for...of语句在可迭代对象（包括 Array，Map，Set，String，TypedArray，arguments 对象等等）上创建一个迭代循环，调用自定义迭代钩子，并为每个不同属性的值执行语句。

对于Set，TypedArray，Map必须用for of才能遍历自身的对象和属性。

Set自身方法Set.prototype.forEach(func)也可以遍历Set自身。

Array和String类数组，可用数组的遍历方式。

***语法：***

for (variable of iterable) {

//statements

}

variable

在每次迭代中，将不同属性的值分配给变量。

iterable

可枚举其枚举属性的对象。

***迭代生成器***

你还可以迭代一个生成器：

function\* fibonacci() { // 一个生成器函数

let [prev, curr] = [0, 1];

for (;;) { // while (true) {

[prev, curr] = [curr, prev + curr];

yield curr;

}

}

for (let n of fibonacci()) {

console.log(n);

// 当n大于1000时跳出循环

if (n >= 1000)

break;

}

***不要重用生成器***

生成器不应该重用，即使for...of循环的提前终止，例如通过break关键字。在退出循环后，生成器关闭，并尝试再次迭代，不会产生任何进一步的结果。

var gen = (function \*(){

yield 1;

yield 2;

yield 3;

})();

for (let o of gen) {

console.log(o);

break;//关闭生成器

}

//生成器不应该重用，以下没有意义！

for (let o of gen) {

console.log(o);

}

***for...of与for...in的区别***

无论是for...in还是for...of语句都是迭代一些东西。它们之间的主要区别在于它们的迭代方式。

for...in 语句以原始插入顺序迭代对象的可枚举属性。

for...of 语句遍历可迭代对象定义要迭代的数据。

以下示例显示了与Array一起使用时，for...of循环和for...in循环之间的区别。

Object.prototype.objCustom = function() {};

Array.prototype.arrCustom = function() {};

let iterable = [3, 5, 7];

iterable.foo = 'hello';

for (let i in iterable) {

console.log(i); // logs 0, 1, 2, "foo", "arrCustom", "objCustom"

}

for (let i in iterable) {

if (iterable.hasOwnProperty(i)) {

console.log(i); // logs 0, 1, 2, "foo"

}

}

for (let i of iterable) {

console.log(i); // logs 3, 5, 7

}

Object.prototype.objCustom = function() {};

Array.prototype.arrCustom = function() {};

let iterable = [3, 5, 7];

iterable.foo = 'hello';

每个对象将继承objCustom属性，并且作为Array的每个对象将继承arrCustom属性，因为将这些属性添加到Object.prototype和Array.prototype。由于继承和原型链，对象iterable继承属性objCustom和arrCustom。

for (let i in iterable) {

console.log(i); // logs 0, 1, 2, "foo", "arrCustom", "objCustom"

}

此循环仅以原始插入顺序记录iterable 对象的可枚举属性。它不记录数组元素3, 5, 7 或hello，因为这些不是枚举属性。但是它记录了数组索引以及arrCustom和objCustom。如果你不知道为什么这些属性被迭代，array iteration and for...in中有更多解释。

for (let i in iterable) {

if (iterable.hasOwnProperty(i)) {

console.log(i); // logs 0, 1, 2, "foo"

}

}

这个循环类似于第一个，但是它使用hasOwnProperty() 来检查，如果找到的枚举属性是对象自己的（不是继承的）。如果是，该属性被记录。记录的属性是0, 1, 2和foo，因为它们是自身的属性（不是继承的）。属性arrCustom和objCustom不会被记录，因为它们是继承的。

for (let i of iterable) {

console.log(i); // logs 3, 5, 7

}

该循环迭代并记录iterable作为可迭代对象定义的迭代值，这些是数组元素 3, 5, 7，而不是任何对象的属性。