Basic HTML Review

HTML Basics

- **Role of HTML**: HTML represents the content and structure of the web page.
- **HTML Elements**: Elements are the building blocks for an HTML document. They represent headings, paragraphs, links, images and more. Most HTML elements consist of an opening tag (`<elementName>`) and a closing tag (`</elementName>`).

Here is the basic syntax:

```
<elementName>Content goes here</elementName>
```

- **Void Elements**: Void elements cannot have any content and only have a start tag. Examples include `img` and `meta` elements.

```
<img>
<meta>
```

It is common to see some codebases that include a forward slash `/` inside the void element. Both of these are acceptable:

```
<img>
<img/>
```

- **Attributes**: An attribute is a value placed inside the opening tag of an HTML element. Attributes provide additional information about the element or specify how the element should behave. Here is the basic syntax for an attribute:

```
<element attribute="value"></element>
```

A boolean attribute is an attribute that can either be present or absent in an HTML tag. If present, the value is true otherwise it is false. Examples of boolean attributes include `disabled`, `readonly`, and `required`.

- **Comments**: Comments are used in programming to leave notes for yourself and other developers in your code. Here is the syntax for a comment in HTML:

```
<!--This is an HTML comment.-->
```

Common HTML elements

- **Heading Elements**: There are six heading elements in HTML. The `h1` through `h6` heading elements are used to signify the importance of content below them. The lower the number, the higher the importance, so `h2` elements have less importance than `h1` elements.

```
<h1>most important heading element</h1>
<h2>second most important heading element</h2>
<h3>third most important heading element</h3>
<h4>fourth most important heading element</h4>
<h5>fifth most important heading element</h5>
<h6>least important heading element</h6>
```

- **Paragraph Elements**: This is used for paragraphs on a web page.

```
<p>This is a paragraph element.</p>
```

- `img` **Elements**: The `img` element is used to add images to the web page. The `src` attribute is used to specify the location for that image. For image elements, it's good practice to include another attribute called the `alt` attribute. Here's an example of an `img` element with the `src` and `alt` attributes:

```
<img src="https://cdn.freecodecamp.org/curriculum/cat-photo-app/lasagna.jpg" alt="A slice of lasagna on a plate.">
```

- `body` **Element**: This element is used to represent the content for the HTML document.

```
<body>
  <h1>CatPhotoApp</h1>
  <p>This is a paragraph element.</p>
</body>
```

- **`section` Elements**: This element is used to divide up content into smaller sections.

```
<section>
  <h2>About Me</h2>
  <p>Hi, I am Jane Doe and I am a web developer.</p>
</section>
```

- **`div` Elements**: This element is a generic HTML element that does not hold any semantic meaning. It is used as a generic container to hold other HTML elements.

```
<div>
  <h1>I am a heading</h1>
  <p>I am a paragraph</p>
</div>
```

- **Anchor (`a`) Elements**: These elements are used to apply links to a web page. The `href` attribute is used to specify where the link should go when the user clicks on it.

```
<a href="https://cdn.freecodecamp.org/curriculum/cat-photo-app/running-cats.jpg">cute cats</a>
```

- **Unordered (`ul`) and Ordered (`ol`) List Elements**: To create a bulleted list of items you should use the `ul` element with one or more `li` elements nested inside like this:

```
<ul>
  <li>catnip</li>
  <li>laser pointers</li>
  <li>lasagna</li>
</ul>
```

To create an ordered list of items you should use the `ol` element:

```
<ol>
  <li>flea treatment</li>
  <li>thunder</li>
  <li>other cats</li>
</ol>
```

- **Emphasis (`em`) Element**: This is used to place emphasis on a piece of text.

```
<p>Cats <em>love</em> lasagna.</p>
```

- **Strong Importance (`strong`) Element**: This element is used to place strong emphasis on text to indicate a sense of urgency and seriousness.

```
<p>
  <strong>Important:</strong> Before proceeding, make sure to wear your safety goggles.
</p>
```

- **`figure` and `figcaption` Elements**: The `figure` element is used to group content like images and diagrams. The `figcaption` element is used to represent a caption for that content inside the `figure` element.

```
<figure>
  <img src="https://cdn.freecodecamp.org/curriculum/cat-photo-app/cats.jpg" alt="Two tabby kittens sleeping together on a couch.">
  <figcaption>Cats <strong>hate</strong> other cats.</figcaption>
</figure>
```

- **`main` Element**: This element is used to represent the main content for a web page.
- **`footer` Element**: This element is placed at the bottom of the HTML document and usually contains copyright information and other important page links.

```
<footer>
  <p>
    No Copyright - <a
href="https://www.freecodecamp.org">freeCodeCamp.org</a>
  </p>
</footer>
```

Identifiers and Grouping

- **IDs**: Unique element identifiers for HTML elements. ID names should only be used once per HTML document.

```
<h1 id="title">Movie Review Page</h1>
```

ID names cannot have spaces. If your ID name contains multiple words then you can use dashes between the words like this:

```
<div id="red-box"></div>
```

- **Classes**: Classes are used to group elements for styling and behavior.

```
<div class="box"></div>
```

Unlike IDs, you can reuse the same class name throughout the HTML document. The `class` value can also have spaces like this:

```
<div class="box red-box"></div>
<div class="box blue-box"></div>
```

Special Characters and Linking

- **HTML entities**: An HTML entity, or character reference, is a set of characters used to represent a reserved character in HTML. Examples include the ampersand (`&`) symbol and the less than symbol (`<`).

```
<p>This is an &lt;img /&gt; element</p>
```

- `link` **Element**: This element is used to link to external resources like stylesheets and site icons. Here is the basic syntax for using the `link` element for an external CSS file:

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

The `rel` attribute is used to specify the relationship between the linked resource and the HTML document. The `href` attribute is used to specify the location of the URL for the external resource.

- `script` **Element**: This element is used to embed executable code.

```
<body>
  <script>
    alert("Welcome to freeCodeCamp");
  </script>
</body>
```

While you can technically write all of your JavaScript code inside the `script` tags, it is considered best practice to link to an external JavaScript file instead. Here is an example of using the `script` element to link to an external JavaScript file:

```
<script src="path-to-javascript-file.js"></script>
```

The `src` attribute is used here to specify the location for that external JavaScript file.

Boilerplate and Encoding

- **HTML boilerplate**: This is a boilerplate that includes the basic structure and essential elements every HTML document needs.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
```

```
    <title>freeCodeCamp</title>
    <link rel="stylesheet" href="./styles.css" />
  </head>
  <body>
    <!--Headings, paragraphs, images, etc. go inside here-->
  </body>
</html>
```

- `DOCTYPE`: This is used to tell browsers which version of HTML you're using.
- `html` **Element**: This represents the top level element or the root of an HTML document. To specify the language for the document, you should use the `lang` attribute.
- `head` **Element**: The `head` section contains important meta data which is behind-the-scenes information needed for browsers and search engines.
- `meta` **Elements**: These elements represent your site's metadata. These element have details about things like character encoding, and how websites like Twitter should preview your page's link and more.
- `title` **Element**: This element is used to set the text that appears in the browser tab or window.
- **UTF-8 character encoding**: UTF-8, or UCS Transformation Format 8, is a standardized character encoding widely used on the web. Character encoding is the method computers use to store characters as data.
  The `charset` attribute is used inside of a `meta` element to set the character encoding to UTF-8.

SEO and Social Sharing

- **SEO**: Search Engine Optimization is a practice that optimizes web pages so they become more visible and rank higher on search engines.
- **Meta (`description`) Element**: This is used to provide a short description for the web page and impacting SEO.

```
<meta
  name="description"
```

```
  content="Discover expert tips and techniques for gardening in small spaces, choosing the right plants, and maintaining a thriving garden."
/>
```

- **Open Graph tags**: The open graph protocol enables you to control how your website's content appears across various social media platforms, such as Facebook, LinkedIn, and many more.

By setting these open graph properties, you can entice users to want to click and engage with your content. You can set these properties through a collection of `meta` elements inside your HTML `head` section.

- `og:title` **Property**: This is used to set the title that displays for social media posts.

```
<meta content="freeCodeCamp.org" property="og:title" />
```

- `og:type` **Property**: The `type` property is used to represent the type of content being shared on social media. Examples of this content include articles, websites, videos, or music.

```
<meta property="og:type" content="website" />
```

- `og:image` **Property**: This is used to set the image shown for social media posts.

```
<meta

content="https://cdn.freecodecamp.org/platform/universal/fcc_meta_1920X1080-indigo.png"
  property="og:image"
/>
```

- `og:url` **Property**: This is used to set the URL that users will click on for the social media posts.

```
<meta property="og:url" content="https://www.freecodecamp.org" />
```

Media Elements and Optimization

- **Replaced elements**: A replaced element is an element whose content is determined by an external resource rather than by CSS itself. An example would be an `iframe` element. `iframe` stands for inline frame. It's an inline element used to embed other HTML content directly within the HTML page.

```
<iframe src="https://www.example.com" title="Example Site"></iframe>
```

You can include the `allowfullscreen` attribute which allows the user to display the iframe in full screen mode.

```
<iframe
  src="video-url"
  width="width-value"
  height="height-value"
  allowfullscreen
></iframe>
```

To embed a video within an `iframe` you can copy it directly from popular video services like YouTube and Vimeo, or define it yourself with the `src` attribute pointing to the URL of that video. Here's an example of embedding a popular freeCodeCamp course from YouTube:

```
<h1>A freeCodeCamp YouTube Video Embedded with the iframe Element</h1>

<iframe
  width="560"
  height="315"
  src="https://www.youtube.com/embed/PkZNo7MFNFg?si=-UBVIUNM3csdeiWF"
  title="YouTube video player"
```

```
  allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share"
  referrerpolicy="strict-origin-when-cross-origin"
  allowfullscreen
></iframe>
```

There are some other replaced elements, such as `video`, and `embed`. And some elements behave as replaced elements under specific circumstances. Here's an example of an `input` element with the `type` attribute set to `image`:

```
<input type="image" alt="Descriptive text goes here" src="example-img-url">
```

- **Optimizing media**: There are three tools to consider when using media, such as images, on your web pages: the size, the format, and the compression. A compression algorithm is used to reduce the size for files or data.
- **Image formats**: Two of the most common file formats are PNG and JPG, but these are no longer the most ideal formats for serving images. Unless you need support for older browsers, you should consider using a more optimized format, like WEBP or AVIF.
- **Image licenses**: An image under the public domain has no copyright attached to it and is free to be used without any restrictions. Images licensed specifically under the Creative Commons 0 license are considered public domain. Some images might be released under a permissive license, like a Creative Commons license, or the BSD license that freeCodeCamp uses.
- **SVGs**: Scalable Vector Graphics track data based on paths and equations to plot points, lines, and curves. What this really means is that a vector graphic, like an SVG, can be scaled to any size without impacting the quality.

Multimedia Integration

- **`audio` and `video` Elements**: The `audio` and `video` elements allow you to add sound and video content to your HTML documents. The `audio` element supports popular audio formats like mp3, wav, and ogg. The `video` element supports mp4, ogg, and webm formats.

```
<audio src="CrystalizeThatInnerChild.mp3"></audio>
```

If you want to see the audio player on the page, then you can add the `audio` element with the `controls` attribute:

```
<audio src="CrystalizeThatInnerChild.mp3" controls></audio>
```

The `controls` attribute enables users to manage audio playback, including adjusting volume, and pausing, or resuming playback. The `controls` attribute is a boolean attribute that can be added to an element to enable built-in playback controls. If omitted, no controls will be shown.

- `loop` **Attribute**: The `loop` attribute is a boolean attribute that makes the audio replay continuously.

```
<audio
  src="https://cdn.freecodecamp.org/curriculum/js-music-player/can't-stay-down.mp3"
  loop
  controls
></audio>
```

- `muted` **Attribute**: When present in the `audio` element, the `muted` boolean attribute will start the audio in a muted state.

```
<audio
  src="https://cdn.freecodecamp.org/curriculum/js-music-player/can't-stay-down.mp3"
  loop
  controls
  muted
></audio>
```

- `source` **Element**: When it comes to audio file types, there are differences in which browsers support which type. To accommodate this, you can use `source` elements inside the `audio` element and the browser will select the first source that it understands. Here's an example of using multiple `source` elements for an `audio` element:

```
<audio controls>
  <source src="audio.ogg" type="audio/ogg" />
  <source src="audio.wav" type="audio/wav" />
  <source src="audio.mp3" type="audio/mpeg" />
</audio>
```

All the attributes we have learned so far are also supported in the `video` element. Here's an example of using a `video` element with the `loop`, `controls`, and `muted` attributes:

```
<video

src="https://archive.org/download/BigBuckBunny_124/Content/big_buck_bunny_720p_surround.mp4"
  loop
  controls
  muted
></video>
```

- **`poster` Attribute**: If you wanted to display an image while the video is downloading, you can use the `poster` attribute. This attribute is not available for `audio` elements and is unique to the `video` element.

```
<video

src="https://archive.org/download/BigBuckBunny_124/Content/big_buck_bunny_720p_surround.mp4"
  loop
  controls
  muted
  poster="https://peach.blender.org/wp-content/uploads/title_anouncement.jpg?x11217"
  width="620"
```

```
></video>
```

Target attribute types

- `target` **Attribute**: This attribute tells the browser where to open the URL for the anchor element. There are four important possible values for this attribute: `_self`, `_blank`, `_parent` and `_top`. There is a fifth value, called `_unfencedTop`, which is currently used for the experimental `FencedFrame` API. You probably won't have a reason to use this one yet.
- `_self` **Value**: This is the default value for the `target` attribute. This opens the link in the current browsing context. In most cases, this will be the current tab or window.

```
<a href="https://freecodecamp.org" target="_self">Visit freeCodeCamp</a>
```

- `_blank` **Value**: This opens the link in a new browsing context. Typically, this will open in a new tab. But some users might configure their browsers to open a new window instead.

```
<a href="https://freecodecamp.org" target="_blank">Visit freeCodeCamp</a>
```

- `_parent` **Value**: This opens the link in the parent of the current context. For example, if your website has an iframe, a `_parent` value in that iframe would open in your website's tab/window, not in the embedded frame.

```
<a href="https://freecodecamp.org" target="_parent">Visit freeCodeCamp</a>
```

- `_top` **Value**: This opens the link in the top-most browsing context - think "the parent of the parent". This is similar to `_parent`, but the link will always open in the full browser tab/window, even for nested embedded frames.

```
<a href="https://freecodecamp.org" target="_top">Visit freeCodeCamp</a>
```

Absolute vs. Relative Paths

- **Path definition**: A path is a string that specifies the location of a file or directory in a file system. In web development, paths let developers link to resources like images, stylesheets, scripts, and other web pages.
- **Path Syntax**: There are three key syntaxes to know. First is the slash, which can be a backslash (`\`) or forward slash (`/`) depending on your operating system. The second is the single dot (`.`). And finally, we have the double dot (`..`). The slash is known as the "path separator". It is used to indicate a break in the text between folder or file names. A single dot points to the current directory, and two dots point to the parent directory.

```
public/index.html
./favicon.ico
../src/index.css
```

- **Absolute Path**: An absolute path is a complete link to a resource. It starts from the root directory, includes every other directory, and finally the filename and extension. The "root directory" refers to the top-level directory or folder in a hierarchy. An absolute path also includes the protocol - which could be `http`, `https`, and `file` and the domain name if the resource is on the web. Here's an example of an absolute path that links to the freeCodeCamp logo:

```
<a href="https://design-style-
guide.freecodecamp.org/img/fcc_secondary_small.svg">
  View fCC Logo
</a>
```

- **Relative Path**: A relative path specifies the location of a file relative to the directory of the current file. It does not include the protocol or the domain name, making it shorter and more flexible for internal links within the same website. Here's an example of linking to the `about.html` page from the `contact.html` page, both of which are in the same folder:

```
<p>
  Read more on the
  <a href="about.html">About Page</a>
</p>
```

Link states

- `:link`: This is the default state. This state represents a link which the user has not visited, clicked, or interacted with yet. You can think of this state as providing the base styles for all links on your page. The other states build on top of it.
- `:visited`: This applies when a user has already visited the page being linked to. By default, this turns the link purple - but you can leverage CSS to provide a different visual indication to the user.
- `:hover`: This state applies when a user is hovering their cursor over a link. This state is helpful for providing extra attention to a link, to ensure a user actually intends to click it.
- `:focus`: This state applies when we focus on a link.
- `:active`: This state applies to links that are being activated by the user. This typically means clicking on the link with the primary mouse button by left clicking, in most cases.