Instructor: Sudad H. Abed / Email: shabed@uoanbar.edu.iq

Lecture Reference: www.w3schools.com

❖ CSS:

- **CSS** stands for **C**ascading **S**tyle **S**heets.
- **CSS describes** how HTML elements are to be **displayed.**
- **CSS saves a lot of work**. It can control the layout of multiple web pages all at once.
- External stylesheets are stored in **CSS files**

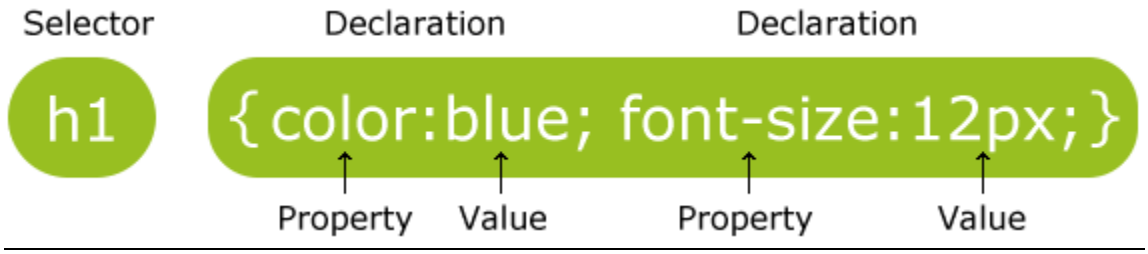
- **CSS Solved a Big Problem (how):**
  - HTML was created to **describe the content** of a web page, like:

    <h1>This is a heading</h1>

    This is a paragraph.
  - HTML was NEVER intended to contain tags for formatting a web page!
  - When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.
  - ✓ To solve this problem, the World Wide Web Consortium (W3C) created CSS. CSS removed the style formatting from the HTML page!

❖ CSS Syntax and Selectors:

A CSS rule-set consists of a selector and a declaration block:



- ✓ The selector points to the HTML element you want to style.
- ✓ The declaration block contains one or more declarations separated by semicolons.
- ✓ Each declaration includes a CSS property name and a value, separated by a colon.
- ✓ A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

❖ CSS Syntax and Selectors:

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

- **The element Selector:**

  The element selector selects elements based on the element name.

  You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

Ex:

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

- **The id Selector:**

  - The id selector uses the id attribute of an HTML element to select a specific element.
  - The id of an element should be unique within a page, so the id selector is used to select one unique element!
  - To select an element with a specific id, write a hash (#) character, followed by the id of the element.
  - The style rule below will be applied to the HTML element with id="para1":

Ex:

```
#para1 {
    text-align: center;
    color: red;
}
```

Note: An id name cannot start with a number!

- **The class Selector:**
  - The class selector selects elements with a specific class attribute.
  - To select elements with a specific class, write a period (.) character, followed by the name of the class.
  - In the example below, all HTML elements with class="center" will be red and center-aligned:

Ex:

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

You can also specify that only specific HTML elements should be affected by a class. In the example below, only <p> elements with class="center" will be center-aligned:

Ex:

```
p.center {
    text-align: center;
    color: red;
}
```

HTML elements can also refer to more than one class. In the example below, the <p> element will be styled according to class="center" and to class="large":

Ex:

```
<p class="center large">This paragraph refers to two classes.</p>
```

Note: An id name cannot start with a number!

- **Grouping Selectors:**

If you have elements with the same style definitions, It will be better to group the selectors, to minimize the code.

- To group selectors, separate each selector with a comma.

Ex:

```
h1, h2, p {
   text-align: center;
   color: red;
}
```

- **CSS Comments:**

A CSS comment starts with /* and ends with */. Comments can also span multiple lines:

Ex:

```
p {
    color: red;
    /* This is a single-line comment */
    text-align: center;
}

/* This is
a multi-line
comment */
```

❖ Ways to Insert CSS:

There are three ways of inserting a style sheet:

- ✓ External style sheet
- ✓ Internal style sheet
- ✓ Inline style

- **External Style Sheet:**

➢ With an external style sheet, you can change the look of an entire website by changing just one file!

➢ Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section:

Ex:

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

- An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.
- Here is how the "mystyle.css" looks:

```
body {
    background-color: lightblue;
}

h1 {
    color: navy;
    margin-left: 20px;
}
```

Note: Do not add a space between the property value and the unit (such as margin-left: 20 px;). The correct way is: margin-left: 20px;

- **Internal Style Sheet:**
  - An internal style sheet may be used if one single page has a unique style.
  - Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

    Ex:

```
<head>
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
```

- **Inline Styles:**

  - An inline style may be used to apply a unique style for a single element.
  - To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.
  - The example below shows how to change the color and the left margin of a <h1> element:

    Ex:

```
<h1 style="color:blue;margin-left:30px;">This is a heading</h1>
```

- **Multiple Style Sheets:**

  - If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Ex:

Assume that an external style sheet has the following style for the <h1> element:

```
h1 {
    color: navy;
}
```

then, assume that an internal style sheet also has the following style for the <h1> element:

```
h1 {
    color: orange;
}
```

If the internal style is defined after the link to the external style sheet, the <h1> elements will be "orange":

Ex:

```
<head>
   <link rel="stylesheet" type="text/css" href="mystyle.css">
   <style>
      h1 {
          color: orange;
      }
   </style>
</head>
```

However, if the internal style is defined before the link to the external style sheet, the <h1> elements will be "navy":

Ex:

```
<head>
   <style>
       h1 {
           color: orange;
       }
    </style>
    <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

- **Cascading Order:**

What style will be used when there is more than one style specified for an HTML element?

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style (inside a specific HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or a browser default value.

❖ CSS Backgrounds:

The CSS background properties are used to define the background effects for elements.

CSS background properties:

- ✓ background-color
- ✓ background-image
- ✓ background-repeat
- ✓ background-attachment
- ✓ background-position

- **Background-Color:**

➢ The background-color property specifies the background color of an element.

The background color of a page is set like this:

Ex:

```
body {
    background-color: lightblue;
}
```

✓ With CSS, a color is most often specified by:

- a valid color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

- **Background-Image:**

➢ The background-image property specifies an image to use as the background of an element.

➢ By default, the image is repeated so it covers the entire element.

The background image for a page can be set like this:

Ex:

```
body {
    background-image: url("paper.gif");
}
```

➢ **background-repeat:**

➢ By default, a background-image is repeated both vertically and horizontally.

➢ The background-repeat property sets if/how a background image will be repeated.

```
background-repeat: no-repeat;  /* No repeat */
background-repeat: repeat-y; /* Repeat vertically */
background-repeat: repeat-x; /* Repeat horizontally */
```

Ex:

```
body {
    background-image: url("paper.gif");
    background-repeat: no-repeat;
}
```

- **background- attachment:**

The background-attachment property sets whether a background image is fixed or scrolls with the rest of the page.

Ex:

```
body {
    background-image: url('w3css.gif');
    background-repeat: no-repeat;
    background-attachment: fixed;
}
```

- **background- position:**

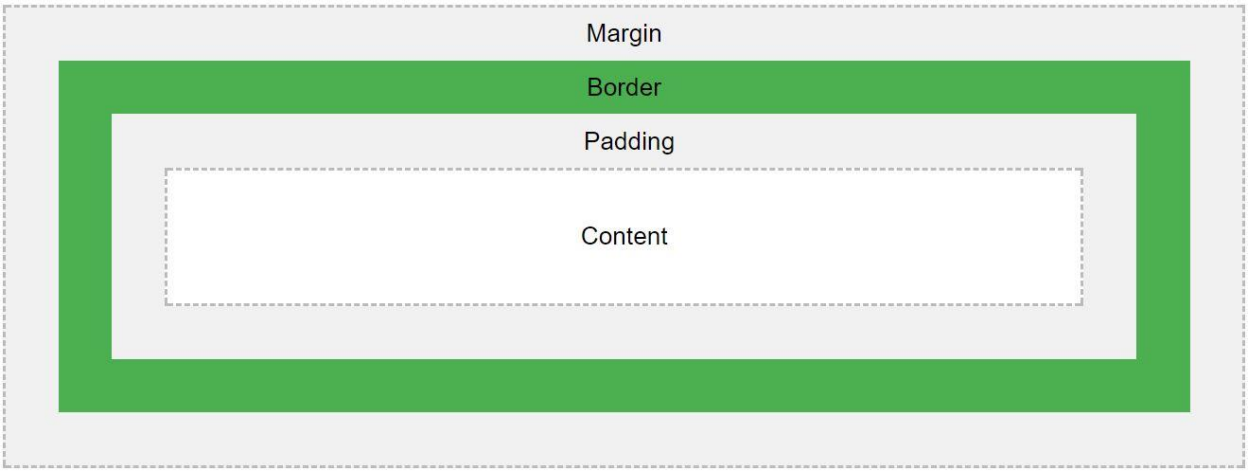
The background-position property sets the starting position of a background image.

Ex:

```
body {
    background-image: url('smiley.gif');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: center;
}
```

❖ CSS Box Model:

➢ The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



Explanation of the different parts:

- ✓ **Content** - The content of the box, where text and images appear
- ✓ **Padding** - Clears an area around the content. The padding is transparent
- ✓ **Border** - A border that goes around the padding and content
- ✓ **Margin** - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

Ex:

```
div {
    width: 300px;
    border: 25px solid green;
    padding: 25px;
    margin: 25px;
}
```

❖ CSS Borders:

The CSS border properties allow you to specify the style, width, and color of an element's border.

• Border Style:

The border-style property specifies what kind of border to display.

- ✓ dotted - Defines a dotted border
- ✓ dashed - Defines a dashed border
- ✓ solid - Defines a solid border
- ✓ double - Defines a double border
- ✓ groove - Defines a 3D grooved border. The effect depends on the border-color value
- ✓ ridge - Defines a 3D ridged border. The effect depends on the border-color value
- ✓ inset - Defines a 3D inset border. The effect depends on the border-color value
- ✓ outset - Defines a 3D outset border. The effect depends on the border-color value
- ✓ none - Defines no border
- ✓ hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

Ex:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

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 Width:**

➢ The border-width property specifies the width of the four borders.

➢ The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

➢ The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).

Ex:

```
p.one {
    border-style: solid;
    border-width: 5px;
}
p.two {
    border-style: solid;
    border-width: medium;
}
p.three {
    border-style: solid;
    border-width: 2px 10px 4px 20px;
}
```

- **Border Color:**

➢ The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border).

➢ If border-color is not set, it inherits the color of the element.

Ex:

```
p.one {
    border-style: solid;
    border-color: red;
}
p.three {
    border-style: solid;
    border-color: red green blue yellow;
}
```

You can also specify all the individual border properties for just one side:

Ex:

```
p {
    border-left: 6px solid red;
    background-color: lightgrey;
}
```

❖ CSS Margins:

The CSS margin properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

- **Margin - Individual Sides:**

CSS has properties for specifying the margin for each side of an element:

- ✓ margin-top
- ✓ margin-right
- ✓ margin-bottom
- ✓ margin-left

Ex:

```
p {
    margin-top: 100px;
    margin-bottom: 100px;
    margin-right: 150px;
    margin-left: 80px;
}
```

❖ CSS Padding:

- ➢ The CSS padding properties are used to generate space around an element's content, inside of any defined borders.
- ➢ With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

- **Padding - Individual Sides:**

CSS has properties for specifying the padding for each side of an element:

- ✓ padding-top
- ✓ padding-right
- ✓ padding-bottom
- ✓ padding-left

Ex:

```
div {
    padding-top: 50px;
    padding-right: 30px;
    padding-bottom: 50px;
    padding-left: 80px;
}
```