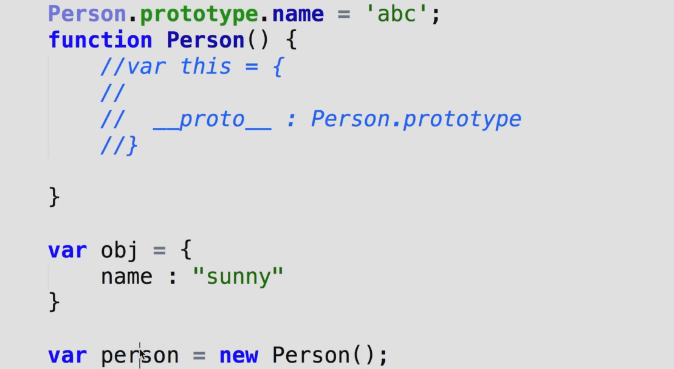
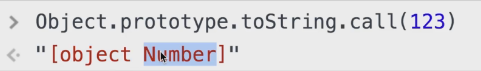
## Javascript小纪

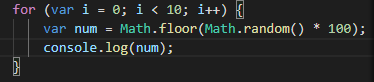
1. javascript通过原型链而不是类来支持面向对象编程。Javascript同样支持函数式编程，因为他们也是对象，函数也不可以保存在变量中，并且像其他对象一样被传递。
2. 原型：
   1. 定义：原型是function对象的一个属性，它定义了构造函数制造出对象的公有祖先。通过该构造函数产生的对象，可以继承该原型的属性和方法。原型也是对象。
   2. 利用原型特点和概念，可以提取公有属性；
   3. 原型属性的增删改查，可以删除自身的属性，但是不能删除原型的属性和方法，原型的属性和方法只能通过自己删除；增：person.name = ‘zhao’,,,,,,,删除:delete



* 1. Constructor（创造出对象的原型）: 可以手动更改
  2. \_\_proto\_\_:存的当前对象的原型（prototype）；当new Person的时候会有一个隐式的this，里面有个\_\_proto\_\_指向这个Person的原型。\_\_proto\_\_可以进行更改，\_\_proto\_\_是隐式的内部属性，但是不能自己增加
  3. Undefined和null不能调用toString方法，因为它们没有包装类；若自定义toString，就叫做方法的重写。Object.prototype.toString.call(123)：相当于让123去调用Object.prototype.toString上的方法。
  4. 调用document.write的时候一定会隐式的调用toString方法。

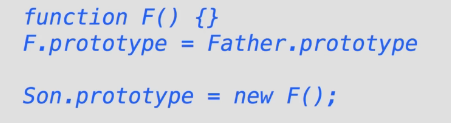


* 1. Js精度不准，可以做如下操作



* 1. Call/apply的区别（必须要先new出来一个构造函数）
     1. 共同点：改变this指向（借用别人的函数实现自己的功能）；不同点：传参列表不同。Call能传单个实参进去，apply只能接收一个数组（call需要把实参按照形参的个数传进去，apply需要传一个arguments）
     2. 函数不new的话person中的this默认指向window，person.call（obj），这个时候this指向obj，this.name变成obj.name

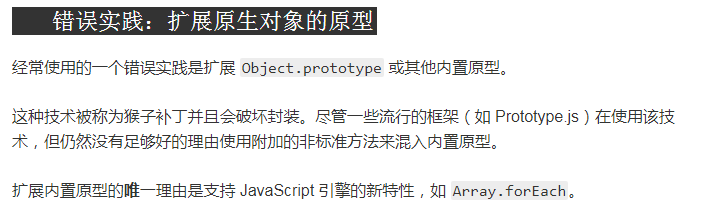
1. 原型链：
   1. 定义：当访问当前对象的属性并没有找到的时候，再去这个对象的prototype上去找，直到找到Object.prototype为止，这种查找方式呈链式连接，所以叫做原型链。
   2. 增删改查，只能通过本人修改，后代不能修改 ；
   3. 绝大多数对象的最终都会继承自Object.prototype；null和object.prototype除外。
   4. Object.create(原型)
2. 继承（继承的时候要先继承再用，不然就会报错或者打印undefined）
   1. 传统继承-------->原型链：过多的继承了没用的属性
   2. 借用构造函数（call || apply来实现）
      1. 不能继承借用构造函数的原型
      2. 每次构造函数都要多走一个函数
   3. 共享原型：不能随意改动自己的原型
   4. 圣杯模式



1. Js创建对象的方式
   1. 通过语法结构创建对象；var a = {a: 1}此处的o继承了Object.prototype，并继承object的一些方法
   2. 通过构造函数创建对象
   3. 使用object.create创建对象
   4. 使用class关键字创造对象

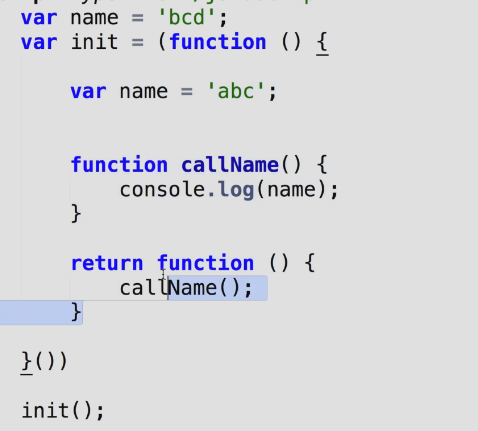
性能：原型链上查找属性比较耗时，比较损耗性能，当试图访问不存在的属性的时候会遍历整个原型链。**要检查对象是否具有自己定义的属性，而不是原型链上某个属性，必须使用object.prototype上的hasOwnProperty（）方法；它是js中唯一一个处理属性并且不会遍历原型链的方法；检查undefined属性的时候检查不出来，可能属性是存在的，但是刚好被设置成undefined。**

* 1. 错误实践

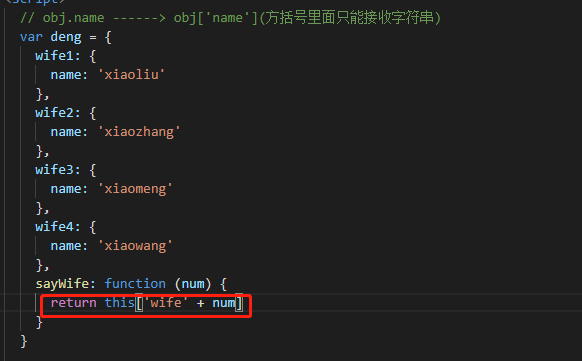


* 1. 拓展原型链的方法
     1. New-initialization
     2. Object.create
     3. Object.setPrototypeOf
     4. \_\_proto\_\_

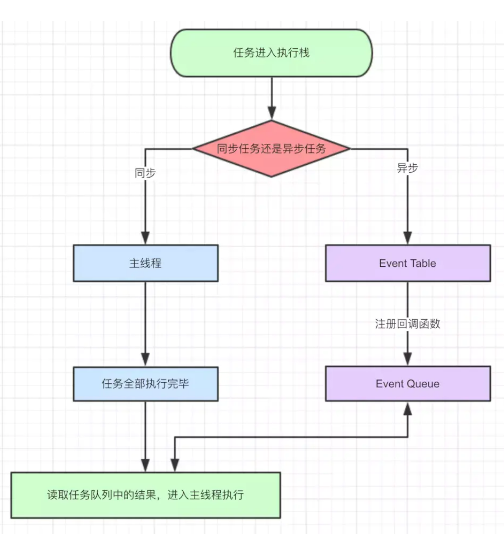
1. 数组的创建方式：（创建数组的一切方法来源于Array.prototype）
   1. 数组字面量： var arr=[10]，长度为1，是10
   2. 系统提供的构造方法 var arr= new Array（10），长度是10，每个都是empty，不能接受一个小数
2. 数组常用的方法（如果有负数-1，就是用-1+数组长度。就是代码数组的最后一位）：
   1. 改变原数组：
      1. Push：向数组最后一位增加新的值
      2. Pop： 会把数组的最后一位剪切出来，不能传参，只能剪切最后一位
      3. Shift：从数组最前面减去一个数值
      4. Unshift：向数组最前面增加一个新值（不只传一个值）
      5. Sort：给数组排序并且改变原数组
         1. 必须写两个形参
         2. 看返回值
            1. 当返回值为负数，那么前面的数放在前面
            2. 为正数，那么后面的数在前
            3. 为0，不动
      6. reverse ：逆转数组，返回原数组
      7. Splice（从第几位开始，截取多少长度，在切口处添加新的数据）
   2. 不改变原数组（直接变成新数组，不改变原来的数组）：
      1. Concat：连接两个数组，变成一个新数组
      2. Join： 括号里最好传一个字符串，用join里面传的东西连接数组，并返回一个字符串
      3. Split：跟join逆转，把字符串转化成数组拆分。Split里面传什么值就是把哪个删除
      4. toString：把数组变成字符串展示出来
      5. Slice（从该位开始截取，截取到该位）
3. Js（object based）是基于面向对象的语言
4. 命名空间：管理变量，防止污染全局，适用于模块化开发



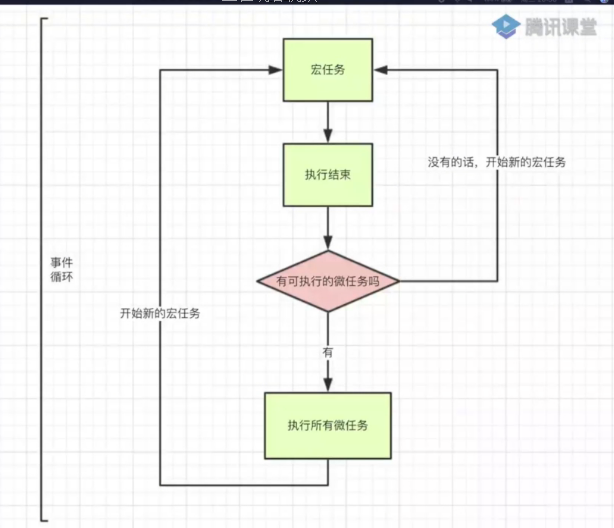
1. 实现方法的连续调用可以return this；obj.eat().smoke().drink()。。。。



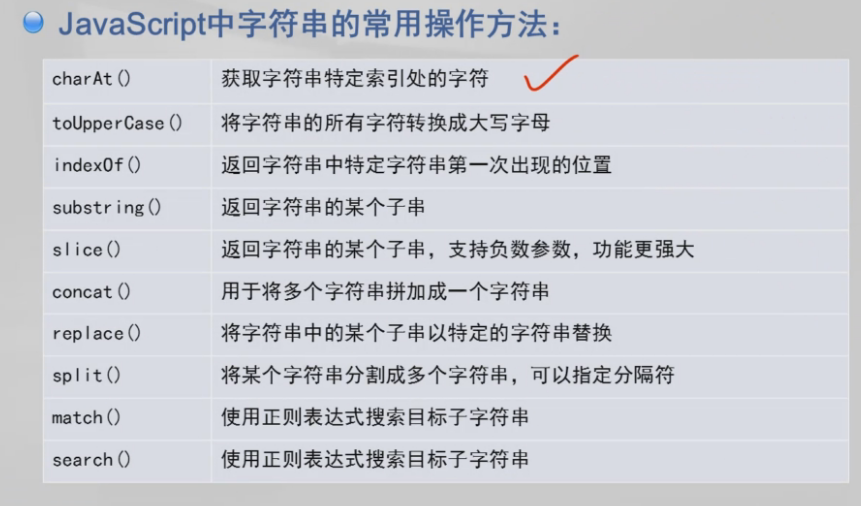
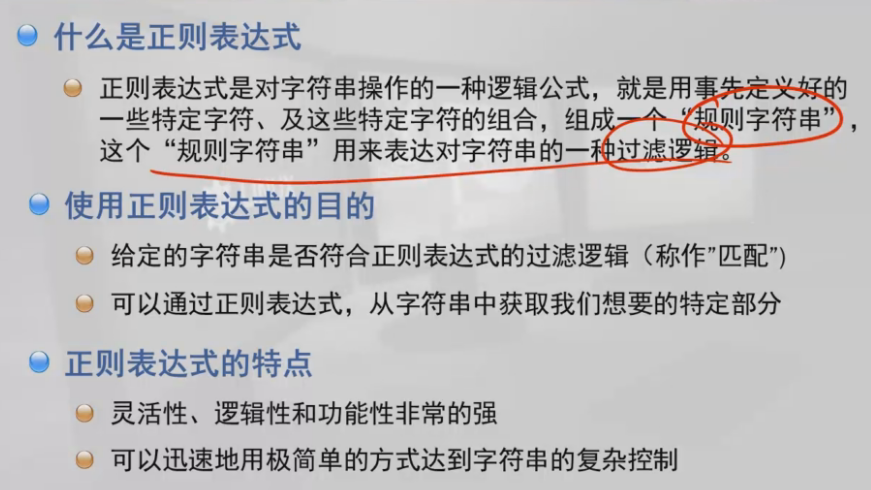
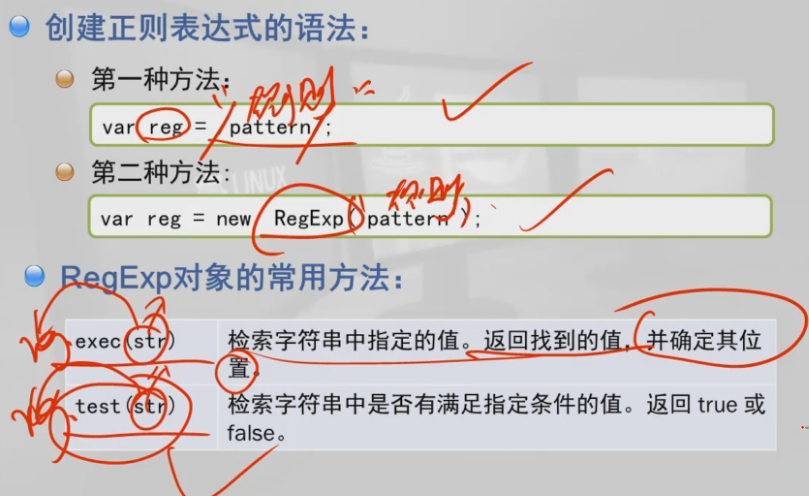
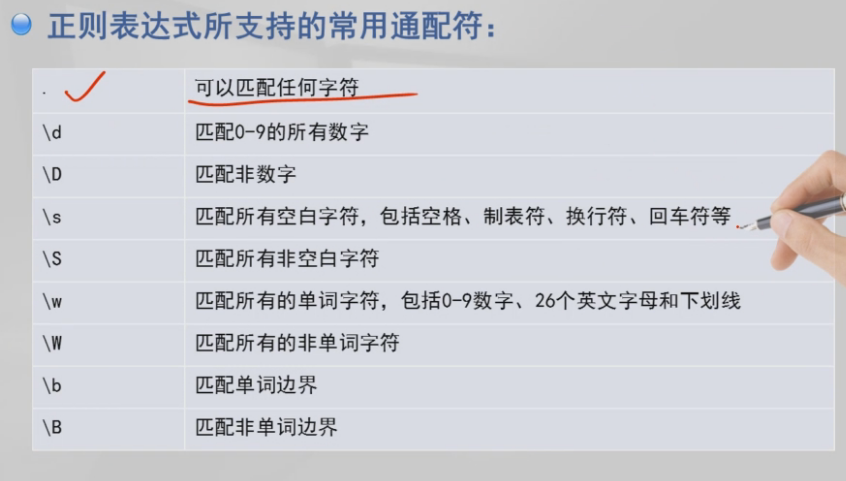
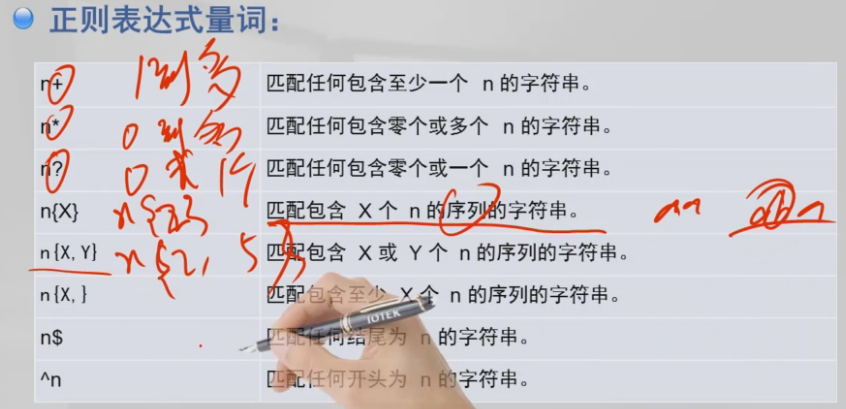
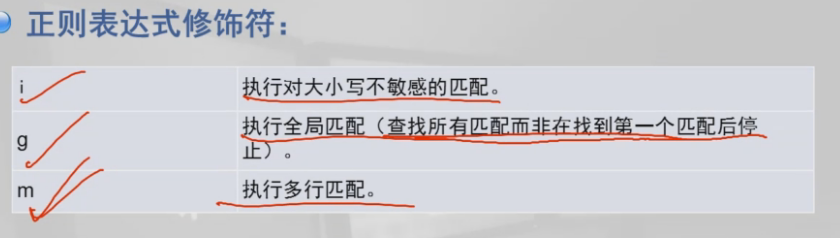
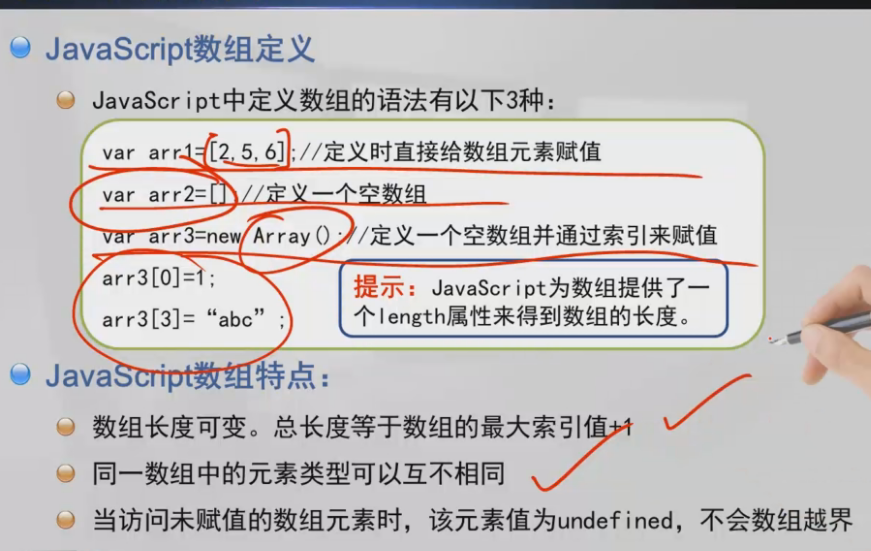
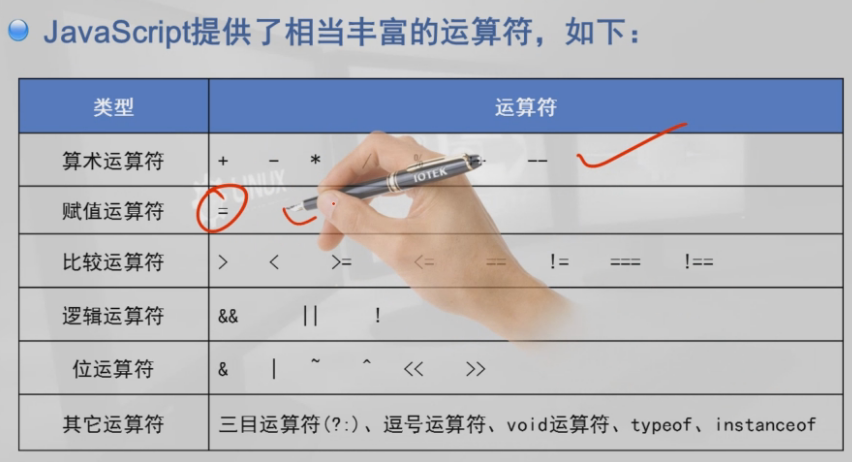
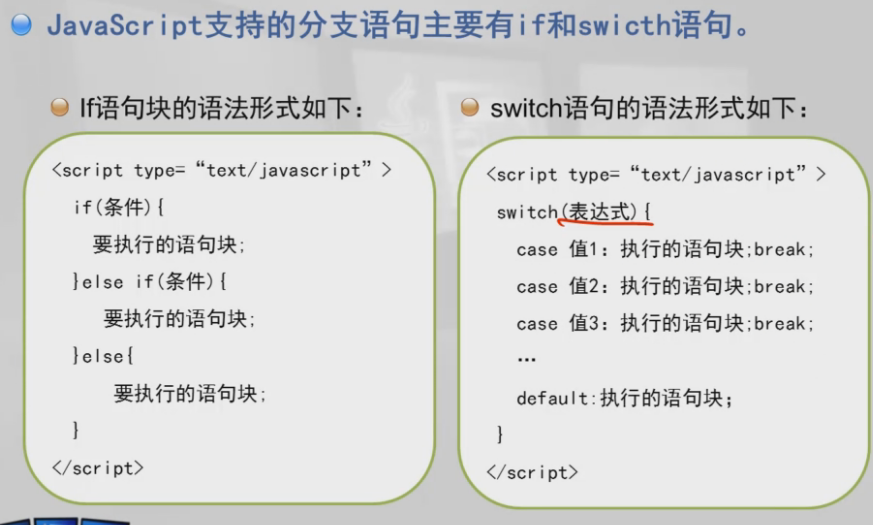
1. Javascript是单线程的解释性语言，解释一行执行一行。是弱类型的语言。
2. 关于运算符“，”，var num = (1, 2),这个时候打印num会返回2，返回后面的那个，前提是要加上“（）”
3. This
   1. 函数预编译过程this ---> window
   2. 全局作用域里this ----> window
   3. Call/apply可以改变函数运行时this指向
   4. Obj.func()；func（）里面的this指向obj--------------->谁调用this，this就指向谁
4. Arguments
   1. Arguments.callee 指向自身
   2. Func.caller 谁调用就指向谁
5. 枚举（遍历）for---in循环来循环对象，对象有几个属性就循环多少圈
   1. For in 遍历对象的时候for(var prop in obj)，调用对象得属性的时候obj.[obj.prop],原因是在调用对象的属性的时候，系统会隐式的把obj.prop转化成obj[‘prop’]，所以系统认为prop（字符串）是个属性。会打印出原型上的东西\_\_proto\_\_，但是不会打印出顶端的属性（\_\_proto\_\_.prototype）,系统自带的原型不能打印出来；要判断一层，if（origin.hasOwnProperty(prop)）{},判断是不是原型上的属性，不copy原型上的属性
   2. hasOwnProperty：判断是否是自己的属性，返回一个布尔值
   3. In： 只能判断对象上有没有那个属性
   4. 区别数组和对象的方法
      1. Instanceof：A instanceof B ： 看A的原型链上有没有B的原型 （aA对象是不是B构造函数构造出来的）
      2. constructor
      3. Object.prototype.toString.call()：系统识别call里面的来判断类型
6. 三目运算符： （条件判断）“？ （是） ：（否）”并且会返回值；条件判断语句，问号前面是条件判断，冒号前面执行是，冒号后面执行否
7. 事件循环机制（Event Loop）：为了提高整个js执行和页面渲染的效率，提高用户的体验



宏任务和微任务都是针对





1. 防抖和节流：作用是为了减少函数必要的触发次数。
   1. 防抖：只有在最后一次操作能被触发
   2. 节流：一定时间内只触发一次，间隔执行
2. Background：contain/cover cover是覆盖整个div，背景图片可能会溢出。所以可能会放大图片。Contain是完全被包含在div里面
3. 
4. 
5. 
6. 
7. 
8. 
9. 1.通过+或toString（）将数值转换成字符串
10. 2.通过parseInt（）将字符串转化成整型
11. 3.通过parseFloat（）将字符串转换成浮点型
12. 
13. 
14. 
15. 
16. 