1. 通过 class 关键字创建类，类名我们还是习惯性定义首字母大写
2. super()

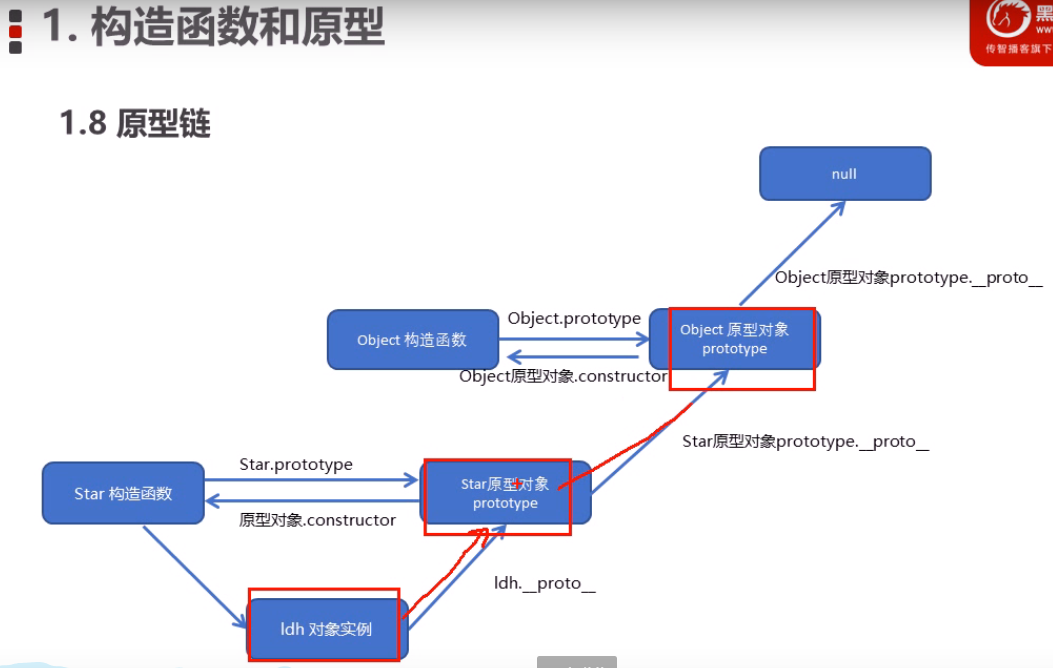
                // super 必须在子类 this 之前调用

                super(x, y);

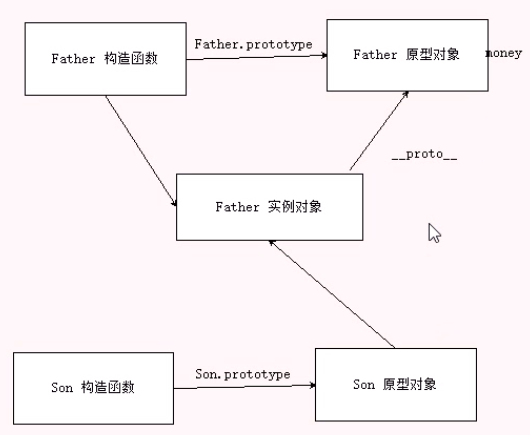
                this.x = x;

                this.y = y;

1. 使用类：
   1. 在 ES6 中类没有变量提升，所以必须先定义类，才能通过类实例化对象
   2. 类里面的共同的属性和方法一定要加this使用
2. 原型链



1. 利用原型对象实现继承



1. defineProperty

var obj = {

            id: 1,

            pname: '小米',

            price: 1999

        }

Object.defineProperty(obj, 'address', {

            value: '中国山东蓝翔技校xx单元',

            // 如果值为false 不允许修改这个属性值 默认值是true

            writable: true,

            // enumerable 如果值为false 则不允许遍历，新增的默认的值是 false

            enumerable: true,

            // configurable 如果值为false 则不允许删除这个属性，新增的默认的值是 false

            configurable: true

        })

1. 改变this指向的三种方法
   1. call()

        function Father(uname, age, sex) {

            this.uname = uname;

            this.age = age;

            this.sex = sex;

        }

        function Son(uname, age, sex) {

            Father.call(this, uname, age, sex)

        }

* 1. apply()

        // 1. 也是调用函数 第二个可以改变函数内部的this指向

        // 2. 但是他的参数必须是数组(伪数组)

        // 3. apply 的主要应用 比如说我们可以利用 apply 借助于数学内置对象求最大值

        var arr = [1, 66, 3, 99, 4];

        var max = Math.max.apply(Math, arr);

        var min = Math.min.apply(Math, arr);

* 1. bind()

        var o = {

            name: 'andy'

        }

        function fn(a, b) {

            console.log(this);

            console.log(a + b);

        }

        var f = fn.bind(o, 1, 2);

        f();

        // 1. 不会调用原来的函数 可以改变原来函数内部的this 指向

        // 2. 返回的是原函数改变this之后产生的新函数

        // 3. 如果有的函数我们不需要立即调用 但是又想改变这个函数内部的this指向此时用bind

        var btns = document.querySelectorAll('button');

        for (var i = 0; i < btns.length; i++) {

            btns[i].onclick = function () {

                this.disabled = true;

                setTimeout(function () {

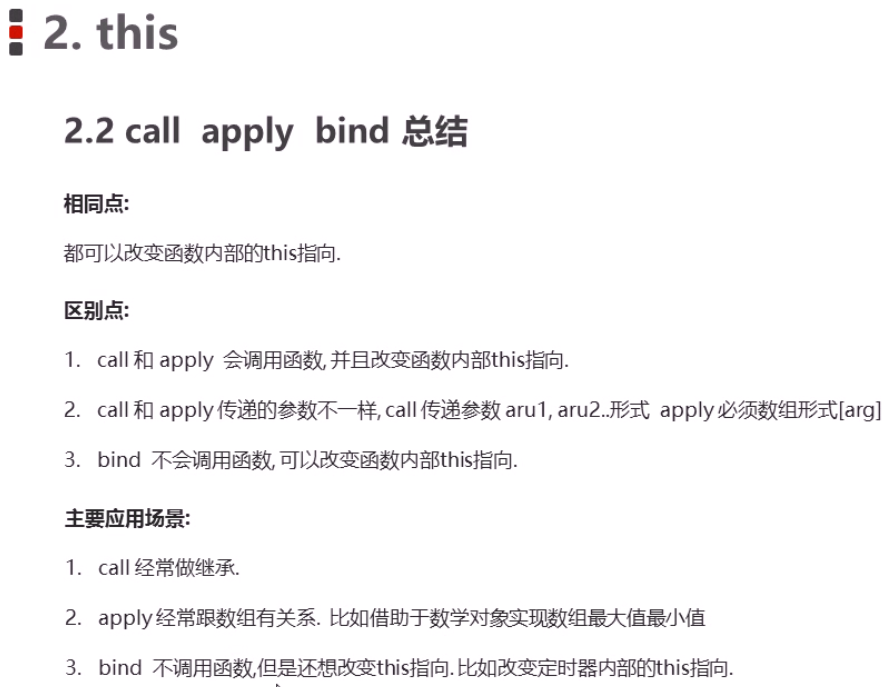
                    // btn[i].disabled = false;

                    this.disabled = false;

                }.bind(this), 2000)

            }

        }



1. 严格模式 'use strict';
2. 闭包

        // 利用闭包的方式得到当前小li的索引号

        for(var i = 0; i < lis.length; i++) {

            // 利用for循环创建了4个立即执行函数

            // 立即执行函数也称为小闭包，因为立即执行函数里面的任何一个函数都可以使用它的i这个变量

            (function(i) {

                // console.log(i);

                lis[i].onclick = function() {

                    console.log(i);

                }

            })(i);

        }

1. 深拷贝

function deepCopy(newobj, oldobj) {

            for(var k in oldobj) {

                var item = oldobj[k];

                if (item instanceof Array) {

                    newobj[k] = [];

                    deepCopy(newobj, item);

                } else if (item instanceof Object) {

                    newobj[k] = {};

                    deepCopy(newobj[k], item);

                } else {

                    newobj[k] = item;

                }

            }

        }