Programming

Namaste JavaScript Ep. 2

How JavaScript Code is executed?

JavaScript

Timestamp	27 December 2024
Resource	https://www.youtube.com/watch?v=iLWTnMzWtj4&list=PLlasXeu85E9cQ32gLCvAvr9vNaUccPVNP&inc
Difficulty	Unraked

Lesson Main Points

When running a code, a ***global*** execution context is created.

Memory Creation Phase

Functions are the heart of JavaScript

After running the function the whole Execution context created for that function is deleted.

Call Stack maintains the order of execution of execution contexts

Short Notes

JavaScript

```
var n = 2;
function square (num) {
  var ans = num * num;
  return ans;
}
var square2 = square(n);
var square4 = square(4);
```

- Memory Creation Phase | Phase 1
  - create spaces on memory upon encountering a variable
  - e.g n: undefined; special value/ placeholder
    - square: {stores the whole code since it is a function}
- Code Execution Phase | Phase 2
  - goes through line by line and execute
  - exe line 1 → places 2 inside n (replacing the undefined value)
  - invoke function → a brand new exe context is created with two components
- Return
  - Return the control of the program where the function was revoked. (for this case Line 6)
  - Replace square2 → undefined with the return ans 4.

- Call Stack

  Control the execution contexts creation, deletion

  It is a stack. Bottom >> global execution context

  When a function is invoke/ created >> Execution context 1 created in the stack

  After execution Execution context 1 popped out of the stack and the control goes back to the global execution context

  Whenever the exeContext created goes into the stack, when done pops out of the stack and only the global exeContext is remained.

  ### Other Names of Call Stack

  1. Execution Context Stack
  2. Program Stack
  3. Control Stack
  4. Runtime Stack
  5. Machine Stack

Takeaway

Executin Context

Call Stack