Episode 3 : Hoisting in JavaScript (variables & functions)

- Let's observe the below code and it's explaination:

```
getName(); // Namaste Javascript
console.log(x); // undefined
var x = 7;
function getName() {
 console.log("Namaste Javascript");
}
```

- It should have been an outright error in many other languages, as it is not possible to even access something which is not even created (defined) yet But in JS, We know that in memory creation phase it assigns undefined and puts the content of function to function's memory. And in execution, it then executes whatever is asked. Here, as execution goes line by line and not after compiling, it could only print undefined and nothing else. This phenomenon, is not an error. However, if we remove var x = 7; then it gives error. Uncaught ReferenceError: x is not defined
- **Hoisting** is a concept which enables us to extract values of variables and functions even before initialising/assigning value without getting error and this is happening due to the 1st phase (memory creation phase) of the Execution Context.
- So in previous lecture, we learnt that execution context gets created in two phase, so even before code execution, memory is created so in case of variable, it will be initialized as undefined while in case of function the whole function code is placed in the memory. Example:

```
getName(); // Namaste JavaScript
console.log(x); // undefined
console.log(getName); // f getName(){ console.log("Namaste JavaScript); }
function getName(){
    console.log("Namaste JavaScript");
}
```

- Now let's observe a different example and try to understand the output.

```
getName(); // Uncaught TypeError: getName is not a function
console.log(getName);
var getName = function () {
    console.log("Namaste JavaScript");
}
// The code won't execute as the first line itself throws an TypeError.
```

Watch Live On Youtube below: