14.对象的新增方法

学习要点：

1. 新增方法

本节课我们来开始学习 ES6 新增的对象的扩展和改进功能。

一. 新增方法

1. ES6 提供了 Object.is()方法来解决“===”中一些恒等中的缺点；

```
console.log(Object.is(100 ,'100')); //false，必须恒等
console.log(Object.is({}, {}));     //false，指向不同
console.log(Object.is(+0, -0));     //false
console.log(Object.is(NaN, NaN));   //true
console.log(NaN === NaN);           //false
console.log(+0 === -0);             //true
```

2. ES6 提供了 Object.assign()方法可以合并指定对象至目标对象内部；

```
//...传参，合并所有对象，返回给参数一的目标对象
console.log(Object.assign(obj1, obj2, obj3));
//第一个对象得到合并后的结果；
console.log(obj1);
```

PS: (1).如果属性有相同，后面的源对象内容会覆盖之前的属性值；
(2).如果直接传非对象内容，会转换为对象；
(3).如果传入的是 undefined 和 null 会报错；

3. ES6 提供了 Object.getPrototypeOf()和 Object.setPrototypeof()方法；

```
let obj = {
    fn() {
        return 'fn';
    }
};

let obj2 = {
    fn() {
        return 'fn2';
    }
};

//以 obj 对象为原型
let f = Object.create(obj);
```

```
//检测是 obj 是否 f 的原型对象
console.log(Object.getPrototypeOf(f) === obj);
//输出原型对象的 fn
console.log(f.fn());

//设置 f 的原型对象为 obj2
Object.setPrototypeOf(f, obj2);
console.log(Object.getPrototypeOf(f) === obj2);
console.log(f.fn());
```

4. ES6 提供了 super 关键字，用于原型中方法的继承功能；

```
let obj = {
    fn() {
        return 'fn';
    }
};

let f = {
    fn() {
        return super.fn() + ', extend!';
    }
};

//设置 obj 是 f 的原型
Object.setPrototypeOf(f, obj);
console.log(f.fn());

//可以再设置以 f 为原型
let h = Object.create(f);
console.log(h.fn());
```