Introduction to the Box Model

Browsers load HTML elements with default position values. This often leads to an unexpected and unwanted user experience, while limiting the views you can create. In this lesson you will learn about the *box model*, an important concept to understand how elements are positioned and displayed on a website.

If you have used HTML and CSS, you have unknowingly seen aspects of the box model. For example, if you have set the background color of an element, you may have noticed that the color was applied not only to the area directly behind the element, but also to the area to the right of the element. Also, if you have aligned text, you know it is aligned relative to something. What is that something?

All elements on a web page are interpreted by the browser as “living” inside of a box. This is what is meant by the box model.

For example, when you change the background color of an element, you change the background color of its entire box.

In this lesson, you’ll learn about the following aspects of the box model:

1. The dimensions of an element’s box.
2. The borders of an element’s box.
3. The paddings of an element’s box.
4. The margins of an element’s box.

Let’s begin!

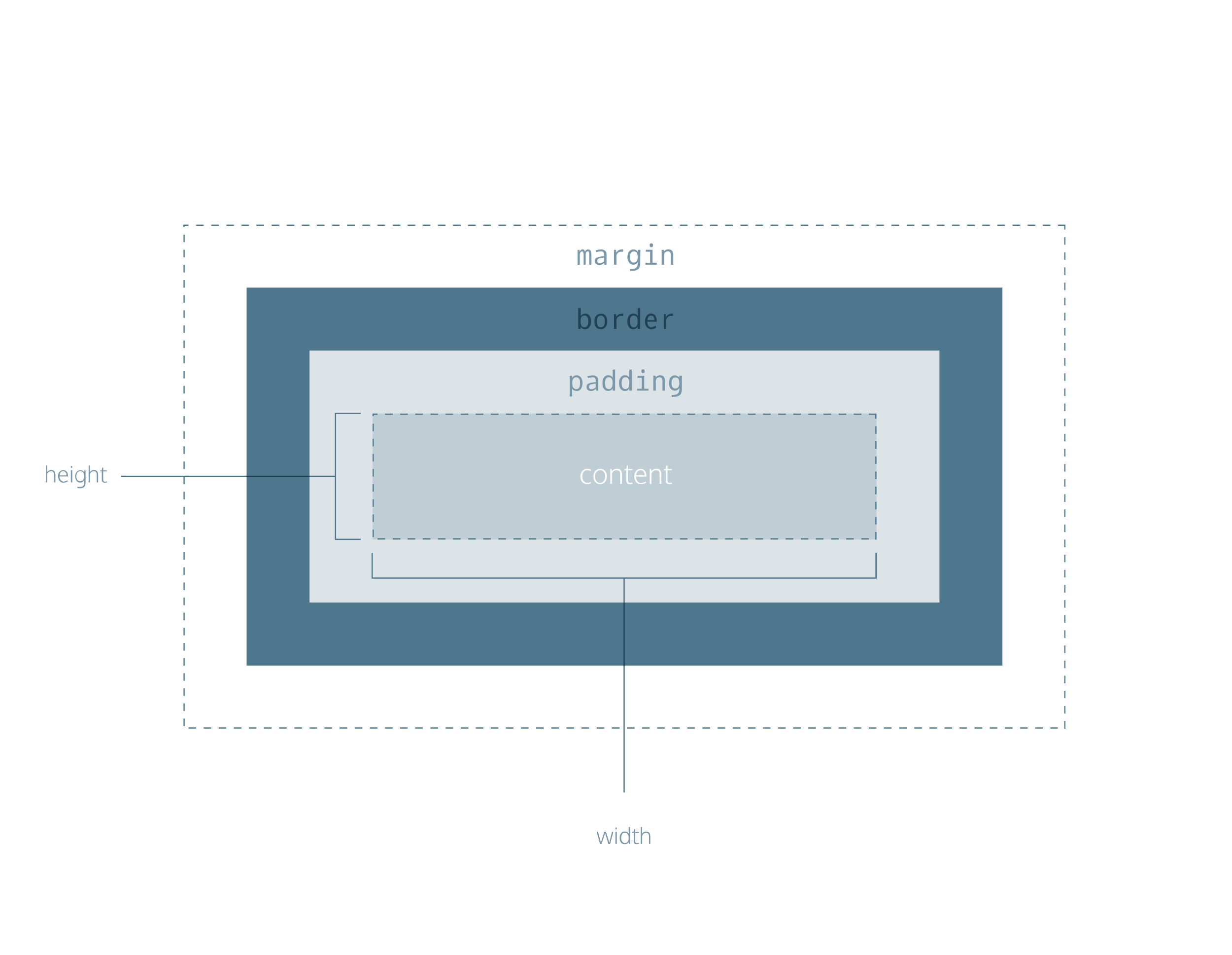
The Box Model

The box model comprises the set of properties which define parts of an element that take up space on a web page. The model includes the content area’s size (*width* and *height*) and the element’s *padding*, *border*, and *margin*. The properties include:

1. Width and height — specifies the width and height of the content area.
2. Padding — specifies the amount of space between the content area and the border.
3. Border — specifies the thickness and style of the border surrounding the content area and padding.
4. Margin — specifies the amount of space between the border and the outside edge of the element.

The image to the right is a visual representation of the box model.

Open [the box model image](https://content.codecademy.com/courses/freelance-1/unit-4/diagram-boxmodel.svg) in a new tab so you can reference the box model as you move through the lesson.



Height and Width

An element’s content has two dimensions: a height and a width. By default, the dimensions of an HTML box are set to hold the raw contents of the box.

The CSS height and width properties can be used to modify these default dimensions.

p {  
  height: 80px;  
  width: 240px;  
}

In this example, the height and width of paragraph elements are set to 80 pixels and 240 pixels, respectively — the px in the code above stands for *pixels*.

Pixels allow you to set the exact size of an element’s box (width and height). When the width and height of an element are set in pixels, it will be the same size on all devices — an element that fills a laptop screen will overflow a mobile screen.

Borders

A *border* is a line that surrounds an element, like a frame around a painting. Borders can be set with a specific width, style, and color.

1. width — The thickness of the border. A border’s thickness can be set in pixels or with one of the following keywords: thin, medium, or thick.
2. style — The design of the border. Web browsers can render any of [10 different styles](https://developer.mozilla.org/en-US/docs/Web/CSS/border-style#Values). Some of these styles include: none, dotted, and solid.
3. color — The color of the border. Web browsers can render colors using a few different formats, including [140 built-in color keywords](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value).

p {  
  border: 3px solid coral;  
}

In the example above, the border has a width of 3 pixels, a style of solid and a color of coral. All three properties are set in one line of code.

The default border is medium none color, where color is the current color of the element. If width, style, or color are not set in the CSS file, the web browser assigns the default value for that property.

p.content-header {  
  height: 80px;  
  width: 240px;  
  border: solid coral;  
}

In this example, the border style is set to solid and the color is set to coral. The width is not set, so it defaults to medium.

Border Radius

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.

You can modify the corners of an element’s border box with the border-radius property.

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

The code in the example above will set *all four corners* of the border to a radius of 5 pixels (i.e. the same curvature that a circle with a radius of 5 pixels would have).

You can create a border that is a perfect circle by first creating an element with the same width and height and then setting the radius equal to the distance between the center of the circle and the width of the box, which is 50%.

div.container {  
  height: 60px;  
  width: 60px;  
  border: 3px solid rgb(22, 77, 100);  
  border-radius: 50%;  
}

The code in the example above creates a <div> that is a perfect circle.

Padding I

The space between the contents of a box and the borders of a box is known as *padding*. Padding is like the space between a picture and the frame surrounding it. In CSS, you can modify this space with the padding property.

p.content-header {  
  border: 3px solid coral;  
  padding: 10px;  
}

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

The padding property is often used to expand the background color and make content look less cramped.

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

1. padding-top
2. padding-right
3. padding-bottom
4. padding-left

Each property affects the padding on only one side of the box’s content, giving you more flexibility in customization.

p.content-header {  
  border: 3px solid fuschia;  
  padding-bottom: 10px;  
}

In the example above, only the bottom side of the paragraph’s content will have a padding of 10 pixels

Padding II

Another implementation of the padding property lets you specify exactly how much padding there should be on each side of the content in a single declaration.

p.content-header {  
  border: 3px solid grey;  
  padding: 6px 11px 4px 9px;  
}

In the example above, the four values 6px 11px 4px 9px correspond to the amount of padding in a clockwise rotation. In order, it specifies the amount of padding on the top (6 pixels), right (11 pixels), bottom (4 pixels), and left (9 pixels) sides of the content.

When using this implementation of the padding property, we must specify a padding value for all four sides of the element.

However, if the top and bottom values for padding will equal each other, and the left and right values for padding will also equal each other, you can use the following shortcut:

p.content-header {  
  padding: 5px 10px;  
}

The first value, 5px, sets the padding value for the top and bottom sides of the content. The second value, 10px, sets the padding value for the left and right sides of the content.

Margins I

So far you’ve learned about the following components of the box model: content, borders, and padding. The fourth and final component of the box model is *margin*.

Margin refers to the space directly outside of the box. The margin property is used to specify the size of this space.

p {  
  border: 1px solid aquamarine;  
  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’s border.

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

1. margin-top
2. margin-right
3. margin-bottom
4. margin-left

Each property affects the margin on only one side of the box, providing more flexibility in customization.

p {  
  border: 3px solid DarkSlateGrey;  
  margin-right: 15px;  
}

In the example above, only the right side of the paragraph’s box will have a margin of 15 pixels. It’s common to see margin values used for a specific side of an element.

Margins II

What if you don’t want equal margins on all four sides of the box?

A similar implementation of the margin property is used to specify exactly how much margin there should be on each side of the box in a single declaration.

p {  
  margin: 6px 10px 5px 12px;  
}

In the example above, the four values 6px 10px 5px 12px refer to the amount of margin around the box in a clockwise rotation. In order, it specifies the amount of margin on the top (6 pixels), right (10 pixels), bottom (5 pixels), and left (12 pixels) sides of the box.

When using this implementation of the margin property, the margin value must be specified for all four sides of the box.

Just like the padding shortcut, when you’re certain that the top and bottom values for margin will equal each other, and that the left and right values for margin will also equal each other, you can use the following shortcut:

p {  
  margin: 6px 12px;  
}

The first value, 6px, sets a margin value for the top and bottom of the box. The second value, 12px, sets a margin value for the left and right sides of the box.

Auto

The margin property also lets you center content. However, you must follow a few syntax requirements. Take a look at the following example:

div {  
  margin: 0 auto;  
}

In the example above, margin: 0 auto; will center the divs in their containing elements. The 0 sets the top and bottom margins to 0 pixels. The auto value instructs the browser to adjust the left and right margins until the element is centered within its containing element.

The div elements in the example above should center within an element that fills the page, but this doesn’t occur. 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 to center an element that takes up the full width of the page.

div.headline {  
  width: 400px;  
  margin: 0 auto;  
}

In the example above, the width of the div is set to 400 pixels, which is less than the width of most screens. This will cause the div to center within a containing element that is greater than 400 pixels wide.

Margin Collapse

As you have seen, padding is space added inside an element’s border, while margin is space added outside an element’s border. One additional difference is that top and bottom margins, also called vertical margins, *collapse*, while top and bottom padding does not.

Horizontal margins (left and right), like padding, are always displayed and added together. For example, if two divs with ids #div-one and #div-two, are next to each other, they will be as far apart as the sum of their adjacent margins.

#img-one {  
  margin-right: 20px;  
}  
   
#img-two {  
  margin-left: 20px;  
}

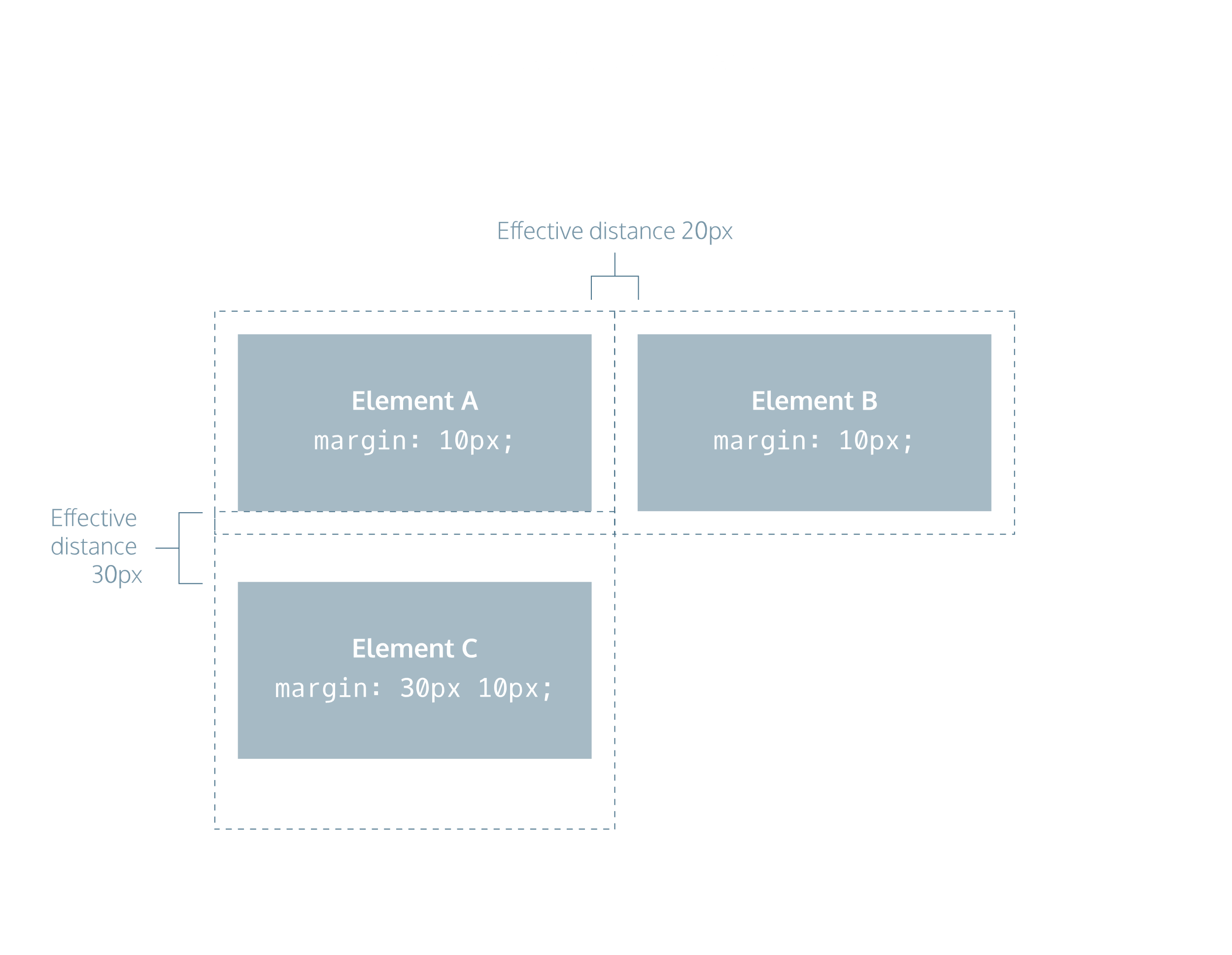
In this example, the space between the #img-one and #img-two borders is 40 pixels. The right margin of #img-one (20px) and the left margin of #img-two (20px) add to make a total margin of 40 pixels.

Unlike horizontal margins, vertical margins do not add. Instead, the larger of the two vertical margins sets the distance between adjacent elements.

#img-one {  
  margin-bottom: 30px;  
}  
   
#img-two {  
  margin-top: 20px;  
}

In this example, the vertical margin between the #img-one and #img-two elements is 30 pixels. Although the sum of the margins is 50 pixels, the margin collapses so the spacing is only dependent on the #img-one bottom margin.

It may be helpful to think of collapsing vertical margins as a short person trying to push a taller person. The tall person has longer arms and can easily push the short person, while the person with short arms cannot reach the person with long arms.



Minimum and Maximum Height and Width

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.

1. min-width — this property ensures a minimum width of an element’s box.
2. max-width — this property ensures a maximum width of an element’s box.

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

In the example above, the width of all paragraphs will not shrink below 300 pixels, nor will the width exceed 600 pixels.

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.

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

1. min-height — this property ensures a minimum height for an element’s box.
2. max-height — this property ensures a maximum height of an element’s box.

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

In the example above, the height of all paragraphs will not shrink below 150 pixels and the height will not exceed 300 pixels.

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. You’ll learn how to work around this issue in the next exercise.

Overflow

All of the components of the box model comprise an element’s size. For example, an image that has the following dimensions is 364 pixels wide and 244 pixels tall.

* 300 pixels wide
* 200 pixels tall
* 10 pixels padding on the left and right
* 10 pixels padding on the top and bottom
* 2 pixels border on the left and right
* 2 pixels border on the top and bottom
* 20 pixels margin on the left and right
* 10 pixels margin on the top and bottom

The total dimensions (364px by 244px) are calculated by adding all of the vertical dimensions together and all of the horizontal dimensions together. Sometimes, these components result in an element that is larger than the parent’s containing area.

How can we ensure that we can view all of an element that is larger than its parent’s containing area?

The overflow property controls what happens to content that spills, or overflows, outside its box. It can be set to one of the following values:

* hidden - when set to this value, any content that overflows will 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.
* visible - when set to this value, the overflow content will be displayed outside of the containing element. Note, this is the default value.

p {  
  overflow: scroll;   
}

In the example above, if any of the paragraph content overflows (perhaps a user resizes their browser window), a scrollbar will appear so that users can view the rest of the content.

The overflow property is set on a parent element to instruct a web browser how to render child elements. For example, if a div’s overflow property is set to scroll, all children of this div will display overflowing content with a scroll bar.

Resetting Defaults

All major web browsers have a default stylesheet they use in the absence of an external stylesheet. These default stylesheets are known as *user agent stylesheets*. In this case, the term “[user agent](https://en.wikipedia.org/wiki/User_agent)“ is a technical term for the browser.

User agent stylesheets 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.

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

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

Note that both properties are both set to 0. When these properties are set to 0, they do not require a unit of measurement.

Visibility

Elements can be hidden from view with the visibility property.

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

1. hidden — hides an element.
2. visible — displays an element.

<ul>  
  <li>Explore</li>  
  <li>Connect</li>  
  <li class="future">Donate</li>  
</ul>

.future {  
  visibility: hidden;  
}

In the example above, the list item with a class of future will be hidden from view in the browser.

Keep in mind, however, that users can still view the contents of the list item (e.g., Donate) by viewing the source code in their browser. Furthermore, the web page will *only* hide the contents of the element. It will still leave an empty space where the element is intended to display.

**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.

Review

In this lesson, we covered the four properties of the box model: height and width, padding, borders, and margins. Understanding the box model is an important step towards learning more advanced HTML and CSS topics. Let’s take a minute to review what you learned.

1. The box model comprises a set of properties used to create space around and between HTML elements.
2. The height and width of a content area can be set in pixels or percentage.
3. Borders surround the content area and padding of an element. The color, style, and thickness of a border can be set with CSS properties.
4. Padding is the space between the content area and the border. It can be set in pixels or percent.
5. Margin is the amount of spacing outside of an element’s border.
6. Horizontal margins add, so the total space between the borders of adjacent elements is equal to the sum of the right margin of one element and the left margin of the adjacent element.
7. Vertical margins collapse, so the space between vertically adjacent elements is equal to the larger margin.
8. margin: 0 auto horizontally centers an element inside of its parent content area, if it has a width.
9. The overflow property can be set to display, hide, or scroll, and dictates how HTML will render content that overflows its parent’s content area.
10. The visibility property can hide or show elements.

**Instructions**

Make some adjustments to the code in the code editor. See if you can improve the appearance of the page by changing the following properties:

1. width
2. height
3. padding
4. border
5. margin
6. overflow

Lesson 2

Why Change the Box Model?

The last lesson focused on the most important aspects of the box model: box dimensions, borders, padding, and margin.

The box model, however, has an awkward limitation regarding box dimensions. This limitation is best illustrated with an example.

<h1>Hello World</h1>

h1 {  
  border: 1px solid black;  
  height: 200px;  
  width: 300px;  
  padding: 10px;  
}

In the example above, a heading element’s box has solid, black, 1 pixel thick borders. The height of the box is 200 pixels, while the width of the box is 300 pixels. A padding of 10 pixels has also been set on all four sides of the box’s content.

Unfortunately, under the current box model, the border thickness and the padding will affect the dimensions of the box.

The 10 pixels of padding increases the height of the box to 220 pixels and the width to 320 pixels. Next, the 1-pixel thick border increases the height to 222 pixels and the width to 322 pixels.

Under this box model, the border thickness and padding are added to the overall dimensions of the box. This makes it difficult to accurately size a box. Over time, this can also make all of a web page’s content difficult to position and manage.

In this brief lesson, you’ll learn how to use a different technique that avoids this problem altogether.

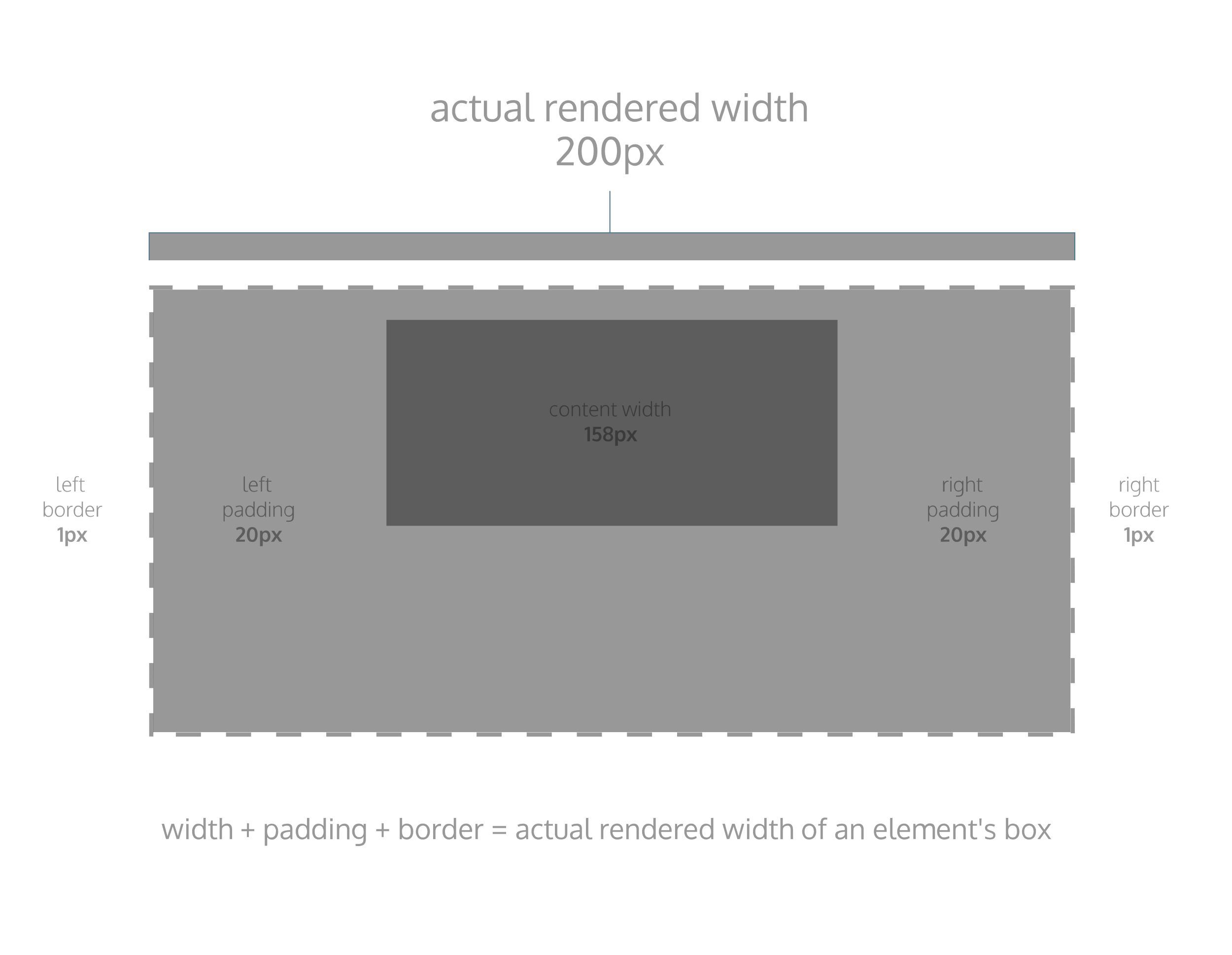
Box Model: Content-Box

Many properties in CSS have a default value and don’t have to be explicitly set in the stylesheet.

For example, the default font-weight of text is normal, but this property-value pair is not typically specified in a stylesheet.

The same can be said about the box model that browsers assume. In CSS, the box-sizing property controls the type of box model the browser should use when interpreting a web page.

The default value of this property is content-box. This is the same box model that is affected by border thickness and padding.



Box Model: Border-Box

Fortunately, we can reset the entire box model and specify a new one: border-box.

\* {  
  box-sizing: border-box;  
}

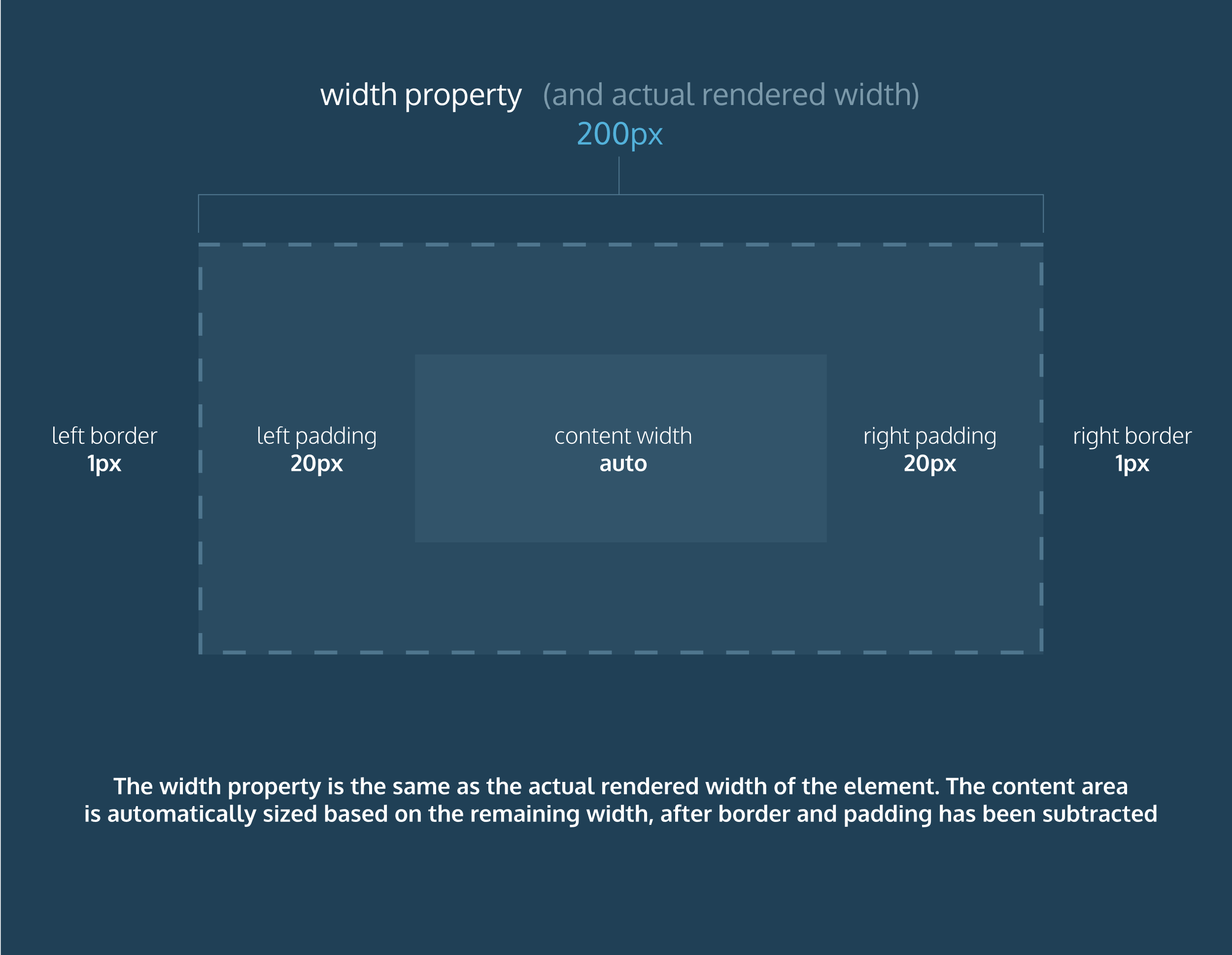
The code in the example above resets the box model to border-box for all HTML elements. This new box model avoids the dimensional issues that exist in the former box model you learned about.

In this box model, the height and width of the box will remain fixed. The border thickness and padding will be included inside of the box, which means the overall dimensions of the box do not change.

<h1>Hello World</h1>

\* {  
  box-sizing: border-box;  
}  
   
h1 {  
  border: 1px solid black;  
  height: 200px;  
  width: 300px;  
  padding: 10px;  
}

In the example above, the height of the box would remain at 200 pixels and the width would remain at 300 pixels. The border thickness and padding would remain entirely *inside* of the box.



The New Box Model

Now that you know about the new box model, let’s actually implement it in the browser.

\* {  
  box-sizing: border-box;  
}

It’s that simple! In the example above, the universal selector (\*) targets all elements on the web page and sets their box model to the border-box model.

Review: Changing the Box Model

In this lesson, you learned about an important limitation of the default box model: box dimensions are affected by border thickness and padding.

Let’s review what you learned:

1. In the default box model, box dimensions are affected by border thickness and padding.
2. The box-sizing property controls the box model used by the browser.
3. The default value of the box-sizing property is content-box.
4. The value for the new box model is border-box.
5. The border-box model is not affected by border thickness or padding.