CSS Display Property – Complete Guide with Examples

The `display` property in CSS defines how elements are rendered on the webpage. It determines whether an element appears as a block, inline, flex, grid, or is hidden.

---

1. display: block

- Elements take up the full width of their container.
- Starts on a new line.
- Examples: `<div>`, `<p>`, `<h1>`-`<h6>`, `<section>`, `<article>`.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Display Block</title>
    <style>
        div {
            display: block;
            width: 200px;
            height: 100px;
            background-color: lightblue;
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
```

```
    <div>Block Element 1</div>
    <div>Block Element 2</div>
</body>
</html>
```

---

2. display: inline

- Elements do not start on a new line.
- Takes up only as much width as necessary.
- Examples: `<span>`, `<a>`, `<strong>`.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Display Inline</title>
```

```
    <style>
        span {
            display: inline;
            background-color: yellow;
            padding: 5px;
        }
    </style>
</head>
<body>
    <p>This is an <span>inline element</span> inside a paragraph.</p>
</body>
</html>
```

---

3. display: inline-block

- Behaves like `inline`, but width and height can be set.
- Useful for buttons, images, or inline-sized elements.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Display Inline-Block</title>
    <style>
        .box {
            display: inline-block;
            width: 100px;
            height: 50px;
            background-color: lightcoral;
            margin: 5px;
        }
    </style>
</head>
<body>
```

```
    <div class="box">Box 1</div>
    <div class="box">Box 2</div>
</body>
</html>
```

---

4. display: none

- Completely hides an element from the webpage.
- The element does not take up space.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Display None</title>
    <style>
        .hidden {
            display: none;
        }
    </style>
</head>
<body>
    <p>This paragraph is visible.</p>
    <p class="hidden">This paragraph is hidden.</p>
</body>
</html>
```

---

5. display: flex

- Enables a flexible box layout.

- Useful for arranging elements in rows or columns.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Display Flex</title>
    <style>
        .container {
            display: flex;
            justify-content: space-around;
            align-items: center;
            height: 200px;
            background-color: lightgray;
        }
        .item {
            width: 100px;
            height: 50px;
            background-color: steelblue;
            color: white;
            text-align: center;
            line-height: 50px;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
        <div class="item">Item 3</div>
    </div>
</body>
</html>
```

---

6. display: grid

- Enables CSS Grid layout for two-dimensional designs.
- Defines rows and columns easily.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Display Grid</title>
    <style>
        .grid-container {
            display: grid;
            grid-template-columns: repeat(3, 1fr);
            gap: 10px;
            background-color: lightgray;
            padding: 10px;
        }
        .grid-item {
            background-color: steelblue;
            color: white;
            padding: 20px;
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="grid-container">
        <div class="grid-item">1</div>
        <div class="grid-item">2</div>
        <div class="grid-item">3</div>
    </div>
</body>
</html>
```

7. display: table

- Makes an element behave like a table.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Display Table</title>
    <style>
        .table {
            display: table;
            width: 100%;
            background-color: lightgray;
        }
        .row {
            display: table-row;
        }
        .cell {
            display: table-cell;
            padding: 10px;
            border: 1px solid black;
        }
    </style>
</head>
<body>
    <div class="table">
        <div class="row">
            <div class="cell">Cell 1</div>
            <div class="cell">Cell 2</div>
        </div>
    </div>
```

```
</body>
</html>
```

---

8. display: list-item

- Used for list elements like `<li>`.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Display List Item</title>
    <style>
        .custom-list {
            display: list-item;
            list-style-type: square;
        }
    </style>
</head>
<body>
    <ul>
        <li class="custom-list">Item 1</li>
        <li class="custom-list">Item 2</li>
    </ul>
</body>
</html>
```

Conclusion

- **Use `block`** for full-width elements.

- **Use `inline` for elements inside text.**
- **Use `inline-block` for inline elements that need width/height.**
- **Use `none` to hide elements.**
- **Use `flex` for flexible layouts.**
- **Use `grid` for structured layouts.**
- **Use `table` to create table-like structures.**