# Normal Display Properties in CSS

The `display` property in CSS determines how an element is displayed in the document, whether it behaves as a block, inline, or another type of layout container. It's one of the most fundamental CSS properties for controlling layout.

---

## Syntax

```css
selector {
  display: value;
}
```

---

## Common `display` Values

### 1. `block`

- The element behaves as a block element.
- It takes up the full width available (stretches as far as its parent allows).
- Starts on a new line, stacking vertically.

**Example:**

```html
<div style="display: block;">I am a block element</div>
```

**Visual Behavior:** Takes up the full width and starts on a new line.

---

### 2. `inline`

- The element behaves as an inline element.
- It takes up only as much width as necessary (content width).
- Does not start on a new line; instead, it flows with surrounding text.

**Example:**

```html
<span style="display: inline;">I am an inline element</span>
```

**Visual Behavior:** Flows inline with text or other inline elements.

---

### 3. `inline-block`

- Combines the characteristics of both `inline` and `block`.
- The element flows inline but behaves like a block in terms of width and height, allowing you to set dimensions.

**Example:**

```html
<span style="display: inline-block; width: 100px; height: 50px; background:
lightblue;">I am inline-block</span>
```

**Visual Behavior:** Respects set width and height while flowing inline.

---

### 4. `flex`

- The element becomes a flex container, enabling the use of flexbox for layout.
- Children of this element are treated as flex items, and their layout is controlled by flexbox properties.

**Example:**

```html
<div style="display: flex;">
  <div>Flex Item 1</div>
  <div>Flex Item 2</div>
</div>
```

**Visual Behavior:** Children are laid out in a flexible, customizable manner (default: horizontal row).

---

### 5. `grid`

- The element becomes a grid container, enabling the use of CSS grid layout.
- Children are laid out in rows and columns based on the grid structure.

**Example:**

```
<div style="display: grid; grid-template-columns: repeat(3, 1fr);">
  <div>Grid Item 1</div>
  <div>Grid Item 2</div>
  <div>Grid Item 3</div>
</div>
```

**Visual Behavior:** Children are aligned into a grid with three equal-width columns.

---

## 6. `none`

- The element is hidden and does not occupy any space in the layout.

**Example:**

```
<div style="display: none;">You can't see me!</div>
```

**Visual Behavior:** The element is completely removed from the visual flow.

---

## 7. `inline-flex`

- Similar to `flex`, but the element behaves as an inline container.
- Useful when you want to apply flexbox to inline elements.

**Example:**

```
<div style="display: inline-flex;">
  <div>Inline Flex 1</div>
  <div>Inline Flex 2</div>
</div>
```

**Visual Behavior:** Flex items are laid out inline with other content.

---

### 8. `table`

- The element behaves like a table (similar to the HTML `<table>` element).
- Useful for creating custom table layouts using CSS.

**Example:**

```html
<div style="display: table;">
  <div style="display: table-row;">
    <div style="display: table-cell;">Cell 1</div>
    <div style="display: table-cell;">Cell 2</div>
  </div>
</div>
```

**Visual Behavior:** Resembles the behavior of an HTML table.

---

### 9. `inline-table`

- Behaves like `table`, but flows inline with text and other inline elements.

**Example:**

```html
<div style="display: inline-table;">
  <div style="display: table-row;">
    <div style="display: table-cell;">Cell 1</div>
    <div style="display: table-cell;">Cell 2</div>
  </div>
</div>
```

---

## Comparison Chart

| Value | Behavior |
|---|---|
| `block` | Takes full width, starts on a new line. |
| `inline` | Takes only as much width as necessary, flows inline with text. |
| `inline-block` | Combines `inline` and `block`: respects width/height, flows inline. |
| `flex` | Creates a flex container for flexible layouts. |

| Value | Behavior |
|---|---|
| `grid` | Creates a grid container for row-column layouts. |
| `none` | Hides the element completely (no layout space). |
| `inline-flex` | Creates an inline container with flex behavior. |
| `table` | Behaves like an HTML `<table>`. |
| `inline-table` | Behaves like an HTML `<table>`, but flows inline with text. |

---

## Interactive Example (HTML + CSS)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Display Property</title>
  <style>
    .container {
      margin-bottom: 20px;
    }
    .block {
      display: block;
      background: lightblue;
      padding: 10px;
    }
    .inline {
      display: inline;
      background: lightgreen;
      padding: 5px;
    }
    .inline-block {
      display: inline-block;
      background: lightcoral;
      padding: 10px;
      width: 100px;
      height: 50px;
    }
    .flex {
      display: flex;
      gap: 10px;
      background: lightgray;
      padding: 10px;
    }
    .grid {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      gap: 10px;
      background: lightyellow;
      padding: 10px;
    }
    .hidden {
      display: none;
    }
  </style>
</head>
<body>
  <div class="container block">I am a block element</div>

  <span class="container inline">I am an inline element</span>
  <span class="container inline">I flow with text</span>

  <div class="container inline-block">I am inline-block</div>

  <div class="container flex">
    <div>Flex Item 1</div>
    <div>Flex Item 2</div>
    <div>Flex Item 3</div>
  </div>

  <div class="container grid">
    <div>Grid Item 1</div>
    <div>Grid Item 2</div>
    <div>Grid Item 3</div>
  </div>

  <div class="container hidden">You cannot see me!</div>
</body>
</html>
```

Try this code in your browser to experiment with different `display` values!