# Using Floating Elements

In this lesson, we'll learn how to use floating elements in CSS.
Let's begin!

WE'LL COVER THE FOLLOWING



The browser renders the elements of an HTML page with a flowing layout. This means that it places the inline elements from left to right, and when it is necessary it starts a new line. Block elements are rendered from the top of the window down to the bottom.

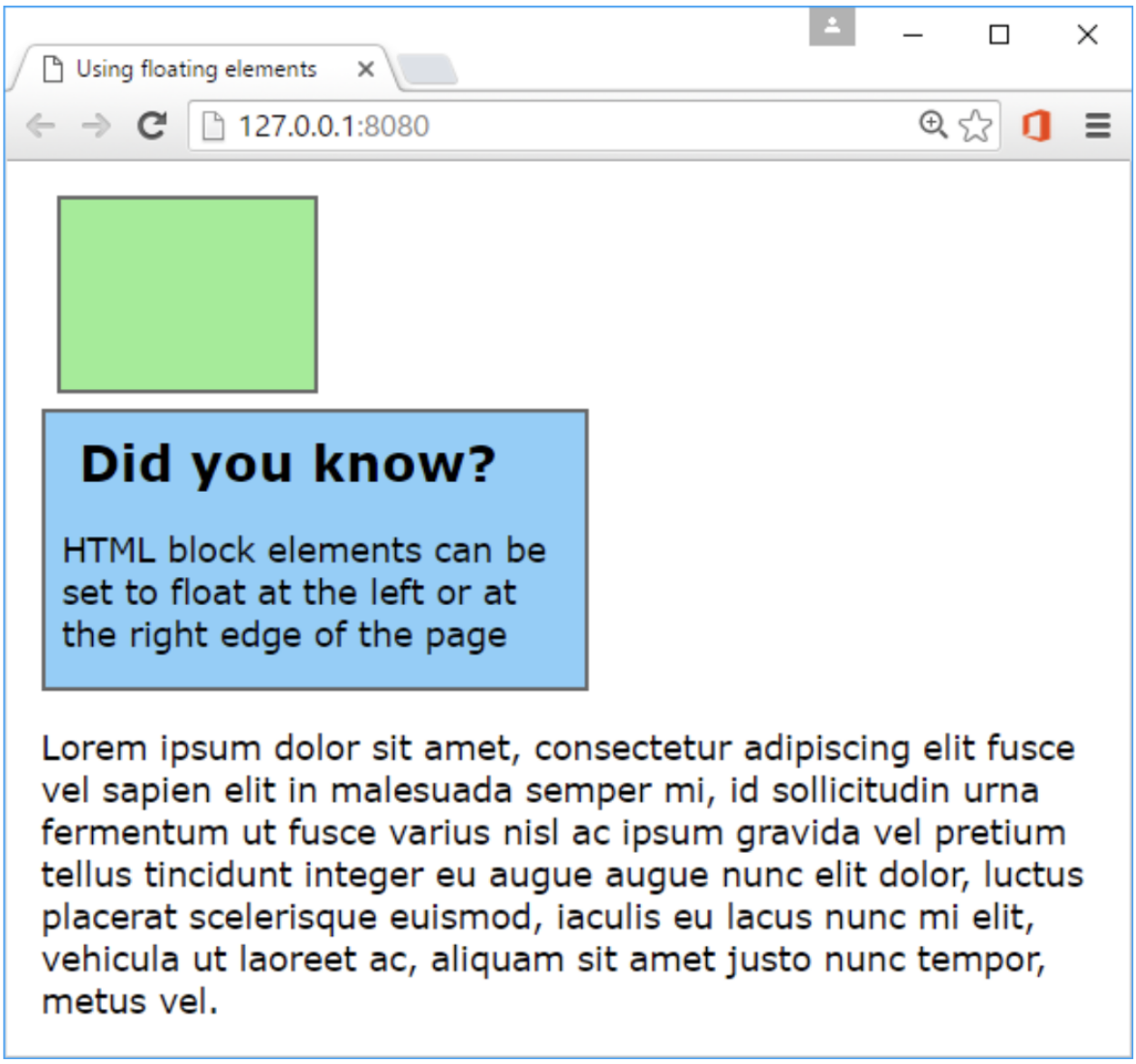
You can change this behavior with the float property; setting its value left or right moves the related block element to the left or to the right, and the subsequent content wraps around the floating element.

Listing 10-12 shows a simple web page that does not contain floating elements, it is displayed in the image that follows.

## Listing 10-12: Using floating elements #

```
<!DOCTYPE html>
<html>
<head>
  <title>Using floating elements</title>
  <style>
    body {
      font-family: Verdana, Arial, sans-serif;
      margin: 16px;
    }

    .image {
      width: 120px;
```

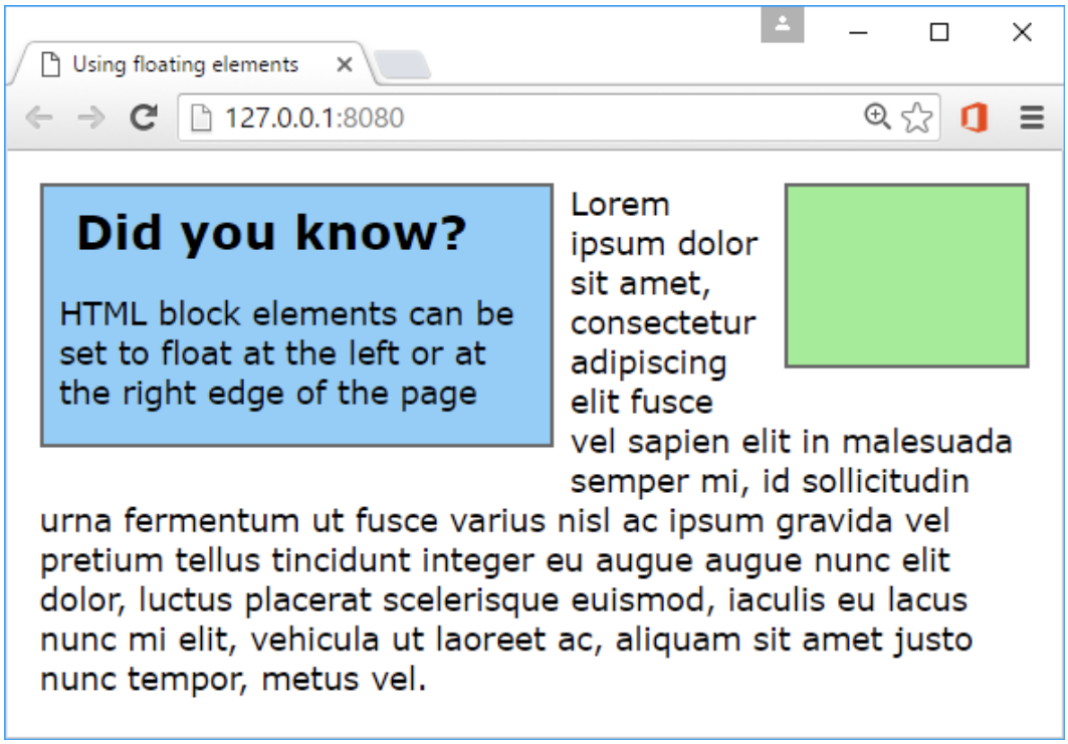
```
      height: 90px;
      margin-left: 8px;
      margin-bottom: 8px;
      border: 2px solid dimgray;
      background-color: lightgreen;
      /*float: right;*/
    }

    .sidebar {
      width: 240px;
      margin-right: 8px;
      margin-bottom: 8px;
      border: 2px solid dimgray;
      background-color: lightskyblue;
      padding: 0 8px;
      /*float: left;*/
    }

    .sidebar h2 { margin: 8px; }
  </style>
</head>
<body>
  <div class="image"></div>
  <aside class="sidebar">
    <h2>Did you know?</h2>
    <p>
      HTML block elements can be set to float
      at the left or at the right edge of
      the page
    </p>
  </aside>
  <p>
    Lorem ipsum dolor sit amet, consectetur
    adipiscing elit fusce vel sapien elit in
    malesuada semper mi, id sollicitudin urna
    fermentum ut fusce varius nisl ac ipsum
    gravida vel pretium tellus tincidunt integer
    eu augue augue nunc elit dolor, luctus
    placerat scelerisque euismod, iaculis eu
    lacus nunc mi elit, vehicula ut laoreet ac,
    aliquam sit amet justo nunc tempor, metus vel.
  </p>
</body>
</html>
```




Page with three block elements

This page contains three block elements (`<div>`, `<aside>`, and `<p>`), and by the standard rendering algorithm they are placed from top to bottom, each block element starting from the left.

Now, if you uncomment the float properties in the code listing, the layout of the page changes. The first empty box (`<div>`) moves to the right, the “Did you know” sidebar (`<aside>`) moves to the left, and the “Lorem ipsum” paragraph wraps around them, as shown below:




Page with floating elements

> **NOTE:** You can turn off floating by setting the float property to none.

When multiple elements float, the rendering engine fills up the width of the browser window as it places the HTML elements in order in the markup. When there's no place for the next element, a new line is started.

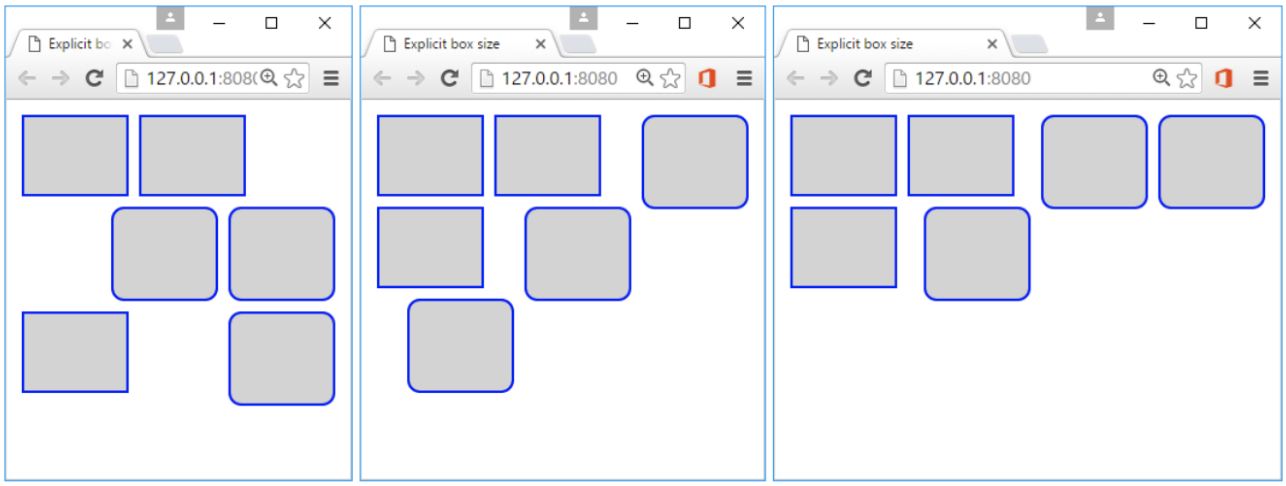
Take a look at Listing 10-13, it contains six floating elements, three moved to the left, and three moved to the right.

## Listing 10-13: Explicit box size #

```
<!DOCTYPE html>
<html>
<head>
  <title>Explicit box size</title>
  <style>
    .left, .right {
      margin: 4px;
      background-color: lightgray;
      border: 2px solid blue;
      width: 80px;
      height: 60px;
    }
```

```
    .left { float: left; }

    .right {
      float: right;
      border-radius: 10px;
      height: 70px;
    }
  </style>
</head>
<body>
  <div class="left"></div>
  <div class="left"></div>
  <div class="right"></div>
  <div class="right"></div>
  <div class="left"></div>
  <div class="right"></div>
</body>
</html>
```

The image below shows three different snapshots of the page while the user resizes the browser window. Try to trace how the rendering engine displays each of these snapshots!




Three snapshots of rendering the six floating elements

> 📜 **NOTE:** Inline elements cannot float.

Sometimes, you need more control over how the floating elements are wrapped around with other content. Although the non-floating elements' content wrap around the floating elements, borders, and background colors do not; they use the full width of the containing element.

For example, if you add an `<h1>` element to Listing 10-12, right before the `<p>`

tag, it may be displayed as shown below:


Borders do not wrap floating elements

This is not the behavior you expect, since you do not need the border behind the floating elements. The `overflow` property is to help you: set it to `hidden` for the heading tag that represents “Borders...”, and you’ll get exactly what you want, as shown below:


The effect of the `overflow: hidden`; property declaration

> 📜**NOTE:** You can find the source code for the above two images in the Exercise-10-18 and Exercise-10-19 folders below.

```
<!DOCTYPE html>
<html>
<head>
  <title>Using floating elements</title>
  <style>
    body {
      font-family: Verdana, Arial, sans-serif;
      margin: 16px;
    }

    .image {
      width: 120px;
      height: 90px;
      margin: 8px;
      margin-top: 0;
      border: 2px solid dimgray;
      background-color: lightgreen;
      float: right;
    }

    .sidebar {
      width: 240px;
      margin: 8px;
      margin-top: 0;
      border: 2px solid dimgray;
      background-color: lightskyblue;
      padding: 0 8px;
      float: left;
    }

      .sidebar h2 {
        margin: 8px;
      }

    h1 {
      background-color: orangered;
      padding: 8px;
      border-top: 4px solid dimgray;
      border-bottom: 4px solid dimgray;
    }
  </style>
</head>
<body>
  <div class="image"></div>
  <aside class="sidebar">
    <h2>Did you know?</h2>
    <p>
      HTML block elements can be set to float
      at the left or at the right edge of
      the page
    </p>
  </aside>
  <h1>Borders...</h1>
  <p>
    Lorem ipsum dolor sit amet, consectetur
    adipiscing elit fusce vel sapien elit in
    malesuada semper mi, id sollicitudin urna
    fermentum ut fusce varius nisl ac ipsum
```

```
    gravida vel pretium tellus tincidunt integer
    eu augue augue nunc elit dolor, luctus
    placerat scelerisque euismod, iaculis eu

    lacus nunc mi elit, vehicula ut laoreet ac,
    aliquam sit amet justo nunc tempor, metus vel.
  </p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Using floating elements</title>
  <style>
    body {
      font-family: Verdana, Arial, sans-serif;
      margin: 16px;
    }

    .image {
      width: 120px;
      height: 90px;
      margin: 8px;
      margin-top: 0;
      border: 2px solid dimgray;
      background-color: lightgreen;
      float: right;
    }

    .sidebar {
      width: 240px;
      margin: 8px;
      margin-top: 0;
      border: 2px solid dimgray;
      background-color: lightskyblue;
      padding: 0 8px;
      float: left;
    }

      .sidebar h2 {
        margin: 8px;
      }

    h1 {
      background-color: orangered;
      padding: 8px;
      border-top: 4px solid dimgray;
      border-bottom: 4px solid dimgray;
      overflow: hidden;
    }
  </style>
</head>
<body>
  <div class="image"></div>
  <aside class="sidebar">
    <h2>Did you know?</h2>
    <p>
      HTML block elements can be set to float
      at the left or at the right edge of
      the page
```

```
    </p>
  </aside>
  <h1>Borders...</h1>

  <p>
    Lorem ipsum dolor sit amet, consectetur
    adipiscing elit fusce vel sapien elit in
    malesuada semper mi, id sollicitudin urna
    fermentum ut fusce varius nisl ac ipsum
    gravida vel pretium tellus tincidunt integer
    eu augue augue nunc elit dolor, luctus
    placerat scelerisque euismod, iaculis eu
    lacus nunc mi elit, vehicula ut laoreet ac,
    aliquam sit amet justo nunc tempor, metus vel.
  </p>
</body>
</html>
```

Although wrapping around floating elements is a useful feature of the browser, there are situations when you want to get rid of it for specific elements. Listing 10-14 demonstrates such a situation.

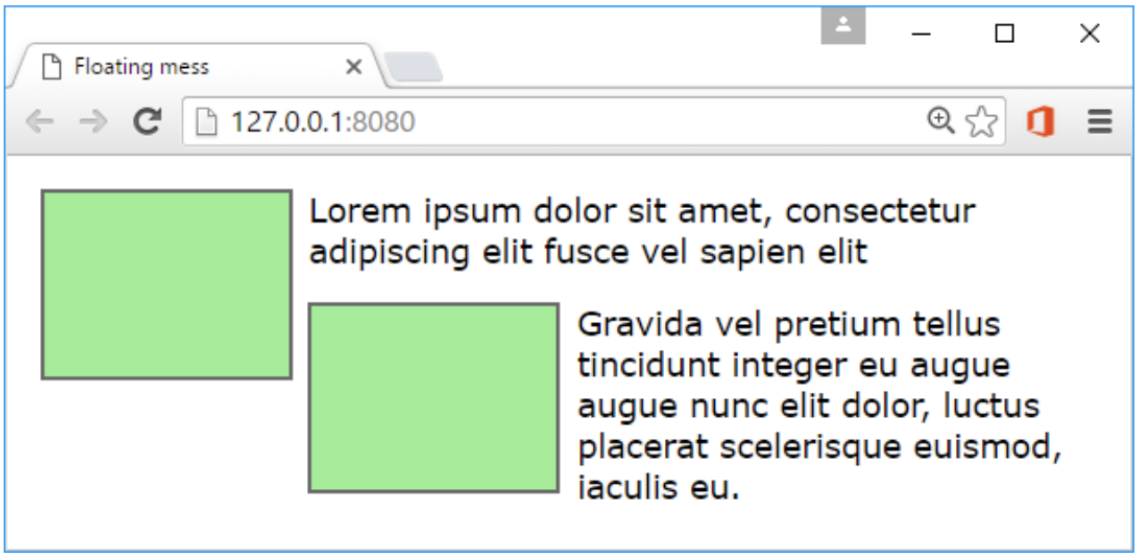
## Listing 10-14: Floating mess #

```
<!DOCTYPE html>
<html>
<head>
  <title>Floating mess</title>
  <style>
    body {
      font-family: Verdana, Arial, sans-serif;
      margin: 16px;
    }

    .image {
      width: 120px;
      height: 90px;
      margin-right: 8px;
      margin-bottom: 8px;
      border: 2px solid dimgray;
      background-color: lightgreen;
      float: left;
    }
  </style>
</head>
<body>
  <div class="image"></div>
  <p>
    Lorem ipsum dolor sit amet, consectetur
    adipiscing elit fusce vel sapien elit
  </p>
  <div class="image"></div>
  <p>
    Gravida vel pretium tellus tincidunt integer
    eu augue augue nunc elit dolor, luctus
    placerat scelerisque euismod, iaculis eu.
  </p>
</body>
```

```
</body>
</html>
```

The `<body>` section contains two content parts, each composed from a `<div>` with a subsequent paragraph. The `<div>` tag represents a thumbnail image, the `<p>` tag is intended to contain some text related to the thumbnail. The `<div>` tags are set to float to left so that the explaining text appears to the right of the thumbnail.

However, what this page displays is not really what you expect, as shown below. Because the text wrapping around the first thumbnail is short, the second thumbnail also wraps the first thumbnail.




Floating elements make this page messy

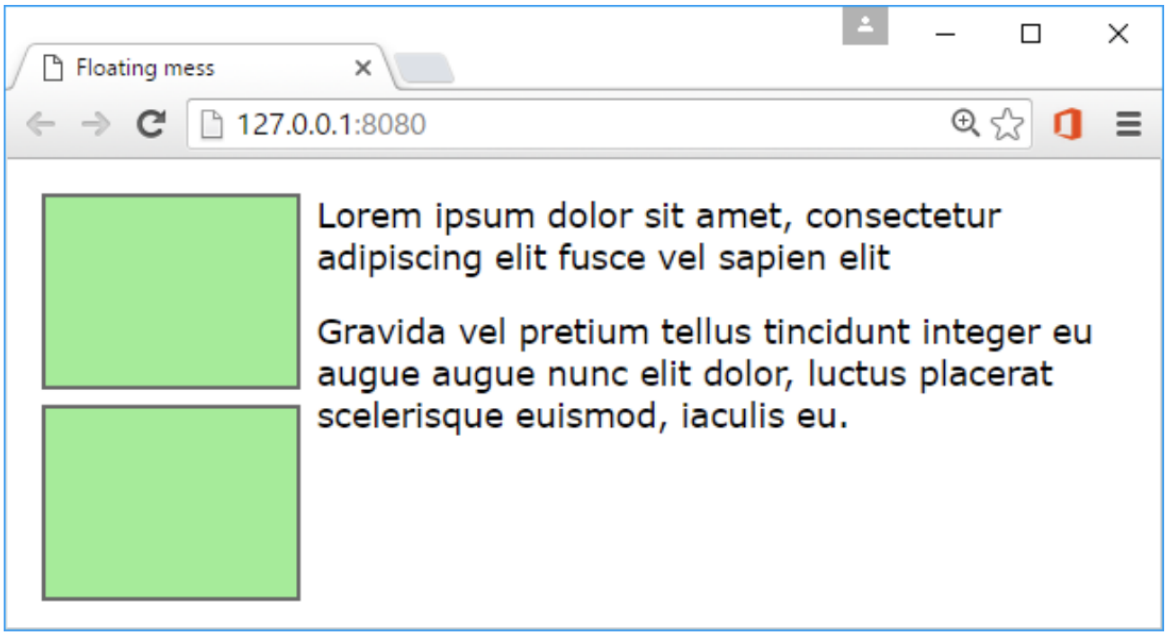
You can use the `clear` property to instruct the browser to not wrap around any floated element. It can set one of the `left`, `right`, `both`, or `none` values. Use `left` to drop below elements that are floated left but intend to still wrap around right-floated objects. When you set this value to `right`, the rendering engine drops below right-floated objects but still wraps around left-floated objects.

As you expect, the both value forces a drop below both left-floated and right-floated elements. You can turn off clearing with `none`; this is the ***default*** value of this property.

To fix the issue with Listing 10-14, add the clear property with left value to the `.image` rule:

```
.image {
      width: 120px;
      height: 90px;
      margin-right: 8px;
      margin-bottom: 8px;
      border: 2px solid dimgray;
      background-color: lightgreen;
      float: left;
      clear: left;
    }
```

The result is shown below. As you see, it aligned the thumbnails to the left and let the text wrap around them.




Using the clear property

> 📜**NOTE:** You can find the source code for the image above in the Exercise-10-21 folder.

```
<!DOCTYPE html>
<html>
<head>
  <title>Floating mess</title>
  <style>
    body {
      font-family: Verdana, Arial, sans-serif;
      margin: 16px;
    }

    .image {
      width: 120px;
      height: 90px;
```

```
      margin-right: 8px;
      margin-bottom: 8px;
      border: 2px solid dimgray;

      background-color: lightgreen;
      float: left;
      clear: left;
    }
  </style>
</head>
<body>
  <div class="image"></div>
  <p>
    Lorem ipsum dolor sit amet, consectetur
    adipiscing elit fusce vel sapien elit
  </p>
  <div class="image"></div>
  <p>
    Gravida vel pretium tellus tincidunt integer
    eu augue augue nunc elit dolor, luctus
    placerat scelerisque euismod, iaculis eu.
  </p>
</body>
</html>
```

You can use percentages to set up the size of floating elements, but sometimes you may be surprised. Take a look at Listing 10-15 that sets up the floating boxes with a 33% width, so three of them can fit in the browser window's width.

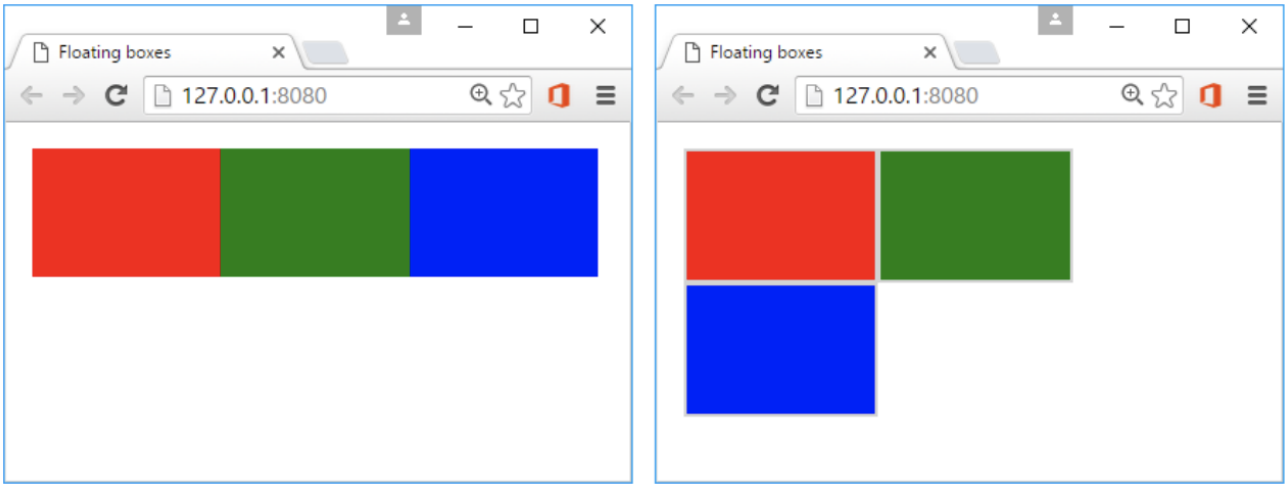
## Listing 10-15: Floating boxes #

```
<!DOCTYPE html>
<html>
<head>
  <title>Floating boxes</title>
  <style>
    body {
      font-family: Verdana, Arial, sans-serif;
      margin: 16px;
    }
    .box {
      width: 33%;
      height: 80px;
      float: left;
      border: 2px solid lightgray
    }
    .red { background-color: red; }
    .green { background-color: green; }
    .blue { background-color: blue; }
  </style>
</head>
<body>
  <div class="red box"></div>
  <div class="green box"></div>
  <div class="blue box"></div>
</body>
```

```
</html>
```

When you display this page, you will see the three boxes in a single row; as shown in the left pane of Figure 10-30.

However, when you add a border to the `.box` rule (for example, `border: 2px solid lightgray`), only two boxes fit in the first row, the third one drops below to the second row, as the right pane shows.




Left—Three boxes fit in a single row. Right—Borders drop the third box below

This phenomenon is not a bug. This is how the browser works by default. As you remember, when you specify percentages, the width of the content is calculated from the width of the containing element, in this case, the browser window's width. While there is no border, three boxes fit in a single row where each has a width of 33%. However, with the borders that add extra width to the boxes, only two fit in the first line.

You can instruct the browser to use a different sizing mode. With the `box-sizing` property, you can change the default content-box option. The browser calculates the content's width, and then adds border and padding widths to `border-box` where the borders and paddings are also considered when calculating the width.

So, if you add the `box-sizing: border-box` property declaration to the `.box` rule in Listing 10-15. It will fix the sizing issue, and the three boxes with borders will fit in a single row.

To make sure the `box-sizing` setting works in all of those browsers, you

To make sure the `box-sizing` setting works in all of those browsers, you should write three declarations like this:

```
-moz-box-sizing: border-box;
-webkit-box-sizing: border-box;
box-sizing: border-box;
```

---

Now, you have learned the important details about the box model, it is time to get acquainted with adding graphics to your web pages starting from the *next lesson*.