Personal Notes Week 3

- HTML tables are not inherently bad; they are essential for presenting tabular data effectively.

- Misusing HTML tables, such as for layout purposes, can lead to accessibility, reusability, and findability issues.

- Tables are ideal for organizing data into rows and columns, facilitating comparison and understanding of relationships within the data.

- Semantic HTML should be embraced, using appropriate elements like the table element to accurately represent content.

- CSS can be utilized to style and arrange tables for different design requirements and screen sizes.

- The key is to use HTML tables when the information is inherently tabular, ensuring both semantic integrity and effective presentation of data.

- HTML tables are constructed using the `<table>`, `<tr>`, `<th>`, and `<td>` elements.

- The `<table>` element marks the beginning and end of the table.

- Each row of the table is defined by the `<tr>` element.

- Within each row, cells are created using the `<td>` element.

- The `<th>` element is used to define header cells, typically in the first row, indicating column headings.

- HTML table cells can contain various types of content, including text, images, videos, and other HTML elements.

- CSS can be applied to style the table's appearance to match design preferences.

- It's important to use `<th>` for header cells and `<td>` for data cells to maintain semantic clarity.

- More complex table structures, such as spanning rows or columns, table headers, bodies, footers, and captions, can be implemented for advanced table layouts.

- While CSS can control the visual presentation of the table, HTML ensures the semantic structure, facilitating accessibility and future styling adjustments.

**CSS:**

CSS, or Cascading Style Sheets, is a style sheet language used to define the presentation and layout of HTML documents. It works in tandem with HTML to enhance the visual appearance of web pages.

- Purpose: CSS adds visual appeal and enhances the presentation of web content by controlling the appearance of HTML elements.

- Connection with HTML: CSS is linked to HTML documents to style and format the content according to design preferences.

- Basic Structure: CSS consists of selectors and declaration blocks.

- The selector specifies the HTML elements to which styles will be applied.

- The declaration block contains the styles (properties and values) that define how the selected elements should be presented.

- Application: When a selector matches HTML elements, the styles defined within its declaration block are applied to those elements.

- Cascading: CSS follows a cascading model, meaning multiple styles can be applied to the same element based on specificity and order of application.

- Simplicity: While CSS can be complex with cascading and specificity, this course will focus on basic CSS principles without delving deep into cascading.

HTML Structure Assessment:

- Begin by examining the HTML structure you want to style.

- Identify elements like paragraphs, headings, and others that you intend to modify.

Element Selection in CSS:

- Use CSS element selectors to pinpoint specific HTML elements for styling.

- Simply write the HTML element's name without angle brackets as the selector (e.g., `p`, `h1`, `h2`).

Style Application:

- Apply styles to the selected elements using CSS properties.

- Properties like `color`, `font-size`, and `background-color` can be used to alter appearance.

Utilizing CSS Comments:

- Employ CSS comments (`/\* \*/`) for notes or explanations within your stylesheet.

- Comments aid in understanding and documenting your CSS code.

Experimentation and Exploration:

- Experiment with different selectors and properties to achieve desired visual effects.

- Play around with various elements and styles to grasp how CSS impacts HTML presentation.

- Class Attributes in HTML:

- Classes are attributes added to HTML elements to provide additional details for styling.

- They create reference points for targeted styling of specific elements.

- Creating Class Selectors:

- To style elements with a specific class, prefix the class name with a dot (`.`) in CSS.

- For example, to style a paragraph with the class "intro" to be green, write `.intro { color: green; }`.

- Targeting Specific Parts of Elements:

- Use the `<span>` element with a class attribute to target specific parts of elements for styling.

- Wrap the desired text within a `<span>` element and assign a class to it.

- Style the class in CSS to achieve the desired appearance.

- Example: Styling Specific Text within a Paragraph:

- To make a specific sentence within a paragraph black, bold, and all uppercase:

- Add a `<span>` element with class "important" around the sentence in the HTML.

- In CSS, write `.important { color: black; font-weight: bold; text-transform: uppercase; }`.

- Order of Style Declarations:

- The order of style declarations in CSS does not affect their application.

- Styles can be applied to entire elements or specific parts within elements, based on class assignments in the HTML.

- \*\*Grouping Selectors in CSS\*\*:

- Grouping selectors allow you to apply the same style to multiple selectors without repeating code.

- Selectors are separated by commas within a single CSS rule.

- For example, to make paragraphs and list items green, write `p, li { color: green; }`.

- Applying Styles to Multiple Elements:

- By grouping selectors, you can apply styles to different types of elements simultaneously.

- This approach reduces redundancy and keeps your CSS concise and manageable.

- Using Classes with Grouped Selectors:

- You can also include classes in grouped selectors to style specific elements with the same style.

- For example, to make text with the class "mineral" green, write `p, li, .mineral { color: green; }`.

- HTML Implementation:

- Apply classes to specific elements in the HTML to target them for styling.

- For example, `<p class="mineral">Mineral water is green.</p>` will make the text "mineral water" green.

- Flexibility and Efficiency:

- Grouping selectors in CSS provides flexibility and efficiency in styling multiple elements with the same styles.

- Whether it's paragraphs, list items, or elements with specific classes, the styles are applied uniformly.

- Practice and Experimentation:

- Practice using grouped selectors to style various elements in your HTML document.

- Experiment with different combinations of selectors to achieve desired styling effects efficiently.

- Understanding Descendant Selectors:

- HTML elements are organized in a hierarchical structure, similar to a family tree, with parent-child relationships.

- Descendant selectors in CSS allow you to target elements that are descendants of another element.

- This relationship can be direct or indirect, traversing through multiple levels of nesting.

- Application of Descendant Selectors:

- To apply styles to specific descendants, use the parent element followed by a space and then the descendant element.

- For example, to style list items within an ordered list, write `ol li { color: blue; font-weight: bold; }`.

- This selector targets list items (`<li>`) that are descendants of ordered lists (`<ol>`).

- Reading Descendant Selectors:

- Despite being written from left to right, descendant selectors are read from right to left.

- For instance, "ol li" means "any list item descended from an ordered list."

- Exercise: Styling Unordered List Items:

- Write a CSS style that selects list items in unordered lists and makes them purple and uppercase.

- For example, use the selector `ul li { color: purple; text-transform: uppercase; }` to achieve this effect.

- Verification and Feedback:

- After writing your CSS style, check the final state of the code pen in the exercise files to confirm if your answer is correct.

- Reviewing the provided solution can help reinforce your understanding of descendant selectors and their usage.

- Named Colors vs. Hexadecimal Codes:

- Named colors are specific colors with assigned names in CSS, but there are only around 130 of them.

- Hexadecimal codes, or hex values, represent colors using six-digit combinations of numbers (0-9) and letters (A-F), where each pair represents the intensity of red, green, and blue (RGB) channels.

- Hexadecimal Representation:

- Hex values are commonly used to specify colors online, providing a wider range of color options.

- Each pair of digits corresponds to a value ranging from 0 to 255 for red, green, and blue channels, respectively.

- For example, the hex value "6495ED" represents the named color "Cornflower Blue."

- Shortened Hexadecimal Codes:

- Hex values can be shortened to three digits if each pair of digits is identical, offering a more concise representation.

- For instance, "7778899" can be shortened to "789" while maintaining the same color.

- Other Color Formats:

- RGB format expresses colors using Base 10 numbers for red, green, and blue channels.

- RGBA includes an additional value for alpha, determining the opacity or transparency of the color.

- HSL and HSLA are alternative color formats occasionally used, especially in platforms like Squarespace.

- Converting Color Formats:

- Online tools like Google and DuckDuckGo provide color pickers that offer various conversions between different color formats.

- Websites like Color Hex offer exploration of shades, tints, and color palettes based on chosen colors, facilitating color selection and manipulation.

-Resource Options:

- Color pickers provided by search engines or dedicated websites offer convenient tools for color conversion and exploration.

- Depending on specific needs, resources like Color Hex can assist in selecting and working with colors effectively.

- Common Web Image Formats:

- GIF: Limited colors, supports transparency and animation.

- PNG: More colors and transparency, but no animation.

- JPEG: Optimized for photographs, supports millions of colors, lacks transparency and animation.

- WebP: Newer format offering high compression for smaller file sizes, suitable for various image types, expected to grow in popularity due to efficiency.

- Choosing the Right Image Format:

- Selecting the appropriate format is crucial to avoid color loss and ensure optimal file size.

- GIFs and PNGs are suitable for illustrations like logos or cartoons, while JPEGs are ideal for photographs.

- WebP offers versatility and efficiency, contributing to faster website loading times.

- Optimizing Image Sizes:

- Reduce image dimensions and file sizes to enhance website loading speed.

- Adjust dimensions, trim or crop unnecessary parts, and resize images as needed.

- Tools like tinypng.com or Adobe Photoshop help in reducing file sizes without compromising quality.

- Including Images in Webpages:

- HTML: Utilize the `<img>` element to place images alongside text, crucial for conveying the page's message (e.g., logos, social media icons).

- CSS: Incorporate background images, primarily decorative and non-essential to the text, allowing for configuration such as repetition or displaying only a portion of the image.

- HTML Implementation:

- Use the `<img>` element with the `src` attribute to specify the image source.

- Example: `<img src="image.jpg" alt="Description">` for embedding images directly into HTML.

- CSS Implementation:

- Apply background images using CSS properties like `background-image`, allowing for customization such as repetition and positioning.

- Example: `background-image: url('image.jpg');` for setting a background image in CSS.

- Considerations:

- Ensure images contribute to the overall user experience by conveying important information or enhancing visual appeal.

- Optimize images for web delivery to improve loading times and website performance.

- Background Image Application:

- Begin by previewing the available background images mentioned in the CSS comments to ensure suitability for the webpage design.

- Blue Dots Image:

- Apply the "blue dots" image as the background for the body element in HTML.

- Note how the image tiles both horizontally and vertically, creating a wallpaper effect.

- Growing Plants Image:

- Set the "growing plants" image as the background for the body element.

- Adjust the background properties using shorthand notation for more flexibility.

- Utilize "repeat-X" or "repeat-Y" to control the direction of image repetition.

- Specify values like "center," "right," or "left" to control where the background image starts.

- Add a height value, such as "97vh," to position the image where desired, such as at the bottom of the page.

- Enhancing Visual Appeal:

- Experiment with different background images and properties to improve the webpage's visual appeal.

- Adjust percentages to fine-tune the positioning of background images, depending on the desired effect.

- Optimizing Background Images:

- Make adjustments to background image properties, such as changing the "bottom" value to around 50%, to enhance the appearance and reduce empty space.

- Exploring Styling Options:

- There are various styling options available for background images, providing inspiration for webpage design.

- Experiment with different values and properties to achieve desired visual effects and enhance the overall webpage design.