面向对象类

类与实例

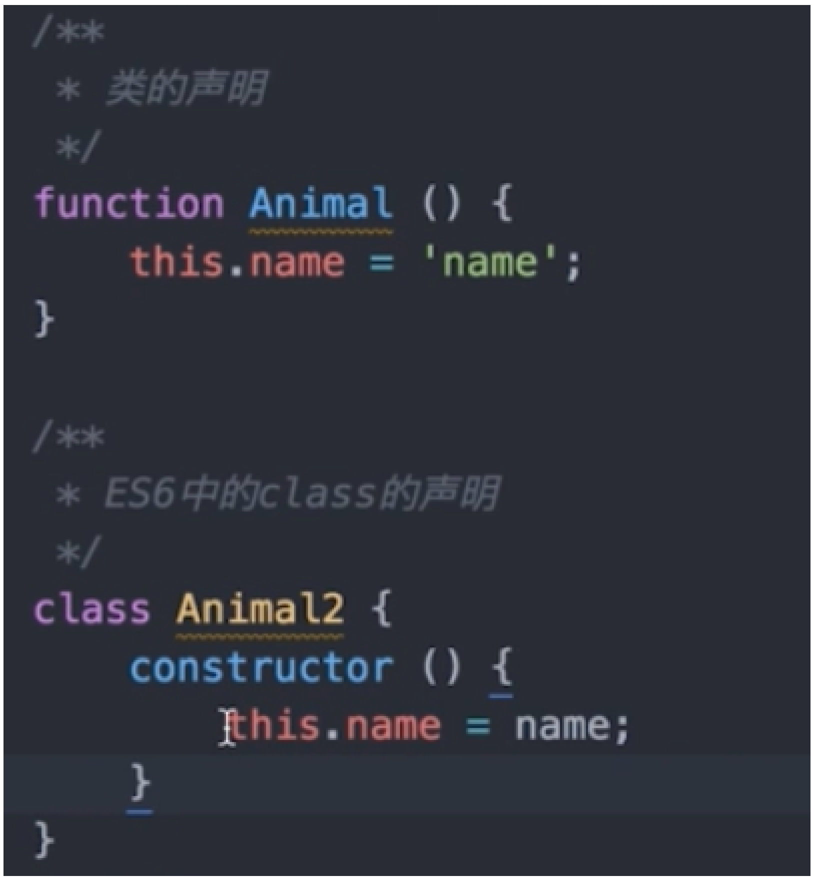
类的声明

生成实例

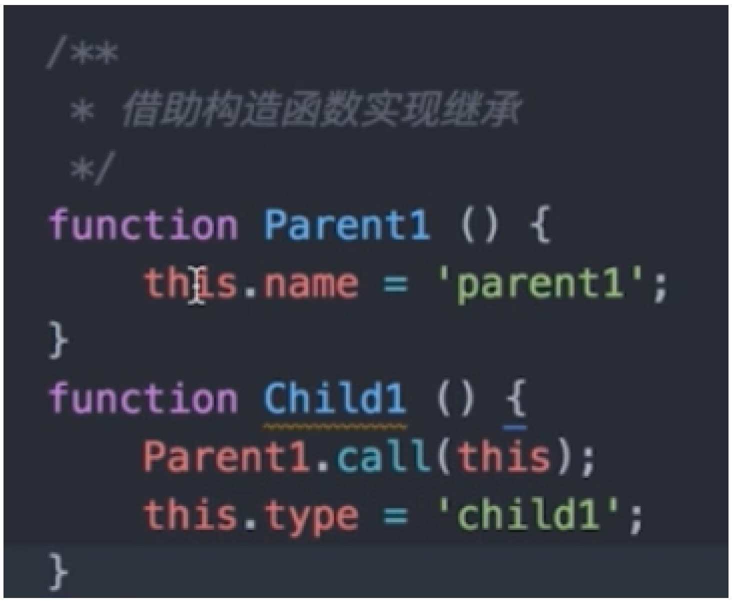
类与继承

如何实现继承

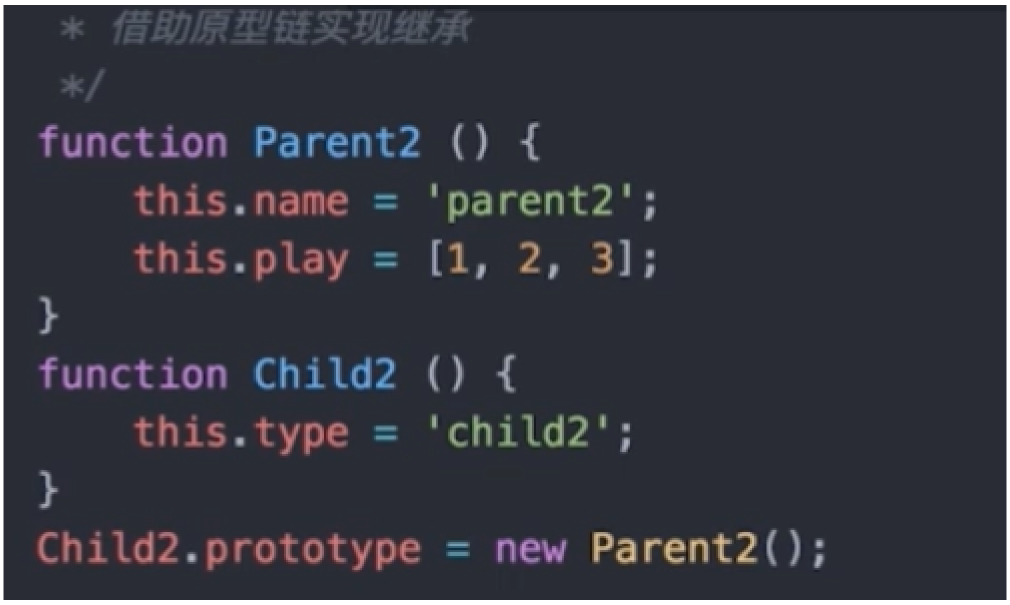
继承的几种方式



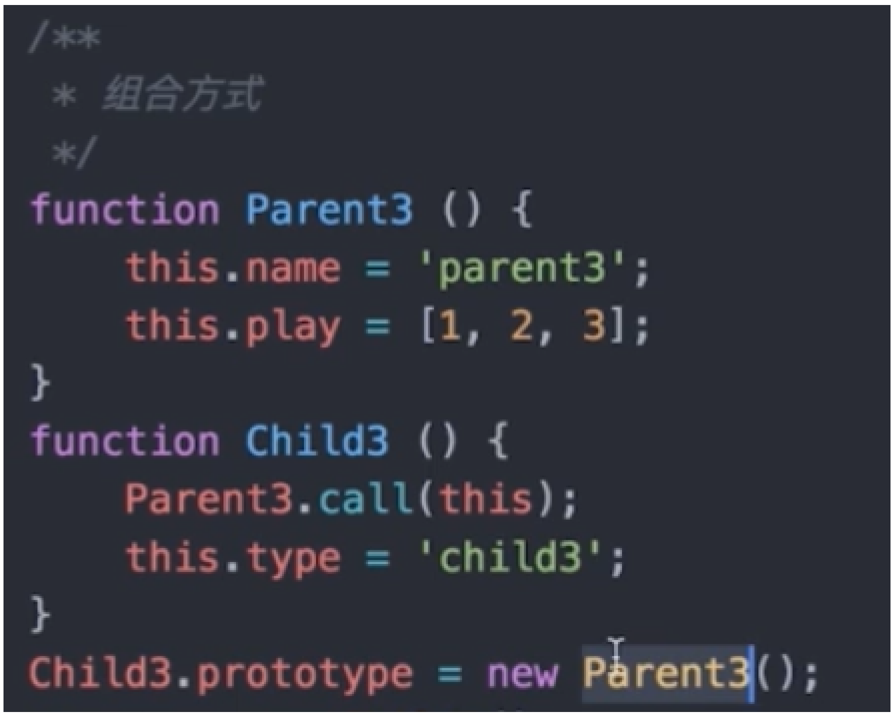
```
/**
 * 类的声明
 */
function Animal () {
    this.name = 'name';
}

/**
 * ES6中的class的声明
 */
class Animal2 {
    constructor () {
        this.name = name;
    }
}
```



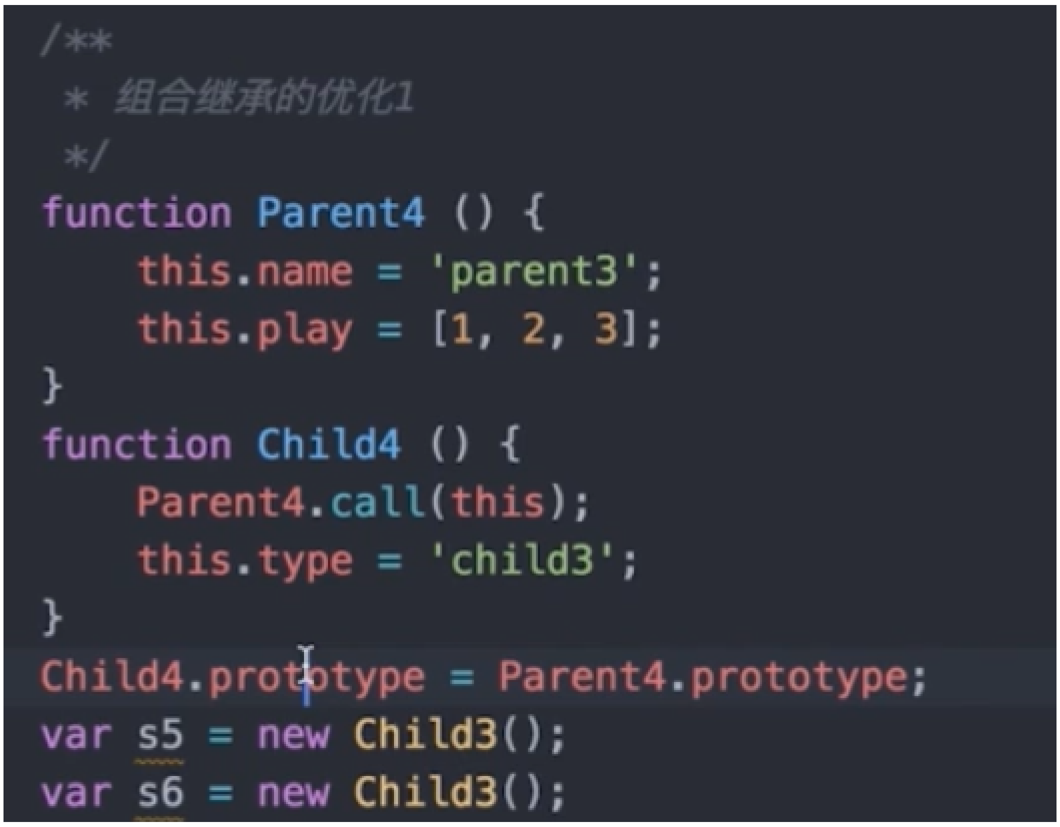
```
/**
 * 借助构造函数实现继承
 */
function Parent1 () {
    this.name = 'parent1';
}
function Child1 () {
    Parent1.call(this);
    this.type = 'child1';
}
```



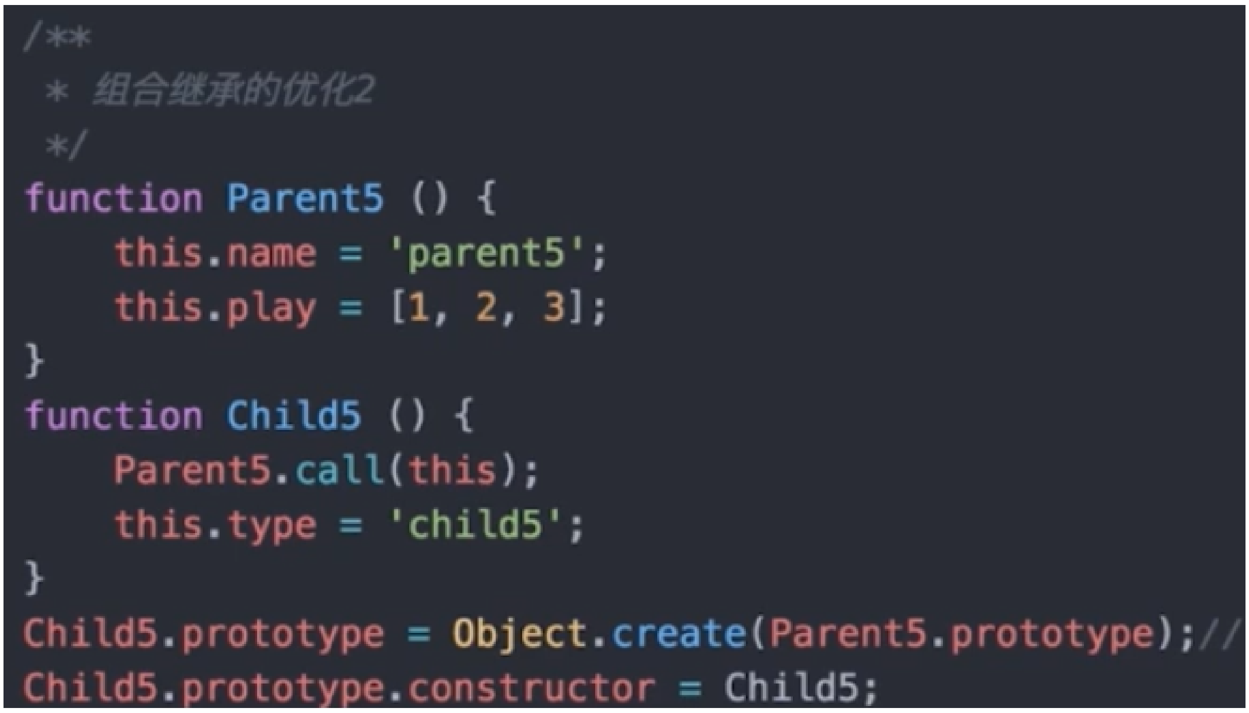
```
 * 借助原型链实现继承
 */
function Parent2 () {
    this.name = 'parent2';
    this.play = [1, 2, 3];
}
function Child2 () {
    this.type = 'child2';
}
Child2.prototype = new Parent2();
```



```
/**
 * 组合方式
 */
function Parent3 () {
    this.name = 'parent3';
    this.play = [1, 2, 3];
}
function Child3 () {
    Parent3.call(this);
    this.type = 'child3';
}
Child3.prototype = new Parent3();
```



```
/**
 * 组合继承的优化1
 */
function Parent4 () {
    this.name = 'parent3';
    this.play = [1, 2, 3];
}
function Child4 () {
    Parent4.call(this);
    this.type = 'child3';
}
Child4.prototype = Parent4.prototype;
var s5 = new Child3();
var s6 = new Child3();
```



```
/**
 * 组合继承的优化2
 */
function Parent5 () {
    this.name = 'parent5';
    this.play = [1, 2, 3];
}
function Child5 () {
    Parent5.call(this);
    this.type = 'child5';
}
Child5.prototype = Object.create(Parent5.prototype);//
Child5.prototype.constructor = Child5;
```