# Session 1 (Javascript + React)

## Javascript is a dynamically typed language

Dynamically-typed languages are those where
type of the variables is assigned at runtime based on the variable's value at that time.

For Example :

var x = 7;  // assigning an integer value

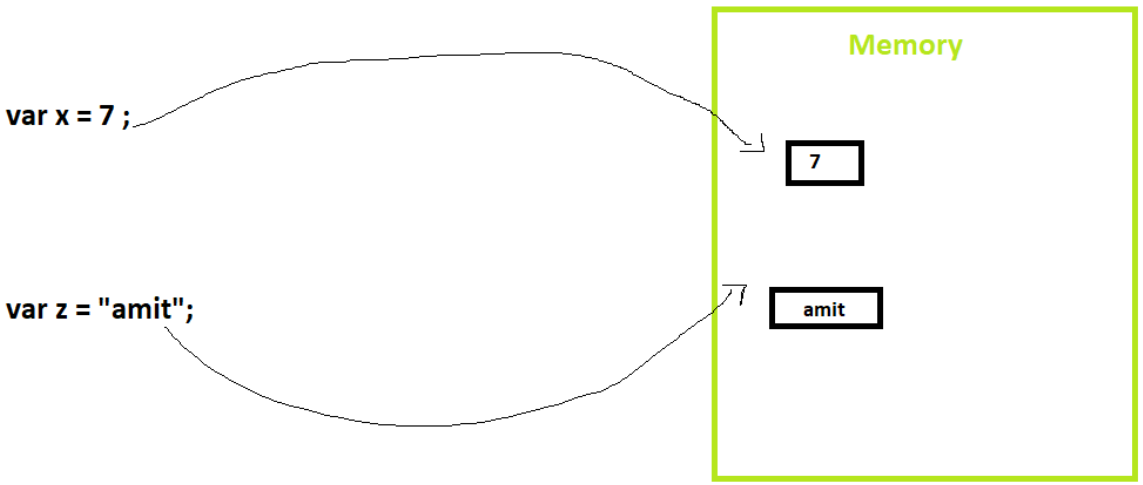
x ="hello";  // assigning a string

x = [2,3,4,];  // assigning an array

In the above example I have assigned different kind of values to a variable “x” and it will not give any error.

### What are variables ?

Variables are the names(references) you give to computer memory locations which are used to store values in a computer program.




## Hoisting in JavaScript (variables & functions)

Let's observe the below code and it's explanation:

```
getName(); // Amit
console.log(x); // undefined var x = 7;

var x = 7;

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

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 javascript, we know that in the 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 an error. Uncaught ReferenceError: x is not defined.

***Hoisting*** *is a concept which enables us to extract values of variables and functions even before initializing/assigning value without getting error and this is happening due to the 1st phase (memory creation 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.

Example1:

```
getName(); // Amit Pachisia

console.log(x); // Uncaught Reference: x is not defined.

console.log(getName); // f getName() { console.log("Amit Pachisia); }

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

# Undefined vs not defined in JS

- Undefined is when memory is allocated for the variable, but no value is assigned yet.
- If an object/variable is not even declared/found in the memory allocation phase, and tries to access it then it is Not defined.
- Not Defined !== Undefined.

When a variable is declared but not assigned value, its current value is undefined. But when the variable itself is not declared but called in code, then it is not defined.

```
console.log(x); // undefined
var x = 25;
console.log(x); // 25
console.log(a); // Uncaught ReferenceError: a is not defined
```

# Let & const in JS, Temporal Dead Zone

> Let and const declarations are hoisted. But it's different from var.

Example1:

```
console.log(a); // ReferenceError: Cannot access 'a' before initialization
let a = 10;
```

Example2:

```
console.log(b); // undefined
var b = 10;
```

It looks like let isn't hoisted, but it is, let's understand

- Both a and b are actually initialized as undefined in the hoisting stage. But var b is inside the storage space of GLOBAL, and a is in a separate memory object called script, where it can be accessed only after assigning some value to it first ie. one can access 'a' only if it is assigned. Thus, it throws an error.

- Temporal Dead Zone : Time since when the let variable was hoisted until it is initialized to some value.
  So any line till before "let a = 10" is the TDZ for a.