**The Box Model**

The box model is an important concept to understand how elements are positioned and displayed on a website.

All elements on a web page are interpreted by the browser as “living” inside of its 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.

Graphical user interface

Description automatically generated

The box model comprises the set of properties that define parts of an element that take up space on a web page. It includes:

* width and height: The width and height of the content area.
* padding: The amount of space between the content area and the border.
* border: The thickness and style of the border surrounding the content area and padding.
* margin: The amount of space between the border and the outside edge of the element.

1. **Height and Width**

An element’s content has 2 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;  
}

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.

1. **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. All three properties are set in one line of code. Any of the 3 can be omitted 🡪 default

* 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. Default = medium
* 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. Default = none
* 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). Default = color (of element)

p {

border: 3px solid coral;

}

1. **Border radius**

Thanks to CSS, a border doesn’t have to be square. We can add the **border-radius** property, to make all four corners of the border to some curvature pixels.

div.container {  
  border: 3px solid blue;  
  border-radius: 5px;  
}

If we want to create a perfect circle border, the element should have the same width and height, and then the radius set to half the width of the box, which is 50%.

border-radius: 50%;

1. **Padding**

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.

In CSS, you can modify this space with the **padding** property.

If you want to be more specific about the amount of padding on each side, you can use the following properties: padding-top/right/bottom/left. Each property affects the padding on only 1 box side.

We can declare these properties in one line. Note that the order will be top-right-bottom-left

* 1 value: all 4 sides has equal space
* 2 values: top(=bottom) right(=left)
* 3 values: top right(=left) bottom
* 4 values top right bottom left

1. **Margin**

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

If we want to be even more specific about the amount of margin on each side of a box, we can use margin-top/right/bottom/left. The margin shorthand order is similar (top-right-bottom-left)

1. **Auto ♣♣**

The margin property also lets you center content. However, you must follow a few syntax requirements.

div.headline {  
  width: 400px;  
  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.

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, since the width of the page can change due to display and/or browser window size.

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.

1. **Margin Collapse**

One additional difference between margin and padding, although they are both spaces, is that top and bottom margins, or vertical margins, collapses.

This means that horizontal margins are always displayed and added together between elements.

However, vertical margins **do not add.** Instead, the larger of the 2 vertical margins sets the distance between adjacent elements.

Graphical user interface, application

Description automatically generated

1. **Minimum and Maximum Height and Width**

To avoid the situation where the content will be too spread out/too small, CSS offers 2 properties that can limit how narrow or how wide an element’s box can be sized to.

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

p {

min-width: 300px;

max-width: 600px;

}

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 (overflow), resulting in content that is not legible.

1. **Overflow**

Sometimes, the components of the box model (width, height, padding, margin, etc. added all together becomes an element’s size) may result in an element that is larger than the parent’s containing area.

To ensure that we can view all of an element that is larger than its parent’s containing area, we add the **overflow** property, which controls what happens to content that overflows. Value options are:

* 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.

There are also other options. View them [here](https://developer.mozilla.org/en-US/docs/Web/CSS/overflow).

Table

Description automatically generated with low confidence

1. **Resetting Defaults**

All major web browsers have a default stylesheet, known as the *user agent stylesheet*. They have default CSS rults that set default values for padding and margin. This affects how 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;  
}

It is often the first CSS rule in an external stylesheet.

1. **Visibility**

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

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

* hidden — hides an element.
* visible — displays an element.
* collapse — collapses 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.