作用域和闭包

题目

- 说一下变量的提升的理解?
- 说明this几种不同使用的场景
- 创建10个a标签，点击的时候弹出来对应的序号
- 如何理解作用域
- 实际开发中闭包的作用

知识点 · 一

执行上下文

- 范围
  - 一段<script>里面的js代码
  - 一个函数
  - 即执行上下文并不是只有一个，而是会存在多个
- 全局 — 会将script中的变量和函数提前声明到全局作用域
- 函数 — 在执行的时候，会将函数中的变量、函数声明、this、arguments提前声明

还有一点必须注意，不要再使用var！！！！！！！！！！ 因为var会导致一些匪夷所思的错误

```
// console.log(a);
// 会报错:Cannot access 'a' before initialization
// 大意是无法在初始化之前访问'a'
// 可以看出来虽然下面a被定义了,但是如果在定义前去使用的话就会报错
// 原因是此时压根就没有a这个变量,尽管它在下面被定义了
let a=100

fn("edison")
function fn(name){
    age=100
    console.log(name,age)
    // 这里的报错和上面一模一样,不在赘述
    let age
}
```

this

要在执行时才能确定其值，定义时无法确认

谁调用this，this就指向谁

```
let a={
    x:"x",
    fn:function(){
        // this要在执行前才能确认
        console.log(this.x)
    }
}
a.fn() // a去调用fn,所以this指向fn
a.fn.call({x:"y"})
let fn1=a.fn
fn1() //undefined
console.log(fn1) // [Function: fn]
// fn1此时是一个函数,是全局函数,所以是window调用的,
// 那么fn1的this就指向window,所以也能解释为什么fn1()为undefined
```

```
function FN(){
    let a=100
    return function(){
        console.log(1,a); //1 100
    }
}
let fn=FN()
let a=200
console.log(2,fn()) //2 undefined
// 这里的undefined是因为fn接受的实质上是FN return 回来的函数
```

闭包

能够让内部函数访问到外部函数的变量

知识点 · 二

作用域链

- 全局作用域外面又称之为0级作用域
- 定义函数或者代码块都会开启的作用域1级/2级/3级/...作用域
- JavaScript会将这些作用域链接在一起形成一个链条, 这个链条就是作用域链
  - 0 ----> 1 ----> 2 ----> 3 ----> 4
- 除 0级 作用域以外, 当前作用域级别等于上一级 + 1

作用域

- 在es6之前，不存在块级作用域一说，这就导致了很多莫名其妙的错误
- 这也是为什么不在使用var，而全部使用let去代替。因为let声明的变量是具有块级作用域的

```
// 如果在这里定义x,if里面x才能生效
let x="edison"
if(1){
    // name只在if这个块中才有效
    let name="edison"
    console.log(x)
}else{
    // err 可见连name只在if块中起作用,即使他是else
    console.log(name)
}
// err  无法访问name
console.log(name)
// 如果在这里定义x,if里面x不会生效
let x="edison"

// 所以,为了避免错误,一定要将需要使用的变量提前声明!!!
```

- 子主题 3