**LUOJIN 2020-05-12(TUES)**

[1，变量 2](#_Toc40657692)

[2， const 常量： 3](#_Toc40657693)

[3，var, let and const的区别： 3](#_Toc40657694)

[4, 解构赋值 3](#_Toc40657695)

[4.1 数组解构 3](#_Toc40657696)

[4.2 对象解构 4](#_Toc40657697)

[5, 箭头函数 - 定义函数的方式 4](#_Toc40657698)

[5.1 箭头函数写法： 4](#_Toc40657699)

[5.2 箭头函数面试题： 4](#_Toc40657700)

[6. 剩余参数 4](#_Toc40657701)

[6.1 当实参的个数大于形参个数时 4](#_Toc40657702)

[6.2 剩余参数和解构配合使用 5](#_Toc40657703)

[7. 扩展运算符 5](#_Toc40657704)

[7.1 剩余 5](#_Toc40657705)

[7.2 扩展运算符 5](#_Toc40657706)

[7.3 应用：合并数组 5](#_Toc40657707)

[8. 构造函数方法 6](#_Toc40657708)

[9, 实例方法 6](#_Toc40657709)

[10. 模板字符串 7](#_Toc40657710)

[11. 字符串方法 7](#_Toc40657711)

[12. set 数据结构 8](#_Toc40657712)

# 1，变量

let

- 1）, 具有块级作用域

- 2）, 防止循环变量变成全局变量

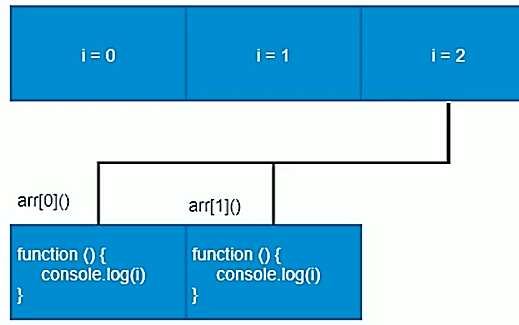
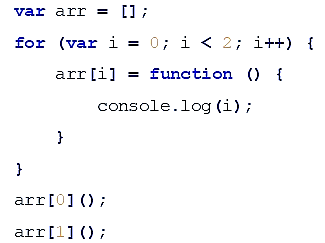
- 3）, 不存在变量提升 （先声明在使用，否则就会报错）

console.log(a); //a is not defined

let a = 20;

- 4, 具有暂时性死区（在块级作用域内部，声明的变量，会被绑定在这个块级中。不受外部影响，两个不会相互影响）； 在变量声明前使用，会报错

**Let 经典面试题：**

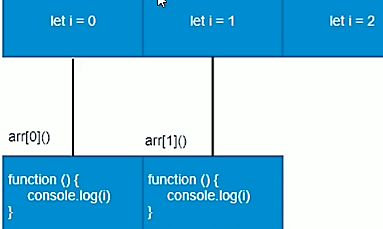
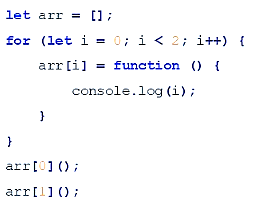


在i=0，i=1，循环体内的函数并没有执行，给数组增加值

循环外：取数组的值，调用函数

**【问题】函数执行完，输出的I = ?**

【回答】：函数内部，没有定义变量i， 在上层作用域上找，也就是全局作用域，函数执行时，循环早就结束了。I=2，已经不满足循环条件了，跳出循环，所以输出都是 i=2



**【经典面试题 - 问题2】将上边的var 改为let， 输出的结果？**

【回答】：let 具有块级作用域，循环结束后产生了两个块级作用域，互不影响。函数内部还是没有自己的变量I, 上级就是循环产生的块级作用域。数组中第一个函数执行块级作用域产生变量i的值0，数组中第二个函数执行块级作用域产生的第二个变量i=1; 所以输出结果为 0 ， 1

# 2， const 常量：

作用：值不可以改，值（内存地址），

* 1），也具有块级作用域
* 2），必须赋初始值 const pi; // missing initializer in const declaration
* 3），常量赋值后，不可以更改；//Assignment to constant variable；可以改数组的值，const ary = [100, 200] ary[0] = ‘a’---这样是可以的，因为没有改变数组内部的内存地址值。但是如果**ary=[‘a’, ‘b’]; 这样改就报错，会改变常量对应的存储地址**。内部的值可以改，不能重新赋值。

# 3，var, let and const的区别：



什么时候使用？

* 数据不需要变化，数学公式中恒定不变的，用const

# **4, 解构赋值**

分解数据结构，**从数组中提取值，按照对应位置，对变量赋值**，对象也可以实现解构。

## 4.1 数组解构

* **Let** [a, b, c] = [1, 2, 3] 一一对应的关系提取和赋值
* **Let** [a, b, c, d, e] = [1, 2, 3] 如果不是一一对应，就是undefined

## 4.2 对象解构

- **let person = {name: ‘luojin’, age: 20};** Let {name, age} = **person; console.log(name)**

**- let {name: myName; age: myAge} = person;**

# 5, 箭头函数 - 定义函数的方式

## 5.1 箭头函数写法：

（）=> { }

const fn =(形参) = { } 通常将箭头函数赋值给一个变量

fn(); // 调用函数

* 1），如果函数体只有一句代码，也执行结果就是返回值，可以省略大括号
* 2），如果形参只有一个，形参的小括号也可以省略
* 3），一般函数 ，谁调用他，函数内部的this就指向谁，但是箭头函数没有这样的功能。
* 4），箭头函数不绑定this，**箭头函数没有自己的this**, 如果在箭头函数使用this， 指向的是**函数定义位置中的this**.

## 5.2 箭头函数面试题：

Var obj = { // obj是一个对象，不能产生作用域，箭头函数实际上被定义在了全局作用域下边，全局作用域没有age属性，所以弹出的是undefined

Age: 20,

Say: ( ) = >{alert(this.age)}

}

obj.say(); ***// undefined***

# **剩余参数**

## 6.1 当实参的个数大于形参个数时

将剩余的**实参放到一个数组**中（输出的时候）

Const sum = (**…args**) => {

Let total = 0;

Args.forEach(item =>(Total += item);

};

Sum(10,20); // 30

Sum(10,20,30); //60

在函数内部，我们如何一次取到所有的实参呢？

在之前呢，使用函数内部的arguments

但是箭头函数不能用arguments, 用剩余参数，在函数形参部分写上对应形参，加上三个点

## 6.2 剩余参数和解构配合使用

Let ary1 = [‘changsan’, ‘lisi’, ‘wangwu’]

Let[s1, **…s2**] = ary1; //…s2同时接收数组中剩余的元素

# 扩展运算符

## 7.1 剩余

剩余的实参放在数组中

## 7.2 扩展运算符

可以将数组或者对象拆分转为逗号分隔的参数序列, 将参数序列放在了console.log方法中,逗号被当成了参数分隔符

Let ary = [1,2,3];

**....ary** // 1,2,3

Console.log(...ary) // 1,2,3

## 7.3 应用：合并数组

方法一：

Let ary1 = [1,2,3]; let ary2 = [6,4,5];

Let **ary3 = […ary1, …ary2];**

**方法二：**

**ary1.push(…ary2)** //可以同时接收多个参数

console.log(ary1);

1. 应用2; 伪数组或者可遍历对象转为真数组

Let oDivs = document.getElementsByTagName(‘div’);

Console.log(oDivs);

var ary = […oDivs]; //真数组可以调用数组中的方法了

**ary.push(‘a’); console.log(ary);**

# 8. 构造函数方法

**Array.from()** 将类数组或者可遍历对象转换为真数组

Let arraylike = {

‘0’: ‘a’,

‘1’: ‘b’,

‘2’: ‘c’,

‘length’: 3

}

**var ary = Array.from(arrayLike)**

**第二个参数 可以为一个函数,数组有几个元素，函数就调用几次**

var ary= newAry **= Array.from(arrayLike, (item)= >{item\*2})**;

# 9, 实例方法

1. find() 找出第一个符合条件的数组成员，如果没有找到返回undefined

let ary = [{id: 1, name ‘luojin’}, {id: 2, name: ‘haha’}];

let target = ary.find(**item** => item.id == 2)

((**item,index**) => { return …}

1. findIndex() 值查找第一满足条件的索引，没有赵丹返回-1

let ary = [1,5,10,13]

let ary1 = ary.findeIndex(item = > { return item>10})

1. includes() 看数组中是不是 包含某一个元素，返回布尔值

[1,2,3].includes(1) //true

在ES6之前判断数组中是不是有某个值使用的是indexOf方法

# 10. 模板字符串

新增的创建字符串的方式，使用反引号定义 = 数字1上边的键

Let name = `这是模板字符串`

* **1）模板字符串中可以解析变量 ${变量名}**

Let **name** = `luo jin`

Let sathello = `my name is **${name}`**

* 2）内容可以换行，比如要写入html中的内容
* 3）可以调用函数

Const **fn** = () = > { return ‘我是函数fn’}

Let html = `我是 模板字符串**${fn()}`**

# 11. 字符串方法

1） **两种字符串方法**

**StartsWith()** 判断字符串是不是以~开头，返回布尔值

**endsWith()** 是不是以~结尾

1. repeat()方法： 将字符串重复几次，返回新的 **‘y’.repeat(5)**

# **12. set 数据结构**

1) 它类似于数组，**但是成员的值都是唯一的，没有重复的值**，用在电商网站中的搜索功能

历史记录，搜索历史关键字。内部会判断是不是重复，重复就不会在存储了

Set本身是一个**构造函数**，用来生成set数据结构

const s = new set(); console.log(**s.size);** **看有几个值**

可以**接受一个数组作为参数初始化**

const s = new set([‘a’, ‘d’, 2, 3]);

**2) 数组去重** set不会输出重复的值，会忽略掉重复的值

Const s3=new set([‘’a, ‘a’, ‘b’)

Const ary = […s3]; // ‘[a’, ‘b’]

3). add(value); //const s = new Set(); s.add(1).add(2).add(4)

delete(value) //s.delete(1)

has(value) //s.has(0)

clear() //s.clear()

1. **从Set()** **数据结构张取值,forEach()遍历**

Const s5 = new Set([‘a’, ‘b’, ‘c’])

S5.forEach(value => { console.log(value)})