# InstanceOf

/\*\*

\* @description 判断对象是否属于某个构造函数

\* @param left 实例对象

\* @param right 构造函数

\* @return void

\* @status public

\*/

function instanceof(left, right)

{

    let rightPrototype = right.prototype; // 获取构造函数的显式原型

    let leftProto = left.\_\_proto\_\_; // 获取实例对象的隐式原型

    while (true)

    {

        // 说明到原型链顶端，还未找到，返回 false

        if (leftProto === null) {

            return false;

        }

        // 隐式原型与显式原型相等

        if (leftProto === rightPrototype) {

            return true;

        }

        // 获取隐式原型的隐式原型，重新赋值给 leftProto

        leftProto = leftProto.\_\_proto\_\_;

    }

}

# 继承

## 原型链继承

//父类型

function Super1() {

    this.supProp = "Super property";

}

//给父类型的原型添加方法

Super1.prototype.showSuperProp = function () {

  console.log(this.supProp);

};

//子类型

function Sub() {

    this.subProp = "sub property";

}

//创建父类型的对象赋值给子类型的原型

Sub.prototype = new Super1();

//将子类型原型的构造属性设置为子类型

Sub.prototype.constructor = Sub;

//给子类型的原型添加方法

Sub.prototype.showSubProp = function () {

    console.log(this.subProp);

};

var sub = new Sub();//创建子类型的对象

sub.showSuperProp();//Super property,可以调用父类型的方法

sub.showSubProp();//sub property,也可以调用自己的方法

## 借用构造函数继承（假的继承）

//父类型

function Person(name, age) {

    this.name = name;

    this.age = age;

}

//子类型

function Student(name, age, sex) {

    //在子类型构造函数中调用父类型构造

    Person.call(this,name,age);

    //上面语句相当于this.Person(name,age)，与下两条注释语句相同

    // this.name = name;

    // this.age = age;

    this.sex = sex;

}

var s = new Student("张三",20,"男");

console.log(s.name,s.age,s.sex);// 张三 20 男

## 组合继承

//父类型

function Person(name, age) {

    this.name = name;

    this.age = age;

}

//给父类型的原型添加方法

Person.prototype.setName = function (name) {

    this.name = name;

};

//子类型

function Student(name, age, sex) {

    //在子类型构造函数中调用父类型构造，为了得到属性

    Person.call(this,name,age);

    //上面语句相当于this.Person(name,age)，与下两条注释语句相同

    // this.name = name;

    // this.age = age;

    this.sex = sex;

}

//子类型的原型等于父类型的实例,不用传参，为了看到父类型的方法

Student.prototype = new  Person();

//将子类型原型的构造属性设置为子类型,修正constructor属性

Student.prototype.constructor = Student;

//给子类型的原型添加方法

Student.prototype.setSex = function (sex) {

    this.sex = sex;

};

var s = new Student("张三",20,"男");

s.setSex("女");

s.setName("李四");

console.log(s.name,s.age,s.sex);// 李四 20 女