Introduction to CSS layout

Overview: CSS layout Next

This lesson recaps some of the CSS layout features we've already touched upon in previous modules, such as different `display` values, as well as introducing some of the concepts we'll be covering throughout this module. It also covers the concept of normal flow in depth.

Prerequisites:	Structuring content with HTML, CSS Styling basics, Fundamental text and font styling.
Learning outcomes:	• Recognize the methods used to implement modern page layouts. • Understand that normal flow is the default way a browser lays out block and inline content. • Know that properties such as `display`, `float`, and `position` are intended to change how the browser lays out content.

CSS page layout techniques allow us to take elements contained in a web page and control where they're positioned relative to the following factors: their default position in normal layout flow, the other elements around them, their parent container, and the main viewport/window.

The page layout techniques we'll mention below and cover in detail through the module have their uses, advantages, and disadvantages. No technique is designed to be used in isolation. By understanding what each layout method is designed for you'll be in a good position to understand which method is most appropriate for each task.

Normal layout flow

Elements on a webpage lay out in **normal flow** if you haven't applied any CSS to change the way they behave. You can change how elements behave either by adjusting their position in normal flow or by removing them from it altogether. Starting with a solid, well-structured document that's readable in normal flow is the best way to begin any webpage. It ensures that your content is readable even if the user's using a very limited browser or a device such as a screen reader that reads out the content of the page. In addition, since normal flow is designed to make a readable document, by starting in this way you're working with the document rather than struggling *against* it as you make changes to the layout.

Before digging deeper into different layout methods, it's worth revisiting some of the things you have studied in previous modules with regard to normal document flow.

How are elements laid out by default?

The process begins as the boxes of individual elements are laid out in such a way that any padding, border, or margin they happen to have is added to their content. This is what we call the **box model**.

By default, a block-level element's content fills the available inline space of the parent element containing it, growing along the block dimension to accommodate its content. The size of inline-level elements is just the size of their content. You can set `width` or `height` on some elements that have a default `display` property value of `inline`, like `<img>`, but the `display` value will still remain `inline`.

If you want to control the `display` property of an inline-level element in this manner, use CSS to set it to behave like a block-level element (e.g., with `display: block;` or `display: inline-block;`, which mixes characteristics from both).

That explains how elements are structured individually, but how about the way they're structured when they interact with one another? The normal layout flow (mentioned in the layout introduction article) is the system by which elements are placed inside the browser's viewport. By default, block-level elements are laid out in the *block flow direction*, which is based on the parent's writing mode (initial: horizontal-tb). Each element will appear on a new line below the last one, with each one separated by whatever margin that's been specified. In English, for example, (or any other horizontal, top to bottom writing mode) block-level elements are laid out vertically.

Inline elements behave differently. They don't appear on new lines; instead, they all sit on the same line along with any adjacent (or wrapped) text content as long as there is space for them to do so inside the width of the parent block-level element. If there isn't space, then the overflowing content will move down to a new line.

If two vertically adjacent elements both have a margin set on them and their margins touch, the larger of the two margins remains and the smaller one disappears. This is known as **margin collapsing**. Collapsing margins is only relevant in the **vertical direction**.

Normal flow example

Let's look at a simple example that explains all of this:

HTML Copy Play

```
<h1>Basic document flow</h1>

<p>
  I am a basic block level element. My adjacent block level elements
sit on new
  lines below me.
</p>

<p>
  By default we span 100% of the width of our parent element, and we
are as tall
  as our child content. Our total width and height is our content +
padding +
  border width/height.
</p>

<p>
  We are separated by our margins. Because of margin collapsing, we are
  separated by the size of one of our margins, not both.
</p>

<p>
  Inline elements <span>like this one</span> and <span>this one</span>
sit on
  the same line along with adjacent text nodes, if there is space on
the same
  line. Overflowing inline elements will
  <span>wrap onto a new line if possible (like this one containing
text)</span>,
  or just go on to a new line if not, much like this image will do:
  <img
    src="https://mdn.github.io/shared-assets/images/examples/long.jpg"
    alt="snippet of cloth" />
</p>
```

CSS Copy Play

```
body {
  width: 500px;
  margin: 0 auto;
}

p {
  background: rgb(255 84 104 / 30%);
  border: 2px solid rgb(255 84 104);
  padding: 10px;
  margin: 10px;
}

span {
  background: white;
  border: 1px solid black;
}
```

Play

Note how the HTML is displayed in the exact order in which it appears in the source code, with block elements stacked on top of one another.

For many of the elements on your page, the normal flow will create exactly the layout you need. However, for more complex layouts you will need to alter this default behavior using some of the tools available to you in CSS. Starting with a well-structured HTML document is very important because you can then work with the way things are laid out by default rather than fighting against it.

Overriding normal flow

The methods that can override normal flow and change how elements are laid out in CSS, which we will cover in detail in this module, are:

The `display` property

Standard values such as `block`, `inline` or `inline-block` can change how elements behave in normal flow, for example, by making a block-level element behave like an inline-level element (we covered these back in the Box model lesson).

Floats

Applying a `float` value such as `left` can cause block-level elements to wrap along one side of an element, like the way images sometimes have text floating around them in magazine layouts.

Positioning

The `position` property allows you to precisely control the placement of boxes inside other boxes. `static` positioning is the default in normal flow, but you can cause elements to be laid out differently using other values, for example, fixing them to the top of the browser viewport using `position: fixed`.

Specific layout systems accessed through `display`

We also have entire layout methods that are enabled via specific `display` values. The most important ones for you to know about are CSS grid and Flexbox, which both alter how child elements are laid out inside their parents.

Responsive design

Responsive design refers to creating layouts that adapt to different devices the web page is rendered on (for example, desktops and mobile phones). Responsive design doesn't provide any specific layout tools of its own; its most significant component is the `@media` at-rule, which allows you to apply different layouts depending on device attributes such as screen width or resolution.

Other layout techniques

There are other layout techniques that are less commonly used, which we won't cover in this module:

Table layout

Features designed for styling parts of an HTML table can be used on non-table elements using `display: table` and associated properties.

Multi-column layout

The multi-column layout properties can cause the content of a block to lay out in columns, as you might see in a newspaper.

Summary

This article has provided a brief summary of all the layout technologies you should know about at this point in your learning! Read on for more information on each individual technology. Next up, we'll look at floats.

Overview: CSS layout Next

This page was last modified on Apr 28, 2025 by MDN contributors.