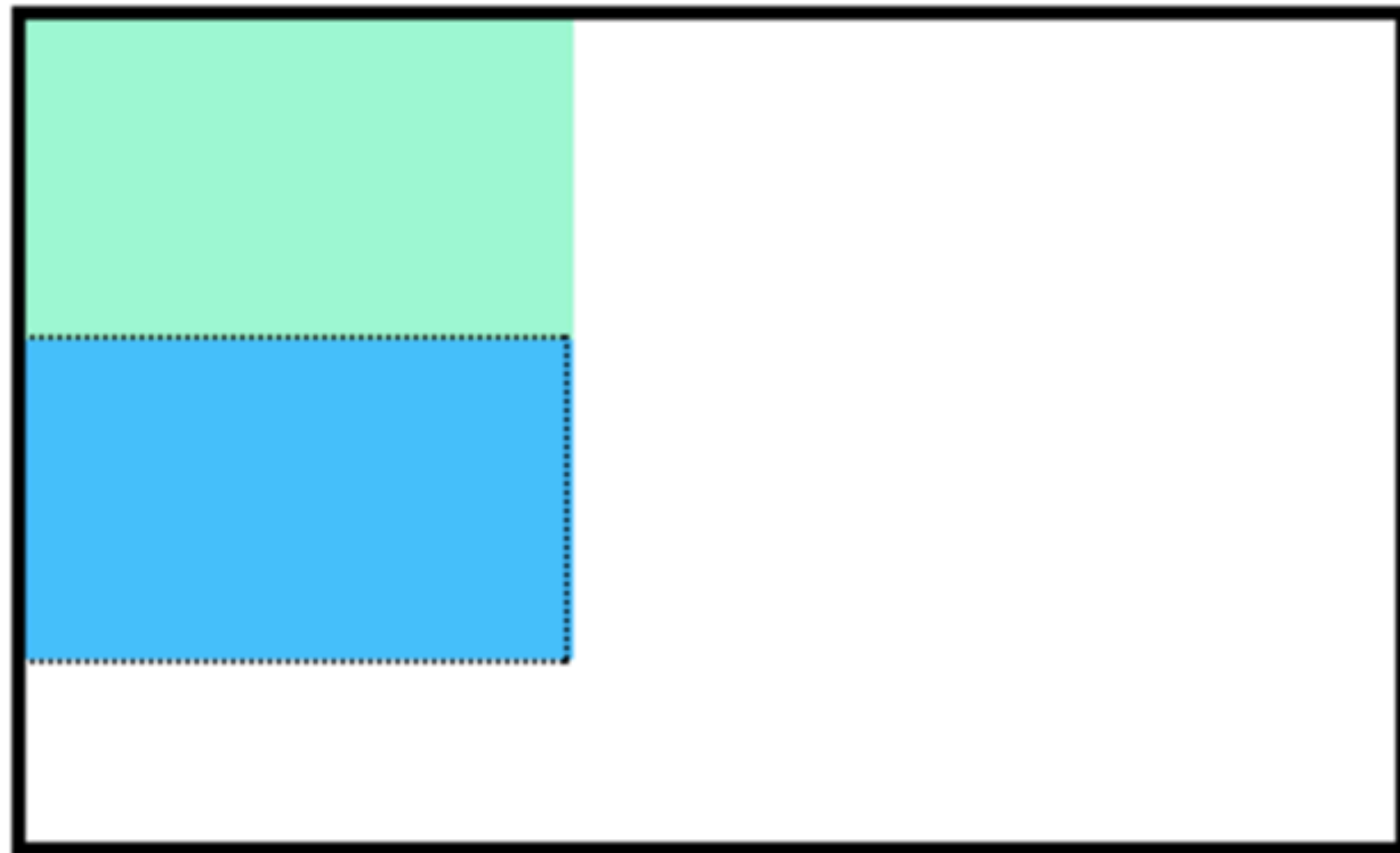


CSS Positioning

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 take up their own line of space and therefore don't overlap each other.

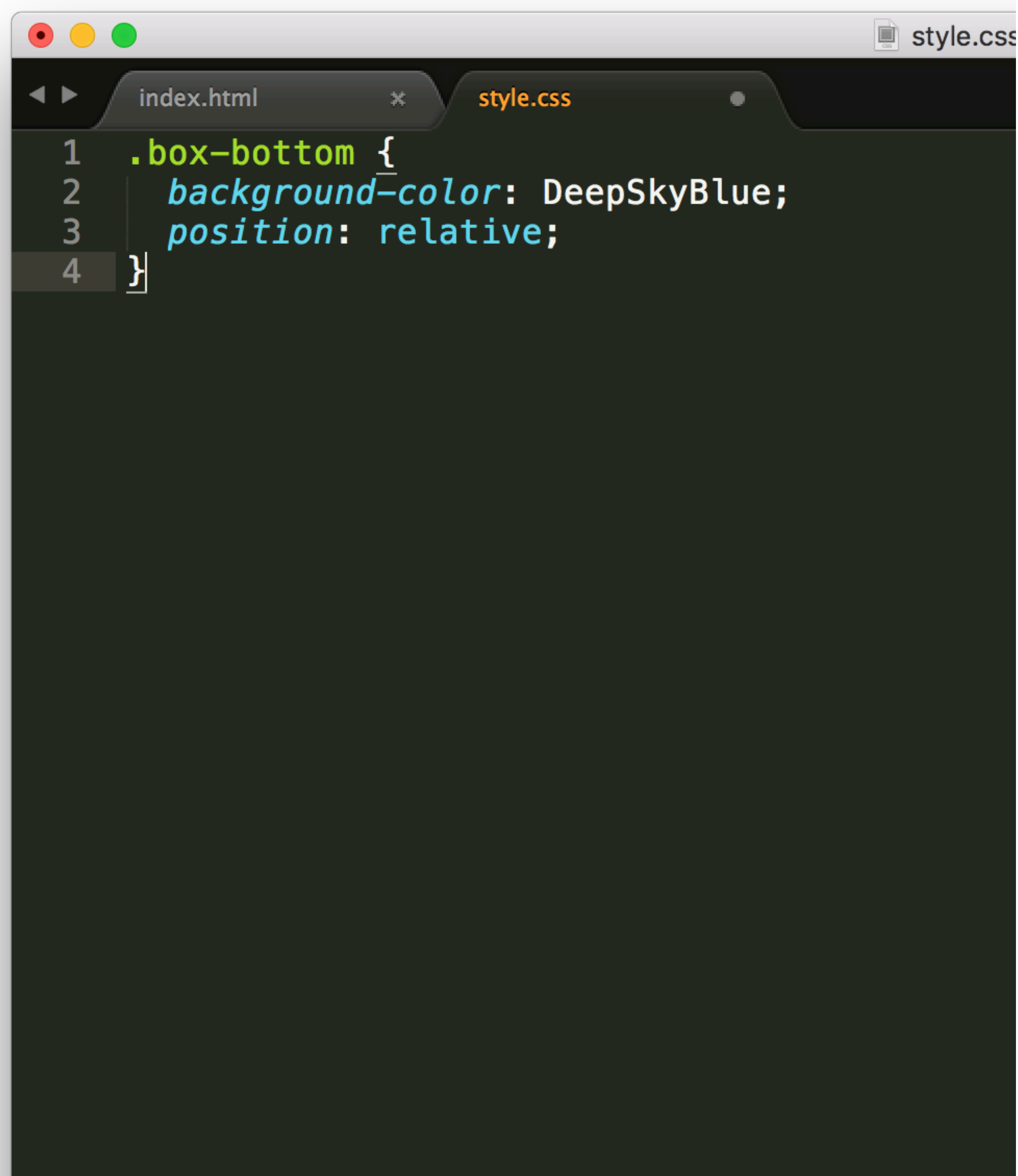
This is the default *position* for block-level elements.

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**

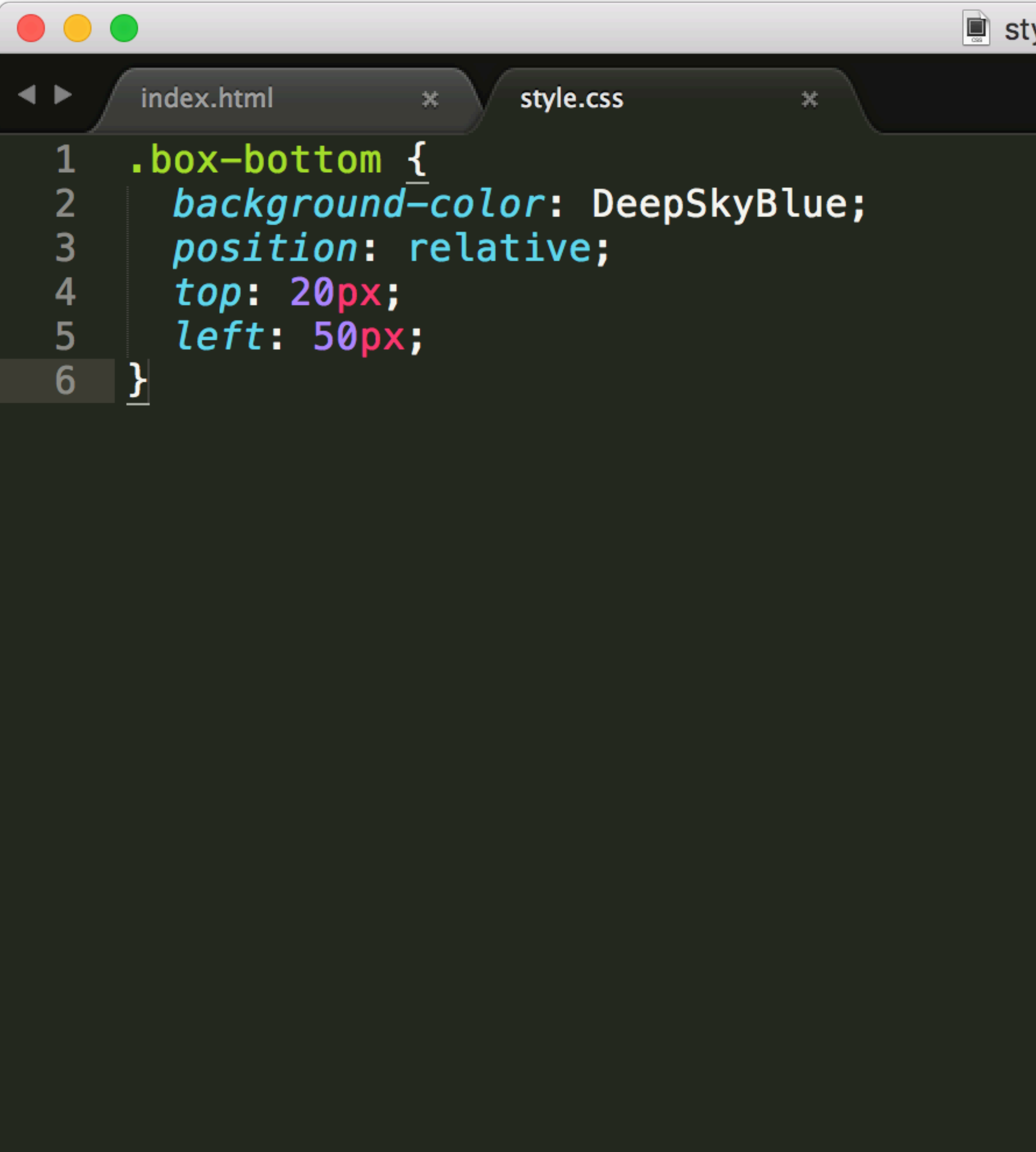
One way to modify the default position of an element is by setting its **position** property to **relative**.

A screenshot of a code editor window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left and a file icon with the text 'style.css' on the right. Below the title bar, there are two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active. The code editor has a dark background with light-colored text. The code is as follows:

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

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

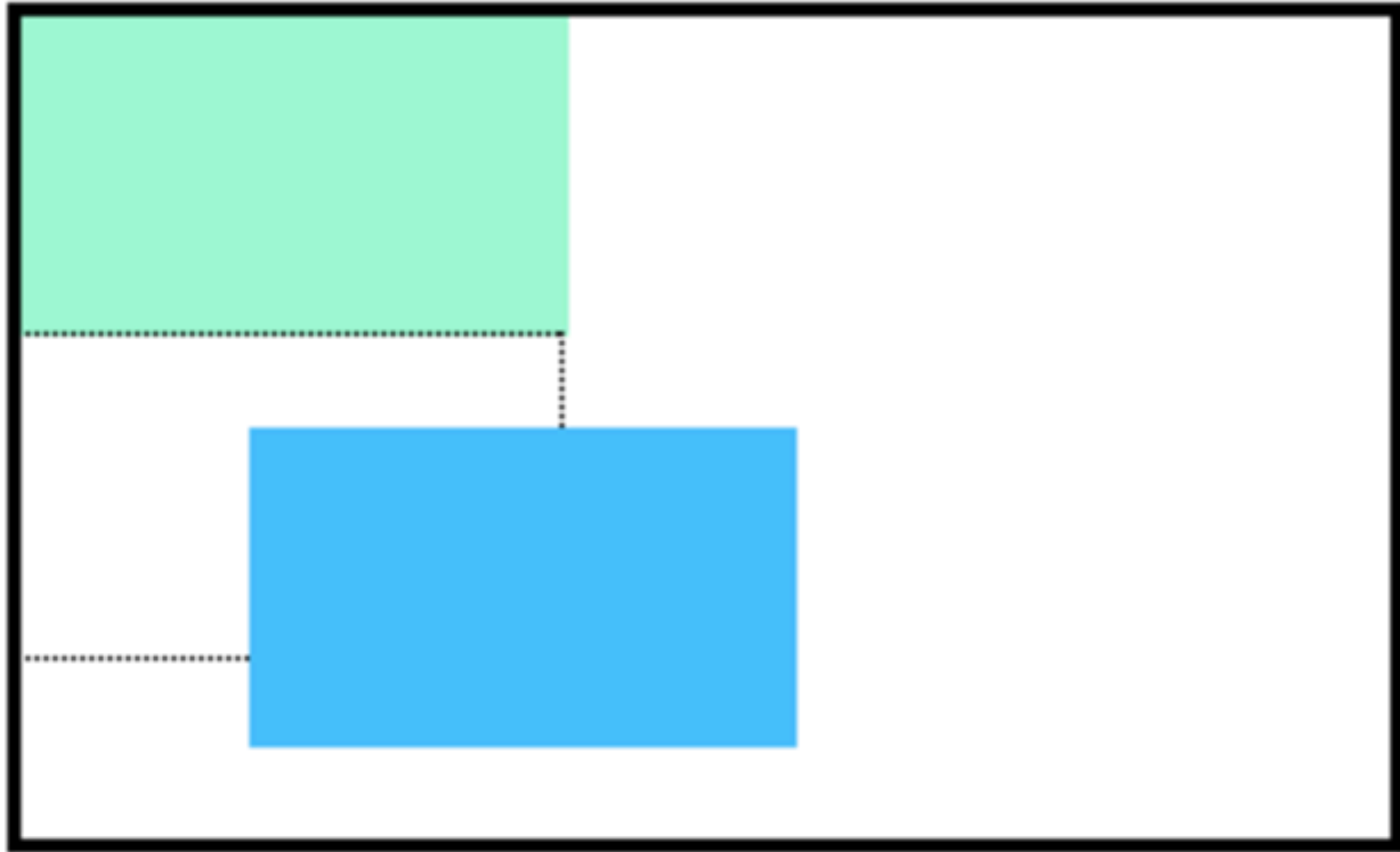
Although the code in the example instructs the browser to expect a relative positioning of the div, it does not specify where the div should be positioned on the page.

A screenshot of a code editor window. The window has a title bar with three colored buttons (red, yellow, green) on the left and a file icon with the text 'sty' on the right. Below the title bar, there are two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active. The code in the editor is as follows:

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

Now 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 two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active, showing a CSS rule for a class named '.box-bottom'. The code is as follows:

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

The code is color-coded: the class name is green, the opening curly brace is blue, the property names are blue, the values are red, and the closing curly brace is blue. The line numbers 1 through 6 are on the left side of the editor.

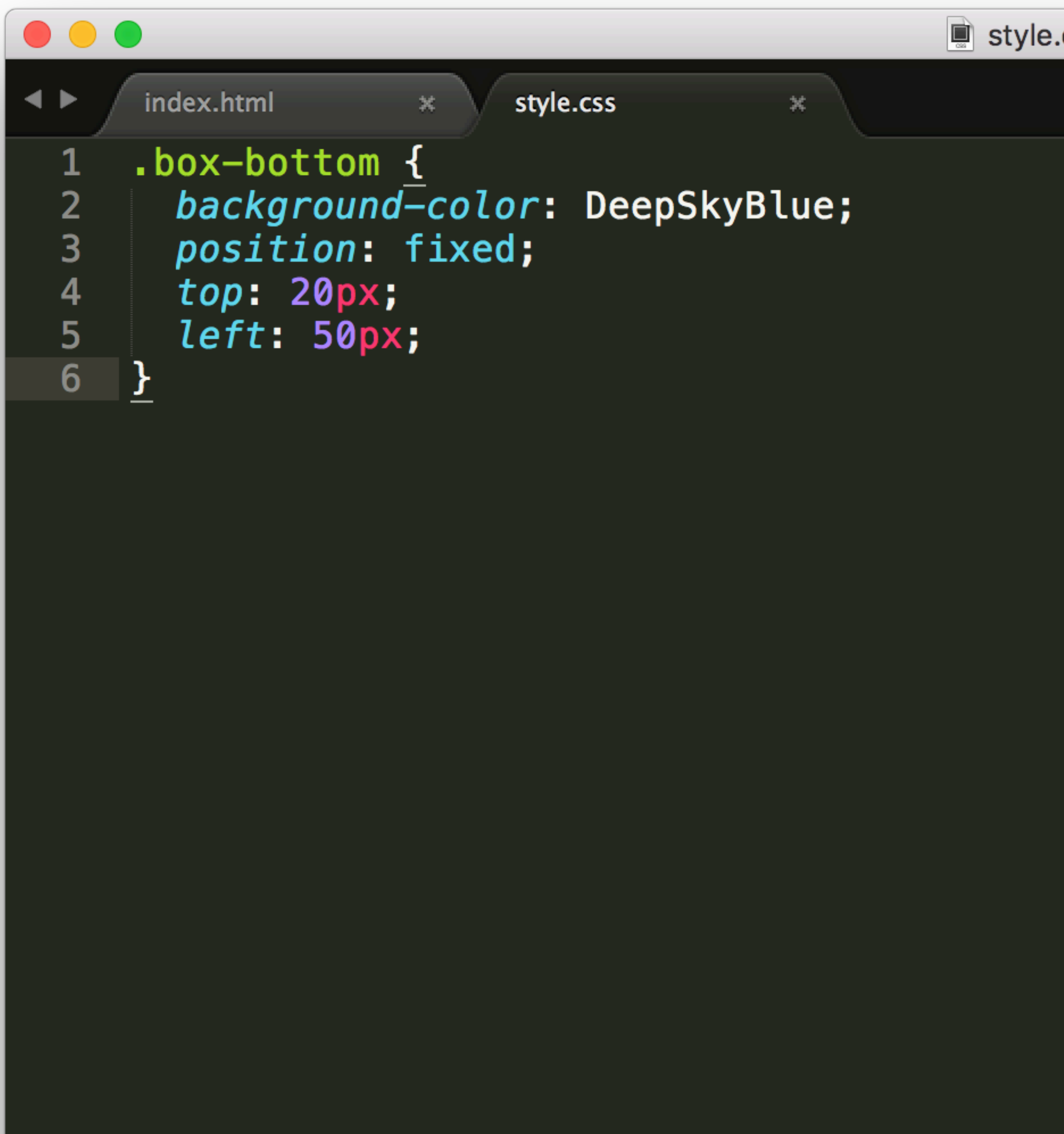
In this example, the **.box-bottom** div will be moved down and right from the top left corner of the view. If offset properties weren't specified, the top box would be entirely covered by the bottom box.



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.

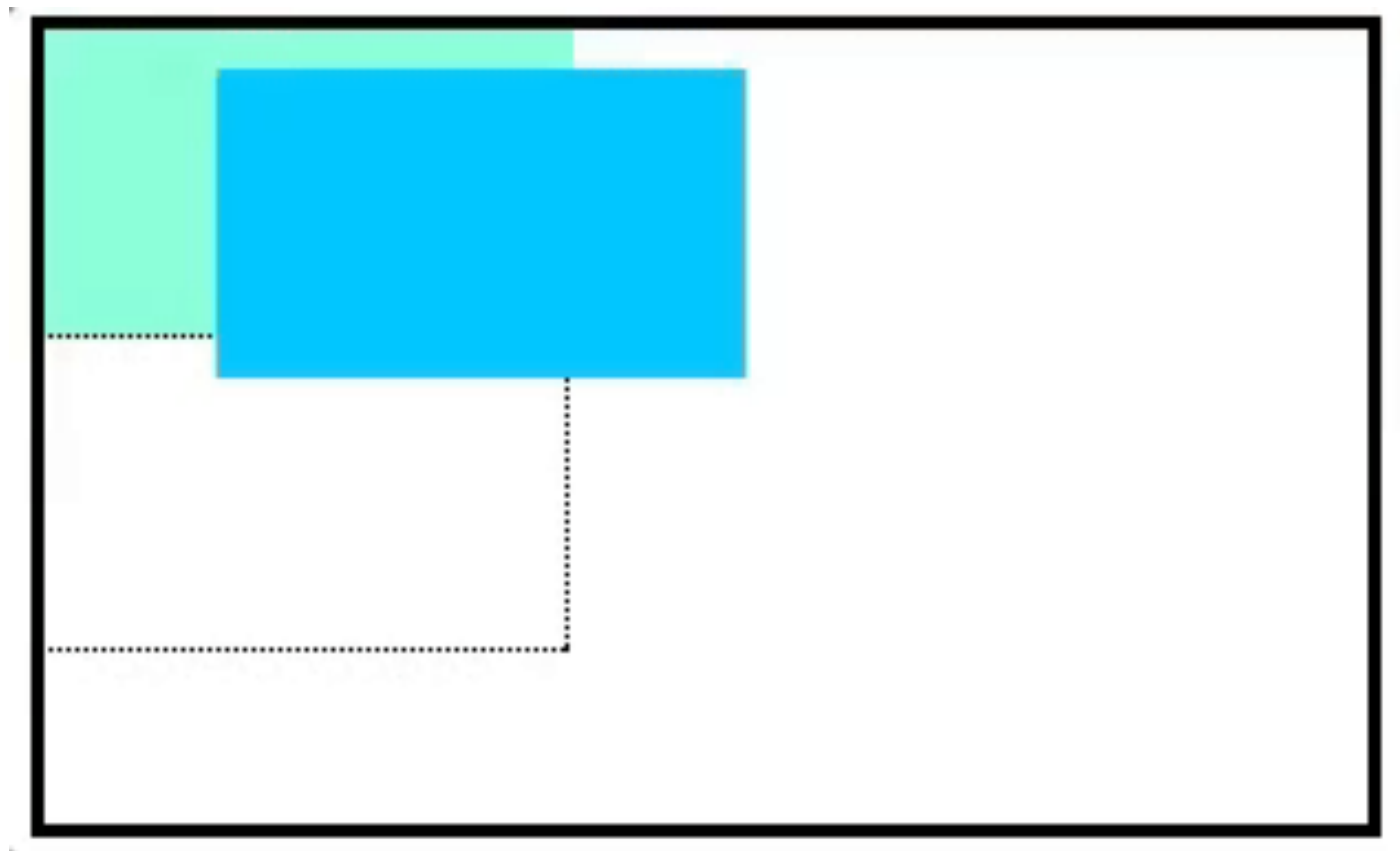
When an element's position is set to **absolute**, as in the last example, the element will scroll out of view when a user scrolls.

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. The window has a title bar with three colored buttons (red, yellow, green) on the left and a file icon with the text 'style.' on the right. Below the title bar, there are two tabs: 'index.html' and 'style.css'. The 'style.css' tab is active. The code editor shows the following CSS code:

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

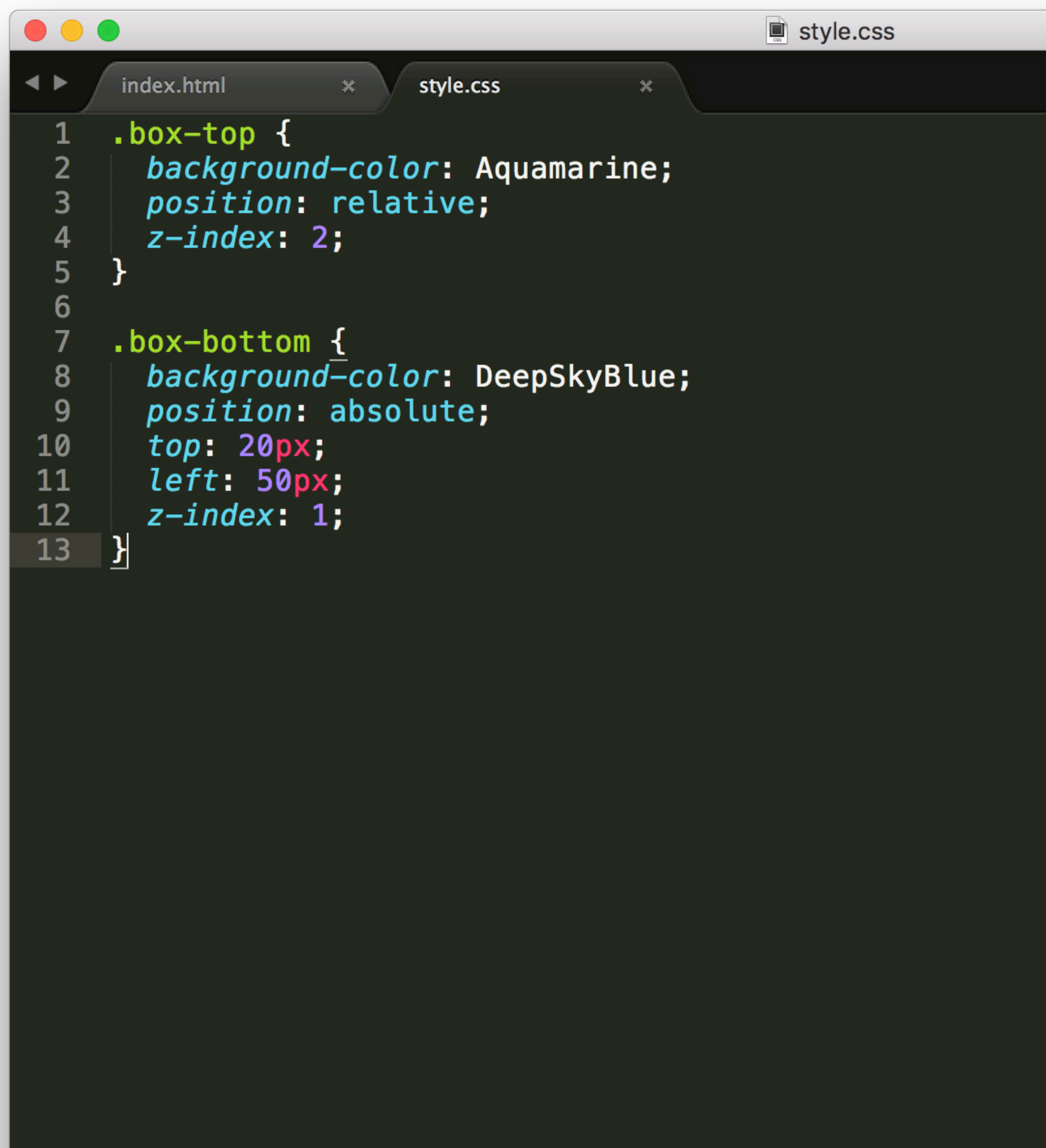
The div will remain fixed to its position no matter where the user scrolls on the page.



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.

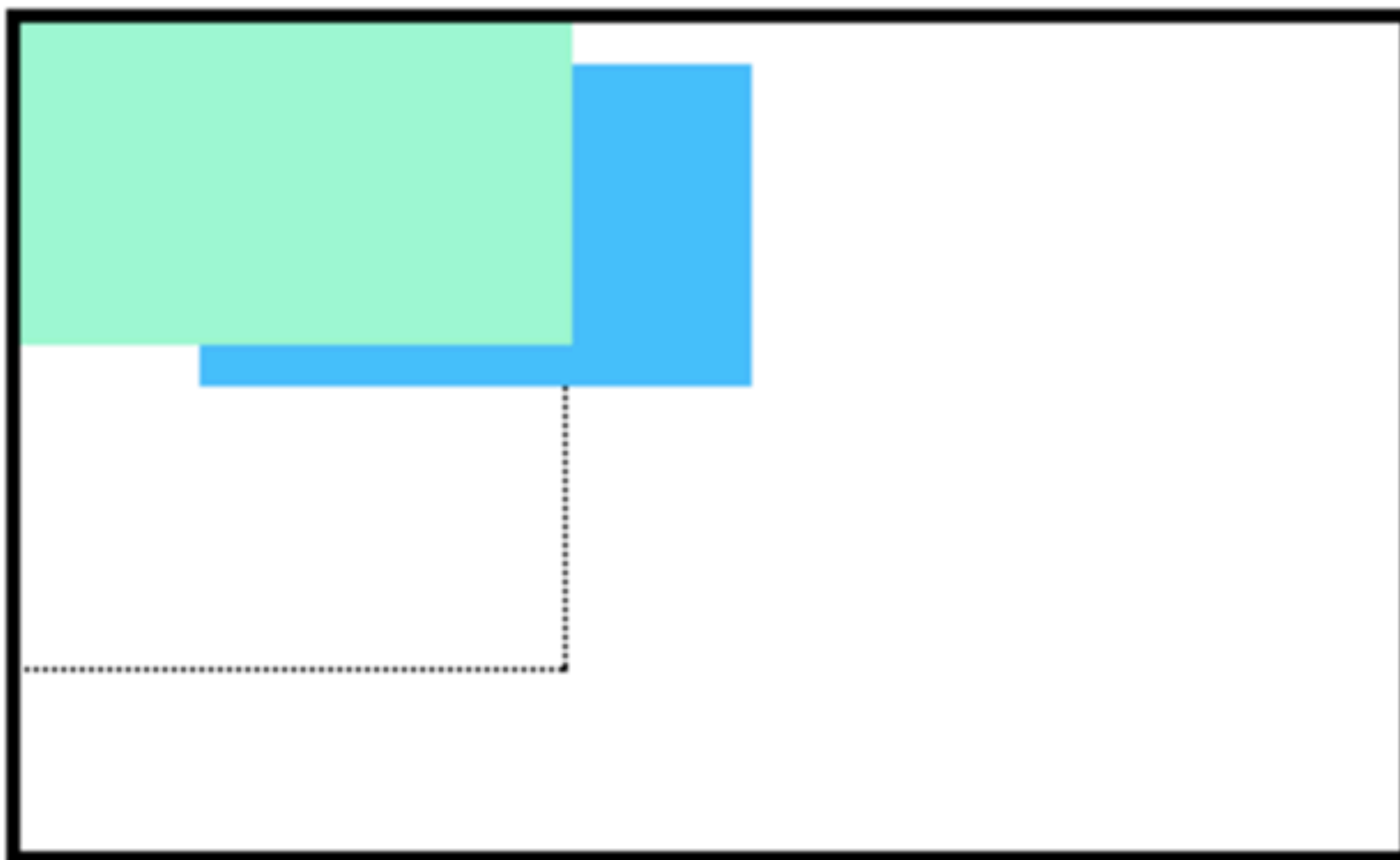
The **z-index** property controls how far "back" or how far "forward" an element should appear on the web page.

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.



```
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 }
```

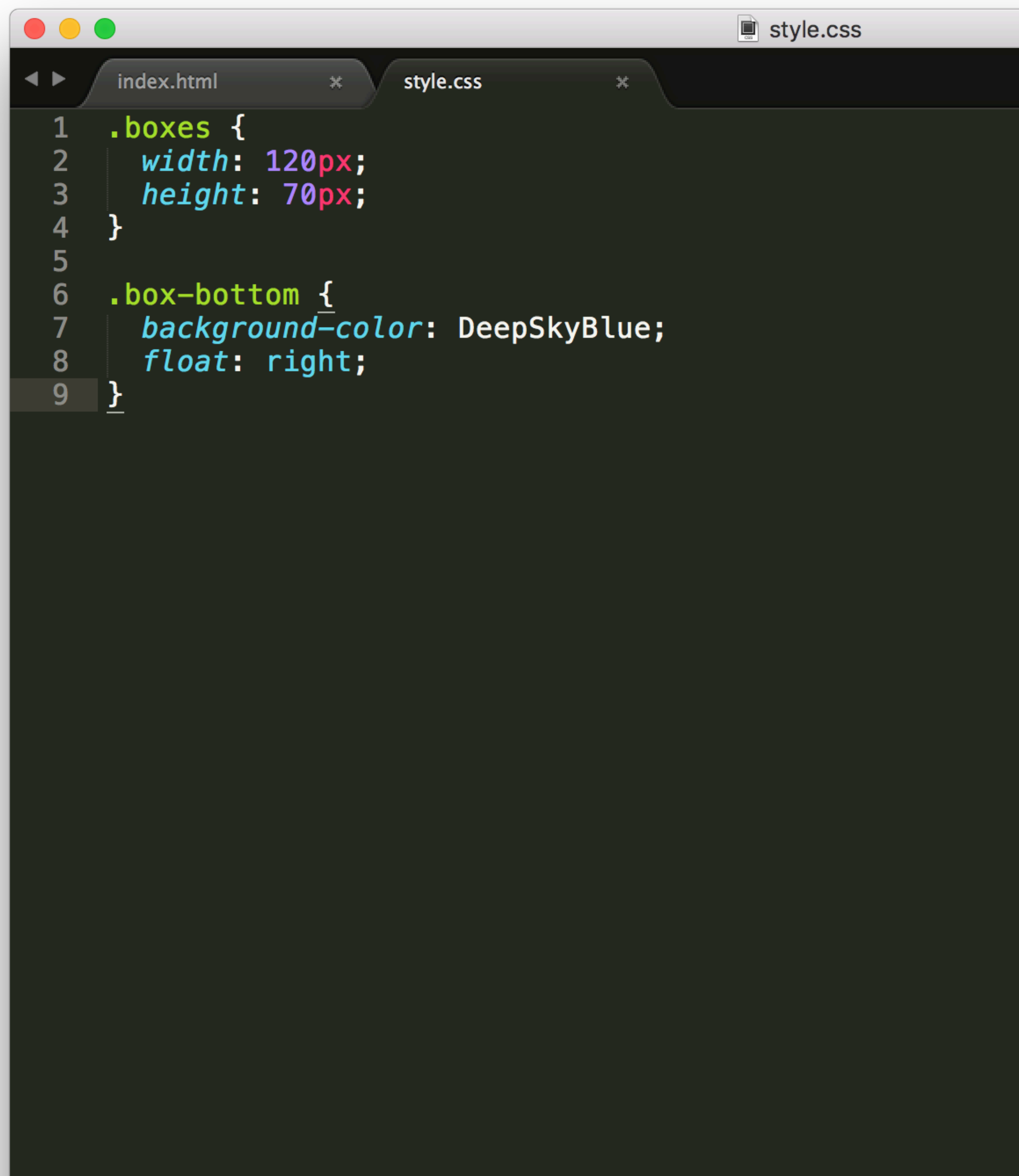
We set the **.box-top** position to relative and the z-index to 2. We changed position to **relative**, because the z-index property does *not* work on static elements. The z-index of **2** moves the **.box-top** element forward, because it is greater than the **.box-bottom** z-index, **1**



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:

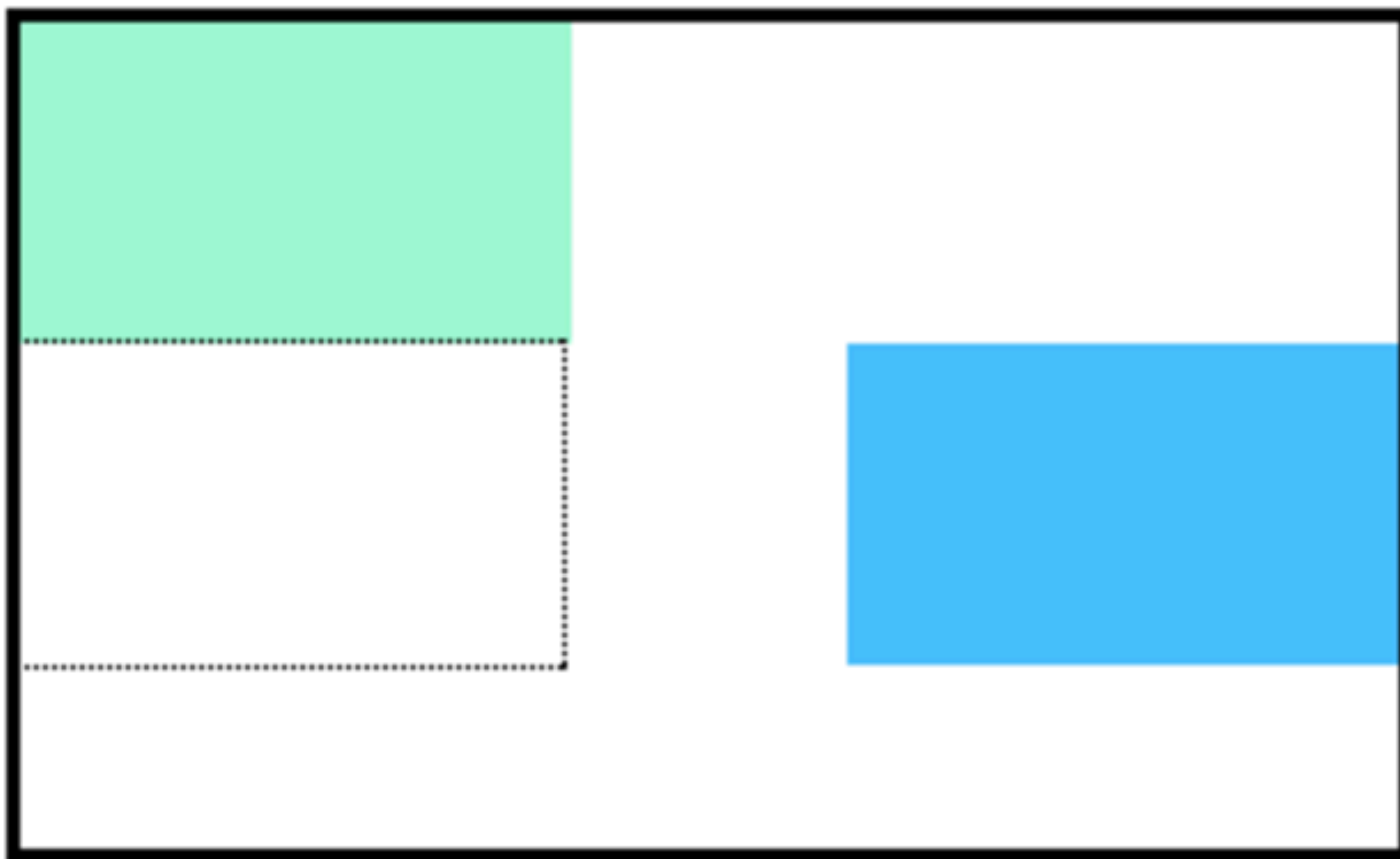
1. **left** – this value will move, or float, elements as far left as possible.
2. **right** – this value will move elements as far right as possible.



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

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

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



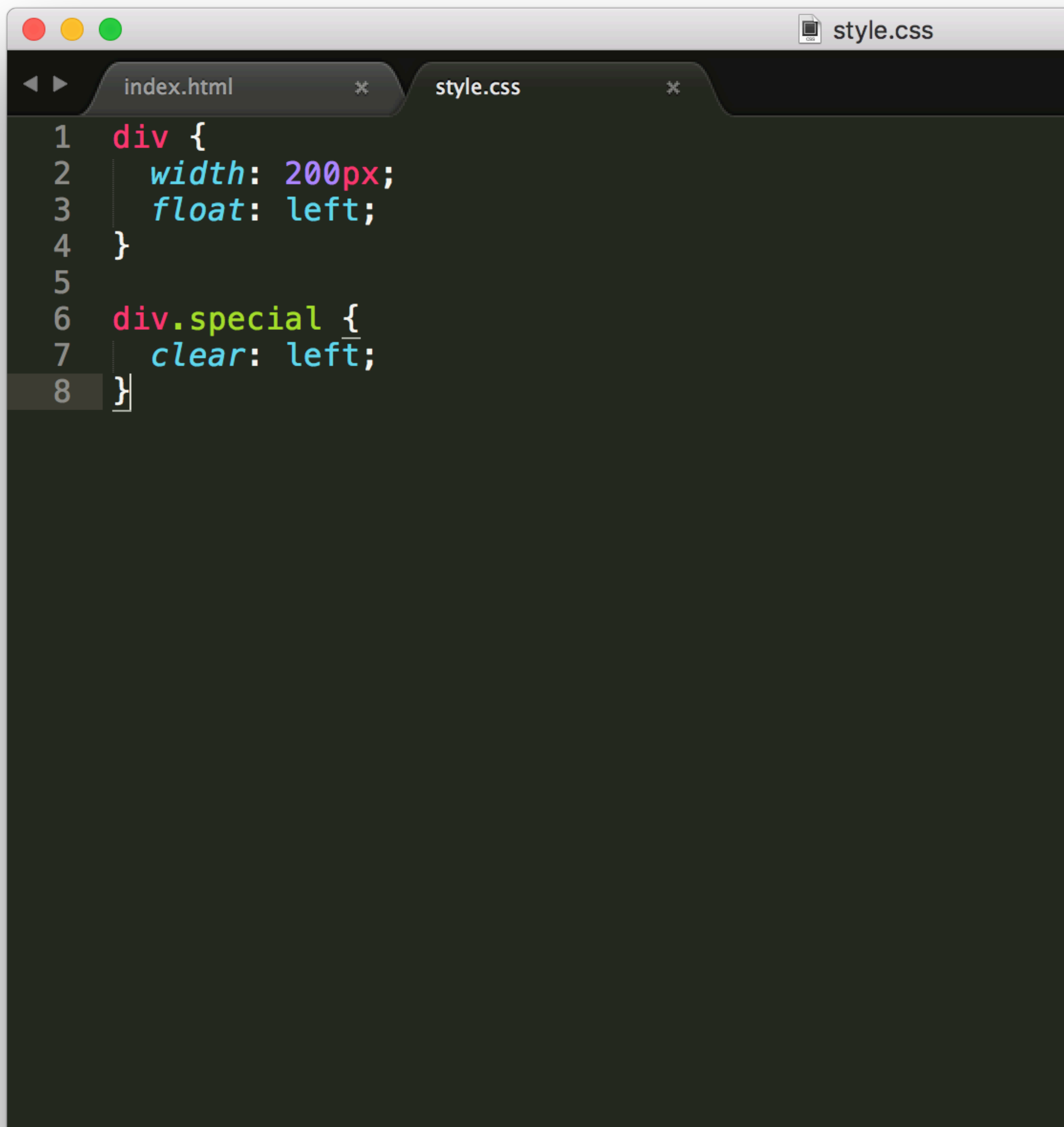
Floated elements must have a width specified, as in the example above. Otherwise, the element will assume the full width of its containing element, and changing the float value will not yield any visible results.

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.



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

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.