Javascript Basics

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

Examples	
Correct Variables	1. var abc123 = 11; 2. var _xy= "Hello";
Incorrect Variables	1. var 99 = "Hello"; 2. var *abc1 = 987;

- Global Variable - A variable i.e. declared outside the function or declared with a window object is known as a global variable. A JavaScript global variable is accessible from any function.
- Local Variable - A JavaScript local variable is declared inside a block or function. It is accessible within the function or block only.

Examples	

Global Variable	function sample(){
	var x=10; //local variable
	}

Local Variable	var value=50; //global variable
	function sample(){
	alert(value);
	}

Data Types

JavaScript offers various data types to store various types of values. In JavaScript, there are two different kinds of data types.

1. Primitive Datatypes
2. Non-Primitive Datatypes

Because the JavaScript engine uses variables in a dynamic manner, it is not necessary to declare the type of a variable while using JavaScript. Here, the data type must be specified using var. It can store any kind of value, including strings, integers, and more.

Primitive Datatypes

Datatype	Description
String	represents a sequence of characters e.g. "Hello World"
Number	represents numeric values e.g. 76
Boolean	represents boolean value either "false" or "true"
Undefined	represents undefined value
Null	represents null i.e. no value at all

Non-Primitive Datatypes

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

Operators

Operator symbols in JavaScript are used to carry out operations on operands. There are the following types of operators in JavaScript.

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

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

- Comparison Operator - The following operators are known as JavaScript comparison operators.

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

- Bitwise Operator - The following operators are known as JavaScript bitwise operators.

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
<<	Bitwise Left Shift
>>	Bitwise Right Shift
>>>	Bitwise Right Shift with Zero

- Logical Operator - The following operators are known as JavaScript logical operators.

Operator	Description
&&	Logical AND
||	Logical OR
!	Logical Not

- Assignment Operator - The following operators are known as JavaScript assignment operators.

Operator	Description
=	Assign
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modulus and assign

- Special Operator - The following operators are known as JavaScript special operators.

Operator	Description
(?:)	The conditional Operator returns a value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as a single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks, if the object has the given property
instanceof	checks if the object is an instance of a 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.

If-Else Statement

Whether the condition is true or false, the code is executed using the if-else expression in JavaScript. In JavaScript, there are three types of if statements.

- If Statement
- If Else Statement
- If Else if Statement

Example - Below example depicts how to write if-else statements in Javascript. Also in the below code snippet, the else statement is true and the output is "x is not equal to 20, 72".

```
var x=73;

if(x==20){
    document.write("x is equal to 20");
}
else if(x==72){
    document.write("x is equal to 72");
}
else{
    document.write("x is not equal to 20, 72");
}
```

Switch

One code from many expressions can be executed using the switch statement in JavaScript. It is identical to an "if- else if" statement but is more flexible because it can be used with letters, numbers, and other data. Below is the format to write a switch statements in Javascript.

```
switch(expression){
    case value1:
     code to be executed;
     break;
    case value2:
     code to be executed;
     break;
    ......

    default:
     code to be executed if above values are not matched;
    }
```

Loops

The JavaScript loops are used *to iterate the piece of code* using for, while, do-while, or for-in loops. It makes the code compact. There are 3 types of loops in Javascript.

- For Loop - The JavaScript for loop iterates the elements for a fixed number of times. It should be used if number of iteration is known. The syntax of for loop is given below.

```
for (initialization; condition; increment)
{
    code to be executed
}
```

- While Loop - The JavaScript while loop iterates the elements for an infinite number of times if the condition is always set to true. It can also be used if a number of iterations are known. For loops can be converted into while loops. The syntax of the while loop is given below.

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

- Do while Loop - The JavaScript do-while loop iterates the elements for an infinite number of times like a while loop. But, code is executed at least once whether the condition is true or false. The syntax of the do-while loop is given below.

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

Functions

Operations are carried out via JavaScript functions. To reuse the code, we can call the JavaScript function repeatedly. There are mainly two advantages of JavaScript functions.

1. Code reusability: We can call a function several times so it saves coding.
2. Less coding: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

The syntax of the function declaration in Javascript is - where args1, args2, ... argsN are arguments passed to the function.

```
function FunctionName([args1, args2, ...argsN]){
    //code to be executed
   }
```

There are a few types of functions -

1. Function Arguments - We call any function by passing arguments.
2. Function with return value - We can return value using a function in Javascript.
3. Function object - The function constructor's job in JavaScript is to build a fresh Function object. The code is run on a global scale. Directly calling the function Object() { [native code] }, however, results in a dynamic, insecure function creation.

Javascript Function Methods -

Method	Description
apply()	It is used to call a function containing this value and a single array of arguments.
bind()	It is used to create a new function.
call()	It is used to call a function containing this value and an argument list.
toString()	It returns the result in a form of a string.