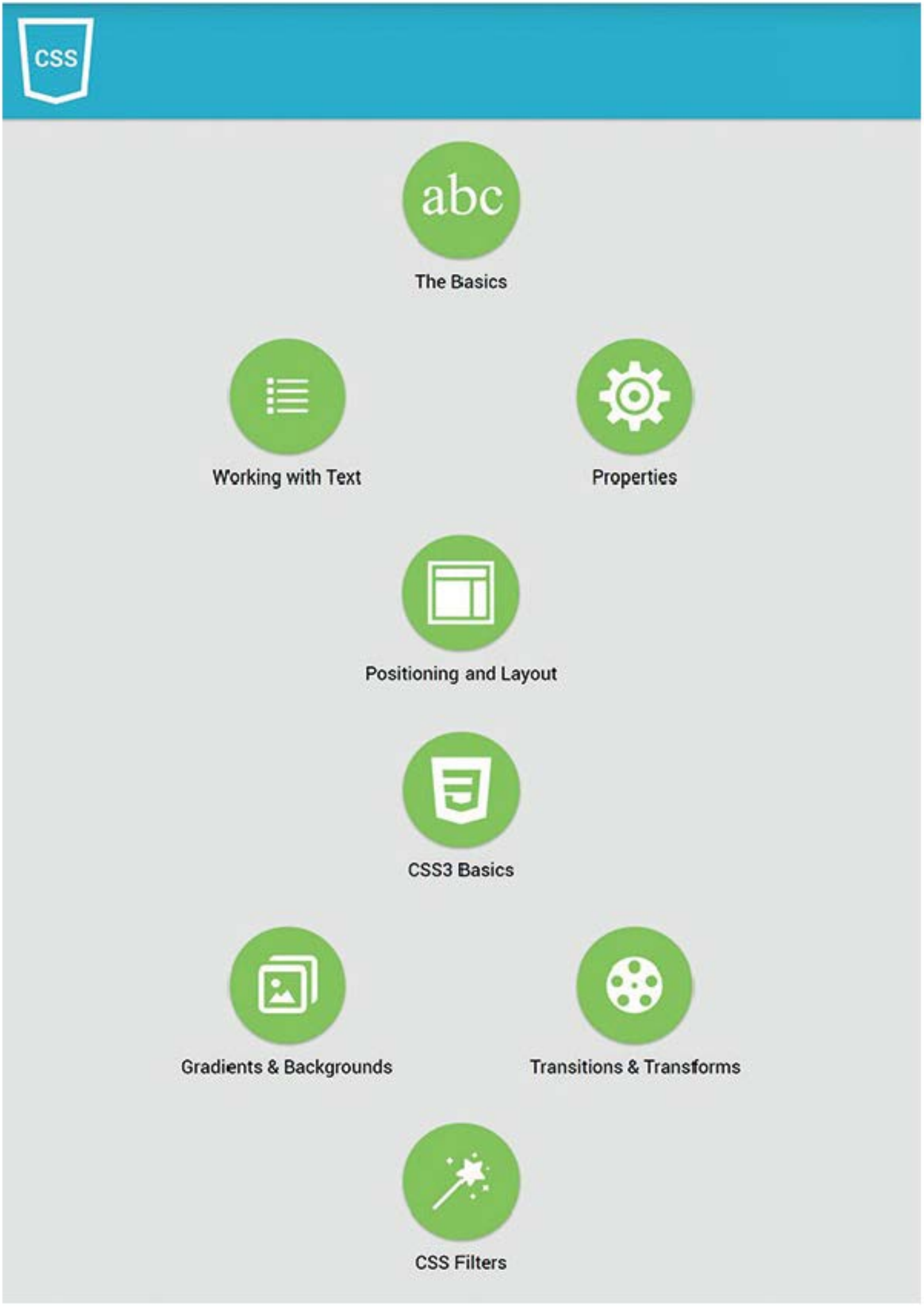

CSS
abc
The Basics
Working with Text
Properties
Positioning and Layout
CSS3 Basics
Gradients & Backgrounds
Transitions & Transforms
CSS Filters

TABLE DES MATIÈRES

The Basics

Welcome to CSS!

CSS stands for **C**ascading **S**tyle **S**heets.

- **Cascading** refers to the way CSS applies one style on top of another.
- **Style Sheets** control the look and feel of web documents.

CSS and **HTML** work hand in hand:
- HTML sorts out the page structure.
- CSS defines how HTML elements are displayed.

> To understand CSS, you should already have a basic knowledge of HTML.
> If you want to study HTML, check out the free SoloLearn **Learn HTML** app.

849 COMMENTS

Why Use CSS?

CSS allows you to apply specific styles to specific HTML elements.

The main benefit of CSS is that it allows you to separate **style** from **content**.

Using just HTML, all the styles and formatting are in the same place, which becomes rather difficult to maintain as the page grows.

> All formatting can (and **should**) be removed from the HTML document and stored in a separate CSS file.

265 COMMENTS

Q&A

Inline CSS

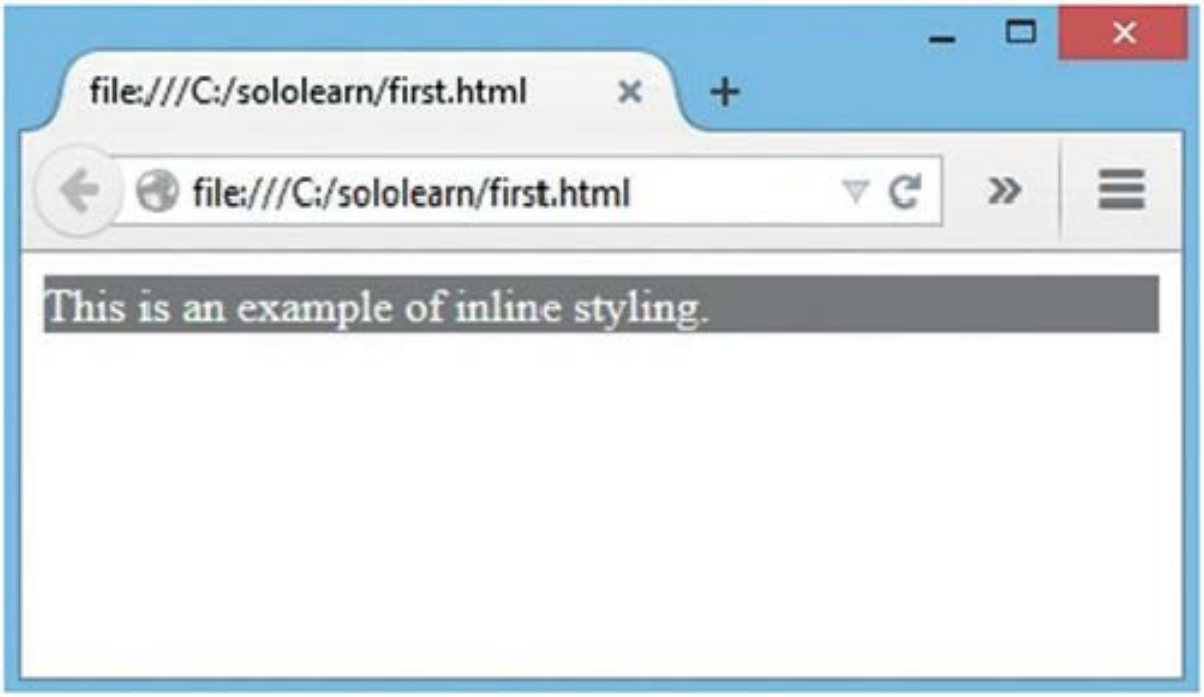
Using an inline style is one of the ways to insert a style sheet. With an inline style, a unique style is applied to a single element.

In order to use an inline style, add the **style attribute** to the **relevant tag**.

The example below shows how to create a paragraph with a gray background and white text:

```
<p style="color:white; background-color:gray;">
  This is an example of inline styling.
</p>
```

Result:



The style attribute can contain any CSS property.

403 COMMENTS

Q&A

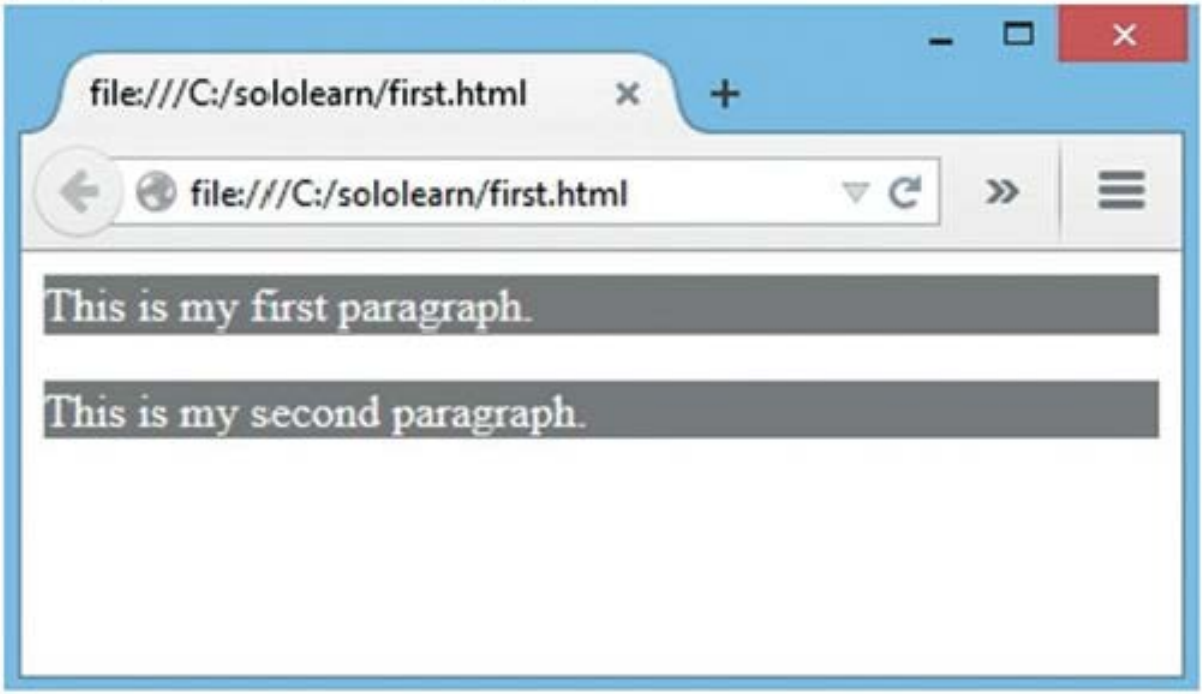
Embedded/Internal CSS

Internal styles are defined within the **<style>** element, inside the **head** section of an HTML page.

For example, the following code styles **all** paragraphs:

```
<html>
  <head>
    <style>
    p {
      color:white;
      background-color:gray;
    }
    </style>
  </head>
  <body>
    <p>This is my first paragraph. </p>
    <p>This is my second paragraph. </p>
  </body>
</html>
```

All paragraphs have a white font and a gray background:



An internal style sheet may be used if one single page has a unique style.

356 COMMENTS

Q&A

External CSS

With this method, all styling rules are contained in a single text file, which is saved with the **.css** extension.

This CSS file is then referenced in the HTML using the **<link>** tag. The <link> element goes inside the head section.

Here is an example:
The HTML:

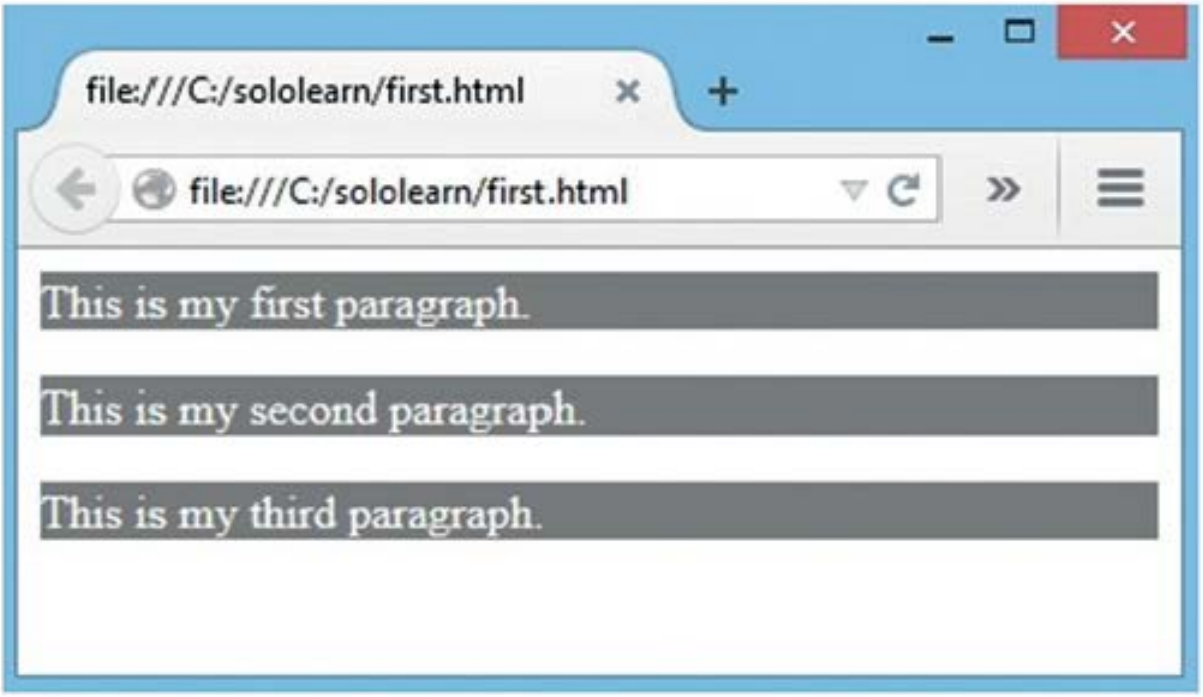
```
<head>
  <link rel="stylesheet" href="example.css">
</head>
<body>
  <p>This is my first paragraph.</p>
  <p>This is my second paragraph. </p>
  <p>This is my third paragraph. </p>
</body>
```

The CSS:

```
p {
  color:white;
  background-color:gray;
}
```

Try It Yourself

Result:



CSS Syntax

CSS is composed of style rules that the browser interprets and then applies to the corresponding elements in your document.
A style rule has three parts: **selector**, **property**, and **value**.

For example, the headline color can be defined as:

```
h1 { color: orange; }
```

Try It Yourself

Where:

Selector Property Value

The selector points to the HTML element you want to style.

The declaration block contains one or more declarations, separated by semicolons.
Each declaration includes a property name and a value, separated by a colon.

316 COMMENTS

Q&A

Type Selectors

The most common and easy to understand selectors are **type selectors**. This selector targets element types on the page.

For example, to target all the paragraphs on the page:

```
p {
  color: red;
  font-size:130%;
}
```

Try It Yourself

A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly braces.

185 COMMENTS

id and class Selectors

id selectors allow you to style an HTML element that has an **id** attribute, regardless of their position in the document tree. Here is an example of an id selector:

The HTML:

```
<div id="intro">
  <p> This paragraph is in the intro section.</p>
</div>
<p> This paragraph is not in the intro section.</p>
```

The CSS:

```
#intro {
  color: white;
  background-color: gray;
}
```

Try It Yourself

To select an element with a specific id, use a hash character, and then follow it with the id of the element.

Result:

Class selectors work in a similar way. The major difference is that IDs can only be applied once per page, while classes can be used as many times on a page as needed.

In the example below, both paragraphs having the class "first" will be affected by the CSS:

The HTML:

```
<div>
  <p class="first">This is a paragraph</p>
  <p> This is the second paragraph. </p>
</div>
<p class="first"> This is not in the intro section</p>
<p> The second paragraph is not in the intro section. </p>
```

The CSS:

```
.first {font-size: 200%;}
```

Try It Yourself

To select elements with a specific class, use a period character, followed by the name of the class.
Do **NOT** start a class or id name with a number!

616 COMMENTS

Descendant Selectors

These selectors are used to select elements that are descendants of another element. When selecting levels, you can select as many levels deep as you need to.

For example, to target only elements in the first paragraph of the "intro" section:

The HTML:

```
<div id="intro">
  <p class="first">This is a <em> paragraph.</em></p>
  <p> This is the second paragraph. </p>
</div>
<p class="first"> This is not in the intro section.</p>
<p> The second paragraph is not in the intro section. </p>
```

The CSS:

```
#intro .first em {
  color: pink;
  background-color:gray;
}
```

Try It Yourself

As a result, only the elements selected will be affected:

The descendant selector matches all elements that are descendants of a specified element.

Comments

Comments are used to explain your code, and may help you when you edit the source code later. Comments are ignored by browsers.

A CSS comment look like this:

```
/* Comment goes here */
```

Example:

```
p {
  color: green;
  /* This is a comment */
  font-size: 150%;
}
```

Try It Yourself

The comment does not appear in the browser:

Comments can also span multiple lines.

250 COMMENTS

Cascade

The final appearance of a web page is a result of different styling rules.

The three main sources of style information that form a cascade are:

- The stylesheet created by the **author of the page**
- The **browser's default styles**
- Styles specified **by the user**

> CSS is an acronym for Cascading Style Sheets.

302 COMMENTS

Inheritance

Inheritance refers to the way properties flow through the page. A child element will usually take on the characteristics of the parent element unless otherwise defined.

For example:

```
<html>
  <head>
    <style>
    body {
      color: green;
      font-family: Arial;
    }
    </style>
  </head>
  <body>
    <p>
    This is a text inside the paragraph.
    </p>
  </body>
</html>
```

Result:

Since the paragraph tag (child element) is inside the body tag (parent element), it takes on any styles assigned to the body tag.

159 COMMENTS

Working with Text

The font-family Property

The font-family property specifies the font for an element.
There are two types of font family names:
- **font family**: a specific font family (like Times New Roman or Arial)
- **generic family**: a group of font families with a similar look (like Serif or Monospace)

Here is an example of different font styles:

Generic family	Font family
Serif	Times New Roman Georgia
Sans - serif	Arial Verdana
Monospace	Courier New Lucida Console

The HTML:

```
<p class="serif">
  This is a paragraph shown in serif font.
</p>
<p class="sansserif">
  This is a paragraph shown in sans-serif font.
</p>
<p class="monospace">
  This is a paragraph shown in monospace font.
</p>
<p class="cursive">
  This is a paragraph shown in cursive font.
</p>
<p class="fantasy">
  This is a paragraph shown in fantasy font.
</p>
```

The CSS:

```
p.serif {
  font-family: "Times New Roman", Times, serif;
}
p.sansserif {
  font-family: Helvetica, Arial, sans-serif;
}
p.monospace {
  font-family: "Courier New", Courier, monospace;
}
p.cursive {
  font-family: Florence, cursive;
}
p.fantasy {
  font-family: Blippo, fantasy;
}
```

Try It Yourself

Result:

Separate each value with a **comma** to indicate that they are alternatives.
If the name of a font family is more than one word, it must be in quotation marks: **"Times New Roman"**.

462 COMMENTS

The font-family Property

The font-family property should hold several font names as a "fallback" system. When specifying a web font in a CSS style, add more than one font name, in order to avoid unexpected behaviors. If the client computer for some reason doesn't have the one you choose, it will try the next one.

It is a good practice to specify a generic font family, to let the browser pick a similar font in the generic family, if no other fonts are available.

```
body {
  font-family: Arial, "Helvetica Neue", Helvetica, sans-serif;
}
```

If the browser does not support the font **Arial**, it tries the next fonts (**Helvetica Neue**, then **Helvetica**). If the browser doesn't have any of them, it will try the generic **sans-serif**.

> Remember to use quotation marks if the font name consists of more than one word.

217 COMMENTS

The font-size Property

The font-size property sets the size of a font. One way to set the size of fonts on the web is to use **keywords.** For example **xx-small, small, medium, large, larger,** etc.

The HTML:

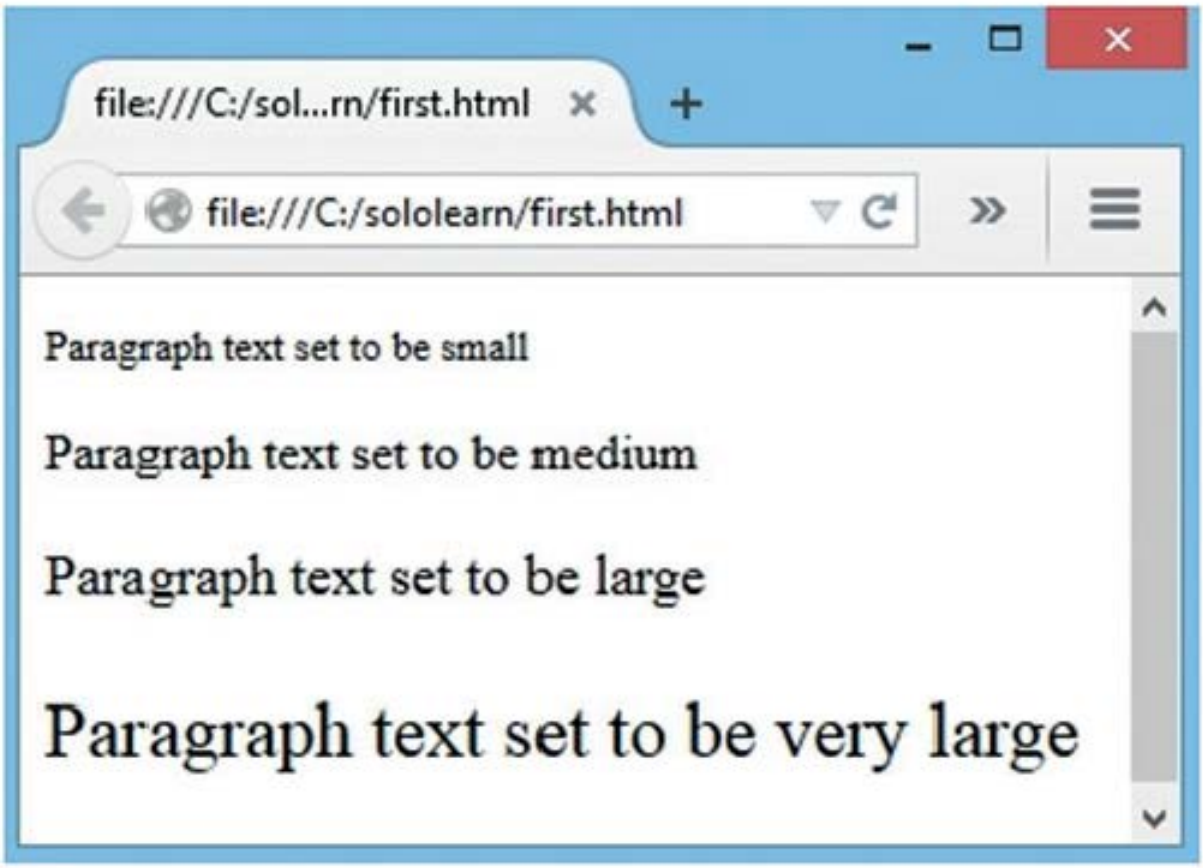
```
<p class="small">
  Paragraph text set to be small
</p>
<p class="medium">
  Paragraph text set to be medium
</p>
<p class="large">
  Paragraph text set to be large
</p>
<p class="xlarge">
  Paragraph text set to be very large
</p>
```

The CSS:

```
p.small {
  font-size: small;
}
p.medium {
  font-size: medium;
}
p.large {
  font-size: large;
}
p.xlarge {
  font-size: x-large;
}
```

Try It Yourself

Result:



Keywords are useful if you do not want the user to be able to increase the size of the font because it will adversely affect your site's appearance.

223 COMMENTS

The font-size Property

You can also use numerical values in **pixels** or **ems** to manipulate font size.
Setting the font size in pixel values (**px**) is a good choice when you need pixel accuracy, and it gives you full control over the text size.
The **em** size unit is another way to set the font size (**em** is a relative size unit). It allows all major browsers to resize the text. If you haven't set the font size anywhere on the page, then it is the browser default size, which is **16px**.

To calculate the em size, just use the following formula: **em = pixels / 16**
For example:

```
h1 {
  font-size: 20px;
}
```

Try It Yourself

```
h1 {
  font-size: 1.25em;
}
```

Try It Yourself

Both of the examples will produce the same result in the browser, because **20/16=1.25em**.

Try different combinations of text size and page zooming in a variety of browsers to ensure that the text remains readable.

165 COMMENTS

Q&A

The font-style Property

The font-style property is typically used to specify italic text.

The HTML:

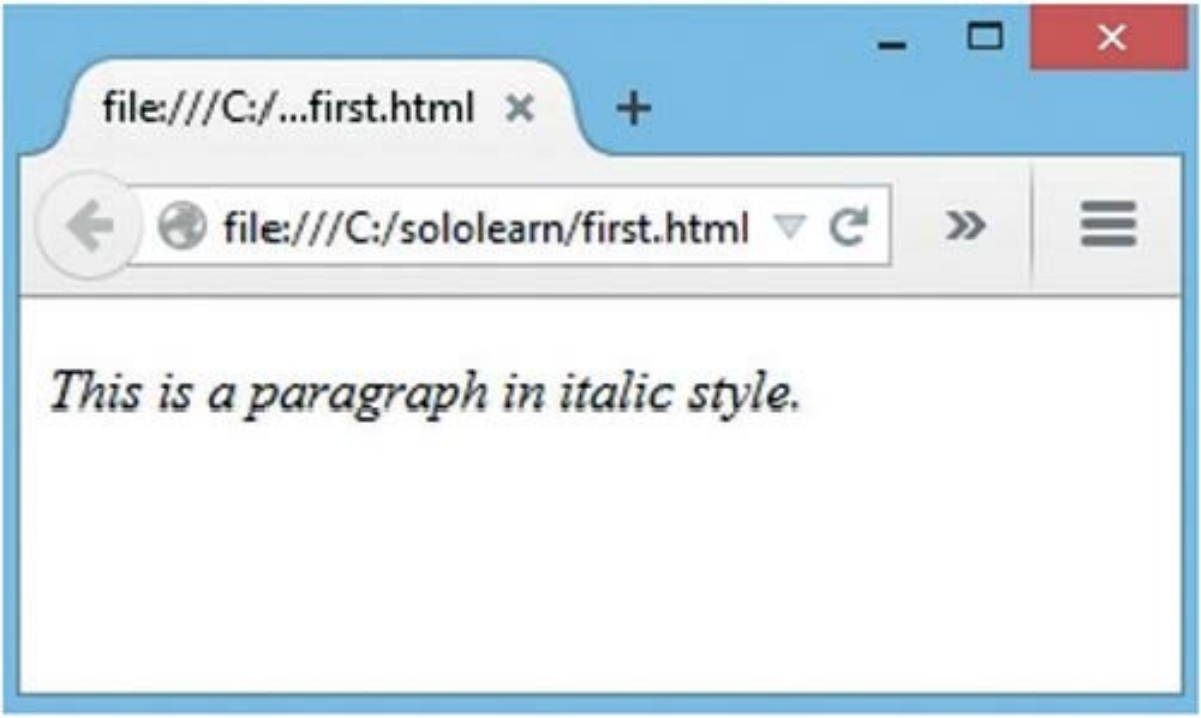
```
<p class="italic">This is a paragraph in italic style.</p>
```

The CSS:

```
p.italic {
  font-style: italic;
}
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

216 COMMENTS

The font-style Property

The font-style property has three values: **normal**, **italic**, and **oblique**.
Oblique is very similar to italic, but less supported.

The HTML:

```
<p class="normal">This paragraph is normal.</p>
<p class="italic">This paragraph is italic.</p>
<p class="oblique">This paragraph is oblique.</p>
```

The CSS:

```
p.normal {
  font-style: normal;
}
p.italic {
  font-style: italic;
}
p.oblique {
  font-style: oblique;
}
```

Try It Yourself

The HTML **<i>** tag will produce exactly the same result as the **italic font style**.

169 COMMENTS

The font-weight Property

The font-weight controls the boldness or thickness of the text. The values can be set as **normal** (default size), **bold**, **bolder**, and **lighter**.

The HTML:

```
<p class="light">This is a font with a "lighter" weight.</p>
<p class="bold">This is a font with a "bold" weight.</p>
<p class="bolder">This is a font with a "bolder" weight.</p>
```

The CSS:

```
p.light {
  font-weight: lighter;
}
p.bold {
  font-weight: bold;
}
p.bolder {
  font-weight: bolder;
}
```

Try It Yourself

Result:

The font-weight Property

You can also define the font weight with a number from **100** (thin) to **900** (thick), according to how thick you want the text to be.
400 is the same as normal, and 700 is the same as bold.

The HTML:

```
<p class="light">This is a font with a "lighter" weight.</p>
<p class="thick">This is a font with a "bold" weight.</p>
<p class="thicker">This is a font with a "700" weight.</p>
```

The CSS:

```
p.light {
  font-weight: lighter;
}
p.thick {
  font-weight: bold;
}
p.thicker {
  font-weight: 700;
}
```

Try It Yourself

Result:

The font-variant Property

The CSS font-variant property allows you to convert your font to all small caps. The values can be set as **normal**, **small-caps**, and **inherit**.

The HTML:

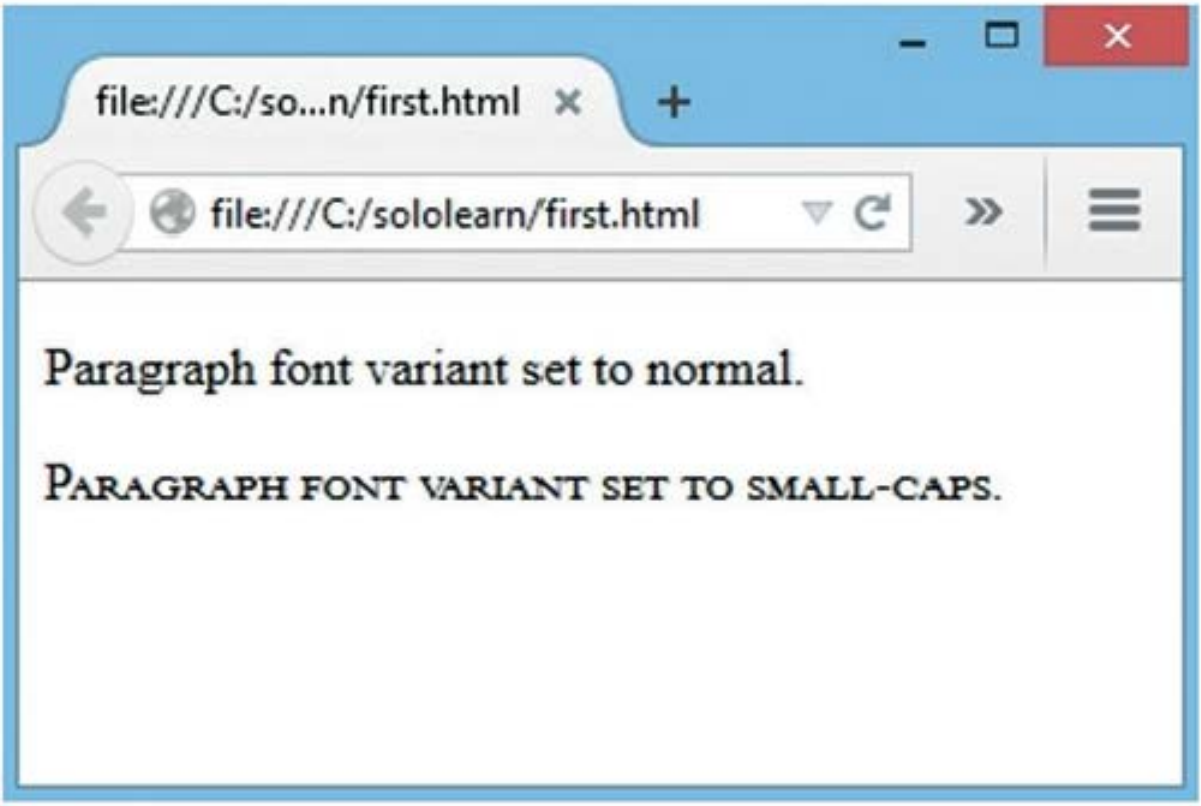
```
<p class="normal">Paragraph font variant set to normal.</p>
<p class="small">Paragraph font variant set to small-caps.</p>
```

The CSS:

```
p.normal {
  font-variant: normal;
}
p.small {
  font-variant: small-caps;
}
```

Try It Yourself

Result:



Not every font supports CSS font-variant, so be sure to test before you publish.

159 COMMENTS

The color Property

The CSS **color** property specifies the color of the text.
One method of specifying the color of the text is using a **color name**: like red, green, blue, etc.
Here's an example of changing the color of your font.

The HTML:

```
<p class="example">The text inside the paragraph is green.</p>
The text outside the paragraph is black (by default).
```

The CSS:

```
p.example {
  color: green;
}
```

Try It Yourself

Result:

Tap **Try It Yourself** to play around with the code!

213 COMMENTS

The color Property

Another way of defining colors is using **hexadecimal values** and **RGB**.
Hexadecimal form is a pound sign (#) followed by at most, **6 hex values** (0-F).
RGB defines the individual values for **Red**, **Green**, and **Blue**.

In the example below, we use hexadecimal value to set the heading color to blue, and RGB form to make the paragraph red.

The HTML:

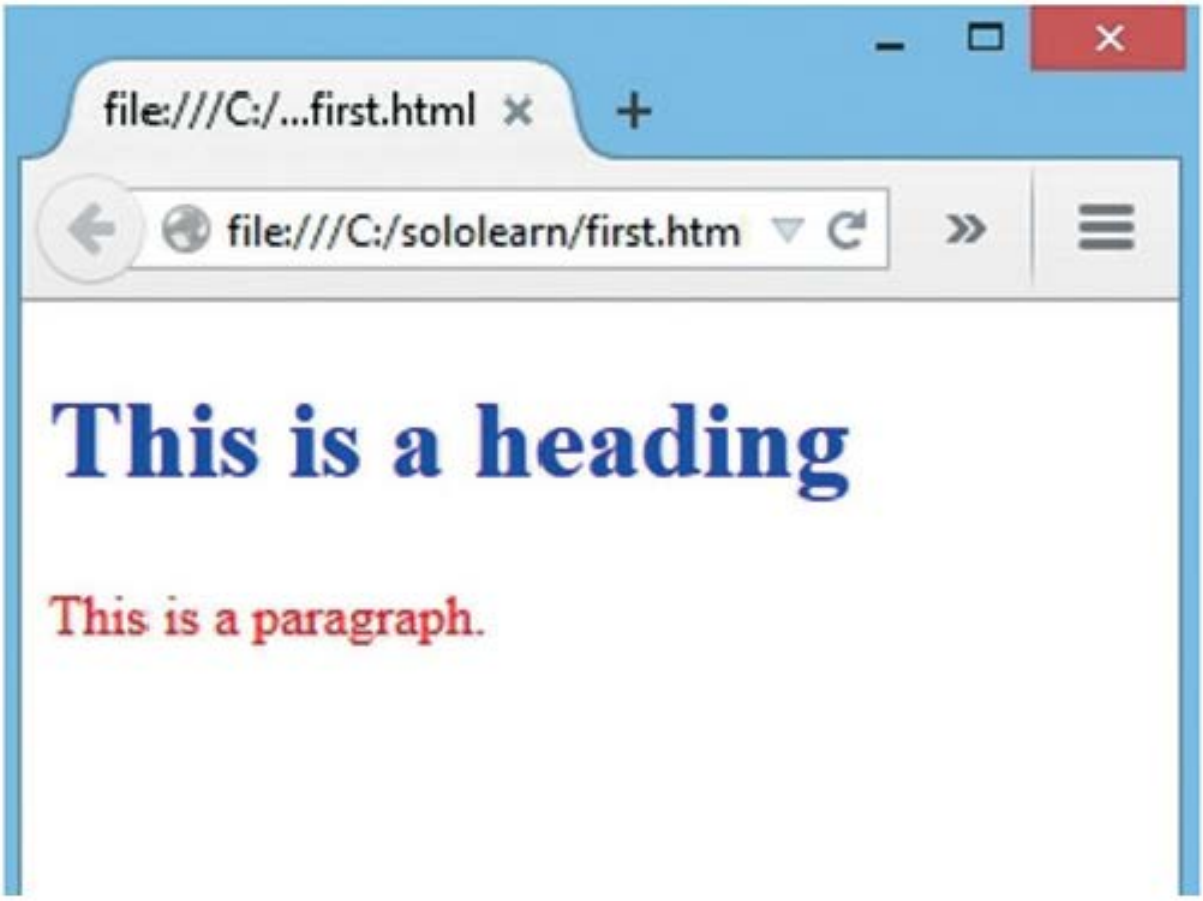
```
<h1>This is a heading</h1>
<p class="example">This is a paragraph</p>
```

The CSS:

```
h1 {
  color: #0000FF;
}
p.example {
  color: rgb(255,0,0);
}
```

Try It Yourself

Result:



The text-align Property

The text-align property specifies the horizontal alignment of text in an element. By default, text on your website is aligned to the left. However, at times you may require a different alignment.

text-align property values are as follows: **left**, **right**, **center**, and **justify**.

The HTML:

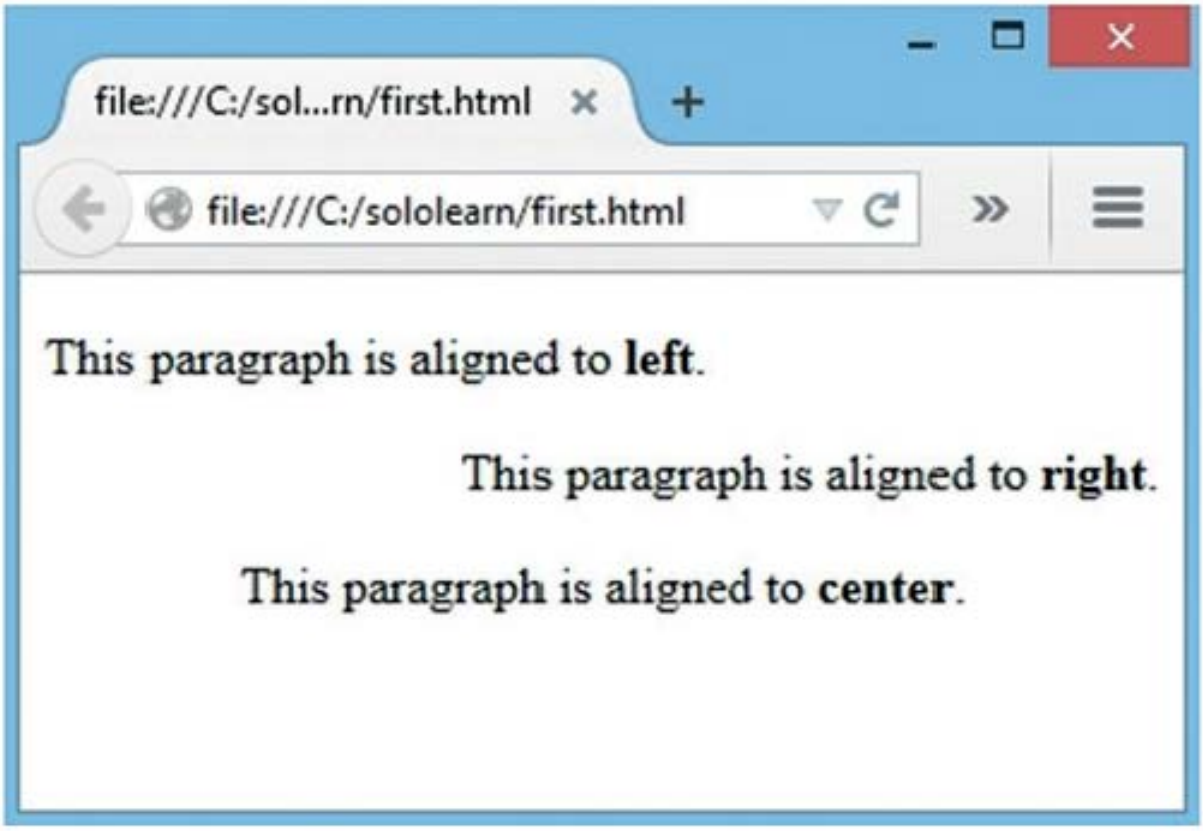
```
<p class="left">This paragraph is aligned to <strong>left.</strong></p>
<p class="right">This paragraph is aligned to <strong>right.</strong></p>
<p class="center">This paragraph is aligned to <strong>center.</strong></p>
```

The CSS:

```
p.left {
  text-align: left;
}
p.right {
  text-align: right;
}
p.center {
  text-align: center;
}
```

Try It Yourself

Result:



The vertical-align Property

The vertical-align property sets an element's vertical alignment. Commonly used values are **top**, **middle**, and **bottom**.

The example below shows how to vertically align the text between the table.

The HTML:

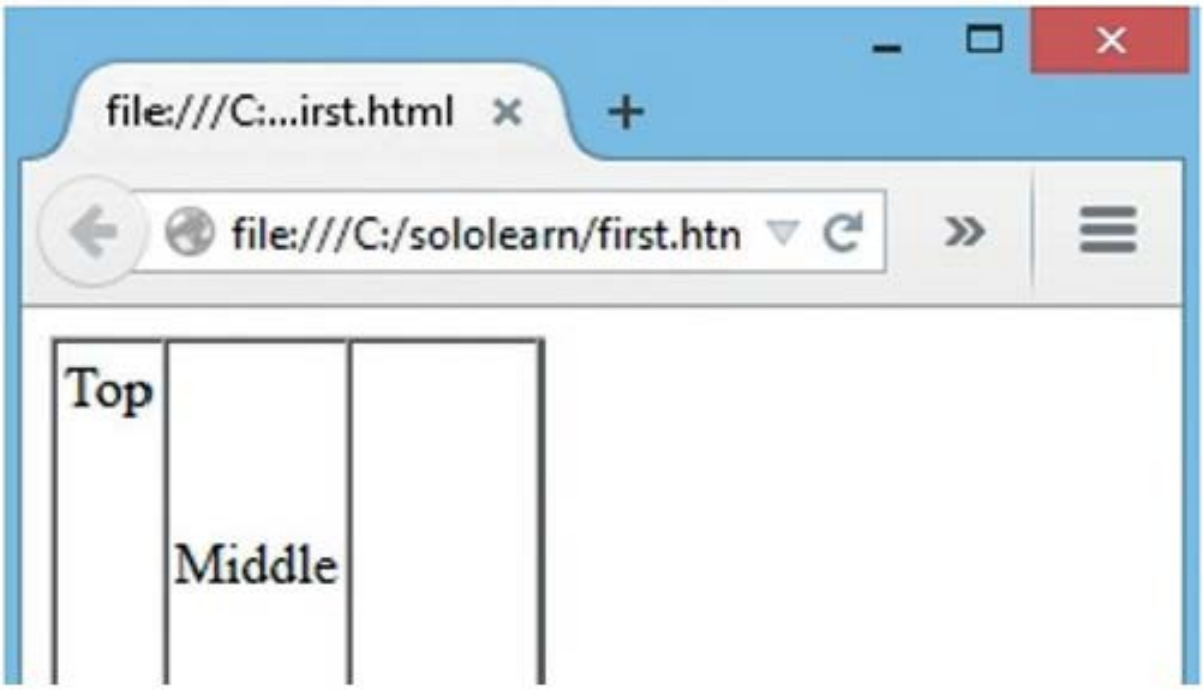
```
<table border="1" cellpadding="2" cellspacing="0" style="height: 150px;">
 <tr>
   <td class="top">Top</td>
   <td class="middle">Middle</td>
   <td class="bottom">Bottom</td>
 </tr>
</table>
```

The CSS:

```
td.top {
  vertical-align: top;
}
td.middle {
  vertical-align: middle;
}
td.bottom {
  vertical-align: bottom;
}
```

Try It Yourself

Result:



The vertical-align Property

The vertical-align property also takes the following values: **baseline**, **sub**, **super**, **%** and **px** (or pt, cm).
The example below shows the difference between them.

The HTML:

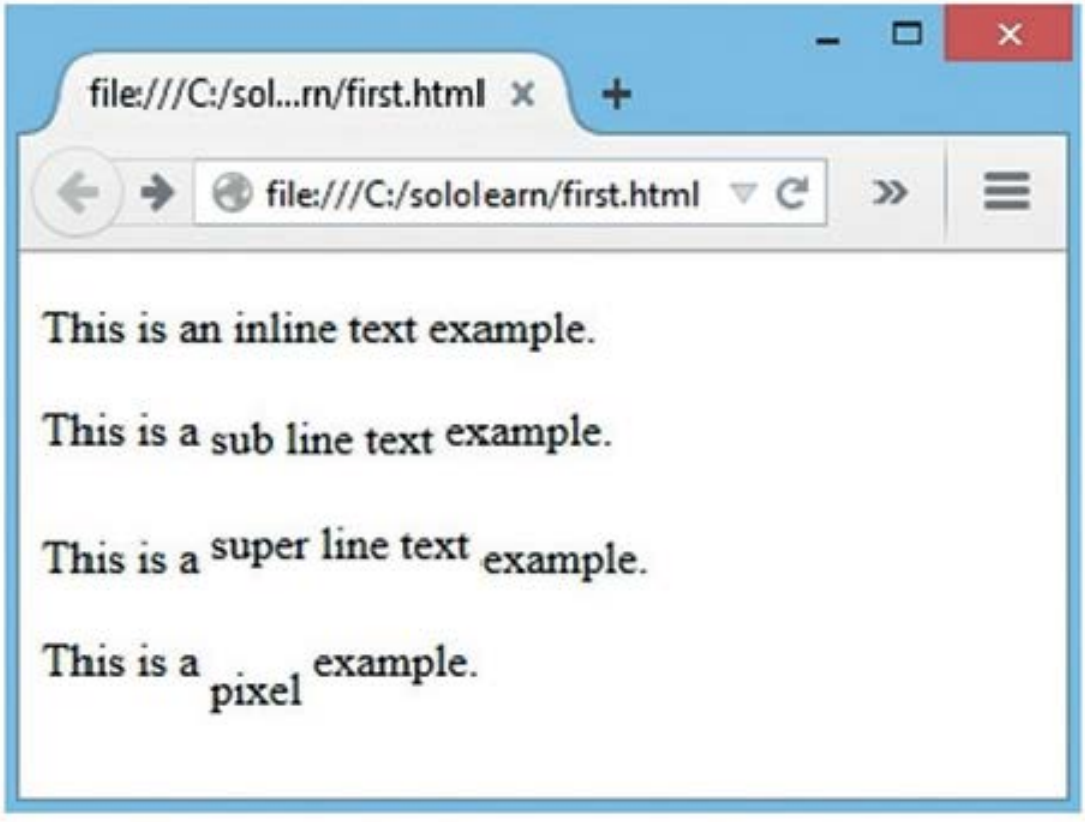
```
<p>This is an <span class="baseline">inline text</span> example.</p>
<p>This is a <span class="sub">sub line text</span> example.</p>
<p> This is a <span class="super">super line text</span> example.</p>
<p> This is a <span class="pixel">pixel</span> example.</p>
```

The CSS:

```
span.baseline {
   vertical-align: baseline;
}
span.sub {
   vertical-align: sub;
}
span.super {
   vertical-align: super;
}
span.pixel {
   vertical-align: -10px;
}
```

Try It Yourself

Result:



The vertical-align Property

Vertical align property does not act the same way for all elements.
For example, some additional CSS styling is needed for div elements.

The HTML:

```
<div class="main">
  <div class="paragraph">
  This text is aligned to the middle
  </div>
</div>
```

The CSS:

```
.main {
   height: 150px; width: 400px;
   background-color: LightSkyBlue;
   display: inline-table;
}
.paragraph {
   display: table-cell;
   vertical-align: middle;
}
```

Try It Yourself

Result:

The text-decoration Property

The text-decoration property specifies how the text will be decorated.

Commonly used values are:
none - The default value, this defines a normal text
inherit - Inherits this property from its parent element
overline - Draws a horizontal line above the text
underline - Draws a horizontal line below the text
line-through - draws a horizontal line through the text (substitutes the HTML **<s>** tag)

The example below demonstrates the difference between each value.

The HTML:

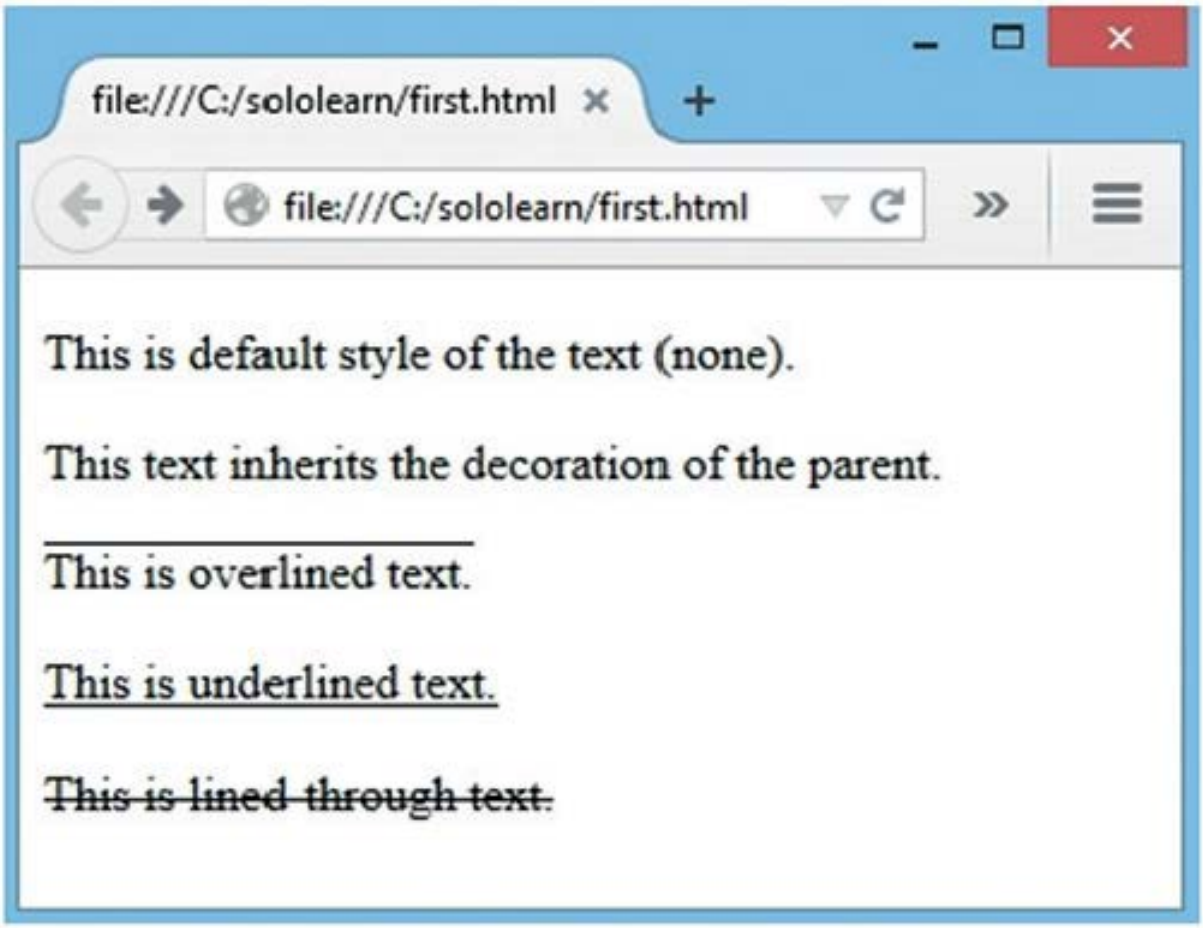
```
<p class="none">This is default style of the text (none).</p>
<p class="inherit">This text inherits the decoration of the parent.</p>
<p class="overline">This is overlined text.</p>
<p class="underline">This is underlined text.</p>
<p class="line-through">This is lined-through text.</p>
```

The CSS:

```
p.none {
  text-decoration: none;
}
p.inherit {
  text-decoration: inherit;
}
p.overline {
  text-decoration: overline;
}
p.underline {
  text-decoration: underline;
}
p.line-through {
  text-decoration: line-through;
}
```

Try It Yourself

Result:



You can combine the **underline**, **overline**, or **line-through** values in a space-separated list to add multiple decoration lines.

211 COMMENTS

The text-decoration Property

Another value of text-decoration property is **blink** which makes the text blink.

CSS syntax looks like this:

```
text-decoration: blink;
```

This value is valid but is deprecated and most browsers ignore it.

200 COMMENTS

The text-indent Property

The text-indent property specifies how much horizontal space should be left before the beginning of the first line of the text. Property values are **length** (px, pt, cm, em, etc.), **%**, and **inherit**.

The HTML:

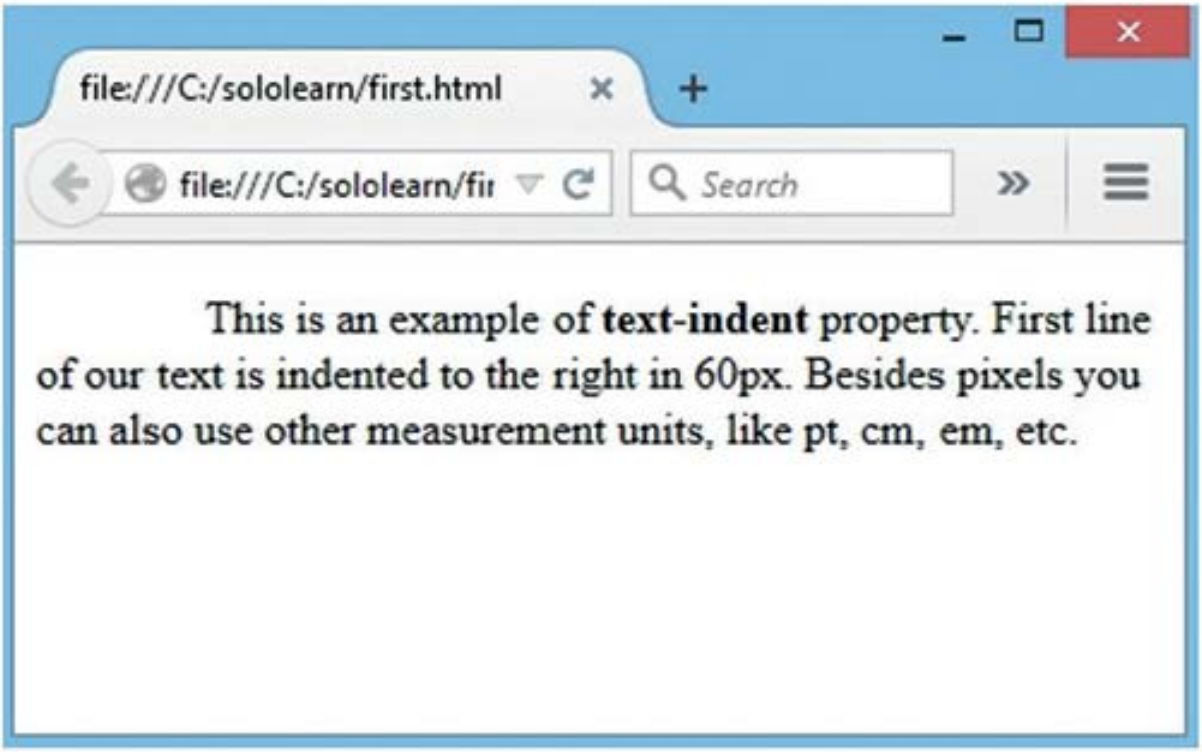
```
<p>This is an example of
<strong>text-indent </strong> property.
First line of our text is indented to the right in 60px.
Besides pixels you can also use other measurement units,
like pt, cm, em, etc. </p>
```

The CSS:

```
p {
  text-indent: 60px;
}
```

Try It Yourself

Result:



Negative values are allowed. The first line will be indented to the left if the value is negative.

145 COMMENTS

The text-shadow Property

The text-shadow property adds shadow to text.
It takes four values: the first value defines the distance of the shadow in the **x (horizontal) direction**, the second value sets the distance in the **y (vertical) direction**, the third value defines the **blur** of the shadow, and the fourth value sets the **color**.

The HTML:

```
<h1>Text-shadow example</h1>
```

The CSS:

```
h1 {
  color: blue;
  font-size: 30pt;
  text-shadow: 5px 2px 4px grey;
}
```

Try It Yourself

In the example above, we created a shadow using the following parameters:
5px – the X-coordinate
2px – the Y-coordinate
4px – the blur radius
grey – the color of the shadow

Result:

text-shadow with Blur Effect

When working with shadows, you can use any CSS-supported color format.

For the x and y offsets, various types of units can be used (like **px, cm, mm, in, pc, pt,** etc). Negative values are also supported.

The example below creates a blue drop-shadow, two pixels higher than the main text, one pixel to the left of it, and with a 0.5em blur.

The HTML:

```
<h1>Text-shadow with blur effect</h1>
```

The CSS:

```
h1 {
  font-size: 20pt;
  text-shadow: rgba(0,0,255,1) -1px -2px 0.5em;
}
```

Try It Yourself

Result:

Internet Explorer 9 and earlier do not support the text-shadow property.

199 COMMENTS

The text-transform Property

The text-transform CSS property specifies how to capitalize an element's text. For example, it can be used to make text appear with each word capitalized.

The HTML:

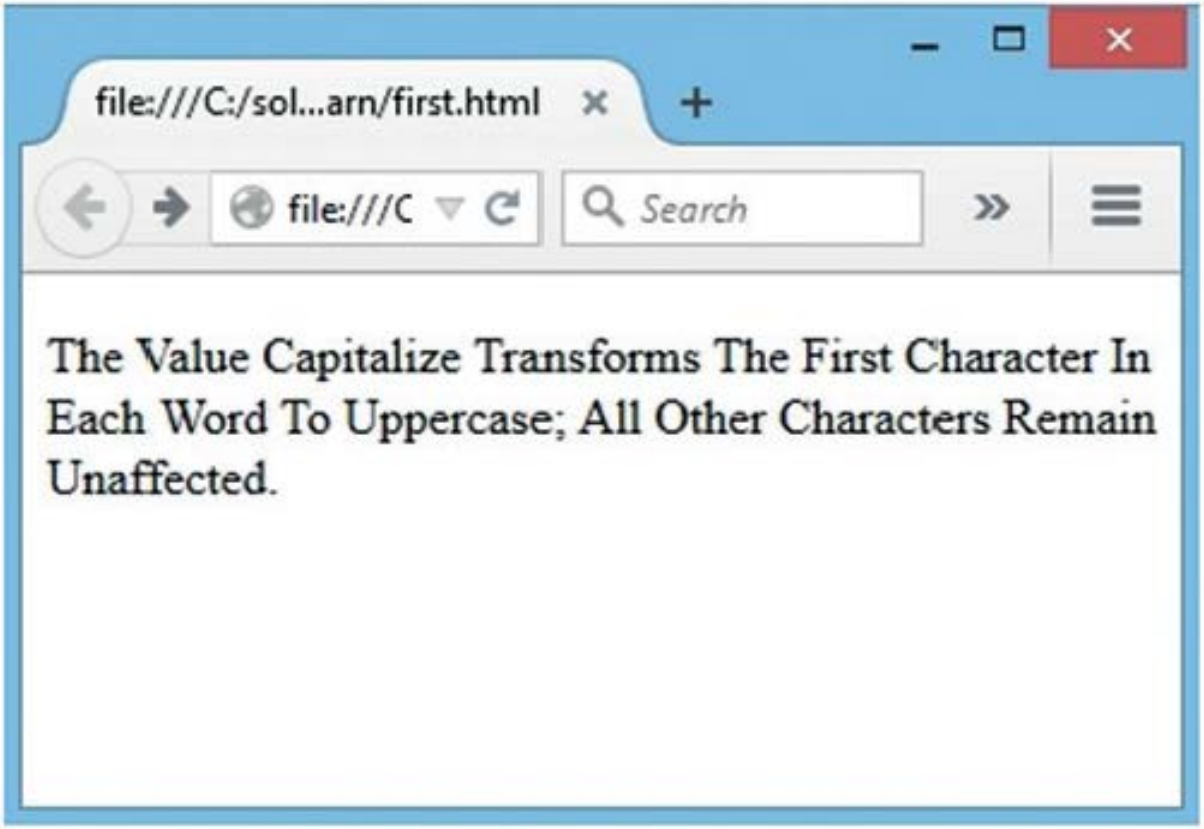
```
<p class="capitalize">
  The value capitalize transforms the first
  character in each word to uppercase;
  all other characters remain unaffected.
</p>
```

The CSS:

```
p.capitalize {
  text-transform: capitalize;
}
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

153 COMMENTS

text-transform Values

Using text-transform property you can make text appear in all-uppercase or all-lowercase. Here is an example:

The HTML:

```
<p class="uppercase">This value transforms all characters to uppercase.</p>
<p class="lowercase">This value transforms all characters to lowercase.</p>
```

The CSS:

```
p.uppercase {
  text-transform: uppercase;
}
p.lowercase {
  text-transform: lowercase;
}
```

Try It Yourself

Result:

The value **none** will produce no capitalization effect at all.

131 COMMENTS

The letter-spacing Property

The letter-spacing property specifies the **space between characters** in a text. The values can be set as:
- **normal** defines the default style with no extra space between characters
- **length** defines an extra space between characters using measurement units like px, pt, cm, mm, etc.;
- **inherit** inherits the property from its parent element;

The HTML:

```
<p class="normal">This paragraph has no additional letter-spacing applied.</p>
<p class="positive ">This paragraph is letter-spaced at 4px.</p>
```

The CSS:

```
p.normal {
  letter-spacing: normal;
}
p.positive {
  letter-spacing: 4px;
}
```

Try It Yourself

Result:

Tap **Try It Yourself** to play around with the code!

Using Negative Values

For defining an extra space between characters, negative values are also permitted.
Here is an example demonstrating the difference between **positive** and **negative** values:

The HTML:

```
<p class="positive">This paragraph is letter-spaced at 4px.</p>
<p class="negative">This paragraph is letter-spaced at -1.5px</p>
```

The CSS:

```
p.positive {
  letter-spacing: 4px;
}
p.negative {
  letter-spacing: -1.5px;
}
```

Try It Yourself

Result:

Always test your result, to ensure the text is readable.

59 COMMENTS

The word-spacing Property

The word-spacing property specifies the **space between words** in a text. Just like the letter-spacing property, you can set the word-spacing values as **normal**, **length**, and **inherit**.

The HTML:

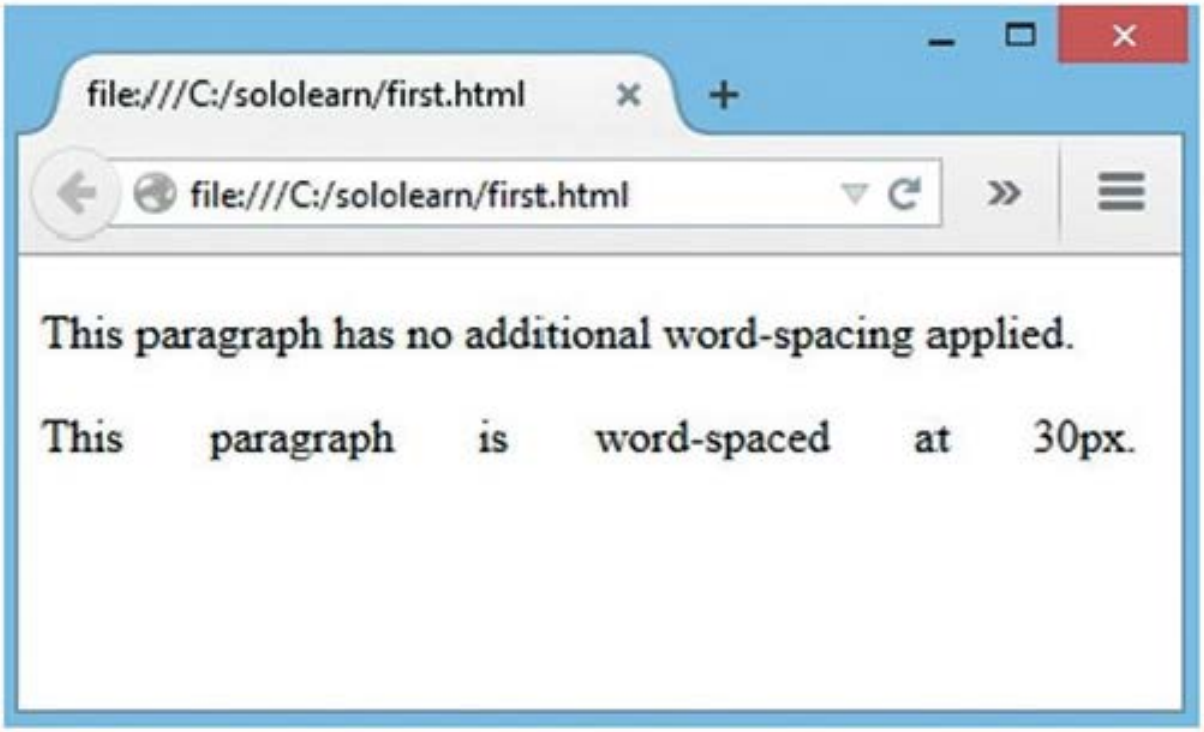
```
<p class="normal">This paragraph has no additional word-spacing applied.</p>
<p class="px">This paragraph is word-spaced at 30px.</p>
```

The CSS:

```
p.normal {
  word-spacing: normal;
}
p.px {
  word-spacing: 30px;
}
```

Try It Yourself

Result:



When a weird spacing is used, and it is necessary to keep the selected paragraph with normal word spacing, the **normal** option is usually used.

75 COMMENTS

Measurement Units

To define an extra space between words, you can use positive measurement values like **px, pt, pc, cm, mm, inches, em,** and **ex**.
Negative values are also permitted. Here is an example to show the difference.

The HTML:

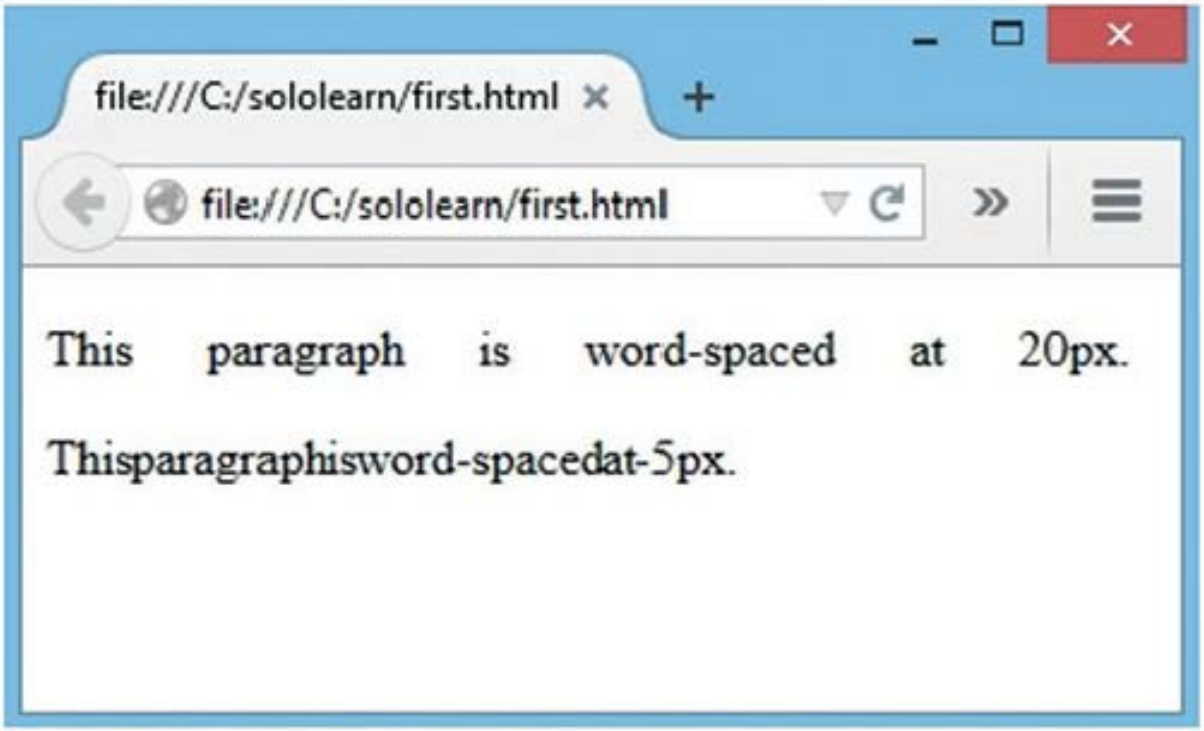
```
<p class="positive">This paragraph is word-spaced at 20px.</p>
<p class="negative">This paragraph is word-spaced at -5px.</p>
```

The CSS:

```
p.positive {
  word-spacing: 20px;
}
p.negative {
  word-spacing: -5px;
}
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

82 COMMENTS

The white-space Property

The white-space property specifies how white-space inside an element is handled. The values can be set as **normal, inherit, nowrap**, etc.

The **nowrap** value makes the text continue on the same line until a
 tag is encountered, and also collapses all sequences of whitespace into a single whitespace.

The HTML:

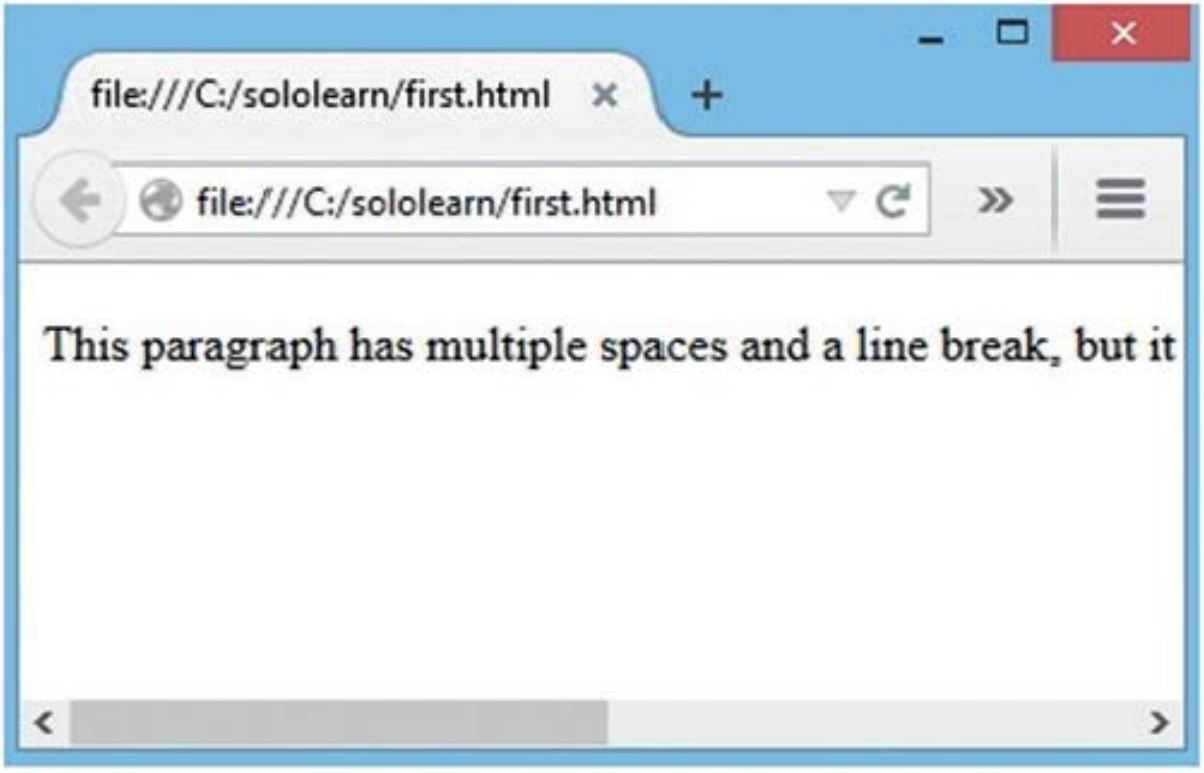
```
<p>
This paragraph has       multiple spaces     and
a line break, but it will be ignored, as we used the nowrap value.
</p>
```

The CSS:

```
p {
  white-space: nowrap;
}
```

Try It Yourself

Result:



The text will continue on the same line until a **
** tag is encountered.

The white-space Values

The white-space property also supports other values:
pre - text will only wrap on line breaks and white space
pre-line - text will wrap where there is a break in code, but extra white space is still ignored
pre-wrap - text will wrap when necessary, and on line breaks

Here is an example in which all three values are used:

The HTML:

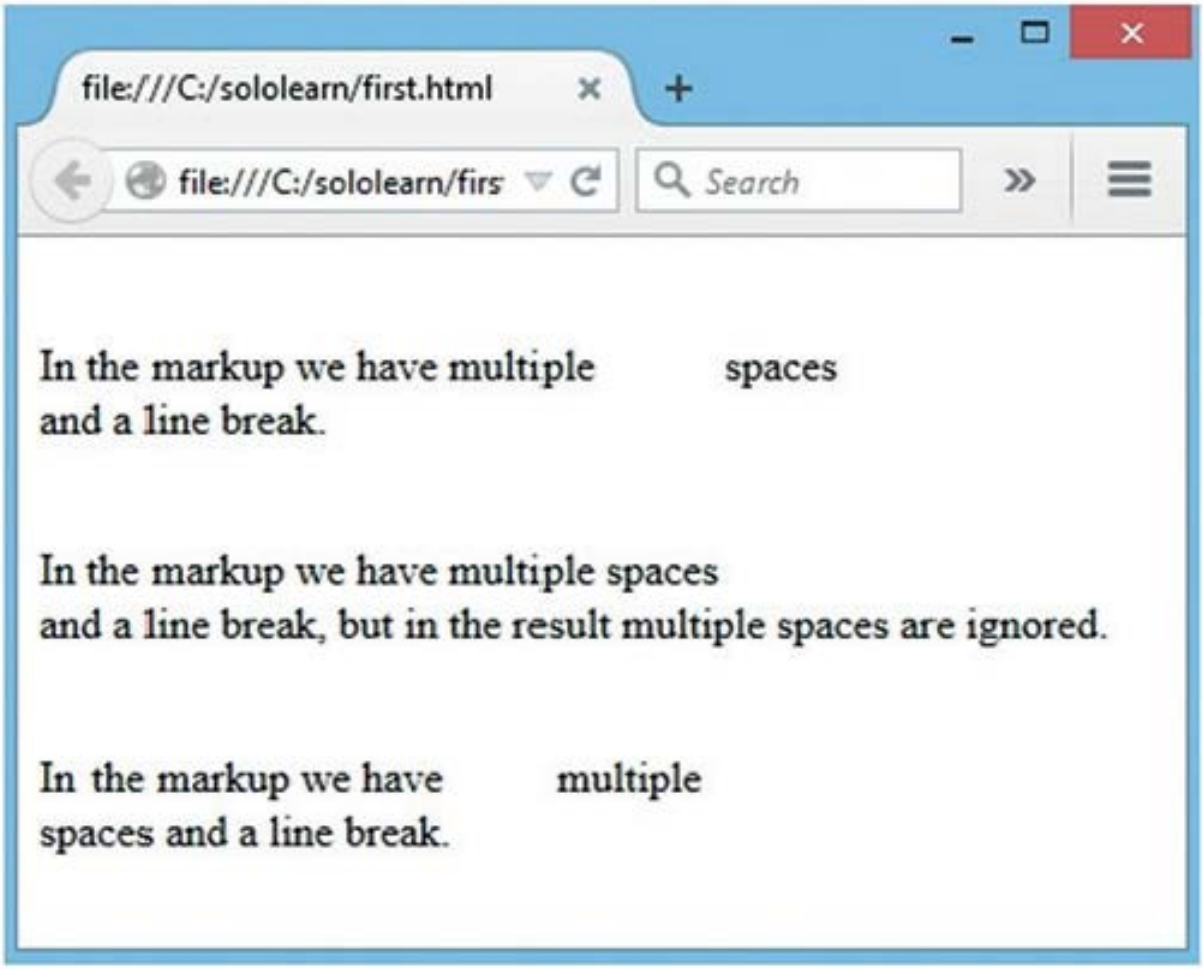
```
<p class="pre">
In the markup we have multiple          spaces
and a line break.
</p>
<p class="preline">
In the markup we have multiple          spaces
and a line break, but in the result multiple spaces are ignored.
</p>
<p class="prewrap">
In the markup we have          multiple
spaces and a line break.
</p>
```

The CSS:

```
p.pre {
  white-space: pre;
}
p.preline {
  white-space: pre-line;
}
p.prewrap {
  white-space: pre-wrap;
}
```

Try It Yourself

Result:



Pre-wrap value behaves as the **pre** value, except that it adds extra line breaks to prevent the text breaking out of the element's box.

211 COMMENTS

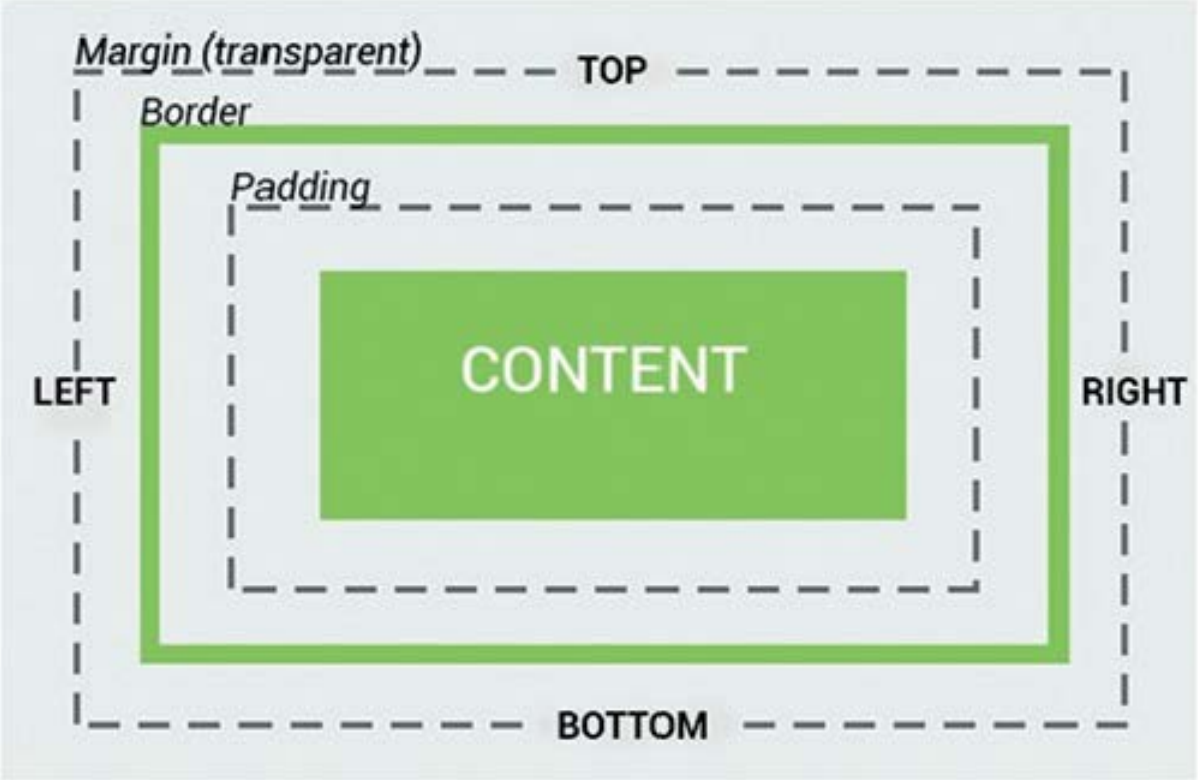
Properties

The CSS Box Model

All HTML elements can be considered as boxes. The CSS box model represents the design and layout of the site. It consists of **margins, borders, paddings,** and the actual **content**.

The properties work in the same order: **top, right, bottom,** and **left**.

The image below illustrates the box model:



The term "box model" is used when talking about design and layout.

275 COMMENTS

More on Box Models

Every element of the webpage is a **box**.

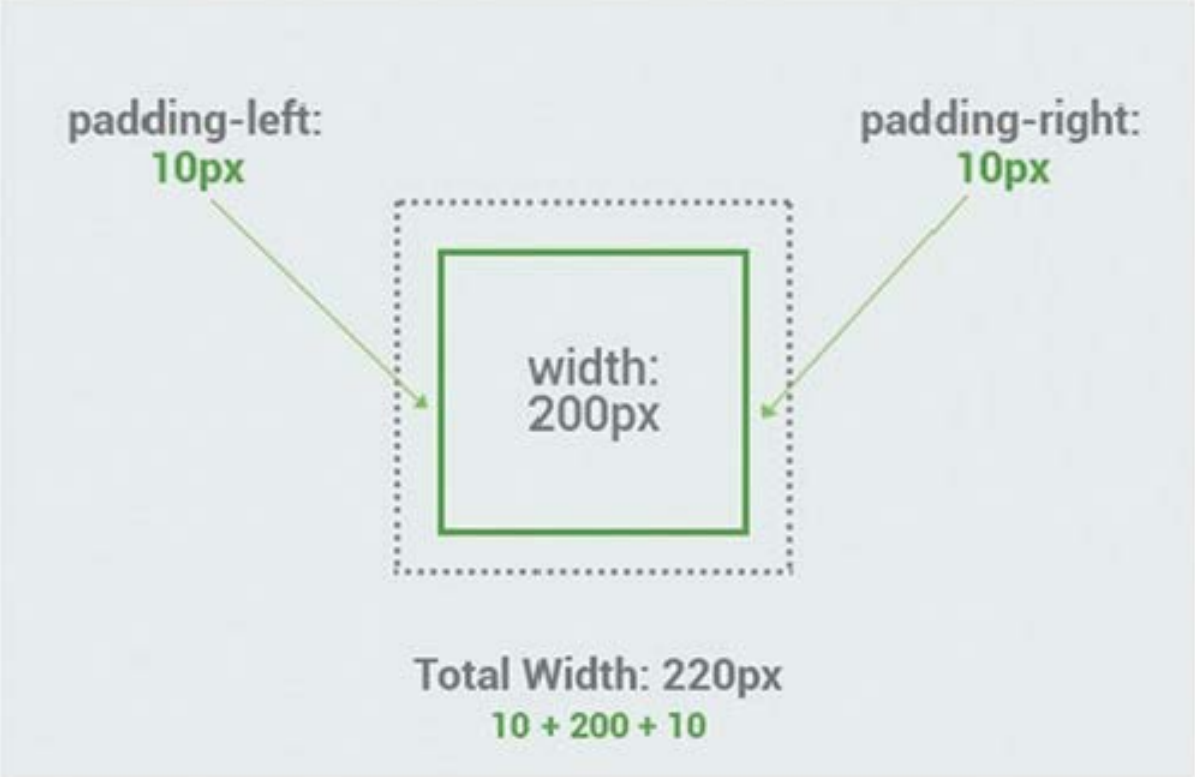
CSS uses the box model to determine how big the boxes are and how to place them.

> The box model is also used to calculate the actual width and height of the HTML elements.

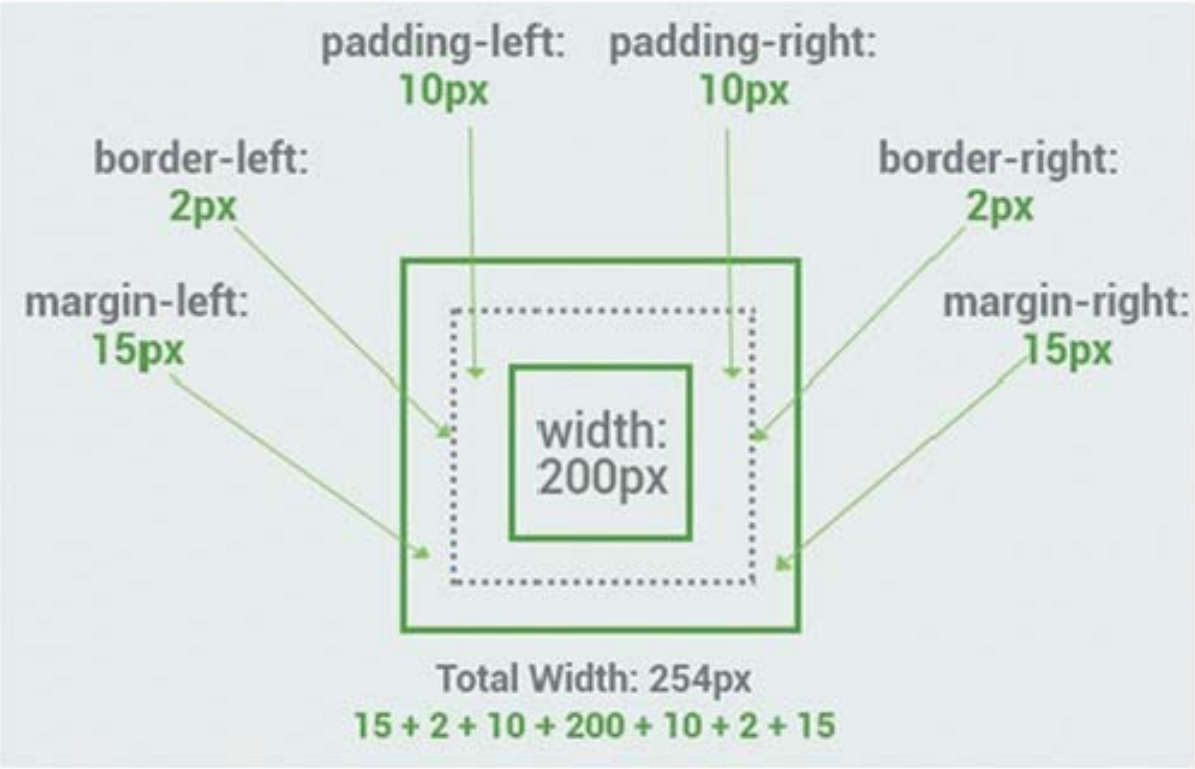
139 COMMENTS

Total Width of an Element

When working with boxes, it is important to understand how the total width of an element is calculated.
For example, the total width of the box with paddings will be the sum of **width** plus **padding left** and **padding right**.

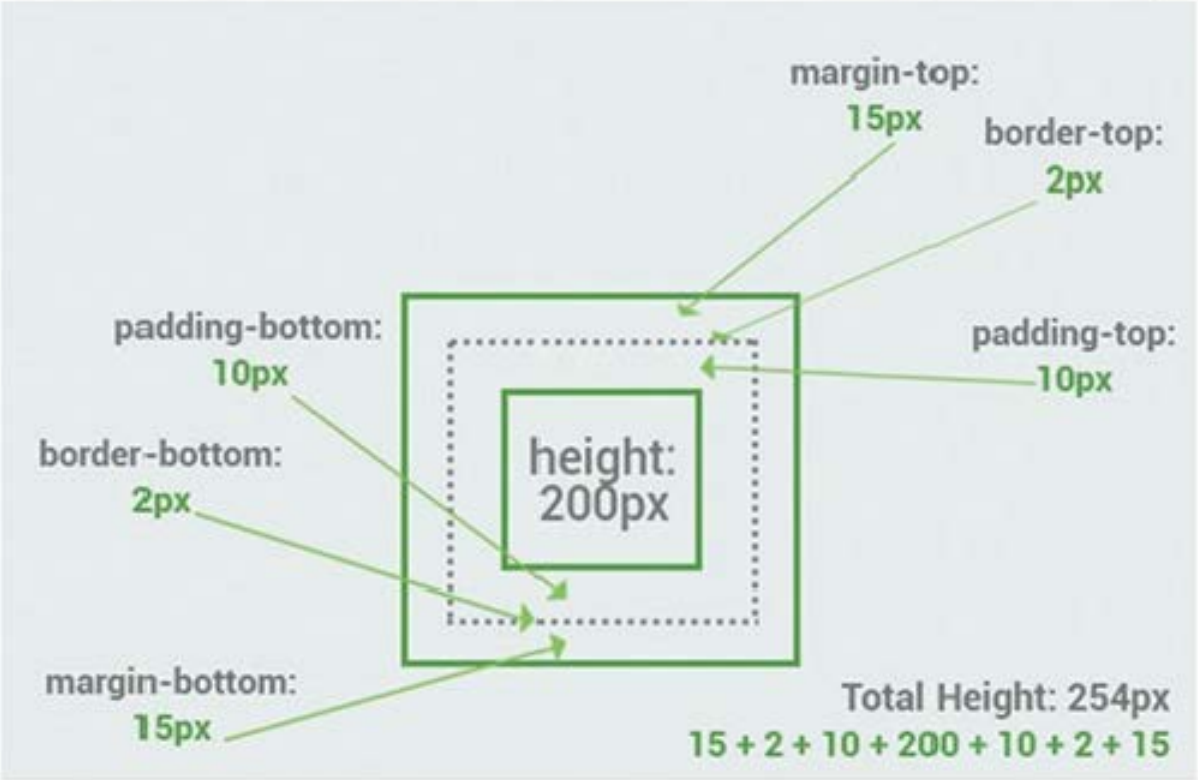


Here is another box with margins, border, and paddings.
The total width is the sum of **left and right margins, left and right borders, left and right paddings,** and the **actual width** of the content.



Total Height of an Element

The total height of an element is calculated the same way as the width.
The example below is the same box from the previous lesson with padding, border and margin.



To summarize, Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

80 COMMENTS

Q&A

The border Property

The CSS border property allows you to customize the borders of HTML elements.
In order to add a border to the element, you need to specify the **size**, **style**, and **color** of the border.
The example below shows how to add a solid green border to the paragraph.

The HTML:

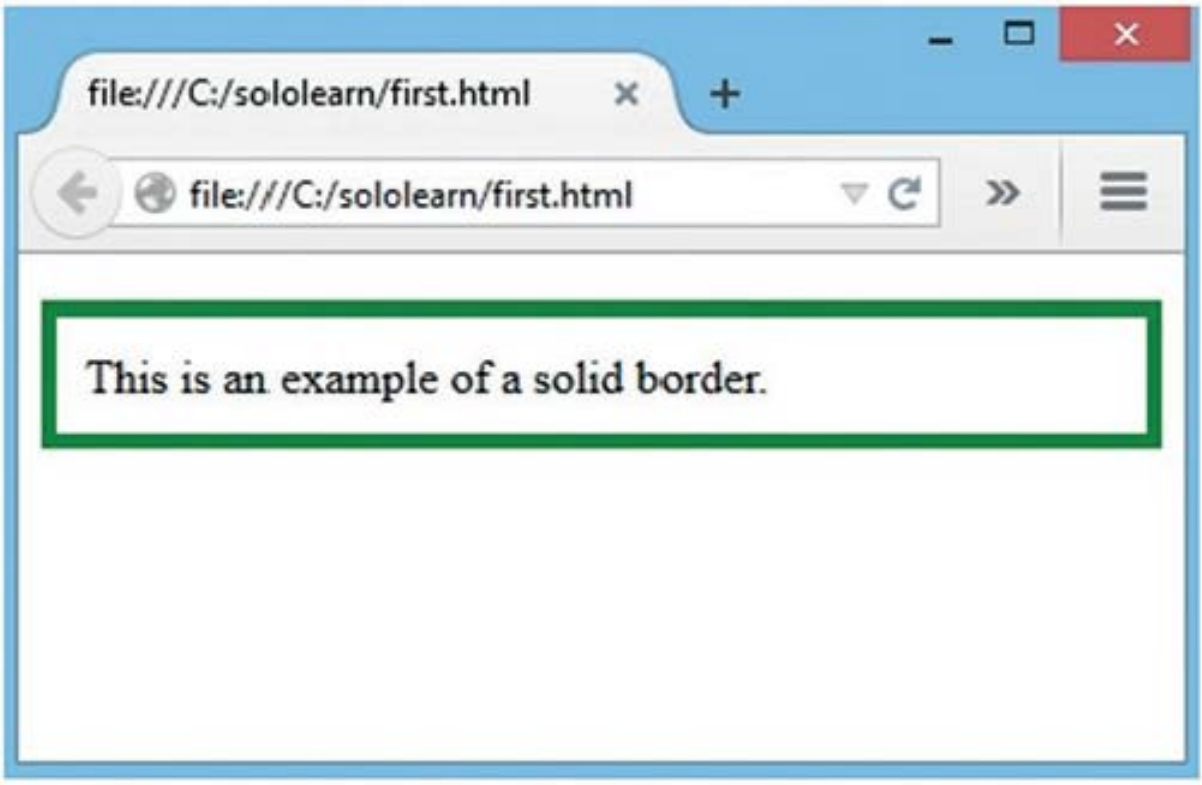
```
<p>This is an example of a solid border.</p>
```

The CSS:

```
p {
  padding: 10px;
  border: 5px solid green;
}
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

210 COMMENTS

Border Width

The properties for the border can be set separately. The **border-width** property specifies the width of the border.

The HTML:

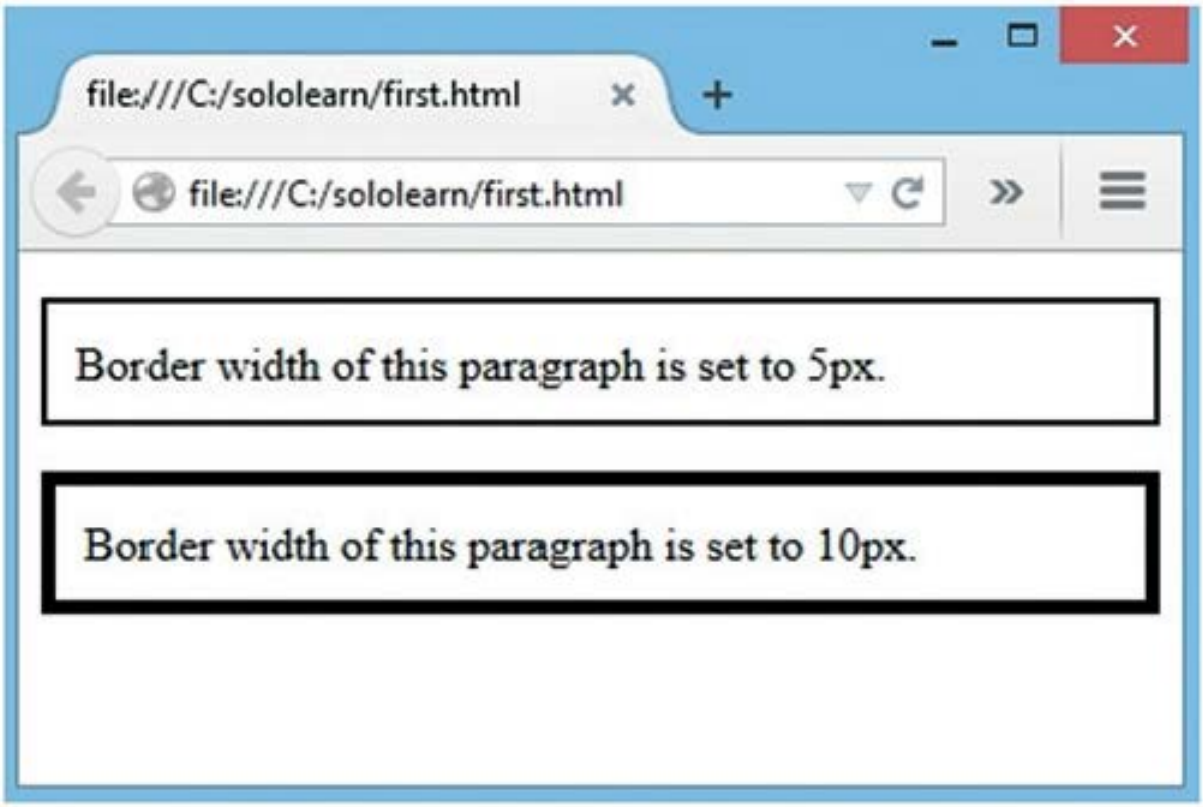
```
<p class="first">
  Border width of this paragraph is set to 2px.
</p>
<p class="second">
  Border width of this paragraph is set to 5px.
</p>
```

The CSS:

```
p.first {
  padding: 10px;
  border-style: solid;
  border-width: 2px;
}
p.second {
  padding: 10px;
  border-style: solid;
  border-width: 5px;
}
```

Try It Yourself

Result:



Border Color

The border color of the element can be defined using a color name, RGB, or Hex values.
The HTML:

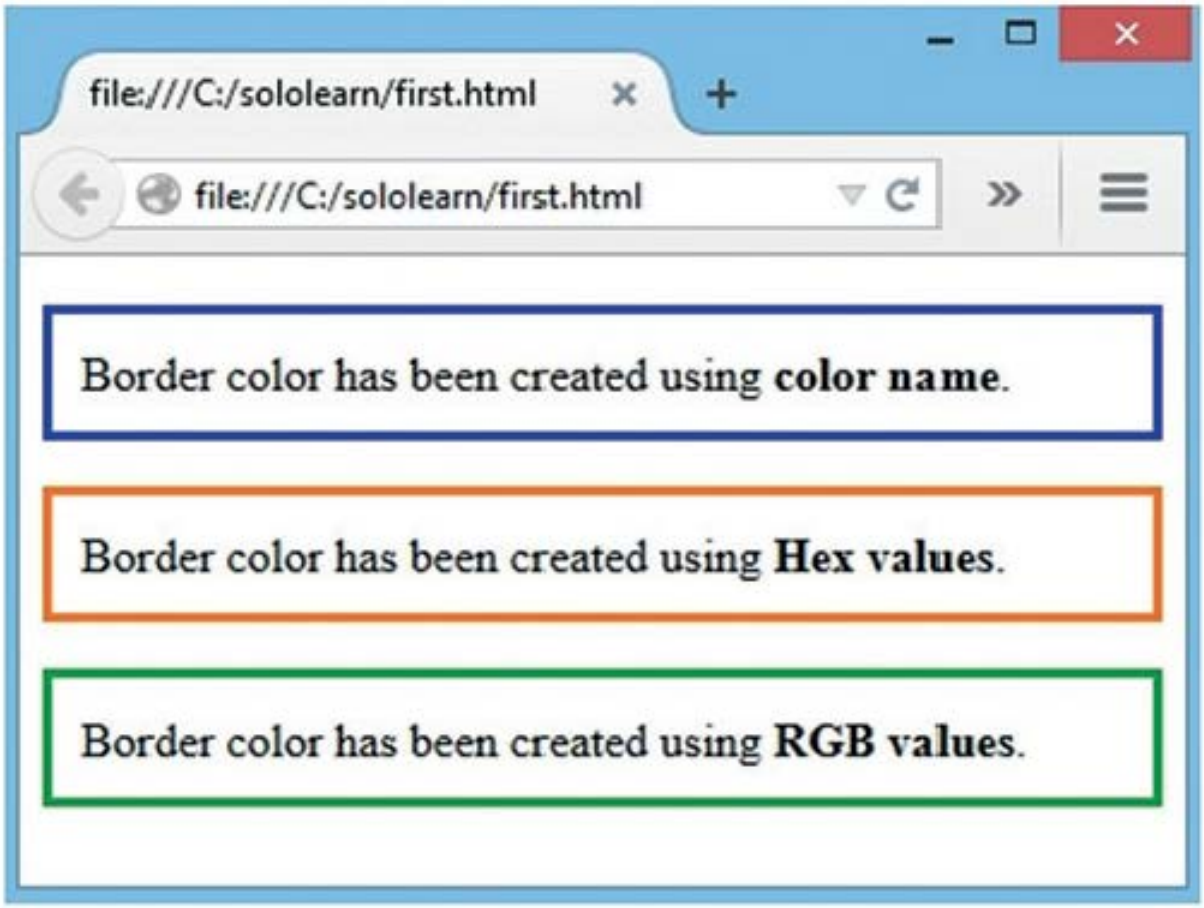
```
<p class="first">
  Border color has been created using <strong>color name.</strong>
</p>
<p class="second">
  Border color has been created using <strong>Hex values.</strong>
</p>
<p class="third">
  Border color has been created using <strong>RGB values.</strong>
</p>
```

The CSS:

```
p.first {
  padding: 10px;
  border-style: solid;
  border-width: 2px;
  border-color: blue;
}
p.second {
  padding: 10px;
  border-style: solid;
  border-width: 2px;
  border-color: #FF6600;
}
p.third {
  padding: 10px;
  border-style: solid;
  border-width: 2px;
  border-color: rgb(0, 153, 0);
}
```

Try It Yourself

Result:



None of the border properties will have any effect unless the **border-style** property is set.

The border-style Property

The default value of border-style is **none**, which defines no border.
There are various styles supported for the border-style property: **dotted**, **dashed**, **double**, etc. The example below illustrates the differences between them.

The HTML:

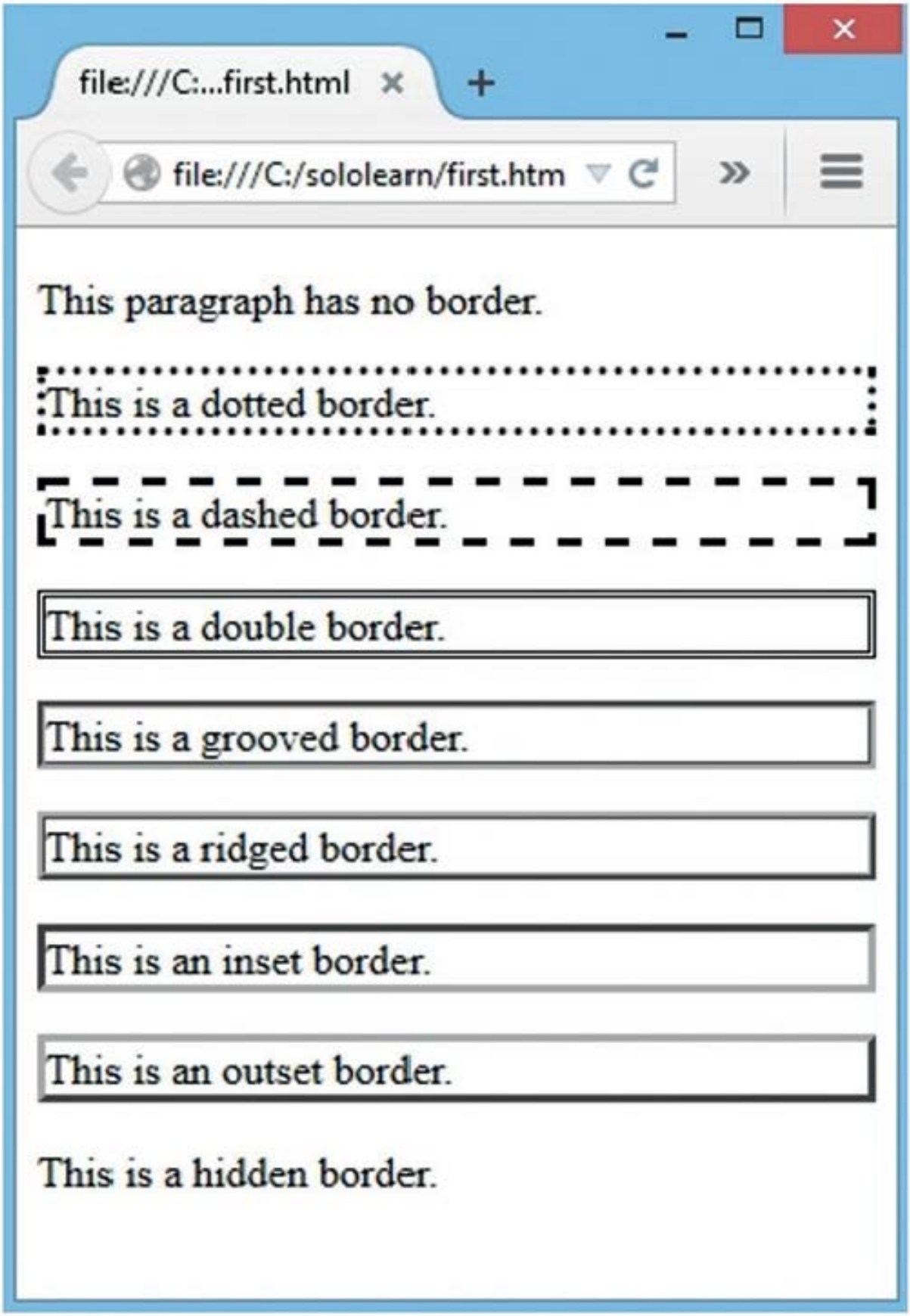
```
<p class="none">This paragraph has no border.</p>
<p class="dotted">This is a dotted border.</p>
<p class="dashed">This is a dashed border.</p>
<p class="double">This is a double border.</p>
<p class="groove">This is a grooved border.</p>
<p class="ridge">This is a ridged border.</p>
<p class="inset">This is an inset border.</p>
<p class="outset">This is an outset border.</p>
<p class="hidden">This is a hidden border.</p>
```

The CSS:

```
p.none {border-style: none;}
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.hidden {border-style: hidden;}
```

Try It Yourself

Result:



In CSS, it is possible to specify different borders for different sides, using the following properties: border-top-style, border-right-style, border-bottom-style, and border-left-style for the corresponding sides.

CSS Width and Height

To style a ** ** element to have a total width and height of **100px**:

The HTML:

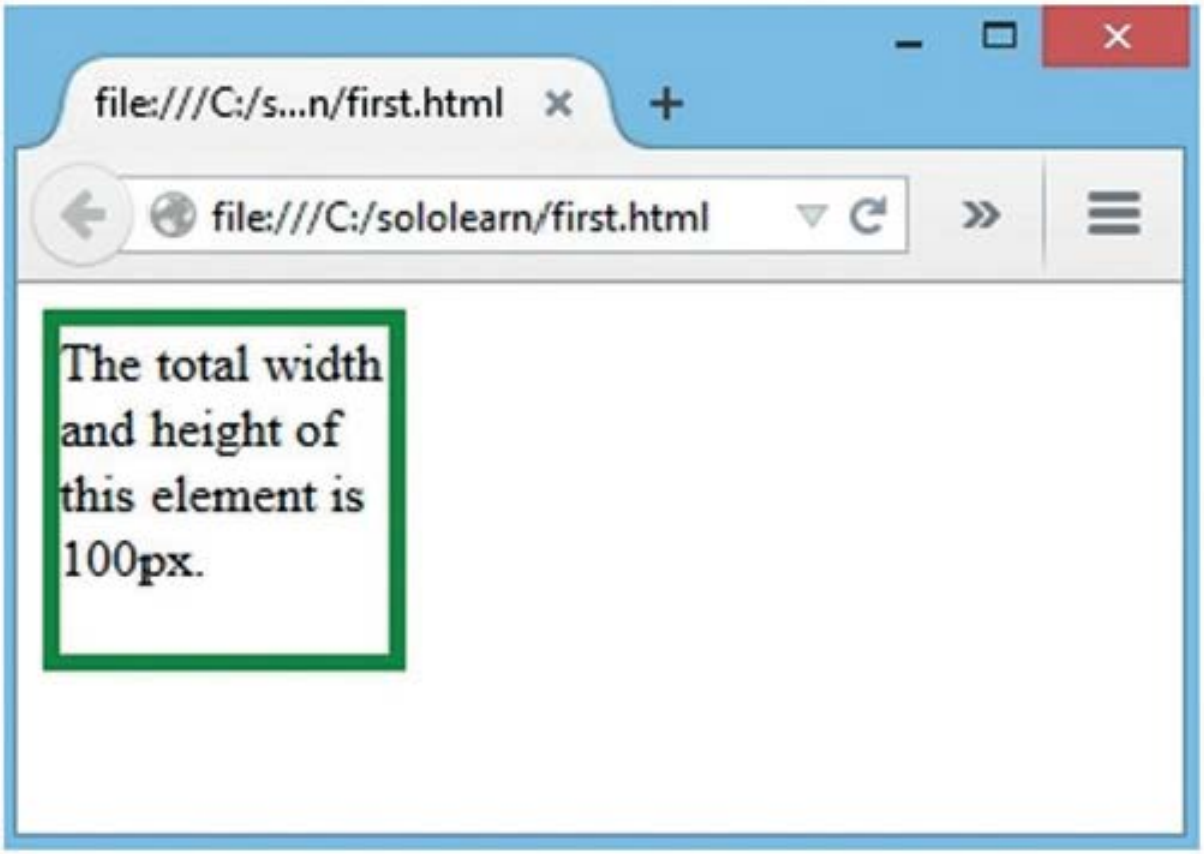
```
<div>The total width and height of this element is 100px.</div>
```

The CSS:

```
div {
  border: 5px solid green;
  width: 90px;
  height: 90px;
}
```

Try It Yourself

Result:



The total width and height of the box will be the 90px+5px (border)+5px(border) = 100px;

157 COMMENTS

Width and Height Measurement

The width and height of an element can be also assigned using **percents.**
In the example below the width of an element is assigned in percentages, the height is in pixels.

The HTML:

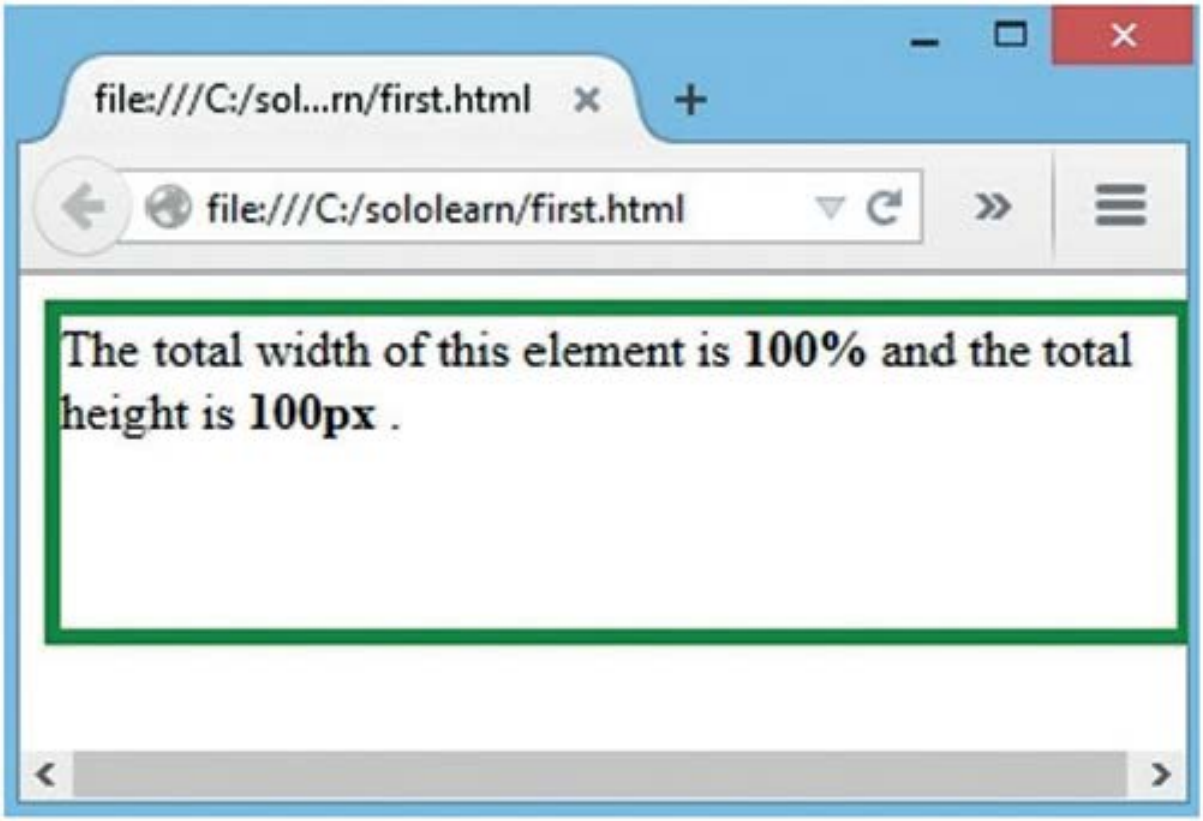
```
<div>The total width of this element is <strong>100%</strong> and the total height is
<strong>100px</strong> .</div>
```

The CSS:

```
div {
  border: 5px solid green;
  width: 100%;
  height: 90px;
}
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

151 COMMENTS

The Minimum and Maximum Sizes

To set the minimum and maximum height and width of an element, you can use the following properties:

min-width - the minimum width of an element
min-height - the minimum height of an element
max-width - the maximum width of an element
max-height - the maximum height of an element

In the example below, we set the minimum height and maximum width of different paragraphs to 100px.

The HTML:

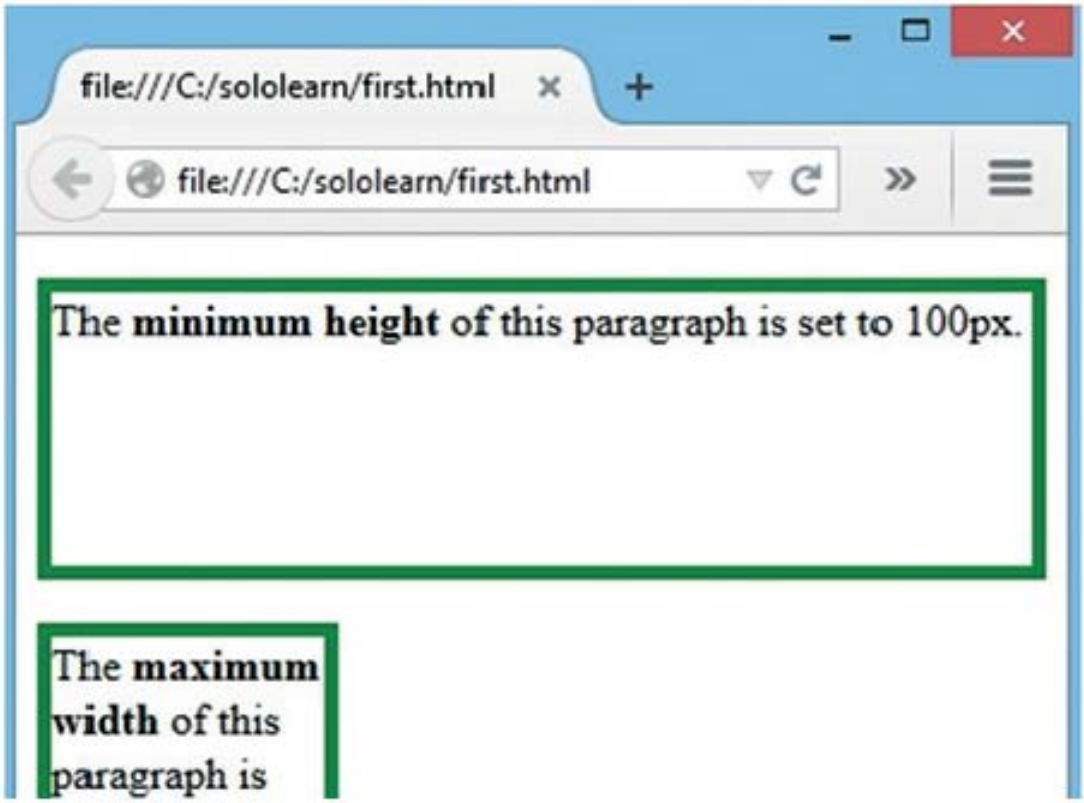
```
<p class="first">The <strong>minimum height </strong> of this paragraph is set to 100px.</p>
<p class="second">The<strong> maximum width </strong> of this paragraph is set to 100px.</p>
```

The CSS:

```
p.first {
  border: 5px solid green;
  min-height: 100px;
}
p.second {
  border: 5px solid green;
  max-width: 100px;
}
```

Try It Yourself

Result:



The background-color Property

The **background-color** property is used to specify the background color of an element.

The HTML:

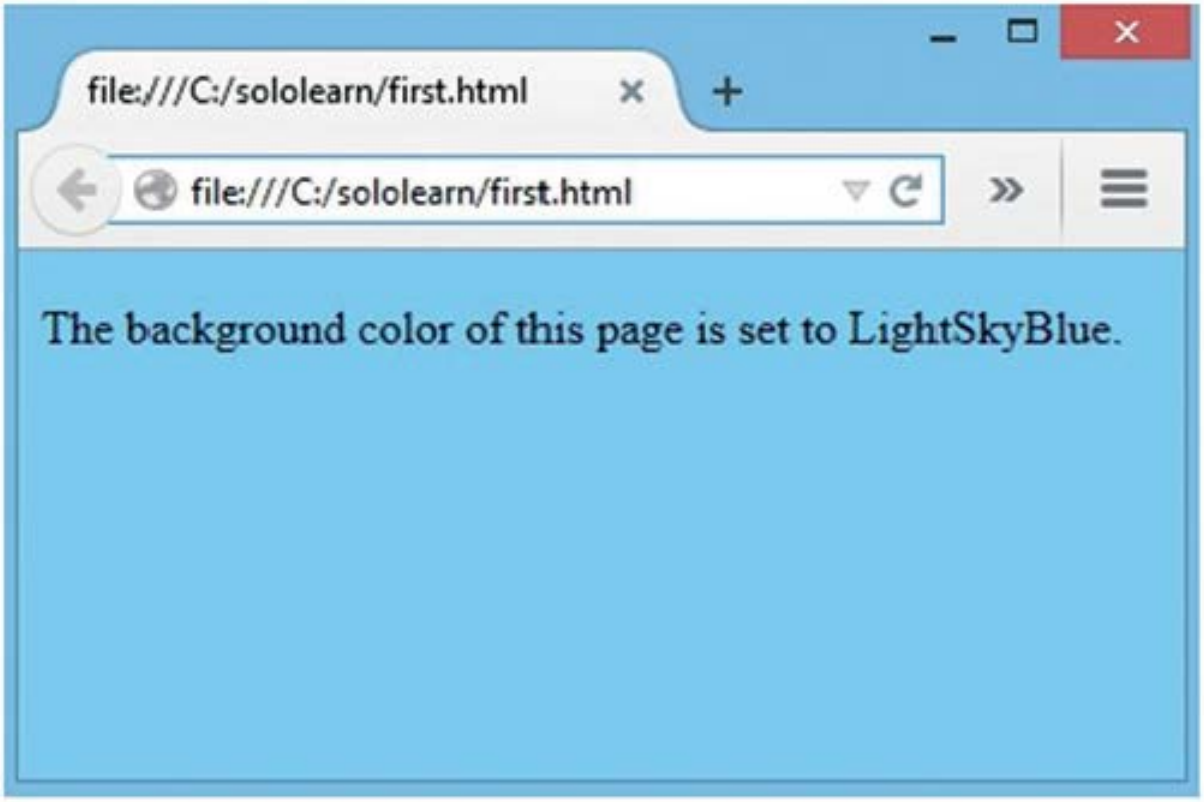
```
<p>The background color of this page is set to LightSkyBlue.</p>
```

The CSS:

```
body {
   background-color: #87CEFA;
}
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

134 COMMENTS

The Background Color Values

The color of the background can be defined using three different formats: a **color name**, **hexadecimal values**, and **RGB**.

In the example below, the body, h1, and p elements are assigned different background colors:

The HTML:

```
<h1>This is a heading</h1>
<p>This is a paragraph</p>
```

The CSS:

```
body {
  background-color: #C0C0C0;
}
h1 {
  background-color: rgb(135,206,235);
}
p {
  background-color: LightGreen;
}
```

Try It Yourself

Result:

The background-image Property

The **background-image** property sets one or several background images in an element. Let's set a background-image for the <body> element.

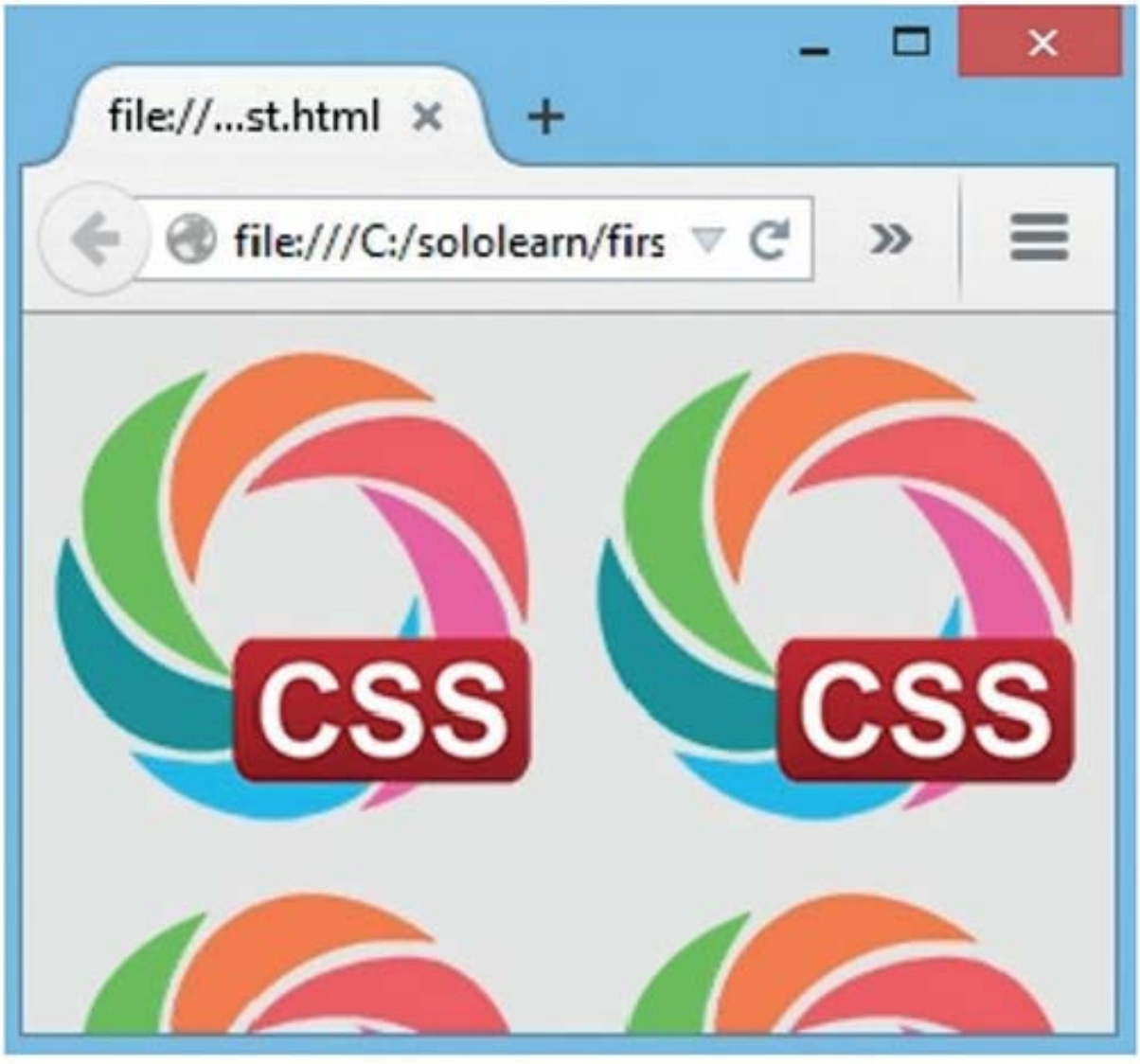
The CSS:

```
body {
  background-image: url("css_logo.png");
  background-color: #e9e9e9;
}
```

Try It Yourself

The **url** specifies the path to the image file. Both relative and absolute paths are supported.

Result:



The background-image Property

Background-image can be set not only for the whole page, but for individual elements, as well. Below we set a background image for the <p> element.

The HTML:

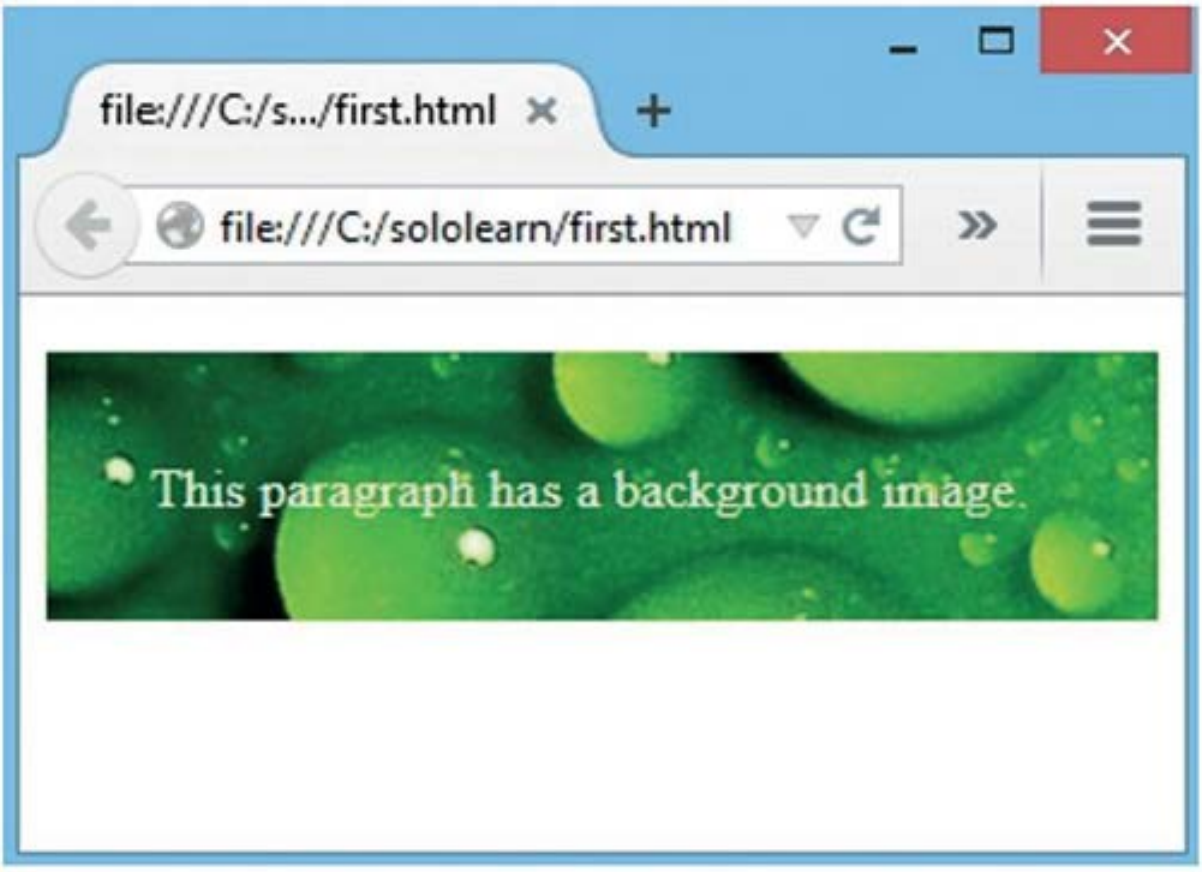
```
<p>This paragraph has a background image.</p>
```

The CSS:

```
p {
  padding: 30px;
  background-image: url("green_photo.jpg");
  color: white;
}
```

Try It Yourself

Result:



To specify more than one image, just separate the URLs with **commas**.

192 COMMENTS

The background-repeat Property

The background repeat property specifies how background images are repeated. A background image can be repeated along the **horizontal axis**, the **vertical axis**, **both axes**, or **not repeated at all**.

The **repeat-x** will repeat a background image only **horizontally**.
The CSS:

```
body {
  background-image: url("css_logo.png");
  background-repeat: repeat-x;
}
```

Try It Yourself

Result:

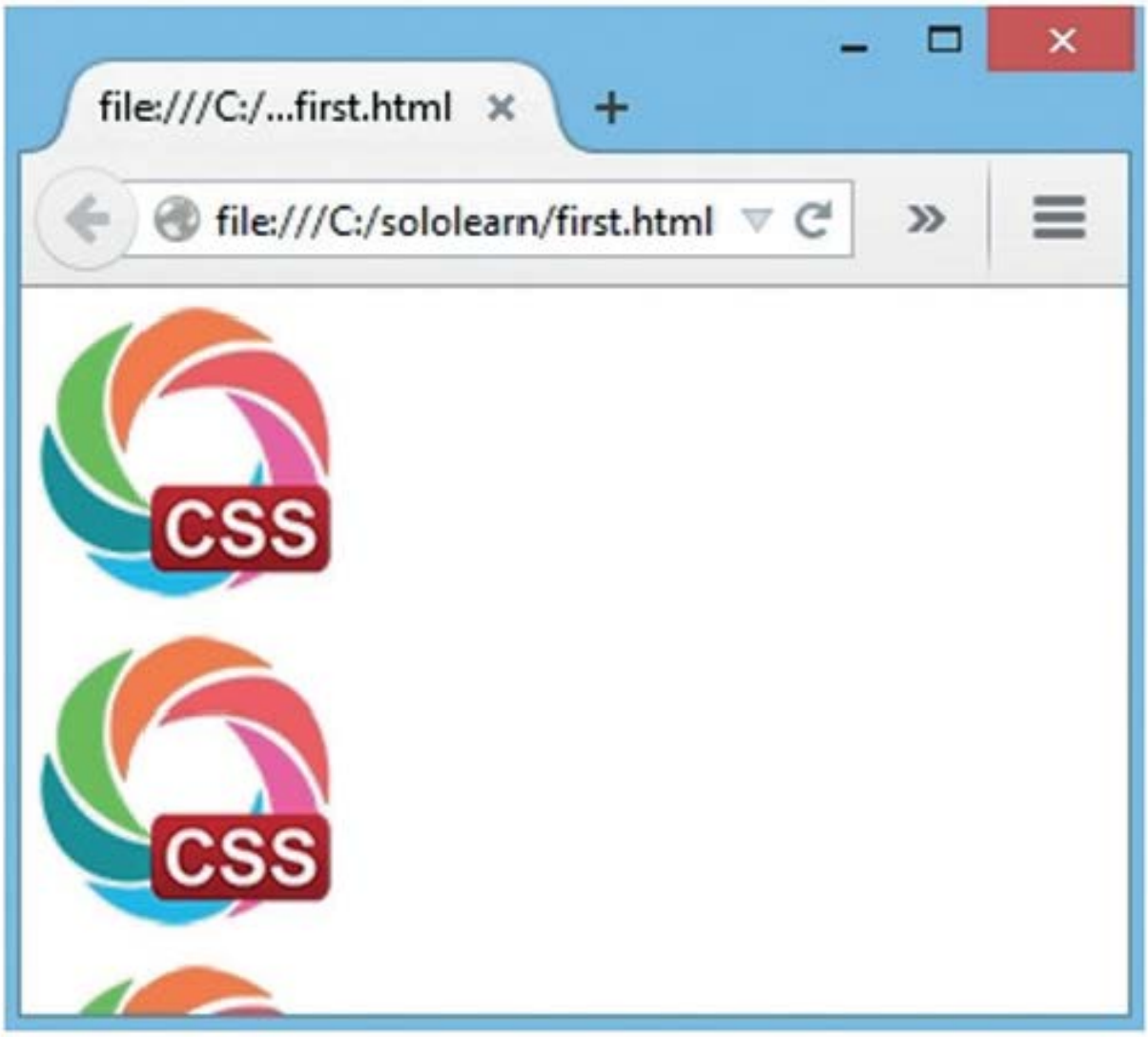
The **repeat-y** will repeat a background-image only **vertically**.

The CSS:

```
body {
  background-image: url("css_logo.png");
  background-repeat: repeat-y;
}
```

Try It Yourself

Result:



If you want the image to be shown only **once**, use the **no-repeat** value.

156 COMMENTS

Setting the Value to Inherit

When you set the background-repeat property to **inherit**, it will take the same specified value as the property for the element's parent.

For example, we set the body background repeat only horizontally. If we set some paragraph background-repeat values to be inherited, they will take the same property value as the body element.

The CSS:

```
body {
  background-image: url("css_logo.png");
  background-repeat: repeat-x;
}
p {
  background-image: url("css_logo.png");
  background-repeat: inherit;
  margin-top: 100px;
  padding: 40px;
}
```

Try It Yourself

Result:

The background-attachment Property

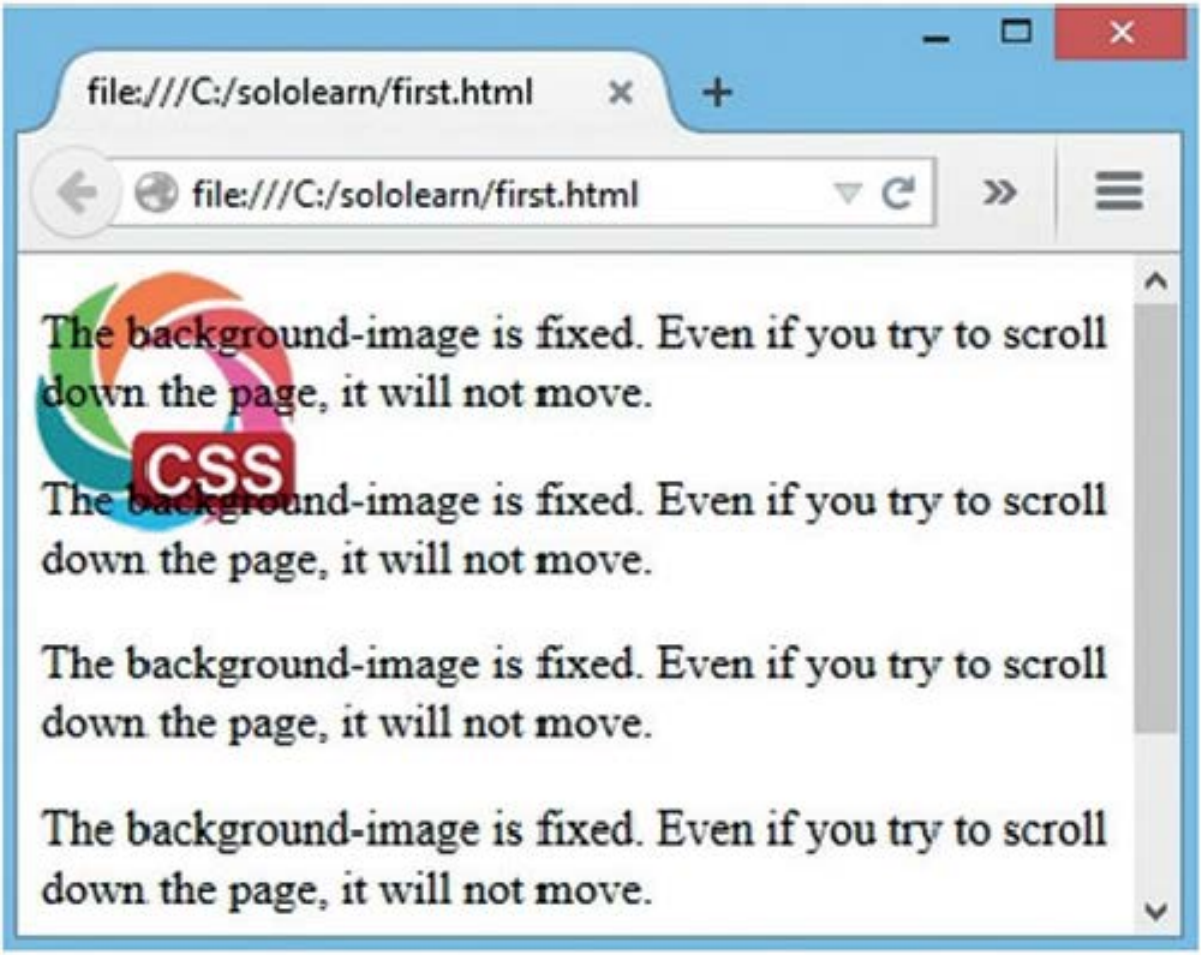
The background-attachment property sets whether a background image is fixed or scrolls with the rest of the page.
Even if an element has a scrolling mechanism, a "fixed" background doesn't move with the element.

The CSS:

```
body {
  background-image: url("css_logo.png");
  background-repeat: no-repeat;
  background-attachment: fixed;
}
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

188 COMMENTS

The background-attachment Values

You can also set the background-attachment to **inherit** or **scroll**.
When you set the background-attachment to **inherit**, it will inherit the value from its parent element.

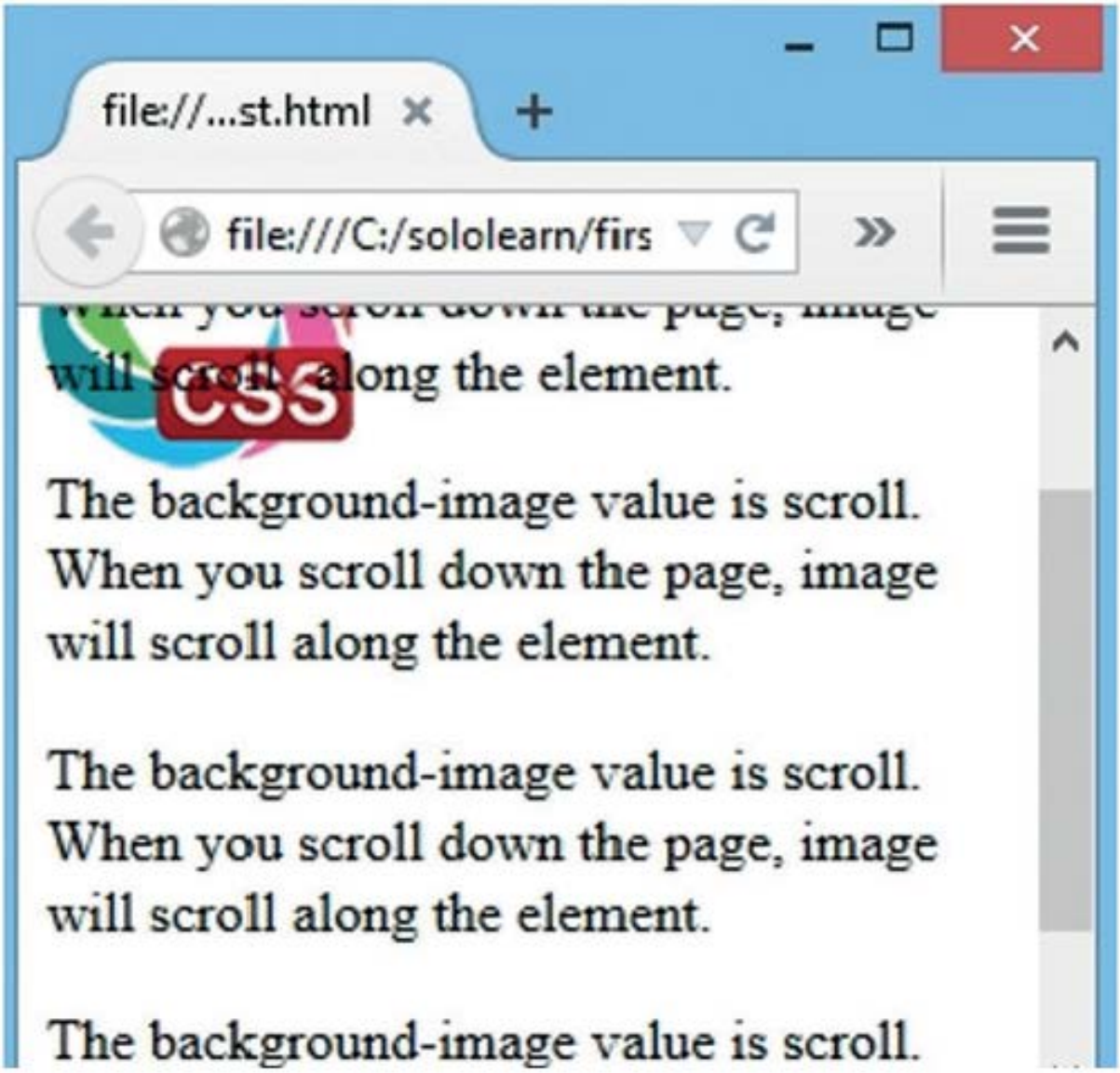
When you set the background-attachment to **scroll**, the background image will scroll with the rest of the content.

The CSS:

```
body {
  background-image: url("css_logo.png");
  background-repeat: no-repeat;
  background-attachment: scroll;
}
```

Try It Yourself

Result:



The list-style-type Property

The CSS list properties allow us to set different list item markers. In HTML, there are two types of lists:
unordered lists (<ul>) - the list items are marked with bullets
ordered lists (<ol>) - the list items are marked with numbers or letters
With CSS, lists can be styled further, and images can be used as the list item marker.
One of the ways is to use the **list-style-type** property, which can be set to **circle, square, decimal, disc, lower-alpha**, etc.

The HTML:

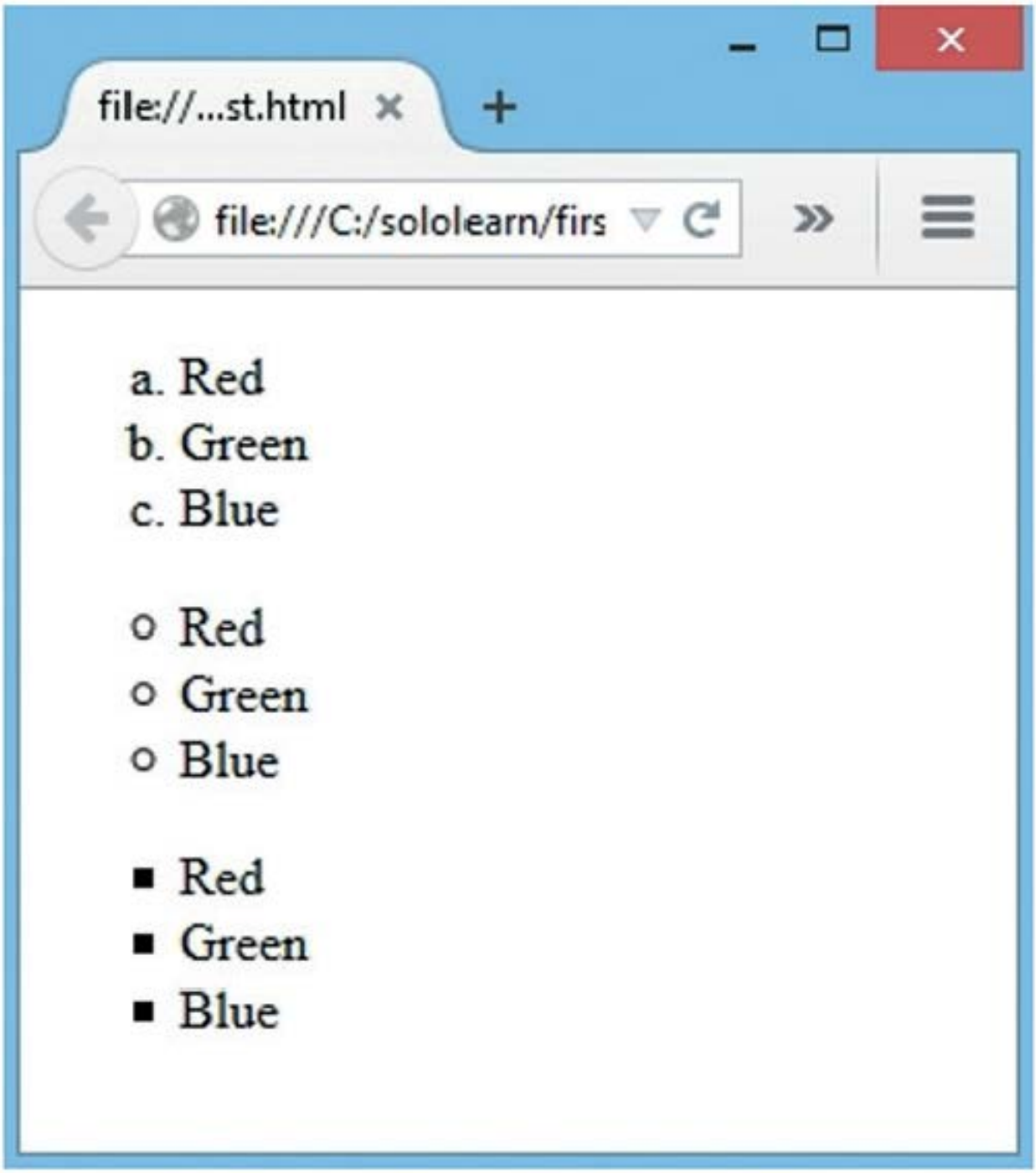
```
<ol class="lower-alpha">
  <li>Red</li>
  <li>Green</li>
  <li>Blue</li>
</ol>
<ul class="circle">
  <li>Red</li>
  <li>Green</li>
  <li>Blue</li>
</ul>
<ul class="square">
  <li>Red</li>
  <li>Green</li>
  <li>Blue</li>
</ul>
```

The CSS:

```
ol.lower-alpha {
  list-style-type: lower-alpha;
}
ul.circle {
  list-style-type: circle;
}
ul.square {
  list-style-type: square;
}
```

Try It Yourself

Result:



Some of the values are for unordered lists, and some for ordered lists.

169 COMMENTS

The List Image and Position

There are also other list properties, such as:
list-style-image - specifies an image to be used as the list item marker.
list-style-position - specifies the position of the marker box (inside, outside).

In the example below, we use an image as the list item marker, and specify the position to be inside of the content flow.

The HTML:

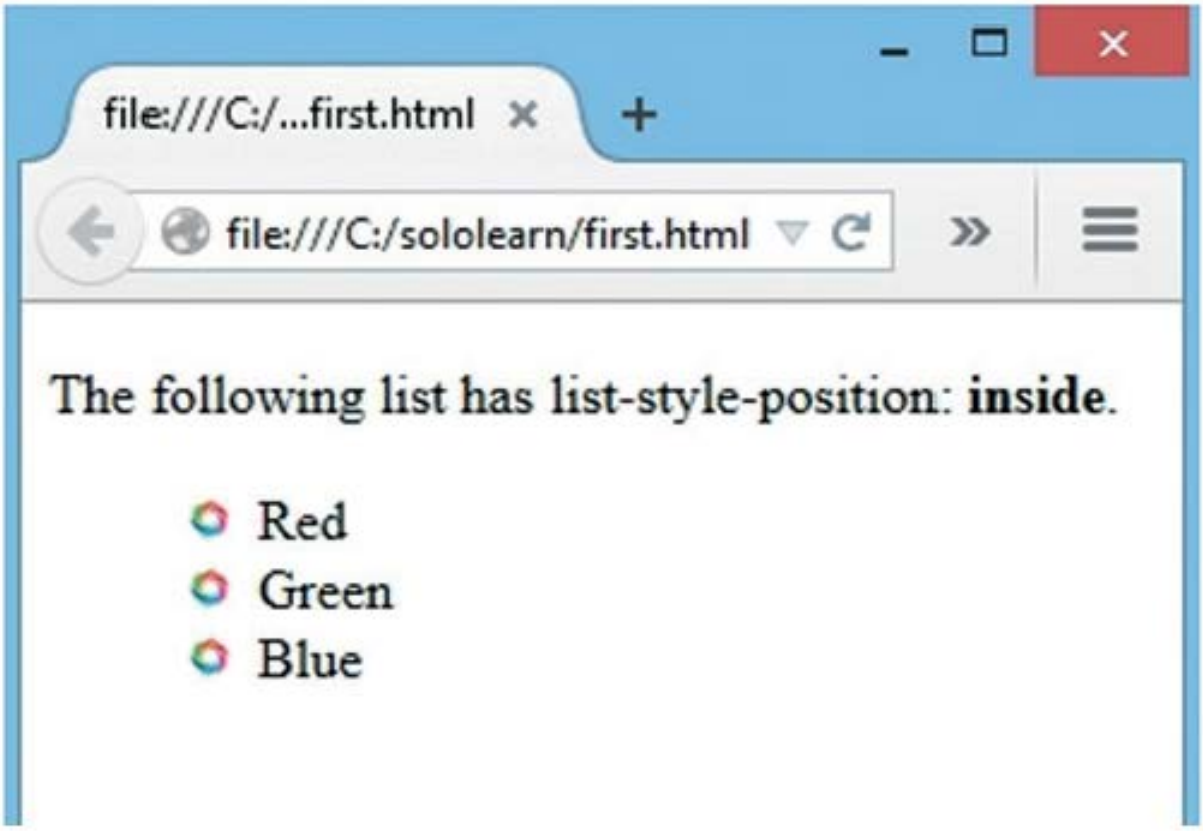
```
<p>The following list has list-style-position: <strong>inside</strong>.</p>
<ul>
  <li>Red</li>
  <li>Green</li>
  <li>Blue</li>
</ul>
```

The CSS:

```
ul {
  list-style-image: url("logo.jpg");
  list-style-position: inside;
}
```

Try It Yourself

Result:



The list-style Property

The **list-style** property is a shorthand property for setting list-style-type, list-style-image and list-style-position. It is used to set all of the list properties in one declaration:

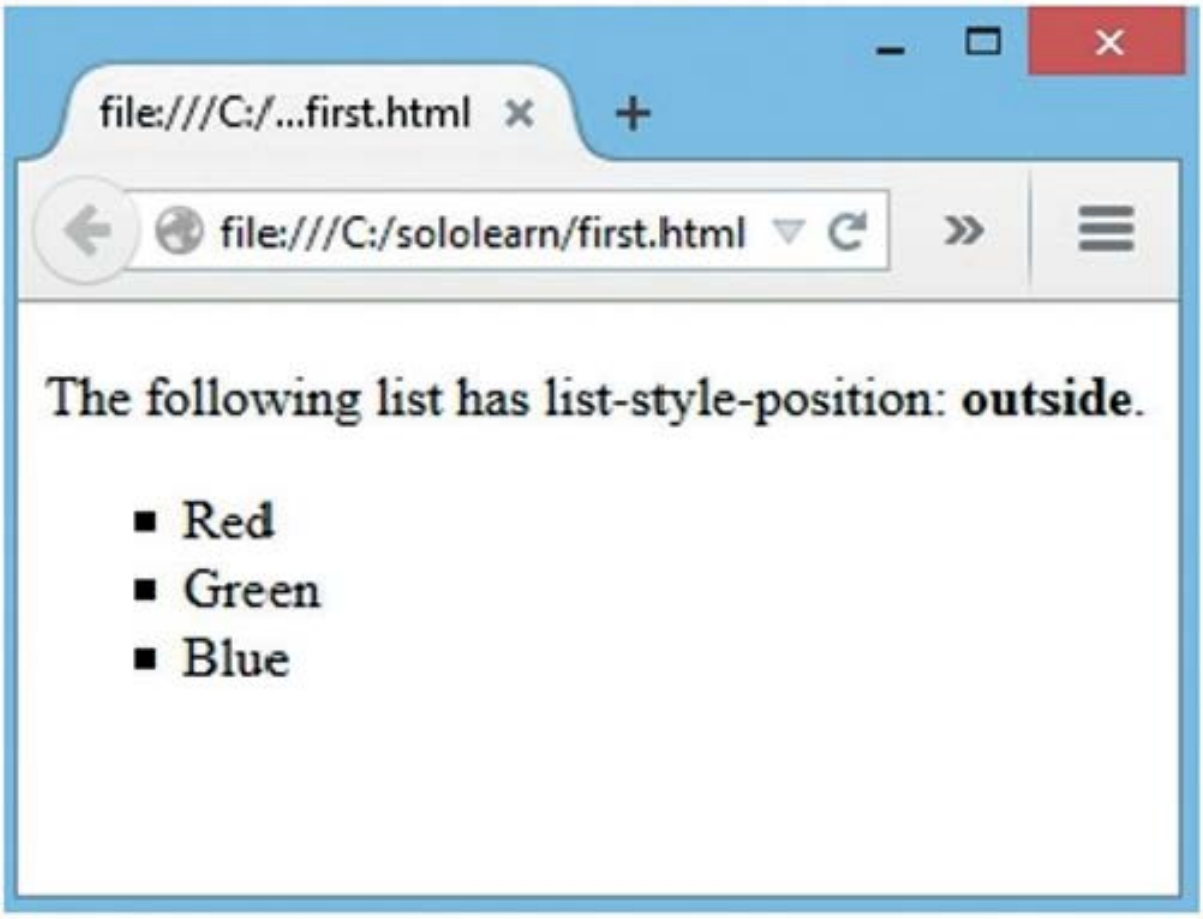
```
ul {
  list-style: square outside none;
}
```

This would be the same as the longhand version.

```
ul {
  list-style-type: square;
  list-style-position: outside;
  list-style-image: none;
}
```

Try It Yourself

Result:



If one of the property values are missing, the default value for the missing property will be inserted, if any.

The Table Properties

The look of an HTML table can be greatly improved with CSS.

The **border-collapse** property specifies whether the table borders are collapsed into a single border or separated as default. If the borders are separate, the **border-spacing** property can be used to change the spacing.

The HTML:

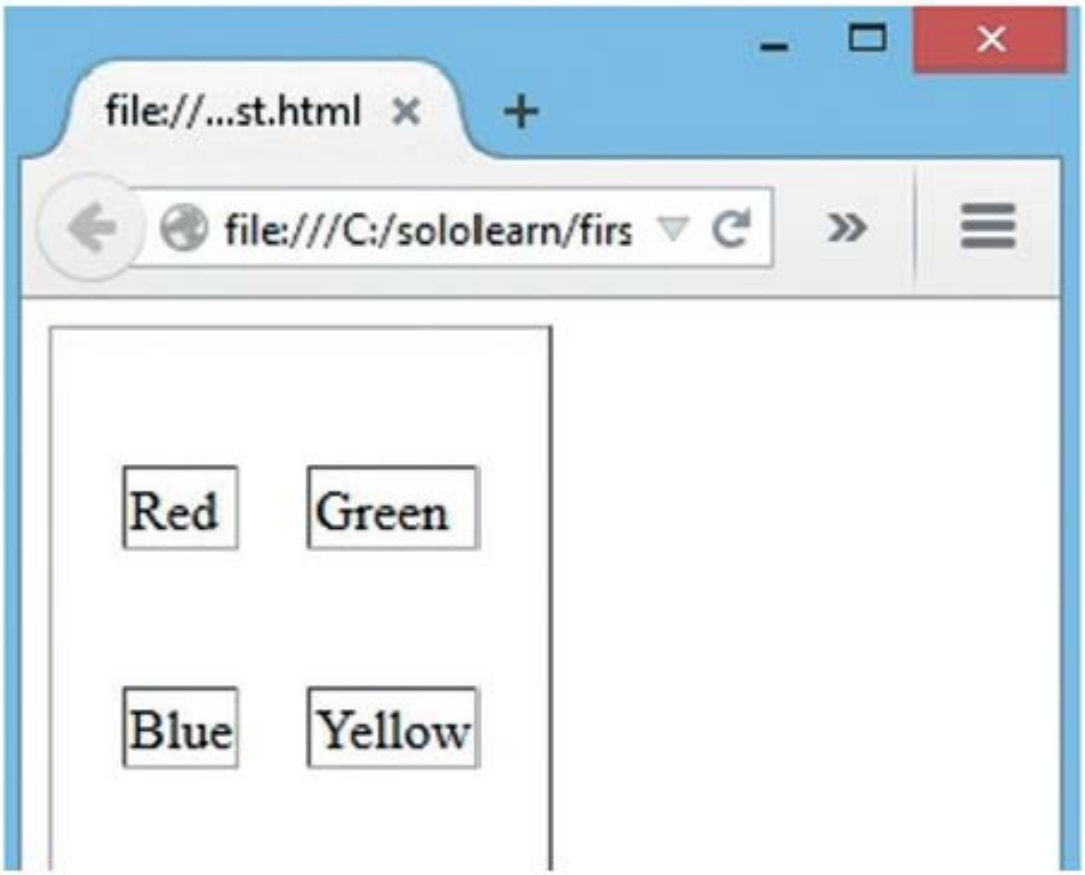
```
<table border="1">
  <tr>
    <td>Red</td>
    <td>Green</td>
  </tr>
  <tr>
    <td>Blue</td>
    <td>Yellow</td>
  </tr>
</table>
```

The CSS:

```
table {
  border-collapse: separate;
  border-spacing: 20px 40px;
}
```

Try It Yourself

Result:



The caption-side Property

The **caption-side** property specifies the position of a table caption. The values can be set as **top** or **bottom**.
In the example below, we specify the placement of a table caption to **top**.

The HTML:

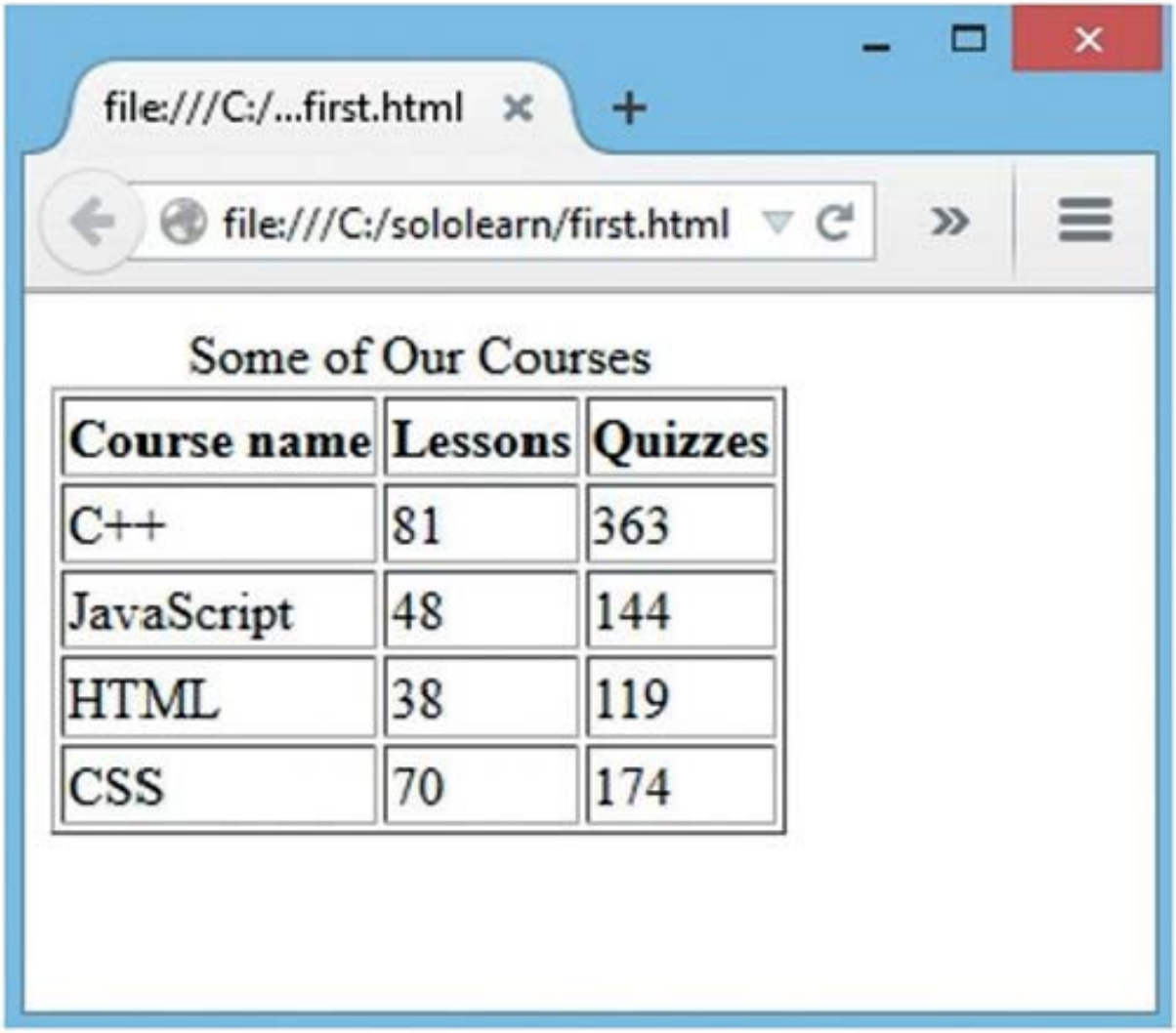
```
<table border="1">
<caption>Some of Our Courses</caption>
<tr>
 <th>Course name</th>
 <th>Lessons</th>
 <th>Quizzes</th>
</tr>
<tr>
 <td>C++</td>
 <td>81</td>
 <td>363</td>
</tr>
<tr>
 <td>JavaScript</td>
 <td>48</td>
 <td>144</td>
</tr>
<tr>
 <td>HTML</td>
 <td>38</td>
 <td>119</td>
</tr>
<tr>
 <td>CSS</td>
 <td>70</td>
 <td>174</td>
</tr>
</table>
```

The CSS:

```
caption {
  caption-side: top;
}
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

421 COMMENTS

The empty-cells Property

The **empty-cells** property specifies whether or not to display borders and background on empty cells in a table.
Possible values are:
show: the borders of an empty cell are rendered
hide: the borders of an empty cell are not drawn

Here is the empty-cells property that is used to hide borders of empty cells in the <table> element.

The HTML:

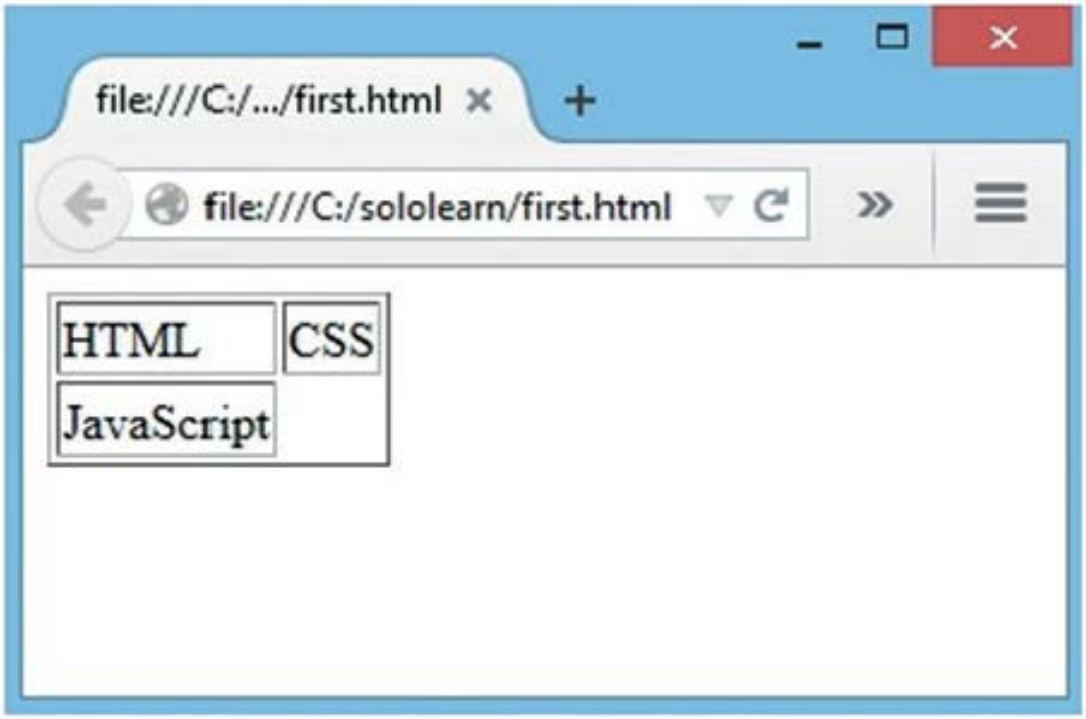
```
<table border="1">
  <tr>
    <td>HTML</td>
    <td>CSS</td>
  </tr>
  <tr>
    <td>JavaScript</td>
    <td></td>
  </tr>
</table>
```

The CSS:

```
table {
  border-collapse: separate;
  empty-cells: hide;
}
```

Try It Yourself

Result:



The table-layout Property

The **table-layout** specifies how the width of table columns is calculated. The possible values are:
auto - when column or cell width are not explicitly set, the column width will be in proportion to the amount of content in the cells that make up the column
fixed - when column or cell width are not explicitly set, the column width will not be affected by the amount of content in the cells that make up the column.

The table layout is set to **auto** by default.
The example below shows the difference between auto and fixed.

The HTML:

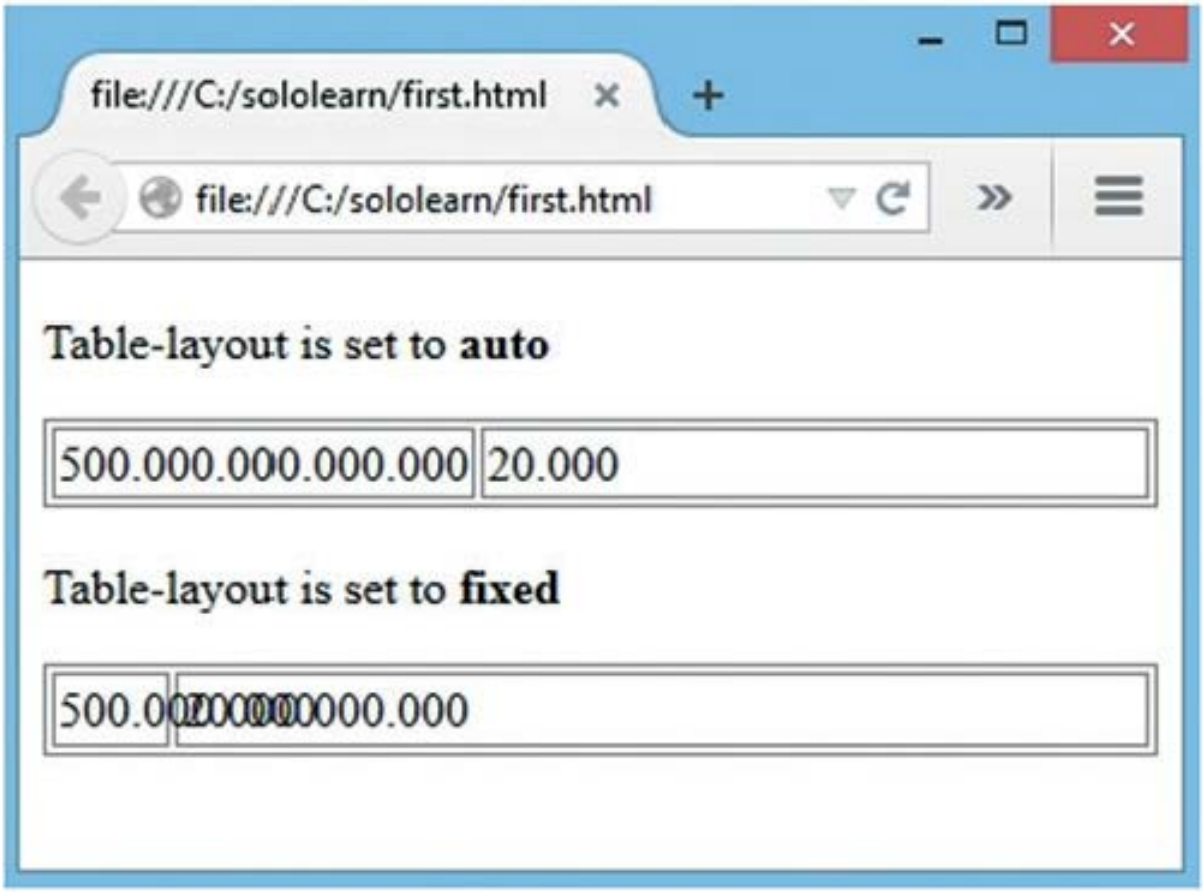
```
<p>Table-layout is set to <strong>auto</strong></p>
<table class="auto">
  <tr>
    <td width="10%">500.000.000.000.000</td>
    <td width="90%">20.000</td>
  </tr>
</table>

<p>Table-layout is set to <strong>fixed</strong></p>
<table class="fixed">
  <tr>
    <td width="10%">500.000.000.000.000</td>
    <td width="90%">20.000</td>
  </tr>
</table>
```

The CSS:

```
table {
  border-collapse: separate;
  width: 100%;
  border: 1px solid gray;
}
td {
  border: 1px solid gray;
}
table.auto {
  table-layout: auto;
}
table.fixed {
  table-layout: fixed;
}
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

91 COMMENTS

Setting Styles to Links

Links can be styled with any CSS property (e.g., color, font-family, background, etc.).
In addition, links can be styled differently, depending on what state they are in. The following pseudo selectors are available:
a:link - defines the style for normal unvisited links
a:visited - defines the style for visited links
a:active - a link becomes active once you click on it
a:hover - a link is hovered when the mouse is over it

The example below creates a link that will change the style when the mouse is moved over it.

The HTML:

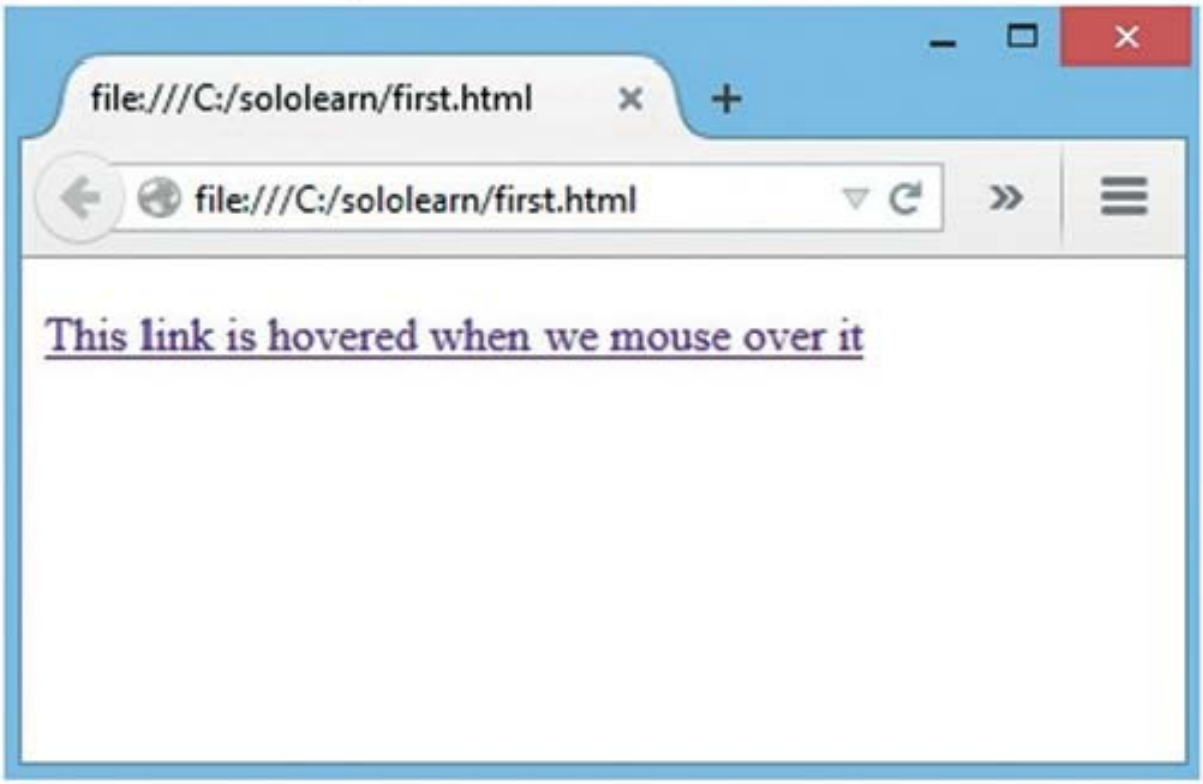
```
<p><a href="http://www.sololearn.com" target="_blank">
  This link is hovered when we mouse over it
</a></p>
```

The CSS:

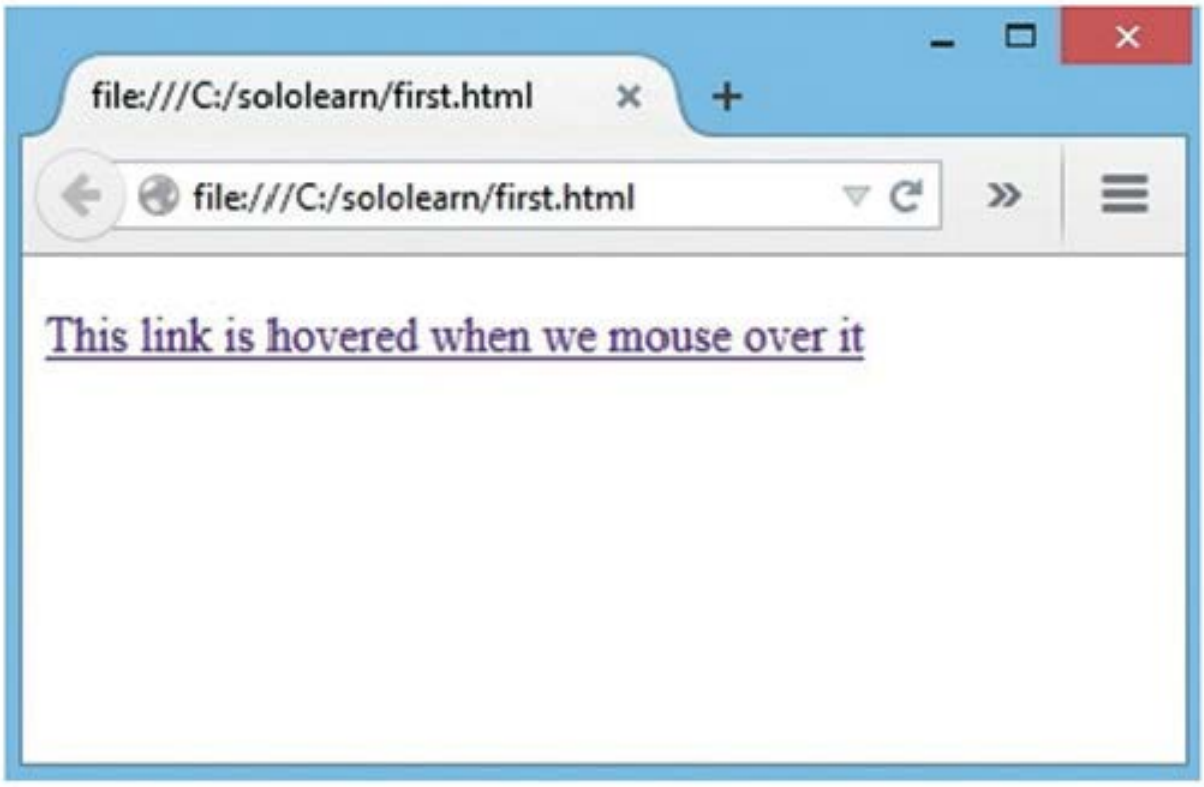
```
a:hover {
  color: red;
}
```

Try It Yourself

The link from the above example would look like this:



The link from the above example would look like this:



But when we mouse over it, it becomes red, as we defined in our stylesheet.

When setting the style for several link states, there are some order rules:
- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

187 COMMENTS

Links' Text Decoration

By default, text links are underlined by the browser.
One of the most common uses of CSS with links is to **remove the underline**. In the example below, the **text-decoration** property is used to remove the underline.

The HTML:

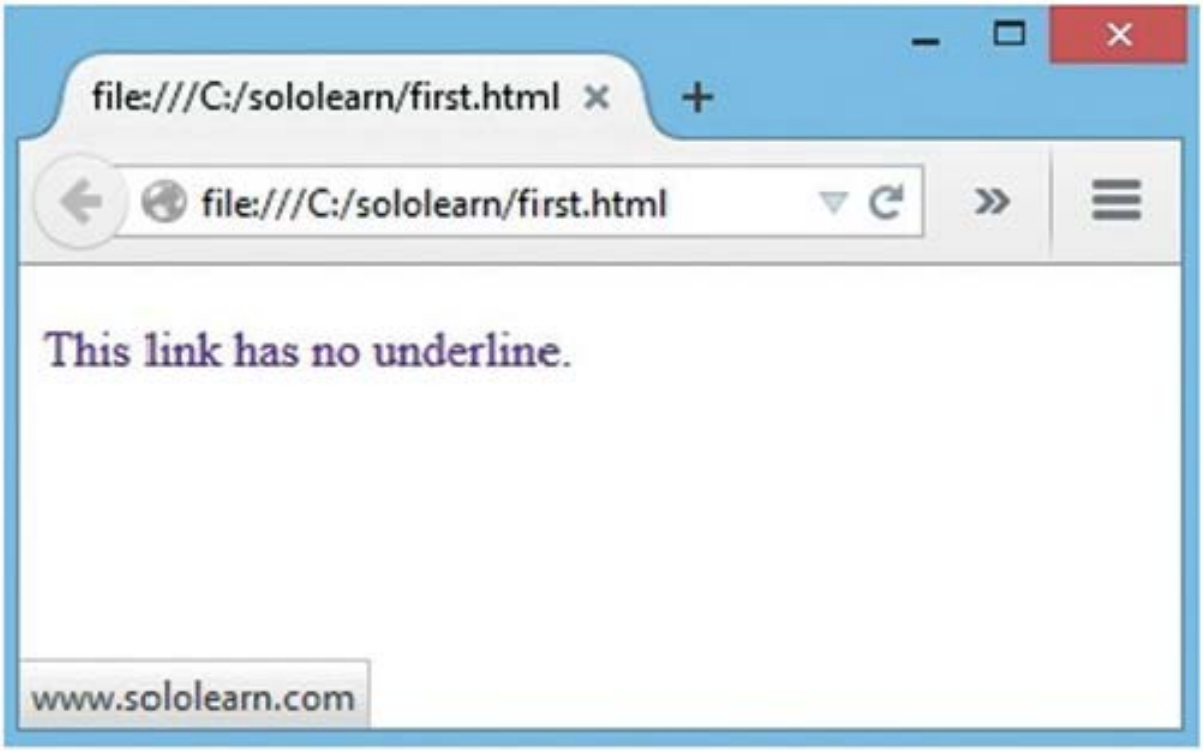
```
<p><a href="http://www.sololearn.com" target="_blank">
  This link has no underline.
</a></p>
```

The CSS:

```
a:link {
  text-decoration: none;
}
```

Try It Yourself

Result:



The following properties are used to control the look and feel of links:
border:none - removes border from images with links
outline:none - removes the dotted border on clicked lines in IE

77 COMMENTS

Setting the Mouse Cursor Style

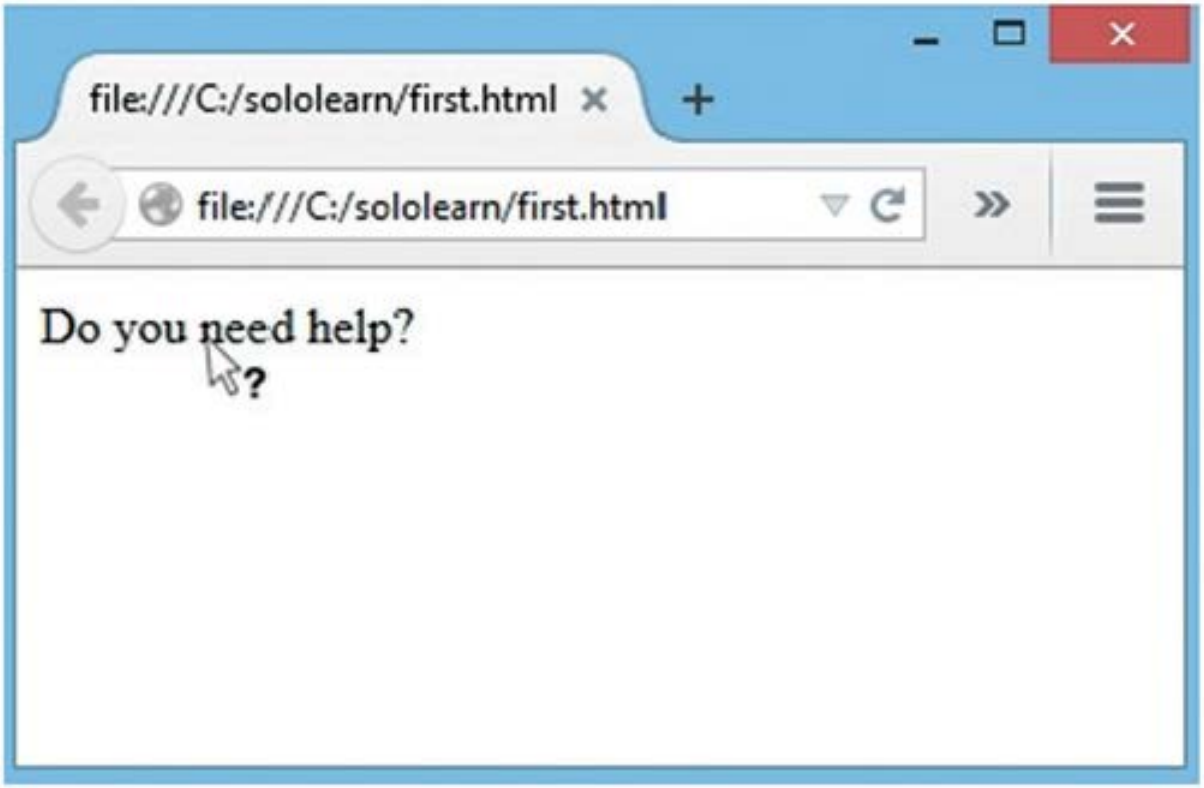
CSS allows you to set your desired cursor style when you mouse over an element. For example, you can change your cursor into a hand icon, help icon, and much more, rather than using the default pointer.

In the example below, the mouse pointer is set to a help icon when we mouse over the span element:

```
<span style="cursor:help;">
  Do you need help?
</span>
```

Try It Yourself

Result:

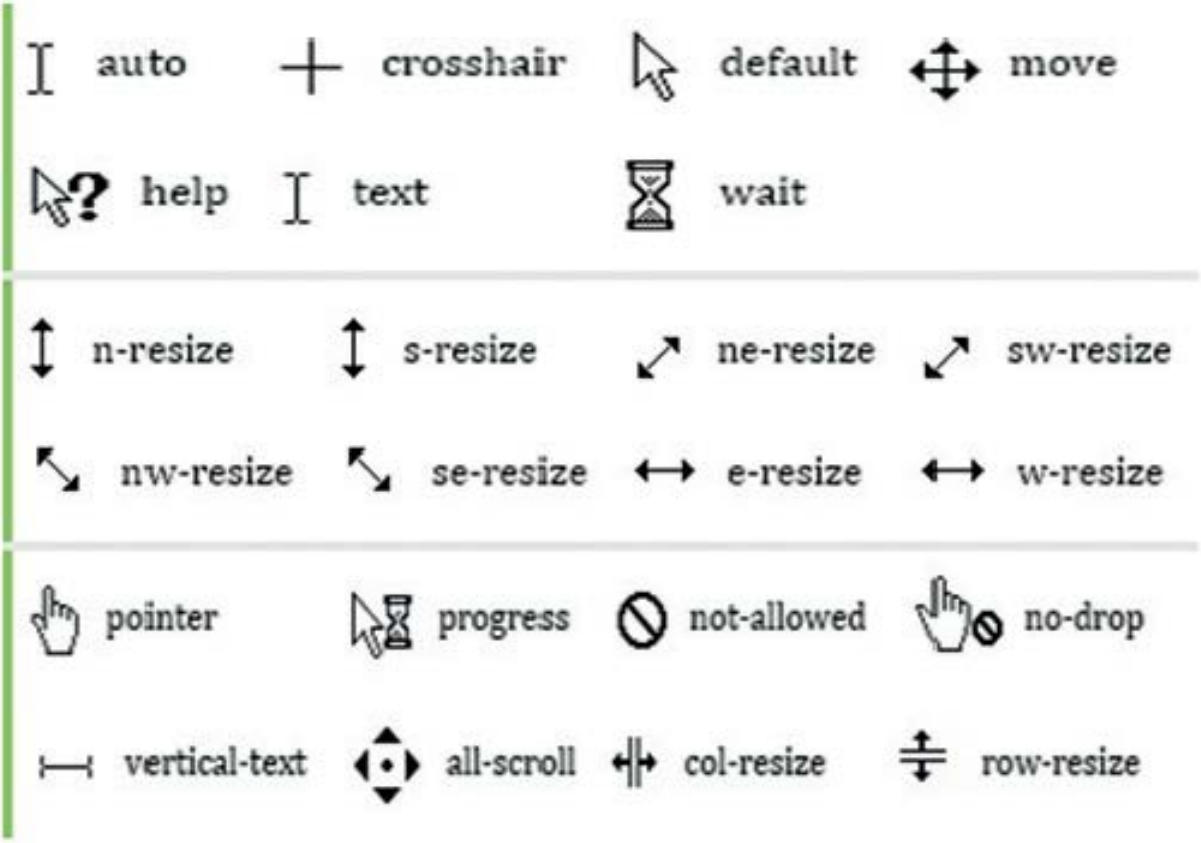


Tap **Try It Yourself** to play around with the code!

142 COMMENTS

The cursor Property Values

There are numerous other possible values for the **cursor** property, such as:
default - default cursor
crosshair - cursor displays as crosshair
pointer - cursor displays hand icon
The list of possible values is quite long. The image below demonstrates the various available styles:



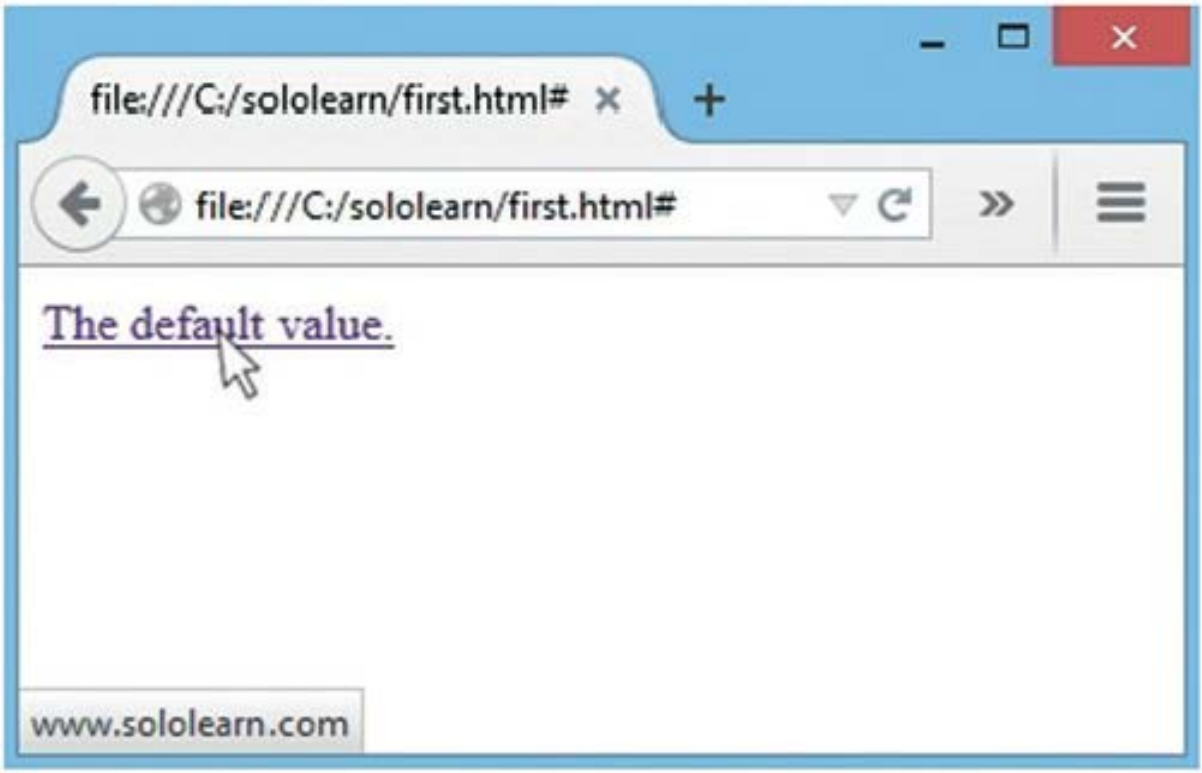
CSS allows you to set your desired cursor style when you mouse over an element.

111 COMMENTS

The default Value

Usually, the appearance of the mouse cursor is altered to provide a more interesting experience for website visitors. However, choosing the wrong cursor style can be misleading, as well.

For example, if the cursor value is set to **default**, users may not "see" the links.



Choose your mouse cursor styles carefully.

81 COMMENTS

Positioning and Layout

display: block

Every element on a web page is a rectangular box.The **display** property determines how that rectangular box behaves. A block element is an element that takes up the fullest width available, with line breaks before and after.
The style rules in the following example display the inline elements as block-level elements:

The HTML:

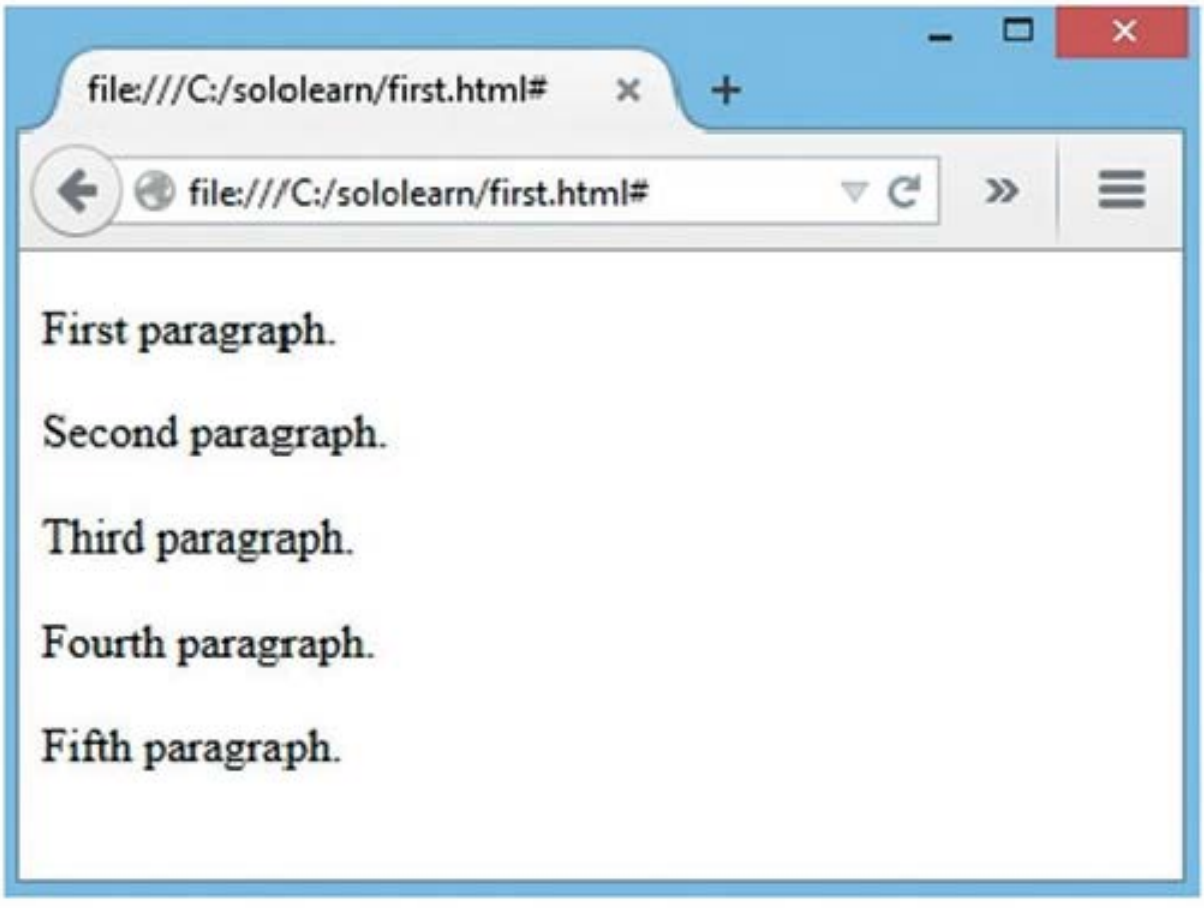
```
<span>First paragraph.</span>
<span>Second paragraph.</span>
<span>Third paragraph.</span>
<span>Fourth paragraph.</span>
<span>Fifth paragraph.</span>
```

The CSS:

```
span {
  display: block;
}
```

Try It Yourself

Result:



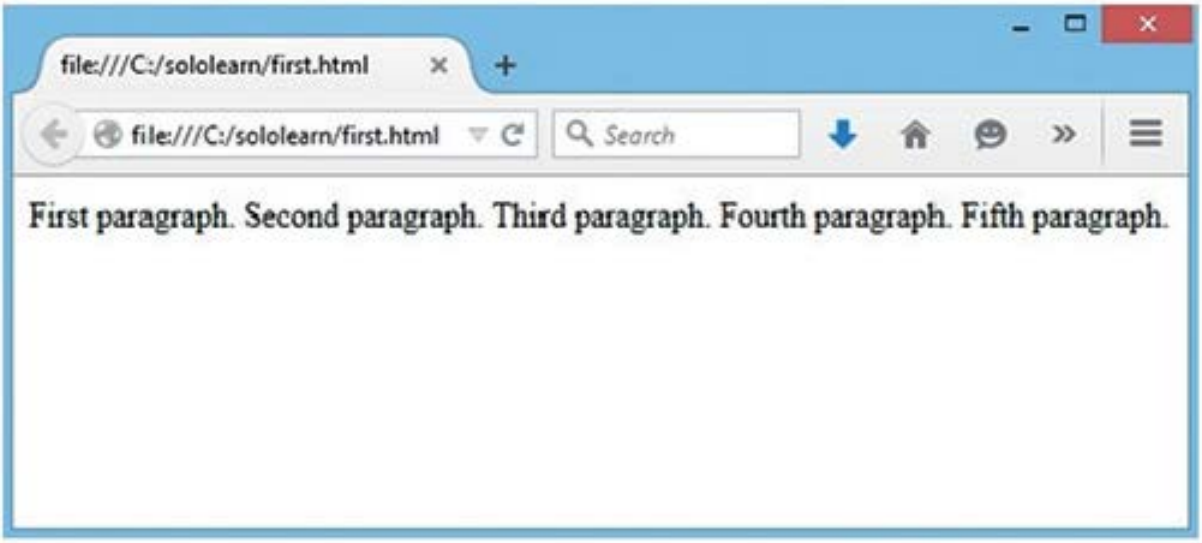
display: inline

An **inline** element only takes up as much width as necessary, and does not force line breaks.
The CSS:

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

Try It Yourself

The code above will produce the following result:



Setting the display property of an element only changes how the element is displayed, not what kind of element it is. So, an inline element with **display:block** is not allowed to have other block elements inside it.

105 COMMENTS

display:none

display:**none** hides an element, so it does not take up any space. The element will be hidden, and the page will be displayed as if the element is not there.

The HTML:

```
<h1>This text will not display, as we set the value to none.</h1>
<p>Headline of this paragraph is not displayed, as we set the value to none.</p>
```

The CSS:

```
h1 {
  display: none;
}
```

Try It Yourself

Result:

There are plenty of other display values, such as **list-item, table, table-cell, table-column, grid**, etc. Just play with values to see the difference.

192 COMMENTS

The visibility Property

The **visibility** property specifies whether an element is visible or hidden. The most common values are **visible** and **hidden**.

Hiding an element can be done by setting the display property to "none" or the visibility property to "hidden". However, notice that these two methods produce different results:
visibility:**hidden** hides an element, but it will still take up the same space as before. The element will be hidden, but it will still affect the layout:
Here is an example:

The HTML:

```
<h1>This is a heading</h1>
<div class="hidden">
  This text will not display in browser.
</div>
<p>
  The space above this paragraph is empty because
  the visibility of the div element is set to hidden.
</p>
```

The CSS:

```
div.hidden {
  visibility: hidden;
}
```

Try It Yourself

Result:

display:**none** hides an element, without holding a place for that element.

Changing visibility: **hidden;** to display: **none;** will produce the following result:

```
div.hidden {
  display: none;
}
```

Try It Yourself

Result:

Tap **Try It Yourself** to play around with the code!

105 COMMENTS

Q&A

Positioning Elements

The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element's content is too big.

Elements can be positioned using the top, bottom, left, and right properties. However, these properties will not work unless the **position** property is set first. They also work differently depending on the positioning method.

Static Positioning

HTML elements are positioned **static** by default. A static positioned element is always positioned according to the normal flow of the page.

The HTML:

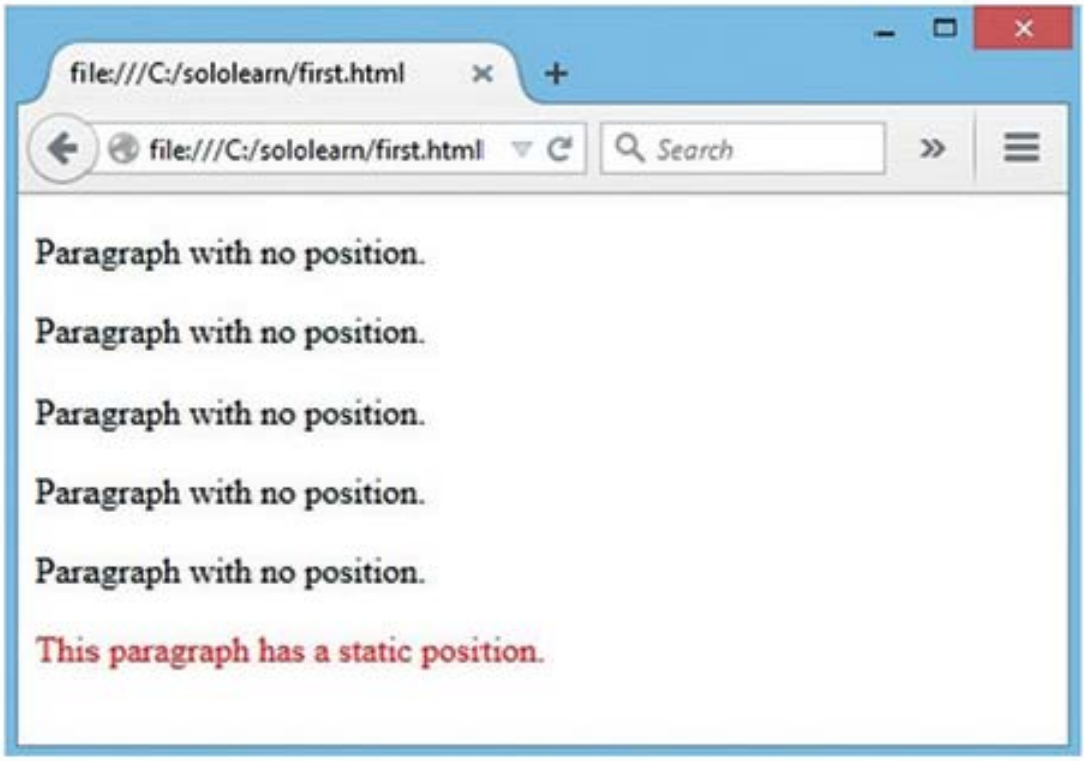
```
<p>Paragraph with no position.</p>
<p>Paragraph with no position.</p>
<p>Paragraph with no position.</p>
<p>Paragraph with no position.</p>
<p>Paragraph with no position.</p>
<p class="position_static">This paragraph has a static position.</p>
```

The CSS:

```
p.position_static {
  position:static;
  top: 30px;
  right: 5px;
  color: red;
}
```

Try It Yourself

Result:



Fixed Positioning

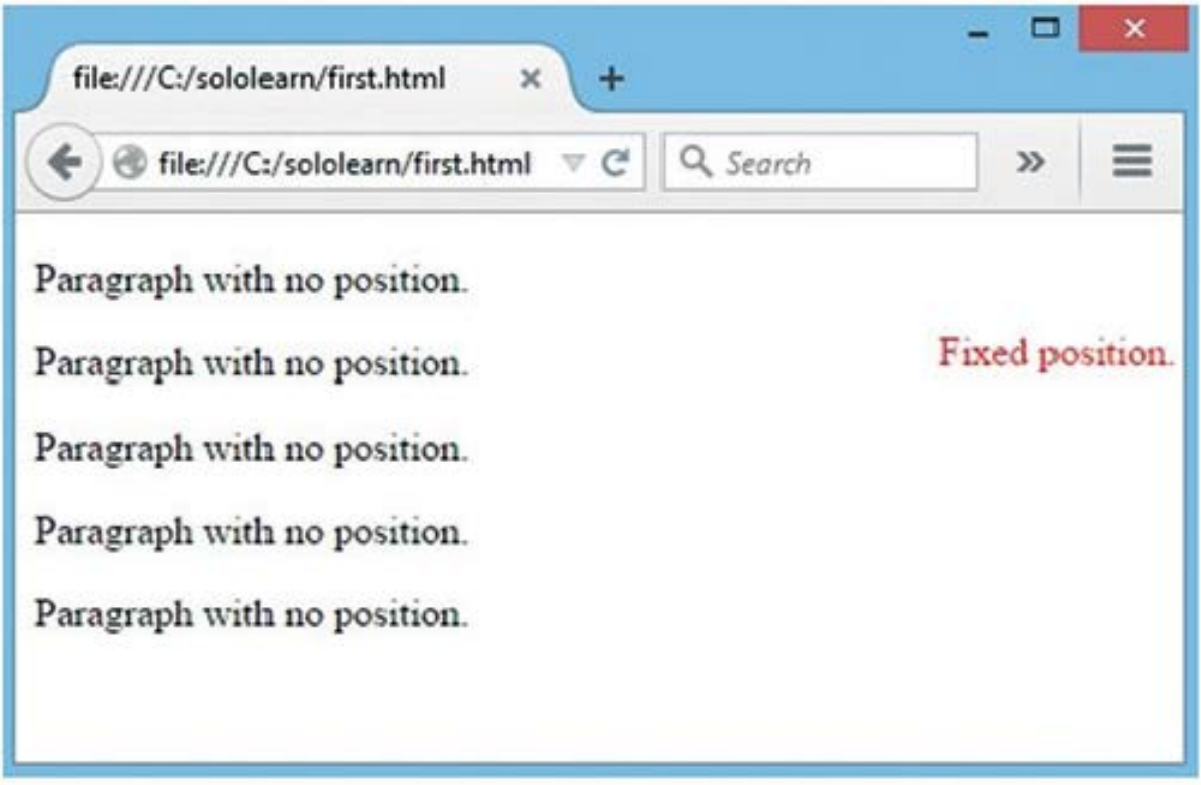
An element with a fixed position is positioned relative to the browser window, and will not move even if the window is scrolled.
The position can be specified using one or more of the properties **top**, **right**, **bottom**, and **left**.
In the example below, the paragraph is fixed to 30px from the top and 5px from the right.

The CSS:

```
p.position_fixed {
  position: fixed;
  top: 30px;
  right: 5px;
  color: red;
}
```

Try It Yourself

Result:



Fixed positioned elements are removed from the normal flow. The document and other elements behave like the fixed positioned element does not exist.
Fixed positioned elements can overlap other elements.

166 COMMENTS

Relative Positioning

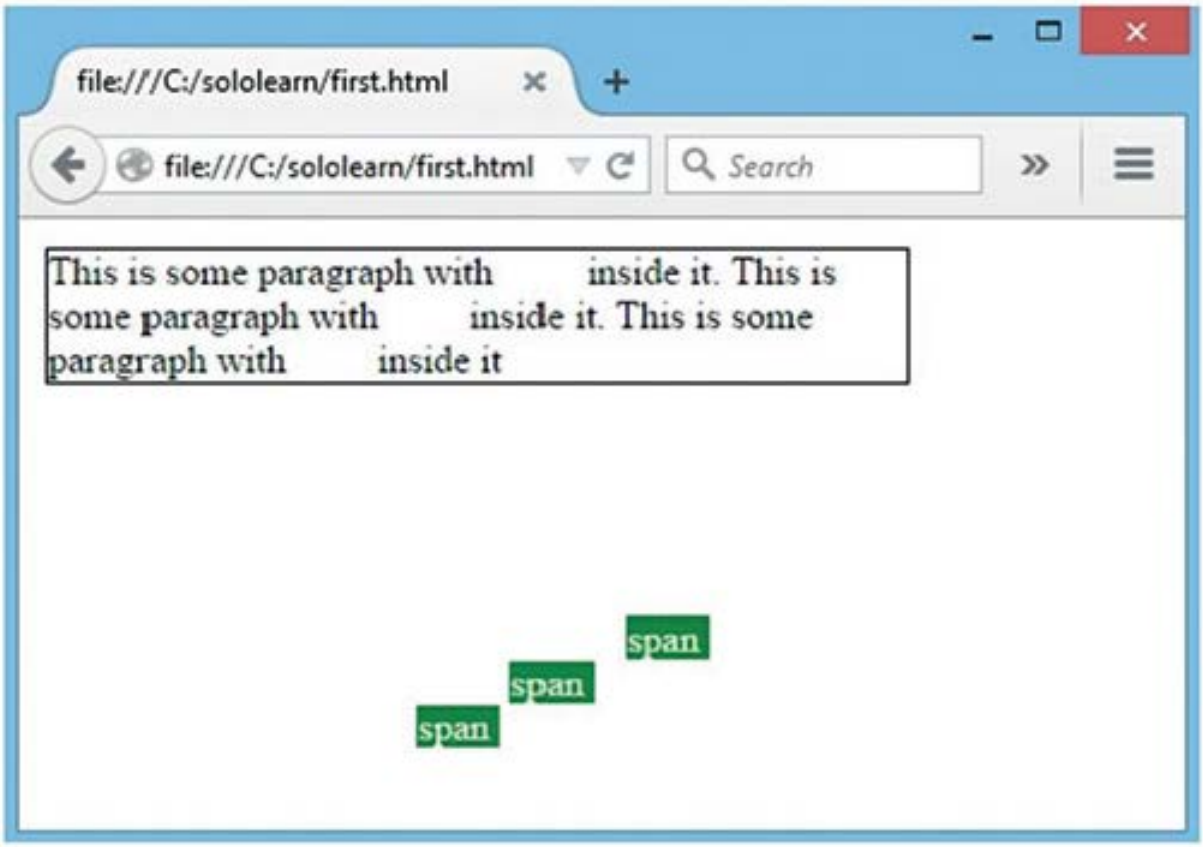
A relative positioned element is positioned relative to its normal position.
The properties **top**, **right**, **bottom**, and **left** can be used to specify how the rendered box will be shifted.

The CSS:

```
p {
  width: 350px;
  border: 1px black solid;
  position: fixed;
}
span {
  background: green;
  color:white;
  position: relative;
  top: 150px;
  left: 50px;
}
```

Try It Yourself

Result:



The content of relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.

> This value cannot be used for **table cells, columns, column groups, rows, row groups,** or **captions.**

Absolute Positioning

An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>.

Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist.

> Absolutely positioned elements can overlap other elements.

196 COMMENTS

Floating

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.
Float is often used with images, but it is also useful when working with layouts.
The values for the float property are **left**, **right**, and **none**.
Left and right float elements in those directions, respectively. **none** (the default) ensures that the element will not float.
Below is an example of an image that is floated to the right.

The HTML:

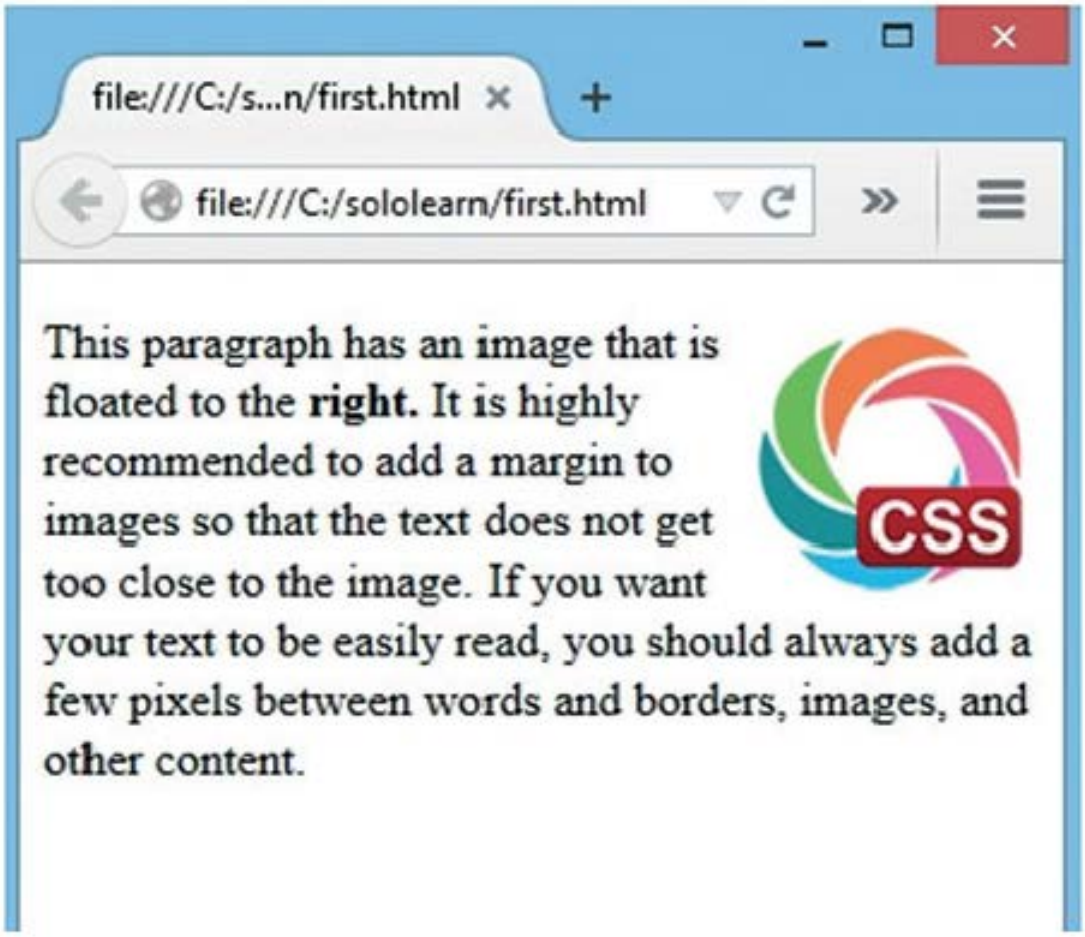
```
<p><img src="css_logo.png" />
This paragraph has an image that is floated to the <strong>right.</strong>
It is highly recommended to add a margin to images so that the text does
not get too close to the image. If you want your text to be easily read, you
should always add a few pixels between words and borders, images,
and other content.
</p>
```

The CSS:

```
img {
  float: right;
}
```

Try It Yourself

Result:



Elements Next to Each Other

If you place several floating elements one after the other, they will float next to each other if there is enough room.
As an example, in a print layout, images may be set into the page such that text wraps around them as needed.

The CSS:

```
img {
  float: left;
  width: 120px;
  margin-right: 10px;
}
p {
  width: 120px;
  float: left;
}
```

Try It Yourself

Result:

Elements are floated horizontally, meaning that an element can only be floated left or right, not up or down.

141 COMMENTS

Clearing the Float

Elements that come after the floating element will flow around it. To avoid this, use the clear property.
The **clear** property specifies the sides of an element where other floating elements are not allowed to be.

In the example below, if we set the float property to the div, only the paragraph that comes after the div will be wrapped around the image.
The HTML:

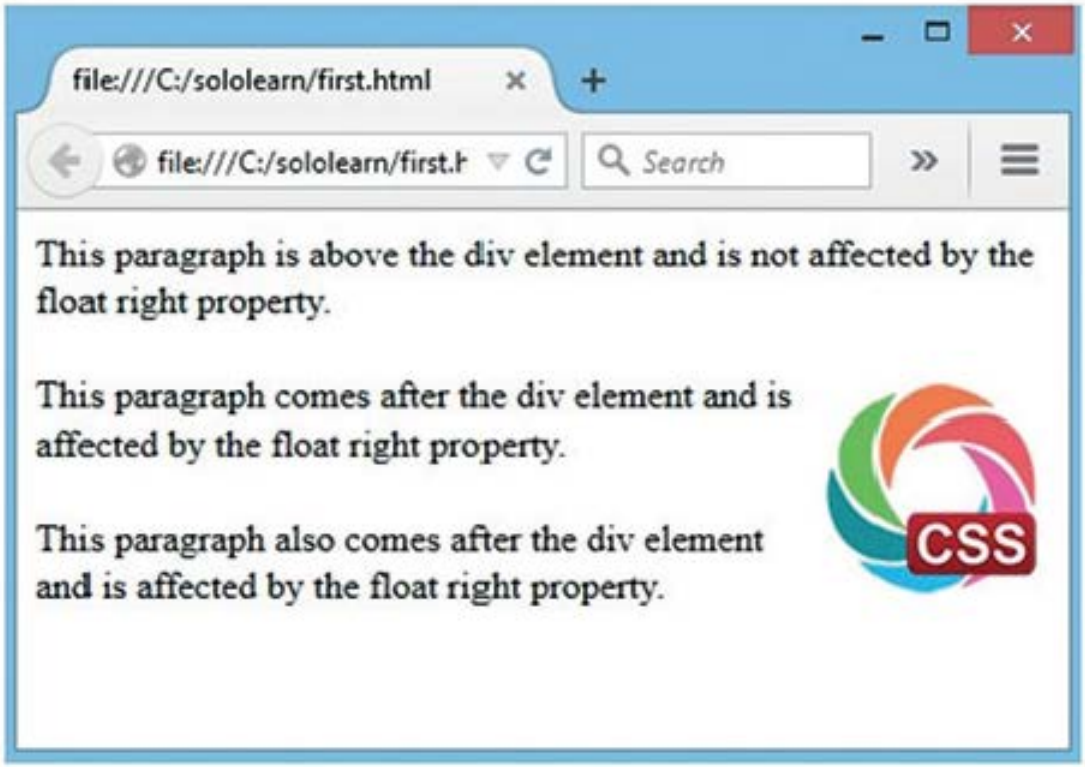
```
This paragraph is above the div element
and is not affected by the float right property.
<br /><br />
<div class="floating">
  <img src="css_logo.png" />
</div>
This paragraph comes after the div element
and is affected by the float right property.
<br /><br />
This paragraph also comes after the div element
and is affected by the float right property.
<br /> <br />
```

The CSS:

```
.floating {
  float: right;
}
```

Try It Yourself

Result:



Using clear

Use the values **right**, **left**, and **both** to specify the sides of an element where other floating elements are not allowed to be.

> The default value is **none**, which allows floating elements on both sides.

63 COMMENTS

Clearing Floats

both is used to clear floats coming from either direction.

The HTML:

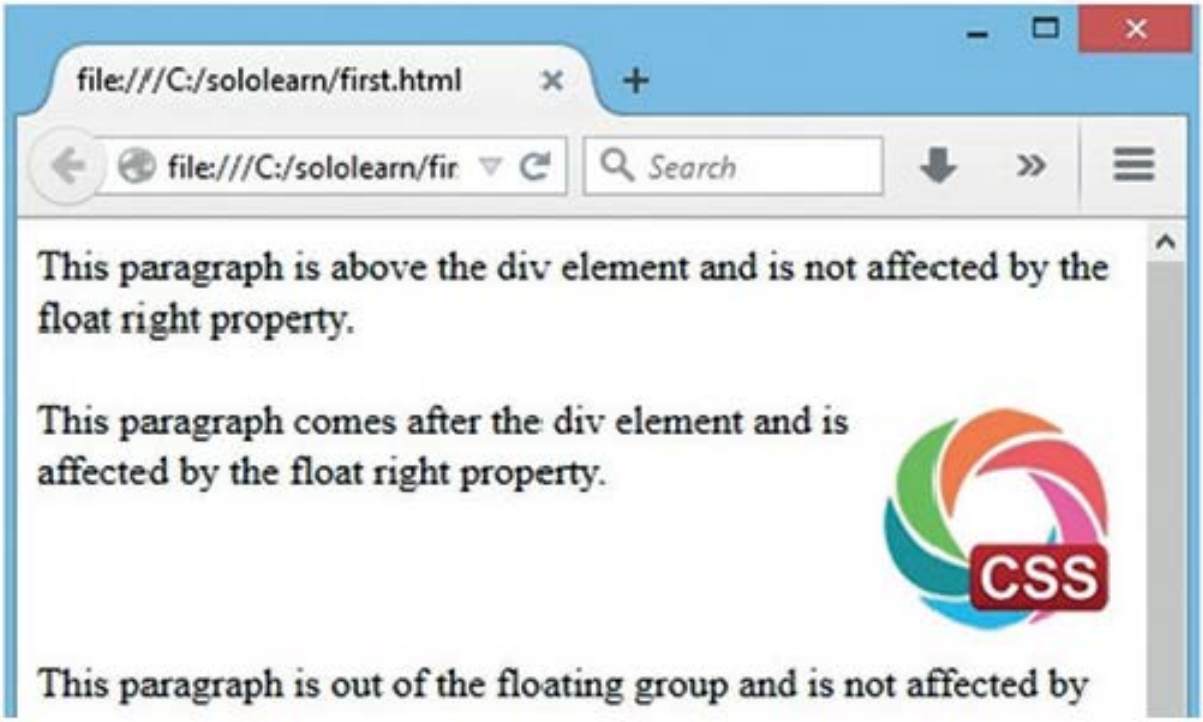
```
This paragraph is above the div element
and is not affected by the float right property.
<br/><br/>
<div class="floating">
  <img src="css_logo.png" />
</div>
This paragraph comes after the div element
and is affected by the float right property.
<br/><br class="clearing"/>
This paragraph is out of the floating group
and is not affected by the float right property.
<br/> <br/>
```

The CSS:

```
.floating {
  float: right;
}
.clearing {
  clear: both;
}
```

Try It Yourself

Result:



The overflow Property

As discussed earlier, every element on the page is a **box**. If the height of the box is not set, the height of that box will grow as large as necessary to accommodate the content.

The HTML:

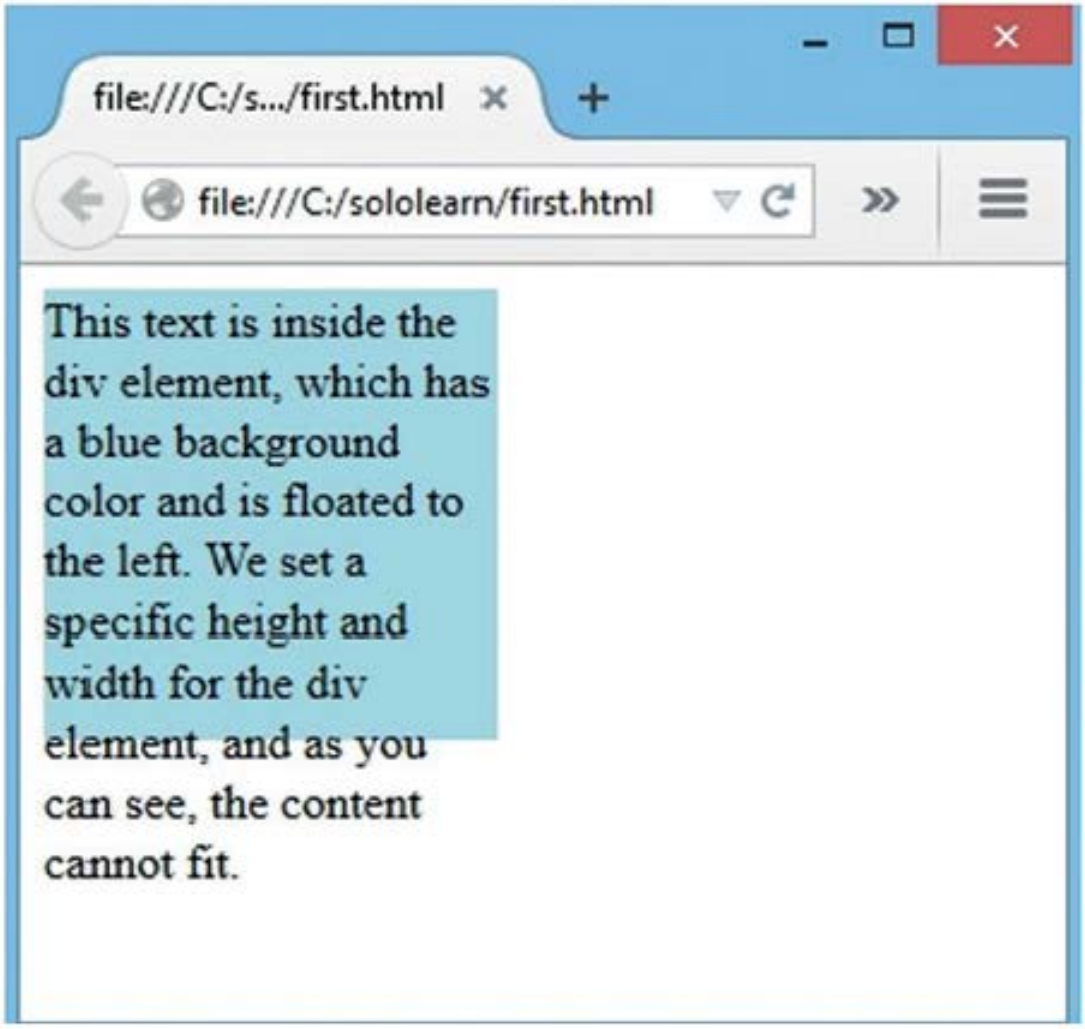
```
<div>
This text is inside the div element, which has a blue
background color and is floated to the left. We set a specific
height and width for the div element, and as you can see,
the content cannot fit.
</div>
```

The CSS:

```
div {
  width: 150px;
  height: 150px;
  background-color: LightBlue;
  float: left;
}
```

Try It Yourself

Result:



The overflow Property Values

There are four values for the overflow property: **visible** (the default value), **scroll**, **hidden**, and **auto**.

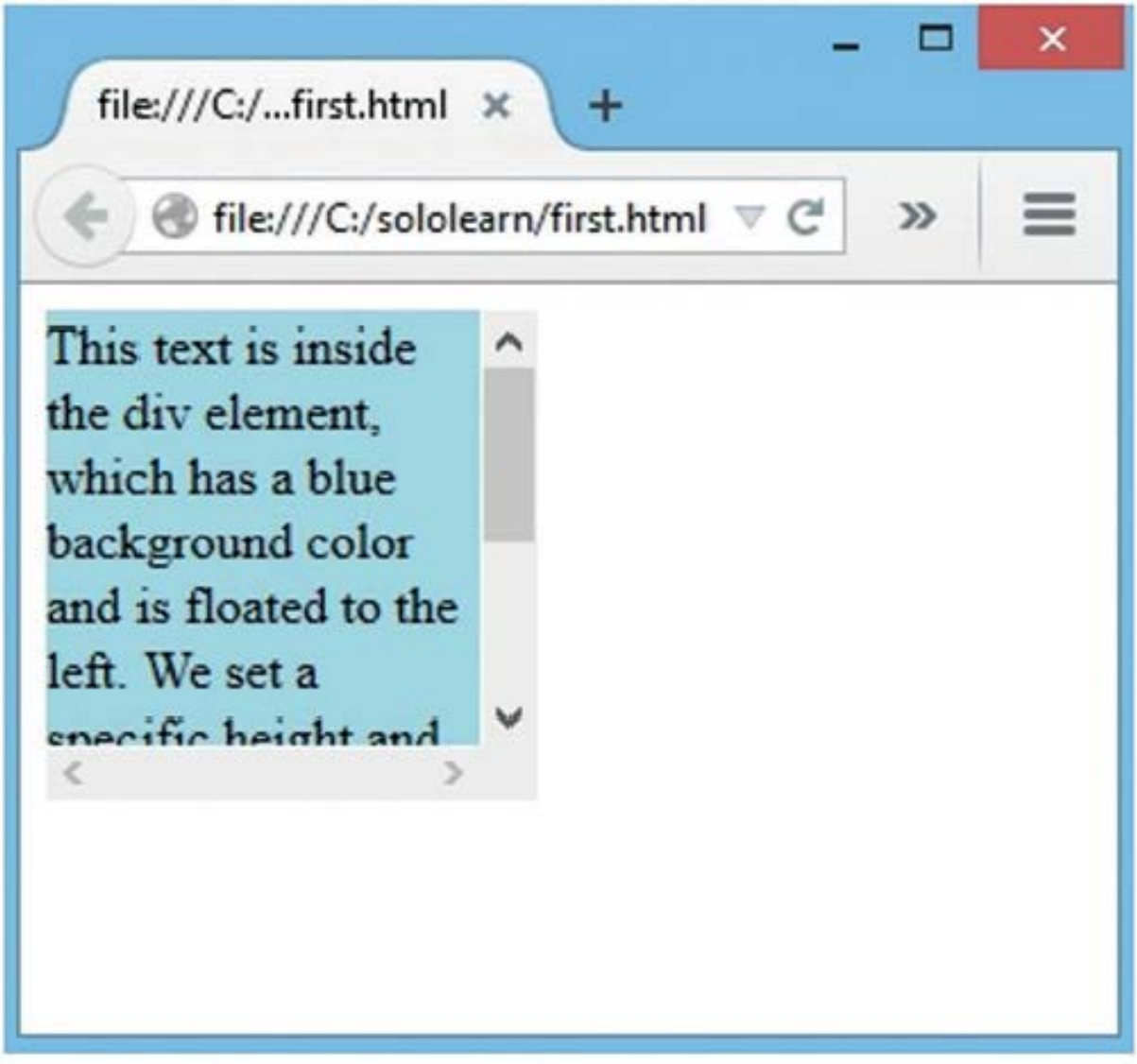
The value **scroll** results in clipped overflow, but a scrollbar is added, so the rest of the content may be seen.

The CSS:

```
div {
  width: 150px;
  height: 150px;
  background-color: LightBlue;
  float: left;
  overflow: scroll;
}
```

Try It Yourself

The code above will produce both **horizontal** and **vertical** scrollbars:



auto and hidden

auto - If overflow is clipped, a scroll-bar should be added to make it possible to see the rest of the content.
hidden - The overflow is clipped, and the rest of the content will be invisible.

The CSS:

```
div {
  width: 150px;
  height: 150px;
  background-color: LightBlue;
  float: left;
  overflow: hidden;
}
```

Try It Yourself

Result:

The z-index Property

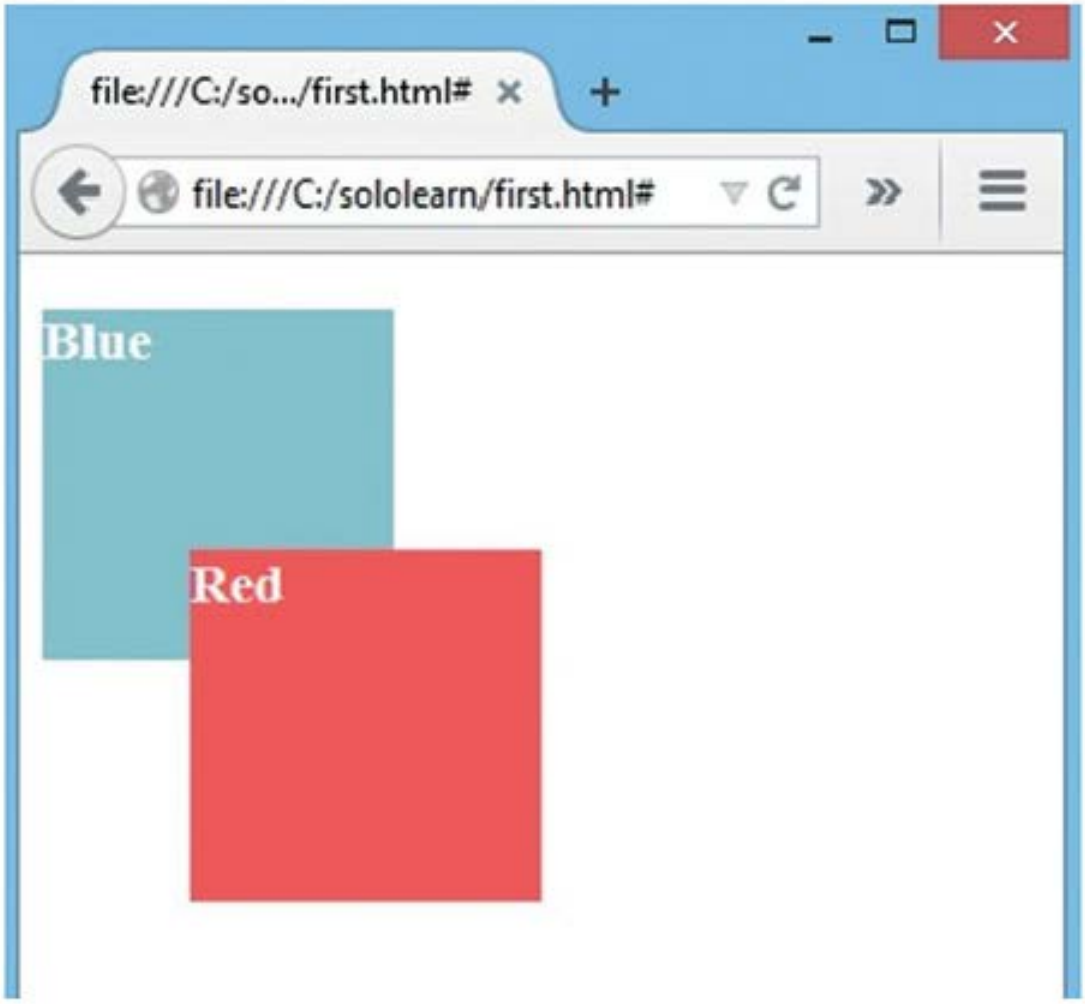
When elements are positioned outside the normal flow, they can overlap other elements.
The **z-index** property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

The CSS:

```
.blue {
  background-color: #8EC4D0;
  margin-bottom: 15px;
  width: 120px;
  height: 120px;
  color: #FFF;
}
.red {
  background-color: #FF4D4D;
  position: relative;
  width: 120px;
  height: 120px;
  color: #FFF;
  margin-top: -50px;
  margin-left: 50px;
}
```

Try It Yourself

Result:



Assigning the z-index Property

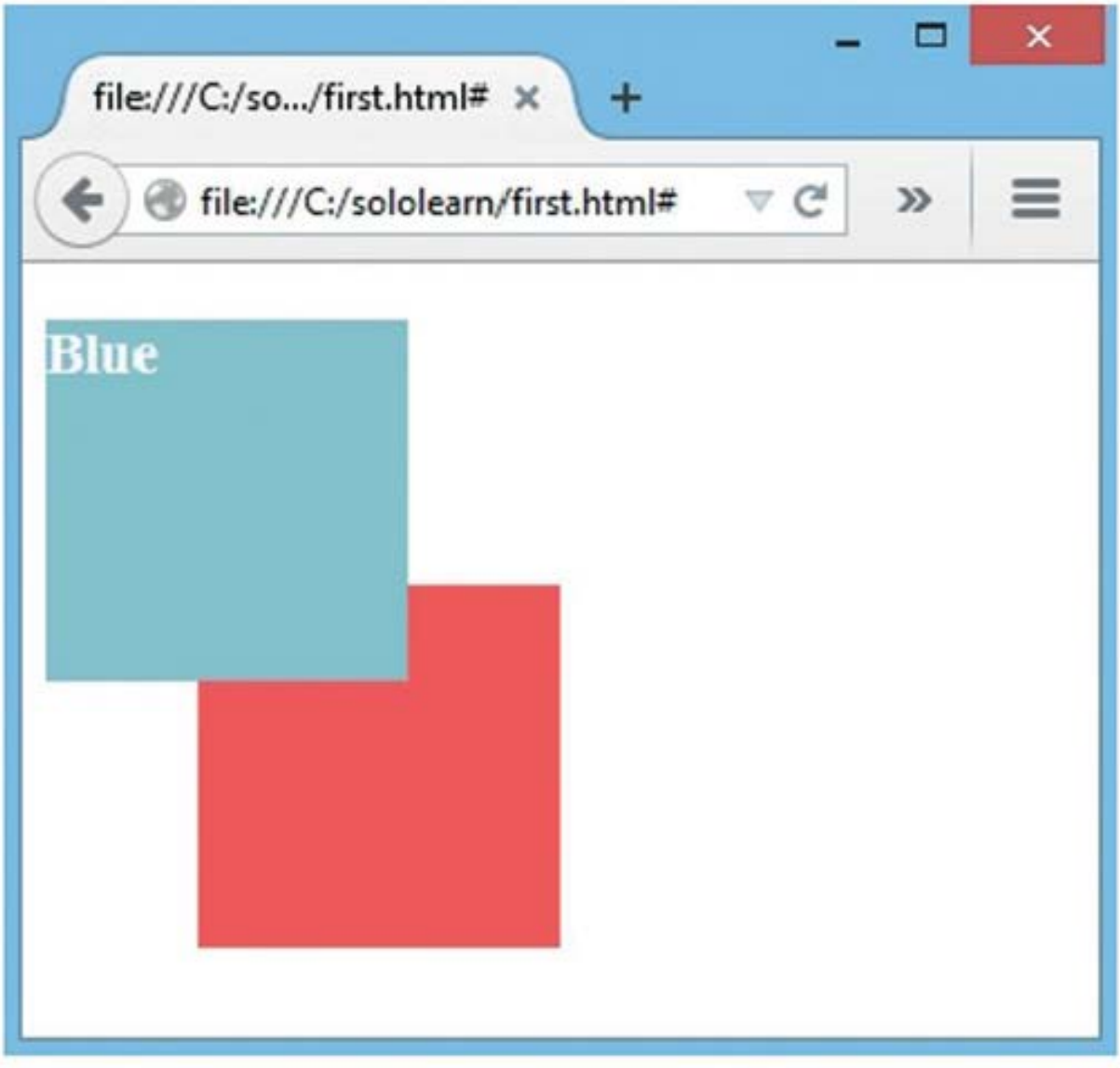
Assigning a higher z-index value to the blue div and a lower z-index value to the red div will result in the following:

The CSS:

```
.blue {
  z-index: 3;
  position: relative;
}
.red {
  z-index: 2;
  position: relative;
}
```

Try It Yourself

Result:



CSS3 Basics

CSS3

CSS3 is the latest standard for CSS.
CSS3 is completely backwards-compatible with earlier CSS versions.

Some of the most significant new features are:
Border radius - allows us to create rounded corners for elements.
Border images - allows us to specify an image as the border around an element.
Multiple backgrounds - applies multiple backgrounds to elements.
Animations and effects, and much more!

> There are lots of other great features that will be discussed in upcoming lessons.

317 COMMENTS

Q&A

CSS3: New Features

To make web development easier and faster, CSS3 introduces additional new features, including the following:

Box Shadow
With the **box-shadow** property, you can attach one or more shadows to an element by specifying values for color, size, blur, and offset.

Gradients
CSS3 gradients allow us to set the background color of the element to a gradient. Two types of gradients are available: **Linear** and **Radial**.

> Most of the new features have been implemented by major web browsers, so you can already enjoy the power of CSS3.

66 COMMENTS

CSS Vendor Prefixes

CSS vendor prefixes or CSS browser prefixes are a way for browser makers to add support for new CSS features during periods of testing and experimentation. Browser prefixes are used to add new features that may not be part of the final and formal CSS specification.

For example, the prefix for Safari and Chrome is **-webkit**. The **border-radius** property is currently supported in Chrome, Safari, and Mozilla, as long as it is accompanied by the browser prefix. To specify the **border-radius** in Chrome and Safari, the following syntax is used:

```
-webkit-border-radius: 24px;
```

Try It Yourself

Result:

The prefix is added to the property to make it work in the unsupported browsers. So, you might end up with multiple definitions of the same property, each with the specific browser prefix.

> While most browsers today will work without prefixes, it is essential to know these for backwards capability and understanding older codes.

136 COMMENTS

Browser Prefixes

Here is the list of vendor prefixes for each browser:

Browser	Vendor Prefix
Firefox	-moz-
Safari	-webkit-
Chrome	-webkit-
Opera	-o-
Internet Explorer	-ms

It might feel annoying and repetitive to have to write the properties two to five times to get them to work in all browsers, but it's temporary. As browsers improve, they add support for the standards based version of the properties, and you can remove the prefixed versions.

155 COMMENTS

The border-radius Property

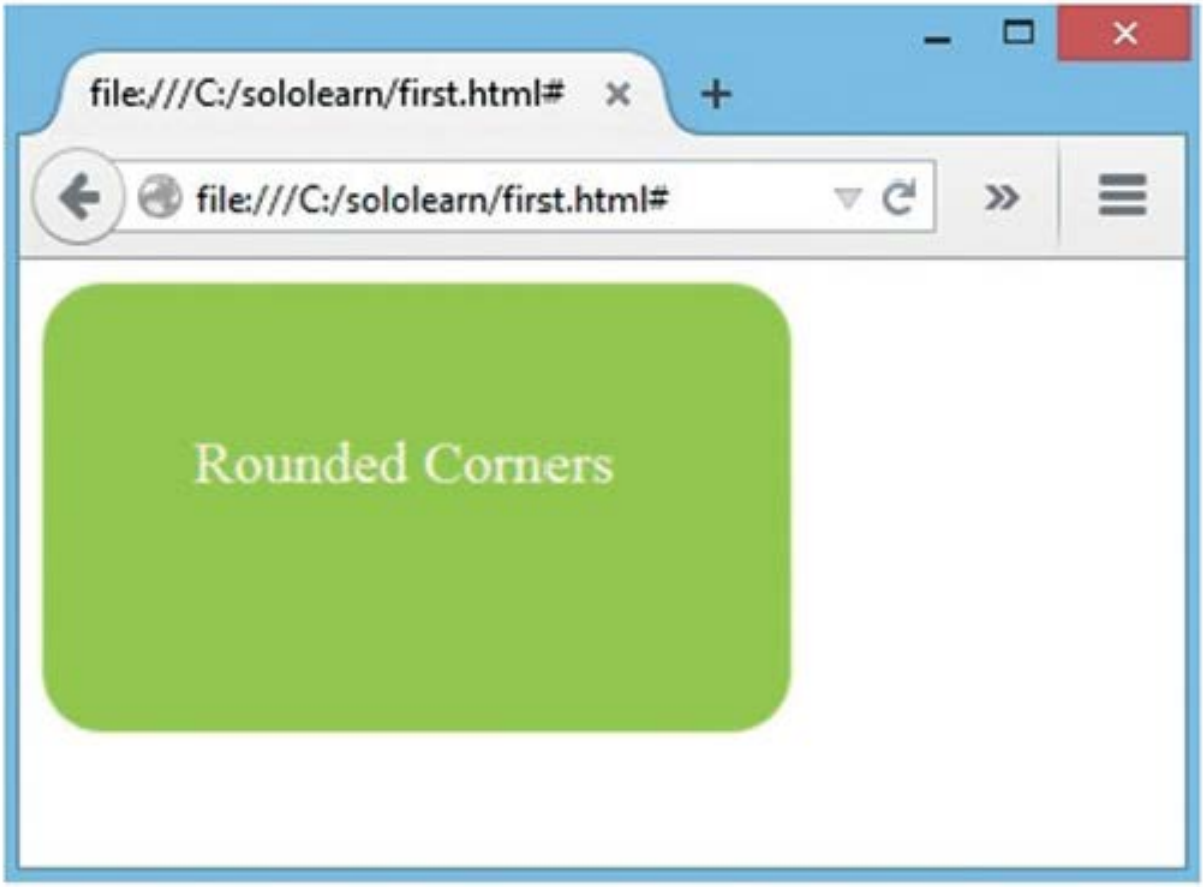
With CSS3, you can give any element "rounded corners" by using the **border-radius** property.

The CSS:

```
border-radius: 20px;
background-color: green;
color: white;
```

Try It Yourself

Result:

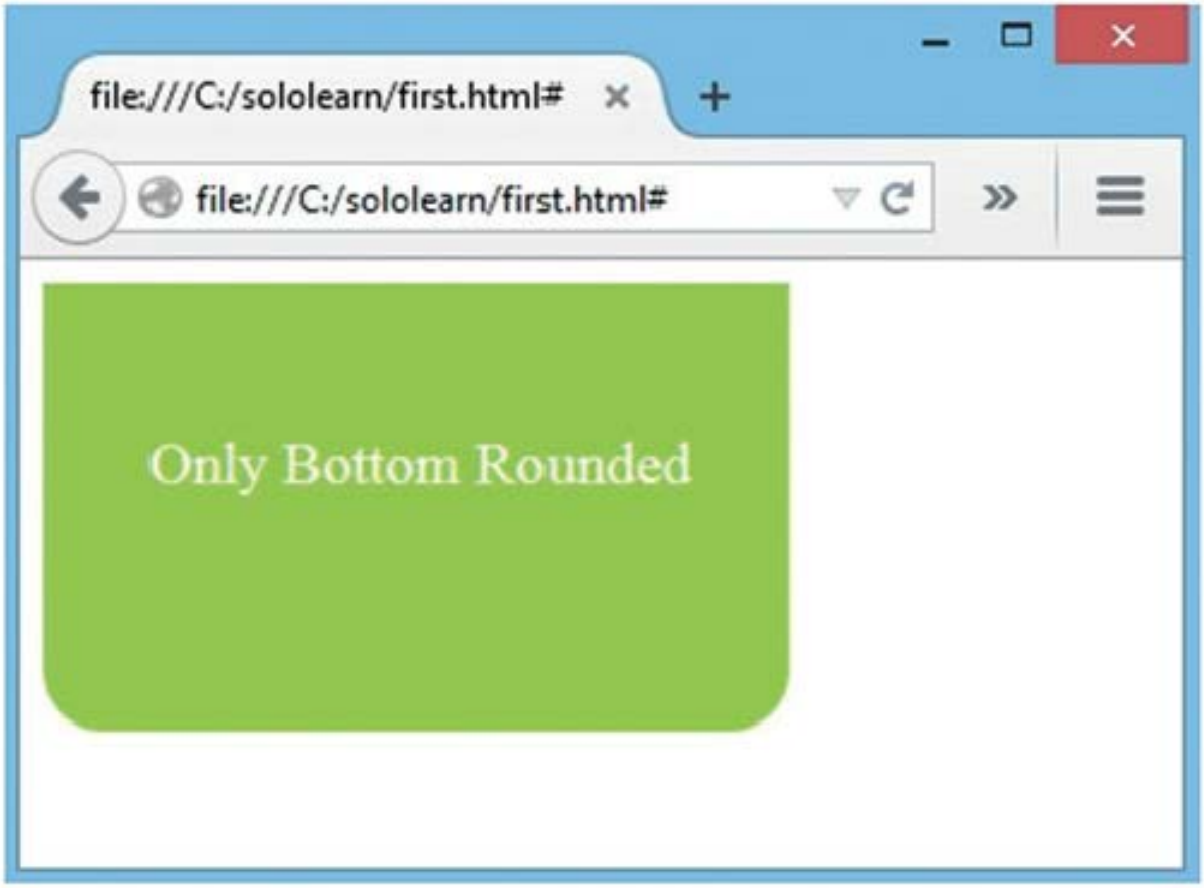


Specific border radius values can be applied for the border-radius property in the following order: **top-left, top-right, bottom-right, bottom-left.**

```
border-radius: 0 0 20px 20px;
```

Try It Yourself

Result:



The value of border-radius can also be specified in **percentages**.

131 COMMENTS

Creating a Circle

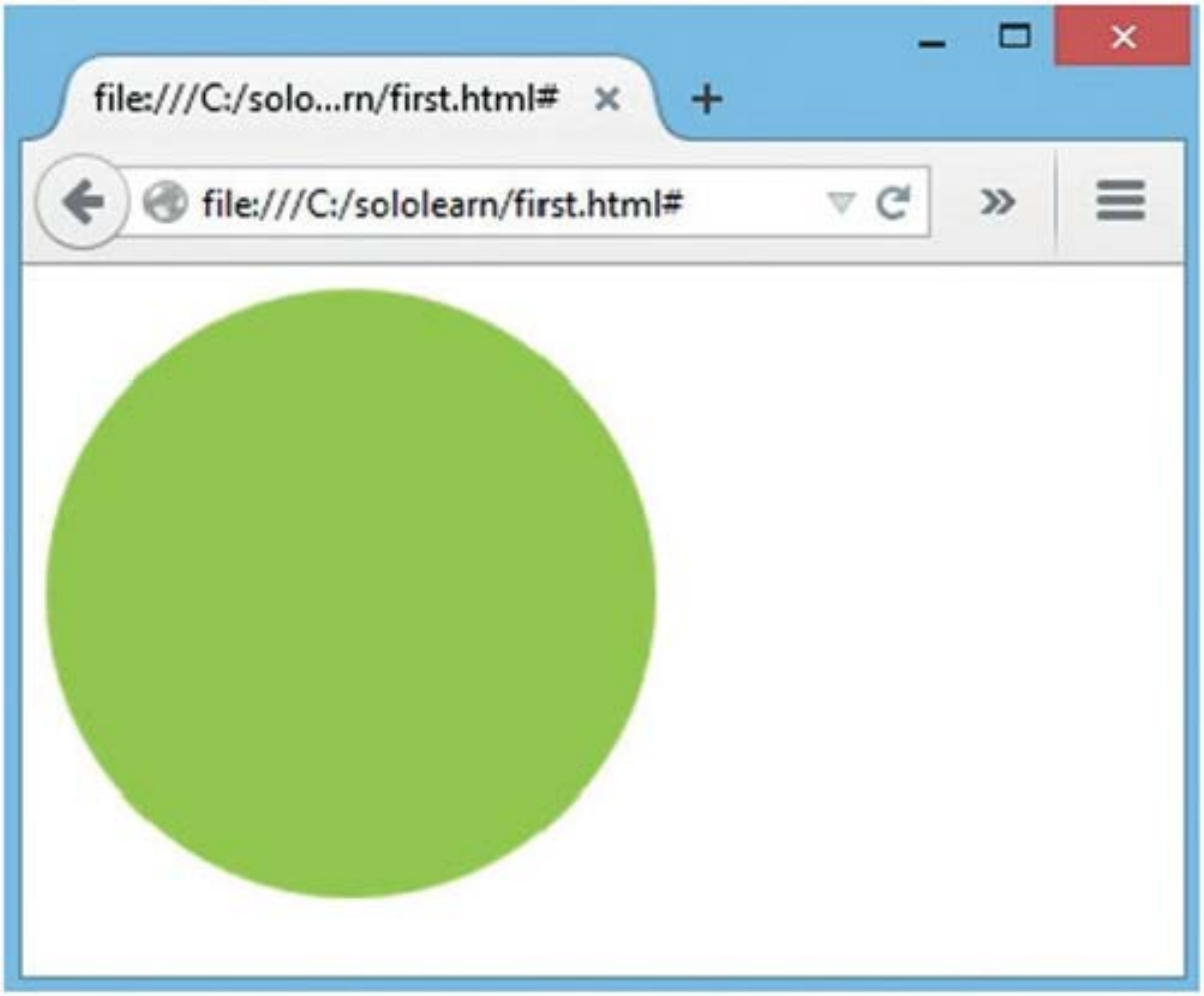
A rectangle can be turned into a circle using only CSS.
To create a circle, the border radius should be half of the height and the width.

The rectangle in the example below has a width and height of 200px. By setting the border radius to **100px**, the corners will be rounded to form a circle:

```
div {
  width: 200px;
  height: 200px;
  border-radius: 100px;
  background-color: green;
  color: white;
}
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

The box-shadow Property

The CSS3 **box-shadow** property applies shadow to elements.
Components of the box-shadow property are decoded by browsers in the following manner:
- The first length for the **horizontal offset** will cast the shadow to the right of the box (required)
- The second length is for the **vertical offset** that will cast the shadow to below the box (required)
- The **color** of the shadow (optional)

In the example below, the horizontal and vertical offsets have been set to 10px:

```
div {
  width: 300px;
  height: 100px;
  background-color: #9ACD32;
  box-shadow: 10px 10px #888888;
}
```

Try It Yourself

Result:

Tap **Try It Yourself** to play around with the code!

146 COMMENTS

Blur and Spread

Besides color, there are also two optional values for the box-shadow element, which are **blur** and **spread**.

The blur and spread values should be used before the color value.

```
box-shadow: 10px 10px 5px 5px #888888;
```

Try It Yourself

Result:

Tap **Try It Yourself** to play around with the code!

122 COMMENTS

Negative Values

Negative values can also be used for the box-shadow property.

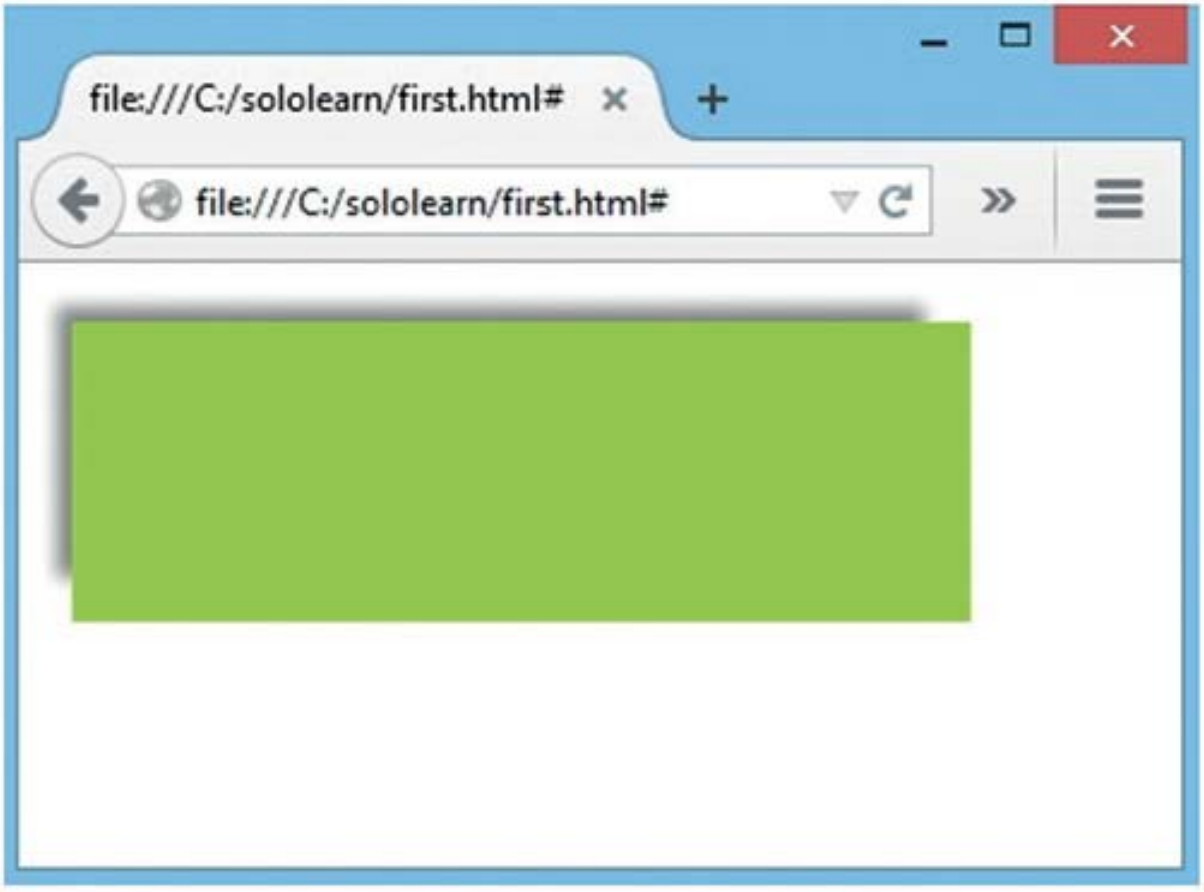
horizontal offset - the shadow will be to the **left** of the box
vertical offset - the shadow will be **above** the box
blur radius - negative values are **not allowed**
spread radius - negative values will cause the shadow to **shrink**

For example:

```
box-shadow: -10px -10px 5px -5px #888888;
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

59 COMMENTS

Creating an Inner Shadow

The **"inset"** keyword allows to draw an inner shadow in the box. To show an inset shadow, just add the inset keyword:

```
box-shadow: inset 10px 10px 5px #888888;
```

Try It Yourself

Result:

You can simultaneously create **inner** and **outer** shadows by separating each shadow with a **comma**.

131 COMMENTS

Layering Multiple Shadows

You can define as many shadows for the same box as you want by writing all of them **comma-separated** in the same rule.

In the example below, two **inner** shadows have been created by separating each shadow with a **comma**.

```
box-shadow:
inset 10px 10px 5px #888888,
inset -10px -10px 5px #888888;
```

Try It Yourself

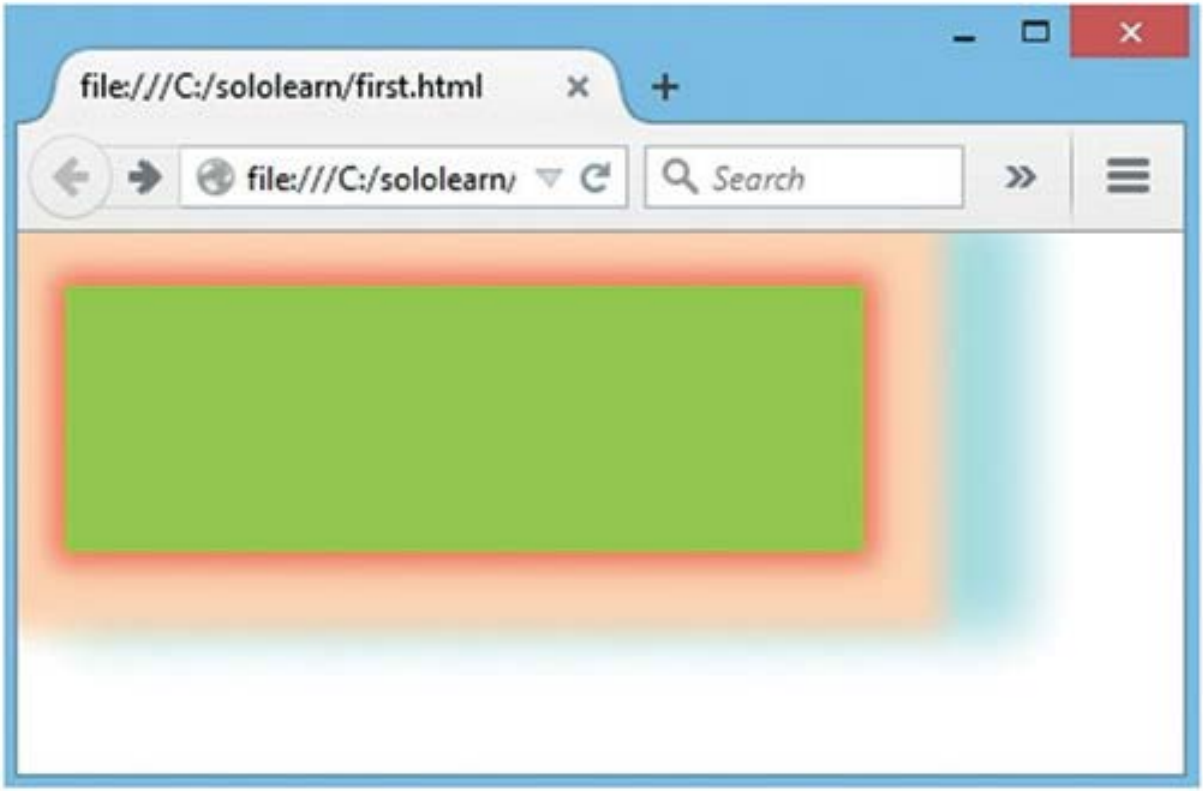
Result:

In case we specify more than one value, the one which comes last will be positioned at the back of all shadows.
Here is an example:

```
box-shadow: 0 0 10px 4px #FF6347,
0 0 10px 30px #FFDAB9,
30px 0 20px 30px #B0E0E6;
```

Try It Yourself

Result:



As expected, the blue shadow (**#B0E0E6**) comes last.

93 COMMENTS

Transparency Effect

Before CSS3, transparent background images were used to create transparency effects. The new features of CSS3 now make it easier to create transparency effects.

CSS supports color names, hexadecimal, and RGB colors.
In addition, CSS3 introduces the following:

RGBA Colors
RGBA color values are an extension of RGB color values with an alpha channel, which specifies the opacity for a color.
An RGBA color value is specified with: **rgba(red, green, blue, alpha)**. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

HSL (Hue, Saturation, Lightness) Colors
An HSL color value is specified with: **hsl(hue, saturation, lightness)**.
Hue is a degree on the color wheel ranging from 0 to 360
0 (or 360) is red, 120 is green, 240 is blue.
Saturation is a percentage value: 100% is the full color.
Lightness is also a percentage; 0% is dark (black) and 100% is white. **HSLA** color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color (just like RGBA).

In the example below, a transparent glass menu bar is created with CSS3.

In the HTML file, a <nav> tag containing an <ul> list with links has been added:

```
<nav>
  <ul>
    <li><a href="#">COURSES</a></li>
    <li><a href="#">DISCUSS</a></li>
    <li><a href="#">TOP LEARNERS</a></li>
    <li><a href="#">BLOG</a></li>
  </ul>
</nav>
```

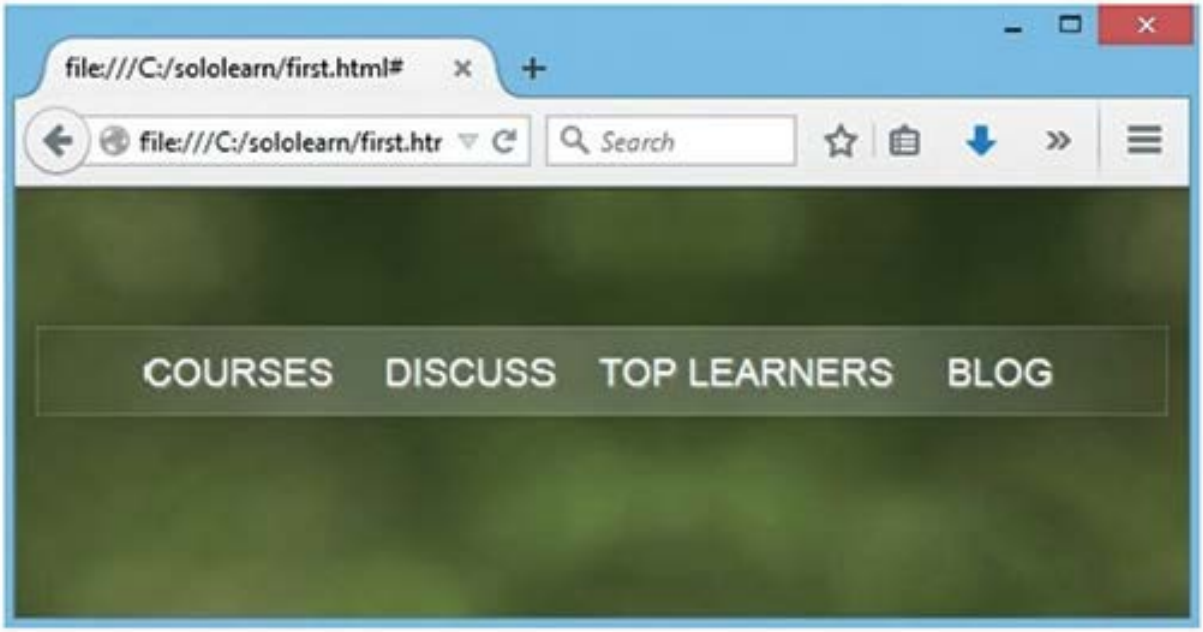
A number of CSS3 features are used to create the effects:

```
body {
  background:url("bg.jpg");
}
nav {
  padding: 50px 0;
  min-width: 500px;
}
nav ul {
  background: linear-gradient(90deg,
    rgba(255, 255, 255, 0) 0%,
    rgba(255, 255, 255, 0.2) 25%,
    rgba(255, 255, 255, 0.2) 75%,
    rgba(255, 255, 255, 0) 100%);
  box-shadow: 0 0 25px rgba(0, 0, 0, 0.1),
    inset 0 0 1px rgba(255, 255, 255, 0.6);
}
```

```
}
nav ul li {
  display: inline-block;
}
nav ul li a {
  padding: 10px;
  color: #FFFFFF;
  font-size: 18px;
  font-family: Arial;
  text-decoration: none;
  display: block;
}
```

Try It Yourself

Result:



The <ul> tag has been styled with a background gradient that is white and transparent. Two box-shadow values have been added, one for an outer, dark shadow; and one for an inner, light edge.

> Some of the properties used (like the background gradients) will be discussed in the upcoming lessons.

344 COMMENTS

Q&A

The text-shadow Property

The **text-shadow** property defines one or more comma-separated shadow effects, to be applied to the text content of the current element.

The image below shows how the text-shadow property is applied:

- The **offset-x** and **offset-y** values are required for the CSS text-shadow property.
- The **color** value is not required, but since the default for the text-shadow is transparent, the text-shadow will not appear unless you specify a color value.

72 COMMENTS

Q&A

Multiple Text Shadows

The text-shadow style can accept multiple values in a comma-separated list.
According to CSS2, the shadows are laid down in the order they appear, so if they overlap, the last one that is specified should appear on top. CSS3 has now changed that so they are applied in **reverse order**.

To create multiple shadows, the shadows are separated with a comma. Here is an example:

```
h1 {
  text-shadow: 5px 10px 2px #93968f,
    -3px 6px 5px #58d1e3;
}
```

Try It Yourself

This example defines two text shadows at different locations, blur radius, and colors. This makes it look like there are two different light sources on the text.

To make a text shadow look realistic, remember these few shadow characteristics:

- A shadow which is close to the text is normally not as blurred as a shadow that is far from the text. A shadow that is far from the text usually implies a light source which is also far from the text.
- A shadow which is close to the text usually implies that the underlying surface is close, that the light is close, or both. A close shadow is often darker than a distant shadow, because less light can get in between the shape and the underlying surface.

To remove a text-shadow, set the text-shadow property to **none;** no shadows will be associated with that element.

Working with Pseudo-Classes

The CSS pseudo-classes allow us to style elements, or parts of elements, that exist in the document tree without using JavaScript or any other scripts. A pseudo-class starts with a ":" (colon).
The most commonly used pseudo-classes are **:first-child** and **:last-child**.

The **:first-child** pseudo-class matches an element that is the first child element of some other element.
In the following example, the selector matches any <p> element that is the first child of the div element:

The HTML:

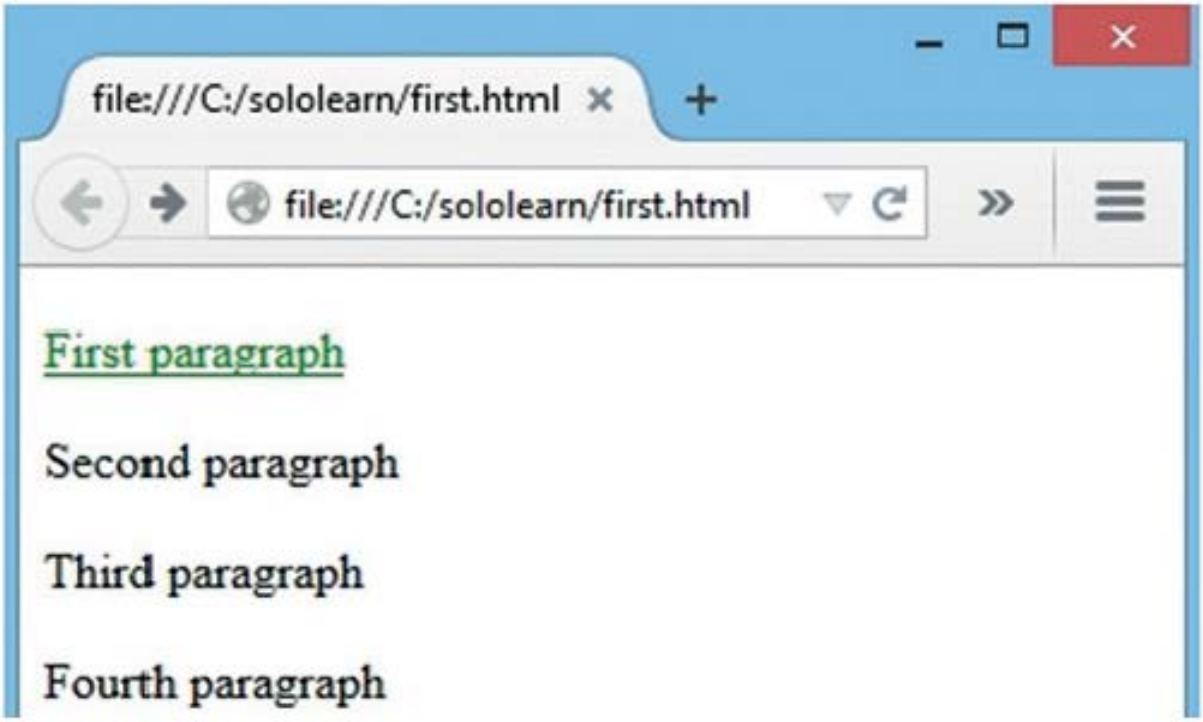
```
<div id="parent">
  <p>First paragraph</p>
  <p>Second paragraph</p>
  <p>Third paragraph</p>
  <p>Fourth paragraph</p>
</div>
```

The CSS:

```
#parent p:first-child {
  color: green;
  text-decoration: underline;
}
```

Try It Yourself

Result:



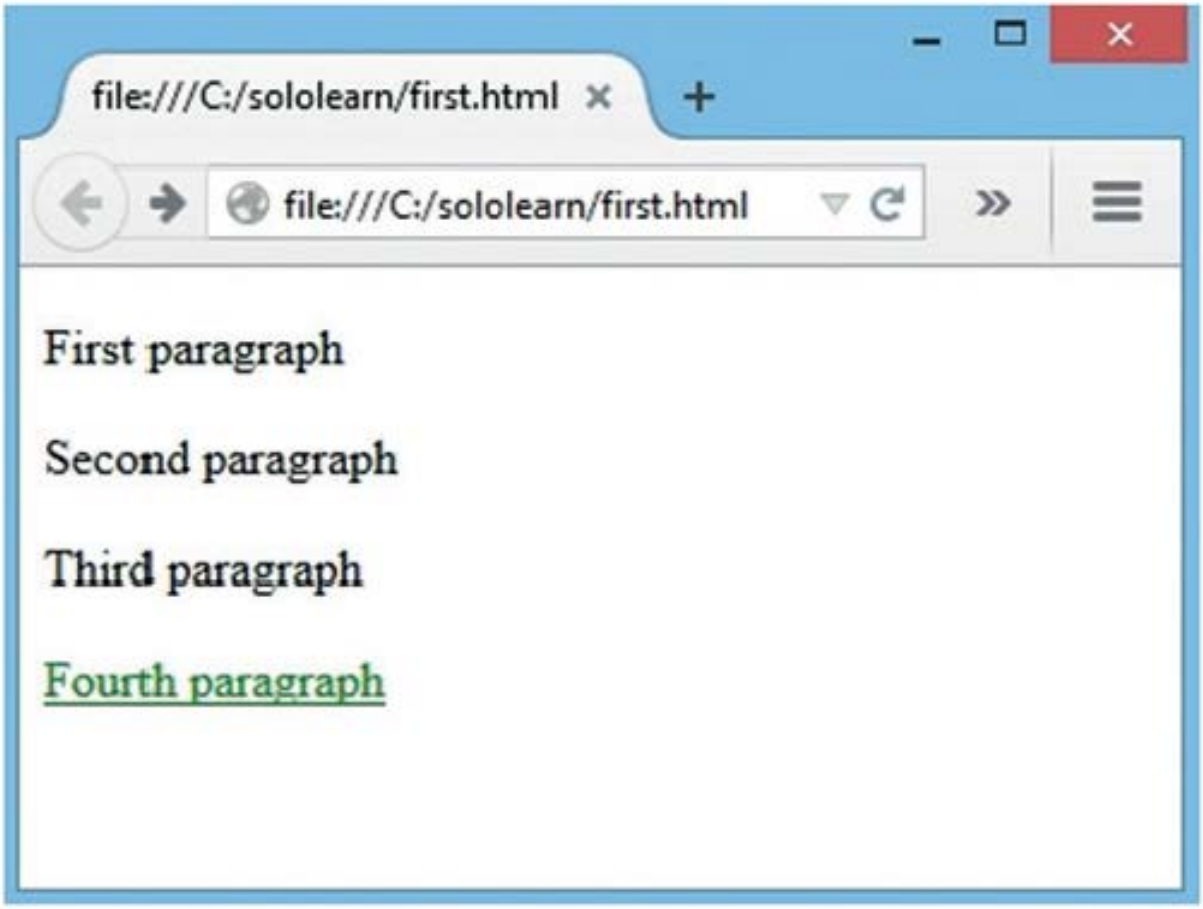
The **:last-child** pseudo-class matches an element that is the last child element of some other element.

In the example below, the selector will match any <p> element that is the last child of the div element:

```
#parent p:last-child {
  color: green;
  text-decoration: underline;
}
```

Try It Yourself

Result:



161 COMMENTS

Working with Pseudo Elements

Pseudo elements are used to select specific parts of an element.
There are five pseudo elements in CSS, each starting with a double colon (::):

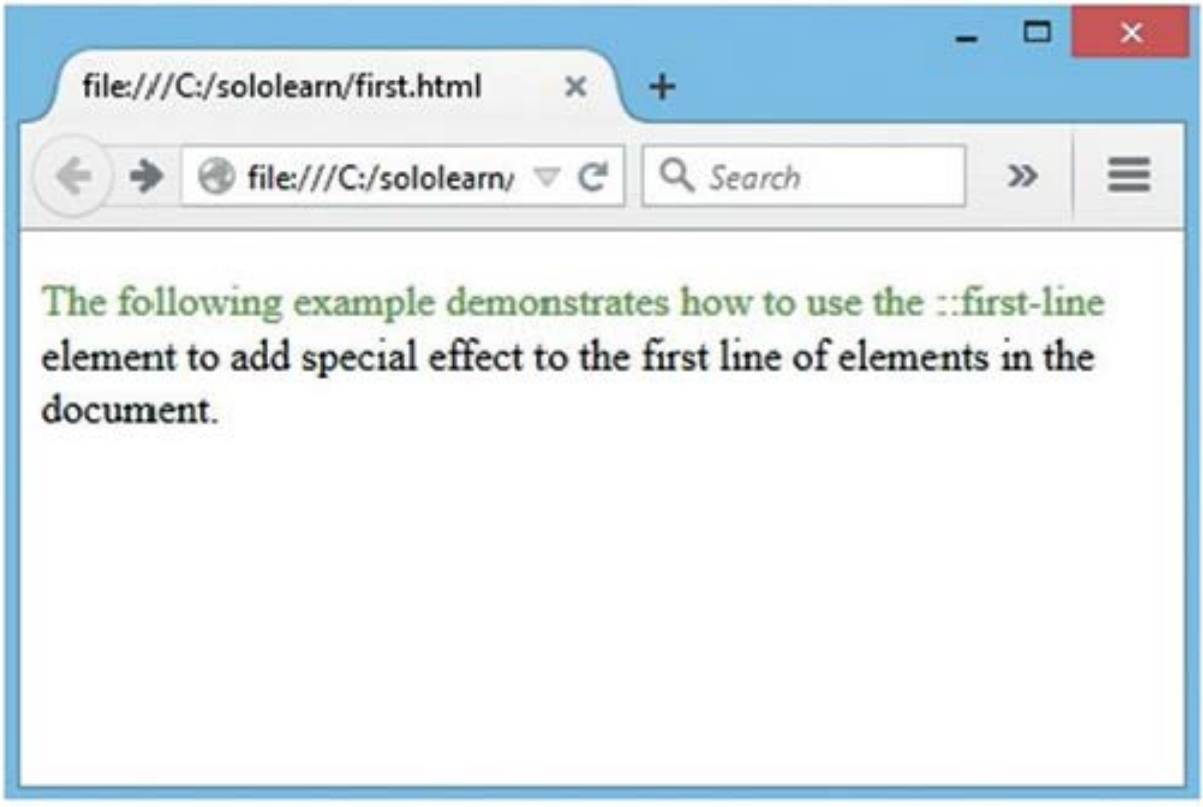
::first-line - the first line of the text in a selector
::first-letter - the first letter of the text in a selector
::selection - selects the portion of an element that is selected by a user
::before - inserts some content before an element
::after - inserts some content after an element

In the example below, the **::first-line** pseudo element is used to style the first line of our text:

```
p::first-line {
  color: #589432;
}
```

Try It Yourself

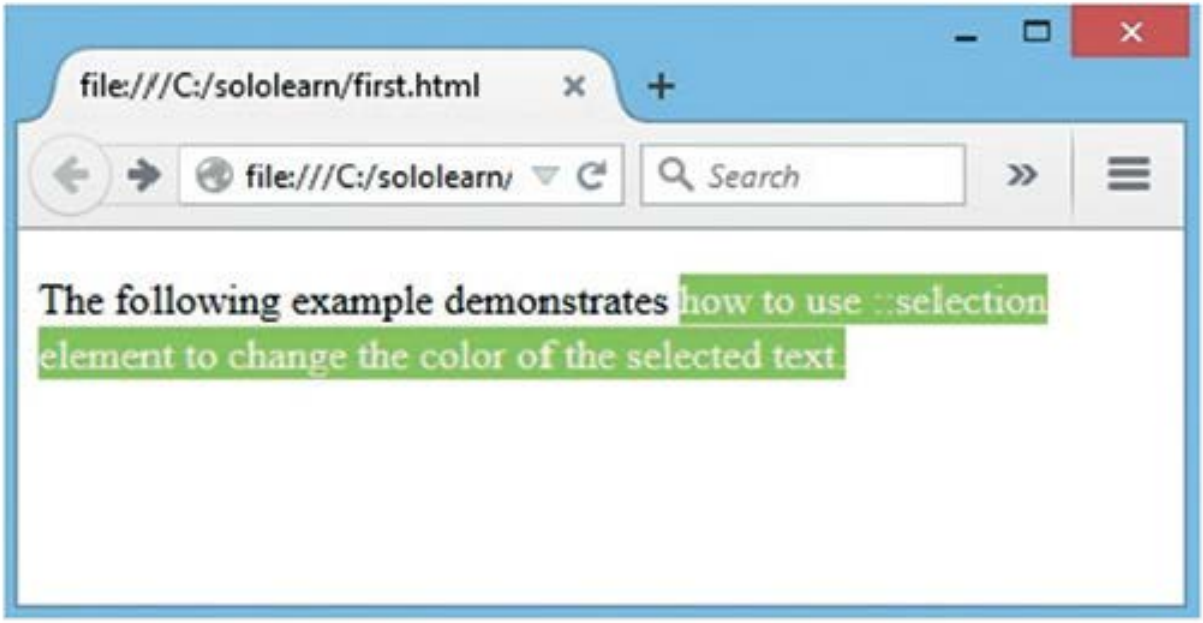
Result:



The **::selection** pseudo element styles the selected text:

```
p::-moz-selection {
  background: #8bc34a;
  color: #fff;
}
```

Result:



The **-moz-** prefix is used, as the **::selection** element is not supported by Mozilla yet.

183 COMMENTS

Working with Pseudo Elements

Using **::before** and **::after** pseudo elements allows us to add a wide variety of content to the page.

In the example below, the **::before** pseudo element is used to add an image before the list.
The HTML:

```
<p>You can insert text, images, and other resources using <strong>:before
</strong>pseudo element.</p>
<p>You can insert text, images, and other resources using <strong>:before
</strong>pseudo element.</p>
<p>You can insert text, images, and other resources using <strong>:before
</strong>pseudo element.</p>
```

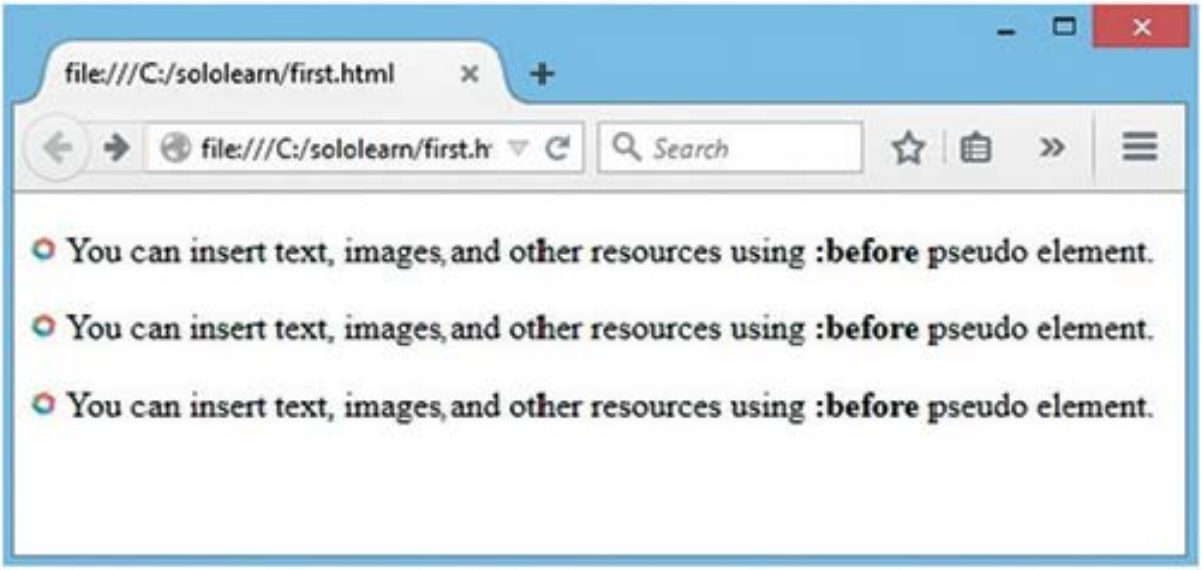
The CSS:

```
p::before {
  content: url("logo.jpg");
}
```

Try It Yourself

Note the **content** keyword in the syntax.

Result:



If you change the **::before** element to **::after** in the example above, the images will appear at the end of the list.

143 COMMENTS

The word-wrap Property

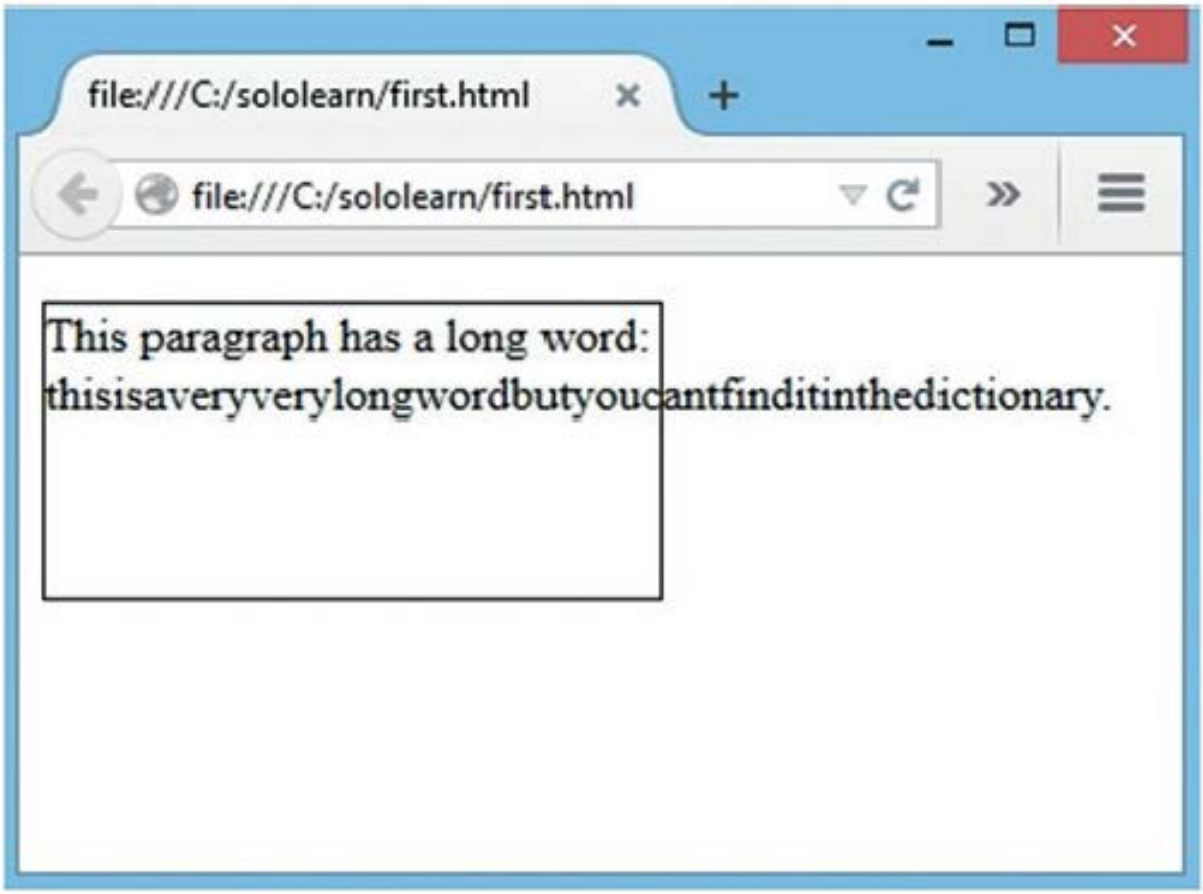
The word-wrap property allows long words to be broken and wrapped into the next line. It takes two values: **normal** and **break-word**.

In the example below, the word-wrap property is set to **normal**.
The CSS:

```
p {
  width: 210px;
  height: 100px;
  border: 1px solid #000000;
  word-wrap: normal;
}
```

Try It Yourself

Result:

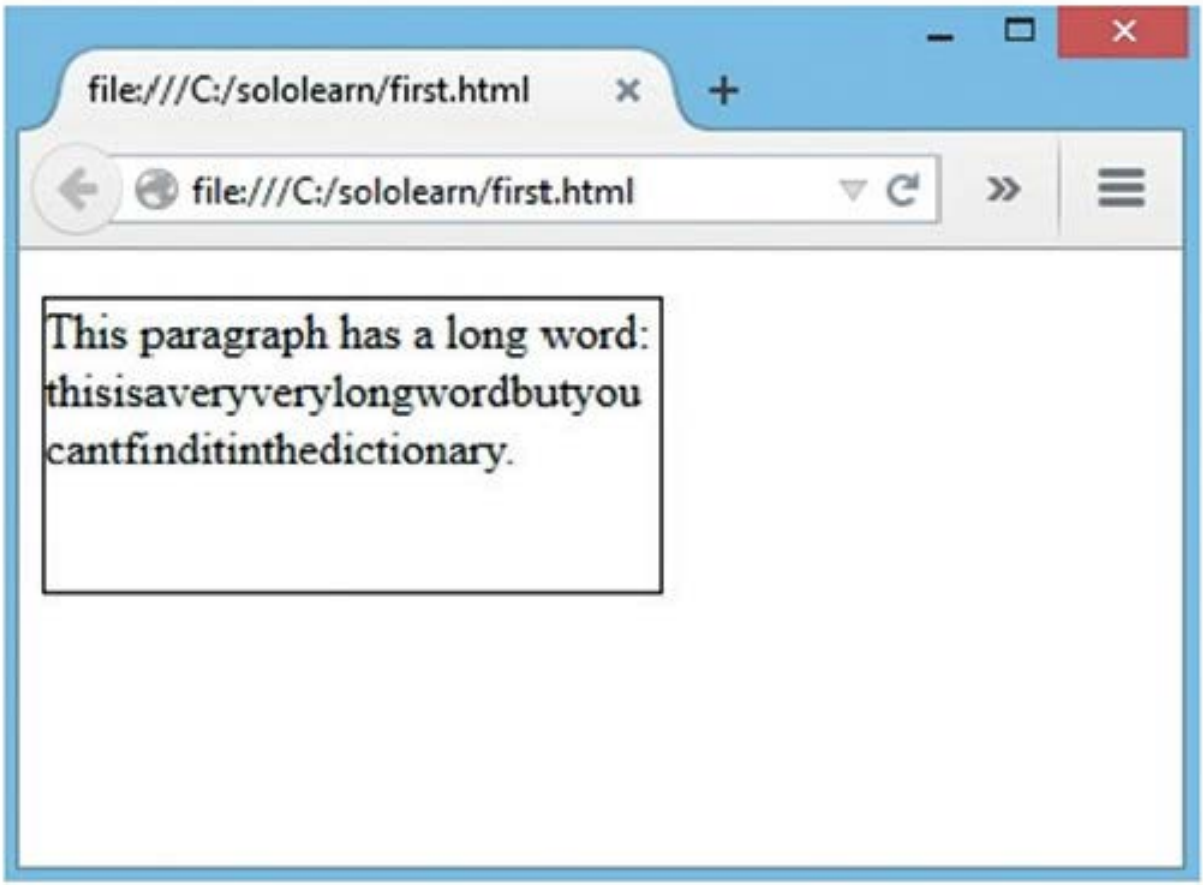


Now let's see what happens when we use this same example and set the word-wrap property to break-word:

```
p {
  width: 210px;
  height: 100px;
  border: 1px solid #000000;
  word-wrap: break-word;
}
```

Try It Yourself

Result:



When the word-wrap property is set to **break-word**, the browser breaks a word when it is too long to fit within its container.

72 COMMENTS

Q&A

The @font-face Rule

The **@font-face** rule allows custom fonts to be loaded into a webpage.
With the help of this rule, designs are no longer limited to the fonts that are installed on a user's computer.

In Internet Explorer 8 and earlier, the URL must point to an **Embedded OpenType** (eot) file, while Firefox, Chrome, etc. support **True Type Fonts** (ttf) fonts and **OpenType Fonts** (otf).

Browser	Font Type
Firefox	.ttf, .otf
Safari	.ttf, .otf
Chrome	.ttf, .otf
Opera	.ttf, .otf
Internet Explorer	.eot

In the @font-face rule, you must first define a name for the font (e.g., myFirstFont), and then point to the font file.

142 COMMENTS

Using the @font-face Rule

Each form of the font family must be declared using the **@font-face** rule. In the example below, a custom font called "Delicious" is loaded and used for the heading.

The HTML

```
<h1>This is Our Headline</h1>
```

The CSS

```
@font-face {
  font-family: Delicious;
  src: url('Delicious-Roman.otf');
}
@font-face {
  font-family: Delicious;
  font-weight: bold;
  src: url('Delicious-Bold.otf');
}
h1{
   font-family: Delicious, sans-serif;
}
```

Internet Explorer has a built-in bug when multiple @font-face rules are defined. Using **#iefix** as shown below fixes the problem:

```
@font-face {
   font-family: Delicious;
   src: url('Delicious-Roman.ttf');
   src: url('Delicious-Roman.eot?#iefix');
}
```

Result:

The question mark fools IE into thinking the rest of the string is a query string and loads just the EOT file. The other browsers follow the spec and select the format they need, based on the src cascade.

187 COMMENTS

Gradients & Backgrounds

Creating Linear Gradients

CSS3 gradients enable you to display smooth transitions between two or more specified colors. CSS3 defines two types of gradients: **Linear** and **Radial**.

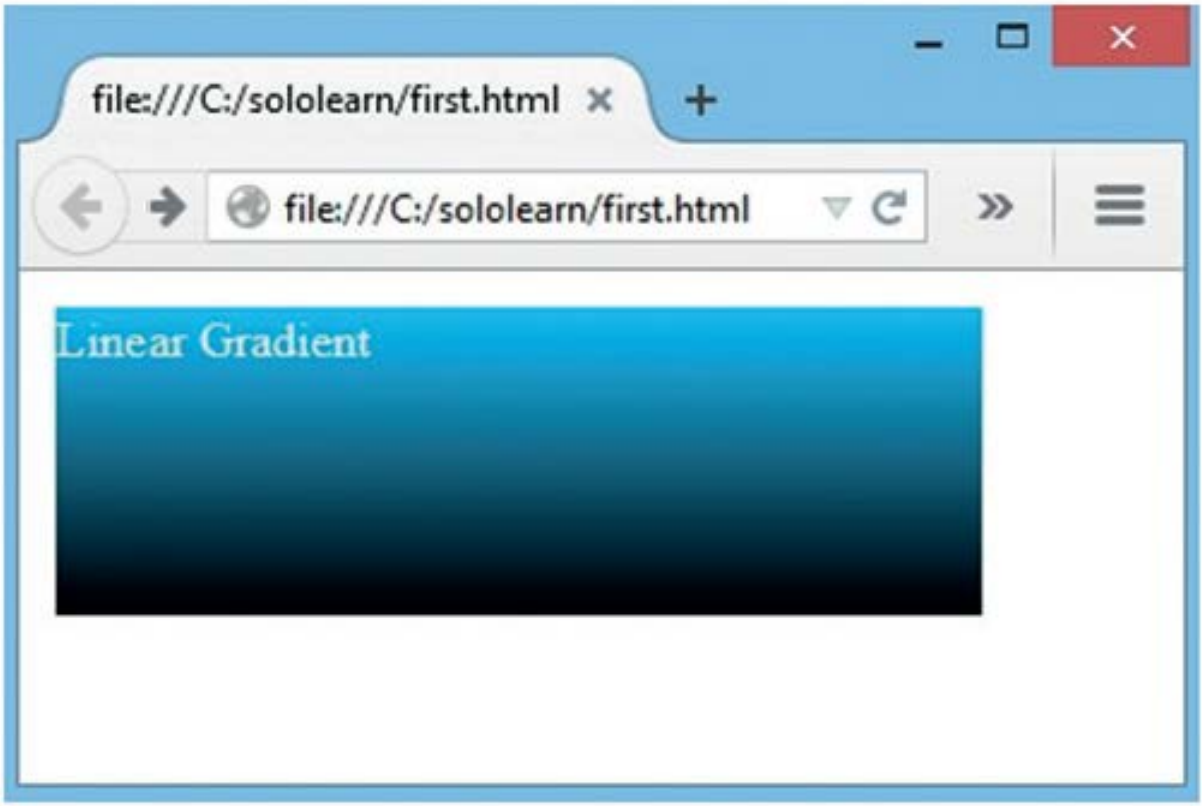
To create a linear gradient, you must define at least two color stops. Color stops are the colors among which you want to render smooth transitions. You can also set a starting point and a direction - or an angle - along with the gradient effect.
In the example below, the colors blue and black are used to create a linear gradient from top to bottom.

```
div {
  float: left;
  width: 300px;
  height: 100px;
  margin: 4px;
  color: #FFF;
  background:-moz-linear-gradient(DeepSkyBlue, Black);
}
```

Try It Yourself

This syntax works in Mozilla (-moz). If you work with a different browser, add the corresponding prefix, so that the browser understands the gradient.

Result:



You can use **color names**, **Hex values**, **RGB**, or **HSL** colors to define the gradient color.

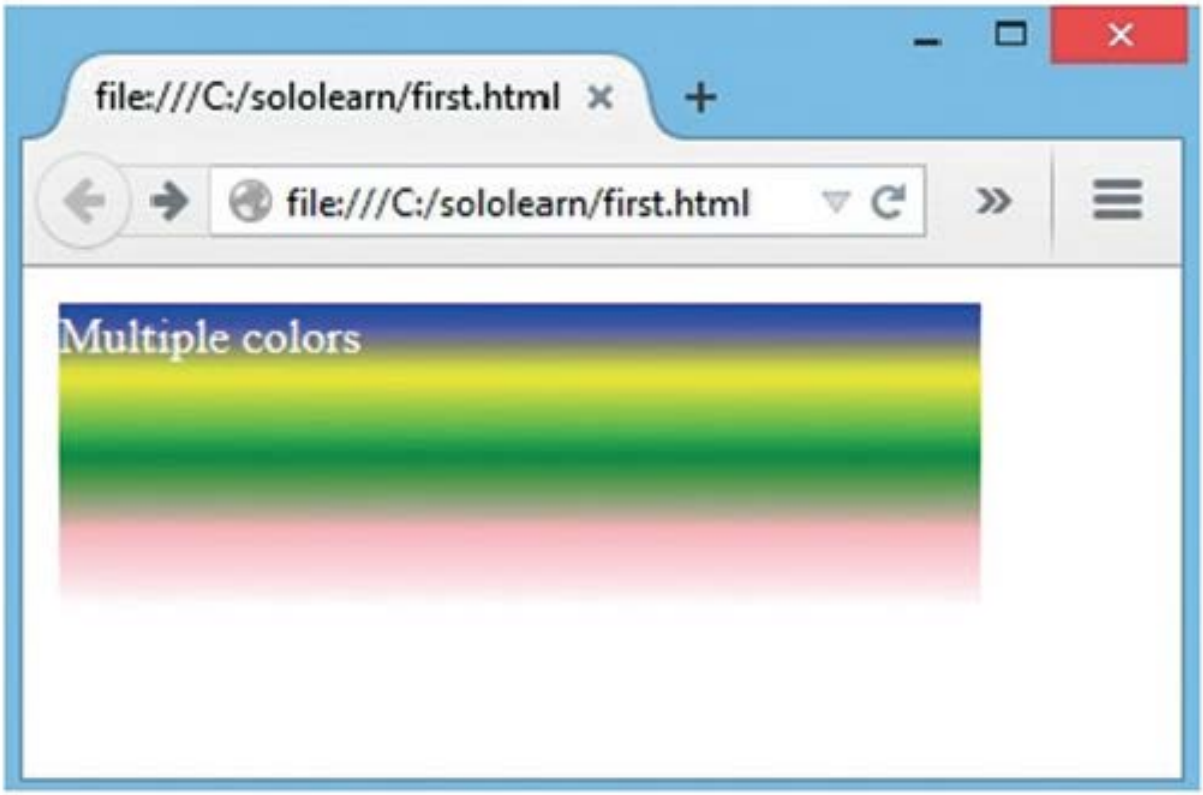
Color Stops

Colors can be added one after the other, separated with a comma. The browser will then determine each color stop position.
In the example below, the linear gradient has multiple color stops and runs from top to bottom.

```
background:-moz-linear-gradient(blue, yellow, green, pink, white);
```

Try It Yourself

Result:

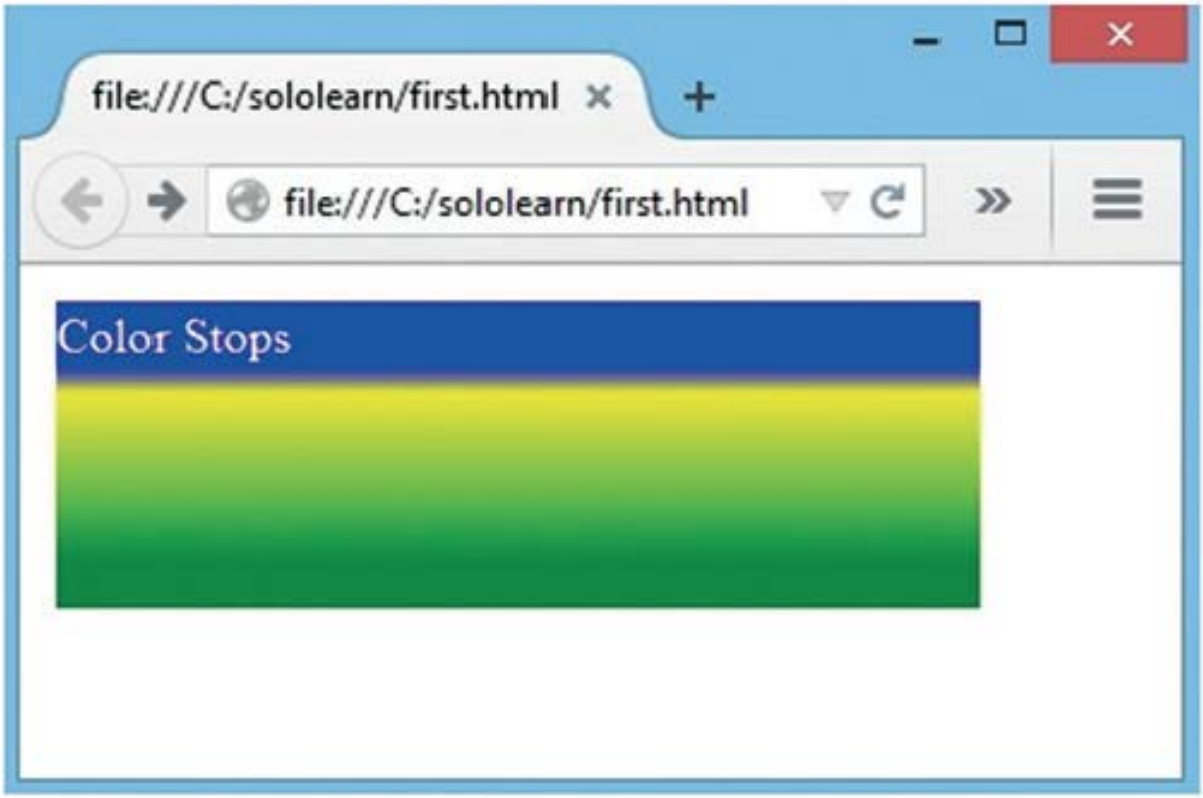


Color stop positions can be specified for each color.

```
background:-moz-linear-gradient(blue 20%, yellow 30%, green 85%);
```

Try It Yourself

Result:



In addition to percentages, you can also use **px**, **em**, and so on, to specify the color stops.
If you use the same color stop position for two colors, a sharp line will be created between them.

122 COMMENTS

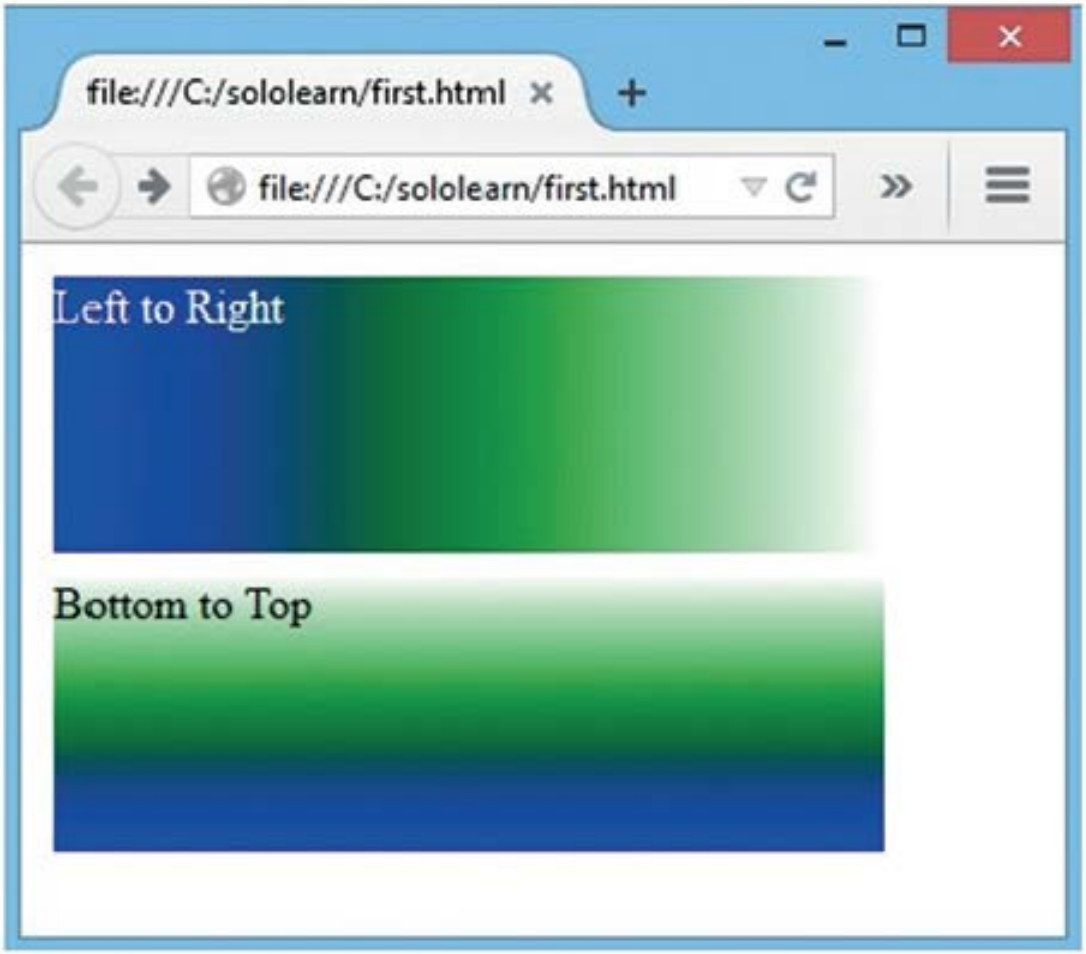
Direction of the Gradient

The direction of the gradient can be changed.
In the example below, the first gradient starts at **left**, moving **right**; the second one runs from **bottom** to **top**.

```
div.first {
  float: left;
  width: 300px;
  height: 100px;
  margin: 4px;
  color: #FFF;
  background:-moz-linear-gradient(left, blue, green, white);
}
div.second {
  float: left;
  width: 300px;
  height: 100px;
  margin: 4px;
  background:-moz-linear-gradient(bottom, blue, green, white);
}
```

Try It Yourself

Result:



left, **right**, **top**, and **bottom** are supported values for the gradient direction. You can also use their various combinations to specify direction (e.g., **bottom right**)

Angle of the Gradient

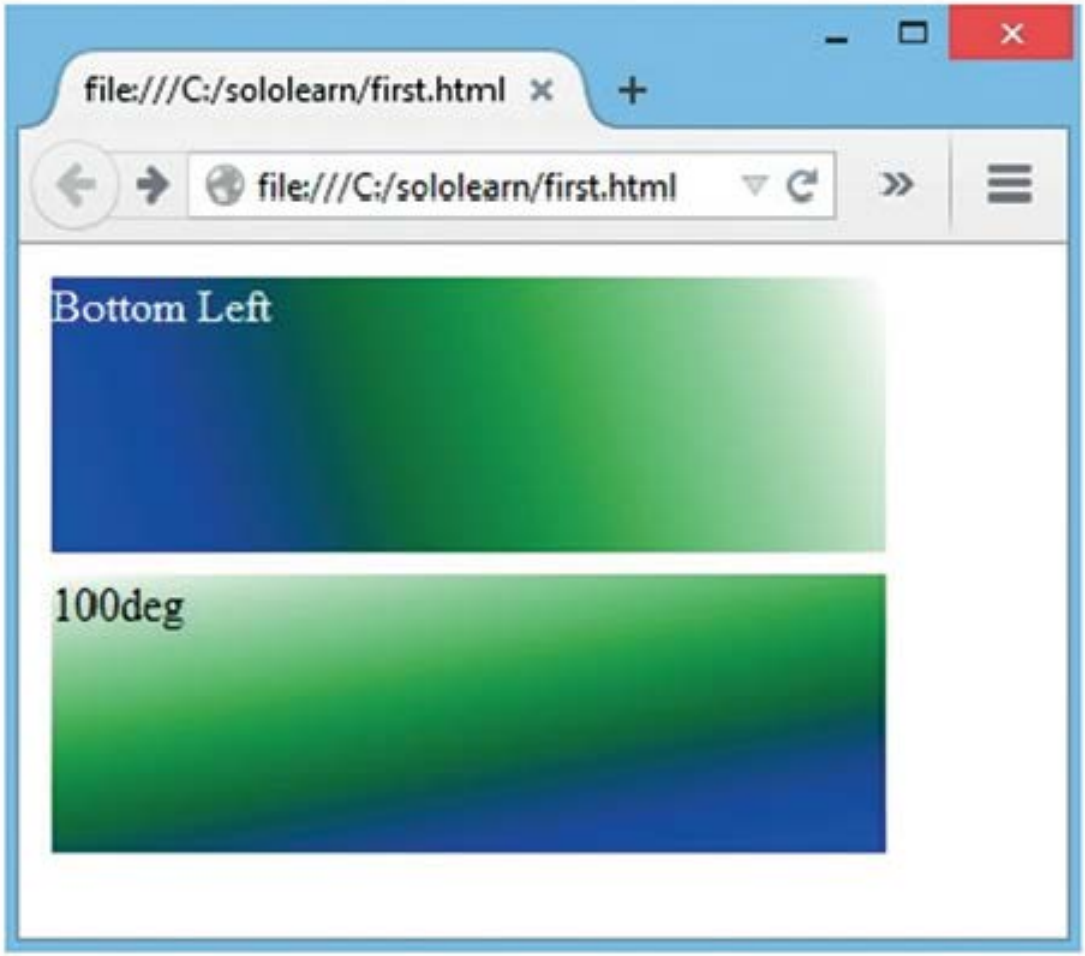
As an alternative to predefined directions (bottom, top, right, left, bottom right, etc.), you can control the gradient's direction by specifying an angle.

The angle is specified as an angle extending between a horizontal line and the gradient line. In other words, 0deg creates a left-to right-gradient, while 90deg generates a bottom-to-top gradient.

```
div.first {
  float: left;
  width: 300px;
  height: 100px;
  margin: 4px;
  color: #FFF;
  background:-moz-linear-gradient(bottom left, blue, green, white);
}
div.second {
  float: left;
  width: 300px;
  height: 100px;
  margin: 4px;
  background:-moz-linear-gradient(100deg, blue, green, white);
}
```

Try It Yourself

Result:



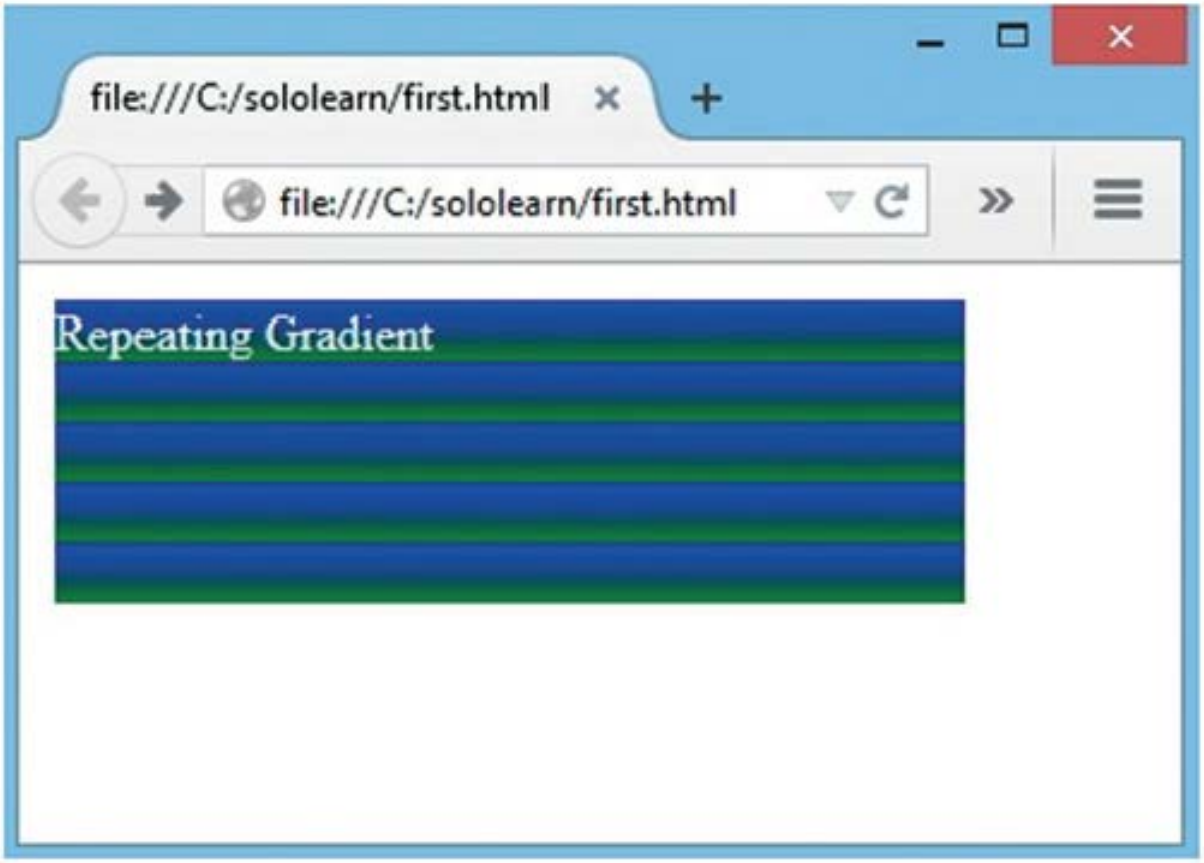
Repeating a Linear-Gradient

The **repeating-linear-gradient()** function is used to repeat a linear gradient:

```
background:-moz-repeating-linear-gradient(blue, green 20px);
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

97 COMMENTS

Radial Gradients

To create a radial gradient, you must define at least **two color stops**.
The radial gradient is defined by its **center**.

The CSS syntax of the radial gradient looks like this:

```
background: radial-gradient(position, shape or size, color-stops);
```

The **first value** defines the gradient position. We can use a descriptive keyword, such as top, bottom, center, or left; or we can specify, for example, 50% 50% to set the gradient at the center or 0% 0% to set the gradient to start at top left.

The **second value** defines the shape and the gradient size. There are two arguments to shape gradients: The first is the **ellipse, which is the default**; and the second is the **circle**.

Lastly, the **third value** defines the color combination.

82 COMMENTS

Setting the Shapes

The shape parameter defines the shape. If you do not define the shape of the radial gradient, it will take the ellipse value by default.

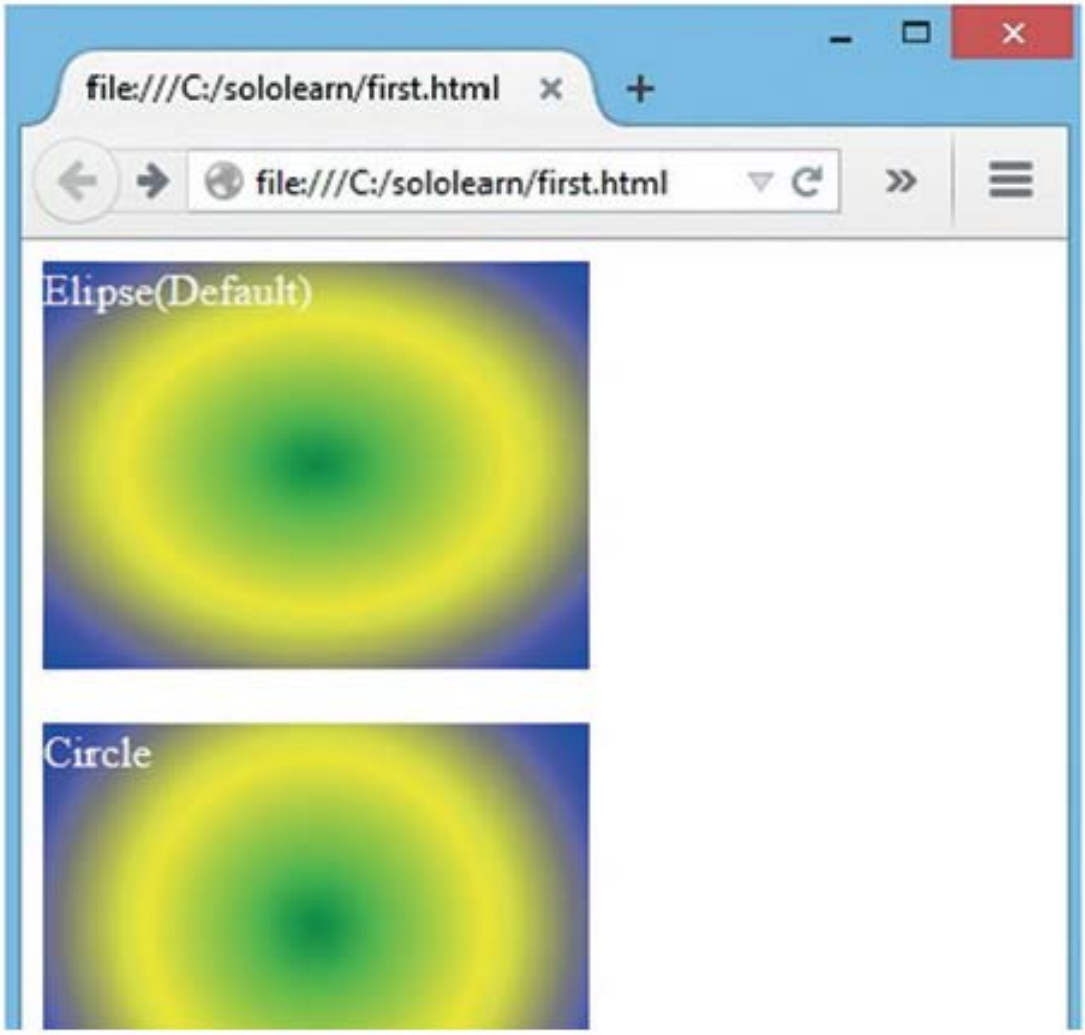
In the example below, we didn't specify the shape of the first div's radial gradient, but for the second, we set the value to circle.
Here is what happened:

The CSS:

```
div.first {
  height: 150px;
  width: 200px;
  color: #FFF;
  background: -moz-radial-gradient(green, yellow, blue);
}
div.second {
  height: 150px;
  width: 200px;
  color: #FFF;
  background: -moz-radial-gradient(circle, green, yellow, blue);
}
```

Try It Yourself

Result:



Radial Gradient Position

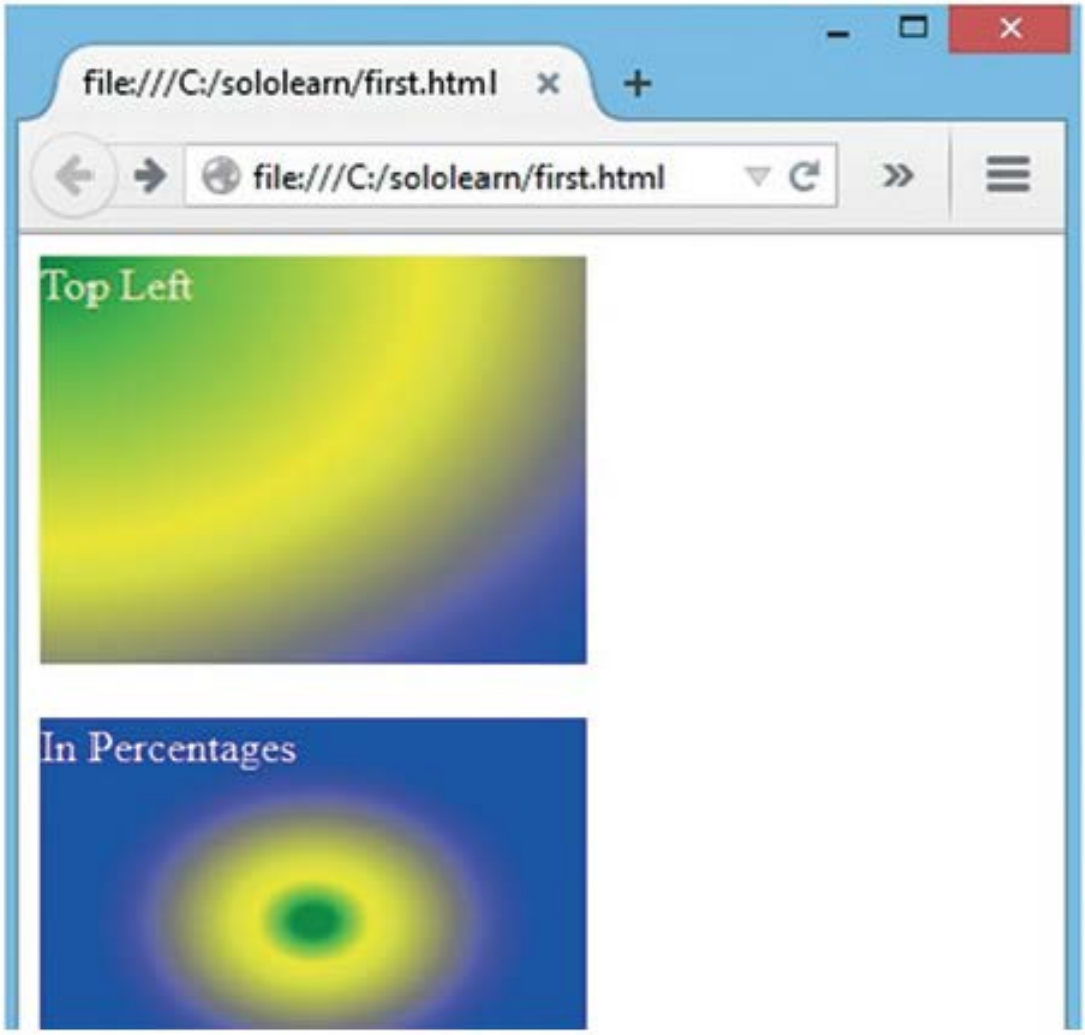
Essentially, we can use the same method used to specify the location of a background image with the background-position CSS property to specify the location of the ellipse's center. We specify the horizontal position of the background, and - optionally - the vertical position using either keywords (left, center right, or top, center, bottom), length values, percentage values, or some combination of these.

In the example below, the first gradient starts from the top left corner; in the second, we set 5% to the green, 15 % to the yellow and 60% to the blue.

```
div.first {
  height: 150px;
  width: 200px;
  color: #FFF;
  background: -moz-radial-gradient(top left, green, yellow, blue);
}
div.second {
  height: 150px;
  width: 200px;
  color: #FFF;
  background: -moz-radial-gradient(green 5%, yellow 15%, blue 60%);
}
```

Try It Yourself

Result:



Setting the Color Stops

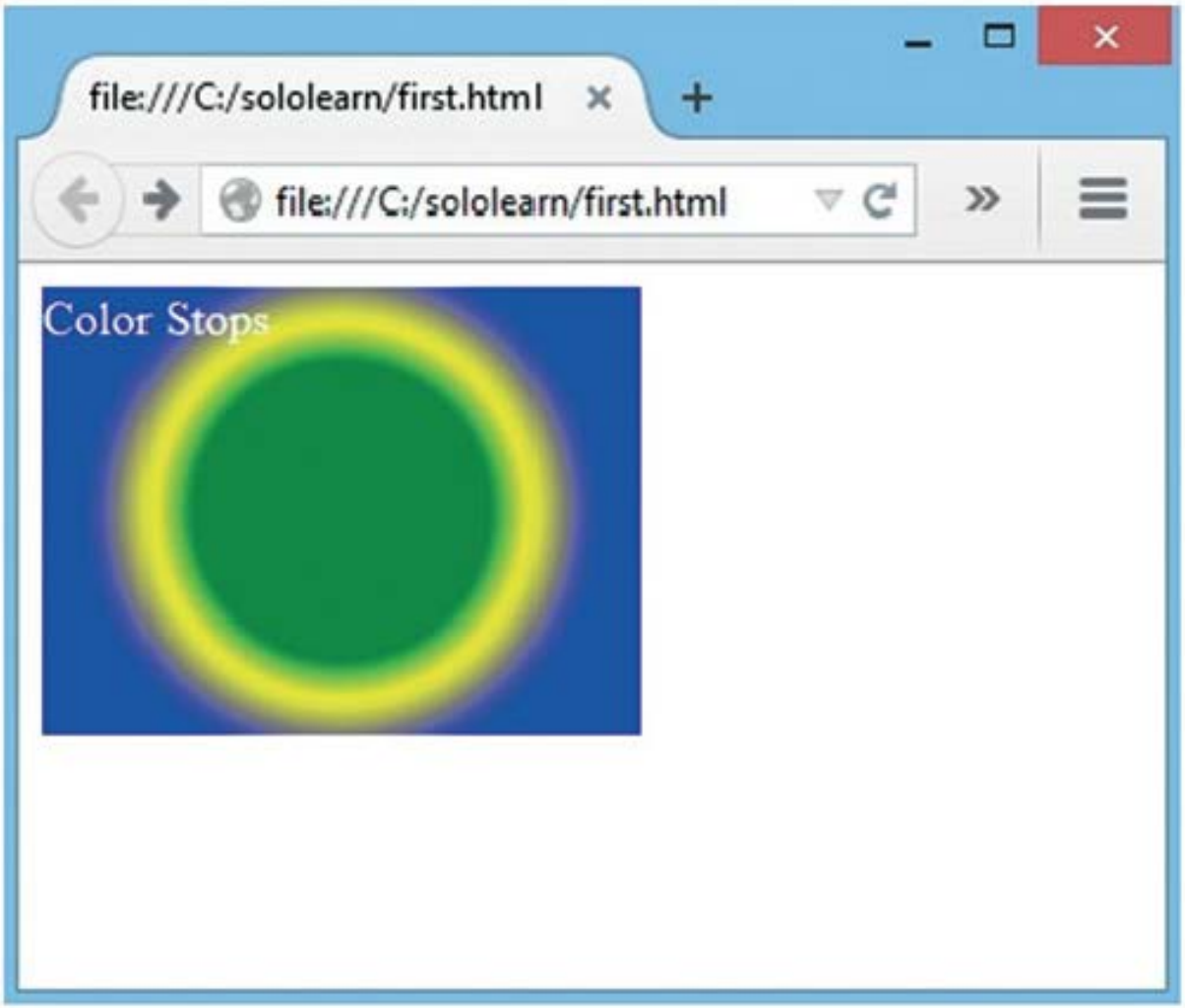
As with linear gradients, a color stop is specified with a color, plus an optional stop position, which is a length or percentage value.

Here's a simple circular gradient with color stops:

```
background: -moz-radial-gradient(circle, green 40%, yellow 50%, blue 70%);
```

Try It Yourself

Result:



Tap **Try It Yourself** to play around with the code!

80 COMMENTS

The background-size Property

The **background-size** property adds new functionality to CSS that allows us to specify the size of background images, using either lengths or percentages.

For example:

```
div {
  height: 150px;
  width: 200px;
  border: 1px solid #000;
  background: url("css_logo.png") no-repeat 50% 50%;
  background-size: 100px 100px;
}
```

Try It Yourself

Result:

Output

The current versions of most popular browsers - including Firefox, Safari, Chrome, Internet Explorer, and Opera - now support background-size, without the need for vendor prefixes.

125 COMMENTS

The background-size Values

The two other possible values for background size are the keywords **contain** and **cover**.

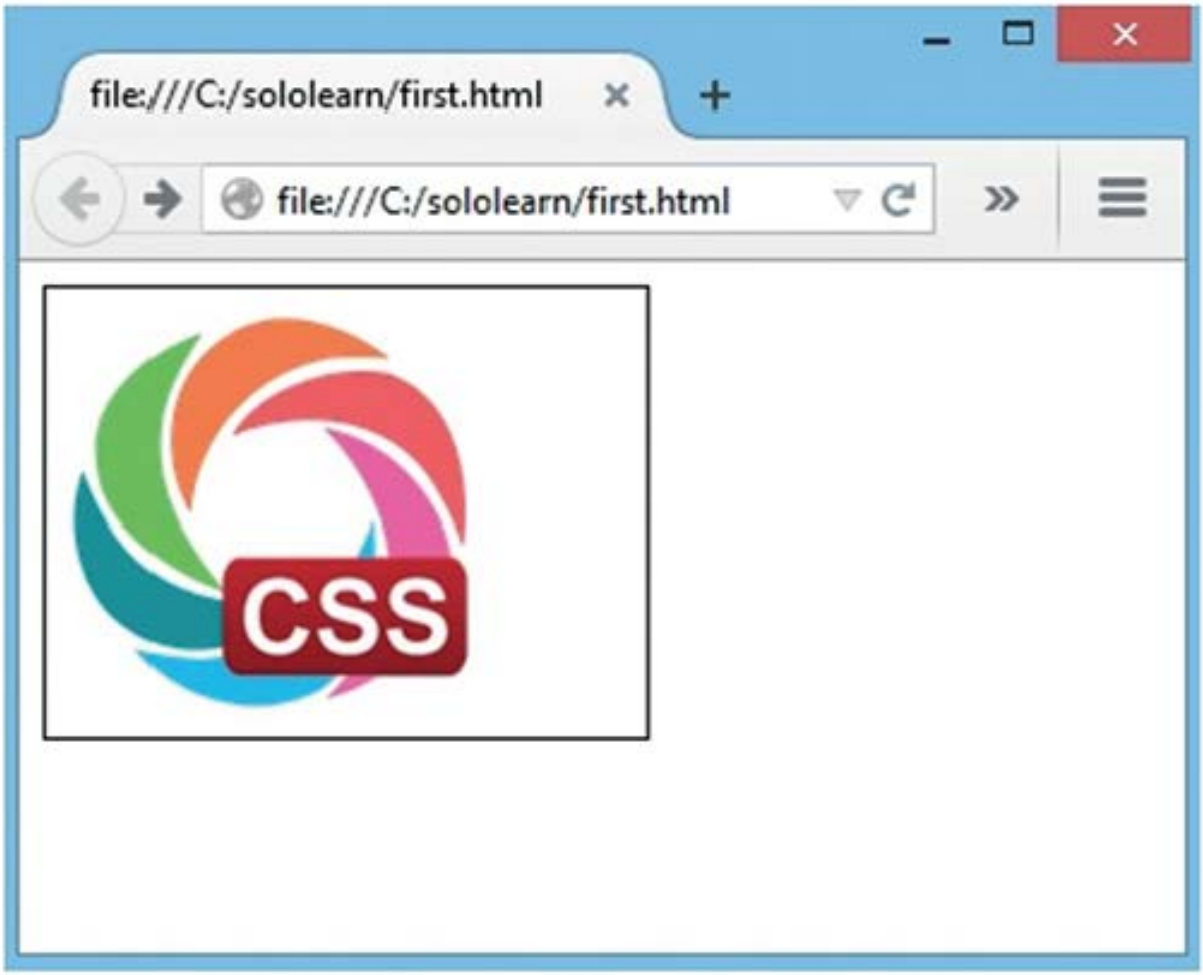
The **contain** keyword scales the image so that it fits the container.
In other words, the image will grow or shrink proportionally, but the width and height will not exceed the container's dimensions:

CSS syntax looks like this:

```
background-size: contain;
```

Try It Yourself

Result:

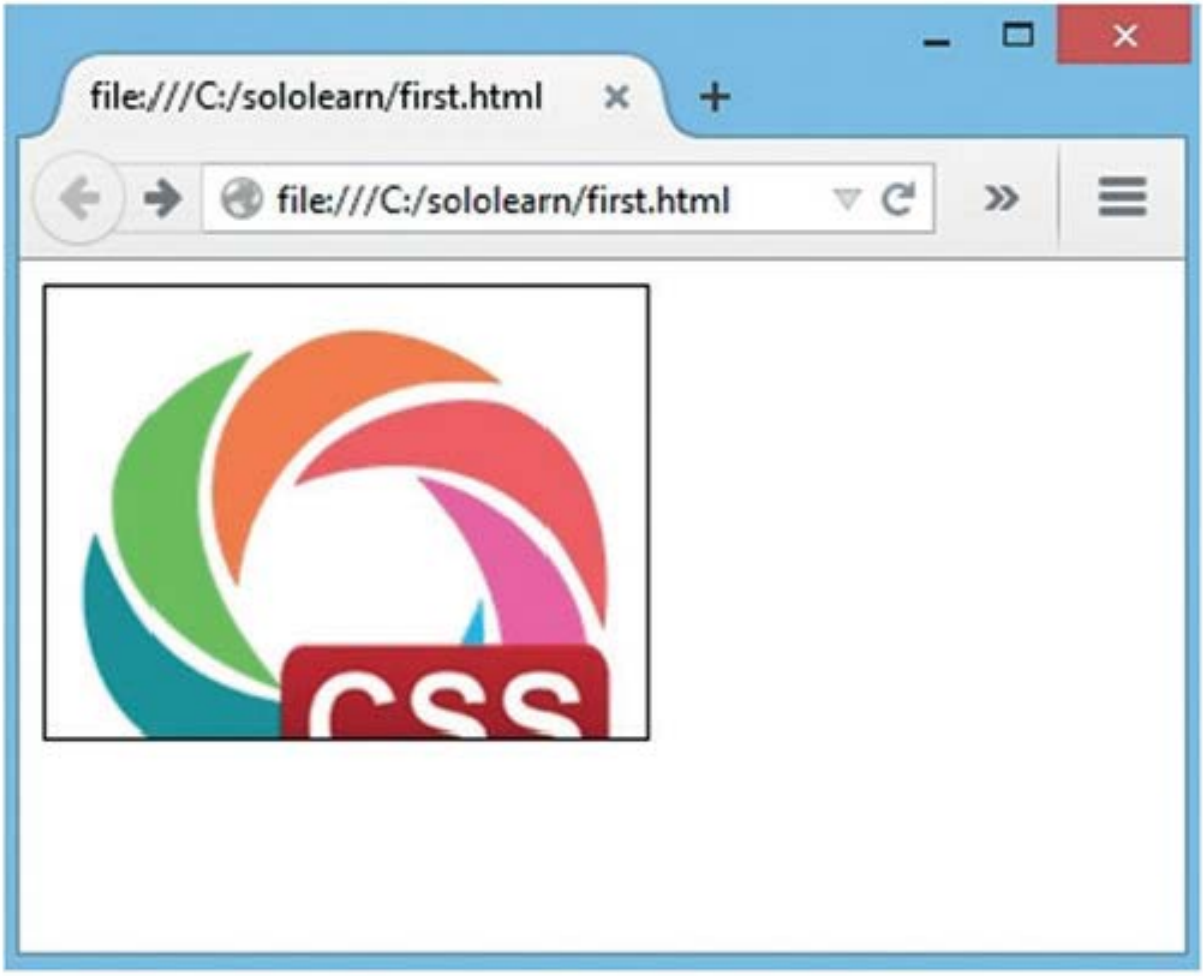


background-size also accepts the **cover** keyword. The image is scaled to fit the entire container; however, if that has a different aspect ratio, the image will be cropped:

CSS syntax looks like this:

```
background-size: cover;
```

Result:



Tap **Try It Yourself** to play around with the code!

70 COMMENTS

The background-clip Property

The **background-clip** property specifies the painting area of the background.

The property takes three different values:

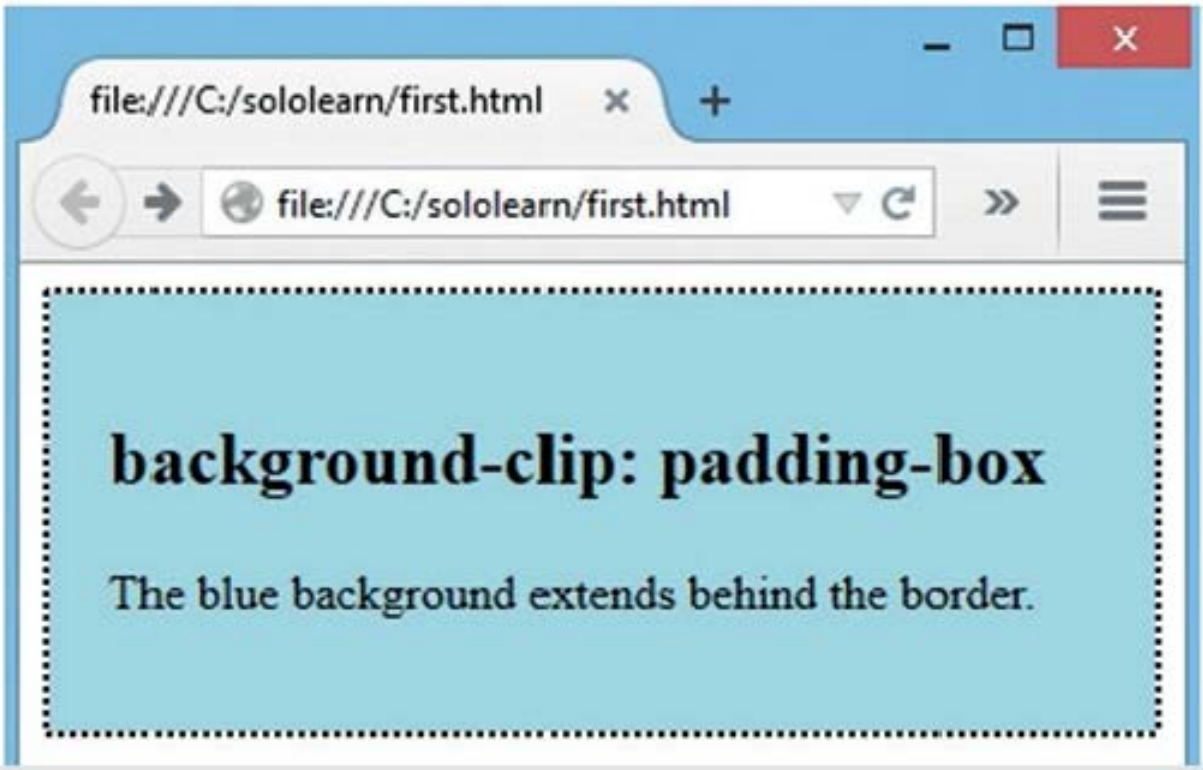
border-box - (default) the background is painted to the outside edge of the border
padding-box - the background is painted to the outside edge of the padding
content-box - the background is painted within the content box

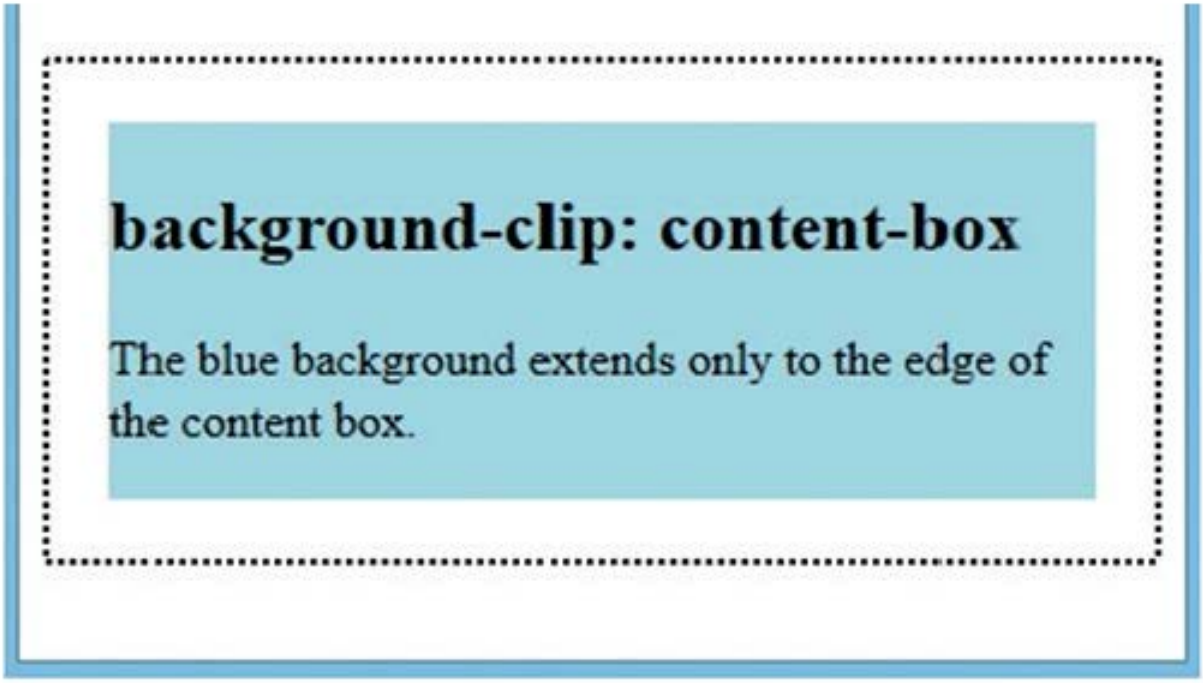
In the example below, the first div with background-clip is set to **padding-box;** in the second div it's set to **content-box**.

```
#first {
  border: 2px dotted black;
  padding: 20px;
  background: LightBlue;
  background-clip: padding-box;
}
#second {
  border: 2px dotted black;
  padding: 20px;
  background: LightBlue;
  background-clip: content-box;
}
```

Try it Yourself

Result:





Tap **Try It Yourself** to play around with the code!

81 COMMENTS

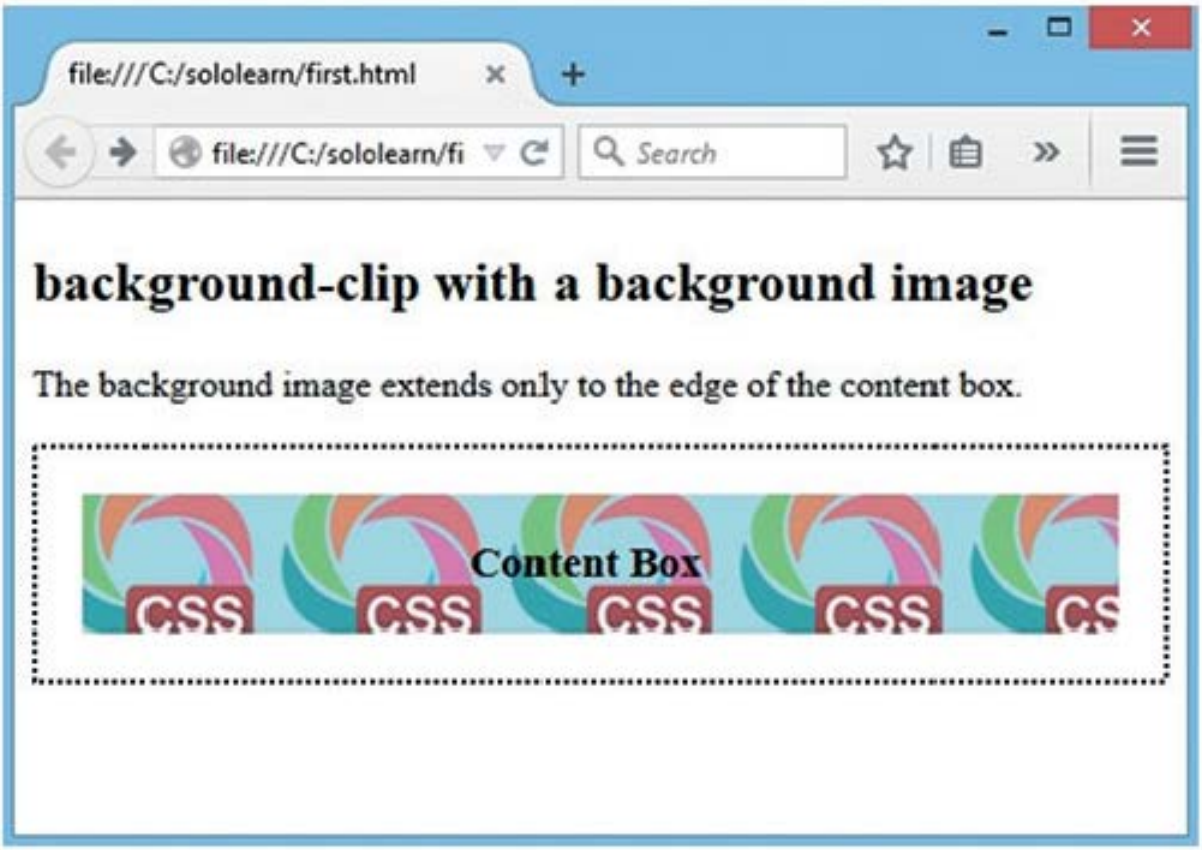
48
Q&A

background-clip with Images

background-clip also applies to background images.
The CSS:

```
div {
  background-image: url("css-logo.png");
  background-clip: content-box;
}
```

Try It Yourself



Tap **Try It Yourself** to play around with the code!

35 COMMENTS

Transparent Borders with background-clip

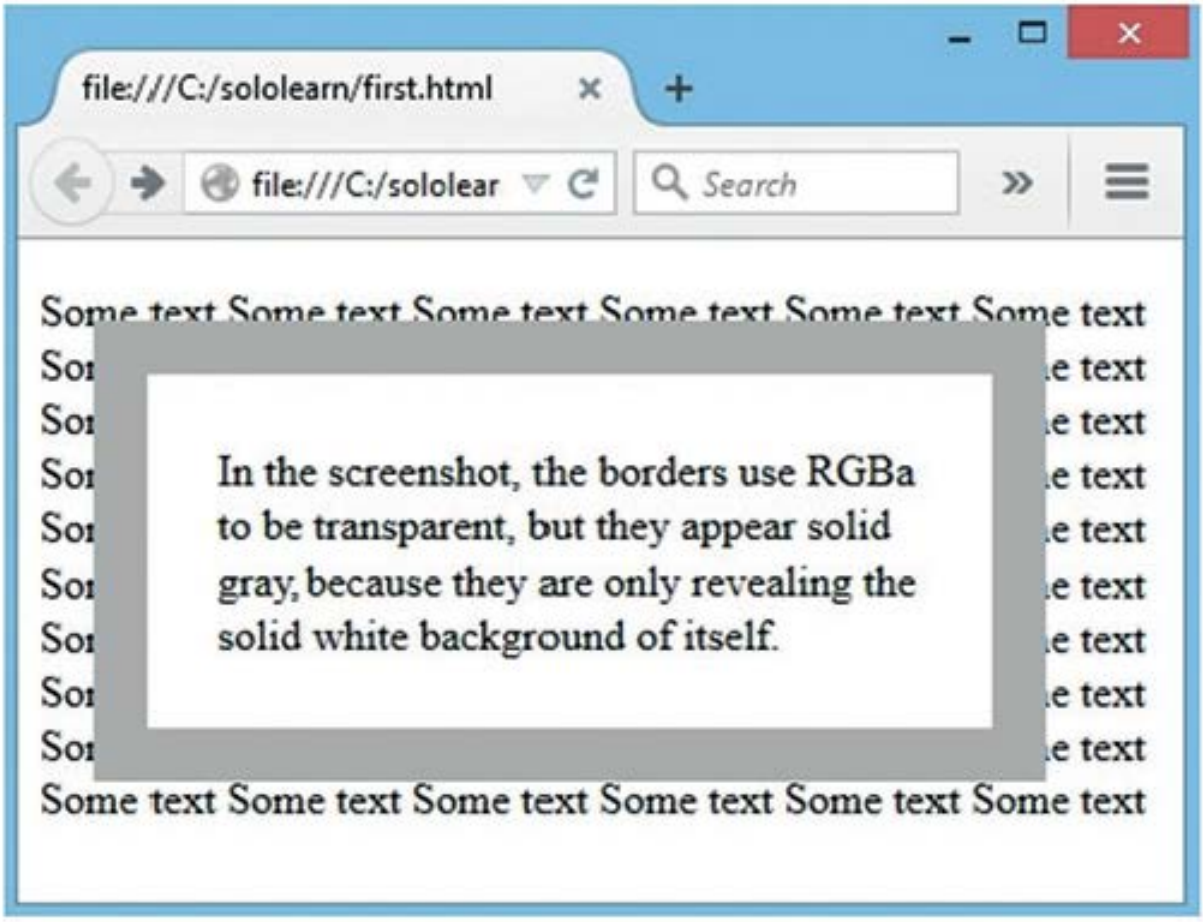
Setting a transparent border on an element will reveal the element's own background under the border.
In the example below, we set the borders to be transparent using RGBA, but they actually appear solid gray.

The CSS:

```
border: 20px solid rgba(0, 0, 0, 0.3);
```

Try It Yourself

Result:



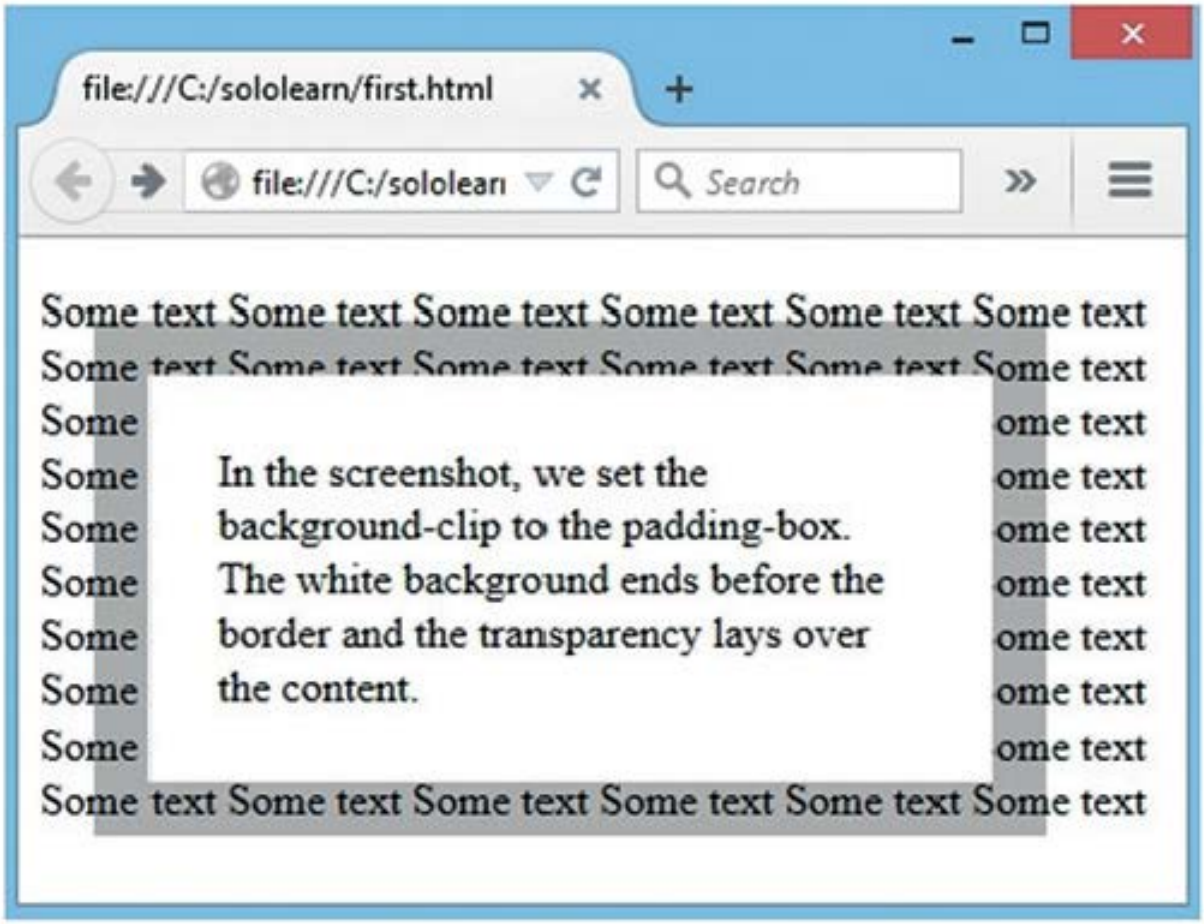
By setting the **background-clip** property to **padding-box**, the borders will be made transparent.

The CSS:

```
border: 20px solid rgba(0, 0, 0, 0.3);
-moz-background-clip: padding-box;
background-clip: padding-box;
```

Try It Yourself

Result:



Transparency effect is achieved with the **background-clip:padding-box**. Without it, the background of the box also goes beneath the borders, making it non-transparent.

101 COMMENTS

Q&A

Multiple Backgrounds

The ability to have multiple background images is a new feature in CSS3.
Multiple background images are specified using a comma-separated list of values for the background-image property. The first image will appear on the top, the last on the bottom.

In the example below, we have two background images: the first is a CSS logo (aligned to the bottom and right); the second is a coding image (aligned to the top-left corner).

The CSS:

```
div {
  width: 400px;
  height: 300px;
  background-image: url(csslogo.png), url(csscode.jpg);
  background-position: right bottom, left top;
  background-repeat: no-repeat;
}
```

Try It Yourself

Result:

Multiple Backgrounds

The position of the background images can be changed, using the background-position property. For example:

```
div {
  width: 400px;
  height: 300px;
  background-image: url(csslogo.png), url(csscode.jpg);
  background-position: right top, left top;
  background-repeat: no-repeat;
}
```

Try It Yourself

Result:

Multiple backgrounds can also be specified using the **background:** shorthand property.

The opacity Property

CSS **opacity** property provides an excellent means of adding opacity to any element.
In the example below, we set different levels of opacity to the same picture, so you can clearly see the difference.

```
#img1 {
  opacity: 1;
}
#img2 {
  opacity: 0.5;
}
#img3 {
  opacity: 0.25;
}
```

Try It Yourself

Result:

The opacity property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

152 COMMENTS

Q&A

Opacity in Internet Explorer

To have the opacity property work in all versions of IE, use the **filter:alpha(opacity=x)** along with the opacity property. x can take a value from 0 to 100.
The value 0 results in a completely transparent element (i.e., 100% transparent), whereas the value 100 makes the element completely opaque (i.e., 0% transparent).

For example, in order to have the code work properly with IE, when the opacity of the image is set at 0.5, it should look like this:

```
#img {
  opacity: 0.5;
  filter: alpha(opacity=50);
}
```

Try It Yourself

The alpha filter is a Microsoft-only property, not a standard CSS property.

145 COMMENTS

Transitions & Transforms

CSS3 Transitions

CSS3 transitions allow us to change from one property value to another over a given duration.
transition-property - specifies the property to be transitioned
transition-duration - specifies the duration over which transitions should occur
transition-timing-function - specifies how the pace of the transition changes over its duration
transition-delay - specifies a delay (in seconds) for the transition effect

In the example below, we set the transition property to **transform**, with a **duration** of 5 seconds, and with an **ease-in** timing function that specifies a transition effect with a slow start.

```
transition: transform 5s ease-in;
```

Transition effects can be applied to a wide variety of CSS properties, including **background-color**, **width**, **height**, **opacity**, and many more.

126 COMMENTS

The Transition Property

In the example below, the div element has width and height of 50px, with a green background. We specified a transition effect for the width property, with a duration of 3 seconds:

The CSS will look like this:

```
div {
  width: 50px;
  height: 50px;
  background: #32CD32;
  transition: width 3s;
}
div:hover {
  width: 250px;
}
```

Try It Yourself

If you hover over the div element, it will move from left to right.

When the cursor is moused out of the element, it will gradually change back to its original style.

198 COMMENTS

Q&A

transition-timing-function

The **transition-timing-function** property specifies the speed curve of the transition effect.
It can have the following values:
ease - the animation starts slowly, then accelerates quickly.
ease-in - starts slowly, then accelerates, and stops abruptly.
ease-out - starts quickly, but decelerates to a stop.
ease-in-out - similar to ease, but with more subtle acceleration and deceleration.
linear - constant speed throughout the animation; often best for color or opacity changes.

Finally, we have **cubic-bezier()**, which allows you to define **your own values in the cubic-bezier function**. Possible values are numeric values from 0 to 1.

```
transition-timing-function: cubic-bezier(0,0,1,1);
```

Try It Yourself

If no timing function is specified, the **default** value is **ease**.

87 COMMENTS

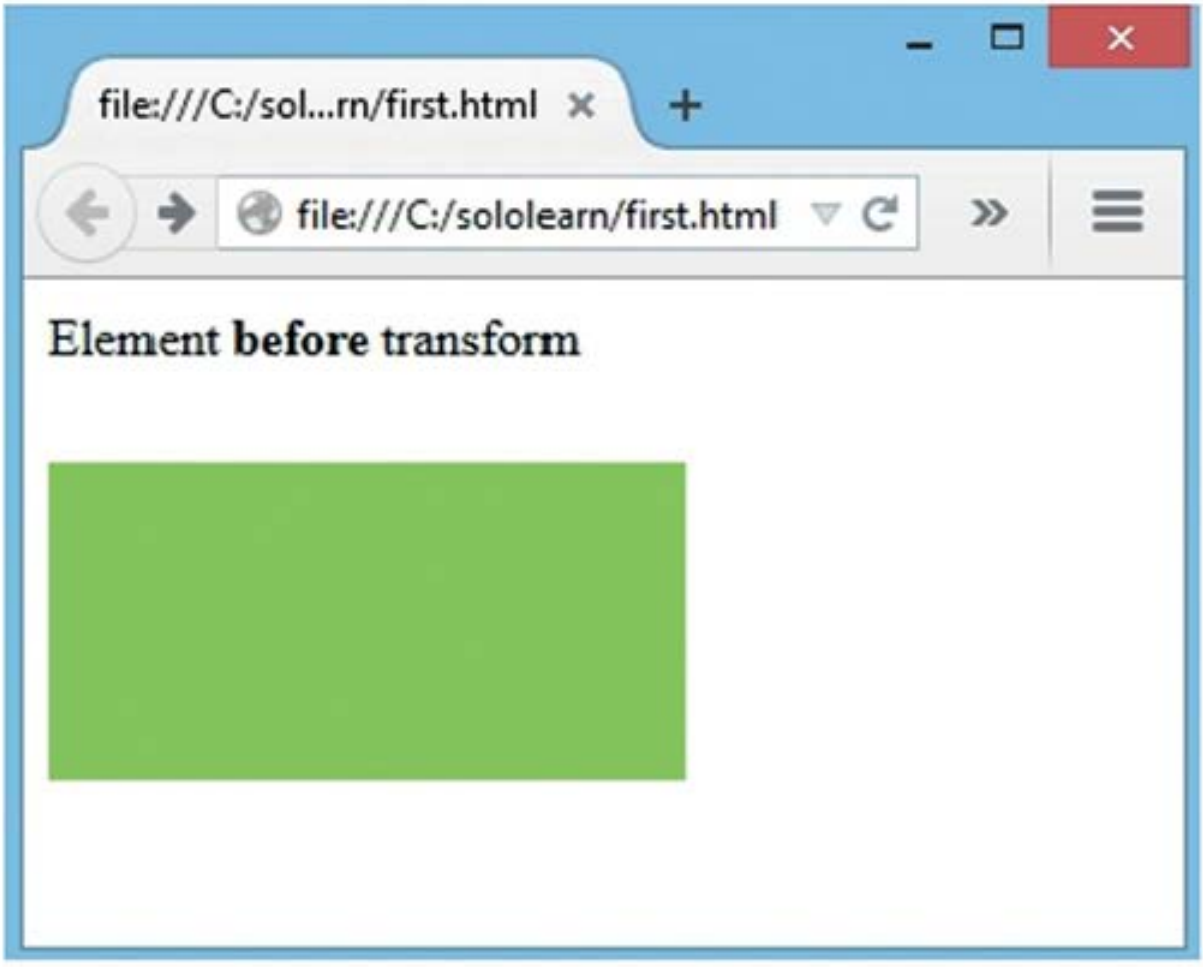
CSS3 Transforms

CSS3 transforms allow you to translate, rotate, scale, and skew elements.
A transformation is an effect that lets an element change shape, size, and position.
CSS3 supports 2D and 3D transformations. Let's take a look at the rotate transformation:

```
div {
  width: 200px;
  height: 100px;
  margin-top: 30px;
  background-color: #32CD32;
}
```

Try It Yourself

The div element before the transform will look like this:

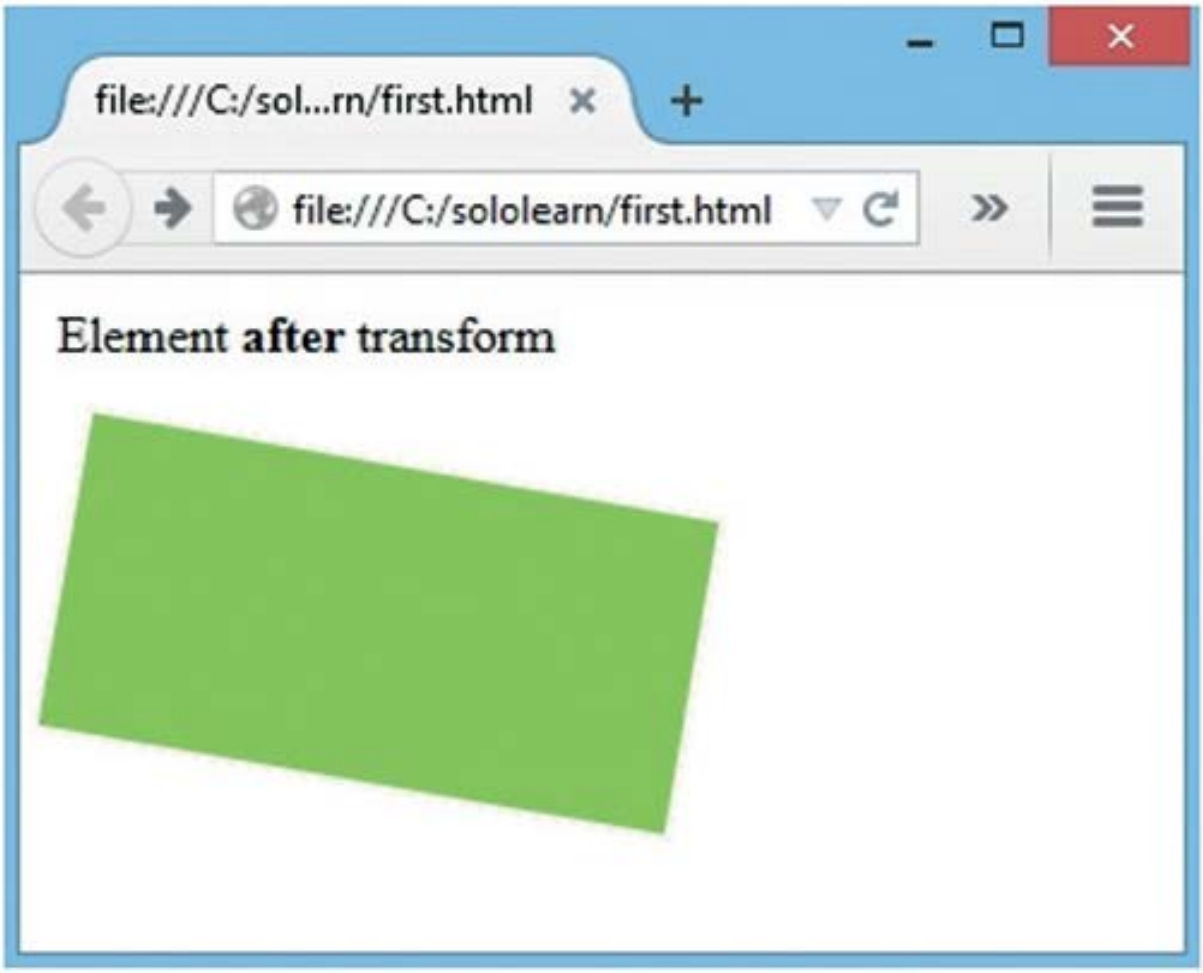


Now let's apply the div element to rotate by **10deg**:

```
div {
  width: 200px;
  height: 100px;
  margin-top: 30px;
  background-color: #32CD32;
  transform: rotate(10deg);
}
```

Try It Yourself

And here is the result:



The rotate() method rotates an element clockwise or counter-clockwise, according to a given degree.

Negative value will result in a counter clockwise rotation.

93 COMMENTS

Q&A

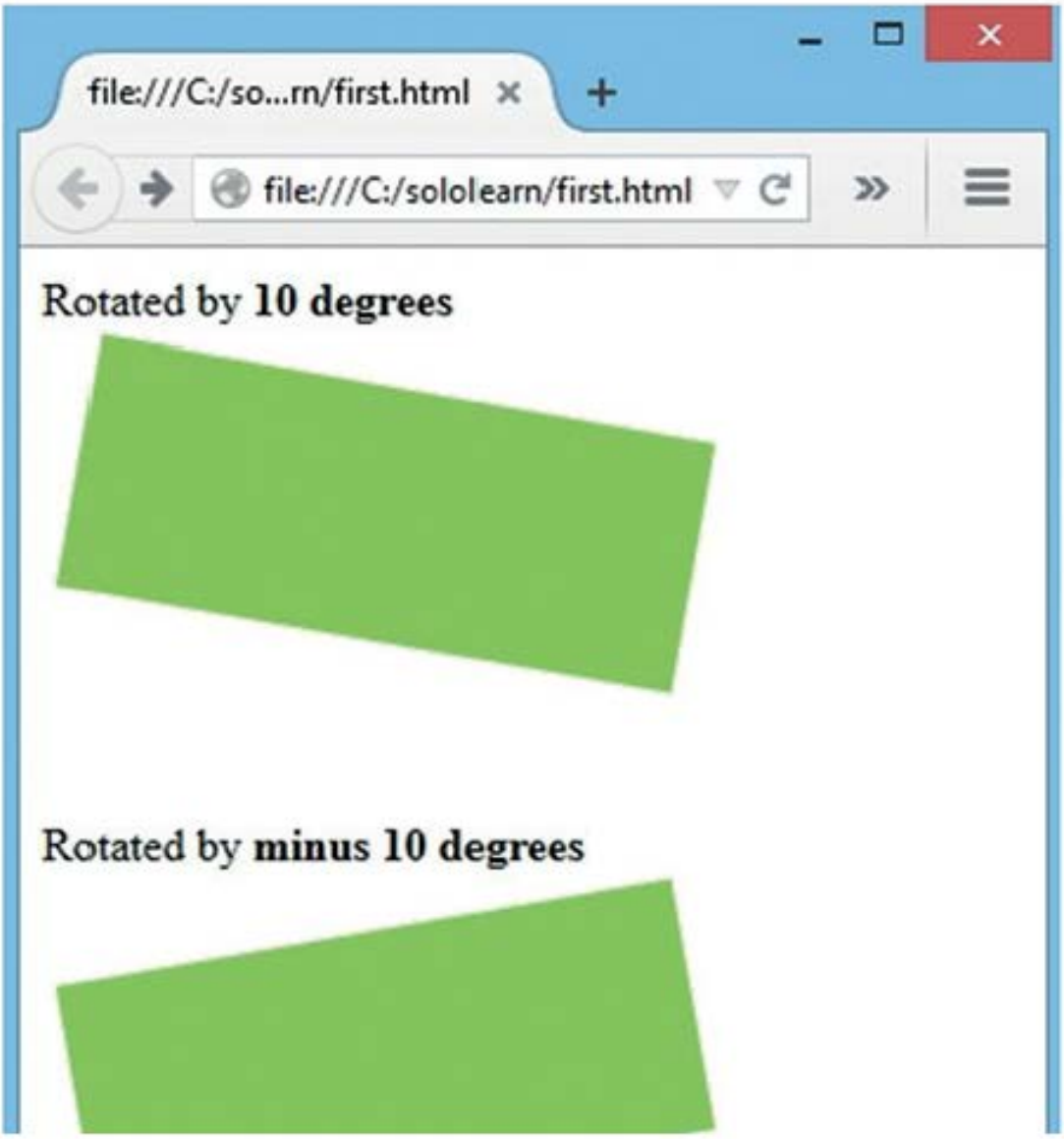
Using Negative Values

As previously mentioned, using a positive value will rotate an element **clockwise**, and using a negative value will rotate the element **counter-clockwise**.

```
div.positive {
  width: 200px;
  height: 100px;
  margin-top: 30px;
  background-color: #32CD32;
  transform: rotate(10deg);
}
div.negative {
  width: 200px;
  height: 100px;
  margin-top: 30px;
  background-color: #32CD32;
  transform: rotate(-10deg);
}
```

Try It Yourself

Result:



transform-origin

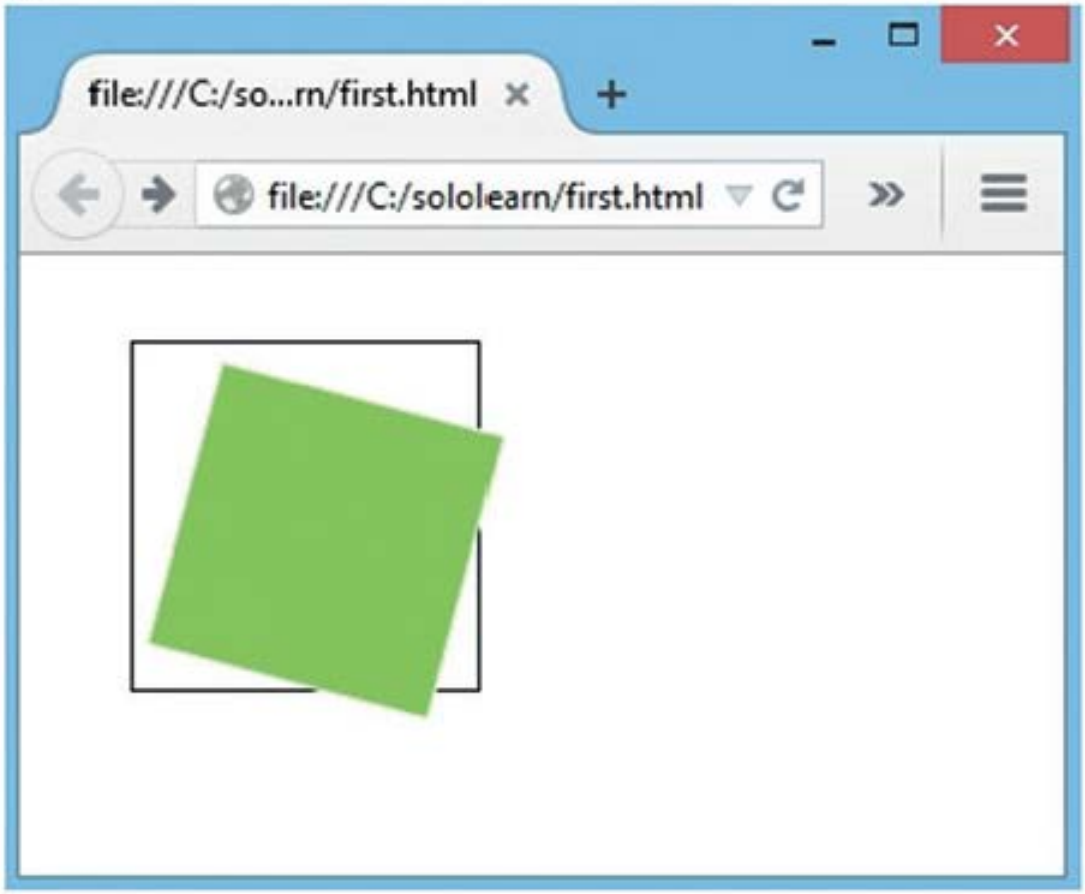
The **transform-origin** property allows you to change the position of transformed elements. The default value for the property is 50% 50%, which corresponds to the center of the element.
In the example below, we use the **transform-origin** property together with **transform-rotate**. The origin of the x-axis (horizontal) is set to 25% from the left. The origin for the y-axis (vertical) is set to 75% from above.

The CSS:

```
div.empty-div {
  position: relative;
  height: 100px;
  width: 100px;
  margin: 30px;
  padding: 10px;
  border: 1px solid black;
}
div.green-div {
  padding: 50px;
  position: absolute;
  background-color: #8bc34a;
  border: 1px solid white;
  transform: rotate(15deg);
  transform-origin: 25% 75%;
}
```

Try It Yourself

Result:



The translate() Method

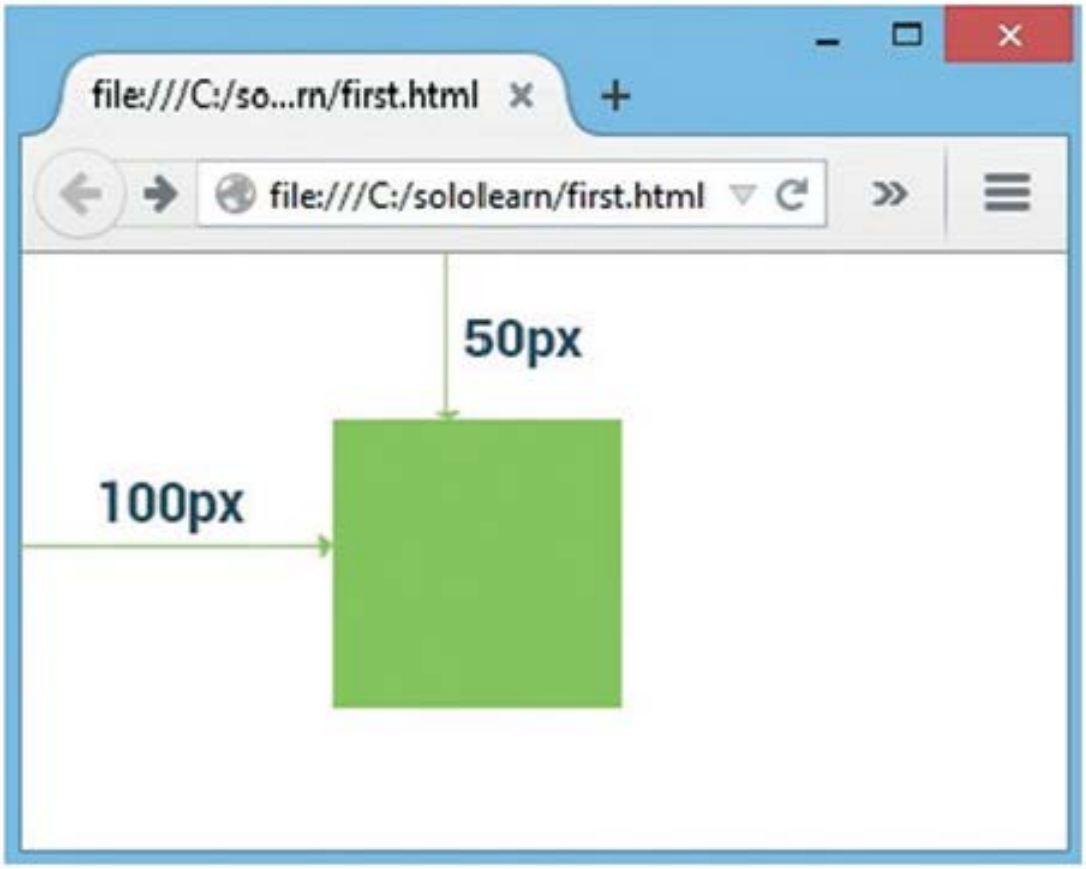
The **translate()** method moves an element from its current position (according to the parameters given for the x-axis and the y-axis). Positive values will push an element down and to the right of its default position, while negative values will pull an element up and to the left of its default position.

In this example below, the div element is moved **100px to the right** and **50px down**:

```
div {
  padding: 50px;
  position: absolute;
  background-color: #32CD32;
  transform:translate(100px, 50px);
}
```

Try It Yourself

Result:



An element can also be moved by setting the margins or by positioning the element, although translate is a better choice for animating elements.

94 COMMENTS

Q&A

The skew() Method

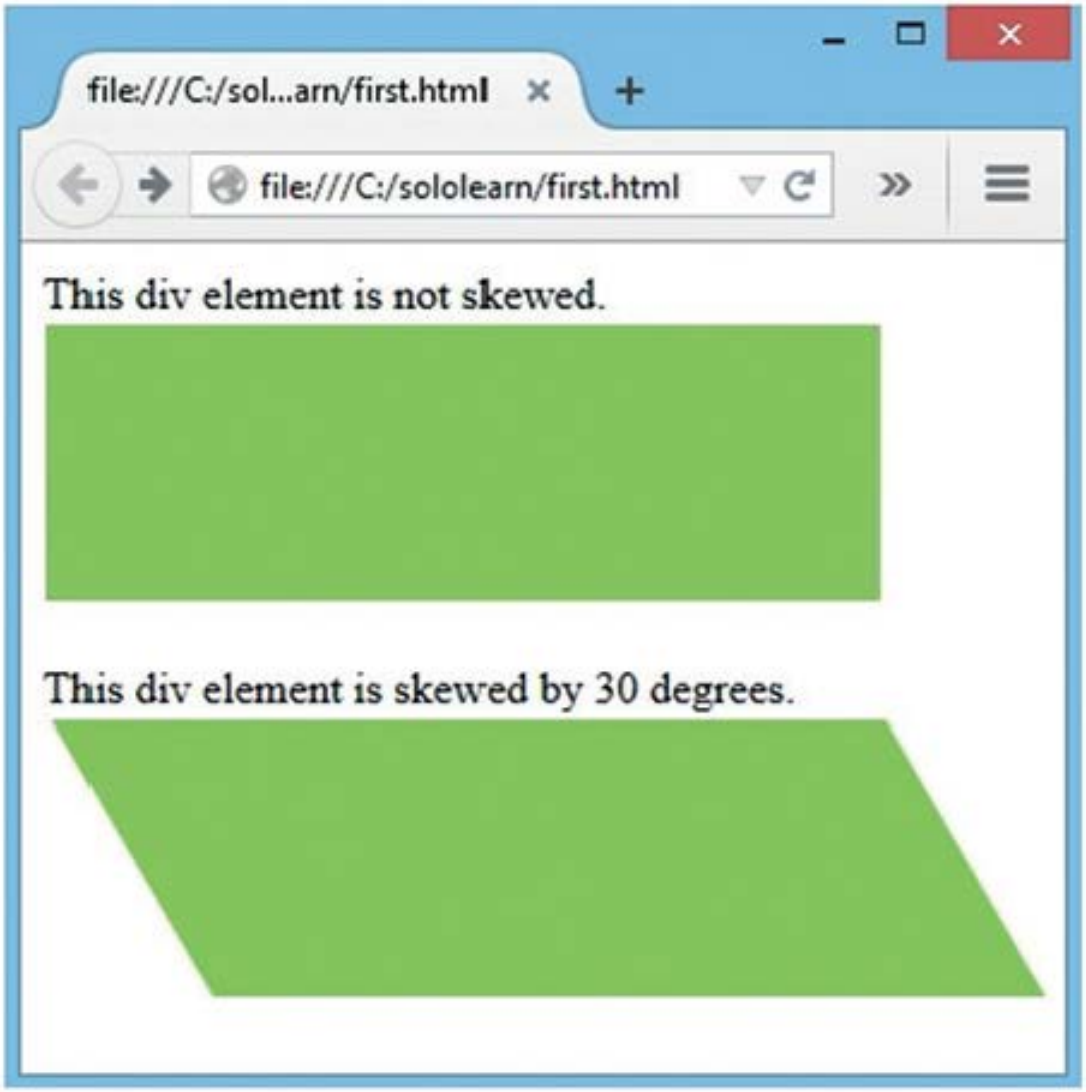
The **skew()** method skews an element along the x-axis and the y-axis by the given angles.

The following example skews the element by 30 degrees along the X-axis:

```
transform: skew(30deg);
```

Try It Yourself

Result:



If the second parameter is not specified, it has a zero value.

66 COMMENTS

The scale() Method

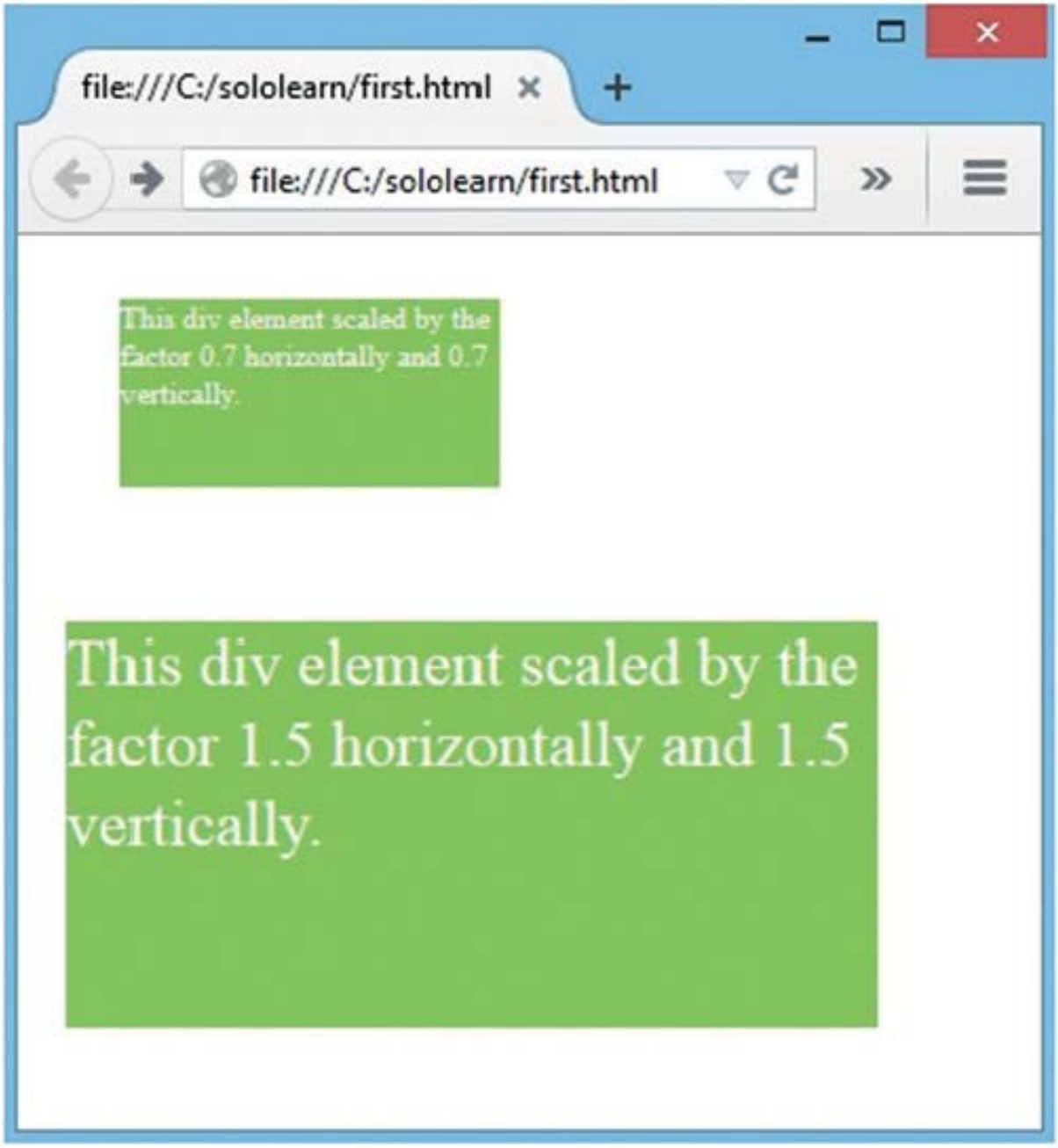
The **scale()** method increases or decreases the size of an element, according to the parameters given for the width and height. 1 stands for the original size, 2 for twice the original size, and so on.

In the example below, we decreased the first div by the factor 0.7 both horizontally and vertically, and increased the second div by a factor of 1.5 horizontally and vertically.

```
div.first {
  width: 200px;
  height: 100px;
  background-color: #8BC34A;
  transform: scale(0.7, 0.7);
  color:white;
}
div.second {
  margin: 60px;
  width: 200px;
  height: 100px;
  background-color: #8bc34a;
  transform: scale(1.5,1.5);
  color:white;
}
```

Try It Yourself

Result:



If only one parameter is passed to the scale() method, it will apply that factor for both the height and the width.

52 COMMENTS

Multiple Transforms

Multiple transforms can be used at once. Rotating and scaling the size of an element at the same time is an example of that.
Applying multiple transforms to an element is simple; just separate them using **spaces**.

Here's an example with two transforms defined:

```
transform: rotate(45deg) translate(100px);
```

Try It Yourself

Result:

If you use **commas** to separate the functions, none of the functions will be applied, so keep in mind not to use commas.

87 COMMENTS

CSS3 Animations

An animation lets an element gradually change from one style to another.
You can change as many CSS properties as you want to, as many times you want to.
Keyframes hold the styles the element will have at certain times

The @keyframes Rule

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.
To get an animation to work, you must bind the animation to an element.

The following example will change the background color of an element three times: when the animation is 50% complete, 70% complete, and when the animation is 100% complete.

The CSS:

```
@keyframes example {
  0%  {background-color: red;}
  50%  {background-color: yellow;}
  70%  {background-color: blue;}
  100% {background-color: green;}
}
```

example is the name of the animation. You can choose any name for your animation.

112 COMMENTS

The @keyframes Rule

As an alternative to using percentages, you can use **from** and **to** keywords, where:
from is a starting offset of **0%**
to is an ending offset of **100%**.

The two examples below are equivalent, and produce the same result:

```
@keyframes colorchange {
  0% {background-color: red;}
  100% {background-color: green;}
}
```

```
@keyframes colorchange {
  from {background-color: red;}
  to {background-color: green;}
}
```

colorchange is the animation name.

65 COMMENTS

The @keyframes Rule

To get an animation to work, you must bind the animation to an element.
In the example below, the animation lasts one second, and changes the background color of the red div to green and blue.

```
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: colorchange;
  animation-duration: 1s;
}
@keyframes colorchange {
  0% {background-color: red;}
  50% {background-color: green;}
  100% {background-color: blue;}
}
```

Try It Yourself

Result:

The **animation-name** property specifies the animation to be used for the element.
The **animation-duration** property specifies the duration of the selected animation.

> If the animation-duration property is not specified, the animation will have no effect, because the default value is 0.

102 COMMENTS

The animation-name Property

animation-name property defines which animation to use.
In this example, the name of the animation is set to **colorchange,** which matches the defined keyframes.

The CSS:

```
div {
  animation-name: colorchange;
  animation-duration: 5s;
}
@keyframes colorchange {
  from { width: 0px; }
  to { width: 100px; }
}
```

Try It Yourself

The **animation-duration** property specifies the duration of the selected animation in seconds.

> If the animation name does not match any keyframe rule, the animation **will not execute**.

110 COMMENTS

Q&A

Animation Properties

The **animation-timing-function** specifies the speed curve of an animation. It can have the following values:
ease - specifies an animation with a slow start, then fast, then end slowly (this is default)
linear - specifies an animation with the same speed from start to end
ease-in - specifies an animation with a slow start
ease-out - specifies an animation with a slow end
ease-in-out - specifies an animation with a slow start and end
cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

The CSS syntax looks like this:

```
animation-timing-function: linear;
```

Try It Yourself

animation-delay - defines the delay before an animation starts. The CSS syntax looks like this:

```
animation-delay: 2s;
```

Try It Yourself

The animation-delay and animation-duration values can be defined in **seconds (s)** or **milliseconds (ms)**.

77 COMMENTS

More Animation Properties

The **animation-iteration-count** property determines the number of times an animation repeats.
For example, you can set the animation to run **5 times:**

```
animation-iteration-count: 5;
```

Try It Yourself

To make the animation repeat **forever**, just use the **infinite** value:

```
animation-iteration-count: infinite;
```

Try It Yourself

The **animation-direction** indicates how the keyframe should be applied.
The values can be set as:
normal - the default value, which means it plays forward from 0 % to 100%.
reverse - plays the keyframe in an opposite direction from 100 % to 0%
alternate - the animation first runs forward, then backward, then forward.
alternate reverse - the animation first runs backward, then forward, then backward.

> If you use 0 or a negative number for the **animation-iteration-count**, the animation will never start.

60 COMMENTS

animation Property

Consider the following example:

```
div {
  animation-name: colorchange;
  animation-duration: 3s;
  animation-timing-function: ease-in;
  animation-delay: 1s;
  animation-iteration-count: infinite;
  animation-direction: reverse;
}
```

Try It Yourself

A single **animation** property can be used to achieve the same result as the above code:

```
div {
  animation: colorchange 3s ease-in 1s infinite reverse;
}
```

Try It Yourself

The order in which each property is declared in the shorthand declaration is important and cannot be altered, or the animation will not work properly.

91 COMMENTS

3D Transforms

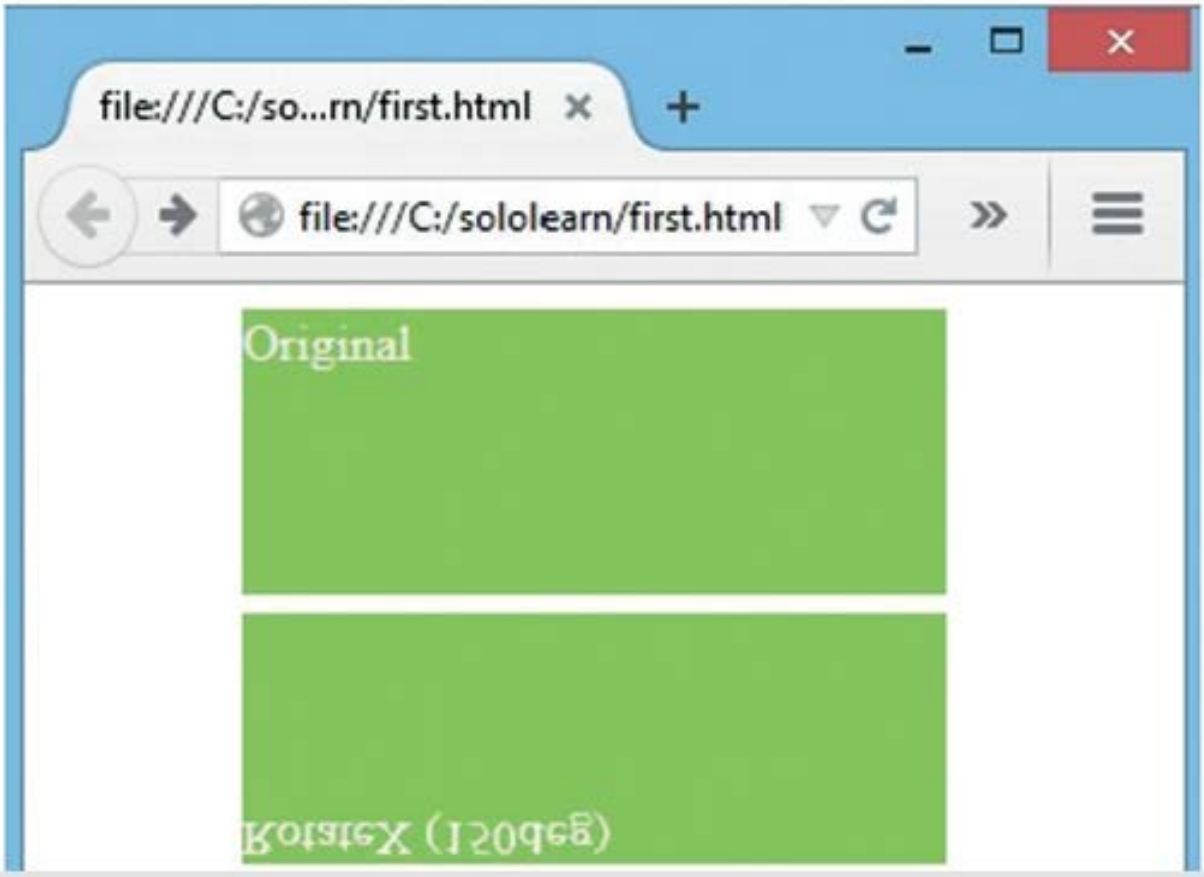
Along with the x and y axes, 3D Transforms introduce the **Z-axis**, which enables 3D manipulations.
3D Transforms are quite similar to 2D Transforms:
rotateX(), rotateY() and rotateZ() rotate an element in 3D space around the corresponding axis at a given degree.

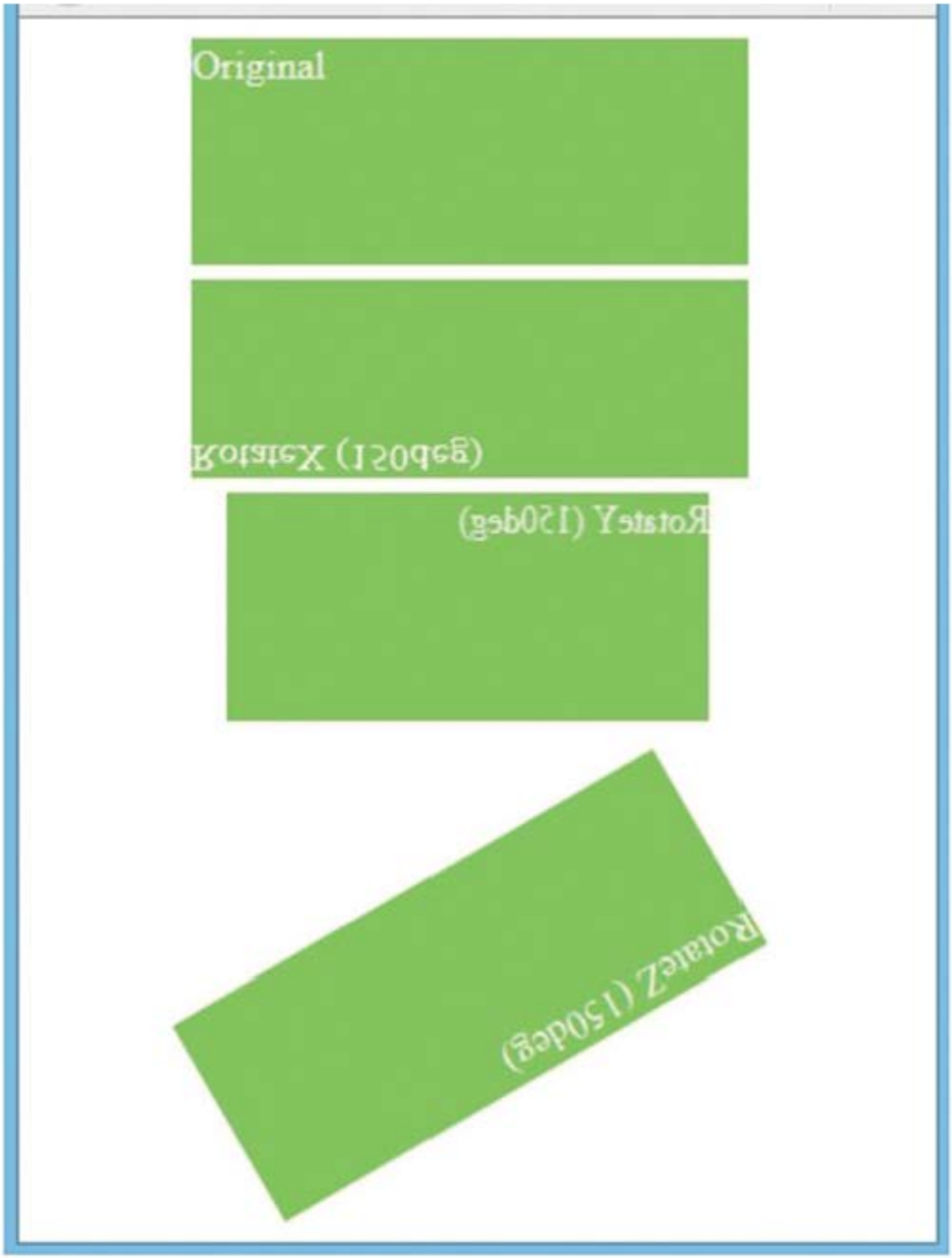
The CSS:

```
div.X {
  transform: rotateX(150deg);
}
div.Y {
  transform: rotateY(150deg);
}
div.Z {
  transform: rotateZ(150deg);
}
```

Try It Yourself

Result:





You can switch off all transformations applied to an element using the none function:
transform: none;

112 COMMENTS

Translations

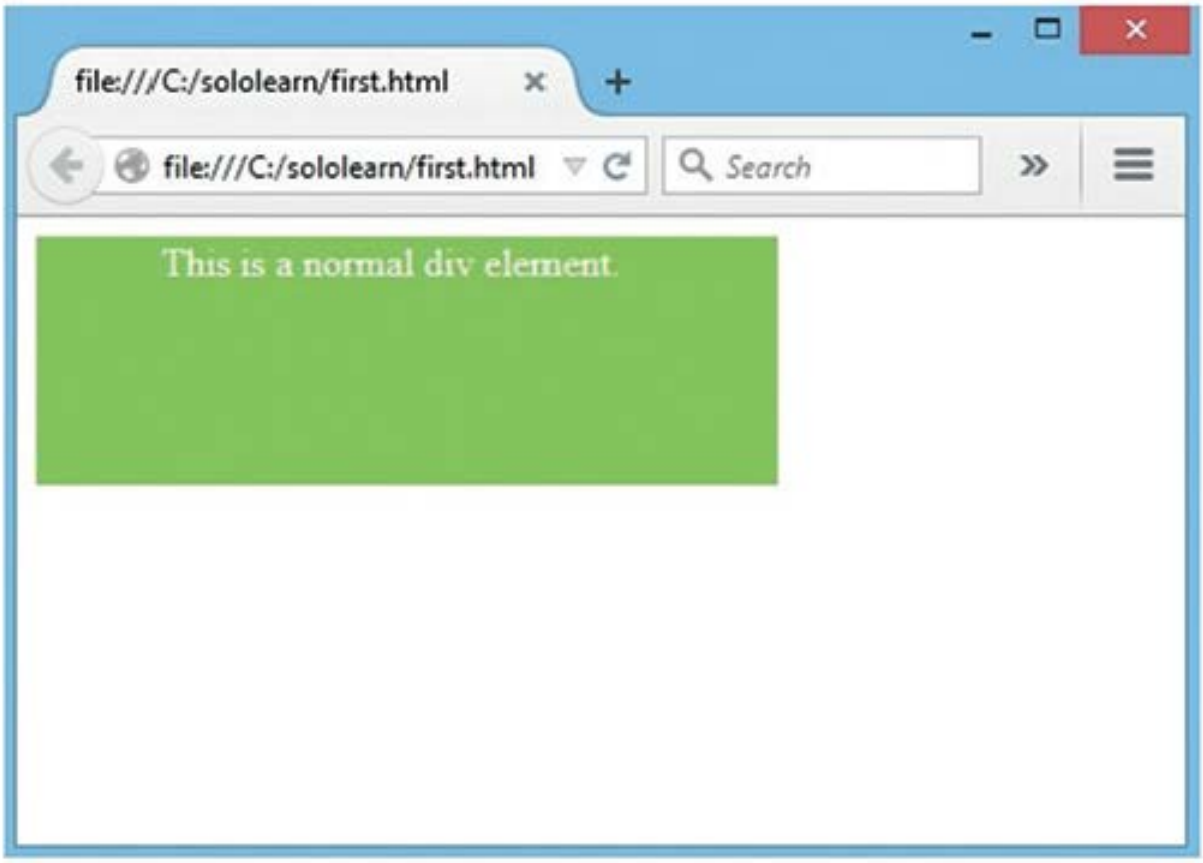
3D translate methods allow you to move the element **horizontally** (translateX), **vertically** (translateY) and **into or out of the screen** (translateZ), using any CSS length units (px, em, %, etc.). Positive values moves the element toward the viewer, negative values away.

The CSS:

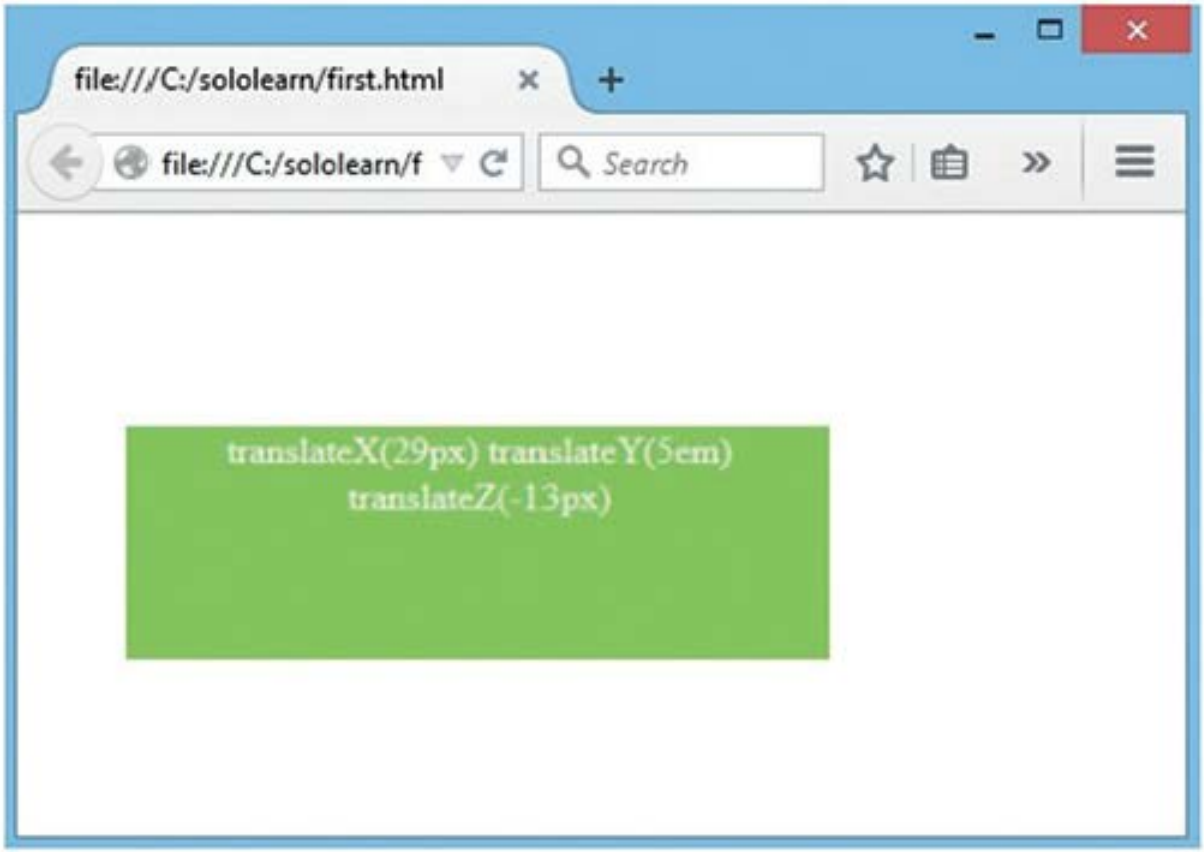
```
#mybox1 {
  transform: translateX(29px)
    translateY(5em)
    translateZ(-13px);
}
```

Try It Yourself

Result (Before translation):



Result (After translation):



The **translate3d()** method allows us to pass the x, y, and z offsets, all at once and in the following order:

```
#mybox1 {
  transform: translate3d(-20px, 4em, 10px);
}
```

Try It Yourself

Like the translate3d() method, there are also **scale3d()** and **rotate3d()**, which are applicable for scaling and rotating elements in 3D.

> Translation of an element is similar to relative positioning - it doesn't affect the document's flow. The translated element will keep its position in the flow and will only appear to have moved.

68 COMMENTS

Perspective

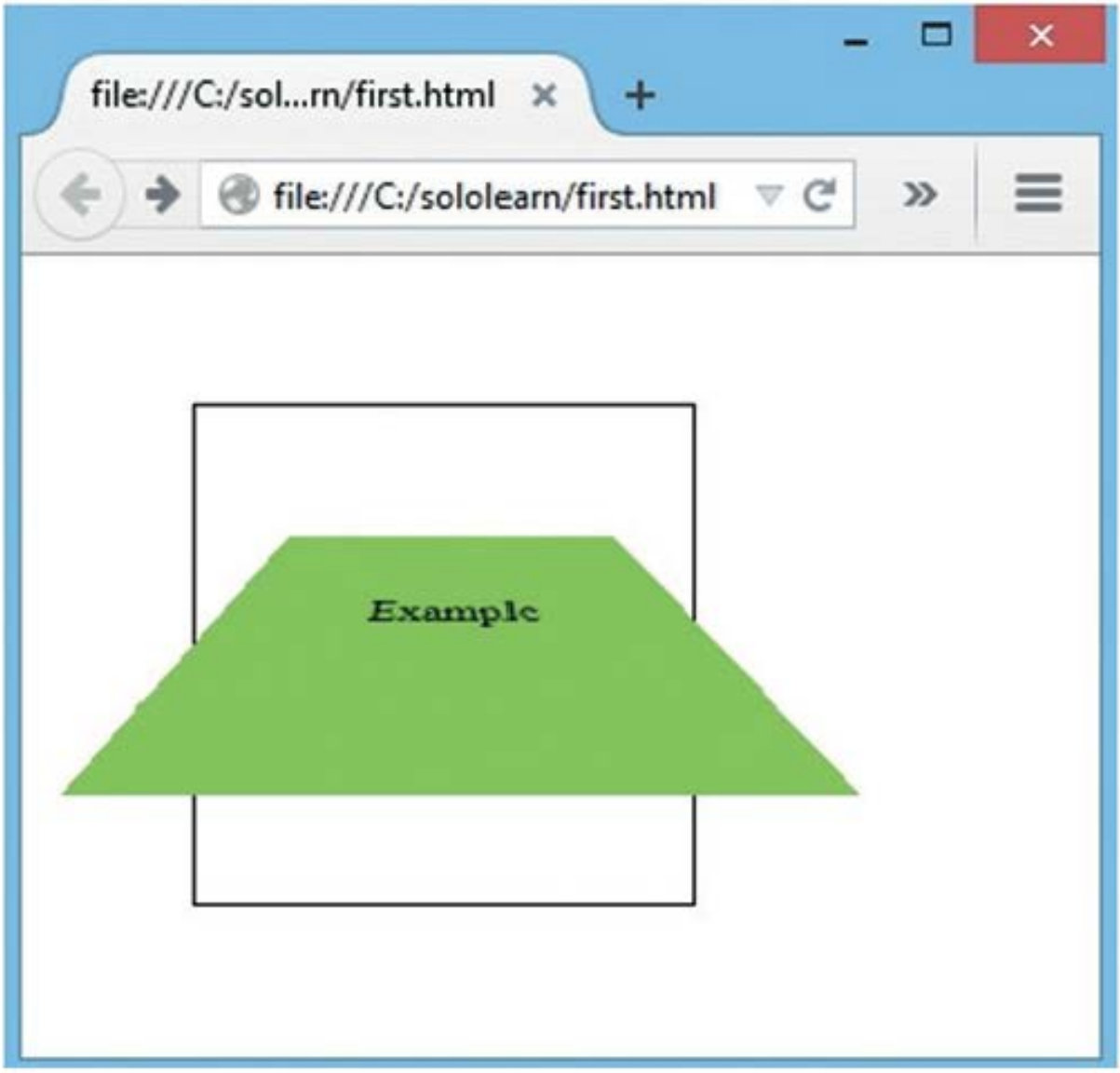
Perspective defines how the depth of the 3D scene is rendered. Think of perspective as a distance from the viewer to the object. The greater the value, the further the distance, so the less intense the visual effect.
When defining the perspective property for an element, it is the ***child*** elements that get the perspective view, ***not*** the element itself.

The CSS:

```
div.empty-div {
  perspective: 100px;
}
div.green-div {
  transform: rotateX(45deg);
}
```

Try It Yourself

Result:



CSS Filters

CSS Filters

The CSS **filter** property lets you apply graphical effects like **blurring** or **color shifting** to an element.
Filters are commonly used to **adjust the rendering** of images, backgrounds, and borders.

Image filtering is useful when you want to have different styling for the same image.
Instead of uploading multiple images to the website, you can upload only one image and then define visual effects using the **filter** property.

Filter functions include **blur()**, **brightness()**, **contrast()**, **drop-shadow()**, **grayscale()**, **hue-rotate()**, **invert()**, **opacity()**, **saturate()** and **sepia()**.

Tap **Continue** to explore the filters and see them in action!

> The filter property is not supported in Internet Explorer, Edge 12, Safari 5.1 and earlier.

66 COMMENTS

The drop-shadow Function

drop-shadow(w h b c) creates a shadow effect that extends beyond an image for the width **w** and height **h** with blur **b** and color **c**.
w, h, and **b** are values in pixels.

For example:

```
.dropshadow {
  filter: drop-shadow(5px 9px 2px blue);
}
```

Try It Yourself

Positive values create the shadow to the **right** and **below** the image.
Negative **width** and **height** values create the shadow **above** and to the **left** of the image.

Tap **Try It Yourself** to play around with the code.

43 COMMENTS

The grayscale Function

The grayscale function converts an image to **grayscale**.
The only parameter defines the **proportion** of the conversion.

0% grayscale is the **original image**, whereas **100%** makes the image **completely grayscale**.

Here is an example using a percentage value to make an image completely grayscale:

```
.filtered {
  filter: grayscale(100%);
}
```

Try It Yourself

Any negative value leaves the image unchanged.

41 COMMENTS

The sepia Function

The **sepia** function converts an image to sepia.
This is similar to using **grayscale** but with a **reddish-brown** color tone.

The idea behind **sepia** filters is that they can make black and white photos look a bit more eye-catching than the basic grayscale version.

0% sepia is the **original image**, whereas **100%** converts the image to **sepia** completely:

```
.filtered {
  filter: sepia(100%);
}
```

Try It Yourself

Tap **Try It Yourself** to play around with the code.

24 COMMENTS

The saturate Function

The **saturate** function controls the **color saturation** for an image. The only parameter determines the **proportion of the saturation** that is applied to the image.
The parameter can be either a percentage value or a number.

0% creates a **completely unsaturated** image (**grayscale**), whereas **100%** is the **original image**.

```
.filtered {
  filter: saturate(50%);
}
```

Try It Yourself

Here is an example using a number value to make an image **super-saturated**:

```
.filtered {
  filter: saturate(2.5);
}
```

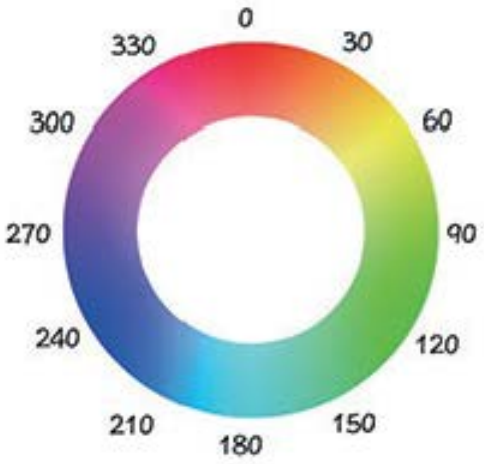
Try It Yourself

The **saturate** function also accepts values over 100%.

34 COMMENTS

The hue-rotate Function

The **hue-rotate** function applies a hue rotation (based on the color circle) to an image.
The function takes an **angle of rotation** as its parameter. The value of angle defines the number of **degrees** around the **color circle** the input samples will be adjusted.



When you use **hue-rotate()** to rotate the **hue**, you are rotating around this color circle.

If the image contains red color, which is at 0 degrees on the color circle, rotating the hue by 240deg will make the red color blue.

Here is an example of a **hue rotation**:

```
.filtered {
  filter: hue-rotate(180deg);
}
```

Try It Yourself

0 and 360deg rotations leave the image unchanged.

Tap **Try It Yourself** to play around with the code.

40 COMMENTS

The invert Function

The **invert** function inverts the colors of an image to make **dark areas bright** and **bright areas dark**.
The function takes the **proportion of the conversion** as its parameter.

The parameter can be either a **percentage value** or a **number**.
0% invert leaves the image unchanged, whereas **100%** creates a **completely inverted** image that is similar to a photographic negative.

Here is an example using a percentage value to make the image **partially inverted**:

```
.filtered {
  filter: invert(70%);
}
```

Try It Yourself

The specification allows values over 100%, but that will have no further effect on the image.

34 COMMENTS

The opacity Function

The **opacity** function sets the opacity of an image to change its transparency.

0% opacity creates a **completely transparent** image, whereas **100%** is the **original image**.
For example:

```
.filtered {
  filter: opacity(70%);
}
```

Try It Yourself

Tap **Try It Yourself** to play around with the code.

33 COMMENTS

Q&A

The brightness Function

The **brightness(amount)** function adjusts the brightness of an image, making it appear brighter or darker.
The **amount** parameter determines the brightness level of the image. The parameter can take either a percentage value or a number.

A value of **0%** results in an image that is **completely black**.
A value of **100%** results in an image that is **unchanged**.
Any amount **over 100%** produces a **brighter image**.

Here is an example using a percentage value to make the image darker:

```
.filtered {
  filter: brightness(50%);
}
```

Try It Yourself

A value under 100% darkens the image, while a value over 100% brightens it.

A number value of 0.5 has the same effect as the percentage value of 50%. A value of 1 is **the same** as 100%.

Here is an example using a number value to make the image brighter:

```
.filtered {
  filter: brightness(1.9);
}
```

Try It Yourself

Any negative value will make the image black.

28 COMMENTS

The contrast Function

The **contrast** function adjusts the contrast of the image.
The **amount** parameter can take either a percentage value or a number.

A value **under 100% decreases** the contrast, while a value over 100% increases it.
A value of **0%** will create an image that is **completely gray**, while a value of 100% leaves the image **unchanged**.
The value **0.5** corresponds to **50%**, while 1 is the same as **100%**.

Here is an example using a percentage value to give the image more contrast:

```
.filtered {
  filter: contrast(140%);
}
```

Try It Yourself

Any negative value leaves the image unchanged.

17 COMMENTS

The blur Function

The **blur** function applies a **blur effect** to an image.
The blur function has only one parameter, **radius**, which defines how many **pixels** on the screen blend into each other. (a larger value creates more blur).
For example:

```
.blured {
  filter: blur(5px);
}
```

Try It Yourself

The parameter is specified as a CSS length, but does not accept percentage values.

If no parameter is provided, then the default value 0 is used, which leaves the image unchanged.
For example:

```
.blur {
  /* Both have no effect on the picture*/
  blur();  /* If no parameter is provided, then a value 0 is used.*/
  blur(0); /* A value of 0 leaves the input unchanged.*/
}
```

Try It Yourself

Tap **Try It Yourself** to play around with the code.

60 COMMENTS

Using Multiple CSS Filters

Multiple CSS filters can be used together by separating them with spaces.
The following code demonstrates the use of the **blur** and the **hue-rotate** functions:

```
.filtered {
  filter: blur(5px) hue-rotate(180deg);
}
```

Try It Yourself

Another Example:

```
.filtered {
  filter: saturate(30%) drop-shadow(5px 9px 2px gray) blur(1px);
}
```

Try It Yourself

Another example from the 19th century:

```
.filtered {
  filter: brightness(150%) sepia(100%);
}
```

Try It Yourself

Tap **Try It Yourself** to play around with the code.
Submit your creative combinations in the comments section below!

119 COMMENTS