RAJPUT UDAY RAVINDRSINH

**WEB DESIGING ASSIGMENT**

**MODULE -2 –[CSS & CSS 3]**

1. **What are the benefits of using CSS?**

**ANS :- Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a document written in HTML or XML. There are several benefits to using CSS in web development:**

**1. Separation of Content and Presentation.**

**2. Consistency.**

**3. Responsive Design.**

**4. Easy Updates and Maintenance.**

**5. Accessibility.**

**6. Animations and Transitions.**

**7.Print Styles.**

1. **What are the disadvantages of CSS?**

**ANS:- While CSS offers numerous advantages in web development, it also has some disadvantages:**

**1. Browser Compatibility.**

**2. Learning Curve.**

**3. Limited Layout Control.**

**4. Global Scope of Styles.**

**5. Performance Impact.**

**6. Limited Variables and Constants.**

**7. Cross-Browser Testing.**

**8. Lack of Dynamic Capabilities.**

**9. Inheritance Challenges.**

**10. Accessibility Challengess.**

1. **What is the difference between CSS2 and CSS3?**

**ANS:- CSS2 and CSS3 are different versions of the Cascading Style Sheet(CSS) language. With CSS3 being the latest and more feature-rich version.**

**EXAMPLE :-**

**CSS2 :- Basic Box model properties like width , height , margin , padding , and border.**

**CSS3 :- Adds feature like box-sizing , box-shadow , border-radius.**

/\* CSS2 \*/

div {

width: 200px;

height: 100px;

margin: 10px;

padding: 20px;

border: 2px solid #000;

}

/\* CSS3 \*/

div {

width: 200px;

height: 100px;

margin: 10px;

padding: 20px;

border: 2px solid #000;

border-radius: 10px;

box-shadow: 5px 5px 10px #888;

}

1. **Name a few CSS style components**

**ANS:- In CSS , Style components refer to the various properties and features that allow you to define the presentation and layout HTML elements. Here are few CSS style components :**

**COLOR :-**

**body {**

**color: #333;**

**background-color: #fff;**

**}**

**Typography :**

**p {**

**font-family: "Arial", sans-serif;**

**font-size: 16px;**

**font-weight: normal;**

**}**

**Layout :**

**.box {**

**width: 200px;**

**height: 100px;**

**margin: 10px;**

**padding: 20px;**

**border: 2px solid #000;**

**}**

**Flexbox :**

**.container {**

**display: flex;**

**justify-content: space-between;**

**align-items: center;**

**}**

1. **What do you understand by CSS opacity?**

**ANS :- In CSS, opacity refers to the level of transparency applied to an element. The `opacity` property is used to control the transparency of an element, allowing you to make it partially or fully transparent. The `opacity` property takes a value between 0 and 1, where 0 represents completely transparent (invisible), and 1 represents fully opaque (completely visible).**

**Here's the basic syntax for using the `opacity` property:**

**css**

**selector {**

**opacity: value;**

**}**

**css**

**.transparent-box {**

**opacity: 0.5;**

**}**

1. **How can the background color of an element be changed?**

**ANS :-** **You can change the background color of an HTML element using the `background-color` property in CSS. The `background-color` property sets the background color of an element, and you can specify the color using various formats, such as color names, hexadecimal values, RGB values.**

**Here's the basic syntax:**

**```css**

**selector {**

**background-color: color;**

**}**

**```**

**- \*\*Examples:\*\***

**1. Using a color name:**

**css**

**.example-element {**

**background-color: red;**

**}**

**2. Using a hexadecimal value:**

**css**

**.example-element {**

**background-color: #87CEEB;**

**}**

**3. Using RGB values:**

**css**

**.example-element {**

**background-color: rgb(0, 128, 0);**

**}**

**(7)** **How can image repetition of the backup be controlled?**

**ANS :- In CSS, you can control the repetition of a background image using the `background-repeat` property. This property allows you to specify whether and how a background image should repeat both horizontally and vertically. The values you can use are:**

**- `repeat`: The default value. The background image will repeat both horizontally and vertically.**

**- `repeat-x`: The background image will repeat only horizontally.**

**- `repeat-y`: The background image will repeat only vertically.**

**- `no-repeat`: The background image will not repeat at all.**

**Examples:**

**1. Repeat (default):**

**css**

**.example-element {**

**background-repeat: repeat;**

**}**

**2. Repeat Horizontally Only:**

**css**

**.example-element {**

**background-repeat: repeat-x;**

**}**

**3. Repeat Vertically Only:**

**css**

**.example-element {**

**background-repeat: repeat-y;**

**}**

**4. \*\*No Repeat:\*\***

**css**

**.example-element {**

**background-repeat: no-repeat;**

**}**

**(8)What is the use of the background-position property?**

**ANS :- The `background-position` property in CSS is used to specify the initial position of a background image within its containing element. It determines where the top left corner of the background image should be placed relative to the element's padding box. The property takes two values: one for the horizontal position and one for the vertical position.**

**Examples:-**

**1. Using Keywords:**

**css**

**.example-element {**

**background-position: top left;**

**}**

**2.Using Length Values:**

**css**

**.example-element {**

**background-position: 10px 20px;**

**}**

**3. Using Percentages:**

**css**

**.example-element {**

**background-position: 25% 75%;**

**}**

**4. Using a Combination of Keywords, Lengths, and Percentages:**

**css**

**.example-element {**

**background-position: center 30px;**

**}**

**(9) Which property controls the image scroll in the background?**

**ANS :- The property that controls the scrolling behavior of a background image is the `background-attachment` property. This property determines whether the background image scrolls with the content of an element or remains fixed in place as the user scrolls.**

**The `background-attachment` property can take the following values:**

**- `scroll`: The default value. The background image scrolls along with the content.**

**- `fixed`: The background image remains fixed in place as the user scrolls.**

**- `local`: The background image scrolls with the element's contents, and it doesn't move with respect to the containing block.**

**Examples:-**

**1. Scrolling Background (Default):**

**css**

**.example-element {**

**background-attachment: scroll;**

**}**

**2. Fixed Background:**

**css**

**.example-element {**

**background-attachment: fixed;**

**}**

**3. Local Scrolling Background:**

**css**

**.example-element {**

**background-attachment: local;**

**}**

**(10) Why should background and color be used as separate properties?**

**Separating the `background` and `color` properties in CSS is a good practice because it allows for more modular and maintainable code. The `background` property in CSS is a shorthand property that combines several background-related properties into one. These properties include `background-color`, `background-image`, `background-repeat`, `background-position`, `background-size`, `background-attachment`, and `background-clip`. Using the `background` property can be convenient when you want to set multiple background-related properties at once.**

**Here are a few reasons why it's beneficial to use `background` and `color` as separate properties:**

**1. \*\*Readability and Maintenance:\*\***

**- Separating properties makes your code more readable. It's easier to understand and modify individual aspects of the background or color without affecting the other.**

**2. \*\*Granular Control:\*\***

**- Using separate properties allows for granular control over different aspects of the background. You can modify the background color without changing the background image or vice versa.**

**3. \*\*Override Specific Properties:\*\***

**- If you have a default background set using the `background` property and later want to override only the background color, you can do so without specifying all the background-related properties.**

**```css**

**/\* Original background setting using shorthand property \*/**

**.element {**

**background: url('background.jpg') repeat-x fixed center / cover #fff;**

**}**

**/\* Override background color without modifying other background properties \*/**

**.element:hover {**

**background-color: #ffcc00;**

**}**

**```**

**4. \*\*Consistent Styles:\*\***

**- Keeping color and background properties separate helps maintain a consistent coding style across your project. It also aligns with the separation of concerns in web development, where content (color) and presentation (background) are distinct.**

**5. \*\*Easier Troubleshooting:\*\***

**- When troubleshooting issues related to background or color, separating properties makes it easier to identify and isolate the source of the problem.**

**6. \*\*Compatibility:\*\***

**- Some older browsers might not support certain shorthand properties or might interpret them differently. Using separate properties ensures better compatibility.**

**\*\*Example:\*\***

**```css**

**/\* Using separate background and color properties \*/**

**.element {**

**background-image: url('background.jpg');**

**background-repeat: repeat-x;**

**background-position: center;**

**background-size: cover;**

**background-color: #fff;**

**color: #333;**

**}**

**```**

**By using separate properties, you can tailor each aspect of the background and color independently, providing more flexibility and control in your stylesheets.**

**(11) How to center block elements using CSS1?**

**ANS :- CSS1, released in 1996, was the first version of the Cascading Style Sheets specification. It provided some basic styling capabilities compared to the more advanced features introduced in later versions. CSS1 didn't have explicit properties for centering block elements, so developers had to rely on other techniques, often involving properties like `text-align` and `margin`. Here's a simple example using CSS1:**

**EXAMPLE:**

**html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<style>**

**body {**

**text-align: center; /\* Center aligns inline content \*/**

**}**

**.center-block {**

**width: 50%; /\* Set a width for the block element \*/**

**margin-left: auto; /\* Auto margin left and right centers the block element \*/**

**margin-right: auto;**

**background-color: #f0f0f0; /\* Just for visibility \*/**

**padding: 20px;**

**}**

**</style>**

**<title>Center Block with CSS1</title>**

**</head>**

**<body>**

**<div class="center-block">**

**<p>This is a block element centered using CSS1 techniques.</p>**

**</div>**

**</body>**

**</html>**

**(12) How to maintain the CSS specifications?**

**ANS :- Maintaining CSS specifications involves staying informed about updates, adhering to best practices, and ensuring compatibility across different browsers. Here are some general guidelines for maintaining CSS specifications:**

**1. Stay Informed:**

**- Regularly check official sources for CSS updates, such as the W3C (World Wide Web Consortium) website (https://www.w3.org/Style/CSS/).**

**- Follow reputable blogs, forums, and social media channels related to web development and CSS to stay updated on best practices and new features.**

**2. Use Latest Specifications:**

**- Embrace and use the latest CSS specifications and features when they become widely supported. This ensures that your stylesheets benefit from improvements, optimizations, and new capabilities.**

**3. Browser Compatibility:**

**- Regularly test your CSS across different browsers to ensure compatibility. Use tools like BrowserStack, CrossBrowserTesting, or the browser developer tools to identify and address any issues.**

**4. Graceful Degradation and Progressive Enhancement:**

**- Implement graceful degradation and progressive enhancement strategies. This means designing your web pages to work even if certain CSS features are not supported or if users are using older browsers.**

**5. Modular and Maintainable Code:**

**- Write modular and maintainable CSS code by following best practices like BEM (Block, Element, Modifier) methodology, keeping stylesheets organized, and using a consistent naming convention.**

**6. Version Control:**

**- Use version control systems like Git to track changes in your CSS code. This helps you roll back changes if needed, collaborate with a team, and maintain a history of modifications.**

**7. Code Reviews:**

**- Conduct regular code reviews to ensure that CSS code aligns with best practices, is well-documented, and follows coding standards. This helps catch issues early and ensures consistency across the codebase.**

**(13) What are the ways to integrate CSS as a web page?**

**ANS:- There are several ways to integrate CSS into a web page, and the choice depends on the requirements of your project. Here are some common methods:**

**1. Inline CSS:**

**- Inline CSS is applied directly within the HTML document using the `style` attribute. This method is suitable for applying styles to a specific element.**

**html**

**<p style="color: blue; font-size: 16px;">This is a paragraph with inline styles.</p>**

**2. Internal/Embedded CSS:**

**- Internal or embedded CSS is placed within the HTML document, typically in the `<head>` section, using the `<style>` tag. This method is useful when styling multiple elements on the same page.**

**html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<style>**

**body {**

**background-color: #f0f0f0;**

**font-family: Arial, sans-serif;**

**}**

**h1 {**

**color: green;**

**}**

**</style>**

**<title>Internal CSS Example</title>**

**</head>**

**<body>**

**<h1>Hello, World!</h1>**

**<p>This is a paragraph with internal styles.</p>**

**</body>**

**</html>**

**3. External CSS:**

**- External CSS involves creating a separate CSS file and linking it to the HTML document. This method is recommended for maintaining a consistent style across multiple pages.**

**html**

**<!-- index.html -->**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<link rel="stylesheet" href="styles.css">**

**<title>External CSS Example</title>**

**</head>**

**<body>**

**<h1>Hello, World!</h1>**

**<p>This is a paragraph with external styles.</p>**

**</body>**

**</html>**

**css**

**/\* styles.css \*/**

**body {**

**background-color: #f0f0f0;**

**font-family: Arial, sans-serif;**

**}**

**h1 {**

**color: green;**

**}**

**(14) What is embedded style sheets?**

**Ans :- Embedded style sheets, also known as internal style sheets, involve placing the CSS code directly within the HTML document. This is done using the `<style>` element in the `<head>` section of the HTML file. The styles defined in the embedded style sheet apply only to the HTML document in which they are declared.**

**Here's an example of an HTML document with an embedded style sheet:**

**EXAMPLE :-**

**html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<style>**

**body {**

**font-family: Arial, sans-serif;**

**background-color: #f0f0f0;**

**}**

**h1 {**

**color: green;**

**}**

**p {**

**font-size: 16px;**

**line-height: 1.5;**

**}**

**</style>**

**<title>Embedded Style Sheet Example</title>**

**</head>**

**<body>**

**<h1>Hello, World!</h1>**

**<p>This is a paragraph with styles applied using an embedded style sheet.</p>**

**</body>**

**</html>**

**(15) What are the external style sheets?**

**ANS:- External style sheets are separate files containing CSS code that can be linked to multiple HTML documents. This approach promotes a clear separation of concerns by keeping HTML and CSS in separate files, making the code more modular, maintainable, and reusable across multiple pages.**

**To use an external style sheet:**

**1. Create the External CSS File:**

**- Create a separate file with a `.css` extension. For example, `styles.css`. This file will contain your CSS rules.**

**css**

**body {**

**font-family: Arial, sans-serif;**

**background-color: #f0f0f0;**

**}**

**h1 {**

**color: green;**

**}**

**p {**

**font-size: 16px;**

**line-height: 1.5;**

**}**

**2. Link the External CSS File to HTML:**

**- In your HTML document, use the `<link>` element in the `<head>` section to link to the external CSS file.**

**html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<link rel="stylesheet" href="styles.css">**

**<title>External Style Sheet Example</title>**

**</head>**

**<body>**

**<h1>Hello, World!</h1>**

**<p>This is a paragraph with styles applied using an external style sheet.</p>**

**</body>**

**</html>**

**(16) What are the advantages and disadvantages of using external style sheets?**

**ANS:- Using external style sheets in web development comes with several advantages and a few potential disadvantages. Here's an overview:**

**### Advantages:**

**1. Reusability:**

**- Styles defined in an external style sheet can be reused across multiple HTML documents. This ensures a consistent design and layout across the entire website.**

**2. Maintenance:**

**- Changes to the styling can be made in a single location—the external CSS file. This makes maintenance more efficient, as updates are automatically reflected on all linked HTML pages.**

**3. Separation of Concerns:**

**- External style sheets contribute to a clear separation of concerns between HTML (structure) and CSS (presentation). This improves code organization and readability.**

**4. Consistency:**

**- External style sheets help maintain a consistent design throughout the website. Design elements such as fonts, colors, and spacing are easily controlled and updated from one central location.**

**5. Caching:**

**- Advantage: External style sheets are cached by the browser after the first visit. This can lead to faster page loading times for subsequent visits to the site.**

**### Disadvantages:**

**1. Additional HTTP Request:**

**- Each external style sheet requires an additional HTTP request, which may slightly increase the page loading time, particularly on the first visit.**

**2. Render Blocking:**

**- External style sheets can be render-blocking, meaning that they may prevent the rendering of the page until the stylesheet is loaded. This can affect the perceived performance of the website.**

**3. Not Suitable for Inline Styles:**

**- External style sheets are not suitable for styles that are specific to a single HTML document or a small portion of a page. For such cases, inline or internal styles may be more appropriate.**

**4. Dependency:**

**- If the external style sheet fails to load or if there are issues with the network connection, it may impact the styling of the entire website.**

**5. Potential for Overuse:**

**- In some cases, developers may create too many external style sheets, leading to a complex and difficult-to-manage system. Proper organization and consolidation of stylesheets are essential.**

**(17) What is the meaning of the CSS selector?**

**ANS :- In CSS (Cascading Style Sheets), a selector is a pattern used to select and style one or more HTML elements. Selectors define the scope of the styles to be applied, determining which elements on the HTML page will be affected by the associated CSS rules.**

**### Types of Selectors:**

**1. Type or Tag Selector:**

**- Selects all instances of a specific HTML tag.**

**css**

**p {**

**color: blue;**

**}**

**2. Class Selector:**

**- Selects elements with a specific class attribute.**

**css**

**.highlight {**

**background-color: yellow;**

**}**

**3. ID Selector:**

**- Selects a single element with a specific ID attribute.**

**css**

**#header {**

**font-size: 24px;**

**}**

**4. Descendant Selector:**

**- Selects an element that is a descendant of another specified element.**

**css**

**article p {**

**font-style: italic;**

**}**

**5. Child Selector:**

**- Selects an element that is a direct child of another specified element.**

**css**

**nav > ul {**

**list-style-type: none;**

**}**

**6. Adjacent Sibling Selector:**

**- Selects an element that is an immediate sibling of another specified element.**

**css**

**h2 + p {**

**font-weight: bold;**

**}**

**7. Attribute Selector:**

**- Selects elements based on the presence or value of their attributes.**

**css**

**input[type="text"] {**

**border: 1px solid #ccc;**

**}**

**8. Pseudo-Class Selector:**

**- Selects elements based on their state or position, such as `:hover`, `:focus`, or `:nth-child()`.**

**css**

**a:hover {**

**text-decoration: underline;**

**}**

**9. Pseudo-Element Selector:**

**- Selects a specific part of an element, like the first line or first letter.**

**css**

**p::first-line {**

**font-weight: bold;**

**}**

**(18) What are the media types allowed by CSS?**

**ANS :- CSS (Cascading Style Sheets) allows you to define styles for different media types, allowing you to tailor the presentation of your content for various devices and situations. The `@media` rule is used to apply styles based on different media types. Here are some common media types:**

**1. all:**

**- This is the default media type, and styles are applied to all devices.**

**css**

**@media all {**

**/\* Styles for all devices \*/**

**}**

**2. print:**

**- Styles for printed pages. These styles are applied when a user prints the web page.**

**css**

**@media print {**

**/\* Styles for printed pages \*/**

**}**

**3. screen:**

**- Styles for computer screens, tablets, and smartphones.**

**css**

**@media screen {**

**/\* Styles for screens \*/**

**}**

**4. speech:**

**- Styles for screen readers that convert text to speech or other audio presentations.**

**css**

**@media speech {**

**/\* Styles for screen readers \*/**

**}**

**5. braille:**

**- Styles for braille tactile feedback devices.**

**css**

**@media braille {**

**/\* Styles for braille devices \*/**

**}**

**6. embossed:**

**- Styles for paged braille printers.**

**css**

**@media embossed {**

**/\* Styles for embossed braille printers \*/**

**}**

**7. handheld:**

**- Styles for handheld devices like mobile phones.**

**css**

**@media handheld {**

**/\* Styles for handheld devices \*/**

**}**

**8. projection:**

**- Styles for projected presentations, such as slideshows.**

**css**

**@media projection {**

**/\* Styles for projected presentations \*/**

**}**

**9. screen and (max-width: 600px):**

**- Styles for screens with a maximum width of 600 pixels. This is an example of using media queries to conditionally apply styles based on specific conditions.**

**css**

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

**/\* Styles for screens with a max-width of 600px \*/**

**}**

**(19) What is the rule set?**

**ANS :- In CSS (Cascading Style Sheets), a rule set is a combination of a selector and a declaration block. It defines the styles that should be applied to HTML elements that match the specified selector. The basic syntax of a CSS rule set is as follows:**

**```css**

**selector {**

**property1: value1;**

**property2: value2;**

**/\* additional properties and values \*/**

**}**

**```**

**Let's break down the components of a rule set:**

**- \*\*Selector:\*\* Specifies which HTML elements the rule set should apply to. Selectors can be based on element names, IDs, classes, attributes, or a combination of these.**

**```css**

**/\* Example rule set with an element selector \*/**

**p {**

**color: blue;**

**font-size: 16px;**

**}**

**/\* Example rule set with a class selector \*/**

**.highlight {**

**background-color: yellow;**

**}**

**/\* Example rule set with an ID selector \*/**

**#header {**

**font-size: 24px;**

**}**

**```**

**- \*\*Declaration Block:\*\* Consists of one or more property-value pairs enclosed in curly braces `{}`. Each property-value pair defines a specific style attribute and its corresponding value.**

**```css**

**/\* Example declaration block \*/**

**{**

**color: blue;**

**font-size: 16px;**

**}**

**```**

**- \*\*Properties and Values:\*\* Properties represent specific style attributes (e.g., `color`, `font-size`), and values define the settings for those attributes (e.g., `blue`, `16px`).**

**```css**

**/\* Example rule set with properties and values \*/**

**p {**

**color: blue;**

**font-size: 16px;**

**}**

**```**

**Multiple properties and values can be included in a rule set to define various styles for the selected elements.**

**### Example:**

**Consider the following CSS rule set:**

**```css**

**/\* Rule set with a class selector and a declaration block \*/**

**.my-text {**

**color: #333;**

**font-size: 18px;**

**font-family: 'Arial', sans-serif;**

**}**

**```**

**- \*\*Selector:\*\* `.my-text` is a class selector, and this rule set will apply to all HTML elements with the class name "my-text."**

**- \*\*Declaration Block:\*\* The block contains three property-value pairs specifying the styles for elements with the class "my-text."**

**When applied to HTML elements like `<p class="my-text">This is a styled text.</p>`, the styles defined in the rule set will be applied to those elements.**

**(20) Create Layouts**

**ANS :- Creating layouts in CSS involves structuring the HTML and applying styles to achieve the desired arrangement of elements on a web page. There are several approaches to creating layouts, and the choice depends on the design requirements and preferences. Here, I'll provide examples of two commonly used layout techniques: the \*\*Float-based Layout\*\* and the \*\*Flexbox Layout\*\*.**

**### 1. Float-based Layout:**

**Floats were traditionally used for layout purposes. Elements are floated to the left or right, allowing other content to flow around them. This approach is suitable for simple layouts.**

**```html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<style>**

**/\* Clearfix for containing floated elements \*/**

**.clearfix::after {**

**content: "";**

**display: table;**

**clear: both;**

**}**

**/\* Style for the main container \*/**

**.container {**

**width: 100%;**

**max-width: 960px;**

**margin: 0 auto;**

**background-color: #f0f0f0;**

**padding: 20px;**

**}**

**/\* Style for columns \*/**

**.column {**

**width: 48%;**

**float: left;**

**margin: 1%;**

**box-sizing: border-box;**

**}**

**/\* Responsive layout for smaller screens \*/**

**@media (max-width: 768px) {**

**.column {**

**width: 100%;**

**margin: 0;**

**}**

**}**

**</style>**

**<title>Float-based Layout</title>**

**</head>**

**<body>**

**<div class="container clearfix">**

**<div class="column">**

**<h2>Column 1</h2>**

**<p>This is the content of column 1.</p>**

**</div>**

**<div class="column">**

**<h2>Column 2</h2>**

**<p>This is the content of column 2.</p>**

**</div>**

**</div>**

**</body>**

**</html>**

**```**

**### 2. Flexbox Layout:**

**Flexbox is a modern layout model that provides a more efficient way to create flexible and responsive layouts. It allows you to align and distribute space among items in a container even when their size is unknown or dynamic.**

**```html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<style>**

**/\* Style for the main container using Flexbox \*/**

**.container {**

**display: flex;**

**justify-content: space-between;**

**flex-wrap: wrap;**

**background-color: #f0f0f0;**

**padding: 20px;**

**}**

**/\* Style for items within the container \*/**

**.item {**

**width: 48%;**

**box-sizing: border-box;**

**margin-bottom: 1%;**

**}**

**/\* Responsive layout for smaller screens \*/**

**@media (max-width: 768px) {**

**.item {**

**width: 100%;**

**margin: 0;**

**}**

**}**

**</style>**

**<title>Flexbox Layout</title>**

**</head>**

**<body>**

**<div class="container">**

**<div class="item">**

**<h2>Item 1</h2>**

**<p>This is the content of item 1.</p>**

**</div>**

**<div class="item">**

**<h2>Item 2</h2>**

**<p>This is the content of item 2.</p>**

**</div>**

**</div>**

**</body>**

**</html>**

**```**