Unit 3

Q 1) Answer the following in brief.
I) What is an ID selector in CSS ?
II) What are style sheets?

Ans.) I) An ID selector in CSS is a way to select and apply styles to a specific element on a web page based on its unique identifier. In CSS, an ID selector is denoted by the "#" symbol followed by the ID name. For example, if you have an HTML element with the ID "myElement", you can target and style it using the CSS ID selector "#myElement". ID selectors are used to apply specific styles to individual elements, and they should be unique within a web page because an ID can only be assigned to a single element.

II) Style sheets in the context of web development refer to files that contain a set of rules and instructions for how the content of a web page should be presented. They are written using CSS (Cascading Style Sheets) and are used to define the visual appearance of elements on a website. Style sheets separate the presentation layer from the structure and content of a web page, allowing developers to define styles once and apply them to multiple elements throughout the site.

By using style sheets, developers can specify various aspects of the design, such as colors, fonts, margins, padding, and layout properties. Style sheets can be applied to HTML documents by including them within the `<style>` tags in the HTML file or by linking to an external CSS file using the `<link>` tag. This separation of style from content promotes consistency, maintainability, and flexibility in web design.

Q 2) Answer the following in detail.
I) Design a form in a colourful way which include username and password and a hyperlink which allows the user to navigate to another form.
II) Explain the pseudo class selector in CSS with an example.
III) Explain various ways to apply CSS styles to a web page.

Ans.) I) Designing a Form with Colorful Elements:
To design a form in a colorful way that includes a username and password field along with a hyperlink to another form, you can use CSS to style the form elements. Here's an example of how you can achieve this:

HTML:
```
<form>
  <label for="username">Username:</label>
  <input type="text" id="username" name="username">
  <br>
  <label for="password">Password:</label>
  <input type="password" id="password" name="password">
```

~~~
  <br>
  <a href="another-form.html">Go to another form</a>
</form>
```
~~~

CSS:

~~~
form {
  width: 300px;
  padding: 20px;
  background-color: #f5f5f5;
  border: 1px solid #ccc;
}

label {
  display: block;
  color: #333;
  font-weight: bold;
  margin-bottom: 10px;
}

input[type="text"],
input[type="password"] {
  width: 100%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

a {
  color: #ff00ff;
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}
```
~~~

II) Pseudo-class Selectors in CSS:

Pseudo-class selectors in CSS allow you to select and style elements based on their state or position within the document. They start with a colon (":") and follow a specific pattern. Here's an explanation with an example:

Example:
```
a:link {
  color: blue;
}

a:hover {
  color: red;
}

input:focus {
  border-color: green;
}
```

In this example, we have three pseudo-class selectors:

1. `:link` is used to select and style links that haven't been visited yet. In this case, it sets the color of unvisited links to blue.

2. `:hover` selects and styles elements when the user hovers over them with the mouse. In this case, it changes the color of links to red when hovered.

3. `:focus` selects and styles elements that have received focus, typically through user interaction, like clicking or tabbing. In this case, it changes the border color of the input element to green when it is in focus.

III) Various Ways to Apply CSS Styles to a Web Page:

1. Inline CSS: You can apply styles directly to individual HTML elements using the `style` attribute. For example:
```html
<h1 style="color: blue;">Hello, World!</h1>
```
However, inline CSS should be used sparingly as it can become difficult to maintain and update styles across multiple elements.

2. Internal CSS: You can embed CSS rules within the `<style>` tags in the `<head>` section of an HTML document. For example:
```html
<head>
  <style>
    h1 {
      color: blue;
    }
  </style>
</head>
<body>
  <h1>Hello, World!</h1>
</body>
```
This method allows you to define styles within the HTML file itself, but it's still limited to a single document.

3. External CSS: You can create a separate CSS file with a `.css` extension and link it to your HTML file using the `<link>` tag within the `<head>` section. For example:
```html
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Hello, World!</h1>
</body>
```
The external CSS file contains all the styles and can be reused across multiple HTML documents, promoting consistency and easier maintenance.

4. CSS Frameworks: CSS frameworks, such as Bootstrap, Foundation, or Bulma, provide pre-defined CSS styles and components to streamline web development. By including the framework's CSS file or using a CDN link, you can leverage the framework's styles and classes to style your web page quickly.

5. CSS Preprocessors: CSS preprocessors like Sass or Less extend the capabilities of CSS by adding variables, nesting, mixins, and more. These preprocessors require compiling into regular CSS before being used in HTML.

These are some of the common methods to apply CSS styles to a web page, each with its own advantages and use cases. Choosing the appropriate method depends on factors such as project size, scalability, and code organization requirements.

Q 3) Answer the following in brief.

I) State the general format of linking CSS with HTML.

II) Write a CSS to define text color and alignment to <p> tag.

Ans.) I) The general format of linking CSS with HTML is done using the `<link>` tag within the `<head>` section of an HTML document. Here is the basic structure:

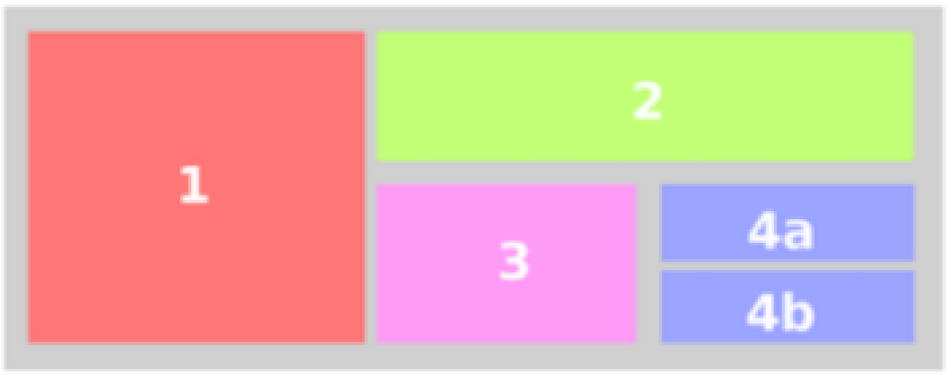
```html
<head>
  <link rel="stylesheet" href="path/to/your/css/file.css">
</head>
```

II) CSS to define text color and alignment to `<p>` tag:

To define text color and alignment for `<p>` (paragraph) tag, you can use the following CSS:

```css
p {
  color: red;
  text-align: center;
}
```

Q 4) Answer the following in detail.
I) List and explain the font and text element properties and values used in CSS.
II) Write an HTML and CSS code for displaying the flexbox items as shown in the given figure.



III) Explain CSS transitions with an example.

Ans.) I) Font and Text Element Properties and Values in CSS:
CSS provides a range of properties to control the appearance and behaviour of fonts and text elements. Here are some commonly used properties along with their values:

1. Font Family (`font-family`): Specifies the typeface or font family for the text. It can be set to either a specific font name (e.g., Arial, Times New Roman) or a generic font

family (e.g., sans-serif, serif, monospace). Multiple font names can be listed, separated by commas, to define fallback options.

2. Font Size (`font-size`): Sets the size of the font. It can be defined using various units such as pixels (`px`), ems (`em`), or percentages (%).

3. Font Weight (`font-weight`): Controls the thickness or boldness of the text. Common values include `normal`, `bold`, `bolder`, and `lighter`. Numeric values (`100`, `200`, etc.) can also be used.

4. Font Style (`font-style`): Defines the style of the text, such as italic or normal. Common values are `normal`, `italic`, and `oblique`.

5. Text Color (`color`): Determines the color of the text. It can be specified using color names (e.g., red, blue) or hexadecimal color codes (#RRGGBB) or RGB values (rgb(255, 0, 0)).

6. Text Alignment (`text-align`): Sets the horizontal alignment of the text within its container. Values include `left`, `right`, `center`, and `justify` (for justified alignment).

7. Text Decoration (`text-decoration`): Adds visual effects to the text, such as underlining, overlining, or striking through. Common values are `none`, `underline`, `overline`, and `line-through`.

8. Text Transform (`text-transform`): Changes the capitalization of the text. Values include `none`, `uppercase`, `lowercase`, and `capitalize`.

9. Text Indent (`text-indent`): Specifies the indentation of the first line of text within a block element. It can be set to a specific length value (e.g., pixels, ems) or a percentage.

II) HTML and CSS code for displaying Flexbox items:

To display Flexbox items as shown in a given figure, here's an example of HTML and CSS code:

HTML:
```html
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

CSS:
```css
.container {
  display: flex;
  justify-content: space-between;
}

.item {
  width: 100px;
  height: 100px;
  background-color: teal;
  color: white;
  display: flex;
  justify-content: center;
  align-items: center;
}
```

III) CSS Transitions with an Example:

CSS transitions allow smooth animation effects when properties of an element change. They are used to create gradual transitions between different states of an element. Here's an example:

HTML:
```html
<button class="btn">Hover Me</button>
```

CSS:
```css
.btn {
  background-color: teal;
  color: white;
  padding: 10px 20px;
  border: none;
  transition: background-color 0.3s ease;
}

.btn:hover {
  background-color: tomato;
}
```

Q 5) Answer the following in brief.
I) Write a CSS code to set background image with non-repeating property.
II) Explain the justify-content property of a grid layout.
Ans.) I) CSS code to set a non-repeating background image:

To set a background image with the non-repeating property in CSS, you can use the `background-image` and `background-repeat` properties. Here's an example:

```css
.element {
  background-image: url("path/to/image.jpg");
  background-repeat: no-repeat;
}
```

II) Explanation of the `justify-content` property in a grid layout:

The `justify-content` property is used in CSS Grid layouts to control the alignment and positioning of grid items along the horizontal axis (the "main axis"). It allows you to distribute and space the items within a grid container.

The `justify-content` property accepts various values:

1. `flex-start`: Items are packed towards the start (left) of the container.

2. `flex-end`: Items are packed towards the end (right) of the container.

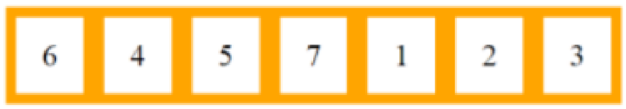
3. `center`: Items are centered horizontally within the container.

4. `space-between`: Items are evenly distributed with space placed between them, but no space at the edges.

5. `space-around`: Items are evenly distributed with equal space around them, including at the edges.

6. `space-evenly`: Items are evenly distributed with equal space around them, including at the edges and between each item.

Q 6) Answer the following in detail.
I) Explain various ways to apply CSS styles to a web page.
II) Explain box model in detail.
III) Write an HTML code for displaying the flexbox items as shown in given figure.



Ans.) I) Various Ways to Apply CSS Styles to a Web Page:
1. Inline CSS: Inline styles are applied directly to individual HTML elements using the `style` attribute. For example:
```html
<h1 style="color: blue;">Hello, World!</h1>
```
Inline CSS should be used sparingly as it can be hard to maintain and update styles across multiple elements.

2. Internal CSS: Internal styles are defined within the `<style>` tags in the `<head>` section of an HTML document. For example:
```html
<head>
  <style>
    h1 {
      color: blue;
    }
  </style>
</head>
<body>
  <h1>Hello, World!</h1>
</body>
```
This method allows you to define styles within the HTML file itself, but it's limited to a single document.

3. External CSS: External styles are defined in a separate CSS file with a `.css` extension and linked to the HTML file using the `<link>` tag within the `<head>` section. For example:
```html
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Hello, World!</h1>
</body>
```
The external CSS file contains all the styles and can be reused across multiple HTML documents, promoting consistency and easier maintenance.

4. CSS Frameworks: CSS frameworks, such as Bootstrap, Foundation, or Bulma, provide pre-defined CSS styles and components to streamline web development. By including the framework's CSS file or using a CDN link, you can leverage the framework's styles and classes to style your web page quickly.

5. CSS Preprocessors: CSS preprocessors like Sass or Less extend the capabilities of CSS by adding variables, nesting, mixins, and more. These preprocessors require compiling into regular CSS before being used in HTML.

6. CSS-in-JS: With CSS-in-JS libraries like Styled Components or Emotion, you can write CSS styles directly within your JavaScript code. These libraries allow you to define styles as JavaScript objects or template literals, which are then dynamically injected into the HTML.

These are some of the common methods to apply CSS styles to a web page, each with its own advantages and use cases. Choosing the appropriate method depends on factors such as project size, scalability, code organization requirements, and personal preference.

II) Box Model in Detail:

The box model is a fundamental concept in CSS that describes how elements are rendered and how their dimensions are calculated. It consists of four layers, or components, that wrap around HTML elements:

1. Content: The actual content of an element, such as text, images, or other HTML elements. It is represented by the innermost layer.

2. Padding: The space between the content and the element's border. Padding is transparent and does not have a background color. It can be set using the `padding` property.

3. Border: The border around the padding and content. It can have a specific width, color, and style defined using the `border` property.

4. Margin: The space between the element's border and adjacent elements. Margin is transparent and does not have a background color. It can be set using the `margin` property.

The box model is crucial for understanding how elements are sized and positioned in CSS. The total width of an element is calculated by adding the content width, padding width, border width, and margin width together. Similarly, the total height is determined by summing the content height, padding height, border height, and margin height.

By manipulating the properties related to the box model, you can control the size, spacing, and appearance of elements on your web page. Understanding the box model is essential for building responsive layouts and handling spacing and

alignment effectively.

III) HTML Code for Displaying Flexbox Items as Shown in a Given Figure:

To display Flexbox items as shown in a given figure, here's an example of HTML code:

```html
<div class="container">
```

```html
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

In this example, we have a container `<div>` element with a class of "container" that holds three item `<div>` elements. Each item is represented by a `<div>` with a class of "item". You can modify the content of the items as needed.

To style the Flexbox layout, you can use CSS. Here's an example of CSS code that aligns the items horizontally:

```css
.container {
  display: flex;
  justify-content: space-between;
}

.item {
  width: 100px;
  height: 100px;
  background-color: teal;
  color: white;
}
```

In this CSS code, the container is set to `display: flex`, which activates Flexbox layout for its children. The `justify-content: space-between` property is used to distribute the items evenly with space between them. Each item has a fixed width and height of `100px`, a background color of teal, and white text color.

Q 7) Answer the following in brief.
I) Explain the overflow property of Box.
II) Enlist the types of position property in CSS.

Ans.) I) The `overflow` property of a box in CSS determines how content that exceeds the dimensions of the box is handled. It controls the behavior of content that overflows beyond the visible area of the box. The `overflow` property can take the following values:

- `visible` (default): Content is not clipped and may extend beyond the box boundaries.
- `hidden`: Content that exceeds the box dimensions is clipped and not visible.
- `scroll`: A scrollbar is always visible, allowing the user to scroll through the content that exceeds the box dimensions.

- `auto`: A scrollbar is displayed only when necessary. If the content fits within the box, no scrollbar is shown.
- `overlay`: Similar to `auto`, a scrollbar is displayed when necessary, but it is overlaid on top of the content, rather than taking up space within the box.

The `overflow` property is commonly used for elements with a fixed height and width to control the visibility and scrolling behavior of content that exceeds the box boundaries.

II) In CSS, there are four types of position property:

1. `static` (default): Elements are positioned according to the normal flow of the document. This is the default behavior, and `top`, `bottom`, `left`, `right`, and `z-index` properties have no effect.

2. `relative`: Elements are positioned relative to their normal position within the document flow. By using `top`, `bottom`, `left`, and `right` properties, the element can be offset from its normal position. Other elements on the page will not be affected by the relative positioning.

3. `absolute`: Elements are removed from the normal document flow and positioned relative to the nearest positioned ancestor or the containing block. If no positioned ancestor is found, the element will be positioned relative to the initial containing block (usually the window). Absolute positioning allows precise control over the element's position using `top`, `bottom`, `left`, and `right` properties.

4. `fixed`: Elements are removed from the normal document flow and positioned relative to the viewport (the browser window). It stays fixed even when the page is scrolled. Like absolute positioning, `top`, `bottom`, `left`, and `right` properties are used to determine the position of the element.

The position property is often used in combination with the `top`, `bottom`, `left`, and `right` properties to precisely position elements on a web page. It provides flexibility in layout and positioning elements relative to their normal flow or other elements.

Q 8) Answer the following in detail.
I) Enlist different types of selectors in CSS. Explain anyone with an example.
II) Explain flex-items properties with an example.
III) Explain text-indent and text-transform CSS properties with appropriate HTML code.

Ans.) I) Different Types of Selectors in CSS:
CSS selectors are used to target specific HTML elements and apply styles to them. Here are some commonly used selectors:

1. Type Selector: Targets elements based on their HTML tag name. For example, to select all `<h1>` elements, you can use:
```css
h1 {
  color: red;
}
```

2. Class Selector: Targets elements based on their class attribute. The class selector is denoted by a dot (`.`) followed by the class name. For example, to select all elements with the class "highlight":
```css
.highlight {
  background-color: yellow;
}
```

3. ID Selector: Targets elements based on their ID attribute. The ID selector is denoted by a hash (`#`) followed by the ID name. IDs should be unique within an HTML document. For example, to select an element with the ID "myElement":
```css
#myElement {
  border: 1px solid black;
}
```

4. Attribute Selector: Targets elements based on their attributes. It can match elements based on attribute names, values, or a combination of both. For example, to select all `<input>` elements with the `type` attribute set to "text":
```css
input[type="text"] {
  background-color: lightblue;
}
```

5. Pseudo-Class Selector: Targets elements based on a specific state or condition. Pseudo-classes are preceded by a colon (`:`). For example, to select the first child of a `<ul>` element:
```css
ul li:first-child {
  font-weight: bold;
}
```

6. Pseudo-Element Selector: Targets specific parts of an element, such as the first letter or first line. Pseudo-elements are also preceded by a colon (`:`). For example, to style the first letter of a `<p>` element:
```css
p::first-letter {
  font-size: 24px;
}
```

II) Flexbox Item Properties with an Example:

Flexbox is a CSS layout module that provides a flexible way to arrange and align elements within a container. The properties below apply to individual flex items within a flex container:

1. `flex-grow`: Specifies how much a flex item should grow relative to other flex items within the same container. It takes a unitless value, which represents the proportionate share of the available space. For example:
```css
.item {
  flex-grow: 1;
}
```
In this example, the flex item will grow to occupy the available space, along with other items that have the same `flex-grow` value.

2. `flex-shrink`: Specifies how much a flex item should shrink when there is not enough space in the container. It also takes a unitless value, indicating the relative shrink factor. For example:
```css
.item {
  flex-shrink: 0;
}
```
In this case, the flex item will not shrink, maintaining its original size even when there is limited space.

3. `flex-basis`: Defines the initial size of a flex item before free space is distributed. It can be specified using a length value (e.g., `px`, `em`) or a percentage value. For example:
```css
.item {
  flex-basis: 200px;
}
```
Here, the flex item will have an initial width of 200 pixels.

4. `flex`: Shorthand property that combines `flex-grow`, `flex-shrink`, and `flex-basis` properties in one declaration. For example:
```css
.item {
  flex: 1 0 200px;
}
```
This example sets the `flex-grow` value to 1, `flex-shrink` value to 0, and `flex-basis` value to 200 pixels.

III) Explanation of `text-indent` and `text-transform` CSS Properties with Appropriate HTML Code:

1. `text-indent`: This CSS property controls the indentation of the first line of text within an element. It is commonly used to create paragraphs with an indented first line. The value can be specified in pixels (`px`), ems (`em`), or percentages (`%`). For example:
```css
p {
  text-indent: 20px;
}
```
In this example, the first line of every `<p>` element will be indented by 20 pixels.

2. `text-transform`: This CSS property is used to change the capitalization or case of text within an element. It offers several values:
- `uppercase`: Transforms the text to all uppercase letters.
- `lowercase`: Transforms the text to all lowercase letters.
- `capitalize`: Capitalizes the first letter of each word in the text.
- `none` (default): No transformation is applied.

For example:
```css
h1 {
  text-transform: uppercase;
}
```
In this case, all the text within `<h1>` elements will be transformed to uppercase.

Here's an example HTML code snippet demonstrating the use of `text-indent` and `text-transform` properties:

```html
<p class="indented-text">This is an example paragraph with an indented first line.</p>

<h1 class="uppercase-text">This heading will be displayed in uppercase letters.</h1>
```

```css
.indented-text {
  text-indent: 20px;
}

.uppercase-text {
  text-transform: uppercase;
}
```

In the above example, the first line of the paragraph will be indented by 20 pixels, and the heading will be displayed in uppercase letters due to the respective CSS styles applied using `text-indent` and `text-transform` properties.

Q 9) Answer the following in brief.
I) Enlist benefits of CSS.
II) Explain the CSS shorthand margin property.
Ans.) I) Benefits of CSS:

1. Separation of Concerns: CSS allows for the separation of style and presentation from the structure and content of a web page. This separation makes the code more maintainable, modular, and easier to update.

2. Consistency: CSS enables consistent styling across multiple web pages. By defining styles in a centralized CSS file, you can ensure a uniform appearance and layout throughout your website.

3. Efficiency: CSS reduces the file size of web pages by eliminating the need for inline styles. External CSS files can be cached by the browser, leading to faster page load times and improved performance.

4. Responsive Design: CSS provides powerful tools for creating responsive layouts and adapting the design of web pages to different screen sizes and devices. Media queries allow you to apply different styles based on the user's viewport.

5. Flexibility and Control: CSS offers a wide range of styling options and properties, allowing for precise control over the appearance of elements. It provides the ability to customize colors, fonts, spacing, borders, and much more.

6. Maintainability: With CSS, you can make global changes to the design of a website by simply modifying a few lines of code in the CSS file. This saves time and effort compared to updating styles individually in multiple HTML files.

7. Browser Compatibility: CSS is supported by all modern web browsers, making it a reliable and widely adopted technology. It helps ensure consistent rendering across different browsers and platforms.

II) Explanation of the CSS Shorthand Margin Property:

The CSS shorthand `margin` property allows you to set the margins for an element in a concise way. It combines four individual margin properties: `margin-top`, `margin-right`, `margin-bottom`, and `margin-left`.

The syntax for the shorthand `margin` property is as follows:

```css
margin: top right bottom left;
```

You can specify one, two, three, or four values to define the margins. Here are the different ways you can use the `margin` shorthand:

- One value: Sets the same margin for all sides.
```css
margin: 10px;
```

- Two values: The first value sets the top and bottom margins, while the second value sets the right and left margins.
```css
margin: 10px 20px;
```

- Three values: The first value sets the top margin, the second value sets the right and left margins, and the third value sets the bottom margin.

```css
margin: 10px 20px 30px;
```

- Four values: Sets the top, right, bottom, and left margins individually.

```css
margin: 10px 20px 30px 40px;
```

Q 10) Answer the following in detail.
I) Explain various grid container properties.
II) Explain any five text formatting properties of CSS.
III) Explain CSS border properties.
Ans.) I) Explanation of Various Grid Container Properties:

CSS Grid Layout is a powerful two-dimensional grid system that allows for complex grid-based layouts. Here are some important properties used on grid containers:

1. `display`: Sets the element as a grid container. The value `grid` is used to create a block-level grid container, and `inline-grid` is used for an inline-level grid container.

2. `grid-template-rows` and `grid-template-columns`: Defines the size and number of rows and columns in the grid. You can specify the size using absolute values (e.g., pixels) or relative values (e.g., percentages, fr units).

3. `grid-template-areas`: Defines named grid areas, which can be referenced by grid items. This property allows for easy placement and rearrangement of items using a visual grid representation.

4. `grid-gap` or `gap`: Specifies the size of the gap between grid rows and columns. It can be a single value to set both row and column gaps or separate values to define row and column gaps individually.

5. `justify-items`: Aligns grid items along the horizontal axis within their grid cells. It accepts values like `start`, `end`, `center`, `stretch`, and `baseline`.

6. `align-items`: Aligns grid items along the vertical axis within their grid cells. It accepts values like `start`, `end`, `center`, `stretch`, and `baseline`.

7. `justify-content`: Aligns the grid along the horizontal axis within the grid container. It defines how the free space is distributed between and around the grid items. Values include `start`, `end`, `center`, `space-between`, `space-around`, and more.

8. `align-content`: Aligns the grid along the vertical axis within the grid container. It defines how the free space is distributed between and around the grid rows. Values include `start`, `end`, `center`, `space-between`, `space-around`, and more.

These grid container properties provide control over the size, alignment, and positioning of grid items within a grid layout. They allow for the creation of flexible and responsive grid-based designs.

II) Explanation of Five Text Formatting Properties in CSS:

1. `text-align`: Specifies the horizontal alignment of text within an element. Common values are `left`, `right`, `center`, and `justify`. For example:
```css
p {
  text-align: center;
}
```
This will center-align the text within all `<p>` elements.

2. `text-decoration`: Adds a visual decoration to text. It can be used to underline, overline, strike through, or apply other decorations to text. For example:
```css
a {
  text-decoration: underline;
}
```
This will underline all hyperlinks (`<a>` elements).

3. `text-transform`: Controls the capitalization of text. It allows you to convert text to uppercase, lowercase, capitalize the first letter of each word, or leave it unchanged. For example:
```css
h1 {
  text-transform: uppercase;
}
```
This will transform the text in all `<h1>` elements to uppercase.

4. `letter-spacing`: Adjusts the spacing between characters in text. It can be used to increase or decrease the space between letters. For example:
```css
h2 {
  letter-spacing: 2px;
}
```

~~~
This will add 2 pixels of space between characters in all `<h2>` elements.

5. `line-height`: Sets the height of each line of text. It can be specified as a number, a unitless value, or a percentage. It affects the spacing between lines, creating more or less vertical space. For example:
```css
p {
 line-height: 1.5;
}
~~~
This will set the line height of all `<p>` elements to 1.5 times the font size.

These text formatting properties provide control over the appearance and layout of text within elements, allowing for customized typography and visual styling.

III) Explanation of CSS Border Properties:

CSS provides several border properties that allow you to style and customize the borders of elements:

1. `border-width`: Sets the width of the border. It can be specified individually for each side (`border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width`) or using the shorthand `border-width`. You can use different units like pixels (`px`), ems (`em`), or percentages (`%`).

2. `border-style`: Specifies the style of the border, such as solid, dotted, dashed, double, groove, ridge, inset, outset, and more. It can be set individually for each side (`border-top-style`, `border-right-style`, `border-bottom-style`, `border-left-style`) or using the shorthand `border-style`.

3. `border-color`: Sets the color of the border. It can be specified individually for each side (`border-top-color`, `border-right-color`, `border-bottom-color`, `border-left-color`) or using the shorthand `border-color`. Colors can be specified using named colors, hexadecimal codes, RGB, or HSL values.

4. `border`: A shorthand property that combines `border-width`, `border-style`, and `border-color` in a single declaration. For example:
```css
div {
  border: 1px solid red;
}
```
This will set a 1-pixel wide solid red border for all `<div>` elements.

5. `border-radius`: Specifies the radius of the border corners, creating rounded corners. It can be set individually for each corner (`border-top-left-radius`, `border-top-right-radius`, `border-bottom-right-radius`, `border-bottom-left-radius`) or using the shorthand `border-radius`.

6. `border-image`: Allows you to use an image as the border. It combines multiple properties to define the border image source, slice, width, outset, and repeat.

Q 11) Answer the following in brief.
I) Explain the text shadow property with an example.
II) What is the key difference between Grid Layout and Flexbox?
Ans.) I) Explanation of the Text Shadow Property:

The `text-shadow` property in CSS allows you to add a shadow effect to text. It adds a shadow behind the text, creating a visual effect that can enhance readability or add decorative elements. The syntax for the `text-shadow` property is as follows:

```css
text-shadow: horizontal-offset vertical-offset blur-radius color;
```

- `horizontal-offset`: Specifies the horizontal position of the shadow relative to the text. Positive values move the shadow to the right, while negative values move it to the left.
- `vertical-offset`: Specifies the vertical position of the shadow relative to the text. Positive values move the shadow down, while negative values move it up.
- `blur-radius`: Sets the blurring effect of the shadow. A higher value creates a more blurred shadow, while a lower value creates a sharper shadow.
- `color`: Defines the color of the shadow. You can use named colors, hexadecimal codes, RGB or HSL values.

Here's an example of using the `text-shadow` property:

```css
h1 {
  text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
}
```

II) The Key Difference between Grid Layout and Flexbox:

Grid Layout and Flexbox are both CSS layout models that provide different approaches to creating flexible and responsive designs. The key difference between the two is their layout orientation and how they handle the arrangement of elements.

1. Orientation:
   - Grid Layout: Grid Layout is a two-dimensional layout model, allowing for the creation of complex grid structures with rows and columns. It enables precise control over both the horizontal and vertical dimensions of the layout.
   - Flexbox: Flexbox is a one-dimensional layout model that operates along a single axis (either horizontal or vertical). It focuses on distributing and aligning elements in a flexible manner along this axis.

2. Layout Structure:
   - Grid Layout: Grid Layout is based on a grid system, where elements are placed into cells within rows and columns. It provides explicit control over the placement and sizing of elements within the grid structure. Grid Layout is ideal for creating grid-like layouts with complex arrangements.
   - Flexbox: Flexbox works with a flex container and flex items. Elements inside a flex container can be flex items. Flexbox automatically adjusts the size and position of flex items to fit the container. It is best suited for creating dynamic and responsive single-row or single-column layouts.

3. Alignment:
   - Grid Layout: Grid Layout offers precise alignment capabilities for both rows and columns. You can align items horizontally and vertically within their grid cells.
   - Flexbox: Flexbox provides flexible alignment options along the main axis and cross axis. You can align items horizontally or vertically, control their order, and distribute remaining space between them.

Q 12) Answer the following in detail.
I) Explain fixed and static position property with an example.
II) Explain CSS padding property with an example.
III) Explain CSS list properties.
Ans.) I) Explanation of Fixed and Static Position Property:

1. Fixed Position:
The `position: fixed` property in CSS positions an element relative to the viewport, meaning it remains in a fixed position even when the page is scrolled. The element is positioned based on the values of `top`, `right`, `bottom`, and `left` properties.

Example:
```css
div {
  position: fixed;
  top: 20px;
  right: 20px;
}
```
In this example, the `<div>` element will be positioned 20 pixels from the top and 20 pixels from the right of the viewport. It will stay fixed in this position even when the user scrolls the page.

2. Static Position:
The `position: static` property is the default positioning for HTML elements. It means the element is positioned in the normal flow of the document. Elements with `position: static` are not affected by the `top`, `right`, `bottom`, or `left` properties.

Example:
```css
p {
  position: static;
}
```
In this example, all `<p>` elements will have a static position and follow the normal flow of the document. They will not be affected by positioning properties.

II) Explanation of CSS Padding Property:

The CSS `padding` property is used to set the space between the content of an element and its borders. It controls the internal spacing within an element and can be applied to all four sides (`padding-top`, `padding-right`, `padding-bottom`, `padding-left`) or using the shorthand `padding`.

Example:
```css
div {
  padding: 10px;
}
```
In this example, all sides of the `<div>` element will have a padding of 10 pixels. The content inside the `<div>` will be pushed away from its borders by 10 pixels.

You can also specify different padding values for individual sides using the longhand properties:
```css
div {
  padding-top: 20px;
  padding-right: 10px;
  padding-bottom: 30px;
  padding-left: 15px;
}
```

III) Explanation of CSS List Properties:
CSS provides several properties for styling and customizing lists. These properties apply to `<ul>` (unordered lists), `<ol>` (ordered lists), and `<li>` (list items) elements:

1. `list-style-type`: Sets the style of the list marker or bullet point. It accepts values like `none`, `disc`, `circle`, `square`, `decimal`, `lower-alpha`, `upper-alpha`, and more.

2. `list-style-image`: Specifies an image to be used as the list marker. It replaces the default marker defined by `list-style-type`. You can specify the URL of the image or use the `none` value to remove the marker.

3. `list-style-position`: Determines the position of the list marker in relation to the content. It accepts `inside` (marker inside the list item) or `outside` (marker outside the list item). When set to `outside`, the marker will be aligned with the left edge of the list.

4. `list-style`: A shorthand property that combines `list-style-type`, `list-style

-image`, and `list-style-position` in a single declaration. For example:
```css
ul {
  list-style: square inside;
}
```
This will set the list marker style to square, and the marker will be placed inside the list item.

Q 13) Answer the following in brief.
I) Describe the float property with an example.
II) Define responsive web design.
Ans.) I) Float Property:

The `float` property in CSS is used to position an element horizontally within its parent container and allow other elements to wrap around it. When an element is floated, it is taken out of the normal document flow, and other elements flow around it.

Example:
```css
img {
  float: left;
  margin-right: 10px;
}
```

II) Responsive Web Design:
Responsive web design is an approach to web development aimed at creating websites that provide an optimal viewing experience across different devices and

screen sizes. With responsive design, the layout and content of a website can dynamically adjust and adapt to fit the screen of the device being used, whether it's a desktop computer, laptop, tablet, or smartphone.

The key principles of responsive web design include:

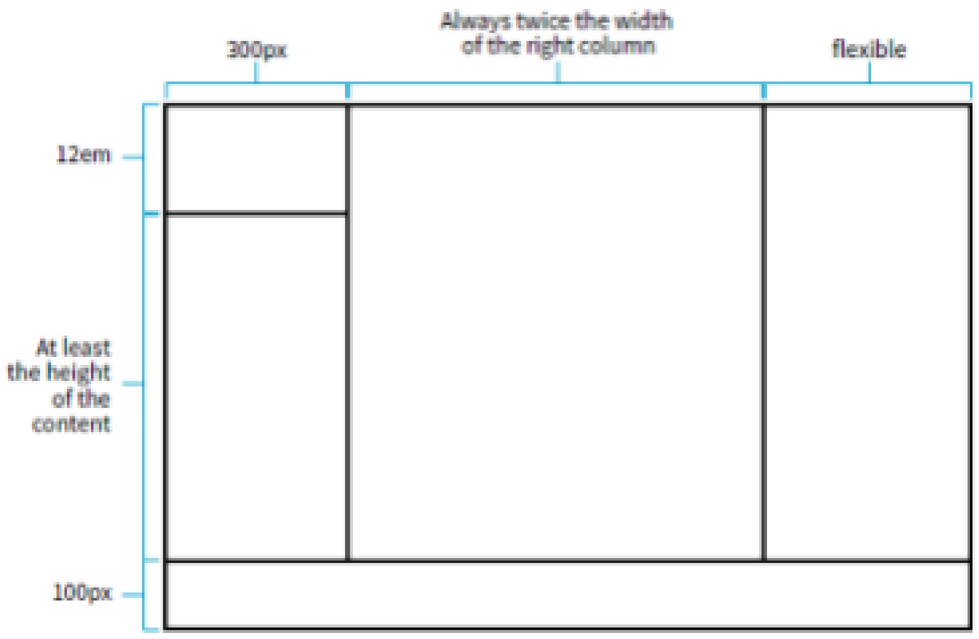
1. Fluid Grids: Using relative units and flexible grid systems to create layouts that can adapt to different screen sizes.

2. Flexible Images: Ensuring images are scalable and can adjust their size based on the available screen space.

3. Media Queries: Employing CSS media queries to apply different styles based on the characteristics of the device, such as screen width, resolution, or orientation.

4. Breakpoints: Defining specific points in the design where the layout needs to adapt to accommodate different screen sizes.

Q 14) Answer the following in detail.
I) Explain CSS background properties.
II) Explain external CSS with an example.
III) Create the grid template CSS for the layout shown in below figure.


300px
Always twice the width of the right column
flexible
12em
At least the height of the content
100px

Ans.) I) Explanation of CSS Background Properties:

CSS provides several background properties that allow you to style and customize the background of elements. These properties include:

1. `background-color`: Sets the background color of an element. You can use named colors, hexadecimal codes, RGB or HSL values.

2. `background-image`: Specifies an image to be used as the background. You can specify the URL of the image or use the `none` value to remove the background image.

3. `background-repeat`: Defines how the background image should be repeated if it doesn't cover the entire element. Values can be `repeat` (default), `repeat-x`, `repeat-y`, or `no-repeat`.

4. `background-position`: Sets the starting position of the background image within the element. You can use keywords like `top`, `bottom`, `left`, `right`, or specify pixel values or percentages.

5. `background-size`: Controls the size of the background image. Values can be `auto` (default), `cover` (image scales to cover the entire element), `contain` (image scales to fit within the element), or specific dimensions like `200px 300px`.

6. `background-attachment`: Specifies whether the background image scrolls with the content or remains fixed in place. Values can be `scroll` (default), `fixed`, or `local`.

These background properties can be combined and used together to create visually appealing backgrounds for elements. For example:
```css
div {
  background-color: #f2f2f2;
  background-image: url('background-image.jpg');
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  background-attachment: fixed;
}
```
In this example, the `<div>` element will have a light gray background color (`#f2f2f2`), an image (`background-image.jpg`) set as the background, centered and covering the element with no repeating. The background image will remain fixed as the content is scrolled.

II) Explanation of External CSS:

External CSS refers to the practice of storing CSS rules in a separate CSS file and linking that file to an HTML document. It allows you to keep the styles separate from the HTML structure, making it easier to manage and update the styles across multiple web pages.

Here's an example of using external CSS:

1. Create a new CSS file with a `.css` extension, such as `styles.css`.

2. In the CSS file, write the CSS rules that you want to apply to your HTML elements. For example:
```css
/* styles.css */
h1 {
  color: blue;
  font-size: 24px;
}
```
3. Save the CSS file and place it in the same directory as your HTML file.
4. In your HTML file, within the `<head>` section, add a link tag to connect the HTML file with the external CSS file. For example:
```html
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Welcome to my website</h1>
</body>
</html>
```
In this example, the CSS rules defined in `styles.css` will be applied to the `<h1>` element in `index.html`. The `href` attribute in the link tag specifies the path to the external CSS file.

Using external CSS has several advantages, including:

- Separation of concerns: It allows separation of HTML structure and CSS styles, making it easier to manage and update the styles.
- Reusability: Styles defined in an external CSS file can be used across multiple web pages, ensuring consistency in the design.
- Efficiency: The browser can cache the external CSS file, reducing the overall page load time for subsequent visits.