Tutorial 2 JavaScript Basics

Introduction

This tutorial gives a quick overview of the syntax and features of JavaScript language for those who have not use the language before or whose knowledge of JavaScript is a bit rusty. Some aspects of the language such as event handling and timing will be covered when they are needed. Try to run the examples in this tutorial. To run them, in most case you only need to copy the scripts into the <script> tags of the following simple HTML document:

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

//put javascript code here

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

For example:

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

var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
document.write(fruits);
console.log(fruits);

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

Here, the first line of the script defines an array called `fruits`. The second line `document.write(fruits)` outputs the array to the HTML page. Of course, you can use other output methods such as `console.log(fruits)` to send the result to the console or use `alert(fruits)` to send it to a message box. Try them out.

JavaScript Variables

Naming convention

- Variable names must begin with a letter
- Variable names can also begin with $ and _
- Variable names are **case sensitive** (y and Y are different variables)

Declaring JavaScript variables

JavaScript variables are declared using the **var** keyword:

```
var carname;
```

Variable declared without a value will have the value **undefined**.

Declare and assign a value to the variable:

```
var carname="Volvo";
```

Many variables can be declared in one statement:

```
var lastname="Doe", age=30, job="carpenter";
```

or

```
        var lastname="Doe",
        age=30,
        job="carpenter";
```

Re-declaring JavaScript Variables

```
        var carname="Volvo";
        var carname; //now carname has a value of undefined.
```

JavaScript has dynamic types

The same JavaScript variable can be used as different types:

```
        var x;            // x is undefined
        var x = 5;        // x is a Number
        var x = "John";   // x is a String
```

JavaScript Data Types

Strings

A string is a series of characters like "John Doe". A string can be any text inside quotes. Single and double quotes are equivalent. E.g.,

```
        var carname="Volvo XC60";
        var carname='Volvo XC60';
        var answer="It's alright";           //apostrophes inside a string
        var answer="He is called 'Johnny'"; //single quotes inside a string
        var answer='He is called "Johnny"'; //double quotes inside a string
```

Numbers

JavaScript has only one type of numbers. Numbers can be written with, or without decimals. E.g.,

```
        var x1=34.00;       // Written with decimals
        var x2=34;          // Written without decimals
        var y=123e5;       // 12300000 in scientific (exponential) notation
        var z=123e-5;      // 0.00123  scientific notation
```

Booleans

Booleans can only have two values: `true` or `false`.

```
        var x=true;
        var y=false;
```

JavaScript Operators

Arithmetic operators

Arithmetic operators are used to perform arithmetic between variables and/or values

+, -, *, /	Normal arithmetic operators
%,	modulus. E.g., suppose y=5, then y%2 equals 1 (remiander)
++	Increment operator. `x = ++y` increase y by 1 first then assign it to x; `x=y++`, assign y to x first then crease y by 1
- -,	Decrement

Assignment operators

=	normal assignment operator
+=	x+= y is equivalent to x=x+y
-=	x-= y is equivalent to x=x-y
*=	x*= y is equivalent to x=x*y
/=	x/= y is equivalent to x=x/y
%=	x%= y is equivalent to x=x%y

String operator

The + operator is also used to concatenate strings. E.g.,

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2; // txt3 will be: What a verynice day" (no space between the two strings)
```

Note: adding a number and a string will return a string. E.g.,

```
x=5;            //numerical value 5
y="5"+5;        //string 55
z="Hello"+5;//string Hello5
```

Comparison operators
Comparison operators are used in relational expressions, e.g., x>y. The result is a logic value, `true` or `false`.

= =	Equal to
= = =	Exact equal to (equal value and equal type)
!=	Not equal to
!==	Not equal to (different value or different type)
>, <, >=, <=	

Logical operators

&&	Logic **AND**
||	Logic **OR**
!	**NOT**

E.g., suppose that **x=6 and y=3**, then
(x < 10 && y > 1) is true
(x==5 || y==5) is false
!(x==y) is true

Conditional operator
JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

```
variablename=(condition)? value1: value2
```

If *`condition`* is **`true`**, then *`value1`* is assigned to *`variablename`*, otherwise *`value2`* is assigned.

JavaScript Loops

for loops

```
for (statement1; statement2; statement3)
{
      the code block to be executed
}
```

`statement1` is executed ***before*** the loop body starts.
`statement2` defines the condition for running the loop. If omitted, the loop will never end.
`statement3` is executed each time ***after*** the loop body has been executed.

E.g.,

```
var cars=["Saab","Volvo","BMW"];
for (var i=0,len=cars.length; i<len; i++)// {
      document.write(cars[i] + "<br>");
}
```

statement1 can be omitted if the variable is initialised. E.g.,

```
var i=1;
for (; i<car.length; i++)
{
      document.write(cars[i] + "<br>");
}
```

Statement3 can also be omitted, but you need to have corresponding code inside the loop. E.g.,

```
        var i=0,len=cars.length;
        for (; i<len; )
        {
              document.write(cars[i] + "<br>");
              i++;
        }
```

for/in loop

The JavaScript `for-in` statement loops through the properties of an `object`. (An JavaScript object can many properties, i.e., name:value pairs) E.g.,

```
        var person={fname:"John",lname:"Doe",age:25};
        var txt='';
        for (var x in person)
        {
              txt=txt + person[x];
        }
```

While loop

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

do … while loop

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

JavaScript Arrays

The arrays are used to store multiple values in a single variable. JavaScript arrays are heterogeneous – you can have different objects in one array.

Create an Array

An array can be created in three ways.

1: Regular:

```
        var myCars=new Array();
        myCars[0]="Saab";
        myCars[1]="Volvo";
        myCars[2]="BMW";
```

2: Condensed:

```
        var myCars=new Array("Saab","Volvo","BMW");
```

3: Literal:

```
        var myCars=["Saab","Volvo","BMW"]; // notice the square brackets
```

Access an array element

The element of an array can be accessed using the array **index** number (0 is the index for the first element). E.g.,

```
        var name=myCars[0]; //Reads the first element in myCars:
        myCars[0]="Opel"; //modifies the first element in myCars:
```

All JavaScript variables are objects. Array elements, arrays themselves, functions are all objects. Because of this, you can have data of different types in the same Array. E.g.,

```
        myArray[0]=Date.now;     // a property of Date object
        myArray[1]=myFunction;  //a function
        myArray[2]=myCars;      //an array
```

Methods and properties of arrays

An array object has many predefined properties and methods, e.g.,

```
        var x=myCars.length         // get the property length of array myCars
        var y=myCars.indexOf("Volvo")   // call the method indexOf
```

Create new methods for an array

New properties and methods for any JavaScript Objects can be added by using a global constructor *Prototype* in JavaScript. E.g., add a method called **ucase** to the array objects:

```
        Array.prototype.ucase=function()
        {
              for (i=0; i<this.length; i++)
              {
                    this[i]=this[i].toUpperCase();
              }
        }
```

Frequently used array methods and examples

push(): Add new elements to the **end** of an array

```
        <!DOCTYPE html>
        <html>
        <body>
        <p id="demo">Click the button to add a new element to the array.</p>
        <button onclick="myFunction()">Try it</button>

        <script>
        var fruits = ["Banana", "Orange", "Apple", "Mango"];
        console.log(fruits);
        function myFunction()
        {
              fruits.push("Kiwi");
              var x=document.getElementById("demo");
              x.innerHTML=fruits;//write to page
              //console.log(fruits);//write to console
        }
        </script>
        </body>
        </html>
```

unshift(): Add new elements to the **beginning** of an array

```
        <!DOCTYPE html>
        <html>
        <body>
        <p id="demo">Click the button to add elements to the array.</p>
        <button onclick="myFunction()">Try it</button>

        <script>
        function myFunction()
        {
              var fruits = ["Banana", "Orange", "Apple", "Mango"];
              fruits.unshift("Lemon","Pineapple");
              var x=document.getElementById("demo");
              x.innerHTML=fruits;
              console.log(fruits);
        }
        </script>
```

```
<p><b>Note:</b> The unshift() method does not work properly in
Internet Explorer 8 and earlier.</p>
</body>
</html>
```

pop(): Remove the last element of an array

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Click the button to remove the last array element.</p>
<button onclick="myFunction()">Try it</button>
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
function myFunction()
{
      fruits.pop();
      var x=document.getElementById("demo");
      x.innerHTML=fruits;
      console.log(fruits);
}
</script>
</body>
</html>
```

shift(): Remove the first element of an array

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Click the button to remove the first element of the
array.</p>
<button onclick="myFunction()">Try it</button>
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
function myFunction()
{
      fruits.shift();
      var x=document.getElementById("demo");
      x.innerHTML=fruits;
      console.log(fruits);
}
</script>
</body>
</html>
```

slice(): Select elements from an array

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Click the button to extract the second and the third
elements from the array.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
      var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
      var citrus = fruits.slice(1,3);
      var x=document.getElementById("demo");
      x.innerHTML=citrus;
      console.log(citrus);
}
</script>
</body>
</html>
```

splice(): Add/Remove elements to an array to specified position position. E.g., add two elements to position 2 in an array (where 2 is the index where the new elements will be inserted to and 0 says that no existing element will be removed, i.e., insertion only):

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Click the button to add elements to the array.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
      var fruits = ["Banana", "Orange", "Apple", "Mango"];
      fruits.splice(2,0,"Lemon","Kiwi");
      var x=document.getElementById("demo");
      x.innerHTML=fruits;
      console.log(fruits);
}
</script>
</body>
</html>
```

The output will be

```
Banana, Orange, Lemon, Kiwi, Apple, Mango
```

The following statement adds two new items at position 2 and removes 1 item:

```
      fruits.splice(2,1,"Lemon","Kiwi");
```

The output will be

```
Banana, Orange, Lemon, Kiwi, Mango
```

Sort(): Sort an array (alphabetically and ascending)

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Click to sort the array.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
      var fruits = ["Banana", "Orange", "Apple", "Mango"];
      fruits.sort();
      var x=document.getElementById("demo");
      x.innerHTML=fruits;
      console.log(fruits);
}
</script>
</body>
</html>
```

JavaScript Objects

An object is just a special kind of data, with properties and methods. In JavaScript almost everything is an object: arrays, function, expressions, and dates are objects. Even primitive data types such as Booleans, Numbers and Strings can be treated as objects (except null and undefined). JavaScript is an object-oriented language, but it does not use classes. JavaScript objects are created from prototype instead of from classes as in most other object-oriented languages.

Accessing object methods and properties

Methods of an object are the operations that can be performed on the object. Properties are the values associated with an object. Methods and properties are accessed using "." Operator:

```
        objectName.propertyName
        objectName.methodName
```

E.g., to find the length property of the a string:

```
        var message="Hello World!";
        var x=message.length; //x will be 12
```

or, to convert a text to uppercase:

```
        var message="Hello world!";
        var x=message.toUpperCase();
```

Creating objects

There are 2 different ways to create a new object:

(1). Define and create a direct instance of an object. E.g.,

```
        person=new Object();
        person.firstname="John";
        person.lastname="Doe";
        person.age=50;
        person.eyecolor="blue";
```

An alternative way is to use object literals. Notice that the curly braces are used.

```
        person={firstname:"John",lastname:"Doe",age:50,eyecolor:"blue"};
```

(2) Use a function to define an object, e.g., an object called `person`, then use it to create new object instances.

```
        function person(firstname,lastname,age,eyecolor)
        {
                this.firstname=firstname;
                this.lastname=lastname;
                this.age=age;
                this.eyecolor=eyecolor;
        }
```

The keyword **`this`** refers to the current instance of the object, because we may have more than one instances of the object person at a time. Note that in the statement `this.firstname=firstname;` the `firstname` on the left hand side is the name of the property of the object, the `fistname` on the right hand side is the name of an argument of the function. Once defined, new instances can be created from an object, e.g., for the person object we create two instances:

```
        var myFather=new person("John","Doe",50,"blue");
        var myMother=new person("Sally","Rally",48,"green");
```

Adding properties to JavaScript objects

New properties can be added to an existing object **by simply giving it a value**. E.g., add a new property called `eyecolor` to the `person` object:

```
        person.firstname="John";
        person.lastname="Doe";
        person.age=30;
        person.eyecolor="blue";//a new property eyecolor is added to object person
```

Adding methods to JavaScript objects

Defining methods to an object is done inside the constructor function: Try this:

```
<!DOCTYPE html>
<html>
<body>
<script>
function person(firstname, lastname, age, eyecolor)
{
      this.firstname=firstname;
      this.lastname=lastname;
      this.age=age;
      this.eyecolor=eyecolor;

      this.changeName=changeName;//note: changeName does not have brackets
      function changeName(name)
      {
            this.lastname=name;
      }
}
myMother=new person("Sally","Rally",48,"green");
myMother.changeName("Doe");
document.write(myMother.lastname);

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