Positioning in CSS

CSS positioning allows you to **move elements** from their default flow and place them **precisely** where you want on the page. It's an essential part of creating modern, interactive layouts.

The `position` Property

There are **five main values**:

Value for Position	Description
`static`	Default. Element stays in the normal document flow
`relative`	Moves the element **relative to its normal position**
`absolute`	Removes from flow; positions **relative to nearest positioned ancestor**
`fixed`	Positions the element **relative to the browser window**, even on scroll
`sticky`	Behaves like `relative`, but **sticks to a position while scrolling**

---

1. `static` (Default)

Every element is positioned statically by default.

```css
div {
  position: static;
}
```

You **can't move** statically positioned elements with `top`, `left`, etc.

---

2. `relative`

Moves the element **relative to where it would normally be**.

```css
.box {
  position: relative;
  top: 20px;
  left: 10px;
}
```

It stays in the document flow, but shifts slightly.

---

3. `absolute`

- Removes the element from normal flow.
- Positions it **relative to the closest ancestor with `position` set** (not `static`).
- If no positioned ancestor, it uses the `<html>` element.

```css
.parent {
  position: relative;
}

.child {
  position: absolute;
  top: 0;
```

```
  right: 0;
}
```

The `.child` will stick to the top-right of `.parent`.

---

4. `fixed`

- Stays in a fixed position **relative to the viewport**.
- Does **not move** when scrolling.

```
.banner {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
}
```

Great for sticky headers, floating buttons, or back-to-top links.

---

5. `sticky`

- Acts like `relative` until a scroll threshold is reached, then behaves like `fixed`.

```
.heading {
  position: sticky;
  top: 0;
  background: white;
}
```

Sticky headers or sidebars often use this behavior.

`top`, `right`, `bottom`, `left`

These properties only work with `relative`, `absolute`, `fixed`, or `sticky`.

```
.box {
  position: absolute;
  top: 50px;
  left: 100px;
}
```

`z-index`

Controls the **stacking order** of overlapping elements.

```
.modal {
  position: absolute;
  z-index: 100;
}
```

Higher `z-index` values appear **above** lower ones.

Summary

- Use `relative` for minor adjustments.
- Use `absolute` to fully control placement inside containers.
- Use `fixed` for elements that stay on screen while scrolling.
- Use `sticky` for scroll-based sticky behaviors.
- Always understand the **positioning context** — especially when using `absolute`.