PERSEVERE

Student Guide to Coding

Responsive Web Design Certification(300 hours)

Responsive Web Design Certification(300 hours)

How To Get Help When You Are Stuck

When you get stuck, remember: Read and Ask.

1. **Read** the documentation or error.
2. **Ask** your instructor for help.

This is the most time-efficient way to handle being stuck, and it's the most respectful of other people's time, too.

Most of the time, you'll solve your problem after just one or two steps of this algorithm.

Just a little word of encouragement from a fellow inmate that has gone through the program that you are about to experience: "The main thing is 'Take it slow', 'Take plenty of notes', and 'STUDY.' For me, I had to study a lot because when I started the program, I knew very little about computers [other than] the occasional touch screen that I used behind closed doors here and there. You know how it is. So the biggest thing is not to get discouraged! Be diligent and if you really want it, you WILL get it. Just think with enough perseverance you can be making over a hundred thousand dollars a year. Awsome, right? So don't give up and remember that a computer is only as smart as you make it. Good luck, and use your code for good."

Basic Vocabulary:

Developer- An individual that builds and creates software and applications. He/She writes, debugs and executes the source code of a software application.

A developer is also known as a software developer, computer programmer, programmer, software coder or software engineer.

Website- A location connected to the internet that maintains one or more pages on the World Wide Web.

Front End Developer- Also known as client-side development. This is the practice of producing HTML, CSS and JavaScript for a website or Web Application so that a user can see and interact with them directly.

Back End Developer- A type of programmer who creates the logical back-end and core computational logic of a website, software or information system. The developer creates components and features that are indirectly assessed by a user through a front-end application or system.

Full Stack Developer- An engineer who can handle all work of databases, servers, systems engineering and clients. Depending on the project, what customers need may be a mobile stack, a Web stack, or a native application stack.

Web Development- Web development is the work involved in developing a website for the Internet (World Wide Web) or an intranet (a private network)... Among web professionals, “web development” usually refers to the main non-design aspects of building websites: writing markup and coding.

User- An end user is the person that a software program or hardware device is designed for. The team is based on the idea that the “end goal” of a software or hardware product is to be useful to the customer.

The end user can be contrasted with the developers or programmers of the product. End users are also in a separate group from the installers or administrators of the product.

To simplify, the end user is the person who uses the software or hardware after it has been fully developed, marketed, and installed. It is also the person who keeps calling the “IT guy” with questions about why the product isn’t working correctly. Generally, the terms “user” and “end user” mean the same thing.

W3 Consortium- The World Wide Web Consortium is the main international standards organization for the World Wide Web. The World Wide Web Consortium (W3C) is an international community where member organizations, a full-time staff, and the public work together to develop Web standards. Led by Web inventor and Director Tim-Berners-Lee and CEO Jeffrey Jaffe, W3C’s mission is to lead the Web to its full potential.

Code- Code, in a general sense, is the language understood by the computer. Computers don’t understand natural language. As such, the human language has to be converted into a set of “words” that can be understood by the computer. The words that initiate a standard action when used in a program are called “keywords”. The arrangement of keywords for successful execution of a desired computation is called “syntax”. The set of keywords and syntax form a programming language.

Code Editor- A source code editor is a text editor program designed specifically for editing source code of computer programs. It may be a standalone application or it may be built into an integrated development environment (IDE) or Web browser. Source code editors are a fundamental programming tool, as the job of programmers is to write and edit source code.

Algorithm- A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

Introduction to Basic HTML & HTML5

HTML, or HyperText Markup Language, is a markup language used to describe the structure of a web page. It uses a special syntax or notation to organize and give information about the page to the browser. Elements usually have opening and closing tags that surround and give meaning to content. For example, there are different tag options to place around text to show whether it is a heading, a paragraph, or a list.

For example:

```
<h1>Top level heading: Maybe a page title</h1>

<p>A paragraph of text. Some information we would like to communicate to the viewer. This can be as long or short as we would like.</p>

<ol>
  <li>Number one on the list</li>
  <li>Number two</li>
  <li>A third item</li>

</ol>
```

Becomes:

Top level heading: Maybe a page title

A paragraph of text. Some information we would like to communicate to the user. This can be as long or short as we would like.

1.Number one on the list

2. Number two

3. A third item

The HyperText part of HTML comes from the early days of the web and its original use case. Pages usually contained static documents that contained references to other documents. These references contained hypertext links used by the browser to navigate to the reference document so the user could read the reference document without having to manually search for it.

As web pages and web applications grow more complex, the W3 Consortium updates the HTML specification to ensure that a webpage can be shown reliably on any browser. The latest version of HTML is HTML5.

This section introduces how to use HTML elements to give structure and meaning to your web content.

HTML Vocabulary:

HTML (Hypertext Markup Language)- HTML (Hypertext Markup Language) is a text-based approach to describing how content contained within an HTML file is structured. This markup tells a web browser how to display the text, images and other forms of multimedia on a webpage.

Syntax- Syntax is the grammar, structure, or order of the elements in a language statement. (Semantics is the meaning of these elements.) Syntax applies to the computer languages as well as to natural languages. Usually, we think of Syntax as “word order.” However, syntax is also achieved in some languages such as Latin by inflectional case endings. In computer languages, syntax can be extremely rigid as in the case of most assembler languages or less rigid in languages that make use of “Keyword” parameters that can be stated in any order.

Browser- A browser is software that is used to access the internet. A browser lets you visit websites and do activities within them like login, view multimedia, link from one site to another, visit one page from another, print, send and receive email, among many other activities. The most common browser software titles on the market are: Microsoft Internet Explorer, Google’s Chrome, Mozilla Firefox, Apple’s Safari and Opera. Browser availability depends on the operating system your computer is using (for example: Microsoft Windows, Linux, Ubuntu, Mac OS, among others).

When you type a web page such as www.allaboutcookies.org into your browser, that web page in its entirety is not actually stored on a server ready and waiting to be delivered. In fact, each web page that you request is individually created in response to your request.

You are actually calling up a list of requests to get content from various resource directories or servers on which the content for that page is stored. It is rather like a recipe for a cake - you have a shopping list of ingredients (request for content) that combined in the correct order bakes a cake (the web page). The page may be made up from content from different sources. Images may come from one server, text content from another, scripts such as date scripts from another, and ads from another. As soon as you move to another page, the page that you just viewed disappears. This is the dynamic nature of websites.

Web App- This is a computer program/application that is not installed and run locally on your computer, but online through a website.

Elements- An HTML element usually consists of a **start** tag and **end** tag, with the content inserted in between.
Example: <tagName>Content goes here...</tagName>

The HTML element is everything from the start tag to the end tag:
Example: <p>My First Paragraph.</p>

Start tag	**Element Content**	**End tag**
<h1>	My First Heading	</h1>
<p>	My First Paragraph	</p>

<br> <hr>
HTML elements such as these with no content are called empty elements. Empty elements do not have an end tag. The <br> tag gives you a line break and the <hr> gives you a horizontal line.

Opening Tag/Closing Tag- In an HTML document, tags surround an HTML element, determine the elements contents, and identify the meaning of the element.

Tags are the HTML tools used to create the elements that make up the web page.

An HTML element is any single component of a web page. For example, a paragraph of text, a navigation menu item, an image and an article heading are each an example of an HTML element. Each element begins where it ends and assigns a meaning to the element.

- **The beginning of an element** is marked by an opening tag. An HTML opening tag consists of a left angle bracket (<), the name of the element ("p" in the case of a paragraph), and a right angle bracket(>).

- **The end of an element** is marked by a closing tag. A closing tag consists of a left angle bracket (<), a forward slash (/), the name of the element ("p" in the case of a paragraph), and a right angle bracket (>).

- **The meaning of an element** is assigned by the name of the element. The name of an element consists of a series of letters and numbers. As we've already mentioned, a paragraph element is marked by the letter "p". When text is surrounded by an opening and closing "p" tag, the browser knows that the text surrounded by the tags is a paragraph of text.

Taken together, an opening tag, element content, and closing tag will look like this in an HTML document.

Attribute- An attribute is a specification that defines a property of an object, element, or file. It may also refer to or set the specific value for a given instance of such. For clarity, attributes should more correctly be considered metadata. An attribute is frequently and generally a property of a property. However, in actual usage, the term attribute can and is often treated as equivalent to a property depending on the technology being discussed. An attribute of an object usually consists of a name and a value; of an element, a type or class name; of a file, a name and extension.

Placeholder Text- Sometimes, while building a web page, you'll want to use placeholder text to have a better idea of what the finished product will look like. When in the early stages of building and design, you know you will have a p element, but you may not know exactly what will go in that p element and how it will affect the rest of the page. You will use placeholder text to give you a better idea of what the final outcome will look like.

Comments- A comment is like a note that is left in your code. Comments are not executed by the computer; they are only there to be read by you and by others who have access to the code.

Comments are very important to coding. It is very rare that, when working, you will be the only person to touch the code you are working on.

Sometimes you will work with a team. Other times you will move on from a project, but someone else will have to edit the code in the future. Or it could be the other way around. Comments are crucial for others to be able to understand why and how you built your code.

Aside from leaving notes for your teammates, comments are a huge help for debugging (finding out why your own code is not working). Debugging involves a lot of going back over your code. So knowing your way around inside of your code is a must.

Nesting- Nesting means putting elements or functions inside of each other, making them work the same just at a deeper level. Some tags can be nested inside of other tags. Nesting is a concept that you will see throughout all coding, not just HTML. For the purposes of HTML, nesting is important for the flow of your code.

HTML Elements:

Heading Tags- <h1>-<h6> elements are headings. They are used for headings like titles. <h1> elements are the largest, and <h6> elements are the smallest. As the number grows, the headings get smaller.

Paragraph- The <p> element is used for paragraphs. For those not familiar, a paragraph is a body of sentences (usually 5 or more) that all pertain to the same subject. When you have a large amount of text for users to read through, you will use a paragraph, rather than a heading.

Header- Represents a container for introductory content or a set of navigational links.
A <header> element typically contains:

- *One or more heading elements(<h1>-<h6>)*
- *Logo and/or Icon*
- *Authorship information*

You can have several <header> elements in one document.
Note: A <header> tag cannot be placed within a <footer>,<address> or another <header> element.

Footer- Defines a footer for a document or section.
A <footer> element should contain information about its containing element.
A <footer> element typically contains:

- *Authorship information*
- *Copyright information*
- *Contact information*
- *Sitemap*
- *Back to the top links*
- *Related documents*

You can have several <footer> elements in one document also.

Main- Specifies the main content of a document.
The content inside the <main> element should be unique to the document. It should not contain any content that is repeated across documents, such as sidebars, navigation links, copyright information, site logos, and search forms.

Note: There must not be more than one <main> element in a document. The <main> element must NOT be a descendant of an <article>, <aside>, <footer>, <header> or <nav> elements.

Section- Defines sections in a document, such as chapters, headers, footers or any other sections of the document.

Form- The <form> element is used to create an HTML form for user input.
The <form> element can contain one or more of the following form elements:

- <input>- The <input> tag specifies an input field where the user can enter data. <input> elements are used within a <form> element to declare input controls that allow users to input data. An input field can vary in many ways, depending on the type attribute.

- <textarea>- The <textarea> element defines a multi-line text input control. A text area can hold an unlimited number of characters, and the text renders in a fixed-width font (usually Courier). The size of a text area can be specified by the col and row attributes, or even better; through CSS' height and width properties.
- <button>- The <button> element defines a clickable button. Inside a <button> element you can add content, like text or images. This is the difference between this element and buttons created with the <input> element.
- <select>- The <select> element is used to create a drop-down list. The <option> tags inside the <select> element define the available options in the list.
- <option> The <option> element defines an option in a select list. <option> elements go inside a <select> or <datalist> element.
- <optgroup>- The <optgroup> element is used to group related options in a drop-down list. If you have a long list of options, groups of related options are easier to handle for a user.
- <fieldset>- The <fieldset> element is used to group related elements in a form. The <fieldset> tag draws a box around the related elements.
- <label>- The <label> element defines a label for a <button>, <input>, <meter>, <output>, <progress>, <select> or <textarea> elements. The <label> element does not render as anything special for the user. However, it provides usability improvement for mouse users, because if the user clicks on the text within the <label> element, it toggles the control. The for attribute of the <label> element should be equal to the id attribute of the related element to bind them together.

Nav- Defines a set of navigation links.
Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major blocks of navigation links.
Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content.

Image- Images are very useful in web design. Websites are full of them. Images may be used for backgrounds, decorations, icons, links, references, and just about anything else you can imagine. The <img> tag is the first self-closing tag we encounter. This means instead of having something like <img>"image address", "alternate text"</img>, we have something more like this:
<img src="https://www.wherever-image-cones-from.com/this-particular-image.jpg" alt="This is an example of an image.">
All of the image info is stuffed inside of one tag.
The <img> tag has several attributes. Here are some of the important ones:

- ***alt-****Specifies an alternative text for an image. In case the image doesn't show for some reason.*

- ***src-****Specifies the url of an image. If the src is not provided, a browser won't be able to display the image, because it won't know where to find it.*

Video- Specifies video, such as a movie clip or other video streams.
Currently, there are 3 supported video formats for the <video> element: MP4, WebM and Ogg.

Article- Specifies independent, self-contained content.
An article should make sense on its own and it should be possible to distribute it independently from the rest of the site.
Potential sources for the <article> element is:

- *Forum post*
- *Blog post*
- *News stories*
- *Comments*

Anchor- These are links on a web page. A link is simply something that, when clicked, will direct a user to another web page. An anchor element is a bit of a hybrid between regular tags and self-closing tag. It has an opening and closing tags. However, like an <img> element, it also takes arguments, like href, inside the opening tag.
Example: <a href="www.google.com">Click here to go to Google</a>

Lists:
Unordered List- The <ul> element defines an unordered (bulleted) list.
Use the <ul> element together with the <li> element inside of the opening and closing <ul> elements to create an unordered list.
Ordered List- The <ol> element defines an ordered list. An ordered list can be numerical or alphabetical. Just like the <ul> tag, use the <li> tag inside to define the list items.

Division - The <div> element defines a division or a selection in an HTML document. The <div> element is often used as a container for other HTML elements to style them with CSS(custom style sheet) and Bootstrap (front-end framework to design) or to perform certain tasks with JavaScript.

Head- The <head> element is a container for all the head elements. The <head> element can include a title for the document, scripts, styles, meta information, and more.
The following elements can go inside the <head> element:

- *<title>- (This element is required in an HTML document)- The <title> tag is required in all HTML documents and if defines the title of the document.*
  The <title> element:

        - *Defines a title in the browser toolbar*
        - *Provides a title for the page when it is added to favorites*
        - *Displays a title for the page in search-engine results*

- *<style>- The <style> element is used to define style information for an HTML document. Inside the <style> element you specify how HTML elements should render in a browser. Each HTML document can contain multiple <style> tags.*

- *<base>- The < base > element specifies the base URL/target for all relative URLs in a document. There can be at maximum one <base>, element in a document, and it must be inside the <head> element.*

- *<link>- The < link > element defines a link between a document and an external resource. The <link> tag is used to link to external style sheets.*

- *<meta>- Metadata is data (information) about data. The <meta>, tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable. Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata. The metadata can be used by browsers (how to display content or reload page), search engines (keywords), or other web services.*

- *<script>- The <script> element is used to define a client side script (JavaScript). The <script> element contains scripting statements, or it points to an external script file through the src attribute. Common uses for the Javascript are image manipulation, form validation, and dynamic changes of content.*

- *<noscript>- The <noscript> element*

Body - The <body> tag defines the document's body. The <body> element contains all the contents of an HTML document, such as text, hyperlinks, images, tables, lists, etc.

Basic HTML Review Questions:

1. What is HTML?

2. What is an element?

3. What does almost every element consist of?

4. As of now, what elements are self-closing?

5. What is an attribute?

6. How many headings are there?

7. What elements are considered as containers?

8. In an image tag, What two attributes must you have for it to work.

9. What element is used to define style information in a document?

10. What tags would you use if you wanted to make a numbered list of items? Please write a list of 3 items. (Don't forget all opening and closing tags)

Write your code here:

Introduction to Basic CSS

Cascading Style Sheets (CSS) tell the browser how to display the text and other content that you write in HTML.

Note that CSS is case-sensitive so be careful with your capitalization. CSS has been adopted by all major browsers and allows you to control:

- color
- fonts
- positioning
- spacing
- sizing
- decorations
- Transitions

There are three main ways to apply CSS styling. You can apply inline styles directly to HTML elements with the style attribute. Alternatively, you can place CSS rules within style tags in an HTML document. Finally, you can write CSS rules in an external style sheet, then reference that file in the HTML document. Even though the first two options have their use cases, most developers prefer external style sheets because they keep the styles separate from the HTML elements. This improves the readability and reusability of your code. The idea behind CSS is that you can use a selector to target an HTML element in the DOM

(Document Object Model) and then apply a variety of attributes to that element to change the way it is displayed on the page.

Basic CSS Vocabulary:

CSS- CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External stylesheets are stored in CSS files.

Inline Styles - An inline style may be used to apply a unique style for a single element. To use inline styles add the style attribute to the relevant element. The style attribute can contain any CSS property.

Ex:

Code - `<h1 style="color:blue;margin-left:30px;">This is a heading</h1>`

Output - **This is a heading**

Style Attribute - The style attribute specifies an inline style for an element.
The style attribute will override any style set globally, e.g. styles specified in the <style> tag or in an external style sheet.

External Style Sheet - With an external style sheet, you can change the look of an entire website by changing just one file! Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section.

Ex:

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

Selectors - In CSS, selectors are patterns used to select the element(s) you want to style. There is a list of CSS selectors at the end of this section.

Class Selector - The .class selector selects elements with a specific class attribute.
To select elements with a specific class, write a period (.) character, followed by the name of the class.

Ex:

```
.intro {
  background-color: yellow;
}
```

Font - the combination of typeface and other qualities, such as size, pitch, and spacing of text.

id Selector - The #id, selector styles the element with the specified id. Just like classes begin with a ".", the id will always begin with "#".

Ex:
```
#firstname {
  background-color: yellow;
}
```

Padding - An element's padding controls the amount of space between the element's content and its border.

Margin - An element's margin controls the amount of space between an element's border and surrounding elements.

Attribute Selector - The [attribute] selector is used to select elements with the specified attribute.

Ex:
```
[type='radio'] {
  margin: 20px 0px 20px 0px;
}
```

CSS Inheritance - Unless specified differently, all CSS objects will inherit their styling from their parent component..

CSS Variables - 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!

Ex:
```
var(custom-name, value)
```

Media Query - Media query is a CSS technique introduced in CSS3.
It uses the @media rule to include a block of CSS properties only if a certain condition is true.

Ex: If the browser window is 600px or smaller, the background color will be lightblue:
```
@media only screen and (max-width: 600px) {
  body {
        background-color: lightblue;
  }
```

```
        }
```

Readability - The ease with which a web page can be read through.

Reusability - The ability to reuse a CSS document for several files, rather than creating new CSS for each file.

CSS Properties:

A

align-content	Specifies the alignment between the lines inside a flexible container when the items do not use all available space
align-items	Specifies the alignment for items inside a flexible container
align-self	Specifies the alignment for selected items inside a flexible container
all	Resets all properties (except unicode-bidi and direction)
animation	A shorthand property for all the animation-* properties
animation-delay	Specifies a delay for the start of an animation
animation-direction	Specifies whether an animation should be played forwards, backwards or in alternate cycles
animation-duration	Specifies how long an animation should take to complete one cycle
animation-fill-mode	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
animation-iteration-count	Specifies the number of times an animation should be played
animation-name	Specifies a name for the @keyframes animation
animation-play-state	Specifies whether the animation is running or paused
animation-timing-function	Specifies the speed curve of an animation

B

backface-visibility	Defines whether or not the back face of an element should be visible when facing the user
background	A shorthand property for all the background-* properties
background-attachment	Sets whether a background image scrolls with the rest of the page, or is fixed
background-blend-mode	Specifies the blending mode of each background layer (color/image)
background-clip	Defines how far the background (color or image) should extend within an element
background-color	Specifies the background color of an element
background-image	Specifies one or more background images for an element
background-origin	Specifies the origin position of a background image
background-position	Specifies the position of a background image
background-repeat	Sets if/how a background image will be repeated
background-size	Specifies the size of the background images
border	A shorthand property for border-width, border-style and border-color
border-bottom	A shorthand property for border-bottom-width, border-bottom-style and border-bottom-color
border-bottom-color	Sets the color of the bottom border
border-bottom-left-radius	Defines the radius of the border of the bottom-left corner
border-bottom-right-radius	Defines the radius of the border of the bottom-right corner
border-bottom-style	Sets the style of the bottom border
border-bottom-width	Sets the width of the bottom border
border-collapse	Sets whether table borders should collapse into a single border or be separated
border-color	Sets the color of the four borders
border-image	A shorthand property for all the border-image-* properties

border-image-outset	Specifies the amount by which the border image area extends beyond the border box
border-image-repeat	Specifies whether the border image should be repeated, rounded or stretched
border-image-slice	Specifies how to slice the border image
border-image-source	Specifies the path to the image to be used as a border
border-image-width	Specifies the width of the border image
border-left	A shorthand property for all the border-left-* properties
border-left-color	Sets the color of the left border
border-left-style	Sets the style of the left border
border-left-width	Sets the width of the left border
border-radius	A shorthand property for the four border-*-radius properties
border-right	A shorthand property for all the border-right-* properties
border-right-color	Sets the color of the right border
border-right-style	Sets the style of the right border
border-right-width	Sets the width of the right border
border-spacing	Sets the distance between the borders of adjacent cells
border-style	Sets the style of the four borders
border-top	A shorthand property for border-top-width, border-top-style and border-top-color
border-top-color	Sets the color of the top border
border-top-left-radius	Defines the radius of the border of the top-left corner
border-top-right-radius	Defines the radius of the border of the top-right corner
border-top-style	Sets the style of the top border
border-top-width	Sets the width of the top border
border-width	Sets the width of the four borders

bottom	Sets the elements position, from the bottom of its parent element
box-decoration-break	Sets the behavior of the background and border of an element at page-break, or, for in-line elements, at line-break.
box-shadow	Attaches one or more shadows to an element
box-sizing	Defines how the width and height of an element are calculated: should they include padding and borders, or not
break-after	Specifies the page-, column-, or region-break behavior after the generated box
break-before	Specifies the page-, column-, or region-break behavior before the generated box
break-inside	Specifies the page-, column-, or region-break behavior inside the generated box

C

caption-side	Specifies the placement of a table caption
caret-color	Specifies the color of the cursor (caret) in inputs, textareas, or any element that is editable
@charset	Specifies the character encoding used in the style sheet
clear	Specifies on which side of an element floating elements are not allowed to float
clip	Clips an absolutely positioned element
color	Sets the color of text
column-count	Specifies the number of columns an element should be divided into
column-fill	Specifies how to fill columns, balanced or not
column-gap	Specifies the gap between the columns
column-rule	A shorthand property for all the column-rule-* properties
column-rule-color	Specifies the color of the rule between columns

column-rule-style	Specifies the style of the rule between columns
column-rule-width	Specifies the width of the rule between columns
column-span	Specifies how many columns an element should span across
column-width	Specifies the column width
columns	A shorthand property for column-width and column-count
content	Used with the :before and :after pseudo-elements, to insert generated content
counter-increment	Increases or decreases the value of one or more CSS counters
counter-reset	Creates or resets one or more CSS counters
cursor	Specifies the mouse cursor to be displayed when pointing over an element

D

direction	Specifies the text direction/writing direction
display	Specifies how a certain HTML element should be displayed

E

empty-cells	Specifies whether or not to display borders and background on empty cells in a table

F

filter	Defines effects (e.g. blurring or color shifting) on an element before the element is displayed
flex	A shorthand property for the flex-grow, flex-shrink, and the flex-basis properties
flex-basis	Specifies the initial length of a flexible item

flex-direction	Specifies the direction of the flexible items
flex-flow	A shorthand property for the flex-direction and the flex-wrap properties
flex-grow	Specifies how much the item will grow relative to the rest
flex-shrink	Specifies how the item will shrink relative to the rest
flex-wrap	Specifies whether the flexible items should wrap or not
float	Specifies whether or not a box should float
font	A shorthand property for the font-style, font-variant, font-weight, font-size/line-height, and the font-family properties
@font-face	A rule that allows websites to download and use fonts other than the "web-safe" fonts
font-family	Specifies the font family for text
font-feature-settings	Allows control over advanced typographic features in OpenType fonts
@font-feature-values	Allows authors to use a common name in font-variant-alternate for feature activated differently in OpenType
font-kerning	Controls the usage of the kerning information (how letters are spaced)
font-language-override	Controls the usage of language-specific glyphs in a typeface
font-size	Specifies the font size of text
font-size-adjust	Preserves the readability of text when font fallback occurs
font-stretch	Selects a normal, condensed, or expanded face from a font family
font-style	Specifies the font style for text
font-synthesis	Controls which missing typefaces (bold or italic) may be synthesized by the browser
font-variant	Specifies whether or not a text should be displayed in a small-caps font

font-variant-alternates	Controls the usage of alternate glyphs associated to alternative names defined in @font-feature-values
font-variant-caps	Controls the usage of alternate glyphs for capital letters
font-variant-east-asian	Controls the usage of alternate glyphs for East Asian scripts (e.g Japanese and Chinese)
font-variant-ligatures	Controls which ligatures and contextual forms are used in textual content of the elements it applies to
font-variant-numeric	Controls the usage of alternate glyphs for numbers, fractions, and ordinal markers
font-variant-position	Controls the usage of alternate glyphs of smaller size positioned as superscript or subscript regarding the baseline of the font
font-weight	Specifies the weight of a font

G

grid	A shorthand property for the grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns, and the grid-auto-flow properties
grid-area	Either specifies a name for the grid item, or this property is a shorthand property for the grid-row-start, grid-column-start, grid-row-end, and grid-column-end properties
grid-auto-columns	Specifies a default column size
grid-auto-flow	Specifies how auto-placed items are inserted in the grid
grid-auto-rows	Specifies a default row size
grid-column	A shorthand property for the grid-column-start and the grid-column-end properties
grid-column-end	Specifies where to end the grid item
grid-column-gap	Specifies the size of the gap between columns
grid-column-start	Specifies where to start the grid item

grid-gap	A shorthand property for the grid-row-gap and grid-column-gap properties
grid-row	A shorthand property for the grid-row-start and the grid-row-end properties
grid-row-end	Specifies where to end the grid item
grid-row-gap	Specifies the size of the gap between rows
grid-row-start	Specifies where to start the grid item
grid-template	A shorthand property for the grid-template-rows, grid-template-columns and grid-areas properties
grid-template-areas	Specifies how to display columns and rows, using named grid items
grid-template-columns	Specifies the size of the columns, and how many columns in a grid layout
grid-template-rows	Specifies the size of the rows in a grid layout

H

hanging-punctuation	Specifies whether a punctuation character may be placed outside the line box
height	Sets the height of an element
hyphens	Sets how to split words to improve the layout of paragraphs

I

image-rendering	Gives a hint to the browser about what aspects of an image are most important to preserve when the image is scaled
@import	Allows you to import a style sheet into another style sheet
isolation	Defines whether an element must create a new stacking content

J

justify-content	Specifies the alignment between the items inside a flexible container when the items do not use all available space

K

@keyframes	Specifies the animation code

L

left	Specifies the left position of a positioned element
letter-spacing	Increases or decreases the space between characters in a text
line-break	Specifies how/if to break lines
line-height	Sets the line height
list-style	Sets all the properties for a list in one declaration
list-style-image	Specifies an image as the list-item marker
list-style-position	Specifies the position of the list-item markers (bullet points)
list-style-type	Specifies the type of list-item marker

M

margin	Sets all the margin properties in one declaration
margin-bottom	Sets the bottom margin of an element
margin-left	Sets the left margin of an element
margin-right	Sets the right margin of an element
margin-top	Sets the top margin of an element
max-height	Sets the maximum height of an element
max-width	Sets the maximum width of an element
@media	Sets the style rules for different media types/devices/sizes
min-height	Sets the minimum height of an element
min-width	Sets the minimum width of an element
mix-blend-mode	Specifies how an element's content should blend with its direct parent background

O

object-fit	Specifies how the contents of a replaced element should be fitted to the box established by its used height and width
object-position	Specifies the alignment of the replaced element inside its box
opacity	Sets the opacity level for an element
order	Sets the order of the flexible item, relative to the rest
orphans	Sets the minimum number of lines that must be left at the bottom of a page when a page break occurs inside an element
outline	A shorthand property for the outline-width, outline-style, and the outline-color properties
outline-color	Sets the color of an outline
outline-offset	Offsets an outline, and draws it beyond the border edge
outline-style	Sets the style of an outline
outline-width	Sets the width of an outline
overflow	Specifies what happens if content overflows an element's box
overflow-wrap	Specifies whether or not the browser may break lines within words in order to prevent overflow (when a string is too long to fit its containing box)
overflow-x	Specifies whether or not to clip the left/right edges of the content, if it overflows the element's content area
overflow-y	Specifies whether or not to clip the top/bottom edges of the content, if it overflows the element's content area

P

padding	A shorthand property for all the padding-* properties
padding-bottom	Sets the bottom padding of an element
padding-left	Sets the left padding of an element

padding-right	Sets the right padding of an element
padding-top	Sets the top padding of an element
page-break-after	Sets the page-break behavior after an element
page-break-before	Sets the page-break behavior before an element
page-break-inside	Sets the page-break behavior inside an element
perspective	Gives a 3D-positioned element some perspective
perspective-origin	Defines at which position the user is looking at the 3D-positioned element
pointer-events	Defines whether or not an element reacts to pointer events
position	Specifies the type of positioning method used for an element (static, relative, absolute or fixed)

Q

quotes	Sets the type of quotation marks for embedded quotations

R

resize	Defines if (and how) an element is resizable by the user
right	Specifies the right position of a positioned element

S

scroll-behavior	Specifies whether to smoothly animate the scroll position in a scrollable box, instead of a straight jump

T

tab-size	Specifies the width of a tab character
table-layout	Defines the algorithm used to lay out table cells, rows, and columns
text-align	Specifies the horizontal alignment of text
text-align-last	Describes how the last line of a block or a line right before a forced line break is aligned when text-align is "justify"

text-combine-upright	Specifies the combination of multiple characters into the space of a single character
text-decoration	Specifies the decoration added to text
text-decoration-color	Specifies the color of the text-decoration
text-decoration-line	Specifies the type of line in a text-decoration
text-decoration-style	Specifies the style of the line in a text decoration
text-indent	Specifies the indentation of the first line in a text-block
text-justify	Specifies the justification method used when text-align is "justify"
text-orientation	Defines the orientation of the text in a line
text-overflow	Specifies what should happen when text overflows the containing element
text-shadow	Adds shadow to text
text-transform	Controls the capitalization of text
text-underline-position	Specifies the position of the underline which is set using the text-decoration property
top	Specifies the top position of a positioned element
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space
transition	A shorthand property for all the transition-* properties
transition-delay	Specifies when the transition effect will start
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
transition-property	Specifies the name of the CSS property the transition effect is for
transition-timing-function	Specifies the speed curve of the transition effect

U

unicode-bidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
user-select	Specifies whether the text of an element can be selected

V

vertical-align	Sets the vertical alignment of an element
visibility	Specifies whether or not an element is visible

W

white-space	Specifies how white-space inside an element is handled
widows	Sets the minimum number of lines that must be left at the top of a page when a page break occurs inside an element
width	Sets the width of an element
word-break	Specifies how words should break when reaching the end of a line
word-spacing	Increases or decreases the space between words in a text
word-wrap	Allows long, unbreakable words to be broken and wrap to the next line
writing-mode	Specifies whether lines of text are laid out horizontally or vertically

Z

z-index	Sets the stack order of a positioned element

CSS Selectors:

Selector	**Example**	**Example description**

.class	.intro	Selects all elements with class="intro"
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element	div, p	Selects all <div> elements and all <p> elements
element element	div p	Selects all <p> elements inside <div> elements
element>element	div > p	Selects all <p> elements where the parent is a <div> element
element+element	div + p	Selects all <p> elements that are placed immediately after <div> elements
element1~element2	p ~ ul	Selects every <ul> element that are preceded by a <p> element
[attribute]	[target]	Selects all elements with a target attribute

[attribute=value]	[target=_blank]	Selects all elements with target="_blank"
[attribute~=value]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[attribute|=value]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"
[attribute^=value]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[attribute$=value]	a[href$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[attribute*=value]	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"
:active	a:active	Selects the active link
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p>

		element
:checked	input:checked	Selects every checked <input> element
:default	input:default	Selects the default <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children (including text nodes)
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> element that is the first child of its parent
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the input element which has focus

:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects input elements with a value within a specified range
:indeterminate	input:indeterminate	Selects input elements that are in an indeterminate state
:invalid	input:invalid	Selects all input elements with an invalid value
:lang(language)	p:lang(it)	Selects every <p> element with a lang attribute equal to "it" (Italian)
:last-child	p:last-child	Selects every <p> element that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element

:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects input elements with no "required" attribute
:out-of-range	input:out-of-range	Selects input elements with a value outside a specified range
::placeholder	input::placeholder	Selects input elements with placeholder text

:read-only	input:read-only	Selects input elements with the "readonly" attribute specified
:read-write	input:read-write	Selects input elements with the "readonly" attribute NOT specified
:required	input:required	Selects input elements with the "required" attribute specified
:root	:root	Selects the document's root element
::selection	::selection	Selects the portion of an element that is selected by a user
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all input elements with a valid value
:visited	a:visited	Selects all visited links

Basis CSS Review Questions:

1. Where would you put your CSS in your document?

2. What is the difference between an external style sheet, and inline styles?

3. Write an example of changing the font size of all <h1> elements.

4. What is a CSS selector?

5. What is a class selector?

6. What is an id selector?

7. What is a CSS property?

8. What is a style attribute?

9. What does CSS stand for?

10. What is the difference between padding and margin?

Introduction to the Applied Visual Design Challenges

Visual Design in web development is a broad topic. It combines typography, color theory, graphics, animation, and page layout to help deliver a site's message. The definition of good design is a well-discussed subject, with many books on the theme.

At a basic level, most web content provides a user with information. The visual design of the page can influence its presentation and a user's experience. In web development, HTML gives structure and semantics to a page's content, and CSS controls the layout and appearance of it.

This section covers some of the basic tools developers use to create their own visual designs.
Vocabulary:

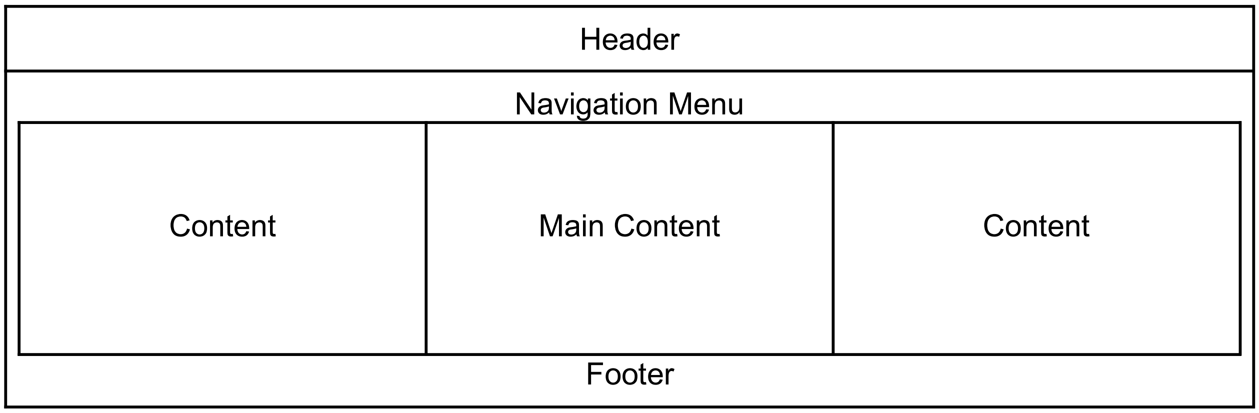
Typography - The art and technique of arranging type to make written language legible, readable, and appealing when displayed.

Color Theory - A body of practical guidance to color mixing and the visual effects of a specific color combination.

Graphics - Visual representations of data displayed on a monitor made on a computer. Computer graphics can be a series of images (most often called video) or a single image.

Animation - Computer animation is a general term for a kind of visual digital display technology that simulates moving objects on-screen.

Page Layout - A website is often divided into headers, menus, content and a footer:
There are tons of different layout designs to choose from. However, the layout below is one of the most common.
Ex:



Text-align - The text-align property specifies the horizontal alignment of text in an element.

Value	Description
left	Aligns the text to the left
right	Aligns the text to the right
center	Centers the text
justify	Stretches the lines so that each line has equal width (like in newspapers and magazines)
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Font-weight - The font-weight property sets how thick or thin characters in text should be displayed.

Value	Description
normal	Defines normal characters. This is default
bold	Defines thick characters
bolder	Defines thicker characters
lighter	Defines lighter characters
100 200 300 400 500 600 700 800 900	Defines from thin to thick characters. 400 is the same as normal, and 700 is the same as bold.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Text-decoration - The text-decoration property specifies the decoration added to text.

Value	Description
text-decoration-line	Sets the kind of text decoration to use (like underline, overline, line-through)
text-decoration-color	Sets the color of the text decoration
text-decoration-style	Sets the style of the text decoration (like solid, wavy, dotted, dashed, double)
initial	Sets this property to its default value
inherit	Inherits this property from its parent element

Font-style - The font-style property specifies the font style for a text.

Value	Description
normal	The browser displays a normal font style. This is default.
italic	The browser displays an italic font style.
oblique	The browser displays an oblique font style.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Horizontal Line - You can use the <hr> tag to add a horizontal line across the width of its containing element. This can be used to define a change in topic or to visually separate groups of content.

Box-shadow - The box-shadow property attaches one or more shadows to an element.

Value	Description
none	Default value. No shadow is displayed.
h-offset	Required. The horizontal offset of the shadow. A positive value puts the shadow on the right side of the box, a negative value puts the shadow on the left side of the box.
v-offset	Required. The vertical offset of the shadow. A positive value puts the shadow below the box, a negative value puts the shadow above the box.

blur	Optional. The blur radius. The higher the number, the more blurred the shadow will be.
spread	Optional. The spread radius. A positive value increases the size of the shadow, a negative value decreases the size of the shadow.
color	Optional. The color of the shadow. The default value is the text color.
inset	Optional. Changes the shadow from an outer shadow (outset) to an inner shadow.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Opacity - The opacity property sets the opacity level for an element.
The opacity-level describes the transparency-level, where 1 is not transparent at all, 0.5 is 50% see-through, and 0 is completely transparent.

Text-transform - The text-transform property controls the capitalization of text.

Value	**Description**
none	No capitalization. The text renders as it is. This is default.
capitalize	Transforms the first character of each word to uppercase.
uppercase	Transforms all characters to uppercase.
lowercase	Transforms all characters to lowercase.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Line-height - The line-height property specifies the height of a line.

Value	**Description**

normal	A normal line height. This is default.
number	A number that will be multiplied with the current font-size to set the line height.
length	A fixed line height in px, pt, cm, etc.
%	A line height in percent of the current font size.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Hover - The :hover selector is used to select elements when you mouse over them.

Position - The position property specifies the type of positioning method used for an element.

Value	**Description**
static	Default value. Elements render in order, as they appear in the document flow.
absolute	The element is positioned relative to its first positioned (not static) ancestor element.
fixed	The element is positioned relative to the browser window.
relative	The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position.
sticky	The element is positioned based on the user's scroll position A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Float - The float property specifies how an element should float.

Value	Description
none	The element does not float, (will be displayed just where it occurs in the text). This is default.
left	The element floats to the left of its container.
right	The element floats the right of its container.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Z-Index - The z-index property specifies the stack order of an element. An element with greater stack order is always in front of an element with a lower stack order.

Display - The display property specifies the display behavior (the type of rendering box) of an element.

Value	Description
inline	Displays an element as an inline element (like <span>). Any height and width properties will have no effect.
block	Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width.
contents	Makes the container disappear, making the child element's children of the element the next level up in the DOM.
flex	Displays an element as a block-level flex container.
grid	Displays an element as a block-level grid container.
inline-block	Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values.
inline-flex	Displays an element as an inline-level flex container.
inline-grid	Displays an element as an inline-level grid container.
inline-table	The element is displayed as an inline-level table.

list-item	Let the element behave like a <li> element.
run-in	Displays an element as either block or inline, depending on context.
table	Let the element behave like a <table> element.
table-caption	Let the element behave like a <caption> element.
table-column-group	Let the element behave like a <colgroup> element.
table-header-group	Let the element behave like a <head> element.
table-footer-group	Let the element behave like a <foot> element.
table-row-group	Let the element behave like a <body> element.
table-cell	Let the element behave like a <td> element.
table-column	Let the element behave like a <col> element.
table-row	Let the element behave like a <tr> element.
none	The element is completely removed.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Complementary colors - Pairs of colors which, when combined or mixed, cancel each other out (lose hue) by producing a grayscale color like white or black. When placed next to each other, they create the strongest contrast for those two colors.

Tertiary colors - Tertiary colors are combinations of primary and secondary colours. There are six tertiary colors; red-orange, yellow-orange, yellow-green, blue-green, blue-violet, and red-violet.

Hue - the attribute of a color by virtue of which it is discernible as red, green, etc., and which is dependent on its dominant wavelength and independent of intensity or lightness.

Saturation - The "colorfulness of an area judged in proportion to its brightness", which in effect is the perceived freedom from whitishness of the light coming from the area.

Lightness - also known as value or tone, is a representation of variation in the perception of a color or colorspace's brightness.

Linear-gradient function - The linear-gradient() function sets a linear gradient as the background image.
To create a linear gradient you must define at least two color stops. Color stops are the colors among which you want to render smooth transitions. You can also set a starting point and a direction (or an angle) along with the gradient effect.

```
Ex:
        background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
```

Transform - The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.

Value	**Description**
none	Defines that there should be no transformation.
matrix(n,n,n,n,n,n)	Defines a 2D transformation, using a matrix of six values.
matrix3d (n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)	Defines a 3D transformation, using a 4x4 matrix of 16 values.
translate(x,y)	Defines a 2D translation.
translate3d(x,y,z)	Defines a 3D translation.
translateX(x)	Defines a translation, using only the value for the X-axis.
translateY(y)	Defines a translation, using only the value for the Y-axis.
translateZ(z)	Defines a 3D translation, using only the value for the Z-axis.
scale(x,y)	Defines a 2D scale transformation.
scale3d(x,y,z)	Defines a 3D scale transformation.

scaleX(x)	Defines a scale transformation by giving a value for the X-axis.
scaleY(y)	Defines a scale transformation by giving a value for the Y-axis.
scaleZ(z)	Defines a 3D scale transformation by giving a value for the Z-axis.
rotate(angle)	Defines a 2D rotation, the angle is specified in the parameter.
rotate3d(x,y,z,angle)	Defines a 3D rotation.
rotateX(angle)	Defines a 3D rotation along the X-axis.
rotateY(angle)	Defines a 3D rotation along the Y-axis.
rotateZ(angle)	Defines a 3D rotation along the Z-axis.
skew(x-angle,y-angle)	Defines a 2D skew transformation along the X- and the Y-axis.
skewX(angle)	Defines a 2D skew transformation along the X-axis.
skewY(angle)	Defines a 2D skew transformation along the Y-axis.
perspective(n)	Defines a perspective view for a 3D transformed element.

initial	Sets this property to its default value. The initial keyword is used to set a CSS property to its default value.
	The initial keyword can be used for any CSS property, and on any HTML element.
	Example:
	Set the text color of the <div> element to red, but keep the initial color for <h1> elements:
	`div {`
	`        color: red;`
	`}`
	
	`h1 {`

	color: initial;
	}

inherit	Inherits this property from its parent element.

@keyframes - The @keyframes rule specifies the animation code. The animation is created by gradually changing from one set of CSS styles to another. During the animation, you can change the set of CSS styles many times. Specify when the style change will happen in percent, or with the keywords "from" and "to", which is the same as 0% and 100%. 0% is the beginning of the animation. 100% is when the animation is complete.

Value	**Description**
animationname	Required. Defines the name of the animation.
keyframes-selector	Required. Percentage of the animation duration. Legal values: 0-100% from (same as 0%) to (same as 100%) Note: You can have many keyframes-selectors in one animation.
css-styles	Required. One or more legal CSS style properties.

Animation - The animation property is a shorthand property for:

Value	**Description**
animation-name	Specifies the name of the keyframe you want to bind to the selector.
animation-duration	Specifies how many seconds or milliseconds an animation takes to complete.
animation-timing-function	Specifies the speed curve of the animation.

animation-delay	Specifies a delay before the animation will start.
animation-iteration-count	Specifies how many times an animation should be played.
animation-direction	Specifies whether or not the animation should play in reverse on alternate cycles.
animation-fill-mode	Specifies what values are applied by the animation outside the time it is executing.
animation-play-state	Specifies whether the animation is running or paused.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Animation-fill-mode - The animation-fill-mode property specifies a style for the element when the animation is not playing (before it starts, after it ends, or both). CSS animations do not affect the element before the first keyframe is played or after the last keyframe is played. The animation-fill-mode property can override this behavior.

Value	**Description**
none	Default value. Animation will not apply any styles to the element before or after it is executing.
forwards	The element will retain the style values that are set by the last keyframe (depends on animation-direction and animation-iteration-count).
backwards	The element will get the style values that are set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period.
both	The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Bezier-function - The cubic-bezier() function defines a Cubic Bezier curve.

A Cubic Bezier curve is defined by four points P0, P1, P2, and P3. P0 and P3 are the start and the end of the curve and, in CSS these points are fixed, as the coordinates are ratios. P0 is (0, 0) and represents the initial time and the initial state, P3 is (1, 1) and represents the final time and the final state.
The cubic-bezier() function can be used with the animation-timing-function property and the transition-timing-function property.

Basic Applied Visual Design Review Questions:

1. What is the difference in a property and the value?

2. What is the font-weight of **bold**?

3. How do you give your page a horizontal line?

4. What property describes the transparency-level? (It’s only 0% to 1%)

5. What selector is used to select elements when you mouse over them?

6. What property applies a 2D or 3D transformation to an element and allows you to you to rotate, scale, move, skew, etc.. elements?

7. What is the Text-transform property?

8. What property can you use to center your text on the page?

9. A website is often divided into _______, ________, ________and a _______. There are tons of different layout designs to choose from. However this is the most common.

10. _________- The art and technique of arranging type to make written language legible, readable, and appealing when displayed.

Introduction to the Applied Accessibility Challenges

"Accessibility" generally means having web content and a user interface that can be understood, navigated, and interacted with by a broad audience. This includes people with visual, auditory, mobility, or cognitive disabilities. Websites should be open and accessible to everyone, regardless of a user's abilities or resources. Some users rely on assistive technology such as a screen reader or voice recognition software. Other users may be able to navigate through a site only using a keyboard. Keeping the needs of various users in mind when developing your project can go a long way toward creating an open web. Here are three general concepts this section will explore throughout the following challenges:

1. Have well-organized code that uses appropriate markup.
2. Ensure text alternatives exist for non-text and visual content.
3. Create an easily-navigated page that's keyboard-friendly.

Having accessible web content is an ongoing challenge. A great resource for your projects going forward is the W3 Consortium's Web Content Accessibility Guidelines (WCAG). They set the international standard for accessibility and provide a number of criteria you can use to check your work.

Form w3.org:

The Web is fundamentally designed to work for all people, whatever their hardware, software, language, location, or ability. When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability.

Thus the impact of disability is radically changed on the Web because the Web removes barriers to communication and interaction that many people face in the physical world. However, when web sites, applications, technologies, or tools are badly designed, they can create barriers that exclude people from using the Web.

Accessibility is essential for developers and organizations who want to create high quality websites and web tools, while not excluding people from using their products and services. The mission of the Web Accessibility Initiative (WAI) is to lead the Web to its full potential to be accessible, enabling people with disabilities to participate equally on the Web.

The Web must be accessible to provide equal access and equal opportunity to people with diverse abilities. Indeed, the UN Convention on the Rights of Persons with Disabilities recognizes access to information and communications technologies, including the Web, as a basic human right.
Accessibility supports social inclusion for people with disabilities as well as others, such as older people, people in rural areas, and people in developing countries.

Accessibility also benefits people without disabilities. The Web Accessibility Perspectives video shows examples of how accessibility is essential for people with disabilities and useful for everyone in a variety of situations.

There is also a strong business case for accessibility. Accessibility overlaps with other best practices such as mobile web design, device independence, multi-modal interaction, usability, design for older users, and search engine optimization (SEO). Case studies show that accessible websites have better search results, reduced maintenance costs, and increased audience reach, among other benefits. Developing a Web Accessibility Business Case for Your Organization details the benefits of web accessibility.

Applied Accessibility Vocabulary:

Applied Accessibility - The extent to which properly designed websites and tools can be used by people with disabilities.

Accessibility - The ability of something to be used by everyone, including those with disabilities.

Web Content - Web content is textual, visual, or aural content that is encountered as part of the user experience on websites. It may include—among other things—text, images, sounds, videos, and animations.

User Interface - the space where interactions between humans and machines occurs. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision-making process.

Mobility - the ability to move or be moved freely and easily. For computer technology, this refers to not being tethered to the same place all the time.

Cognitive Disabilities - a term used when a person has certain limitations in mental functioning and in skills such as communicating, taking care of him or herself, and social skills.

Assistive Technology - any item, piece of equipment, or product system, whether acquired commercially off the shelf, modified, or customized, that is used to increase, maintain, or improve functional capabilities of a person with a disability.

Screen Reader - the generic term for a program that helps blind people use a computer. Simply put, a screen reader will "read" (speak) the content of a page to the blind user.

Voice Recognition - Ability of an electronic security device to recognize the voice (which is unique as a fingerprint) of a particular person.

Software - Software, in its most general sense, is a set of instructions or programs instructing a computer to do specific tasks. Software is a generic term used to describe computer programs. Scripts, applications, programs and a set of instructions are all terms often used to describe software.

W3 Consortium Web Content Accessibility Guidelines (WCAG) - *Listed in a separate document*

Alternate text - Text that is read when an image cannot be displayed to a user.
Ex:

```
<img src="importantLogo.jpeg" alt="Company logo">
```

Headings: <h1> - <h6> elements are headings. They are used like titles. <h1> elements are the largest, and <h6> are the smallest. As the number grows, the headings get smaller.

Main– specifies the main content of a document.
The content inside the <main> element should be unique to the document. It should not contain any content that is repeated across documents such as sidebars, navigation links, copyright information, site logos, and search forms.

Note: There must not be more than one <main> element in a document. The <main> element must NOT be a descendant of an <article>, <aside>, <footer>, <header>, or <nav> element.

Article - The <article> tag specifies independent, self-contained content.

An article should make sense on its own and it should be possible to distribute it independently from the rest of the site.

Potential sources for the <article> element:

- Forum post
- Blog post
- News story
- Comment

Header– represents a container for introductory content or a set of navigational links.
A <header> element typically contains:

- one or more heading elements (<h1> - <h6>)
- logo or icon
- authorship information

You can have several <header> elements in one document.
Note: A <header> tag cannot be placed within a <footer>, <address> or another <header> element.

Nav - defines a set of navigation links.
Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major blocks of navigation links.
Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content.

Footer– defines a footer for a document or section.
A <footer> element should contain information about its containing element.
A <footer> element typically contains:

- authorship information
- copyright information
- contact information
- sitemap
- back to top links
- related documents

You can have several <footer> elements in one document.

Audio - The <audio> tag defines sound, such as music or other audio streams.
Currently, there are 3 supported file formats for the <audio> element: MP3, WAV, and OGG.

Figure - The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

While the content of the <figure> element is related to the main flow, its position is independent of the main flow, and if removed it should not affect the flow of the document.

Label - The <label> tag defines a label for a <button>, <input>, <meter>, <output>, <progress>, <select>, or <textarea> element.
The <label> element does not render as anything special for the user. However, it provides usability improvement for mouse users, because if the user clicks on the text within the <label> element, it toggles the control.

The for attribute of the <label> tag should be equal to the id attribute of the related element to bind them together.

Fieldset - The <fieldset> tag is used to group related elements in a form.
The <fieldset> tag draws a box around the related elements.

Time - The <time> tag defines a human-readable date/time. This element can also be used to encode dates and times in a machine-readable way so that user agents can offer to add birthday reminders or scheduled events to the user's calendar. Additionally search engines can produce smarter search results.

Contrast - The difference in luminance or colour that makes an object (or its representation in an image or display) distinguishable. In visual perception of the real world, contrast is determined by the difference in the color and brightness of the object and other objects within the same field of view.

Access Key - The accesskey attribute specifies a shortcut key to activate/focus an element.

Ex:
```
        <a href="https://www.w3schools.com/html5" accesskey="h">HTML5</a><br>
        <a href="https://www.w3schools.com/css3" accesskey="c">CSS3</a>
```

Tabindex - The tabindex attribute specifies the tab order of an element (when the "tab" button is used for navigating).

Ex:
```
        <a href="https://www.w3schools.com/" tabindex="2">W3Schools</a>
        <a href="http://www.google.com/" tabindex="1">Google</a>
        <a href="http://www.microsoft.com/" tabindex="3">Microsoft</a>
```

Basic Applied Accessibility Review Questions:

1. What is the correct attribute that you should use inside the <img> tag, in case the image doesn't load correctly. So that people will know what is supposed to be there?

2. What is accessibility?

3. How would you make your website so that it will be accessible to the blind or dissabled?

4. What element would you use to input a time/date?

5. The <label> tag defines a label for _________, ________, _________, _________, ___________, _________ or ___________.

6. What tag is used to group related elements in a form and draws a box around the related elements?

7. What tag is to be used for mostly sound/music like; MP3, WAV, and OGG's?

8. What tag should you use for your navigation links?

9. What tag should you use for the main content of your page? (One can ONLY have one.)

10. Why is it important for you to design your website to be accessible?

Introduction to the Responsive Web Design Challenges

Today, there are many types of devices that can access the web. They range from large desktop computers to small mobile phones. These devices have different screen sizes, resolutions, and processing power. Responsive Web Design is an approach to designing web content that responds to the constraints of different devices. The page structure and CSS rules should be flexible to accommodate these differences. In general, design the page's CSS to your target audience. If you expect most of your traffic to be from mobile users, take a 'mobile-first' approach. Then add conditional rules for larger screen sizes. If your visitors are desktop users, then design for larger screens with conditional rules for smaller sizes. CSS gives you the tools to write different style rules, then apply them depending on the device displaying the page. This section will cover the basic ways to use CSS for Responsive Web Design.

Responsive Web Design Vocabulary:

Responsive Web Design - An approach to designing web content that responds to the constraints of different devices.

Desktop Computer - A desktop computer is a personal computer designed for regular use at a single location on or near a desk or table due to its size and power requirements. The most common configuration has a case that houses the power supply, motherboard (a printed circuit board with a microprocessor as the central processing unit (CPU), memory, bus, and other electronic components), disk storage (usually one or more hard disk drives, optical disc drives, and in early models a floppy disk drive), a keyboard and mouse for input, and a computer monitor, speakers, and, often, a printer for output. The case may be oriented horizontally or vertically and placed either underneath, beside, or on top of a desk.

<u>Laptop Computer</u> - A laptop computer (also shortened to just laptop; or called a notebook or notebook computer) is a small, portable personal computer (PC) with a "clamshell" form factor, typically having a thin LCD or LED computer screen mounted on the inside of the upper lid of the clamshell and an alphanumeric keyboard on the inside of the lower lid. The clamshell is opened up to use the computer. Laptops are folded shut for transportation, and thus are suitable for mobile use.

<u>Netbook</u> - Netbook is a generic name given to a category of small, lightweight, legacy-free, and inexpensive laptop computers that were introduced in 2007. Netbooks compete in the

same market segment as mobiles and Chromebooks (a variation on the portable network computer).

Tablet - A tablet, or tablet PC, is a portable computer that uses a touchscreen as its primary input device. Most tablets are slightly smaller and weigh less than the average laptop. While some tablets include fold out keyboards, others, such as the Apple iPad and Motorola Xoom, only offer touchscreen input.

Smart-phone - Smartphones are a class of mobile phones and of multi-purpose mobile computing devices. They are distinguished from feature phones by their stronger hardware capabilities and extensive mobile operating systems, which facilitate wider software, internet (including web browsing over mobile broadband), and multimedia functionality (including music, video, cameras, and gaming), alongside core phone functions such as voice calls and text messaging. Smartphones typically include various sensors that can be leveraged by their software, such as a magnetometer, proximity sensors, barometer, gyroscope and accelerometer, and support wireless communications protocols such as Bluetooth, Wi-Fi, and satellite navigation.

Smartwatch - A smartwatch is a wearable computer in the form of a wristwatch; modern smartwatches provide a local touchscreen interface for daily use, while an associated smartphone app provides for management and telemetry (such as long-term biomonitoring). While early models could perform basic tasks, such as calculations, digital time telling, translations, and game-playing, 2010s smartwatches have more general functionality closer to smartphones, including mobile apps, a mobile operating system and WiFi/Bluetooth connectivity. Some smartwatches function as portable media players, with FM radio and playback of digital audio and video files via a Bluetooth headset. Some models, called 'watch phones' (or vice versa), have mobile cellular functionality like making calls.

Smart TV - A smart TV is a traditional television set with integrated Internet and interactive "Web 2.0" features which allows users to stream music and videos, browse the internet, and view photos. Smart TV is a technological convergence of computers, television sets and set-top boxes. Besides the traditional functions of television sets and set-top boxes provided through traditional broadcasting media, these devices can also provide Internet TV, online interactive media, over-the-top content (OTT), as well as on-demand streaming media, and home networking access.

<u>Resolution</u> - The display resolution or display modes of a digital television, computer monitor or display device is the number of distinct pixels in each dimension that can be displayed. It can be an ambiguous term, especially as the displayed resolution is controlled by different factors in cathode ray tube (CRT) displays, flat-panel displays (including liquid-crystal displays) and projection displays using fixed picture-element (pixel) arrays.

<u>Pixel</u> - A minute area of illumination on a display screen, one of many from which an image is composed.

<u>Processing Power</u> - Often known as CPU power, CPU cycles, and various other names, processing power is the ability of a computer to manipulate data.
<u>Mobile-First Approach</u> - designing an online experience for mobile before designing it for the desktop Web—or any other device. In the past, when users' focus was on the desktop Web, mobile design was an afterthought.

<u>Conditional Rules</u> - CSS conditional rules allow web designers to define a set of rules that will only apply based on the capabilities of the device being used to view the content.

Style Rules - The style enforced by conditional rules.

Media Query - Media query is a CSS technique introduced in CSS3. It uses the @media rule to include a block of CSS properties only if a certain condition is true.

Ex: If the browser window is 600px or smaller, the background color will be light-blue:

```
@media only screen and (max-width: 600px) {
  body {
      background-color: lightblue;
  }
}
```

Max-width - The max-width property defines the maximum width of an element.
If the content is larger than the maximum width, it will automatically change the height of the element.

If the content is smaller than the maximum width, the max-width property has no effect.
Note: This prevents the value of the width property from becoming larger than max-width.
The value of the max-width property overrides the width property.

CSS Syntax:

```
max-width: none|length|initial|inherit;
```

Viewport Units - There are four viewport based units in CSS. These are vh, vw, vmin and vmax.

- Viewport Height (vh) — This unit is based on the height of the viewport. A value of 1vh is equal to 1% of the viewport height.

- Viewport Width (vw) — This unit is based on the width of the viewport. A value of 1vw is equal to 1% of the viewport width.

- Viewport Minimum (vmin) — This unit is based on the smaller dimension of the viewport. If the viewport height is smaller than the width, the value of 1vmin will be equal to 1% of the viewport height. Similarly, if the viewport width is smaller than the height, the value of 1vmin will be equal to 1% of the viewport width.

- Viewport Maximum (vmax) — This unit is based on the larger dimension of the viewport. If the viewport height is larger than the width, the value of 1vmax will be equal to 1% of viewport height. Similarly, if the viewport width is larger than the height, the value of 1vmax will be equal to 1% of the viewport width.

Basic Responsive Web Design Review Questions

1. What is Responsive Web Design?

2. What is Pixel?

3. Define Mobile-First-Approach.

4. What are the four viewport based units in CSS?

5. What are style rules?

Introduction to the CSS Flexbox Challenges

A website's User Interface ("UI") has two components. First, there are the visual elements, such as colors, fonts, and images. Second, there is the placement or positioning of those elements. In Responsive Web Design, a UI layout must accommodate many different browsers and devices accessing the content.

CSS3 introduced Flexible Boxes, or flexbox, to create page layouts for a dynamic UI. It is a layout mode that arranges elements in a predictable way for different screen sizes and browsers. While somewhat new, all popular modern browsers support flexbox. This section covers how to use flexbox and the different layout options it offers.

CSS Flexbox Vocabulary:

CSS Flexbox - The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

Font - the combination of typeface and other qualities, such as size, pitch, and spacing of text.

Flexible Boxes - The Flexible Box Module, usually referred to as flexbox, was designed as a one-dimensional layout model, and as a method that could offer space distribution between items in an interface and powerful alignment capabilities.

Layout Mode - A CSS layout mode, sometimes simply called “layout”, is an algorithm that determines the position and size of boxes based on the way they interact with their sibling and ancestor boxes.

Flex-direction - The flex-direction property defines in which direction the container wants to stack the flex items.

```
CSS Syntax:
flex-direction: column | column-reverse | row | row-reverse;
```

Flex-wrap - The flex-wrap property specifies whether the flex items should wrap or not.

```
CSS Syntax:
flex-wrap: wrap | nowrap | wrap-reverse;
```

Flex-flow - The flex-flow property is a shorthand property for setting both the flex-direction and flex-wrap properties.

```
CSS Syntax:
flex-flow: row-wrap;
```

Justify-content - The justify-content property is used to align the flex items.

```
CSS Syntax:
justify-content: center| flex-start | flex-end | space-around | space-between;
```

Align-items - The align-items property is used to align the flex items vertically.

```
CSS Syntax:
align-items: center| flex-start | flex-end | stretch | baseline;
```

Align-content - The align-content property is used to align the flex lines.

```
CSS Syntax:
align-content: space-between| space-around | stretch | center | flex-start | flex-end;
```

Order - The order property specifies the order of the flex items.

Flex-grow - The flex-grow property specifies how much a flex item will grow relative to the rest of the flex items

Flex-shrink - The flex-shrink property specifies how much a flex item will shrink relative to the rest of the flex items.

Flex-basis - The flex-basis property specifies the initial length of a flex item.

Flex - The flex property is a shorthand property for the flex-grow, flex-shrink, and flex-basis properties.

Align-self - The align-self property specifies the alignment for the selected item inside the flexible container. The align-self property overrides the default alignment set by the container's align-items property.

Child Elements - The direct child elements of a flex container automatically becomes flexible (flex) items.

Introduction to the CSS Grid Challenges

CSS Grid helps you easily build complex web designs. It works by turning an HTML element into a grid container with rows and columns for you to place “children elements” where you want within the grid.

CSS Grid Vocabulary:

CSS Grid - The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

Grid Container - To make an HTML element behave as a grid container, you have to set the display property to grid or inline-grid.
Grid containers consist of grid items, placed inside columns and rows.

Children Elements - Items within a grid container.

Grid Columns - The vertical line of grid items are called columns.

Grid Rows - The horizontal line of grid items are called rows.

Grid Gaps - The space between each column/row are called gaps.

```
CSS Syntax:
        .grid-container {
          display: grid;
          grid-column-gap: 50px;
          grid-row-gap: 50px;
        }
OR:
        .grid-container {
          display: grid;
          grid-gap: 50px 100px;
        }
OR:
        .grid-container {
          display: grid;
          grid-gap: 50px;
        }
```

Grid Lines - The line between columns are called column lines. The line between rows are called row lines.
Grid-template-columns - The grid-template-columns property specifies the number (and the widths) of columns in a grid layout. The values are a space separated list, where each value specifies the size of the respective column.

```
CSS Syntax:
        grid-template-columns: none|auto|max-content|min-content|length|initial|inherit;
```

Grid-template-rows - The grid-template-rows property specifies the number (and the heights) of the rows in a grid layout. The values are a space-separated list, where each value specifies the height of the respective row.

```
CSS Syntax:
        grid-template-rows: none|auto|max-content|min-content|length|initial|inherit;
```

CSS Grid Units:

fr: sets the column or row to a fraction of the available space,
auto: sets the column or row to the width or height of its content automatically,
%: adjusts the column or row to the percent width of its container.

Grid-column - The grid-column property specifies a grid item's size and location in a grid layout, and is a shorthand property for the following properties:

- grid-column-start
- Grid-column-end

CSS Syntax:
```
        grid-column: grid-column-start / grid-column-end;
```

Grid-row - The grid-row property specifies a grid item's size and location in a grid layout, and is a shorthand property for the following properties::

- grid-row-start
- grid-row-end

CSS Syntax:
```
        grid-row: grid-row-start / grid-row-end;
```

Justify-self - The CSS justify-self property set the way a box is justified inside its alignment container along the appropriate axis.

CSS Syntax:
```
        justify-self: stretch | center | start | end;
```

Align-self - The align-self CSS property aligns flex items of the current flex line overriding the align-items value. If any of the item's cross-axis margin is set to auto, then align-self is ignored. In Grid layout align-self aligns the item inside the grid area.

CSS Syntax:
```
        align-self: stretch | center | start | end;
```

Justify-items - The CSS justify-items property defines the default justify-self for all items of the box, giving them all a default way of justifying each box along the appropriate axis.

CSS Syntax:
```
        justify-items: stretch | center | start | end;
```

Align-items - The CSS align-items property sets the align-self value on all direct children as a group. The align-self property sets the alignment of an item within its containing block. In Flexbox it controls the alignment of items on the Cross Axis; in Grid Layout it controls the alignment of items on the Block Axis within their grid area.

CSS Syntax:
```
        align-items: stretch | center | start | end;
```

Grid-template-areas - The grid-template-areas property specifies areas within the grid layout.
You can name grid items by using the grid-area property, then reference to the name in the grid-template-areas property.
Each area is defined by apostrophes. Use a period sign to refer to a grid item with no name.

```
CSS Syntax:
        grid-template-areas: none|item names;
```

Grid-area - The grid-area property specifies a grid item's size and location in a grid layout, and is a shorthand property for the following properties:

- grid-row-start
- grid-column-start
- grid-row-end
- grid-column-end

The grid-area property can also be used to assign a name to a grid item. Named grid items can then be referenced to by the grid-template-areas property of the grid container.

```
CSS Syntax:
grid-area: grid-row-start / grid-column-start / grid-row-end / grid-column-end | itemname;
```

Value	Description
grid-row-start	Specifies on which row to start displaying the item.
grid-column-start	Specifies on which column to start displaying the item.
grid-row-end	Specifies on which row-line to stop displaying the item, or how many rows to span.
grid-column-end	Specifies on which column-line to stop displaying the item, or how many columns to span.
itemname	Specifies a name for the grid item.

Basic CSS Flexbox and CSS Grid Review Questions

1. What is CSS Flexbox?

2. What is the flex-direction property?

3. What is a Layout Mode?

4. What does UI stand for?

5. How many components does a website's UI have?

6. What is CSS Grid?

7. Using justify-items property. What order is the syntax?

8. What value specifies on which row to start displaying the item?

9. What value specifies on which column-line to stop displaying the item, or how many items to span?

10. What value specifies on which row-line to stop displaying the item, or how many rows to span?

Introduction to the Responsive Web Design Projects

Time to put your newly learned skills to work! By working on projects you will have the opportunity to apply all the skills, principles, and concepts you have learnt so far: HTML, CSS, Visual Design, Accessibility, etc.
In this section you get the chance to:

- Build a Tribute Page
- Build a Survey Form
- Build a Product Landing Page
- Build a Technical Documentation Page
- Build a Personal Portfolio Webpage

By the end of this exercise, you will have 5 responsive websites under your belt that you can show off to friends, family, employers, etc. Have fun, and remember to use the Read-Search-Ask method if you get stuck.

Projects:

Tribute Page

Objective: Fulfill the below user stories and get all of the tests to pass. Give it your own personal style.

- You can use HTML, JavaScript, and CSS to complete this project. Plain CSS is recommended because that is what the lessons have covered so far and you should get some practice with plain CSS. You can use Bootstrap or SASS if you choose. Additional technologies (for example: jQuery, React, Angular, or Vue) are not recommended for this project, and using them is at your own risk. Other projects will give you a chance to work with different technology stacks like React. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
- **User Story #1:** My tribute page should have an element with a corresponding id="main", which contains all other elements.
- **User Story #2:** I should see an element with a corresponding id="title", which contains a string (i.e. text) that describes the subject of the tribute page (e.g. "Dr. Norman Borlaug").
- **User Story #3:** I should see a div element with a corresponding id="img-div".
- **User Story #4:** Within the img-div element, I should see an img element with a corresponding id="image".
- **User Story #5:** Within the img-div element, I should see an element with a corresponding id="img-caption" that contains textual content describing the image shown in img-div.
- **User Story #6:** I should see an element with a corresponding id="tribute-info", which contains textual content describing the subject of the tribute page.
- **User Story #7:** I should see an <a> element with a corresponding id="tribute-link", which links to an outside site that contains additional information about the subject of the tribute page. HINT: You must give your element an attribute of target and set it to _blank in order for your link to open in a new tab (i.e. target="_blank").
- **User Story #8:** The img element should responsively resize, relative to the width of its parent element, without exceeding its original size.
- **User Story #9:** The img element should be centered within its parent element.

Survey Form

Objective: Fulfill the below user stories and get all of the tests to pass. Give it your own personal style.

- You can use HTML, JavaScript, and CSS to complete this project. Plain CSS is recommended because that is what the lessons have covered so far and you should get some practice with plain CSS. You can use Bootstrap or SASS if you choose. Additional technologies (for example: jQuery, React, Angular, or Vue) are not recommended for this project, and using them is at your own risk. Other projects will give you a chance to work with different technology stacks like React. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!

- **User Story #1:** I can see a title with id="title" in H1 sized text.
- **User Story #2:** I can see a short explanation with id="description" in P sized text.
- **User Story #3:** I can see a form with id="survey-form".
- **User Story #4:** Inside the form element, I am required to enter my name in a field with id="name".
- **User Story #5:** Inside the form element, I am required to enter an email in a field with id="email".
- **User Story #6:** If I enter an email that is not formatted correctly, I will see an HTML5 validation error.
- **User Story #7:** Inside the form, I can enter a number in a field with id="number".
- **User Story #8:** If I enter non-numbers in the number input, I will see an HTML5 validation error.
- **User Story #9:** If I enter numbers outside the range of the number input, which are defined by the min and max attributes, I will see an HTML5 validation error.
- **User Story #10:** For the name, email, and number input fields inside the form I can see corresponding labels that describe the purpose of each field with the following ids: id="name-label", id="email-label", and id="number-label".
- **User Story #11:** For the name, email, and number input fields, I can see placeholder text that gives me a description or instructions for each field.
- **User Story #12:** Inside the form element, I can select an option from a dropdown that has a corresponding id="dropdown".
- **User Story #13:** Inside the form element, I can select a field from one or more groups of radio buttons. Each group should be grouped using the name attribute.
- **User Story #14:** Inside the form element, I can select several fields from a series of checkboxes, each of which must have a value attribute.
- **User Story #15:** Inside the form element, I am presented with a textarea at the end for additional comments.
- **User Story #16:** Inside the form element, I am presented with a button with id="submit" to submit all my inputs.

Product Landing Page

- **Objective:** Fulfill the below user stories and get all of the tests to pass. Give it your own personal style.
- You can use HTML, JavaScript, and CSS to complete this project. Plain CSS is recommended because that is what the lessons have covered so far and you should get some practice with plain CSS. You can use Bootstrap or SASS if you choose. Additional technologies (for example: jQuery, React, Angular, or Vue) are not recommended for this project, and using them is at your own risk. Other projects will give you a chance to work with different technology stacks like React. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
- **User Story #1:** My product landing page should have a header element with a corresponding id="header".

- **User Story #2:** I can see an image within the header element with a corresponding id="header-img". A company logo would make a good image here.
- **User Story #3:** Within the #header element I can see a nav element with a corresponding id="nav-bar".
- **User Story #4:** I can see at least three clickable elements inside the nav element, each with the class nav-link.
- **User Story #5:** When I click a nav-link button in the nav element, I am taken to the corresponding section of the landing page.
- **User Story #6:** I can watch an embedded product video with id="video".
- **User Story #7:** My landing page has a form element with a corresponding id="form".
- **User Story #8:** Within the form, there is an input field with id="email" where I can enter an email address.
- **User Story #9:** The #email input field should have a placeholder text to let the user know what the field is for.
- **User Story #10:** The #email input field uses HTML5 validation to confirm that the entered text is an email address.
- **User Story #11:** Within the form, there is a submit input with a corresponding id="submit".
- **User Story #12:** When I click the #submit element, the email is submitted to a static page (use this mock URL: https://www.freecodecamp.com/email-submit) that confirms the email address was entered and that it posted successfully.
- **User Story #13:** The navbar should always be at the top of the viewport.
- **User Story #14:** My product landing page should have at least one media query.
- **User Story #15:** My product landing page should utilize CSS flexbox at least once.

Technical Documentation Page

- **Objective:** Fulfill the below user stories and get all of the tests to pass. Give it your own personal style.
- You can use HTML, JavaScript, and CSS to complete this project. Plain CSS is recommended because that is what the lessons have covered so far and you should get some practice with plain CSS. You can use Bootstrap or SASS if you choose. Additional technologies (for example: jQuery, React, Angular, or Vue) are not recommended for this project, and using them is at your own risk. Other projects will give you a chance to work with different technology stacks like React. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
- **User Story #1:** I can see a main element with a corresponding id="main-doc", which contains the page's main content (technical documentation).
- **User Story #2:** Within the #main-doc element, I can see several section elements, each with a class of main-section. There should be a minimum of 5.
- **User Story #3:** The first element within each main-section should be a header element which contains text that describes the topic of that section.
- **User Story #4:** Each section element with the class of main-section should also have an id that corresponds with the text of each header contained within it. Any spaces should be replaced with underscores (e.g. The section that contains the

header "Javascript and Java" should have a corresponding id="Javascript_and_Java").

- **User Story #5:** The main-section elements should contain at least 10 p elements total (not each).
- **User Story #6:** The main-section elements should contain at least 5 code elements total (not each).
- **User Story #7:** The main-section elements should contain at least 5 li items total (not each).
- **User Story #8:** I can see a nav element with a corresponding id="navbar".
- **User Story #9:** The navbar element should contain one header element which contains text that describes the topic of the technical documentation.
- **User Story #10:** Additionally, the navbar should contain link (a) elements with the class of nav-link. There should be one for every element with the class main-section.
- **User Story #11:** The header element in the navbar must come before any link (a) elements in the navbar.
- **User Story #12:** Each element with the class of nav-link should contain text that corresponds to the header text within each section(e.g. if you have a "Hello world" section/header, your navbar should have an element which contains the text "Hello world").
- **User Story #13:** When I click on a navbar element, the page should navigate to the corresponding section of the main-doc element (e.g. If I click on a nav-link element that contains the text "Hello world", the page navigates to a section element that has that id and contains the corresponding header.
- **User Story #14:** On regular sized devices (laptops, desktops), the element with id="navbar" should be shown on the left side of the screen and should always be visible to the user.
- **User Story #15:** My Technical Documentation page should use at least one media query.

<u>Personal Portfolio Webpage</u>

- **Objective:** Fulfill the below user stories and get all of the tests to pass. Give it your own personal style.
- You can use HTML, JavaScript, and CSS to complete this project. Plain CSS is recommended because that is what the lessons have covered so far and you should get some practice with plain CSS. You can use Bootstrap or SASS if you choose. Additional technologies (for example: jQuery, React, Angular, or Vue) are not recommended for this project, and using them is at your own risk. Other projects will give you a chance to work with different technology stacks like React. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
- **User Story #1:** My portfolio should have a welcome section with an id of welcome-section.
- **User Story #2:** The welcome section should have an h1 element that contains text.
- **User Story #3:** My portfolio should have a projects section with an id of projects.
- **User Story #4:** The projects section should contain at least one element with a class of project-tile to hold a project.

- **User Story #5:** The projects section should contain at least one link to a project.
- **User Story #6:** My portfolio should have a navbar with an id of navbar.
- **User Story #7:** The navbar should contain at least one link that I can click on to navigate to different sections of the page.
- **User Story #8:** My portfolio should have a link with an id of profile-link, which opens my GitHub or FCC profile in a new tab.
- **User Story #9:** My portfolio should have at least one media query.
- **User Story #10:** The height of the welcome section should be equal to the height of the viewport.
- **User Story #11:** The navbar should always be at the top of the viewport.

Pick One Project To Do and Turn It In To Your Instructor