fit. Objects in the normal flow influence the position of the surrounding content (sibling elements).

In CSS positioning, blocks are defined as being either in the normal flow or removed from the normal flow. Floating and positioning elements changes their relationship to the normal flow, as discussed in the following sections.

Floating

If you've ever aligned an image to the right or left margin and allowed text to wrap around it, then you understand the concept behind floats in CSS. In fact, that is precisely the functionality that the `float` property was created to provide. The primary difference is that you can float any element with CSS (paragraphs, lists, `div`s, and so on), not just images.* It is important to note that floating is not a positioning scheme; it is a unique feature with some interesting behaviors to be aware of, as discussed later in this section.

Floats are useful for far more than just occasionally pushing an image off to one side. In fact, they are one of the primary tools used in modern CSS-based web design. Floats are used to create multicolumn layouts, navigation toolbars from unordered lists, table-like alignment without tables, and more. See Chapter 24 for examples.

To make an element float to the left or right and allow the following text to wrap around it, apply the `float` property to the element.

float

Values:	`left | right | none | inherit`
Initial value:	`none`
Applies to:	All elements
Inherited:	No

In this simple example, the `float` property is used to float an image to the right (Figure 21-1).

```
img {float: right; margin: 20px;}

<p><img src="img/placeholder.gif">Aliquam pulvinar volutpat...</p>
```

As you can see in Figure 21-1, the `float` property applied to the `img` element effectively replaces the deprecated `align` attribute. In this image example, the margin does the work of the deprecated `hspace` and `vspace` attributes. The advantage of `margin` is that you can apply different amounts of margin space on each side of the

* Some browsers allow table elements to be floated with the align attribute as well.

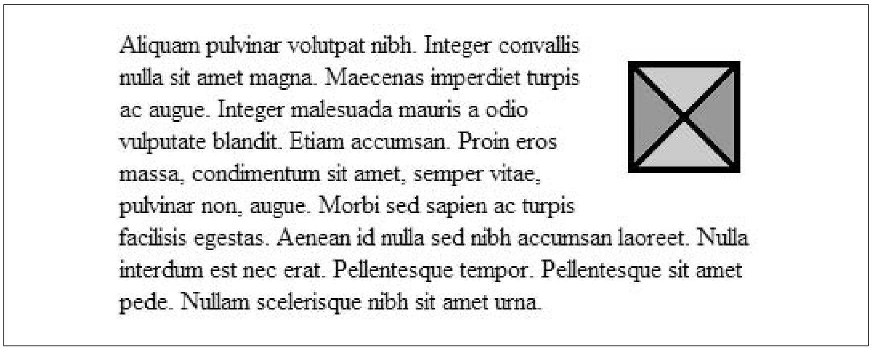

Aliquam pulvinar volutpat nibh. Integer convallis nulla sit amet magna. Maecenas imperdiet turpis ac augue. Integer malesuada mauris a odio vulputate blandit. Etiam accumsan. Proin eros massa, condimentum sit amet, semper vitae, pulvinar non, augue. Morbi sed sapien ac turpis facilisis egestas. Aenean id nulla sed nibh accumsan laoreet. Nulla interdum est nec erat. Pellentesque tempor. Pellentesque sit amet pede. Nullam scelerisque nibh sit amet urna.

Figure 21-1. Floating an image to the right

image (hspace and vspace apply the same amount of space on opposite sides). Padding may also be used to add space around the contents of a floated element.

Although the behavior in this example should be familiar to those who have worked with HTML, it is quite interesting when considered in terms of the CSS visual layout model. Floated elements are removed from the normal flow of the document, yet they still have an effect on other elements in the layout—surrounding content is reflowed to stay out of their way. To use one popular analogy, they are like islands in a stream—they are out of the normal flow, but the stream has to flow around them. Floated elements are unique in this regard, because elements removed from the flow normally cease to have influence on other elements (this will be discussed in the upcoming positioning sections).

Floating Basics

The float property is not limited to images; it can be applied to any element. In this slightly more ambitious example shown in Figure 21-2, the float property is applied to a selection of text (known in CSS as an “inline non-replaced element”). Note that the dotted lines are a device for pointing out the parts of the element boxes in this figure and would not actually appear in the browser.

```
span.note {
    float: right;
    width: 200px;
    margin: 20px;
    background-color: #999;
    font-weight: bold; }

p {border: solid 2px #666; padding: 30px;}

<p><span class="note">I'm going to go over here for a little while. Don't
mind me. </span> Lorem ipsum dolor sit amet, consectetuer  . . .
```

The results reveal some basic behaviors of element floating:

- All floated elements (even inline elements, as shown in the example) take on block behaviors. It is equivalent to setting display: block (although it is not necessary to do so).

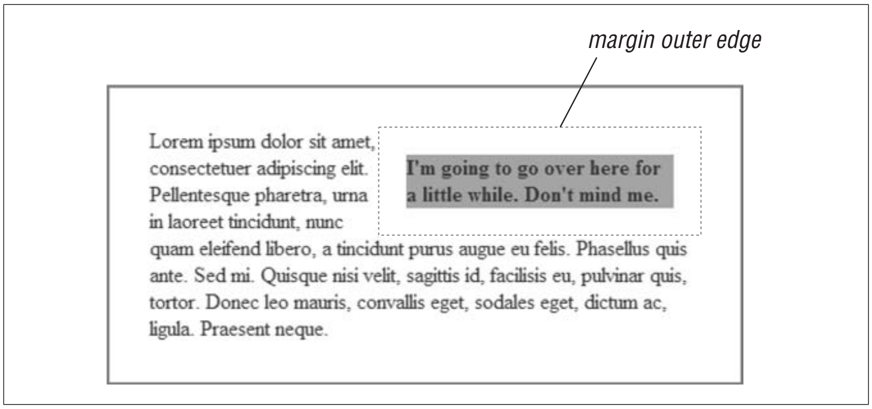

margin outer edge
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Pellentesque pharetra, urna in laoreet tincidunt, nunc quam eleifend libero, a tincidunt purus augue eu felis. Phasellus quis ante. Sed mi. Quisque nisi velit, sagittis id, facilisis eu, pulvinar quis, tortor. Donec leo mauris, convallis eget, sodales eget, dictum ac, ligula. Praesent neque.
I'm going to go over here for a little while. Don't mind me.

Figure 21-2. Floating an inline text element

- When floating a non-replaced (i.e., text) element, it is necessary to specify the `width` for the element. Not doing so can result in the content area box collapsing to its narrowest possible width.
- The floated element stays within the content area of its *containing block* (the nearest block-level ancestor element). It does not cross into the padding.
- Margins are maintained on all sides of the floated element. In other words, the entire element box (from outer edge to outer edge) is floated, and the surrounding content flows around it.
- Unlike normal elements, margins around floated elements never collapse (even vertically).

The elements following the floated element exhibit unusual behavior as well. In the following example and Figure 21-3, the floated graphic is taller than its parent paragraph element and hangs down over the following paragraph. The second paragraph (named "boxed") has been given a background and border to show the boundaries of its element box compared to its contents.

```
img { float: left; }
p.boxed { background-color: #999; border: solid 2px #333; }
```

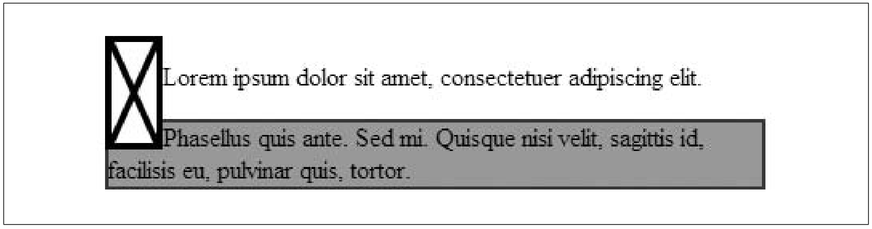

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Phasellus quis ante. Sed mi. Quisque nisi velit, sagittis id, facilisis eu, pulvinar quis, tortor.

Figure 21-3. Wrapped element behavior

The border and background position show that the position of the second paragraph's element box is unchanged by the presence of the floated image element. Only its content moves over to make way for the floated image. Notice also that the floated image overwrites (appears "in front of") the background and border for the following paragraph. This is the prescribed behavior for floated elements. Other overwriting behaviors are discussed in the "Negative Margins and Overlap" section ahead.

Floating Behavior

The CSS 2.1 specification provides eight precise rules restricting the positioning of floated objects, which are summarized here. If you need the details, go right to the source, at *www.w3.org/TR/CSS21/visuren.html#float-position*. Eric Meyer provides a useful translation and illustration of the rules in his book *Cascading Style Sheets: The Definitive Guide* (O'Reilly).

In addition to requiring that floated elements stay within the inner edge (or content area) of their containing blocks, there are a number of rules designed to prevent the overlapping of floated objects.

Browsers (even current standards-conformant browsers) may be inconsistent in the way they handle floated objects due to a certain amount of leeway in the specification and because they follow historical and expected practice. Be sure to test.

Floated elements in close proximity in the source document are not permitted to overwrite one another. Instead, the rules prescribe:

- If elements are floated in the same direction, each subsequent floated object should move in that direction until it reaches the inner edge of the containing block or until it bumps into another floated element. This rule results in multiple floated elements accumulating against the targeted edge.
- If there is not enough room for floated elements to appear side by side, then the second floated object should move down until there is enough room for it to display without overlapping the first object.

The effects of these rules are demonstrated in Figure 21-4.

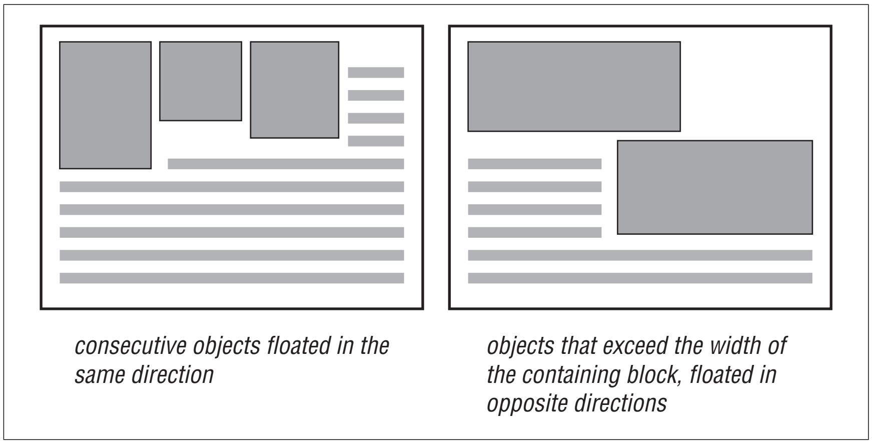

consecutive objects floated in the same direction
objects that exceed the width of the containing block, floated in opposite directions

Figure 21-4. Floated objects accumulate or bump down instead of overlapping

Other rules restrict how high the top edge of a floated element may be positioned.

- The top of a floated element must stay within the top inner edge of its parent element.
- The top of a floated element that is not contained in a block element may not be higher than a preceding block-level element. The float is essentially "blocked" from floating above it.
- The top of a floated element may not start higher than a floated element that precedes it in the document source.
- If a floated element starts in the middle of the text flow of an element, it does not float to the top of that element, but rather starts at the top of the line box for the surrounding text (Figure 21-5).

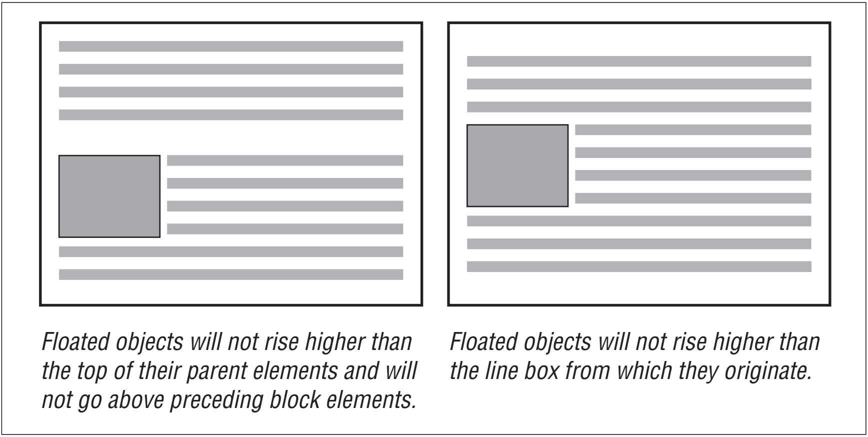

Floated objects will not rise higher than the top of their parent elements and will not go above preceding block elements.
Floated objects will not rise higher than the line box from which they originate.

Figure 21-5. Top edge restrictions on floated elements

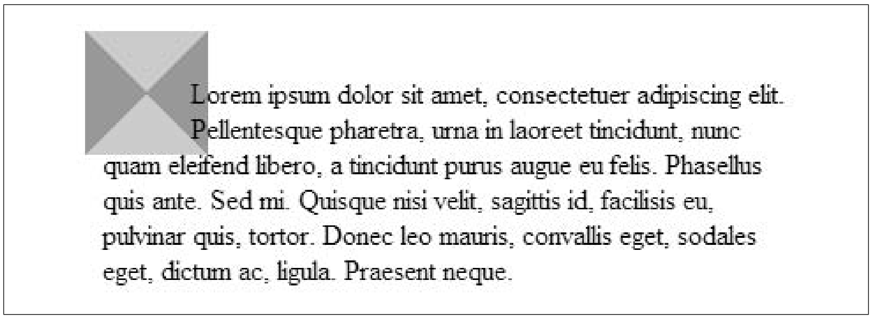
Floating elements are also not permitted to stick out of the edge of their containing elements, unless they are too wide to fit (like a wide image). This prevents sequential floated elements from accumulating against an edge and growing wider than the containing block. When the stack grows too wide, the element that doesn't fit gets bumped down so that it clears the floated elements above.

The final two rules state, given all of the established restrictions, floated objects should be put as far left or right (as specified) and as far upward as possible until they reach a defined constraint. A higher position is preferable to one that is farther left or right. I like to picture floated objects on a page jockeying for position, pushing upward and outward until they bump into the edge of the containing block, another floated element, or an imposed ceiling from a previous block element or the like.

Negative Margins and Overlap

The two big rules for the placement of floated objects are that they should never go beyond the content area of their containing block and they should not overlap other elements. These guidelines seemingly get tossed out the window when you apply negative margins to a floated element, as shown in this example and in Figure 21-6.

```
img { float: left; margin: -10px; }
```



Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Pellentesque pharetra, urna in laoreet tincidunt, nunc quam eleifend libero, a tincidunt purus augue eu felis. Phasellus quis ante. Sed mi. Quisque nisi velit, sagittis id, facilisis eu, pulvinar quis, tortor. Donec leo mauris, convallis eget, sodales eget, dictum ac, ligula. Praesent neque.

Figure 21-6. A floated element with negative margins

The negative margin setting pulls the content area of the floated element out of its positioned element box, allowing the content to fall outside the confines of the containing block. There are no rules preventing elements with negative top margins from overwriting preceding content that has already been displayed, so negative vertical margins are best avoided.

Negative margins may also cause the flowed content to overlap the floated object. In these instances, the CSS 2.1 specification prescribes different rules for inline boxes and block boxes.

- When an inline box overlaps with a float, the entire element box (including the content, background, and border) overwrites or appears "in front of" the floated element. Be prepared that if you have a floated element with negative margins and you apply backgrounds or borders to inline elements in the wrapped text, those inline boxes may obscure the floated element.
- When block boxes overlap a float, the content of that box appears "in front of" the floated element, but the background and border of the element are overwritten by (appear "behind") the float. This is consistent with the example in Figure 21-3, but allows the text to go in front of the float in the instance of negative margins.

Clearing

Wrapping can be a nice, space-saving layout effect, but it is not always appropriate. There are certainly cases in which you want the area on the side of the floated element to be held clear and the following element to start at its normal position in the containing block. For those instances, use the `clear` property to prevent an element from appearing next to a floated element.

clear

Values: `left | right | both | none | inherit`