CSS layout

- Prerequisites
- Tutorials and challenges
- Test your skills
- Additional tutorials
- See also

{{NextMenu("Learn_web_development/Core/CSS_layout/Introduction", "Learn_web_development/Core")}}

In previous modules we looked at how to style and manipulate the boxes that your content sits inside. Now it's time to look at how to correctly lay out your boxes in relation to one another, and the browser viewport. This module looks at floats, positioning, other modern layout tools, and building responsive designs that will adapt to different devices, screen sizes, and resolutions.

Prerequisites

Before starting this module, you should be familiar with HTML, the basic fundamentals of CSS, and CSS text styling.

> [!NOTE] If you are working on a computer, tablet, or another device where you can't create files, you can try running the code in an online editor such as CodePen or JSFiddle.

Tutorials and challenges

- Introduction to CSS layout

- : This lesson recaps some of the CSS layout features we've already touched upon in previous modules, such as different {{cssxref("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.

- Floats
  - : Originally for floating images inside blocks of text, the {{cssxref("float")}} property became one of the most commonly used tools for creating multiple column layouts on webpages. With the advent of flexbox and grid it has now returned to its original purpose, as this article explains.
- Positioning
  - : Positioning allows you to take elements out of normal document flow and make them behave differently, for example, sitting on top of one another or always remaining in the same place inside the browser viewport. This article explains the different {{cssxref("position")}} values and how to use them.
- Flexbox
  - : Flexbox is a one-dimensional layout method for laying out items in rows or columns. Items flex to fill additional space and shrink to fit into smaller spaces. This article explains all the fundamentals.
- CSS grid layout
  - : CSS grid layout is a two-dimensional layout system for the web. It lets you organize content into rows and columns and offers many features to simplify the creation of complex layouts. This article will explain all you need to know to get started with grid layout.
- Fundamental layout comprehension Challenge
  - : This challenge will test your knowledge of the layout features we've covered so far in the module, namely flexbox, floats, grid, and positioning. By the end you will have developed a webpage layout using a variety of techniques.
- Responsive design
  - : As more diverse screen sizes have appeared on web-enabled devices, the concept of responsive web design (RWD) has appeared: a set of practices that allows web pages to alter their layout and appearance to suit different screen widths, resolutions, etc. It is an idea that changed the way we design for a multi-device web, and in this article we'll help you understand the main techniques you need to know to master it.
- Media query fundamentals
  - : The **CSS Media Query** gives you a way to apply CSS only when the browser and device environment matches rules that you specify. Media queries are a key part of responsive web design because they allow you to create different layouts depending on the size of the viewport. In this lesson, you will learn about the syntax used in media queries then use them in an interactive example showing how a simple design might be made responsive.

Test your skills

You will find "Test your skills" articles placed between the tutorial articles to check whether you have retained the most important information before you move on. If you want to explore all of these together, you can find them listed at Test your skills: CSS layout.

Additional tutorials

These tutorials are not part of the learning pathway, but they are interesting nonetheless — you should consider these as stretch goals, to optionally study when you are done with the main Core articles.

- Multiple-column layout
  - : The multiple-column layout specification provides you with a method for laying content out in columns, as you might see in a newspaper. This article explains how to use this feature.
- Practical positioning examples
  - : This article shows how to build some real-world examples to illustrate what kinds of things you can do with positioning.
- Legacy layout methods
  - : Grid systems are a very common feature used in CSS layouts, and before CSS grid layout they tended to be implemented using floats or other layout features. You imagine your layout as a set number of columns (e.g., 4, 6, or 12), and then fit your content columns inside these imaginary columns. In this article we'll explore how these older methods work, in order that you understand how they were used if you work on an older project.
- Supporting older browsers
  - : Visitors to your website may include users who either use older browsers or use browsers that do not support the CSS features you've implemented. This is a common scenario on the web, where new features are continuously being added to CSS. Browsers differ in their support for these features because different browsers tend to prioritize implementing different features. This article explains how you as a web developer can use modern web techniques to ensure that your website remains accessible to users with older technology.

See also

- CSS layout cookbook

    - : The CSS layout cookbook aims to bring together recipes for common layout patterns, things you might need to implement in your sites. In addition to providing code you can use as a starting point in your projects, these recipes highlight the different ways layout specifications can be used and the choices you can make as a developer.
- Learn Flexbox and Learn CSS Grid *MDN learning partner*
    - : These courses from Scrimba provide interactive lessons teaching you all you need to know about Flexbox and Grid.

{{NextMenu("Learn_web_development/Core/CSS_layout/Introduction", "Learn_web_development/Core")}}

□□□ #2
□ □□□ □□□ 2026-10-31 11:27:53 UTC
□ □□□ □□□ 2026-10-31 11:27:53 UTC