All right! Now that you know all about CSS, it's time to learn the last (but certainly not least) major piece of the puzzle: positioning.

Controlling the position of HTML elements allows you incredibly fine control over how your pages look. No longer will your <div>s sit directly on top of one another! (Unless you want them to.)

As we mentioned, elements populate the page in what's known as the **CSS box model**. Each HTML element is like a tiny box or container that holds the pictures and text you specify.

Display:

Display:block -

Good work! If you didn't notice much of a difference, don't worry. Our<div>s were block elements by default; as we specify different display values, they'll start to move around.

Display Inline-block -

As mentioned, any element that comes in as a block (say, a paragraph) will automatically take up the full width of the page, no matter how much or how little content you put in.

If we specify a display of inline-block, however, our blocks are still blocks, but will be able to sit next to each other on the same line.

Display :inline -

Did you see that? Your <div>s all moved onto the same line! You can already start to see how this type of positioning can be useful for navigation bars like the one at the top of the main Codecademy page (where you can click "Learn," "Teach," and so on).

The inline-block value allows you to put several block elements on the same line. The inline value places all your elements next to one another, but not as blocks: they don't keep their dimensions.

Display:none -

The good news is, inline places all your elements on a single line. The bad news is that it doesn't maintain their "box"ness: as you saw, all your<div>s got squished to the smallest possible width!

The inline display value is better suited for HTML elements that are blocks by default, such as headers and paragraphs.

Finally, we'll try out the display valuenone. As you might expect, this prevents the page from displaying the selected element. As you might *not*expect, this removes the selected element from the page *entirely*, including any children and any content. Poof! Gone! (But not gone forever—changing the display value away fromnone will bring everything back.)

MARGIN - Let's start with our margins. Adjusting our margins not only moves our element relative to other elements on the page, but also relative to the "walls" of the HTML document.

For instance, if we take an HTML element with a specific width (such as our <div> in the editor) and set itsmargin to auto, this tells the document to automatically put equal left and right margins on our element, centering it on the page.

You can specify margins with top, right, bottom, left

PADDING -  padding is the space between your border and your innermost layer: the actual content.

You can specify just like with margins with top, right, bottom, left

NEGATIVE VALUES

When you give CSS a positive padding or margin value, it puts that space between the element and its reference: for instance, if you have a <div> and you give it a margin-left of 20px, it puts twenty pixels between the left margin of that <div> and the side of the screen. This effectively moves the<div> twenty pixels to the right.

If you want to move an element in the other direction, you can give CSS anegative value: margin-left: -20pxwill move the element twenty pixels to the left.