HTML ASSIGNMENT

-T.LAKSHMI PRASANNA

TRN NO: 2K23CSUN01234

1.QUESTION

The CSS Box model is a fundamental concept in web design that describes the layout and design of elements on a webpage as a rectangular box, which consists of several parts: content, padding, border and margin.

The CSS box model affects the layout of elements on a webpage by influencing the spacing and positioning of elements relative to each other. By adjusting the padding, border, and margin properties, you can control the spacing between elements, creative visual separation, and influence the overall structure and design of the webpage.

EXERCISE:

<!DOCTYPE html>

<html>

<head>

    <title>DIWALI IMPORTANCE</title>

    <style>

 .box{

            width: 200px;

            height: 200px;

            background-colour: pink;

            margin: 40px;

            padding: 10px;

            border: 4px solid orange;

            box-sizing: border-box;

        }

    </style>

</head>

<body>

    <div class="box">

        <p>it symbolises the spiritual victory of light over darkness, good over evil, and knowledge over ignorance</p>

    </div>

</body>

</html>

**2.QUESTION**

The concept of CSS specificity refers to a set of rules and principles that determine which CSS rules should be applied to an HTML element when multiple conflicting CSS rules target the same element. It is a crucial aspect of styling web pages because it helps resolve conflicts and ensures that the intended styles are applied consistently. CSS specificity is essential for creating well-organized and maintainable stylesheets.

CSS specificity is determined by the combination of selectors in a CSS rule. It's typically represented as a numerical value, and the higher the specificity, the more "specific" a rule is. Specificity is often expressed as a four-part value with each part contributing to the overall specificity score:

1. Inline styles: Inline styles have the highest specificity. If you apply styles directly to an HTML element using the **style** attribute, those styles will override any conflicting styles from external stylesheets or embedded styles.
2. IDs: Selectors that use ID attributes have a high specificity. For example, **#myElement** will have a higher specificity than class selectors or element selectors.
3. Classes, attributes, and pseudo-classes: Selectors like **.myClass**, **[data-attribute]**, or **:hover** have a moderate level of specificity.
4. Elements and pseudo-elements: Selectors that target HTML elements, like **p**, **a**, or **::before**, have the lowest specificity.

When multiple CSS rules conflict, the browser calculates the specificity of each rule, and the rule with the highest specificity takes precedence. If two or more rules have the same specificity, the one that appears later in the stylesheet or in the HTML source code will override the previous rules. In cases of equal specificity and order, the "cascading" nature of CSS is applied, and the rule declared last will be applied.

EXERCISE:

<!DOCTYPE html>

<html>

<head>

<style>

 h1 {

colour: pink ;

}

#specificity-1 {

colour: orange;

}

p {

colour: grey ;

}

specificity-2 {

colour: purple;

}

#specificity-3 {

colour: crimson;

}

</style>

</head>

<body>

<h1>BEST FRIENDS</h1>

<p id="specificity-1">POOJYA LAKSHMI</p>

<p class="specificity-2"> SUJITHA</p>

<p class="specificity-2" id="specificity-3">SRUTHI</p>

<p>TENTH CLASS BEST FRIENDS</p>

</body>

</html>

3.QUESTION

CSS Flexbox and CSS Grid are two powerful layout models in CSS that allow you to design and structure the layout of web pages. They each serve different purposes and are best suited for specific layout requirements:

**CSS Flexbox (Flexible Box Layout):**

* Flexbox is designed for one-dimensional layout, allowing you to arrange items in a row or column.
* Use Flexbox when you want to distribute space within a container along a single axis (horizontally or vertically). It's particularly useful for creating flexible and dynamic layouts where items can vary in size, and you need to align them or distribute space evenly.

Example:

<!DOCTYPE html>

<html lang="en">

<head>

<style>

.flex-container {

    display: flex;

    justify-content: space-between;

  }

  .flex-item {

    flex: 1;

  }

  </style>

</head>

<body>

    <div class="flex-container">

        <div class="flex-item">Item 1</div>

        <div class="flex-item">Item 2</div>

        <div class="flex-item">Item 3</div>

      </div>

        </body>

</html>

**CSS Grid:**

* CSS Grid is designed for two-dimensional layout, allowing you to create a grid structure with both rows and columns.
* Use Grid when you need precise control over the layout, and you want to define a grid of rows and columns for placing items. It's great for complex layouts like grids of images, page structures, and aligning content in both dimensions

Example:

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

    display: grid;

    grid-template-rows: auto 1fr auto;

    grid-template-columns: 1fr 2fr;

    grid-gap: 10px;

  }

  .grid-item {

    padding: 10px;

    border: 1px solid #000;

  }

</style>

</head>

<body>

    <div class="grid-container">

        <div class="grid-item">Header</div>

        <div class="grid-item">Sidebar</div>

        <div class="grid-item">Main Content</div>

        <div class="grid-item">Footer</div>

      </div>

       </body>

</html>

EXERCISE:

CSS Flexbox (Flexible Box Layout):

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

    display: flex;

    justify-content: space-between;

  }

  .flex-item div {

    width: 200px; height: 150px;

 background-colour: pink; border: 1; }

  </style>

</head>

<body>

    <div class="flex-container">

    <div class="flex-item">Alphabets</div>

    <div class="flex-item">Vowels</div>

    <div class="flex-item">Consonants</div>

  </div>

  </body>

</html>

CSS Grid Layout:

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

    display: grid;

    grid-template-rows: auto 1fr auto;

    grid-template-columns: 1fr 2fr;

    grid-gap: 10px;

  }

  .grid-item {

    padding: 10px;

    border: 1px solid #000; background-colour: pink;

 text-align: center; }

  </style>

</head>

<body>

    <div class="grid-container">

    <div class="grid-item">Alphabets</div>

    <div class="grid-item">Vowels</div>

    <div class="grid-item">Consonants</div>

    <div class="grid-item">Articles</div>

  </div>

</

body>

</html>

4.QUESTION:

 In CSS, the position property specifies how an element is positioned on the web page. There are five possible values for this property: static, relative, fixed, absolute, and sticky. Each of these values affects the element’s position in a different way. Here is a brief explanation of each value and some examples of when and how to use them:

* position: relative; means that the element is positioned relative to its normal position. You can use the top, bottom, left, or right properties to adjust the element’s position away from its normal position. The other elements in the document will not be affected by the relative positioning of the element. You can use this value when you want to slightly modify the position of an element, without changing the layout of the document.
* position: absolute; means that the element is positioned relative to the nearest positioned ancestor, which is the closest ancestor element that has a position value other than static. If there is no such ancestor, the element is positioned relative to the initial containing block, which is the root element of the document. The element is removed from the normal flow of the document, and it can overlap other elements. You can use the top, bottom, left, or right properties to specify the position of the element within its containing block. You can use this value when you want to create an element that is positioned independently of the document flow, such as a popup or a modal.
* position: fixed; means that the element is positioned relative to the viewport, which is the visible area of the browser window. The element will stay in the same place even if the page is scrolled. You can use the top, bottom, left, or right properties to specify the position of the element within the viewport. The element will not leave a gap in the document where it would normally be located. You can use this value when you want to create an element that is always visible on the screen, such as a navigation bar or a banner.

EXERCISE:

<html>

<head>

<style>

 .container {

 width: 400px;

 height: 300px;

 background-colour: light grey

 position : relative;

 }

 .box1 {

 width: 100px;

 height: 100px;

 background-colour:  pink;

 position: relative;

 top: 30px;

 left: 30px;

 }

 .box2 {

 width: 100px;

 height: 100px;

 background-colour: orange;

 position: absolute;

 top: 50px;

 left: 50px

 }

 .box3 {

 width: 100px;

 height: 100px;

 background-colour: light green;

 position: fixed;

 top: 20px;

 right: 20px;

 }

</style>

</head>

<body>

 <div class="container">

 <div class="box1">Box 1 (position: relative)</div>

 <div class="box2">Box 2 (position: absolute)</div>

 <div class="box3">Box 3 (position: fixed)</div>

</div>

</body>

</html>

5.QUESTION

* ::**before Pseudo-element**creates a pseudo-element that is the first child of the selected element. It is often used to add cosmetic content to an element with the content property. It is inline by default.

Example:

<!DOCTYPE html>

<**html**>

<**head**>

    <**title**>before Demo</**title**>

    <**style**>

        body {

            background-colour: pink;

            colour: green;

            font-size: large;

            text-align: center;

        }

        p::before {

            content: '"';

            colour: red;

            font-size: 30px;

        }

    </**style**>

</**head**>

<**body**>

    <**h1**>Geeks For Geeks</**h1**>

<**h2**>::before element</**h2**>

    <**p**>

        This is a paragraph to which

        we added red colour quotation

        marks using ::before element.

    </**p**>

</**body**>

</**html**>

**::after Pseudo-element** creates a pseudo-element that is the last child of the selected element. It is often used to add cosmetic content to an element with the content property. It is inline by default.

Example:

<!DOCTYPE html>

<**html**>

<**head**>

    <**title**>after Demo</**title**>

    <**style**>

        body {

            background-colour: orange;

            colour: green;

            font-size: large;

            text-align: center;

        }

        P::after{

Content:’”’;

            colour: red;

            font-size: 30px;

        }

    </**style**>

</**head**>

<**body**>

    <**h1**>Geeks For Geeks</**h1**>

    <**h2**>::after element</**h2**>

    <**p**>

        This is a paragraph to which

        we added red colour quotation

        marks using ::after element.

    </**p**>

</**body**>

</**html**>

EXERCISE:

<!DOCTYPE html>

<html>

<head>

<style>

 .list {

 list-style-type: none;

 margin: 0;

 padding: 0;

 }

 .list-item {

 position: relative;

 padding: 10px;

 }

 .list-item::before {

 content: '•';

 position: absolute;

 left: -20px;

 colour: orange;

 }

 .list-item::after{

  content: '';

 position: absolute;

 width: 100%;

 height: 1px;

 background-colour: purple;

 bottom: 0;

 left: 0;

 }

</style>

</head>

<body>

<ul class="list">

 <li class="list-item">lays</li>

 <li class="list-item">sweet</li>

 <li class="list-item">chips</li>

 <li class="list-item">biscuits</li>

</ul>

</body>

</html>

6.QUESTION

Responsive Design is a web design approach that aims to adapt web page layout to any screen from maximum screen sizes like TVs, monitors, etc. to minimum screen sizes like mobiles, etc. with good usability. Media Queries is a CSS3 Feature that makes a website page adapt its layout to different screen sizes and media types. Media queries are CSS rules that apply specific styles based on the characteristics of the user’s device or viewport. These characteristics can include screen width, height, orientation, resolution, and more.

Media Queries helps the developers to create a consistent and optimized user experience across all devices. Without responsive design, websites may appear unstable and broken on certain screens which results in poor user experience. With a better user experience, it will be easy to use the product.

Example:

<!DOCTYPE html>

<html >

<head>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

header {

background-colour: pink

padding: 10px;

text-align: center;

}

ul {

list-style-type: none;

margin: 0;

padding: 0;

}

li {

display: inline;

margin-right: 20px;

}

.main-content {

max-width: 800px;

margin: 20px auto;

padding: 20px;

}

@media screen and (max-width: 600px) {

header {

padding: 15px;

}

nav ul {

text-align: center;

margin-top: 10px;

}

li {

display: block;

margin-bottom: 5px;

}

.main-content {

padding: 10px;

}

}

</style>

</head>

<body>

<header>

<h1>My Responsive Website</h1>

<nav>

<ul>

<li><a href="#">Home</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Contact</a></li>

</ul>

</nav>

</header>

<section class="main-content">

<h2>Welcome to my website!</h2>

<p>This is some sample content for the main section of the webpage.</p>

</section>

</body>

</html>

EXERCISE:

<!DOCTYPE html>

<html>

<head>

 <style>

 body {

 font-family: Arial, sans-serif;

 }

 .container {

 width: 50%;

 margin: 0 auto;

 padding: 20px;

 background-colour: pink;

 }

 @media only screen and (max-width: 600px) {

 .container {

 width: 80%;

 }

 }

 @media only screen and (max-width: 400px) {

 .container {

 width: 100%;

 }

 }

 </style>

</head>

<body>

 <div class="container">

 <h1>Welcome to Our Responsive Webpage</h1>

 <p>this is a simple information about the responsive design exercise</p>

 </div>

</body>

</html>

7.QUESTION

Accessibility in web development is crucial as it ensures that websites and web applications are usable by people of all abilities and disabilities. It's about providing equal access to information and functionality, regardless of any physical or cognitive limitations.

ARIA (Accessible Rich Internet Applications) roles and attributes are a set of attributes that can be added to HTML elements to help assistive technologies, such as screen readers, interpret and convey the information to users with disabilities.

Example:

<!DOCTYPE html>

<html>

<head>

<style>

 .container {

 margin: 20px;

 padding: 20px;

 border: 1px solid #ddd;

 }

</style>

</head>

<body>

 <div class="container">

 <h1>Welcome to MR-Campus</h1>

 <form>

 <label for="username">Username:</label>

 <input type="text" id="username" name="username" aria-label="Enter

your username" aria-required="true"><br><br>

 <label for="password">Password:</label>

 <input type="password" id="password" name="password" aria-label="Enter your password" aria-required="true"><br><br>

 <button type="sign in" aria-label="Submit form">Sign in</button>

 </form>

 </div>

</body>

</html>

EXERCISE:

<!DOCTYPE html>

<html>

<head>

<style>

 .container {

 margin: 30px;

 padding: 30px;

 border: 2px solid #ddd;

 }

</style>

</head>

<body>

 <div class="container">

 <h1>Welcome to Zomato</h1>

 <div role="list">

 <div role="list item">Food Item 1</div>

 <div role="list item">Food Item 2</div>

 <div role="list item">Food Item 3</div>

 <div role="list item">Food Item 4</div>

 </div>

 <input type="text" aria-label="Name" aria-required="true"

placeholder=" name and address">

 <button role="button">order</button>

 </div>

</body>

</html>

8.QUESTION

The **DOCTYPE** declaration is an instruction to the web browser about what version of HTML the page is written in. It is usually the first line of code in an HTML document and is used to ensure that the web page is rendered correctly by the browser.

Different browsers have different rendering engines that interpret HTML and CSS differently. The **DOCTYPE** declaration helps the browser to determine which rendering engine to use. If the **DOCTYPE** declaration is missing or incorrect, the browser may render the page in **quirks mode** instead of **standards mode**

EXERCISE:

<!DOCTYPE html>

<html>

<head>

    <title>SUNDAR PICHAI BIOGRAPHY</title>

</head>

<body>

 <h1>SUNDAR PICHAI BIOGRAPHY</h1>

 <P1>Pichai Sundararajan (born June 10, 1972 ), better known as Sundar Pichai is an Indian-born American business executive. He is the chief executive officer (CEO) of Alphabet Inc. and its subsidiary Google. Pichai began his career as a materials engineer.</P1>

</body>

</html>