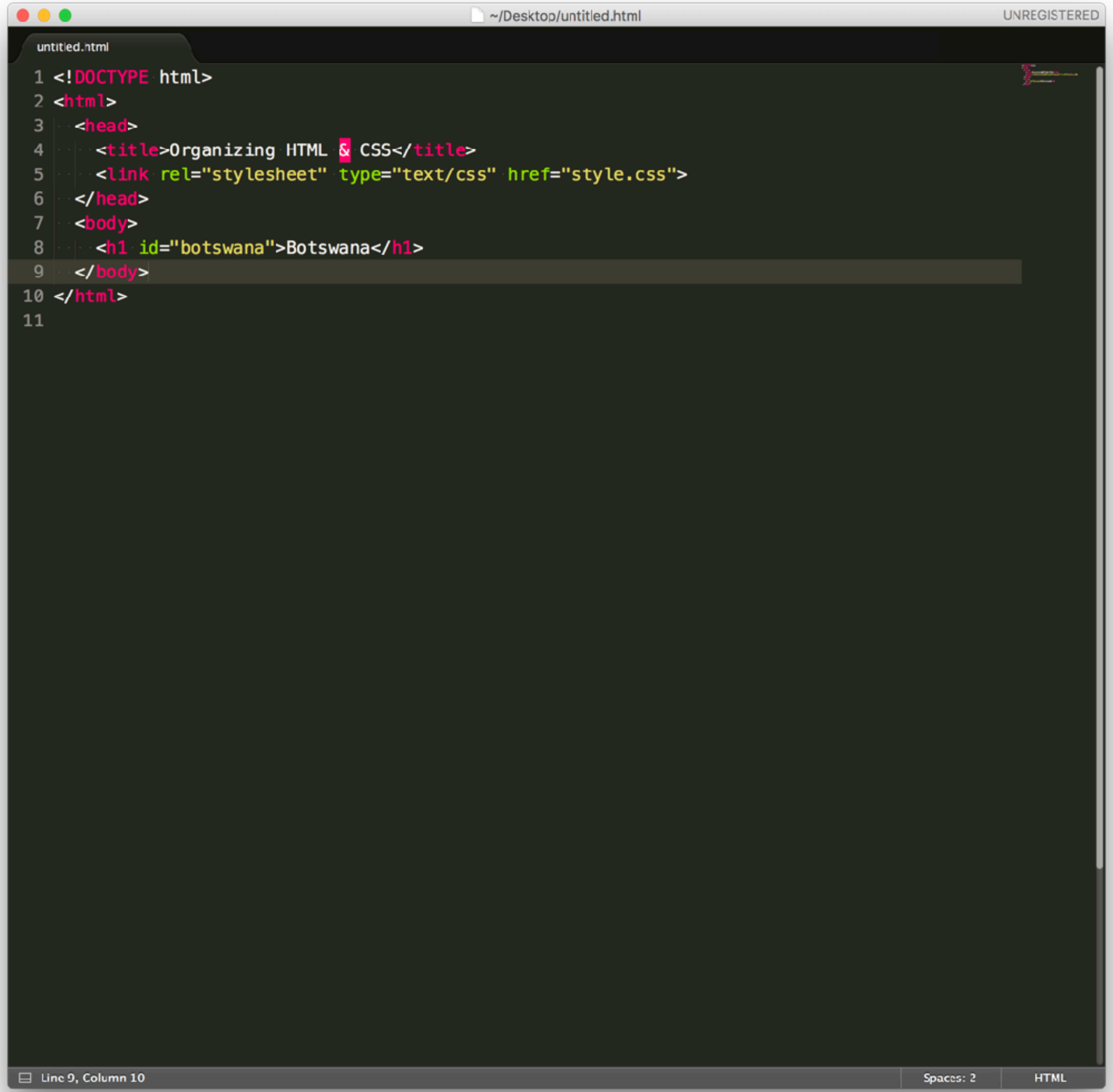
Organizing HTML & CSS

Up until this point we’ve been styling every type of a certain element. For example, all the paragraphs or all the h1 headers.

With the proper labels, we can style individual HTML elements! Specifically, we can label HTML elements with a unique identifier, or *ID*. We can then style that specific element in the stylesheet.

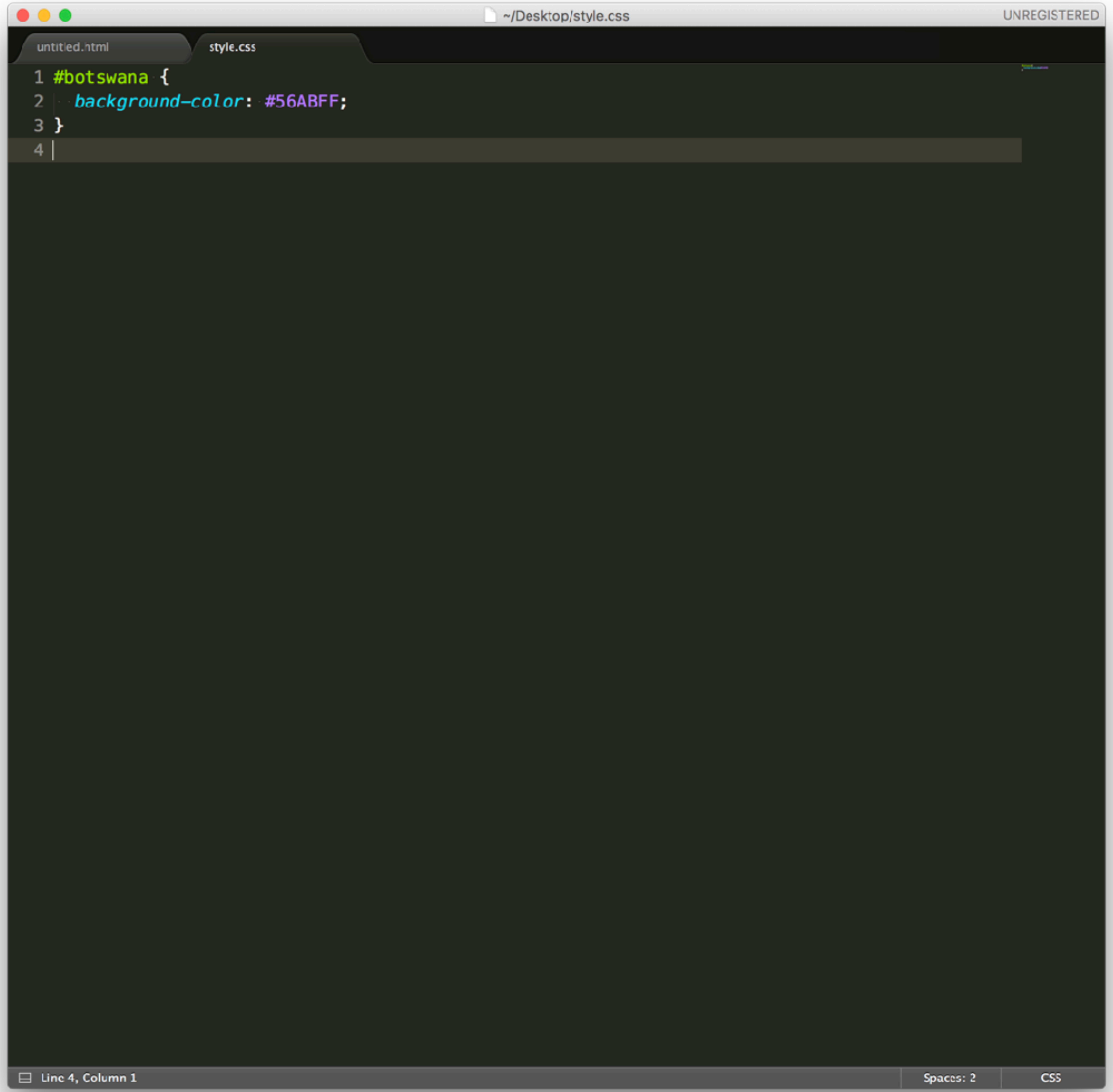


```
<!DOCTYPE html>
<html>
  <head>
    <title>Organizing HTML & CSS</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1 id="botswana">Botswana</h1>
  </body>
</html>
```

To label an element with an ID, we can use the id attribute on an HTML element.

What purpose do IDs serve? IDs are intended to label unique elements in an HTML file. No two HTML elements should ever share the same ID — that would defeat the purpose of a unique identifier!

Now that you know how to label HTML elements with an ID, we can style that specific element in the stylesheet.



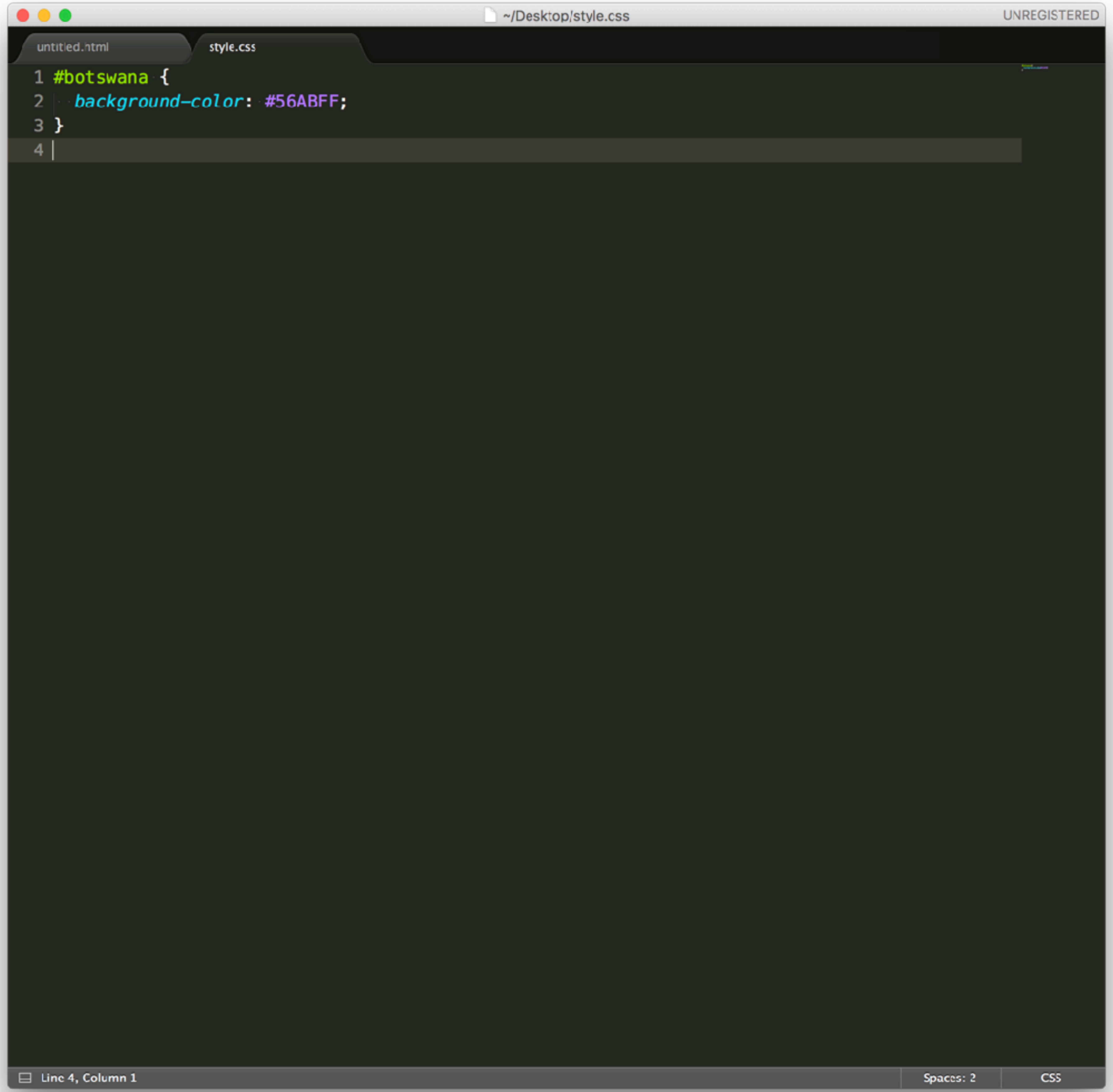
```
#botswana {
  background-color: #56ABFF;
}
```

To style a specific element labeled with an ID, you can use an *ID selector* in the stylesheet.

The ID selector always starts with a hash symbol.

IDs are great for labeling unique elements, but IDs have a limitation. Because unique IDs should not be used across multiple HTML elements, they are insufficient for quickly styling elements that should all share a specific style.

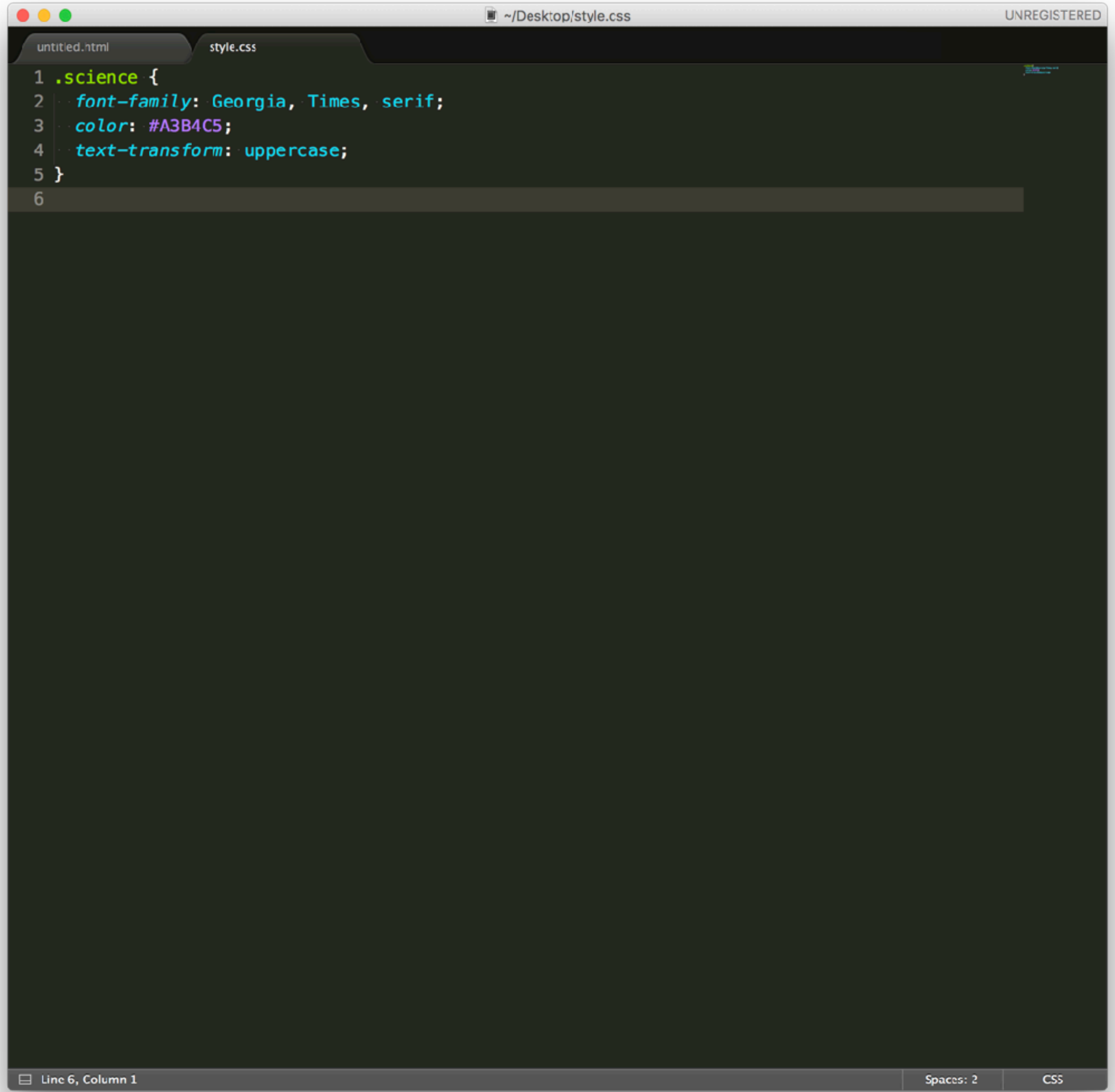
CSS offers a solution to this limitation. For multiple elements that should share the same styling, *classes*can be used to label them.



```
#botswana {
  background-color: #56ABFF;
}
```

To label an element with a class, we can use the **class** attribute on an HTML element.

Now that you know how to label HTML elements with a class, we can style elements belonging to the same class at once. How exactly do you select them in CSS, though?



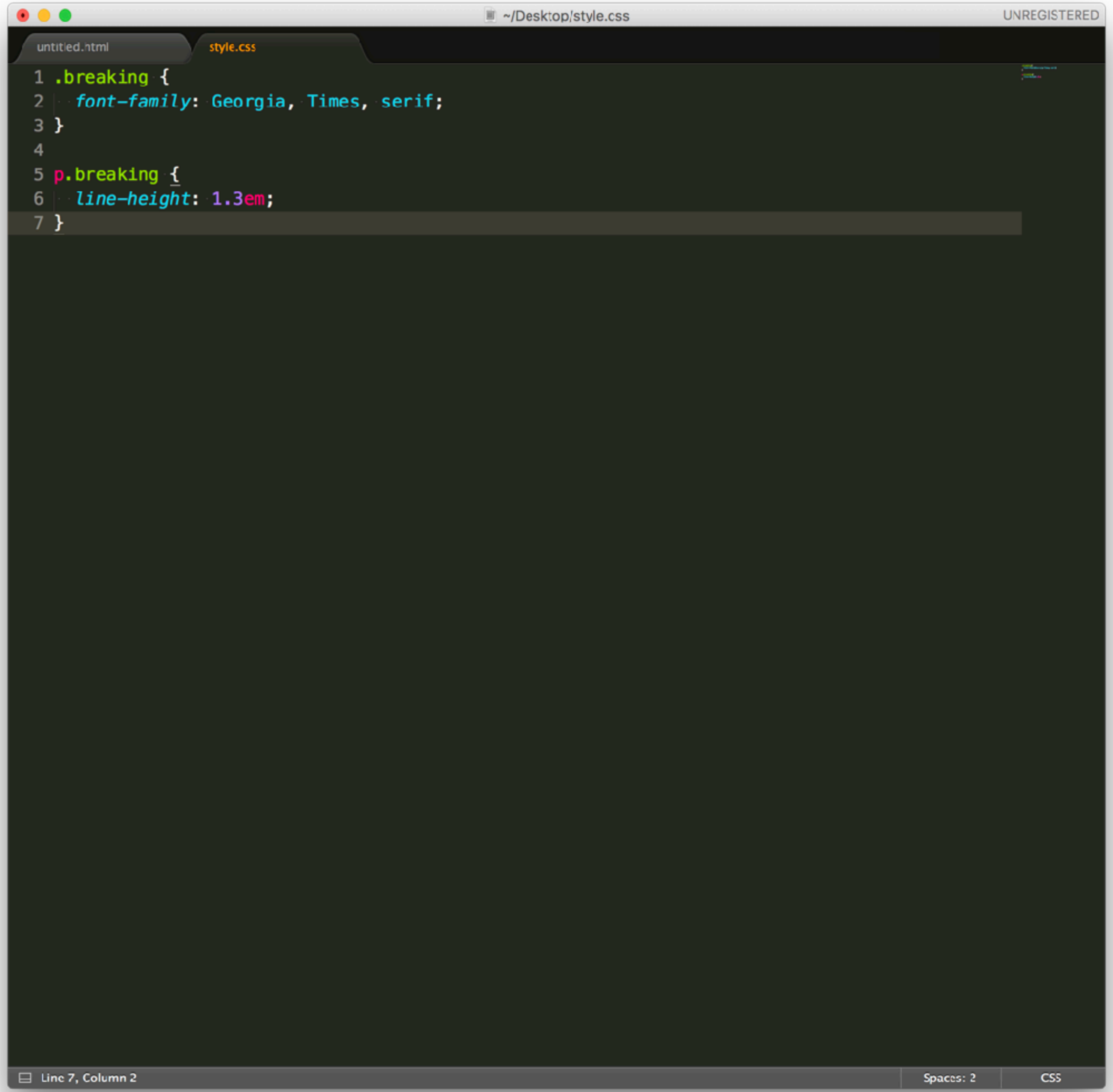
```css
.science {
  font-family: Georgia, Times, serif;
  color: #A3B4C5;
  text-transform: uppercase;
}

```

Class selectors begin with a period (.) and are immediately followed by the name of the class.

Classes are by far the most commonly used for styling groups of elements.

The class selector targets *all* elements of a particular class. It's possible, however, for multiple elements on a web page to share a specific styling, but for one of those elements to differ slightly.

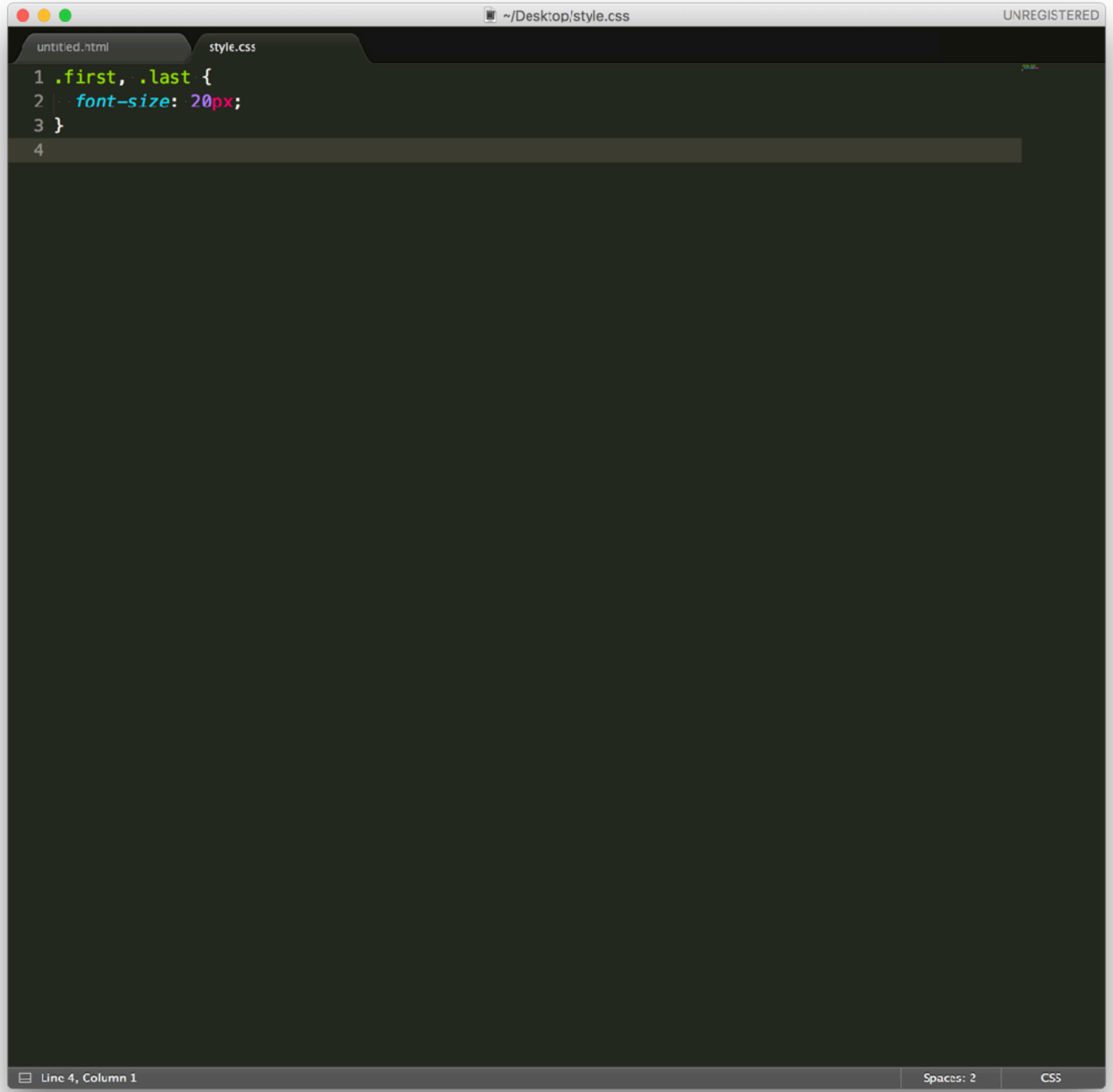
For example, a heading and a paragraph (both with a class of breaking) may need to share the same typeface, but the paragraph may require a styling better suited for paragraphs, as in the following example.



```css
.breaking {
  font-family: Georgia, Times, serif;
}

p.breaking {
  line-height: 1.3em;
}
```

The .breaking selector targets *all* elements with a class of breaking.
The p.breaking selector targets *only* <p>elements with a class of breaking. This type of selector allows you to be even more specific about a particular element's styling, even when that element must share some styling with other elements.

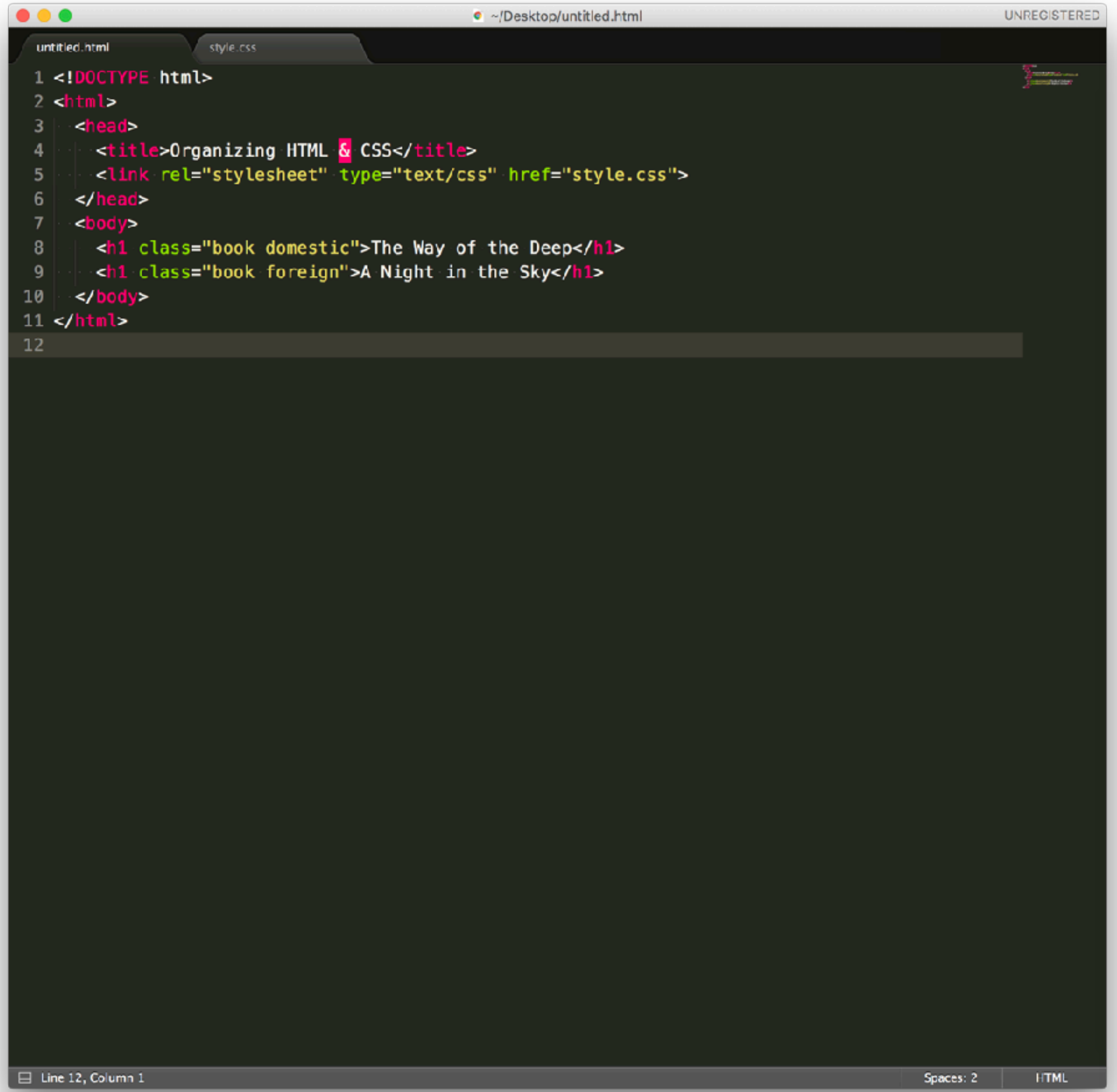
Note that multi-selectors can be used with classes and IDs just as we saw with HTML tags…



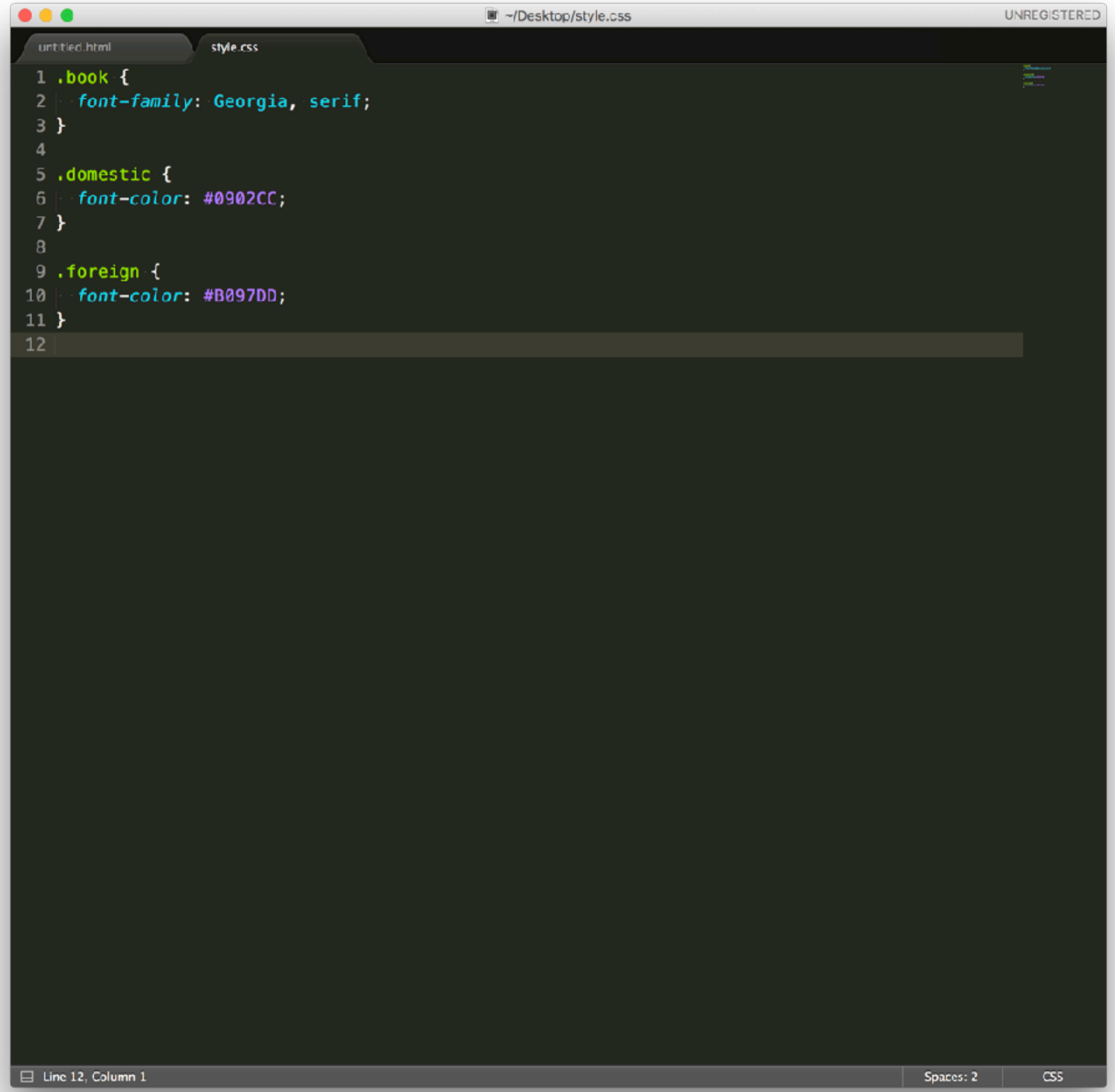
```css
.first, .last {
  font-size: 20px;
}

```

Using a multiple class selector is an efficient way of styling multiple HTML elements.

It's also possible to label HTML elements with more than one class.



```html
<!DOCTYPE html>
<html>
  <head>
    <title>Organizing HTML & CSS</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1 class="book domestic">The Way of the Deep</h1>
    <h1 class="book foreign">A Night in the Sky</h1>
  </body>
</html>

```

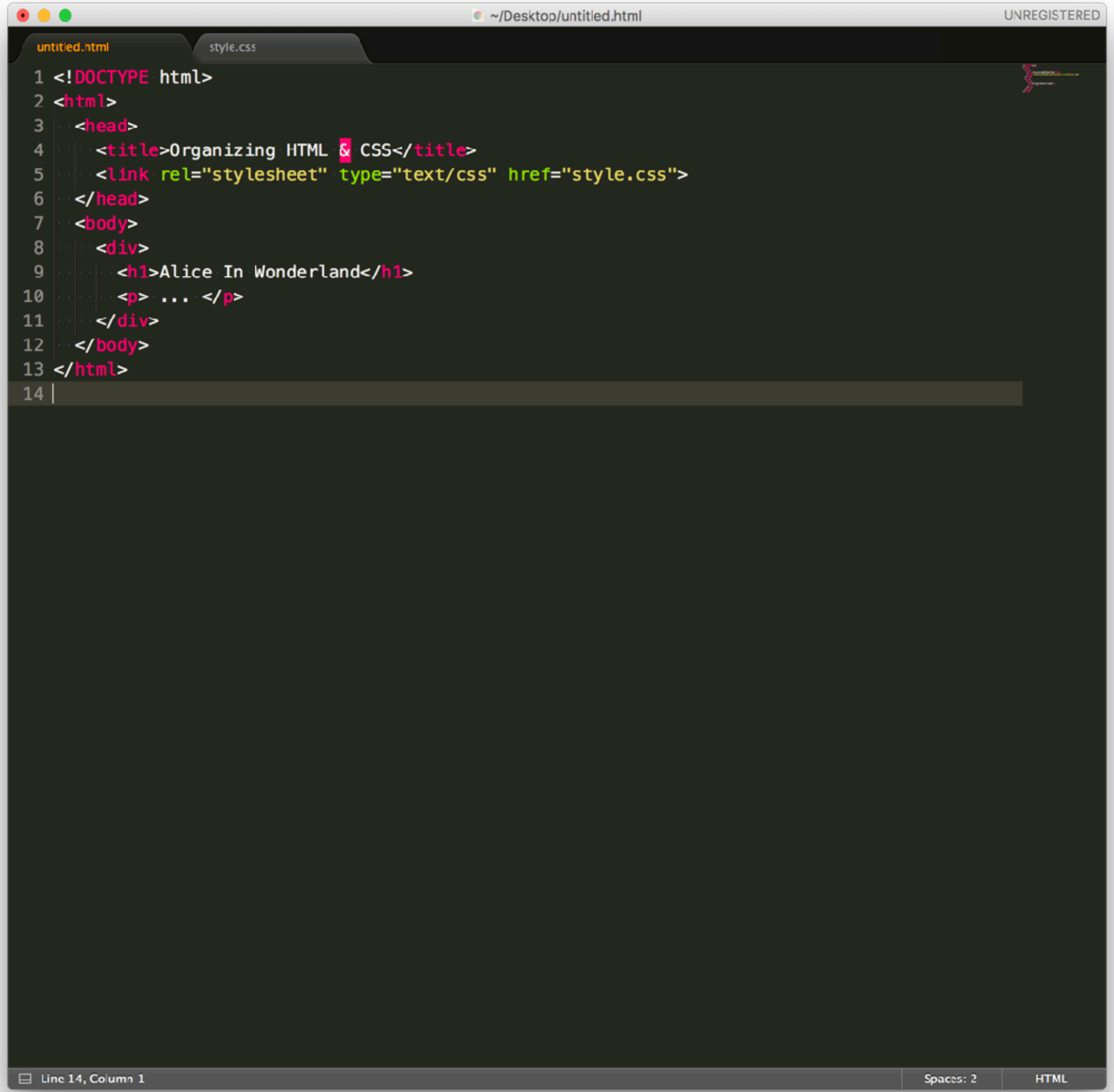


```css
.book {
  font-family: Georgia, serif;
}

.domestic {
  font-color: #0902CC;
}

.foreign {
  font-color: #B097DD;
}

```

HTML offers an element that is the backbone of code organization: the *div*, represented by <div> in HTML.

You can think of the div as a box, or container, that groups elements that belong together.

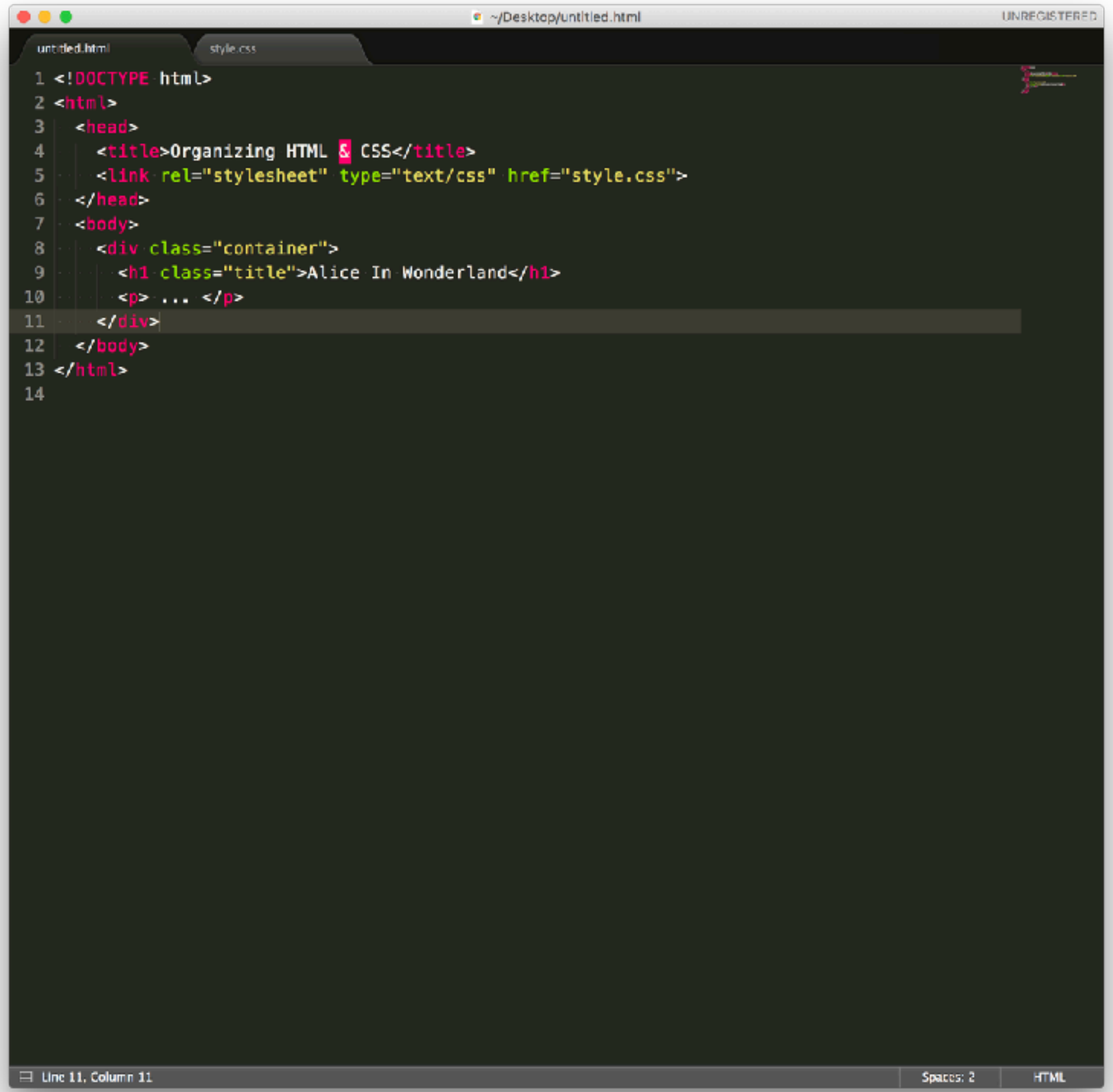
For example, a <div> can be used to group together all of the elements that make up the navigation for a web page, or any other section of a page.



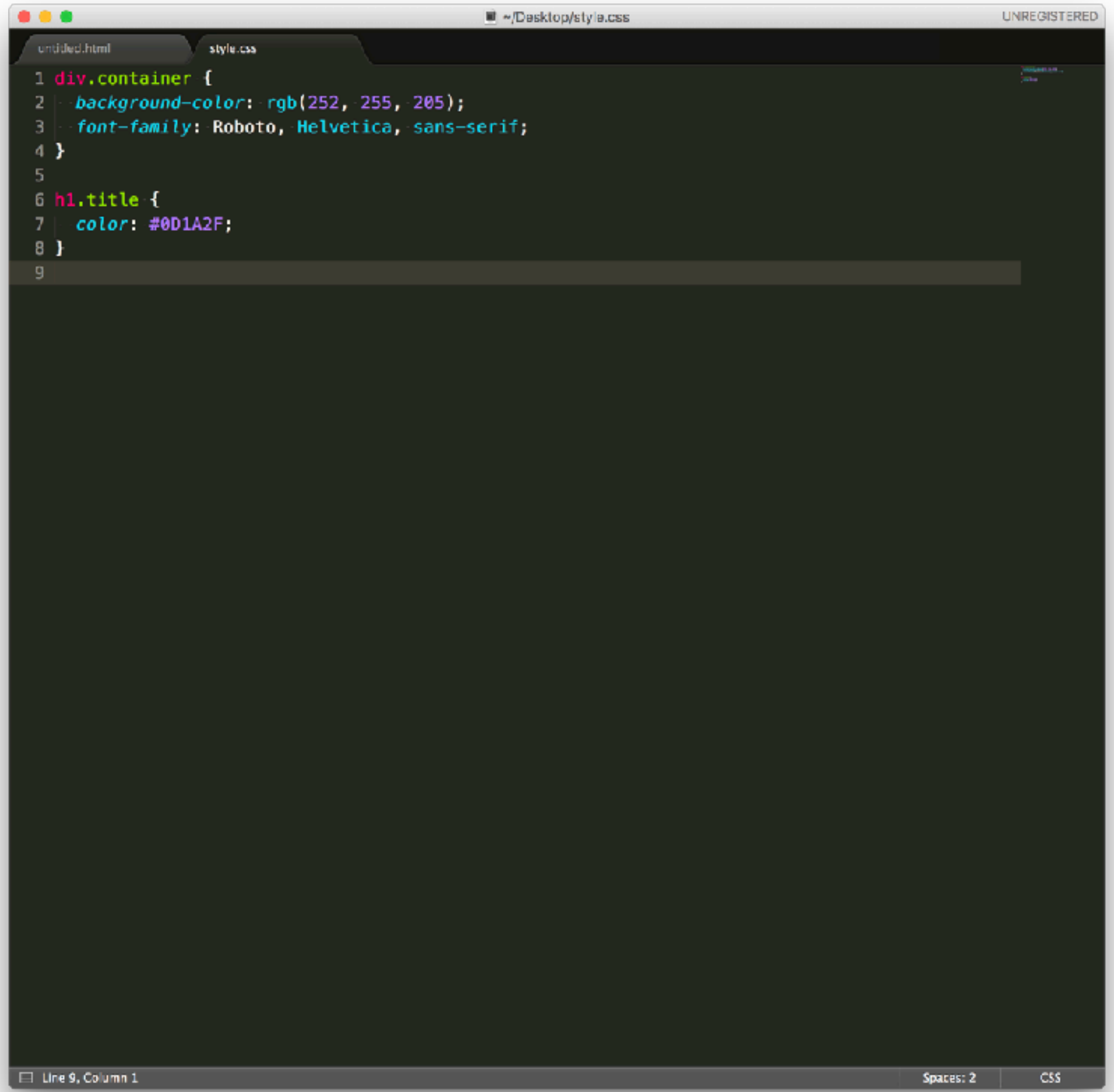
```html
<!DOCTYPE html>
<html>
  <head>
    <title>Organizing HTML & CSS</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <div>
      <h1>Alice In Wonderland</h1>
      <p> ... </p>
    </div>
  </body>
</html>
```

A heading for "Alice In Wonderland" and a brief paragraph description of the book are contained within a single <div>. If more books were to appear on the page, additional divs could be used to organize them. This would keep the code easier to read, maintain, and style.

Now that you know how to organize code into sections using divs, we can start labeling divs with classes instead, rather than labeling individual elements with classes.



```html
<!DOCTYPE html>
<html>
  <head>
    <title>Organizing HTML & CSS</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <div class="container">
      <h1 class="title">Alice In Wonderland</h1>
      <p> ... </p>
    </div>
  </body>
</html>
```

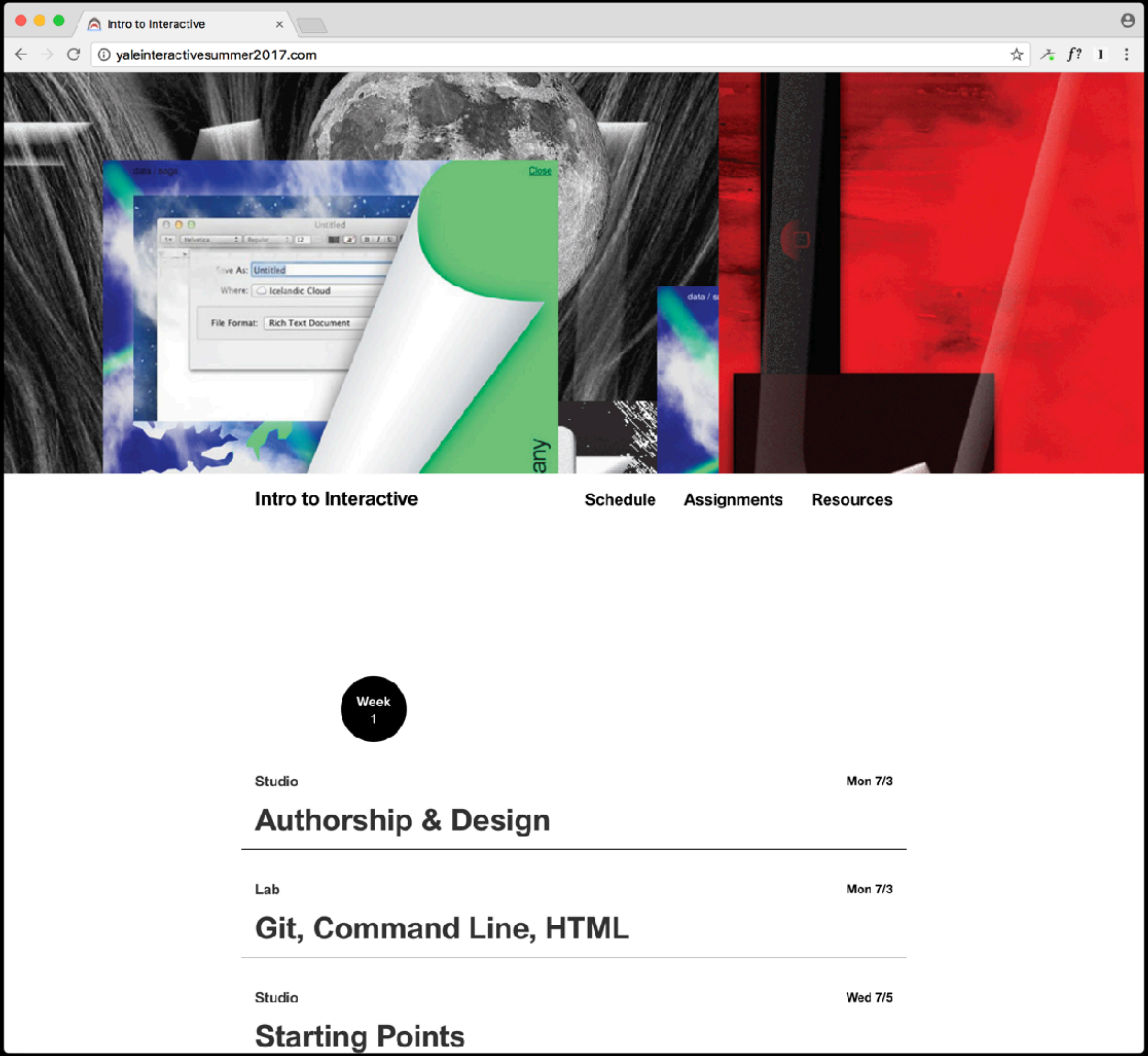


```css
div.container {
  background-color: rgb(252, 255, 205);
  font-family: Roboto, Helvetica, sans-serif;
}

h1.title {
  color: #0D1A2F;
}
```

When a div is styled, *all elements inside* of the div will *inherit* the styling applied to the div. This example illustrates how easy it is to style sections of a web page using div.

The div is one of the most commonly used elements in all of HTML. Modern web pages make extensive use of the div, and learning how to use divs for organization and styling is a critical skill.

Take a look at the next page and think about what elements might be grouped into divs


Intro to Interactive

yaleinteractivesummer2017.com

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

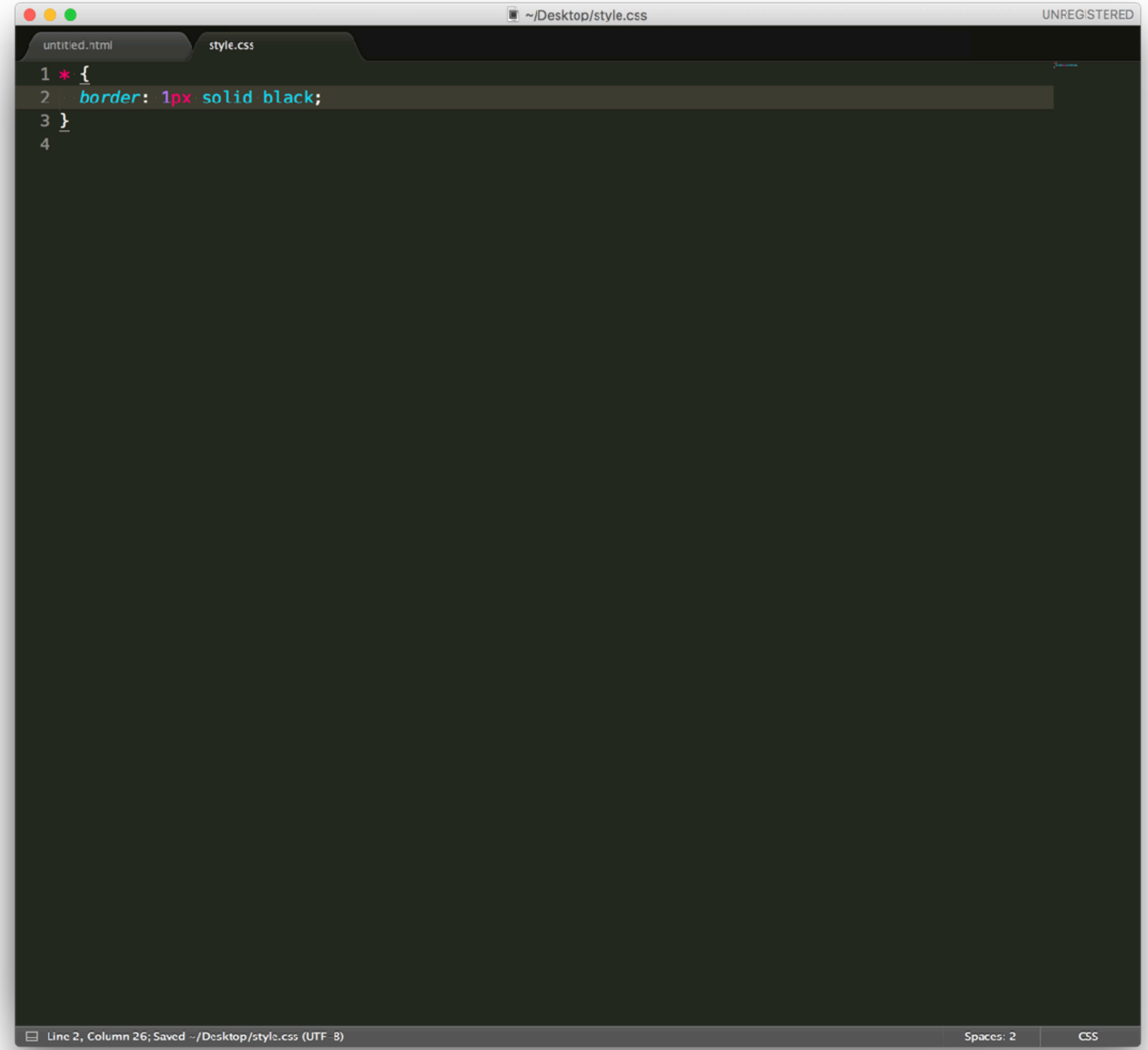
Starting Points

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*.

When you changed the background color of an element, you changed the background color of its entire box. When you aligned text, the text was aligned relative to the element's entire box.

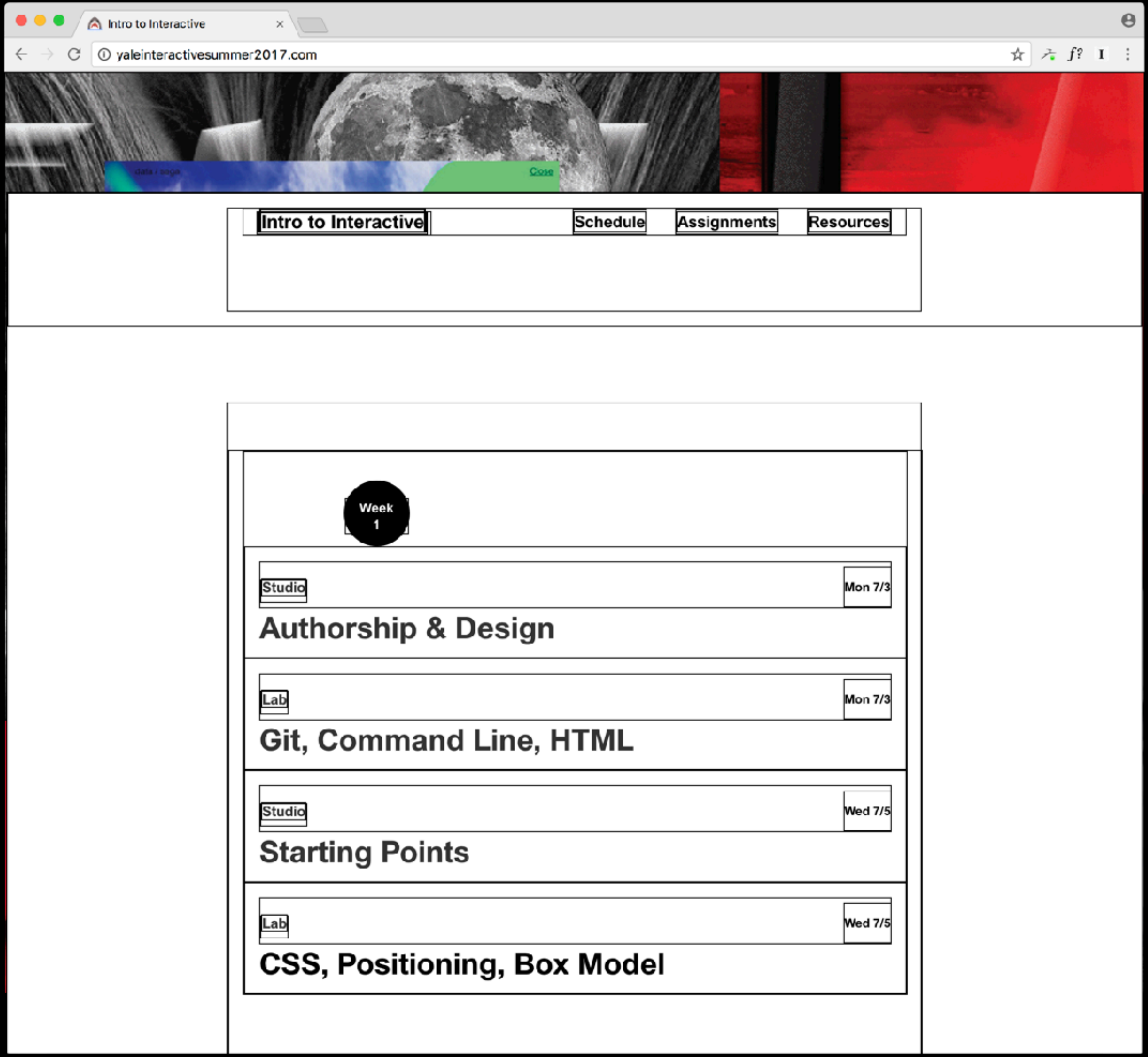
Where are the boxes that supposedly contain all HTML elements? They're invisible, so we'll have to reveal them!



```
* {
  border: 1px solid black;
}
```

Line 2, Column 26; Saved ~/Desktop/style.css (UTF-8)

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

yaleinteractivesummer2017.com

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

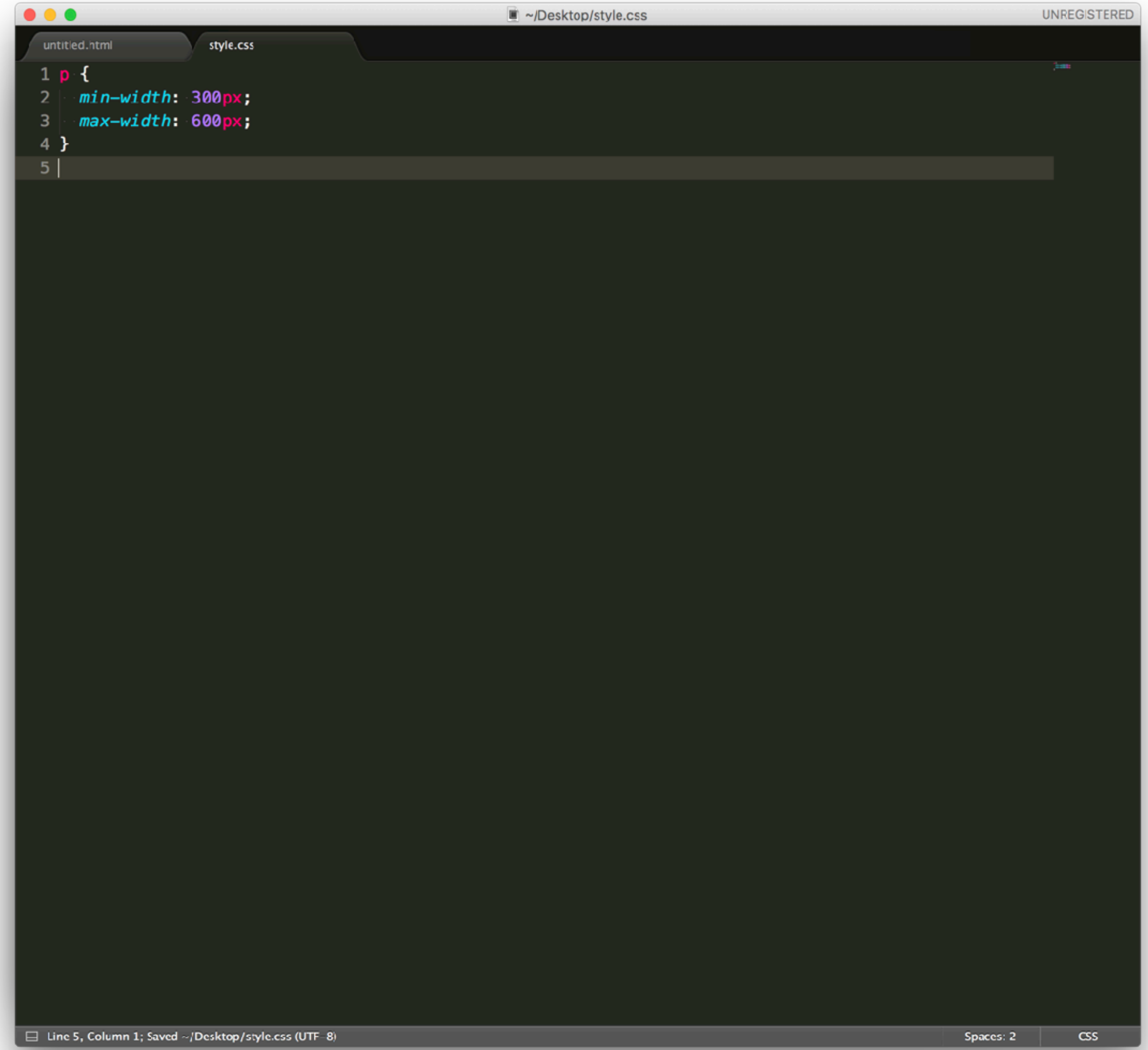
CSS, Positioning, Box Model

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 hold the raw contents of the box.

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.

Because a web page can be viewed through displays of differing screen size, the content on the web page can suffer from those changes in size. To avoid this problem, CSS offers two properties that can limit how narrow or how wide an element's box can be sized to.

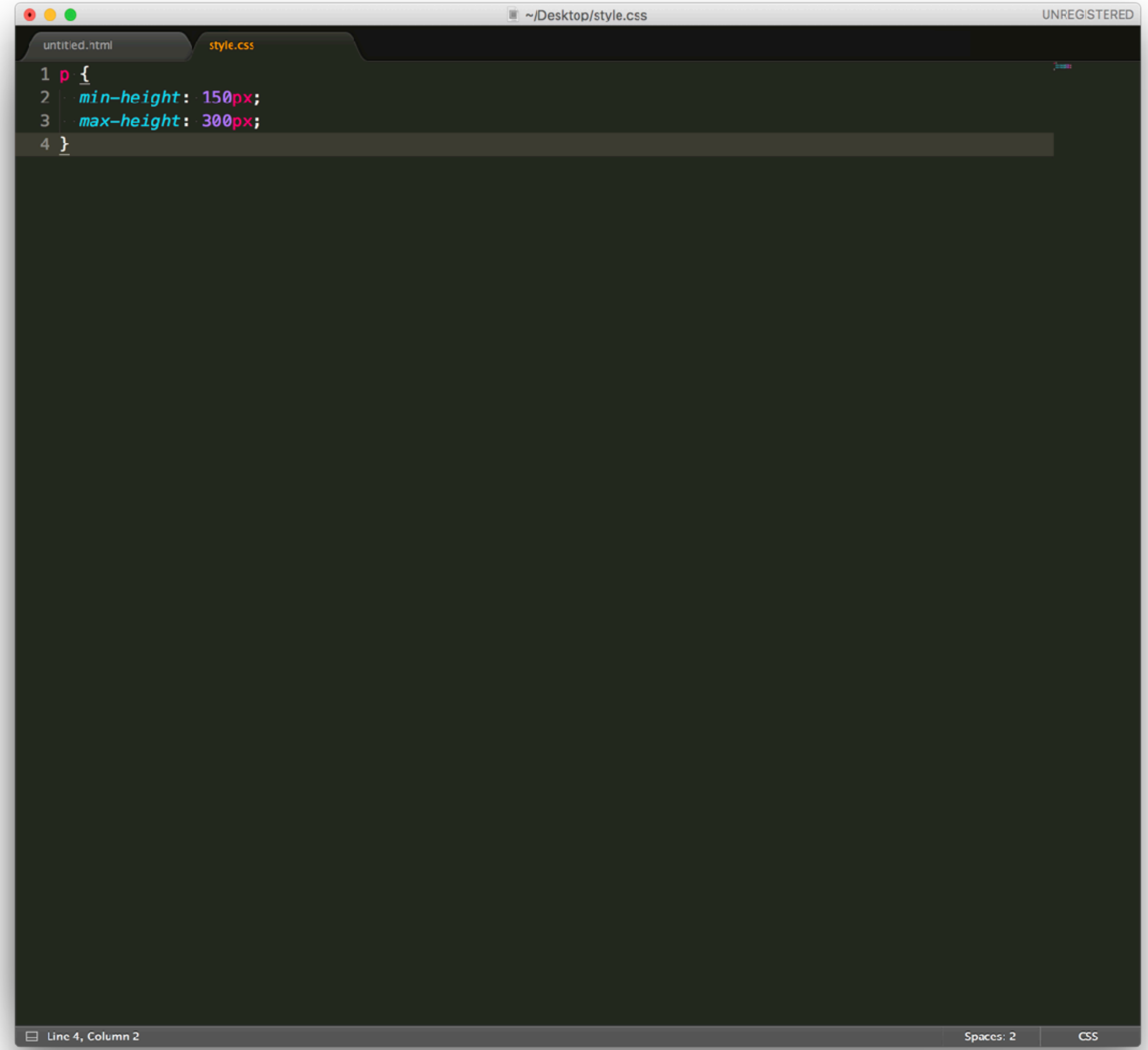


```css
p {
  min-width: 300px;
  max-width: 600px;
}

```

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.

Content, like text, can become difficult to read when a browser window is narrowed or expanded. These two properties ensure that content is legible by limiting the minimum and maximum widths of an element.


untitled.html
style.css

```
p {
  min-height: 150px;
  max-height: 300px;
}
```

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

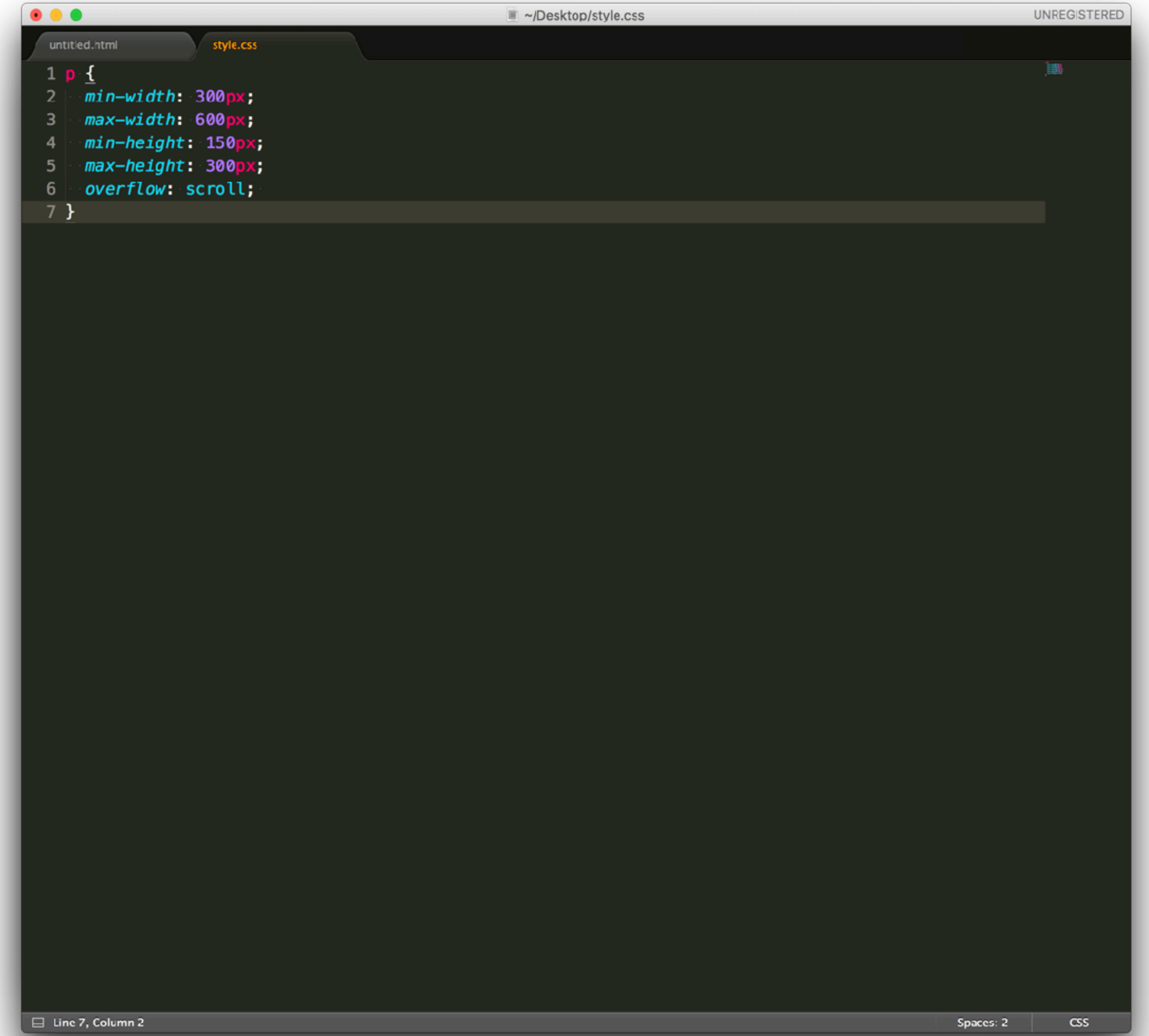
min-height – this property ensures a minimum height for an element's box.

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

How can we ensure that this doesn't happen?

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



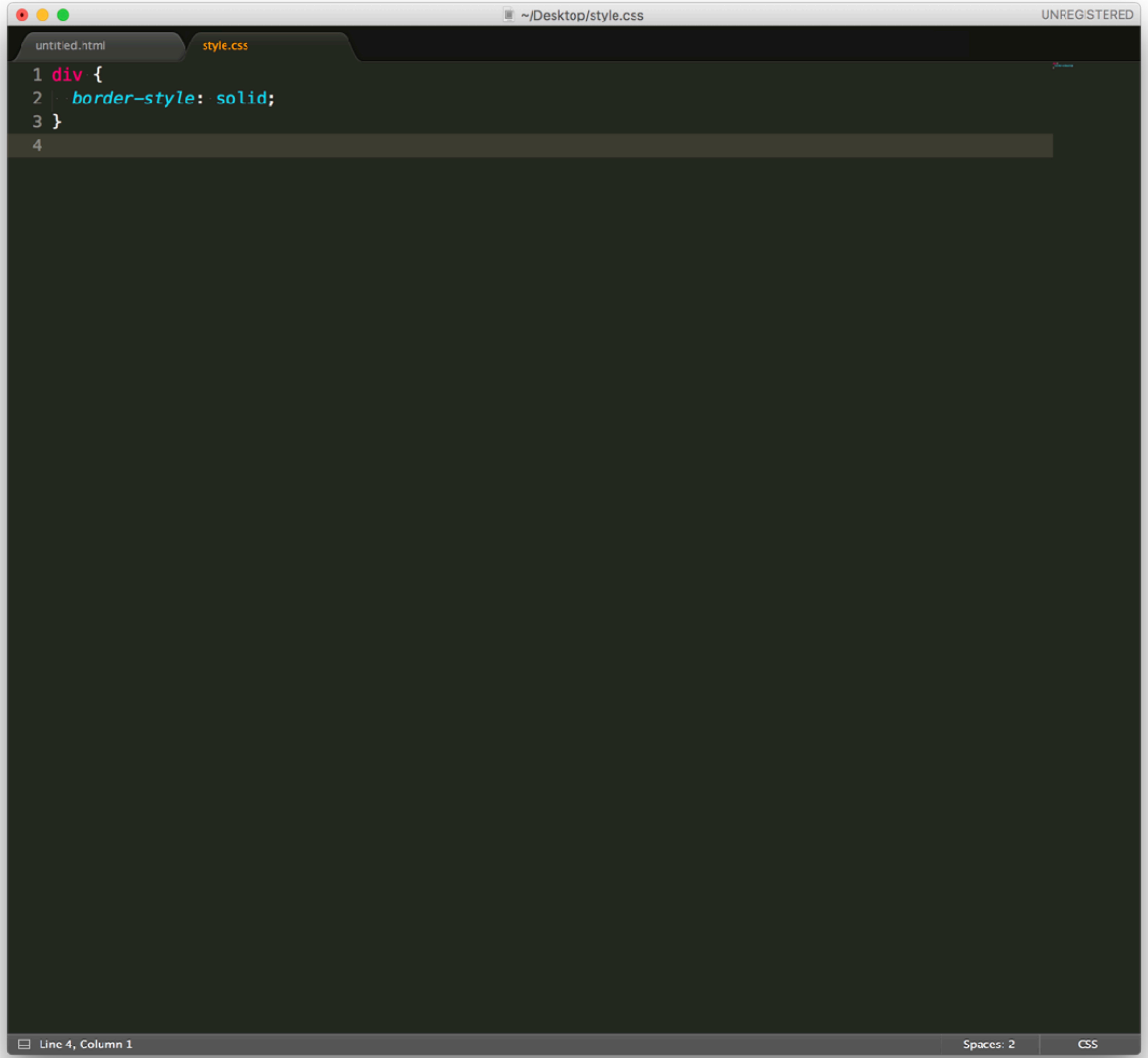
```css
p {
  min-width: 300px;
  max-width: 600px;
  min-height: 150px;
  max-height: 300px;
  overflow: scroll;
}
```

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.

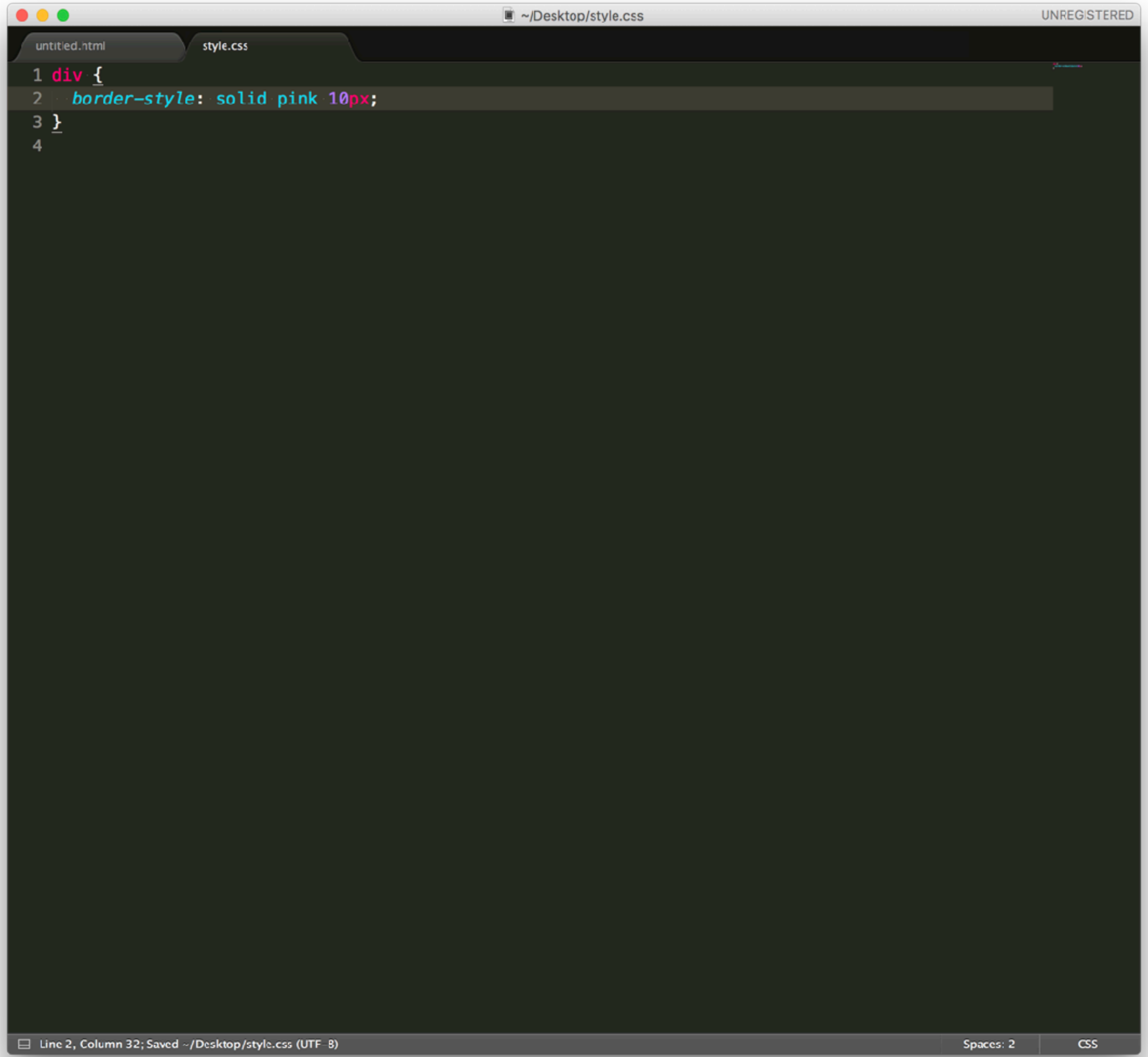
It's not possible to view a box's border if the border's *style* has not been set. A border's style can be set with the border-style property.



```css
div {
  border-style: solid;
}

```

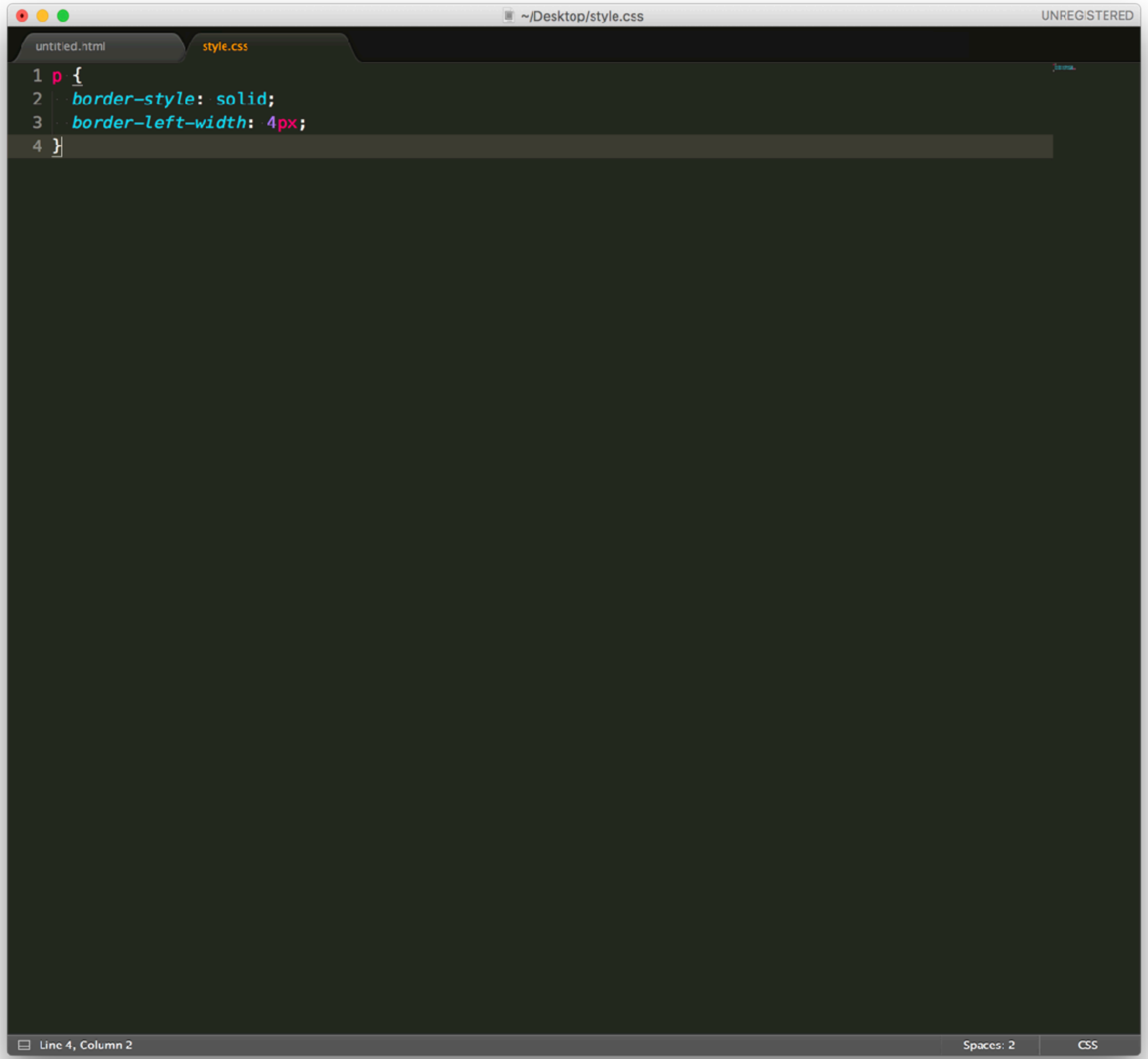
This property can take on one of the following values:

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



```css
div {
  border-style: solid pink 10px;
}
```

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



~/Desktop/style.css

untitled.html | style.css

```css
p {
  border-style: solid;
  border-left-width: 4px;
}
```

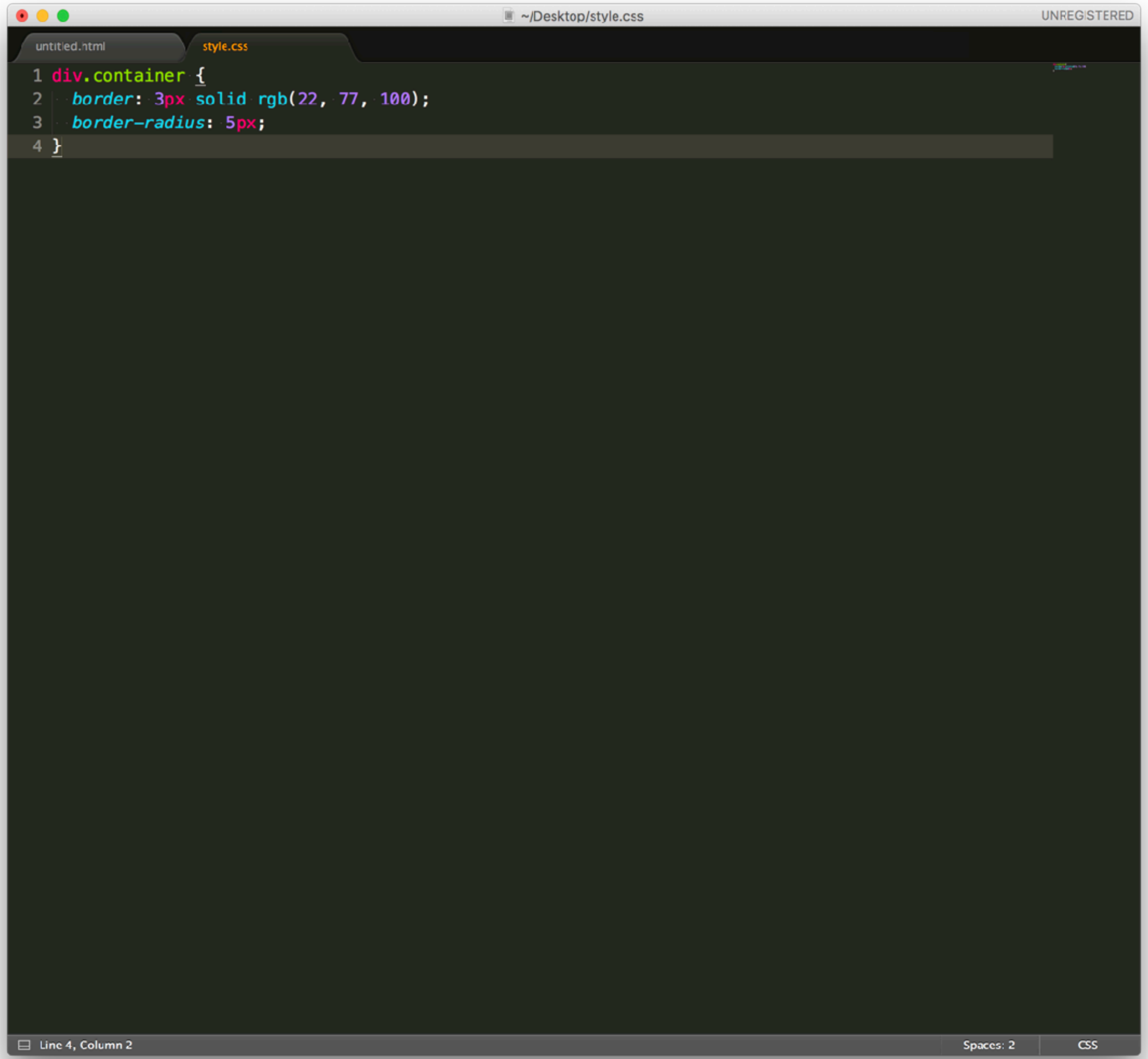
Line 4, Column 2 | Spaces: 2 | CSS

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.

Ever since we revealed the borders of boxes, you may have noticed that the borders highlight the true shape of an element's box: square. Thanks to CSS, a border doesn't have to be square.



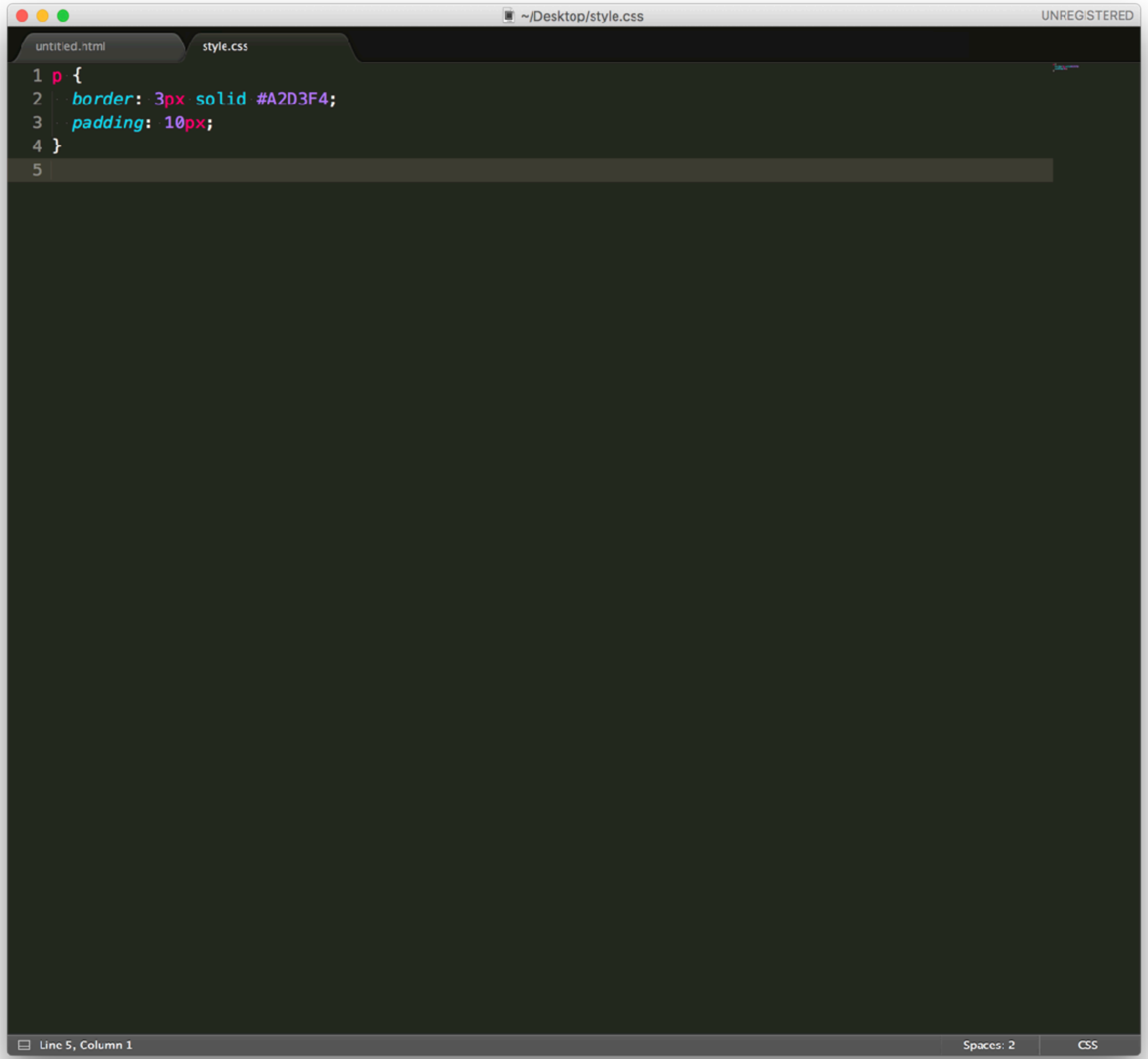
```css
div.container {
  border: 3px solid rgb(22, 77, 100);
  border-radius: 5px;
}
```

The corners of an element's border box can be modified with the `border-radius` property.

You can create a border that is a perfect circle by setting the radius equal to the height of the box, or to `100%`.

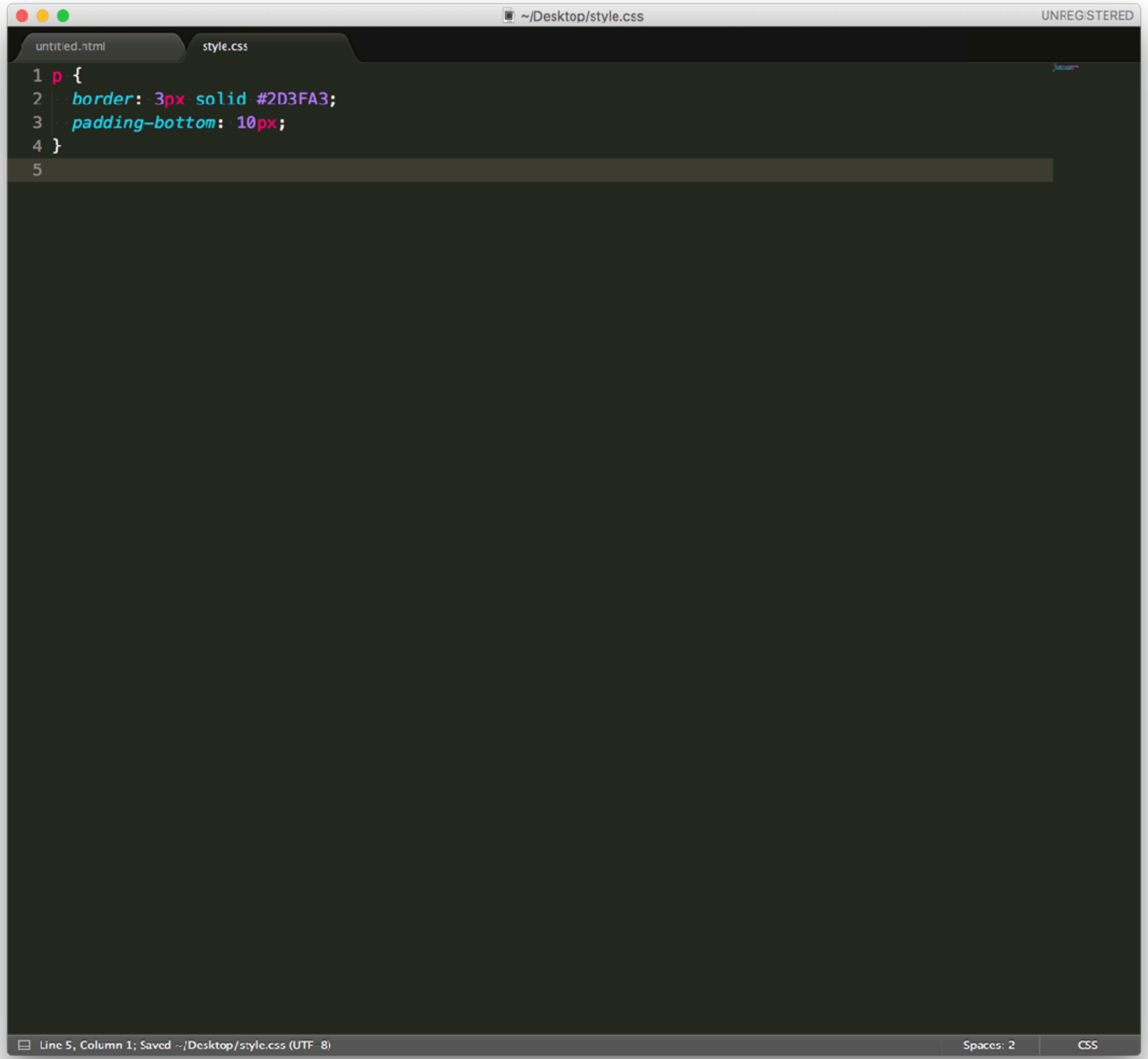
Box dimensions and borders are just the beginning of the vast number of box properties you can modify in CSS. Lets take a look at how to modify the spacing around the content *inside* of the box.

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



```css
p {
  border: 3px solid #A2D3F4;
  padding: 10px;
}
```

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.



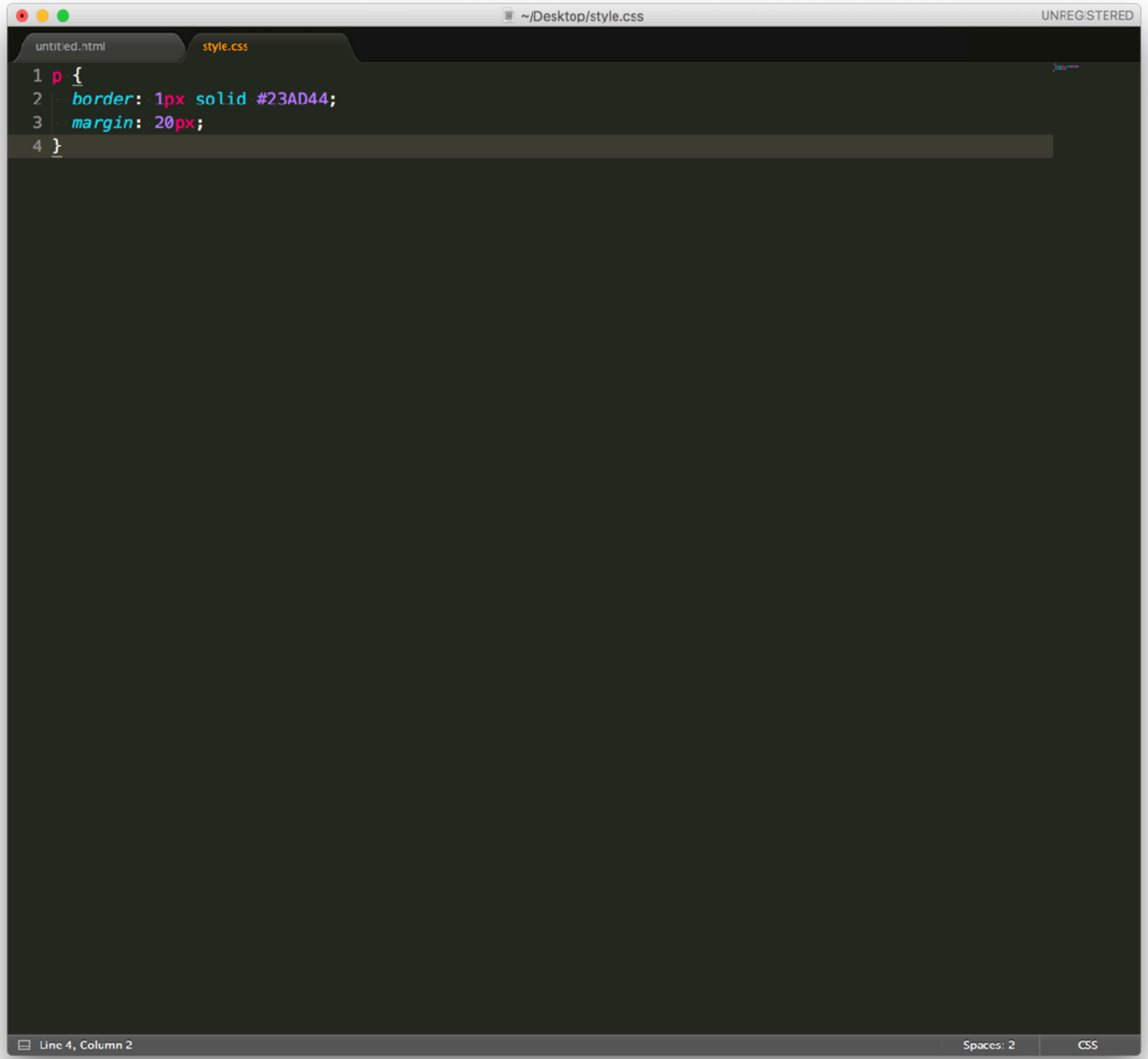
```css
p {
  border: 3px solid #2D3FA3;
  padding-bottom: 10px;
}
```

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

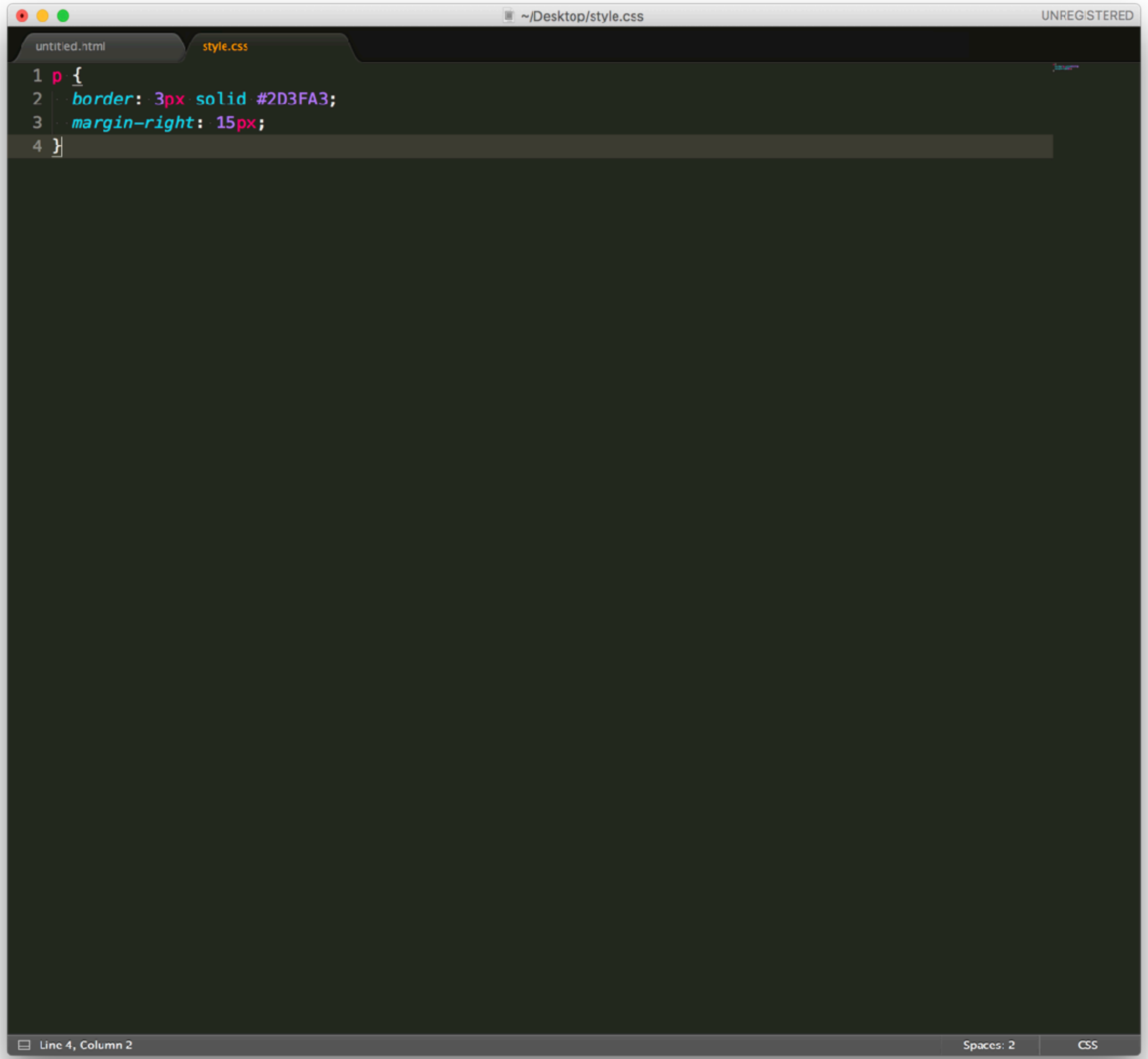
So far, we’ve learned about the following aspects of the box model: dimensions, borders, and padding. The fourth and final aspect of the box model is *margin*.

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



```css
p {
  border: 1px solid #23AD44;
  margin: 20px;
}
```

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

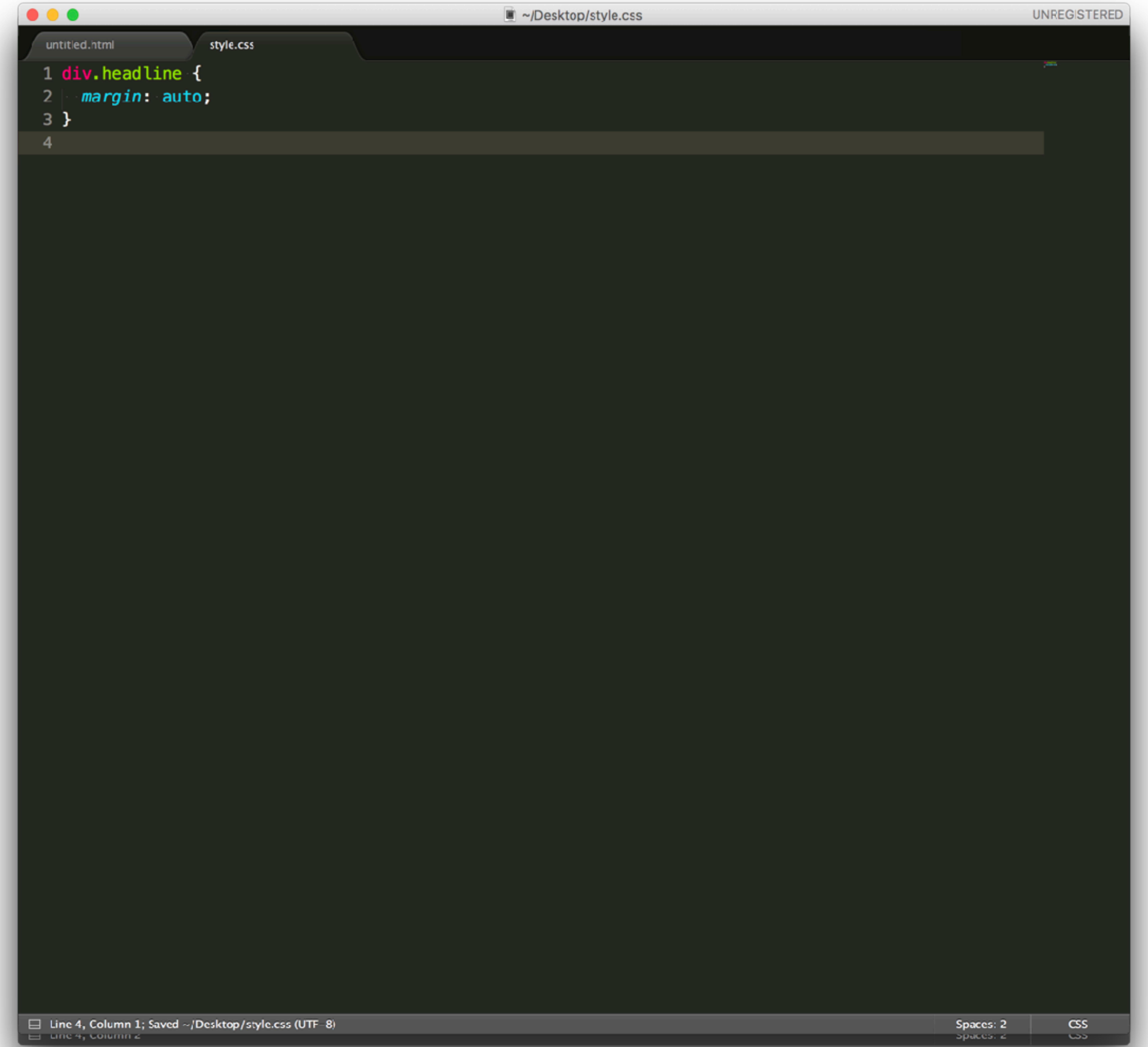


```css
p {
  border: 3px solid #2D3FA3;
  margin-right: 15px;
}
```

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.

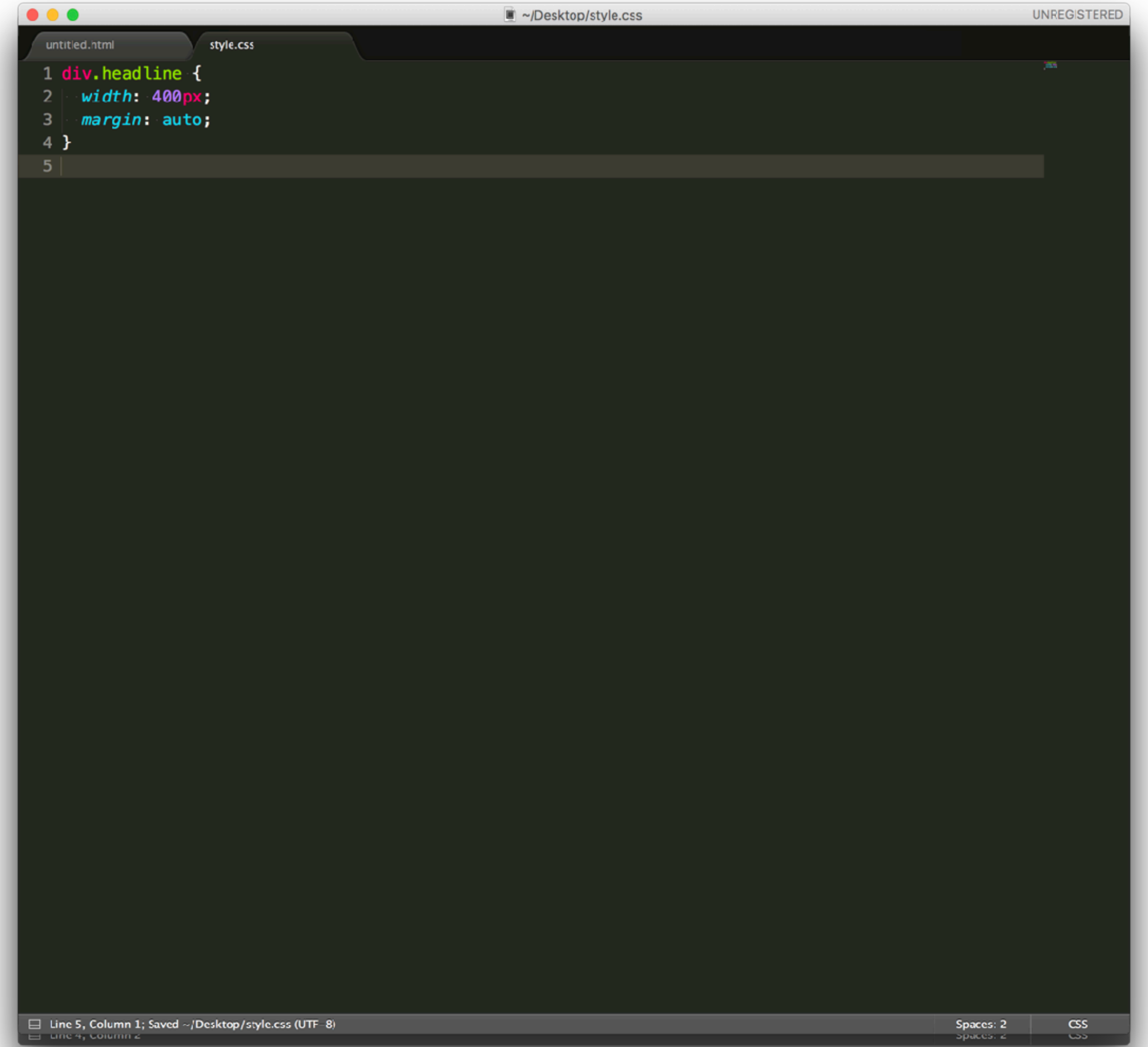


```
div.headline {
  margin: auto;
}
```

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.

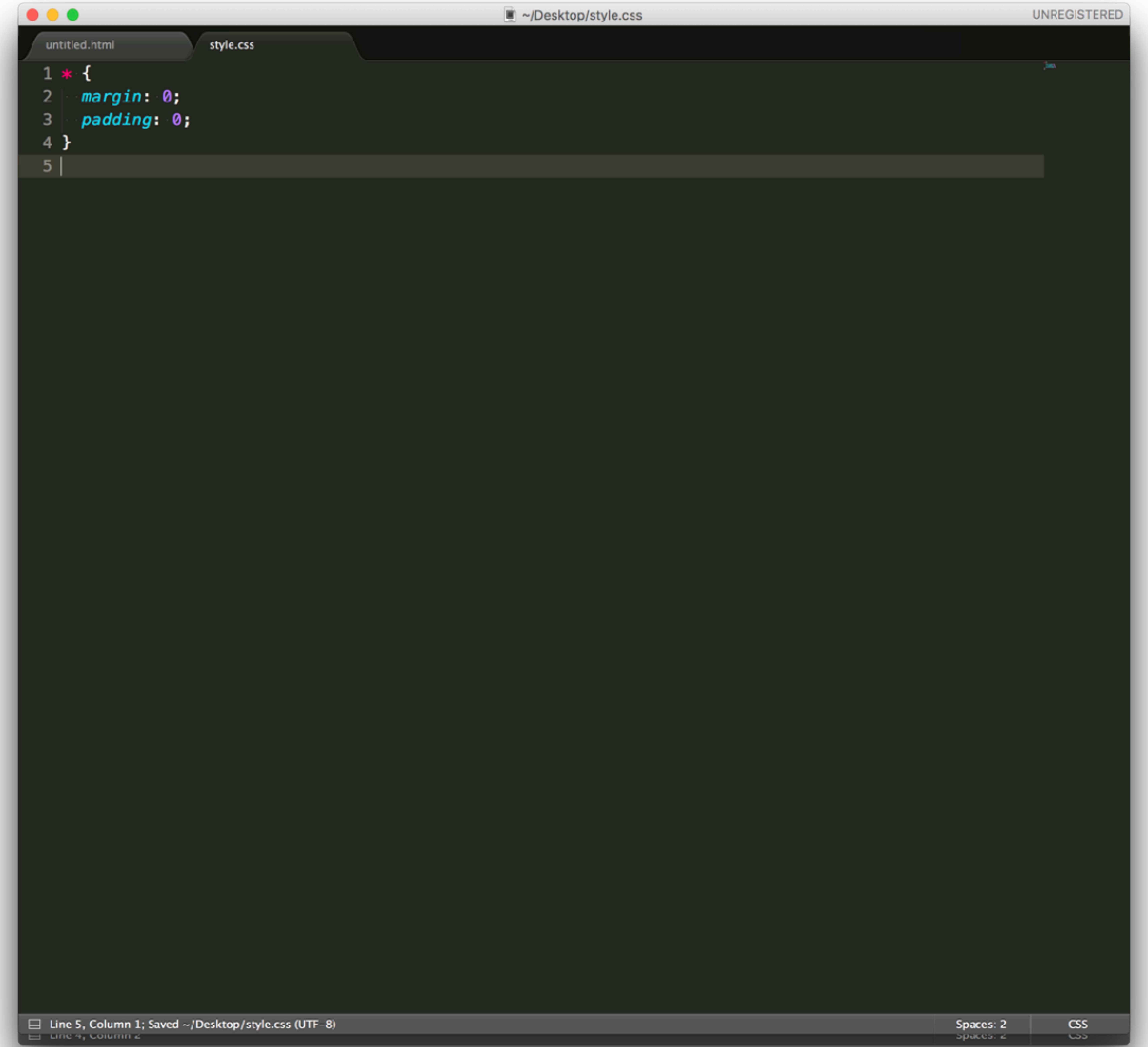


```css
div.headline {
  width: 400px;
  margin: auto;
}
```

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 example above 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.



```css
* {
  margin: 0;
  padding: 0;
}
```

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

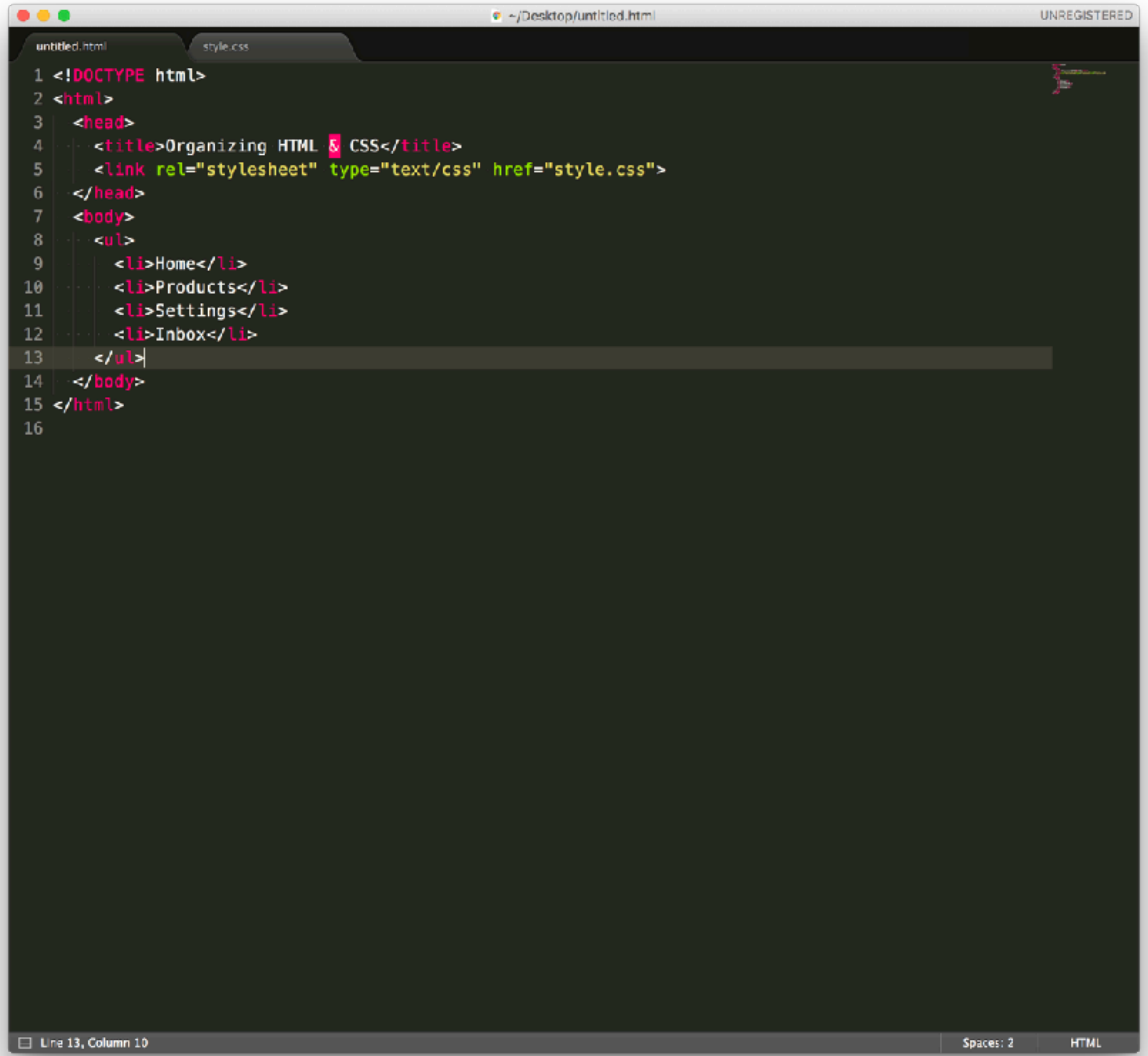
The display property can take on one of four values:

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.

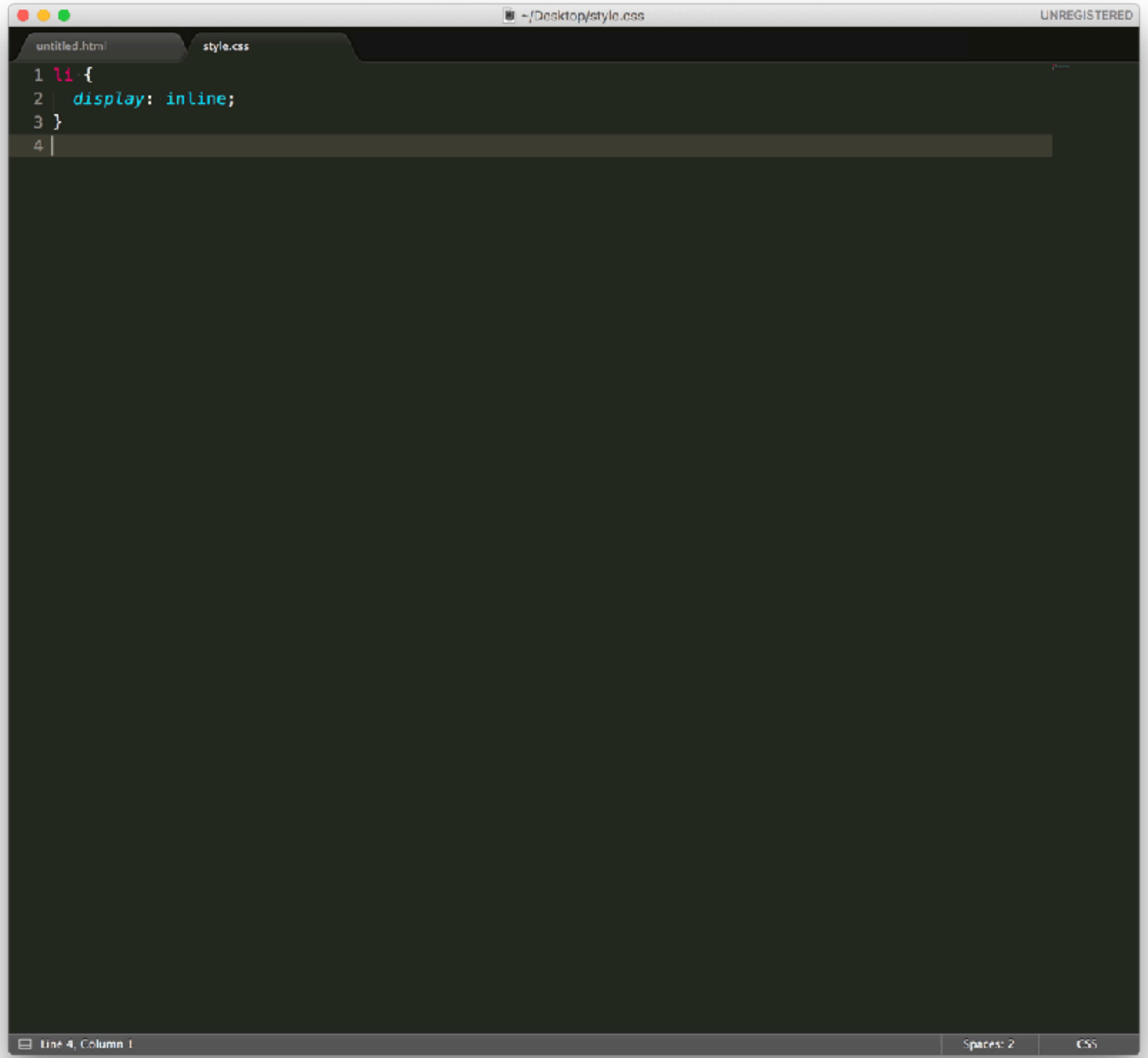
4. none - removes an element from view. The rest of the web page will act as if the element does not exist.


~/Desktop/untitled.html UNREGISTERED

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>

```

Line 13, Column 10 Spaces: 2 HTML

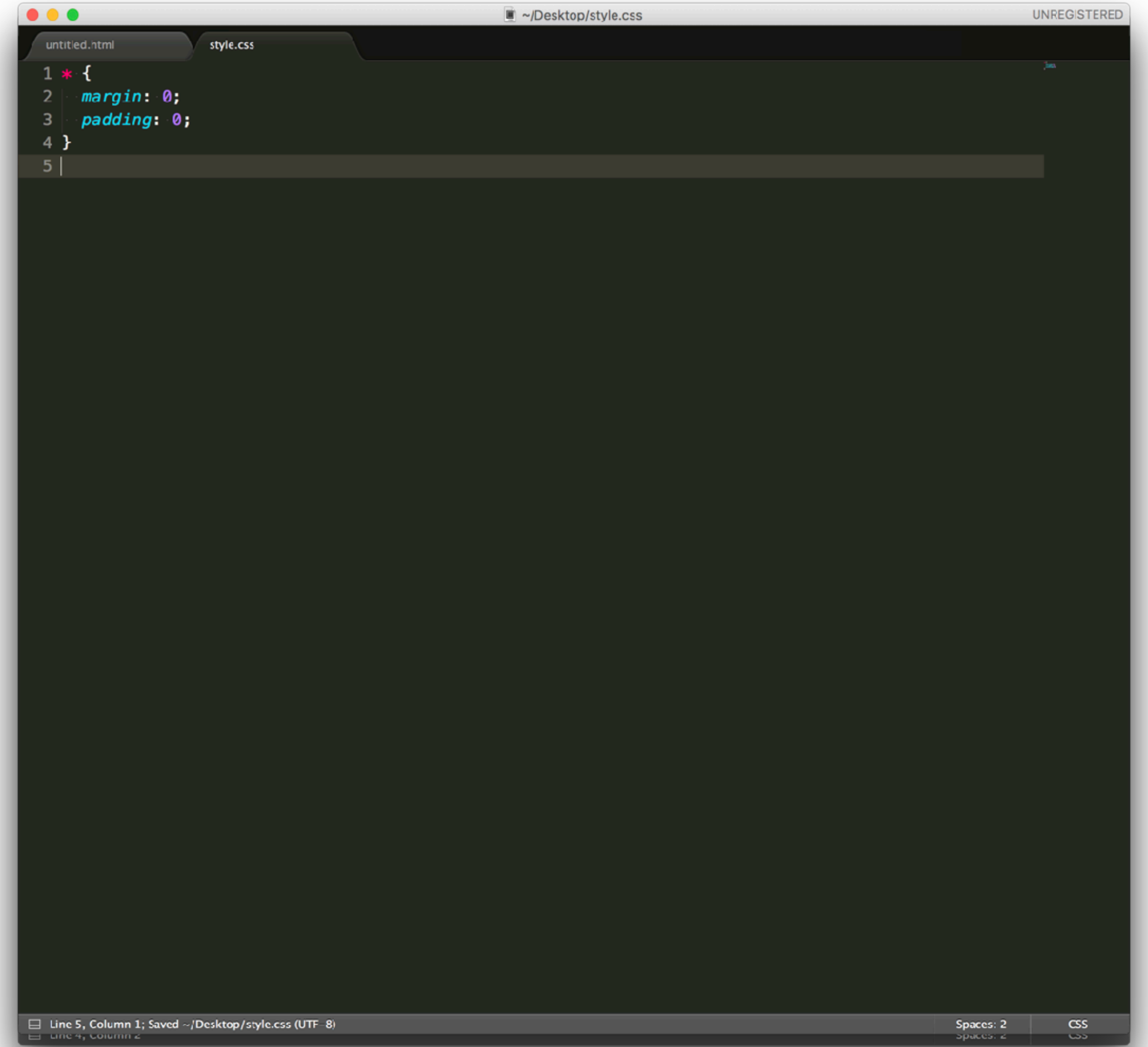

~/Desktop/style.css UNREGISTERED

untitled.html style.css

```css
li {
  display: inline;
}

```

Line 4, Column 1 Spaces: 2 CSS

Elements can be hidden from view with the visibility property.



```
* {
  margin: 0;
  padding: 0;
}

```

Line 5, Column 1; Saved ~/Desktop/style.css (UTF-8)   Spaces: 2   CSS

he visibility property can be set to one of the following values:

1. hidden – hides an element.

2. visible – displays an element.

Note: What's the difference between display: none and visibility: hidden? An element with display: none will be completely removed from the web page. An element with visibility: hidden, however, will not be visible on the web page, but the space reserved for it will.