

介绍

自ECMAScript 2015起，`symbol`成为了一种新的原生类型，就像`number`和`string`一样。

`symbol`类型的值是通过`Symbol`构造函数创建的。

```
let sym1 = Symbol();

let sym2 = Symbol("key"); // 可选的字符串key
```

`Symbols`是不可改变且唯一的。

```
let sym2 = Symbol("key");
let sym3 = Symbol("key");

sym2 === sym3; // false, symbols是唯一的
```

像字符串一样，`symbols`也可以被用做对象属性的键。

```
let sym = Symbol();

let obj = {
  [sym]: "value"
};

console.log(obj[sym]); // "value"
```

`Symbols`也可以与计算出的属性名声明相结合来声明对象的属性和类成员。

```
const getClassNameSymbol = Symbol();

class C {
  [getClassNameSymbol]() {
    return "C";
  }
}

let c = new C();
let className = c[getClassNameSymbol](); // "C"
```

众所周知的Symbols

除了用户定义的`symbols`，还有一些已经众所周知的内置`symbols`。
内置`symbols`用来表示语言内部的行为。

以下为这些`symbols`的列表：

`Symbol.hasInstance`

方法，会被`instanceof`运算符调用。构造器对象用来识别一个对象是否是其实例。

Symbol.isConcatSpreadable

布尔值，表示当在一个对象上调用`Array.prototype.concat`时，这个对象的数组元素是否可展开。

Symbol.iterator

方法，被`for-of`语句调用。返回对象的默认迭代器。

Symbol.match

方法，被`String.prototype.match`调用。正则表达式用来匹配字符串。

Symbol.replace

方法，被`String.prototype.replace`调用。正则表达式用来替换字符串中匹配的子串。

Symbol.search

方法，被`String.prototype.search`调用。正则表达式返回被匹配部分在字符串中的索引。

Symbol.species

函数值，为一个构造函数。用来创建派生对象。

Symbol.split

方法，被`String.prototype.split`调用。正则表达式来用分割字符串。

Symbol.toPrimitive

方法，被`ToPrimitive`抽象操作调用。把对象转换为相应的原始值。

Symbol.toStringTag

方法，被内置方法`Object.prototype.toString`调用。返回创建对象时默认的字符串描述。

Symbol.unscopables

对象，它自己拥有的属性会被`with`作用域排除在外。