Introduction To Javascript

JavaScript is a very powerful client-side scripting language. It is used mainly for enhancing the webpage by making it more lively and interactive. It is one of the most popular programming languages now a days and used to create various mobile apps, desktop apps, games and many more…

Client-side Programming : It is the program that runs on the client machine (browser) and deals with the user interface/display and any other processing that can happen on client machine

Write First JavaScript Code

There are so many ways to write a JavaScript code. The first is writing with in an HTML document using SCRIPT tag. The SCRIPT tag can be placed with in HEAD tag or BODY tag.

Remember that, If we are placing the script in the head section, Then it will be executed before the <body> is rendered. So the page load time will be more. If we want to speed up the page load time then we should place our script at the end of the body tag.

Refer : Example1.html and Example2.html

document.write

alert()

// (Single Line Comment) : Ignores rest of the code on same line

/*…*/ (Multi Line Comment) : Ignores all the text with in the block

What is ECMAScript or ES ?

ECMAScript (or ES) is a general-purpose programming language, standardized by ECMA International according to the document ECMA-262. It is a JavaScript standard meant to ensure the interoperability of Web pages across different Web browsers.

Variables in JavaScript

Variable is a named memory location to hold value. The value can be changed at any time. var keyword can be used to declare a variable name and = (assignment operator) can be used to assign value to a variable. Declaration and assignment can be done at same time. Remember, a variable declared with out any value, have the value *undefined*. If we redeclare a variable, it will not loose its value.

Refer : Example5.html

Variable Naming Rules in JavaScript

- JavaScript variable names are case sensitive so x and X are two different names.
- JS Variable names can contain letters, digits, underscores, or dollar signs but first character can not be a digit.
- JS Variable names can not contain reserved keywords. Remember, Reserved words are in lower case so to avoid mistake start the first letter of the variable name with upper case letter.

we should use var Var

Data Types in JavaScript

To properly operate data we must know it's data type. JavaScript provides eight different data types which are undefined, null, Boolean, string, symbol, bigint, number, and object.

Remember :

- Same variable can hold different data type if assigned
- typeof operator can be used to get the data type of variable
- JS Numbers are stored as double precision floating point numbers
- Strings are mentioned with in single quotes or double quotes. Escape characters (\") can be used to put double quotes inside double quotes or single quotes inside single quotes.
- Boolean data type has two values only true or false. These are very helpful when we have to store only two values like yes or no.

Refer : Example7.html

More About Double Precision Floating Point Number

It's called a floating point number because the decimal point can "float" around (move left and right).

The "precision" of a number refers to how many digits it can have.

A “Single precision floating point number” can store only 32 bits!

A "double precision floating point number" is a decimal-type number that can store 64 bits (binary digits) of information.

Arithmetic Operators

+ (addition) : Adds both the operands

- (Subtraction) : Subtracts the right operand from left

* (Multiplication) : Multiplies both the operands

/ (Division) : Divides left operand with right

% (Modulus) : Performs integer division and gives the reminder

Refer : Example8.html

Increment & Decrement Operators in JavaScript

var++ (Post Increment) : Uses the original value or var in expression then increases the value by 1

++var (Pre Increment) : Increases the value by 1 Then uses the modified new value in the expression.

Var-- (Post Decrement) : Uses the original value or var in expression then decreases the value by 1

--var (Pre Decrement) : Decreases the value by 1 Then uses the modified new value in the expression.

Refer : Example9.html

```
<title>Increment & Decrement Operators in JavaScript</
<script type="text/javascript">
    a = 5;b = 6
    document.write(a++ + b+"<br/>");
    a = 5;b = 6
    document.write(++a + b+"<br/>");
    a = 5;b = 6
    document.write(a-- + b+"<br/>");
    a = 5;b = 6
    document.write(--a + b+"<br/>");
</script>
```

Assignment Operators

= : Assigns the value of right operand to left

+= : adds both the operands and assigns value to left operand

-= : subtracts right operand from left and assigns value to left operand

*= : product of both the operands assigns to left operand

/= : Divides left operand with right and assigns result to left operand

%= : Finds the modulus and assigns to left operand

Refer : Example10.html

```
//a*=b // a = a * b
//document.write(" ",a)
//a /=b // a = a / b
//document.write(" ",a)
a%=b // a = a % b
document.write(" ",a)
```

Comparison Operators in JavaScript

== : (Equal to) Returns true if the value of both the operands are same

=== : (Identical to) Returns true if the value and data type of both the operands are same

!= : (Not Equal to) Returns true if both are not equal

!== : (Not Identical) Returns true if both are not identical

> : (Greater Than) Returns true if left operand is greater then right

>= : (Greater Than or Equal To) Returns true if left operand is greater than or equal to right

< : (Less Than) Returns true if left operand is less than right

<= : (Less Than or Equal To) Returns true if left operand is less than or equal to right

Refer : Example11.html

```
//  document.write(5==3)
//  document.write(5==5)
//  document.write(5=='5')
//  document.write(5==='5')
    document.write(5===5)
//  document.write(5!='5')
//  document.write(5!=3)
//  document.write(5!=='5')
//  document.write(5 > 3)
```

Logical Operators in JavaScript

&& (Logical And) : Returns true if both the operands are true

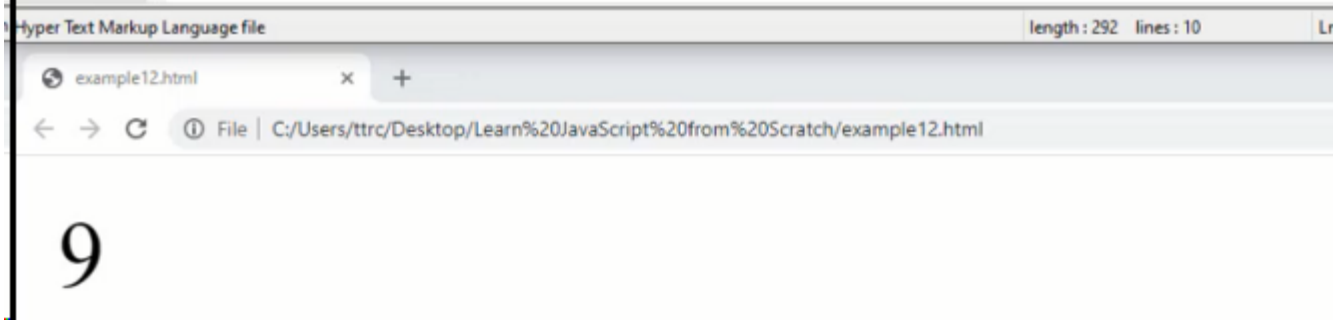
|| (Logical Or) : Returns true if one of the operands is true

! (Not Operator) : Returns true if the operand is false and returns false if the operand is true.

Refer : Example12.html

```
//document.write((5>3)&&(5>4))
//document.write((5>3)&&(5>6))
document.write((5>3)||(5>6))
//document.write(!(5>6))
```

```
6     document.write((5>3)&&(5+4))
7     //document.write((5>3)&&(5>6))
8     //document.write((5>3)||(5>6))
9     //document.write(!(5>6))
10 </script>
```




```
//document.write((5>3)||(5>6))
//document.write(!(5>6))
//document.write((5>3)&&(5+4))
//document.write((5<3)&&(5+4))
//document.write((5>3)||(5+6))
document.write((5<3)||(5+6))
```

Ternary Operators in JavaScript

Syntax :

Variable = (conditional expression) ? Value if true : value if false

If the conditional expression is evaluated true, value if true will be assigned to variable other wise value if false will be assigned to the variable.

Remember, Nested ternary operators can also be used if necessary.

Refer : Example13.html

```
<script type="text/javascript">
    age=17
    document.write(age>=18?"Major":"Minor")
</script>
```

if statement in JavaScript

If is a conditional statement. It checks for the specified condition and evaluates a block of statements if the condition is true.

Syntax :

```
if (condition) {
        Block of Statements
}
```

Refer : Example14.html

if…else statement in JavaScript

If…else is a conditional statement. It checks for the specified condition and evaluates if block if the condition is true other wise evaluates the else block. It is an extended version of ternary operator.

Syntax :

```
if (condition) {
        Block of Statements
}else{
        Block of Statements
}
```

Refer : Example15.html

switch statement in JavaScript

It can be used in place of multiple else if statements. The expression is evaluated and compared with all case values. If there is a match the associated block of statements will be executed. If found break statements the control will come out of switch statement other wise it will execute all other following blocks till break is found or switch statement ends. If there is no match the default block will be executed.

Syntax :

```
switch (expression) {
        case value1:
                Block1 of Statements
                break;
        case value2:
                Block1 of Statements
                break;
        default:
                Block1 of Statements
}
```

Refer : Example17.html

```
// Print day of week
x = 5
switch(x){
    case 1:
        document.write("Sunday");
        break
    case 2:
        document.write("Monday");
        break
    case 3:
        document.write("Tuesday");
        break
```

For loop in JavaScript

for(initialization statements, conditional expression, increment statements){

Block of statements

}

1. Initialisation statements execute once before the loop starts.
2. If conditional expression evaluates true, than only the block of statements executes otherwise loop terminates.
3. Increment statements executes each time after the execution of loop.

Remember, if you want to use more then one Initialisation statements then separate them by commas. If you omit all three, it will be a non-terminating loop.

```
<script type="text/javascript">
    x=1;
    for(;;){
        if(x<=50){
            document.write(x,"<br/>");
            x++;
        }else{
            break;
        }
    }
</script>
```

While loop in JavaScript

```
while (conditional expression){
        Block of statements
}
```

The block of statements executes till the conditional expression is true. If the condition is always true, the loop will be a non-terminating loop.

```
x=1;
while(true){
    if(x<=20){
        document.write(x,"<br/>");
        x++;
    }else{
        break;
    }
}
```

Popup boxes in JavaScript

JavaScript offers three types of popup boxes.

alert – Displays the specified messages and restricts the user to access webpage until the user clicks on the ok button. To display line breaks in popup box escape character \n can be used.

confirm – It displays the specified message with wo buttons ok and cancel. Ok returns true and cancel returns false. It also restricts the user to access webpage until the box is closed.

prompt – Asks the user for input and returns it as string. If the user clicks on ok button without any value, it returns a blank string. If the user clicks on cancel button, it returns null. The prompt method takes two parameters, 1. message to display and 2. (optional) place holder for the input. It also restricts user to access webpage until closing.

Break and Continue in JavaScript

break statement breaks the loop execution and transfers the control out of the loop.

continue statement ignores the statements after it and transfers the control again to the starting of loop.

```
<script type="text/javascript">
for(x=1;x<=10;x++)
{
    if(x==5){
        continue;
    }
    document.write(x,"<br\>");
}
</script>
```

Introduction to functions in JavaScript

A JavaScript function is a block of code combinedly executes a certain task. It is useful when you need a specific block of code repeatedly with different values. A function can only be executed when we call it.

function keyword followed by function name and after that a block of statements with in curly braces is the process creating an user defined function. Function naming rule is same as variable naming rules.

Parameters and Arguments in JavaScript

With in the set of parenthesis in the function declaration we can write variable names that are to be used with in the function block. These variables are called parameters.

While calling this function the value for these variables has to be placed with in the parenthesis of the calling function. These values are called arguments.

Generally parameters and arguments are used to call the same function with different values.

Remember to put a semicolon at the end of calling function. If arguments are passed with out parameters or more then number of parameters, then they will be stored in an array of arguments and can be accessed as arguments[0], arguments[1] and so on. We will discuss more about arrays in later chapters. If a function is called with less arguments, the missing values are set to undefined.

Function return statement in JavaScript

A return statement can be used in a function to return a value. If a function is not returning anything means it is returning undefined. Return statement in a function is optional.

Real World Objects

Let us consider a Car as a real world object. A car may have different properties like its color, weight. And a car may have different methods like start, stop etc.

Introduction to Objects in JavaScript

We already know that variables are named memory locations. A variable is used to store a value.

Objects are also named memory locations but can contain many values as PropertyName:value pairs (PropertyName, value pair is separated by a colon) separated by commas. Values may be of primitive data type or other objects.

Example :

```
var Student = {
        firstName : ”Ramesh”,
        lastName : ”Mohanty”,
        rollNo:15
}
```

Remember, an object definition can be of multiple lines.

Object Properties in JavaScript

JavaScript object properties can be accessed by :

- objectName.propertyName
- objectName[‘propertyName’]

Example :

Student.firstName

Student.lastName

Student[‘rollNo’]

Refer : Example30.html

Object Constructor Function in JavaScript

We can create object constructor function to create multiple objects of same type. Let us create a Student object constructor so that we can create so many objects of student type if required.

Syntax : function objectName(value1, value2…){

this.property1 = value1,

this.property2 = value2

…

}

Refer : Example31.html

```
function Student(firstName, lastName) {
    this.firstName = firstName
    this.lastLame = lastName
}

var student1 = new Student('moin', 'ahamed')
var student2 = new Student('ali', 'ahamed')

document.write(student1.firstName + "<br>")
document.write(student1.lastLame + "<br>")
document.write(student2.firstName + "<br>")
document.write(student2.lastLame + "<br>")
```

this.firstName is the property of the student object or object constructor

Object Methods in JavaScript

Object Methods are functions that are stored as value for object properties. Methods can be defined inside the constructor function or outside the constructor function.

Refer : Example32.html & Example33.html

Arrays in JavaScript

Arrays are used to store multiple values in a single variable. Array elements can be accessed by their index number. Array index numbering starts from 0 (zero). If we try to access an index which is not in array then it will return value undefined.

```
<script type="text/javascript">
    roll1 = 15;
    roll2 = 16;
    roll3 = 17;

    var rolls = new Array(15,16,17);

    document.write(rolls[0]+"  "+rolls[1]+"  "+rolls[2]);
```

```
    var rolls = new Array(3); // ano
    rolls[0]=15;
    rolls[1]=16;
    rolls[2]=17;
```

```
var rolls = new Array(); // Arrays are dynamic so parameter is optional
rolls[0]=15;
rolls[1]=16;
rolls[2]=17;
```

Faster and Simpler Method of Array Creation in JavaScript

var arrayName = [item1, item2, item3…]

Refer : Example35.html

```
<script type="text/javascript">
    var names = ["Ramesh","Suresh","Sulagna"];
    var rolls = [15,16,17]; // Faster and simpler then all other

    document.write(names[0]+"   "+names[1]+"   "+names[2]);
    document.write(rolls[0]+"   "+rolls[1]+"   "+rolls[2]);
```

Array Properties in JavaScript

length : sets or returns the number of elements in an array

prototype : Allows you to add properties and methods to an Array Object

```
Array.prototype.Ucase = function
(i) {
    return this[i].toUpperCase()
}

student_name = ['tito', 'hafiz']

document.write
(student_name.Ucase(0))
```

Array Methods in JavaScript

ArrayName.toString() : Converts an array to a string of comma separated array items.

arrayName.join("separationString") : Joins all the items of an array to a string separating by the mentioned separationString.

arrayName.shift() : removes the first item and returns the removed item.

arrayName.pop() : removes the last item and returns the removed item.

arrayName.push(item) : adds the specified item at the end of array and returns new array length.

arrayName.unshift(item) : adds the specified item at the starting of array and returns new array length.

Array Methods in JavaScript

ArrayName.splice(position, noOfElementsToRemove, item1, item2…) : Removes the mentioned number of elements from mentioned position and adds the specified items.

Array1.concat(array2) : concatenates array1 with array2 and assigns to array1.

arrayName.slice(startindex, uptoindex) : slice the array from start index to uptoindex (up to index number is not included) number and returns it. If uptoindex is not included, if slices up to end.

```
students_name = ['tito',
'hafiz', 'tuhin']
document.write
(students_name.toString() + "
<br>")

for (let i = 0; i <
students_name.length; i++) {
    document.write
(students_name[i].toUpperCase
() + "<br>")
}
```

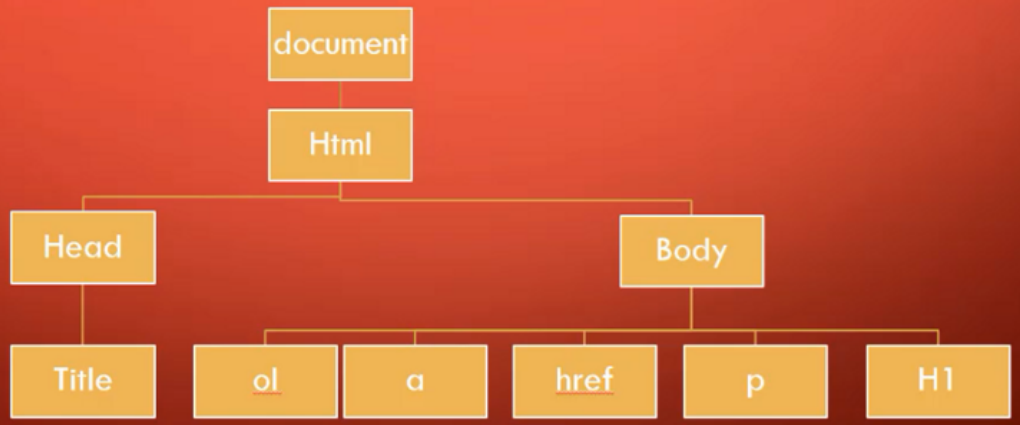
Selecting DOM Elements by tag name in JavaScript

getElementsByTagName(tagName) : This method can be used to access all HTML elements having specified tag name. So, it returns an array of all the elements with specified tag name.

document is the root we are accessing the whole web page

What is DOM

When we open a webpage, The HTML of the page is loaded and rendered visually on the screen. When a webpage is loaded, the browser creates a Document Object Model of the page. This DOM of page is constructed as a tree of objects. The tree may look something like below image :



What JavaScript can do with DOM

Java Script can

- Change all the HTML elements in the page
- Change all the HTML attributes in the page
- Change all the CSS styles in the page
- Remove existing HTML elements and attributes
- Add new HTML elements and attributes
- React to all the existing HTML events
- Can create new HTML events in the page

Simply to say, JavaScript can be used to manipulate the HTML DOM of a page to dynamically add, delete and modify elements.

Selecting DOM Elements by ID in JavaScript

getElementById(idName) : This method can be used to access any HTML element by its ID.

Selecting DOM Elements by class name in JavaScript

getElementsByClassName(className) : This method can be used to access all HTML elements having specified class name. So, it returns an array of all the elements with specified class name.

DOM Node Types

There are 12 types of nodes, But generally we work with 4 of them :

1. document – the entry point of DOM
2. element nodes – HTML tags
3. text nodes – contain text
4. comment nodes – HTML comments do not display on web browser but JS can read them from DOM

getElementsById return a single elements

but

getElementsByClass return a objects

Manipulating DOM Nodes

element.childNodes : returns an array of the specified elements child nodes

element.firstChild : returns the first child node of spesified element

element.lastChild : returns the last child node of specified element

element.hasChildNodes : returns true if the specified element has any child nodes

element.nextSibling : returns the previous at the same tree level

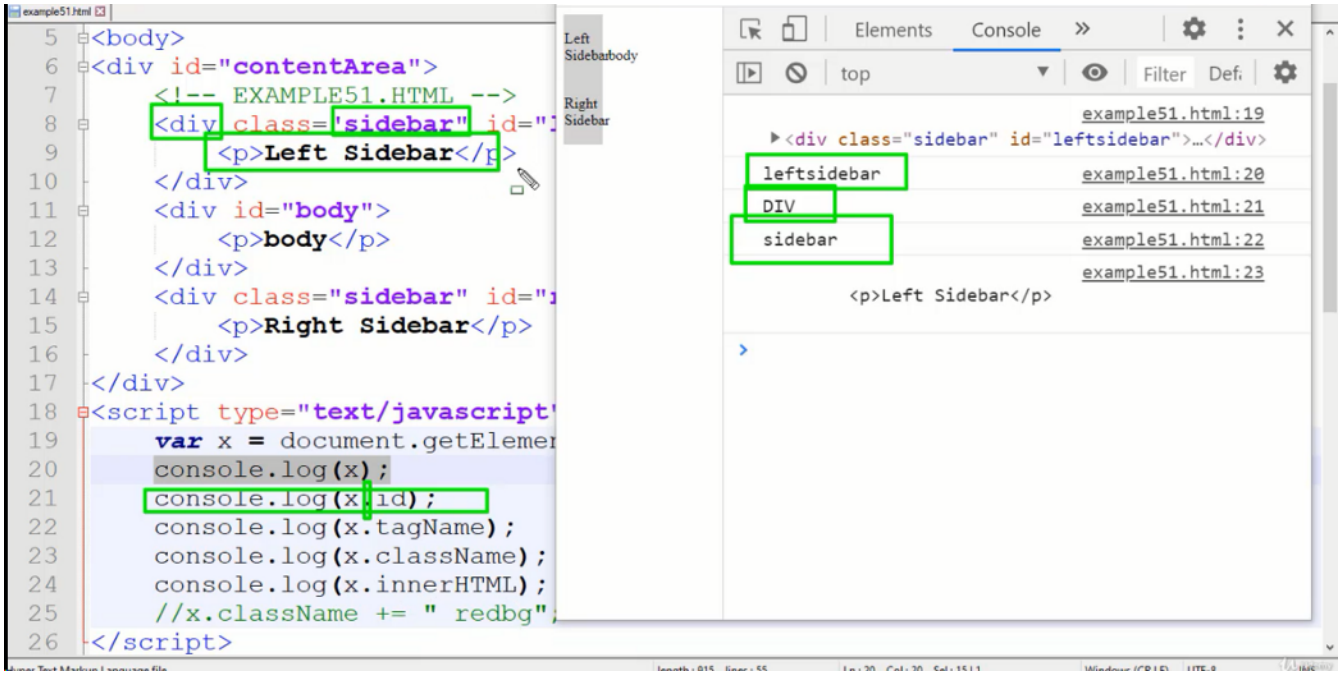
element.previousSibling : returns the previous sibling at the same tree level

element.parentNode : returns the parent node of the specified element

Changing attributes of DOM elements

We know various methods to access the DOM elements. After accessing we can change the attributes of the DOM elements.

element.attribute = newValue;



Style object properties of DOM elements

element.style.backgroundColor = “colorvalue”;

Sets or returns the background color of an element.

element.style.border = “ border-width border-style border-color ”;

Sets or returns the width style and color of border

element.style.margin = “top right bottom left”;

Sets or returns the margins of an element

element.style.padding = “top right bottom left”;

Sets or returns the padding of an element

```
git config --global http.sslBackend "openssl"
```



```
git config --global http.sslCAInfo "C:\Program Files\Git\mingw64\ssl\cert.pem"
```