4.0 Display: block, inline, inline-block, float and clear

4.0.0 Display: block

- Block-level elements always start on a new line and take up the full available width of their parent container.
- They respect width, height, margin, and padding properties.
- Examples include div, p, h1, ul, li.

Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <h2>display: block (Headings/Paragraphs)</h2>
    <h1 style="display: block; background: lightblue">
      Amazon - Online Shopping
    </h1>
    <p style="display: block; background: lightgreen">
      This is a full-width paragraph like a blog or product description.
    </p>
  </body>
</html>
```

Output:

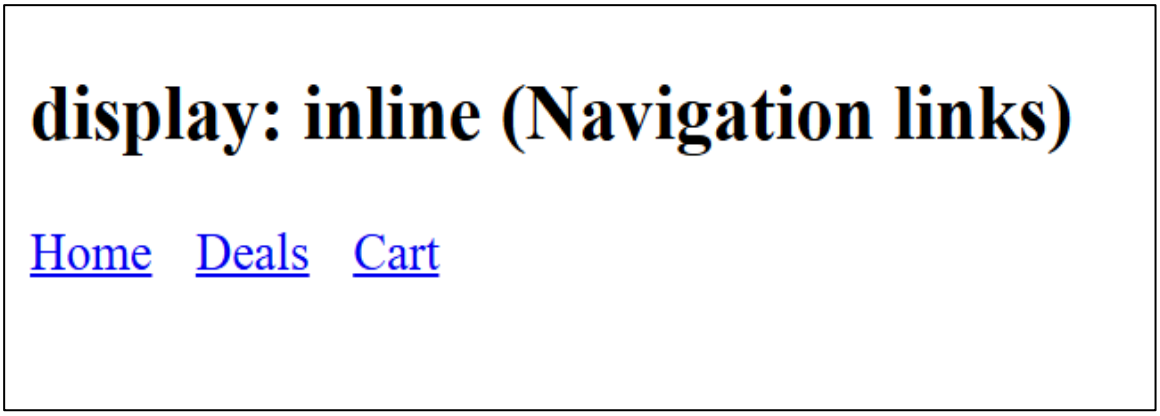
4.0.1 display: inline

- Inline elements do not start on a new line and only take up as much width as their content requires.
- They do not respect width and height properties. Top and bottom margin and padding are applied but may visually overlap with surrounding content.
- Examples include span, a, strong, em.

Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <h2>display: inline (Navigation links)</h2>
<a href="#" style="display:inline; margin-right:10px;">Home</a>
<a href="#" style="display:inline; margin-right:10px;">Deals</a>
<a href="#" style="display:inline; margin-right:10px;">Cart</a>
  </body>
</html>
```

Output:



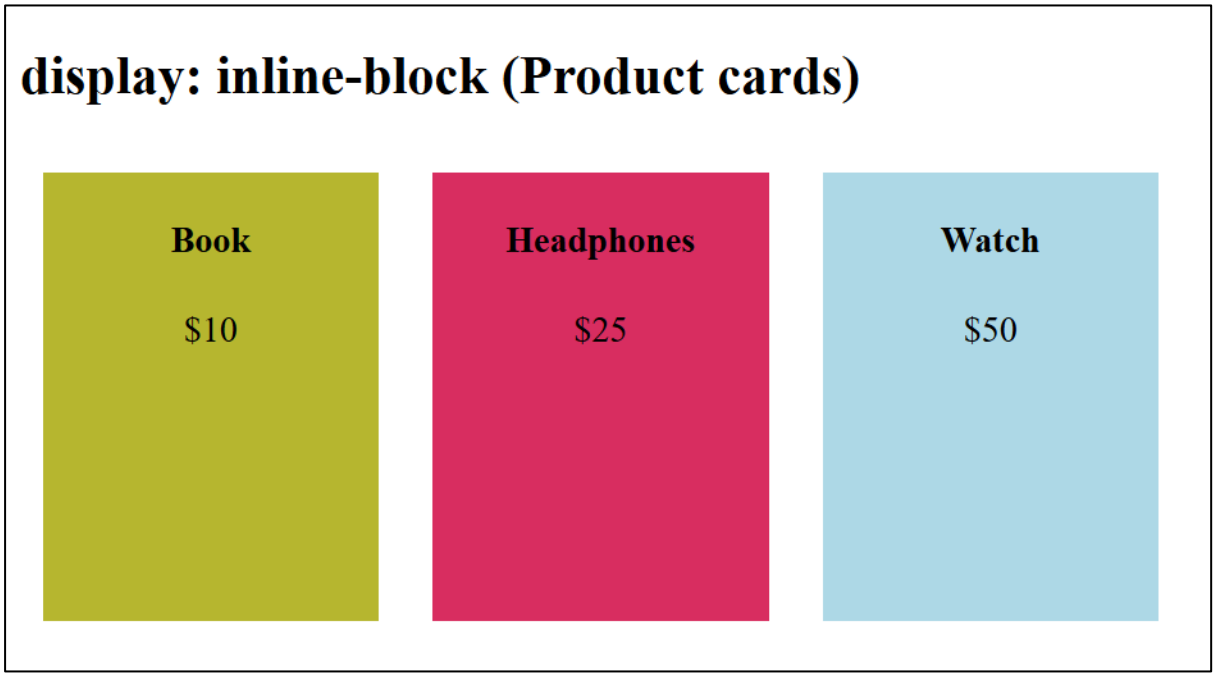
4.0.2 display: inline-block

- Inline-block elements combine characteristics of both inline and block elements.
- They do not start on a new line, allowing other elements to sit alongside them.
- They respect width, height, margin, and padding properties, behaving like block elements in this regard.
- This display value is often used for creating grid-like layouts where elements need to sit side-by-side but also require specific dimensions.

Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
  <h2>display: inline-block (Product cards)</h2>
<div style="display:inline-block; width:150px; height:200px;
background:rgb(182, 182, 47); margin:10px; text-align:center;">
  <h4>Book</h4>
  <p>$10</p>
</div>
<div style="display:inline-block;
background:rgb(216, 45, 96); margin:10px; text-align:center;">
     <h4>Headphones</h4>
  <p>$25</p>
</div>
<div style="display:inline-block; width:150px; height:200px;
background:lightblue; margin:10px; text-align:center;">
  <h4>Watch</h4>
  <p>$50</p>
</div>
</html>
```

Output:



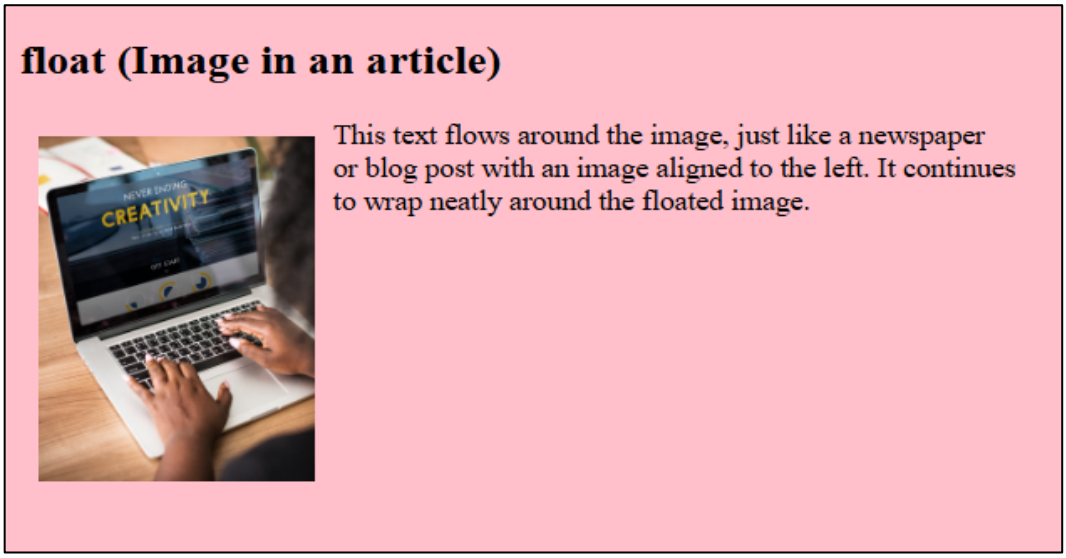
4.0.3 float

- The float property takes an element out of the normal document flow and positions it to the left or right within its containing element.
- Content (like text) will wrap around the floated element.
- Commonly used for image and text wrapping or creating multi-column layouts, though modern CSS layout methods like Flexbox and Grid are often preferred for complex layouts.
- Values include left, right, none.

Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body bgcolor="pink">
  <h2>float (Image in an article)</h2>
<img src="woman-working-laptop.jpg" style="float:left;width:150px;
height:200px;  margin:10px;">
<p>
This text flows around the image, just like a newspaper <br>or blog post with an
image aligned to the left.
It continues <br>to wrap neatly around the floated image.
</p>
  </body>
</html>
```

Output:



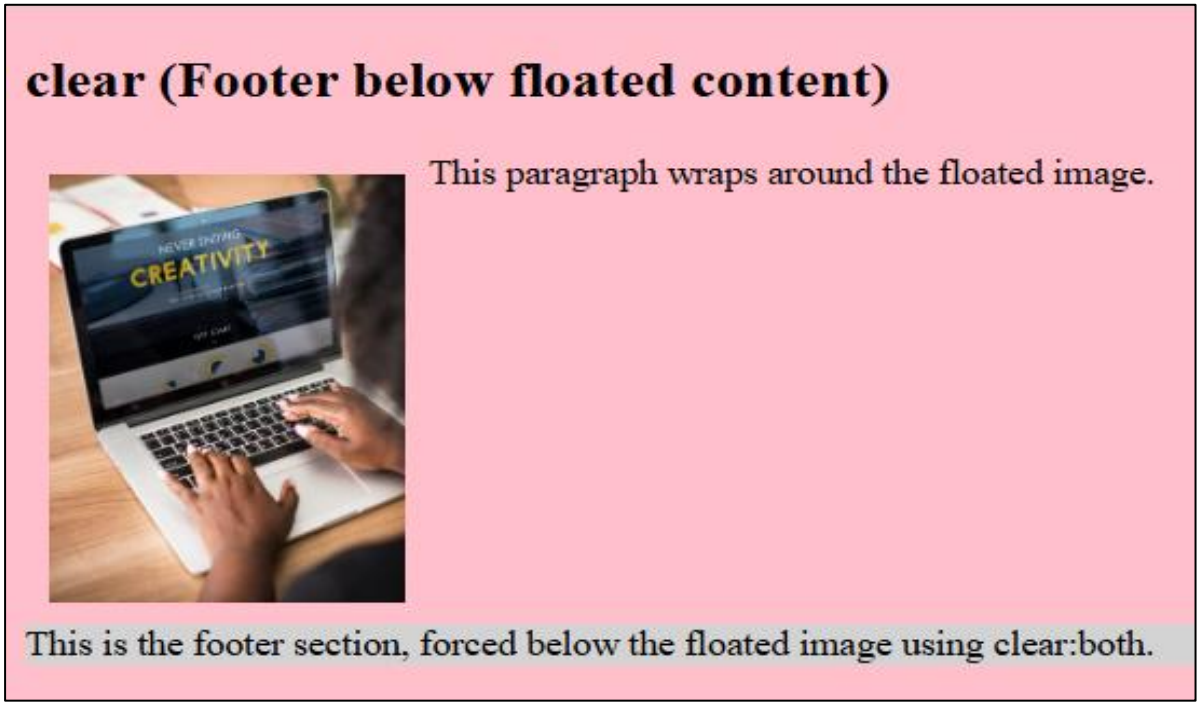
4.0.4 clear

- The clear property is used in conjunction with float to control the positioning of elements that follow a floated element.
- It prevents an element from sitting alongside a floated element on the specified side.
- Values include left (clears floats on the left), right (clears floats on the right), both (clears floats on both sides), and none (default, allows floats on both sides).
- Often used to prevent content from wrapping around a floated element or to ensure a container expands to encompass its floated children (e.g., using a "clearfix" hack).

Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
    <title>Document</title>
  </head>
  <body bgcolor="pink">
  <h2>clear (Footer below floated content)</h2>
<img src="woman-working-laptop.jpg" style="float:left;width:150px;
height:200px; margin:10px;">
<p>This paragraph wraps around the floated image.</p>
<p style="clear:both; background:lightgray;">This is the footer section,
forced below the floated image using clear:both.</p>
  </body>
</html>
```

Output:



4.1 Introduction to Flexbox: justify-content, align-items, flex-direction

Flexbox

Flexbox (Flexible Box Layout) is a modern CSS3 layout system designed to align, distribute, and organize elements in a container efficiently. It is extremely useful for creating responsive web applications and solving alignment issues that older techniques (like floats and inline-block) could not handle.

4.1.0 Introduction to Flexbox

Flexbox works on a parent-child relationship where the parent (container) uses flex properties, and the children (items) adjust themselves accordingly. Flexbox provides control in two axes:

- Main Axis – controlled by flex-direction and justify-content.
- Cross Axis – controlled by align-items.

Example:

```
.container {
  display: flex; /* flexbox activate */
}
```

4.1.1 flex-direction

The 'flex-direction' property defines the direction of the main axis of the flex container.

Values:

- **row (default):** Items placed left to right.
- **row-reverse:** Items placed right to left.
- **column:** Items placed top to bottom.
- **column-reverse:** Items placed bottom to top.

Example:

```
.container {
  display: flex;
  flex-direction: column;
}
```

4.1.2 justify-content

The 'justify-content' property aligns flex items along the main axis (horizontal for row, vertical for column).

Values:

- **flex-start:** Items **aligned at the start.**
- **flex-end**: Items aligned at the end.
- **center:** Items aligned at the center.
- **space-between:** Equal space between items; first and last at edges.
- **space-around:** Equal space around items with half space at edges.
- **space-evenly:** Equal space between all items including edges.

Example:

```
.container {
 display: flex;
 justify-content: space-between;
}
```

4.1.3 align-items

The 'align-items' property aligns items along the cross axis (perpendicular to main axis).

Values:

- **stretch (default):** Items stretch to fill container.
- **flex-start:** Items align to start of cross axis.
- **flex-end:** Items align to end of cross axis.
- **center:** Items aligned at center of cross axis.
- **baseline:** Items align based on text baseline.

Example:

```
.container {
 display: flex;
 justify-content: space-between;
}
```