对象新增方法

1. Object.is()

`Object.is()` 用来比较两个值是否严格相等，与严格比较运算符（`===`）的行为基本一致，只有两点不同：

> - `+0` 不等于 `-0`
> - `NaN` 等于自身

```
+0 === -0 //true
NaN === NaN // false

Object.is(+0, -0) // false
Object.is(NaN, NaN) // true
```

2. Object.assign()

`Object.assign()` 方法用于对象的合并，第一个参数是目标对象，后面的参数都是源对象。将源对象（`source`）的所有可枚举属性，复制到目标对象（`target`）：

```
const target = { a: 1 };

const source1 = { b: 2 };
const source2 = { c: 3 };

Object.assign(target, source1, source2);
target // {a:1, b:2, c:3}
```

如果目标对象与源对象有同名属性，或多个源对象有同名属性，则后面的属性会覆盖前面的属性。

```
const target = { a: 1, b: 1 };

const source1 = { b: 2, c: 2 };
const source2 = { c: 3 };

Object.assign(target, source1, source2);
target // {a:1, b:2, c:3}
```

如果只有一个参数，Object.assign() 会直接返回该参数。

```
const obj = {a: 1};
Object.assign(obj) === obj // true
```

Object.assign()拷贝的属性是有限制的，只拷贝源对象的自身属性（不拷贝继承属性），也不拷贝不可枚举的属性（enumerable: false）。

```
Object.assign({b: 'c'},
  Object.defineProperty({}, 'invisible', {
    enumerable: false,
    value: 'hello'
  })
)
// { b: 'c' }
```

上面代码中，`Object.assign()` 要拷贝的对象只有一个不可枚举属性invisible，这个属性并没有被拷贝进去。

属性名为 Symbol 值的属性，也会被Object.assign()拷贝。

```
Object.assign({ a: 'b' }, { [Symbol('c')]: 'd' })
// { a: 'b', Symbol(c): 'd' }
```