

# JavaScript进阶 (ES6)

---Class

# 内容提纲

---

- **ES6 Class基本语法**
- **ES6 Class静态方法、静态属性**
- **ES6 Class的继承**



# ES6 Class语法

- ES5中使用构造函数定义并生成新的对象（模拟类-类继承）

- 与传统面向对象语言的差异比较大，不利于开发和维护复杂的应用程序

```
function Point(x,y){  
    this.x = x;  
    this.y = y;  
}  
Point.prototype.show = function(){  
    console.log("Point:",this.x,this.y);  
}  
var p = new Point(1,2);  
p.show();  
Point: 1 2
```

回顾：

p.constructor是什么？

p.\_\_proto\_\_是什么？

Point.prototype是什么？

Point.\_\_proto\_\_是什么？

Point.prototype.constructor是什么？

如何添加私有属性？

访问私有属性的方法是否可以定义在原型上？

# ES6 Class语法

## • ES6中引入了Class（类）作为对象的模板

- ES6中的class是一个语法糖（核心内容同ES5），仔细阅读理解下述语法并与ES5对比
- 与ES5不同的是类内定义的方法是**不可枚举的**

```
class Point{
  constructor(){
    this.x = 1;
    this.y = 2;
  }
  show(){
    console.log("Point:",this.x,this.y);
  }
}
let p = new Point();
p.show();
```

Point: 1 2

注意：**方法前不加function**，方法之间不用逗号分隔，如果没有写constructor方法，会添加一个**默认的constructor**

**思考：**show方法是定义在p对象身上了，还是定义在**p对象的原型**上了？

使用下述方法进行测试  
Object.getOwnPropertyNames

# ES6 Class语法

## • ES6中通过通过class实例化的对象的原型

- 与ES5一样，实例化出的对象的原型是共享的，下例中实例化的对象的原型是Point.prototype

```
class Point {  
  constructor(x, y) {  
    this.x = x;  
    this.y = y;  
  }  
  show() {  
    console.log("Point:",x,y);  
  }  
}  
  
var p1 = new Point(1,2);  
var p2 = new Point(1,2);  
p1.__proto__ === p2.__proto__;//true
```

```
class Point {  
  constructor(x, y) {  
    this.x = x;  
    this.y = y;  
  }  
}  
  
var p1 = new Point(2,3);  
var p2 = new Point(3,2);  
p1.__proto__.printName = function () {  
  console.log('Oops')  
};  
p1.printName() // "Oops"  
p2.printName() // "Oops"  
var p3 = new Point(4,2);  
p3.printName() // "Oops"
```

## ES6 Class语法补充

- class表达式形式（ES6支持class表达式形式）
- class的立即执行表达式
- 不存在class提升（养成良好代码习惯，使用前定义）
- Class的name属性（同构造函数的name属性）
- 在“类”的内部可以使用get和set关键字作为过滤
- ES6 为new命令引入了一个new.target属性

# 内容提纲

---

- ES6 Class基本语法
- ES6 Class静态方法、静态属性
- ES6 Class的继承



# ES6 Class静态方法、静态属性

## • 静态方法与实例方法

- 静态方法指的是 Class 本身的方法，而不是定义在实例对象上的方法
- 通过关键字 static 定义静态方法，静态方法中的this指向类本身

```
class Foo {  
    static classMethod() {  
        console.log(this); // Foo类本身  
        return 'hello';  
    }  
}  
Foo.classMethod(); // 'hello'  
var foo = new Foo();  
// foo.classMethod(); // TypeError
```



# ES6 Class静态方法、静态属性

## • 静态属性与实例属性

- 静态属性指的是 Class 本身的属性，而不是定义在实例对象上的属性
- ES6 规定Class 内部只有静态方法，没有静态属性
- 新的ES提案中包括了静态属性

```
class Foo {  
}  
Foo.prop = 1;  
Foo.prop; // 1
```

```
class Foo {  
  static prop = 1; //ES6暂不支持  
}
```



# 内容提纲

---

- ES6 Class基本语法
- ES6 Class静态方法、静态属性
- ES6 Class的继承



# ES6 Class的继承

- ES6中通过class实现继承的语法

```
class Point {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }
}
class ColorPoint extends Point {
  constructor(x, y, color) {
    super(x, y); // 调用父类的constructor(x, y)
    this.color = color;
  }
  show() {
    console.log(this.x, this.y, this.color);
  }
}
var cp = new ColorPoint(1, 2, 3);
cp.show();
```

通过`extends`关键字实现继承，比ES5的通过修改原型链实现继承，要清晰和方便

注意：`constructor`与`super`

参见实例demo06  
ES6 class与继承

# ES6 Class的继承

- class中的super (当作函数使用, 也可以当作对象使用)

- 当做函数时, 子类构造函数之中的`super()`, 代表调用父类的构造函数
- 当做对象时, 在普通方法中, 指向父类的原型对象; 在静态方法中, 指向父类

```
class A {  
  constructor() {  
    console.log(new.target.name);  
  }  
}
```

```
class B extends A {  
  constructor() {  
    super(); // super虽然代表了父类A的构造函数, 但是返回的是子类B的实例  
  }  
}
```

```
new A(); // A
```

```
new B(); // B
```

# ES6 Class的继承

- class中的super（当作函数使用，也可以当作对象使用）
  - 当做函数时，子类构造函数之中的super()，代表调用父类的构造函数
  - 当做对象时，在实例（原型）方法中，指向父类的prototype属性；在静态方法中，指向父类

```
class A {  
  p() {return 2;}  
}  
class B extends A {  
  constructor() {  
    super();  
    console.log(super.p()); // 2  super指向A.prototype  
  }  
  f(){  
    console.log(super.p()); // 2  super指向A.prototype  
  }  
}  
let b = new B();//2  
b.f();//2
```

参见实例demo07 super实例 Part2

The background of the slide is decorated with various abstract shapes in shades of green and yellow. These shapes, which include circles, ovals, and irregular blobs, are scattered across the top and right sides of the slide, creating a modern, organic feel.

# Thank You!



河北师范大学软件学院  
Software College of Hebei Normal University