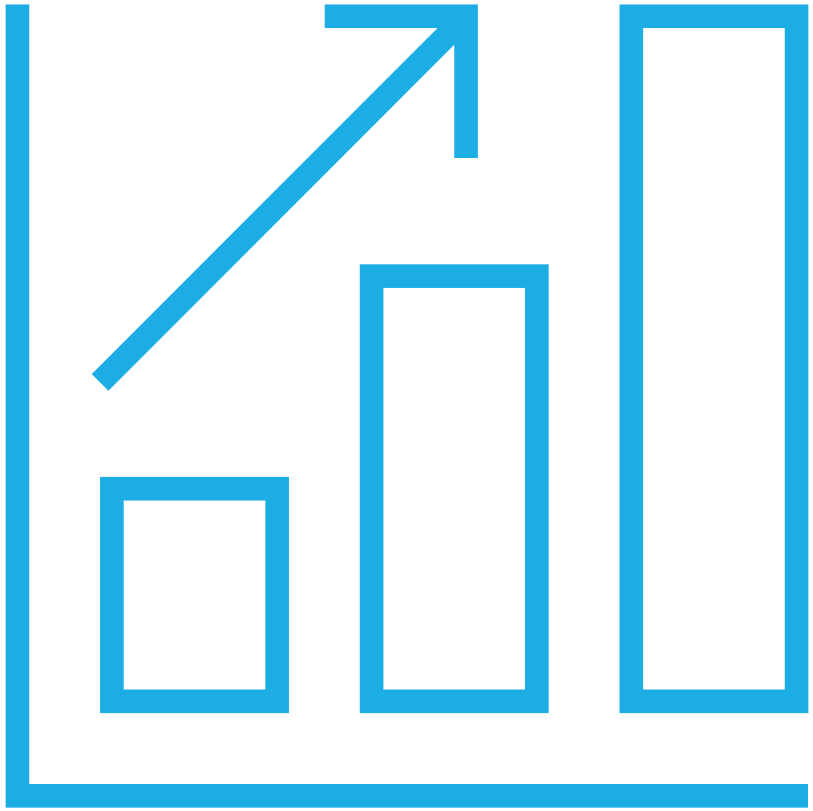
MORE GRID



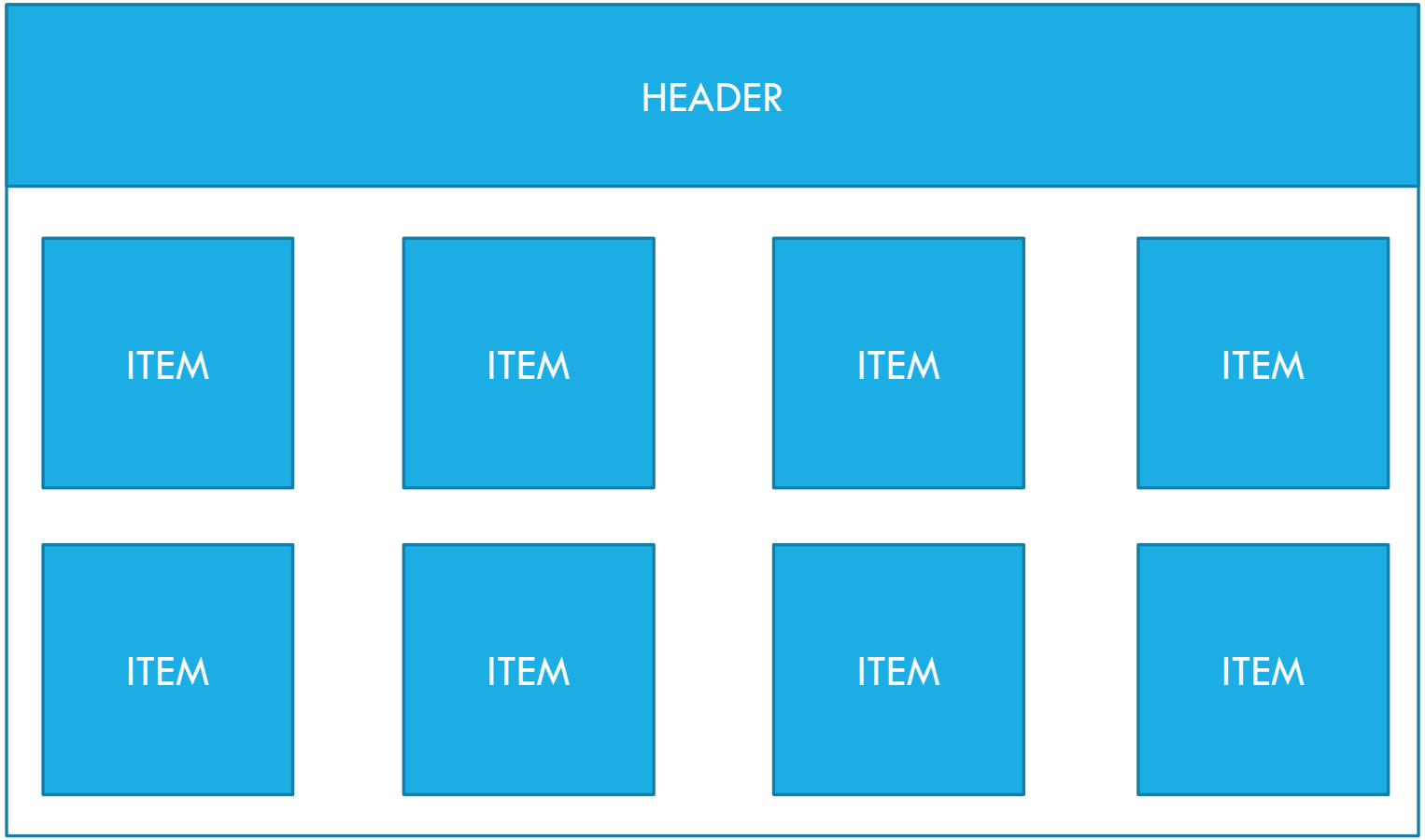
INTRODUCTION

We have some nice grid tools at our disposal now. We can imagine our layouts as grids and we can use CSS to create those grids on our pages.

While what we have is pretty awesome, grid has some extra tricks we can use to start entering the world of **responsive web design**. This is where our designs will start adapting to screen size changes and allow our sites to work on multiple devices.

ADAPTIVE GRIDS

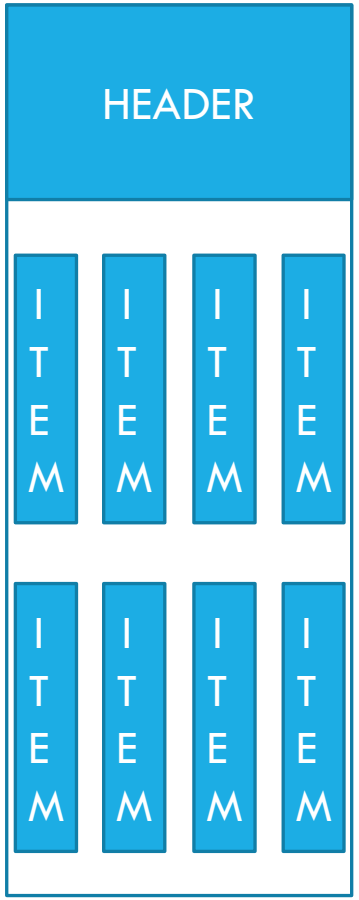
Let's imagine we are creating a site that is supposed to show many items for purchase to a customer. We want to achieve the following design:



The idea is fairly simple, four columns. What happens when we move to different sized screens?

ADAPTIVE GRIDS CONT.

When we move to a phone screen, our 4 columns are crunched up next to each other. This design is pretty much broken and will be impossible for users to navigate.



We need some way to tell our grid to be a bit more adaptive.

MAGIC

I am going to start by showing you the line of code that is going to make our grid much more responsive to screen size changes. We are going to see what it does, and then break down each individual piece.

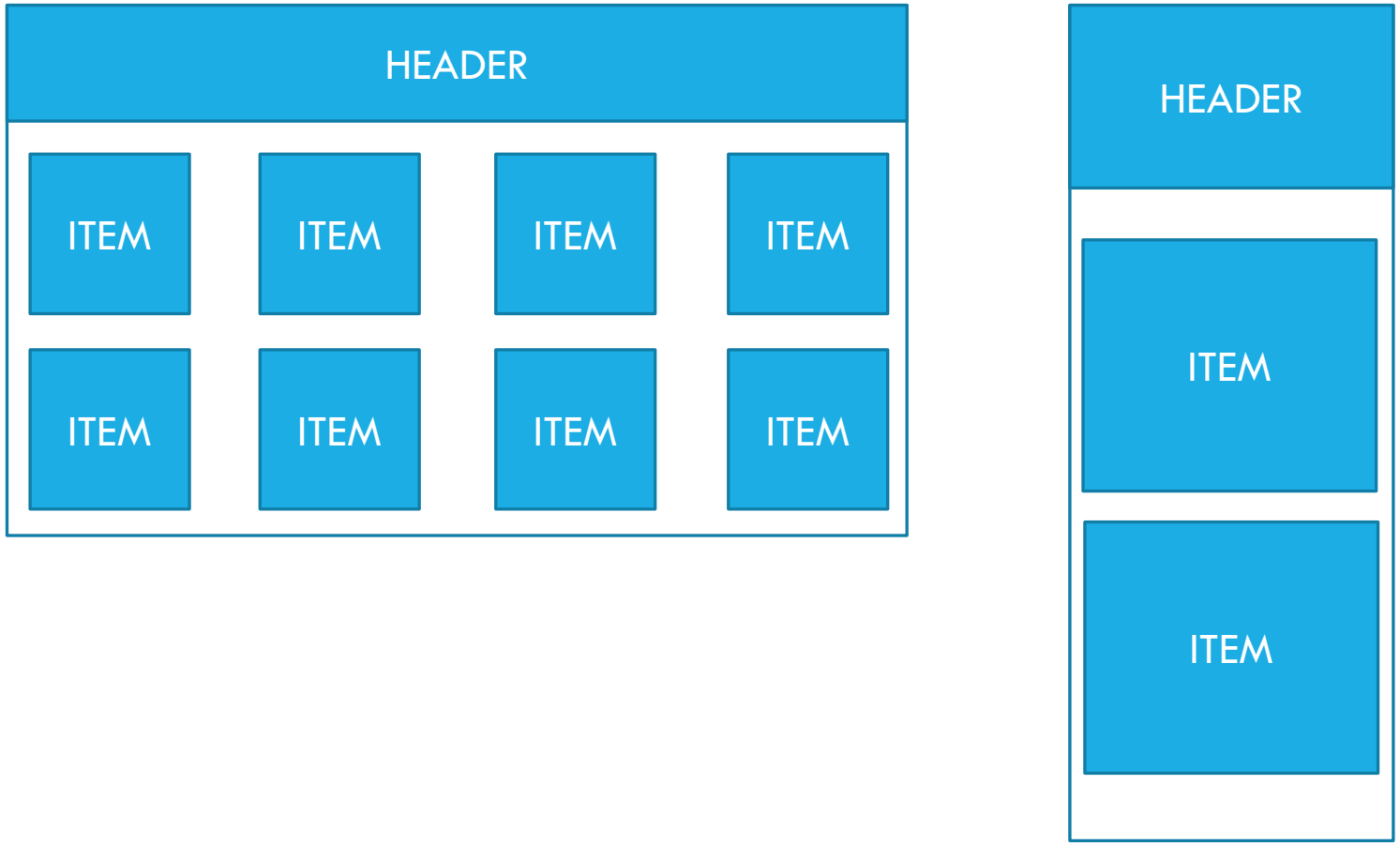
We change our column rule to be:

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

Just like magic, when we add this to our code we get a layout that adapts much better to screen sizes.

MAGIC CONT.

And just like that, our layout auto magically fits our grid items into much nicer looking layout patters. Let's break down what our CSS did there.



MAGIC EXPLAINED

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

I'm going to explain the above line in plain English first and then break down each component.

Our grid will now automatically **fit** as many 250px width columns into our container width as possible. If it can't fit a child into a column, it will put it into a new row automatically. Any remaining space is taken up by the children as they are allowed to grow and take up that extra width.

repeat: This is our first introduction to a CSS function. We need 2 things to make repeat work, the amount of times to repeat, and the thing to repeat. A simple example – **repeat(4, 1fr).**

auto-fit: This is a special CSS keyword that tells the grid to automatically fit as many children into a row as possible. When used with repeat, it will go and count the children for you.

minmax: This is another CSS function used in grid. It is designed to give an upper and lower bound for the size of something. A simple example would be - **minmax(200px, 400px)**

KNOWLEDGE CHECK

The repeat function can make your life much easier when it comes to making responsive and adaptive layouts. Even more basic, it can create good looking structures layouts.

1. Create a folder in Scratch called ResponsiveGrid
2. Connect this folder to Git/GitHub
3. Create an index.html and style.css
4. Add a section with 15 img tags inside of it
    1. Just use the same image over and over again
    2. Remember, images don't like not being told what height or width to be!
5. Use grid to make the section show the images at least 300px wide but stacking on top each other when no more room is available
6. Add commit and push your code.

AUTO-FIT VS AUTO-FILL

If you ever venture onto the internet looking at other things CSS grid, you might notice that you can also use auto-fill instead of the auto-fit term as the first **argument** to our **repeat function.**

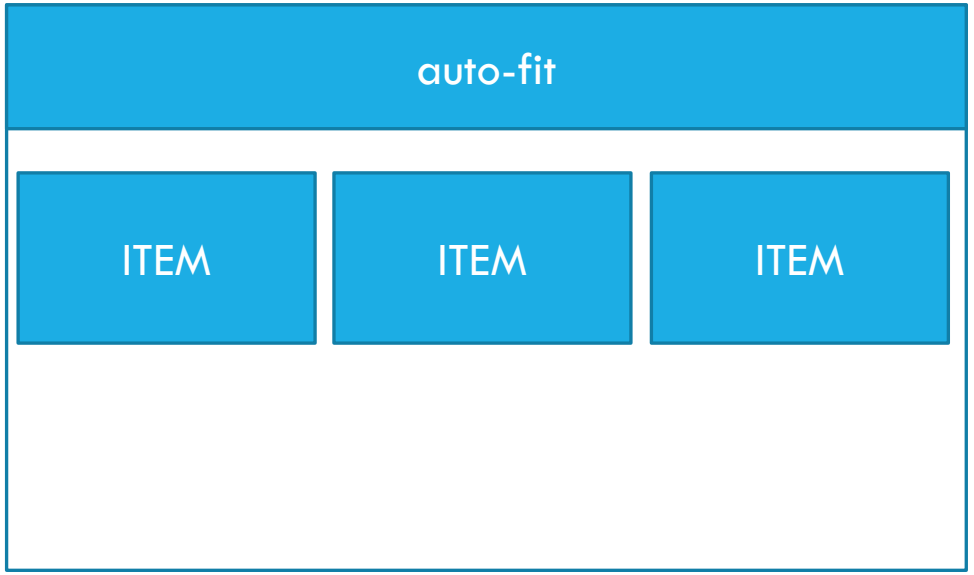
While these keywords are similar, they do have different impacts on how your layout adapts! This is best explained visually, but to give it a definition:

auto-fit will try to *fit* as many children columns into a row as possible. It will also allow for the expansion of children to take up extra space as needed.

auto-fill will try to *fill* a row with as many small columns as possible. This means it does not allow for the children to expand to take up screen space when there are no more children, it will place in empty columns.

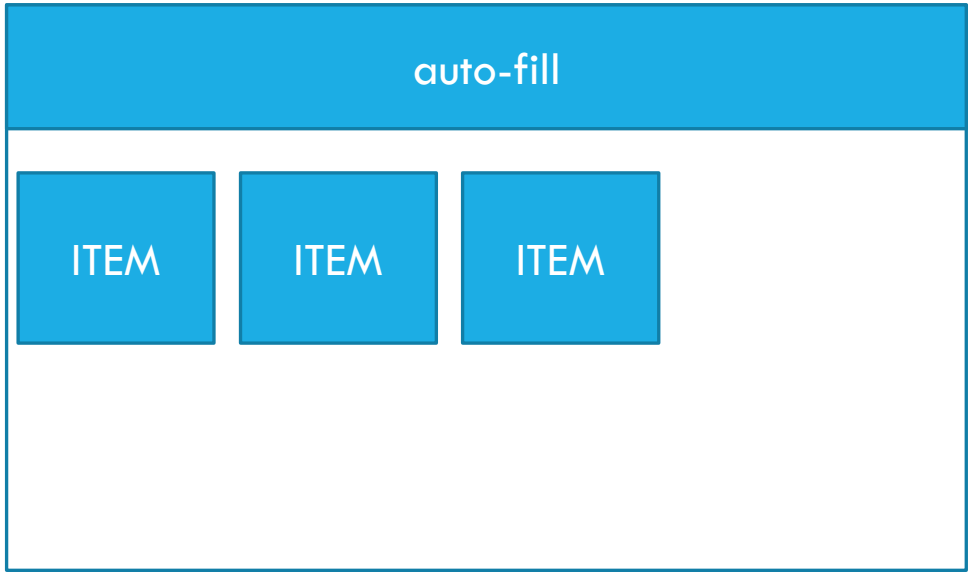
AUTO-FIT VS AUTO-FILL CONT.

Our auto-fit property allows the children to take up the full space of the grid container



AUTO-FIT VS AUTO-FILL CONT.

Our auto-fill property will fill the grid container with as many minimally sized columns as possible.



KNOWLEDGE CHECK

Let's make an even cooler page!

1. Modify your original code to no longer just be a section of images, but now a section filled with article tags.
    1. Move each image into it's own article tag
2. Add a p tag to each article after the image with random text
3. Turn each article into a grid.
    1. Each article will put the text beside the image if it has room, otherwise below.
4. Add commit and push your code.

MORE GRIDS

With this one tool added to our belt, we can truly start creating some adaptive layouts.

In our example in the slides we used a very simple layout, but you can use this property to totally transform your site as you move from screen size to screen size. While we will need to learn some more CSS to truly handle responsive design, this is a great start.