Unit-2

CSS
Style Sheets

Need Of CSS

- CSS is a language that describes the style of an HTML document.
- CSS describes how HTML elements should be displayed.
- **CSS** to define how HTML elements should appear.

CSS Selectors

1. Element Selector
2. Id Selector
3. Class Selector
4. Group selector

Element Selector

- The element selector select the element based on element name

```
p
{
Background-color:red;
}
```

Id selector

- Id selector use the id attribute of the HTML element to select specific element.

```
#abc
{
background-color:red;
}
```

Class selector

- The class selector element based on class attribute.

```
.abc
{
Background-color:red;
}
```

Group selector

- If we have elements with same style.

```
p
{ background-color:red;}
h1
{ background-color:red;}
```

we can write above code as

```
p,h1
{ background-color:red;}
```

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS describes **how HTML elements are to be displayed on screen.**
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once.[With an external stylesheet file, you can change the look of an entire website by changing just one file!]
- External style sheets are stored in **CSS files.**

CSS Backgrounds

- CSS background properties:

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position

Background Color

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

Example

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

Background Color Example 2

- Example

```
h1 {
        background-color: green;
    }

div {
        background-color: lightblue;
    }

p {
        background-color: yellow;
    }
```

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.

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

Repeat Horizontally or Vertically

```
body
{
        background-image: url("gradient_bg.png");
        background-repeat: repeat-x;
    }
```

Background-repeat values:

repeat-x

repeat-y

no-repeat

Background-position

```
body {
      background-image: url("img_tree.png");
      background-repeat: no-repeat;
      background-position: right top;
   }
```

- It has three different types of values:

Length values (e.g. 100px 5px)

Percentages (e.g. 100% 5%)

Keywords (e.g. Right top)

Background-position

- The default values are 0 0. This places your background image at the top left of the container.
- The first value is the horizontal position, second value is the vertical position.
- So 100px 5px will move the image 100px to the right and five pixels down.

Background-position

- Keywords are just shortcuts for percentages. It's easier to remember and write top right than 0 100%, and that's why keywords are a thing. Here is a list of all five keywords and their equivalent values:

  top: 0% vertically

  right: 100% horizontally

  bottom: 100% vertically

  left: 0% horizontally

  center: 50% horizontally if horizontal isn't already defined. If it is then this is applied vertically.

Background Image - Fixed position

```
body
 {
      background-image: url("img_tree.png");
      background-repeat: no-repeat;
      background-position: right top;
      background-attachment: fixed;
   }
```

Shorthand Property Background

```
body
 {
background: #ffffff url("img_tree.png") no-repeat  fixed right top;
    }
```

Sequence:

background-color

background-image

background-repeat

background-attachment

background-position

CSS Borders

Different border Properties

- border-width:medium/thick/thin
- border-style (required)
- border-color
- Border-collapse: initial/seperate/collapse

Border Width

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

one

```
{
    border-style: solid;
    border-width: 5px;
}
```

Border Style

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

The following values are allowed:

- 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

Border Style

- 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

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).

```
.one {
    border-style: solid;
    border-color: red;
  }
```

Border - Individual Sides

- From the examples above you have seen that it is possible to specify a different border for each side.

```
p {
    border-top-style: dotted;
    border-right-style: solid;
    border-bottom-style: dotted;
    border-left-style: solid;
}
```

Border - Shorthand Property

```
p {
    border: 5px solid red;
  }
```

Sequence

- border-width
- border-style
- border-color

CSS Margins

- The CSS margin properties are used to generate space around elements.
- The margin properties set the size of the white space outside the border.
- CSS has properties for specifying the margin for each side of an element:

1. margin-top
2. margin-right
3. margin-bottom
4. margin-left

Example

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

Margin - Shorthand Property

```
p {
    margin: 100px 150px 100px 80px;
  }
```

Sequence

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

Margin - Shorthand Property

margin: 100px 150px 100px 80px;

Top Right Bottom Left

margin: 100px 150px 100px;

Top Right/Left Bottom

margin: 100px 150px;

Top/bottom Right/left

Marging: 100px;

All

CSS Padding

- The CSS padding properties are used to generate space around content.
- The padding clears an area around the content (inside the border) of an element

1. padding-top
2. padding-right
3. padding-bottom
4. padding-left

Example

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

Padding - Shorthand Property

```
p {
    padding: 50px 30px 50px 80px;
}
```

Sequence

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

Padding - Shorthand Property

padding: 100px 150px 100px 80px;
Top Right Bottom Left

padding: 100px 150px 100px;
Top Right/Left Bottom

padding: 100px 150px;
Top/bottom Right/left

padding: 100px;
All

CSS Text

TEXT CSS Properties

- color
- Text-alignment: center/inherit/justify/left/right
- text-decoration:blink/inherit/line-through/none/overline/underline
- Text-transform:capitalize/inherit/lowercase/uppercase
- Text Indent
- Letter Spacing
- Line-Height
- Direction:rtl/ltr
- Word-spacing
- Text-shadow: *h-shadow v-shadow color*|none|initial|inherit;

Text Color and Text Align

```
h1 {
      color: green;
   }


h1 {
      text-align: center/left/right;
   }
```

Text Decoration

```
a {
    text-decoration: none;
  }
```

Other Values for Text Decoration

overline

line-through

underline

Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.

```
p {
    text-transform: uppercase;
}
```

- Other values should be

uppercase

lowercase

capitalize

Text Indentation

- The text-indent property is used to specify the indentation of the first line of a text:

```
p {
    text-indent: 50px;
}
```

Output

In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since. 'Whenever you feel.

Letter Spacing & Line Height

```
h1 {
      letter-spacing: 3px;
}


p {
      line-height: 0.8;
   }
```

Text Direction and Word Spacing

- The direction property is used to change the text direction of an element:

```
div {
    direction: rtl;
}
```

The word-spacing property is used to specify the space between the words in a text.

```
h1 {
    word-spacing: 10px;
}
```

Text Shadow

- The text-shadow property adds shadow to text.

The following example specifies the position of the horizontal shadow (3px), the position of the vertical shadow (2px) and the color of the shadow (red):

```
h1 {
    text-shadow: 3px 2px red;
}
```

CSS FONT

CSS Font Family

- The font family of a text is set with the font-family property
- Example

```
p {
    font-family: Times New Roman, Times, serif;
  }
```

Font Style

- The font-style property is mostly used to specify italic text.
- This property has three values:
- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Font Style

- Example

```
p {
   font-style: normal;
}
```

Font Size

```
h1 {
      font-size: 40px;
   }
```

Set Font Size With Em [emphemeral unit]

pixels/16=*em*

e.g. 16pixels is 1em.

Font Weight

- The font-weight property specifies the weight of a font:

```
p {
    font-weight: normal/bold;
}
```

Font Variant

- The font-variant property specifies whether or not a text should be displayed in a small-caps font.
- In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

Font Variant: Example

```
p{
   font-variant: normal;
}

h1 {
   font-variant: small-caps;
}
```

Chapter 5

JS

JAVASCRIPTS

USE of JavaScript

- JavaScript is one of the **3 languages** all web developers **must** learn:

  1. **HTML** to define the content of web pages.
  2. **CSS** to specify the layout of web pages.
  3. **JavaScript** to program the behaviour of web pages.

JavaScript Introduction

- Change HTML Content

```
document.getElementById("demo").innerHTML
  = "Hello JavaScript";
```

- Change HTML Attributes

```
<input type=“button”
onclick="document.getElementById('myImage').src='
pic_bulbon.gif‘ ">
```

JavaScript Introduction

- Change HTML Styles (CSS)

```
document.getElementById("demo").style.fontSize = "25
    px";
```

- Hide HTML Elements

```
document.getElementById("demo").style.display = "
none";
```

JavaScript Introduction

- Show HTML Elements

```
document.getElementById("demo").style.displa
    y = "block";
```

Define

- In HTML, JavaScript code must be inserted between <script> and </script> tags.

- Example

```
<script>
document.getElementById("demo").innerHTML =
"My First JavaScript";
</script>
```

JavaScript in <head> EXAMPLE

```
<html>
<head>
  <script>
  function myFunction()
  {
      document.getElementById("demo").innerHTML = "Paragraph changed.";
  }
  </script>
  </head>
```

```
<body>
<p id="demo">A Paragraph</p>

<input type="button" onclick="myFunction()"
   value="Try it">

</body>
</html>
```

JavaScript in <body>

- In this example, a JavaScript function is placed in the <body> section of an HTML page.
- The function is invoked (called) when a button is clicked:
- Placing scripts at the bottom of the <body> element improves the display speed, because script compilation slows down the display.

- Example:

Next Slide

```
<!DOCTYPE html>
<html>
<body>

<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<input type="button" onclick="myFunction()" value="Try it">

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragrap
h changed.";
}
</script>

</body>
</html>
```

External JavaScript

- External file named: abc.js
- In this file we can directly define function for the particular event.

- Example

```
function myFunction()
{
    document.getElementById("demo").innerHTML
 = “Hello.";
}
```

How to Include External JS

- We can include external Js anywhere in the HTML page like below code:

```
<script src="myScript.js"></script>
```

You can place an external script reference in <head> or <body> as you like.

Note:

External scripts cannot contain <script> tags.

External JavaScript Advantages

- Placing scripts in external files has some advantages:

1. It separates HTML and code.
2. It makes HTML and JavaScript easier to read and maintain.
3. Cached JavaScript files can speed up page loads.

JavaScript Display Possibilities

- JavaScript can "display" data in different ways:

1. Writing into an alert box,
   using **window.alert()**.
2. Writing into the HTML output
   using **document.write()**.
3. Writing into an HTML element,
   using **innerHTML**.
4. Writing into the browser console,
   using **console.log()**.

Using window.alert()

- You can use an alert box to display data:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert(“Hello How are u?”);
</script>

</body>
</html>
```

Using document.write()

- ```
  <!DOCTYPE html>
  <html>
  <body>

  <h1>My First Web Page</h1>
  My first paragraph.

  <script>
  document.write(5 + 6);
  </script>

  </body>
  </html>
  ```
  ```

Using innerHTML

- ```
  <!DOCTYPE html>
  <html>
  <body>

  <h1>My First Web Page</h1>
  My First Paragraph

  <script>
  document.getElementById("demo").innerHTML = 5 + 6;
  </script>

  </body>
  </html>
  ```
  ```

Using console.log()

- <!DOCTYPE html>
  <html>
  <body>

  <h1>My First Web Page</h1>
  My first paragraph.

  <script>
  console.log(5 + 6);
  </script>

  </body>
  </html>

JavaScript Statements

- In HTML, JavaScript statements are "instructions" to be "executed" by the web browser.

  document.getElementById("demo").innerHTML = "Hello Dolly.";

JavaScript Programs

- The statements are executed, one by one, in the same order as they are written.
- In this example x, y, and z are given values, and finally z is displayed:
- Example
- var x, y, z;
  x = 5;
  y = 6;
  z = x + y;
  document.getElementById("demo").innerHTML = z;

JavaScript Identifiers

- The general rules for constructing names for variables (unique identifiers) are:

1. Names can contain letters, digits, underscores, and dollar signs.
2. Names must begin with a letter
3. Names can also begin with $ and _
4. Names are case sensitive (y and Y are different variables)
5. Reserved words (like JavaScript keywords) cannot be used as names

JavaScript Data Types

- In programming, text values are called text strings.
- JavaScript can handle many types of data, but for now, just think of numbers and strings.
- Strings are written inside double or single quotes. Numbers are written without quotes.
- If you put a number in quotes, it will be treated as a text string.

Example

```
var x = 3;  //this is int variable
var y = 3.2 //this is float variable
var a = “Hello"; //this is string variable
var b = ‘Hello !'; //this is string variable
```

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).
- Example

```
function myFunction(p1, p2)
{
    return p1 * p2;              // The function returns the product of p1 and p2
}
```

JavaScript Function Syntax

- function *name*(*parameter1, parameter2, parameter3*)

{

code to be executed

}

Function Invocation

- The code inside the function will execute when "something" **invokes** (calls) the function:
- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Return

- When JavaScript reaches a **return statement**, the function will stop executing.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- Functions often compute a **return value**.

Example

- var x = abc(4, 3); // Function is called, return value will end up in x

```
function abc(a, b)
{
    return a * b;
}
```

JAVASCRIPT OPERATORS

JavaScript Operators

- Assignment Operator: it is used to assign the value to variable from right to left.
- We can also apply shorthand operator like C.

Example:

```
var x = 5;        // assign the value 5 to x
var y = 2;        // assign the value 2 to y
var z = x + y;    // assign the value 7 to z
```

JavaScript Arithmetic Operators

Operators	Description
+	Addtion
-	Subtraction
*	Multiplication
/	Division
%	Modulo
++	Increment
--	Decrement

JavaScript String Operators

- The + operator can also be used to add (concatenate) strings.
- Example

```
txt1 = “Hello";
txt2 = “Friends";
txt3 = txt1 + " " + txt2;
```

Adding Strings and Numbers

- Adding two numbers, will return the sum, but adding a number and a string will return a string:

  x = 2 + 5; // output: 7
  y = “2" + 5; // output: 25
  z = "Hi" + 5; // output: Hi5

JavaScript Comparison Operators

Operators	Description
==	Equal
!=	No equal
>	Greater than
>=	Greater than equal to
<	Less than
<=	Less than equal to
? :	ternary

```
<script>
var x = 5;
document.getElementById("demo").innerHTML
   = (x == 8);
document.getElementById("demo").innerHTML
   = (x != 8);
document.getElementById("demo").innerHTML
   = (x > 8);
document.getElementById("demo").innerHTML
   = (x < 8);
document.getElementById("demo").innerHTML
   = (x >= 8);
</script>
```

JavaScript Logical Operators

Operators	Description
&&	Logical AND
||	Logical OR
!	Logical Not

```
<script>
var x = 6;
var y = 3;

document.getElementById("demo").innerHTML
   =
(x < 10 && y > 1) + "<br>" +
(x < 10 && y < 1);
</script>
```

```
<script>
var x = 6;
var y = 3;

document.getElementById("demo").innerHTML
   =
(x == 5 || y == 5) + "<br>" +
(x == 6 || y == 5) + "<br>" +
(x == 6 || y == 3);
</script>
```

```
<script>
var x = 6;
var y = 3;

document.getElementById("demo").innerHTML
   =
!(x == y) + "<br>" +
!(x > y);
</script>
```

JavaScript Type Operators

Operators	Description
typeof	Returns type of variable
instanceof	Returns true if an object is an instance of an object type

JavaScript Bitwise Operators

Operators	Description
&	AND
|	OR
~	NOT
^	XOR
<<	Left shift
>>	Right Shift

JavaScript Arrays

- JavaScript arrays are written with square brackets.
- Array items are separated by commas.
- Example

```
<script>
var a = [“rajan",“suresh",“ramesh"];
document.getElementById(“abc").innerHTML =
   a[1];
</script>
```

JavaScript Objects

- JavaScript objects are written with curly braces.
- Object properties are written as name:value pairs, separated by commas.
- Example

```
var person = {firstName:“Hiren",
   lastName:“Mer", age:29};
```

```
<html>
<body>
<p id=“abc"></p>
<script>
var person = {
   firstName : “HIREN",
   lastName  : “MER",
   age : 30
};
document.getElementById(“abc").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>
</body>
</html>
```

JavaScript - if...else Statement

- In JavaScript we have the following conditional statements:

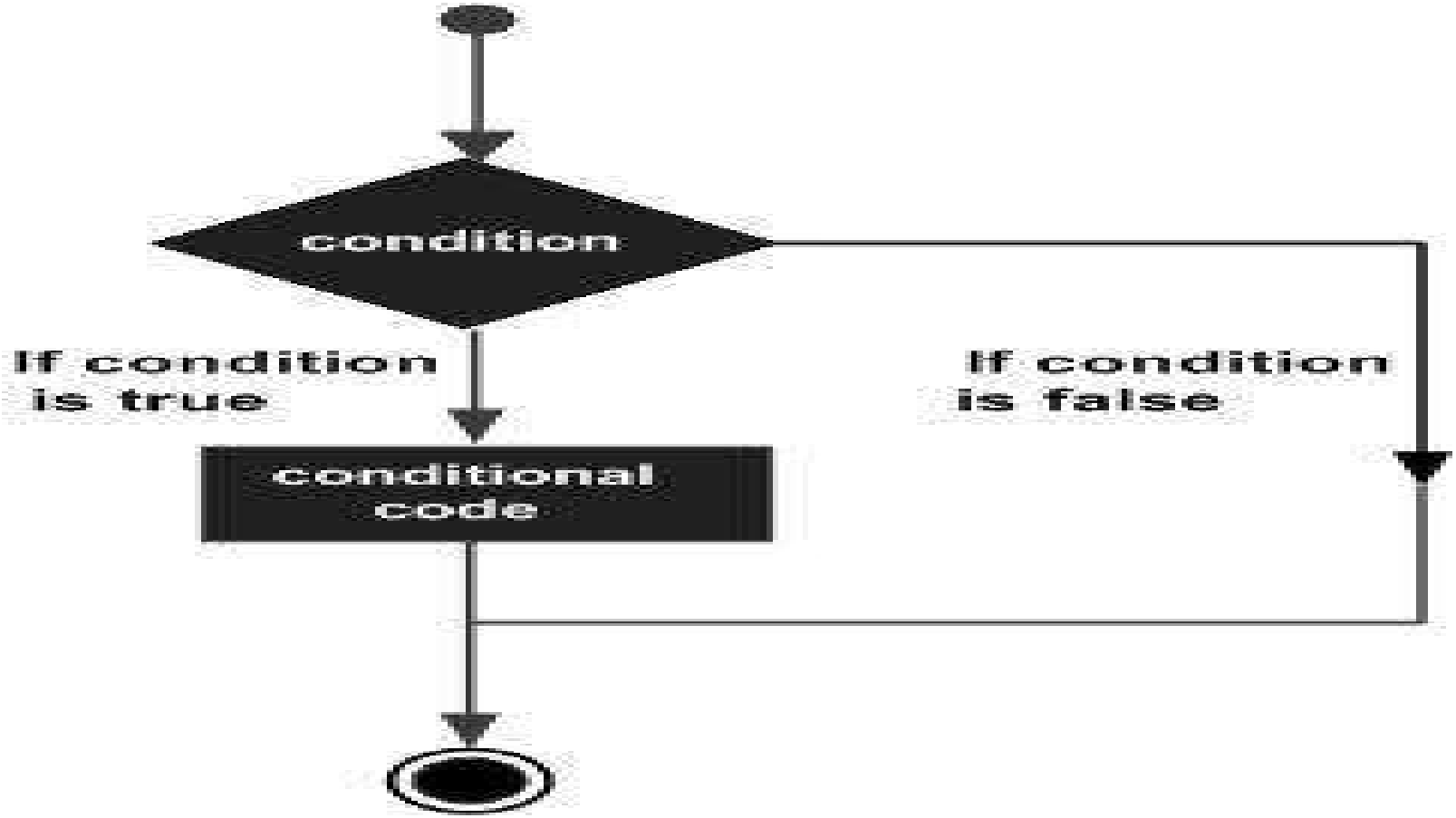
1. Use **if** to specify a block of code to be executed, if a specified condition is true
2. Use **else** to specify a block of code to be executed, if the same condition is false
3. Use **else if** to specify a new condition to test, if the first condition is false
4. Use **switch** to specify many alternative blocks of code to be executed

The if Statement

- Syntax

```
if (condition)
 {
      Statement-x  // if true
   }
```

Flow Chart of IF



If Example

```
<body>
<h1 id=“abc”></h1>
<script>
Var x=10;
if (x < 18)
 {
 document.getElementById(“abc").innerHTML = "Good
    day!";
}
</script>
</body>
```

The if..else Statement

```
if (condition)
 {
      statement-x  //if condition true
   }
 else
{
      statement-y  //if condition false
   }
```

```
<body>
<p>Click the button to display greeting:</p>

<input type=“button” onclick="myFunction()“ value=“click Me”>

<p id=“abc"></p>
<script>
function myFunction() {
    var x;  var greeting;
    if (x < 18)
    {
        greeting = "Good day";
    }
   else
            {
        greeting = "Good evening";
    }
    document.getElementById(“abc").innerHTML = greeting;
}
</script>
<body>
```

The else if Statement

- Syntax

if (*condition1*) {
    statement-x // if true
} else if (*condition2*) {
    statement-y // if true
} else {
    statement-z // if all condition false
}

-

- Example

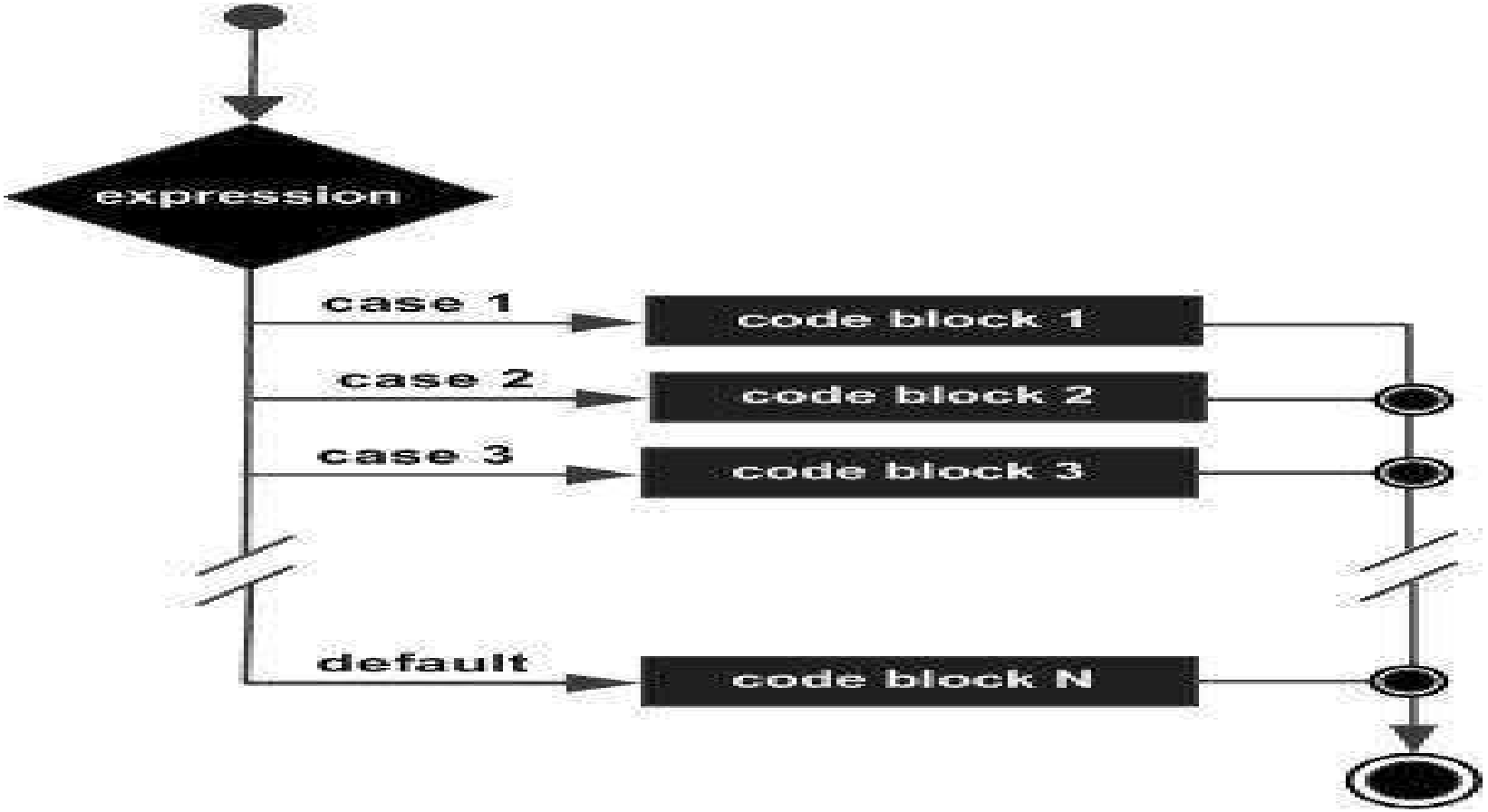
```
if (time < 10)
 {
      greeting = "Good morning";
   }
else if (time < 20) {
      greeting = "Good day";
   }
else {
      greeting = "Good evening";
   }
```

JavaScript Switch Statement

- Use the switch statement to select one of many blocks of code to be executed.
- Syntax

```
switch(expression) {
    case n:
        code block
        break;
    case n:
        code block
        break;
    default:
        code block
}
```

Flow Chart of SWITCH case



Example:

```
<script>
Var day=5;
switch (day) {
    case 0:
        day = "Sunday";
        break;
    case 1:
        day = "Monday";
        break;
    case 2:
        day = "Tuesday";
        break;
```

```
case 3:
      day = "Wednesday";
      break;
case 4:
     day = "Thursday";
     break;
  case 5:
     day = "Friday";
     break;
  case  6:
     day = "Saturday“;
```

```
default:
        window.alert(“Invalid input”);
        break;
}

document.getElementById(‘def’).innerHTML =
   "Today is " + day;
</script>
```

Javascript Loop

Different Kinds of Loops

- JavaScript supports different kinds of loops:

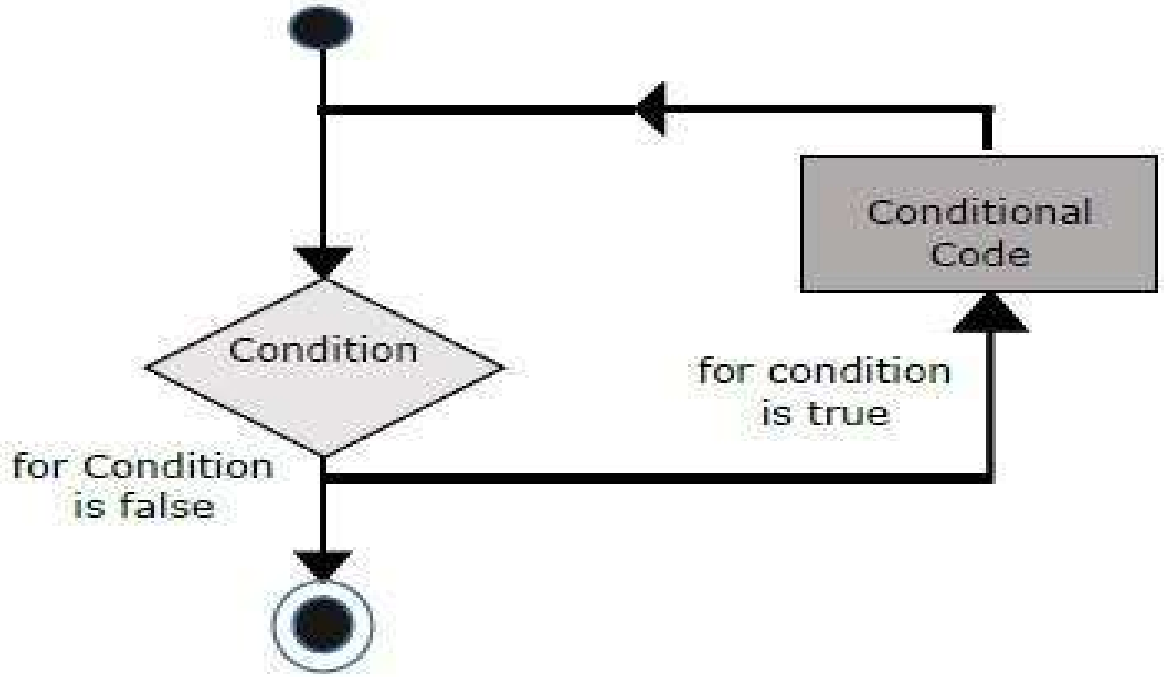
1. **for** - loops through a block of code a number of times
2. **for/in** - loops through the properties of an object
3. **while** - loops through a block of code while a specified condition is true
4. **do/while** - also loops through a block of code while a specified condition is true

JavaScript For Loop

- Loops, if you want to run the same code over and over again, each time with a different value.
- Syntax

```
for (initialization; condition;increrent/decrement)
 {
     loop body;
   }
```

Flow chart of FOR loop



Example

```
<script>
var a = [“rajan", “ramesh", “Suresh”];
var  x = "";
var  i;
for (i = 0; i < a.length; i++)
 {
    x =x + cars[i] + "<br>";
}
document.getElementById(“test").innerHTML = x;
</script>
```

The For/In Loop

- Syntax

```
for (var in object)
 {
      body of loop;
   }
```

- Example1

```
var person =
{fname:“Kapil", lname:“Sharma", age:25};

   var text = "";
   var x;
   for (x in person) {
       text = text + person[x];
   }
```

- Example 2

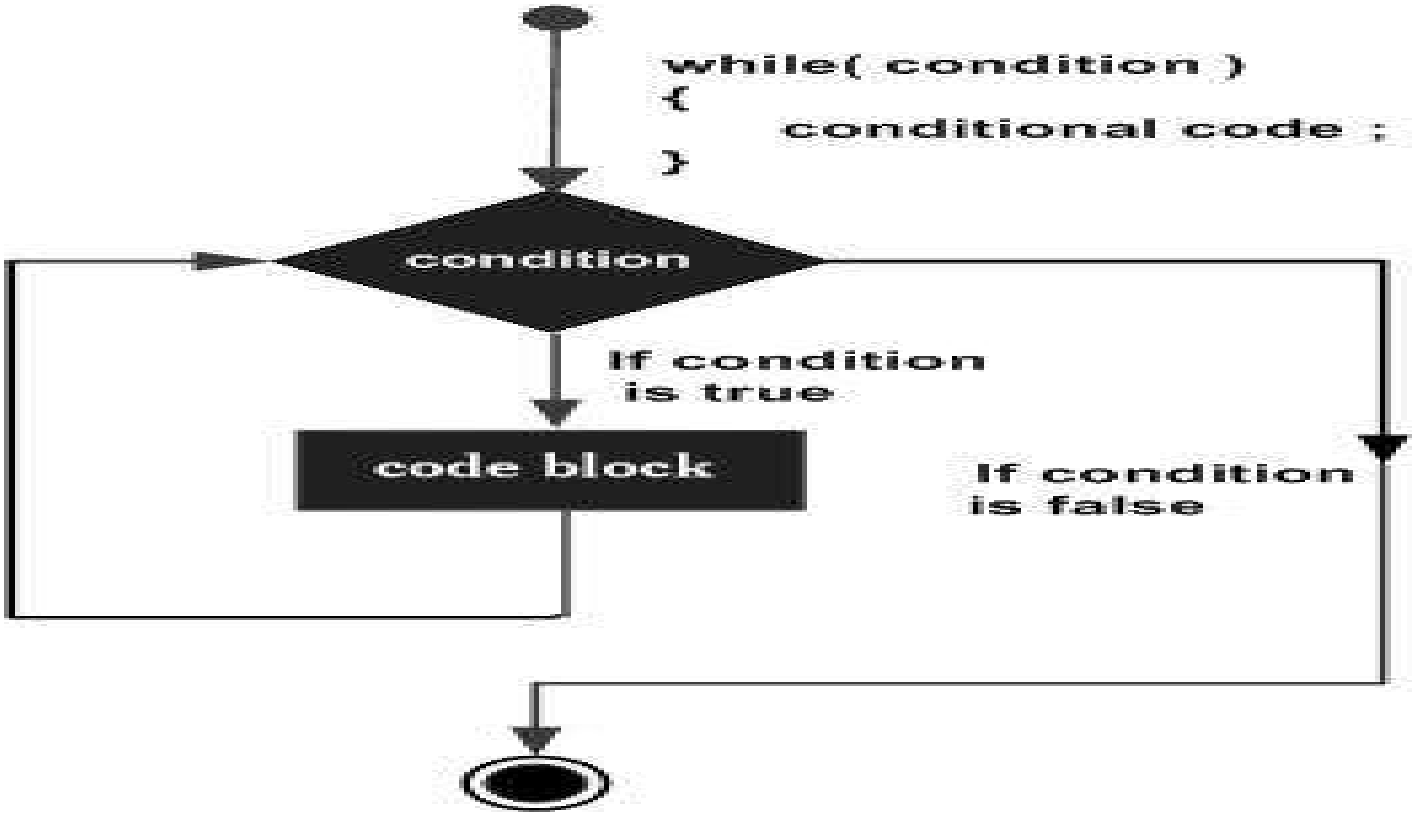
```
<script>
var cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat",
    "Audi"];
var text = "";
var i;
for (i in cars) {
    text = text + cars[i] + "<br>";
}
document.getElementById(“abc").innerHTML = text;
</script>
```

The While Loop

- Syntax

```
while (condition)
 {
      code block to be executed
   }
```

Flow Chart of While Loop



- Example

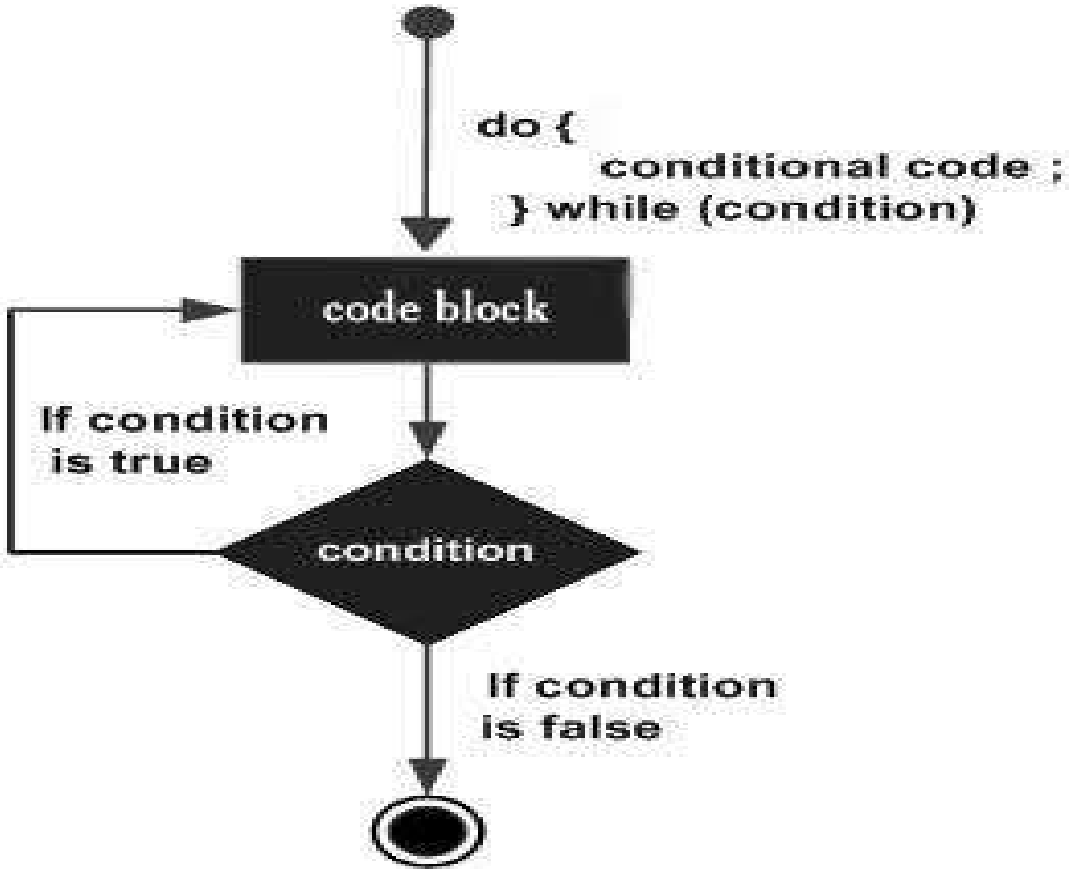
```
<h1>JavaScript while</h1>

<p id=“abc"></p>

<script>
var text = "";
var i = 0;
while (i < 10) {
    text = text + "<br>The number is " + i;
    i++;
}
document.getElementById(“abc").innerHTML = text;
</script>
```

The Do/While Loop

- Syntax

```
do {
    code block to be executed
}
while (condition);
```

Flow chart of Do While Loop



- Example

```
do {
      text  = text + "The number is " + i;
      i++;
   }
   while (i < 10);
```

JavaScript - Page Redirection

- You can redirect the page using click event and automatiocally by settimeout function.

- Using window.location function we can redirect page to respective page which you want.

```
window.location="http://www.google.com";
```

- Example

```
<script type="text/javascript“>
 function redirect() {
   window.location="http://www.google.com";
 }
 </script>
<p>Click the This button</p>
 <input type="button" value="Redirect Me"
   onclick=“redirect();" />
```

Redirect using setTimeout

- We can call any function using settimeout built-in function.

setTimeout(‘functionname', 10000);

10000 =10 seconds

Java script POP UP Boxes

- Alert Dialog Box

  alert (message);

- Confirmation Dialog Box

  confirm (message);

- Prompt Dialog Box

  prompt (message);

Chapter 5 PART 2

Advance JAVASCRIPTS

JavaScript and Object

We can Create the Object for the Javascript

```
var person = {
    firstName:“Kapil ",
    lastName:“Sharma",
    age:30,

};
```

With new keyword

```
var person=new Object ();
person.firstName = “Kapil ";
person.lastName = “Sharma";
person.age = 30;
```

Different Examples

```
var x1 = new Object();   // A new Object object
var x2 = new String();   // A new String object
var x3 = new Number();   // A new Number object
var x4 = new Boolean();  // A new Boolean object
var x5 = new Array();    // A new Array object
var x6 = new Function(); // A new Function object
var x7 = new Date();     // A new Date object
```

Object Examples

```
<html>
<head>
 <title>User-defined objects</title>
<script type="text/javascript">
var book = new Object();
book.subject = “WT";
book.author = "MT Savaliya"; </script> </head>
<body>
 <script type="text/javascript">
 document.write("Book name is : " + book.subject + "<br>");
document.write("Book author is : " + book.author + "<br>");
</script>
</body>
</html>
```

```
<html>
<head>
<title>User-defined objects</title>
<script type="text/javascript">
 function book(title, author)
{ this.title = title; this.author = author; } </script>
</head>
 <body>
<script type="text/javascript">
var myBook = new book(“WT", "MT savaliya");
document.write("Book title is : " + myBook.title + "<br>");
document.write("Book author is : " + myBook.author + "<br>");
</script>
</body>
 </html>
```

```
<html>
<head>
 <title>User-defined objects</title>
<script type="text/javascript">
function addPrice(amount)
{    this.price = amount;    }
 function book(title, author)
{
 this.title = title;
 this.author = author;
 this.addPrice = addPrice;
}
 </script>
 </head>
```

```
<body>
 <script type="text/javascript">
 var myBook = new book(“WT", "MT Savaliya");
myBook.addPrice(100);
document.write("Book title is : " + myBook.title + "<br>");
document.write("Book author is : " + myBook.author + "<br>");
document.write("Book price is : " + myBook.price + "<br>");
</script>
 </body> </html>
```

Javascript Date Object

- This Object returns today's date and time.

```
<script type="text/javascript">
 var dt = Date();
 document.write("Date and Time : " + dt );
</script>
```

Output

Date and Time : Tue Feb 07 2017 09:33:34 GMT-0800 (Pacific Standard Time)

getDate()

```
<html>
 <head>
 <title>JavaScript getDate Method</title>
</head>
<body>
 <script type="text/javascript">
var dt = new Date("December 25, 1995 23:15:00");
    document.write("getDate() : " + dt.getDate() );
</script>
</body>
</html>
```

Date Methods

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Javascript Validation

- Email Validation

/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/

- Check All letters

/^[A-Za-z]+$/;

- Check All Numbers

/^[0-9]+$/;

- Check Floating point number

/^[-+]?[0-9]+\.[0-9]+$/;

Javascript Validation

- yyyy-mm-dd

/^(\d{4})-(\d{1,2})-(\d{1,2})$/

Credit card

/^(\d{4})-(\d{4})-(\d{4})-(\d{4})$/

/^(?:5[1-5][0-9]{14})$/;.

Phone number

/^\d{10}$/;

JAVASCRIPT DOM

DOCUMENT OBJECT MODEL

JAVASCRIPT DOM

- What is the DOM?
- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:

  "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."
- The W3C DOM standard is separated into 3 different parts:

1. Core DOM - standard model for all document types
2. XML DOM - standard model for XML documents
3. HTML DOM - standard model for HTML documents

What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements
- In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

JavaScript - HTML DOM Methods

- HTML DOM methods are **actions** you can perform (on HTML Elements).
- HTML DOM properties are **values** (of HTML Elements) that you can set or change.
- In the DOM, all HTML elements are defined as **objects**.
- The programming interface is the properties and methods of each object.
- A **property** is a value that you can get or set (like changing the content of an HTML element).
- A **method** is an action you can do (like add or deleting an HTML element).

For Example

```
<p id="demo"></p>

    <script>
    document.getElementById("demo").innerHTML = "Hello
    World!";
    </script>
```

- Here getElementById is a **method**, while innerHTML is a **property**.

The HTML DOM Document Object

- The document object represents your web page.
- If you want to access any element in an HTML page, you always start with accessing the document object.
- Here we have some examples of how you can use the document object to access and manipulate HTML.

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Adding Events Handlers

Method	Description
document.getElementById(*id*).onclick = function(){*code*}	Adding event handler code to an onclick event

JavaScript HTML DOM Elements

- Finding HTML Elements

There are some of ways to do this:

1. Finding HTML elements by id
2. Finding HTML elements by tag name
3. Finding HTML elements by class name
4. Finding HTML elements by CSS selectors
5. Finding HTML elements by HTML object collections

Finding HTML Element by Id and

```
var myElement =
   document.getElementById("intro");
```

```
<input type=“text” id=“intro”/>
```

Here we can get the values from any element by getElementById method.

Finding HTML Elements by Tag Name

```
<p>Hello World!</p>
<p>The DOM is very useful.</p>
<p>This example demonstrates the </p>
<p id="demo"></p>
<script>
var x = document.getElementsByTagName("p");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) is: ' + x[0].innerHTML;
</script>
```

Finding HTML Elements by Class Name

```
<p>Hello World!</p>
<p class="intro">The DOM is very useful.</p>
<p class="intro">This example</p>
<p id="demo"></p>
<script>
var x = document.getElementsByClassName("intro");
document.getElementById("demo").innerHTML =
'The first paragraph with class="intro": ' + x[0].innerHTML;
</script>
```

Finding HTML Elements by CSS Selectors

```
<p>Hello World!</p>
<p class="intro">The DOM is very useful.</p>
<p class="intro">This example demonstrates the <b>querySelectorAll</b>
    method.</p>
<p id="demo"></p>
<script>
var x = document.querySelectorAll("p.intro");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) with class="intro": ' +
    x[0].innerHTML;
</script>
```

JavaScript HTML DOM Events

- A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.
- To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

onclick=*JavaScript*

List of Events

- Examples of HTML events:

1. When a user clicks the mouse
2. When a web page has loaded
3. When an image has been loaded
4. When the mouse moves over an element
5. When an input field is changed
6. When an HTML form is submitted
7. When a user strokes a key

Examples on Click

```
<h1 onclick="changeText(this)">Click on this
   text!</h1>

   <script>
   function changeText(id) {
      id.innerHTML = "Ooops!";
   }
   </script>
```

Examples on Click

```
<input type=“button” onclick="displayDate()“ value=“change”>

<script>
function displayDate()
{
   document.getElementById("demo").innerHTML = Date();
}
</script>

<p id="demo"></p>
```

Examples on Click

```
<input type=“button” id=“mybtn” value=“change”>
<p id="demo"></p>
<script>
document.getElementById("myBtn").onclick = displayDate;
function displayDate()
 {
   document.getElementById("demo").innerHTML = Date();
}
</script>
```

Body Onload Example

```
<body onload=“displayDate ()”>
<p id="demo"></p>
<script>
function displayDate()
 {
   document.getElementById("demo").innerHTML = Date();
}
</script>
</body>
```

The onchange Event Example

```
<head>
<script>
    function myFunction() {
      var x = document.getElementById("fname");
      x.value = x.value.toUpperCase();
    }
</script>
</head>
<body>
Enter your name:
 <input type="text" id="fname" onchange="myFunction()">
```

The onmouseover and onmouseout Events

```
<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-color:#D94A38;width:120px;height:20px;padding:40px;">
Mouse Over Me</div>

<script>
function mOver(obj) {
   obj.innerHTML = "Thank You"
}

function mOut(obj) {
   obj.innerHTML = "Mouse Over Me"
}
</script>
```

The onmousedown, onmouseup and onclick Events

```
<div onmousedown="mDown(this)" onmouseup="mUp(this)"
style="background-color:#D94A38;width:90px;height:20px;padding:40px;">
Click Me</div>

<script>
function mDown(obj) {
    obj.style.backgroundColor = "#1ec5e5";
    obj.innerHTML = "Release Me";
}

function mUp(obj) {
    obj.style.backgroundColor="#D94A38";
    obj.innerHTML="Thank You";
}
</script>
```

Events

- Input Events
  1. onblur - When a user leaves an input field
  2. onchange - When a user changes the content of an input field
  3. onchange - When a user selects a dropdown value
  4. onfocus - When an input field gets focus
  5. onselect - When input text is selected
  6. onsubmit - When a user clicks the submit button
  7. onreset - When a user clicks the reset button
  8. onkeydown - When a user is pressing/holding down a key
  9. onkeypress - When a user is pressing/holding down a key
  10. onkeyup - When the user releases a key

Mouse Events

1. onmouseover/onmouseout - When the mouse passes over an element
2. onmousedown/onmouseup - When pressing/releasing a mouse button
3. onmousedown - When mouse is clicked: Alert which element
4. onmousedown - When mouse is clicked: Alert which button
5. onmousemove/onmouseout - When moving the mouse pointer over/out of an image
6. onmouseover/onmouseout - When moving the mouse over/out of an image

Click Events

- onclick - When button is clicked
  ondblclick - When a text is double-clicked

Load Events

- onload - When the page has been loaded

  onload - When an image has been loaded

  onerror - When an error occurs when loading an image

  onunload - When the browser closes the document

  onresize - When the browser window is resized

Event Listener

HTML DOM EventListener

- Syntax

  document.addEventListener(*event*, *function*)

- The document.addEventListener() method attaches an event handler to the document.

Example

- document.addEventListener("click", function()
  {
  document.getElementById("demo").innerH
  TML = "Hello World";
  });

Example

- *element*.addEventListener("click", myFunction);

```
function myFunction() {
    alert ("Hello World!");
}
```

Multiple Event Listener

```
<script>
var x = document.getElementById("myBtn");
x.addEventListener("click", myFunction);
x.addEventListener("click", someOtherFunction);

function myFunction() {
   alert ("Hello World!");
}
function someOtherFunction() {
   alert ("This function was also executed!");
}
</script>
```

```
<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>
var x = document.getElementById("myBtn");
x.addEventListener("mouseover", myFunction);
x.addEventListener("click", mySecondFunction);
x.addEventListener("mouseout", myThirdFunction);

function myFunction() {
   document.getElementById("demo").innerHTML += "Moused over!<br>";
}

function mySecondFunction() {
   document.getElementById("demo").innerHTML += "Clicked!<br>";
}

function myThirdFunction() {
   document.getElementById("demo").innerHTML += "Moused out!<br>";
}
</script>
```

Add an Event Handler to the Window Object

- The addEventListener() method allows you to add event listeners on any HTML DOM object such as HTML elements, the HTML document, the window object, or other objects that support events.

Example

```
<p id="demo"></p>
<script>
window.addEventListener("resize", function(){
 document.getElementById("demo").innerHTML
   = Math.random();
});
</script>
```

The removeEventListener() method

- *We can remove the event listener which we created by addeventlistener method.*

element.removeEventListener("mousemove", myFunction);

MATH OBJECT

- Math.PI // returns 3.141592653589793
- Math.round()

  Example

  Math.round(4.7); // returns 5

- Math.pow()
- Math.pow(8, 2); // returns 64

MATH OBJECT

- Math.sqrt()

Math.sqrt(64); // returns 8

- Math.abs()

Math.abs(-4.7); // returns 4.7

- Math.ceil()

Math.ceil(4.4); // returns 5

- Math.floor()

Math.floor(4.7); // returns 4

MATH OBJECT

- Math.min() and Math.max()

```
Math.min(0, 150, 30, 20, -8, -200); // returns -200
Math.max(0, 150, 30, 20, -8, -200); // returns 150
```

- Math.random()

```
Math.random();     // returns a random number
```

Math.random() returns a random number between 0 and 1

DATE Object

- <script>
  var d = new Date();
  document.getElementById("demo").innerHT
  ML = d. getDay();
  </script>

DATE Methods

getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

DHTML
Dynamic HTML

Document with
HTML,CSS,JAVASCRIPT

DHTML

- DHTML stands for **D**ynamic **HTML**.
- DHTML is NOT a language or a web standard.
- DHTML is a TERM used to describe the technologies used to make web pages dynamic and interactive.
- DHTML means the combination of HTML, JavaScript, DOM, and CSS.
- According to the World Wide Web Consortium (W3C): *"Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."*

Thank you So much