AQUENT
GYMNASIUM

RESPONSIVE WEB DESIGN:
BUILD A PORTFOLIO FOR ALL DEVICES

LESSON 5

ABOUT THIS DOCUMENT

This handout is an edited transcript of the Responsive Web Design lecture videos. There's nothing in this handout that isn't also in the videos, and vice versa. Some students work better with written material than by watching videos alone, so we're offering this handout to you as an optional, helpful resource.

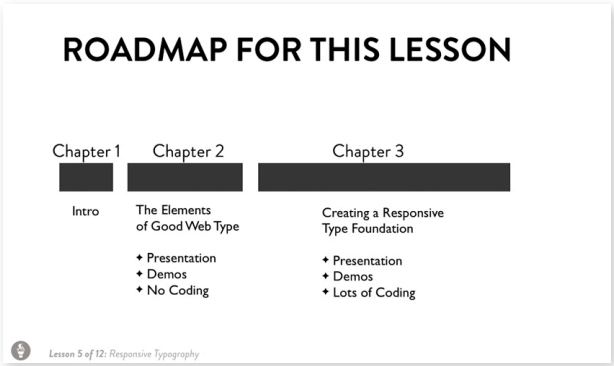
Some elements of the instruction, like live coding, can't be recreated in a document like this one. We encourage you to use this handout alongside the videos, rather than as a replacement of them.

CHAPTER 1: RESPONSIVE TYPOGRAPHY OVERVIEW

This is Responsive Web Design, an online course developed by Aquent. Responsive Web Design, or Promote Yourself Responsively: Build A Portfolio For All Devices. This is lesson 5, Responsive Typography.

As always, there'll be an assignment and a brief quiz at the end of this lesson. At any given point, you'll want to use the pause button in order to stop, reflect, take a look at the code, work on your own code, and finally, if you have any questions be sure to hit the Forum. There we have TAs and teachers waiting to answer your questions, and your most important asset will be your other classmates.

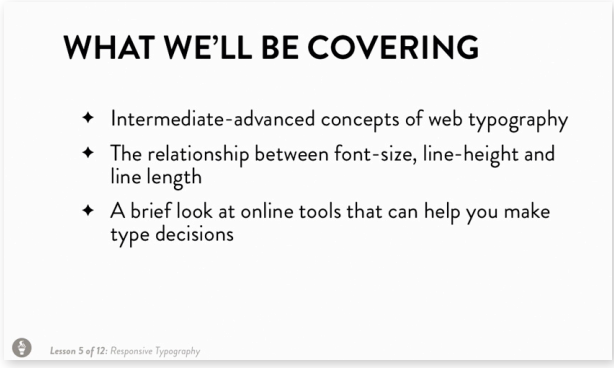
Let's just talk a little bit about the roadmap for this lesson. We're currently here on chapter 1 in the Intro. Chapter 2, we'll be talking about the Elements of Good Web Type, and there you'll be seeing a presentation, some short demos, but there will be no coding. Chapter 3 is Creating a Responsive Type Foundation. There you'll also have presentations, demos and lots of coding.



This is Responsive Web Design.

CHAPTER 2: ELEMENTS OF GOOD WEB TYPE

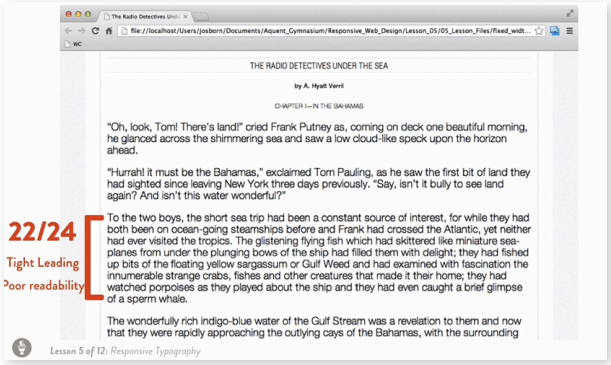
So, to begin this lesson, let's talk about the elements of good Web type. We'll be covering specifically intermediate to advanced concepts of Web typography. So we are assuming you know some basics, such as how to set font-size and line-height and so forth, although we'll be covering some of those aspects. Additionally, we'll be taking a look at the relationship between font-size, line-height, and line-length. And we'll also be taking a brief look at online tools that can help you make type decisions.



The overall objective is to understand the challenges of responsive typography as well as create a strong typographical foundation for your portfolio. So, type is extremely important on the Web. Let's get started. We'll talk about these components: font-size, line-height-- also known as leading-- and line length.

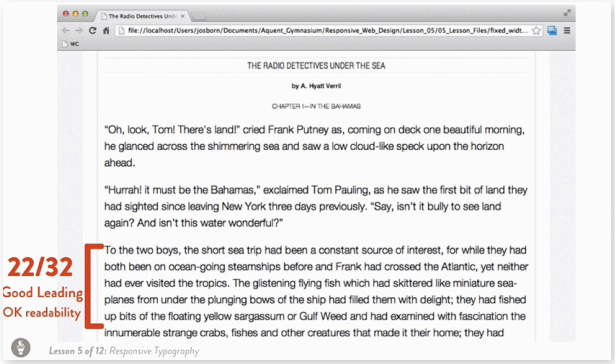
So, let's take a look at some of the issues that we typically face with what I would call traditional typography. So here we have a number of paragraphs that have been set, and just from a readability perspective, we can see that there's very tight leading here. And in fact, tight leading typically leads to poor readability.

So the numbers 22 and 24 here, 22 refers to the font-size in pixels, and 24 refers to the line-height in pixels. And this is a traditional convention that we use. We can go ahead and increase the line-height here. So at this point, the line-height is 32. And we have much better leading. There's more space between those lines. However, the readability is still not great. It's better than it was before, but there's still some other things that we could do.



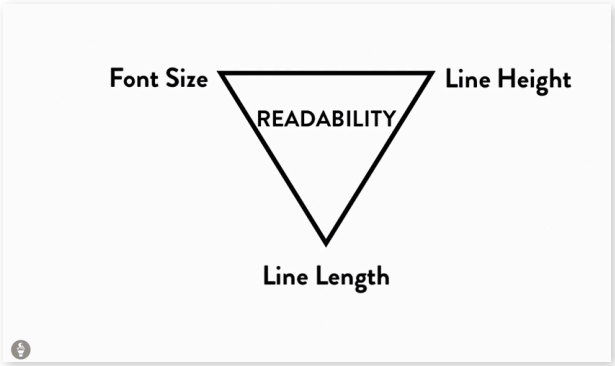
Oftentimes, designers feel like they're playing around with these two concepts of font-size and line-height. And it's a little bit like a mad scientist, where you're trying one font-size and then trying a different line-height, and you're just hoping for a good result.

Now oftentimes, this can be solved if we consider the third element, which is not always obvious, and that is line length. So, font-size, line-height, and line -engths are all connected, and ultimately, they determine whether or not something is readable or not. And they're all connected, as well. So if you change font-size, you may have to go in and change line-height or line length.

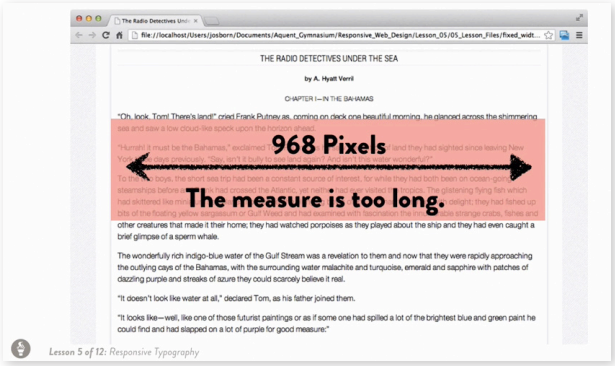


Just a few notes on line length. We also refer to this as line width, sometimes, or content width or traditionally, this is called measure. There is no CSS property for line length. So here we have a rule for the paragraph-- it's 420 pixels wide-- and that would determined the line length in this case.

So, let's examine how line length could affect readability given the fact that we're working with font-size and line-height also. So, here we have a content box which is 968 pixels wide, and we could say that the measure or the line width is too long. Now, when line length is too long, the reader's eye has a hard time focusing on the text. It makes it difficult to get an idea of where the line starts and ends, and it can be difficult to continue from the correct line in large blocks of text.

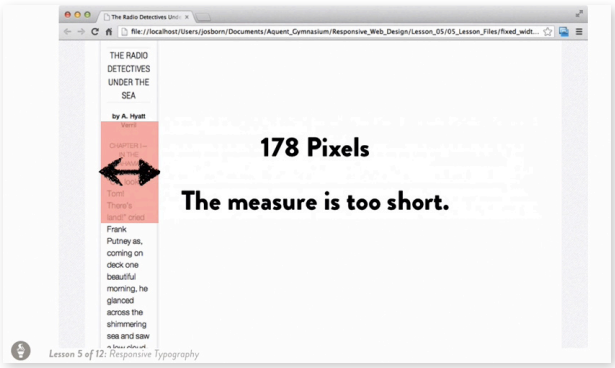


Here, I'm picking a particularly absurd example of 178 pixels, and it's clear that the measure, or the line width, is too short. But still, when a line is too short, the eye has to travel back and forth, and this breaks the reader's rhythm. Too-short lines tend to stress people, making them begin on the next line before they finish the current one, and then they potentially skip important words.

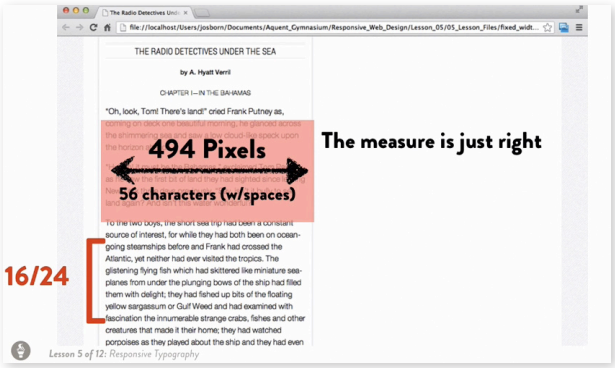


Somewhere in between here, we have a "just-right" measure. In fact, the pixel count here is a little bit of a red herring. Instead of focusing on pixels, we should focus on characters, so this particular width at this font-size and leading has 56 characters. A well-accepted measure for body text is between 50 and 60 characters per line. I'm just going to go up a little bit. You could go up as much as 75 characters per line.

This concept has been relatively accepted over the years. One of the gentlemen who introduced this concept was a well-known typographer called Emil Ruder. This is his book here, "Typography", and you should buy it if you can. Although, you may need to save your paycheck. It's very difficult to find.



Now, what I'd like to show you here is a tool that I use that helps you determine how many characters per line any given page of text has. So, this is a bookmarklet that you can install fairly easily, and I'll show you the URL in just a second. But the way it works is you go to a page, you highlight some text as I'm doing here, you go up to your bookmarklet, click it, and you get the result-- in this case, 131 characters. Let's switch over to our ideal content width. Let's select this line, click our bookmark, and we can see 56 characters.



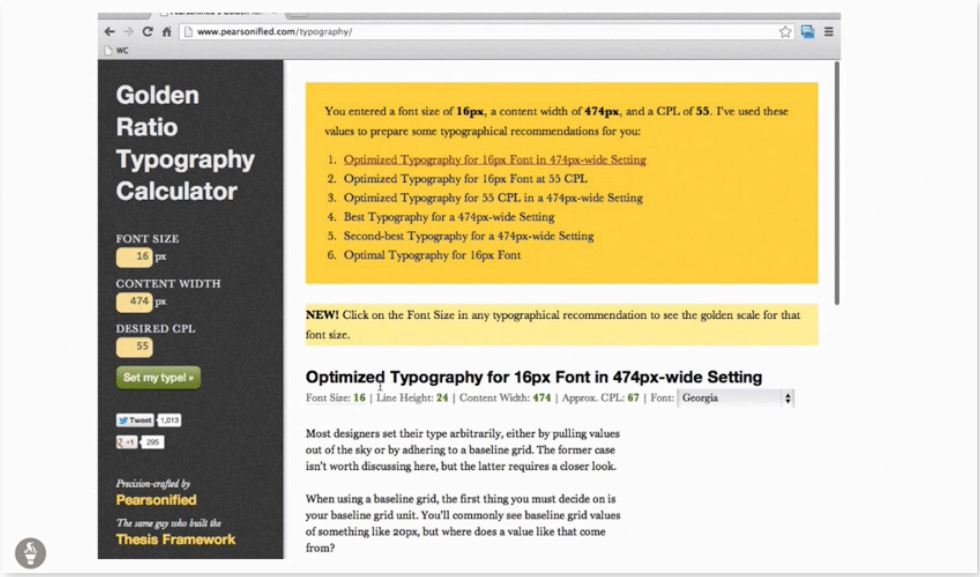
You could also pay attention to words, so some people like to focus on the number of words per line versus the number of characters. Either one is fine. Here's the URL where you can install this bookmark. It's extremely easy to do, and I recommend doing it.

OK, so this does sound a little theoretical, I agree. And all these numbers-- line-height, leading, and line length-- sometimes can get a little confusing. So here's a tool that you can use to help you figure out some ideal numbers. It's called the golden ratio typography calculator.

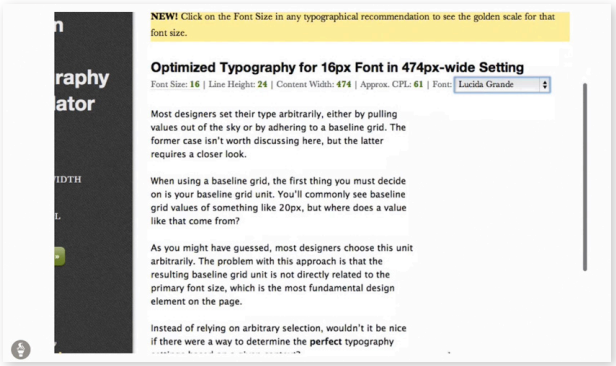
And the way it works is-- well, why don't I just show you? I'm going to go into this page here, and I'm going to enter in a font-size-- in this case, 16 pixels. I'm then going to add content width, 474 pixels. Now if you want, you can put in a desired CPL-- so characters per line. You don't need to do this.

But in the end, you click Set my type, and this is the result. You get five or six different options here. I'm just going to run through a few of them.

The first thing I want to point out is you do need to make sure that you're choosing the font that you're using. This will make a difference. So, just make sure you're choosing a representative font.



And then you can see what's happening here. We have the content width, 474 pixels. And it tells us that a font size of 16 and a line height of 24 would be something that works. I think you'd agree that this looks pretty good.



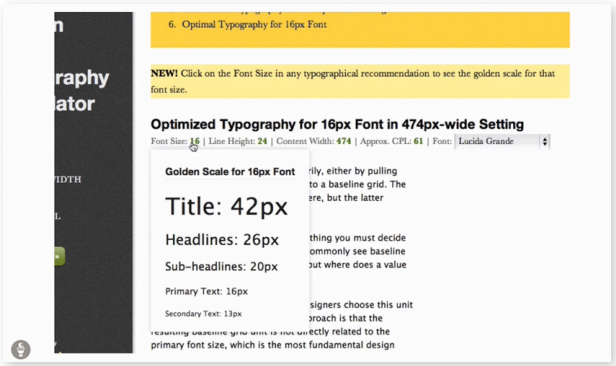
Now, what's nice about this is there are a few other options that you can explore. I'll click on link number 2 here. And of course, I need to change my font to the same font, Lucida. And what this will do is this will change your content width. So, you can see here that the content width is 388. So, this is for designers who don't mind changing the width of their columns, for example.

And there's a few other options here, as well. So optimize typography for 55 characters per line in a 474-pixel-wide setting. And again, in this case, you can see that the font-size and the line-height have changed. So again, this is for designers who don't mind playing around with different font-sizes and line-heights, depending on the project.

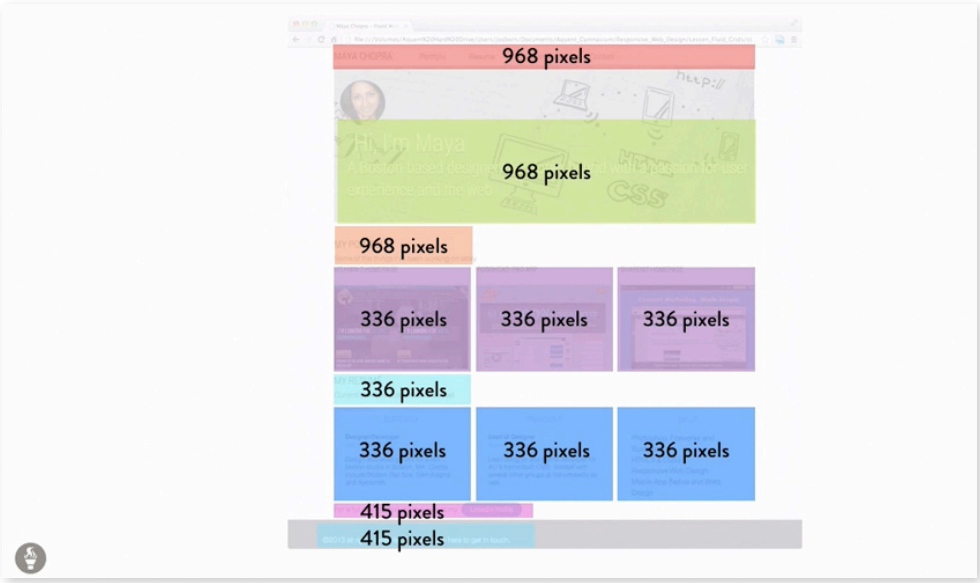
There is a little hidden bonus, and this is one thing I do like about this. If you click on the font-size, you can see that there are some suggested pixel sizes for your font. So in other words, if you're choosing 16, you can see that subheadlines will work at 20, headlines will work at 26, and the title will work at 42.

This is actually using something called the golden ratio or the golden scale. So this is a concept that many designers are familiar with. I'm not going to get into the details of the golden ratio here, but you can definitely explore it on this site.

In the end, fixed-width layout makes controlling type easier. For many years, designers have stuck with fixed-width layouts. And the reason for this is, well, honestly, setting type is much, much easier when things don't move. If I'm creating a 960-pixel layout, I can create a number of different boxes here, and they have set widths. And I can set my line-height and my font-size and my line width extremely reliably.



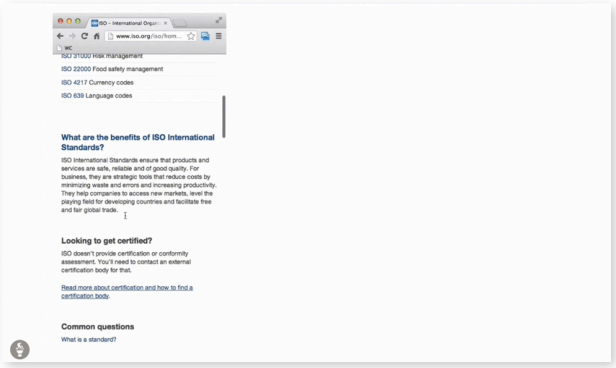
On the flipside, fluid-width layout-- and remember, fluid-width layouts have been around for years-- makes controlling type harder. Just as an example, I've got that same text, and this might work well as a single column, but of course, there's no container here. There's no layout.

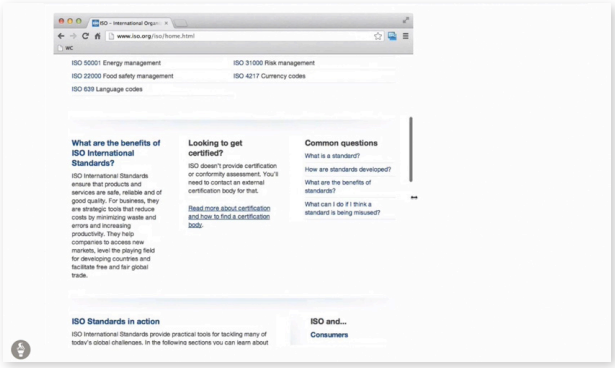


So, the farther I expand the width here, the line length gets increasingly longer and longer. That's going to make it more and more difficult to read, and this opens up a whole can of worms.

So, now I'm going to show you an example of some of the responsive design challenges. This is iso.org. What I'd like you to do is to focus on this paragraph here underneath the heading, What are the Benefits of ISO International Standards?

So, here's our mobile view here, and it seems to work pretty well at this width. But let's go ahead and expand it. As we begin to expand, of course, the line width gets longer. And then at a certain point, it's going to flip over into a more narrow column.





As we keep going, it flips again, and of course, the grid is also changing. But we can see here that the designers had to address some of the challenges with the type. And they did it pretty well.

Remember that media queries have not been around for that long. When media queries first came out, a lot of the weight was on the novelty of the media query. That meant that aspects such as font-size, leading, and line length really didn't get full attention.

What's happening now, of course, is that we're seeking a balance. We're seeking a balance where these properties are balanced with the media queries: where we do get good readability across all our different devices? Of course, that's one of the things that we're going to be doing in the next section.

Just a little bit of warning-- good responsive type requires discipline. It's not something that you want to dash off. As you're about to see, there is some math involved. There's a little bit of careful planning. But it all pays off in the end, and we'll take a look at that next.

This is Responsive Web Design.

CHAPTER 3: CREATE A RESPONSIVE TYPE FOUNDATION

In this chapter, you'll learn how to create a responsive type foundation. And specifically, you'll be learning why the em value is best for responsive typography. You'll also be creating a baseline scale to create vertical rhythm for your body copy and other text. And finally, you'll be updating the media query styles for your three-column desktop layout.

WHAT WE'LL BE COVERING

- Why the em value is best for responsive typography
- Creating a baseline scale to create vertical rhythm for your body copy and other text
- Updating the media query styles for our three column desktop layout

In the last section, we talked a little bit about the challenge of setting type for the Web. Let's explore that just a little bit more. Keep in mind that as a designer on the Web, you give up an awful lot of control as soon as you publish your document. Users can go into the View menu on Firefox-- or many other browsers, for that matter-- and change your layout.

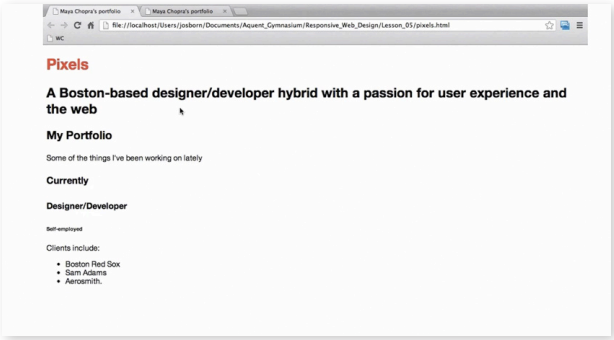
How? Simply by zooming the page or zooming just the text. And this is the effect of breaking your layout, or at least modifying it. In this case, we can see that by increasing the text size or the zoom size several times, this has radically changed our layout.

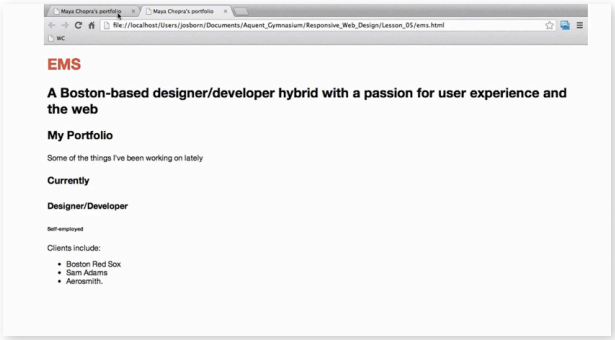
So, the main lesson here is that you need to stop for a second and let go of the control. Have your Zen moment. Realize that you're publishing documents to the Web, and you have no control over how they're going to be used.

And that's OK. That's the nature of the Web. It always has been the nature of the Web. It's not print.

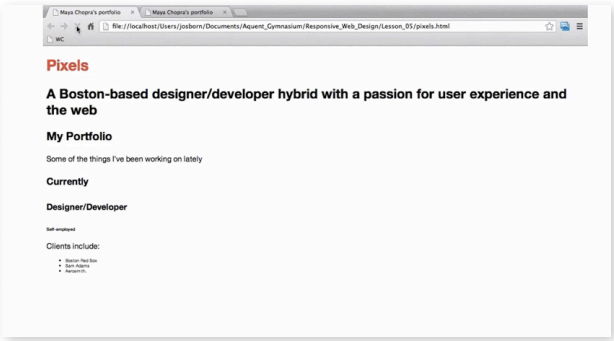
So, having said that, let's move on and talk about ems. Because ems are really ideal for this state of mind, for responsive design. Em units are relative to the font size of the parent element that you're styling.

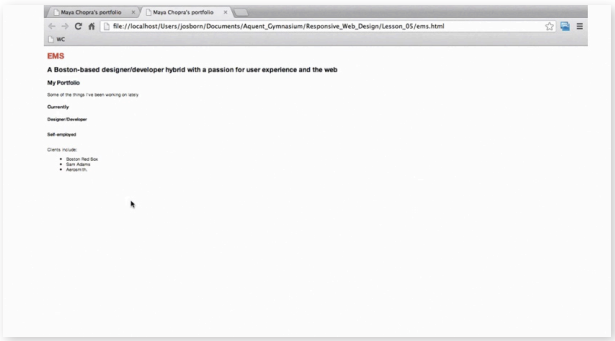
Let's make this a little bit more explicit. I'll give you a sample. We have a document called ems.html-- we can see the red EMS at the top-- and a document called pixels.html. Both of these look exactly the same.





We're to go into the pixels.html document. We're going to set the font size to 50%, and we're going to do the same thing for the ems. We're going to set the font size for 50%. Let's save those documents. Let's go back to our browser and reload.





And what happens with the pixel document? Well, not a whole lot. We can see that that text changes slightly.

However, when we do the same thing for the em base document, everything changes. It all scales by 50%. By changing the font size of the body, we have a linked relationship to all of the other text elements on the page, so they can all change in unison with one single command.

And we could do the same thing if we changed the font size to 200%. You can probably guess what's going to happen here. Everything increases by double. This is why we like ems. We get this responsiveness right off the bat.

If you needed more convincing, Internet Explorer's "text size" tool does not resize any text that was set in pixels. This is particularly true of older versions of IE, but it's still an issue. And if you needed even more convincing, pixels represent a world where fixed layouts were the norm.

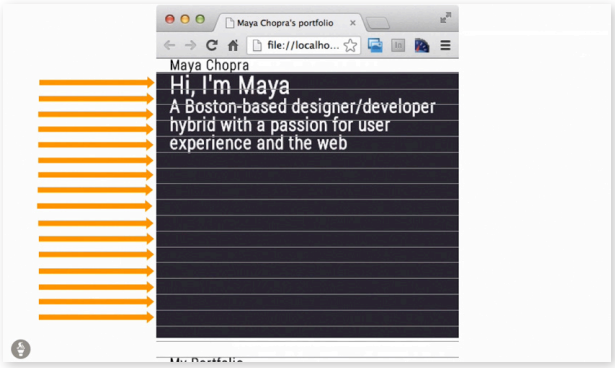
So, for the last 10-plus years, we've been living in this pixel world. Taking Photoshop mock-ups and converting them to Web pages had something to do with this. But for the most part, we no longer live in such a fixed-layout world.

We're living more responsively. We have different devices, and pixels don't cut it as much as they used to. For more details on all of this, I recommend this article by Ethan Marcotte, "Type study: Sizing the legible letter." You can find it on the Typekit blog, and we'll provide the URL in the classroom.

So, having covered that, let's talk about our main goal. The main goal is to add some vertical rhythm to this page, and we need to figure out what that means. So, vertical rhythm is essentially finding the ideal combination of font-size and line-height for all the text on your page. A good vertical rhythm helps engage and guide the reader's eye down the page, makes it more readable, and it sets the stage for more complex grid layout to come.

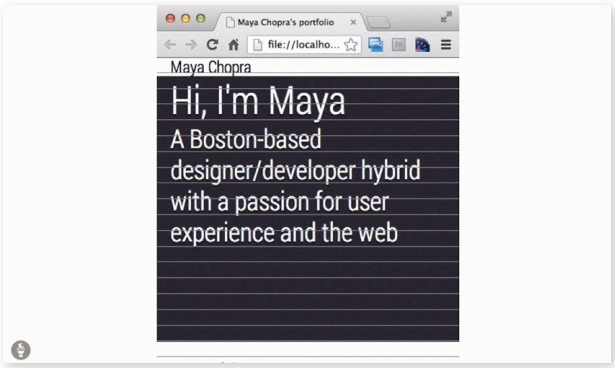
Now speaking of grids, one way to illustrate this vertical rhythm is through something called a baseline grid. Now, this is what a baseline grid looks like on our current page. The distance between each one of these lines represents the combination of your font size plus your line height.

Baseline grids have been around for a long time in graphic design. Josef Muller-Brockman's book, "Grid Systems", was published in 1961, and this book offers a conceptual as well as practical framework for creating page layout. Baseline grids are actually built into modern page layout programs such as InDesign. Here, it's completely optional to use, but actually really easy. You can specify baseline grid, turn on the guides to see it, force your text to align to the grid, and then turn off the grid, and this is what you get. So, that's the world of print. Let's see how this translates to Web.



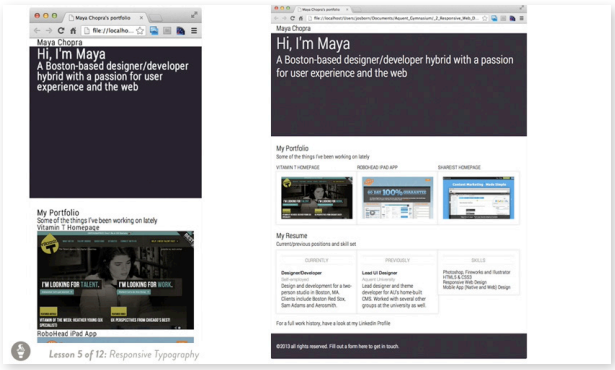
Here's your lesson file with a baseline grid, and you can see that the text is sometimes above the line, sometimes it's below it, sometimes it intersects it. So, there's no intent going on here. The text is just sitting wherever it falls. And therefore, we'd say this page actually has very poor vertical rhythm.

So, let's take a look at the page after some typographic work has been done to it. This is our end goal. When a page has good vertical rhythm, the text lines up nicely, and so do the headings. And we've got great rhythm here.



Now obviously, the grid is just a visual aid. You take that away, and the invisible structure remains. And on both a conscious and unconscious level, the reader's eye appreciates that structure, and it results in a more readable and satisfying page.

So, you have two challenges in this lesson. Number one, you've got to learn the basics of creating this vertical rhythm for a single column, like we have here, our smartphone view. And the second challenge is making sure that the same rhythm works responsively. In other words, how is the text going to make the translation to larger screens and, eventually, multi-column layout?



So, in a well-known and much-discussed blog post from 2006, Oliver Reichenstein-- hope I'm pronouncing his name correctly-- who is the founder of Information Architects, a well-regarded design agency, he proposed that Web design

is 95% typography. Whether you agree with that exact number isn't super important. It does drive home the point that it's worth it to take this time and to be rigorous and disciplined about type.

Which leads me to you a warning about this lesson. So, this exercise you're about to follow is probably the most advanced one in the course. And by advanced, I simply mean there's some concepts and even mathematical calculations that are going to require you to be pretty focused at the task at hand. Now, the rest of the course isn't going to be like this, but if there was one exercise where I would ask for your complete and full attention, it's going to be this one.

So, there you go. Mini-lecture over. You've been warned. Let's just start it off by taking a step back and discussing what we know versus what we don't know about the text on this page.

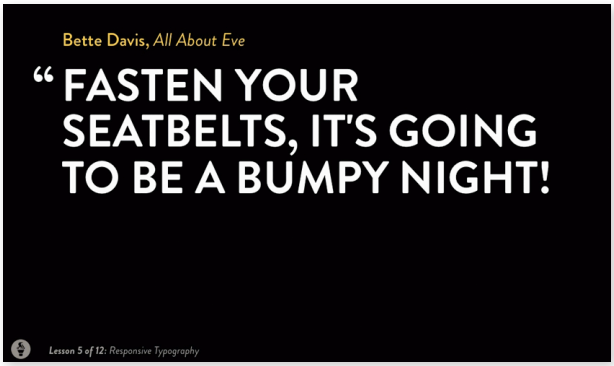
So, we know that the Web font, "Roboto Condensed Light," is being used as the primary typeface for this page. It's currently the only rule on the page for text. It's being determined in the body. No other font-sizes have been set for any other text on the page.

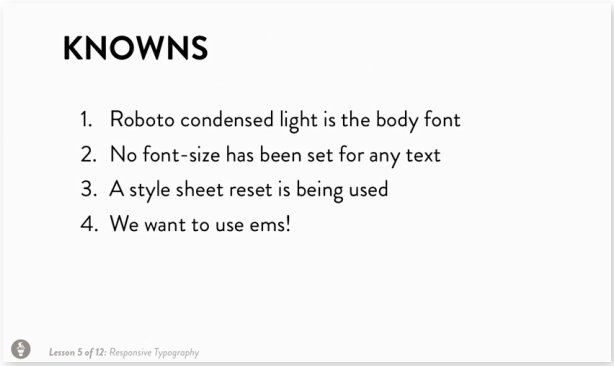
We also know that a stylesheet reset is being used. And this affects all the text by zeroing out the margins. And if you're not on top of those last two concepts or understand what I'm talking about, they were covered in Lesson 2. And if you're shaky on that, I really recommend going back and reviewing those chapters first.

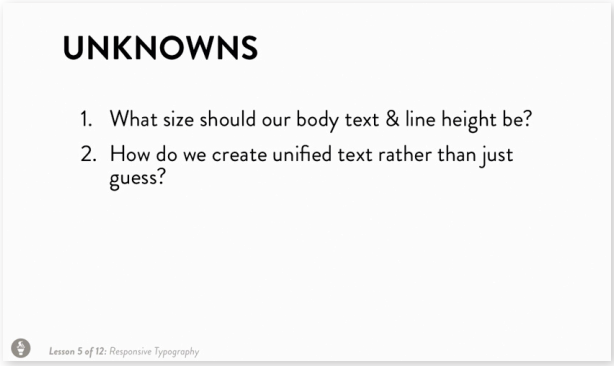
We also know that we want to use em units, and we just discussed the reasons why. Using em units is at the core of responsive typography, and we want to get there sooner rather than later.

Now, let's talk about the unknowns. We don't know what size our default body text and line-height should be. Same thing for our headings and other components, like lists. But the biggest unknown is what combination of settings will work so everything harmonizes and the text works across all screen sizes.

Now, there's one more thing that's unknown to you, but something I know, and that's between the last lesson and this one, I took the time to add some new code into the lesson files in order to save some time. So, let's take a look at this code together. It includes some properties that are really important to understand.







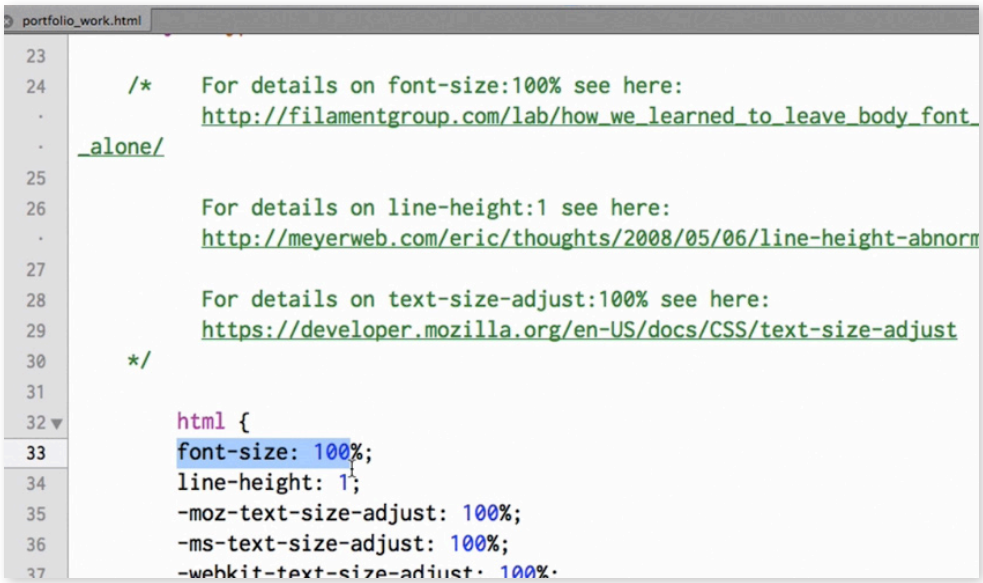
So, first things first. Open the portfolio_start.html document. Save it as portfolio_work.html to make a copy. And now take a look at your style section. I've put some things in here for you.

Inside the style for the HTML element, the font-size is set to 100% and the line-height is set to 1. Now, if you're interested in the details of what's going on here, you can see that in the comments, there's a link to the article that explains this in more depth. But I'll just summarize it here.

We're starting with the body text set to 100%. Furthermore, all browsers have basically agreed that 100% equals 16 pixels. So we're just making sure this is truly the case and there's no wiggle room.

Same thing for the line-height. We've set here to a value of 1. Both of these steps are going to help us when we make the transition to ems. It's going to help keep our type consistent across browsers and make everything a little bit more consistent.

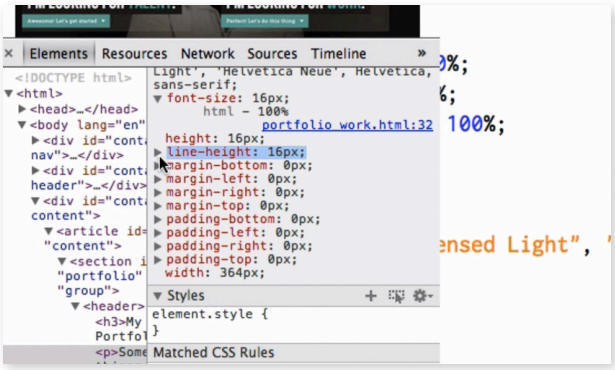
Now before we take a look at this next section of code, anything going off in your head, any bells? Are you wondering why this line-height has no value, why it's just 1? So, it's not 1 em or 1 pixel. It's just 1.



So that's OK, because line-height is actually a special CSS property. It uses something called a unitless value. In other words, it's just a number. And to make everything crystal clear, I'm going to go into our browser and we're going to take a look at how the browser interprets the text on the page.

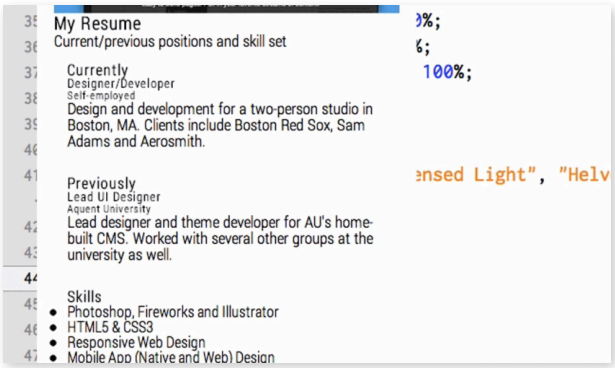
So, if we toggle over to our browser here-- and I'm using Google Chrome-- we're going to open up our developer tools, and we're going to go into this computed style section. So, this is essentially the values that the browser is putting for our text. And the value for a paragraph is 16 pixels, and the line is also 16. And both of those relate back to those settings we just took a look at.

Now, while I'm in here, I'm also going to check out the size for Heading 1. So look at that. Turns out the browser wants a Heading 1 to be 32 pixels and the line height to be 32. Same thing for the Heading 2.



So, that's cool. We're going to head back to our code and realize that both of these properties are essentially just an extension of resetting our styles, and then we're going to build them back up in a consistent way. And you might also be wondering what this text-size-adjust code is. So, out of

the interest of time, I'm just going to summarize, and there's a link up there if you want more detail.



Basically, this property only affects text on smartphones. Browsers such as the iPhone's Mobile Safari will sometimes automatically scale small text in an attempt to be nice to users. However, this behavior can actually get in the way of someone who wants to control the text in a mobile-first design. That happens to be us! So this code is simply disabling that auto-scale feature by saying 100%. And we also have three lines of code here for different browsers.

OK, so we've covered a lot of things. We haven't even touched our code yet. Let's start by setting the default font-size and line-height for our paragraphs. We're going to use that as a starting point, get it right, and we can take those lessons and apply them to other text.

Let's go to our browser for a second, and let's find something that's a representative paragraph. In this case, we'll use this section here, under "Currently." Our goal is to find a sweet spot for the text. We're using this condensed light font. And at 16 pixels, the letters seem a little weak to me on the screen.

So, I'm going to go ahead and go back to the code and bump that up a notch to 17 pixels. Let's try that. Let's refresh the page, and this actually works better. So, it's a little hard to see the difference on the video, most likely, but on your own page, you should see that it works better. The letters are a little more crisp, and the slightly larger size will translate better for a mobile screen. So, that's good. We're going to keep it there.

Let's turn to line-height, because clearly there's not enough space here. In the real world, I would go ahead and experiment by slowly nudging these values up until I got what I wanted. But I'm going to go ahead and out of the interest of time, let's just put in 20 pixels. Double-check that it's OK, and it is, so we're good.

So, this is our foundation, but it's still in pixels. We need to convert them to ems. So how do we do that? We got a handy formula. We're going to take our target value for the font-size, 17, and we're going to divide it by the default font-size of the page, which is 16.

So, 17 divided by 16 is ugly. OK. So, we've got to get our calculator out. 17 divided by 16 is 1.063 em. Wow, OK. So, let's put that in there.

Now, let's do the line-height. So, we'll go ahead and we'll copy this comment here, and we'll paste it. And what's the math? 20 divided by 16 equals 1.25. 1.25. Let's make sure we've replaced both the real values here, so that we don't have any problems.

Let's look at some simple ways to put this into action. We're going to focus on the top part of the page here. So this text here is a Heading 3. Then we have Heading 1, then a Heading 2. We're going to set the font-size for all three of these in just a moment, but let's go back to our code and do one thing first.

```
        -moz-text-size-adjust: 100%;
        -ms-text-size-adjust: 100%;
        -webkit-text-size-adjust: 100%;
    }

    body {
        font-family: "Roboto Condensed Light", "Helv
sans-serif;
    }
    p {
        font-size:1.063em; /*17px / 16px = 1.063em *
        line-height:1.25;/*20px / 16px = 1.25em */
    }
```

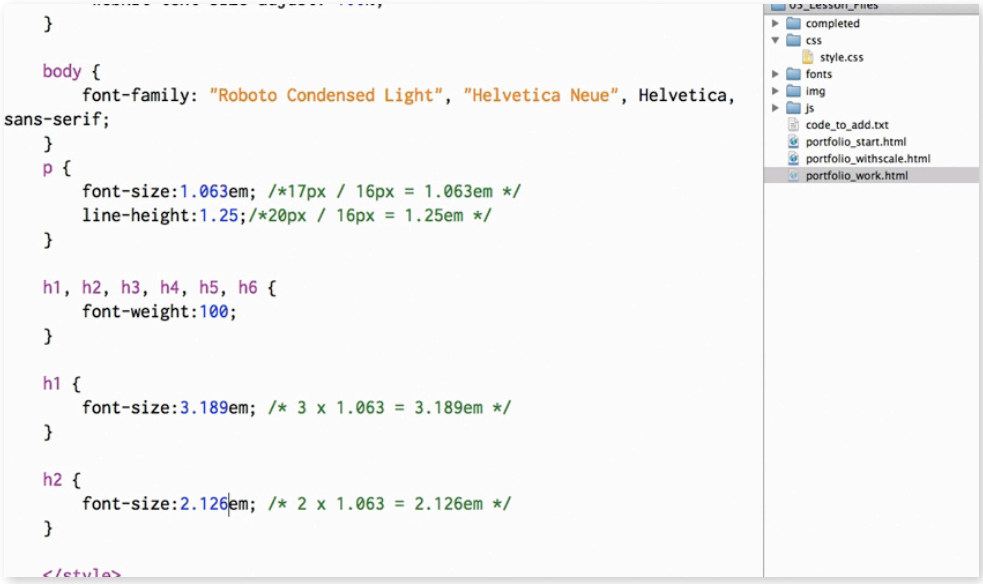
I'm going to say all of the font styles should be a font-weight of 100. So that looks something like this-- h1, h2, h3, h4, h5 and h6, font-weight 100. And what this does is change all

the headings to the lightest weight possible. Later on, we can go back and build up the specific headings if we want to. You can actually see the difference here between the two pages. I'm just toggling back and forth between the prior version and this version.

Let's turn to the font-size for those headings. Here's a really simple rule of thumb. Your Heading 1 should be three times larger than your body text. Your Heading 2 should be about twice as large.

If we do the math, that turns into this. 3 times 1.063 equals 3.189 em. That's for the Heading 1. And then 2 times 1.063 is the Heading 2, and we get 2.126 em.

Now, for the line-height, we're just going to use the same value we had for our paragraph. Let's go ahead and save that, reload it in the browser, and look at that. What we can see here is the beauty of this system. The amount of space here between the lines of our headings is proportionally the same amount of space as our paragraphs. This gives us instant harmony.



Remember, the line-height does not need to include units, as I mentioned before. Just as a side note, if you absolutely have to understand more about the logic of this, I'm going to point you to this article by Eric Meyer, where he goes way deep into the logic. Totally need to warn you about that-- you'll get lost in a lot of details. It's cool stuff, great stuff, but the short answer is line-height is really just a tricky property to control when you use ems. So, when you use no value, it's more reliable.

OK, back to our code. We took care of our Heading 1 and our Heading 2 by using that simple rule of thumb. However, those are the easy ones. If we start looking at the other parts of our page, such as our Heading 3s and 4s, we don't have such clear-cut rules.

What would be really nice is if we did have some sort of a rule of thumb for everything else. And it turns out we can do that. It's just a little trickier.

So how do we do it? Well, let's go back to our paragraph style, and we have these two properties-- font-size value of 1.063 em and line-height value of 1.25. It turns out that we can multiply those two values, and the result is something called a typographic scale. It's just a number.

In this case, 1.063 times 1.25 equals 1.32875, and what we're going to do is we're going to simplify that to a simpler number. We're going to round it down to 1.3. So, you don't know it yet, but that 1.3 number, that's our gold. And to be really explicit, that 1.3 scale is simply a number that reflects a single line of body text, including the line-height.

And what we're going to do is we're going to use that scale as part of an equation, and that's going to determine the spacing of virtually everything on our page-- so our headings, our lists, the amount of space between different sections of the page. We can use it for margins and padding. And I'm going to show you how it works in just a moment, but I actually want you to do me a favor right now.

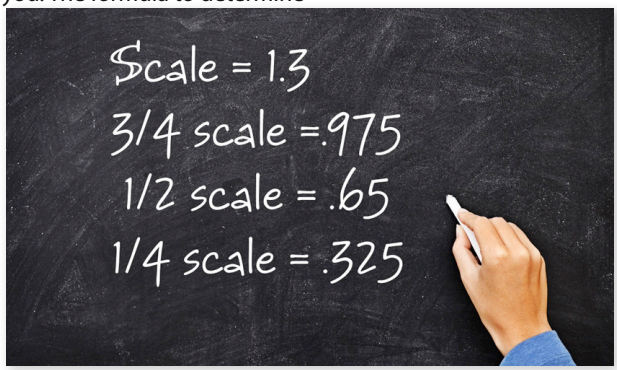
Please take out a piece of paper and pencil, or any writing implement, and I want you to write down these numbers. Scale equals 1.3. 3/4 scale equals 0.975. 1/2 scale equals 0.65. Quarter scale

```
40  body {
41      font-family: "Roboto Condensed Light", "Helvetica Neue", Helvetica,
sans-serif;
42  }
43  p {
44      font-size:1.063em; /*17px / 16px = 1.063em */
45      line-height:1.25;/*1.063 x 1.25 = 1.32875 Round to 1.3 */
46  }
47
48  h1, h2, h3, h4, h5, h6 {
49      font-weight:100;
50  }
51
52  h1 {
53      font-size:3.189em; /* 3 x 1.063 = 3.189em */
54      line-height:1.25;
55  }
56
57  h2 {
58      font-size:2.126em; /* 2 x 1.063 = 2.126em */
59      line-height:1.25;
60  }
61
```

```
style.css
fonts
img
js
code_to_add.txt
portfolio_start.html
portfolio_withscale.htm
portfolio_work.html
```

equals 0.325. I asked you to write those down because those are essentially your building blocks. Whenever you need to increase or decrease space on your page, you can use some combination of these rather than just random numbers, making it up.

So, to make this more clear, let's actually use this. Let's turn back to our paragraph and use it here to get a more accurate line-height. So, here's another formula for you. The formula to determine line-height is as follows. You take your scale, which is 1.3, and you divide it by the font-size. So this gives us a value of 1.223 instead of 1.25. That doesn't seem like a huge difference, but it's enough, and we'll see how this applies later on.

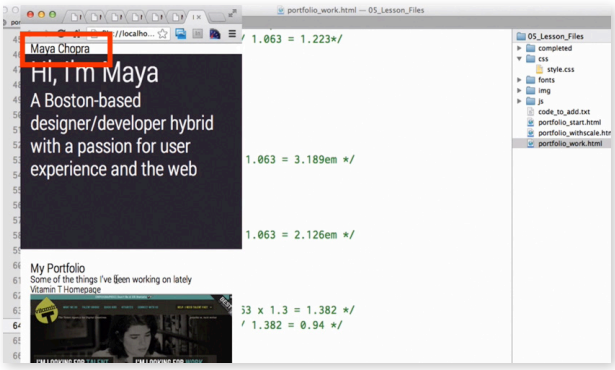


In fact, let's go ahead and see how this 1.3 scale can be applied to an element we haven't even touched yet. Let's go back to our browser, back to the top, and how big do we want Maya's name to be at the top of the page? That's a Heading 3, and we want to get the font size right first, and then we can determine the line height.

So, we know that headings should be bigger than paragraphs, right? But how much bigger? Let's use our scale. 1.063 times 1.3, this equals 1.382. Let's add that, save it, test it. Nice. That looks good. I'm cool with that.

And what about the line-height for this Heading 3? How do we do that? Now, you remember what we just did with the paragraph? We used a formula. We took our scale, 1.3, and we divided it by the font-size, 1.382. This equals 0.94.

Let's punch that in for line-height. Not so good. Looking a little squished. This means we need more room. But again, how much more?



What we're going to do is we're going to increase our scale. Remember your notes? Look back at your notes.

So, to increase our scale, we can take 1.3 and add one half to it. So 1.3 plus 0.65 equals 1.95. Now this does mean we have to do our calculation again. 1.95 divided by our font-size of 1.382 gives us 1.41.

OK, so I want to see what this looks like with just a little less space. So what component could we use to reduce the line height even further? Let's try 1.3 plus one quarter. Look back at your notes, right? A quarter is 0.325.

So, we add those two up, and we get 1.625, and we go back to our formula one more time. Divide 1.625 by 1.382. Gives us a crazy decimal, and we round it down to 1.176. That. That's our new line height for the Heading 3s on our page. Let's take a look at it. I like it.

OK, so I know that's some crazy math. And again, we really only have to do this once to get it right. But let's take a step back for a second and visually reinforce what we've just done.

Remember the beginning of the lesson, where we saw that lack of rhythm? Well, what you've done is just add rhythm to those three elements up top, and I'm going to show you this visual overlay, so you can see it. For our Heading 3, we can see Maya's name is sitting nicely on the baseline. And for Heading 2 and Heading 3, it looks like they're around about the quarter baseline.

```
        line-height:1.223;
    }

    h3 {
        font-size:1.382em; /* 1.063 x 1.3 = 1.382 */
        line-height:1.41; /* 1.625 / 1.382 = 1.176 */
    }


    #content article {
        padding-bottom:1.3em;
    }
    </style>
    </head>

    <body lang="en">
        <div id="container-nav">
            <section id="nav">
```

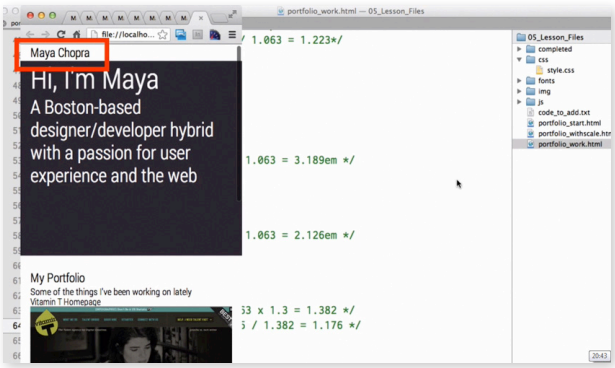
The important part, though, is that they're consistent. See that consistent space between each line? That's rhythm. That is rhythm for you. And you can see that in contrast, because if we look down the rest of the page, everything falls apart. There's no rhyme or even reason, for that matter.

OK. So let's just take a little quick break from that. In the next section, we're going to finish this up and do the second half of this page.

CHAPTER 4: RESPONSIVE TYPOGRAPHY - WRAP-UP

OK, so we're back now. And we just worked on the vertical rhythm of the top of the page. We laid the foundation. How else can we use this new and shiny scale, this number of 1.3?

Well, how about positioning for elements? We have these three examples here in the portfolio section, and they're pretty cramped. What we need to do is add some space between them and let them breathe. In our HTML, we can see that these are three article elements inside of the ID named "content".

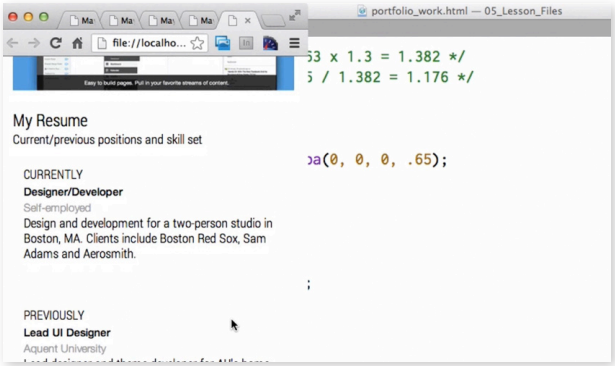


So, we're going to take that rule-- #content article-- and we're typically going to use margins or padding, so let's use padding. And should we add any number? Think you know the answer to that. The answer's no. We're going to use our scale, and let's go for the bottom padding first. And we're going to put in the full scale-- 1.3 em.

Let's go ahead and save it, look at it in the browser. That almost does the trick, but we need some more space on the top. From the looks of it, we only need a little. So let's go back, and let's try a half scale of .65em. Let's save that, preview it-- looks good. Let's leave it.

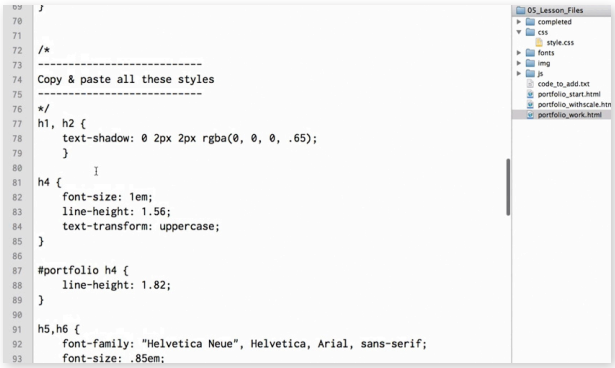
OK, next up, you're going to cheat. In the first chapter, I went through pretty slowly and thoroughly with the code, but now we're going to speed things up. What I want you to do is to go into the code_to_add.txt file. And you see all those styles there? Those are various styles I've already configured for you in the appropriate scale. Isn't that nice of me?

So, what you're going to need to do is simply copy everything from here all the way down to here. Copy it. Go back to your internal stylesheet and drop it in. Save it. Test it in your browser and voila-- instant rhythm. Isn't that cool? So, a lot of other styles in there as well.



We could talk about what's happening with all the styles. But a lot of it is fairly straightforward CSS. I'm going to assume that you know most of the concepts there and can explore it on your own time. I do want to point your attention to this section of the page in the resume section. We're going to dress it up in a second with some more styles.

I'm going to point out the .box h4 rule. You can see here that, again, I've used this three quarter scale for the margin. But take a look at this rule for the box shadow, which, if you haven't used it before, is a great feature when you use it sparingly. However, the values aren't super obvious.



The first value is the position of the horizontal shadow, which, in this case, is actually 0. The second value is the vertical shadow, which is also 0. But the third value is the blur distance, and what we've done here is made it .325em, which should sound familiar. Look down at your notes. That's your quarter scale. And the last value is simply the spread, which we've kept very small and subtle.

The point here is we can use our scale for another element on our page, in this case, this box shadow effect. Let's go ahead and copy that code, throw it in. Save your page, refreshing the browser-- nice shadow.

So, here we are. It's now good news, bad news time. The good news is that, by taking all those shortcuts, we've arrived at a really nice single-column layout. Go ahead and take a second to scroll through everything. We have a really nice proportion for our text, thanks to our scale.

But the bad news is that this is supposed to be a responsive foundation, right? So how does it look when we break out of our single column? Hm, the answer is not bad. Look at that. Maybe this is not bad news.

Our vertical rhythm pretty much holds up across the board, especially when you keep in mind that we're not even doing grid layout. That's the next chapter. That's the next lesson.

But there is a criticism that I want to address. I feel like, in the desktop view, this text size might be a little too small. Maybe it's because we've got three columns going on. Maybe it's because it's condensed text. Whatever it is, I'd like to experiment with scaling up all the text on the page when we hit a certain breakpoint.

So, what I'm going to say is that whenever we get wider than 720 pixels, we should bump up the text a bit. Now more good news for you, I've actually crafted that media query here in our code_to_add.txt file. All you have to do is copy and paste it.

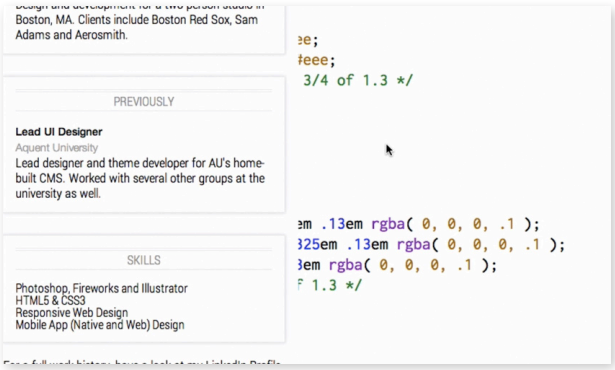
```
    border-top: 4px double #eee;
    border-bottom: 1px solid #eee;
    margin-bottom: .975em; /* 3/4 of 1.3 */
    text-align: center;
}

figure,
.box {
    background: #fff;
    -moz-box-shadow: 0 0 .325em .13em rgba( 0, 0, 0, .1 );
    -webkit-box-shadow: 0 0 .325em .13em rgba( 0, 0, 0, .1 );
    box-shadow: 0 0 .325em .13em rgba( 0, 0, 0, .1 );
    padding: .975em; /* 3/4 of 1.3 */
}

section footer {
    clear: both;
}
```

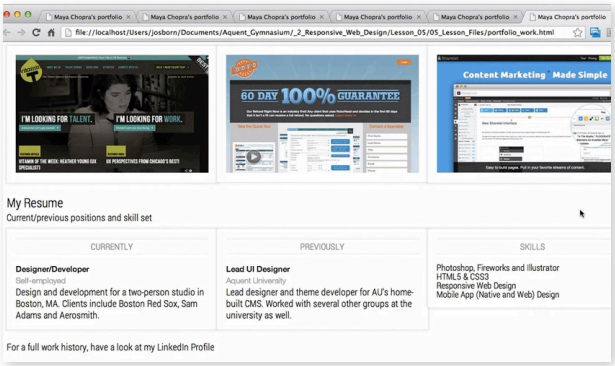
I will explain what it does first, though. We have this media query with a min-width of 45em, which my Gymnasi-tron 3000 translates to any screen greater than 45em or 720 pixels will use this style. But to bump up the text, I want to make sure that it's a consistent scale, so I'm going to bump up the scale from 1.3 to 1.4, But I need to convert that to a number I can use somewhere.

So, the math involved looks like this-- 1.4, our new scale, divided by 1.3, our old scale. The result is 1.07 blah, blah, blah-- one of our favorite crazy numbers. So, what we're going to do is, to turn into something useful, we'll multiply it by 100, and the result is basically 107% plus change.



So, go ahead and copy that. We already have a media query for this in our stylesheet that we added in the last lesson. So, we're going to take that rule for body, copy it, go to our external stylesheet, locate that media query section, and paste it. Lets save our file and check it out in the browser.

So, we changed width of the browser. We can see right there is where our break is. It's definitely a perceptible change, but it's not jarring. And we did all of our hard work in the base stylesheet. But because of the power of ems and because of our scale, we can make all our text larger and still maintain our proportions.



So, looking ahead, we've still got lots to do-- the grid layout for the desktop, little awkward. We've got some work to do there, but that's our next chapter where we talk about fluid grids. meantime, some homework.

As always, there's a short quiz. Assignment number 2-- go back to your code_to_add.txt file, and review all the notes inside that file. They describe the logic behind our scale, so make sure you understand those. If you don't, either watch the video again or go to the Forum, ask some questions. There's also some additional readings and links to videos in the class handouts that could come in useful.

Assignment number three-- add a vertical rhythm to your own portfolio in progress. The first step is to determine the ideal font size and line height, just as we've reviewed. One thing I do want you to pay attention to, remember that different typefaces are going to have different characteristics, such as x-height, cap height. Be sure to pay attention to those. Don't just use my numbers. They may not work on your design.

That's it for now. I'll see you on the Forum, and I'll see you in the next section.