什么是原型？

1.原型产生的原因

实例化其实是一个很耗费内存空间的，通过构造函数去实例对象的时候，每一个对象中的属性和方法都是在自己的空间，那么当有多个对象的时候，这些属性和方法就有多少个在各自的空间中存在在，所以很浪费空间，因此，为了解决这个问题，实现数据共享，原型就产生了。

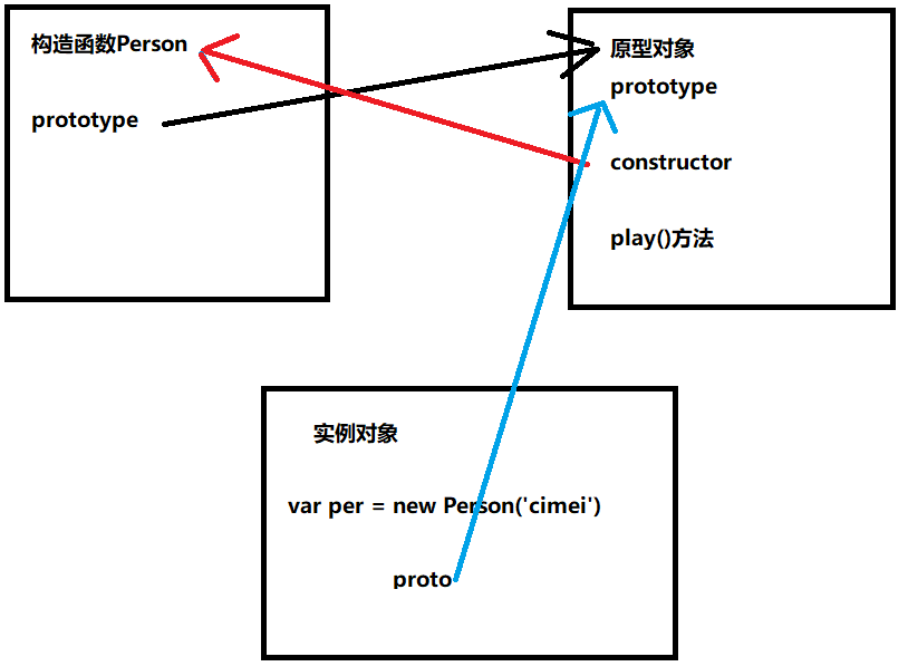
2.构造函数中有个prototype，是原型，也是我们平时所使用的

3.而我么通过构造函数所实例化的对象中-Proto-属性，这也是原型，是浏览器使用的，这个下划线原型指向的就是该实例对象中的构造函数中的prototype

4.如何访问的到函数中的prototype呢？

实例对象.__proto__ 这样就可以实现访问prototype中的属性和方法啦

```
        function Person(name){
            this.name = name ;
        }
        Person.prototype.play = function(){
            console.log('打球');
        }
        var p1 = new Person('cimei');
        p1.play();
        console.dir(Person);//查看构造函数的结构
        console.dir(p1);//查看实例对象的结构
        console.log(p1.__proto__== Person.prototype)//看他们的指向是否一致，此处结果是true
```



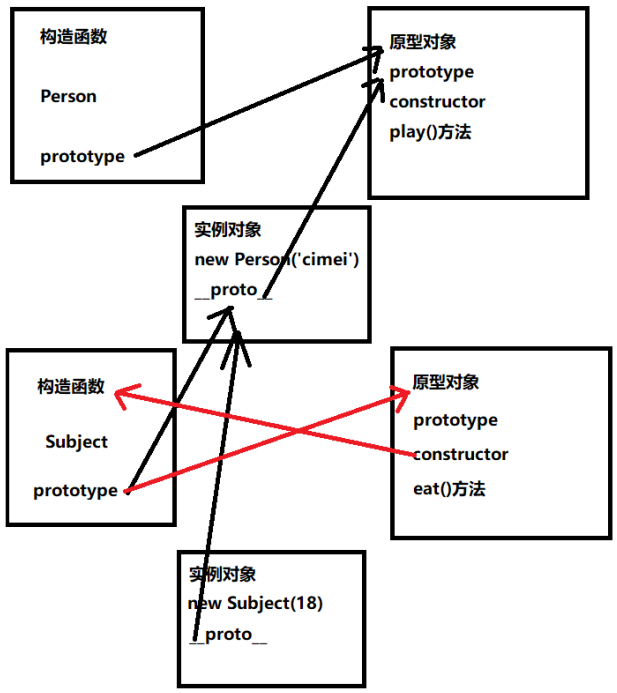
原型的指向问题

构造函数中的this就是实例对象（谁调用的就是谁的）

原型对象中的方法中的this就是实例对象

```
//Person的构造函数
   function Person(name){
            this.name = name;
        }
//Person的原型对象上的方法
        Person.prototype.play = function(){
            console.log('打球');
        }
//Subject的构造函数
        function Subject(age){
            this.age = age;
        }
//Subject的原型对象上的方法
   Subject.prototype.eat = function(){
            console.log('今天吃饺子');
        }
//Subject的原型，指向了Person的实例对象
        Subject.prototype = new Person('cimei');
        var sub = new Subject(18);
//此时发现可以调用Person原型上的方法
```

```
        sub.play();
//此时会报错
  sub.eat();
```



黑色线是真正的走向，红色的线是被改变了的，已经不存在了。

```
总结：
实例对象的原型__proto__指向的是该对象的构造函数的原型对象

构造函数的原型对象prototype 指向如果改变了，那么实例对象的原型__proto__指向自然也就会跟着改变

原型链

其实实例对象与构造函数并没有直接的关系，两者之间的关系就通过实例对象中的那个__proto__和原型对象（prototype）来联系的。而我们也称之为原型链
```

原型指向改变的话如何添加方法并访问呢?

1.如上述例子，改变了Subject的原型的指向，发现调用不了原来Subject原型上的方法了

解决办法：可以在原型改变之后 再添加在其原型上添加方法。

2.访问属性或方法

说白了，就是就近原则，实例对象有的话，就直接用，实例对象上没有，就去原型对象上找。这里有个问题，就是如果找不到也不会报错，会返回一个undefined，因为js是一门动态类型的语言，对象没有的话，只要有点(.)，那么这个对象就有了这个东西，没有我们要的这个东西，只要有对象.属性这个名字，对象就有了这个属性，但是还没有给其赋值，所以是undefined的。总结就是js有立即创造属性的能力，大方的很，你没有，我就给你一个。

封装：其实就是把一些相似的对象放在js文件按中

继承：js中的继承是通过构造函数去模拟类，然后通过原型来实现的。其中原型实现了数据的共享，为我们节约了内存空间。

属性的继承体现在构造函数的继承上,调用父类的构造函数使用call，apply,方法的继承就是prototype的指向问题了

那么原型的作用也就有了两个：一是节约了内存空间，二是实现了继承

多态：一个对象有不同的行为，或者是一个行为针对不同的对象，产生不同的结果，要想实现多态，就要先有继承