CSS Summary & Cheatsheet

Agenda:

What is CSS?

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a document written in HTML or XML (including various XML languages like SVG, MathML or XHTML). It separates the presentation of the document from the content, enabling developers to apply a consistent look and feel to a website, ensuring that the visual aspects of a web page are separate from the underlying structure and content.

CSS works by defining rules for how specific HTML elements should be displayed. These rules include properties like color, font, size, positioning, and more. CSS rules can be defined within an HTML document, in a separate CSS file, or within a <style> tag in the <head> section of an HTML document.

CSS uses a selector to specify which elements on a web page the style should be applied to. The selector can be an HTML tag, an ID, a class, or a combination of these. Once the selector is specified, properties can be defined using property-value pairs separated by a semicolon.

CSS supports a wide range of selectors and properties, giving web developers the ability to customize the look and feel of their websites in virtually endless ways. It also supports advanced layout techniques like flexbox and grid, making it easier to create complex page layouts.

CSS is a cornerstone technology of the web along with HTML and JavaScript, and it is widely used by web developers to create visually engaging and interactive websites.

CSS Syntax

The syntax of CSS (Cascading Style Sheets) consists of three main components: selector, property, and value.

The general format is as follows:

```
selector {
  property: value;
  property: value;
  ...
}
```

Selector: It defines the HTML element to which the style is to be applied. The selector can be an HTML tag, class, or ID.

Property: It defines the specific style to be applied to the selected HTML element, such as color, font-size, background-color, etc.

Value: It defines the value for the property being applied to the selected HTML element.

Here is an example of CSS syntax:

```
h1 {
  color: blue;
  font-size: 24px;
  text-align: center;
}
```

In this example, the selector is the h1 tag, and the properties being applied are color, font-size, and text-align. The values for those properties are blue, 24px, and center, respectively.

CSS also allows for multiple selectors to be grouped together and styled with the same properties. Here is an example:

```
h1, h2, h3 {
  font-family: Arial, sans-serif;
  font-weight: bold;
  color: #333333;
}
```

In this example, the h1, h2, and h3 selectors are grouped together, and the properties being applied are font-family, font-weight, and color.

CSS also supports commenting by using /* */ like this:

```
/* This is a CSS comment */
```

Overall, the syntax of CSS is quite straightforward and easy to understand once you become familiar with its three main components: selector, property, and value.

CSS Styling

To use CSS in HTML, there are several ways:

1. **Inline CSS:** Inline CSS is used to style a particular HTML element. It is specified using the style attribute within the opening tag of the HTML element. For example:

```
<p style="color: blue; font-size: 20px;">This is a paragraph.</p>
```

2. **Internal CSS:** Internal CSS is used to apply styles to a whole HTML page. It is specified using the <style> element within the <head> section of the HTML document. For example:

```
<head>
  <style>
    p {
      color: blue;
      font-size: 20px;
```

```
    }
  </style>
</head>
<body>
  <p>This is a paragraph.</p>
</body>
```

3. **External CSS:** External CSS is used to separate the presentation of an HTML document from its content. It is specified in a separate .css file and linked to the HTML document using the <link> element within the <head> section of the HTML document. For example:

styles.css file:

```
p {
  color: blue;
  font-size: 20px;
}
```

HTML file:

```
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <p>This is a paragraph.</p>
</body>
```

CSS Colors

CSS provides a range of options for defining colors. Colors can be defined using named colors, hexadecimal codes, RGB values, and HSL values.

Named Colors: There are 147 named colors available in CSS that can be used to define colors. For example, you can use the color name "red" to set the color of a text or background.

Hexadecimal Colors: In CSS, hexadecimal colors are represented by a hash (#) symbol followed by a six-digit code representing the red, green, and blue (RGB) values. Each pair of digits represents the intensity of the color for each of the RGB components. For example, #FF0000 represents pure red.

RGB Colors: RGB colors are defined using the "rgb()" function, which takes three parameters: the red, green, and blue values. Each value can range from 0 to 255. For example, rgb(255, 0, 0) represents pure red.

HSL Colors: HSL (Hue, Saturation, and Lightness) colors are defined using the "hsl()" function. Hue is represented as an angle from 0 to 360 degrees on the color wheel, while saturation and lightness are represented as percentages. For example, hsl(0, 100%, 50%) represents pure red.

To use these color options in CSS, you can specify the color property in the CSS selector and set it to the desired color value. For example, to set the color of a text to red using named color, you can use the following CSS:

```
h1 {
  color: red;
}
```

To set the background color of a div using hexadecimal color, you can use the following CSS:

```
div {
  background-color: #FF0000;
}
```

To set the color of a link using RGB color, you can use the following CSS:

```
a {
  color: rgb(255, 0, 0);
}
```

To set the color of a text using HSL color, you can use the following CSS:

```
p {
  color: hsl(0, 100%, 50%);
}
```

In addition to these options, CSS also provides color functions and color opacity options that can be used to create complex color effects as the fourth value.

CSS Backgrounds

In CSS, background refers to the area behind the content of an element. It can be set using various properties to define the color, image, and other effects of the background.

CSS Background Color:

The background-color property sets the color of the background. It can be set using various formats such as color names, RGB values, HEX values, HSL values, and RGBA values.

Example:

To set the background color of an element to blue, you can use the following CSS code:

```
background-color: blue;
```

CSS Background Image:

The background-image property sets the image to be used as the background. The image can be a URL or a local file.

Example:

To set an image as the background of an element, you can use the following CSS code:

```
background-image: url('example.jpg');
```

CSS Background Repeat:

The background-repeat property sets whether the background image should be repeated or not. It can have the values "repeat", "repeat-x", "repeat-y", and "no-repeat".

Example:

To set the background image to repeat only horizontally, you can use the following CSS code:

```
background-repeat: repeat-x;
```

CSS Background Position:

The background-position property sets the position of the background image. It can be set using keywords or length values.

Example:

To set the background image to be positioned at the top left corner of the element, you can use the following CSS code:

```
background-position: top left;
```

CSS Background Attachment:

The background-attachment property sets whether the background image should scroll with the element or remain fixed.

Example:

To set the background image to remain fixed while scrolling the element, you can use the following CSS code:

```
background-attachment: fixed;
```

CSS Background Size:

The background-size property sets the size of the background image. It can be set using length values, percentages, or keywords.

Example:

To set the background image size to cover the entire element, you can use the following CSS code:

```
background-size: cover;
```

CSS Gradient Background:

The gradient background is a CSS property that allows you to create a smooth transition between two or more colors.

Example:

To set the background to a gradient that transitions from blue to green, you can use the following CSS code:

```
background: linear-gradient(to bottom, blue, green);
```

CSS Background shorthand:

The background shorthand property allows you to set multiple background properties in one line.

Example:

To set the background color, image, repeat, position, attachment, and size in one line, you can use the following CSS code:

```
background: #ffffff url('example.jpg') no-repeat top left fixed cover;
```

CSS Borders

CSS borders are used to define the border of an HTML element. Borders are used to separate the content of an element from its surrounding elements, and they can add style and visual appeal to a web page. Borders can be applied to any HTML element, including divs, paragraphs, tables, images, and more.

The CSS border property is used to set the width, style, and color of a border. It has three components: border-width, border-style, and border-color. The following is an example of how to set a border:

```
border: 1px solid black;
```

This sets a 1-pixel wide border that is solid and black. Here's a breakdown of each component:

border-width: This sets the width of the border. It can be a value in pixels, ems, or other length units.
border-style: This sets the style of the border. It can be solid, dashed, dotted, double, groove, ridge, inset, outset, or none.
border-color: This sets the color of the border. It can be any valid color value, such as a hex code, RGB value, or named color.
In addition to the border property, there are several other CSS properties that can be used to control the appearance of borders, including border-radius (to create rounded corners), border-image (to use an image for the border), and border-collapse (to control the spacing between table borders).

Here's an example of how to create a border with rounded corners:

```
border-radius: 10px;
```

This creates a border with 10-pixel rounded corners.

CSS borders can also be applied to specific sides of an element using the following properties: border-top, border-right, border-bottom, and border-left. These properties work the same way as the border property but only apply to the specified side.

```
border-top: 2px solid red;
border-right: 1px dashed blue;
border-bottom: 3px dotted green;
border-left: 4px double yellow;
```

This sets different border styles and colors for each side of the element.

CSS Margins

In CSS, the margin property defines the space outside of an element's border. It is used to create space between elements or between an element and its container. The margin property can have up to four values, each of which sets the margin for a different side of the element. The syntax for the margin property is as follows:

margin: value1 value2 value3 value4;

Here, value1 sets the top margin, value2 sets the right margin, value3 sets the bottom margin, and value4 sets the left margin. Each value can be specified in different units such as pixels, ems, rems, percentages, etc. Alternatively, the margin property can be set individually for each side using the following properties:

margin-top: sets the margin for the top of an element
margin-right: sets the margin for the right of an element
margin-bottom: sets the margin for the bottom of an element
margin-left: sets the margin for the left of an element
Here is an example of using the margin property to create space around a div element:

```
<!DOCTYPE html>
<html>
<head>
	<style>
		div {
			background-color: lightblue;
			margin: 20px;
			padding: 10px;
		}
	</style>
</head>
<body>
	<div>This is a div element with 20px margin all around.</div>
</body>
</html>
```

In this example, the margin property is set to 20px for all sides of the div element, creating space around it. The padding property is also used to add space inside the element.

CSS Padding

In CSS, padding is a property that specifies the space between an element's content and its border. Padding can be applied to any HTML element, and it can be set using different units of measurement such as pixels, ems, and percentages.

The syntax for the padding property is as follows:

```
selector {
  padding: value;
}
```

The value can be a single value that applies to all four sides of the element, or it can be a set of values that apply to different sides of the element, specified in the order top, right, bottom, and left.

For example, the following code sets the padding of a div element to 10 pixels on all four sides:

```
div {
  padding: 10px;
}
```

Alternatively, you can specify different padding values for different sides, as shown in the following code:

```
div {
  padding-top: 5px;
  padding-right: 10px;
  padding-bottom: 5px;
  padding-left: 10px;
}
```

In addition to the basic padding properties, CSS also provides shorthand properties that allow you to set multiple properties at once. The shorthand property for padding is padding, and it accepts up to four values, in the same order as the individual properties:

```
div {
  padding: 5px 10px 5px 10px;
```

```
}
```

In the example above, the top and bottom padding is set to 5 pixels, and the right and left padding is set to 10 pixels.

Padding can be used to create space between an element and its content, or to create visual separation between different elements on a page. It can also be used in combination with other CSS properties, such as borders and backgrounds, to create more complex visual effects.

CSS Width/Height

In CSS, you can set the width and height of an element using the width and height properties respectively. These properties define the dimensions of the content area of the element, excluding any padding, borders, and margins.

The width property sets the width of the content area, while the height property sets the height of the content area.

There are several units that you can use to specify the width and height of an element:

Pixels (px): This is the most commonly used unit and represents a fixed size in pixels. For example, width: 200px; will set the width of an element to 200 pixels.

Percentage (%): This unit represents a relative size based on the size of the parent element. For example, width: 50%; will set the width of an element to 50% of the width of its parent element.

Viewport units (vw and vh): These units represent a relative size based on the size of the viewport. For example, width: 50vw; will set the width of an element to 50% of the viewport width.

em and rem: These units represent a relative size based on the font size of the element or its parent element. For example, width: 2em; will set the width of an element to twice the font size of the element.

Here are some examples of using the width and height properties:

```
/* Set the width and height of an element using pixels */
div {
  width: 200px;
  height: 100px;
}
```

```
/* Set the width of an element to 50% of its parent element */
div {
```

```
  width: 50%;
}
```

```
/* Set the width of an element to 50% of its parent element */
div {
  width: 50%;
}
```

```
/* Set the width of an element to 50% of the viewport width */
div {
  width: 50vw;
}
```

```
/* Set the width of an element to twice the font size of the element */
div {
  font-size: 16px;
  width: 2em;
}
```

It's important to note that setting the width and height of an element can affect its layout and the layout of other elements on the page. It's generally a good practice to use responsive design techniques to ensure that your layout adjusts to different screen sizes and devices.

CSS Box Model

The CSS Box Model is a concept that explains how elements are laid out on a web page. Each HTML element is considered as a rectangular box with its content, padding, border, and margin areas. The box model consists of four components:

Content: It is the area that contains the actual content of the element, such as text, images, or videos.

Padding: It is the area between the content and the border. Padding can be used to add extra space within an element.

Border: It is a line that surrounds the content and padding areas. The border can be used to add a decorative element to the element.

Margin: It is the area outside the border that separates the element from other elements on the page. Margin can be used to add space between elements.

The following CSS properties can be used to control the box model:

width: It sets the width of the content area.

height: It sets the height of the content area.

padding: It sets the padding area around the content.

border: It sets the border around the element.

margin: It sets the margin area around the element.

For example, to create a box with a width of 200px, a height of 100px, a padding of 10px, a border of 2px, and a margin of 20px, the following CSS code can be used:

```
.box {
  width: 200px;
  height: 100px;
  padding: 10px;
  border: 2px solid black;
  margin: 20px;
}
```

This will create a rectangular box with a total width of 244px (200px + 22px border + 220px margin) and a total height of 144px (100px + 22px border + 220px margin).

CSS Text Styling

CSS Text Properties are used to style the text content in HTML. These properties allow you to control various aspects of text such as font, size, color, alignment, decoration, spacing, etc. Here are some of the commonly used CSS text properties:

font-family: This property specifies the font family of the text. You can specify multiple font families separated by commas. The browser will use the first font family that is available on the user's computer.

Example:

```
p {
   font-family: Arial, sans-serif;
}
font-size: This property sets the font size of the text.
```

```

```

Example:

```
p {
  font-size: 16px;
}
```

font-style: This property sets the font style of the text to italic or normal.

Example:

```
p {
  font-style: italic;
}
```

font-weight: This property sets the font weight of the text to bold or normal.

Example:

```
p {
  font-weight: bold;
}
```

color: This property sets the color of the text.
Example:

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

text-align: This property sets the horizontal alignment of the text.

Example:

```
p {
  text-align: center;
}
```

text-decoration: This property sets the text decoration such as underline, overline, or line-through.

Example:

```
p {
  text-decoration: underline;
}
```

letter-spacing: This property sets the spacing between the characters in the text.

Example:

```
p {
  letter-spacing: 2px;
}
```

word-spacing: This property sets the spacing between the words in the text.

Example:

```
p {
  word-spacing: 5px;
}
```

line-height: This property sets the height of each line of text.

Example:

```
p {
  line-height: 1.5;
}
```

CSS Fonts

CSS fonts are used to style the text in HTML documents. They allow you to set the font family, size, weight, style, and other properties of text. Here are some of the most common CSS font properties:

font-family: This property is used to set the font of an element. You can specify multiple font families, separated by commas, in case the first font is not available.

Example:

```
p {
  font-family: Arial, sans-serif;
}
```

font-size: This property is used to set the size of the font. You can specify the size in pixels, ems, or other units.

Example:

```
p {
  font-size: 16px;
}
```

font-weight: This property is used to set the weight of the font. You can use values like "normal", "bold", or numbers like 400, 700, etc.

Example:

```
p {
  font-weight: bold;
}
```

font-style: This property is used to set the style of the font. You can use values like "normal", "italic", "oblique", etc.

Example:

```
p {
  font-style: italic;
}
```

font-variant: This property is used to set the variant of the font. You can use values like "normal" or "small-caps".

Example:

```
p {
  font-variant: small-caps;
}
```

CSS Lists

CSS provides several properties to style lists. Some of the commonly used properties for styling lists are:

list-style-type: This property sets the type of bullet or numbering style for the list. Possible values include disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, and none.

Example:

```
ul {
```

```
  list-style-type: square;
}
```

list-style-position: This property sets the position of the bullet or numbering style for the list. Possible values include inside and outside.

Example:

```
ul {
  list-style-position: inside;
}
```

list-style-image: This property sets the URL of an image to be used as the bullet or numbering style for the list.

Example:

```
ul {
  list-style-image: url("bullet.png");
}
```

list-style: This is a shorthand property that sets all the above three properties (list-style-type, list-style-position, and list-style-image) in one declaration.

Example:

```
ul {
  list-style: square inside url("bullet.png");
}
```

text-indent: This property sets the indentation of the first line of the list item text.

Example:

```
li {
  text-indent: 20px;
}
```

padding-left: This property sets the padding on the left side of the list item, which can be used to move the bullet or numbering style away from the text.

Example:

```
li {
```

```
  padding-left: 30px;
}
```

CSS Tables

CSS provides a wide range of styling options for HTML tables. Here are some examples of CSS table styling:

Borders: To add borders to the table and its elements, you can use the "border" property.

For example:

```
table {
  border: 1px solid black;
}

th, td {
  border: 1px solid black;
}
```

Backgrounds: To set a background color or image for the table, you can use the "background-color" and "background-image" properties.

For example:

```
table {
  background-color: #f2f2f2;
}

th {
  background-color: #4CAF50;
  color: white;
}
```

Padding and spacing: To add space between the table elements, you can use the "padding" and "border-spacing" properties.

For example:

```
table {
  border-spacing: 5px;
}

th, td {
```

```
  padding: 10px;
}
```

Text alignment: To align the text inside table elements, you can use the "text-align" property.

For example:

```
th {
  text-align: left;
}

td {
  text-align: center;
}
```

Font styling: To style the text inside table elements, you can use the usual font properties such as "font-family", "font-size", "font-weight", etc.

For example:

```
th {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 18px;
  font-weight: bold;
}
```

CSS Display

The CSS display property is used to set how an HTML element should be displayed on a web page. The possible values for the display property are:

none: The element will not be displayed.
block: The element will be displayed as a block-level element, with a line break before and after the element.
inline: The element will be displayed as an inline-level element, with no line break before or after the element.
inline-block: The element will be displayed as an inline-level element, but can have a width and height set.
flex: The element will be displayed as a flex container.
grid: The element will be displayed as a grid container.
table: The element will be displayed as a table.
inline-table: The element will be displayed as an inline-level table.
list-item: The element will be displayed as a list item.
inherit: The element will inherit the display value from its parent element.

initial: The element will have the initial display value.
unset: The element will have the display value of its parent element if it is set, otherwise it will have the initial display value.

The display property can be used to create different layouts for web pages. For example, by setting an element's display property to flex, you can create a flexible layout with elements that can be easily repositioned and resized.

Here's an example of using the display property to create a simple flexbox layout:

HTML:

```
<div class="container">
  <div class="box">Box 1</div>
  <div class="box">Box 2</div>
  <div class="box">Box 3</div>
</div>
```

CSS:

```
.container {
  display: flex;
  flex-wrap: wrap;
}

.box {
  flex: 1 0 200px;
  height: 100px;
  background-color: #ccc;
  margin: 10px;
}
```

In this example, the container element is set to display: flex, which makes its child elements behave as flex items. The flex-wrap: wrap property ensures that the flex items will wrap to a new row when they reach the edge of the container. Each box element has a flex property that tells the browser how to distribute available space among the flex items. In this case, each box should take up equal space, but can shrink or grow as needed within a minimum and maximum width of 200px. Finally, the margin property adds some space between the boxes.

CSS Max-Width

The max-width property in CSS sets the maximum width of an element. If the element's content is wider than the max-width value, the element will shrink to fit the content. If the content is narrower than the max-width value, the element will expand to fill the available space.

Syntax:

```
selector {
  max-width: value;
}
```

Example:

```
.container {
  max-width: 800px;
}
```

In the above example, the .container element will have a maximum width of 800 pixels.

We can also use percentage values for max-width:

```
.container {
  max-width: 80%;
}
```

In this example, the .container element will have a maximum width of 80% of its parent element's width.

max-width is commonly used in responsive design to ensure that elements don't become too wide on larger screens, and to prevent horizontal scrolling on smaller screens.

CSS Z-Index

CSS z-index is a property that specifies the stacking order of an element along the z-axis (i.e., the depth axis). It is used to control which elements appear in front of or behind other elements. Elements with a higher z-index value will be displayed in front of elements with a lower z-index value.

The z-index property can be applied to any positioned element, which includes elements with position: relative, position: absolute, position: fixed, and position: sticky.

The z-index value can be either a positive or negative integer, or the keyword auto. The default z-index value is auto, which means that the element will be stacked in the order in which it appears in the HTML document.

When using z-index, it is important to remember that it only applies to elements that have a position value other than static. If an element has position: static, the z-index property will have no effect.

Here is an example of how to use the z-index property:

```
<style>
   .box {
      position: relative;
      width: 100px;
      height: 100px;
      background-color: blue;
      z-index: 1;
   }

   .box2 {
      position: absolute;
      top: 50px;
      left: 50px;
      width: 100px;
      height: 100px;
      background-color: red;
      z-index: 2;
   }
</style>

<div class="box"></div>
<div class="box2"></div>
```

In this example, the box2 element will appear in front of the box element because it has a higher z-index value. The box element has a z-index value of 1, while the box2 element has a z-index value of 2.

CSS Position

CSS position is a property that specifies the positioning of an element in relation to its parent or to the viewport. The position property can take on several values, including static, relative, absolute, fixed, and sticky.

Static: It is the default value. It means that the element is positioned according to the normal flow of the document. This property doesn't take any additional values.

Example:

```
<div class="container">
  <p>This is a paragraph inside a container.</p>
</div>
```

```
/* CSS */
```

```
.container {
  width: 300px;
  height: 150px;
  background-color: lightblue;
}

p {
  color: white;
}
```

Relative: It is positioned relative to its normal position. This means that an element with position: relative; will move 10 pixels to the right and 20 pixels down from its normal position.

Example:

```
<div class="container">
  <p>This is a paragraph inside a container.</p>
</div>
```

```
/* CSS */
.container {
  position: relative;
  width: 300px;
  height: 150px;
  background-color: lightblue;
}

p {
  position: relative;
  left: 10px;
  top: 20px;
  color: white;
}
```

Absolute: The element is positioned relative to its nearest positioned ancestor. If there is no positioned ancestor, it will be positioned relative to the initial containing block (usually the body element).

Example:

```
<div class="container">
  <p>This is a paragraph inside a container.</p>
  <img src="image.png" alt="image" />
</div>
```

```
/* CSS */
.container {
  position: relative;
  width: 300px;
  height: 150px;
  background-color: lightblue;
}

p {
  color: white;
}

img {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

Fixed: The element is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.

Example:

```
<div class="fixed">
  <p>This is a fixed paragraph.</p>
</div>
```

```
/* CSS */
.fixed {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  background-color: lightblue;
  color: white;
  padding: 10px;
}
```

Sticky: The element is positioned based on the user's scroll position. A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

Example:

```
<div class="sticky">
  <p>This is a sticky paragraph.</p>
</div>
```

```
/* CSS */
.sticky {
  position: sticky;
  top: 0;
  background-color: lightblue;
  color: white;
  padding: 10px;
}
```

CSS Overflow

CSS overflow property is used to control the behavior of the content that overflows the boundaries of an element. It determines whether to clip the content or to add scrollbars to view the content.

The overflow property can have the following values:

visible: Default value. Content is not clipped and may overflow the element.
hidden: The overflow content will be hidden and not displayed.
scroll: Adds scrollbars to view the content that overflows.
auto: Adds scrollbars only if necessary, when the content overflows.

Example:

```
div {
  width: 200px;
  height: 100px;
  overflow: scroll;
}
```

In the above example, the div element has a fixed height and width of 200px and 100px respectively. If the content inside the div overflows the boundaries, the scroll value for the overflow property will add scrollbars to view the overflowed content.

CSS Float

CSS float is a property that allows an element to float to the left or right of its container. When an element is floated, it is taken out of the normal document flow and positioned along the left or right edge of its

container. Other elements can then flow around the floated element. The float property is commonly used for creating multi-column layouts and for positioning images and other content.

The syntax for using float in CSS is as follows:

```
selector {
  float: left | right | none | inherit;
}
```

The float property can be set to one of the following values:

left: The element floats to the left of its container.
right: The element floats to the right of its container.
none: The element is not floated and remains in the normal document flow.
inherit: The element inherits the float value of its parent element.
It is important to note that floated elements are taken out of the normal document flow, which can have implications for the layout of other elements on the page. To prevent this, it is often necessary to use clear-fix techniques or to add a clearfix class to the container element to ensure that it expands to include the floated elements.

Here is an example of how to use the float property to create a multi-column layout:

```
<div class="container">
  <div class="column-left">Left column</div>
  <div class="column-right">Right column</div>
</div>
```

```
.container {
  overflow: hidden; /* Prevents container from collapsing */
}

.column-left {
  float: left;
  width: 50%;
}

.column-right {
  float: right;
  width: 50%;
}
```

In this example, the two columns are floated to the left and right of the container, respectively, with a width of 50% each. The overflow: hidden property is used to prevent the container from collapsing due to the floated elements.

CSS Opacity

CSS opacity is a property that specifies the transparency level of an element. The value of the opacity property ranges from 0 to 1, where 0 indicates the element is completely transparent, and 1 indicates the element is completely opaque.

The syntax for setting the opacity property is as follows:

```
selector {
  opacity: value;
}
```

Here, the selector is the element to which the opacity property is being applied, and value is the transparency level specified in decimal format.

For example, let's say we have an HTML element with the class name box. We can apply an opacity level of 0.5 to this element using the following CSS:

```
.box {
  opacity: 0.5;
}
```

This will make the box element half-transparent.

It's important to note that the opacity property affects not only the element itself, but also its child elements. If you want to make only the background of an element transparent, while keeping the text or other content opaque, you can use the background-color property along with the rgba() color function. The rgba() function allows you to specify a color with an alpha channel, which determines the transparency level.

For example, let's say we want to make the background of a div element transparent, while keeping the text opaque. We can use the following CSS:

```
div {
  background-color: rgba(255, 255, 255, 0.5);
}
```

This will set the background color of the div element to white with an opacity of 0.5, making it half-transparent.

CSS Dropdowns

CSS drop-down menus are a type of menu that allows users to select options from a list that appears when they hover or click on a menu item. They are commonly used in website navigation menus.

To create a CSS drop-down menu, you can use the following steps:

Create an unordered list for the menu items.
Set the CSS display property of the list items to "inline-block" or "block".
Set the CSS position property of the nested list to "absolute".
Set the CSS visibility property of the nested list to "hidden".
Add a hover or focus event to the parent menu item to change the visibility property of the nested list to "visible".
Here is an example of a simple CSS drop-down menu:

HTML:

```
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">Products</a>
      <ul>
        <li><a href="#">Product 1</a></li>
        <li><a href="#">Product 2</a></li>
        <li><a href="#">Product 3</a></li>
      </ul>
    </li>
    <li><a href="#">Services</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>
```

CSS:

```
nav ul {
  list-style: none;
  margin: 0;
  padding: 0;
  background-color: #333;
}
```

```
nav li {
  position: relative;
  display: inline-block;
}

nav ul ul {
  position: absolute;
  visibility: hidden;
}

nav li:hover > ul {
  visibility: visible;
}

nav a {
  display: block;
  padding: 0 10px;
  color: #fff;
  font-size: 16px;
  line-height: 60px;
  text-decoration: none;
}

nav ul ul {
  top: 60px;
}

nav ul ul li {
  display: block;
  float: none;
  background-color: #444;
  padding: 0;
}

nav ul ul a {
  line-height: 40px;
}
```

In this example, the top-level menu items are displayed inline-block and have a hover event that changes the visibility property of the nested unordered list to visible. The nested unordered list is positioned absolutely and has a hidden visibility property. The nested list items are displayed as blocks and have a background color. The menu items have a background color and padding to make them more visually appealing.

CSS Image

In CSS, images can be styled and manipulated in a variety of ways. Here are some common CSS properties used to style images:

width and height: These properties can be used to set the dimensions of an image.

Example:

```
img {
  width: 200px;
  height: 150px;
}
```

object-fit: This property is used to control how an image fits into its container. Possible values are "fill", "contain", "cover", "none", "scale-down".

Example:

```
img {
  width: 100%;
  height: 200px;
  object-fit: cover;
}
```

border: This property can be used to add a border around an image.

Example:

```
img {
  border: 2px solid black;
}
```

box-shadow: This property can be used to add a shadow effect to an image.

Example:

```
img {
  box-shadow: 2px 2px 2px rgba(0,0,0,0.5);
}
```

filter: This property can be used to apply various visual effects to an image such as blur, brightness, contrast, hue-rotate, etc.

Example:

```
img {
  filter: brightness(150%);
}
```

transition: This property can be used to add a smooth transition effect when an image is hovered over.
Example:

```
img {
  transition: transform 0.5s ease;
}
img:hover {
  transform: scale(1.2);
}
```

CSS Specificity

CSS specificity is the mechanism used by CSS to determine which styles should be applied to an element when multiple selectors target the same element. It is based on the specificity value assigned to each selector.

Specificity values are calculated based on the number and types of selectors used in a CSS rule. A selector can have four types of selectors, in order of increasing specificity:

Type selector (e.g. div, p, a)
Class selector (e.g. .my-class)
ID selector (e.g. #my-id)
Inline style (e.g. style="color: red;")
To calculate the specificity value of a selector, we assign a weight to each type of selector, with ID selectors having the highest weight (100), followed by class selectors (10), and type selectors (1). Inline styles have a specificity value of 1000. Then, we add up the weights of all selectors in a rule to get the specificity value.

For example, the selector div .my-class#my-id has a specificity value of 111, calculated as follows:

Type selector: 1 (div)
Class selector: 10 (.my-class)
ID selector: 100 (#my-id)
Here are some examples to illustrate how CSS specificity works:

Example 1:

HTML:

```
<div class="my-class">Hello World!</div>
```

CSS:

```
.my-class {
  color: red;
}
```

In this case, the color property will be applied to the div element because the class selector has a specificity value of 10.

Example 2:

HTML:

```
<div id="my-id">Hello World!</div>
```

CSS:

```
#my-id {
  color: blue;
}
.my-class {
  color: red;
}
```

In this case, the color property will be applied to the div element because the ID selector has a specificity value of 100, which is higher than the class selector's value of 10.

Example 3:

HTML:

```
<div style="color: green;">Hello World!</div>
```

CSS:

```
.my-class {
  color: red;
}
```

In this case, the color property will be applied to the div element because inline styles have a specificity value of 1000, which is higher than the class selector's value of 10.

CSS Selectors

CSS selectors are patterns used to select and style HTML elements on a web page. They allow developers to target specific elements or groups of elements and apply styles such as color, font, size, and layout.

There are five main categories of CSS selectors:

1. Simple selectors: Select elements based on their name, ID, class, or attributes.
2. Combinator selectors: Select elements based on their relationship to other elements.
3. Pseudo-class selectors: Select elements based on their state, such as when a user hovers over an element.
4. Pseudo-element selectors: Select and style a specific part of an element, such as the first letter or line of text.
5. Attribute selectors: Select elements based on the presence or value of their attributes.

Simple Selectos

CSS Simple Selectors are the most basic type of selectors in CSS, used to select elements based on their name, ID, or class. There are three main types of CSS Simple Selectors:

Element Selectors: An element selector selects all elements with a specific HTML tag name. It is defined by the name of the tag. For example, the following selector selects all <p> elements:

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

ID Selectors: An ID selector selects an element based on the value of its id attribute. It is defined by a hash (#) followed by the ID value. For example, the following selector selects the element with id="myDiv":

```
#myDiv {
  background-color: yellow;
}
```

Class Selectors: A class selector selects all elements with a specific class attribute value. It is defined by a dot (.) followed by the class name. For example, the following selector selects all elements with class="myClass":

```
.myClass {
  font-size: 16px;
}
```

You can also use a combination of these selectors to make your CSS more specific. For example, the following selector selects all <p> elements with class="myClass":

```
p.myClass {
  font-weight: bold;
}
```

The universal selector is represented by the * symbol and it matches any element. It can be used alone or in combination with other selectors to target all elements in a document or a specific set of elements.

For example, the following CSS rule targets all elements in the document:

```
* {
  color: red;
}
```

And this CSS rule targets all div elements:

```
div * {
  color: blue;
}
```

In both cases, the universal selector is used to select all elements or all elements that are descendants of div elements.

Combinator Selectors

A CSS combinator selector is used to select elements based on their relationship with other elements. There are four subcategories of combinator selectors:

1. **Descendant Selector:** This selector is used to select elements that are descendants of another element. It is represented by a space ( ) between two selectors. For example, if you want to select all the paragraph elements that are descendants of a div element, you can use the following selector:

```
div p {
  /* Styles */
}
```

2. **Child Selector:** This selector is used to select elements that are direct children of another element. It is represented by a greater than symbol (>) between two selectors. For example, if you want to select all the paragraph elements that are direct children of a div element, you can use the following selector:

```
div > p {
  /* Styles */
}
```

3. **Adjacent Sibling Selector:** This selector is used to select an element that immediately follows another element. It is represented by a plus sign (+) between two selectors. For example, if you want to select all the h2 elements that immediately follow an h1 element, you can use the following selector:

```
h1 + h2 {
  /* Styles */
}
```

4. **General Sibling Selector:** This selector is used to select elements that follow another element, regardless of their position in the hierarchy. It is represented by a tilde (~) between two selectors. For example, if you want to select all the li elements that follow an h1 element, you can use the following selector:

```
h1 ~ li {
  /* Styles */
}
```

In all of the above examples, the first selector (div, h1, etc.) is called the "reference selector" and the second selector (p, h2, li, etc.) is called the "dependent selector". The dependent selector is only selected if it satisfies the relationship criteria specified by the combinator selector.

Pseudo class selectors

Sure, here are all the 31 pseudo-class selectors in CSS with explanations and examples:

:active - Selects an element that is currently being clicked or activated by the user.

Example:

```
button:active {
  background-color: yellow;
}
```

:hover - Selects an element when the user hovers over it with the mouse pointer.

Example:

```
a:hover {
  color: red;
}
```

:focus - Selects an element when it has focus (usually as a result of being tabbed to).

Example:

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

:visited - Selects a link that has been visited by the user.

Example:

```
a:visited {
  color: purple;
}
```

:link - Selects a link that has not been visited by the user.

Example:

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

:first-child - Selects the first child element of its parent.

Example:

```
li:first-child {
  font-weight: bold;
}
```

:last-child - Selects the last child element of its parent.

Example:

```
li:last-child {
  font-style: italic;
}
```

:nth-child(n) - Selects the nth child element of its parent, where n is a number or an expression.

Example:

```
li:nth-child(2) {
  background-color: yellow;
}
```

:nth-last-child(n) - Selects the nth child element of its parent, counting from the last child.

Example:

```
li:nth-last-child(2) {
  color: green;
}
```

:only-child - Selects an element that is the only child of its parent.

Example:

```
span:only-child {
  text-decoration: underline;
}
```

:empty - Selects an element that has no child elements or text.

Example:

```
p:empty {
  display: none;
}
```

:target - Selects an element that is the target of the current page URL.

Example:

```
#section1:target {
  background-color: yellow;
}
```

:enabled - Selects a form element that is enabled.

Example:

```
input:enabled {
  background-color: white;
}
```

:disabled - Selects a form element that is disabled.

Example:

```
input:disabled {
  background-color: gray;
}
```

:checked - Selects a checkbox or radio button that is checked.

Example:

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

:not(selector) - Selects all elements that do not match the given selector.

Example:

```
:not(p) {
  background-color: gray;
}
```

:first-of-type - Selects the first element of its type within its parent.

Example:

```
p:first-of-type {
  font-weight: bold;
}
```

:last-of-type - Selects the last element of its type within its parent.

Example:

```
p:last-of-type {
  font-style: italic;
}
```

Pseudo element selectors

CSS Pseudo-Element Selectors allow you to style parts of an element rather than the whole element. They start with a double colon (::) followed by the name of the pseudo-element.

Here are some examples of CSS Pseudo-Element Selectors and their uses:

::before: This pseudo-element adds content before the selected element. For example, you can use it to add a decorative icon before a link.

```
a::before {
  content: "🔗";
}
```

This will add a link icon before every link.

::after: This pseudo-element adds content after the selected element. For example, you can use it to add a decorative icon after a button.

```
button::after {
  content: "→";
}
```

This will add a right arrow after every button.

::first-line: This pseudo-element selects the first line of text within an element. For example, you can use it to change the font size and color of the first line of a paragraph.

```
p::first-line {
  font-size: 24px;
  color: red;
}
```

This will change the font size and color of the first line of every paragraph.

::first-letter: This pseudo-element selects the first letter of text within an element. For example, you can use it to change the font size and color of the first letter of a heading.

```
h1::first-letter {
```

```
  font-size: 48px;
  color: blue;
}
```

This will change the font size and color of the first letter of every heading.

::selection: This pseudo-element selects the portion of text that is highlighted by the user. For example, you can use it to change the background color and color of highlighted text.

```
::selection {
  background-color: yellow;
  color: black;
}
```

This will change the background color and color of highlighted text to yellow and black, respectively.

::marker is a valid pseudo-element selector in CSS. It is used to select the marker box of a list item, which contains the bullet or number for unordered and ordered lists, respectively. Here's an example:

```
ul li::marker {
  color: red;
}
```

This will set the color of the marker box for each list item in an unordered list to red. Note that ::marker is only supported in Firefox and is not a part of the current CSS specification. Instead, you can use the ::before or ::after pseudo-elements to achieve similar effects in a more cross-browser compatible way.

Attribute selectors

Attribute selectors in CSS allow you to select HTML elements based on the presence or value of their attributes.

Attribute Exists Selector ([attr]) - Selects all elements that have the specified attribute, regardless of its value.

```
/* Select all anchor tags with a href attribute */
a[href] {
  color: blue;
}
```

Attribute Equals Selector ([attr=value]) - Selects all elements that have the specified attribute with a value exactly equal to the specified value.

```
/* Select all input tags with type="submit" */
input[type="submit"] {
  background-color: green;
}
```

Attribute Contains Selector ([attr*=value]) - Selects all elements that have the specified attribute with a value containing the specified substring.

```
/* Select all input tags with a name attribute containing "email" */
input[name*="email"] {
  border: 1px solid blue;
}
```

Attribute Starts With Selector ([attr^=value]) - Selects all elements that have the specified attribute with a value beginning exactly with the specified value.

```
/* Select all input tags with a type attribute starting with "text" */
input[type^="text"] {
  background-color: yellow;
}
```

Attribute Ends With Selector ([attr$=value]) - Selects all elements that have the specified attribute with a value ending exactly with the specified value.

```
/* Select all anchor tags with a href attribute ending in ".pdf" */
a[href$=".pdf"] {
  color: red;
}
```

Attribute Contains Word Selector ([attr~=value]) - Selects all elements that have the specified attribute with a value containing a specific word.

```
/* Select all paragraph tags with a class attribute containing the word "important" */
p[class~="important"] {
  font-weight: bold;
}
```

Attribute Not Equal Selector ([attr!=value]) - Selects all elements that have the specified attribute with a value not equal to the specified value.

```
/* Select all image tags with an alt attribute not equal to "logo" */
img[alt!="logo"] {
  border: 1px solid black;
}
```