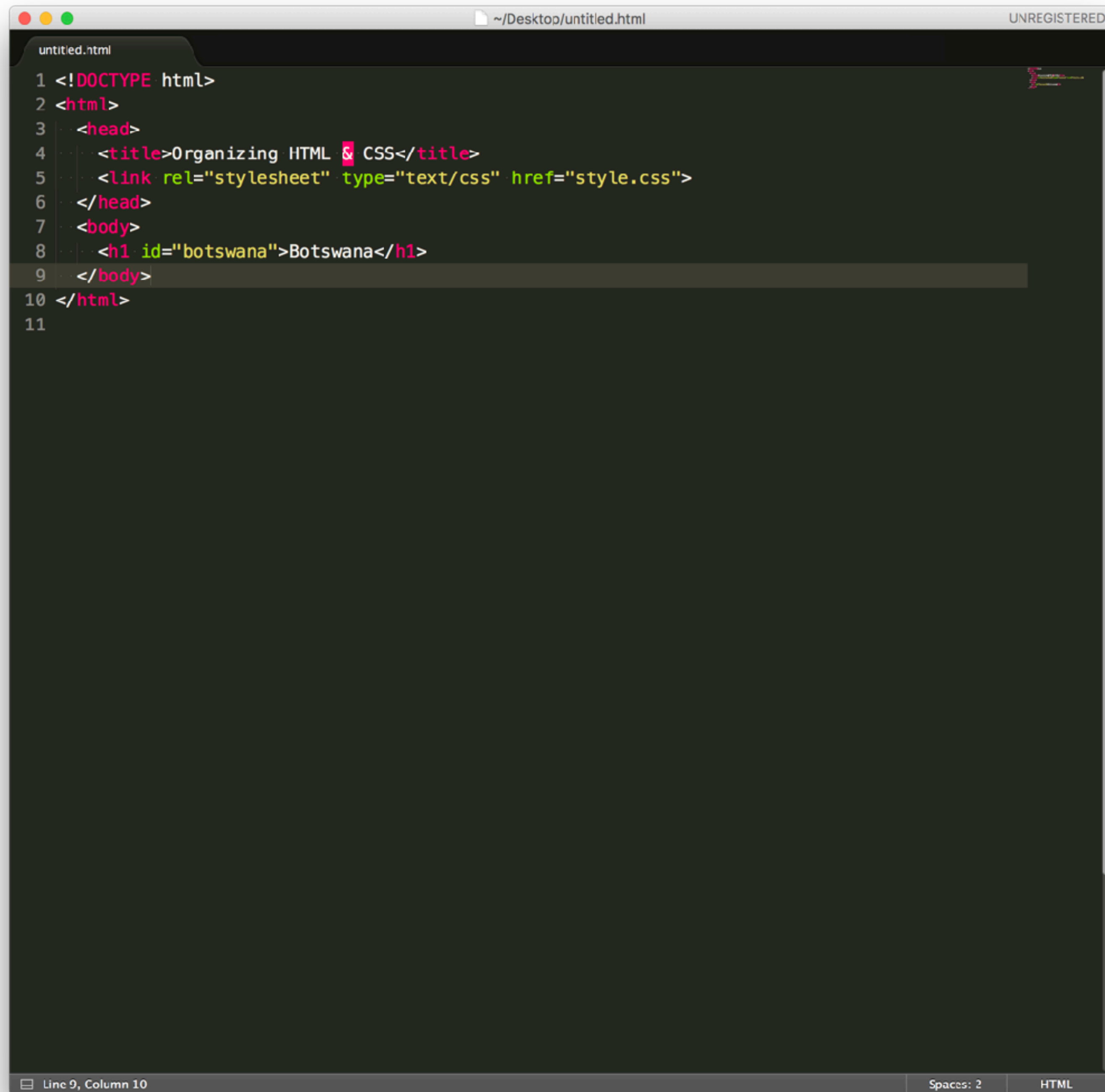


Organizing HTML & CSS

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.

A screenshot of a code editor window titled 'untitled.html' with a file path of '~/Desktop/untitled.html' and a status of 'UNREGISTERED'. The editor contains the following HTML code:

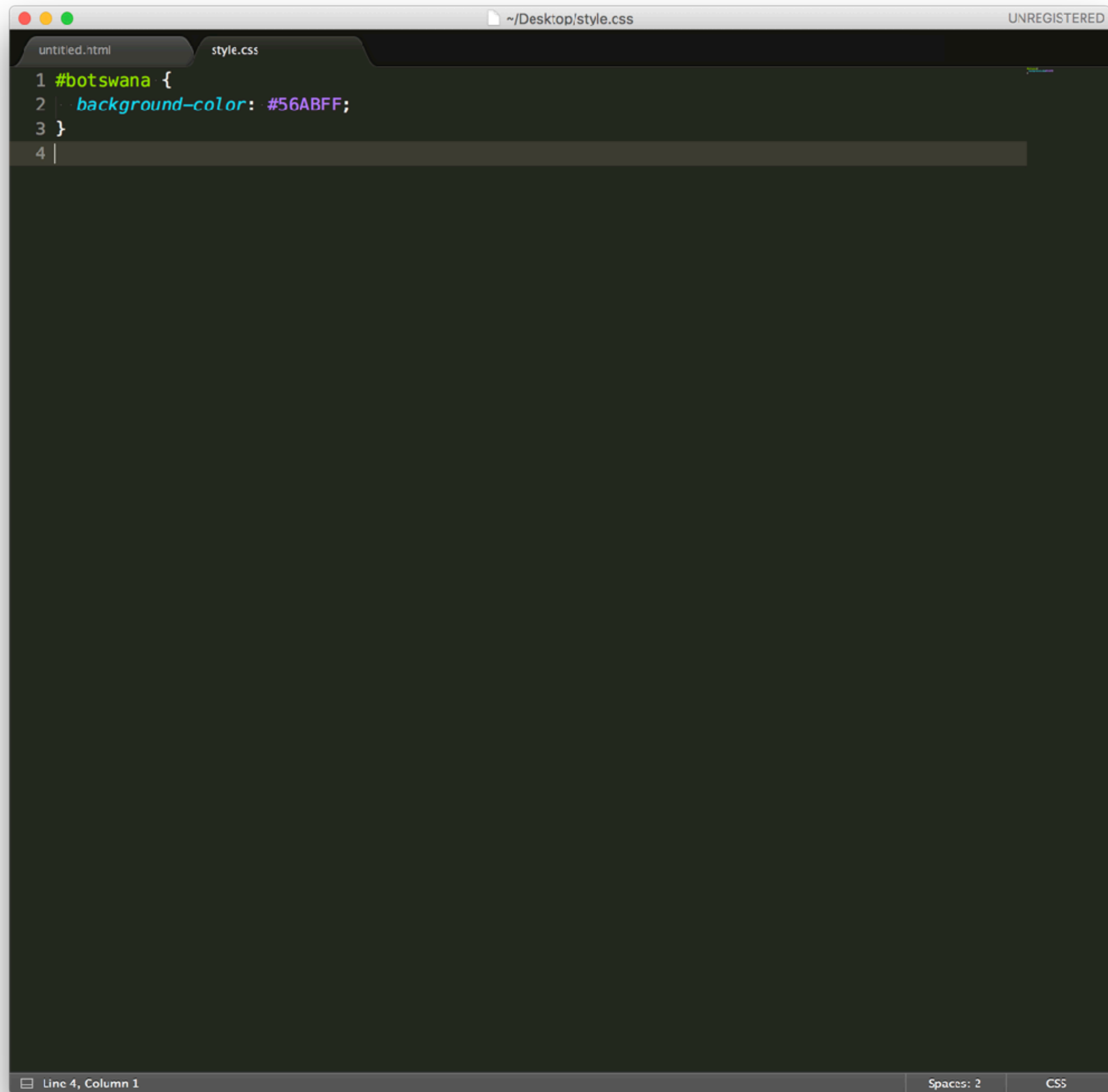
```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Organizing HTML & CSS</title>
5     <link rel="stylesheet" type="text/css" href="style.css">
6   </head>
7   <body>
8     <h1 id="botswana">Botswana</h1>
9   </body>
10 </html>
11
```

The code is syntax-highlighted. The 'id' attribute in the h1 tag is highlighted in green. The status bar at the bottom shows 'Line 9, Column 10', 'Spaces: 2', and '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.

A screenshot of a code editor window. The title bar shows the file path ~/Desktop/style.css and the status UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 #botswana {  
2   background-color: #56ABFF;  
3 }  
4 |
```

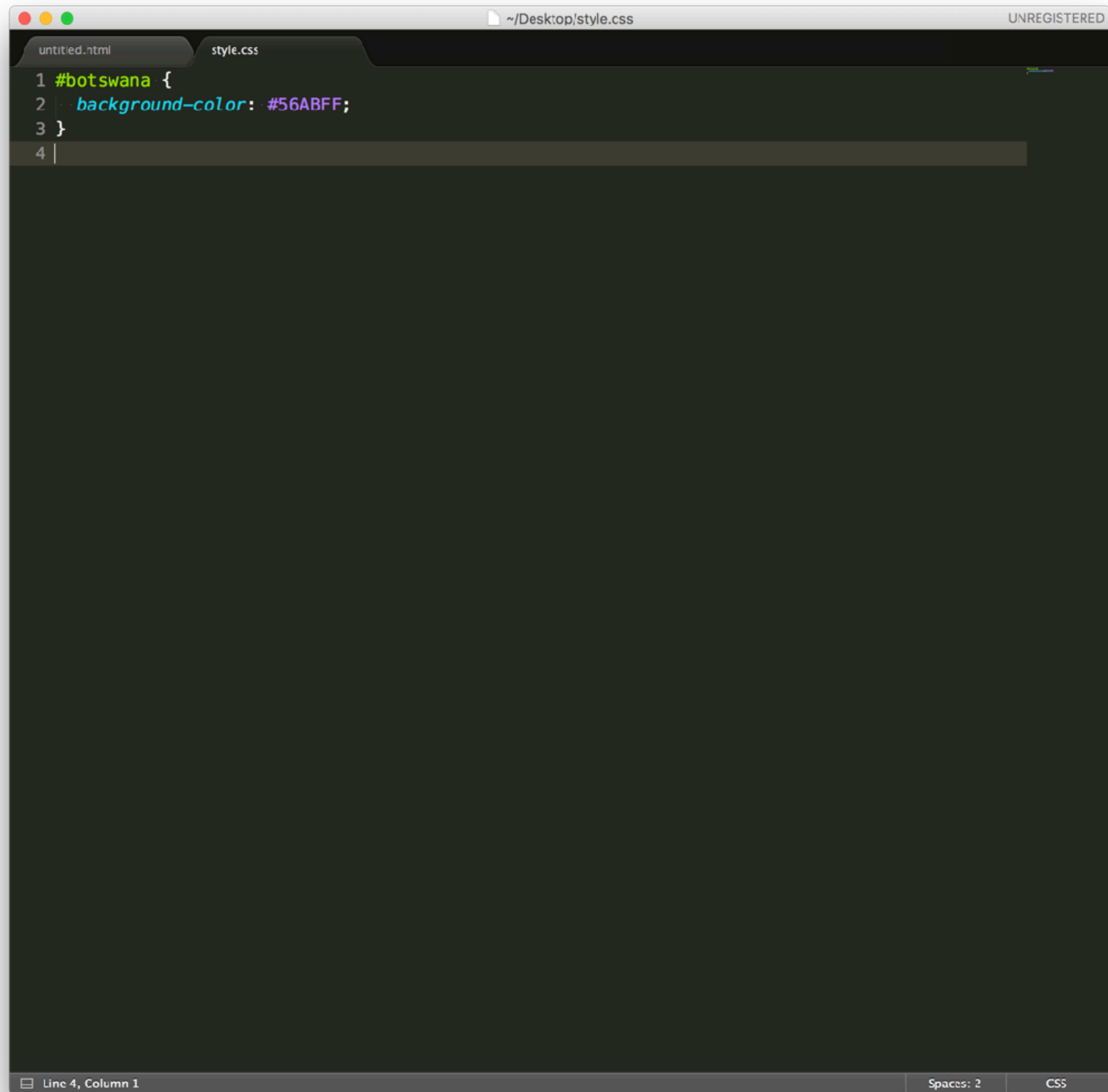
The code is syntax-highlighted: the ID selector '#botswana' is green, the property 'background-color' is blue, and the value '#56ABFF' is purple. The cursor is at the end of line 4, column 1. The status bar at the bottom shows 'Line 4, Column 1', 'Spaces: 2', and 'CSS'.

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.

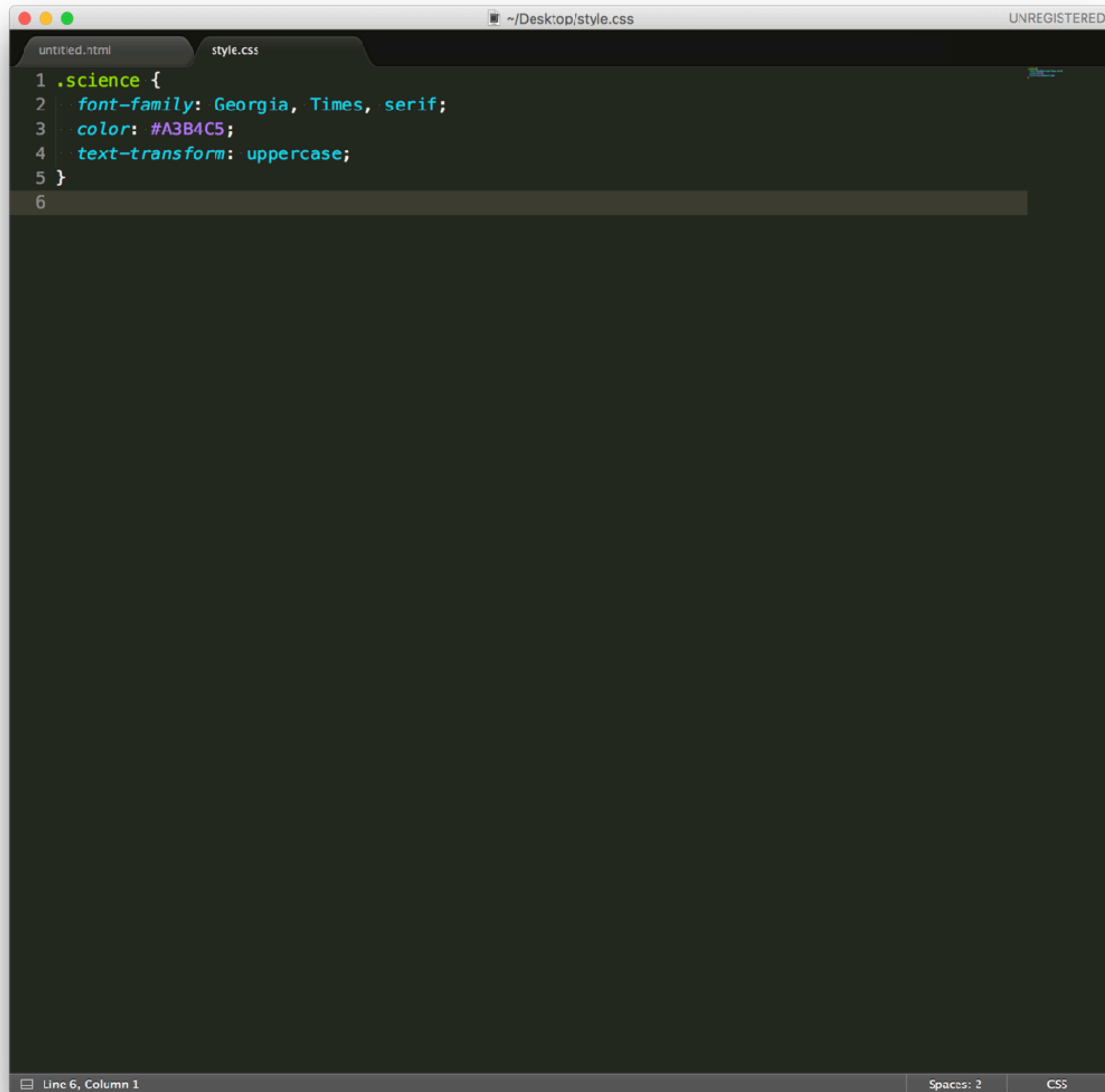
A screenshot of a code editor window. The title bar at the top shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. Below the title bar, there are two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 #botswana {  
2   background-color: #56ABFF;  
3 }  
4 |
```

The code is syntax-highlighted: '#' is green, 'botswana' is yellow, '{' is blue, 'background-color' is blue, ':' is blue, '#56ABFF' is purple, and ';' is blue. The cursor is at the end of line 4, column 1. The status bar at the bottom shows 'Line 4, Column 1', 'Spaces: 2', and 'CSS'.

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?



The image shows a code editor window with a dark theme. The title bar at the top indicates the file path is `~/Desktop/style.css` and the application is `UNREGISTERED`. There are two tabs: `untitled.html` and `style.css`. The `style.css` tab is active, showing the following CSS code:

```
1 .science {  
2   font-family: Georgia, Times, serif;  
3   color: #A3B4C5;  
4   text-transform: uppercase;  
5 }  
6
```

The code is syntax-highlighted: the class selector `.science` is green, the opening curly brace is blue, the property names are blue, the values are purple, and the closing curly brace is blue. The line numbers 1 through 6 are visible on the left side of the editor.

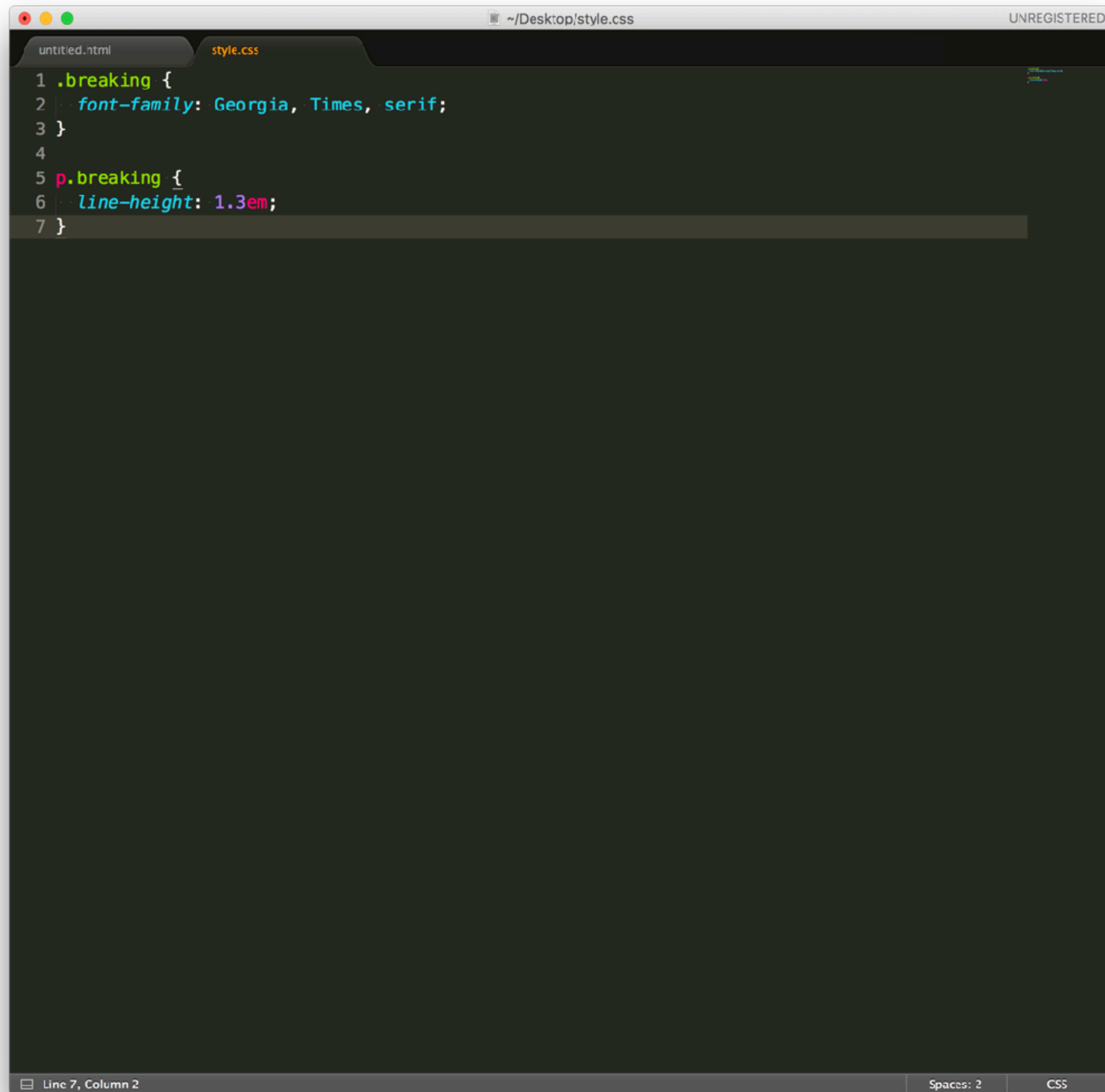
At the bottom of the editor, there is a status bar showing `Line 6, Column 1`, `Spaces: 2`, and `CSS`.

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.

A screenshot of a code editor window. The title bar shows the file path as ~/Desktop/style.css and the status as UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

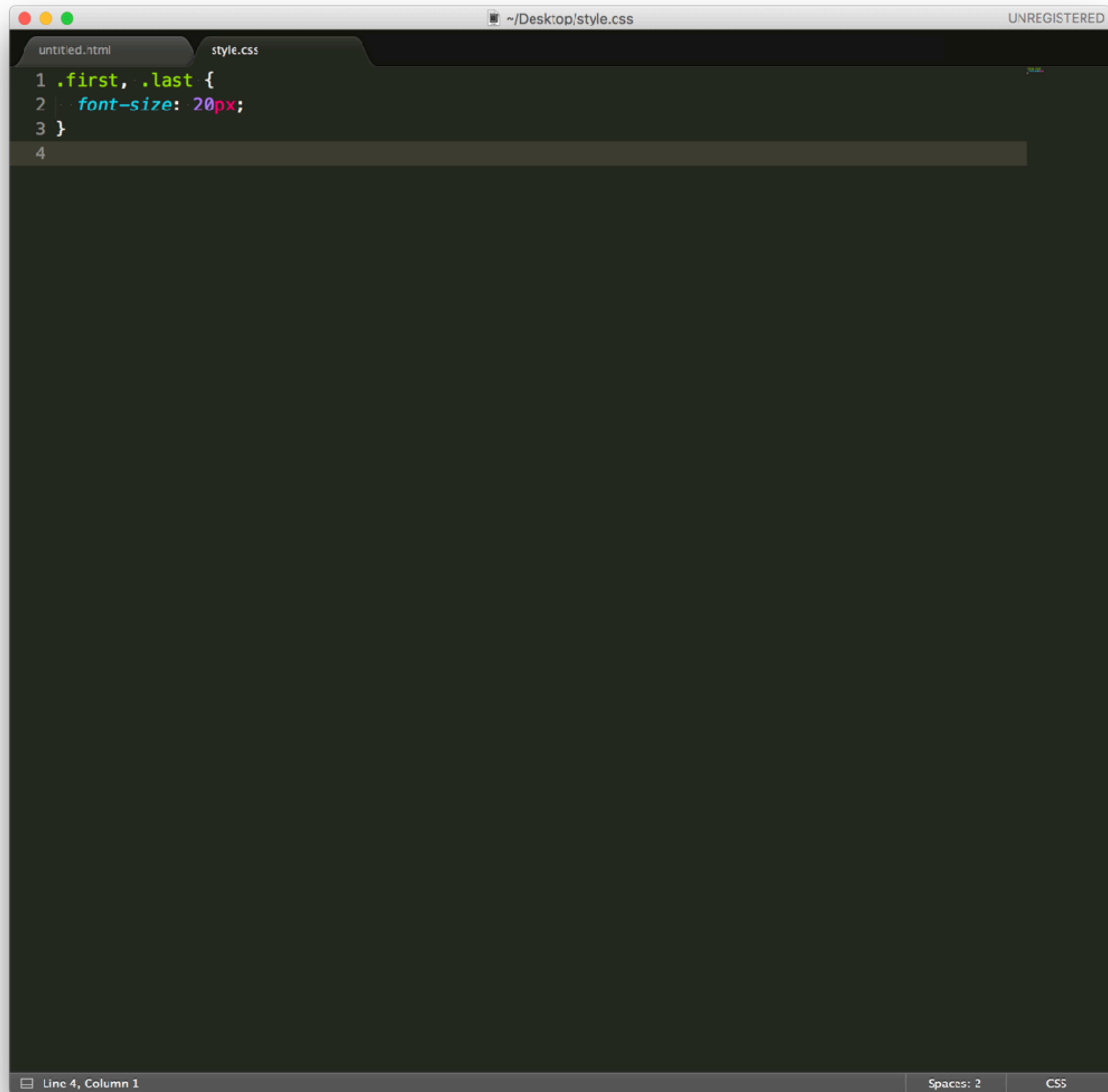
```
1 .breaking {  
2   font-family: Georgia, Times, serif;  
3 }  
4  
5 p.breaking {  
6   line-height: 1.3em;  
7 }
```

The code is syntax-highlighted. The status bar at the bottom indicates 'Line 7, Column 2', 'Spaces: 2', and 'CSS'.

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...



The image shows a code editor window with a dark theme. The title bar at the top indicates the file path is `~/Desktop/style.css` and the application is `UNREGISTERED`. The editor has two tabs: `untitled.html` and `style.css`. The `style.css` tab is active, showing the following CSS code:

```
1 .first, .last {  
2   font-size: 20px;  
3 }  
4
```

The code uses a multiple class selector `.first, .last` to apply the `font-size: 20px` rule to both `.first` and `.last` classes. The editor's status bar at the bottom shows `Line 4, Column 1`, `Spaces: 2`, and `CSS`.

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.

~/Desktop/untitled.htmlUNREGISTERED

untitled.htmlstyle.css

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Organizing HTML & CSS</title>
5     <link rel="stylesheet" type="text/css" href="style.css">
6   </head>
7   <body>
8     <h1 class="book domestic">The Way of the Deep</h1>
9     <h1 class="book foreign">A Night in the Sky</h1>
10  </body>
11 </html>
12
```

Line 12, Column 1Spaces: 2HTML

~/Desktop/style.cssUNREGISTERED

untitled.htmlstyle.css

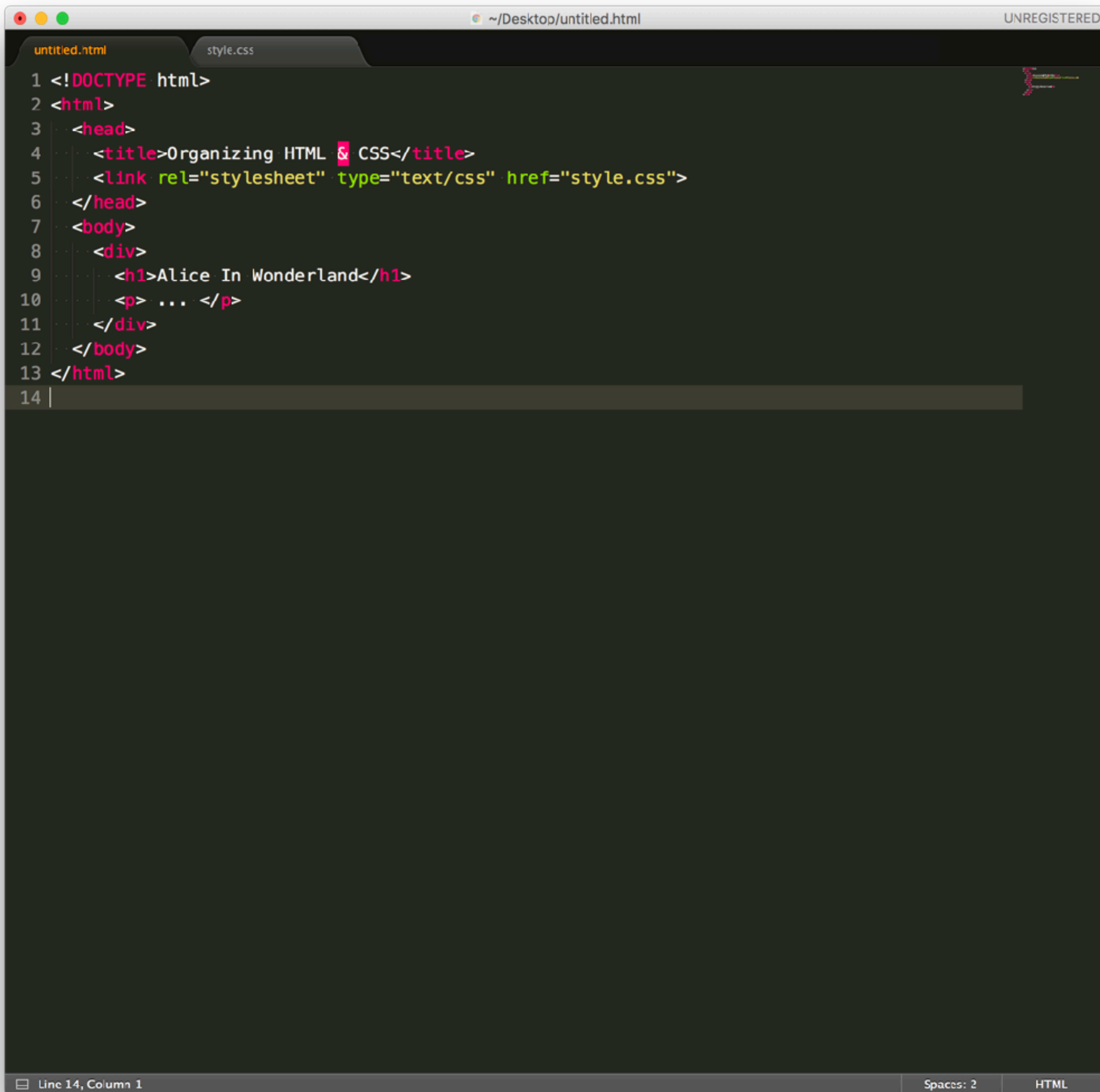
```
1 .book {
2   font-family: Georgia, serif;
3 }
4
5 .domestic {
6   font-color: #0902CC;
7 }
8
9 .foreign {
10  font-color: #B097DD;
11 }
12
```

Line 12, Column 1Spaces: 2CSS

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.



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Organizing HTML & CSS</title>
5     <link rel="stylesheet" type="text/css" href="style.css">
6   </head>
7   <body>
8     <div>
9       <h1>Alice In Wonderland</h1>
10      <p>... </p>
11    </div>
12  </body>
13 </html>
14 |
```

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.

```
untitled.html style.css
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Organizing HTML & CSS</title>
5   <link rel="stylesheet" type="text/css" href="style.css">
6 </head>
7 <body>
8   <div class="container">
9     <h1 class="title">Alice In Wonderland</h1>
10    <p> ... </p>
11  </div>
12 </body>
13 </html>
14
```

Line 11, Column 11 Spaces: 2 HTML

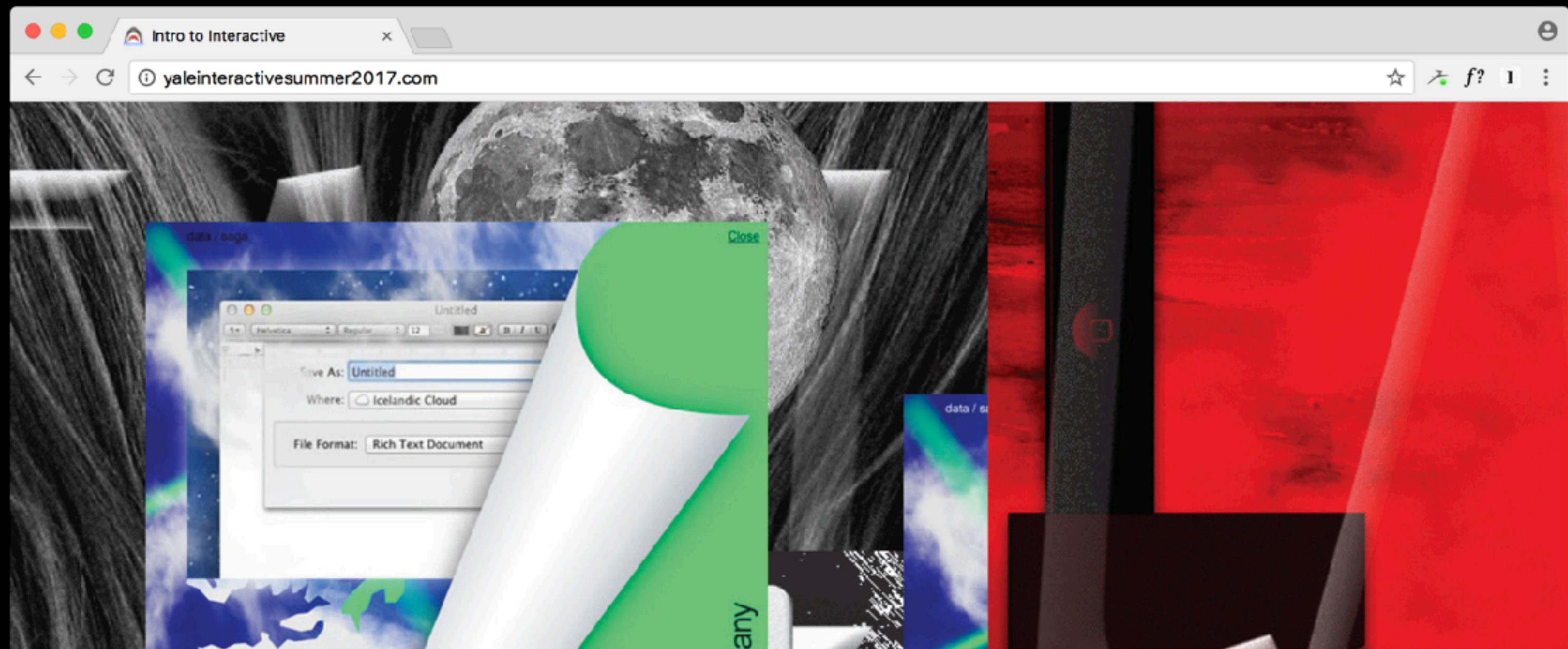
```
untitled.html style.css
1 div.container {
2   background-color: rgb(252, 255, 205);
3   font-family: Roboto, Helvetica, sans-serif;
4 }
5
6 h1.title {
7   color: #0D1A2F;
8 }
9
```

Line 9, Column 1 Spaces: 2 CSS

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](#)

[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

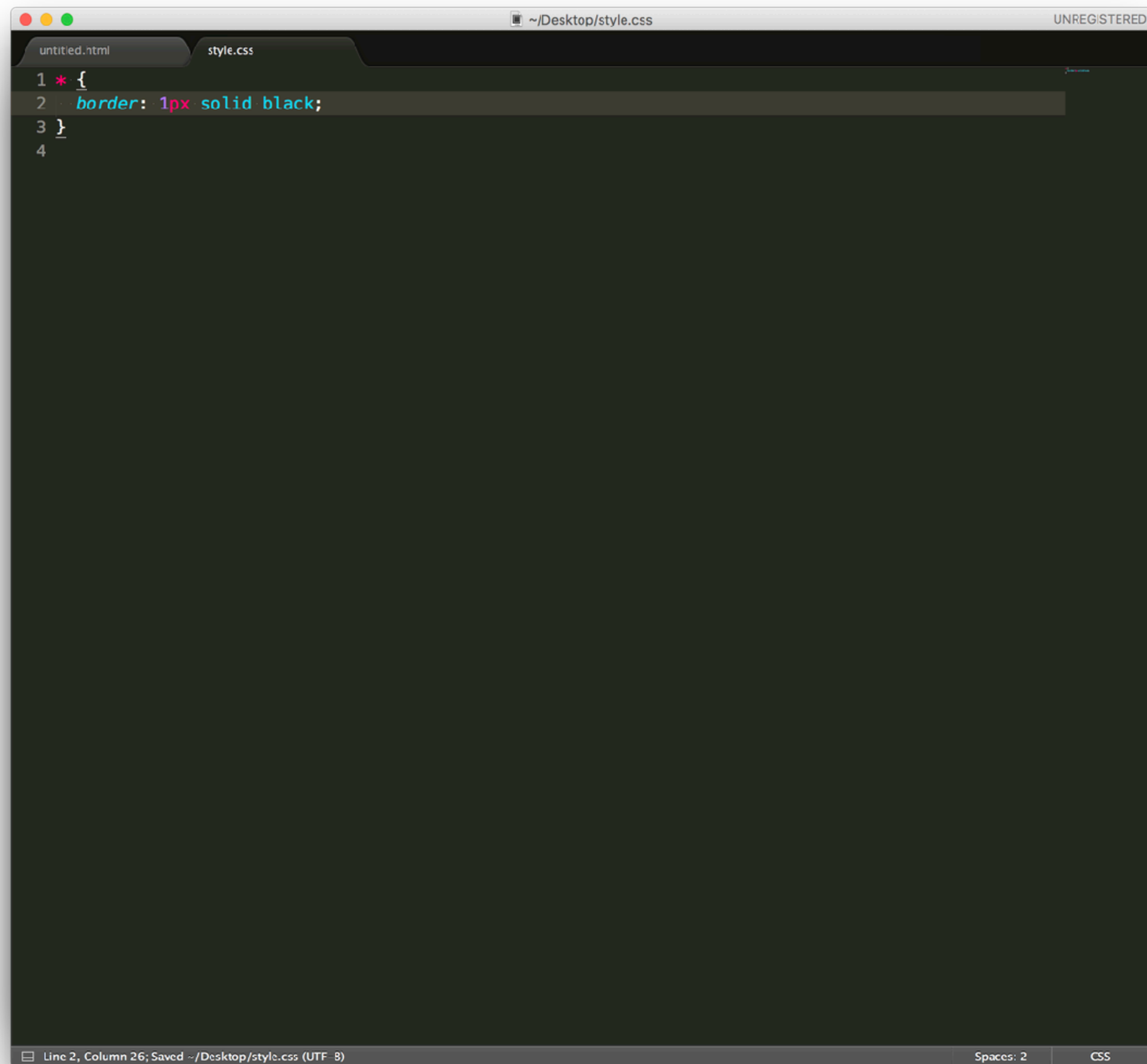
The Box Model

(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!

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, a file path icon and text '~ / Desktop / style.css' in the center, and the word 'UNREGISTERED' on the right. Below the title bar, there are two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active. The code editor area has a dark background. The first three lines of code are visible: '1 * {', '2 border: 1px solid black;', and '3 }'. Line numbers 1, 2, 3, and 4 are shown on the left side of the editor. The status bar at the bottom shows 'Line 2, Column 26; Saved -- / Desktop / style.css (UTF-8)' on the left, 'Spaces: 2' in the center, and 'CSS' on the right.

```
1 * {  
2 border: 1px solid black;  
3 }  
4
```

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

data / design

Code

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

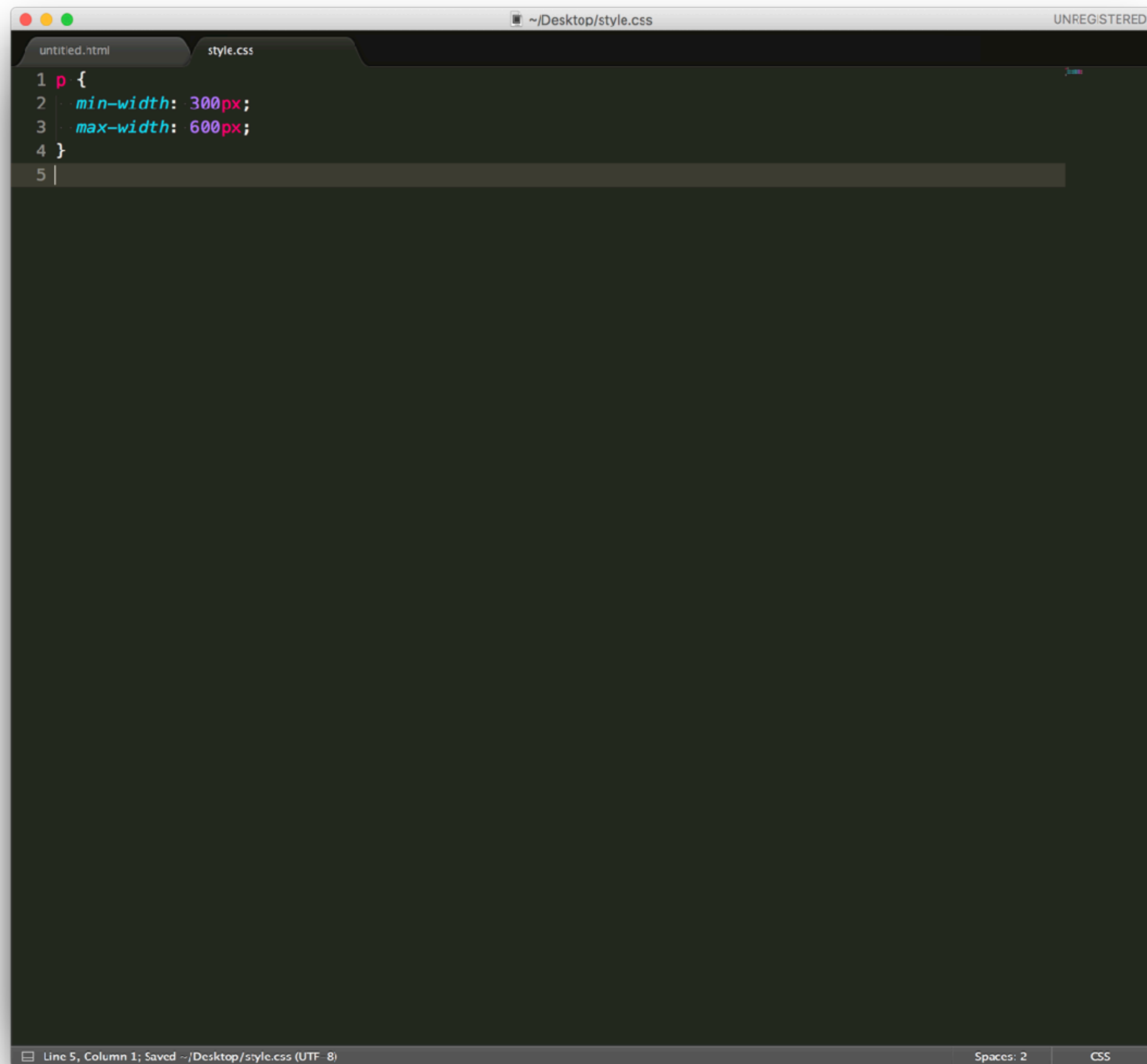
CSS, Positioning, Box Model

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.

A screenshot of a code editor window. The title bar shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

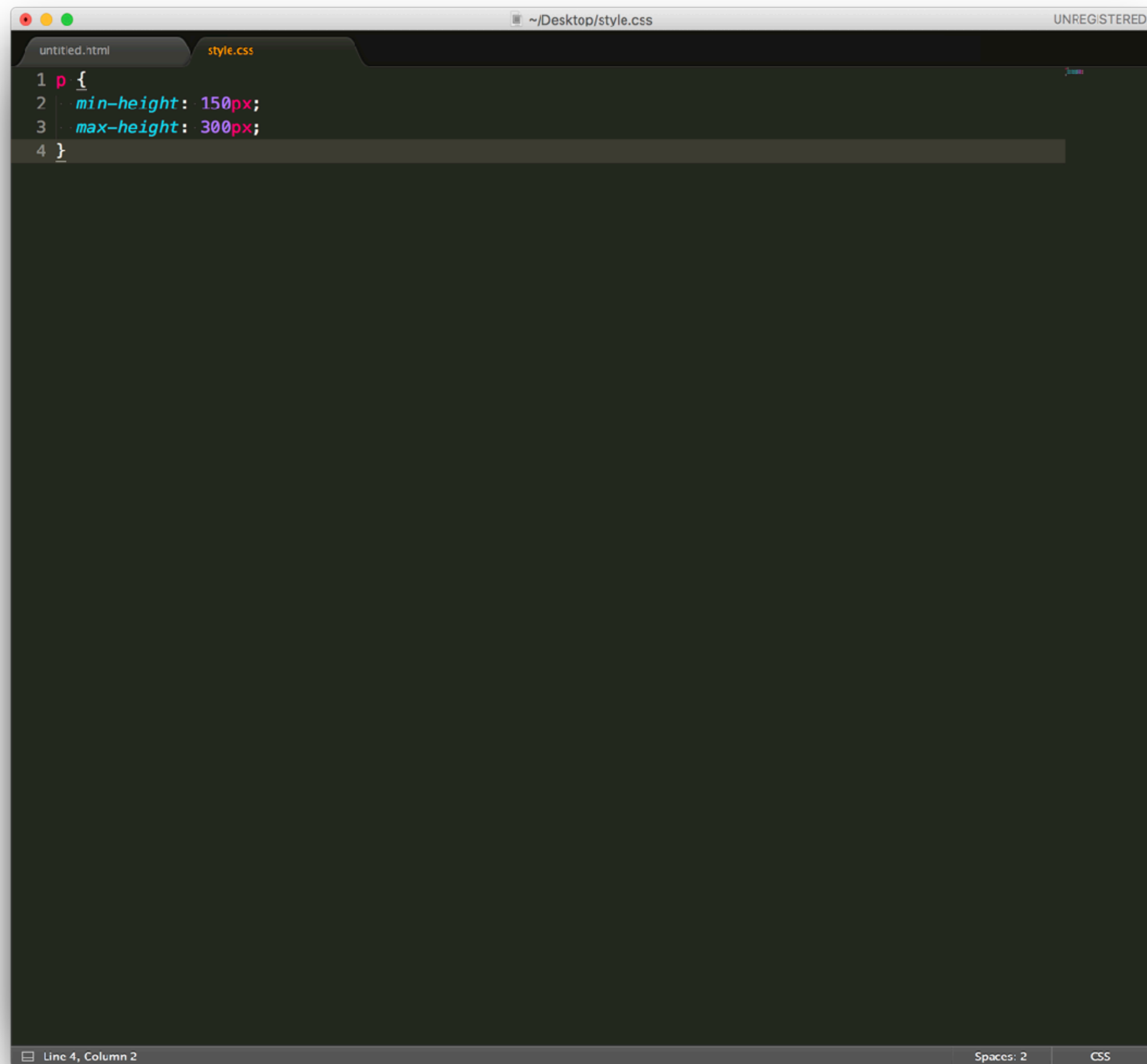
```
1 p {  
2   min-width: 300px;  
3   max-width: 600px;  
4 }  
5 |
```

The code is syntax-highlighted: 'p' is blue, '{' is red, 'min-width' is green, '300px' is blue, 'max-width' is green, '600px' is blue, and '}' is red. The status bar at the bottom shows 'Line 5, Column 1; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

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.

A screenshot of a code editor window. The title bar shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 p {  
2   min-height: 150px;  
3   max-height: 300px;  
4 }
```

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

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

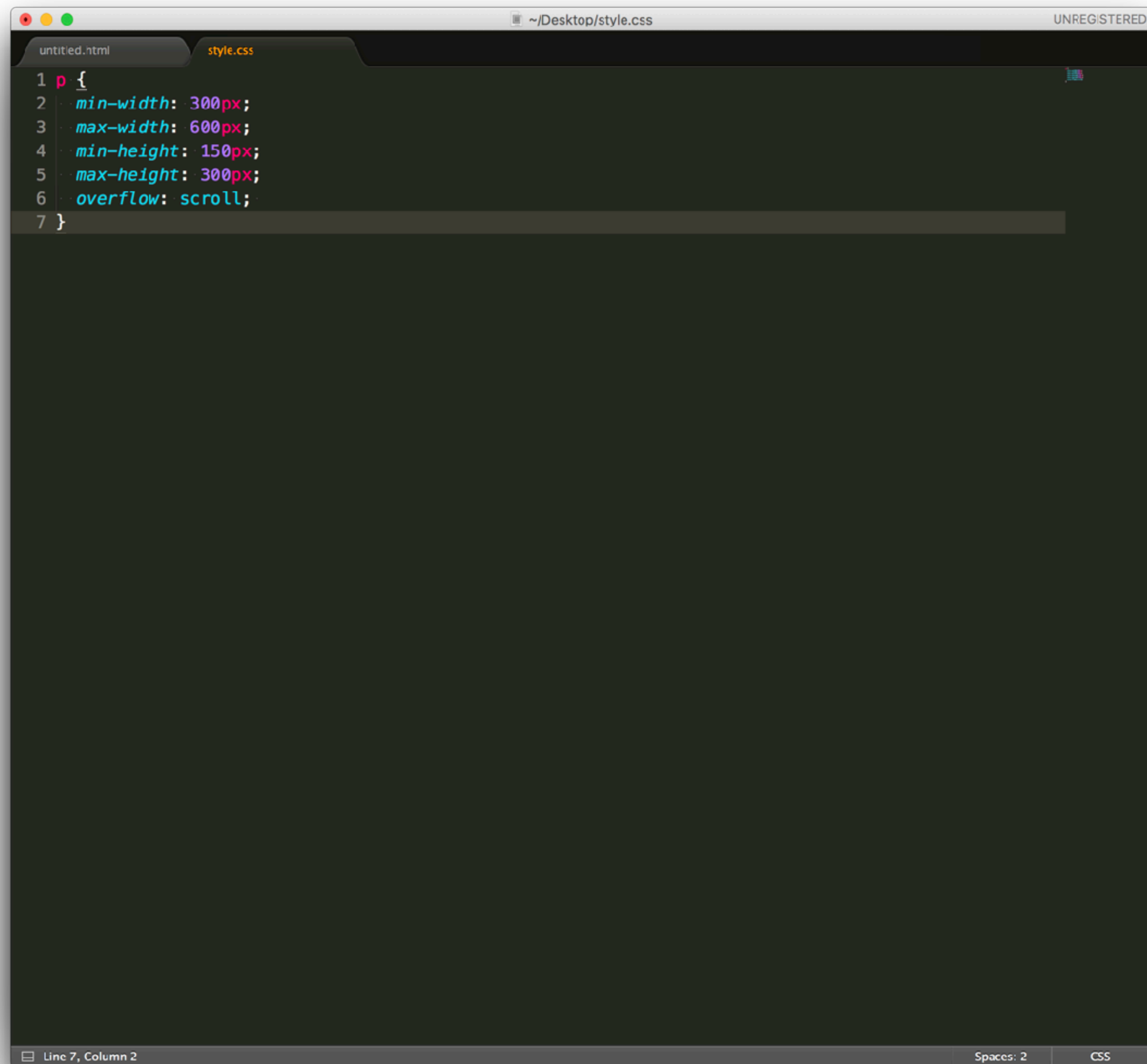
min-height – this property ensures a minimum height for an element's box.

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.

A screenshot of a code editor window. The title bar shows a file named "style.css" located at "~/Desktop/style.css" and indicates it is "UNREGISTERED". The editor has two tabs: "untitled.html" and "style.css". The "style.css" tab is active, showing a CSS rule for a paragraph element. The code is as follows:

```
1 p {  
2   min-width: 300px;  
3   max-width: 600px;  
4   min-height: 150px;  
5   max-height: 300px;  
6   overflow: scroll;  
7 }
```

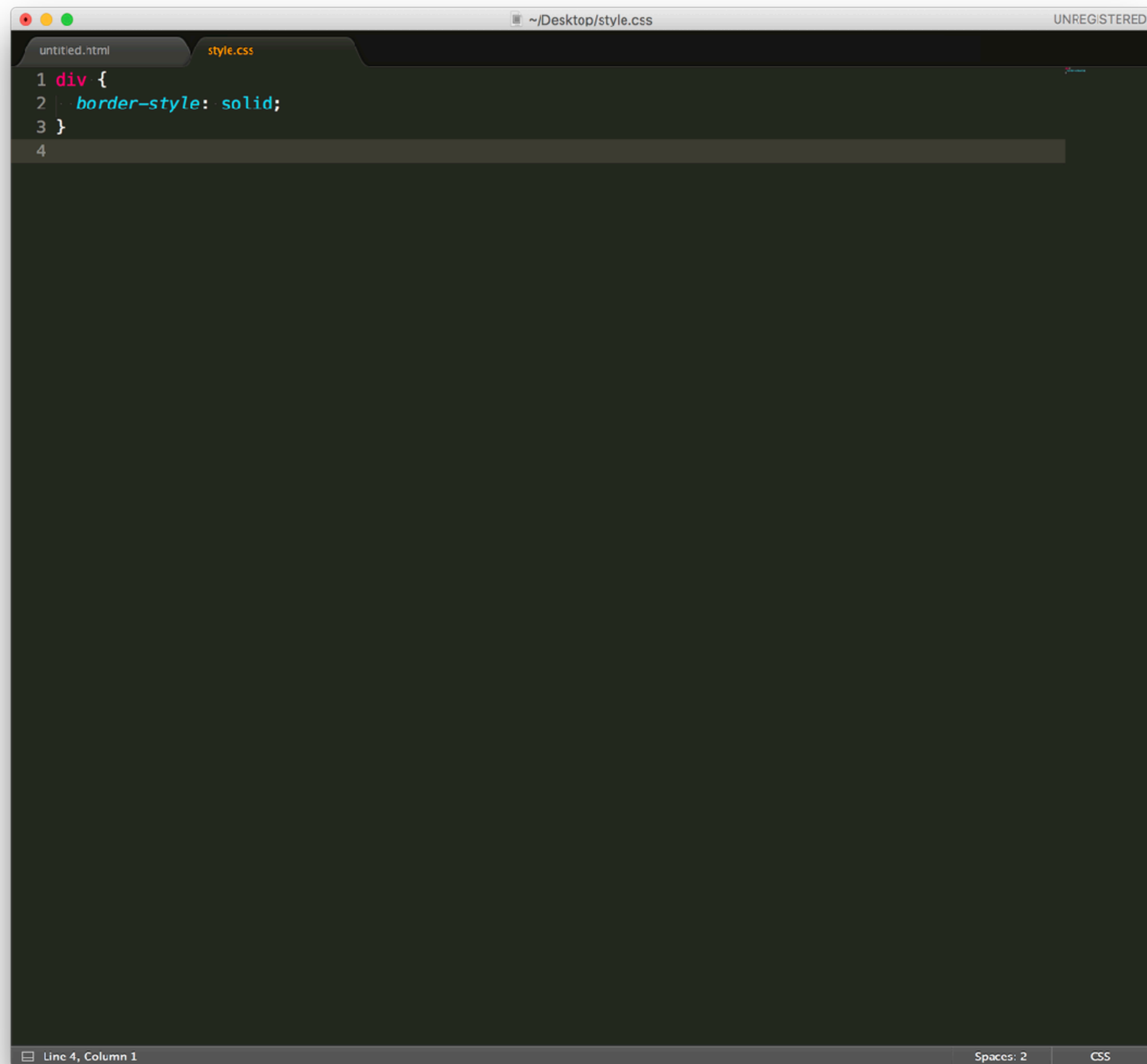
The status bar at the bottom indicates "Line 7, Column 2", "Spaces: 2", and "CSS".

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.

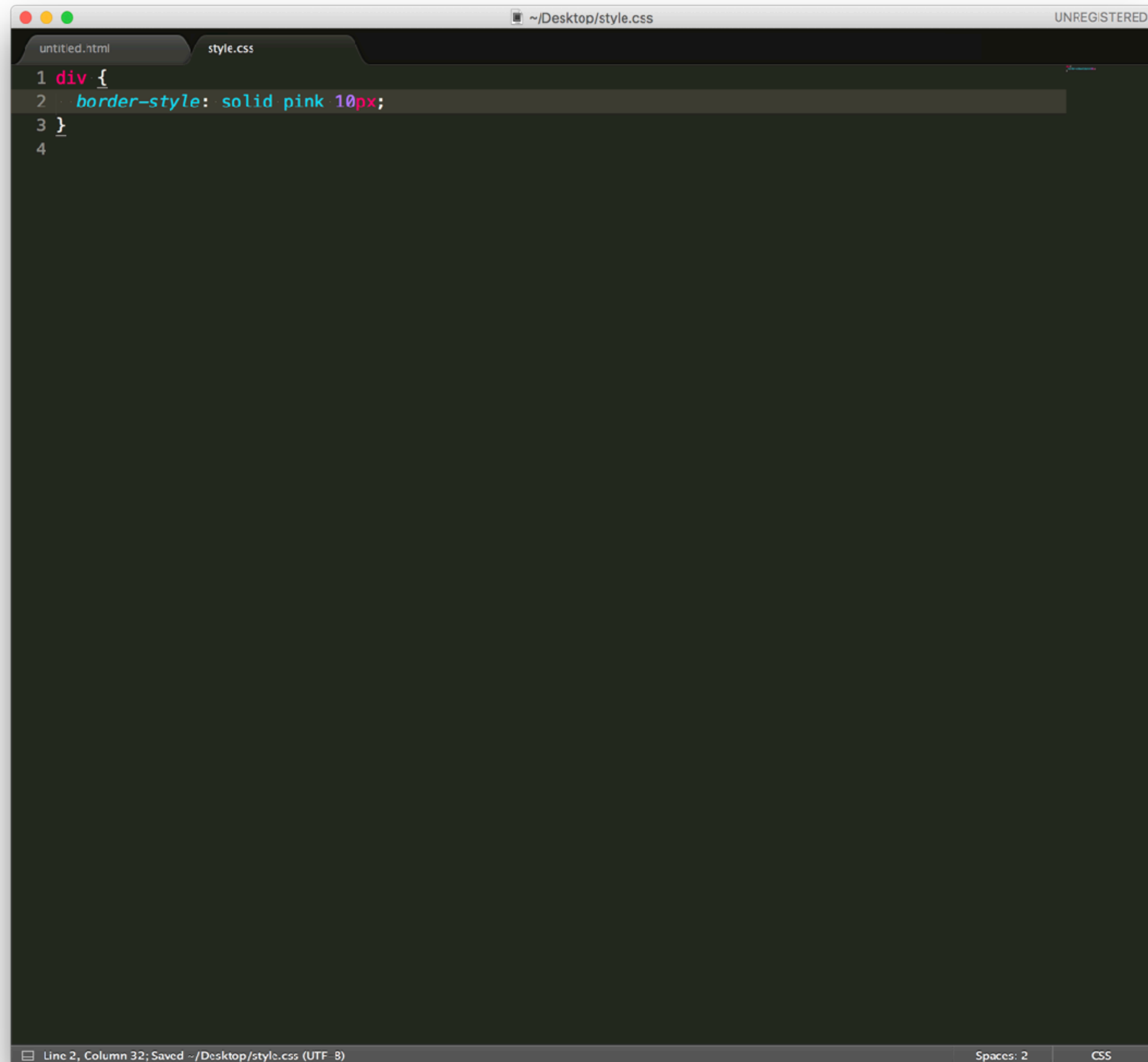
A screenshot of a code editor window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left, a file path icon and text '~ / Desktop / style.css' in the center, and the word 'UNREGISTERED' on the right. Below the title bar, there are two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active. The code editor area has a dark background and shows the following CSS code:

```
1 div {  
2   border-style: solid;  
3 }  
4
```

The code is syntax-highlighted: 'div' is pink, '{' is light blue, 'border-style:' is light blue, 'solid;' is light blue, and '}' is light blue. A vertical line of numbers 1, 2, 3, and 4 is on the left side of the code. At the bottom of the editor, there is a status bar with three sections: 'Line 4, Column 1' on the left, 'Spaces: 2' in the middle, and 'CSS' on the right.

This property can take on one of the following values:

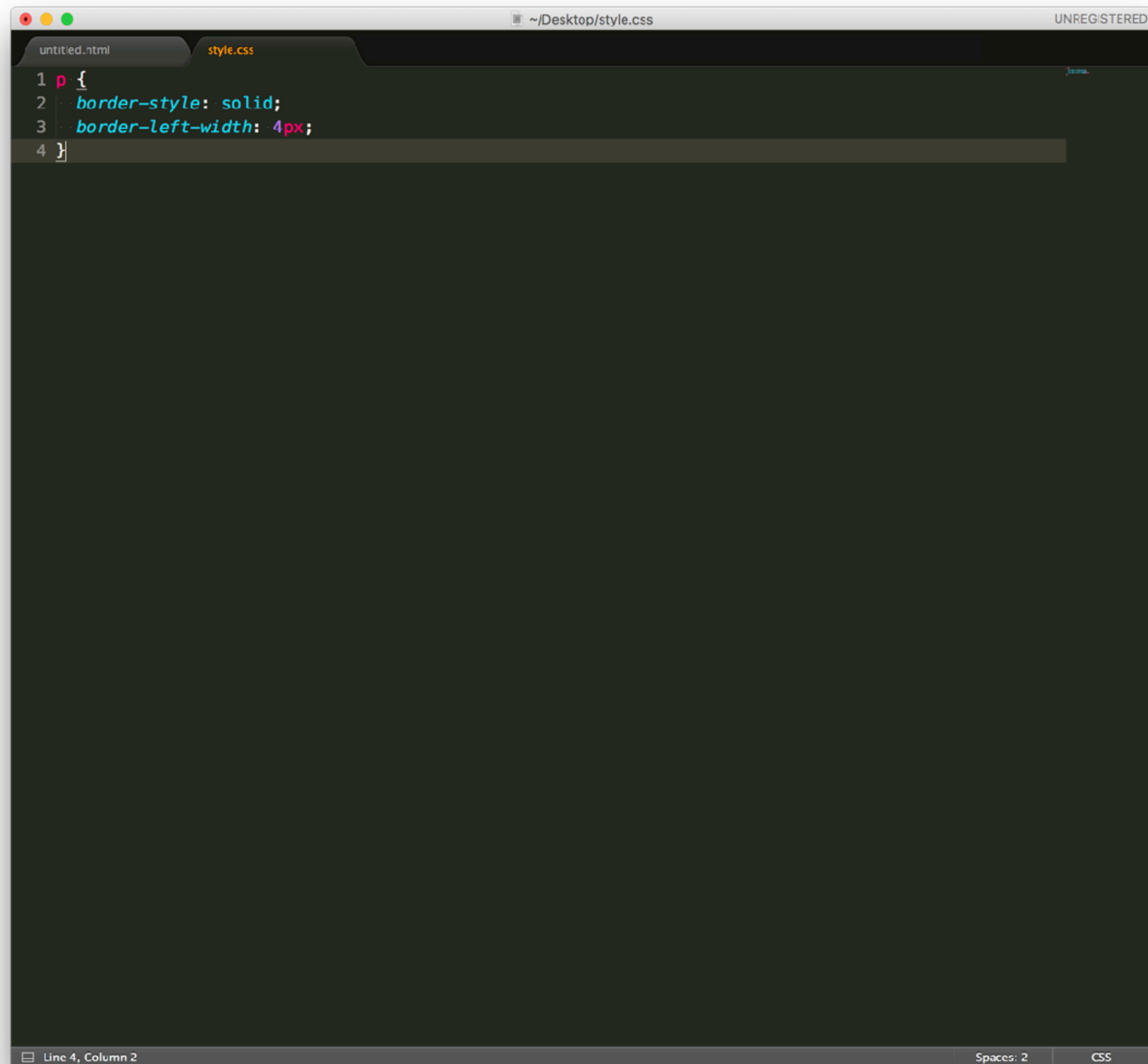
solid, dashed, dotted, double, groove, inset, outset, ridge, hidden or none.

A screenshot of a code editor window. The title bar at the top shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. Below the title bar, there are two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing a dark-themed code editor with the following CSS code:

```
1 div {  
2   border-style: solid pink 10px;  
3 }  
4
```

The code is syntax-highlighted: 'div' is pink, '{' is blue, 'border-style:' is light blue, 'solid' is light blue, 'pink' is pink, '10px;' is pink, and '}' is blue. The status bar at the bottom shows 'Line 2, Column 32; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

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

A screenshot of a code editor window. The title bar shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing four lines of CSS code: 1 p {, 2 border-style: solid;, 3 border-left-width: 4px;, 4 }. The code is syntax-highlighted. The status bar at the bottom shows 'Line 4, Column 2', 'Spaces: 2', and 'CSS'.

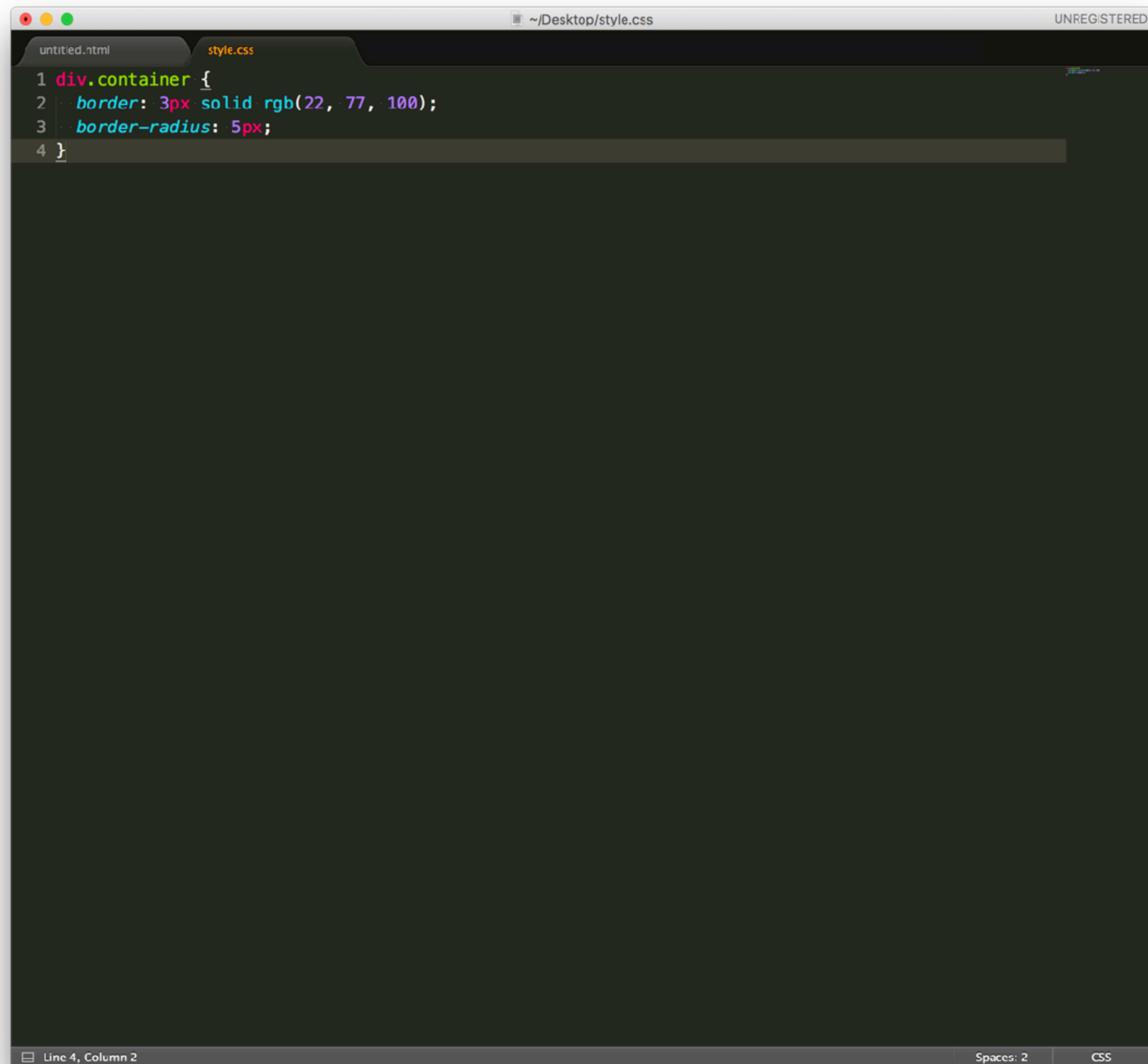
```
1 p {  
2   border-style: solid;  
3   border-left-width: 4px;  
4 }
```

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.

A screenshot of a code editor window. The title bar shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 div.container {  
2   border: 3px solid rgb(22, 77, 100);  
3   border-radius: 5px;  
4 }
```

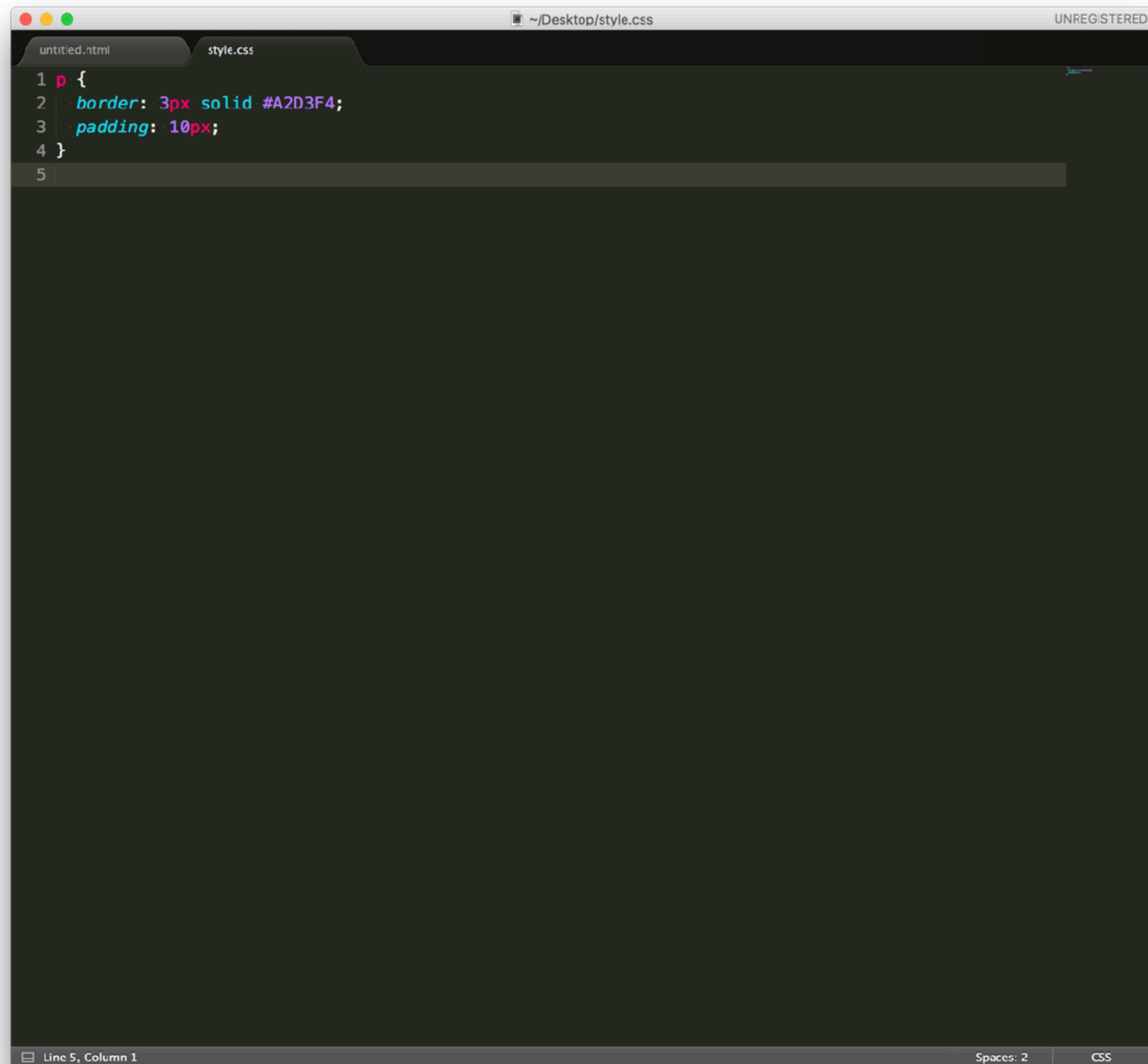
The code is syntax-highlighted. The status bar at the bottom shows 'Line 4, Column 2', 'Spaces: 2', and 'CSS'.

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.

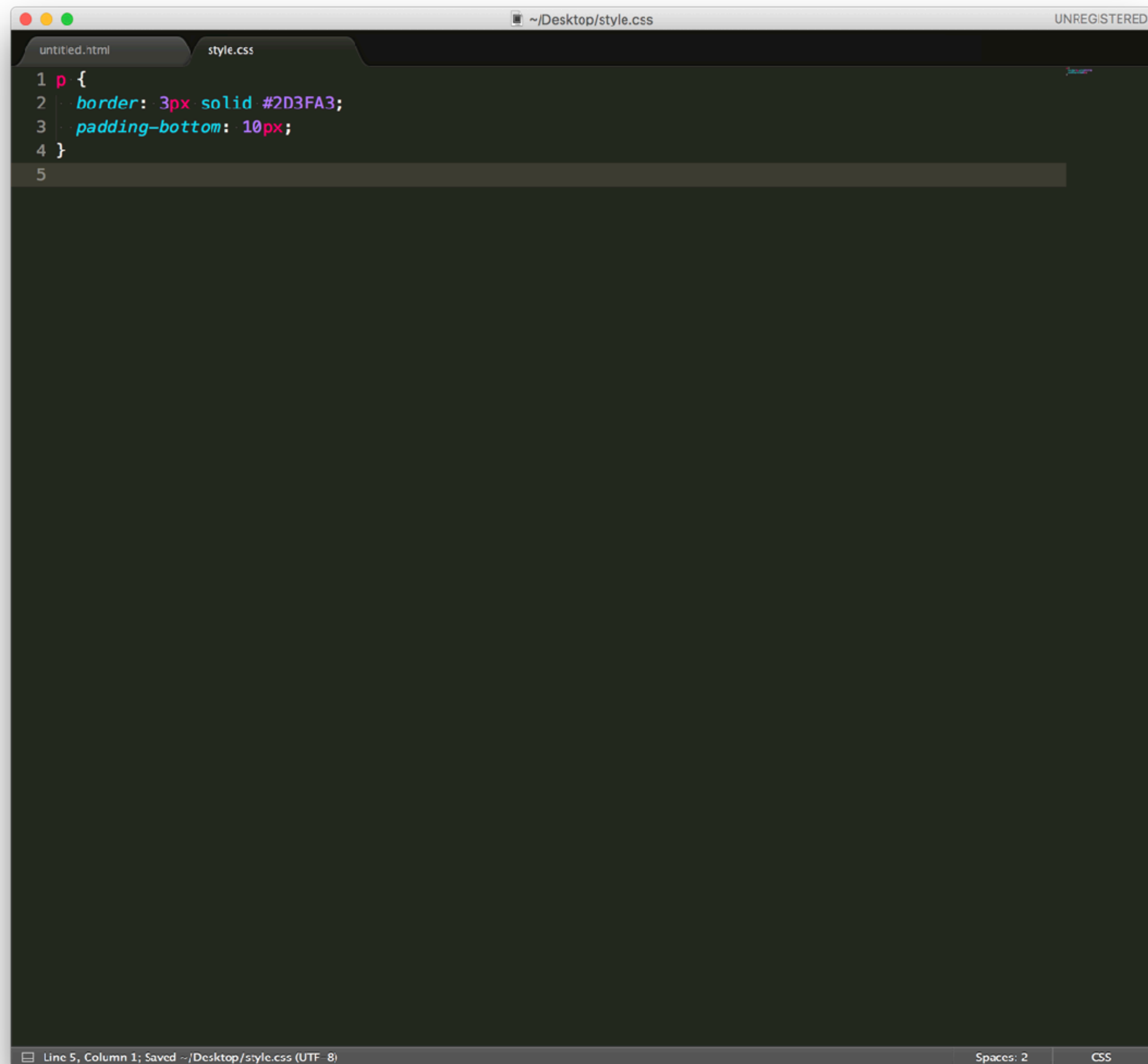
The space between the contents of a box and the borders of a box is known as *padding*.

A screenshot of a code editor window. The title bar at the top shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. Below the title bar, there are two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 p {  
2   border: 3px solid #A2D3F4;  
3   padding: 10px;  
4 }  
5
```

The code is syntax-highlighted: 'p' is blue, '{' is blue, 'border' is blue, '3px' is red, 'solid' is blue, '#A2D3F4' is purple, 'padding' is blue, '10px' is red, and '}' is blue. The line numbers 1 through 5 are on the left. The bottom status bar shows 'Line 5, Column 1', 'Spaces: 2', and 'CSS'.

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.

A screenshot of a code editor window. The title bar shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 p {  
2   border: 3px solid #2D3FA3;  
3   padding-bottom: 10px;  
4 }  
5
```

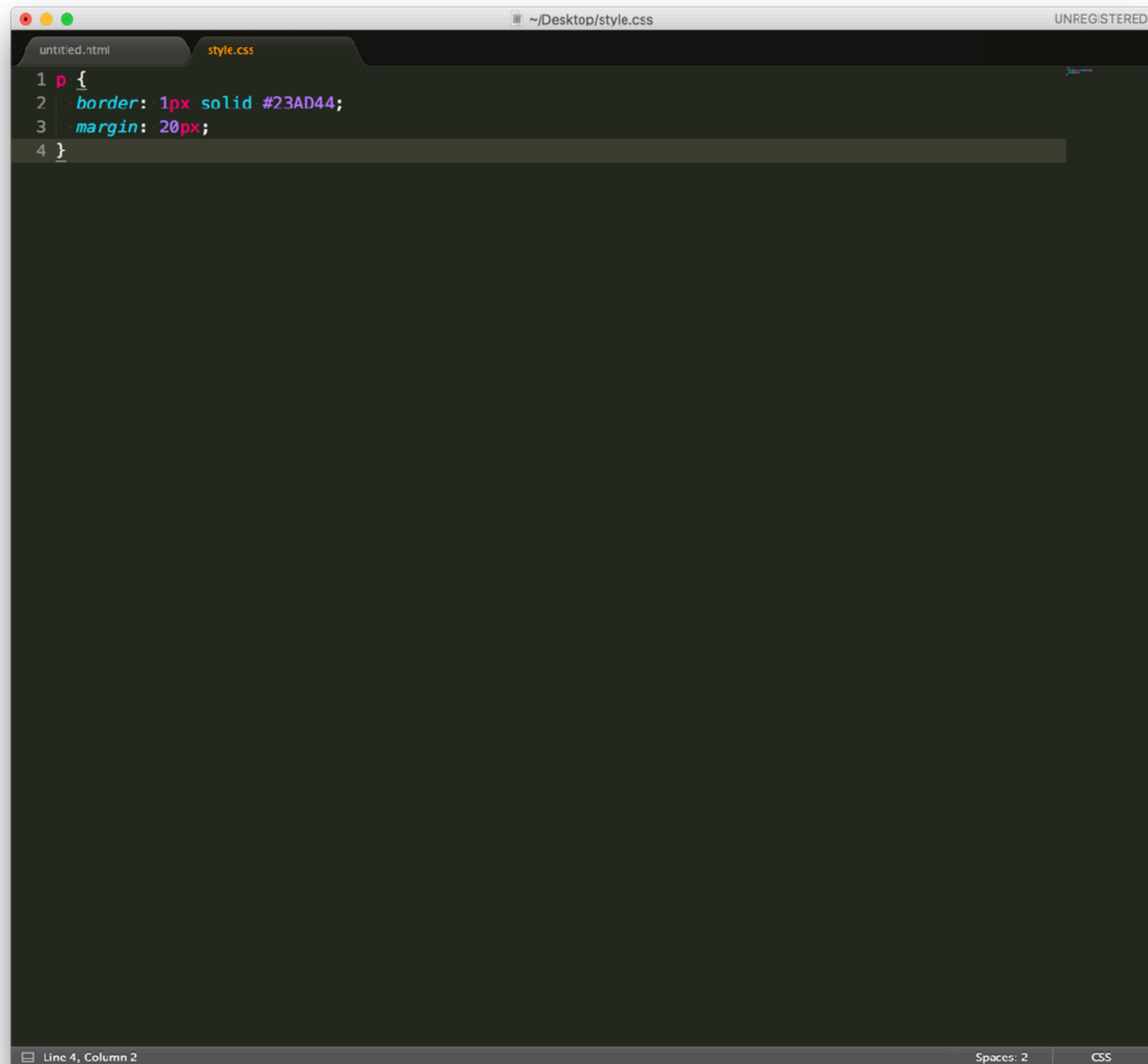
The code is syntax-highlighted. The status bar at the bottom shows 'Line 5, Column 1; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

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*.

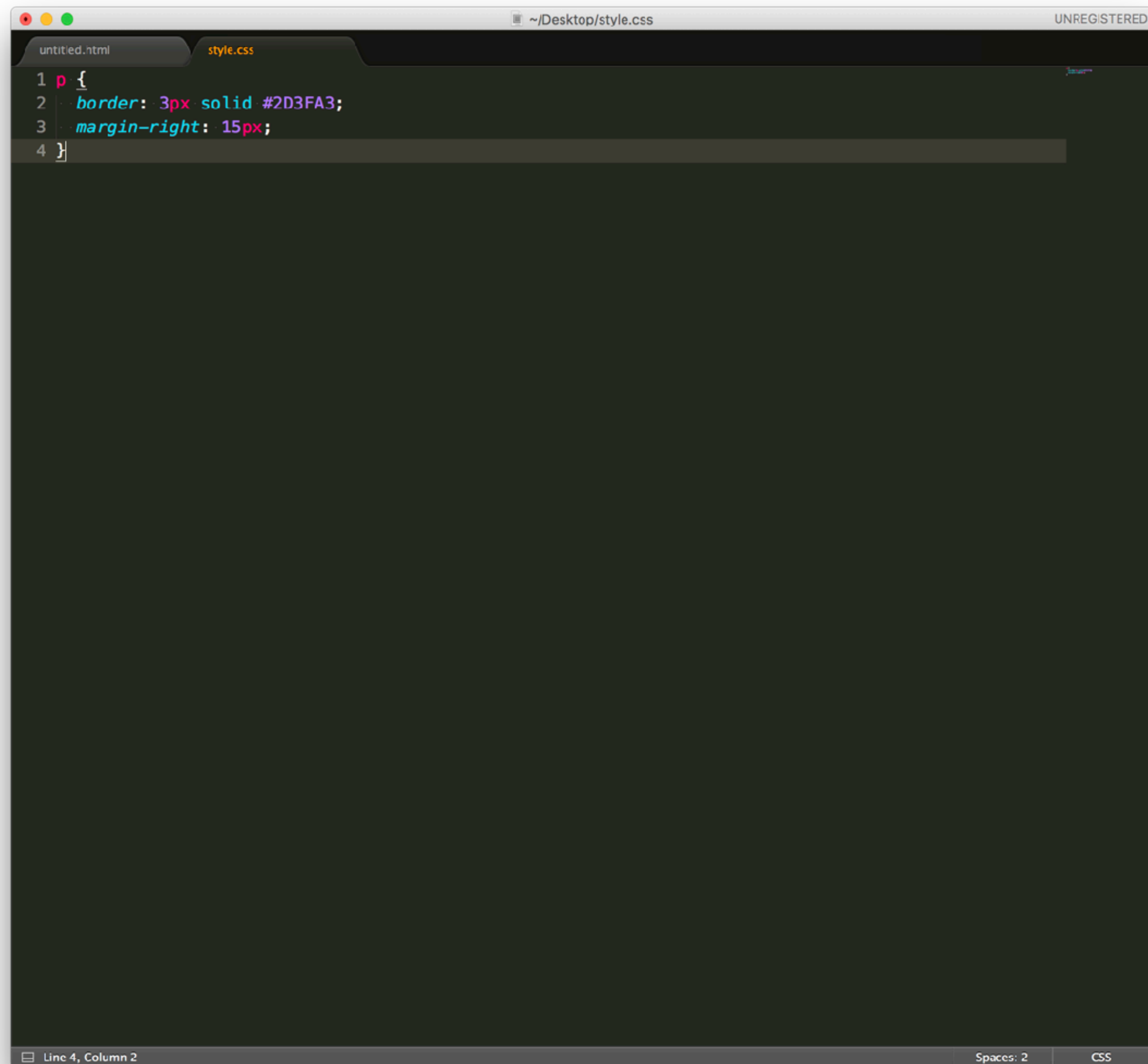
The margin refers to the space directly outside of the box.

A screenshot of a code editor window. The title bar shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

```
1 p {  
2   border: 1px solid #23AD44;  
3   margin: 20px;  
4 }
```

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

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

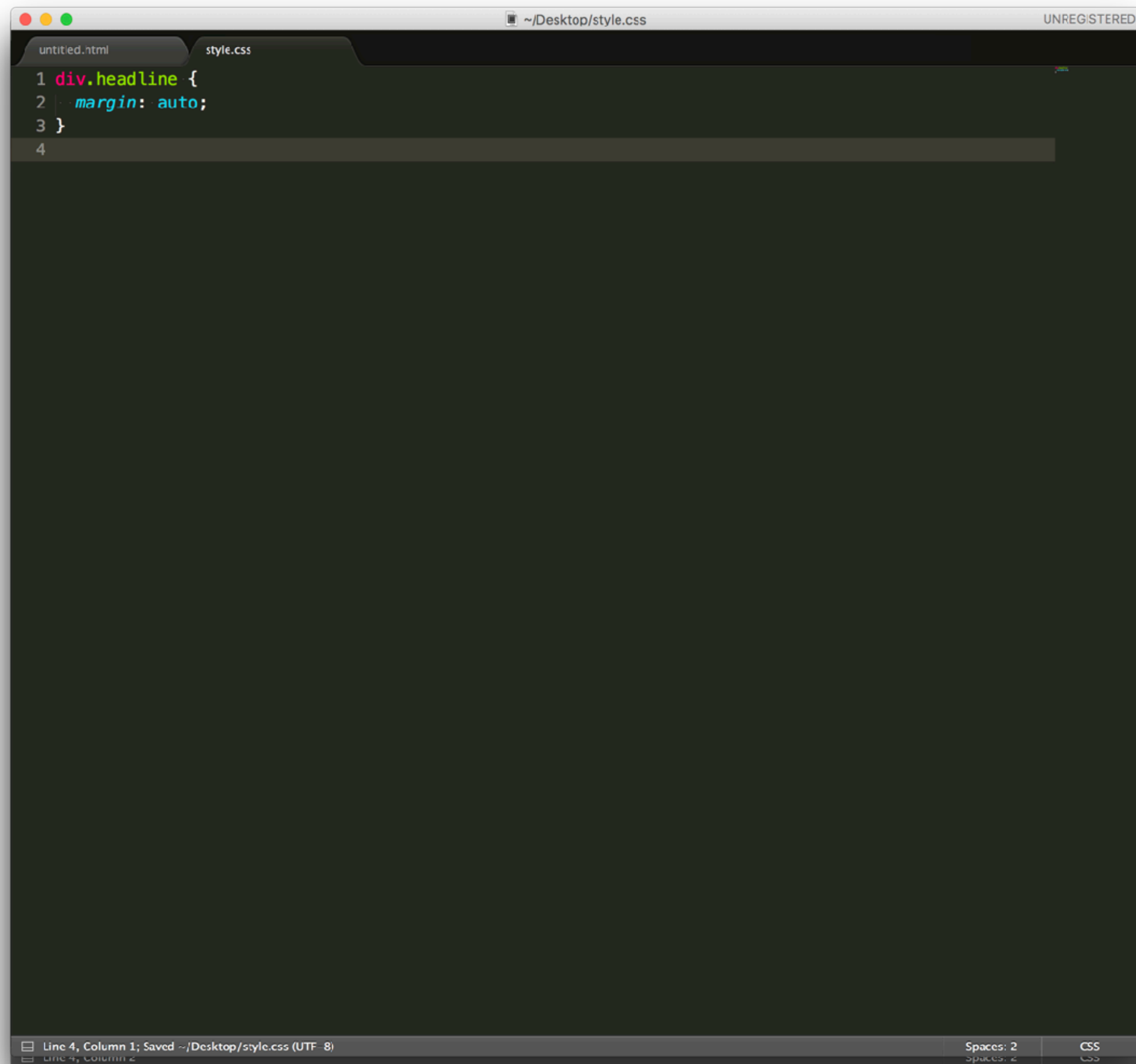
A screenshot of a code editor window. The title bar shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing four lines of CSS code: 1 p {, 2 border: 3px solid #2D3FA3;, 3 margin-right: 15px;, 4 }. The code is color-coded: 'p' is pink, '{' is light blue, 'border:' is light blue, '3px' is pink, 'solid' is light blue, '#2D3FA3;' is light blue, 'margin-right:' is light blue, '15px;' is pink, and '}' is light blue. The status bar at the bottom shows 'Line 4, Column 2', 'Spaces: 2', and 'CSS'.

```
1 p {  
2   border: 3px solid #2D3FA3;  
3   margin-right: 15px;  
4 }
```

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.



The image shows a code editor window with a dark theme. The title bar at the top indicates the file path is ~/Desktop/style.css and the application is UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

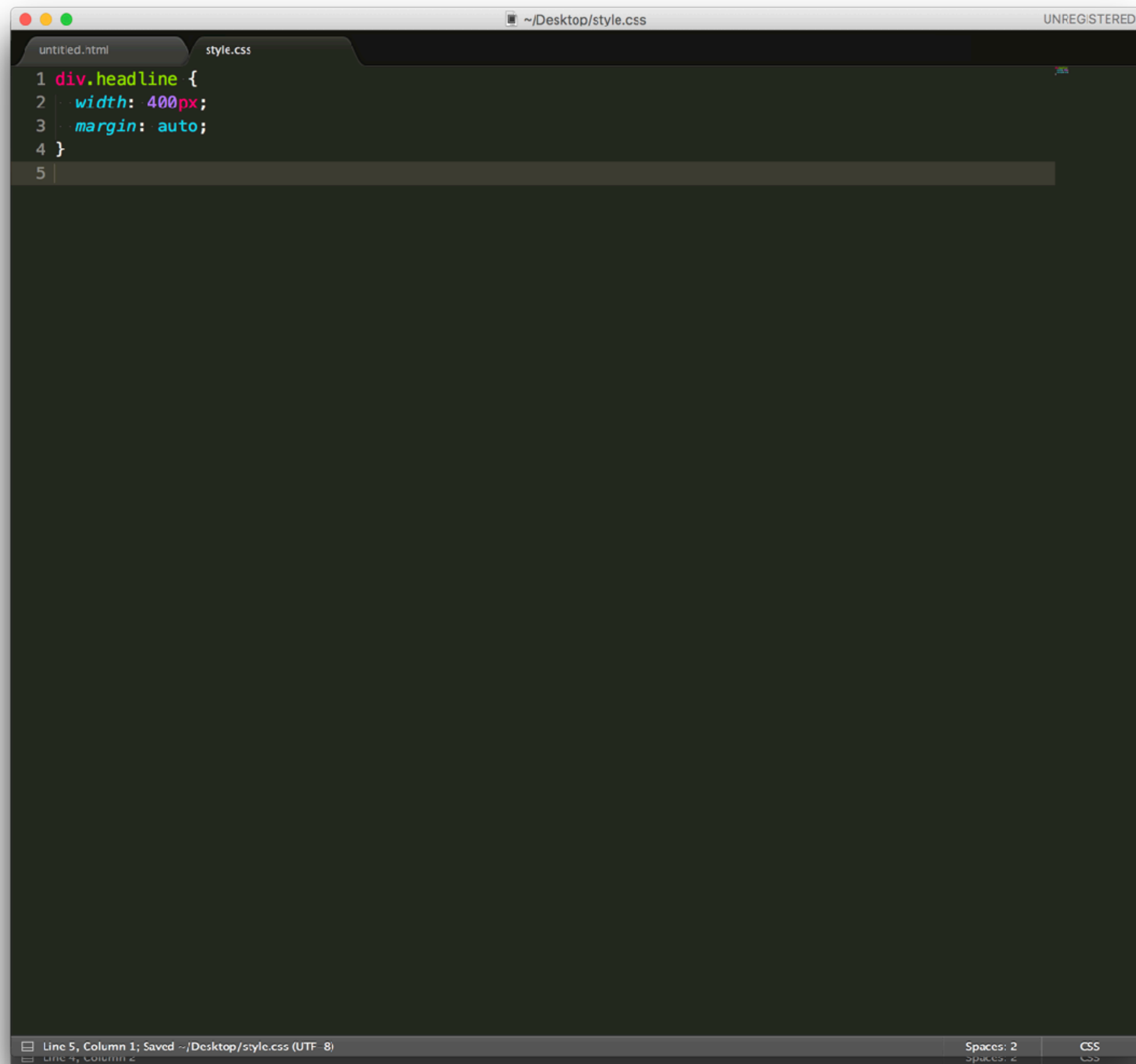
```
1 div.headline {  
2   margin: auto;  
3 }  
4
```

The code is syntax-highlighted: 'div' is pink, 'headline' is green, '{' is blue, 'margin' is light blue, 'auto' is light blue, and ';' is light blue. The line numbers 1, 2, 3, and 4 are on the left. The status bar at the bottom shows 'Line 4, Column 1; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

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.

A screenshot of a code editor window. The title bar at the top shows a file icon, the path ~/Desktop/style.css, and the text UNREGISTERED. Below the title bar, there are two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing the following CSS code:

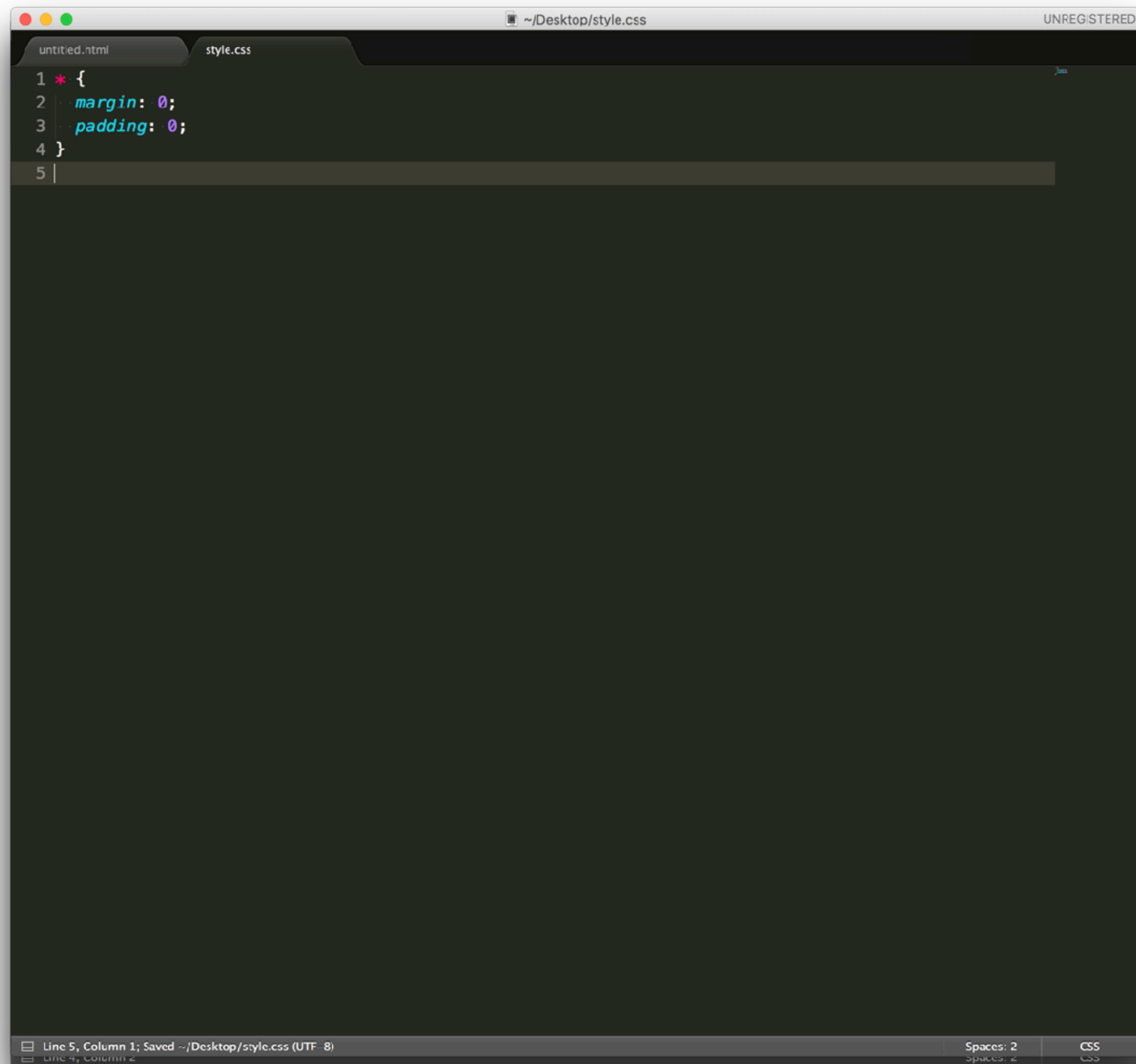
```
1 div.headline {  
2   width: 400px;  
3   margin: auto;  
4 }  
5
```

The code is syntax-highlighted: 'div' is blue, 'headline' is green, '{' is red, 'width' is blue, '400px' is green, 'margin' is blue, 'auto' is green, and '}' is red. The editor has a dark background. At the bottom, there is a status bar with the text 'Line 5, Column 1; Saved ~/Desktop/style.css (UTF-8)', 'Spaces: 2', and 'CSS'.

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.

A screenshot of a code editor window. The title bar shows the file path as ~/Desktop/style.css and the status as UNREGISTERED. The editor has two tabs: 'untitled.html' and 'style.css'. The 'style.css' tab is active, showing a CSS reset rule. The code is as follows:

```
1 * {  
2   margin: 0;  
3   padding: 0;  
4 }  
5 |
```

The code is syntax-highlighted: the asterisk is red, the opening curly brace is blue, the property names 'margin' and 'padding' are green, and the values '0' are blue. The closing curly brace is blue. The editor has a dark background and a light-colored scrollbar on the right. The status bar at the bottom shows 'Line 5, Column 1; Saved ~/Desktop/style.css (UTF-8)' and 'Spaces: 2'.

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.

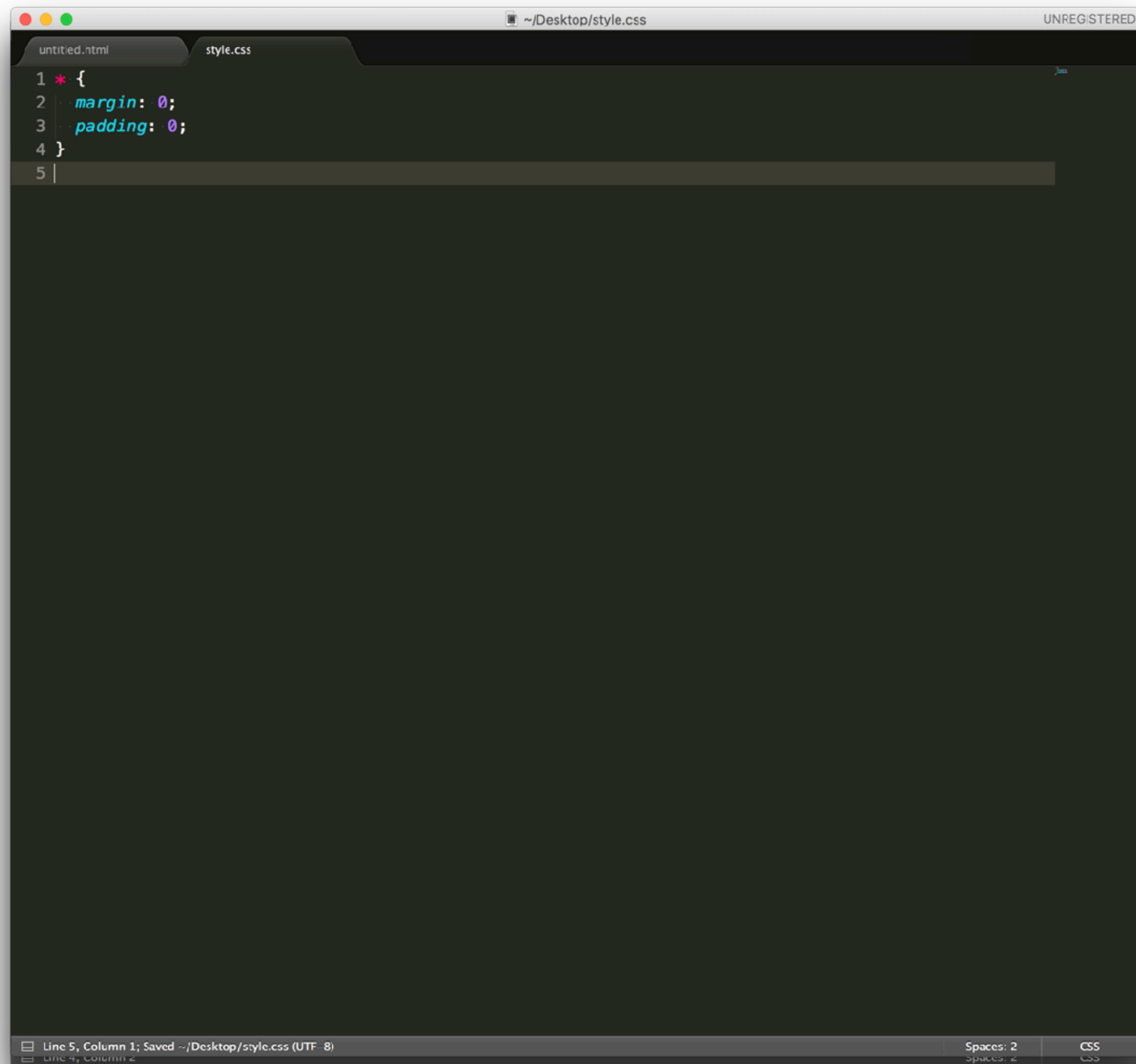
```
~/Desktop/untitled.html UNREGISTERED
untitled.html style.css
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Organizing HTML & CSS</title>
5 <link rel="stylesheet" type="text/css" href="style.css">
6 </head>
7 <body>
8 <ul>
9 <li>Home</li>
10 <li>Products</li>
11 <li>Settings</li>
12 <li>Inbox</li>
13 </ul>
14 </body>
15 </html>
16
```

Line 13, Column 10 Spaces: 2 HTML

```
~/Desktop/style.css UNREGISTERED
untitled.html style.css
1 li {
2 display: inline;
3 }
4
```

Line 4, Column 1 Spaces: 2 CSS

Elements can be hidden from view with the **visibility** property.

A screenshot of a code editor window. The title bar at the top shows a file icon, the path "~/Desktop/style.css", and the text "UNREGISTERED". Below the title bar, there are two tabs: "untitled.html" and "style.css". The "style.css" tab is active, showing the following CSS code:

```
1 * {  
2   margin: 0;  
3   padding: 0;  
4 }  
5 |
```

The code is syntax-highlighted: the asterisk is red, the opening and closing curly braces are blue, the property names "margin" and "padding" are green, and the values "0" are blue. The line numbers 1 through 5 are on the left. The editor has a dark background. At the bottom, there is a status bar with the text "Line 5, Column 1; Saved ~/Desktop/style.css (UTF-8)", "Spaces: 2", and "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.