JavaScript Execution Context

- How JavaScript code is going to be executed

a) Global Execution Context: -always created first - referred or allocated through this keyword.

1. In node environment - empty object, {}, console.log(this) – {}

2. In the browser environment - window or global object (depends on the context), console.log(this) – Window {properties}

b) Function Execution Contexts: Created when a function is invoked and destroyed after it returns/exits

Above are two major ECs

c) Eval Execution Context - It is a property of global execution

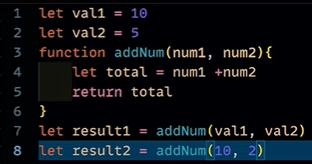
JS is single-threaded. Everything in JS is process

Two Phases of JS execution:

1. Memory Creation Phase (Creation Phase) – memory is allocated to variables but not executed

2. Execution Phase

Steps:

1. Global Execution (or Global Environment) – this
2. Memory Phase – memory allocation

val1 -> undefined

val2 -> undefined

addNum -> definition

result1 -> undefined

result2 -> undefined

1. Execution Phase –

val1 <- 10 …(1)

val2 <- 5 …(2)

Note: In execution phase, in line 3, we have nothing to execute currently, we just hold the function definition

Result1 Result queried for addNum, now go to addNum …(3)

addNum creates its own execution context

addNum <- new Variable Environment

or Memory Phase : Execution Phase: …(4)

Global environment val1 -> undefined num1 -> 10

+ val2 -> undefined num2 -> 5

Execution thread total -> undefined total -> 15

Deleted once finished

Note: In addNum’s Execution Phase, total value will be returned to global execution context.

Result1 = 15

Result2

Again Result2 queried addNum, again New variable Environment

Result2 <- 12

**Call Stacks:**