

途虎养车

养车 就是途虎

JS原型链

马珩

大纲总览

- 面向对象编程
- 原型和原型链
- 原型继承
- Class继承
- 课堂小测验
- Q&A

面向对象编程

虎

核心概念

- 封装
- 继承
- 多态

继承

可以让某个类型的对象获得另一个类型的对象的属性的方法

提供一种能力：可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展

如何实现继承

// java中通过extends关键字实现继承

```
Class A {
```

```
}
```

```
Class B extends A {
```

```
}
```

// C++中通过指定访问修饰符来继承

```
Class A {
```

```
}
```

```
Class B: public A {
```

```
}
```

Javascript该如何继承?

原型和原型链

虎

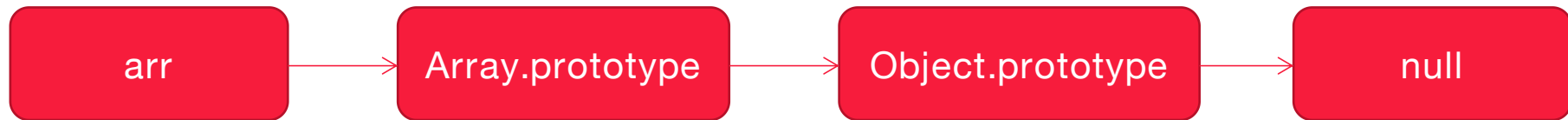
原型对象

JavaScript对每个创建的对象都会设置一个原型属性，指向它的原型对象，可以通过__proto__这个非标准属性访问到。

原型链

当我们用`obj.xxx`访问一个对象的属性时，JavaScript引擎先在当前对象上查找该属性，如果没有找到，就到其原型对象上找，如果还没有找到，就一直上溯到`Object.prototype`对象，最后，如果还没有找到，就只能返回`undefined`。

```
var arr = [1, 2, 3]
```



构造函数

定义：通过 new 函数名 来实例化对象的函数

和普通函数的区别：主要从功能上进行区分。构造函数的主要功能为初始化对象，特点是和new一起使用。new就是在从无到有创建一个对象，构造函数就是在为初始化的对象添加属性和方法。

构造函数定义时首字母大写（规范）。

```
// 构造函数
function Student (props){
  this.name = props.name;
  this.hello = function(){
    console.log('hello world')
  }
}
//创建实例对象
let xiaoming = new Student('xiaoming')
let xiaohong = new Student('xiaohong')
```

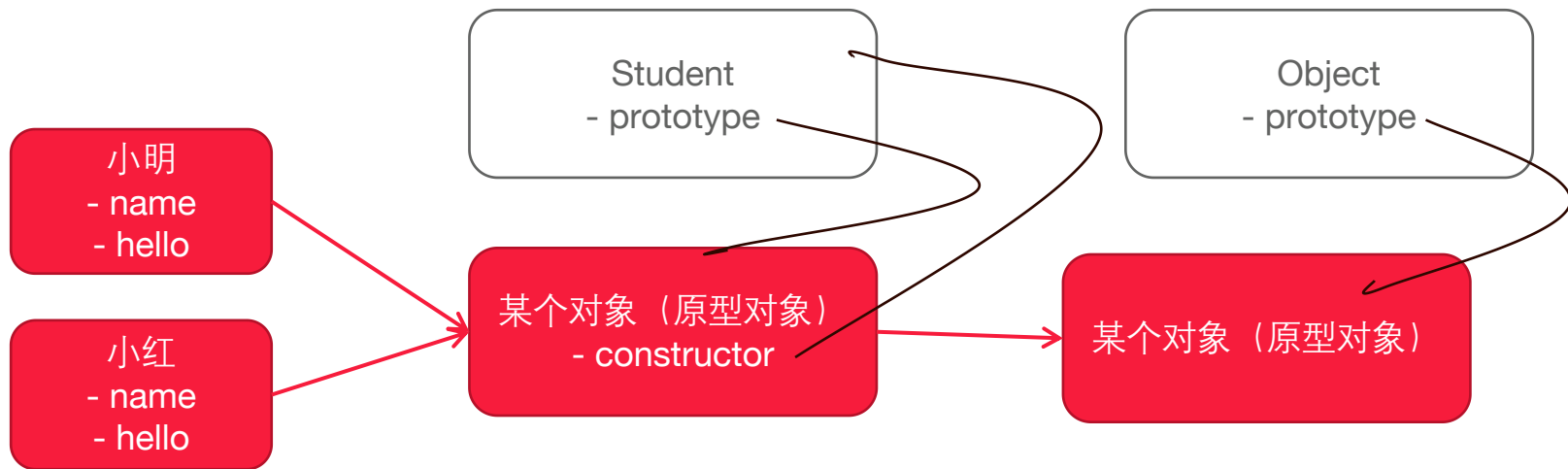
new调用时发生了什么

使用new调用函数会执行几步操作

1. 创建一个全新的对象obj={}
2. 执行prototype链接，设置新对象的_proto_属性指向构造函数的prototype对象
`obj._proto_ = Function.prototype`
3. 新对象obj会绑定到函数调用的this;
`Function.call(obj)`
4. 如果函数没有返回其他对象，则new表达式的函数调用会自动返回这个新对象

constructor

使用new关键字创建的对象还从原型上获得一个constructor属性，指向函数本身




原型继承

回到继承

在传统的基于Class的语言如Java、C++中，继承的本质是扩展一个已有的Class，并生成新的Subclass，由于这类语言严格区分类和实例，继承实际上是类型的扩展。

基于Student扩展出PrimaryStudent

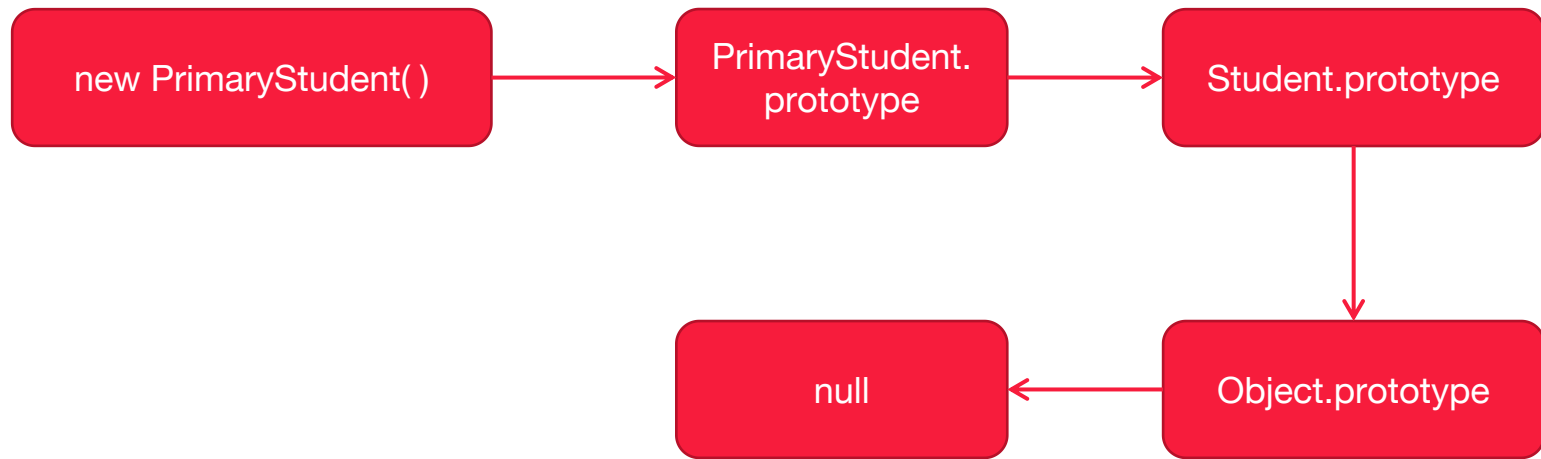


```
function PrimaryStudent(props) {  
  // 调用Student构造函数, 绑定this  
  Student.call(this, props);  
  this.grade = props.grade || 1;  
}
```

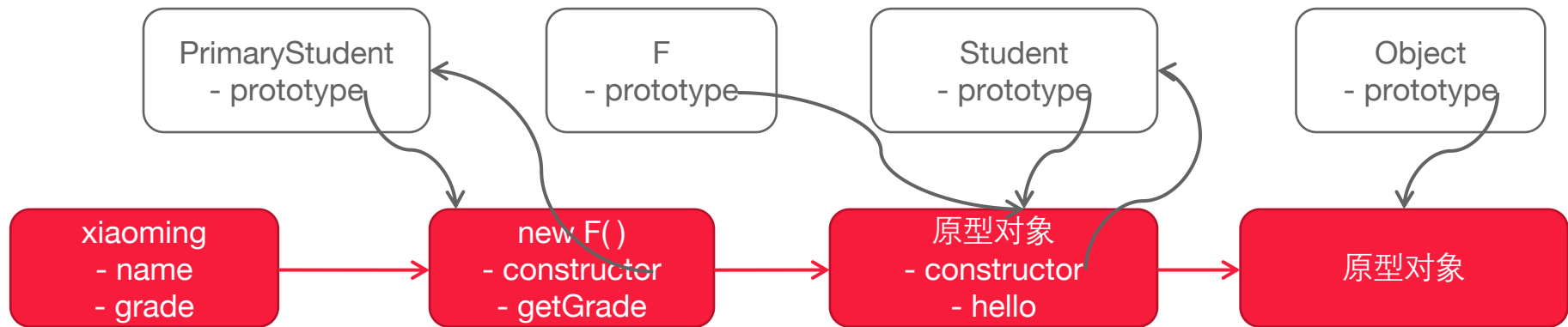
此时的原型链



我们期望的原型链



现在的原型链



Class继承

ES6 Class

Class的目的就是让定义类更简单，让JS引擎去做原型继承的事情。

```
Class PrimaryStudent extends Student{
  constructor(name, grade) {
    super(name); // 调用父类的构造方法
    this.grade = grade;
  }

  getGrade() {
    console.log('my grade:' + this.grade);
  }
}
```

课堂小测验



Q&A

途虎养车

养车 就是途虎