

## Lecture 2

In this lecture we will learn

1. Arrays
2. Expressions and Operators
3. Functions
4. If-else construct
5. Switch Construct
6. loop constructs
  - For
  - While
  - do-while

### More about data types

As I told in my previous lecture, we can declare a variable by

`var variable name;`

for example:

```
var x;
```

The variable 'x' can store integers, characters, string, float, boolean.

There are five types of data or five data types in Javascript.

Integer

character

string

float

boolean

For all the data types, there is a common keyword **var**.

**boolean** values are only true or false.

**String** is a sequence (연달아 일어남, 연속) of characters enclosed (둘러싸다) inside double quotes (" "). For example "hello" .

**Integer** is a numeric (수) value without decimal. For example 2

**Float** is a numeric value with decimal. For example 2.5 or 2.0

**character** is a single character enclosed (둘러싸다) inside single quote (' '). For example 'a'.

## Arrays

Like other programming languages, Javascript supports creating arrays.

An array is a collection of data of same type.

By data of same type, I mean, all the data can be either integer or character or float or string or boolean but not a combination (결합, 짝맞춤, 배합) of either (공정문에서) (둘 중) ) of these.

Like C language, starting index is 0.

For example:

```
<body>
  <script language="JavaScript">
    <!--
      var myArray = new Array (5);
      myArray [0] = 0;
      myArray [1] = 1;
      myArray [2] = 2;
      myArray [3] = 3;
      myArray [4] = 4;
      myArray [5] = 5;

      var secondArray = new Array(6,7,8,9);

      // printing the elements of the first array
      document.write("The elements of myArray are <br>");
      document.write(myArray[0] + "<br>");
      document.write(myArray[1] + "<br>");
      document.write(myArray[2] + "<br>");
      document.write(myArray[3] + "<br>");
      document.write(myArray[4] + "<br>");
      document.write(myArray[5] + "<br>");

      printing the elements of the second array
      document.write("The elements of SecondArray are <br>");
```

```
document.write(secondArray[0] + "<br>");
document.write(secondArray[1] + "<br>");
document.write(secondArray[2] + "<br>");
document.write(secondArray[3] + "<br>");

// -->
</script>
</body>
```

There are 2 ways of declaring arrays in the above example.

1. `var array_name = new Array (size of array);`  
`array_name[0] = value1;`  
`array_name[1] = value2;`  
`array_name[2] = value3;`  
.  
.  
.  
`array_name[size of array] = valuen;`
2. `var array_name = new Array (value1, value12 , value3 ,.....,valuen);`

## **Expressions and Operators**

Expression is a sequence (연달아 일어남, 연속) of operators and operands.

For example: `2 + 3` .

The above expression consist of 2 operands and 1 operator.

- `+` is an operator
- 2 and 3 are operands.

Another example: `z = x + y`

The above expression consist of 3 operands and 2 operators.

- `+` and `=` are operators. (`=` is called as assignment operator)
- `x`, `y` and `z` are operands.

There are 4 types of expressions.

1. Arithmetic expression
2. Logical expression
3. String Expression
4. Relational expression

## **Arithmetic Expression**

Expression formed using arithmetic operator are called as **arithmetic expression**. There are 5 types of **Arithmetic operators**.

1. addition (+)
2. Subtraction (-)
3. Multiplication (\*)
4. Division (/) - gives the quotient of division operation.
5. modulus (%) - gives the remainder of division operation

## **Relational expression**

Expressions formed using comparison operator are called as **relational expression**. There are 6 types of **Comparison operator**.

1. equal to ( == )
2. not equal to ( != )
3. Less than ( < )
4. Less than or equal to ( <= )
5. greater than ( > )
6. greater than or equal to ( >= )

Return value of a relational expression is boolean i.e. true or false.

## **Logical Expression**

Logical expressions are used to combine (**결합, 짝맞춤, 배합**) different comparisons. There are 2 types of **logical operators**.

1. Logical AND (&&)
2. Logical OR (||)

For example:

```
a = (2==3);           ----- (1)
c = (2 != 3);        ----- (2)
e = a && c             ----- (3)
```

Equation (1) and (2) are relational expression.  
Equation (3) is a **logical expression**.

Logical expression also return boolean values i.e. True or false.

## **String Expression**

Expression formed using Strings and '+' operator are called as **String expression**. '+' operator has a dual (**둘의; 둘을 나타내는, 양자의**) functionality. When '+'

operator is used with numbers it functions as an arithmetic operator. When '+' operator is used with strings it functions as a concatenation operator.

Example of '+' as concatenation operator:

```
var name = " huh";  
var greeting = "Welcome";  
var welcome = greeting + name;
```

welcome variable contains "Welcome huh".

## **Operator Precedence(앞섬)**

Arithmetic operators has the highest priority (앞섬). After arithmetic operator comes comparison operator and then comes logical operator.

Within ((장소·시간·거리·범위·한계 등)) each group, operators have priority or precedence. For example, within the arithmetic operators, '%' has highest priority, followed by '\*' and '/' and then at the end is '+' and '-'.

The priority of arithmetic operator is

1. '%'
2. '\*', '/'
3. '+', '-'

Remember these rules

1. Operators are executed in the order of their priority, i.e. operators with highest priority are executed first.
2. Operators having same priority are executed in Left to right order.
3. you can override (보다 우위에 서다) operator priority by enclosing (둘러싸다) an expression inside parenthesis (괄호, 소괄호).

For example:

```
var c = 5/9 * F - 32; ----- (1)
```

In the above expression we are converting (변하게 하다, 전환하다) Fahrenheit (화씨의) to Celsius (섭씨의).

If F = 50, then

```
var c = 5/9 * 50 - 32;
```

There are 3 operators on RHS: '/', '\*', '-'. Among (...의 사이에, ...의 가운데에, ...에 들

러싸여) '/', '\*', '-' operators, '/' and '\*' have highest priority. That means either of '/' or '\*' will be executed first.

'/' and '\*' have equal priority, that means, they will be evaluated from left to right.

In equation (1), '/' appear on the left, so first '/' operation is executed. That means  $5/9$  is executed first, and  $5/9 = 0.556$

$$\text{var } c = 0.556 * 50 - 32$$

After '/' comes '\*', that means  $0.556 * 50 = 27.8$

$$\text{var } c = 27.8 - 32$$

And then comes '-', that means  $27.8 - 32 = -4.2$

$$\text{var } c = -4.2$$

If we apply parenthesis around  $F - 32$  i.e

$$\text{var } c = 5 / 9 * (F - 32) \quad \text{----- (2)}$$

If  $F = 50$ , then

$$\text{var } c = 5 / 9 * (50 - 32)$$

Rule 3 is applied first, parenthesis are always evaluated first, that means  $(50 - 32) = 18$ .

$$\text{var } c = 5 / 9 * 18$$

Now rule 2 is applied, because '/' and '\*' have same priority. This means, execute from left to right.

$$\text{var } c = 0.556 * 18$$

finally answer is

$$\text{var } c = 10.$$

## **Functions**

Like many other programming languages, Javascript also support functions. A function is a self (자기, 자신) contained (억제[자제]하는, 조심스러운) operation that can be called by function name. If function expects some input parameters (【수학·컴퓨터】 파라미터, 매개 변수) then pass few parameters also.

To call a function, you simply use the following form:

```
functionName (argument1, argument2, argument3,.....);
```

If a function has no input arguments, you still need parenthesis , for example:

```
functionName ();
```

Also, if a function returns some value, you can use the function call to assign ( **배당하다** ) the returned value to some variable, for example:

```
var returnValue = functionName ();
```

Functions are of two types,

1. System Defined Functions
2. User Defined Functions

System defined functions are those which are provided by the system, whereas User defined functions are those for which are made by the user.

An example of System defined function

```
<body>
  <script language="Javascript">
    <!--
      window.alert("How are you");
    // -->
  </script>
</body>
```

Output should be a pop-up dialog with message, "How are you".

The above Javascript code, uses a system defined function, `window.alert()`.

Window is an object. Function `alert()` is provided by window object.

### **Confirming with User**

There is another function, which you can be used to ask a question from user, and get his response ( **대답** ) as True or False. The name of the function is `window.confirm()`; Window object provides `confirm()` function, which displays a dialog box containing text message, and provides two options, "OK" and "Cancel".

This function also returns true or false, depending upon what user clicks. If user clicks "OK", `confirm()` returns true, and if user clicks "Cancel", then `confirm()` returns false.

For example:

```
<body>
  <script language="JavaScript">
    <!--
      var result=window.confirm("How are you");
      document.write(result);
    // -->
  </script>
</body>
```

Output should be "true", if you click "Ok" or "false", if you click "Cancel".

### **Taking input from User**

Sometimes we may need to take some input from the user. We can ask a user to enter his name or to enter a number. Javascript provides a function `window.prompt()`. 'window' is an object in Javascript and `prompt()` is a function of window object.

To understand `window.prompt()`, here is an example:

```
<body>
  <script language="Javascript">
    var name = window.prompt("enter your name");
    window.alert("welcome " + name);
  </script>
</body>
```

### **Creating your own Functions**

Like I told above, functions are of two types, System defined functions and User defined functions. We have already discussed about System Defined Functions, Now we will discuss about User defined functions.

Creating a User Defined function goes like this

```
function functionName()
{
```

```
statement1;  
statement2;  
.  
.  
.  
statementn;  
}
```

For example:

```
<HTML>  
<HEAD>  
  <script language="JavaScript">  
    function displayHello()  
    {  
      document.write("Hello");  
    }  
  </script>  
</HEAD>  
<BODY>  
  
  <script language="JavaScript">  
    displayHello();  
  </script>  
  
</BODY>  
</HTML>
```

In the above program, I have declared a function "displayHello()" in the head section of the document. Function "displayHello()" is called in the body section of the document.

### **Passing an Argument to Your Functions**

Arguments are needed in a function to pass (유월절) the

values. To create a function that accepts arguments, you must specify names for each argument in the argument definition.

Syntax of function declaration with input arguments,

```
function functionName(argumentName1)
{
    statement1;
    statement2;
    .
    .
    .
    statementn;
}
```

For example:

```
<HTML>
<HEAD>
    <script language="JavaScript">
        function square(num)
        {
            var result=num * num;
            document.write(result);
        }
    </script>
</HEAD>
<BODY>

    <script language="JavaScript">
        square(10);
    </script>

</BODY>
</HTML>
```

## **Return Values From Function**

Now, we will discuss another example where function will return a value. In the previous example, our function received a number, the square of that number was computed, and displayed. But now, our function will receive a number, compute the square and return the square back.

To return a value, **return** command is used.

Syntax:

```
function functionName()  
{  
    statement1;  
    statement2;  
    .  
    .  
    statementn;  
    return value;  
}
```

For example:

```
<HTML>  
<HEAD>  
    <script language="JavaScript">  
        function square(num)  
        {  
            var result=num * num;  
            return result  
        }  
    </script>  
</HEAD>  
<BODY>
```

```
<script language="JavaScript">
    var ans=square(10);
    document.write(ans);
</script>
```

```
</BODY>
</HTML>
```

## Passing Multiple Parameters to Your Function

In the Example above, we discussed about passing single argument, now we will discuss about how to pass more than one argument. It is very simple similar to the code above.

```
function functionName(argument1, argument2, argument3,.....)
```

We will write a program that passes two arguments to a function, that function multiplies the value contained in those two arguments, and returns back the result.

```
<HTML>
<HEAD>
    <script language="JavaScript">
        function multiply(num1,num2)
        {
            var result=num1 * num2;
            return result;
        }
    </script>
</HEAD>
<BODY>

    <script language="JavaScript">
        var ans=multiply(10,20);
        document.write(ans);
```

```
</script>
```

```
</BODY>
```

```
</HTML>
```

## **If-Construct**

Syntax

```
if (condition or comparison expression)
```

```
{
```

```
Statement1;
```

```
Statement2;
```

```
.
```

```
.
```

```
.
```

```
Statementn;
```

```
}
```

```
else
```

```
{
```

```
Statement11;
```

```
Statement12;
```

```
.
```

```
.
```

```
.
```

```
Statement1n;
```

```
}
```

If condition evaluates (평가하다) to true, first block is executed. If condition evaluates to false, second block is executed.

For example:

```
x=2;  
y=3;  
if (x == y)  
{  
    document.write ("x and y are equal");
```

```
}
else
{
    document.write("x and y are not equal");
}
```

Another example of If-construct

In this example, we will ask the user for his Id and password. If he enters correct Id and password, we will welcome him, otherwise an error message is displayed.

```
<html>
<head>
<script language="javascript">

var username="user";
var password="haveaniceday";
var entered_username=prompt("enter username: ", " ");
var entered_password=prompt("enter password: ", " ");

if ((entered_username == username) && (entered_password == password))
{
    document.write("You have been Logged in!")
}
else
{
    document.write("Sorry, either the username or password is incorrect.")
}

</script>
</head>

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

## **Switch-Case Construct**

If we want to use multiple if, which we do when have lot of conditions to check and do something. For example, If there are 3 types of user: administrator, developer, and user.

If user enters his password, we display "you have user access".

If administrator enters his password, we display, "you have administrator access"

If developer enters his password, we display, "You have developer access"

In the above scenario ( **【연극】 대본** ), we need to use more than 1 If-else. Using more than 1 if-else can sometimes become clumsy. We can use switch-case instead ( **그 대신에** ) of using more than one if-else.

For example

```
<html>
<head>
<script language="javascript">

var password_administrator="ice";
var password_developer="snow";
var password_user="water";

var entered_password=prompt("enter the secret code to continue: ", " ");

    switch (entered_password) {
        case "ice":
alert("you have administrator access");
        break
        case "snow":
alert("you have developer access");
        break
        case "water":
alert("you have user access");
        break
        default:
alert("password incorrect!");
        break
    }

</script>
</head>

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

## **Looping statements**

loops are used when we want a part of code to be repeated ( **되풀이하다, 반복하다** ) till the condition is true. As the condition becomes false, loop terminates ( **<행동>상** )

태 등을> 끝낸다 ).

There are three types of looping constructs available in Javascript.

1. For
2. While
3. do-while

Syntax of For:

```
for (initialize statement; condition; increment/decrement statemtn)
{
    statement1;
    statement2;
    .
    .
    .
    statementn.
}
```

An example of for loop.

```
<html>
  <head>
    <title>Associating a Function with a Link</title>
    <script language="JavaScript">
      <!--
          for(var i=0;i<10;i++)
              document.write(i+"<br>");
      // -->
    </script>
  </head>
  <body">

  </body>
</html>
```

Output for the above code is display numbers from 0 to 9.

Syntax of while loop

```
initialize statement;
while (condition)
{
    statement1;
    statement2;
    .
    .
    .
    statementn;
    increment/decrement statement;
}
```

Following is a while equivalent of the above code.

```
<html>
  <head>
    <title>Associating a Function with a Link</title>
    <script language="JavaScript">
      <!--
        var i=0;
        while (i<10)
        {
            document.write(i+"<br>");
            i=i+1;
        }
      // -->
    </script>
  </head>
</body>
</html>
```

Exercise:

What is the output of the following program?

```
<html>
```

```

<head>
<script language="javascript">

var hour;
var ampm;

for (hour=1; hour<24; hour++)
{
  if (hour < 12) {ampm="am"}
  if (hour >= 12) {ampm="pm"}

  if (hour < 13)
  {
    document.write(hour + ampm)
  }
  else
  {
    document.write((hour-12) + ampm)
  }
  document.write("<br>");
}

</script>
</head>

<body>
</body>
</html>

```

In do-while, a block of code is executed first before testing the condition. Syntax is

```

do
{
    statement1;
    statement2;
    .
    .
    .
    statementn;
} (condition);

```

Statements inside do-while are executed at least once irrespective (보통 다음 성구로) of whether (간접의문문의 명사절을 이끌어 ...인지 어떤지) condition evaluates to true or false.

Here is an example that uses do-while loop, In this example, we ask the user to enter any number between 1 - 12, and 0 to exit.

The output is name of month corresponding (상응하는, 일치하는, 대응하는) to a number. i.e. If user enters 1, output is January. if user enters 2, output is February. If user enters 12, output is December.

```
<html>
<head>
<script language="javascript">

var requested_month=0;
var MonthArray=
[
  "",
  "January",
  "February",
  "March",
  "April",
  "May",
  "June",
  "July",
  "August",
  "September",
  "October",
  "November",
  "December"
]

document.write("Enter a a number between 1 and 12, and I will tell you the name
of the month.<br>");
do
{
requested_month=prompt("Enter a number between 1 and 12 (or 0 to exit)", 0);
document.write(MonthArray[requested_month] + "<BR>");
} while (requested_month != "0")

</script>
</head>

<body>
</body>
```

Exercise:

Write the above program with using while loop.