

## 第十五章 Class类

一样的在线教育，不一样的教学品质



# 目录 Contents

- ◆ Class基本使用
- ◆ Class原理分析
- ◆ constructor方法与实例对象
- ◆ Class的继承
- ◆ Class的getter和setter函数
- ◆ Class的静态方法
- ◆ Class的静态属性

## 小节导学

在JavaScript中传统对象的创建方式通过**构造方法**完成。

```
function Person(name, age) {  
    this.name=name;  
    this.age=age;  
}
```

**缺点：**

构造方法与传统的面向对象编程语言，例如 C#,Java等差别较大，不利于具有其他编程语言基础的学员学习；

**解决方法：**

ES6提供更接近传统面向对象编程语言的写法，引入了Class(类)来创建对象。

**通过class创建类，class可看做一个语法糖，其绝大部分功能，在ES5中皆可实现。**

基本定义：

```
class Person {  
    constructor(name, age) {  
        this.name = name;  
        this.age = age;  
    }  
    getShow() {  
        console.log(this.name, this.age);  
    }  
}  
let p = new Person('zs', 20);  
p.getShow();
```

**constructor()**：表示类的构造方法，是固定写法；new一个实例时，会执行该构造方法，并完成参数的传递。

**This**：表示类的实例对象；在构造函数中将传递过来的参数值赋给对应的属性。

**getShow()**：表示方法；

**注意**：类中定义方法时，前面不需要加function关键字，方法之间不需要添加分号进行分隔，否则会出错。

**new**：表示创建对象；会执行构造方法，完成传值。



# 目录 Contents

- ◆ Class基本使用
- ◆ Class原理分析
- ◆ constructor方法与实例对象
- ◆ Class的继承
- ◆ Class的getter和setter函数
- ◆ Class的静态方法
- ◆ Class的静态属性

示例：

将上节课所写案例转换为ES5，进行分析。

经过分析可得出如下结论：

- ◆ ES6的类完全可看作是构造函数的另一种写法；
- ◆ 类中定义的方法都定义在类的prototype属性上；



# 目录 Contents

- ◆ Class基本使用
- ◆ Class原理分析
- ◆ constructor方法与实例对象
- ◆ Class的继承
- ◆ Class的getter和setter函数
- ◆ Class的静态方法
- ◆ Class的静态属性

## ◆ constructor方法

constructor方法是类的默认方法；通过new 命令生成对象实例时自动调用该方法，**类中必须有constructor方法；若没有显示的定义，一个空的constructor方法会被默认添加。**

## ◆ 实例对象

生成实例对象的方式是通过new命令完成的，与ES5一致；  
在类中定义的属性，若不是直接定义在this对象上，则都是定义在原型上。





# 目录 Contents

- ◆ Class基本使用
- ◆ Class原理分析
- ◆ constructor方法与实例对象
- ◆ Class的继承
- ◆ Class的getter和setter函数
- ◆ Class的静态方法
- ◆ Class的静态属性

## 基本用法：

Class 之间可**通过extends关键字实现继承**，比ES5通过修改原型链实现继承更加简单方便。

```
class Person {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
    getShow() {
        console.log(this.name, this.age);
    }
}

class Student extends Person {
    constructor(name, age, gender) {
        super(name, age);
        this.gender = gender;
    }
    getGender() {
        console.log(this.gender);
    }
}

let stu = new Student('zs', 19, '男');
stu.getShow();
stu.getGender();
```

使用Class继承时需注意的问题：

- ◆ 子类必须在constructor方法中调用super方法，否则无法创建实例对象。
- ◆ 在子类的构造函数中，只有先调用了super方法后，才可使用this关键字。

将继承的代码转换成ES5代码后，有一段非常重要的代码，如下所示：

```
subClass.prototype = Object.create(superClass && superClass.prototype, {
  constructor: {
    value: subClass,
    writable: true,
    configurable: true
  }
});
```

以上的代码等同于如下代码：

```
function F() {}
F.prototype = superClass.prototype
subClass.prototype = new F()
subClass.prototype.constructor = subClass
```

**结论如下：**

- ◆ 子类的 `__proto__` 属性，总是指向父类。
- ◆ 子类的 `prototype` 属性的 `__proto__` 属性总是指向父类的 `prototype` 属性。



# 目录 Contents

- ◆ Class基本使用
- ◆ Class原理分析
- ◆ constructor方法与实例对象
- ◆ Class的继承
- ◆ Class的getter和setter函数
- ◆ Class的静态方法
- ◆ Class的静态属性

# Class的getter和setter函数

基本用法：

与ES5一致，**Class内部可使用get和set关键字对某个属性设置存值函数和取值函数，拦截该属性的存取行为。**

```
class Student {  
    constructor() {  
  
    }  
    get age() {  
        return 18  
    }  
    set age(value) {  
        console.log(value);  
    }  
}  
let zs = new Student();  
console.log(zs.age);  
zs.age = 20;
```



# 目录 Contents

- ◆ Class基本使用
- ◆ Class原理分析
- ◆ constructor方法与实例对象
- ◆ Class的继承
- ◆ Class的getter和setter函数
- ◆ Class的静态方法
- ◆ Class的静态属性

# Class的静态方法

调用在类中定义的方法，可通过该类创建其对应的实例（对象名）；并通过该实例名称加上点运算符即可完成调用；若在方法前加上 'static' 关键字，表示该方法是“静态方法”，在**调用时不能通过实例的方式进行调用，只能通过类的方式进行调用。**

```
class Student {  
    constructor() {  
  
    }  
    static hello() {  
        console.log('hello');  
    }  
}  
let zs = new Student();  
Student.hello();  
zs.hello(); //调用出错
```





# 目录 Contents

- ◆ Class基本使用
- ◆ Class原理分析
- ◆ constructor方法与实例对象
- ◆ Class的继承
- ◆ Class的getter和setter函数
- ◆ Class的静态方法
- ◆ Class的静态属性

静态属性属于类，不用实例就能调用。

```
class Student {  
  
    }  
    Student.age = 20;  
    console.log(Student.age);  
}
```

以上为ES6唯一的写法，ES7中可采用如下写法：

```
class Student {  
    static age = 23;  
  
    }  
    console.log(Student.age);  
}
```



一样的在线教育，不一样的教学品质