

# Introduction to CSS

Course Code: CSC 3222

Course Title: Web Technologies



**Dept. of Computer Science**  
**Faculty of Science and Technology**

<b>Lecturer No:</b>	<b>09</b>	<b>Week No:</b>	<b>09</b>	<b>Semester:</b>	
<b>Lecturer:</b>	<i>Sazzad Hossain</i> <a href="mailto:sazzad@aiub.edu">sazzad@aiub.edu</a>				

# Lecture Outline



1. Introduction to CSS
2. CSS Selectors
3. CSS Insertion Types
4. Customizing HTML elements with CSS
5. CSS Position and Layouts

# Introduction to CSS

What is CSS?



- CSS stands for **Cascading Style Sheets**
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- 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 is used to define **styles** for your web pages, including the **design**, layout and variations in display for different **devices and screen sizes**.



# Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}
h1 {
  color: white;
  text-align: center;
}
p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>
<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Output:

## My First CSS Example

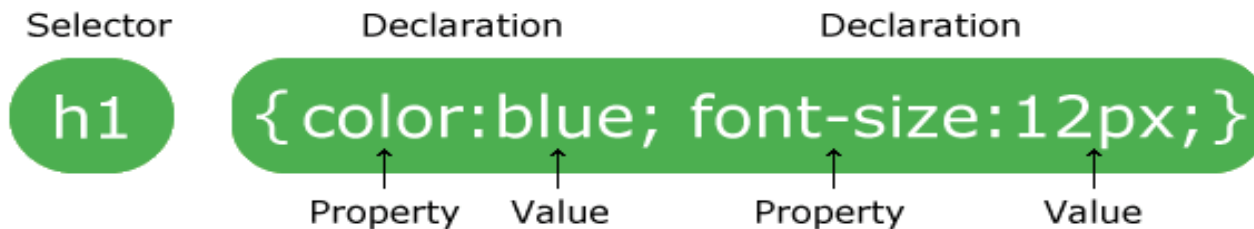
This is a paragraph.



## Explanation

- HTML was **NEVER** intended to contain tags for formatting a web page!
- HTML was created to describe the content of a web page, like:  
`<h1>This is a heading</h1>`  
`<p>This is a paragraph.</p>`
- Tags like `<font>`, and color attributes were added to the HTML 3.2 specification.
- Development of large websites, where fonts and color information were added to every single page, became a **long** and **expensive** process.
- CSS removed the style formatting from the HTML page!

# CSS Syntax



- **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**.
- Can not add a space between the **property value** and the **unit**



## Example

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

## Explanation

- `p` is a **selector** in CSS.
- `color` is a **property**, and `red` is the property value
- `text-align` is a **property**, and `center` is the property value

# CSS Selectors



CSS selectors divided into five categories:

- **Simple selectors** (select elements based on name, id, class)
- **Combinator selectors** (select elements based on a specific relationship between them)
- **Pseudo-class selectors** (select elements based on a certain state)
- **Pseudo-elements selectors** (select and style a part of an element)
- **Attribute selectors** (select elements based on an attribute or attribute value)





## The CSS element Selector

The element **selector** selects **HTML elements** based on the element name.

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



# The CSS id Selector

- The **id** selector uses the id attribute of an HTML element to select a specific element.
- The **id** of an element is **unique within a page**
- To select an element with a specific id with a hash (#) character, followed by the id of the element.
- An id name **cannot** start with a **number**

## Example

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

The CSS rule below will be applied to the HTML element with id="para1":



# The CSS class Selector

The class **selector** selects HTML **elements** with a specific **class** attribute. To select elements with a specific class need to use (.) character, followed by the class name.

In this example all HTML elements with class="center" will be red and center-aligned:

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

It can be specified only specific HTML elements should be affected by a class.

In this example only <p> elements with class="center" will be center-aligned:

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



# The CSS Universal Selector

The universal selector (\*) selects all HTML elements on the page.

Example

The CSS rule below will affect every HTML element on the page:

```
* {  
  text-align: center;  
  color: blue;  
}
```



# The CSS Grouping Selector

The **grouping selector** selects all the HTML 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.

Example

In this example we have grouped the selectors from the code above:

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

# Inserting CSS



There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS



# External CSS

- External style sheet is an external file which can be imported to the HTML file.
- HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.
- External styles are defined within the <link> element, inside the <head> section of an HTML page.

## Example

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

## External CSS

- An external style sheet can be written with a .css extension.
- The external .css file should not contain any HTML tags.

"**mystyle.css**"

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

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





# Internal CSS

- An internal style sheet may be used if one single HTML page has a unique style.
- The internal style is defined inside the <style> element, inside the head section.

## Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}
h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```



# Inline CSS

- used to apply a unique style for a single element.
- To use inline styles, must use **style** attribute to the relevant element
- The **style** attribute contains <!DOCTYPE html>

```
<html>
```

```
<body>
```

```
<h1 style="color:blue;text-align:center;">This is a heading</h1>
```

```
<p style="color:red;">This is a paragraph.</p>
```

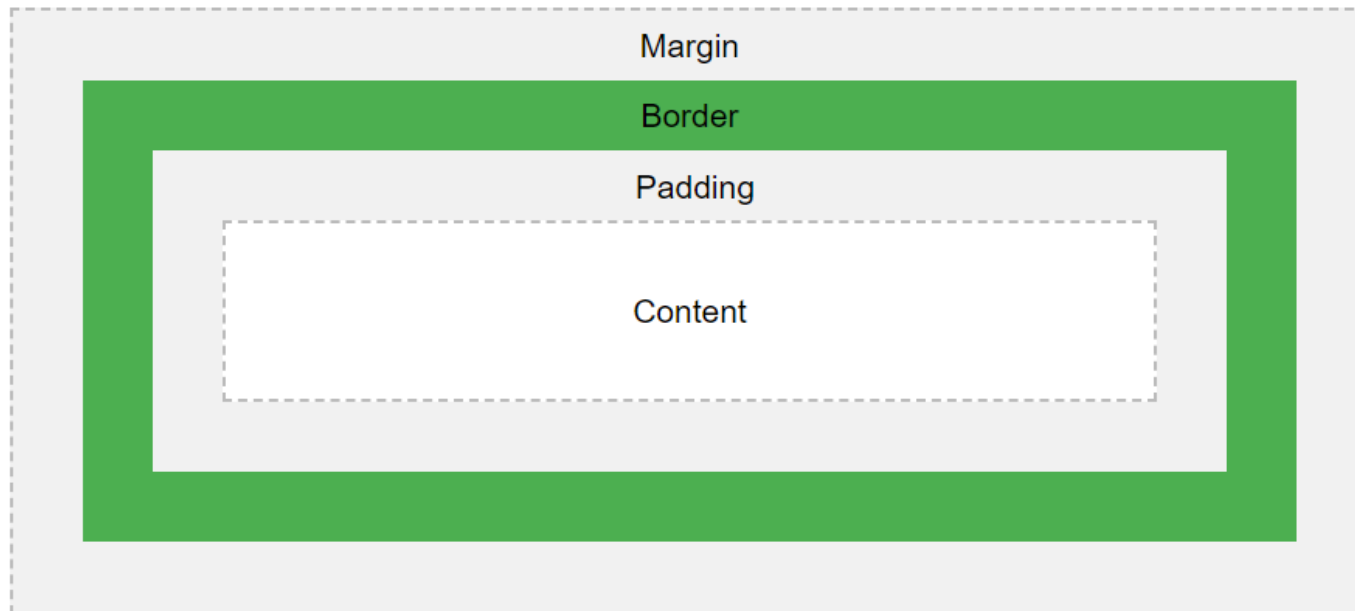
```
</body>
```

```
</html>
```

# 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.





# CSS Box Model

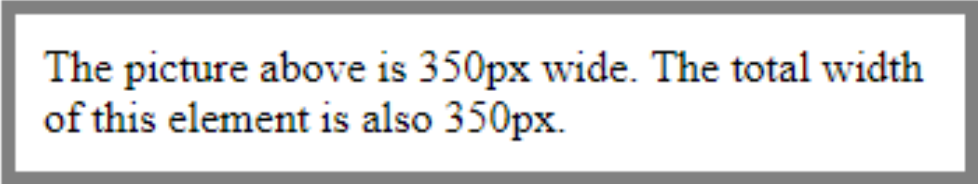
**Content** - The content of the box, where text and images appear

**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

```
div {  
  width: 320px;  
  padding: 10px;  
  border: 5px solid gray;  
  margin: 0;  
}
```

A diagram illustrating the CSS Box Model. It shows a rectangular box with a gray border. Inside the box, the text reads: 'The picture above is 350px wide. The total width of this element is also 350px.' The text is centered and has a light blue background.

The picture above is 350px wide. The total width of this element is also 350px.

The `<div>` tag defines a division or a section in an HTML document. The `<div>` element is often used as a container for other HTML elements to style them with CSS or to perform certain tasks with JavaScript.



# CSS Box Model

The total width of an element should be calculated :

320px (width)  
+ 20px (left + right padding)  
+ 10px (left + right border)  
+ 0px (left + right margin)  
**= 350px**

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

# CSS Layout and Position



The **position** property specifies the type of positioning method used for an element.

There are five different position values:

**static**

**relative**

**fixed**

**absolute**

**Sticky**

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.



## position: static;

- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.
- An element with position: static; is not positioned in any special way;
- it is always positioned according to the normal flow of the page:

This <div> element has position: static;  
Here is the CSS that is used:

Example

```
div.static {  
    position: static;  
    border: 3px solid #73AD21;  
}
```

### **position: static;**

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This div element has position: static;



## position: relative;

- An element with position: relative; is positioned relative to its normal position.
- Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.
- Other content will not be adjusted to fit into any gap left by the element.

This <div> element has position: relative;  
Here is the CSS that is used:

Example

```
div.relative {  
  position: relative;  
  left: 30px;  
  border: 3px solid #73AD21;  
}
```

**position: relative;**

An element with position: relative; is positioned relative to its normal position:

This div element has position: relative;





## position: fixed;

- An element with position: fixed; is positioned relative to the viewport
- it always **stays in the same place** even if the page is **scrolled**.
- The top, right, bottom, and left properties are used to position the element.
- A fixed element does not leave a gap in the page where it would normally have been located.

```
div.fixed {  
  position: fixed;  
  bottom: 0;  
  right: 0;  
  width: 300px;  
  border: 3px solid #73AD21;  
}
```



## position: absolute;

- absolute is positioned relative to the nearest positioned ancestor.
- an absolute positioned element has no positioned ancestors, it uses the document body, and **moves** along with page **scrolling**.
- "positioned" element is one whose position is anything except static.

```
div.relative {  
  position: relative;  
  width: 400px;  
  height: 200px;  
  border: 3px solid #73AD21;  
}  
div.absolute {  
  position: absolute;  
  top: 80px;  
  right: 0;  
  width: 200px;  
  height: 100px;  
  border: 3px solid #73AD21;  
}
```

This <div> element has position: relative;

This <div> element has  
position: absolute;



## position: sticky;

- sticky is positioned based on the user's scroll position.
- A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place.

```
div.sticky {  
  position: sticky;  
  top: 0;  
  background-color: green;  
  border: 2px solid #4CAF50;  
}
```

I am sticky!

quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.

# Website Layout



A website is often divided into headers, menus, content and a footer:





# Header And Navigation Bar

- A header is usually located at the top of the website
- It often contains a logo or the website name.
- A navigation bar contains a list of links

```
.header {  
  background-color: #F1F1F1;  
  text-align: center;  
  padding: 20px;  
}  
.topnav {  
  overflow: hidden;  
  background-color: #333;  
}  
.topnav a {  
  float: left;  
  display: block;  
  color: #f2f2f2;  
  text-align: center;  
  padding: 14px 16px;  
  text-decoration: none;  
}  
.topnav a:hover {  
  background-color: #ddd;  
  color: black;  
}
```

```
<body>  
  
<div class="header">  
  <h1>Header</h1>  
</div>  
  
<div class="topnav">  
  <a href="#">Link</a>  
  <a href="#">Link</a>  
  <a href="#">Link</a>  
</div>  
  
</body>
```

## Header

Link Link Link



# Main Body with Columns

- The main content is the biggest and the most important part of site.
- The side content is often used as an alternative navigation or to specify information relevant to the main content.
- Widths can be changed and that it should add up to 100% in total.

```
.column {  
  float: left;  
  padding: 10px;  
}
```

```
/* Left and right column */
```

```
.column.side {  
  width: 25%;  
}
```

## Side

Lorem ipsum dolor sit  
amet, consectetur  
adipiscing elit..

## Main Content

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Maecenas sit amet pretium  
urna. Vivamus venenatis velit nec neque  
ultrices, eget elementum magna tristique.  
Quisque vehicula, risus eget aliquam placerat,  
purus leo tincidunt eros, eget luctus quam orci  
in velit. Praesent scelerisque tortor sed  
accumsan convallis.

## Side

Lorem ipsum dolor sit  
amet, consectetur  
adipiscing elit..

```
/* Middle column */
```

```
.column.middle {  
  width: 50%;  
}
```



# Footer

The footer is placed at the bottom of page.  
It often contains information like copyright and contact info.

```
.footer {  
  background-color: #F1F1F1;  
  text-align: center;  
  padding: 10px;  
}
```



## References

- MySQL - [www.mysql.com](http://www.mysql.com)
- W3Schools Online Web Tutorials- [www.w3schools.com](http://www.w3schools.com)
- PHP Manual - [www.php.net](http://www.php.net)





## Books

- Sams Teach Yourself Ajax JavaScript and PHP All in One; Phil Ballard and Michael Moncur;
- Sams Publishing; 2010
- JavaScript Phrasebook; Christian Wenz; Sams Publishing; 2007
- PHP and MySQL Web Development, 4/E; Luke Welling and Laura Thomson; AddisonWesley Professional; 2009
- JavaScript for Programmers Paul J. Deitel and Harvey M. Deitel; Prentice Hall; 2009