10. Introduction to Javascript

Table of Content

1. Basics of Javascript

JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and much more.

▼ **Where to put**

In HTML, JavaScript code is inserted between `<script>` and `</script>` tags.

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

You can place any number of scripts in an HTML document.

Scripts can be placed in the `<body>`, or in the `<head>` section of an HTML page, or in both.

We can also create separate External Javascript.

```
<script src="myScript1.js"></script>
```

▼ Keywords

In JavaScript you cannot use these reserved words as variables, labels, or function names:

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

▼ Javascript Statements

A **computer program** is a list of "instructions" to be "executed" by a computer.

In a programming language, these programming instructions are called **statements**.

A **JavaScript program** is a list of programming **statements**.

```
let a, b, c;  // Declare 3 variables
a = 5;        // Assign the value 5 to a
b = 6;        // Assign the value 6 to b
c = a + b;    // Assign the sum of a and b to c
```

2. Variables & Data Types

▼ Variables

Variables are containers for storing data (storing data values). We can declare a variable by using these keywords.

- Using `var` for declaring function-scoped variables (old)
- Using `let` for declaring block-scoped variables (new)
- Using `const` for declaring constant variables

Note: It is recommended we use let instead of var. However, there are a few browsers that do not support let.

Rules for naming variables:

1. Variable names must start with either a letter, an underscore `_`, or the dollar sign `$`. For example,

```
//valid
let a = 'hello';
let _a = 'hello';
let $a = 'hello';
```

2. Variable names cannot start with numbers. For example,

```
//invalid
Let 1a = 'hello'; // this gives an error
```

3. JavaScript is case-sensitive. For example,

```
let y = "hi";
let Y = 5;

console.log(y); // hi
console.log(Y); // 5
```

4. Keywords cannot be used as variable names. For example,

```
//invalid
let new = 5; // Error! new is a keyword.
```

▼ Data Types

A data type, in programming, is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error.

A string, for example, is a data type that is used to classify text and an integer is a data type used to classify whole numbers.

Some Common Data Types in Javascript are:

- `String` for “Hello”, ‘hi’ etc.
- `Number` for 45, -12 etc.
- `Boolean` for true and false
- `undefined` for un-initialized variables
- `Object` key-value pairs of collection of data

▼ typeof operator

To find the type of a variable, you can use the `typeof` operator. For example,

```
const name = 'ram';
typeof(name); // returns "string"

const number = 4;
typeof(number); //returns "number"

const valueChecked = true;
typeof(valueChecked); //returns "boolean"
```

3. Javascript Type Conversion

We use these functions to convert types:

- `Number()`
- `String()`
- `Boolean()`

Note:

1. JavaScript considers 0 as false and all non-zero numbers as true. And, if true is converted to a number, the result is always 1.
2. String() takes `null` and `undefined` and converts them to string.
3. In JavaScript, `undefined`, `null`, `0`, `NaN`, `''` converts to false. All other values give `true`.

Use this table for reference:

Value	String Conversion	Number Conversion	Boolean Conversion
1	"1"	1	`true`
0	"0"	0	`false`
"1"	"1"	1	`true`
"0"	`"0"`	0	`true`
"ten"	`"ten"`	`NaN`	`true`
`true`	"true"	1	`true`
`false`	"false"	0	`false`
`null`	"null"	0	`false`
`undefined`	"undefined"	`NaN`	`false`
"	""	0	`false`
''	" "	0	`true`

4. Javascript Arithmetic Operators

As with algebra, you can do arithmetic with JavaScript variables, using operators like = and +

```
const number = 3 + 5; // 8
```

We have Arithmetic Operators : +, -, /, *, ++, — and **

5. Conditional Statements in Javascript

▼ Javascript Comparison Operators

Operator	Description	Example
`==`	**Equal to:** `true` if the operands are equal	`5==5; //true`
`!=`	**Not equal to:** `true` if the operands are not equal	`5!=5; //false`
`===`	**Strict equal to:** `true` if the operands are equal and of the same type	`5==='5'; //false`
`!==`	**Strict not equal to:** `true` if the operands are equal but of different type or not equal at all	`5!=='5'; //true`
`>`	**Greater than:** `true` if the left operand is greater than the right operand	`3>2; //true`
`>=`	**Greater than or equal to:** `true` if the left operand is greater than or equal to the right operand	`3>=3; //true`
`<`	**Less than:** `true` if the left operand is less than the right operand	`3<2; //false`
`<=`	**Less than or equal to:** `true` if the left operand is less than or equal to the right operand	`2<=2; //true`

▼ ternary Operator

A ternary operator evaluates a condition and executes a block of code based on the condition.

Its syntax is:

```
condition ? expression1 : expression2
let result = (marks >= 40) ? 'pass' : 'fail';
```

The ternary operator evaluates the test condition.

- If the condition is `true`, **expression1** is executed.
- If the condition is `false`, **expression2** is executed.

The ternary operator takes **three** operands, hence, the name ternary operator. It is also known as a conditional operator.

▼ If-else, else-if

In computer programming, there may arise situations where you have to run a block of code among more than one alternatives. For example, assigning grades **A**, **B** or **C** based on marks obtained by a student.

In such situations, you can use the JavaScript `if...else` statement to create a program that can make decisions.

```
if (condition) {
    // block of code if condition is true
} else {
   // block of code if condition is false
}
```

You can also write multiple `else if` in between the `if` and the `else` blocks.

▼ logical Operators

Operator	Description	Example
`&&`	**Logical AND:** `true` if both the operands/boolean values are true, else evaluates to `false`	`true && false; // false`
`||`	**Logical OR:** `true` if either of the operands/boolean values is `true`. evaluates to `false` if both are `false`	`true || false; // true`
`!`	**Logical NOT:** `true` if the operand is `false` and vice-versa.	`!true; // false`

▼ Switch Statements

The JavaScript `switch` statement is used in decision making.

The `switch` statement evaluates an expression and executes the corresponding body that matches the expression's result.

```
// program using switch statement
let a = 2;

switch (a) {
    case 1:
        a = 'one';
        break;
    case 2:
        a = 'two';
        break;
    default:
        a = 'not found';
        break;
}
console.log(`The value is ${a}`);
```

Assignments

1. Build a Calculator Application (without the UI) using Arithmetic operators
2. Build an Average Marks Generator. using Arithmetic operators

3. Build a BMI calculator using Arithmetic operators
4. Build a Grading System based on Marks using Switch-case statements.