JAVA SCRIPT

EP-2

Q.WHAT HAPPENS WHEN YOU RUN JAVASCRIPT CODE?

ANS.

Code:

var n=2;

function square(num){

var ans = num\*num;

return ans;

}

var square2=square(n);

var square 4=square(4);

Console:

Behind the scenes working:

1ST PHASE WORKING:

(MEMORY EXECUTION PHASE)

|  |  |
| --- | --- |
| Memory | Code |
| n:undefined  square:{…}  square 2:  undefined  square 4:  undefined |  |

* Memory is allocated to variables and functions.
* For variable name(which is key) it assigns a value of **undefined**
* For the function name(which is key) it assigns the entire function code as value.

2ND PHASE WORKING:

(CODE EXECUTION PHASE)

|  |  |
| --- | --- |
| Memory | Code |
| n:2  square(num){ | |  |  | | --- | --- | | MEMORY | CODE | | num:  undefined  ans:  undefined |  | |
| Square 2:~~undefined4~~  Square 4:  ~~undefined~~ 16 | |  |  | | --- | --- | | MEMORY | CODE | | num:2  ans:4 | num\*num |  |  |  | | --- | --- | | MEMORY | CODE | | Num:2  Ans:4 | Return ans |  |  |  | | --- | --- | | MEMORY | CODE | | Num: ~~undefined4~~  Ans: ~~undefined~~ 16 | Return ans | |

JS MANAGES EXCECUTION CONTEXT USING CALL STACK :

* Whenever a function is invoked a new execution context is created and is stored in the call stack.
* When stored it is pushed into the call stack.
* When deleted it is popped and the control returns to global execution context.
* Call stack maintains the order of execution contexts.
* Call stack is also called as:

1. Execution context stack
2. Program stack
3. Control stack
4. Runtime stack
5. Machine stack