**JavaScript**

**Section 1:**

What is javaScript?

Javascript is a dynamic computer programming language. It is most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.

The JavaScript client-side mechanism provides many advantagesThe JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

Advantages of JavaScript :

• Less server interaction

• Immediate feedback to the visitors

• Increased interactivity

• Richer interfaces

Section 2 : Basics

Variables:

Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

Variables are declared with the var keyword

Eg. var name;

var age;

Global Variables: A global variable has global scope which means it can be defined anywhere in your JavaScript code.

Local Variables: A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

JavaScript is untyped language. This means that a JavaScript variable can hold a value of any data type.

Functions:

A function is a set of statements that take inputs, do some specific computation, and produce output. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can call that function.

Example

// Function in JavaScript

function avg(a, b) {

    c = (a + b) / 2;

    return c;

}

c1 = avg(4,6);

c2 = avg (14,16);

console.log(c1,c2);

Section 3: Operators

JavaScript supports the following types of operators.

Assume variable A holds 10 and variable B holds 20, then:

• Arithmetic Operators

1. + (Addition) Adds two operands Ex: A + B will give 30

2. - (Subtraction) Subtracts the second operand from the first Ex: A - B will give -10

3. \* (Multiplication) Multiply both operands Ex: A \* B will give 200

4. / (Division) Divide the numerator by the denominator Ex: B / A will give 2

5. % (Modulus) Outputs the remainder of an integer division Ex: B % A will give 0

6. ++ (Increment) Increases an integer value by one Ex: A++ will give 11

7. -- (Decrement) Decreases an integer value by one Ex: A-- will give 9

• Comparison Operators

Assume variable A holds 10 and variable B holds 20

1. == (Equal) Checks if the value of two operands are equal or not, if yes, then the condition becomes true. Ex: (A == B) is not true.

2 . != (Not Equal) Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. Ex: (A != B) is true.

3. > (Greater than) Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. Ex: (A > B) is not true.

4. < (Less than) Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. Ex: (A < B) is true.

5. >= (Greater than or Equal to) Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: (A >= B) is not true.

6. <= (Less than or Equal to) Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: (A <= B) is true.

• Logical Operators

Assume variable A holds 10 and variable B holds 20

1. && (Logical AND) If both the operands are non-zero, then the condition becomes true. Ex: (A && B) is true.

2. || (Logical OR) If any of the two operands are non-zero, then the condition becomes true. Ex: (A || B) is true.

3. ! (Logical NOT) Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. Ex: ! (A && B) is false.

• Assignment Operators

1. = (Simple Assignment ) Assigns values from the right side operand to the left side operand Ex: C = A + B will assign the value of A + B into C
2. += (Add and Assignment) It adds the right operand to the left operand and assigns the result to the left operand. Ex: C += A is equivalent to C = C + A
3. -= (Subtract and Assignment) It subtracts the right operand from the left operand and assigns the result to the left operand. Ex: C -= A is equivalent to C = C - A
4. \*= (Multiply and Assignment) It multiplies the right operand with the left operand and assigns the result to the left operand. Ex: C \*= A is equivalent to C = C \* A
5. /= (Divide and Assignment) It divides the left operand with the right operand and assigns the result to the left operand. Ex: C /= A is equivalent to C = C / A
6. %= (Modules and Assignment) It takes modulus using two operands and assigns the result to the left operand. Ex: C %= A is equivalent to C = C % A
7. • Bitwise Operators
8. 1. & (Bitwise AND) It performs a Boolean AND operation on each bit of its integer arguments. Ex: (A & B) is 2.
9. 2. | (BitWise OR) It performs a Boolean OR operation on each bit of its integer arguments. Ex: (A | B) is 3.
10. 3. ~ (Bitwise Not) It is a unary operator and operates by reversing all the bits in the operand.

Section 4: Control Flow

1. IF-ELSE

The syntax of an if-else statement is as follows:

if (expression){

Statement(s) to be executed if expression is true

}

else{

Statement(s) to be executed if expression is false

}

Here JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the ‘if’ block, are executed. If the expression is false, then the given statement(s) in the else block are executed.

var age = 15;

if( age > 18 ){

console.log("Qualifies for driving");

}

else{

console.log("Does not qualify for driving");

}

1. Switch-Case

switch (expression) {

case condition 1: statement(s)

break;

case condition 2: statement(s)

break;

...

case condition n: statement(s)

break;

default: statement(s)

}

Eg. var grade='A';

switch (grade) {

case 'A': console.log ("Good job");

case 'B': console.log ("Pretty good");

case 'C': console.log("Passed");

case 'D': console.log ("Not so good");

case 'F': console.log ("Failed");

default: console.log ("Unknown grade")

}

1. While Loop

The syntax of while loop in JavaScript is as follows:

while (expression){

Statement(s) to be executed if expression is true

}

Eg. var count = 0;

while (count < 10){

console.log("Current Count : " + count );

count++;

}

1. Do…while Loop

do{

Statement(s) to be executed;

} while (expression);

Eg. var count = 0;

do{

console.log("Current Count : " + count + "");

count++;

}while (count < 5);

1. For Loop

for (initialization; test condition; iteration statement){

Statement(s) to be executed if test condition is true

}

Eg.

var count;

for(count = 0; count < 10; count++){

console.log("Current Count : " + count );

}

1. For-In Loop

for (variablename in object){

statement or block to execute

}

Eg. const persons ={

    name: 'Mosh',

    age: 30,

}

for(let key in persons) {

    console.log(key, persons[key]);

}