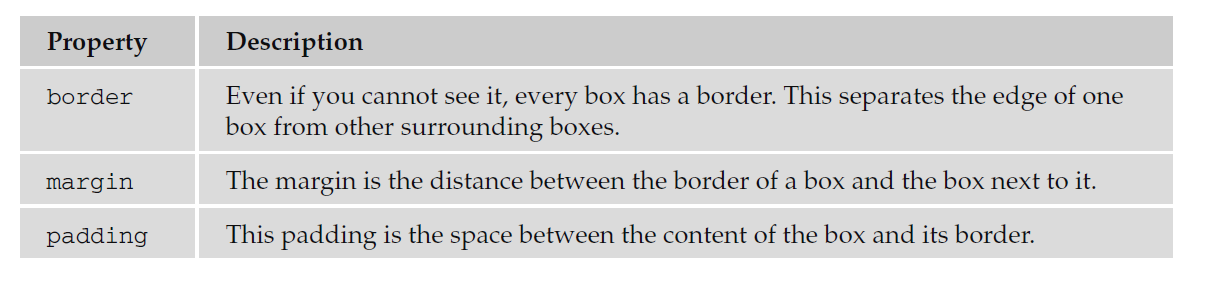
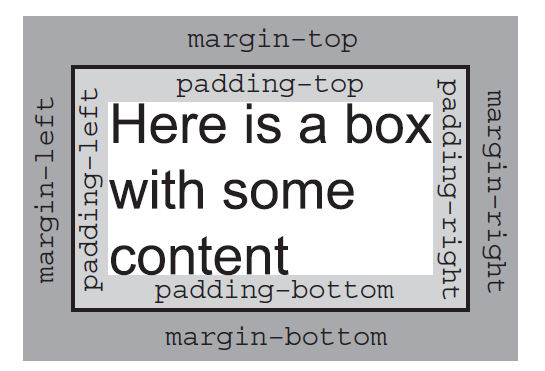
**Introducing the Box Model**

The *box model* is a very important concept in CSS because it determines how elements are positioned within the browser window. It gets its name because CSS treats every element as if it were in a *box*.

As you can see in the table that follows, every box has three properties you must be aware of.



You can get a better idea of these properties in Figure below, which shows the various parts of the box (the black line is the border).

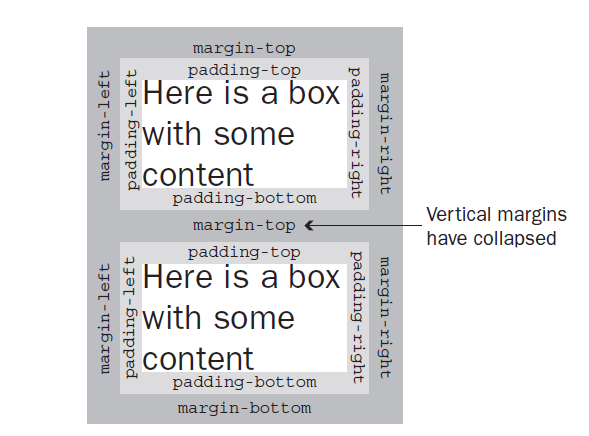


You can use CSS to individually control the border, margin, and padding on each side of a box; you can specify a different width, line - style and color for each side of the boxes’ border.

The padding and margin properties are especially important in creating what designers refer to as ***white space*;** this is the space between the various parts of the page. For example, if you have a box with a blackborder and the box contain black text, you would not want the text to touch the border because itwould make the text harder to read. Giving the box padding helps separate the text from the line aroundthe edge.

Meanwhile, suppose you have two boxes next to each other, both with borders. If there is no margin between them the boxes will run into each other, and ***the line where the boxes meet could look thicker than the other lines.***

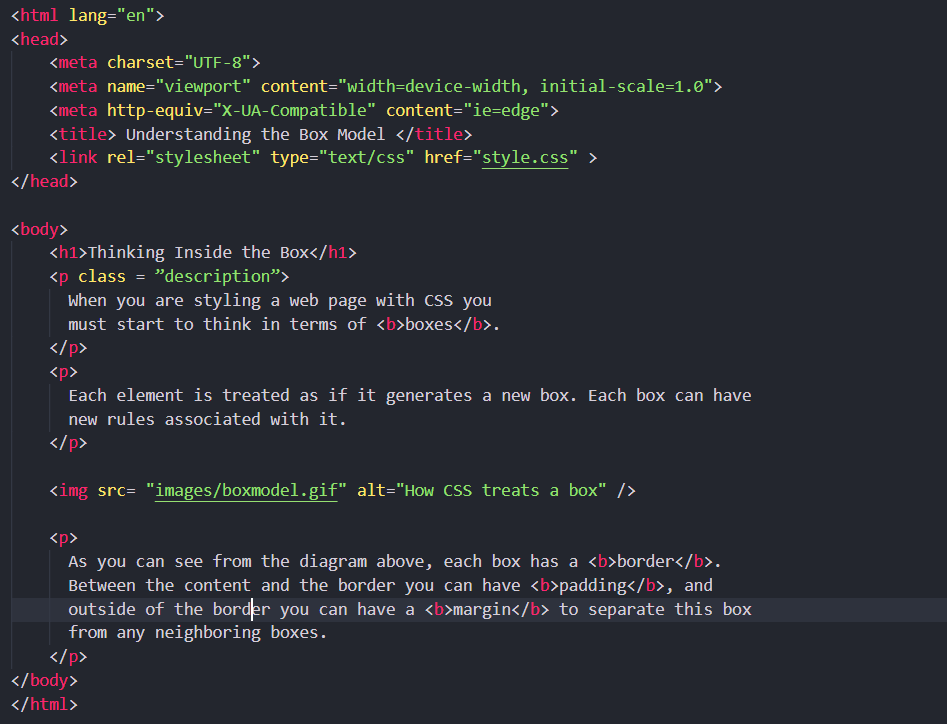
There is, however, **an interesting issue with margins:** when a bottom margin of one element meets the top margin of another, only the larger of the two will show. (If they are the same size, then the margin will be equivalent to the size of just one of the margins.) Figure below shows the vertical margins of two adjacent boxes collapsing. **(Note that this only applies to vertical margins; the same is not true for left and right margins)**.



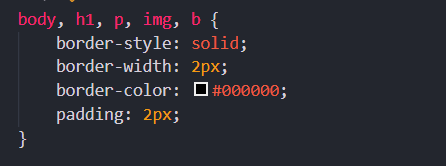
To really understand how the box model works with elements, take a look at the example in the next section.

**An Example Illustrating the Box Model:**

To illustrate the box model, we will add a border to each of the elements in a web page. The <body> element creates one box that contains the whole page, and inside that box each heading, paragraph, image, or link creates another box. First, here is the HTML for the page:



Using just one CSS rule you can see how each element involved with the body of the document — <body>, <h2>, <p>, <img>, and <b> — gets treated as if it were in a separate box. You can do this by adding a CSS rule that will add a border around each of these elements, you will learn more about these properties shortly.



Each box can be presented differently, for example we can give the <h1> and <bold> elements a gray background to help distinguish them from other elements.

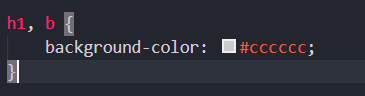
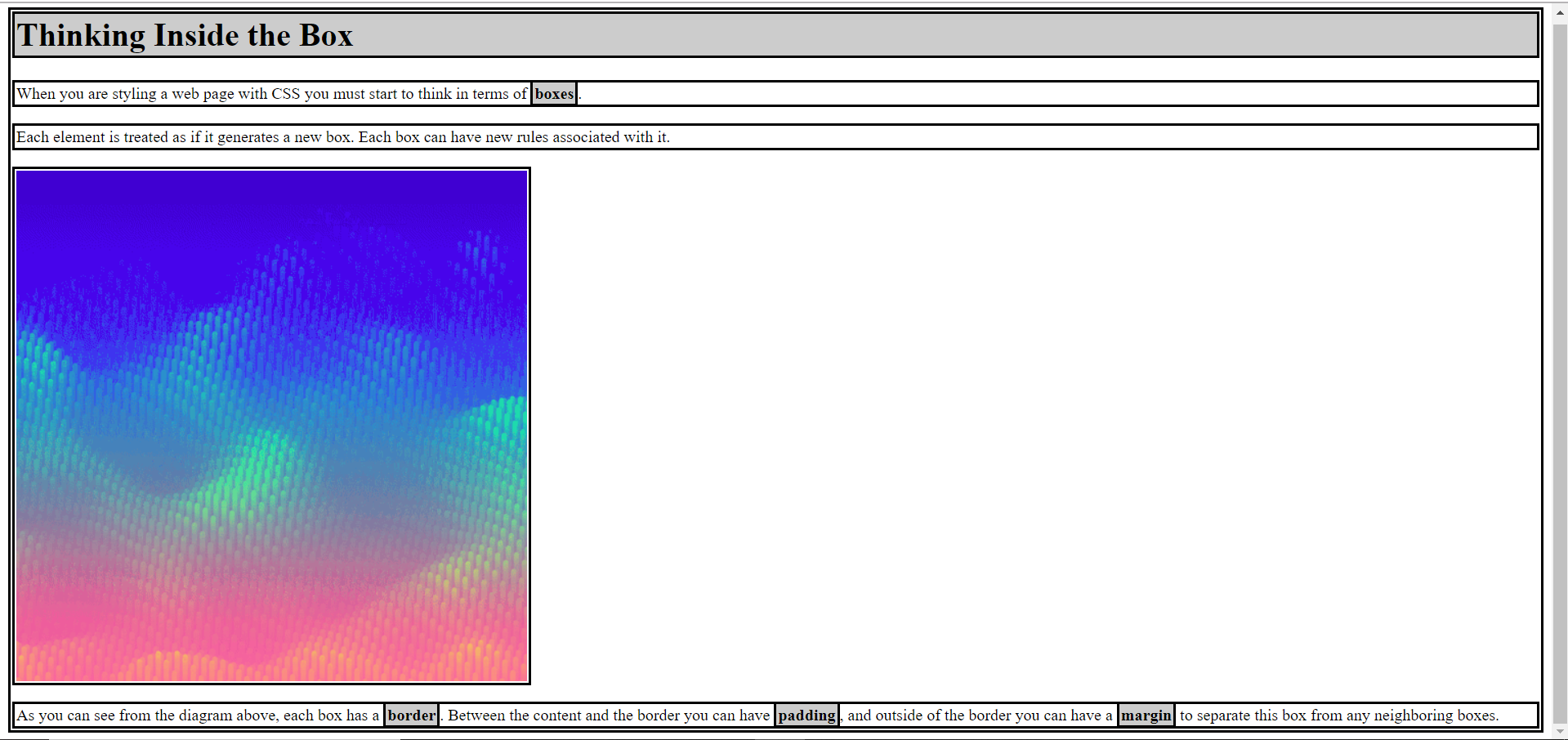


Figure below, shows you what this page looks like in a browser. While it is not too attractive, the lines show you the borders of the boxes (and demonstrate how boxes are created for each element).



You may remember from Chapter 3(html) that there is a difference between *block level elements* and *inline elements*; the difference becomes quite important when working with CSS because it determines howeach box is displayed.

* The <h1> and <p> elements are examples of block - level elements. Each block level element starts on a new line, and the box around a block level element takes up the full width of the browser (or the full width of the element it sits inside).
* The <b> element is an example of an inline element. Its box sits in the middle of the paragraph and it does not take up the width of a whole line (it *flows* within its containing element).

The <img> element may look like it is a block - level element because it starts on its own line, but ***it is actually an inline element***. You can tell this because the border around it takes up only the width of the image; if it were a block - level element, the border would reach across the full width of the browser. The image is on its own line only because the elements on either side of it *are* block - level elements (and therefore, the surrounding elements appear on their own lines).

In Strict XHTML, this image element should be placed inside a block - level element, as you are only supposed to have block - level elements as children of the < body > element. While it does not matter in Transitional XHTML, you could simply fix this issue by putting the element inside a <div> element (which you might remember is a block - level grouping element).

Now that you know how each element is treated as if it were in its own box, let’ s take a look at the properties that control the borders, margins, and padding for each box.

**The Border Properties:**

The border properties allow you to specify how the border of the box representing an element should look. There are three properties of a border you can change:

* **border-color** to indicate the color a border should be
* **border-style** to indicate whether a border should be a solid, dashed, or double line, or one of the other possible values
* **border-width** to indicate the width a border should be

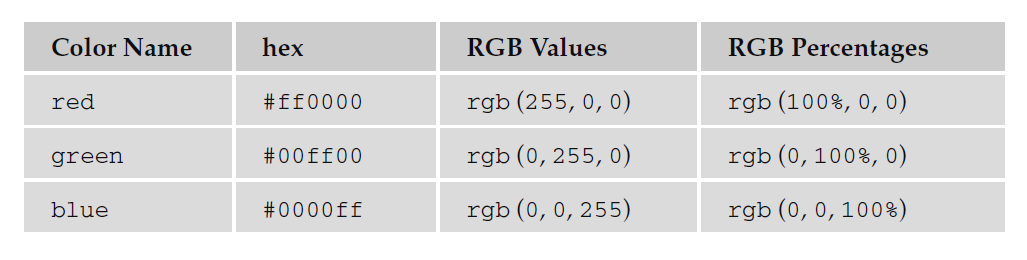
**The border-color Property:**

The border - color property allows you to change the color of the border surrounding a box.

For example:

p {border-color: #ff0000;}

The value can be a color name or a hex code for the color. It can also be expressed as values for red, green, and blue; between 0 and 255; or percentages of red, green, and blue. See the table that follows for examples.



You can individually change the color of the bottom, left, top, and right sides of a box’ s border using the following properties:

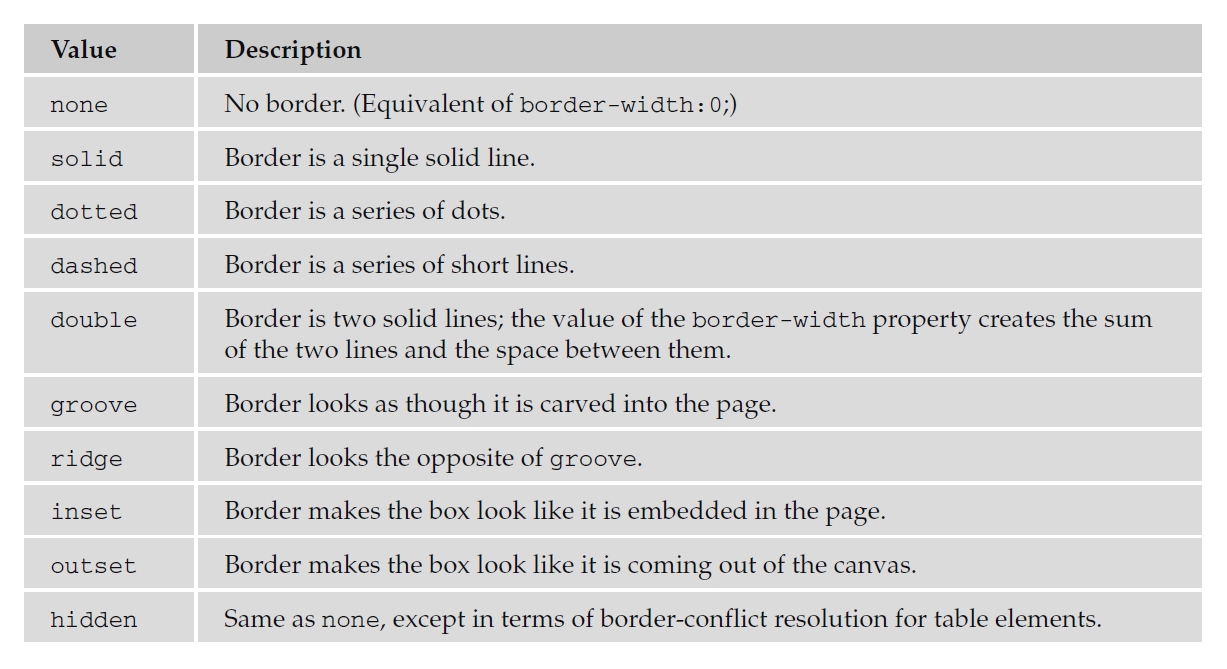
* border- bottom-color
* border- right-color
* border- top-color
* border- left-color

**The border-style Property:**

The border-style property allows you to specify the line style of the border:

p {border-style: solid;}

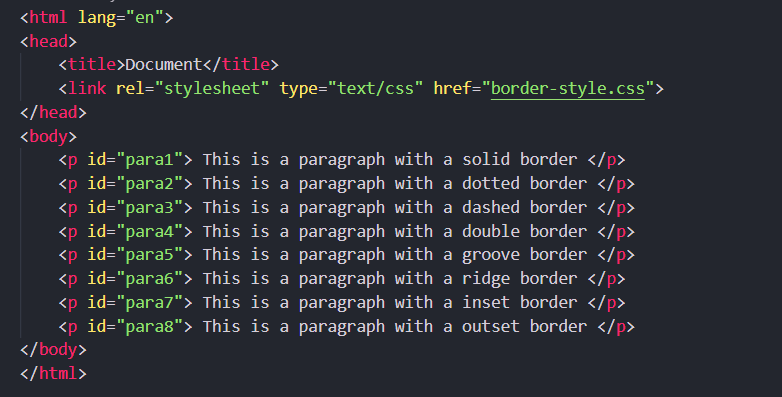
The default value for this property is none, so no border would be shown automatically. The table that follows shows the possible values.

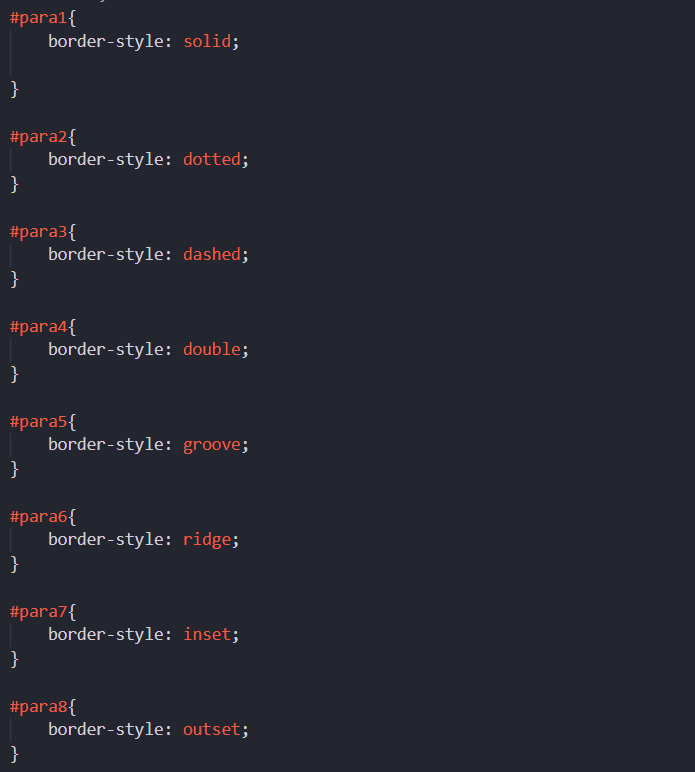


You can individually change the style of the bottom, left, top, and right borders of a box using the following properties:

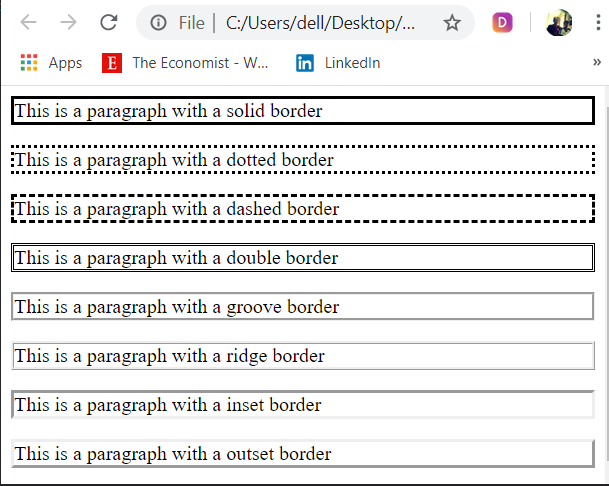
* border-bottom-style
* border-right-style
* border-top-style
* border-left-style

Example: (html and css file)





Output:



**The border-width Property:**

The border - width property allows you to set the width of your borders; usually the width is specified in pixels.

p {

border-style: solid;

border-width: 4px;

}

The value of the border - width property cannot be given as a percentage, although you could use any absolute unit or relative unit, or one of the following values:

* thin
* medium
* thick

The actual width of the thin, medium, and thick values is not specified in the CSS recommendation in terms of pixels; so, the actual width that corresponds to these keywords is dependent on the browser.

You can individually change the width of the bottom, top, left, and right borders of a box using the following properties:

* border-bottom-width
* border-right-width
* border-top-width
* border-left-width

**Expressing Border Properties Using Shorthand:**

The border property allows you to specify color, style, and width of lines in one property:

p {border: 4px solid red;}

If you use this shorthand, the values should not have anything (other than a space) between them. You can also specify the color, style, and width of lines individually for each side of the box in the same way using these properties:

* border-bottom
* border-top
* border-left
* border-right

**The padding Property:**

The padding property allows you to specify how much space should appear between the content of an element and its border:

td {padding:5px;}

The value of this property is most often specified in pixels, although it can use any of the units of length we met earlier, a percentage, or the word inherit.

The padding of an element will not inherit by default, so if the <body> element has a padding property with a value of 50 pixels, this will not automatically apply to all other elements inside it. If the value inherit is applied to any elements, only then will they have the same padding as their parent elements.

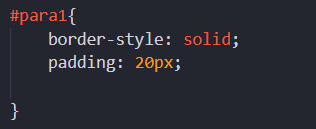
If a percentage is used, the percentage is of the containing box, and if the value of 10 percent is specified, there would be 5 percent of each side of the box as padding.

You can specify different amounts of padding inside each side of a box using the following properties:

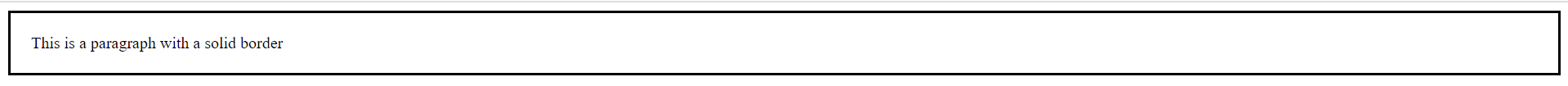
* padding-bottom
* padding-top
* padding-left
* padding-right

The padding attribute is especially helpful in creating white space between the content of an element and any border it has.

Example: (css file)



Output:



**The margin Property:**

The margin property controls the gap between boxes, and its value is either a length, a percentage, or inherit, each of which has exactly the same meaning as it did for the padding property you just saw.

p {margin:20px;}

As with the padding property, the values of the margin property are not inherited by child elements unless you use the value inherit.

Also, remember that when one box sits on top of another box, only the larger of the two margins will show (or if both are equal the size of one margin).

You can also set different values for the margin on each side of the box using the following properties:

* margin-bottom
* margin-top
* margin-left
* margin-right

If you look at the following example (see example below), you can see three paragraphs, which look as if they are spaced equally. However, they have taller margins on the top than the bottom, and therefore where two boxes meet, the bottom margin is ignored: the margins are collapsed (note that this only happens to the vertical margins, not the left and right margins).

The example also shows how to set the left and right margins on the side of inline elements — where you see the highlighted words. Again, this is not the most attractive example, but it illustrates how both block and inline boxes use margins.

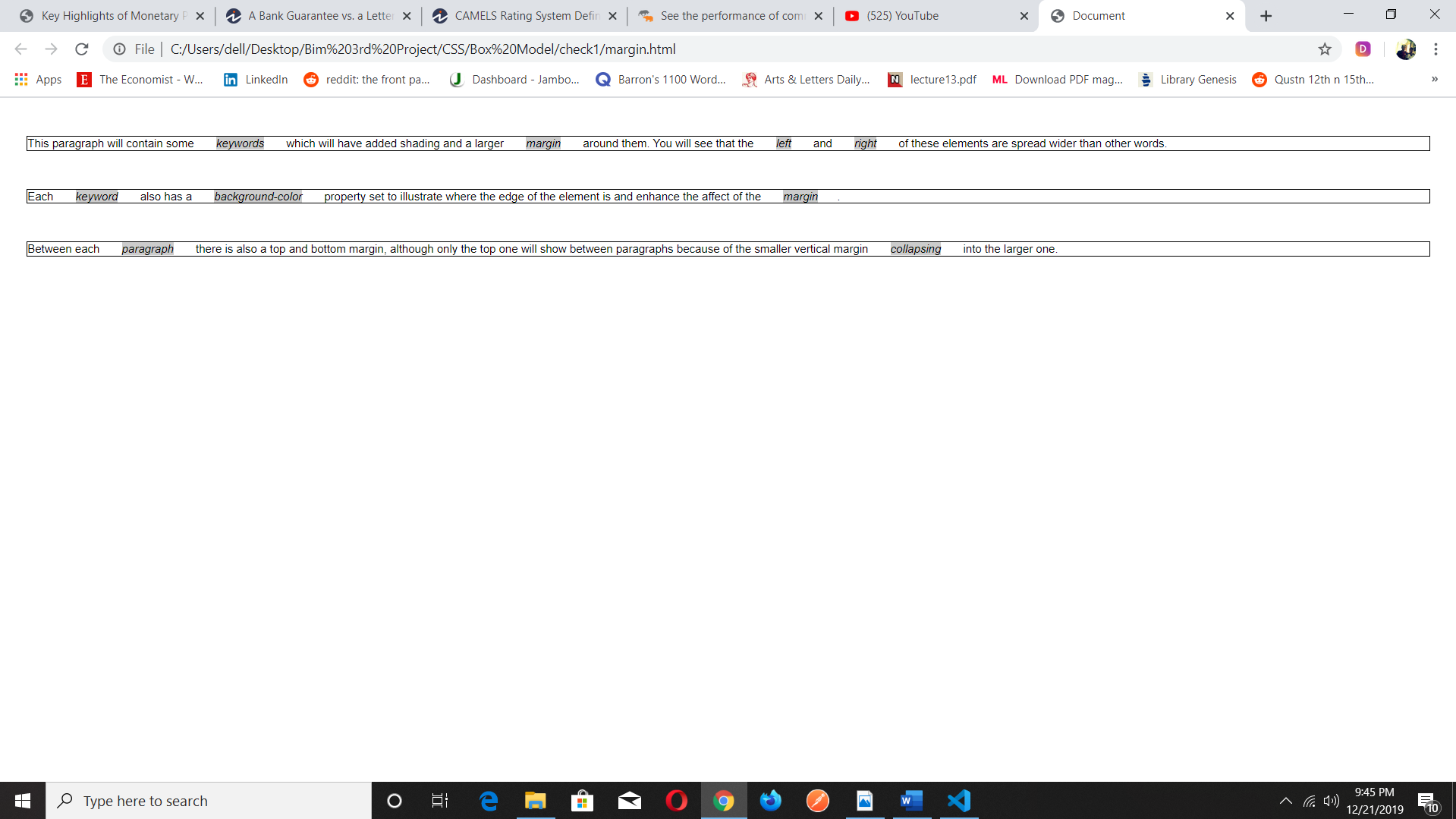
The words in the paragraphs that are emphasized using the <em> element have margin - left and margin - right properties set. Because these <em> elements also have a background color set, you can really see below how the margins to the left and the right separate the words from the surrounding words.

Example: (html and css)



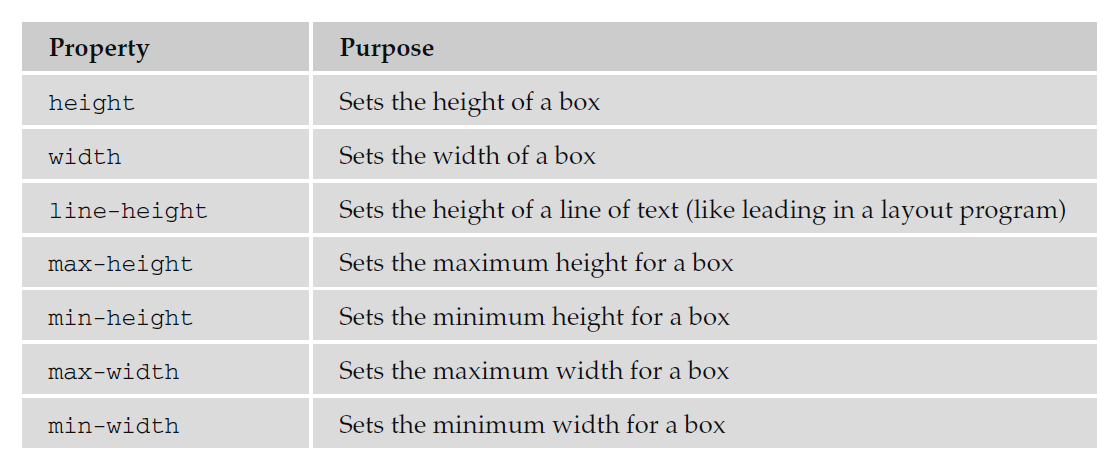


Output:



**Dimensions of a Box:**

Now that you’ve seen the border that surrounds every box, the padding that can appear inside the border of each box, and the margin that can go outside the border, it is time to look at the properties that allow you to control the dimensions of boxes.

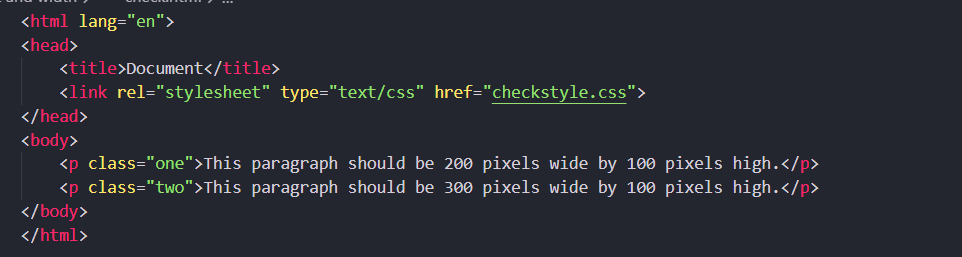


**The height and width Properties:**

The height and width properties allow you to set the height and width for boxes. They can take values of a length, a percentage, or the keyword auto (the default value being auto, which means the box is just large enough to house its contents).

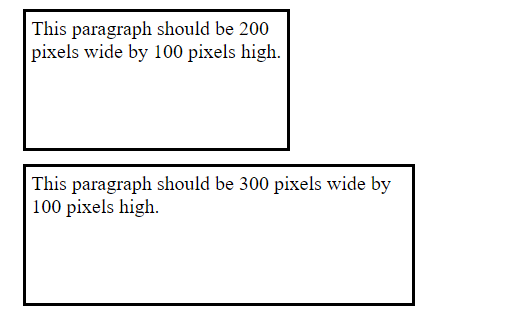
Here you can see the CSS rules for two paragraph elements, the first with a class attribute whose value is one and the second whose class attribute has a value of two:

Example:





Output:



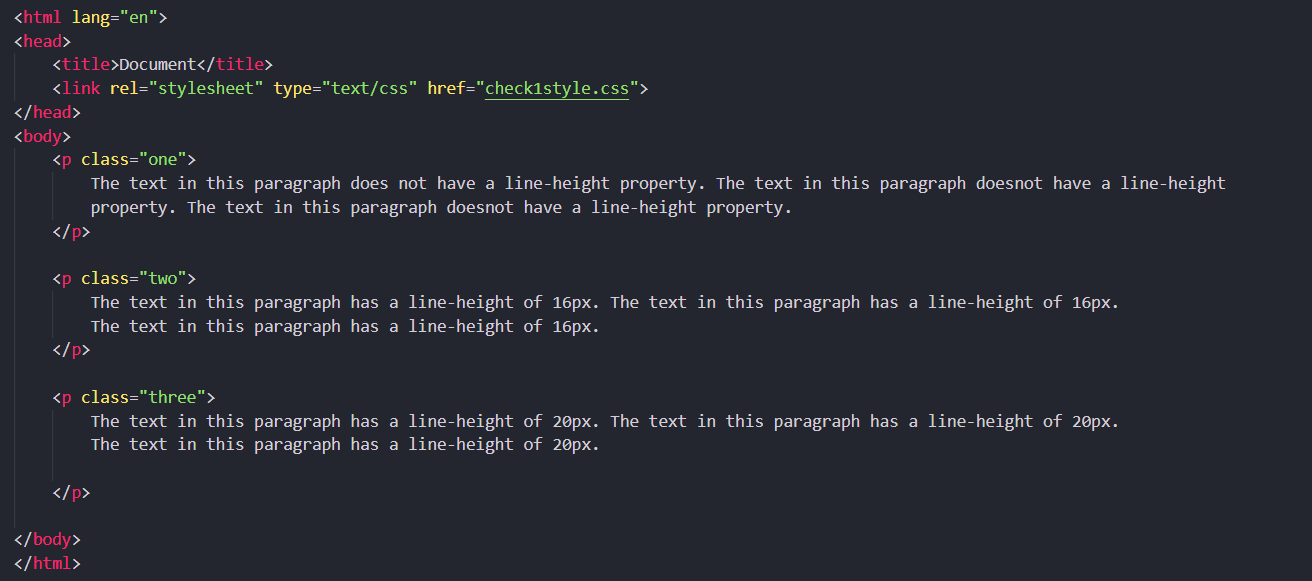
The most common unit of measurement for boxes is pixels, although percentages and em’s are often used in layouts that stretch and contract to fit the size of the browser window.

**The line-height Property:**

The line-height property is one of the most important properties when laying out text. It allows you to increase the space between lines of text ***(known to print designers as leading).***

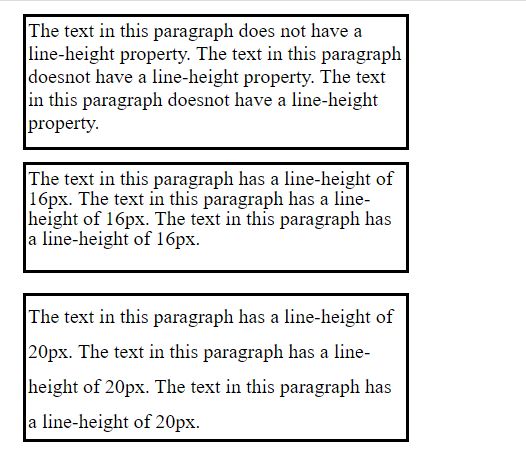
The value of the line-height property can be a length or a percentage. It is a good idea to specify this property in the same measurement in which you specify the size of your text.

Example:





Output:



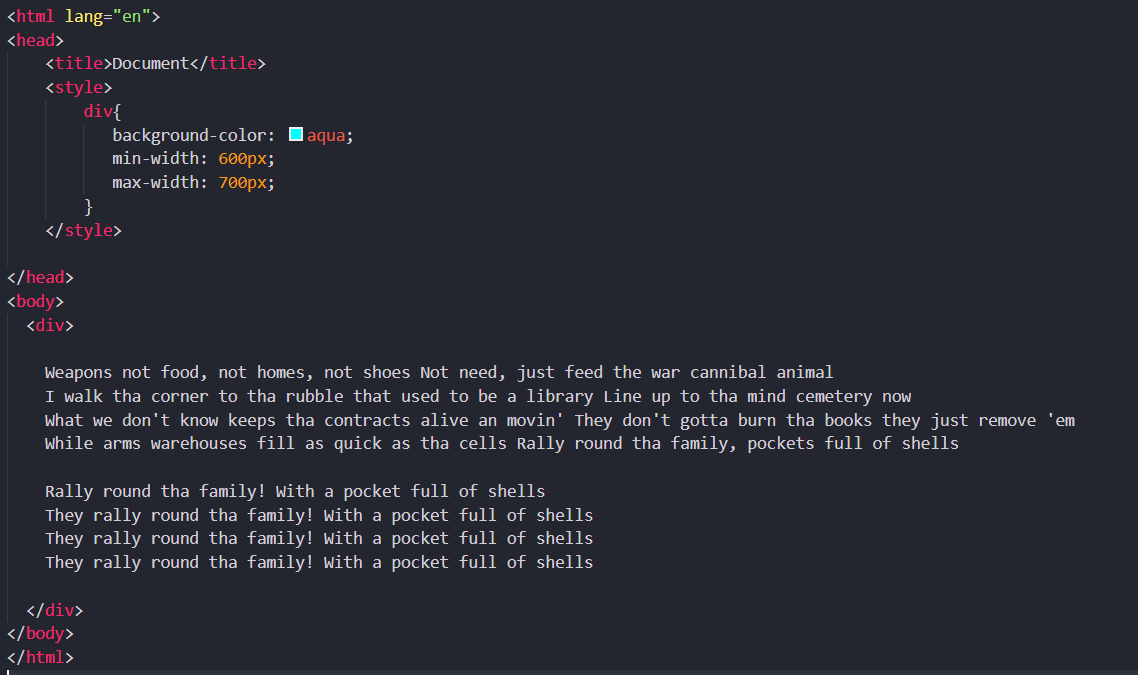
As you can see in above output, the first paragraph does not have a line - height attribute, whereas the second and third paragraphs correspond to the preceding rules. Adding some extra height between each line of text can often make it more readable, especially in longer articles. In long paragraphs I try to use leading of around 1.5 times the height of the font. This property can also be helpful when you need to add spacing around single lines of text.

**The max-width and min-width Properties:**

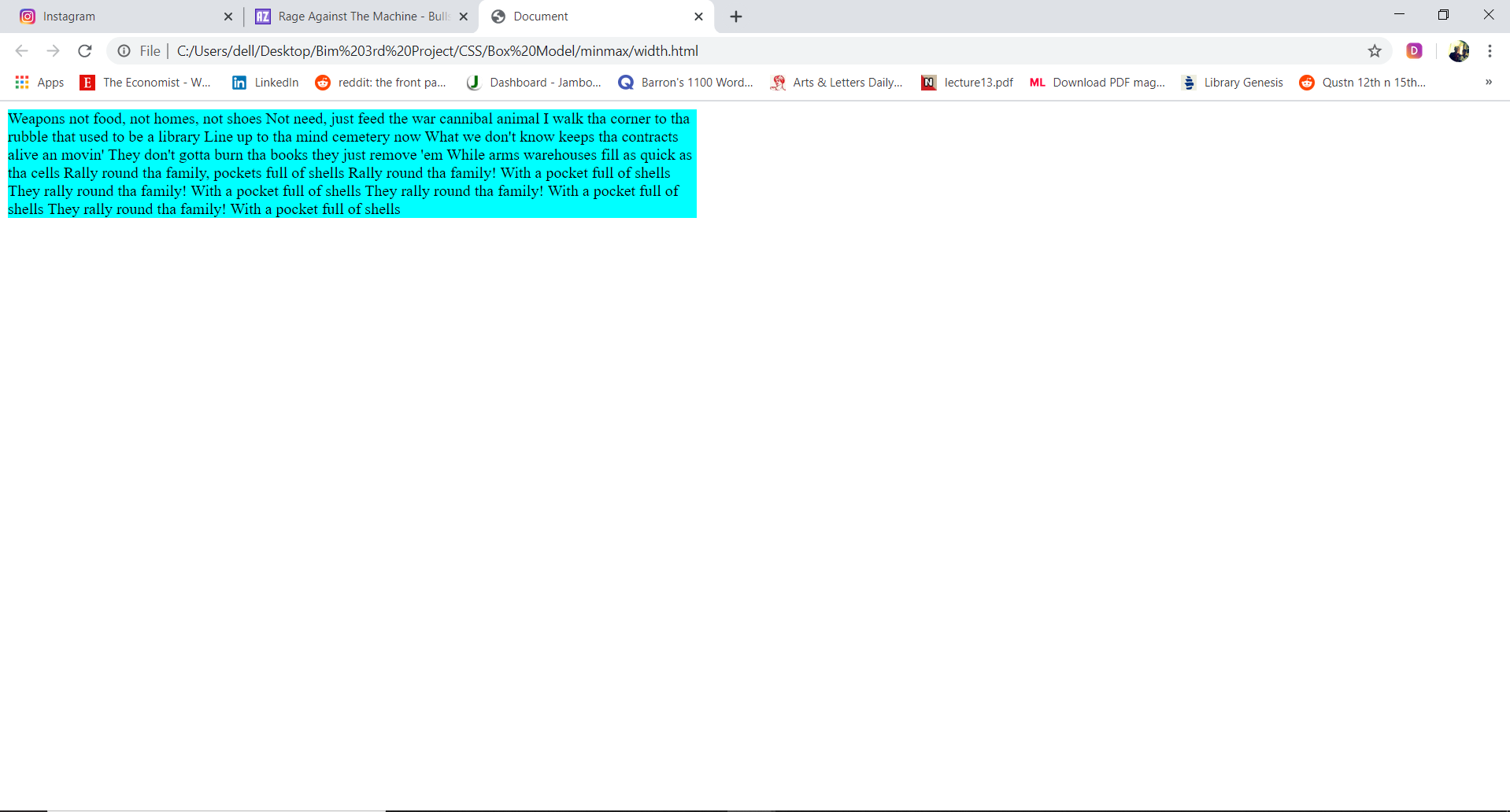
The max-width and min-width properties allow you to specify a maximum and a minimum width for a box. This should be particularly useful if you want to create parts of pages that stretch and shrink to fit the size of user’s screens. **The max - width property will stop a box from being so wide that it is hard to read (lines that are too long are harder to read on screens), and min - width will help prevent boxes from being so narrow that they are unreadable.** It is important to note, however, that IE7 and FF2 were the first of the major browsers to support these properties.

The value of these properties can be a number, a length, or a percentage, and negative values are not allowed. For example, take a look at the following rule, which specifies that a <div> element may not be less than 600 pixels wide and no wider than 700 pixels:

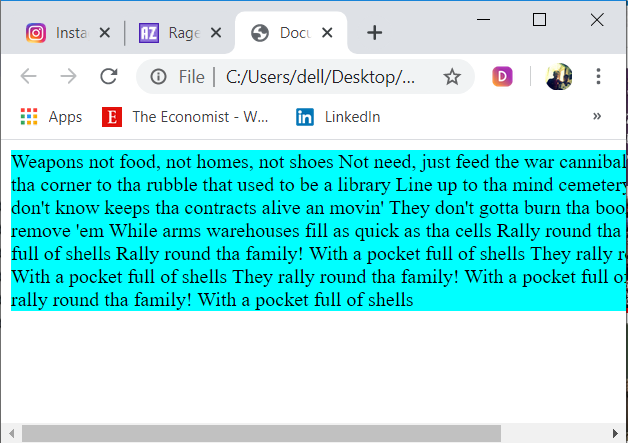
Example:



Output:



Output: When min width is below 600px



**The min-height and max-height Properties:**

The min-height and max-height properties correspond with the min-width and max-width properties, but specify a minimum height and maximum height for the box. Again, it is important to note that IE7 and Firefox 2 were the first major browsers to support these properties.

The value of these properties can be a number, a length, or a percentage, and negative values are not allowed. Take a look at the following example:

div {

min-height:50px;

max-height:200px;

padding:5px;

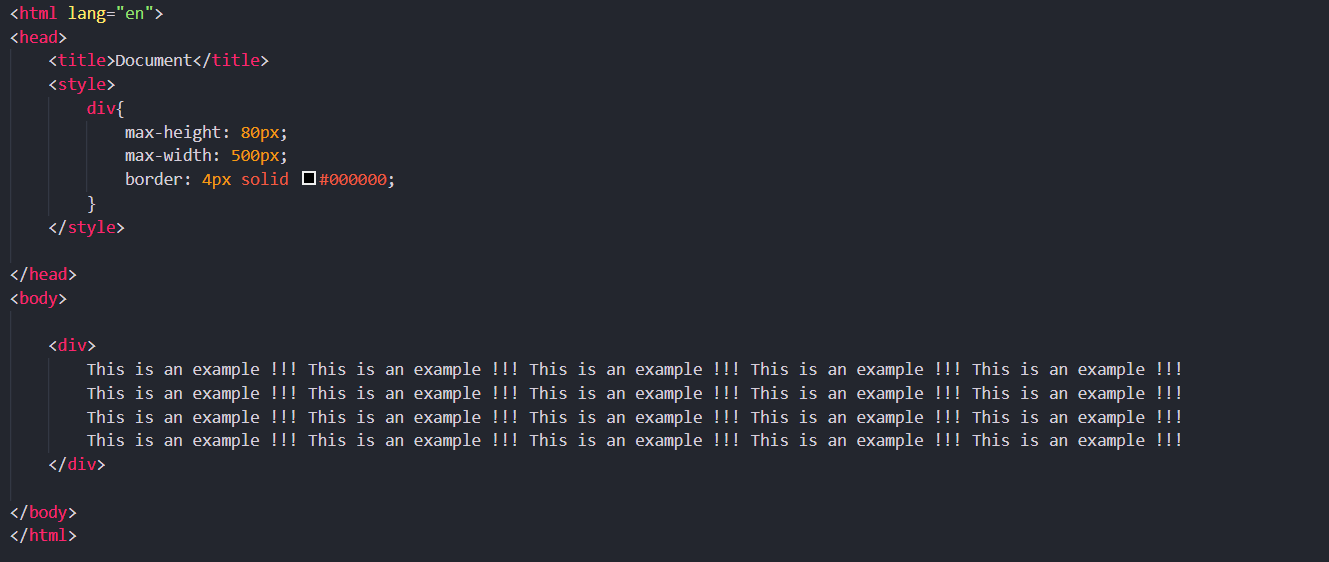
border:1px solid #000000;

}

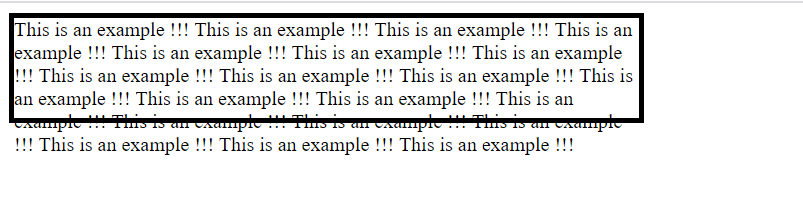
**The overflow Property:**

When you want to control the size of a box, the content you want to fit in the box might require more space than you have allowed for it. This not only happens with the min-height and max-height or min-width and max-width properties, but also for a number of other reasons.

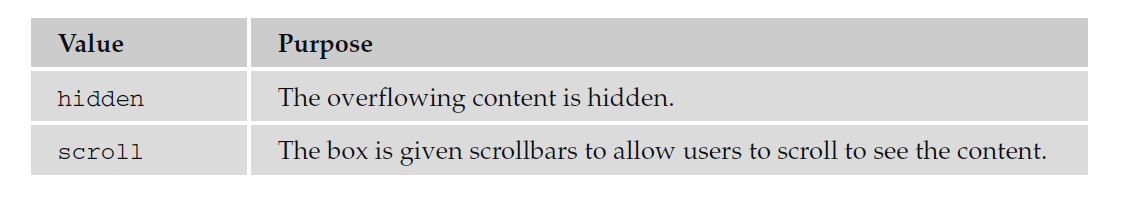
Example:



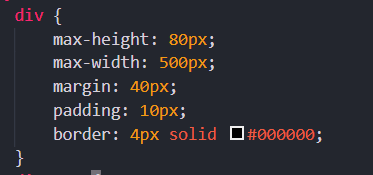
Output:

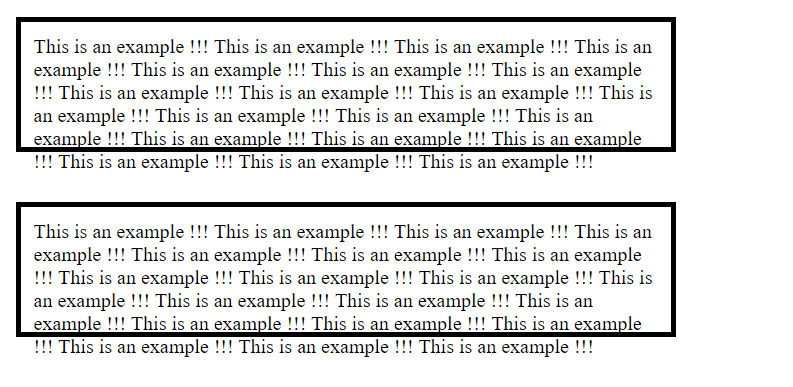


The overflow property was designed to deal with these situations and can take one of the values shown in the table that follows.



Example:



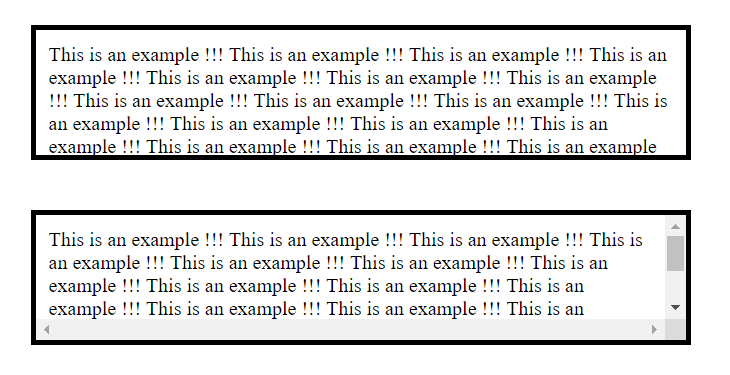


Take a look at the following example, where the width of two <div> elements has been controlled by the max-height and max-width properties so that the content of the <div> elements does not fit in the box. For the first element, I have set the overflow property to have a value of hidden and the second to have a value of scroll:

Example(final):

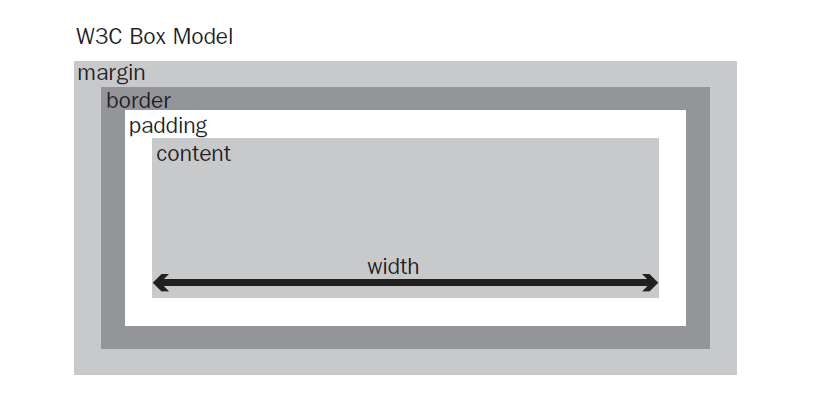


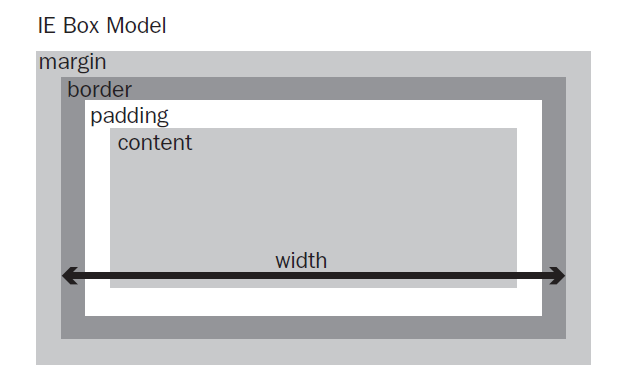
Now take a look at Figure below. You can see the effect of these two properties — in the first box the text is simply cut off when it runs out of space, and in the second box a scrollbar is created allowing users to scroll to the appropriate content.



**Internet Explorer Box Model Bug:**

When you come to build sites using CSS you will probably come across a well - documented bug in the Internet Explorer Box Model that affects the width of boxes.IE treated the width of a box as though it included the width of any border it had been given and the width of the padding in the size of the box. However, the CSS specification says that the width of a box should only be the width of the content (not including the border or padding). You can see this in figure below:





IE6 resolved this problem because it was able to run in two “modes”:

* Standards - compliant mode follows the CSS specification. You can make sure IE6 or above is running in standards - compliant mode by including one of the !DOCTYPE declarations that you met in earlier Chapter.
* Quirks mode retains the bug, and IE6 or above will run in Quirks mode when there is no !DOCTYPE declaration.