Content

**The Content Inside**

Box dimensions and borders are just the beginning of the vast number of box properties you can modify in CSS.

In this lesson, you'll learn how to modify the spacing around the content *inside* of the box. This concept is critical in understanding how to customize the way content is displayed to users.

Let's begin!

**Padding I**

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

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.

The padding property is particularly useful at making text easier to read when the text has a border around it.

**Note:** In the last couple of lessons, you learned how to set the width and height of a box. When padding is set for a box (as in the example above) it will be added to the width and height of a box, increasing the dimensions of the box. We'll learn how to avoid this problem in the next lesson.

# Padding II

What if you don't want all four sides of a box's content to have equal padding?

In that case, another version of the padding property lets you specify exactly how much padding there should be on each side of the content.

p {

border: 3px solid #XXXXXX;

padding: 5px 10px 5px 10px;

}

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

When this version of the padding property is used, a padding value must be specified for all four sides of the content.

# Padding III

Take a look at the following code. What do you notice about the padding values?

p {

padding: 5px 10px 5px 10px;

}

The top and bottom values for padding are the same (5px) and the left and right values for padding are also the same (10px).

When you're certain that the top and bottom values for padding will equal each other, and that the left and right values for padding will also equal each other, you can use the following shortcut:

p {

padding: 5px 10px;

}

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

**Padding IV**

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:

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 {

border: 3px solid #2D3FA3;

padding-bottom: 10px;

}

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

# Margin I

So far, you'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. You can adjust this spacing with the margin property.

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.

An understanding of the margin property is crucial in order to later understand element positioning in the browser.

# Margins II

Just like padding, what if you don't want equal margin on all four sides of the box?

In that case, another version of the margin property lets you specify exactly how much margin there should be on each side of the box.

p {

margin: 6px 12px 6px 12px;

}

In the example above, the four values 6px 12px 6px 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 (12 pixels), bottom (6 pixels), and left (12 pixels) sides of the box.

When this version of the margin property is used, a margin value must be specified for all four sides of the box.

# Margin III

Take a look at the following code. What do you notice about the margin values?

p {

margin: 6px 12px 6px 12px;

}

The top and bottom values of margin are the same (6px) and the left and right value of margin are also the same (12px).

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 padding 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 sides of the box. The second value, 12px sets a margin value for the left and right sides of the box.

**Margin IV**

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, giving you more flexibility in customization.

p {

border: 3px solid #2D3FA3;

margin-right: 15px;

}

In the example above, only the right side of the paragraph's box will have a margin of 15 pixels.

Because the margin property directly modifies the space outside of a box, it's very common to see margin values used for only specific sides of boxes. As you learn more about web development, you'll likely come across many different ways of positioning elements with the margin property.

# Margin: Auto

The margin property also lets you center content, if you follow certain requirements. Take a look at the following example.

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.

div.headline {

width: 400px;

margin: auto;

}

In the example above, 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. This is important to keep in mind when attempting to center nested elements, like divs.

# 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.

**Display**

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.

It's common to use the display property to create a navigation bar from list items, like so:

<ul>

<li>Home</li>

<li>Products</li>

<li>Settings</li>

<li>Inbox</li>

</ul>

li {

display: inline;

}

In the example above, the block-level list items will now behave as inline elements. The result will be list items appearing on the same line, like a navigation bar.

**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.