**什么是原型？**

我们所创建的每一个函数，解析器都会向函数中添加一个属性prototype 属性的值为一个空对象 {}

每一个函数都有一个自己的唯一的prototype

这个属性对应着一个对象，这个对象就是所谓的原型对象prototype

**对象中的原型是何时被添加的？**

**1.**如果函数作为普通函数调用，有prototype属性，但是没什么意义

function fn() {

console.log(fn.prototype.age)

}

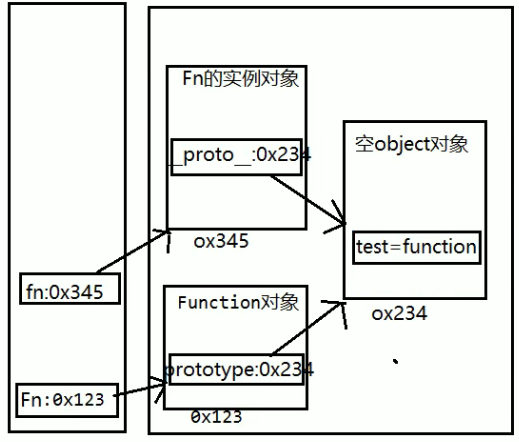
fn.prototype.age = 11

fn()//可以输出11

**2.**当函数以构造函数的形式调用时，它所创建的每一个实例化对象中都会被添加一个隐含的属性 是在new的过程中添加的

实例化对象的\_\_proto\_\_属性===构造函数的prototype属性

两者存放的地址值指向同一个对象 即构造函数的原型对象



**为什么要使用原型？**

每次创建一个实例对象的时候，内部又会为每一个方法或者属性独立开辟一个新的内存空间，如果实例化对象过多，占用内存就越大，就采用原型来解决（构造函数的属性和方法共享）

原型对象就相当于一个公共的区域，所有同一个类的实例都可以访问到这个原型对象，
我们可以将对象中共有的内容，统—设置到原型对象中，这样不用分别为每一个对象添加，也不会影响到全局作用域，就可以使每个对象都具有这些属性和方法了

**理解：**

new一个A类的对象B

不论是A中的prototype还是B中的\_\_proto\_\_都指向A类构造函数中的prototype原型对象

//向MyClass的原型中添加属性a

Myclass.prototype.a = 123;

//向MyClass的原型中添加一个方法

Myclass.prototype.sayHello = function( ){

alert( "hello");

};

**使用in检查对象中是否含有某个属性时，只管有或者没有**

如果对象中没有但是原型中有，也会返回true

用途：判断是否存在某个属性

语法：属性 in 对象

console.log( "name" in mc ) ;

**使用对象的hasOwnProperty(()来检查对象自身中是否含有该属性**

console.log(mc .hasOwnProperty("age"));

**原型链**

**原型对象也是对象,所以它也有原型**

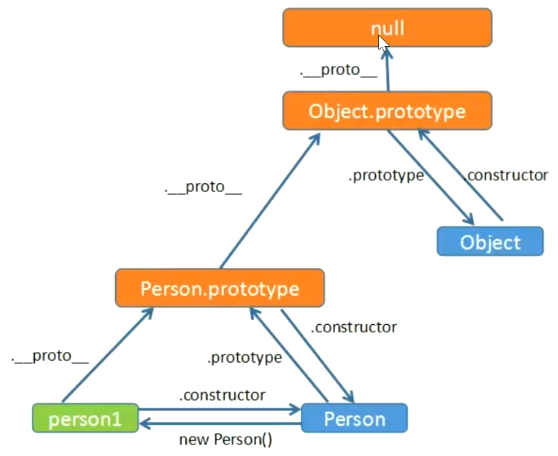
当我们使用一个对象的属性或方法时，会现在自身中寻找，

自身中如果有，则直接使用，

如果没有则通过\_\_proto\_\_去原型对象(的构造函数prototype)中寻找，如果原型对象的构造函数中有，则使用，

如果没有则去原型的原型中寻找，直到找到Object对象的原型

Object对象的原型没有原型，值为null，如果在Object对象中依然没有找到，则返回undefined



**理解：**

当我们通过实例化对象去调用原型中的方法或者获取原型中的属性时，通过\_\_proto\_\_访问构造函数的原型

当我们需要给构造函数的原型添加公共的方法时，通过构造函数.prototype来访问原型添加方法

ES6之前程序员不能直接操作隐式原型 现在可以

**原型：盒子拖拽**

写法一：

function DragBox(el) {

this.el = document.querySelector(el)

}

DragBox.prototype = {

constructor: DragBox,

dragStart() {

this.el.onmousedown = (e) => {

let disX = e.pageX - this.el.offsetLeft

let disY = e.pageY - this.el.offsetTop

this.dragIng({ x: disX, y: disY })

this.dragEnd()

}

},

dragIng({ x, y }) {

document.onmousemove = (e) => {

this.el.style.left = e.pageX - x + "px"

this.el.style.top = e.pageY - y + "px"

}

},

dragEnd() {

document.onmouseup = () => {

document.onmousemove = document.onmouseup = null

}

},

}

写法二：

function DragBox(el) {

this.el = document.querySelector(el)

}

DragBox.prototype.dragStart = function () {

this.el.onmousedown = (e) => {

let disX = e.pageX - this.el.offsetLeft

let disY = e.pageY - this.el.offsetTop

this.dragIng({ x: disX, y: disY })

this.dragEnd()

}

}

DragBox.prototype.dragIng = function ({ x, y }) {

document.onmousemove = (e) => {

this.el.style.left = e.pageX - x + "px"

this.el.style.top = e.pageY - y + "px"

}

}

DragBox.prototype.dragEnd = function () {

document.onmouseup = () => {

document.onmousemove = document.onmouseup = null

}

}

**原型**

**1.什么是原型?**

原型, 英文名prototype是函数中一个自带的属性,我们创建的每个函数都有一个 prototype(原型)属性,这个属性是一个对象.

**2.原型的作用?**

原型的作用是: 可以让同一个构造函数创建的所有对象共享属性和方法. 也就是说, 你可以不在 构造函数中定义对象的属性和方法, 而是可以直接将这些信息添加到原型对象中。

**3.原型的好处?**

优点: 1, 实例对象都有自己的独有属性 2, 同时共享了原型中的方法,最大限度的节省了内存 3, 支持向构造函数传递参数 (初始值)

**4.什么是原型链?**

当访问一个对象的某个属性时，会先在这个对象本身属性上查找，如果没有找到，则会去它的\_\_proto\_\_隐式原型上查找，即它的构造函数的prototype，如果还没有找到就会再在构造函数的prototype的\_\_proto\_\_中查找，这样一层一层向上查找就会形成一个链式结构，我们称为原型链

