var variable and function declarations are hoisted. But let, const, and function expressions are not hoisted.

var variables are Function/Global scoped while let and const variables are code block ({..}) scoped.

Closure keeps its surrounding environment states that already popped off from the call stack.

Execution Context (EC) phases:

In creation phase:

1. Any variables declared with var? Put the variables in Lexical Environment (**LE**) with value undefined. Skip those variables inside function declaration, but not in if/for/while.
2. Any variables declared with let or const? Put them in TDZ. Skip those variables inside of block code.
3. Any function declarations? Put in the LE as whole.

In execution phase:

Execute code line by line. Push/pop the execution contexts to/from the call stack based on the given code and scopes.

Note: in the following examples, I only showed the LE states for each phase. I didn't show the results of pushing/popping ECs in the call stack sequences.

```
console.log(str);
var str = "hello";
console.log(str);
```

Global EC in creation LE:{outer: null, str: undefined}
Global EC in execution LE:{outer: null, str: “Hello”}

```
let str = "hello";
console.log(str);
```

Global EC in creation LE:{outer: null} , TDZ{str}
Global EC in execution LE:{outer: null, str: “Hello”}

```
console.log(str);
if (true) {
    var str = "hello";
}
console.log(str);
```

Global EC in creation LE:{outer: null, str: undefined}
Global EC in execution LE:{outer: null, str: “hello”}

```
if (true) {
    let str = "hello";
    console.log(str);
}
console.log(str);
```

Global EC in creation LE:{outer: null}
Global EC in execution LE:{outer: null}
if EC in creation LE:{outer: Global }, TDZ:{str}
if EC in execution LE:{outer: Global, str: ”Hello” }

```
function foo(arg) {
    if (arg) {
        var str = "hello";
    }
    console.log(str);
}
foo(1);
console.log(str);
```

Global EC in creation LE:{outer: null, foo:fn}
Global EC in execution LE:{outer: null, foo:fn}
foo EC in creation LE:{outer: Global, arguments:{0:1,length:1}, arg:1, str: undefined}
foo EC in execution LE:{outer: Global, arguments:{0:1,length:1}, arg:1, str: “hello”}

```
foo("Hi", 23);
function foo(arg) {
    if (arg) {
        var str = "hello";
    }

    console.log(str);
}
```

Global EC in creation LE:{outer: null, foo:fn}
Global EC in execution LE:{outer: null, foo:fn}
foo EC in creation LE:{outer: Global, arguments:{0:”hi”,1:23,length:2}, arg:”Hi”, str: undefined}
foo EC in execution LE:{outer: Global, arguments:{0:”hi”,1:23,length:2}, arg:”Hi”, str: “hello”}

```
var foo = function () {
    console.log("Hello");
}
foo();
```

Global EC in creation LE:{outer: null, foo: undefined}
Global EC in execution LE:{outer: null, foo: fn}

function EC in creation LE:{outer: Global, arguments:{length:0}}
function EC in execution LE:{outer: Global, arguments:{length:0}}

```
let foo = function () {
    console.log("Hello");
}
foo();
```

Global EC in creation LE:{outer: null}, TDZ{foo}
Global EC in execution LE:{outer: null, foo: fn}

function EC in creation LE:{outer: Global, arguments:{length:0}}
function EC in execution LE:{outer: Global, arguments:{length:0}}

```
function b() {
    function a() {
        console.log(x); //10
    }
    const x = 10;
    console.log(x); //10
    a();
}
const x = 20;
b();
```

Global EC in creation LE:{outer: null, b: fn}, TDZ{x}
Global EC in execution LE:{outer: null, b: fn, x: 20}
b EC in creation LE:{outer: Global, arguments: {length:0}, a: fn}, TDZ{x}
b EC in execution LE:{outer: Global, arguments: {length:0}, a: fn, x:10}
a EC in creation LE:{outer: b, arguments: {length:0}}
a EC in execution LE:{outer: b, arguments: {length:0}}

```
function a() {
    console.log(x); //20
}
function b() {
    const x = 10;
    console.log(x); //10
    a();
}
const x = 20;
b();
```

Global EC in creation LE:{outer: null, a: fn, b: fn}, TDZ{x}
Global EC in execution LE:{outer: null, a: fn, b: fn, x: 20}
b EC in creation LE:{outer: Global, arguments: {length:0}}, TDZ{x}
b EC in execution LE:{outer: Global, arguments: {length:0}, x:10}
a EC in creation LE:{outer: Global, arguments: {length:0}}
a EC in execution LE:{outer: Global, arguments: {length:0}}
