Make better Web pages by understanding the CSS box model

Version 1.0
August 15, 2006

By Melonfire

Takeaway

Many novice Web developers use CSS positioning and layout directives without a sound understanding of how they really work. A brief introduction to the box model explains what it is and how you can use it to make better decisions about positioning your HTML elements on a Web page.

CSS

Cascading Style Sheets (CSS) have gradually become the de facto standard for HTML layout and positioning. They're easy to use, don't require any special software, and work uniformly on most major browsers. However, in their very simplicity lies a danger: many novice Web developers use CSS positioning and layout directives without a sound understanding of how they *really* work. When these directives end up producing unexpected results, such novice developers tend to solve the problem through trial and error, rather than an analysis of fundamentals; this usually creates layout that works well on some browsers but "breaks" on others.

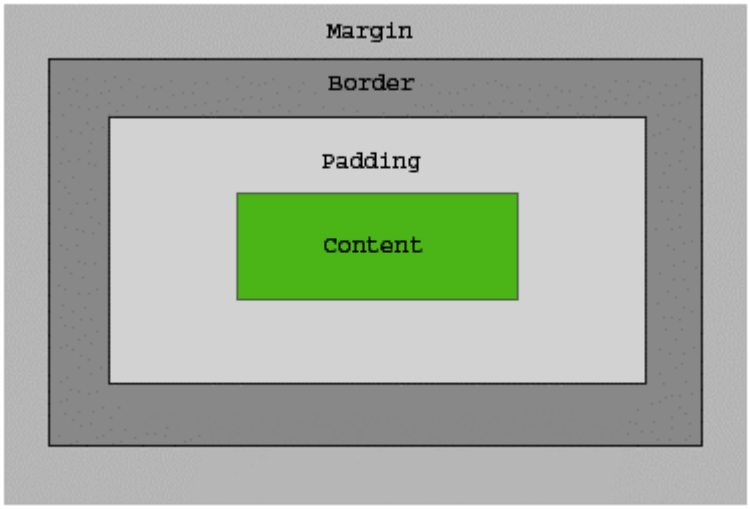
To mitigate this problem, it is worth spending some time to understand some of the core CSS drivers and concepts. And one of the most important concepts a developer can understand is the so-called CSS "box model", which underpins most of CSS layout and positioning. This article provides a brief introduction to this box model, explaining what it is and how you can use it to make better decisions about positioning your HTML elements on a Web page.

The CSS box model represents every element on a Web page in a bounded box which contains four components:

- the element content itself at the core;
- an envelope of padding around the element;
- a border around the padding, which demarcates the visible area of the element;
- a margin around the border.

These four components of the box model can be visually represented as shown in **Figure A**.

Figure A



Components of the CSS box model

From the above, it should be clear that when considering how much space an element takes up on a page, the width and height of the content itself are not sufficient. Rather, one must also consider the width and height of the element's padding, borders and margins. This fact, though obvious on reflection, is one that most novice developers are unaware of and is, in fact, the reason behind many instances of overlapping page elements or elements that do not resize correctly with the rest of the page.

To see how this works in practice, let's work through an illustrative example. Consider (**Listing A**):

Listing A

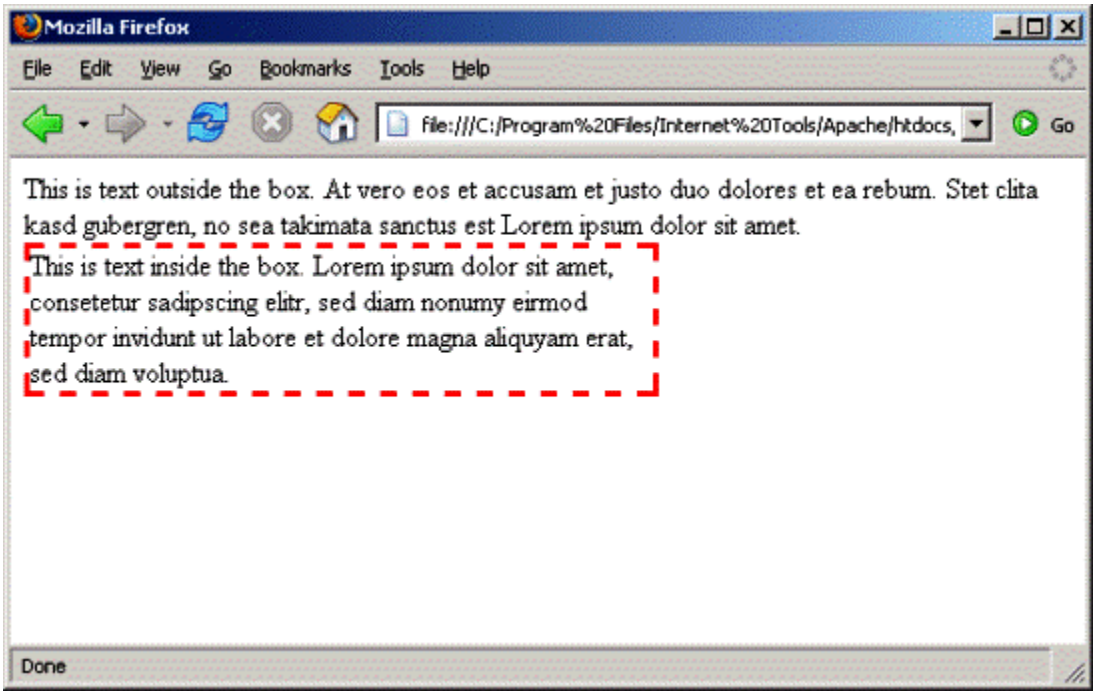
```
<!doctype html public "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<style type="text/css">
#box {
    width: 350px;
    border-color: red;
    border-style: dashed;
}
</style>
</head>
<body>

This is text outside the box. At vero eos et accusam et justo duo dolores et ea
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit
amet.

<div id="box">
This is text inside the box. Lorem ipsum dolor sit amet, consetetur sadipscing
elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam
erat, sed diam voluptua.
</div>
</body>
</html>
```

This is a fairly simple page - two paragraphs of text, one inside a `<div>` and one outside it. The text inside the `<div>` has been assigned a red, dashed border to make the box model easier to understand. The total box width at this point is 350px. Here's what it looks like (**Figure B**).

Figure B



Example 1

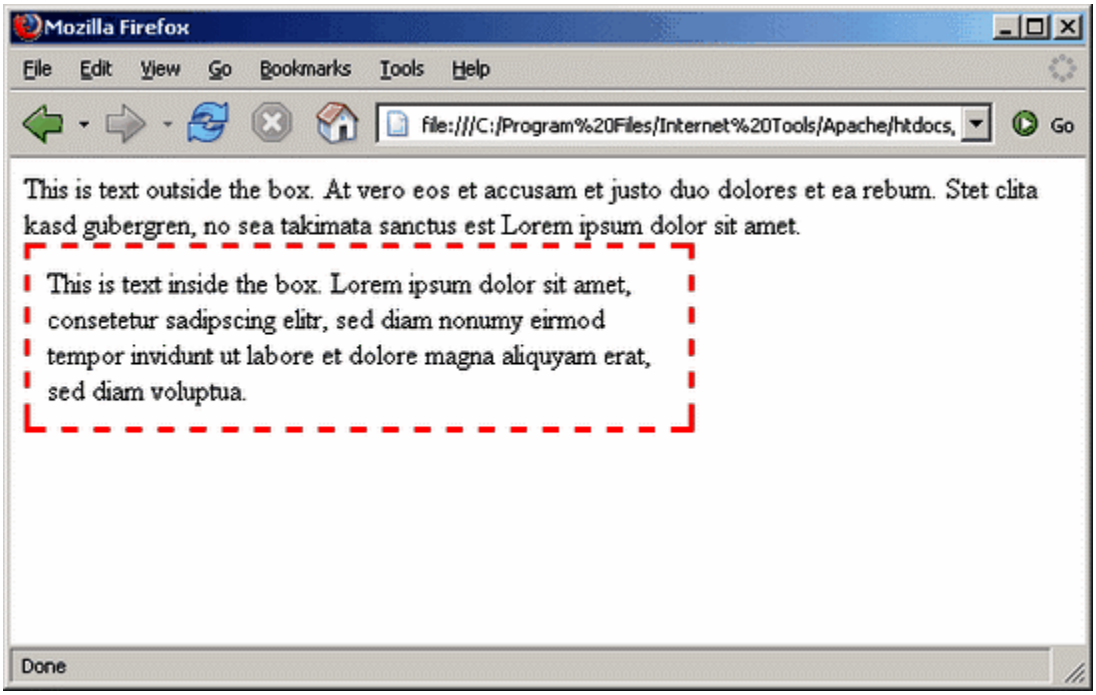
Next, let's add some padding (**Listing B**):

Listing B

```
#box {
    width: 350px;
    border-color: red;
    border-style: dashed;
    padding: 10px;
}
```

Figure C shows you the output.

Figure C



Padding added

As you can see, the additional padding on all four sides of the text has increased the space between the border and the element content by 10px. The total width of the box is now (350+10+10) = 370px.

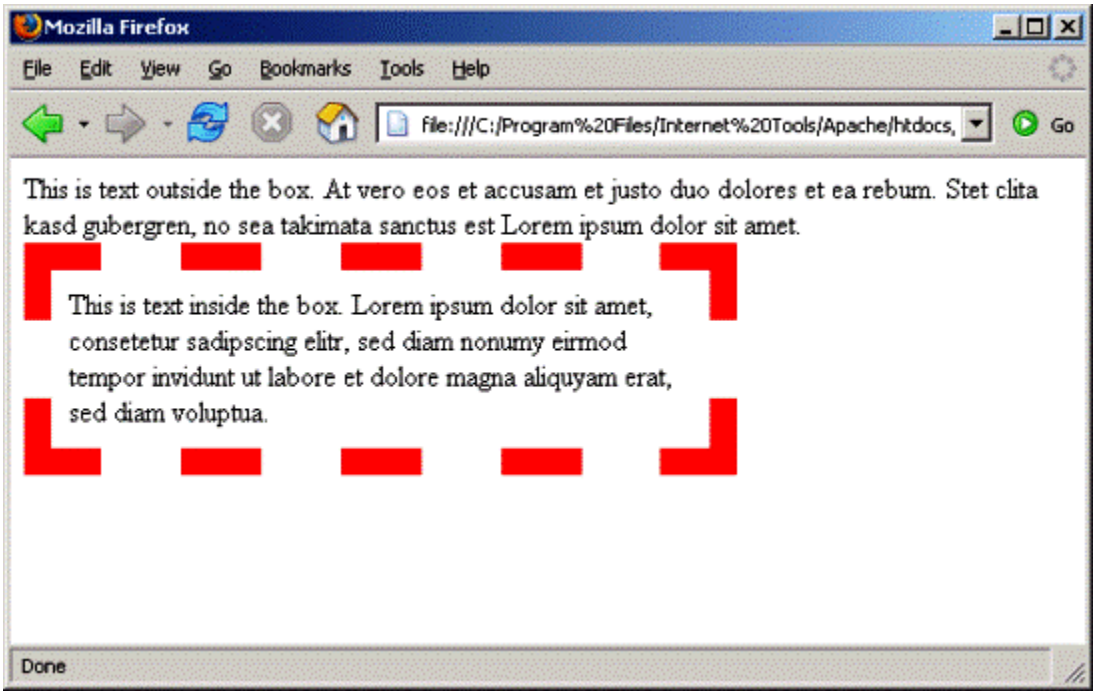
Next, how about increasing the thickness of the border (**Listing C**):

Listing C

```
#box {
    width: 350px;
    border-color: red;
    border-style: dashed;
    padding: 10px;
    border-width: 15px;
}
```

Here's the output (**Figure D**):

Figure D



Thicker border

The difference is immediately visible. The new width of the box is (370+15+15) = 400px.

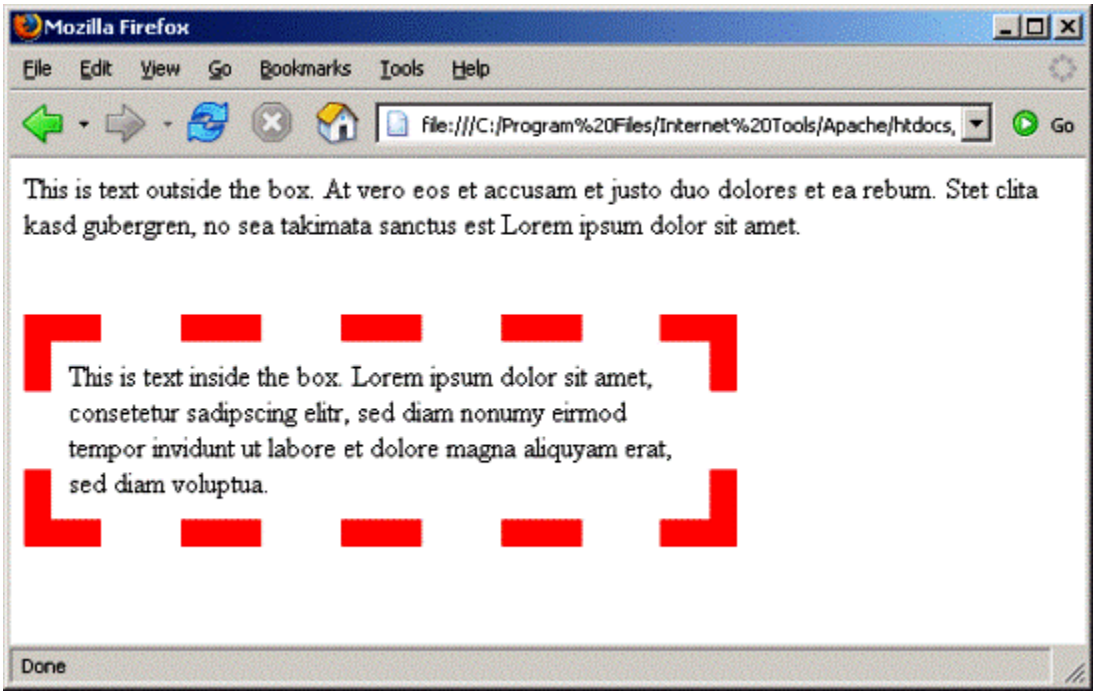
The border demarcates the visible area of the element. Other elements flow around the outer border of the element. If you'd like to increase the spacing between this element's border and others on the page, you must turn to the third layer of the box: margins. Consider the next rule (**Listing D**), which adds a margin of 40px to the top border of the element:

Listing D

```
#box {
    width: 350px;
    border-color: red;
    border-style: dashed;
    padding: 10px;
    border-width: 15px;
    margin-top: 40px;
}
```

In this case, the line of text above the `<div>` will be separated from the outside border of the `<div>` by an extra 40px. **Figure E** shows you what it looks like.

Figure E



Increased top margin

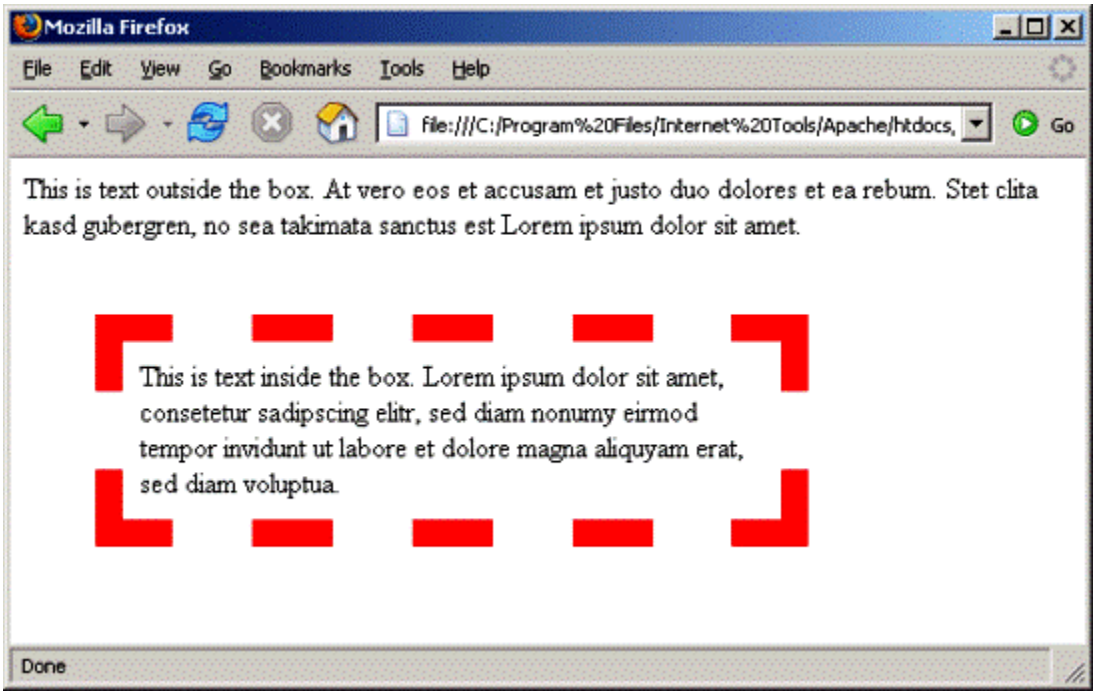
Of course, because the margin's only been added to the top, the width of the box remains 400px. However, the height increased by the size of the top margin. If you'd prefer a uniform margin, consider the following rule (**Listing E**), which adds a margin of 40px on all sides of the box:

Listing E

```
#box {
    width: 350px;
    border-color: red;
    border-style: dashed;
    padding: 10px;
    border-width: 15px;
    margin: 40px;
}
```

And here's the revised output (**Figure F**):

Figure F



Margins on all sides

The total box width in this case would be (400+40+40) = (350+10+10+15+15+40+40) = 480px.

As the examples above illustrate, adding margins, borders and padding to a content fragment can cause it to take up significantly more place on a page than the actual amount of content would warrant. Therefore, when positioning elements on a page, a sound understanding of the box model, and its impact on neighboring elements, goes a long way to helping you lay your page out more accurately.

The above is, of course, only the tip of the iceberg. For more information on the box model, take a look at the official W3C specification. It's also worth noting that most modern browsers render CSS boxes in the manner described above; however, older versions of Netscape and Internet Explorer operate in a non-standard manner and require you to make adjustments to your code in order to have your pages render correctly.

Hopefully you found the above interesting, and now have a better idea of what goes on behind the CSS directives you use so frequently. Drop me a line if you'd like to read more and until then...happy coding!

Additional resources

- TechRepublic's Downloads RSS Feed XML
- Sign up for TechRepublic's Downloads Weekly Update newsletter
- Sign up for TechRepublic's Web Development Zone newsletter
- Check out all of TechRepublic's free newsletters
- Create a photo gallery using CSS
- Put some style in your Web site with CSS and XHTML

Version history

Version: 1.0
Published: August 15, 2006

Tell us what you think

TechRepublic downloads are designed to help you get your job done as painlessly and effectively as possible. Because we're continually looking for ways to improve the usefulness of these tools, we need your feedback. Please take a minute to drop us a line and tell us how well this download worked for you and offer your suggestions for improvement.

Thanks!

—The TechRepublic Downloads Team