ES6

ES

什么是ES?

const 标识常量

```
    const LIMIT = 10
    const OBJ_MAP = {
        a: 'A',
        b: 'B'
    }

    // 常量 常的是什么？常量有什么特征?
```

1. 不允许重复声明赋值

```
    // 变量重新赋值
    var arg1 = 'yy'
    arg1 = 'student'

    // 常量
    // ES5
    Object.defineProperty(window, 'arg2', {
        value: 'yy',
        writable: false
    })
    console.log(arg2)
    arg2 = 'student'
    console.log(arg2)

    // ES6
    const arg3 = 'yy'
    console.log(arg3)
    arg3 = 'student'
    // ERROR：常量不可被变更

    const arg4
    arg4 = 'yy'
    // ERROR：使用常量的时候，在声明时需要初始化

    const arg5 = 'yy'
    const arg5 = 'student'
    // ERROR: 重复声明
```

2. 块级作用域

```
    if (true) {
        var arg1 = 'yy'
    }
    console.log(arg1)

    if (true) {
        const arg1 = 'yy'
    }
    console.log(arg1)
```

3. 无变量提升

```
    console.log(arg1)
    var arg1 = 'yy'

    // 相当于
    var arg1
    console.log(arg1)
    arg1 = 'yy'

    console.log(arg2)
    const arg2 = 'yy'

    // dead zone
    if (true) {
        console.log(arg3)
        const arg3 = 'yy'
    }
```

进一步探讨

```
    const obj = {
        teacher: 'yy',
        course: 'zhaowa'
    }
    obj.course = 'es'
    obj = {}

    // 引用类型
    // obj => 栈（地址） => 堆（内容值）

    // 面试：破局方法？- 指向地址 - Object.freeze()
    const obj2 = {
        teacher: 'yy',
        course: 'es'
    }
    Object.freeze(obj2)
    obj2.course = 'ts'
    // 属性被冻结

    // 追问：
    const obj3 = {
```

```
        teacher: 'yy',
        course: 'es',
        zhaowa: {
            score: 5
        }
    }
    Object.freeze(obj3)
    obj3.zhaowa.score = 4.9
    // freeze只能冻结当前层
    function deepFreeze(obj) {
        Object.freeze(obj)
        (Object.key(obj) || []).forEach(key => {
            if (typeof obj[key] === 'object') {
                deepFreeze(obj[key])
            }
        })
    }
```

deconstruction 解构

```
    const zhaowa = {
        teacher: 'yy',
        course: 'es'
    }
    // 原始
    const teacher = zhaowa.teacher
    const course = zhaowa.course

    // ES6
    const {
        teacher,
        course
    } = zhaowa

    const arr = ['yy', 'zhaowa', 'es']
    const a = arr[0]
    const b = arr[1]
    const c = arr[2]

    const [a, b, c] = arr

    // 别名
    const {
        teacher: {
            name,
            age
        },
        course,
        name: className
    } = zhaowa

    // 变量交换
    let a = 1
    let b = 2
    let tmp = a
    a = b
    b = tmp
```

```
38
39      let a = 1
40      let b = 2
41      [b, a] = [a, b]
```

arrow_function 箭头函数

```
1       // 传统函数
2       function test(a, b) {
3           return a + b
4       }
5       // 传统函数表达式
6       const test = function(a, b) {
7           return a + b
8       }
9
10      const result = test(1, 1)
11
12      // ES6箭头函数
13      const test = (a, b) => {
14          return a + b
15      }
16
17      const test = (a, b) => a + b
18
19      const test = x => x + 1
```

上下文

```
1       const obj = {
2           teacher: 'yy',
3           course: 'es',
4           zhaowa: {
5               score: 5
6           },
7           getTeacher: function() {
8               console.log('function', this)
9           },
10          getCourse: () => {
11              console.log('arrow_function', this)
12          }
13      }
```

注意以下场景

```
1       // dom操作
2       const btn = document.querySelector('#btn')
3       btn.addEventListener('click', function() {
4           this.style.color = '#fff'
```

```
})

// 类操作
function Obj(teacher, course) {
    this.teacher = teacher
    this.course = course
}

const Obj1 = (teacher, course) => {
    this.teacher = teacher
    this.course = course
}

const o = new Obj('yy', 'es')
console.log('o', o)

const o1 = new Obj1('yy', 'es')
console.log('o1', o1) // Error

Obj.prototype.learn = function() {
    console.log('function', this.teacher, this.course)
}

Obj.prototype.learn = () => {
    console.log('arrow_function', this.teacher, this.course)
} // Error

// arguments 操作
const test = function(teacher) {
    console.log(arguments)
}
test('yy')

const test = teacher => {
    console.log(arguments)
} // 无法使用arguments
```