第六课 原型与原型链

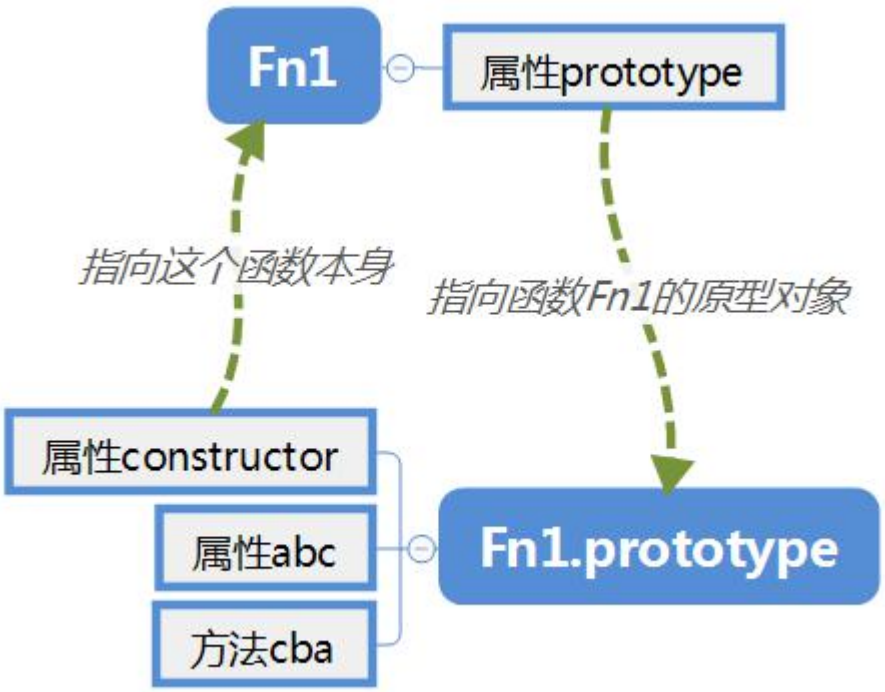
学习目录

- **prototype 与__proto__**
- **原型链**

一．prototype 与__proto__

prototype 一般称为显式原型，__proto__一般称为隐式原型。

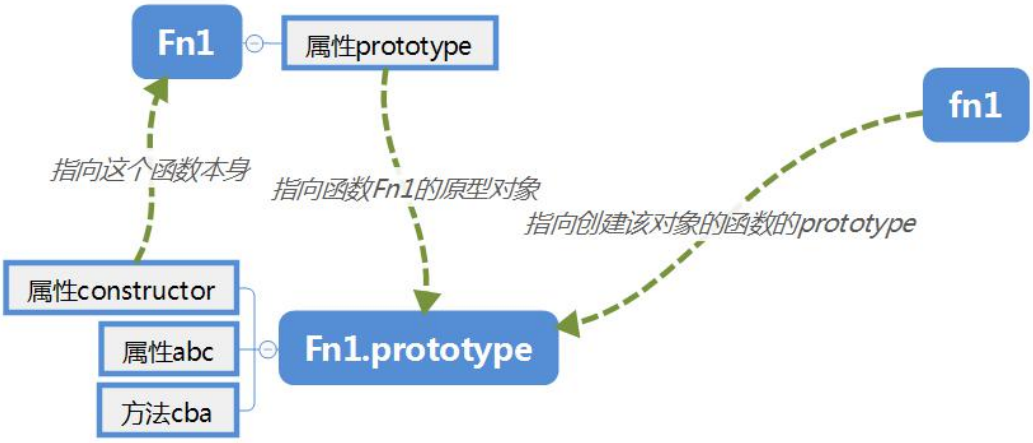
每一个函数在创建之后，在默认情况下，会拥有一个名为 prototype 的属性，这个属性表示函数的原型对象，同时这个原型对象本身有一个 constructor 的属性，指向这个函数本身。



除此之外，每个 js 对象都有一个隐藏的原型属性——“__proto__”，我们可以发现

fn1.__proto__ == Fn1.prototype，由此可以说明隐式原型表示创建这个对象的函数的prototype。

你会发现 fn1.proto 和 Fn1.prototype 的属性与方法一样，因为 fn1 这个实例对象本质上是被 Fn1 这个函数创建的，因此 fn1.proto == Fn1.prototype。



二．原型链

当我们访问一个对象的属性时，js 会先在这个对象定义的属性中进行查找，如果没有找到，就会沿着__proto__这个隐式原型关联起来的链条向上一个对象查找，这个链条就是原型链。

```
function Fn1(){};

Fn1.prototype.abc = '星星课堂';

Fn1.prototype.cba = function(){};


var fn1 = new Fn1();

fn1.xyz = 100;

console.log(fn1.xyz);//100
```

```
console.log(fn1.abc);//星星课堂，这个星星课堂是 fn1 从 Fn1 继承来的
```

fn1 是 Fn1 函数 new 出来的实例对象，fn1.xyz 是 fn1 这个实例对象的属性，fn1.abc 是从 Fn1.prototype 得来，因为 fn1 的__proto__隐式原型指向的是 Fn1.prototype 这个 Fn1 的原型对象。原型链某种意义上是让一个引用类型继承另一个引用类型的属性和方法。

这种获取到上一层对象属性和方法的过程称之为继承，在 js 中继承一般都是实现继承，而且绝大多数是必须依靠原型链来做到的。

函数中的属性来源

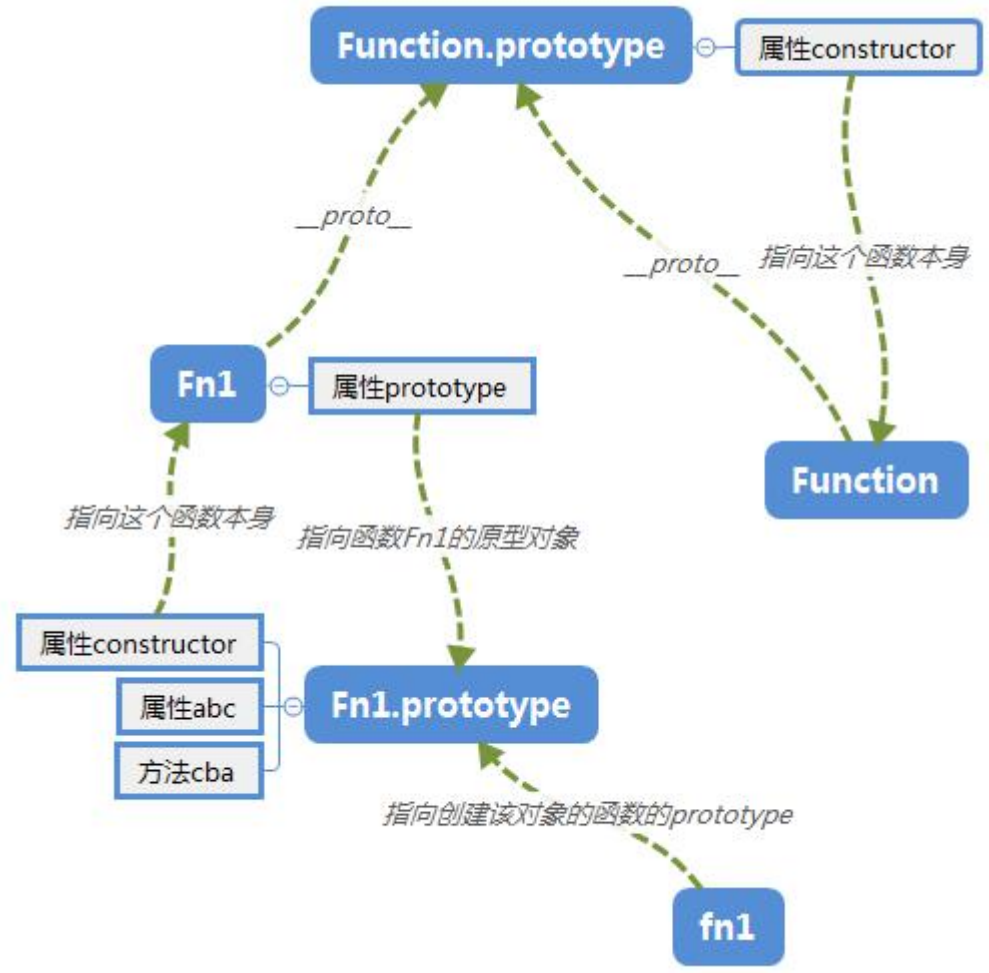
```
function fn2(a,b){
    console.log(arguments);
    return a + b;
}
fn2(1,2);
console.log(fn2.length);
```

每个声明出来的函数都有 length 参数长度的属性，也可以调用 call、bind 等函数特有的方法，并且在函数内部都有 arguments 参数集合等属性。

因为每个函数，本质上都是 Function 这个函数类型创建出来的，因此这些属性和方法其实都是通过原型链由 Function 函数继承来的，按照之前的介绍，被实例化出来的对象身上的属性和方法是由 Function.prototype 原型对象上继承来的属性和方法。

因此 Function 身上的属性和方法也是由 Function.prototype 原型对象上继承来的。

```
console.log(Function.__proto__ == Function.prototype);
```



区别属性和方法来源

我们可以通过对象身上的 hasOwnProperty 方法来发现对象的某个属性是自身定义的还是通过原型链继承来的。

console.log(fn1.hasOwnProperty('xyz'));//true

console.log(fn1.hasOwnProperty('abc'));//false

那这个 hasOwnProperty 在 fn1 身上没有 ,在 Fn1 身上也没有定义 ,在 Fn1.prototype 身上也没有定义，实际上这个方法是从 Object.prototype 身上继承来的。

之前说过 js 中一切皆对象，虽然不是特别严谨，但是基本的思路是对的，对于理解原型链也是特别好的参考依据，所有的对象都会按照原型链找到 Object.prototype，因此所有的对象都会继承来自 Object.prototype 的属性和方法。

```
function Fn1(){}
var fn1 = new Fn1();
console.log(Fn1.prototype.hasOwnProperty);
console.log(fn1.toString());//[object Object]
console.log(fn1.constructor);//ƒ Fn1(){}
```

只要某个对象 a 检测引用类型来源另一个对象 b，就说明 a 这个对象的__proto__指向 b 这个对象，同时 a 这个对象的属性方法也是继承自 b 这个对象。

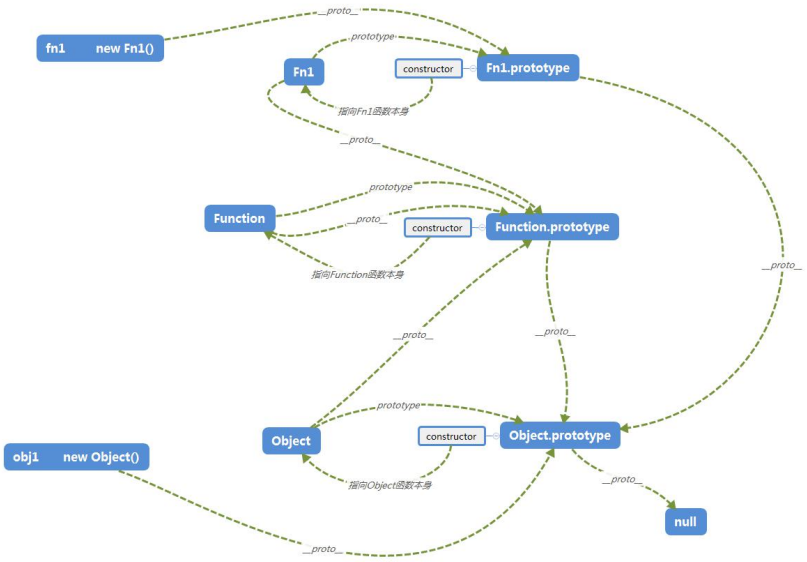
```
console.log(Object instanceof Function);//true
console.log(Function instanceof Object);//true
```

可以看出，所有的对象最终都会继承来自 Object.prototype 的属性和方法，那在来看下 Object.prototype.__proto__指向哪里，它的属性方法来自哪里。

```
console.log(Object.prototype.__proto__);//null
```

结合之前我们学习到的显式原型、隐式原型、引用类型检测等知识，因此一条完整的原型链结构就能理出来了。利用原型链的特性，我们就能实现对象与对象之间的继承，有了继承，我们就可以编写出可以复用的组件，多个子组件从同一个父组件中继承已有的属性和方法，在扩展自己的属性和方法，这样就能灵活的编写出各种组件了。

原型链



谢谢观看！

我是星星课堂老师：周小周