Aligning and Justifying the grid

[The screen reads: "Grid by Example. Aligning and justifying the grid". A Codepen is displayed. The display is shown on the left side and the HTML and CSS code is shown on the right side. Boxes are displayed within the display section. The HTML code is as follows:

*" *

* Box 1 *

* Box 2 *

* Box 3 *

* Box 4 *

* Box 5 *

* Box 6 *

* ". Only part of the CSS code is shown. The CSS code is as follows:*

".grid {

display: grid;

grid-template-columns: 150px 150px 150px;

grid-auto-rows: 100px;

grid-gap: 20px;

Width: 700px;

height: 500px;

border: 2px solid #666;

}

.one {

grid-column: 1 / 3;

grid-row: 1 /3;

}".]

Rachel: In a previous video, I showed how to align and justify the grid items. You can also align and justify the tracks themselves, assuming that you've got additional space in your grid container. To demonstrate this, I've got a grid here and I've actually set a width and a height on the grid container, so it's 700 pixels wide and 500 pixels tall, and then I've created some columns and I've got three 150-pixel columns, which obviously adds up to less than the 700 pixels of width and so the columns are set in the top left corner. The auto rows are also 100 pixels, so again, the combination of these rows is shorter than the height available and the border here shows the grid container.

To justify the tracks, we use justify-content and we can say end or center or the default, which is start.

[Rachel adds "justify-content" to the grid class in the CSS code. She then showcases the different examples. After she types "end", all of the boxes shift from the start of the container to the end horizontally. She then deletes "end" and replaces it with "center". The boxes then shift to the center horizontally. She then deletes "center" and replaces it with "start". The boxes then shift to their original state.]

And then on the block access, we can use align-content, and again, you could use end, and you could use center, or the default, which is start.

[Rachel adds "align-content" to the grid class in the CSS code. She then showcases the different examples. After she types "end", all of the boxes shift from the start of the container to the end vertically. She then deletes "end" and replaces it with "center". The boxes then shift to the center vertically. She then deletes "center" and replaces it with "start". The boxes then shift to their original state.]

So if you wanted the tracks to be dead center, we just set both of these to center.

[Rachel changes "justify-content" and "align-content" to both be centered. The boxes shift to the very center of the container.]

So here, we're just pushing around our grid inside the container. If you've used Flexbox, you'll be aware that there's a different value for justify-content, that of a space-between or space-around, these keyword values that allow you with a Flex container to space the items out equally. You can use this with grid layout as well. So we can say space-between and we can do the same for align.

[Rachel changes "justify-content" and "align-content" to be "space-between". The boxes are now all separated with space between them.]

You can see something interesting here. We've got a grid gap on our grid. We said grid gap, 20 pixels, but of course, as soon as you said space-between, it's that gap has ended up spreaded out so we've now got a much larger gap than the 20 pixels that we specified. And something else is happening. You can see that these boxes, these were all set to be 100 pixels tall with the auto rows, and of course this now is much more than the expected 100 pixels plus 100 pixels for this track and the 20 pixels of the gap because you've got this extra gap and this element spans over it. So that's something you'd need to be aware of. The actual containers are going to get larger. The items on the grid are going to get larger when you span over tracks if you're using a space-between, and you can see exactly the same happen with Box One, which spans column tracks. This is essentially the gap and it's spanning over that to span the two tracks, and we can also use space-around, just as you do with Flexbox.

[Rachel changes "justify-content" and "align-content" to be "space-around". The boxes are now all separated with space around them.]

And then you get an equal amount of space all around each item so on the ends you get this half-sized space and that's just the same as Flexbox except we're working in two dimensions.

[End of video.]