# HTML Layouts

HTML layouts provide a way to arrange web pages in well-mannered, well-structured, and in responsive form or we can say that HTML layout specifies a way in which the web pages can be arranged. Web-page layout works with arrangement of visual elements of an HTML document.



These are define the layout

* <header>: It is used to define a header for a document or a section.
* <nav>: It is used to define a container for navigation links
* <section>: It is used to define a section in a document
* <article>: It is used to define an independent self-contained article
* <aside>: It is used to define content aside from the content (like a sidebar)
* <footer>: It is used to define a footer for a document or a section
* <details>: It is used to define additional details
* <summary>: It is used to define a heading for the <details> element

**First we have to understand flow of web page.**

**Basic document or Normal flow of web page.**

When we display some content with the help of html element and without using CSS.

In that case our content is display on browse in normal flow.

If you want to change the normal flow of web site we have three CSS properties

* **Display**
* **Float**
* **Position**

1. **The display Property**

The display property specifies if/how an element is displayed.

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Block-level Elements follow some properties

* A block-level element always starts on a new line
* A block-level element takes always difficult width of view port extreme left to right.
* A block-level element accept custom width and height.
* A block-level element accept padding and margin.

Examples of block-level elements:

* <div>
* <h1> - <h6>
* <p>
* <form>
* <header>
* <footer>
* <section>

Inline Elements follow some properties

* An inline element try to display content in a single line.
* A inline element takes always difficult width according to content width .
* A inline element not accept custom width and height.
* A inline element not accept custom margin but custom padding can apply.

Examples of inline elements:

* <span>
* <a>
* <img>

Conversation is possible with the help of display properties.

* **block-level element to Inline element**
* **Inline element to block-level element**
* *<!*DOCTYPE html*>*
* *<*html lang="en"*>*
* *<*head*>*
* *<*meta charset="UTF-8"*>*
* *<*meta name="viewport" content="width=device-width, initial-scale=1.0"*>*
* *<*title*>*Document*</*title*>*
* *<*style*>*
* Span{
* Display: block;
* }
* div{
* Display: inline;
* Display: inline-block;
* }
* *</*style*>*
* *</*head*>*
* *<*body*>*
* *<*span*> convert inline to block*  *</*span*>*
* *<*div*>* *convert block to inline*  *</*div*>*
* *</*body*>*
* *<*/html*>*

**We can combine both inline and block-level elements with the help of inline-block properties**

**Inline-block properties work as a inline element but we can set custom width \ height and padding\margin.**

Display: none;

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them.

# The position Property

The **position** [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) property is used to specify how an element is positioned in a document.

There are five different position values and four different position properties ( Top, Right, Bottom and Left)

**value**

* static

The CSS position values decide how an element is positioned in a document. And the CSS position properties determine the final location of positioned elements.

* Relative
* absolute
* fixed
* sticky

1. **position: static** **:-** HTML elements are positioned static by default .Static positioned elements are not affected by the top, bottom, left, right and Z-Index properties.
2. **position: relative** **:-**We can positioned any element corresponding to the current window using top, right, bottom and left properties. And these properties determine the final location of positioned elements.

Note

When we positioned any element with the help of relative value old positions are maintained.

1. **position: absolute** **:-** The element is positioned relative to its closest positioned ancestor using top, right, bottom and left properties. And these properties determine the final location of positioned elements.

Note

When we positioned any element with the help of absolute value old positions are not maintained.

1. **position: fixed** **:-** it always stays in the same place even if the page is scrolled. this element is stack till the range of window size. The top, right, bottom, and left properties are used to position the element.
2. **position: sticky** **:-** It is used to stack any element . this element is stack till the range of parent container