Exercise 1.7: AI Tools

Coolors: Used to refine portfolio color palette. The site's contrast checker and colorblindness tool allowed me to select more accessible colors. Additionally, I created palettes for the color change animation in my Mt. Rainier SVG.

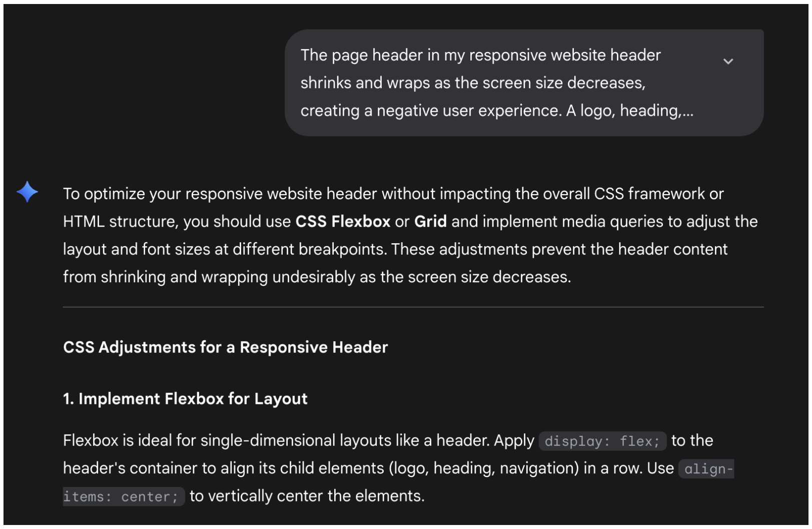
ColorSpace: Used to find complementary colors to my original color palette. It also helped me to quickly identify specific colors to animate in my Mt. Rainier SVG.

Vecteezy: Used to find a vector for the "About" page.

SVG converter: Used to convert the downloaded Vecteezy vector.

Gemini Prompt 1: The page header in my responsive website header shrinks and wraps as the screen size decreases, creating a negative user experience. A logo, heading, and navigation bar are required items. What adjustments to CSS should be made to optimize the responsiveness without impacting the overall CSS framework and HTML structure?

Gemini Prompt 1 Response: This worked well at the onset but I didn't like the elements (logo, heading, nav) evenly spaced apart. It didn't address the navigation list items and also caused the default font to shrink. It also used "min-font-size," which is invalid.



The page header in my responsive website header shrinks and wraps as the screen size decreases, creating a negative user experience. A logo, heading,...

To optimize your responsive website header without impacting the overall CSS framework or HTML structure, you should use **CSS Flexbox** or **Grid** and implement media queries to adjust the layout and font sizes at different breakpoints. These adjustments prevent the header content from shrinking and wrapping undesirably as the screen size decreases.

CSS Adjustments for a Responsive Header

1. Implement Flexbox for Layout

Flexbox is ideal for single-dimensional layouts like a header. Apply `display: flex;` to the header's container to align its child elements (logo, heading, navigation) in a row. Use `align-items: center;` to vertically center the elements.

To optimize your responsive website header without impacting the overall CSS framework or HTML structure, you should use **CSS Flexbox** or **Grid** and implement media queries to adjust the layout and font sizes at different breakpoints. These adjustments prevent the header content from shrinking and wrapping undesirably as the screen size decreases.

CSS Adjustments for a Responsive Header

1. Implement Flexbox for Layout

Flexbox is ideal for single-dimensional layouts like a header. Apply `display: flex;` to the header's container to align its child elements (logo, heading, navigation) in a row. Use `align-items: center;` to vertically center the elements.

CSS

```css
.header-container {
  display: flex;
  justify-content: space-between; /* Distributes items with space between them */
  align-items: center; /* Vertically centers the items */
  padding: 10px 20px;
}
.logo, .navigation, .heading {
  /* Set a minimum width or flex-shrink property if needed */
  flex-shrink: 0;
}
```

- `justify-content: space-between;` spaces the logo on the left, the heading in the center, and the navigation on the right.
- `flex-shrink: 0;` on the child elements prevents them from shrinking below their intrinsic size.

2. Use Media Queries for Breakpoints

Media queries are essential for adapting the layout at different screen sizes. Define breakpoints where the header's layout needs to change.

CSS

```css
/* For larger screens (desktops and tablets) */
.header-container {
  /* Flexbox properties as defined above */
}
/* Adjustments for medium screens (e.g., tablets) */
@media (max-width: 768px) {
  .header-container {
    flex-direction: column; /* Stacks items vertically */
    text-align: center;
  }
  .navigation {
    width: 100%; /* Makes navigation full-width */
    margin-top: 10px;
  }
}
/* Adjustments for small screens (e.g., mobile phones) */
@media (max-width: 480px) {
  .header-container {
    flex-direction: column; /* Ensures content remains stacked */
    padding: 10px;
  }
   .logo img {
    max-width: 80%; /* Adjust logo size */
  }
  .navigation a {
    font-size: 0.9em; /* Reduces font size of navigation links */
    padding: 5px;
  }
}
```

This approach allows you to redefine the flex direction, font sizes, and spacing for smaller screens, preventing the content from wrapping awkwardly.

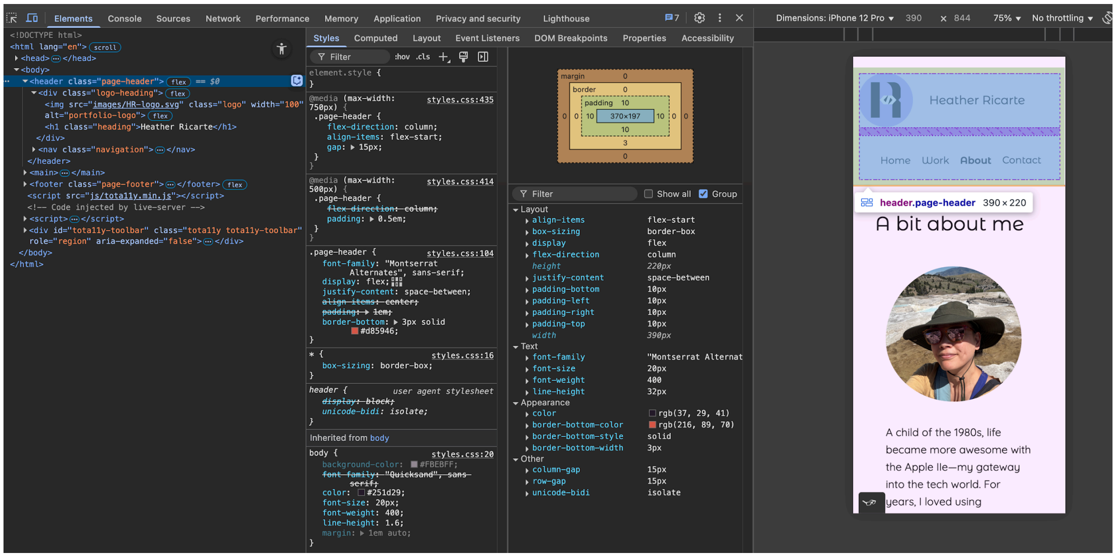
3. Optimize Font and Image Sizing

Using **relative units** like `em`, `rem`, and `vw` for font sizes helps them scale with the viewport, while `max-width: 100%;` for images ensures they don't overflow their containers.

```
CSS

.heading {
  font-size: 2.5vw; /* Scales with viewport width */
  min-font-size: 1.5em; /* Prevents font from becoming too small */
}
.logo img {
  max-width: 100%;
  height: auto;
}
```

By applying these three types of CSS adjustments, you can create a header that adapts gracefully to different screen sizes, providing a positive user experience without altering the existing HTML or CSS framework.



```
<!DOCTYPE html>
<html lang="en"> scroll
 <head> ··· </head>
 <body>
  <header class="page-header"> flex == $0
   <div class="logo-heading"> flex
     <img src="images/HR-logo.svg" class="logo" width="100"
     alt="portfolio-logo"> flex
     <h1 class="heading">Heather Ricarte</h1>
    </div>
   <nav class="navigation"> ··· </nav>
  </header>
  <main> ··· </main>
  <footer class="page-footer"> ··· </footer> flex
  <script src="js/tota11y.min.js"></script>
  <!-- Code injected by live-server -->
  <script> ··· </script>
  <div id="tota11y-toolbar" class="tota11y tota11y-toolbar"
  role="region" aria-expanded="false"> ··· </div>
 </body>
</html>
```

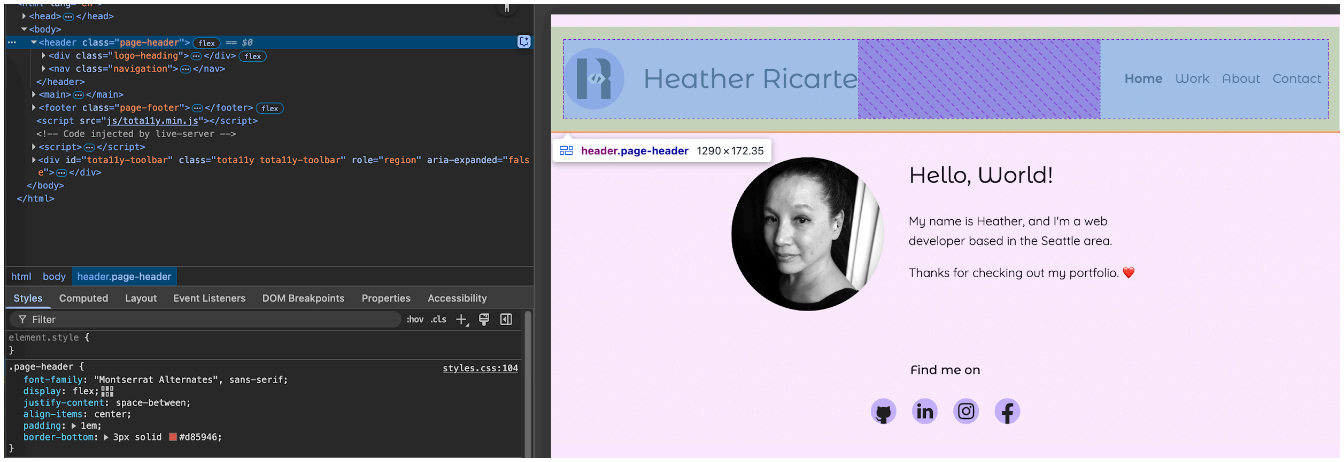
```
element.style {
}
@media (max-width:                 styles.css:435
750px) {
 .page-header {
    flex-direction: column;
    align-items: flex-start;
    gap: ▸ 15px;
 }
}
@media (max-width:                 styles.css:414
500px) {
 .page-header {
    flex-direction: column;
    padding: ▸ 0.5em;
 }
}
.page-header {                     styles.css:104
   font-family: "Montserrat
         Alternates", sans-serif;
   display: flex;
   justify-content: space-between;
   align-items: center;
   padding: ▸ 1em;
   border-bottom: ▸ 3px solid
         #d85946;
}
* {                                 styles.css:16
   box-sizing: border-box;
}
header {               user agent stylesheet
   display: block;
   unicode-bidi: isolate;
}
Inherited from body
body {                              styles.css:20
   background-color: #FBEBFF;
   font-family: "Quicksand", sans-
         serif;
   color: #251d29;
   font-size: 20px;
   font-weight: 400;
   line-height: 1.6;
   margin: ▸ 1em auto;
}
```

```
Layout
  align-items          flex-start
  box-sizing           border-box
  display              flex
  flex-direction       column
  height               220px
  justify-content      space-between
  padding-bottom       10px
  padding-left         10px
  padding-right        10px
  padding-top          10px
  width                390px
Text
  font-family          "Montserrat Alternat
  font-size            20px
  font-weight          400
  line-height          32px
Appearance
  color                rgb(37, 29, 41)
  border-bottom-color  rgb(216, 89, 70)
  border-bottom-style  solid
  border-bottom-width  3px
Other
  column-gap           15px
  row-gap              15px
  unicode-bidi         isolate
```

A bit about me

A child of the 1980s, life became more awesome with the Apple IIe—my gateway into the tech world. For years, I loved using

The web inspector helped me to make adjustments more effectively than tabbing between open VSC editors.



```
<head>⋯</head>
<body>
  <header class="page-header"> flex == $0
    <div class="logo-heading">⋯</div> flex
    <nav class="navigation">⋯</nav>
  </header>
  <main>⋯</main>
  <footer class="page-footer">⋯</footer> flex
  <script src="js/tota11y.min.js"></script>
  <!-- Code injected by live-server -->
  <script>⋯</script>
  <div id="tota11y-toolbar" class="tota11y tota11y-toolbar" role="region" aria-expanded="false">⋯</div>
</body>
</html>
```

```
element.style {
}
.page-header {                                        styles.css:104
  font-family: "Montserrat Alternates", sans-serif;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: ▸ 1em;
  border-bottom: ▸ 3px solid #d85946;
}
```

Hello, World!

My name is Heather, and I'm a web developer based in the Seattle area.

Thanks for checking out my portfolio. ❤️

Find me on

Gemini Prompt 2: Is there a CSS property for minimum font size? *(based on prompt 1's response and lack of results for "min-font-size". The first prompt also caused the header font to shrink.*

Gemini Prompt 2 Response: This seems to work well for my needs. I don't really understand viewport with regards to font but will read more about it. I initially used the suggested values but then adjusted them to my preferences.

There is no direct CSS property called `min-font-size`. However, you can achieve the same effect using a combination of modern CSS functions and units. The most effective way to set a minimum font size is by using the **`clamp()`** or **`max()`** CSS functions.

Using `clamp()` for Fluid Typography

The `clamp()` function is the most elegant solution for this problem. It allows you to set a minimum, preferred, and maximum value for a CSS property. For font size, it creates "fluid typography" that scales with the viewport but never goes below a set minimum or above a set maximum.

The syntax is `clamp(min, preferred, max)`.

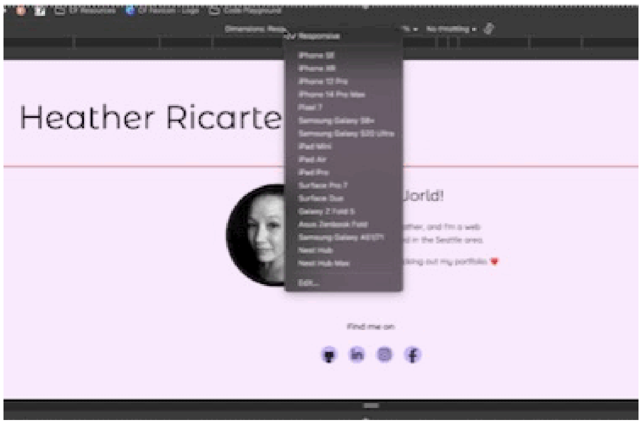
- **min:** The minimum font size. The browser will not render the font smaller than this value.
- **preferred:** The ideal or responsive font size.[1] This is usually a viewport-relative unit like `vw` (viewport width).
- **max:** The maximum font size. The browser will not render the font larger than this value.

Example:

CSS

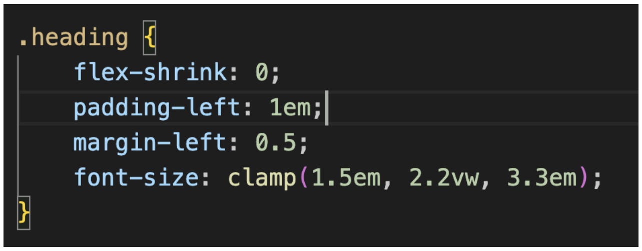
```css
h1 {
  font-size: clamp(2rem, 5vw, 4rem);
}
```

Using suggested <clamp> values →



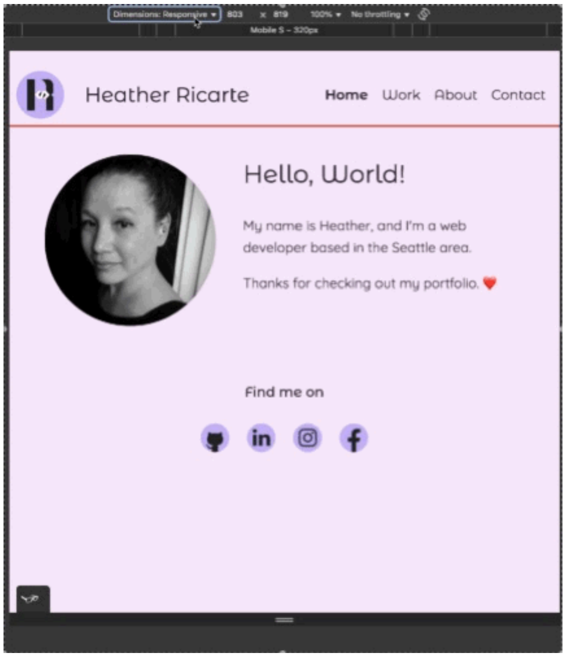
In this example, the `h1` will:

- Never be smaller than **2rem**.
- Never be larger than **4rem**.
- Fluidly scale based on **5vw** when the viewport size is between the min and max limits.



```css
.heading {
    flex-shrink: 0;
    padding-left: 1em;
    margin-left: 0.5;
    font-size: clamp(1.5em, 2.2vw, 3.3em);
}
```

Modifying the <clamp> values to yield a more responsive layout →



Hello, World!

My name is Heather, and I'm a web developer based in the Seattle area.

Thanks for checking out my portfolio. ❤

Using `max()` for a Simple Minimum

The `max()` function is a simpler alternative if you only need a minimum size and no maximum. It selects the largest (most positive) value from a list of comma-separated expressions.[2] By combining a fixed unit like `px` or `rem` with a flexible unit like `vw`, you can ensure the font size is never smaller than the fixed value.

The syntax is `max(val1, val2, ...)`

Example:

CSS

```css
p {
  font-size: max(16px, 1.2vw);
}
```

In this case, the browser will always choose the larger value between **16px** and **1.2% of the viewport width**. This guarantees that the paragraph font size will be at least 16px, preventing it from becoming unreadably small on narrow screens.

Gemini Prompt 3: The logo, title, and navigation bar in my website header are in a flexbox and styled as "justify-content: space between," as generated in Flexbox properties above. I do not want the items to be spaced so far apart, though.

I want the logo to be left-aligned, the title to be 40px left-padded from the logo, and for the navigation bar to be right-aligned. The elements need to be responsive to smaller screens without shrinking or wrapping at awkward breakpoints. What CSS styles will help achieve my needs?

Gemini Prompt 3 Response: The results were mixed but closer to what I wanted initially with Prompt 1.

To achieve your desired header layout, you'll need to adjust the CSS Flexbox properties to control alignment and spacing, and then use media queries to ensure responsiveness without shrinking or wrapping. The key is to group your logo and title together using a nested flexbox container.

CSS Adjustments for Specific Spacing

1. **Wrap the Logo and Title:** To keep the logo and title together and apply a specific space between them, place them inside their own `<div>` container. This container will be a flex item within the main header flexbox.
2. **Apply Flexbox to the New Container:** Use `display: flex;` on this new container to align the logo and title.
3. **Control the Main Header Flexbox**: The main header container will use `justify-content: space-between;` as you initially had. Its flex items will be the new logo/title container and the navigation bar. This will push the logo/title group to the left and the navigation bar to the right.

Example HTML Structure

HTML

```html
<header class="header-container">
  <div class="logo-and-title">
    <img src="logo.png" alt="Logo" class="logo">
    <h1 class="heading">Website Title</h1>
  </div>
  <nav class="navigation">
    <ul>
```

```
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
    </ul>
  </nav>
</header>
```

CSS Styling

Here are the specific CSS styles to implement this:

1. **Main Header Container:** This container holds the logo/title group and the navigation.
    - `display: flex;`
    - `justify-content: space-between;`
    - `align-items: center;` (to vertically center all items)
    - `padding: 20px;`
2. **Logo and Title Container:** This is the nested flexbox that controls the spacing between the logo and the title.
    - `display: flex;`
    - `align-items: center;`
    - `gap: 40px;` or `padding-left: 40px;` on the heading.
3. **Prevent Shrinking:** Add `flex-shrink: 0;` to all elements inside the header to prevent them from shrinking below their content size. This is crucial for avoiding the "awkward shrinking" issue.

Final CSS

CSS

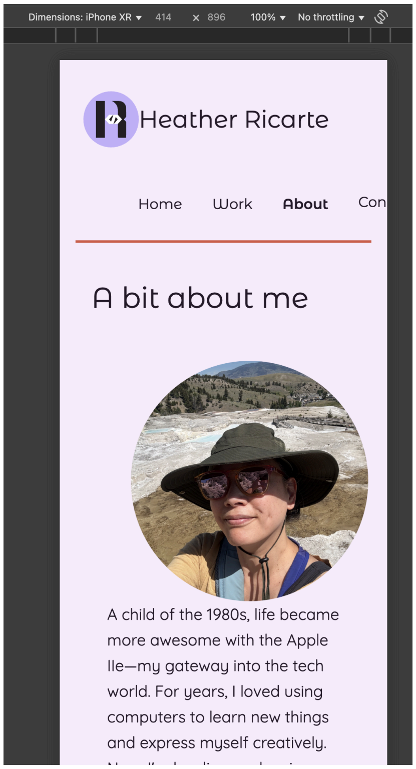
```
.header-container {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 20px;
}
.logo-and-title {
  display: flex;
  align-items: center;
  flex-shrink: 0;
}
.logo {
  flex-shrink: 0;
  height: 50px; /* Adjust as needed */
}
.heading {
  margin-left: 40px; /* Creates a 40px space to the left of the heading */
  flex-shrink: 0;
  font-size: clamp(1.5rem, 3vw, 2.5rem);
}
.navigation {
  flex-shrink: 0;
}
/* Optional: Additional styling for navigation links */
.navigation ul {
  list-style: none;
  display: flex;
  gap: 20px;
}
```

Addressing Responsiveness

To prevent wrapping and ensure the layout adapts gracefully, use media queries to change the layout for smaller screens. A common approach is to switch from a horizontal layout to a vertical one.

CSS

```css
@media (max-width: 768px) {
  .header-container {
    flex-direction: column; /* Stack items vertically on smaller screens */
    align-items: flex-start; /* Left-align stacked items */
    gap: 15px; /* Add some space between stacked items */
  }
  .logo-and-title {
    flex-direction: row; /* Keep logo and title in a row */
    align-items: center;
    gap: 15px; /* Adjust spacing for smaller screens */
  }
  .heading {
    margin-left: 0; /* Remove the 40px margin in the mobile view */
  }
  .navigation {
    width: 100%; /* Make navigation full-width */
  }
}
```



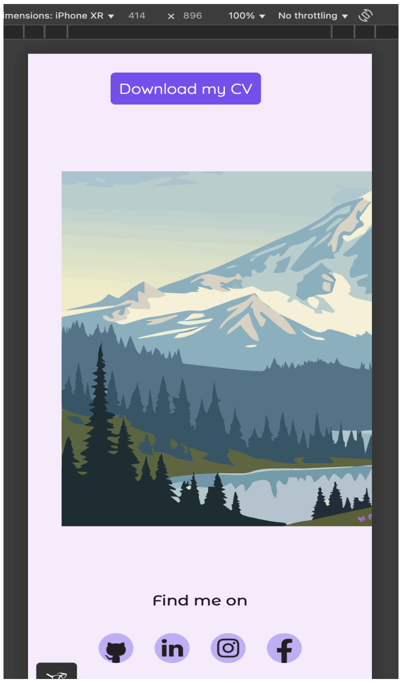
A bit about me

A child of the 1980s, life became more awesome with the Apple IIe—my gateway into the tech world. For years, I loved using computers to learn new things and express myself creatively.

This ensures the elements don't shrink or wrap awkwardly. The logo and title will remain together, while the main header layout switches to a more mobile-friendly, stacked design.

Gemini Prompt 4: What steps in CSS to make an .svg image responsive to smaller screens? *(based on svg getting cut-off)*

Gemini Prompt 4 Response: I omitted Step 3 after getting undesirable results.



To make an SVG image responsive to smaller screens, you should follow these steps in your CSS:

1. Set the SVG as a Block-Level Element

First, ensure your SVG is treated as a block-level element. This allows you to control its width and height more easily.

CSS

```css
svg {
  display: block;
}
```

---

2. Control Width and Height

Use `width` or `max-width` to make the SVG scale with its container. Use `height: auto` to maintain the aspect ratio. This is the most critical step for responsiveness.

CSS

```css
svg {
  width: 100%; /* The SVG will scale to the width of its parent container */
  height: auto; /* The height will adjust automatically to maintain the aspect ratio */
}
```

Alternatively, you can set a `max-width` if you don't want the SVG to grow beyond a certain size on large screens.

CSS

```css
svg {
  max-width: 600px;
  height: auto;
}
```

---

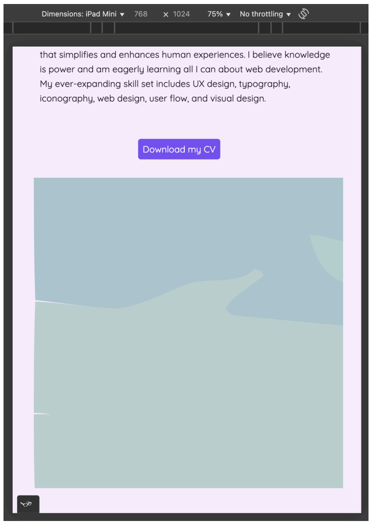
3. Use `viewBox` in the SVG File

This isn't a CSS step, but it's essential and works with the CSS above. The `viewBox` attribute inside the `<svg>` tag defines the coordinate system and aspect ratio of the SVG. Without it, the SVG won't scale properly. The four values are `min-x`, `min-y`, `width`, and `height`. Make sure it's set correctly in your SVG file itself.

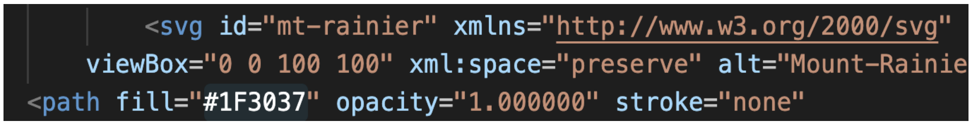
HTML

```html
<svg viewBox="0 0 100 100">
  </svg>
```

zoomed in on a portion of the original .svg. →



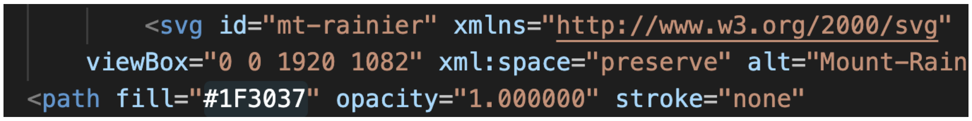
that simplifies and enhances human experiences. I believe knowledge is power and am eagerly learning all I can about web development. My ever-expanding skill set includes UX design, typography, iconography, web design, user flow, and visual design.



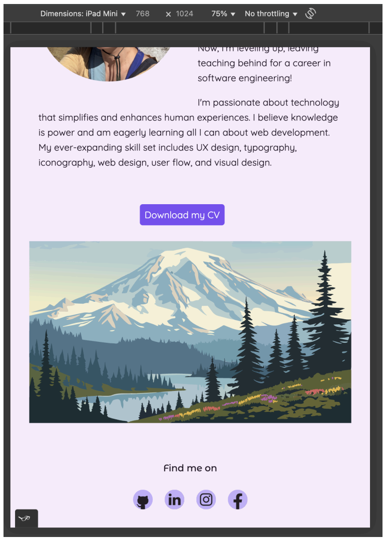
```
      <svg id="mt-rainier" xmlns="http://www.w3.org/2000/svg"
    viewBox="0 0 100 100" xml:space="preserve" alt="Mount-Rainie
<path fill="#1F3037" opacity="1.000000" stroke="none"
```

The browser uses the `viewBox` along with the CSS `width` and `height` properties to scale the SVG content appropriate.

Skipping step 3 yielded the results I sought. →



```
      <svg id="mt-rainier" xmlns="http://www.w3.org/2000/svg"
    viewBox="0 0 1920 1082" xml:space="preserve" alt="Mount-Rain
<path fill="#1F3037" opacity="1.000000" stroke="none"
```



Now, I'm leveling up, leaving teaching behind for a career in software engineering!

I'm passionate about technology that simplifies and enhances human experiences. I believe knowledge is power and am eagerly learning all I can about web development. My ever-expanding skill set includes UX design, typography, iconography, web design, user flow, and visual design.