基础进阶(柯里化,面向对象,jQuery)

2020年1月14日

10:18

* 1. **柯里化**

1.1 什么是柯里化?

柯里化是指这样一个函数,他接受函数A作为参数,运行后能返回一个新的函数.并且这个函数能处理函数A的剩余参数.

1.2 例子

函数A

function A(a, b, c) {

// do something

}

柯里化通用函数createCurry

var \_A = createCurry(A)

以下结果均等价

\_A(1, 2, 3);

\_A(1, 2)(3);

\_A(1)(2, 3);

\_A(1)(2)(3);

A(1, 2, 3);

**\_A即A的柯里化函数**

**1.3 柯里化通用封装函数**

// 简单实现，参数只能从右到左传递

function createCurry(func, args) {

var arity = func.length;

var args = args || [];

return function() {

var \_args = [].slice.call(arguments);

[].push.apply(\_args, args);

// 如果参数个数小于最初的func.length，则递归调用，继续收集参数

if (\_args.length < arity) {

return createCurry.call(this, func, \_args);

}

// 参数收集完毕，则执行func

return func.apply(this, \_args);

}

}

使用 :

var \_add = createCurry(add,[1])//注意第二个参数是数组

\_add(1)(2)

//实现类似于add(10,10)(20) = 40 , add(10)(10) = 20 的不定参数柯里化

function add() {

let args = [].concat(...arguments);

let curry = function () {

args.push(...arguments);

return curry;

};

curry.toString = function () {

try {

return args.reduce(function (a, b) {

return a + b;

})

} catch (e) {

}

};

return curry;

}

* 1. **面向对象**

**2.1 通过构造函数生成对象**

var li = new Person('li');

**关于构造函数返回值的影响**

--构造函数Person没有返回值,或者返回值为基本数据类型和null,则new 返回值是个Person的实例

--构造函数Person返回值为object,function,new返回值是构造函数的返回值

**2.2 通过原型创建共有方法和属性**

Person.prototype.getName = function() {

return this.name;

}

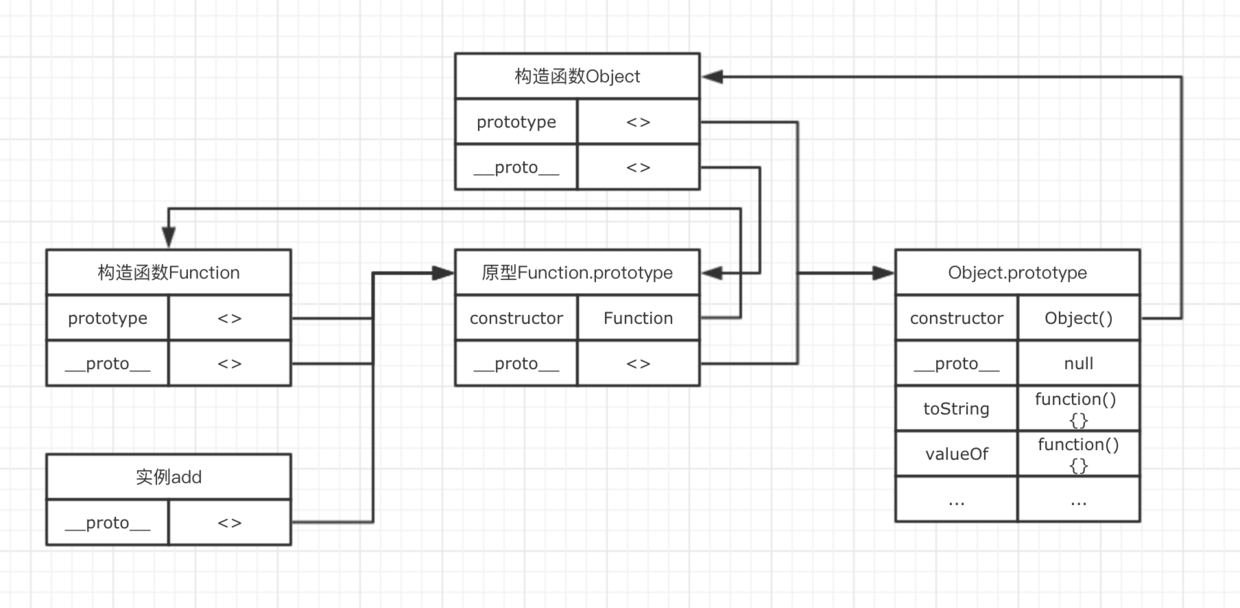
判断对象是否有某个属性

'name' in p1

**2.3 继承**

随意声明一个函数 function add() {}

如下图表示这个函数的原型链



**构造函数的继承**

function cPerson(name, age, job) {

Person.call(this, name, age);

this.job = job;

}

**原型的继承**

//Student原型的原型设置为Person的原型,实现对Person的继承

Student.prototype.\_\_proto\_\_ = Person.prototype

**对象属性的属性类型**

**--设置属性类型**

// 并设置name的属性类型的configurable为false，表示不能再用delete删除

Object.defineProperty(person, 'name', {

configurable: false,

value: 'Jake' , // 设置name属性的值

enumerable: false, //不可枚举

writable: false,//不可修改

get: function() {

return 'TOM'

},

set: function(value) {

// 设置name属性时，返回该字符串，value为新值

console.log(value + ' in set');

}

})

**--读取属性的特性值**

Object.getOwnPropertyDescriptor

//读取属性的特性值

var descripter = Object.getOwnPropertyDescriptor(person, 'name');

console.log(descripter); // 返回结果如下

descripter = {

configurable: false,

enumerable: false,

value: 'alex',

writable: false

}

* 1. **面向对象实战之拖曳对象封装**

* 1. **浅析jQuery**