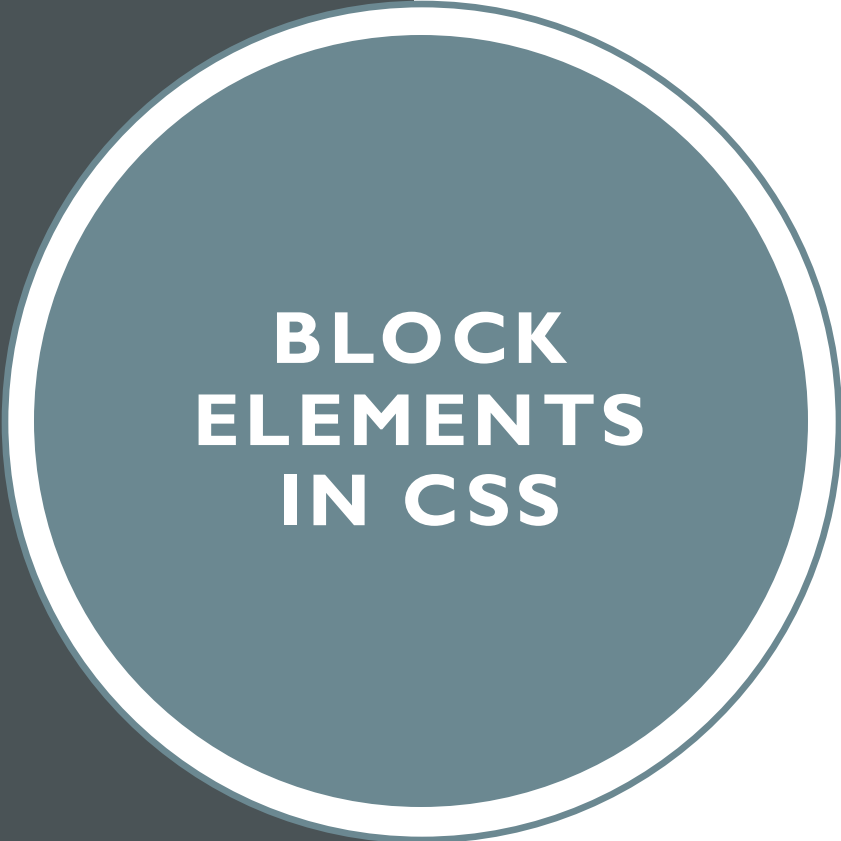


# **WORKING WITH BLOCK ELEMENTS AND OBJECTS IN CSS**

**Unit-III**

**SCS181 Web Technologies-I**



# BLOCK ELEMENTS IN CSS

- Block elements are the fundamental building blocks of web page layouts.
- These elements typically occupy the full width of their parent container and stack on top of each other, starting on a new line.
- In CSS, you can manipulate block elements and objects to control their size, positioning, and behavior on the page.

## WHAT ARE BLOCK ELEMENTS?

- Block elements are HTML elements that by default take up the entire width available in their container, and they always start on a new line. Some common block elements include:
- `<div>`, `<p>`, `<h1>` - `<h6>`, `<section>`, `<article>`, `<header>`, `<footer>`, and `<nav>`.
- By contrast, **inline elements** only take up as much width as needed and do not force a new line break (e.g., `<span>`, `<a>`, `<em>`, `<strong>`).

## EXAMPLE OF BLOCK ELEMENTS:

- `<div>This is a block-level element.</div>`
- `<p>This is another block-level element.</p>`
- `<h1>This is a heading block-level element.</h1>`

# WIDTH AND HEIGHT

Block elements, by default, take up 100% of the width of their parent container. You can control their size using the width and height properties.

## a. width

Defines the width of the block element.

**Values:** Length values (px, em, %), auto (default).

### Example:

```
div { width: 80%; /* The div will take up 80% of its  
parent's width */ }
```

## b. height

Sets the height of the block element.

**Values:** Length values (px, em, %), auto (default).

### Example:

```
section { height: 400px; /* The section element will  
have a height of 400 pixels */ }
```

# PADDING, BORDER, AND MARGIN

- Block elements are affected by the box model, which includes padding, border, and margin to control space inside and around elements.

## a. padding

Sets the space between the content of the block element and its border. Padding increases the size of the element's content box without affecting its outer size (border area).

### Example:

```
div { padding: 20px; /* Adds 20px of space inside the div */ }
```

## b. border

Defines the border around the block element. You can set the thickness, style, and color of the border.

### Example:

```
div { border: 2px solid black; /* Adds a black border around the div */ }
```

## c. margin

Controls the space outside the block element. Margin creates space between neighboring block elements or other objects on the page.

### Example:

```
div { margin: 30px 0; /* Adds 30px margin above and below the div */ }
```

# DISPLAY PROPERTY FOR BLOCK ELEMENTS

The display property can change how elements behave in the layout. By default, many elements (like `<div>` and `<p>`) have a `display: block;` property. You can manipulate this property to convert inline elements into block elements or vice versa.

## a. `display: block`

Forces an element to behave as a block-level element, even if it's normally inline.

### Example:

```
span { display: block; /* The span will now behave like  
a block-level element */ }
```

## b. `display: inline-block`

Makes an element behave like a block but still flow inline with other elements (e.g., inside a paragraph).

### Example:

```
button { display: inline-block; padding: 10px 20px;  
margin: 10px; }
```

# POSITIONING BLOCK ELEMENTS

The **position** property allows you to control how block elements are positioned relative to their normal flow, the page, or their parent container.

## a. position: relative

Positions the block element relative to its normal position without affecting other elements. You can then adjust its position using top, right, bottom, and left.

### Example:

```
div { position: relative; top: 10px; /* Moves the div 10px down from its normal position */ left: 20px; /* Moves the div 20px right from its normal position */ }
```

## b. position: absolute

The element is positioned relative to its nearest positioned ancestor (not the default static element). It is removed from the normal document flow, so it doesn't affect surrounding elements.

### Example:

```
.container { position: relative; } .child { position: absolute; top: 0; right: 0; /* The child element will be positioned in the top-right corner of the container */ }
```

## c. position: fixed

Fixes the block element to a specific position on the page, and it stays there even when the page is scrolled.

### Example:

```
header { position: fixed; top: 0; width: 100%; /* The header stays fixed at the top of the screen */ }
```



# CENTRING BLOCK ELEMENTS

Block elements naturally take up the full width of their container, but you can center them using `margin: auto` if a width is defined.

## Example:

```
div { width: 50%;
```

```
margin: 0 auto;
```

```
/* Centers the div horizontally */ }
```

For vertically centering elements inside their container, using **Flexbox** or **Grid** is the best approach.

## Example using Flexbox:

```
.container {
```

```
display: flex;
```

```
justify-content: center;
```

```
align-items: center;
```

```
height: 100vh;
```

```
/* Makes sure the container takes up the full  
viewport height */
```

```
}
```

# WORKING WITH OVERFLOW IN BLOCK ELEMENTS

Sometimes, the content inside a block element can overflow its box. The overflow property manages how the content is displayed when it goes beyond the element's size.

## **overflow**

Controls the handling of content overflow.

### **Values:**

visible (default): Content is not clipped.

hidden: Content is clipped, and overflow is hidden.

scroll: Adds scrollbars to view overflowing content.

auto: Scrollbars appear only if needed.

### **Example:**

```
div { width: 200px; height: 100px; overflow: scroll; /* Adds  
scrollbars if content exceeds these dimensions */ }
```

# FLOATING BLOCK ELEMENTS

The float property allows block elements to float left or right, with content (like text or other inline elements) flowing around them. It's commonly used for wrapping text around images or aligning elements horizontally.

## Example of Floating an Image:

```
img { float: left; margin: 0 10px 10px 0; /* Wraps text  
around the image with some margin */ }
```

To clear floating elements and prevent layout issues, use the clear property or a clearfix.

## Clearfix Example:

```
.clearfix::after { content: ""; display: table; clear: both; }
```