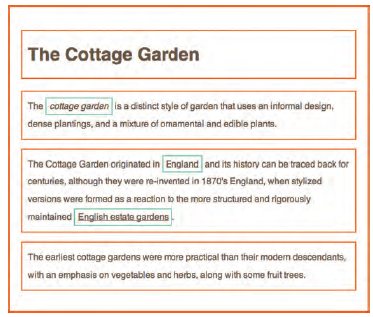
The key to understanding how CSS works is to imagine that there is an invisible box around every HTML element.

CSS allows you to create rules that control the way that each individual box (and the contents of that box) is presented.

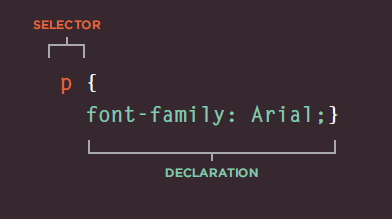
I have added outlines to each of the elements so that you can see how CSS will treat each elements if it lives inside its own box. In this example, block level elements are shown with red borders, and inline elements have green borders.



CSS works by associating rules with HTML elements. These rules govern

How the content of specified elements should be displayed. A CSS rule

Contains two parts: a selector and a declaration.



CSS declarations sit inside curly brackets and each is made up of two

Parts: a property and a value, separated by a colon. You can specify

Several properties in one declaration, each separated by a semi-colon.





If there are two or more rules

If there are two or more rules that apply to the same element, it is important to understand which will take precedence. LAST RU Le

if the two selectors are identical, the latter of the two will take precedence. Here you can seethe second I selector takes precedence over the first.

SPECIFICITY

If one selector is more specific than the others, the more specific rule will take precedence over more general ones. In this example:

h1 is more specific than \*

P b is more specific than P

p#intro is more specific than p

IMPORTANT

You can add !important after any property value to indicate that it should be considered more important than other rules that apply to the same element.

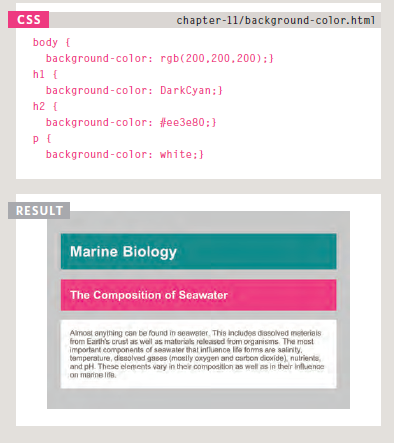
If you specify the **font-family** or **color** properties on the**<body>** element, they will apply to most child elements. This is because the value of the **font-family** property is **inherited** by child elements. It saves you from having to apply these properties to as many elements (and results in simpler style sheets).

You can compare this with the **background-color** or **border** properties; they are **not inherited** by child elements. If these were inherited by all child elements then the page could look quite messy.

You can force a lot of properties to inherit values from their parent elements by using **inherit** for the value of the properties. In this example, the**<div>** element with a **class** called **page** inherits the padding size from the CSS rule that applies to the **<body>** element



CSS treats each HTML elements if it appears in a box, and the **background-color** property sets the color of the background for that box.If you do not specify background color, then the background is transparent



**@font-face** allows you to use font, even if it is not installed on the computer of the person browsing, by allowing you to specify a path to a copy of the font, which will be downloaded if it is not on the user's machine.

The **font-weight** property allows you to create bold text. There are two values that this property commonly takes:

**Normal** This causes text to appear at abnormal weight.

**Bold** This causes text to appear bold.

If you want to create italic text,you can use the **font-style** property. There are three values this property can take:

**Normal** This causes text to appear in abnormal style (as opposed to italic or oblique).

**Italic** This causes text to appear italic.

**Oblique** This causes text to appear oblique.

The **text-transform** property is used to change the case of text giving it one of the followingvalues:

**Uppercase** This causes the text to appearuppercase.

**Lowercase** This causes the text to appearlowercase.

**Capitalize** This causes the first letter of each word to appear capitalized.

The **text-decoration** propertyallows you to specify the following values:

**None** This removes any decoration already applied to the text.

**Underline** This adds a line underneath the text.

**Overline** This adds a line over the top of the text.

**line-through** This adds a line through words.

**Blink** This animates the text to make it flash on and off (however this is generally frowned upon, as it is considered rather annoying).

Increasing the **line-height** makes the vertical gap between lines of text larger. A good starter setting is around 1.4 to1.5em. Because users can adjust the default size of text in their browser, the value of the **lineheight**property is best given items, not pixels, so that the gap between lines is relative to the size of text the user has selected.

Alignment

The **text-align** property allows

you to control the alignment of text. The property can take one of four values:

**left** This indicates that the text should be left-aligned.

**Right** This indicates that the text should be right-aligned.

**Center** This allows you to center text.

**Justify** This indicates that every line in paragraph, except the last line, should be set to take up the full width of the containing box.

Styling Links

In CSS, there are two **pseudoclasses**that allow you to set different styles for links that have and have not yet been visited

.**: link** This allows you to set styles for links that have not yet been visited

.**: visited** This allows you to set styles for links that have been clicked on. They are commonly used to control colors of the links and also whether they are to appear underlined or not.

Responding to Users

There are three pseudo-classesthat allow you to change the appearance of elements when abuser is interacting with them.

**: hover** This is applied when a user hovers over an element with appointing device such as a mouse. This has commonly been used to change the appearance of links and buttons when a user places their cursor over them. Itis worth noting that such events do not work on devices that use touch screens (such as the iPad) because the screen is not able to tell when someone is hovering their finger over an element.

**: active** This is applied when an element is being activated by a user; for example, when a button is being pressed or a link being clicked. Sometimes this is used to make button or link feel more like itis being pressed by changing the style or position of the element slightly.

**: focus** This is applied when an element has focus. Any element that you can interact with, such as link you can click on or any form control can have focus.

****

Overflowing Content

The **overflow** property tells the browser what to do if the content contained within a box is larger than the box itself. It can have one of two values:

**Hidden** This property simply hides any extra content that does not fit in the box.

**Scroll** This property adds a scrollbar tithe box so that users can scroll to see the missing content.

Border, Margin & Padding

Every box has a border (even if it is not visible or is specified Tobe 0 pixels wide). The border separates the edge of one box from another.

Margins sit outside the edge of the border. You can set the width of a margin to create agape between the borders of two adjacent boxes.

Padding is the space between the border of a box and any content contained within it.Adding padding can increase the readability of its contents



You can control the style of border using the **border-style** property. This property can take the following values:

**solid** a single solid line

**dotted** a series of square dots(if your border is 2px wide, then the dots are 2px squared with a2px gap between each dot)

**dashed** a series of short lines

**double** two solid lines (the value of the **border-width** property creates the sum of the two lines)

**groove** appears to be carved into the page

**ridge** appears to stick out from the page

**inset** appears embedded into the page

**outset** looks like it is coming out of the screen

**hidden** / **none** no border isshownYou can individually change the styles of different borders using:

**border-top-style**

**border-left-style**

**border-right-style**

**border-bottom-styleResu**

The **padding** property allows you to specify how much space should appear between the content of an element and itsborder.The value of this property is most often specified in pixels(although it is also possible tousle percentages or ems). If percentage is used, the padding is a percentage of the browser window (or of the containing box if it is inside another box).Please note: If a width is specified for a box, padding is added onto the width of the box

You can specify different values for each side of a box using:

**padding-top**

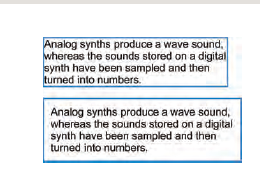
**padding-right**

**padding-bottom**

**padding-left**

Or you can use a shorthand (where the values are in clockwise order: top, right, bottom, left):

**padding: 10px 5px 3px 1px;**



The **margin** property controls

the gap between boxes. Its value is commonly given in pixels, although you may also use percentages or emissive one box sits on top of another, margins are collapsed , which means the larger of the two margins will be used and the smaller will be disregarded. Please note: If the width of a boxes specified then the margin is added to the width of the box. You can specify values for each side of a box using:

**margin-top**

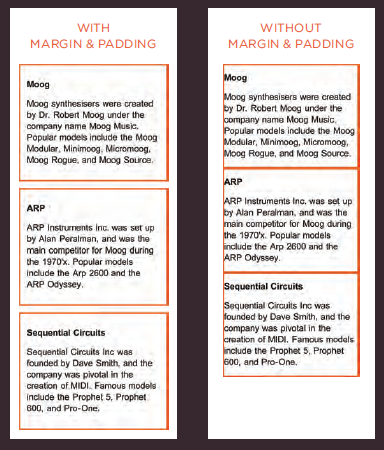
**margin-right**

**margin-bottom**

**margin-left**

You can also use the shorthand (where the values are in clockwise order: top, right, bottom, left):

**margin: 1px 2px 3px 4px;**



Border width/style/color

The **border** property allows you to specify the width, style and color of a border in one property (and the values should be coded in that specific order.

**Padding**

The **padding** property allows you to specify how much space should appear between the content of an element and itsborder.The value of this property is most often specified in pixels (although it is also possible to use percentages or ems). If percentage is used, the padding is a percentage of the browser window (or of the containing box if it is inside another box).Please note: If a width is specified for a box, padding is added onto the width of the box. As you can see, the second paragraph here is much easier to read because there is a space between the text and the borderof the box. The box is also wider because it has padding. You can specify different values for each side of a box

using:

**padding-toppadding-rightpadding-bottompadding-left**

Or you can use a shorthand (where the values are in clockwise order: top, right, bottom, left)

:**padding: 10px 5px 3px 1px;**

The value of the **padding** property is not inherited by child elements in the same way that the **color** value of the **font-family** property is; so you need to specify the **padding** for every element that needs to use it.

**Margin**

The **margin** property controls the gap between boxes. Its value is commonly given in pixels, although you may also use percentages or emissive one box sits on top of another, margins are collapsed, which means the larger of the two margins will be used and the smaller will be disregarded. Please note: If the width of a boxes specified then the margin is added to the width of the box. You can specify values for each side of a box using:**margin-topmargin-rightmargin-bottommargin-left**You can also use the shorthand(where the values are in clockwise order: top, right, bottom, left):**margin: 1px 2px 3px 4px;**Sometimes you might see the following, which means that the left and right margins should be10 pixels and the top and bottom margins should be 20 pixels: **margin: 10px 20px;**(This same shorthand shown above can also be applied top adding.)

**Center a Box**

If you want to center a box onthe page (or center it inside the element that it sits in), youkan set the **left-margin** and **right-margin** to **auto**. In order to center a box on the page, you need to set a **width** for the box (otherwise it will take up the full width of the page).Once you have specified the width of the box, setting the left and right margins to **auto** will make the browser put an equal gap on each side of the box. This centers the box on the page (or within the element that the boxsets inside).

The **text-align** property is inherited by child elements. You therefore also need to specify the **text-align** property on the centered box if you do not want the text inside it to be centered.

**Display**

The **display** property allows you to turn an inline element into a block-level element or vice versa, and can also be used to hide an element from the page. The values this property can take are:**inline**This causes a block-level element to act like an inlineelement.**block**This causes an inline element toast like a block-level element.**inline-block**This causes a block-level element to flow like an inline element, while retaining other features of a block-level element.**none**This hides an element from the page. In this case, the element acts as though it is not on the page at all (although a user could still see the content of the box if they used the ***view source*** optioning their browser).If you use this property, it is important to note that inline boxes are **not** supposed to create block-level elements.

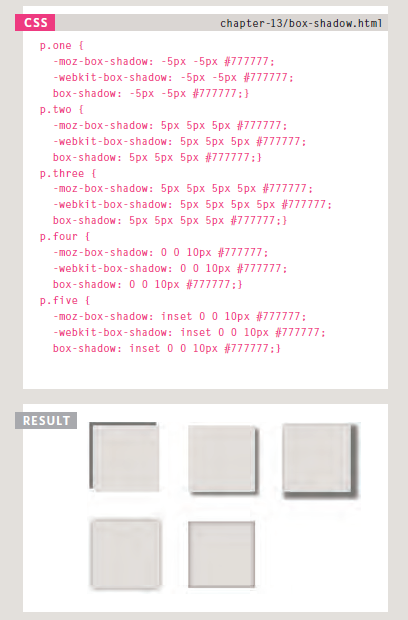
In this example you can see a list. Each item in the list is usually treated as a block-level element, but the rule for the**<li>** elements indicates that they should be treated as inline elements, which means they will sit alongside each other rather than appearing on new lines. This technique is often used to create navigation for a site, and in this example a margin has been added to the right of each of the items to separate them out. The rule that applies to the**<li>** element whose **class** is **coming-soon** has been hidden as if it were not in the page at all.

****

If the **visibility** of an elements set to **hidden**, a blank space will appear in its place. If you do not want a blank space to appear, then you should use the **display** property with value of **none** instead (as covered on the previous page).

**BOX-Shadow**

The **box-shadow** property allows you to add a drop shadow around a box. It works just like the **text-shadow** property that you met on page 288. It must use at least the first of these two values as well as a color: Horizontal offset Negative values position the shadow to the left of the box. Vertical offset Negative values position the shadow to the top of the box. Blur distance If omitted, the shadow is a solid line like a border’s read of shad owe used, a positive value will cause the shadow to expand in all directions, and a negative value will make it contract. The **inset** keyword can also be used before these values to create an inner-shadow. Chrome, Firefox, and Safari were quick to support this property using the **-most-box-shadow**land **-web kit-box-shadow** properties. These are not in these specification but using them can help this style to work in these browsers.

****

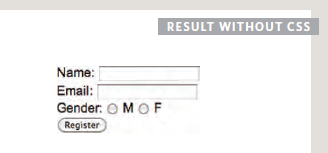
CSS3 introduces the ability to create rounded corners on any box, using a property called **border-radius**. The value indicates the size of the radius in pixels. You can specify individual values for each corner of a box using:**border-top-right-radiusborder-bottom-right-radiusborder-bottom-left-radiusborder-top-left-radius**You can also use a shorthand of these four properties (in clockwise order: top, right, bottom, left). For example:

**border-radius: 5px, 10px,5px, 10px;**

Lists, Tables and Forms:

**text-transform** to convert the content of the table headers to uppercase

Labels for form elements are often different lengths, which means that the form controls will not appear in a straight line.



Each row of the form has a title telling users what they need to enter. For the text inputs, the title is in the **<label>** element.

For the radio buttons, the title is in a **<span>** element. Both have **class** attribute with a value of**title**.We can use a property called **float** to move the titles to the left of the page.

By setting the **width** property on those elements, we know that the titles will each take up the same width. Therefore, the form controls next to them will line up. The **text-align** property issued to align the titles to the right, and **padding** is used to make sure there is a gap between the text in the title boxes and the form controls. Styles are also applied to the**<div>** elements that contain each row of the form (fixing their width and creating vertical space between each row). The submit button is also aligned tithe right.



**LAYOUT:**

Building Blocks

CSS treats each HTML element as if it is in its own box. This box will either be a **block-level** box or an **inline** box. Block-level boxes start on a new line and act as the main building blocks of any layout, while inline boxes flow between surrounding text. You can control how much space each box takes up by setting the width of the boxes (and sometimes the height, too). To separate boxes, you can use borders, margins, padding, and background colors.

****

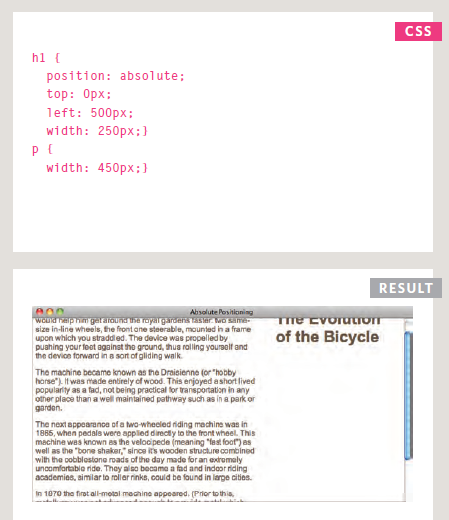
positioning schemes

**position:relative**

Relative positioning moves an element in relation to where it would have been in normal flow. For example, you can move it 10pixels lower than it would have been in normal flow or 20% tithe right. You can indicate that an element should be relatively positioned using the **position** property with a value of **relative**. You then use the offset properties (**top** or **bottom** and **left** or **right**) to indicate how far to move the element from where it would have been in normal flow. To move the box up or down, you can use either the **top** or **bottom** properties. To move the box horizontally, you can use either the **left** or **right** properties. The values of the box offset properties are usually given in pixels, percentages or ems.

**position:absolute**

When the **position** property is given a value of **absolute**, the box is taken out of normal flow and no longer affects the position of other elements onthe page. (They act like it is not there.)The box offset properties (**top** or **bottom** and **left** or **right**)specify where the element should appear in relation to its containing element. In this example, the heading has-been positioned at the top of the page and 500 pixels from its left edge. The width of the heading asset to be 250 pixels wide. The **width** property has also been applied to the **<p>**elements in this example to prevent the text from overlapping and becoming unreadable.

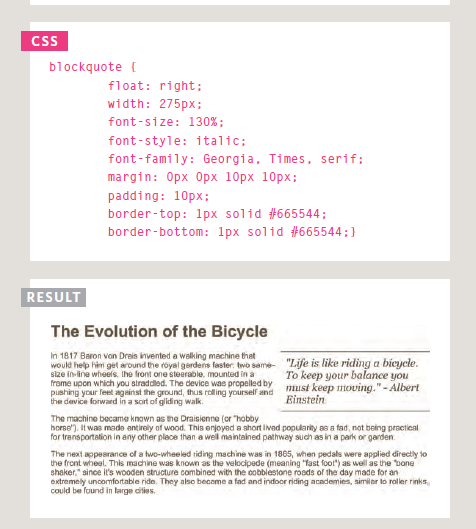
****

**position:fixed**

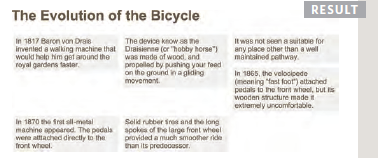
Fixed positioning is a type of absolute positioning that requires the **position** property to have a value of **fixed**. It positions the element in relation to the browser window. Therefore, when a user scrolls down the page, it stays in the exact same place. It is a good idea to try this example in your browser to see the effect. To control where the fixed position box appears in relation to the browser window, the box offset properties are used. In this example, the heading has been positioned to the top left hand corner of the browser window. When the user scrolls down the page, the paragraphs disappear behind the heading. The **<p>** elements are in normal flow and ignore the space that the **<h1>** element would have taken up. Therefore, the **margin-top** property has been used to push the first **<p>**element below where the fixed position **<h1>** element is sitting.

**float**

The **float** property allows you to take an element in normal flow and place it as far to the left or right of the containing element as possible. Anything else that sits inside the containing element will flow around the element that isfloated.When you use the **float** property, you should also use the **width** property to indicate how wide the floated element should be. If you do not, results can be inconsistent but the box is likely to take up the full width of the containing element (just like it would in normal flow).In this example, a**<block quote>** element issued to hold a quotation. It’s containing element is the**<body>** element. The **<block quote>** elements floated to the right, and the paragraphs that follow the quote flow around the floated element

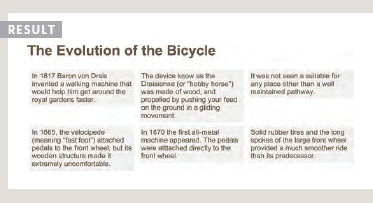
****

In this example, you can see six paragraphs, each of which has w**idth** and a **float** property set. The fourth paragraph does not go across to the left hand edge of the page as one might expect. Rather it sits right under the third paragraph. The reason for this is that the fourth paragraph has space to start under the third paragraph, but it cannot go any further tithe left because the second paragraph is in the way. Setting the height of the paragraphs to be the same height as the tallest paragraph would solve this issue, but itis rarely suited to real world designs where the amount of text in a paragraph or column may vary. It is more common to use the **clear** property.

****

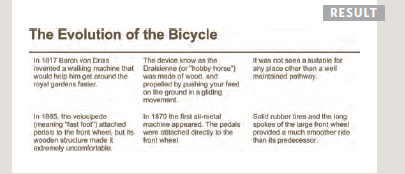
**Clear**

The **clear** property allows you to say that no element (within the same containing element)should touch the left or righthandsides of a box. It can take the following values:**left**The left-hand side of the box should not touch any other elements appearing in the same containing element.**right**The right-hand side of the box will not touch elements appearing in the same containingelement.**both**Neither the left nor right-hand sides of the box will touch elements appearing in the same containing element.**none**Elements can touch either side. In this example, the fourth paragraph has a class called **clear**. The CSS rule for this class uses the **clear** property to indicate that nothing should touch the left-hand side of it. The fourth paragraph is therefore moved further down the pages no other element touches its left-hand side.

****

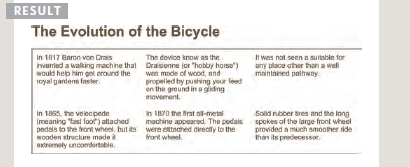
**Problem:**

If a containing element *only* contains floated elements, some browsers will treat it as if it is zero pixels tall. As you can see in this example, the one pixel border assigned to the containing element has collapsed, so the box looks like two pixel line.

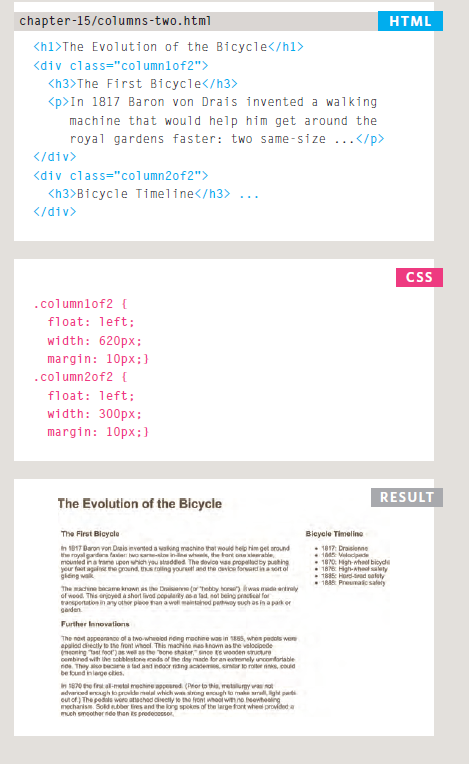
****

The **overflow** property is given a value **auto**.

The **width** property is set to**100%**.



Many web pages use multiple columns in their design. Thesis achieved by using a **<div>**element to represent each column. The following three CSSproperties are used to position the columns next to each other:**width**This sets the width of thecolumns.**float**This positions the columns next to each other.**margin**This creates a gap between thecolumns.A two-column layout like the one shown on this page would need two **<div>** elements, one for the main content of the page and one for the sidebar. Inside each of the **<div>**elements there can be headings, paragraphs, images, and even other **<div>** elements.

****

Similarly, a three column layout could be created by floating three **<div>** elements next teach other, as shown on this page.

Liquid layout🡪percentage

Fixed:px

IMAGES

1: The **float** property is added to the class that was created to represent the size of the image(such as the **small** class in our example).2: New classes are created with names such as **align-left** or **align-right** to align the images to the left or right of the page. These class names are used in addition to classes that indicate the size of the image. In this example you can see the **align-left** and **align-right** classes used to align the image. It is also common to add margin to the image to ensure that the text does not touch their edges.

By default, images are inline elements. This means that they flow within the surrounding text. In order to center an image, it should be turned into a blocklevelelement using the **display** property with a value of **block**. Once it has been made into a block-level element, there are two common ways in which youkan horizontally center an image:1: On the containing element, you can use the **text-align** property with a value of **center**.2: On the image itself, you can use the use the **margin** property and set the values of the left and right margins to **auto**. You can specify class names that allow any element to be centered, in the same way that you can for the dimensions or alignment of images.

The **background-image** property allows you to place an image behind any HTMLelement. This could be the entire page or just part of the page. By default, a background image will repeat to fill the entire box.

The **background-repeat** property can have four values:

**repeat**The background image is repeated both horizontally and vertically (the default way itis shown if the **backgroundrepeat**property isn't used).

**repeat-x** the image is repeated horizontally only (as shown in the first example on the left).

**repeat-y** the image is repeated vertically only.

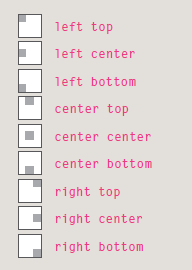
**no-repeat** The image is only shown once.

The **background-attachment** property specifies whether abackground image should stay in one position or move as the user scrolls up and down the page. Incan have one of two values:

**Fixed** The background image stays in the same position on the page

.**scroll** The background image moves up and down as the user scrolls up and down the page.

When an image is not being repeated, you can use the**background-position**property to specify where in the browser window the background image should be placed. This property usually has a pair of values. The first represents the horizontal position and the second represents the vertical.

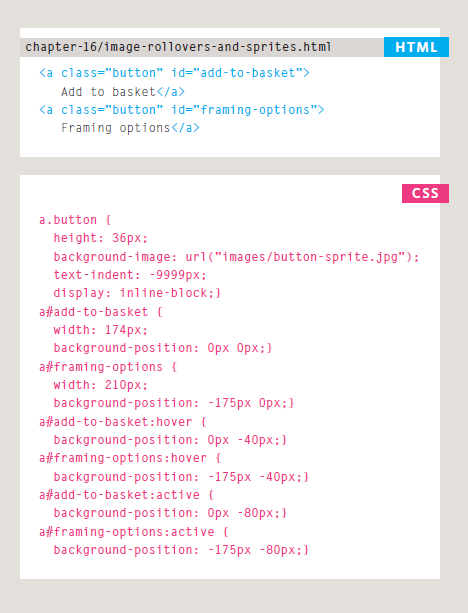
****

If you only specify one value, the second value will default to **center**.The **background** property actslike a shorthand for all of theother background propertiesyou have just seen, and also the**background-color** property.The properties must be specifiedin the following order, but youcan miss any value if you do not want to specify it.

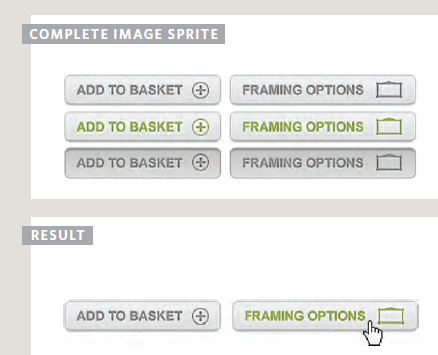
1: **background-color**2: **background-image**3: **background-repeat**4: **background-attachment**5: **background-position**

Image Rollovers & Sprites

Using CSS, it is possible to create link or button that changes to second style when a user moves their mouse over it (known as r**ollover**) and a third style when they click on it.This is achieved by setting abackground image for the link or button that has three different styles of the same button (but only allows enough space to show one of them at a time).You can see the image we arousing in this example on the right. It actually features two buttons on the one image. When the user moves their mouse over the element, or clicks on it, the position of thebackground image is moved to show the relevant image. When a single image is used for several different parts of an interface, it is known as a **sprite**. You can add the logo and other interface elements, as well as buttons to the image. The advantage of using sprites is that the web browser only needs to request one image rather than many images, which can make the web page load faster.

****

In this example, you can see two links that look like buttons. Each of the buttons has three different states. These are all represented in a single image. Because the **<a>** element is an inline element, we set the **display** property of these links to indicate that they should be **inline-block** elements. This allows us to specify the width and height of each **<a>** elements that it matches the size of its corresponding button. The **background-position** property is used to move the image in order to show the button in the right state. When the user hovers over link, the **:hover** pseudo-class has a rule that moves the**background-position** of the image to show a different state for that button. Similarly, when the user clicks on the link, the **:active** pseudoclasshas a rule to show the third state for that button. Touch screen devices will not change a link's state when the user hovers over it, as the screens do not yet have a way to tell when the user is hovering. However, they will change their appearance when the user activates them.

****