Rotes

# Yingyu

That will be really helpful if you can.

Parse 解析

Callback 回调函数

Filed 字段

# Question

jQuery documentation

1111：离圣诞还有43天

CSS Datatables

Ajax

把英语提高，每天来公司

超市：

早餐，带公司的小零食

上班穿的衣服，小西服外套，白衬衫

# Pandas

Tings should know:

Homebrew

Problem fix : download homebrew -> install sql -> install

1028 HTML, JavaScript, go through documentation

CSS

A list of job hold and other: see issue #44

# HTML

No data being entered by the user, use <a>

<a class="btn btn-primary btn-lg" href="sample\_log/" target="\_blank">Sample Log Report</a>

+ <a class="btn btn-primary btn-lg" href="call\_log/" target="\_blank">Call Log Report</a>

Have data being entered by the user, use button submit.

<button type="submit" formaction="call\_log/" class="btn btn-primary">

- Call Log Report

- </button>

# Django

Unittest. : python standard library

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# Javascript

## 菜鸟--- 用法

document.getElementById(“id\_name”)

document.getElementById().innerHTML= “Hello”

<script></script>

Alert

Document.write(“<p>Hello</p>”)

Onclick=”myFunction()”

外部的JavaScript：保存在外部文件。<script src="myScript.js"></script>

## 菜鸟---输出

window.alert() 弹出警告框

document.write() 将内容写道HTML文档中

innerHTML 写入到HTML元素

console.log() 写入到浏览器的控制台

## 菜鸟---语法

Literal 字面量

Scripting language 脚本语言

Var

Keyword 关键字

Identifier 标识符

注释 //

大小写敏感

使用Unicode字符集

## 菜鸟---语句statement

语句后添加；

空格：JS会忽略多余空格

字符串中可以使用 \ 进行换行

document.write("你好 \

世界!");

## 菜鸟---注释comment

多行注释 /\* \*/

## 菜鸟---变量variable

声明变量

声明可以横跨多行：

Var lastname="Doe",

age=30,  
job="carpenter";

声明的多个变量不可以同时赋同一个值：

## 菜鸟---数据类型type of data

值类型：string, umber, Boolean, Null, Undefined, Symbol

引用数据类型：Object, Array, Function

## 菜鸟---对象object

对象的属性和寻址：使用 { } var car = {type:"Fiat", model:500, color:"white"};

Addressing 寻址

New 用来声明变量类型：var x=      new Number;

对象中的方法可以是函数：

var person = {

firstName: "John",

lastName : "Doe",

id : 5566,

fullName : function()

{

return this.firstName + " " + this.lastName;

}

};

## 菜鸟---函数

函数是包裹在 { } 中的代码块， 前面使用到了关键字function

大小写敏感

函数变量和传参必须以相同顺序出现

Local variable 局部变量

Global variable 全局变量

变量的生存期

为声明变量可删除： delete var1；

## 菜鸟---作用域scope

全局变量有俩种：

在函数外定义

在函数内不使用var 关键字声明

## 菜鸟---事件event

HTML事件：html页面完成加载 / input 字段改变 / 按钮被点击

可以在html中使用single quote 或者double quote 来添加 JS代码：

<*some-HTML-element* *some-event*=**"*JavaScript 代码*"**>

Quote里面也可以是JS函数：

<button onclick="displayDate()">点这里</button>

<script>

function displayDate(){

document.getElementById("demo").innerHTML=Date();

}

</script>

<p id="demo"></p>

This.innerHTML= 改变自身元素的内容

常见HTML event ：

onchange 元素改变

onclick 点击元素

onmouseover 在元素上移动鼠标

onmouseout 在元素上移开鼠标

onkeydown 按下键盘按键

onload 浏览器已完成页面的加载

## 菜鸟---字符串string

引号怎么使用

字符串长度： str.length

特殊字符： \’ \” \\ \n \r \t \b \f

New 可以使用new 关键字将string定义为objact

=== ：绝对等号

字符串属性： constructor, length, prototype

字符串方法：charAt(), charCodeAt(), Concat(), fromCharCode(), indexOf(), lastIndexOf(), localeCompare(), match(),

replace(),search(), slice(), split(), substr(), substring(), toLocaleUpperCase(), toLocaleLowerCase(),

toLowerCase(), toString(0, toUpperCase(), trim(), valueOf()

## 菜鸟---运算符operator

算数运算符： + - \* / % ++ ---

赋值运算符： = += -== \*= /= %=

字符串相加

## 菜鸟---比较

比较运算符： == ==== != !==不觉对等于

逻辑运算符： && : and ||: or ！: not

## 菜鸟---条件语句if

if (time<20)

{

x="Good day";

}

## 菜鸟--- Switch语句

用于基于不同条件来执行不同的动作

Default 关键字来规定匹配不存在时

## 菜鸟--- for循环

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

{

document.write(cars[i] + "<br>");

}

【不同类型的循环】

For/in :

var person={fname:"Bill",lname:"Gates",age:56};

for (x in person) // x 为属性名

{

txt=txt + person[x];

}

## 菜鸟--- While循环

Do/while 循环 ： 是while的变体，循环会至少执行一次

## 菜鸟--- Break 和continue语句

Break ： 用于跳出循环，之后会继续执行循环后的代码

Continue： 用于跳过循环中的一个迭代。中断循环中的迭代

JS标签：

label:

statements

语法：

break labelname;

continue labelname;

## 菜鸟--- Typeof

检查数据类型

typeof "John”;

## 菜鸟---类型转换

JS有6种不同的数据类型： string number boolean object function symbol

3中对象类型： Object Date Array

2个不含任何值的数据类型： null undefined

constructor 属性：返回所有JS变量的构造函数

用来判断是否为日期，是否为string ：myDate.constructor.toString().indexOf("Date")

String() : 可用于任何类型的数字，字母，变量，表达式 转换成string

== .toString()

Number 方法：

toFixed() : 把数字转换为字符串，结果的小数点后有指定位数的数字

toPrecision(): 把数字格式化为指定的长度

【将布尔值转化为字符串】

【将日期转化为字符串】

【将字符串转换为数字】

Operator + 可用于将变量转换为数字

var y = "5";      // y 是一个字符串

var x = + y;      // x 是一个数字

【将日期转化为数字】 Number()

【自动转换为字符串】

【不同数值转换为数字，字符串，布尔值的结果】

## 菜鸟---正则表达式regular expression

语法：

/正则表达式主体/修饰符(可选)

【使用字符串方法】

Search() : 检索字符串中指定的字符串，或检索与正则表达式相匹配的子字符串，并返回子串的起始位置

Replace() ： 在字符串中用一些字符替换另一些字符，或替换一个与正则表达式匹配的子串

【正则表达式修饰符】 regex modifier

Modifier 修饰符

正则表达式方法：test() exec()

/e/.test("The best things in life are free!") // 结果true

【判断输入字符串是否为数字、字母、下划线组成】

【判断输入字符是否全部为字母】

【判断输入字符串是否全部为数字】

## 菜鸟---错误

throw 抛出

throw exception 抛出异常

Try 语句测试代码块的错误

catch 语句处理错误

throw 语句创建自定义错误

finally 语句在try 和 catch 语句之后，无论是否有触发异常，该语句都会执行

语法grammar：

try {

... //异常的抛出

} catch(e) {

... //异常的捕获与处理

} finally {

... //结束处理

}

【throw】

## 菜鸟---调试debugging

Built-in 内置

浏览器启用调试工具一般是按下 F12 键，并在调试菜单中选择 "Console" 。

【个浏览器debugging的方法】

## 菜鸟---变量提升hoisting

JS中函数和变量的声明都将被提升到函数的最顶部，也就是说可以先使用再声明。

初始化不会提升。

可以

x = 5；

var x; // 声明 x

不可以，会报错：

var x = 5; // 初始化 x

elem = document.getElementById("demo"); // 查找元素

elem.innerHTML = x + " " + y;           // 显示 x 和 y

var y = 7; // 初始化 y

## 菜鸟---严格模式strict mode

【use srict】

<script>

"use strict";

myFunction();

function myFunction() {

y = 3.14; // 报错 (y 未定义)

}

</script>

## 菜鸟---使用误区

浮点数： JS 中所有数据都是以64位浮点数来存储 // 可以使用整数乘除法来解决

【return使用注意事项】

数组中不能使用名字来索引，使用名字会被重新定义为标准对象

【Null 和 Undefined区别】

## 菜鸟---表单form

fname 表单字段

Constraint validation约束验证

document.forms[“formname”][“fname”].value

【约束验证】 表单被提交时浏览器用来实现验证的一种算法

HTML约束验证基于：

【HTML输入属性】 disabled max min pattern required type

【CSS伪类选择器 : disabled :invalid :optional :required :valid

【DOM属性和方法】

## 菜鸟---表单验证validation

【email 验证】

## 菜鸟---验证API

checkValidity()如果input元素中的数据是合法的返回true，否则返回false

setCustomValidity()用于自定义错误提示信息的方法

object.validationMessage

【约束验证DOM属性】 ： validity validationMessage willValidate

【Validity属性】 有很多，看菜鸟

## 菜鸟---保留关键字reversed keywords

避免使用关键字作为名字

【window关键字】

【HTML事件句柄】

## 菜鸟--- This

表示对当前对象的一个引用。但是会随着执行环境的改变而改变：

在方法中，this便是该方法所属的对象。

单独使用时，this表示全局对象。

在函数中，this表示全局对象。

在函数中，严格模式下，this是未定义的。

在事件中，this表示接受事件的元素。

类似call() 和apply()方法允许切换函数执行的上下文环境，即this绑定的对象，可以将this引用到任何对象。

This.stayle.display=’none’ // 消失

【显示函数绑定】

Call() apply() --函数对象的方法

var person1 = {

fullName: function() {

return this.firstName + " " + this.lastName;

}

}

var person2 = {

firstName:"John",

lastName: "Doe",

}

person1.fullName.call(person2); // 返回 "John Doe"

## 菜鸟--- Let 和 const

Let声明的变量只在let命令所在的代码块内有效 // 用来保证局部变量和全局变量不冲突

Const声明一个只读的常量，一旦声明，常量的值就不能改变。

JS只有俩种作用域：全局变量 与 函数内的局部变量。

在JS中，全局作用域是针对JavaScript环境。

在HTML中，全局作用域是针对window对象。

【var和let区别】：

使用var关键字声明的变量在任何地方都可以修改

在相同的作用域或块级作用域中，不能使用let关键字来重置var/let关键字声明的变量

在不同的作用域或块级作用域中，可以使用let关键字来重置let关键字声明的变量

Let/ const不能提升变量

const用于声明一个或多个常量，声明时必须进行初始化，且之后不可修改。

【const和let相同点】：

都是块级作用域

都不能和所在作用域内的其他变量或函数拥有相同的名称

Const本质：定义的对象或者数组，是可变的。但不能对常量数组重新赋值

Const在不同作用域可以重新声明赋值

## 菜鸟--- Json ： JavaScript Object Notation

语法：

数据为键/值 对

数据由逗号分隔

大括号保存对象

方括号保存数组

JSON.parse() ：  JavaScript 内置函数 JSON.parse() 可以将字符串转换为 JavaScript 对象

JSON.stringify(): 用于将 JavaScript 值转换为 JSON 字符串。

## 菜鸟--- Void

Void是JS中非常重要的关键字，该操作符operator指定要计算一个表示式但是不返回值。

Grammar:

void func()

javascript:void func()

or

void(func())

javascript:void(func())

【href="#"与href="javascript:void(0)"的区别】：

#包含了一个位置信息

javascript:void(0)仅仅表示一个死链接

## 菜鸟---异步编程Asynchronous

同步：按照代码顺序执行

异步：不按照代码顺序执行。效率更高

回调函数callback

setTimeout(arg1,arg2) : arg1是函数，arg2是时间

XMLHttpRequest() 属性：onload onerror open() send()

## 菜鸟--- Promise

是一个ECMAScript 6 提供的类，一些旧的浏览器并不支持.

Promise的构造函数（constructor）只有一个参数，是一个函数，这个函数在构造之后会直接被异步运行，所以我们称之为起始函数。起始函数包含2个参数resolve和reject。

新建：

new Promise(function (resolve, reject) {

// 要做的事情...

});

new Promise(function (resolve, reject) {

console.log(1111);

resolve(2222);

}).then(function (value) {

console.log(value);

return 3333;

}).then(function (value) {

console.log(value);

throw "An error";

}).catch(function (err) {

console.log(err);

});

方法：

.then() 将参数中的函数添加到当前的正常执行序列

.catch() 设定promise的异常处理序列

.finally() 最后一定会执行的序列

resolve() 中可以放置一个参数用于向下一个 then 传递一个值，then 中的函数也可以返回一个值传递给 then。但是，如果 then 中返回的是一个 Promise 对象，那么下一个 then 将相当于对这个返回的 Promise 进行操作。

* resolve 和 reject 的作用域只有起始函数，不包括 then 以及其他序列；
* resolve 和 reject 并不能够使起始函数停止运行，别忘了 return

【一些then catch finally的顺序及使用的问题】

## 菜鸟---代码规范

命名规则CamelCase

每行代码字符小于80个

HTML CSS的横杠（-）字符

## 菜鸟--- JS 函数

### 函数定义

【自调用函数】 ：表达式后面紧跟 ()，会自动调用

【JS函数可以作为表达式使用】

var x = myFunction(4, 3) \* 2;

【函数是对象，有属性和方法】 还有call() apply()

arguments.length : 这个属性返回函数调用过程收到的参数个数

toString() ：这个方法将函数作为一个字符串返回。

function myFunction(a, b) {  
    return a \* b;  
}  
  
var txt = myFunction.toString();

【箭头函数】

(参数1, 参数2, …, 参数N) => { 函数声明 }

(参数1, 参数2, …, 参数N) => 表达式(单一)

// 相当于：(参数1, 参数2, …, 参数N) =>{ return 表达式; }

没有参数时：

() => {函数声明}

// ES5

var x = function(x, y) {

return x \* y;

}

// ES6

const x = (x, y) => x \* y;

箭头函数的this的值会默认绑定外层this的值。

箭头函数不能提升，使用前需要定义

### 函数参数

【显性参数】

functionName(parameter1, parameter2, parameter3) {

// 要执行的代码……

}

【隐形参数】：函数调用时传递给函数的真正的值

【为参数设置一个默认值】 y = y ||0；

ES6 支持函数带有默认参数

【argument对象】 JS函数的内置对象

### 函数调用

JS函数有4种调用方式，他们的不同在于this 的初始化。

【this指向】 ：一般指向函数执行时的当前对象。

【构造函数】： 函数调用前使用 new 关键字。

【函数方法调用函数】；

call() apply() 他们俩个方法的第一个参数必须是对象本身。

他俩区别是传参方式，apply传入一个参数数组

### 闭包

## HTML DOM

### DOM 简介

### DOM HTML

### DOM CSS

### DOM 事件

### DOM EventListener

Event monitoring 事件监听

Bubble 冒泡

Capture 捕获

addEventListener()

removeEventListener()

语法：

*element*.addEventListener(*event, function, useCapture*);

第一个参数是事件的类型 (如 "click" 或 "mousedown").

第二个参数是事件触发后调用的函数。

第三个参数是个布尔值用于描述事件是冒泡还是捕获。该参数是可选的。

可以向同一个元素中添加多个事件句柄

### DOM 元素

### HTMLCollection对象

### NodeList对象

## JavaScript高级教程

### JavaScript 对象

### JavaScript prototype

### JavaScript Number 对象

### JavaScript String

### JavaScript Date（日期）

### JavaScript Array（数组）

### JavaScript Boolean（布尔）

### JavaScript Math（算数）

### JavaScript RegExp（对象）

## JS浏览器BOM