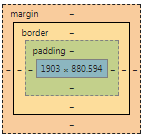
Layout Concepts

**Box Model**: Under the box model, every element in the HTML document is contained within a box. This box has four important properties: content, padding, border, and margin.

The *content* is your media, whether text, image, video, audio, forms, etc.

The *padding* is the space wrapped around your content.

The *border* is the delineation at the edge of your padding.

The *margin* is the space outside of the border; you can think of it as more padding.

Let’s see this model applied to a home in the suburbs.

The *content* is your physical house itself. Then, you might have a fence around the house, which is the *border*. And in between the house and the fence, you have a nice yard with freshly-mowed grass and some flowers. Or maybe balloons. Point is, putting the fence or *border* up against the house would look terrible, so we have some space in between, which is the *padding*.

And what happens if you have a neighbor with a house much like yours, and a fence like yours, pressed right up against your fence? Well technically that would be fine too, just like having a fence right against your house would be fine, but does it ever work that well? Because what if Joe grows a bunch of trees and their pointy branches are getting very close to your balloons, and you passive-aggressively erect a giant sign to express your feelings, and now everyone is claustrophobic and unhappy?

You might say, well, couldn’t we have some extra space between his fence and ours?

Yes, and you would be right on the money. We call this space *margin*.

So to reiterate, we have first the content, next the padding, then the border, and lastly margin. And this would constitute your box, both with your home and your element.

Next, let’s delve deeper and explore a related property of every element called *box-sizing*.

Box-sizing is by default set to “content-box”, which just means that when you set a height and width for your element, that stated size includes only the content. It doesn’t take into account padding or border, so the final size of your element can become larger than the figures you specified. Conversely, the second option is “border-box” which includes border and padding in the figure. If you increase border or padding, it’ll decrease the content size to accommodate.

Let’s illustrate why it’s usually better to set box-sizing to “border-box” instead of “content-box.”

Say there’s two new neighbors, Joe and Chris. They each get a contract that says their “home” can be 1000 feet wide.

Joe uses border-box method, and measures the 1000 feet as the space between each fence. He builds his house, 700 feet wide, then has 150 feet of padding on each side for his yard. Joe is a (box) model citizen!

On the other hand, Chris uses content-box for box-sizing. He builds his house 1000 feet wide as per the contract, which is technically okay too, and things are fine for a year.

But then, Chris decides he doesn’t have enough space! He says he wants a thick stone wall and space to park his car in! So he says to Joe, “Hey man, can’t you move over?” But Joe can’t move over, he has Janice on the other side. Plus, he paid for this space and it’s rightfully his! To which Chris replies, “But dude, I need the space.” –cue staring- And so when Joe is away on vacation, Chris ignores Joe’s fence and parks his monster truck on it. And builds a fifty foot wall through Joe’s house. And the cops are called, and everything’s a clusterf\*ck.

Anyway, moral of the story is: if you use content-box, you are likely to be a twat.

To avoid problems in your coding, it’s good practice to set all elements to use border-box instead of the default content-box.

\*, \*:before, \*:after {

-webkit-box-sizing: border-box;

-moz-box-sizing: border-box;

box-sizing: border-box;

}

**Specificity**: If a given element is targeted by multiple CSS rules that conflict, it will always accept the rule that has higher specificity. CSS has a point-system for determining the specificity of a rule, and higher points always win out. For simplicity’s sake, we can use base-10:

Element tags (p, div, section, h1) are worth 1 pt.

Class selectors and pseudoclass (.visible, .contact, div:hover) are worth 10 pts.

Id tags (#presidentImg, #editedSpeech) are worth 100 pts.

Inline style attributes are worth 1000 pts.

Points are added up, so a rule selector with two element tags and one class tag is worth 1 + 1 + 10 or 12.

The exception is the “!important” modifier that can be attached to an attribute to make it take precedence no matter what.

.last {

margin-right: 0 !important;

}

**Positioning (Static, Relative, Absolute, Fixed, Sticky):**

Static is the default attribute value, meaning the DOM will arrange itself according to the natural flow of the HTML document.

Relative means an element will not be removed from the natural order of the document, but can be moved relative from its original position if attributes like left, right, top, bottom are applied. Elements may overlap.

Absolute will remove an element from the natural order of the document, so everything will get rearranged as if that element did not exist on the page. Elements may overlap. The element will be positioned in reference to its parent element.

Fixed means the element will stay where it is on the page even when scrolled, and is positioned in reference to the window.

Sticky acts like relative until scrolled to a given point, then it will act like a fixed element.

**Visibility vs Hidden:**

Visibility will allow the element to be placed on the page; it isn’t visible, but there will still be an empty placeholder space.

Hidden will remove the element so it no longer occupies the space. It still exists in the DOM and can be manipulated by JS, but other elements will take the space it would’ve been in.