JavaScript

- **JavaScript** (JS) is a lightweight, interpreted programming language
- It is most well-known as the scripting language for Web pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation).
- JavaScript was designed to add interactivity and programming to HTML web pages.
    - Performs calculations such as totaling the price or computing sales tax.
    - Verifies data such as with credit card applications.
    - Adapts the display to user needs.
- JavaScript is case sensitive.
- JavaScript is object-based language as it provides predefined objects.
- JavaScript is both dynamically and loosely typed language.
    - "Loosely typed" means language does not bother with types too much, and does conversions automatically.
    - In dynamic typing, types are associated with values not variables.
    - "Dynamically typed language" -- you do not need to declare variable type before use.

<u>Limitations</u>

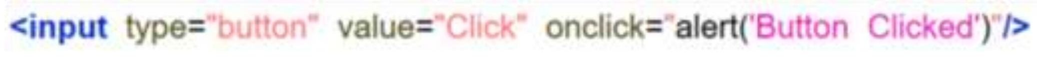
- Client-side JavaScript does not allow the reading or writing of files.
- It cannot be used for networking applications because there is no such support available.
- It doesn't have any multithreading or multiprocessor capabilities.

<u>Uses</u>

- Client-side validation
- Dynamic drop-down menus
- Displaying date and time
- Validate user input in an HTML form before sending the data to a server.
- Build forms that respond to user input without accessing a server.
- Change the appearance of HTML documents and dynamically write HTML into separate Windows.
- Open and close new windows or frames.
- Manipulate HTML "layers" including hiding, moving, and allowing the user to drag them around a browser window.
- Build small but complete client side programs .
- Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)
- Displaying clocks etc.

<u>Embedding JavaScript</u>

- JavaScript scripts are embedded, either directly or indirectly, in XHTML documents.

1. Between the <body> </body> tag of html (Inline JavaScript)
    - When java script was written within the html element using attributes related to events of the element then it is called as inline java script.
    - Example: 

```
<input type="button" value="Click" onclick="alert('Button Clicked')"/>
```

2. Between the <head> </head> tag of html (Internal JavaScript)
   - When java script was written within the head section using <script> element then it is called as internal java script.
   - Example: <script type = “text/javascript” > </script>

3. In .js file (External JavaScript)
   - <script type = “text/javascript” src = “file.js” > </script>
   - It separates HTML code from JavaScript code
   - It makes HTML and JavaScript easier to read and maintain
   - Cached JavaScript files can speed up page loads

JavaScript Comments

- Use // for single line comment and use /* */ for multiline comment

Variable Declaration

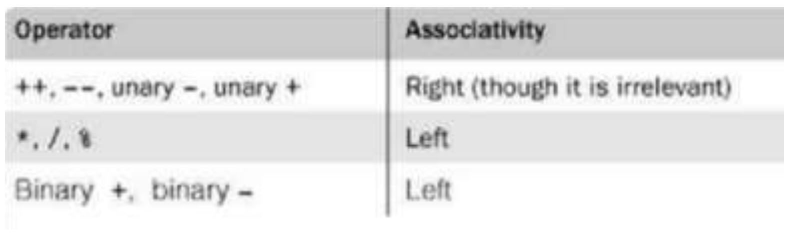
- In JavaScript variables are declared using keyword ‘**var**’.

Primitives, Operations & Expressions

- JavaScript has five primitive types: **Number, String, Boolean, Undefined, and Null**
- A **Boolean** represents only one of two values: true, or false.
  - Var a=true; var b=false;
- There is only one type of **Number** in JavaScript. Numbers can be written with or without a decimal point.
  - var x1 = 34.00; // Written with decimals
  - var x2 = 34; // Written without decimals
- In JavaScript, a variable without a value, has the value **undefined**
  - var data; (if you print the value of data the output will be - undefined)
- In JavaScript **null** is "nothing". It is supposed to be something that doesn't exist.
- **Strings** are written with quotes. You can use single or double quotes:
  - var name1= "Volvo XC60"; // Using double quotes
  - var name2 = 'Volvo XC60'; // Using single quotes

✓ **JavaScript has dynamic types. This means that the same variable can be used to hold different data types**
Var a=5; var a=”scorpio” ; var a=true;

Operators



Operator	Associativity
++, --, unary -, unary +	Right (though it is irrelevant)
*, /, %	Left
Binary +, binary -	Left

Logical operators

❖ && Logical and
❖ | | Logical or
❖ ! Logical not

Comparison operators

- == Equal to
- === Equal value and equal type
- != Not equal
- !== Not equal value or not equal type
- > Greater than
- < Less than
- >= Greater than or equal to
- <= Less than or equal to

Implicit conversions(coersions)

Operation	**Output**	**remarks**
3+2=	5	Integer addition
"3"+2=	32	String concatenation
"4"+"2"	42	String concatenation
"4"-"2"	2	subtraction
"one"+2	One2	String concatenation

Explicit conversion

toString() method: The toString() method returns a number as a string.

Example: var cost=150;

```
Var str_cost=cost.toString();
```

Number() method: The Number() method converts a string to a number.

Example: var str="42";

```
Var num=Number(str);
```

parsefloat() method: The parseFloat() method converts a string into a point number (a number with decimal points). You can even pass in strings with random text in them.

Example: var text = '3.14someRandomStuff';

```
var pointNum = parseFloat(text);
// returns 3.14
```

parseInt() method: The parseInt() method converts a string into an integer (a whole number).

Example: var text = '42px';

```
var integer = parseInt(text, 10);
// returns 42
```

Window and Document

- JavaScript models the XHTML document with the Document object.
- The Document object represents the document being displayed using DOM
- The window in which the browser displays an XHTML document is modeled with the Window object
- The Window object represents the window in which the document containing the script is being displayed
- The Window object is the default object for JavaScript, so properties and methods of the Window object may be used without qualifying with the class name
- The **write() method of document object** is used to create XHTML code.
- The write() method writes HTML expressions or JavaScript code to a document.
- The write() method is mostly used for testing: If it is used after an HTML document is fully loaded, it will delete all existing HTML.
- The parameter of write can include any XHTML tags and content.

  Example: var a=42;

  Document.write("the value of a is ",a,"<br />");

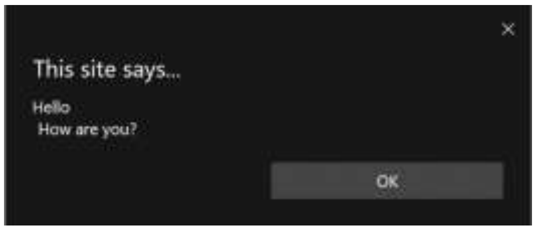
❖ Window includes three methods that create dialog boxes for three specific kinds of user interactions.

- ➢ Alert() method
  - o The alert() method displays an alert box(that can be closed) with a specified message (string)and an OK button.
  - o The string parameter of alert is plain text.
  - o Alert() box prevents the user from accessing other parts of the page until the box is closed.
  - o Syntax: alert(message);
  - o Example:

    alert("Hello\n How are you?");

    var sum=5;

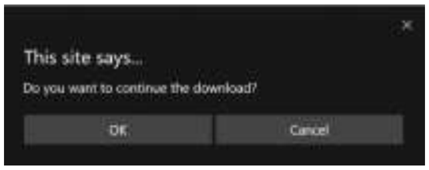
    alert("The sum is:" + sum + "\n");



This site says...

Hello
 How are you?

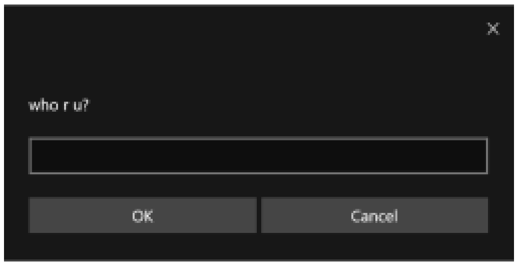
- ➢ Confirm() method
  - o The confirm() method displays a dialog box with a specified message, along with an OK and a Cancel button.
  - o A confirm box is often used if you want the user to verify or accept something.
  - o The confirm() method returns true if the user clicked "OK", and false otherwise.
  - o Example:
    confirm("Press a button!\n Either OK or Cancel.");
    var question = confirm("Do you want to continue the download?");



This site says...

Do you want to continue the download?

- Prompt() method
  - The prompt() method displays a dialog box that prompts the visitor for input.
  - A prompt box is often used if you want the user to input a value.
  - When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
  - The prompt() method returns the input value if the user clicks "OK". If the user clicks "cancel" the method returns null.
  - If the user clicks OK without entering any text, an empty string is returned.
  - Example:
    Var name = prompt("What is your name?", " ");



who r u?

Strings

- The JavaScript string is an object that represents a sequence of characters.
- The String object includes one property, length, which gives number of characters in a string
- Example:

  Var str="hello world";

  Var len=str.length; // len=11

String methods

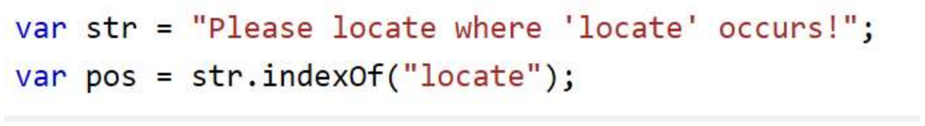
1. charAt() : Returns the character at the specified index (position)
   example:

   var str = "George";

   str.charAt(2) is 'o'

2. indexOf(): Returns the position of the first found occurrence of a specified value in a string.
   Example:
   var str = "George";

   str.indexOf('r') is 3



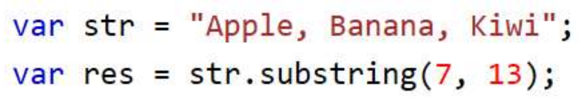
```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
```

pos is 7

3. substring():Returns substring of the String object, between two specified indices

   example: var str = "George";

   str.substring(2, 4) is 'org'



```
var str = "Apple, Banana, Kiwi";
var res = str.substring(7, 13);
```

res=banana

4. toLowerCase() & toUpperCase: convert uppercase letters to lowercase and vice-versa.

Example: str s1="abc";
Str s2=s1.toUpperCase(); //s2 becomes ABC ,similar for toLowerCase()

5. slice() method : slice() extracts a part of a string and returns the extracted part in a new string.

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(7, 13);
```

res=Banana

6. replace() method: to replace part of a string with another string. This replace() method returns a new string, it doesn't affect the original string that will remain unchanged.
   Syntax : str.replace(regexp|substr, newSubstr).
   Example: var str = "Color red looks brighter than color blue.";
   var result = str.replace("color", "paint");
   alert(result); // Outputs: Color red looks brighter than paint blue.
7. split() method: The split() method can be used to splits a string into an array of strings, using the syntax str.split(separator, limit).
   The seperator argument specifies the string at which each split should occur, whereas the limit arguments specifies the maximum length of the array.
   Example: var fruitsStr = "Apple, Banana, Mango, Orange, Papaya";
   var fruitsArr = fruitsStr.split(", ",2); //output: Apple,Banana

   To split a string into an **array of characters**, specify an empty string ("") as a separator.
   Example: var str = "INTERSTELLAR";
   var strArr = str.split(""); // strArr[0] is I, strArr[1] is N etc

Arrays

- Arrays are complex variables that store more than one value or a group of values under a single variable name.
- JavaScript arrays can store any valid value, including strings, numbers, objects, functions, and even other arrays, thus making it possible to create more complex data structures such as an array of objects or an array of arrays.
- Example:

```
var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];
document.write(fruits[0]); // Prints: Apple
document.write(fruits[1]); // Prints: Banana
```

OR

```
var fruits = new Array("Apple", "Banana", "Mango", "Orange", "Papaya");
```

[In JavaScript, arrays are really just a special type of objects which has numeric indexes as keys. The typeof operator will return "object" for arrays.]

- The length property returns the length of an array, which is the total number of elements contained in the array. (e.g: alert(fruits.length); //output=5)

- Printing array elements sequentially
  - For loop:
    Example:
    ```
    for(var i = 0; i < fruits.length; i++)
    {
        document.write(fruits[i] + "<br>"); // Print array element

    }
    ```
  - For-of loop:
    Example:
    ```
    for(var fruit of fruits)
    {
        document.write(fruit + "<br>"); // Print array element

    }
    ```
  - For-in loop:
    Example:
    ```
    for(var i in fruits)
    {
        document.write(fruits[i] + "<br>"); //not recommended
    }
    ```

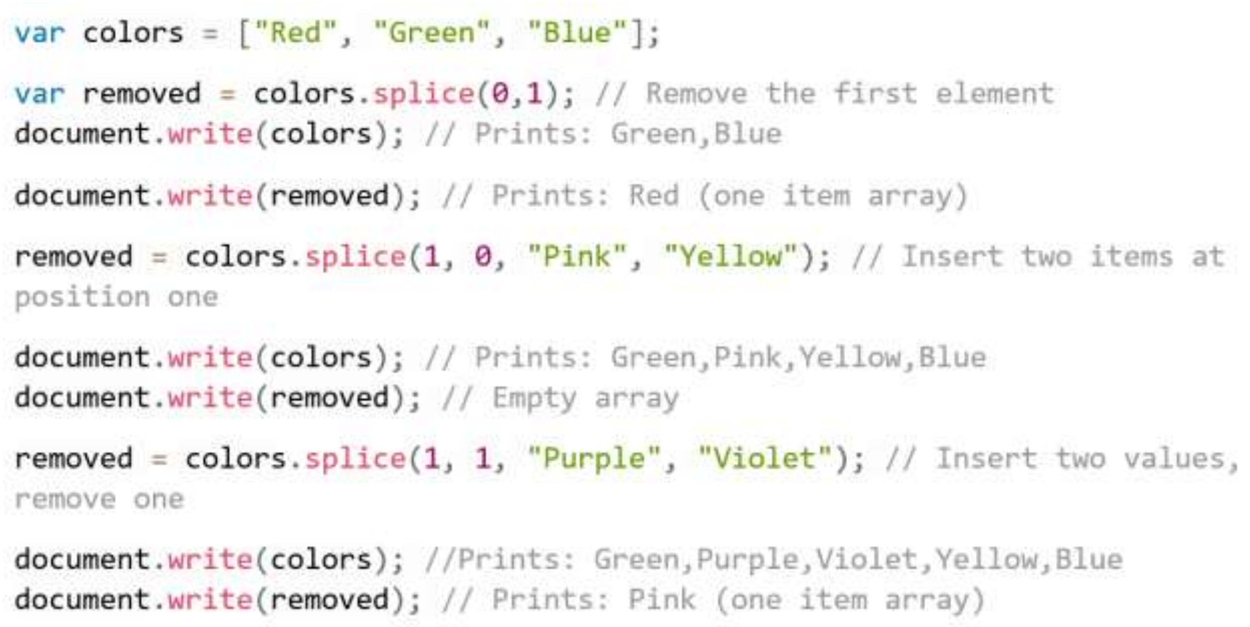
- To add a new element at the end of an array, simply use the **push() method**.
  Example:
  ```
  var colors = ["Red", "Green", "Blue"];
  colors.push("Yellow");
  document.write(colors); // Prints: Red,Green,Blue,Yellow
  ```

- to add a new element at the beginning of an array use the **unshift() method**
  Example:
  ```
  var colors = ["Red", "Green", "Blue"];
  colors.unshift("Yellow");
  document.write(colors); // Prints: Yellow,Red,Green,Blue
  ```

- To remove the last element from an array you can use the **pop() method.** This method returns the value that was popped out.
  Example:
  ```
  var colors = ["Red", "Green", "Blue"];
  var last = colors.pop();
  document.write(last); // Prints: Blue
  ```

- remove the first element from an array using the **shift() method**. This method also returns the value that was shifted out.
  Example:
  ```
  var colors = ["Red", "Green", "Blue"];
  var first = colors.shift();
  document.write(first); // Prints: Red
  ```

❖ Adding and removing element at any position.
  - The **splice() method** is used to add or remove elements from any index, using the syntax **arr.splice(startIndex, deleteCount, elem1, ..., elemN).**
  - This method takes three parameters:
    - the first parameter is the **index** at which to start splicing the array, it is required;
    - the second parameter is the **number of elements to remove** (use 0 if you don't want to remove any elements), it is optional;
    - the third parameter is a set of **replacement elements**, it is also optional.
  - The splice() method returns an array of the deleted elements, or an empty array if no elements were deleted.
  - Examples:



```
var colors = ["Red", "Green", "Blue"];

var removed = colors.splice(0,1); // Remove the first element
document.write(colors); // Prints: Green,Blue

document.write(removed); // Prints: Red (one item array)

removed = colors.splice(1, 0, "Pink", "Yellow"); // Insert two items at
position one

document.write(colors); // Prints: Green,Pink,Yellow,Blue
document.write(removed); // Empty array

removed = colors.splice(1, 1, "Purple", "Violet"); // Insert two values,
remove one

document.write(colors); //Prints: Green,Purple,Violet,Yellow,Blue
document.write(removed); // Prints: Pink (one item array)
```

❖ The **join() method** is used to create string by joining array elements.
  - This method takes an optional parameter which is a separator string that is added in between each element. If you omit the separator, then JavaScript will use comma (,) by default.
  - Example:
    var colors = ["Red", "Green", "Blue"];
    document.write(colors.join()); // Prints: Red,Green,Blue
    document.write(colors.join("")); // Prints: RedGreenBlue
    document.write(colors.join("-")); // Prints: Red-Green-Blue

❖ The **slice() method** is used to extract out a portion of a array.
  - Syntax: arr.slice(startIndex, endIndex)
  - Example: var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];
    var subarr = fruits.slice(1, 3);
    document.write(subarr); // Prints: Banana,Mango

- The **concat() method** can be used to merge or combine two or more arrays.
    - Example:

```
var pets = ["Cat", "Dog", "Parrot"];
var wilds = ["Tiger", "Wolf", "Zebra"];

// Creating new array by combining pets and wilds arrays
var animals = pets.concat(wilds); (OR combine multiple arrays as pets.concat(wilds, bugs);)
document.write(animals); // Prints: Cat,Dog,Parrot,Tiger,Wolf,Zebra
```

- Sorting an Array (alphabetically)

Example:

```
var fruits = ["Banana", "Orange", "Apple", "Papaya", "Mango"];

var sorted = fruits.sort();

alert(fruits); // Outputs: Apple,Banana,Mango,Orange,Papaya

alert(sorted); // Outputs: Apple,Banana,Mango,Orange,Papaya
```

- Reversing an Array

Example:

```
var counts = ["one", "two", "three", "four", "five"];
var reversed = counts.reverse();
alert(counts); // Outputs: five,four,three,two,one
alert(reversed); // Output: five,four,three,two,one
```

Example : Sorting numeric array

```
var numbers = [5, 20, 10, 75, 50, 100];
// Sorting an array using compare function
numbers.sort(function(a, b) {
        return a - b;
});
alert(numbers); // Outputs: 5,10,20,50,75,100
```

Functions

- A function is a group of statements that perform specific tasks
- Syntax:

```
function functionName() {
        // Code to be executed
}
```

- We can store function expression in a variable, and then the variable can be used as a function:

Example:

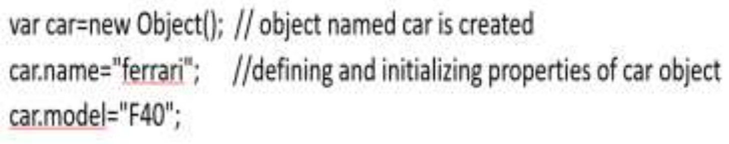
```
var getSum = function(num1, num2) {
var total = num1 + num2;
  return total;
};
alert(getSum(5, 10)); // 0utputs: 15
var sum = getSum(7, 25);
alert(sum); // 0utputs: 32
```

Object

- A JavaScript object is just a collection of named values. These named values are usually referred to as properties of the object.
- An object can be created with curly brackets {} with an optional list of properties. A property is a "key: value" pair.
- Additionally, properties with functions as their values are often called **methods**
- Also new operator can be used to create objects.
- Examples:

```
var person = {
    name: "Peter",
    age: 28,
    gender: "Male",
    displayName: function() {
        alert(this.name);
    }
};
```

OR using new



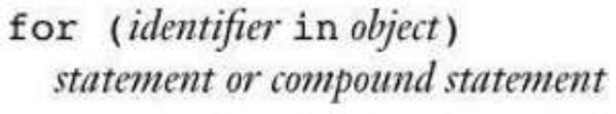
```
var car=new Object(); // object named car is created
car.name="ferrari";   //defining and initializing properties of car object
car.model="F40";
```

- The properties of an object can be accessed with **dot notation**, in which the first word is the object name and the second is the property name.
- To access or get the value of a property, you can use the dot (.), or square bracket ([]) notation,

Example:

```
var book = { "name": "Harry Potter and the Goblet of Fire",
             "author": "J. K. Rowling",
              "year": 2000
             };

// Dot notation
document.write(book.author); // Prints: J. K. Rowling
// Bracket notation
document.write(book["year"]); // Prints: 2000
```

- Looping through object properties
  - Example:



```
for (identifier in object)
    statement or compound statement
```

```
var person = {
    name: "Peter",
    age: 28,
    gender: "Male"
};
// Iterating over object properties
for(var i in person) {
    document.write(person[i] + "<br>");   // Prints: name, age and gender
}
```

  - Setting new property

```
// Setting a new property

person.country = "United States";
document.write(person.country); // Prints: United States
```

- // Deleting property

  delete person.age;
  alert(person.age); // Outputs: undefined

- Calling methods

  person.displayName(); // Outputs: Peter

Constructors

- special methods that create and initialize the properties of newly created objects.
- Every new expression must include a call to a constructor whose name is the same as that of the object being created.
- The **this** variable is used to construct and initialize the properties of the object.
- Example:

```
function car(new_make, new_model, new_year) {
    this.make = new_make;
    this.model = new_model;
    this.year = new_year;
}
```

could be used as in the following statement:

Var my_car = new car("Ford", "Contour SVT", "2000");

```
function display_car() {
    document.write("Car make: ", this.make, "<br/>");
    document.write("Car model: ", this.model, "<br/>")
    document.write("Car year: ", this.year, "<br/>");
}
```

The following line must then be added to the `car` constructor:

```
this.display = display_car;
```

Now the call `my_car.display()` will produce the following output:

```
Car make: Ford
Car model: Contour SVT
Car year: 2000
```

```
Another example:    function Person(first, last, age, eyecolor) {
                            this.firstName = first;
                            this.lastName = last;
                            this.age = age;
                            this.eyeColor = eyecolor;
                            this.name = function() {return this.firstName + " " +
        this.lastName;};
        }
```

Object: var Myself=new Person(“John”,”Doe”,23,”blue”);

JavaScript Data & Time

- The Date object is a built-in JavaScript object.
- It allows you to get the user's local time by accessing the computer system clock through the browser.
- The Date object also provides several methods for managing, manipulating, and formatting dates and times.
- You can declare a new Date object without initializing its value.
- In this case, the date and time value will be set to the current date and time on the user's device on which the script is run.

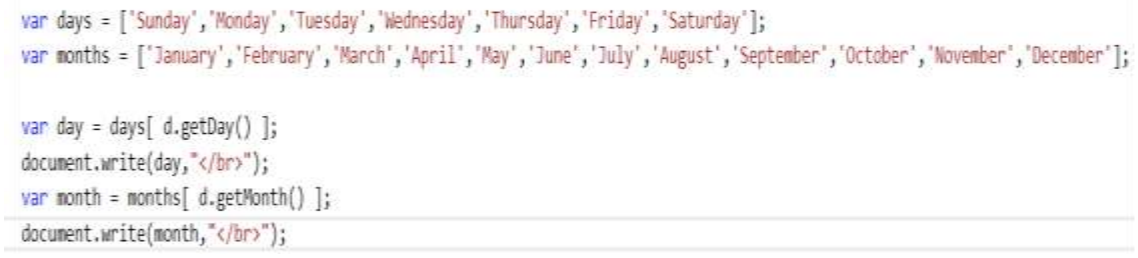
```
var d = new Date();  //get the current date-time
document.write(d);    //Fri Jul 12 2017 12:28:06 GMT+0545 (Nepal Standard Time)
```

- document.write(d.toDateString());   // Fri Jul 12 2017
- alert(d.toLocaleDateString());   // 7/12/2017
- alert(d.toLocaleTimeString());   // 12:34:30 PM

- Methods of extracting individual pieces of information from a Date object.

  - alert(d.getDate()); // Display the day of the month
  - alert(d.getDay()); // Display the number of days into the week (0-6)
  - alert(d.getMonth()); // Display the number of months into the year (0-11)
  - alert(d.getFullYear()); // Display the full year (four digits)

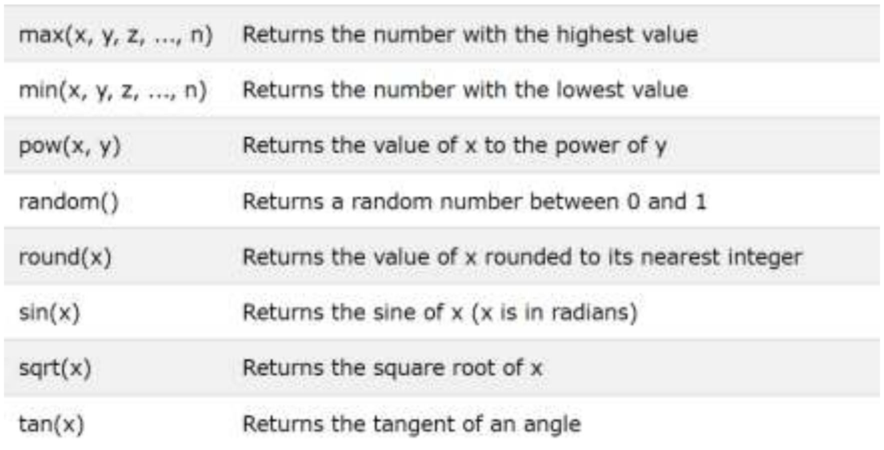
  - alert(d.getHours()); // Display the number of hours into the day (0-23)
  - alert(d.getMinutes()); // Display the number of minutes into the hour (059)
  - alert(d.getSeconds()); // Display the seconds into the minute (0-59)
- To display month names and days:



```
var days = ['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'];
var months = ['January','February','March','April','May','June','July','August','September','October','November','December'];

var day = days[ d.getDay() ];
document.write(day,"</br>");
var month = months[ d.getMonth() ];
document.write(month,"</br>");
```

JavaScript Math Operations

- The JavaScript Math object provides a number of useful properties and methods for performing mathematical tasks.



max(x, y, z, ..., n)	Returns the number with the highest value
min(x, y, z, ..., n)	Returns the number with the lowest value
pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Returns the value of x rounded to its nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

- Example:
  - The Math.random() method is used to generate a floating-point random number in the range from 0 inclusive up to but not including 1.
  - if you want a random integer between zero and an integer higher than one, you could use the following solution:

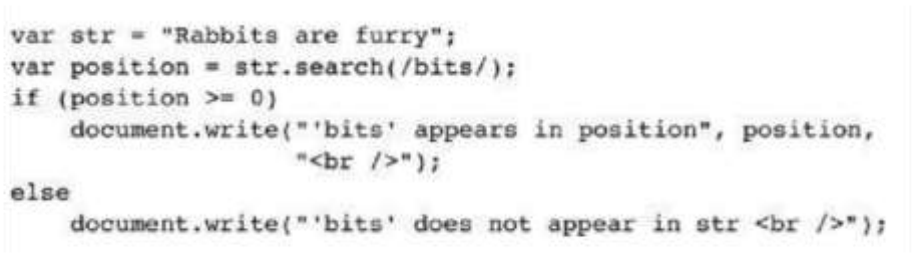
```
// Function to create random integer

function getRandomInt(max) {
        return Math.floor(Math.random() * max);
}
document.write(getRandomInt(3)); // Expected output: 0, 1 or 2
```

JavaScript Pattern Matching

- There are two approaches to pattern matching in JavaScript: one that is based on the **RegExp object** and one that is based on methods of the **String object**.
- The regular expressions used by these two approaches are the same.
- **Regular Expressions**, commonly known as "regex" or "RegExp", are a specially formatted text strings used to find patterns in text.
    - Syntax: /pattern/modifiers;
- It can be used to verify whether the format of data i.e. name, email, phone number, etc. entered by the user is correct or not, find or replace matching string within text content etc.

JavaScript's built-in methods for performing pattern-matching

1. Search()
    - The simplest pattern-matching method is search, which takes a regular expression as a parameter.
    - Syntax: string.search(RegExp);
    - The search method returns the index of the first match in the String object (through which it is called).
    - If there is no match, search returns '–1'.
    - Example:



```
var str = "Rabbits are furry";
var position = str.search(/bits/);
if (position >= 0)
    document.write("'bits' appears in position", position,
                   "<br />");
else
    document.write("'bits' does not appear in str <br />");
```

Output: 'bits' appears in position 3

2. Replace()
    - The replace() method returns a modified string where the pattern is replaced.
    - The replace method is used to replace substrings of the String object that match the given pattern
    - Syntax: string.replace(*searchvalue, newvalue*);
    - Example:

```
var str = "Visit Microsoft!";
var res = str.replace(/microsoft/i, "Facebook");     // res="Visit Facebook!"
```

3. Match()
    - Search for a match in a string. It returns an array of information or null on mismatch.
    - Syntax : string.match(*regexp*);
    - Example:

```
var info="cap is famous in capital of Nepal";
var m=info.match(/(cap)\w*/g); //list words starting with cap
document.write("</br>",m);          //output : cap,capital
```

Character and Character-Class Patterns

- **Metacharacters** are characters that have special meanings in some contexts in patterns.
- The "normal" characters are those that are not metacharacters.
- The pattern metacharacters are:

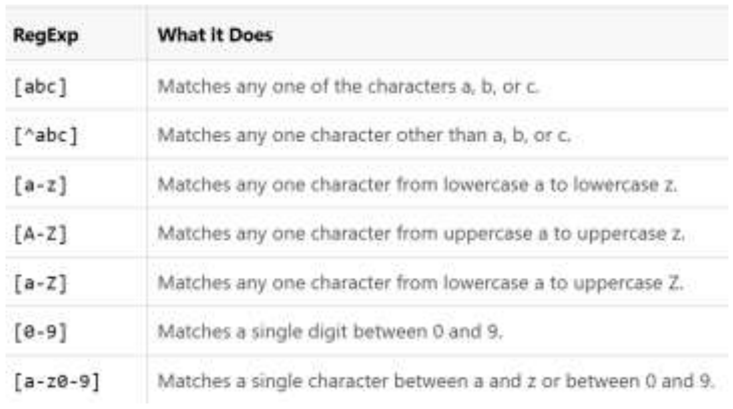
```
\ | ( ) [ ] { } ^ $ * + ? .
```

- Metacharacters can themselves be matched by being immediately preceded by a backslash.

- Square brackets surrounding a pattern of characters are called a <u>**character class**</u>

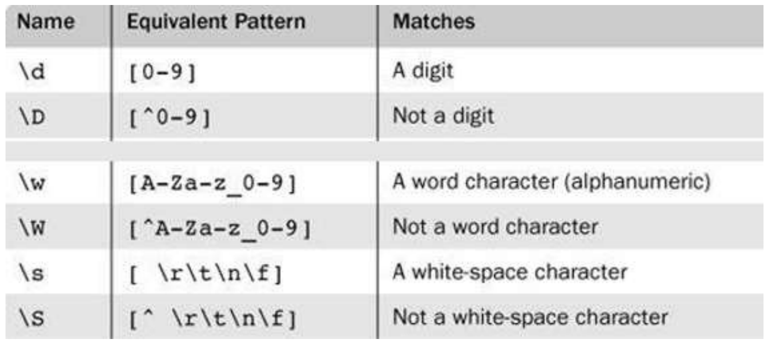
  e.g. [abc].

- Character class examples:



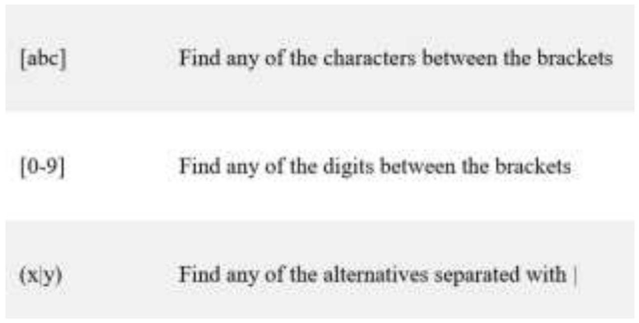
RegExp	What it Does
[abc]	Matches any one of the characters a, b, or c.
[^abc]	Matches any one character other than a, b, or c.
[a-z]	Matches any one character from lowercase a to lowercase z.
[A-Z]	Matches any one character from uppercase a to uppercase z.
[a-Z]	Matches any one character from lowercase a to uppercase Z.
[0-9]	Matches a single digit between 0 and 9.
[a-z0-9]	Matches a single character between a and z or between 0 and 9.

- Predefined character classes



Name	Equivalent Pattern	Matches
\d	[0-9]	A digit
\D	[^0-9]	Not a digit
\w	[A-Za-z_0-9]	A word character (alphanumeric)
\W	[^A-Za-z_0-9]	Not a word character
\s	[\r\t\n\f]	A white-space character
\S	[^ \r\t\n\f]	Not a white-space character

- **Brackets** are used to find a range of characters:



[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

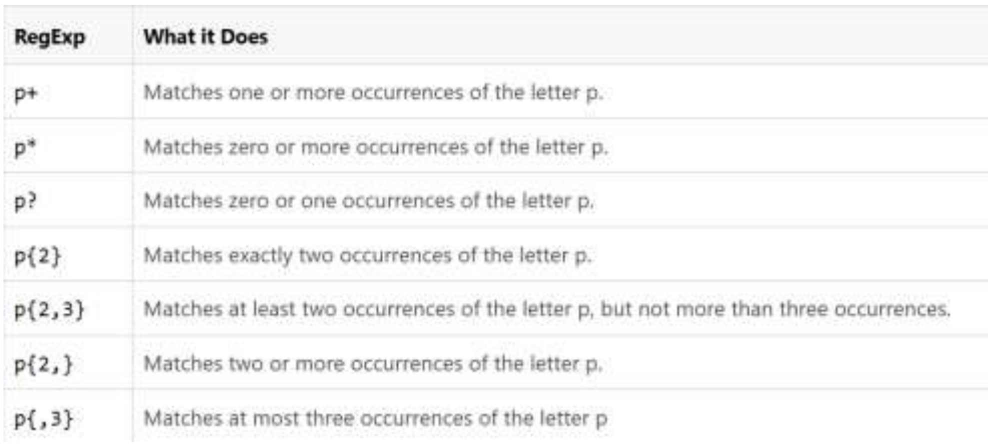
- **Period(.)** matches any single character except newline \n
  - Example: the following pattern matches "snowy", "snowe", and "snowd", among others

    /snow./
- If a **circumflex character** (^) is the first character in a class, it inverts the specified set
  - Example: the following pattern matches any character other than a,e,i,o,u.

    [^aeiou]
- To repeat a pattern, a numeric quantifier, delimited by braces, is attached.
  - Example: The pattern /xy{4}z/ matches xyyyyz

- **Symbolic quantifiers**
  - An **asterisk** means zero or more repetitions.
  - A **plus sign** means one or more repetitions
  - A **question mark** means one or none.

- Various ways to quantify a problem.



RegExp	What it Does
p+	Matches one or more occurrences of the letter p.
p*	Matches zero or more occurrences of the letter p.
p?	Matches zero or one occurrences of the letter p.
p{2}	Matches exactly two occurrences of the letter p.
p{2,3}	Matches at least two occurrences of the letter p, but not more than three occurrences.
p{2,}	Matches two or more occurrences of the letter p.
p{,3}	Matches at most three occurrences of the letter p

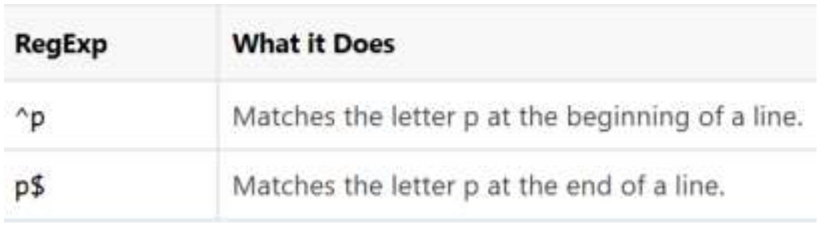
  - Examples:

    /x*y+z?/ :matches strings that begin with any number of x's (including zero), followed by one or more y's, possibly followed by z.

    /\d+\.\d*/ : matches a string of one or more digits followed by a decimal point and possibly more digits.

    /[A-Za-z]\w*/ : matches the identifiers (a letter, followed by zero or more letters, digits, or underscores)

- Position anchors
  - Anchors are used to match at the beginning or end of a line, word, or string.
  - Two common anchors are **caret (^)** which represent the start of the string, and the **dollar ($)** sign which represent the end of the string.
  - Use:



RegExp	What it Does
^p	Matches the letter p at the beginning of a line.
p$	Matches the letter p at the end of a line.

  - Examples:

    /^Nepal/ ->> matches Nepal only if it is at the beginning of a string .

    /Nepal$/ ->> matches Nepal only if it is at the end of a string .(-1 if no match)

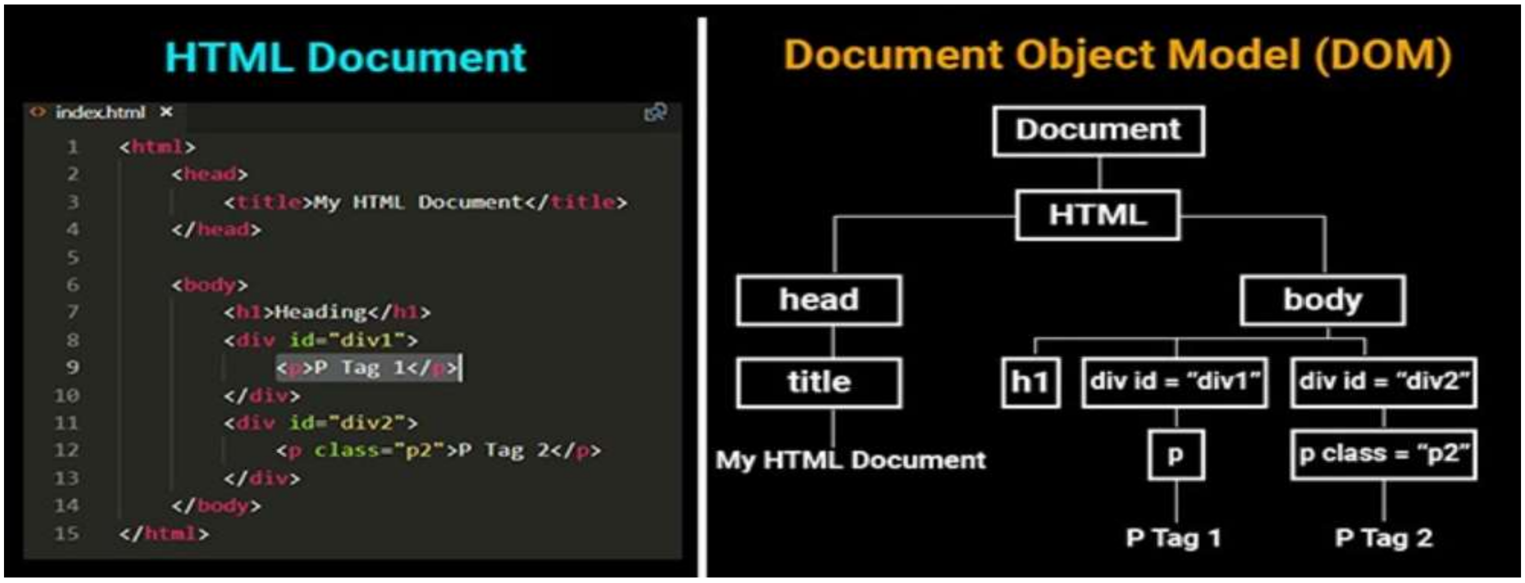
- Pattern modifiers
  - A pattern modifier allows you to control the way a pattern match is handled.
  - Pattern modifiers are placed directly after the regular expression.
  - **g** : To perform a global match/search i.e. finds all occurrences.
  - **i** : To Make the match/search case-insensitive .
  - Examples:

    Given the sentence: Color red is more visible than color blue in daylight.
    The pattern **/color/gi** matches Color,color

- A **word boundary** character (\b) helps you search for the words that begins and/or ends with a pattern.
    - For example, the regexp /\bcar/ matches the words beginning with the pattern car, and would match cart, carrot, or cartoon, but would not match oscar.
    - Similarly, the regexp /car\b/ matches the words ending with the pattern car, and would match oscar or supercar, but would not match cart.
    - Likewise, the /\bcar\b/ matches the words beginning and ending with the pattern car, and would match only the word car.

DOM

- ➢ The Document Object Model (DOM) is a platform and language independent model to represent the HTML or XML documents.
- ➢ It defines the logical structure of the documents and the way in which they can be accessed and manipulated by an application program.
- ➢ In the DOM, all parts of the document, such as elements, attributes, text, etc. are organized in a hierarchical tree-like structure;
- ➢ With the HTML DOM, you can use JavaScript to
  - o build HTML documents
  - o navigate their hierarchical structure
  - o add, modify, or delete elements and attributes or their content, and so on.
- ➢ *The Document Object Model or DOM is basically a representation of the various components of the browser and the current Web document (HTML or XML) that can be accessed or manipulated using a scripting language such as JavaScript.*
- ➢ The DOM is a W3C (World Wide Web Consortium) standard.
- ➢ *"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."*
- ➢ When a web page is loaded, the browser creates a Document Object Model of the page.



```
<html>
    <head>
        <title>My HTML Document</title>
    </head>

    <body>
        <h1>Heading</h1>
        <div id="div1">
            <p>P Tag 1</p>
        </div>
        <div id="div2">
            <p class="p2">P Tag 2</p>
        </div>
    </body>
</html>
```

- ➢ The above diagram demonstrates the parent/child relationships between the nodes. The topmost node i.e. the Document node is the root node of the DOM tree, which has one child, the <html> element. Whereas, the <head> and <body> elements are the child nodes of the <html> parent node.

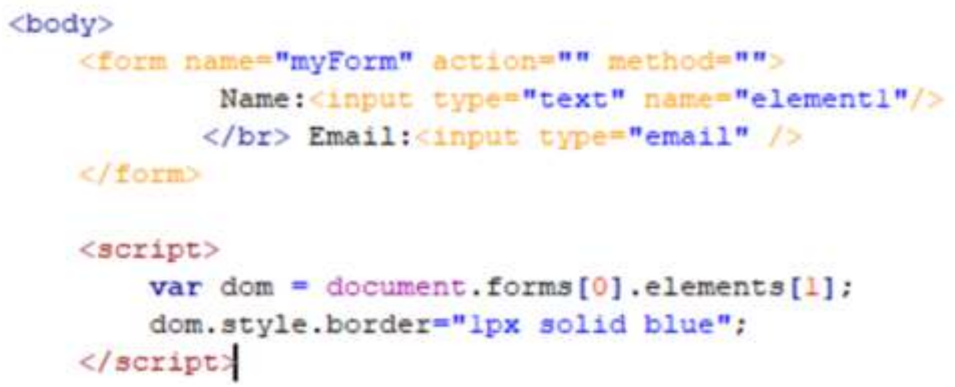
Element access in JavaScript

- ❖ Element access using the DOM address of the element
  Example:

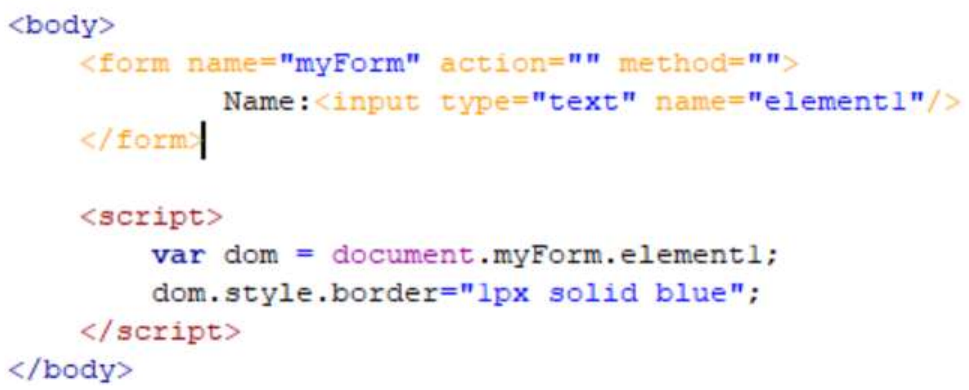
```
var x = document.forms[0].elements[0];
```

- ❖ Element access using element's name
  Example:

```
var dom = document.myForm.element1;
```

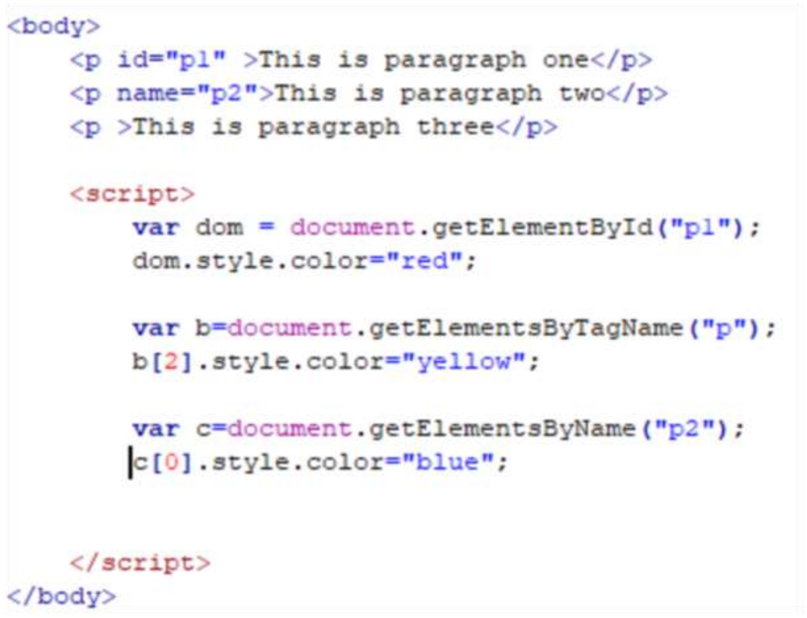


```
<body>
    <form name="myForm" action="" method="">
            Name:<input type="text" name="element1"/>
           </br> Email:<input type="email" />
    </form>

    <script>
        var dom = document.forms[0].elements[1];
        dom.style.border="1px solid blue";
    </script>
```



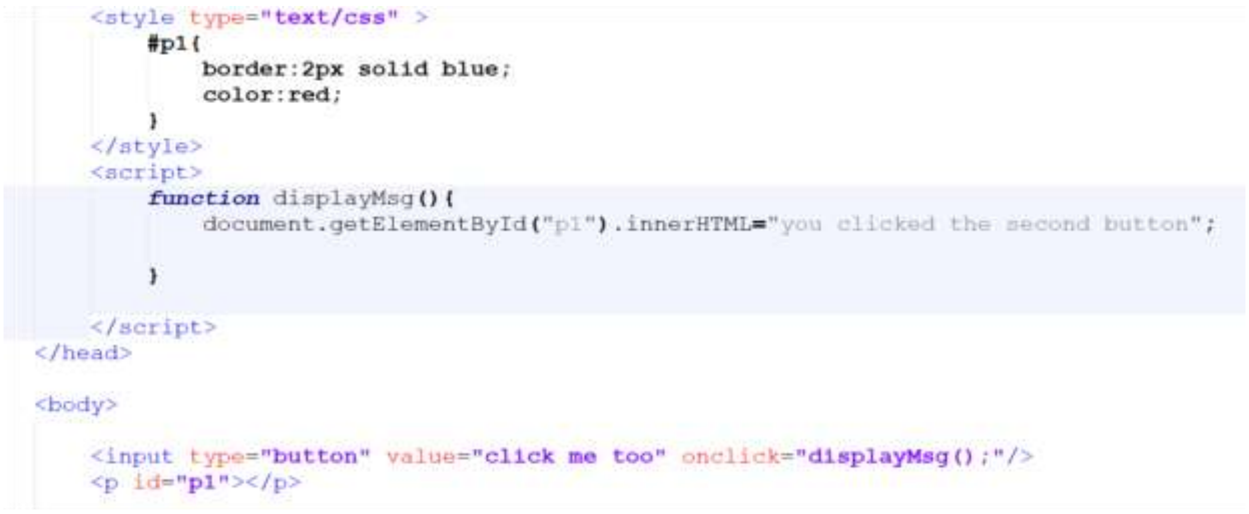
```
<body>
    <form name="myForm" action="" method="">
            Name:<input type="text" name="element1"/>
    </form>

    <script>
        var dom = document.myForm.element1;
        dom.style.border="1px solid blue";
    </script>
</body>
```

- ❖ Element access using tag name
  Example:
  var x = document.getElementsByTagName("p");
- ❖ Element access using element's name:
  Example:
  Var x=document.getElementsByName("p2");
- ❖ Element access using element's id:
  Var x = document.getElementById("id1");



```
<body>
    <p id="p1" >This is paragraph one</p>
    <p name="p2">This is paragraph two</p>
    <p >This is paragraph three</p>

    <script>
        var dom = document.getElementById("p1");
        dom.style.color="red";

        var b=document.getElementsByTagName("p");
        b[2].style.color="yellow";

        var c=document.getElementsByName("p2");
        c[0].style.color="blue";


    </script>
</body>
```

- ✓ InnerHTML
  - ➢ The innerHTML property sets or returns the HTML content (inner HTML) of an element.
  - ➢ Syntax: *HTMLElementObject*.innerHTML = *"text";*
  - ➢ Example:



```
    <style type="text/css" >
        #p1{
            border:2px solid blue;
            color:red;
        }
    </style>
    <script>
        function displayMsg(){
            document.getElementById("p1").innerHTML="you clicked the second button";

        }

    </script>
</head>

<body>

    <input type="button" value="click me too" onclick="displayMsg();"/>
    <p id="p1"></p>
```