

Advanced CSS Training

Browser-compatibility Issues

Lesson 1, Activity 2: Quirks Mode

Modern browsers have at least two modes in which they can display content:

1. Standards-compliant mode
2. Quirks mode

In standards-compliant mode, the browser does its best to display the page according to the latest CSS standards. In quirks mode, the browser displays the page as if it were an older version of the same browser. The purpose of quirks mode is to make pages that were designed for the old browsers display the same in the newer browsers.

You won't find much difference between quirks mode and standards-compliant mode in Mozilla, Opera or Safari, but you may see big differences in Internet Explorer.

Turning Off Quirks Mode

By default, quirks mode is on. It can be turned off by adding the correct DOCTYPE declaration at the top of the HTML page. The following DOCTYPE declarations will turn on standards-compliant mode for all modern browsers:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

One gotcha in Internet Explorer is that quirks mode is always enabled if you include an XML declaration (`<?xml version="1.0"?>`), so we recommend you do not include one.

Quirks mode provides an excellent means of testing how pages will look on different browser versions.

Design Strategy

When designing your pages, we recommend you use the following strategy:

1. Design for the most standards-compliant browsers first (e.g, Firefox).
2. Check your pages in the latest version of Internet Explorer and fix any bugs. We'll discuss the best way to isolate this code shortly.
3. Check your pages in Internet Explorer using quirks mode by temporarily cutting the `DOCTYPE` declaration and fix any bugs.

Depending on your need to support other browsers, you may wish to repeat steps two and three on those browsers as well.

Lesson 1, Activity 3: Internet Explorer Conditional Statements

Internet Explorer provides a way of adding conditional statements that allow the designer to specify which version(s) of the browser should display certain sections. These conditional statements appear as simple HTML comments to other browsers (e.g, Firefox) and are therefore ignored. This can be very useful for addressing CSS bugs in older versions of Internet Explorer without having any impact on other browsers.

The syntax is shown below.

Syntax

```
<!--[if IE 7]>
  HTML Code only shows up on Internet Explorer 7
<![endif]-->
```

As you can see, the beginning `if` statement starts with the beginning of an HTML comment (`<!--`) and the `endif` statement ends with the end of an HTML comment (`-->`), so all other browsers view the whole contents as a comment and do not render any of the contained code.

The code shown above will only be displayed in Internet Explorer 7, but you can use operators to specify multiple versions that will display a block of code. The operators are shown below:

Conditional Operators

Operator	Description	Example
!	not	<code><!--[if ! IE 7]></code>
lt	less than	<code><!--[if lt IE 7]></code>
gt	greater than	<code><!--[if gt IE 7]></code>

lte	less than or equal	<!--[if lte IE 7]>
gte	greater than or equal	<!--[if gte IE 7]>

Note that these conditional comments can not be embedded within CSS. The following is *invalid*:

```
<style type="text/css">
.MyStyle {
  background-color: red;
  <!--[if lt IE 7]>
    background-color: blue;
  <![endif]-->
}
</style>
```

You would instead structure this as follows:

```
<style type="text/css">
.MyStyle {
  background-color: red;
}
</style>

<!--[if lt IE 7]>
<style type="text/css">
.MyStyle {
  background-color: blue;
}
</style>
<![endif]-->
```

The following example shows how you might structure an HTML page using Internet Explorer's conditional comments to deal with IE-specific bugs.

Code Sample:

[BrowserBugs/Demos/ConditionalCommentsStructure.html](#)

```
<!DOCTYPE HTML>
```

```
<html>
<head>
<meta charset="UTF-8">
<head>
<link href="main.css" rel="stylesheet">
<!--[if IE 6]>
<link href="ie6.css" rel="stylesheet">
<![endif]-->
<!--[if lt IE 6]>
<link href="ie5.css" rel="stylesheet">
<![endif]-->

<title>Conditional Comments Structure</title>
</head>
<body>

</body>
</html>
```

The advantage of such a structure is that it separates the CSS hacks from the main style sheet, so main.css should standalone as a nice clean CSS file that works great with the browsers that support the standard well.

In the following section, we'll look at some common browser bugs and use what we have learned in this section to address them.

Lesson 1, Activity 5: Common Browser Bugs / Issues

Positioning Legends

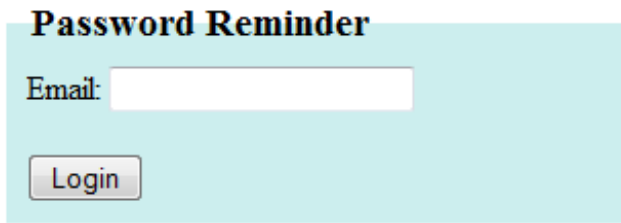
In the [Styling Forms with CSS lesson](#), we pointed out that the positioning of legends gets messed up in Internet Explorer when the border for the fieldset is set to 0. For example, take a look at the simplified form below:

Code Sample:

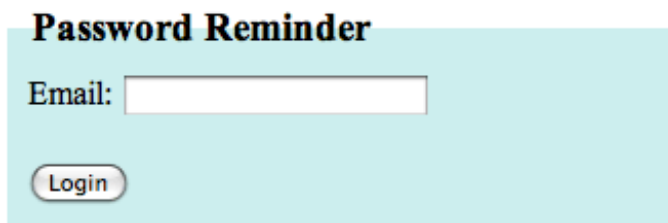
BrowserBugs/Demos/legend.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
fieldset {
  border:0;
  background-color:#cee;
  padding:10px;
}
legend {
  font-weight:bold;
  font-size:1.2em;
}
</style>
<title>Password Reminder</title>
</head>
<body>
<form method="post" action="action.php">
  <fieldset>
    <legend>Password Reminder</legend>
    <label for="Email">Email:</label>
    <input type="text" size="20" name="Email" id="Email" title="Email">
  </fieldset>
  <fieldset id="buttons">
    <input type="submit" id="Submit" name="Submit" value="Login">
  </fieldset>
</form>
</body>
</html>
```

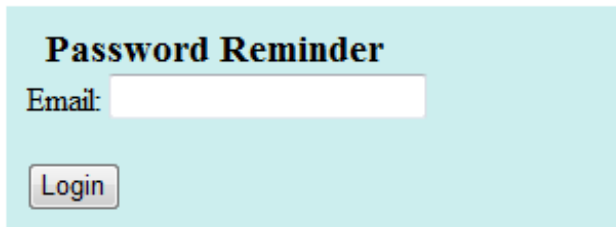
The result is shown below in Firefox 3:



Safari shows a similar result:



But Internet Explorer moves the legend down:



Again, this is a result of the `border: 0` declaration in the rule for `legend`.

This can be fixed using relative positioning, with this code:

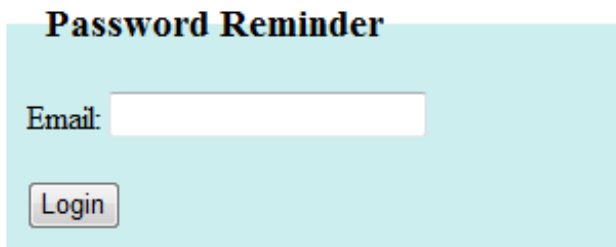
```
legend {  
  position: relative;  
  top: -.6em;  
}
```

But if we put that into the general rule for `legend`, it will mess up other

browsers, so we need to use an Internet Explorer conditional comment:

```
<!--[if IE]>
<style type="text/css">
legend {
  position: relative;
  top: -.6em;
}
</style>
<![endif]-->
```

The result:



And now the legend will look similar in all browsers.

Double-Margin Bug

The double-margin bug manifests itself in Internet Explorer 6 and earlier when an element has a margin in the same direction in which it floats (e.g, a left margin on an element that floats left). In this case, IE 6 and earlier double the margin as shown in the example below.

Code Sample:

[BrowserBugs/Demos/DoubleMargin.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Double Margin</title>
<style type="text/css">
#SideBar {
```

```

float: left;
margin-left: 50px;
width: 200px;
background-color:green;
}

#MainPage {
float: left;
width: 400px;
background-color:yellow;
}
---- C O D E   O M I T T E D ----

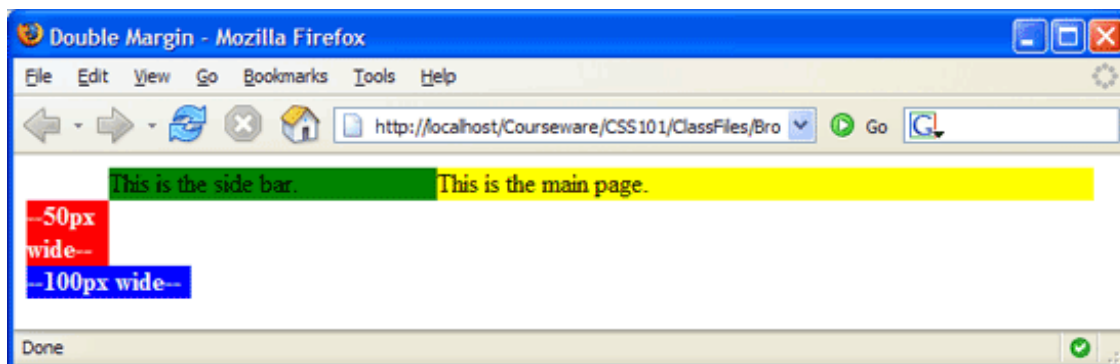
</style>
</head>

<body>
<div id="SideBar">
  This is the side bar.
</div>
<div id="MainPage">
  This is the main page.
</div>

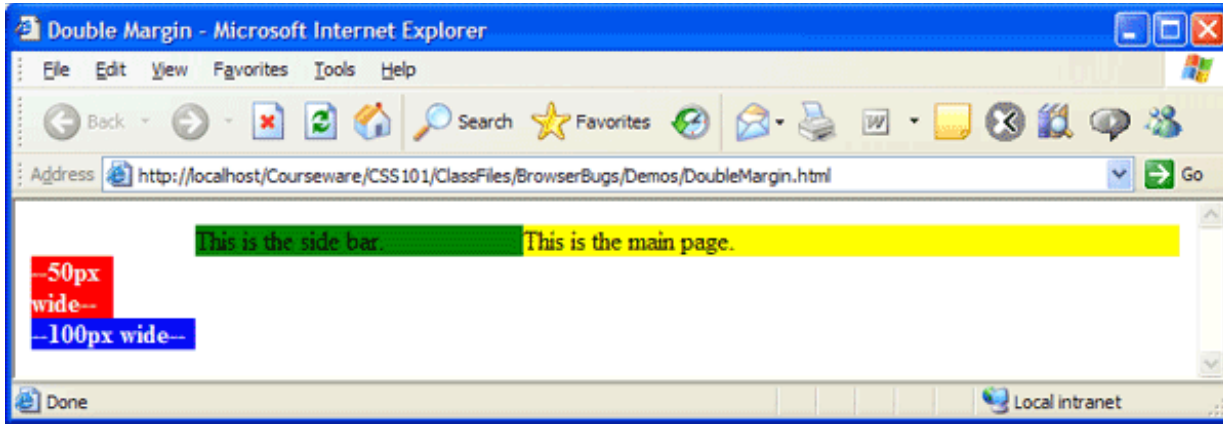
<div id="w50">--50px wide--</div>
<div id="w100">--100px wide--</div>
</body>
</html>

```

The page should display as follows:



But in IE 6 and earlier, it displays as follows:



The fix is very strange: you simply set the `display` property of the offending element (in this case, the sidebar) to "inline". Because all floats are block elements, doing so has no effect on the display in other browsers; however, it does, for some reason, fix the display in Internet Explorer.

Although you could make this fix directly in the `SideBar` rule, we recommend separating it out into an IE-specific stylesheet with conditional comments as shown below.

Code Sample:

[BrowserBugs/Demos/DoubleMargin-Fixed.html](http://localhost/Courseware/CSS101/ClassFiles/BrowserBugs/Demos/DoubleMargin-Fixed.html)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Double Margin</title>
<style type="text/css">
#SideBar {
  float: left;
  margin-left: 50px;
  width: 200px;
  background-color:green;
}
---- C O D E   O M I T T E D ----
</style>
```

```

<!--[if IE]>
<style type="text/css">
#SideBar {
    display:inline;
}
</style>
<![endif]-->
</head>

<body>
<div id="SideBar">
    This is the side bar.
</div>
<div id="MainPage">
    This is the main page.
</div>

<div id="w50">--50px wide--</div>
<div id="w100">--100px wide--</div>
</body>
</html>

```

Float-Width Bug

When a floating element is followed by a block-level element with a width defined, Internet Explorer incorrectly redefines the content area for the block-level element. To understand this, let's first look at how it is supposed to work.

Code Sample:

BrowserBugs/Demos/Float.html

```

<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Float</title>
<style type="text/css">
#SideBar {
    float: left;
    margin-left: 50px;
    width: 200px;
    border:1px solid green;
    display:inline;

```

```

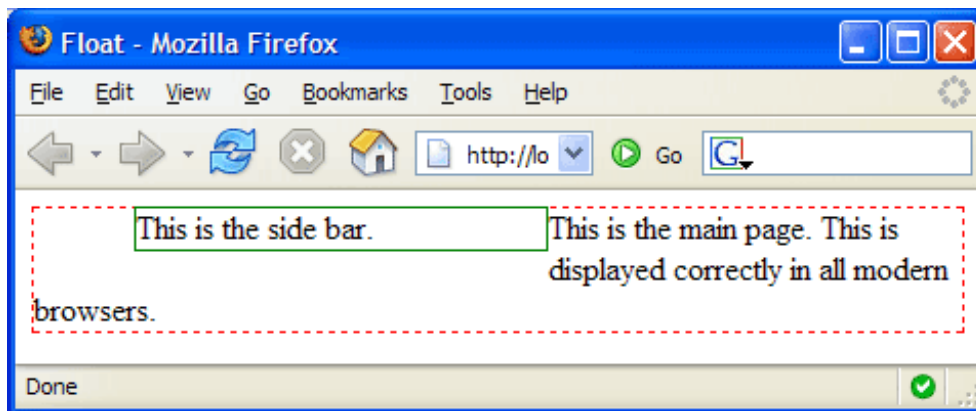
}

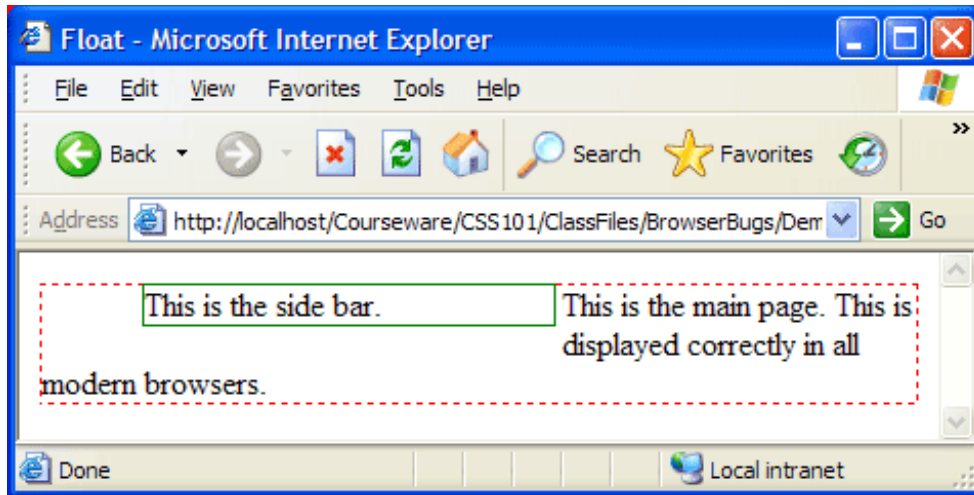
#MainPage {
  border:1px dashed red;
}
</style>
</head>

<body>
<div id="SideBar">
  This is the side bar.
</div>
<div id="MainPage">
  This is the main page. This is displayed correctly in all modern browsers.
</div>
</body>
</html>

```

Block-level elements that come after the float are supposed to simply ignore the float; however, their inline content should not pass behind the float. The result is shown in Firefox and Internet Explorer below.

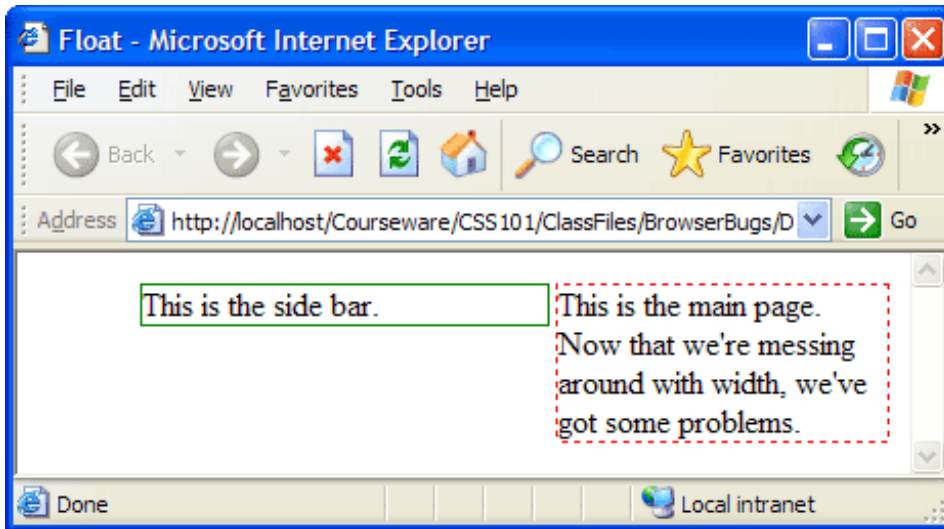




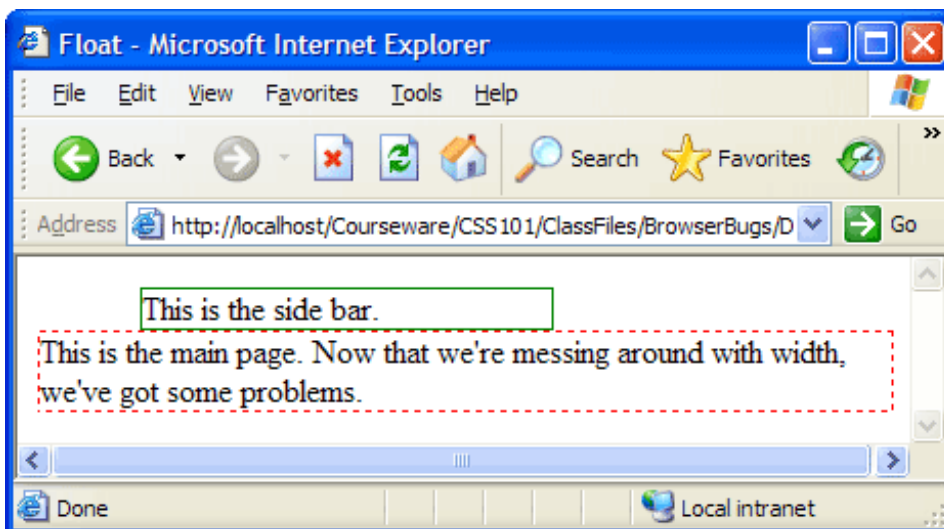
This is handled correctly in both browsers. The problem arises when a width is added to the block-level element. Internet Explorer 5.5 and earlier handle this by changing the available content area to start at the right side of the floating div, rather than the left side of the viewport (the browser window). To illustrate, let's add a width of 100% to the block-level element, which is called MainPage:

```
#MainPage {
  border:1px dashed red;
  width:100%;
}
```

This should not change anything as the default width of an element is always 100%. However, IE 5.5 and earlier will render this page as shown below ([BrowserBugs/Demos/FloatWithWidth-Quirks.html](#)):



Internet Explorer 6 and 7 correctly set the 100% width, but clear the element when the width is set ([BrowserBugs/Demos/FloatWithWidth.html](http://localhost/Courseware/CSS 101/ClassFiles/BrowserBugs/Demos/FloatWithWidth.html)):



There is no simple solution to this problem. One option is to not define the width of the element that follows the float and to handle its positioning using `margin-left` (see [BrowserBugs/Demos/FloatWithMarginLeft.html](http://localhost/Courseware/CSS 101/ClassFiles/BrowserBugs/Demos/FloatWithMarginLeft.html)). Another option is to float the second element as well and then use `margin-left` to create a buffer between it and the preceding element (see

[BrowserBugs/Demos/FloatWithFollowingFloat.html](#)).

3-Pixel Gap Bug

Internet Explorer has a weird bug, in which it adds a three-pixel margin between a floating element and the subsequent element. If the subsequent element has a width defined, Internet Explorer pushes the whole element to the right. If no width is defined, Internet Explorer just pushes the content to the right. Examine the code below.

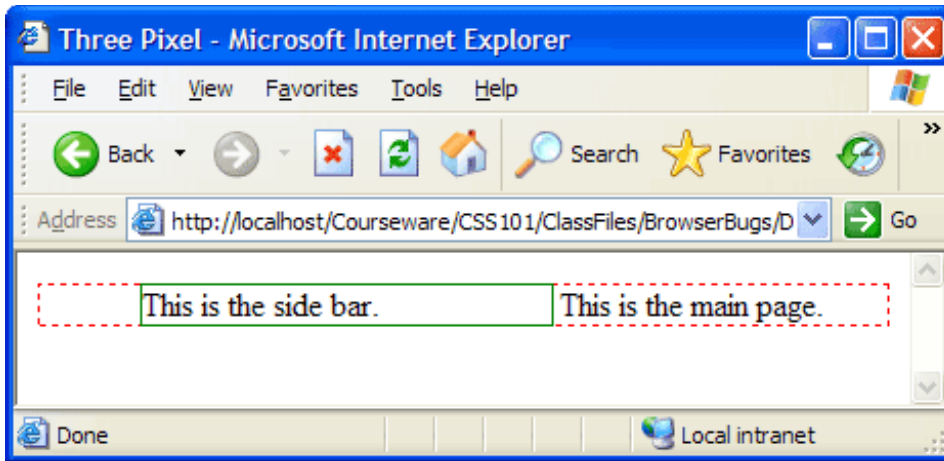
```
#SideBar {
  float: left;
  margin-left: 50px;
  width: 200px;
  border: 1px solid green;
}

#MainPage {
  border: 1px dashed red;
}
```

The page should be displayed with the subsequent text butt up against the floating div as shown below ([BrowserBugs/Demos/ThreePixel.html](#)):



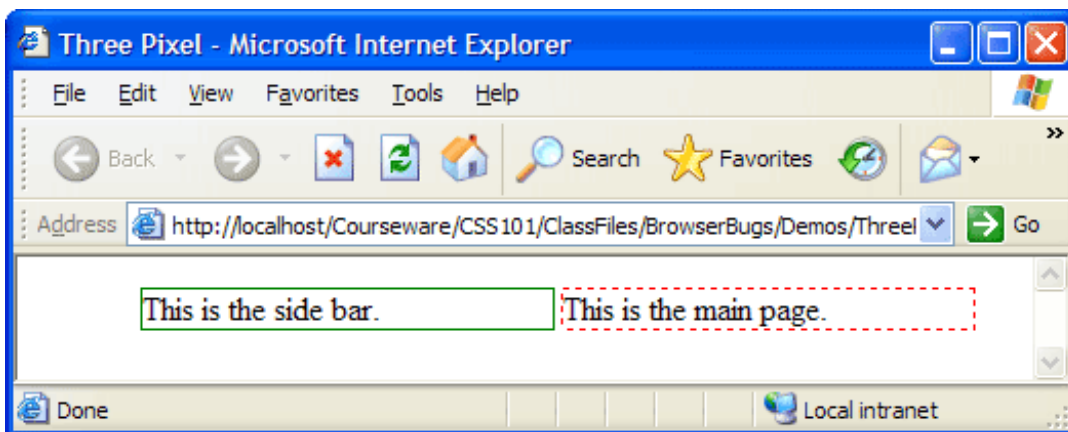
Internet Explorer display it like this:



Notice the tiny amount of added space between the text "This is the main page." and the floating element. Well, that's picky! And really, it doesn't create much of a problem, until you add a width to the subsequent element:

```
#MainPage {
  border:1px dashed red;
  width:200px;
}
```

Now the whole element is pushed over to the right
([BrowserBugs/Demos/ThreePixelWithWidth.html](http://localhost/Courseware/CSS 101/ClassFiles/BrowserBugs/Demos/ThreePixelWithWidth.html)):



This may not be a problem either, except in the case of a very tight layout. Those extra three pixels could cause the MainPage element to

clear down under the float, which would destroy the design.

Luckily, the fix is easy. Simply set the `margin-right` of the floating div to -3 pixels as shown below:

Code Sample:

BrowserBugs/Demos/ThreePixel-fixed.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>Three Pixel Fixed</title>
<style type="text/css">
#SideBar {
    float: left;
    margin-left: 50px;
    width: 200px;
    border:1px solid green;
}

#MainPage {
    border:1px dashed red;
    width:200px;
}
</style>
<!--[if IE]>
<style type="text/css">
#SideBar {
    display:inline;
    margin-right:-3px;
}
</style>
<![endif]-->
</head>

<body>
<div id="SideBar">
    This is the side bar.
</div>
<div id="MainPage">
    This is the main page.
</div>
</body>
</html>
```

IE 5 and 5.5 Box Model Bug

IE 5.5 has very little market share now and most developers don't need to worry about it, but if you do, this is an important bug to know about.

The CSS Box Model requires that an element's padding, borders, and margins are added onto its `width` property. So, an element defined as follows:

```
#Box {  
  width: 200px;  
  padding-left: 40px;  
  padding-right: 40px;  
  border: 10px solid black;  
}
```

would spread 300 pixels from left to right. The `width` property itself only applies to the content area. Internet Explorer 5.5 and earlier handle this incorrectly. The padding and border-width actually eat into the width of the content area. In these browsers, this element would only spread 200 pixels from left to right. We can see this by using quirks mode.

First, take a look at the code below, which simply creates the above rule and then two other rules to reveal pixel width.

Code Sample:

[BrowserBugs/Demos/BoxModel.html](#)

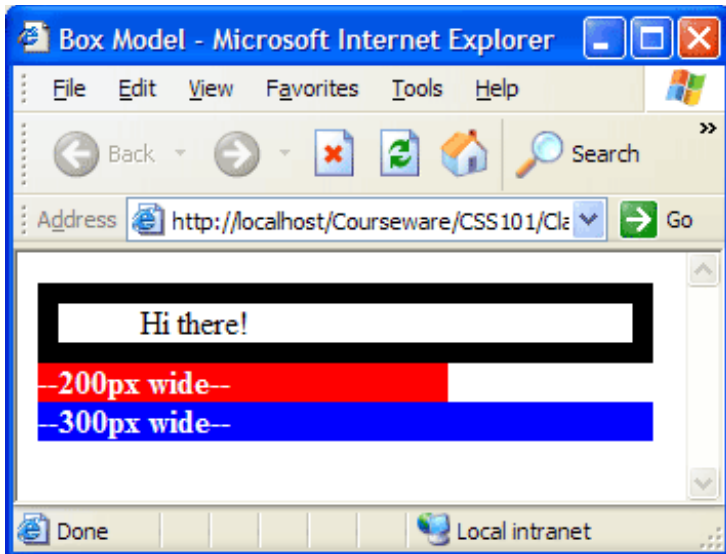
```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<style type="text/css">
#Box {
  width: 200px;
  padding-left: 40px;
  padding-right: 40px;
  border: 10px solid black;
}

#w200 {
  width: 200px;
  background-color:red;
  color:white;
  font-weight:bold;
}

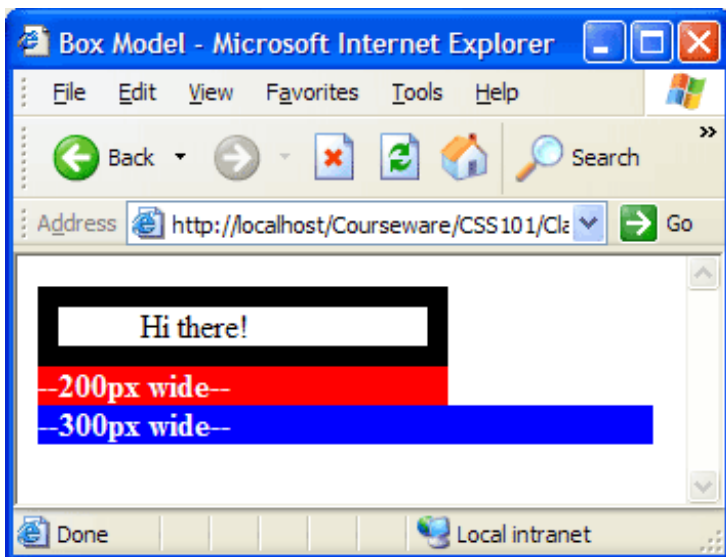
#w300 {
  width: 300px;
  background-color:blue;
  color:white;
  font-weight:bold;
}
</style>
<title>Box Model</title>
</head>
<body>
<div id="Box">
  Hi there!
</div>
<div id="w200">--200px wide--</div>
<div id="w300">--300px wide--</div>

</body>
</html>
```

The page will display correctly in Internet Explorer 6.0 and later as we are using standards-compliant mode:



However, if we remove the DOCTYPE declaration from the code and refresh the page, we'll see what the page would display like in older versions of Internet Explorer:



As you can imagine, this can cause page layout headaches. There is a famous hack that solves this problem called the "Box Model Hack" . Although the creators of this hack were certainly very clever, we don't recommend using it, because...

- the hack is ugly

- it will make your CSS invalid
- Internet Explorer's conditional comments allow for a much cleaner and better way for handling such inconsistencies among browser versions

So, we recommend you use Internet Explorer's conditional comments instead.

Lesson 1, Activity 6: Fixing the Box Model Bug (optional)

Duration: 15 to 25 minutes.

In this exercise, you will use Internet Explorer's conditional comments to fix the box model bug.

1. Open [BrowserBugs/Exercises/BoxModel.html](#) for editing. Notice that no DOCTYPE is declared, so the page will be rendered in quirks mode.
2. Modify the page so that the box containing the words "Hi there!" extends 300 pixels. Note that, because we're using quirks mode, your solution will have to take into account Internet Explorer as well. If we were using standards-compliant mode, this bug would not manifest itself in Internet Explorer, so the conditional statement would only specify IE browsers.
3. Test your solution in Internet Explorer.

Solution:

[BrowserBugs/Solutions/BoxModel.html](#)

```
<html>
<head>
<style type="text/css">
#Box {
  width: 200px;
  padding-left: 40px;
  padding-right: 40px;
  border: 10px solid black;
}

#w200 {
  width: 200px;
  background-color:red;
  color:white;
  font-weight:bold;
}

#w300 {
  width: 300px;
  background-color:blue;
```

```
    color:white;
    font-weight:bold;
}
</style>
<!--[if IE]>
<style type="text/css">
#Box {
    width: 300px;
}
</style>
<![endif]-->
<title>Box Model</title>
</head>
<body>
<div id="Box">
    Hi there!
</div>
<div id="w200">--200px wide--</div>
<div id="w300">--300px wide--</div>

</body>
</html>
```