Display & Positioning

CSS controls how elements appear on the page using **display** values, **positioning**, **z-index**, and **visibility rules**.
Understanding these concepts is essential for layout design and responsive UI development.

This chapter covers display types, position properties, z-index stacking, and the difference between visibility and display.

5.1 Display Property

The display property defines **how an element is shown** in the document flow.

---

1. display: block

Characteristics:

- Starts on a **new line**
- Takes **full width**
- Can set **width, height, margin, padding**

Examples of block elements:

<p>

<h1>–<h6>

<section>

Example:

```
div {
        display: block;
}
```

---

2. display: inline

Characteristics:

- Does **not** start on a new line
- Only takes **content width**
- **Cannot** set width and height
- Best for text-level elements

Examples of inline elements:

<a>

Example:

```
span {
    display: inline;
}
```

---

3. display: inline-block

Characteristics:

- Behaves like **inline** (stays in same line)
- But accepts **width, height, padding, margins** like block
- Useful for navigation menus, cards, buttons

Example:

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

---

4. display: none

Characteristics:

- Completely removes the element from layout
- Other elements shift as if it does not exist
- Invisible + no space taken

Example:

```
.hidden {

        display: none;

}
```

Display Type Comparison

Display	Starts new line?	Width	Can set width/height?	Visible?	Space reserved?
block	Yes	Full width	Yes	Yes	Yes
inline	No	Content width	No	Yes	Yes
inline-block	No	Content width	Yes	Yes	Yes
none	No	None	N/A	No	No

5.2 Position Property

The position property controls **how elements are positioned** relative to the page or other elements.

---

1. static (default)

Characteristics:

- Normal flow of the document
- Not affected by top/left/right/bottom
- Default behavior for all elements

Example:

```
div {

    position: static;

}
```

---

2. relative

Characteristics:

- Positioned **relative to itself**
- The element remains in normal flow
- top/left moves it visually **without affecting surrounding elements**

Example:

```
.box {

    position: relative;

    top: 20px;

    left: 20px;

}
```

---

3. absolute

Characteristics:

- Removed from normal flow
- Positioned relative to the **nearest positioned ancestor** (ancestor with position: relative/absolute/fixed)
- If no ancestor → relative to the **viewport**

Example:

```
.child {
    position: absolute;
    top: 10px;
    right: 20px;
}
```

---

4. fixed

Characteristics:

- Fixed relative to the **viewport**
- Does not move even when scrolling
- Used for fixed headers, sidebars, floating buttons

Example:

```
nav {
    position: fixed;
    top: 0;
    width: 100%;
}
```

---

5. sticky

Characteristics:

- Behaves like **relative** until a scroll threshold is reached
- Then behaves like **fixed**
- Used for sticky headers, sticky table columns, sticky sidebars

Example:

```
header {
    position: sticky;
    top: 0;
}
```

Position Property Summary Table

Position	In normal flow?	Moves with top/left?	Scrolls?	Based on
static	Yes	No	Yes	Normal position
relative	Yes	Yes	Yes	Own position
absolute	No	Yes	No	Positioned ancestor
fixed	No	Yes	No	Viewport
sticky	Yes → No	Yes	Sometimes	Scroll position

5.3 z-index

z-index controls **stacking order** of elements on the z-axis (front to back).

Rules:

- Higher z-index = appears **above**
- Works only with **positioned elements**
  (position: relative, absolute, fixed, sticky)

Example:

```
.box1 {

        position: absolute;

        z-index: 1;

}


.box2 {

        position: absolute;

        z-index: 5;

}
```

Here, .box2 appears on top.

Important Notes on z-index

- z-index does **not work** on elements with position: static
- Parent stacking context affects children
- Negative z-index is allowed

5.4 Visibility vs Display

Many developers get confused between:

1. display: none

- Element is **removed** from the layout
- No space taken
- Not visible

Example

```
div {

    display: none;

}
```

---

2. visibility: hidden

- Element is **invisible**
- **Space is reserved**
- Still affects layout

Example:

```
div {

    visibility: hidden;

}
```

Display vs Visibility Comparison

Property	Visible?	Space Taken?	In Document Flow?
display: none	❌	❌	❌
visibility: hidden	❌	✔	✔

Practical Example

```
.visible {
        visibility: hidden;
}

.hidden {
        display: none;
}
```

- .visible → invisible but still takes space
- .hidden → removed completely

Real-world Uses

display: none

- Removing menu items
- Dynamic UI elements
- Conditional rendering

visibility: hidden

- Keeping table cell alignment
- Temporarily hiding elements without layout shift

Interview Questions (Display & Positioning)

1. Difference between block, inline, and inline-block?
2. What is the difference between position: absolute and fixed?
3. How does position: sticky work?
4. What is z-index and when does it work?
5. Difference between display: none and visibility: hidden?