**Symbol 类型、...扩展运算符、()=>{}**

**Symbol类型**

1，标识一个独一无二的值。 作用：防止属性名冲突

ES5类型：undefined、null、布尔值（Boolean）、字符串（String）、数值（Number）、对象（Object）。

第7种类型symbol （属于基本类型）

2，Symbol.prototype.description

Symbol(‘desc’)

const sym = Symbol('foo');

String(sym) // "Symbol(foo)"

sym.toString() // "Symbol(foo)"

const sym = Symbol('foo');

sym.description // "foo"

3, Symbol 作为属性名

let mySymbol = Symbol();

// 第一种写法

let a = {};

a[mySymbol] = 'Hello!';

// 第二种写法

let a = {

[mySymbol]: 'Hello!'

};

// 第三种写法

let a = {};

Object.defineProperty(a, mySymbol, { value: 'Hello!' });

// 以上写法都得到同样结果

a[mySymbol] // "Hello!"

Symbol 值作为对象属性名时，不能用点运算符

const mySymbol = Symbol();

const a = {};

a.mySymbol = 'Hello!';

a[mySymbol] // undefined

a['mySymbol'] // "Hello!"

5.属性名的遍历

Symbol 作为属性名，该属性不会出现在for...in、for...of循环中，也不会被Object.keys()、Object.getOwnPropertyNames()、JSON.stringify()返回。

但是，它也不是私有属性，有一个Object.getOwnPropertySymbols方法，可以获取指定对象的所有 Symbol 属性名。

const obj = {};

let foo = Symbol("foo");

Object.defineProperty(obj, foo, {

value: "foobar",

});

for (let i in obj) {

console.log(i); // 无输出

}

Object.getOwnPropertyNames(obj)

// []

Object.getOwnPropertySymbols(obj)

另一个新的 API，Reflect.ownKeys方法可以返回所有类型的键名，包括常规键名和 Symbol 键名。

let obj = {

[Symbol('my\_key')]: 1,

enum: 2,

nonEnum: 3

};

Reflect.ownKeys(obj)

// ["enum", "nonEnum", Symbol(my\_key)]

由于以 Symbol 值作为名称的属性，不会被常规方法遍历得到。我们可以利用这个特性，为对象定义一些非私有的、但又希望只用于内部的方法。

let size = Symbol('size');

class Collection {

constructor() {

this[size] = 0;

}

add(item) {

this[this[size]] = item;

this[size]++;

}

static sizeOf(instance) {

return instance[size];

}

}

let x = new Collection();

Collection.sizeOf(x) // 0

x.add('foo');

Collection.sizeOf(x) // 1

Object.keys(x) // ['0']

Object.getOwnPropertyNames(x) // ['0']

Object.getOwnPropertySymbols(x) // [Symbol(size)]

6, Symbol.for() Symbol.keyFor()

获取相同的symbol

7，js singleton模式下属性覆盖解决

8，内置的Symbol值

Symbol.hasInstance 对象的instanceof

Symbol.isConcatSpreadable 数组concat 时是否展开

Symbol.species 对象衍生（数组，promise）

Symbol.match string match 方法调用

Symbol.replace string replace 方法调用

Symbol.search

Symbol.split

Symbol.iterator …扩展运算符调用改方法

Symbol.toPrimitive 对象转换

Symbol.toStringTag

Symbol.unscopables with w3c上不推荐使用with语法

**...扩展运算符**