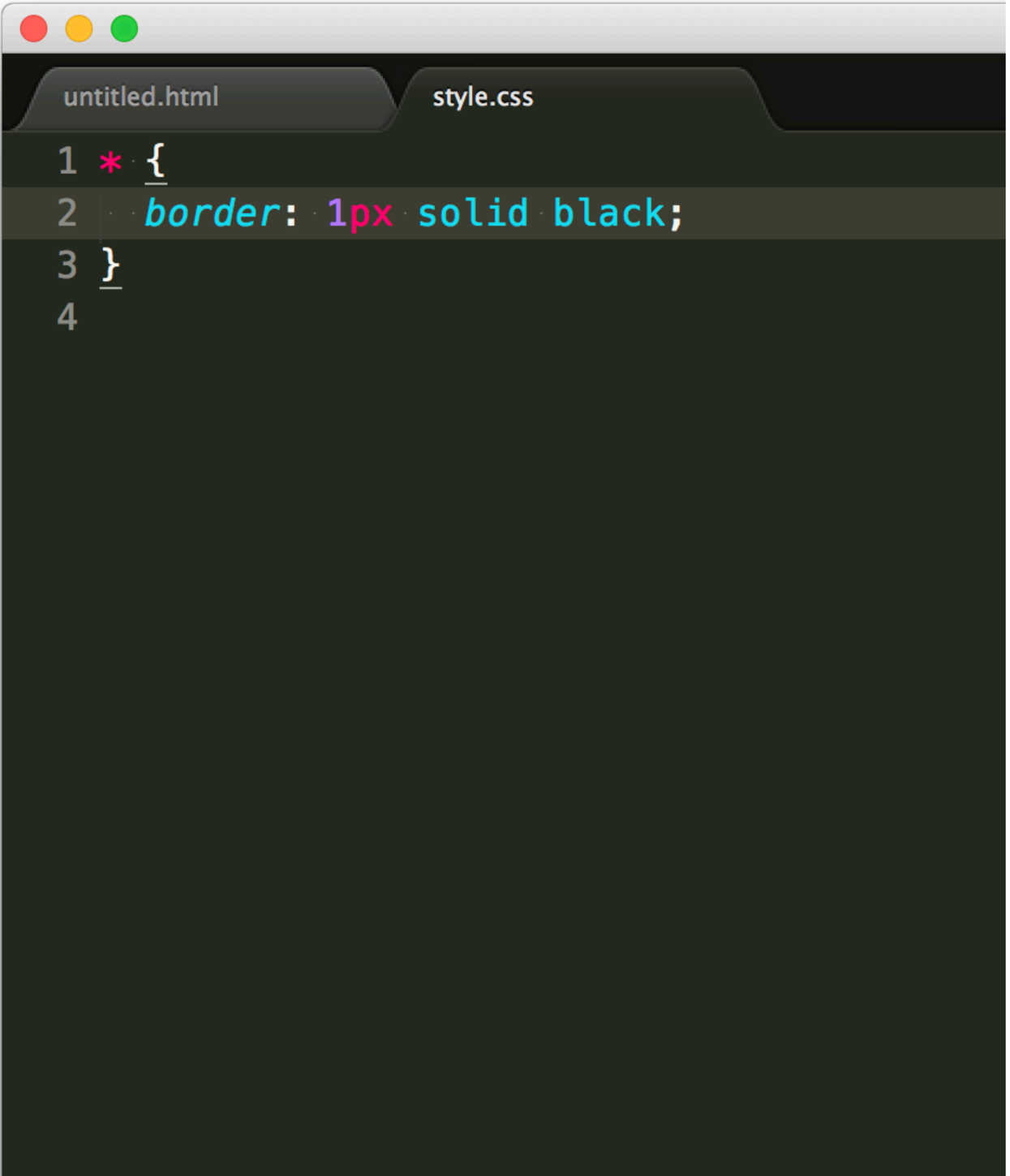
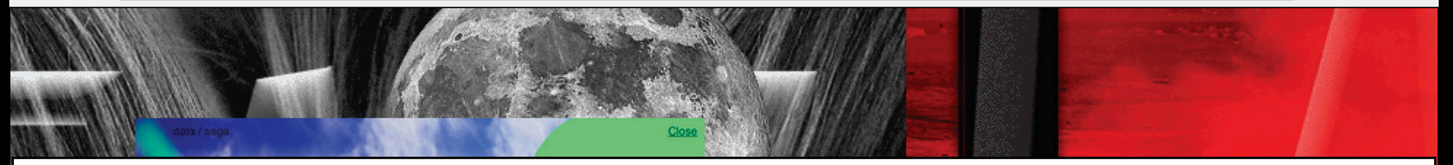
The Box Model (Layout positioning)

All HTML elements live within a box. Elements on a web page are understood by the browser as "living" inside of a container, or a box. This is what is meant by the *box model*.

Where are these boxes that supposedly contain all HTML elements? They're invisible but we can reveal them.



This code selects *all* elements on the page (using the universal selector you learned about earlier) and reveals the borders of their box.



Intro to Interactive | Schedule | Assignments | Resources

Week 1

Studio — Mon 7/3

Authorship & Design

Lab — Mon 7/3

Git, Command Line, HTML

Studio — Wed 7/5

Starting Points

Lab — Wed 7/5

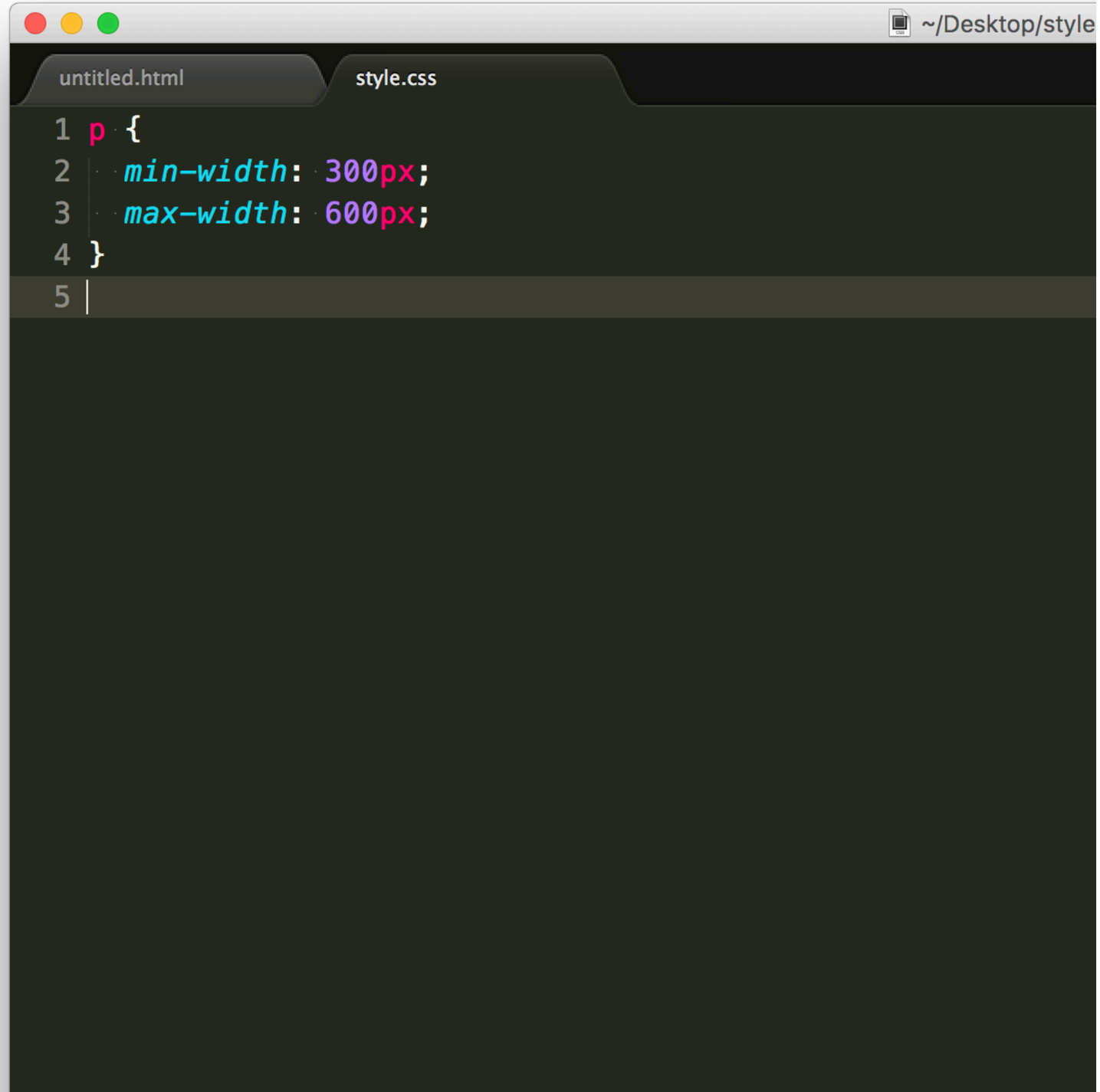
CSS, Positioning, Box Model

An element's box has two dimensions: a *height* and a *width*. In HTML, all boxes have default dimensions. These default dimensions are automatically set to the size of their parent container

To modify the default dimensions an element's box in CSS, you can use the width and height properties. These can be set in pixels, percentages or ems.

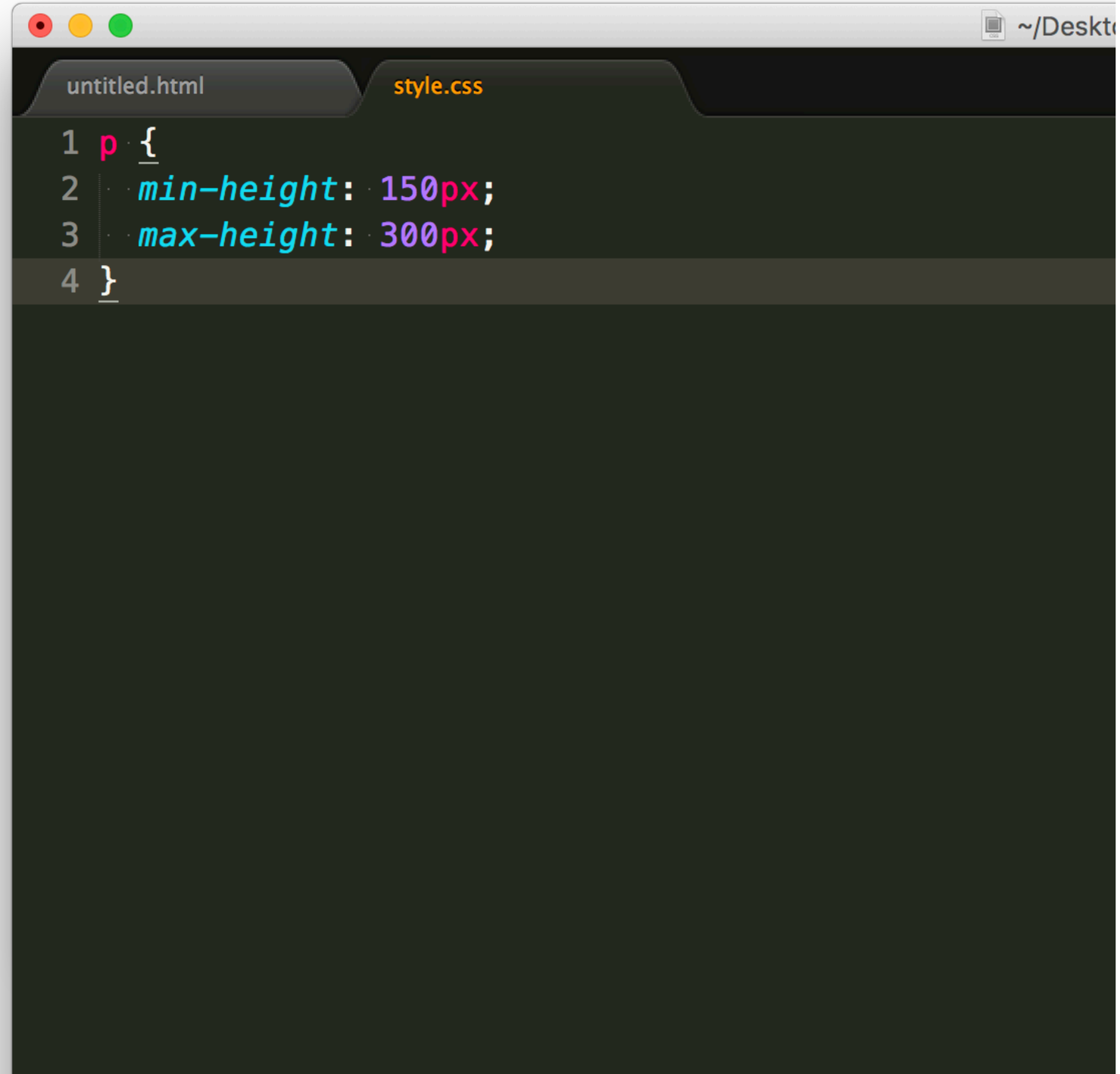
Try creating two divs with distinct class names. Give the first a width of 200px and a height of 200px. Give the second a width of 100% and a height of 500px. Give each a unique background color.

Just as you can make a div larger than it’s contents,
you can also set limts to it’s height and width.



min-width - this property ensures a minimum width for an element's box.

max-width - this property ensures a maximum width for an element's box.



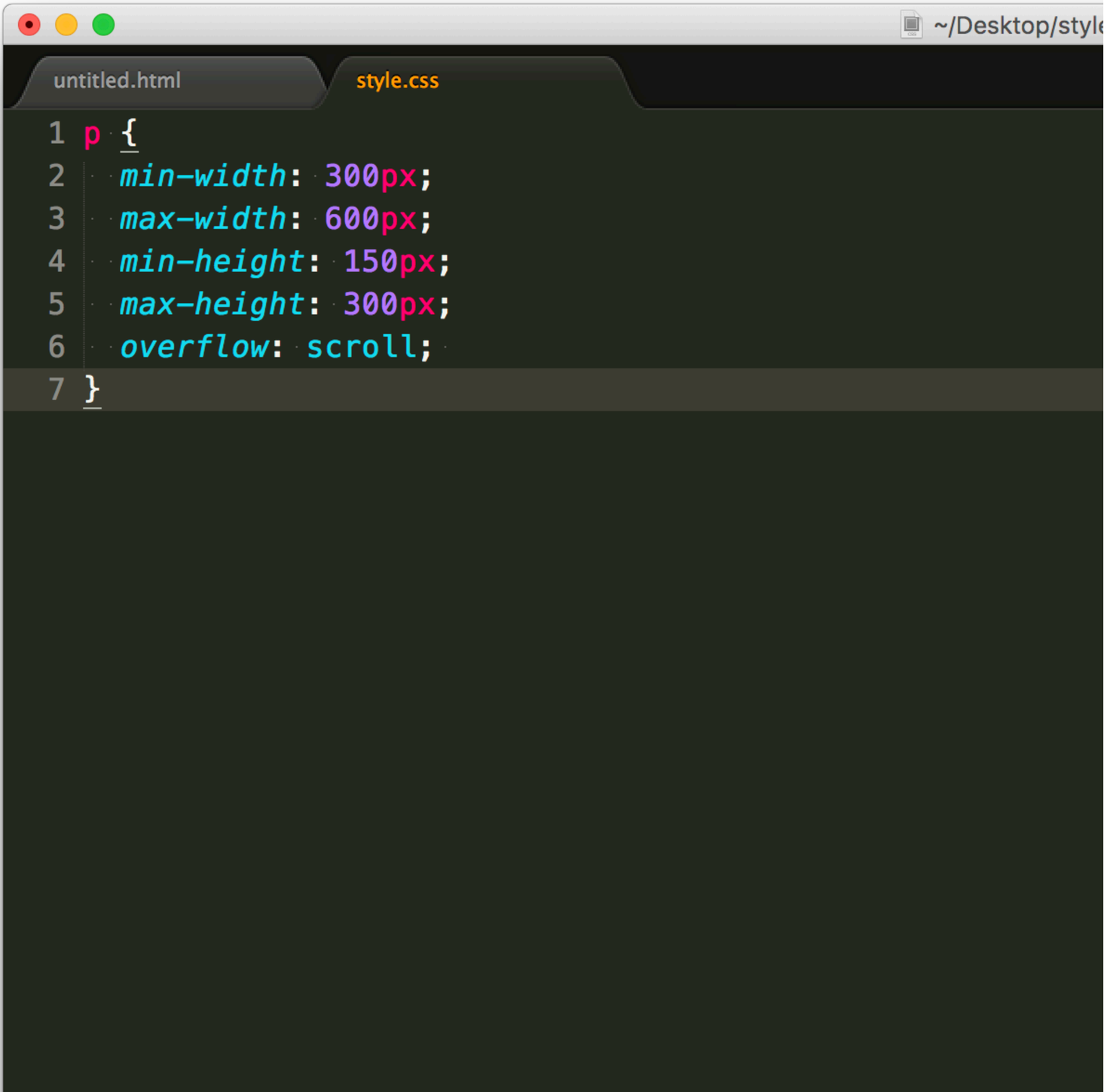
You can also limit the minimum and maximum *height* of an element.

`min-height` - this property ensures a minimum height for an element's box.

`max-height` - this property ensures a maximum height for an element's box.

What will happen to the contents of an element's box if the max-height property is set too low? It's possible for the content to spill outside of the box, resulting in content that is not legible

The overflow property controls what happens to content when it spills, or *overflows*, outside of its box.

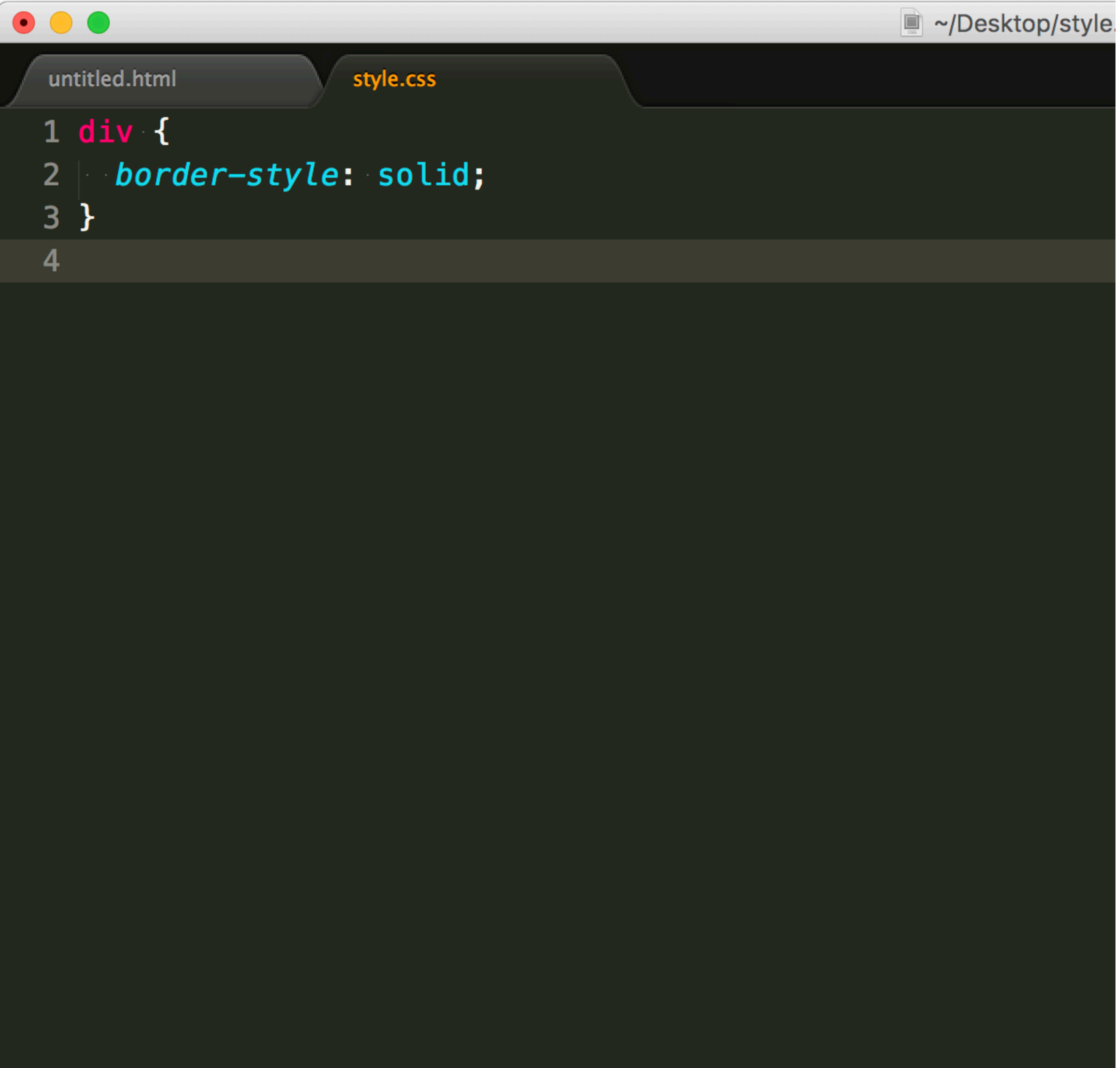


The **overflow** property controls what happens to content when it spills, or *overflows*, outside of its box.

hidden - when set to this value, any content that overflows be hidden from view.

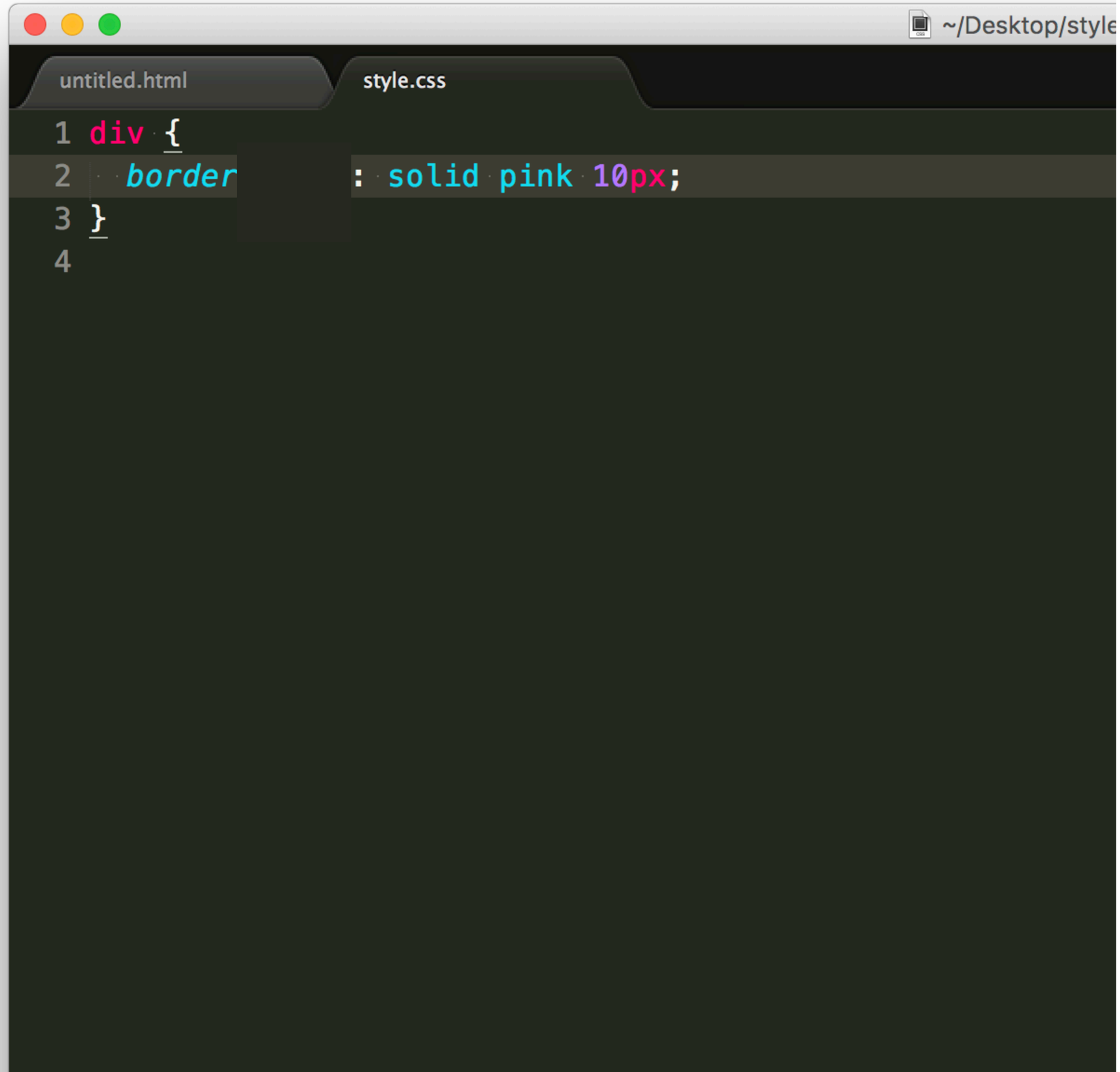
scroll - when set to this value, a scrollbar will be added to the element's box so that the rest of the content can be viewed by scrolling.

It's not possible to view a box's border if the border's *style* has not been set. A border's style can be set with the border-style property.

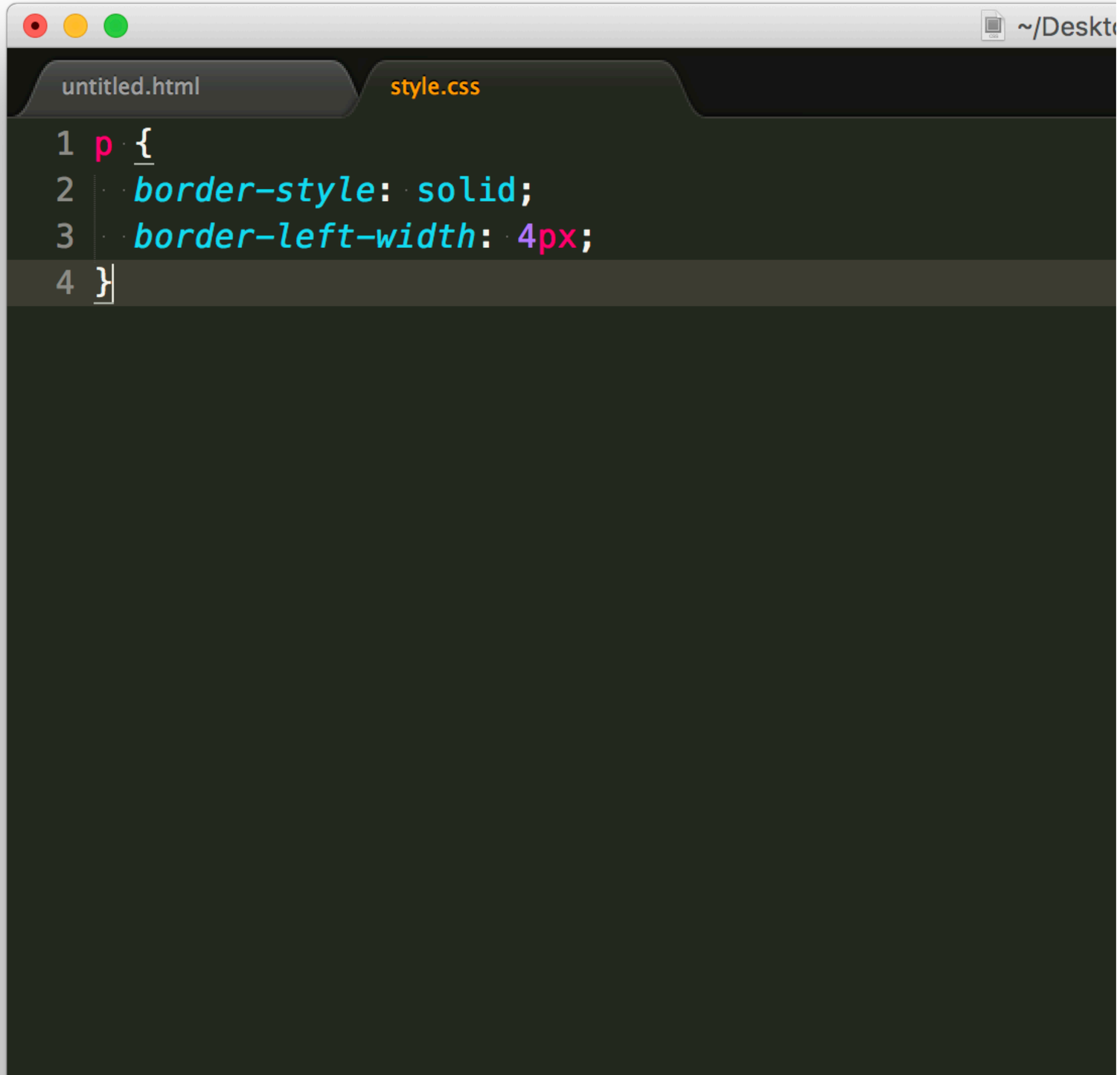


This property can take on one of the following values:

solid, dashed, dotted, double, groove, inset, outset, ridge, hidden or none.



Often times you will also want to set the borders color and width. You can do all this in a single line of code.



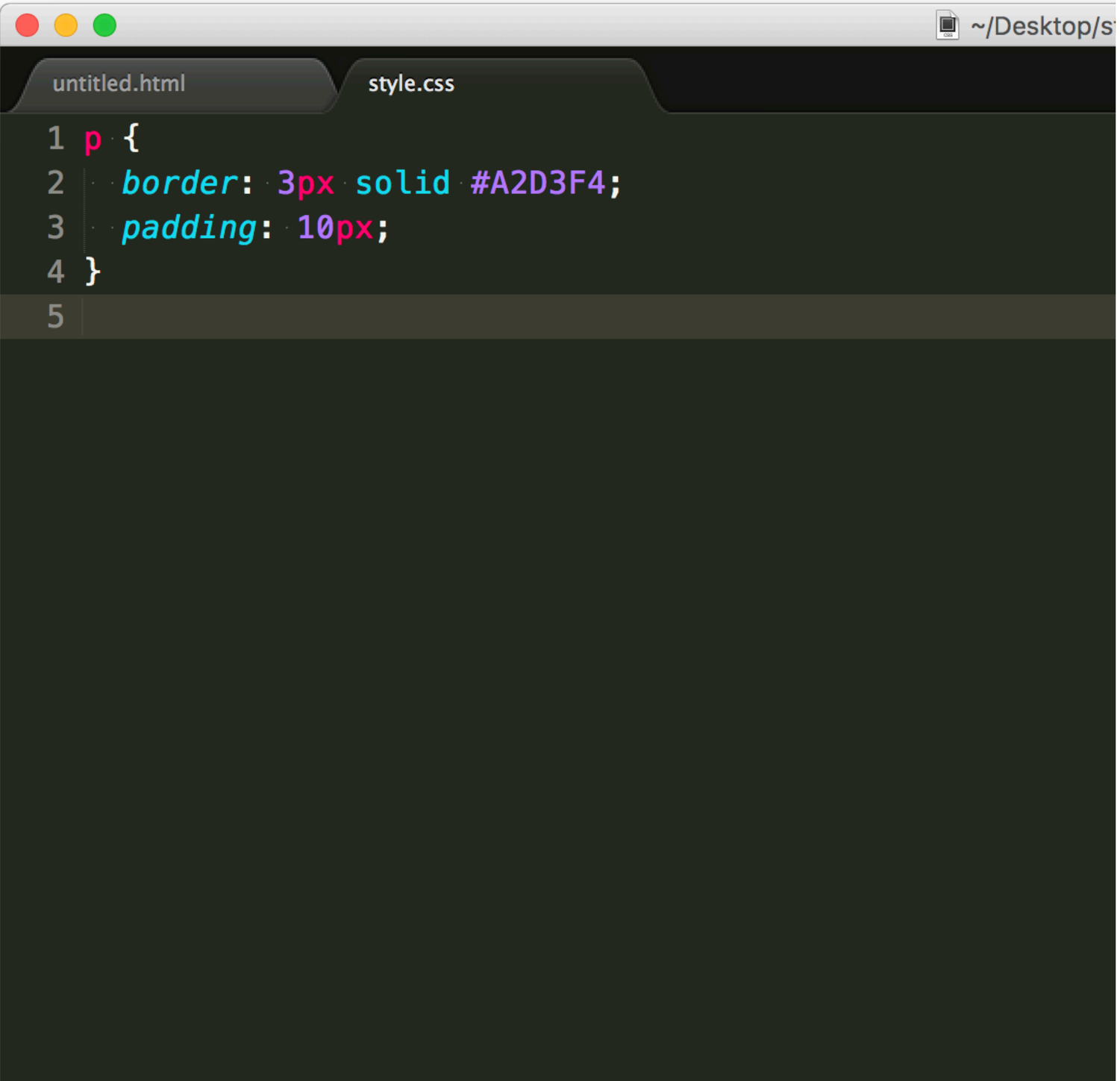
If you'd like to be even more specific about the width of different sides of the border, you can use the following properties:

`border-top-width`,
`border-right-width`,
`border-bottom-width`
`border-left-width`

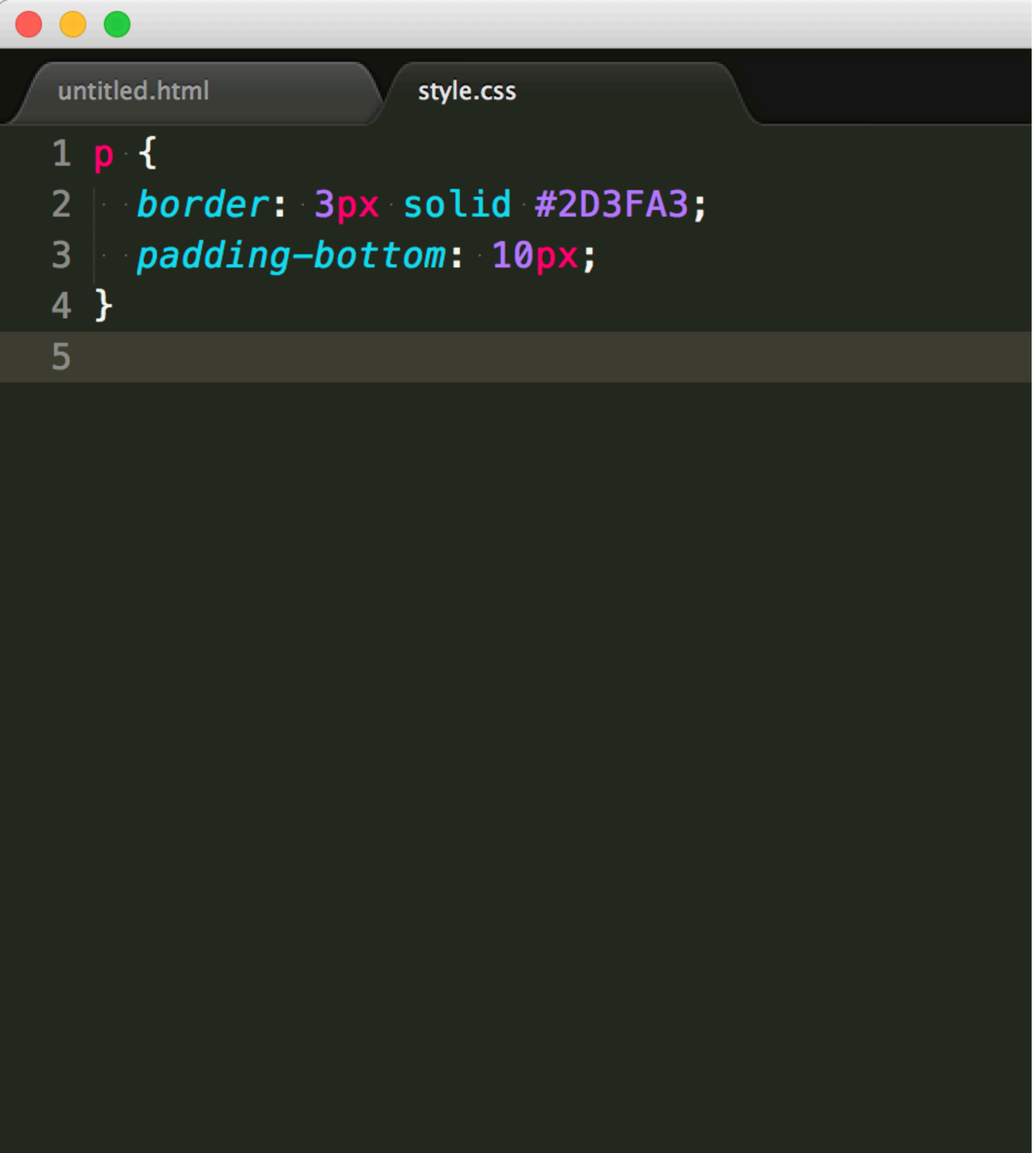
Each property affects the width of only one side of the border, giving you more flexibility in customization.

Box dimensions and borders are just the beginning of the vast number of box properties you can modify in CSS. Lets take a look at how to modify the spacing around the content *inside* of the box.

The space between the contents of a box and the borders of a box is known as *padding*.



The code in the example will put 10 pixels of space between the content of the paragraph (the text) and the box borders, on all four sides.

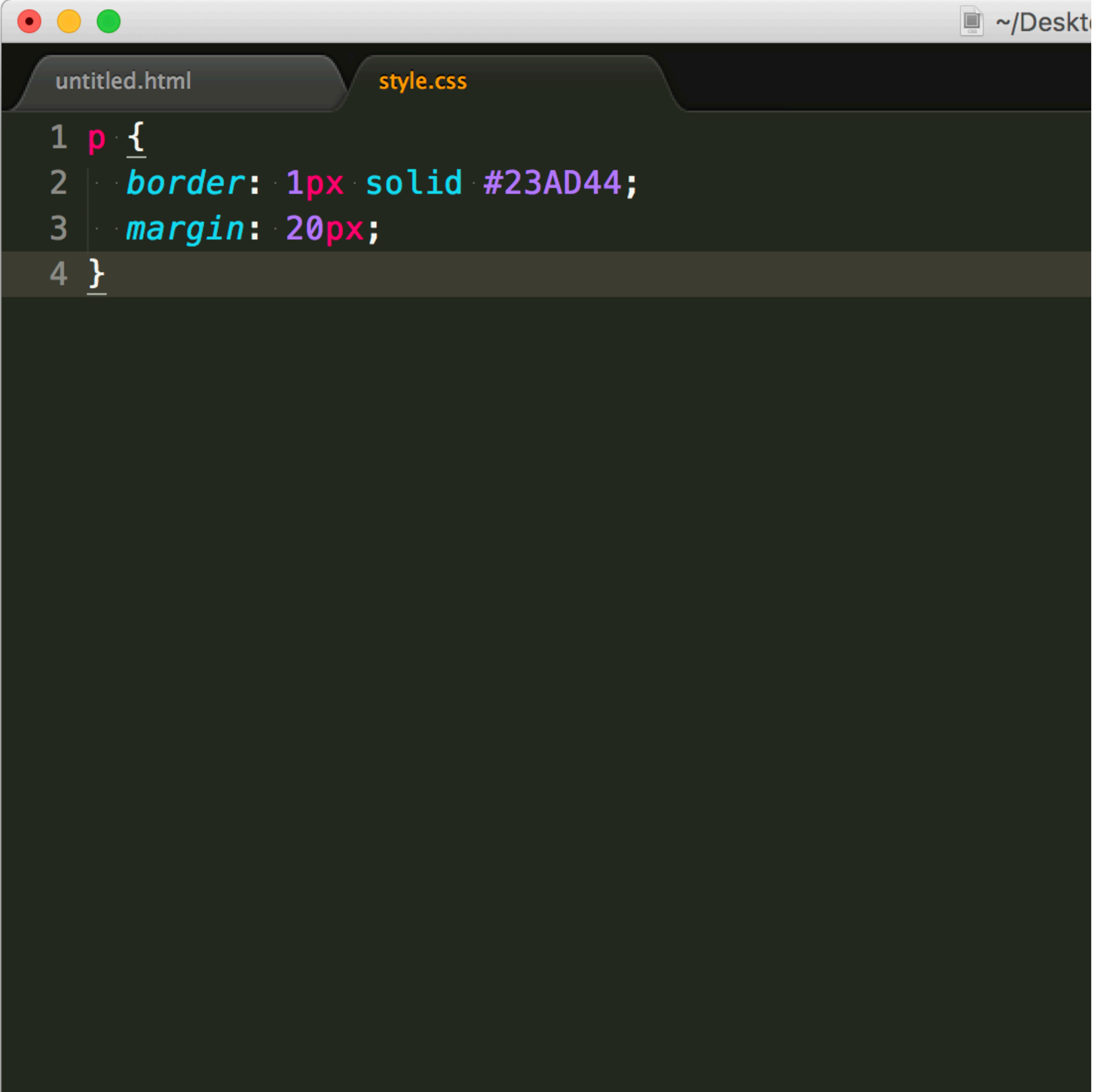


If you want to be even more specific about the amount of padding on each side of a box's content, you can use the following properties:

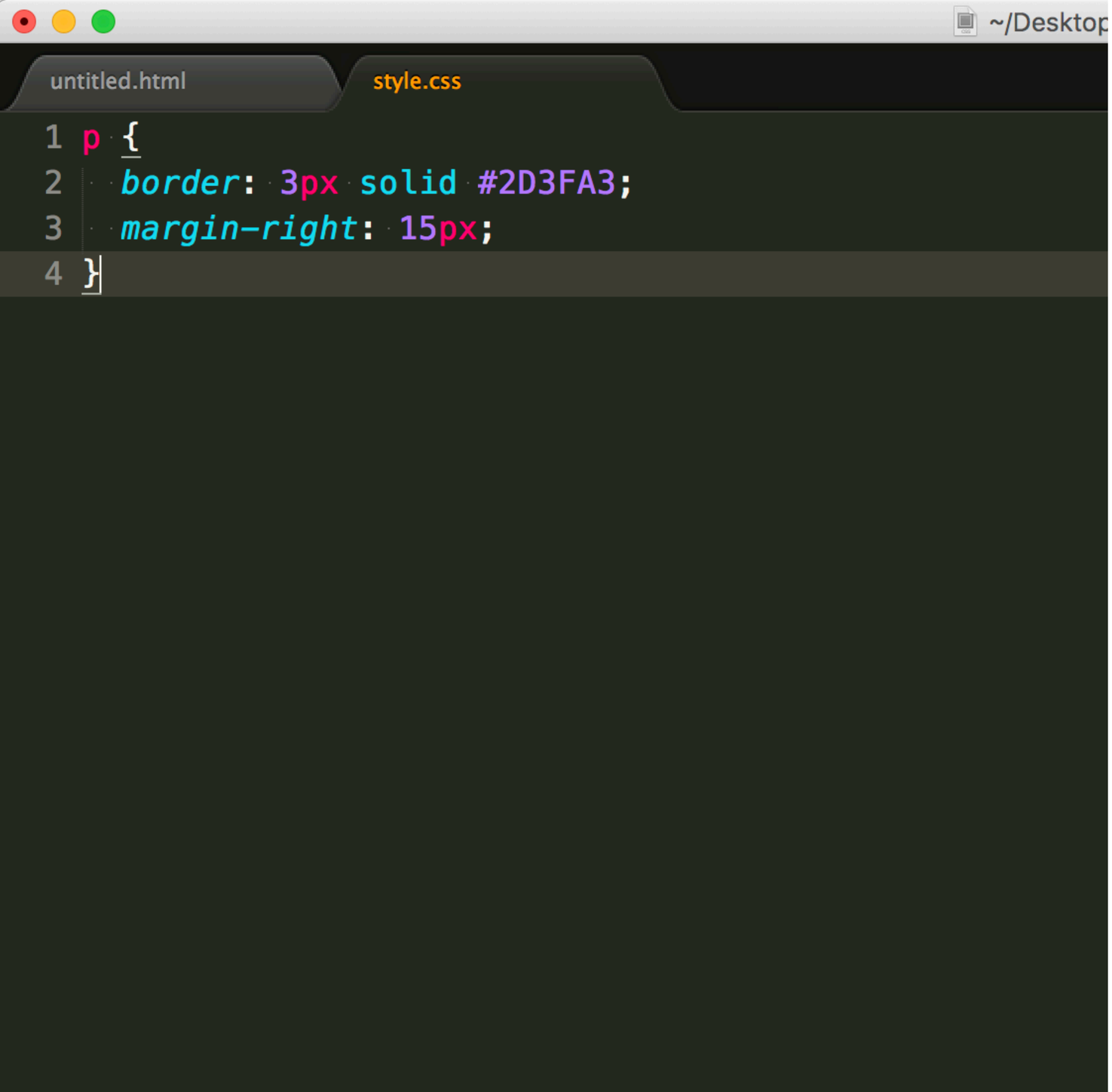
padding-top
padding-right
padding-bottom
padding-left

So far, we've learned about the following aspects of the box model: dimensions, borders, and padding. The fourth and final aspect of the box model is *margin*.

The margin refers to the space directly outside of the box.



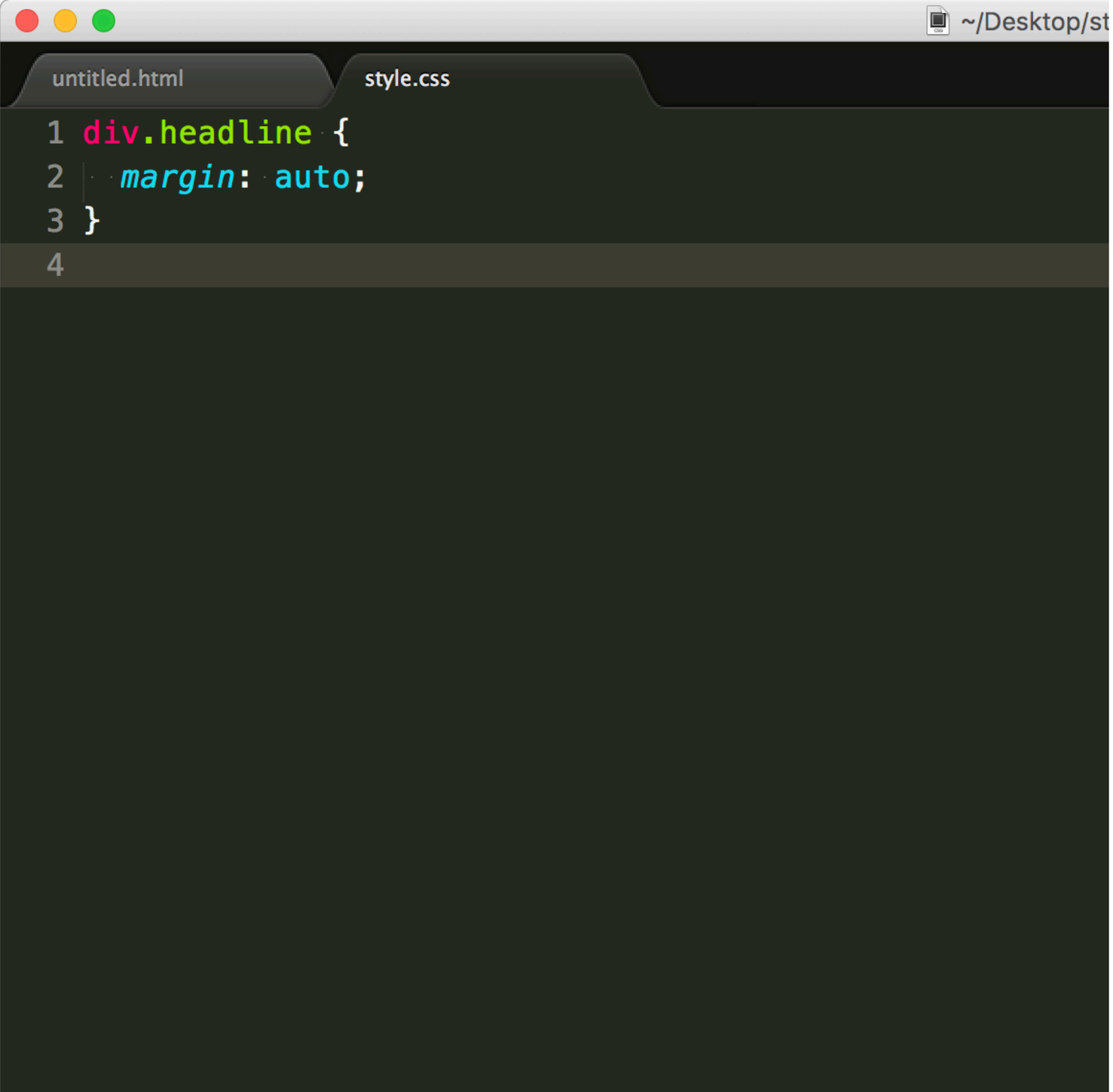
The code in the example above will place 20 pixels of space on the outside of the paragraph's box, on all four sides. This means that other HTML elements on the page cannot come within 20 pixels of the paragraph



If you want to be even more specific about the amount of margin on each side of a box's content, you can use the following properties:

margin-top
margin-right
margin-bottom
margin-left

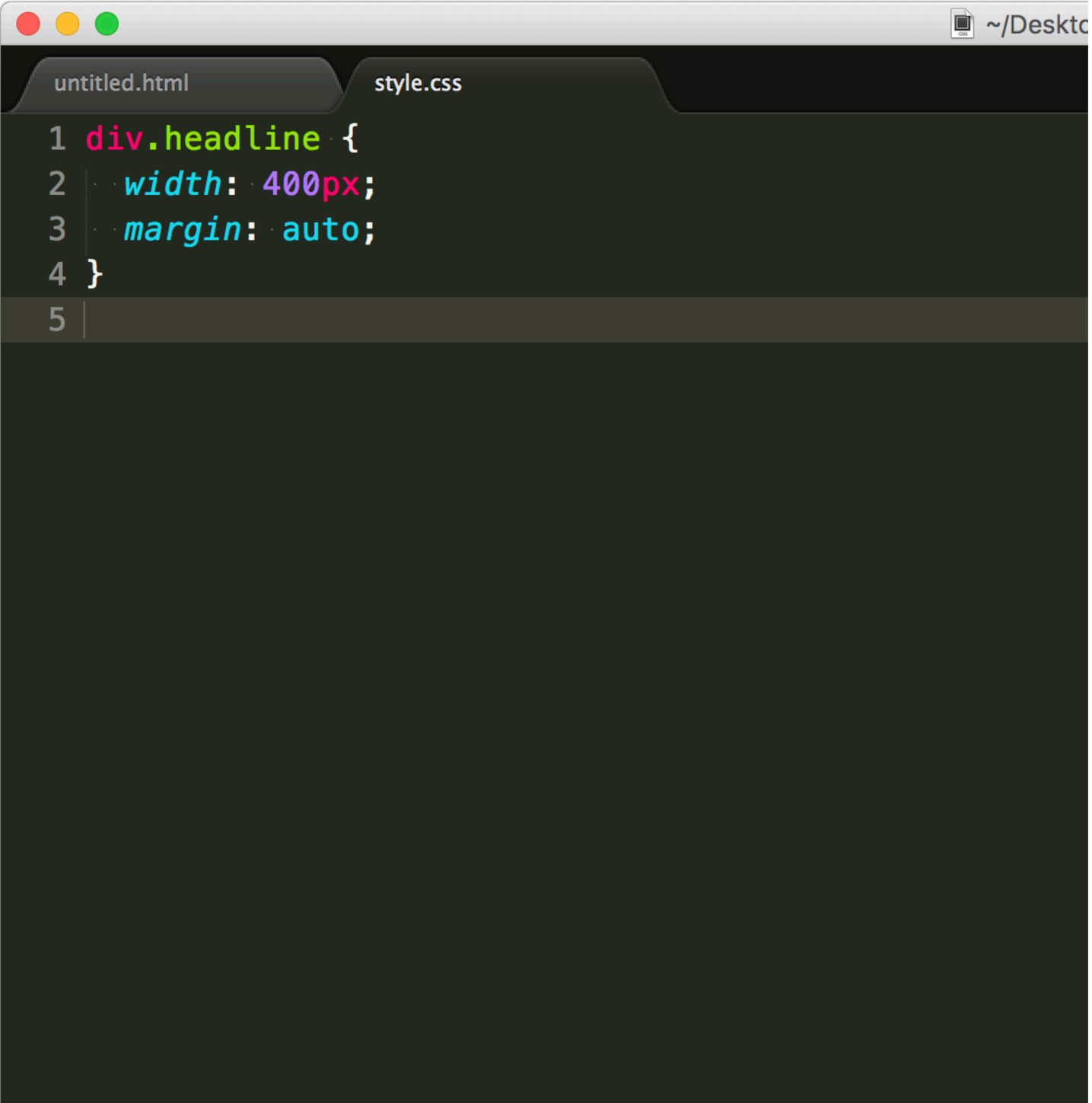
The margin property also lets you center content, if you follow certain requirements.



When the margin property is set to auto, the element being styled will center in the page.

In theory, the div in the example above should center on the page, but it doesn't. Why?

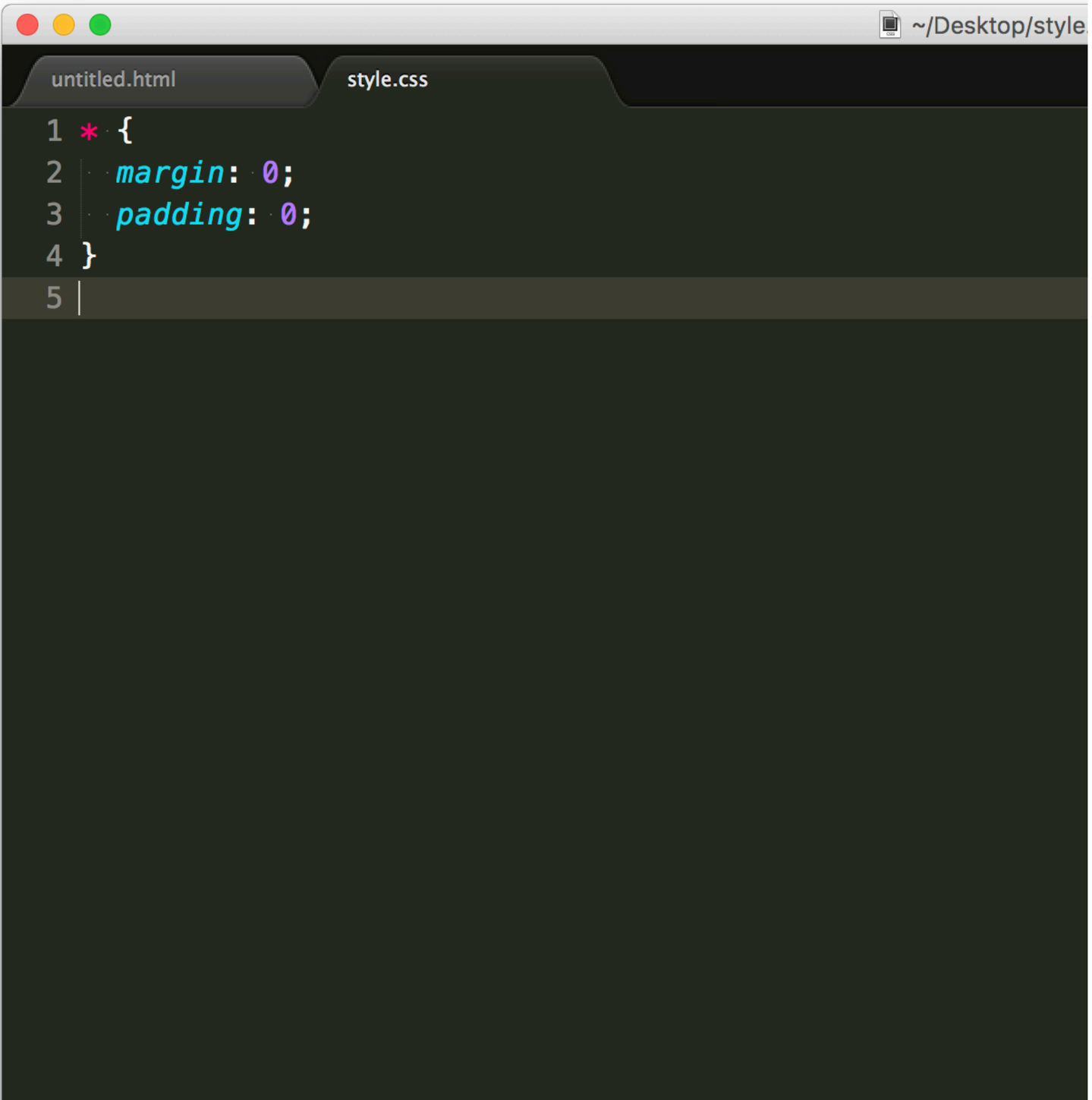
In order to center an element, a width must be set for that element. Otherwise, the width of the div will be automatically set to the full width of its containing element, like the <body>, for example. It's not possible, therefore, to center an element that takes up the full width of the page.



The width of the div is set to 400 pixels, which is less than the width of the page's body. This will cause the div to center properly on the page.

Note: When margin: auto is used, an element will center *relative* to its container. For example, the div in the previous example was centered relative to the width of the body. If the div was contained in larger div, the smaller div would center relative to the width of the larger div.

Browsers often have default CSS rules that set default values for padding and margin. This affects how the browser displays HTML elements, which can make it difficult for a developer to design or style a web page.



Many developers choose to reset these default values so that they can truly work with a clean slate.

The code in the example resets the default margin and padding values of all HTML elements. It is often the first CSS rule in an external stylesheet.

All HTML elements can be classified as one of the following: *inline* elements or *block-level* elements.

1. Inline elements - elements that display *inline* with text, without disrupting the flow of the text (like links).

2. Block-level elements - elements that use an entire line of space in a web page and disrupt the natural flow of text. Most of the common HTML elements are block-level elements (headings, paragraphs, divs, and more).

In CSS, you can change the default behavior of elements with the display property. Why is this useful?

Modifying the display property of an element can help achieve a desired layout for a web page

The display property can take on one of four values:

1. inline - causes block-level elements (like a div) to behave like an inline element (like a link).

2. block - causes inline elements (like a link) to behave like a block element (like a div).

3. inline-block - causes block-level elements to behave like an inline element, but retain the features of a block-level element.

4. none - removes an element from view. The rest of the web page will act as if the element does not exist.

~/Desktop/untitled.html

untitled.html | style.css

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Organizing HTML & CSS</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <ul>
      <li>Home</li>
      <li>Products</li>
      <li>Settings</li>
      <li>Inbox</li>
    </ul>
  </body>
</html>

```

~/Desktop/style.cs

untitled.html | style.css

```css
li {
  display: inline;
}

```

An example of inline-block. Notice what happens when you remove inline-block.