js实现继承

```
//1、继承第一种方式：对象冒充
function Parent(username){
    this.username = username;
    this.hello = function(){
        alert(this.username);
    }
}
function Child(username,password){
    //通过以下3行实现将Parent的属性和方法追加到Child中，从而实现继承
    //第一步：this.method是作为一个临时的属性，并且指向Parent所指向的对象，
    //第二步：执行this.method方法，即执行Parent所指向的对象函数
    //第三步：销毁this.method属性，即此时Child就已经拥有了Parent的所有属性和方法
    this.method = Parent;
    this.method(username);//最关键的一行
    delete this.method;


    this.password = password;
    this.world = function(){
        alert(this.password);
    }
}
var parent = new Parent("zhangsan");
var child = new Child("lisi","123456");
parent.hello();
child.hello();
child.world();
```

```
/*2、继承第二种方式：call()方法方式
  call方法是Function类中的方法
  call方法的第一个参数的值赋值给类(即方法)中出现的this
  call方法的第二个参数开始依次赋值给类(即方法)所接受的参数*/


function test(str){
    alert(this.name + " " + str);
}
var object = new Object();
object.name = "zhangsan";
test.call(object,"langsin");//此时，第一个参数值object传递给了test类(即方法)中出现的this，而第二个参数"langsin"则赋值给了test类(即方法)的str

function Parent(username){
    this.username = username;
    this.hello = function(){
        alert(this.username);
```

```
    }
}
function Child(username,password){
    Parent.call(this,username);

    this.password = password;
    this.world = function(){
        alert(this.password);
    }
}
var parent = new Parent("zhangsan");
var child = new Child("lisi","123456");
parent.hello();
child.hello();
child.world();
```

```
/*3、继承的第三种方式：apply()方法方式
  apply方法接受2个参数，
    A、第一个参数与call方法的第一个参数一样，即赋值给类(即方法)中出现的this
    B、第二个参数为数组类型，这个数组中的每个元素依次赋值给类(即方法)所接受的参数*/

function Parent(username){
    this.username = username;
    this.hello = function(){
        alert(this.username);
    }
}
function Child(username,password){
    Parent.apply(this,new Array(username));

    this.password = password;
    this.world = function(){
        alert(this.password);
    }
}
var parent = new Parent("zhangsan");
var child = new Child("lisi","123456");
parent.hello();
child.hello();
child.world();
```

```
//4、继承的第四种方式：原型链方式，即子类通过prototype将所有在父类中通过prototype追加的属性和方法都追加到Child，从而实现了继承
function Person(){
}
Person.prototype.hello = "hello";
Person.prototype.sayHello = function(){
    alert(this.hello);
}

function Child(){
```

```
}
Child.prototype = new Person();//这行的作用是：将Parent中将所有通过prototype追加的属性和方法都追加到Child，从而实现了继承
Child.prototype.world = "world";
Child.prototype.sayWorld = function(){
    alert(this.world);
}

var c = new Child();
c.sayHello();
c.sayWorld();
```

```
/*5、继承的第五种方式：混合方式
  混合了call方式、原型链方式*/

function Parent(hello){
    this.hello = hello;
}
Parent.prototype.sayHello = function(){
    alert(this.hello);
}
function Child(hello,world){
    Parent.call(this,hello);//将父类的属性继承过来
    this.world = world;//新增一些属性
}

Child.prototype = new Parent();//将父类的方法继承过来
Child.prototype.sayWorld = function(){//新增一些方法
    alert(this.world);
}

var c = new Child("zhangsan","lisi");
c.sayHello();
c.sayWorld();
```