CSS Layout

[“CSS Layout.”]

FEMALE NARRATOR: CSS layout. There are different ways to lay out your web page using CSS. *[“Page Layout.” A picture of an animated webpage is shown.]* Page layout is the process of placing and arranging the text and image content on the page to produce a visually pleasing page. *[A webpage wireframe is shown with boxes showing where pictures and words will be.]* We need to place all the elements in the proper place on the page, so it will look like our wireframe. So what's the best way to do that? *[A webpage is shown.]* Up until now, our elements have been in normal flow, meaning each element is displayed one after another based on how we placed them in the HTML. Some might take more or less room, but they're all in order according to the HTML. This is normal flow. What are some ways to take blocks or elements out of normal flow?

[A picture of a webpage is shown. It has text and a right-aligned picture. Next to the picture of the webpage is text saying, “img { float: right; }” ”https://www.w3schools.com/css/css_float.asp”]

There are options like float that will shift an element to the right or left and allow content to display wrapped around it. The only time I use floats is if I have a small image, button, or logo, thatI want text to wrap around. Keep in mind that float should not be used to layout an entire page. I rarely will use float.

[“position: static;” “An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:” “This div element has position: static;” “https://www.w3schools.com/css/tryit.asp?filename=trycss_position_static"]

The position property can also be used to take elements out of their normal flow. All elements in normal flow have a position property value of static. Meaning in normal flow, all elements display one after the other and if you scroll, they're going to scroll with the page. Position relative changes this flow. *[“position: relative; left: 30px” is placed within the CSS code.]* It would reference a certain point from where the element would normally be in normal flow. *[Within the div.relative and the div. absolute are added position, top, right, and width.]* Position absolute would be removed completely from normal flow and positioned relative either to its parent, if the parent has the position relative, or absolute to the view port of the page itself. It usually will lead normal flow and go to the top left of the page. Then you use offset values to move it where you want to. Bulk position relative and position absolute should be used very sparingly. I've only used these properties when I have a very specific reason to do so. But again, not to lay out the entire page. They can be very challenging to use, especially for beginning web learners. And if they are overused, they can quickly become a nightmare.

[“position: fixed;” “An element with position: fixed; is [positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.” “This div element has position: fixed;” “div.fixed {[position: fixed; bottom: 0; right: 0; width: 300px; border: 3px solid #73AD21;}” “https://www.w3schools.com/css/tryit.asp?filename=trycss_position_fixed”]

Position fixed will fix an element in place so that it does not scroll with the page. *[A webpage is shown, with a navigation bar that does not move as you scroll.]* This is popular for navigations or background images to remain in place as the user scrolls. But again, it would not address the needs of laying out the entire page.

[A picture is shown with 12 numbered boxes. “The flex-wrap Property” “The ‘flex-wrap: wrap;’ specifies that the flex items will wrap if necessary:” “.flex-container {display: flex; flex-wrap: wrap;}” “https://www.w3schools.com/css/css3_flexbox_container.asp”]

Flexbox is another layout method if you want your items to be in a row or column. It uses display property with a value of flex. The children of that element will lay out as rows or columns. What's great about flexbox is how responsive it can make your page.

[A video of a webpage is shown. It then shows the same numbered boxes, and as they adjust the size, the boxes rearrange as well.]

It's great for things like product galleries or photo galleries where the rows can start out as four across, and as the device gets smaller, they can go to three and then two and then one; when you get down to a phone. It makes it really easy to have beautifully responsive areas on your page when there are a number of items grouped together. *[A diagram of a wepage is shown.]* I also love flexbox for the horizontal links in the navigational menu. It does a great job of spacing that links to give them a nice look, especially at one link is longer than the rest. But again, it's not the best for laying out an entire page. It works great on certain content, but not the whole page.

[A picture of an animated computer, phone, and tablet are shown. They each have gridlines.]

Grid is the method we'll use in this course to lay out our page. It's a lot like flexbox in many ways, but where Flexbox is one-dimensional, a row or a column, grid is two-dimensional, rows and columns at the same time. *[A wireframe of a webpage is shown.]* Grid makes it much easier to lay out a page without having to ever use float or positioning.

[Two webpages are shown. One has elements of the page in random order, and the other has them neatly arranged.]

Grid also allows you to move blocks around on the page regardless of the order that they are in, in HTML. You could still use Flexbox for certain parts of the page. Grid and flexbox can be used together. When you start taking items out of the natural flow, sometimes they can overlap on top of each other. So far, we've seen values that reference top, bottom, right, left, that have to do with a two-dimensional X and Y axis. But there is a z axis that is three-dimensional and has to do with which element will be on top of overlapping elements, or the stacking order. Perhaps you've used Word processors are other programs that allow you to bring items to the front or send them to the back. It's the same idea. It's called z-index. Z-index is the property name and the value you give it is a number. *[“h1{z-index: 1;}”]* The bigger the number, the more toward the front that element will be in the stacking order.

[End of video]