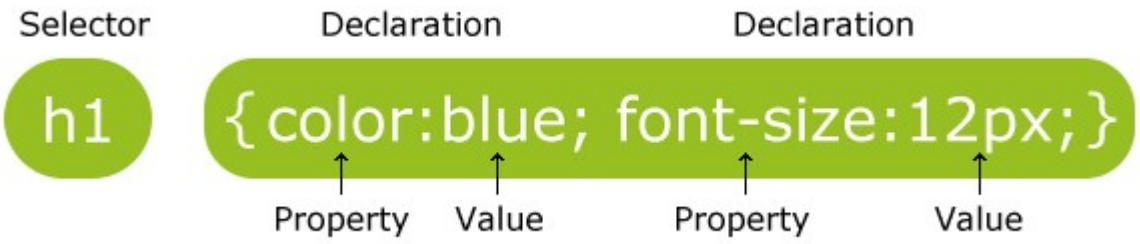
Web Technology (PS02CMCA33)

Unit 2

CSS3 selectors



The selector points to the HTML element you want to style.

In CSS, other aspect of selectors are patterns used to select the element(s) you want to style. For example, The .class selector selects elements with specific class attribute, CSS class reference can be given as below:

```
<head>
<style>
.test
{
.
.
.
}
</style>
```

```
</head>
<body>
<p class=”test”> Hello Students </p>
</body>
```

CSS3 classes

The .class selector selects elements with specific class attribute.

```
<head>
<style>
.test
{
background-color: yellow;
}
</style>
</head>
<body>
<p class=”test”> Hello Students </p>
</body>
```

CSS3 precedence rules

When the browser needs to resolve what styles to apply to a given HTML element, it uses a set of CSS precedence rules. Given these rules, the browser can determine what styles to apply. The rules are:

1. !important after CSS properties.
2. Specificity of CSS rule selectors.
3. Sequence of declaration.

I will explain what these rules mean in the following sections.

Note, that CSS precedence happens at CSS property level. Thus, if two CSS rules target the same HTML element, and the first CSS rule takes precedence over the second, then all CSS properties specified in the first CSS rule takes precedence over the CSS properties declared in the second rule. However, if the second CSS rule contains CSS properties that are not specified in the first CSS rule, then these are still applied. The CSS rules are combined - not overriding each other.

e.g. font-size: 16px **!important**;

Introduction to Media Query

Media queries are a feature of CSS that enable webpage content to adapt to different screen sizes and resolutions. They are a fundamental part of responsive web design and are used to customize the appearance of websites for multiple devices.

```
<style>

body {

  background-color: lightgreen;

}



@media only screen and (max-width: 600px) {

  body {

    background-color: lightblue;

  }

}

</style>
```

Introduction to JavaScript

JavaScript is Client-Side scripting language.

It is a language that is largely used in the World Wide Web (WWW) to add client-side interactivity to web pages.

Client side JavaScript is simply JavaScript that runs on client.

// Add examples of Simple if, else.. if and for loop (Which shows it
// is easy to learn).

JavaScript Syntax

JavaScript can be added in between <script> and </script>.

Syntax:

<script>

.

.

</script>

Types of JavaScript and their syntax:

1. Internal Java Script:

   <script type=”text/javascript”>

   .

   .

   </script>

   OR

   <script language=”javascript”>

   .

   .

   </script>

2. External JavaScript:

```
<script type=”text/javascript”    src=”test.css”>
.
.
</script>
```

OR

```
<script language=”javascript”    src=”test.css”>
.
.
</script>
```

Variables

- **Declaration**

  JavaScript variables are containers for storing data values.

  Declare / Create a JavaScript variable with the ***var*** keyword.

  e.g.

  ```
  var studnum;

  var a = 5;

  var b = 7;

  var res = a + b;
  ```

- **Data type**

  Following are different data types in JavaScript:

  Strings – Represents sequence of characters

  Number – Represents numeric values

Boolean – Represents Boolean (True / False)

Undefined – Represents undefined value

Null – Represents null i.e. no value at all

Strings

Represents sequence of characters.

e.g.

```
var txt = “GDCST”;

var txtlen = txt.length;
```

Numbers

JavaScript has only one type of number. Numbers can be written with or without decimals.

e.g.

```
var a = 10; // A number without decimals.

Var b = 15.25; // A number with decimals.
```

Arrays

An array is a special variable, which can hold more than one value at a time.

JavaScript arrays are used to store multiple values in a single variable.

e.g.

var cars = [“abc”,”def”,”ghi”,”ijk”,”lmn”];

Operators

An operator performs some operation on single or multiple operands (data values) and produces a result.

1. Assignment operator (=):

   var a = 25;

2. Arithmetic operators (* / + - %):

var res = a + b;

3. Increment / Decrement operators (++ --):

   ++a a++ --b b--

4. String operator (+):

   var a = “abc”;

   var b = “def”;

   var c = a + b;

5. Relational operators (< <= > >= == != === !==):

   if (a==b)

   {

   }

6. Logical operators (&& || !):

   if (a>b && a>c)

   {

   }

Functions

In JavaScript, function is defined by using the **function** keyword followed by function name and parentheses ( ).

A function can have zero or more parameters separated by commas.

Syntax:

Function Definition:

```
function fun_nm( )
{
    // function body
}
```

Function call:

```
fun_nm( );
```

Variable Scope

Scope determines the accessibility (visibility) of variables.

In JavaScript there are two types of scope:

1. Local scope:

   Variables declared within a JavaScript function.

2. Global scope:

   A variable declared outside a function. All scripts and functions on a web page can access it.

Event Handling

HTML events are things that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can “react” on these events.

We can handle events using Internal JavaScript / External JavaScript.

// Add appropriate examples.

Client-side Form Validation

Client-side form validation can be done by JavaScript. If a form field is empty, this function alerts a message, and returns false, to prevent the form being submitted.

JavaScript

```
function test1( )
{
var x = document.forms[“form”][“num”].value;
if (x === “ “)
{
alert(“fill the field”);
return false;
}
}
```

HTML form

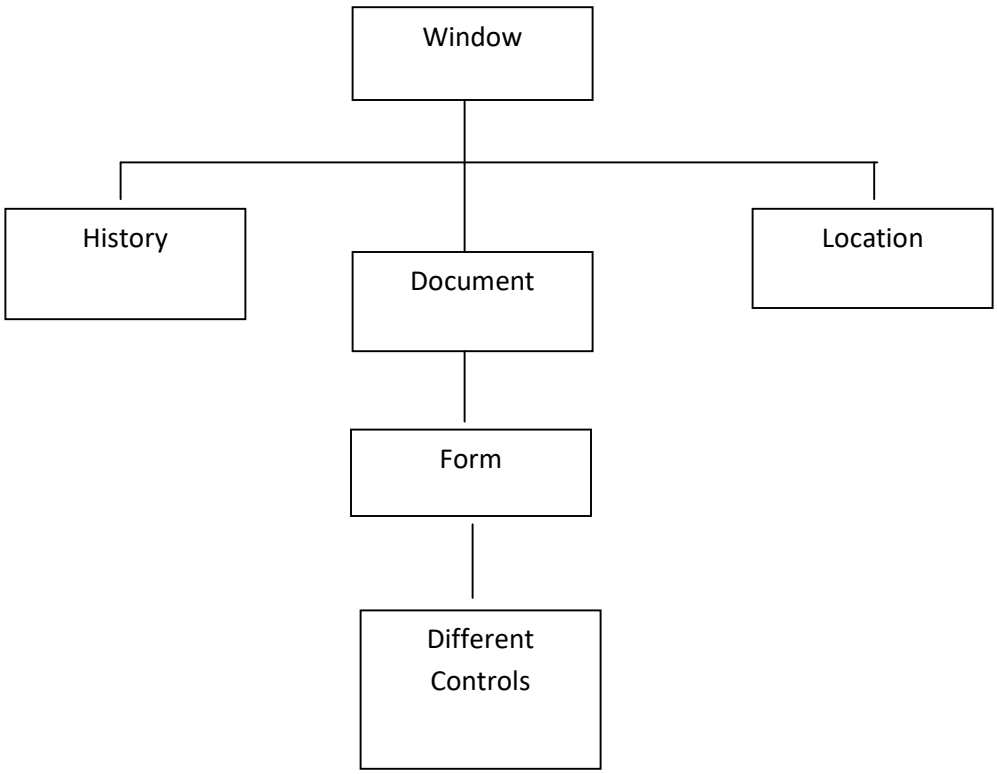
<form name=”form” action=”home.php” onsubmit=”return test1( )”>

Enter Number: <input type=”text” name=”num”/>

<input type=”submit” value=”Submit”/>

</form>

DOM access and manipulation from JavaScript



// Write different examples (e.g. all types of validations, Popup // boxes like, alert( ) , confirm( ) , prompt( ).

Built-in objects

In JavaScript, almost "everything" is an object.

- Booleans can be objects (if defined with the new keyword)
- Numbers can be objects (if defined with the new keyword)
- Strings can be objects (if defined with the new keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

  e.g var person = "John Doe";