IT Workshop

[2ES111]

Computer Engineering & Information Technology

Semester- 2

U V Patel College of Engineering

2. JavaScript Basic:

JS Introduction, JS Output, JS Statements, JS Syntax, JS Comments, JS Variables, JS Operators, JS Data Types

JS Introduction:

JavaScript is a very powerful **client-side scripting language**. JavaScript is used mainly for enhancing the interaction of a user with the webpage. *JavaScript* was initially created to *"make web pages alive"*. The programs in this language are called ***scripts***. you can make your webpage more lively and interactive, with the help of JavaScript. JavaScript is also being used widely in game development and Mobile application development. **JavaScript and Java are very much unrelated. Java is a very complex programming language whereas JavaScript is only a scripting language**. The syntax of JavaScript is mostly influenced by the programming language C.

JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called **the JavaScript engine.** The browser has an embedded engine sometimes called a "**JavaScript virtual machine**".

Different engines have different "codenames". For example:

- **V8** – in Chrome and Opera.
- **SpiderMonkey** – in Firefox.

➢ **How do engines work?**

Engines are complicated. But the basics are easy.

- The engine (embedded if it's a browser) reads ("parses") the script.
- Then it converts ("compiles") the script to the machine language.
- And then the machine code runs, pretty fast.

➢ **What can in-browser JavaScript do?**

- Add new HTML to the page, change the existing content, modify styles.
- React to user actions, run on mouse clicks, pointer movements, key presses.
- Send requests over the network to remote servers, download and upload files (so-called AJAX and COMET technologies).
- Get and set cookies, ask questions to the visitor, show messages.
- Remember the data on the client-side ("local storage").

- **Tools You Need:**

  To start with JS, you need a text editor to write your code and a browser to display the web pages you develop. You can use a text editor of your choice including **Notepad++, Visual Studio Code, Sublime Text, Atom** or any other text editor you are comfortable with. You can use any web browser including **Google Chrome, Firefox, Microsoft Edge, Internet Explorer** etc.

- **How to Run JavaScript:**

  The browser is responsible for running JavaScript code. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it. The main advantage of JavaScript is that **all modern web browsers support** JavaScript. So, you do not have to worry about whether your site visitor uses Internet Explorer, Google Chrome, Firefox or any other browser. JavaScript will be supported. Also, JavaScript **runs on any operating system** including Windows, Linux or Mac.

- **A simple JavaScript Program:**

  You should place all your JavaScript code within **<script> tags** (<script> and </script>) if you are keeping your JavaScript code within the HTML document itself. or you can also make script file with **.js** extension.

  Code:

```
<html>
<head>
    <title>My First JavaScript code!!!</title>
    <script type="text/javascript">
            alert("Hello World!");
    </script>
</head>
<body>
</body>
</html>
```

JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

JS Output:

There are four different ways to display the output in JavaScript.

1. By getting **id** of html element you can display the output of JS.
   Example:

   Code:

```
<html>
<body>
<p id="display"></p>
<script>
  document.getElementById("display").innerHTML = 10 + 10;
</script>
</body>
</html>
```

   Output:

```
20.
```

2. Displaying the output using document.write().
   Code:

```
<script>
  document.write(5 + 3);
</script>
```

   Output:

```
8
```

3. Displaying the output through console.log()
   Code:

```
<script>
  console.log(2+ 3);
</script>
```

   In debugger console, the output:

```
5
```

4. Displaying the output through alert box:
   Code:

```
<script>
  alert(2+ 3);
</script>
```

   Output:

```
5
```

JS Statements:

A **JavaScript program** is a list of programming **statements**. JavaScript statements are composed of:Values, Operators, Expressions, Keywords, and Comments.

Example:

```
document.getElementById("demo").innerHTML = "Hello Dolly.";
```

This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo".

The statements are executed, one by one, in the same order as they are written. where each statements are separated by **semicolon ";"**

JS Keywords:

JavaScript statements often start with a **keyword** to identify the JavaScript action to be performed.

Keyword	Description
break	Terminates a switch or a loop
continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript, and calls (if available) the debugging function
do ... while	Executes a block of statements, and repeats the block, while a condition is true
for	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements
var	Declares a variable

JS Syntax:

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page. You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags. The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

syntax:

```
<script ...>
  JavaScript code
</script>
```

The script tag takes two important attributes −

- **Language** − This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.

- **Type** − This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

JS Comments:

The **JavaScript comments** are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code.

The JavaScript comment is ignored by the JavaScript engine i.e. embedded in the browser.

- **Advantages of JavaScript comments:**

There are mainly two advantages of JavaScript comments.

1. **To make code easy to understand** It can be used to elaborate the code so that end user can easily understand the code.
2. **To avoid the unnecessary code** It can also be used to avoid the code being executed. Sometimes, we add the code to perform some action. But after sometime, there may be need to disable the code. In such case, it is better to use comments.

- **Types of JavaScript Comments**

There are two types of comments in JavaScript.

1. Single-line Comment:

   It is represented by double forward slashes (//). It can be used before and after the statement.

   Example::

```
<script>
// It is single line comment
document.write("hello javascript");
</script>
```

2. Multi-line Comment:

   It can be used to add single as well as multi line comments. So, it is more convenient. It is represented by forward slash with asterisk then asterisk with forward slash. It can be used before, after and middle of the statement.

   Example:

Code:

```
<script>
/* It is multi line comment.
It will not be displayed */
document.write("example of javascript multiline comment");
</script>
```

JS Variables:

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

Correct JavaScript variables

```
var x = 10;
var _value="sonoo";
```

Incorrect JavaScript variables

```
var  123=30;
var *aa=320;
```

Example:

Code:

```
<script>
var x = 10;
var y = 20;
var z=x+y;
document.write(z);
</script>
```

Output:

```
30
```

JS local variable and global variable:

JS local variable: A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.

For example:

```
<script>
function abc(){
var x=10;//local variable
}
</script>
```

JS global variable: It is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable For example:

```
<script>
var value=50;//global variable
function a(){
alert(value);
}
function b(){
alert(value);
}
</script>
```

Declaring JavaScript global variable within function:

To declare JavaScript global variables inside function, you need to use **window object**. For example:

window.value=90;

Now it can be declared inside any function and can be accessed from any function. For example:

```
function m(){
window.value=100;//declaring global variable by window object
}
function n(){
alert(window.value);//accessing global variable from other function
}
```

Internals of global variable in JavaScript:

When you declare a variable outside the function, it is added in the window object internally. You can access it through window object also. For example:

```
var value=50;
function a(){
alert(window.value);//accessing global variable
}
```

JS Operators:

JavaScript operators are symbols that are used to perform operations on operands. For example:

var sum=10+20;

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

JavaScript Arithmetic Operators:
Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

JavaScript comparison Operators:

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

JavaScript Bitwise Operators:

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

JavaScript Logical Operators:

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

JavaScript Assignment Operators:

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

JavaScript Special Operators:

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.

JS Datatypes:

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

```
var a=40;//holding number
var b="Rahul";//holding string
```

JavaScript primitive data types:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript non-primitive data types:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

Type Conversions:

Most of the time, operators and functions automatically convert a value to the right type. That's called "type conversion".
For example, alert automatically converts any value to a string to show it. Mathematical operations convert values to numbers.
There are also cases when we need to explicitly convert a value to put things right.

- **ToString**

String conversion happens when we need the string form of a value.
For example, alert(value) does it to show the value.

We can also use a call String(value) function for that:

```
var value = true;
alert(typeof value); // boolean

value = String(value); // now value is a string "true"
alert(typeof value); // string
```

String conversion is mostly obvious. A false becomes "false", null becomes "null" etc.

- ToNumber

Numeric conversion happens in mathematical functions and expressions automatically.
For example, when division / is applied to non-numbers:

```
alert( "6" / "2" ); // 3, strings are converted to numbers
```

We can use a Number(value) function to explicitly convert a value:

```
var str = "123";
alert(typeof str); // string
```

```
var num = Number(str); // becomes a number 123
alert(typeof num); // number
```

Explicit conversion is usually required when we read a value from a string-based source like a text form, but we expect a number to be entered.

If the string is not a valid number, the result of such conversion is NaN, for instance:

```
let age = Number("an arbitrary string instead of a number");
alert(age); // NaN, conversion failed
```

Numeric conversion rules:

Value	Becomes...
Undefined	NaN
Null	0
True and false	1 and 0

- **ToBoolean**

Boolean conversion is the simplest one.
It happens in logical operations (later we'll meet condition tests and other kinds of them), but also can be performed manually with the call of Boolean(value).
The conversion rule:

- Values that are intuitively "empty", like 0, an empty string, null, undefined and NaN become false.
- Other values become true.

For instance:

```
alert( Boolean(1) ); // true
alert( Boolean(0) ); // false
```

```
alert( Boolean("hello") ); // true
alert( Boolean("") ); // false
```

Please note: the string with zero `"0"` is `true`
Some languages (namely PHP) treat "0" as false. But in JavaScript a non-empty string is always true.

```
alert( Boolean("0") ); // true
alert( Boolean(" ") ); // spaces, also true (any non-empty string is true)
```

Exercise:

1. **Write "Hello world" program with "hello.js" as an external file.**

2. **Define the following terms:**
   - **JS Statements**
   - **JS Keywords**
   - **JS Syntax**
   - **JS Comments**

3. Write syntax of JS variable to perform addition operation.

4. **Perform the following tasks in sequence:**
   - Declare two variables: admin and name.
   - Assign the value "Your Name (Write your name)" to name.
   - Copy the value from name to admin.
   - Show the value of admin using alert (must output “Your Name”).

5. **Write output of following code:**

```
<script>
        var
            one
        =


        1,
                two
                =
                "two"


                document.getElementById("p1").innerHTML = one;
                document.getElementById("p2").innerHTML = two;
    </script>
```

6. Write output of following code:

```
<script type="text/javascript">
var one = 22;
var two = 3;
var add = one + two;
var minus = one - two;
var multiply = one * two;
var divide = one/two;
        document.write("First No: = " + one + "<br />Second No: = " + two + " <br />");
        document.write(one + " + " + two + " = " + add + "<br/>");
        document.write(one + " - " + two + " = " + minus + "<br/>");
        document.write(one + " * " + two + " = " + multiply + "<br/>");
        document.write(one + " / " + two + " = " + divide + "<br/>");
</script>
```

7. **Write a JS to perform mathematical operation by assigning the variable as a local and global variable.**

8. **Write output of following JS:**

```
<script>
var x = 16 + 4 + "Volvo";
document.getElementById("demo").innerHTML = x;
</script>
```

and

```
<script>
var x = "Volvo"+16 + 4 ;
document.getElementById("demo").innerHTML = x;
</script>
```

9. **Write output of following JS:**

```
<script>
                var myVar = 100;
                myVar = true;
                myVar = null;
                myVar = undefined;
                myVar = "Steve";

                alert(myVar);
  </script>
```

10. What are results of these expressions?

```
"" + 1 + 0
"" - 1 + 0
true + false
6 / "3"
"2" * "3"
4 + 5 + "px"
"$" + 4 + 5
"4" - 2
"4px" - 2
7 / 0
"  -9\n" + 5
"  -9\n" - 5
null + 1
undefined + 1
```