# 数据类型

基本数据类型：Boolean，String，Number

## let

let是块级作用域变量定义的关键字，不存在变量提升的特点，必须先定义再使用。

## const

const是块级作用域定量定义的关键字，不存在变量提升的特点，必须先定义再使用。

## Map

Map属于高级变量，用于存储键值对，Map可用于存储Number，Boolean，String，Function数据；可使用for of或foreach遍历Map

定义：let map = new Map();

存键值对：map.set('name','Jarry');

根据键名取值：let name = map.get('name');

## 暂时性死区

{…}里外定义的变量名即便一样，块级作用域仍然会起作用，一块归一块使用。

# 解构赋值

## 扩展运算符

//…代表将数据扩展到另一个数组中，在作为函数参数时必须放置到最后一个参数

let a = [1,2,3,4];

let b = [5,6,7,8];

b = [...a,...b];

console.log(b);//输出结果 [1, 2, 3, 4, 5, 6, 7, 8]

## 别名赋值

let {name:n,age:g,sex:s} = {name:'张三',age:17,sex:'男'};//只要记住:是替名【通常用于对象】

## 默认值赋值

let [a,b,c=’暂无数据’] = [’aa’,’bb’];

## 对象解构

let obj = {

a:[1,2,3,4,5,6,7],

b:{

name:'Jack',

age:18,

sex:'man'

}

}

let {a,b:{name,age,sex}} = obj;//b不是变量故而不可使用，仅是模式，解构时其名一定与obj一样。

console.log(a,name,age,sex);//

# 数组

## 扯平数组

//无论数组中有多少嵌套数组，都可以扯成一维数组

let\_arr = [1, [7, [8,9, [10]]]]**.flat(Infinity)**;//结果[1,7,8,9,10],Infinity还可以被替换成数字表示从最里面那层开始计数扯n层。建议一般使用**Infinity**

## 查找max/min值

//查找数组中的最大值

let num\_arr=[3,4,5,6,1,2];

console.log(**Math.max**(...this.num\_arr));//结果：6

//查找数组中的最小值

let num\_arr=[3,4,5,6,1,2];

console.log(**Math.min**(...this.num\_arr));//结果：3

## 查找第一个符合条件的数组值

//find(***Fun***)的参数是一个回调函数【需有return】，函数参数最多3个，最少1个。

let arr = [1,2,3,4,5,6,7,8,9,10];

let num = arr**.find**((item,index,arr)=>{ return item>6 });//返回数组中第一个大于6的数据

## 查找第一个符合条件的数组值的位置

//find(***Fun***)的参数是一个回调函数【需有return】，函数参数最多3个，最少1个。

let arr = [1,2,3,4,5,6,7,8,9,10];

let num = arr**.findIndex**((item,index,arr)=>{ return item>6 });//返回数组中第一个大于6的数据在数组中的位置

## 数组填充

//填充数组，可以填充某个字符，也可以限制填充位置，也可以只写一个字符，表示填充整个数组

let let\_arr=['a','a','a','a','a','a','a','a','a'].fill('b',1,4);//用b元素覆盖元素组中第1位到第3位

## 排序函数

//对数字型数组排序升序

let num = [0,7,2,5,4,6,8,9,6,1,3];

console.log(num**.sort()**);//结果[0, 1, 2, 3, 4, 5, 6, 6, 7, 8, 9]

## 循环函数

//对数组循环，并执行指定方法，不建议在循环中直接改变arr数据，如arr[index]

let arr = [11,22,33,44,55];

arr**.forEach**((val,index,arr) => {

console.log(val,index,arr)

});

## 映射函数

/\*对数组不但循环而且进行相应的操作，并且需要返回值的的情况下使用，如将数组中所有11的倍数除以11并返回\*/

let arr = [11,1,2,22,33,44,55];

let newarr = arr**.map**((value) => {

if ( value % 11==0) {

return value / 11;

}else{

return value;

}

});

console.log(newarr,arr);//打印结果：[1, 1, 2, 2, 3, 4, 5]

## 过滤函数

/\*回调函数返true则留下元素，返false去除元素。注：回调函数仅支持返Boolean值，结果为数组\*/

let num = [1,2,3,4,5,6,7,8,9,10,11,12,13];

let newnum = num**.filter**((item,index,arry)=>{

if ( item % 2 == 0 ) {

return true;

} else {

return false;

}

});

console.log(newnum);//打印结果：[2, 4, 6, 8, 10, 12]

## 查找函数

//数组中是否有需查找的元素，或计算结果是否符合需求，返回Boolean值，结果是Boolean值

let num = [1,2,3,4,5,6,7,8,9,10,11,12,13];

let num2 = num**.some**((item,index,arry)=>{

return item == 121;

})

console.log(num2);//打印结果：false

## 检查函数

//检查数组中每一个值是否符合规则，返回Boolean值，接收结果为Boolean值

let num = [2,4,6,8,10];

let num3 = num**.every**((item,index,arry)=>{

return item%2 == 0;

});

console.log(num3);//返回结果：true

## 循环运算函数

/\*从左往右将数组中每一个值纳入计算，最终返回计算总值；第一个参数值暂存当前运算结果\*/

let num = [2,4,6,8,10];

let num4 = num.reduce((pre,cur,index,array)=>{

return pre\*cur;

})

console.log("乘：",num4);//打印结果：乘： 3840

/\*从右往左将数组中每一个值纳入计算，最终返回计算总值；第一个参数值暂存当前运算结果\*/

let num = [1,2,3];

let num4 = num.reduceRight((pre,cur,index,array)=>{

return pre\*\*cur;

})

console.log("阶乘：",num4);//打印结果：乘： 9

## ES6 for…of…循环

//循环遍历数组，数组某一项.entries(),序号.keys()，建议 of 对象即可。

let num = [1,2,3];

for (const iterator of num) {

console.log(iterator);//打印结果：1，2，3

}

## 补充

JavaScript中的数组也拥有key，value值，key值即数组元素index值，value值即数组元素；entries()，keys() 和 values()是ES6提供的可用于遍历数组时分别获取当前数组的对象，key值，value值的新方法。

# 数字

## 判断变量是否存在值

//判断一个变量是否是有值，不管变量是整数还是小数都会返回true，其余的全部返回false

let a = 100.90,b,c=NaN;

console.log(Number**.isFinite**(a),Number**.isFinite**(b),Number**.isFinite**(c));//打印结果：true false false

## 判断变量是否=NaN

//是就返true，否就返false

let b,c=NaN;

console.log(Number**.isNaN**(b),Number**.isNaN**(c));//打印结果：true false

## 判断变量是否为整数

//是就返true，否就返false

let aa = 1.03,bb = 2;

console.log(Number**.isInteger**(aa),Number**.isInteger**(bb)); //打印结果：false true

## 转整型

//无论Number还是String类型的都可以转换成整型，返回整型Number数据

let aa = '1.03', a = 100.9;

console.log(Number.parseInt(a),Number.parseInt(aa));//打印结果：1 100

## 转浮点型

let aa = '1.03', a = 100.9;

console.log(Number.parseFloat(aa),Number.parseFloat(a));//打印结果：1.03 100.9

## ECMA2016 数据精度计算

numObj**.toFixed**(精度位数)//12.546保留2位精度就会输出12.55

numObj**.toPrecision**(保留长度)//包括整数在内的保留数据长度，如果数据不足则用0填充

## Math系列函数

Math.trunc(…);//取整：返回一个数据的整数部分

Math.sign(…);//判断正负，正返回1，0返回0，负返回-1；可以判断String类型数字

指数运算符\*\* let a = 2\*\*3;//2的三次方

# 字符串

## 字符串模板

//ES6提供的字符串模板，支持无转义字符进行换行，使用${…}包裹其他值，若字符串存在反引号，则用\转义，若涉及到模板，其中js部分使用<%= …… %>包裹，但js这块在vue项目根本用不上.

let n = '张三',g = 18;

let introduce = `大家好，我叫**${**n**}**，我今年**${**g**}**岁了`;

console.log(introduce);

## 获取字符串长度

StrObj.length;//获取字符串长度，返回Number类型

## 清除首部空格

StrObj.trimStart();//不改变StrObj，返回清除StrObj首部空格的结果

## 清除尾部空格

StrObj.trimEnd();//不改变StrObj，返回清除StrObj尾部空格的结果

## 清除首尾空格

StrObj.trim();//不改变StrObj，返回清除StrObj首尾空格的结果

## 字符串原样输出

String.raw('…');保证字符串原样输出，不区分转义字符了；在vue项目中极少用。

## 字符串查找函数

//查找子字符串是否存在于目标字符串中

let introduce = `大家好，我叫${n}，我今年${g}岁了`;

console.log(introduce,introduce**.includes**('大家好'));//返回值true false

## 检测字符串开头函数

//验证字符串是否以xxx开头，也可写第二个参数Number型，表示从第n个字符开始检查

let introduce = `大家好，我叫${n}，我今年${g}岁了`;

console.log(introduce**.startsWith(**'家好'));//返回值Boolean值

## 检测字符串结尾函数

//验证字符串是否以xxx结尾

let introduce = '大家好，我叫张三，我今年18岁了';

console.log(introduce**.endsWith**('岁了'));//返回值Boolean值

## 字符串重复函数

//将目标字符串重复n次再赋值回去【n值必须>1】

let i = 'good';

console.log(i**.repeat**(5));//打印结果：goodgoodgoodgoodgood

## 字符串头部填充

/\*截取字符串填入目标字符串首部，.padStart(,),第一个参数是填充后字符串的总长度，如果结果总长度≤i，返回结果不会改变；长度大于i则会截取在j中从头开始截取( 总长度 - i.length)的长度填充到i首部。\*/

let i = 'efghijklm',j = 'abcd';

console.log(i**.padStart**(i.length+j.length, j));//打印结果abcdefghijklm

## 字符串尾部填充

/\*将截取字符串填入目标字符串尾部，padStart(length,str),第一个参数是结果总长度，如果结果总长度≤i，返回的结果不会改变，长度大于i则会截取在j中从头开始截取( 总长度 - i.length)的长度填充到i尾部。\*/

let i = 'abcd',j = 'efghijklm';

console.log(i**.padEnd**(i.length+j.length,j));//打印结果abcdefghijklm

# 函数

## 申明

块级作用域可以申明，并且只允许在定义块级用，

## 参数允许设定默认值

//默认值用等号连接，注意：第一个参数必须传值，不然报错！故而第一个参数没必要设定默认值

function show( a = 10, b = 10 ){

console.log( a/b );

}

show(100,10);//结果10

show();//结果1

show(20,);//结果2

//show(,5);//错误-第一个参数必须传值

/\*对象做参数仍然通用\*/

function show2( { a = 'Game' , b = 'over' }={} ){

console.log( a,b );

}

show2();//输出Game over

show2({a:'Junk',});//输出Junk over

show2({a:'Junk',b:'food'});//输出Junk food

## ||运算符

y = y||’hello’//如果y没有值，哪怕是空字符串‘’，都会被hello替代

typeof(…);//判断数据是什么类型，返回值String类型

## 箭头函数

箭头函数格式：let 函数名=(参数)=>{函数内容}

注意：

1. this问题，不绑定任何this，会捕捉其所在的上下文的this作为自己的this
2. 箭头函数里面没有arguments，用’…’
3. 箭头函数不能当构造函数