* Hoisting in JavaScript:

Hoisting is a phenomena in Javascript by which we can access this variables and functions. even before we have initialised it or even before we have put some value to it. We can access it anywhere without any error in the program.

Examples:-

(1)

```
Var x = 7;
function getName(){
console.log("Namaste JS");
}
getName();
console.log(x);
```

O/p:-

Namaste JavaScript
7

(2) In this example, here we are going to access 'x' & getName() even before we have initialised it.

```
getName();
console.log(x);
Var x = 7;
function log("Nam
function getName(){
console.log("Namaste JS");
}
```

O/p:- Namaste Javascript
Undefined.

(3)

```
getName();
console.log(x);
function getName(){
console.log("Namaste JS");
}
```

→ O/P:-

Error: x is not defined.

• It is giving us Error because when it goes through the code in 1st phase the JS find getName() function only. and in our code there is no 'x' is there. so, it gives us Error.

(4)

```
function getName(){
Console.log("Namaste JS");
}
Console.log(getName);
```

→ Here we are not invoking the fn but just tries to log this getName method.

→ O/P:-

```
f getName(){
console.log("Namaste JS");
}
```

(5) Here we are trying to log out the getName method even before we have initialised it.

```
Console.log(getName);
var x = 7;
function getName(){
console.log("Namaste JS");
}
```

O/P:-

```
f getName(){
console.log("Namaste JS");
}
```

→ To understand, why JS is behaving like this. Just remember in previous class we have studied. that whenever a JS code run a Execution Context is created and it is created in 2 phase.

(1) Memory Creation Phase (2) Code Execution Phase.

→ So the ans lies in Memory Creation phase only.

→ We know that
Even before the code start executing, memory is allocated to each & every variable & function present inside the program

→ If we write

```
var getName = () => {
  console.log("Namaste JS");
}
```

```
var getName = function () {
  console.log("Namaste JS");
}
```

→ These two are different ways of writing a function.
But it will be stored in memory as a variable.
With a special value "undefined".

```
function getName() {
  console.log("NamasteJS");
}
```

→ If we write function like this then only
It will store the whole function in 1st phase.