ECMAScript 支持面向对象（OO）编程，但不使用类或者接口。对象可以在代码执行过程中创建和增强，因此具有动态性而非严格定义的实体。在没有类的情况下，可以采用下列模式创建对象：

- 工厂模式：

使用简单的函数创建对象，为对象添加属性和方法，然后返回对象。这个模式被构造函数所取代

- 构造函数模式：

可以创建自定义引用类型，可以像创建内置对象实例一样使用 new 操作符。
缺点：每个成员都无法得到复用，包括函数。

- 原型模式：

使用构造函数的 prototype 属性来指定那些应该共享的属性和方法。组合使用构造函数模式和原型模式，使用构造函数定义实例属性，使用原型定义共享的属性和方法。

166

Javascript 主要通过原型链实现继承。原型链的构建是通过将一个类型的实例赋值给另一个构造函数的原型实现。这样，子类型就能够访问超类型的所有属性和方法，这一点与类的继承很相似。原型链的问题是对象实例共享所有继承的属性和方法，因此不宜单独使用。解决这个问题的技术是借用构造函数，即在子类型构造函数的内部调用超类型构造函数。这样就可以做到每个实例都有自己的属性，同时还能保证只使用构造函数模式来定义类型。使用最多的继承模式是组合继承，这种模式使用原型链继承共享的属性和方法，而通过借用构造函数继承实例属性。

其他继承模式：

- 原型式继承：可以在不必预先定义构造函数的情况下实现继承，其本质是执行给定对象的浅复制。而复制得到的副本还可以得到进一步改造
- 寄生式继承：与原型式继承非常相似，也是基于某个对象或某些信息创建一个对象，然后增强对象，最后返回对象。为了解决组合继承模式由于多次调用超类型构造函数而导致的低效率问题，可以将这个模式与组合继承一起使用
- 寄生组合式继承：集寄生式继承与组合继承的优点与一身，是实现基于类型继承的最有效的方式