**Create A Portfolio Website: Part Two -** [Eric Hu](https://huericnan.medium.com/?source=post_page-----896b3fc2011--------------------------------) Feb 9, 2022

Last time, we created a portfolio website with only HTML. In this article, we are going to make our portfolio website look more appealing with CSS.

**Create A Responsive Layout**

Let’s first start by designing a responsive layout for our portfolio. Remember that a responsive layout should have two key components, a container and a grid system.

*You can download the source code for this tutorial here:*

[*Download the Source Code*](https://www.ericsdevblog.com/index.php/beginners-roadmap-to-web-development/)

First, we should have a fluid container that changes width as you resize your browser. We usually set a maximum width on large screens so that the container does not stretch to the edge. However, on smaller screens, we usually set its width to 100% to utilize as much space as possible.

This is what we can do:

<body>  
 <div class="container"> </div>  
 </body>/\* Small Screen \*/  
.container {  
 max-width: 100%;  
 display: flex;  
 flex-wrap: wrap;  
}/\* Large Screen \*/  
@media only screen and (min-width: 1200px) {  
 .container {  
 max-width: 1140px;  
 margin: auto;  
 display: flex;  
 flex-wrap: wrap;  
 }  
}

To keep things simple, here we only have one breakpoint. If the screen width is less than 1200px, the container stretches to its edge. If the screen is larger than 1200px, the maximum width of the container will be set to 1140px. Notice that in the second scenario, we defined margin: auto;, this is to make sure that the container is always centred.

Shape, rectangle

Description automatically generated

Small Screen

A picture containing graphical user interface

Description automatically generated

Medium Screen

Shape, rectangle

Description automatically generated

Large Screen

Second, we need to have different grid systems for the small screen (mobile phone), medium screen (tablet) and large screen (desktop). Each grid system contains 12 columns, and different elements could take up different columns on different screens. This could be realized by either the [regular grid system](https://www.ericsdevblog.com/index.php/2021/07/css-basics-4-grid-system-and-responsive-design/) or the [flexbox](https://www.ericsdevblog.com/index.php/2021/08/css-basics-5-some-advanced-topics/).

<body>  
 <div class="container">  
 <div class="col-6 col-s-8">Div1</div>  
 <div class="col-6 col-s-4">Div2</div> <div class="col-2 col-s-10">Div3</div>  
 <div class="col-4 col-s-2">Div4</div>  
 </div>  
 </body>/\* For mobile phones: \*/  
[class\*="col-"] {  
 width: 100%;  
}/\* For tablets: \*/  
@media only screen and (min-width: 600px) {  
 .col-s-1 {width: 8.33%;}  
 .col-s-2 {width: 16.66%;}  
 .col-s-3 {width: 25%;}  
 .col-s-4 {width: 33.33%;}  
 .col-s-5 {width: 41.66%;}  
 .col-s-6 {width: 50%;}  
 .col-s-7 {width: 58.33%;}  
 .col-s-8 {width: 66.66%;}  
 .col-s-9 {width: 75%;}  
 .col-s-10 {width: 83.33%;}  
 .col-s-11 {width: 91.66%;}  
 .col-s-12 {width: 100%;}  
}/\* For desktop: \*/  
@media only screen and (min-width: 768px) {  
 .col-1 {width: 8.33%;}  
 .col-2 {width: 16.66%;}  
 .col-3 {width: 25%;}  
 .col-4 {width: 33.33%;}  
 .col-5 {width: 41.66%;}  
 .col-6 {width: 50%;}  
 .col-7 {width: 58.33%;}  
 .col-8 {width: 66.66%;}  
 .col-9 {width: 75%;}  
 .col-10 {width: 83.33%;}  
 .col-11 {width: 91.66%;}  
 .col-12 {width: 100%;}  
}

This is the exact same code we talked about before in this article: [CSS Basics #4: Grid System and Responsive Design](https://www.ericsdevblog.com/index.php/2021/07/css-basics-4-grid-system-and-responsive-design/). Let’s see how it works.

A picture containing chart

Description automatically generated

Small Screen

Graphical user interface, application

Description automatically generated

Medium Screen

A picture containing chart

Description automatically generated

Large Screen

As a guide, the portfolio page we design should look like this:

**Small screen:**

[small-screen-layout](https://www.ericsdevblog.com/wp-content/uploads/2021/07/small-screen-layout.png)

**Medium screen:**

[medium-screen-layout](https://www.ericsdevblog.com/wp-content/uploads/2021/07/medium-screen-layout.png)

**Large screen:**

[large-screen-layout](https://www.ericsdevblog.com/wp-content/uploads/2021/07/large-screen-layout.png)

Next, before we start designing the webpage, let’s first choose the colour scheme and the font we are going to use. This might be difficult if you are not a professional designer, but don’t worry, there are several tools that can help us.

To find a nice-looking colour scheme, we can use this website: <https://coolors.co/palettes/trending>, which offers hundreds of different colour palettes with colour codes. As for the fonts, the common practice is to use [Google Fonts](https://fonts.google.com/). We can select any font and use it in our CSS file.

Application

Description automatically generated

Google Fonts

Here are the fonts I choose:

@import url("https://fonts.googleapis.com/css2?family=Comfortaa:wght@700&family=Courier+Prime:ital,wght@0,400;0,700;1,400;1,700&family=Crimson+Text:ital,wght@0,400;0,600;0,700;1,400;1,600;1,700&family=Poppins:ital,wght@0,200;0,300;0,400;0,500;0,600;0,700;1,200;1,300;1,400;1,500;1,600;1,700&display=swap");

This line of code imports all the necessary fonts we are going to use. Remember you should put it at the **top** of the CSS file.

h1 {  
 font-family: "Comfortaa", cursive;  
}h2,  
h3,  
h4 {  
 font-family: "Crimson Text", serif;  
}p,  
a {  
 font-family: "Poppins", sans-serif;  
}

**Navigation Bar**

When designing web pages, we should always follow the rule of “mobile-first”. It means we should always start by designing the small screen layout.

Let’s start with the navigation bar.

First, we wrap <header> and <nav> blocks inside a <div> block. The <div> block takes 12 columns on all devices.

<div class="col-12 col-s-12 header-and-nav">  
 <header>  
 ...  
 </header>  
 <nav>  
 ...  
 </nav>  
</div>

Next, on small devices, we want <header> to stay on top of <nav>, and on larger screens, we want to put <header> on the left side and <nav> on the right side.

To achieve this, we need to make sure that the <div> element is also a flexbox container. So, this is what we can do:

.header-and-nav {  
 display: flex;  
 flex-wrap: wrap;  
}<div class="col-12 col-s-12 header-and-nav">  
 <header class="col-4 col-s-4"> </header>  
 <nav class="col-8 col-s-8"> </nav>  
</div>

Navbar on Small Screen

A picture containing graphical user interface

Description automatically generated

This is the navbar we have. Next, since we only care about small devices right now, we need to centre the header and the nav list. This is what we can do:

<!--Header and Navigation Bar-->  
<div class="col-12 col-s-12 header-and-nav">  
 <header class="col-4 col-s-4 header">  
 <h1>Eric's Portfolio</h1>  
 </header>  
 <nav class="col-8 col-s-8">  
 <ul class="nav-list">  
 <li class="nav-item">  
 <a href="/portfolio.html">Home</a>  
 </li>  
 <li class="nav-item">  
 <a href="https://www.ericsdevblog.com">Blog</a>  
 </li>  
 <li class="nav-item">  
 <a href="#project">Projects</a>  
 </li>  
 </ul>  
 </nav>  
</div>.header {  
 text-align: center;  
}.nav-list {  
 text-align: center;  
}.nav-item {  
 display: inline-block;  
}.nav-item a {  
 display: inline-block;  
}

Graphical user interface, application

Description automatically generated

Finally, let’s add some colours, paddings, margins, text decorations and so on.

.header {  
 color: white;  
 text-align: center;  
}.nav-list {  
 list-style-type: none;  
 margin: 0px;  
 padding: 0px; text-align: center;  
}.nav-item {  
 display: inline-block;  
}.nav-item a {  
 display: inline-block;  
 color: white;  
 text-decoration: none;  
 padding: 27px 20px;  
}.nav-item a:hover {  
 background-color: #457b9d;  
 text-decoration: underline;  
}

Graphical user interface, text, application

Description automatically generated

Navbar on Small Screens

Finally, we need to change how the navbar looks on desktops, apparently, we don’t want everything to be centred. Instead, we want the links to align to the right side.

@media only screen and (min-width: 768px) {  
 .nav-list {  
 list-style-type: none;  
 margin: 0px;  
 padding: 0px; text-align: right;  
 }  
}



Navbar on Large Screens

**Self Introduction Section**

A picture containing text, cat, mammal, domestic cat

Description automatically generated

<!--Self Introduction Section-->  
<section class="col-12 col-s-12 self-intro">  
 <img  
 src="/frontend/images/profile-image.jpg"  
 class="col-4 col-s-4 profile-image"  
 />  
 <div class="col-8 col-s-8 self-intro-text">  
 <h2>Hello, I am XXXX. Welcome to My Portfolio.</h2>  
 <p>  
 ...  
 </p>  
 </div>  
</section>/\* Self Introduction Section \*/  
.self-intro {  
 padding: 10px;  
 display: flex;  
 flex-wrap: wrap;  
}.profile-image {  
 object-fit: cover;  
}.self-intro-text h2 {  
 text-decoration: underline;  
}@media only screen and (min-width: 768px) {  
 .self-intro-text {  
 padding: 0px 10px;  
 }  
}

**Newsletter Section**

Chart, bar chart

Description automatically generated

On small screen

Graphical user interface, application, website

Description automatically generated

On large screen

<!--Newsletter Signup Section-->  
<section class="col-12 col-s-12 newsletter-signup">  
 <h2 class="col-6 col-s-6">  
 Would you like to see more tips and tutorials on web development?  
 </h2>  
 <form class="col-6 col-s-6 newsletter-signup-form">  
 <label for="firstname">First Name:</label><br />  
 <input type="text" id="firstname" />  
 <br />  
 <label for="lastname">Last Name:</label><br />  
 <input type="text" id="lastname" />  
 <br />  
 <label for="email">Enter your email:</label><br />  
 <input type="email" id="email" name="email" />  
 <button type="submit">Submit</button>  
 </form>  
</section>/\* Newsletter Signup Section \*/  
.newsletter-signup {  
 background-color: #a8dadc;  
 padding: 10px;  
 display: flex;  
 flex-wrap: wrap;  
}.newsletter-signup h2 {  
 text-align: center;  
}.newsletter-signup-form {  
 margin: 0 auto;  
}.newsletter-signup-form label {  
 font-family: "Poppins", sans-serif;  
}.newsletter-signup-form input {  
 width: 100%;  
 padding: 12px 20px;  
 margin: 8px 0;  
 box-sizing: border-box;  
}.newsletter-signup-form button {  
 background-color: #a8dadc;  
 border: solid 3px #e63946;  
 color: #e63946;  
 width: 100%;  
 padding: 10px 0;  
 margin: 10px 0;  
 cursor: pointer;  
 font-family: "Poppins", sans-serif;  
 font-size: 1em;  
 font-weight: 600;  
}.newsletter-signup-form button:hover {  
 background-color: #e63946;  
 border: solid 3px #e63946;  
 color: white;  
}@media only screen and (min-width: 768px) {  
 .newsletter-signup {  
 background-color: #a8dadc;  
 padding: 10px;  
 display: flex;  
 flex-wrap: wrap;  
 align-items: center;  
 }  
 .newsletter-signup h2 {  
 text-align: center;  
 padding: 10px;  
 }  
}

**Skills Section**

Text, letter

Description automatically generated

<!--Skills Section-->  
<section class="col-12 col-s-12 skills-section">  
 <h2 class="col-12 col-s-12">My Skills</h2>  
 <ul class="col-8 col-s-8 skills-list">  
 <li>HTML (100%)</li>  
 <li>CSS (90%)</li>  
 <li>JavaScript (90%)</li>  
 <li>Python (90%)</li>  
 <li>PHP (100&#37;)</li>  
 <li>Java (90&#37;)</li>  
 <li>Vue.js (80&percnt;)</li>  
 <li>Django (90&percnt;)</li>  
 <li>Laravel (90&percnt;)</li>  
 </ul>  
</section>/\* Skills Section \*/  
.skills-section {  
 padding: 10px;  
 display: flex;  
 flex-wrap: wrap;  
}.skills-section h2 {  
 text-decoration: underline;  
}.skills-list {  
 column-count: 2;  
}@media only screen and (min-width: 768px) {  
 .skills-list {  
 column-count: 2;  
 margin: 0 auto;  
 }  
}

One small trick worth mentioning here is that we used column-count: 2; to split the list into two columns.

**Projects Section**

A group of kittens in a nest

Description automatically generated with low confidence

On small screen

A picture containing text, cat, screenshot

Description automatically generated

On large screen

<!--Projects Section-->  
<section class="col-12 col-s-12 porject-section" id="project">  
 <h2 class="col-12 col-s-12">My Projects</h2>  
 <div class="col-4 col-s-4">  
 <div class="project-card">  
 <h3>First Project</h3>  
 <img src="/frontend/images/p1.jpg" />  
 <p>  
 ...  
 </p>  
 </div>  
 </div> ...  
</section>/\* Project Section \*/  
.porject-section {  
 padding: 10px;  
 display: flex;  
 flex-wrap: wrap;  
 background-color: #a8dadc;  
}.porject-section h2 {  
 text-decoration: underline;  
}.project-card {  
 border: solid 2px gray;  
 border-radius: 5px;  
 padding: 5px;  
 margin: 5px;  
}.project-card img {  
 max-width: 100%;  
 object-fit: cover;  
}

**Footer**



<!--Footer-->  
<footer class="col-12 col-s-12 footer">  
 <p>Created by Eric Hu</p>  
 <p><a href="mailto:huericnan@gmail.com">huericnan@gmail.com</a></p>  
</footer>/\* Footer \*/  
.footer {  
 background-color: #1d3557;  
 color: white;  
 text-align: center;  
}.footer a {  
 color: white;  
}