## Floats

Floats have commonly been used to create entire web site layouts featuring multiple columns of information floated so they sit alongside one another (the default behavior would be for the columns to sit below one another, in the same order as they appear in the source).

simple float example

First, we'll start off with some simple HTML — add the following to your HTML body, removing anything that was inside there before:

<h1>Simple float example</h1>

<div class="box">Float</div>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. </p>

<p>Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.</p>

<p>Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est</p>

Now apply the following CSS to your HTML and check the output

body {

width: 90%;

max-width: 900px;

margin: 0 auto;

font: .9em/1.2 Arial, Helvetica, sans-serif

}

.box {

width: 150px;

height: 100px;

border-radius: 5px;

background-color: rgb(207,232,220);

padding: 1em;

}

Now add float rule

To float the text around it add the [float](https://developer.mozilla.org/en-US/docs/Web/CSS/float) and [margin-right](https://developer.mozilla.org/en-US/docs/Web/CSS/margin-right) properties to the .box

.box {

float: left;

margin-right: 15px;

width: 150px;

height: 100px;

border-radius: 5px;

background-color: rgb(207,232,220);

padding: 1em;

}

Floating the content to the right has exactly the same effect, but in reverse — the floated element will stick to the right, and the content will wrap around it to the left. Try changing the float value to right and replace [margin-right](https://developer.mozilla.org/en-US/docs/Web/CSS/margin-right) with [margin-left](https://developer.mozilla.org/en-US/docs/Web/CSS/margin-left) in the last ruleset to see what the result is

Floating the content to the right has exactly the same effect, but in reverse — the floated element will stick to the right, and the content will wrap around it to the left. Try changing the float value to right and replace [margin-right](https://developer.mozilla.org/en-US/docs/Web/CSS/margin-right) with [margin-left](https://developer.mozilla.org/en-US/docs/Web/CSS/margin-left) in the last ruleset to see what the result is

.special {

background-color: rgb(79,185,227);

padding: 10px;

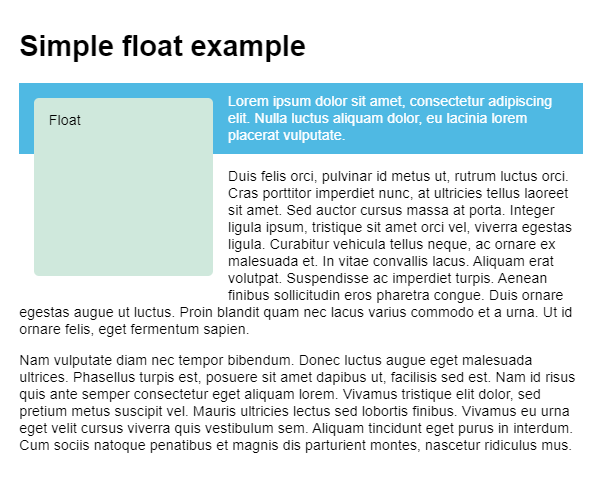
color: #fff;

}

To make the effect easier to see, change the margin-right on your float to margin for .box, so you get space all around the float.

You will be able to see the background on the paragraph running right underneath the floated box

The [line boxes](https://developer.mozilla.org/en-US/docs/Web/CSS/Visual_formatting_model#Line_boxes) of our following element have been shortened so the text runs around the float, but due to the float being removed from normal flow the box around the paragraph still remains full width.



## Clearing floats

We have seen that the float is removed from normal flow and that other elements will display beside it, therefore if we want to stop the following element from moving up we need to clear it; this is achieved with the [clear](https://developer.mozilla.org/en-US/docs/Web/CSS/clear) property.

In your HTML from the previous example, add a class of cleared to the second paragraph below the floated item. Then add the following to your CSS:

.cleared {

clear: left;

}

You should see that the following paragraph clears the floated element and no longer comes up alongside it. The clear property accepts the following values:

* left: Clear items floated to the left.
* right: Clear items floated to the right.
* both: Clear any floated items, left or right.

## Clearing boxes wrapped around a float

You now know how to clear something following a floated element, but let's see what happens if you have a tall float and a short paragraph, with a box wrapped around both elements. Change your document so that the first paragraph and our floated box are wrapped with a [<div>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div) with a class of wrapper.

The clearfix hack

The way that this situation has traditionally been dealt with is to use something known as a "clearfix hack". This involves inserting some generated content after the box which contains the float and wrapping content, and setting that to clear both.

Add the following CSS to our example:

.wrapper::after {

content: "";

clear: both;

display: block;

}

## positioning

Positioning allows you to take elements out of the normal document layout 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.

Static positioning

Static positioning is the default that every element gets — it just means "put the element into its normal position in the document layout flow — nothing special to see here

Relative positioning

This is very similar to static positioning, except that once the positioned element has taken its place in the normal layout flow, you can then modify its final position, including making it overlap other elements on the page. Go ahead and update the position declaration in your code.

position: relative;

If you save and refresh at this stage, you won't see a change in the result at all.  So how do you modify the element's position? You need to use the [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top), [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom), [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left), and [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right) properties

Absolute positioning

Absolute positioning brings very different results. Let's try changing the position declaration in your code as follows:

 position: absolute;

  top:30px;

  left:30px;

First of all, note that the gap where the positioned element should be in the document flow is no longer there — the first and third elements have closed together like it no longer exists! Well, in a way, this is true. An absolutely positioned element no longer exists in the normal document layout flow. Instead, it sits on its own layer separate from everything else. This is very useful: it means that we can create isolated UI features that don't interfere with the position of other elements on the page.  For example, popup information boxes and control menus; rollover panels; UI features that can be dragged and dropped anywhere on the page; and so on...

Second, notice that the position of the element has changed — this is because [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top), [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom), [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left), and [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right) behave in a different way with absolute positioning.

Instead of specifying the direction the element should move in, they specify the distance the element should be from each containing element's sides. So in this case, we are saying that the absolutely positioned element should sit 30px from the top of the "containing element", and 30px from the left.

Positioning contexts

Which element is the "containing element" of an absolutely positioned element? This is very much dependent on the position property of the ancestors of the positioned element.

If no ancestor elements have their position property explicitly defined, then by default all ancestor elements will have a static position. The result of this is, the absolutely positioned element will be contained in the **initial containing block**. The initial containing block has the dimensions of the viewport, and is also the block that contains the [<html>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/html) element. Simply put, the absolutely positioned element will be contained outside of the [<html>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/html) element, and be positioned relative to the initial viewport.

We can change the **positioning context** — which element the absolutely positioned element is positioned relative to. This is done by setting positioning on one of the element's ancestors — to one of the elements it is nested inside (you can't position it relative to an element it is not nested inside). To demonstrate this, add the following declaration to your body rule:

position: relative;

Fixed positioning

Let's now look at fixed positioning. This works in exactly the same way as absolute positioning, with one key difference: whereas absolute positioning fixes an element in place relative to its nearest positioned ancestor (or the document's root element if there isn't one, i.e. the <html> element),**fixed positioning** fixes an element in place relative to the browser viewport itself. This means that you can create useful UI items that are fixed in place, like persisting navigation menus.

update the body rule to remove the position: relative; declaration and add a fixed height, like so

body {

width: 500px;

height: 1400px;

margin: 0 auto;

}

Now we're going to give the [<h1>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/h1) element position: fixed;, and get it to sit at the top of the viewport. Add the following rule to your CSS:

h1 {

position: fixed;

top: 0;

width: 500px;

margin-top: 0;

background: white;

padding: 10px;

}

Now give little margin to p element as below

p:nth-of-type(1) {

margin-top: 60px;

}

position: sticky

There is another position value available called position: sticky, which is somewhat newer than the others. This is basically a hybrid between relative and fixed position, which allows a positioned element to act like it is relatively positioned until it is scrolled to a certain threshold point (e.g. 10px from the top of the viewport), after which it becomes fixed. This can be used to for example cause a navigation bar to scroll with the page until a certain point, and then stick to the top of the page.

.positioned {

position: sticky;

top: 30px;

left: 30px;

}

## Overlapping Elements

When elements are positioned, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a positive or negative stack order:

img {  
  position: absolute;  
  left: 0px;  
  top: 0px;  
  z-index: -1;  
}

## Getting Started with Bootstrap

Bootstrap is the world’s most popular framework for building responsive, mobile-first sites and applications. Inside you’ll find high quality HTML, CSS, and JavaScript to make starting any project easier than ever.

Starter Template – <https://v4-alpha.getbootstrap.com/getting-started/introduction/>

<!DOCTYPE html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/bootstrap.min.css" integrity="sha384-rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ" crossorigin="anonymous">

</head>

<body>

<h1>Hello, world!</h1>

<!-- jQuery first, then Tether, then Bootstrap JS. -->

<script src="https://code.jquery.com/jquery-3.1.1.slim.min.js" integrity="sha384-A7FZj7v+d/sdmMqp/nOQwliLvUsJfDHW+k9Omg/a/EheAdgtzNs3hpfag6Ed950n" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.4.0/js/tether.min.js" integrity="sha384-DztdAPBWPRXSA/3eYEEUWrWCy7G5KFbe8fFjk5JAIxUYHKkDx6Qin1DkWx51bBrb" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/js/bootstrap.min.js" integrity="sha384-vBWWzlZJ8ea9aCX4pEW3rVHjgjt7zpkNpZk+02D9phzyeVkE+jo0ieGizqPLForn" crossorigin="anonymous"></script>

</body>

</html>

Grid System – <https://v4-alpha.getbootstrap.com/layout/grid/>

Bootstrap includes a powerful mobile-first flexbox grid system for building layouts of all shapes and sizes. It’s based on a 12 column layout and has multiple tiers, one for each [media query range](https://v4-alpha.getbootstrap.com/layout/overview/#responsive-breakpoints). You can use it with Sass mixins or our predefined classes.

container and container-fluid

## Working of Grid System

* Rows must be placed within a *.container* class for proper alignment and padding.
* For responsive width use *.container* class and for fixed width across all viewport, use the *.container-fluid* class.
* Use rows to create horizontal groups of columns.
* Content should be placed within the columns, and only columns may be the immediate children of rows.
* Columns contain padding for controlling the space between them.
* If you place more than 12 columns in a row, then the columns will be placed in a new line.
* Columns create gaps between column content via padding. Therefore, you can remove the margin from rows and padding from columns with *.no-gutters* class on the row.
* You can make grid system responsive by using five grid breakpoints such as extra small, small, medium, large, and extra large.
* Predefined grid classes like .col-4 are available for quickly making grid layouts. LESS mixins can also be used for more semantic layouts

Media Object

The [media object](http://www.stubbornella.org/content/2010/06/25/the-media-object-saves-hundreds-of-lines-of-code/) helps build complex and repetitive components where some media is positioned alongside content that doesn’t wrap around said media. Plus, it does this with only two required classes thanks to flexbox.

<https://v4-alpha.getbootstrap.com/layout/media-object/>

Handfull classes for styling

bg-primary

Components

<https://v4-alpha.getbootstrap.com/components/alerts/>

Nav and Navbar

Card

Utility

<https://v4-alpha.getbootstrap.com/utilities/colors/>