█字符串

●字符串模板

使用`包裹，也就是Tab上面的那个键。包裹的字符串可以使用${变量名}来拼接，而且是支持计算变量的。还可以使用HTML标签

如下例：

let a='JOE';

let b=2;

let c=3;

let d=`<h3>这里是字符串模板的例子：拼接${a}的内容,和运算的结果${b+c}</h3>`;

document.write(d);

●includes方法

let a='abcde';

let b='c';

// 判断a是否包含b，返回布尔值

document.write(a.includes(b));

// 等同于下面使用ES5的解决办法

// document.write(a.indexOf(b)>=0);

●startsWith()方法和endWith()方法

let a='abcde';

let b='e';

// 判断a的开头是否是b，返回布尔值

document.write(a.startsWith(b));

// 判断a的结尾是否是b，返回布尔值

document.write(a.endsWith(b));

●复制字符串

// 复制'abc'5次

document.write('abc'.repeat(5));

█数组

●JSON的数组格式

●Array.from()方法

// 标准的json数组格式写法如下：

let json = {

'0': 'jspang',

'1': '技术胖',

'2': '大胖逼逼叨',

length:3

}

// 使用Array.from(json的数组格式),将json的数组格式转化成一个数组

let arr=Array.from(json);

console.log(arr)

●Array.of()方法

// 比如后台传过来的数组是这样的字符串

let a='[1,2,3,4,5,6]';

// 可以使用Array.of方法将乱七八糟的东西转化成数组

console.log(Array.of(1,2,3,4,5,6));

●find()实例方法

注意，不是一般的方法，是实例方法

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

console.log(a.find(function(value,index,arr){

// value是当前查找的值

// index是索引

// arr是原型

// 比如查找并返回一个大于5的值，注意，只返回一个。下面例子返回的是6

return value > 5;

}))

console.log(a.find(function(value,index,arr){

// value是当前查找的值

// index是索引

// arr是原型

// 比如查找并返回一个索引大于2的值，注意，只返回一个。下面例子返回的是4

return index > 2;

}))

●fill()实例方法

// fill()也是一个实例方法，它的作用是把数组进行填充，

// 它接收三个参数：

// 第一个参数是填充的变量

// 第二个是开始填充的位置

// 第三个是填充到的位置，注意，到这个位置结束，而不替换这个位置，也就是说填充的内容到这个位置之前结束

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

arr.fill('jspang',2,5);

console.log(arr);

// 上边的代码是把数组从第二位到第五位用jspang进行填充，注意，不填充第5位。

// 结果是[0, 1, "jspang", "jspang", "jspang", 5, 6, 7, 8, 9]

●for…of循环

在ES6中用for…of循环来代替for循环来遍历数组

let arr=['jspang','技术胖','大胖逼逼叨']

for (let item of arr){

console.log(item);

}

配合keys()实例方法来遍历数组的索引

let arr=['jspang','技术胖','大胖逼逼叨']

for (let index of arr.keys()){

console.log(index);

}

配合entries()实例方法来同时遍历内容和索引

let arr=['jspang','技术胖','大胖逼逼叨']

for (let [index,val] of arr.entries()){

console.log(index+':'+val);

}

●entries()实例方法的用途

// entries()实例方式生成的是Iterator形式的数组

// 这种形式的好处就是可以让我们在需要时用next()手动跳转到下一个值。

let arr=['jspang','技术胖','大胖逼逼叨']

let list=arr.entries();

console.log(list.next().value);

console.log(list.next().value);

console.log(list.next().value);

●in的用法

// in是用来判断对象或者数组中是否存在某个值的。

// 对象判断

let obj={

a:'jspang',

b:'技术胖'

}

console.log('a' in obj); //true

// 判断数组

// 以前会使用length属性进行判断，为0表示没有数组元素。但是这并不准确

let arr=[,,,,,];

console.log(arr.length); //5 其实数组是空的，但是判断结果显示5个空位

// 那用ES6的in就可以解决这个问题。

let arr1=[,,,,,];

console.log(0 in arr1); //false

let arr2=['jspang','技术胖'];

console.log(0 in arr2); // true

█解构赋值

// 数组的解构赋值

let [a,[b1,b2],c]=[1,[2,3],4];

console.log(a);

console.log(b1);

console.log(b2);

console.log(c);

// 对象的解构赋值

let {foo,bar} = {foo:'JSPang',bar:'技术胖'};

console.log(foo+bar); //控制台打印出了“JSPang技术胖”

// 注意：对已声明的变量进行对象解构赋值，需要用括号包起来，否则会报错。如下例

let foo1;

({foo1} ={foo1:'JSPang'});

console.log(foo1); //控制台输出jspang

// 字符串的解构赋值

const [aa,bb,cc,dd,ee,ff]="JSPang";

console.log(aa);

console.log(bb);

console.log(cc);

console.log(dd);

console.log(ee);

console.log(ff);

// 对象的函数解构

let json1 = {

aaa:'jspang',

bbb:'技术胖'

}

function fun({aaa,bbb='默认值'}){

console.log(aaa,bbb);

}

fun(json1);

// 数组的函数解构

let arr = ['jspang','技术胖','免费教程'];

function fun1(ar,br,cr){

console.log(ar,br,cr);

}

fun1(...arr);

█扩展运算符

●扩展运算符（…）的用法

// 对象扩展运算符（…）

function jspang(...arg){

console.log(arg[0]);

console.log(arg[1]);

console.log(arg[2]);

console.log(arg[3]);

}

jspang(1,2,3);

// 扩展运算符的妙用

let arr1=['www','jspang','com'];

let arr2=arr1;

console.log(arr2);

arr2.push('shengHongYu');

console.log(arr1);

// 由于对象复制是引用关系，所以改变arr2的时候arr1也被改变了，输出结果如下

// ["www", "jspang", "com"]

// ["www", "jspang", "com", "shengHongYu"]

// 解决以上问题可以使用扩展运算符

let arr3=['www','jspang','com'];

let arr4=[...arr3]; //代替let arr4=arr3;

console.log(arr4);

arr4.push('shengHongYu');

console.log(arr4);

console.log(arr3);

●rest运算符（…）

// 比如只确定第一个参数first而之后有几个参数不确定，就用...rest运算符

function jspang(first,...arg){

console.log(arg.length);

}

jspang(0,1,2,3,4,5,6,7);

█箭头函数

// 知识点一：如果函数体只有一句，一个return则可省略return，直接在箭头右侧写return的值

let a=()=>3;

console.log(a());

// 等同于

// var a=function(){

// return 3;

// }

// console.log(a());

let b=(x,y)=>x+y;

console.log(b(2,3));

// 等同于

// var b=function(x,y){

// return x+y;

// }

// console.log(b(2,3));

// 知识点二：如果函数体不只一句retrun，则需要将箭头右侧用{}包裹起来

let c=(x,y)=>{

let c1=x+y;

return c1;

}

console.log(c(3,5));

// 等同于

// var c=function(x,y){

// var c1=x+y;

// return c1;

// }

// console.log(c(3,5));

█数字

// 数字验证Number.isFinite( xx )

// 只要是数字，不论是浮点型还是整形都会返回true，其他时候会返回false。

let a= 11/4;

console.log(Number.isFinite(a));//true

console.log(Number.isFinite('jspang'));//false

console.log(Number.isFinite(NaN));//false

console.log(Number.isFinite(undefined));//false

// NaN验证

console.log(Number.isNaN(NaN)); //true

// 判断是否为整数Number.isInteger(xx)

let b=123.1;

console.log(Number.isInteger(b)); //false

// 整数转换Number.parseInt(xxx)和浮点型转换Number.parseFloat(xxx)

let c='9.18';

console.log(Number.parseInt(c));

console.log(Number.parseFloat(c));

// 取值范围，如下面2的53次方

let d = Math.pow(2,53)-1;

console.log(d); //9007199254740991

// 最大安全整数

console.log(Number.MAX\_SAFE\_INTEGER);

// 最小安全整数

console.log(Number.MIN\_SAFE\_INTEGER);

// 安全整数判断isSafeInteger( )

let e= Math.pow(2,53)-1;

console.log(Number.isSafeInteger(e));//true