1. CSS Syntax and Selectors:

- CSS (Cascading Style Sheets) is a style sheet language used for describing the look and formatting of a document written in HTML. It consists of rules that define how elements should be displayed on a web page.
- CSS selectors are used to target specific HTML elements and apply styles to them. Selectors can be based on element names, classes, IDs, attributes, and more.

Example:

```
/* Selects all <h1> elements */
h1 {
  color: blue;
}

/* Selects elements with class "highlight" */
.highlight {
  background-color: yellow;
}

/* Selects elements with ID "logo" */
#logo {
  width: 200px;
}
```

Use Cases:

- Changing the color and font of text.
- Setting background images or colors.
- Applying margins, padding, and borders.
- Creating layout structures.

2. CSS Rules and Declarations:

- CSS rules are composed of selectors and declarations. Selectors target specific elements, and declarations define the styles to be applied.
- Declarations consist of a property and a value. The property determines what aspect of the element to style, and the value specifies how to style it.

Example:

```
/* Selector: <h1> elements */
 /* Declaration: color property set to red */
 h1 {
   color: red;
 }

 /* Selector: elements with class "highlight" */
 /* Declaration: background-color property set to yellow */
 .highlight {
   background-color: yellow;
 }
```

Use Cases:

- Applying different colors to different headings.
- Setting the background color of specific sections.
- Changing the font size and style of paragraphs.

3. Applying CSS Styles to HTML Elements:

- CSS styles can be applied to HTML elements using selectors and declarations.
- Selectors target specific elements, and declarations define the styles to be applied.

Example:

```
<h1>This is a heading</h1>
<p class="highlight">This paragraph has a highlight class.</p>
```

```
h1 {
    color: blue;
  }

  .highlight {
    background-color: yellow;
  }
```

Use Cases:

- Styling headings, paragraphs, links, buttons, etc.
- Creating consistent styling across multiple pages.

4. Using Inline, Internal, and External Stylesheets:

- CSS styles can be applied in three ways: inline, internal, and external.
- Inline styles are defined within the HTML element using the `style` attribute.
- Internal style sheets are defined within the `<style>` tags in the `<head>` section of an HTML document.
- External stylesheets are separate CSS files linked to the HTML document using the `<link>` tag.

Example:

```
<!-- Inline style -->
<h1 style="color: blue;">This is a heading</h1>

<!-- Internal stylesheet -->
<style>
  h1 {
    color: blue;
  }
</style>

<!-- External stylesheet -->
<link rel="stylesheet" href="styles.css">
```

Use Cases:

- Inline styles for quick, one-off styling.
- Internal stylesheets for small-scale styling within a single HTML document.
- External stylesheets for reusable styles across multiple HTML pages.

5. CSS Comments:

- CSS comments allow you to add notes or explanations within your CSS code. Comments are ignored by the browser and are only meant for developers to read.

Example:

```
/* This is a comment */
 h1 {
   color: blue;
 }
```

Use Cases:

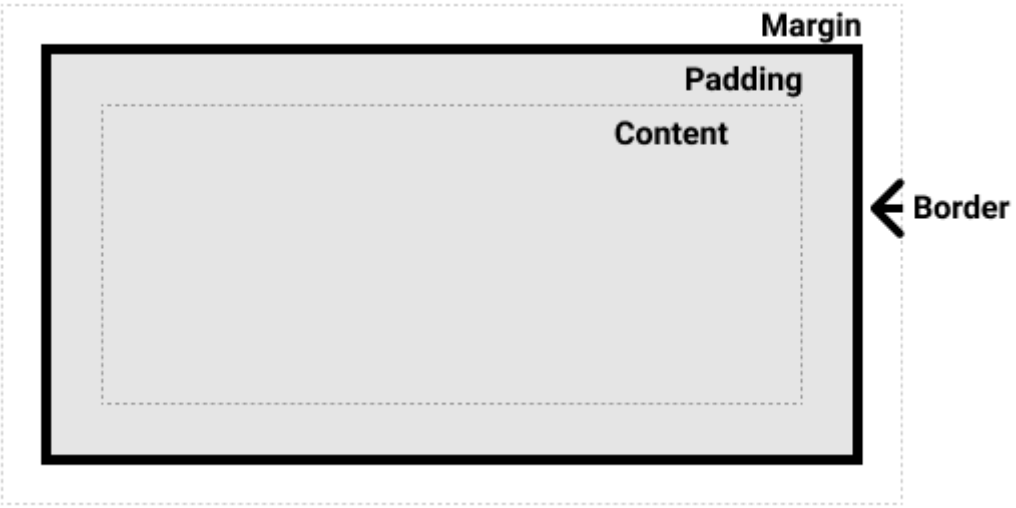
- Documenting and explaining CSS code for future reference.
- Temporarily disabling specific styles without deleting them.

6. Understanding the Box Model:

- The box model is a fundamental concept in CSS that defines how elements are rendered and how their dimensions are calculated.
- It consists of four parts: content, padding, border, and margin. The total space an element occupies is calculated by adding these four parts together.

Example:

```
 .box {
   width: 200px;
   height: 100px;
   padding: 10px;
   border: 1px solid black;
   margin: 20px;
 }
```



Use Cases:

- Creating space around elements.
- Defining element sizes and positions.
- Applying borders and padding.

7. Controlling Element Dimensions (Width, Height):

- CSS allows you to control the dimensions of elements using the `width` and `height` properties.
- These properties specify the width and height of an element's content area, excluding padding, border, and margin.

Example:

```
.box {
  width: 200px;
  height: 100px;
}
```

Use Cases:

- Specifying the size of images, buttons, and containers.
- Creating fixed or responsive layouts.

8. Box Sizing (Content-box vs. Border-box):

- The `box-sizing` property controls how the total width and height of an element are calculated.
- The default value is `content-box`, which includes only the content dimensions.
- The `border-box` value includes padding and border in the total dimensions, making it easier to create predictable layouts.

Example:

```
.box {
  box-sizing: border-box;
}
```

Use Cases:

- Simplifying the sizing of elements when using padding and borders.
- Ensuring consistent dimensions across different elements.

9. Margin Collapsing:

- Margin collapsing is a behaviour where adjacent vertical margins collapse into a single margin.
- This happens when the top and bottom margins of sibling elements touch or overlap.

Example:

```
.element1 {
  margin-bottom: 20px;
}

.element2 {
  margin-top: 30px;
}

/* The margin collapse results in a 30px margin between element1 and element2 */
```

Use Cases:

- Creating spacing between elements without adding extra margin.

10. Adding Borders and Padding:

- Borders and padding can be added to elements using the `border` and `padding` properties, respectively.
- The `border` property specifies the border width, style, and color, while the `padding` property sets the space between the content and the border.

Example:

```
.box {
  border: 1px solid black;
  padding: 10px;
}
```

Use Cases:

- Adding visual separation between elements.
- Creating space around content.

11. Positioning Elements (Static, Relative, Absolute, Fixed):

- CSS provides several positioning options for elements.
- `static` (default) positions an element according to the normal flow of the document.
- `relative` positions an element relative to its normal position.
- `absolute` positions an element relative to its nearest positioned ancestor.
- `fixed` positions an element relative to the viewport, so it remains fixed even when scrolling.

Example:

```
.box {
  position: relative;
  top: 20px;
  left: 50px;
}

.popup {
  position: absolute;
```

```
  top: 0;
  right: 0;
}

.header {
  position: fixed;
  top: 0;
  width: 100%;
  background-color: #f1f1f1;
}
```

Use Cases:

- Creating complex layouts.
- Positioning elements precisely.

12. Floating Elements:

- Floating an element allows it to float to the left or right of its container, allowing text and other elements to wrap around it.
- Floated elements are taken out of the normal flow of the document.

Example:

```
.image {
  float: left;
  margin-right: 10px;
}
```

Use Cases:

- Creating multicolumn layouts.
- Wrapping text around images.

13. Display Property (Block, Inline, Inline-block, Flex, Grid):

- The `display` property controls how an element is rendered on the page.
- `block` elements start on a new line and take up the full width available.
- `inline` elements flow within the text and do not start on a new line.
- `inline-block` elements behave like inline elements but can have a width, height, and margin.

- `flex` and `grid` enable powerful layout capabilities.

Example:

```
.block-element {
  display: block;
}

.inline-element {
  display: inline;
}

.inline-block-element {
  display: inline-block;
}

.flex-container {
  display: flex;
}

.grid-container {
  display: grid;
}
```

Use Cases:

- Creating column layouts.
- Aligning and distributing elements.

14. CSS Units of Measurement (Pixels, Percentages, Em, Rem):

- CSS supports various units of measurement to define sizes and positions.
- `px` (pixels) is a fixed unit.
- `%` (percentage) is relative to the parent element's size.
- `em` is relative to the font size of the element.
- `rem` is relative to the root (`html`) font size.

Example:

```css
.element {
  width: 200px;
  height: 50%;
  font-size: 1.2em;
  margin: 1rem;
}
```

Use Cases:

- Creating responsive layouts.
- Scaling elements based on parent or root sizes.

15. CSS Layout Techniques (Floats, Flexbox, Grid):

- CSS provides various layout techniques to position and arrange elements.
- Floats are used for simple layout and text wrapping.
- Flexbox is a one-dimensional layout model for creating flexible and responsive designs.
- Grid is a two-dimensional layout model for creating complex grid-based layouts.

Example:

```css
.float-layout {
  float: left;
}

.flexbox-container {
  display: flex;
  justify-content: center;
  align-items: center;
}

.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-gap: 10px;
}
```

Use Cases:

- Creating column-based layouts.
- Aligning and distributing elements.

16. Setting Font Family, Size, and Weight:

- CSS allows you to control the font family, size, and weight of text.
- The `font-family` property specifies the font to use.
- The `font-size` property sets the size of the text.
- The `font-weight` property defines the thickness of the text.

Example:

```
.text {
   font-family: Arial, sans-serif;
   font-size: 16px;
   font-weight: bold;
 }
```

Use Cases:

- Customizing the typography of a website.
- Ensuring consistent font styles across elements.

17. Text Alignment and Indentation:

- CSS provides properties to align and indent text within elements.
- The `text-align` property controls the horizontal alignment of text.
- The `text-indent` property sets the indentation of the first line of text.

Example:

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

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

Use Cases:

- Aligning text within headings, paragraphs, and navigation items.
- Indenting quotes or paragraphs.

18. Text Decoration (Underline, Overline, Line-through):

- CSS allows you to add decorations to text, such as underlines, overlines, and strikethroughs.
- The `text-decoration` property controls the decoration of text.

Example:

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

.overline {
  text-decoration: overline;
}

.line-through {
  text-decoration: line-through;
}
```

Use Cases:

- Adding emphasis to specific text.
- Indicating links or completed tasks.

19. Line Height and Spacing:

- The `line-height` property controls the height of lines within a block of text.
- The `letter-spacing` property adjusts the spacing between characters.
- The `word-spacing` property adjusts the spacing between words.

Example:

```
.text {
  line-height: 1.5;
  letter-spacing: 1px;
  word-spacing: 2px;
```

```
}
```

Use Cases:

- Improving readability and legibility of text.
- Adjusting spacing for specific design needs.

20. Transparency and Opacity:

- CSS allows you to control the transparency of elements.
- The `opacity` property sets the overall opacity of an element, affecting both its content and background.

Example:

```
.transparent-box {
  background-color: rgba(255, 0, 0, 0.5);
}

.opaque-box {
  opacity: 0.8;
}
```

Use Cases:

- Creating translucent backgrounds.
- Fading elements in and out.

21. Pseudo-classes and Pseudo-elements:

- Pseudo-classes and pseudo-elements target specific states or parts of elements.
- Pseudo-classes start with a colon (`:`) and target elements based on user interactions or states.
- Pseudo-elements start with a double colon (`::`) and target specific parts of elements.

Example:

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

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

Use Cases:

- Styling links on hover or focus.
- Formatting specific parts of text or elements.

22. Overriding Styles and Using !important:

- CSS styles can be overridden by more specific selectors or styles declared later in the code.
- The `!important` declaration can be added to a style to give it the highest priority and override other styles.

Example:

```
.box {
  background-color: blue !important;
}

.container .box {
  background-color: red;
}
```

Use Cases:

- Resolving conflicting styles.
- Ensuring specific styles are always applied.

23. Timing Functions (Linear, Ease, Ease-in, etc.):

- CSS animations and transitions can have timing functions that control how the animation progresses over time.
- Timing functions define the speed curve of an animation, specifying acceleration and deceleration.

Example:

```
.box {
  transition: width 1s ease-in-out;
}
```

Use Cases:

- Creating smooth and natural animations.
- Controlling the speed and timing of transitions.

24. CSS Animation Keyframes and Properties:

- CSS animations allow you to animate properties of elements over a specified duration.
- Keyframes define the intermediate steps of the animation.
- Animation properties control the animation's duration, delay, and timing.

Example:

```
@keyframes move {
  0% { transform: translateX(0); }
  100% { transform: translateX(100px); }
}

.box {
  animation: move 2s infinite alternate;
}
```

Use Cases:

- Adding engaging animations to elements.
- Creating interactive and visually appealing effects.

25. Delaying and Repeating Animations:

- CSS animations can be delayed and repeated using animation properties.
- The `animation-delay` property specifies the delay before an animation starts.
- The `animation-iteration-count` property controls how many times an animation should repeat.

Example:

```
.box {
  animation: move 2s infinite alternate;
  animation-delay: 1s;
  animation-iteration-count: 3;
}
```

Use Cases:

- Creating animations that start after a certain time.
- Repeating animations for specific durations.

26. Media Queries and Responsive Breakpoints:

- Media queries allow you to apply different styles based on the characteristics of the device or viewport.
- They can be used to create responsive designs that adapt to different screen sizes and devices.

Example:

```
@media screen and (max-width: 600px) {
  .container {
    flex-direction: column;
  }
}
```

Use Cases:

- Adapting layouts for mobile, tablet, and desktop screens.
- Modifying styles based on device capabilities.

Prepared By :
Divyansh Singh (LinkedIn : rgndunes)