Assignment CSS

Q 1: what is the benefit of using CSS?

- Ans: 1) Faster Page Speed. More code means slower page speed. ...
- 2) Better User Experience. CSS not only makes web pages easy on the eye, it also allows for user-friendly formatting. ...
- 3) Quicker Development Time. ...
- 4) Easy Formatting Changes. ...
- 5) Compatibility Across Devices

Q 2 What are the disadvantages of CSS?

- Ans. Browser Compatibility. CSS may render differently in various web browsers, leading to inconsistencies in the visual presentation. ...
- Learning Curve. ...
- Lack of Security. ...
- Limited Layout Control. ...
- Performance Impact. ...
- Overriding Styles. ...
- Maintenance Challenges

Q 3: What is the difference between CSS2 and CSS3?

Ans. CSS3 has compatibility with external font styles through Google fonts and typecast. It was not possible with earlier CSS1 and CSS2. The selectors in CSS3 have increased while CSS1 and CSS2 only had simple selectors.CSS1 and CSS2 didn't have provisions to specifically design the web layout.

Q 4: Name a few CSS style components

Ans. CSS (Cascading Style Sheets) is a versatile language used for styling web documents. CSS allows you to control the appearance and layout of HTML elements. Here are a few CSS style components and properties:

1. Color: You can specify the color of text, backgrounds, borders, and other elements using properties like `color`, `background-color`, and `border-color`.

2. Font: CSS allows you to set the font family, size, style, and weight using properties like `font-family`, `font-size`, `font-style`, and `font-weight`.

3. Text: You can control text properties, such as alignment, decoration, spacing, and more, using properties like `text-align`, `text-decoration`, `line-height`, and `letter-spacing`.

4. Box Model: The box model defines how elements are laid out in terms of their content, padding, borders, and margins. You can manipulate these properties with `padding`, `border`, and `margin` properties.

5. Layout and Positioning: You can control the layout and positioning of elements using properties like `display`, `position`, `float`, and `z-index`.

6. Flexbox: CSS flexbox is a layout model that enables flexible and efficient distribution of space among items in a container. It uses properties like `display: flex`, `flex-direction`, and `justify-content`.

7. Grid Layout: CSS grid layout is a two-dimensional layout system that allows you to create complex, grid-based layouts. You can use properties like `display: grid`, `grid-template-columns`, and `grid-template-rows`.

8. Transitions and Animations: CSS allows you to create smooth transitions and animations using properties like `transition` and `animation`. You can animate properties like `color`, `width`, and `transform`.

9. Backgrounds: You can set background properties for elements, including background color, image, and positioning, using properties like `background-color`, `background-image`, and `background-position`.

10. Borders: Customize the borders of elements with properties like `border-width`, `border-style`, and `border-radius`.

11. Text Shadows and Box Shadows: You can add shadows to text and elements using properties like `text-shadow` and `box-shadow`.

12. Pseudo-classes and Pseudo-elements: CSS provides a range of pseudo-classes (e.g., `: hover`, `: active`, `: focus`) and pseudo-

elements (e.g., `:: before`, `:: after`) to apply styles to specific states or generate additional content.

These are just a few examples of CSS-style components. CSS is a powerful and extensive language with many more properties and features for controlling the appearance and behavior of web pages.

Q 5: What do you understand by CSS opacity?

Ans. CSS `opacity` is a property that controls the transparency of an element, affecting how visible it is on a web page. It is used to make elements, such as text, images, or entire containers, partially or fully transparent.

Q 6: How can the background color of an element be changed?

Ans. You can change the background color of an HTML element using CSS

Q7: How can image repetition of the backup be controlled?

Ans. To control the repetition of a background image in CSS, you can use the `background-repeat` property

Q 8: What is the use of the background-position property?

Ans. The `background-position` property in CSS is used to specify the positioning of a background image within an HTML element that has a background image set. It allows you to control where the background image is placed relative to the element's content box.

Q 9: Which property controls the image scroll in the background?

Ans. The background-attachment property sets whether a background image is with the rest of the page or is fixed.

Q 10: Why should background and color be used as separate properties?

Ans. Using `background` and `color` as separate properties in CSS is a matter of separating concerns and maintaining code clarity and flexibility. Here are some reasons why it's generally recommended to use separate properties for setting background and text color:

1. **Clarity and Readability**: Using separate properties makes your code more readable and self-explanatory. When someone else reads your code, or you revisit it in the future, it's clear what each property is responsible for. It also reduces the chance of making unintended changes when modifying the style.
2. **Maintainability**: Separating background and text color allows for easier maintenance and updates. If you want to change the background color or the text color independently, you can do so without affecting the other property. This makes your CSS more modular and less error-prone.
3. **Cascading and Specificity**: Separating background and text color makes it easier to manage the specificity and inheritance of styles. In complex styling scenarios, combining both colors into a single property can lead to unexpected interactions, especially when dealing with cascading styles or specificity conflicts.
4. **Accessibility**: When working with web accessibility, it's important to have clear control over text color to ensure readability and contrast. Separating text color from background color allows you to make informed decisions about color choices that meet accessibility guidelines.

Here's an example illustrating the separation of background and text color:

```
.my-element background-color
color
.my-element background
```

In the "Good practice" example, it's clear which property sets the background color and which one sets the text color. In the "Avoid" example, the 'background' property is ambiguous, making it difficult to determine which part of the style it affects.

Separating background and text color properties enhances code maintainability, readability, and the overall quality of your CSS code.

Q 11 How do center block elements use CSS1?

Ans.TO centrally align the block element, we can simply make use of the <center>tag. All the elements within the <center> tag will be centrally aligned.

Q12: How to maintain the CSS specifications?

Ans. Maintaining CSS specifications and ensuring consistency in your styles across a project or website is essential for a professional and efficient web development process. Here are some best practices to help you maintain your CSS specifications:

1. **Document Your Styles:**
   - Create a style guide or documentation for your project. This guide should include information on naming conventions, typography, colors, spacing, and any custom components or patterns you use in your project.
2. **Use a CSS Preprocessor:**
   - Consider using a CSS preprocessor like Sass or Less. Preprocessors provide features like variables, nesting, and mixins, making it easier to maintain and organize your CSS code.
3. **Modularize Your Styles:**
   - Break down your CSS into smaller, manageable modules or components. Each module should have a specific purpose and encapsulate its style. This helps with maintainability and reusability.
4. **Follow Naming Conventions:**
   - Adopt a naming convention like BEM (Block, Element, Modifier), or any other convention that suits your project. Consistent class naming helps make your styles predictable and maintainable.

5. **Version Control:**
    - Use version control systems like Git to keep track of changes in your CSS code. Commit your changes with meaningful messages to keep a history of your style modifications.
6. **Lint Your CSS:**
    - Use a CSS linter (e.g., Stylelint) to enforce coding standards, catch errors, and maintain consistency in your CSS code.
7. **Responsive Design:**
    - Ensure that your CSS is responsive to different screen sizes and devices. Test and adjust your styles for mobile, tablet, and desktop views.
8. **Browser Compatibility:**
    - Regularly test your styles in multiple browsers to ensure cross-browser compatibility. Consider using CSS vendor prefixes or auto prefixing tools to handle browser-specific CSS properties.
9. **Minimize Redundancy:**
    - Avoid redundant or duplicate CSS rules. Use inheritance and cascading principles effectively to keep your styles concise.
10. **Perform Code Reviews:**
    - Collaborate with other developers and conduct code reviews to maintain code quality and adherence to style guidelines.
11. **Optimize for Performance:**
    - Minimize unnecessary styles and optimize your CSS for faster loading times. This includes using CSS minification and reducing the use of heavy selectors.
12. **Testing and Quality Assurance:**
    - Regularly test your styles for visual consistency and correctness. Automated testing tools can help identify regressions.
13. **Update and Refactor:**
    - Periodically review and refactor your CSS code to eliminate outdated or unnecessary styles. Keep your codebase up to date with best practices.
14. **Documentation Maintenance:**
    - Keep your style guide and documentation up to date as your project evolves. Ensure that it accurately reflects the current state of your styles.
15. **Keep Learning:**
    - Stay updated with the latest CSS features and best practices. CSS is constantly evolving, and learning new techniques can lead to more efficient and maintainable styles.

Q13: What are the ways to integrate CSS as a web page?

Ans. Using CSS. CSS can be added to HTML documents in 3 ways: Inline - by using the style attribute inside HTML elements. Internal - by using a <style> element in the <head> section.

What is embedded style sheets?

Ans. An embedded style sheet is declared within the <head> element of an XHTML document. It applies to the whole document, rather than just one element. Each style declaration (or CSS rule) gets applied to everything in the document that matches that rule.

What are the external style sheets?

Ans. A separate CSS file that can be accessed by creating a link within the head section of the webpage

What are the advantages and disadvantages of using external style sheets?

- Ans. Modularity and Maintainability: By using external style sheets, you separate your CSS code from your HTML code. ...
- Consistency: External style sheets allow you to maintain a consistent look and feel across your entire website.

Q16: What is the meaning of the CSS selector?

Ans. a pattern of elements and other terms that tell the browser which HTML elements should be selected to have the CSS property values inside the rule applied to them

Q17: What are the media types allowed by CSS?

1. Ans. **all**: This is the default media type and applies to all devices and media features.
2. **screen**: Styles specified for the screen media type apply to computer screens, tablets, smartphones, and other similar devices.
3. **print**: Styles specified for the print media type are used when printing a web page. You can control the page layout, margins, and other print-specific styles.
4. **speech**: The speech media type is used for screen readers and other text-to-speech devices. You can define styles to optimize the content for audio rendering.

5. **projection**: Styles for projection media type are used when displaying content on a projector or similar medium. You can control aspects like fonts, layout, and colors.
6. **handheld**: The handheld media type is intended for small, handheld devices like mobile phones. You can define styles to optimize the content for these devices.
7. **TV**: Styles for TV media type apply when displaying web content on a television screen. You can adjust the layout and styling for this medium.

Q18: What is the rule set?

Ans. In CSS (Cascading Style Sheets), a "rule set" is the fundamental building block used to define the styling for HTML elements. A rule set consists of two main components: a selector and a declaration block.

1. **Selector**: The selector specifies the HTML element or elements to which the styles should be applied. It is the part of the rule set that identifies the target of the styling. Selectors can be element names, class names, IDs, or other attribute selectors. For example:
   - Element Selector:

     copy code

   - Class Selector:

     copy code
     ```
     .my-class
     ```
   - ID Selector:

     copy code
     ```
     #my-id
     ```
2. **Declaration Block**: The declaration block is enclosed in curly braces `{ }` and contains one or more CSS property-value pairs. Each property defines a specific aspect of the element's style (e.g., color, font size, margin), and the associated value sets the value for that property. For example:

   copy code
   ```
   color font-size 16px margin 10px
   ```

In the example above, the rule set applies styles to all `<p>` elements in the HTML document. It sets the text color to blue, the font size to 16 pixels, and a margin of 10 pixels for each matching element.

Multiple rule sets can be defined in a CSS file to control the styling of various elements on a web page. The browser applies these styles according to the specificity of the selectors and the cascade (the process of resolving conflicting styles). The "cascading" part of CSS refers to the rules for determining which styles should be applied when there are conflicting styles from different rule sets.

Here's a simple example with multiple rule sets:

copy code

```
h1 { font-size: 24px; color: red; } p { font-size: 16px; color: black; }
```

In this example, an `<h1>` element will have a font size of 24 pixels and red text color, while a `<p>` element will have a font size of 16 pixels and black text color. These rule sets illustrate how CSS allows you to specify different styles for different elements.

Practical

Html.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="box">
        <div class=" col col-1">
            <img src="./img/img-(1).webp" alt="">
        </div>
        <div class=" col col-2">
            <img src="./img/img-(2).webp" alt="">
        </div>
        <div class=" col col-3">
            <div class="container">
                <nav>
```

```
                    <div class="logo">
                        <h1>image galary</h1>
                    </div>
                    <div class="link">
                        <ul>
                            <li><a
href="#">Home</a></li>
                            <li><a
href="#">photos</a></li>
                            <li><a
href="#">bio</a></li>
                            <li><a
href="#">blog</a></li>
                            <li><a
href="#">contact</a></li>
                        </ul>
                    </div>
                    <div class="icon">
                        <i class="fa-brands fa-square-
facebook"></i>
                        <i class="fa-brands fa-
twitter"></i>
                        <i class="fa-brands fa-
Instagram"></i>
                    </div>
                </nav>
            </div>

        </div>
        <div class=" col col-4">
            <img src="./img/images 4.jpeg" alt="">
        </div>
        <div class=" col col-5">
            <img src="./img/img-(5).jpeg" alt="">
```

```html
        </div>
        <div class=" col col-6">
            <img src="./img/img-(6).jpeg" alt="">
        </div>
        <div class=" col col-7">
            <img src="./img/img-(7).jpg" alt="">
        </div>
        <div class=" col col-8">
            <img src="./img/img-(8).jpeg" alt="">
        </div>
        <div class=" col col-9">
            <img src="./img/img-(9).jpeg" alt="">
        </div>
        <div class=" col col-10">
            <img src="./img/img-(10).webp" alt="">
        </div>
        <div class=" col col-11">
            <img src="./img/img-(3).webp" alt="">
        </div>

    </div>
</body>
</body>
</html>
```

Css.

```css
.box {
    display: grid;
    grid-template-rows: 400px 400px 400px 220px;
    grid-template-columns: repeat(11, 1fr);
    grid-auto-flow: row;
    gap: 10px;
    background-color: rgba(0, 0, 0, 0.796);
}

.col-1 {
    grid-column: span 6;
}

.col-2 {
    grid-column: span 3;
}

.col-3 {
    grid-column: span 2;
    grid-row: span 4;
    background-color: rgba(0, 0, 0, 0.47);
}

.col-4 {
    grid-column: span 2;
}

.col-5 {
    grid-column: span 5;
}

.col-6 {
    grid-column: span 2;
}

.col-7 {
    grid-column: span 5;
}

.col-8 {
    grid-column: span 4;
}

.col-9 {
    grid-column: span 3;
}

.col-10 {
    grid-column: span 3;
}

.col-11 {
    grid-column: span 3;
}

a:hover {
    color: red;

}

.container {
    width: 100%;
    text-align: right;
    color: white;
}

nav {
    padding: 20px;
}

li {
    list-style: none;
    padding: 10px 0;
```

```css
}

a {
    text-decoration: none;
    color: white;
    padding: 10px 0;
}


i {
    font-size: 25px;
    padding-left: 20px;
}

img {
    width: 100%;
    height: 100%;
}

@media (min-width:376px) {
    .box {
        display: grid;
        grid-template-rows: 70px 200px 200px 200px
200px 200px 200px 200px 200px 200px 200px;
        grid-template-columns: repeat(11, 1fr);
        gap: 10px;
        background-color: rgba(0, 0, 0, 0.796);
    }

    .col-1 {
        grid-column: span 11;
        order: 2;
    }
    .col-2 {
        grid-column: span 11;
        order: 3;
    }

    .col-3 {
        grid-column: span 11;
        background-color: rgba(0, 0, 0, 0.47);
        grid-row: 1;
        order: 1;
    }

    .col-4 {
        grid-column: span 11;
        order: 4;
    }

    .col-5 {
        grid-column: span 11;
        order: 5;
    }

    .col-6 {
        grid-column: span 11;
        order: 6;
    }

    .col-7 {
        grid-column: span 11;
        order: 7;
    }

    .col-8 {
        grid-column: span 11;
```

```css
        order: 8;
    }

    .col-9 {
        grid-column: span 11;
        order: 9;
    }

    .col-10 {
        grid-column: span 11;
        order: 10;
    }

    .col-11 {
        grid-column: span 11;
        order: 11;
    }

    .container {
        width: 100%;
        text-align: right;
        color: white;
    }

    nav {
        padding: 20px;
        text-align: center;
    }

    li {
        list-style: none;
        padding: 0px 10;
        display: inline-block;
    }

    a {
        text-decoration: none;
        color: white;
        padding: 10px 0;
    }

    i {
        display: none;
    }

}

@media (min-width:576px) {
    .container {
        width: 100%;
        text-align: right;
        color: white;
    }

    nav {
        padding: 0px;
        display: flex;
        justify-content: space-between;
        align-items: center;
        flex-wrap: wrap;

    }

    li {
        list-style: none;
        padding: 0px 5px;
        display: inline-block;
    }

    a {
        text-decoration: none;
        color: white;
        padding: 0px 5px;
    }

    i {
        font-size: 20px;
        padding-left: 5px;
        display: none;
    }
}

@media (min-width:762px) {
    .box {
        display: grid;
        grid-template-rows: 200px 200px 200px 200px
200px;
        grid-template-columns: repeat(11, 1fr);
        gap: 10px;
        background-color: rgba(0, 0, 0, 0.796);
    }

    .col-1 {
        grid-column: span 4;
        order: 1;
    }

    .col-2 {
        grid-column: span 4;
        order: 2;
```

```
}

.col-3 {
    grid-column: span 3;
    grid-row: span 5;
    background-color: rgba(0, 0, 0, 0.47);
    order: 3;
}

.col-4 {
    grid-column: span 4;
    order: 4;
}

.col-5 {
    grid-column: span 4;
    order: 5;
}

.col-6 {
    grid-column: span 4;
    order: 6;
}

.col-7 {
    grid-column: span 4;
    order: 7;
}

.col-8 {
    grid-column: span 4;
    order: 8;
}
```

```css
    .col-9 {
        grid-column: span 4;
        order: 9;
    }

    .col-10 {
        grid-column: span 4;
        order: 10;
    }

    .col-11 {
        grid-column: span 4;
        order: 11;
    }

    .container {
        width: 100%;
        text-align: right;
        color: white;
    }

    nav {
        padding: 20px;
        display: block;
    }

    li {
        list-style: none;
        padding: 10px 0;
        display: block;

    }

    a {
        text-decoration: none;
        color: white;
        padding: 10px 0;
    }

    i {
        display: none;
    }

}


@media (min-width:992px) {
    .box {
        display: grid;
        grid-template-rows: 400px 400px 400px 220px;
        grid-template-columns: repeat(11, 1fr);
        grid-auto-flow: row;
        gap: 10px;
        background-color: rgba(0, 0, 0, 0.796);
    }

    .col-1 {
        grid-column: span 6;
        order: 1;
    }

    .col-2 {
        grid-column: span 3;
        order: 2;
    }

    .col-3 {
        grid-column: span 2;
```

```css
        grid-row: span 4;
        background-color: rgba(0, 0, 0, 0.47);
        order: 3;
    }

    .col-4 {
        grid-column: span 2;
        order: 4;
    }

    .col-5 {
        grid-column: span 5;
        order: 5;
    }

    .col-6 {
        grid-column: span 2;
        order: 6;
    }

    .col-7 {
        grid-column: span 5;
        order: 7;
    }

    .col-8 {
        grid-column: span 4;
        order: 8;
    }

    .col-9 {
        grid-column: span 3;
        order: 9;
    }
    .col-10 {
        grid-column: span 3;
        order: 10;
    }

    .col-11 {
        grid-column: span 3;
        order: 111;
    }


}
```