第三阶段

day8.26

张东

微信: 18301092802

准备三样东西:

1. 犀牛书——JavaScript权威指南 第六版

2. 微信公众号——前端大全——总结鄙视题

3. 上届笔记和视频:

共享资料/视频:

http://code.tarena.com.cn

用户名: tarenacode

密码: code\_2015

提示: 视频较大，优先找项目经理要

规矩: 不要记笔记

第三阶段内容: 概述

JS高级: JS最核心的原理和编程

闭包， 面向对象， ES5， ES6， ES7

问题: 都不能操作网页

DOM: 制作网页的交互效果

问题: 太繁琐

JQUERY: 对DOM的每一步进行的终极简化

问题: 不是最简单，没有简化做事的步骤。

VUE: 彻底的简化了所有重复劳动——今后所有大项目的终极做法

正课:

## 正则表达式: ptt

我操，我草，我艹，我cao,我 草，卧 槽

([我卧]|w\s\*o)\s\*([艹草操槽]|c\s\*a\s\*o)

+86 18983848750 电话号码

0086 18983848750

((\+86|0086)\s+)?1[3-8]\d{9}

+86 李18983848750

((\+86|0086)\s+)?[\u4e00-\u9fa5]1\d{10}

京A·BC345 车牌号

[\u4e00-\u9fa5][A-Z]·[A-Z0-9]{5}

500228199409167639 身份证验证

11010219981226125x

1101021998122612

^\s\*\d{15}(\d\d[0-9x])?\s+

4556 123784 短信验证码

\d{4,6}

‘微信’敏感字符串

微信 微 信 wx w x 微 xin

(微|w|wei)\s\*(信|xin|x) 改进写法:(微|w(ei)?)\s\*(信|x(in)?) 问号在这里满足贪婪模式

匹配每个单词首字母

\b[a-zA-Z]

1815679735@qq.com

1815679735@qq

xiaohua@chinaren.com

匹配邮箱 ([a-z0-9]\*@[a-z0-9]{2,10}.com) 改进写法\w+@\w{2,10}\.[a-z]{2,3}(\.[a-z]{2})?

## 字符串中的正则函数:

### 查找敏感词: 4种:

#### indexOf (查找一个敏感词位置)

var i=str.indexOf("敏感词",fromi)

返回值:

如果找到，返回关键词第一个字在字符串中的下标位置

如果没找到，返回-1

问题: 不支持正则

#### search (模糊查找一个敏感词位置)

var i=str.search(/正则/i)

返回值: 同indexOf()的返回值

问题: 只能返回敏感词的位置，不能返回敏感词的内容

#### match (查找敏感词的内容)

var arr=str.match(/正则/i)

返回值:返回一个数组对象: arr[ "0":"敏感词内容", "index": 敏感词位置 ]

强调: 如果一个函数有可能返回null，那么都应该先验证不是null，才能使用！

问题: match默认只能找第一个敏感词。 解决: 在正则表达式第2个/后加后缀g global

但是: 一旦match加上g，返回值就变了！

如果找到，还是返回数组，但数组中仅包含多个敏感词内容的列表，不再包含位置

arr[ 敏感词1, 敏感词2, ... ]

共找到几个敏感词: arr.length

如果没找到，返回null

所以:

要获得本次找到的敏感词，可用: arr["0"]，简写为arr[0]，

要获得本次找到的敏感词的位置，可用: arr["index"]，可简写为arr.index

补: js最核心原理之一:

js中一切对象和数组底层本质上都是关联数组。无论是数组下标，和对象属性，都可用["下标或属性名"]方式访问。只不过，如果属性名是单词，不是数字时，可简写.属性名。如果下标是数字时，可省略""而已。

查找函数返回值的规律:

如果查找函数本来返回的是下标i，则如果找不到，返回-1，比如: indexOf()和search()

如果查找函数本来返回的是数组，则如果找不到返回的null，比如: match() 和 reg.exec()

#### reg 查找每个敏感词内容和敏感词下标位置:

RegExp对象:Regular Expression

什么是: 专门在程序中保存一条正则表达式，并提供用正则表达式执行操作的函数的对象

何时: 只要在js中用到正则表达式，第一步都是先创建正则表达式对象，然后再按需使用正则表达式对象的功能

1. 两种创建方式

1. 用//创建: var reg=/正则/ig

问题: 不支持动态生成正则表达式

因为: //中是正则表达式的地盘，不能写js

2. 用new创建:

var reg=new RegExp("正则表达式","ig")

优点: 可用js动态生成任意想要的字符串

强调: 如果用这种方式创建正则表达式对象，就不用//。

2. 函数：

1. 验证格式:

var bool=reg.test(str)

用正则表达式reg去验证str的格式是否符合要求！

返回值:

如果str中有格式符合reg的要求的字符串，则返回true，否则，返回false

坑: 默认，正则表达式只要在str中找到部分内容符合正则表达式的要求，就返回true，而不要求完整匹配！

解决: 今后，凡是验证，都要同时前加^，后加$，表示必须从头到尾完全匹配！

2. 查找函数:

既找每个敏感词的内容，又找每个敏感词的位置:

var arr=reg.exec(str)

去str中找下一个符合reg正则表达式要求的敏感词。

返回值: 和match不加g时，返回值完全一样

arr:[ "0":"敏感词内容", "index", 位置 ]

与match()的异同：

match()即使反复调用，也只能找第一个

reg.exec()如果反复调用，会自动找下一个。所以，reg.exec()和循环do-while搭配使用，可找到所有敏感词的内容和位置。

arr=regexec(str) 判断条件是：arr=!null(null没有index属性，所以循环内要判断是否为null)

作业:

1. 匹配“微信”，“weixin”，“w x”等情况，并防止中间加空格

2. 使用正则匹配电子邮件: ——自己百度一个记住。

day8.27

规矩:

每个完整功能的案例必须写三遍:

1. 注释+代码抄一遍

2. 删掉代码，保留注释，自己把代码填回来

3. 全部删掉，自己先写注释，再把代码填回来！（时间不够，只要写出注释就算成功！）

1. 正课:

1. 字符串中的正则函数

2. RegExp类型:

3. Function 类型:

### 替换敏感词: 2种:

#### 1）简单替换

1. 简单替换: 将所有敏感词都替换为同一个新值

str=str.replace(/正则/ig, "新值")

在str中查找所有符合正则要求的敏感词并替换为同一的新值。

补: 字符串在js中是不可变类型: 一旦定义，值不可改变！所有对于字符串的修改，都无权修改原字符串，还能返回新字符串。如果希望获得修改后的新字符串，只能自己用变量保存起来。如果不保存，新值就扔了。

总结: 今后所有字符串操作的函数，要想获得处理结果，只能自己用变量接住修改后的新字符串

#### 2）高级替换

2. 高级替换: 根据不同敏感词动态选择不同的新值替换

str=str.replace(/正则/ig, function(keyword){

return 根据不同keyword，加工后的新值

})

其中:

//第二个参数是一个回调函数

//这个回调函数有两个要求:

//1. 必须有一个形参，比如: keyword

// 在执行时，会自动接收到本次找到的一个敏感词

//2. 回调函数内，应该根据keyword变量接住的不同敏感词，加工后生成不同的新值，返回到外部

//结果: replace()，会用正则表达式去字符串中找每个符合条件的敏感词，每找到一个，就自动调用一次回调函数，每次自动调用回调函数时，都会自动传入本次找到的一个敏感词给回调函数的形参，然后回调函数内部根据不同敏感词加工生成不同的新值。最后replace会收到回调函数返回的新值，并替换到本次找到的敏感词位置。

补: 回调(callback)函数:

什么是: 自己定义的函数，自己不调用，而是交给别人去自动调用。

何时: 当一段逻辑中，缺少一段代码片段时，就需要用回调函数方式补充！

比如:

ajax: xhr.send()

问题: 收到服务端响应后干啥？缺收到数据后如何处理，所以需要用xhr.onreadystatechange=function(){}提前规定好一段代码，等到响应回来后，自动执行（回调函数调用机制）——信任

str.replace(/正则/g)可以找到敏感词，但是，找到敏感词后，干什么？！也需要一个函数来补充缺少的那一部分逻辑。

总结: 回调函数的学习方法是:

在回到函数内输出:"被调用了一次"

如果回调函数有参数，同时输出本次接收的参数值。

结果: 随着回调函数被自动反复执行，就会反复输出回调函数的执行次数和参数值。

#### 替换操作的衍生操作: 删除敏感词

其实就是将敏感词替换为空字符串

笔试: 去掉一个字符串开头或结尾的空字符:

### 切割

什么是: 将一个字符串，按某个切割符，切割为多段子字符串

如何:

1. 简单切割: 切割符是固定不变的

var arr=str.split("切割符")

将字符串str按固定的切割符切割为多段子字符串，保存在一个数组中返回。

强调: 切割后，数组中不包含切割符

2. 复杂切割: 切割符是不固定的

var arr=str.split(/正则/)

将字符串str按所有符合正则要求的字符进行切割，变成多段子字符串。

3. 衍生操作: 打散字符串为字符数组:

为什么: 字符串虽然和数组长得像，但是却不是数组家孩子。

字符串 vs 数组:

相同: 1. 下标, 2..length 3. 可用for遍历

不同: 不是同一类型，函数不通用！

何时: 如果希望字符串也能调用数组家函数时，都要先将字符串切割为字符数组，然后再调用数组家函数，最后再拼接回字符串

如何: 按空字符串切割""

var chars=str.split("")

复习: 字符串:

什么是: 一串字符组成的只读字符数组！

vs 数组:

相同:

1. 下标: 都可以用[i]访问每个每个字符/元素

2. .length: 字符/元素的个数

3. 遍历

不同: 类型不同，导致函数不通用

特例: .slice() 两家人都有

比如:

var pid="110102199812262151"

//012345678901234567

// ↑ ↑

var birth=//pid.slice(6,14) 切割内容不包括索引为14的位置

//pid.slice(6,-4)

pid.slice(6,pid.length-4)

var gender=pid[pid.length-2]%2==1?"男":"女"; //倒数第2位是奇数，是男，偶数是女

作业:

1. 实现去掉开头或结尾的空字符的三个函数: ltrim() rtrim() trim()

单词列表:

1. slice 截取

2. splice 修改: 删除 插入 替换

3. split 切割

day8.28

## function

### 1.定义

在程序中，有一段代码需要反复使用，将其封装成一段可重用的代码段的程序结构叫做函数。防止冗余重用代码

### 2.创建方式

①声明方式

function 函数名（形参列表）{

函数体；

return 返回值；

}

1）形参是局部变量

2）返回值根据用户需求来决定是否return返回值

3）声明函数，不调用，则不执行。

（1）问题：声明提前

声明提前：在程序开始执行前，js引擎先将var声明的变量和function声明的函数提前到当前作用域的顶部，集中创建。赋值留在原地。

②赋值创建

var 函数名=function（形参列表）{

函数体

}

优点：不会被声明提前！

揭示：函数其实是一个对象，函数名只是一个变量，函数名通过函数对象的唯一地址引用这函数对象

自我总结：

1.变量 var a=100;

变量声明赋值的过程：（1）声明：var a;（2）赋值： a=100;

声明会提前到当前作用域顶部，赋值在原位置执行

2. 函数

（1）function声明创建

存在声明提前

function 函数名(){函数体} 全部提前

（2）function赋值创建（相比声明创建，赋值比较常用）

不存在声明提前

var a=function(){} 同变量，(1)var a;提前 (2)a= function(){};不提前（理解记忆）

另外，这里的a这个变量赋值的是函数所指向的地址

### 3.调用函数

（1）重载

概念：相同函数名，不同形参列表的多个函数，在调用时，可自动根据传入实参列表的不同，自动匹配对应的函数

为什么使用重载：减少函数名个数，减轻调用的负担。

注意：js语法默认不支持重载语法！会覆盖

解决：借助一个特殊的对象：argument（参数） paramter(参数)

argument：每个函数都有的参数argument，就是接收的实参列表，就是一个类数组对象。

可用属性:.length,[index],for遍历。

用argument.length在函数中用if判断传入实参列表的不同进行不同代码执行，实现重载效果。

### 4.匿名函数

1.概念：函数创建后，不被任何变量使用

2.功能：（1）节约内存，匿名函数没有变量引用，使用后内存自动释放！

（2）划分临时区域，避免使用全局变量。污染全局。

3.使用情况：

（1）如果函数只使用一次，就不要起名。

（2）今后，所有的js代码都应该包含在一个匿名函数内。

4.使用方法：

（1）绝大多数回调函数，都是匿名函数

function(…){…}

（2）匿名函数自调用。创建函数后，立即调用函数。（js代码用匿名函数包裹：立即执行，立即释放）

(function(…){…})();

## 作用域和作用域链

作用域（scope）概念：

1.用途：规定一个变量的可用范围

2.本质：就是内存中保存变量的对象

作用域防止了不同范围的变量间相互污染

作用域（2级）：

1.全局作用域：window对象

（1）特点：在任何位置都可以访问到。（2）包含：全局变量。 （3）优缺点：可反复使用，但极易污染

另外，特殊情况：如果在哪里都找不到这个变量，又强行赋值，则自动在全局创建该变量

2.函数作用域

（1）特点：仅函数内可以使用。

（2）包含：局部变量，在函数内var出的变量和形参变量。

（3）优缺点：不可重用，只能在函数内使用，不污染全局

|  |
| --- |
| 原理：定义函数时，起始每一个函数都有自己的“好友列表”- - -作用域链  作用域链中先有一个指向window的引用  当调用函数时，才临时创建函数作用域，并将函数作用域的引用插入“好友列表”中window之前  执行函数时，按照“好友列表”中离自己远近的顺序，先尝试从函数作用域中找变量（局部变量）使用，如果函数作用域中没有，才去全局找  当函数调用完成后，本次调用函数会创建临时函数作用域会被释放！导致所有局部变量异同被释放。  理解：“好友列表”就是作用域链，“铁哥们儿”和“备胎”就是作用域。见下图 |

作用域链：

概念：有多级作用域组成的列表，作用域链保存在一个函数内。

作用域链中包含：一个函数可用的所有作用域对象的引用，也就是包含着这个函数可用的所有变量

作用域链控制着：变量的使用顺序（先局部再全局）

for循环不会形成一个封闭的作用域。

## 闭包（closure）

closure 封闭，闭合

（1）闭包：

1.用途：既重用变量，又保护变量不被污染的一种编程方法

2.本质：外层函数的作用域对象，被内层函数引用着。外层函数的作用域对象就成为闭包

为什么要使用闭包：全局变量和局部变量有不可兼得的优缺点。

使用情形：既要重用一个变量，又要保护变量不被污染时

（2）闭包形成步骤：

（1）定义外层函数，包裹要保护的变量和内层函数

（2）外层函数返回内层函数到外部

（3）调用外层函数，获得返回的内层函数保存在变量中

总结：外层函数调用后，外层函数的作用域对象被内层函数引用着，无法释放

（3）闭包的缺点: 比普通的函数占用一块更多的内存——外层函数的作用域对象

笔试时：画简图：找两样东西

1.外层函数的局部变量

2.外层函数返回的内层函数是哪个

单词：

1. hoist 吊车 声明提前

2. arguments 参数

parameter 参数

3. scope 范围

4. closure 封闭 闭合

笔试题：

1.

function fun(o){

o.name="西西";

o={};

o.name="小丽"

}

var obj={name: "小红", age:11};

fun(obj);

console.log(obj); //? 结果为Object:{age: 11,name: "西西"}

2.

function fun(){

for(var i=0,arr=[];i<3;i++){

arr[i]=function(){ console.log(i) }

}

return arr

}

var funs=fun();

funs[0]();//? 3

funs[1]();//? 3

funs[2]();//? 3

3.

var a=10;

function fun(a){//局部变量a拿到的只是全局变量a的副本，修改局部变量a的值，和全局变量a无关！

//形参变量，也是局部变量

//如果有局部变量a，则以下a都和全局无关！

console.log(a); //10

a=100;

console.log(a);

}

fun(a); //10 100

console.log(a);//10