JAVASCRIPT-TUTORIAL

**VARIABLES**

Any application that we write is going to be based on data. We use variables to hold this information. Data is stored in the computer memory.

**Ex:**

var x = 5;

var y = 10;

var z = x + y;

document.write(z);

**DATA TYPES**

Data types describe the different types or kinds of data that we're gonna be working with and storing in variables.

In Javascript, there are five basic, or primitive, types of data. The five most basic types of data are strings, numbers, booleans, undefined, and null. We refer to these as primitive data types. A single variable can only store a single type of data. That means it’s important for you to learn to store the data correctly.

**NUMBER:**

Represents numeric values e.g. 100

**EX:**

var x = 10;

var y = 20;

var z = x+y;

console.log(z);

**STRINGS:**

Represents sequence of characters e.g. "hello"

**EX:**

var type1 = “andhra”;

var type2 = “Pradesh”;

console.log(type1 + “” + type2);

**BOOLEAN:**

Represents Boolean value either false or true

**EX:**

var x = 15;

console.log(x > 25);

var x = 50;

console.log(x < 55);

**ARRAY**

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

**EX:**

**There are two syntaxes for creating an empty array:**

var array = new Array();

var array = [];

**Almost all the time, the second syntax is used. We can supply initial elements in the brackets**

var fruits = [“Apple”, “Orange”, ”Plum”];

**Array elements are numbered, starting with zero.**

**We can get an element by its number in square brackets:**

var fruits = [“Apple”, “Orange”, ”Plum”];

console.log(fruits[0]); // “Apple”

console.log(fruits[1]); //”Orange”

console.log(fruits[2]);”Plum”

**We can replace an element:**

fruits [2] = “Lemon”; // now [“Apple”, “Orange”, ”Lemon”];

**Or add a new one to the array:**

fruits [3] = “Banana”; // now [“Apple”, “Orange”, ”Lemon” “Banana”];

**The total count of the elements in the array is its length:**

var fruits = [“Apple”, “Orange”, ”Plum”];

console.log (fruits.length);

Popping and Pushing

When you work with arrays, it is easy to remove elements and add new elements.

This is what popping and pushing is:

Popping items **out** of an array, or pushing items **into** an array.

var fruits = [“Banana”, “Orange”, “Apple”, “Mango”];

fruits.pop();

console.log(fruits);

var fruits = [“Banana”, “Orange”, “Apple”, “Mango”];

fruits.push(“Kiwi”);

console.log(fruits);

## Shifting Elements

The shift() method removes the first array element and "shifts" all other elements to a lower index.

var fruits = [“Banana”, “Orange”, “Apple”, “Mango”];

fruits.shift();

console.log(fruits);

The unshift() method adds a new element to an array (at the beginning), and unshifts older elements:

var fruits = [“Banana”, “Orange”, “Apple”, “Mango”];

fruits.unshift(“Kiwi”, “Pineapple”);

console.log(fruits);

**OBJECTS**

**objects** in **JavaScript** may be defined as an unordered collection of related data, of primitive or reference types, in the form of “key: value” pairs. These keys can be variables or functions and are called properties and methods, respectively, in the context of an **object**.

**EX:**

emp.id = 101;

emp.name = “john”;

emp.salary = 50000:

console.log(emp.id + “” + emp.name + “ “ + emp.salary);

Object constructor

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

The **this keyword** refers to the current object

**EX:**

this.id = id;

this.name = name;

this.salary = salary;

}

e=new emp(103,”john”,30000);

console.log(e.id+””+e.name+””+e.salary);

**FUNCTIONS**

**JavaScript functions** are used to perform operations. We can call JavaScript function many times to reuse the code.

#### Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don’t need to write many lines of code each time to perform a common task.

JavaScript Function Syntax

function functionName([arg1, arg2, ...argN]){

}

JavaScript Function Example

**<script>**

function msg(){

alert("hello! this is message");

}

**</script>**

**<input** type="button" onclick="msg()" value="call function"**/>**

Maximum And Minimum function

var computerscienceMarks = [57, 34, 56, 78, 98];

var mechMarks = [56, 77, 45, 33, ];

var electricMarks = [45, 65, 33, 23, 48, 12];

function getMaxMarks(marks) {

var maxValue = 0;

for(i=0; i < marks.length; i++){

if(marks[i] > maxValue) {

maxValue = marks[i];

}

}

return (maxValue);

}

var compDeptMaxiMarks = getMaxMarks(computerscienceMarks);

var mechDeptMaxiMarks = getMaxMarks(mechMarks);

var electricDeptMaxiMarks = getMaxMarks(electricMarks)

console.log(compDeptMaxiMarks);

console.log(mechDeptMaxiMarks);

console.log(electricDeptMaxiMarks);

Minimum

var computerscienceMarks = [57, 34, 56, 78, 98];

var mechMarks = [56, 77, 45, 33];

var electricMarks = [45, 65, 33, 23, 48, 12];

function getMinMarks(marks) {

var minValue = marks[0];

for(i=0; i<marks.length; i++){

if(marks[i] < minValue) {

minValue = marks[i];

}

}

return (minValue);

}

var compDeptMinMarks = getMinMarks(computerscienceMarks);

var mechDeptMinMarks = getMinMarks(mechMarks);

var electricDeptMinMarks = getMinMarks(electricMarks)

console.log(compDeptMinMarks);

console.log(mechDeptMinMarks);

console.log(electricDeptMinMarks);