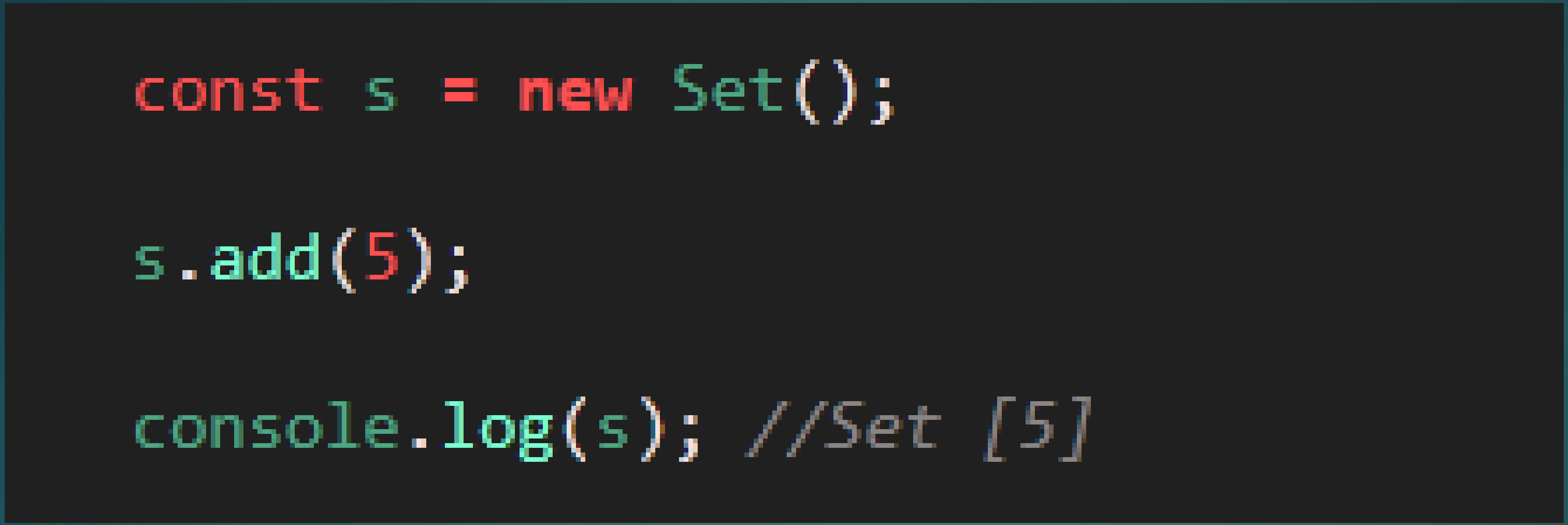
Set 和 扩展运算符

Set

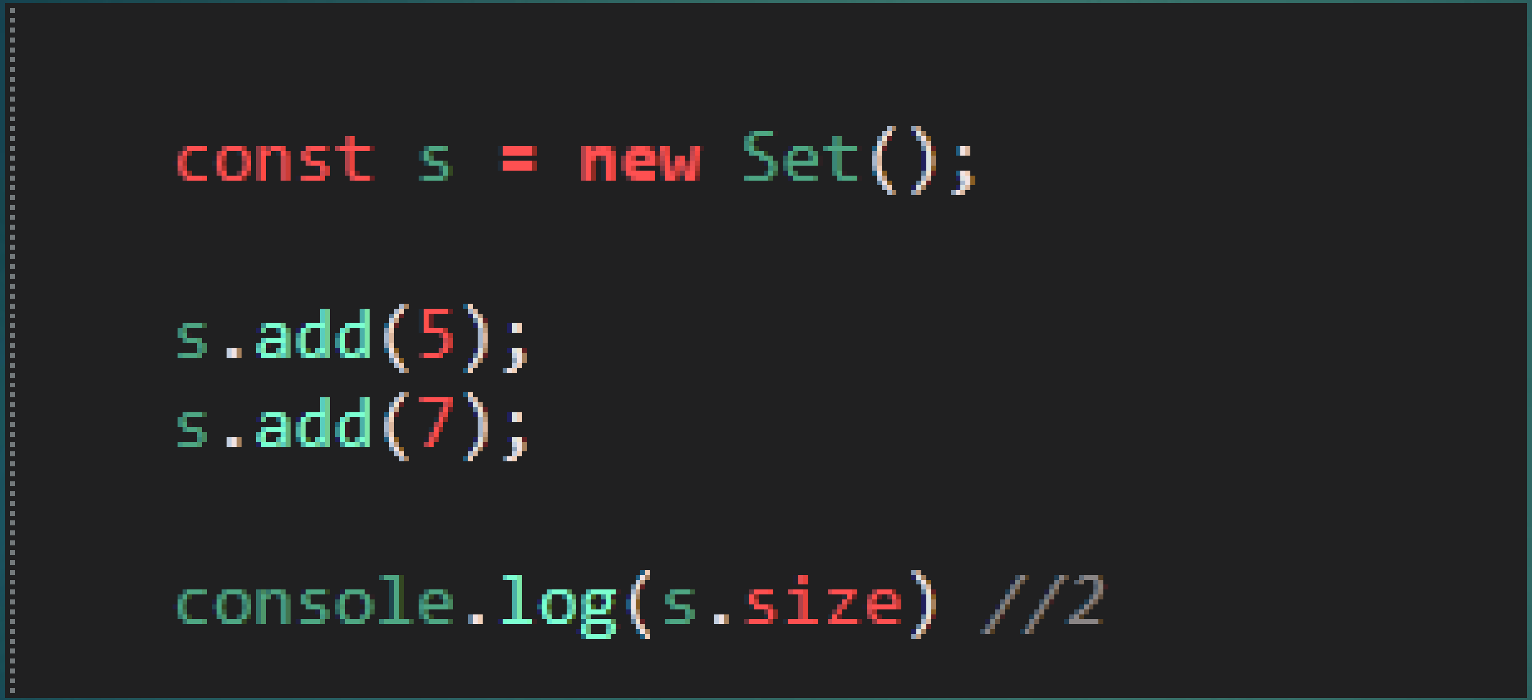
ES6 提供了新的数据结构 SET。它类似于数组，但是成员的值都是唯一的，没有重复的值。

Set基本用法



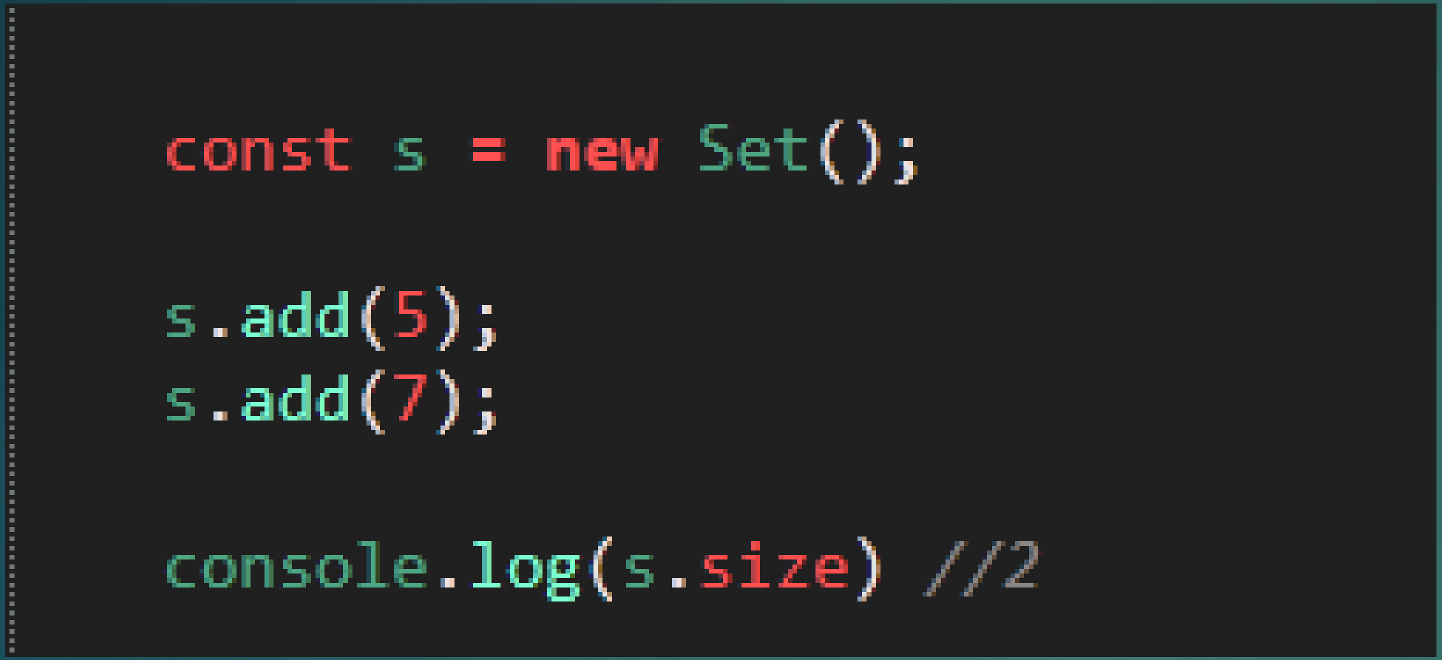
```
const s = new Set();

s.add(5);

console.log(s); //Set [5]
```

Set属性　size



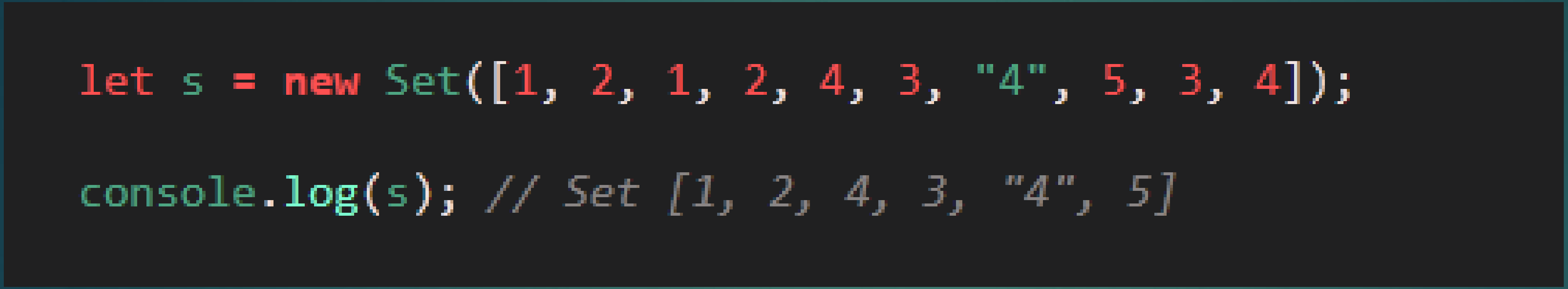
```
const s = new Set();

s.add(5);
s.add(7);

console.log(s.size) //2
```

Set方法　add



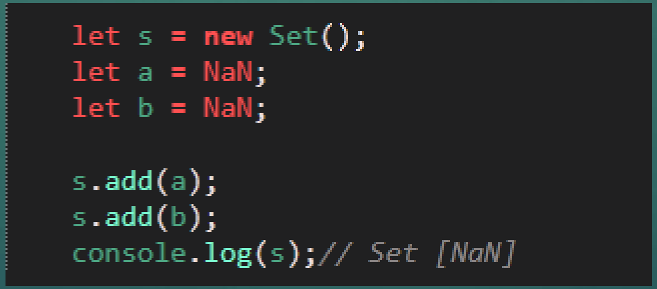
```
const s = new Set();

s.add(5);
s.add(7);

console.log(s.size) //2
```

add方法说明



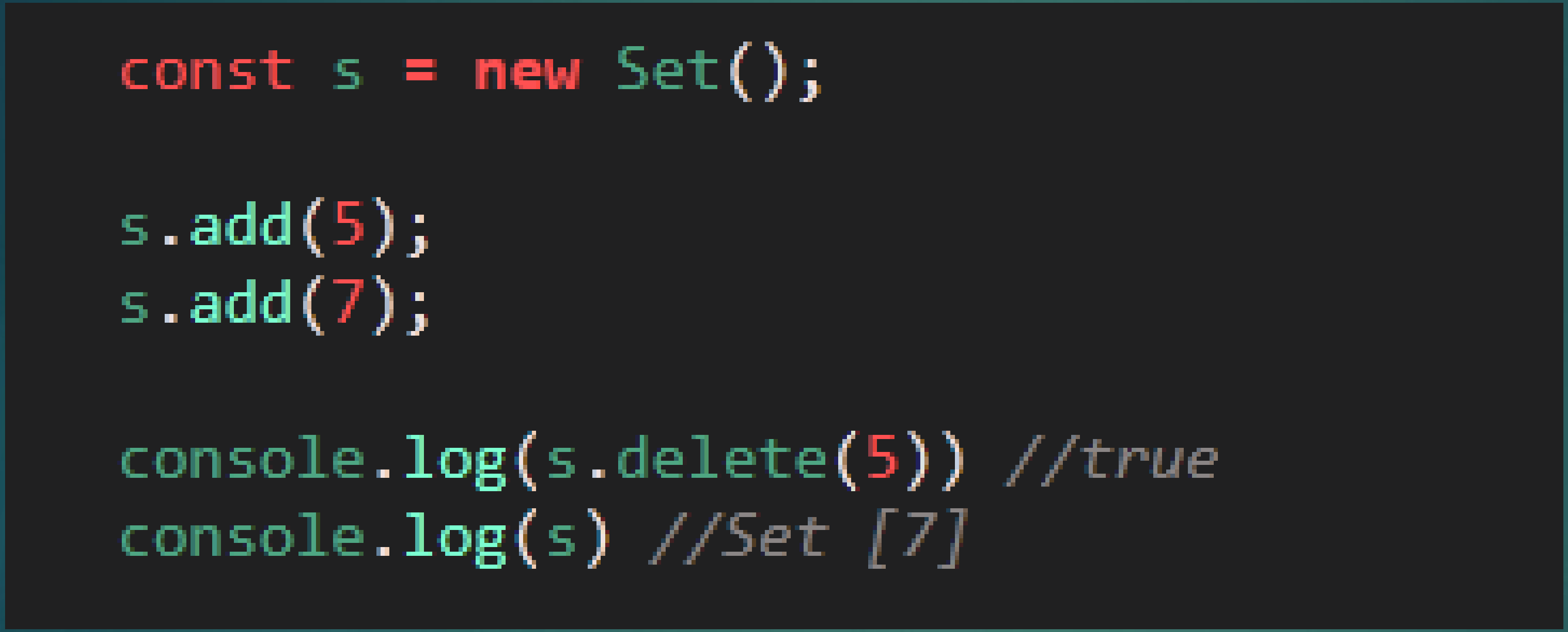
```
let s = new Set([1, 2, 1, 2, 4, 3, "4", 5, 3, 4]);

console.log(s); // Set [1, 2, 4, 3, "4", 5]
```

1. Set 结构不会添加重复的值;
2. 向 Set 加入值的时候，不会发生类型转换;
3. Set 内部判断两个值是否不同 ===



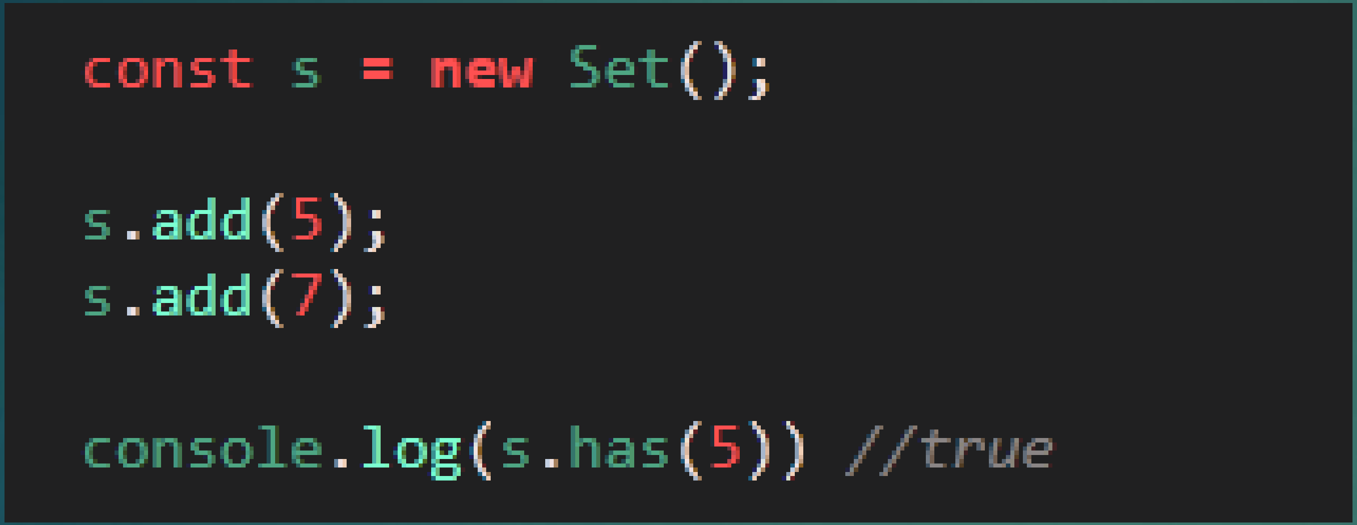
```
let s = new Set();
let a = NaN;
let b = NaN;

s.add(a);
s.add(b);
console.log(s);// Set [NaN]
```

Set方法　delete



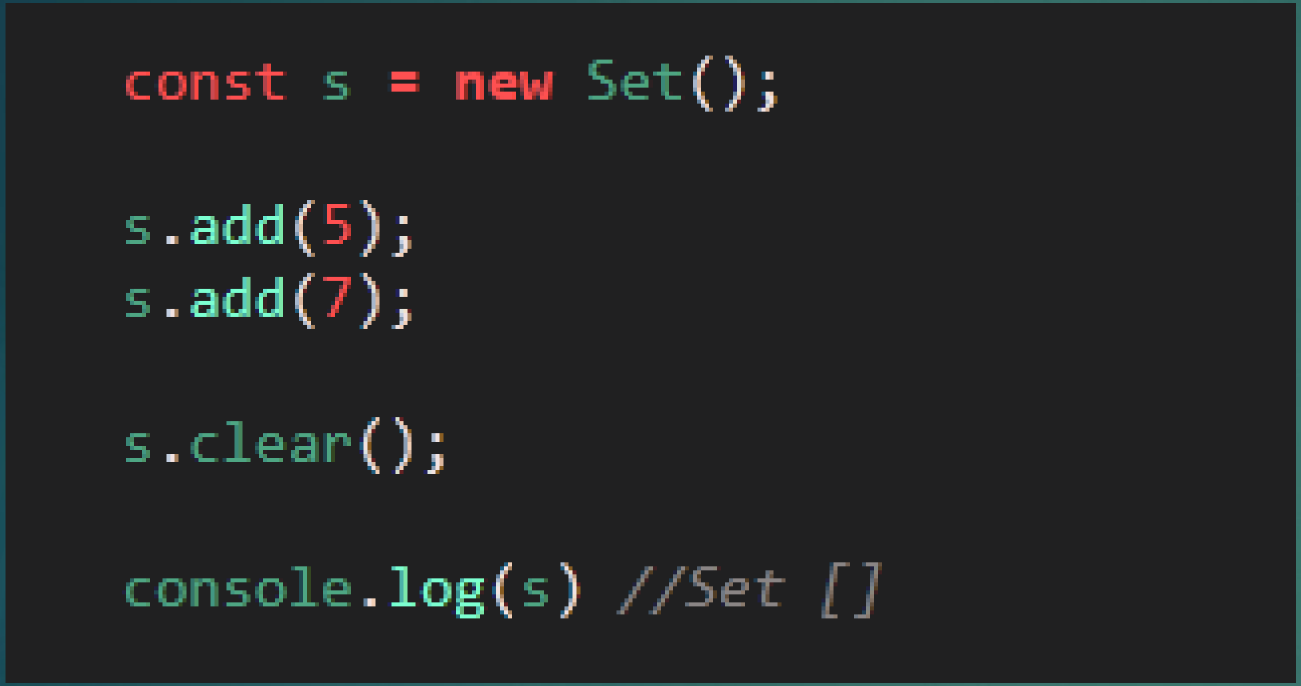
```
const s = new Set();

s.add(5);
s.add(7);

console.log(s.delete(5)) //true
console.log(s) //Set [7]
```

Set方法　has



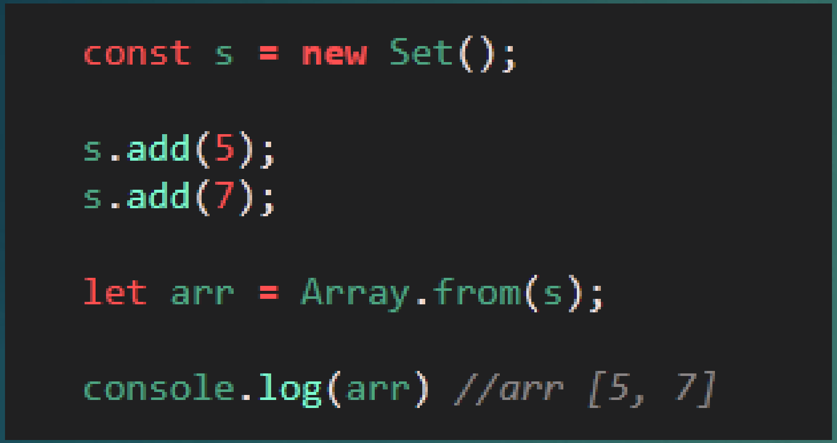
```
const s = new Set();

s.add(5);
s.add(7);

console.log(s.has(5)) //true
```

Set方法　clear

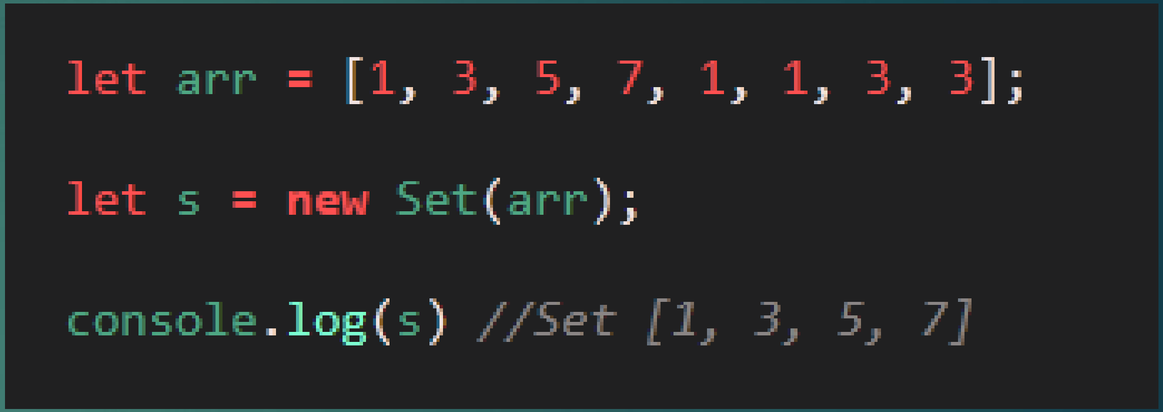


```
const s = new Set();

s.add(5);
s.add(7);

s.clear();

console.log(s) //Set []
```

Set 与 arr 转换

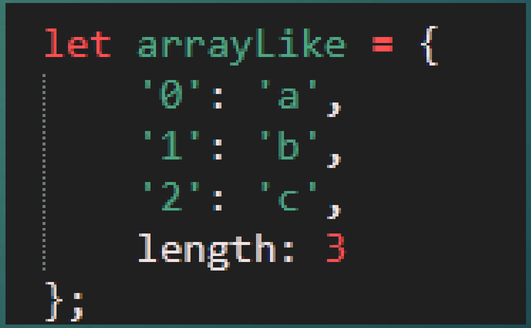


```
const s = new Set();

s.add(5);
s.add(7);

let arr = Array.from(s);

console.log(arr) //arr [5, 7]
```



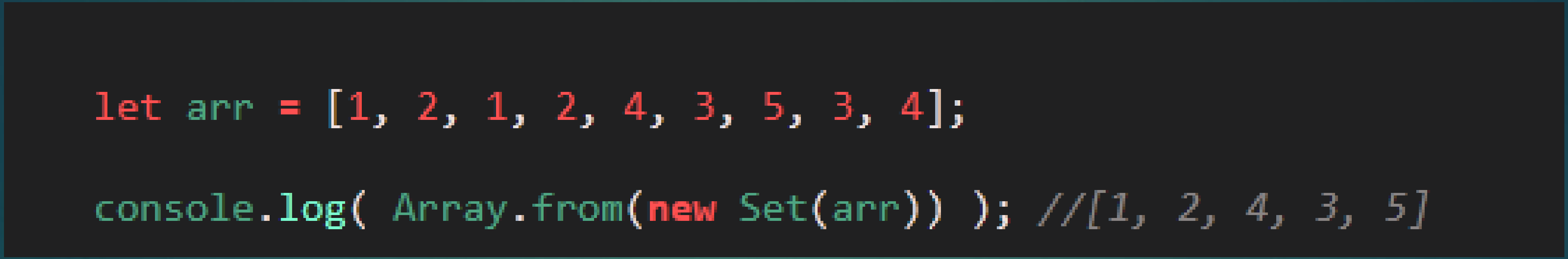
```
let arr = [1, 3, 5, 7, 1, 1, 3, 3];

let s = new Set(arr);

console.log(s) //Set [1, 3, 5, 7]
```

Array.from方法用于将两类对象转为真正的数组：类似数组的对象（array-like object）和可遍历（iterable）的对象（包括 ES6 新增的数据结构 Set 和 Map）。



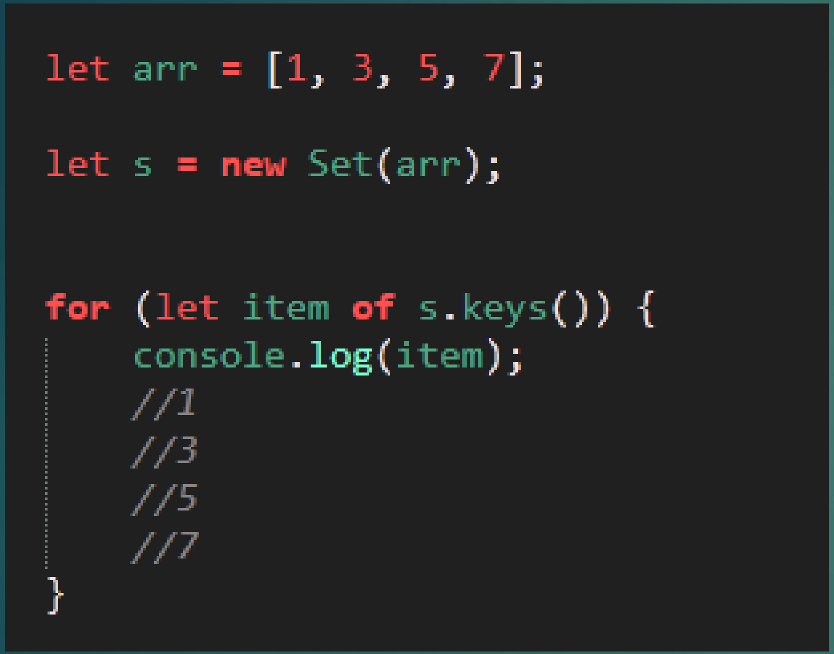
```
let arrayLike = {
    '0': 'a',
    '1': 'b',
    '2': 'c',
    length: 3
};
```

数组去重



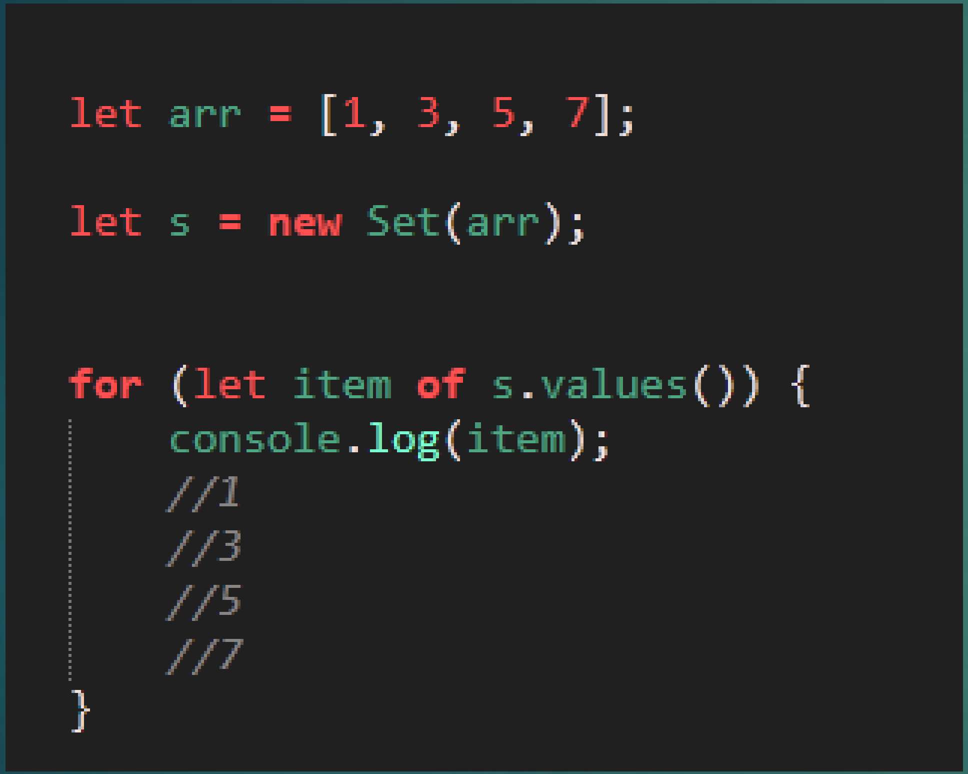
```
let arr = [1, 2, 1, 2, 4, 3, 5, 3, 4];

console.log( Array.from(new Set(arr)) ); //[1, 2, 4, 3, 5]
```

Set 遍历方法 keys()　返回键名



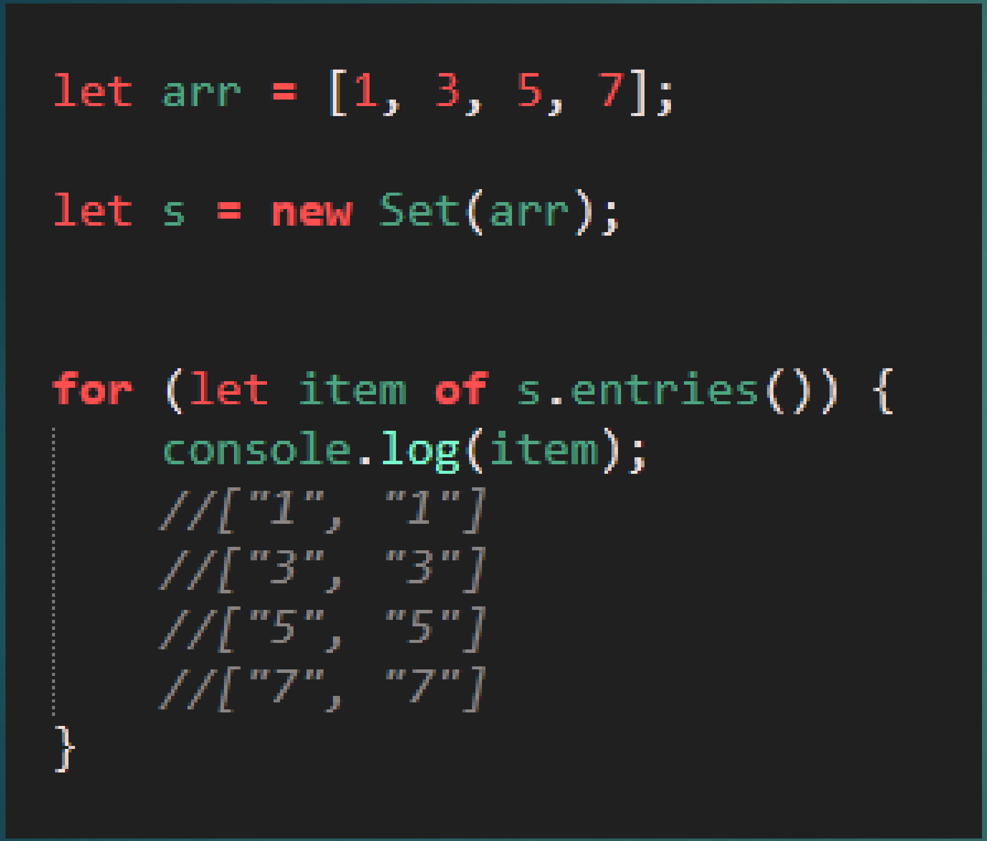
```
let arr = [1, 3, 5, 7];

let s = new Set(arr);


for (let item of s.keys()) {
    console.log(item);
    //1
    //3
    //5
    //7
}
```

Set 遍历方法 values()　返回键值



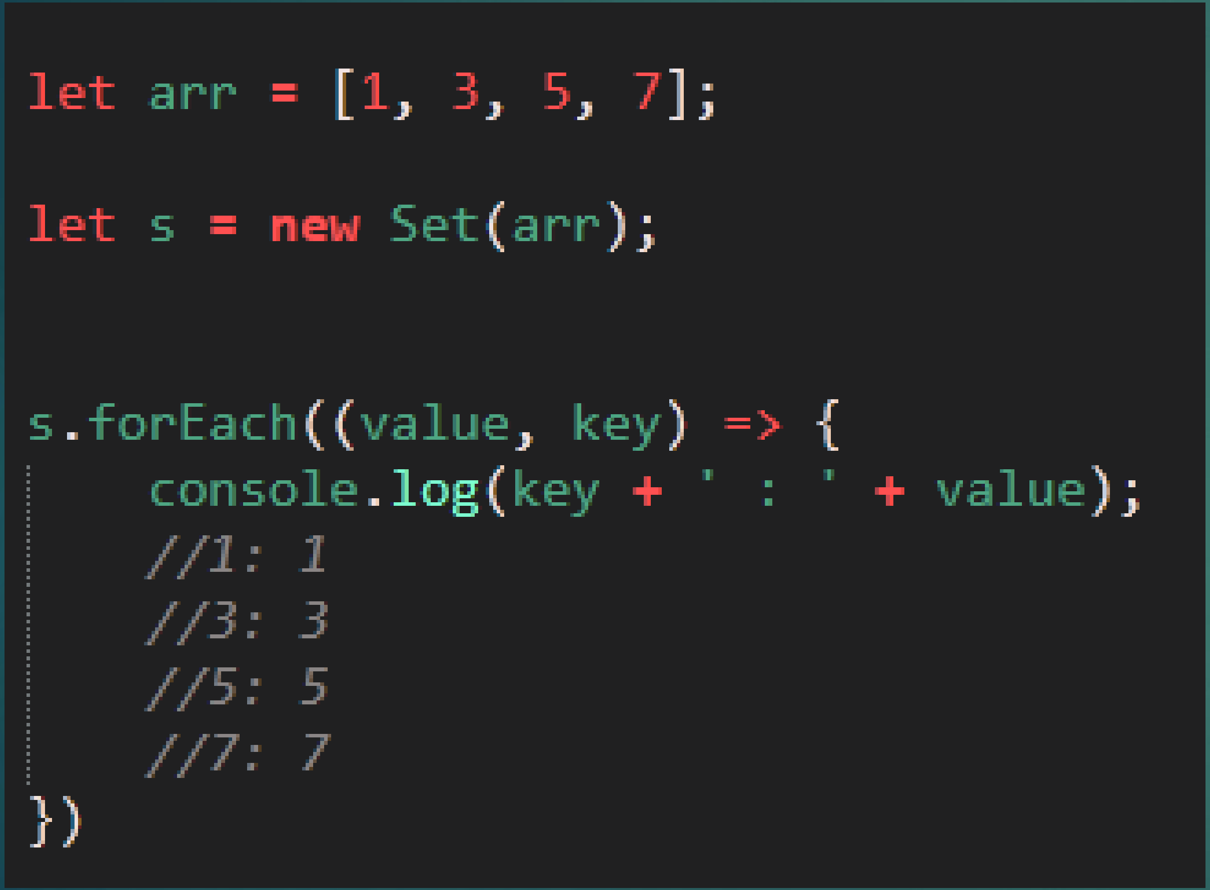
```
let arr = [1, 3, 5, 7];

let s = new Set(arr);


for (let item of s.values()) {
    console.log(item);
    //1
    //3
    //5
    //7
}
```

Set 遍历方法 entries()　返回键值对



```
let arr = [1, 3, 5, 7];

let s = new Set(arr);


for (let item of s.entries()) {
    console.log(item);
    //["1", "1"]
    //["3", "3"]
    //["5", "5"]
    //["7", "7"]
}
```

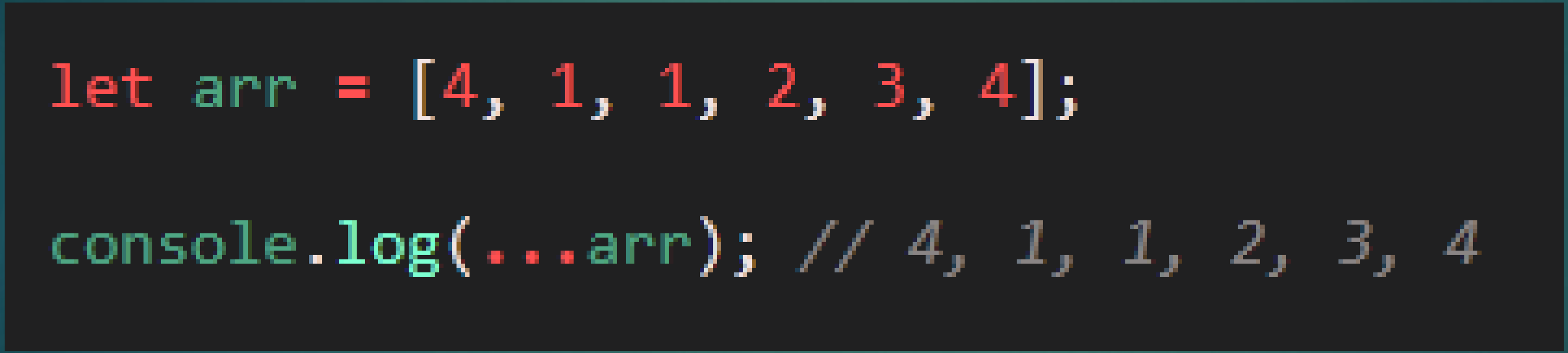
Set 遍历方法 forEach()



```
let arr = [1, 3, 5, 7];

let s = new Set(arr);


s.forEach((value, key) => {
    console.log(key + ' : ' + value);
    //1: 1
    //3: 3
    //5: 5
    //7: 7
})
```

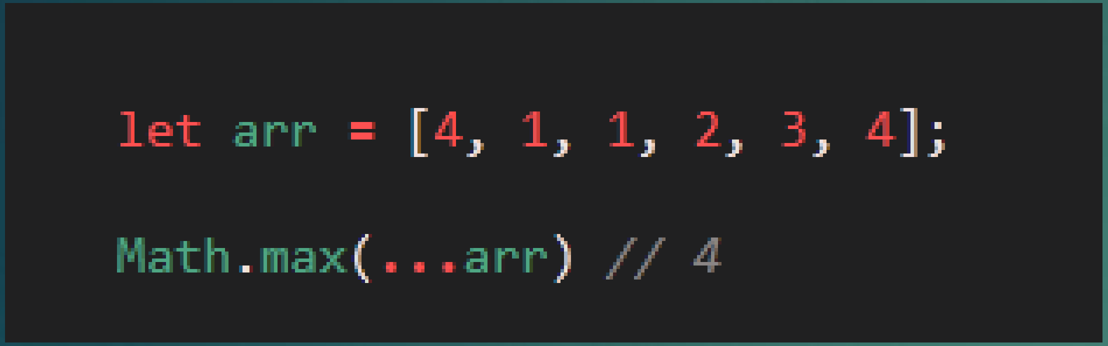
扩展运算符 ...

将一个数组转为用逗号分隔的参数序列。

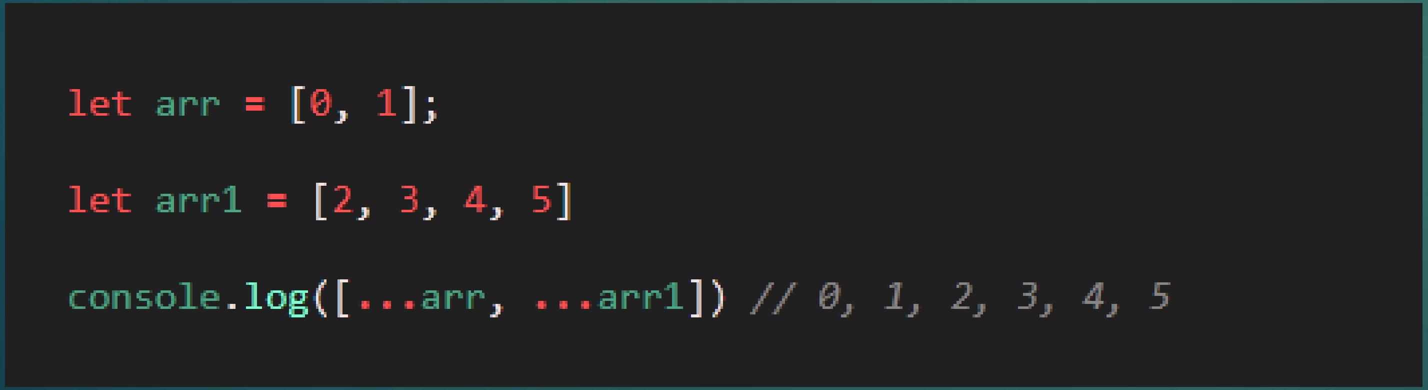


```
let arr = [4, 1, 1, 2, 3, 4];

console.log(...arr); // 4, 1, 1, 2, 3, 4
```

扩展运算符 ...

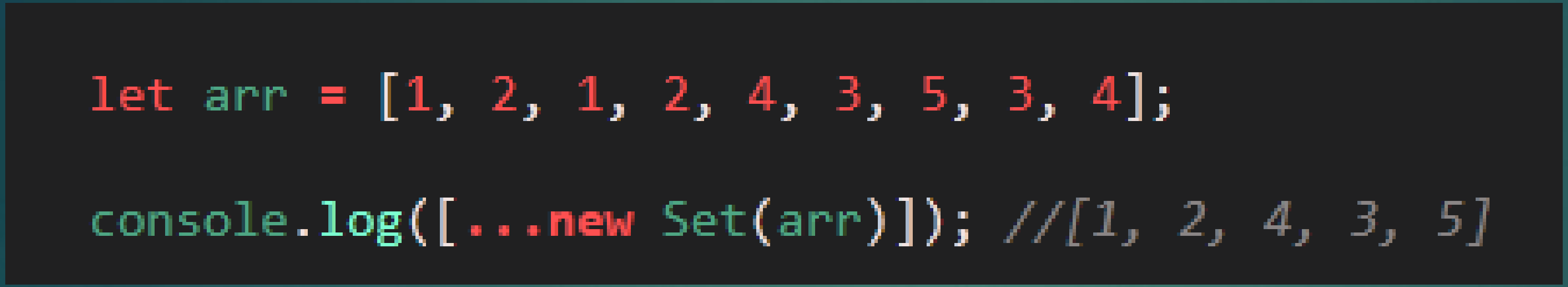


```
let arr = [4, 1, 1, 2, 3, 4];

Math.max(...arr) // 4
```



```
let arr = [0, 1];

let arr1 = [2, 3, 4, 5]

console.log([...arr, ...arr1]) // 0, 1, 2, 3, 4, 5
```

数组去重



```
let arr = [1, 2, 1, 2, 4, 3, 5, 3, 4];

console.log([...new Set(arr)]); //[1, 2, 4, 3, 5]
```