Autumn-23 Assignment-01

Course Code: CSE-3532

Course Title: Tools and Technologies for Internet Programming

Submitted by:

Name: Minhaj Ul Alam Bhuiyan Sohan

ID- C213032

Section: 5AM

Date of Submission: 07.08.23

Submitted to:

Mohammad Shabaj Khan

Task-9

ANS-(A)

Block-level elements: Block-level elements commence on a new line and occupy the entire available width within their parent container. They form a "block" that pushes other elements horizontally. Block-level elements can enclose other block-level and inline elements. Examples of block-level elements:

- <section>: Represents a thematic grouping of content.
- <article>: Represents a self-contained piece of content.
- <header>: Represents a group of introductory content at the top of a section or page.
- <nav>: Represents navigation links.
- <main>: Represents the main content of the document.
- <footer>: Represents the footer section of a document or a section.

Inline elements: Inline elements do not initiate a new line and only take up the necessary width to fit their content. They flow within the text and do not disrupt the surrounding content. Inline elements cannot contain block-level elements but can include other inline elements. Examples of Inline elements:

- <span>: A generic inline container used for styling and scripting purposes.
- <a>: Represents a hyperlink.
- <strong>: Used for indicating strong importance.
- <em>: Used for emphasizing text.
- <img>: Displays an image.
- <input>: Used for input fields in forms.

- <button>: Represents a clickable button. In summary, block-level elements create blocks that occupy the entire width, while inline elements flow within the text and take up only the necessary width. Block-level elements can enclose both block-level and inline elements, but inline elements cannot contain block-level elements.

ANS-(b)

Semantic tags in HTML are elements designed to carry specific meaning and convey the structural organization of the enclosed content. They provide valuable context and define the purpose of the content, facilitating comprehension for both automated systems (such as search engines) and developers in understanding the layout of the document. Semantic tags significantly enhance the accessibility, search engine optimization (SEO), and maintainability of the web page by organizing the content in a meaningful and logical manner.

Examples of Semantic Tags:

- <header>: Represents the introductory section, often containing logos, headings, and navigation menus.

- <nav>: Defines a section that contains navigation links, guiding users to various parts of the website.

- <main>: Represents the primary content of the web page, excluding headers, footers, and sidebars.

- <article>: Represents an independent, complete, and self-contained piece of content, such as a blog post or news article.

- <section>: Represents a thematic grouping of content within the document. Examples of Non-Semantic Tags:

- <div>: A generic container without any inherent meaning or semantic value, often used for layout purposes.

- <span>: A generic inline container utilized for applying styles or scripting purposes, lacking any specific meaning.

- <b>: Represents bold text styling but does not convey the reason for the text's importance or significance.

- <i>: Represents italicized text but does not provide context for why the text is in italics.

- <font>: Used to apply font styles, sizes, and colors but lacks semantic meaning and is not recommended for modern web development.

In essence, semantic tags enrich the structure and semantics of the HTML document, making it more accessible, SEO-friendly, and easier to maintain. They convey the purpose and relationship of content, fostering better understanding for both humans and machines.

ANS-(c)

Ordered list: A numbered list is employed to present a series of items in a specific sequence, where each item is preceded by a number or letter (typically starting from 1). The list items' order holds significance and can be indicated using numerical or alphabetical ordering.

Ordered Lists (<ol>): - Utilized to generate a list with numbers or letters as markers. - Each list item is prefixed with a number (default is sequential numbering). - The order of items in the list matters, and they follow a specific sequence.

Unordered list: An unordered list is used to represent a collection of items with no defined sequence or order. Each item in the list is preceded by a bullet point (customizable using CSS) or other marker.

Unordered Lists (<ul>): - Employed to create a list with bullet points or other custom markers. - Each list item is preceded by a bullet point (default) or any customized marker. - The order of items in the list does not carry significance; they form a simple collection of points.

ANS-(d)

Inline:

The inline styles will exclusively impact the HTML element to which the style attribute with CSS-property values is implemented. The initial paragraph in the example below will be tailored with a red color and a font size of 20px. The properties exclusively affect the initial line of the code, not the entire code.

Example:

```
<body>

        <p style=”color:blue; font-size: 10px;”>This is our first HTML
code.</p>

        <p>This is our second HTML code</p>

</body>
```

Internal:

Embedded CSS is one of the most favored CSS approaches for modifying, customizing, and adjusting the distinctive styles of an individual web page. You can employ embedded CSS by incorporating the <style> element within the <head> section of an HTML web page. Embedded CSS is suitable for styling a single web page, although it does not apply to multiple web pages. However, you can utilize the same code to style multiple web pages.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Internal Stylesheet Example</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
    }
    h1 {
      color: #007bff;
    }
  </style>
</head>
<body>
  <!-- Your HTML content goes here -->
</body>
</html>
```

External CSS:

In this method, you link an external CSS file to your HTML document using the <link> element within the <head> section of the HTML file.

Example:

```
<!DOCTYPE html>

<html>

<head>

  <title>Inline Styles Example</title>

</head>

<body>

  <h1 style="color: #eee;">This is a heading with inline style.</h1>

  <p style="font-size: 10px; color: #fff;">This is a paragraph with inline
style.</p>

</body>

</html>
```

ANS-(e)

Content Width: The width of the content is explicitly set to 300px.
Border Width: The overall border width is 30px (15px on the left + 15px on the right) since the border is 15px wide on each side.

Padding Width: The combined padding width is 100px (50px on the left + 50px on the right) as the padding is 50px wide on each side.

Margin Width: The total margin width is 40px (20px on the left + 20px on the right) since the margin is 20px wide on each side.

To determine the total width of the <div> element:
Total width = Content width + Total Padding width + Total Border width + Total Margin width

Total width = 300px + 200px + 30px + 40px = 470px

Hence, the <div> element will have a total width of 670px.

ANS-(F)

A pseudo-class is a selector that chooses elements in a particular condition, such as being the initial element of their kind or being hovered over by the mouse pointer. These selectors function as if you had applied a class to some section of your document, often reducing the need for excessive classes in your markup and allowing you to have more adaptable, maintainable code.
State-based Styling: Pseudo-classes permit you to style elements based on their current condition or user interaction. For instance, you can alter the style of a link when it is hovered over or clicked, modify the

appearance of a form element when it is in focus, or apply styles to an element when it is being selected.

Position-based Styling: Pseudo-classes also enable you to target elements based on their position within the document tree. For example, you can style the first child or last child of a parent element differently, select even or odd elements, or target specific elements based on their relationship with other elements.

Interactive Effects: Pseudo-classes are vital for generating interactive and dynamic effects on web pages. They facilitate the creation of animations, transitions, and other effects based on user interactions or element states.

ANS-(g)

The CSS rule `margin: 15px 70px;` is responsible for adding space around an element, creating a gap between it and nearby elements.

In this specific rule, the `margin` property is used with two values: 15px and 70px. These values indicate the top/bottom margin and left/right margin, respectively. The order of the values is essential and follows the shorthand notation for specifying margins.

The significance of the values can be further explained as follows:

- 15px: This value represents the vertical margins, adding a margin of 15 pixels above and below the element.

- 70px: This value signifies the horizontal margins, providing a margin of 70 pixels on the left and right sides of the element.

As a result, the CSS rule `margin: 15px 70px;` will establish a margin of 15 pixels above and below the element, as well as a margin of 70 pixels on the left and right sides of the element.

This technique is frequently utilized to create spacing between an element and its surrounding content or other elements on the page. The specific values can be adjusted as needed to achieve the desired layout and visual appearance.

ANS-(h)

The CSS descendant selector is utilized to target elements that are descendants of a specific parent element. The term "descendant" signifies elements nested anywhere within the DOM tree, whether they are direct children or located deeper within multiple levels of nesting.

The descendant combinator is denoted by a single space between two selectors. It combines two selectors: the first selector represents the ancestor (parent, grandparent, etc.), and the second selector represents the descendants. Elements matched by the second selector will be selected if they have an ancestor element that matches the first selector. The descendant selector uses the concept of descendant combinators to accomplish this selection.

In summary, the CSS descendant selector is a powerful tool for selecting and styling elements based on their hierarchical relationship with another specific element in the HTML document.