**Positioning and Layout in CSS**

Up to this point, you have learned how the content of each element is represented in CSS using a box, and you’ ve seen many of the properties you can use to affect the appearance of the box and its content. Now it’ s time to look at how to control where the boxes should be positioned within a page.

In CSS, there are three *positioning schemes* that allow you to control layout of a page: *normal,* *float,* and *absolute* positioning. In the following sections, you’ ll be seeing how you can use each of these to indicate where the content of an element should appear on the page.

While the CSS positioning schemes were not really intended to be a mechanism for controlling the layout of pages, they have become the standard way to lay out pages on the Web. For the rest of the chapter, we will be looking at *how* you can control where boxes appear on the page using CSS; then in the next chapter we will look at how to apply this knowledge to create attractive layouts.

*Before CSS, web designers commonly used tables to control the layout of web pages. While you will still occasionally see tables used for this purpose, they were designed to contain tabular data, and you should aim to control layout of new pages using CSS instead. If you use CSS to control layout rather than tables, your pages will be smaller (in terms of lines of code), easier to adapt to different devices, easier to redesign, faster to load, and more visible to search engines.*

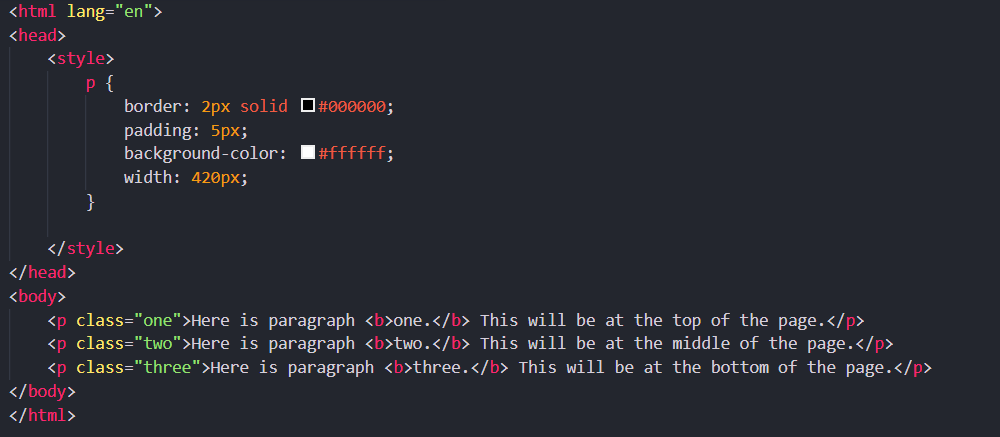
**Normal Flow:**

By default, elements are laid out on the page using what is known as *normal flow*. In normal flow, the block - level elements within a page will flow from top to bottom (remember that each block - level element will appear as if it is on a new line), and inline elements will flow from left to right (because they do not start on a new line).

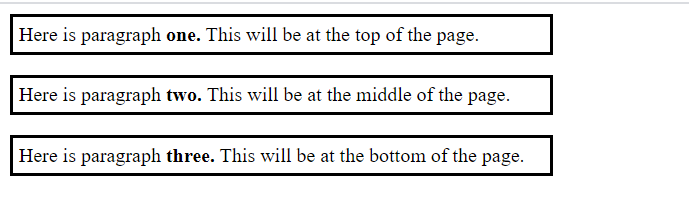
For example, each heading and paragraph should appear on a different line, whereas the contents of elements such as <b> , <em> , and <span> sit within a paragraph or other block-level element; they do not start on new lines.

Figure below, illustrates this with three paragraphs, each of which is a block - level element sitting on top of the other. Inside each paragraph is an example of an inline element, in this case the <b> element.

Example:



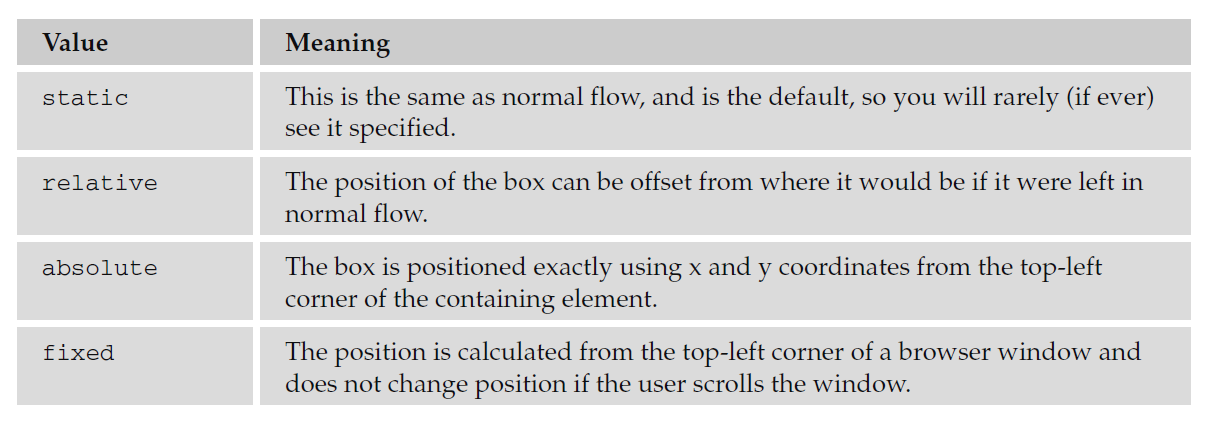
Output:



If you want the content of elements to appear in other places than where they would in normal flow, you have two properties to help you: position and float.

***The position Property:***

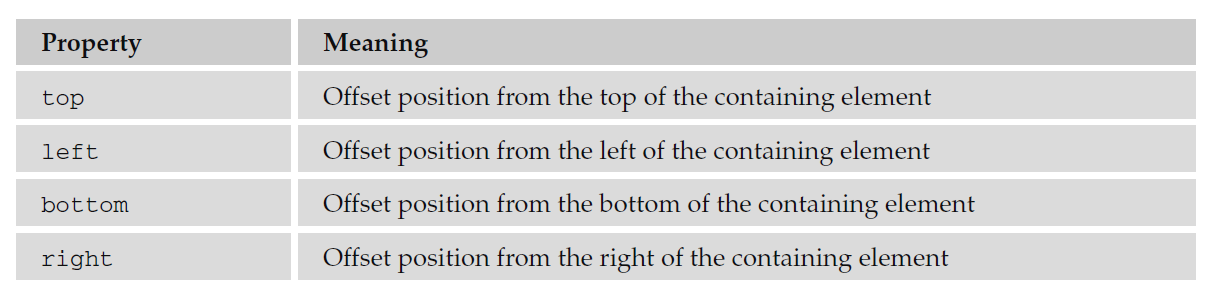
The position property allows you to specify how you want to control the position for a box (and is generally used to take items out of normal flow). It can take the four values listed in the table that follows:



You will see how these are used in the coming sections.

**Box Offset Properties:**

As you’ ll see in the coming sections, when boxes have a position property whose value is relative, absolute, or fixed, they will also use *box offset* properties to indicate where these boxes should be positioned. The table that follows lists the box offset properties.

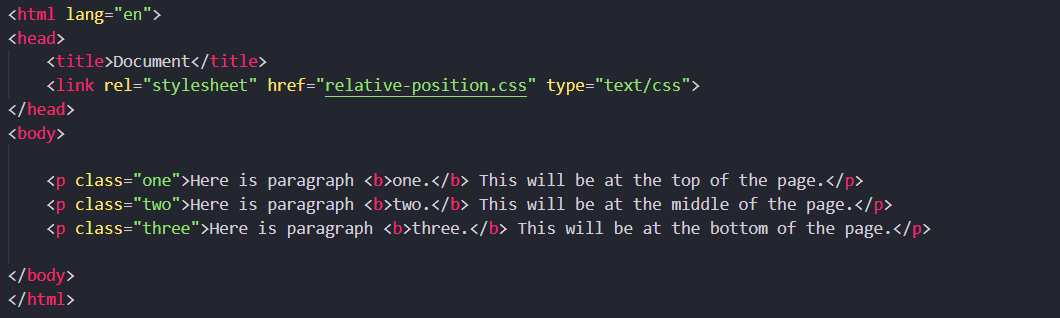


Each can take a value of a length, a percentage, or auto. Relative units, including percentages, are calculated with respect to the containing boxes’ dimensions or properties.

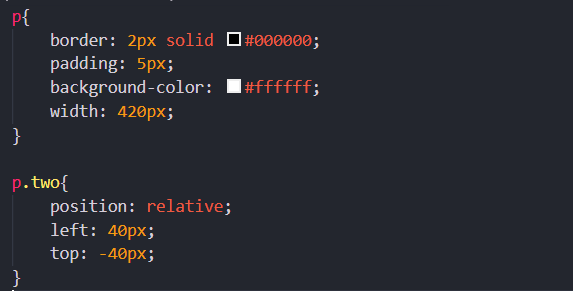
**Relative Positioning:**

Relative positioning allows you to move a box in relation to where it would appear in normal flow. For example, you might move a box 30 pixels down from where it would appear in normal flow, or 100 pixels to the right. It is displaced from where it would be in normal flow using the box offset properties.

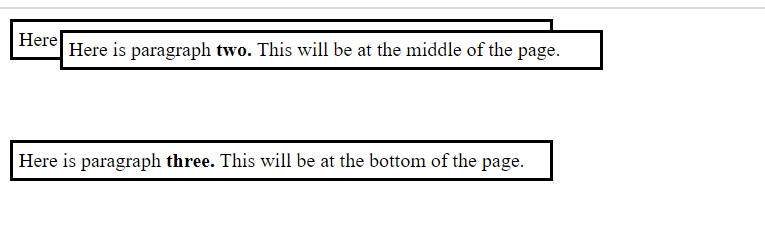
Example (html file):



(CSS file):



Output:



The value of the box offsets is most commonly given in pixels or a percentage.

*You should specify only a left or right offset and a top or bottom offset. If you specify both left and right or both top and bottom, the right or bottom offset will be ignored.*

When you are using relative positioning, you can end up with some boxes overlapping others, as in the previous example. Because you are offsetting a box relative to normal flow, if the offset is large enough, one box will end up on top of another. This may create an effect you are looking for; however, there are a couple of pitfalls you should be aware of:

* Unless you set a background for a box (either a background color or image) the box will be transparent by default, making any overlapping text an unreadable mess. In the preceding example, I used the background - color property to make the background of the paragraphs white and thereby prevent this from happening.
* The CSS specification does not say which element should appear on top when relatively positioned elements overlap each other, so there can be differences between browsers (although you can control this using the z - index property, which you will meet shortly).

**Absolute Positioning:**

Absolute positioning takes an element out of normal flow, allowing you to fix its position. You can specify that an element’s content should be absolutely positioned by giving it the position property with a value of absolute; then you use the box offset properties to position it where you want.

The box offsets fix the position of a box relative to the *containing block*—which is slightly different from a containing element because it is a containing element whose position property is set to relative or fixed.

**Note: Refer to the code folder for example**

**Fixed Positioning:**

The final value you need to be aware of for the position property is the value fixed. This value specifies that the content of the element should not only be completely removed from normal flow, but also that the box should not move when users scroll up or down a page.

While Firefox and Safari have offered support for fixed positioning for a while, IE7 was the first version of Internet Explorer to support it.

We’ ll use the following sample of XHTML to demonstrate fixed positioning. This example continues with several more paragraphs so that you can see the page scrolling while the content of the <div> element remains fixed at the top of the page:

**Note: Refer to the code folder for example**

***The z-index Property:***

Elements positioned using absolute and relative positioning often overlap other elements. When this happens the default, behavior is to have the first elements underneath later ones. This is known as *stacking context.* You can specify which of the boxes appears on top using the z- index property. If you are familiar with graphic design packages, the stacking context is similar to using the “bring to top” and “send to back” features.

The value of the z-index property is a number, and the higher the number the nearer the top that element should be displayed (for example, an item with a z-index of 10 will appear on top of an item with a z-index of 5).

**Note: Refer to the code folder for example**

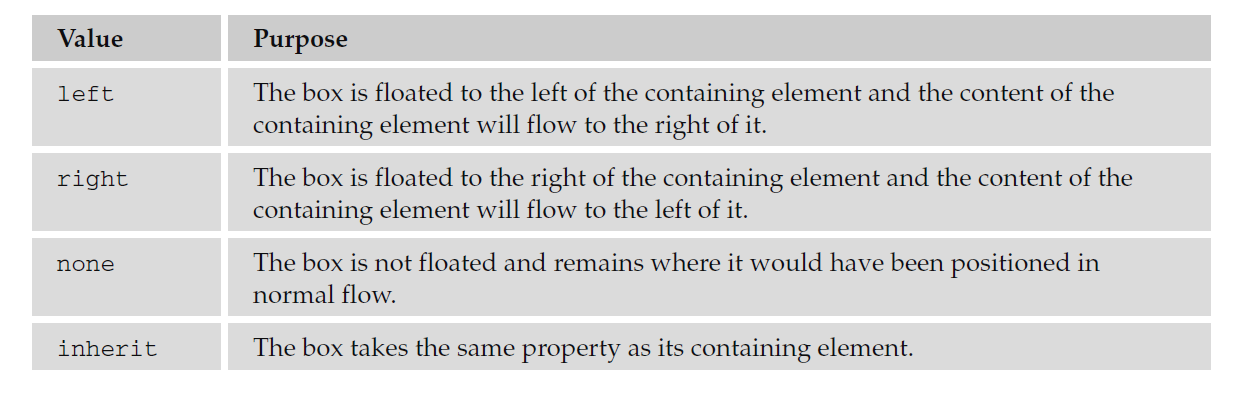
**Floating Using the float Property:**

The float property allows you to take an element out of normal flow and place it as far to the left or right of a containing box as possible.

Anything else that lives in the containing element will flow around the element that is associated with the float property (just like text and other elements can flow around an image).

Whenever you specify a float property on an element, you must also set a width property indicating the width that the box should take up; otherwise, it will automatically take up 100 percent of the width of the containing box, leaving no space for things to flow around it and therefore making it appear just like a plain block-level element.

To indicate that you want a box floated either to the left or the right of the containing box, you set the float property, which can take one of the values listed in the table that follows.

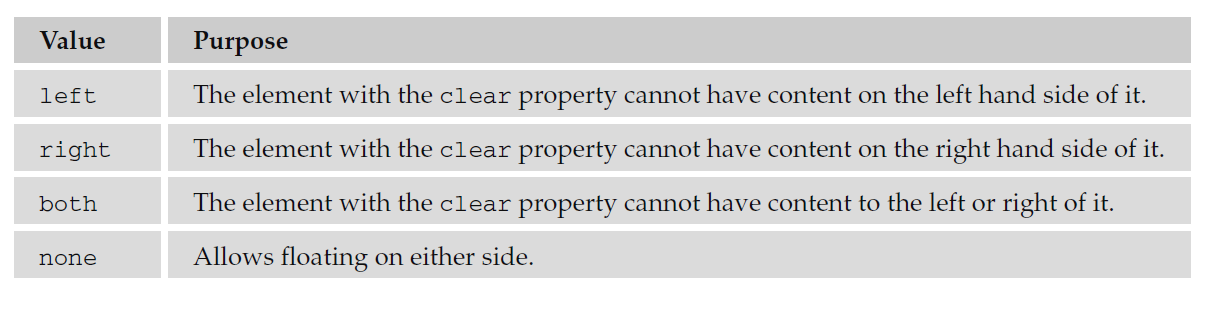


When a box uses the float property, vertical margins will not be collapsed above or below it like block boxes in normal flow can be (because it has been taken out of normal flow). The floated box will be aligned with the top of the containing box.

**Note: Refer to the code folder for example**

**The clear Property:**

The clear property is especially helpful when working with boxes that are floated. As you just saw in example just saw in class, content can flow around a floated element; however, you might not want this to happen—you might prefer that nothing sit next to the floated element, and that surrounding content be pushed underneath the floated element. This is what the clear property is for, and the following table shows you the values that this property can take.



**Note: Refer to the code folder for example**