Thinking in CSS

A focused guide designed to help you understand the basics of CSS, how it works, and how to start creating modern websites

Aravind Shenoy

Thinking in CSS

A focused guide designed to help you understand the basics of CSS, how it works, and how to start creating modern websites

Aravind Shenoy

Thinking in CSS

Credits

About the Author

Aravind Shenoy is an in-house author at Packt Publishing. An engineering graduate from the Manipal Institute of Technology, his core interests include technical writing, web designing, and software testing. He is a native of Mumbai, India, and currently resides there. He has penned down books such as *Hadoop: A Big Data Initiative* and *Thinking in JavaScript*. He has also authored the bestselling book: *HTML5 and CSS3 Transition, Transformation, and Animation, Packt Publishing*, (`http://www.packtpub.com/html5-and-css3-for-transition-transformation-animation/book`). He is a music buff with *The Doors*, *Oasis*, and *R.E.M* ruling his playlists.

www.PacktPub.com

At `www.PacktPub.com`, you can read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

![PACKTLIB logo]

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

▶ Fully searchable across every book published by Packt

▶ Copy and paste, print and bookmark content

▶ On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Overview

You will gain an intermediate knowledge of CSS after reading this book. Instead of wandering through loads of theory, we will understand CSS more practically so that we can design a webpage using CSS. We have used Notepad for the examples in this book. Alternatively, you can also use Notepad++ or any advanced editor. All that you need to do is copy the code and paste it into Notepad. Upon execution, you will get the output as depicted in the screenshots. Screenshots are provided for each sample code.

Coding gets better with practice. The examples in this book are compatible with almost every browser. Instead of using the verbatim code, you can modify the code and see the change in the output, thereby understanding the subtle nuances of CSS. By the end of the book, with practice, you can achieve better things and get more acquainted with CSS.

Thinking in CSS

HTML originated from a prototype of a language created by Tim Berners-Lee in 1992. Berners-Lee felt that there was a possibility of linking documents together using hypertext, and the concept of HTML evolved from this. A breakthrough in this field of development was the introduction of CSS along with HTML 4.0. Prior to the introduction CSS, web designers and developers were using HTML for formatting purposes. Formatting tags and styling using HTML defeats the purpose of HTML as HTML elements and attributes must only define the structure of a webpage. The purpose of CSS was to take styling out from structural markup.

With the introduction of CSS, we could separate presentation from content. As a result, presentation could be removed from the HTML document and stored in a separate file, which would be included in the document using a `link` tag. Thus, all of the presentational HTML elements and attributes were replaced by CSS to provide versatility and better accessibility. Now, we can just define or modify the look of a webpage by making changes in the stylesheet without actually touching the code. In this book, we will take a look at the essentials. We are going to touch on those topics that every web designer or developer needs to know. Instead of wandering through loads of theory, we will understand CSS more practically so that we can design a web page using CSS by the time you are done with the book. We expect you to know the basics of HTML as we will incorporate CSS in simple HTML code.

Incorporating CSS in HTML

We can incorporate CSS in HTML in the following ways:

- ▶ Using inline styles
- ▶ Using internal styles
- ▶ Using external stylesheets
- ▶ Using the `import` command

Let's take a look at each of these methods in the next sections.

Using inline styles

We can define CSS within the HTML tags. This approach is rarely used; however, we will look at an example to see how it works. In the following code snippet, we have demonstrated the style inside the p element by defining the lime color for the p element:

```html
<html>
<p style="color: lime">Packt Publishing</p>
</html>
```

The output of the code would be as follows:

Packt Publishing

Inline styles are not frequently used as they defy the concept of semantic markup and make the code more complicated. They must be used only when there is no other way of applying CSS. We need to understand that designing a webpage needs a systematic approach.

> At times, the output may vary depending on the browser in use.

Using internal styles

Internal styles are used when we want to define styling for the whole page. We embed CSS into HTML code using the style tag. It is a good practice to include the style tag between the head tags. We need to include <style type="text/css"> to define the start of CSS and end it with </style>.

Let's look at an example to understand it better:

```html
<html>
  <head>
    <title> Basic example of CSS </title>
    <style type="text/css">
      p{ color: red;}
    </style>
  </head>
  <body>
    <p> Packt Publishing </p>
  </body>
</html>
```

The output of the code would be as follows:

Packt Publishing

Using external stylesheets

External stylesheets are the preferred and appropriate way to incorporate CSS in HTML code. We must always remember that HTML and CSS should be kept separate. It is good practice to keep the CSS stylesheet as a standalone document. We need to use the following code snippet in the `head` section of the HTML code to invoke the CSS stylesheet:

```
<link rel="stylesheet" href="yourcssfile.css">
```

The following code is a simple HTML code. We will define the CSS sheet after the HTML code. In the following example code, our CSS file would be called `main.css`:

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="main.css">
  </head>
  <div>
    Packt Publishing has an online library called PacktLIB
  </div>
</html>
```

Now we create a `main.css` file (an external stylesheet) and define the `div` element as follows:

```
div {
  color: red;
}
```

The output of the code would be as follows:

Packt Publishing has an online library called PacktLIB

Imagine we have a lot of web pages, which is the norm for any website. If we use the inline or internal way of defining CSS, we would have to search for the styled elements in each HTML page. However, if we maintain a separate standalone stylesheet for CSS, we just have to modify the properties in the stylesheet and the modification would be very systematic. It is good practice to keep the CSS stylesheet separate from the HTML page. Moreover, if we have lots of common styles defined for several HTML elements, all we need to do is redefine the styles in the standalone CSS stylesheet, thereby maintaining a more semantic approach.

Using the import command

Let's understand how the `import` command works. Suppose we have defined an external stylesheet for a website. Say a new requirement comes up where we need the color of the font to be changed to yellow for the first page when it is actually defined as blue in the `main.css` CSS stylesheet. We use the `import` command for this purpose. Hence, we create a new stylesheet and give it a name of our choice, say `change.css`. Then we invoke the `change.css` stylesheet using the `import` command. This way, only the first page is affected, and the rest stay the same. The following code snippet shows how we incorporate the `import` command in our HTML code:

```html
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="main.css" >
    <style type="text/css">     @import url(change.css);     </style>
  </head>
</html>
```

We have to define the `import` command between the `style` tags.

We have discussed the various ways in which we can apply CSS to a webpage.

However, we need to remember that the most appropriate way to include CSS is to maintain a standalone stylesheet. It is more systematic and easily modifiable that way.

In this book, for the sake of convenience, we will use the internal way of implementing CSS.

Defining CSS

Let's look at the basic structure of CSS:

```css
selector {
  property:value;
}
```

The `selector` (in this case, `div`) is to be written first. The `property` and `value` have to be defined within the curly brackets. `color`, `text-decoration`, and `font-size` are examples of properties used in CSS. A value is given to each property. For example, `red` is a value given to the `color` property. A colon separates the property and the value. The semicolon that follows the property completes the structure. If you observe the previous code snippet, you can see that we have defined `property` twice. It demonstrates that multiple properties can be defined and are separated by semicolons. Let's have a look at the following code snippet:

```css
div {
  color: red;
  margin: 20px;
}
```

red is the value given to the `color` property. The `margin` property is also defined after the `color` property. The `color` and `margin` properties are separated by a semicolon.

Selectors

Selectors assist us in selecting the element we want to style. There are different kinds of selectors in CSS. An element, ID, or class can be used as selectors, to mention a few. We will have a look at the various kinds of selectors in CSS as we go through the book.

Class and ID selectors

The most commonly used selectors in CSS are the class and ID selectors. The class selector is preceded by a period (.) whereas the ID selector is preceded by a # sign. The difference between an ID and a class selector is that an ID is unique whereas a class selector can be used multiple times.

Let's have a look at the following code to see how it works:

```
<html>
  <head>
    <title> Id and Class Selectors </title>
    <style type="text/css" >
      #pubman {color: red;}
      .packt {color: navy;}
    </style>
  </head>
  <body>
    <p class="packt"> Packt, Always finding a way </p>
    <p id="pubman"> Packt Publishing : Packt Online Library:
      Packtlib </p>
  </body>
</html>
```

The output of this code would be as follows:

Packt, Always finding a way

Packt Publishing : Packt Online Library: Packtlib

If we have a look at the previous code snippet, we can see that the class selector is `packt` whereas the ID selector is `pubman`. The `packt` selector is preceded by a period whereas `pubman` is preceded by a # sign.

Class and ID are used extensively in CSS. We can also group and nest selectors within one another.

Grouping selectors

In CSS, we can group any number of selectors to define properties that are common. We do not have to define selectors separately, as that would create a humongous amount of code. Grouping helps us in writing less and systematic code. Let's have a look at the following code to see how it works:

```html
<html>
  <head>
    <title> Example of Grouping selectors </title>
    <style type="text/css" >
      p, div {color: navy;}
    </style>
  </head>
    <p> Packt Publishing </p>
    <div> Packt Online Library </div>
</html>
```

The output of the code would be as follows:

If we observe the output, it is evident that the `color` property is common for the `p` and `div` elements. Hence, the text defined between the `p` and `div` elements has the same `navy` color. See how we have separated `p` and `div` by a comma. We can also club ID and class selectors along with the other selectors.

The following code snippet will show you how it is done:

```css
p, #packt, .pubman { color: red; }
```

As we can see, the `p` element, the `packt` ID, and the `pubman` class have been clubbed together and have been assigned the same property and value.

Let's now have a look at the concept of nesting selectors within one another.

Nesting selectors

We can apply styles to a selector within a selector. Let's have a look at the following code to see how it is done:

```html
<html>
<head>
<title> Nesting Selectors </title>
<style type="text/css">
p{color: lime;}
div{background-color: yellow;}
```

```
#packt p {color: orangered;}
</style>
</head>
<body>
<p> Packt Publishing </p>
<div id="packt">Packt Publishing, Always finding a way
<p> Packt Online Library </p>
</div>
<p> Packt Publishing Article Network</p>
</body>
</html>
```

The output of the code would be as follows:
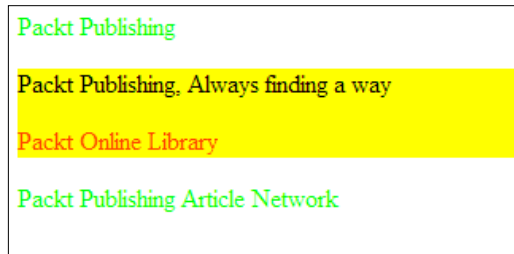
In the previous code, we have defined the `lime` color for all the `p` elements initially. After that, we have defined the `yellow` background for the `div` element, which is represented by a unique `packt` (`#packt`) ID. Inside the `div` element, we have defined a `p` element to which we have assigned the `orangered` color. It is evident from the output that the `p` elements outside `div` are lime in color. However, the style for the intrinsic `p` element in `div` is orangered in color. Hence, **Packt Publishing** and **Packt Publishing Article Network** are lime in color whereas **Packt Online Library** is orangered in color.

Fonts and text

In HTML, the size of a font can be increased using tags such as `h1` and `h2` to mention a few. However, as we already know, HTML is to be used for markup, whereas CSS is to be used for styling. Hence, fonts are imperative in CSS. Let's have a look at the following code to understand the various font styles in CSS:

```
<html>
  <head>
    <title> Fonts in CSS </title>
    <style type= "text/css" >
      P {
        font-family: 'Times New Roman', Times, serif;
        font-style: normal;
```

```
      color: lime;
      font-size: 1.5em;
    }
    div {
      font-family:Arial,Helvetica
      font-weight: bold;
      font-style: oblique;
      color: orangered;
      font-size: 3em;
    }
  </style>
</head>
<body>
  <p> Packt Publishing </p>
  <div id= "packt"> Packt Online Library </div>
</body>
</html>
```

The output of the code would be as follows:

If you have a look at the previous code, you can see that the font is defined as `Arial` and `Helvetica` for the `div` element. If the browser doesn't support `Arial`, it will incorporate the next option, `Helvetica`. If the `font-family` property is not defined, it will take the default font defined in the browser.

The `font-size` property is defined in `em`; however, it can also be defined in percentage as well as in pixels (`em` is the current size of the font; here `font-size: 3em` will be three times the calculated size of the font.)

The `font-weight` property is used for defining the text as `bold` or `normal`.The `font-size` property, as the name suggests, will decide the size of the font. `font-style` will define whether the text is `normal`, `italic`, or `oblique`. Text in CSS can be aligned and defined in a lot of ways. Let's have a look at the following code to understand it better:

```
<html>
  <head>
    <title> Text in CSS </title>
    <style type="text/css">
      p { color: orangered;}
      #packt {        color: lime;
```

```
          text-decoration: underline;
          text-align: center;
        }
        #pubman {
          color: orangered;
          text-align: left;
          text-indent:50px;
        }
        div {
          text-transform:uppercase;        color: navy;
          text-decoration: line-through;
          text-align:justify;        }
      </style>
    </head>
    <body>
      <p> Packt Publishing </p>
      <p id= "packt"> Packt Online Library </p>
      <p id= "pubman" > Packt Article Network </p>
      <div> Packt publishes books customized for you on various
        technologies </div>
    </body>
  </html>
```

The output of the code would be as follows:

Packt Publishing

                                                              Packt Online Library

         Packt Article Network

PACKT PUBLISHES BOOKS CUSTOMIZED FOR YOU ON VARIOUS TECHNOLOGIES
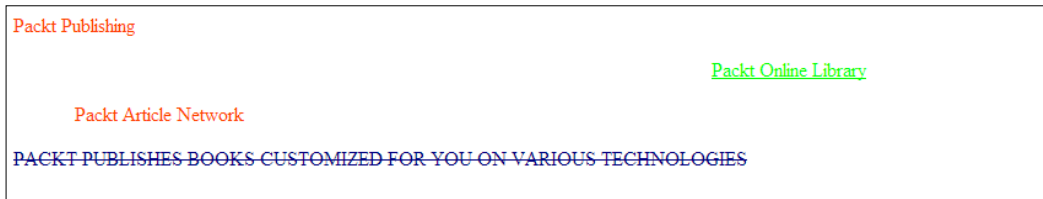
`text-decoration` determines whether a line runs over, under, or through the text. `text-align` will position the text to the right, left, center, or justify it according to the layout of the web page. For indentation of the text, we use the `text-indent` property. `text-transform` will change the case of the text. For example, we can change the text to uppercase or lowercase using the `text-transform` property.

If you observe the output of the code, you can see that line `text-decoration: line-through` inserts a line through the text, whereas `text-transform:uppercase` changes the case of the line to uppercase. So you can see how fonts and text can be manipulated in CSS.

Anchor pseudo classes

If you check out modern websites nowadays, the hyperlinks are in blue color and the color changes to purple if that web page has been visited. We achieve this using the anchor pseudo classes in CSS. Let's have a look at the following code to see how this works:

```html
<html>
  <head>
    <title> Anchor Pseudo Classes in CSS </title>

    <style type="text/css">
      a:link {color: blue;}     a:visited {color: lime;}     a:hover
{color: orangered;}    </style>

  </head>
  <body>
    <p> <a href="http://packtpub.com"> Packt Publishing </a></p>
    <br>
    <p>
      <a href="http://packtlib.packtpub.com">
        Packt Online Library </a>
    </p>
  </body>
</html>
```

The output of the code would initially be as follows:

Packt Publishing

Packt Online Library
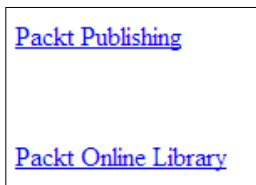
Say we click on the **Packt Publishing** link; we would see the link turn lime. The output would be as follows:

Packt Publishing

Packt Online Library

If we check the previous CSS code, we see that we have used the following styles for the `link` tags:

```
a:link {color: blue;}
a:visited {color: lime;}
a:hover {color: orangered;}
```

Initially, the links would be blue in color; however, the color changes to lime if we click on the link at least once. When we hover over the link, we can see that the color of the link changes to orangered, as shown in the following screenshot:
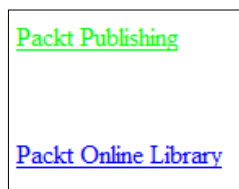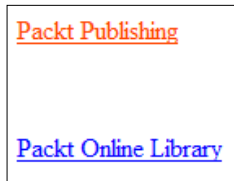

Border

We use the `border` property to apply various styles and colors to an element's border.

Let's have a look at the following code to understand it better:

```html
<html>
  <head>
    <title> Border in CSS </title>
    <style type="text/css">
      p {
        border-style: dotted;        border-width: 2px;
        border-color: lime;
        border-radius: 25px;
        display:inline;
        margin: 25px;
        padding: 10px 10px;
      }
    </style>
  </head>
  <body>
    <br> <br><br>
    <p>Packt Publishing </p>
  </body>
</html>
```

The output of the code would be as follows:

Packt Publishing
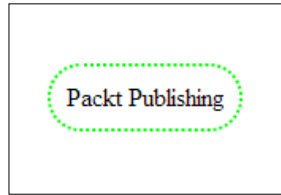
If we observe the previous code and the subsequent output, we can see that the border has a dotted pattern. We can also use `solid` or `dashed` values to define a `border-style`. The `border-width` property can set the width of the border. The `border-radius` property is a **CSS3** property and defines the curvature of the border. The `border-color` helps us to assign a color to the border. In the previous code, we have used a dotted border of lime color with a border width of 2 pixels and a border radius of 25 pixels. In the previous code, we have used the margin and padding properties, which will be explained in the next section.

Margin and padding

Margin and padding are used extensively in designing. Margin is the space outside any defined element, whereas padding is the space inside any defined element.

Let's have a look at the following code to understand it better:

```html
<html>
  <head>
    <title> Margin and Padding in CSS </title>
    <style type="text/css">
      p {
        background-color: lime;
        border-style:solid;          display: inline;
      }
      #packt {
        margin-top: 200px;
        margin-right: 50px;
        margin-bottom: 20%;
        margin-left: 150px;
      }
    </style>
  </head>
  <body>
    <p> Packt Publishing </p>
    <br>
    <p id="packt"> Packt Publishing </p>
  </body>
</html>
```

The output of the code would be as follows:
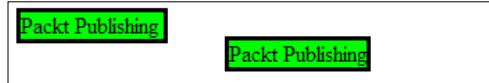
In the previous code, the element defined in CSS by the ID selector has the margin defined. Hence, in the output, the position of the element referenced by ID is different from the previous element that has no margin defined. An element has four sides: top, bottom, left, and right. Hence, the margin is the distance from each side to the neighboring element or the borders of the page. If we observe the output of the code, the element referenced by the `packt` ID is at a different position from the other one, which has no margin. The margin cannot be seen as it has no background color and is transparent.

There is a shorthand way in which `margin` can be defined as shown in the following code snippet:

```
p { margin: 50px 30px;}
```

The top and bottom margin for the element would be 50 pixels whereas the right and left margin would be 30 pixels.

We will now discuss padding. Padding is different from margin in the sense that the distance between elements is not taken into consideration. In padding, what matters is the distance between border and content of an element. We will have a look at the same HTML code, but we will tweak it a bit in this case and introduce padding for the defined elements. Let's have a look at the following code to understand it better:

```
<html>
  <head>
    <title> Margin and Padding in CSS </title>
    <style type="text/css">
      p {
        background-color: lime;
        border-style:solid;          display: inline;
        padding:10px 20px;
      }
      #packt {
        margin-top: 300px;
        margin-right: 150px;
        margin-bottom: 20%;
        margin-left: 150px;
      }
    </style>
  </head>
  <body>
    <br>
    <p> Packt Publishing </p>
```

```
        <br><br>
        <p id="packt"> Packt Publishing </p>
    </body>
</html>
```

The output of the code would be as follows:
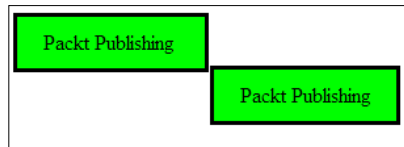
Both the elements have a padding of 10 pixels at the top and bottom, whereas the padding towards the right and left is around 20 pixels. There is enough space between the word **Packt Publishing** and the border surrounding it. Hence, padding decides the space between the border and the content of the element.

We can define padding as we defined margin in the following way:

```
padding-top:25px;padding-bottom:25px;padding-right:50px;padding-
left:50px;
```

But to reduce the lines of code, we can use the shorthand way, which we have used in the previous code where we used the line `padding: 10px 20px;` This will introduce a padding of 10 pixels at the top and bottom as well as a padding of 20 pixels at the right and left sides.

Display

The `display` property specifies the manner in which an element is displayed in a webpage. The `display` property can be used in the following ways:

- `display: none;`
- `display: inline;`
- `display: block;`

When we use `display: none`, the element will be hidden and will not take up any space in the window. It will not affect the layout.

In HTML, there are block and inline elements. Block elements are those that take the complete width along with a line break incorporated in it (`div`, and `p` to mention a few).

An inline element consumes space that is necessary (based on its content) and does not include a line break. (`br`, `a`, and `span` to mention a few). Let's have a look at the following code to understand it better:

```
<html>
  <head>
    <style type="text/css">
      p {         border: solid;
        border-color: red;
        color: lime;
      }     </style>
  </head>
  <body>
   <p> Packt Publishing </p>
  </body>
</html>
```

The output of the code would be as follows:

In the code, we have not included the `display` property. Hence, `p` being a block element takes the full width.

Now we will tweak the code a bit. We will include the `display` property with the inline value assigned to it. Let's have a look at the following code to understand it better:

```
<html>
  <head>
    <style type="text/css ">
      p {         border: solid;
        border-color: red;
        color: lime;
        display: inline;
      }     </style>
  </head>
  <body>
    <p> Packt Publishing </p>
  </body>
</html>
```

The output of the code would be as follows:

It is quite evident from the output that including `display: inline` in CSS will change the display of the `p` element to inline. It will not change the nature of the element. It will just change the way the element is displayed. Similarly, the display of an inline element such as `span` can be changed to block by using the `display: block` property.

Positioning

When it comes to complex websites, the most essential thing is a precise and advanced layout. With the positioning feature, we can place items anywhere on the page. The commonly used positioning techniques are **relative positioning** and **absolute positioning**. Let's have a look at the following code. The following code does not include any positioning:

```html
<html>
  <head>  </head>
  <body>
    <p id="packt"> Packt Publishing </p>
    <p id="packtpub"> Packt Online Library </p>
    <p> Packt Web Apps </p>
  </body>
</html>
```

The output of the code would be as follows:

Packt Publishing

Packt Online Library

Packt Web Apps

Suppose we want to demonstrate relative positioning. We will include the `position` property and assign the `relative` value to it. Let's have a look at how the output would be if we apply these styles to the previous code:

```html
<html>
  <head>
    <style type="text/css">
      #packtpub  {
        position: relative;
        left: 70px;
        top:  100px;
      }
    </style>
  </head>
  <body>
    <p id="packt"> Packt Publishing </p>
    <p id="packtpub"> Packt Online Library </p>
    <p> Packt Web Apps </p>
  </body>
</html>
```

The output of the code would be as follows:

Packt Publishing

Packt Web Apps

        Packt Online Library

If we observe the output, **Packt Online Library** has been shifted from its original position and is positioned 70 pixels to the left and 100 pixels below its original position. Hence, the `position: relative` feature positions the element with respect to its original position. Notice how there is an empty space at the original position of the element. This explains the concept of relative positioning.

Now, we will have a look at absolute positioning. We will tweak the same code and, instead of `relative`, we will assign the `absolute` value to the `position` property.

Let's have a look at the following code to see how it works:

```
<html>
  <head>
    <style type="text/css">
      #packtpub  {
        position: absolute;
        left: 70px;
        top:  100px;
      }
    </style>
  </head>
  <body>
    <p id="packt"> Packt Publishing </p>
    <p id="packtpub"> Packt Online Library </p>
    <p> Packt Web Apps </p>
  </body>
</html>
```

The output of the code would be as follows:

Packt Publishing

Packt Web Apps

        Packt Online Library

Observe the output. The original position of **Packt Online library** is occupied by **Packt Web Apps**. There is no empty space after it has been repositioned. The other elements behave as though the absolutely positioned elements do not exist at all. Hence, it affects the layout by not leaving any empty space. This is the concept of absolute positioning.

Float

We can move the elements to the left or right using the `float` property. It is useful if we want to position images and wrap the text around it. However, the `float` property is not limited to images and can even be used for positioning elements in a web page. The `float` property is extremely useful as block elements will not line up beside one another. Using `float`, content can be wrapped around the floating element. Let's have a look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
  <head>
    <style type="text/css">
      img {        border: dashed;
        float:right;
      }
    </style>
  </head>
  <body>
    <p>
      Packt is one of the most prolific and fast-growing tech book
        publishers in the world. Originally focused on open source
        software, Packt pays a royalty on relevant books directly
        to open source projects. These projects have received over
        $400,000 as part of Packt's Open Source Royalty Scheme
        to date.
    </p>
    <p>
      <img src="Packt1.png"/>
      Our books focus on practicality, recognising that readers
        are ultimately concerned with getting the job done.
        Packt's digitally-focused business model allows us to
        publish up-to-date books in very specific areas. With
        over 1000 books published, Packt now offers a subscription
        service. This app and a PacktLib subscription now makes
        finding the information you need easier than ever before.
        PacktLib is Packt's online digital book library. Launched
        in August 2010, it allows you to access and search almost
        100,000 pages of book content, to find the solutions you
        need. As part of the Open Source community, Packt aims to
        help sustain the projects which it publishes books on.
```

```
            When we sell a book written on an Open Source project, we
            pay a royalty to that project. Open Source projects have
            received over $400,000 through this scheme to date.
        </p>
    </body>
</html>
```

The output of the code would be as follows:
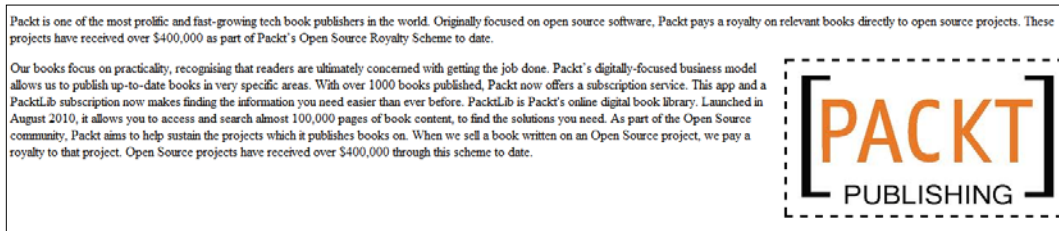

Packt is one of the most prolific and fast-growing tech book publishers in the world. Originally focused on open source software, Packt pays a royalty on relevant books directly to open source projects. These projects have received over $400,000 as part of Packt's Open Source Royalty Scheme to date.

Our books focus on practicality, recognising that readers are ultimately concerned with getting the job done. Packt's digitally-focused business model allows us to publish up-to-date books in very specific areas. With over 1000 books published, Packt now offers a subscription service. This app and a PacktLib subscription now makes finding the information you need easier than ever before. PacktLib is Packt's online digital book library. Launched in August 2010, it allows you to access and search almost 100,000 pages of book content, to find the solutions you need. As part of the Open Source community, Packt aims to help sustain the projects which it publishes books on. When we sell a book written on an Open Source project, we pay a royalty to that project. Open Source projects have received over $400,000 through this scheme to date.

From the output, it is quite evident that the image has floated to the right of the web page and the content in the paragraph tags wraps itself around the left of the image.

> Even an inline element will turn into a block-level element if the `float` property is assigned to it.

The `clear` property in CSS is used with `float` when there is a need to clear floated elements on the left or right-hand side of an element.

Let's have a look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .packt
{
        float:left;
        border: solid;
        margin:10px;
      }
    </style>
  </head>
  <body>
    <h1> Packt Publishing</h1>
    <img class="packt" src="Packt1.png" width="200" height="110">
    <img class="packt" src="Packt2.png" width="220" height="110">
    <img class="packt" src="Packt3.png" width="220" height="110">
    <img class="packt" src="Packt4.png" width="220" height="110">
    <h2> The next row of pictures </h2>
```

```
    <img class="packt" src="Packt5.png" width="220" height="110">
    <img class="packt" src="Packt6.png" width="220" height="110">
    <img class="packt" src="Packt7.png" width="220" height="110">
  </body>

</html>
```
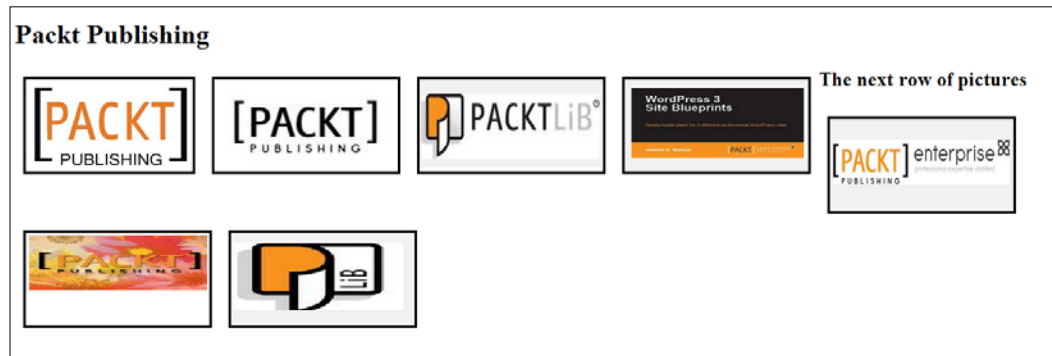
The output of the code would be as follows:

The output shows that the elements after the floating elements will flow around it. Hence, we see that the **The next row of pictures** heading is not aligned correctly. To resolve this problem we use the clear property.

Let's now tweak the same code a bit and include the `clear` property in the CSS styles. Let's have a look at the following code to understand it better:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .packt
{
        float:left;
        border: solid;
        margin:10px;
      }
      #pub
 {
        clear: both;
        margin-top: 2px;
        margin-bottom: 4px;        }
    </style>
  </head>
  <body>
```

```
    <h1> Packt Publishing</h1>
    <img class="packt" src="Packt1.png" width="200" height="110">
    <img class="packt" src="Packt2.png" width="220" height="110">
    <img class="packt" src="Packt3.png" width="220" height="110">
    <img class="packt" src="Packt4.png" width="220" height="110">
    <h2 id="pub"> The next row of pictures </h2>
    <img class="packt" src="Packt5.png" width="220" height="110">
    <img class="packt" src="Packt6.png" width="220" height="110">
    <img class="packt" src="Packt7.png" width="220" height="110">
  </body>
</html>
```

The output of the code would be as follows:

After observing the output, it is quite evident that the positioning is done correctly by the use of the `clear` element. We have used `clear:both` in the code as we do not want floating elements on the right and left-hand side of the **The next row of pictures** heading statement.

Summary

We have covered the essentials of CSS in this book. However, one needs to understand that coding gets better with practice and that the learning curve must always show an upward trend. However, the things you have learned in this book are compatible with almost any browser. You need to realize that, by reading this book, you have just reached the shore of an island. The sea of knowledge is far beyond. With practice, you can achieve better things and get more acquainted with CSS.

Broaden your horizons with Packt

If you're interested in CSS, you've come to the right place. We've got a wide range of products that should appeal to budding and proficient web designers and developers. For a 35% discount on any of these titles, or any of our catalog, please visit packtpub.com and enter ThinkCSS35 in the 'promotional code' section of the checkout.

HTML5 and CSS3 Transition, Transformation, and Animation

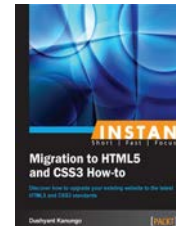HTML5 and CSS3 Responsive Web Design Cookbook

Responsive Web Design with HTML5 and CSS3

Sass and Compass for Designers

Mobile First Design with HTML5 and CSS3

Instant Migration to HTML5 and CSS3 How-to

If the right book for you isn't listed here, you can visit our online library PacktLib at `http://packtlib.packtpub.com/` for our full list of CSS titles as well as a wide range of other technologies. Alternatively, you can check out our responsive browser app at `https://app.packtpub.com/` using your smart phone or tablet.

Feel free to let us know about the kind of titles that you'd like to see on CSS. You can get in touch via `contact@packtpub.com`.

Your input is very important to us; it helps us produce better products that help the community more. If you have any feedback on the books, or are struggling with something we haven't covered, then surely let us know; `customercare@packtpub.com`.

To know more about our future releases, our full catalogue, and daily deals, please visit `http://www.packtpub.com/`.