Lesson:

Display property and Normal Flow

PW SKILLS

Topics Covered

- Normal Flow
- Display Property

Normal Layout

The **Normal or Flow Layout** refers to the initial arrangement of Block and Inline elements on a page, prior to any modifications made to their layout.

When elements are arranged in Flow Layout, inline elements appear in the same direction, and block elements, on the other hand, appear consecutively.

This layout method does not require any special CSS properties and is best suited for simple web pages with a linear structure.

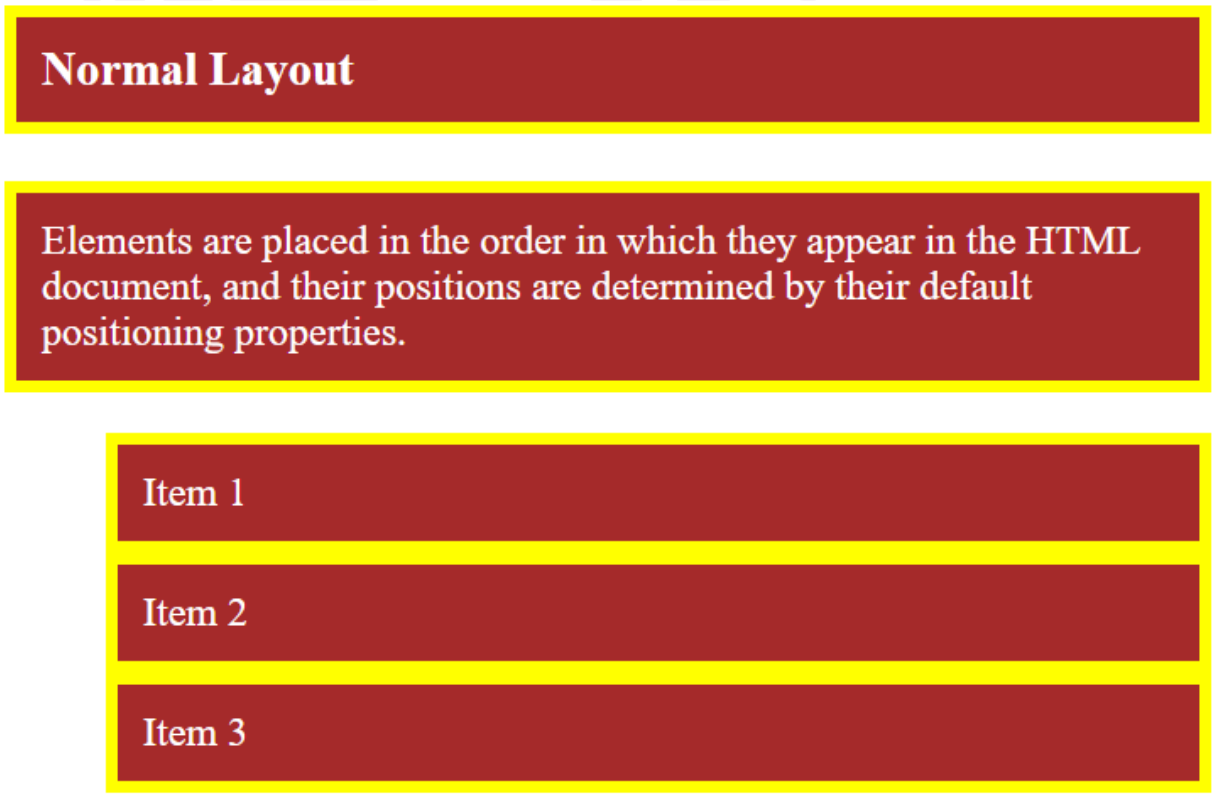
Example:-
index.html

```
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Unordered Lists</title>
    <style>
       h3,p,li {
        background-color: brown;
           color: white;
           border: 5px solid yellow;
           padding: 10px;

       }
    </style>
  </head>
  <body>
    <h3>
      Normal Layout
    </h3>
```

```
<p>
      Elements are placed in the order in which they
appear in the HTML document, and their positions are
determined by their default positioning properties.
    </p>
    <ol>
      <li>
        Item 1
      </li>
      <li>
        Item 2
      </li>
      <li>
        Item 3
      </li>
    </ol>
  </body>
</html>
```

Browser Output:-



In the above example, we have not modified Layout, so elements are arranged according to the normal flow.

We can alter Normal Layout behaviour by changing the display property of elements.

Display Property
We can alter the default layout, to some another layout options available using display property as follows,

There are two kinds of display types,

1. **Outer :** The outer display type of an element, can be **inline, block, inline-block, none,** which determines its role in flow layout.
2. **Inner :** Inner display type defines how the children elements will be arranged within the container element.

Block

Creates a block level element, generating line breaks before and after the element in Normal Flow.

CSS Syntax-

```
display: block;
```

Properties of Block Elements

- It starts onto a new line.
- The dimensions set by the width and height properties will be applied.
- Padding, margin, and border will influence the positioning of neighbouring elements, causing them to be displaced away from the box.
- If the width is not explicitly defined, the box will expand horizontally in the direction of the text flow to occupy the available space within its container. Generally, it will stretch to match the width of its container, occupying 100% of the available space.

Example

```
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Block Elements</title>
    <style>
      .item {
        display: block; /* Div elements are by default block*/
        background-color: brown;
        color: white;
```

```
 border: 5px solid yellow;
        padding: 10px;
      }

      .item2 {
        height: 100px;
      }
    </style>
  </head>
  <body>
    <div>
      <div class="item1 item">Item 1</div>
      <div class="item2 item">Item 2</div>
      <div class="item3 item">Item 3</div>
    </div>
  </body>
</html>
```

Browser Output:-

In above example, all items are **block** elements thats why, each item is onto a new line and each element takes the whole width available.

Inline
Creates an inline level element that does not generate line breaks, and starts on the same line in Normal Flow.

CSS Syntax

```
display: inline;
```

Properties of Inline Elements

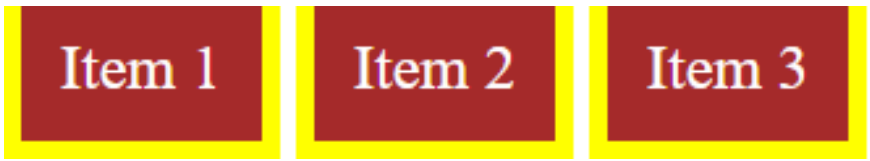
- The box will remain on the same line without wrapping onto a new line.
- The width and height properties will not have any effect on the box.
- Vertical Padding, margins, and borders won't take effect.
- Horizontal, padding, margins, and borders will be effective.

Example-

```
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Inline Elements</title>
    <style>
      .item {
        display: inline;
        background-color: brown;
        color: white;
        border: 5px solid yellow;
        padding: 10px;
      }

      .item2 {
        height: 100px;
      }
    </style>
  </head>
  <body>
    <div>
      <div class="item1 item">Item 1</div>
      <div class="item2 item">Item 2</div>
      <div class="item3 item">Item 3</div>
    </div>
  </body>
</html>
```

Browser Output:-



In the above example, we are making div elements **inline** explicitly that's why all elements are aligned on the same line taking only required space.

inline-block

The display inline-block property in CSS is used to make an element behave like an inline element while retaining some features of a block-level element. It combines characteristics of both inline and block elements.

- Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.
- Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.
- Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

CSS Syntax

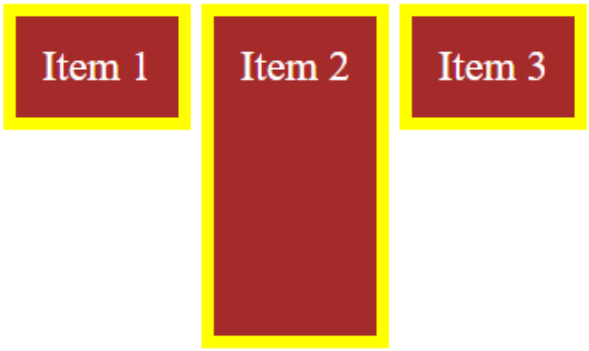
```
display: inline-block;
```

Example-

```
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Inline-Block Elements</title>
    <style>
      .item {
        display: inline-block;
        background-color: brown;
        color: white;
        border: 5px solid yellow;
        padding: 10px;
      }

      .item2 {
        height: 100px;
      }
    </style>
  </head>
  <body>
    <div>
      <div class="item1 item">Item 1</div>
      <div class="item2 item">Item 2</div>
      <div class="item3 item">Item 3</div>
    </div>
  </body>
</html>
```

Browser Output:-



In the above example, we have changed **item2** height. If it was a pure **inline** element, then we will not be able to set the **height** of the element. Since the element is **inline-block**, that's why we are able to use **block** level element properties.

none

The display **inline-block** property in CSS is used to make an element invisible, and the element will not take any space in the document.

CSS Syntax

```
display: none;
```

Similarly, we have other methods by which we can change layouts.

```
.item {
    display: none;
    background-color: brown;
    color: white;
    border: 5px solid yellow;
    padding: 10px;
}
```

Browser Output:-

If you observe, the item is not visible.

One question arises in mind: the same task we can accomplish with **opacity:0** and **visibility:hidden**, so what's the difference?

opacity:0	visibility:hidden	display:none
Element remains in document flow	Element remains in document flow	Element is removed from document flow
Occupies space	Occupies space	Does not occupy space
Content is still accessible to assistive technologies	Content is still accessible to assistive technologies	Content is not accessible to assistive technologies

Example: opacity:0 or visibility: hidden

```
<style>
 img {
     opacity: 0;
 }
</style>
<body>
   <p>Text before image</p>
   <img src="./assets/photo.jpg" width="500px" />
   <p>Text after image</p>
</body>
```

Browser Output:-

Text before image

Text after image

Example: display: none

```
<style>
 img {
     display: none;
 }
</style>
<body>
   <p>Text before image</p>
   <img src="./assets/photo.jpg" width="500px" />
   <p>Text after image</p>
</body>
```

Browser Output:-

Text before image

Text after image

If you observe, output incase of **visibility:hidden** and **opacity:0** is same, but incase of **display:none**, it is different because image does not take space in document.