The display property in CSS determines how an element is visually presented on the page.

**1. block**

* **Behavior**: The element takes up the full width available (by default) and starts on a new line.
* **Examples**: <div>, <p>, <h1>, <section>
* **Usage**: Block elements will always begin on a new line, stacking vertically.

div {

display: block;

}

**2. inline**

* **Behavior**: The element takes up only as much width as necessary, and it doesn’t cause line breaks.
* **Examples**: <span>, <a>, <strong>
* **Usage**: Inline elements flow within text, meaning they don’t interrupt the flow of other elements, appearing on the same line if there’s space.

span {

display: inline;

}

**3. inline-block**

* **Behavior**: The element behaves like an inline element (stays on the same line as other inline elements) but can have block-level styling (e.g., width and height).
* **Examples**: <img>, <button>, <input>
* **Usage**: Useful when you want elements to align horizontally but still need to control their dimensions.

button {

display: inline-block;

width: 100px;

height: 50px;

}

**4. none**

* **Behavior**: The element is not displayed at all. It takes up no space in the layout and is completely removed from the document's flow.
* **Examples**: Used for hiding elements (commonly in JavaScript or CSS transitions).
* **Usage**: Typically used when you want to hide an element without removing it from the DOM.

div {

display: none;

}

**5. flex**

* **Behavior**: The element is treated as a flex container, and its children become flex items. The items are laid out according to the flexbox model, allowing for flexible and dynamic layouts.
* **Examples**: Used in layouts that need flexible and responsive alignment.
* **Usage**: Great for creating rows or columns of items that adjust based on the container size.

.container {

display: flex;

justify-content: space-between;

}

**6. grid**

* **Behavior**: The element is treated as a grid container, and its children become grid items. This layout model allows for the creation of complex 2D layouts with rows and columns.
* **Examples**: Used for more complex layouts than flexbox, with both rows and columns.
* **Usage**: Ideal for creating more structured, multi-row, and multi-column designs.

.container {

display: grid;

grid-template-columns: repeat(3, 1fr);

}

**Summary Chart**

| **display Value** | **Description** | **Example Use Case** |
| --- | --- | --- |
| block | Element takes full width and starts on a new line. | Block-level elements like <div>, <section>, <p> |
| inline | Element only takes necessary width, stays in line with others. | Inline elements like <span>, <a>, <strong> |
| inline-block | Inline, but can have block-level styling (width/height). | Elements like <img>, buttons that align horizontally |
| none | Element is not displayed and takes up no space. | Hide elements from the page (e.g., for JS or animations) |
| flex | Flexbox layout, items align based on flex model. | Responsive, flexible layouts like navbars, cards |
| grid | Grid layout with rows and columns. | Complex 2D layouts with structured rows/columns |

Each of these display values can be used to control the positioning, alignment, and layout of elements, giving you a variety of options for building different types of web designs.