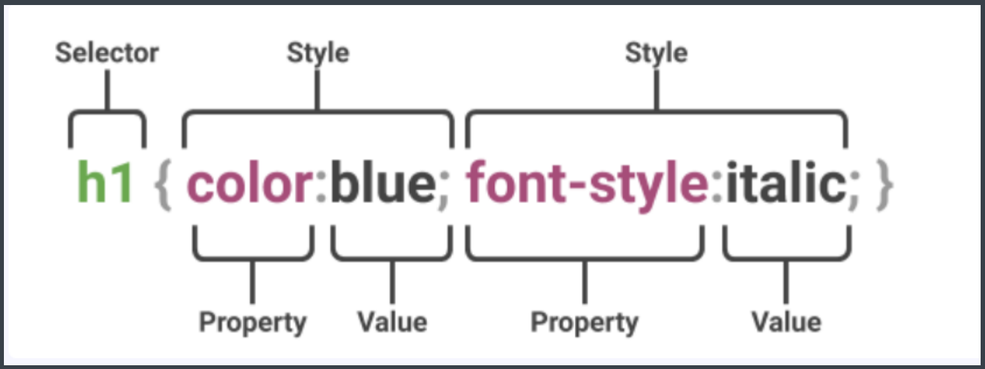
Internal or Embedded CSS

```
<html>
<head>
    <style>
        p {
            color: green;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <p>This is styled using embedded CSS.</p>
</body>
</html>
```

External CSS

External CSS is defined in separate CSS files with a .css extension.These CSS files are linked to HTML documents using the <link> element in the <head> section.

```
<link rel="stylesheet" href="path/file.css">
```



External CSS

```
<html>
<head>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <p>This is styled using an external CSS file.</p>
</body>
</html>
```

```
p {
    color: purple;
    font-size: 20px;
}
```

This is styled using an external CSS file.

Linking CSS to HTML

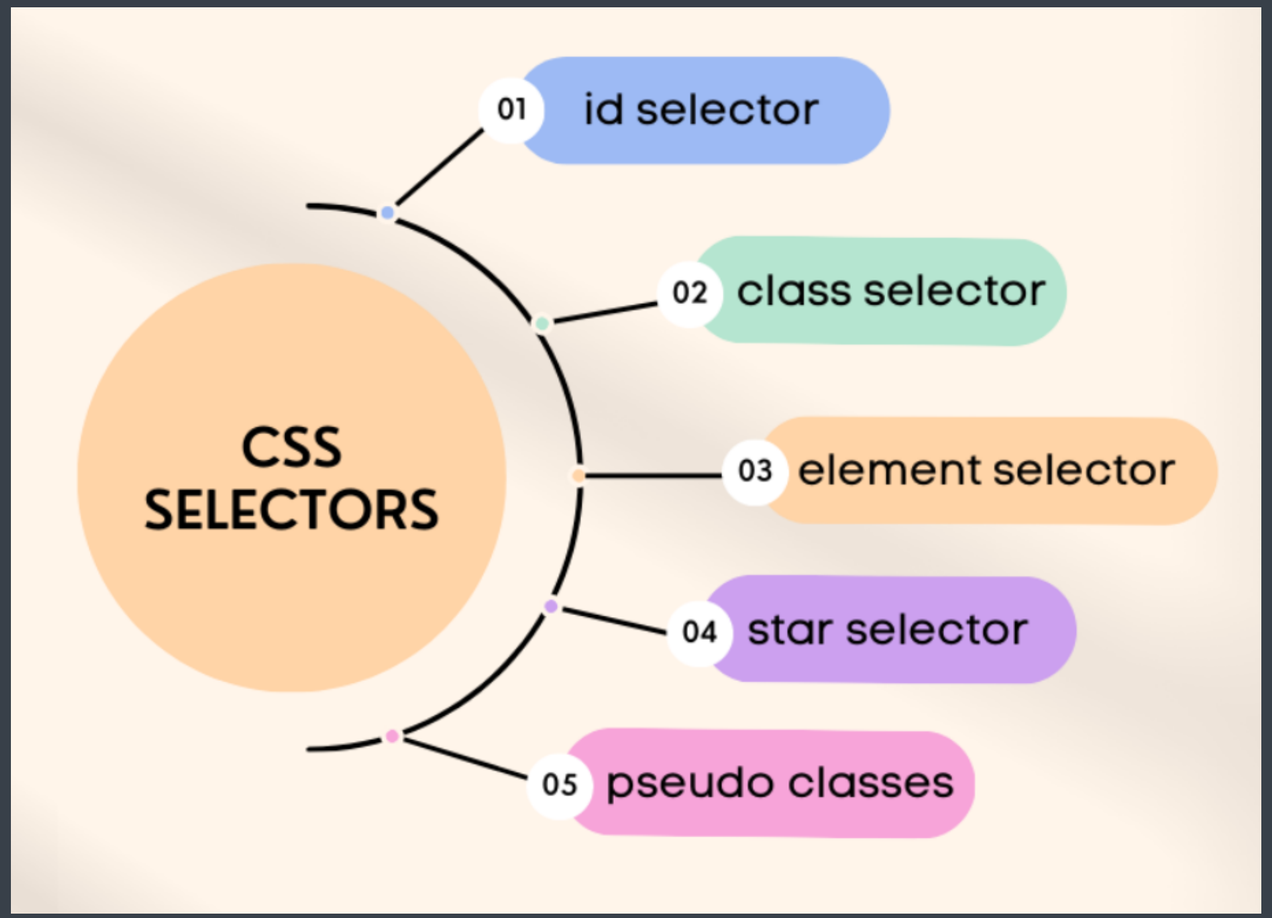
1. Linking External CSS (<link>):

Create a separate CSS file (e.g., "styles.css") and define your styles within it.

Steps:

1. Create a CSS File
2. Create an HTML Document
3. Link the CSS File
4. Place HTML Content
5. Save and Open

CSS Selectors



CSS Selectors

- CSS selectors are patterns used to select and style HTML elements on a web page.
- They determine which elements in the HTML document will be affected by the defined styles.

1. **Element**
2. **Class**
3. **ID**
4. **Star Selector**

Element Selector

An element selector in CSS is used to target and style all HTML elements of a specific type. It selects elements based on their HTML tag name.

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

Class Selector

A class selector in CSS is used to target and style HTML elements that have a specific class attribute. It allows you to apply styles to one or more elements on a page that share the same class.

```
.highlight {
    background-color: yellow;
}
```

ID Selector

An ID selector in CSS is used to target and style a single HTML element that has a specific ID attribute. It allows you to apply styles to a unique element on a page.

```
#header {
    font-size: 24px;
}
```

Star(*) Selector

An element selector in CSS is used to target and style all HTML elements of a specific type. It selects elements based on their HTML tag name.

Styling Colors

You can specify colors using various formats

- Color names
- Hexadecimal values
- RGB values
- HSL values.

Styling Text Color

The color property allows you to set the text color of an HTML element.

```
p {
    color: red;
}
h1 {
    color: #336699;
}
a {
    color: rgb(255, 0, 0);
}
span {
    color: hsl(120, 100%, 50%);
}
```

Styling Background Color

The background-color property allows you to set the background color of an HTML element. Like the color property, you can specify background colors

```
div {
    background-color: lightgray;
}
section {
    background-color: #f0f0f0;
}
header {
    background-color: rgb(255, 204, 102);
}
footer {
    background-color: hsl(240, 60%, 70%);
}
```

Styling Text, Colors, and Backgrounds

Text Properties

- Text-align
- Text-decoration
- Font-weight
- Font-family
- Line-height
- Text-transform

Text-align

- text-align : left / right / center

```
.center-text {
    text-align: center;
}
```

```
.center-text {
    text-align: right;
}
```

```
.center-text {
    text-align: left;
}
```

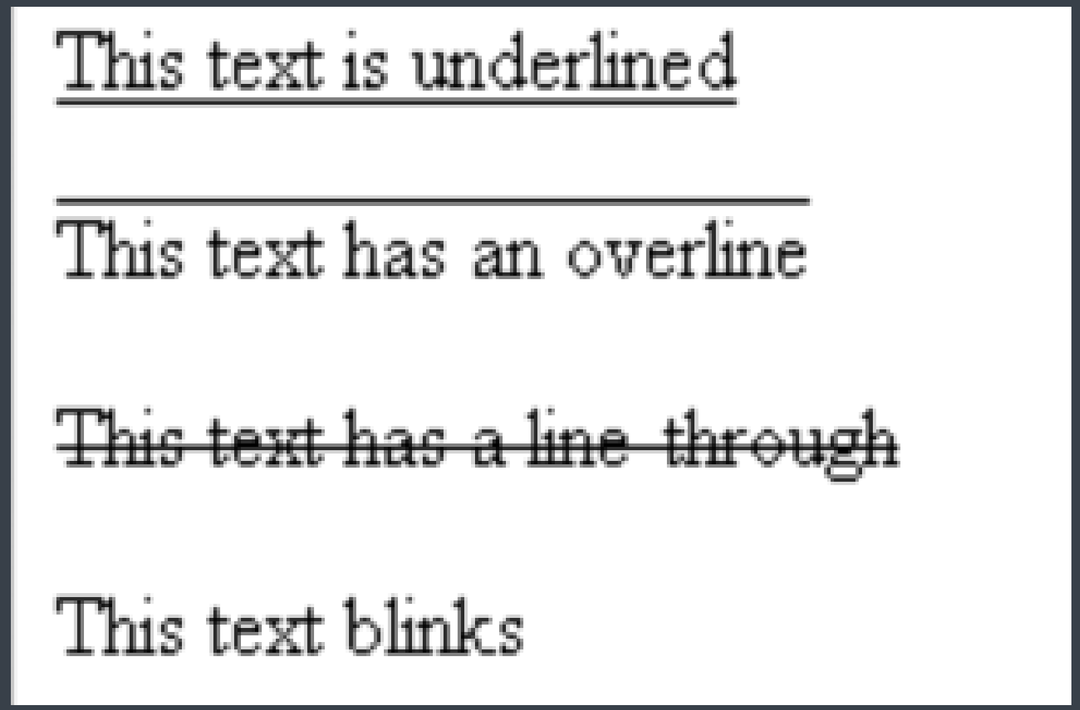
text-align: center;

text-align: right;

text-align: left;

Text-decoration

- text-decoration : underline / overline / line-through




```
a:hover {
    text-decoration: underline;
}
```

Font-weight

- font-weight : normal / bold / bolder / lighter
- font-weight : 100-900

normal

bold

bolder.

lighter.

```
strong {
    font-weight: bold;
}
```

Font-family

- font-family : arial
- font-family : arial, roboto

serif
sans-serif
cursive
fantasy
monospace

```
body {
    font-family: Arial, sans-serif;
}
```

Line-height

- line-height : 2px
- line-height : 3
- line-height : normal

```
.spaced-text {
    line-height: 1.5;
}
```

Line-height

Line Height

Here, we are setting the line height with a number. This number will be multiplied with the current font-size.

Line-Height: 0.5

This is a paragraph with the recommended line-height.
It is set to 0.5.

Line-Height: 1.5

This is a paragraph with the recommended line-height.
It is set to 1.5.

Line-Height: 2.5

This is a paragraph with the recommended line-height.

It is set to 2.5.

Text-transform

- Text-tranform : uppercase / lowercase / capitalize / none

```
.uppercase-text {
    text-transform: uppercase;
}
```

HERE IS SOME BASIC TEXT.

here is some basic text.

Here Is Some Basic Text.

Colors

Used to set the color of foreground

color: red;
color: pink;
color: blue;
color: green;

Type of Color Systems

- Color Names
- RGB Notation
- RGBA Color Notation
- Hexadecimal Color Notation
- HSL notation
- Gradient Colors

Type of Color Systems

```
color: rgb(255, 0, 0); /* RGB notation for red */

color: rgba(255, 0, 0, 0.5); /* Red with 50% transparency */

color: #FF5733; /* Hexadecimal notation for orange */

color: hsl(0, 100%, 50%); /* HSL notation for pure red */

background: linear-gradient(to right, #FF5733, #00AABB);
background: radial-gradient(circle, #FF5733, #00AABB);
```

Background Properties

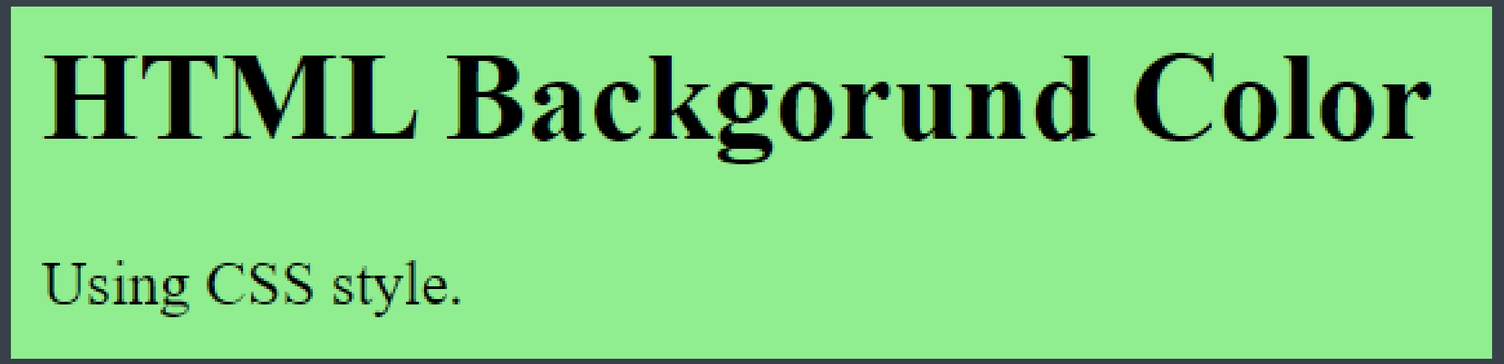
- Background color property
- Background Image Property
- Background- Repeat Property
- Background size property
- Background position property
- Background shorthand property

Background color property

Used to set the color of background

- background-color: red;
- background-color: pink;
- background-color: blue;

```
body {
    background-color: lightgreen;
}
```



Background Image Property

Used to set an image as background

- background-image: url("image.jpeg");

```
.header {
    background-image: url('girl.jpg');
}
```

Background- Repeat Property

The background-repeat property repeats an image both horizontally and vertically.

- background-repeat: repeat-x/repeat-y/no-repeat;

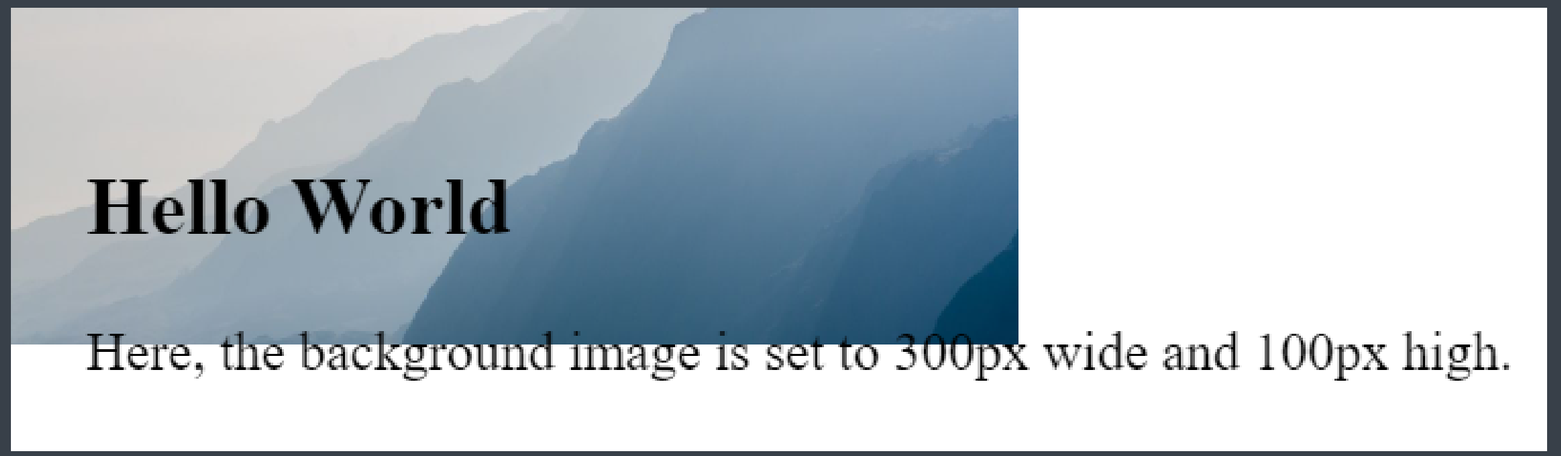
Background size property

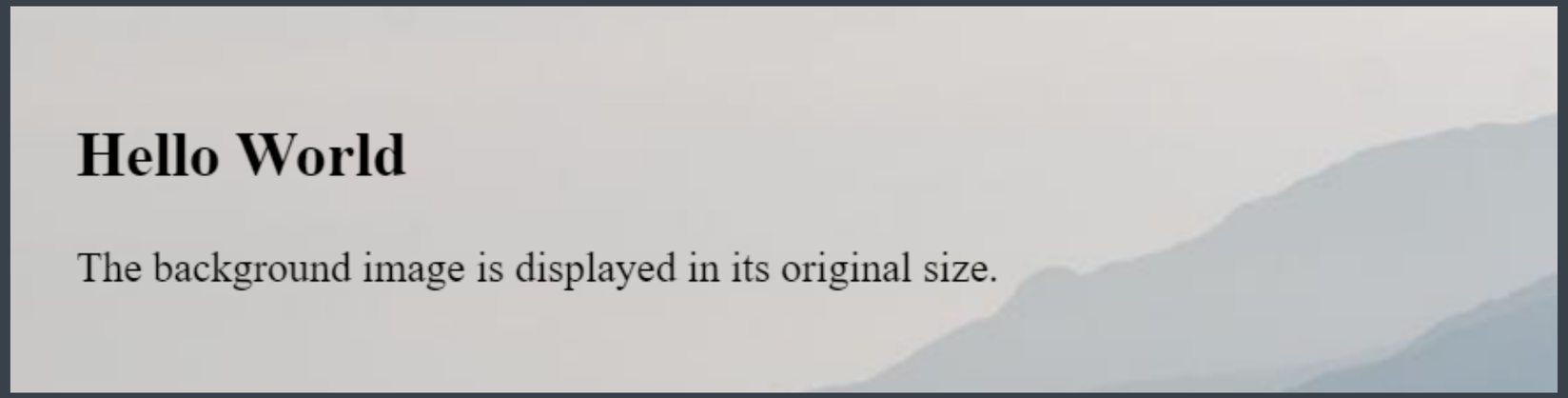
The background-size property specifies the size of the background images.

- background-size: cover / contain / auto;

```
.example {
    background: url(mountain.jpg);
    background-repeat: no-repeat;
    background-size: 300px 100px;
}
```

Background size property





Background position property

- background-position: left/top/center/bottom/right;

Background shorthand property

```
background: [background-color] [background-image] [background-repeat]

                                [background-attachment] [background-position];
```

- background: #00AABB url('background-image.jpg') no-repeat center center;

Units, Position & Display properties

Units in CSS

- CSS has several different units for expressing a length.
- Many CSS properties take "length" values, such as width, margin, padding, font-size, etc.
- Length is a number followed by a length unit, such as 10px, 2em, etc.

Units in CSS

ABSOLUTE

Pixels (px)

Inches (in)

Centimeters (cm)

Millimeters (mm)

Points (pt)

Picas (pc)

RELATIVE

Percentages (%)

Font sizes (em, rem)

Character sizes (ex,ch)

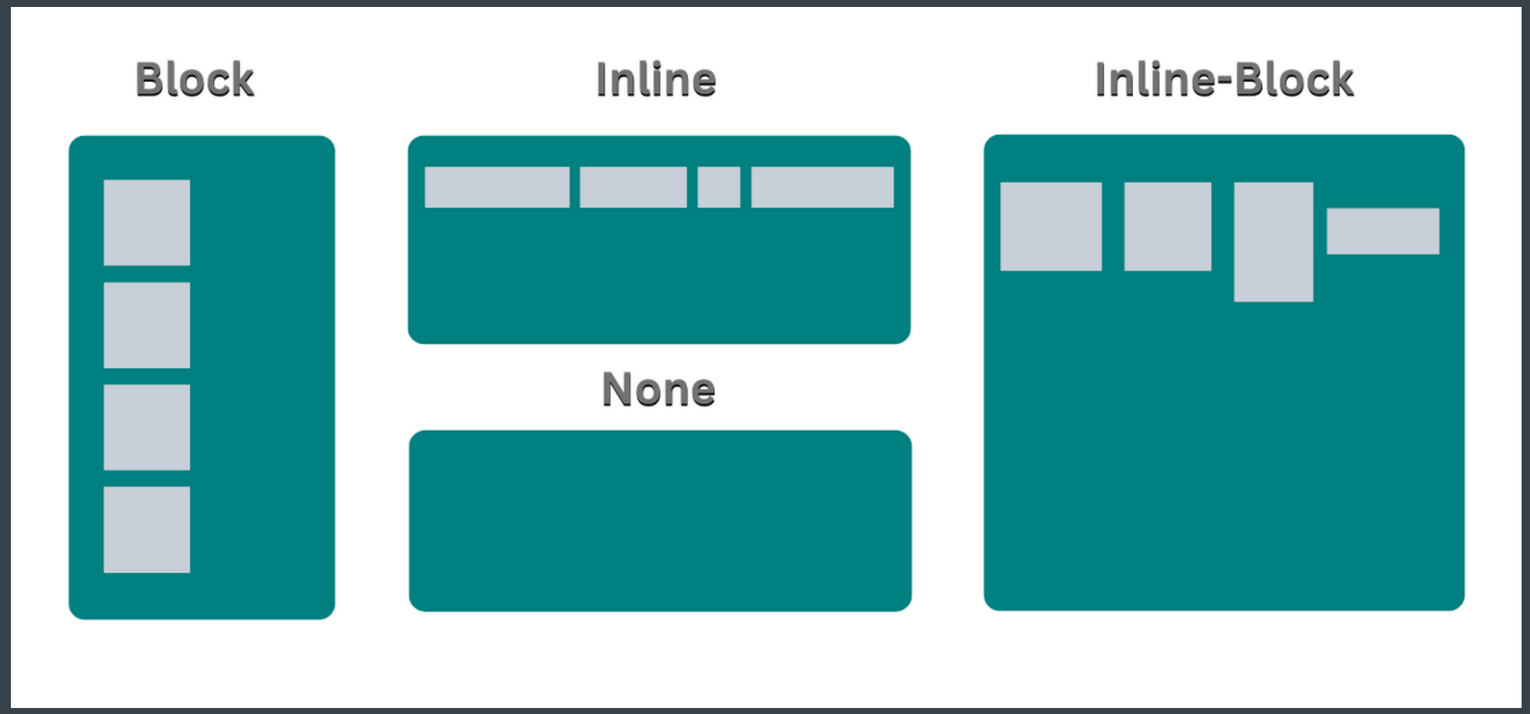
Viewport dimensions (vh, vw)

Viewport max (vmax)

Viewport min (vmin)

CSS Display Property

- The display property specifies the display behavior (the type of rendering box) of an element.



CSS Display Property

display: inline / block / inline-block / none

- inline - Takes only the space required by the element. (no margin/ padding)
- block - Takes full space available in width.
- inline-block - Similar to inline but we can set margin & padding.
- none - To remove element from document flow.

Visibility

- The visibility property specifies whether or not an element is visible.

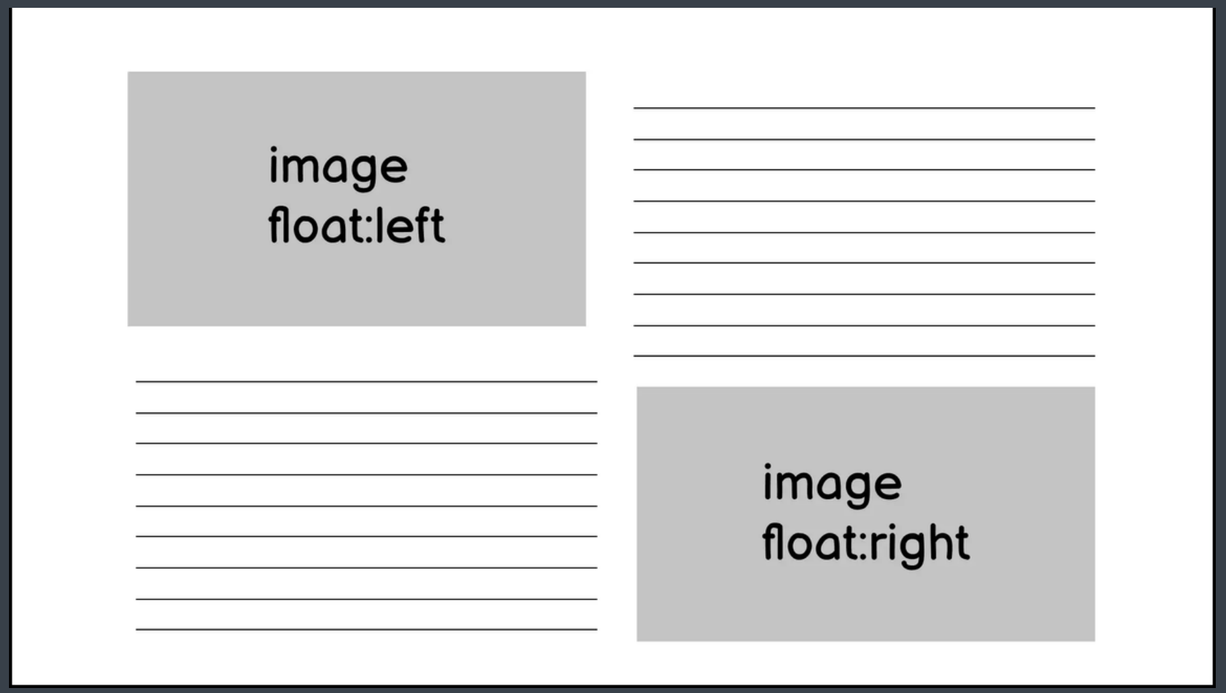
Visibility

visibility: visible | hidden | collapse;

Note : When visibility is set to none, space for the element is reserved. But for display set to none, no space is reserved or blocked for the element.

CSS Float Property

- The float property specifies whether an element should float to the left, right, or not at all.

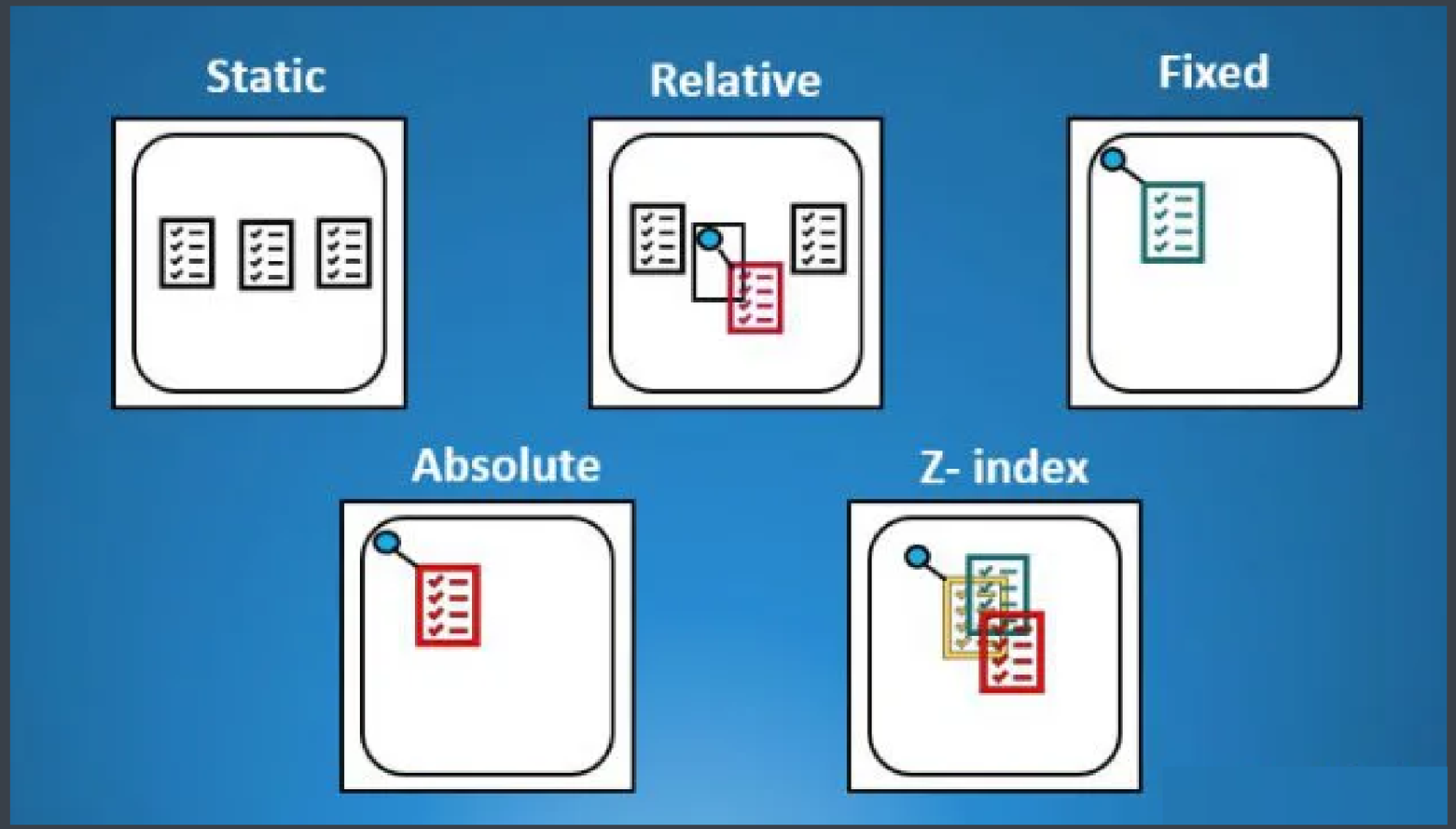


CSS Position Property

- The position CSS property sets how an element is positioned in a document.

position : static / relative / absolute / fixed

CSS Position Property



Position

- static - default position (The top, right, bottom, left, and z-index properties have no effect)
- relative - element is relative to itself. (The top, right, bottom, left, and z-index will work)
- absolute - positioned relative to its closest positioned ancestor. (removed from the flow)
- fixed - positioned relative to browser. (removed from flow)
- sticky - positioned based on user's scroll position

z-index

It decides the stack level of elements

- z-index : auto (0)
- z-index : 1 / 2 / ... z-index : -1 / -2 / ...

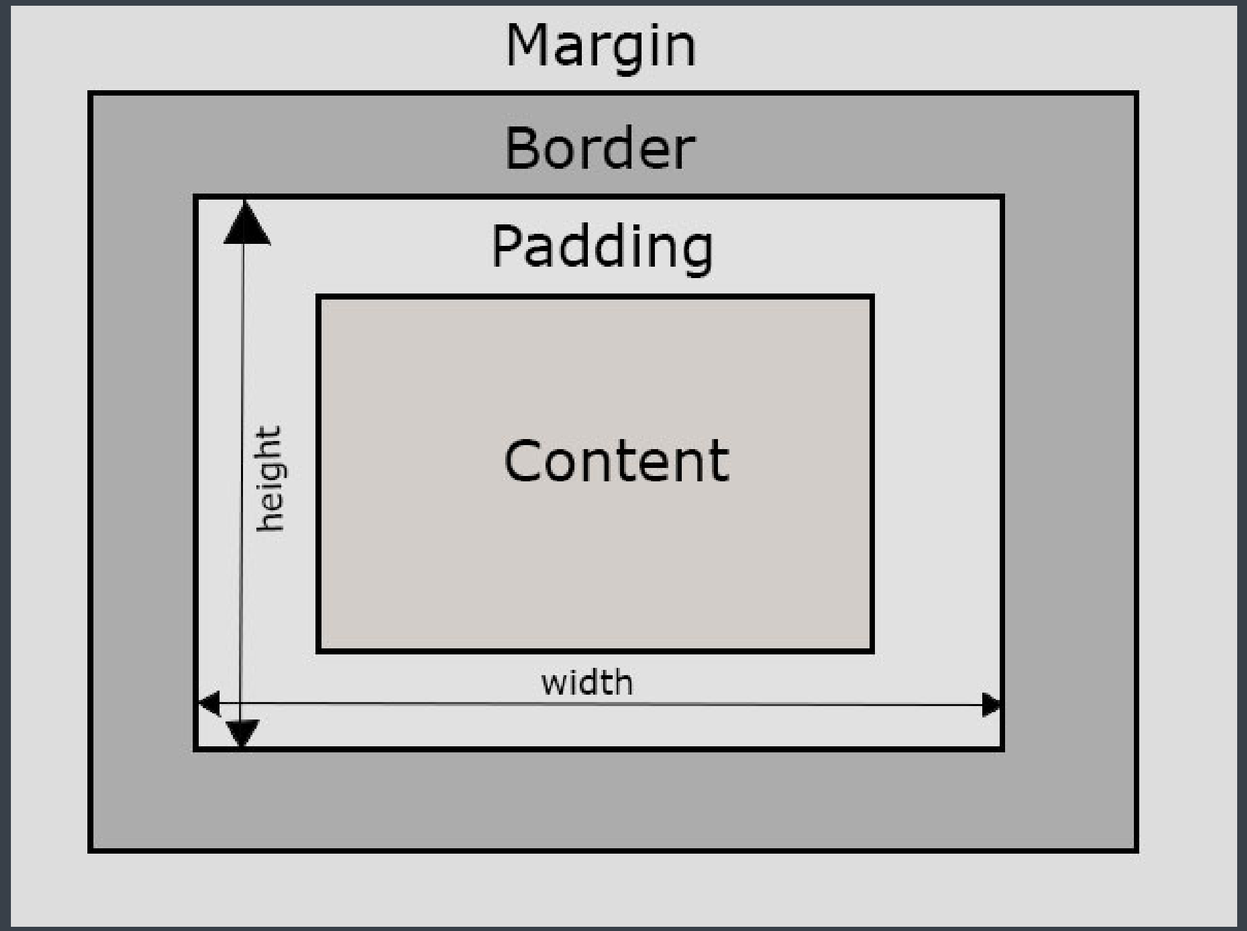
NOTE: Overlapping elements with a larger z-index cover those with a smaller one.

CSS Box Model and Layout

CSS Box Model

- Height
- Width
- Border
- Padding
- Margin
- Content

CSS Box Model



CSS Box Model

input

```
div {
  height: 100px;
  width: 320px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;
}
```

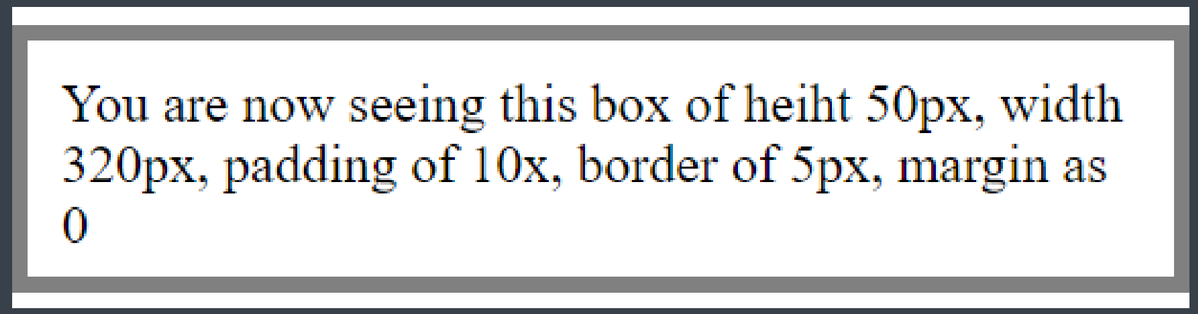
output

You are now seeing this box of heiht 100px, width 320px, padding of 10x, border of 5px, margin as 0

Height

- By default, it sets the content area height of the element

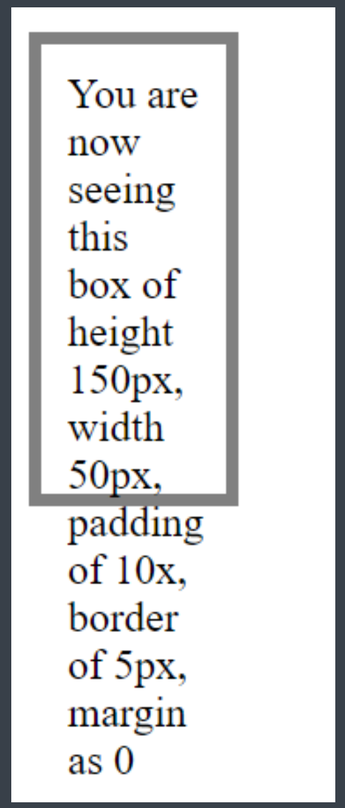
```
div {
        height: 50px;
 }
```



Width

- By default, it sets the content area width of the element

```
div {
        width: 50px;
 }
```



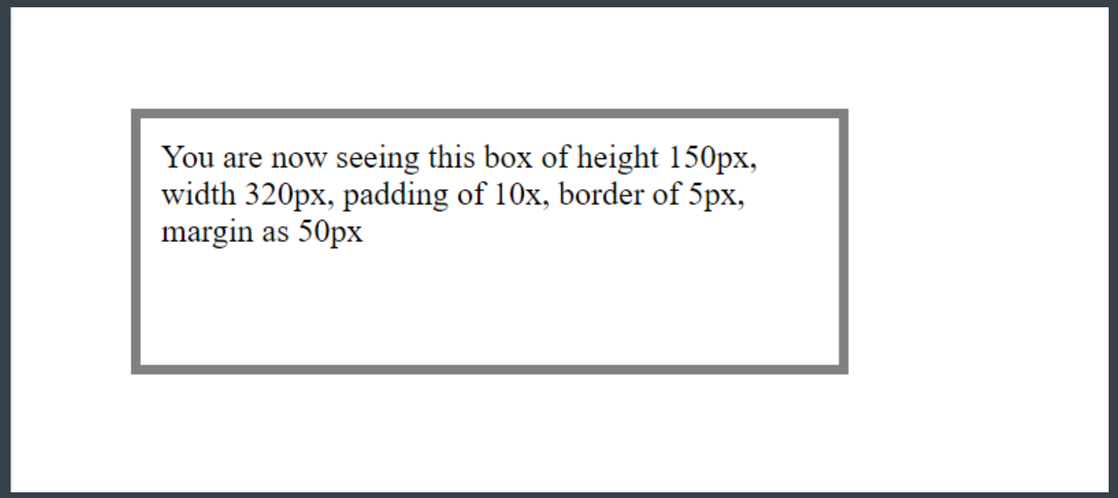
Margin

- margin-right
- margin-left
- margin-top
- margin-bottom

Margin shorthand

- margin: 50px;
- Margin margin: 1px 2px 3px 4px;

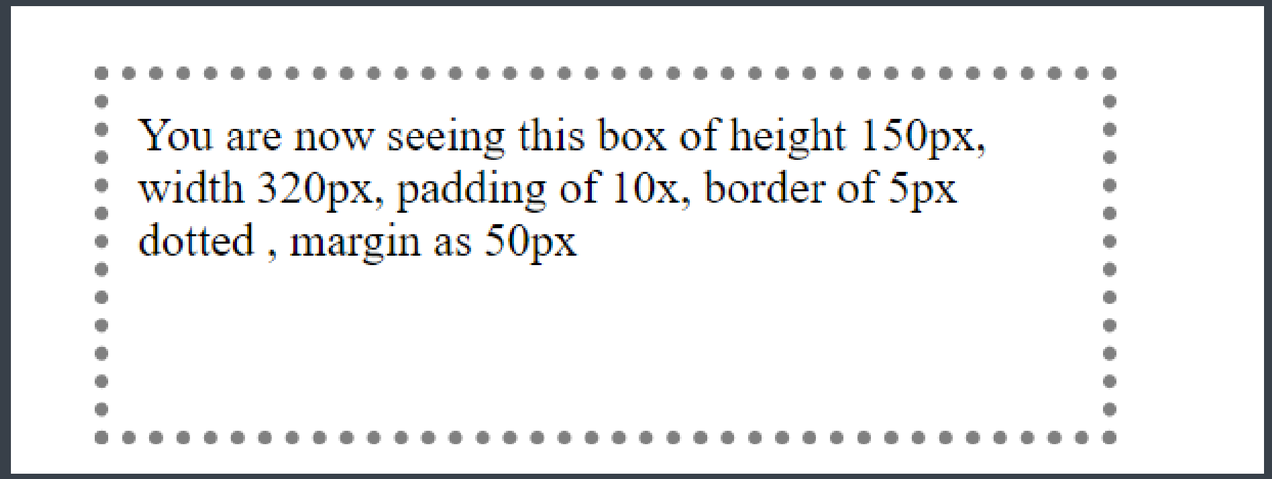
top | right | bottom | left -> clockwise



Border

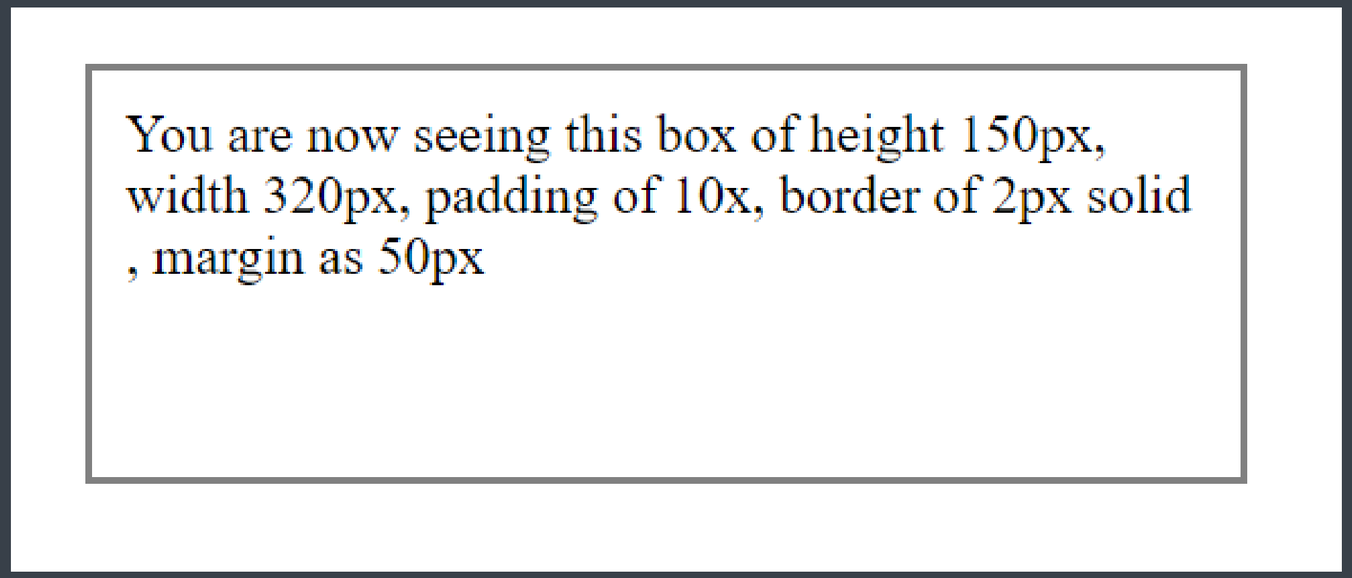
Used to set an element's border

- border-width : 2px;
- border-style : solid / dotted / dashed;
- border-color : black;



Border Shorthand

- border : 2px solid black;



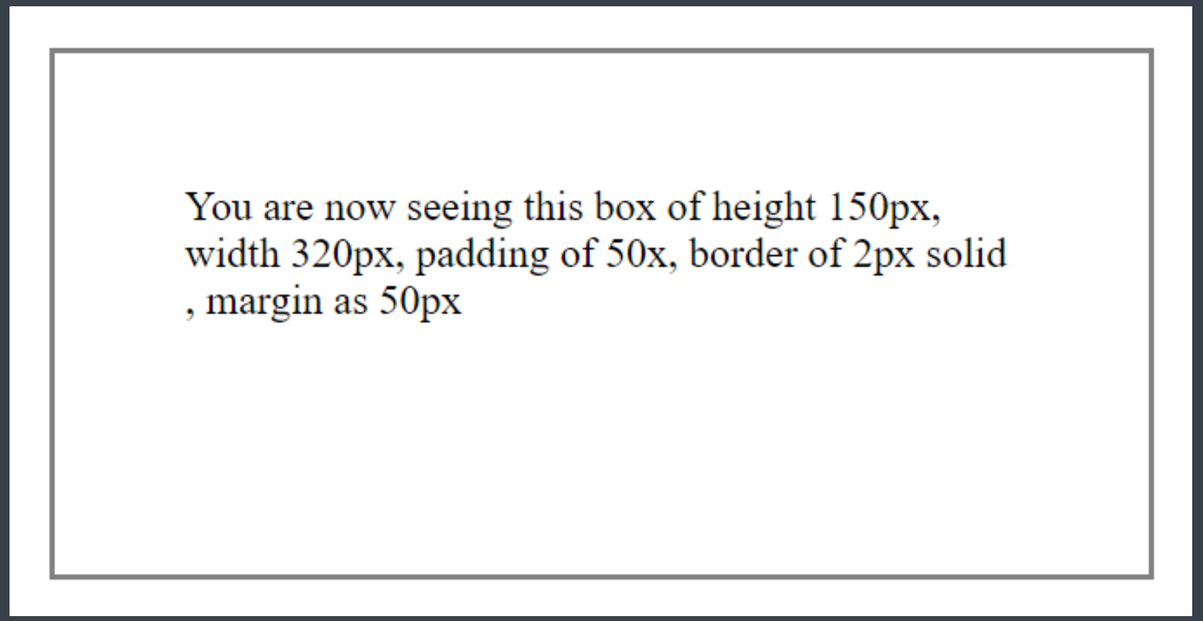
Padding

- padding-left
- padding-right
- padding-top
- padding-bottom

Padding Shorthand

- padding: 50px;
- Padding padding: 1px 2px 3px 4px;

top | right | bottom | left -> clockwise



CSS FlexBox & Grid Layout

CSS Flexbox

- CSS Flexbox
- The Flex Model
- Flexbox Properties (for Container)
- Flexbox Properties (for Item)

CSS Flexbox

It provides a way to distribute space and align content in a container, even when their sizes are unknown or dynamic.

Key properties include display:

- flex-direction
- justify-content
- align-items
- flex-grow

The Flex Model

Flexbox Properties (for Container)

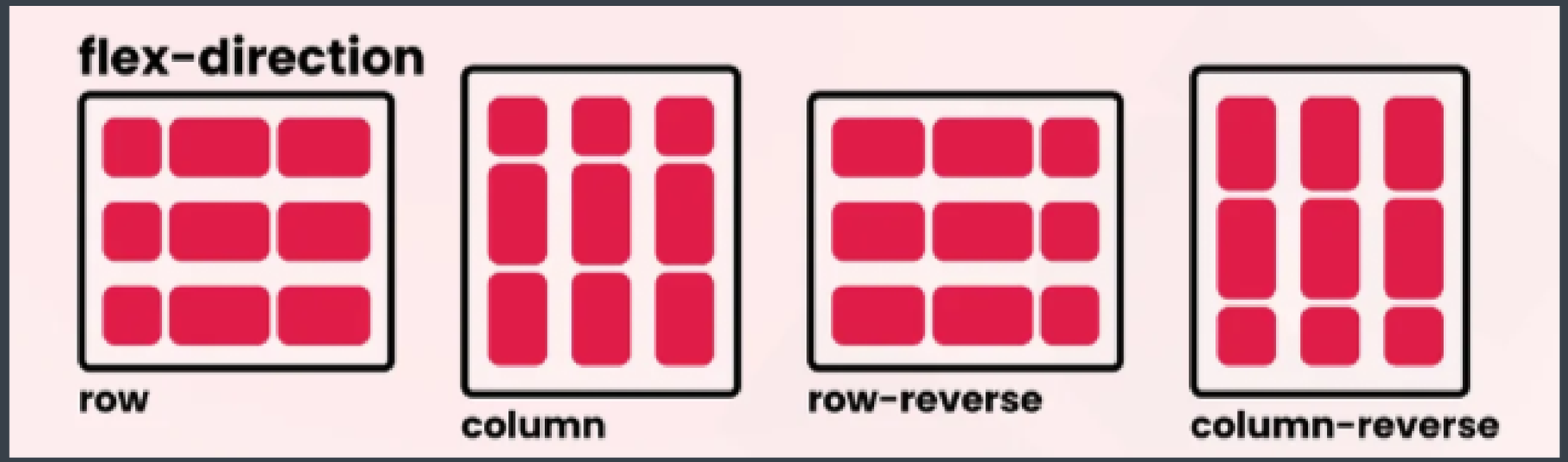
- flex-direction.
- flex-wrap.
- flex-flow.
- justify-content.
- align-items.
- align-content.

Flexbox direction

It sets how flex items are placed in the flex container, along which axis and direction.

- flex-direction : row; (default)
- flex-direction : row-reverse;
- flex-direction : column;
- flex-direction : column-reverse;

Flexbox direction



Flexbox wrap

The flex-wrap property specifies whether the flex items should wrap or not.

- flex-wrap : nowrap / wrap / wrap-reverse

Flexbox flex-flow.

The flex-flow property is a shorthand property for setting both the flex-direction and flex-wrap properties.

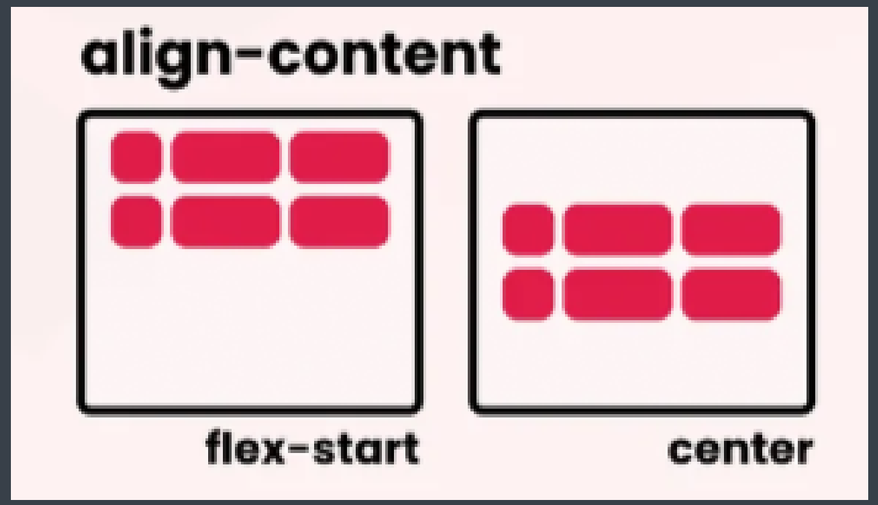
- flex-flow : row wrap;

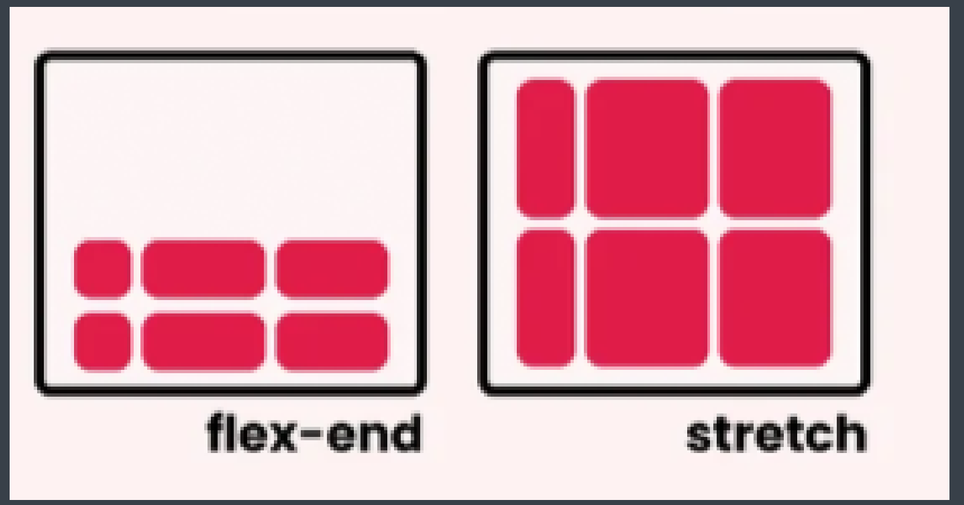
Flexbox Justify content

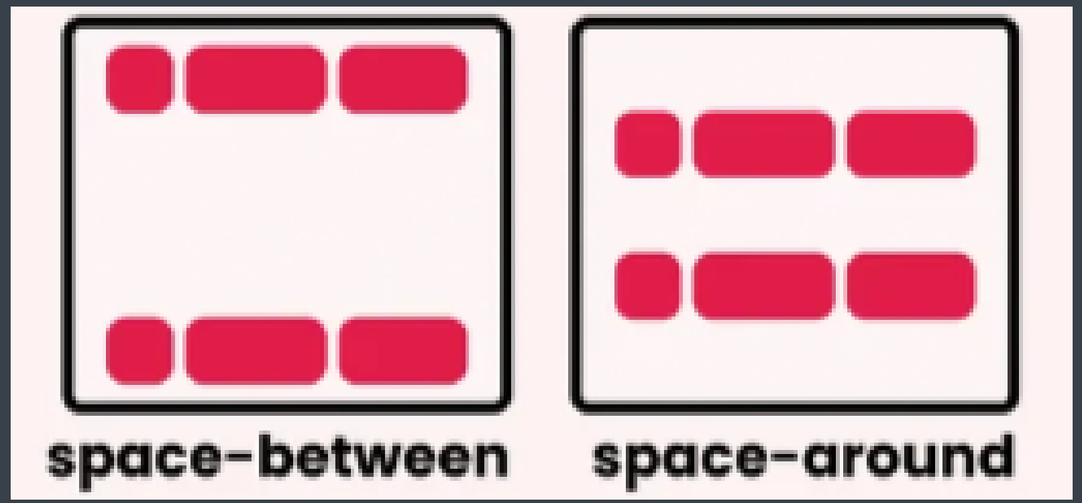
- justify-content : alignment along the main axis.

  flex-start / flex-end / centre / space-evenly

Flexbox Justify content

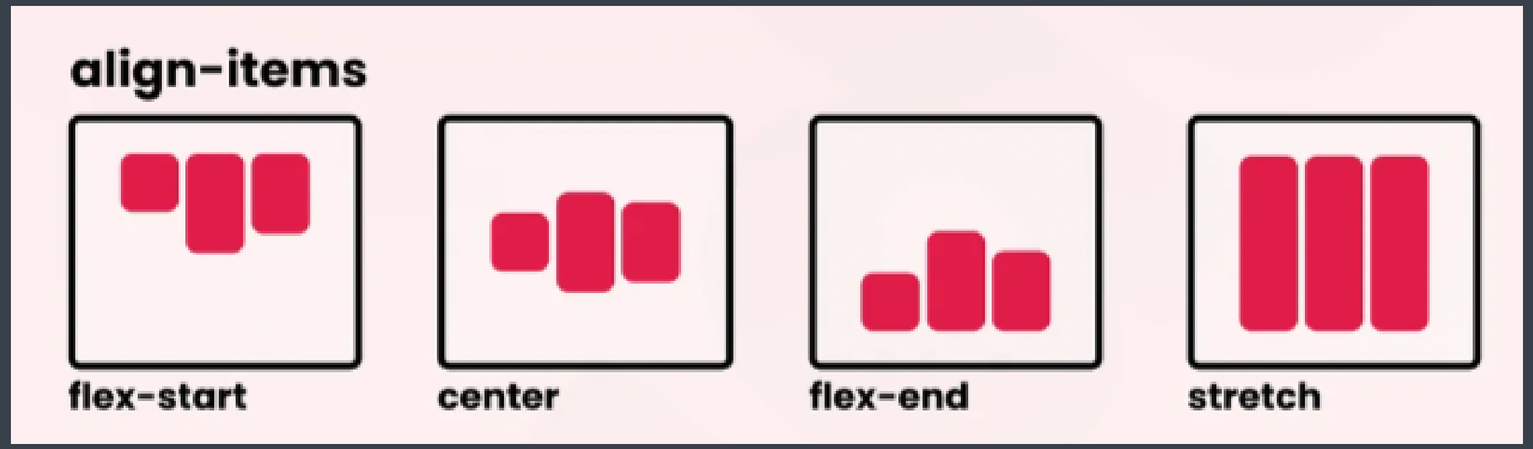






Flexbox align items

- align-items : alignment along the cross axis.



Flexbox align-content

The align-content property is used to align the flex lines.

- align-content : alignment of space between & around the content along cross-axis

Flexbox Properties(for Item)

These properties determine how each item behaves within the flex container.

- align-self : alignment of individual along the cross axis.
- flex-grow : how much a flex item will grow relative to the rest of the flex items if space is available
- flex-shrink : how much a flex item will shrink relative to the rest of the flex items if space is available

Grid Layout

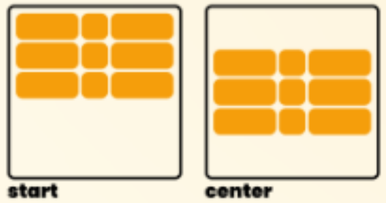
- CSS Grid Layout is another layout model that allows you to create two-dimensional grid-based layouts.
- It's useful for creating more complex grid structures compared to Flexbox.

Grid Layout

- grid-template-columns
- grid-template-rows
- grid-auto-columns
- grid-auto-rows
- justify-content
- justify-items
- align-content
- align-items

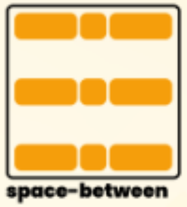
CSS Grid Layout

align-content


start

center


end

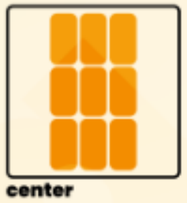

space-between


space-around


stretch

justify-content


start


center


end

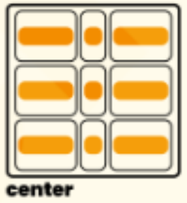

space-between


space-around


stretch

align-items

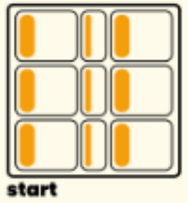

start

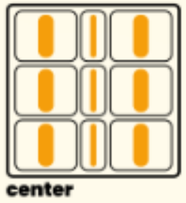

center


end


stretch

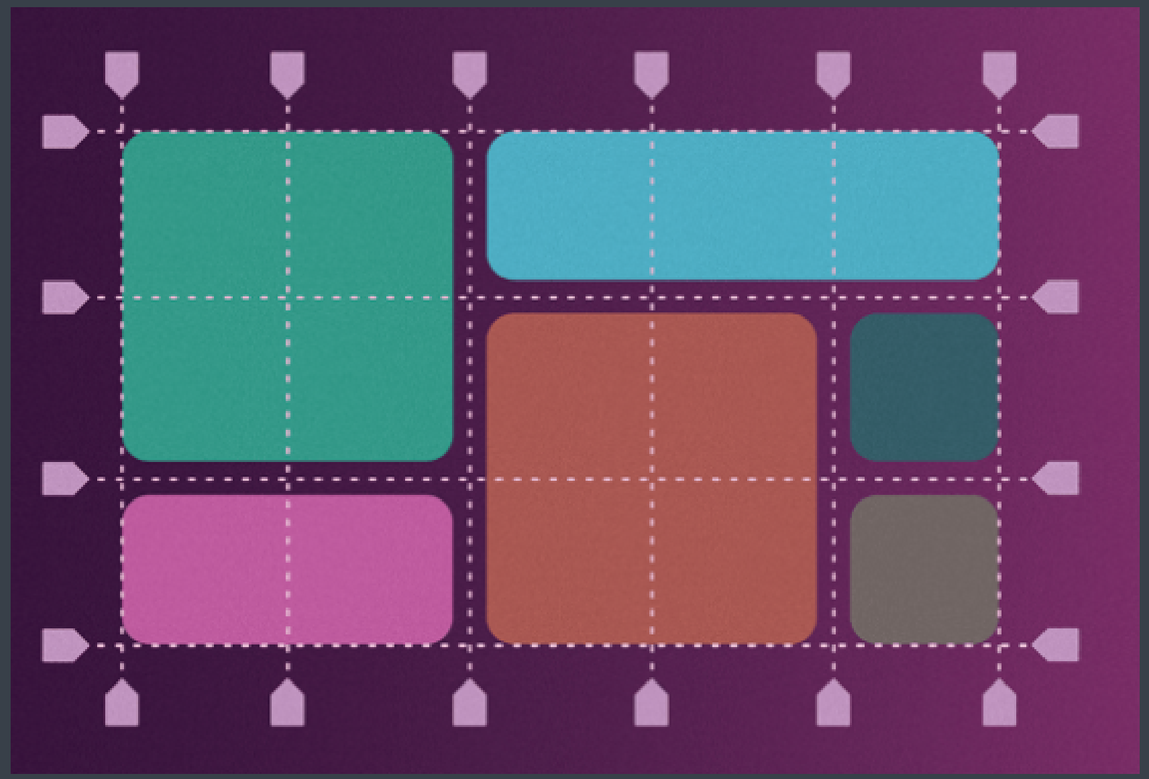
justify-items


start


center


end


stretch



Responsive Web Design

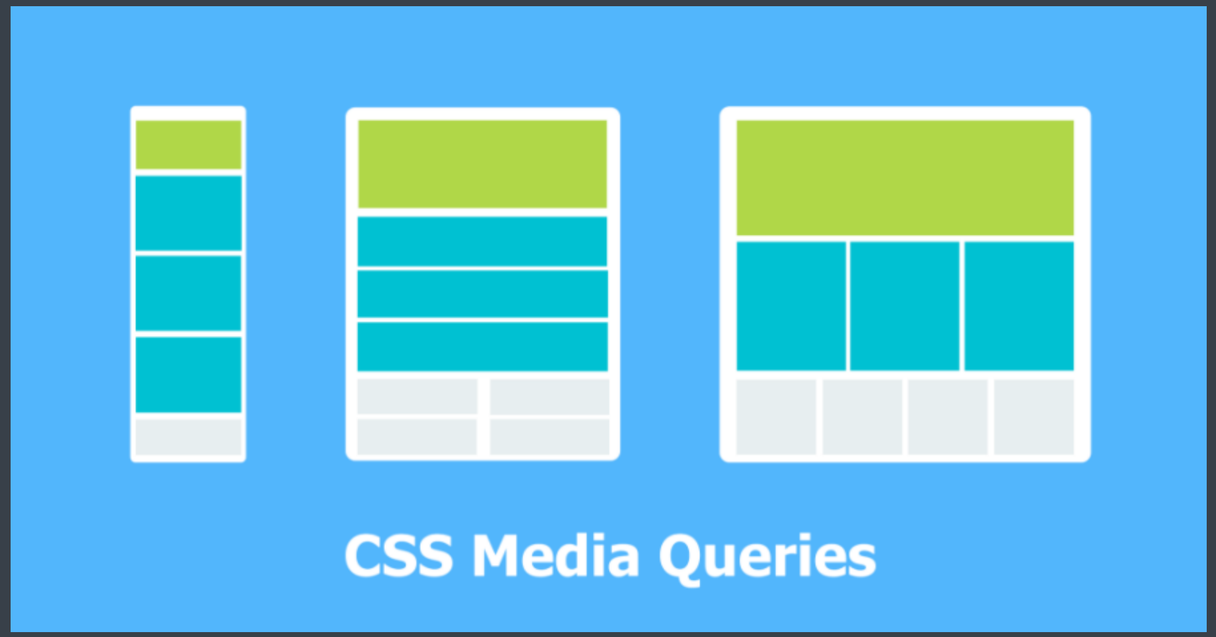
Introduction to Responsive Web Design

Responsive Web Design

- An approach to web design that aims to make web pages look good on various devices and screen sizes.
- It's useful for creating more complex grid structures compared to Flexbox.

Media Queries:

Rules that allow you to apply different styles to a web page based on the characteristics of the device, such as screen width or height.essential for creating responsive web designs.



Media Queries

```
@media (width : 600px) {
    div {
        background-color : red;
    }
}

@media (min-width : 600px) {
    div {
        background-color : red;
    }
}
```

Media Queries

```
@media (min-width : 200px) and (min-width : 300px) {
    div {
        background-color : red;
    }
}
```

Mobile-First Design Principles

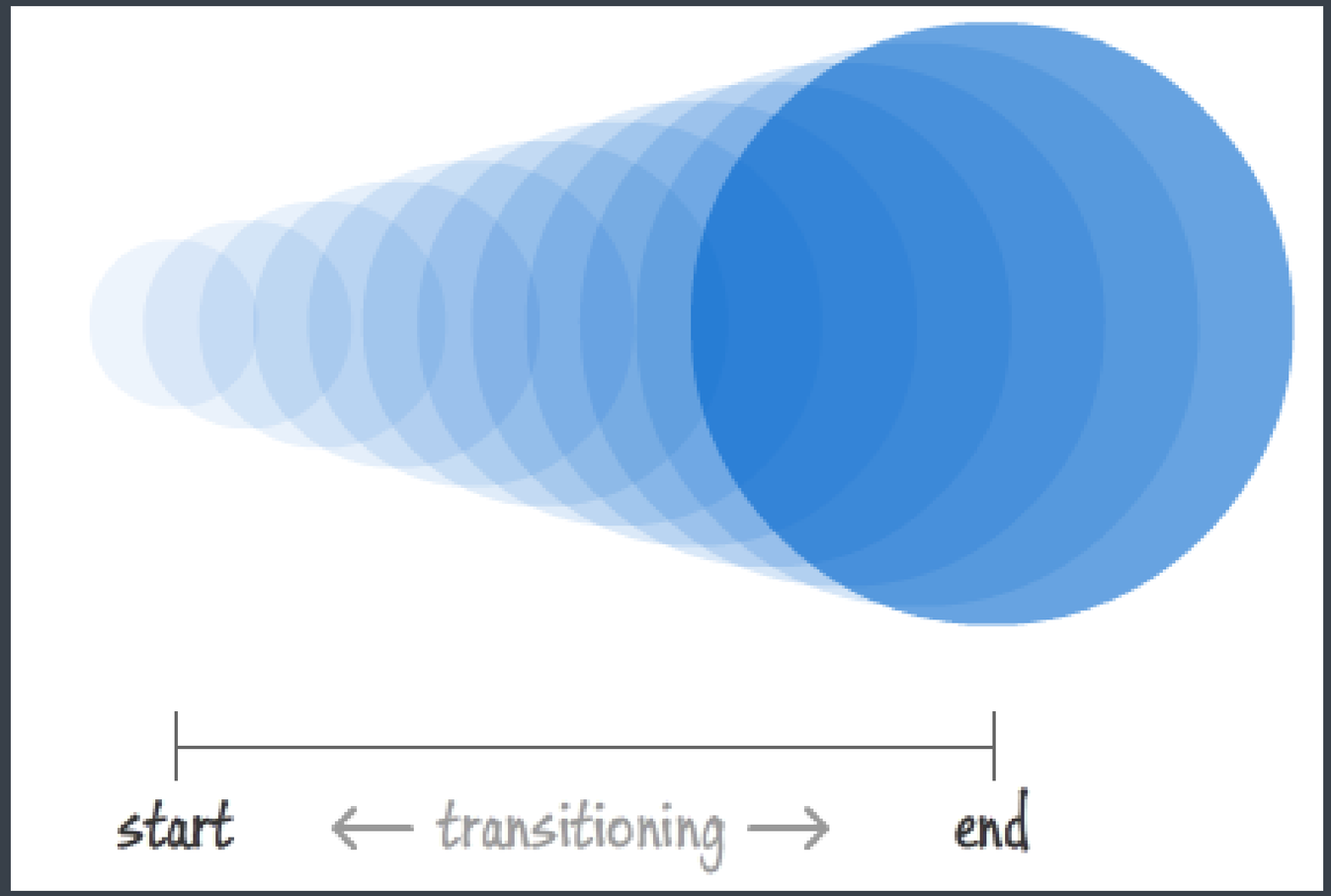
Mobile First design in CSS refers to the approach of designing and developing a website by prioritizing the styling and layout for mobile devices first, and then using CSS media queries to adapt the design for larger screen sizes

Transitions & Transform

CSS Transition & Transform

- CSS Transitions
- CSS Transform

CSS Transition



CSS Transition

- CSS transitions allow you to smoothly change property values over a specified duration.
- This creates fluid animations and interactions on web elements.

Transition Property

Specifies the CSS property you want to apply the transition to.

```css
.box {
  transition-property: width;
}
```

Transition Duration

Sets the time it takes for the transition to complete.

```
.box {
  transition-duration: 0.5s; /* 0.5 seconds */
}
```

Transition Timing-function

Defines the acceleration curve of the transition.

```
.box {
  transition-timing-function: ease-in-out; /* Ease in and out */
}
```

Transition Delay

Adds a delay before the transition starts.

```
.box {
  transition-delay: 0.2s; /* 0.2 seconds delay */
}
```

CSS Transition Shorthand

- transition-property
- transition-duration
- transition-timing-function
- transition-delay

CSS Transform

- The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.

CSS Transform

```
element {
  transform:
    translate(x, y)
    translateX(x)
    translateY(y)
    scale(x, y)
    scaleX(x)
    scaleY(y)
    rotate(deg)
    skew(x, y)
    skewX(x)
    skewY(y)
    matrix(a, b, c, d, tx, ty)
    matrix3d(values)
    perspective(value)
    rotateX(deg)
    rotateY(deg)
    rotateZ(deg)
    rotate3d(x, y, z, angle);
}
```

Animation

CSS Animation

- Animation Properties
- Animation Shorthand
- % in Animation

CSS Animation

CSS animations allow you to create dynamic and visually appealing effects on web elements.

CSS Animation

@keyframes: Defines the animation sequence and keyframes at specific points in time.

```
@keyframes slide-in {
  0% {
    transform: translateX(-100%);
  }
  100% {
    transform: translateX(0);
  }
}
```

Animation Properties

- animation-name
- animation-duration
- animation-timing
- animation-delay
- animation-iteration
- animation-direction

Animation Name

- Specifies the name of the keyframes animation.

```
.element {
  animation-name: slide-in;
}
```

Animation Duration

- Sets the duration of the animation (e.g., 2 seconds).

```
.element {
  animation-duration: 2s;
}
```

Animation Timing

- Defines the acceleration curve (e.g., ease-in-out).

```
.element {
  animation-timing-function: ease-in-out;
}
```

Animation Delay

- adds a delay before the animation starts (e.g., 1 second).

```
.element {
  animation-delay: 1s;
}
```

Animation Iteration

- Specifies how many times the animation repeats (e.g., infinite for continuous looping).

```
.element {
  animation-iteration-count: infinite;
}
```

Animation Direction

- determines the animation's direction (e.g., alternate for back-and-forth).

```
.element {
  animation-direction: alternate;
}
```

Animation Shorthand

- The animation shorthand property combines multiple animation-related properties into a single declaration.

```
.element {
  animation: slide-in 2s ease-in-out 1s infinite alternate;
}
```

Percentage in Animation

- In CSS animations, you can use percentages to specify keyframes' timing. Percentages represent the point in time during the animation's duration.

```
@keyframes fade-in {
  0% {
    opacity: 0;
  }
  100% {
    opacity: 1;
  }
}
```

JavaScript

What is JavaScript

- Widely-used programming language that is primarily used for web development.
- It allows developers to add interactivity and dynamic behavior to websites.
- JavaScript is a client-side scripting language.

JS Role in Web Development

- By enabling the creation of dynamic and interactive web applications.
- It enhances the user experience by providing features like .

1. form validation
2. responsive design
3. real-time updates,

First JS code

JavaScript within an HTML document using <script> tags

```
<!DOCTYPE html>
<html>
<head>
    <title>My First JavaScript</title>
</head>
<body>
    <script>
        // Your JavaScript code goes here
        alert('Hello, world!');
    </script>
</body>
</html>
```

Output:

"Hello, world!"

JS Syntax Overview

Variable declaration

```
var greeting = 'Hello, world!';
```

Function definition

```
function sayHello() {
    console.log(greeting);
}
```

Function call

```
sayHello();
```

JS Output

- innerHTML.
- document.write().
- window.alert().
- console.log().

Inner HTML

Input:

```
<script>
    var element = document.getElementById("output");
    element.innerHTML = "Hello, World!";
</script>
```

Output:

"Hello, world!"

Document Write

Input:

```
<script>
    document.write("Hello, World!");
</script>
```

Output:

"Hello, world!"

Window Alert

Input:

```
<script>
    window.alert("Hello, World!");
</script>
```

Output:

"Hello, world!"

Window Alert

Input:

```
<script>
    console.log("Hello, World!");
    console.log(123);
    console.log(document);
    console.log("Hello World", 123);
</script>
```

Output:

Hello, world!
123
Hello world 123

JS Basics

- Variables and Data Types.
- Operators.
- Conditional Statements.
- Loops.
- Arrays.
- Objects.

Variables

- Automatically
- Using var
- Using let
- Using const

Automatically

Input:

```
<script>
    var x = 5;
    var y = 6;
    var z = x + y;
</script>
```

Output:

11

Var

Input:

```
var x = 5;
var y = 6;
var z = x + y;
```

Output:

11

Let

Input:

```
let x = 5;
let y = 6;
let z = x + y;
```

Output:

11

Const

Input:

```
const x = 5;
const y = 6;
const z = x + y;
```

Output:

11

Datatypes

1. String
2. Number
3. Bigint
4. Boolean
5. Undefined
6. Object

Numbers

```
let length = 16;
let weight = 7.5;
```

Strings

```
let color = "Yellow";
let lastName = "Johnson";
```

Bigint

```
let x = BigInt("12345678901234567890123456789O");
```

Booleans

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

Undefined

```
let x;
x = 5;
x = "John";
```

Object

```
const person = {firstName:"John", lastName:"Doe"};
```

Operators

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. String Operators
5. Logical Operators
6. Bitwise Operators
7. Ternary Operators
8. Type Operators

Arithmetic operators

Addition (+)
Subtraction (-)
Multiplication (*)
Exponentiation (ES2016) (**)
Division (/)
Modulus (Division Remainder) (%)
Increment (++)
Decrement (--)

Assignment operators

(=) Assignment operator
(+=) Addition assignment
(-=) Subtraction Assignment
(*=) Multiplication Assignment
(/=) Division Assignment
(%=) Remainder Assignment
(**=) Exponentiation Assignment

Comparision operators

(==) equal to
(===) equal value and equal type
(!=) not equal
(!==) not equal value or not equal type
(>) greater than
(<) less than
(>=) greater than or equal to
(<=) less than or equal to
(?) ternary operator

Logical operators

(&&) logical and

(||) logical or

(!) logical not

Conditional Statements

Conditional statements like if, else if, and else allow you to make decisions in your code based on conditions.

Loops

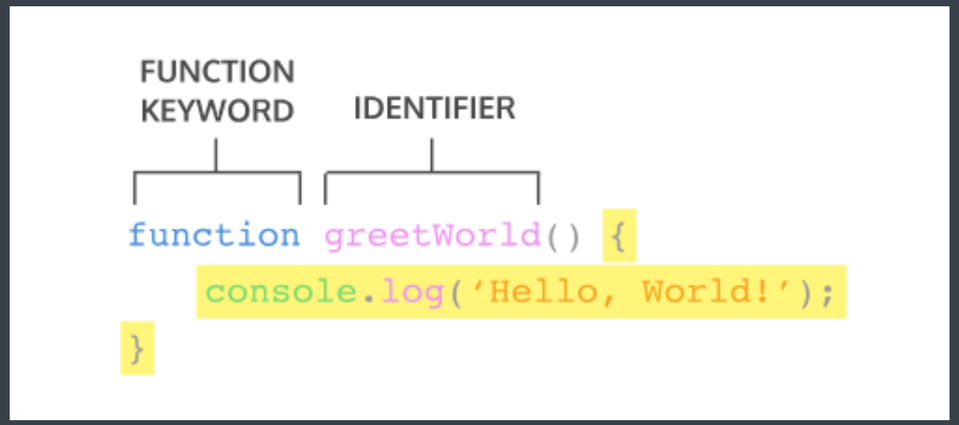
Loops like for and while enable you to execute code repeatedly. They are useful for iterating through arrays, lists, or performing tasks a specific number of times.

Javascript Functions

Javascript Functions

- Function Declaration
- Function Expression
- Parameters
- Arguments
- Return Statements
- Scope and Closures
- Anonymous Functions

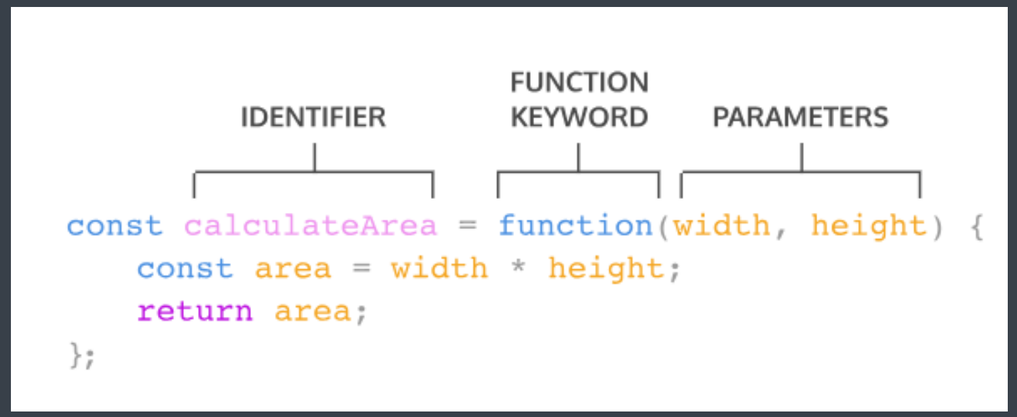
Function Declaration




```
function greetWorld(name) {
    console.log('Hello, World!');
}
```

- JavaScript functions are defined with the function keyword.
- You can use a function declaration or a function expression.

Function Expression

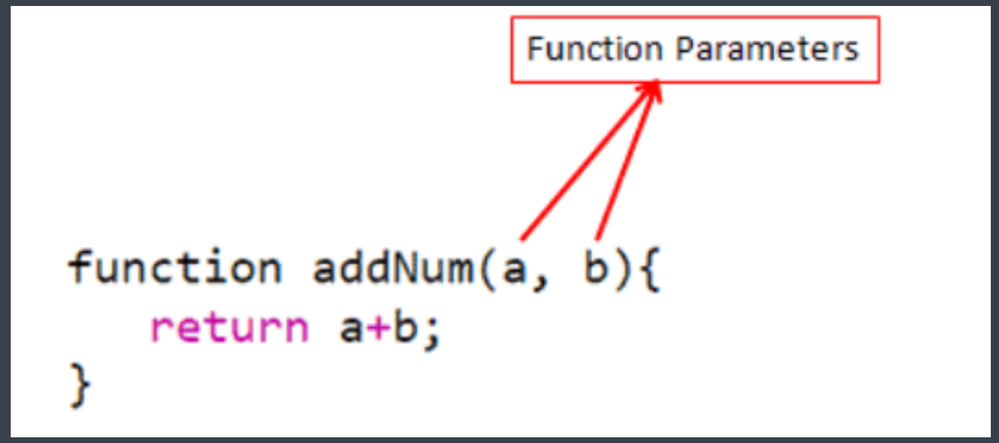



```
const calculateArea = function(x, y) {
  const area = x * y;
  return area;
};
```

- A JavaScript function can also be defined using an expression.
- A function expression can be stored in a variable

Parameters

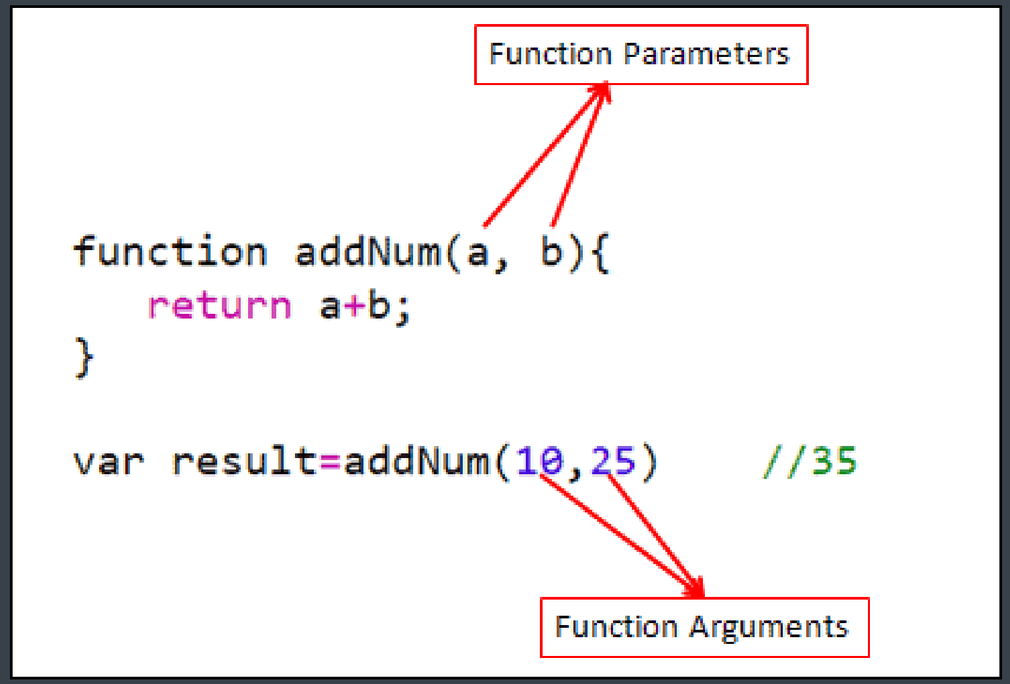
```
function addNum(a, b) {
  return a + b;
}
```



- **Function parameters are the names listed in the function definition.**

Arguments

- **Function arguments are the real values passed to (and received by) the function.**




```
function addNum(a, b) {
  return a + b;
}

var result = addNum(10, 25); // The result will be 35.
```

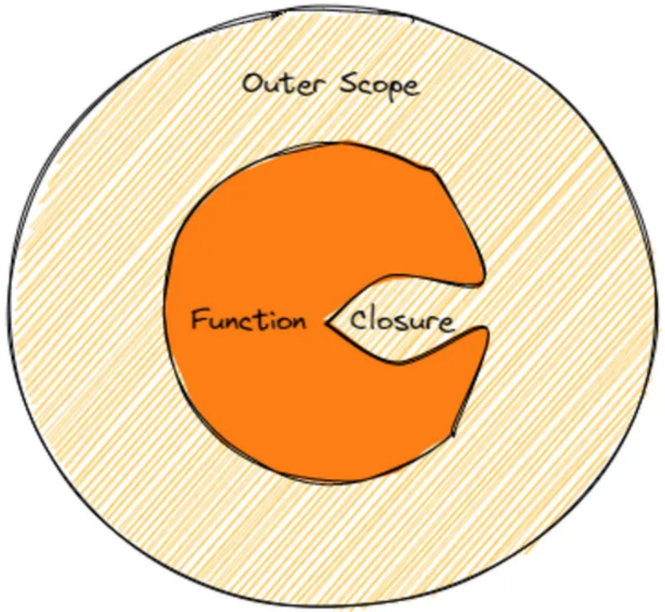
Return Statement

- The return statement in the add function returns the sum of x and y, which is stored in the result variable.es passed to (and received by) the function.

```
function addNum(a, b) {
  return a + b;
}

var result = addNum(10, 25); // The result will be 35.
```

Scope and Closure



Scope and Closure

- A scope in JavaScript defines what variables you have access to. There are two kinds of scope – global scope and local scope.
- A closure is usually returned so you can use the outer function’s variables at a later time.

Anonymous Function

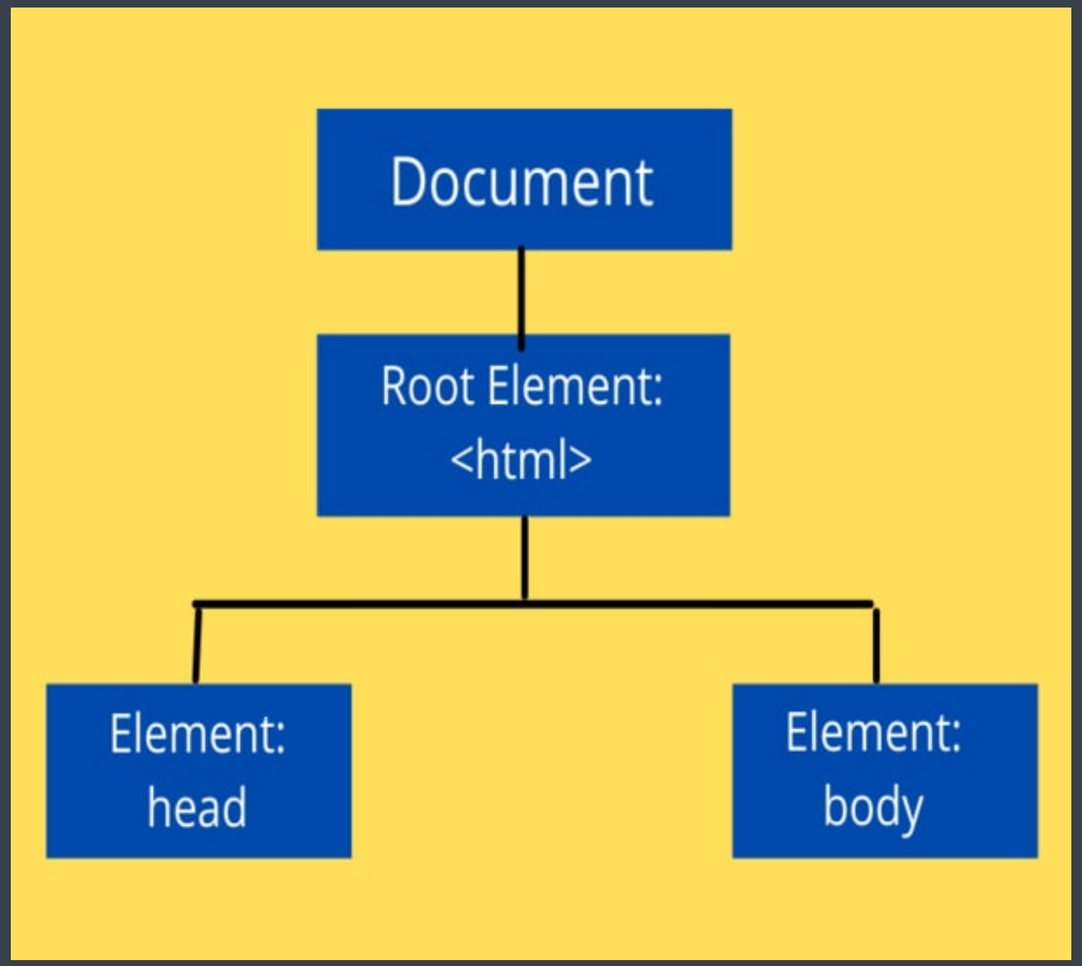
- It is a function that does not have any name associated with it.
- An anonymous function is not accessible after its initial creation, it can only be accessed by a variable it is stored in as a function as a value.
- An anonymous function can also have multiple arguments, but only one expression.

DOM Manipulation

Dom Manipulation

- Introduction to the DOM
- Selecting DOM Elements
- Modifying DOM Elements
- Creating and Deleting Elements
- Event Handling
- Traversing the DOM
- Manipulating CSS Classes
- Forms and Form Handling
- AJAX and DOM Manipulation

Dom Manipulation



Dom Manipulation

- Document Object Model of the page.
- The HTML DOM model is constructed as a tree of Objects
- The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

Selecting DOM Elements

- document.getElementById("myElement") allows you to select an element by its unique ID.
- document.getElementsByClassName("myClass") returns a collection of elements with the specified class name.

```
// Selecting elements by ID
const elementById = document.getElementById("myElement");

// Selecting elements by class
const elementsByClass = document.getElementsByClassName("myClass");
```

Selecting DOM Elements

- document.getElementsByTagName("div") returns a collection of elements with the specified tag name.
- document.querySelector(".myClass") allows you to select elements using CSS selector syntax.

```
// Selecting elements by tag name
const elementsByTag = document.getElementsByTagName("div");

// Selecting elements using querySelector
const elementByQuery = document.querySelector(".myClass");
```

Modifying DOM Elements

- .textContent property allows you to change the text content of an element.
- .setAttribute("attributeName", "attributeValue") lets you modify element attributes.
- .style.property allows you to change element styles (e.g., .style.color to change text color).

```
element.textContent = "New content";
element.setAttribute("src", "newimage.jpg");
element.style.color = "red";
```

Creating & Deleting Elements

- document.createElement("div") creates a new element, which can be added to the DOM.
- .appendChild(newElement) adds a child element to another element.
- .parentNode.removeChild(element) removes an element from the DOM.

```
const newElement = document.createElement("div");
document.body.appendChild(newElement);
const elementToRemove = document.getElementById("elementToRemove");
elementToRemove.parentNode.removeChild(elementToRemove);
```

Event Handeling

- addEventListener(eventType, callback) lets you attach event listeners to elements.

```
// Adding an event listener
element.addEventListener("click", function() {
  alert("Element clicked!");
});
```

Event Handeling

- event.preventDefault() is used to prevent the default behavior of an event (e.g., preventing a form from submitting).

```
// Handling form submissions
const form = document.getElementById("myForm");
form.addEventListener("submit", function(event) {
  event.preventDefault();
  // Process form data
});
```

Traversing the DOM

- ‘.parentNode’ and .childNodes allow you to navigate the DOM hierarchy.

```
// Navigating parent and child elements
const parentElement = element.parentNode;
const childElements = element.childNodes;
```

Traversing the DOM

- .nextElementSibling and .previousElementSibling help you find adjacent sibling elements.

```
// Finding siblings
const nextSibling = element.nextElementSibling;
const previousSibling = element.previousElementSibling;
```

Manipulating CSS Classes

- .classList allows you to work with an element's classes.
- .classList.add("className") adds a class to an element.

```
// Adding a CSS class
element.classList.add("active");
```

Manipulating CSS Classes

- .classList.remove("className") removes a class.
- .classList.contains("className") checks if an element has a specific class.

```
// Removing a CSS class
element.classList.remove("inactive");

// Checking if a class exists
const hasClass = element.classList.contains("active");
```

Form Handling

- Forms can be accessed and manipulated using JavaScript.
- inputField.value allows you to get or set the value of form input fields.

```
// Accessing form elements
const inputField = document.getElementById("inputField");
const submitButton = document.getElementById("submitButton");
```

Form Handling

- Form submissions can be intercepted and processed using event listeners.

```
// Form submission
submitButton.addEventListener("click", function() {
  const inputValue = inputField.value;
  // Process input value
});
```

AJAX and DOM Manipulation:

- AJAX (Asynchronous JavaScript and XML) allows you to retrieve and update data from a server without refreshing the entire web page.
- The Fetch API and libraries like jQuery simplify making AJAX requests and updating the DOM with fetched data.

```
// Using Fetch API to retrieve data and update the DOM
fetch("https://api.example.com/data")
  .then(response => response.json())
  .then(data => {
    // Update DOM with fetched data
  });
```