1. **js基础**
2. **深入js**
3. 函数返回

可以没有return，一个函数应该只返回一种类型

1. 函数传参

arguments 不定参（可变参）

---------------------------------------------------------------------------------------------------------------------------------

Jquery中

css(oDiv, ‘width’) //获取width

css(oDiv, ‘width’, ‘200px’) //设置width

根据arguments.length来实现

---------------------------------------------------------------------------------------------------------------------------------

获取非行间样式（\*应用单一属性）

oDiv.style.color //行间样式

oDiv.currentStyle.color //非行间样式 IE

getComputedStyle(oDiv, false).color //非行间样式 chrome,firefox

单一css属性 color

复合css属性 background(backgroundColor, backgroundImage...单一属性组成的)

\*99%的兼容性问题都是if来解决的

C、数组基础

i、[]与new Array() 建议用[]性能略高简单

ii、length 既可以获取也可以设置 例如：快速清空数组==length=0

iii、添加删除替换

添加 删除 替换

头部 Push()尾部添加 pop() 尾部删除

尾部 unshift()头部添加 shift()头部删除

任意 splice(起点，长度0，元素) splice(起点，长度) splice(起点，长度，元素)

iiii、数组的连接

[].concat([],[],[]) //连接成数组,结果为书写的顺序 --- 不改变原来

[].join(分隔符) //连接成字符串

iiiii、数组的排序

[].sort() //直接数字排序不对，数字当做字符串处理

[].sort(function(next, pre){

If(next < pre){

Return -1

}else if(next > pre){ == [].sort(function (next, pre) { return next-pre })

Return 1

}else {

Return 0

}

})

iiiiii、arguments和通过document的方法获取的NodeList都是伪类，没有数组的方法。

1. **DOM操作基础应用**

**JS**  **jquery**

innerHTML/value text()

父级.appendChild(node) 父级.append(strELE,无限量的新元素) 元素内部嵌入

父级.insertBefore(node, startNode) 父级.prepend(strELE,无限量的新元素) 元素内部嵌入

$(ele).after() 元素外部追加

$(ele).before() 元素外部追加

父级.removeChild(node) $(ele).remove() 移除元素本身及其子元素

$(ele).empty() 移除子元素

childNodes[]

parentNodes[]

odiv.offsetwidth $().width()

---------------------------------------------------------------------------------------------------------------------------------

文档碎片

DOM每次删除/添加节点都会重新渲染，可以先把dom操作放在文档碎片上，最后统一作用在DOM中。

Document.createDocumentFragmet

\*只是在低级浏览器ie6,ie7中有显著的效果，高级浏览器几乎看不到效果

\*几乎没有太大的用处

1. **DOM高级应用**
2. 表格操作

tBodies\tHead\tFoot\rows\cells

\*tHead\tFoot 是对象

oTab.getElementByTagName(‘tBody’)[0] == oTab.tBodies[0]

.getElementByTagName(‘tr’)[0] == .rows[0]

.getElementByTagName(‘td’)[0] == .cells[0]

1. 模糊搜索 str.search(‘’) str.split()
2. 添加删除
3. 排序 arr.sort()
4. **js事件基础**
5. event对象和事件冒泡
6. 什么事event对象---描述事件详细信息的一个对象
7. 鼠标位置clientX, 键盘按键
8. document的本质：document.childNodes[0].tagName == !doctype
9. 获取event对象（兼容性写法）

var oEvent = ev || event //ev是兼容ff要写function(ev), 现在高版本已弃用ev用event了

1. 事件流

事件冒泡-----发生事件后，一直往上传递，直到document//用处不多，经常带来困扰

1. 取消冒泡： oEvent.cancelBubble = true;
2. DOM事件流：发生事件后，一直往上传递，直到document
3. 鼠标事件
4. 可视区位置clientX, clientY

例子：跟随鼠标的DIv //\*基本上用到clientY的都要用到scrollTop不然会出错

1. 获取鼠标在页面的绝对位置

封装函数 getPos()

例子2：一串跟追鼠标的div //offsetClientX offsetClientY

1. 键盘事件
2. keycode //不同键盘上keycode是一样的

onkeydown //enter==13

1. 其他属性

ctrlKey 、shiftKey、altKey

1. **JS事件中级**
2. 默认行为
3. 默认行为---浏览器自带的东西都叫

例子：oncentextmenu 右键菜单

1. 阻止默认行为

普通写法：return false

例子：屏蔽右键菜单

例子2：只能输入数字的文本框 onkeydown //0--48 9--57

1. 拖拽
2. onmousedown() {  
    onmousemove(){

}

onmouseup(){

onmousedown = null

onmuseup = null

}  
}

//div太小 ----事件都加载document上

//FF下空div拖拽bug ----阻止默认事件

//防止拖拽出页面 ----修正位置 赋值为0

1. **JS事件高级应用**
2. 事件绑定------多个部门写oDIV.onclick = function(){}。会互相覆盖，用绑定就不会
3. ie方式 //ie10及以下

attachEvent(‘onclick’，function)

detachEvent(‘onclick’，function)

1. DOM方式 //ie10以上及其他浏览器

addEventListener(‘click’，function, 捕获)

removeEventListener(‘click’，function, 捕获)

1. 事件绑定封装
2. 绑定匿名函数会无法删除
3. 高级拖拽1
4. 复习拖拽原理 //距离 down move up
5. 限制范围
6. 对位置的判断

例子：不能拖出指定对象的DIV

1. 磁性吸附
2. 高级拖拽2
3. 图片拖拽

return false //chrome ff ie

1. 文字选中

i、return false //chrome ff ie9

ii、ie下拖动问题 --- 事件捕获 //ie9及以下

setCapture() 整个网页的所有事件都集中在一个对象上,其他对象的事件都没作用了

releaseCapture() 解除捕获，使其他对象事件都正常

1. 与DOM配合

i、带框的拖拽

ii、保留原位置的拖拽

1. 自定义滚动条

//计算比例

//去除y的滚动

//div3滚动高度的计算

1. **ajax基础**
2. 什么是服务器  
   a) 网页浏览过程分析

b) 如何配置自己的服务器程序（AMP）

1. 什么是ajax
2. 无刷新数据读取
3. 用户注册、在线聊天
4. 同步、异步
5. 使用ajax
6. 基础：请求并显示静态TXT文件

字符集编码 //网页是uft-8编码，那服务器中的文件也必须是utf-8编码,不然中文会乱

//所有的文件编码,必须统一不管是utf-8,还是GB-312。

缓存、阻止缓存 //ie下的缓存比较严重

//缓存原理：根据url来缓存的

//阻止缓存的方法在url后面加一个’？t=new Date().getTime()’

1. 动态数据：请求js或json文件

eval的使用 //从服务器读出来的都是字符串，用eval把字符串转换层js

DOM创建元素 // innerHTML

1. 局部刷新：请求并显示部分网页文件
2. AJAX原理

GET------用于获取数据（如：浏览器帖子）

POST-----用于上传数据（如：用户注册）

\*GET\POST的区别

// get 通过网址，post 不通过网址（最本质的区别）

// get容量小，post容量大（2G）

// get安全差，post相对好

// get有缓存，post没有缓存

1. **AJAX中级**

A、ajax四个步骤

手机-----创建Ajax对象

拨号-----链接到服务器

说---------发送请求

听---------接受返回值

B、自己实现一个ajax

1. var oAjax = new XMLHttpRequest() //非IE6浏览器

var oAjax = new ActiveXObject(‘Microsoft.XMLHTTP’) //IE6浏览器

//\*用没有定义的变量----报错 alert(a)

//\*用没有定义的属性----undefined alert(window.a)

1. oAjax.open(‘GET’, ‘a.TXT’, true) //方式、文件名、同步或异步

//同步--一件一件执行

//异步--多个事件一起执行

1. oAjax.send()
2. oAjax.onreadystatechange = function(){ //onreadystatechange服务器返回自动触发

//readyState 浏览器与服务器的交互状态

if(oAjax.readyState === 4){

//0 (未初始化)还没有调用open()方法

//1 (载入) 已调用send（）方法，正在发送请求

//2 (载入完成) send()方法完成，已收到全部响应内容

//3 (解析)正在解析响应内容 \*载入的内容可能是加密压缩了的，自己解析比较麻烦，所以交给浏览器解析

//4 (解析完成)响应内容解析完成，可以在客户端调用了

if(oAjax.status == 200) {//200 404

alert(‘读取成功’+ oAjax.responseText)

}else {

alert(‘读取失败’)

}

}

}

C、Ajax数据类型

a) 数据类型

什么叫数据类型

XML（老式程序在用）、大部分都是json

b) 字符集

所有文件字符集(文件编码)相同

1. **js面向对象基础**

A、面向对象的特点

1. 抽象：抓住核心问题，把主要特征，跟问题相关的特征抽出来
2. 封装：实现过程封装起来，给出接口
3. 继承：从父类继承一些属性、方法、子类。

多重继承。多个父类

多态：js中意义不大

B、对象的组成

方法（对象上）---函数

属性（对象上）---变量

\*this 当前的方法属于谁就指谁。闭包函数和匿名函数和箭头函数都属于window对象，里面的this指window

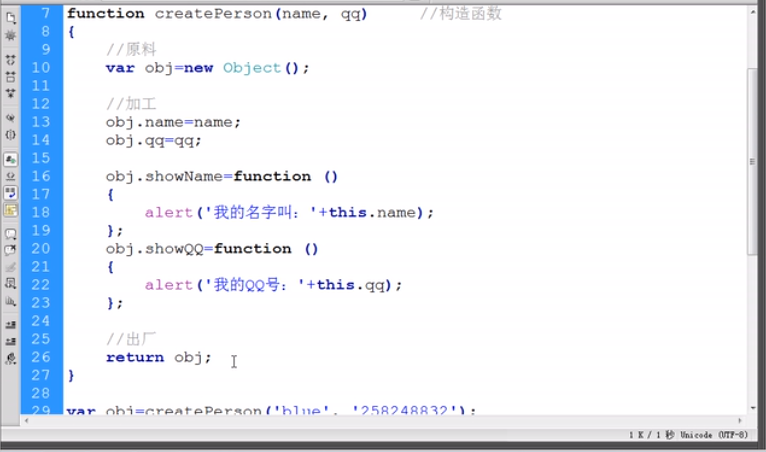
1. 为对象添加方法和属性

不能在系统（DATE）对象中随意附加方法、属性否则会覆盖已有的方法

Object对象

1. 第一个面向对象实例

工厂方法



工厂方法的问题：没有new； 每个实例的方法重复浪费资源

1. 函数和new

Show(){console.log(this)}

Show() //window

new Show() //Object

1. New 系统背后实现了this = new Object和return
2. prototype 解决 资源浪费问题 class 和 style的区别，prototype是一个指针，指向一对象，该对象包含该类实例对象共享的属性和方法。
3. 
4. 类和对象

类---模子 对象---产品

Arry.push() × arr.push()√

类没有实际功能 对象才有实际功能

1. 混合方式写类（流行的面向对象书写方式）

原则：构造函数加属性；原型加方法

类命名规范： 首字母大写

1. 总结什么是面向对象

只关注对象提供的功能，而不关注实现过程（面向过程）。

1. **js面向对象实例（写一个选项卡）**
2. 把一个面向过程的程序，改成一个面向过程的程序

原则:不能有函数嵌套，但是可以有全局变量

过程：

Onload -------> 构造函数

全局变量 -------> 属性

函数 -------> 方法

改错：this\*onclick=function(){this}\事件、闭包、传参

1. **JS面向对象高级**

JSON方式的面向对象（单体）

A、把方法包在一个json里面

* 1. 有人管他叫------命名空间
  2. 在公司里面，把同一类方法，包在一起。Baidu.zhidao=getUser() ; Baidu.map=getUser()

优点：简单；

缺点：没有new，不适用于多个对象

B.proptotype = A.prototype prototype是一个引用类型 //两个原型重合在一起了，子类prototype改了，父类prototype也改变。父类prototype改了，子类prototype也会改变

for(i in A.prototype) {  
 B.prototype[i] = A.prototype[i]  
} // 两个原型分开了，子类prototype改了，父类prototype不会变。父类prototype改了所有的子类prototype都变

1. 拖拽和继承

属性继承----构造函数+call

方法继承----prototype原型链

1. 系统对象
2. 本地对象（非静态对象） -------js运行环境无关（是js自身具备的）

什么叫做本地对象---需要通过new这个过程

Object、Function、Array、String、Number、Boolean、Error、Date、RegExp等

1. 内置对象（静态对象） -------js运行环境无关（是js自身具备的）

什么叫做内置对象----不用通过new这个过程

Global、Math对象

1. 宿主对象（由浏览器提供的对象）------js运行环境有关（浏览器）
2. **DOM、BOM**
3. BOM基础
4. 打开、关闭窗口

window.open //open有返回值

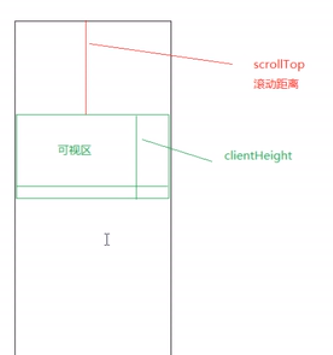
Window.close //兼容性火狐中open脚本打开的才能用close关

1. 常用属性

window.navigator.useAgent //判断浏览器、操作系统的类型

window.location //地址

1. 尺寸及坐标



1. 可视区尺寸

document.documentElement.clientWidth

window.innerHeight||document.documentElement.clientHeight

1. 滚动距离

document.body.scrollTop // chrome 最新chrome已经弃用了

document.documentElement.scrollTop // IE、FF、chrome

1. client元素可见相关属性(height,width)，对于document是可见的，但是ele就是ele自身的高度。

offset元素偏移属性(height,left,width,top,left,parent), Left, Top等都表示相对文档的偏移量。

scroll元素边缘和整体相关属性(height,left,width,top,left)，一般用在document对象上，ele对象上一般为0；

doc/ele.clientWdith == doc/ele.offsetWdith == doc/ele.scrollWdith //所有浏览器中都是

1. getBoundingClientRect(); (top,left,right,left只读属性)都是元素四边相对于可视区left和top的距离。
2. 常用的事件和方法
3. 系统对话框

alert()：没有返回值

confirm：返回boolean

prompt：返回字符串或null

1. window对象常用事件

onload\onscroll\onresize

1. **值传递和引用传递/赋值一样的适用**

基本类型是值传递

引用类型是引用传递 -----\*传整个数组是引用传递，传数组里面的一个对象是值传递

面试题

1. 什么是原型链
2. 什么是闭包
3. apply,call,bind的区别和共同

相同：① 都是改变对应函数中内this的指向；

② 三者 第一个参数为函数内部this想指的对象。当它为null 或 没有参数时 this指window。

③ 三者都利用后续参数传参

不同：

①立即执行--- apply,call 对应的函数会立即执fnc.call(obj)。bind会返回一个新的函数 fnc.bind(obj)()，并且参数可以在执行的时候添加。

②参数 ---- call和bind把参数按顺序传递进去，apply是把参数放在数组里面

应用：

1. 伪数组变数组 Array.prototype.slice
2. 数组最大值 Math.max() --- max是不接受数组参数的。
3. 自定义console.log() --- console内部实现用了this args是一个数组。
4. document.write() --- 调用的时候默认是window对象。
5. var foo = {

bar : 1,

eventBind: function(){

var \_this = this;

$('.someClass').on('click',function(event) {

console.log(\_this.bar);

});

}

}

var foo = {

bar : 1,

eventBind: function(){

$('.someClass').on('click',function(event) {

console.log(\_this.bar);

}.bind(this));

}

}

\*\*绑定函数时，bind可以执行多次，call/apply任何时候只立即执行一次，click时也没有效果

1. 遍历对象和遍历数组的方法
2. es6与lodash有哪些不同
3. es6箭头函数与es5的函数有什么区别
4. es6面向对象与es5面向对象实现实例化与继承
5. 数组的哪些方法要改变原来的数组
6. 数组排序，查找，去重
7. 实现一个懒加载
8. 函数柯里化
9. 时间戳-----就是把1970年01月1日到当前时间转化为毫秒

new Date().getTime() 或者 +new Date();

1. 为什么用redux?
2. react和jquery的区别