1 parseInt() 转换成整数 parseInt(变量/值)

parseInt(“123456”) 打印输出123456

parseInt(“12abc”) 打印输出12

注意 当变量或者值的是字符串或者. \* / 等， 输出的将是NaN

比如 parseInt(“abc123”) 打印输出的是NaN

但是parseInt(“-123abc”) 打印输出的是-123

parseInt(“+123abc”) 输出的是123

2 parseFloat() 转换成浮点型 parFloat(变量/值)

parseFloat(“123.45”) 输出的是123.45

注意小数点开始的值 parseFloat(“.1aa”) 输出的是0.1

parseFloat(“0.123”) 输出的是0.123

当遇到字符串时执行马上结束并输出结果

parseInt(“123ab123”) 输出123

parseFloat(“123.1abc56”) 输出123.1

window.prompt();输入提示框 该结果输出来的是字符 ，通过parseInt()或者parseFloat()转换成数值才能使用

var name=”a”

var age=18;

var result =name+age;

输出”a18”

任何数值遇到字符串相加时，得出的结果都是字符串

由此可以通过数值后面加”” 实现输出的是字符串

Result= age +”” 输出字符串”18”

比较运算符

== , ===, != ,!==

注意这三个的运算符

1==1 ：true 数值对比

1===1 ：true 数值和变量类型对比

1==’1’ ：true 数值对比

1===’1’ ：false 数值和变量类型全等才是true

5!=5 ：false

5!==5 ：false 根据字面意思理解 5本身是全等于5的 这里说5不全等于5 所以结果是false

5!==’5’ ：true 5不等于字符5 ，是正确的，所以true

2、有些值，放在条件中，会自动转换为false

if(0){}

if(0.0){}

if(null){}

if(undefined){}

if(NaN){}

if(""){}

if(''){}

除以上情况外，其余数值放在 条件中，都默认为真

if(123){} 真

if("Hello") 真

注意 if(“0”)或者if(“0.0”) 值为真

/////////////////////////////////////////////////////////////////

1. 获取页面元素
2. 节点间关系
3. 元素的内容
4. 元素的属性
5. 元素样式

获取节点，包括文本和元素 而获取元素只是获取元素，不获取元素外的文本

parentNode 只是获取上一级别的元素，并不是文本，而且只有一个元素

元素内容 innerHTML 读取元素的内容 包括换行，空格文本元素

元素属性 读取属性 element.attributes[下标].value

修改属性 element.setAttribute(name,value);同时可以用于创建属性

元素样式

获取内部、外部样式表

DOM:

document.defaultView.getComputedStyle("元素对象").样式属性;

IE :元素对象.currentStyle.属性

Element.style.属性名

注意要把属性中有字符（-），去掉（-）变成驼峰写法

element.style.backgroundColor

**数据类型转换**

**隐式转换 不需要程序员干预，js自动完成的类型转换**

**+运算中，有一个数据是字符串都自动转换为字符串**

**强制转换 主动调用函数实现的类型转换**

**toString String() 其中String 输入的可以是null和undefined 但toString不可以**

纯字符串相减得出的结果是数字（字符串值相减得出数字） ‘6’-‘2’=4

纯字符串相加得出的结果是字符串合起来 ‘2‘+‘3’=‘23’’

数字乘以数字字符串结果是数字 5\*’3’=15

数字减去数字字符串结果是数字 5-‘3’=2

不是数字字符串相减的结果是NaN ‘ASR ’- ‘ASDFASDF’=NaN

do{} while()与while 的区别

while是先判断后执行 do while 是执行一次再判断

如果有默认操作 那么用两者都可以，

如果没有默认操作，只能用while 因为do while 至少执行一次

eval 执行字符串格式的代码或表达式

var num=”1+2+3+4”;

var result=eval(num);

numd的结果是10 eval 执行了字符串的代码

eval还可以执行function

var fun=’function (){var a=1,b=2; console.log(a+b); }’

var func=eval(‘(’+fun+’)’)();

这个时候要在eval里加上括号字符；

通过for(var key in arr[]){console.log(arr[key])}来遍历哈希数组

Key相当于数组的下标

利用哈希数组去掉数组的重复元素

var arr[1,2,2,2,3,3,1,4,5,]

var res=[],hash=[];

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

if(!hash[arr[i]]){//判断哈希数组里对应的arr[i]下标名有没有值，如果有值则不存数组的值到res[]

hash[arr[i]]=1;//没有重复的时候写入1到hash数组，作为if的判断

res[res.length]=arr[i];

}

方法二 短路运算

！hash[arr[i]]&&(hash[arr[i]]=1,res[res.length]=arr[i]);

}

concat 不修改原数组而是返回新数组，必须用变量接住新数组

splice 是直接修改原数组

splice 删除，插入， 替换 :splice(1,1,0)第二个参数为1时，即可把对应下标的数值替换

split split() 方法用于把一个字符串分割成字符串数组。

比较字符串大小时，是按照字符在ASCII码表中对应的值大小进行比较的

str.lastIndexOf("敏感词"[,fromi])

fromi等于负数（不管负数是多少）时，lastInexOf将从0位置开始用搜索 慎用lastIndexOf

运算符类型

算术运算符 + - \* / % ++ --

关系运算符 > < >= <= == === != !===

逻辑运算符 && || ！

赋值运算符 = += -+ \*= /= %=

字符连接运算符 +

条件运算符 ?

var provs=["-请选择-","北京市","天津市","山东省","河南省"];

var html="<select><option>"

+provs.join("</option><option>")

+"</option></select>";

arr.sort(); 如果不传比较器函数 默认将每个元素转为字符串，再按升序排序 如果有比较强函数就按照比较器函数排序

函数内部声明变量的时候，一定要使用var命令。如果不用的话，实际上声明了一个全局变量！

**闭包**

**what** 有权访问另一个函数作用域的中的变量的函数（是一种写法）

在函数A里定义函数B，当函数B执行时访问了A的的内部变量，B就是一个闭包

how 在一个函数内部创建另一个函数

https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Closures

优点：保护了局部变量，避免全局污染

缺点：占用内存，容易造成内存泄漏

this

**谁调用，this就是谁**

Mozilla 链接

https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/this

o.b = {

g: independent,

prop: 42

};

console.log(o.b.g()); // logs 42

把一个方法g当作对象o.b的函数调用。在这次执行期间，函数中的this将指向o.b。事实上，这与对象本身的成员没有多大关系，最靠近的引用才是最重要的。

**面向对象**

**实例、构造函数、原型三者关系**

构造函数

实例

原型

constructor

new

prototype

\_\_proto\_\_

**继承**

**实现方式**

**1 修改构造函数的原型**

**2只修改一个对象的原型，不影响其他对象的原型**

**Object.setPrototypeOf(子对象,父级对象)**

**Math**

求最大值和最小值

Math.max(值1,值2,...)

Math.min(值1,值2,...)

如何获取数组中的最大值：固定套路

Math.max.apply(Math,arr);

随机数

Math.random() 0<=r**<**1的小数//注意不等于1

// 返回一个大于等于0，小于1的伪随机数

function getRandom() {

return Math.random();

}

// 返回一个介于min和max之间的随机数

function getRandomArbitrary(min, max) {

return Math.random() \* (max - min) + min;

}

// 返回一个介于min和max之间的整型随机数

// Using Math.round() will give you a non-uniform distribution!

function getRandomInt(min, max) {

return Math.floor(Math.random() \* (max - min + 1) + min);

}

如果min从0开始,可简写为：

Math.floor(Math.random()\*(max+1))

function Foo() {

getName = function () { alert (1); };

return this;

}

Foo.getName = function () { alert (2);};

Foo.prototype.getName = function () { alert (3);};

var getName = function () { alert (4);};

function getName() { alert (5);}

//答案：

Foo.getName();//2

getName();//4

Foo().getName();//1

getName();//1

new Foo.getName();//2

new Foo().getName();//3

new new Foo().getName();//3

**判断数据类型**

**几乎适合所有类型****Object.prototype.toString.call(判断的数据 ).indexOf(‘数据五大类型’s)!=-1 就是判断的数据类型**

**判断是否是数组：**

1. **Array.isArray(数据) 返回true 就是数组**
2. **数据 instanceof Array 返回true就是数组 obj除外**
3. **typeof 数组和obj判断都是obj**

**4 instanceof 和typeof 混合使用**

**string 、 object 、boolean 、null 四种数据 用2 .3. 4 方法**

**数组API**

**toString() 返回的是字符串**

//输出每个元素的内容，用逗号分隔

Array.prototype.slice.call() 类数组对象转换为数组

call 和apply 作用一样， apply传递的参数是**数组**

**.join(“连接字符”) 返回的是字符串**

**//输出每个元素的内容，使用自定义的连接符连接。如果省略连接符（arr.join()），则等效于toString**

**1. 无缝拼接数组为字符串：**

**var chars=['H','e','l','l','o']; //”Hello”**

**var word=chars.join("");**

**2. 将单词，拼接成句子：**

**var words=['no','zuo','no','die']**

**var stmt=words.join(" ");//”no zuo no die”**

**.concat**

**\*不修改原数组，而是返回新数组，必须用变量接住新数组**

**var newArr=arr1.concat(值1,值2,arr2,...)**

**获得子数组 .slice(starti ,endi+1)**

**starti: 开始位置的**下标

**endi: 结束位置的**下标**,含头不含尾**

**splice：删除 插入 替换 ——直接修改原数组**

**1. 删除：[var deleted=]arr.splice(starti,n)**

**starti表示开始删除的位置, n表示删除的个数**

**deleted表示被删除的元素，临时组成的子数组**

**2. 插入：arr.splice(starti,0,值1,值2,...)**

**starti表示开始插入的位置，**

**0表示一个都不删除**

**值1,...:要在starti位置插入的新值。**

**3. [var deleted=]arr.splice(starti,n,值1,值2,...)**

**谈谈html:**

**超文本标记语言**

**ｈ５新特性：新增视频音频标签 video audio**

**表单验证input type属性新增(email tel 、number、date)**

**localStorage（永久保存） 和 sessionStorage（关闭浏览器窗口就清除）**

**描述下 cookies ,sesionStorage和localStorage的区别**

**cookies 是为了标示用户身份而存储在用户本地的，始终在****同源的http（所谓的同源就是要求这个URL的协议，主机，端口三部分都相同）请求总携带， 数据大小不超过4k**

**sesionStorage和****localStorage不会自动把数据发给服务器 ，仅在本地保存储存数据可达到5M或者更大**

**sesionStorage关闭了浏览器窗口数据就会自动删除**

**localStorage浏览器关闭后，数据也不会自动删除，除非手动删除**

**href和src的区别**

**href 表示超文本引用（hypertext reference），在 link和a 等元素上使用。src 表示来源地址，在 img、script、iframe 等元素上。src 的内容，是页面必不可少的一部分，是引入。href 的内容，是与该页面有关联，是引用。区别就是，引入和引用。**

**谈谈css:**

**css3:**

**新增伪类选择器：nth-child（n）**

**圆角（border-radius）**

**transition 过渡 ，从一种样式过渡到另一种样式**

**transition: width 0.5s, height 2s ease-in;**

**动画animation 动画属性简写属性**

**#demo {**

**animation-name:demo;**

**animation-duration:2s;**

**}**

**@keyframes 规定动画**

**@keyframes testanimations {**

**from { opacity: 1; }**

**to { opacity: 0; }**

**}**

**谈谈javascript:**

**是什么：是一种运行于JavaScript引擎中的解释型脚本语言,**

**//由三部分组成ECMAScript、文档对象模型DOM、浏览器对象模型BOM**

**特点：代码可以使用任何编辑工具编写，一种弱类型语言，无需编译**

**编写地方：**

**1在浏览器控制台中输入脚本执行**

**2 将JS脚本嵌入到html页面中执行（html元素事件,在<script>标签中进行编写,通过引入外部的js文件**

**浏览器兼容性**

**chrome Safari 需要前缀-webkit-**

**Firefox -moz-**

**Opera –o-**

**用css hack 样式添加前缀**

**IE6 \_height “\_”下划线**

**IE7 \*height “\*”**

**javascript 兼容性**

**event事件 在IE 变为window.event**

**firefox 事件源event.target IE event.srcElement**

ajax new XMLHttpRequest() IE浏览器 new ActiveXObject(‘Microsoft.XMLHttp’)

**前端性能优化**

**1 压缩代码**

**3减少http请求**

**合拼css合拼js ，小图标合拼成一张图 通过移动background 的位置来显示小图，标，data嵌入图片 图片转换为base64编码后写到img 的src中**

**4减少DOM 操作 原生JS 大量dom操作 通过文档片段来加载**

**5 css 文件和js文件放到外部文件引用**

**6 CDN 加速 引用库时，使用官方的库链接 减少本地服务器压力**

**网页生成过程（浏览器渲染过程）**

1 html 代码转化为DOM

2 css 代码转化为CSSOM

3 结合DOM和CSSOM生成渲染树

4 生成布局（layout）

5 将布局绘制在屏幕上

**字符串API**

切割字符串：

var arr=str.split("分割符");//返回的是数组\*\*\*\*

切割后的数组中不包含分隔符的

固定套路：1. 将字符串打散成字符数组

var arr=str.split("")

2. 将英文句子，切割为单词

var arr=str.split(" ");

substring()

用于提取字符串中介于两个指定下标之间的字符。

返回的子串包括 *start* 处的字符，但不包括 *stop* 处的字符。

HTTP

**是什么**： 是一种将html文档从web服务器传送到客户端的浏览器通讯协议

**组成部分**：

协议(http/https /ftp):// <user>:<pwd>(用户名、密码)@<host>:

<port>(主机/端口)/<path>;<params>(路径)?<query>#<frag>(参数)(参数)

一共有5个版本 0.9 , 1.0 , 1.0+, 1.1 , 2.0

**请求和响应的流程**

客户端发送http请求 请求进行DNS(域名比如：www.baidu)查询 DNS服务器返回DNS响应解析出ip地址，然后进行HTTP请求 >>到达TCP层(传输层) >> IP(网络层)>>网络接口(数据访问层)

返回的时候向上层传输返回

**区别** https ：加密的http 就是在http层的下一层多了一个SSL or TLS 安全层

https网络端口是443 http网络端口是80

**http状态码：**

**100 服务器收到了请求初始部分，请客户的继续**

**200 响应完成，主体中包含请求的数据**

**201 PUT请求提交的数据已经被保存**

**301 请求资源已经被移除，location 响应指定了请求资源当前的url**

**303 告知客户端使用另一个url来获取资源**

**304 请求的资源未被修改**

**400 请求数据格式错误**

**403 请求被服务器拒绝了**

**404 请求资源不存在**

**405 发起的请求中带有url所不支持的方法**

**500 服务器端程序执行过程中发生错误**

**503 服务器端暂时无法处理请求**

AJAX

**是什么**：ajax就是异步的javascript 和XML

**优点** ：ajax能够在不刷新整个网页的前提下与服务器进行通讯，增加客户体验感，

可以减少服务器请求

缺点：不安全，暴露了与服务器交互的细节。

**如何使用ajax**：

1 先创建核心对象 就是（new XMLHttpRequest() IE浏览器 new ActiveXObject(‘Microsoft.XMLHttp’)）

2 与服务器建立连接 核心对象的open(method ,url ,是否异步)

3 向服务器发送数据send()

4 接收服务器端的响应内容 核心对象的onreadystatechange事件 readyState 属性状态是4 以及status==200（http状态码）时就是响应完毕

Jquery

**DOM对象转换为JQuery**

var obj=document.getElementById(‘d1‘);

var $obj=$(obj);

**JQuery 对象转换为DOM对象**

var $obj=$(‘#d1’);

var obj=$obj.get(0);

**子元素过滤选择器**

//不加空格表示：选择class为one的div并且该div是其父元素下第二个子元素

$("#ch:first").css("color", "red");

//加空格表示 选择class为one的div内的第二个子元素

$("#ch :nth-child(2)").css("background", "blue"); //nth-child是从1开始的 1就是子元素第一个

**jQuery UI**

button 样式

draggable 拖动 获取元素.draggable()

droppable 放置 获取元素.droppable();

控件

accordion 折叠 获取元素.accordion()

autoComplete 自动补全 input中打字自动找到数据库的内容补全， 获取元素 .autoComplete({source.[数据]})

Datepicker 日期 获取元素.datapicker();

**jQuery Mobile**

事件

pageinit DOM加载完执行

tab 触摸事件

tabhold 长按事件

swipe滑动事件

swipeleft 向左滑动事件

swiperight 向右滑动事件

scrollstart 滑动开始事件

scrollstop 滑动停止事件

orientationchange 设备方向改变（水平，垂直）触发

<div data-role=”page”>

<div data-role=”header”> </div>

<div role=”main”> </div>

<div data-role=”footer”> </div>

data

</div>

<label>

<input type=”checkbox”/>

</label>

ul –shadow 阴影 ui-btn 按钮 data-theme 主题 a 白色 b黑色

设置方向data-type:horizontal

**angular**

ng-show ,ng-if 的区别：ng-if会移除dom，生成dom，而ng-show只是改变其display属性

.directive (指令)

restrict指令在DOM里面以什么形式被声明

E(元素) A(属性) C(类) M(注释)

**replace**默认值为false，设置为true时不显示指令定义的名，而是显示template中模板内容

**scope**

可选参数，（布尔值或者对象）默认值为false，可能取值：

（1）默认值false。 表示继承父作用域

（2）true 表示继承父作用域，并创建自己的作用域（子作用域）

（3）{} 表示创建一个全新的隔离作用域

作用：可以隔离作用域,防止父作用域污染,创建课重用的组件时隔离作用域是个好选择

ng-bind-html

解析后台返回的各种各样的标签 **必须要使用$sce这个服务（严格的上下文模式）**

封装成过滤器

app.filter('to\_trusted', ['$sce', function ($sce) {

return function (text) {

return $sce.trustAsHtml(text);

};

}]);

在html中 使用 <div ng-bind-html="html | to\_trusted"></div>

**排序**

冒泡排序

var arr=[5,2,4,3,1,6];//冒泡排序

for(var i=1; i<arr.length; i++){

for(var j=arr.length-1; j>=I ;j--){

arr[j]<arr[j-1]&&(

arr[j]+=arr[j-1],

arr[j-1]=arr[j]-arr[j-1],

arr[j]-=arr[j-1]

)

}

}

console.log(arr);

快速排序

var arr2 = [1, 44, 67, 8, 98, 32, 5];

var quickSort = function(arr) {

if (arr.length <= 1) {

return arr;//当左右数组元素小于等于1时跳出递归

}

var pivoIndex = Math.floor(arr.length / 2);

var pivot = arr.splice(pivoIndex, 1)[0]; //找出基准元素 并从数组中删除 用splice是为了减小数组长度，提高运行效率

var left = [];

var right = [];

//遍历数组，小于基准元素的放左边数组，大于基准元素的放右边数组

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

if (arr[i] < pivot) {

left.push(arr[i]);

} else {

right.push(arr[i]);

}

}

//递归

return quickSort(left).concat([pivot], quickSort(right));

}

//arr2 原数组没有发生改变

var arrNew=quickSort(arr2);

console.log(arrNew);

**数组去重**

**var arr = [6, 6, 5, 2, 7, 7, 5];**

**var newArr = [];**

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

**var result = true;**

**for (var j = 0; j < newArr.length; j++) {**

**if (arr[i] == newArr[j]) {**

**result = false;**

**break;**

**}**

**}**

**if (result == true) {**

**newArr.push(arr[i]);**

**}**

**}**

**console.log(newArr);**

**//数组去重2 推荐使用**

**var arr2 = [5, 2, 2, 4, 6, 8, 6, 5];**

**var obj = {},**

**newArr = [];**

**for (var i = 0; i < arr2.length; i++) {**

**if (!obj[arr2[i]]) {**

**newArr.push(arr2[i]);**

**obj[arr2[i]] = 1;**

**}**

**}**