Unit 2: Cascading Style Sheet

How CSS fit with HTML page?

CSS stands for Cascading Style Sheets. CSS stands for Cascading Style Sheets, and is used to style HTML documents. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. CSS allows you to define styles for HTML elements, such as font styles, colors, margins, padding, and more. The syntax for CSS is as follows:

Types of CSS

CSS can be added to an HTML document in three different ways:

a. External CSS:

In this method, CSS is stored in a separate file with a .css extension, and then linked to the HTML document using the <link> element in the head section of the HTML document.

```
<!DOCTYPE html>
<html>
 <head>
 <link rel="stylesheet" type="text/css" href="style.css">
 </head>
 <body>
 <!-- HTML content here -->
 </body>
</html>.
```

b. Internal CSS:

In this method, CSS is defined within the HTML document using the <style> element in the head section of the HTML document.

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 /* CSS rules here */
 </style>
 </head>
 <body>
 <!-- HTML content here -->
 </body>
</html>.
```

c. Inline CSS:

In this method, CSS is defined within the HTML element using the style attribute.

```
<!DOCTYPE html>
<html>
 <head>
 </head>
 <body>
 <h1 style="color: red; font-size: 24px;">Hello World</h1>
 </body>
</html>.
```

CSS Selectors

CSS selectors are used to select and style HTML elements. There are many types of CSS selectors, including:

a. Element Selector:

The element selector is used to select all elements of a particular type. For example:

```
 p {
 color: red;
}
```

This selector selects all <p> elements and sets their color to red.

b. Class Selector:

The class selector is used to select all elements with a specific class. For example:

```
.my-class {
 font-size: 16px;
}
```

This selector selects all elements with class="my-class" and sets their font size to 16 Pixels.

c. Attribute Selector:

The attribute selector is used to select elements based on their attributes. For example:

```
a[target="_blank"] {
 color: blue;
}
```

This selector selects all <a> elements with target="_blank" and sets their color to blue.

d. Descendant Selector:

The descendant selector is used to select elements that are descendants of another element. For example:

```
ul li {
 list-style-type: square;
```

}

This selector selects all <li> elements that are descendants of <ul> elements and sets their list style type to square.

e. Child Selector:

The child selector is used to select elements that are direct children of another element. For example:

```
ul > li {
 font-weight: bold;
}
```

This selector selects all <li> elements that are direct children of <ul> elements and sets their font weight to bold.

f. Universal Selector:

The universal selector is used to select all elements on the page. For example:

```
* {
 margin: 0;
 padding: 0;
}
```

This selector selects all elements on the page and sets their margin and padding to zero. These are just a few of the many CSS selectors that are available. By using different types of selectors, you can create highly targeted styles that are applied only to specific elements on your page.

CSS Properties for text, list, table, background, link formatting

a. **CSS Properties for text:**
b. **CSS Properties for list**
c. **CSS Properties for Table**
d. **CSS Properties for Background**
e. **CSS Properties for link formatting**

Pseudo-classes

CSS pseudo-classes are used to select and style elements based on their state or position within the document. They allow you to create styles that are applied only under certain conditions, such as when an element is being hovered over, focused, or clicked. Here are some commonly used CSS pseudo-classes:

```
 <!DOCTYPE html>
<html>
<head>
<title>CSS Pseudo-classes Example</title>
<style>
button {
background-color: #4CAF50;
```

```
color: white;
padding: 12px 20px;
border: none;
border-radius: 4px;
cursor: pointer;
transition: background-color 0.3s ease;
}
button:hover {
background-color: #3e8e41;
}
button:active {
background-color: #1e5c17;
}
button:focus {
background-color: #b3ff99;
}
</style>
</head>
<body>
<button>Click me or tab to me!</button>
</body>
</html>.
```

Custom list numbering using content property

In CSS, you can use the content property along with the counter and counter-increment properties to create custom list numbering. Here's an example:

HTML

```
<ol class="custom-list">
 <li>Item one</li>
 <li>Item two</li>
 <li>Item three</li>
</ol>
```

CSS

```
.custom-list {
```

```
 counter-reset: my-counter;
 list-style-type: none;
}
.custom-list li:before {
 counter-increment: my-counter;
 content: counter(my-counter) ". ";}
```

In this example, we first reset the my-counter counter to 0 using the counter-reset property. We then remove the default list styling by setting the list-style-type property to none. Next, we use the :before pseudo-class to add the counter to each list item. We increment the my-counter counter by 1 for each list item using the counter-increment property. We then use the content property to display the counter followed by a period and a space.

You can customize the numbering by changing the counter name, the starting value of the counter, the content of the :before pseudo-element, and the counter increment value to suit your needs.

CSS Box Model: margin, padding and border

The CSS box model is a fundamental concept in web design that defines how content, padding, border, and margin are applied to HTML elements. The box model is used to create the layout of web pages, including spacing and positioning of content on a page. Here are the three components of the box model

g. Content: The content is the actual content of an HTML element, such as text, images, or videos.

h. Padding: The padding is the space between the content and the element's border. Padding can be set using the padding property in CSS.

i. Border: The border is the line that surrounds the element's padding and content. Borders can be set using the border property in CSS.

j. Margin: The margin is the space between the element's border and the neighboring elements. Margins can be set using the margin property in CSS. Here's an example of how these components work together:

Creating Layouts with display, position and float property

CSS offers several properties that can be used to create different layouts on a web page, including display, position, and float.

Display property:

The display property controls how an element is displayed on the page. The most commonly used values for the display property are: block: Elements with this display value take up the full width of their parent container and start on a new line. Examples include and <p> elements. inline: Elements with this display value only take up as much width as their content and don't start on a new line. Examples include and <a> elements. inline-block: Elements with this display value take up only as much width as their content but can have padding and margin applied to them. Examples include <button> and <input> element.

Position property

The position property controls the positioning of an element on the page. The most commonly used values for the position property are:

static: This is the default value and means that an element will be positioned according to the normal flow of the document.
relative: This value allows you to position an element relative to its normal position in the document flow. You can use the top, bottom, left, and right properties to adjust the element's position.
absolute: This value allows you to position an element relative to its closest positioned ancestor. If there is no positioned ancestor, the element will be positioned relative to the document itself.
fixed: This value positions an element relative to the viewport, meaning that it will stay in the same place even as the user scrolls the page

Float property

The float property is used to align an element to the left or right of its container, allowing other content to wrap around it. When an element is floated, its container will collapse to the height of its content, so you may need to use a clearfix or other technique to prevent layout issues. Here's an example that demonstrates the use of these properties:

Fixed and Liquid design of the page

Fixed and liquid (also known as fluid) design are two common approaches to creating layouts for web pages.Fixed design refers to a layout where the width of the content is fixed and does not change with the width of the viewport or device screen. This means that the layout looks the same on any screen size, regardless of whether the screen is large or small. Fixed design layouts are often used for websites with a lot of content, such as news websites or blogs. Here's an example of a fixed design:

```
body {
 width: 960px;
 margin: 0 auto;
}
```

In this example, we've set the width of the body to 960 pixels and centered it on the page using the margin property. This means that the layout will always be 960 pixels wide, regardless of the size of the screen or viewport. Liquid design, on the other hand, refers to a layout where the width of the content changes with the width of the viewport or device screen. This means that the layout adjusts to fit the
screen, making it a more responsive design. Liquid design layouts are often used for websites with a lot of images or multimedia content. Here's an example of a liquid design:

```
body {
 width: 90%;
 max-width: 1200px;
 margin: 0 auto;
}
```

In this example, we've set the width of the body to 90% of the viewport width, with a maximum width of 1200 pixels. This means that the layout will adjust to fit the screen, but won't get too wide on larger screens. The margin property is used to center the layout on the page. Both fixed and liquid designs have their own advantages and disadvantages. Fixed design can be easier to create and maintain, but it may not be suitable for all devices and screen sizes. Liquid design can be more flexible and responsive, but it can be more challenging to create and may require more testing. Ultimately, the choice of which design approach to use depends on the specific needs and goals of the website.