js对象继承

原型继承

理解：子类原型是父类的实例。

特点：即继承了父类的模板，又继承了父类的原型对象。

例：

```
// 父类
function Person(){}
// 子类
function Boy(){}
Boy.prototype = new Person();
```

类继承

理解：借用构造函数的方法继承(call,apply)

特点：只继承模板，不继承原型对象

例：

```
// 父类
function Person(name,age){
   this.name = name;
   this.age = age;
}
// 子类
function Boy(name,age,sex){
```

```
    Person.call(this,name,age);
    this.sex = sex;
}
var b = new Boy('张三', 20, '男');
```

混合继承(原型继承+借用构造函数继承[类继承])

理解：使用call、apply方式改变要继承的对象内部的作用域并传入参数进行调用，获取另一对象的属性以及方法模板；再把另一对象实例(直接new Person()不传参数)赋值给该对象的原型，从而继承父类的原型(实际上同样继承了父类的模板)；

特点：继承了两次父类的模板，继承了一次父类的原型对象。

例：

```
// 父类
function Person(name,age){
    this.name = name;
    this.age = age;
}
// 父类的原型对象属性
Person.prototype.id = 10;
Person.prototype.sayName = function(){
    console.log(this.name);
};
// 子类
function Boy(name,age,sex){
    // call、apply
    // call绑定父类的模板函数，实现借用构造函数继承，只复制了父类的模板
    Person.call(this,name,age); // 1.借用构造函数继承
    this.sex = sex;
}
// 2.原型继承
```

```
// 只与 _父类的实例_ 和 _父类的原型对象_ 产生关系。
// 原型继承的方式：即继承了模板，又继承了父类的原型对象
Boy.prototype = new Person(); // 继承父类的原型(同样复制了
Person模板)
var b = new Boy('张三', 20, '男');
(new Person(),不传参数，调用Person内的属性和方法会返回
undefinedbaid)
```