Your first “About Me” project

Music

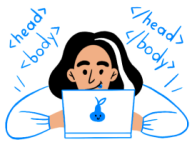
I'm a singer in an all-female power metal band called "Great Vengeance." We rehearse in my garage and play at local shows here in Oregon every couple of months.

Friends

I love hanging out with my girls after rehearsals, sipping on yerba maté, and watching wrestling. I hope we go on a big tour one day.

Family

My partner Sean and I are proud parents of two. Our older one, Lola, is a tween and baby Payton is one and a half. They're a lot of fun, but sometimes they can be quite a handful.



Web Dev

I'm super excited to learn how to program. I've been a full-time mom for a while and I'm ready to get back to work and start a new career.

GreatVengeanceMeg x Practicum

Project structure and starter code

To get started, download the starter kit linked to on the "Your First Independent Project" page. After downloading the archive, you should have a folder named "About Me Starter Code" on your computer. If you open it, you'll see that this folder contains four items:

- A `fonts/` folder containing files with the custom `Permanent Marker` font. You don't need to do anything with this folder. This is where that cool font from the example project is stored. This means that you'll be able to use the same font for your project.
- An `images/` folder for storing pictures that will be used in the project. In order to display images on the four cards of your webpage, you'll first need to add those images to this folder to be able to access them.
- The last two items are the `index.html` and `style.css` files.

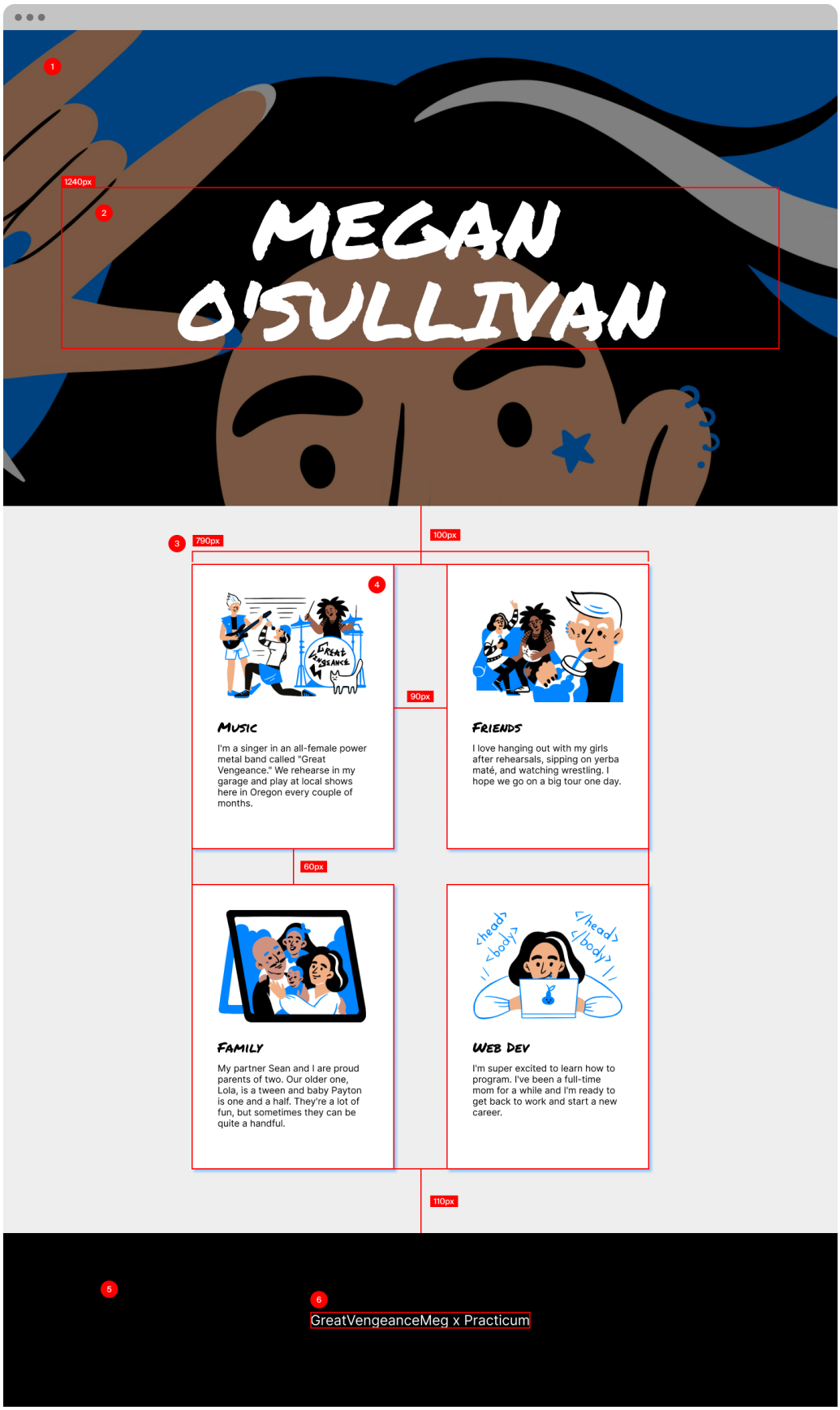
Open the `index.html` and `style.css` in VSCode. You'll notice that each of them already contains some code.

Here's a breakdown of the code that's already been written for you:

- The HTML file already contains the basic structure for your webpage and includes some comments to show you where to write the rest of your code.
- In the CSS file, the custom font has already been loaded for you using the `@font-face` rule. The `font-family` property is used to specify a font name for the project. The `src` property contains links to both file formats.
- Recall that the browser automatically applies styles to some HTML elements. After the `@font-face` rule, we've added a CSS rule that overrides this behavior by resetting the margins, font size, and weight for the `<body>` element, headings, paragraphs, and links.
- We've also applied the `box-sizing: border-box;` declaration to all elements using the `*` selector. Recall that the `[border-box](<https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing#values>)` value means that the values of the `border` and `padding` properties are included in the `width` and `height` values specified for an element.
- Note that in the HTML file, the `<body>` element has the `page` class. In the CSS file, we've create a rule with the `.page` selector, which applies a default font and

background color for the page.

There's more code than this in the starter kit, but we'll explain the rest in more detail in the instructions for each section.


1
1240px
2
MEGAN O'SULLIVAN
3
790px
100px
4
90px
MUSIC
I'm a singer in an all-female power metal band called "Great Vengeance." We rehearse in my garage and play at local shows here in Oregon every couple of months.
FRIENDS
I love hanging out with my girls after rehearsals, sipping on yerba maté, and watching wrestling. I hope we go on a big tour one day.
60px
FAMILY
My partner Sean and I are proud parents of two. Our older one, Lola, is a tween and baby Payton is one and a half. They're a lot of fun, but sometimes they can be quite a handful.
WEB DEV
I'm super excited to learn how to program. I've been a full-time mom for a while and I'm ready to get back to work and start a new career.
110px
5
6
GreatVengeanceMeg x Practicum

Header

(1)

Let's start building the project with the header. If you look in the HTML file, you'll notice that there's already a `<header>` element with the `header` class. If you look in the CSS file, you'll see that we've already created a rule for this class.

Right now, it contains the `height: 100vh;` declaration. This ensures that the height of the webpage will always equal the height of the user's display, no matter what size their screen is.

Now it's time for you to start writing some code. Start off by adding a background image to the header. Here's what you need to do:

- Add a picture of your choice to the `images/` folder.
- Then, use the `background-image` property to specify a path to it. To specify the path to your image, the value of the background-image property should look something like this: `url(./images/name-of-file.jpg)`.

- Use the `background-size` and `background-position` properties to set the image size and position.
- Open the `index.html` file in your browser to make sure it looks good.

You may have noticed that the CSS file contains a rule for the `.overlay` selector. This superimposes a semi-transparent layer on top of the background image using the `background-color` property.

(2)

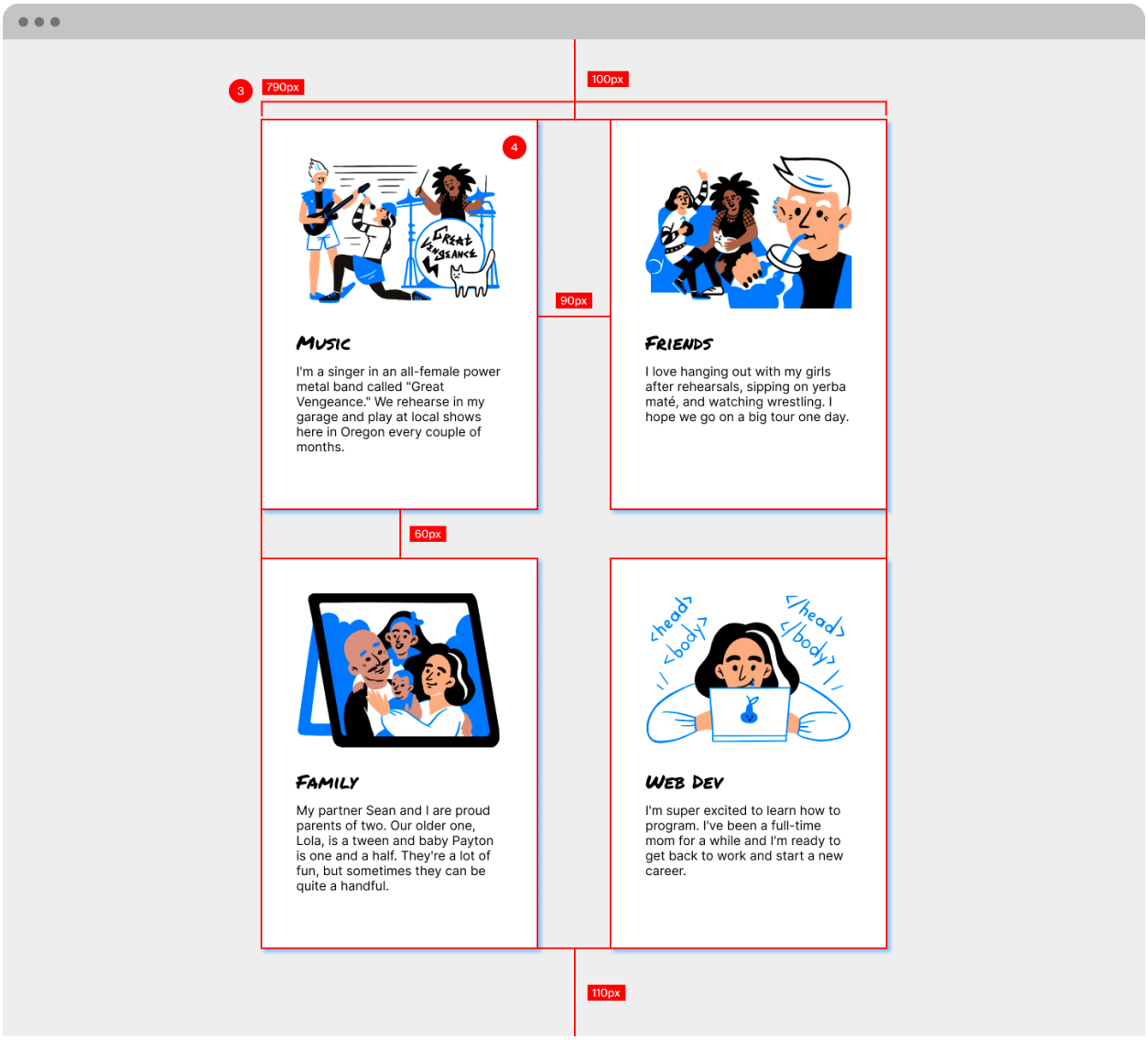
Next, we'll work on the main page heading. Open the `index.html` file and look for the comment that says `<!-- add the page title here -->`. Replace this comment with an `<h1>` element. Give it a descriptive class name and write your name between the opening and closing tags. Refresh the page in the browser. Your name should appear in the top-left corner of the page.

Next, you need to add some styles to your heading. Create a new CSS rule in the `index.css` file using the class name you created as the selector. Add the following declarations to the rule:

- `font-family: "Permanent Marker", cursive;`
- `font-size: 140px;`
- `line-height: 140px;`
- `color: white;`
- `text-transform: uppercase;`
- `max-width: 1240px;` — to limit the width
- `margin: auto;` — this will center the page title both horizontally and vertically. It works for vertical centering in this case because its parent element with the `overlay` class has the magical `display: flex;` declaration applied to it.

To make sure the title text is always aligned in the center of the page, add the `text-align: center;` declaration to the CSS rule for the header. We need this because while the `margin: auto;` declaration applied to the title ensures that the `<h1>` element itself is centered, its actual content, i.e. the text, is aligned to the left of the element's borders by default.

Cards



(3)

After the header, it's time to move on to the content section. This is where you'll build the cards that show interesting facts about yourself. There are four cards that should be centered horizontally and wrapped onto two lines. To make this happen, we've already added three CSS declarations to the rule with the `.content` selector:

`display: flex;` — you're already familiar with this one. The `flex` value allows us to center our elements. It also provides us with a wide range of options for arranging our

elements. You'll learn more about these in the full program, but we'll use two options for this project:

- `flex-wrap: wrap;` — when this declaration is used, items within a container will wrap onto a new line if there isn't enough room for all of them with their specified dimensions. Otherwise, by default, the `flex` value would cause the elements to squeeze onto one line.
- `justify-content: space-between;` — this declaration evenly distributes the items within its container. In our case, the container for the cards will be the `<section>` element with the `content` class.

Before creating the cards add the following declarations to the CSS rule with the `.content` selector to define the dimensions of the container:

- `max-width: 790px;`
- `margin: 100px auto 50px;`

(4)

Let's get back to the HTML file. Note that every card on the design has the same structure. It contains an image, title, and text. Inside the `<section>` element, create a `<div>` element with the `card` class. This will be a parent element for the card content. Inside this `<div>` element, create an `<img>` element with following attributes:

- `class` to add a name of the corresponding selector. Remember that it should be descriptive (for example, `card__image`)
- `src` to specify a path to the image
- `alt` to add an alternative text for the image

Beneath the `<img>` element, add an `<h2>` element for the card title and a `<p>` element for its description. Assign class names to them as well.

Now you have the basic structure for one card. You need four cards, so copy the `<div>` element and paste it three more times below.

Now you can start adding some content to the cards. Add the images you want to the `images/` folder, then link to them via the src attribute of each `<img>` element. Fill the text elements with information about yourself.

Next, let's style our cards. Add a CSS rule with the `.card` selector with the following declarations:

- `width: 350px;`
- `margin-bottom: 60px;`
- `padding: 45px 45px 70px;`
- `background: #ffffff;`
- `box-shadow: 4px 4px 5px rgba(0, 132, 255, 0.4);` — to add a shadow effect

Add a CSS rule for the card title (i.e. the `<h2>` element). Don't forget to specify styles for fonts and the bottom margin:

- `font-family: "Permanent Marker", cursive;`
- `font-size: 24px;`
- `margin-bottom: 10px;`

The CSS rule for the card description should contain just the `font-size` property with a value of `16px`.

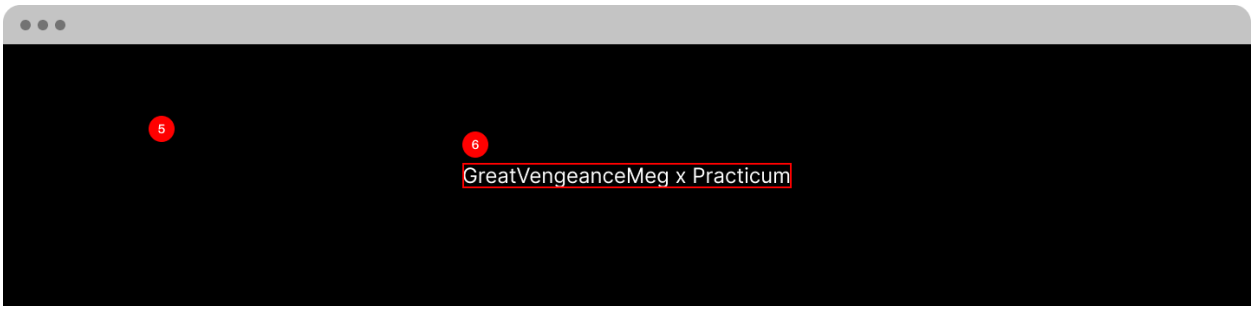
Finally, add a rule for the card image with the following declarations:

- `width: 100%;`
- `margin-bottom: 25px;`

If your images have different sizes and you notice that some of the cards have different heights, you can deal with this by assigning the following declarations to the `.card__image` class rule:

- `height: 200px;` — to add a fixed height
- `object-fit: contain;` — this declaration makes the element fit within the fixed content box, while preserving its aspect ratio

Footer



(5)

Let's move to the footer. One declaration `display: flex;` has already been added to the corresponding CSS rule to center content of the footer. Set the height and background color according to the design:

- `min-height: 300px;`
- `background-color: #000000;`

(6)

Add your signature inside the footer using the `<p>` tag and style it setting the font size to `24px`, the color to `#ffffff`, and center it with the `margin: auto;` declaration.