题目

- this 的不同应用场景，如何取值？
- 手写 bind 函数
- 实际开发中闭包的应用场景，举例说明

题目

```
// 创建 10 个`<a>`标签，点击的时候弹出来对应的序号
let i, a
for (i = 0; i < 10; i++) {
    a = document.createElement('a')
    a.innerHTML = i + '<br>'
    a.addEventListener('click', function (e) {
        e.preventDefault()
        alert(i)
    })
    document.body.appendChild(a)
}
```

知识点

- 作用域和自由变量
- 闭包
- this

作用域

```
let a = 0
function fn1() {
    let a1 = 100

    function fn2() {
        let a2 = 200

        function fn3() {
            let a3 = 300
            return a + a1 + a2 + a3
        }
        fn3()
    }
    fn2()
}
fn1()
```

作用域

- 全局作用域
- 函数作用域
- 块级作用域（ES6 新增）

作用域

```
// ES6 块级作用域
if (true) {
    let x = 100
}
console.log(x)  // 会报错
```

闭包

- 作用域应用的特殊情况，有两种表现：
- 函数作为参数被传递
- 函数作为返回值被返回

闭包

imooc

```
// 函数作为返回值
function create() {
    let a = 100
    return function () {
        console.log(a)
    }
}
let fn = create()
let a = 200
fn()
```

```
// 函数作为参数
function print(fn) {
    let a = 200
    fn()
}
let a = 100
function fn() {
    console.log(a)
}
print(fn)
```

自由变量

- 一个变量在当前作用域没有定义，但被使用了
- 向上级作用域，一层一层依次寻找，直至找到为止
- 如果到全局作用域都没找到，则报错 xx is not defined

```
// 函数作为参数被传递
function print(fn) {
    const a = 200
    fn()
}
const a = 100
function fn() {
    console.log(a)
}
print(fn) // 100

// 闭包：自由变量的查找，是在函数定义的地方，向上级作用域查找
//       不是在执行的地方！！！
```

this

- 作为普通函数
- 使用 call apply bind
- 作为对象方法被调用

this

- 在 class 方法中调用
- 箭头函数

this 取什么的值，只在函数执行时确定的，而不是在函数定义 中

this

```
function fn1() {
    console.log(this)
}
fn1() // window

fn1.call({ x: 100 }) // { x: 100 }

const fn2 = fn1.bind({ x: 200 })
fn2() // { x: 200 }
```

this

```
const zhangsan = {
    name: '张三',
    sayHi() {
        // this 即当前对象
        console.log(this)
    },
    wait() {
        setTimeout(function() {
            // this === window
            console.log(this)
        })
    }
}
```

```
const zhangsan = {
    name: '张三',
    sayHi() {
        // this 即当前对象
        console.log(this)
    },
    waitAgain() {
        setTimeout(() => {
            // this 即当前对象
            console.log(this)
        })
    }
}
```

this

```
class People {
    constructor(name) {
        this.name = name
        this.age = 20
    }
    sayHi() {
        console.log(this)
    }
}
const zhangsan = new People('张三')
zhangsan.sayHi() // zhangsan 对象
```

题目解答

- this 的不同应用场景，如何取值？
- 手写 bind 函数
- 实际开发中闭包的应用场景，举例说明

手写 bind 函数

```
Function.prototype.bind1 = function () {
    // 将参数解析为数组
    const args = Array.prototype.slice.call(arguments)
    // 获取 this（取出数组第一项，数组剩余的就是传递的参数）
    const t = args.shift()
    const self = this // 当前函数
    // 返回一个函数
    return function () {
        // 执行原函数，并返回结果
        return self.apply(t, args)
    }
}
```

实际开发中闭包的应用

- 隐藏数据
- 如做一个简单的 cache 工具