episode 08

'let' and 'const' in javascript / Temporal dead zone

- ['let' and 'const' declarations are hoisted.]

```
console.log(b);
var b=100;
```

⇒ undefined [No error]

```
console.log(a);
let a=10;
```

⇒ cannot access 'a' before initialization

Refrence Error

```
let a=10;
console.log(a);
```

⇒ 10 (No error)

```
console.log(a);
```

} temporal dead zone of 'a' [will error]

```
let a=10;
```

```
console.log(x);
let a=10;
```

⇒ x is not defined

Reference Error

In case of let and const, they are also allocated memory but they are stored in different memory space than global and you cannot access these memory space, i.e. let and const declarations before you have put some value in them and this is what is hoisting is in let and const

[temporal dead zone is the time since when let variable was hoisted and till it is initialized some value. Time between this is called temporal dead zone.]

we get

```
let a=10;
console.log(a);
var b=100;
```

[whenever you try to access a variable in temporal dead zone, it will give you a reference error
[cannot access 'a' before initialization]]

this.a as undefined because they are not stored on window object

in console ⇓ | result ⇓
this.b ⇒ 100
this.a ⇒ undefined

duplicate redeclaration

```
let a=10;
let a=100;
```

identifier 'a' has already been declared

syntax Error

```
let a=10;
var a = 100;
```

identifier 'a' has already been declared

syntax Error

in case of var we get no error

```
var b= 100;
var b=1000;
console.log("Hi");
```

⇒ Hi

[possible in 'var' but not in 'let']

["const" is more strict than "let"]

valid

```
let a;
a=10;
console.log(a);
```

⇓

10

invalid

```
const b;
b=1000;
console.log(b);
```

⇓

Missing initializer in const declaration

Syntax error

```
const b=1000;
```

⇓

correct way to do this

[always use const and let] [let and const behave differently when they are hoisted]

[const should be initialised and declared on one line]

```
- const b=1000;
  b=10;
```

⇒ Assignment to constant variable

TypeError

- [Best way to avoid temporal dead zone is to always put your declarations and initializations on the top, so that temporal dead zone will be 'zero'. i.e. we are shrinking it to zero.]