类，原型，构造函数，继承

类 Class

ES6引入了 **Class**（类）这个概念。它的绝大部分功能，ES5都可以做到。新的class写法只是让对象原型的写法更加清晰、更像面向对象编程的语法。

Class不存在变量提升。

```
//定义类
class Point {
    constructor(x, y) {       //定义类的构造函数方法
//constructor 方法是类的默认方法。一个类必须有constructor方法。如果没有显式定义，一个空的constructor方法会被默认添加。
//constructor方法默认返回实例对象（即this），也可以指定返回另外一个对象。
        this.x = x;       //实例属性
        this.y = y;
    }

    toString() {        //定义类的原型方法
//定义类的方法时，前面不需要加上function这个关键字，直接把函数表达式放进去。
//类的所有方法都定义在类的prototype属性上面。
        return '(' + this.x + ', ' + this.y + ')';
    }
}

//生成类的实例
var point = new Point(2, 3);        //通过new命令生成对象实例时，自动调用constructor方法。

//通过实例的__proto__属性改写原型
var p1 = new Point(2,3);
var p2 = new Point(3,2);          //类的所有实例共享一个原型对象。

p1.__proto__.printName = function () { return 'Oops' };
//通过实例的__proto__属性为原型对象添加方法

p1.printName()    // "Oops"
p2.printName()    // "Oops"
```

继承

Class之间可以通过**extends**关键字实现继承。这比ES5的通过修改原型链实现继承，要清晰和方便很多。

```
//继承
class ColorPoint extends Point {
    constructor(x, y, color) {
        super(x, y);    // 调用父类的constructor(x, y)
//super关键字表示父类的构造函数，用来新建父类的this对象。子类必须在constructor方法中调用super方法，否则新建实例时会报错。因为子类没有自己的this对象，而是继承父类的this对象，然后对其进行加工。如果不调用super方法，子类就得不到this对象。
```

```
        this.color = color;
    }

    toString() {
        return this.color + ' ' + super.toString();    // 调用父类的toString()
    }
}
```

Class的静态方法

在一个方法前，加上static关键字，就表示该方法不会被实例继承，而是直接通过类来调用。

父类的静态方法，可以被子类继承。