CSS Styling Buttons

CSS allows us to style HTML buttons in many different ways to make them **attractive, interactive, and user-friendly**.
Buttons are commonly created using:

- <button>
- <input type="button">
- <a> (styled as a button)

Common CSS Properties Used for Buttons

- **background-color** – sets button background color
- **color** – sets text color
- **border** – sets button border
- **padding** – space between text and border
- **border-radius** – rounds button corners
- **box-shadow** – adds shadow
- **text-align** – aligns text
- **font-size** – sets text size
- **text-decoration** – removes underline from <a>
- **cursor** – changes mouse pointer on hover

CSS Basic Button Styling

Example

```
.button {
  background-color: red;
  border: none;
```

```
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  cursor: pointer;
}
```

Use: Creates a simple and reusable button style.

CSS Button Colors

Buttons can have different background and text colors.

Example

```
.button1 { background-color: #04AA6D; } /* Green */
.button2 { background-color: #008CBA; } /* Blue */
.button3 { background-color: #f44336; } /* Red */
.button4 { background-color: #e7e7e7; color: black; } /* Gray */
.button5 { background-color: #555555; } /* Black */
```

Use: Visual differentiation of actions.

CSS Button Sizes

Using font-size

```
.button1 { font-size: 10px; }
.button2 { font-size: 12px; }
.button3 { font-size: 16px; }
```

.button4 { font-size: 20px; }

.button5 { font-size: 24px; }

Using padding

.button1 { padding: 10px 24px; }

.button2 { padding: 12px 28px; }

.button3 { padding: 14px 40px; }

.button4 { padding: 32px 16px; }

.button5 { padding: 16px; }

Use: Controls button size and spacing.

CSS Rounded Buttons

Example

.button1 { border-radius: 2px; }

.button2 { border-radius: 4px; }

.button3 { border-radius: 8px; }

.button4 { border-radius: 12px; }

.button5 { border-radius: 50%; }

Use: Modern and smooth button appearance.

CSS Button Borders

Example

.button1 { border: 2px solid #04AA6D; }

.button2 { border: 2px dotted #008CBA; }

.button3 { border: 2px dashed #f44336; }

.button4 { border: 1px solid #e7e7e7; }

.button5 { border: 1px solid #555555; }

Use: Highlight button boundaries.

CSS Hoverable Buttons

Hover effect changes style when mouse is over the button.

Example

.button1:hover { background-color: #04AA6D; color: white; }

.button2:hover { background-color: #008CBA; color: white; }

.button3:hover { background-color: #f44336; color: white; }

.button4:hover { background-color: #e7e7e7; color: black; }

.button5:hover { background-color: #555555; color: white; }

Use: Improves user interaction.

CSS Buttons with Shadow

Example

```
.button1 {
  box-shadow: 0 8px 16px rgba(0,0,0,0.6);
}


.button2:hover {
  box-shadow: 0 8px 16px rgba(0,0,0,0.6);
}
```

Use: Adds depth and 3D look.

CSS Disabled Button

Example

```
.disabledbtn {
  opacity: 0.6;
  cursor: not-allowed;
}
```

Use: Shows inactive or unavailable actions.

CSS Button Width

Example

```
.button1 { width: 250px; }
.button2 { width: 50%; }
.button3 { width: 100%; }
```

Use: Fixed or responsive button size.

CSS Horizontal Button Group

Example

```
.btn-group {
  display: flex;
  flex-wrap: wrap;
}

.button {
  background-color: #04AA6D;
  border: none;
  color: white;
```

```
  padding: 15px 32px;
  font-size: 16px;
  cursor: pointer;
}

.btn-group .button:hover {
  background-color: dodgerblue;
}
```

Use: Group related buttons horizontally.

CSS Bordered Button Group

Example

```
.btn-group .button {
  border: 1px solid green;
}

.btn-group .button:not(:last-child) {
  border-right: none;
}
```

Use: Clean grouped buttons without double borders.

CSS Vertical Button Group

Example

```
.btn-group {
  display: flex;
```

```
  flex-direction: column;
}
```

Use: Sidebar menus or stacked buttons.

CSS Animated Buttons (Examples)

Hover Effect

```
.button:hover {
  opacity: 0.8;
}
```

Pressed Effect

```
.button:active {
  transform: translateY(2px);
}
```

Fade In

```
.button {
  transition: 0.5s;
}
```

Use: Enhances user experience with animation.

CSS Pagination

Learn how to create a responsive pagination using CSS.

If you have a website with lots of pages, you may want to add some sort of pagination on each page.

Pagination is typically a series of links, wrapped in an unordered list (<ul>). Each link represents an individual page number. In addition there are "previous" and "next" controls:

Example

A simple pagination:

```
.pagination {
  display: flex;
  justify-content: center;
  list-style: none; /* remove list bullets */
  padding: 0px;
}

.pagination li a {
  display: block; /* let links fill the list item */
  padding: 8px 12px;
  text-decoration: none;
  border: 1px solid gray;
  color: black;
  margin: 0 4px;
  border-radius: 5px; /* add rounded borders */
}
```

CSS Multi-column Layout

The **CSS Multi-column Layout Module** is used to divide text into **multiple columns**, similar to **newspapers or magazines**.
It helps in presenting long text in a clean and readable format.

CSS Multi-column Properties

The following properties are used to create and control multi-column layouts:

- column-count
- column-gap
- column-rule-style
- column-rule-width
- column-rule-color
- column-rule
- column-span
- column-width

---

1. CSS Create Multiple Columns (column-count)

The column-count property specifies the **number of columns** an element should be divided into.

Example

```
div {
  column-count: 3;
}
```

Use: Divides text inside a into **three columns**.

---

2. CSS Column Gap (column-gap)

The column-gap property defines the **space between columns**.

Example

```
div {
  column-gap: 40px;
}
```

Use: Adds spacing between columns for better readability.

---

3. CSS Column Rule

Column rules are **vertical lines** shown between columns.

Column Rule Style

```
div {
  column-rule-style: solid;
}
```

Column Rule Width

```
div {
  column-rule-width: 1px;
}
```

Column Rule Color

```
div {
  column-rule-color: lightblue;
}
```

---

4. CSS Column Rule (Shorthand)

The column-rule property sets **width, style, and color** in one line.

Example

```
div {
  column-rule: 1px solid lightblue;
}
```

Use: Adds a clean separator between columns.

---

5. CSS Column Span (column-span)

The column-span property specifies how many columns an element should span across.

Example

```
h2 {
  column-span: all;
  text-align: center;
}
```

Use: Headings span across **all columns**.

---

6. CSS Column Width (column-width)

The column-width property defines the **ideal width** for each column.
The browser automatically decides the number of columns.

Example

```
div {
  column-width: 100px;
}
```

Use: Creates responsive multi-column layouts.

CSS Media Queries

CSS **Media Queries** allow you to apply different CSS styles based on the **device screen size, resolution, orientation, or user preferences**.
They are mainly used to create **responsive web pages** that work well on **mobiles, tablets, and desktops**.

The @media rule is used to write media queries in CSS.

Media Query Syntax

A media query includes:

- an optional **media type**
- one or more **media features**

Syntax

```
@media [not] media-type and (media-feature: value) {
  /* CSS rules */
}
```

How Media Queries Work

- A media query returns **true** or **false**
- If the condition is **true**, the CSS inside the block is applied
- If the condition is **false**, the styles are ignored
- Normal CSS cascading rules still apply

Meaning of Keywords

not

The not keyword **reverses** the meaning of the media query.

```
@media not screen {
  body {
    background-color: gray;
  }
}
```

---

and

The and keyword is used to **combine multiple conditions**.

```
@media screen and (min-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

CSS Media Types

Media types define the type of device the styles apply to.

Media Type	Description
all	Default, works on all devices
screen	Screens like mobiles, tablets, desktops
print	Print preview or printed pages

CSS Media Features

Media features describe **device characteristics**.

Feature	Description
min-width	Minimum width of viewport
max-width	Maximum width of viewport
width	Exact width of viewport
min-height	Minimum height
max-height	Maximum height
orientation	Portrait or landscape

Feature	Description
resolution	Screen resolution
prefers-color-scheme	Light or dark mode

Media Query Examples

Example 1: Minimum Width

Change background color if screen width is **480px or more**.

```
@media screen and (min-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
```

Use: Tablets and larger devices.

Example 2: Width Range

Apply styles only when screen width is **between 480px and 768px**.

```
@media screen and (min-width: 480px) and (max-width: 768px) {
  body {
    background-color: lightgreen;
  }
}
```

Use: Tablet-specific layout.

Why Media Queries Are Important

- Make websites **responsive**

- Improve **user experience**
- Adapt layout for **different devices**
- Essential for **mobile-friendly design**
