🌐 CSS Positioning – Complete Notes

🔹 Overview

Positioning controls how elements are placed on a webpage.
There are **four main types of positioning**:

1. **Static**
2. **Relative**
3. **Absolute**
4. **Fixed**

Each defines *how an element sits in relation to other elements or the viewport.*

🧱 1. Static Positioning (Default)

- Every HTML element is **static by default**.
- Static means elements appear in the **normal document flow** (one after another, top to bottom).
- Even if you write position: static;, it changes nothing—it's already applied.
- Properties like top, left, right, bottom **don't affect** static elements.

🟩 Example:

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

➡️ The element just sits where HTML naturally places it.

🎯 2. Relative Positioning

- position: relative; moves an element **relative to its original static position**.
- The space the element originally occupied **is still preserved** in the layout.
- You can shift it using top, left, right, or bottom.

📘 Example:

```
.box {
 position: relative;
 top: 50px;
 left: 50px;
```

}

➡️ The element moves down and right 50px from its normal place.

🔸 **Important:**
"Relative" means *relative to itself*, **not** to other elements.

🧭 3. Absolute Positioning

- position: absolute; removes the element from normal flow.
- The element is placed **relative to its nearest positioned ancestor** (an ancestor with any position other than static).
- If no positioned ancestor exists, it's placed **relative to the page (top-left corner of the document)**.

📘 Example:

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

➡️ Moves 50px down and 50px right from either:

- its nearest positioned ancestor, or
- the top-left of the page (if none found).

🔸 To control where it anchors, set the parent as:

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

🧩 z-index

- Controls **stacking order** of overlapping elements.
- Higher z-index = element appears **on top**.
- Works only on positioned elements (relative, absolute, or fixed).

📘 Example:

```
.box1 { position: absolute; z-index: 1; }
```

```
.box2 { position: absolute; z-index: 5; }
```

➡️ .box2 sits above .box1.

🧠 Default z-index = 0
You can use negative values to push elements **behind others**.

📌 4. Fixed Positioning

- position: fixed; locks an element relative to the **browser window**, not the page.
- It **doesn't move when you scroll**.
- Always stays at the same coordinates within the viewport.

📘 Example:

```
nav {
 position: fixed;
 top: 0;
 left: 0;
}
```

➡️ Navigation bar stays visible even as you scroll down.

⚙️ Position vs Margin

- **Margins** create *space around* an element in the document flow.
- **Positioning** moves an element *independently of* other elements.
- When both exist, position shifts the element **after** margins are applied.

🎨 Bonus: Creating a Circle in CSS

To make a perfect circle:

```
.red-circle {
 width: 200px;
 height: 200px;
 background-color: red;
 border-radius: 50%;
}
```

➡️ border-radius: 50% turns any square into a perfect circle.

🧠 **Key Takeaways**

- **Static:** Default, no manual control.
- **Relative:** Shift from original position but still in flow.
- **Absolute:** Out of flow; positioned relative to nearest positioned ancestor.
- **Fixed:** Out of flow; positioned relative to viewport.
- **z-index:** Controls stacking on the Z-axis.
- **Position ≠ Margin:** Position works independently of spacing.