Relative positioning allows you to shift the generated box in one or two dimensions. The element still occupies the space as if it was static, but the generated box can be shifted to another position. Relative positioning is useful in combination with floats to create layouts where the presentational order differs from the source order.



Two options exist: floats or absolute positioning. Both methods have their advantages and drawbacks, but if you want a full-width footer and don’t know in advance which column will be the longest, then floats are necessary to ensure the integrity of your design.

The problem with floats is that they only shift to the left or right until they touch the edge of the parent block, or another float. That means floated columns have to appear in the right order in your markup. But sometimes it’s desirable to have a presentational order that is different from the source order. You may want to have the content before the navigation, for instance, to enhance usability for keyboard navigation and to improve search engine optimization. This is possible to achieve, even with floats, with some judicious use of negative margins and relative positioning—let’s have a look at how to do this. Let’s begin with a skeleton, or wireframe, HTML document.

1. Copy the code below into your text editor and save the file as layout.html.

<title>Static and Relative Positioning</title>

<link rel="stylesheet" type="text/css" href="layout.css">

</head>

<body>

<div id="header">Header</div>

<div id="main">Main content</div>

<div id="sidebar">Sidebar</div>

<div id="nav">Navigation</div>

<div id="footer">Footer</div>

</body>

1. Next, you’ll create the style sheet. Copy the code below into your text editor and save the file as layout.css.

**#header** {

background-color: #369;

color: #fff;

}

**#sidebar** {

background-color: #ff6;

}

**#nav** {

background-color: #ddd;

}

**#footer** {

border-top: 1px solid #369;}

1. Save both files and load the page in your browser. The five divisions appear in order, from top to bottom.

Imagine your design department has specified that the navigation must be on the left and the sidebar on the right, with the main content column in the middle. The header and footer should extend across the whole page width and we don’t know which of the three columns in between will be the longest. The source order is mandated by your accessibility and usability experts and isn’t negotiable. How can you combine all those requirements into a working layout?

You are going to have to add an extra element into the markup for this to work. You need an element that wraps around the three “columns”.

1. Update your HTML document:

<div id="header">Header</div>

<div id="wrapper">

<div id="main">Main content</div>

<div id="sidebar">Sidebar</div>

<div id="nav">Navigation</div>

</div><div id="footer">Footer</div>

The designers have stipulated that the navigation needs to be 12em wide while the sidebar should be 14em. The main content column should have a fluid width, so that the layout adapts to different window sizes, since fixed-width layouts aren’t very user friendly. To prevent lines of text from being too long, impeding readability, you need to constrain the layout to a maximum width. In order to prevent overlap in extremely narrow windows you also need to constrain the layout to a minimum width. Within those constraints, the layout should be centered horizontally within the browser window.

1. Next, update the CSS file to assign the widths to the navigation and the sidebar and set the width constraints and general centering.

body **{**

**margin: 0 auto;**

**min-width: 40em;**

**max-width: 56em;**

**}**

**#sidebar** {

**width: 13em;**

**padding: 0 0.5em;**

background-color: #ff6;

}

**#nav** {

**width: 11em;**

**padding: 0 0.5em;**

background-color: #ddd;}

1. Save the files and reload—you should see that the yellow sidebar and the grey navigation elements have the widths you want. If your browser window is wide enough, you will also see that the whole page is constrained in width and is centered horizontally.
2. Try changing the window size and see how the layout adapts.

If you look closely at the code you’ll see that the widths were set to 13em and 11em instead of 14em and 12em. That’s because you need some horizontal padding; you don’t want the content of those columns to lie flush with the edges, because it doesn’t look very nice. Padding adds to the width, so 13em + 0.5em + 0.5em adds up to the 14em you want.

**Making columns**

Okay, you have your basic building blocks, but they just appear one after the other. You want three columns, so you need to start floating them.

1. Add the following rules to your CSS file:

**#main** {

**float: left;**

}

**#sidebar** {

**float: left;**

width: 13em;

padding: 0 0.5em;

background-color: #ff6;

}

**#nav** {

**float: left;**

width: 11em;

padding: 0 0.5em;

background-color: #ddd;}

That floats them, all right, but they’re in the wrong order. Also, the main content column is too narrow. And what happened to our footer?

1. Let’s deal with the footer first. The problem is that the three columns are floated, which takes them out of the document flow. The footer is pushed up against the header and the line box containing the text is shortened so that the word “Footer” appears to the right of the floats. You can remedy this by making sure the footer is cleared from all the floated columns. Add the following rule to the CSS file:

**#footer** {

**clear: left;**

border-top: 1px solid #369;}

1. Now for the three columns.

The key to this whole trick is the wrapper element. We will set a left and right margin on it that corresponds to the widths of your side columns (the navigation and the sidebar). The main content column will occupy the whole width of the wrapper, while the side columns will be shifted into the space vacated by the margins. First, set up the margins for the wrapper, by adding the following rule to the CSS file:

**#wrapper** {

margin: 0 14em 0 12em;

padding: 0 1em;}

1. The next step is to make the main content column take up the full width of its wrapper parent;

**#main** {

float: left;

**width: 100%;**

**background-color: lime;}**

1. Save and reload—you’ll see a bright lime green content column, with the sidebar and navigation below it. You’ll also notice that there is a lot of white space on both sides. The trick is to get our side columns to slip into that white space.

Next I’ll move you on to the sidebar—it’s floated and it has the right width, but since the #main column is 100% wide, it pushes the sidebar down. How do you get it to go up and stay next to #main, although #main occupies the whole width? Let’s do it in two small steps: first, you’ll move it up; then you’ll shift it out into the margin.

1. Here you’ll use a nifty trick to get the floated sidebar, which has been pushed down, to move back up again—make the following addition to the #sidebar rule:

**#sidebar** {

float: left;

width: 13em;

padding: 0 0.5em;

background-color: #ff6;

**margin-left: -14em;}**

1. Save and reload, and you’ll see that the sidebar is now on the same vertical level as the content column. By setting a negative left margin equal to the width of the sidebar, we move the element back into the wrapper and it isn’t pushed down. The problem is that it overlaps the content.
2. You need to shift it out into the margin without making it drop down again, and this is where relative positioning—finally—comes in. It does precisely what we want: it shifts the generated box without moving the element itself. Add the highlighted properties below into the rule for #sidebar:

**#sidebar** {

float: left;

width: 13em;

padding: 0 0.5em;

background-color: #ff6;

margin-left: -14em;

**position: relative;**

**left: 15em;}**

Note that you had to shift it 15em, not 14em—that’s because there’s 1em of right padding on the wrapper that you need to get past. The sidebar is now where it belongs: out in the margin, next to the content column, lining up nicely with the right-hand edges of the header and the footer.

1. Now you need to do the same with the navigation this is done in a similar way, but it has a twist of its own. Moving and shifting the sidebar was easy, because the movements were essentially the same as the column’s width: 14em negative margin and a 14em+1em shift to the right. But the navigation column needs to be moved all the way across the content column and then be shifted even further out into the margin.

Our friend here is percentages. A percentage value on the margins of the navigation column will be relative to the width of its parent, the wrapper. You want to move the column all the way across the wrapper—add the property highlighted below to the rule for #nav:

**#nav** {

float: left;

width: 11em;

padding: 0 0.5em;

background-color: #ddd;

**margin-left: -100%;}**

1. Hey presto! Save and reload again, and you should see the navigation overlapping the left-hand side of the content column. All you need to do now is to shift it out into the margin. Add the following highlighted properties to the rule for #nav:

**#nav** {

float: left;

width: 11em;

padding: 0 0.5em;

background-color: #ddd;

margin-left: -100%;

**position: relative;**

**right: 13em;}**

1. Again, the width of the navigation is 12em, but you have 1em of wrapper padding to get past so you need to shift the box 13em. You’re shifting it to the left, in other words from the right edge, which is why the right property is being used.