Module 4: Web Design and CSS

Module 4 Objective: Learn the fundamentals of CSS and begin styling your digital portfolio.

Getting started:

Before we even start setting up our site structure, we need to set up our project directory.

As with our previous examples, we will need to use our terminal to set up the initial files and directories that you will be working with for this task.

Our first task will be to CD into our module 4 directory and set up our new folder structure.

```
$ cd TKH_Modules/Module4
```

Next, we will copy our work over from Module3 to our current directory. Please note if the source_file ends in a /, the contents of the directory are copied rather than the directory itself.

```
$ cp -R module3/ ./
```

Next, let's just finish up by creating our style sheet:

```
$ touch css/style.css
```

So now our module 3 folder should look like:

- Module3/
    - index.html
    - Resume.html
    - images/
    - CSS/
        - style.css

- js/

Next visual hierarchy: Visual hierarchy is one of the most important principles behind effective web design. This article will examine why developing a visual hierarchy is crucial on the web, the theory behind it, and how you can use some very basic exercises in your own designs to put these principles into practice.

We are going to use HTML headings to create some Visual hierarchy using font size. HTML headings are defined with the `<h1>` to `<h6>` tags.

`<h1>` defines the most important heading. `<h6>` defines the least important heading.

For example:

Heading 1
Heading 2
Heading 3
Heading 4
Heading 5
Heading 6

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Lets change our header tags to match our page content:

Index.html:

```
***more above***
<div class="about-me">
            <div class="profile">...
            </div>
            <div class="about-me-text">
                  <h2>About Me</h2>
                  <p>Now, this is a story all about how My life got
flipped-turned upside down And I'd like to take a minute Just sit right there
I'll tell you how I became the prince of a town called Bel Air</p>
            </div>
      </div>
      <div class="career">
            <div class="goals">
                  <h2>My Goals</h2>
                  <p>My career goals are making dope things on a
computer and getting paid accordingly.</p>
            </div>
            <div class="skills">
                  <h2>My Skills</h2>
                  <ul>...
        </ul>
            </div>
      </div>
      <div class="projects">
            <h2>MY PROJECTS</h2>
            <div class="project">
                  <a href="index.html"><h3>Project Title</h3></a>
                  <img src="images/screenshot.png" width="100px">
                  <p>This is my profolio site</p>
            </div>
      </div>
      <div class="contact-form">
```

```
            <h2>Contact Me!</h2>
            <form class="contact">...
            </form>
        </div>
        <div class="footer">
</div>
***more below***
```

Resume.html:

```
***more above***
<div class="jobs">
            <h2>MY EXPERIENCE</h2>
            <div class="job">
                <h3>CTO at The Knowledge House</h3>
                <p class="date">2014-Present</p>
                <p class="desc">Along with my co-founder I help develop and implement a curriculum of digital learning, entrepreneurship, and advanced technical career skills for young adults from underserved communities in NYC.</p>
                <ul class="duties">...
                </ul>
            </div>
            <div class="job">
                <h3>Director Of Product at CoC.org</h3>
                <p class="date">2015-2017</p>
                <p class="desc">Managed all technology development, platform upgrades and new product development for all technologies associated with the ColorOfChange advocacy platform</p>
                <ul class="duties">...
                </ul>
            </div>
        </div>
        <div class="education">
            <h3>CUNY BARUCH</h3>
            <p>Majored in Business Admin</p>
        </div>
***more below***
```

What is CSS: CSS stands for "Cascading Style Sheet." Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML.

For example:

```
body{
        min-height: 100vh;
}
```

Using an external style sheet: With an external style sheet, you can change the look of an entire website by changing just one file, in the case of our project that will be our style.css file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

For example:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Now let's link our stylesheet in both our index.html and our resume.html files:

```
<!DOCTYPE html>
```

```
<html>
<head>
        <title>Joe's Portfolio</title>
        <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
<body>
***more below***
```

CSS Custom Properties (Variables): The var() function can be used to insert the value of a custom property.

Variables in CSS should be declared within a CSS selector that defines its scope. For a global scope, you can use either the: root or the body selector.

The variable name must begin with two dashes (--) and is case sensitive!

For example:

```
:root {
  --main-bg-color: coral;
  --main-txt-color: blue;
  --main-padding: 15px;
}

#div1 {
  background-color: var(--main-bg-color);
  color: var(--main-txt-color);
  padding: var(--main-padding);
}

#div2 {
  background-color: var(--main-bg-color);
  color: var(--main-txt-color);
  padding: var(--main-padding);
}
```

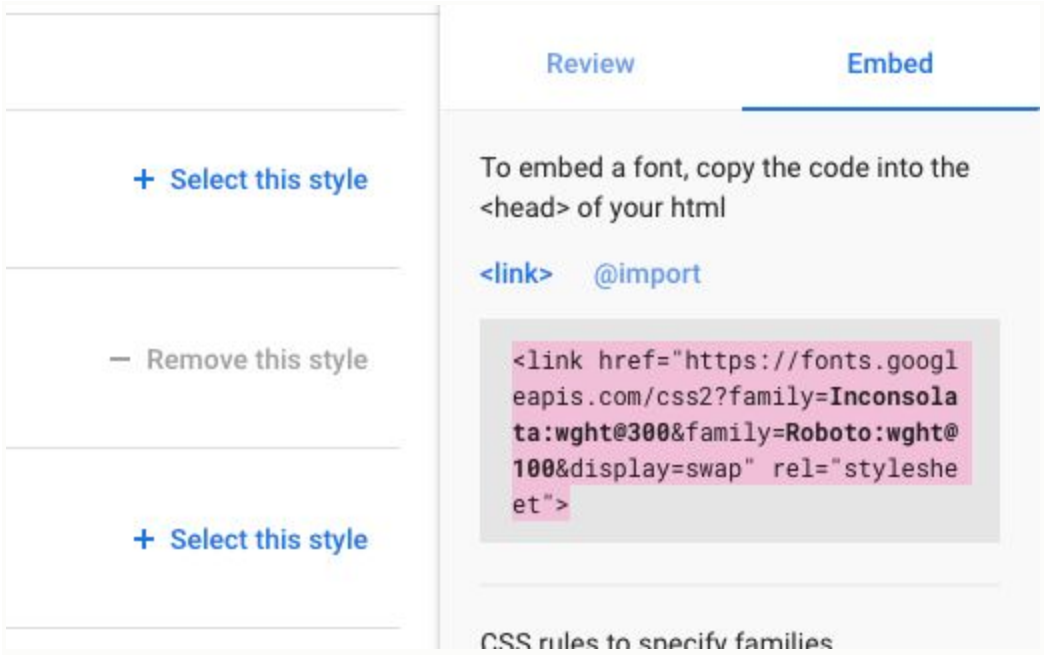
Let's add the colors for our website as css variables to our style.css:

```
:root {
```

```
  --first-color: #dbe4e8;
  --second-color: #f5f5f5;
  --third-color: #990099;
  --fourth-color: black;
}
```

Now let's pick a font from Google Fonts, install it, and create a CSS variable for our chosen font

Select your font and copy the link here:



Add the link to the head of both your index.html and resume.html files:

```
<!DOCTYPE html>
<html>
<head>
        <title>Joe's Portfolio</title>

        <link
href="https://fonts.googleapis.com/css2?family=Inconsolata:wght@300
```

```
&family=Roboto:wght@100&display=swap" rel="stylesheet">
        <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
***more below***
```

Now add the font we installed as css variable:

```
:root {
  --first-color: #dbe4e8;
  --second-color: #f5f5f5;
  --third-color: #990099;
  --forth-color: black;
  --font:'Inconsolata';
  }
```

Lastly, we are going to add a little texture to our webpage by adding a box-shadow property to some of our HTML elements. We will set a CSS variable for our box-shadow property value.

```
:root {
  --first-color: #dbe4e8;
  --second-color: #f5f5f5;
  --third-color: #990099;
  --forth-color: black;
  --font:'Inconsolata';
  --shadow: 0 3px 6px rgba(0,0,0,0.16), 0 3px 6px rgba(0,0,0,0.23);
}
```

How to style your HTML using a selector: CSS selectors are used to "find" (or select) the HTML elements you want to style.

You can style an element with the element selector which selects HTML elements based on the element name.

For example:

```
p {
  text-align: ;
  color: red;
}
```

Now let's add some styles to our responsive styles to ensure our website looks great on all screens. Please note we use the responsive CSS Units:

- em(Relative to the font-size of the element)
    - (2em means 2 times the size of the current font))
- vw(Relative to 1% of the width of the viewport*) :

```
***more above***
/* Makes Font sizes respond to the device width to look great on all screens.*/
html { font-size: calc(1em + 1vw) }
/* button and input elements for some reason don't inherit their parent's font-size so let's set it below*/
button { font-size: inherit }
input { font-size: inherit }
```

One of the most common selectors is the class selector which selects HTML elements with a specific class attribute. To select elements with a specific class, write a period (.) character, followed by the class name.

For example:

```
.center {
  text-align: center;
  color: red;
}
```

We've added a few classes as we built our website. Let’s add them to our style.css:

```
/*banner*/
.banner{}

.title{}


/*about me*/
.about-me{}
.profile{}
.profile-pic{}
.about-me-text{}

/*career*/
.career{}
.goals{}
.skills{}
/*projects*/
.projects{}
.project{}

/*contact*/
.contact-form{}
.contact{}
/*form styles*/
Input{}
/*You can specify that a specific input element should be affected by attribute. */
input[type=button],{}

/*resume*/
.jobs{}
.job{}
```

```
/*education*/
.education{}

/*footer*/
.footer{}
```

You can also specify that only specific HTML elements should be affected by a class.

For example:

```
p.center {
  text-align: center;
  color: red;
}
```

You can also specify styles for children (or nested elements) of HTML elements affected by a class.

For example:

```
.navbar li {
  display: block;
  margin: 16px;
}
```

Lets add selectors for elements we want affected by a specific class:

banner:

```
***more above***
/*banner*/
.banner{}
```

```
.title{}
.banner ul {}
.banner li {}
***more below***
```

about:

```
***more above***
/*about me*/
.about-me{}
.profile{}
.profile-pic{}
.about-me-text{}
/*social media*/
.social-media li {}
.social-media ul {}
***more below***
```

HTML elements can also refer to more than one class.

For example:

```
<p class="center large">This paragraph refers to two classes.</p>
```

Using CSS Grid: CSS Grid Layout is the most powerful layout system available in CSS. It is a 2-dimensional system, meaning it can handle both columns and rows, unlike flexbox which is largely a 1-dimensional system. You work with Grid Layout by applying CSS rules both to a parent element (which becomes the Grid Container) and to that element's children (which become Grid Items).

CSS grid allows you to create fluid width columns that break into more or fewer columns as space is available, with no media queries!

For example:

```
grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  /* This is better for small screens, once min() is better supported */
  /* grid-template-columns: repeat(auto-fill, minmax(min(200px, 100%), 1fr)); */
  grid-gap: 1rem;
  /* This is the standardized property now, but has slightly less support */
  /* gap: 1rem */
}
```

Let's add the CSS grid to the body of our website. Please note display: grid | inline-grid defines the element as a grid container and establishes a new grid formatting context for its contents.

```
:root {
  --first-color: #dbe4e8;
  --second-color: #f5f5f5;
  --third-color: #990099;
  --forth-color: black;
  --font:'Inconsolata';
  --shadow: 0 3px 6px rgba(0,0,0,0.16), 0 3px 6px rgba(0,0,0,0.23);
}

html { font-size: calc(1em + 1vw) }
button { font-size: inherit }
input { font-size: inherit }

body{
        font-family: var(--font);
        margin: 0;
        background-color: var(--first-color);
        padding-top: 4rem;
        display: grid;
/* justify-items is used to center grid i vertically */
        justify-items:center;
}

***more below***
```

Using Flexbox: The Flexbox Layout (Flexible Box) module (a W3C Candidate Recommendation as of October 2017) aims at providing a more efficient way to layout, align and distribute space among items in a container, even when their size is unknown and/or dynamic (thus the word “flex”).
The main idea behind the flex layout is to give the container the ability to alter its items’ width/height (and order) to best fill the available space (mostly to accommodate all kinds of display devices and screen sizes). A flex container expands items to fill available free space or shrinks them to prevent overflow.

Most importantly, the flexbox layout is direction-agnostic as opposed to the regular layouts (block which is vertically-based and inline which is horizontally-based). While those work well for pages, they lack flexibility (no pun intended) to support large or complex applications (especially when it comes to orientation changing, resizing, stretching, shrinking, etc.)

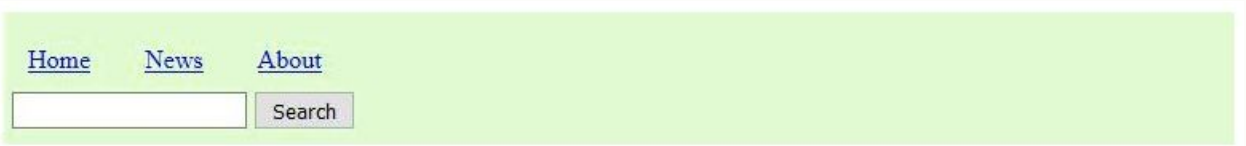
For example:

```
.topheader{
    display: flex; /* turn topheader into flex container! */
    background: #E3FFD2;
    padding: 10px 5px;
}

.topheader nav.left a{
    padding: 10px;
    display: inline-block;
}
```

With display: flex

Without display: flex



First let's use flexbox to create our navbar

```
***more above***
/*nav*/
.banner{
        width: 100%;
        display: flex;
        flex-flow:wrap;
        align-items: center;
        list-style-type: none;
        background-color: var(--forth-color);
        position: fixed;
        top: 0;
        color: var(--third-color);
        box-shadow: var(--shadow);

}
***more below***
```

Time to style our navigation menu:

```
***more above***
.title{
        margin-left: 1em;
        text-align: center;
        font-size: 0.75em;
        color: #fff;
}

a{
  color: var(--third-color);
```

```
  width: 100px;
  text-decoration: none;
        }
}
/*The :hover selector is used to select elements when you mouse over them.*/
a:hover{
            Color: var(--first-color);
}


.banner li {
  display: inline;
  margin: 1rem;
}

.banner ul {
  list-style-type: none;

}
***more below***
```

Now lets style the rest of our website section by section:

About me:

```
***more above***
/*about me*/
.about-me{
            display: flex;
            width: 80vw;
            padding: 1rem;
            margin: .5rem;
            border-radius: 6px;
            background-color: var(--second-color);
            box-shadow: var(--shadow);
}


.profile{
```

```
        padding: 1em;
}
.about-me-text{
        padding: 1em;
}

.social-media li {
        font-size: .75em;
        display: inline;
}
.social-media ul {
        padding: .3rem;
        list-style-type: none;
}
***more below***
```

Career:

```
***more above***
/*career*/

.career{
        display: flex;
        margin: .5rem;
        width: calc(80vw + 2rem);

}

.goals{
        width: 45vw;
        padding: 2rem;
        border-radius: 6px;
        margin-right:1rem;
        background-color: var(--second-color);
        box-shadow: var(--shadow);
}

.skills{
        width: 35vw;
        padding: 2rem;
```

```
        border-radius: 6px;
        background-color: var(--second-color);
        box-shadow: var(--shadow);
}
***more below***
```

Projects:

```
***more above***
/*projects*/
.projects{
        display: flex;
        flex-flow: row wrap;
        width: 80vw;
        padding: 1rem;
        margin: .5rem;
        border-radius: 6px;
        background-color: var(--second-color);
        box-shadow: var(--shadow);

}
.projects h2{
        width: 100%;
        margin: 1rem;

}

.project{
   text-align: center;
   padding: 1rem;
   margin: 1rem;
   width: 25%;
   box-shadow: var(--shadow);
   color: #fff;
   background-color: var(--third-color);
   border-radius: 8px;

}
.project a{
```

```
    color: #fff;
}

/*CSS transitions allows you to change property values smoothly, over a given duration.*/
.project:hover {
        transition: all 0.3s cubic-bezier(.25,.8,.25,1);
        box-shadow: 0 14px 28px rgba(0,0,0,0.25), 0 10px 10px rgba(0,0,0,0.22);
}
***more below***
```

Contact me:

```
***more above***
/*contact*/
.contact-form{
        display: flex;
        flex-flow: column;
        justify-content: center;
        text-align: center;
        width: 80vw;
        padding: 1rem;
        margin: .5rem;
        border-radius: 6px;
        background-color: var(--second-color);
        box-shadow: var(--shadow);
}
.contact{
        text-align: left;
        margin: 0 auto;

}

input{
        min-width: 30vw;
        margin-bottom: 15px;
        margin-top: 5px;
        padding: .5 rem;
}
```

```
input[type=submit]{
        background-color: var(--third-color);
        color: white;
        padding: 10px;
        width: calc(30vw + .5rem);
        border-radius: 8px;
        box-shadow: var(--shadow);

}
input[type=submit]:hover{
        transition: all 0.3s cubic-bezier(.25,.8,.25,1);
        box-shadow: 0 14px 28px rgba(0,0,0,0.25), 0 10px 10px rgba(0,0,0,0.22);
}

***more below***
```

Resume:

```
***more above***
/*resume*/
.jobs{
        display: flex;
        flex-flow: row wrap;
        justify-content: center;
        width: 80vw;
        margin: .5rem;
        padding: 2rem;
        border-radius: 10px;
        background-color: var(--second-color);
        box-shadow: var(--shadow);

}
.job{
        padding: 1rem;
}
***more below***
```

Education:

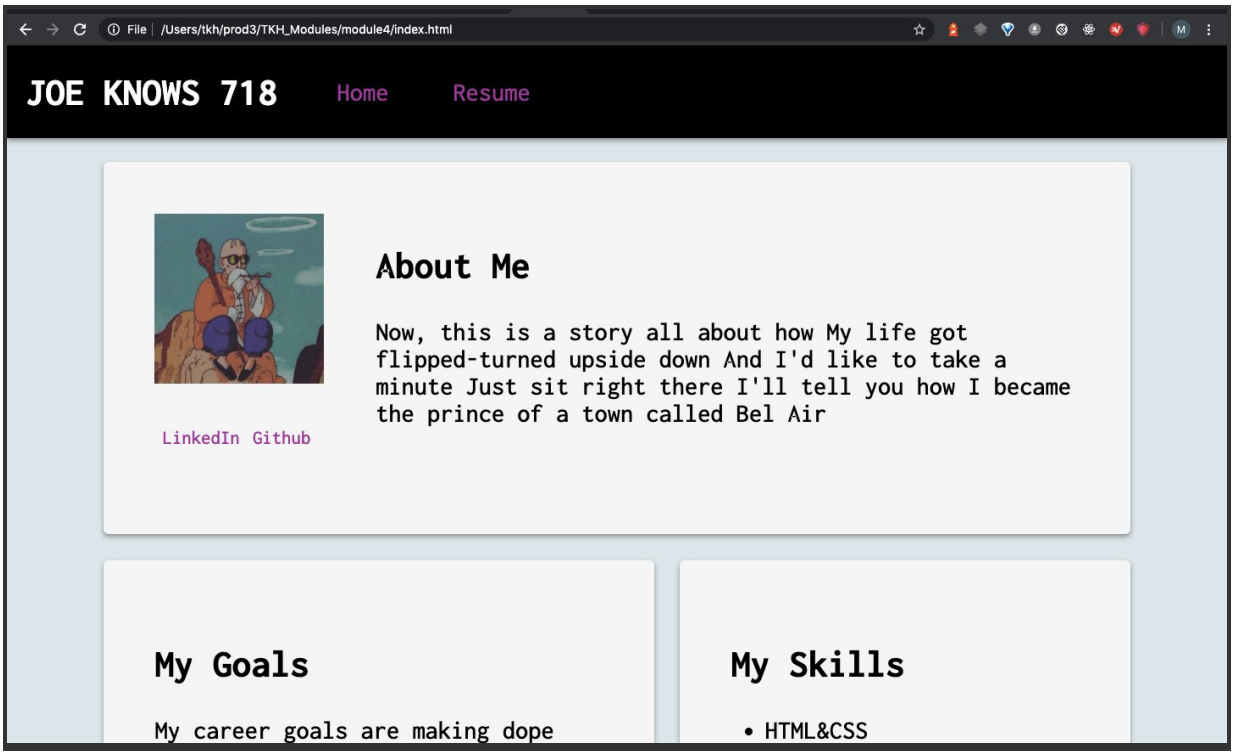
```
***more above***
/*education*/
.education{
        width: 80vw;
        margin: .5rem;
        padding: 2rem;
        border-radius: 10px;
        background-color: var(--second-color);
        box-shadow: var(--shadow);

}
***more below***
```

Footer:

```
***more above***
/*footer*/
.footer{
        margin-top: 1rem;
        width: 100vw;
        background-color: var(--forth-color);
        color: #fff;
        text-align: center;
}
***more below***
```

Now we should be done with both files and it should look something like this:



Submitting Module 4:

You will be submitting your HTML only site through Github.
Go into the module 3 folder on your master branch, and inside this folder add another folder called 'portfolio'.

Module_4_CSS
- Portfolio
  - Index.html
  - resume.html
  - images/
    - Image1.jpg *This is a placeholder for your image files
  - CSS/
    - style.css

This must be submitted through Github.

Make a copy of your portfolio folder to host your portfolio on github.io
Once Hosted submit your link on Google Classroom along with your repo link.