Front-end Advanced

Scope, Closure and Hoisting

Contents

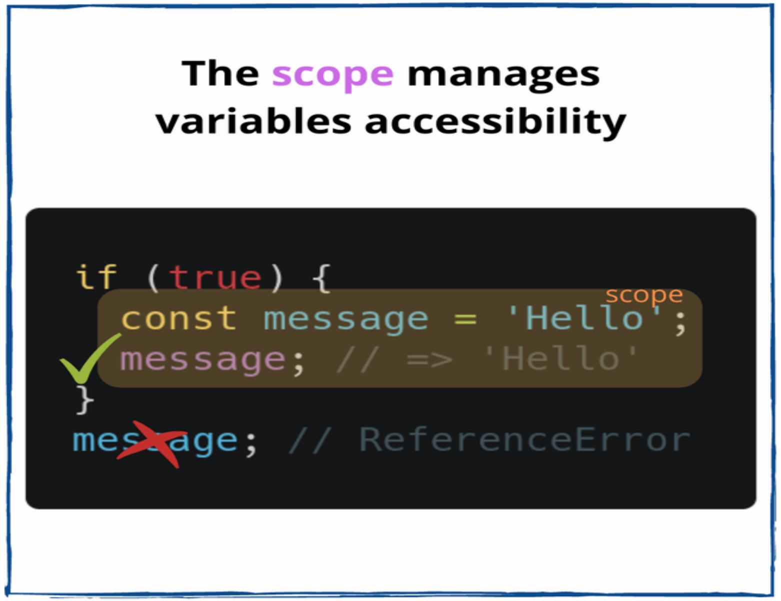
Section 1

Scope

Scope

➢ What is Scope?

- Scope determines the accessibility of variables, objects, and functions from different parts of the code.



```
if (true) {
  const message = 'Hello';
  message; // => 'Hello'
}
message; // ReferenceError
```

Scope

➢ What is Scope?

- Scope is used for the purpose of assigning a value to a variable (assignment) or it may be for the purpose of taking its value (look up)
- Example of asigment and lookup
  - Assignment: var a = 2; (left side of '=' operator)
  - Look-up:var a = 10; var b = a; (right side of '=' operator)

Scope

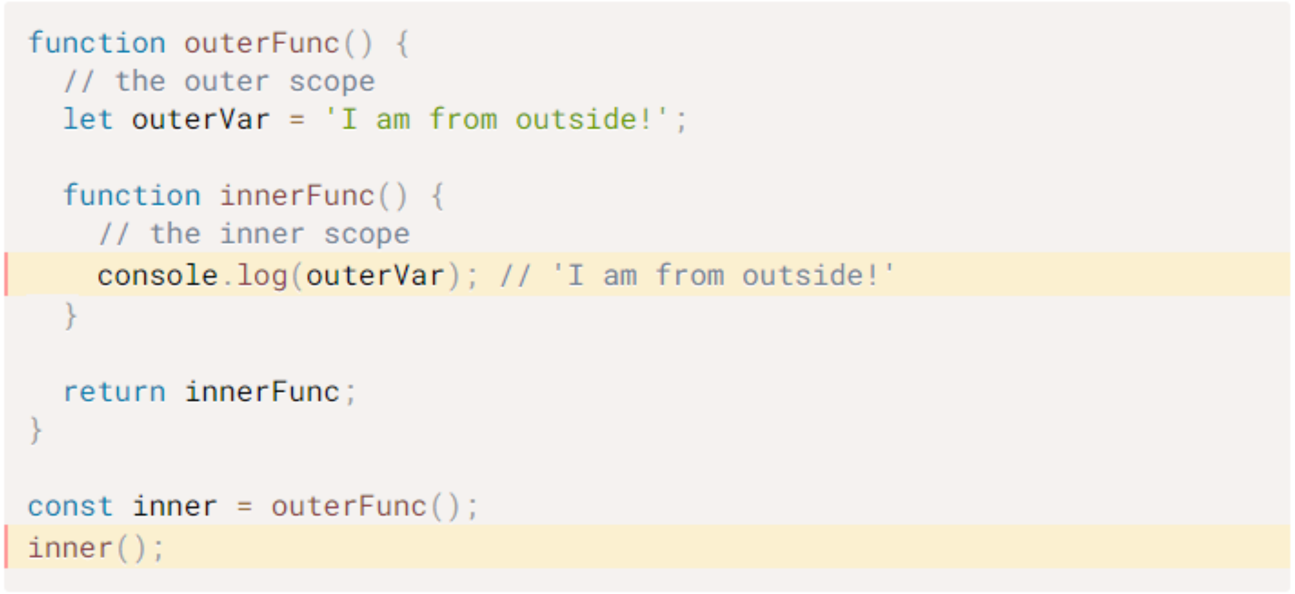
- **Types of scope:** From ES6, there are 3 types of scope
  - Function scope
  - Global scope (when you declare variable at top level)
  - Block scope (ES6)

Section 2

Lexical Scope

Lexical Scope

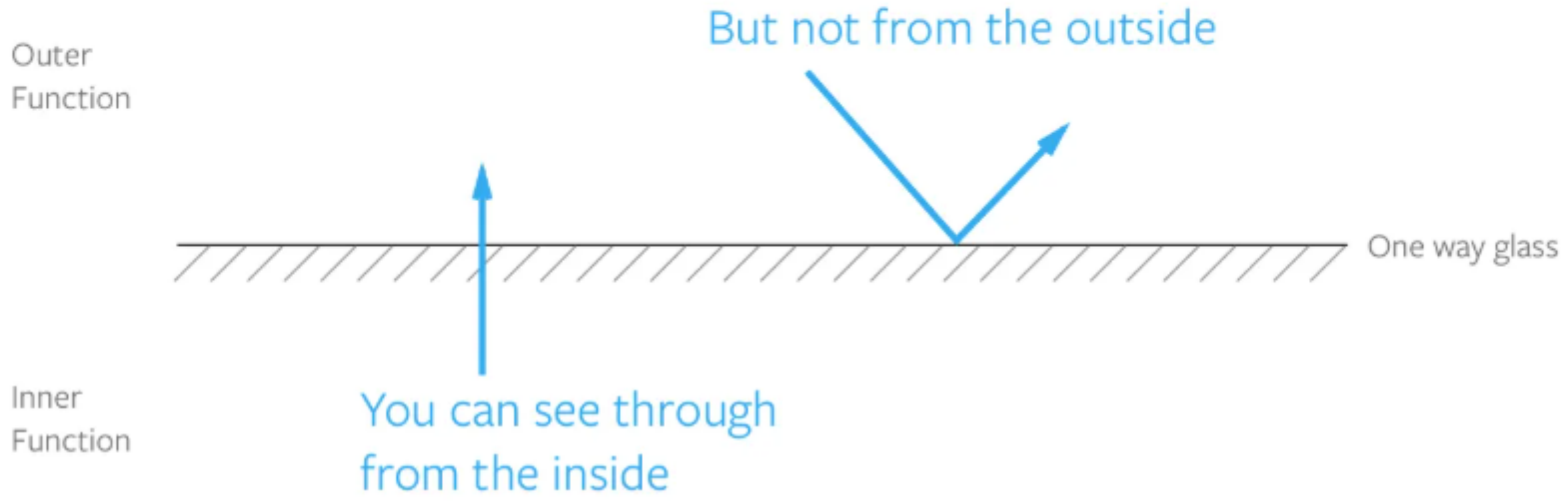
- There is a **lexical scope** when having nested functions.
- Example:



```
function outerFunc() {
  // the outer scope
  let outerVar = 'I am from outside!';

  function innerFunc() {
    // the inner scope
    console.log(outerVar); // 'I am from outside!'
  }

  return innerFunc;
}

const inner = outerFunc();
inner();
```

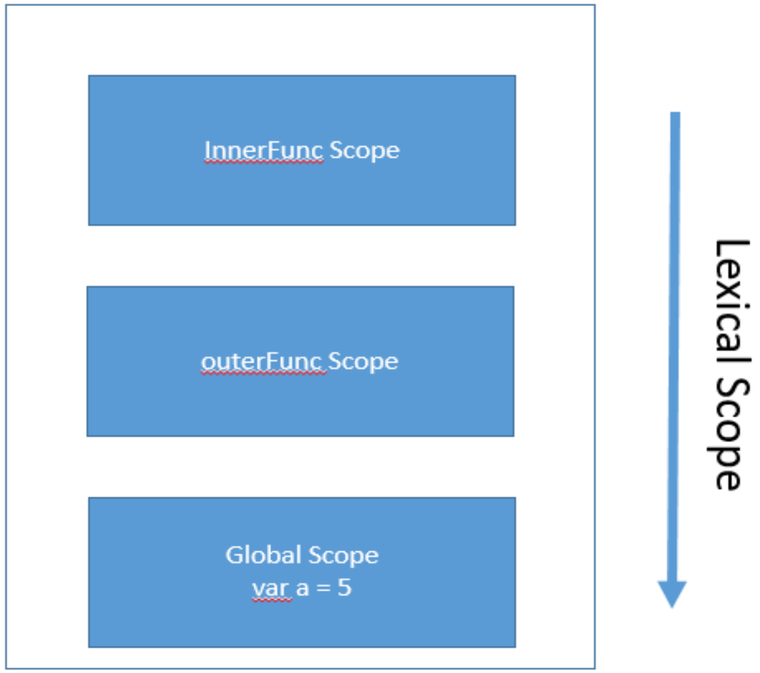
Lexical Scope

➢ **The lexical scope** is “one way glass”: ***Inner function*** can acces to variables of outer function but ***outer function*** cannot.



Lexical Scope

➢ **The lexical scope** is defined in compiler phase.



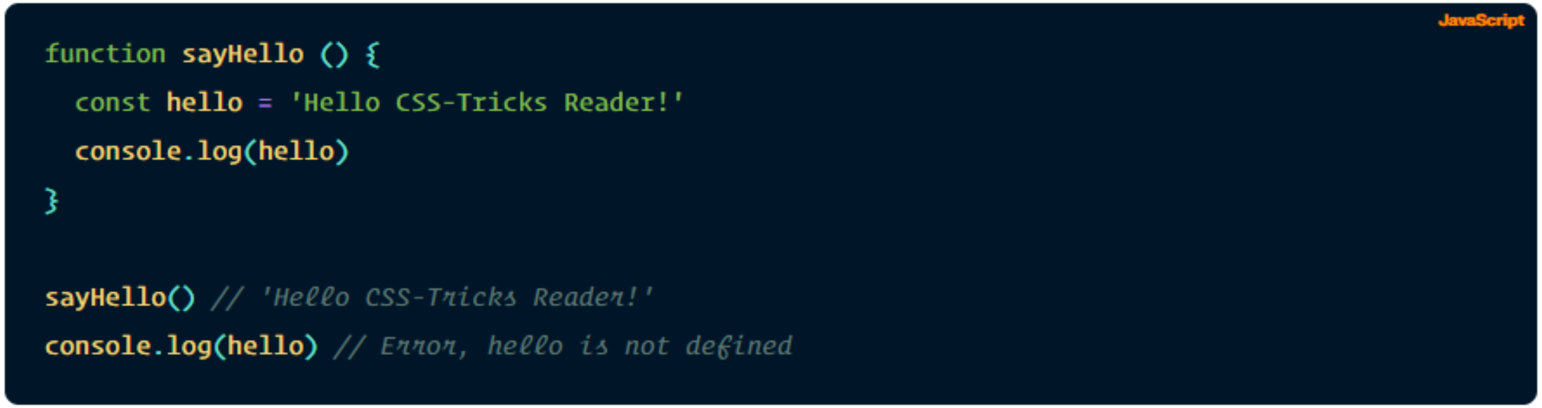
Section 3

Function scope and Block scope

Function scope and Block scope

➢ **What is function scope?**

- The variable which is declared inside the function is called function scope.
- The function scope variable cannot be accessed or modified out side the function.

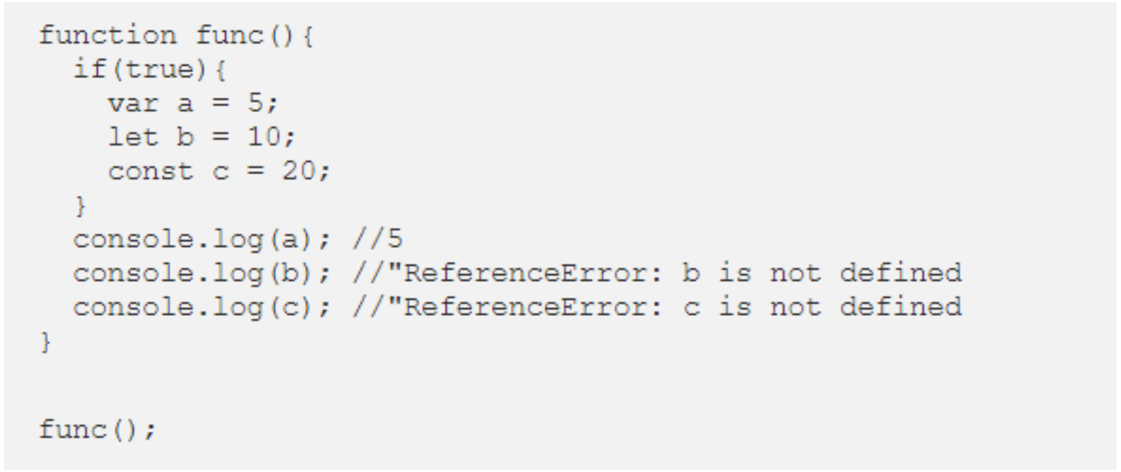


JavaScript

```
function sayHello () {
  const hello = 'Hello CSS-Tricks Reader!'
  console.log(hello)
}

sayHello() // 'Hello CSS-Tricks Reader!'
console.log(hello) // Error, hello is not defined
```

Function scope and Block scope

- **What is block scope?**
  - The variable which is defined with ***const*** or ***let*** within a curly brace ({}) called block scope.
  - The block scope variable can be accessed only within the block of code not outside of the curly braces.



```
function func(){
  if(true){
    var a = 5;
    let b = 10;
    const c = 20;
  }
  console.log(a); //5
  console.log(b); //"ReferenceError: b is not defined
  console.log(c); //"ReferenceError: c is not defined
}

func();
```

Sumary

- Understand what is scope?
- 3 types of scope
- Mechanism of lexical
- Function scope and Block scope

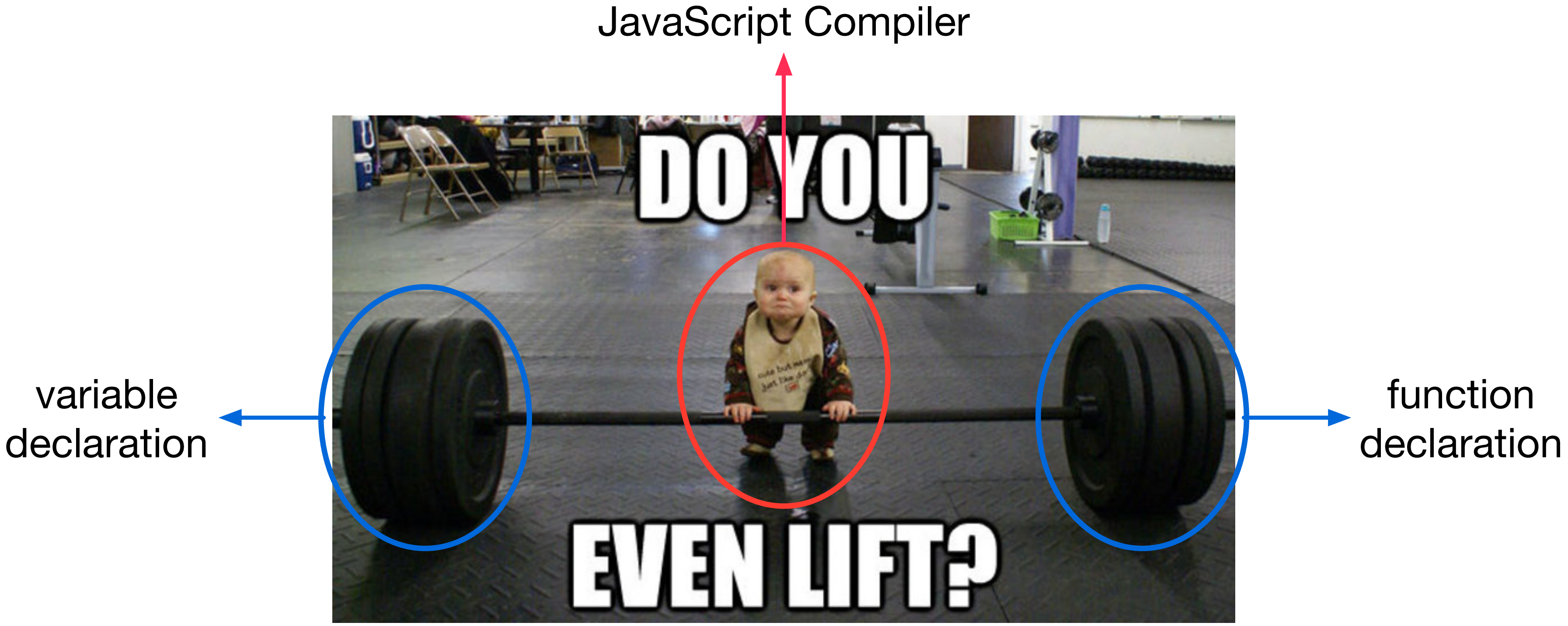
Section 4

Hoisting

Hoisting – Hard Version

- **What is Hositing?**
  - Every declaration (function declration, variable declaration) is moved to the top of current scope (where it is declared in Source Code) during Compilation phase

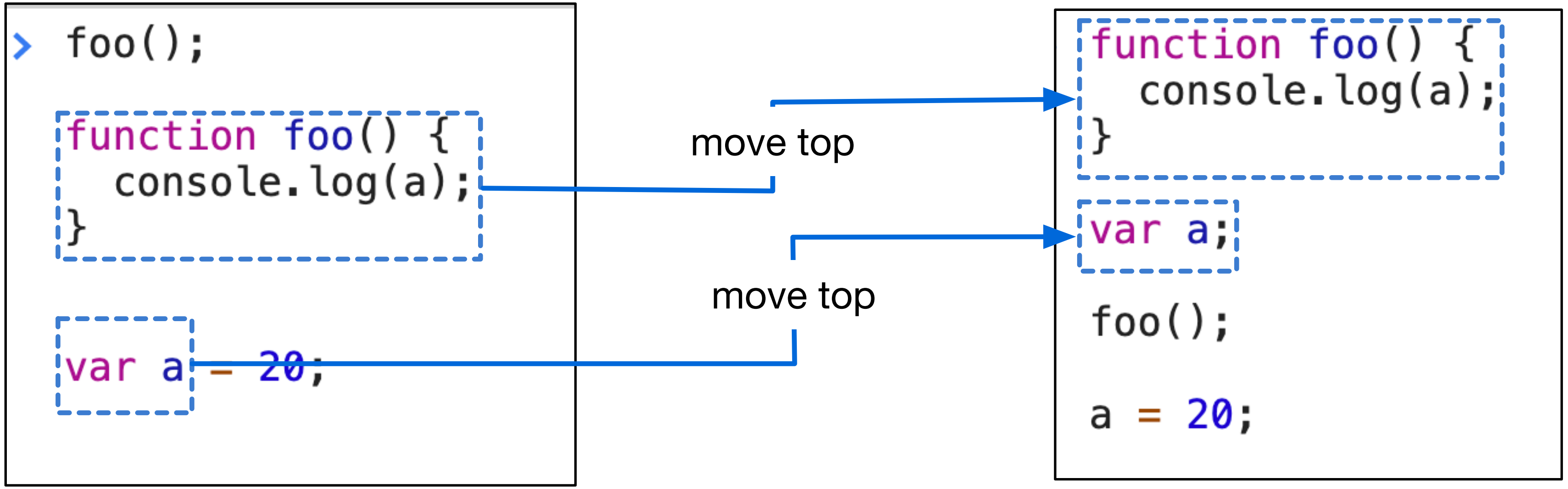
Hoisting – Easy version



Hoisting – Example

What Developer see

What Compiler see



```
> foo();

function foo() {
  console.log(a);
}

var a = 20;
```

```
function foo() {
  console.log(a);
}
var a;
foo();

a = 20;
```

Sumary

- Understand what is Hoisting?
- Understand the process behind Hoisting
- Able to reproduce Source Code after hoisting

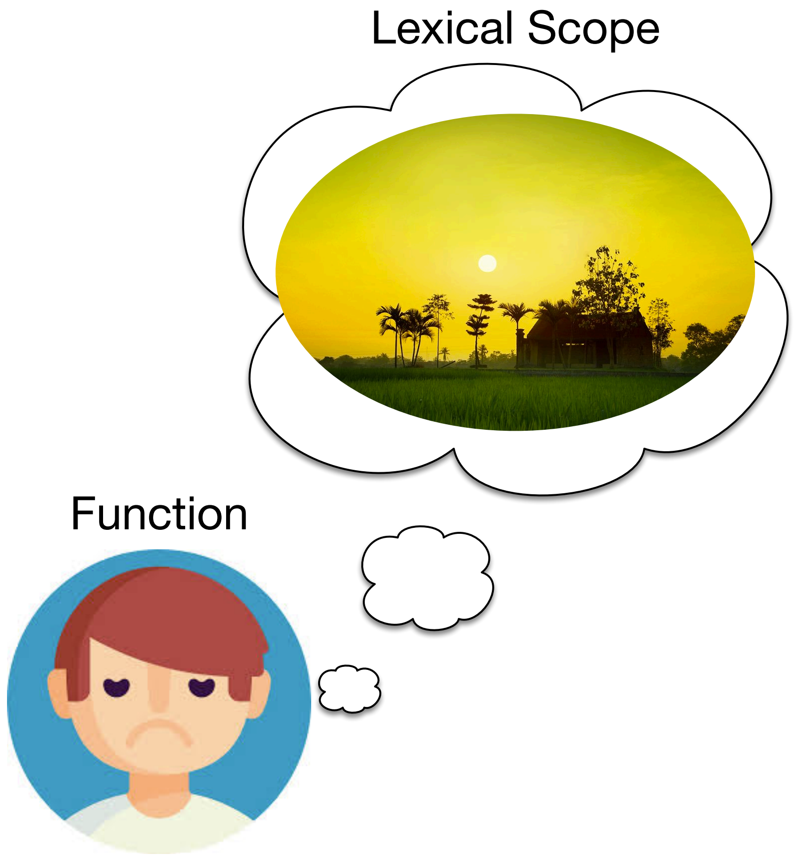
Section 4

Closure

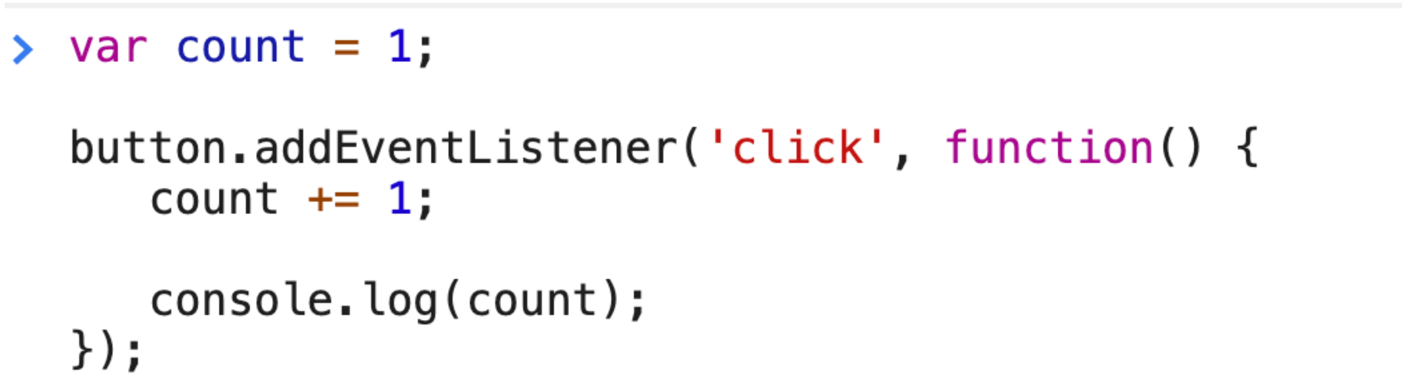
Closure

- **What is Closure?**
  - "Closure is when a function is able to remember and access its lexical scope even when that function is executing outside its lexical scope.”

Closure

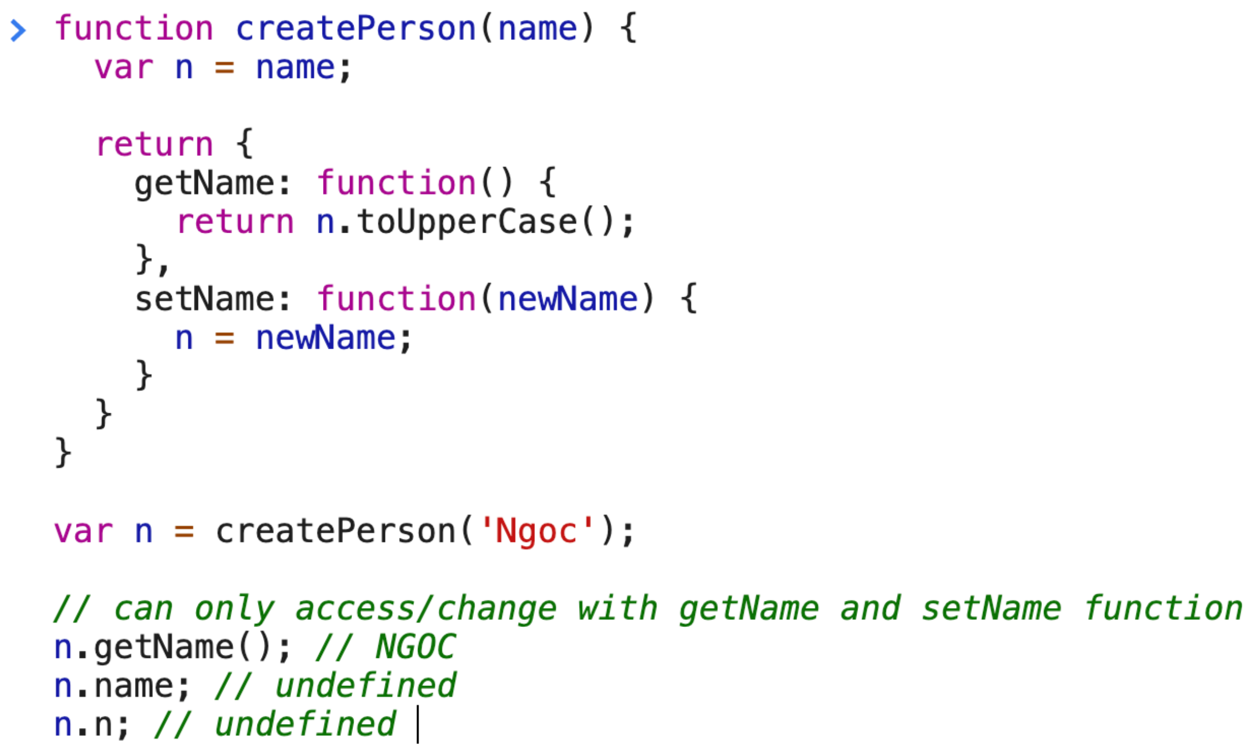


Closure: Event Handler



```
> var count = 1;

  button.addEventListener('click', function() {
      count += 1;

      console.log(count);
  });
```

Closure usage: Encapsulation



```
> function createPerson(name) {
    var n = name;

    return {
      getName: function() {
        return n.toUpperCase();
      },
      setName: function(newName) {
        n = newName;
      }
    }
  }

  var n = createPerson('Ngoc');

  // can only access/change with getName and setName function
  n.getName(); // NGOC
  n.name; // undefined
  n.n; // undefined
```

Sumary

- Understand the definition of Closure
- Understand Closure example in Event Handler and Encapulation

Reference

https://github.com/getify/You-Dont-Know-JS/blob/1st-ed/scope%20%26%20closures/ch1.md

Thank you