Hoisting

Javascript hoisting is an interesting concept that allows you to access variables and functions even before their actual declaration

Example

So let's understand it by example.

In the code snippet.
It will simply log the value of x and getMessage function.

What will happen if we called variables and functions before their declaration?

```
var x = "pizza";
function getMessage(){
console.log( I love`${x}`)

console.log(x);
console.log(getName())
```

```
x => pizza
getMessage => I love pizza
```

Continue...

What will happen when the compiler executes this statement? It does not know about any function or any variable in the code. So how can it understand what to do with these two statements? The answer lies in JavaScript's hoisting behavior where all variables are available even before they are declared.

```
console.log(x);
console.log(getName())

var x = "pizza";
function getMessage(){
console.log( I love`${x}`)
```

```
x=>undefined
getMessage=> I love undefined
```

How it works

So when we run a javascript program an execution context is created and it is created in two phases.

- *Memory Allocation.*
- *Code Execution.*

We'll concentrate on the first phase because it's where the entire concept lies.

Memory	Code

Global Execution Context

Back to Agenda Page

Memory Allocation

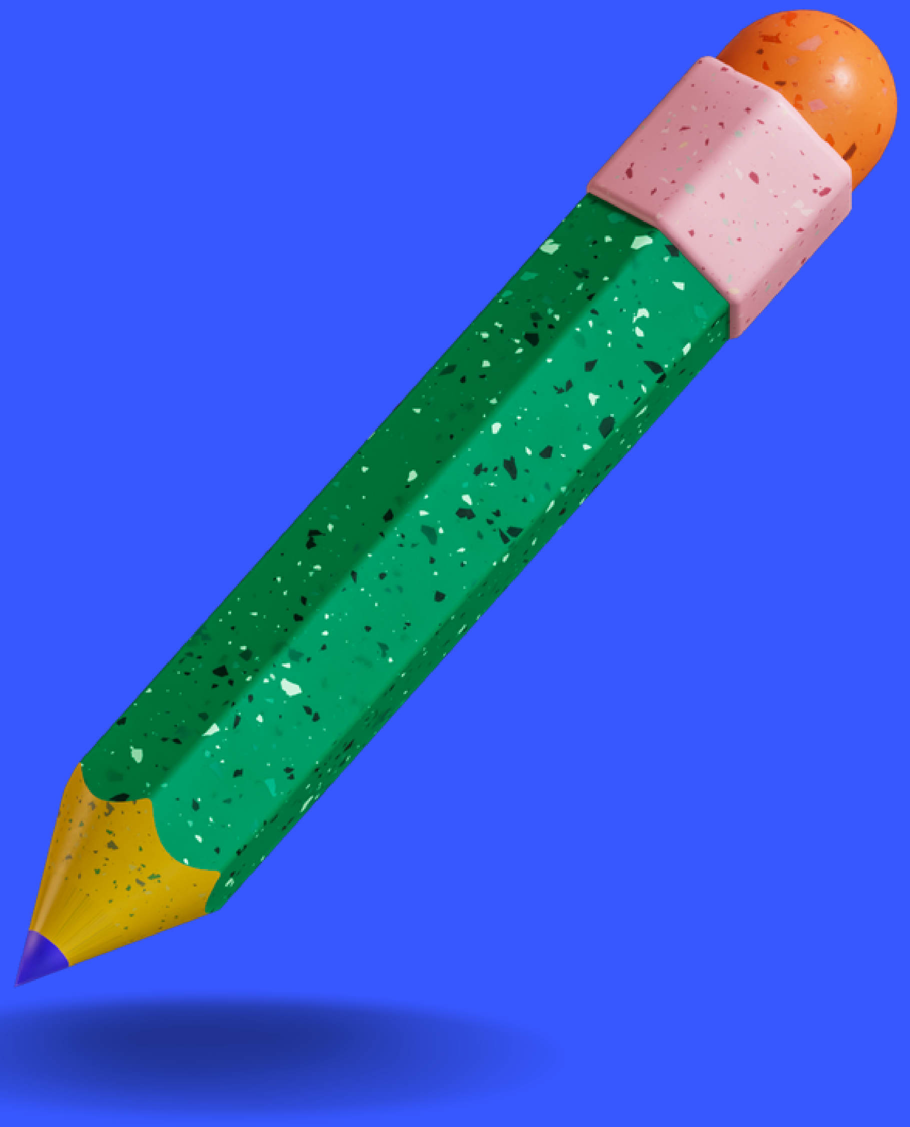
In Phase one JavaScript just skims through the program and it will allocate memory to every variable and function present.

In this way, we will be able to access this function and variables before we declare them.

Memory	Code
x:undefined getMessage:undefined	

Global Execution Context

Thank you for reading



Helpful, share with your friends.
Save it for later.

For questions, DM me: