

CSS

CSS

- **CSS** stands for **Cascading Style Sheets**
 - Styles define **how to display** HTML elements
 - Styles are normally stored in **Style Sheets**
 - Multiple style definitions will **cascade** into one

Anatomy of CSS Rule

Syntax

- Is made up of three parts:
 - a selector, a property and a value:

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

```
• Ex: body {  
  color: black;  
}
```

Element selector

- If the value is multiple words, put quotes around the value:
 - `p {
font-family: "Times New Roman", Times, serif;
}`
- To specify more than one property, you must separate each property with a semicolon.
 - `p {
text-align:center; color:red
}`

Class selector

- With the class selector, different styles can be defined for the same type of HTML element.
 - `p.right {text-align: right}`
 - `p.center {text-align: center}`
 - `p.bold{font-weight:bold}`
- You have to use the class attribute in your HTML element:
 - `<p class="right"> This paragraph will be right-aligned. </p>`
 - `<p class="center"> This paragraph will be center-aligned. </p>`
- To apply more than one class per given element, the syntax is:
 - `<p class="center bold"> This is a paragraph. </p>`
 - The paragraph above will be styled by the class "center" AND the class "bold".

Anonymous class selector

- The tag name in the selector can be omitted to define a style that will be used by all HTML elements that have a certain class.
- In the example below, all HTML elements with class="center" will be center-aligned:
- `.center {text-align: center;}`
- Ex: `<h1 class="center"> This heading will be center-aligned </h1><p class="center"> This paragraph will also be center-aligned. </p>`

Add Styles to Elements with Particular Attributes

- Styles can be applied to HTML elements with particular attributes.
- The style rule below will match all input elements that have a type attribute with a value of "text":
- Ex: `input[type="text"] {background-color: blue}`

ID selector

- Define styles for HTML elements with the id selector.
- The id selector is defined as a #.
- The style rule below can be used with an element that has an id with a value of "para1":
- `#para1 { text-align: center; color: red }`

External Style Sheet

```
<head>
```

```
<link rel="StyleSheet" href="mystyle.css" type="text/css" />
```

```
</head>
```

Internal Style Sheet

```
<head>
<style type="text/css">
  h1 {color: red}
  p {margin-left: 20px}
  h1.BlueOnes{color:blue;}
  h1.RedOnes{color:red;}
  #hcolor{color:red;}
</style>
</head>
```

```
<h1>Am I in red</h1>
<h1 CLASS="BlueOnes">Blue color text</h1>
<h1 id="hcolor">Red color text</h1>
```

Inline Styles

`<p style="color: sienna; margin-left: 20px"> This is a paragraph </p>`

CASCADING effect

- Styles can be specified
 - inside a single HTML element
 - inside the <head> element of an HTML page
 - In an external CSS file.
- Even multiple external style sheets can be referenced inside a single HTML document.
- All the styles will "cascade" into a new "virtual" style sheet by the priority rules:
 - Browser default
 - External style sheet or Linked
 - Internal style sheet or Embedded (inside the <head> tag)
 - Inline style (inside an HTML element)

Example

```
<!DOCTYPE html>
<html>
<head>
<link rel="StyleSheet" href="../CSS/mystyle.css"
type="text/css" />
<style>
body{ background-color:black
}
h1 {
border:1px orange inset;
color: white;
text-align: center
}
</style>
</head>
```

```
<body>
<h1 style="color:yellow">My First CSS
Example</h1>
</body>
</html>
```

```
//mystyle.css
h1 {
color: red;
text-align: right;
height:100px;
/* background-color:gray; */
}
```

Box-model



Margin

- `margin: length | auto | initial | inherit;`
- Default: 0
- `margin: 10px 5px 15px 20px;`
 - top margin is 10px
 - right margin is 5px
 - bottom margin is 15px
 - left margin is 20px
- `margin: 10px 5px 15px;`
 - top margin is 10px
 - right and left margins are 5px
 - bottom margin is 15px
- `margin: 10px 5px;`
 - top and bottom margins are 10px
 - right and left margins are 5px
- `margin: 10px;`
 - all four margins are 10px

Border

- border: *border-width border-style border-color* | initial | inherit;
- Default: medium none elementcolor
- border: 3px inset red;

Eg: border-bottom: 6px solid red;
background-color: lightgrey;
border-color: red green blue yellow;
border-top-style: dotted;

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.

Border radius

- border-radius: *1-4 length | % / 1-4 length | %* | initial | inherit;
- Default: 0
- border-radius: 15px 50px 30px 5px:
- border-radius: 15px 50px 30px:
- border-radius: 15px 50px:
- border-top-left-radius: 1em 5em
border-top-right-radius
border-bottom-right-radius
border-bottom-left-radius

Padding

- padding: *length* | initial | inherit;
- padding: 2cm 4cm 3cm 4cm;

Dimension

- height: auto | *length* | initial | inherit;
- width: auto | *value* | initial | inherit;
- max-height: none | *length* | initial | inherit;
- max-width: none | *length* | initial | inherit;
- min-height: *length* | initial | inherit;
- min-width: *length* | initial | inherit;

Overflow

- **Overflow:** Specifies what happens if content overflows an element's box
- **overflow-x:** Specifies what to do with the left/right edges of the content if it overflows the element's content area
- **overflow-y:** Specifies what to do with the top/bottom edges of the content if it overflows the element's content area
 - (visible, hidden, auto, scroll)

Box-sizing

- By default, the width and height of an element is calculated like this:
 - $\text{width} + \text{padding} + \text{border} = \text{actual width of an element}$
 - $\text{height} + \text{padding} + \text{border} = \text{actual height of an element}$
- The CSS box-sizing property allows us to include the padding and border in an element's total width and height.
- **box-sizing: content-box | border-box | initial | inherit;**
 - **content-box:** Default. The width and height properties (and min/max properties) includes only the content. Border and padding are not included
 - **border-box:** The width and height properties (and min/max properties) includes content, padding and border

Note

In general,

- Margins of horizontally aligned elements cumulate
- Margins of vertically aligned elements collapse
- box-sizing: border-box is preferred by the developers

Background

- background: *bg-color bg-image position/bg-size bg-repeat bg-origin bg-clip bg-attachment* initial|inherit;
- background-color: *color*|transparent|initial|inherit;
- background-image: *url*|none|initial|inherit;
- background-position: *value/xpos ypos/ x% y%*;
- background-size: auto|*length*|cover|contain|initial|inherit;
- background-repeat: repeat|repeat-x|repeat-y|no-repeat|initial|inherit;
- background-origin: padding-box|border-box|content-box|initial|inherit;
- background-clip: border-box|padding-box|content-box|initial|inherit;
- background-attachment: scroll|fixed|initial|inherit;

Font

- font: *font-style font-variant font-weight font-size/line-height font-family* | initial | inherit;
- font-style: normal | italic | oblique | initial | inherit;
- font-variant: normal | small-caps | initial | inherit;
- font-weight: normal | bold | bolder | lighter | *number* (100-900) | initial | inherit;
- font-size: medium | xx-small | x-small | small | large | x-large | xx-large | smaller | larger | *length* | initial | inherit;
- line-height: normal | *number* | *length* | initial | inherit;
- font-family: *font* | initial | inherit;

Font family

Serif

- Georgia, Palatino Linotype, Book Antiqua, Times New Roman, Times etc.

Sans-Serif

- Arial, Helvetica, Arial Black, Gadget, Comic Sans MS, cursive, Impact, Charcoal, Lucida Sans Unicode, Lucida Grande etc.

Monospace

- Courier New, Courier, Lucida Console, Monaco etc.

Text

- text-align: left | right | center | justify | initial | inherit;
- text-decoration: none | underline | overline | line-through | initial | inherit;
- text-indent: *length* | initial | inherit;
- text-overflow: clip | ellipsis | *string* | initial | inherit;
- text-shadow: *h-shadow v-shadow blur-radius color* | none | initial | inherit;
- text-transform: none | capitalize | uppercase | lowercase | initial | inherit;
- vertical-align: baseline | *length* | sub | super | top | text-top | middle | bottom | text-bottom | initial | inherit;
- direction: ltr | rtl | initial | inherit;

Table

- border-collapse: separate | collapse | initial | inherit;
- border-spacing: *length* | initial | inherit;
- caption-side: top | bottom | initial | inherit;
- empty-cells: show | hide | initial | inherit;

List

- `list-style: list-style-type list-style-position list-style-image | initial | inherit;`
- `list-style-image: none | url | initial | inherit;`
- `list-style-position: inside | outside | initial | inherit;`
- `list-style-type: value;`
- Ex:
 - `ul.a {list-style-type: circle;}`
 - `ul.b {list-style-type: square;}`
 - `ol.c {list-style-type: upper-roman;}`
 - `ol.d {list-style-type: lower-alpha;}`
 - `ul { list-style-image: url('sqpurple.gif'); }`

Display

Inline: Default value. Displays an element as an inline element (like)

Block: Displays an element as a block element (like <p>)

Inline-block: Display inline element which can accommodate height and width

None: The element will not be displayed at all (has no effect on layout)

Initial: Sets this property to its default value.

Inherit: Inherits this property from its parent element.

```
<html>
<head> <style>
p {
  display: inline;
} </style>
</head>
<body> <p>This is a paragraph.</p> <p>This is a paragraph.</p><p>This is a paragraph.</p>
<p>This is a paragraph.</p> <p>This is a paragraph.</p>
</body>
</html>
```