面向对象继承的方式

目录：

直接复制property对象

浅拷贝

深拷贝

原型链

直接复制property对象

构造函数：*Drag*.call(**this**, elm);

方式：*Draglimite.prototype = Drag.prototype;*

优点：

缺点：*子级改变父级也会改变*

浅拷贝

构造函数：*Drag*.call(**this**, elm);

方式：*Object.assign(Draglimite.prototype, Drag.prototype)*

并且*：Draglimite.prototype.constructor=Draglimite;*

优点：

缺点：*只是浅拷贝，如果父级中有对象，这样复制，只是地址的引用*。并且需要改变自己的构造函数

深拷贝

构造函数：*Drag*.call(**this**, elm);

方式：*Draglimite.prototype=deepAssign(Drag.prototype);*

并且*：Draglimite.prototype.constructor=Draglimite;*

优点：*深度拷贝，不存在值的引用*

缺点：需要自己封装深拷贝的函数，函数考虑的方面比较多。并且需要改变自己的构造函数

原型链

构造函数：*Drag*.call(**this**, elm);

方式1：*Draglimite*.**prototype**=**new** *Drag*();

方式2：*Draglimite.prototype.\_\_proto\_\_=Drag.prototype;*

*缺点：子类的改变父类也就改变*

方式3：*Draglimite.prototype=Object.create(Drag.prototype)*

优点：利用了继承的思想，改变原型上的公共的内容

缺点：

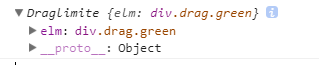
es6的extend方法

方式：**class** Draglimite **extends** Drag

优点：直接讲父级的构造函数的私有和共有内容同时复制下来

缺点：

深拷贝





原型链

clipboard.png

