CSS Basics

CSS, or Cascading Style Sheets, is a way to style our markup (usually HTML). It is what you'd use if you wanted to change the color of some text or create simple animations. That and limitless other styling options are possible with the use of CSS.

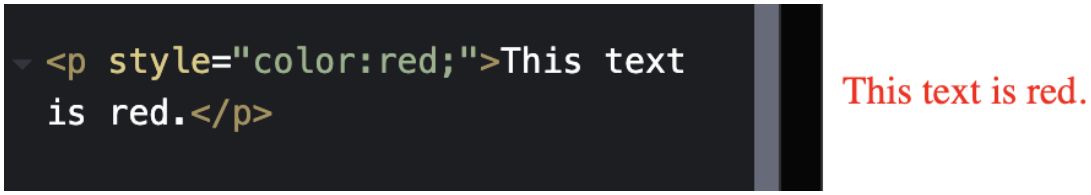
Essentially, every website, however shiny, is made with the same CSS that we'll be learning in this Training.

3 Methods to Add CSS to Your Web Page

To add CSS to a web page, you can use one of the following three methods:

Inline CSS: Inline CSS is added directly to the HTML element using the "style" attribute. For example, to set the text color of a paragraph to red, you can use the following code:

This text is red.



Internal CSS: Internal CSS is added to the head section of the HTML document using the "style" tag. For example:

```
<!DOCTYPE html>
<html>
  <head>
      <title>Student Info Collection Form</title>
      <style>
      p {
        color: red;
        }
      </style>
  </head>
      <p>This text is red.</p>
  <body>

  </body>
</html>
```

```html
<!DOCTYPE html>
<html>
  <head>
      <style>
      p {
        color: red;
        }
      </style>
  </head>
      <p>This text is red.</p>
  <body>

  </body>
</html>
```

This text is red.

External CSS: External CSS is added to a separate CSS file with a ".css" extension. The CSS file is then linked to the HTML document using the "link" tag in the head section of the HTML document. For example:

```
<head>
      <link rel="stylesheet" href="styles.css">
</head>
```

```html
<!DOCTYPE html>
<html>
  <head>
      <link rel="stylesheet" href="styles.css">
  </head>
      <p>This text is red.</p>
  <body>

  </body>
</html>
```

This text is red.

On the "styles.css" file

```css
p {
  color: red;
}
```

The sequence in which styles are applied

Inline styles : priority 1

internal styles : priority 2

Styles from an external style sheet : priority 1

Understanding the Basics of CSS Syntax and CSS Selectors

CSS syntax consists of a selector followed by one or more declarations enclosed in curly braces.

A declaration consists of a property, followed by a colon, and a value, separated by a semicolon.

CSS selectors, on the other hand, are used to target specific elements on a web page and apply styles to them. Selectors can target elements based on their element type, class, ID, attribute values, and more.

```
p {
  color: red;
}
```

- In the above example, the CSS rule opens with a selector. This selects the HTML element that we are going to style. In this case, we are styling the paragraph (p) element.
- We then have a set of curly braces { }.
- Inside the braces will be one or more declarations, which take the form of property and value pairs.

  Property : Value;

CSS comment syntax used to add notes or reminders to the stylesheet for developers to reference later.

In CSS, comments can be added by enclosing the comment in /* and */.

Example

```
/* This is a CSS comment that spans multiple lines.
You can add any notes or reminders here that can help you or other
developers understand the code. */
h1 {
color: blue; /* This sets the color of all h1 elements to blue */
}
```

CSS selectors

In CSS, selectors are used to target the HTML elements on our web pages that we want to style.

A CSS selector is the first part of a CSS Rule. It is a pattern of elements and other terms that tell the browser which HTML elements should be selected to have the CSS property values inside the rule applied to them. The element or elements which are selected by the selector are referred to as the subject of the selector.

We have already seen how to select single elements from html, we will see other forms of selecting.

Types of selectors

Element selector

An element selector selects one or more of a particular HTML element. An example of an element selector would be p { }, which, as we saw, styles all the paragraph (<p>) elements on a page.

For example:

HTML

```
<p>This text is red.</p>
```

This text is red.

CSS

```
p {
  color: red;
}
```

Class selector

A class selector targets all elements that have a particular class name. Just as a refresher, we add a class name to an element using the class attribute on an HTML element. For example:

HTML

```
<h4 class="question-text">How many sides are there in a triangle?</h4>
<p class="answer-text">3</p>
<h4 class="question-text">What does Pikachu evolve into?</h4>
<p class="answer-text">Raichu</p>
```

How many sides are there in a triangle?

3

What does Pikachu evolve into?

Raichu

CSS

```
.question-text {
  font-weight: bold;
}
.answer-text {
  color: green;
}
```

ID selector

ID selectors are quite similar to class selectors, except that we set an ID to an HTML element using the id attribute and select it in CSS using the hash or pound character before the ID name.

HTML

```
<p id="help-text">The send button submits the form</p>
```

The send button submits the form

CSS

```
#help-text {
  color: red;
}
```

IDs are used for the particular unique instance of an element that doesn't repeat on the page while **classes** are used for elements that are repeated.

Selector lists

If you have more than one thing which uses the same CSS then the individual selectors can be combined into a selector list so that the rule is applied to all of the individual selectors.

For example, if I have the same CSS for an h1 and also a class of .special, we could combine those selectors.

```
h1, .special {
  color: blue;
}
```

Combinators

combinators, they combine other selectors in a way that gives them a useful relationship to each other and the location of content in the document.

Descendant combinator

The descendant combinator — typically represented by a single space (" ") character — combines two selectors such that elements matched by the second selector are selected if they have an ancestor (parent, parent's parent, parent's parent's parent, etc.) element matching the first selector.

CSS

In the example below, we are matching only the <p> element which is inside an element with a class of .box.

```
<div class="box">

    <p>Text in .box</p>

</div>

<p>Text not in .box</p>
```

Text in .box

Text not in .box

```
.box p {
  color: red;
}
```

Child combinator

The child combinator (>) is placed between two CSS selectors. It matches only those elements matched by the second selector that are the direct children of elements matched by the first. Descendant elements further down the hierarchy don't match.

In this next example, we have an unordered list, nested inside of which is an ordered list. The child combinator selects only those <li> elements which are direct children of a <ul>, and styles them with a top border.

```
<ul>
  <li>Unordered item</li>
  <li>Unordered item
    <ol>
      <li>Item 1</li>
      <li>Item 2</li>
    </ol>
  </li>
</ul>
```

- Unordered item
- Unordered item
    1. Item 1
    2. Item 2

```
ul > li {
    border-top: 5px solid red;
 }
```

If you remove the > that designates this as a child combinator, you end up with a descendant selector and all <li> elements will get a red border.

Cascade, specificity, and inheritance

CSS stands for Cascading Style Sheets, and that first word cascading is incredibly important to understand — the way that the cascade behaves is key to understanding CSS.

Cascade

Stylesheets cascade — at a very simple level, this means that the origin, the cascade layer, and the order of CSS rules matter. When two rules from the same cascade layer apply and both have equal specificity, the one that is defined last in the stylesheet is the one that will be used.

```
<h1>This is my heading.</h1>
```


```
h1 {
  color: red;
}
h1 {
  color: blue;
}
```

Inheritance

Inheritance also needs to be understood in this context — some CSS property values set on parent elements are inherited by their child elements, and some aren't.

For example, if you set a color and font-family on an element, every element inside it will also be styled with that color and font, unless you've applied different color and font values directly to them

```
<p>As the body has been set to have
a color of blue this is inherited
through the descendants.</p>

<p>We can change the color by
targeting the element with a
selector, such as this
<span>span</span>. </p>
```

As the body has been set to have a color of blue this is inherited through the descendants.

We can change the color by targeting the element with a selector, such as this span.

```css
body {
  color: blue;
}

span {
  color: black;
}
```

CSS Colours explained

3 types of color values:

- hex values e.g. #FFFFFF
- RGB values e.g. rgb(255, 255, 255)
- color names e.g. white

To set:

- foreground (text) color use color:
- For Background color use background-color:

CSS Text Formatting

Example:

```html
<p id="help-text">The send button
submits the form</p>
```

The send button submits the form

```css
#help-text {
  color:blue;
  font-weight:bold;
  font-style:italic;
  text-decoration:underline;
  text-align:center;
  font-family:Arial;
  font-size:18px;

}
```

CSS Font

CSS font properties are used to adjust the appearance of the text in an HTML document. Using the CSS fonts properties, we can customize the font family, size, weight, style, and color of text.

```
body {
  font-family: Helvetica;
  font-size: 16px;
}
```

Here,

`font-family: Helvetica` - sets the font family of text to Helvetica within the body

`font-size: 16px` - sets the font size of the text to 16px within the body

Basic Font Properties

In CSS, we have the following seven important font properties that are used to change different attributes of the text.

- font-family: defines the font applied to the text, can be listed multiple and browser with select the first one it loads
- font-size: sets the size of the font
- font-weight: sets the thickness i.e increase the boldness or lightness of the font
- font-style: sets the font to italic or oblique
- font-variant: changes the font to small-caps
- font-stretch: expands or narrows the text
- line-height: sets the distance between lines of the text

```
<p>As the body has been set to have a
color of blue this is inherited through
the descendants.</p>

<p>We can change the color by targeting
the element with a selector, such as this
<span>span</span>. </p>
```

As the body has been set to have a color of blue this is inherited through the descendants.

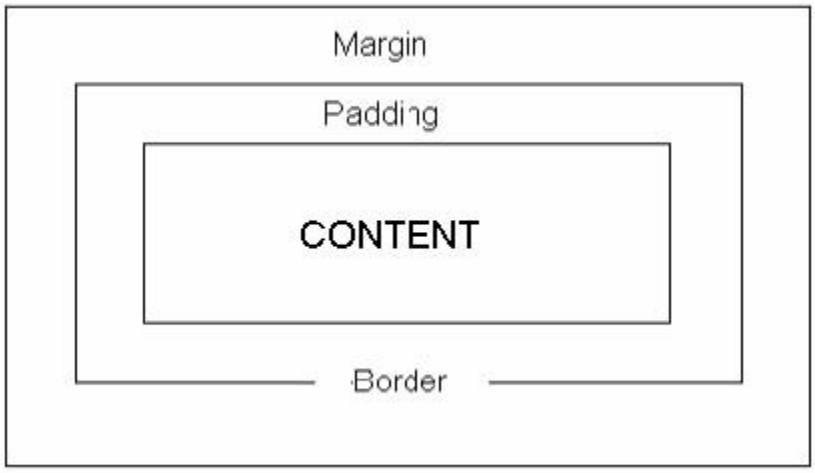
We can change the color by targeting the element with a selector, such as this span.

```css
body {
  font-family: Courier, monospace;
  font-size: 18px;
  font-style: italic;
  /*normal,italic */
  font-variant: small-caps;
  font-weight: bold;
}
```

CSS BOX MODEL

The HTML document is considered as a series of boxes which can be used to style the layout.

A pictorial representation of CSS Box Model is shown below.



As you can see, around the actual content, the CSS box model contains Padding, Border and Margin.

- **Padding in CSS:** The space around the content box. You can specify the size as required.
- **Border in CSS:** The line that separates the padding and margin. You can create dashed line, normal line and so on.
- **Margin in CSS:** The space after border till the end as shown in the image above. You can specify the size as required.

CSS Box Model Example

```
<div class="div1">
          Content of div1.
</div>
<div class="div2">
          Content of div2.
</div>
```

In the above code, there are two div tags, each with a different box style.

The CSS applied to the above code is given below.

```
.div1 {
       height:80px;
       width:400px;
       padding:30px;
       border:2px solid #FF3F34;
       margin:20px;
       background-color:#C4334D;
       }
.div2 {
       height:40px;
       width:500px;
       padding:10px;
       border:5px dashed #000123;
       }
```

CSS FLOAT

Browsers follows line by line order in displaying HTML documents. But with CSS Float property, the line by line order can be discarded and can make the HTML elements to float left or right.
Mostly CSS Float is used to make images float to make the text appear in the sides of the images instead of next line

Example
The HTML elements can float only left or right, not up or down.
Float Left CSS : If you specify the float property to be left, the HTML element will try to float to left as much as possible.
Float Right CSS : If you specify the float property to be right, the HTML element will try to float to right as much as possible.

CSS Float Example

```
<div style="float:left; width:50%;">
          <p>This is text floating on the left.</p>
</div>
<div style="float:left; width:50%">
          <img src="https://placehold.co/300" />
</div>
```

This is text floating on the left.

Float Clear

The HTML elements below the floating HTML elements will flow around it in the empty space.

To avoid it, the property "clear" with the value "both" is used as shown in the below code.

```
<div style="clear:both;">
</div>
```

CSS POSITION

Position and Float properties are very important to create a flexible layout.
CSS Position property defines the position of an element. The different kinds of positions are absolute, relative, static and fixed.

The position values are explained below using table.

Value	Description
Absolute	removes the element from normal flow of the page and pins it in the specified position of the page.
Relative	this property can be used to specify top, bottom, right and left of an element.
Static	it does nothing, the element will follow the normal flow of the page.
Fixed	it is similar to absolute, but instead of web page, it pins the element to the browser window.

CSS Position Example

```
<div style="position:fixed; width:50%">
          <img src="https://placehold.co/150"/>
</div>
```

Try To scrol through your page and see the effect: TRY IT

CSS DISPLAY

There are times when it is required to change the display property of some elements. For instance, **to create a simple navigation links**, the easiest way is to change the display property of list elements to show the list in the same line instead of one per each line.

The CSS display property's values and description is provided in the below table.

Value	Description
Inline	As the name implies, the inline property makes the element to follow the flow of a line.
Block	It makes the HTML element to be a stand alone element, it is helpful to manipulate height and width of a specific element effectively.
None	It makes the element to disappear, the code will be there but the element won't be displayed.

Example

HTML

```
<ul>
     <li><a href="" target="_parent">Home</li>
     <li><a href="" target="_parent">About Us</li>
     <li><a href="" target="_parent">Contact Us</li>
</ul>
```

- Home
- About Us
- Contact Us

CSS

```
li {
   display:inline;
   }
```

CSS grid

CSS grid is a really powerful tool used to structure parts of your webpage in a grid so that you can organize and control the layout of your page. A grid in CSS is exactly what it is in real life; a space divided by horizontal and vertical lines into rows, columns, and cells—a table, if you'd like to think about it that way.

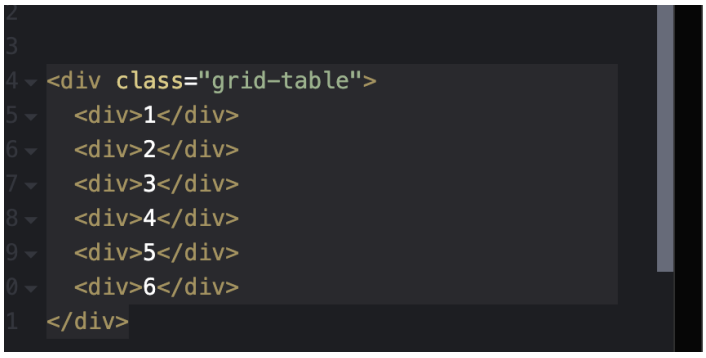
With CSS grid, we first define how our grid is structured in terms of percentages, pixels or any other measurements, and for each item that we put into our grid, we define what cell that element should go in, roughly speaking.

Let's look at an example using a simple 2x3 table:

1	2	3
4	5	6

To make this table, we do the following:

HTML



```
<div class="grid-table">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```

1 2 3
4 5 6

CSS

```
.grid-table {
  display: grid;
  grid-template-rows: auto auto;
  Grid-template-columns: auto auto auto;
}
```

`display: grid` tells the browser that this is a grid container (i.e. we intend to have rows and columns inside this element), while grid-template-rows and grid-template-columns do exactly as their names suggest, set the number and dimension of each row and column in the grid respectively.

We define a grid with two auto width rows and three auto width columns. Auto width is an automatically set width, which means our grid is flexible and can end up looking very asymmetric depending on what content we put inside it.

For instance, this is also a 2x3 grid, exactly the same in terms of the number of rows and columns, but the columns (and rows) have different specified dimensions.

Instead of auto, you can use percentages (50% 50%) or pixels (300px 300px) or any other unit of measurement at your disposal. Grids are very versatile. If you can draw it on paper with a ruler, you can make a grid out of it!

So, why do we need a grid? It has to do with how your content will be viewed on different screen sizes. That brings us to our next topic, designing for mobile responsiveness.

Mobile responsiveness

It is imperative that we embrace responsive web design right from the start and remember that our website should be just as usable on a laptop or a smartphone.

media queries

A media query is a conditional block of CSS code that only runs the CSS code inside if the condition in the media query is met. Let's consider an example now.

```
grid {
  display: grid;
  grid-template-columns: 100%;
}

@media (min-width: 600px) {
  .grid {
    grid-template-columns: 50% 50%;
  }
}

@media (min-width: 900px) {
  .grid {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
```

grid-template-columns, as we saw, sets the number of columns and their dimension. Here, it is taking the value 100% (one column with full width), 50% 50% (two columns with 50% widths each) and 1fr 1fr 1fr (three columns with 1 fractional unit width each. I.e. ⅓ or three equal sized columns).

Notice the two lines starting with a @media. Those are our media queries. The condition we're using here is min-width, which essentially says to run this code if the width is at least the specified number. So, if the screen size is more than 600 px, it will use 2 columns, and if it is above 900px, it will use 3 columns. 600px and 900px here are formally known as breakpoints, as in, those are the widths at which our webpage transforms into something different.

Note that we didn't repeat *display: grid*. This is because unchanging properties don't need to be duplicated. That's how all responsive web development is done, no matter how complicated it might seem at first.

Using this we can redo the grid example above and check for 3 different screen sizes

```css
.grid {
  display: grid;
  grid-template-rows: auto auto;
  Grid-template-columns: 100% ;
}

@media (min-width: 600px) {
  .grid {
    grid-template-columns: 50% 50%;
  }
}

@media (min-width: 900px) {
  .grid {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
```

For smaller screens <600px

```html
<div class="grid">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>

</div>
```

1
2
3
4
5
6

For screens 600px <= screensize < 900px

```html
<div class="grid">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>

</div>
```

1
3
5

2
4
6

For larger screens >900px

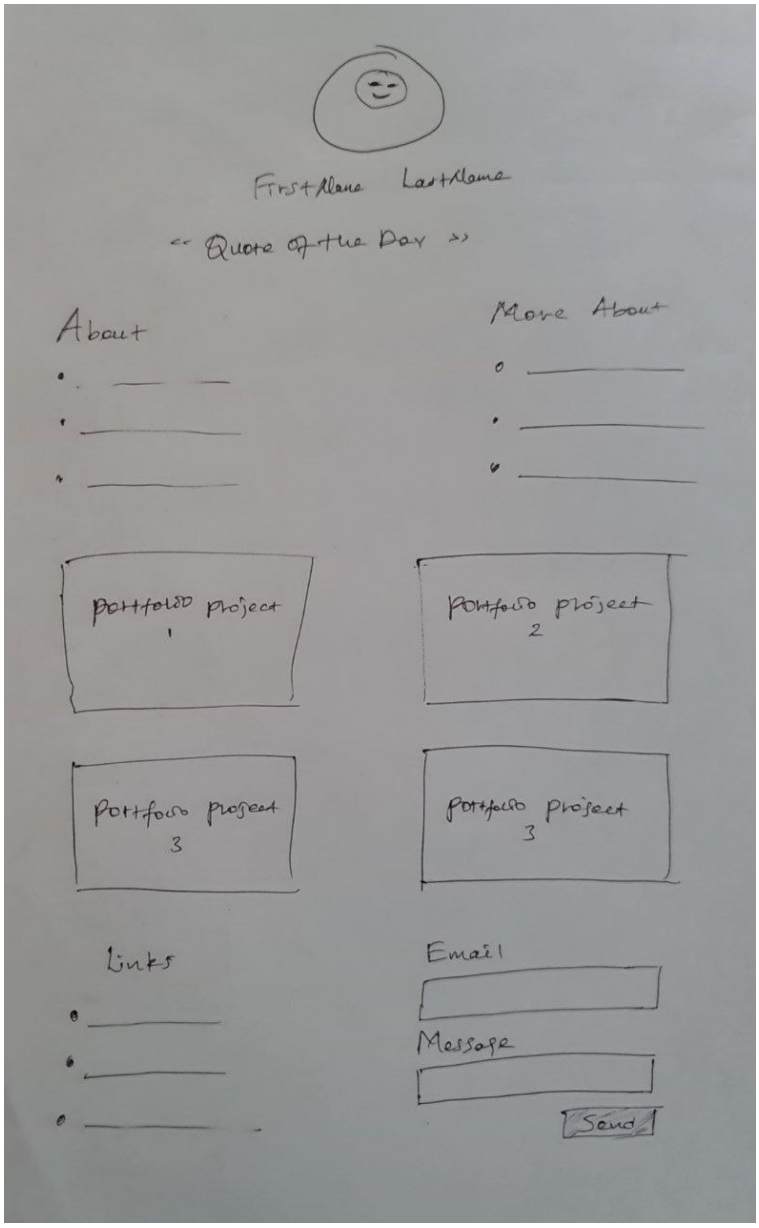
```
<div class="grid">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>

</div>
```

1 2 3
4 5 6

Back to our Portfolio Site

Positioning elements on your page

Remember the wireframe from the previous HTML work we did? Let’s look at it again



As per our wireframe, we wish to give our website a fixed width and to center it on the screen. To do that, we'll wrap all of our content in an HTML element and set equal 'margins' (or empty space) on both sides.

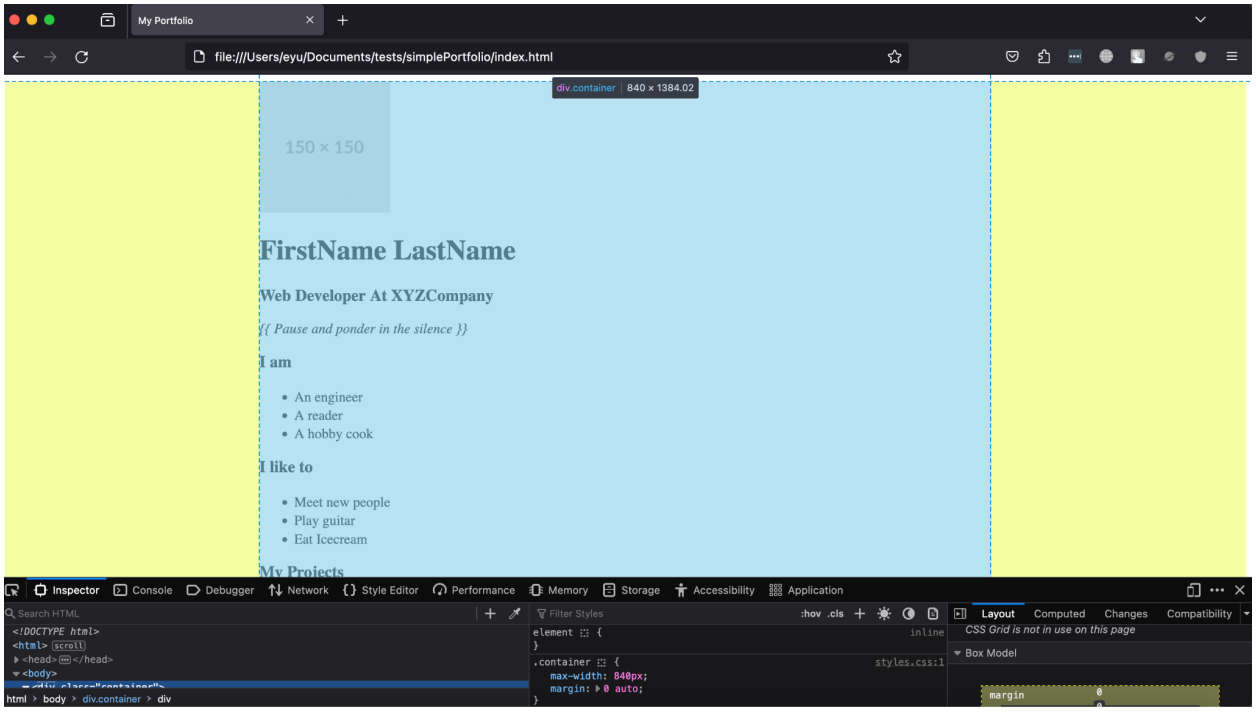
HTML

```
<body>

  <div class="container">

    <!-- all of our existing content goes here -->

  </div>

</body>
```

CSS

```
.container {
    max-width: 840px;
    margin: 0 auto;
}
```

If you refresh the page, you'll see that there's a margin on both sides of our content. This is called a fixed width layout (because we specified a particular width).



Next up, we would like to have the name, professional title, and quote center aligned. We'll name the div holding our content 'intro', and then style it.

HTML

Change

```
    <div>
        <img src="https://placehold.co/150" />
        <h1>FirstName LastName</h1>
        <h3>Web Developer At XYZCompany</h3>
        <p> <i> {{ Pause and ponder in the silence }} </i> </p>
    </div>
```

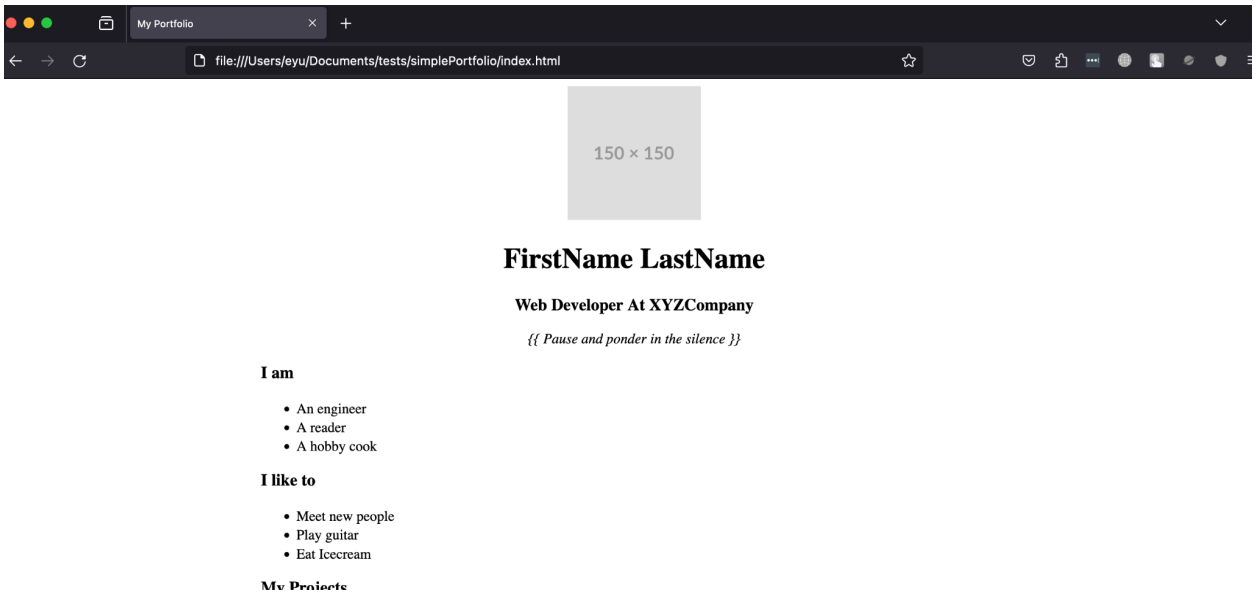
To

```
    <div class="intro">
        <img src="https://placehold.co/150" />
        <h1>FirstName LastName</h1>
        <h3>Web Developer At XYZCompany</h3>
        <p> <i> {{ Pause and ponder in the silence }} </i> </p>
    </div>
```

CSS

Add this .intro section to your css file which should now look like

```
.container {
    max-width: 840px;
    margin: 0 auto;
}
.intro {
  text-align: center;
}
```



This is what you should see. The Page is coming up to the structure we wanted right?

Good. Let's move onto the next section. We need the 'about me' lists to be arranged horizontally, one on the left and the other on the right. Let's use CSS grid!

HTML

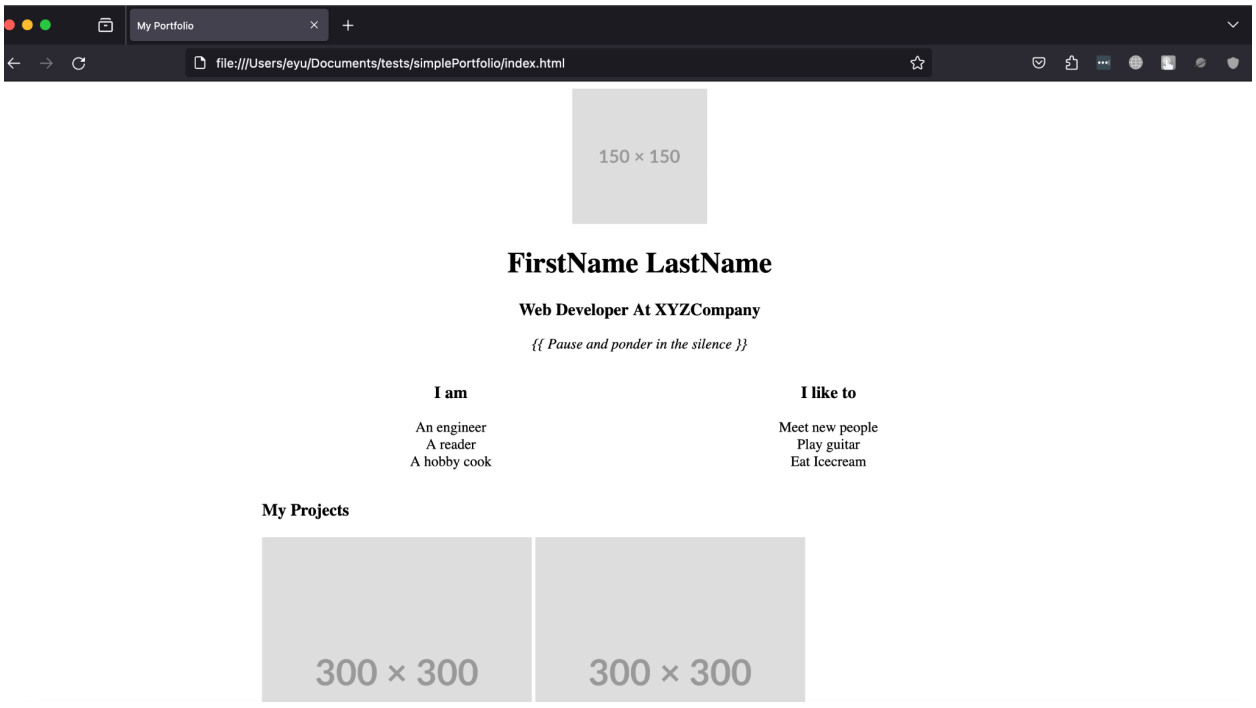
Change this

```
<div>

    <div>
        <h3>I am</h3>
        <ul>
        <li>An engineer</li>
        <li>A reader</li>
        <li>A hobby cook</li>
        </ul>
    </div>
    <div>
        <h3>I like to</h3>
        <ul>
        <li>Meet new people</li>
        <li>Play guitar</li>
        <li>Eat Icecream</li>
        </ul>
    </div>

</div>
```

To

```html
<div class="about-grid">

    <div class="i-am">
        <h3>I am</h3>
        <ul class="about-list">
            <li>An engineer</li>
            <li>A reader</li>
            <li>A hobby cook</li>
        </ul>
    </div>
    <div class="i-like">
        <h3>I like to</h3>
        <ul class="about-list">
            <li>Meet new people</li>
            <li>Play guitar</li>
            <li>Eat Icecream</li>
        </ul>
    </div>

</div>
```

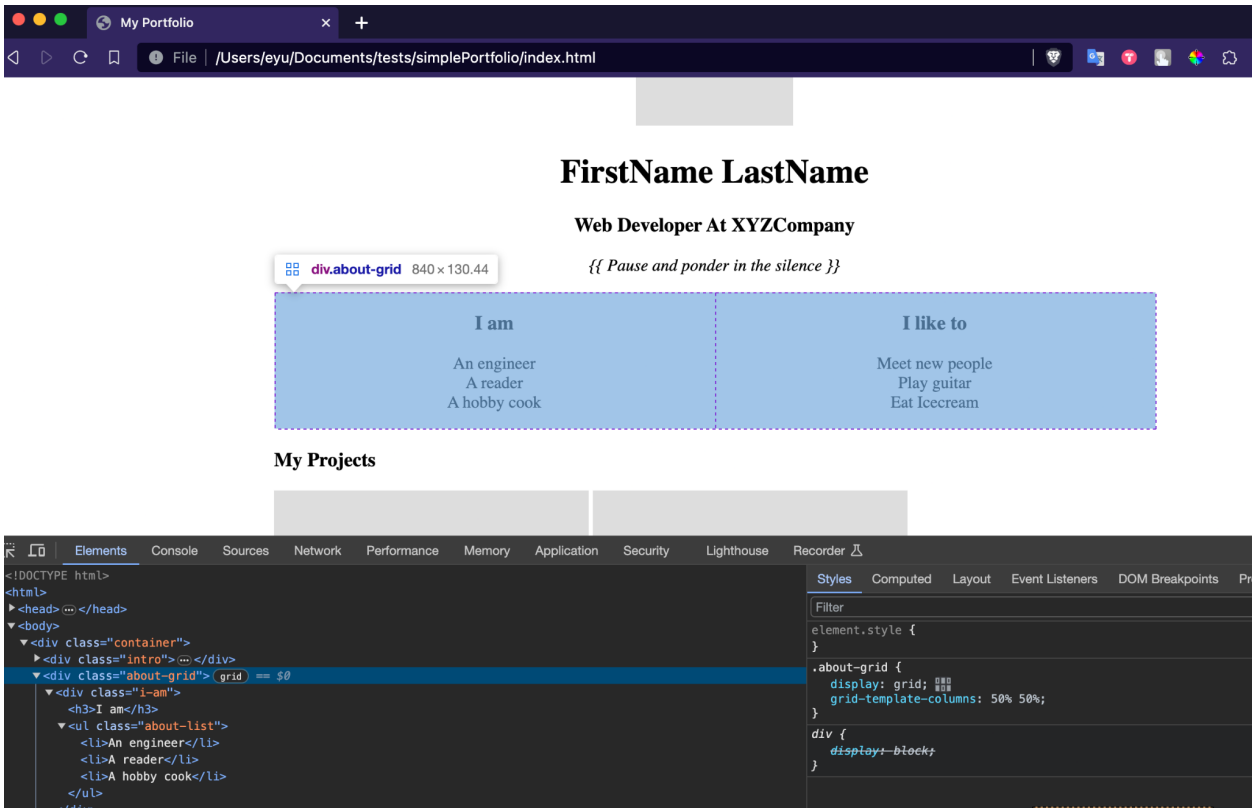
CSS



Notice how we declare a 2-column grid in the **.about-grid** class with 50% percent width each. As you learn more about grids, you'll learn more ways of accomplishing this very

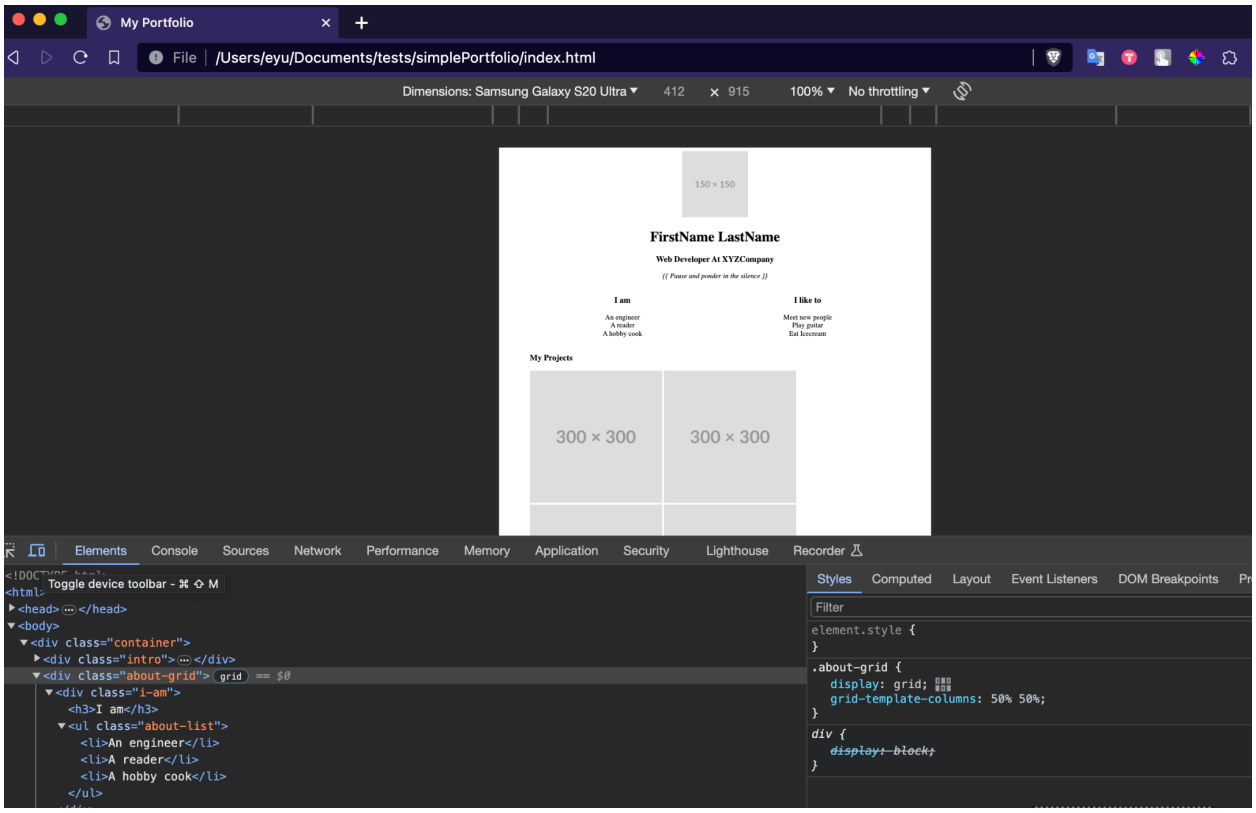
thing, and depending on the situation, one way will have certain advantages over the other. For our use case, 50% is good.

Notice how you got the two lists centered and split in the middle. We can use Chrome Developer Tools to inspect the grid and see exactly what is happening.



Looks good. But I wonder what happens if we resize the browser and simulate a mobile screen? To do that, in the Developer Tools, click on the “Toggle device toolbar” button. If you can’t find it, just use the shortcut **Ctrl+Shift+M**.

You should see a reduced sized window pop up.



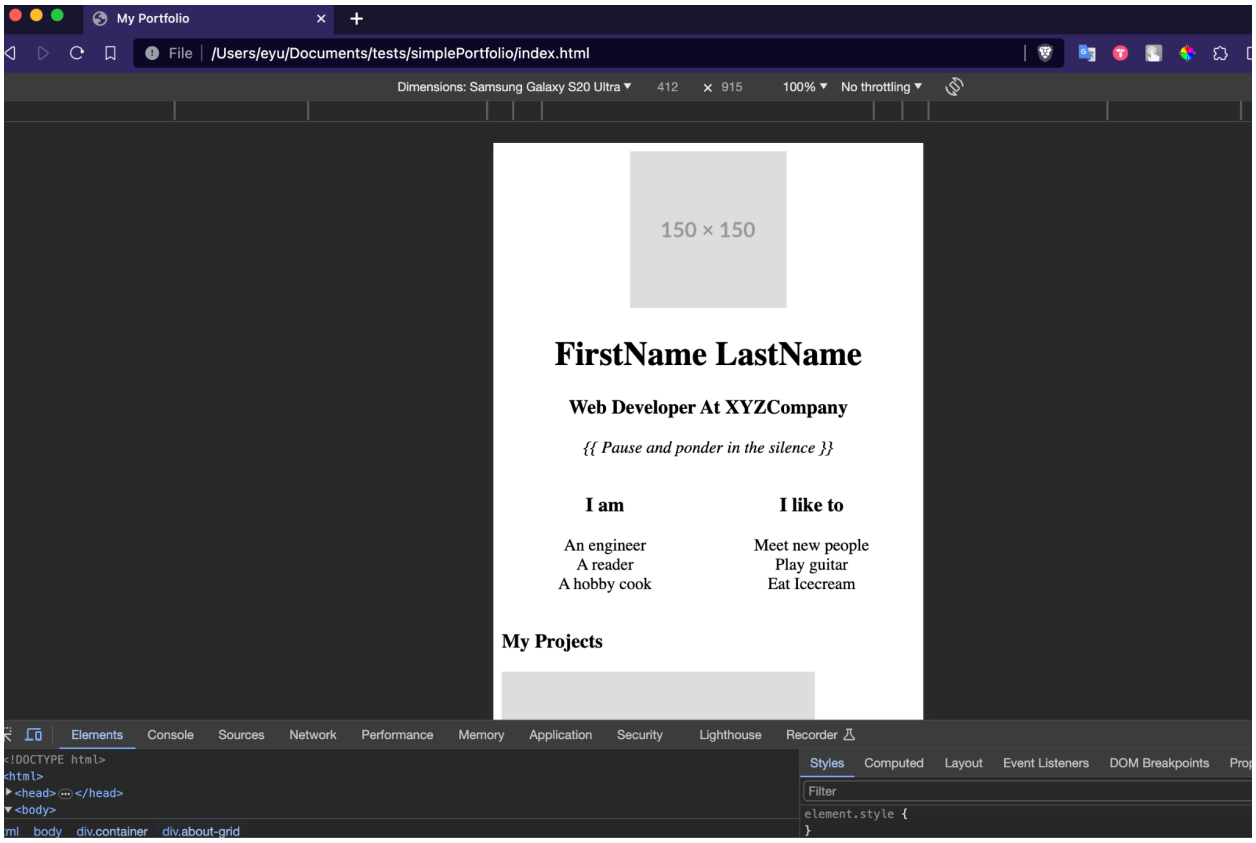
Note the menu on top where it says “Responsive”. Here you can simulate the screen sizes of different devices (say you want to see how a page looks on iPhone 6). If you look at our website, it looks a bit weird. We need to fix this by adding a new HTML tag in our <head>. We place it just before the closing head (</head>) tag. This tag hints the browser to show the webpage in mobile size (and not a zoomed out desktop version of the website).

HTML

Add the meta tag as shown bellow

```
<head>
    <title>My Portfolio</title>
    <link rel="stylesheet" href="styles.css">

    <meta name="viewport" content="width=device-width, initial-scale=1">

</head>
```

Let’s add the tag and refresh our browser.

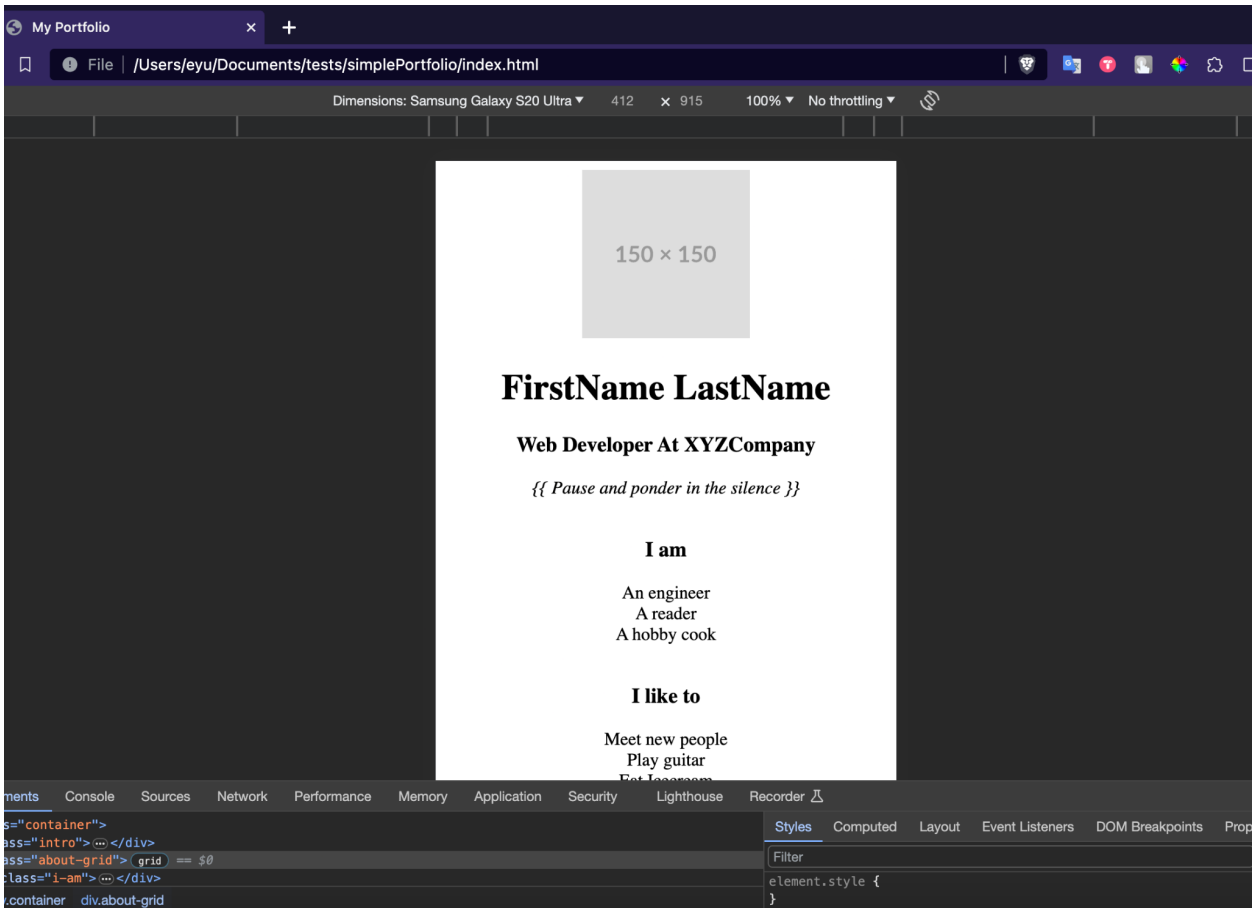


Okay, now that looks less weird, although still not perfect. We would ideally like to have the “About me” lists to be one below the other on mobile devices and screens that are something like below 480px. Let’s write the code for that.

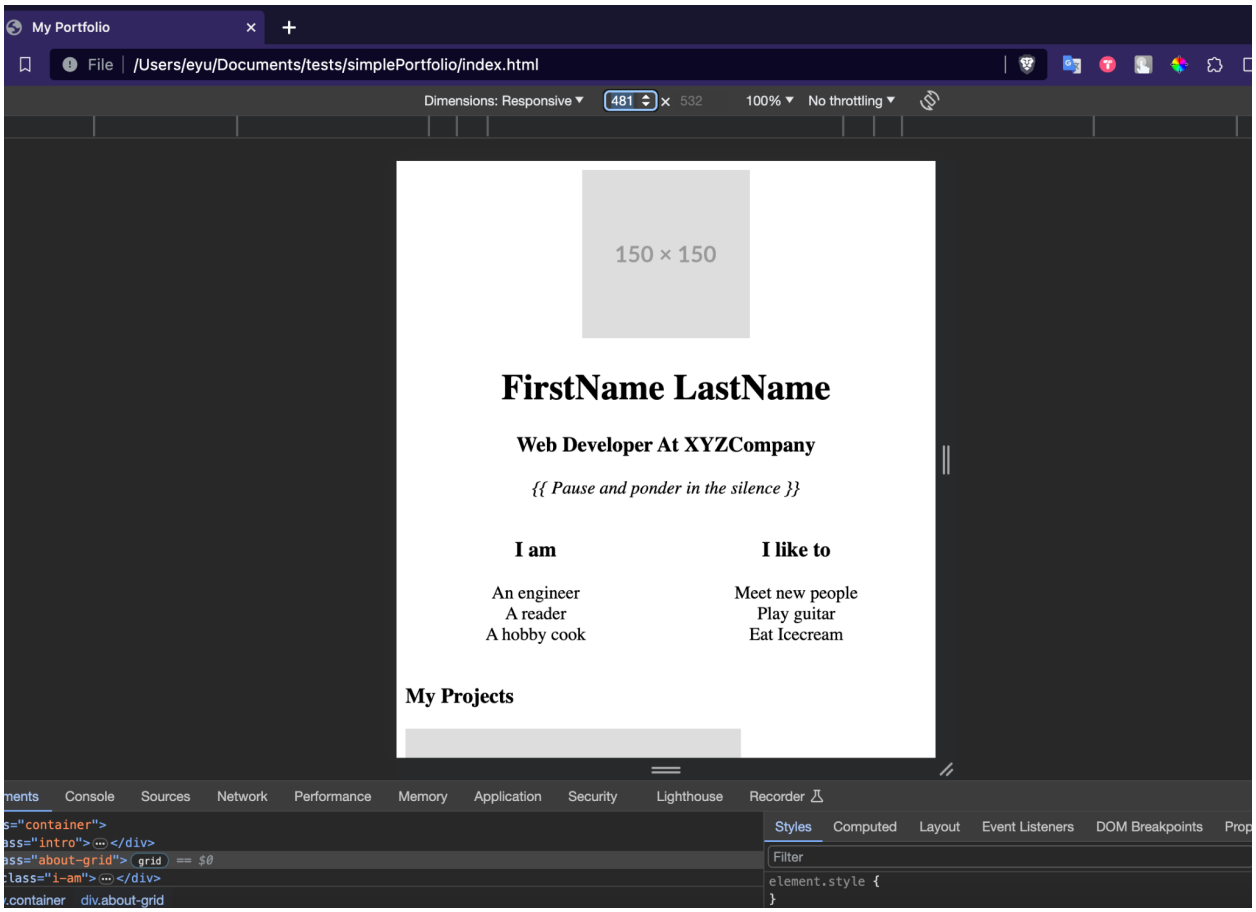
CSS

```css
@media (max-width: 480px) {
  .about-grid {
    grid-template-columns: 100%;
  }
}
```

That’s it. We define a breakpoint of 480px, and say that if the screen size goes below that (with max-width), the code inside should run. Let’s refresh our browser and check if that works.



That's better! To make sure the media query works, change the width of the window (use the input field above the window that has the **Responsive** dropdown) in which our website is rendered from 412px to 481px (just over our breakpoint).

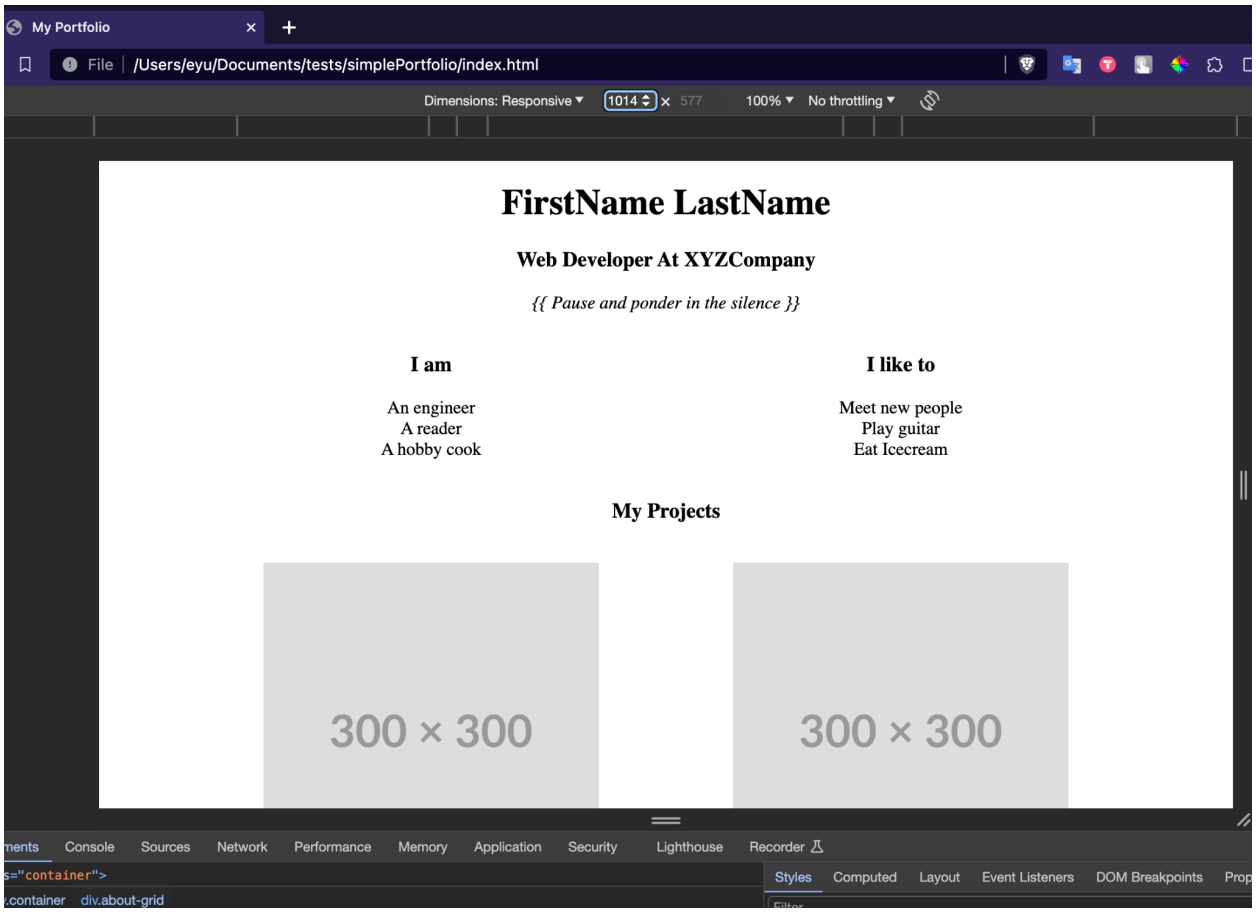


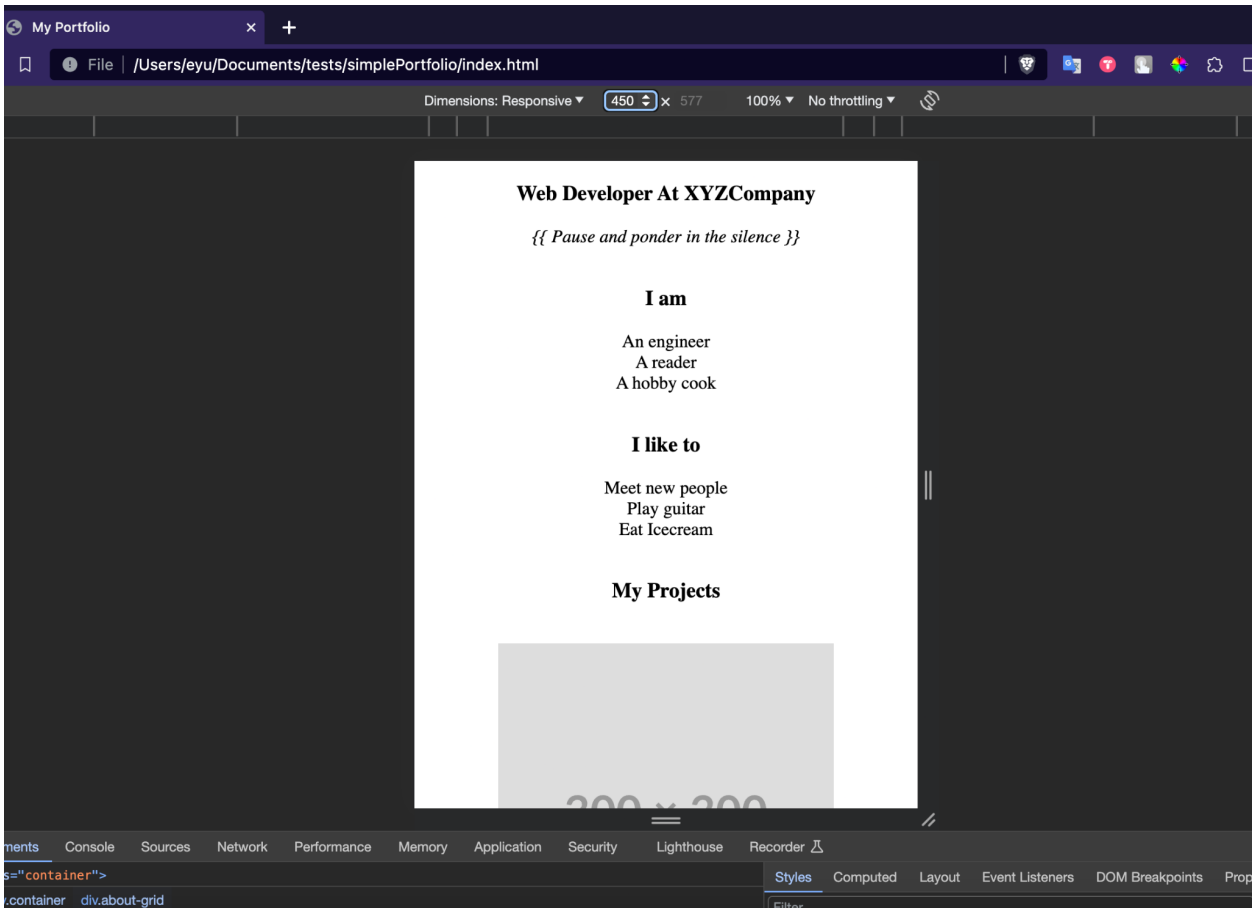
Did you see how it changed to the 50% split that we coded earlier above 480px width? That's the power of media queries!

We can now make similar changes to all the remaining sections. For the projects grid, we'll make it 2x2 on desktop and 4x1 on mobile. At this point, you should try to read the code yourself and understand what's being done. That will also get you into the habit of reading code, yet another skill a developer needs to have.

TRY TO DO IT YOURSELF

It should look like this once you are done


My Portfolio
File /Users/eyu/Documents/tests/simplePortfolio/index.html
Dimensions: Responsive 1014 x 577 100% No throttling
FirstName LastName
Web Developer At XYZCompany
{{ Pause and ponder in the silence }}
I am
An engineer
A reader
A hobby cook
I like to
Meet new people
Play guitar
Eat Icecream
My Projects
300 × 300
300 × 300
Console Sources Network Performance Memory Application Security Lighthouse Recorder
Styles Computed Layout Event Listeners DOM Breakpoints


My Portfolio
File /Users/eyu/Documents/tests/simplePortfolio/index.html
Dimensions: Responsive 450 x 577 100% No throttling
Web Developer At XYZCompany
{{ Pause and ponder in the silence }}
I am
An engineer
A reader
A hobby cook
I like to
Meet new people
Play guitar
Eat Icecream
My Projects
Console Sources Network Performance Memory Application Security Lighthouse Recorder
Styles Computed Layout Event Listeners DOM Breakpoints

Here is the Code answer

HTML

```html
    <h3  class="projects-heading" >My Projects</h3>
    <div class="projects-grid">

        <img class="project-image" src="https://placehold.co/300"/>
        <img class="project-image" src="https://placehold.co/300"/>
        <img class="project-image" src="https://placehold.co/300"/>
        <img class="project-image" src="https://placehold.co/300"/>
    </div>
```

CSS

```css

.projects-heading {
    text-align: center;
}

.projects-grid {
    display: grid;
    grid-template-columns: 50% 50%;
}

.project-image {
    justify-self: center;
    padding: 4%;
}

@media (max-width: 650px) {
    .projects-grid {
        grid-template-columns: 100%;
    }
}
```

What We are left with is the links and contact section.

This section should also be divided into two columns, but not split by 50-50. Since links are relatively narrower, let's split this into 30-70.

HTML

```
<div class="links-and-contact">
    <div class="links">
        <h3>Links</h3>
        <ul class="links-list">
            <li>
                <a href="https://github.com/<user>">Github</a>
            </li>
            <li>
                <a href="https://twitter.com/<user>">Twitter</a>
            </li>
            <li>
                <a href="https://linkedin.com/in/<user>">Linkedin</a>
            </li>
        </ul>
    </div>

    <div>
        <form action="#">
            <label for="email">
                Email: <input type="email" id="email" placeholder="Enter your email" />
            </label>
            <label for="message">
                Message: <textarea id="message">Your Message</textarea>
            </label>
            <input type="submit" valid="Send Message" />
        </form>
    </div>
</div>
```

CSS

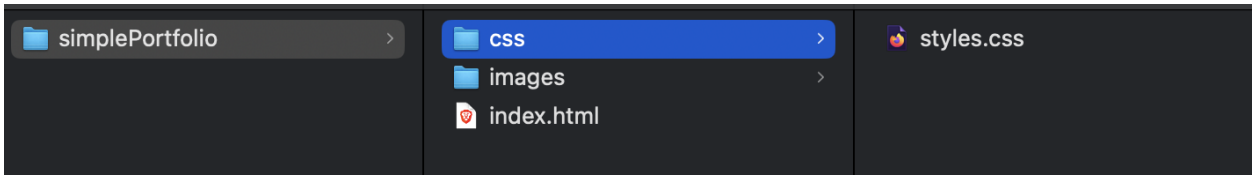
```
.links-and-contact {
  display: grid;
  grid-template-columns: 30% 70%;
}

.links {
  justify-self: center;
}

.links-list {
  list-style: none;
  padding: 0;
}
@media (max-width: 650px) {
  .links-and-contact {
    grid-template-columns: 100%;
  }
}
```

Prepping the ‘images’ folder

In general, whenever we work with images on a webpage (which is almost always), we try to separate files according to their types.

- So all images go into one folder,
- CSS files go into another folder, and so on.

So lets first create a new folder called css to store our css files eventhough we have only one css file styles.css



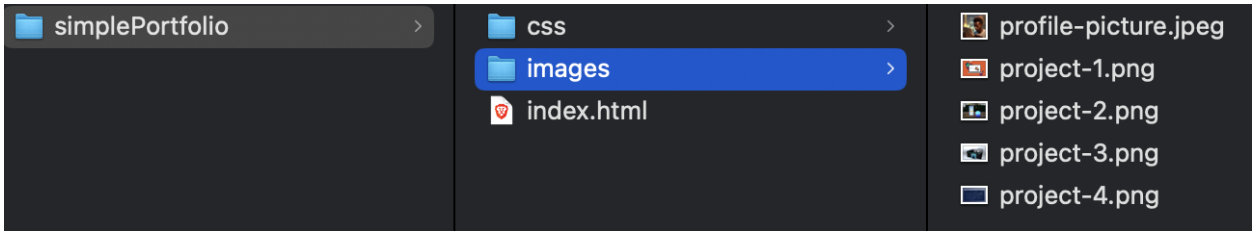
Lets change the css import link statement in our header

```
<head>
    <title>My Portfolio</title>
    <link rel="stylesheet" href="css/styles.css">
```

We will also create a new folder called ‘images’ to hold all of our images.

In this ‘images’ folder, you’ll need to add some images that we’ll be using in this course. I’ll provide you with the images

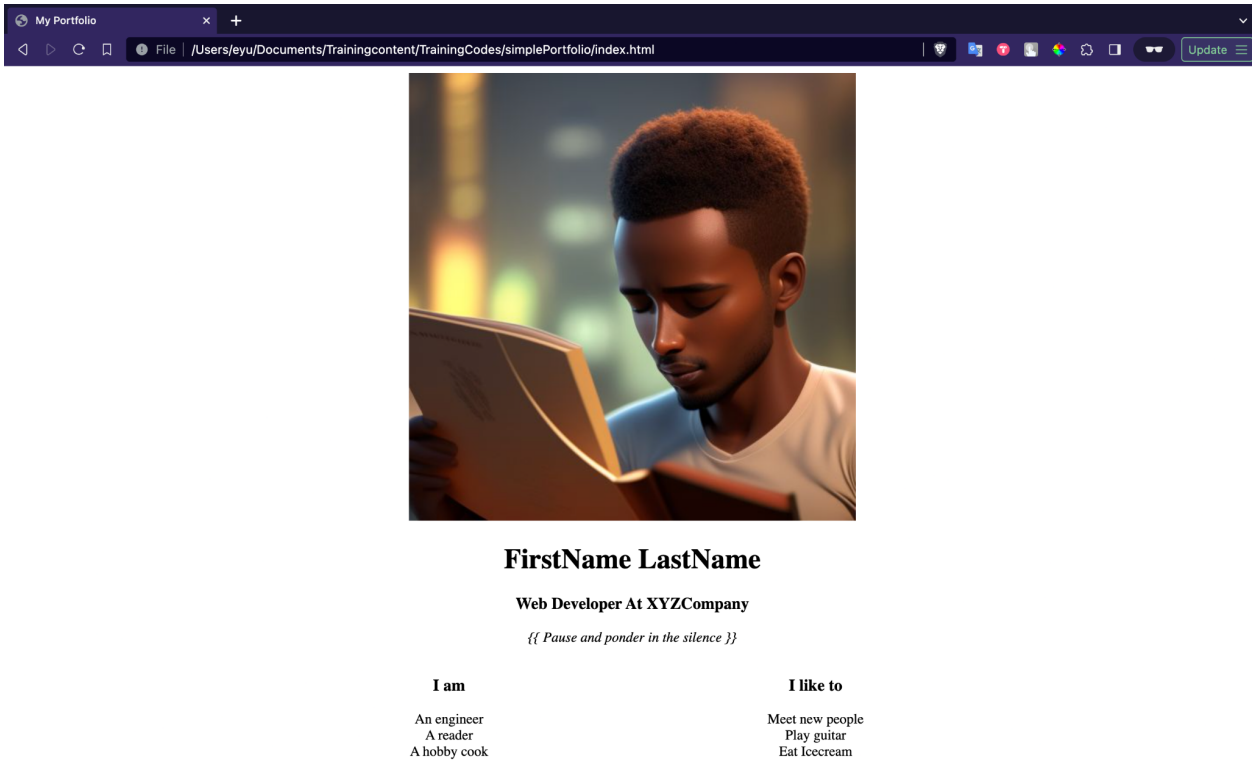
download the images file from **this link** and unzip it inside the folder you just created. Once that’s done, this is how your folder structure should look like.



Now we’re ready to start using our images. First, we’ll replace the profile picture. In our HTML file, we’ll replace this line in the intro section:

```
<body>
    <div class="container">
        <div class="intro">
            <img src="images/profile-picture.jpeg" />
            <h1>FirstName LastName</h1>
```

Notice that the src attribute has changed. Essentially, we're telling the browser to pick the image stored locally (hence no **https://**) and display it. Let's reload the browser and see how it looks!



Not bad, but maybe a bit plain as a square image., plus it is too large, Let's make it round and add some shadows, add height and width in css.

Add a class called `*profile-picture*` to the same image tag in your HTML file and then select it in CSS, like so:

```
<div class="container">
        <div class="intro">
            <img class="profile-picture" src="images/profile-picture.jpeg" />
            <h1>FirstName LastName</h1>
            <h3>Web Developer At XYZCompany</h3>
```

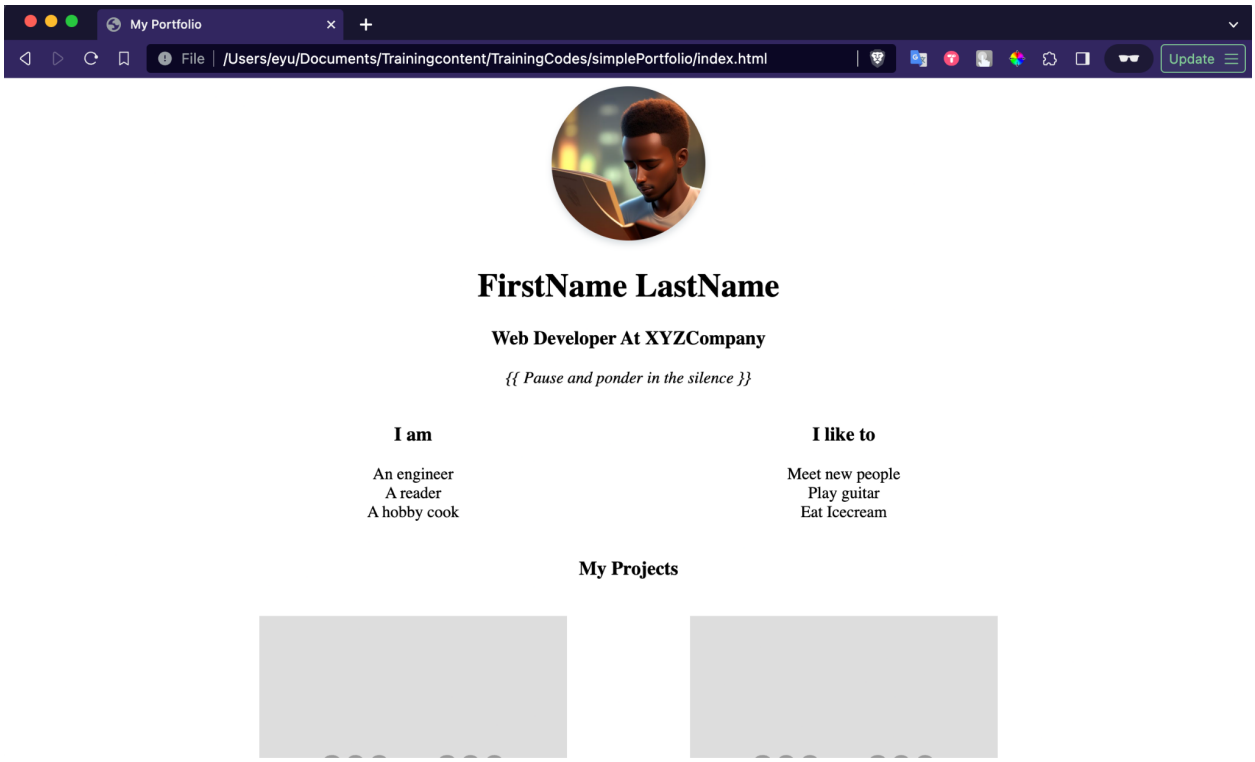
CSS

```css
.profile-picture {
  border-radius: 50%;
  width: 150px;
  height: 150px;
  box-shadow: 0 4px 6px 0 rgba(34, 60, 80, .16);
  transition: all ease-in-out .2s;
}

.profile-picture:hover {
  box-shadow: 0 8px 12px 0 rgba(34,60,80,.16);
}
```

Ookay, that's a mouthful. There are a couple of new properties in there that we haven't seen before. Let's explore those briefly now:

- **border-radius** turns the square image into a circle. By controlling the size of the border-radius, we can get anything from a nice round-cornered rectangle to a circle. Try setting it to a different percentage and see what happens.
- **box-shadow** adds nice shadows to an HTML element. The value for that property is best described by this page on **Mozilla Dev Docs**.
- **.profile-picture:hover** { .. } sets CSS properties for the element when we hover over it with our mouse pointers (or touch it on smartphones and other touchscreen devices). Here, we're just increasing the size of the shadow to make it feel like it reacts to our gesture.

Refresh your browser and notice the shadow. Now move your mouse pointer over the image and see how the shadow changes, giving the illusion that the image is being raised from the canvas. Beautiful, isn't it?



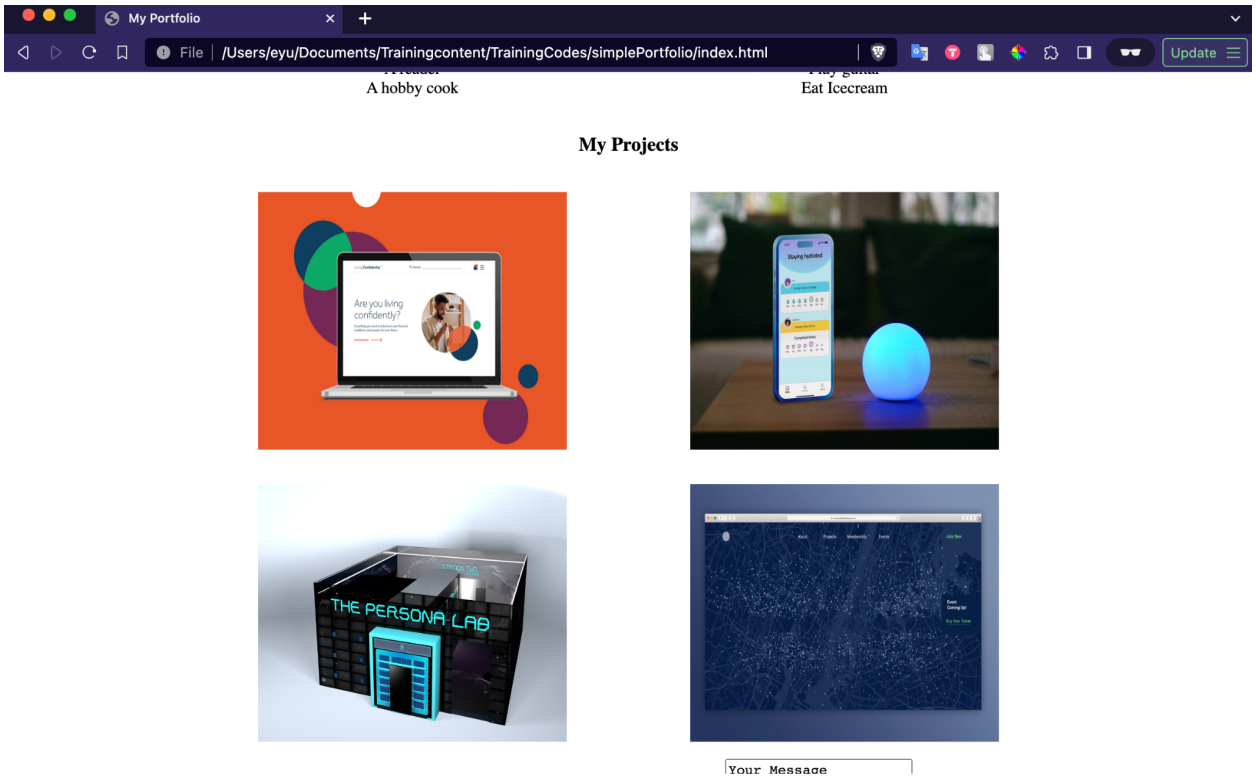
Replacing project placeholder images with our own

Let's also replace the images in the project section with our own images. In the folder you downloaded earlier today, there are four images named project-1.png through project-4.png. We'll replace the placeholder images with each of these.

HTML

```
<h3  class="projects-heading" >My Projects</h3>
<div class="projects-grid">

    <img width="300" height="250" class="project-image" src="images/project-1.png"/>
    <img width="300" height="250" class="project-image" src="images/project-2.png"/>
    <img width="300" height="250" class="project-image" src="images/project-3.png"/>
    <img width="300" height="250" class="project-image" src="images/project-4.png"/>
</div>
```

So now you should have the more colorful images showing up. In the future, these images could be replaced by some real portfolio projects that you work on!



Adding effects to your images

Lets try to add cool effects to your images. Let's add some rounded corners, hover effects and hover text. We already did something similar for the profile picture a couple of sections above, so not everything is new here.

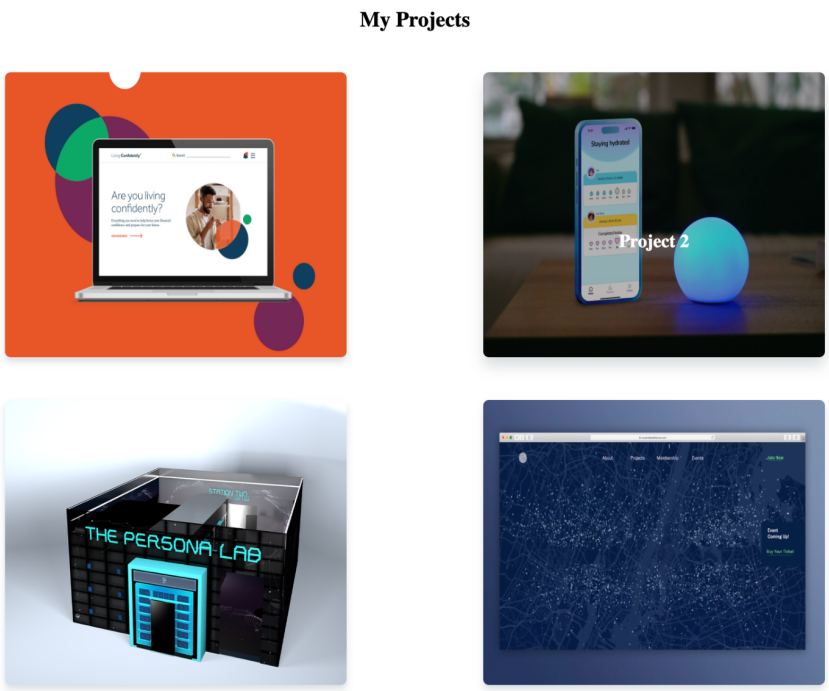
Essentially, here's what we're doing. We'll give the images some rounded corners. Then we'll give them some shadow and some deeper shadow on hover. So far, that's exactly what we did for the profile picture. But now, we'll add each image into a and add an h4 (that's a heading 4). We'll make it so that the heading only shows when we hover on the image.

We'll also add a little brightness filter to the image so that the text is readable, and a transition to make everything a bit smoother.

```
<h3  class="projects-heading" >My Projects</h3>
<div class="projects-grid">

    <div class="project-image-wrapper">
        <h4>Project 1</h4>
        <img width="300" height="250" class="project-image" src="images/project-1.png"/>
    </div>

    <div class="project-image-wrapper">
        <h4>Project 2</h4>
        <img width="300" height="250" class="project-image" src="images/project-2.png"/>
    </div>

    <div class="project-image-wrapper">
        <h4>Project 3</h4>
        <img width="300" height="250" class="project-image" src="images/project-3.png"/>
    </div>

    <div class="project-image-wrapper">
        <h4>Project 4</h4>
        <img width="300" height="250" class="project-image" src="images/project-4.png"/>
    </div>


</div>
```

CSS

```
.project-image {
  border-radius: 5px;
  box-shadow: 0 4px 6px 0 rgba(34, 60, 80, .16);
  transition: all ease-in-out .2s;
}

.project-image:hover {
  filter: brightness(.75);
  box-shadow: 0 8px 12px 0 rgba(34, 60, 80, .16);
}

.project-image-wrapper {
  justify-self: center;
  padding: 4%;
  position: relative;
}

.project-image-wrapper:hover > h4 {
  visibility: visible;
}

.project-image-wrapper > h4 {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  color: white;
  visibility: hidden;
  z-index: 2;
  transition: all ease-in-out .2s;
}
```



That looks great, doesn't it? Here's a brief explanation of the new CSS properties used:

- **visibility:** [visible/hidden] shows or hides an element
- **position:** [relative/absolute] **position: absolute** positions an element relative to the parent with a **position: relative**. That is, the parent with **position: relative** becomes a reference point and the child with **position: absolute** can be arranged relative to this parent like we've done by positioning the h4 to be 50% from the top and 50% from left, essentially centering it inside the parent viz. **.project-image-wrapper** class.
- **transform:** translate(X, Y) moves an element X percent along the x-axis and Y percentage along the y-axis. This is helpful when centering a child element inside a parent.
- **z-index** is about the stacking order. If two elements overlap each other, the element with a higher z-index will appear on top of the one with a lower z-index.
- **filter:** brightness(0.75) creates a brightness filter and reduces the brightness to 75% when this property is active.

Setting custom fonts

A font define the style with which text is displayed. We web developers love to make pretty websites, and sometimes we need to use a font (i.e. style of text) that's not given to us by the browser directly (browsers do provide us with some fonts out of the box).

That's not a problem, because CSS allows us to download and use any custom font that we want to use.

For this training, we've selected two beautiful fonts for you.

- For headings and titles, we'll use **Roboto Mono**, and
- for body text we'll use **Noto Sans**.

The good thing about both of these fonts is that they're free to use and are available at **https://fonts.google.com/** for download.

To use them, we'll simply paste this line into the <head> section of our HTML page just above the line that we use to import our styles.css

```
<link
href="https://fonts.googleapis.com/css?family=Noto+Sans|Roboto+Mono&display=swa
p" rel="stylesheet">
```

```
<head>
    <title>My Portfolio</title>

    <link href="https://fonts.googleapis.com/css?family=Noto+Sans|Roboto+Mono&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="css/styles.css">

    <meta name="viewport" content="width=device-width, initial-scale=1">
```

We'll use the font-family property on the element we wish to style in this particular font, and add the value 'Roboto Mono' or 'Noto Sans'.

For a quick start, we'll make the **default body** font Noto Sans and **default title** font Roboto Mono:

Add this at the top of styles.css

CSS

styles.css

css > styles.css > .i-am

```css
body {
  font-family: 'Noto Sans', sans-serif;
}

h1, h2, h3, h4 {
  font-family: 'Roboto Mono', monospace;
}
```

Refresh the page and notice the difference in font styles! That's how we add and use custom fonts.

BEFORE

FirstName LastName

Web Developer At XYZCompany

{{ Pause and ponder in the silence }}

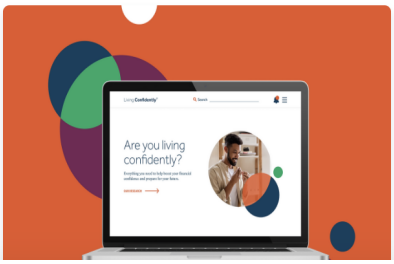
I am

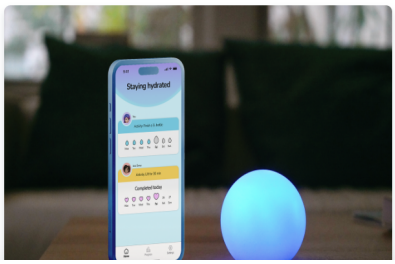
An engineer
A reader
A hobby cook

I like to

Meet new people
Play guitar
Eat Icecream

My Projects





AFTER

FirstName LastName

Web Developer At XYZCompany

{{ Pause and ponder in the silence }}

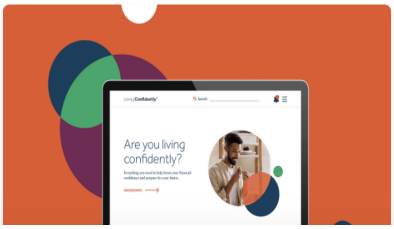
I am

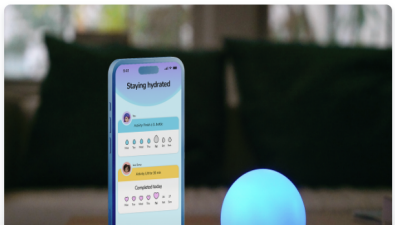
An engineer
A reader
A hobby cook

I like to

Meet new people
Play guitar
Eat Icecream

My Projects





Footer fixes

Now that our intro and portfolio sections have been taken care of, let's focus on the footer. Let's break the form in the footer so that each input label and input box appears on its own line. That makes it a bit cleaner.

HTML

Change this

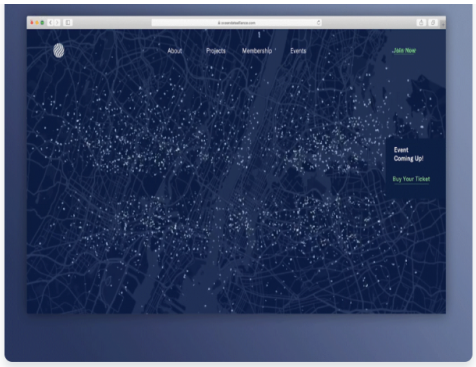
```
<div>
    <form action="#">
        <label for="email">
            Email: <input type="email" id="email" placeholder="Enter your email" />
        </label>
        <label for="message">
            Message: <textarea id="message">Your Message</textarea>
        </label>
        <input type="submit" valid="Send Message" />
    </form>
</div>
```

TO

```html
<div>
    <form action="#">
        <label for="email">
            <h4>Email</h4>
            <input type="email" id="email" placeholder="Enter your email" />
        </label>
        <label for="message">
            <h4>Message</h4>
            <textarea id="message">Your Message</textarea>
        </label>
        <div class="submit-button-wrapper">
            <input type="submit" valid="Send Message" />
        </div>
    </form>
</div>
```

CSS

```css
form input, textarea {
  padding: 5px;
  border-radius: 5px;
  width: 240px;
}

form {
  width: 250px;
  margin: 0 auto;
}

form input[type="submit"] {
  width: 250px;
}

.submit-button-wrapper {
  margin: 20px 0;
}
```



Links

Github
Twitter
Linkedin

Email

Enter your email

Message

Your Message

Submit

Notice the different variations of CSS that we're using:

- **form input** to select input fields that are inside of a form,
- **form input[type="submit"]** to select input fields inside a form that have a **type** attribute set to 'submit' (this is the submit button at the bottom).

5. Changing the page's background color

Now that we've taken care of some finer details, let's zoom out a bit and see if we can make any page-wide changes that would further enhance the aesthetics of our webpage. First of all, let's set a nice soothing background color to our page and add some physical separation between the sections of our webpage.

The HTML <hr/> element, h-r for **horizontal rule**, is a perfect candidate to visually separate sections of a page.

Adding a margin also helps create a visual separation.

For the background color, we're using a shade of blue, close to sky blue. Blue is usually a color of choice for web designers and you'll see it used quite a lot more than other colors (see Facebook, Twitter, Linkedin etc).

ADD an <hr/> element after the intro div and before the bout-grid div

```
<div class="container">

    <div class="intro">
        <img class="profile-picture" src="images/profile-picture.jpeg" />
        <h1>FirstName LastName</h1>
        <h3>Web Developer At XYZCompany</h3>
        <p> <i> {{ Pause and ponder in the silence }} </i> </p>
    </div>

    <hr/>

    <div class="about-grid">

        <div class="i-am">
            <h3>I am</h3>
            <ul class="about-list">
                <li>An engineer</li>
                <li>A reader</li>
                <li>A hobby cook</li>
            </ul>
        </div>
```

CSS

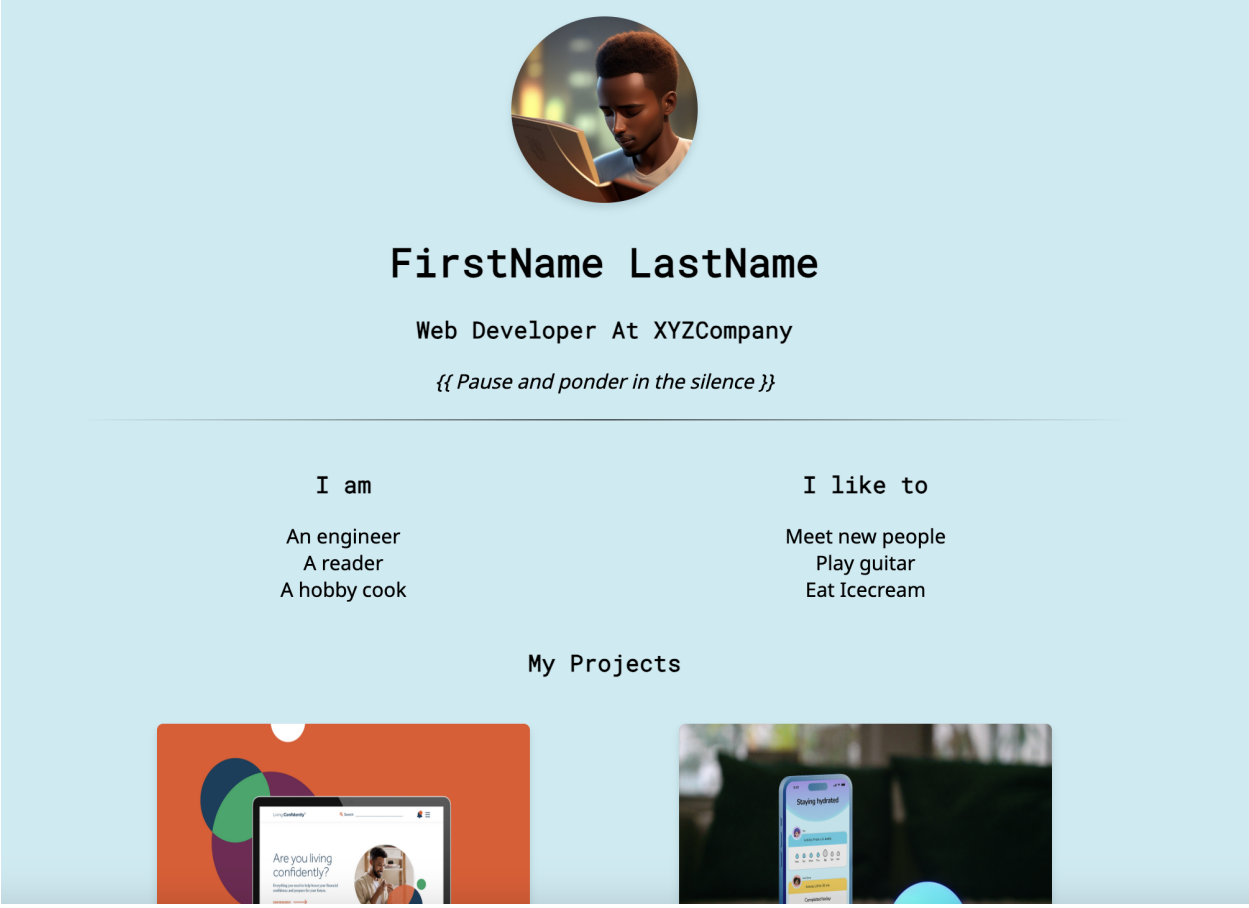
```
body {
  background-color: #caebf2;
}

.container > div {
  margin: 20px auto;
}

hr {
  border: 0;
  height: 1px;
  background-image: linear-gradient(to right, rgba(0, 0, 0, 0), rgba(0, 0, 0, 0.75), rgba(0, 0, 0, 0));
}
```

We will also style the <hr/> element by using the **background-image: linear-gradient()** CSS property.

We pass four values to this function: **direction**, **start**, **middle** and **end color (and opacity)**.

For Opacity : It starts off with 0% opacity (transparent), goes to 75% in the middle and goes back to 0% towards the end.

The other interesting function here is **rgba(red, green, blue, alpha)**, where red, green and blue are the color intensities (that is, darkness. So 0 is no color), and **alpha is the transparency/opacity setter**.



6. Changing the font color

While black is the default font color on the web, a slightly different shade doesn't hurt. However, it's important to keep in mind that as web developers, we need to make sure that the contrast between the text and the background color is maintained in order to ensure readability and accessibility.

CSS

```css
body {
  color: #001f3f;
}
```

FirstName LastName

Web Developer At XYZCompany

{{ Pause and ponder in the silence }}

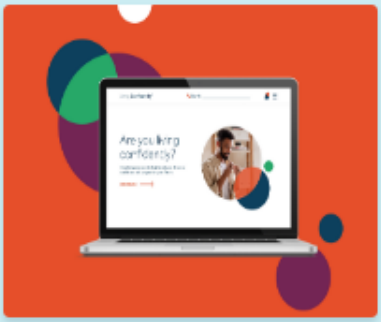
I am

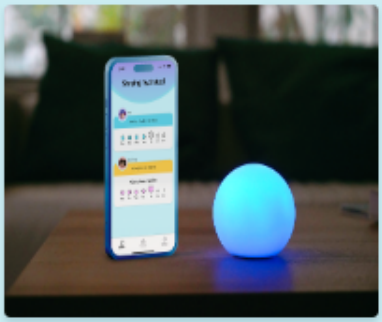
An engineer
A reader
A hobby cook

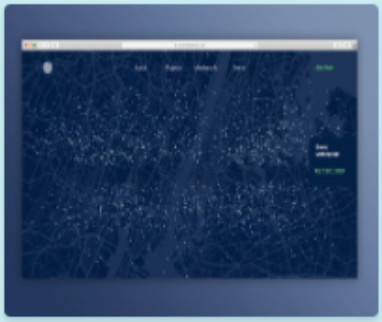
I like to

Meet new people
Play guitar
Eat Icecream

My Projects







Links

Github
Twitter
Linkedin

Email

Enter your email

Message

Your Message

Submit