The loading attribute on an img element can be set to lazy to tell the browser not to fetch the image resource until it is needed (as in, when the user scrolls the image into view). As an additional benefit, lazy loaded elements will not load until the non-lazy elements are loaded - this means users with slow internet connections can view the content of your page without having to wait for the images to load.

Give your new img element a loading attribute set to lazy.

**Step 34**

Use the minmax function to make your columns responsive on any device. The minmax function takes two arguments, the first being the minimum value and the second being the maximum. These values could be a length, percentage, fr, or even a keyword like max-content.

Wrap each of your already defined values of the grid-template-columns property in a minmax function, using each value as the second argument. The first argument should be 2rem, min-content, and 2rem respectively.

grid-template-columns: minmax(2rem, 1fr) minmax(min-content, 94rem) minmax(2rem, 1fr);

# Step 35

To add space between rows in the grid layout, you can use the row-gap property. Give the main selector a row-gap property of 3rem.

# Step 36

Your magazine will have three primary sections. You already set the overall layout in the main rule, but you can adjust the placement in the child rules.

One option is the grid-column property, which is shorthand for grid-column-start and grid-column-end. The grid-column property tells the grid item which grid line to start and end at.

Create a .heading rule and set the grid-column property to 2 / 3. This will tell the .heading element to start at grid line 2 and end at grid line 3.

# Step 39

Now you can style the content of the .heading element with CSS Grid.

The CSS repeat() function is used to repeat a value, rather than writing it out manually, and is helpful for grid layouts. For example, setting the grid-template-columns property to repeat(20, 200px) would create 20 columns each 200px wide.

Give your .heading element a grid-template-columns property set to repeat(2, 1fr) to create two columns of equal width.

# Step 41

Remember that the grid-column property determines which columns an element starts and ends at. There may be times where you are unsure of how many columns your grid will have, but you want an element to stop at the last column. To do this, you can use -1 for the end column.

Create a .hero selector and give it a grid-column property set to 1 / -1. This will tell the element to span the full width of the grid.

# Step 44

Create an img selector and give it a width property set to 100%, and an object-fit property set to cover.

The object-fit property tells the browser how to position the element within its container. In this case, cover will set the image to fill the container, cropping as needed to avoid changing the aspect ratio.

# Step 53

Now the auto-placement algorithm will kick in when you add a new icon element. However, the algorithm defaults the new column width to be auto, which will not match your current columns.

You can override this with the grid-auto-columns property. Give the .social-icons selector a grid-auto-columns property set to 1fr.

# Step 56

Your .text element is not a CSS Grid, but you can create columns within an element without using Grid by using the column-width property.

Give your .text selector a column-width property set to 25rem.

# Step 58

The ::first-letter pseudo-selector allows you to target the first letter in the text content of an element.

Create a .first-paragraph::first-letter selector and set the font-size property to 6rem. Also give it a color property set to orangered to make it stand out.

# Step 74

The gap property is a shorthand way to set the value of column-gap and row-gap at the same time. If given one value, it sets the column-gap and row-gap both to that value. If given two values, it sets the row-gap to the first value and the column-gap to the second.

Give the .image-wrapper selector a gap property set to 2rem.

# Step 75

The place-items property can be used to set the align-items and justify-items values at the same time. The place-items property takes one or two values. If one value is provided, it is used for both the align-items and justify-items properties. If two values are provided, the first value is used for the align-items property and the second value is used for the justify-items property.

Give the .image-wrapper selector a place-items property set to center.