What is hoisting?

@codeWithSimran

Ex 1

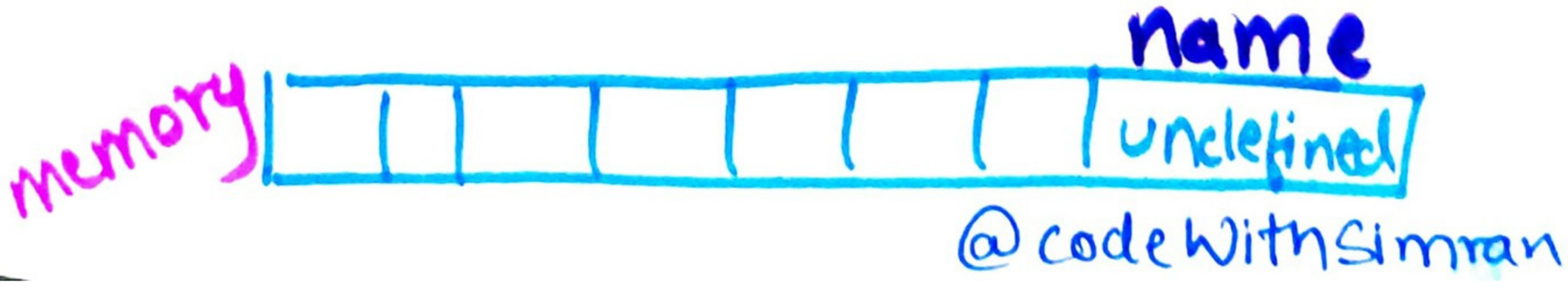
```
console.log(name)
var name = 'Simran'
```

> undefined

Why undefined? Should'nt it say name 'is ~~no~~ dosen't exist (Reference error)

It's because of hoisting!!!

During the creation phase JE sees this variable (name) and allocates memory for it and keeps undefined as its value.



@codeWithSimran

So since hoisting happens in creation phase there is already a variable named name in the memory with value undefind. @codeWithSimran

So during execution phase when it runs the file from top to bottom thus is what it sees [not physically we'll see that in a while]

```
var name = undefinded
sonsole.log(name)
name = 'Simran')
```

this is hoisting

note this is an assignment and does not have var

@codeWithSimran

codeWithSimran codeWithSimran_

NOTE: Variables are partially hoisted

@codeWithSimran

Meaning: Jhey are allocated memory but not assigned actual value what we give. They're simply assigned undefined.

Example 2

```
var name = 'Simran'
var name = 'Simran'
```

What about this? How many times will name be hoisted?

ONCE!! Why! Since the value dosent matter, JE will say I already have hoisted name.

Example 3


```
sayHello();
function sayHello() {
    console.log('say Hello')
}
```

> Hello

Just like variables, functions are also hoisted.

BUT!! Functions are completely hoisted.
Meaning: The entire function with defination is hoisted.
So it wont give us undefined.

Example 4

@codeWithSimran

Just a clarification

JE doesn't physically move variables and functions to top of file. It just allocated memory for them & before ✦ running the code.

```
sayHello();  → function expression

var sayHello = function(){
    console.log('Hello')
}
> undefined
```

It was treated like a var and given undefined as it doesn't care about the value

Examples

@codeWithSimran

```
sayHello();

~~sayHel~~

function sayHello(){
    console.log("Hello");
}
function sayHello(){
    console.log("Bollo");
}
```

> Bollo

Since functions are completely hoisted, when same function comes 2nd time, JE will say I've already allocated memory for sayHello, let me replace its content from console.log("Hello") to console.log("Bollo")

Example 6

@codeWithSimran

TRICKY ONE

Just remember when a function is called a new execution context is created for that function.

```
var name = 'Simran'
var changeName = function(){
    console.log("Name", name);
    Var name = " John";
    console.log ("Changed name",
                    name);
};
changeName();
```

@codeWithSimran

what you probably expect

> Name Simran

> Changed Name John

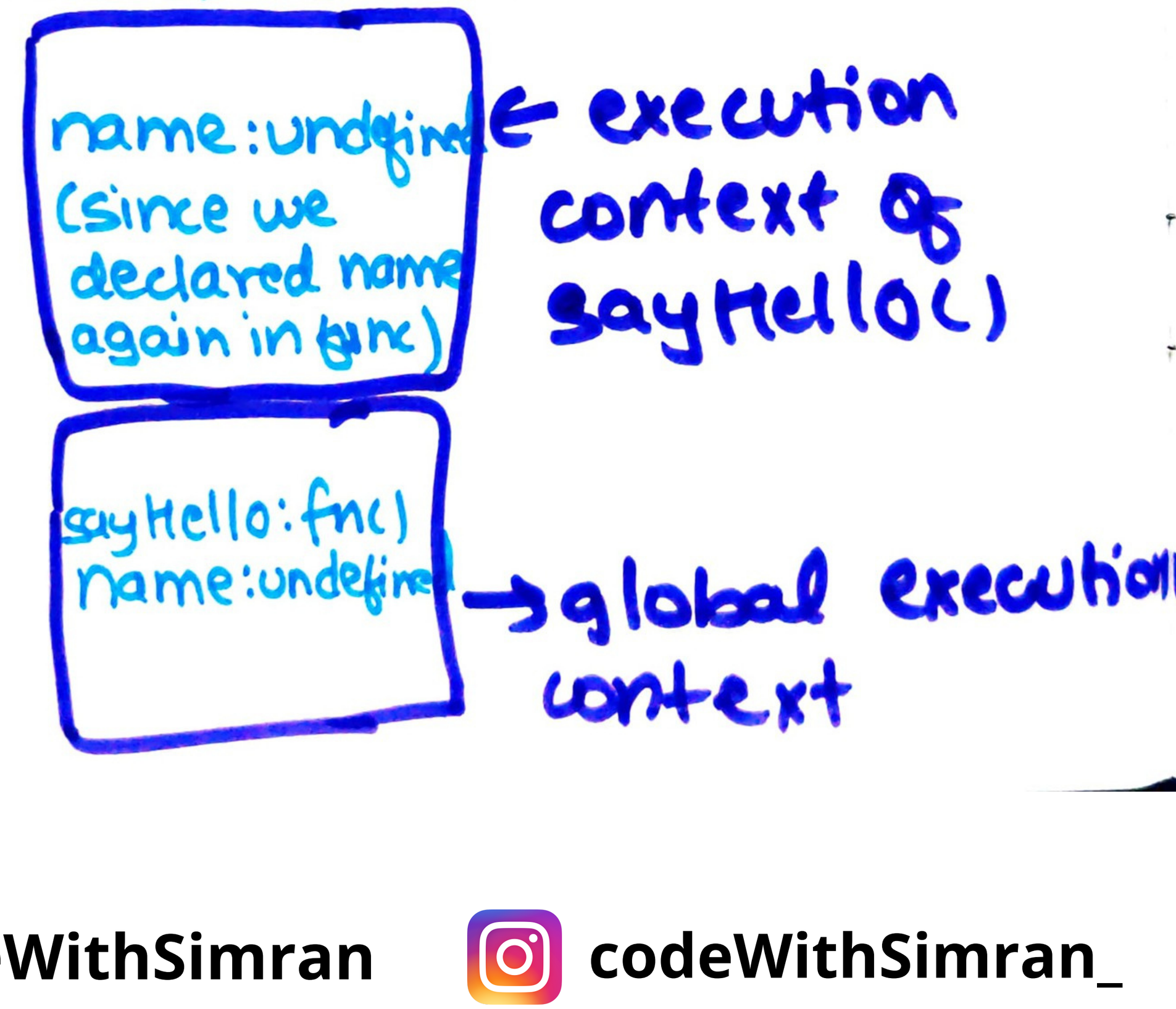
Wrong ⊗

> Name undefined
> changed Name John ✓

@codewithSimran

What ?? Why?

When you called a new execution context was created on top of global execution context.
This execution context is created for the function when it is called



Go and look at the problem again, you'll get it. :)

BUT STILL, JS is tricky and weird :/

Oh! guess why they introduced let and const :)

They are [redacted] hoisted !!!

@codewithsimran

Even though they're hoisted they're not assigned anything (not even undefined) and we get reference error if we try to access them before using them