面向对象

掌握面向对象编程思想

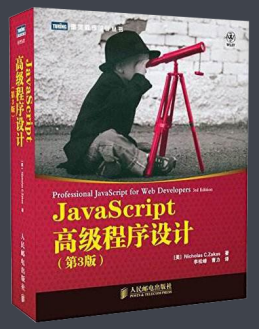
前面学习的 JavaScript 的内容，属于初学 JavaScript 必经的一个阶段，但是如果想进一步使用该语言撰写更有效率的代码，理解 JavaScript 面向对象（object-oriented, OO）的精髓是很重要的。

什么是对象

- Everything is object （万物皆对象）。
- 我们可以从两次层次来理解：
  （1）对象是单个事物的抽象。
  （2）对象是一个容器，封装了属性（property）和方法（method）。

什么是对象

- 对象是单个事物的抽象。



什么是对象

- 对象是一个容器，封装了属性（property）和方法（method）。
- 属性：对象的状态
- 方法：对象的行为

什么是对象

- 在实际开发中，对象是一个抽象的概念，可以将其简单理解为：数据集或功能集。
- ECMAScript-262 把对象定义为：无序属性的集合，其属性可以包含基本值、对象或者函数。

什么是面向对象

- 面向对象编程 —— Object Oriented Programming，简称 OOP ，是一种编程开发思想。
- 它将真实世界各种复杂的关系，抽象为一个个对象，然后由对象之间的分工与合作，完成对真实世界的模拟。

面向对象编程中常见概念深入解析

继承：继承自拖拉机，实现了扫地的接口。
封装：无需知道如何运作，开动即可。
多态：平时扫地，天热当风扇。
重用：没有额外动力，重复利用了发动机能量。
多线程：多个扫把同时工作。
低耦合：扫把可以换成拖把而无需改动。
组件编程：每个配件都是可单独利用的工具。
适配器模式：无需造发动机，继承自拖拉机，只取动力方法。
代码托管：无需管理垃圾，直接扫到路边即可。

面向对象与面向过程对比

- 面向过程就是亲力亲为，事无巨细，面面俱到，步步紧跟，有条不紊
- 面向对象就是找一个对象，指挥得结果
- 面向对象将执行者转变成指挥者
- 面向对象不是面向过程的替代，而是面向过程的封装

面向对象的特性

- 封装性
- 继承性
- [多态性]抽象

总结

- 在面向对象程序开发思想中，每一个对象都是功能中心，具有明确分工，可以完成接受信息、处理数据、发出信息等任务。
- 因此，面向对象编程具有灵活、代码可复用、高度模块化等特点，容易维护和开发，比起由一系列函数或指令组成的传统的过程式编程（procedural programming），更适合多人合作的大型软件项目。

体验面向过程和面向对象

案例

- 处理学生的成绩表，打印输出学生成绩。

面向对象的设计思想

- 抽象出 Class(构造函数)
- 根据 Class(构造函数) 创建 Instance(实例)
- 指挥 Instance 得结果

创建对象的几种方式

创建对象的几种方式

- new Object() 构造函数
- 对象字面量 {}
- 工厂函数
- 自定义构造函数

构造函数和实例对象的关系

- 构造函数是根据具体的事物抽象出来的抽象模板
- 实例对象是根据抽象的构造函数模板得到的具体实例对象
- 每一个实例对象都通过一个 constructor 属性，指向创建该实例的构造函数

  注意：constructor 是实例的属性的说法不严谨，具体后面的原型会讲到
- 可以通过 constructor 属性判断实例和构造函数之间的关系

  注意：这种方式不严谨，推荐使用 instanceof 操作符，后面学原型会解释为什么

静态成员和实例成员

- 使用构造函数方法创建对象时，可以给构造函数和创建的实例对象添加属性和方法，这些属性和方法都叫做成员。
- 实例成员：在构造函数内部添加给 this 的成员，属于实例对象的成员，在创建实例对象后必须由对象调用。
- 静态成员：添加给构造函数自身的成员，只能使用构造函数调用，不能使用生成的实例对象调用。

构造函数的问题

- 浪费内存

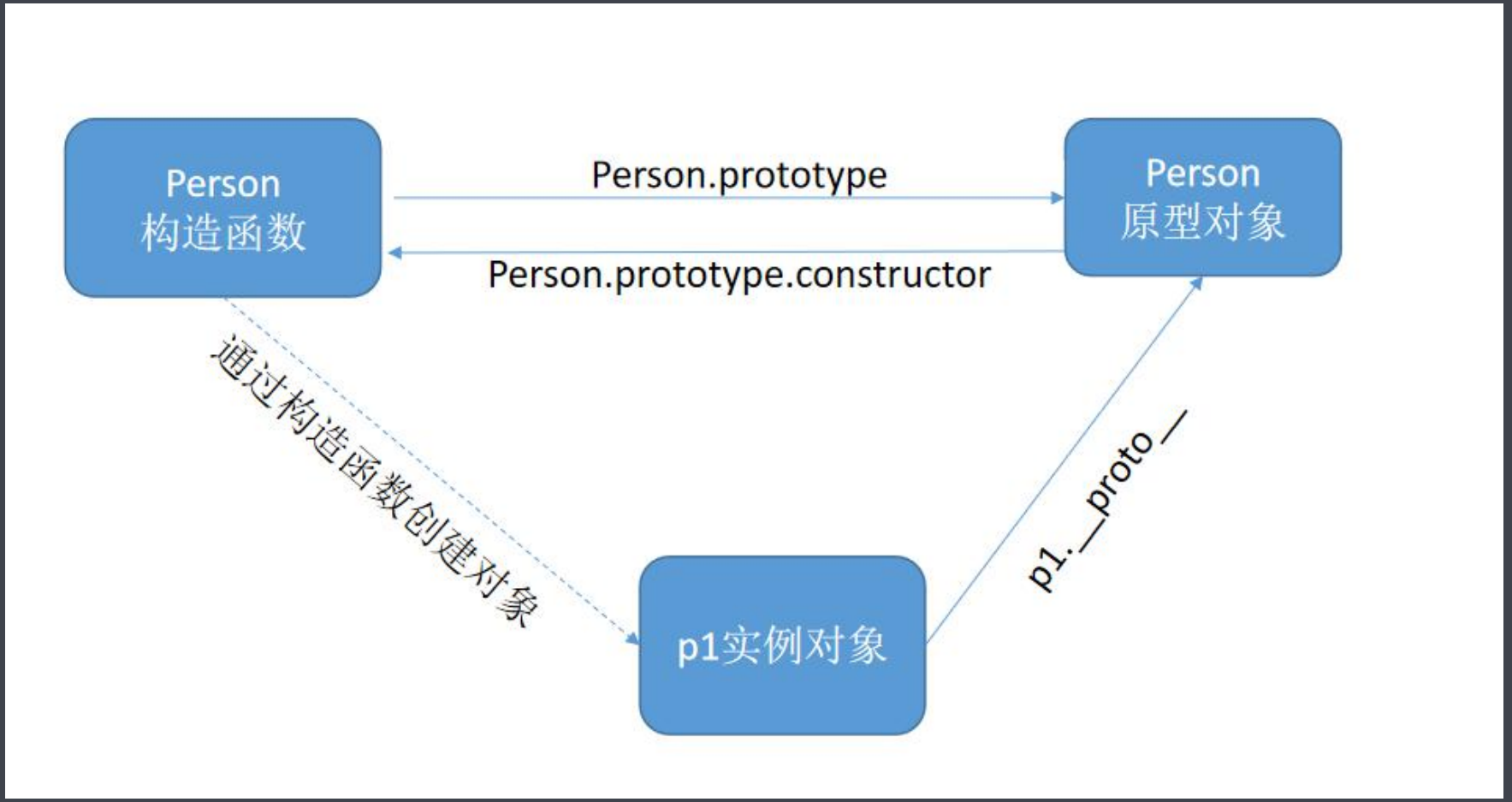
原型

使用原型对象可以更好的解决构造函数的内存浪费问题。

prototype 原型对象

- 任何函数都具有一个 prototype 属性，该属性是一个对象。
- 可以在原型对象上添加属性和方法。
- 构造函数的 prototype 对象默认都有一个 constructor 属性，指向 prototype 对象所在函数。
- 通过构造函数得到的实例对象内部会包含一个指向构造函数的 prototype 对象的指针 __proto__。
- 实例对象可以直接访问原型对象成员。

构造函数、实例、原型对象三者之间的关系



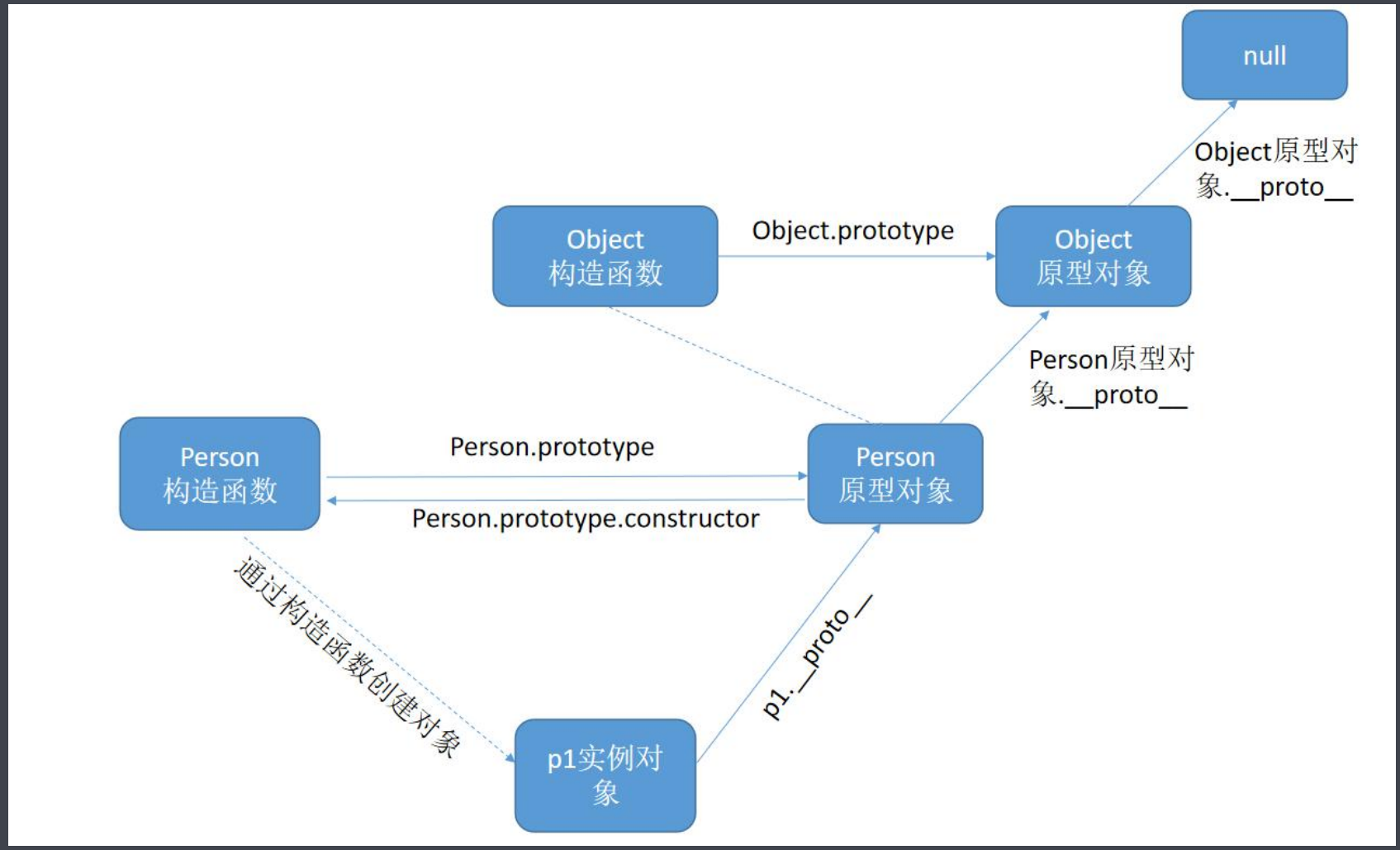
解决方法

- JavaScript 规定，每一个构造函数都有一个 prototype 属性，指向构造函数的原型对象。
- 这个原型对象的所有属性和方法，都会被构造函数的实例对象所拥有。
- 因此，我们可以把所有对象实例需要共享的属性和方法直接定义在 prototype 对象上。
- 解决内存浪费问题

原型链

思考：为什么实例对象可以调用构造函数的 prototype 原型对象的属性和方法？

原型链


null
Object原型对象.__proto__
Object
构造函数
Object.prototype
Object
原型对象
Person原型对象.__proto__
Person
构造函数
Person.prototype
Person.prototype.constructor
Person
原型对象
通过构造函数创建对象
p1实例对象
p1.__proto__

原型链查找机制

每当代码读取某个对象的某个属性时，都会执行一次搜索，目标是具有给定名字的属性：

1. 搜索首先从对象实例本身开始
2. 如果在实例中找到了具有给定名字的属性，则返回该属性的值
3. 如果没有找到，则继续搜索指针指向的原型对象，在原型对象中查找具有给定名字的属性
4. 如果在原型对象中找到了这个属性，则返回该属性的值

实例对象读写原型对象成员

实例对象读写原型对象成员

读取：

- 先在自己身上找，找到即返回
- 自己身上找不到，则沿着原型链向上查找，找到即返回
- 如果一直到原型链的末端还没有找到，则返回 undefined

实例对象读写原型对象成员

值类型成员写入（实例对象.值类型成员 = xx）：

- 当实例期望重写原型对象中的某个普通数据成员时实际上会把该成员添加到自己身上
- 也就是说该行为实际上会屏蔽掉对原型对象成员的访问

引用类型成员写入（实例对象.引用类型成员 = xx）：

- 同上

实例对象读写原型对象成员

复杂类型成员修改（实例对象.成员.xx = xx）：

- 同样会先在自己身上找该成员，如果自己身上找到则直接修改
- 如果自己身上找不到，则沿着原型链继续查找，如果找到则修改
- 如果一直到原型链的末端还没有找到该成员，则报错（实例对象.undefined.xx = xx）

更简单的原型语法

更简单的原型语法

前面在原型对象每添加一个属性和方法就要书写一遍 Person.prototype 。

为减少不必要的输入，更常见的做法是用一个包含所有属性和方法的对象字面量来重写整个原型对象，将 Person.prototype 重置到一个新的对象。

注意：原型对象会丢失 constructor 成员，所以需要手动将 constructor 指向正确的构造函数。

原型对象使用建议

在定义构造函数时，可以根据成员的功能不同，分别进行设置：

- 私有成员（一般就是非函数成员）放到构造函数中
- 共享成员（一般就是函数）放到原型对象中
- 如果重置了 prototype 记得修正 constructor 的指向

原生构造函数的原型对象

JS 原生构造函数的原型对象

所有函数都有 prototype 属性对象。

JavaScript中的内置构造函数也有 prototype 原型对象属性：

- Object.prototype
- Function.prototype
- Array.prototype
- String.prototype
- Number.prototype
- ...

练习

为数组对象扩展原型方法。

随机方块案例

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容