NOTE 1: JS is a dynamically typed language.

Variable is loosely typed

String name = “Rekha” 🡺JAVA

var name = “Rekha”; 🡺JS (name is of var datatype)

name = 7; (name is of number datatype)

(ie., during the runtime, based on the input the datatype will be decided.)

NOTE 2: To change the datatype explicitly

NOTE 3: Difference b/w == and ===

NOTE 4: pre increment and post increment

Pre increment will increment and then initialise

Post increment will initialise and then increment

NOTE 5: declaration of variable is of 3 types

var, let, const

JS Execution

1. On browser
2. Inline embedded

Syntax: <script></script>

1. External embedded

<script src = “filepath.js”></script>

1. Off browser

Conditional Statements

In JavaScript we have the following conditional statements:

* Use if to specify a block of code to be executed, if a specified condition is true
* Use else to specify a block of code to be executed, if the same condition is false
* Use else if to specify a new condition to test, if the first condition is false
* Use switch to specify many alternative blocks of code to be executed
* **The if Statement**

Syntax

if (condition) {  
  //*block of code to be executed if the condition is true*  
}

* **The else Statement**

Syntax

if (*condition*) {  
  //*block of code to be executed if the condition is true*} else {  
  //*block of code to be executed if the condition is false*}

🡺 **The else if Statement**

Syntax

if (*condition1*) {  
  //*block of code to be executed if condition1 is true*} else if (*condition2*) {  
  //*block of code to be executed if the condition1 is false and condition2 is true*  
} else {  
  //*block of code to be executed if the condition1 is false and condition2 is false*}

* **The JavaScript Switch Statement**

Syntax

switch(*expression*) {  
  case *x*:  
*// code block*    break;  
  case *y*:  
*// code block*    break;  
  default:  
    // *code block*  
}

Looping statements:

JavaScript supports different kinds of loops:

* for - loops through a block of code a number of times
* for/in - loops through the properties of an object
* for/of - loops through the values of an itterable object
* while - loops through a block of code while a specified condition is true
* do/while - also loops through a block of code while a specified condition is true
* **The For Loop: [returns the index]**

Syntax

for (*statement 1*;*statement 2*;*statement 3*) {  
  // *code block to be executed*  
}

* **The For/In Loop: [returns the index]**

Syntax

for (key in object) {  
  // *code block to be executed*  
}

* **The For/Of Loop: [returns the element]**

Syntax

for (variable of itterable) {  
  // *code block to be executed*  
}

* **The While Loop**

Syntax

while (*condition*) {  
*// code block to be executed*  
}

* **The Do/While Loop**

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

do {  
*// code block to be executed*}  
while (*condition*);

Programme execution in memory

Whenever we execute the JavaScript code, a global execution context will be created.

In global execution context, a global object will be created named “window”.

Along with window object, global variable will be created. This global variable is called as ”this”

At global level, window===this (window is strictly equal to this ).

Then 2 phases will be created

🡪Creation phase

🡪Execution phase

Creation phase:

In creation phase memory will be created to the entire code, and the name of the memory created, will be the variable name or function name

By default dummy value ie., undefined will be assigned for all the variables.

And for function, the implementation of the function will be stored inside a memory

Then the entire code will reach the execution phase.

Execution phase:

In execution phase, undefined will be overrided with the actual values for the variables and if the function is invoked in the execution phase, the function will be executed.

Function execution in a memory

Whenever the function is called, a function execution context will be created on top of global execution context in call stack.

After the function is executed, the function execution context will get popped off from the call stack.

**Variable hoisting:**

Whenever we execute a JavaScript code,

In creation phase: a separate memory will be created for all the variables and by default, undefined will be assigned to all the variables.

In execution phase: the undefined will be overrided by the actual value of the variable.

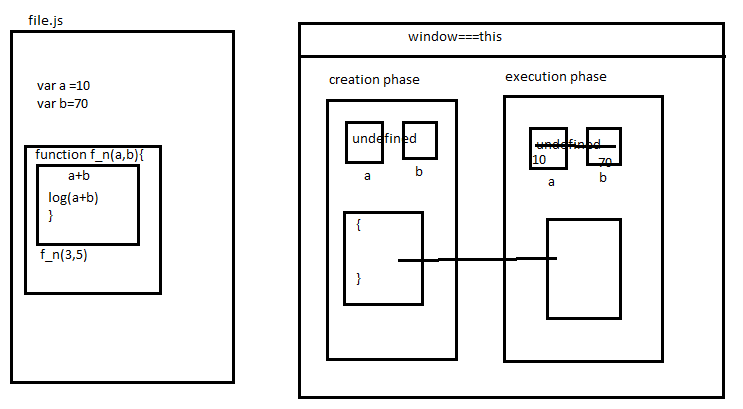
**Function hoisting:**

Whenever we execute a JavaScript code, in case of function

In creation phase: a separate memory will be created for all the functions and the function implementation will be stored inside the memory.

In the execution phase: If the function is invoked in the execution phase, the function will be executed.

Program execution pictorial representation



Different blocks which will get created during the programme execution

