**2018年web前端经典面试题及答案**

**JavaScript中如何检测一个变量是一个String类型？请写出函数实现**

方法1、

function isString(obj){

return typeof(obj) === "string"? true: false;

// returntypeof obj === "string"? true: false;

}

方法2、

function isString(obj){

return obj.constructor === String? true: false;

}

方法3.

function isString(obj){

return Object.prototype.toString.call(obj) === "[object String]"?true:false;

}

如：

var isstring = isString('xiaoming');

console.log(isstring); // true

**请用js去除字符串空格？**

**方法一：使用replace正则匹配的方法**

去除所有空格: str = str.replace(/\s\*/g,"");

去除两头空格: str = str.replace(/^\s\*|\s\*$/g,"");

去除左空格： str = str.replace( /^\s\*/, “”);

去除右空格： str = str.replace(/(\s\*$)/g, "");

str为要去除空格的字符串，实例如下：

var str = " 23 23 ";

var str2 = str.replace(/\s\*/g,"");

console.log(str2); // 2323

**方法二：使用str.trim()方法**

str.trim()局限性：无法去除中间的空格，实例如下：

var str = " xiao ming ";

var str2 = str.trim();

console.log(str2); //xiao ming

同理，str.trimLeft()，str.trimRight()分别用于去除字符串左右空格。

**方法三：使用jquery,$.trim(str)方法**

$.trim(str)局限性：无法去除中间的空格，实例如下：

var str = " xiao ming ";

var str2 = $.trim(str)

console.log(str2); // xiao ming

**你如何获取浏览器URL中查询字符串中的参数？**

测试地址为：[http://www.runoob.com/jquery/misc-trim.html?channelid=12333&name=xiaoming&age=23](http://www.runoob.com/jquery/misc-trim.html?channelid=12333&name=xiaoming&age=23" \t "_blank)

实例如下：

function showWindowHref(){

var sHref = window.location.href;

var args = sHref.split('?');

if(args[0] == sHref){

return "";

}

var arr = args[1].split('&');

var obj = {};

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

var arg = arr[i].split('=');

obj[arg[0]] = arg[1];

}

return obj;

}

var href = showWindowHref(); // obj

console.log(href['name']); // xiaoming

**js 字符串操作函数**

　　我这里只是列举了常用的字符串函数，具体使用方法，。

* concat() – 将两个或多个字符的文本组合起来，返回一个新的字符串。
* indexOf() – 返回字符串中一个子串第一处出现的索引。如果没有匹配项，返回 -1 。
* charAt() – 返回指定位置的字符。
* lastIndexOf() – 返回字符串中一个子串最后一处出现的索引，如果没有匹配项，返回 -1 。
* match() – 检查一个字符串是否匹配一个正则表达式。
* substr() 函数 -- 返回从string的startPos位置，长度为length的字符串
* substring() – 返回字符串的一个子串。传入参数是起始位置和结束位置。
* slice() – 提取字符串的一部分，并返回一个新字符串。
* replace() – 用来查找匹配一个正则表达式的字符串，然后使用新字符串代替匹配的字符串。
* search() – 执行一个正则表达式匹配查找。如果查找成功，返回字符串中匹配的索引值。否则返回 -1 。
* split() – 通过将字符串划分成子串，将一个字符串做成一个字符串数组。
* length – 返回字符串的长度，所谓字符串的长度是指其包含的字符的个数。
* toLowerCase() – 将整个字符串转成小写字母。
* toUpperCase() – 将整个字符串转成大写字母。

**怎样添加、移除、移动、复制、创建和查找节点？**

 1）创建新节点

　　createDocumentFragment() //创建一个DOM片段  
　　createElement() //创建一个具体的元素  
　　createTextNode() //创建一个文本节点

2）添加、移除、替换、插入  
　　appendChild() //添加  
　　removeChild() //移除  
　　replaceChild() //替换  
　　insertBefore() //插入

3）查找  
　　getElementsByTagName() //通过标签名称  
　　getElementsByName() //通过元素的Name属性的值  
　　getElementById() //通过元素Id，唯一性

**写出3个使用this的典型应用**

（1）、在html元素事件属性中使用，如：

<input type=”button” onclick=”showInfo(this);” value=”点击一下”/>

（2）、构造函数

function Animal(name, color) {

　　this.name = name;

　　this.color = color;

}

（3）、input点击，获取值

<input type="button" id="text" value="点击一下" />

<script type="text/javascript">

var btn = document.getElementById("text");

btn.onclick = function() {

alert(this.value); //此处的this是按钮元素

}

</script>

(4)、apply()/call()求数组最值

var numbers = [5, 458 , 120 , -215 ];

var maxInNumbers = Math.max.apply(this, numbers);

console.log(maxInNumbers); // 458

var maxInNumbers = Math.max.call(this,5, 458 , 120 , -215);

console.log(maxInNumbers); // 458

**比较typeof与instanceof？**

**相同点：**JavaScript 中 typeof 和 instanceof 常用来判断一个变量是否为空，或者是什么类型的。

typeof的定义和用法：返回值是一个字符串，用来说明变量的数据类型。

**细节：**

(1)、typeof 一般只能返回如下几个结果：number,boolean,string,function,object,undefined。

(2)、typeof 来获取一个变量是否存在，如 if(typeof a!="undefined"){alert("ok")}，而不要去使用 if(a) 因为如果 a 不存在（未声明）则会出错。

(3)、对于 Array,Null 等特殊对象使用 typeof 一律返回 object，这正是 typeof 的局限性。

Instanceof定义和用法：instanceof 用于判断一个变量是否属于某个对象的实例。

**实例演示：**

a instanceof b?alert("true"):alert("false"); //a是b的实例？真:假

var a = new Array();

alert(a instanceof Array); // true

alert(a instanceof Object)  // true

如上，会返回 true，同时 alert(a instanceof Object) 也会返回 true;这是因为 Array 是 object 的子类。

function test(){};

var a = new test();

alert(a instanceof test) // true

**细节**：

(1)、如下，得到的结果为‘N’,这里的 instanceof 测试的 object 是指 js 语法中的 object，不是指 dom 模型对象。

if (window instanceof Object){ alert('Y')} else { alert('N');} // 'N'

**如何理解闭包？**

1、定义和用法：当一个函数的返回值是另外一个函数，而返回的那个函数如果调用了其父函数内部的其它变量，如果返回的这个函数在外部被执行，就产生了闭包。

2、表现形式：使函数外部能够调用函数内部定义的变量。

3、实例如下：

(1)、根据作用域链的规则，底层作用域没有声明的变量，会向上一级找，找到就返回，没找到就一直找，直到window的变量，没有就返回undefined。这里明显count 是函数内部的flag2 的那个count 。

var count=10; //全局作用域 标记为flag1

function add(){

var count=0; //函数全局作用域 标记为flag2

return function(){

count+=1; //函数的内部作用域

alert(count);

}

}

var s = add()

s();//输出1

s();//输出2

4、变量的作用域

要理解闭包，首先必须理解Javascript特殊的变量作用域。

变量的作用域分类：全局变量和局部变量。

特点：

1、函数内部可以读取函数外部的全局变量；在函数外部无法读取函数内的局部变量。

2、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，你实际上声明了一个全局变量！

 5、使用闭包的注意点

1）滥用闭包，会造成内存泄漏：由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。

2）会改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

**什么是跨域？跨域请求资源的方法有哪些？**

**1、什么是跨域？**

由于浏览器同源策略，凡是发送请求url的协议、域名、端口三者之间任意一与当前页面地址不同即为跨域。存在跨域的情况：

* 网络协议不同，如http协议访问https协议。
* 端口不同，如80端口访问8080端口。
* 域名不同，如qianduanblog.com访问baidu.com。
* 子域名不同，如abc.qianduanblog.com访问def.qianduanblog.com。
* 域名和域名对应ip,如www.a.com访问20.205.28.90.

**2、跨域请求资源的方法：**

**(1)、porxy代理**

定义和用法：proxy代理用于将请求发送给后台服务器，通过服务器来发送请求，然后将请求的结果传递给前端。

实现方法：通过nginx代理；

注意点：1、如果你代理的是https协议的请求，那么你的proxy首先需要信任该证书（尤其是自定义证书）或者忽略证书检查，否则你的请求无法成功。

**(2)、CORS 【Cross-Origin Resource Sharing】**

定义和用法：是现代浏览器支持跨域资源请求的一种最常用的方式。

使用方法：一般需要后端人员在处理请求数据的时候，添加允许跨域的相关操作。如下：

res.writeHead(200, {

"Content-Type": "text/html; charset=UTF-8",

"Access-Control-Allow-Origin":'http://localhost',

'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',

'Access-Control-Allow-Headers': 'X-Requested-With, Content-Type'

});

**(3)、jsonp**

定义和用法：通过动态插入一个script标签。浏览器对script的资源引用没有同源限制，同时资源加载到页面后会立即执行（没有阻塞的情况下）。

特点：通过情况下，通过动态创建script来读取他域的动态资源，获取的数据一般为json格式。

实例如下：

<script>

function testjsonp(data) {

console.log(data.name); // 获取返回的结果

}

</script>

<script>

var \_script = document.createElement('script');

\_script.type = "text/javascript";

\_script.src = "http://localhost:8888/jsonp?callback=testjsonp";

document.head.appendChild(\_script);

</script>

缺点：

　　1、这种方式无法发送post请求（这里）

　　2、另外要确定jsonp的请求是否失败并不容易，大多数框架的实现都是结合超时时间来判定。

**谈谈垃圾回收机制方式及内存管理**

**回收机制方式**

1、定义和用法：垃圾回收机制(GC:Garbage Collection),执行环境负责管理代码执行过程中使用的内存。

2、原理：垃圾收集器会定期（周期性）找出那些不在继续使用的变量，然后释放其内存。但是这个过程不是实时的，因为其开销比较大，所以垃圾回收器会按照固定的时间间隔周期性的执行。

3、实例如下：

function fn1() {

var obj = {name: 'hanzichi', age: 10};

}

function fn2() {

var obj = {name:'hanzichi', age: 10};

return obj;

}

var a = fn1();

var b = fn2();

fn1中定义的obj为局部变量，而当调用结束后，出了fn1的环境，那么该块内存会被js引擎中的垃圾回收器自动释放；在fn2被调用的过程中，返回的对象被全局变量b所指向，所以该块内存并不会被释放。

4、垃圾回收策略：标记清除(较为常用)和引用计数。

**标记清除：**

　　定义和用法：当变量进入环境时，将变量标记"进入环境"，当变量离开环境时，标记为："离开环境"。某一个时刻，垃圾回收器会过滤掉环境中的变量，以及被环境变量引用的变量，剩下的就是被视为准备回收的变量。

　　到目前为止，IE、Firefox、Opera、Chrome、Safari的js实现使用的都是标记清除的垃圾回收策略或类似的策略，只不过垃圾收集的时间间隔互不相同。

**引用计数：**

定义和用法：引用计数是跟踪记录每个值被引用的次数。

基本原理：就是变量的引用次数，被引用一次则加1，当这个引用计数为0时，被视为准备回收的对象。

**内存管理**

1、什么时候触发垃圾回收？

垃圾回收器周期性运行，如果分配的内存非常多，那么回收工作也会很艰巨，确定垃圾回收时间间隔就变成了一个值得思考的问题。

IE6的垃圾回收是根据内存分配量运行的，当环境中的变量，对象，字符串达到一定数量时触发垃圾回收。垃圾回收器一直处于工作状态，严重影响浏览器性能。

IE7中，垃圾回收器会根据内存分配量与程序占用内存的比例进行动态调整，开始回收工作。

2、合理的GC方案：(1)、遍历所有可访问的对象; (2)、回收已不可访问的对象。

3、GC缺陷：(1)、停止响应其他操作；

4、GC优化策略：(1)、分代回收（Generation GC）;(2)、增量GC

**开发过程中遇到的内存泄露情况，如何解决的？**

1、定义和用法：

内存泄露是指一块被分配的内存既不能使用，又不能回收，直到浏览器进程结束。C#和Java等语言采用了自动垃圾回收方法管理内存，几乎不会发生内存泄露。我们知道，浏览器中也是采用自动垃圾回收方法管理内存，但由于浏览器垃圾回收方法有bug，会产生内存泄露。

2、内存泄露的几种情况:

(1)、当页面中元素被移除或替换时，若元素绑定的事件仍没被移除，在IE中不会作出恰当处理，此时要先手工移除事件，不然会存在内存泄露。

实例如下:

<div id="myDiv">

<input type="button" value="Click me" id="myBtn">

</div>

<script type="text/javascript">

var btn = document.getElementById("myBtn");

btn.onclick = function(){

document.getElementById("myDiv").innerHTML = "Processing...";

}

</script>

解决方法如下：

<div id="myDiv">

<input type="button" value="Click me" id="myBtn">

</div>

<script type="text/javascript">

var btn = document.getElementById("myBtn");

btn.onclick = function(){

btn.onclick = null;

document.getElementById("myDiv").innerHTML = "Processing...";

}

</script>

(2)、由于是函数内定义函数，并且内部函数--事件回调的引用外暴了，形成了闭包。闭包可以维持函数内局部变量，使其得不到释放。

实例如下：

function bindEvent(){

var obj=document.createElement("XXX");

obj.onclick=function(){

//Even if it's a empty function

}

}

解决方法如下：

function bindEvent(){

var obj=document.createElement("XXX");

obj.onclick=function(){

//Even if it's a empty function

}

obj=null;

}

**javascript面向对象中继承实现？**

面向对象的基本特征有：封闭、继承、多态。

在JavaScript中实现继承的方法：

1. 原型链（prototype chaining）

2. call()/apply()

3. 混合方式(prototype和call()/apply()结合)

4. 对象冒充

继承的方法如下：

1、prototype原型链方式：

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

function teacher(name){

this.name = name;

}

teacher.prototype.sayName = function(){

console.log("name is "+this.name);

}

var teacher1 = new teacher("xiaoming");

teacher1.sayName();

function student(name){

this.name = name;

}

student.prototype = new teacher()

var student1 = new student("xiaolan");

student1.sayName();

// name is xiaoming

// name is xiaolan

2、call()/apply()方法

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

function teacher(name,age){

this.name = name;

this.age = age;

this.sayhi = function(){

alert('name:'+name+", age:"+age);

}

}

function student(){

var args = arguments;

teacher.call(this,args[0],args[1]);

// teacher.apply(this,arguments);

}

var teacher1 = new teacher('xiaoming',23);

teacher1.sayhi();

var student1 = new student('xiaolan',12);

student1.sayhi();

// alert: name:xiaoming, age:23

// alert: name:xiaolan, age:12

3、混合方法【prototype,call/apply】

function teacher(name,age){

this.name = name;

this.age = age;

}

teacher.prototype.sayName = function(){

console.log('name:'+this.name);

}

teacher.prototype.sayAge = function(){

console.log('age:'+this.age);

}

function student(){

var args = arguments;

teacher.call(this,args[0],args[1]);

}

student.prototype = new teacher();

var student1 = new student('xiaolin',23);

student1.sayName();

student1.sayAge();

// name:xiaolin

// age:23

4、对象冒充

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif View Code

**javascript相关程序计算题**

**1、判断一个字符串中出现次数最多的字符，统计这个次数**

var str = 'asdfssaaasasasasaa';

var json = {};

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

if(!json[str.charAt(i)]){

json[str.charAt(i)] = 1;

}else{

json[str.charAt(i)]++;

}

};

var iMax = 0;

var iIndex = '';

for(var i in json){

if(json[i]>iMax){

iMax = json[i];

iIndex = i;

}

}

console.log('出现次数最多的是:'+iIndex+'出现'+iMax+'次');

结果如下：出现次数最多的是:a出现9次

**JavaScript 数组(Array)对象**

**1、Array相关的属性和方法**

这里只是做了相关的列举，具体的使用方法，[请参考网址](http://www.w3school.com.cn/jsref/jsref_obj_array.asp" \t "_blank)。

**Array 对象属性**

constructor 返回对创建此对象的数组函数的引用。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var test=new Array();

if (test.constructor==Array){

document.write("This is an Array");

}

if (test.constructor==Boolean){

document.write("This is a Boolean");

}

if (test.constructor==Date){

document.write("This is a Date");

}

if (test.constructor==String){

document.write("This is a String");

}

length 设置或返回数组中元素的数目。

prototype 使您有能力向对象添加属性和方法。

**Array 对象方法**

concat() 连接两个或更多的数组，并返回结果。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = [1,2,3,4];

var arr2 = [5,6,7,8];

var arr3 = arr.concat(arr2);

console.log(arr3); // 连接之后返回的数组为：[1, 2, 3, 4, 5, 6, 7, 8]

join() 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = ['xiao','lin','qiqi','mingtian'];

var arr2 = arr.join(',');

console.log(arr2); // 根据','隔开返回的字符串为："xiao,lin,qiqi,mingtian"

pop() 删除并返回数组的最后一个元素。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = [2,3,4,5];

var arr2 = arr.pop();

console.log(arr2); // 删除的数组的最后一个元素为：5

console.log(arr); // 删除元素之后的数组为：[2, 3, 4]

shift() 删除并返回数组的第一个元素

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = [2,3,4,5];

var arr2 = arr.shift();

console.log(arr2); // 删除的数组的第一个元素为：2

console.log(arr); // 删除元素之后的数组为：[3, 4，5]

push() 向数组的末尾添加一个或更多元素，并返回新的长度。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = [2,3,4,5];

var arr2 = arr.push(6);

console.log(arr2); // 返回的数组长度：5

console.log(arr); // [2, 3, 4, 5, 6]

unshift() 向数组的开头添加一个或更多元素，并返回新的长度。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = ['xiao','ming','qiqi','aiming'];

var arr1 = arr.unshift('lang');

console.log(arr1); // 返回的数组的长度： 5

console.log(arr); //向数组开头添加元素返回的结果：["lang", "xiao", "ming", "qiqi", "aiming"]

reverse() 颠倒数组中元素的顺序。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = [2,3,4,5];

arr.reverse();

console.log(arr); // [5, 4, 3, 2]

slice() 从某个已有的数组返回选定的元素

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = [2,3,4,5];

var arr2 = arr.slice(1,3);

console.log(arr2); // 截取区间返回的数组为：[3, 4]

console.log(arr); // [2, 3, 4, 5]

sort() 对数组的元素进行排序

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

借助排序函数，实现数值由小到大排序

function sortNumber(a,b){

return a - b

}

var arr = [23,30,42,5];

var arr2 = arr.sort(sortNumber);

console.log(arr2); // [5, 23, 30, 42]

console.log(arr); // [5, 23, 30, 42]

借助排序函数，实现数值由大到小排序

function sortNumber(a,b){

return b - a

}

var arr = [23,30,42,5];

var arr2 = arr.sort(sortNumber);

console.log(arr2); // [42, 30, 23, 5]

console.log(arr); // [42, 30, 23, 5]

splice() 删除元素，并向数组添加新元素。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

语法:arrayObject.splice(index,howmany,item1,.....,itemX)

index:必需。整数，规定添加/删除项目的位置，使用负数可从数组结尾处规定位置。

howmany:必需。要删除的项目数量。如果设置为 0，则不会删除项目。

item1, ..., itemX:可选。向数组添加的新项目。

// 创建一个新数组，并向其添加一个元素

var arr = [1,2,3,4];

arr.splice(2,0,5);

console.log(arr); // [1, 2, 5, 3, 4]

// 删除位于 index 2 的元素，并添加一个新元素来替代被删除的元素：

var arr = [1,2,3,4];

arr.splice(2,1,5);

console.log(arr); // [1, 2, 5, 4]

toSource() 返回该对象的源代码。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

浏览器支持

只有 Gecko 核心的浏览器（比如 Firefox）支持该方法，也就是说 IE、Safari、Chrome、Opera 等浏览器均不支持该方法。

<script type="text/javascript">

function employee(name,job,born){

this.name=name;

this.job=job;

this.born=born;

}

var bill = new employee("Bill Gates","Engineer",1985);

document.write(bill.toSource());

</script>

输出：({name:"Bill Gates", job:"Engineer", born:1985})

toString() 把数组转换为字符串，并返回结果。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = ['xiao','ming','qiqi','aiming'];

arr.toString();

console.log(arr); // ["xiao", "ming", "qiqi", "aiming"]

toLocaleString() 把数组转换为本地数组，并返回结果。

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = ['xiao','ming','qiqi','aiming'];

arr.toLocaleString();

console.log(arr); // ["xiao", "ming", "qiqi", "aiming"]

valueOf() 返回数组对象的原始值

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

var arr = ['xiao','ming','qiqi','aiming'];

arr.valueOf('lang');

console.log(arr); // ["xiao", "ming", "qiqi", "aiming"]

**2、编写一个方法 去掉一个数组的重复元素**

方法一：

var arr = [0,2,3,4,4,0,2];

var obj = {};

var tmp = [];

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

if( !obj[arr[i]] ){

obj[arr[i]] = 1;

tmp.push(arr[i]);

}

}

console.log(tmp);

结果如下： [0, 2, 3, 4]

 方法二：

var arr = [2,3,4,4,5,2,3,6],

arr2 = [];

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

if(arr2.indexOf(arr[i]) < 0){

arr2.push(arr[i]);

}

}

console.log(arr2);

结果为：[2, 3, 4, 5, 6]

 方法三：

var arr = [2,3,4,4,5,2,3,6];  
var arr2 = arr.filter(function(element,index,self){  
return self.indexOf(element) === index;  
});  
console.log(arr2);

结果为：[2, 3, 4, 5, 6]

**3、求数组的最值？**

方法一：

* 求数组最大值：Math.max.apply(null,arr);

var arr = [3,43,23,45,65,90];  
var max = Math.max.apply(null,arr);  
console.log(max);  
// 90

* 求数组最小值：Math.min.apply(null,arr);

var arr = [3,43,23,45,65,90];  
var min = Math.min.apply(null,arr);  
console.log(min);  
// 3

方法二：Array.max = function(arr){} / Array.min = function(arr){}

var array = [3,43,23,45,65,90];

Array.max = function( array ){

return Math.max.apply( Math, array );

};

Array.min = function( array ){

return Math.min.apply( Math, array );

};

var max = Array.max(array);

console.log(max); // 90

var min = Array.min(array);

console.log(min); // 3

方法三：Array.prototype.max = function(){};Array.prototype.min = function(){};

* 求数组最大值(基本思路：将数组中的第一个值赋值给变量max ,将数组进行循环与max进行比较，将数组中的大值赋给max,最后返回max;)

var arr = [3,43,23,45,65,90];

Array.prototype.max = function() {

var max = this[0];

var len = this.length;

for (var i = 0; i < len; i++){

if (this[i] > max) {

max = this[i];

}

}

return max;

}

var max = arr.max();

console.log(max); // 90

* 求数组最小值：

var arr = [3,43,23,45,65,90];

Array.prototype.min = function() {

var min = this[0];

var len = this.length;

for(var i = 0;i< len;i++){

if(this[i] < min){

min = this[i];

}

}

return min;

}

var min = arr.min();

console.log(min); // 3

**4、数组排序相关**

    结合sort和函数排序：

* 数组由小到大进行排序：sort,sortnum;

var arr = [3,43,23,45,65,90];

function sortnum(a,b){

　　return a-b;

}

arr = arr.sort(sortnum);

console.log(arr);

// [3, 23, 43, 45, 65, 90]

* 数组由大到小进行排序：sort,sortnum;

var arr = [3,43,23,45,65,90];

function sortnum(a,b){

　　return a+b;

}

arr = arr.sort(sortnum);

console.log(arr);

// [90, 65, 45, 23, 43, 3]

冒泡排序：即实现数组由小到大进行排序；思路为：每次比较相邻的两个数，如果后一个比前一个小，换位置。如果要实现由大到小排序，使用reverse()即可；

var arr = [3, 1, 4, 6, 5, 7, 2];

function bubbleSort(arr) {

var len = arr.length;

for (var i = len; i >= 2; --i) {

for (var j = 0; j < i - 1; j++) {

if (arr[j + 1] < arr[j]) {

var temp;

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

return arr;

}

var arr2 = bubbleSort(arr);

console.log(arr2); // [1, 2, 3, 4, 5, 6, 7]

var arr3 = arr2.reverse();

console.log(arr3); //  [7, 6, 5, 4, 3, 2, 1]

快速排序：

思路：采用二分法，取出中间数，数组每次和中间数比较，小的放到左边，大的放到右边。

var arr = [3, 1, 4, 6, 5, 7, 2];

function quickSort(arr) {

if(arr.length == 0) {

return []; // 返回空数组

}

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

var c = arr.splice(cIndex, 1);

var l = [];

var r = [];

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

if(arr[i] < c) {

l.push(arr[i]);

} else {

r.push(arr[i]);

}

}

return quickSort(l).concat(c, quickSort(r));

}

console.log(quickSort(arr));

//[1, 2, 3, 4, 5, 6, 7]

**5、数组的翻转（非reverse()）**

方法一：

var arr = [1,2,3,4];

var arr2 = [];

while(arr.length) {

var num = arr.pop(); //删除数组最后一个元素并返回被删除的元素

arr2.push(num);

}

console.log(arr2);

// [4, 3, 2, 1]

方法二：

var arr = [1,2,3,4];

var arr2 = [];

while(arr.length){

var num = arr.shift(); //删除数组第一个元素并返回被删除的元素

arr2.unshift(num);

}

console.log(arr2);

**jQuery相关**

**1、jQuery 库中的 $() 是什么？**

$() 函数是 jQuery() 函数的别称。$() 函数用于将任何对象包裹成 jQuery 对象，接着你就被允许调用定义在 jQuery 对象上的多个不同方法。你可以将一个选择器字符串传入 $() 函数，它会返回一个包含所有匹配的 DOM 元素数组的 jQuery 对象。

**2、如何找到所有 HTML select 标签的选中项？**

$('[name=selectname] :selected')

**3、$(this) 和 this 关键字在 jQuery 中有何不同？**

$(this) 返回一个 jQuery 对象，你可以对它调用多个 jQuery 方法，比如用 text() 获取文本，用val() 获取值等等。

而 this 代表当前元素，它是 JavaScript 关键词中的一个，表示上下文中的当前 DOM 元素。你不能对它调用 jQuery 方法，直到它被 $() 函数包裹，例如 $(this)。

**4、jquery怎么移除标签onclick属性？**

获得a标签的onclick属性: $("a").attr("onclick")

删除onclick属性：$("a").removeAttr("onclick");

设置onclick属性：$("a").attr("onclick","test();");

**5、jquery中addClass,removeClass,toggleClass的使用。**

$(selector).addClass(class)：为每个匹配的元素添加指定的类名

$(selector).removeClass(class)：从所有匹配的元素中删除全部或者指定的类，删除class中某个值；

$(selector).toggleClass(class)：如果存在（不存在）就删除（添加）一个类

$(selector).removeAttr(class);删除class这个属性；

**6、JQuery有几种选择器?**

(1)、基本选择器：#id，class,element,\*;

(2)、层次选择器：parent > child，prev + next ，prev ~ siblings

(3)、基本[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8" \t "_blank)选择器：:first，:last ，:not ，:even ，:odd ，:eq ，:gt ，:lt

(4)、内容[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8" \t "_blank)选择器： :contains ，:empty ，:has ，:parent

(5)、可见性[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8" \t "_blank)选择器：:hidden ，:visible

(6)、属性过滤器选择器：[attribute] ，[attribute=value] ，[attribute!=value] ，[attribute^=value] ，[attribute$=value] ，[attribute\*=value]

(7)、子元素过滤器选择器：:nth-child ，:first-child ，:last-child ，:only-child

(8)、表单选择器： :input ，:text ，:password ，:radio ，:checkbox ，:submit 等；

(9)、表单过滤器选择器：:enabled ，:disabled ，:checked ，:selected

**7、jQuery中的Delegate()函数有什么作用？**

   delegate()会在以下两个情况下使用到：

 1、如果你有一个父元素，需要给其下的子元素添加事件，这时你可以使用delegate()了，代码如下：

$("ul").delegate("li", "click", function(){ $(this).hide(); });

 2、当元素在当前页面中不可用时，可以使用delegate()

**8、$(document).ready()方法和window.onload有什么区别？**

 (1)、window.onload方法是在网页中所有的元素(包括元素的所有关联文件)完全加载到浏览器后才执行的。

 (2)、$(document).ready() 方法可以在DOM载入就绪时就对其进行操纵，并调用执行绑定的函数。

**9、如何用jQuery禁用浏览器的前进后退按钮？**

实现代码如下：

<script type="text/javascript" language="javascript">

　　$(document).ready(function() {

　　　　window.history.forward(1);

　　　　//OR window.history.forward(-1);

　　});

</script>

**10、 jquery中$.get()提交和$.post()提交有区别吗？**

相同点：都是异步请求的方式来获取服务端的数据；

异同点：

1、请求方式不同：$.get() 方法使用GET方法来进行异步请求的。$.post() 方法使用POST方法来进行异步请求的。

2、参数传递方式不同：get请求会将参数跟在URL后进行传递，而POST请求则是作为HTTP消息的实体内容发送给Web服务器的，这种传递是对用户不可见的。

3、数据传输大小不同：get方式传输的数据大小不能超过2KB 而POST要大的多

4、安全问题： GET 方式请求的数据会被浏览器缓存起来，因此有安全问题。

**11、写出一个简单的$.ajax()的请求方式？**

$.ajax({

url:'http://www.baidu.com',

type:'POST',

data:data,

cache:true,

headers:{},

beforeSend：function(){},

success:function(){},

error:function(){},

complete:function(){}

});

**12、jQuery的事件委托方法bind 、live、delegate、on之间有什么区别？**

**(1)、bind 【jQuery 1.3之前】**

定义和用法：主要用于给选择到的元素上绑定特定事件类型的监听函数；

语法：bind(type,[data],function(eventObject))；

特点：

　　(1)、适用于页面元素静态绑定。只能给调用它的时候已经存在的元素绑定事件，不能给未来新增的元素绑定事件。

　　(2)、当页面加载完的时候，你才可以进行bind()，所以可能产生效率问题。

实例如下：$( "#members li a" ).bind( "click", function( e ) {} );

**(2)、live 【jQuery 1.3之后】**

定义和用法：主要用于给选择到的元素上绑定特定事件类型的监听函数；

语法：live(type, [data], fn);

特点：

　　(1)、live方法并没有将监听器绑定到自己(this)身上，而是绑定到了this.context上了。

　　(2)、live正是利用了事件委托机制来完成事件的监听处理，把节点的处理委托给了document，新添加的元素不必再绑定一次监听器。

　　(3)、使用live（）方法但却只能放在直接选择的元素后面，不能在层级比较深，连缀的DOM遍历方法后面使用，即$(“ul”").live...可以，但$("body").find("ul").live...不行；

实例如下：$( document ).on( "click", "#members li a", function( e ) {} );

**(3)、delegate 【jQuery 1.4.2中引入】**

定义和用法：将监听事件绑定在就近的父级元素上

语法：delegate(selector,type,[data],fn)

特点：

　　(1)、选择就近的父级元素，因为事件可以更快的冒泡上去，能够在第一时间进行处理。

　　(2)、更精确的小范围使用事件代理，性能优于.live()。可以用在动态添加的元素上。

实例如下：

$("#info\_table").delegate("td","click",function(){/\*显示更多信息\*/});

$("table").find("#info").delegate("td","click",function(){/\*显示更多信息\*/});

**(4)、on 【1.7版本整合了之前的三种方式的新事件绑定机制】**

定义和用法：将监听事件绑定到指定元素上。

语法：on(type,[selector],[data],fn)

实例如下：$("#info\_table").on("click","td",function(){/\*显示更多信息\*/});参数的位置写法与delegate不一样。

说明：on方法是当前JQuery推荐使用的事件绑定方法，附加只运行一次就删除函数的方法是one()。

 总结：.bind(), .live(), .delegate(),.on()分别对应的相反事件为：.unbind(),.die(), .undelegate(),.off()

**HTML & CSS:**

**1、什么是盒子模型？**

在网页中，一个元素占有空间的大小由几个部分构成，其中包括元素的内容（content），元素的内边距（padding），元素的边框（border），元素的外边距（margin）四个部分。这四个部分占有的空间中，有的部分可以显示相应的内容，而有的部分只用来分隔相邻的区域或区域。4个部分一起构成了css中元素的盒模型。

**2、行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？**

行内元素：a、b、span、img、input、strong、select、label、em、button、textarea  
块级元素：div、ul、li、dl、dt、dd、p、h1-h6、blockquote  
空元素：即系没有内容的HTML元素，例如：br、meta、hr、link、input、img

**3、CSS实现垂直水平居中**

一道经典的问题，实现方法有很多种，以下是其中一种实现：  
HTML结构：

<div class="wrapper">

<div class="content"></div>

</div>

CSS：

.wrapper {

position: relative;

width: 500px;

height: 500px;

border: 1px solid red;

}

.content{

position: absolute;

width: 200px;

height: 200px;

/\*top、bottom、left和right 均设置为0\*/

top: 0;

bottom: 0;

left: 0;

right: 0;

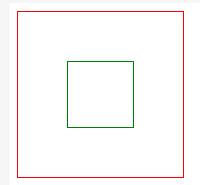
/\*margin设置为auto\*/

margin:auto;

border: 1px solid green;

}

效果如下：



**4、简述一下src与href的区别**

href 是指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的链接，用于超链接。

src是指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求src资源时会将其指向的资源下载并应用到文档内，例如js脚本，img图片和frame等元素。

当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行完毕，图片和框架等元素也如此，类似于将所指向资源嵌入当前标签内。这也是为什么将js脚本放在底部而不是头部。

**5、简述同步和异步的区别**

同步是阻塞模式，异步是非阻塞模式。  
同步就是指一个进程在执行某个请求的时候，若该请求需要一段时间才能返回信息，那么这个进程将会一直等待下去，直到收到返回信息才继续执行下去；  
异步是指进程不需要一直等下去，而是继续执行下面的操作，不管其他进程的状态。当有消息返回时系统会通知进程进行处理，这样可以提高执行的效率。

**6、px和em的区别**

相同点：px和em都是长度单位；

异同点：px的值是固定的，指定是多少就是多少，计算比较容易。em得值不是固定的，并且em会继承父级元素的字体大小。  
浏览器的默认字体高都是16px。所以未经调整的浏览器都符合: 1em=16px。那么12px=0.75em, 10px=0.625em。

**7、浏览器的内核分别是什么?**

IE: trident内核

Firefox：gecko内核

Safari：webkit内核

Opera：以前是presto内核，Opera现已改用Google Chrome的Blink内核

Chrome：Blink(基于webkit，Google与Opera Software共同开发)

**8、什么叫优雅降级和渐进增强？**

渐进增强 progressive enhancement：  
针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

优雅降级 graceful degradation：  
一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

区别：

a. 优雅降级是从复杂的现状开始，并试图减少用户体验的供给

b. 渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要

c. 降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带

**9、sessionStorage 、localStorage 和 cookie 之间的区别**

共同点：用于浏览器端存储的缓存数据

不同点：

(1)、存储内容是否发送到服务器端：当设置了Cookie后，数据会发送到服务器端，造成一定的宽带浪费；

        web storage,会将数据保存到本地，不会造成宽带浪费；

(2)、数据存储大小不同：Cookie数据不能超过4K,适用于会话标识；web storage数据存储可以达到5M;

(3)、数据存储的有效期限不同：cookie只在设置了Cookid过期时间之前一直有效，即使关闭窗口或者浏览器；

        sessionStorage,仅在关闭浏览器之前有效；localStorage,数据存储永久有效；

(4)、作用域不同：cookie和localStorage是在同源同窗口中都是共享的；sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；

**10、Web Storage与Cookie相比存在的优势：**

(1)、存储空间更大：IE8下每个独立的存储空间为10M，其他浏览器实现略有不同，但都比Cookie要大很多。

(2)、存储内容不会发送到服务器：当设置了Cookie后，Cookie的内容会随着请求一并发送的服务器，这对于本地存储的数据是一种带宽浪费。而Web Storage中的数据则仅仅是存在本地，不会与服务器发生任何交互。

(3)、更多丰富易用的接口：Web Storage提供了一套更为丰富的接口，如setItem,getItem,removeItem,clear等,使得数据操作更为简便。cookie需要自己封装。

(4)、独立的存储空间：每个域（包括子域）有独立的存储空间，各个存储空间是完全独立的，因此不会造成数据混乱。

**11、Ajax的优缺点及工作原理？**

**定义和用法:**

AJAX = Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）。Ajax 是一种用于创建快速动态网页的技术。Ajax 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。

传统的网页（不使用 Ajax）如果需要更新内容，必须重载整个网页页面。

**优点：**

1.减轻服务器的负担,按需取数据,最大程度的减少冗余请求

2.局部刷新页面,减少用户心理和实际的等待时间,带来更好的用户体验

3.基于xml标准化,并被广泛支持,不需安装插件等,进一步促进页面和数据的分离

**缺点：**

1.AJAX大量的使用了javascript和ajax引擎,这些取决于浏览器的支持.在编写的时候考虑对浏览器的兼容性.

2.AJAX只是局部刷新,所以页面的后退按钮是没有用的.

3.对流媒体还有移动设备的支持不是太好等

**AJAX的工作原理：**

1.创建ajax对象（XMLHttpRequest/ActiveXObject(Microsoft.XMLHttp)）

2.判断数据传输方式(GET/POST)

3.打开链接 open()

4.发送 send()

5.当ajax对象完成第四步（onreadystatechange）数据接收完成，判断http响应状态（status）200-300之间或者304（缓存）执行回调函数

**12、请指出document load和document ready的区别？**

共同点：这两种事件都代表的是页面文档加载时触发。

异同点：

ready 事件的触发，表示文档结构已经加载完成（不包含图片等非文字媒体文件）。

onload 事件的触发，表示页面包含图片等文件在内的所有元素都加载完成。

正则表达式

**1、写一个function，清除字符串前后的空格。（兼容所有浏览器）**

function trim(str) {

if (str && typeof str === "string") {

return str.replace(/(^\s\*)|(\s\*)$/g,""); //去除前后空白符

}

}

**2、使用正则表达式验证邮箱格式**

var reg = /^(\w)+(\.\w+)\*@(\w)+((\.\w{2,3}){1,3})$/;

var email = "example@qq.com";

console.log(reg.test(email)); // true

开发及性能优化

**1、规避javascript多人开发函数重名问题**

* 命名空间
* 封闭空间
* js模块化mvc（数据层、表现层、控制层）
* seajs
* 变量转换成对象的属性
* 对象化

**2、请说出三种减低页面加载时间的方法**

* 压缩css、js文件
* 合并js、css文件，减少http请求
* 外部js、css文件放在最底下
* 减少dom操作，尽可能用变量替代不必要的dom操作

**3、你所了解到的Web攻击技术**

（1）XSS（Cross-Site Scripting，跨站脚本攻击）：指通过存在安全漏洞的Web网站注册用户的浏览器内运行非法的HTML标签或者JavaScript进行的一种攻击。  
（2）SQL注入攻击  
（3）CSRF（Cross-Site Request Forgeries，跨站点请求伪造）：指攻击者通过设置好的陷阱，强制对已完成的认证用户进行非预期的个人信息或设定信息等某些状态更新。

**4、web前端开发，如何提高页面性能优化？**

**内容方面：**

1.减少 HTTP 请求 (Make Fewer HTTP Requests)

2.减少 DOM 元素数量 (Reduce the Number of DOM Elements)

3.使得 Ajax 可缓存 (Make Ajax Cacheable)

**针对CSS：**

1.把 CSS 放到代码页上端 (Put Stylesheets at the Top)

2.从页面中剥离 JavaScript 与 CSS (Make JavaScript and CSS External)

3.精简 JavaScript 与 CSS (Minify JavaScript and CSS)

4.避免 CSS 表达式 (Avoid CSS Expressions)

**针对JavaScript ：**

1. 脚本放到 HTML 代码页底部 (Put Scripts at the Bottom)

2. 从页面中剥离 JavaScript 与 CSS (Make JavaScript and CSS External)

3. 精简 JavaScript 与 CSS (Minify JavaScript and CSS)

4. 移除重复脚本 (Remove Duplicate Scripts)

**面向图片(Image)：**

1.优化图片

2 不要在 HTML 中使用缩放图片

3 使用恰当的图片格式

4 使用 CSS Sprites 技巧对图片优化

**5、前端开发中，如何优化图像？图像格式的区别？**

**优化图像：**

1、不用图片，尽量用css3代替。 比如说要实现修饰效果，如半透明、边框、圆角、阴影、渐变等，在当前主流浏览器中都可以用CSS达成。

2、 使用矢量图SVG替代位图。对于绝大多数图案、图标等，矢量图更小，且可缩放而无需生成多套图。现在主流浏览器都支持SVG了，所以可放心使用！

3.、使用恰当的图片格式。我们常见的图片格式有JPEG、GIF、PNG。

基本上，内容图片多为照片之类的，适用于JPEG。

而修饰图片通常更适合用无损压缩的PNG。

GIF基本上除了GIF动画外不要使用。且动画的话，也更建议用video元素和视频格式，或用SVG动画取代。

4、按照HTTP协议设置合理的缓存。

5、使用字体图标webfont、CSS Sprites等。

6、用CSS或JavaScript实现预加载。

7、WebP图片格式能给前端带来的优化。WebP支持无损、有损压缩，动态、静态图片，压缩比率优于GIF、JPEG、JPEG2000、PG等格式，非常适合用于网络等图片传输。

**图像格式的区别：**

矢量图：图标字体，如 font-awesome；svg

位图：gif,jpg(jpeg),png

区别：

　　1、gif:是是一种无损，8位图片格式。具有支持动画，索引透明，压缩等特性。适用于做色彩简单(色调少)的图片，如logo,各种小图标icons等。

　　2、JPEG格式是一种大小与质量相平衡的压缩图片格式。适用于允许轻微失真的色彩丰富的照片，不适合做色彩简单(色调少)的图片，如logo,各种小图标icons等。

　　3、png:PNG可以细分为三种格式:PNG8，PNG24，PNG32。后面的数字代表这种PNG格式最多可以索引和存储的颜色值。

关于透明：PNG8支持索引透明和alpha透明;PNG24不支持透明;而PNG32在24位的PNG基础上增加了8位（256阶）的alpha通道透明;

优缺点：

　　1、能在保证最不失真的情况下尽可能压缩图像文件的大小。

　　2、对于需要高保真的较复杂的图像，PNG虽然能无损压缩，但图片文件较大，不适合应用在Web页面上。

**6、浏览器是如何渲染页面的？**

**渲染的流程如下：**

1.解析HTML文件，创建DOM树。

   自上而下，遇到任何样式（link、style）与脚本（script）都会阻塞（外部样式不阻塞后续外部脚本的加载）。

2.解析CSS。优先级：浏览器默认设置<用户设置<外部样式<内联样式<HTML中的style样式；

3.将CSS与DOM合并，构建渲染树（Render Tree）

4.布局和绘制，重绘（repaint）和重排（reflow）