

再谈javascripts原型与原型链及继承相关问题

发布于 2019-12-04 03:52:35 494 0

举报

文章被收录于专栏：[IMWeb前端团队](#)

本文作者：IMWeb 周陆军的微博 原文出处：[IMWeb社区](#) 未经同意，禁止转载

什么是原型语言

只有对象,没有类;对象继承对象,而不是类继承类。

“原型对象”是核心概念。原型对象是新对象的模板，它将自身的属性共享给新对象。一个对象不但可以享有自己创建时和运行时定义的属性，而且可以享有原型对象的属性。

每一个对象都有自己的原型对象，所有对象构成一个树状的层级系统。root节点的顶层对象是一个语言原生的对象，只有它没有原型对象，其他所有对象都直接或间接继承它的属性。

原型语言创建有两个步骤

使用“原型对象”作为“模板”生成新对象：这个步骤是必要的，这是每个对象出生的唯一方式。以原型为模板创建对象，这也是“原型”(prototype)的原意。

初始化内部属性：这一步骤不是必要的。通俗点说，就是对“复制品”不满意，我们可以“再加工”，使之获得不同于“模板”的“个性”。

所以在JavaScript的世界里，万物皆对象这个概念从一而终。

js本地对象、内置对象、宿主对象

全局对象：一般全局对象会有两个，一个是ecma提供的Global对象，一个是宿主提供。如在浏览器中是window、在nodejs中是global。【所以啊，在浏览器中全局对象Global+window】 **通常情况下ecma提供的Global对象对是不存在的，没有具体的对象**

宿主对象 - host object：即由 ECMAScript实现的宿主环境提供的对象，包含两大类，一个是宿主提供，一个是自定义类对象，ECMAScript官方未定义的对象都属于宿主对象,所有非本地对象都是宿主对象。宿主提供对象原理--->由宿主框架通过某种机制注册到ECscript引擎中的对象，如宿主浏览器（以远景为参考）会向ECscript注入window对象，构建其实现javascript。

内置对象 - Build-in object：由 ECMAScript实现提供的、独立于宿主环境的所有对象，在 ECMAScript程序开始执行时出现，即在引擎初始化阶段就被创建好的对象。这意味着开发者不必明确实例化内置对象，它已被实例化了Global（全局对象）、Math、JSON

基本包装类型对象：ECMAScript还提供了3个特殊的引用类型：Boolean、Number、String。这些类型与其他内置对象类型相似，但同时具有各自的基本类型相应的特殊行为。实际上，**每当读取一个基本类型值得时候，后台就会创建一个对应的基本包装类型的对象，从而让我们能够调用一些方法来操作这些数据。** *包装类型，是一个专门封装原始类型的值，并提供对原始类型的值执行操作的 API 对象*

其他内置对象与基本包装类型对象的区别？

普通的内置对象与基本包装类型的主要区别就是对象的生命期，使用new操作符创建的引用类型的实例，在执行流离开当前作用域之前都一直保存在内存中，而自动创建的基本包装类型的对象，则只是存在于一行代码的执行瞬间，然后立即被立即销毁。这意味着我们不能再运行时为基本包装类型值添加属性和方法。

代码语言: javascript 复制

```
1 | var s1="some text";
2 | s1.color="red";
3 | var s2=new String("some text");
4 | s2.color="red";
5 | console.log(s1.color);//undefined
6 | console.log(s2.color);//red
7 | console.log(s1==s2);//true
8 | console.log(s1===s2);//false
```

在第二行为s1添加一个color属性，第三行代码执行时，再次访问s1，结果s1的color属性被销毁了。详情推荐阅读《[JavaScript内置对象--基本包装类型\(Boolean、Number、String\)详解](#)》

IMWeb前端团队

LV.1
腾讯 前端开发工程师

文章 1.4K 获赞 5.7K

作者相关精选

帮助面向对象开发者理解关
ES6 + Babel + React低版本
前端开发规范之命名规范、r

目录

▶ 什么是原型语言

原型语言创建有两个步骤
js本地对象、内置对象、宿主对象
其他内置对象与基本包装类型对象
原型 - 显式原型 - 隐式原型 - 共
原型对象
对象与函数
原型链是实现继承的主要方法
原型链基本思路：
类 - 对象冒充 - class

添加站长 进交流群
领取 10元无门槛券，专享 最新干货

腾讯云

云开发-个人版

免费体验1个月

立即领取

原生对象 - native object：也叫内部对象、本地对象。独立于宿主环境的ECMAScript实现提供的对象。与宿主无关，在 javascript（远景浏览器）、nodejs（node平台）、jsript（ie浏览器）、typescript（微软平台）等等中均有这些对象。简单来说，本地对象就是 ECMA-262 定义的类（引用类型）。

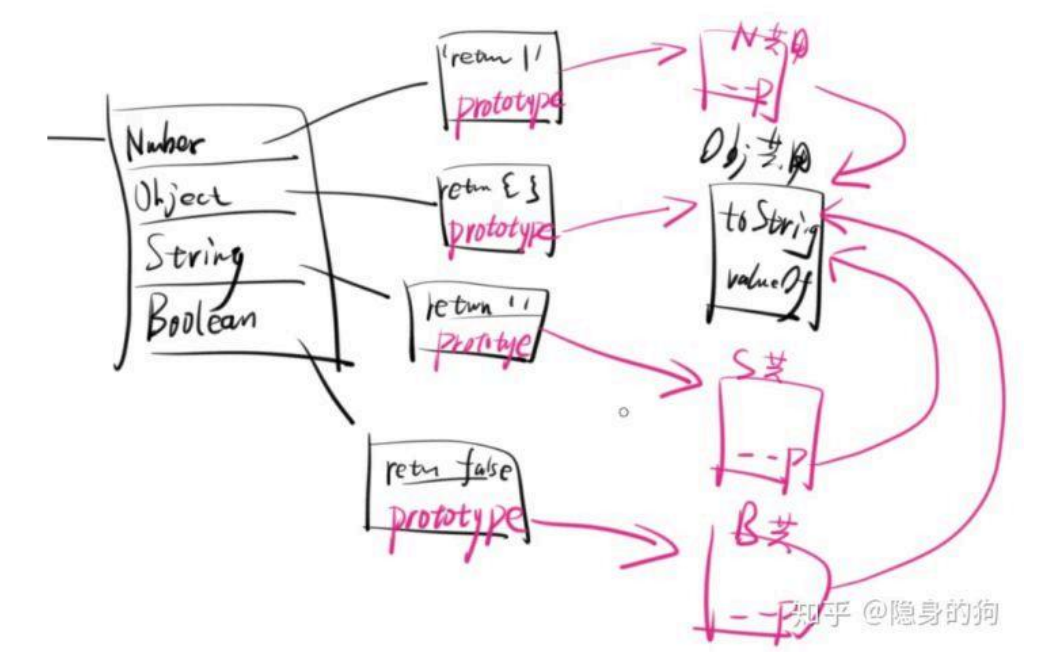
Object、Function、Array、String、Boolean、Number、Date、RegExp、Error、EvalError、RangeError、ReferenceError、SyntaxError、TypeError、URIError、Global

在运行过程中动态创建的对象，需要new，以Number为例：

代码语言： javascript复制

```
1 var n1=Number('1');
2 var n2=1;
3 n2.xxx=2;console.log(n2); //undefined
4 console.log(n1===n2)//false
5 console.log(n1.toString()===n2.toString())//true
6 console.log(n1.__proto__===Number)//false
7 console.log(n2.__proto__===Number)//false
8 console.log(n1.__proto__===n2.__proto__)//true
9 `
```

n1和n2虽然数值都是1，但n2的类型属于'object'，n1则为'number'，身为基本类型number的n1直接指向了数字1，而n2指向了一个地址，这个地址中存放了数值1，这就是对象和基本类型的区别。



但是原型对象只存在于函数对象。也就是本质上只要是通过new Function创建的函数对象会有一个原型对象。而对于其他非Function的引用类型归根结底也是通过new Function创建的。如上面提到的Array类型、Object类型。实际上，在每个函数对象创建的时候，都会自带一个prototype的属性，这个属性相当于一个指针，指向他本身的原型对象，这个原型对象里包含着自定义的方法属性，

代码语言： javascript复制

```
1 function a(){
2     this.name='xiaoming';
3     this.sayName=function () {
4         console.log(this.name);
5     }
6 }
```

在默认情况下，a.prototype下会带有一个constructor属性，这个属性指向创建当前函数对象的构造函数，比如这里constructor指向构造函数a本身也就是说：a.prototype.constructor==a //true 另外默认还有一个proto属性，这个属性指向由创建这个函数对象的引用类型中继承而来的属性和方法。当通过构造函数实例化一个对象b时，即new a();

首先这个new出来的对象属于普通对象，所以没有prototype属性。但他有proto这个属性，这个属性指向创建它的引用类型的原型对象，在这个例子中指向a.prototype,从而继承来自引用类型a的属性和方法。推荐阅读《JS 的 new 到底是干什么的？》

var 对象 = new 函数对象 这个声明形式可以引申出：

代码语言: javascript复制

```
1  函数.__proto__ ===Function.prototype
2  Function.__proto__ === Function.prototype
3  Object.__proto__ === Function.prototype //Objec也是个函数，函数都是由Function构造出来的。
4  Number.__proto__ === Function.prototype
5  构造函数.prototype.__proto__ ===Object.prototype
6  Function.prototype.__proto__ ===Object.prototype
7  Number.prototype.__proto__ ===Object.prototype
8  Object.__proto__ .__proto__ ===null
```

理解了以上的关系后，'proto'是对象的属性、'prototype'是函数的属性这句话也就懂了

null是对象原型链的终点，其值既有（是一个对象）又无（不引用任何对象），代表着对象本源的一种混沌、虚无的状态，正与老子《道德经》中的“道”，有着同等的意义(心中一万只“尼玛奔腾而过，还是写java爽啊)。比如：《undefined与null的区别》

在JS中，undefined是全局对象的一个属性，它的初始值就是原始数据类型undefined，并且无法被配置，也无法被改变。undefined从字面意思上理解为“未定义”，即表示一个变量没有定义其值。

而null是一个JS字面量，表示空值，即没有对象。与undefined相比，null被认为是“期望一个对象，但是不引用任何对象的值”，而undefined是纯粹的“没有值”。

代码语言: javascript复制

```
1  // null为对象原型链的终点
2  console.log(Object.getPrototypeOf(Object.prototype)); // null
3  // null是一个对象
4  console.log(typeof null); // object
5  // null 为空
6  console.log(!null); // true
```

JS中的所有事物都是对象，对象是拥有属性和方法的数据。

为了描述这些事物，JS便有了“原型（prototype）”的概念。

原型模式是js对继承的一种实现：使用原型，能复用代码，节省内存空间 (java类的代码在内存只有一份，然后每个对象执行方法都是引用类的代码，所有子类对象调用父类方法的时候，执行的代码都是同一份父类的方法代码。但是JS没有类，属性和方法都是存在对象之中，根本没有办法做到java那样通过类把代码共享给所有对象)。

推荐阅读《深刻理解JavaScript基于原型的面向对象》

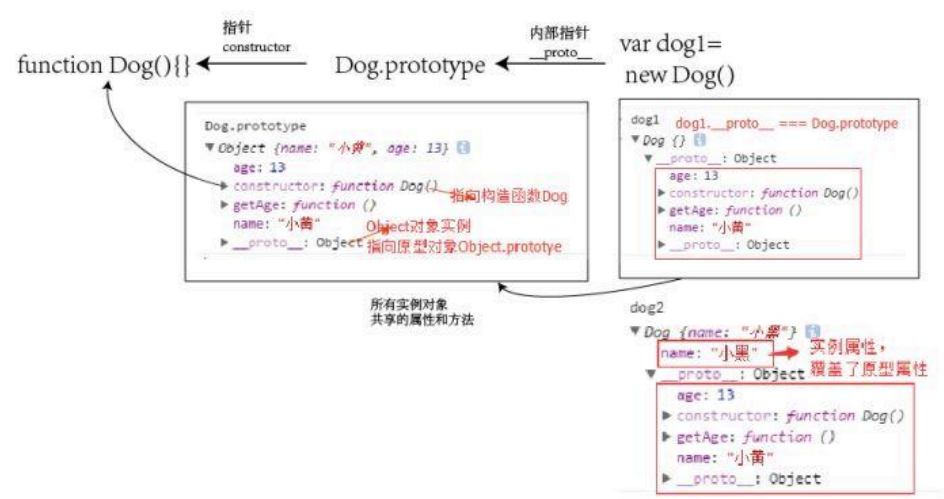
从一张图看懂原型对象、构造函数、实例对象之间的关系



prototype：构造函数中的属性，指向该构造函数的原型对象。

constructor：原型对象中的属性，指向该原型对象的构造函数

proto：实例中的属性，指向new这个实例的构造函数的原型对象



在JavaScript中，每个对象都有一个指向它的原型（prototype）对象的内部链接。这个原型对象又有自己的原型，直到某个对象的原型为 null 为止（也就是不再有原型指向），组成这条链的最后一环。这种一级一级的链结构就称为原型链（prototype chain）。

要清楚原型链,首先要弄清楚对象

普通对象

代码语言： javascript复制

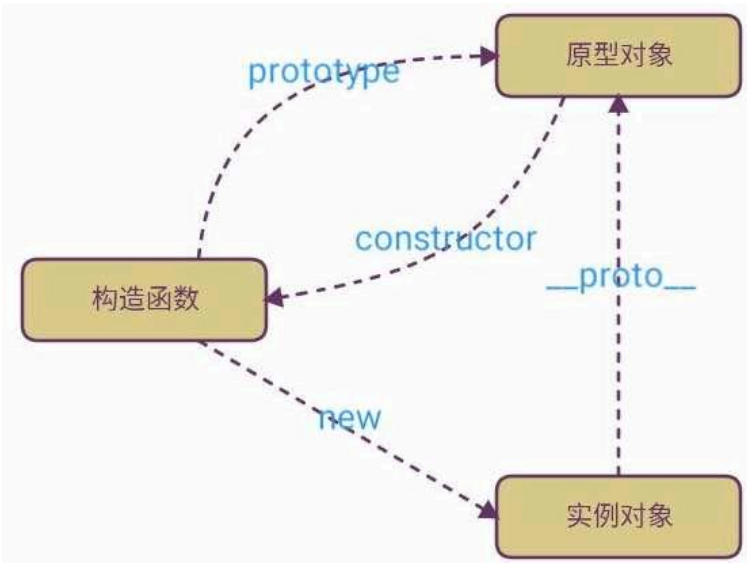
```
1 最普通的对象：有__proto__属性（指向其原型链），没有prototype属性。
2
3 原型对象(Person.prototype 原型对象还有constructor属性（指向构造函数对象）)
```

函数对象：

代码语言： javascript复制

```
1 凡是通过new Function()创建的都是函数对象。
2
3 拥有__proto__、prototype属性（指向原型对象）。
```

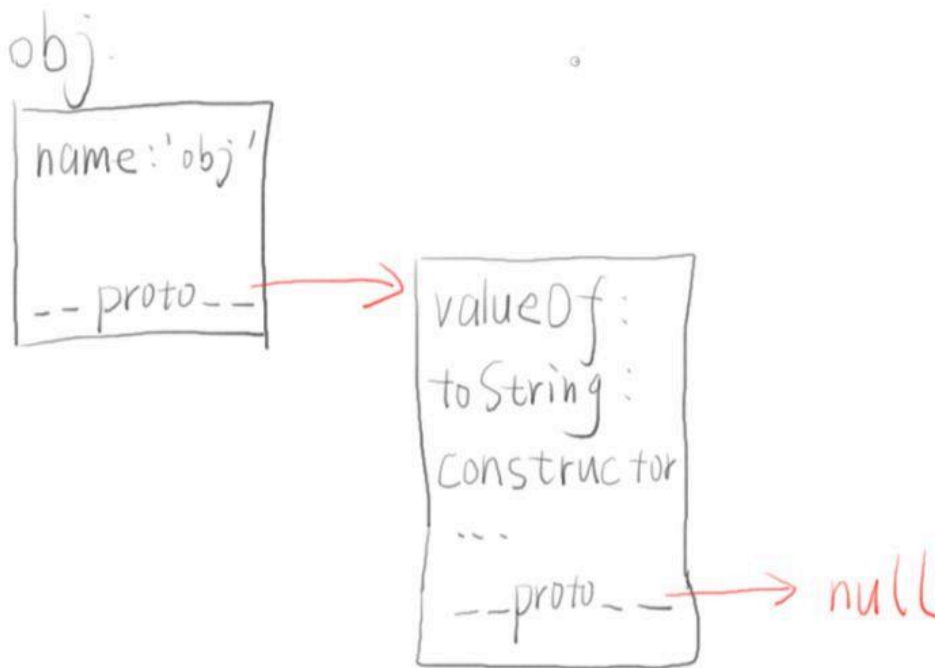
JavaScript对象是动态的属性“包”（指其自己的属性）。JavaScript对象有一个指向一个原型对象的链。当试图访问一个对象的属性时，它不仅仅在该对象上搜寻，还会搜寻该对象的原型，以及该对象的原型的原型，依此层层向上搜索，直到找到一个名字匹配的属性或到达原型链的末尾。



原型 - 显式原型 - 隐式原型 - 共享原型链

显式原型（explicit prototype property）每一个函数在创建之后都会拥有一个名为prototype的属性，这个属性指向函数的原型对象。用来实现基于原型的继承与属性的共享。

隐式原型 (implicit prototype link) JS中任意对象都有一个内置属性`__proto__` (部分浏览器为`[[prototype]]`)，指向创建这个对象的函数（即构造函数）`constructor`的`prototype`。用来构成原型链，同样用于实现基于原型的继承。



当我们「读取」`obj.toString` 时，JS 引擎会做下面的事情：

看看 `obj` 对象本身有没有 `toString` 属性。没有就走到下一步。

看看 `obj.__proto__` 对象有没有 `toString` 属性，发现 `obj.__proto__` 有 `toString` 属性，于是找到了

如果 `obj.__proto__` 没有，那么浏览器会继续查看 `obj.__proto__.__proto__`，如果 `obj.__proto__.__proto__` 也没有，那么浏览器会继续查 `obj.__proto__.__proto__.__proto__`

直到找到 `toString` 或者 `proto` 为 `null`（不管你从那个属性开始，连续引用`proto`的指针，最后输出的那个值就是`null`）。

上面的过程，就是「读」属性的「搜索过程」。

而这个「搜索过程」，是连着由 `__proto__` 组成的链子一直走的。

这个链子，就叫做「原型链」。

要搞清楚 `valueOf` / `toString` / `constructor` 是怎么来的，就要用到 `console.dir` 了。

共享原型链 (Shared prototype chain) 此模式所有子对象及后代对象都共享一个原型（都是通过`b.prototype=a.prototype`；这种模式连接的对象），在这些后代对象上修改原型，会影响所以处在同一共享原型链上的所有对象。而且此模式只继承原型链上的属性和方法，通过`this`定义的属性和方法无法访问和继承

```

▼ Object
  name: "obj"
  ▼ __proto__: Object
    ▶ __defineGetter__: function __defineGetter__()
    ▶ __defineSetter__: function __defineSetter__()
    ▶ __lookupGetter__: function __lookupGetter__()
    ▶ __lookupSetter__: function __lookupSetter__()
    ▶ constructor: function Object()
    ▶ hasOwnProperty: function hasOwnProperty()
    ▶ isPrototypeOf: function isPrototypeOf()
    ▶ propertyIsEnumerable: function propertyIsEnumerable()
    ▶ toLocaleString: function toLocaleString()
    ▶ toString: function toString()
    ▶ valueOf: function valueOf()
    ▶ get __proto__: function __proto__()
    ▶ set __proto__: function __proto__()
  
```

那么 `obj.toString` 和 `obj2.toString` 其实是同一个东西，也就是 `obj2.__proto__.toString`。

这有什么意义呢？

如果我们改写 `obj2.__proto__.toString`，那么 `obj.toString` 其实也会变！

这样 `obj` 和 `obj2` 就是具有某些相同行为的对象，这就是意义所在。

如果我们想让 `obj.toString` 和 `obj2.toString` 的行为不同怎么做呢？

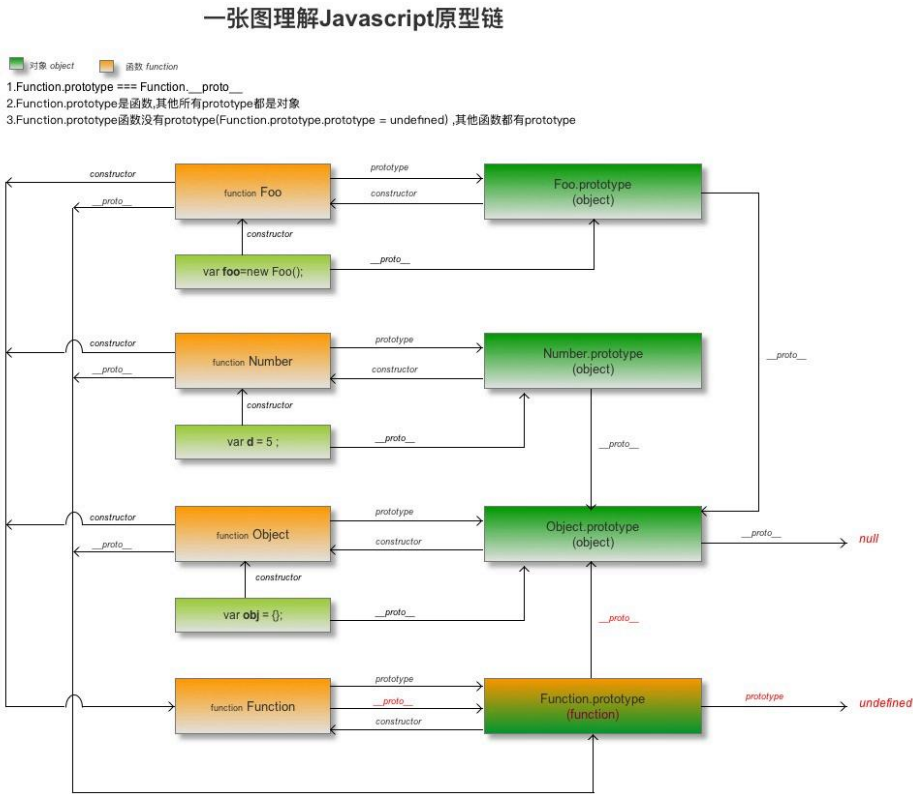
直接赋值就好了：

```
obj.toString = function(){ return '新的 toString 方法' }
```

原型对象

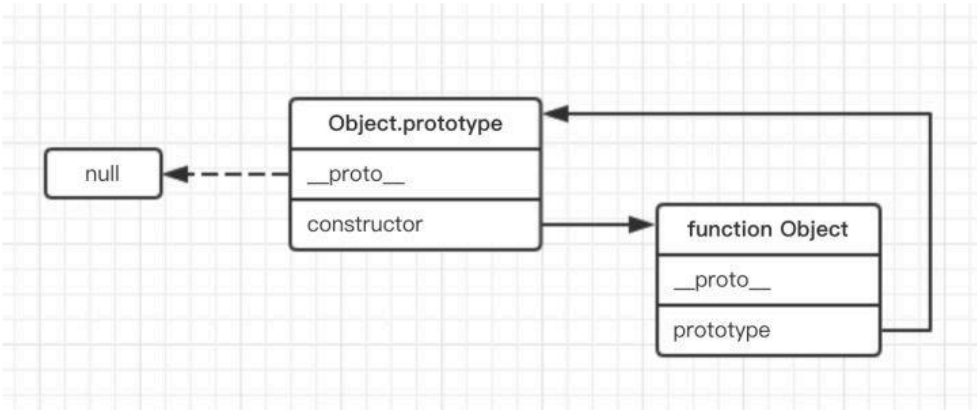
每创建一个函数都会有一个prototype属性，这个属性是一个指针，指向一个对象（通过该构造函数创建实例对象的原型对象）。原型对象是包含特定类型的所有实例共享的属性和方法。原型对象的好处是，可以让所有实例对象共享它所包含的属性和方法。

原型对象属于普通对象。Function.prototype是个例外，它是原型对象，却又是函数对象，作为一个函数对象，它又没有prototype属性。

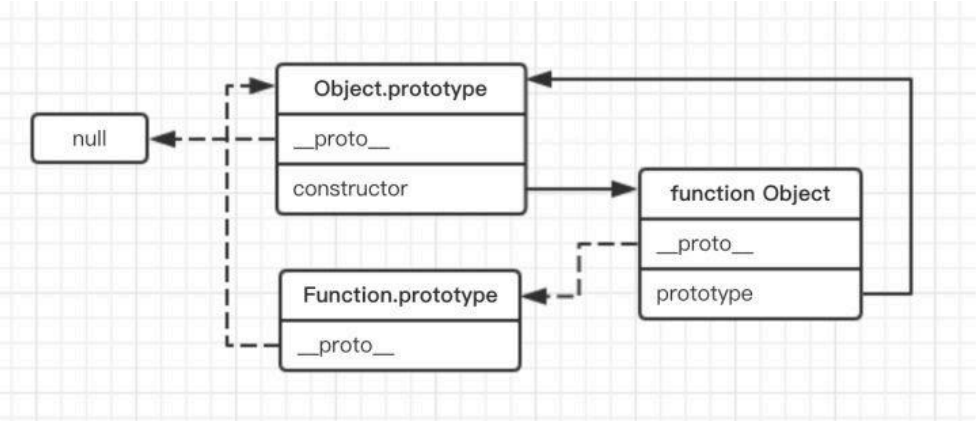


对象与函数

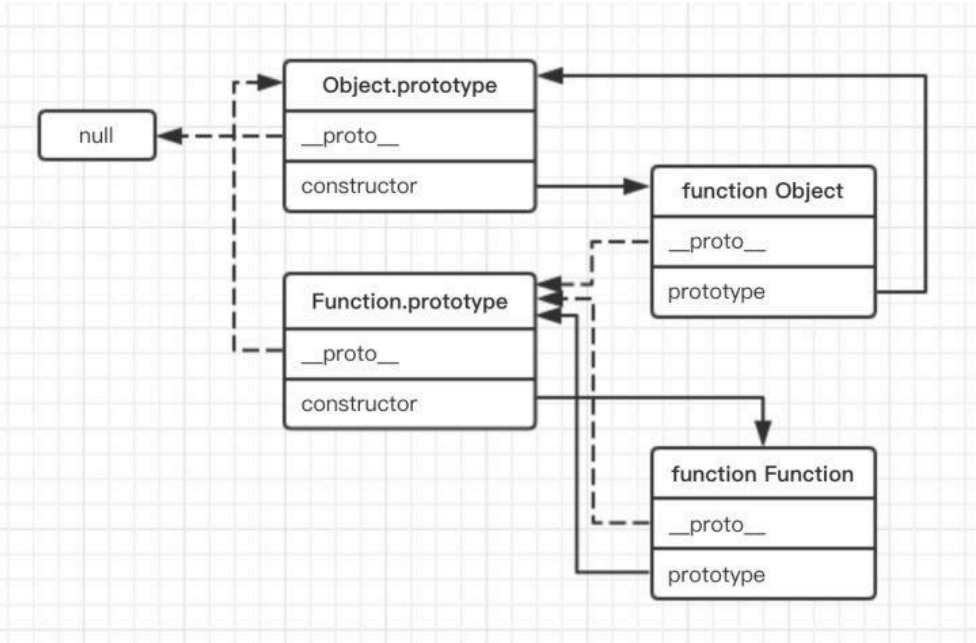
拥有了描述事物的能力，却没有创造事物的能力，显然是不完整的，因此需要一个Object的生成器来进行对象的生成。**JS将生成器以构造函数constructor来表示，构造函数是一个指针，指向了一个函数。**函数（function）函数是指一段在一起的、可以做某一件事的程序。构造函数是一种创建对象时使用的特殊函数。



对象的构造函数function Object同时也是一个对象，因此需要一个能够描述该对象的原型，该原型便是Function.prototype，函数的原型用来描述所有的函数。对象的构造函数的proto指向该原型。



函数的原型本身也是对象，因此其**proto**指向了对象的原型。同样，该对象也需要一个对应的生成器，即其构造函数function Function。



函数的构造函数是由函数生成的一个对象，所以其原型即为函数的原型，其隐式原型也同样为函数的原型Function.prototype。

instanceof操作符的内部实现机制和隐式原型、显式原型有直接的关系。instanceof的左值一般是一个对象，右值一般是一个构造函数，用来判断左值是否是右值的实例。它的实现原理是沿着左值的**proto**一直寻找到原型链的末端，直到其等于右值的prototype为止。

instanceof 的作用是判断一个对象是不是一个函数的实例。比如 obj instanceof fn， 实际上是判断fn.prototype是不是在obj的原型链上。所以

instanceof运算符的实质：用来检测 constructor.prototype是否存在于参数 object的原型链上。

根据上图展示的Object和Function的继承依赖关系，我们可以通过instanceof操作符来看一下Object和Function的关系：

代码语言： javascript复制

```
1 console.log(Object instanceof Object); // true
2 console.log(Object instanceof Function); // true
3 console.log(Function instanceof Object); // true
4 console.log(Function instanceof Function); // true
```

函数与对象相互依存，分别定义了事物的描述方法和事物的生成方法，在生成JS万物的过程中缺一不可。

代码语言： javascript复制

```
1 | Function instanceof Function // true, why? Function.prototype是原型对象，却是函数对象
```

Object特殊在Object.prototype是凭空出来的。语法上，所有的{}都会被解释为new Object();

Function特殊在**proto == prototype**。语法上，所有的函数声明都会被解释为new Function()。

我们来看Function和Object的特殊之处：

- Object是由Function创建的：因为Object.**proto** === Function.prototype；
- 同理，Function.prototype是由Object.prototype创建的；
- Function是由Function自己创建的！
- Object.prototype是凭空出来的！

推荐阅读 《JavaScript内置对象与原型链结构》与《JavaScript中的难点之原型和原型链》

这几句话能解释一切关于原型方面的问题：

- 当 new 一个函数的时候会创建一个对象，『函数.prototype』等于 『被创建对象.**proto**』
- 一切函数都是由 Function 这个函数创建的，所以『Function.prototype === 被创建的函数.**proto**』
- 一切函数的原型对象都是由 Object 这个函数创建的，所以『Object.prototype === 一切函数.prototype.**proto**』

推荐阅读：《对原型、原型链、Function、Object 的理解》

原型链是实现继承的主要方法

先说一下继承，许多OO语言都支持两张继承方式：接口继承、实现继承。

代码语言： javascript复制

- 1 | - 接口继承：只继承方法签名
- 2 | - 实现继承：继承实际的方法

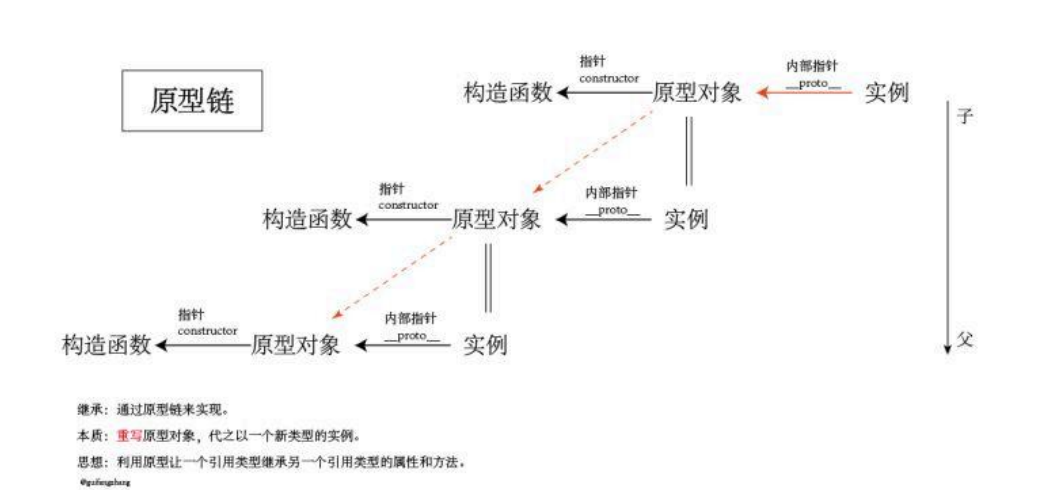
由于函数没有签名，在ECMAScript中无法实现接口继承，只支持实现继承，而实现继承主要是依靠原型链来实现。

原型链基本思路：

利用原型让一个引用类型继承另一个引用类型的属性和方法。

每个构造函数都有一个原型对象，原型对象都包含一个指向构造函数指针(constructor)，而实例对象都包含一个指向原型对象的内部指针(**proto**)。如果让原型对象等于另一个类型的实例，此时的原型对象将包含一个指向另一个原型的指针(**proto**)，另一个原型也包含着一个指向另一个构造函数的指针(constructor)。假如另一个原型又是另一个类型的实例.....这就构成了实例与原型的链条。

原型链基本思路（图解）：



推荐阅读 《JS重点整理之JS原型链彻底搞清楚》

类 - 对象冒充 - class

类（Class）是面向对象程序设计（OOP，Object-Oriented Programming）实现信息封装的基础。类是一种用户定义类型，也称类类型。每个类包含数据说明和一组操作数据或传递消息的函数。**类的实例称为对象。**

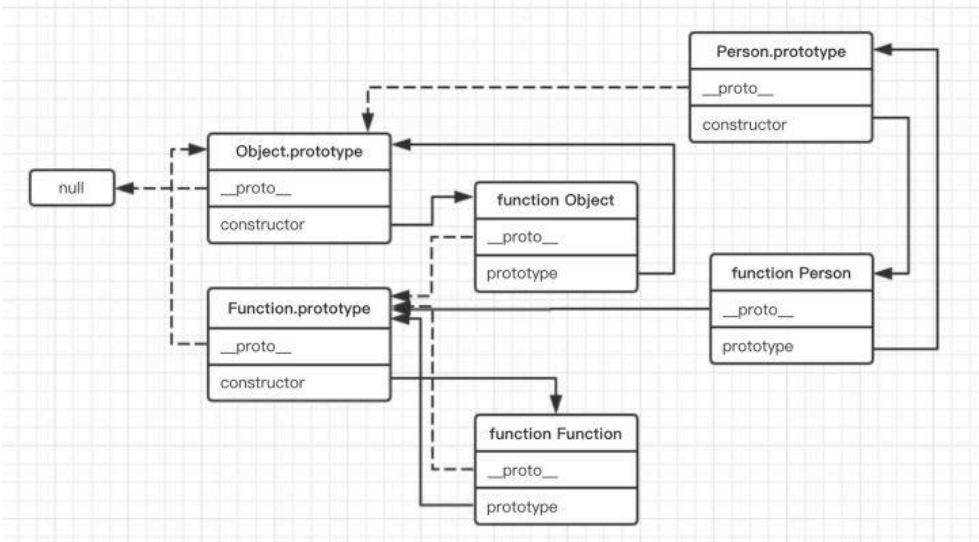
在ECMAScript 2015 中引入的JS类（classes）之前，要在JS中实现类便是采用原型继承的方式。

当把一个函数作为构造函数，使用new关键字来创建对象时，便可以把该函数看作是一个类，创建出来的对象则是该类的实例，其隐式原型**proto**指向的是该构造函数的原型。

在访问该对象的属性或方法时，JS会先搜索该对象中是否定义了该属性或方法，若没有定义，则会回溯到其**proto**指向的原型对象去搜索，若仍然未搜索到，则会继续回溯该原型的原型，直到搜索到原型链的终点null；

这种特性可以理解为：**构造函数生成的实例，继承于该构造函数的原型。**

得益于这种特性，我们可以使用定义构造函数的方式来定义类。



代码语言： javascript

复制

```
1 function Person() {} // 定义Person构造函数
2 // 通常以大写字母开头来定义类名
3 console.log(new Person() instanceof Person); // true
```

以上定义了Person类，该构造函数是由Function构造而来，所以其隐式原型指向函数的原型，而为了描述该事物，同时生成了该类的原型Person.prototype，该原型又是由Object构造而来，所以其隐式原型指向了对象的原型。

后记：文字有点乱，就是多篇文章的精华提炼。发现把一个自己懂的事情，深入浅出讲明白，并非易事。文有不妥之处，请留言告知，谢谢。

参考文字：

js高级--本地对象、内置对象、宿主对象 <https://www.cnblogs.com/flyings/p/7079829.html>
js原型与原型链 <https://zhuanlan.zhihu.com/p/42963985>
道生万物—理解Javascript原型链 <https://zhuanlan.zhihu.com/p/31822475>
「每日一题」什么是 JS 原型链？ <https://zhuanlan.zhihu.com/p/23090041>
JS理解原型、原型链 <https://zhuanlan.zhihu.com/p/23036843>
一张图弄清Javascript中的原型链、prototype、__proto__的关系 <https://zhuanlan.zhihu.com/p/22784477>
js中的原型、原型链、继承模式 <https://zhuanlan.zhihu.com/p/32194154>
说说原型（prototype）、原型链和原型继承 <https://zhuanlan.zhihu.com/p/35790971>
浅谈JS原型和原型链 <https://zhuanlan.zhihu.com/p/23026595>
原型语言解释 <https://blog.csdn.net/rogerjava/article/details/51283572>
基于类 vs 基于原型 <https://blog.csdn.net/prehistorical/article/details/53671415>

原文：https://www.zhoulujun.cn/html/webfront/ECMAScript/js/2015_0715_119.html，文有不妥之处，请源站留言告知，拜谢！

本文参与 腾讯云自媒体分享计划，分享自作者个人站点/博客。

原始发表：2019-05-17，如有侵权请联系 cloudcommunity@tencent.com 删除

前往查看

java 编程算法 面向对象编程 https 网络安全

评论

登录 后参与评论

推荐阅读

编辑精选文章

换一批

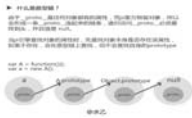
眼看他搭中台，眼看他又拆了	6448	QQ 25年技术巡礼 技术探索下的清...	1004
【万字长文】论如何构建一个资金账...	3183	我攻克的技术难题 - 因为一部遮天， ...	2254
得物 Redis 设计与实践	1976	【万字长文】K8s部署前后端分离we...	4016

关于javascript的原型和原型链，看我就够了（三）

编程算法 javascript

原型对象有一个constructor属性，指向该原型对象对应的构造函数 foo 为什么有 constructor 属性？ 那是因为 foo 是 Foo 的实例。 那 Foo.prototype 为什么有 constructor 属性？ ？ 同理， Foo.prototype...

陌上寒 2019-04-02 453 0



Javascript之其实我觉得原型链没有难的那么夸张！

javascript 数据结构

原型链、闭包、事件循环等，可以说是js中比较复杂的知识了，复杂的不是因为它的概念，而是因为它们本身都涉及到很多的知识体系。所以很难串联起来，有一个完整的思路。我最近想把js中有点意思的知识都总结整理一下，虽然逃不开一些一模一样...

zaking 2020-08-18 690 0

原型和原型链的深入浅出

javascript 编程算法 面向对象编程

对象是 javascript 基本数据类型。对象是一种复合值： 它将很多值（原始值或者其它对象）聚合在一起，可通过名字访问这些值。

前端老鸟 2022-03-07 370 0

JavaScript原型链与继承

面向对象编程 编程算法

只要是对象，一定有原型对象，就是说只要这个东西是个对象，那么一定有proto属性。（错的）


用户7043603 2022-02-26 1.5K 0

第202天：js---原型与原型链终极详解

其他

JavaScript 中，万物皆对象！但对象也是有区别的。分为普通对象和函数对象， Object 、Function 是 JS 自带的函数对象。下面举例说明

半指温柔乐 2018-09-10 905 0




JS 原型链

javascript

Function 是JavaScript 里最顶层的构造器，它构造了系统中的所有对象，包括定义对象、系统内置对象、甚至包括它自己。

用户8921923 2022-10-24 2.3K 0



彻底搞懂JS原型与原型链

面向对象编程 javascript

说到JavaScript的原型和原型链，相关文章已有不少，但是大都晦涩难懂。本文将换一个角度出发，先理解原型和原型链是什么，有什么作用，再去分析那些令人头疼的关系。

hellocoder2029 2022-10-17 1.3K 0

详解原型与原型链

编程算法 javascript

其实，刚开始学 JavaScript 时，就有学过原型与原型链的相关知识了，只是当时还没有养成写笔记的习惯，导致现在已经忘的七七八八了。



赤蓝紫 2023-01-02 365 0

instanceof运算符的实质：Java继承链与JavaScript原型链

javascript

instanceof 严格来说是Java中的一个双目运算符，用来测试一个对象是否为一个类的实例

周陆军 2021-08-15 476 0

【JS】479- 又见原型和原型链

javascript objective-c

在前端这块领域，原型与原型链是每一个前端er必须掌握的概念。我们多次在面试或者一些技术博客里面看见这个概念。由此可见，这个玩意对于前端来说有多重要。其实它本身理解起来不难，但是很多刚入行前端的同学，看到prototype、__proto__理解...

pingan8787 2020-02-17 647 0

前端day18-JS高级(完整的原型链)学习笔记

编程算法 面向对象编程

01-面向对象三大特征（封装、继承、多态） a.封装：将某个具体功能封装在对象中，只对外部暴露指定的接口，外界在使用的时候，只考虑接口怎么用，不用考虑内部怎么实现（前面学习的api其实就...

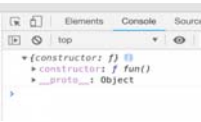


帅的一麻皮 2020-05-07 463 0

关于javascript的原型和原型链，看我就够了（二）

编程算法 javascript

Object就是一个构造函数，是js内置的构造函数.上面的例子中Object就是obj的构造函数，这个例子似乎不太明显，我们继续看



陌上寒 2019-04-02 455 0

JavaScript原型和原型链（ prototype 与 __proto__ ）

javascript

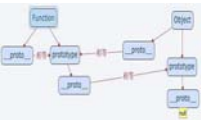
var a = new A; //a 类型： 对象 //A 类型： 函数 var Object = new Function(); //var 对象 = new 函数; Object.__proto__ === Function.prototype; //对象.__proto__ === 函数.prototype; Function.__proto__ === Function.prototype; //因为 Function 也是...

Leophen 2019-08-23 765 0

[我的理解]Javascript的原型与原型链

java ecmaascript json

一、原型与原型链的定义 原型：为其他对象提供共享属性的对象 注：当构造器创建一个对象，为了解决对象的属性引用，该对象会隐式引用构造器的"prototype"属性。程序通过constructor.prototype...



sam dragon 2018-01-16 817 0

一张图带你搞懂Javascript原型链关系

javascript

为了更好的图文对照，我为每条线编了标号，接下来的细节讲解，都会用到这张图里的编号：

xing.org1^ 2021-08-09 674 0

领券

一文带你彻底搞懂JavaScript原型链

javascript
Brendan Eich(布兰登·艾奇) 作为JavaScript的作者曾说过 “它是C语言和Self语言一夜情的产物。”



【前端基础进阶】JS原型、原型链、对象详解

javascript 编程算法
在上面的例子中 o1 o2 o3 为普通对象， f1 f2 f3 为函数对象。怎么区分，其实很简单，凡是通过 new Function() 创建的对象都是函数对象，其他的都是普通对象。f1,f2,归根结底都是通过 new Function...



js中的原型和原型链

javascript 前端
在js中每个函数（非箭头函数，一般关于原型的有关知识我们都只考虑构造函数）都会拥有一个 prototype 属性，该属性值是一个对象，我们称之为原型对象。原型对象上默认会有 constructor 属...



JavaScript继承和原型链

面向对象编程 javascript ecmaScript 编程算法
JS在加载构造函数时，会在内存中生成一个对象，这个对象称为函数的原型对象(prototype)。

JavaScript学习总结(四)——this、原型链、javascript面向对象

javascript
根据题目要求，对给定的文章进行摘要总结。



社区	活动	资源	关于
专栏文章	自媒体分享计划	技术周刊	社区规范
阅读清单	邀请作者入驻	社区标签	免责声明
互动问答	自荐上首页	开发者手册	联系我们
技术沙龙	技术竞赛	开发者实验室	友情链接
技术视频			
团队主页			
腾讯云TI平台			

热门产品	域名注册 云存储	云服务器 视频直播	区块链服务	消息队列	网络加速	云数据库	域名解析
热门推荐	人脸识别 SSL 证书	腾讯会议 语音识别	企业云	CDN加速	视频通话	图像分析	MySQL 数

Copyright © 2013 - 2024 Tencent Cloud. All Rights Reserved. 腾讯云 版权所有
深圳市腾讯计算机系统有限公司 ICP备案/许可证号：粤B2-20090059 深公网安备号 44030502008569
腾讯云计算（北京）有限责任公司 京ICP证150476号 | 京ICP备11018762号 | 京公网安备11010802020287

