Programming Fundamentals

Reference Document for JavaScript Syntax

CONTENTS

Reference Document for JavaScript Syntax

1. VARIABLE

Syntax:

```
var variableName = value
```

Example:

```
var foo = 2
var fooBar =  “hello world”
```

2. PRINT STATEMENT

Syntax:

```
console.log(value/variable)
```

Example:

```
console.log(“Foo Bar”)
```

3. SELECTION

3.1. IF

Syntax:

```
if(condition){
   //block of statements
}
```

Example:

```
if(foo>3){
    console.log("foo is greater than 3")
}
```

3.2. IF ELSE

Syntax:

```
if(condition){
    //block of statements
}
else{
    //block of statements
}
```

Example:

```
if(foo > 3){
            console.log("foo is greater than 3")
}
else{
            console.log("foo is not greater than 3")
}
```

3.3. ELIF LADDER

Syntax:

```
if(condition){
    //block of statements
}
else if(condition){
    //block of statements
}
else{
    //block of statements
}
```

Example:

```
if(foo == 1){
     console.log("foo  equals 1")
}
else if(foo == 2){
     console.log("foo equals 2")
}
else{
    console.log("foo value is other than 1 and 2")
}
```

3.4. NESTED IF

Syntax:

```
if(condition){
    //block of statement
    if(condition){
              //block of statements
    }
    else{
             //block of statements
    }
else{
    //block of statements
}
```

Example:

```
if(foo > 0){
    if(foo > 30){
              console.log("foo is greater than 30")
     }
     else{
              console.log ("foo is not greater than 30")
      }
}
else{
       console.log ("foo is not greater than 0")

}
```

3.5. SWITCH CASE

Syntax:

```
switch(expression){
        case value1:
                        //statements
                        break
        case value2:
                       //statements
                        break
        ....
        case valueN:
                      //statements
                     break
        default:
                      //statements
}
```

Example:

```
var foo=10
switch(foo){
		case 5:
							console.log("Five")
							break
		case 10:
							console.log("Ten")
							break
		default:
							console.log("Invalid number")
}
```

4. ITERATION

4.1. WHILE LOOP

Syntax:

```
while (expression){
		//Statement(s) to be executed if expression is true
}
```

Example:

```
foo=5
while (foo < 10){
		console.log("Current Count : ",foo)
		foo++
}
```

4.2. FOR LOOP

Syntax-1:

```
for (initialization; test condition; iteration statement){
			//Statement(s) to be executed if test condition is true
}
```

Example-1:

```
for(fooBar=0; fooBar<5; fooBar++){
        console.log(fooBar)
}
```

5. BREAK

Syntax:

```
break
```

Example:

```
for(fooBar=0; fooBar<4; fooBar++){
    console.log(fooBar)
    if(fooBar==2){
       break
    }
}
```

6. CONTINUE

Syntax:

```
continue
```

Example:

```
for(fooBar=0; fooBar<4; fooBar++){
        if(fooBar == 1){
            continue
        }
        console.log(fooBar)
}
```

7. ARRAY

Syntax:

```
var arrayName=[value1, value2, … value n]
//or
var arrayName= new Array (value1, value2, … value n)
```

Example:

```
var foo= [1,2,3,4]
//or
var fooBar = new Array(1,2,3,4)
```

7.1. APPEND

Syntax:

```
arrayName.push(element)
```

Example:

```
var fooBar= [1,2,3,4]
fooBar.push(5)
```

7.2. SPLICE

Syntax:

```
arrayName.splice(index,numberOfElementsRemove,elementToInsert)
```

Example:

```
var fooBar= [1,2,3,4]
fooBar.splice(1,1,6)
```

7.3. REVERSE

Syntax:

```
arrayName.reverse()
```

Example:

```
var fooBar= [1,2,3,4]
fooBar.reverse()
```

8. LIBRARIES

8.1. STRING

Syntax:

```
variable.replace("old_string","new_string")
variable.search("string_to_find")
variable.startsWith("string_to_match")
variable. endsWith("string_to_match")
isNaN(variable)
variable.toUpperCase()
variable.toLowerCase()
variable.split("string_based_on_split")
variable.slice(startPosition,endPosition)
```

Example:

```
foo=”I love python”
foo.replace(“l”,”L”)
foo.search(“python”)
foo.startsWith(“I”)
foo. endsWith(“on”)
isNan(foo)
foo.toUpperCase()
foo.toLowerCase()
foo.split(“ ”)
foo.slice(2,5)
```

8.2. TIME

Syntax:

```
var foo=new Date()
var foo1=foo.toLocaleString()
var foo2=foo.getTimezoneOffset()
var foo3=foo.toGMTString()
```

Example:

```
var foo=new Date()
var foo1=foo.toLocaleString()
var foo2=foo.getTimezoneOffset()
var foo3=foo.toGMTString()
```

8.3. MATH

Syntax:

```
Math.ceil(decimal_value)
Math.floor(decimal_value)
Math.abs(decimal_value)
```

Example:

```
Math.ceil(9.6)
Math.floor(9.6)
Math.abs(9.6)
```

9. EXCEPTION

9.1. TRY-EXCEPT

Syntax:

```
try{
    //perform operations here
}
catch(e){
   //If there is any exception, then execute this block
}
```

Example:

```
try {
    functionName()
}
catch(e) {
   console.log("Not defined..")
}
```

9.2. TRY-EXCEPT-FINALLY

Syntax:

```
try{
      //Perform operations here
}
catch(e){
       //If there is any exception, then execute this block
}
finally{
       //This would always be executed
}
```

Example:

```
try {
    functionName()
}
catch(e) {
   console.log("Not defined..")
}
finally{
   console.log("Program is terminating")
}
```

11. FUNCTION

Syntax:

```
function functionName(parameterList){
       //function body
       [return]
}

functionName(values)
```

Example:

```
function  sum(foo,fooBar){
        console.log(foo+fooBar)
}

sum(5,5)
```

9.1. POSITIONAL ARGUMENTS

Syntax:

```
function functionName(parameter1,parameter2){
        //Function body
        [return]
}


functionName(value1,value2)
```

Example:

```
function  sum(foo,fooBar){
        console.log(foo+fooBar)
}
```

9.2 VARIABLE NUMBER OF ARGUMENTS

Syntax:

```
var functionName=function(){
              //Function body
              [return]
}

function_name(value1/value1,value2)
```

Example:

```
var  sum=function(){
       for (i=0;i<arguments.length;i++){
              console.log(arguments[i])
       }
}

sum(2,4,6)
//or
sum(1,2)
```

10. VARIABLE SCOPE

10.1. GLOBAL VARIABLE

Syntax:

```
variable1=value            //Global access, can be accessible anywhere.

function functionName(){
      //function body
      [return]
}
```

Example:

```
foo=100

function function1(){
     foo+=1
}

console.log(foo)
function1()
console.log(foo)
```

10.2. LOCAL VARIABLE

Syntax:

```
function functionName(){
       variable1=value //Local access, can accessible only inside this function.
}
```

Example:

```
function function1(){
      foo=100
      foo+=1
}

function1()
console.log(foo)  //This statement will give an error as variable,foo is  local to
function1
```