CIS 4004: Web Based Information Technology
Fall 2012

Cascading Style Sheets – Page Layout - Part 3

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790

Department of Electrical Engineering and Computer Science
University of Central Florida
The CSS Box Model

- top margin
- top border
- top padding
- left padding
- right padding
- left border
- right border
- bottom padding
- left margin
- right margin
- bottom border
- bottom margin
- Content
The *position* Property

- In the two previous CSS – Page Layout sections of notes, we looked in detail at the box border, padding, and margins, as well as the float and clear properties.

- In this section of notes, we’ll look more closely at the *position* property. The *position* property is at the heart of all CSS-based layouts. The *position* property determines the reference point for the positioning of each element box.

- There are four values for the *position* property: `static`, `absolute`, `fixed`, and `relative`. 
The **position** Property

- We’ll set up a running example demonstration HTML5/CSS3 to illustrate the `position` property.

- The basic HTML5 is shown on the next page, with its rendering on the following page.

- Notice that the default `position` property for any element is **static**.

- In the running example, the third paragraph is a special paragraph (styled differently from the other paragraphs) so that we can see the difference in the various position property values.
This is the first paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed. The key to working with the position property is to understand that every element is positioned with respect to another element; which element that is can be changed by changing the value of the position property.

This is the second paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed.

This is the third paragraph of the positioning demo. This paragraph has an ID so we can change its position value without affecting the other paragraphs. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed.

This is the fourth paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed. I've added a lot of extra text into this paragraph so that the layout is longer than the browser window to better demonstrate the effect of fixed positioning. I've added a lot of extra text into this paragraph so that the layout is longer than the browser window to better demonstrate the effect of fixed positioning. I've added a lot of extra text into this paragraph so that the layout is longer than the browser window to better demonstrate the effect of fixed positioning.
This is the first paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed. The key to working with the position property is to understand that every element is positioned with respect to another element; which element that is can be changed by changing the value of the position property.

This is the second paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed.

This is the third paragraph of the positioning demo. This paragraph has an ID so we can change its position value without affecting the other paragraphs. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed.

This is the fourth paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed. I've added a lot of extra text into this paragraph so that the layout is longer than the browser window to better demonstrate the effect of fixed positioning. I've added a lot of extra text into this paragraph so that the layout is longer than the browser window to better demonstrate the effect of fixed positioning.

This version illustrates static positioning.
Static Positioning

• The default position for any HTML5 element is static.

• With static positioning, each element is simply laid out one after the other (in normal flow), so the paragraphs in our demo appear under each other, with their default margin settings creating the space between them.

• To break away from this sequential (normal flow) layout of the elements provided by the default static positioning, you must change a box’s position property to one of the other three possible values.
Relative Positioning

- Relative positioning allows you to use the `top`, `right`, `bottom` and `left` attributes to move the element with respect to the position in which it would appear using normal flow.

- In our running demo example, notice on the next page that we’ve changed the style for the special third paragraph to now have `position: relative`.

```css
p#specialpara {position: relative;
    top: 30px;
    left: 20px;
}
```
The special paragraph now uses relative positioning. Its top is moved down by 30 pixels and to the right by 20 pixels from where it would appear in normal rendering.

```html
<meta charset="utf-8">
<title>Relative Positioning Demo</title>
<style type="text/css">

body {background-color:#FFC;}

p {border:1px solid #000;}

p#specialpara { position: relative; top: 30px; left: 20px; color:red; background:#EEE;}

</style>

<body>

<p>This is the first paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed. The key to working with the position property is to understand that every element is positioned with respect to another element; which element that is can be changed by changing the value of the position property. </p>

<p>This is the second paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed. </p>

<p id="specialpara">This is the third paragraph of the positioning demo. This
```
Notice that although the element moves relative to its original position, nothing else changes. The space occupied by the original static element is retained, as is the positioning of the other elements.
An even more drastic relative position move, with the top set to 120 pixels. Notice that although the element moves relative to its original position, nothing else changes. The space occupied by the original static element is retained, as is the positioning of the other elements.
Negative values also work which have the effect of moving an element up and to the left. In this case: top was set to -40px and left was set to -20px.
Relative Positioning

• The thing to remember about relative positioning is that if you move an element in this manner, you must allow space for it.

• Using the example on page 10, you might take the next step of adding a margin-top value of 30 pixels or greater to the fourth paragraph in order to move it down, thus preventing it from being overlapped by the repositioned third paragraph. (See next page.)
The fourth paragraph is now styled to have a margin-top: 40px. Which moves it out from under the relocated third paragraph.
Absolute Positioning

- Absolute positioning is a whole different beast from static and relative positioning, since this type of positioning takes an element entirely out of normal flow.

- With absolute positioning, the default positioning context is the body of the document.

- In the running demo, we’ll modify the special paragraph to be positioned absolutely.

  ```css
  p#specialpara {position: absolute;
                      top: 30px;
                      left: 20px;
  }
  ```
The special paragraph now uses absolute positioning. Its top is set at 30 pixels from the top of the body element and its left side is set at 20 pixels from the left side of the body element.
Notice that the space previously occupied by the absolutely positioned element is gone. The absolutely positioned element has become entirely independent of the surrounding elements in the markup and is now positioned with respect to the `<body>` element.
Positioning Context

• The default positioning context of an absolutely positioned element is the body element.

• As the screen shot on the previous page illustrates, the offset provided by the top and left attribute values moves the element with respect to the body element – the top ancestor container in the markup hierarchy – not with respect to the element’s default position in the document flow (as is the case with relative).

• The next slide illustrates the same example with top:50px and left:80px.
This is the first paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed. The key to working with the position property is to understand that every element is positioned with respect to another element: which element that is can be specified. This paragraph has an ID so we can change its position value without affecting the other paragraphs. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed.

This is the fourth paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed. I've added a lot of extra text into this paragraph so that the layout is longer than the browser window to better demonstrate the effect of fixed positioning. I've added a lot of extra text into this paragraph so that the layout is longer than the browser window to better demonstrate the effect of fixed positioning.
Positioning Context

• Because the absolutely element’s positioning context is body, the element moves when the page is scrolled to retain its relationship to the body element, which also moves when the page scrolls.

• Before we look at how to use a different element than body as the positioning context for an absolutely positioned element, let’s look at the last of the four positioning properties – fixed positioning.
Fixed Positioning

- Fixed positioning is similar to absolute positioning, except that the element’s positioning context is the viewport (the browser window or the screen of a handheld device, for example), so the element does not move when the page is scrolled.

- To really see this effect, you’ll need to download the demo XHTML/CSS documents from this set of notes and pay particular attention to the fixed positioning example.
The special paragraph now uses fixed positioning. Its top is set at 25 pixels from the top of the browser window and its left side is set at 30 pixels from the left side of the browser window.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Fixed Positioning Demo</title>
    <style type="text/css">
      body {background-color:#FFF;}
      p {border:1px solid #000;}
      p#specialpara { position: fixed; top: 25px; left: 30px; color:red; background:#EEE; }
    </style>
  </head>
  <body>
    <p>This is the first paragraph of the positioning demo. The objective is to demonstrate the difference between the four position property values: static, relative, absolute and fixed. The key to working with the position property is to understand that every element is positioned with respect to another element; which element that is can be changed by changing the value of the position property.</p>
  </body>
</html>
```
Note that the paragraph has remained at a position 25px from the top and 30 pixels from the left even though the body element has scrolled.
Fixed Positioning

• This “nailed-to-the-browser” effect enables you to simulate the effect of what are now deprecated frames (recall the three flavors of XHTML: Strict, Transitional, and Frameset).

• For example, you can now create a navigation element that stays put on the page when the page scrolls without the problems that were associated with managing multiple documents in a frameset (the old way of doing this).

• **NOTE:** the fixed position property does not work in IE6, but does work in IE7 and above.
More On Positioning Context

• Now that we’ve seen all four types of positioning, let’s go back and look at positioning context in more detail.

• Simply put, contextual positioning means that when you move an element using the attributes top, right, bottom, or left, you are moving that element with respect to another element. That other element is known as its positioning context.

• As we saw in the example on page 16, for absolute positioning, the default positioning context for an absolutely positioned element is body, unless you change it.
More On Positioning Context

• The body element is the containing element of all other elements in your markup, but you can use any ancestor element as a positioning context of another element by changing the ancestor’s position value to relative.

• Consider the markup shown on the next page and its rendering on the following page.

• QUESTION: Why isn’t the inner <div> 10 pixels down from the top of the outer <div> and 20 pixels to the left, as specified in the CSS?
<title>Positioning Context Demo - Version 1</title>

<style type="text/css">

body {background-color:#FFC;}
div#outer_div {width:250px; margin:100px 40px; border-top:3px solid red;}
div#inner_div {top:10px; left:20px; background:#AAA;}
#ruler {position:relative; left:-58px; top:0px; margin-bottom:5px;}

</style>

</head>

<body>

<img id="ruler" src="ruler_1000px.gif" alt="ruler" />

<div id="outer_div">

<div id="inner_div">This is some text for a paragraph to demonstrate contextual positioning. Here are two divs, one nested in the other. The outer div has red top border and the inner one has a gray background. Both elements have default static positioning.</div>

</div>

</div>

</body>

</html>
Positioning Context

The outer `<div>` has a solid red 3 pixel border. And it can be seen setting behind the inner `<div>`.

Why do the two `<div>` elements share the same origin (top-left) point?

Answer on the next page.
Positioning Context

• The answer to the question posed in the last example, is that the inner (and irrelevantly, the outer) `<div>` element has the default positioning of `static`. This means it is rendered in normal flow, and because the outer `<div>` has no content, the inner `<div>` starts in the same place.

• Only when you set an element to one of the other three positioning options – `relative`, `absolute`, or `fixed`, - do the `top`, `right`, `bottom`, and `left` attribute values actually do anything.

• To illustrate this fact, consider the modified markup shown on the next page, where the `left` and `top` attribute values have been reset for the inner `<div>`. Notice that since we left it with its default position it didn’t move!
Greatly different values for top and left attributes
The two <div> elements are still in exactly the same relative positions, even though the top and left attribute values are quite a bit different between version 1 and version 2.

In version 1 top = 10px and left = 20px

In version 2 top = 100px and left = 200px
Positioning Context

• Now let’s see what happens if we set the inner `<div>` element’s `position` property to `absolute`.

• We’ll modify the CSS to be:

```css
body {background-color:#FFC;}

div#outer_div {width:250px; margin:100px 40px; 
border-top:3px solid red;}

div#inner_div{position:absolute; top:10px;
left:20px; background-color:#AAA;}
```

• The inner `<div>` element is now absolutely positioned, but with respect to what? Where do you expect the inner `<div>` element to be positioned?
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Positioning Context Demo - Version 3</title>
  <style type="text/css">
    body {background-color:#FFC;}
    div#outer_div {width:250px; margin:100px 40px; border-top:3px solid red;}
    div#inner_div {position:absolute; top:10px; left:20px; background:#AAA;}
    #ruler {position:relative; left:-58px; top:0px; margin-bottom:5px;}
  </style>
</head>
<body>
  <img id="ruler" src="ruler_1000px.gif" alt="ruler" />
  <div id="outer_div">
    <div id="inner_div">This is some text for a paragraph to demonstrate contextual positioning. Here are two divs, one nested in the other. The outer div has red top border and the inner one has a gray background. Both elements have default static positioning.</div>
  </div>
</body>
</html>
This is some text for a paragraph to demonstrate contextual positioning. Here are two divs, one nested in the other. The outer div has red top border and the inner one has a gray background. Both elements have default static positioning.
Positioning Context

• As you can see on the previous page, since there is no other relatively positioned element for the inner `<div>` to reference, it positions itself by default with respect to the `<body>` element (so it is overlayed over the ruler).

• The top border of the outer `<div>` is set to red so you can see where it is located. Its margins push it 50 pixels down and 40 pixels to the left of the top corner of the browser window.

• Because the inner `<div>`’s position property is set to absolute, it is positioned relative to the `<body>` element, because `<body>` is the default positioning context.
Positioning Context

- In other words, the inner `<div>` element entirely ignores its parent (the outer `<div>` element), and its `top` and `left` attributes offset it with respect to the `<body>` element, as shown in the rendering on pages 34 and below.

This `<div>` has position `static` by default. Thus the inner `<div>` has no relatively positioned element to base its absolute position on other than the default positioning context of the `<body>` element.
**Positioning Context**

- As the final example for explaining positioning context, let’s now set the outer `<div>` element’s position property to `relative`.

- This will now cause the positioning context of the absolutely positioned inner `<div>` element to become the outer `<div>` element in which it is nested.

- This means the setting the `top` and `left` attributes of the inner `<div>` element now positions it with respect to the outer `<div>` element.
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Positioning Context Demo - Version 4</title>
    <style type="text/css">
      body {background-color:#FFC;}
      div#outer_div {position: relative; width:250px; margin:100px 40px; border-top:3px solid red;}
      div#inner_div {position:absolute; top:10px; left:20px; background-color:#AAA;}
      #ruler {position:relative; left:-58px; top:0px; margin-bottom:5px;}
    </style>
  </head>
  <body>
    <img id="ruler" src="ruler_1000px.gif" alt="ruler" />
    <div id="outer_div">
      <div id="inner_div">
        This is some text for a paragraph to demonstrate contextual positioning. Here are two divs, one nested in the other. The outer div has red top border and the inner one has a gray background. Both elements have default static positioning.
      </div>
    </div>
  </body>
</html>
Once the outer `<div>` has a relative positioning property set, absolutely positioned descendants position themselves relative to it, as defined by their `top` and `left` attributes.

This is some text for a paragraph to demonstrate contextual positioning. Here are two `divs`, one nested in the other. The outer `div` has red top border and the inner one has a gray background. Both elements have default static positioning.
Positioning Context

• If you set the top and left attribute values of the outer `<div>` element to anything other than 0, the inner `<div>` would move to maintain its positioning relationship to the outer `<div>`, which is its positioning context.

• This last example more clearly illustrates this (it really is the last example this time).

• In this very last example, we’ll reset the margins of the outer `<div>` element drastically from their original position. The thing to notice is how the inner `<div>` element moves with respect to the new position of the outer `<div>`.
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Positioning Context Demo - Version 5</title>
  <style type="text/css">
    body {background-color:#FFC;}
    div#outer_div {position: relative; width:250px; margin:250px 300px; border-top:3px solid red;}
    div#inner_div {position:absolute; top:10px; left:20px; background-color:#AAA;}
    #ruler {position:relative; left:-58px; top:0px; margin-bottom:5px;}
  </style>
</head>
<body>
  <img id="ruler" src="ruler_1000px.gif" alt="ruler" />
  <div id="outer_div">
    <div id="inner_div">This is some text for a paragraph to demonstrate contextual positioning. Here are two divs, one nested in the other. The outer div has red top border and the inner one has a gray background. Both elements have default static positioning.</div>
  </div>
</body>
</html>
This is some text for a paragraph to demonstrate contextual positioning. Here are two divs, one nested in the other. The outer div has red top border and the inner one has a gray background. Both elements have default static positioning.
The **display** Property

- Just as every element has a `position` property, every element also has a `display` property.

- Although there are quite a number of `display` property values, the most commonly used elements have a default `display` property value of either **block** or **inline**.

- **Block elements**, such as paragraphs, headings, and lists, sit one above another when displayed in the browser.

- **Inline elements**, such as `anchor`, `span`, and `img`, sit side-by-side when they are displayed in the browser and only appear on a new line if there is insufficient room on the previous line.
The **display** Property

- The ability to change block elements to inline elements, and vice versa is a powerful capability that allows you, for example, to force an inline element to fill its containing element.

- Changing an element’s display property is done like this:

```css
p {display: inline; }
```

<table>
<thead>
<tr>
<th>Block by default</th>
</tr>
</thead>
<tbody>
<tr>
<td>p {display: inline; }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inline by default</th>
</tr>
</thead>
<tbody>
<tr>
<td>a {display: block; }</td>
</tr>
</tbody>
</table>

© Dr. Mark Llewellyn
The **display** Property

- The other value for the `display` property that is worth discussing here is `none`.

- When an element’s `display` property is set to `none`, that element, and any elements nested inside it, are not displayed on the page. Any space that was occupied by the element is removed; its as if the related markup did not exist.

- NOTE: This contrasts with the visibility property, which simply has the values `visible` or `hidden`. If an element’s visibility is set to `hidden`, the element is hidden, but the space it occupied remains. We’ll see more on this later.
<title>Display Property Demo</title>

<style type="text/css">
!--
.box1 { width: 200px; height: 200px; background-color: #cccccc; border: dashed; padding: 5px; }
.box2 { width: 100px; height: 100px; background-color: #ffffff; border: ridge; padding: 5px; }
.box3 { width: 150px; height: 150px; background-color: blue; border: solid; padding: 5px; color: white; }
-->
</style>
</head>
<body>
<div class="box1">
    This is the first box.
</div>
<div class="box2">
    This is the second box.
</div>
<div class="box3">
    This is the third box.
</div>
</body>
All three `<div>` elements display in their normal block style sitting one on top of another. All are displayed.
The `<div>` element styled using class `box2` has its display property set to `none`.

Notice on the next page that the space that would have been occupied by the second box has disappeared and `box3` moves into that space.
This is the first box.

This is the third box.
The z-index Property

• The z-index property is used to modify the stacking order of elements on a Web page.

• When using only HTML5 there is no easy way to “stack” elements other than configuring backgrounds for pages or tables.

• The z-index property provides flexibility in the display of elements.

• The default z-index value is “0”. Elements with higher z-index values will appear stacked on top of elements with lower z-index values rendered on the same position of the page.

• The Web page shown on the next page is configured using absolute positioning and z-index properties. The HTML5 code is shown on the following page.
<head>
  <meta charset="utf-8">
  <title>Z-index Property Demo - version 1</title>
  <style type="text/css">
    /*
    #thor {position:absolute; left:100px; top:100px; z-index:1;}
    #cat {position:absolute; left:200px; top:150px; z-index:2;}
    #alex {position:absolute; left:300px; top:200px; z-index:3;}
    #ruler {position:relative; left:-51px; top:0px; margin-bottom:5px; }
    */
  </style>
</head>

<body>
  <div>
    <img id="ruler" src="ruler_1000px.gif" alt="ruler" />
    <div id="thor">
      <img src="thor.jpg" alt="Thor Hushovd" height="150" width="150" />
    </div>
    <div id="cat">
      <img src="cat.jpg" alt="Cat Osterman" height="150" width="150" />
    </div>
    <div id="alex">
      <img src="alex.jpg" alt="Alex the Parrott" height="150" width="150" />
    </div>
  </div>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Z-index Property Demo - version 2</title>
<style type="text/css">
/*
#thor { position:absolute; left:100px; top:100px; z-index:3; }
#cat { position:absolute; left:200px; top:150px; z-index:2; }
#alex { position:absolute; left:300px; top:200px; z-index:1; }
#ruler {position:relative; left:-51px; top:0px; margin-bottom:5px; }
-->
</style>
</head>
<body>
<div>
<img id="ruler" src="ruler_1000px.gif" alt="ruler" />
<div id="thor">
<img src="thor.jpg" alt="Thor Hushovd" height="150" width="150" />
</div>
<div id="cat">
<img src="cat.jpg" alt="Cat Osterman" height="150" width="150" />
</div>
<div id="alex">
<img src="alex.jpg" alt="Alex the Parrott" height="150" width="150" />
</div>
</div>
</body>
</html>
Backgrounds

• One final aspect of positioning elements is backgrounds, which provide a means of adding color and images into an element’s background.

• If you have ever worked with a graphics program like Adobe Photoshop or Adobe Fireworks, you will be familiar with the concept of layers.

• Every element box can be thought of as having two layers. An element’s foreground layer is made up of the content of the element (such as text or an image) and the border of the box. The element’s background layer can be filled with a solid color, using the `background-color` property, and can also contain any number of images, using the `background-image` property, which stacks the images on top of the background color.
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Background Demo - Version 1</title>
  <style type="text/css">
    p {
      font-size: 28px; font-family: helvetica, arial, sans-serif; font-weight: bold;
      width: 410px; height: 220px; margin: 20px auto; padding: 10px;
      color: red; border: 4px solid #aaa; background-color: gray;
      background-image: url(blue_circle2.gif);
    }

    body {background-image: url(imagePath/imageName);}
  </style>
</head>
<body>
  <p>Background images tile across and down from the top left corner.</p>
</body>
</html>
Background images tile across and down from the top left corner.

Notice that the tiled images are cutoff on the right side and bottom due to the size of the box.
Backgrounds

• The default settings of across and down repeating and top left origin position can be changed by `background-repeat` and `background-position` respectively.

• There are four possible values for `background-repeat`.
  - `repeat` is the default value, which as shown in the previous example, repeats the image horizontally and vertically as many times as needed to fill the encompassing element.
  - `repeat-x`, controls horizontal repeating.
  - `repeat-y`, controls vertical repeating.
  - `no-repeat`, causes the image to display only one time.

• The example on the next page illustrates each of these properties.
background-repeat: repeat
(default cause)
The image tiles across and
down from the top left corner.

background-repeat: repeat-x
The image tiles across the
background area.

background-repeat: repeat-y
The image tiles down the
background area.

background-repeat: no-repeat
The image displays only once.
NOTE

CSS3 offers a couple of, as of yet mostly unsupported, ways to make the repeats fill the element an exact number of times.

background-repeat: round rescales the image until the repeats fill an exact number of times.

background-repeat: space adds space between the tiles until they fit the element exactly.

Currently, Opera supports both round and space. See next page for an example.
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Background Demo - Version 2</title>
  <style type="text/css">
    @
    p#one { font-size:28px; font-family:helvetica, arial, sans-serif; font-weight:bold; width:410px; height:120px; margin:20px auto; padding:10px; color:red; border:4px solid #aaa; background-color:gray; background-image:url(blue_circle2.gif); background-repeat:repeat; }
    p#two { font-size:28px; font-family:helvetica, arial, sans-serif; font-weight:bold; width:410px; height:120px; margin:20px auto; padding:10px; color:red; border:4px solid #aaa; background-color:gray; background-image:url(blue_circle2.gif); background-repeat:round; }
    p#three { font-size:28px; font-family:helvetica, arial, sans-serif; font-weight:bold; width:410px; height:120px; margin:20px auto; padding:10px; color:red; border:4px solid #aaa; background-color:gray; background-image:url(blue_circle2.gif); background-repeat:space; }
  </style>
</head>
<body>
  <p id="one">background-repeat:repeat; (default, cause)<br />
The image tiles across
</p>
</body>
</html>
background-repeat: repeat
(default cause)
The image tiles across and
down from the top left corner.

background-repeat: round
The image is resized to fit an
integral number of times.

background-repeat: space
Space is added between the
images to achieve an integral
number of tiles.
Backgrounds

• The default settings of across and down repeating and top left origin position can be changed by `background-repeat` and `background-position` respectively.

• There are four possible values for `background-repeat`.
  - `repeat` is the default value, which as shown in the previous example, repeats the image horizontally and vertically as many times as needed to fill the encompassing element.
  - `repeat-x`, controls horizontal repeating.
  - `repeat-y`, controls vertical repeating.
  - `no-repeat`, causes the image to display only one time.

• The example on the next page illustrates each of these properties.