**CSS Complete Reference**

1. **What is CSS?**

**Definition:**  
CSS (Cascading Style Sheets) is a stylesheet language used for describing the presentation of a document written in HTML. It is used to style the layout, colors, fonts, spacing, and overall appearance of web pages.

**History:**  
CSS was created by Håkon Wium Lie in 1994 to separate content from presentation on web pages. This innovation allowed web pages to be more flexible, consistent, and accessible.

1. **CSS Syntax** CSS is written using the following syntax:

selector {

property: value;

}

* **Selector:** Specifies which HTML element(s) the styles will apply to.
* **Property:** The aspect of the element to be styled (e.g., color, font-size).
* **Value:** The value assigned to the property (e.g., red, 16px).

**Example:**

h1 {

color: blue;

font-size: 36px;

}

In this case, h1 is the selector, and it is styled with the properties color and font-size.

1. **CSS Selectors** CSS selectors allow you to target specific HTML elements for styling.

* **Universal Selector \*:**  
  Targets all elements in the document.

\* {

margin: 0;

padding: 0;

}

* **Element Selector:**  
  Targets elements by their tag name.

p {

color: gray;

}

* **Class Selector .class-name:**  
  Targets elements with a specific class attribute.

.container {

width: 80%;

margin: 0 auto;

}

* **ID Selector #id-name:**  
  Targets an element with a specific id.

#header {

background-color: #333;

color: white;

}

* **Descendant Selector (space):**  
  Selects elements that are descendants of a specific element.

.nav li {

display: inline-block;

margin: 0 10px;

}

* **Child Selector >:**  
  Selects direct child elements.

.parent > .child {

color: red;

}

* **Attribute Selector [attribute]:**  
  Targets elements with a specific attribute.

input[type="text"] {

border: 1px solid #ccc;

}

* **Pseudo-classes:**
  + :hover: Target an element when it is hovered.
  + :focus: Target an element when it gains focus.
  + :nth-child(n): Select an element based on its position in the parent.
  + :last-child: Select the last child of a parent.
  + :checked: Select a checked input element.

Example:

a:hover {

color: red;

}

input:focus {

border-color: blue;

}

* **Pseudo-elements:**
  + ::before: Insert content before an element’s actual content.
  + ::after: Insert content after an element’s actual content.
  + ::first-letter: Style the first letter of a text block.
  + ::first-line: Style the first line of a text block.

Example:

p::first-letter {

font-size: 2em;

color: green;

}

div::before {

content: "Before content";

color: gray;

}

1. **CSS Box Model** The CSS Box Model defines how elements are rendered and their layout on the page. It consists of:

* **Content:** The actual content inside the element.
* **Padding:** Space between the content and the element’s border.
* **Border:** Surrounds the padding (if present) and content.
* **Margin:** Space outside the border.

**Example:**

div {

width: 300px;

padding: 20px;

border: 2px solid black;

margin: 10px;

}

This example applies padding inside the box, a border around the content, and a margin outside the box.

1. **CSS Flexbox** Flexbox is a one-dimensional layout system that allows easy alignment of items in a row or column.

* **display: flex;**  
  Defines a flex container.
* **flex-direction:**  
  Specifies the direction of the flex items.

.container {

display: flex;

flex-direction: row;

}

* **justify-content:**  
  Aligns items along the main axis.

.container {

justify-content: space-between;

}

* **align-items:**  
  Aligns items along the cross axis.

.container {

align-items: center;

}

* **flex-wrap:**  
  Allows items to wrap onto the next line.

.container {

flex-wrap: wrap;

}

* **flex:**  
  Specifies how an item should grow or shrink.

.item {

flex: 1;

}

**Example:**

.container {

display: flex;

justify-content: space-between;

align-items: center;

}

.item {

flex: 1;

margin: 10px;

}

1. **CSS Grid Layout** CSS Grid is a two-dimensional layout system that allows both columns and rows to be designed simultaneously.

* **display: grid;**  
  Defines a grid container.
* **grid-template-columns:**  
  Defines the columns of the grid.

.container {

display: grid;

grid-template-columns: repeat(3, 1fr);

}

* **grid-template-rows:**  
  Defines the rows of the grid.

.container {

display: grid;

grid-template-rows: auto auto;

}

* **grid-gap:**  
  Defines the space between grid items.

.container {

display: grid;

grid-gap: 10px;

}

* **grid-column:**  
  Specifies the grid column where an item should be placed.

.item {

grid-column: span 2;

}

**Example:**

.container {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

grid-gap: 20px;

}

.item {

background-color: #ddd;

padding: 20px;

}

1. **CSS Transitions and Animations**

* **Transitions:**  
  Create smooth transitions when changing property values.

button {

background-color: blue;

transition: background-color 0.3s ease;

}

button:hover {

background-color: green;

}

* **Animations:**  
  Define animations with keyframes.

@keyframes slide {

0% { left: 0; }

100% { left: 100px; }

}

div {

position: relative;

animation: slide 2s infinite alternate;

}

**Font Properties:**

1. **font-family**  
   Defines the font to be used.  
   **Values:** serif, sans-serif, monospace, Arial, Verdana, etc.
2. **font-size**  
   Defines the font size.  
   **Values:** 12px, 1em, larger, small, etc.
3. **font-weight**  
   Defines the thickness of the font.  
   **Values:** normal, bold, bolder, lighter, 100, 200, etc.
4. **font-style**  
   Defines the style of the font.  
   **Values:** normal, italic, oblique.
5. **line-height**  
   Defines the height between lines of text.  
   **Values:** normal, 1.5, 20px, etc.
6. **font-variant**  
   Defines a variant of a font (e.g., small-caps).  
   **Values:** normal, small-caps.
7. **font-stretch**  
   Defines how the text is stretched horizontally.  
   **Values:** normal, condensed, expanded.

**Text Properties:**

1. **text-align**  
   Defines the horizontal alignment of text.  
   **Values:** left, right, center, justify.
2. **text-transform**  
   Controls the capitalization of text.  
   **Values:** none, capitalize, uppercase, lowercase.
3. **text-indent**  
   Specifies the indentation of the first line of text.  
   **Values:** 20px, 1em, etc.
4. **letter-spacing**  
   Defines the spacing between characters.  
   **Values:** normal, 2px, 0.1em, etc.
5. **word-spacing**  
   Defines the spacing between words.  
   **Values:** normal, 5px, etc.
6. **text-decoration**  
   Specifies text decoration, like underlines.  
   **Values:** none, underline, line-through, overline.
7. **text-shadow**  
   Applies a shadow effect to text.  
   **Values:** 2px 2px 5px gray, 0 0 10px rgba(0, 0, 0, 0.7).
8. **white-space**  
   Defines how white space inside an element is handled.  
   **Values:** normal, nowrap, pre, pre-wrap, pre-line.
9. **text-overflow**  
   Specifies what happens when text overflows its container.  
   **Values:** clip, ellipsis.

**Example Using Font and Text Properties:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Font and Text Properties Example</title>

<style>

.example {

font-family: 'Arial', sans-serif;

font-size: 20px;

font-weight: bold;

font-style: italic;

line-height: 1.6;

text-align: center;

text-transform: uppercase;

letter-spacing: 2px;

word-spacing: 5px;

text-decoration: underline;

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);

white-space: nowrap;

}

</style>

</head>

<body>

<div class="example">

This is a sample text demonstrating font and text properties.

</div>

</body>

</html>

In this example:

* **Font properties** like font-family, font-size, and font-weight style the text.
* **Text properties** like text-align, text-transform, and text-decoration affect how the text is displayed.

**Link**

* **color:** Defines the color of the text.  
  Values: red, #ff0000, rgb(255, 0, 0).
* **text-decoration:** Defines text decoration.  
  Values: none, underline, line-through.
* **text-transform:** Defines text capitalization.  
  Values: uppercase, lowercase, capitalize.

**Border and Outline**

* **border:** Defines the border of an element.  
  Values: `1px solid black, 2px dashed red`, etc.
* **outline:** Similar to border but does not affect layout.  
  Values: none, 2px solid blue.

**Margin, Padding, and Positioning**

* **margin:** Defines the space outside an element.  
  Values: 10px, auto, 5%.
* **padding:** Defines the space inside an element.  
  Values: 10px, 1em, 5%.
* **position:** Defines how an element is positioned.  
  Values: static, relative, absolute, fixed, sticky.

**Visibility, Display, and Box Model**

* **visibility:** Specifies if an element is visible or hidden.  
  Values: visible, hidden, collapse.
* **display:** Defines how an element is displayed.  
  Values: block, inline, inline-block, flex, grid, none.

**Box Shadow, Text Shadow, and Transparency**

* **box-shadow:** Adds shadow to an element’s box.  
  Values: 10px 10px 15px rgba(0, 0, 0, 0.5).
* **text-shadow:** Adds shadow to text.  
  Values: 2px 2px 3px rgba(0, 0, 0, 0.5).
* **opacity:** Defines the transparency of an element.  
  Values: 1 (fully opaque), 0.5 (50% transparent), 0 (fully transparent).

1. **Defining Colors and Transparency** There are several ways to define colors in CSS:

* **Named Colors:**  
  Values: red, blue, green, etc.
* **Hexadecimal Colors:**  
  Format: #RRGGBB  
  Example: #ff0000 (Red)
* **RGB Colors:**  
  Format: rgb(red, green, blue)  
  Example: rgb(255, 0, 0) (Red)
* **RGBA Colors:**  
  Format: rgba(red, green, blue, alpha)  
  Example: rgba(255, 0, 0, 0.5) (50% transparent red)

**Units Used in CSS:**

* **px** (Pixels): Absolute unit for screen resolution.
* **em**: Relative to the font size of the element.
* **rem**: Relative to the root element’s font size.
* **%**: Relative to the parent element’s dimensions.
* **vw** (Viewport width): Relative to the width of the viewport.
* **vh** (Viewport height): Relative to the height of the viewport.