**面向对象**

1. 基础

function Parent() {}

Parent.prototype = {

p1: function(){}

}

1、构造函数的继承：

(1) 对象冒充

function Child() {

this.temp = Parent;

this.temp()

delete this.temp;

}

(2) call

function Child () { Parent.call(this) }

2、原型链的继承

方法一：在继承了Parent.prototype的同时，也继承了this

Child.prototype = new Parent ();

方法二：只继承了Parent.prototype。可以通过call继承this

Child.prototype = Parent.prototype;

方法三：生成与方法一类似的原型链对象结构，但没有继承this。 IE11以下均不支持。

Child.prototype = { \_\_proto\_\_: Parent.prototype };

可以通过 if(Object.\_\_proto\_\_) 判断浏览器是否支持该属性

3、静态属性的继承

for (var p in Parent){

if(Parent.hasOwnProperty(p) && p !== 'prototype'){

Child[p] = Parent[p]

}

}

4、通过instanceof判断构造函数

var c = new Child;

在继承了父对象原型链的情况下（以上原型链继承的任意方式），

c instanceof Parent 返回 true

5、constructor 返回实例的构造函数

没有继承的情况下，c.constructor返回 function Child(){}

在继承了父对象原型链的情况下，c.constructor 返回 function Parent(){}

可通过constructor获取构造函数，调用构造函数的静态属性

1. 进阶

实现一个class类

功能：通过属性的方式实现对父类的继承

1. es6对象定义与继承