|  |  |  |  |
| --- | --- | --- | --- |
| 继承方式 | 实现 | 优点 | 缺点 |
| 原型链 | subType.prototype=new supertype() | 方法复用 | 原型属性包含引用类型；不能向父类传递参数 |
| 借用构造函数 | Function subType(){  Supertype.call(this);  } | 每个子类拥有自己的属性，解决引用类型问题；可以向父类传递参数； | 函数定义在构造函数中，无函数复用。 |
| 组合继承(原型和借用构造函数) | 方法定义在原型上，属性定义在构造函数中。 | 集以上两种的优点 | 调用两次超类型构造函数。创建子类原型和子类构造函数内部 |
| 原型式继承 | 一个对象最为另一个对象的基础，将这个对象传递给Object()函数.实质是一次浅复制，h5新增的方法object.create() | 当只想让一个对像与另一个对象的形式一样，则没有必要使用构造函数时，可以使用原型式 | 包含引用类型时，始终会共享值，与原型模式一样。 |
| 寄生式继承 | 类似原型式继承，只不过是将原型式继承封装在一个函数中，该函数中再对该对象加以增强。 | 不仅与另个对象形式一样，而且也具有增强的方法。 | 方法没有复用，这点类似构造函数模式。 |
| 寄生组合继承 | 创建超类型原型的的副本 | 调用一次超类构造函数 |  |