**Introduction to Scrum - 7 Minutes**

today this video is intended for people

who are just getting started with the

scrum framework and need a quick primer

hi my name is Steve stemman and I am the

founder of uzility software in this

video we will cover the basics behind

scrum and we'll take a look at how scrum

compares to Waterfall development and

we'll examine the three roles three

artifacts and three ceremonies that make

up

scrum first let's take a look at how

scrum compares to the older alternative

of waterfall

development waterfall typically goes

through a lengthy planning process which

could take several months followed by

building the product which again could

take many months and then testing the

product reviewing and eventually

deploying the product at this point you

may end up bringing the wrong product to

Market if market demand or technology

has changed since the original plan was

developed there are several problems

with this method first of all the

planning must be completed before any

work begins and in most cases the

planning is done without entirely

understanding the project once

development is being done often times

things get sent back to the planning

phase and the project either needs to

start over or the developers are just

criticized for not understanding the

plan this cycle can happen many times

when development is done building the

product it gets thrown over the fence to

test where when problems are encountered

it bounces back to development and

sometimes back to planning the same

issues occur in the next few steps with

lots of backstepping and doing over this

can lead to lag times and many months to

several years in order to get a product

out the

door with scrum and implementation of

agile the process is broken up into

smaller pieces first we do just enough

planning to get started with building

the minimal feature set we build what

was planned next we test and review that

small feature set and get it ready to

ship when that cycle is complete we end

up with a potentially shippable

product this process usually occurs in a

time period of 1 to 3 weeks this is then

repeated time and time again reducing

the time from planning to development to

testing each time through the planning

process we're doing just enough planning

to complete the next incremental release

you end up with several incremental

releases called Sprints a Sprint usually

takes from 1 to 3 weeks and you just

keep repeating these Sprints until your

product is feature

complete sometimes you may end up

shipping your product after the second

Sprint or the third or the fourth or

even further but you eventually end up

with a shipping

product

in scrum there are three key roles that

are needed for the framework to work

well first the product owner this is the

person responsible for defining the

features that are needed in the product

the product owner has the bright ideas

that turn into

products the scrum

Master is a servant leader to the team

responsible for protecting the team and

the process running the meetings and

keep keeping things going the team can

be made up of developers testers writers

and anyone else that helps in building

the

product team members often play multiple

roles some days developers may end up

doing test or testers may end up writing

either way the team Works to get the

product

done there are three artifacts or

documents that are used in scrum first

the product backlog this is where

product owners create a prioritized list

of features known as user stories that

could go into the product this list

evolves and changes priority with every

Sprint user stories are a way of

describing a feature set that follows

the as a user I need something so that

reason format this way of phrasing a

user story allows the product owner to

specify the right amount of detail for

the team to estimate the size of the

task the highest priority user stories

go into the Sprint backlog these get

estimated for size and are committed to

for the next Sprint burndown charts show

the progress during a Sprint on the

completion of tasks in the Sprint

backlog this chart should approach zero

points as the work is being

completed there are three ceremonies

that make up scrum think of these as

meetings or discussions Sprint planning

is where the product owner scrum master

and team meet to discuss the user

stories and estimate their relative

sizes the daily scrum is a brief standup

meeting where the team discusses what

they have completed since the previous

meeting what they're working on and

anything that might be blocked or need

help the Sprint review and retrospective

occurs at the end of the

Sprint this is where the team

demonstrates the completed work to the

product owner and then the team

discusses what they can do to improve

the process going

forward

let's bring it all together and take a

look at the scrum workflow start with

the product

backlog which is where the product owner

builds a list of the bright ideas and

features that could go into the product

the product owner prioritizes the list

and brings the top items to the

team Sprint planning is where the team

product owner and scrum Master discuss

the top priority user stories

determining what can go into the next

Sprint the output from the Sprint

planning meeting is the Sprint backlog

this is a list of user stories that have

been committed to for the next Sprint

the entire team and product owner have a

solid understanding of what each of the

user stories involves based on the

discussions from the Sprint planning

meetings the Sprint is a 1 to 3we time

box where the work committed to in the

Sprint backlog is worked on through to

completion during the Sprint the daily

scrum occurs as a standup meeting where

the team discusses what they have

completed and what they are working on

on as well as any blocked items the

outcome of the Sprint is a potentially

shippable product potentially shippable

means that the product owner can decide

if it is ready to ship or if there are

any additional features needed before it

ships at the end of the Sprint a Sprint

review and Sprint retrospective meeting

occurs the Sprint review is where the

team showcases their work to the product

owner and the retrospective is where the

team works on what they can do to

improve their

process repeat this workflow for each

Sprint now for a software solution to

help manage the workflow uzility has

been built around the scrum process to

help

people filling the three scrum roles

manage the three artifacts and better

run the three ceremonies visit uil.com

for a free trial and for additional

scrum

# Introduction To Responsive Web Design - HTML & CSS Tutorial

## Intro

building websites used to be simple, we had a single design, we had to make the site look exactly like that design. And then we were done. Now there's some problems with that

browsers used to render things differently. But at the end of the day, we just had this one layout to do, it was completely static, we just had to recreate what probably a Photoshop

document that we were given look like. And that was it. Now we have to take design, and we have to make it work on every device from sometimes a watch screen up to an 80 inch

television. That's not necessarily something that's super easy to do. But we have the fundamentals

down, we understand how to make a static site. So it's time to get into the responsive frame of mind. And we're going to work our way up from getting a design to simply work on a

single screen size, to be able to being able to get it to work on any device, thinking responsively isn't easy at first. But we are going to work our way up, we're going to be

exploring the do's and don'ts. And we're going to look at how we can plan things out to make our life a little bit easier in the long run. But do know, this is just an introduction

to thinking responsively we're not covering everything there is in this module about how to build a responsive website, we're getting our feet wet with it, we're gonna start looking

at a few new things. But I don't want to overwhelm. As I've said, before, I'd like teaching through

projects, we get a project, we're gonna learn the different things, we need to get that project to work. And then as we get into more complex projects, we can add on more complexity

to the CSS, so we're always adding on new things. We're not just dumping it all on it first. And this module is really going to set the stage for everything that's going

to come, we're going to look at how to approach a layout. So you see a late late when you when you see that late, what do you do? How do you start planning, what do you how your

system start moving forward? Before we jump into the layouts, that we're going to look at CSS units, because we've seen pixels so far, but we have a whole bunch of different

ones, we have ones that are absolute, we have relatives, and percentage or sort of relative units as well. And we're going to be looking at all of those, we're going to be looking

at the basics of Flexbox. And if you did the CSS Crash Course, you touched on Flexbox,

right at the end of it. But we didn't really get into it, we just saw what it makes columns. And that was it. So this one's not going to be a deep dive, I have a full module later

on in the course, we're going to do a deep dive into Flexbox to really make sure we understand the ins and outs of it. Because Flexbox is complex. And again, I don't want to throw

all this complexity at you and overwhelm you, here's how we can start using Flexbox to just make things start working. And once you are comfortable being able to do that, later on,

we've had a lot of practice with it, we can up the complexity on it a little bit, and also the basics of media queries. So media queries are an essential tool in adapting

our site to different screen sizes, we're going to be seeing in this just how they can work and how we can start using them to make our websites work across different sizes,

we are going to work on a few different things. But this is the main thing we're going to be focusing on, which is a three page website that we're going to be doing and we're going

to be making it fully responsive, we're going to look at how we can go from the mobile view of that up to this full screen site just like this one. If you need a lot of fun, I'm really

looking forward to it. So let's jump into the lessons and start learning how we can start thinking responsively. Before we can really take a dive into making responsive

## 1. Starting to think responsively

websites, the first thing that we need to do is take a bit of a deeper dive into CSS units. Because there are many different types of units that we can use. We have absolute

units. And this is what we've already been using, we have relative units. And we also have percentage, which is a bit of a special case. So absolute units, you've already seen

these, this is the pixel, they're the easiest ones to understand, because they're a fixed size, and they're always the same size. We also have a whole bunch of other ones that

you don't see come up too often your points centimeters, millimeters inches. If you can think of a unit that exists in the real world, you can use that unit in your designs. Now

the thing that's weird about those ignoring pixels, we're just thinking like the point centimeter millimeter inch, if you put your like a ruler up on the screen, they're not

really going to correspond to it, you shouldn't really be using them. Unless you're doing print styles, you can make a stylesheet or a set of styles for a document to be printed.

In that case, that's fine. But the rest of the time, if you need an absolute fixed unit go with your pixel, we also have percentages. So percentages are mainly used for width.

And they're pretty easy to understand. Because it's a percentage that's relative to the parent. Now I said they're usually used for width, because on height, things get a little weird.

Sometimes it's relative to the height of the parent, sometimes it's actually relative to the width of the parents. But we're not setting heights very often, we're going to see some

use cases later on in this course where we will be setting heights. But for now we're going to be sticking with percentages for width, because that's usually where we're

going to be using them. And it makes it really easy both to do something that's going to be like 80% of the whole browser window. Or if it's children, sometimes you want to call

them so one of them is 60%. And one of them is 40% or 70% and 30%. And you know they're

taking up 100% total. So it's much easier than trying to calculate the actual pixel dimensions on things when you use percentages. The last type is relative units. And there

are two different types of relative units. There's units that are relative to font sizes,

and there's units that are relative to the viewport. So the viewport is your browser window. In this starting to think responsively we're not going to get into the viewport unit

we're going to focus on the one ones that are relative to font size, because they're used so much more. The viewport units have special cases, they are definitely used in

some situations, and we are going to look at them in this course, just not in this module. Now, the idea of having units that are relative to font size, it might sound a little bit

complicated, but once you start playing around with them, it's really not that bad. So as I said, we're only going to be paying attention to the ones that are relative to font size

for now, and not that the viewport units. So the two units we're going to be seeing are M and REM. There are actually other ones as well, you have a CH unit and some others,

but we're not going to get into them. For now we're going to focus on the M and the REM, because these are the ones that are used the most often. The viewport units, just you

know them, we have the VW VHV min V max, they stand for the viewport width and height, and then the viewport minimum or maximum, which those ones are weird, and they they have specific

use cases. But the VW and VH are pretty handy. We're gonna as I said, we're not going to get to them in this module, though. So that's it for our quick introduction into the different

units. In the next one, we're going to get our hands dirty with a little bit of code. And since we've already been using pixels, and which is an absolute unit a lot, we're

## 2. CSS Units

going to take a dive in the next video and look at how percentages work and get our hands dirty with a layout. So we're gonna start with percentages instead of m and m just because

it's much easier to understand how they work. And so if we look at a layout like this, when

we will be making this layout in the next module of the course, once we've learned how to think responsibly, because we're going to make it responsive, we're going to make

this work on small screens up to big screens. And that's going to be our project in the next one. But for now just looking at this and thinking about it with what we know at

the moment, I would have a set width on the total size of this. So say I say the entire document here is 960 pixels. So I set a width, I put a container on the whole thing or something

and they say width 960 pixels. So that's fine. And this is a common size we used to use in the really, really old days. And then I have the two parts here. So I need to define an

individual width for the left and the right side. So I might say, well, I want that left side to be about 70% of that. So I need to calculate what that is. So I do 960 times

point seven. And that would give me 672 pixels. So now I know the width and height, set the

width of 672 pixels for my left side. But then there's also the other side, and I have to find out the fixed pixel value for that side. And all then I also have padding in

there. So then I have to start subtracting padding and doing all this. It's annoying, right? You don't want to have to start thinking about all of these things. And this is for

a fixed width. Imagine now you're trying to bring all those numbers into it for you're trying to like adjust it for different screen sizes. So instead of complicating matters,

when we think responsibly, we want to simplify matters a big part of thinking responsively. And I sorry, if I ever say respond responsibly, they're so similar. So if we when we start

to think responsively, it's all about trying to simplify and not complicate matters. So you remember this layout, we just finished it. But let's say we wanted to turn this into

a more responsive layout, instead of the fixed layout like we have. Now, we can do that.

So let's go and take a look at how we can do it. And you can probably guess the one thing that we really need to change is this fixed container width, because right now it

has a width of 620 pixels. So we have a set width on that container. I'm also going to turn this border back on, just so we can really visualize what's happening. While we're playing

with this, I'm gonna save that for the moment, we're gonna come and take a look at it. And you can see when I play with the screen size, we're set at 620 pixels. So it's working at

big screens. But if we go smaller, all of a sudden, I run into this problem where I have side scrolling. So I get a scroll bar to go left and right. Because my screen is

too small for the content that's in there, my content is wider than my viewport width. And that causes the side scrolling issue. So if I was going to open this page on a cell

phone, that would it just wouldn't work, right, I wouldn't get this issue, I wouldn't be able to see it, I get this weird problem there that I'm not even going to worry about right

now. It just does not look good. So how can we fix it, as I mentioned, we want to change

this from a set width to a percentage. So what I want you to do right now is change this over to a percentage, it has to be 100% or less, and I would go with less than 100%,

it would defeat the purpose of the container. If not, try different sizes, see what they look like play around with the screen size in the player here. And then once you're happy,

and you sort of see what's going on, I'm going to dive in and we'll look into it in a little bit more detail. Great, so um, you know, it's pretty evident, let's say I do 50% on this,

## 3. CSS Units - Percentage

I think it's obvious what's going to happen is those containers are now 50% of the total screen size, you might have got a little surprise with your image sticking off the side. And

we'll get to that in a little bit. But we can see with the container. Now as I grow or shrink my screen. That container is adjusting itself automatically, which is pretty cool,

I think. Because we know the containers with this set to 50%. I know I have 25% on the

left there and 25% on the right leftover. So that's pretty cool that I can sort of make this container that adapts to my screen size. Now the problem is my image. So before we

see how we can actually fix that thing, that's going With the image and understanding that what I really want you to understand is what is this 50% of. So if my container is 50%,

it's always going to be 50% of its parent. So we have a few different containers here. As we can see, this container here is 50% of section one, this one is 50% of my header,

and this one is 50% of section two. Now, my header, my section one, and my section two

don't have widths on them. And block level elements, by default have a width of 100%.

So section one, Section two, and my header are all 100% of my screen size. So that just means that my container ends up being 50% of my screen size. Now what would happen if

I came and set a width of 500 pixels on my header? Can you take a guess, before you come

in, look at what's actually happening on the page, what do you think has happened to the container that was inside my header. So let's go and take a look at it. And you can see

that I've limited my header to 500 pixels, and this container is 50% of that 500. So

I had 25% there, 50% here, and then another 25% there. So this container is now smaller

than this container on this side. Because each one is only looking at its parent, and it's not looking at anything else, then normally, I'm not gonna have a set width on something,

because that makes that whole thing look pretty ugly. So we're gonna get rid of that and go back and take a look at it. And there we go, we fixed that. So just remember, always, when

you're using a percentage, if that percentage is always relative to its parent, but a lot of the time the parent is defaulting to 100%, we're going to run into situations where maybe

that's a little bit different. Even if you feel a little bit confused right now, in the long run, it does actually make our lives much, much, much easier. I'm going to leave

this video for here. So you can play around with it a little bit, make sure you really understand how that's working. If you're a little bit unclear, try playing with the width

of your header of your section one in your section two to really see how this affects it and then maybe even play around with this width as a percentage, just to really make

sure it's clear. And then once you figure that out, in the next video, we're going to look at how we can fix that problem that's going on with the image. Okay, so we have

our containers that are set at 50% of our total screen size. But we have this really awkward problem that's going on with our image, our image just does not look very good when

it's sticking out the side like that. So how can we fix it? Can you think of a way that we could solve that problem? Is there something that we've looked at so far that would solve

it, because we have all the tools, we need to be able to fix that. So see if you can think about it coming to code, try playing around with something and see if you can solve

the width of the image. So hopefully, what you thought of is you can actually set a width

on your image. So if we take a look here, images are a little bit weird. And what you probably don't remember is when we first were looking at block and inline, I mentioned that

images are actually an inline element, they can flow with text, if I put an image in a paragraph, the text on that line will will actually flow with a put that image in the

paragraph itself, they're a bit of a special case, there's things that are a little bit different with them, but they are technically an inline element. So they're not defaulting

to 100% when they're defaulting to the size of themselves what that size of the image is. So what we can actually do is we can come in, and I like grouping it sort of here with

all the other things that I'm doing in the general area instead of the more specific

areas where I'm styling things. And I can select my images. So IMG. I'm not putting

a dot, I'm not putting anything else because I'm selecting my image itself. So I'm going to select my image, I'm gonna give this a width. Now this width for now, I'm going to

give this a width of 250 pixels. It's a set unit, we're not going to leave it there. But just to show you what that does. So look at that my my image size has shrunk down, the

one thing you have to be really careful with if you set a width on an image is not to also set a height to it. Because if I set a height on this image, which you can do, it will actually

end up stretching the image and distorting it and making it look really, really ugly. Whereas if you only put one or the other, so I could say a height, or I could say a

width, either one is going to work. And if I if I only have one of them, it's automatically going to handle the second property. So it's going to keep it in proportion instead of

stretching it. So you know now that we can set a width on an image, how do you think we can get that image to match the size of the container? So hopefully, you said use

a percentage. And what I want you to do is set it and see if you picked the right percentage try and set and actual size on this using percentage and see if it's fitting the way

you think it would. Alright, so there's two answers people usually think of one of them

is to give it a width of 100%. And the other one is a width of 50%. And the reason a lot of people think a 50% is because on my container, we gave that a width of 50% here. So if this

has a width of 50% it makes sense to give the image a width of 50%. But if you did that

You might have noticed it didn't work is pretty small. But if you shrink this, you'll see it's it is growing and shrinking with my page. So the percentage is working, just not what

we thought it was. So we just have to remember what I said before about how percentages work,

a percentage is always based on the width of the parent. So if we come in, we look in our index here, when we set the width of the container, this was 50%, of the width of section

## 4. Controlling the width of images

one, which automatically was the full size of my screen. So it ends up being 50% of my

screen width. This image, on the other hand, its parent is the container itself. So this

image is whatever percentage I set for a width and image will be a percentage of this container

width. So if we come into my styles, now, my image actually wanted to be 100%. So if

we go and look, again, we can see that it has been fixed. And now whatever I do here, my image is going to grow and shrink with the size of that container, and match everything

that's around it, which just makes the layout work much better. So if I come down and get rid of that magenta border, and we go and look at it again, we can see that it looks

nice and fancy, and it fits well. With the overall look of the document, the only thing

is a 50% width is a little bit small. So if I come down to this, everything is getting a little bit smushed, I'd rather that it's sticking closer to the sides. And I'm not

getting like this massive, it's almost acting like padding. You know, look how small that

images, I can barely see it. So it's really common to not have a width of at 50%. But

to do like a 90% or something like that, because on small screens, that's going to look a lot better. That's sort of that that cell phone type of look that you're probably used to

having and seeing. But as this gets bigger, we do at one point, now we're getting too

big lines of text going from one side all the way to the other is really, really awkward.

So we want to try and avoid having text that stretches from one side to the other. Like what's happening right now. And the next video, we're going to look at how we can do that.

We've got our layout working, it's looking nice and dandy, but it's getting too big. Now when we're getting up to these big sizes, we don't want that to happen. What we can

do to do that is a new property, we're not worrying about a new value. Right now we're thinking of a new property, it works with the width. And you probably saw the name of

this lesson was min and max width, so we can actually set a width and a maximum or minimum width on an element as well. So I'm going to come and put that border back on here of

two pixels solid magenta, just so we can really see what's happening once again. And let's

come and give this we have a width of 90%. But I'm also going to give it a max width

of 600 pixels. Let's do 620 because that was our original design. So we'll stick with the 620 that we originally had. So now if we come and take a look at this, and I refresh, you're

gonna see it's it's working in our small screen sizes, or containers growing, it's staying at 90%. But at one point, it's going to get locked into play. And right around here, it

stops growing, because it's hit its maximum width. And that means at large screen sizes,

my text and line lengths aren't going to get too long. But at small screens, everything works nice and perfectly. So we've sort of got the best of both worlds going now, we

can set a set value for the maximum size we want, where we can keep a percentage to let it shrink when we want it to, we can also set a min width. So if I came on here and

said min width, I'm going to set it pretty big of 500 pixels, I wouldn't normally do this on the container. But just for demonstration purposes, we have our max, so it's never going

to grow bigger than 620. And if I go this way, it will never get smaller than 500. So

when I hit 500 pixels, you can see it stops. And then I get sidescrolling again, which is why I wouldn't set a min width on my container. But it does demonstrate how that property

works. So for now, I'm going to take this off, I just wanted to introduce the idea of the minimum width in there. But for the max width, I will leave it on there, when we're

starting to think respond responsively for the container, this is a really really common property to set on something to give a maximum size, that size really depends on the layout

and what you're building. So I'm not going to say that there's a specific MAX SIZE that you should be going, you will base it on the design that you're doing, I will be going

more in depth some best practices for total length and total sizes in a little bit. Interestingly

enough, if you put your max width above your width, it's still going to work because there are two different values max width and width are two different ones. Same with min width,

you can put them in any order in your CSS and it won't have an effect, it's going to work no matter what. So play around with them, see if you can get used to it or figure it

out. If it's something that you're finding a little bit confusing, don't stress about it too much. We're gonna be using these a lot in the following projects that we're going

to be building, it's just gonna help reinforce it, the more we use it, so you'll get the hang of it in no time. So we're moving on to the M unit. But if you're wanting to get

more practice with percentages, don't worry. We're going to get a lot we're going to be using them throughout this course. But there's two others that we really need to look into.

So a quick recap on what relative units are, they call them relative units, because they're relative to something else, they're either relative to a font size, or to the size of

## 5. min-width and max-width

a viewport. So the M and the ram are both relative to the font size of other elements. So M's are always relative to their parents font size, and the font size is an inherited

property. So if you don't declare it anywhere, it's getting it from the size that you set on the body. And if you didn't set it, the body is actually inheriting it from the root

the HTML element, and that has the the 16 sit on it. Now that I say by default, his

people can override the font size in the browser in their settings. And so that can actually change what the default is on that property. But we're not going to worry about that at

all right now. But let's go and take a look at how they work. So I've kept the site that

we were already working on, but I've deleted some stuff. And we're gonna be playing around with this a little bit. So before we get into the unit itself, I want you to think way back,

and we're going to create a list. So remember how lists work, you can do either an ordered or unordered list, I really don't mind, see if you can remember how they are. But if you

don't, that's okay, it's been a long time since we've seen a list. But do your best to try and remember, before I go and put one in here and just give it like 234 elements,

make sure there's not just one list item in there, but there should be multiple list items in there. Alright, so I'm going to go ahead and create my own list, I'm going to do an

unordered list. So it's a ul, and I'm going to close that ul. And I'm going to come into here and just create a few list items. So open and close Li and I'm going to put in

a few here. And I'll fast forward while I put the content inside of these. Alright, so let's go and take a look quickly at what my list looks like there. So we have a bulleted

list since I used an unordered list. And we can start styling things up a little bit. So let's come over to my CSS file here. And what I'm going to do is I'm going to set my

URL to have a font size of one M. And when I do that, nothing's actually going to change

because one means copy the font size of the parent, now the parent doesn't have a font size on it. So it's going to go down and down and down until it finds something that does

have a font size on it. So the font size is being declared all the way up here on my body.

So the font size is 18 pixels here. So that means if I change this 18 pixels to say 25

pixels, everything stays the same. So my paragraphs got bigger, this got bigger because as an

inherited property, it's sort of the same as setting something as one m, you're saying match, one M is saying match the font size of the parent. So it won't actually have an

effect, where it will have an effect is and I'm going to bring my URL all the way to the top here, just so I don't have to scroll up and down. If I change this font size to 1.5,

that's the same as saying that this is so 1.5 m is equal to 150% of the font size of

the parent. So we can go and look at how that would work. So if I refresh, now, this is massive. It's now 150%, the size of my paragraphs here, which are the 25 pixel. So whatever

## 6. CSS Units - The em unit

I change on here, if I change this down to 10 pixels, this will now so it'd be 1.5. m

is 150%. So that means in this case, it would be equal to 15 pixels. So you can do a quick math to figure out exactly as you can see, this is super, super tiny, down here at the

bottom. Where is this list, at least it's at a readable size. So it's always a relative

unit. But as I said, it's always relative to its parent. So in this case, it's inside of section one. So if I said section one has a font size of 20 pixels, this font size is

now 30 pixels, because it's going to be 1.5 times bigger than the font size of its parent.

So it's always a font size that is relative to its parent. Now, you may be wondering why we'd want to do that. But imagine on these, let's get rid of this for the moment, I'm

going to comment it out because we're going to need that later. But let's say on my h1 here, instead of setting a font size of 36 pixels, I said this is three M and then I

said this one here is two M. And then I have my body which is set to say 16 pixels for

now. So if I come and take a look at that, everything is sized and it's all looking good.

But if I come and change this number, and I set this to 10 pixels, everything is going to adjust with that. So now all of those units have automatically adjusted in size. My headings

are smaller, my list is smaller, my paragraphs are smaller, my headings have shrunk, everything is reacting together instead of each being a completely separate unit. And this can be

really, really handy, where you can change one font size and have it affect your entire site. It's very handy. It's very useful, but there is a big problem with EMS. We're going

to see what that is in the next video. Before we get there though, if you want to just play around in here, try different sizes. See if you can get the hang of it play around maybe

with your h1 and h2 sizes, and then modifying this type Playing with apparent font sizes

if you want to, and just get a little bit of a handle of how it's working. Just like percentages, though, these are units we're going to be using a lot from now on, I just

want you to get a vague idea of how they're working before we dive in and start using them a lot. So even if it's a little bit weird, right now, it will be something you get used

to a lot faster than you might think. So as I mentioned, at the end of the last video,

there is a problem with the M unit, M's are super convenient. For some things, we're going to see some use cases where I use them all the time. But when we use them for font size,

it can create this weird cascading effect where things just get out of control. And it can be really, really mad. So if we come and look, I'm just continuing where we left

off, nothing is different from before, what I'm going to do is up here, I'm going to keep

my font size at 10 pixels. So it's really, really tiny on my body. But what I'm going to do on section one is I'm going to give section one, a font size to two M. So the

font size on my section one is twice as big as on the rest of it. But if we come and look

at what's actually happened is now this is an h2 and this is an h2. But this h2 is huge.

And why is it so much bigger, and this is size 10. This is not 1.5 times bigger than

that this is not 15 pixels, but I have my list here is set to 1.5 M. So why is it so

gigantic, I mean, what's going on. And to make matters even worse, why not? You know, you can make this even crazier. on my list items themselves, if I decided to set a font

size of 1.5 m just for fun, you know why not? All of a sudden, they're massive, you know,

this is getting out of control, literally getting out of control. And if you had other nested elements, it can get even worse and worse and worse, because it's looking at the

font size of the parent. So right now my list item is looking at the font size of my list,

and it's gonna be 1.5 times bigger than my list. So it's doing this multiplication through

all of these different elements until it finally you know, multiplies all throwing from 10

times two. So we're at 20 times 1.5, we're hitting 30. And then we're going to use 1.5.

So you know it's getting massive, it's just getting out of control big. That's also why

my section one has a font size of two M, my h2 has a font size of two M. So we're getting

that cascading effect even here. Which is why if we look here, my h2 at the top here

is a lot bigger than the h2 that's down here, even though their font size is set to the same value. And it's because it's this one is multiplying that two M by the 1.5 m to

get to this total gigantic size. If you're really good, and you're really know how to control this stuff, maybe you can do this on purpose and get away with it. But there's

a solution that just makes our lives so much easier. So I don't do this, I don't mess around with AMS for fun sighs so let's go and see what the solution is. Before I go into the

solution, which is the REM unit, I just don't want to poop on the M too much. It is a super useful unit. And after we've seen these two and we understand what a REM is, I'm going

to go into when we might want to choose which one and hopefully make all of it a lot more clear on why both of these are super useful and super important when we start thinking

responsively. So the REM is an M, but it's short for root m. And what does that mean?

Well, it means it's always instead of being relative to the parent like the M is it's always relative to the root of our document. And I've used the word route a few times.

## 7. The problem with ems

And the root of an HTML page is always the HTML element itself. That's where a lot of the defaults are coming from, even though we haven't actually been styling that in our

CSS up until now. So you may be wondering how does this make our life easier. And let's go and take a look at how. So remember right now we get these giant sizes coming in. Even

though this is set to two m and this is set to two M, this one is getting gigantic, it's getting out of control, because it's two m times 1.5 m, and it's scaling up and it's

a compounding factor that's going on. Whereas if we use root m, we lose that. So on my font

sizes here, I'm going to switch that to an REM instead of an M. And the same thing here. And the same thing here. And the same thing on my headings. And let's go and take a look

at it right away without doing anything else. And look at that this font size is now the same as this font size. And this is a lot bigger than this one. But that's because I

do have my font size set to one and a half REM here. So this font size will be bigger, I guess we'll get rid of that comment because it doesn't really apply anymore. And here

we used to have that double compounding factor that was going on my list item was compounding through this which was then compounding again through that. If I take this off now and we

take a look, it's going to look exactly the same because it's just set to that one size that's relative to the HTML element. But you remember that cool thing with me is when I

change the font size of my body and everything just worked around that and sort of scaled up and down with that one unit. So it's almost like EMS and rems Ramos a creating a scale

instead of creating these set font sizes. So what we can do is on our tml element, we

can set a font size on here. So I'm gonna say a font size of 10 pixels. And we go and

take a look at what that's done. Everything has shrunk down, everything on this page has gotten a lot smaller. And if I come on the HTML element here, and I make this 30 pixels,

everything will have gotten much, much, much bigger except for this paragraph, why hasn't that paragraph gotten bigger? What's going on? Can you figure it out? See if Remember,

the paragraph is inheriting its font size, the section two doesn't have a font size on it, nothing that is contained except for the body has it. And this is an absolute unit,

it's not relative to anything, we have this in the pixel. So this is where it's sitting, and it will not change. So we've, what we've managed to do. So if we switch this over from

that to one M, and M by one m, I do mean one REM for font size, I'm always going to stick

with rems. And you can see here what's happened is, everything is going to keep that nice scaling feature where everything is scaling up or down with something else without the

## 8. The Solution: Rems

possibility of running into that compounding issue. Now the one thing I wouldn't actually do is set a set font size on my HTML element like this. Usually, I'm just leaving it at

the default. Or I might use a percentage if I need to play with it. And you'll see some people recommend setting the percentage really low here to shrink things. If you set it to

62.5%. This makes the default, which was 16, now become 10. So when you instead of having

to think of one rim as 16 pixels, and then it's hard to do math, like what 16 times 1.5.

It's easy now because this is 10, and then that becomes 10. So this is 20. This is 15, it makes it much easier to think that way. I don't do that I you did for a little while,

because it is easier, I will admit, but after some time, you really do get used to working on the scale of 16 pixels. And when you need to scale this up, and you're gonna see when

we start getting into mobile, when we start doing different designs you're going to have on a mobile screen, you're probably gonna have a smaller font size. And when you get

to a bigger screen, you will up the font size, you can up the scale on it. But instead of using the percentage, maybe you just keep this at 100 as the default and then the bigger

size, you just bumped it up to 110 and everything or 120. And everything just scales up with

it. So I tend to just not even bother with this and I leave my rems like that and play

around with it. And I've gotten used to doing that. And that's how I'm going to be working throughout the rest of this course. So you will get a chance to see and I'm going to

explain myself every time I'm using these units, especially for the next little while, while we're doing them, including the next video where I'm going to explain that ends

still do have a purpose. And we're going to see what the purpose is, and how I decide between m and REM and what my rule of thumb for them is. Alright, so we've seen percentages,

we've seen EMS, we've seen RAMs, of course, we've seen pixels. So how do we decide which one to use, and in what situation if we go back to pixels, and when we're using them

a bit of a history lesson before we move forward, the pixels used to cause some pretty big problems because they were fixed unit but not in the same way like a centimeter is because the

centimeter is you take a measure and you can measure it, whereas a pixel was a.on your screen. And that's technically what a pixel really is. But in CSS, the pixel doesn't look

at the dots on the screen anymore. It now follows what we call the reference pixel. A reference pixel is a fixed size. And they had to come up with a solution because the

problem was we started getting these devices that were different resolutions. So imagine if you had a retina display and a regular display, and you set something to 18 pixels,

it looked fantastic on your normal screen. But on that Retina display, it would be half that size, and then just be really small. And so we couldn't set things in pixels at

all. So AMS and Rams verse, you know, we needed them pretty much. But the reference pixel, the idea here is they switched the pixel. So instead of following a doubt on the screen,

it's following a set size just like an inch or a centimeter would that size, I'm not going to go too much into it, it's pretty much 196 of an inch and the way they calculated as

the if you're an arm's length away from the screen on a 96 DPI screen. And I know it's

kind of complicated, it really doesn't matter for what we're up to right now. But it's just to say it's a fixed unit. It's not a.on your screen anymore. It sounds weird. And it sort

of is it's had a lot of benefit and a lot of really good things. But it also makes my answer to what unit we should use a little less concrete than I used to have. And I used

to always say you use EMS for this rems for this and you just avoid pixels. But you can get away with using pixels for a lot of things now, because the fact that a pixel if you

said something to 10 pixels on all the screens, it should look like it's the same size, regardless

of the resolution of that display. So a lot of those problems we had with pixels don't exist anymore. There are other advantages to using M and REM. So for font size, we're

going to stick with them some of those advantages we're going to see as we progress through this course so I am going to be sticking with them. And my general rule of thumb is for

font size, I'm going to use REM because it prevents that cascading awkwardness from happening.

And for padding and margin. I'm going to Use m, and we're going to see why in the next video, we're going to run through this, we're going to be using all of those. For width,

I'm either using M or percentage, I mean, you're going to use an M, when it's a set size, or I'm going to use a percentage, we might also use pixels in there to be honest,

it you can sort of get away with on width, so I'm not too concerned about which one you're going to be using. So if you're more comfortable with pixels on widths, that's fine. I tend

to do that a lot. But it is sort of I sort of like sticking with the same units whenever I'm using them. So that is my general rule of thumb. But those are a rule of thumb. And

they're not hard and fast rules. Sometimes I will set a font size and an M if I have a good reason for it, because I understand how they work. And sometimes I will set a

margin or padding using REM, because I want it to be a set size. And I don't want it to be relative to whatever I don't want it to be relative to that font size, I want to be

## 9. Picking which unit to use

relative to the root for one reason or another. This might all seem really theoretical and confusing and not super clear right now. So in the next video, we're going to see how

these work with an actual example. And hopefully, it just helps cement things a little bit. The next one is just more of an example. And then we're going to build out a layout. And

we're going to be using all of these. And we're going to sort of be cementing all the different concepts that we've covered up until now, in this how to think responsively module

that's going to be looking at an actual example, you might recognize this, I wanted to first

run through an example that we've seen before. So we're not focused too much on the markup, we're just focused on how the different units affect different things. And then we're going

to go through and actually build a full layout. In this, this is the original code that we sort of left off with when we did a lot of this in creating our this little card component

thing that we made together. So if we take a look at it now, one of the reasons I did

choose this is because there's button buttons in there and buttons are a great example for lots of things including setting up and using M's and rems, and when you might want to use

one for one thing and something else for the other. And what I'm going to do is I'm going to create a extra class that I can put on my buttons, it's going to shrink or grow the

font size, because it's really going to illustrate everything. So let's call it button big enough

that a font size of let's say 1.5 REM, which is going to give us a pretty big font size.

And then let's come on a button small. And let's give that one a font size of point seven,

five REM which will be pretty small. So now I should be able to put those font sizes on those two buttons. So let's come and do that. I'm going to give my first one here, the class

of button big. And the second one here, the class of button small. So this is sort of

my default button class. And then we're modifying it with these two modifiers, like we did with the colors before. So if we come and look at how that's affected things, you can see

it has grown in shrunk the font sizes on those. And that's awesome. But the button hasn't

really this button was forced to get bigger because the content inside of it do get bigger. So we do have the padding that's on these. The problem is on this big button, the text

is taking up a lot more room than it is on this small one, this button didn't really scale up when I change that font size. And let's let's bring in like a default button

here to actually I'll stick that one right in the middle. We'll just put default here.

And we won't have any modifier class on that just so we have a reference point of what the original button look like, we see our three button sizes. If you look at this button,

the space on this side looks a lot bigger than the space on the side it read through the exact same size. But the problem is my font size has gotten a lot bigger, my button

really should be scaling up with those sizes. So remember, I always said I put font sizes

in rim, it stops some weird stuff from happening. But I like putting my margins and my padding

using AMS. So if we come here where I had my padding of 15 pixels and 30 pixels, I'm

actually going to change this and I'm going to change my padding on this on the top and the bottom 2.5 M and on the left and the right to 1.25 M. And let's go and take a look at

my buttons. Now. Notice how the shape of them is much more similar one to the next, learn more about me, I'm going to change that just because it looks different. Because it's such

a long amount of text inside the button, it's going to keep the length of the text more similar in all of my buttons. And you can see it just looks like the button is getting

bigger and bigger or smaller and smaller. Because the padding around it is scaling up and down with the font size of that button. Now you might be wondering one thing, why

## 10. ems and rems - an example

is it scaling with the font size that I'm putting on the button itself, because this

it's not paying attention to the parent is paying attention to the font size of its own

element, right? Like this is my padding here is point five and 1.25 on the button. And

so if I look at that button, it's paying attention to the padding that or it's paying attention to the font size that I put on this class. So it's looking at this font size. This is

where I mentioned that it's a little bit weird when you're using EMS, because with EMS, if

you're doing it on the font size property, it's looking at the parent if you do it on Any other property, it's no longer looking at the parent, it's M is relative to the font

of this element should put, say, font size. So m becomes relative to the font size of

that element of this element here, when it's put on your padding, or when it's put on your margins. Or even if you did it on like a width or height or something like that, you would

be looking at this one size rather than the parents font size. This is where scaling and

this compounding factor is good because it won't compound out of control, it's going to compound or scale according to what's here. Whereas if I use rems, for this, it becomes

a set value just like when I use pixels. So if I come back and I set those in REM, this

space is exactly the same as this space, which is exactly the same as this space, these two sizes are exactly the same, but because the font size is bigger, this looks a lot smaller

visually than it does on this one. Let's switch those back to em just so we can see it again. And now we can see that it's more of this like scaling up that's going on rather than

this set size around the text of my button. So that is how I choose between M's and rems,

when it comes to setting things and why I'm putting M's for margins and paddings. And

why I'm setting rems, for my font sizes, the font sizes, it stops that weird cascading from happening AMS because I can take advantage and make it relative to the font size that's

in there. This also makes it really easy when I'm sitting margins on paragraphs or on heading

because if I come and I go, I want to add I want to change the margin bottom on my h1.

I know my font size is 24 pixels. So if I came and I said margin bottom is one m, I

know now that my margin bottom is exactly 24 pixels. So that would literally be 24 pixels.

For my margin bottom. If I came in I said margin bottom is two M, well, then my margin

bottom would actually be 48 pixels, and so on and so forth. So it's relative to the font

size of that actual element. Now, obviously, and now I've said I don't like setting pixels here. So you do have to think about it a little bit. But that also means if I come in, I set

my font size here to three REM and you don't know exactly what now we've got a nice big font size on that. But we don't know exactly what three REM is, well, it doesn't really

matter, I don't have to worry about exactly how big that is, I can come and look at my font size and I go I want the space underneath, we will have that I can come and say margin.

Bottom is point five m because it's point five of whatever this font size is. So now

my space is about half of my font size, where I go, I want it to be bigger than my font size, I do two M and now my space after is twice whatever my font size was. So it's always

relative to that element itself, which makes it really, really easy when you're dealing

with your typography. And the big advantages. When I change my font size, this will adapt automatically. And I won't have to change both of them. Because when we start changing

font sizes, which we will be doing for different screens, we're gonna have a small screen and a big screen, the font size has to change on it. So if you're changing the font size,

it's nice that you don't have to go and change your margins. Because they've all been set in em the same way here, when I'm changing my font sizes, I don't have to go and change

the padding for those sizes, then of course there are percentages and in general percentages, I will be using for the width, you could technically use them on a font size, and they'll work

a lot like m but that also runs into that scaling and compounding issue. And the same reason I don't like to use M's on font sizes is the same reason I wouldn't use a percentage

on a font size, the two of them work extremely similar to one another. So REM is sort of the safe bet there now in we're going to be jumping back into Flexbox. And actually building

out layouts and practicing and using all of these things that we have learned up until now plus going back into Flexbox, which we haven't seen in a really long time since pretty

much the crash course and we're going to be looking at understanding it and how it works a little bit better as we learn to start thinking responsibly. Alright, so let's get into responsive

layout basics using Flexbox. before we actually start making the layout, which we will be doing in this video, just a refresher on Flexbox. So elements normally have a display of block

or display of inline as their default from the browser. So display block would be are things like our divs, or header or footer, or main are each one all the way to h6, our

paragraphs are lists and list items are also included in that. And we have our inline elements like are a strong m n span. So those ones stay in the flow with the items that are around

them. We are going to be using divs for our columns. And we can't you know because divs

are block they're going 100% of the width. But they're also even if we change that even if we say they're 10% width, they're still going to stack one on top of each other. We

can change this behavior by setting the display property to flex on the parent element. When

we do that it changes the behavior of all of its direct children. So if we have something

that looks like this. So this is a a section of content. And then inside that section, we have four divs, like this, if we set display flex on that parent section, the items inside

of it will turn into columns automatically. So they'll switch and they won't stack, they'll

also no longer be 100% went, they're going to shrink to fit the content that's inside of them on the horizontal axis. So it's very, very different. But it's also very easy to

do. So we're going to practice with this layout here. It's not the most beautiful layout, but it's going to bring us into being able to do some fantastic stuff. Because if you

see here, we are looking at how we can do this for a big screen. But we're also going to get to the point where we can make this work on a small screen as well, we want it

to work on both. There's a lot of interesting things that are in this layout. So we have content and stretching the full size. Here we have three columns, here, we have two columns,

these two have different colored backgrounds on them, this one is bigger than those two. And this one's even bigger than that. So how can we do this? How can we set it all up to

be the right sizes, and to get spacing between them and get all these different things that we need in here. So we're gonna be doing all of this not in this video, but over the course

of the next few videos. But we're going to sort of set the stage with the HTML in this one, we're going to be ignoring the mobile layout for the moment, we're going to focus

on how we're going to set up our columns and organize it. Because this is the most important thing now is how is this content organized, the hardest thing to do with a layout is when

you look at it, just breaking it down in how you're going to organize it, where do you need to do Where are all the different pieces going to fall? It's not super easy. At the

very beginning, the very first thing is, all of this is being held in the middle of the screen with a certain size on it. In the last module, we had three containers because we

needed backgrounds that were full size in this situation, we don't need that. So we can have one single container for all of the content on our page. And that container is

now this light pink background. But what we also need is we need to create some columns. So I'm going to put a div of columns here. And I'm gonna have a div of columns here,

that's going to have these. So once we have those columns in place, then we need to create individual the individual columns. So I have this here is one columns. And then another,

## 11. Flexbox refresher and setting up some HTML

you'll also see these often and there's a popular framework called bootstrap, that will say row. So that's a row of content with three columns inside of it. So we have our you know,

we have a row of content or my columns container, I'm going to call it columns, because it's

my columns container. And then inside my columns container, I have three individual columns.

And then over here, I have this big one that's gonna have display flex, and then it's gonna have two of them inside of it. So this is sort of the structure of my site when I'm

breaking it down visually. The reason I know that I need to have these green boxes, and

then for each individual column, is because if not, if we didn't have those, and I just

put my all my age twos and my paragraphs directly in the columns parent, then each individual

paragraph and each h2 would become its own column, they would each break off into their own things they wouldn't keep, they wouldn't stay organized. So when we look at something

like that, we have to visualize where each individual column is, so we can organize it properly. And in this case, this is how I'm going to be organizing it. So I'm going to

leave this here, and I'm going to see if you can go ahead and do it, I've already set up the file, I've linked it to a style sheet that has nothing in it, I've given you all

the text. So just organizing it look at this picture, and go based on what you see here.

So go ahead and do it. If you don't want to copy and paste all the text at least make the structure like you've seen me do in previous videos, where you're putting in all the elements.

And then I'm going to go through and do it as well. And I'll bring the text and so if you don't want to bother copying the text back and forth, you don't have to. But if

you want to go through the full length, by all means go for it. I'm only going to do that in this video. In the next one, we're going to start applying the CSS to it. But

if you want to jump ahead and just try and do the whole thing on your own, go for it. In the next video though, I am sharing the colors, the exact colors that I'm going to

be using and the font sizes. But if you just want to estimate and try your best to get it to work based on what you've learned so far, go for it by all means. But if you're

not too sure, on the Flexbox thing, it's been a little while you can wait for me. So let's jump into it right now. So here in my body, as I said, I'm going to start with a div with

a class of container, which is just going to hold all of my content. Now we could in

this case, actually set a width on the body, I tend to try and avoid doing stuff like that, because all of a sudden, somebody wants to come in and add like a new section to your

site that's actually full width. And then you go Oh man, I have to change. So much content

had to change on my markup. So I like having a container or a wrapper that's going to contain all of my content and not touch the width on my body. personal preference there. But

I like thinking ahead and thinking things are always changing. So I don't want to run into any problems in the long run. In my container, I need my h1 and I need my image. So that's

gonna have an SRC on it and an Alp on it. And then we can come down and start doing

the rest of it to need our two parent column containers. So that was it. What I said I would call columns. So div class equals columns, close div. And we're gonna have two of those.

So you can copy and paste that or just write it all out, depending on how fast you can type. And there we go, we have the basic structure. Now inside of this column, I'm going to have

three actual columns. So I'm going to do a div class equals just co L, because it's faster,

right. And we can close that div. And I'm going to take that, and we need three of them

inside of here. And I'm going to take that same thing here. And I'm going to have two of them over there. So that is sort of the basic like actual structure of my site. Right

now, the different pieces are in place, obviously, I need to put the content inside of those columns. But from a structural point of view, this is all of my layout, stuff like this

is my main layout items that I put into place. Now it's just adding the content to those

to the layout parts of my markup. So let's go through and do that really quickly. So

here, it would be an h2, followed by a paragraph, I have the same thing in the next one. Except

in this case, we actually end up with two paragraphs. And then my last one, we just have a single paragraph all on its own, which is the exact same thing we have here. And

we can copy this one and bring it down there. Because that is what we have, we have h2 paragraph,

this is an h2 with two paragraphs. This is just one big long paragraph, and h2 with a

paragraph and then our individual paragraph there. So now we not only have the structure

of it, but we have all of the content elements in place, we just have to place the content

actually in there, I'm not going to do that in this video. When you start the next video, all the content will already be there, all of the colors and font sizes will be on a

slide in that video. So if you want to jump to there, now, you'll be able to try and do all the styling, and all of the content will already be in place for you. It's time to

make the columns in this layout into actual columns instead of stacking one on top of each other. So as promised, I've put all the content in here and I want you to give it

a go. So we have everything we need in place, we have all my paragraphs, all my content is in here, we just need to actually get this to be columns instead of stacking one on top

of each other. So if you want to go ahead and go for it, just as a quick reminder, we

want these two parents to be the display flex, and then everything should more or less fall

into place a little bit. I've also given you the colors that we're going to be using and the font sizes. So I'm going to start this one off by putting in the columns. And then

I'll also throw in the colors for my text and setting up the font sizes. But first, I'm just gonna set up my columns to make sure they're working to make sure we understand

that, then I'm going to run through and set up all my colors except for the background color on these two, because I'm going to look at that in the next video as a quick refresher,

which is going to get us ready to set the different sizes on these two middle columns here. So you have the right starting point, do as much as you can on your own before you

watch me do it. See if you can get the whole thing done. Even if you get stuck anywhere, of course, come and see how I did it or when you're done, come and see if we did it in

the same way. Alright, so I'm gonna jump over to my CSS file here. And as I said, I'm going

to start with my container normally and start with my body. But let's just get the lay of stuff working first. So let's actually come up here and make a comment called layout.

And I'll come here and my container, I said, I want it to have a maximum width of 980 pixels.

Now if we left it like that, when we come and look at it, if we make our screen big enough, we're gonna see that it's, it's nothing's gonna grow outside of that space. So I could

set my max width of 980. And then I could come on here and set the margin of zero auto

to center it on the screen. And now it's looking pretty good. And this is one of those rare times because we don't have a background color that's going anywhere one side to the other.

That default margin on the body, maybe it's not such a bad thing, because it's preventing our text from touching the side there, it's getting a little bit too close. For me, I

like keeping things a little bit further off than that. So I am also going to give this a width of about 90%. So it will always keep a little bit more distance off the sides than

what we had before. It's not perfect, but I like that a little bit better. So I'm going to stick with that maybe even 95 we could get away with let's bump that up to 95. And

keep going. The more fun part and the part that you have been waiting for is coming onto my call and then on here, setting this up. So on my columns here, we do a display of

flex. And just like that, we should be able to come into here and see I have three columns.

And if I scroll down, I get my two columns there. Now it doesn't look exactly like the layout that we had because this one is much longer than those two. And it really depends

on just how the screen is and how it is set up. Plus this column is supposed to be wider than the other ones. So when we do that it should fix everything we want. Now you'll

notice there's something weird happening down here where it's sort of already falling into line, but this one is much Smaller are narrower than this one. So we do have some work to

## 12. Basic Styles and setting up the columns

do. Now we are having a big issue, which is with my image, it is causing some side scrolling. So the first thing I'm gonna ask you to do right now is go ahead and fix that issue with

the side scrolling. If you remember how I did it last time. Somebody come here, I'm

gonna stick under layout, because images are part of my layout most of the time, and I'm going to give this a max width of 100%. Now, you probably did a width of 100%, which is

perfectly fine. And what you want to be doing anyway, the only difference between giving

it a width of 100% or a maximum width of 100%, is it means my image can never get bigger than it was supposed to be. So if my max width here was actually like 1200, and I made this

much bigger than what we had before, my image one stop growing when it gets to its native

size of 980 pixels, whereas it will still shrink to fit the content. If my its area

is smaller than it read, the only reason I would do that is if an image goes beyond or bigger than it originally was, it will start losing a little bit of quality. So because

of that, I'm going to stick with a max width of 100%, instead of just having a width of 100%. And letting it grow. It's not too noticeable now. But if it really does get a lot bigger

than it was intended to be, it's gonna get really, really blurry and nasty looking. So I'm going to stick with a max width there. And we'll bring this down to 980. Just like

that. And we have our nice three columns that are coming into place. So that is nice and

done. So we are well on the way, the next thing I'm going to do is just to go through and fix up some of these colors, except I'm going to save these two background colors

for the next video, like I had already mentioned. So let's go and do that. I'm going to come up to the top here. And we're going to look at this slide really fast. And I'm just going

to be running through all of the things we see here. So I'm going to set the font size and my colors. And then that's it for now. So that's all in place when we jump into the

next one. So on the body itself, I'm going to come up here and we had the font size that I said would be 1.125 rim, the next thing we want is the color because the color throughout

the entire design, except for those areas that are on the white background and the h1 are using that 7070 color. So I'm going to set that right there, I'm also going to go

and put the margin to zero just because I tend to do that in every design, even this one where it's not essential to have. Now we can get into our type biography. And I'll

come into here and start writing a little bit. So my h1, which has a font size of three

rim, and also has that color of 3126 14, I'm going to leave the margin alone, I think for

this, in this case, I don't think we really need to change it. And I'm going to come on my h2 and give that font size of 1.5 rim. And I think that's all we really need to do.

Let's go and just take a quick look. And the one thing I did forget actually is that span in the middle. So let's set that up, I'm going to come into my index, and you want me to

do I'm going to give you a little mini challenge. I'm going to put a span here, but I'm not going to put a class on it. Now this isn't something I would normally do. But it's not

terribly uncommon actually to see this with spans. Because spans are always nested inside of an element. So a lot of the time that span will be nested in titles all the time. So

you have a span that's always in an h1 and h2 or h3, and that span is always doing the exact same thing. Instead of taking time to give it a class people will make a compound

selector. So using a compound selector, see if you can change the color of that, instead of giving it a class. So what we can do is my h1, then there we go. I spell it right.

And on that we can give that one the color to overwrite. So that would be B seven 832

F, just like that. And let's go take a look. And there we go, Oh, I just one last thing

that I forgot on there is just the text align center text align, center that I can put on

that as well. So everything is in place. In the next video, as I said, we're going to give these their dark background color. And then after that, we'll see how we can actually

get these columns to start behaving properly. Alright, so our layer is falling into place.

Now we want to add the background color to these two. So before we do that, I'm going to give you a hint. And I'm going to see if you can solve it if you haven't already. And

that is we're going to do this in a very similar way to how we did our buttons, where we have a main class that's controlling our button. And then we had a secondary class that we

would use to modify that to add colors or change the colors of our button. So with that hint to see if you can figure out how to add that background color and a little bit of

padding to those two divs. And once you get it or if you get stuck, I'll show you how

I would do it. Alright, so I'm gonna go back into my index to do this. And I'm going to

find the divs in question, which is this one is the first one we need because it's the third column inside of that first section there. And on that call, I'm going to give

this a call BG class. Now it could just be background. It could be pretty much anything

you want. The reason I'm starting it with the C Yo L is because I'm modifying a column. The specific reason for this is to act as a column modifier. So I'm putting called BG,

to modify my column, it just helps me stay a little bit more organized, could you do it in another way, 100%, you might have another name that you came up with, which was amazing.

And it probably is, there's not one naming convention that you have to use or follow. In fact, there's tons of really good and well thought out naming conventions out there,

that all look at things in a little bit of a different way. The important thing is that you do have a naming convention. And there's a little bit of logic to how you're doing

it, because then other people can figure it out. Or when you come back to a project six months later, you understand what you were doing. So I'm going to put called BG there.

And I'm going to come down here and do a call BG on this one as well. And then I'm going to come up into my styles where I set up my columns, I will do my call bg. And I'll give

that the background color that we need. So I'll write background, and let's get that dark color that we want. And let's give it a little bit of padding. So I'm gonna give

it padding of 8.5 M, because I don't want a lot. And as I said, I'm usually using M's for my padding. Again, I'm thinking my font size is 16 pixels, half of that sounds pretty

good. Maybe I want one, but I'm going to look at it and see what it actually looks like. And then I can decide if I thought if I think it looks alright, they're stuck together right

now, which is a little bit of an issue. So let's actually come on to my columns here and just give us a bit of a margin of like one m top and bottom and zero on the left

and right, just to help give us a little bit of space between things. And we can see that

the padding doesn't look too bad, probably a little bit small. Actually, because I'm looking at the two sides there, I can see that the top and bottom is about what I want.

But that's including that margin that's on my text by default. So let's go and boost that up to maybe a one at the end. And I think that looks a little bit better. So over here,

I'm going to take off the margin that's on that text, especially on the top. So we're just going to go into my typography section here and add in my paragraph and give that

a margin top of zero. And there we go, it is working this bottom one will end up with

a little bit of extra space. As you can see, actually right now they're stretching, we're going to look at how we can fix this stretching behavior eventually as well. And it will leave

## 13. Adding the background color

a little bit of extra space on the bottom. But it isn't the end of the world. Oh, and looking at this, this text is a little bit hard to read. So on that call BG anytime a

column has a background color, we can also have it come and get an actual color too. And we'll give it the FFF or just white keyword would be fine as well. So at least we can

read our text a little bit better, super. So our layout is coming together, it's working, it is responsive, it's changing in size, it will work in different screens. That's pretty

exciting, right? It's not set at one screen now. But our column widths are all screw with and we want to change those I did the background color first on these on purpose, because it's

the same technique that we're going to use to actually change the size of the other ones,

we're gonna use a modifier class that can control the size of our columns. So I'm going to try and go ahead and do that right now on this one, and whether you get it or not,

we'll see in the next video, how I am going to approach it, the one thing not to worry about yet is this spacing between them, it's just thinking about the logic behind it. And

then we can go through and worry about how to create those spaces, because we're actually going to see a new property to be able to do that. Alright, so let's fix the width on

our columns here to get them to be the sizes that we want them to be. So if you remember I said that I'm gonna use a modifier class once again. Now you don't have to necessarily

use a modifier class, but we're going to get really used to using modifier classes. So I want you to sort of get in that mindset. Now you originally I was going to look at

it was we can use our call, which would be this would just be one width, and we could

have a call two and a call three. But I'm going to set up the same idea. But I might have a call one call to call three and there could technically I guess be a call for that

would stretch the whole width. Now I am going to keep this slide. I didn't keep this slide here though, before I thought of that, because I did want to look at the idea of why I'm

calling it call two and why I'm calling it column three. And the way I'm looking at it is this is a four column layout I have, if I divided this into equal spaces, I have four

if I took the smallest column, I could have four of them across the entire width of this layout. So it's a four column layout. This one is double the width of this one, and this

is three times the width of my sort of default or my smallest column here. So I have my column

two, which is going to be double the size and this one which is going to be triple the size. So what I want you to do is if you didn't do it on the last video, when I said you have

a chance because you weren't really sure what to do, try giving it a go now that I've explained a little bit more, and then I'm going to look at sort of the trickiest thing was how do

you decide exactly how big they should be? Alright, so I hope you got it. So what I'm

gonna do now is I'm gonna come over here, and I'm going to come in actually to my index

here first, and we have to add those class columns to here. So this is a column and it's going to have the width of a column one. Then I'm going to have this one have the column

two and this one is also going Maybe a column one. Down here, I'm gonna have a column three.

And this last one will be a column one. So if I come over to my styles now, and I come

and create some styles for those, so we're gonna have a call one, a call to, and a call

three. And what unit do you think would make the most sense for setting the sizes on these?

Like, if you were going to do what's the easiest way you can decide the width that one of these would be? You know, pixels would be a nightmare? Just because you don't? How many pixels big?

Should it be the screens always changing. And then well, then it's not going to work at all the screen sizes, M's, and Rams Same deal. It just it's not going to function.

This is where percentages really come into play. So my call one, I can give this a width.

The question is, how big do I make it? Well, I just said when we're looking at that last slide, we're going to divide this up into four. So if I made this 25%, then I know that

this one should be a width of 50%. And I know that this one here should be a width of 75%.

So let's come and look and see how that worked on my layout. And look at that. Everything

is lining up. This is a call one and this is a call one. So now they're actually the same size as one another, they're working out the way we'd wanted them to work out before

they're matching each other. That's that's perfect right before this was bigger, but it was a lot bigger. Now everything is lining up with one another. The only problem we have

## 14. Setting the column widths

now is there's no space between our columns. And there's a lot of different ways we can approach this lack of space, a lot of popular ways are using margins and padding to create

space. But then this first column needs a different style, because you can't have any margin on the left or this one needs a different style because it can't have anything on the

right side. And then you know the same thing here, you start getting into a little bit of issues or problems with it. And luckily, Flexbox actually has a really good way to

solve this problem. We're gonna see that solution in the next video. Alright, so everything

in our layout is starting to come together. But the problem now is our columns are stuck together. So as I mentioned in the last video, we can fix this by adding some margin to them.

But then you have the problem, how much margin do you add, and then stuff doesn't really fit as well. So luckily, Flexbox has a way to make it a little bit easier. And it's something

called justified content. And this is a new property that we're going to be using. We've already seen when we have Flexbox. When we do display flex, this is what's happening,

right, the parent gets display flex, and we get columns. Now we can add a new thing called

justify content. And there's a few different properties on justify content, but one of them is called space between. And what that's going to do is it's going to take the extra

space that we had, and it's going to put it between all the elements. So we look here as an example, we had all this extra space that was left over on the left side. So when

we add this justify content space between it's taking that extra space, and it's putting it in between the columns, instead of leaving it over on the right side. So what I'm going

to do is first I'm going to exaggerate that a lot, where I'm going to change all these numbers, but let's make them like pretty small, I'm gonna say it's 1020, and 30%. And as you

can see, we get all this leftover space and it's just floating over here on the side of our page, we get all this empty space like that. So what I can do is on where I have

display flex, so it's always on the parent in the same place that you declared display flex, we can then come down and you are just d phi d phi content. So it's just like that

two words, but they're hyphenated all properties, they'll always just like we've started seeing max width is max width with a hyphen in it. So justify content is the same. And then it

will be space between there are two other vowels you can use. And we're going to look at those in a second. But this is the one we're going to be focusing on for now, which

is space between. So now if we go and look at our design, we can see that we have this giant space between them because it's taken all that space that was left over on the right

side. And it's evenly distributed it between our columns now, which is awesome. Now in

this exact situation, it isn't the best, but it is pretty cool. Now the other two options you do have between other than space between your space around. So what space around is

going to do is it's also going to put the space on the left and the right space between make sure that the item that's farthest to the left and the item that's farthest to the

right stay all the way on the sides. We're a space around we'll put space around each item. Now the thing that's a little bit weird is the way evenly distributes that space,

it's taking the space and it's evenly putting some on the right side of our element here

and on the left side, and then it's putting some on the right side of this one and on the left side and the right and the left. So what that does is it does create this thing

where the right space and the left space are always smaller than the spaces in between,

because it's doubled. Since this one you're getting equal spacing on each side of each element. So these get bigger spaces than on the two ends. So more recently, they've come

up with one that to fix that problem. And that one is spaced evenly. And what space

evenly does is it tries to make it visually look like it's balanced. This despite width

display. than anything else, it will make all the spaces be even, instead of doing that

## 15. Spacing out the columns

weird space around things, so space seemingly is a really nice solution. But for now we're going to stick with space between and it's probably the one that you're going to be using

the most often. So what we can do now is we can fix the actual width we had on here. Now Originally, we had this as 2550, and 75. But the problem with that is, that doesn't leave

any space for our space between to redistribute because when we add all of those up, we're

always hitting 100% 25 and 25 is 50 plus 50, is 100. And if we come and look down at the

bottom, we have 75 plus 20 fives, we have 100% space there as well. And so for simplicity,

now, I'm just going to reduce a lot, we're gonna take 5% off each one of these, so this will become 45. And this one will become 70, we take 5% off on all of them, it's going

to leave us a whole bunch of extra space that this space between can now redistribute in between our different items. So if we go and take a look, we have our nice big gaps in

between all of them. Now, everything should be lining up pretty well. And it is working out nicely when we go on bigger or smaller screens. Now it's not 100% perfect, because

our padding is making these columns a little bit bigger than what they were. But for the purposes of what we're up to right now, I do think it is good enough, and it is working

really well. So I am really happy with it. In the last one, we saw how we can control

our columns on the horizontal and we can space things out on the horizontal axis, which is our main axis. In this video, we're going to be seeing the Align items property, which

works with Flexbox, just like justify content does. And what this does is it allows us to control our items on the vertical axis instead of the horizontal. So when we do our display

flex, we get all those different block level elements, which turned into columns and stacked next to each other when they become flex items. And one of the things they do when this happens

is they actually stretch the whole height of the div and the height of the div is generally controlled by the item which has the biggest height. So this a lot of time is actually

a good thing. It's something that we wanted in CSS forever, because it used to be almost impossible to match heights of different items. And now we can actually do it really easily.

But it's not something we always want to do. Sometimes we want things to align to the top. And so we can use align items, and we use the value of flex start to be able to do it

where they'll all shrink and be just dependent on the height of whatever is inside of them. And this is one of the ones you'll probably use the most often. The next one you can do

is Center, which is also really, really handy vertically centering in CSS used to be an absolute nightmare. And a lot of the time you want to be able to vertically center stuff,

you can do it with Flexbox, which is super nice. And if you want to align things all on the bottom, you can do it with flex. And there's also the stretch, which is the default.

And just if you ever need to reset and go back to the way things were if to overwrite something, there's also another one called baseline that I'm not really going to get

into too much. Now. It's the one that people don't use very often it deals with the text inside the elements. And it's a little bit weird, but there is the baseline one as well.

So to take a look at an actual layout, what I've done is I've taken the layout that we were working on before and they've eliminated a bunch of the content just so we can focus

on these three columns here, I put a blue border on our columns container just so we can see it. And I've put the pink border or the magenta border here on our columns, all

of these borders are just to help illustrate what's happening with all of it. So as I mentioned, when we do a display flex, everything turns into a column. And by default, they stretch

to match the height of their parent and the height of the parent. In this case, if you remember, any block level element, by default will be a height of zero, but grow to fit

the content that's inside of them. This is the one that has the biggest height on it. So our parent is growing to fit this middle container. And that means these other ones

are stretching to match its height. Sometimes that's a good thing. But it's kind of weird that this brown box on the right side is stretching to fit the height there, I find that really,

really awkward. So what I'm gonna do is I'm actually gonna put this on the side. So we can take a look at it while we're working and really see things live while I'm editing

some code here. So let's come down on to where we have display flex, and we have our justify

content. So again, that's for the main axis, or generally the horizontal, we can switch

that and we're gonna see that a little bit later on. But for now justify content is on the horizontal axis and then align items, we'll be dealing with the vertical axis, or

you're going to start hearing me say across axis a lot more. So up and down at the current moment. So what I can do on this is I can do a flex start. And when I do that, you'll

see here, everything has shrunk to fit the content of what's inside of it. And this is sort of the behavior that you might expect to be the default. And it's the one that you

## 16. Controlling the vertical position of flex items

as I said, you probably if you want to override the default, this is the one you'll probably use the most often. If I turn off the border, which is right here, you won't really see

you won't actually see a difference on those first two columns when I do it, it's just that brown column that's jumping to really long or shrinking down. In reality, this one

is doing it to it just because there's no background and there's no border on it. It doesn't actually make a big difference. So if you don't have any background colors or

any borders, you probably won't have to worry about This too much. But as soon as you have something that has a border has a background color has a shadow on it, which we'll be exploring

a bit later, all of those things will affect what's going on. So that's where you might want to bring in your flex start to make sure everything is shrinking down to where you

want it to be, we also saw that we have a flex end. So that's going to push everything to sit along the bottom might seem a little bit weird it is. But you might be wondering

when I would use it, not terribly often, probably, but it is something that could come up. And

then we have our center, which you'll use more often than you might think. And it can be really, really handy. As I also mentioned there is the baseline, you'll notice that

the way the baseline one works is it's taking this first line of text and it's making this

is a bigger font size than this one, it's making sure all the text on that first line is actually sitting on like there's this imaginary line that goes the whole way across under

the first line of text and every box. So if we actually came and just for one, change the h2 font size here, let's just double it up to three, it's pushed my brown box further

down. So everything is matching this. The first line is lorem ipsum. Right at the top,

the bottom of those letters is lined up with the bottom of this, the baseline is dealing with the bottom of the first line of the text, you probably won't use it very often. But

I figure I'll bring it up because you might run into it at one point or another. So that is how the Align items works. Let's go back to flex start. Now. The one thing that does

drive me a little nuts is its flex start flex and of course center and flex center. I don't know how they decided that or why they decided it. But that's what it is. And we do have

to live with it. For any module, the course is all about starting to think responsively.

And one of the most important things to building a responsive layout is understanding media queries. So we've been looking at this big screen layout. But we want to be able to make

this work at small screens. And to be able to do that we need to know how media queries work. So you might be asking what is a media query? Well, it's this feature in CSS that

lets us add new styles that target only specific conditions. So the basic syntax of a media query looks something like this. But there are a few other things that we do need to

include in here for it to actually do something. So the first one is the media type. And we

can also include media features. So media features are certain things about the media

type that we're looking at. So we can be very specific with the situations that we're trying to target to put in additional CSS, the media type itself, it lets us target different types

of media. As the name implies, we have a screens, we can target our screens specifically. So I can read at media screen, we also have print, and I said there was a third one. So the third

one is speech. Under media conditions. Those let us target specific conditions within that

media type be include width, so you can do something like that media, min width 600 pixels, orientation. So you can actually say like I want an orientation is landscape or portrait.

You can also look for specific features of that device like hover, can that user's primary input device actually hover? So where are they using a mouse? Or are they not you might

be on a tablet, and then the browser actually will know that you're not using a mouse and that you can't really hover. So you can actually put write specific CSS that targets only devices

where you have that or maybe where you don't have that ability. So that's really, really cool that we can do that. Both media types and conditions are optional. So you don't

have to include both all the time. But you do need to have one or the other. Because if you don't have a type or condition, it's kind of useless, then we're not targeting

anything. So we do want to always have one or the other. So for example, we can target only screens. So I can say at media screen and have some some styles there, or maybe

more commonly would be at media print. And you might be going well, why why would you target print, we're building a website. And you're right, but let's say you had a recipe

website, and you wanted to make it. So if somebody hit command p, they can actually print out the recipe directions really well, I'm sure you've also seen these recipe websites,

where it's like 10,000 words, before you actually see the recipe you can hold up, you can actually like literally make all of that through CSS disappear, and only have the recipe itself

show up. And you can format it perfectly to print on the page. And people would love you for that. So something really cool you could do through a media query and some CSS, we

can also choose only a condition such as the width of the viewport. So if I don't include the media type, and I just put the condition, it's actually defaulting to a media type of

all so this is gonna look at all media that have a minimum width of 960 pixels. In this case, the one thing that's really important is if you want to combine a type and the condition

with it, you actually have to use the word and so we would write something like this at media screen, and min width of 960 pixels really quickly. While we're here, you might

also see times where there's the only word in there, don't worry about it. It was for old browsers, we don't have to worry about it too much. These days, we can literally

## 17. Media Query basics

just write screen. Another example here would be screen and orientation portrait. For now

we're really going to be focused on the sizes so we're not going to worry too much about some other stuff. We're all going to take a look at orientation in this video as well

though, but for this course, the focus As on min and max width and making sure we understand

those, because those are the ones that we'll be using the most often. And let's start with a really, really, really basic example. And this is not a realistic example. But it's

the type of example that sort of hammers home exactly how these actually work. So let's give this a background that is pink. And we can take a look at it. Well, now my my website

has a pink background. And that's it. That's all right. But let's come in here and write

a media query. So I'm going to do at media, I'm going to do my opening, close parentheses.

And then I'm going to do my squiggly braces. So we come into here, and I'm going to use my min width, min width of 400 pixels, which is pretty small. But again, this is just a

really basic example. So if you remember, this is the same as saying all and minimum width of 400 pixels, we really don't need to bother with that. So we can just have our

media condition here of 400 pixels for all of our media types. And let's change our background

color. Now, one thing, I do this all the time, but I'll just write background here. But this

doesn't work, and then go whoops. The reason that doesn't work is because we need to tell

it well, what are we changing the background of so you literally have to come and write your selector again. And this looks a little bit weird when you're not used to it that

we have this like nesting inside of CSS and HTML, we're used to seeing nesting, but in CSS, it can look a little strange. Just have to get used to it, though, because that's

how it works. So we have my media query opening close curly braces, and then I have my body selector inside of there. And I can say that the background here is let's change it over

to purple. So now if I come and take a look at small screens, it will be pink. And then when they pass a certain width, it will be purple. So what's happening is this here is

saying from a minimum width of 400 pixels and bigger from that size and bigger because

it's starting at that minimum width. And we can do another one here, let's say an app media, minimum width 650 pixels. And we can say on this one, my body has a background

of orange. So let's go take a look again, let's go smaller, so it really small screens,

it's going to our default pink, that's just like, Well, when I say default pink, I just mean like the original style on my body of pink, then as we make it bigger, oh, we're

getting over to that minimum with the 400 pixels. And then as we keep getting bigger, we're going to hit there's my 650 pixels, and then it will stay that color for as long

as we're there. So that is the basics of how it works. Now one thing that's really important

here is, let's bring this comment down here and put 650. Because the order of these is

really, really important. And this goes back to a little bit the same way. If you remember when we were looking at the pseudo classes for links, a link, a hover a visited a active,

all of those, you could actually be in more than one of those states at a time. Because if you're clicking on something, it's active, but you're also hovering over it. So the order

that you put those selectors in changed, if what you actually wanted would would kick

in, right. So it's the same thing here. So if I move this one, up to here, it's just

always gonna be purple. So let's go take a look. It's purple. And it's purple, and it's

purple. And then it's pink, this completely gets skipped over. Because it's looking at this and it's saying from a minimum width of 650 pixels and bigger. But well, this condition

matches, you know, this is 400, and bigger. So 400, and bigger includes 650, and bigger.

So it goes here, but it goes Oh, well. It's the same as if I had here, I paragraphs should be red color, red. And then I said my paragraphs should be color. yellow, yellow is going to

win because yellow is second here. So if we come and look, my paragraphs are yellow, which we don't want to do. But it's the same concept that's coming here. This is true. And this

is true. So I'm going to take the second one. So the order is really, really important. Now one thing you can actually do, which we didn't see in the slides, you can combine

two different media conditions as well. So I could also write an ad here and say a max width of 649 pixels. So now we've this CSS will only work between 400 pixels, and 649

pixels. So now if we go and take a look, we will still get our purple, but we will also

get our orange afterwards. Because this no longer works. Once we get to above 649. max

width is always from zero Intel. This is a maximum width of. So that's also really, really

important. So from a minimum of 400 pixels, up to a width of 649 pixels. I'm going to

do one more I'm not going to delete this. I'm just going to comment it out. And I do want to look at this, but I'm gonna change the number in here because it's going to make

it a little bit easier to play with. So let's just bring this down to 500 pixels. So right

now we have the pink background which we'll switch over to orange when we hit that specific screen size. So pink over to orange, I said, we're going to focus on a min and max width,

but I just do want to look at orientation a little bit. So orientation. And I'm going to make this one landscape. So a landscape orientation when something is some when something

is wider than it is tall, and you have portrait, which is when it's taller than it is wide.

So if we have the orientation of landscape on there, we can see it's pink, it's pink, it's pink, oh, well, now we have it as orange, because you can see that it's wider than it

is tall. But that's fine, that little ground here. So here, you can see actually, when

I'm switching only the height and not the width, that it's actually switching between the two of them, because right now it's wider than it is tall. And now it's taller than

it is wide. So we can actually have CSS that looks at the viewport height and width, and

applies certain rules too, which can be really, really interesting. So it can also help when you're dealing with like tablets or phones. And when people are moving things around in

the orientation, you can even look at specific aspects ratios, there's a whole bunch of extra stuff that I didn't even really mention that I'll link to in the notes of this lesson,

it's on the MDM page, I don't really want you to take a deep dive into that, because it can seem awfully confusing. But I do want to make you aware that media queries, you

can do a whole bunch of stuff with it, it's actually really, really powerful. But I'd really encourage you to do now is to play around a little bit with what we have here.

And I'm going to take this one off, actually, and alternate this one back on and play around

with this, we're going to be using media queries a lot in the project we're working on from now on. But I'd really like you to play around with it a little bit to make sure that you're

comfortable with them, because they're super, super important to understand. And when you're feeling a bit more comfortable with it, you can jump into the next video where we're going

to be applying all of this into the project that we've been working on. Our layout is

already a little bit responsive, because when we grow it is working. But when we shrink it a will shrink with our site. But we run into this problem that we have it just getting

way too narrow at one point. And we do end up with a bit of side scrolling, just because the text doesn't have enough room to live and to work. So we want to make it fully responsive.

How can we do that? Right, we want to make it so at one point, these columns break and it turns into them stacking on top of each other instead. To do this, we actually we've

already started using display flex. And so when we use display flex, it creates columns. And that's because there's something called flex direction and it's set to row. This is

the default behavior. So when we say row width means that the parent is a row. So that means all the children become columns within that row, this is the default. So if you don't

declare this, it's just going to be there. So you don't have to declare it a flex direction row. It's just that's the way it's going to work. But we can switch that flex direction

from row two column, this switch is the main axis and that I'm going to be talking about the main axis a lot in the next video, we're just going to see a little bit of how we can

work for our layout in this one. But basically what it means is the parent has become a column, so all of the children inside of it have become rows. So if we have our display flex, it's

doing this where all the children have become columns. And when we add flex direction column,

it's making the parent now a column so the parent is going up and down and all of the children become rows. And you might say this looks a little bit like the default when we

don't even have display flex. And you'd be right. But there's reasons that we'd want to do this. So let's go and give it a try with our layout and see how it can work. We're

going to want to do this with a media query as well. So let's just come to here where we have our display flex. And as I said, the default on that is already a flex direction

of row. So we don't need to declare that because it's already there. If you want to, you always can. But we don't need to. But what I want to do is come and do a media query. So if

you're feeling really comfortable with media queries, and you're sort of got the hang of it, when you're playing around in the last video, I'd encourage you to go ahead now and

create a media query, we're going to target a screen size of about 600 pixels. And because we've already set it up for big screens, we're going to use max width. So remember, max width

is meaning from small Intel that maximum size. So for the small screens, I'm going to target

a max width of about 600 pixels, and we're going to switch the flex direction on it. So if you want to give that a try, you think you can do it, go ahead and do that in a few

seconds. I'll do it myself. Let's go and make my media query now. So remember, it is always

add media, then we always have our parentheses. And then we have our curly braces, I'm going to push return to give myself the space. But I'm going to come in here and set my max width.

So max width 600 pixels. And then we can come into here. Now one thing and I do this too,

and I've been writing media queries for a long time, is I'll just start writing the properties I want. So I'll come in here and do a flex direction. So if you did this and

it didn't work, it's because we have to choose our selector again. So it's a little weird that we have like these nested selectors inside immediate queries, but that is how we are

going to be working. So I'm going to do a dot columns to select my columns and then make a opening closing curly braces and now I can come in here and do a flex direction.

## 18. Making out layout responsive with flex-direction

And my flex direction will be calling. So let's go and see if this works. So everything

is looking great on the big sizes. And then when we shrink down and we hit 600 pics Boom, look at that everything is stacking on top of each other. Now, it's not perfect, right?

It's not great, but everything is stacking, which is what we want it to happen to begin with. So that is really, really good. But what's the problem? Why are these so not taking

up all the space like we want them to? Well, if you remember, we set widths on all of these.

So that is where the problem is coming in right now. So what we can do is create a new media query that is going to change all of those to give them all a width of 100%. So

what I'd encourage you to do now is actually to go ahead and try doing that, and seeing if you can get it to work. So we can do it all with one selector. But I want you to go

ahead and try doing that. Now that you've seen how I've done it, we want to target the exact same screen size. And once you've done it, I do want to look at a bit of a best practice

of using media queries as well. Right, so what we can do is I'm going to copy this media

query that we already have right here, and I'm going to paste it down at the bottom. And we can delete this here. If you tried redefining these columns inside this media

query, it wouldn't have worked. The reason it wouldn't have worked is because first we'd be doing the media query, and then we'd be following it up with these. And these would

overwrite the media query, because this would only be looking at a max width. But this is looking at all situations. So just like we saw in the last video, the order of things

really matter. Media Queries should always come after the default selector that you already have. So we can come into here. And you could either have done just call like that, because

all of my columns have the also have the call selector on them. So I can say call width

100%. And it will work. Or if you went and you did call one call to and call three, and

gave them the width of 100%. That would also work. I actually prefer this even though it's a little bit longer, just because we've set the sizes here. And then we're explicitly

setting the sizes here. So it makes it really obvious that we're resetting these specifically. Whereas if you went with that call selector just to see Oh, well, it might not be as obvious

that you're targeting these ones specifically. But either way, I think it is perfectly fine. So let's go and take a look and see if that worked. And look at that it is working. Oh

my goodness, isn't that cool? Isn't that really fun, that everything is pretty much falling into place. And our whole layout is changing, like boom, just like that, the whole thing

comes and just it works, which is really, really handy. And really, really fun and cool. Now, one thing, before we leave this video, we do have that padding issue, I'm not going

to worry about that one, right now we're going to worry about that in a future video. The one thing that you are noticing now is actually if two media queries, and you could simplify

things by having just one media query that has everything in it. And a lot of people

will do this, where they're gonna have all their styles. And then at the very, very, very bottom of their style sheet, they're gonna put all their media queries so that

way, they don't have to put like 10 media queries that are all targeting the same size, they can just have one media query. So even this called eg I could bring up to here, just

to clean it up. And then I have my media query at the bottom that's making the changes that I need to, it does make for you writing less CSS, but sometimes when you have a really

complicated stylesheet, personally, I find it complicates matters a little bit. And while it will make your CSS file a little bit heavier and a little bit bigger, I've seen some there

have been some tests done on when you're repeating yourself a lot with media queries, how big

of an impact it has, and it's really not that bad. So if you prefer keeping your media like

your columns, and then your media query for your column here, and then your these and then the media query for those there, there's no problem with it. That's actually the way

I prefer to work, I find it much easier when I want to make changes, all my columns, styles are grouped together. And then all of these are grouped together. But if you want to work

one way or another way, that is completely fine. So let's take a look at how flex directions

actually working. And what's really going on with this. So when we have a display flex,

we've seen the decree its columns, the reason it's creating columns is because the flex direction on it is row. And that's the default behavior. So if you don't declare wrote this,

it will be row and there's nothing you really have to do there. So that's where it makes columns. And it's doing that because it's treating the parent or the flex container,

like a row of content. So that means inside of it, all the flex items, or all the direct children are going to turn into individual columns. Now what we can do is we can change

our flex directions, we can say flex direction is column. And what that's doing, is it switching the main axis. What do I mean by that? If we have a something that has a display flex

on it, it looks something like this, we have our flex container, it's a row. So we have our columns of children inside of it. But if I change my flex direction to column, now

the parent is becoming the column. And these are becoming rows inside of it. And this sort of makes it look like the original default behavior. So you might be going well what's

the real point of this? It's really important to understand you're switching the main axis.

So you're keeping all of the Flexbox properties, all the amazing things that you can do. Now

we've rotated our main axis from being horizontal to being vertical. Why does that matter? Well,

let's say we have something that looks like this. We've done our display flex on it's we've made our columns, we know we can use justify content space between and that's going

to space all of them out. It takes all the empty space that we had that was left over here, and it redistributes it in between each one of my columns. And that's just super handy.

Well, if I take my display flex, but I go in and add a flex direction of column on it

with my justify content space between now it's going to redistribute the space that

was down here between each one of them, because justify content is always working on the main

axis. And we've switched the main axis it was going left to right, but now it's working vertically, it's working up and down. So if I wanted redistribute the space, I can use

my justify content vertically now. So that's really cool. So always remember, when I change my flex direction, justify content will now work vertically, and align items will now

work horizontally, because we switch them in axis. justify content always works on the

main axis align items always works on the cross axis. So a few reasons you might want to do something like that are examples of how things change. So we have my flex container

over here, and the line items flex start, they'll usually line up at the top like that, instead of stretching to their whole height. If I switch the flex direction, they're going

to do something like this, where they're all getting sucked to the flex start, which is on the left side. Now, my flex end would be on the opposite side over there. We know that

we can center items vertically, which is super amazing with Flexbox. Well, now we can center them like this. And of course, you can combine that with your justify content. So it's not

the line items is centering it horizontally, and then by justified content space between is spacing them out vertically, as long as the container is big enough to have empty

space on the vertical axis, they're always remember, when you use flex direction, you're switching the main axis align items now works horizontally and justified content now works

## 19. flex-direction explained

vertically. Element because the now it's time to look at how we can create a navigation

for a website. And all this flex stuff really comes into play when we're styling a responsive navigation for a site. Before we get into the CSS, we need to know the markup behind

them. Traditionally speaking, we use an unordered list and the unordered list will be placed in another element because the nav will be telling the browser that this is the main

navigation for our site. So it will look something like this where we have a nav inside the nav, we have the UL which is our unordered list. And then we have our list items which are

links, there is some debate these days on whether or not we really need it to be in the list. The reason that we have traditionally done it this way is for accessibility reasons.

And because it is a list of the different pages that are on our site. These days, there is some discussion about whether we need to or if we could just have a nav with a bunch

of links in it. Some people do this, but most sites you ever work on will be set up this

way. And this is still by far the most popular and common way that you will see navigations created. So it's really important that we understand how to style a navigation when

it's set up like this, if you can do it this way, it's much simpler if you didn't have the list items. So I do want to show you the more complicated way because again, it is

the one that you will run into in the real world. So I want you able to do that. Once you're done in this course through the next bunch of videos, this is what we're going

to be working on which will be the navigation for the bigger site that we will be working on. And we're going to start with this one here, then we're going to move into making

it look like this instead. And then we're going to make this the responsive version of this one right here. I'm going to go back to this slide and leave this one here for

now and encourage you just to try and write it all up and set it all up. So here are all the colors and all the different things that I will be using. And just to give you a little

bit of just see how I visualize this, when I'm looking at it, I put these pink boxes

in to say I have my big, I have this big thing that's going to be the whole size here, then

I'm going to have an element here. And that's going to have my h1 and a paragraph. And then

I'm gonna have another one here, which is the nav with the honor list. And the reason that I'm setting it up like this is this is going to be my header. And I'm, I need to

keep these two things grouped together, this is just going to be a div called site title or whatever you want to call it, it could be called logo, it could be site title, or

just title, which is the main sort of title of my actual website. It's almost like a logo, but we're just using some text for it. And then as I said, we need to have our NAB. So

that's what this is going to be for. If you want to start off and do it on your own before you watch me do it. In this video, I'm only going to be doing the markup you can see I

have given you everything you need here to do it. And I've also already linked to the Google Font and given you the two font families listed right here. So if you want to go ahead

and do more, by all means, see if you can get the whole thing working. In this video though, I'm only doing the markup. And then the next one, I'm going to start doing the

CSS for it. Alright, so let's jump right into it. So I'm gonna go in my body. As I mentioned,

I'm gonna have my header and close header. And let's get inside of there. And we can

start putting some stuff. So I had that first part that had my title in it. And I said, I'm going to group that in a div. So let's give that a div class. And I'm going to close

that div. And then we can give our class a name. So let's just call this site title. And in there, I'm going to have an h1 because it's really the title of my entire page or

## 20. Creating a navigation

my entire site, and I'll come down and also put a paragraph now my paragraph I will give a class to have some title, and close paragraph, I'm not using an h2 because h2 is really aren't

meant to be subtitles, they're meant to be titles for different sections of the website, we build out the full site, hope we're gonna see them, we're gonna have a few h2 and threes

in the larger scale site once we get to that. So for now, I'm just doing a p class of subtitles.

So I can style that different from my other paragraphs that I will have within this site, once we get to a bigger one. After that, I'm going to come and give myself my nav, and

close nav. And we can come into here. And I'll have my ul, and close ul. So that's my

unordered list. And then I need my list items. So Li, and inside my Li, I do need a link.

So a href is equal to and close a, and let's just take this whole thing and copy it, paste

them there. And now I can come through and fill this out. I'm just gonna put hashtags for now because we don't have anywhere to go. Once we build up the whole site for this

rectory I have a three page sites, we're gonna have to put in real links in here eventually. But for now, we'll put the placeholder, this is my home ism in life. There we go, there

is all my content. And you can see it doesn't look at all like what we want it to. So in the next video, we are going to start styling this up and making it look better, we have

to deal with some extra spacing, we have these bullet points we need to disappear, there is a whole bunch of stuff that we need to do here. So we're gonna see how we get rid

of those bullet points, and then how we can bring in Flexbox, and how it's going to work for all of this and how it makes our life much easier in setting up everything, how

many bullet points, we've got everything set up, but it doesn't look so nice right now.

So let's go and take a look at how we can do it, there is something new that we need to look at is how do we even get rid of those bullet points, which is kind of interesting.

So for this one, I'm going to leave this on the side here. So we can at least see the changes that I'm making happen live, we're going to jump over to my styles. Now for now

we have all these fancy designs and the colors and everything, I'm not going to worry about those for now. Because I really want you to be focused just on how Flexbox works. And

doing the overall styling and setting this up. I don't want to distract with other CSS. But we will be building out this whole thing for an actual website. So we will be getting

into all of the different styles that we need to do for this eventually, because I'm not worried about it, I'm not going to be terribly concerned with him organizing my CSS. So let's

start with my nav. And I don't have a lot to do on that. Now you'll notice here that I have my nav and a ul Li and I have my a and we need to style all of these, there's

stuff that we need to do on all of them. But what I what you will see some people do is you will see people give classes to this. And I actually do this on a regular basis

where you'll have a nav list. And then this would be a all three of these would be a class of nav list item, or nav Li and then my links themselves could have a class on them of nav

link or something like that. And I have done many projects like this, I think it makes a lot of sense, it does make your CSS a little bit more organized, or maybe not more organized,

but a little bit easier to read. So if you'd prefer to take that approach, by all means go for it. For now, though, I do want to show that it is not the end of the world use compound

selectors. And this is a really common time to use it, where I'll take my nav ul, we're gonna have some styles on that, I'm gonna have my nav Li, I'm gonna have some styles

on that. I'm gonna have my nav a, and have some styles on that. And of course, my nav

a hover and have some stuff on there as well. Now, what I will not do is have my nav, ul

Li and you will see tutorials online that do this. And then here it would be my nav, ul Li a. In the old days, it was a performance issue, the more complicated your selectors,

the browser would take longer to read that selector, because it would have to read Oh, I got all my links. Oh, but now I have to find the ones only in list items that are

only in my unordered lists that are only in my nav and it would actually slow the rendering

of your CSS down a little bit. But browsers have gotten really, really good and really, really fast at reading CSS. So it's less of a performance issue. It does hit on specificity

a little bit in that this is becoming more specific for nothing. And it's just makes

it complicated to read and takes a bit more of a cognitive load. When you're looking at it, you got to think for half a second when you don't really want to be thinking whereas

if I just leave it like this, it keeps it nice and simple. So personally, I like to keep it like that. And most people were advocate if you are going to have compound selectors,

which are already sort of frowned upon, keep them really, really simple when you do use them. Before I jump in, I'm going to fast forward I'm actually going to put a border

on all of these just so we can really easily visualize what's actually happening. Alright, so there we go. We can see all of my elements. I didn't put one on the hover because I don't

think we need it. But we can see my ul my Li and my a so the big pink box is the UL

## 21. Using flexbox to start styling our navigation

the dotted orange box is my Li and then these green boxes there are the links themselves

which are the AE and we can see that the UL and the Li are block level elements because even though the text is really short, those are stretching the full size of what's happening.

But you will notice that my L eyes are starting here. There's this big empty space next to

it, that's because of the way the URL is set up to give us room for those dots. So the first thing I'm going to do is turn off those dots. And there is a list style that we haven't

really looked at, I'm just gonna put list style to none, which will make those dots disappear, I just can't forget my semi colon, there we go. And now my list style of none,

it turns off the dots, there are other list styles that you can go through. But for now, we're just going to switch them off because we don't need them. Now, if we think about

this, I have this ul, which is the big pink box. And then inside of there, I have these three orange dotted boxes. And those are my allies. These are block level elements, how

can we make block level elements that are one on top of each other switch and be one next to each other instead, hope you said display flex. So I can come on this ul and

I can use the display flex, and all of a sudden, they shrink down to the content that's inside of them. So these dotted orange boxes are now the same size as my green one, we can

barely see those orange dots anymore, because there's a little bit of overlap going on, and they've gone next to one another. So that's fantastic and really good. Now with display

flex, we've seen that we can do a justify content and do a space either around or between

or evenly. So I'm going to do that really fast. And you can see it space things out. But we have this weird empty space left here on the side, still, all of our paragraphs

and our headings have a default margin, top and bottom. Because that we know erage because

the list normally has bullets or numbers on the side, they have that default margin, top and bottom. But they also have a default padding. So it's really common to say padding zero,

that's going to get rid of that extra space on the left sides, you can control things properly. Now the problem with using space between like this is it spreads things out

as far as we can get them to go. So depending on the size of the window, this could actually cause some problems. Or you can come in and give this a width of 350 pixels. And now I'm

starting to get the spacing that I want. And it looks nice. But now what happens if I add a fourth item or a fifth item, and well, your page when you first are creating it might

only have three pages later on, you might create more whether it's a personal site for you or a client site, then this number has to be changed, or you have to change a media

query or it's a nightmare. So instead of doing that, I'm going to remove the width from here, and I'm going to remove my justify content. And what I'm going to do is go into the list

items themselves. And I'm going to add a little bit of margin to push them away from each other. So in here, I'm going to do a margin of zero for the top and the bottom. And when

I'm for the left and the right, the reason I'm doing left and the right is because eventually this will be a centered layout. And I want the equal space on the left and the right.

If it was a left aligned or right aligned text, we'd have to look at it differently. And we're going to see that in a little bit. But for a center to align one, I want that

margin on the left, and on the right now I do have the problem that they're not centered. And one thing people often will try is to do a text align center. And it doesn't work.

The reason this doesn't work is because text align center will send your text within its

parent. So if we look at this living social life that's in my header, so if I come to my header, and I do a text align, center, it's centering Alright, actually, it's inside

the h1. So it's centering in that h1, which is full width. And this paragraph, it's this

text here is centering within that paragraph, which is full width. The problem here is home is centered, but it's centered within its Li and the lie is super small. So it's it's

super small because of our display flex here. So I can't use a text align center on these

to actually move them to the middle. And I won't lie I have been holding out on you. We can use justify content for this. So the same way we saw with a line items, we have

a flex and flex start and the center, we actually have the same thing here, I can do flex end

and push things all the way to the end, or flex start and keep things all the way, this is the default. And it keeps things just starting from the left side. Or I can do a center,

and it will center it exactly on the page. So with that done, if I come through, and I remove all these annoying borders, because we don't really need them right now. So I'm

gonna leave this one here. If you want to try and put additional styles or finish styling it go for that. Or if you just want to play around with the positioning and play around

with flex and the different things you can do with it, by all means jump ahead and have some fun with it. And then in the next videos, after that, we're going to look at how we

can modify it a little bit and then start making feet through taking off where we left

off. And we're gonna make this look a little bit more like the actual design that we want to have. So let's start from the top and sort of work our way down. So I already have my

header there, but this doesn't need a background on it. So let's add the background. And the background for that one is the f8 f8 f8, which is a really light color and you can't really

see it too well. So I'm going to make it a bit darker for now. Just make it pink so we can see it and we do see we have a bit of a problem with the it's not touching the sides

of our page. And while we're here let's just give this a padding on the top and bottom. I'm going to put two m n zero just as a rough estimate and we can always modify that but

I think it will work nicely. The reason I'm doing two M is just through experience. I know that 16 one M is 16 bits To me is going to be 32 pixels. That sounds about right to

me. It's one of those things that I practice with, I try and just use round numbers on my M's. One thing and one piece of advice I will give you in giving padding to things

is try not to go too small, but we're based on a design and I think two M is going to match my design more or less, I'm not aiming to make it pixel perfect. I'm just making

me to make it look as close as I can. But Sorry, I got sidetracked, we have this, this

problem of the space around. So how can I fix that I want you to go and do it before I do. So hopefully you came up under here and you did, you're not making your body and

put in a margin of zero on there. At the same time while I'm here. If I look at this, and

I also know the whole design, but I have this subtitle here. And this is using Ubuntu. and only the title is using Laura. So what I'm actually going to do, and I also I you haven't

seen it, but I've seen the whole design for this, I'm also going to take the Ubuntu font family and put it on the entire thing. So I don't have to read Eclair it throughout

any of these other things, the font size, I'm going to leave alone the color for now I'm going to leave alone as well. But maybe when we get to the full site and we see the

whole layout, we can worry about adding a color on the body for the header itself. I think that is good, except we want to switch this now to our f8 f8 f8 f8. Now we want to

start doing some type. As I said, I usually do my type after my body. So I'm going to come here, I'm going to make this a pretty big section. So when I say big section, I

just put this big thing at the top and I'm right type type for a word, I say all the time, I have a lot of trouble saying it. There we go, okay biography. And I'm going to come

down and we can start styling that up. So I have my h1. And for some reason, everything here is indented. So let's fix that. We go, and it's everything in nothing. So my h1 and

let's give this the font family of Laura, actually, we should just copy and paste to be a little bit faster. Now, one thing you might notice, or you might not have noticed,

but it is it might be showing as bold for you. It's definitely I see this is bold. And the reason it's showing as bold is I installed Laura on my computer. And as we know, all

the headings, by default are bold, but I don't want it to be bold, and I didn't even install

the bold version, I only have a Laura, I only took the link to Google fonts for the regular.

But because I have the bold on my computer, I see it, but other people visiting this site might not. So I am going to explicitly say I want the font weight to be 400. In this

## 22. Making out navigation look good

case, I did not put that here, but I didn't list bold, I probably should have written lower regular or lower 400. And I didn't so but I'm gonna change my font weight there.

And while we're here, we can set the color. So the color for this one is the dark blue, which is 143 774. And of course my font size, which was which is to ram, I'm going to come

down now to my subtitle. So my subtitle is a the font family is already Okay, the font

weight in this case, if you are bold, because I said to be bold in my instructions here, awesome. It works. If you noticed that in my link, it was 700 all the better. But either

one is perfectly fine. I'm going to put 700 because that is what we want to get used to

writing, designers will often say bold or black or extra bold because in the design software, it doesn't use the numbers it gives the weight, it just gives like a light or

a bold or things like that developers who work a lot with coders might give you the number, but there are some that won't. So just something to keep in mind along the way.

If you wrote bold here, though, it's going to give you the exact same result. So don't worry too much about it. We'll also do the color, which is 1792 D two, don't forget the

hashtag at the beginning or it won't work. And last but not least my font size because it needs to be pretty small, which is point seven, five. And it's tiny, tiny. Now, the

one thing I want to do is I want to pull it up, we have way too much space between these. So can you fix that, see if you can fix it and get rid of the space that's on there.

So I'm guessing the very first thing you did was put a margin of zero on here and nothing changes. So why did nothing change, I'll give you a second to think about it. Now maybe

you already figured it out. And you did it when I first asked you to. And if you did amazing, that's awesome. The reason nothing changed though is while this had a margin,

top and bottom that we just took off. This also has a margin top and bottom, which is still there. And this also has a margin top and bottom on it, which is still there. So

this big space underneath is coming from my h1. And this space underneath is coming from my ul here. So on my subtitle, my margin zero is helping but I'm an h1, I'm also going to

put a margin of zero on that to suck everything up. It's also going to balance out that top part I had a little bit better. Now there are a couple of things, the text in this one

is all caps. And it is spread out a little bit. I'm not going to worry about that quite yet. We're going to get into that eventually. The same with these with their all cap and

they're spread out I'm not going to stress about that. I just want to set up the basics of this for now we're going to explore typography a whole lot more in the next module, which

is stepping up our style. But for now we're just worried about responsiveness. I don't want to overwhelm with too many new properties. So let's get into this navigation because

we definitely need that To look a little bit better, I tend because I need to change the color of my nav a, and I need to remove the text decoration of my nav a, I tend to put

all of my typography styles for my navigation in the A here. So let's go and do that. I'm

going to say that their text decoration, since we just mentioned that decoration is none, we can set the color of it, which is the 70 7070, I can set my font weight to 700. And

of course, we want our nav a hover here. So this is going to have a color on it. And I'm just gonna use the bright blue. So it's really obvious when we're on top of something, and

it really changes but I made a mistake. Oh, my goodness, what's the mistake that I made, I forgot something. Whenever we have a hover, we should also have a focus state. So I'm

going to add that on there as well. Now, putting the focus on here is not 100% necessary, because

when we focus, we're going to get the browser's default to having a focus state already. So

you, I get like this glowing box that shows up on something as well as my hover color right now. But I prefer always making sure that I have a focus state styled, that adds

more visual impact than what just the default is. It's not 100% necessary, but don't turn

off that glove box, I'm not even going to tell you how to do it right now. And there's one last trick and that last trick is the underlying that's on here, can you think of

a way to do it, it does involve adding another class and everything else i think you know how to do. So I'm going to let you see if you can figure out how to do that. And in

the next video will give you the solution. I'm going to come in. So not all websites

will include something like this. But it is quite common to have something that shows you what page you're actively on, am I on the homepage, the about page, the Recent Posts

Page, or something like that. And as I mentioned, it does involve an extra class. So I'm going to come over here into my index. And on my home, I'm going to come in and I'm gonna add

a class and we'll call it current page, you will often see people call this active. So

this is the active page. The problem I have with that is the active we have it's on a

link. And we active is a pseudo class for our link for when we're clicking on it. So people get really mixed up or they think they can style the active state of a page by using

that pseudo class. And it's not at all how it works the active if you remember, it's when you're clicking on it. So I like either using current or current page, I'm gonna use

current page because I think it's super obvious what it is. And I'm going to keep that in my navigation stuff here. And actually, I put a big typography comment there. And I

never kept those going. So while we're here, let's just really quickly do that. This would be layout. And my nav is sort of a sub component of the layout. So for that, I'm just going

to do navigation like this, instead of having it as a big section. So we're gonna have our

navigation. And inside of my navigation styles, this is part of it, we have my current page, sometimes you'll see the current page is styled as a different color. So you can literally

at this point come in and say color, and change the color of it. And you can see that that is now red, but it's going to keep that hover color, but we have my color. And we can change

that. In this case, we don't want to change the color, what I want to do is I want to add an underline underneath, there's two different ways of doing it. One of them is with something

called a pseudo element. And that just complicates matters a little bit. And we're going to get into them much later on, because they're super useful for decoration. But we can get away

with just using a border to border I can say it's a one pixel solid, and I can set the

current color that we're using to make it the same as my text. Whoops, I made a mistake, though, it's on all the sides. And we haven't done a lot with borders. So I'm going to show

this one to you is because you might have forgotten that we can just do a border bottom. And now it's only going to put that underline on the bottom. Now if you find it's a little

bit too close to the text itself, you can also come through and add a little bit of padding, the one thing I wouldn't do is only add the padding to the current page, I would

add it to all the pages. And this is in general a good idea for your navigation anyway. Because if somebody is on a mobile version of your site, they have to click with their fingers.

padding makes your button a little makes that nav link a little bit bigger and it makes it easier to click on. So if I come on here, I might do padding, I went to five M and zero

on the left and the right. You don't have to do this here on the left and the right. The only problem is this border grows to match that padding on the left and the right side.

For the mobile reason that I just explained, it's probably better. But for now, I'm just going to leave that as a zero. So the underlying matches, and it does still give us a little

bit of extra room for what we can click on. And the last thing that I'm going to look at right now, though, is we want something like this to show that it's an interactive

element. This is technically an interactive element because it's a link to my homepage still, but I'm already on this page. So why would I click on it again. So even though

## 23. Adding the underline

it's still gonna stay as a link, I don't really want it to look like it's a link like I want these two to look. So what I can do is actually say that my current page, hover has a color

that is back to my default color because I don't want it to change And if I do that,

now, when I'm hovering on top, it's not changing. I know in the scrim of video, you just see it as like a blue point your arrow. But if you're looking at it in your own, it would

be a hand cursor. So it is people will still know it's interactive, but it's sort of double hinting at them that we probably don't want to click on that, because we're already there.

So it's like an extra hint along the way. Whereas these ones, it's really obvious that those are things that we might want to click on and visit. So with that done, I think it's

looking pretty good. In the next video, let's look at how we can actually make this on the left side and throw this over onto the right side. And then after that, we can actually

make the whole thing off. So let's go into we want to take this nice navigation we did

and I think it already looks pretty good. But we want to up the game of it a little bit and make it look like this, which is quite common on sites that you see these days where

we have the navigation on the right and probably a logo or something else on the left side. And we can just keep going using what we know about Flexbox to pull this off. So let's go

and take a look, what I'm gonna do is it will be a bit wider. So I'm going to leave this at a large size like this just for now. So it doesn't mean we have to shrink and open

it. But that's okay. So before we get into it, what I want to take a look at is how I'm going to structure it. So we have already my header which has that light gray background

on it and some padding on the top and the bottom. Then I have already this div for my

site title. And I have my nav over here that has my ul in it. But I need something that's

going to hold all of my content in the middle of the page just to stop it from stretching from one extreme end to the other. So for that, you can probably guess the name of it

already. I'm going to call that div class equals container. And I can come all the way

after my nav here, and I can close that container. And I'll put a comment here, close container

just so I can remember what that is, if ever I you know, I'm not sure why I have this random div floating around. And because all of this is nested inside of that container, I'm going

to select it all and push tab on my keyboard, and it will just push it all over. So I can see through the nesting and through my spacing here that I'm inside of my container. I'll

jump back over to my style sheet now. And we can start styling this. So I might say biography, I'm going to come here to layout. And the container is something very generic.

So here's my header, I'm going to put because it's more generic, I'm going to do this right at the top, I'm going to say container, let's give it a width of let's say 90%, because

that's sort of what we've been doing and a max width. And I'll stick with E, I'm going

to do 900 pixels. For now, we can always play with this a little bit later to make it bigger if we need to. And just so we can really see it for now I'm going to give it a border of

one pixel solid magenta, just like we've been doing up until now. So there it is, we just

need it to be centered on the page. Now, because we can see it is limiting to that max width. And it is shrinking when we're going to the smaller sizes. So I'm going to let you center

it on the screen before I go ahead and do it. I know I've had you do this one a lot.

But it is one of those things that people forget with the margin auto on the left and the right, especially early on when they're writing a lot of CSS, they use a text align

and it doesn't work. And eventually they remember and it sort of segments in so I'm trying to submit it in as fast as I can by getting you to do it as often as possible. So you can

see here now that it is working, and it's centering on the screen, and because of my max width, it does limit it to the total size there. But now we want to take this and put

it on that side and take this and put it on this side, we don't have the rest of our content. And normally I would write all of my HTML first. And then I would come through and write

my CSS. But we're not doing it this time, just because we're focusing on just the navigation here. So I wouldn't want to put a display flex on a container if that container is going

to be reused in many places. So like I've been talking about before, I like having modifier

classes, I'm going to call it container nav. And we can come back in. And now I can edit

## 24. A more complicated navigation

that. So I'm container, I would come down right below that and add a container nav and

give this a display of flex. If you did this ahead of time, and you were just sort of jumping ahead because you wanted to practice and you thought you know, and you give it a different

name, or you put it directly on the container, whatever that div that you gave in, that's completely fine. Or if you just had some sort of like nav box and you put all of these styles

on it, it's okay, I just know I'm going to be reusing the width, the max width and my margin zero auto a few times on this site. So I figure having a default container that

I can reuse multiple times makes a lot of sense. And then if I need to change it, I can so display a flex it makes our columns we come and take a look. There we go. We have

two columns. Now. Now there are a few things that we want to change. The easy way to fix is this is getting centered in here. And we don't want anything to be centered. That was

in my heading header header textilene Center, which we don't need to have anymore, so we

can remove that and at least it will fix that problem. And this part is perfectly fine.

But I want to move these all the way over to the right side. Can you think of a way we can do that that we've already looked at. So if you remember this pink box here is my

container and it's also my container. Our nav and on the container nav, I did display flex, and have all this empty space left over on this side. And how can redistribute empty

space to put it in between things we can use justify content space between, so justify

content, space between. And there's only two items in this case. So it just means all that

empty space gets put here in the middle. And one last thing, as you can see, there is a bit of a gap here. And you might be thinking that's not a big deal, and probably isn't

the end of the world. But because I know I'm going to be using a container and other content here. And I want things to line up really nicely. little differences like this make

the difference between someone who's a good coder or a good designer, and someone who's a little bit sloppy, and people who are hiring are paying attention to details. So you do

need to learn how to be detail oriented and look for these little things where things aren't lining up. It's not as obvious when you don't have a border on something. But

you'd still notice that it's not perfectly lined up. So if you remember what I did, when I created this, because it was center aligned, I said, we can do a margin on the left and

the right side and everything would be fine, because it's centered. And that means it's Eve, there'll be equal margins on the left and the right, which will keep it centered.

Now we have it on the right side. So that means I only want my margins on the left. This is one of the few times when I won't use the shorthand. And I'll just say margin

left. In this case, I'll do two n. And we can take a look. And now we've gotten rid

of that space. But I've kept everything here, I did two M instead of one because before

I had one m on home and one m on about me here. So one plus one, I was getting two on

the side. And I was getting two on this side here as well. Whereas now, I'm only putting march on the left, so I'm putting it as to M that does mean there's a two M margin on

this side here. But that's fine. That's not getting in the way of anything, it's just giving us a little bit of breathing room. And if anything, it's going to stop us when

we're shrinking from hitting that the the social life tech sits right there. So it's going to prevent us from smacking into that. And that gives us the room that we need. Now

it's starting to look pretty ugly when I do this, though. So the first thing I'm going to do right now is I'm going to turn off this border of magenta, because we don't want that

on there anymore, which makes it look a little bit better. And in the next video, we're going to add in a media query to prevent this weirdness from happening. And we're going to add it.

So this actually falls underneath. And we have a few different changes going on, we'll get some practice with media queries. We want to make our navigation responsive. And we

have to do that using a media query. So this is what we currently have. And we want to make it look like this when it reaches small screen sizes, I'm gonna give you a little

bit of a hint right now I'm going to use the breakpoint of 675 pixels when I do it. And

everything we need to be able to do this we have learned now media queries we haven't done a lot with. So if you're not super comfortable with them, that's cool. Watch me and see how

I do it. But I'm going to encourage you to see if you can figure it out. If you make a mistake, that's fine, I'd even say go and watch the video on media queries, and then

come back and do this because I want you to make mistakes, because making mistakes is how you're going to learn cuz you're gonna get stuck on something and you're gonna watch

me do it. And then it's going to be like, Oh, that's the stupid thing I was missing. And then you're going to be able to do it, you're going to do it, and it's going to reinforce

it. So try if you get stuck, go and watch the media query video, if you still can't do it, then you can watch me do it. Or once you've got it, you can see how I do it, just

to see if we're on the same page or not. And, of course, I will try and explain things in as much detail as possible on how I'm doing things. And again, the breakpoint I'm going

to be using is a max width of 675 pixels. Alright, so the very first thing I'm going

to do is I'm going to come to my container nav here, because this is where we have my display of flex. And we know that if I do a flex direction of column, it's going to

switch things. So now these are stacking one on top of each other. Whereas before, when we turn that off, they're going one next to each other like that. So this is what I want

to be able to toggle on and off. So I mentioned to max width, we're gonna stick with that for now. And I'm going to create my media query. So add media, I'm gonna put a space,

put my parentheses, put my my curly braces and put a empty space. And I can come in and

put in my max width of 675 pixels. Now I got that number it was through a little bit of

experimentation, and be talking about how you can figure out what numbers you want here. Not in this video, though. But when we start working on the bigger layout. So in here,

I want to select once again, my container now. And this is one of the mistakes I still

make, as I mentioned before, where I forget that I have to redo my selector because I'm looking here. So if that was the mistake you made, just always remember when your intermediate

query, you still need your selectors, just like as if you were outside of it. The other thing is always the media query after your selector and not before it or it will not

work. In this case, it might because we're not overriding something, but still always try and keep your media queries after your selector. So we can in here do our flex direction

of column. And so now if we come back and take a look at it, at large screens, it's

looking normal but when I hit that 675 Which is really close to when the text there, you can see it, the whole thing hops down. So this is a good start. But now I also want

to change the direction, I want to bring these to be stacking on one on top of each other. And I want everything to be centered. So let's see if we can do that. So here is where I

have my header. So we need another media query for it. If you're going well, I just made a media query, I didn't want to write the whole thing out. And I did say before, you

can always keep them grouped together to make it a little bit easier. You could also say I'm in my layout, I'm going to put all my layout ones together at the bottom. If you

## 25. Making the navigation responsive

didn't separate media query, that's completely fine. You're not bloating, your CSS file too much. Maybe it is, we talked about dry, and it's not super dry, always writing them over.

And they are kind of long to write. So there's nothing wrong with say grouping all my layout ones together here. So still inside the media query, and I'm paying attention like this

is the end of my media query. So I'm coming inside my media query, and I'll select my header. And on here, I'm going to do my text align center. So that takes care of this.

And it centers it because if you remember before, we've switched that we originally had it on there, and we turned it off. So now I'm adding that text align center back

in to center this and it goes back to almost that original layout that we had the first time now. But the cool thing is here, it's like that. And then as we shrink down, we

switch from left align to center, and they pop one on top of each other. So it's a really good start. The next thing that I want to do though, is I do want these to stack one

on top of each other as well, because we want it to work at really small screens, and we get to this tiny screen size, this looks pretty terrible. So for that I have all my navigation

stuff, let's come all the way to the bottom of all of that, I can put my media query, I'm just pasting it because I already had it copied. And I want to keep the same breakpoint,

you will have set you will have times we have multiple breakpoints, but try not to have them like 676. And then this other one 679. If you have something at 675, just keep them

all at 675. And just so you don't get a whole bunch of weird jumps going on, it makes your

life a lot easier as well to know when things will be changing. We already have a display

flex on my nav ul. So I see my nav ul display flex is making them go next to each other.

So that means the same way we did before I can select that nav ul. And I can put a flex

direction on there, of column. And now we can see that they're stacking one on top of

each other, but they're not centered. That's weird. Why aren't they centered. That's because

of this margin left that I put on my nav Li. So that means I have to come through, I have

to come down here and go to my nav, Li and I want to redefine my margin. Now in this

case, they're really stuck close to each other up and down. But the margin is breaking in the left and right. So on my margin shorthand here, I'm going to give it like a point 5am

on the top and the bottom, and on the left in the right, I'm going to set it to zero. But if we just did a point five on all sides, it would probably be fine. And we can come

take a look. And look at that it is looking awesome. And it's working at small screen sizes, and it's working at big screen sizes, and everything is great. And somewhere along

the way, I did lose my underlines if you want a bit of practice, you could go and add that back in yourself if you feel like it. But right now I'm super happy with that you've

made a ton of progress navigations for one of the simpler and smaller elements on a page, they're super complicated, there's so many different pieces to them through the header.

So we always have our header, we have something on the left, we have something on the right, then we have to get our navigational pieces to go on the left and the right of each other.

And oh, for something that's really small, and they they do take a lot of styling, though

you just did this, you just did one of the more complicated parts of most websites. So a bit of a pat on the back, probably a nice time to take a little bit of a break. If you

feel like you want to revisit media queries a little bit and go back through them do that now before we move on, because we will be using them a lot that we're going to start

building a whole layout. And I think it'd be a good time to take a break now just to help let everything absorb and sort of sink in a little bit, before we go into the full

big layout that we're going to be building, which is just going to be taking everything we've learned so far and giving us a lot of practice on all of that to help reinforce

things even more. But it will take us a while to get through that layout, because it is a pretty big thing that we will be building out with lots of different pieces. We're doing

the whole thing, we're doing some to build out the rest of the site. And we're doing the whole thing, we're doing three whole pages for this, it's really going to help reinforce

everything. And it's gonna also give you a bit of an idea of how it works when you're doing a whole site. I'm going to try and go through naming conventions a little bit, as

well as just seeing how there's a lot of similarity between the different pages. So once you've done the homepage, you're actually really close to being done, which is always really

nice. And we're also gonna be seeing how we can make it responsive. So we're going to be doing the small screens and the big screens, I have all three of them done for the big

screens only two of them for the small screens just because they're really similar to one another the homepage and this Recent Posts Page are almost identical so I didn't bother

doing a design for it. On the small screens for there. We are actually going to be building You know, from the small screens to the big screens, which is the opposite of what we've

done so far, but it is the more common way to work. It's called mobile first. And it's much more common to work that way, it's a little bit harder to share these in depth

designs through screen, but I'm gonna have screenshots of them, I'm gonna be highlighting the different things that I'm working on to try and explain things as clearly as possible.

So we have a visual to go along with what we're working on. But you can also come to this link. This is I designed all of this using Adobe XD, and this link will give you

all the design specs, and it's going to make it really easy for you to get the text and any information you need. So when you first click on that link, it will bring you to a

page that looks a little bit like this, it's a little bit bigger, and you can choose which one of the designs you want to start looking at. And you can click on it, and it will bring

you into that design. Once you're in the design, it will probably be on the comment section.

And what we want to do is we want to click where that big arrow is, which is to get into the assets and to start seeing all the different pieces that are in there. So once you're in,

you can get all the different colors I've used. So I am going to list out the colors, I'm gonna, I'm gonna have this as an available slide, but with the actual hex codes written

out. But if you're in here and you click on a color, it's automatically going to copy it into your clipboard. So you can just paste it right into the editor afterwards. And there

is all the font sizes and everything that I've used here and all the different fonts that I've used. So you can get all the different character styles, it gives you the font weight,

the font color, the font size, all of that stuff. The only problem is it also gives it all in pixels. If you want to work in pixels, because it's going to be easier, by all means

I'm going to do it with REM. But if you want to do it in pixels, there's nothing wrong with using pixels, like there used to be with font sizes. So if you want to go that route,

there's no issue with it, when you want to get the text, what you want to do is actually click on one of those boxes. So if I click on this, it's going to show me the size, don't

worry too much about the exact sizes of things, it's going to give you spacing, again, I didn't make this a pixel perfect design. So if the spacing in your final one isn't exactly the

same, that is completely fine. But when you click on a box, it's actually going to give you all the text that's in there. Now the only problem is it does not include line breaks.

So the continue reading and the title of it will sort of be all jammed up in one big paragraph. But it makes it a lot easier that you can sort of click through and get the different

pieces of text. That way, I'm also going to have a file in a scramble in the next video

where you can get the text from if you don't want to be using the Adobe XD thing here. If you're on a page and you want to get back out of there, you want to go see one of the

other pages, you can navigate through them, you can't see it at the bottom of the page, you can there's a little navigation to go through the different pages at the bottom.

## 26. Taking a look at the rest of the project

Or you can just click up here where it says the living living the simple life and it'll bring you back to that grid that you started on. So then you can choose which other page

you want to go and visit. So in a little more detail, here are the three pages. For desktops.

When I'm writing my markup, I'm always focused on looking at the most complicated part of the design when I do my CSS, um, as I mentioned, we're going to start mobile first. But when

it comes to the markup, we need to really see the structure of the website when all the different pieces are in the places that they're going to end up at. And it's always

a more complicated layout when we're on a desktop screen. So when I start writing HTML, I'm going to be looking at these designs. And then when I start my CSS, I might actually

be looking at the small screens, and then we're gonna move on up to these bigger screens. So in the next video, what I'm going to do is I'm gonna break down how I analyze the

homepage, and I'm gonna start reading the markup for that, and we're gonna make our way and write the markup for that, we're gonna get the whole thing down, we're gonna make

it responsive, it's gonna look great. And then we're gonna move on to those other two pages, because as I mentioned, they're going to go pretty quickly, once we've finished

everything else here, I'm really looking forward to this, I think it's gonna be so much fun. So looking forward to seeing in the next video where we can start calling. Guess on certain

things. Okay, so we're going to be focusing on starting the basic structure. For our homepage, I'm going to be doing the homepage, just the markup over a few videos just to break it

up a little bit, because it's going to make it a bit easier for you to find the parts that you want or to focus on certain things. So what I'm going to be doing in this one

is we're just doing the homepage, and I'm looking at the basic overall structure of my page. So we already have the navigation, I'm going to be nice to you, we're not going

to redo the whole thing, we're going to be continuing from where we left off with that navigation, we're going to be adding this whole section here as well as our footer down

at the bottom. And I can see that everything here is being held in place by something, it's not stretching out. So just like we used on our navigation, we're going to use the

container to hold everything in place there. Then, when I look at that, I see this is my

big piece that's holding everything. Okay, what's the next step? That's the big picture. But do I have something smaller than that? So yes, when I look inside of there, I see

that there's two columns of content, I see there's this main big column on this side. And then this column on the other side, there's no weird pieces sticking out, everything is

living inside of it very distinct column. So I know I need to set up sort of a main

structure area and a sidebar on the side. Then inside of those areas, I have these different

pieces. So to explain this color coding that I've done a little bit, if we look here, this is sort of this unique part of the website. This is the featured post, it's the one that

stands out the most The layout is unique. So that is its own component. It's going to have its own sort of markup the way the code or the HTML is written for that is going to

be a little bit unique. It's gonna have its own class name. Then I have these three that repeat themselves, they're all exactly the same thing, just the content in them is changing,

the layout is the same, the colors are the same, everything is identical, I just have to change the text and the image for each one. So each one of those will be exactly

the same structure, just with different texts going in. So they can all have the same classes on there, and they're going to work fine. And then over on this side, same thing, I

have these two components, these two little sidebar widget type things, they're looking exactly the same from one another, the content in them is different. But this mean styling,

the heading on them is identical. The big parts, the important parts of them are the same. So I'm going to give those the same class as well. In this video, I'm just focusing

on that I'm going to do the big picture, I'm going to set everything up. Then in the next one, I'm going to break each one of these up into its own video where we're going to

look at how to do the HTML just for this one, just for this one, and then a third video on the HTML for this one. And then we'll start going into the CSS and we'll go back through

and do the CSS for each one of those as well starting mobile first, and then graduating up to big screens. So if you'd like and you want to use that Adobe XD link, and you didn't

open it before, you can click here and open it up. Or if you prefer, I have put text files

for everything, including all the pages. So if you're being very ambitious, all the text is currently there. The one that isn't there is the recent posts, because the text is exactly

the same as the homepage, the layout will change a little bit. But all the text is the same. I didn't I put the sidebar and the footer is separate text files. So on the homepage,

it's just those articles. And for the sidebar, you can grab that from this file here, I have already included all the images, but you're gonna notice they're not the right size. And

## 27. Setting up the structure

I only have one version of all the images even though the images are changing throughout this. Now, you could have two different versions, you can have the thumbnail version and a big

version, you'll notice even the cropping is different than them, I'm going to show you a trick where you can use one image. So it just lessens the load a little bit, we don't

have to do two downloads, because we have the same image here. And here. For example, we can do all that with one image. Even though the cropping is different, it might not be

the most common practice to do it this way. But I want to show you a really neat trick on how we can re crop images in a way that is sort of new to CSS actually. And it's pretty

easy to do. So all the images are there, everything is ready for you to get started. And we just

have to come right here after our header and get to work. So I want you to do as much of this as possible on your own. And of course, then I'm going to go through and explain as

much as I can, as I'm going through it. And again, I'm really going to be breaking up my HTML a lot over different videos just to keep the video lens down. Because I don't

want to give you one really long video where you get if you have to come back to it later and you have trouble finding the spot you're looking for, I think it'd be a little bit

easier this way. So let's jump into it and start with this main structure here where

I just have this big main area. And then I have this footer down at the bottom. So what

I'm going to do, and I'm going to introduce a few new elements here. And if you did this on your own, I wouldn't have expected you to have these, we have our header at the top,

which often includes things like the logo and the navigation, that sort of what you usually find up in your header. And next up, if we go and look at our design, again, we

have this yellow box, right. But this yellow box is pretty much holding in the position

of what we have up here where we have our navigation and we use a container for that. So we might as well start with a container. So we can do our div class container, because

all of that does have to be held in the middle of our page. And below all the way to the

bottom, we can have a footer and close our footer, the footer, super simple. So we can

just throw in a couple of paragraphs right now, when we do the footer, I might come back and give these classes. But for the moment, I really want to focus on the structure of

what's inside my container. Here, we're going to look at two elements, which have a lot of semantic meaning and which are really important for the good structure of a site. So I'm going

to do a main and close main. And after here, I'm going to put an aside and close aside.

So the main is literally the main content of your website. The main, the main content

of the website with the focus of this page is so the focus of this page is all these

articles that I have on my homepage here, inside that we can have all of my articles aside is it's a bit of a weird one. It can fill in different roles depending in the context

that it's being used in, which can definitely be confusing, I've been confused by aside because there's lots of different information out there on it. But the main way you can

think about it, if it's not being used inside of a section of a main section of content.

So a div is not a main section of content. And the side is sort of a secondary information

for your whole page. So if we come and look at this, this is the main content. And then I have the secondary stuff, which is like these little widgets that are showing up on

the side here. So this is secondary. This is the main content that I want people to do. So the aside can be that as we go through and find other situations. Hopefully we can

understand a little bit more how things like a site and main do work. Main is the easy one though, because it is the main content and that also means you can only have one

main page mean is an important element to use on a page. And it can actually affect the accessibility experience of people that are visiting your website. Because they use

assistive technologies. And one of the things those can look for is your main, there are literally people who are blind that visit websites. So there are programs that will

read things out to them, this can be blind or just have very bad vision, and it will read the page out to them. Or it could be if you have an injury, and you're having trouble

using a mouse and using keyboard navigation, things like that. Having your header and having

a nav and having a main and an aside, this really can help the experience some people

have on your website. Now the main has very good browser support. But older browsers including

Internet Explorer 11, which people still use, don't understand it. So if you really want to be good, you can also add the rule of main on there as well. It's not required for newer

browsers. If you do want to support some older browsers such as Internet Explorer, then you want to have your roll of Maine on there as well. So I'm going to leave it now if this

is a little bit too much, or you're like this isn't what I'm here for right now. That's cool. I'm gonna put a link in the notes though, for people who are interested to the ally

project, which is a one one y, which is short for the accessibility project. It has lots

of information on accessibility, and best steps you can take in putting your website together, it is a fantastic resource to keep on learning and learn about a part of the

web. That's super important. But the not enough people focus on that with that little sidebar to the way on my side, I'm going to give this a class of sidebar. Now the reason I'm doing

that is not just keeping me aside it is possible for websites have multiple sides on like the main which can only have one of the side can be used for a few different things like I

mentioned. So because of that, just like a div, I'm always going to give it a class. And aside, I'm also always going to give a class that sort of works out for the main

content that we have here. Because I've put in these two columns now, and we have our footer, you can use, let's jump into our main and start adding a few things. And once again,

there is a new element that we're going to be exploring, which is the article and I think this one's pretty self explanatory. So I'm not going to do too much explanation on it.

But it is when you have an article on a website, you can use the article element. So it's saying this is an article. And an article sort of this self contained little elements were when

we're inside of an article, if I put a heading. So say I put an h2 in here, it's the headings for this article, they know that that heading is related to this article, which is really

cool and really useful. It's a little bit self contained compared to other things. Now if we come and look at this, this article is a little bit different from the other three

articles we have on the page. So what I'm going to do is I'm going to give this first article, a class of Article feature. And then on these ones here, I'm going to give them

a class of RT Cole recent, and I'll explain my naming in one second, let's just add this

on to those ones. So first, I'm starting with article and then putting featured just so

when I have in my CSS, I'm going to have my article featured and my article recent, they're not going to be separate from one another, like them keeping everything together, it

makes my CSS a little bit easier for me to read, if you prefer putting featured article in recent article, because that's how you'd actually say it. There's nothing wrong with

that at all. I prefer starting things with sort of the the name of what it is, it's almost like going into a modifier class of button. And then button accent. It's my article featured

and my article recent. So I know they're both articles, because that's the first thing I see. And then I can see, oh, it's different because this is my featured one or recent

one. In this article. If we take a look at it, I have my image first I have the date. And then I have a title, a paragraph. And the last but not least, I do have the continue

reading, I'm going to do something a little weird though. And then I'm not going to put them in this order. So we can explore something else with Flexbox when it comes time to style

it. And because we want to sort of follow best practice. So the very first thing I'm going to put in here is the h2. And I'm going to give this a class of article title and

close h2. After my article title. Now what I'm going to do is put my image, so we need

an SRC on that, which we'll get to after and then a lt, which we'll do after. And we're

also going to add a class on this of our default image, we can follow that up with a paragraph.

And actually, we need two different paragraphs because we have one for the date and one for the rest of the text. And then we need my link down at the bottom. On the first paragraph.

There are two different paragraphs, but we're using them for two different things. This one is the date and how many comments the post has this paragraph here is just sort

of like a regular paragraph. So on this first one here, I'm going to give it a class of

article, Article info. And the reason I'm doing info is because I'm going to put the date and the comments in there and not just have one or the other. So we can so if it

was just the data if you'd call it article date, or if it was just the comments, we could do article comments, but I'm just going to call it article info like that. Then we can

come down to this one. I'm going to Give it a class even though I don't know if it's really necessary, but I'm going to call it article body. And the reason body is body text, if

## 28. Featured article structure

you're coming, I come from a you come actually from a print background regular text you seen an article is the body text, it's also we have our you know, that's why we have the

body, it's the body of the page, this is the body text of the so it's like the main content,

don't know if I'm actually going to end up styling it, but you sort of get the idea that I'm using a class for everything here. Because when I get to my CSS, that way, I know, it's

super easy to select anything that I want to select. So then I also have this down here,

I'm going to give this a class of, I'm gonna call it read more instead of continue reading

just because continues really long. And I'll probably make a typo. And it still gets the same message across. So I think that works out fine. Now, again, the order of this is

not the same order as here. So if you did it in a different order, there's nothing really wrong with that. The reason I'm doing it in this order, is imagine if somebody were to

come on this page, and for some reason, the CSS doesn't load, if they get it in the actual order. And they see the content in the order that we see here with no CSS, the hierarchy

and the order things are in there not really the best, we have this useless sort of information

first, to be honest, followed by the title. Also, imagine if using a screen reader, and it's just going one piece to the next piece through this head, it reads this, and then

it reads the title, it's kind of weird, you read the title first. And you know if you're interested in that without knowing the date that it was published first. So I like putting

it this way. And we can actually use a cool Flexbox technique and feature to change the order of things here visually, without impacting the actual order that they are in the markup.

So this actually makes more sense when you're looking at it from a content perspective. But with the styles, we can change the order of it all, which is really, really cool. Now

these recent articles, they're a little bit different. So I'm actually gonna break those down in the next video, I was planning on doing it in this one, but I don't want to

overwhelm we already went through quite a bit there. So in the next video, we're going to look at this because if we go and look at it, oh my goodness, we actually have two

columns in these. So let's see how they've closed in our recent articles here. As I mentioned,

right at the end of the last video, we actually have two columns. Oh, my goodness, that complicates matters. No, doesn't it. So I'm going to come in here and create two divs, one for each

column, div, close div. But this first column here is going to have a class on it of Article

recent main. And I'm going to give this one a class is equal to Article recent secondary.

So it's like my main content and the secondary content of my article. Now if you don't really

like those names, I maybe could come up with a better name for them. A lot of people don't like the idea that I'm repeating article recent. I don't like the idea of just using main and

secondary because maybe I you have a main and I have a secondary for something else two that are just completely different. Could you instead of doing it this way, have an

article recent and then use a descendant selector, your compound selector to say like article recent main space main and article recent space secondary, yes, you're 100%. Good. But

again, I like trying to avoid those descendant or those compound selectors when I can. So I'm going to use my main and my secondary on this, it does create a long class name

to have to write when we get to CSS. But it does also make my CSS really easy to read and understand when I'm working in it. So I do like that, even if it does mean being

a little bit more verbose. The main content here, this is where I'm going to have my h2, which can have the same class as I have here, is sometimes a featured article will have

a different like font settings. But right now it's the same font size, the same font color, the same font, everything. If these two different articles had different font

sizes, I'd be looking at this in a completely different way, I'd be giving this one like featured article title and this one recent article title or something like that, it's

the same, they're both the exact same styles on them, I'm going to stick with the same class, because I think that makes a lot of sense. inside of there. I also have my body.

And I also have the continued readings, you know what, we can copy all this and paste

it in here. And exactly like before, these have the exact same styling on it as they do up here. So we can keep the same class names once you've styled at once we've styled

the whole thing, the main difference between our featured article and this one is just the structure of them is a little bit different. So we can have the two columns in one, whereas

the other one doesn't have the two columns in it, these two, I can grab, and I can throw in this secondary over here, because they really are secondary, secondary information.

And now that I have one of those built, well, I can just copy that whole thing and paste it right here. There's all the structure of that. In the next video, I'll take a look

at the site. Now, the sidebar is actually going to be pretty easy to set up pretty much

because we have just two different things to create. They're really, really simple. We only have one thing to actually create in all of this, which is I'm going to call

it div class equals sidebar widget. Now, if you are going Kevin just call it a widget.

Go for it. You can call it just a widget. I just want to be super explicit that this widget is meant to live in my sidebar, I don't think I'd have a widget that would live somewhere

else. But when I'm styling it up, and when I have it, I just want to be really clear what this widget is actually being used for. We have two widgets. So the widget, if we

## 29. The home page - HTML for the recent articles

come and look back at the design, the widget is the same with a big gray background on it. And then inside my two widgets, so it's one and two, there's a little bit of content,

but it's really not that much on two separate things a little bit. The first widget here is going to have a h two on it. And the class will be I'm just going to call it widget title.

Because I think if I put sidebar widget title, people would start getting mad at me, I think

widget title is fine. It's very straightforward what it's being used for. I do have an image in here. So we have our SRC and our al t that we need. And then in this case, I have a paragraph.

So you probably guessed it, I want a class on this to let's call it widget, body, close

paragraph. And then I have my other widget over here. Now this widget is a little bit different, I have the same styling on this as I do on this one. But the contents a little

bit different. I just have an image and a title. So what I'm gonna say is this widget sidebar here. So we're gonna start off with one thing, which will be an h2, just like

before to have the class of widget title, close h2. And then, instead of having an image

in a paragraph, we have a few like, they're sort of like articles, but they're not really articles. So I don't think they deserve to be called an article. So I'm just going to

use a div class. And it's going to be called a widget, recent post, close div. And we have

three recent posts, I'm just going to do one and we'll copy and paste it a few times. For my recent post my widget recent posts, the reason I'm doing a widget Recent Posts is

if I come and look at it, I sort of have an image, I have a title and then I've underlined image, title, underline image title. And that last one doesn't actually have an underline.

So we'll have to figure out how we can turn off an underline on the last one. It's not something hard to do. But it's something new and really cool that you can do with CSS.

I always get excited about the CSS parts, these markup parts aren't as exciting. But I digress. Let's go and look. So what I would do here is I'm going to put an h3 in this

case. And the reason I'm doing an h3 is because they are a title. It's a title for the post. But it's a sub section to this. So recent posts, this is like the category recent posts.

And then I have these three different recent posts that are in there. And once again, I'm putting my h3 before the image. And one part of it is because I really want to practice

something with Flexbox, where you can change the visual order of things. And I do think it does make more sense to have it structured this way with the title followed by an image.

## 30. Home Page - HTML for the aside

If you did it the other way around, though, it probably wouldn't be the end of the world. So this do we give it the really long title of widget recent post title, I'm going to

keep the widget recent post title if you wanted to shorten it up. If you're doing your own site, and you said widget post title, I think that'd be perfectly fine. I don't think there's

any issue with that. The reason I wouldn't want just post title or when Yeah, the reason I wouldn't want something that doesn't have the word widget in it is because I wouldn't

want people to get it confused with this article recent because we have article recent but then this article title here. So if we only put something in here that involve the word

recent, say recent post title, that can really get confused. And when you're looking at it, you'd think it actually belongs up with my recent articles. It's very verbose. But what's

wrong with being a little bit verbose, it just makes your code much easier in the long run, when to know exactly what it's being used for. So I don't mind being a little bit

verbose. If you have a shorter way of writing it, or you can come up with a better name, there's nothing wrong with that, by all means come up with shorter ways of writing it, I

might be a little if this was a personal site that I was doing, I might try and come up with things that are a little bit shorter, just because I don't like typing. But when

we're learning especially, I think it's super useful to being super clear what things are being used for. And it is a good habit to build up when you're coming up with your class

names. Anyway, it does make class naming a little bit harder sometimes, but it's usually worth the trouble. So my image has a source on it, it will have an Alp on it. And it will

have a class on it. And you know what, I forgot to put an image. I forgot to put a class on

this one. Now I think both images are gonna be treated the same way. So I'm going to call this one widget image. And with with his widget spelt with a D. Man, which is a weird word

a, I was thinking maybe it was only widget, but it's with the JIT. All of a sudden, after

looking at a whole bunch of times, I went oh my goodness, I just spelt it wrong throughout this entire thing. So yeah, I think we can take this class here and put the same one

on this because the two images will be behaving or the four images will behave behaving in

a super similar way. And once I mentioned we have our widget recent post once that is

finished, we can just copy that one and paste it a few times. And now we have everything we need for our homepage intact are ready to go I'm not going to make you go through

and put all the text into everything here. But if you want to, and you're adventurous, go through right now and put all the text in and just see what you can do with it. See

if you can start styling this up and organizing things, at least on a basic level. I know we haven't looked at how we can reorganize things visually, when the order in the markup

is wrong, we will be looking at in the videos, we're also going to see a little bit more on the images and how we can crop them properly. But you have most of the tools, you have to

be able to do the site. If you don't want to do the text, I'm going to do it all off camera, when you start the next video, all of the text will be in place and all the images

will be there. So if you just want to start with all of the content already in place, hit pause right at the start, and then you could try and style it up from there. I don't

blame you, I probably do. I do think it's going to we're going to be getting started

with the CSS, I'm so excited by this. I love working on big projects like this and sharing how I work on big projects like this. Well, I can't see the work you do. I'm excited for

you to challenge yourself. And you'll have some fun with this. So here is that Adobe XD link again, just in case you never opened it, I do think it's going to make your life

a lot easier. So I really encourage you to click through and use that to get the specs. But if you'd rather I'm going to keep this slide open. The nice thing with the Adobe

XD version of it is if you go on the homepage here, you can zoom in on different parts of it and look at them in a little bit more detail, which might make it a little bit easier to

work on. In this video, we're just focusing on the global styles sort of setting the stage for everything is going to take us a little while to get through all the CSS that we need

to do on this one. But let's jump into it and actually get started. Because it is the first time we're looking at a big project I am going to start out with you want to do

it on your own, by all means, do as much of this on your own as you can with what you have. And with everything. You know, as I've already mentioned, there are a few things

we have not covered yet that I'm going to use this project as an opportunity to teach you about. But constantly throughout all the lessons we're building out this page, I will

be stopping you and asking you to do things that you've already done. So it's not just going to be following along with me, I do want you to be pushing yourself and learning

while we're doing all of this as well. So if you didn't get that Adobe XD link already, it will also be in the notes of the lesson too, just in case you don't want to have to

try and worry about the slides or anything like that. So as promised, I put in all the

content here. So that will definitely make your life a little bit easier. And let's go over to the CSS, we already have some CSS in here. Because when we did this when we

did our navigation, right, so we're going to leave this all here, it does make our life a little bit easier to work with, because we've already set a few things up. Now, I'm

not going to be constantly coming and looking at the layout throughout this, but I'm going to have it open on my own screen on the side like I'd encourage you to do with the link

that I have shared you because it's a lot easier when you're looking at your design, I will be opening it occasionally. But I don't want to constantly be opening and closing

it just so it's I think that'd be annoying for you. But whenever I'm sort of looking at bigger subjects or when we have to come and examine something I want to explain myself,

I will open up these, I will open up the slide just to take a little bit of look at it. But I'm just grabbing font sizes or doing small things, I might just refer to as you can see

in the slide or as you can see in the design. So there's a lot of commonality that's going

on in this all of our titles are looking exactly the same. We have our continued reading, we just have the basic body. And the setup of that. As I said, I like starting with global

styles. I already mentioned this videos about the global setup. So that's what I'm going to be focusing on. And of course, we I've given you a whole bunch of the typography.

Now I didn't put the colors, but I showed you what it looked like. So I'm assuming you can figure out what color you need and stuff like that. So let's go and set all of that

up. So where do you have our h1 and our subtitle set up here. So we might as well keep going through this and set up the other parts of our typography. Because I do think that's

the best place to start whenever doing your CSS because it really sets the stage when you start doing the layout stuff. Sometimes if you don't have the type set up, things

don't look and you start getting frustrated and you're moving things around. And then you change a font size. And you have to undo other stuff you did. So always starting for

## 31. Starting the CSS for our page

me, it makes a lot of sense. Starting with the type. I think it prevents problems or other issues from cropping up along the way. So we had our subtitle, which I could put

here, just a comment to myself that this is like the logo subtitle because I didn't give

it the best name to be honest, because we also have our article title. And I don't want to confuse that my subtitle like this is the subtitle for my h1 or maybe I could even put

h1 subtitle. So I know that it's not dealing with my article title that I created. And

it's it's not a subtitle for my article. Now if I did have subtitles, which I don't hear, but if we had like a secondary title for my all my article titles, I would have had article

hyphen, subtitle, just to clarify my naming and all of that. So I'm going to go through

and set this one up. So the font family is Laura. And what what's the easiest thing to do to get that font family it's to come up to here and to copy it. But wait a second,

if I have to come somewhere and copy it, that already means that maybe I shouldn't be doing that maybe I should be combining two selectors together. Probably a good idea because I also

am going to want my font weight here to be shrunk down as well. So I think what I'm going to do is I'm going to take both of those off of my h1 and actually come all the way up

here and put an h1 h2 now I'm doing it on the h1 and h2, I could also be doing this

on my article title. If you think that makes more sense, by all means go for it. But I

am only going to have my h2 says article titles in this page, it's very possibly of designs

that are different, and that aren't like that. And you know what my h threes have that as well, because my h threes are going to actually open up this, I said I wouldn't do much. But

these are my h threes. And they're sharing a lot of commonality with everything, my h1 and h2. So even though they do have a class on them, the way I usually do it is I'll start

with a really generic h1, h2 h3 do as much like global stuff on that. And then when it makes sense to have my class, to put things on that, it really depends on the layout and

the design that you're creating as well. Sometimes you only need these, you never have a class, sometimes you have a lot more on your classes. I wish there was like this is the answer that

I could give you. One of the fun things I think with CSS is there is no this is the answer that I can give you, which can be very frustrating when you're learning but I do

want to show you that it's not all just like this cookie cutter cut and dry thing. And that if you didn't do this, and you just put everything on the class, that'd be perfectly

fine. There's not a problem with that. But if you find yourself repeating yourself, try and make combinations. And again, you're gonna see you could even have an h1, h2, h3 and

then maybe you know, have your article title in there as well as a selector, even though that might be a bit redundant, maybe you're gonna have an article title, that's a paragraph

for some odd reason, you never know. And you just want it for visual reasons. And not necessarily

for other reasons, you know, I can have my article title here, as well as like another selector that's in that. Now, I'm not doing that this time, because my h2 and my article

titles are the same thing. But I'm gonna leave it just like this for now. But if you're not too sure, either put everything on your class, and you're completely fine. There's nothing

wrong with working that way. It's how it worked for the longest time. But now whenever I find myself having to come and copy something, I'm just going to group it up to here. And

because this was just my regular h1 tag, I'm just sticking with h2, h3, but if I put my classes up there as well, that's completely fine. And completely normal. You know what

else is the same on all of them actually, is the color, they all use that same dark blue color. So we can also put this up here, and we've done a lot, the only thing I really

need my article title for is the font size. So my font size on this case, it's 24 pixels, which comes out to 1.5. REM. So if you were looking at the Adobe XD, you would have seen

24. It's one of the very common number for font sizes. And it works well. 1.5 is a nice,

easy one to do. Now I do have the body text font size, which I have is 1.125. REM. So

I have a choice now of you know, this is my body. And then I have my typography grouped here, there are some people who will come and put like this and say body and put a font

size down here as well. Again, I'm not doing it on my paragraph, because I don't want it to only affect my paragraphs, I want it to affect everything. And it's not so much going

to be affecting on this page. But we will have a project where you'll we will see a big difference on that. So I'm going to come up here onto my body. And I'm going to set

the font size here to 1.125 rim. And I think that is it. So that's good. And the next thing

we want to do on this one, so we have the links at the end, which always have that article read more that say continue reading in them. So we need to style that. So article read

more as the font size of 0.875. REM. And if you leave off the leading zero, there's no

problem, you can just do just do it like this, I tend to do it that way. Because it's one less keystroke, I know a lot of people prefer the zero, it's up to you which one you want

to have. And there's the dates that have the same font size on them. So why don't we do this. But we also have, I put dates in my Adobe XD file. But if you remember, we actually

called it article info. Because it's not just the date, there is some extra information in there as well. So the two of those have the same font size. Look, don't forget the

dot. But then there are some differences between them. So on my article read more, we also

want to be throwing in here a few other things. The color on that link is that bright blue

bars there links. So if we actually I'm going to take a look at this for the first time since we've set all of this up, we should see it's starting to come together. Because

we're setting things up really nicely. We forgot to change the font weight on the body.

And notice how it's a little bit bolder than in my design. So in my XD file I earn even

on the slide here, where I put the style, I definitely mentioned that it should be Ubuntu light and I didn't make that change. So here with my continue reading button, we can see

there's an underline on it. And we don't want that underline to be there. So we can turn that off text decoration of none, and the font weight of 700 because it is the regular

bold if you wrote the word bold that that would be perfectly fine to the one thing we won't be looking at, as I mentioned when we were looking at the logo is the spacing on

the letters. The spacing on the letters is a little bit spaced out. We're going to see in the next module, we're going to come back to this project and we're going to be stepping

up our style we're going to make things look a little bit fancy And nicer. So let's come up to my body on that one thing that I forgot, which is my font weight, font weight should

be 300. And actually, and let's go look, I don't think it's working. There's a reason

why it's because I didn't bring in the I didn't load it in with my Google fonts. But I want to take this opportunity to show you a nice little trick. Because sometimes you realize

there's a font weight along the way that you need that you didn't bring in or you didn't load. When you were at Google fonts. It's annoying. If you've closed your tab to find

the right font to open up the little black bar thing to go back to customize, get the link, copy it back. You know, it's not, it's not that many steps that I hold them. If you

can avoid it, it's always nice. So when we're bringing in our Google fonts, I've shown you that it adds the numbers here, it's just putting in the numbers and comma separating them.

So if you forgot one, and you want to add it in, after, you can literally just come here and add it in like that, it's really, really simple, really, really easy. And now

we have access to our 300 as well. So if we come in look, we should see ha, there we go.

The fonts are nice and light, and they look much, much nicer. My continue reading is also looking Okay. One thing I didn't do on that though, was I made a bit of a booboo. I did

not include a hover. So I'm going to come in and add a hover to it. And if we have a

hover, we need one other thing. What is it? Hope you said focus on these long ones I like to copy and paste to make life a little bit easier. I'm going to put take a minute, I'm

going to remove the font weight because we don't want to change that, I want to change my color to the darker color. Just so they're just we have a color. But I also want to add

the text decoration back in. Because I find the subtle, it's going to be a shifting color. But I think adding the underline will really just make it more pronounced. So the original

text decoration, we haven't looked much at this. But if you want to, it's just writing the word underline the default style that's already there. So we can add it back in by

doing a text decoration underline or you can put that on something that doesn't have an underline and you can create once you get a span where you want it an underline. This

is one way you can do it. So now if we go and take a look at what we have, we have the everything is looking pretty good. And you can see the changes color. And I'm getting

that underline that's coming on there as well. So we have two things left to do in the typography section before we finish off this video, but I'm going to do them as a challenge to you

before I do them. They're both things we've looked at before. And I won't lie to you, I forgot to do this at the beginning. So you might have noticed at the beginning of this

video, we didn't have the bold or the link here. I've just added them in now. Sorry about that. But now that they are here, I need you to style them. For me. The problem with the

bold one is it's not bold enough, it looks a little bit bold, but it's not really quite getting there. And I'll explain why in a second. And then the other problem is this, it just,

you know, some basic styling on our link, we want to make sure we have a hover, change the color and all of that on there. So I'll let you do those two, and then I'll take a

look and explain a little bit of what's going on with the strong. Right. So I'm assuming

you put them down here when you did it on line 60. And I left a space there to do it. So that's completely fine. Just to see if I was doing this myself, I'd actually leave

this space here because I'm going into a next section of content anyway. But I would actually bring those up here where I have my general, like my h1, h2 h3 selectors, because then

I'm getting into classes. And I just like organizing things. And this is just personal preference on my part, there's no right or wrong answer here. But I would have my a here

as well as my strong tag. If I had paragraph styles. If I had anything that's just a regular tag, before I get into the typography of my classes, I would include all of the general

ones first. So on my eight here, we can set the color of it, which is the light color. Now when that you might have also found a way to sort of combine it with other selectors.

And if you did that, that's really cool. I think that's the only change we needed there. But of course, we need our a hover, and a focus, we also want to give it I'm just going

to do the same color change that I did on those read more links. And as I said on, we need to style the strong tag. Because the font isn't bold enough, it's a little bit

bold, and it's not really getting very bold. And the reason for this is, depending on the

browser, using actually bold can have two different settings on it. One of them is bold,

which will just bring it to 700 automatically, but the specification actually outlines that it should be bolder, like there's a there's a literally a boulder keyword. So if I come

on my link here just so we can stay in one spot and not create a new class or anything.

If I set the font weight on this to Boulder, you'll see it looks a lot like my strong tie here, it's gained a little bit of weight. What boulder means I'm going to go back to

my index for a second is it means we're currently here, go one step bolder. So it's going to

go from 300 to the next available font in the scale whereas if you use the actual bold

keyword, so instead of bold it goes to a bolder it goes to bold. Now you can see it's gotten

super fat, like look at the difference there between this one and my link. So bold, really bold will always jump to 700 Where's bolder, we'll go to the next available. This depends

what browser In Google, Chrome will actually normally go to bold, whereas Firefox and some other browsers will go to Boulder. But boulder is the official specification, that's the

way it should be working. That's one of the reasons on a strong tag. Even if it looks good on your screen, different browsers might be doing different stuff. So always tell it

what font weight you want to have, in this case, it should be 700. And we can take this font weight of my aof. And if you put the word bold, that would be fine too bold and

700 should end up at the same place, I prefer the numbers a, they're faster, they're a bit more consistent. And when you get start getting into like the in betweens, which you're often

going to be using, it's just a lot easier to use a number. So there we go. Now I have the nice big bold font on there, and everything else is looking good. My link is clearly something

we can click on. And that's it, we don't have anything else to do the dates, we don't actually have to this article info, we don't actually have to do any other more styling than the

font size. I know in my outline, I put the distribute light, but because I put the light font weight on my body here, that's automatically going to get applied to everything. Now pretty

much unless I explicitly override it by putting a font weight on it. So everything is done,

we're good to go. In the next video, we can start with the layout of curfew first. In

this video, we're just looking at the big picture, I always start big picture. And now typography maybe isn't big picture. But as I mentioned, when you set up layout, and then

things aren't looking good, then you change the type things change again, then you go back to layout. So I always like doing typography first, then big layout, and then slowly working

my way to like more and more like all the different smaller pieces of my layout, which we often call components, we have a few different components. On this page, we have our widget

component or featured article component. And our recent article component is sort of how I'm going to break this page down in the same way when we were doing our markup how I looked

at it. So we need something, it's going to hold the page in the middle. And then we need that to have two different columns on it. And we need to set those two columns to have

the right sizes on them. A few decisions that have to be made now about a few different things. Now one thing that we've already done, if we scroll down, and we find where we did

our layout, we already created a container. So that's pretty cool, because that means our content here is already inside of a container, it's already being held within the same place.

Because when we created this, we gave it that same container. So if I go really, really big now, I can't go in my content, we'll stop growing at one point, which is awesome. So

we don't have to worry too much about the container itself. The first thing I'm gonna worry about is setting up those two columns, though. So in my layout, if we come back,

actually, we should look at our indexes, you're always gonna be jumping back and forth between the two of them unless you have a really good memory for your classes, we had my main and

my main is all my Main articles, and then we had an aside right there, and then all

my aside is that secondary widget on the side. And all of that is living inside of this container.

Now the container, I don't want to give it a display flex, because I might get other content inside of a container that I don't want to have a display flex on. So that's

the same reason I had my container and my container nav here. And when I did that, it's because I wanted to make this container have a display flex on it. Now what I'm realizing

is I'm coming in, I want to do the same thing here. So having container nav here wouldn't make a lot of sense. I could call it container main or main content, but why not come and

give it a class like container flex. And the idea here is I have a container, but I'm also

making this container a Flexbox. If I do container, flex there, and then I come up here and I

change this one. The reason I'm changing that one now is because I'm going oh, I want these two to have the exact same behavior, I might as well give them the exact same class, it

This might not work in every situation you run into. Because if you remember when we did this, we had display flex, but we also had this justify content space between. In

this case, that's a good thing. It's exactly what we want, because we want to push our two things as far apart as possible. So container nav display flex, that's fine. So we can just

change this from nav to container flex. And that means anytime we have a container, we

need to display flex on width is justify content space between we can use this class and we're modifying so we have a normal container if we need it. And we can take that a step further

when we want to with this modifier of that original container. Now technically, and a lot of places we'll just call this like flex, or though it'd be like a flex class, right?

## 32. Starting the layout - looking at the big picture

It'd be flex, and you might add something to the name to indicate that it's also doing this or if you really get into something called atomic CSS, it would have like a diff, you'd

have literally D flex, which would just be display flex, then you can have something that else is just justify content that just applies justify content. I'm personally not

a fan of doing that. But I think you could just call it flex if you think you'd use this

for a whole bunch of reasons. I tend to do more, this type of thing personally. But one

thing we CSS and CSS naming conventions, there's about a million different approaches to it. So I'm throwing a few ideas out there. I'm gonna stick with this one for now, if we've

worked through our project, sometimes you end up running into a situation where maybe you end up changing the name again. Now part of CSS is planning things ahead. But it's

really hard to do when you're a complete beginner. So I want to show you the type of things that you might run into. If you've planned things out really well from the beginning, because

you've made 100 sites, and you can analyze the design really well right from the get go, you might not need to be making these things. Or you might realize I always need

to have something like this. And right from the beginning, you're always giving yourself some sort of flex class that just has like these basic settings on it that you can use

whenever you need it, simplify your life as much as possible. Let me go take a look at our layout. Now. We should have two columns. Now everything's a little bit broken, because

my images are huge. So we can see though we have two columns, that's amazing. We're almost finished honestly, with the layout. And the two big things that we need to go through

and fix right now are the image sizes, and it's really going to help and we can set our column sizes as well, I'm going to go all the way up to the top where my body is. And

right after that, I'm going to put my image here, I find this a really like global generic thing, it doesn't really fit into layout. It's definitely not prototype biography. It's

a sort of my general thing that lives up here at the top MSA max width is 100%. I'm also

going to put a display of block on here. I'm doing it now not to confuse you. But just

to say I remember when I said images are usually display inline, it does lead to this issue where there's a small when something when an image is display, inline, which it is,

by default, it gets this little little space underneath that I'm not going to worry about why that happens right now. But what what it means is, when you're setting spacing on

them, it doesn't always line up exactly how you want it to. So a really, this is like

every site you ever do, literally, you might do something like this, where it just solves

98% of the problems you'll ever have with an image, we have one other thing we're going to do to deal with the cropping on them. But this is just like, if you're setting up your

images, these two things will just make your life a lot easier. So I sort of recommend always having it's like body this image this and then you're done. So if we go and take

a look, now, hey, look, things have already gotten much, much, much better. I also made another mistake in my typography, we'll fix that when we get to our widget. So the last

thing we'll do is sort of set the proper sizes on those. So that was my main and my aside.

So I'm definitely going to come down to my layout, layout, containers really sort of generic, then I have my header, this is all header, we might as well go right here, I

have my navigation, I have this in its own category, I might, you know, depending on how you want to work, you might keep your navigation up here with your header stuff.

Because the navigations in there, I'm just gonna come right here and say my main and we have my aside, I want you to set some sizes on this, see how we can look and see if we

end up agreeing on the sizes we're going to put on this. Alright, so on my main on this

one, I'm going to put a width of 75%. And on my side, I'm gonna give this a width of

20%. That's good. Take a look now may look a little bit wider than when we needed to be. But that's okay. I think overall, we're sort of hitting where we what we need and

what we want it to look like. So in C, it's working. Now we do need to build a breakpoint

into this as well, right? Because Oh man, that sucks when we get to the small screens. And I know that there's a problem my navigation, I want you to try and fix it and see what

it is see if you can figure out what it is because we have the right code. But now there's an issues, see if you can solve it. Did you find it? This is finding little changes, whether

it's a typo, or just sort of running through things quickly and trying to find issues is

a really, really good skill to have. It's something you can get pretty good at typos are like spelling color wrong, or just setting the wrong property on something spelling direction,

but you mix up the C and the T, things like that knows a lot of problems. Now in this case, that's not exactly what it was. But it's because I renamed my container flex.

But in my media query, I never redefine that and change the name here. So remember, we had container nav, we change it to container flex, I need to do that here too, to make

sure that it's working. And hey, look at that we have that working wonderfully. But we've run into a problem of we still have the wrong widths on things. Now these are using something

called max width instead of min width. And I mentioned that we sort of want to be doing

a mobile first approach to things instead of working from the big screen to the small screen, it generally makes life a lot easier, you end up writing less CSS, because the defaults

are a bit more of your friend. So in the next video, we're going to explore how we can switch our min width to a max width and a little bit of refactoring we can do and then you're

going to see how much easier it just makes it so you're we're not overriding as much we can write a lot less code. So I'm actually going to take this. So to start thinking mobile

first, what I'm actually going to do is completely remove my media query. So let's take this

out and take that out. So we're left with this. Now we don't want this code just floating around. Like that. So I'm actually going to take this text align center and make it the

new default. And I'm going to take this flex direction column, and I'm going to make that

the new default. And we can delete this extra stuff here. What I'm also going to do is I'm

going to delete these for the moment. And you're gonna see why. Because when I delete that, and if we hadn't had the media query there at all, and we go and take a look at

it, at small screen sizes, we're almost done. We've, we've sort of done the mobile first

design, there's there's not a lot to it here that we're missing, there's some spacing issues, we need a little bit of margin to separate some stuff, a few little decorations here

or there. This just needs the box, we need to fix the typography thing I talked about. But you're sort of 90% of the way there. The problem with working on the big screen first

is writing a bunch of code, and then you're overwriting it all at the small screen size, you're resetting your columns back to what they were before, you're changing things to

be back the way, if you hadn't done it, it's exactly what it would look like. Now the problem is, when we get to the big screens, it just starts stretching and not looking so nice.

So what we want to do is build in that point where all of a sudden, at one point we go,

Okay, we're big enough, now we're going to add the two columns in there. So that's where the media query comes in. So we can add that back in now at media. But what I'm going to

say this time, is, we're going to do a min width instead of a max width of 675 pixels.

So now we can say our container, flex, flex direction, is row. So right away by doing

that, if we come and take a look, when we hit the right screen size, oh, now we have two columns, and we have one, and it's working pretty good. Now, I never said any sizes on

this, this is just the defaults that are popping up right now is now I can say my main cause

the width of 75. My side has the width of 20. And that's it, I don't have to worry about

Whoops, I don't want that there. I don't have to worry about overriding this anywhere, I don't have to go and all of a sudden say, whoop, I also need to now immediately like

I don't have this as my default that I then need to reset up to 100%. So we've just said at one time, we've redefined it for the big screen because the small screen was perfect.

And in general, small screens are much simpler. So it's a lot easier to do all of your small screen stuff, and then add on a few little things here and there afterward. So now if

we go look, we have our gap between there and everything is starting to fall into place.

spacing on things isn't the most work we sort of have to do is with this, we need to add

this line that's going to be coming through on here and just add some spacing between our articles in general, because if we come and take a look at them, we'll go to our small

screen. And we are not spaced out properly, the spacing on things isn't very good. If

## 33. Starting to think mobile first

these are really stuck together. So we're going to add in that spacing that we need right now, when we come and we look at the original design, we do have a lot more space

are on my date, this spacing actually looks pretty good. But we'd need a lot more space here in here with this line coming through. So let's start with just getting the space

around my date and make three comments thing that's right there. Now where would we do that? We don't even need to go into our article. To do that right now we need the first part,

since we're just looking at something in there is actually in the typography so we can come back up. And that is my article info, we don't have a selector for it. So I'm just going

to copy that. We have the read more, read more. So we can come down here and put my

article info, I want you to create the spacing for it with one that line of CSS don't want you to use the shorthand and add in that empty space. So you weren't sure if it's margin

or padding. Again, people mix those up all the time. But I think you won't have confused it yet. Because we've been using almost exclusively margin up until now once we start doing more

projects with padding, you might start mixing them a little bit, the spacing is pretty big on it, I want to make it even on the two sides. This is where it depends a little bit how

you like to work, though. And the reason I'm saying that is you could some I tend to always

like putting margin bottoms only. But I have exceptions. Because if you only deal with

margin bottoms, it definitely makes your life a little bit easier. But sometimes it's nice to have a margin top just to simplify things. So I'm going to do that. So I don't have to

add a margin bottom on my image. And I'm just going to put this space and this space all on the date, it looks quite a bit bigger than my font size. So I'm gonna start with like

a 2.5 M and a zero, maybe it's too big, I think it might be but I tend to go bigger and then shrink down. And there's a really good reason that I do that. A lot of the time

with margins, there's this thing that happens where you always end up putting them too small and then you think it's not bad, but maybe no just go bigger always go bigger than you

think. So start with a big number and slowly reduce until you're happy with it. Oh, it does look a little bit big. So let's just drop that down to two. And I think that looks

much better. And here's the fun part and that is going to be Adding in the underlying after

our continue reading, here's that line that goes all the way across the bottom, I want you to think about where you do that, how you do that, how you pull it off how you create

the spacing that you need on the two sides of that line. So what you know, if you have to jump into the index, look at the markup a little bit, try and analyze it and see where

you'd put that and how you think you might be able to pull that off. If you get a little bit stuck, that's fine. There's a bit of a trick to this one. It's not complicated, but

it's not always something you'd think of when you're first starting off with this stuff.

So hope you had success with that, and you figured it out. If not, then really don't worry about it. But what I'm going to do is in my layout, I'm going to come down and down

## 34. Styling the featured article

and down and down. And we're going to create a new section here called articles, because

I am going into my article itself. And I'm breaking that down a little bit now. So for

this, I want to take my article featured, because this is the only one that has the underline on it. And I'm going to add a border bottom to it. So that's going to create that

line a lot like we did on remember negation actually. So border bottom, and we can give it the color that we need. We'll give it one pixel. And we'll make it a solid line. And

let's go and take a look at how that looks. So there we go, we have the line that's actually

showing up, but the spacing and everything is off. It's just not where exactly we need it to be. So how can we fix that. But we need two different things. The first one is I'm

going to add padding, but I only want the padding on the bottom. And I'm going to do with the shorthand just illustrate why. So I'm going to do one m of padding, and we'll

go take a look. And you can see it's pushing everything inward. And you know what, let's also give this a background color to make it super clear what's going on. So we're gonna

give this a pink background, so we can look at what's happening. So we can see the size of my featured articles exactly the same. But when I added that padding, it pushed everything

inside of that box a little bit. And if we make that a bigger number, it's gonna push it even more. So we're pushing all inside. But you'll notice the border is staying on

the outside of my padding. So the borders all the way over here. So if this padding was only on the bottom, instead of being on all the sides, it would just be creating empty

space right here where we want it. So I can take this down to maybe back down to a one.

And I can also switch it over to padding bottom, if you want to use the padding shorthand, it's perfectly fine there mentioned, if you're only putting it on one side, I don't think

personally, there's anything wrong with using the padding, bottom or padding, right or that type of thing, it's when you're using two or three sides, it's a little bit faster to

actually do it with the shorthand. But if you use the shorthand for everything, it's kind of cool to be consistent, so I don't blame you. Um, so you can see it has created

that space we want my border is right there. Now, I also want to increase this space a little bit. And we can do that with my margin. So it's an margin bottom. And I'm gonna make

it way too big for now, just to illustrate that it is working. And so you can see this is my pink box, the padding was working inside. So that's why we have a gap right here, then

I have my border, and then I have my margin after. So if you go way back to when we learned about the box model, it was always padding first, then a border, then a margin, we don't

use borders terribly often. So sometimes we forget where they flow into the whole box model thing. So if you didn't get this, it's completely fine. Now this is too big of a

space, I'm going to drop that down to it too. And of course, turn off the pink background. And you know what, I switched that, but I think a two and two looks a little bit better,

just to make this spacing completely equal. I think it'll look better in the long run if we do that. So there we go. I think that is great for my featured article, everything

is good there, then we're going to jump into styling this one here. And this is where we

had that trick that I talked about. And actually, you know, I think they would do better in its own video, because it's a little bit weird. And I want to be able to focus on it or find

this video again, if you need to reference it. So I'll see you in the next video, we'll we'll take a look at how we can change the order of the content in there to match the

design where the image is actually on top. And when we look at this one, the text is

what we currently have. And so this image on the left is what we currently have. And we're aiming something that looks a little bit more like this. And you can see there's

a pretty big difference between the two of them right now with the order of everything. And I said there's a good reason for that. So in this one, we're just going to fix the

order of it, then we're going to worry about positioning things a little bit within that space. If we come down and we take a look. Let's go back to our markup here and come

down to our recent articles. So we've done is created our article recent. inside of our

article recent we have the main and we have the recent secondary. Now this is gonna be really useful, we get to the big screen sizes because we want two columns. That's the main

reason I created these two separate divs is so at large screens, we can have the two columns. Right now though, we only have one column because we're on small screens, but we need

to rearrange the order of everything. So what I want to do is I want to give let's come

up so we'll come over here to our styles and we can do our article. Recent with display

flex, we can change the order but First, when I do display flex, it has created two columns.

So I want you to go ahead now and change that back. So it wouldn't be two columns, I want them to stack one on top of each other. So we do that with our flex direction and switch

that over to a column. And now we have it back pretty much exactly how we had it before.

But now we can play around with ordering. And this is pretty cool. So what I'm going to do is to change the order of something, we go and we do it on the child itself. So

we have our article, recent main, and we had the article, recent secondary. And on different

things within a flex, you can literally apply an order to them. So if I said order two,

and I can come on this and say order. One, it's going to switch the order of them, you

might have even noticed that the order of it like sort of jumped around. So we're going to take a really quick look, you can see that it actually has changed our images on top

and everything else is on the bottom. And that's awesome. That's super cool, right? So why would we do this? Again, it's to be focused on keeping the markup making more

sense and the logical order here, as if if there was no CSS that loaded in so somebody

could still read it in a way that makes sense, as if there's no CSS. And if we want to make visual changes, we can, if we want to make visual reordering, we can because we have

a hierarchy that we were following, that still is drawing the eye to the right place, if the CSS were to fail, it would make a lot more sense the order that we see it in here.

Now, closer to when you are the first it's going to come you're literally saying this should be first this should be second, this should be third, just for fun, if you wanted

to, you could try doing what would happen if I gave it zero, well guess what it means it's smaller than one. So it's, it's going first. Or then what happens if I give this

a negative one? Well, that's smaller now. So it goes first. So it's the smaller your

number, the earlier it is, and you could have 100 items, and you could give them all an order. And that would be a nightmare to control and to set up. But if that's what floats your

## 35. Changing the visual order with flex box

boat, go for it. But I'm going to stick with my two and a one. If you have two things that

have equal order, they're going to go in the order, they were originally in the markup. So they're going to follow whatever they had here because they have the same value. So

if you don't apply anything, now we'll switch this one back to a two, it might even be worth putting a comment in your code just to explain why you're bothering to do this. If you think

it's not clear enough, or if you'd be coming back later, and maybe think you might be confused by something like that. This lesson is focused really on order. But I do want to just finish

up a little bit on here. So the other thing I want to do is on my article recent just add a little bit of margin bottom to them have say to em to create that space here,

just so it's not sticking to the image of the next one that we have in there. Because if not, it doesn't look very nice. So just adding in that empty space is a good thing.

And again, go with a bigger margin than you might expect. We do have a layout to be basing things on if you're looking back at your layout as well. The last thing while we're here,

though, is you might be going well, you know, we just added some space here. But the last thing we might you might be wondering about is this gigantic space that we're getting

here. Part of it is we have our h2 here that has a margin top and we have this which we

added a big margin bottom on, and that's causing some problems. And you might be going well shouldn't those margins be collapsing, and then normally, they would be collapsing. But

there's something weird with Flexbox and collapsing margins, they don't collapse anymore. That's

only for the direct children. So I'm not going to get too much into that we're definitely going to run into more situations with it. But we need the fix for this for the for this

little timeframe anyway is the fix which we need to do on our site at large. Anyways,

I'm just gonna scroll all the way up to my h1 h2 h3 here and say that it should have

a margin top of zero because we're going to run into other issues with that, if not anyway, so we go and take a look at that just sort of sucks everything up a little bit closer

and looks a little bit nicer. I'm going to explore how you could potentially get that to be even closer. When we look at the media query in the next one where we're going to

pull this over to the right side and suck that up even more. That's an exploration video

more than a best practice video, but we'll see that consequence didn't notice that there

was one unintended consequence didn't notice before, which was we got rid of the spacing on our titles, and that got this stuck on this. We'll fix this in this video as well.

But the main focus will be here, like I mentioned, getting it to pop on over to this site at

certain sizes. The way I'm going to do it now the reason I'm saying it's not a best practice video as much as a exploration video is because if you had a really long title,

this would cause some problems. And there's the potential for overlapping but I just want to show you something that we haven't looked at yet. It's something that I try and avoid

when I can but I want to show it to you anyway and I think is fun for the layout purposes on this one and something that's worth exploring because while you try and avoid it, it's good

to know that It is a possibility. And that is negative margins. So I'm gonna go all the way down to my layout area. And we're gonna go all the way down until we find my articles.

There we go. And I'm going to create a new media query, because we're going to be dealing with these two, pretty much. So at media, as far as the size of where do I create this

media query for, don't create media queries based on the size of certain devices or set

devices, because then you can always do things that you want to do, or those devices, you know, people will make too many media queries, because they're trying to match certain devices

when they don't need one, or in this case, I'm trying to really base it on my layout, I'm sort of just guessing at the size, we're gonna tweak it a little bit as we go. So I'm

going to do 500 pixels for now. And let's see if you see how it goes from there. So

inside my media query, what I want to do is select my article recent main, and I'm going

to give this a margin. Top of negative, I don't know what's out with negative one M

and just see how that affects things. So you can see they sort of gotten a little closer together, let's just try changing that maybe to negative two, insert seeing these are starting

to overlap, and we can bring that maybe up to a whole negative three. And they're starting to line up how we want them to I'm going to switch that to a 2.5 because I think it's

gonna be bang on how I want the top of the text is sort of lining up. Now, as I mentioned,

this can create text overlap, so it's probably not a best practice, overall cure. Yeah, so

this is my longest one. So I'm gonna base it on this title. But what would happen if you had a really long title, it would break this whole layout. So I'm going to show you

how we can do it with a negative margin. So you know, if I bring this up really big, you can see it's pulling it up on top of other things. And it's one of the problems with

negative margins. But it is important to know you can have a negative margin because sometimes it does come into use, and it is useful to have. So that is perfect. Now the other thing

I'm going to do in this media query is do my article info, and say there's a text align

to the right. And there we go, we have the layout that I wanted, where we have those

going like that. Now the problem is, where's the media query coming in, where's 550 pixels, so we're going to shrink, and it's really, really close. But on this one, it's looking

perfect. So you know, it's one of those ones where you have to judge it by eye a little bit, I'm going to pump it up to 550, just because I don't want those to get so close

## 36. Playing with the title’s position, and the downsides of negative margins

to each other. So it gives us this small little range where we have two columns, and then we're pops over back to the big layout, where we're gonna have to change what that looks

like anyway. So that is how we can go and do something like that with a little bit of hackery to give herself this fake two column look based on the way our markup is. But again,

make sure you keep this really, really simple because it's keeping cooking simple. But if this was how to keep your cooking simple, all of a sudden, I'm a little worried. We

have overlap, and it's causing problems. So while negative margins can be fun for creating cool layouts, or cool effects, sometimes and sometimes you want overlapping content, it's

always possible, you want two things to sort of overlap. And there's other ways of doing it with positioning as well, which we're going to explore later. But just when you use the

negative margin be very, very aware that this is a possibility. So I think what I'm going to do is while we've explored it, what we've seen it if you want to sort of rebuild this

on your own at one point, and keep it in your code, because I do think it looks really nice with how I've set this up, it's just not something we can do in a responsible way. So we've looked

at how to do it. But now we're just going to take it all off. Because I think it's going to cause more problems in the long run. And it's really important that you keep your code

as functional as possible, and that it can accommodate for all these situations. So if

you're ever putting together a demo site, make sure you don't only use short text, it's really if you're doing a business card template, don't use a really short name, KEVIN POWELL

is not a good name to use if you want to get a really, really long name on the template to make sure that he can work with super long names. And it's not running into issues. So

the same thing here, we you want to test the short titles and long titles to make sure everything will work with your code. now is use a really, so what we're going to do now

is use a really interesting property called object fit. It's a relatively new properties

as far as properties go. And it is really, really awesome. It's actually something that existed for background images, which we haven't even looked at yet, which is crazy. We're

doing all this cool stuff, we haven't even done a background image yet, I'm only going to do it inside the media query. Because I think the screen sizes, I just want to see

my complete image. I think that looks nice that we have a nice big image like that. But when I get to these two columns, I don't want my image to be so tiny, I want to I want to

make it taller, but I don't if I make this taller, it's also going to have to be wider because we're you know, we're keeping the image in proportion. I'm going to cheat a

little here. These are article images, Article image, I think is what I call it. I'm going

to give them a width of 100%. But I'm in the height of 300% just for fun and to show you

what happens. So we can look at the super stretch there so overlapping each other which is a bit of a problem. But we can see that these images are actually Super stretched

out. So I can't make my images taller without the risk of stretching them like this, unless

we bring in something called object fit. In object fit, we have a few different options,

the one that you're probably always going to use is called cover, when we use cover when it's actually going to do is it's gonna crop my image inside of that space. So let's

fix this to be a little bit more of a realistic number. Let's put like 400 pixels here. And

you can see it's cutting off the sides of my image, my image is much bigger, my image should look like this. This is what the original one looked like right here. Whereas on this

version here, I can see that it's cropping the image. Whereas if I turn off the object fit cover, it's squishing the image. And this looks terrible. I never want to see this on

a website, if you do this on a website, you show it to me, I won't be proud of myself, because I didn't teach you properly, I made a mistake if you do this, so do not squish

your images. But this object fit cover, it saves us it lets us pull off controlling the

size of an image. Now the one thing is it will crop your image. So well these two are looking great here, we're sort of we have this little leaf sticking out the side, it

isn't ideal. Now one thing you can do is you can also do an object position. And I can

say left. And what that means is it's keeping the left side of my imaging instead of the

center of the image, which is the default, but now I've ruined this image. And I probably ruin that image as well, because it's keeping the left side and left side of this. So we're

just seeing like this part of my image here right now, which we don't want, that doesn't look very good. So for this one, I am going to leave it as just the object fit cover,

I'm not going to bother with my object position. But I do want you to be aware that that is an option. And I think I'm going to bring this down to like 250 pixels. And I'm also

going to give this the minimum height of 250 pixels. If it needs to get bigger, for some

reason, sure, why not. But I want to make sure, at minimum, the height is 250 pixels.

And I think that looks a little bit closer to what we had in our layout. Maybe we could make that 300 instead of 250. You can see I think at the smaller screens, that it's

## 37. Changing the visual order with flex box

looking pretty good, it's getting narrow, it has a nice aspect ratio to it. And then as it gets bigger, I'm pretty happy with that, if you'd prefer a different number, you can

play around with it a little bit, and find something that you think looks a little bit nicer, it's completely up to you. But I'm gonna stick with that, I think that's a really

cool property, a nice little trick for letting us bring in one image in two different places and having them cropped completely differently. And having it work at the other screen size.

And it's still showing up in here. So like this is the same image that we have here, which is really, really cool and handy that we can do that all with one image instead

of having to load in three different versions of the same image. Now, sometimes you might want to do that there are ways of actually doing loading in different images based on

different screen sizes. But it's a nice little trick early on. While we're just getting our feet wet with all of this, this is a nice easy way to be able to do a problem here,

we're fine. Let's start by fixing this problem here we're finding simplicity and life is stuck to the top, you get 100% do this is the margin top somewhere here just to push

things down. But I like trying to use margin bottoms as much as possible to create space, instead of mixing up margin top sometimes the margin bottom Other times, there's always

going to be exceptions, there's always times where maybe you won't do that. But I think this makes the most sense. Because you don't know if you're always gonna have a title as

your very first thing, maybe you have something else that's actually coming up here as the very first thing. So I think the best course of action is actually to put a margin bottom

on this rather than having a margin top on this because again, maybe another page, the first thing will be an image or it'll be something other than your your title here, I think it's

just a bit more robust, doing that. So this doesn't even have to go anywhere, you just do this on our nav nav nav header, I think we have a header there, we do. So we can just

come on here and add a margin bottom of I'll do three m to give us a very generous space,

because so the spacing here is here looks more or less equal, I think it just looks nice and balanced. If you'd want that to be too I wouldn't argue with you, I think you

could pull that off as well. So that fixes that. But the more important thing for this video is, while our recent articles are looking good at small screen sizes, when we get to

big screen sizes, we want this to become a two column layout. That just looks quite a bit different. So let's go and do that right, go all the way down to the bottom because

I'm in my article section. And I want to create a media query. So I'm going to ask you to create your media query first on here and actually see if you can remember how to turn

it from a one column into a two column all on your own. I'm going to use the same breakpoint that I used for my navigation here because I want everything changing at the same time.

I think it's a bit more consistent when you do that way. Instead of just having all these random different numbers throughout. Of course, that does depend on the layout. You saw in

the last video, I use a different number for my breakpoint, but in general, I try and stick with the same breakpoints just for consistency and it makes my life a little easier to Did

you get it to work? I hope you did. So I'm going to come in here and create my media query. So It's an ad media, I want my parentheses and then my curly braces, I'm gonna come back

up into here and do my min width 675. And now we can come down here and add in what

we want. Now, don't do the mistake where we just start writing stuff. Obviously, I've done that a million times, but we have to choose the selector we want first. So the

selector I want for now is article recent. Now, we've already said that article recent

is display flex, so we don't have to declare it again, if you did that, where you put display flex, here, it's fine, it's not going to do anything, it doesn't cause any problems. But

since we already have declared it, here is the general rule, we don't have to read Eclair it again, once we're inside of our media query here, all they want to do is redefine my flex

direction, so I can do my flex direction. And his column, I'm going to switch that over to a row. And right away, we should see now we have two rows of content in here. Now we

can come on to my article, recent main, we're gonna want to do a little bit of work on that,

and my article, recent, secondary, and do a little bit of work on that one as well.

## 38. Styling recent articles for large screens

So I want the secondary one to actually be pretty small. So I'm going to give this one a width of about 25%. If I go and look though, it's gonna be tiny, tiny, tiny, tiny and this

is Flexbox, trying to be smart about things, because it knows this one should be 100%.

Why is that 100%? Though, think about it, why is this one trying to be 100%? It's trying

to be 100%, because we never gave it a default width. And by default, any block level element will be set to 100%. So 100 and 125. It can't fit in this area. So it's doing its best to

go Okay, well, this one's allowed to be smaller. So I'm shrinking it way down is sort of how Flexbox thinks we're going to get more into actually what's happening with this a little

bit later on. But for now, what I'm just going to do is come on here and say this one has a width of 70%. And if you go and look, the only problem is we have all this leftover

space right here, I don't want leftover space here, I want that leftover space to be in between the two of them. How can we do that? Do you remember, go ahead and do it if you

do. So to put the space between them, instead of all the way in the right, I can come on

here and do a justify content space between my doing that. Now I get the gap coming here

instead, you might say it's not reaching the end here. But it's because the lines are breaking,

if I have a long word that can't fit here, it's just going to go down to that side. So it really is making it all the way out to the edge, it just really depends on the the

text that you have in there. And the size of the screen that you have it, sometimes it'll look like it's closer. And other times it might look in again, it's just based on

where the words are breaking. I actually don't like that breakdown of stuff so much. So I'm

going to make this one a little bit bigger, I'm gonna push you up to 30. And I think I'll bring this one down to like a 68, or 67. Because I think that space can be a little bit tighter

like that. So cool. That's looking pretty good. We have my two columns that are working. And when we get to the small screen sizes, everything is stacking and our order is looking

fine there. I know our order is looking fine here. So everything is fantastic. Except for

one thing. And the one thing that's not fantastic is the order here is not good. If we come and take a look, when we're a big screens, we want our image all the way at the top,

and we want to have an info here, then we want to have the title. And that's just not what we have going on here because I put it all in the wrong order when I did the markup.

So what I want you to do is we can work inside of this media query, I want you to play around and get that to work on large screens to be in the right order for everything. Of course,

once you've done that, I'll go ahead and look at how we can do it. But it's all things we already know how to do. So go ahead and tackle that. So I hope you found it if you didn't

Don't worry too much about it because it was trying to trick you a little bit with this one and make you think a little bit. So if we come and look at my markup, we have the

featured article here. My feature article has everything we need in it. But we want to change the order of these and see you did article title just for fun. If I came down

to here and I say article, title, order 99. That should push it all the way to the end

should become the last thing everywhere. But nothing happens. It's not actually changing. So if you were playing around with the order of things, and you're like, why isn't it changing?

It's normal. The reason it's not changing is we can't use order unless we have display

flex on something already. So what I need to do is article featured display flex. But

of course when I do that, it just causes chaos because now everything is becoming a column I even got some so it's just a mess. So I want you to fix that now. And once you've

done that, if you weren't able to get the order working, go in and put the order on it and we'll take a look at the rest of it together in a second. Alright, so fixed the

first thing was the flex direction. We want that to be a not a row a column. And that

should at least fix the craziness that was going on before and everything sort of falls back to how it was more or less and then what we can Do my article image has an order of

negative one. Now I'm actually going to make this a negative two, because I also want to give my article info and order of negative one, to pull it up to in between those two.

Because if I don't include that, it's first going to be my image, then it's my title,

then it's this, and I want this to be on top there, whoops, on my console, so I can have

that negative one on there. So first, I have my image, then I have this, then I have my

title, and then I can get into the rest of it. And it's starting to look pretty good. I'm pretty happy. Overall, with how this is starting to come together, I think it's looking

really the Adobe XD for our widgets, we have the sidebar widgets that we created here.

So we have the div class sidebar widget, and we want to make them look a little bit like this. So they have a really big thick border on them. The color is not this dark gray is

a lighter gray than that, I'll let you experiment a little bit. Or if you link to the Adobe XD file, you should be able to get the exact color that that is, as well as the exact thickness,

I'm just going to estimate it and hope for the best. The one thing I'm a little bit worried about with our current layout, though, is this space is already pretty narrow here.

And it's going to get a little bit tighter once we add that MP once we add the border and that spacing inside of it. So what I'm going to do is I'm actually going to right

now change the size of those before we get into those widgets. Because I am scared we're

going to run out a room for them, I just think it's gonna be a little bit too tight. So here, I'm going to change this width down to a 70. And this one up to a 25. I don't know what

might have to readjust again after but I think it's going to help us out just by giving us a little bit of extra space there. And now what I want you to do, let's go all the way

back down. And I'm going to create a little comment here saying with jets. And I want you to come down and style them as much as you can, it's using a bunch of stuff we've

already looked at. So go ahead and try your best at least adding the border as well as that space on the inside. And if you feel up to it also doing the typography once again,

I didn't give it to you here but we know what the fonts are, you can adjust and play with the font size. If you do look at the Adobe XD link, you will be able to get the actual

font size that was used, but it will be in pixels from there. So make sure you convert it over to rim. If you're not sure how to do that you can watch when I do it, because

in just a few seconds after I give it to you, I'll be doing it myself. As usual, they'll try and do as much as you can and see if you can pull it off before I take over. So I hope

you have a lot of success. Now let's dive in. And it's not just widgets, it's my sidebar widget. And the first thing we do is give it the border. So I'm gonna give this 20 pixels

solid. And for the color, I'm just guessing right now, but I know it's a really light gray. And I know EF EF f is a light gray because I've been doing this for a long time. So we

can see that it's looking pretty good. Actually, I'm happy with that. Now they're also stuck together. So I'm going to add a margin bottom on this to help out a little bit. So margin

bottom, I'm going to try one M and see what it looks like. And it's not bad, it's probably

a little bit too tight. So it's boosted up to a two. But before we go and look at it again, you might be going, Kevin, why are using pixels here. But m here, borders are

one of the few places where I have a habit of using pixels. Because a lot of the time we have a one pixel border, maybe a two pixel border, it's really rare that you're gonna

have a really big size on a border like this. Now, as you can see here, it has happened, but it's not too much. So I just fall in the habit of using pixels because one pixel in

M is just not doing that. If you'd rather set this because it's a bigger border and M or M and M and that's what you did before then by all means go for it, it would work

beautifully as well. And before we go and check it out, though, I'm also going to add some padding, and we'll try one aim of padding on that. And that doesn't look too bad. I'm

pretty happy with how that looks. But let's just see, when we get to the smaller screens, if we run into any problems, this font size is going to get smaller, these are getting

really, really narrow. But overall, nothing's broken the image, the titles are a little bit too big, the images are really, really small. And we're sort of running into two

## 39. Setting up the widgets and talking breakpoints

options. Now when we're making something responsive like this, because it seemed to be working super well. I think at the large screen size, it's exactly what I want it to look like.

So I don't want to modify too much, we sort of have two options. One of them is to change our breakpoint. And if we do that we're going to be going through potentially changing it

in multiple places. I don't really feel like doing that because I don't want to have to reconfigure a whole bunch of things. So another option we have is to come on this and actually

make sure it can't get too small. Because right now what's happening is it's clearly just getting too small. Whereas these are working fine. They're not ideal, but they're

working. Okay. So I think that's what I'm going to do. So if we come on to my widget, let's go and find not my widget but my sidebar. So here we had our main and our aside, I think

I'm going to give this a minimum width. So I want it ideally to be the 25% but I also don't want it to get smaller than 250 pixels. So well. It's even too big. I think There's

one disadvantage that when we do this, and that's that that justify or justify content

space between stops having enough of an effect on it, because at one point, when we lock in that size at one point, they can actually hit each other, because now that's locked

in at that minimum size of 200 pixels, which I think is roughly the right size. Because here, it doesn't look super ridiculously small, it's borderline, maybe I could actually change

this breakpoint to be a little bit bigger. But while we're here, why don't we just come on the side here and add a margin left of one M to the whole thing, which is not a lot

of space. But we don't want too much when we're at this really small screen size anyway. So at the big sizes, I think it's going to look pretty good. And then we get smaller,

it's just going to prevent the two columns from colliding into each other by having that one mm space on the side there. So now we get smaller. Again, maybe this is a good point

when we could be modifying the breakpoint on it if we really wanted to, but I'm not in not hating necessarily. And there we go. And this comes into a really good discussion

about when you want to have your breakpoints. And it's really about adjusting to the layout and when things are working or not working. So if you really don't like this, you could

come in and adjust your breakpoint at this point. I'm gonna say it again, I've said I've said this before, I'm gonna keep saying it. Again, breakpoints are not device specific.

They are layout specific. When your layout is breaking, when your layout is not looking good. That's when you go ahead and create a new breakpoint or adjust a breakpoint that

you already have. It's about your layout, it's not about what device, it's looking at hundreds, maybe 1000s of different phones. Now, they all have a little bit of a different

size. There's so many tablets, there's laptops, and these little laptops. And there's big laptops, you don't know the device size somebody is on. So don't worry about finding specific

numbers to use, and adjust it where your layout is working and just make it work at the different screen sizes. However you can I'm pretty happy with this. If somebody happens to be here,

there might not be a million people with this device width on it. But overall, I think I'm happy with that and how the whole thing is working, we could also allow this to get a

little bit wider, I just wanted to make sure that we're limiting the max width of it. For some people who are doing this tutorial on a smaller screen, they can see it reaching

that maximum width. I'm gonna end this lesson here, because I think we covered some easy things with our setting that up. The last thing we're gonna do in the next one is finishing

up everything in the site. So we're gonna fix our titles, we're going to fix the order on this. And the one forgotten element is our footer all the way at the bottom, I think

we can knock that all out of the park really, really quickly. So let's wrap this tiling

the titles, they need to be adjusted a little bit. So let's go and do that. And let's go and look at our index first. What do we call them, we call that widget title. So this is

where we have an h2 that it's really useful to have a class on. Because pretty much all

the styling. On our other h twos, we didn't need to do very much on these, right, because

when we look back at our styles, and we go all the way back up in our file here, a lot of it is handled right here on this h2, we set up the font family, we set up that we

set up the color, we set up the margin, we did everything here. But we're still dealing with an h2, this one just happens to visually look completely different. So this is where

it's really really handy. To have this, we have all of our articles stuff, let's come right here, widget title. And let's give this one a font size of one REM, the colors probably

Alright, but we need to change the font family over to Ubuntu just there we go. And we also

want to make the font weight to be 700. Because we want it to be not 7700, we want it to be

bold, I don't even know how bold 7000 might be. So there we go, that's looking not too

bad. Let's just go look back at the original design, maybe the font size can be slightly bigger. But I'm gonna stick with that. I'm pretty happy with it actually, just based

on looking at it visually. We want to style these now and actually change the order of

things over here. So how can we do that? Let's go and take a look back at our markup and see how we did it. So we had a div of widget recent post. And then we have the order of

our widget recent title, you know what I actually think is widget recent title. So that's looking

good. Now these are too big. And I think these are actually supposed to be the same size as this. So let's come back up to our CSS file here. And widget title, I think what

we could do is keep my widget title, but also have my post title. Don't forget the dot and

give those the font size, font size of one REM here. So we don't need to have it necessarily

on that one. So that it helps shrink down those a little bit. Just visually they're not as in our face. And what you need to do is change the order of them because this should

be underneath the image and not on top of the image. It doesn't look right like now the spacing is all off. So we've already looked at how we can do that when we did all those

other order things. Give it a try. I hope you got it to work. If you didn't. Let's go

and look at our markup, the one reason you might not have is we have this widget recent post. So this widget Recent Posts needs to have the display flex on it. So we can change

the order of things that are inside of it. So I'm going to come here, I'm just going to go all the way down where we have our widgets. Because now I am looking at layout. Don't

forget the.at the front. And we can give this a display of flex. And when we do that, it

will make columns, we don't want columns. So flex direction, will become column. So

the things inside our rows, and they're stacking on top of each other. So it goes sort of back to how they were. But now we have the advantage of having the order on these now. And realistically,

I might not actually switch the order on them, like I did here, I might have just put image

## 40. Using a new pseudo-class to wrap-up the homepage

and then title, I think it's fine in the markup to do it that way, I just wanted you to practice a little bit more and play with the order a little bit more, since it was a new concept

coming into this project. so here we can say that my widget image has an order of negative

one, and that should have jumped it over to the top, there we go. So we have that we have that. Now we're just missing one thing. So we can see those here, there's these little

little thin lines that are coming in between on those sets, we need a division here in a division here, there is a trick though, because this one, we don't want to do it.

That's okay though. For now add one here, add one here, I'm going to show you a trick to get rid of that last one. Okay, so let's go and add those in, I'm going to do it like

I did before, where I'm just gonna add a border. So we're gonna say border, one pixel solid.

And for the color, we'll go with that dark Ray that we've already been using. If we go and look at it, now we have a line Oh, no, it's on all four sides, I only want it to

be on the bottom. So let's fix that right away and go look, again, that's looking nice, but I need a little bit of space underneath. And I want some space in here. So we can right

away at a margin bottom. And we'll just go with one m, let's see, I use a lot of training

round numbers, one m one and a half for M to ram, it helps with the consistency along the way. And doesn't that just we go, that looks pretty good, I think except we don't

want this one here, this one is going to cause us some problems. So instead of styling it

like this, so I'm going to show you a trick, something we haven't looked at before, I'm going to do a widget recent post. Last Child. So this is another pseudo class, this is like

hover and active and visited. But it's saying if it's the last child inside of something, I'm gonna say border zero, and I'm gonna say margin of zero, because we want to take that

margin bottom off, so it doesn't add extra space. So I'll explain this in a second. But let's go and look. And there we go. Now everything is getting balanced out, we have a line here,

we have a line here, but we don't have a line coming on this one at the bottom. So to explain

what this is in a little bit more detail, let's go and look in my index. So we have

our widget, our widget here, and our widget here. And this is the last child inside of

this parent, if I were to come and add a paragraph here, example. And we were to come back. Now

that we have a another item in there, this is no longer the last child. So this now has a border on it. That's pretty fun, right? I think that's pretty cool. So it's always

about the state is this, if there is a widget recent post that happens to be the last child

somewhere, we're going to turn off that border in that margin that we just added here. But otherwise, they should always have it just like when we're hovering on something. So

we have a normal style in a certain state. And when our mouse is hovering on top of a link, we're changing the state this instead of looking at a mouse, it's looking at its

relationship to its parent and to its siblings. So it's a really, really fun and cool pseudo class that can come up on something. So let's go and take out that extra paragraph that's

on here. And I'm not going to save the I'm going to do the footer. Now I said we're going to wrap up the whole thing. And then the footer is so nice and easy. I never even put the

text in there. So there we go, we have our paragraphs inside the footer. And if we go and look, it's just has that text, the first one is bold, the second one isn't. So let's

wrap this one up in a strong tag. And close strong right there. And we can come into our

styles. I'm going to come up a little bit here all the way to here where I was doing like my header, my containers, all of that I'm actually gonna put the footer here because

the footer is this really generic item. That's just everywhere. So we're gonna give this one the background of that dark blue, we're going to give it a color. In this case, I'm

going to give it white for now I'm going to come back to this color later on when we're coming back in in the next module of this course, stepping up style, because we're going

to show you a cool trick if you set another color on it or want to actually you know, use a color picker or getting a specific value just to match that. That'd be cool. When we

do stepping up or style. I'm going to dive a little bit deeper into this. A text align center on there and some padding. We'll go with the three M and zero because the padding

looks pretty big on the top and bottom and there we go. It is Looking pretty good, the homepage is all wrapped up, and we're ready to move on to the other two pages, which are

going to just start by coming over to my favorites. And some of the recent articles page, which

is this one here, I'm not even gonna bother zooming in, it's so similar to this one, we can knock this whole thing out without even writing one line of CSS, we just have to literally

edit a little bit of markup. So for this Pedram and do is I'm going to start by coming over to my files here and clicking on new file, and creating a page called recent hyphen posts

dot html. Now, this could just be recent instead of recent posts, it's really up to you. But

if you're going to do recent posts, it's either a hyphen, or underscore, make sure there's no space in there. And what we want to do is come over to my index, I'm going to do

Ctrl A, or Command A, if you're on a Mac, to select everything. And then I'm going to Ctrl or Command C to copy it, I'm going to come over to my recent posts, and I'm gonna

paste that right in there. Because we have everything in place that we need for this page to work. The only difference between the two layouts is this one is styled just

like all my other recent articles, whereas on the homepage, it was done as a featured

article. So part of the reason I did this was to show you a little bit of how CSS gets

reused across our pages, we don't have to keep recreating and rewriting styles, we have everything we need literally everything we need. If this was a real page, it probably

be dynamic, and the content would be automatically put in there, we might have more than four posts on a page. And these two pages wouldn't be so close to each other in terms of content,

but layout wise and CSS wise, they'd still be identical to one another. So if we come through, and we look, here, we have my featured article. But I want that to be a recent article.

So I'm going to copy my recent article here, Command C, paste it in here. And then I'm going to start just taking little pieces. So if you do a Command or Ctrl x, it's a cut

instead of a copy, I can cut that title out. And then I can come and put that title right here. I can take my image, and I can put that image right here, I have my info. It goes

right there. And this paragraph, it goes right here. Continued reading is the same in both

of them. So we can just delete that whole thing and not even worry about it. And we can actually see that page yet because we haven't set up the links to be able to do

it. Because then the only place where those were linking together is in the navigation. And right now in my navigation, I just have these placeholder links. So what I want you

to do is fix the navigation links. So you can go back and forth when you're on one page or the other page make all of the navigation links work, except for the about page because

obviously we haven't done that one yet. So we are working. So on this one, we want to

have our index dot HTML here and on my recent posts, so we can add that in even though recent

posts, this is the page we're currently on. And then I'll come over to my index. And I want to do the exact same thing there. So whether you're copying and pasting, or you're

rewriting it again, here, something that a lot of people don't realize, or they don't always think about is that you need to have all of the links and all of your pages to

be updated. So that you can go back and forth between the different pages. So if we save that income, take a look at it. This is my homepage. And if I click on recent posts,

or Recent Posts is working perfectly, and everything is good there. And we haven't written one line of CSS, and we just finished a second page. So this shows you once you're finished

## 41. Creating the recent posts page

one page, it's exaggerating a little bit, you often have to write a little bit of extra CSS. But in this case, we didn't have to write any at all, we're going to go on to the About

Me page, which is a little bit different from the rest of them. So we might have to write a few extra styles. But overall, it is really similar. There isn't a lot for the amount

Me page, there isn't a lot to do once again, but there is a little bit that we need to do. So I'm going to give you the link to the Adobe XD file, once again in the show notes

as well as in the slides and just one second, just to make it easier to get the text but also to see what text is bold in here, because we do have quite a bit of text that you're

going to want to make bold when you do this one. And we also have these h threes that are different from all the other h threes. So that need their own style, their own color,

their own font size, and all of that setup on them. But otherwise, there isn't that much. Now the very beginning of this is exactly like we did before, we can go to one of our

pages, copy all the content and paste it into a new page. So I want you to try and do that and delete everything that you do not need, we can keep some stuff but we do need to delete

some other things. So go ahead and give it a try. Place the content and see if it works the way you expect it all to work. And of course as usual, I'm going to be jumping in

and taking a look at it with you in just a minute. Okay, so let's go and make that new

file. So new file, we can call it a boat mi dot html. Again, if you're going to have two

words in it, and you need namespacing do not put a space with your keyboard. Make sure that is hyphens separated on my recent posts. We can copy all that come over to my about

me and paste it all footers good. In my sidebar. The sidebar is a little bit different on this

one We do not need this first widget anymore, since we're on the About Me page. So we can delete that sidebar widget. And we don't need

any of the articles. But I am going to keep the main there. So let's come through and just delete all of these articles, they're all vanished, they're all gone. And we can

just come in here and start working directly in the main, I'm just going to fast forward, when I do this part, there's not a lot to it, it's just, you know, we just have to place

content and put the right tags on everything. So I'll be back with you in a second with all of the content in place. So all of the content is in place or ready to go now and

do the last finishing touches on this. So there's not a lot to do, if we come in Look, I've also been really nice, and I've set up all the links for you. So if you go and look

between all the different pages, I have put those links in there. So if we come through and look, now, all of the links are working, we can jump through between the different

pages, and everything is all awesome, there's only two things is I need to change the color of that looks like I made a little typo there somewhere too, we want to change the color

of my h threes, maybe change the spacing on them a tad. And of course, we want to make

some modifications on this picture. Because we looked at the original design, it should be able to crop itself, similar to how these images were cropping. So the first thing I

want you to do is go ahead and do that we've seen it with those other ones. If we come and look at my About Me page, I've given it the class of image full. The idea here is

that I see this page sort of acting as a template page as well. So just like these are sort

of templated pages, if you were to click on Continue reading, it can use this exact same layout for a blog instead of for, for the full blog post, instead of format, just my

About Me page, if you're going to do that, I'd probably have the article tag, the whole

thing wrapped in that. So article open all the way at the bottom article close. But in this case, it's not really an article, so I'm not going to include it. Now if you're

## 42. Setting up the About Me page

wondering how you could do that, where you're creating template pages, that's not in the scope of this course. But there are different ways there's languages like PHP, which WordPress

is built on. So you could actually make your own WordPress theme. Or I use a language called Jekyll for my personal site. Right now I'm looking at changing that maybe to eleventy,

or there's many, many others, there's a few of them that I'm looking into. For the next time I redo my own personal site, which are called static site generators, where it's

mostly using HTML and CSS. So you're setting everything up with those. But then you can have the content if you write a new blog post that sort of automatically putting the content

into the template file that you created. And we're well on our way of having something that could be taken to that next step once you're ready, but I wouldn't get there yet,

don't get ahead of yourselves, let's first master how to make the responsive page. And then once you're super comfortable with that, you can start worrying about templating. So

for the image, I want you to go ahead and do that set up the image. So it is cropping just like we did on the other ones, I would do with a maximum height, so limit the maximum

height or image you can have, and also maybe add a little bit of margin on the bottom of it. And in the next video, I'm going to show you how I would do that I gave you in the

last we'll be wrapping up this module in the next video. But this is the finishing touches on this site. Before we move on to stepping up our style, we're going to look at some

of the more finer tune stuff, but let's not get ahead of ourselves. And let's see what we can do. So the first thing is the challenge that I gave you in the last one of fixing

that image, I'm going to come and put it all the way up with my other image class, my image

style, it could probably come into layout here as well. Because this is all layout related stuff. The reason I'm you know, my other ones were article related, but this one technically

can be used anywhere it could be used inside an article, if I ran into another post where maybe there is no sidebar, and it's just this full image that takes up the full size, then

it would work there too. So I think it makes sense in this case, just to throw it up here where I have my images, which is all the way at the top here. Because it's sort of like

this image helper class. So I called it image full. And all I want to do is give this a

max width and a max width, I said to give it a max height, I'm going to give it 300

pixels. But when I do that, it's trying to keep my image in proportion. So I'm going to give it a width of 100%. So it always stays at 100%. But the problem now is it's all squished

and he looks a little bit chubbier than he should. So we want to come on and give it the object fit property with that cool cover value that we looked at where it will crop

the image for me and it's looking perfection right there. And I think my image flow will

always have a margin underneath it. So our margin bottom, I'll just give it a one m to

give us a little bit of space. And you know, I don't know if that'd be enough. So I'm gonna do two before I even go and look at it. Half love it. I think that looks fantastic. So

that's I'm super happy with that. It's looking really good. The next thing we do is change the color of these and I think I'm done for this the site. I'm really happy with it. As

I said in the next one we'll be looking at stepping up our style. So for that I'm in my typography section now let's go and look I actually those were just h threes. I never

even gave them a class which is cool. So I can just come here and say my h3 has a color

of the bright blue color. Um, no i did group that with here. I could break the color out

of here and apply it but then I You know, I'm being a little bit lazy with my CSS. Now we're defining two different colors. And it's just always the second one that's going to

win. I just don't feel like making another selector for it, to be honest with you. And I don't mind overriding things a little bit, it's easier than breaking this out having

my h1, h2 have all of h3 have all of it, and then another h1, h2 that just does the color,

oh, I'm just gonna put it like this where I'm overriding that a little bit, because it's going to work and my colors are there. Oh, no, oh, my goodness, look, it changed

these ones as well, that's not good. So we can go and fix that just really fast. As I said, we're wrapping up right now. So I'd want to come down to here and those with my

widget, recent post title, so we want to grab that, paste it in there and change the color

of those color to the dark blue. And with that, that is looking awesome, we have the darker titles back to there. I'm gonna remove that because it's driving me absolutely bananas.

And we got one last thing that we need to take a look at. And that one last thing we

## 43. Fixing up some loose ends

need to do it has to do with our navigation, can you think of what it is we'll try and find this mistake here. So our navigation is working all fine. All the different pages

are working. But our current page class got lost. So I want you to go in and add the current

page to the correct page on all the pages to finally wrap up this site. So all we need

to do for that is if you remember on our index, I'm going to do it on this one first, and

then add the class of current page. Let's go take a look, we get the underlying we gets

rid of that hover, but these ones are still working. But when I go to my about me, I want this one now to be that. So if I want to go to my about me, let's actually make our life

easier by copying all of this. And then I can go to my about me find the right spot

right there and paste it in. And then I can go over to my recent posts. And I can do the exact same thing there, where I paste it in. So we're saying what page we're currently

on. And now when we go and look, all the pages, get that nice little underline on it to indicate

what page we're on while we're navigating around. And with that we are done, we have wrapped up this site, everything is looking really really good. I'm really happy with

how it turned out. I really hope you are too. We covered a lot throughout this module. So in the next video, we're going to do a wrap up of all the different things we've covered

and where we go. So this is a really important note before you move on, or before you graduate

from scuba air, you start doing things on your own. And that is we have a viewport meta

tag. And I didn't include it in the lessons of this module. When we take a look in the

browser window on our desktop computer and we play with the width, it works really well. But this doesn't fully simulate how a phone will render a page. So the best case scenario

is actually testing out the sights you're doing on your actual phone, or multiple phones on an iPad on your Android tablet, whatever it is, we want to actually test across different

devices. The reason we have this viewport meta tag we're going to see in a second. But mobile devices, what they started doing is to prevent problems in how the page was rendered.

When sites are not optimized for mobile devices and small screens, they would render a zoomed

out version of the page. And then they'd shrink it down to fit inside the viewport of their mobile device. So they all every device has its sort of default size. So you might be

on a phone like an iPhone that has a width of 360 pixels, but what they're gonna do is they're actually going to bring in the version, they're telling the browser that it's 960.

And it's going to take the version of your site as if it was at 960 pixels wide, and it's going to shrink the whole thing down. And it's gonna be super zoomed out, it's gonna

look like your desktop site, but really, really small. Even if you did all the hard work of building your media queries and doing everything you wanted to make it responsive. So to prevent

it from doing this on our pages, which we've optimized for small screens with the media queries, we need to add a meta tag to the head of our document. There are a lot of different

attributes that we can add, but the basic meta tag that you'll use will look something like this, or it will look exactly like this really. And this can be added anywhere in

the head of our document. This will make sure that the page actually loads properly on on mobile devices, which after all the hard work we did, we really wanted to do. So once you've

graduated from here on screen, but and you're working on your own pages, and you're testing them in different devices. Remember to include this in your head, I can go anywhere up here,

I already have a meta tag for that. I'm gonna go right after that and paste it in right there. So what this is doing is it's saying that it's a meta tag is always referencing

different things. We haven't really looked at meta tags, but it's referencing like different stuff, you have a description of the site, you have the author of the site. So if we

say meta name equals author content is equal to KEVIN POWELL. For this individual page

of my website, I've said that the author of this page was KEVIN POWELL. I know the browser knows that there are other ones that you may use along the way as well. But in this case,

we're looking at the viewport once we're saying that this meta tag is related to our viewport and the first one here's the really important one. What this is doing? Is it saying the

width is actually the device width. So this is where I said that mobile devices would

simulate being a larger device and then shrink everything down. So it would say, No, I'm actually a device that's 960 pixels. And then it wouldn't take that and shrink that whole

thing down. This is saying, nope, the width isn't what you're telling me it is, it's the actual device width and the initial scale, you don't, I don't think really, really need

it. But it's pretty standard to have it in there. And it's just setting the initial zoom level when the page is first loaded. So you could actually have it zoomed in more or less.

## 44. Important Note. The viewport meta tag

But I think that'd be really, really awkward. So you generally want it just to be zoomed to 100%, or like the default 1.0 type of thing. To have it not scaled when the page is loaded.

So once you're done here on screen, but and you're making your own web pages, please please remember always to be including this in here, because the browser window on your computer

is not the same as on your phone or on your tablet. So you want to test across different devices. And most importantly, if you don't want things to look really zoomed out, make

sure every single page that you create has this meta tag right here. Really scratch this,

you did it, you should be super proud of yourself. Before we get into being proud of yourself and giving you a pat on the back. Let's turn remember some of the things we learned in

here. So we covered Flexbox, we looked into a more of a deeper exploration into what Flexbox

is. And we've only really scratched the surface with Flexbox. despite there being a lot more to it, we've learned enough to leverage it and to do some stuff with it. And we've done

that by starting to play with justify content and align items to position things where we

want them to be on the page. So we know we can on both of these do a flex start flex and or a center. And we've also seen that with the justify content, we can do like a

space between or a space around or a space evenly. So we can redistribute the leftover

empty space with our content to help make our layouts without having to complicate things and get all the numbers perfect. We've also seen flex direction, which is switching the

main axis. So it's taking that justify content in the line items and making them work the other way around a little bit, which is kind of weird. I agree. But that's how it works.

But we've seen how we can switch the main axis and why we might want to switch the main axis. So in some circumstances, it was useful to do it just permanently, such as on those

widgets. In the widget, little title cards, it was good to do it on those just so we could work with Flexbox. But we could also switch it using media queries. And when we use our

media queries, we were changing the way the layout worked at small screens and big screens. And this was super powerful and super useful. We sort of went through a whole bunch of stuff

on media queries. But once we started using them, we were just doing the really basic stuff, we're setting some min widths, and it's working. And that's it doesn't have to

be complicated. I don't want you to overcomplicate it when you are using your media queries.

Another thing we learned wasn't necessarily about thinking responsively, like a lot of Flexbox and media queries are all about. But we did learn how to make it responsive was

how to build and style and navigation was a lot more complicated than you might have thought it would be. But we got it to work, it looks fantastic, you should be really happy,

really proud of yourself. And you should give yourself a huge pat on the back for making it this far, it's a lot of work and a lot of material that like an insane amount of

material that we've covered so far. Really, you should be super proud of yourself at this point. Take a break, please, please take a break, there's been tons of stuff, you need

to award yourself with a couple of days off, or a couple of days of doing some other stuff. If you really want try and build the site. Again, going back over everything you have

the Adobe XD link, do it on your own, don't follow my instructions, see if you can pull it off again. Or maybe if your own colors, your own layout, or just try and do your own

thing. Maybe with a similar layouts, you're not trying to go too far out from what we've done so far. But trying to push things and do things on your own a little bit would be

really, really good to use everything we've been learning so far. But do make sure you relax and breathe a little bit. If you're going to do any work at this point, I would

really make sure it's reinforcing what you've learned up until now. And once you're feeling ready, and once you're feeling comfortable, then it's time to step up your style with

the next module in this course. But please don't rush into it. Take your time to get there, relax a little bit, go on Twitter, social media, YouTube, whatever it is chill

out, you know, some Netflix and chill. Whatever it is you'd like to do for a little bit just to let everything absorb. And then when you're ready, and you're feeling up to it. In the

## 45. Module wrap up

next one. When we see how we can step up our style and start making things you have done it you've made it through a massive amount of content. You've wrapped up all of that.

And I hope you've learned a lot of things along the way. Now, as I mentioned at the very beginning, all of these lessons are part of one module of six in the responsive web

design bootcamp that accreted over unscrambler. That course is available right now over there. So if you'd like to go and check that out, by all means you can it goes in from the very

fundamentals of CSS, we're covering the basics. We have this module that you just did we go into stepping up are style where it's not so much about responsiveness, but it's about

understanding CSS, taking things up to the next level, making things look a little bit nicer. We get into typography as well as some of the more flashy and fun things you can

do in CSS. Before taking a real deep dive into Flexbox, to make sure you really understand

it, because in this, we started using Flexbox, we started scratching the surface with it. But in that module, we really go all the way in, we go deep into it, followed by a deep

dive into CSS Grid, which is CSS is newest layout tool. It's the really, it's the way

that large scale layouts should be made these days, it is a fantastic tool. So we take a deep dive into that. And then we wrap the whole project up with a final project where

we take everything we've learned, put it into practice, and also add a few cool things in along the way. So if you did enjoy these lessons, you can go over to Scrivener, and check that

course out. Again, it's called the responsive web design bootcamp. Of course, you can also find me over on my own YouTube channel, which is at KEVIN POWELL, you can find me over there

with a quick search. And whenever you do, if you haven't yet subscribed here on Free Code Camp, I'd also really encourage you to do that. They just have tons of amazing content

here and they keep putting out some great great stuff. So make sure you go ahead and subscribe to their own channel here as well. And as I usually like to sign off on my own

channel, until next time, don't forget to make your corner of the internet just a little bit more awesome.

English

# GitHub Pages Deploy & Domain

hey what's going on guys so in this

video I want to show you how to use

github pages to deploy your applications

to host your front-end applications or

your static websites anything that

doesn't have a back-end so if you're

using like nodejs or Python or PHP then

you'll have to find something else like

like digitalocean or something like that

but if you have just a regular website

or just a complete front-end application

using react angular View github pages is

a nice free solution you also get free

SSL now I'm going to show you how to use

something called gh-pages which is an

NPM module that is just awesome what it

does is it creates it allows you to

create a separate branch called gh-pages

and that's where all the stuff that you

want to host goes and we can do this by

just creating a simple NPM script so to

give you an example if we look at my

Modern Portfolio repository this is the

master branch okay so this has like all

the sass we use sass in this project so

it has all the s CSS files C package dot

J's and stuff like that and then the

dist folder is just the stet the HTML

stuff the compiled CSS compiled

JavaScript the stuff that you want to

deploy now if we go back to the root

here and I go to my branches you'll see

I have a branch called gah - pages and

if we look at that that's where the did

everything in the dist folder is with

the addition of the cname file ok

because I actually attached this to a

real domain name and I'm gonna show you

guys how to do that as well ok so

basically this gh-pages module will

allow us to just simply set it up so we

can do NPM run build and it'll

automatically push to this branch and

it'll automatically deploy our website

okay now I actually have this hook to

the domain of traversée app com and

there it is

okay HTTPS and everything so I'm going

to show you guys how to do that with

just a static site if we have time I

might do it with a react app it's

basically the same thing when you do npm

run build with react it builds out a

folder called build and you would just

use that instead of dist okay same thing

with angular if you do ng build it

builds out I believe a folder called

dist

and you would use that okay and if you

took my react front-to-back course we

did this in that course so let's start

off here with just jumping into vs code

I have an empty folder called my website

I'm not actually gonna build a site or

anything we're just gonna have a simple

HTML page but we can pretend that it's a

real site so what I'll do first is

create a folder inside here and I'll

call it dist okay you could call it

build or site or whatever you want

really it's just what you want to be

deployed so I'm gonna create and here

just a file called index dot HTML and

we'll just add some boilerplate here

let's let's change this title to my

website and then we'll just add an h1

here and we'll say welcome to my site

okay let's actually create one more file

we'll call it a boat HTML and I'll just

grab everything here and paste that in

and we'll just change this to about yeah

just just about and save okay so this is

this this is what I want to deploy to

get hub pages so I'm going to now let's

closes these close these up I'm going to

now install gh pages okay now I know a

lot of you guys know this but to use NPM

which is the node package manager you

need nodejs installed so if you don't

have that just go to no js' org download

and install it very self-explanatory

just like installing anything on your

machine and then if you do NPM - -

version you should see a version okay

now before we actually install gh-pages

we just want to run NPM in it to

initialize what's called the

package.json file and we're just going

to add a - why so enough to answer any

questions

it'll just use the defaults and now if I

go to package Jason it looks like this

okay so now I'm going to just install gh

- pages okay you can do NPM install or

NPM I doesn't matter and we'll install

that that'll get added here as a

dependency and now there's just a couple

things we need to add here one is gonna

be the home page value so our home page

key value so we want and here it's you

you're gonna want to do HTTP and then

whatever your github username is mine is

Brad Travis II and then you want to do

dot github dot

and slash whatever you're going to call

the repository I'm gonna call it my

website okay so that's that and then the

last thing we want to do is just create

a script to be able to deploy to the

gh-pages repo so I'm just going to

replace test with deploy and then all

this is going to do is take gh-pages and

we're gonna add ad flag for the

directory and then whatever the

directory name is where we want to put

you know whatever we want to deploy

which in our case is dist

okay so we'll save that and now we just

want to create a github repo so let's go

back over to github and let's create a

new repository okay and I'm gonna call

it my website I'm going to delete this

after okay and not going to go over the

basics of get I do have a git crash

course so I'll put the link in the

description but basically we just need

to initialize our repository can add

commits and then push to github so let's

go ahead and do that I'm just going to

use my term I integrated terminal but

you can use your regular terminal if

you're on Windows

you can use your command prompt or get

bash whatever it is that you use so

first thing we'll do here is let's

actually create a in the route here

we'll create a dot git ignore file

because we don't want this node modules

folder being pushed to our repository

that's just all the dependencies so we

just want to add in here node underscore

modules and save that and close it okay

so now we can do get an it that just

initializes a git repository basically

creates an hidden folder called dot git

okay now we want to do get add and I'm

gonna add everything so I'm just going

to use a dot here that adds everything

to the staging area okay now we're gonna

do get commit - M and then just any

comment I'll just say initial commit

okay that adds it to the staging area on

our local machine now we need to add our

remote repository so we'll go back to

this page grab this right here this git

remote command and paste that in run

that okay then we want to go back and

grab this last one which is just the

initial push to the master branch and

we'll run that I already have my SSH key

set up so if you don't have SSH key

setup you can use HTTPS instead of

instead of doing this right here you

could change this I'm sorry this right

here to HTTPS and do it like that you

just have to add put in your password

and your username for github all right

so now if I reload this page there it is

everything that we push to the master

branch now if I look at our branches we

only have master we don't have the gh

pages so to deploy to gh pages all we

have to do now is run NPM run deploy

because we added this right here our

deploy script so let's go ahead and run

that just takes a couple seconds ok so

now it says published so now let's go

back to our repository and real

and now if I go to branches you'll see

we have a gh-pages branch if we look at

that it's just the index and a boat

that's what we want to be hosted okay so

we have that branch and then we have our

master branch so now if I go to settings

and we go down to where it says get up

pages you'll see that the gh-pages

branch is already selected for us and we

even have our URL right here which if I

click there it is there's our website

okay so you can do this with with any

front-end application react angular view

as you just need to make sure you choose

your build folder or dist whatever it's

called as your as the folder that you

want to be deployed okay so now let's

work on a domain now I'm using Namecheap

but you can use any domain registrar as

long as you can chain you can add

records and things like that which is

pretty much every domain registrar I'm

going to just bring this over real quick

so this is for a domain called Traverse

e aptech and um this is like the panel

where I can this is the advanced DNS

panel okay and I'll put an I put a link

to Namecheap um there might be an

affiliate link if I have one if not I'll

just put the regular link but what we

want to do here is add for a records to

certain domain certain IPs ok so we're

gonna add one and let me just grab the

IPS real quick okay so this first one

the host is going to be at so it's just

the root and then the IP is going to be

this right here 185 199 108 1 5 3 ok

then I'm gonna say add new record again

we want to do at and then we want to do

the same one same IP except right here

it's going to be 109 ok we want to add

another one and this one here is going

to be 110 and then we want to add

another one last one and this one here

is going to be 111 ok so you just want

to increment

this third one by one alright and I'll

do save all changes okay so it's gonna

save all those a records and let's go

back to our repository which is right

here go down to github pages and add the

domain Traverse e-tec or whatever domain

you're using what was it no Travis II

app dot tech and let's save that all

right now I believe this will work right

away I think the cname which we're gonna

do in a minute takes might take a little

while but this should work let's try it

out so if we go to http I don't can we I

don't know if we can use SSL yet I think

we might have to wait but let's try it

so it's a travesty apt tech Ericka there

it is so we have HTTPS working travesty

optech we have our site showing here now

if I do HTTPS wwf/e CEATEC it doesn't

work okay so for this to work we

actually need to add a cname cname

record okay so you'll see right here

cname right now it's pointed to a Heroku

record so I'm gonna change this we want

to change this to your username your

github username so in my case Fred

Travis e and then dot github dot

that's it no slack no subdirectory or

anything won't work and save that

hopefully it lets us should know

erricka pride charity duck github dot

why didn't ever occur okay so now it

worked I don't know I don't know what

happened there but you just want to add

that cname record make sure it's your

user name obviously and then let's see I

have a you a redirect here that's fine

so now what we want to do is go back to

our repository and we want to go see we

want to go into the gh-pages branch oh

it created it automatically cname oh wow

I didn't know it did that okay so if you

see the cname file here that's fine if

you don't just go ahead and create new

file call it just cname no extension

nothing like that and just put in the

domain like this so now I don't think

it'll work right away

let's see proceed okay so it does why

www dot so the WWV version it just is

showing up as this okay so our site is

now fully deployed with an SSL okay so

that's pretty damn good for free and

it's pretty simple so I like to add that

to my workflow just even if it's a

static website even if it's a four or

five page site and you want to host it

on github pages just put it in a

separate folder that you want for the

gh-pages branch install gh-pages and

deploy simple now real quick I'm just

going to kind of do the same thing with

a react app just to show you that it's

possible okay so let's see I'll just

leave this as is and let's create a

whole new folder for the react app so

I'm just gonna create a new folder I'll

call it my react try to do this quick

and let's open this with vs code

okay and let's see we're going to use

create react apps so I'm going to open

up a terminal and we'd use npx which

allows us to use something like create

react app without installing it on our

system so we'll say create - react - app

and I'm going to do dot because I want

it in this current folder and we'll just

use the boilerplate landing page

whatever you want to call it and well

actually while that's going let's create

our git repository for it so we'll

create another repository and it'll call

it my react okay let's go back that's

still going when you use npx it takes a

little longer than if you install it on

your system alright so let's just make

sure that that it that it works I'm just

going to do NPM start just runs the dev

server just to make sure that it runs

okay good

close that up oops and let's stop the

server with ctrl C and now what we want

to do is install gh-pages so gah - pages

just like we did with the the regular

static site now let's see package Jason

and we're going to do the same type of

thing here we want to put in a home page

value so it's a home page it's gonna be

HTTP Brad Travis seed dots

oops dot github dot and I called it my

react okay now for the deploy script

let's go down to scripts and

deploy now you have a couple options

here we could do G H dash pages dash D

and then react when you build it out

with NPM run build it actually puts it

in a folder called build so you want to

use that as your directory not dist

alright so you're trying to think of how

I want to do this yeah this should be

fine so let's save this and let's if we

look at our git ignore that comes with

create react app node modules is already

there so that's not going to be pushed

to the master branch build which is the

folder that we want to host that's the

static assets that are created that's

not pushed to the master branch but

we're going to use that with gh-pages

so let's first of all just push this to

the master branch so we'll say git and

net okay you'll do git add all will do

gits commit initial commit okay now we

want to just grab real quick the remote

link right here or the remote command

and paste that in okay just like we did

with the other one and then push to the

master branch alright so if I go when I

reload now we have our master branch

with our react app and that's it we only

have the master so now let's do the gh

pages so first thing I'm going to do is

just build out our static assets

so the create react app comes with a

build command so we can just simply do

NPM run build alright so now you can see

that we now have a build folder which

has

our index.html and all static assets

everything like that so now that's the

folder that's actually going to be

deployed to the gh-pages branch because

that's what we put right here so let's

go ahead and do NPM run deploy and

hopefully this works okay let's go back

and let's see let's reload this page and

let's see we have gh pages okay it looks

like every all of our whole static our

whole build folder is in there okay

let's go to settings go down to github

pages and click on this link here and

there it is all right so we have a react

app and again you could just connect a

domain just like we did with this one

okay

same process now one thing you could do

and one thing we do do in my react

front-to-back course when we use this is

we put the build process in the deploy

so you could do like npm run build

double ampersand and then do the github

pages deploy that way you don't have to

do n PM run build by hand you can just

do deploy and it'll build it out for you

okay so you could do that as well

alright guys so hopefully you enjoyed

this little tutorial and hopefully it

helps helps you understand github pages

a little more and gives you a little bit

of insight on adding it to your workflow

in deployment I think it's a fantastic

service and yeah that's it if you like

this video please leave it a like and

I'll see you next time

English (auto-generated)

# Figma UI Design Tutorial: Get Started in Just 24 Minutes!

## Intro

hi in this video i'm going to show you

the best way to start practicing

designing apps and websites in figma

so in this video i'm going to give you

step-by-step instructions you can

literally follow click by click i'll

only tell you the stuff that you need to

get started designing interfaces so

let's get started so we're going to be

## Advantages of using Figma

looking at a tool called figma and it

has a few advantages one most

importantly for you it's free to get

started if you're working by yourself

we like using it at agn smart because it

also has really good collaboration so we

can have multiple people working on the

same design file at the same time it's

also really fast it works on any

computer whether you have a mac or a pc

or linux whatever you have it works

right in the browser and it also has a

mobile companion app so you can preview

your designs on a mobile screen so there

are really no downsides to starting with

a tool like figma as you're watching the

video if you have any questions about

how to do a particular effect in figma

or any comment or something that you

want to recommend please put it in the

comments below and if you want to find

out more tips about ui and ux make sure

to subscribe to our free newsletter the

link to that is in the description below

and it's a great resource for anyone

starting an ui and ux so this is the

## How to log in into figma.com

website you just go to figma.com

and i'm already signed in but you can

sign up very quickly even with your

google account and get started but

before we jump right into figma i want

to show you the way i would recommend to

## Why you should copy other designs

get started so you just want to start

practicing

now for that i'm not going to ask you to

start designing something from scratch

because i believe that would be very

hard with someone especially if you're a

complete beginner in this space

and you have no grounding in design

principles and things like that so the

best way for you to get started is

actually to copy other designs and the

reason this is so good is because

you can see how this design was created

so that when you get stuck on something

you can actually see how this person who

created this file achieved a particular

effect or look inside of figma and this

is totally fine in the beginning because

you're not going to be

selling these you're not going to be

saying that you designed something when

you copied it from someone else this is

just for your own practice and it's a

really good way to get started so as you

## How to start a project from a TEMPLATE

can see here this is what figma looks

like after you log in and start a file

and i haven't even shown you how to

start a file because i want you to use

another file as your starting point as

opposed to a blank file and like i said

we're not going to cover everything that

you can see here on the screen in terms

of what all the various buttons do we're

just going to focus about how you can

get started now to do that

i wanted to start off with a template

and what i literally did was i typed

into google sigma resources and i got a

bunch of results i clicked on a few of

them and the third one called literally

figmaresources.com

is a really good one and i have it open

here so when you look at that you'll

find a bunch of free templates and this

is a great thing about figma there are a

lot of free resources on the web that

you can use to start practicing to

incorporate into your designs so you

don't have to start from scratch

and i picked one of these that is a food

delivery app ui

and i have it open here so the way it

works with finger ma is because it's all

web-based you can literally click this

button here that says copy to figma

and it will copy that file over to your

figma account well first it's going to

open it in a new browser tab where you

can see it and there might be even like

other people viewing this file at the

same time but you can see here at the

top that it says view only and we want

to be able to make edits to this so what

we do in this case when we're looking at

someone else's file is we just click on

the name

and then we say duplicate to your drafts

so if you're signed into figma it's just

going to create a copy of this file in

your own figma account and that's

exactly what i did

and i have it open in this tab right

here so you can see it no longer says

view only like it says in that tab and i

can start editing this all right so now

we have a copy of that file that we're

going to use as our starting point so

in figma as you can see i can just

scroll up and down to see the different

screens

in this file and i can very easily

rearrange them you'll notice that in the

original file

they were placed side by side and i have

more of a vertical arrangement and i can

just do that by grabbing the title and

moving it around with my mouse cursor

like that

i can also zoom in and zoom out with a

pinch gesture on my trackpad that way i

can see all of my screens at once or i

can zoom in to see a particular screen

so let's say now i want to start copying

## Interface OVERVIEW

this design what do i do well just to

give you the basics on the left here i

just have

a bunch of layers and then when i select

an element i get the settings for that

element on the right okay

so what i want to do now is create a

## Create a FRAME

rectangle like this

for my own design

now this has a specific shape and this

is called a frame so the first tool you

need to know about is the frames here or

you hit f on your keyboard and anything

i press will show here on

the bottom left corner so you know

exactly what i pressed

to activate a particular tool and then i

can start drawing a size but now i don't

know how to get the exact same size as

this and so what i'm going to do is i'm

going to delete this

and

i'm going to click on this

screen that exists already

and if i look at the frame here and i

click on that it shows me that this is

using an iphone 11 pro max right so this

is a standard size and what i can do in

figma

is i can hit f again to go into

the frame mode and then i can select

here from a bunch of templates and i'll

select iphone 11 pro max and then it

automatically puts in a frame of the

perfect size i'll just place that right

next to the original frame and i can

start copying the design right

so i can see here that they have a fancy

## SHAPE and COLOR creation

design element here with the purple but

let's keep things simple for now and

let's just assume i just want a purple

background right so the easiest way i

could accomplish this

is

i could either give this whole frame a

background color or i could draw a

rectangle

that's purple right that can be in my

background so if i want to draw a

rectangle i can do that from here you

can see that this is the rectangle tool

and it already says like r next to it so

i'm going to hit r on my keyboard

and draw a rectangle

right so i don't want it to take up i

don't need it to take up the whole

screen this is fine now i can see by

default this is purple

so i'll click on it and see here like i

said whenever i select something the

settings for that element will be on the

right so i'll see here that here is the

gray color and i don't want it to be

gray

i want it to be that purple so i can

either try to change things here or i

can use the eyedropper and this is

something that you might already know

from something like photoshop if you've

ever used a image editing tool or

anything like that so i'll use that and

i'll pick just this color right so this

is good enough for now

next i want to create this card so this

is just another rectangle essentially

right so i'll just draw another

rectangle here and try to have it to be

approximately the same height so i'll

notice here that well the color is

different and i don't have the fancy

rounding so let's start with the color

since we already covered that and this

looks like a very light gray it's not

white because i can see that the circle

is white so what i can do quickly is

again

i'll grab the eyedropper tool select

here

so now i have the same color that's on

the left here right but i can notice

## CORNER RADIUS adjustment

also that the corners here it has these

nice rounded corners that i don't have

and for that right under the

position and the dimensions so if i

change this this is just changing

the x coordinates and i'm just going to

do command z or control z if you're on a

mac to go back

and these are the y coordinates and by

the way

i can change these by clicking and

dragging you you see the cursor turns to

this

sideways arrow and i can click and drag

left and right to change these values or

i can type them in right

i can also change the width

and the height

and here is the angle for rotation but

what i want to focus on right now is the

corner radius right so this is that

rounding effect and so with the corner

radius tool i can adjust how rounded

these corners are now i'm getting closer

to the look of that original file right

but i can see on the bottom that i have

these rounded corners but i don't want

them because the original one doesn't

have it now there is a fix for this in

figma but the easy and cheating solution

would be to just extend this beyond the

frame and then it's hidden outside the

frame

of the phone screen right but if you

want to do it the correct way

you would come to the corner radius

and you would click on independent

corners and what this allows you to do

is set each corner independently and so

if i change this to zero you'll see that

it's no longer rounded and if i want to

keep the top rounded but not the bottom

then i just change the bottom ones right

and now it looks correct

## Create a CIRCLE

now the next thing would be to create

the circle and the way to create a

circle is from the shapes menu you have

an ellipse which you can also do with

the letter

o on your keyboard now by default you'll

be able to draw

kind of a circle with any

aspect ratio but if you want to be a

perfect circle all you need to do is

hold shift on your keyboard as you're

dragging and it'll just be a perfect

circle

now if i want to find out the exact

dimensions of this i can click once and

i can see on the left that i've selected

the whole group i can double click again

and i can still see that i haven't

selected the circle i keep double

clicking and now i have the circle and

it shows me that it's 104 by 104 so i'll

go here

and change the dimensions

of my circle here and i'll type 104

and now i have something with the exact

dimensions now i want to know where to

place this right how high should it be

like because i don't know a lot about

spacing right now and i just want to

copy what the person did

so again i'll double click here

double click again and now i've selected

this and i want to know how far it is

from that edge and the way i can do that

is i hold option on a mac or alt on a

windows computer and that will show me

the distance from all of the neighboring

elements

if i hover over the uh the text here you

can see it's showing me it's 24 pixels

um if i select the card it's showing me

the distance between

all of the edges here right and so what

i want to know is that it's 64 pixels

away from the top and so i select my

circle and then i can start while

holding option i can hit the down arrow

and

now i have 64 right and i can see that

left and right it's perfectly centered

and figma will help you do this if you

move something it will snap to the

center as you can see here now i have my

circle in the middle right let's quickly

change the color now we're experts at

this

uh this is just white so this is very

easy

uh now there's also here an icon

## How to use an ICON

and icons are things that you'll often

need when designing mobile interfaces

because you're you'll wanna have an icon

for a home screen or something like that

and

you can literally google free figma

icons and again you'll have a result

from google that says here are 460 free

icons and this is also a figma file so i

can literally come here this is a figma

file i just hid the interface here by

right clicking you can hide the

interface and i can come here and select

an icon let's say i want to copy this

one i literally just use the copy

shortcut which is command c or control c

i come back to my file

and i paste that element right so now i

have that icon and figma also helps you

with positioning it automatically

centers things as you're moving them

around so if i resize this

without any modifier keys

it might distort the image so i'll do

command z and what i'll do instead is

hold shift like we did with the circle

that way it'll always resize perfectly

and what i can also do is that if i

already have this centered i can hold

both shift and option

or alt on a pc and start resizing and it

will resize from the center out right so

this will just save you a little bit of

time right

so now i have my icon or i can

alternatively just grab this icon double

click double click

and double click again and that now i've

selected this icon i can delete the one

that i already have

and

paste it here

and now i have that same icon so it's

really important for you to have a place

to go to to find icons and this one that

i found here i literally found it in

three seconds so i'm sure there are

other free icons because this might not

have everything that you need

## How to paste IMAGES

and another thing that would be very

handy for you is

somewhere to get free images to use in

your designs because as you can see here

uh those will come in very handy in a

lot of applications

and so let's say that i wanted to get an

image here instead of this blank color

since i can't do this fancy effect yet

and we don't want to focus on fancy

effects just yet right so one good

website is called unsplash and i just

typed delivery in search and i found

this image here and with figma it's very

easy to paste in an image

literally all i have to do is come to

unsplash right click on the image

then i can copy the image

go back to my figma file

i select the background so that it

pastes it over this particular area and

i hit command v or control v if you're

on a pc

and then i can just reposition this

i'm not going to worry too much about

you know perfect positioning or fancy

effects but this is how easy it is to

get an image in right

now let's quickly finish

this screen now the next thing that i

## How to use TEXT

need is text here that says non-contact

deliveries and the way i get text is

through the t icon here i can just hit t

on my keyboard

and

text so

non-contact

deliveries

right now obviously this doesn't look

the same as the other image there are a

few things i want to find out first is

the

size of the text so to do that double

click double click

and then i look here at

the settings so one i can see it's using

sf pro display

so you may or may not have the font

locally on your machine that the person

is using

but chances are you will

and i can see that the size is 34. and

it's bold right so i'm gonna apply the

same setting so i'm gonna click here

type 34

and

i'm gonna use sf pro

text

i'm just typing it but you can also

select it here

and i'm gonna change this to

bold right

now this doesn't look exactly the same

because this is over two lines and

by default sigma will just have the text

be as long

or as wide as the text is long and i can

change that by just

dragging this to resize it and this gets

me the effect that i want i just need to

center it now

and i want to know how

far away it is from that

circle right so we saw this already

i'll just double click double click and

hold option and i can see it's 24 pixels

away

and

i'll just nudge this up

a little bit and now it's perfect and i

think the color

is different here

so

again

i'll hit the color hit the eyedropper

and i think this is dark purple

and now

i have that color right

excellent i think this might be doing

some other effects to the text but we're

not going to worry about it right now

we're going to need to create something

that looks like

the paragraph right here and so i'll hit

the t icon again and by default it's

gonna use the same settings as i used

before just for speed i'll copy this

text here so you don't have to wait for

me to type it out and as you can see it

was very wide and i can resize this

and i can also see here how i can find

out how wide this is by double clicking

on it and then seeing okay the width is

three seven four so i go back here type

three seven four

and that's exactly the same width and

i'll just center it

and i'll see how far away this element

was from the top 25

so

if i nudge it up using my arrow keys

now it's right

and as you can see the screen is coming

## BUTTON UI

along and you can do this as someone

who's never

used figma before or any design tool

now i need a button so for that

uh i'm gonna hit the rectangle key again

on my keyboard and just do this

right and then i'm going to cheat and

look at the dimensions of the this

button so 374 again

and 56

so i'm going to do that

374

and 56

and now

i'm going to center it

and see how far it is

from the element above it so that's 48

but my element is different here so

that's okay you don't need to worry

about that so

what i can do is have the height here

adjust automatically

to the text and i think i have an extra

line that was the reason

so now i can

let me remember what this number was

again

so this is 48

and

48 here would be like this

all right perfect

so now i'm gonna grab the color

and let me see if i can also steal the

corner radius so this shows eight

perfect i'll just type it here

now i need some text to go on top and

let me see what the settings for that

are

so sf protext semi-bold oh sorry

and

15.

so i'll do that real quick

oh it actually copied it over because i

started editing that

and so i say

order now

all caps and i'll just center this

and make it white that's very easy i

don't need to grab that color

and

now i'll just copy this and you can copy

an element real quick either by

selecting it doing command c command v

and it'll paste it right over it so if i

move it here you can see it because it's

white on white

or i can

hold option or alt and that'll also

duplicate the element so this is a

little bit faster let me change the

color of that

to be

the color of the dismiss

so i'll just grab that from here

right and move it under the button

change it to dismiss

the last thing i want to see is how far

away it was so that was 32

and

oh sorry no from the button it's 32

and there

now you've created something that's very

very close to the original design as you

can see starting with a template that's

## Conclusion

already in figma allows you to cheat in

the beginning

and copy over elements and you should

just look for free resources that have

designs of mobile apps of web apps and

start copying these and see how you can

recreate these elements and over time

you'll start to internalize all of the

principles that are in place like how to

space text elements far apart you'll be

able to do it just by looking at it more

or less of course at some point it's

really good if you start learning about

the actual theory of design but this is

a way to get started without being

bogged down so maybe you can switch

you'll do a little bit of practice a

little bit of theory and start applying

what you learned but this is one way to

get started without knowing any theory

and just by copying existing designs now

## Next steps

the next step from this would be to

start designing things in figma when

you're comfortable with it and you can

pretty much recreate everything that you

see so you might see an app that you

like that's not already created in figma

so maybe you look at instagram or any

other app that you like or tick-tock or

anything and you can start designing

that in figma and when you get stuck you

google it you find out how this fancy

effect was created or you can google can

i do you know x or y in figma can i play

a video in figma for example if i want

to do something like tik tok and you'll

start to familiarize more and more with

the tool and the step after that is to

look at an app that you think could be

improved so maybe your banking app is

really really bad and you've always

wanted to improve it well now you know

how to use figma enough to start putting

together a new design so i hope this

video was helpful for you please let us

know in the comments

if you have any questions about next

steps something you got stuck on and if

you want more tips and tricks about ui

and ux we have a free newsletter that

you can subscribe to the link is in the

description below and it's very helpful

for someone who's just starting out like

you because we go out and we read all of

the news

about ui and ux and we just send you the

best stuff so if you're just starting

out make sure to subscribe to the

newsletter and i'll see you in the next

video

[Music]

you

English (auto-generated)

# React Native Tutorial for Beginners - Build a React Native App

## Introduction

welcome to my ultimate react native course Amash I don't wanna be your instructor in this course you're gonna learn how to build fast and beautiful

mobile apps using react native I'm assuming in this course you know nothing about react native and want to learn everything from scratch I will explain

every line of code awright so you learn and understand all the underlying principles now you're not gonna work on a dummy to-do app or a calculator you're

gonna work on a real full-fledged app called done with it this app is a marketplace for selling this stuff you don't need any more anything you are

done with here's the front screen of our app we can login or register let's log

in with my account here you can see all the items that people are selling for

example you have this guy over here he's selling his red jacket for $100 of course this is all dummy data that I have generated now we can tap an item to

see more details about it you can see who the seller is we can see where the item is located on the map we can also send a message to the seller and this

will send a push notification to the target mobile device now we can pull this down to close it we can also pull down the list to refresh it just like

the apps you use on a daily basis we can filter the list and look at the cameras that are for sale we can add a new listing first we select an image we can

add a second image up to three images now let's give it a title my first item

we give it a price $99 now let's assign it to a category so

let's put it in the category of furniture now when it posts us to the server you're going to see a progress bar and a beautiful animation at the end

take a look there you go and done here are the listings I have

posted you also have my account page where we can access my listings and the

messages I have received now we can swipe an item to the left to delete it just like the apps you use on a daily basis so if you follow along by the end

of this course you're gonna master react native and you'll be able to build mobile apps with confidence so are you excited to build this app with me then

jumping and get started

## Prerequisites

to take this course you don't need any familiarity with react native or mobile development in general but you need to know JavaScript not react because react

native is built on top of react but instead of targeting the browser it targets mobile platforms you don't need to be a react expert but you need to

know all the essential concepts such as components JSX props state and so on if you wanna learn react I have a two hour tutorial on my

youtube channel as well as a complete 13 hour course that teaches you everything from the basics to more advanced concepts I'll put the links down below

if you're interested so what is react native and is it the right tool for you

## What is React Native?

or not well react native is a framework for building native apps for iOS on

Android using JavaScript so if you know JavaScript you can use your JavaScript skills to build real native apps for iOS and Android these apps are truly native

so they are not web apps that look like a mobile app so with react native you don't need to know iOS or Android programming unless you want to build a

really complex app and you need to talk directly with the native API of this platform for the most part you don't need to do that so we can write pretty

much all of your application code in JavaScript and share it across iOS and Android that's why a lot of companies these days prefer to build their apps

using react native because they don't need to hire two separate teams of developers maintaining two different code bases one for iOS written in Swift

or objective-c and the other for Android and Java or Cartland now one of the

common misconceptions about react native is that you cannot use it to build any serious apps for example we have this guy over here his name is John Smith is

a developer with a lot of strong opinions he believes that the only way to build anything serious is by using the native languages on tools well

that's not true here are five apps are built with react native Facebook Instagram Pinterest Skype uber AIDS and many many more so if

you have an idea that you want to turn into a real app using your JavaScript skills react native is the best tool to have your toolbox

## Expo

there are two ways to build react native apps we can use plain react native or Expo Expo is a set of tools and a framework that sits on top of react

native and hides a lot of complexity from us it makes it incredibly fast and easy to build react native apps if you have never done mobile development

before Expo is the way to go you can build and run your first app within a few minutes the other option is to use react native CLI or command-line

interface when we create a project with react native CLI our project will look like this so we have these folders Android and iOS

these are native Android and iOS projects we also have our JavaScript code on a site that can be shared across these two platforms so this approach is

suitable for people who have some experience with iOS or Android programming in this course I'm assuming you don't have any prior experience in

mobile development so we're gonna use Expo let me create a project with Expo we're not gonna have these Android and iOS sub projects we only have JavaScript

code so that means we cannot work directly with the native API of these platforms we are limited to what expo gives us in terms of the native features

now honestly this is not a problem for a lot of apps because Expo gives us a lot of native features so we can build a real complete app using just expo and

that's what i'm gonna show you in this course but if you do have some experience with mobile development and you want to have some extra flexibility

if you want to customize or tweak some native components you can always eject from expo and get access to the underlying ios and android projects i

will show you how to do that later in this course so next i'm gonna show you how to set up your development environment alright the first thing i

## Setting up the development environment

want you to do is to make sure that you're running node version 12 or higher so here in the terminal window let's run node dash V I'm running node

version 12 point fourteen point one make sure you are running node version 2 or higher now let's install expose CLI globally so NPM if you're on a Mac and

you haven't configured permissions properly you have to prefix this with sudo so npm install - g expo - CLI

is going to take a while so I'm going to pause the recording

all right Expo CLI is installed you might get some warnings don't worry about them they don't really matter so with Expo CLI we can easily create

and run a react native project now you should also install expert client on

your phone this is an app that you should download from the App Store it's available for both iOS and Android with this we can easily run our app on a

physical device now as my code editor I'm gonna use Visual

Studio code or PS code you can use any code editor that you prefer but it's easier if you download vs code and code along with me because throughout the

course I'm gonna show you a lot of tips and tricks a lot of shortcuts to write code fast you're gonna love these tips so you can download vs code from code

that Visual Studio comm like hearing vs code we're gonna use a bunch of extensions these extensions are optional but they make our job easier let me show

you so over here let's search for react native the first extension we're going

to use is react native tools this is built by Microsoft and with this we can debug our react native applications inside vs code it's very popular the

second extension we're going to use is react native slash react slash Redux snippets this gives us a bunch of code snippets so we can type a few shortcuts

and generate code really fast I love this extension I also use prettier for formatting my code so let's search for prettier there you go

prettier code formatter made by spen peterson it's a very popular extension

you probably have it yourself I also use material icon theme

let's take a look with this extension installed we get pretty icons in our

project so the files in our project are gonna get icons like these depending on their type now let's apply a setting so the moment we save our changes our code

gets reformatted so on the top we go to preferences and then settings search for

format on save so make sure to enable this option so anytime you press ctrl +

S or command + S vs code will use prettier to format your code now that we

## Your First React Native App

have set up our development environment let's create our first Expo project so here in the terminal window let's type expo and it done with it that's the name

of the app we're going to build you can call it anything you want so let's go with this now we have to choose a workflow for building this app we can

use one of the managed workflows or one of the bare workflows if you use a manage workflow export is going to take care of all the complexity behind the

scene so with a managed workflow we're not gonna see those iOS on Android projects we only have a pure JavaScript project if you use a bare workflow we're

gonna have a bare bone react native project so we'll have those iOS on Android projects in this course I'm gonna go with this blank template we

also have a template with typescript but I'm not gonna use that in this course because I want to bring extra complexity so let's use the blank manage workflow

template now this gonna take a while so I'm gonna pause the recording

all right our project is ready so let's go into this folder and then open it in

vs code run this app let me give you a quick overview of what we have in this project

so we have this assets folder this is where we put all the images audio files

videos and so on any kind of asset that you want to bundle with our app now we also have after J s this is a basic react native component

so on the top we are importing react we're also importing a couple of components from react native so this view that we have here is like a div in

the web world and text is used to display text on the screen so in react native we don't have HTML elements like div paragraph anchor and

so on we have to use the building blocks or the components that are provided by a react native here is an example this app is a function component so by default

react native uses function components because there are simpler they're more lightweight you can use class components but it's better to use function

components so here we're returning a JSX expression we have a view which is like

a div this view has some style we'll look at that later and inside this view we have this text component for displaying this text on the screen now

what is this style here well we're referencing this object Styles the container we're creating the Stars object over here using the

stylesheet object so we call the create method and passes an object this object has a property called container that contains all the styles for our

container now if you look at the name of your Stars they look familiar for example we have background color this is like a CSS attribute but this is not CSS

this is just a plain JavaScript property when we compile our app react native

it's gonna translate these properties and the components were using to their native widgets for example this view over here if we build this app for iOS

this view is gonna be mapped to UI view if we build it for Android is gonna be

mapped to Android view so with these components we can represent our UI in an abstract or platform-independent way when we compare our app react native is

gonna map these components into their native widgets so that's why the apps that we build with react native are real native apps now we should open a

terminal window to start Expo server to serve our app so on the

top we go to the view menu look at the shortcut for the terminal window on Mac is control and backtick so let's open up the terminal window here we type NPM

start now this opens our browser pointing to

this address the port number might be different on your machine depending on what you are running so this is what we call Metro bundler it's the JavaScript

bundler for react native so it's responsible for compiling all of our JavaScript files into a single file now here on the Left we have a few commands

we can run our app on an Android device or emulator we can run it on an iOS simulator we can run it in a web browser we can send a

link to our app with email so other people can try it we can also publish our app to expo so anyone in the world can view our app using expo client this

is much faster and easier than going through app stores if you have done any kind of mobile development before you know that going through app stores is

very tedious there are so many steps you have to follow with expo we don't have to worry about this we can simply publish our app to Expo and anyone in

the world can easily view our app of course this is purely for development and testing not for production but talk about that later in this section so this

is Metro bundler now if you go back to the terminal window in vs code you can see these commands and their shortcut these are the same commands that you saw

in Metro bundler for example we can press a to run our app on an android

emulator or I to run it on an iOS simulator and this is where the logs for

our app will appear so if anything goes wrong this is the first place we want to look at all the errors and log messages will appear here so we want to have this

terminal window open at all times all right now that metro bundler is running next I'm going to show you how to run our app on an iOS simulator

## Running on an iOS simulator

in this lesson I'm gonna show you how to run our app on an iOS simulator to do

this you'll need a Mac if you don't have a Mac don't worry you can still run the app on your phone whether it's an iPhone or an Android phone you can also run the

app on an Android virtual device which I'm going to show you in the next lesson now I want you to install Xcode you can get it from the App Store so open our

App Store and search for Xcode here you can get the latest version Xcode is a large app so installing it is gonna take a while

perhaps 20 to 30 minutes depending on your internet connection once you install Xcode run it and then go to Xcode preferences

on this window go to the locations panel and make sure you have installed the

latest command-line tools now we can start analyst simulator so let's close

this window we go to Xcode open developer tool simulator

so here's our iPhone simulator we can move it around we can resize it we can

change the type of this device by going to file open device under iOS you can

see various iOS devices for example we can start an iPhone 8 it's gonna start a

second simulator now unfortunately at the time of recording this video expert gets confused when you have multiple iOS simulators running so I'm gonna close

this iPhone 8 I'm going to use this iPhone 11 now we go to Metro bundler and via app and the iOS simulator

we can also write it in vs code by pressing I in the terminal window so

this is gonna start our project in iOS simulator the first time you're gonna see this message popping up open and expo so let's open it

so this is the expert client that you install on your phone let's close the welcome message and here's the output of our app we have this text in the center

of the screen now let me show you something really cool I want to put this on the side and open our project on the left side now let's change the text here

to hello world now the moment I save the changes

our app is gonna refresh take a look save and here's the hello world message this is one of the powerful features of react native called fast refresh so we

can see our changes as soon as we save we don't have to wait for native bills to finish now in this IRS simulator we can bring up the developer menu by

pressing command + D now unfortunately this does not work on my mission anymore I think this happened after I upgraded my Xcode so I figured out to make this

work I have to press control D first and then command D so this is the developer

menu I was talking about this is part of expert client so here we can reload our app so in case something goes wrong and fast refresh doesn't work we can

manually trigger a reload we can also copy the link to clipboard we can go to home so here we can see other expo projects we are working on at the moment

we have a single project and it's here so let's get back to it so you can bring up the developer menu by pressing ctrl + D and then command on D like this so

this is all about the iOS simulator next I'm going to show you how to run this app on an Android virtual device

## Running on an Android emulator

in this lesson I'm gonna show you how to set up an Android virtual device to run

your apps now while you can always run your react native apps on your phone during development it's a lot easier to run it on a simulator or a virtual

device because you don't have to constantly touch your phone so to set up an Android virtual device first we need to install Android studio you can get it

from developer.android.com slash studio so go ahead and download the latest version of Android studio the first time you run Android studio you're going to

see a setup wizard like this so click on next go with the standard installation

because this is going to install all the necessary components you need to create an Android virtual device so click on next here you can select a light or a

dark theme we don't really care about this because we're not gonna develop in Android studio so let's click on next now look at the components that are

gonna get downloaded we have Android emulator Android SDK build tools Android

SDK platform Android SDK platform tools Intel x86 emulator accelerator as well

as a bunch of other components these are the components you need to create an Android virtual device if these components are not listed here don't

worry I'm gonna show you how to install them later but if you go with this standard installation you shouldn't have any problems so let's click on finish

this is going to take a while so I'm going to pause the recording

right Android studio is ready now let's go to the configure menu and select SDK

manager if you follow the standard installation here you should have all those necessary components so on the sdk platforms tab you should have the latest

stable version of android at the time of recording this video that is under ten or queue under sdk tools you should have android sdk build tools as well as

android emulator Android SDK platform tools and Intel emulator accelerator if

these components are not selected select them click OK and then Android studio is going to install them for you now if you're in a Mac or Linux there is an

extra step unit to follow if you're a Windows user you don't have to follow this step but please watch the rest of this video because there are more steps

you need to follow so head over to Doc's that Expo at i/o now on the left under

manage workflow you can find all the official instructions to install an Android virtual device so we have almost

finished the first step so we installed Android studio now down the bottom of the first step you can find the extra step for Mac and

Linux users so you need to add 100 SDK to your path so you should edit your

bash profile or bash RC and add this export statement so let's copy this

export statement now open a terminal window and use your

favorite editor to edit your bash profile so I use Visual Studio code now

we go to our home directory and open - under line profile

so here's my bash profile at the end of bash profile I'm gonna paste what I copied from the official documentation of Expo

so we're exporting Android SDK and here's the path to Android SDK but you

need to replace this path with the path of Android SDK on your machine how do we get it very easy back to Android studio let's go to configure SDK manager here

we can find the path to Android SDK so copy there's the back to your favorite

editor let's replace the path with white be copied now here in Expo documentation

if you're on Mac there is one extra step you need to follow you need to add platform tools to a path as well so copy the second export statement and paste it

at the end of your bash profile now once again we need to replace the path to Android SDK so let's delete this and use what we have over here so here's the

complete path library Android SDK platform tools okay save the changes now

on my machine I'm using zs8 or z shell this is the fancy terminal window I have here for this I need to edit a second file that is in the home directory that

is z SH RC now back to bash profile I'm gonna copy these two expert statements

and paste them into Z shrc so copy and pace over here save the changes now

back in the terminal window at this point we should be able to run a DB if you get an error saying command not found that means you didn't follow one

of these steps properly all right we're done with instructions for Mac and Linux users so everyone should follow the rest of this video to set up an Android

virtual device so here on Android studio let's go to configure a video manager if

it is short for Android virtual device on the screen let's click create virtual

device here we should select the hardware we want to emulate we have so many options we have various pixel devices we have Nexus and so on if you

don't know where to start usually the latest pixel device is a good place to start I prefer to use the one with Play Store installed so I'm

gonna go with pixel three a you can see the size resolution and density of this

device so let's click on next now here we should select the image or the

operating system we want to install on this device you can select any of the images here in the recommended tab I prefer to use the latest stable version

that is often the second item here so let's go with this now Android studio is

down within this image to install on our emulated hardware it's gonna take a while so I'm going to pause the recording

all right we're done with the step so let's click on next like here we can give this device a name I'm gonna use the default that specifies

the type of hardware and the image we have installed on it now let's click on finish all right our virtual devices ready we

can run it by clicking on this play button here so here's our virtual device

or emulator we can simply drag the sides and resize it

so when I make it smaller and put it on the side now to run our app in our new

Android virtual device we go back to metro bundler and on the Left select run on Android device or emulator

we can also go to our terminal window and press a to run this app in our

Android emulator but make sure that the emulator is up and running before you press a now over here you can see that expert client was installed on the

device so now metro bundler is building our JavaScript bundler so here is expert client and as you see it's downloading

our JavaScript bundle and here's our react native app now here we have hello world because in the

previous lesson I changed this message if you skip the previous lesson let me show you something really cool so I put vehicle on the left side and

now let's change this message to hello react native now the moment I save this

change our app is gonna get refreshed immediately take a look so save and here's hello react native this is the beauty of react native so we see our

changes the moment we save them we don't have to wait for native bills to finish so this makes development incredibly fast and easy now on this virtual device

we can bring up the developer menu by pressing command + M on Mac or ctrl + M

on Windows so here's the developer menu we can manually reload the app in case

the automatic reload didn't work we can copy the link to clipboard we can go to home here you can see the list of projects we

are working on currently we have only a single project that is done with it which is hosted and this address Expo here's the IP address of my machine and

here's the port number now we can select it and go back to our app once again can

bring up the developer menu by pressing command on M on Mac or ctrl LM on Windows here we have a few other items these are

you sport debugging we'll talk about them later in this section next I'm going to show you how to run this app on a real physical device as I told you

## Running on a physical device

before during Duvall map it's a lot easier to test your app and a virtual device or a simulator because you don't have to constantly touch your phone

however simulators are well simulators they don't always function like a real

device also certain features are not available in these simulators so you want to test your app on a real physical device whether it's an iPhone or an

Android phone so the first step is to install expert on your phone you can get this from the App Store once you install this to run your app on

your phone all you have to do is to go to Metro bundler and look at this QR code so point your phone at this QR code and open the camera app your phone is

gonna pick this up and run your app in Expo as simple as that just make sure your phone is connected to the same wireless network as your computer

otherwise this is not gonna work now when you're running your app on a real physical device you can bring up the developer menu by simply shaking your

device so when you shake your device you're gonna see something like this this is what we call the developer menu

## Logging

over the next few videos I'm going to show you a few tools and techniques to debug your react native applications we have a few different tools here you can

choose the tool or the tools that you prefer and it really depends on what you want to do so the simplest way to debug our react native applications is using

the good old console deadlock statements so here in app component

we can do a constant blog and say app executors now save the changes bring up

the embedded terminal we have two entries for app executed because I have two virtual devices connected to Metro bundler I have an iOS simulator and an

Android virtual device and here in Metro bundler on the top left you can see the

connected devices so I've connected an iPhone 11 Promax and an Android virtual device now we can click on any of these devices and see

the log for that device so console that log is the simplest way to debug or react native apps however this concept the log statements can have a negative

impact on the performance of our apps in production so use these only during development or testing once you're done with your debugging sessions make sure

to remove them before building your application for production next I'm going to show you how to debug your apps in Chrome let's see how we can debug

## Debugging with Chrome

this app in Chrome so I'm going to create a bug in this app component let's declare a variable called X and without initializing it let's call X to 2 string

let's save the changes we get this red box where we can find information about the error that just occur so here's the error message

undefined is not an object evaluating X to 2 string and here you can see we're

in our component tree this error accurate in this case it occurred in app component now to debug this app first we need to dismiss the screen so we press

the escape button it's gone now we need to bring up the developer menu I explain how to do this in the previous lessons if you're using a real device you have

to shake it if you're using an Android virtual device you have to press command + M on Mac or ctrl + M on Windows and if you're using an iOS simulator you have

to press command + D on my machine that doesn't work so I have to press ctrl + D

and then command + D now we need to enable remote debugging when we do this the JavaScript code for this app will end up executing in Chrome

let me show you so we tap on debug remote yes opens up a new tab in chrome pointing to his address localhost / debugger - UI so

now the JavaScript for this app will execute remotely in Chrome and that's gonna slow down our app so when you're done with your debugging session make

sure to stop remote debugging now let's open up chrome developer tools

here on the console tab you can see the message that we logged as well as the error that occurred cannot read property two string of undefined

now to debug this app we go to the sources tab over here click on this icon pause on exceptions and then select pause

uncaught exceptions let me enable this option if an exception is caught in our app chrome is gonna stop the execution and highlight the line where the

exception occurred let me show you so back to our app let's bring up the developer menu and reload our app now we can see the line where the

exception occurred X the two string so Chrome paused on this exception now let

me show you a few tools that we have here for debugging we can click on any line to insert a breakpoint and when you reload our app Chrome is gonna stop

execution right on that line so from that point we can execute our code line by line we can watch the value of various variables to see if our

computations are correct or not so let's reload our app one more time so

bring up the developer menu we also have a shortcut for it on Mac we can press command + R on an iOS simulator so reload

now line five is highlighted and it's about to get executed over here we have

various tools for executing our code so we can step over this line or we can step into it this is useful if in this line we are calling a function that we

have written so we can step into that function and execute the code in that function line by line we also have step out so if you step into a function and

we're done debugging that function we can step out of that function now in this case console the blog is not a function we have written so it doesn't

matter if we step over it or step into it so I'm gonna step over it and by the

way look at this shortcut here in the tooltip on Windows it's f10 on Mac its command and a single quote so when debugging always use this shortcut

because it's much faster than constantly clicking on these icons so step over

this now we are on this line and you can see that X is undefined that is the

reason why we got this exception now here we also have this watch window where we can watch the value of various variables so click on this plus sign and

type X and then enter so as you can see X is undefined so this watch window is

really useful to see what's going on now we're done with our debugging we know why we got this exception so we should remove this breakpoint otherwise next

time we reload our app chrome is going to pause execution on this line so let's go back to our code delete these two lines save the changes

then reload our app okay there is gone so we should stop remote debugging

otherwise our app is gonna be really slow so once again bring up the developer menu and stop remote debugging so this is how we can debug your apps in

Chrome now here we also have the network tab this is useful when we have API

calls to our back-end you haven't gotten there yet but we're gonna get back to it in the future now what about the LMS tab well the elements that you see here are

not the elements were the components of your app these are the elements of the debugger window so unlike a web app we cannot select an element and look at its

properties so this is all about debugging in chrome we can also debug

## Debugging in VSCode

our react native apps in vs code let me show you first we bring up the developer menu and enable remote debugging so this opens this window in chrome we have to

close it otherwise we'll get an error but in this case in this demo I'm not going to close this window because I want to show you that error now back to

vs code let's go to the extensions panel search for react native and make sure

you have installed this extension react native tools with this extension we can debug react native apps in vs code now let's go to the debug panel the first

thing we have to create this file launch JSON this is where we store our debug configurations currently we don't have this file in our project so vs code is

suggesting to create it so click on this link now from this drop-down list select

react native now sometimes we ask o doesn't show you this drop down list I'm not sure why but it immediately creates the launch duck JSON file if this

happens to you don't worry just continue watching you'll figure out what to do in a second so we select react native now here we see various configurations for

debugging react native apps these configurations are installed by that extension react native tools so if you don't install that extension you're not

going to see these configurations now by default debug Android is selected I'm

gonna deselect this and select attach to packager honestly this is the only

configuration I'm familiar with so we select attach to packager and click ok

and here's launch dot JSON so sometimes we ask Oh doesn't show you that drop-down list instead it immediately creates this file

now here we have various debug configurations currently in our configurations array we have a single object a single configuration and the

name of this configuration is attached to packager if you don't have this object here you can always add a configuration you can either click on

this button add configuration or you can go to the run menu and select add

configuration now in this context menu we search for react native and this

brings up various configurations for debug and react native apps so here we have attached to a packager we also have debug Android and so on so for example

let's select a second configuration like debug Android so now we have a second

object a second configuration called debug Android a same the changes we're

done with this file I think is you're curious this file is located inside this

folder dot vs code now let's go to app that yes

I'm going to declare a variable X and set it to one we're going to use this

during our debugging session now we can insert a breakpoint on this line and start an app in debug mode and execute it line by line exactly the way we debug

our app in chrome now let's go to the debug panel in this drop-down list we can see various debug configurations currently

we have two configurations debug Android and attach the package er I'm gonna select attach the package ER and click on this play button our debugger is

started and then it immediately feels stuck to see why this happened you go to the View menu and then look at debug console

so here's the arrow - packager are you sure there's a packager and it is running in this port 8081

so by default PS code tries to connect to this packager react native packager

to get the JavaScript code on this port 8081 we have to change the sport to this

port over here nineteen thousand and one so back to vs code on the top we go to

the code menu then preferences settings now under user tab here research for

react - native that packager dot port so look here is support 80 81 we have to

change this to nineteen thousand and one now we're done with this let's bring up

the debug panel one more time and start our app in debug mode

debugger failed one more time so let's go to view debug console

here's the second error could not debug another debugger is already connected to packager this happened because we have this window

open in the background so sometimes you have a million tabs or windows open you don't know that you have this window open somewhere that's why nothing works

on your machine so make sure to find this window and close it because you can either debug in Chrome or vyas code at a single point in time not both these

processes so now we're going to debug in vs code which means our JavaScript code

is going to get executed inside vs code so let's bring up the debug panel one

more time and start our app all right we establish a connection beautiful so

here's our simulator bring up the developer menu and reload

app so the debugger loaded now let me close

the debug console take a look we are in apt at Jas and the first line where I

inserted a breakpoint is highlighted so now we can execute our code line by line we have the same tools that we had in Chrome we can step over the current line

we can step into it if you're calling a function that we have written if you're inside a function we can step out of it we can restart our debugger and we can

disconnect from our debugging session so let's step over this line and by the way

this shortcut is f10 now here on the left side we can see the value of

various variables so vs code automatically detected the variables in scope so here we have X the value of X is 1 we also have our watch window so we

can selectively watch various variables in case they are not detected over here so we can type X and we can see the value of X so now we can execute our

code line by line and see what's going on when we are done we should always stop the debugging session because our JavaScript code is executed remotely

inside vs code so we disconnect here then we go to our app bring up the

developer menu and stop remote debugging now we get this error run time is not

ready for debugging don't worry about this we just need to reload our app and here you can find this shortcut for reloading on Mac its command + R so

now our app is running just like before and the debugging session is terminated

## Publishing

so far we have been serving our app from this address local host port nineteen

thousand and two and that means if we turn off our computer or if you stop expose CLI we won't be able to open our app with expert client this is where

publishing comes to the rescue so we can publish our app to Expo and then our app is gonna have a public and a permanent address that anyone can use

to open it with expert client just like how we can publish NPM packages to NPM

directory we can publish our hab to Expo and it will be visible to anyone in the world this is much easier than going through app stores if you have done any

kind of app development before you probably know that going through app stores is very tedious there are certain steps that you have to follow with

export we don't have to worry about it we simply publish our app to Expo and anyone in the world can open our app and test it with expert client of course

this is purely for development and testing not production when you want to put our app to production we have to go through app stores that's the topic for

the future so this video I'm going to show you how to publish your app to Expo now we can publish using Metro bundler here so we can click on publish or

republish project alternatively we can open a new terminal

window and type Expo publish either way you have to get back to this terminal we

know because Expo is gonna ask you a few questions let me show you so in this demo I'm not gonna run Expo publish here instead I'm gonna use the publish come

out here so click the Expo is asking the name of our app this is loaded from

after JSON so if you look at after JSON you can see various settings about our

app here's the name of our app here's a URL slug for it this will be part of our

apps URL on Expo you will see it in a second so back to Metro bundler we can

optionally specify the github source URL as well as a description we can optimize

our assets so they will be compressed and minified by default this is enabled now the URL of our app is gonna look like this expert at IO at sign after

this we're gonna have our username which we're gonna create in this video and then we have done with it so this is our URLs luck now let's click on publish

project so we get back to our terminal window in vs code this is where you're

running X + CLI so export is asking how would you like to authenticate we can make a new Expo account or login with an existing account so let's create a new

export count we have to enter our email so I'm gonna go with programming with Maj at gmail.com for my username I'm gonna type

programming with Marsh let's give me the password and confirm it

all right now Expo is building our iOS and JavaScript bundles

all right back to Metro Bunner our app is successfully published at its address let's click on it so here's our app currently our app doesn't have an icon

that is why we have this empty box we'll come back and fix this in the future now here we have this QR code so anyone can scan this code with their phone and open

our app with export client as simple as that we don't have to go through app stores now our app currently doesn't have any descriptions so let's go back

to Metro bundler click on and publish one more time and this time give our app

a description I'm gonna say a marketplace for selling this stuff you

don't need any more now let's publish it one more time

now back in vs coke let's open up the JSON down the bottom you can see our new

description so our app is republished let's open it one more time and here's

the updated description beautiful hey guys ma chère I just want to let you know that this tutorial is the first two

hours of my ultimate react native course the complete course is divided into two parts basics and advanced topics but each part being about five to six hours

long so the complete course is over ten hours long it includes all the source code we write in this course every section has it before and after source

code so you can easily code along with me plus you will get plenty of exercises and step-by-step solutions and a certificate of completion that you can

add to your resume if you're interested I put the link down below I'm offering a discount to the first hundred students so if you're interested enroll now

before it's too late now let's continue to the next lesson in

## Fundamental Concepts

this section we're going to look at some of the core components and api's in react native we'll be talking about view text image button touchable alert and so

on there are more components in react native than we can cover in this section or in this course let's route the course you're going to learn the ones that you

will use most of the time now if you're curious you can find the complete list of these core components and api's on react native website so head over to

react native dot dev then go to API here on the Left we can see all the core

components that we can use for building user interfaces these components are cross-platform so when we compile our app they will be mapped to their

negative equivalent so if we use a button here on Android this will look

like a standard Android button and on iOS it will look like a standard iOS button we can also customize the look and feel of these buttons using Styles

you learn in the next section so here are all the cross-platform components you'll also have a bunch of components specific to Android and iOS and we also

have a bunch of api's these api's give us access to native functions there are not UI widgets for example we can use keyboard to control

the soft keyboard or we can use stylesheet to create a bunch of styles

again these API is our cross platform we also have a bunch of AP is specific to

Android and iOS again will study most of the essential components at API but we

don't have time to look at every single component so once you learn the foundations you can learn the other components on your own so next we're

going to talk about views so earlier I told you that in react native we don't

## View

have HTML elements like div span paragraph and so on so we have to build our UI using the built-in components in react native view is the most basic and

fundamental component for building UIs it's like a div so it's a container component that we can use for grouping or laying out children so this way we

have over here has a bunch of styles now we'll talk about styles in detail in the

next section but for now let me briefly explain what these styles are so we have

flex set to 1 this means that this view is flexible and it will grow both horizontally and vertically to fill the free space in this case it grows and

takes the entire screen now the background color of this view is white let's change this to Dodger blue so here we can use named colors we can also use

RGB colors exactly the same way with specific colors in web applications now

save the changes look this view is filling the entire space now one thing I want you to pay attention to is this notch or the edge

on the new iPhones sometimes you want to make sure that this notch doesn't cover your content for example back to our Styles let's remove these two properties

align items and justify content with these we can put a component in the

center of this view that is the reason why our text appears in the middle of the screen now we'll talk about layouts and aligning

components in the next section for now let's just remove these two properties save the changes now look our text is over here it's too small let me make the

simulator bigger look so part of the text is covered by the knotch this is

where we can use a special type of view called safe area of view so on the top we import safe area view from react native now we can replace this view with

safe area of view so with this view selected press command + D on Mac or ctrl + D on Windows now both views are selected we have two

carats this is called multi cursor editing so we can replace both of these at the same time so type safe area view now press escape to cancel Multi cursor

editing okay save the changes take a look our text is no longer behind this notch so safe area view adds a bit of padding

on the top to make sure that our content is within the safe area alright so this

is all about views for now next we're going to talk about the text component

## Text

the second most fundamental component in react native is text and we use it for displaying text so here we cannot place text just anywhere within JSX as we do

in web applications we should always wrap our text with the text component now this text component has a few interesting props let's look at the

documentation for this component so here on react native web site under the API

section let's look at the documentation for the text component so here we have a

bunch of props I'm going to talk about the most important ones the first one is number of lines when we set this if our text is longer it will

get truncated let me show you so back in vs code let's change this to a really

really long text now I want to make this even longer and see what happens so

now our text is wrapped we have two lines but if you set the number of lines to one our text will get truncated take a look so we set the number of lines to

one now say take a look so we have dot dot dot at the end of the first line

very useful we can also make this text act like a link using the unpress prop

or event so here we set on press we set this to a function we can write a

function inline here like this we can pass an error function the constant of

log saying text click but this is useful for very small

functions one-liners if you have a fair amount of logic we don't throw all that logic here in the middle of our JSX so we should implement it in a separate

function so before our return statement we can define a function by convention

we prefix the function name with handle so we want to handle the unprecedent we

call this function handle press now we set it to an error function and this is

where we do our console deadlock text pressed okay now we said on press to

handle press save the changes take a look so I tap on this now if you look at

the terminal window you see our lock message so we talked about the two essential props for the text component now as you're building

apps get yourself used to reading this documentation for every component you come across it helps you better understand the capabilities and

limitations of that component next we're going to talk about images now let's

## Image

talk about rendering images first let's restore our app to its original state so

let's change the background color to white now to put our content in the

center of the screen we have to set two properties justify content we set this

to Center and align items we should set this to Center as well

okay so now anything that we put inside this container that is our safe area of

you will be in the center of this screen like this okay now to render an image

first we import the image component from react native

with this component we can display both local images that we bundled with our app as well as network images that we download over the Internet so currently

in our assets folder we have two images icon and splash which is used in our app is loading so let's display the icon right below the text so here we type

image now because we're not going to put anything in between the image tags we

can use the self-closing syntax that makes our code cleaner now here we should set the source prop to load our icon we use the require function and

specify the path to our image so our app component is right here we should go to

the assets folder and load icon the PNG so here we type period which represents

the current folder then we go to assets and load icon that PNG so when we use

the require function react native packager will include this file in our bundle so it's gonna increase the size of our app so we should use static

images if they really have to be shipped with our app like the icon or splash screen otherwise we should download images from the internet so now save the

changes here's our image beautiful now what do you think this required function

returns it doesn't return an image or a string it returns a number that is a reference to our image let me show you so before our return statement

let's do a console log and print require assets slash icon that PNG now look in

the terminal so to is the reference to our image okay alright now let's delete this line so this is how we can render local or

starting images but what about Network images well let's head over to pick some

dead photos this is a random image generator here's an example look at this URL every time we hit this URL we'll get a random image with these

dimensions 200 by 300 so let's copy this now back to vs code for Network images

instead of the required function we have to pass an object here now this object

will have a property called URI which we set to a string and this is where we paste the URL that we copied so save the changes back to our simulator our image

is not visible because react native doesn't know its dimensions so we have to manually specify the dimensions for network images for local images we

didn't have to do this because the require function reads the metadata about our images okay so back to our code

and the object that we passed to the source prop we set two extra properties

with the 200 and height to 300 save the changes so here's our image component

and here's the result beautiful now here on the official

documentation you can see various props for the image component let's talk about a few useful props here we have blur radius

this applies a blur effect to our image so here we can set the radius to let's say 10 now look our image looks blurry

we also have loading indicator source this is similar to the source prop so we

can give it a local image using the required function or we can pass the URI of a network image the image we pass here will be displayed while the actual

image is being downloaded we have another prop called fake duration as you

can see in this table it's only supported in Android so Android loads this image with a fade effect and by default this fade effect takes 300

milliseconds let me show you here's my Android virtual device we can reload the

app by pressing R twice now look at the Fed effect did you see that we can

change this duration to make it more pronounced so here we can set fade duration to let's say 1 second now let's reload the app and here's our fade

effect beautiful so this only works on Android it has no effect on iOS another

useful prop is resize mode which is used if the dimensions of our image is different from the dimensions we specify so here we have various resize mode we

have cover contain stretch repeat and so on these are the same resize modes we

have in web applications we also have a bunch of events like unload on load and

unload start and so on with these we can tap into certain moments when an image is being loaded

## Touchables

in the previous lesson we added this image to our app now let's take this app to the next level and allow the user to tap on this image with our text

component we achieve this by handling the unprecedent right but the image component doesn't have this prop or event this is where we can use touch

upon components so we can make anything touchable here are the touchable components in react native we have touchable highlight touchable opacity

touchable without feedback the touchable component we use will depend on the kind of feedback we want to give to our users let me show you so back to our code

let's import touchable without feedback and note that I'm not typing the

complete name of this component I'm using shortcuts so I'm typing a little bit of it like tou WF and then press Enter this is much

faster than typing the complete name of a component okay now to make this image touchable we have to wrap it inside a touchable without feedback component so

once again I'm gonna use a shortcut to touchable without feedback press Enter

now let's move this image inside touchable without feedback so I'm

holding alt and then pressing the up arrow with this we can move code like this okay now this touchable without feedback has

an event called unpress it also has an event called on long press this is

useful let me want to allow the user to tap and hold on a component so here let's handle the unprecedent here we can pass a

function let's do a consult log and say image tapped now save the changes

when I tap on this image nothing happens because we're using touchable without feedback it doesn't give us any visual feedback but if you look at our terminal

we can see our message now let's replace this with a different kind of touchable

so on the top let's import touchable opacity now let's replace this with

touchable opacity so touchable opacity which will also replace this tag take a

look so let me tap on this image its opacity gets reduced so we can see the

background this is why this touchable is called touchable opacity for a fraction of a second it reduces the opacity of what we are making touchable okay now

let's look at touchable highlight so we imported on the top touchable highlight

and then use it over here now using command + D as I told you

before we can select both instances and replace them in one go

touchable highlight save the changes take a look

so when we tap on this image its background is darkened for a fraction of a second this is the effect of touchable highlight so these are the three

cross-platform touchable components we have in react native we also have a touchable that is specific to Android it's called touchable native feedback

it's not supported on iOS so when we use it we get a warning let me show you so

let's import touchable native feedback now

let's use it here touchable native feedback save the changes on iOS we get

this red box touchable native feedback is not supported on this platform so later in this section I will show you how you can detect the platform in which

this app is running so if he's running on iOS perhaps we can use touchable opacity if it's running on Android we can use touchable native feedback now

let's try this touchable on an Android virtual device so I'm gonna bring up my Android virtual device when I tap on this image nothing happens because this

touchable doesn't really work with images it works with views that have a background color let me show you so back to our code on the top let's import the

view component now we're going to replace this image

with a view let's give this view a few styles so we

set this time prop to an object with a few properties with let's say 200 height

I'm gonna use 70 and background color let's set this to Dodger blue let's save

the changes take a look this is the native feedback effect that we have on Android so this is all about

touchable next we're going to talk about buttons

## Button

now let's talk about the button component so I've cleaned up the code here we only have a safe area view no text no image no touchable component

here we want to add a button to our view so we can import it on the top and then add it over here but let me show you a shortcut here we can type button now

hearing the intellisense look we have auto import so we can have vs code automatically import this for us so here we press enter and our button is

imported here beautiful now with buttons we can use the self-closing syntax because we don't put anything in between them so here we set

the title - lets say click me and we handled the unprecedent pretty

straightforward let's log button tapped now save the changes

so here's our button on an iPhone and here's our button on an Android phone

so each platform has a different way of presenting this button component because as I told you before this button component that we're using here gets

mapped to its native equivalent son Android buttons look like this by default now we can change the color of this button for example we can set the

color to our inch save the changes now our button has an orange background

and on iPhone it has an orange text we can also create a custom button that has

a unique look and feel I'll show you how to do that in the next section when we talk about styling next we're going to talk about alerts

## Alert

let's make this app more interesting instead of printing something on the console let's display a standard alert box so I'm going to delete console blog

and use the alert function so the alert function that we have in browsers also

works here so here we can display a message like button tapped save now take a look on iOS we get this standard iOS alert

box and on Android we get a different kind of alert again this is because the

alert we display gets mapped to its native equivalent now let me use the alert function we get a box the title of this box is alert and here we have a

single button called okay if this doesn't work for you you can always customize it you can change the title you can change the buttons here let me

show you so on the top we import alert this is not a component that has a

visual representation this is on an API so it's an object with a couple of methods let me show you so I'm gonna put this on a new line now instead of alert

you're gonna type alert dot look here we have two methods alert for displaying a

message and prompt for asking a question and getting an answer so let's use the

alert method this method has a few parameters the first one is the title of our alert box let's set it to my title the second parameter is our message I'm

going to set this to my message the third parameter is the array of buttons

so I'm going to pass an array here in this array we add an object now press

ctrl + space you can see the properties of this object so every button can have

a text it can have a style and it also raises the unprecedent so let's set the

text of the first button to yes now we add a second button and set its text to

no save the changes so this is what we have now take a look now we have a

custom alert the title has changed and we have two buttons here now how do we know which button was clicked that's very easy we just have to handle the

unpress event of these two buttons so here we can say on press we can set this

to a function to a console that log of yes and over here we can set on press to a different

function and here we can log a different message on the console take a look yes now back in the terminal here's our

message beautiful now let's look at the prop method so I'm gonna delete this via

lines here we call learnt that prompt this

method also takes a title so here's the title of our box we also give it a

message now the third parameter is a callback or buttons so we can pass a callback function that takes a parameter of type

string this is the text that the user enters into the box so we can get that imprinted on the console pretty straightforward so save the changes

now let's train on iOS so here we have this input box where we can type something so let's say hello world now if we press ok our callback

function will get called so we get our message in the console

now one thing you need to know is that this API only works in iOS at the time of recording this video so on Android when I tap on this button nothing

happens hopefully this will change in the future another useful API you need

## StyleSheet

to know about is this dive sheet API you have seen this before with this we can define stats but let me clarify a few things about stars in a react native app

first of all as I said before these does we have here they are not CSS their

names are inspired by CSS but they are not CSS these are just regular JavaScript properties so when we build our app react native will map is

platform agnostic components to their native equivalent and then it will apply these properties on them okay so that means this object that we're passing

here is essentially a plain JavaScript object so instead of referencing this object Styles that container we can pass an inline object here and set the

background color to let's say orange now look we have this orange bar on the top

because we are playing a single style we are only setting the background color okay we can also define this object somewhere else for example I can cut

this from here let's define a constant called container style and say to this object now we can reference this object over here

container style our app still works now previously we were using styles that

container styles is the object that we're defining over here so when we call

strategy that create we pass an object this object that we pass here is the

same object that we get as a result so our Astana's object has a property

called container which is an object itself this is the reason why we can reference styles that container over here so

whatever we pass to the create method we get it as a result now you might ask but

what is the purpose of this method why can't we just use a plain JavaScript object why do we have to call stylesheet that create well there are two reasons

the first reason is that this method validates the properties that we use here so it ensures that we don't accidentally misspelled a property so if

he misspelled this property let's say we add an e at the end look what happens we

get this red box background color with an E is not a valid style property this validation doesn't happen if we use a plain JavaScript object directly so let

me revert this back now let's pass an inline object and set its background

let's misspell it to something else look we don't get any errors but our app

doesn't have the right look either so this is one benefit of using stylesheet that create let me remove this so star as that container so this method

validates the properties that we pass here the second benefit is that the react native team have been working on implementing some kind of optimizations

behind the scene as far as I know these optimizations are not available at the time of recording this but they're coming in the future so it's the

recommended practice to use the statute API to define stats so this is all about this stylesheet api now here you can also combine multiple styler objects so

instead of passing a single object we can pass an array of objects so in this

array first we have styles that container and then we can add a second object like this object that we defined over here now take a look

the result is the combination of these two style objects kind of similar to how we can apply multiple CSS classes to an HTML element but here the results are

more predictable so the object on the right over eights the properties of the

object on the left in this case container style is defining a single

property that's background color so this is over writing the background color

that we defined earlier so the results are more predictable now another

question you might have is do we have to put these styles in the same file not really you can extract this from here put it in a separate file and then

import it in this file that's totally fine but it's very conventional to define their size below it component because quite often you need to work

with both the structure of your component as well as it stars in the same session if you put the size in a separate file you have to constantly go

back and forth between two files I personally even though I'm all about writing clean code and separation of concerns I prefer to have those styles

below my components it has worked better for me but if you don't like it that's totally fine you can separate your status in the next section I'm going to

talk about stars in detail for now that's all you need to know about stars

## Platform-specific code

there are times way to detect the platform in which our app is running and customize some styles or behaviors here's an example look at our status

object let's remove these two properties so our button is no longer at the center of the screen take a look so it's here on the top but look at our

Android virtual device our button is below the status bar so this safe are

your view component that we used earlier it only works for iOS so it makes sure that our content is not covered by this notch on iPhone it has no effect on

Android at the time of recording this video so in this case we need to manually add some padding on the top to push this button down so it's no longer

behind the status bar this is where we use the platform module so on the top we

import the platform module from react native now over here we can add a third

style padding top we want to set this dynamically if the current platform is

Android we're gonna set this to let's say 20 otherwise we want to set it to zero so here we type platform now this object has a few properties os returns

the operating system which can be Android iOS and so on we can also get the version of our platform we can check to see if this is an iPad if this is a

TV and so on so let's read this property we can compare

this with now here we press ctrl + space we can see various values that are accepted so we have Android iOS Mac iOS web and windows so if this is Android

you want to set padding top to 20 otherwise we're going to say 2-0 say

take a look now our button is below the status bar but wait 20 well this was

just for a demo the proper way to do this is to calculate the height of the status bar and use that as the value for this style so on the top we gonna import

the statusbar api now we're gonna replace 20 with status bar dot current

height this is the proper way to add padding on the top because the height of the status bar might be different from one Android phone to another so save the

changes now our button is just below the status bar

hey guys Marsh here I just wanted to let you know that this tutorial is the first two hours of my ultimate react native course the complete course is divided

into two parts basics and advanced topics but each part being about five to six hours long so the complete course is over ten hours long it includes all the

source code rewriting this course every section has it before and after source code so you can easily code along with me plus you will get plenty of exercises

and step-by-step solutions and a certificate of completion that you can act your resume if you're interested I put the link down below I'm offering a

discount to the first hundred students so if you're interested in role now before it's too late now let's continue to the next lesson in

## Layouts

this section you're gonna learn how to create layouts in react native and this is where the fun begins because we're gonna build the first two

screens of our app so we're gonna talk about dimensions device orientation flexbox as well as absolute and relative positioning so pay close attention to

the lessons and take note because you're going to use these materials as part of the exercises at the end of this section so I'm super excited about this section

I hope you're too now let's jump in and get started

## Dimensions

let's talk about the dimensions of components on the screen so let's import

the view component from react native now in our safer your view let's add a view

we give it a style let's set the background color to Dodger blue I'm

gonna give this a width of 150 and a height of 70 now these numbers we have

here are if density-independent pixels or dips the actual size equals

density-independent pixels times scale factor of the device for example iPhone

4 can display 320 by 480 points these points are abstract units they're not

pixels now the skill factor of iPhone 4 is 2 or 2x that means every point contains 2 pixels so the screen resolution of iPhone 4

equals 640 by 960 pixels now if the with our view is 150

density-independent pixels or dips its actual width on an iPhone 4 will equal

150 times 2 which is 300 pixels that is roughly around half of the screen width

right now in contrast iPhone 11 pro max can display 414 by 896 points with a

scale factor of 3 here's the screen resolution of iPhone 11 pro max now what

is the width our view on this iPhone it is 150 times 3 equals 450 pixels so

again it's roughly around half of the screen with roughly not exactly in fact

our view looks a little bit smaller on an iPhone 11 pro max but don't worry about memorizing any of these numbers they don't matter what matters is that

by expressing the size of our components in density-independent pixels we can feel relaxed that they look almost the same size across different devices now

if you want to make sure that this width is exactly half of the screen width which with a percentage value here so we

replace 150 by 50% make sure to put this in quotes because

this is a string value now save so here's our view it's taken exactly half

of the screen now in some situations you want to fine tune the size of a component according to the size of the screen

in those cases we can use the dimensions API so on the top we import

dimensions from react native now let's lock

dimensions that get here we should pass a string that can be either window or

screen screen returns the size of the entire screen whereas window returns the

size of the visible application window on iOS these dimensions are equal they're only different on Android so the window size is a bit smaller than the

screen size so I'm gonna pass screen here now save

at the terminal so here you can see the width and the height of this iPhone as well as is scale factor so every point on this iPhone has 3 pixels so this is

how we can get the dimensions of the device using the dimensions API now the problem in this API is that it doesn't respond to orientation changes so if the

user rotates the device these numbers don't get updated we'll talk about how to handle that in the next AESA

## Detecting orientation changes

there are times want to detect our screen orientation and resize our components accordingly for example let's set the width of this

view to 100% and its height to 30% so imagine this is a media player in

portrait mode you want to have this video player on top and in landscape mode we want to have this take the entire screen right now the height of

this imaginary video player is always 30% of the height of the screen this is where we need to detect the orientation of our screen and resize this component

accordingly and by the way to rotate this iOS simulator we hold down the command key and use the left or right arrows like this okay and for Android we

have this toolbar now look at the shortcut on Mac its command + left on Windows it's probably ctrl + left I'm not entirely sure

so to support different orientations first we go to app to JSON

by default the orientation of our app is set to portrait so it only supports the portrait mode we can set this to landscape to only support the landscape

mode but this is not very common we can also set this to default to support both modes so let's save the changes now to detect screen orientation we're going to

use this library called hooks develop by react Native community so on this page

you can see we have various hooks hooks are functions that bring extra capabilities to function components for example we can add state to a function

component you see one of the built-in hooks in react if you're not familiar with hooks I highly encourage you to watch the last section of my react

course I covered hooks in detail in that section so here we have various hooks or various functions all these hooks by convention start with use for example we

have used back Handler to work with the back button on Android we have used camera roll we have used dimensions use device orientation and so on so first

let's install this library here in the terminal window at react - native -

community / hooks

all right this is installed now let's go back to after Jas

first special import use dimensions from at react native community and slash

hooks with this hook we can get the correct dimensions of the screen whether we are in portrait mode or in the landscape mode this hook always returns

to updated dimensions this is one of the limitations of the dimensions API in react native so this is the preferred way to get the dimensions of the screen

if you support multiple orientations so let's do a console that log statement

and call use dimensions say now here you can see the width of our screen is 414

now if I rotate this device in the landscape mode we get this new with 896

so every time we rotate our device our component gets recalled and here we

recalculate the updated dimensions now we also have another hook for detecting

the screen orientation use device orientation so let's call it here

so let's reload the app in portrait mode okay so we get an object this object has two properties landscape is false and

portrait is true now if I rotate this simulator we get an eel object now landscape is true and portrayed as false so to make

this imaginary video player take the entire screen in landscape mode we can write code like this first we call this function then we

store the result in this object or we can use object destructuring here and

pick the landscape property from that object now we can calculate the height

dynamically so if you're in the landscape mode we're gonna set the height to 100% otherwise we're gonna say it to 30% take a look

so let's reload the app

all right in landscape mode our video player is taking the entire screen we have this white edges this is because we put this inside of a safe area of view

perhaps for a video player this is not something we want to do we want to make sure that the video player takes the entire screen but let's not worry about

it in this lesson now if we rotate this too right the height of our video player will be 30% of our screen height so using the

hooks in this library we can get the dimensions and the orientation of our device now for the app that we're going to build on this course we're not going

to support the landscape mode so I'm going to go back to after JSON and set

the orientation back to portrait

## Flexbox

now let's talk about flags for a flexbox with flex we can easily build complex

layouts that work consistently across different screen sizes we are the same concept in CSS but flexbox in react native is a little bit different so make

sure to watch the next few lessons even if you are familiar with flexbox in CSS so I've removed all the code we have written so far we're only importing the

view component from react native now in our app component let's return a view and give this view a couple of stats so I'm gonna set the

background color to dodge or blue and flex to one

when we set Flex to one this view grows to take the available free space so save

our view is taking the entire screen what if you set flex to 0.5 now our view

takes half of the screen okay so with Flex we typically set up a view as a container and then align children inside that container so let's imagine this

view is our container so let's change its background color to white and set

flex to 1 so it takes the entire screen now inside this view we're gonna add

another view here we can use the self-closing syntax because we're not

gonna put anything inside this view now let's give this view a couple of styles

so I'm gonna set the background color to Dodger blue and flex to one let's see

what we get once again we get a blue screen because our parent or container view is taking the entire screen and inside this

container we have this other view the blue view which is growing to take the

available free space so it feels it's container that is why the entire screen looks blue now with this view select that let's hold down shift and alt and

then press the down arrow for this we can duplicate coding vs code so let's

duplicate this one more time now we have three views the first view is Dodger

blue the second view let's make it gold and the third view let's make it tomato

that's a kind of red so this is what we get so now our screen is divided into

three segments each view is taking 1/3 of its container or 1/3 of the screen

now what if we set the Flex property of this blue view to 2

now the blue view is twice as big as the other views with this setup we're essentially dividing the space into four segments why four well we have two plus

one plus one so we have four segments two out of four segments is allocated to

the first view to the blue view so this view is taking half of the screen or half of its container and these other views each is taking one-fourth of the

screen so this is the basics of flex over the next few lessons we're gonna study other properties of flex

## flexDirection

now let's give this viewers a fixed-size so in this blue view I'm gonna remove the Flex property let's set the width to 100 and height to 100 as well now you're

going to repeat so let's change the second view and the

third view here's what we get our views are laid up vertically in a column because that

makes more sense in mobile apps typically we hold our phones in portrait mode so that's why by default react native vertically aligns the items in a

container if you have worked with flexbox in CSS you probably know that by default items are laid out horizontally so this is one of the differences we

have in react native now to lay this out horizontally we go to our container

so here's our container we set the Flex Direction property to row

here's what we get we also have real reverse

so items are laid out from right to left and we also have column reverse

so here's what we get now any time if you wonder what values are acceptable here simply delete this press ctrl + space so here are the list of values

column which is the default value column reverse row and row reverse now let's

set this back to row with this setup we say our main or primary axis is the

horizontal axis and the cross axis is the vertical axis this is an important

property of flex we're going to get back to it over and over next we're going to talk about aligning items

## justifyContent, alignItems and alignSelf

in this lesson we're going to talk about aligning items in a container so currently our items or our views are appearing at the top left corner of the

screen what if you want to push this to the right or put them in the center of the screen we go to our container and set the justified content property to

one of these values so let's set it to center here's what we get so with this

property we can align items along the main or the primary axis what is our

main axis here it's the horizontal axis because earlier we set Flex direction to

row so now our main axis is the horizontal axis now what if we change

this to column now our items are appearing at the center of the vertical axis because the vertical axis is now our primary or main

axis this changes back to Rome and look at the other values so we have

flex and now our items are appearing at the end of the horizontal axis we also

have flex start which is the default value and we have three properties for

distributing space we have space around so with this property the space between

each two items is equal so look at the space here it's equal but the space

around the first and last item and the edges of the screen is different so here

we have less space if it changed this to space evenly

now the space is evenly distributed between items and we also have space

between with this setup the first and the last items are pushed to the edges of the screen and the remaining space is

distributed between the other items so let's change this back to Center

now we have another property called aligned items and with this we can align

items across the secondary axis so what is our secondary axis here is the

vertical axis right so if we set this to Center now our items appear at the

center of this screen let's look at the other values you have baseline

the impact of this unless we change the height of one of these views so let's

change the height of the first view to 300 and the second view to 200

now these viewers have the same baseline okay let's look at another value flex and this is pretty self-explanatory

so our items appear at the end of the secondary axis okay what else do we have

here flex start now the items appear at the start of the vertical axis and finally we have

stretch which is the default value now we don't see the impact of this property unless we remove the height of one of these views so I'm gonna comment this

out see what happened the blue view stretched to fill the entire vertical

axis so this is the default value if we comment out aligned items we get the

exact same result so let's bring it back and change it to Center and bring back

the height property as well so we justify content we align items across

the main axis and with align items we align them across the secondary axis now

the first time I learned about these properties I was a bit confused I was wondering why the names are not consistent every time I wanted to align

something I wasn't sure which property I had to use honestly it's a bit weird I

know but you will get used to it as we go through the course you're gonna use this properties over and over so that would be second nature to you now what

if you want to change the alignment of one of these items let's say we want to put the blue view over here so we go to this view and set aligned self to flex

start so look at these two properties we have aligned items which we apply to the

container and we have aligned self which we apply to an individual item now

here's what we get so this is all about aligning legs we're gonna talk about wrapping items

## flexWrap and alignContent

let's talk about wrapping items so I'm going to go to the last view

this view over here let's duplicate this change the background color of the new

view to gray so here's our new view and we have a bit of extra space here so

what do you think will happen if we add an extra view here let's find out so let's duplicate this one more time and change the background color of the

new view to green yellow do you see what happened the blue view got shrunk so the

green view can fit on the screen so this is the default behavior if our items overflow across the main access one or more items get shrunk so other items can

fit on the screen we can change this behavior by enabling wrapping so we go

to our container here's a container we apply a new style

called flex wrap the default value is nowrap we said this to wrap but when I

save the changes the alignment of this items is gonna get screwed take a look see what happened so we have wrapping because the green view is appearing on

the second line and the blue view is no longer shrunk but what happened to our vertical alignment we wanted all these items to be vertically in the center of

the screen this is where a lot of people get confused so let me clarify for you when we enable wrapping the align autos property behaves a bit differently so if

you have multiple lines the align autos property determines the alignment of items within each line let me show you with a good example so I'm gonna change

the height of the first view the blue view to 300 see what's happening within each line our items are vertically centered now

currently we have a single item on the second line but if you had multiple items with different heights all these items would also be vertically centered

so a line items property determines the alignment of items along the secondary

axis with an H line now if you want to put all these items together in the

center of the screen or in the center of the vertical axis we use a different property it's called aligned content so we want

to align the entire content as a whole we said this to Center now all these

items are appearing in the center of the vertical or the secondary axis so this

is the difference between a line items and aligned content the line items determines the alignment of items within each line and aligned content determines

the alignment of the entire content that line content only works if we have wrapping if you don't have wrapping it has no effect so this is all about

wrapping so we have covered all the essential properties of flex but there

## flexBasis, flexGrow and flexShrink

are three other properties that you need to understand you may not use them as much but it's good to know them in case you see them in someone else's code so

here's our blue view here we can apply a property called flex basis

and with this we can set the size of an item along the primary axis so our

primary axis here is the horizontal axis so if we set this property to 100 this

is equivalent to setting the width to 100 so if I comment out this property and save the changes look we get the same result knife our primary axis was

the vertical axis setting Flex basis would be equivalent to setting the height property so Flex basis can map to width or height

now we have another property flex grow if we said this to one the moment I save

the changes you will see the blue view grow to fill the available space take a

look here it is in fact selling flex grow is exactly the same as setting the

Flex property now if I say the changes we are not going to see the same result

because I don't know if there is a problem with a simulator or the tooling so let's verify this save look we don't get the same result

but if you manually refresh using the developer menu we see the same result as applying the Flex grow property okay now we have

another property called Flex shrink honestly it's not something used that often but let me explain how it works it's essentially the opposite of Flex

grow so to simplify things let's get rid of these two properties and set the

width to 400 now our blue view is taken so much space so the orange view is not

fitting on the screen we have overflowing in this case we can set Flex shrink to 1 and with this we are saying that hey if you have overflowing this

item can get shrunk so other items can fit on the screen take a look save there you go I told you that flex is a shorthand for flex grow and flex

rank so if it said flex to a negative number this is exactly the same as

setting Flex rank I save the changes we don't get the same result because there

is a problem going on here so you have to manually refresh now we get the same result as setting the Flex shrink property so this is all

about Flex bases Flex grow and flex rank

## Absolute and Relative Positioning

in this lesson we're going to talk about absolute and relative positioning in react native so we have our container with three items in it just like before

now what if you want to move this gold view without changing the layout around it it's very easy so here's our gold view we can set top to 20 now when I

save the changes this gold view will move 20 independent pixels from the top look here it is we can also set a negative value

so our gold view move- 20 pixels from the top this is exactly the same as

setting the bottom property so if I said bottom to 20 that means you want to move

this gold view 20 pixels from the bottom we also have left on right so let's move

this 20 pixels from the left or 20 pixels from the right

so with these properties we can position a component relative to its current position without affecting the layout around it so in all these examples the

blue and the red views have been exactly where they're supposed to be they didn't move around right this happens because in react native all

components by default have their positions set to relative so this is how

relative positioning works in CSS as well in contrast to relative positioning we have absolute positioning now if I save the changes

this gold view would be positioned relative to its parent which is this container that takes the entire screen and these other views will move around

as a result of this positioning let me show you before I save the changes let's set top to 20 and left to 20 as well so we can see clarity so save look our gold

view is positioned 20 pixels from the left and top of its parent that's the container that takes the entire screen and as a result of this positioning

these other views moved around so they're not in their original position let me show you one more time so I'm going to change position back to

relative look with relative positioning these two

views did not move they stayed in their original position if we use absolute

positioning for this gold view it will be positioned relative to its parent and these other views will be repositioned look they moved around okay so to recap

in react native all components by default have their position set to relative which means we can move them relative to their current position

without changing the layout around them if we change the position to absolute we can move a component relative to its parent and the layout around it will get

affected this is all about absolute and relative positioning in react native

## Exercises

all right now it's time for an exercise so I want you to implement two screens of our app the welcome screen and the view image screen for the welcome screen

we have this logo and this background image you can download these below this video now in this section our focus is purely on laying out components on the

screen so we're not concerned about styling for example our text here it's very small and it's close to the logo we don't want to worry about these details

we're gonna talk about styling in the next section so then we're gonna come back and fix those problems now this red and green boxes they are placeholders

for the login and register buttons that we're gonna create in the next section now for the view image screen again we have two placeholders for the closed and

delete icons that we're gonna add in the next section so go ahead and implement these two screens this is gonna take about 15 minutes of your time

once you're done come back see my solution all right the first thing I

## Welcome Screen

want to do is to create a new folder here in the root call app I'm gonna put

all of our application source code inside this folder this is a good practice to follow because with this we can separate our application code from

the code that is automatically generated by our tooling let's say tomorrow something crazy happens with this project we can create a brand new react

native project and simply copy your application code into that project okay so here's our app folder let's move the assets folder inside this folder

now we need to go to after JSON and update the path to these two images so

app slash assets and one more time save let's make sure that everything is

working properly good so in our app folder

let's add a new folder called screens I like to add all our screen components

inside this folder this is a convention that a lot of react native developers follow you don't have to follow if you don't like it

so in the screens folder let's add a new file called welcome screen look at the

naming convention I'm using to name this file so for this components I like to use the word screen in the file and the components name

now here we want to implement a function component earlier in the course I told

you to install this extension react native react Redux snippets with this

extension we can quickly type code so throughout the course I'm going to use the shortcuts that come with this extension you can always learn about

this shortcut down below or on the github page of this project ok so here

is an example we type our SF that is short for react stateless function now

we have the basic template for a function component with multi cursor editing enabled so here we can rename this component if you want to once we're

done we press the escape button to disable multi cursor editing now first we want to add an image background component here so let's delete the div

element we type image background here we have auto import so let's import this

now we need to set the source of this image so source equals here we call the require function then currently we are in the screens folder so let's go one

level up now we go to assets and load background dot jpg okay save the changes

that we should give this component a style so after this component I'm gonna type r n SS that is short for react native

stylesheet so this quickly generates this code style sheet that create we

store the result in this object Styles and unfortunately this is not imported on the top so we have to manually import it style sheet okay so in this object

we're gonna have a bunch of key value pairs so let's add a new key value pair it's called background this is where we're gonna add all the styles for our

image background component so I'm gonna set flex to 1 so this image background takes the entire screen ok now

here we sell our style to styles that background save let's go to our app

component and render our welcome screen so I'm going to delete all the code here

we don't need it anymore so we type welcome screen and have vyas code

automatically imported on the top beautiful let's see what we get so

here's our background image now let's add our buttons so over here inside this

component we're gonna add a view let's give it a style

so we're gonna set this to styles that login button now let's create that over

here so as you see I'm not writing in nine stands anymore I like to separate these so login button let's say it's with to 100% and its height to 70 pixels

the background color should be now let me look at my cheat sheet so that's

going to be F c5 c6 5 save the changes we can find variable view because we

didn't import it on the top so let's add it to this list say so our button is

currently on the top we want to put it down below over here how do we do that well earlier we talked about flex direction so by default flex Direction

is set to call him so our primary access is this vertical line I told you that

using justified content we can align items along the primary access so we go

to the styles for our container which is this background image here we set

justify content to flex and the default value is flex start that is the reason

why our button appears here on the top so we said this to flex and save the

changes now our button is here beautiful let's duplicate this to add the second button I'm gonna rename this to register button

now let's duplicate these styles and rename is to register button I'm gonna

use the same width and height but change the background color to 4e c-d-c for now

you might be thinking I'm typing too fast but at this point you should have already done your exercise you shouldn't code along with me so I

don't want to waste a lot of people's time by typing really slowly okay so save the changes here's our second button beautiful now we need to add the

logo on the top but here's a problem if you had the logo because we said justify

content to flex and our logo is gonna end up here so how do we put it on the top this is where we can use absolute positioning so we can position the logo

relative to its parent which is this background image so

let's hat here image now let me show you something in this case auto import didn't work

this item with this blue icon so if you look on the right side we don't have auto import sometimes this happens in the context menu that appears here so

look we have two image instances the second one where the orange icon has

auto import so make sure to select that one so now you can see image is imported

on the top so here I'm using the self-closing syntax because we're not

gonna put anything inside this image now let's set its source once again we used

a required function we go to the assets folder and load logo - red dot PNG let's

test the result up to this point our logo is way too big so let's give it a couple of stars we set style to styles that logo now over here

and by the way note that I'm sorting these keys so if you have background

login button logo and so on this makes it easier to maintain your applications

so let's set the size of this logo to 100 by 100 so width is 100 and height is

100 as well save okay this is good our logo is down below

now we should use absolute positioning so I'm gonna set position to absolute

and let's put this 50 pixels from the top of the screen so here's what we get

we can push it down a little bit let's say 70 that is better but how do we put

in center well Ernie we talked about the primary and the secondary access in flex so our primary axis is the vertical line and the secondary axis is the horizontal

line using what property can be aligned items along the secondary axis using

align items so you go to our container over here

we said align items to Center and with this we can align items along the

secondary axis take a look now the image is right in the middle of the screen

beautiful so finally we need to add our tagline below this logo

so let's add a text component so we have this orange icon here so we can

automatically import this what was our tagline cell what you don't need say

our text appears here so what we need to do is to put the text and the logo

inside a container and I apply absolute positioning on that container okay so

let's add a new view here then we're gonna move the image and text components

inside this view so with these lines selected I'm gonna hold down the Alt key and then press the up arrow with this we can move code okay so now let's apply a

style here we can say styles that logo container whatever you want to call it

now let's create this so here's logo now we create local container

this is very simple I just want to move these two properties inside this object

let's see what happens so our text is in the right position but

our logo is not exactly in the middle of the screen this happened because once we

added this new container the alignment of this container is reset to flex start

by default so in this container once again we need to set align items to

center so we can align the items along the secondary or the horizontal axis

take a look and here's the final result in the next section we're going to apply

a bunch of styles to make this screen pretty now if you want my source code you can get it from the zip file that I gave you at the beginning of the course

in the first section that zip file contains all the source code that we're gonna write throughout this course so every section has a start and a finish

folder where you can find a relevant source code

## View Image Screen

all right now in the screens folder let's add a new file called view image

screen that Jay has here we're gonna create a function component I'm happy

with the name now in this component first we want to add an image so image

let's set its source to require we go to the assets folder and load chair jpg

now let's test our application up to this point so we go to after j/s and

render view image screen ok let's see what we get so we get this kind of

whitish screen because this image that I have supplied is very big and it doesn't fit on the screen so we need to apply a bunch of stars here we set style to

first we should create a style to object so we type our NSS good now let's import

style sheet on the top so in this object we're going to find a

bunch of stands for our image I'm gonna set the width to 100% and the height to

100% as well and finally let's add style to styles

that image let's see what we get so here's our image but if you pay close

attention the sides of the image are cut off let me show you so here in the project let's take a look I just shared that jpg

do you see the sides of this basket is cut off here this is because of the resize mode so to solve this problem

to our image and set the resize mode to contain now our image perfectly fits on

the screen but we have this white background so we have to change it so let's wrap this inside the view that's going to be our container so view let's

import it now we give it a bunch of styles so style equals styles the container now let's define the stance it's a

container once again I'm trying to sort these alphabetically this makes it

easier to maintain our application also if you see eslint instant automatically does this for you but in this course I'm not using es lane

because it keeps complaining and it creates a bad experience for you so I have to manually sort these so this container we're gonna set its background

color to black and I also like to set Flex to one to make sure that this view

takes the entire screen so here's what we get beautiful now let's add the

placeholders for the clothes and delete buttons so back to our container let's

add another view give it a style let's say styles that close icon now

we define the styles close icon comes before container so let's set its width

to 15 the height 15 as well I'm gonna set the background color to FC 5 C 6 5

now if we save the changes our button appears here to solve this problem we're

gonna use absolute positioning we want to put this exactly right here so we set

position to absolute so now we can position this component relative to its parent which is the container so let's say we want to position this 40 pixels

from the top and 30 pixels from the left so it appears here I'm happy with this

so let's move on now let's add our second placeholder so I'm gonna select

this hold down shift alt and down to duplicate this code now let's rename this to delete icon

and then when you find a stylist delete icon

now to save time I'm going to copy these fuel styles I know copying code is

bad but they're gonna fix this in the future so for now don't worry about it so let's change the background color to 4e CD C for now we want this to be 30

pixels from the right of the screen take a look here's what we get so I'm

happy for now as we go through the course we're gonna improve our code we're gonna refactor it we're gonna make it more professional

## Refactoring

alright one problem I see in our current implementation is the duplication of these color codes we have repeated this in a few places so what we can do now is

to extract these color codes put them in a separate file like colors of Jas and with this we have all the colors that our application users in a single place

so if tomorrow we decided to rebrand this app and use different colors there is a single file we have to modify so in our project inside the app folder let's

add a new folder called config here we add a new file called color CAS now in

this file I'm gonna export a default object with these properties we can

define our primary color so we're gonna set this to F c5 c6 5 and our secondary

color to 4e CD C for now save the changes we go back to our view image

screen on the top we import colors from config slash colors and note that I'm

separating the import statement from my code from the import statement from our third-party libraries again this is a convention that a lot of people follow

to make their code clean and maintainable so we imported colors this

is an object that we exported over here ok so we can replace these hard-coded values with colors that primary and colors

that's secondary save the changes so take a look we get the same result as

before but our application code is more maintainable now we have a single place where we can define our color palette now some people argue that even black or

white should not be hard-coded they should be part of the palette so in our

color palette we can define a new property black and set it to zero zero

zero because tomorrow if we decide to rebrand this app we might use a different dark color rather than pure black so it's a good practice to include

it here so save now we can replace this hard-coded value with colors that's

black that's good now we have another problem here the duplication of these

properties so both our icons these icons over here have the width and height of

50 so you might be wondering if we can define this properties in a single place and reuse them in two places we certainly can but we're gonna get rid of

this view in the future we're gonna use a real icon so for now I'm not too worried about this this is just a temporary solution

hey guys ma chère so as I told you before this tutorial is the first two hours of my ultimate react native course the complete course is divided into two

parts basics and advanced topics but each part being about five to six hours long so the complete course is over ten hours long it includes all the source

code rewriting this course every section has a before and after source code so you can easily code along with me plus you will get plenty of exercises and

step-by-step solutions and a certificate of completion that you can add to your resume if you're interested I put the link down below I'm offering a discount

to the first 100 students so if you're interested enroll now before it's too late thank you and I hope to see you in the

course

English

# Debugging JavaScript - Chrome DevTools 101

KAYCE BASQUES: If you're still using

console.log to find and fix JavaScript issues,

you might be spending more time debugging than you need to.

This tutorial shows you how to make

the most of Chrome DevTools so that you

can debug your JavaScript as quickly as possible.

I'm going to use this buggy demo here

to demonstrate all of the debugging tools in DevTools.

I recommend that you open up this demo

yourself and follow along with each step.

The link is in the description.

Whatever issue you're debugging, you always

want to start by reproducing the issue.

You want to find a series of actions that consistently

reproduces the bug.

In this demo, when I add 5 and 1,

the result down here at the bottom is 51.

Obviously that should be 6.

This is the bug that I need to track down.

At this point, you might be tempted

to use console.log to infer where the code is going wrong.

Sure, it may get the job done, but it's rather inefficient.

First you have to find the relevant code, which could take

a while in a big code base.

Then you have to sprinkle console.log statements

throughout the code.

Then you got to reload the page and look at the logs.

But maybe the logs didn't give you

all the information you need.

So you got to go back and add more logs and so on.

With DevTools you can pause the code

in the middle of its execution, inspect the variables that

are in scope at that moment in time,

and walk through your code one line at a time.

Open DevTools by pressing Command Option J on Mac

or Control Shift J on Windows and Linux.

Then click the Sources tab.

The Sources panel is where you debug JavaScript.

It has three main parts.

At the top left here is the file navigator pane

where you can inspect the files that the page uses.

After clicking on a file, you can see the contents of it

here in the Code Editor pane.

And down here is the JavaScript debugging pane,

which you'll learn more about shortly.

Note that when your DevTools window is wide,

the JavaScript debugging pane moves to the right.

If you take a step back and think about how the app works,

it's probably safe to guess that the incorrect sum gets

computed in the click Event Listener that's

associated to the button.

Therefore you want to pause the code right when

the Event Listener executes.

Event listener breakpoints let you do exactly that.

Expand the Event Listener breakpoint section,

then expand the mouse category, then check the click check box.

DevTools now pauses on the first line of any click Event

Listener that executes.

Back in the demo, click the button again.

And DevTools pauses on the first line of this on click function.

The blue highlight indicates what line of code

you're currently paused on.

If you're paused on a different line of code,

press Resume Script Execution until you're

paused on the correct line.

You paused on the wrong line because you

have a Chrome extension, which registers click Event Listeners

on every page that you visit.

If you try the workflow I'm describing in an incognito

window, you can see that you always pause

on the correct line of code.

Event Listener breakpoints are just one type of breakpoint.

DevTools offers many other types.

For example, you can set a breakpoint

on a specific line of code, or when a DOM node changes,

or when an exception gets thrown and so on.

After this tutorial, I recommend checking out

our breakpoints guide, which teaches you

when and how to use each type.

The link to that guide is in the description, too.

I'm paused in the click listener,

and now I want to execute the code one line at a time.

The code stepping controls right here let you do that.

Click the Step Into Next Function call button

to investigate a function further.

For example, when I step into the inputs are empty function,

it jumps me to the first line of that function.

When I'm confident that a function is working

as expected I can click the Step Over Next Function call button.

The function executes, but I don't walk through it

line by line.

For example, if I click Step into Next Function call,

this line here would be highlighted blue,

meaning this is where I'm currently paused.

But when I click Step Over Next Function Call,

the function executes to completion

and I pause on the next line of the function

that I'm currently stepping through.

Last, suppose I'm stepping through a function

and I realize it's not relevant to my bug.

In that case I can press Step Out of Current Function

and DevTools executes the rest of the function.

If this doesn't make complete sense right now,

I recommend creating a snippet, which

is a little block of JavaScript code

that you can execute at any time.

Set a breakpoint in your snippet and play around

with the code stepping controls yourself in order to understand

how they all work.

Back in the script I can tell that the bug

is probably somewhere in the Update Label function.

Rather than stepping through a bunch of irrelevant code,

I can set a line of code breakpoint

right around where the bug probably occurs.

To do that, I just click the line number next to the code.

This blue icon indicates that there's

a breakpoint on this line.

When I press Resume Script Execution,

DevTools runs all the code up until that line

and then pauses before that line executes.

Over here in the JavaScript debugging pane,

I can see the call stack that caused this code to execute.

I can click on the functions to see where each one got caught.

The scope section shows me all of the local and global

variables that are currently defined at this moment in time.

I can click on Values to edit them,

but I'm not going to do that right now.

However, when I look at these values, they look suspicious.

They're wrapped in double quotes, which

means that they're strings.

This could be the cause of the bug.

I'm going to investigate this further over in the Watch

Expression section.

Here I can watch the values of variables over time.

You can store any valid JavaScript expression here.

For example, I click Add Expression,

then type Type of Sum, then press Enter,

and I can now see the value of that expression.

As I suspected, Sum is a string when it should be an integer.

Now I'm going to open up the console to test out

a potential fix for the bug.

When I'm paused on a line of code,

the console has access to all the variables

that are currently in scope.

For example, when I evaluate add in 1, it returns 5.

I think I can fix this bug by converting the add in 1

and add in 2 variables to integers before adding them.

So let me try that now.

Yep, that fixes it.

I can verify that this change fixes the bug by editing

the code directly from DevTools.

First I'll resume Script Execution, then make my change.

To save the change, I press Command-S on Mac or Control-S

on Windows and Linux.

Then I click Deactivate Breakpoints

so that I can test out the app without triggering

the breakpoints I've set.

Now the app sums up numbers as expected.

All right.

You now have an excellent foundation

in how to effectively debug JavaScript using DevTools.

From here I recommend learning about all of the breakpoints

that DevTools has to offer, as I mentioned before.

We also have a JavaScript debugging reference

where you can learn about literally every feature related

to debugging in DevTools.

Links to both are in the description.

Thanks for watching, and happy bug hunting.

[MUSIC PLAYING]

English - CC (English)

# 5 ways to listen better | Julian Treasure | TED

We are losing our listening.

We spend roughly 60 percent of our communication time listening,

but we're not very good at it.

We retain just 25 percent of what we hear.

Now -- not you, not this talk,

but that is generally true.

(Laughter)

Let's define listening as making meaning from sound.

It's a mental process,

and it's a process of extraction.

We use some pretty cool techniques to do this.

One of them is pattern recognition.

(Crowd noises) So in a cocktail party like this,

if I say, "David, Sara, pay attention" -- some of you just sat up.

We recognize patterns to distinguish noise from signal,

and especially our name.

Differencing is another technique we use.

If I left this pink noise on for more than a couple of minutes,

(Pink noise) you would literally cease to hear it.

We listen to differences; we discount sounds that remain the same.

And then there is a whole range of filters.

These filters take us from all sound

down to what we pay attention to.

Most people are entirely unconscious of these filters.

But they actually create our reality in a way,

because they tell us what we're paying attention to right now.

I'll give you one example of that.

Intention is very important in sound, in listening.

When I married my wife,

I promised her I would listen to her every day

as if for the first time.

Now that's something I fall short of on a daily basis.

(Laughter)

But it's a great intention to have in a relationship.

(Laughter)

But that's not all.

Sound places us in space and in time.

If you close your eyes right now in this room,

you're aware of the size of the room

from the reverberation and the bouncing of the sound off the surfaces;

you're aware of how many people are around you,

because of the micro-noises you're receiving.

And sound places us in time as well,

because sound always has time embedded in it.

In fact, I would suggest that our listening is the main way

that we experience the flow of time

from past to future.

So, "Sonority is time and meaning" -- a great quote.

I said at the beginning, we're losing our listening.

Why did I say that?

Well, there are a lot of reasons for this.

First of all, we invented ways of recording --

first writing, then audio recording and now video recording as well.

The premium on accurate and careful listening has simply disappeared.

Secondly, the world is now so noisy,

(Noise) with this cacophony going on visually and auditorily,

it's just hard to listen;

it's tiring to listen.

Many people take refuge in headphones,

but they turn big, public spaces like this,

shared soundscapes,

into millions of tiny, little personal sound bubbles.

In this scenario, nobody's listening to anybody.

We're becoming impatient.

We don't want oratory anymore; we want sound bites.

And the art of conversation is being replaced -- dangerously, I think --

by personal broadcasting.

I don't know how much listening there is in this conversation,

which is sadly very common, especially in the UK.

We're becoming desensitized.

Our media have to scream at us with these kinds of headlines

in order to get our attention.

And that means it's harder for us to pay attention

to the quiet, the subtle, the understated.

This is a serious problem that we're losing our listening.

This is not trivial,

because listening is our access to understanding.

Conscious listening always creates understanding,

and only without conscious listening

can these things happen.

A world where we don't listen to each other at all

is a very scary place indeed.

So I'd like to share with you five simple exercises,

tools you can take away with you,

to improve your own conscious listening.

Would you like that?

Audience: Yes!

Good. The first one is silence.

Just three minutes a day of silence is a wonderful exercise

to reset your ears and to recalibrate,

so that you can hear the quiet again.

If you can't get absolute silence,

go for quiet, that's absolutely fine.

Second, I call this "the mixer."

(Noise) So even if you're in a noisy environment like this --

and we all spend a lot of time in places like this --

listen in the coffee bar to how many channels of sound can I hear?

How many individual channels in that mix am I listening to?

You can do it in a beautiful place as well, like in a lake.

How many birds am I hearing?

Where are they? Where are those ripples?

It's a great exercise for improving the quality of your listening.

Third, this exercise I call "savoring," and this is a beautiful exercise.

It's about enjoying mundane sounds.

This, for example, is my tumble dryer.

(Dryer)

It's a waltz -- one, two, three; one, two, three; one, two, three.

I love it!

Or just try this one on for size.

(Coffee grinder)

Wow!

So, mundane sounds can be really interesting --

if you pay attention.

I call that the "hidden choir" -- it's around us all the time.

The next exercise is probably the most important of all of these,

if you just take one thing away.

This is listening positions --

the idea that you can move your listening position

to what's appropriate to what you're listening to.

This is playing with those filters.

Remember I gave you those filters?

It's starting to play with them as levers,

to get conscious about them and to move to different places.

These are just some of the listening positions,

or scales of listening positions, that you can use.

There are many.

Have fun with that. It's very exciting.

And finally, an acronym.

You can use this in listening, in communication.

If you're in any one of those roles --

and I think that probably is everybody who's listening to this talk --

the acronym is RASA,

which is the Sanskrit word for "juice" or "essence."

And RASA stands for "Receive," which means pay attention to the person;

"Appreciate," making little noises like "hmm," "oh," "OK";

"Summarize" -- the word "so" is very important in communication;

and "Ask," ask questions afterwards.

Now sound is my passion, it's my life.

I wrote a whole book about it. So I live to listen.

That's too much to ask for most people.

But I believe that every human being needs to listen consciously

in order to live fully --

connected in space and in time to the physical world around us,

connected in understanding to each other,

not to mention spiritually connected,

because every spiritual path I know of has listening and contemplation

at its heart.

That's why we need to teach listening in our schools as a skill.

Why is it not taught? It's crazy.

And if we can teach listening in our schools,

we can take our listening off that slippery slope

to that dangerous, scary world that I talked about,

and move it to a place where everybody is consciously listening all the time,

or at least capable of doing it.

Now, I don't know how to do that,

but this is TED,

and I think the TED community is capable of anything.

So I invite you to connect with me, connect with each other,

take this mission out.

And let's get listening taught in schools,

and transform the world in one generation

to a conscious, listening world -- a world of connection,

a world of understanding

and a world of peace.

Thank you for listening to me today.

(Applause)

English