

# The Box Model Explained

First, let's start with some simple paragraph text...

**This is a paragraph.** Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Next we will add a background color to `<p>` to see how it renders.

**Background color added.** Notice the lack of margins or padding. Background color occupies the lowest layer of an element's visual rendering and takes up only as much internal space as the element occupies.

**Notice the paragraph's width.** Because a paragraph is a "block level element", it naturally takes up 100% of it's containing parent element unless told otherwise.

So let's set the width to 50%.

**Set the width to 50%. Notice three important things:**

**1)** how the paragraph text is forced to wrap sooner, elongating the element, **2)** how the element is still sitting on the left edge of the viewport, and **3)** how it is now 50% of it's parent (**<body>** tag's) width.

Now we can see how a background image works.



**Next, let's add a background image.** A background image is set as a css property and sits above the background color but below the actual content of the element. Thus, it covers the background image but not the element content (in this case text). Like a background color, it will attempt to fill the background space of an element, but you can specify special display options.

To illustrate how these layers stack....

**Next, let's add a background image.** A background image is set as a CSS property and sits above the background color but below the actual content of the element. Thus, it covers the background image but not the element content (in this case text). Like a background color, it will attempt to fill the background space of an element, but you can specify special display options.

## Visualization of CSS Element Depth

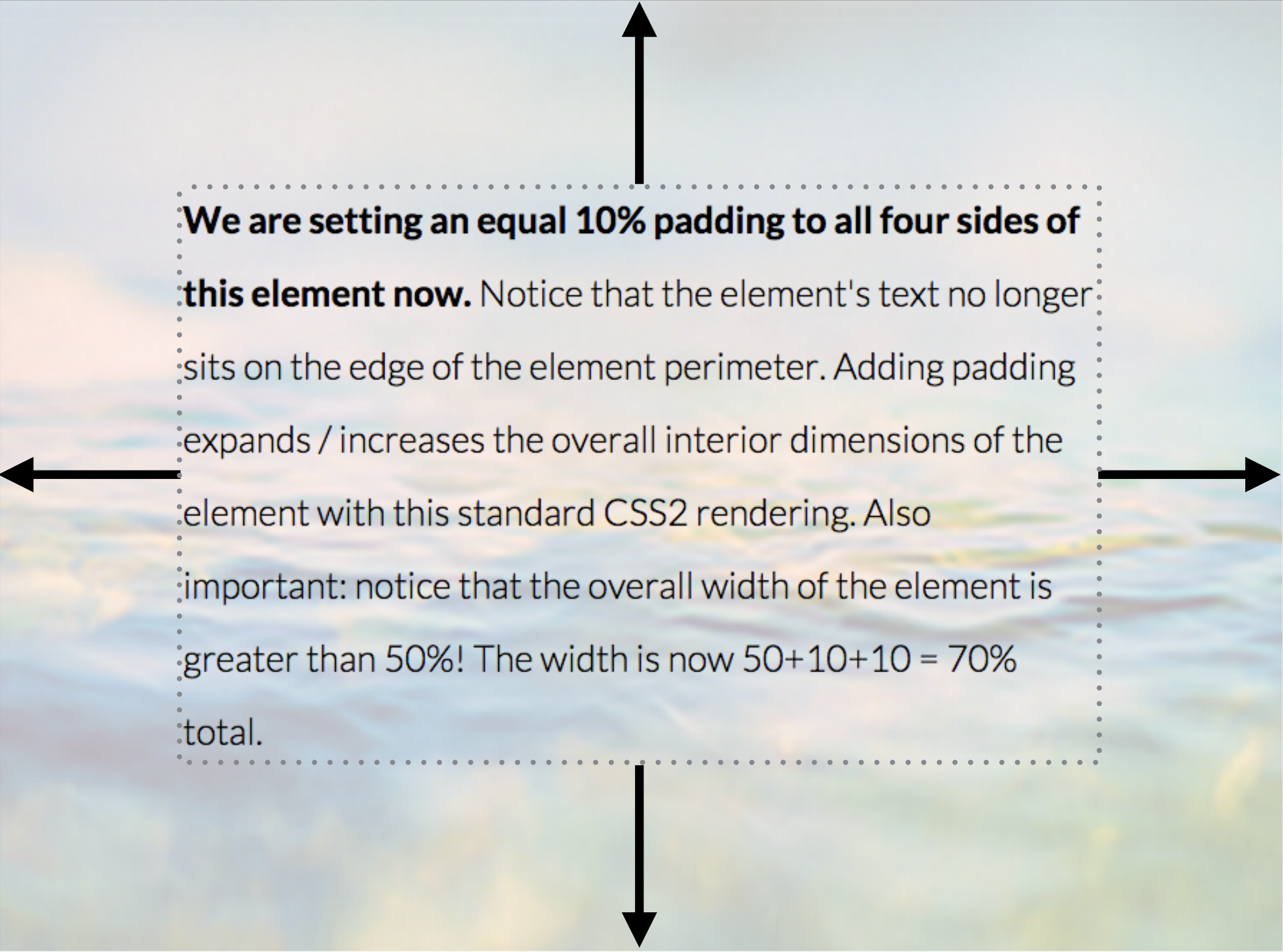
*top:* “p” element

*middle:* background-image

*bottom:* background-color

Now that we have explored the **layers** of a CSS element, we can take a look **how it expands outward**.

**We are setting an equal 10% padding to all four sides of this element now.** Notice that the element's text no longer sits on the edge of the element perimeter. Adding padding expands / increases the overall interior dimensions of the element with this standard CSS2 rendering. Also important: notice that the overall width of the element is greater than 50%! The width is now  $50 + 10 + 10 = 70\%$  total.

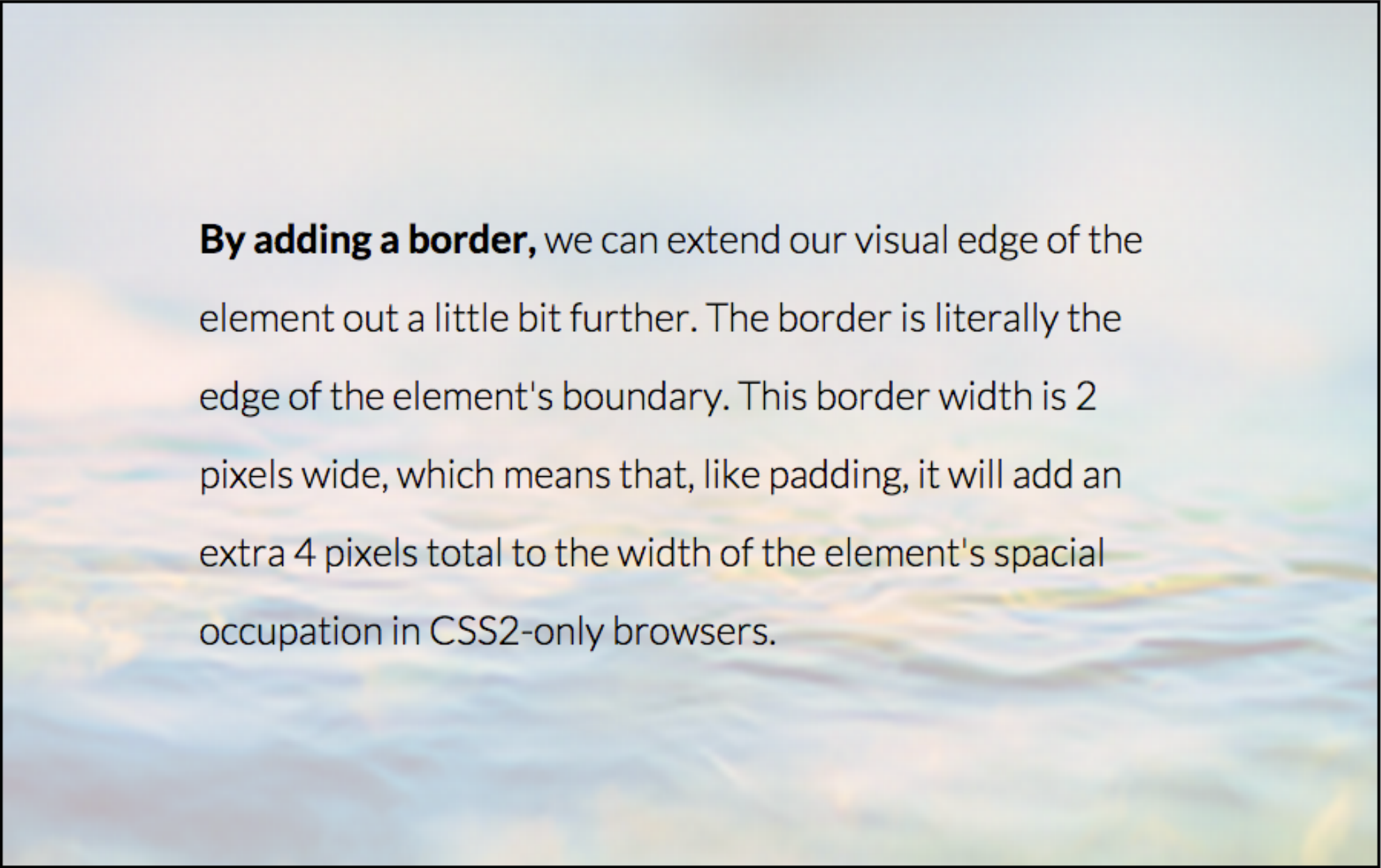


**We are setting an equal 10% padding to all four sides of this element now.** Notice that the element's text no longer sits on the edge of the element perimeter. Adding padding expands / increases the overall interior dimensions of the element with this standard CSS2 rendering. Also important: notice that the overall width of the element is greater than 50%! The width is now  $50 + 10 + 10 = 70\%$  total.

We can extend our visual edge of the element out a little bit further  
by **adding a border**.

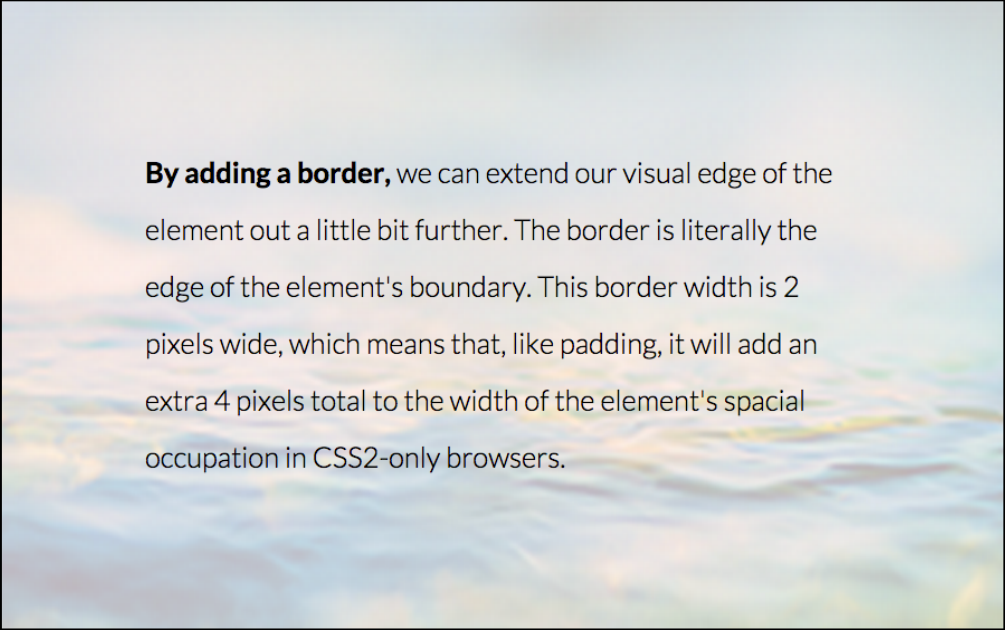
The border is literally the edge of the element's boundary.





**By adding a border,** we can extend our visual edge of the element out a little bit further. The border is literally the edge of the element's boundary. This border width is 2 pixels wide, which means that, like padding, it will add an extra 4 pixels total to the width of the element's spacial occupation in CSS2-only browsers.





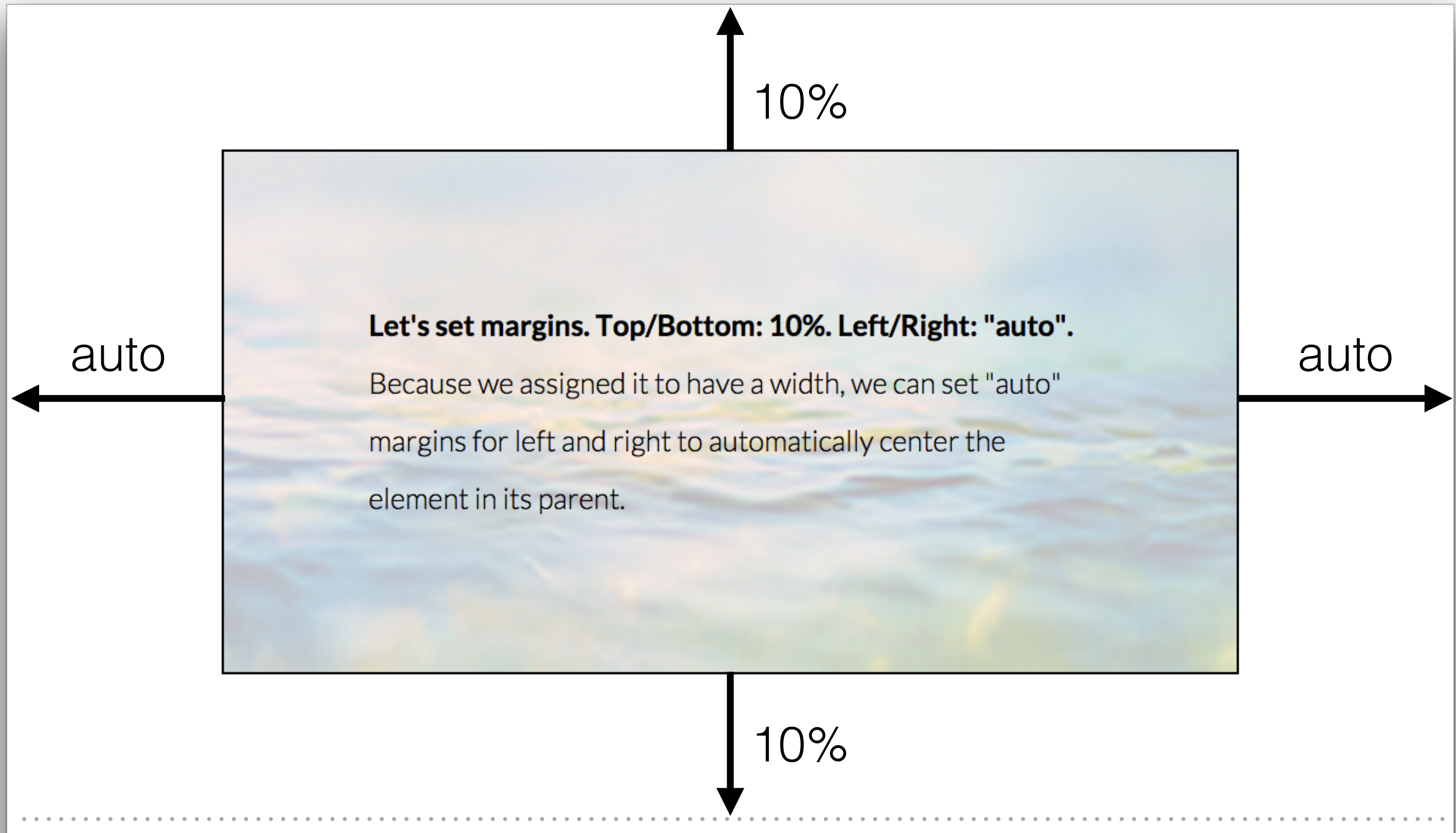
**By adding a border**, we can extend our visual edge of the element out a little bit further. The border is literally the edge of the element's boundary. This border width is 2 pixels wide, which means that, like padding, it will add an extra 4 pixels total to the width of the element's spacial occupation in CSS2-only browsers.

Ideally, an element like this one would also need **margins** added so that it does not hug the edges of other elements.

In this example, the `<p>` element is hugging the left edge of its parent element, the `<body>`.

**Let's set margins. Top/Bottom: 10%. Left/Right: "auto".**

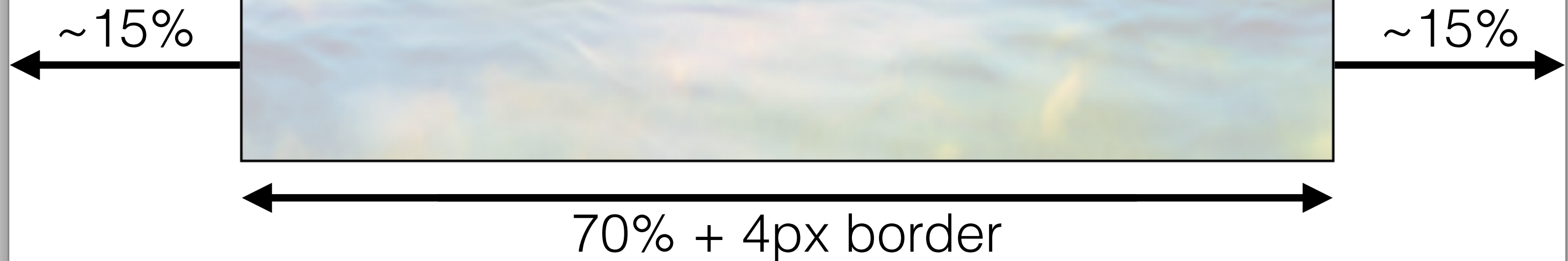
Because we assigned it to have a width, we can set "auto" margins for left and right to automatically center the element in its parent.



The browser subtracts the cumulative set width of `<p>` from the parent's width and then divides it in half to automatically assign left and right margins with the "auto" value.

**Let's set margins. Top/Bottom: 10%. Left/Right: "auto".**

Because we assigned it to have a width, we can set "auto" margins for left and right to automatically center the element in its parent.



Notice, however, that after added padding and borders, we are far off from our original goal of achieving an overall 50% width.

# box-sizing: border-box;

It is now safe to design for modern browsers using the CSS3 property, **box-sizing**.

The example property above indicates that the element's box model will be resized so that padding and border will be calculated to fit within the specified "width" of the element.

**It basically makes the math a lot easier.**

**Apply "box-sizing: border-box;" to the element.** It is now reasonably safe to design for modern browsers using the CSS3 property, box-sizing. The example property above indicates that the element's box model will be resized so that padding and border will be calculated to fit within the specified 50% "width" of the element. **It basically makes the math a lot easier.**

25%

25%

50%