

Chapter 8: CSS3 and HTML5

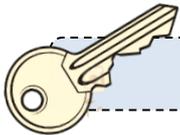
Learning Outcomes:

- ✓ Identify and implement a range of properties from the CSS3 standard
- ✓ Identify the key differences in the HTML5 standard
- ✓ Implement HTML5 elements into a web page



Prerequisite Knowledge:

- ✓ Complete Chapter 7
- ✓ Be confident in applying a wide range of CSS properties using different selectors
- ✓ Be able to link a web page to external CSS
- ✓ Be able to construct simple web page layouts



Web Kit

Compatibility

Gradient

Argument

Parameter

Semantic

Keywords

8.1 Theory: CSS3 Today

CSS3 is the latest standard of CSS; in fact, (at the time of this text) it is still under development by the W3C (World Wide Web Consortium). Despite being in the process of development, many of the CSS3 standards have already been implemented by most modern-day browsers. For reference, CSS3 has been demonstrated in the previous chapter, Chapter 7, where the `border-radius` property was used to round the borders of HTML divisions. The `text-shadow` property was another CSS3 property previously demonstrated. These CSS properties are new to the CSS3 standard; if these properties were demonstrated in earlier browser versions, borders would appear unrounded and text would appear without a shadow. In essence, CSS3 is a collection of new properties that are accessible to web developers (without the need to use complex Flash or JavaScript). A key advantage of CSS3 is that it is backwards compatible with the earlier versions; put simply, it includes the CSS properties from the previous versions of CSS too. In this chapter, CSS3 properties will be examined and demonstrated. Unfortunately, as there are so many new properties, not all of them can be covered; therefore, a selection of the 'coolest' ones will be shown instead. For reference purposes, all of the new CSS3 properties are available on the W3Schools website, accessible here: <http://www.w3schools.com>

CSS3 Compatibility

When writing CSS, the aim should be to write syntax that is backwards-compatible; syntax that will be visually consistent on all browsers and versions. Unfortunately, this is often easier said than done. However, there are techniques that aid backwards compatibility. One such technique is using the 'web kits' that are made available for individual browsers; web kits are prefixed properties that apply CSS syntax to specific browsers, while the same syntax is ignored by other browsers. They are typically used to implement proprietary CSS features, prior to their official release by the W3C. The idea behind the concept is that any visual inconsistencies can be removed between browsers, until the prefixed properties are eventually finalised and implemented into the browser. For example, a rounded border can be applied in older browser versions by using web kits. For reference, up-to-date browsers will ignore the web kits and use the non-prefixed CSS property instead; it is generally considered good practice to place the non-prefixed property after the prefixed, web kit properties. See example below:

```
.myclass{
    -moz-border-radius:2em; /*For older versions of Firefox*/
    -ms-border-radius:2em; /*For older versions of Internet Explorer*/
    -o-border-radius:2em; /*For older versions of Opera*/
    -webkit-border-radius:2em; /*For older versions of Chrome and Safari*/
    border-radius:2em; /*For modern browsers*/
}
```

8.2 Practical: New CSS3 Properties

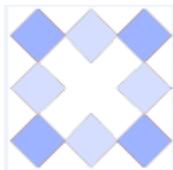
Borders

CSS3 has some new additional properties that can be used to style the borders of a HTML element; one such familiar property is the `border-radius` property, as previewed in the previous chapter. However, there are also two other useful border properties in CSS3, and they are the `box-shadow` property and the `border-image` property. The `box-shadow` property adds a shadow effect to the outline of a HTML element. The `border-image` property creates a border from a specified image. Both of these properties are demonstrated in the syntax below:

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS3 is cool!</title>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="my_style_sheet.css">
</head>
<body>
  <div id="mydiv"></div>
  <div id="mydiv2"></div>
</body>
</html>
```

```
#mydiv{
  height:100px;
  width:50%;
  margin-left:auto;
  margin-right:auto;
  margin-bottom:30px;
  background-color:pink;
  box-shadow:11px 11px 5px #00FF00;
}
#mydiv2{
  height:100px;
  width:50%;
  margin-left:auto;
  margin-right:auto;
  background-color:pink;
  border:30px; /*this property is still required!*/
  border-image:url(border.png) 30 30 30 30 round;
}
```

The image used to create the border applied to the second division is shown below:



HTML Tips!

When setting the `box-shadow` property, the first argument specifies, in pixels, the distance of the shadow to the right. The second is the distance of the shadow downwards, the third is the blur radiance (in pixels) and the final argument is the colour (in hexadecimal).

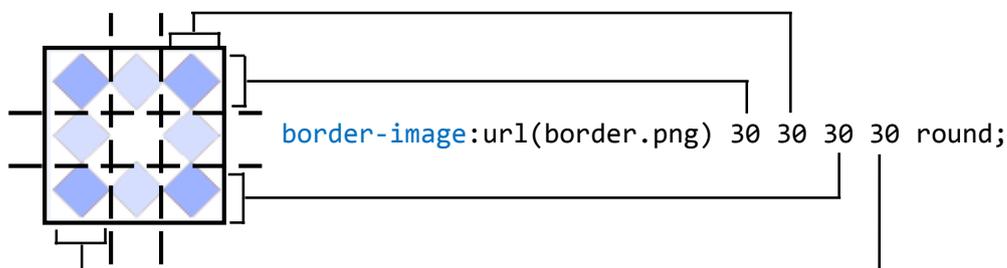


HTML Tips!

Note that when setting the `border-image` property, the value is set in pixels, but for some strange reason, the 'px' unit is not specified!



When setting the `border-image` property, the `30 30 30 30 round;` syntax specifies, in order, the top, right, bottom and left width of the image border; from these arguments the browser can determine which part of the image to show to create the border effect. Note that the space left between the opposite sides will be used to determine the middle section of the border; for example, if an image is 90 pixels wide and 90 pixels high, and all arguments are set to 30 pixels, this would leave 30 pixels in the centre of each side; this remaining part of the image would be used to create the middle border.



Finally, the `round` value is used to round the pixel values evenly so that the image border does not appear abruptly cut off. For reference purposes, it is worth trying the `border-image` property with and without the `round` value to witness the effect of it first-hand.

Remember, like all properties in this chapter, that the `border-image` property is part of the new CSS3 specification and thus may not exist with earlier browser versions. However, note that this property is available as part of the web kits for earlier versions of Safari and Opera, and thus the syntax could be amended to the following for greater compatibility:

```

...
border:30px; /*this property is still required!*/
-webkit-border-image:url(border.png) 30 30 30 30 round;
-o-border-image:url(border.png) 30 30 30 30 round;
border-image:url(border.png) 30 30 30 30 round;
...

```

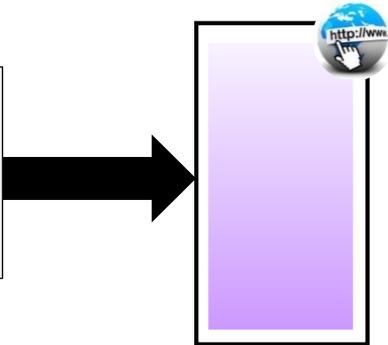
Gradients

In Chapter 7, a gradient background was added to the 'advanced web page' by using a small .png image, combined with the `background-image` and `background-repeat` properties. In CSS3, this technique is not necessarily required, as gradient effects can now be created in the browser itself, without the need to use an image. This is achieved by using the `background` property and setting the values accordingly. Another cool feature, part of CSS3, is that these gradients do not necessarily have to be linear in fashion; they can also be radiant in appearance. Below is the syntax that can be used to create a liner gradient:

```

#mydiv{ /*Top to Bottom*/
  height:200px;
  width:100px;
  background:linear-gradient(#F9F3FF, #CC99FF);
}

```



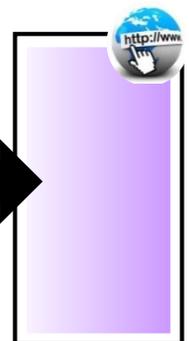
Similar to the `border-image` property, although this is a new CSS3 property it is still available in earlier versions of web browsers, by using the individual web kits that are available for each web browser. Therefore, to help resolve compatibility issues, the above syntax would be better written as:

```
#mydiv{ /*Top to Bottom*/
  height:200px;
  width:100px;
  background:-webkit-linear-gradient(#F9F3FF, #CC99FF); /*For early Safari*/
  background:-o-linear-gradient(#F9F3FF, #CC99FF); /*For early Opera*/
  background:-moz-linear-gradient(#F9F3FF, #CC99FF); /*For early Firefox*/
  background:linear-gradient(#F9F3FF, #CC99FF); /*keep me last! Modern day*/
}
```

The previous example created a gradient from top to bottom. However, it is also possible to change the direction of the gradient; for example, a horizontal gradient (in this case left to right), as opposed to a vertical gradient. The CSS syntax to achieve this is demonstrated below. This syntax also includes the more 'compatible' version of the CSS syntax:

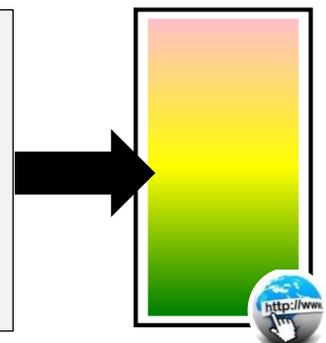
```
#mydiv{ /*Left to Right*/
  height:200px;
  width:100px;
  background:-webkit-linear-gradient(left, #F9F3FF, #CC99FF);
  background:-o-linear-gradient(right, #F9F3FF, #CC99FF);
  background:-moz-linear-gradient(right, #F9F3FF, #CC99FF);
  background:linear-gradient(to right, #F9F3FF, #CC99FF);
}
```

```
analytics" href="no
target="_blank"><
src="http://c.com/"
alt="analytics"></i
```



In CSS3, it is also possible to create gradients that consist of more than two colours. This is achieved by specifying additional colour arguments, separated by the comma. An example of this is demonstrated in the CSS syntax below, but be aware that it is possible that more than three colours can be used. Again, the more compatible version is written for reference purposes.

```
#mydiv{ /*Top to Bottom*/
  height:200px;
  width:100px;
  background:-webkit-linear-gradient(pink, yellow, green);
  background:-o-linear-gradient(pink, yellow, green);
  background:-moz-linear-gradient(pink, yellow, green);
  background:linear-gradient(pink, yellow, green);
}
```

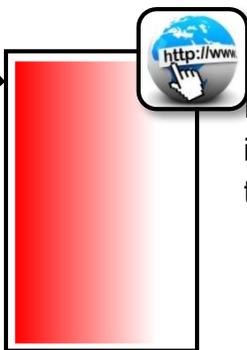


HTML Tips!

A linear gradient does not have to be just vertical or horizontal. For example, angles can be created by using either the key word 'bottom right' (note that this differs for different web kits) or by entering the number of degrees followed by the 'deg' unit; for example, '180deg'.

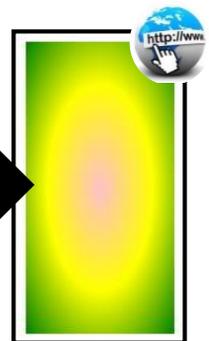
One of the most useful features in CSS3, when creating gradients, is the ability to be able to blend the gradient into the background; this is visually effective and can be used to produce fading effects. This is achieved by using the 'rgba' function; this function is used to specify a colour (by specifying a red, green and blue value between 0 and 255) and setting the transparency. The transparency is the last argument, and can be set to either 0 (for transparent) or 1 (for full colour). Below is the CSS syntax that can be used to create a gradient with a fading effect into transparency:

```
#mydiv{ /*Left to Right*/
height:200px;
width:100px;
background:-webkit-linear-gradient(left, rgba(255,0,0,1), rgba(255,0,0,0));
background:-o-linear-gradient(right, rgba(255,0,0,1), rgba(255,0,0,0));
background:-moz-linear-gradient(right, rgba(255,0,0,1), rgba(255,0,0,0));
background:linear-gradient(to right, rgba(255,0,0,1), rgba(255,0,0,0));
}
```



In CSS3, it is also possible to create radial gradients (as opposed to linear gradients), and this is achieved by changing the property value keyword from 'linear', to 'radial'. The CSS syntax to achieve this is demonstrated in the box below:

```
#mydiv{ /*Radial*/
height:200px;
width:100px;
background:-webkit-radial-gradient(pink, yellow, green);
background:-o-radial-gradient(pink, yellow, green);
background:-moz-radial-gradient(pink, yellow, green);
background:radial-gradient(pink, yellow, green);
}
```



Activity 8.1

Create a web page that includes a suitable title and three divisions that are centred and span 80% of the page width. Ensure there is a suitable gap between each division. The first division should have a box shadow and a radial gradient background; use any colours for the border and radial gradient. The second division should have a border image (use any suitable image) and should have a faded gradient (from colour to transparent). The final division can have any border type; however, the background colour should be a multicoloured gradient.

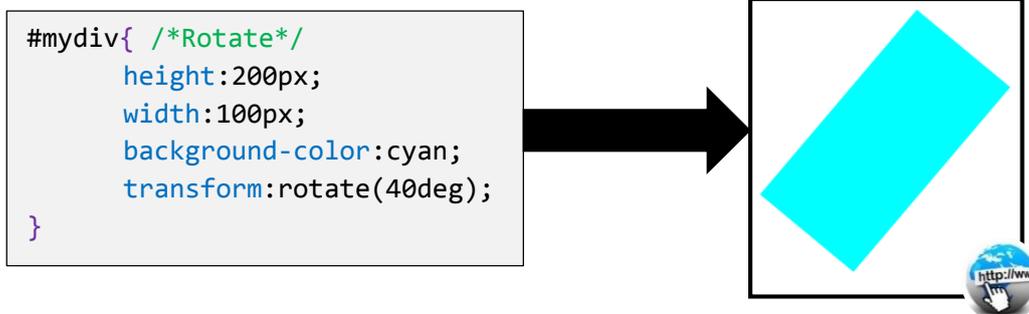


HTML Tips!

Be aware that Internet Explorer version 9 and earlier does not support gradients.

Transforms

Since the early CSS days, there have been some serious innovations, one of which being the ability to 'transform' HTML elements like never before. For example, it is now possible to turn, scale, spin and stretch elements. In CSS3, it is possible to apply both 2D and 3D transformations to HTML elements, although, this text will focus primarily on 2D transformations, the first of which is the 'rotate' method. The syntax for this is demonstrated below:

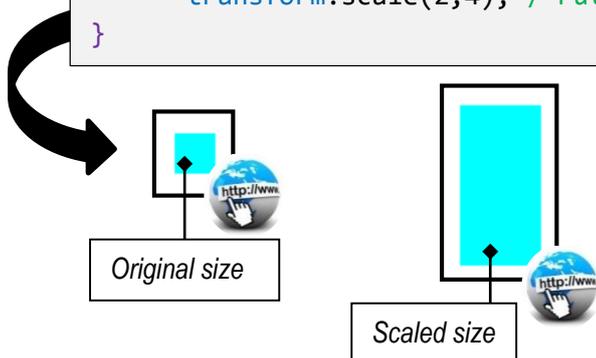


In the example above, the division is rotated 40 degrees. Although the `transform` property is part of the CSS3 specification, this feature is also available in earlier versions of Internet Explorer, Chrome, Safari and Opera. The more compatible version of this syntax is demonstrated below:

```
#mydiv{ /*Rotate*/
  height:200px;
  width:100px;
  background-color:cyan;
  -ms-transform:rotate(40deg); /*For early IE*/
  -webkit-transform:rotate(40deg); /*For early Safari, Chrome & Opera*/
  transform:rotate(40deg); /*Put me last! Modern day browsers!*/
}
```

Another cool transform method is 'scale'. This can be used to significantly increase or decrease the size of a HTML element; this is based on arguments that are given to the width and height parameters. For example, this syntax `scale(2,4)` would result in the width being twice its original size, and the height would be four times the original size. An example of how to use this property, including the more compatible version, is demonstrated below:

```
#mydiv{ /*Scale*/
  height:20px;
  width:20px;
  background-color:cyan;
  -ms-transform:scale(2,4); /*For early IE*/
  -webkit-transform:scale(2,4); /*For early Safari, Chrome & Opera*/
  transform:scale(2,4); /*Put me last! Modern day browsers!*/
}
```

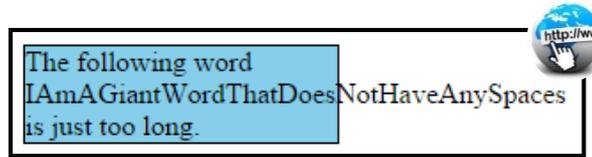


Activity 8.2

Create a web page that includes two divisions, each with a different background colour. Enter a small amount of text into each division, before using the rotate and scale methods to transform their appearance, respectively.

Word Wrapping

CSS can sometimes be frustrating, especially when the visual results are not at all as expected. In the early days of CSS, if a single word in a paragraph was too long and the containing element too small, the word would not naturally text-wrap and would 'overspill' onto the page (and outside the HTML element). Not only was this extremely annoying, it was also very ugly!



Thankfully, CSS3 has a new property called `word-wrap` that can be used to wrap words onto the next line that are too long. The syntax to achieve this is demonstrated below:



Transitions

The W3C believe that the future of the web is HTML5 and CSS3. They believe in the notion that web pages are interactive and accessible on any device, regardless of the browser type or device manufacturer. Although the world is moving towards this concept, there are still many accessibility problems; for example, Flash (a platform used to create interactive content) is a popular technology, but unfortunately is not supported by many devices. HTML5 and CSS3 aim to rectify this by making the web a place where no additional plugins or platforms are required to create interactive content.

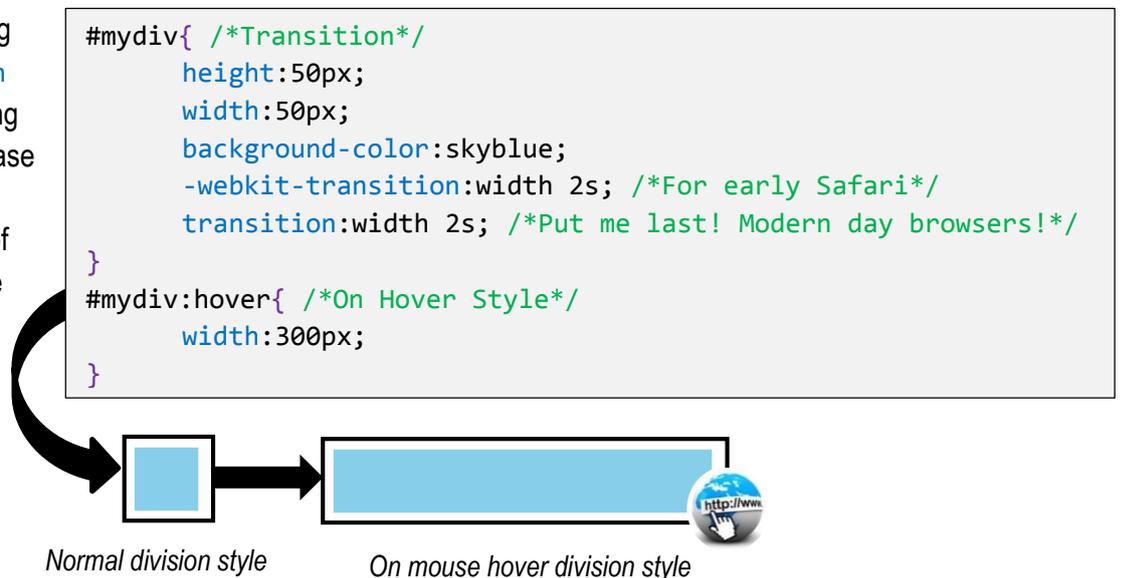


HTML Tips!

The value 2s is short for two seconds. Also, be aware that the transition CSS property is not available in many of the earlier browsers, including Internet Explorer.

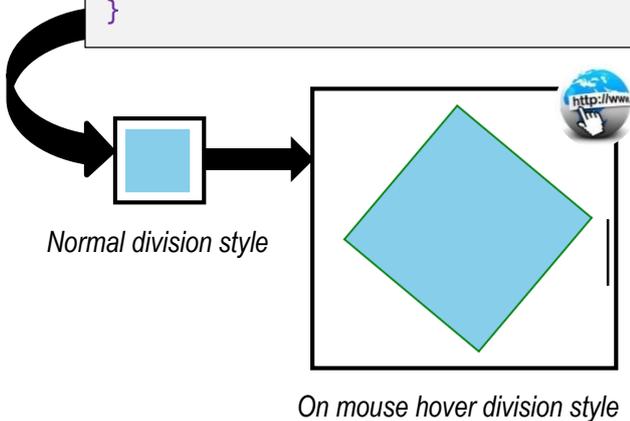
Transitions in CSS3 are an example of this advancement in thinking; in CSS3, HTML elements can have multiple interactive styles (that 'transition' from one style to another) without the need to use Flash, or even JavaScript.

A transition is applied to a HTML element using the CSS `transition` property and specifying the property (in this case width) to transition and set the duration of the effect (in this case two seconds). An example of this property is illustrated right.



The transition property can also be used to transition more than one CSS property. This is achieved by specifying multiple values (both the targeted CSS property and the duration of the transition) preceded and separated by a comma; for example, `transition:width 2s, height 3s`; An example of using multiple transitions is demonstrated below:

```
#mydiv{ /*Transition*/
  height:50px;
  width:50px;
  background-color:skyblue;
  -webkit-transition: width 2s, height 2s, border 2s, transform 2s;
  transition:width 2s, height 2s, border 2s, transform 2s;
}
#mydiv:hover{ /*On Hover Style*/
  width:200px;
  height:200px;
  border:2px solid green;
  -ms-transform:rotate(40deg);
  -webkit-transform:rotate(40deg);
  transform:rotate(40deg);
}
```



HTML Tips!

The transition property does not have to be used solely with the 'hover' pseudo-class; it could (for example) be used with the 'active' pseudo-class (on mouseclick). However, it must be said that mouse hover is the most likely pseudo-class to be used with this property.



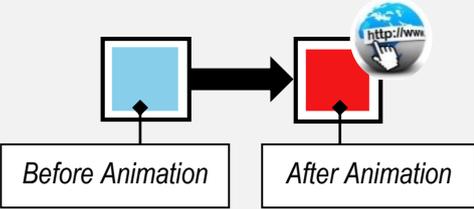
Activity 8.3

Create a web page that includes a single division; use a 'transition' to animate the division when the mouse cursor is hovered over the division.

Animations

Similar to 'transitions', animations is a CSS property that aims to remove the need for Flash and, to some degree, even JavaScript. To create an animation in CSS3, the `@keyframes` rule is used (CSS syntax that is used to generate the animation). The 'keyframe' is used to contain a CSS style that the selected animation will gradually change to. The 'keyframe' must be given a unique name, as well as a transition start and end property/value. The `animation` property itself must, as a minimum, contain the name of the targeted keyframe and the duration of the animation, in seconds. A simple example (including the compatible web kits) of how to use the `animation` property is demonstrated below:

```
#mydiv{ /*Animation*/
  height:50px;
  width:50px;
  background-color:skyblue;
  -webkit-animation:coolweb 3s; /*Chrome, Safari, Opera*/
  animation:coolweb 3s;
}
@-webkit-keyframes coolweb{ /*Animation Style for Chrome, Safari, Opera */
  from{background-color:skyblue;}
  to{background-color:red;}
}
@keyframes coolweb{ /*Animation Style*/
  from{background-color:skyblue;}
  to{background-color:red;}
}
```

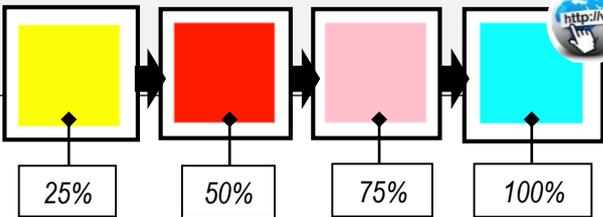


The example above is a very simplistic example of CSS3 animations; however, it is possible to make them more advanced by using multiple CSS properties in the animation. Another useful feature is the ability to be able to specify when a change should happen; this is achieved by using a percentage between 0 and 100. An example of this is demonstrated in the syntax below:

```
#mydiv{ /*Animation*/
  height:50px;
  width:50px;
  background-color:skyblue;
  -webkit-animation:coolweb 8s; /*Chrome, Safari, Opera*/
  animation:coolweb 8s;
}
@-webkit-keyframes coolweb{ /*Animation Style for Chrome, Safari, Opera */
  25%{background-color:yellow;}
  50%{background-color:red;}
  75%{background-color:pink;}
  100%{background-color:cyan;}
}
@keyframes coolweb{ /*Animation Style*/
  25%{background-color:yellow;}
  50%{background-color:red;}
  75%{background-color:pink;}
  100%{background-color:cyan;}
}
```

Activity 8.4

Create a web page that includes a single division; use an animation to manipulate its appearance. Research how to make the division move and apply this too.



Multiple columns

There are some new properties available in CSS3 that allow web developers to easily create column effects, similar to that of a newspaper. Using columns is a popular layout technique; this is probably because it is aesthetically pleasing. The CSS property used to create columns is the `column-count` property; this is used by setting its value to the desired number of columns. There is also a `column-gap` and a `column-rule` property; these are used to specify the gap (in pixels) between columns and to set the width, style and colour of the rule that appears between the columns. All of these properties are demonstrated in the syntax below:

```
#mydiv{ /*Columns*/
  -webkit-column-count:3; /*Chrome, Safari, Opera*/
  -moz-column-count:3; /*Firefox*/
  column-count:3;
  -webkit-column-gap:30px; /*Chrome, Safari, Opera*/
  -moz-column-gap:30px; /*Firefox*/
  column-gap:30px;
  -webkit-column-rule:5px solid skyblue; /*Chrome, Safari, Opera*/
  -moz-column-rule:5px solid skyblue; /*Firefox*/
  column-rule:5px solid skyblue;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugiat nulla facilisi. Nam liber tempor

cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lecto legere me lius quod ii legunt saepius

Box Resizing

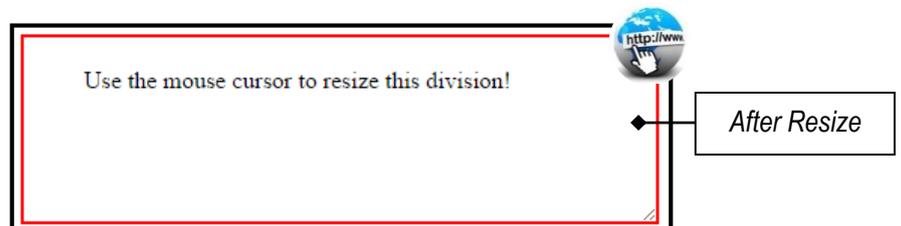
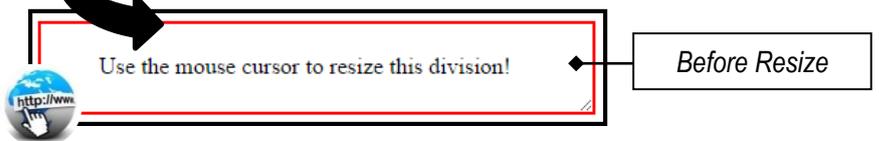
The final CSS3 property examined in this chapter is the `resize` property. This property specifies whether a user is able to resize a HTML element or not. The value of this property can be set to 'both' (to resize both the width and height of an element), 'horizontal' (to resize the width only) or 'vertical' (to resize the height only). The syntax of how to use this property is demonstrated below:

Activity 8.5

Create a web page that includes two divisions; the first should contain a lot of text and be split into four columns. The rule between the columns should be styled too. The second division should also include text, but should be made so that it can be resized by the user.



```
#mydiv{ /*Resize*/
  border:2px solid red;
  padding:20px 40px;
  width:300px;
  resize:both;
  overflow:auto;
}
```



8.3 Theory: The New HTML5 Standard

What is so different about HTML5?

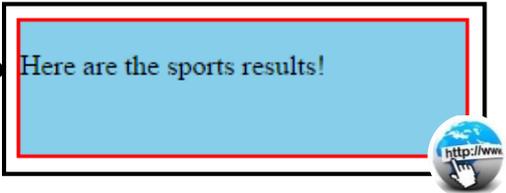
All of the web pages demonstrated in this text are HTML5 pages; they have all been declared using the appropriate 'doc type' (`<!DOCTYPE html>`) and all have followed the HTML5 standard. For example, all containing tags are closed and written in lower case. However, some of the more interesting elements belonging to the HTML5 standard have not been touched on, elements that really stand it apart from all other previous versions. Therefore, from this point forwards, this chapter will provide a quick 'crash course' into the magical world of HTML5.

Before looking at what is new in HTML5, it is best to identify what existing tags (from the previous version of HTML) have been removed. Some of the once popular tags that are now deprecated (no longer part of the HTML standard) include `<applet>`, `<big>`, `<center>`, ``, `<frame>`, `<frameset>` and `<tt>`. Although these elements are no longer part of the standard, their 'purpose' has not been removed; for example, the font of an element is now manipulated using CSS and frame-like pages can still be created using the `<iframe>` tag.

One of the most interesting concepts about HTML5 is that it has returned to a 'true' markup language; in other words, developers can now define their own 'semantic' HTML elements. Semantic means 'to give meaning to'. Consider if a developer created a division on a web page that would hold sports results: the `<div>` tag is a HTML element, but it informs nothing about the content within the division. To a search engine the `<div>` tag offers no meaning. Yet, if this tag was replaced with, for example, `<sportsresults>`, this tag would instantly be meaningful to both people and machines alike. An example of how self-defined semantic tags can be used in HTML5 is demonstrated below:

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML5 is cool!</title>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="my_style_sheet.css">
</head>
<body>
  <sportsresults>
    <p>Here are the sports results!</p>
  </sportsresults>
</body>
</html>
```

```
sportsresults{
  display:block;
  border:2px solid red;
  height:75px;
  width:250px;
  background-color:skyblue;
}
```



Here are the sports results!



HTML Tips!

When adding new semantic tags in HTML5, use CSS to set their `display` property to 'block'; this will ensure that the elements behave as though they were a normal HTML division.

Compatible HTML5

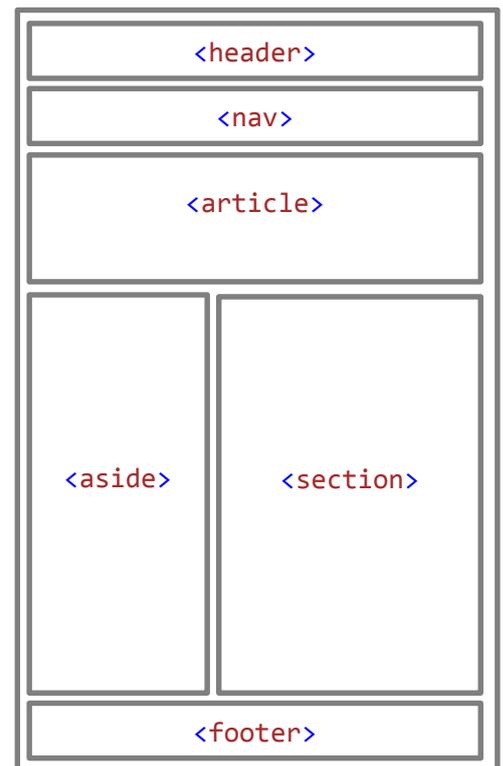
Unfortunately, there are some known compatibility issues when implementing HTML5; the problem being of course that, although HTML5 is catered for by modern-day browsers, older browser versions (especially Internet Explorer) do not allow for these new semantic elements to be implemented. Thankfully there are some small steps that can be taken to maximise the potential for browser compatibility. One of the most important is adding the 'HTML5 enabling JavaScript', to the head section of a HTML5 document; this is highlighted in the syntax below. Notice that the script is placed inside a comment; this is because modern-day browsers will ignore the script, whereas earlier versions of Internet Explorer will read and interpret it. In theory, this script, written by Sjoerd Visscher, will help to make earlier versions of Internet Explorer more compatible with HTML5.

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML5 is cool!</title>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="my_style_sheet.css">
  <!--[if lt IE 9]>
    <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
  <![endif]-->
</head>
<body>
  <sportsresults>
    <p>Here are the sports results!</p>
  </sportsresults>
</body>
</html>
```

HTML5 Semantic Tags

As stated earlier, some tags are non-semantic in their meanings; for example, the HTML elements `<div>` and `` offer no indication about their content. Yet, some HTML tags were already semantic; for example, the `<table>` and `` tags clearly define their content, and thus can be classed as semantic elements. To make HTML more 'semantic', HTML5 has included a set of new tags that help to define meaning.

As most web developers often label their divisions with meaningful names anyway, for example `<div id="nav">` or `<div id="footer">`, it made sense to create elements that could describe their content more closely; therefore, HTML5 introduced some new elements, including `<header>`, `<footer>`, `<aside>`, `<nav>`, `<section>`, `<article>`, `<main>` and `<summary>`. How some of these new elements can be applied in practice is clearly shown in the wireframe that is illustrated to the right.



8.4 Practical: HTML5 Elements

HTML5 Layout

Rather than using ID element selectors, although perfectly valid, in HTML5 it is preferable to use the new semantic elements, as demonstrated in the wireframe on the previous page. Written below is the HTML5 and CSS syntax that can be used to create a simple web page layout, using the new semantic elements:

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML5 is cool!</title>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="my_style_sheet.css">
  <!--[if lt IE 9]>
    <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
  <![endif]-->
</head>
<body>
  <header>
    <h1>Semantic Tags Rock!</h1>
  </header>
  <nav>
    <a href="#">The Footer Tag</a> |
    <a href="#">The Header Tag</a> |
    <a href="#">The Article Tag</a> |
    <a href="#">The Nav Tag</a>
  </nav>
  <article>
    <h1>Independent Article</h1>
    <p>I am a self-contained article!</p>
  </article>
  <aside>
    <h1>Side Bar</h1>
    <p>I am a side bar with relating information to the page!</p>
  </aside>
  <section>
    <h1>Section</h1>
    <p>I am general content with a header!</p>
  </section>
  <footer>
    <p>I am the footer text!</p>
  </footer>
</body>
</html>
```

Activity 8.6



Create a semantic web page that includes at least a header, footer, sidebar, navigation bar and main placeholder. The web page should be styled so that it is attractive; perhaps research colour schemes and complementing colours. The page can be on any topic, but the content should be lengthy enough to fill the sections created.

```

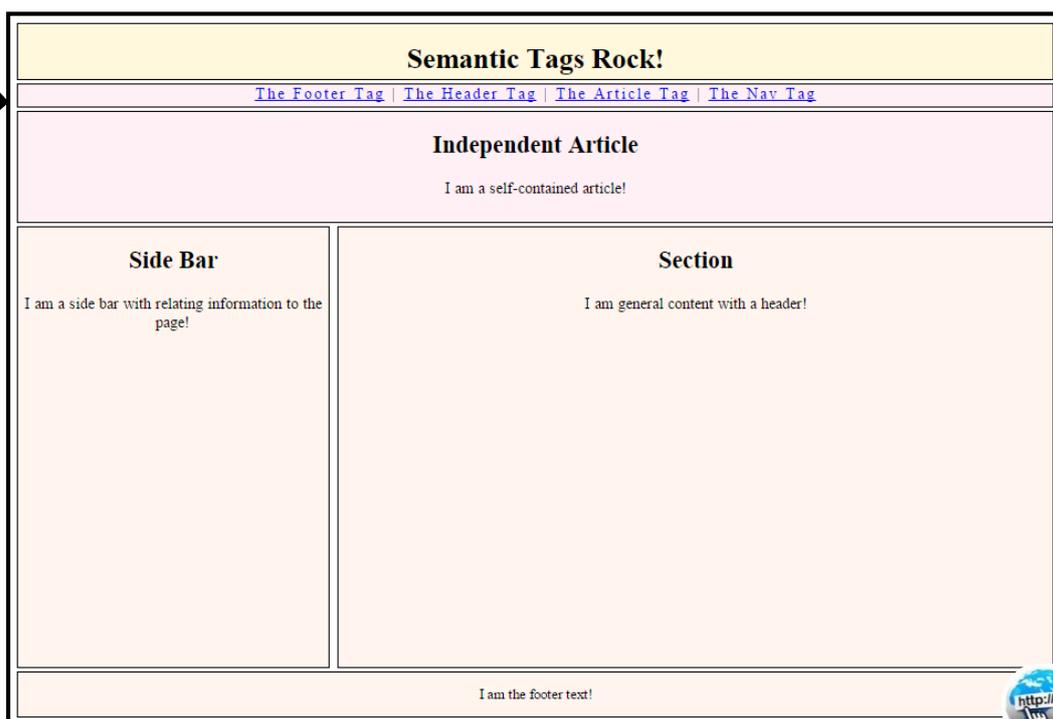
header, nav, article, aside, section, footer{
    display:block;
}
header{
    width:100%;
    height:50px;
    border:1px solid black;
    background-color:Cornsilk;
    text-align:center;
    font-size:0.8em;
}
nav{
    margin-top:3px;
    width:100%;
    height:20px;
    border:1px solid black;
    background-color:LavenderBlush;
    text-align:center;
    font-size:0.9em;
    letter-spacing:2px;
}
article{
    margin-top:3px;
    width:100%;
    height:100px;
    border:1px solid black;
    background-color:LavenderBlush;
    text-align:center;
    font-size:0.9em;
}

```

```

aside{
    margin-top:3px;
    width:30%;
    height:400px;
    border:1px solid black;
    background-color:SeaShell;
    text-align:center;
    font-size:0.9em;
    float:left;
}
section{
    margin-top:3px;
    margin-left:31%;
    width:69%;
    height:400px;
    border:1px solid black;
    background-color:SeaShell;
    text-align:center;
    font-size:0.9em;
}
footer{
    margin-top:3px;
    width:100%;
    height:40px;
    border:1px solid black;
    background-color:SeaShell;
    text-align:center;
    font-size:0.8em;
}

```



To clarify the meaning (and the correct use) of the new semantic tags, refer to the table below:

<code><article></article></code>	The article element should be used to contain independent information that is not necessarily directly linked to the website itself. The content should be completely understandable and readable on its own, without referring to any other part of the website or web page.
<code><aside></aside></code>	The aside element is used to contain information that is to be placed into a side bar. The information in a side bar should technically be related to the subject on the main web page.
<code><footer></footer></code>	The footer element is used to indicate the 'footer' of a web page. A footer element should typically be used to contain information about the web page author, copyright details, contact information and/or hyperlinks to other useful web pages on the website; for example, the 'terms of use' or the full copyright policy.
<code><header></header></code>	The header element is used to specify the 'header' of a web page and, therefore, should generally contain introductory content. The header element can also be used multiple times in a single web page.
<code><main></main></code>	The main element is used to contain the 'main' information that would normally be displayed on a web page. The content stored in this section should be unique to the web page and not repeated. There should not be more than one main element on a single web page.
<code><nav></nav></code>	The nav element is used to define a section of a web page that will contain navigation hyperlinks. Although this element is intended to contain navigation hyperlinks, this does not mean that all hyperlinks have to be placed inside this element.
<code><section></section></code>	The section element is used to define a 'section' in a web page; in other words, a grouping of content, normally with a heading. This element should be used to split a web page into 'sections'; for example, introduction, content, contact, etc.

HTML5 Graphical Elements

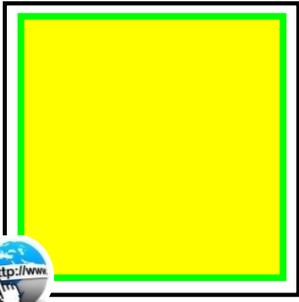
In HTML5 it is now possible to draw scalable shapes to a web page, which otherwise would be achieved using images. The benefit of drawing an image to a web page is that it uses less physical space than a typical image, as well as providing a developer with more control over the graphics appearance; this is especially useful when scaling a drawn graphic for different screen resolutions.

To create graphics in HTML5 the `<svg>` element is used. SVG is short for Scalable Vector Graphic and is a W3C recommendation. For reference purposes, a vector graphic is an image that is created from a mathematical expression/calculation and thus is scalable without the pixels becoming distorted. The SVG element also provides several methods that enable boxes, circles and text to be generated. An example of how to use the `<svg>` element, specifically to create a rectangle, is demonstrated below:

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML5 Graphics</title>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="my_style_sheet.css">
</head>
<body>
  <svg width="200" height="200" >
    <rect width="200" height="200">
      Your browser does not support SVG Graphics!
    </rect>
  </svg>
</body>
```

This text is displayed if the browser does not support the SVG element

```
rect{
  fill:rgb(255,255,0); /*fill colour*/
  stroke-width:10; /*stroke size*/
  stroke:rgb(0,255,0); /*stroke colour*/
}
```



HTML Tips!
A 'stroke' is an outside border that is applied to a graphic; the stroke width is set in pixels, but the unit (px) is not specified.

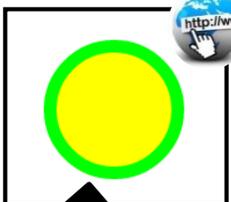
The SVG element can also be used to create a circle graphic; this is shown below:

```
...
<body>
  <svg width="200" height="200" >
    <circle cx="100" cy="100" r="50">
      Your browser does not support SVG Graphics!
    </circle>
  </svg>
</body>
...
```

Radius

X coordinate

Y coordinate



```
circle{
  fill:rgb(255,255,0); /*fill colour*/
  stroke-width:10; /*stroke size*/
  stroke:rgb(0,255,0); /*stroke colour*/
}
```

Activity 8.7
Create a web page that demonstrates the SVG element, by creating a circle and a rectangle on the web page. Style both shapes differently.

8.5 Theory: HTML5 Multimedia Elements

With HTML5 it is possible to play media elements (both audio and video) without the need for plugins, JavaScript or any other third-party platforms. In the past playing multimedia elements was problematic; this was because different browsers supported different file types and there was no consistency. Hopefully, with HTML5, this is 'a product of the past'; in theory, all HTML5-compatible browsers will support video playback of the *.MP4, *.WebM and *.Ogg file types. HTML5-compatible browsers will also support the following audio types: *.MP3, *.WAV and *.Ogg.

HTML5 Video Elements

To play videos in HTML5 the `<video>` element is used (all modern-day browsers support this element). The `width` and `height` attributes are used to set the size of the element, and the `controls` attribute determines whether the playback controls are visible or not (note that the controls differ in appearance between browsers). A `<source>` element is necessary to set the location of the video that is to be played; it is set using the `src` attribute. The video playback type is a mandatory property too and this is defined using the `type` attribute. An example of how to use the HTML5 video element is displayed below:



HTML Tips!

Note that it is considered good practice to provide two or more video playback types (and sources); this will help to ensure that the video is browser compatible. The browser will play the first recognised video type.

An appropriate error message should also be written between the opening and closing video tags; this text will only be displayed if the browser (earlier versions) does not support the video element.

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML5 Multimedia</title>
  <meta charset="UTF-8">
</head>
<body>
  <video width="200" height="200" controls>
    <source src="my_movie.mp4" type="video/mp4">
    <source src="my_movie.ogg" type="video/ogg">
    Your browser does not support video playback!
  </video>
</body>
</html>
```



Video playback can be set to start automatically in HTML5, by adding the `autoplay` attribute to the video element. For example, `<video width="200" height="200" autoplay>` would begin to play the video as soon as the web page is loaded into the browser. In HTML5, it is also considered good practice to specify the width and the height of the video; this will enable the browser to reserve space on the web page while the video is loading. If these attributes are not set, the web page will resize itself once the video has loaded.

HTML5 Audio Elements

Similar to the video element, HTML5 also provides a new 'audio' element that enables audio files to be played in the browser too; again, without the need for additional plugins. The element used to achieve this is the `<audio>` element. Similar to the `<video>` element, the `<source>` element is necessary to set the location path of the audio file. An example of how to use the `<audio>` element is shown below:

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML5 Multimedia</title>
  <meta charset="UTF-8">
</head>
<body>
  <audio controls>
    <source src="my_song.mp3" type="audio/mpeg">
    <source src="my_song.ogg" type="video/ogg">
    Your browser does not support audio playback!
  </audio>
</body>
</html>
```

Enables the audio controls, for example, play, pause and volume



Chapter Summary



- ✓ When writing CSS, the aim should be to write syntax that is backwards-compatible, syntax that will be visually consistent on all browsers and versions. In CSS3, this should be achieved by using, where available, the appropriate web kit for each browser.
- ✓ The `box-shadow` property adds a shadow effect to the outline of a HTML element.
- ✓ The `border-image` property creates a border from a specified image.
- ✓ CSS3 enables gradients to be created and applied to the background of HTML elements. For example, `background: linear-gradient(#F9F3FF, #CC99FF);` would create a linear gradient from top to bottom. This can be further amended to change the direction of the gradient, to change the gradient type and to add additional colours to the gradient.
- ✓ The `transform` property can be used to manipulate the appearance of HTML elements, including 'scaling' them (making them larger or smaller) and rotating them.
- ✓ The `word-wrap` property can be used to wrap text inside a HTML element, so that the text does not spill outside the container.
- ✓ The `transition` property can be used to 'transition' HTML elements between two different CSS styles. For example, a 'mouse hover' that applies a different style to the HTML element than the original style (when the HTML is in its default state).
- ✓ The `animation` property, combined with key frames, can be used to create animations in the browser, without the need for using third-party platforms, such as Flash.
- ✓ HTML5 now includes semantic tags, tags that describe their content more closely. Some of the new semantic tags include `<header>`, `<footer>`, `<aside>`, `<nav>`, `<section>`, `<article>`, `<main>` and `<summary>`. In HTML5 it is also possible for developers to define their own semantic tags.
- ✓ The `<audio>` and `<video>` elements can be used to embed multimedia (audio and video files) into a web page.