Execution Context


```
function getMessage()
{
    return 'Hi Simran';
}
function SayHello()
{
    return getMessage()
}
sayHello();
```


Javascript Engine (JE) sees these brackets and says 'OH I need to execute this function, let me create an execution context for it.

Okay.... What is execution context ???

② JE then sees sayHello is calling getMessage() so creates another EC

① This is basically the execution context

BUT there is a base execution context (global context)

So basically the JE will always first create the global execution context.

this is always there as long as code is executing

That means every line of code is part of an execution context (eg. global), or a function)

So what do I do with this global Execution context?

↳ It gives us 2 things

- Global execution context

↙ Global Object (which is the window object)

↘ this keyword in JS (the one that everyone loves :))

GO AND CREATE AN EMPTY JS FILE

Go to console and type

> this

⟶ window object

So this prints the global object. In a browser global object is nothing but window object.

SO CHECK

```
this === window // true
```

Are they always going to be same? :/

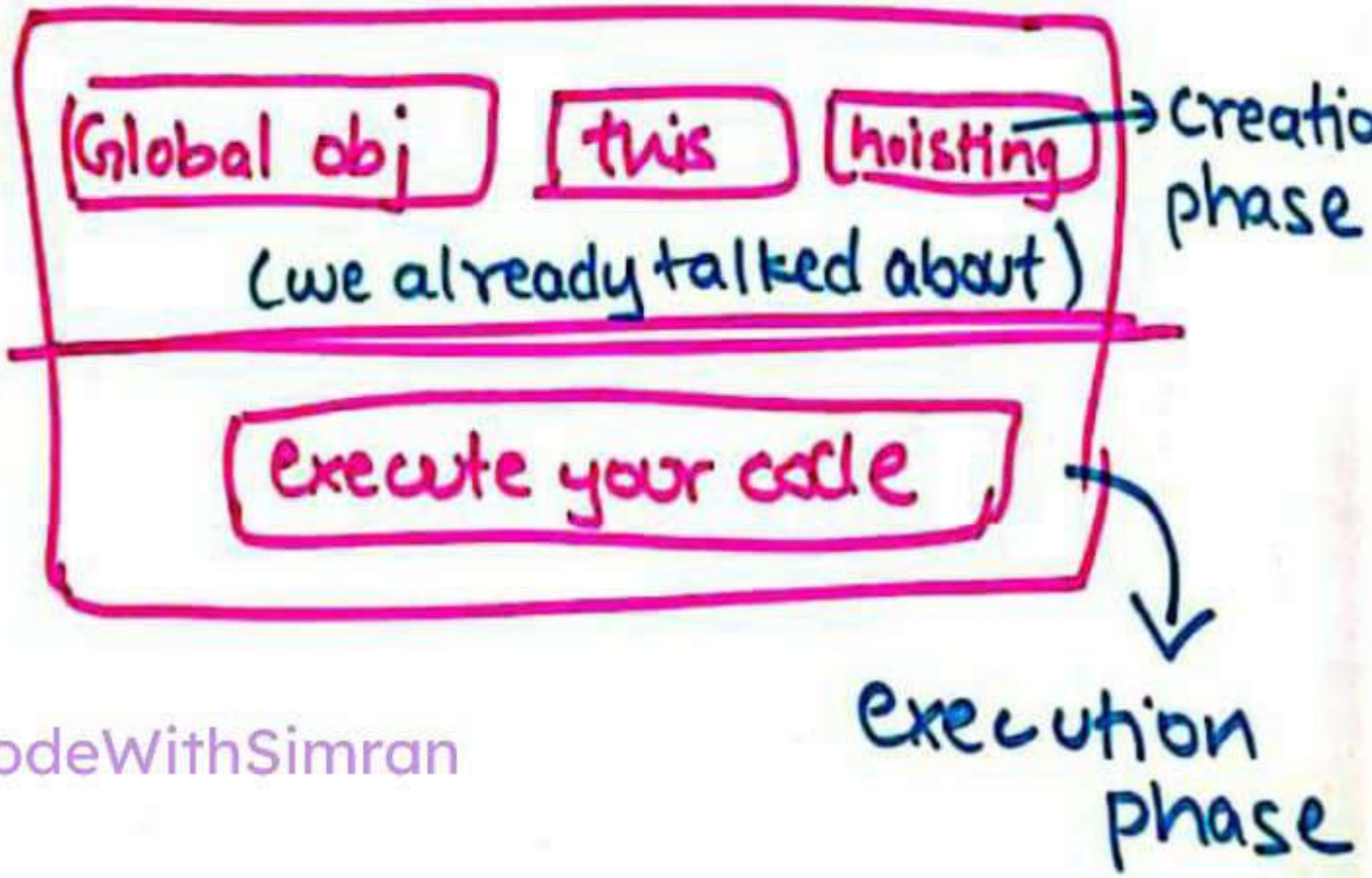
↳ NO !!! We have a whole new section to understand this keyword. Hold on :D

GO TO CONSOLE AND DECLARE A VARIABLE


```
> var name = "Simran"
> window → try to see window
{
  name: 'Simran'
}
```

[window object will have name variable]

There are 2 phases of an execution context



Creation phase → global Obj(window)
(happens first) this
hoisting

execution phase → run code
(happens next)