JavaScript进阶

---深入理解JS的继承方式

河北师范大学软件学院
Software College of Hebei Normal University

内容提纲

- JS对象-对象原型继承
- 通过构造函数模拟类-类的继承
- JS继承补充部分

JS原型继承的方式及其优缺点

- JavaScript的原型继承是对象-对象的继承
  - 每个对象都有一个原型对象（可动态的指定原型，来改变继承关系，最原始的原型是null）
  - 思考并回答三种方式创建的对象的原型都是什么？
  - 多个对象继承于一个原型时，存在原型共享（节省内存如共享方法，但也带来了共享问题）

```
var superObj = {
    x:1,
    y:2
};
var subObj_First = Object.create(superObj);
var subObj_Second = Object.create(superObj);
subObj_First.__proto__.x = 5; 若此行写为subObj_First.x = 5;结果又是如何?
console.log(subObj_Second.x);
```


参见实例demo07

JS原型继承的方式及其优缺点

- 构造函数实现的对象-对象的原型继承的原型共享问题

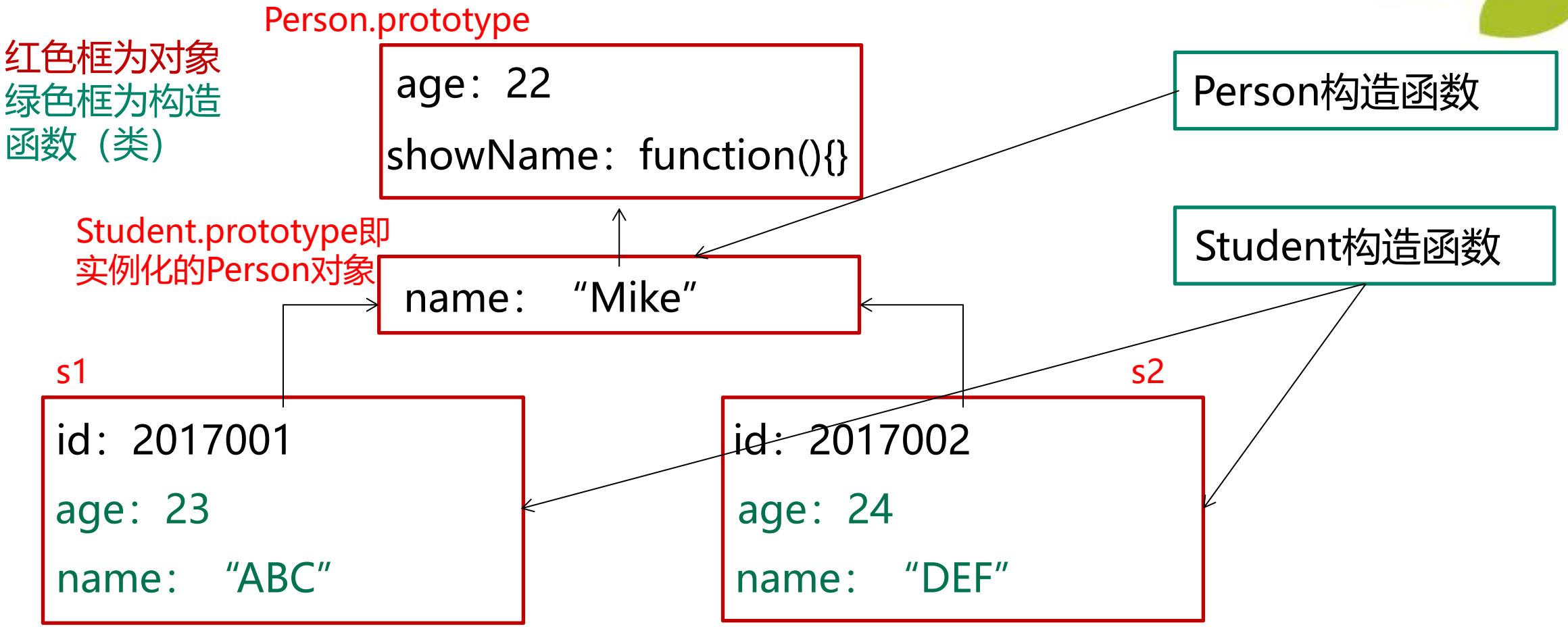
```
function Person(name){
    this.name = name;
};
Person.prototype.age = 22;
Person.prototype.showName = function(){console.log(this.name);};
function Student(id){
    this.id = id;
}
Student.prototype = new Person("Mike");
var s1 = new Student(2017001);
var s2 = new Student(2017002);
//测试如下代码，思考为什么，这样的继承有什么弊端
console.log(s1.name,s1.age,s1.id);
console.log(s2.name,s2.age,s2.id);
s1.__proto__.name = "Jack";
console.log(s2.name);
s2.__proto__.__proto__.age = 99;
console.log(s2.age);
```

左侧的代码有什么问题
思考共享的弊端，如何给每个Studnent对象添加自有的name属性，s1.name =” Jack”，和原型的name属性什么关系，思考这样的话是否造成内存的浪费，具体参见下页图解

参见实例demo08

JS原型继承的方式及其缺点

- 上页代码图解（原型共享问题）



参见实例demo08 原型继承的原型共享问题

内容提纲

- JS对象-对象原型继承
- **通过构造函数模拟类-类的继承**
- JS继承补充部分

模拟类-类继承的形式一

```
function Person(name,age){
    this.name = name;
    this.age = age;
};
Person.prototype.showName = function(){
    console.log(this.name);
};
function Student(name,age,id){
    Person.call(this,name,age);
    this.id = id;
}
Student.prototype.__proto__ = Person.prototype;
var s1 = new Student("xxx",22,2017001);
var s2 = new Student("www",23,2017002);
```

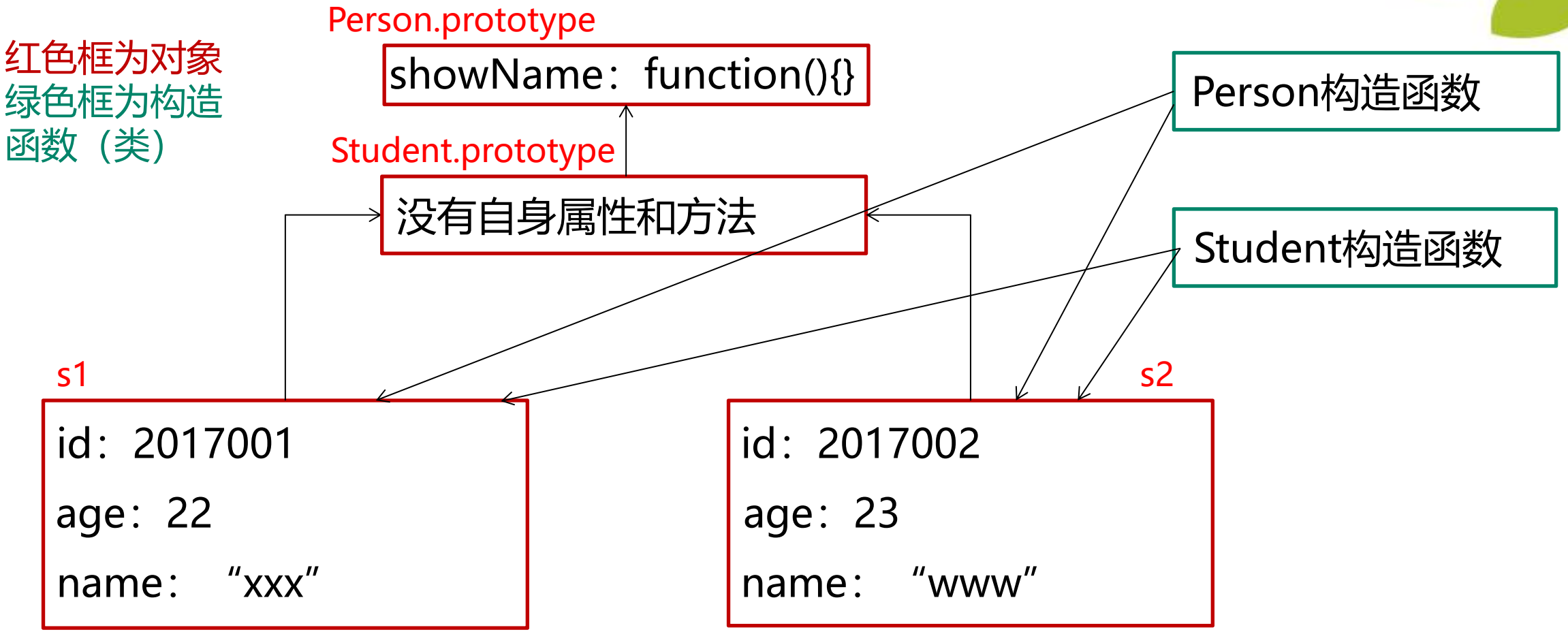
思考：name属性添加到哪个对象上了？Person.prototype、Student.prototype还是实例化的对象上？

推荐：将方法添加到对象的原型上（即构造函数的prototype上）便于共享，节省内存

参见实例demo09 Part1 JS面向对象继承

模拟类-类继承的形式-图解

- 构造函数实现的对象-对象的原型继承的原型共享问题



参见实例demo08 原型继承的原型共享问题

模拟类-类继承的形式 二

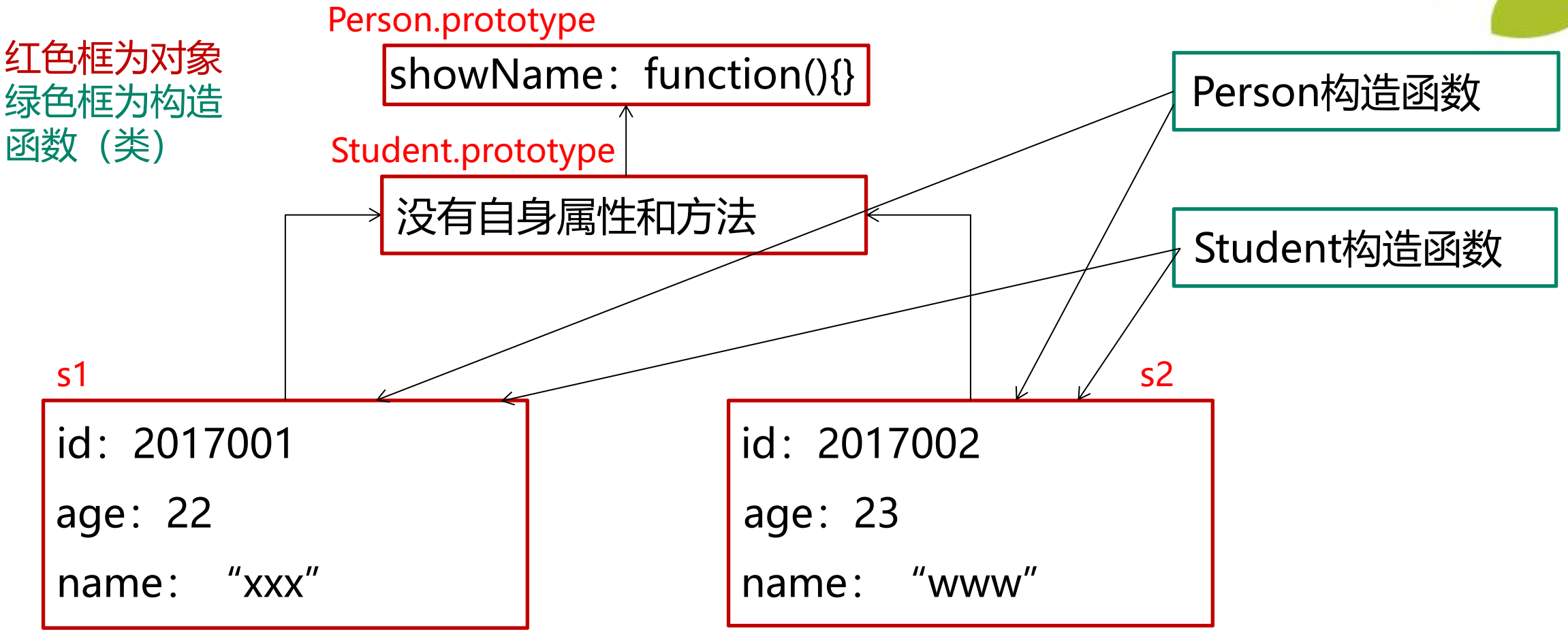
```
function Person(name,age){
    this.name = name;
    this.age = age;
};
Person.prototype.showName = function(){
    console.log(this.name);
};
function Student(name,age,id){
    Person.call(this,name,age);
    this.id = id;
}
Student.prototype = Object.create(Person.prototype);
Student.prototype.constructor = Student;
var s1 = new Student("xxx",22,2017001);
var s2 = new Student("www",23,2017002);
```

如果不把
Student.prototype.constructor
指回Student，那它将指向谁?

参见实例demo09 Part2 JS面向对象继承

模拟类-类继承的形式-图解

- 构造函数实现的对象-对象的原型继承的原型共享问题



参见实例demo08 原型继承的原型共享问题

内容提纲

- JS对象-对象原型继承
- 通过构造函数模拟类-类的继承
- JS继承补充部分

JS继承补充部分

- 静态方法与原型方法的区别
  - 静态方法是构造器函数对象（类）的方法，原型方法是实例化对象（对象）的原型的方法
  - 使用形式有什么不同，区别在哪里？（属性共享）
  - 思考Object.getPrototypeOf(...)与Object.prototype.isPrototypeOf(...)

```
var BaseClass = function() {};
BaseClass.prototype.f2 = function () {
    console.log("This is a prototype method ");
};
BaseClass.f1 = function(){//定义静态方法
    console.log("This is a static method ");
};
BaseClass.f1();//This is a static method
var instance1 = new BaseClass();
instance1.f2();//This is a prototype method
```

参见实例demo10 静态方法与原型方法

JS继承补充部分

- 再谈对象原型的constructor属性
  - 因为对象实例从原型中继承了constructor，所以可以通过constructor得到实例的构造函数
  - 确定对象的构造函数名、创建相似对象、constructor可用于指定构造函数

```
function Foo() {}
var f = new Foo();
console.log(f.constructor.name);//Foo

function Constr(name) {this.name = name;}
var x = new Constr("Jack");
var y = new x.constructor("Mike");
console.log(y,y instanceof Constr);
```

```
▶ Constr {name: "Mike"} true
```


参见实例demo11 Part1 constructor应用

JS继承补充部分

- 对象的公有属性、私有属性（回顾闭包）

```
function A(id) {
    this.publicId = id;
    var privateId = 456;
    this.getId = function () {
        console.log(this.publicId,privateId);
    };
}
var a = new A(123);
console.log(a.publicId);//123
// console.log(a.privateId);
a.getId();//123 456
```

涉及到访问私有属性时，需将间接访问私有变量的函数定义在构造函数中

参见实例demo11 Part2 公有属性、私有属性

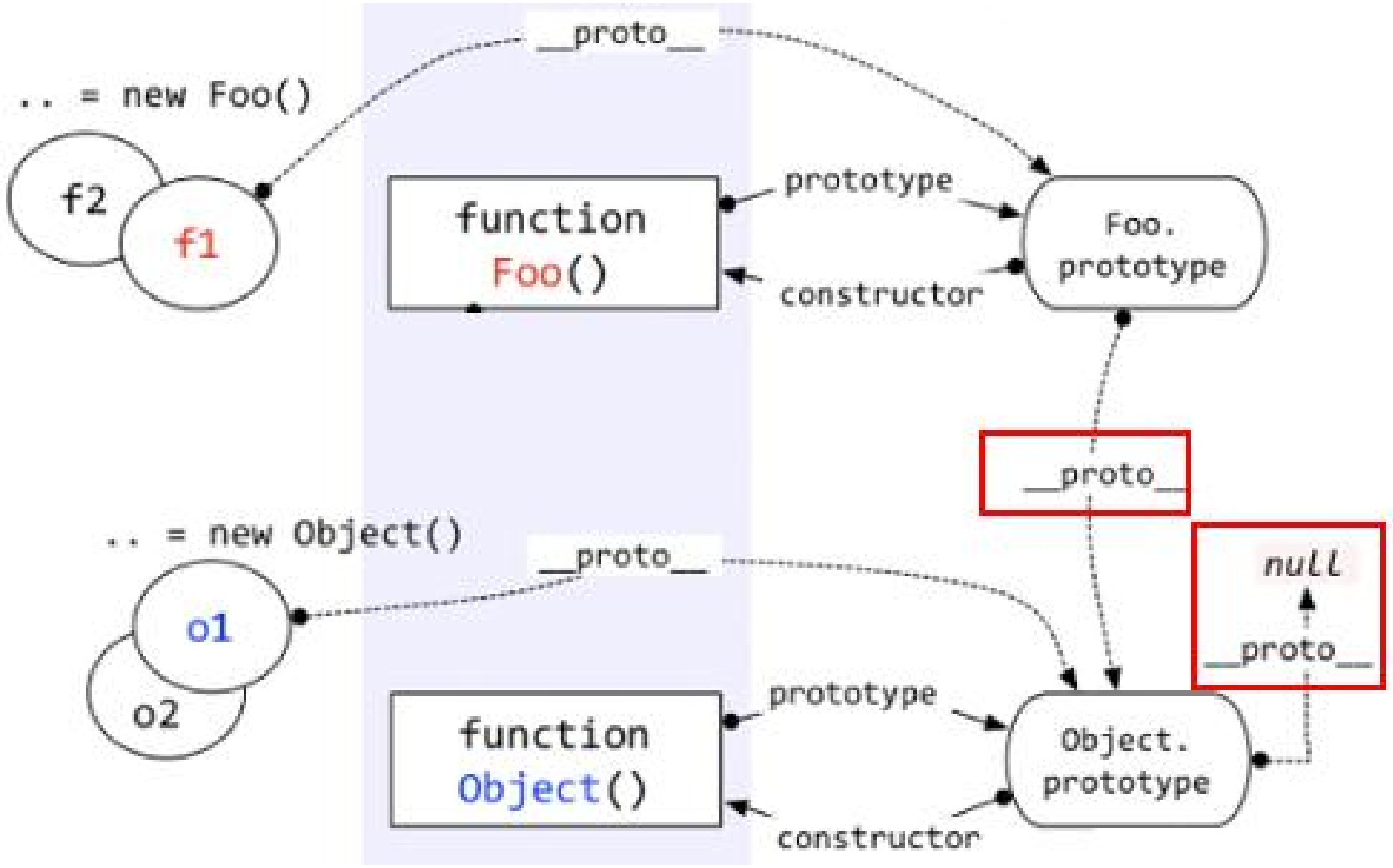
Thank You!

作业

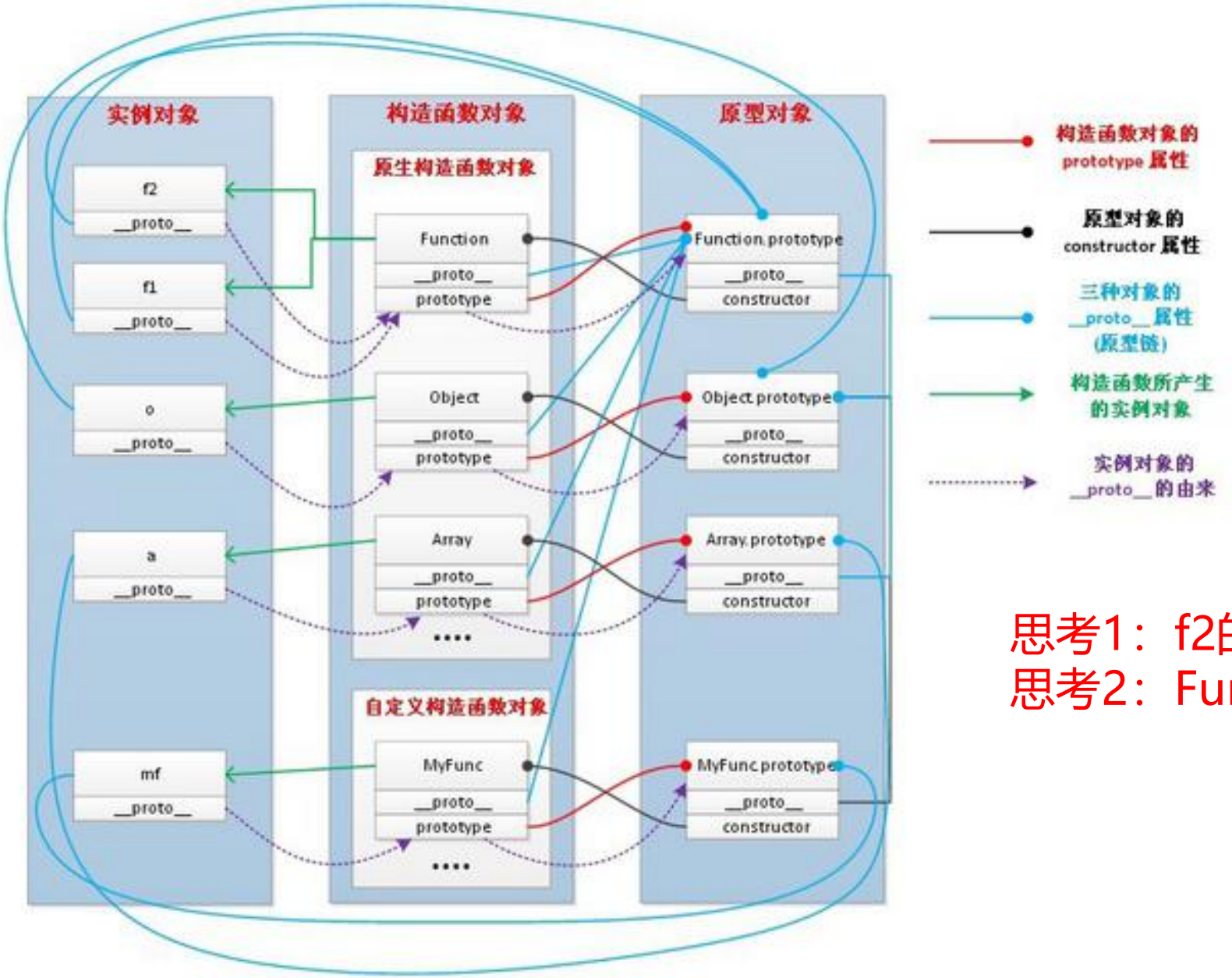
- 阅读《深入理解JavaScript》的第17章
- 学习并重写FlappyBird案例

  http://pan.baidu.com/s/1ge3H8YJ

原型链图解


__proto__
.. = new Foo()
f2
f1
function
Foo()
prototype
constructor
Foo.
prototype
__proto__
.. = new Object()
__proto__
o1
o2
null
__proto__
prototype
function
Object()
Object.
prototype
constructor

原型链图解



思考1：f2的prototype是谁

思考2：Function.prototype是谁

阅读下述文章-JavaScript万物诞生记

- http://web.jobbole.com/88493/