[CSS SELECTORS & STYLING]

Niraj _Vadhiya

[WD - CSS AND CSS 3]

[CSS Selectors & Styling]

Question 1: What is a CSS selector? Provide examples of element, class, and ID selectors.

Ans: - A CSS selector is a pattern used to select the elements you want to style in your web document. It allows you to apply styles to specific HTML elements based on their type, class, ID, attributes, and more.

Examples of CSS Selectors:

1. **Element Selector:**

Targets all instances of a specific HTML element.

Example:

```
p {
  color: blue;
}
```

This will apply the color blue to all <p> (paragraph) elements.

2. Class Selector:

Targets elements with a specific class attribute.

Denoted by a dot (.) before the class name.

Example:

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

This will apply a yellow background to all elements with the class highlight.

3. ID Selector:

Targets a single element with a specific ID attribute.

Denoted by a hash (#) before the ID name.

```
#header {
  font-size: 24px;
```

}

This will apply a font size of 24px to the element with the ID header.

Question 2: Explain the concept of CSS specificity. How do conflicts between multiple styles get resolved?

Ans: - CSS specificity is a set of rules that browsers use to determine which styles to apply when multiple styles conflict. Specificity is calculated based on the types of selectors used in the rule.

1. **Specificity Hierarchy:**

Inline Styles: Styles applied directly to an element using the style attribute (e.g.,). These have the highest specificity.

ID Selectors: Have higher specificity than class selectors and element selectors.

[CSS Selectors & Styling]

Class Selectors, Attribute Selectors, and Pseudo-classes: Have higher specificity than element selectors.

Element Selectors and Pseudo-elements: Have the lowest specificity.

2. Specificity Calculation:

Inline styles: 1000

ID selectors: 0100

Class selectors, attribute selectors, and pseudo-classes: 0010

Element selectors and pseudo-elements: 0001

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    p { color: blue; } /* Specificity: 0001 */
    .highlight { color: green; } /* Specificity: 0010 */
```

```
    #unique { color: red; } /* Specificity: 0100 */
  </style>
</head>
<body>
  <p id="unique" class="highlight">This is a paragraph.</p>
</body>
</html>
```

In this example, the paragraph will be red because the ID selector #unique has the highest specificity.

3. Resolving Conflicts:

When multiple rules apply to the same element, the rule with the highest specificity takes precedence.

If two rules have the same specificity, the one that appears last in the CSS (or the last one loaded) will be applied.

[CSS Selectors & Styling]

Question 3: What is the difference between internal, external, and inline CSS? Discuss the advantages and disadvantages of each approach.

Ans: -

1. **Internal CSS**

Defined within the <style> tag inside the <head> section of an HTML document.

Example:

```
<head>
  <style>
    p {
      color: blue;
    }
  </style>
</head>
```

Advantages:

Useful for single-page websites or when styles are specific to a single page.

Easier to manage and debug for small projects.

Disadvantages:

Not reusable across multiple pages.

Can lead to larger HTML files and slower load times.

2. External CSS:

Defined in a separate .css file and linked to the HTML document using the <link> tag.

Example:

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

Advantages:

Styles can be reused across multiple pages, promoting consistency.

Keeps HTML files cleaner and smaller.

Easier to maintain and update styles in one place.

Disadvantages:

Requires an additional HTTP request to load the CSS file, which can affect load times.

Changes to the CSS file affect all linked pages, which can be problematic if not managed carefully.

3. Inline CSS:

Defined directly within the style attribute of an HTML element.

Example:

This is a paragraph.

[CSS Selectors & Styling]

Advantages:

Useful for quick, specific style changes.

Does not require additional files or HTTP requests.

Disadvantages:

Not reusable and can lead to code duplication.

Makes HTML code harder to read and maintain.

Inline styles have the highest specificity, which can make overriding styles difficult.