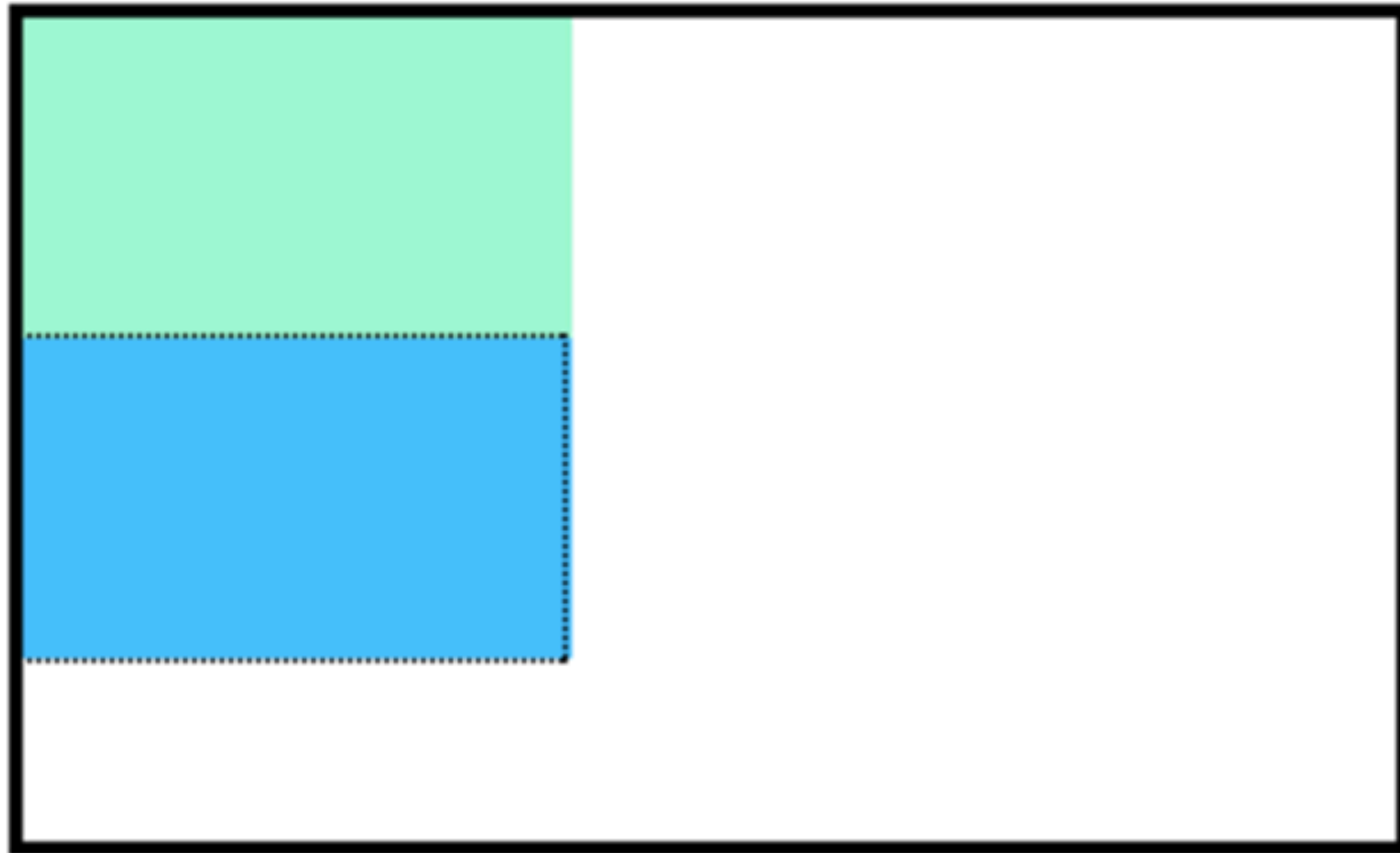


CSS Positioning

The box model is the first step in understanding how the browser lays out HTML elements. Visually appealing websites, however, are often the result of well positioned elements.



The boxes in the image were created with the following CSS:

```
.boxes {  
  width 120px;  
  height 70px;  
}
```

Notice the block-level elements in the image above take up their own line of space and therefore don't overlap each other.

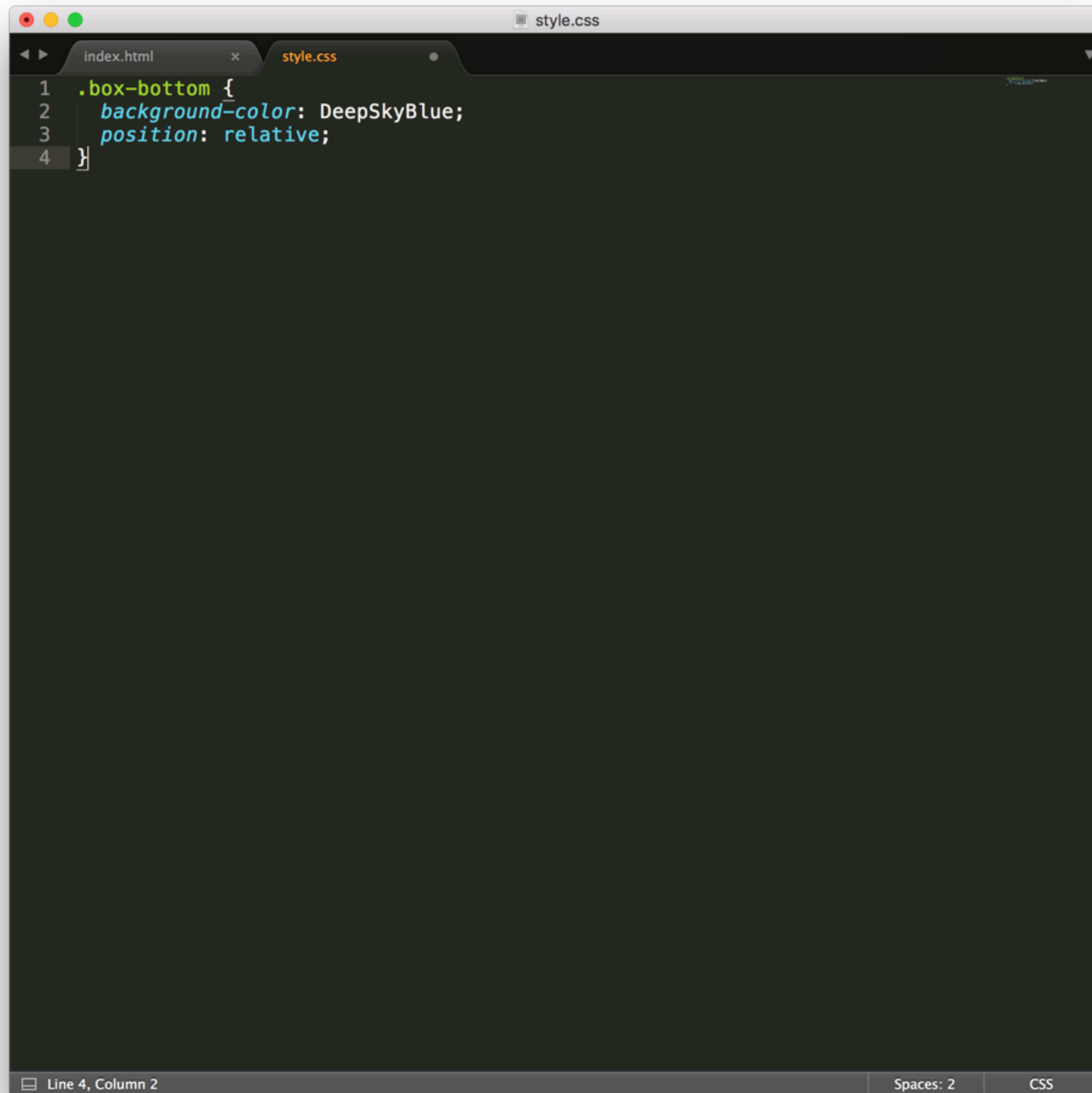
This is the default *position* for block-level elements.

The default position of an element can be changed by setting its **position** property.

The **position** property can take one of four values:

1. **static** – the default value (it does not need to be specified)
2. **relative**
3. **absolute**
4. **fixed**

One way to modify the default position of an element is by setting its **position** property to **relative**.

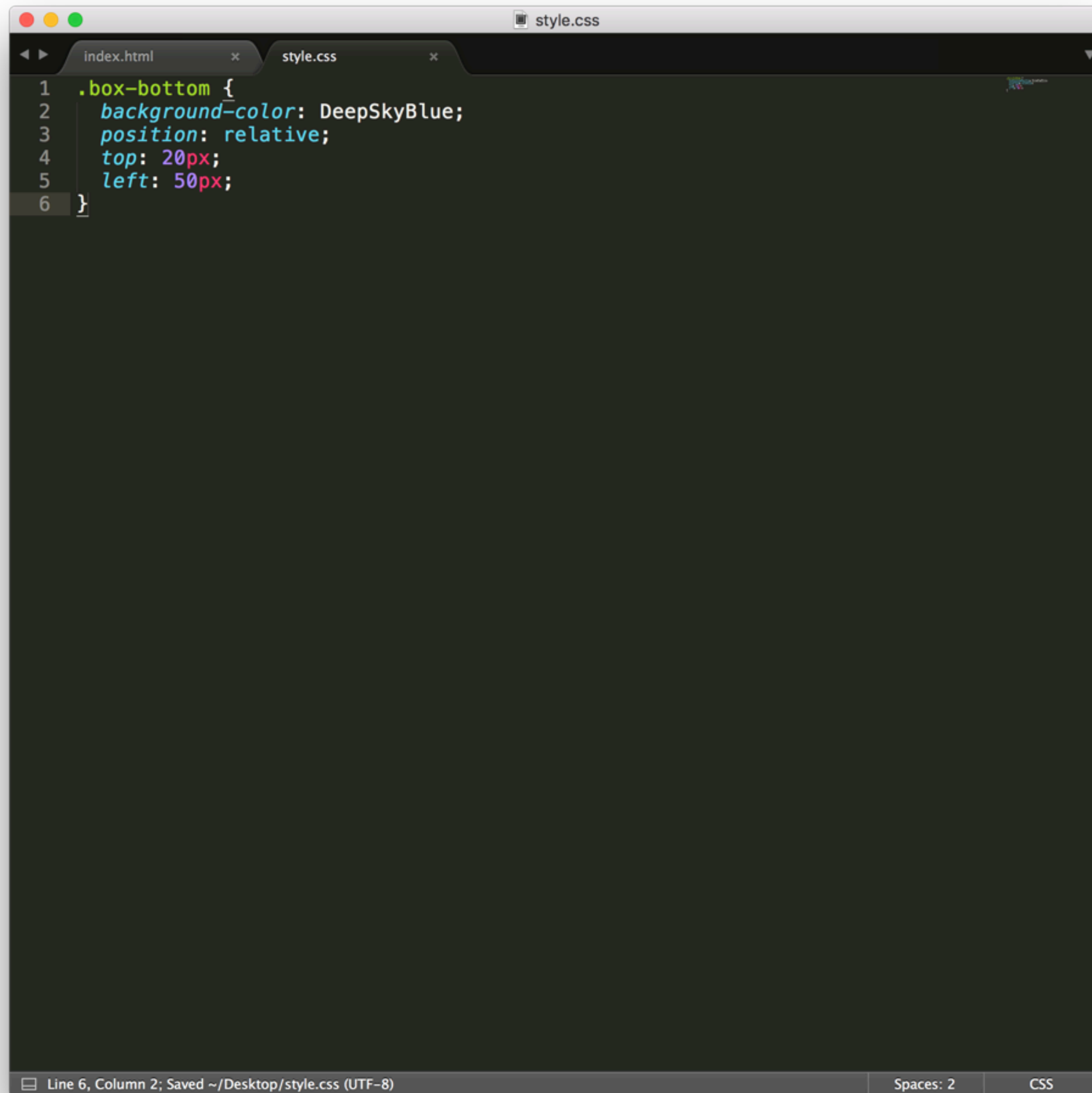
A screenshot of a code editor window with a dark theme. The window has two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 .box-bottom {  
2   background-color: DeepSkyBlue;  
3   position: relative;  
4 }
```

The code is syntax-highlighted. The status bar at the bottom indicates 'Line 4, Column 2', 'Spaces: 2', and 'CSS'.

This value allows you to position an element *relative* to its default static position on the web page.

Although the code in the example above instructs the browser to expect a relative positioning of the div, it does not specify where the div should be positioned on the page.



```
1 .box-bottom {  
2   background-color: DeepSkyBlue;  
3   position: relative;  
4   top: 20px;  
5   left: 50px;  
6 }
```

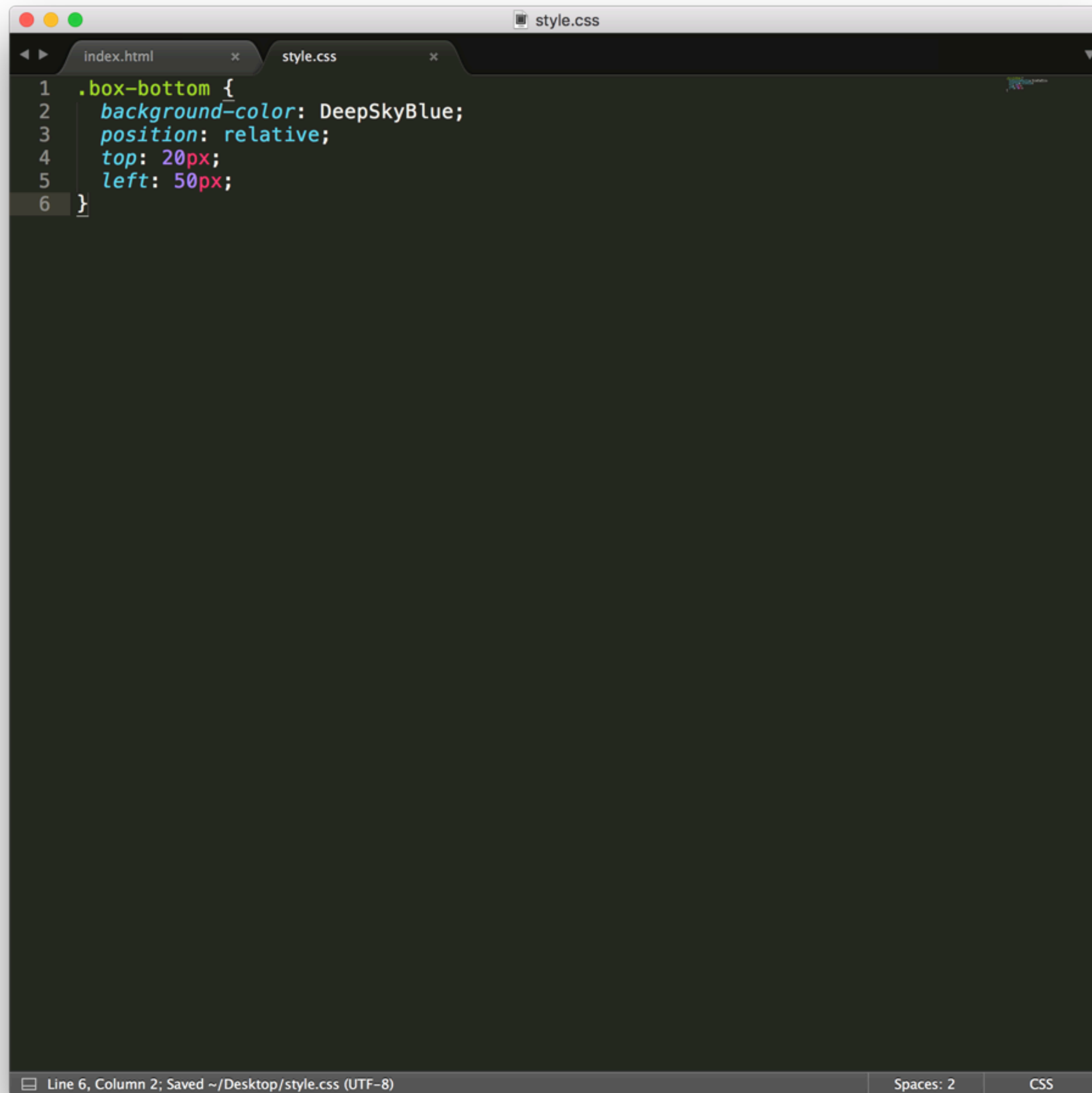
The screenshot shows a code editor window with two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, displaying the CSS code for the '.box-bottom' class. The code is as follows:

```
1 .box-bottom {  
2   background-color: DeepSkyBlue;  
3   position: relative;  
4   top: 20px;  
5   left: 50px;  
6 }
```

The status bar at the bottom indicates 'Line 6, Column 2; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

The div has been positioned using two of the four *offset properties*. The valid offset properties are:

- top** – moves the element down.
- bottom** – moves the element up.
- left** – moves the element right.
- right** – moves the element left.



```
1 .box-bottom {  
2   background-color: DeepSkyBlue;  
3   position: relative;  
4   top: 20px;  
5   left: 50px;  
6 }
```

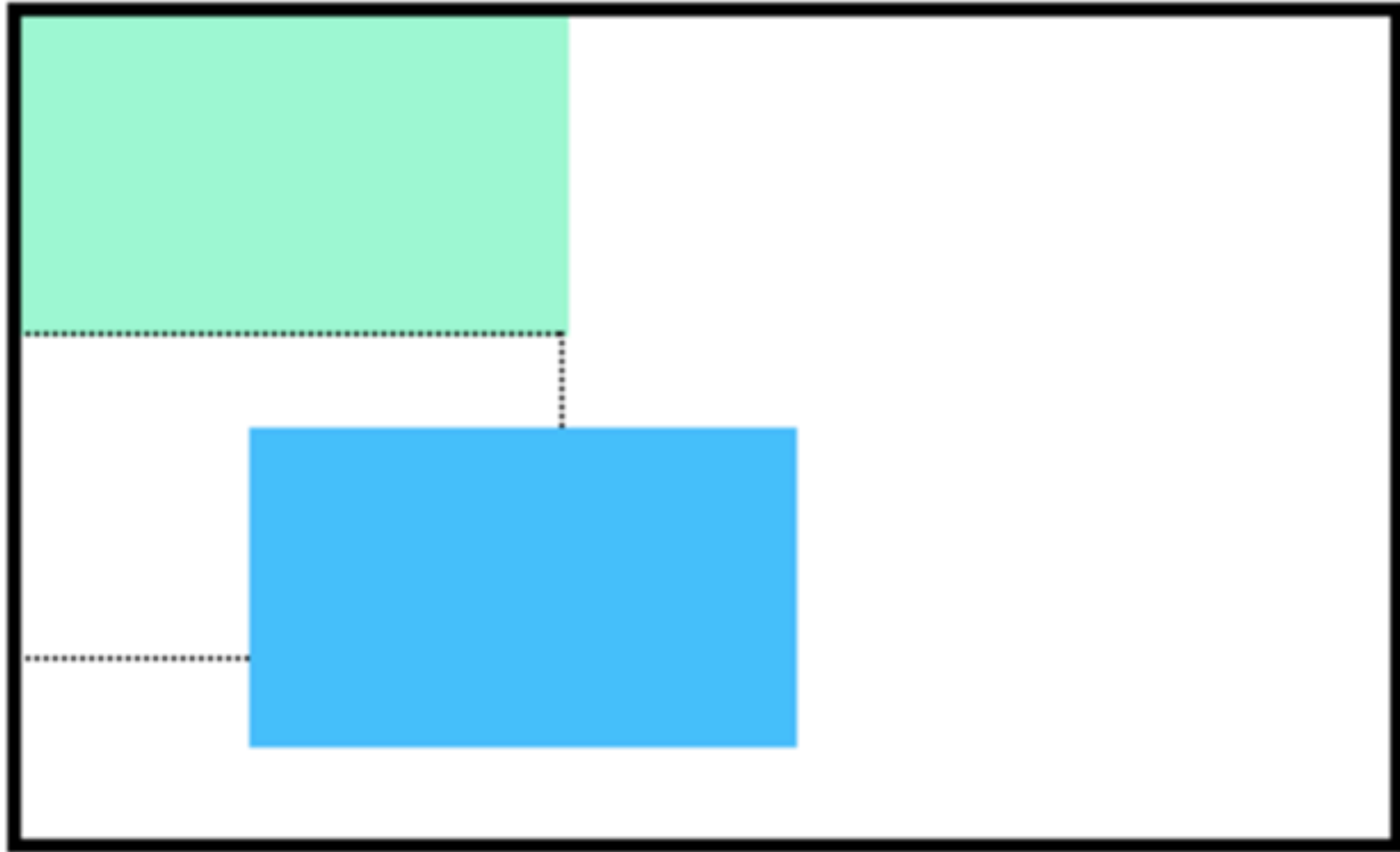
The screenshot shows a code editor window with two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, displaying the CSS code for the '.box-bottom' class. The code is as follows:

```
1 .box-bottom {  
2   background-color: DeepSkyBlue;  
3   position: relative;  
4   top: 20px;  
5   left: 50px;  
6 }
```

The status bar at the bottom indicates 'Line 6, Column 2; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

The div has been positioned using two of the four *offset properties*. The valid offset properties are:

- top** – moves the element down.
- bottom** – moves the element up.
- left** – moves the element right.
- right** – moves the element left.

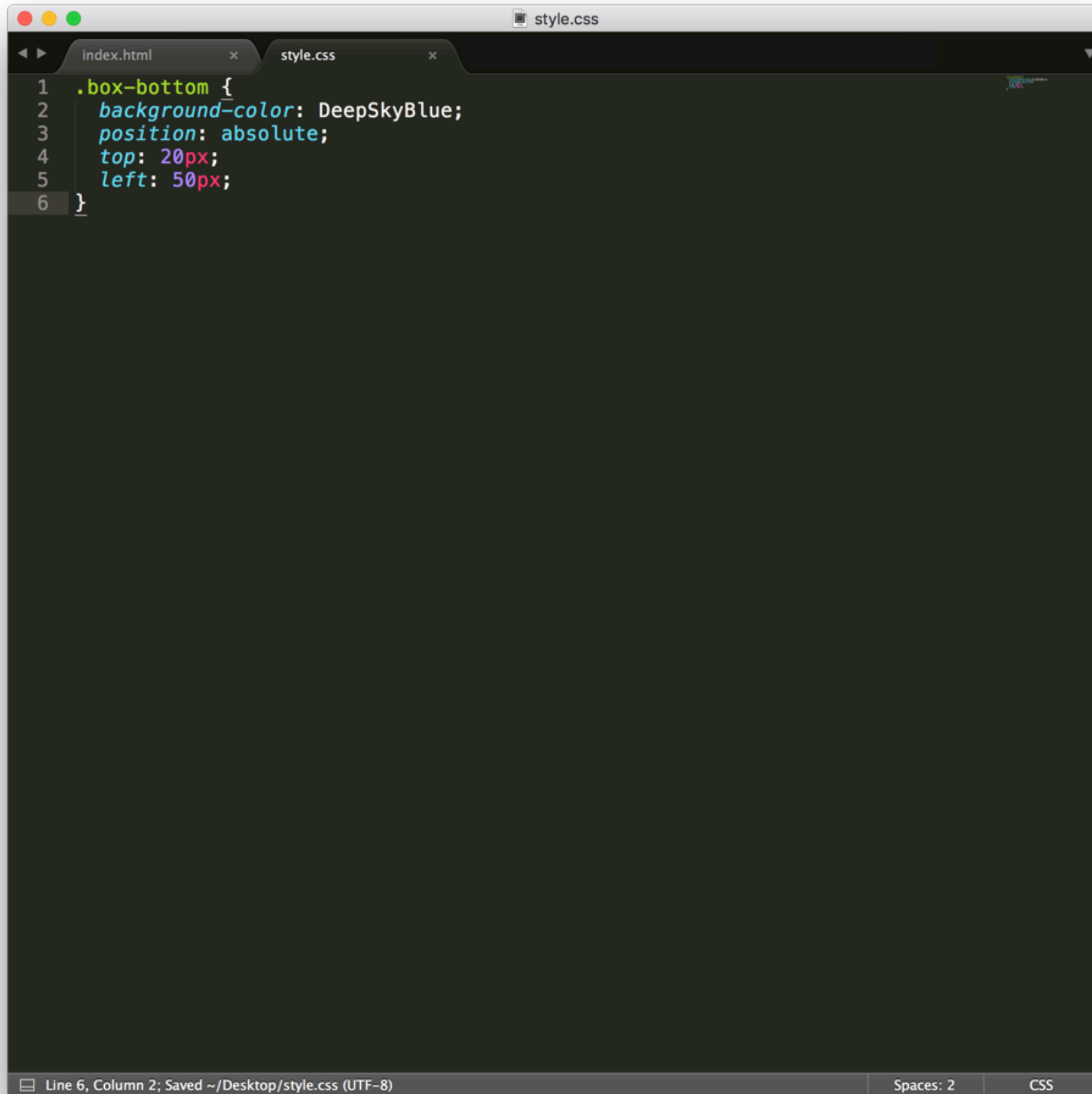


So in this example, the div will be moved down 20 pixels and to the right 50 pixels from its default static position. The image displays the new position of the box. The dotted line represents where the statically positioned (default) box was positioned.

Units for offset properties can be specified in pixels, ems, or percentages. Note that offset properties will not work if the position of the element is not set to **relative**.

Another way of modifying the position of an element is by setting its position to **absolute**

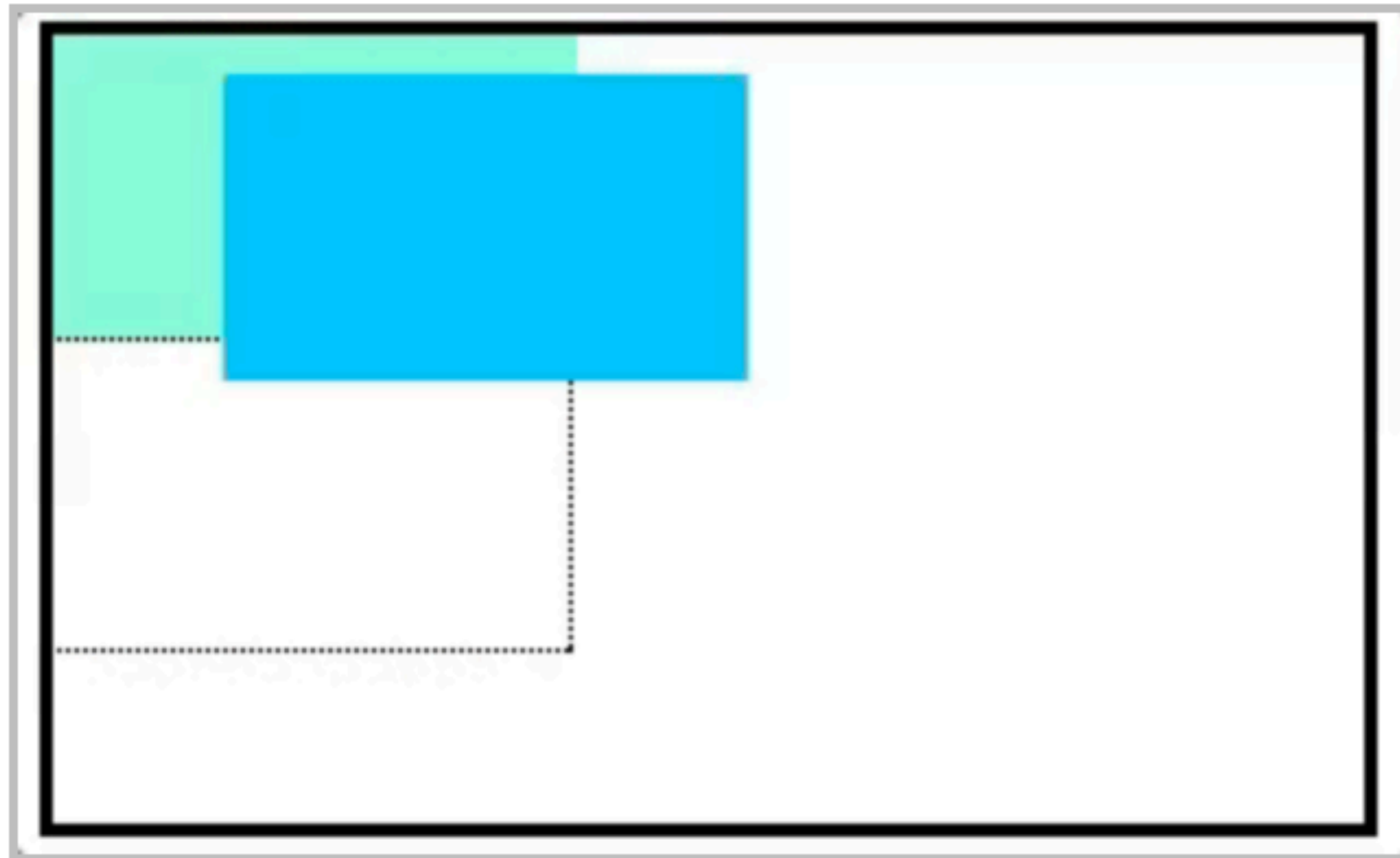
When an element's position is set to **absolute** all other elements on the page will *ignore* the element and act like it is not present on the page.

A screenshot of a code editor window with a dark theme. The window has two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 .box-bottom {  
2   background-color: DeepSkyBlue;  
3   position: absolute;  
4   top: 20px;  
5   left: 50px;  
6 }
```

The code is syntax-highlighted. The status bar at the bottom indicates 'Line 6, Column 2; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

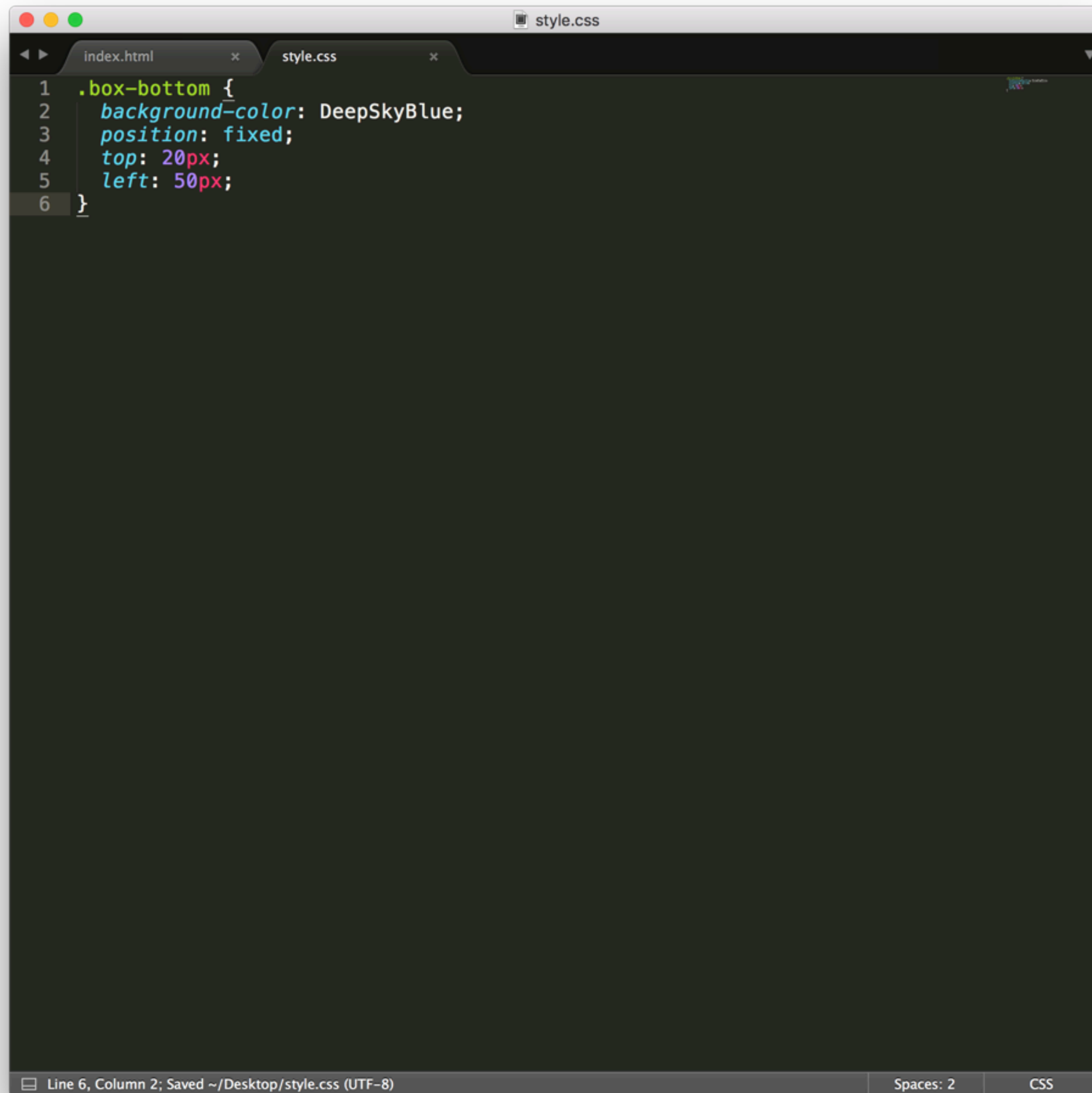
In the example above, the **.box-bottom** div will be moved down and right from the top left corner of the view. If offset properties weren't specified, the top box would be entirely covered by the bottom box.



The bottom box in this image (colored blue) is displaced from the top left corner of its container. It is 20 pixels lower and 50 pixels to the right of the top box.

When an element's position is set to **absolute**, as in the last exercise, the element will scroll out of view when a user scrolls.

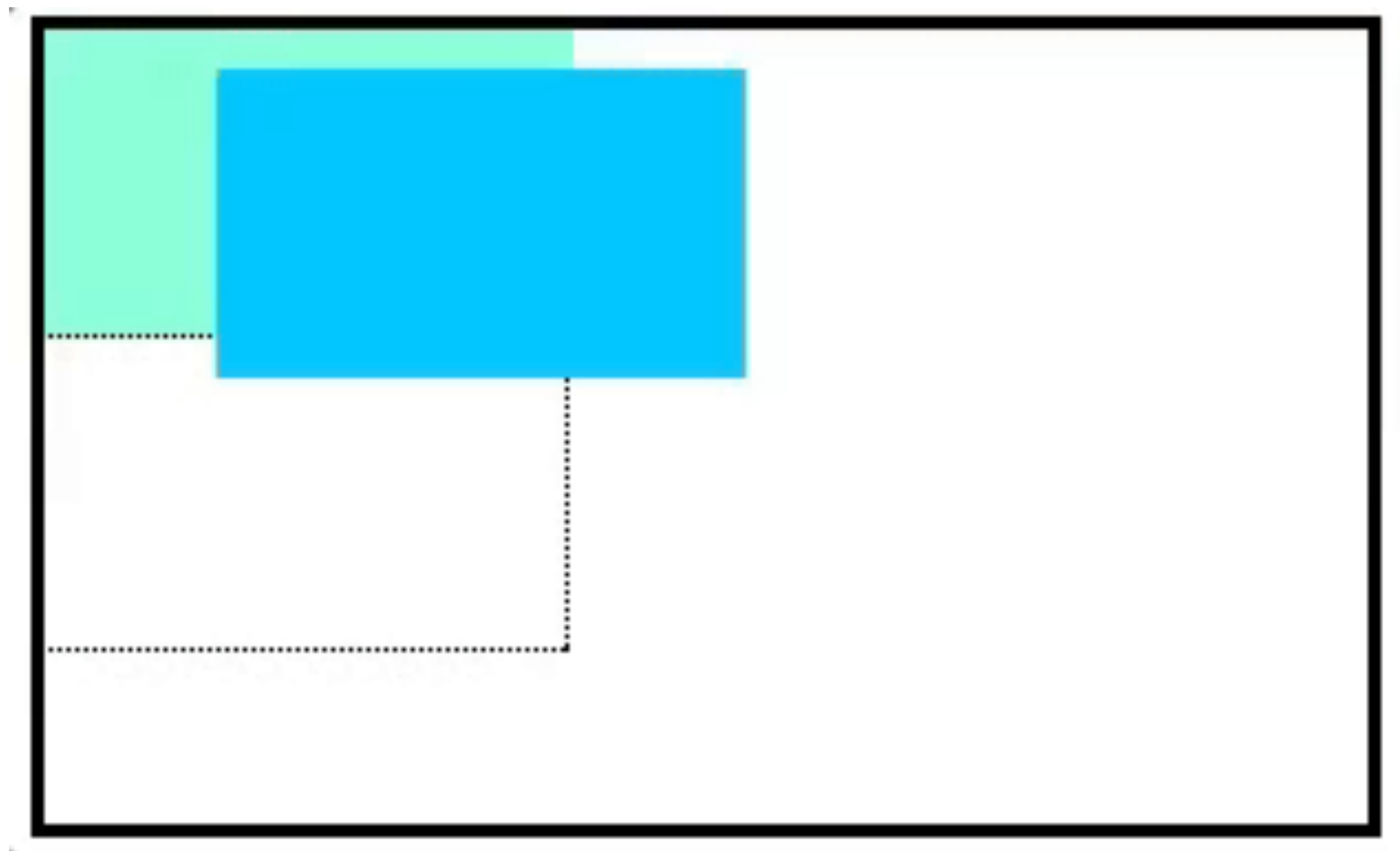
We can *fix* an element to a specific position on the page (regardless of user scrolling) by setting its position to **fixed**.

A screenshot of a code editor window with a dark theme. The window has two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing a CSS rule for '.box-bottom'. The code is as follows:

```
1 .box-bottom {  
2   background-color: DeepSkyBlue;  
3   position: fixed;  
4   top: 20px;  
5   left: 50px;  
6 }
```

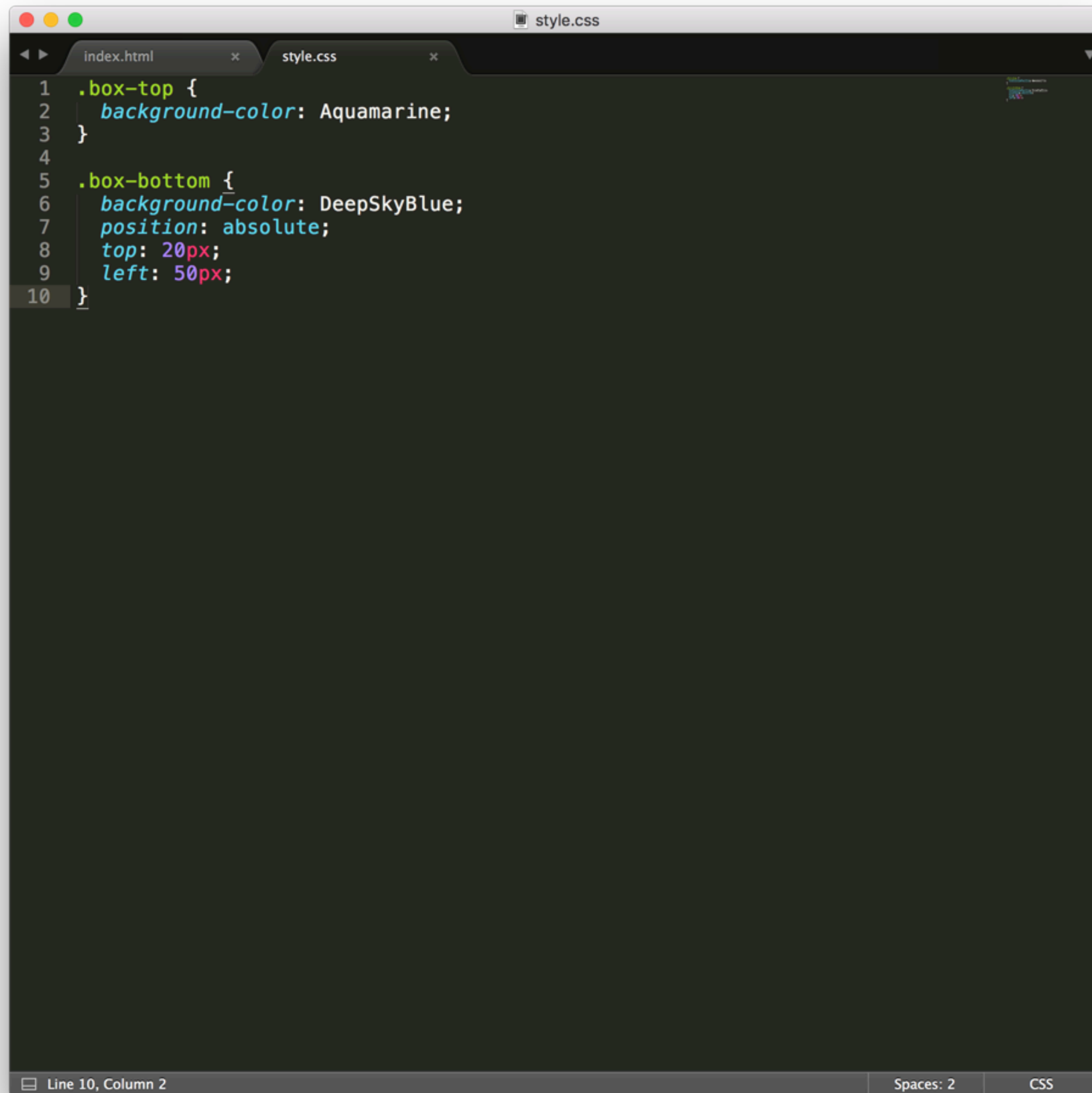
The code is color-coded: class names are green, property names are blue, values are purple, and keywords like 'fixed' are pink. The editor's status bar at the bottom shows 'Line 6, Column 2; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

The div will remain fixed to its position no matter where the user scrolls on the page.



When boxes on a web page have a combination of different positions, the boxes (and therefore, their content) can overlap with each other, making the content difficult to read or consume.

When boxes on a web page have a combination of different positions, the boxes (and therefore, their content) can overlap with each other, making the content difficult to read or consume.



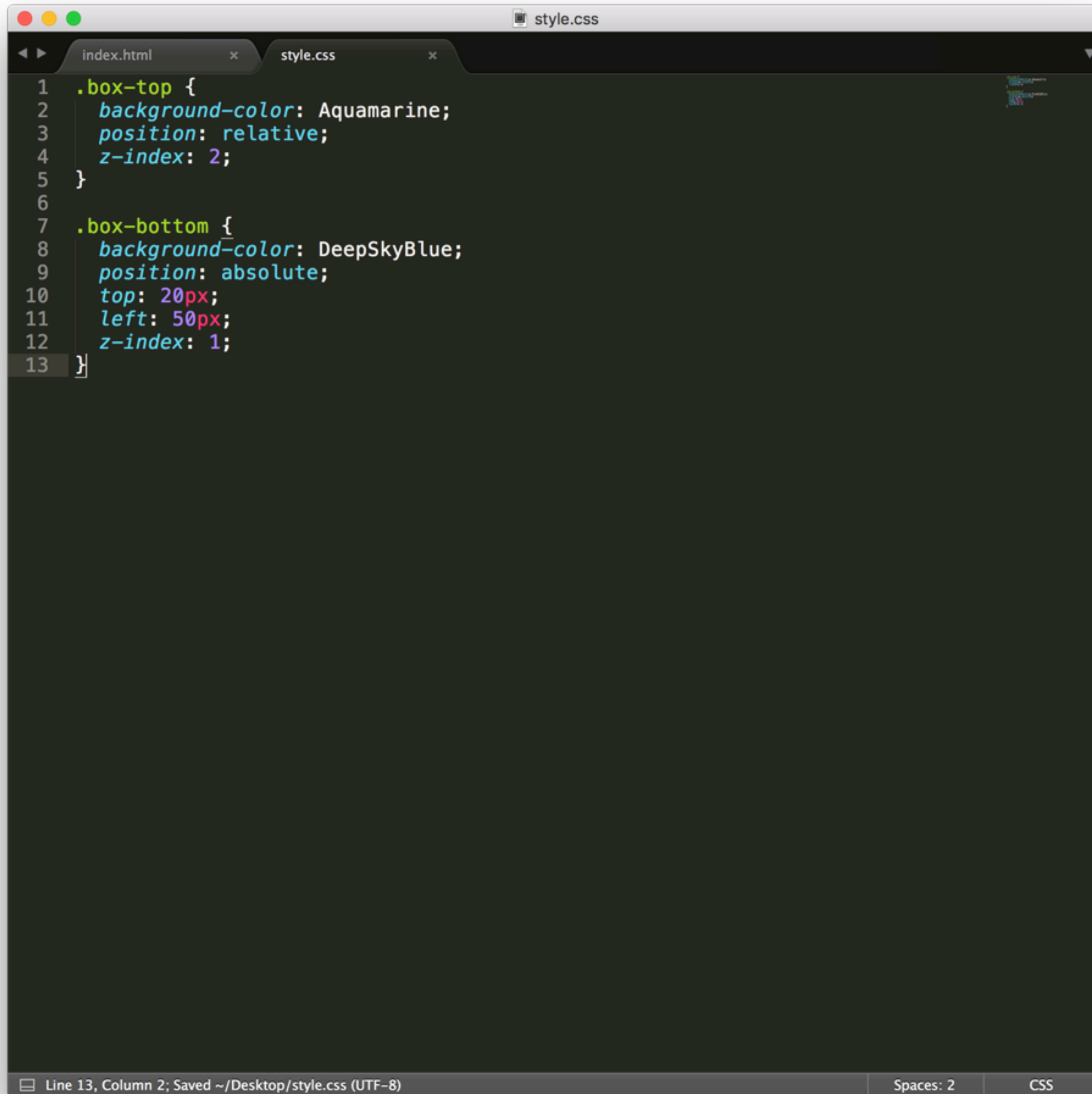
```
1 .box-top {  
2   background-color: Aquamarine;  
3 }  
4  
5 .box-bottom {  
6   background-color: DeepSkyBlue;  
7   position: absolute;  
8   top: 20px;  
9   left: 50px;  
10 }
```

The image shows a code editor window with two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, displaying the CSS code above. The code defines two classes: '.box-top' with a light blue background, and '.box-bottom' with a darker blue background, absolute positioning, and specific top and left offsets. The editor interface includes a dark theme, line numbers on the left, and a status bar at the bottom indicating 'Line 10, Column 2', 'Spaces: 2', and 'CSS'.

In this example, the **.box-bottom** div ignores the **.box-top** div and overlaps it as a user scrolls.

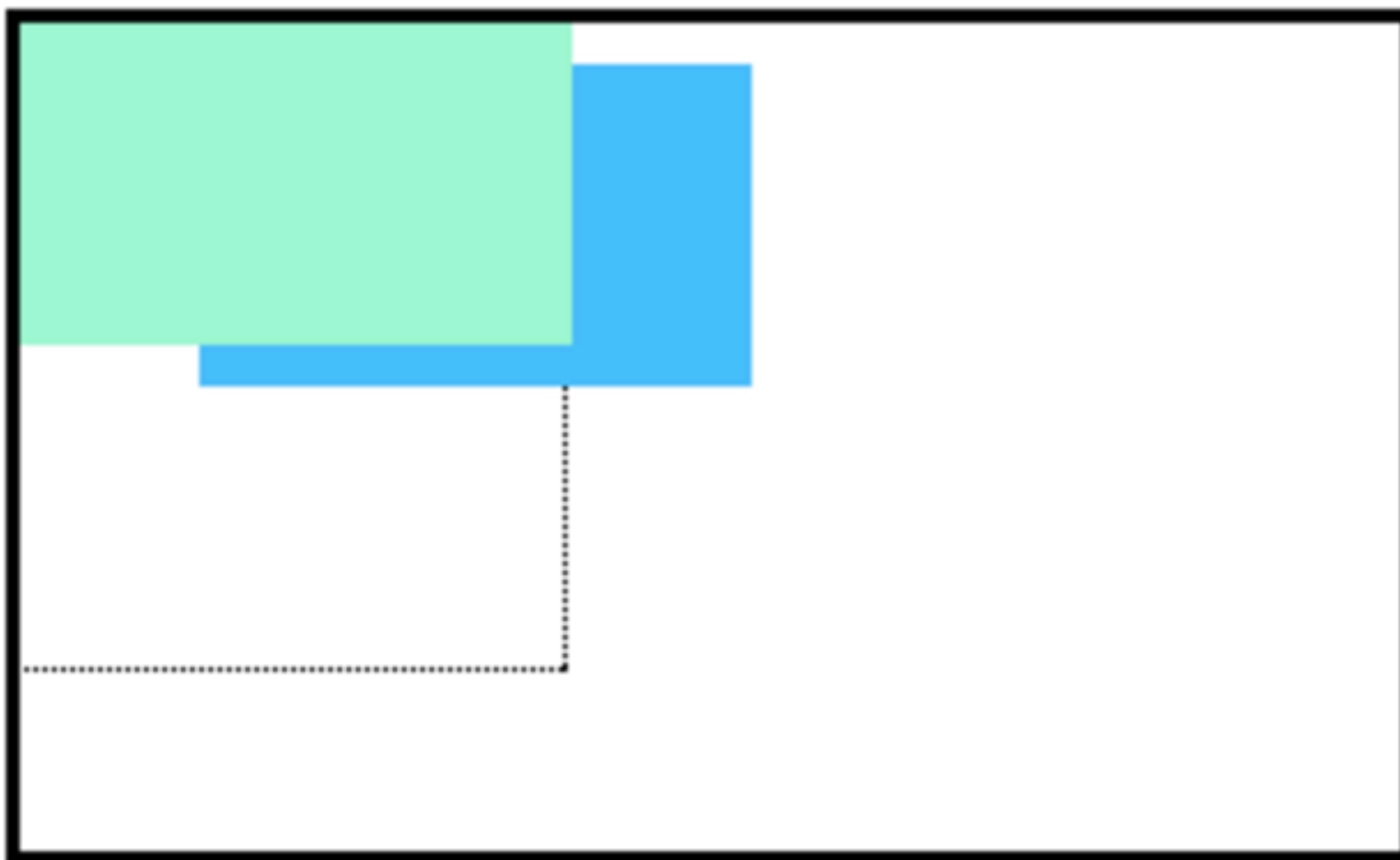
The **z-index** property controls how far "back" or how far "forward" an element should appear on the web page.

The **z-index** property accepts integer values.
Depending on their values, the integers instruct the browser on the order in which elements should be displayed on the web page.

A screenshot of a code editor window with two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing CSS code. The code defines two classes: '.box-top' and '.box-bottom'. The '.box-top' class has a background color of 'Aquamarine', a position of 'relative', and a z-index of '2'. The '.box-bottom' class has a background color of 'DeepSkyBlue', a position of 'absolute', a top offset of '20px', a left offset of '50px', and a z-index of '1'. The code is line-numbered from 1 to 13. The status bar at the bottom indicates 'Line 13, Column 2; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

```
1 .box-top {  
2   background-color: Aquamarine;  
3   position: relative;  
4   z-index: 2;  
5 }  
6  
7 .box-bottom {  
8   background-color: DeepSkyBlue;  
9   position: absolute;  
10  top: 20px;  
11  left: 50px;  
12  z-index: 1;  
13 }
```

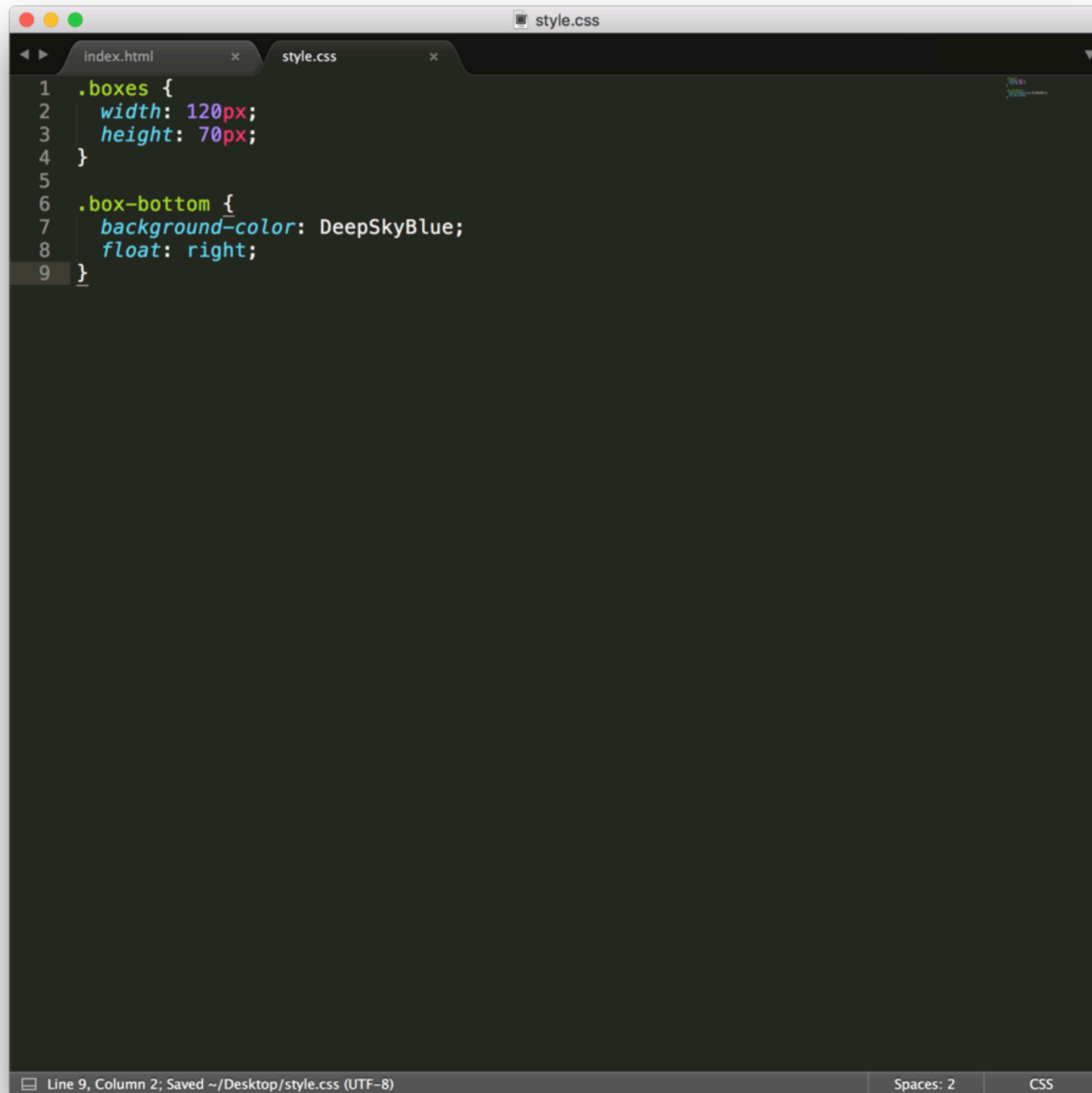
We set the **.box-top** position to relative and the z-index to 2. We changed position to **relative**, because the z-index property does *not* work on static elements. The z-index of **2** moves the **.box-top** element forward, because it is greater than the **.box-bottom** z-index, **1**



So far, we've learned how to specify the exact position of an element using offset properties. If you're simply interested in moving an element as far left or as far right as possible on the page, you can use the **float** property.

The **float** property can be set to one of two values:

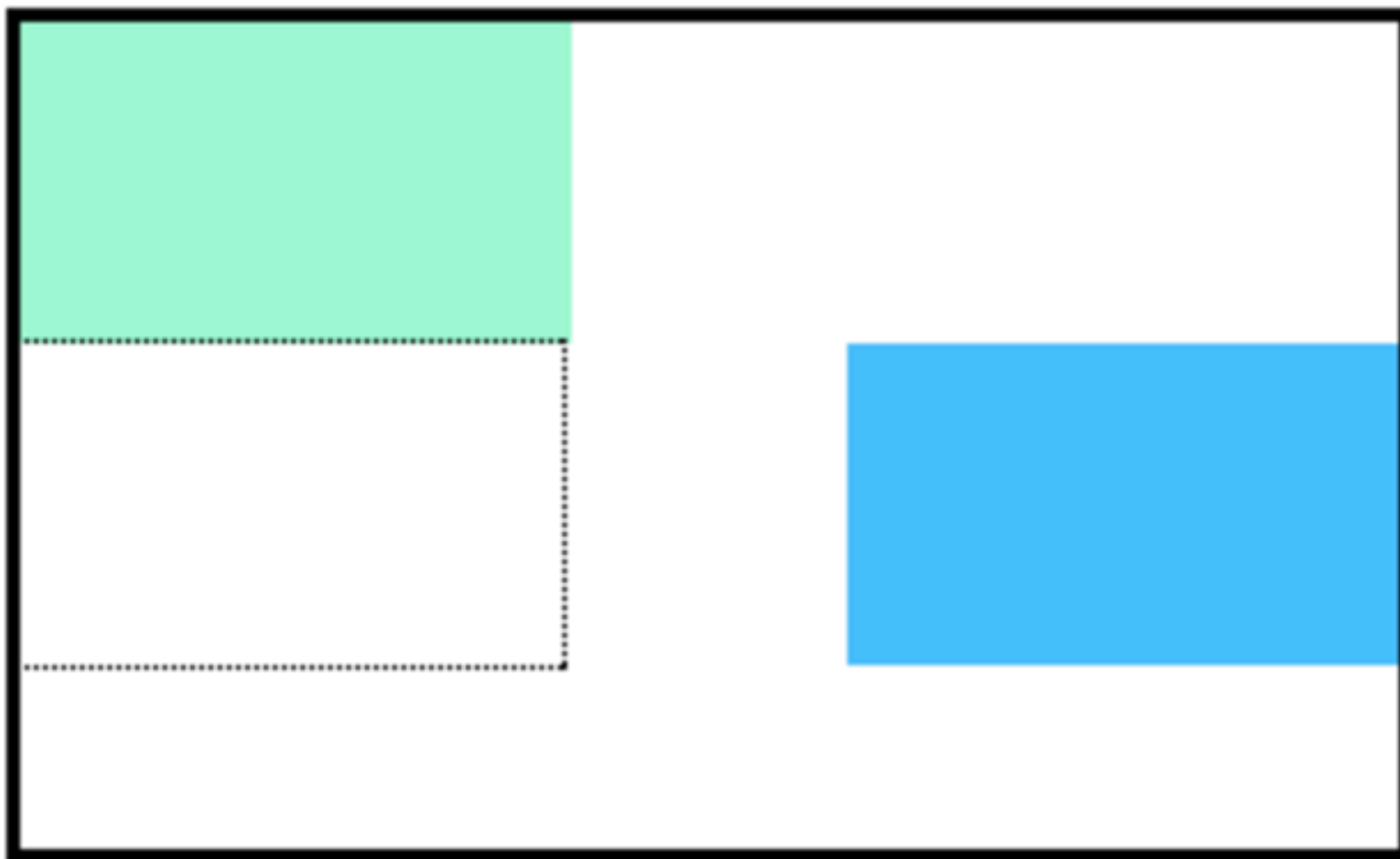
1. **left** – this value will move, or float, elements as far left as possible.
2. **right** – this value will move elements as far right as possible.



```
1 .boxes {
2   width: 120px;
3   height: 70px;
4 }
5
6 .box-bottom {
7   background-color: DeepSkyBlue;
8   float: right;
9 }
```

The image shows a code editor window with two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, displaying the CSS code. The code defines two classes: '.boxes' with width 120px and height 70px, and '.box-bottom' with a background color of DeepSkyBlue and a float of right. The editor has a dark theme and a status bar at the bottom indicating 'Line 9, Column 2; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

Here we float the **.box-bottom** element to the **right**. This works for static and relative positioned elements.



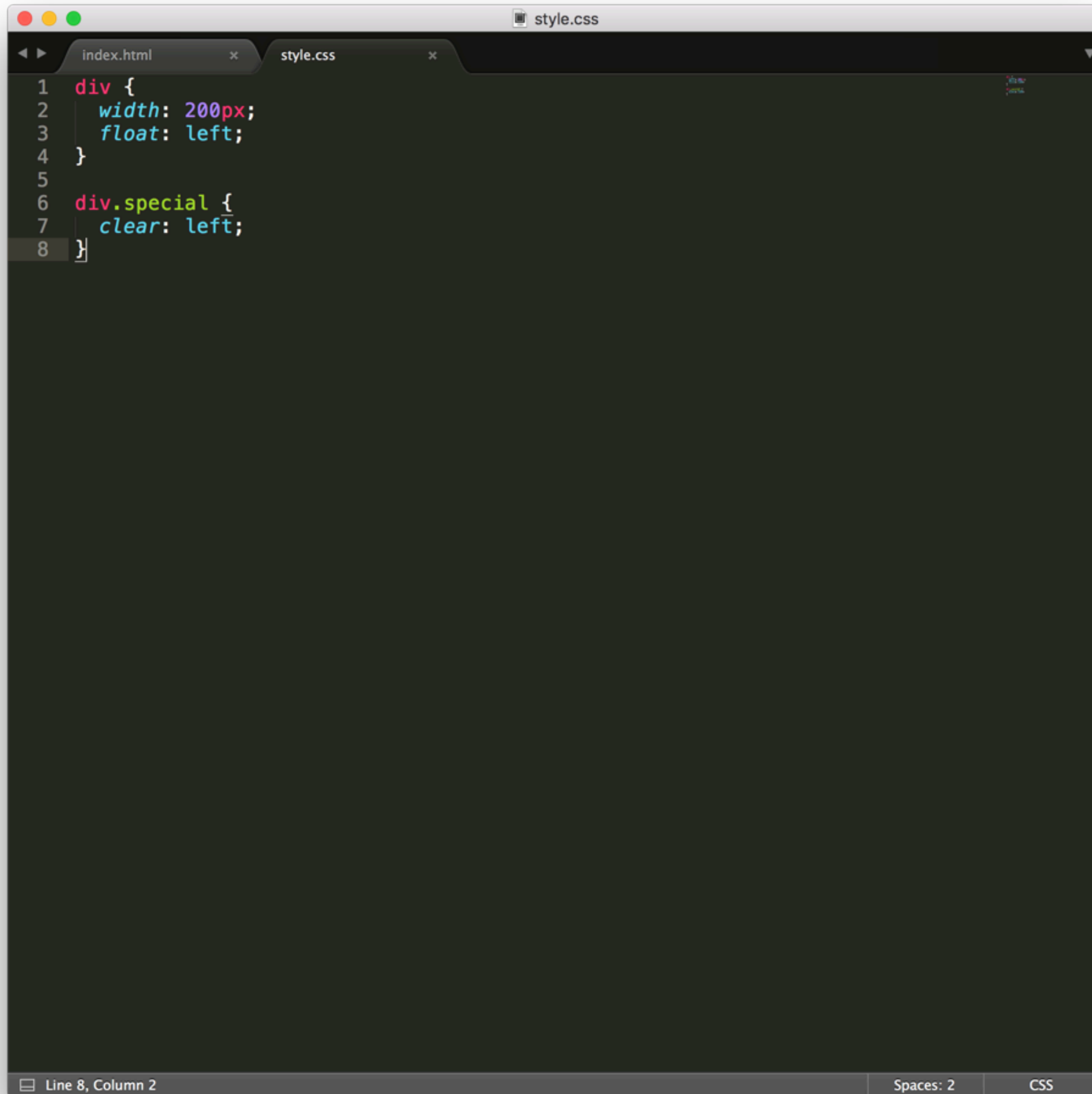
Floated elements must have a width specified, as in the example above. Otherwise, the element will assume the full width of its containing element, and changing the float value will not yield any visible results.

The **float** property can also be used to float multiple elements at once. However, when multiple floated elements have different heights, it can affect their layout on the page. Specifically, elements can "bump" into each other and not allow other elements to properly move to the left or right.

The **clear** property specifies how elements should behave when they bump into each other on the page.

It can take on one of the following values:

1. **left** — the left side of the element will not touch any other element within the same containing element.
2. **right** — the right side of the element will not touch any other element within the same containing element.
3. **both** — neither side of the element will touch any other element within the same containing element.
4. **none** — the element can touch either side.

A screenshot of a code editor window with a dark theme. The window has two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 div {  
2   width: 200px;  
3   float: left;  
4 }  
5  
6 div.special {  
7   clear: left;  
8 }
```

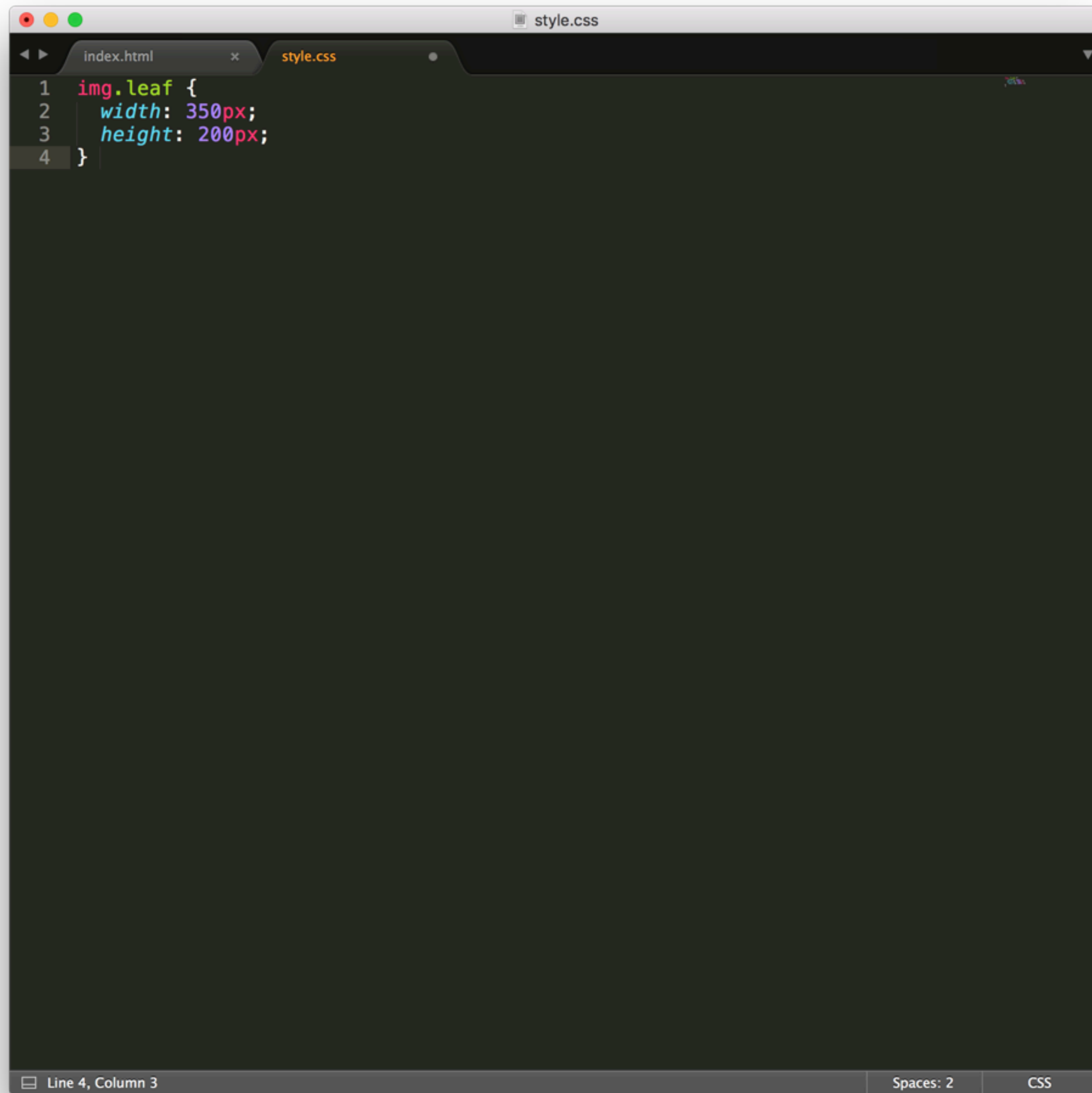
The code is syntax-highlighted: 'div' is red, '{' and '}' are black, 'width' is blue, '200px' is green, 'float' is blue, 'left' is blue, 'div.special' is green, and 'clear' is blue. The status bar at the bottom shows 'Line 8, Column 2', 'Spaces: 2', and 'CSS'.

All divs on the page are floated to the left side. The div with class **special** did not move all the way to the left because a taller div blocked its positioning. By setting its **clear** property to **left**, the **special** div will be moved all the way to the left side of the page.

Working with images

A few lessons ago we looked at how to add images to your web page using the `` element. While this technique is still valid, it's possible to also add images and style them using CSS techniques.

As with any other element, the dimensions of an image can be set with the **height** and **width** properties.

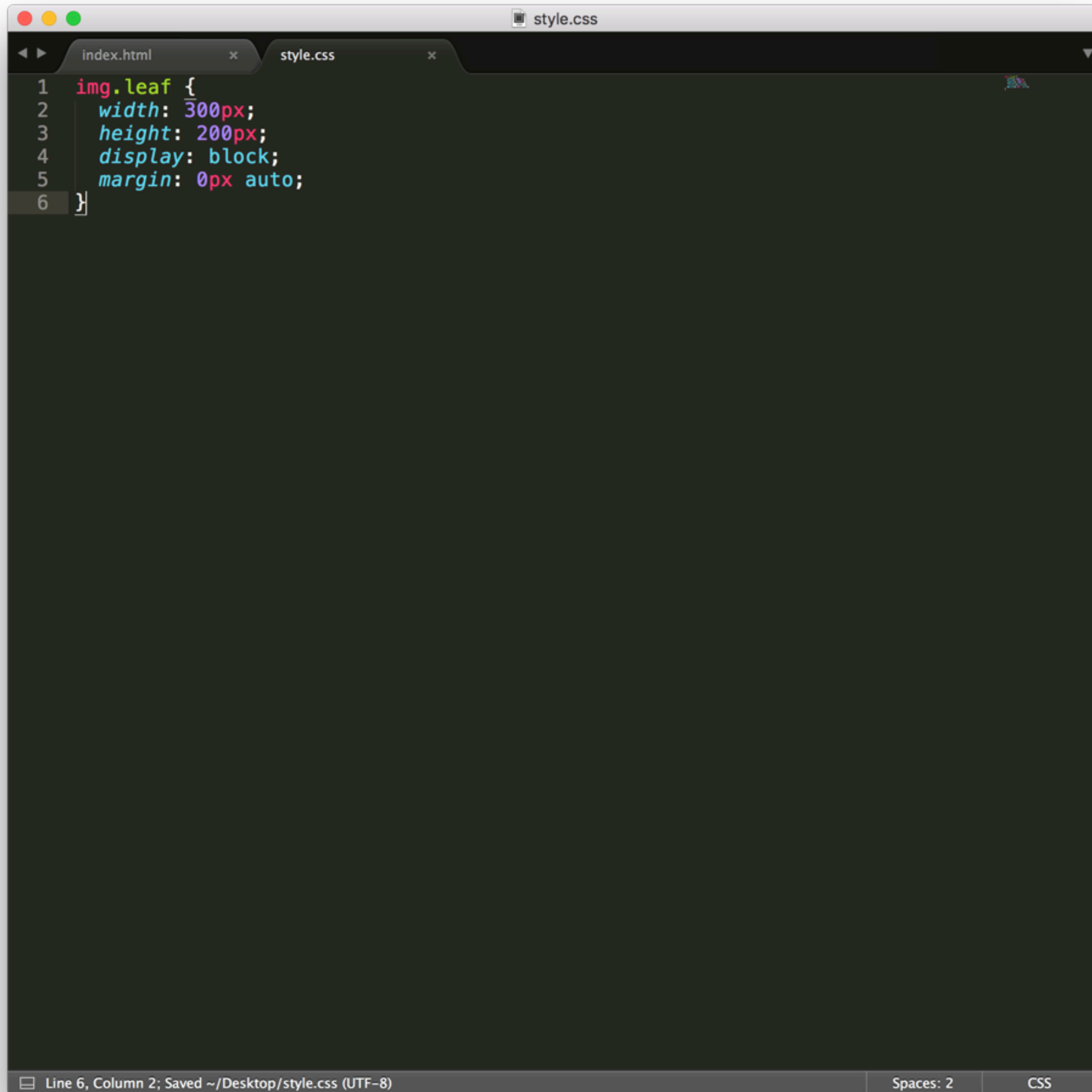
A screenshot of a code editor window with a dark theme. The window has a title bar with three colored buttons (red, yellow, green) and a tab labeled 'style.css'. The editor shows a CSS rule for 'img.leaf' with 'width: 350px;' and 'height: 200px;'. The code is syntax-highlighted: 'img.leaf' is in green, '{' is in blue, 'width:' is in blue, '350px;' is in purple, 'height:' is in blue, '200px;' is in purple, and '}' is in blue. Line numbers 1 through 4 are visible on the left. The status bar at the bottom shows 'Line 4, Column 3', 'Spaces: 2', and 'CSS'.

```
1  img.leaf {  
2    width: 350px;  
3    height: 200px;  
4  }
```

Specifying the dimensions of an image helps the browser determine how much space should be reserved for the image.

Note: Images should be saved at the dimensions they will be displayed in on the web page. Using dimensions for an image that exceed the original dimensions will distort the image.

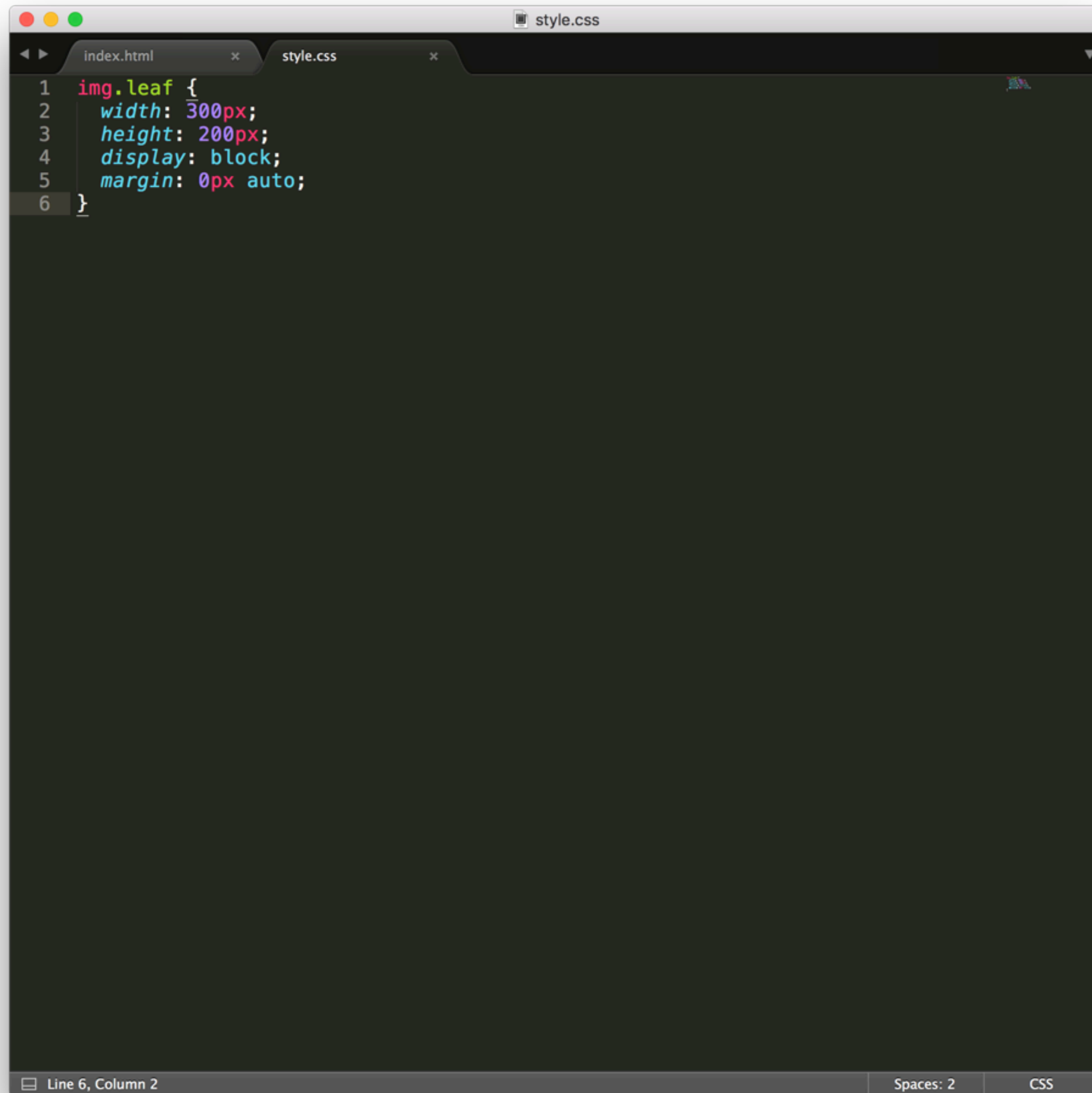
Images can also be centered, but first, their default behavior must be changed. By default, images are inline elements. For images to center properly, they must behave as block-level elements.

A screenshot of a code editor window with a dark theme. The window has two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing a CSS rule for 'img.leaf'. The code is as follows:

```
1 img.leaf {  
2   width: 300px;  
3   height: 200px;  
4   display: block;  
5   margin: 0px auto;  
6 }
```

The code is color-coded: 'img.leaf' is in green, '{' is in blue, 'width' is in blue, '300px' is in pink, 'height' is in blue, '200px' is in pink, 'display' is in blue, 'block' is in pink, 'margin' is in blue, '0px' is in pink, 'auto' is in pink, and '}' is in blue. The status bar at the bottom shows 'Line 6, Column 2; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

The image's **display** property has been set to **block**. Now the image can be aligned as a block-level element.

A screenshot of a code editor window with a dark theme. The window has two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing a CSS rule for 'img.leaf'. The code is as follows:

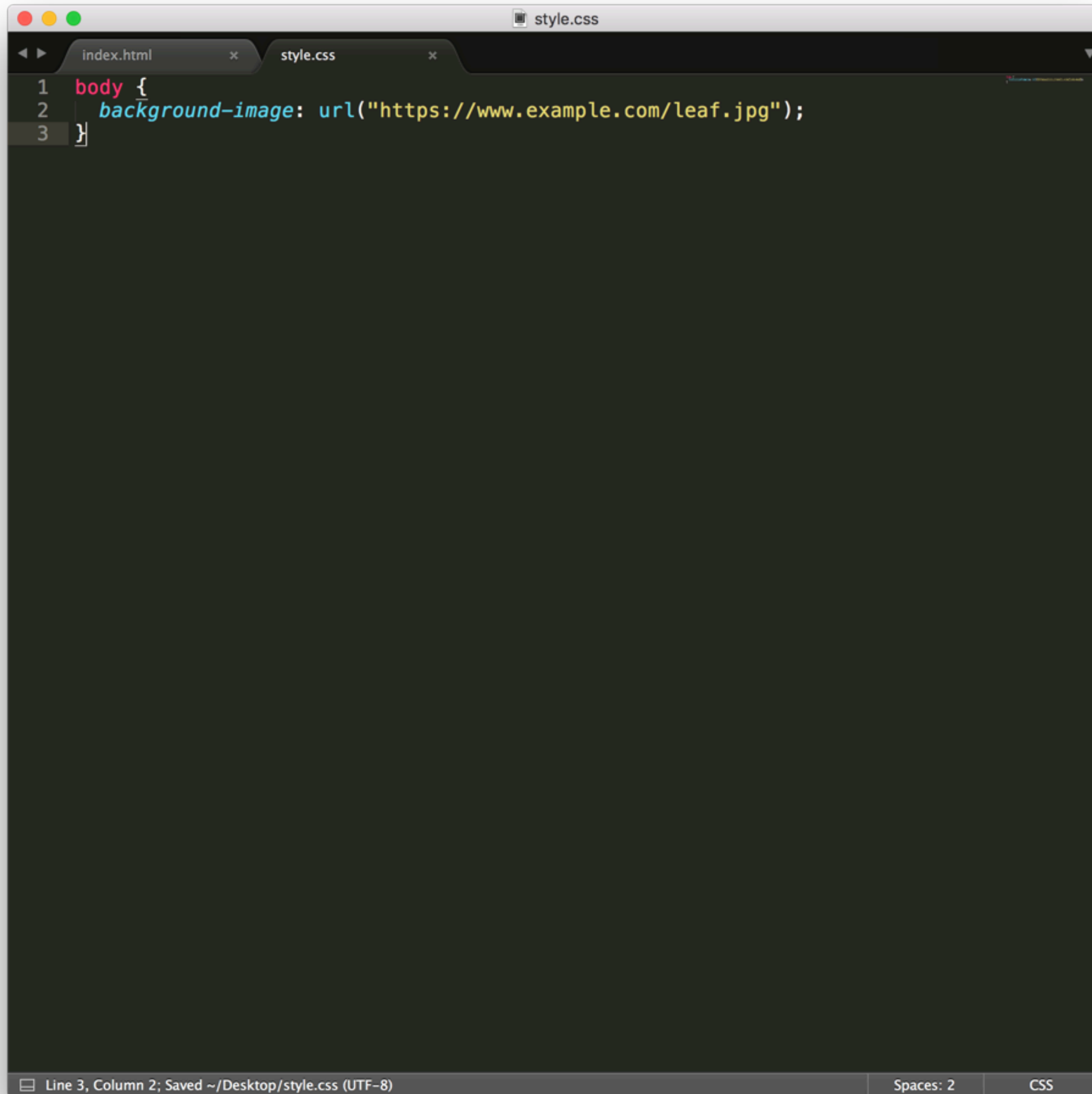
```
1 img.leaf {  
2   width: 300px;  
3   height: 200px;  
4   display: block;  
5   margin: 0px auto;  
6 }
```

The code is syntax-highlighted. The 'img.leaf' selector is in green, and the properties are in blue. The values '300px', '200px', 'block', and '0px auto' are in red. The closing brace on line 6 is in green. The status bar at the bottom shows 'Line 6, Column 2', 'Spaces: 2', and 'CSS'.

The image is aligned using the **margin** property. The top and bottom margins of the image's box are set to **0** margin. The left and right margins are set to **auto**, which automatically sets the exact amount of margin needed on the left and right sides in order to center the image.

Note: To align images to the left or right side of a page, you can use the **float** property we learned about earlier.

Images can also be added to the backgrounds of HTML elements with the **background-image** property.

A screenshot of a code editor window with a dark theme. The window has two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing three lines of CSS code. Line 1: 'body {'; Line 2: 'background-image: url("https://www.example.com/leaf.jpg");'; Line 3: '}'. The status bar at the bottom shows 'Line 3, Column 2; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

```
1 body {  
2   background-image: url("https://www.example.com/leaf.jpg");  
3 }
```

The **background-image** can be set to a value of **url()**.

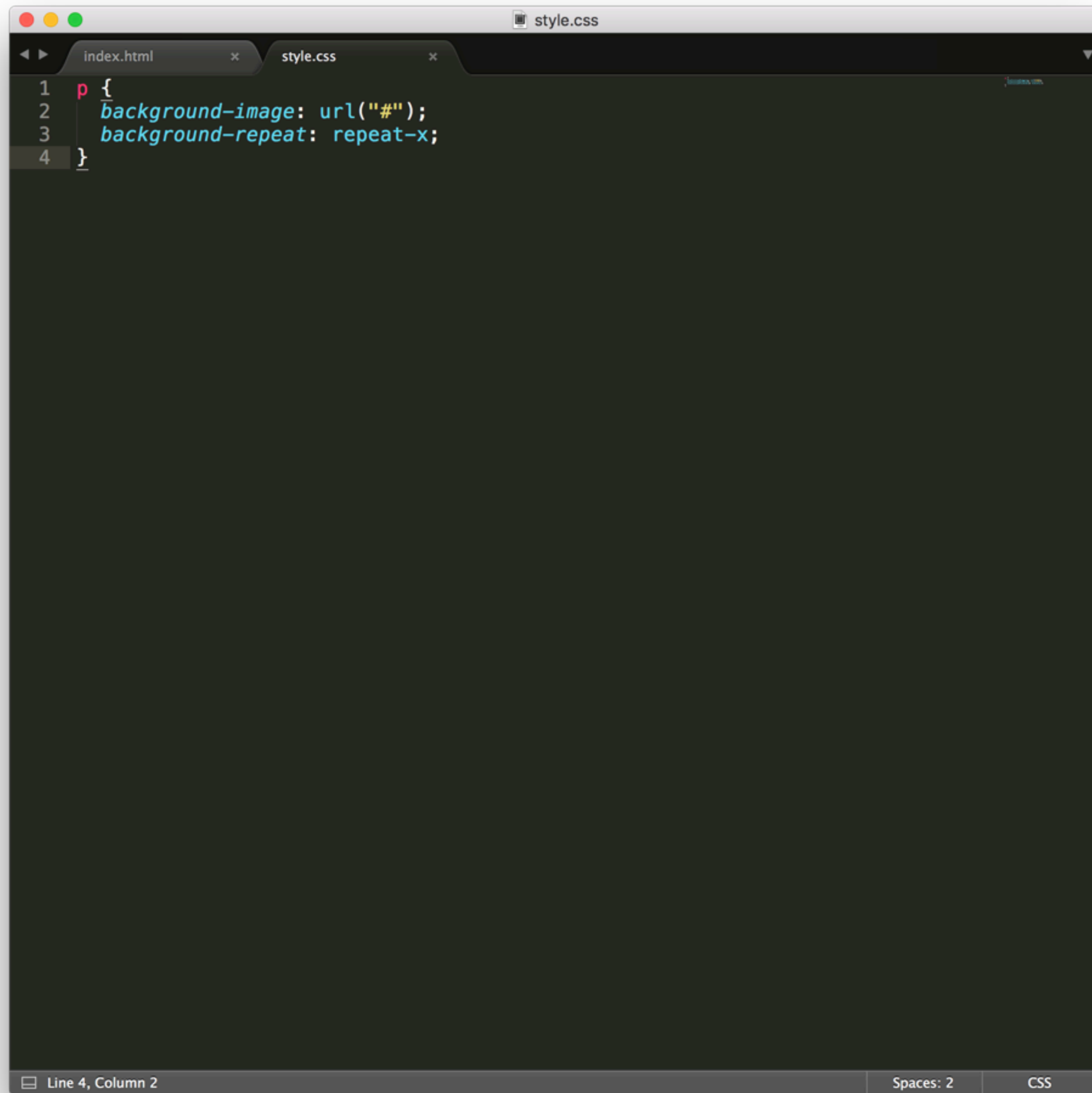
url() – contains the URL of the image, enclosed in double quotes.

In the example above, the background of the body will be set to the image specified in double quotes. This technique can be used on nearly any HTML element.

In the last example, if the image is not big enough to fill up the container you've added it to, it will repeat itself. You can control how a background image *repeats* with the **background-repeat** property.

This property can take one of four values:

1. **repeat** – the default value — the image will repeat horizontally and vertically.
2. **repeat-x** – the background image will be repeated only along the x-axis (horizontally).
3. **repeat-y** – the background image will be repeated only along the y-axis (vertically).
4. **no-repeat** – the background image will not be repeated at all and will appear only once.

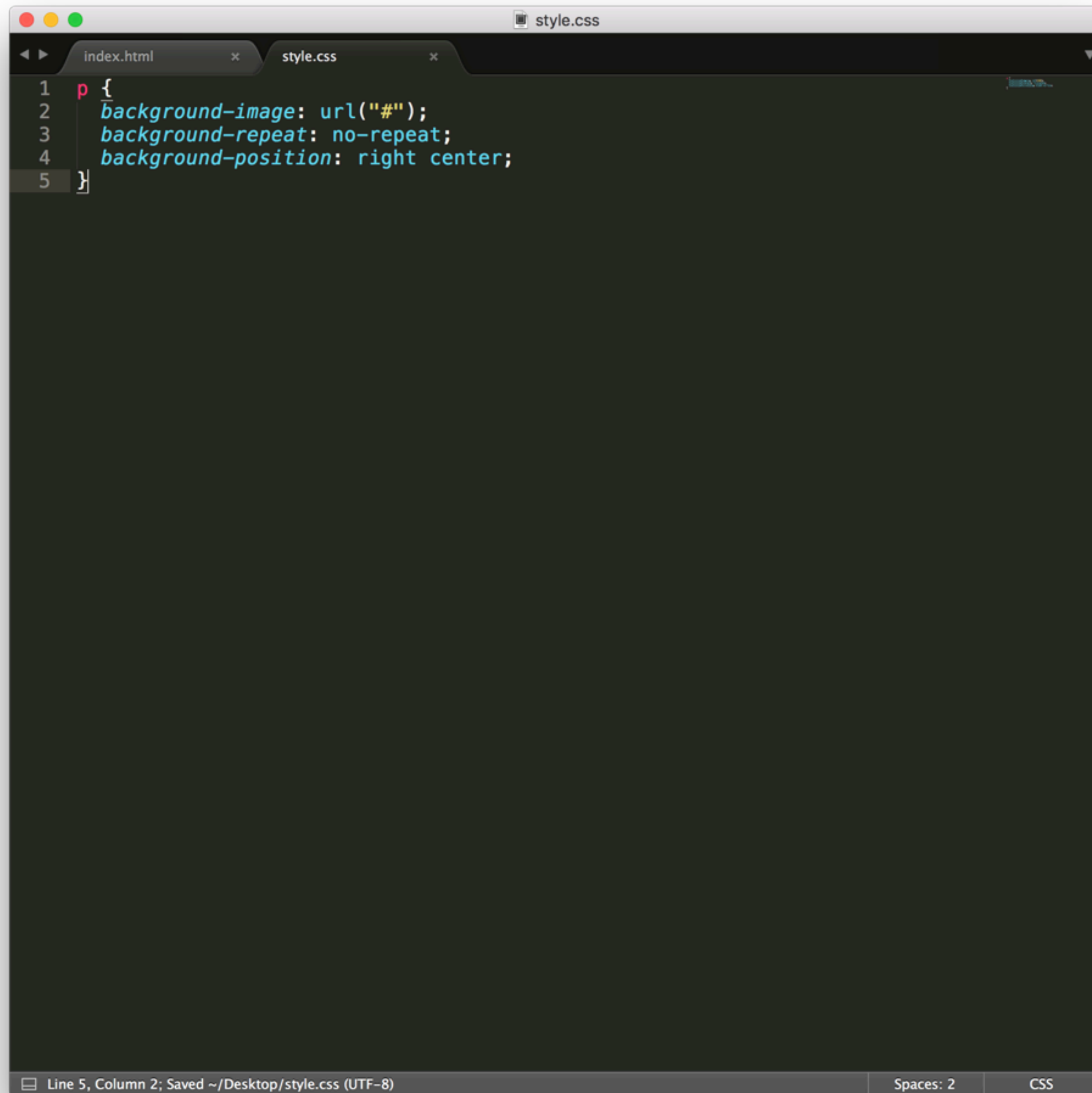
A screenshot of a code editor window with a dark theme. The window has a title bar with three colored buttons (red, yellow, green) and a tab labeled 'style.css'. Inside the editor, there are two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 p {  
2   background-image: url("#");  
3   background-repeat: repeat-x;  
4 }
```

The code is syntax-highlighted: 'p' is red, '{' is blue, 'background-image' is blue, 'url' is yellow, '#' is yellow, ';' is blue, 'background-repeat' is blue, 'repeat-x' is blue, and '}' is blue. The line numbers 1, 2, 3, and 4 are on the left. The status bar at the bottom shows 'Line 4, Column 2', 'Spaces: 2', and 'CSS'.

The paragraph's background image will repeat horizontally in this example.

When a background image is not repeated, its position can be modified with the **background-position** property.

A screenshot of a code editor window with a dark theme. The title bar shows 'style.css'. The editor has two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 p {  
2   background-image: url("#");  
3   background-repeat: no-repeat;  
4   background-position: right center;  
5 }
```

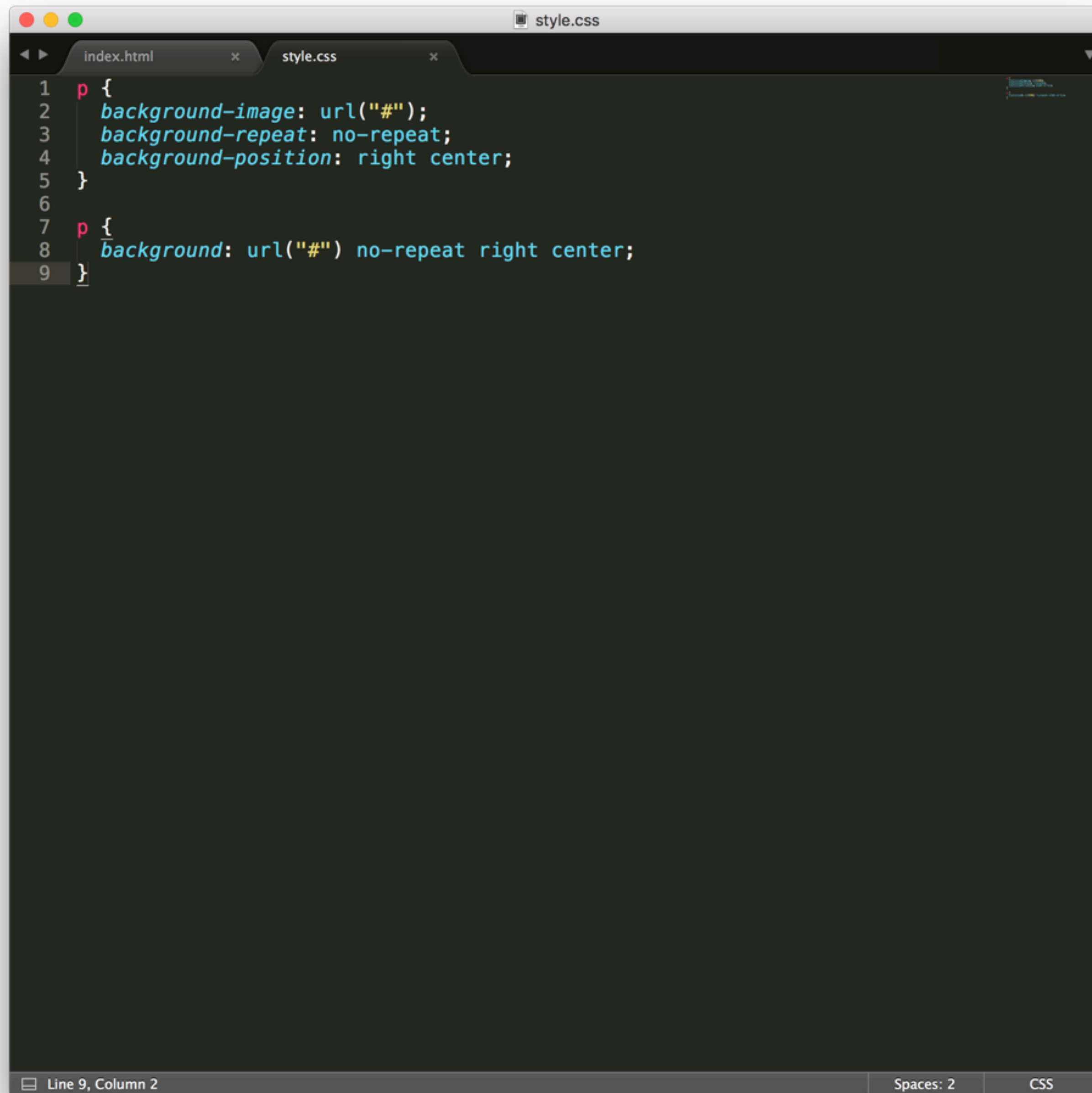
The status bar at the bottom indicates 'Line 5, Column 2; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

A background image is positioned using a 3 by 3 grid (three rows, three columns), meaning there are 9 total possible positions for the image:

left top, left center, left bottom,
right top, right center, right
bottom, center top, center
bottom, center center.

Note: When setting this property, if only one value is specified, the second value will default to **center**.

CSS allows you to set all three properties at once using a shorthand property: `background`.

A screenshot of a code editor window with two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing CSS code for a paragraph element. The code is as follows:

```
1 p {  
2   background-image: url("#");  
3   background-repeat: no-repeat;  
4   background-position: right center;  
5 }  
6  
7 p {  
8   background: url("#") no-repeat right center;  
9 }
```

The code is color-coded: 'p' is red, '{' and '}' are blue, and the property names and values are green. The status bar at the bottom indicates 'Line 9, Column 2', 'Spaces: 2', and 'CSS'.

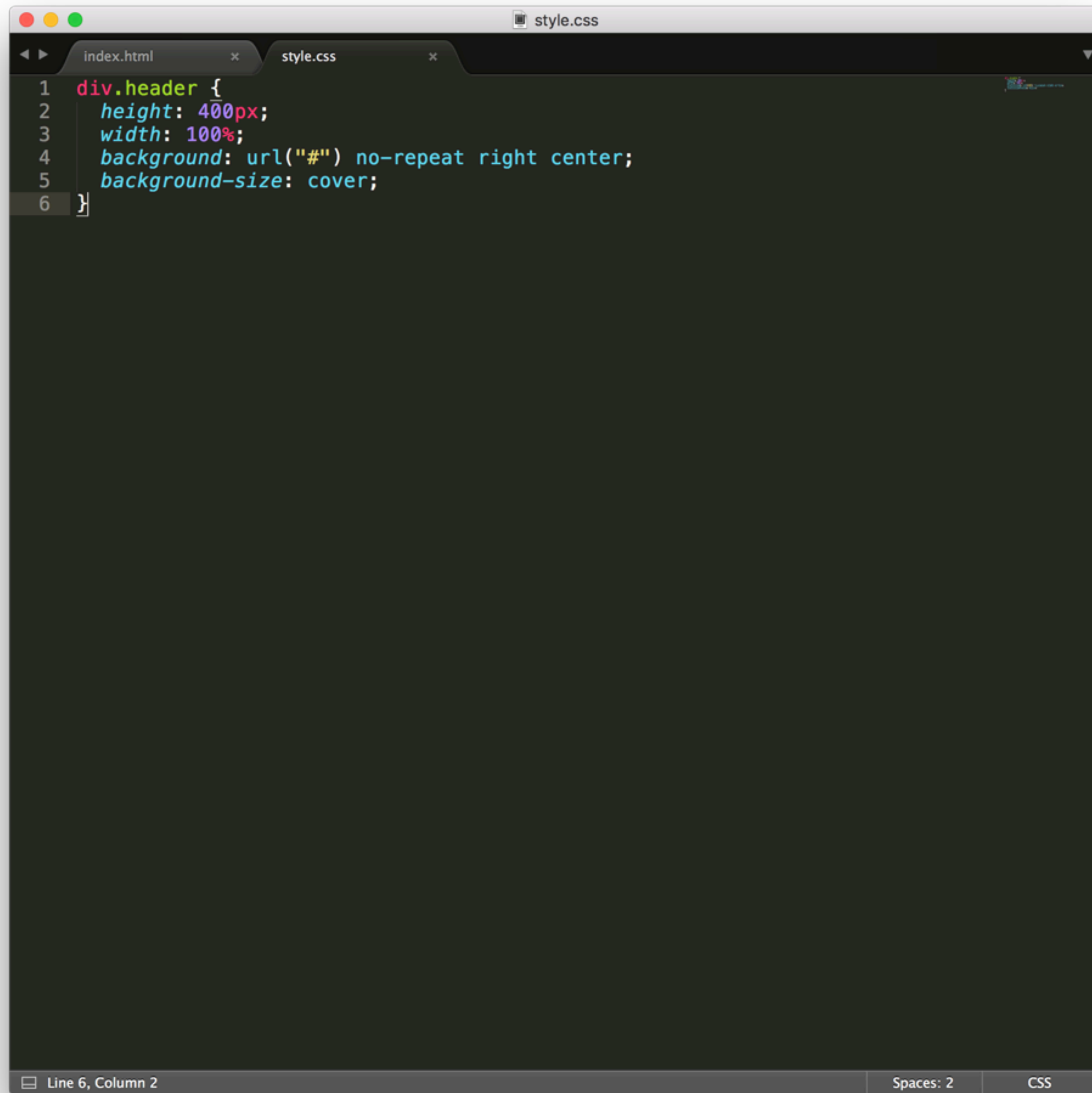
Note that the **background** property includes all of the properties that we previously styled individually: background image, repetition, and position (in that order). It's considered best practice to follow this order of values when setting the **background property**

To modify a background image's size, you can use the `background-size` property.

The two most common values of the **background-size** property are:

cover – expands the image as large as possible to cover the full width or height of a container. If the dimensions of the container (say, a div) are larger than the dimensions of the image, the image will become distorted.

contain – expands the image as large as possible, but the image will be **letterboxed**, which means it won't cover the full width or height of a container.

A screenshot of a code editor window with a dark theme. The window has a title bar with three colored buttons (red, yellow, green) and a tab labeled 'style.css'. Inside the editor, there are two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 div.header {  
2   height: 400px;  
3   width: 100%;  
4   background: url("#") no-repeat right center;  
5   background-size: cover;  
6 }
```

The code is syntax-highlighted. The status bar at the bottom shows 'Line 6, Column 2', 'Spaces: 2', and 'CSS'.

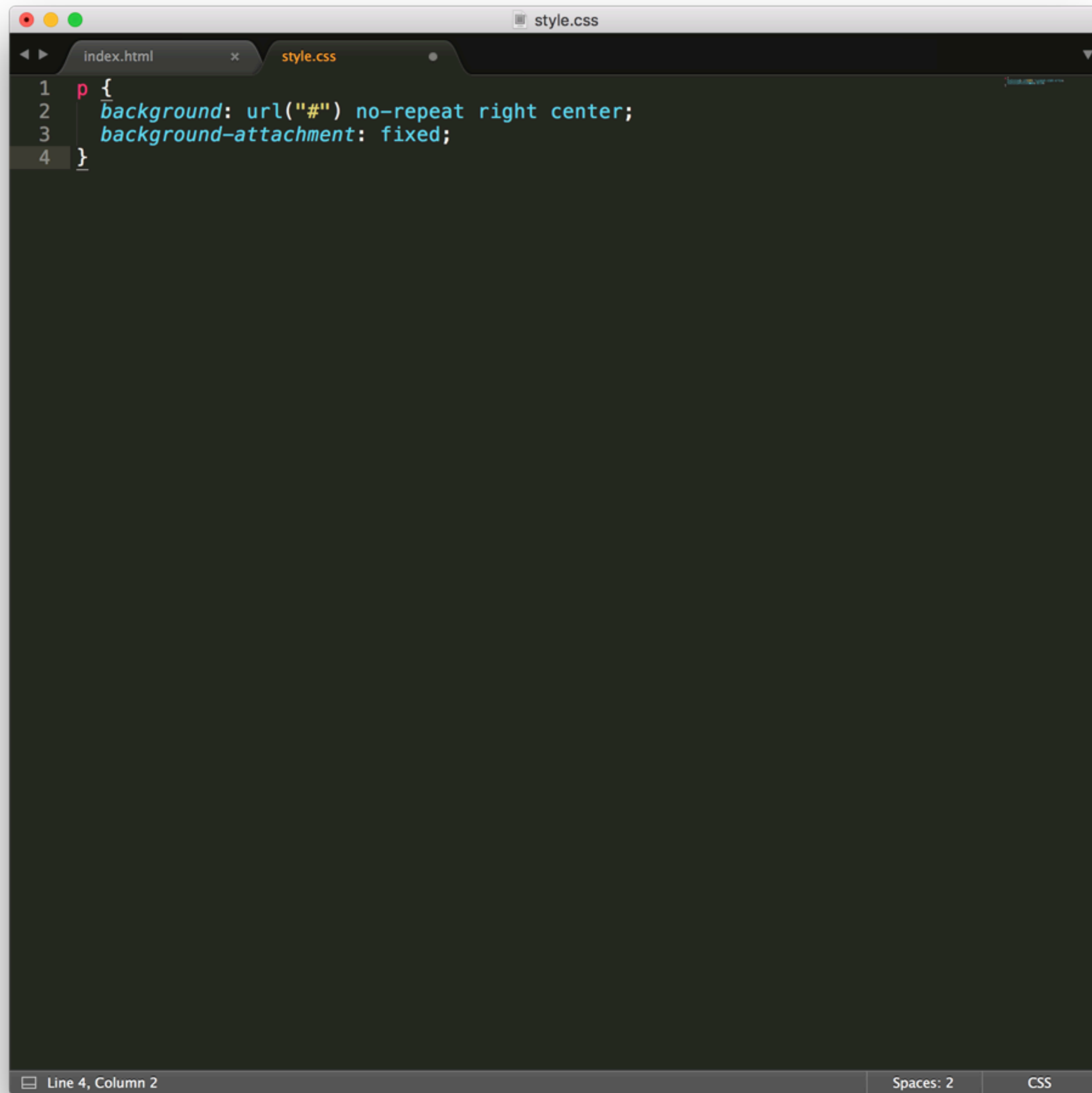
In this example, the background image will expand to cover the full size of the div.

With the **background-attachment** property, you can specify whether or not a background image should remain at one position on the web page or whether it should move up and down as the user scrolls through a web page.

The **background-attachment** property can take one of two values:

scroll – this value allows the image to move up and down as a user scrolls on the web page (this is the default value).

fixed – this value pins the image's position on the page.

A screenshot of a code editor window with a dark theme. The window has a title bar with three colored buttons (red, yellow, green) and a tab labeled 'style.css'. The editor shows four lines of CSS code for a paragraph selector. Line 1: 'p {'; Line 2: 'background: url("#") no-repeat right center;'; Line 3: 'background-attachment: fixed;'; Line 4: '}'. The status bar at the bottom indicates 'Line 4, Column 2', 'Spaces: 2', and 'CSS'.

```
1 p {  
2   background: url("#") no-repeat right center;  
3   background-attachment: fixed;  
4 }
```

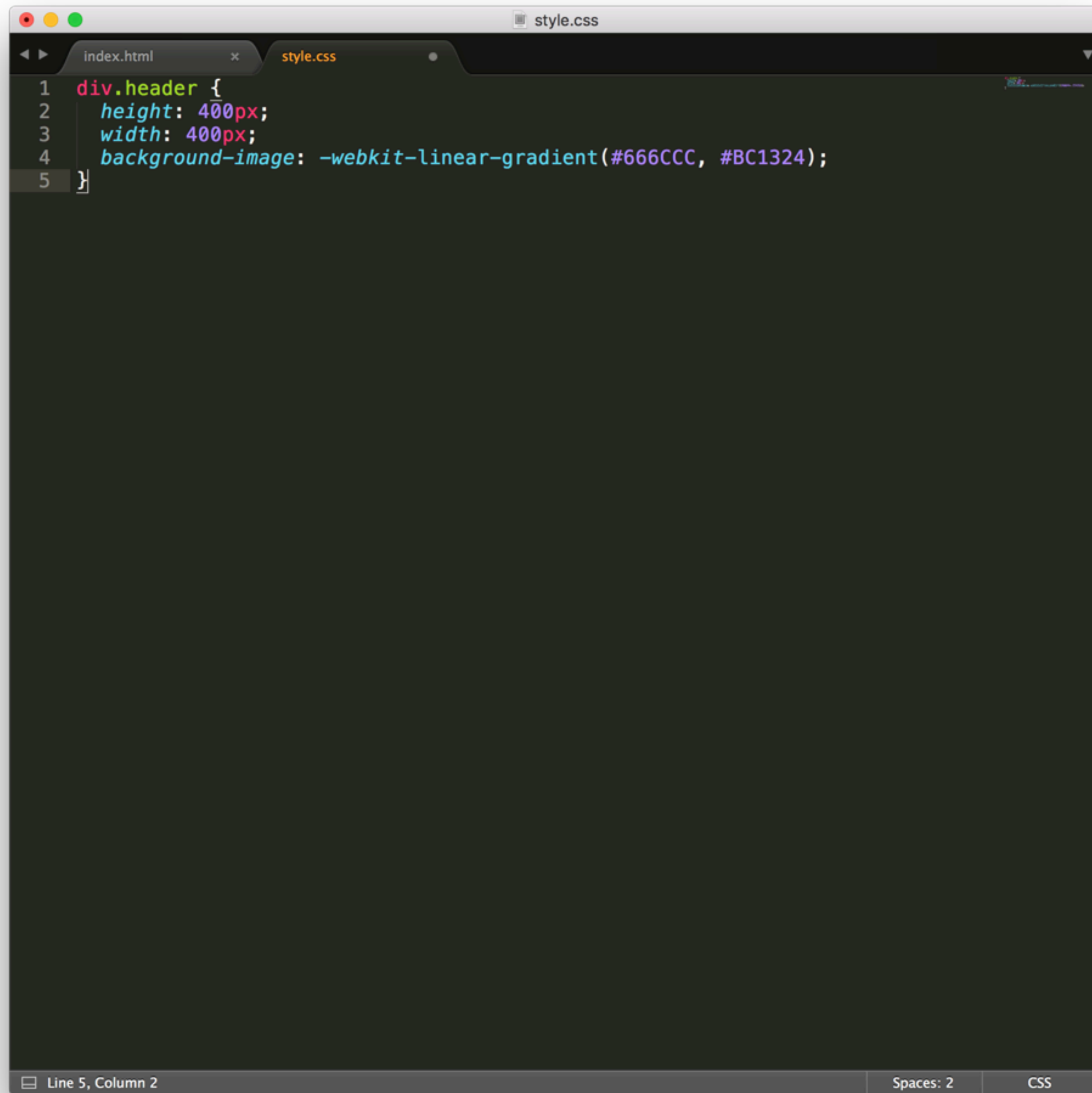
In this example, the background image will remained fixed as a user scrolls through the web page.

The **background-image** property you learned about earlier allows you to do more than simply set the background image of an element. It also lets you use color gradients in your web page.

Gradients are planned to be a part of the newest revision of CSS, CSS3. Due to the many browsers available, there is no standard way of defining a gradient using CSS (different browsers require different syntax). In this exercise, we'll look at one value supported on a couple of major browsers.

The **background-image** property can be set to the following value:

-webkit-linear-gradient() – this value accepts two arguments: the two colors the linear gradient will transition to and from. The colors are usually specified as hex color codes.

A screenshot of a code editor window with a dark theme. The window has a title bar with three colored buttons (red, yellow, green) and a tab labeled 'style.css'. The editor shows five lines of CSS code for a 'div.header' selector. The code is: 1. 'div.header {' 2. 'height: 400px;' 3. 'width: 400px;' 4. 'background-image: -webkit-linear-gradient(#666CCC, #BC1324);' 5. '}' The code is syntax-highlighted. The status bar at the bottom shows 'Line 5, Column 2', 'Spaces: 2', and 'CSS'.

```
1 div.header {  
2   height: 400px;  
3   width: 400px;  
4   background-image: -webkit-linear-gradient(#666CCC, #BC1324);  
5 }
```

In this example, a linear gradient will be created between the two hex colors specified.

You've learned how to add images to a web page using the `` element and the `background` property. What's the difference between these two methods? When should one method be used over another?

You've learned how to add images to a web page using the `` element and the `background` property. What's the difference between these two methods? When should one method be used over another?

The method used depends on the type of image.

Some images are part of the content of a web page (icons, logos, album photos, etc.) and they communicate important information to a user. Other images are intended simply to style a web page (header backgrounds, etc.), not to communicate important information.

When an image communicates important information, you can use the `` element and style the image using CSS, if needed.

When an image is intended to style a web page, you can use the `background` property and further style it with CSS.