1.js中“5”+4=?

答案：54

2.js中void(0)=?

答案：undefined

3.js中NaN\*4=?

答案：NaN

4.js中null\*4.5=?

答案：0

5.js中alert(5\*015===5.075)

答案：false，结果不一样。

6.js中13>>2=? -13>>2=?

答案：3 ，-4 ，除以4，然后向下取整。

7.js中13|5=? 13&5=?

答案：按位或：13，按位与：5。

8.js中怎么获取当前日期的月份

答案：

<span style="font-size:18px;">var date = new Date(); var mouth = date.getMonth();</span>

9.js中数组排序方法是?该方法实现了什么的排序算法？

答案：排序方法是sort()，实现了按字符排序的算法。  
例var arr = [1,2,55,12,88];arr.sort();//ASCII字符代码从小到大排，arr结果为[1,12,2,55,88];

10.js中怎么判断chrome浏览器?

答案：

<span style="font-size:18px;">isChrome = window.navigator.userAgent.indexOf("Chrome") !== -1;//

当isChrome=true时，为chrome浏览器</span>

11.js中var b=”hello”;a=b; 怎么显示出a的值(貌似这题最简单了)

 答案：

<span style="font-size:18px;">document.write(a);</span>

12.根据以下xml请写出对应的json

<span style="font-size:18px;"><xml>

<list>

<item>

<id>12</id><name>张三</name>

</item>

<item><id>13</id><name>李四</name>

</item>

</list>

</xml></span>

答案：

<span style="font-size:18px;">var lists = [{"id":"12","name":"张三"},{"id":"13","name":"李四"}];</span>

13.js中怎么把十进制数123转化成二进制数?

答案：

<span style="font-size:18px;">123.toString(2);</span>

14.js中怎么才能按下回车键可以提交

<span style="font-size:18px;"><script type=”text/javascript”>

document.onkeydown = function(event){

event = event?event:window.event;

if (event.keyCode == 13) {

alert(“hello world!”);

}

};

</script></span>

**编程题**

1.js中var s=”tencent is sb”,编写js使其变成tencent1 is2 sb3

<span style="font-size:18px;"><script type="text/javascript">

var s = "tencent is perfect";

var array = s.split(" ");

s = "";

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

s += array[i] + (i+1) + " ";

}

document.write(s);

</script></span>

2.编写js的类，使其拥有public和private类型的属性和方法

<span style="font-size:18px;"><script type="text/javascript">

function Person(\_name,\_age,\_sex,\_salary){

//public

this.name = \_name;

this.age = \_age;

//privare

var sex = \_sex;

var salary = \_salary;

//public method

this.getName = function(){

return this.name;

}

this.getAge = function(){

return this.age;

}

//private methd

function getSex(){

return sex;

}

function getSalary(){

return salary;

}

this.display = function(){

document.write(this.getName() + "---" + this.getAge() + "---" + getSex() + "----" + getSalary());

}

}

var smirk = new Person("zy","21","f","5000");

smirk.display();

</script></span>

3.说出一些常用的网络优化工具

答：优化大师，超级兔子  
SEO（Search Engine Optimization）缩写而来， 中文意译为“搜索引擎优化”。  
SEO优化工具：  
1.TrafficTravis——SEO分析工具  
2.Backlinkwatch.com—反链检测  
3.XENU Link Sleuth—死链检测  
4.SEO Tool Bar (火狐插件)  
5.SEO Quake (火狐插件)

# 2015阿里巴巴前端实习生在线笔试题

前几天参加了2015阿里巴巴前端实习生在线笔试，发现题目真的挺难的，后来再重新回想的时候，除了本来就不会做的，还发现自己做错了几道题，现在整理一下那些笔试题，这里的笔试题部分是自己的，也有部分是工作室其他小伙伴们提供的。感谢楷豪师兄和闻东师兄的相关指导，感谢工作室其他小伙伴们提供的题目和答案。下面选择题给出的一些解析和答案都是自己上网查过相关资料然后根据自己的理解给出的，不保证全部正确哈，仅作参考作用。

## 一、（单项选择）对于下列程序运行结果，符合预期的是

function f1() {

console.time('time span');

}

function f2() {

console.timeEnd('time span');

}

setTimeout(f1, 100);

setTimeout(f2, 200);

function waitForMs(n) {

var now = Date.now();

while (Date.now() - now < n) {}

}

waitForMs(500);

* A、time apan:700.077ms
* B、time apan:0.066ms
* C、time apan:500.077ms
* D、time apan:100.077ms

解析：

console.time()语句和console.timeEnd()语句是用来对程序的执行进行计时的。因为f1和f2被都setTimeout()事先设置的定时器装到一个事件队列里面。本来f1应该在100ms后就要执行了，但是因为waitForMs()占用了线程，而执行JavaScript是单线程的，所以就没办法在100ms后执行那个f1，所以需要等500ms等waitForMs()执行完，然后再执行f1和f2，这时候f1和f2就几乎同时执行了。所以应该选时间最短的一项，所以答案应该选B **（楷豪师兄提供的解答）**

## 二、（单项选择）请选择结果为真的表达式

* A、null instanceof Object
* B、null == undefined
* C、NaN == NaN
* D、false == undefined

解析：

**A、**未定义的值和定义未赋值的为undefined，null是一种特殊的object，所以typeof null 返回的应该是object，但是为什么null instanceof Object返回的是false呢？原因就是null是个特殊的Object类型的值 ，表示空引用的意思 。instanceof表示某个变量是否是某个对象的实例 ,所以为false 。

**B、**undefined == null是正确的，尽管如此，和其他相似之处，但null和undefined并不是等价的。每个作为其独特的类型的唯一成员,undefined是Undefined类型和null是Null类型。所以undefined === null是不正确的，因为他们虽然值相等，但是类型不相等。区分这两个值，可以认为undefined代表一个意想不到的没有值而null作为预期没有值的代表。

**（null:）**是一个对象，但是为空。因为是对象，所以typeof null返回object 。null是JavaScript保留关键字。 null参与数值运算时其值会自动转换为0，因此，下列表达式计算后会得到正确的数值： 表达式：123 + null 结果值：123 表达式：123 \* null 结果值：0

**（undefined：）**是全局对象（window）的一个特殊属性，其值是未定义的。但typeof undefined返回undefined。

**C、**NaN是一个值类型,同是也是一个数值.意思是Not A Number,这个都知道是什么意思.值比较特殊,特殊在于NaN是一个数字,是一个与任何数值都不相等的数字。所以NaN == NaN返回false。

**D、**undefined被转换为布尔值为false，Boolean(undefined)返回的是false，但是undefined不等于false。所以false == undefined返回false。

所以最终答案应该为B。

## 三、（单项选择）下面程序的执行结果是

var name = 'World!';

(function () {

if (typeof name === 'undefined') {

var name = 'Jack';

console.log('Goodbye ' + name);

} else {

console.log('Hello ' + name);

}

})();

* A、Goodbye Jack
* B、Hello Jack
* C、Hello undefined
* D、Hello World

解析：

因为JavaScript中的变量的查找是就近原则去寻找var定义的变量，当就近没有找到的话就会找外层。题目中因为if判段语句(typeof name === 'undefined')就近定义的name就在其执行完的下一行，所以name就被预解析了，实际上可以理解成在if判段语句(typeof name === 'undefined')上面var name这样定义了name，但是尚未被赋值。而在它执行完后面再去为name赋值name = 'Jack';，所以name的值是undefined。所以typeof name === 'undefined'成立，所以判断语句会走if成立部分。 所以最终答案应该为A。

## 四、（不定项选择）以下关于DOM事件流的表述哪些是正确的

* A、事件流包括两个阶段：事件捕获阶段、事件冒泡阶段
* B、IE跟标准浏览器对于DOM事件流实现不一样
* C、假设parentEle是childEle的父节点，绑定事件：parentEle.addEventListener("click", fn1, false)和childEle.addEventListener("click", fn2,false),当点击childEle的时候fn1将先于fn2触发
* D、addEventListener第三个参数true代表支持捕获，false代表不支持捕获

解析：

在W3C事件模型中，任何事件会首先被捕获直至到达目标元素然后再冒泡回去。事件流包括3个阶段：事件捕获阶段、处于目标阶段和事件冒泡阶段。所以A选项是错的。Web开发者可以选择将事件处理程序注册在捕获或者冒泡阶段。这可以通过addEventListener()方法来实现。如果该方法传入的最后一个参数值为true，表示事件处理程序被注册在捕获阶段，如果为false表示件处理程序被注册在冒泡阶段。所以D选项也是错的。 假设有如下程序(childEle是parentEle的子元素)：

1.parentEle.addEventListener("click", parentDoSomething, true);

2.childEle.addEventListener("click", childEleDoSomething, false);

如果用户点击子元素childEle会发生如下事情：

a、点击事件开始于捕获阶段。它会先查询是否有childEle的任何祖先元素在捕获阶段绑定了onclick事件。

b、它发现祖先元素1在捕获阶段绑定了onclick事件，于是parentEle.parentDoSomething()首先被执行。

c、事件一直查询到目标元素childEle都没有再发现别的在捕获阶段绑定的onclick事件，事件转到它的冒泡阶段并执行childEleDoSomething()(注册在childEle上的在冒泡阶段执行的事件处理程序)。

d、事件再次向上查询并检查是否有任何祖先元素在冒泡阶段绑定了onclick事件，并没有查询到，所以什么都没有发生。

再看相反的例子：

1.parentEle.addEventListener("click", parentDoSomething, false);

2.childEle.addEventListener("click", childEleDoSomething, false);

现在如果用户点击childEle，下面的事情会按顺序发生：

a、点击事件发生于捕获阶段。事件查询childEle是否有任何祖先元素在捕获阶段绑定了onclick事件并且没有查找到这样的元素。

b、事件查询到目标元素childEle自己。事件转为冒泡阶段并执行childEleDoSomething()。

c、事件再次向上查询并检查目标元素是否有任何祖先元素在冒泡阶段绑定了onclick事件。

d、它找到了满足条件的parentEle，然后执行parentDoSomething()。所以选项C是错的，应该是fn2先触发。

因为IE没有提供对事件捕获阶段的支持，所以IE跟标准浏览器对于DOM事件流实现不一样。

所以最终答案应该为B。

## 五、（不定项选择）通过下面的哪些方法可以获取页面的html元素

* A、document.getElementById
* B、document.getElementsByClassName
* C、document.querySelector
* D、document.querySelectorAll

解析：

**A、**页面的html元素可以通过id获取，具有唯一性，如：var divObjId = document.getElementsById("test");。所以A是正确的。

**B、**页面的html元素可以通过class获取，会选择页面上所有class名为test的所有元素，返回的对象是NodeList。如：var divObjClass = document.getElementsByClassName("test");。所以B是正确的。

**C、D 、**document.querySelector只返回匹配的第一个元素，如果没有匹配项，返回null。document.querySelectorAll返回匹配的元素集合，如果没有匹配项，返回空的NodeList(节点数组)。这两个方法都可以接受三种类型的参数：id(#)，class(.)，标签，很像jquery的选择器。如：

var obj = document.querySelector("#id");

var obj = document.querySelector(".classname");

var obj = document.querySelector("div");

var el = document.body.querySelector("style[type='text/css'], style:not([type])");

var elements = document.querySelectorAll("#score>tbody>tr>td:nth-of-type(2)");

var elements = document.querySelectorAll("#id1, #id2, .class1, class2, div a, #list li img");

所以C和D都是正确的。

所以最终的答案是A,B,C,D。

## 六、（不定项选择）下面选项中，对javascript事件的描述不正确的是

* A、IE使用attachEvent/detachEvent方法来添加和删除事件监听器；w3c使用addEventListener/removeEventListener方法
* B、IE是将event对象作为参数传递给监听器,w3c事件监听器内使用的是一个全局的Event对象
* C、IE提供了对事件捕获阶段的支持
* D、要停止事件的传递，IE的做法是设置event对象的cancelBubble为true，而w3c的做法是设置执行stopPropagation方法

解析：

**A、**IE使用attachEvent/detachEvent方法来添加和删除事件监听器；w3c使用addEventListener/removeEventListener方法。这是正确的。

**B、**IE事件监听器内使用的是一个全局的Event对象，而w3c是将event对象作为参数传递给监听器。所以B是错误的。

**C、**IE没有提供对事件捕获阶段的支持。所以C也是错误的。

**D、**要想阻止冒泡，在Microsoft模型中，需要将事件的cancelBubble属性设置为true。在W3C模型中，需要调用事件的stopPropagation()方法。

这两种方法阻止了事件的所有冒泡。如果想解决浏览器兼容问题，可以像下面这样写：

function doSomething(e){

e = window.event || e;

e.cancelBubble = true;

if(e.stopPropagation){

e.stopPropagation();

}

}

在不支持cancelBubble属性的浏览器中设置它的值并不会报错。浏览器会忽略它并创建这个属性。当然，它并不能真正地阻止冒泡，但是给它分配值的操作本身是安全的。

所以最终的答案是B,C。

## 七、（单项选择）

var array1 = [1,2];

var array2 = array1;

array1[0] = array2[1];

array2.push(3);

console.log(array1);

console.log(array2);

执行上面的代码array1和array2的值分别是什么？

* A、Array1的值为[2,2];Array2的值为[1,2,3]
* B、Array1的值为[2,2,3];Array2的值为[1,2,3]
* C、Array1的值为[2,2,3];Array2的值为[2,2,3]
* D、Array1的值为[1,2,3];Array2的值为[1,2,3]

解析：

数组对象是引用的关系，array2改变，array1也会改变。array1改变，array2也会改变。

所以最终的答案是C。

## 八、（不定项选择）有如下代码

function Test(name,age){

this.name = name;

this.age = age;

};

Test.prototype = {

name:'aliyun',

hasOwnproperty:function(){

return false;

}

};

var instance = new Test('alibaba',102);

以下关于原型链的说法正确的是：

* A、JavaScript对象有两种不同的属性，一种是对象自身的属性，另一种是继承于原型链上的属性
* B、instance.name == 'aliyun'为true
* C、instance.hasOwnproperty('age')结果将是false
* D、所有对象都继承自Object.prototype

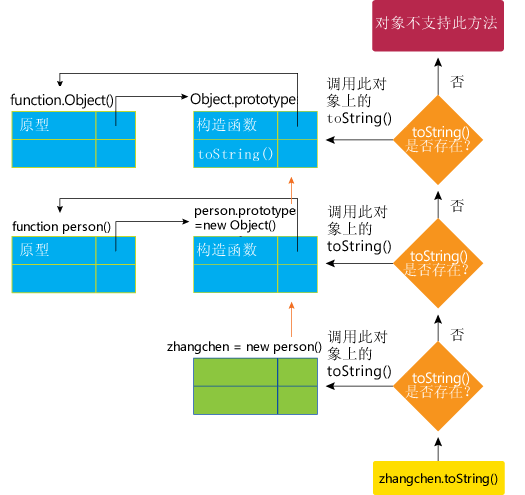
解析：

**A、**javascript对象有两种不同的属性来源，一个是对象自身属性，另一是继承于原型链上的属性,所以A是正确的。

**B、**因为instance是Test对象的一个实例，如果我们在该实例中创建了name这个属性，这个属性的值将会屏蔽原型中的那个属性。所以instance.name == 'aliyun'为false，instance.name的值应为alibaba，所以B是错误的。

**C、**因为instance是Test对象的一个实例，所以同样拥有hasOwnproperty这个方法，所以返回的结果是false.

**D、**每个JavaScript对象都继承一个原型链，而所有原型都终止于Object.prototype。注意，这种继承是活动对象之间的继承。它不同于继承的常见概念，后者是指在声明类时类之间的发生的继承。因此，JavaScript继承动态性更强。它使用简单算法实现这一点，如下所示：当您尝试访问对象的属性/方法时，JavaScript将检查该属性/方法是否是在该对象中定义的。如果不是，则检查对象的原型。如果还不是，则检查该对象的原型的原型，如此继续，一直检查到Object.prototype。下图说明了此解析过程

[](https://camo.githubusercontent.com/b6c6329a5cc63f0c826f101f12e1827d6af74846/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313530343034313534323432313337)

所以D是正确的。

所以最终的答案是A、C、D。

## 九、实现函数range([start,]stop[,step])返回一个数组（step大于1）

> range(1,11); => [1,2,3,4,5,6,7,8,9,10]

> range(0); => []

> range(10); => [0,1,2,3,4,5,6,7,8,9]

> range(0,30,5); => [0,5,10,15,20,25]

解析： 实现代码如下：

function range(){

var argLength = arguments.length,

newArray = [],

i = 0,

start = arguments[0],

stop = arguments[1],

step = arguments[2];

switch(argLength){

case 0 : throw Error('至少输入一个参数,限止数组在哪里结束');

case 1 : {

stop = arguments[0];

for( i = 0 ; i < stop ; i++){

newArray.push(i);

}

return newArray;

}

case 2 : {

for( i = start ; i < stop ; i++){

newArray.push(i);

}

return newArray;

}

case 3 : {

if(step < 1) {

throw Error('step > 1');

}else{

for( i = start ; i < stop ; i += step){

newArray.push(i);

}

return newArray;

}

}

default : throw Error('最多传入三个参数');

}

}

## 十、背景：

1、对象A直接调用对象B的某个方法，实现交互逻辑。但是导致的问题是A和B紧密耦合，修改B可能导致A调用B的方法失效。 2、为了解决耦合导致的问题，我们可以设计成： 对象A生成消息 -> 将消息通知给一个消息处理器（Observable）-> 消息处理器将消息传递给B 具体的调用过程变成： A.emit(‘message’,data); B.on(‘message’,function(data){}); 请实现这一事件消息代理功能 //请将事件消息功能补充完整 function EventEmitter(){}

解析：

实现代码如下：

function EventEmitter() {

this.eventFunctionMap = {};

}

EventEmitter.prototype.emit = function(eventName, data){

this.eventFunctionMap[eventName].call(this, data);

}

EventEmitter.prototype.on = function(eventName, callback) {

this.eventFunctionMap[eventName] = callback;

}

## 十一、实现下图的布局

<main>

<div>

A

</div>

<div>

B

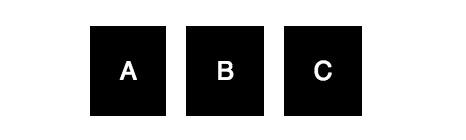
</div>

<div>

C

</div>

</main>

[](https://camo.githubusercontent.com/74efca00c5baee8cb84e773691a35fa03f413842/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313530343034313534323330343837)

解析： 实现代码如下：

div{

height: 50px;

width: 50px;

background-color: black;

display: inline-block;

margin-left: 10px;

color: white;

font-size: 20px;

text-align: center;

line-height: 50px;

}

下面是锦欣大神提供的另一种答案：

/\*\*base style\*\*/

div{

background:#000;

color:#fff;

height:50px;

width:50px;

text-align:center;

line-height:50px;

}

/\*\*start here\*\*/

main{

text-align:center;

}

div{

display:inline-block;

\_display:inline;/\*hank IE\*/

\_zoom:0;/\*hank IE\*/

}

## 十二、有一个包含数据列表的页面，数据行数不确定。每一行数据都有一个删除按钮，单击删除按钮删除该列数据，请用JavaScript实现该功能。

解析：

实现代码如下：

<script type="text/javascript">

window.onload = function(){

var oUl = document.getElementsByTagName('ul')[0];

oUl.onclick = function(ev){

var ev = ev || window.event;

var target = ev.target || ev.srcElement;

if(target.tagName.toLowerCase() == 'button'){

var tr = target.parentNode;

tr.parentNode.removeChild(tr);

}

}

}

</script>

<body>

<ul>

<li>一<button>删除一</button></li>

<li>二<button>删除二</button></li>

<li>三<button>删除三</button></li>

<li>四<button>删除四</button></li>

<li>五<button>删除五</button></li>

<li>六<button>删除六</button></li>

</ul>

</body>

## 十三、编写CSS让一个已知宽高的DIV,在PC/手机端水平垂直居中。

解析：

实现代码如下：

div{

width : 300px;

height : 300px;

border : 1px solid red;

position : absolute;

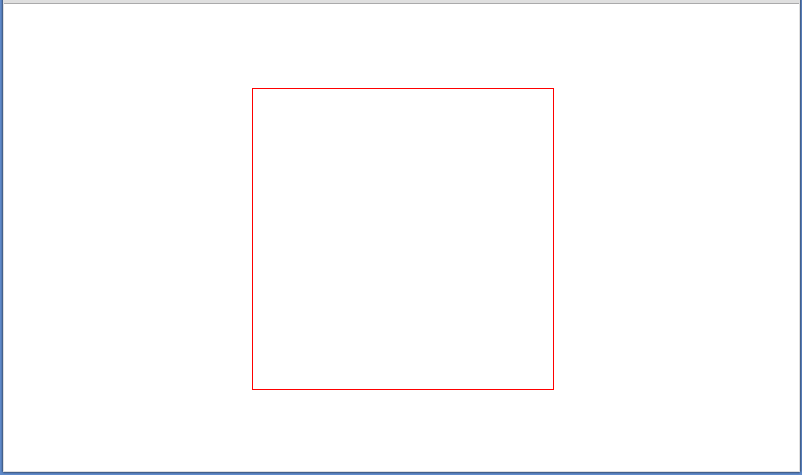
top : 50%;

left : 50%;

margin-top : -150px;

margin-left : -150px;

}

[](https://camo.githubusercontent.com/66d055655e7f97ba467fdea047d6d3a246f036f3/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313530343034313534333532353132)

## 十四、使用语义化的 HTML 标签及css完成以下布局

[这里写图片描述](https://camo.githubusercontent.com/b730cdb1e7be1f36e03119d8588f628435ecdb21/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313530343034313534343235313437)

• 容器默认宽度320px，图片100\*100

• hover 时容器宽度变为400px

• 右侧文字宽度自适应，考虑模块化和扩展性

解析：

实现代码如下：

<style type="text/css">

div{

width : 320px;

}

div:hover{

width : 400px;

}

img{

float : left;

width : 100px;

height : 100px;

}

h1{

color : #333;

margin-bottom : 8px;

font-size : 20px;

}

p{

color : #666;

font-size : 12px;

line-height: 1.2em;

}

</style>

<div>

<img src="d:\1.png">

<h1>(最多两行20px #333,顶部对齐图片，底部间距8px)</h1>

<p>(12px #666 行高1.2)使用语义化的HTML标签完成以下布局，考虑模块化和扩展性。容器默认为320px，右侧文字宽度自适应</p>

</div>

## 十五、写一个可以暂停执行的JavaScript函数

解析：

#### 1、使用函数闭包来实现

<input type="button" value="继续" onclick='st();'/>

<script>

/\*需要执行的函数\*/

function test(x){

alert(x++);

alert(x++);

return function(){

alert(x++);

}

}

var st = test(10);

</script>

#### 2、使用setTimeOut来实现

<script language="javascript">

/\*Javascript中暂停功能的实现

Javascript本身没有暂停功能（sleep不能使用）同时 vbscript也不能使用doEvents，故编写此函数实现此功能。

javascript作为弱对象语言，一个函数也可以作为一个对象使用。

比如：

function Test(){

alert("hello world");

this.NextStep=function(){

alert("NextStep");

}

}

我们可以这样调用 var myTest=new Test();myTest.NextStep();

我们做暂停的时候可以吧一个函数分为两部分，暂停操作前的不变，把要在暂停后执行的代码放在this.NextStep中。

为了控制暂停和继续，我们需要编写两个函数来分别实现暂停和继续功能。

暂停函数如下：

\*/

function Pause(obj,iMinSecond){

if (window.eventList==null) window.eventList=new Array();

var ind=-1;

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

if (window.eventList[i]==null) {

window.eventList[i]=obj;

ind=i;

break;

}

}

if (ind==-1){

ind=window.eventList.length;

window.eventList[ind]=obj;

}

setTimeout("GoOn(" + ind + ")",iMinSecond);

}

/\*

该函数把要暂停的函数放到数组window.eventList里，同时通过setTimeout来调用继续函数。

继续函数如下：

\*/

function GoOn(ind){

var obj=window.eventList[ind];

window.eventList[ind]=null;

if (obj.NextStep) obj.NextStep();

else obj();

}

/\*

该函数调用被暂停的函数的NextStep方法，如果没有这个方法则重新调用该函数。

函数编写完毕，我们可以作如下测试：

\*/

function Test(){

alert("hellow");

Pause(this,1000);//调用暂停函数

this.NextStep=function(){

alert("NextStep");

}

}

</script>

## 十六、用JavaScript写一个Ajax的get请求

解析：

/\* 创建 XMLHttpRequest 对象 \*/

var xmlHttp = null;

function GetXmlHttpObject(){

if (window.XMLHttpRequest){

// code for IE7+, Firefox, Chrome, Opera, Safari

Xmlhttp = new XMLHttpRequest();

}else{// code for IE6, IE5

Xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

}

return xmlhttp;

}

// -----------ajax方法-----------//

function getLabelsGet(){

xmlHttp = GetXmlHttpObject();

if (xmlHttp == null){

alert('您的浏览器不支持AJAX！');

return;

}

var id = document.getElementById('id').value;

var url="http://timtsang.github.io?id="+id+"&t/"+Math.random();

xmlHttp.open("GET",url,true);

xmlHttp.onreadystatechange=favorOK;//发送事件后，收到信息了调用函数

xmlHttp.send(null);

}

function getOkGet(){

if(xmlHttp.readyState == 1||xmlHttp.readyState == 2||xmlHttp.readyState == 3){

// 本地提示：加载中...

}else if (xmlHttp.readyState == 4){

if((xmlHttp.status >= 200 && xmlHttp.status < 300) || xmlHttp.status == 304){

var d = xmlHttp.responseText;

// 处理返回结果

}else{

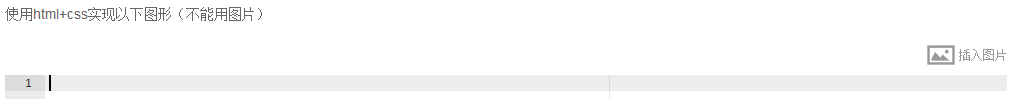
alert("Request was unsuccessful:" + xmlHttp.status);

}

}

}

## 十七、使用html+css实现以下图形（不能用图片）

[](https://camo.githubusercontent.com/d70e7cbb357cf22ecb4214b21ae90ff688519f50/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313530343034313534353439393737)

解析：

这道题感觉是做的最蛋疼的一道题，因为我之前根本就不理解题目到底要我实现哪个图形，狂刷新了N多遍网页，也没看到什么图形，后来经过闻东师兄提醒之后才恍然大悟，原来真的是要我实现右边那个插入图片的小图标。不过就算当初知道这要实现这个小图标，我也做不出来，因为我根本不知道那两个三角形要怎么实现。

<title>picture</title>

<style type="text/css">

#rectangle{

background: #FFF;

border: 10px solid #999999;

padding:10px;

height: 120px;

width: 200px;

}

#circle{

position:absolute;

margin-left: 140px;

margin-top: 10px;

width: 40px;

height: 40px;

background: #999999;

-moz-border-radius: 20px;

-webkit-border-radius: 20px;

border-radius: 20px;

}

#triangle{

width: 0;

height: 0;

position:absolute;

margin-top: 40px;

border-bottom: 80px solid #999999;

border-left: 50px solid transparent;

border-right: 70px solid transparent;

}

#triangle2{

width: 0;

height: 0;

position:absolute;

margin-top: 60px;

margin-left: 80px;

border-bottom: 60px solid #999999;

border-left: 30px solid transparent;

border-right: 70px solid transparent;

}

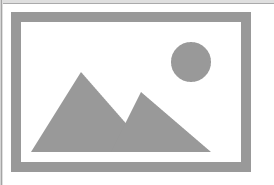
</style>

</head>

<div id="rectangle"><span id="circle"></span><span id="triangle"></span><span id="triangle2"></span></div>

</body>

效果图如下：

[](https://camo.githubusercontent.com/e60b5506614975394e6e273cf6925c59b6f11318/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313530343034313534353332383938)

## 十八、数组去重，然后使数组元素按降序排序

解析：

在排序的算法中，快速排序是相对比较高效的：

Array.prototype.quickSort = function(){

if(this.length <= 1){

return this;

}else{

var leftArray = [],

rightArray = [],

num = Math.floor((this.length)/2),

middleValue = this.splice(num,1);

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

if(this[i] > middleValue){

leftArray.push(this[i]);

}else{

rightArray.push(this[i]);

}

}

return leftArray.quickSort().concat(middleValue,rightArray.quickSort());

}

}

对于数组去重，有以下三种方法： 第一种：

Array.prototype.unique1 = function(){

var newArray = [this[0]];//结果数组

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

if(newArray.indexOf(this[i]) == -1){

newArray.push(this[i]);

}

}

return newArray;

}

第二种：

Array.prototype.unique2 = function(){

var newArray = [this[0]];//结果数组

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

if(this.indexOf(this[i]) == i){

newArray.push(this[i]);

}

}

return newArray;

}

第三种：

Array.prototype.unique3 = function(){

var arrayMap = {}, newArray = [];

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

if(!arrayMap[this[i]]){

arrayMap[this[i]] = true;

newArray.push(this[i]);

}

}

return newArray;

}

先对数组按降序排序，再去重便可实现题目的要求,这样效率也会高一点：

var newArray = [1,2,1,9,10,3,4,3,5,2,4,6,5,7,4,8,7,3,5,3];

var nowArray = newArray.quickSort().unique1();

console.log(nowArray);

## 十九、已知：

var data = {

name: 'xiaoming',

age: '18'

};

var template = "My name is {$name}, my age is {$age}".

//写一个函数，实现：

function shenmegui(template, data) {

} => My name is xiaoming, my age is 18.

解析：

var data = {

name:'xiaoming',

age:'18'

};

var template = "My name is {$name},my age is {$age}."

function outputTemplate(template,data){

var regName = /(**\{\$**name**\}**)/g;

var regAge = /(**\{\$**age**\}**)/g;

template = template.replace(regName,data.name);

template = template.replace(regAge,data.age);

return template;

}

alert(outputTemplate(template,data));

## 二十、利用闭包，实现点击<p>时，被点击的p的背景色变为#eee

解析：

window.onload = function(){

var pObj = document.getElementsByTagName("p");

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

pObj[i].onclick = (function(){

return function(){

this.style.background = "#eee";

}

})(i);

};

}

<p>第一个</p>

<p>第二个</p>

<p>第三个</p>

<p>第四个</p>

<p>第五个</p>

### 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;

background-color: #ddd;

}

.content{

background-color:#6699FF;

width:200px;

height:200px;

position: absolute; //父元素需要相对定位

top: 50%;

left: 50%;

margin-top:-100px ; //二分之一的height，width

margin-left: -100px;

}

### 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、什么是CSS Hack?

一般来说是针对不同的浏览器写不同的CSS,就是 CSS Hack。  
IE浏览器Hack一般又分为三种，条件Hack、属性级Hack、选择符Hack（详细参考CSS文档：[css文档](http://www.phpstudy.net/css3/)）。例如：

// 1、条件Hack

<!--[if IE]>

<style>

.test{color:red;}

</style>

<![endif]-->

// 2、属性Hack

.test{

color:#090\9; /\* For IE8+ \*/

\*color:#f00; /\* For IE7 and earlier \*/

\_color:#ff0; /\* For IE6 and earlier \*/

}

// 3、选择符Hack

\* html .test{color:#090;} /\* For IE6 and earlier \*/

\* + html .test{color:#ff0;} /\* For IE7 \*/

### 6、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。

### 7、px和em的区别

px和em都是长度单位，区别是，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

### 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、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)

[回到顶部](http://www.cnblogs.com/bigboyLin/p/5272902.html#_labelTop)

## JavaScript部分

### 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

### 实现一个函数clone，可以对JavaScript中的5种主要的数据类型（包括Number、String、Object、Array、Boolean）进行值复制。

/\*\*

\* 对象克隆

\* 支持基本数据类型及对象

\* 递归方法

\*/

function clone(obj) {

var o;

switch (typeof obj) {

case "undefined":

break;

case "string":

o = obj + "";

break;

case "number":

o = obj - 0;

break;

case "boolean":

o = obj;

break;

case "object": // object 分为两种情况 对象（Object）或数组（Array）

if (obj === null) {

o = null;

} else {

if (Object.prototype.toString.call(obj).slice(8, -1) === "Array") {

o = [];

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

o.push(clone(obj[i]));

}

} else {

o = {};

for (var k in obj) {

o[k] = clone(obj[k]);

}

}

}

break;

default:

o = obj;

break;

}

return o;

}

### 如何消除一个数组里面重复的元素？

// 方法一：

var arr1 =[1,2,2,2,3,3,3,4,5,6],

arr2 = [];

for(var i = 0,len = arr1.length; i< len; i++){

if(arr2.indexOf(arr1[i]) < 0){

arr2.push(arr1[i]);

}

}

document.write(arr2); // 1,2,3,4,5,6

### 在Javascript中什么是伪数组？如何将伪数组转化为标准数组？

伪数组（类数组）：无法直接调用数组方法或期望length属性有什么特殊的行为，但仍可以对真正数组遍历方法来遍历它们。典型的是函数的argument参数，还有像调用getElementsByTagName,document.childNodes之类的,它们都返回NodeList对象都属于伪数组。可以使用Array.prototype.slice.call(fakeArray)将数组转化为真正的Array对象。

function log(){

var args = Array.prototype.slice.call(arguments);

//为了使用unshift数组方法，将argument转化为真正的数组

args.unshift('(app)');

console.log.apply(console, args);

};

### Javascript中callee和caller的作用？

caller是返回一个对函数的引用，该函数调用了当前函数；

callee是返回正在被执行的function函数，也就是所指定的function对象的正文。

### 请描述一下cookies，sessionStorage和localStorage的区别

sessionStorage用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储。而localStorage用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

web storage和cookie的区别

Web Storage的概念和cookie相似，区别是它是为了更大容量存储设计的。Cookie的大小是受限的，并且每次你请求一个新的页面的时候Cookie都会被发送过去，这样无形中浪费了带宽，另外cookie还需要指定作用域，不可以跨域调用。  
除此之外，Web Storage拥有setItem,getItem,removeItem,clear等方法，不像cookie需要前端开发者自己封装setCookie，getCookie。但是Cookie也是不可以或缺的：Cookie的作用是与服务器进行交互，作为HTTP规范的一部分而存在 ，而Web Storage仅仅是为了在本地“存储”数据而生。

### 统计字符串中字母个数或统计最多字母数。

var str = "aaaabbbccccddfgh";

var obj = {};

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

var v = str.charAt(i);

if(obj[v] && obj[v].value == v){

obj[v].count = ++ obj[v].count;

}else{

obj[v] = {};

obj[v].count = 1;

obj[v].value = v;

}

}

for(key in obj){

document.write(obj[key].value +'='+obj[key].count+'&nbsp;'); // a=4 b=3 c=4 d=2 f=1 g=1 h=1

}

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

### 如何理解闭包？

### 跨域请求资源的方法有哪些？

### 谈谈垃圾回收机制方式及内存管理

### 开发过程中遇到的内存泄露情况，如何解决的？

[回到顶部](http://www.cnblogs.com/bigboyLin/p/5272902.html#_labelTop)

## HTTP

### 一次完整的HTTP事务是怎样的一个过程？

基本流程：

a. 域名解析

b. 发起TCP的3次握手

c. 建立TCP连接后发起http请求

d. 服务器端响应http请求，浏览器得到html代码

e. 浏览器解析html代码，并请求html代码中的资源

f. 浏览器对页面进行渲染呈现给用户

### HTTP的状态码有哪些？

### HTTPS是如何实现加密？

[回到顶部](http://www.cnblogs.com/bigboyLin/p/5272902.html#_labelTop)

## 算法相关

### 手写数组快速排序

关于快排算法的详细说明，可以参考阮一峰老师的文章[快速排序](http://www.ruanyifeng.com/blog/2011/04/quicksort_in_javascript.html)  
"快速排序"的思想很简单，整个排序过程只需要三步：  
（1）在数据集之中，选择一个元素作为"基准"（pivot）。  
（2）所有小于"基准"的元素，都移到"基准"的左边；所有大于"基准"的元素，都移到"基准"的右边。  
（3）对"基准"左边和右边的两个子集，不断重复第一步和第二步，直到所有子集只剩下一个元素为止。  
参考代码：

var quickSort = function(arr) {

　　if (arr.length <= 1) { return arr; }

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

　　var pivot = arr.splice(pivotIndex, 1)[0];

　　var left = [];

　　var right = [];

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

　　　　if (arr[i] < pivot) {

　　　　　　left.push(arr[i]);

　　　　} else {

　　　　　　right.push(arr[i]);

　　　　}

　　}

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

};

### JavaScript实现二分法查找

二分法查找，也称折半查找，是一种在有序数组中查找特定元素的搜索算法。查找过程可以分为以下步骤：  
（1）首先，从有序数组的中间的元素开始搜索，如果该元素正好是目标元素（即要查找的元素），则搜索过程结束，否则进行下一步。  
（2）如果目标元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半区域查找，然后重复第一步的操作。  
（3）如果某一步数组为空，则表示找不到目标元素。  
参考代码:

// 非递归算法

function binary\_search(arr, key) {

var low = 0,

high = arr.length - 1;

while(low <= high){

var mid = parseInt((high + low) / 2);

if(key == arr[mid]){

return mid;

}else if(key > arr[mid]){

low = mid + 1;

}else if(key < arr[mid]){

high = mid -1;

}else{

return -1;

}

}

};

var arr = [1,2,3,4,5,6,7,8,9,10,11,23,44,86];

var result = binary\_search(arr,10);

alert(result); // 9 返回目标元素的索引值

// 递归算法

function binary\_search(arr,low, high, key) {

if (low > high){

return -1;

}

var mid = parseInt((high + low) / 2);

if(arr[mid] == key){

return mid;

}else if (arr[mid] > key){

high = mid - 1;

return binary\_search(arr, low, high, key);

}else if (arr[mid] < key){

low = mid + 1;

return binary\_search(arr, low, high, key);

}

};

var arr = [1,2,3,4,5,6,7,8,9,10,11,23,44,86];

var result = binary\_search(arr, 0, 13, 10);

alert(result); // 9 返回目标元素的索引值

[回到顶部](http://www.cnblogs.com/bigboyLin/p/5272902.html#_labelTop)

## Web安全

### 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。

[回到顶部](http://www.cnblogs.com/bigboyLin/p/5272902.html#_labelTop)

## 前端性能

### 如何优化图像、图像格式的区别？

### 浏览器是如何渲染页面的？

[回到顶部](http://www.cnblogs.com/bigboyLin/p/5272902.html#_labelTop)

## 设计模式

### 对MVC、MVVM的理解

[回到顶部](http://www.cnblogs.com/bigboyLin/p/5272902.html#_labelTop)

## 正则表达式

### 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

}

}

### 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

## 前端开发面试知识点大纲：

HTML&CSS： 对Web标准的理解、浏览器内核差异、兼容性、hack、CSS基本功：布局、盒子模型、选择器优先级及使用、HTML5、CSS3、移动端适应。

JavaScript： 数据类型、面向对象、继承、闭包、插件、作用域、跨域、原型链、模块化、自定义事件、内存泄漏、事件机制、异步装载回调、模板引擎、Nodejs、JSON、ajax等。

其他： HTTP、安全、正则、优化、重构、响应式、移动端、团队协作、可维护、SEO、UED、架构、职业生涯

## 1.请你谈谈Cookie的弊端

cookie虽然在持久保存客户端数据提供了方便，分担了服务器存储的负担，但还是有很多局限性的。  
第一：每个特定的域名下最多生成20个cookie

1.IE6或更低版本最多20个cookie

2.IE7和之后的版本最后可以有50个cookie。

3.Firefox最多50个cookie

4.chrome和Safari没有做硬性限制

IE和Opera 会清理近期最少使用的cookie，Firefox会随机清理cookie。

cookie的最大大约为4096字节，为了兼容性，一般不能超过4095字节。

IE 提供了一种存储可以持久化用户数据，叫做uerData，从IE5.0就开始支持。每个数据最多128K，每个域名下最多1M。这个持久化数据放在缓存中，如果缓存没有清理，那么会一直存在。

### 优点：极高的扩展性和可用性

1.通过良好的编程，控制保存在cookie中的session对象的大小。

2.通过加密和安全传输技术（SSL），减少cookie被破解的可能性。

3.只在cookie中存放不敏感数据，即使被盗也不会有重大损失。

4.控制cookie的生命期，使之不会永远有效。偷盗者很可能拿到一个过期的cookie。

### 缺点：

1.`Cookie`数量和长度的限制。每个domain最多只能有20条cookie，每个cookie长度不能超过4KB，否则会被截掉。

2.安全性问题。如果cookie被人拦截了，那人就可以取得所有的session信息。即使加密也与事无补，因为拦截者并不需要知道cookie的意义，他只要原样转发cookie就可以达到目的了。

3.有些状态不可能保存在客户端。例如，为了防止重复提交表单，我们需要在服务器端保存一个计数器。如果我们把这个计数器保存在客户端，那么它起不到任何作用。

## 2.浏览器本地存储

在较高版本的浏览器中，js提供了sessionStorage和globalStorage。在HTML5中提供了localStorage来取代globalStorage。

html5中的Web Storage包括了两种存储方式：sessionStorage和localStorage。

sessionStorage用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储。

而localStorage用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

## 3.web storage和cookie的区别

Web Storage的概念和cookie相似，区别是它是为了更大容量存储设计的。Cookie的大小是受限的，并且每次你请求一个新的页面的时候Cookie都会被发送过去，这样无形中浪费了带宽，另外cookie还需要指定作用域，不可以跨域调用。

除此之外，Web Storage拥有setItem,getItem,removeItem,clear等方法，不像cookie需要前端开发者自己封装setCookie，getCookie。

但是Cookie也是不可以或缺的：Cookie的作用是与服务器进行交互，作为HTTP规范的一部分而存在 ，而Web Storage仅仅是为了在本地“存储”数据而生

浏览器的支持除了IE７及以下不支持外，其他标准浏览器都完全支持(ie及FF需在web服务器里运行)，值得一提的是IE总是办好事，例如IE7、IE6中的UserData其实就是javascript本地存储的解决方案。通过简单的代码封装可以统一到所有的浏览器都支持web storage。

localStorage和sessionStorage都具有相同的操作方法，例如setItem、getItem和removeItem等

## 4.CSS 相关问题

### display:none和visibility:hidden的区别？

display:none 隐藏对应的元素，在文档布局中不再给它分配空间，它各边的元素会合拢，

就当他从来不存在。

visibility:hidden 隐藏对应的元素，但是在文档布局中仍保留原来的空间。

### CSS中 link 和@import 的区别是？

A：(1) link属于HTML标签，而@import是CSS提供的; (2) 页面被加载的时，link会同时被加载，而@import引用的CSS会等到页面被加载完再加载;(3) import只在IE5以上才能识别，而link是HTML标签，无兼容问题; (4) link方式的样式的权重 高于@import的权重.

### position的absolute与fixed共同点与不同点

A：共同点：

1.改变行内元素的呈现方式，display被置为block；2.让元素脱离普通流，不占据空间；3.默认会覆盖到非定位元素上

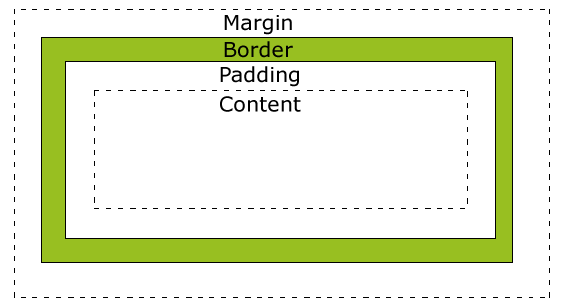
B不同点：

absolute的”根元素“是可以设置的，而fixed的”根元素“固定为浏览器窗口。当你滚动网页，fixed元素与浏览器窗口之间的距离是不变的。

### 介绍一下CSS的盒子模型？

1）有两种， IE 盒子模型、标准 W3C 盒子模型；IE的content部分包含了 border 和 pading;

2）盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border).

[](http://7u2qrr.com1.z0.glb.clouddn.com/blog_csshezimoxing.png)CSS盒子模型

### CSS 选择符有哪些？哪些属性可以继承？优先级算法如何计算？ CSS3新增伪类有那些？

\* 1.id选择器（ *# myid）*

2.类选择器（.myclassname）

3.标签选择器（div, h1, p）

4.相邻选择器（h1 + p）

5.子选择器（ul > li）

6.后代选择器（li a）

7.通配符选择器（ \* ）

8.属性选择器（a[rel = "external"]）

9.伪类选择器（a: hover, li:nth-child）

\* 可继承的样式： font-size font-family color, text-indent;

\* 不可继承的样式：border padding margin width height ;

\* 优先级就近原则，同权重情况下样式定义最近者为准;

\* 载入样式以最后载入的定位为准;

优先级为:

!important > id > class > tag

important 比 内联优先级高,但内联比 id 要高

CSS3新增伪类举例：

p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。

p:last-of-type 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。

p:only-of-type 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。

p:only-child 选择属于其父元素的唯一子元素的每个 <p> 元素。

p:nth-child(2) 选择属于其父元素的第二个子元素的每个 <p> 元素。

:enabled :disabled 控制表单控件的禁用状态。

:checked 单选框或复选框被选中。

### 列出display的值，说明他们的作用。position的值， relative和absolute分别是相对于谁进行定位的？

1.

block 象块类型元素一样显示。

inline 缺省值。象行内元素类型一样显示。

inline-block 象行内元素一样显示，但其内容象块类型元素一样显示。

list-item 象块类型元素一样显示，并添加样式列表标记。

2.

\*absolute

生成绝对定位的元素，相对于 static 定位以外的第一个祖先元素进行定位。

\*fixed （老IE不支持）

生成绝对定位的元素，相对于浏览器窗口进行定位。

\*relative

生成相对定位的元素，相对于其在普通流中的位置进行定位。

\* static 默认值。没有定位，元素出现在正常的流中

\*（忽略 top, bottom, left, right z-index 声明）。

\* inherit 规定从父元素继承 position 属性的值。

### CSS3有哪些新特性？

CSS3实现圆角（border-radius），阴影（box-shadow），

对文字加特效（text-shadow、），线性渐变（gradient），旋转（transform）

transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);*//旋转,缩放,定位,倾斜*

增加了更多的CSS选择器 多背景 rgba

在CSS3中唯一引入的伪元素是::selection.

媒体查询，多栏布局

border-image

### 为什么要初始化CSS样式。

因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对CSS初始化往往会出现浏览器之间的页面显示差异。

当然，初始化样式会对SEO有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化。

\*最简单的初始化方法就是： \* {padding: 0; margin: 0;} （不建议）

淘宝的样式初始化：

body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt, dd, ul, ol, li, pre, form, fieldset, legend, button, input, textarea, th, td { margin:0; padding:0; }

body, button, input, select, textarea { font:12px/1.5tahoma, arial, \5b8b\4f53; }

h1, h2, h3, h4, h5, h6{ font-size:100%; }

address, cite, dfn, em, var { font-style:normal; }

code, kbd, pre, samp { font-family:couriernew, courier, monospace; }

small{ font-size:12px; }

ul, ol { list-style:none; }

a { text-decoration:none; }

a:hover { text-decoration:underline; }

sup { vertical-align:text-top; }

sub{ vertical-align:text-bottom; }

legend { color:#000; }

fieldset, img { border:0; }

button, input, select, textarea { font-size:100%; }

table { border-collapse:collapse; border-spacing:0; }

### 对BFC规范的理解？

BFC，块级格式化上下文，一个创建了新的BFC的盒子是独立布局的，盒子里面的子元素的样式不会影响到外面的元素。在同一个BFC中的两个毗邻的块级盒在垂直方向（和布局方向有关系）的margin会发生折叠。

（W3C CSS 2.1 规范中的一个概念，它决定了元素如何对其内容进行布局，以及与其他元素的关系和相互作用。）

### 解释下 CSS sprites，以及你要如何在页面或网站中使用它。

CSS Sprites其实就是把网页中一些背景图片整合到一张图片文件中，再利用CSS的“background-image”，“background- repeat”，“background-position”的组合进行背景定位，background-position可以用数字能精确的定位出背景图片的位置。这样可以减少很多图片请求的开销，因为请求耗时比较长；请求虽然可以并发，但是也有限制，一般浏览器都是6个。对于未来而言，就不需要这样做了，因为有了`http2`。

## 5.html部分

### 说说你对语义化的理解？

1，去掉或者丢失样式的时候能够让页面呈现出清晰的结构

2，有利于SEO：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：爬虫依赖于标签来确定上下文和各个关键字的权重；

3，方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；

4，便于团队开发和维护，语义化更具可读性，是下一步吧网页的重要动向，遵循W3C标准的团队都遵循这个标准，可以减少差异化。

### Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?

（1）、*<!DOCTYPE>* 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器以何种模式来渲染文档。

（2）、严格模式的排版和 JS 运作模式是 以该浏览器支持的最高标准运行。

（3）、在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。

（4）、DOCTYPE不存在或格式不正确会导致文档以混杂模式呈现。

### 你知道多少种Doctype文档类型？

该标签可声明三种 DTD 类型，分别表示严格版本、过渡版本以及基于框架的 HTML 文档。

HTML 4.01 规定了三种文档类型：Strict、Transitional 以及 Frameset。

XHTML 1.0 规定了三种 XML 文档类型：Strict、Transitional 以及 Frameset。

Standards （标准）模式（也就是严格呈现模式）用于呈现遵循最新标准的网页，而 Quirks

（包容）模式（也就是松散呈现模式或者兼容模式）用于呈现为传统浏览器而设计的网页。

## 6.HTML与XHTML——二者有什么区别

区别：

1.所有的标记都必须要有一个相应的结束标记

2.所有标签的元素和属性的名字都必须使用小写

3.所有的XML标记都必须合理嵌套

4.所有的属性必须用引号""括起来

5.把所有<和&特殊符号用编码表示

6.给所有属性赋一个值

7.不要在注释内容中使“*--”*

8.图片必须有说明文字

## 7.常见兼容性问题？

\* png24位的图片在iE6浏览器上出现背景，解决方案是做成PNG8.也可以引用一段脚本处理.

\* 浏览器默认的margin和padding不同。解决方案是加一个全局的\*{margin:0;padding:0;}来统一。

\* IE6双边距bug:块属性标签float后，又有横行的margin情况下，在ie6显示margin比设置的大。

\* 浮动ie产生的双倍距离（IE6双边距问题：在IE6下，如果对元素设置了浮动，同时又设置了margin-left或margin-right，margin值会加倍。）

#box{ float:left; width:10px; margin:0 0 0 100px;}

这种情况之下IE会产生20px的距离，解决方案是在float的标签样式控制中加入 ——\_display:inline;将其转化为行内属性。(\_这个符号只有ie6会识别)

\* 渐进识别的方式，从总体中逐渐排除局部。

首先，巧妙的使用“\9”这一标记，将IE游览器从所有情况中分离出来。

接着，再次使用“+”将IE8和IE7、IE6分离开来，这样IE8已经独立识别。

css

.bb{

background-color:#f1ee18;*/\*所有识别\*/*

.background-color:#00deff\9; */\*IE6、7、8识别\*/*

+background-color:#a200ff;*/\*IE6、7识别\*/*

\_background-color:#1e0bd1;*/\*IE6识别\*/*

}

\* IE下,可以使用获取常规属性的方法来获取自定义属性,

也可以使用getAttribute()获取自定义属性;

Firefox下,只能使用getAttribute()获取自定义属性.

解决方法:统一通过getAttribute()获取自定义属性.

\* IE下,event对象有x,y属性,但是没有pageX,pageY属性;

Firefox下,event对象有pageX,pageY属性,但是没有x,y属性.

\* 解决方法：（条件注释）缺点是在IE浏览器下可能会增加额外的HTTP请求数。

\* Chrome 中文界面下默认会将小于 12px 的文本强制按照 12px 显示,

可通过加入 CSS 属性 -webkit-text-size-adjust: none; 解决.

\* 超链接访问过后hover样式就不出现了 被点击访问过的超链接样式不在具有hover和active了解决方法是改变CSS属性的排列顺序:

L-V-H-A : a:link {} a:visited {} a:hover {} a:active {}

\* 怪异模式问题：漏写DTD声明，Firefox仍然会按照标准模式来解析网页，但在IE中会触发怪异模式。为避免怪异模式给我们带来不必要的麻烦，最好养成书写DTD声明的好习惯。现在可以使用[html5](http://www.w3.org/TR/html5/single-page.html)推荐的写法：`<doctype html>`

\* 上下margin重合问题

ie和ff都存在，相邻的两个div的margin-left和margin-right不会重合，但是margin-top和margin-bottom却会发生重合。

解决方法，养成良好的代码编写习惯，同时采用margin-top或者同时采用margin-bottom。

\* ie6对png图片格式支持不好(引用一段脚本处理)

### 解释下浮动和它的工作原理？清除浮动的技巧

浮动元素脱离文档流，不占据空间。浮动元素碰到包含它的边框或者浮动元素的边框停留。

1.使用空标签清除浮动。

这种方法是在所有浮动标签后面添加一个空标签 定义css clear:both. 弊端就是增加了无意义标签。

2.使用overflow。

给包含浮动元素的父标签添加css属性 overflow:auto; zoom:1; zoom:1用于兼容IE6。

3.使用after伪对象清除浮动。

该方法只适用于非IE浏览器。具体写法可参照以下示例。使用中需注意以下几点。一、该方法中必须为需要清除浮动元素的伪对象中设置 height:0，否则该元素会比实际高出若干像素；

### 浮动元素引起的问题和解决办法？

浮动元素引起的问题：

（1）父元素的高度无法被撑开，影响与父元素同级的元素

（2）与浮动元素同级的非浮动元素会跟随其后

（3）若非第一个元素浮动，则该元素之前的元素也需要浮动，否则会影响页面显示的结构

解决方法：  
使用CSS中的clear:both;属性来清除元素的浮动可解决2、3问题，对于问题1，添加如下样式，给父元素添加clearfix样式：

.clearfix:after{content: ".";display: block;height: 0;clear: both;visibility: hidden;}

.clearfix{display: inline-block;} */\* for IE/Mac \*/*

**清除浮动的几种方法：**

1，额外标签法，<div style="clear:both;"></div>（缺点：不过这个办法会增加额外的标签使HTML结构看起来不够简洁。）

2，使用after伪类

*#parent:after{*

content:".";

height:0;

visibility:hidden;

display:block;

clear:both;

}

3,浮动外部元素

4,设置`overflow`为`hidden`或者auto

### IE 8以下版本的浏览器中的盒模型有什么不同

IE8以下浏览器的盒模型中定义的元素的宽高不包括内边距和边框

### DOM操作——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()

removeChild()

replaceChild()

insertBefore() *//在已有的子节点前插入一个新的子节点*

（3）查找

getElementsByTagName() *//通过标签名称*

getElementsByName() *//通过元素的Name属性的值(IE容错能力较强，会得到一个数组，其中包括id等于name值的)*

getElementById() *//通过元素Id，唯一性*

### html5有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何区分 HTML 和 HTML5？

\* HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加。

\* 拖拽释放(Drag and drop) API

语义化更好的内容标签（header,nav,footer,aside,article,section）

音频、视频API(audio,video)

画布(Canvas) API

地理(Geolocation) API

本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失；

sessionStorage 的数据在浏览器关闭后自动删除

表单控件，calendar、date、time、email、url、search

新的技术webworker, websocket, Geolocation

\* 移除的元素

纯表现的元素：basefont，big，center，font, s，strike，tt，u；

对可用性产生负面影响的元素：frame，frameset，noframes；

支持HTML5新标签：

\* IE8/IE7/IE6支持通过document.createElement方法产生的标签，

可以利用这一特性让这些浏览器支持HTML5新标签，

浏览器支持新标签后，还需要添加标签默认的样式：

\* 当然最好的方式是直接使用成熟的框架、使用最多的是html5shim框架

*<!--[if lt IE 9]>*

*<script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>*

*<![endif]-->*

如何区分： DOCTYPE声明\新增的结构元素\功能元素

## 8.iframe的优缺点？

1.<iframe>优点：

解决加载缓慢的第三方内容如图标和广告等的加载问题

Security sandbox

并行加载脚本

2.<iframe>的缺点：

\*iframe会阻塞主页面的Onload事件；

\*即时内容为空，加载也需要时间

\*没有语意

## 9.如何实现浏览器内多个标签页之间的通信?

调用localstorge、cookies等本地存储方式

## 10.webSocket如何兼容低浏览器？

Adobe Flash Socket 、 ActiveX HTMLFile (IE) 、 基于 multipart 编码发送 XHR 、 基于长轮询的 XHR

## 11.线程与进程的区别

一个程序至少有一个进程,一个进程至少有一个线程.

线程的划分尺度小于进程，使得多线程程序的并发性高。

另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。

线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。

从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。

## 12.你如何对网站的文件和资源进行优化？

期待的解决方案包括：

文件合并

文件最小化/文件压缩

使用 CDN 托管

缓存的使用（多个域名来提供缓存）

其他

## 13.请说出三种减少页面加载时间的方法。

1.优化图片

2.图像格式的选择（GIF：提供的颜色较少，可用在一些对颜色要求不高的地方）

3.优化CSS（压缩合并css，如margin-top,margin-left...)

4.网址后加斜杠（如www.campr.com/目录，会判断这个“目录是什么文件类型，或者是目录。）

5.标明高度和宽度（如果浏览器没有找到这两个参数，它需要一边下载图片一边计算大小，如果图片很多，浏览器需要不断地调整页面。这不但影响速度，也影响浏览体验。

当浏览器知道了高度和宽度参数后，即使图片暂时无法显示，页面上也会腾出图片的空位，然后继续加载后面的内容。从而加载时间快了，浏览体验也更好了。）

6.减少http请求（合并文件，合并图片）。

## 14.你都使用哪些工具来测试代码的性能？

Profiler, JSPerf（http:*//jsperf.com/nexttick-vs-setzerotimeout-vs-settimeout）, Dromaeo*

## 15.什么是 FOUC（无样式内容闪烁）？你如何来避免 FOUC？

FOUC - Flash Of Unstyled Content 文档样式闪烁

<style type="text/css" media="all">@import "../fouc.css";</style>

而引用CSS文件的@import就是造成这个问题的罪魁祸首。IE会先加载整个HTML文档的DOM，然后再去导入外部的CSS文件，因此，在页面DOM加载完成到CSS导入完成中间会有一段时间页面上的内容是没有样式的，这段时间的长短跟网速，电脑速度都有关系。

解决方法简单的出奇，只要在<head>之间加入一个<link>或者<script>元素就可以了。

## 16.null和undefined的区别？

null是一个表示”无”的对象，转为数值时为0；undefined是一个表示”无”的原始值，转为数值时为NaN。

当声明的变量还未被初始化时，变量的默认值为undefined。  
null用来表示尚未存在的对象，常用来表示函数企图返回一个不存在的对象。

undefined表示”缺少值”，就是此处应该有一个值，但是还没有定义。典型用法是：

（1）变量被声明了，但没有赋值时，就等于undefined。

（2) 调用函数时，应该提供的参数没有提供，该参数等于undefined。

（3）对象没有赋值的属性，该属性的值为undefined。

（4）函数没有返回值时，默认返回undefined。

null表示”没有对象”，即该处不应该有值。典型用法是：

（1） 作为函数的参数，表示该函数的参数不是对象。

（2） 作为对象原型链的终点。

## 17.new操作符具体干了什么呢?

1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。

2、属性和方法被加入到 this 引用的对象中。

3、新创建的对象由 this 所引用，并且最后隐式的返回 this 。

var obj = {};

obj.\_\_proto\_\_ = Base.prototype;

Base.call(obj);

## 18.JSON 的了解？

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。

它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小

{'age':'12', 'name':'back'}

## 19.js延迟加载的方式有哪些？

defer和async、动态创建DOM方式（创建script，插入到DOM中，加载完毕后callBack）、按需异步载入js

## 20.如何解决跨域问题?

jsonp、 document.domain+iframe、window.name、window.postMessage、服务器上设置代理页面

jsonp的原理是动态插入script标签

具体参见：[详解js跨域问题](http://segmentfault.com/blog/trigkit4/1190000000718840)

## 21.documen.write和 innerHTML的区别

document.write只能重绘整个页面

innerHTML可以重绘页面的一部分

## 22. .call() 和 .apply() 的区别和作用？

作用：动态改变某个类的某个方法的运行环境。  
区别参见：[JavaScript学习总结（四）function函数部分](http://segmentfault.com/blog/trigkit4/1190000000660786#articleHeader15)

## 23.哪些操作会造成内存泄漏？

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。

setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。

闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）

详见：[详解js变量、作用域及内存](http://segmentfault.com/blog/trigkit4/1190000000687844)

## 24.JavaScript中的作用域与变量声明提升？

详见：[详解JavaScript函数模式](http://segmentfault.com/blog/trigkit4/1190000000758184#articleHeader5)

## 25.如何判断当前脚本运行在浏览器还是node环境中？

通过判断Global对象是否为window，如果不为window，当前脚本没有运行在浏览器中

## 26.其他问题？

### 99%的网站都需要被重构是那本书上写的？

\* 网站重构：应用web标准进行设计（第2版）

### 什么叫优雅降级和渐进增强？

优雅降级：Web站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，则代码会检查以确认它们是否能正常工作。由于IE独特的盒模型布局问题，针对不同版本的IE的hack实践过优雅降级了,为那些无法支持功能的浏览器增加候选方案，使之在旧式浏览器上以某种形式降级体验却不至于完全失效.

渐进增强：从被所有浏览器支持的基本功能开始，逐步地添加那些只有新式浏览器才支持的功能,向页面增加无害于基础浏览器的额外样式和功能的。当浏览器支持时，它们会自动地呈现出来并发挥作用。

详见：[css学习归纳总结（一）](http://segmentfault.com/blog/trigkit4/1190000000800711#articleHeader30)

### 对Node的优点和缺点提出了自己的看法？

\*（优点）因为Node是基于事件驱动和无阻塞的，所以非常适合处理并发请求，

因此构建在Node上的代理服务器相比其他技术实现（如Ruby）的服务器表现要好得多。

此外，与Node代理服务器交互的客户端代码是由javascript语言编写的，

因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。

\*（缺点）Node是一个相对新的开源项目，所以不太稳定，它总是一直在变，

而且缺少足够多的第三方库支持。看起来，就像是Ruby/Rails当年的样子。

### 对前端界面工程师这个职位是怎么样理解的？它的前景会怎么样？

前端是最贴近用户的程序员，比后端、数据库、产品经理、运营、安全都近。

1、实现界面交互

2、提升用户体验

3、有了Node.js，前端可以实现服务端的一些事情

前端是最贴近用户的程序员，前端的能力就是能让产品从 90分进化到 100 分，甚至更好，

参与项目，快速高质量完成实现效果图，精确到1px；

与团队成员，UI设计，产品经理的沟通；

做好的页面结构，页面重构和用户体验；

处理hack，兼容、写出优美的代码格式；

针对服务器的优化、拥抱最新前端技术。

## 27.你有哪些性能优化的方法？

（[详情请看雅虎14条性能优化原则](http://segmentfault.com/blog/trigkit4/1190000000656717)）。

（1） 减少http请求次数：CSS Sprites, JS、CSS源码压缩、图片大小控制合适；网页Gzip，CDN托管，data缓存 ，图片服务器。

（2） 前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数

（3） 用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。

（4） 当需要设置的样式很多时设置className而不是直接操作style。

（5） 少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。

（6） 避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)。

（7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

## 28.http状态码有那些？分别代表是什么意思？

100-199 用于指定客户端应相应的某些动作。

200-299 用于表示请求成功。

300-399 用于已经移动的文件并且常被包含在定位头信息中指定新的地址信息。

400-499 用于指出客户端的错误。400 1、语义有误，当前请求无法被服务器理解。401 当前请求需要用户验证 403 服务器已经理解请求，但是拒绝执行它。

500-599 用于支持服务器错误。 503 – 服务不可用

详情：<http://segmentfault.com/blog/trigkit4/1190000000691919>

## 29.一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？

分为4个步骤：

（1），当发送一个URL请求时，不管这个URL是Web页面的URL还是Web页面上每个资源的URL，浏览器都会开启一个线程来处理这个请求，同时在远程DNS服务器上启动一个DNS查询。这能使浏览器获得请求对应的IP地址。

（2）， 浏览器与远程Web服务器通过TCP三次握手协商来建立一个TCP/IP连接。该握手包括一个同步报文，一个同步-应答报文和一个应答报文，这三个报文在 浏览器和服务器之间传递。该握手首先由客户端尝试建立起通信，而后服务器应答并接受客户端的请求，最后由客户端发出该请求已经被接受的报文。

（3），一旦TCP/IP连接建立，浏览器会通过该连接向远程服务器发送HTTP的GET请求。远程服务器找到资源并使用HTTP响应返回该资源，值为200的HTTP响应状态表示一个正确的响应。

（4），此时，Web服务器提供资源服务，客户端开始下载资源。

请求返回后，便进入了我们关注的前端模块

简单来说，浏览器会解析HTML生成DOM Tree，其次会根据CSS生成CSS Rule Tree，而javascript又可以根据DOM API操作DOM

详情：[从输入 URL 到浏览器接收的过程中发生了什么事情？](http://segmentfault.com/blog/trigkit4/1190000000697254)

## 30.平时如何管理你的项目？

先期团队必须确定好全局样式（globe.css），编码模式(utf-8) 等；

编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；

标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；

页面进行标注（例如 页面 模块 开始和结束）；

CSS跟HTML 分文件夹并行存放，命名都得统一（例如style.css）；

JS 分文件夹存放 命名以该JS功能为准的英文翻译。

图片采用整合的 images.png png8 格式文件使用 尽量整合在一起使用方便将来的管理

## 31.说说最近最流行的一些东西吧？常去哪些网站？

Node.js、Mongodb、npm、MVVM、MEAN、three.js,React 。

网站：w3cfuns,sf,hacknews,CSDN,慕课，博客园，InfoQ,w3cplus等

## 32.javascript对象的几种创建方式

1，工厂模式

2，构造函数模式

3，原型模式

4，混合构造函数和原型模式

5，动态原型模式

6，寄生构造函数模式

7，稳妥构造函数模式

## 33.javascript继承的6种方法

1，原型链继承

2，借用构造函数继承

3，组合继承(原型+借用构造)

4，原型式继承

5，寄生式继承

6，寄生组合式继承

详情：[JavaScript继承方式详解](http://segmentfault.com/blog/trigkit4/1190000002440502)

## 34.ajax过程

(1)创建XMLHttpRequest对象,也就是创建一个异步调用对象.

(2)创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息.

(3)设置响应HTTP请求状态变化的函数.

(4)发送HTTP请求.

(5)获取异步调用返回的数据.

(6)使用JavaScript和DOM实现局部刷新.

详情：[JavaScript学习总结（七）Ajax和Http状态字](http://segmentfault.com/blog/trigkit4/1190000000691919)

## 35.异步加载和延迟加载

1.异步加载的方案： 动态插入script标签

2.通过ajax去获取js代码，然后通过eval执行

3.script标签上添加defer或者async属性

4.创建并插入iframe，让它异步执行js

5.延迟加载：有些 js 代码并不是页面初始化的时候就立刻需要的，而稍后的某些情况才需要的。

## 36.前端安全问题？

（XSS，sql注入，CSRF）

CSRF：是跨站请求伪造，很明显根据刚刚的解释，他的核心也就是请求伪造，通过伪造身份提交POST和GET请求来进行跨域的攻击。

\*\*完成CSRF需要两个步骤：\*\*

1.登陆受信任的网站A，在本地生成COOKIE

2.在不登出A的情况下，或者本地COOKIE没有过期的情况下，访问危险网站B。

## 37.ie各版本和chrome可以并行下载多少个资源

IE6 两个并发，iE7升级之后的6个并发，之后版本也是6个

Firefox，chrome也是6个

## 38.javascript里面的继承怎么实现，如何避免原型链上面的对象共享

用构造函数和原型链的混合模式去实现继承，避免对象共享可以参考经典的extend()函数，很多前端框架都有封装的，就是用一个空函数当做中间变量

## 39.grunt， YUI compressor 和 google clojure用来进行代码压缩的用法。

YUI Compressor 是一个用来压缩 JS 和 CSS 文件的工具，采用Java开发。

使用方法：

//压缩JS

java -jar yuicompressor-2.4.2.jar --type js --charset utf-8 -v src.js > packed.js

//压缩CSS

java -jar yuicompressor-2.4.2.jar --type css --charset utf-8 -v src.css > packed.css

详情请见：[你需要掌握的前端代码性能优化工具](http://segmentfault.com/blog/trigkit4/1190000002585760)

## 40.Flash、Ajax各自的优缺点，在使用中如何取舍？

1、Flash ajax对比

Flash适合处理多媒体、矢量图形、访问机器；对CSS、处理文本上不足，不容易被搜索。

Ajax对CSS、文本支持很好，支持搜索；多媒体、矢量图形、机器访问不足。

共同点：与服务器的无刷新传递消息、用户离线和在线状态、操作DOM

## 41.请解释一下 JavaScript 的同源策略。

概念:同源策略是客户端脚本（尤其是Javascript）的重要的安全度量标准。它最早出自Netscape Navigator2.0，其目的是防止某个文档或脚本从多个不同源装载。

这里的同源策略指的是：协议，域名，端口相同，同源策略是一种安全协议。  
指一段脚本只能读取来自同一来源的窗口和文档的属性。

### 为什么要有同源限制？

我们举例说明：比如一个黑客程序，他利用Iframe把真正的银行登录页面嵌到他的页面上，当你使用真实的用户名，密码登录时，他的页面就可以通过Javascript读取到你的表单中input中的内容，这样用户名，密码就轻松到手了。

## 42.什么是 “use strict”; ? 使用它的好处和坏处分别是什么？

ECMAscript 5添加了第二种运行模式：”严格模式”（strict mode）。顾名思义，这种模式使得Javascript在更严格的条件下运行。

设立”严格模式”的目的，主要有以下几个：

- 消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;

- 消除代码运行的一些不安全之处，保证代码运行的安全；

- 提高编译器效率，增加运行速度；

- 为未来新版本的Javascript做好铺垫。

注：经过测试IE6,7,8,9均不支持严格模式。

缺点：  
现在网站的JS 都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被 merge 后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节。

## 43.GET和POST的区别，何时使用POST？

GET：一般用于信息获取，使用URL传递参数，对所发送信息的数量也有限制，一般在2000个字符

POST：一般用于修改服务器上的资源，对所发送的信息没有限制。

GET方式需要使用Request.QueryString来取得变量的值，而POST方式通过Request.Form来获取变量的值，

也就是说Get是通过地址栏来传值，而Post是通过提交表单来传值。

然而，在以下情况中，请使用 POST 请求：

无法使用缓存文件（更新服务器上的文件或数据库）

向服务器发送大量数据（POST 没有数据量限制）

发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

## 44.哪些地方会出现css阻塞，哪些地方会出现js阻塞？

**js的阻塞特性：**所有浏览器在下载JS的时候，会阻止一切其他活动，比如其他资源的下载，内容的呈现等等。直到JS下载、解析、执行完毕后才开始继续并行下载其他资源并呈现内容。为了提高用户体验，新一代浏览器都支持并行下载JS，但是JS下载仍然会阻塞其它资源的下载（例如.图片，css文件等）。

由于浏览器为了防止出现JS修改DOM树，需要重新构建DOM树的情况，所以就会阻塞其他的下载和呈现。

嵌入JS会阻塞所有内容的呈现，而外部JS只会阻塞其后内容的显示，2种方式都会阻塞其后资源的下载。也就是说外部样式不会阻塞外部脚本的加载，但会阻塞外部脚本的执行。

CSS怎么会阻塞加载了？CSS本来是可以并行下载的，在什么情况下会出现阻塞加载了(在测试观察中，IE6下CSS都是阻塞加载）

当CSS后面跟着嵌入的JS的时候，该CSS就会出现阻塞后面资源下载的情况。而当把嵌入JS放到CSS前面，就不会出现阻塞的情况了。

根本原因：因为浏览器会维持html中css和js的顺序，样式表必须在嵌入的JS执行前先加载、解析完。而嵌入的JS会阻塞后面的资源加载，所以就会出现上面CSS阻塞下载的情况。

嵌入JS应该放在什么位置？

1、放在底部，虽然放在底部照样会阻塞所有呈现，但不会阻塞资源下载。

2、如果嵌入JS放在head中，请把嵌入JS放在CSS头部。

3、使用defer（只支持IE）

4、不要在嵌入的JS中调用运行时间较长的函数，如果一定要用，可以用`setTimeout`来调用

### Javascript无阻塞加载具体方式

* **将脚本放在底部。**<link>还是放在head中，用以保证在js加载前，能加载出正常显示的页面。<script>标签放在</body>前。
* **成组脚本**：由于每个<script>标签下载时阻塞页面解析过程，所以限制页面的<script>总数也可以改善性能。适用于内联脚本和外部脚本。
* **非阻塞脚本**：等页面完成加载后，再加载js代码。也就是，在window.onload事件发出后开始下载代码。  
  （1）defer属性：支持IE4和fierfox3.5更高版本浏览器  
  （2）动态脚本元素：文档对象模型（DOM）允许你使用js动态创建HTML的几乎全部文档内容。代码如下：

<script>

var script=document.createElement("script");

script.type="text/javascript";

script.src="file.js";

document.getElementsByTagName("head")[0].appendChild(script);

</script>

此技术的重点在于：无论在何处启动下载，文件额下载和运行都不会阻塞其他页面处理过程。即使在head里（除了用于下载文件的http链接）。

## 45.闭包相关问题？

详情请见：[详解js闭包](http://segmentfault.com/blog/trigkit4/1190000000652891)

## 46.js事件处理程序问题？

详情请见：[JavaScript学习总结（九）事件详解](http://segmentfault.com/blog/trigkit4/1190000002174034)

## 47.eval是做什么的？

它的功能是把对应的字符串解析成JS代码并运行；

应该避免使用eval，不安全，非常耗性能（2次，一次解析成js语句，一次执行）。

## 48.写一个通用的事件侦听器函数?

*// event(事件)工具集，来源：github.com/markyun*

markyun.Event = {

*// 页面加载完成后*

readyEvent : function(fn) {

if (fn==null) {

fn=document;

}

var oldonload = window.onload;

if (typeof window.onload != 'function') {

window.onload = fn;

} else {

window.onload = function() {

oldonload();

fn();

};

}

},

*// 视能力分别使用dom0||dom2||IE方式 来绑定事件*

*// 参数： 操作的元素,事件名称 ,事件处理程序*

addEvent : function(element, type, handler) {

if (element.addEventListener) {

*//事件类型、需要执行的函数、是否捕捉*

element.addEventListener(type, handler, false);

} else if (element.attachEvent) {

element.attachEvent('on' + type, function() {

handler.call(element);

});

} else {

element['on' + type] = handler;

}

},

*// 移除事件*

removeEvent : function(element, type, handler) {

if (element.removeEnentListener) {

element.removeEnentListener(type, handler, false);

} else if (element.datachEvent) {

element.detachEvent('on' + type, handler);

} else {

element['on' + type] = null;

}

},

*// 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)*

stopPropagation : function(ev) {

if (ev.stopPropagation) {

ev.stopPropagation();

} else {

ev.cancelBubble = true;

}

},

*// 取消事件的默认行为*

preventDefault : function(event) {

if (event.preventDefault) {

event.preventDefault();

} else {

event.returnValue = false;

}

},

*// 获取事件目标*

getTarget : function(event) {

return event.target || event.srcElement;

},

*// 获取event对象的引用，取到事件的所有信息，确保随时能使用event；*

getEvent : function(e) {

var ev = e || window.event;

if (!ev) {

var c = this.getEvent.caller;

while (c) {

ev = c.arguments[0];

if (ev && Event == ev.constructor) {

break;

}

c = c.caller;

}

}

return ev;

}

};

## 49.Node.js的适用场景？

高并发、聊天、实时消息推送

## 50.JavaScript原型，原型链 ? 有什么特点？

\* 原型对象也是普通的对象，是对象一个自带隐式的 **\_\_proto\_\_** 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为null的话，我们就称之为原型链。

\* 原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链。

## 51.页面重构怎么操作？

编写 CSS、让页面结构更合理化，提升用户体验，实现良好的页面效果和提升性能。

## 52.WEB应用从服务器主动推送Data到客户端有那些方式？

html5 websoket

WebSocket通过Flash

XHR长时间连接

XHR Multipart Streaming

不可见的Iframe

<script>标签的长时间连接(可跨域)

## 53.事件、IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。

2. 事件处理机制：IE是事件冒泡、firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件。；

3. ev.stopPropagation();注意旧ie的方法 ev.cancelBubble = true;

## 54.ajax 是什么?ajax 的交互模型?同步和异步的区别?如何解决跨域问题?

详情请见：[JavaScript学习总结（七）Ajax和Http状态字](http://segmentfault.com/blog/trigkit4/1190000000691919)

1. 通过异步模式，提升了用户体验

2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用

3. Ajax在客户端运行，承担了一部分本来由服务器承担的工作，减少了大用户量下的服务器负载。

2. Ajax的最大的特点是什么。

Ajax可以实现动态不刷新（局部刷新）

readyState属性 状态 有5个可取值： 0=未初始化 ，1=启动 2=发送，3=接收，4=完成

ajax的缺点

1、ajax不支持浏览器back按钮。

2、安全问题 AJAX暴露了与服务器交互的细节。

3、对搜索引擎的支持比较弱。

4、破坏了程序的异常机制。

5、不容易调试。

跨域： jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面

## 55.js对象的深度克隆

function clone(Obj) {

var buf;

if (Obj instanceof Array) {

buf = []; *//创建一个空的数组*

var i = Obj.length;

while (i--) {

buf[i] = clone(Obj[i]);

}

return buf;

}else if (Obj instanceof Object){

buf = {}; *//创建一个空对象*

for (var k in Obj) { *//为这个对象添加新的属性*

buf[k] = clone(Obj[k]);

}

return buf;

}else{

return Obj;

}

}

## 56.AMD和CMD 规范的区别？

详情请见：[详解JavaScript模块化开发](http://segmentfault.com/blog/trigkit4/1190000000733959)

## 57.网站重构的理解？

网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。也就是说是在不改变UI的情况下，对网站进行优化，在扩展的同时保持一致的UI。

对于传统的网站来说重构通常是：

表格(table)布局改为DIV+CSS

使网站前端兼容于现代浏览器(针对于不合规范的CSS、如对IE6有效的)

对于移动平台的优化

针对于SEO进行优化

深层次的网站重构应该考虑的方面

减少代码间的耦合

让代码保持弹性

严格按规范编写代码

设计可扩展的API

代替旧有的框架、语言(如VB)

增强用户体验

通常来说对于速度的优化也包含在重构中

压缩JS、CSS、image等前端资源(通常是由服务器来解决)

程序的性能优化(如数据读写)

采用CDN来加速资源加载

对于JS DOM的优化

HTTP服务器的文件缓存

## 58.如何获取UA？

<script>

function whatBrowser() {

document.Browser.Name.value=navigator.appName;

document.Browser.Version.value=navigator.appVersion;

document.Browser.Code.value=navigator.appCodeName;

document.Browser.Agent.value=navigator.userAgent;

}

</script>

## 59.js数组去重

以下是数组去重的三种方法：

Array.prototype.unique1 = function () {

var n = []; *//一个新的临时数组*

for (var i = 0; i < this.length; i++) *//遍历当前数组*

{

*//如果当前数组的第i已经保存进了临时数组，那么跳过，*

*//否则把当前项push到临时数组里面*

if (n.indexOf(this[i]) == -1) n.push(this[i]);

}

return n;

}

Array.prototype.unique2 = function()

{

var n = {},r=[]; *//n为hash表，r为临时数组*

for(var i = 0; i < this.length; i++) *//遍历当前数组*

{

if (!n[this[i]]) *//如果hash表中没有当前项*

{

n[this[i]] = true; *//存入hash表*

r.push(this[i]); *//把当前数组的当前项push到临时数组里面*

}

}

return r;

}

Array.prototype.unique3 = function()

{

var n = [this[0]]; *//结果数组*

for(var i = 1; i < this.length; i++) *//从第二项开始遍历*

{

*//如果当前数组的第i项在当前数组中第一次出现的位置不是i，*

*//那么表示第i项是重复的，忽略掉。否则存入结果数组*

if (this.indexOf(this[i]) == i) n.push(this[i]);

}

return n;

}

## 60.HTTP状态码

100 Continue 继续，一般在发送post请求时，已发送了http header之后服务端将返回此信息，表示确认，之后发送具体参数信息

200 OK 正常返回信息

201 Created 请求成功并且服务器创建了新的资源

202 Accepted 服务器已接受请求，但尚未处理

301 Moved Permanently 请求的网页已永久移动到新位置。

302 Found 临时性重定向。

303 See Other 临时性重定向，且总是使用 GET 请求新的 URI。

304 Not Modified 自从上次请求后，请求的网页未修改过。

400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。

401 Unauthorized 请求未授权。

403 Forbidden 禁止访问。

404 Not Found 找不到如何与 URI 相匹配的资源。

500 Internal Server Error 最常见的服务器端错误。

503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。

## 61.cache-control

网页的缓存是由HTTP消息头中的“Cache-control”来控制的，常见的取值有private、no-cache、max-age、must-revalidate等，默认为private。

Expires 头部字段提供一个日期和时间，响应在该日期和时间后被认为失效。允许客户端在这个时间之前不去检查（发请求），等同max-age的效果。但是如果同时存在，则被Cache-Control的max-age覆盖。

Expires = "Expires" ":" HTTP-date

例如

Expires: Thu, 01 Dec 1994 16:00:00 GMT （必须是GMT格式）

如果把它设置为-1，则表示立即过期

Expires和max-age都可以用来指定文档的过期时间，但是二者有一些细微差别

1.Expires在HTTP/1.0中已经定义，Cache-Control:max-age在HTTP/1.1中才有定义，为了向下兼容，仅使用max-age不够；

2.Expires指定一个绝对的过期时间(GMT格式),这么做会导致至少2个问题：1)客户端和服务器时间不同步导致Expires的配置出现问题。 2）很容易在配置后忘记具体的过期时间，导致过期来临出现浪涌现象；

3.max-age 指定的是从文档被访问后的存活时间，这个时间是个相对值(比如:3600s),相对的是文档第一次被请求时服务器记录的Request\_time(请求时间)

4.Expires指定的时间可以是相对文件的最后访问时间(Atime)或者修改时间(MTime),而max-age相对对的是文档的请求时间(Atime)

如果值为no-cache,那么每次都会访问服务器。如果值为max-age,则在过期之前不会重复访问服务器。

## 62.js操作获取和设置cookie

*//创建cookie*

function setCookie(name, value, expires, path, domain, secure) {

var cookieText = encodeURIComponent(name) + '=' + encodeURIComponent(value);

if (expires instanceof Date) {

cookieText += '; expires=' + expires;

}

if (path) {

cookieText += '; expires=' + expires;

}

if (domain) {

cookieText += '; domain=' + domain;

}

if (secure) {

cookieText += '; secure';

}

document.cookie = cookieText;

}

*//获取cookie*

function getCookie(name) {

var cookieName = encodeURIComponent(name) + '=';

var cookieStart = document.cookie.indexOf(cookieName);

var cookieValue = null;

if (cookieStart > -1) {

var cookieEnd = document.cookie.indexOf(';', cookieStart);

if (cookieEnd == -1) {

cookieEnd = document.cookie.length;

}

cookieValue = decodeURIComponent(document.cookie.substring(cookieStart + cookieName.length, cookieEnd));

}

return cookieValue;

}

*//删除cookie*

function unsetCookie(name) {

document.cookie = name + "= ; expires=" + new Date(0);

}

### 1、新的 HTML5 文档类型和字符集是？

HTML5 文档类型很简单：

|  |  |
| --- | --- |
| 1 | <!doctype html> |

HTML5 使用 UTF-8 编码示例：

|  |  |
| --- | --- |
| 1 | <meta charset=”UTF-8″> |

### 2、HTML5 中如何嵌入音频？

HTML5 支持 MP3、Wav 和 Ogg 格式的音频，下面是在网页中嵌入音频的简单示例：

|  |  |
| --- | --- |
| 1  2  3  4 | <audio controls>      <source src=”jamshed.mp3″ type=”audio/mpeg”>      Your browser does’nt support audio embedding feature.  </audio> |

### 3、HTML5 中如何嵌入视频？

和音频类似，[**HTML5**](http://www.cnblogs.com/lhb25/category/146076.html) 支持 MP4、WebM 和 Ogg 格式的视频，下面是简单示例：

|  |  |
| --- | --- |
| 1  2  3  4 | <video width=”450″ height=”340″ controls>    <source src=”jamshed.mp4″ type=”video/mp4″>     Your browser does’nt support video embedding feature.  </video> |

### 4、除了 audio 和 video，HTML5 还有哪些媒体标签？

[**HTML5**](http://www.yyyweb.com/tag/html5) 对于多媒体提供了强有力的支持，除了 audio 和 video 标签外，还支持以下标签：

<embed> 标签定义嵌入的内容，比如插件。

|  |  |
| --- | --- |
| 1 | <embed type=”video/quicktime” src=”Fishing.mov”> |

<source> 对于定义多个数据源很有用。

|  |  |
| --- | --- |
| 1  2  3  4 | <video width=”450″ height=”340″ controls>       <source src=”jamshed.mp4″ type=”video/mp4″>       <source src=”jamshed.ogg” type=”video/ogg”>  </video> |

<track> 标签为诸如 video 元素之类的媒介规定外部文本轨道。 用于规定字幕文件或其他包含文本的文件，当媒介播放时，这些文件是可见的。

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <video width=”450″ height=”340″ controls>       <source src=”jamshed.mp4″ type=”video/mp4″>       <source src=”jamshed.ogg” type=”video/ogg”>       <track kind=”subtitles” label=”English” src=”jamshed\_en.vtt” srclang=”en” default></track>        <track kind=”subtitles” label=”Arabic” src=”jamshed\_ar.vtt” srclang=”ar”></track>  </video> |

### 5、HTML5 Canvas 元素有什么用？

Canvas 元素用于在网页上绘制图形，该元素标签强大之处在于可以直接在 [**HTML**](http://www.cnblogs.com/lhb25/category/146076.html) 上进行图形操作，

|  |  |
| --- | --- |
| 1  2 | <canvas id=”canvas1″ width=”300″ height=”100″>  </canvas> |

* [不可思议的 HTML5 Canvas 应用试验](http://www.cnblogs.com/lhb25/p/html5-canvas-demos.html)
* [18个基于 HTML5 Canvas 的图表库](http://www.cnblogs.com/lhb25/archive/2011/04/12/1964343.html)
* [20个惊艳的 HTML5 Canvas 应用试验](http://www.cnblogs.com/lhb25/archive/2012/05/29/impressive-html5-canvas-experiments.html)
* [16款 HTML5 Canvas 开发的网页游戏](http://www.cnblogs.com/lhb25/archive/2011/03/27/1993819.html)
* [推荐14款强大的HTML5素描及绘图工具](http://www.cnblogs.com/lhb25/archive/2011/03/12/1964298.html)

### 6、HTML5 存储类型有什么区别？

HTML5 能够本地存储数据，在之前都是使用 cookies 使用的。[**HTML5**](http://www.yyyweb.com/tag/html5) 提供了下面两种本地存储方案：

* **localStorage**用于持久化的本地存储，数据永远不会过期，关闭浏览器也不会丢失。
* **sessionStorage**同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储

### 7、HTML5 有哪些新增的表单元素？

HTML5 新增了很多表单元素让开发者构建更优秀的 Web 应用程序。

* datalist
* datetime
* output
* keygen
* date
* month
* week
* time
* color
* number
* range
* email
* url

### 8、HTML5 废弃了哪些 HTML4 标签？

HTML5 废弃了一些过时的，不合理的 [**HTML**](http://www.yyyweb.com/tag/html5) 标签：

* frame
* frameset
* noframe
* applet
* big
* center
* basefront

### 9、HTML5 标准提供了哪些新的 API？

HTML5 提供的应用程序 API 主要有：

* Media API
* Text Track API
* Application Cache API
* User Interaction
* Data Transfer API
* Command API
* Constraint Validation API
* History API

### 10、HTML5 应用程序缓存和浏览器缓存有什么区别？

应用程序缓存是 [**HTML5**](http://www.yyyweb.com/tag/html5) 的重要特性之一，提供了离线使用的功能，让应用程序可以获取本地的网站内容，例如 HTML、CSS、图片以及 JavaScript。这个特性可以提高网站性能，它的实现借助于 manifest 文件，如下：

|  |  |
| --- | --- |
| 1  2  3  4 | <!doctype html>  <html manifest=”example.appcache”>  …..  </html> |

与传统浏览器缓存相比，它不强制用户访问的网站内容被缓存。

**内容**

我是一个ASP.NET MVC的开发者，最近在我找工作的时候被问到很多与HTML5相关的问题和新特性。所以以下40个重要的问题将帮助你复习HTML5相关的知识。

这些问题不是你得到工作的高效解决方案，但是可以在你想快速复习相关主题的时候有所帮助。

快乐地找工作。

SGML（标准通用标记语言）和HTML（超文本标记语言），XML（可扩展标记语言）和HTML的之间有什么关系？

SGML（标准通用标记语言）是一个标准，告诉我们怎么去指定文档标记。他是只描述文档标记应该是怎么样的元语言，HTML是被用SGML描述的标记语言。

因此利用SGML创建了HTML参照和必须共同遵守的DTD，你会经常在HTML页面的头部发现“DOCTYPE”属性，用来定义用于解析目标DTD

|  |  |
| --- | --- |
| 1 | <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"> |

现在解析SGML是一件痛苦的事情，所以创建了XML使事情更好。XML使用了SGML，例如：在SGML中你必须使用起始和结束标签，但是在XML你可以有自动关闭的结束标签。

XHTML创建于XML，他被使用在HTML4.0中。你可以参考下面代码片段中展示的XML DTD

|  |  |
| --- | --- |
| 1 | <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> |

总之，SGML是所有类型的父类，较旧的HTML利用SGML，HTML4.0使用派生自XML的XHTML

什么是HTML5？

HTML5是最新的HTML标准，他的主要目标是提供所有内容而不需要任何的像flash，silverlight等的额外插件，这些内容来自动画，视频，富GUI等

HTML5是万维网联盟（W3C）和网络超文本应用技术工作组（WHATWG）之间合作输出的

为什么HTML5里面我们不需要DTD（Document Type Definition文档类型定义）？

HTML5没有使用SGML或者XHTML，他是一个全新的东西，因此你不需要参考DTD，对于HTML5，你仅需放置下面的文档类型代码告诉浏览器识别这是HTML5文档

如果我不放入<! DOCTYPE html> ，HTML5还会工作么？

不会，浏览器将不能识别他是HTML文档，同时HTML5的标签将不能正常工作

哪些浏览器支持HTML5？

几乎所有的浏览器Safari，Chrome，Firefox，Opera，IE都支持HTML5

HTML5的页面结构同HTML4或者更前的HTML有什么区别？

一个典型的WEB页面包含头部，脚部，导航，中心区域，侧边栏。现在如果我们想在在HTML4的HTML区域中呈现这些内容，我们可能要使用DIV标签。

但是在HTML5中通过为这些区域创建元素名称使他们更加清晰，也使得你的HTML更加可读

以下是形成页面结构的HTML5元素的更多细节：

* <header>：代表HTML的头部数据
* <footer>：页面的脚部区域
* <nav>：页面导航元素
* <article>：自包含的内容
* <section>：使用内部article去定义区域或者把分组内容放到区域里
* <aside>：代表页面的侧边栏内容

HTML5中的datalist是什么？

HTML5中的Datalist元素有助于提供文本框自动完成特性，如下图所示：

以下是DataList功能的HTML代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <input list="Country">  <datalist id="Country">    <option value="India">    <option value="Italy">    <option value="Iran">    <option value="Israel">    <option value="Indonesia">  </datalist> |

HTML5中什么是不同的新的表单元素类型？

这里有10个重要的新的表单元素在HTML5中被介绍

1. Color
2. Date
3. Datetime-local
4. Email
5. Time
6. Url
7. Range
8. Telephone
9. Number
10. Search

让我们一步一步了解这10个元素

如果你想显示颜色选择对话框

|  |  |
| --- | --- |
| 1 | <input type="color" name="favcolor"> |

如果你想显示日历对话框

|  |  |
| --- | --- |
| 1 | <input type="date" name="bday"> |

如果你想显示含有本地时间的日历

|  |  |
| --- | --- |
| 1 | <input type="datetime-local" name="bdaytime"> |

如果你想创建一个含有email校验的HTML文本框，我们可以设置类型为“email”

|  |  |
| --- | --- |
| 1 | <input type="email" name="email"> |

对于URL验证设置类型为”url”，如下图显示的HTML代码

|  |  |
| --- | --- |
| 1 | <input type="url" name="sitename"> |

如果你想用文本展示数字范围，你可以设置类型为“number”

|  |  |
| --- | --- |
| 1 | <input type="number" name="quantity" min="1" max="5"> |

如果你想显示范围控制，你可以使用类型”range”

|  |  |
| --- | --- |
| 1 | <input type="range" min="0" max="10" step="2" value="6"> |

想让文本框作为搜索引擎

|  |  |
| --- | --- |
| 1 | <input type="search" name="googleengine"> |

想只能输入时间

|  |  |
| --- | --- |
| 1 | <input type="time" name="usr\_time"> |

如果你想使用文本框接受电话号码

|  |  |
| --- | --- |
| 1 | <input type="tel" name="mytel"> |

HTML5中什么是输出元素？

当你需要计算两个输入的和值到一个标签中的时候你需要输出元素。例如你有两个文本框（如下图），你想将来自这两个输入框中的数字求和并放到标签中。

下面是如何在HTML5中使用输出元素的代码

|  |  |
| --- | --- |
| 1  2  3  4  5 | <form onsubmit="return false"  öninput="o.value = parseInt(a.value) + parseInt(b.value)">    <input name="a" type="number"> +    <input name="b" type="number"> =    <output name="o" />  </form> |

为了简单起见，你也可以使用“valueAsNumber”来代替“parseInt”。你同样能在output元素中使用“for”使其更加可读

|  |  |
| --- | --- |
| 1 | <output name="o" for="a b"></output> |

什么是SVG（Scalable Vector Graphics可缩放矢量图形）？

SVG（Scalable Vector Graphics可缩放矢量图形）表示可缩放矢量图形。他是基于文本的图形语言，使用文本，线条，点等来进行图像绘制，这使得他轻便，显示更加迅速

我们能看到使用HTML5的SVG的简单例子么？

比方说，我们希望使用HTML5 SVG去显示以下简单的线条

下面是HTML5代码

|  |  |
| --- | --- |
| 1  2  3 | <svg id="svgelem" height="[object SVGAnimatedLength]" xmlns="http://www.w3.org/2000/svg">  <line style="stroke: rgb(255, 0, 0); stroke-width: 2px;" y2="[object SVGAnimatedLength]" x2="[object SVGAnimatedLength]" y1="[object SVGAnimatedLength]" x1="[object SVGAnimatedLength]">  </line> |

HTML5中canvas是什么？

Canvas是HTML中你可以绘制图形的区域

我们如何使用Canvas来画一条简单的线？

* 定义Canvas区域
* 获取访问canvas上下文区域
* 绘制图形

定义Canvas区域

定义Canvas区域你需要使用下面的HTML代码，这定义了你能进行绘图的区域

|  |  |
| --- | --- |
| 1 | <canvas id="mycanvas" width="600" height="500" style="border:1px solid #000000;"></canvas> |

获取画布区域的访问

在画布上进行绘图我们首先需要获取上下文区域的关联，下面是获取画布区域的代码。

|  |  |
| --- | --- |
| 1  2 | var c=document.getElementById("mycanvas");  var ctx=c.getContext("2d"); |

绘制图形

现在一旦你获取了访问上下文，我们就可以开始在上下文中绘制了。首先调用“move”方法并从一个点开始，使用线条方法绘制线条然后使用stroke方法结束。

|  |  |
| --- | --- |
| 1  2  3 | ctx.moveTo(10,10);  ctx.lineTo(200,100);  ctx.stroke(); |

以下是完整的代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <body  onload="DrawMe();">  <canvas id="mycanvas" width="600" height="500" style="border:1px solid #000000;"></canvas>  </body>  <script>  function DrawMe()  {  var c=document.getElementById("mycanvas");  var ctx=c.getContext("2d");  ctx.moveTo(10,10);  ctx.lineTo(200,100);  ctx.stroke();  } |

你可以得到以下输出

Canvas和SVG图形的区别是什么？

Note：-如果你看了之前的两个的问题，Canvas和SVG都可以在浏览器上绘制图形。因此在这个问题中，面试官想知道你在什么时候选用哪种方式。

|  |  |
| --- | --- |
| SVG | Canvas |
| 这个就好像绘制和记忆，换句话说任何使用SVG绘制的形状都能被记忆和操作，浏览器可以再次显示 | Canvas就像绘制和忘记，一旦绘制完成你不能访问像素和操作它 |
| SVG对于创建图形例如CAD软件是良好的，一旦东西绘制，用户就想去操作它 | Canvas在绘制和忘却的场景例如动画和游戏是良好的 |
| 因为为了之后的操作，需要记录坐标，所以比较缓慢 | 因为没有记住以后事情的意向，所以更快 |
| 我们可以用绘制对象的相关事件处理 | 我们不能使用绘制对象的相关事件处理，因为我们没有他们的参考 |
| 分辨率无关 | 分辨率相关 |

如何使用Canvas和HTML5中的SVG去画一个矩形？  
HTML5使用SVG绘制矩形的代码

|  |  |
| --- | --- |
| 1  2  3 | <svg xmlns="http://www.w3.org/2000/svg" version="1.1">  <rect style="fill: rgb(0, 0, 255); stroke-width: 1px; stroke: rgb(0, 0, 0);" height="[object SVGAnimatedLength]" width="[object SVGAnimatedLength]">  </rect> |

HTML5使用Canvas绘制矩形的代码

|  |  |
| --- | --- |
| 1  2  3  4 | var c=document.getElementById("mycanvas");  var ctx=c.getContext("2d");  ctx.rect(20,20,150,100);  ctx.stroke(); |

CSS（cascading style sheets级联样式表）中的选择器是什么？  
选择器在你想应用一个样式的时候，帮助你去选择元素。举例，下面是简单的被命名为”instro”的样式，他适用于HTML元素显示红色背景

|  |  |
| --- | --- |
| 1  2  3  4  5 | <style>  .intro{    }  </style> |

应用上面的”intro”样式给div，我们可以使用”class”选择器，如下图所示

|  |  |
| --- | --- |
| 1  2  3  4 | <div class="intro">  <p>My name is Shivprasad koirala.</p>  <p>I write interview questions.</p>  </div> |

如何使用ID值来应用一个CSS样式？  
假设，你有一个HTML段落标签，使用id是”mytext”，就和下面的片段中显示的那样

|  |  |
| --- | --- |
| 1 | <p id="mytext">This is HTML interview questions.</p> |

你可以使用”#”选择器和”id”的名字创建一种样式，并把CSS值应用到段落标签中，因此应用样式到”mytext”元素，我们可以使用”#mytext”，如下所示

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <style>  #mytext  {    }  </style> |

迅速修订一些重要的选择器

设置所有段落标签背景色为黄色

|  |  |
| --- | --- |
| 1  2  3  4 | div p  {    } |

设置所有div内部的段落标签为黄色背景

|  |  |
| --- | --- |
| 1  2  3  4 | div p  {    } |

设置所有div之后的段落标签为黄色背景

|  |  |
| --- | --- |
| 1  2  3  4 | div+p  {    } |

设置所有含有“target”属性的变为黄色背景

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | a[target]  {    }  <a href="http://www.questpond.com">ASP.NET interview questions</a>  <a href="http://www.questpond.com" target="\_blank">c# interview questions</a>  <a href="http://www.questpond.org" target="\_top">.NET interview questions with answers</a> |

当控制得到焦点的时候设置所有的元素为黄色背景

|  |  |
| --- | --- |
| 1  2  3  4 | input:focus  {    } |

根据相关连接操作设置超链接样式

|  |  |
| --- | --- |
| 1  2  3  4 | a:link    {color:green;}  a:visited {color:green;}  a:hover   {color:red;}  a:active  {color:yellow;} |

CSS中使用列布局是什么？  
CSS列布局帮助你分割文本变为列，例如考虑下面的杂志新闻在一个大的文本中，但是我们需要在他们之间使用边界划分为3列，这里HTML5的列布局就有所帮助了

为了实现列布局我们需要指定以下内容

* 我们需要把text划分为多少列

指定列数我们需要使用column-count，对于Chrome和firefox分别需要”webkit”和“moz-column”

|  |  |
| --- | --- |
| 1  2  3 | -moz-column-count:3; /\* Firefox \*/  -webkit-column-count:3; /\* Safari and Chrome \*/  column-count:3; |

* 两列之间我们想要多少差距

|  |  |
| --- | --- |
| 1  2  3 | -moz-column-gap:40px; /\* Firefox \*/  -webkit-column-gap:40px; /\* Safari and Chrome \*/  column-gap:20px; |

· 你想在这些列之间画一条线么？如果是，那么多厚呢？

|  |  |
| --- | --- |
| 1  2  3 | -moz-column-rule:4px outset #ff00ff; /\* Firefox \*/  -webkit-column-rule:4px outset #ff00ff; /\* Safari and Chrome \*/  column-rule:6px outset #ff00ff; |

以下是完整代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <style>  .magazine  {  -moz-column-count:3; /\* Firefox \*/  -webkit-column-count:3; /\* Safari and Chrome \*/  column-count:3;    -moz-column-gap:40px; /\* Firefox \*/  -webkit-column-gap:40px; /\* Safari and Chrome \*/  column-gap:20px;    -moz-column-rule:4px outset #ff00ff; /\* Firefox \*/  -webkit-column-rule:4px outset #ff00ff; /\* Safari and Chrome \*/  column-rule:6px outset #ff00ff;  }  </style> |

你可以使用class属性来应用样式到文本

|  |  |
| --- | --- |
| 1  2  3 | <div class="magazine">  Your text goes here which you want to divide in to 3 columns.  </div> |

你能解释一下CSS的盒子模型么？  
CSS和模型是围绕在HTML元素周围的定义Border(边界)，padding(内边距)和margin(外边距)的矩形空间  
Border(边界)：定义了元素包含的最大区域，我们能够使边界可见，不可见，定义高度和宽度等；  
Padding(内边距)：定义了边界和内部元素的间距  
Margin：定义了边界和任何相邻元素的间距

例如以下是简单的CSS代码定义了盒子的边界，内边距和外边距值

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | .box {      width: 200px;      border: 10px solid #99c;      padding: 20px;      margin: 50px;  } |

现在如果我们应用了以上的CSS到一个如下显示的DIV标签，你输出将会和下面图形中显示的那样。我已经创建两个测试“Some text”和“Some other text”，因此我们能看到多少margin(外边距)的属性功能

|  |  |
| --- | --- |
| 1  2  3  4 | <div align="middle" class="box">  Some text  </div>  Some other text |

你能解释一些CSS3中的文本效果么？

这里面试官期待你回答两个Css的文本效果，以下是两种需要注意的效果

阴影文本效果

|  |  |
| --- | --- |
| 1  2  3  4 | .specialtext  {  text-shadow: 5px 5px 5px #FF0000;  } |

文字包装效果

|  |  |
| --- | --- |
| 1  2  3  4 | <style>  .breakword  {word-wrap:break-word;}  </style> |

什么是Web Workers？为什么我们需要他们？  
考虑以下会执行上百万次的繁重的循环代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | function  SomeHeavyFunction()  {  for (i = 0; i < 10000000000000; i++)  {  x = i + x;  }  } |

比方说上面的循环代码在HTML按钮点击以后执行，现在这个方法执行是同步的，换句话说这个浏览器必须等到循环完成才能操作

|  |  |
| --- | --- |
| 1 | <input type="button" onclick="SomeHeavyFunction();" /> |

这个会进一步导致浏览器冻结并且没有相应，屏幕还会显示如下的异常信息

如果你能移动这些繁重的循环到Javascript文件中，采用异步的方式运行，这意味着浏览器不需要等到循环接触，我们可以有更敏感的浏览器，这就是web worker的作用

Web worker帮助我们用异步执行Javascript文件

Web Worker线程的限制是什么？  
Web worker线程不能修改HTML元素，全局变量和Window.Location一类的窗口属性。你可以自由使用Javascript数据类型，XMLHttpRequest调用等。

我们如何在JavaScript中创建一个worker线程？  
创建一个worker线程，我们需要通过Javascript文件名创建worker对象

|  |  |
| --- | --- |
| 1 | var worker = new Worker("MyHeavyProcess.js"); |

我们需要使用“PostMessage”发送信息给worker对象，下面是相同的代码。

|  |  |
| --- | --- |
| 1 | worker.postMessage(); |

当worker线程发送数据的时候，我们在调用结束的时候，通过”onMessage”事件获取

|  |  |
| --- | --- |
| 1  2  3  4 | worker.onmessage = function (e)  {  document.getElementById("txt1").value = e.data;  }; |

这个繁重的循环在“MyHeavyProcess.js”的Javascript文件中，以下代码，当Javascript文件想发送信息，他使用”postmessage”，同时任何来自发送者的信息都在“onmessage”事件中接收到。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var x =0  self.onmessage = function (e) {      for (i = 0; i < 1000000000; i++)      {          x = i + x;      }      self.postMessage(x);  }; |

如何中止Web Worker?

|  |  |
| --- | --- |
| 1 | w.terminate(); |

为什么我们需要HTML5的服务发送事件？  
网络世界的普遍需求是从服务器更新。以一个股票应用为例，浏览器必须定期从服务器更新最新的股票值。

现在实现这类需求开发者通常写一些PULL的代码，到服务器同时抓取某些区间数据。现在PULL的解决方案是很好的，但是这使得网络健谈有很多的调用，同时增加了服务器的负担。

因此相比于PULL，如果我们能采用某种PUSH的解决方案那会是很棒的。简而言之，当服务器更新的时候，将会发送更新到浏览器客户端，那可以被接受通过使用”SERVER SENT EVENT”

因此首要的是浏览器需要连接将会发送更新的服务器资源，比方说我们有一个”stock.aspx”页面会发送股票更新，因此连接该页面，我们需要使用附加时间来源对象，如下所示：

|  |  |
| --- | --- |
| 1 | var source = new EventSource("stock.aspx"); |

当我们将要接受服务器发送的更新信息时，我们需要附加功能。我们需要附加功能到”onmessage”事件就像以下显示的那样。

|  |  |
| --- | --- |
| 1  2  3 | source.onmessage = function (event) {    document.getElementById("result").innerHTML += event.data + "";  }; |

现在来自服务端，我们需要去发送事件，下面是一些用命令需要从服务端发送的重要事件列表

|  |  |
| --- | --- |
| Event | Command |
| 发送数据到客户端 | data : hello |
| 告诉客户端10s内重试 | retry : 10000 |
| 提出具体事件与数据 | event : successdata : You are logged in. |

因此，举例说明，如果你想下面的ASP.NET代码一样发送数据，请标记内容类型设置给文本/事件

|  |  |
| --- | --- |
| 1  2  3  4 | Response.ContentType="text/event-stream";  Response.Expires=-1;  Response.Write("data: " + DateTime.Now.ToString());  Response.Flush(); |

以下是设置10s后重试的命令

|  |  |
| --- | --- |
| 1 | Response.Write("retry: 10000"); |

如果你想附加事件，我们需要使用“addEventListener”事件，如下代码所示：

|  |  |
| --- | --- |
| 1  2  3 | source.addEventListener('message', function(e) {    console.log(e.data);  }, false); |

来自服务器端的以下信息将会触发Javascript的”message”方法

|  |  |
| --- | --- |
| 1  2 | event: message  data : hello |

HTML5中的本地存储概念是什么？  
很多时候我们会存储用户本地信息到电脑上，例如：比方说用户有一个填充了一半的长表格，然后突然网络连接断开了，这样用户希望你能存储这些信息到本地，当网络恢复的时候，他想获取这些信息然后发送到服务器进行存储  
现代浏览器拥有的存储被叫做“Local Storage”,你可以存储这些信息。

我们如何从本地存储中添加和移除数据？  
数据添加到本地存储采用键值对，以下示例显示了城市数据”India”添加了键”Key001”

|  |  |
| --- | --- |
| 1 | localStorage.setItem(“Key001”,”India”); |

从本地存储中检索数据我们可以提供键名并使用”getItem”方法

|  |  |
| --- | --- |
| 1 | var country = localStorage.getItem(“Key001”); |

你也可以使用以下代码，存储Javascript对象在本地存储中

|  |  |
| --- | --- |
| 1  2  3  4  5 | var country = {};  country.name = “India”;  country.code = “I001”;  localStorage.setItem(“I001”, country);  var country1 = localStorage.getItem(“I001”); |

如果你想存储Json格式，你可以使用“JSON.stringify”方法，如下所示：

|  |  |
| --- | --- |
| 1 | localStorage.setItem(“I001”,JSON.stringify(country)); |

本地存储的生命周期是什么？  
本地存储没有生命周期，它将保留知道用户从浏览器清除或者使用Javascript代码移除。

本地存储和cookies（储存在用户本地终端上的数据）之间的区别是什么？

|  |  |  |
| --- | --- | --- |
|  | Cookies | Local storage |
| 客户端/服务端 | 客户端和服务端都能访问数据。Cookie的数据通过每一个请求发送到服务端 | 只有本地浏览器端可访问数据，服务器不能访问本地存储直到故意通过POST或者GET的通道发送到服务器 |
| 大小 | 每个cookie有4095byte | 每个域5MB |
| 过期 | Cookies有有效期，所以在过期之后cookie和cookie数据会被删除 | 没有过期数据，无论最后用户从浏览器删除或者使用Javascript程序删除，我们都需要删除 |

什么是事务存储？我们如何创建一个事务存储？  
会话存储和本地存储类似，但是数据在会话中有效，简而言之数据在你关闭浏览器的时候就被删除了。  
为了创建一个会话存储你需要使用“sessionStorage.variablename.”在以下的代码我们创建了一个名为”clickcount”的变量；  
如果你刷新浏览器则数目增加，但是如果你关闭浏览器，“clickcount”变量又会从0开始。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | if(sessionStorage.clickcount)  {  sessionStorage.clickcount=Number(sessionStorage.clickcount)+1;  }  else  {  sessionStorage.clickcount = 0;  } |

本地存储和事务存储之间的区别是什么？  
本地存储数据持续永久，但是会话在浏览器打开时有效知道浏览器关闭时会话变量重置

什么是WebSQL？  
WebSQL是一个在浏览器客户端的结构关系数据库，这是浏览器内的本地RDBMS(关系型数据库系统)，你可以使用SQL查询

WebSql是HTML5的一个规范吗？  
不是，许多人把它标记为HTML5，但是他不是HTML5的规范的一部分，这个规范是基于SQLite的

我们如何使用WebSQL？  
第一步我们需要做的是使用如下所示的“OpenDatabase”方法打开数据库，第一个参数是数据库的名字，接下来是版本，然后是简单原文标题，最后是数据库大小；

|  |  |
| --- | --- |
| 1 | var db=openDatabase('dbCustomer','1.0','Customer app’, 2 \* 1024 \* 1024); |

为了执行SQL，我们需要使用“transaction”方法，并调用”executeSql”方法来使用SQL

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | db.transaction(function (tx)  {  tx.executeSql('CREATE TABLE IF NOT EXISTS tblCust(id unique, customername)');  tx.executeSql('INSERT INTO tblcust (id, customername) VALUES(1, "shiv")');  tx.executeSql('INSERT INTO tblcust (id, customername) VALUES (2, "raju")');  } |

万一你要使用“select”查询你会得到数据”result”集合，我们可以通过循环展示到HTML的用户界面

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | db.transaction(function (tx)  {    tx.executeSql('SELECT \* FROM tblcust', [], function (tx, results) {     for (i = 0; i < len; i++)  {       msg = "<p><b>" + results.rows.item(i).log + "</b></p>";       document.querySelector('#customer).innerHTML +=  msg;  }   }, null);  }); |

HTML5中的应用缓存是什么？  
一个最需要的事最终是用户的离线浏览，换句话说，如果网络连接不可用时，页面应该来自浏览器缓存，离线应用缓存可以帮助你达到这个目的  
应用缓存可以帮助你指定哪些文件需要缓存，哪些不需要。

HTML5中我们如何实现应用缓存？  
首先我们需要指定”manifest”文件，“manifest”文件帮助你定义你的缓存如何工作。以下是”mainfest”文件的结构

|  |  |
| --- | --- |
| 1  2  3  4 | CACHE MANIFEST  # version 1.0  CACHE :  Login.aspx |

* 所有manifest文件都以“CACHE MANIFEST”语句开始.
* #（散列标签）有助于提供缓存文件的版本.
* CACHE 命令指出哪些文件需要被缓存.
* Mainfest文件的内容类型应是“text/cache-manifest”.

以下是如何在ASP.NET C#使用manifest缓存

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | Response.ContentType = "text/cache-manifest";  Response.Write("CACHE MANIFEST \n");  Response.Write("# 2012-02-21 v1.0.0 \n");  Response.Write("CACHE : \n");  Response.Write("Login.aspx \n");  Response.Flush();  Response.End(); |

创建一个缓存manifest文件以后，接下来的事情实在HTML页面中提供mainfest连接，如下所示：

|  |  |
| --- | --- |
| 1 | <html manifest="cache.aspx"> |

当以上文件第一次运行，他会添加到浏览器应用缓存中，在服务器宕机时，页面从应用缓存中获取

我们如何刷新浏览器的应用缓存？  
应用缓存通过变更“#”标签后的版本版本号而被移除，如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CACHE MANIFEST  # version 2.0(new)  CACHE :  Login.aspx  Aboutus.aspx  NETWORK :  Pages.aspx |

应用缓存中的回退是什么？  
应用缓存中的回退帮助你指定在服务器不可访问的时候，将会显示某文件。例如在下面的manifest文件中，我们说如果谁敲击了”/home”同时服务器不可到达的时候，”homeoffline.html”文件应送达

|  |  |
| --- | --- |
| 1  2 | FALLBACK:  /home/ /homeoffline.html |

应用缓存中的网络是什么？  
网络命令描述不需要缓存的文件，例如以下代码中，我们说”home.aspx”永远都不应该被缓存或者离线访问。

|  |  |
| --- | --- |
| 1  2 | NETWORK:  home.aspx |

为一名前端开发人员，HTML，HTML5以及网站优化都是必须掌握的技术，下面列举一下HTML， HTML5, 网站优化等常见的面试题：

**HTML常见面试题：**

1. 什么是Semantic HTML（语义HTML）?

Semantic HTML是一种编码风格, 它通过添加能够被计算器所理解的语义(Meta data)，从而使HTML成为一个通用的信息交换媒介。在语义HTML中，<b></b>,<i></i>这类其中的内容不具有实际意义的标签是不应该被使用的，因为他们只是为了进行格式化，没有提供要表达的意义及页面结构。

2. DOCTYPE 有什么作用？

DOCTYPE主要作用是告诉浏览器这个网页是哪种HTML， 浏览器根据这个标示进行页面渲染。如果DOCTYPE声明不当或没有声明，浏览器将会用quirks mode（怪异模式）对页面进行渲染。

3. 什么是quirks mode（怪异模式）？

怪异模式（英语：quirks mode）是指在计算机领域中，一些网页浏览器为了维持对较旧的网页设计的向后兼容性，而使用的一种技术，有别于严格遵循万维网联盟（W3C）与互联网工程任务组（IETF）标准而设计的“标准模式”。

4. 标准模式和怪异模式之间的区别是什么？

两者之间突出的不同是对 CSS IE盒模型缺陷的处理。在IE6之前，Internet Explorer 曾经使用一种决定一个元素的盒模型的宽度和高度的，与 CSS 规范所指定相冲突的算法，而且由于 Internet Explorer 的流行，很多依赖于这种不正确的算法的网页被创建。而在IE 6, Internet Explorer 在标准模式下渲染时使用了 CSS 规范的算法，而在 quirks 模式下使用先前的，不规范的算法。  
另一个值得一提的不同点是某些行内 (inline) 元素的垂直对齐；很多早期的浏览器对齐图片至包含它们的盒子的下边框，虽然 CSS 的规范要求它们被对齐至盒内文本的基线。标准模式下，基于 Gecko 的浏览器将会对齐至基线，而在 quirks 模式下它们会对齐至底部。

**HTML5常见面试题：**

1. html5有哪些新的页面元素？

主要有<article>, <aside>, <bdi>, <command>, <details>, <figure>, <figcaption>, <summary>, <header>, <footer>, <hgroup>, <mark>, <meter>, <nav>, <progress>, <ruby>, <rt>, <section>, <time>, 和<wpr>

2. HTML5去除了哪些页面元素？

<frame> 和 <frameset>，另外一些不在支持的元素包括<noframe>, <applet>, <bigcenter> 和<basefront>.

3. HTML5有哪些新的表单元素？

datalist, datetime, output, keygen, date, month, week, time, number, range, email, 和 url.

4. HTML5增加了哪些标准的API?

Media API, Text Track API, Application Cache API, User Interaction, Data Transfer API, Command API, Constraint Validation API, 和  History API.

5.  HTML5 和老的 HTML之间的缓存机制有什么不同？

HTML5 有了一个新的Application Cache, 它为web应用创建了一个离线的缓存，包括HTML文件、CSS、图片和JS等，能提高访问性能。

**1 . HTML5 之前的 HTML 版本是？**

窗体顶端

* HTML 4.01
* HTML 4
* HTML 4.1
* HTML 4.9

窗体底端

**2 . HTML5 的正确 doctype 是？**

窗体顶端

* <!DOCTYPE html>
* <!DOCTYPE HTML5>
* <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 5.0//EN" "http://www.w3.org/TR/html5/strict.dtd">

窗体底端

**3 . 在 HTML5 中，哪个元素用于组合标题元素？**

窗体顶端

* <group>
* <header>
* <headings>
* <hgroup>

窗体底端

**4 . HTML5 中不再支持下面哪个元素？**

窗体顶端

* <q>
* <ins>
* <menu>
* <font>

窗体底端

**5 . HTML5 中不再支持下面哪个元素？**

窗体顶端

* <cite>
* <acronym>
* <abbr>
* <base>

窗体底端

**6 . 在 HTML5 中，onblur 和 onfocus 是：**

窗体顶端

* HTML 元素
* 样式属性
* 事件属性

窗体底端

**7 . 用于播放 HTML5 视频文件的正确 HTML5 元素是：**

窗体顶端

* <movie>
* <media>
* <video>

窗体底端

**8 . 用于播放 HTML5 音频文件的正确 HTML5 元素是：**

窗体顶端

* <mp3>
* <audio>
* <sound>

窗体底端

**9 . 在 HTML5 中不再支持 <script> 元素的哪个属性？**

窗体顶端

* rel
* href
* type
* src

窗体底端

**10 . 在 HTML5 中，哪个方法用于获得用户的当前位置？**

窗体顶端

* getPosition()
* getCurrentPosition()
* getUserPosition()

窗体底端

**11 . 新的 HTML5 全局属性，"contenteditable" 用于：**

窗体顶端

* 规定元素的上下文菜单。该菜单会在用户点击右键点击元素时出现。
* 规定元素内容是否是可编辑的。
* 从服务器升级内容。
* 返回内容在字符串中首次出现的位置。

窗体底端

**12 . 在 HTML5 中，contextmenu 和 spellcheck 是：**

窗体顶端

* HTML 属性
* HTML 元素
* 事件属性
* 样式属性

窗体底端

**13 . 在 HTML5 中，您能够直接将 SVG 元素嵌入 HTML 页面中。**

窗体顶端

* 错误
* 正确

窗体底端

**14 . 由 SVG 定义的图形是什么格式的？**

窗体顶端

* CSS
* HTML
* XML

窗体底端

**15 . HTML5 中的 <canvas> 元素用于：**

窗体顶端

* 显示数据库记录
* 操作 MySQL 中的数据
* 绘制图形
* 创建可拖动的元素

窗体底端

**16 . 哪个 HTML5 内建对象用于在画布上绘制？**

窗体顶端

* getContent
* getContext
* getGraphics
* getCanvas

窗体底端

**17 . 在 HTML5 中，哪个属性用于规定输入字段是必填的？**

窗体顶端

* required
* formvalidate
* validate
* placeholder

窗体底端

**18 . 哪种输入类型定义滑块控件？**

窗体顶端

* search
* controls
* slider
* range

窗体底端

**19 . 哪种输入类型用于定义周和年控件（无时区）？**

窗体顶端

* date
* week
* year

窗体底端

**20 . 哪个 HTML5 元素用于显示已知范围内的标量测量？**

窗体顶端

* <gauge>
* <range>
* <measure>
* <meter>

窗体底端

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

**1.HTML5 之前的 HTML 版本是？**

正确答案：HTML 4.01

**2.HTML5 的正确 doctype 是？**

正确答案：<!DOCTYPE html>

**3.在 HTML5 中，哪个元素用于组合标题元素？**

正确答案：<hgroup>

**4.HTML5 中不再支持下面哪个元素？**

正确答案：<font>

**5.HTML5 中不再支持下面哪个元素？**

正确答案：<acronym>

**6.在 HTML5 中，onblur 和 onfocus 是：**

正确答案：事件属性

**7.用于播放 HTML5 视频文件的正确 HTML5 元素是：**

正确答案：<video>

**8.用于播放 HTML5 音频文件的正确 HTML5 元素是：**

您的回答：<audio>

**9.在 HTML5 中不再支持 <script> 元素的哪个属性？**

正确答案：type

**10.在 HTML5 中，哪个方法用于获得用户的当前位置？**

正确答案：getCurrentPosition()

**11.新的 HTML5 全局属性，"contenteditable" 用于：**

正确答案：规定元素内容是否是可编辑的。

**12.在 HTML5 中，contextmenu 和 spellcheck 是：**

正确答案：HTML 属性

**13.在 HTML5 中，您能够直接将 SVG 元素嵌入 HTML 页面中。**

正确答案：正确

**14.由 SVG 定义的图形是什么格式的？**

正确答案：XML

**15.HTML5 中的 <canvas> 元素用于：**

正确答案：绘制图形

**16.哪个 HTML5 内建对象用于在画布上绘制？**

正确答案：getContext

**17.在 HTML5 中，哪个属性用于规定输入字段是必填的？**

正确答案：required

**18.哪种输入类型定义滑块控件？**

正确答案：range

**19.哪种输入类型用于定义周和年控件（无时区）？**

正确答案：week

**20.哪个 HTML5 元素用于显示已知范围内的标量测量？**

正确答案：<meter>

原文出自：[www.hangge.com](http://www.hangge.com/)  转载请保留原文链接：<http://www.hangge.com/blog/cache/detail_865.html>

# 1、考察this

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | var length = 10;  function fn() {    console.log(this.length);  }    var obj = {    length: 5,    method: function(fn) {      fn();      arguments[0]();    }  };    obj.method(fn, 1); |

输出：10 2

第一次输出10应该没有问题。我们知道取对象属于除了点操作符还可以用中括号，所以第二次执行时相当于arguments调用方法，this指向arguments，而这里传了两个参数，故输出arguments长度为2。

# 2、var和函数的提前声明

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | function fn(a) {    console.log(a);    var a = 2;    function a() {}    console.log(a);  }    fn(1); |

输出：function  a() {} 2

我们知道var和function是会提前声明的，而且function是优先于var声明的（如果同时存在的话），所以提前声明后输出的a是个function，然后代码往下执行a进行重新赋值了，故第二次输出是2。

# 3、局部变量和全局变量

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | var f = true;  if (f === true) {    var a = 10;  }    function fn() {    var b = 20;    c = 30;  }    fn();  console.log(a);  console.log(b);  console.log(c); |

输出：10 报错 30

这是个我犯了很久的错误，很长一段时间我都以为{…}内的新声明的变量是局部变量，后来我才发现function内的新声明的变量才是局部变量，而没有用var声明的变量在哪里都是全局变量。再次提醒切记只有function(){}内新声明的才能是局部变量，while{…}、if{…}、for(..) 之内的都是全局变量（除非本身包含在function内）。

# 4、变量隐式声明

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5 | if('a' in window) {    var a = 10;  }    alert(a); |

答案：10

前面我说过function和var会提前声明，而其实{…}内的变量也会提前声明。于是代码还没执行前，a变量已经被声明，于是 ‘a’ in window 返回true，a被赋值。

# 5、给基本类型数据添加属性，不报错，但取值时是undefined

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var a = 10;  a.pro = 10;  console.log(a.pro + a);    var s = 'hello';  s.pro = 'world';  console.log(s.pro + s); |

答案：NaN undefinedhello

给基本类型数据加属性不报错，但是引用的话返回undefined，10+undefined返回NaN，而undefined和string相加时转变成了字符串。

# 6、函数声明优于变量声明

JavaScript



|  |  |
| --- | --- |
| 1  2  3 | console.log(typeof fn);  function fn() {};  var fn; |

答案：function

因为函数声明优于变量声明。我们知道在代码逐行执行前，函数声明和变量声明会提前进行，而函数声明又会优于变量声明，这里的优于可以理解为晚于变量声明后，如果函数名和变量名相同，函数声明就能覆盖变量声明。所以以上代码将函数声明和变量声明调换顺序还是一样结果。

# 7、判断一个字符串中出现次数最多的字符，并统计次数

* hash table方式：

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | var s = 'aaabbbcccaaabbbaaa';  var obj = {};  var maxn = -1;  var letter;  for(var i = 0; i < s.length; i++) {    if(obj[s[i]]) {      obj[s[i]]++;      if(obj[s[i]] > maxn) {        maxn = obj[s[i]];        letter = s[i];      }    } else {      obj[s[i]] = 1;      if(obj[s[i]] > maxn) {        maxn = obj[s[i]];        letter = s[i];      }    }  }    alert(letter + ': ' + maxn); |

* 正则方式：

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | var s = 'aaabbbcccaaabbbaaabbbbbbbbbb';  var a = s.split('');  a.sort();  s = a.join('');  var pattern = /(\w)\1\*/g;  var ans = s.match(pattern);  ans.sort(function(a, b) {    return a.length < b.length;  });;  console.log(ans[0][0] + ': ' + ans[0].length); |

# 8、经典闭包

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <!-- 实现一段脚本，使得点击对应链接alert出相应的编号 -->  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <body>    <a href='#'> 第一个链接 </a> </br>    <a href='#'> 第二个链接 </a> </br>    <a href='#'> 第三个链接 </a> </br>    <a href='#'> 第四个链接 </a> </br>  </body> |

* dom污染法：

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <!-- 实现一段脚本，使得点击对应链接alert出相应的编号 -->  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <body>    <a href='#'> 第一个链接 </a> </br>    <a href='#'> 第二个链接 </a> </br>    <a href='#'> 第三个链接 </a> </br>    <a href='#'> 第四个链接 </a> </br>    <script type="text/javascript">      var lis = document.links;      for(var i = 0, length = lis.length; i < length; i++) {        lis[i].index = i;        lis[i].onclick = function() {          alert(this.index);        };      }    </script>  </body> |

* 闭包：

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <!-- 实现一段脚本，使得点击对应链接alert出相应的编号 -->  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <body>    <a href='#'> 第一个链接 </a> </br>    <a href='#'> 第二个链接 </a> </br>    <a href='#'> 第三个链接 </a> </br>    <a href='#'> 第四个链接 </a> </br>    <script type="text/javascript">      var lis = document.links;      for(var i = 0, length = lis.length; i < length; i++) {        (function(i) {          lis[i].onclick = function() {            alert(i + 1);          };        })(i);      }    </script>  </body> |

# 9、this

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | function JSClass() {    this.m\_Text = 'division element';    this.m\_Element = document.createElement('div');    this.m\_Element.innerHTML = this.m\_Text;    this.m\_Element.addEventListener('click', this.func);    // this.m\_Element.onclick = this.func;  }    JSClass.prototype.Render = function() {    document.body.appendChild(this.m\_Element);  }    JSClass.prototype.func = function() {    alert(this.m\_Text);  };    var jc = new JSClass();  jc.Render();  // add div  jc.func();  // 输出 division element  //click添加的div元素division element会输出underfined，为什么？ |

答案：division element undefined

第一次输出很好理解，第二次的话仔细看，this其实已经指向了this.m\_Element，因为是this.m\_Element调用的addEventListener函数，所以内部的this全指向它了。可以试着加上一行代码this.m\_Element.m\_Text = ‘hello world’，就会alert出hello world了。

# 10、split

请编写一个JavaScript函数 parseQueryString，它的用途是把URL参数解析为一个对象，如： var url = “http://witmax.cn/index.php?key0=0&key1=1&key2=2″

JavaScript



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | function parseQueryString(url) {    var obj = {};    var a = url.split('?');    if(a === 1) return obj;    var b = a[1].split('&');    for(var i = 0, length = b.length; i < length; i++) {      var c = b[i].split('=');      obj[c[0]] = c[1];    }    return obj;  }  var url = 'http://witmax.cn/index.php?key0=0&key1=1&key2=2';  var obj = parseQueryString(url);  console.log(obj.key0, obj.key1, obj.key2);  // 0 1 2 |