Week 4 Personal Notes

Understanding Fonts and Coding:

- After learning about fonts, we're now exploring how to use them in coding to make websites look better.

Making Changes for Better Appearance:

- In the previous coding challenge, the website looked okay but not great. So, we're making some tweaks to improve its appearance.

Choosing Fonts:

- We're going to change the font used on the website. But how do we decide which font to use?

- We'll start with a list of fonts. The browser will try each font in the list until it finds one that's available on the user's computer.

Comparing Arial and Helvetica:

- We'll use Arial, Helvetica, and then a generic Sans-serif font as our font options.

- By comparing the plus sign with both Arial and Helvetica, we can see the subtle differences between them.

Considerations for Different Devices:

- We need to remember that fonts can look different on different devices.

- While we aim for a good appearance, it's impossible to control exactly how fonts will look on every device.

Improving Readability:

- We're aiming for a balance between making the website look good and making sure it's easy to read.

- Headings will use a different font from regular text to create contrast and improve readability.

Experimenting with Font Properties:

- We can also try italicizing text or adjusting the font weight to give the headings a unique appearance.

Overall Improvement:

- By making these changes, we're already seeing a big improvement in the website's appearance and readability.

These details delve into the process of choosing and using fonts in coding to enhance the appearance and readability of websites.

Font Sizing in Web Design:

- There are two types of font sizing: absolute and relative.

- Absolute sizes (like points or pixels) stay the same no matter the screen size.

- Relative units (like percentages or "rem") adjust based on the page size, offering better flexibility.

- Web designers often prefer relative units for their flexibility.

Understanding "rem" Units:

- "rem" is a popular unit for font sizes in web development.

- 1 "rem" is equal to 16 pixels, making it easy to convert between the two.

- Online calculators can help convert font sizes between different units.

Point to REM Converter Tool:

- Tools like the Point to REM Converter help web designers convert between different size units.

- This tool is handy for converting sizes, especially if designers provide measurements in points.

Applying Sizes in CSS:

- In CSS, you can set font sizes using the "font-size" property.

- Experimenting with different sizes helps you find the right fit for your text.

- Font sizes can be adjusted for specific elements like headings (e.g., h1, h3) or the entire page body.

Cascading and Inheritance in CSS:

- Changes to the body font size affect all text except elements with explicitly defined sizes.

- Understanding cascading and inheritance in CSS is important for managing font sizes across a webpage.

Flexibility with Heading Sizes:

- While there's a default size order for headings (h1, h2, h3, etc.), you have the flexibility to adjust them as needed.

- CSS allows you to modify heading sizes without altering the HTML structure, ensuring accessibility and document outlining.

Identifying Spacing Issues:

- Our webpage looks decent but lacks proper spacing between text and color bars.

- For instance, words like "play" touch the edge of color boxes, while there's too much gap between some sections.

Introducing the Box Model:

- In web development, HTML elements are like boxes with various properties.

- These properties include content (text), border, padding (space between content and border), and margin (space between elements).

Understanding Box Model Properties:

- Content: Text inside the box.

- Border: A line surrounding the content, with various styles available.

- Padding: Space between the border and content, which pushes the box's edge away from the content.

- Margin: Space between elements on the page, used to separate them.

Applying Dimensions with CSS:

- CSS properties like "border," "margin," and "padding" can be applied to all four sides of the box or individually.

- Shorthand values provide a concise way to set these properties.

Adjusting Box Model Properties:

- Using CSS properties like "border-left" or "margin-right" allows for specific adjustments to sides of the box.

- Fractional REMs can be used for padding, and similar shorthand notations exist for margin and padding properties.

Focus on Applying Changes:

- We'll focus on making changes to all four sides or specific sides of the box, known as left, right, top, and bottom.

- By understanding these concepts, we can address spacing issues on our webpage using borders, margins, and padding effectively.

Identifying Issues:

- The webpage has layout issues, including misplaced background leaves and spacing problems.

Addressing Body Element:

- Start by removing the default margin from the body element by setting it to zero.

- Move the background image from the body to a newly added footer element at the bottom of the page.

- Give the footer a height and add a top border to separate it from the rest of the page.

Adjusting Headings:

- Remove margins from the H1 heading to push it to the top of the page.

- Add padding to create space between the H1 heading and the edge of the page.

- Remove the background color from H2 and H3 headings and introduce it separately.

- Remove top margin from H2 and adjust padding for a visually appealing appearance.

- Add decorative borders to the bottom of H2 and top of H3 headings.

Centering Images:

- To center the image in the middle of the page, wrap it inside a paragraph with a class of "circle."

- Set the text-align property of the paragraph to center.

Applying Border Radius:

- Apply a class of "circle-img" to the image to round its corners using the border-radius property.

- Border-radius values can be specified in pixels or percentages for desired shapes.

Achieving Better Design:

- By making these CSS modifications, the webpage will look more visually appealing and well-designed.

- With careful attention to spacing, fonts, and CSS properties, a well-designed page can be achieved with minimal CSS.

Understanding Default Link Styles

- Links on web pages are typically blue and underlined. After being visited, they often turn purple.

Targeting Links with CSS:

- To change link colors, we can use CSS and target the "a" anchor tag, which represents links.

- By setting the color property for "a," we can change the color of all links on the page.

Handling Different Link States:

- Links have different states, such as normal, hover, visited, etc.

- We can style each state differently using CSS selectors like "a:hover" for hover effects and "a:visited" for visited links.

Order of CSS Styles:

- The order of CSS styles is important. It's recommended to first define the general link styles, then hover styles, and finally visited link styles.

Considering Underlines:

- It's generally good practice to keep underlines on links within long text for better accessibility.

- However, in navigation bars or areas where users expect clickable links, underlines can be removed for a cleaner appearance.

Conclusion:

- These simple tips help in styling webpage links in an appealing and user-friendly way, considering both aesthetics and accessibility.

In CSS, inheritance and specificity are fundamental concepts that influence how styles are applied to HTML elements.

Styling the Body Element:

- When we style the "body" element in CSS, such as setting the font family to Arial, Helvetica, sans-serif, those styles apply not only to the body itself but also to all elements within it.

- This means that all text inside the body will inherit the specified font family, ensuring consistency in appearance throughout the webpage.

Understanding Font-related Styles:

- Font-related styles, like font family, size, and color, typically inherit down the HTML hierarchy.

- This inheritance ensures that all text elements inside the body are styled consistently, avoiding a mix of different fonts and maintaining visual coherence.

Difference with Box Model Styles:

- However, styles related to the box model, such as borders and padding, usually apply only to the specified element and do not inherit.

- For example, if a border is applied directly to the body element, it won't automatically apply to all elements inside it, preventing chaos in styling.

Illustrating with Examples:

- If we use a wildcard selector "\*", which selects all elements on the page, and apply a border to it instead of the body, every HTML element will be selected and given a border.

- This scenario demonstrates what would happen if borders were inherited, resulting in a messy appearance where every element has a border.

Importance of Controlling Inheritance:

- While inheritance simplifies styling by propagating styles down the HTML hierarchy, it's essential to control it to avoid unintended effects.

- Developers must be aware of how inheritance works to ensure that styles are applied as intended and to maintain a clean and organized stylesheet.

In summary, inheritance in CSS means that certain styles applied to parent elements are inherited by their child elements, ensuring consistency in appearance across the webpage. However, not all styles inherit, especially those related to the box model, to prevent chaos in styling. Understanding and controlling inheritance is crucial for maintaining a well-designed and coherent webpage.

Debugging CSS can sometimes be tricky, especially when you're unsure which element to style or why your styles aren't working as expected. One helpful trick in such situations is to use borders and background colors to reveal hidden information about the layout and styling of your elements.

Adding Borders for Visual Clarity:

- If you want to understand the positioning of an element, such as an unordered list (UL), you can add a border to it using CSS.

- For example, applying a style like "border: 2px solid red;" to the UL element will display a red border around it, revealing its full width within the body element.

Distinguishing Block and Inline Elements:

- By comparing the UL to its child elements, like links (A) within list items (LI), you can observe the difference between block and inline elements.

- Block elements, such as UL and LI, span the full width of their container, while inline elements like links (A) are only as wide as their content.

Solving Styling Challenges:

- When faced with challenges like creating hover effects that span the entire width of an element, you can experiment with changing the display property of the elements.

- For instance, changing links (A) to block-level elements using "display: block;" in CSS stretches them across the page, allowing for consistent hover effects.

Iterative Problem-solving with Borders and Colors:

- Debugging CSS by experimenting with borders and background colors is an iterative process.

- You can gradually solve styling issues by visually inspecting elements with borders and colors, making adjustments, and testing until the desired functionality is achieved.

Focus on Functionality First:

- It's essential to prioritize functionality over aesthetics during the debugging process.

- While your layout may look unattractive with borders and background colors initially, focus on getting it to work correctly before refining the design.

In summary, using borders and background colors to debug CSS is a valuable approach for identifying styling issues and understanding the layout of your web page. By visually inspecting elements and experimenting with styles, you can efficiently troubleshoot problems and achieve the desired functionality before refining the design.