CSS

****Introduction to CSS****

CSS (Cascading Style Sheets) is used to style and layout web pages.

****Ways to Add CSS****

****Inline CSS :-****Applied directly within an HTML element using the style attribute.

****Internal CSS :-****Placed within a <style> tag inside the <head> section of an HTML document.

****External CSS :-****Linked via a <link> tag to an external CSS file.

****Cascading Order :-****Order of priority: Inline > Internal > External.

****Comments in CSS****

****Single-line Comment :-****/\* This is a single-line comment \*/

****Multi-line Comment:-****/\* This is a multi-line comment

                                                   It can span multiple line \*/

****Useful VS Code Extensions****

* ****CSS Peek :-****Allows you to see the CSS code directly in the HTML file.
* ****Color Highlight :-****Highlights colors in your CSS files.
* ****Image Preview :-****Shows image previews when you hover over image URLs.
* ****HTML CSS Support :-****Provides CSS class and ID completion support in HTML files****.****
* ****HTML to CSS Autocompletion :-****Auto-completes CSS properties and values.
* ****Stylelint :-****Checks your CSS for errors and helps you follow best practices.
* ****Intellisense for CSS Class Names :-****Suggests class names that are already defined in your project.

****Simple Selectors****

****Tagname Selector  :-****Selects all elements of a specified tag.

****Example:****    p { }

****ID Selector :-****Selects a single element with a specific ID.

****Example:****#unique-id { }

****Class Selector :-****Selects all elements with a specific class.

****Example:****.class-name { }

****Universal Selector :- Selects all elements.****

****Example:**** \* { }

****Grouping Selector :-****Groups multiple selectors to apply the same styles.

****Example:****h1, h2, p { }

****Class Naming Convention****

****Introduction****

1. Class naming conventions help keep your code organized.
2. They make the code more readable.
3. Facilitate understanding of the code structure.
4. Enable easier code navigation and debugging.
5. Promote best practices in web development.

****Importance****

1. Consistent class names improve code maintainability.
2. Meaningful class names make it easier for others to understand the code.
3. Reduces the chance of conflicts and duplication.
4. Helps in better project management and scalability.
5. Enhances teamwork by providing a common coding language.

****Popular Naming Convention****

1. ****BEM (Block, Element, Modifier)**** is a structured naming convention.
2. ****Block:**** Represents a standalone entity, e.g., block.
3. ****Element:**** Represents a part of the block, e.g., block\_\_element.
4. ****Modifier:**** Represents a different state or version of a block or element, e.g., block--modifier.
5. ****Example:**** block\_\_element--modifier clearly defines relationships and states.

### ****Color in CSS****

****Color Name****

1. Use simple color names to set colors.
2. Easy to remember and use.
3. ****Common names include red, blue, green.****
4. ****Example: color: red;****
5. Limited predefined names available.

****Hex Color****

1. Uses hexadecimal values to define colors.
2. ****Consists of a # followed by six digits/letters.****
3. Offers precise color selection.
4. ****Example: color: #ff0000; (red)****
5. ****Can be shortened for some values, e.g., #fff for white.****

****RGB****

1. Stands for Red, Green, Blue.
2. Defines colors by mixing these three colors.
3. Values range from 0 to 255.
4. ****Example: color: rgb(255, 0, 0); (red)****
5. Useful for creating custom colors.

****RGBA****

1. Similar to RGB but includes an Alpha value.
2. Alpha controls the transparency (opacity) of the color.
3. Values range from 0 (fully transparent) to 1 (fully opaque).
4. ****Example: color: rgba(255, 0, 0, 0.5); (red with 50% opacity)****
5. Useful for overlay and blending effects.

****HSL****

1. Stands for Hue, Saturation, Lightness.
2. Hue is a degree on the color wheel (0-360).
3. Saturation is a percentage (0% is gray, 100% is full color).
4. Lightness is a percentage (0% is black, 100% is white).
5. ****Example: color: hsl(0, 100%, 50%); (red)****

****HSLA****

1. Similar to HSL but includes an Alpha value.
2. Alpha controls the transparency (opacity) of the color.
3. Values range from 0 (fully transparent) to 1 (fully opaque).
4. ****Example: color: hsla(0, 100%, 50%, 0.5); (red with 50% opacity)****
5. Useful for sophisticated color and transparency control.

****Opacity****

1. Sets the opacity level of an element.
2. Values range from 0 (fully transparent) to 1 (fully opaque).
3. Applies to the whole element, including background and borders.
4. ****Example: opacity: 0.5; (50% opacity)****
5. Useful for creating overlay effects and emphasizing content.

### ****Background in CSS****

****Background Color****

1. Sets the background color of an element.
2. Can use color names, hex values, RGB, RGBA, HSL, or HSLA.
3. ****Example: background-color: blue;****
4. Provides a solid color background.
5. Useful for enhancing readability and visual appeal.

****Background Image****

1. Sets an image as the background.
2. URL specifies the path to the image file.
3. ****Example: background-image: url('image.jpg');****
4. Can use local or external image sources.
5. Enhances design with visual elements.

****Background Repeat****

1. Controls if/how the background image repeats.
2. ****Options include repeat, repeat-x, repeat-y, no-repeat.****
3. ****Example: background-repeat: no-repeat;****
4. Prevents the image from repeating.
5. Useful for custom background patterns.

****Background Attachment****

1. Controls the scroll behavior of the background.
2. ****Options include scroll, fixed, local.****
3. ****Example: background-attachment: fixed;****
4. Keeps the background image fixed in place during scrolling.
5. Useful for creating parallax effects.

****Background Position****

1. Sets the starting position of the background image.
2. ****Common values include left, center, right, top, bottom.****
3. ****Example: background-position: center;****
4. Centers the background image.
5. ****Can use pixel or percentage values for precise positioning.****

****Background Shorthand****

1. Combines all background properties into one declaration.
2. Properties include color, image, repeat, attachment, position.
3. ****Example: background: #ff0000 url('image.jpg') no-repeat fixed center;****
4. Simplifies CSS code by consolidating multiple properties.
5. Useful for setting multiple background attributes quickly.

------------------------------------------------------------------------------------------------------------------------------------------------------

****Typography - Text Properties:****

1. ****Text Color:**** Defines the color of the text using color names, hexadecimal, RGB, or HSL values.
2. ****Text Decoration:****Adds decorations like underline, overline, line-through, etc., to the text.
3. ****Text Alignment:****Aligns text to left, right, center, or justify within its container.
4. ****Text Transformation:**** Transforms text to uppercase, lowercase, or capitalize.
5. ****Text Spacing:****Adjusts the spacing between characters using properties like letter-spacing and word-spacing.
6. ****Text Shadow: Adds shadows to the text for emphasis or style.****

****Typography - Font Properties:****

1. ****Font Size:**** Sets the size of the text.
2. ****Font Style:**** Sets the style of the font, like normal, italic, or oblique.
3. ****Font Shorthand:**** Combines font properties like size, style, weight, etc., into a single declaration.
4. ****Google Font:**** A collection of free fonts provided by Google for web use.
5. ****How to Use Google Font:**** Include a link to the font in your HTML file and specify the font-family in your CSS.

****List and Table Styling:****

1. ****List Styling:**** Defines the appearance of lists, including bullet style, image markers, etc.
2. ****Table Styling:****Defines the appearance of tables, including borders, spacing, alignment, etc.

****CSS Box Model:****

1. ****Introduction to Box Model:****Describes the structure of an HTML element, comprising content, padding, border, and margin.
2. ****Content:****The actual content of the element.
3. ****Padding:**** Clears an area around the content inside the border.
4. ****Border:****A border surrounding the padding.
5. ****Margin:****Clears an area outside the border.

****Outline:****

1. ****Outline and Its Properties:**** Similar to border but does not take up space or affect layout. It's often used for visual emphasis.
2. ****Difference Between Border and Outline:**** Borders are part of the box model, while outlines are drawn outside the border and don't affect layout.

****Width and Height:****

1. ****Width:****Sets the width of an element.
2. ****Min and Max Width:****Specifies the minimum and maximum width an element can have.
3. ****Height:****Sets the height of an element.
4. ****Min and Max Height: Specifies the minimum and maximum height an element can have.****
5. ****Box-Sizing:**** Defines how the total width and height of an element are calculated, including padding and border, if any.

****Introduction to CSS Unit:****

1. ****CSS Units and Their Types:**** Different units like pixels (px), percentages (%), em, rem, etc., used to define measurements in CSS.
2. ****Usage of Different CSS Units:**** Each unit serves different purposes, like absolute sizes (px), relative sizes (em, rem), and viewport-based sizes (vw, vh).

### ****CSS Specificity****

CSS specificity determines which CSS rule is applied when multiple rules target the same element.

### ****CSS !Important****

The !important declaration in CSS overrides normal specificity rules, ensuring that a specific style is applied no matter what.

### ****Introduction to !Important****

When you use !important in a CSS rule, it makes that rule the highest priority, so it will be applied even if other rules usually would take precedence.

### ****Drawbacks of Using !Important****

Using !important can make your code messy and hard to manage. It can be difficult to predict which styles will be applied, and it can cause conflicts, especially in large projects.

------------------------------------------------------------------------------------------------------------------------------------------------------

Lesson : Positions in CSS

****Introduction to CSS Layout****

****Normal Flow****

****Display and Visibility****

****Float and Clear****

* ****Float Layout****: Floats elements to the left or right within a container, allowing others to flow around them.
* Float Property:
  + float: left;
  + float: right;
* Clear Property: Controls how elements clear floated elements.
  + clear: left; (clears left-floated elements)
  + clear: right; (clears right-floated elements)
  + clear: both; (clears both left and right-floated elements)
  + clear: none; (no clearing)
  + clear: inherit; (inherits from parent element)

****CSS Positions (Part 1)****

* ****CSS Positioning****: Controls element layout on a webpage.
* ****Types:**** Static, Relative, Absolute, Fixed, Sticky.
* ****Benefits:**** Control layout, relative positioning, remove from flow, overlap elements, maintain position during scrolling.
* ****Position Properties:**** Top, Bottom, Left, Right.

****CSS Positions (Part 2)****

* Absolute vs Relative Positioning:
  + Relative: Moves element, doesn't break flow.
  + Absolute: Positioned relative to ancestor, breaks flow.
* Fixed Positioning:
  + Positioned relative to the viewport, stays fixed during scrolling.
* Sticky Positioning:
  + Initially behaves like a relative, then becomes fixed at the scroll point.
* Fixed vs Sticky:
  + Fixed: Position relative to the viewport, breaks flow.
  + Sticky: Initially behaves like a relative, then becomes fixed.

****Overflow and Z-Index****

* Overflow:
  + Determines behavior of content exceeding container boundaries.
  + Values include visible, hidden, scroll, and auto.
  + visible: Content overflows container.
  + hidden: Overflowing content is clipped.
  + scroll: Adds scrollbars.
  + auto: Scrollbars appear when needed.
* Index:
  + Determines the stacking order of elements along the z-axis.
  + Higher values place elements on top of lower ones.
  + Applied using the z-index property.

------------------------------------------------------------------------------------------------------------------------------------------------------

 Lesson : Flexbox in CSS

****Introduction to CSS Flexbox****

* Flexbox (Flexible Box):
  + Designed for UIs, aligns and distributes space among items.
  + Ensures consistent layout across screen sizes.
* Axes Understanding:
  + Main axis: Primary layout direction (horizontal or vertical).
  + Cross axis: Perpendicular to main axis, determines alignment.
* Usage:
  + Apply display: flex; to enable Flexbox.
  + Default: Horizontal main axis, vertical cross axis.
* Direction:
  + flex-direction: row;: Horizontal main axis.
  + flex-direction: column;: Vertical main axis.
* Cross Axis:
  + Determined by perpendicular axis to flex-direction.
  + Horizontal cross axis for flex-direction: row;, vertical for flex-direction: column;.

****Flex container properties****

* ****Align Items****
  + Aligns flex items along the cross-axis.
  + Possible values: stretch, center, flex-start, flex-end.
  + The default value is stretch.
* ****Align Content****
  + Aligns the flex lines within a flex container when there is extra space on the cross-axis.
  + Possible values: ****stretch****, ****center****, ****flex-start****, ****flex-end****, ****space-between****, ****space-around****.
  + Used when there are multiple lines of flex items.

* ****Justify Items****
  + Aligns individual items along the main axis within their alignment container.
  + Possible values: ****auto****, ****normal****, ****start****, ****end****, ****center****, ****stretch****.
  + Influences the distribution of space around content items.
* ****Justify Content****
  + Aligns the entire flex container's items along the main axis.
  + Possible values: ****flex-start****, ****flex-end****, ****center****, ****space-between****, ****space-around****, ****space-evenly****.
  + Determines how space is distributed around items.
* ****Gap****
  + Sets the spacing between items in a flex container.
  + Can be defined for rows (row-gap) and columns (column-gap), or both (gap).
  + Useful for creating consistent spacing without margins.

* ****Flex Wrap****
  + Controls whether flex items are forced into a single line or can wrap onto multiple lines.
  + Possible values: ****nowrap****, ****wrap****, ****wrap-reverse****.
  + The default value is ****nowrap****.
* ****Flex Flow****
  + Shorthand for setting both flex-direction and flex-wrap.
  + Values are space-separated: flex-direction flex-wrap.
  + Defines both the direction and wrapping behavior of flex items.
* ****Flex Direction****
  + Specifies the direction of the flex items in the flex container.
  + Possible values: row, row-reverse, column, column-reverse.
  + Determines the main axis direction (horizontal or vertical).

****Flex item :****

* A child element of a flex container.
* Inherits all the properties and behaviors of flexbox layout.
* Can be manipulated individually using flex properties (order, flex-grow, flex-shrink, flex-basis.).
* ****order****
  + Defines the item’s order within the container.
  + Default: 0.
* ****flex-grow****
  + Determines how much the item can grow.
  + Default: 0 (no growth).
* ****flex-shrink****
  + Determines how much the item can shrink.
  + Default: 1.
* ****flex-basis****
  + Sets the initial size of the item.
  + Default: auto.
* ****flex****
  + Shorthand for ****flex-grow****, ****flex-shrink****, ****flex-basis****.
  + Default: 0 1 auto.
* ****align-self****
  + Overrides ****align-items**** for an individual item.
  + Possible values: ****auto****, ****flex-start****, ****flex-end****, ****center****, ****baseline****, ****stretch****.

------------------------------------------------------------------------------------------------------------------------------------------------------

Lesson : Grid in CSS

****Introduction to Grid****

* ****Layout System****: CSS Grid is a 2-dimensional layout system for arranging elements in rows and columns.
* ****Activation****: Use display: grid; on a container to enable grid layout.
* ****Structure****: The grid consists of a grid container (parent) and grid items (children).
* ****Grid Lines****: Define column and row boundaries within the grid.
* ****Grid Properties:**** Includes grid-template-columns, grid-template-rows, grid-column, and grid-row for defining layout specifics.
* ****Advantages:**** Offers flexibility, simplicity in creating layouts, and precise control over element positioning.

****Grid container properties****

* ****Display****: Activates grid layout on an element using display: grid;.
* ****grid-template-columns:**** Defines the number and sizes of columns in the grid layout.
* ****grid-template-rows:**** Specifies the number and sizes of rows in the grid layout.
* ****grid-auto-rows:**** Sets the size of rows that are not explicitly defined in grid-template-rows.
* ****grid-auto-columns:**** Sets the size of columns that are not explicitly defined in grid-template-columns.
* ****grid-template-areas:**** Defines named grid areas where grid items can be placed.
* ****grid-column-gap:**** Specifies the size of gaps between columns in the grid layout.
* ****grid-row-gap:**** Specifies the size of gaps between rows in the grid layout.
* ****justify-content:**** Aligns grid items along the main axis (horizontal) of the grid container.
* ****align-content:**** Aligns grid lines along the cross axis (vertical) of the grid container when there is extra space.
* ****justify-items:**** Aligns grid items along the inline (horizontal) axis within their grid area.
* ****align-items:**** Aligns grid items along the block (vertical) axis within their grid area.

****Grid item properties****

* ****grid-column-start****and ****grid-column-end****: Specifies the starting and ending positions of a grid item along the columns.
* ****grid-column****: Shorthand for setting grid-column-start, grid-column-end, or both.
* ****grid-row-start**** and ****grid-row-end****: Specifies the starting and ending positions of a grid item along the rows.
* ****grid-row****: Shorthand for setting grid-row-start, grid-row-end, or both.
* ****grid-area****: Defines a grid item's size and location within the grid by referencing named grid areas.
* ****justify-self****: Aligns a grid item along the inline (horizontal) axis within its grid area.
* ****align-self****: Aligns a grid item along the block (vertical) axis within its grid area.
* ****order****: Controls the order in which grid items appear in the grid layout. Items with lower order values appear first.

------------------------------------------------------------------------------------------------------------------------------------------------------

 Lesson : Media Query

****Introduction to Media Queries & RWD****

* ****Responsive Web Design (RWD)****: Ensures websites adjust to various devices and screen sizes, enhancing user experience.
* ****Advantages****:
  + ****Improved User Experience****: Ensures consistency and usability across devices.
  + ****Increased Mobile Traffic****: Accommodates growing mobile device usage.
  + ****Cost and Time Efficiency****: Maintains a single codebase for all devices.
  + ****SEO Benefits****: Enhances visibility and search engine rankings.
* ****Implementation Methods****: Use flexible layouts, responsive images, and media queries for adaptive designs.
* ****Media Query****: CSS feature allowing conditional styling based on device attributes like screen size and orientation.
* ****Anatomy****: Uses @media rule, specifies media type (screen, print, etc.), features (width, height), and logical operators (and, or, not) for precise styling conditions.

****Writing media queries for different screen sizes****

* Purpose: Media queries in CSS enable targeted styling based on screen dimensions, ensuring websites are responsive across various devices.
* ****Standard Breakpoints:****
  + ****Extra small devices:**** Max-width of 576px.
  + ****Small devices:**** Min-width of 576px.
  + ****Medium devices:**** Min-width of 768px.
  + ****Large devices:**** Min-width of 992px.
  + ****Extra large devices:**** Min-width of 1200px.
* Implementation: Use ****@media**** rules to apply specific styles based on the device's width, optimizing layout and usability across different screen sizes.

------------------------------------------------------------------------------------------------------------------------------------------------------

Lesson : Summary

### ****Introduction to CSS Functions****

* CSS functions are used to perform data processing or calculations to return a CSS value for a property.
* These functions represent complex data types and often take input arguments to calculate their return value.

### ****attr()****

* The attr() function retrieves the value of an attribute from an HTML element and uses it as a property value in CSS.
* Syntax: property: attr(attribute\_name)

### ****calc()****

* The calc() function performs calculations while defining values for CSS properties.
* Syntax: property: calc(expression)
* Supports standard mathematical operators: addition (+), subtraction (-), multiplication (\*), and division (/).

### ****max()****

* The max() function sets the largest value from a list of comma-separated expressions as the value of a CSS property.
* Syntax: property: max(value1, value2)

### ****min()****

* The min() function sets the smallest value from a list of comma-separated expressions as the value of a CSS property.
* Syntax: property: min(value1, value2)

### ****var()****

* The var() function defines and uses custom CSS variables, allowing for reusable values throughout the stylesheet.
* Syntax:
  + Declare: :root { --variable-name: value; }
  + Use: selector { property: var(--variable-name); }

### ****2D Transforms****

* In CSS, 2D transformations allow you to manipulate the position, rotation, and scaling of elements in a two-dimensional space on a web page.

#### ****translate()****

* Moves an element along the X and Y axes.
* Syntax: transform: translate(x, y)
* translateX() moves the element along the x-axis.
* translateY() moves the element along the y-axis.

#### ****scale()****

* Changes the width and height of an element.
* Syntax: transform: scale(x, y)
* scaleX() changes the width of an element.
* scaleY() changes the height of an element.

#### ****rotate()****

* Rotates an element based on an angle.
* Syntax: transform: rotate(angle)
* Positive values rotate the element clockwise.
* Negative values rotate the element counterclockwise.

#### ****skew()****

* Skews the element along the X and Y axes.
* Syntax: transform: skew(x-angle, y-angle)
* skewX() skews the element along the x-axis.
* skewY() skews the element along the y-axis.

#### ****matrix()****

* Defines a homogeneous 2D transformation matrix.
* Syntax: transform: matrix(a, b, c, d, tx, ty)
* Combines scaling, skewing, and translating in one function.

### ****3D Transforms****

3D transformations in CSS allow you to manipulate elements in a three-dimensional space on a web page, adding depth and creating more dynamic visual effects.

#### ****rotateX()****

* Rotates an element around the X-axis in 3D space.
* Creates a tilting or flipping effect along the horizontal axis.
* Syntax: transform: rotateX(angle)

#### ****rotateY()****

* Rotates an element around the Y-axis in 3D space.
* Creates a horizontal flipping or spinning effect.
* Syntax: transform: rotateY(angle)

#### ****rotateZ()****

* Rotates an element around the Z-axis in 3D space.
* Creates a twisting or spinning effect.
* Syntax: transform: rotateZ(angle)

### ****What is Transition?****

* CSS transitions enable control over the speed of animation when CSS attributes are modified.
* Components of a CSS transition:
  + Trigger for the animation
  + Start delay
  + Duration of the transition
  + Method of running the transitions

### ****Transition Properties****

CSS transitions allow for the smooth change of property values over a specified duration. Key transition properties include:

#### ****transition-property****

* Specifies which properties will be transitioned.
* Examples:
  + transition-property: none;
  + transition-property: color;
  + transition-property: width;
  + transition-property: all;

#### ****transition-duration****

* Specifies the duration of the transition in seconds or milliseconds.
* Examples:
  + transition-duration: 6s;
  + transition-duration: 120ms;

#### ****transition-timing-function****

* Specifies the timing function used for the transition.
* Predefined timing functions:
  + transition-timing-function: ease;
  + transition-timing-function: ease-in;
  + transition-timing-function: ease-out;
  + transition-timing-function: ease-in-out;

#### ****transition-delay****

* Specifies the delay before the transition starts in seconds or milliseconds.
* Examples:
  + transition-delay: 3s;
  + transition-delay: 3000ms;

### ****Transition Shorthand****

* Transitions can be defined in one line using the shorthand syntax.
* Syntax: transition: property\_name duration easing\_function delay

### ****What is a Filter?****

The CSS property called filter is used to apply various graphical effects, such as color shifts or blurs, to an element. Filters are typically employed to modify how images, backgrounds, and borders are displayed.

### ****Syntax of Filter****

To apply filters, use the following syntax:

sql

Copy code

filter: <filter-function>

Filter functions are predefined functions used to apply various visual effects to elements on a web page.

### ****Common Filter Functions****

* ****blur()****
  + Applies a Gaussian blur to the input image.
  + Syntax: filter: blur(radius)
* ****brightness()****
  + Makes the image appear brighter or darker.
  + Syntax: filter: brightness(amount)
* ****grayscale()****
  + Converts the input image to grayscale.
  + Syntax: filter: grayscale(amount)
* ****hue-rotate()****
  + Rotates the hue of an element and its contents.
  + Syntax: filter: hue-rotate(angle)
* ****drop-shadow()****
  + Applies a drop shadow effect to the input image or text.
  + Syntax: drop-shadow(offset-x offset-y blur-radius color)

### ****Combining Multiple Filters****

* Multiple filters can be combined to give more visual effects.
* Syntax: filter: <filter-function-1> <filter-function-2>

### ****What is CSS Animation?****

* CSS animations enable transitions between different CSS style configurations.
* Consist of two parts: a style declaration and a collection of keyframes.
* Keyframes define the start, end, and possible intermediate states of the animation.

### ****Style Declaration****

* ****animation-name****: Specifies the name of the animation.
* ****animation-duration****: Defines how long the animation runs (seconds or milliseconds).
* ****animation-timing-function****: Controls animation speed (e.g., linear, ease, ease-in, ease-out).
* ****animation-delay****: Sets a delay before the animation starts.
* ****animation-iteration-count****: Specifies the number of times the animation repeats (e.g., a number or infinite).
* ****animation-direction****: Determines the direction of the animation (normal, reverse, alternate, alternate-reverse).
* ****animation-fill-mode****: Controls animation behavior before and after it runs (forwards, backwards, both, none).
* ****animation-play-state****: Indicates if the animation is running or paused (running, paused).

### ****Animation Shorthand Syntax****

Combines multiple animation properties into one line:

animation: name duration timing-function delay iteration-count direction fill-mode;



### ****Animation Sequence - Keyframes****

* Keyframes define the visual changes at different points in the animation sequence.

### ****Animation vs Transition****

|  |  |  |
| --- | --- | --- |
| ****Feature**** | ****Animation**** | ****Transition**** |
| ****Behavior**** | Moves from initial to final state with intermediate steps. | Moves only from initial to final state. |
| ****Iteration**** | Can set loop count using animation-iteration-count. | Runs only once. |
| ****Trigger**** | Can run automatically or with a trigger. | Runs on a trigger (e.g., hover). |
| ****Direction**** | Runs forward, in reverse, or alternate directions. | Runs forwards on trigger and reverses on trigger release. |
| ****Complexity**** | Ideal for creating complex series of movements. | Suitable for simple movements. |

### ****Introduction to CSS Webkit****

* ****CSS Webkit**** is an extension specific to Webkit-based browsers like Safari and Google Chrome.
* It provides additional CSS properties prefixed with -webkit- to enhance styling capabilities.
* These extensions are designed to work with Webkit's rendering engine to achieve unique visual effects not covered by standard CSS.

### ****Comparison between Standard CSS and Webkit CSS****

* ****Standard CSS**** encompasses styles that work across all modern browsers, including Firefox, Edge, and Chrome.
* It adheres strictly to official CSS specifications defined by W3C.
* ****WebKit CSS****, on the other hand, includes properties and values specific to Webkit-based browsers.
* These properties are prefixed with -webkit- and are not part of the official CSS standard.
* They are tailored for Safari and Chrome, offering capabilities such as custom scrollbars and specific visual effects.

### ****Common Webkit Properties****

* ****-webkit-border-radius****: Adds rounded corners to elements for a modern appearance.
* ****-webkit-box-shadow****: Applies drop shadow effects to elements.
* ****-webkit-transition****: Creates smooth transitions when CSS properties change.
* ****-webkit-transform****: Enables transformations like rotation and scaling.
* ****-webkit-overflow-scrolling****: Controls scrolling behavior on touch devices for improved user experience.

### ****Understanding CSS Custom Properties****

* ****CSS custom properties****, also known as CSS variables or cascading variables, are defined by CSS authors to store specific values for reuse throughout a document.
* They are declared using custom property notation, e.g., --main-color: black;, and accessed using the var() function, e.g., color: var(--main-color);.
* Custom properties offer efficiency by centralizing values that are repeated across a stylesheet, simplifying global changes.
* They enhance code readability with semantic identifiers, e.g., --main-text-color is more understandable than #00ff00.
* Variable names should adhere to CSS naming conventions, starting with double dashes (--) and containing only letters and dashes (-).