



前端面试资料

Web前端笔试题（一）

第一部分：结构与样式

1、列举10个html块元素标签和5个html行内标签，并简述块元素与行内元素的区别

块: div dl form h1 h2 h3 h4 h5 h6 hr ol li p pre table td tr

行内: a b s i u span br img input textarea sub sup

两者区别:

以图例来表述行内元素和块级元素的区别会更加直观:

注: 行内元素会再一条直线上, 是在同一行的。我是行内元素SPAN标签 我是行内元素strong标签

注: 块级元素各占一行。是垂直方向的!

我是块级元素div标签

我是块级元素p标签

如果你要将行内元素变成块级元素, 那么就只需要在该标签上加上样式 `display:block`; 这里已经牵涉到了CSS内容。块级元素可以用样式控制其高、宽的值。

行内元素不可以控制宽和高, 除非你想将它转变成为块级元素。它的宽和高, 是随标签里的内容而变化。

参考: <http://www.w3cwhy.com/css-html/html-hkmarka.html>

2、如何在IE浏览器下使元素拥有布局

参考: <http://www.nowamagic.net/librarys/veda/detail/1395>

http://www.cnblogs.com/ShepherdIsland/archive/2010/06/29/IE_Haslayout.html

3、写出使DIV#box（宽度为200px，高度为100px）水平垂直居中的样式代码

```
div#box {  
    position: absolute;  
    left: 50%;  
    top: 50%;  
    margin-left: -100px;  
    margin-top: -100px;  
}
```

4、描述css定位（什么是定位？定位分类？区别？其他有关定位知识）

参考: http://www.w3school.com.cn/css/css_positioning.asp

5、请简化下面的css代码

```

a. margin:0px;

b. padding:10px 0 10px 0;

c. border-width:1px; border-style:solid; border-color:#ff5500;

margin:0;

padding: 10px 0;

border: 1px solid #f50;

```

6、应用CSS方法实现隐藏下面DIV（至少三种方法）

```

div{

    display:none;

    visibility: hidden;

    position: absolute/relative

    left: -10000px;

}

```

第二部分：JavaScript与jQuery

7、使用Js在页面上循环输出1-100的奇数

```

function getOddNumber(){

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

        if(i%2==1){

            document.write(i);

        }

    }

}

```

8、编写js函数，用于计算 $a + (a-1) + (a-2) + \dots + 1$ 的和

方法一：

```

function sum1(a){

    var sum = a;

    for(var i=a; i >= 1; i--){

        sum += (i-1);

    }

    return sum;

}

```

```
alert(sum1(3));
```

方法二:

```
function sum2(a){  
    if(a==1){  
        return 1;  
    }else{  
        return a + sum2(a-1);  
    }  
}  
  
alert( sum2(3) );
```

方法三:

```
function sum3( a1, an, d ){  
    var n = (an - a1)/d + 1;  
    return n*a1 + n*(n-1)*d/2;  
}  
  
alert(sum3(1,5,1));
```

9、编写js函数，用于获得输入参数的后缀名，如输入abc.txt，返回.txt

```
function getSuffix(filename){  
    var dotLast = filename.lastIndexOf(".");  
  
    if(dotLast===-1){  
        return "文件名格式不正确";  
    }else{  
        return filename.substr(dotLast);  
    }  
}  
  
alert( getSuffix("a.jpeg") );
```

10、写出下列代码的执行结果

(1)

```
var i=2;  
  
for(var i=0,j=0;i<5;i++){  
    console.log(i);  
}  
  
console.log(i);
```

```
0,1,2,3,4,5
```

(2)

```
var a = (1&&2&&5) || 3;
```

```
console.log(a);
```

```
5
```

(3)

```
var a = [5,6];
```

```
var b = a;
```

```
b[0] = 'MyValue';
```

```
console.log(a);
```

```
["MyValue", 6]
```

11、完成下列程序段，实现b数组拷贝a数组，实现每个元素的拷贝(方法越多越好)

```
var a = [1,"yes",3];
```

```
var b;
```

方法一:

```
var a = [1,"yes",3];
```

```
var b = new Array();
```

```
for(var i=0; i<3; i++){
```

```
    b.push(a[i]);
```

```
}
```

```
alert(b);
```

方法二:

```
var a = [1,"yes",3];
```

```
var b = [].concat(a);
```

```
alert(b);
```

方法三:

```
var a = [1,"yes",3];
```

```
var b = a.slice(0,a.length);
```

```
alert(b);
```

12、写出jQuery事件绑定方法（至少三种）

```
on() bind() live() delegate()
```

13、描述parent()、parents()与closest()方法的区别，find()与filter()方法的区别

parent(): 查找父元素

parents(): 查找祖先元素

closest(): 从自身开始查找祖先元素，找到即刻返回

find(): 查找后代元素

filter(): 筛选集合中的元素

14、写出选择选择器~、>、+、“”对应的四个方法

nextAll() children() next() find()

第三部分: Ajax及后端相关

15、简述JSONP实现的方法，写出关键语句

```
<script type="text/javascript"
src="http://server2.example.com/RetrieveUser?UserId=1823&jsonp=parseResponse">
</script>

parseResponse({"Name": "Cheeso", "Id" : 1823, "Rank": 7})
```

16、已知XHR对象实例创建为xhr，写出向后台index.php同步发送数据为name=user, age=20的语句

```
xhr.open('POST', 'index.php', false);

xhr.send('name=user&age=2');
```

17、已知后台数据页面为data.json，写出获得改json文件，异步，成功在控制台打印“OK”，失败打印“fail”的语句。（应用jQuery框架实现）

```
$.ajax({
    url: "data.json",
    dataType: "json",
    data: "name=user&age=20",
    success: function(dada) {
        console.log('ok');
    },
    error: function() {
        console.log('fail');
    }
})
```

18、写出应用PHP函数打印 2014-05-10 17:53:27 的语句

```
echo date("Y-m-d H:i:s");
```

19、编写PHP程序，实现将字符串"a|b|c|d"拆分开，分别在页面上换行打印

```
$str = "a|b|c|d";  
$exp = explode("|",$str);  
for(var i=0;i<count($exp);i++){  
    echo $exp[i]."<br/>";}
```

20、写出 SQL语句的格式：插入，更新，删除

表名：User

表字段和数据如下：

Name	Tel	Content	Date
张三	13333663366	大专毕业	2006-10-11
张三	13612312331	本科毕业	2006-10-15
张四	021-55665566	中专毕业	2006-10-15

(a) 有一新记录(小王 13254748547 高中毕业 2007-05-06)请用SQL语句新增至表中

(b) 请用sql语句把张三的时间更新成为当前系统时间

(c) 请写出删除名为张四的全部记录

```
insert into User(Name,Tel,Content,Date) values('小王','13254748547','高中毕业','2007-05-06');  
update User set Date=now() where Name='张三';  
delete from User where Name='张四'
```

Web前端笔试题(第二套)



1、七只皮鞋逻辑题。（5分）

这些皮鞋中有六只从逻辑的角度看属于同一类型。还有一只是“另类”——是哪一只，为什么？

第七只鞋 高跟鞋没鞋带 低跟的有鞋带 第七只鞋是高跟鞋并且有鞋带

2、君王的遗愿逻辑题。（5分）

古代有一位君王，是国际象棋的迷恋者。他在临终之前，最放心不下的，就是那笔巨额财产的归属。这笔财产应该传给他三个儿子中的哪一个呢？他的财产其实就是用钻石宝珠制成的国际象棋。他决定，每个儿子可以从现在开始下棋，他的这笔财产将给予下棋盘数正好等于他存活天数一半的儿子。大儿子拒绝了，他说不知道父亲还能活多久。二儿子也拒绝了，理由同上。小儿子接受了。他怎样才能遵从他父亲的愿望呢？

每两天下一盘棋

1. 两种算法实现打印5的倍数。（每种算法5分，共10分）

方法一：

```
function sum (end) {
    for (var i=0; i<=end; i++) {
        if (i%5==0) {
            console.log (i);
        }
    }
}
```

方法二：

```
function sum (end) {
    for (var i=5; i<=end; i+=5) {
        console.log (i);
    }
}
```

1. 求二维数组所有数的和 (封装函数)。（10分）


```
function sum(arr){
    var total=0;
    for(var i=0;i<arr.length;i++){
        for(var j=0;j<arr[i].length;j++){
            total+=arr[i][j]
        }
    }
    return total;
}
```

1. 用户输入秒数，点击确定后开始倒计时。（20分）

```
<input type="text" placeholder="请输入秒数" id="sec"/>
<button id="show">确定</button>
<div id="secshow"></div>
Show.onclick=function (){
    Var secT=parseInt(sec.value);
    Var time=setInterval(function(){
        If(secT=0){
            secshow.innerHTML=secT--;
        }else{
            clearInterval(time);
        }
    },1000)
}
```

1. 用js实现随即选取10--100之间的10个数字，存入一个数组。（20分）

```
function select(start,end){
    var o=end-start+1;
    return Math.floor(Math.random()*o+start);}
var arr=new array();
for(var i=0;i<10;i++){
    arr.push(select(10,100));
}
alert(arr);
```

1. 请使用javascript语言创建一个对象代表一个学生,学生主要有以下属性:姓名Jeriy(字符串类型)/年龄22(整型)/三个朋友:Li、Chen、Zhang（数组）/会踢足球(类型为方法),并且调用Jeriy踢足球(弹出football字符串即可)。要求使用构造函数和原型组合方法。（30分）

```
function xs(name,age,friend){
    this.name=name;
    this.age=age;
    this.friend=friend;
}
xs.prototype.play=function(){
    alert("football")
}
var s = new xs("Jeriy",22,["li","chen","zhang"]);
```

```
s.play();
```

Web前端笔试题（三）

1. 使用Js: 根据当前时间段在页面上给出不同提示语（至少分5个时间段）。

```
getDate()
```

```
getDay()
```

```
getMonth()
```

```
getFullYear()
```

```
getHours()
```

```
getMinutes()
```

```
getMilliseconds()
```

2、使用JS显示地址栏参数p1、p2和p3，点击链接跳转到另一页面，并附参数p3、p2、p1（参数顺序依然是p1-p3，但取值颠倒）

```
window.onload = function(){
    function parseQueryString(url){
        var params = {};
        var arr = url.split("?");
        if (arr.length <= 1)
            return params;
        arr = arr[1].split("&");
        for(var i=arr.length-1; i>=0; i--){
            var a = arr[i].split("=");
            params[a[0]] = a[1];
        }
        return params;
    }
    var ps = parseQueryString(location.href);

    var str = "?";
    var count = 0;
    for( i in ps) {
        count++;
        str += ("p"+count) + "=" + ps[i] + "&";
    }
    var a = document.getElementsByTagName("a")[0];
    a.onclick = function(e){
        location.href = this.href + str.substr(0,str.length-1);
    }
}
```

```

        return false;
    }
}

```

3、判断中用户输入的字符串中是否含有“shianyun”

```

/shianyun/i.test()

```

4、显示中用户输入的字符串中含有几个a（不区分大小写）

```

var str = "cloucccccdchen";
var find = "c";
var reg = new RegExp(find,"g")
var c = str.match(reg);
alert(c?c.length:0);

```

5、设计js程序实现伪登录（仅当用户输入admin，密码123456时提示登录成功）：

Login (username,password) 必须有，仅用于判断登录是否成功。

```

function Login( username, password ){
    if( username === 'admin' && password === '123456' ){
        alert('登录成功');
    }
}

```

6、编写js函数，用于测试输入的字符串是不是如下格式：xxx-xxx-xxxx-0，x为0-9的数字。

```

/\d{3}-\d{3}-\d{4}-0/.test()

```

1. 请实现，鼠标点击页面中的任意标签，alert请标签的名称（注意兼容性）

```

<!DOCTYPE html>
<html>
<head>
<meta charset='utf-8' />
<title>鼠标点击页面中的任意标签，alert该标签的名称</title>
<style>
div{ background:#0000FF;width:100px;height:100px;}
span{ background:#00FF00;width:100px;height:100px;}
p{ background:#FF0000;width:100px;height:100px;}
</style>
<script type="text/javascript">
document.onclick = function(evt){
    var e = window.event || evt;
    var tag = e["target"] || e["srcElement"];
    alert(tag.tagName);
};
</script>

```

```

</head>
<body>
<div id="div"><span>SPAN</span>DIV</div>
<span>SPAN</span>
<p>P</p>
</body>
</html>

```

1. 请用javascript找出所有ClassName包含text的标签, 并将他们背景颜色设置成黄色

```

function changeBackgroundColor(tagName){

var eles=null;

eles=ele.getElementsByTagName(tagName)

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

    if(eles[i].className.search(new RegExp("\\b" + className +

"错误！超链接引用无效。用正则表达式来判断是不是包含此类名

    eles[i].style.backgroundColor="yellow";

    }

}

```

Web前端笔试题（四）

- 将字符串 border-left-color, moz-viewport 转换成 borderLeftColor, mozViewport格式(封装函数)

```

function toCamelCase( str ){
    var subStr = str.split("-");
    var str = subStr[0];
    for( i = 1; i < subStr.length; i++ ){
        str += subStr[i].substr(0,1).toUpperCase()+subStr[i].substr(1);
    }
    return str;
}
alert( toCamelCase('border-left-color') );

```

- 输出明天的日期

```

<script type="text/javascript">

    function GetDateStr(AddDayCount) {

        var dd = new Date();

        dd.setDate(dd.getDate()+AddDayCount);//获取
AddDayCount天后的日期

        var y = dd.getFullYear();

```

```

        var m = dd.getMonth()+1;//获取当前月份的日期

        var d = dd.getDate();

        return y+"-"+m+"-"+d;

    }

    document.write("前天: "+GetDateStr(-2));

    document.write("<br />昨天: "+GetDateStr(-1));

    document.write("<br />今天: "+GetDateStr(0));

    document.write("<br />明天: "+GetDateStr(1));

    document.write("<br />后天: "+GetDateStr(2));

    document.write("<br />大后天: "+GetDateStr(3));

</script>

```

3. 完成下列功能

1. 为字符串扩展删除左侧、右侧及左右两侧空格

1. 消除字符串左边的空格

```

function leftTrim(str){

return str.replace(/^\\s*/,"");//^符号表示从开头即左边进行匹配

}

```

2. 消除字符串右边的空格

```

function rightTrim(str){

return str.replace(/\\s*$/,"");

}

```

3. 消除字符串两边的空格

```

function trim(str){

return str.replace(/(^\\s*)|(\\s*$)/g,"");

}

//alert("111"+trim(" aaa ")+ "111");

```

1. 为数组扩展indexOf、remove及removeat方法

```

1.var oldArrayIndexOf = Array.indexOf;//判断是否原始浏览器是否存在indexOf方法
array.prototype.indexOf = function(obj) {
    if(!oldArrayIndexOf) {

```

```

    for(var i = 0, imax = this.length; i < imax; i++) {
        if(this[i] === obj) {
            return i;
        }
    }
    return -1;
} else{
    return oldArrayIndexOf(obj);
}
}

```

```

2. Array.prototype.remove = function (dx) {
    if (isNaN(dx) || dx > this.length) {
        return false;
    }
    for( var i = dx; i < this.length; i++ ){
        this[i] = this[i+1];
    }
    this.length -= 1;
};
var arr = ['a','b','a','a'];
arr.remove( 1 );
alert(arr);

```

```

3. Array.prototype.removeAt=function(index){
    this.splice(index,1); }

```

- 实现兼容的getElementsByClassName()方法

```

var $ = {
    getEleByClass: function( cls, parent, tag ){
        var parent = parent || document;
        var tag = tag || "*";
        if( parent.getElementsByClassName ){
            return parent.getElementsByClassName( cls );
        } else {
            var aClass = [];
            var reg = new RegExp( "(^| )" + cls + "( |$)" );
            var aEle = this.getEleByTag( tag, parent );
            for( var i = 0, len = aEle.length; i < len; i++ ){
                reg.test( aEle[i].className ) && aClass.push( aEle[i] );
            }
            return aClass;
        }
    },
    getEleByTag: function( ele, obj ){
        return ( obj || document ).getElementsByTagName( ele );
    },
    hasClass: function( ele, cls ){
        return ele.className.match( new RegExp( "(^|\\s)" + cls + "(错误！超链接引用无效。)" ));
    },
    addClass: function( ele, cls ){
        if( !$hasClass( ele, cls ) ){

```

```

        ele.className += " " + cls;
    }
},
removeClass: function( ele, cls ){
    if( $.hasClass( ele, cls ) ){
        var reg = new RegExp( "(^|\\s)" + cls + "(错误! 超链接引用无效。)" );
        ele.className = ele.className.replace( reg, " " );
    }
}
};

```

- 实现查找对象的方法（三组）
- 实现获得和设置对象的宽和高
- 实现给对象设置和获得属性

```

var a=document.getElementById("tet");
/*
alert(a.getAttribute("title"));

a.setAttribute("name","cc");

alert(a.getAttribute("name"));*/

```

- 实现hasClass、addClass及removeClass的功能（二组）
- 根据以下xml请写出对应的json，并解析在页面上，要求以表格格式打印（要求兼容所有浏览器）（一组）

```

<xml>
<list>
    <item>
        <id>12</id>
        <name>张三</name>
    </item>
    <item>
        <id>13</id>
        <name>李四</name>
    </item>
</list>
</xml>

```

```

demo.json
[
  {
    "id": 12,
    "name": "张三"
  },
  {
    "id": 13,
    "name": "李四"
  }
]

```

```

]

var str = "";
$.ajax({
    url:"demo.json",
    dataType: "json",
    async: false,
    success:function(data){
        $.each( data,function(index,value){
            str+="|<td>"+value.cid+"</td><td>"+value.ename+"</td></tr>"
        });
    }
});
$('table').html(str);

|  |

```

- 实现isArray、inArray功能（一组）

```

function isArray(obj) {
    return Object.prototype.toString.call(obj) === '[object Array]';
}

//needle 待检测字符串
//haystack 数组或者是以|分割的字符串
function in_array(needle,haystack) {
    haystack=isArray(haystack)?haystack:haystack.split("|");
    if(typeofneedle =='string' || typeofneedle =='number') {
        for(var i in haystack) {
            if(haystack[i] == needle) {
                Return true;
            }
        }
    }
    return false;
}

```

Web前端笔试题（五）

一、函数

- 1.应用JavaScript函数递归打印数组到HTML页面上，数组如下：

```

var item = [
    {
        name:'Tom',
        age: 70,
        child: [{
            name:'Jerry',
            age: 50,
            child: [
                name: 'William',

```



```

        age: 20,
        child: [.....]
    ],
    {
        name: 'Jessi',
        age: 30
    }
]
}
]

```

输出的HTML如下:

```

<ul>
  <li>Name:Tom</li>
  <li>Age:70</li>
  <li>Child:
    <ul>
      <li>Name: Jerry</li>
      <li>Age: 50</li>
      <li>Child:
        <ul>
          <li>Name: William</li>
          <li>Age: 20</li>
          <li>Child: .....</li>
        </ul>
      </li>
    </ul>
  <ul>
    <li>Name: Jerry</li>
    <li>Age: 50</li>
  </ul>
</li>
</ul>
var str = "";
function displayProp(obj){
  str += "<ul>";
  for( var i in obj ){
    if( typeof obj[i] == "string" ){
      str += "<li>" + ( i + "/" + obj[i] ) + "</li>";
    }else{
      str += "<li>";
      displayProp( obj[i] );
      str += "</li>";
    }
  }
  str += "</ul>";
}
displayProp( item );
//alert( str );

```

```
document.getElementById('box').innerHTML = str;
```

二、闭包

1. 举例说明变量的作用域。举例说明如何从外部读取局部变量。

```
//函数内部直接读取全局变量
//var n=999;
function f1(){
    //alert(n);
}
f1(); // 999
//在函数外部自然无法读取函数内的局部变量。
function f2(){
    a=44;
}
f2();
alert(a); // 999
//如何从外部读取局部变量?
function f1(){
    n=111; //局部变量对fa可见
    function f2(){
        alert(n); //f2内部的局部变量，对f1 就是不可见的
    } //子对象会一级一级地向上寻找所有父对象的变量
    return f2; //f1读取f2的内部变量 只要把f2作为返回值
}
//var result=f1();
//result(); // 999
```

2. 简述闭包的概念。

闭包的概念：定义在一个函数内部的函数

3. 简述闭包的用途。

闭包的用途：可以读取函数内部的变量，让这些变量的值始终保持在内存中

4. 使用闭包注意的注意点。

造成内存泄漏

会影响浏览器的反应速度境地用户体验或造成浏览器无响应

5. 说出下列两段代码的输出结果？解释为什么。

```
var name = "The Window";
var object = {
    name : "My Object",
    getNameFunc : function(){
        return function(){
            return this.name;
        };
    }
};
```

```
alert(object.getNameFunc()());
```

The Window

/*在闭包中使用this对象也可能会导致一些问题, this 对象是在运行时基于函数的执行环境绑定的, 如果this在全局范围就是window, 如果在对象内部就指向这个对象。而闭包却在运行时指向window的, 因为闭包并不属于这个对象的属性或方法。 */

```
var name = "The Window";
var object = {
    name : "My Object",
    getNameFunc : function(){
        var that = this;
        return function(){
            return that.name;
        };
    }
};
alert(object.getNameFunc()());
```

My Object

/*javascript是动态（或者动态类型）语言，this关键字在执行的时候才能确定是谁。所以this永远指向调用者，即对'调用对象'者的引用。第一部分通过代码：object.getNameFunc()调用返回一个函数。这是个返回的函数，它不在是object的属性或者方法，此时调用者是window。因此输出是 The Window

第二部分，当执行函数object.getNameFunc()后返回的是：

```
function() {
    return that.name;
}
```

此时的that=this。而this指向object，所以that指向object。无论你执行多少次，他都是对object的引用，所以输出My Object*/

6. 用三种方法实现打印li的索引（必须有闭包的方法）

方法一

```
var lis = document.getElementsByTagName("li");

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

    lis[i].setAttribute("data-num", i);

}

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

    lis[i].onclick = function(){

        alert( this.getAttribute("data-num") );

    }

}
```

//方法二

```
var lis = document.getElementsByTagName("li")

//foa = 0;

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

    lis[i].index = i;
```

```

lis[i].onclick = function(){
    alert( this.index );
}
}

```

方法三:

```

for(var i=0, len = lis.length; i<len; i++){
    (function(i){
        lis[i].onclick = function(){
            alert( lis[i] );
        }
    })(i);
}

```

三、面向对象

1. 完成下列JS面向对象案例

- 定义父类: Shape (形状) 类, Shape 只用一个属性 color, 并有相应的 getColor 和 setColor 方法
- Shape 类有两个子类: Rectangle (矩形) 类和 Circle (圆形) 类, 子类继承了父类的 color 属性和 getColor、setColor 方法。
- 为两个子类增加相应的属性和 getArea 方法, 可调用 getArea 方法获得矩形和圆形的面积。

/*//定义形状类

```

function Shape( color ){
    this.color = color;
}

Shape.prototype.getColor = function(){
    return this.color;
}

Shape.prototype.setColor = function( newColor ){
    this.color = newColor;
}

```

//定义矩形子类

```

function Rectangle( color){

```

```

        Shape.call( this, color );
    }

    //实现继承

    Rectangle.prototype = new Shape();

    Rectangle.prototype.getArea=function( width, height ){

        return width*height;

    }

    //定义圆形子类

    function Circle(){

        Shape.call( this, color );

    }

    //实现继承

    Circle.prototype = new Shape();

    Circle.prototype.getArea = function( radius ){

        return Math.PI*Math.pow(radius,2);

    }

    var r1 = new Rectangle( "blue" );

    alert( r1.getColor() );

    alert( r1.getArea( 30, 40 ) );

    /*var s1 = new Shape( "yellow" );

    alert( s1.getColor() );

    s1.setColor( "red" );

    alert( s1.getColor() );*/

```

四、ajax 与 json

1.应用原生JS实现ajax封装

```

var createAjax = function() {

    var xhr = null;

    try {

        //IE系列浏览器

```

```

        xhr = new ActiveXObject("microsoft.xmlhttp");
    } catch (e1) {

        try {

            //非IE浏览器

            xhr = new XMLHttpRequest();

        } catch (e2) {

            window.alert("您的浏览器不支持ajax，请更换！");

        }

    }

    return xhr;

};

//核心函数。

var ajax = function(conf) { // 初始化

    var type = conf.type; //pe参数,可选

    var url = conf.url; //url参数,必填

    var data = conf.data; //data参数可选，只有在post请求时需要

    var dataType = conf.dataType; //datatype参数可选

    var success = conf.success; //回调函数可选

    if (type == null) {

        type = "get"; //pe参数可选，默认为get

    }

    if (dataType == null) {

        dataType = "text"; //dataType参数可选，默认为text

    }

    var xhr = createAjax(); // 创建ajax引擎对象

    xhr.open(type, url, true); // 打开

    if (type == "GET" || type == "get") { // 发送

        xhr.send(null);

    } else if (type == "POST" || type == "post") {

        xhr.setRequestHeader("content-type",

```

```

        "application/x-www-form-urlencoded");

        xhr.send(data);
    }

    xhr.onreadystatechange = function() {

        if (xhr.readyState == 4 && xhr.status == 200) {

            if (dataType == "text" || dataType == "TEXT") {

                if (success != null) {

                    success(xhr.responseText); //普通文本

                }

            } else if (dataType == "xml" || dataType == "XML") {

                if (success != null) {

                    success(xhr.responseXML); //接收xml文档

                }

            } else if (dataType == "json" || dataType == "JSON") {

                if (success != null) {

                    success(eval("(" + xhr.responseText + ")")); //将json字符串转换为js对象

                }

            }

        }

    };

};

//此函数的用法。

ajax({

    type: "post",

    url: "test.jsp",

    data: "name=dipoo&info=good",

    dataType: "json",

    success: function(data) {

        alert(data.name);

    }

}

```

```
});
```

2.应用原生JS实现JSON的封装（兼容浏览器）

```
function ShowJsonString(){  
    response = ("[{  
        name: 'Joe',  
        age: '30',  
        gender: 'M'  
    }, {  
        name: 'Chandler',  
        age: '32',  
        gender: 'M'  
    }, {  
        name: 'Rose',  
        age: '31',  
        gender: 'M'  
    }]  
");//字符串形式  
};  
  
var response1 = "{(  
    name: 'Vicson',  
    age: '30',  
    gender: 'M'  
    )}  
";//字符串形式,这里的小括号不能少  
  
json = eval(response);  
json1 = eval(response1);  
  
alert(json[0].name + "," + json[1].age + "," + json[2].gender);  
alert(json1.name);  
}  
  
ShowJsonString();*/
```

Web前端笔试题（六）

1 , JavaScript框架都有哪些 (国外和国内)

国外: jQuery, MooTools, Dojo, ExtJS, YUI3

国内:

- Arale(支付宝) - <http://aralejs.org/>
- Como(盛大) - <http://www.comsome.com>
- EasyJs - <https://github.com/chenmnkken/easyjs>
- EdoJs - <http://www.edojs.com/>
- DWZ - <http://j-ui.com/>
- **JX(腾讯)** - <http://alloyteam.github.com/JX>
- JSI - <http://code.google.com/p/jsi/>
- **KISSY(淘宝)** - <http://www.kissyui.com>
- KindEditor - <https://github.com/kindsoft/kindeditor>
- Mass - <https://github.com/RubyLouvre/mass-Framework>
- ModJS(腾讯) - <http://madscrip.com/modjs/>
- ModuleJS(腾讯) - <http://modulejs.github.com/modulejs/>
- NEJ(网易) - <http://nej.netease.com/>
- NJF - <http://code.google.com/p/njf/>
- Puerh(百姓网) - <https://github.com/baixing/Puerh>
- QWrap(百度) - <http://www.qwrap.com>
- SeaJS(淘宝) - <http://seajs.org/>
- **Tangram(百度)** - <http://tangram.baidu.com>
- **UEditor(百度)** - <http://ueditor.baidu.com>
-

2 , Web页面的测试工具都有哪些 ?

YSlow:

<http://www.kuqin.com/webpagedesign/20080513/8446.html>

<http://www.kuqin.com/webpagedesign/20080513/8441.html>

FireBug:

Chrome Developer Tools

3 , display:none和visibility:hidden的区别是什么 ?

visibility: hidden----将元素隐藏,但是在网页中该占的位置还是占着。

display: none----将元素的显示设为无,即在网页中不占任何的位置。

4 , 如何为元素绑定多个事件 ? (要求同时支持ie和火狐)

```
var addHandler = function(element, type, handler){  
    if (element.addEventListener){  
        element.addEventListener(type, handler, false);
```

```

    } else if (element.attachEvent){

        element.attachEvent( "on" + type, handler);

    } else {

        element[ "on" + type] = handler;

    }

}

```

5 , 列出ie和火狐事件对象的不同点

```

Var getEvent = function( event ){

    return event ? event : window.event;

}

```

6 , 请简单说明js实现拖动的原理

当mousedown时记下鼠标点击位置离拖拽容器左边沿的距离和上边沿的距离；mousemove时通过定位拖拽容器的style.left/style.top，使拖拽容器进行移动，定位到哪里则由刚刚的tmpX/tmpY和当前鼠标所在位置计算得出；mouseup时，结束移动。

<http://wsqinwei00.blog.163.com/blog/static/46512701201372102517293/>

7 , 前端页面由哪三层构成，分别是什么？作用是什么？

网页的结构层 (structural layer) 由 HTML 或 XHTML 之类的标记语言负责创建。标签，也就是那些出现在尖括号里的单词，对网页内容的语义含义做出了描述，但这些标签不包含任何关于如何显示有关内容的信息。例如，p 标签表达了这样一种语义：“这是一个文本段。”

网页的表示层 (presentation layer) 由 CSS 负责创建。CSS 对“如何显示有关内容”的问题做出了回答。

网页的行为层 (behavior layer)

负责回答“内容应该如何对事件做出反应”这一问题。这是 Javascript 语言和 DOM 主宰的领域。

网页的表示层和行为层总是存在的，即使我们未明确地给出任何具体的指令也是如此。此时，Web 浏览器将把它的默认样式和默认事件处理函数施加在网页的结构层上。例如，浏览器会在呈现“文本段”元素时留出页边距，有些浏览器会在用户把鼠标指针悬停在某个元素的上方时弹出一个显示着该元素的 title 属性值的提示框，等等。

分离

在所有的产品设计活动中，选择最适用的工具去解决问题是最基本的原则。具体到网页设计工作，这意味着：

- (1) 使用 (X)HTML 去搭建文档的结构。
- (2) 使用 CSS 去设置文档的呈现效果。
- (3) 使用 DOM 脚本去实现文档的行为。

8, 标准浏览器和不标准浏览器 (部分ie) 添加与移除侦听事件的函数方法分别是什么?

```
addHandler: function(element, type, handler){
    if (element.addEventListener){
        element.addEventListener(type, handler, false);
    } else if (element.attachEvent){
        element.attachEvent( "on" + type, handler);
    } else {
        element[ "on" + type] = handler;
    }
},
removeHandler: function(element, type, handler){
    if (element.removeEventListener){
        element.removeEventListener(type, handler, false);
    } else if (element.detachEvent){
        element.detachEvent( "on" + type, handler);
    } else {
        element[ "on" + type] = null;
    }
}
```

9, 如何提高网页运行性能?

尽量减少HTTP请求

减少DNS查找

避免跳转

缓存Ajax

推迟加载

提前加载

减少DOM元素数量

用域名划分页面内容

减小iframe的大小

避免404错误

<http://www.kuqin.com/webpagedesign/20080513/8446.html>

• 简述javascript的封装

对象的定义; 对象的继承; 单例模式; 私有属性、私有方法、公有属性、公有方法

• JS主要数据类型

Undefined、Null、Boolean、Number和String；Object

- **如何解决ajax的跨域问题**

JSONP

- **请写出三种减低页面加载时间的方法**

- (1) 尽量减少页面中重复的HTTP请求数量
- (2) 服务器开启gzip压缩
- (3) css样式的定义放置在文件头部
- (4) Javascript脚本放在文件末尾
- (5) 压缩Javascript、CSS代码
- (6) Ajax采用缓存调用
- (7) 尽可能减少DOM元素
- (8) 使用多域名负载网页内的多个文件、图片
- (9) 应用CSS Sprite

<http://www.seowhy.com/bbs/thread-495488-1-1.html>

- **列举三种强制类型转换和两种隐式类型转换**

<http://www.nixi8.com/javascript-data-types-cast-and-implicit-conversion.html>

- **split()和join()的区别**

join() 方法用于把数组中的所有元素放入一个字符串。元素是通过指定的分隔符进行分隔的。指定分隔符方法join("#"); 其中#可以是任意。

与之相反的是split()方法：用于把一个字符串分割成字符串数组.stringObject.split(a,b)这是它的语法a是必须的，决定从a这分割b不是必须的，可选。该参数可指定返回的数组的最大长度。如果设置了该参数，返回的子串不会多于这个参数指定的数组。如果没有设置该参数，整个字符串都会被分割，不考虑它的长度。

- **call和apply的区别**

- (1) **call方法**

调用一个对象的一个方法，以另一个对象替换当前对象。

call([thisObj[,arg1[, arg2[, ...,argN]]]])

参数

thisObj 可选项。将被用作当前对象的对象。

arg1, arg2, ..., argN 可选项。将被传递方法参数序列。

说明

call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由thisObj指定的新对象。

如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。

(2) apply方法

官方（JavaScript手册）：

应用某一对象的一个方法，用另一个对象替换当前对象。

`apply([thisObj[,argArray]])`

参数

thisObj 可选项。将被用作当前对象的对象。

argArray 可选项。将被传递给该函数的参数数组。

说明

如果 **argArray** 不是一个有效的数组或者不是 **arguments** 对象，那么将导致一个 **TypeError**。

如果 没有提供 **argArray** 和 **thisObj** 任何一个参数，那么 **Global** 对象将被用作 **thisObj**，并且无法被传递任何参数。

两者的区别：

两者实现的功能是完全一样的，只是参数传递方式不一样，**call**是将各个参数以逗号（,）隔开，而**apply**是将所有参数组成一个数组进行传递。

• 写一个让b继承a的方法

```
function A( age, name ){
    this.age = age;
    this.name = name;
}

A.prototype.show = function(){

    alert('父级方法');

}

function B(age,name,job){
    A.apply( this, arguments );
    this.job = job;
}

B.prototype = new A();

var b = new A(14,'侠客行');

var a = new B(15,'狼侠','侠客');

a.show();
```

• ==和===的区别

“==”与“===”是不同的,一个是判断值是否相等,一个是判断值及类型是否完全相等。

http://zhidao.baidu.com/link?url=IP7UC8Tzum2b0G-bCA6Uns1eq06pKdhOrtpLs9CYYSD3694PRmqxzUWnev5Ziu61pTetITHY_1hxjqUZBmyxyK

• 列举常用的操作DOM的方法以及他的作用

见DOM-operation.gif

• JS实现一个类，包含私有属性，公有属性，私有方法和公有方法

```
function myObject () {  
    var x = 20;  
    var privateFoo = function() {  
        return 30;  
    }  
    this.y = 50;  
    this.publicFoo = function(){  
        return x;  
    }  
}  
var f = new myObject();  
alert( f.publicFoo() );
```

1301A笔试题（七）

1. 简述setTimeout和setInterval的作用以及区别

setTimeout方法是**超时调用**，也就是在什么时间以后干什么。干完了就拉倒。setInterval方法**间歇调用**，则是表示间隔一定时间反复执行某操作。

setInterval() 方法可按照指定的周期（以毫秒计）来调用函数或计算表达式。

setInterval() 方法会不停地调用函数，直到 clearInterval() 被调用或窗口被关闭。由 setInterval() 返回的 ID 值可用作 clearInterval() 方法的参数。

1. 简述Ajax的实现步骤

要完整实现一个AJAX异步调用和局部刷新,通常需要以下几个步骤:

- (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.
- (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.

1. 同步和异步的区别

举个例子：普通B/S模式（同步）AJAX技术（异步）

同步：提交请求->等待服务器处理->处理完毕返回 这个期间客户端浏览器不能干任何事

异步：请求通过事件触发->服务器处理（这是浏览器仍然可以作其他事情）->处理完毕

看看open方法的几个参数。

```
.open (http-method, url, async, userID, password)
```

(后面是帐号和密码, 在禁止匿名访问的http页面中, 需要用户名和口令)

其中async是一个布尔值。如果是异步通信方式(true), 客户机就不等待服务器的响应; 如果是同步方式(false), 客户机就要等到服务器返回消息后才去执行其他操作。我们需要根据实际需要来指定同步方式, 在某些页面中, 可能会发出多个请求, 甚至是有组织有计划有队形大规模的高强度的request, 而后一个会覆盖前一个的, 这个时候当然要指定同步方式: false。

1. post和get的区别

(1)、Get是用来从服务器上获得数据, 而Post是用来向服务器上传递数据。

(2)、Get将表单中数据的按照variable=value的形式, 添加到action所指向的URL后面, 并且两者使用“?”连接, 而各个变量之间使用“&”连接; Post是将表单中的数据放在form的数据体中, 按照变量和值相对应的方式, 传递到action所指向URL。

(3)、Get是不安全的, 因为在传输过程, 数据被放在请求的URL中。Post的所有操作对用户来说都是不可见的。

(4)、Get传输的数据量小, 这主要是因为受URL长度限制 (GET方式提交的数据最多只能有1024字节, 而POST则没有此限制); 而Post可以传输大量的数据, 所以在上传文件只能使用Post。

(5)、Get限制Form表单的数据集的值必须为ASCII字符; 而Post支持整个ISO10646字符集。默认是用ISO-8859-1编码。

(6)、Get是Form的默认方法。

1. 一个页面能否能同时提交到两个页面, 写出代码

```
<script>

function subform(i) {

    var theform=document.form1;

    theform.action=i;

    theform.submit();

}

</script>

<form name="form1">

    <input type="text" value="1fdsa"/>

    <input type="button" onClick="subform('11.php')" value="提交" />

    <input type="button" onClick="subform('22.php')" value="新" />

    <input type="button" onClick="subform('33.php')" value="新2" />
```

</form>

1. 分别说明在正则表达式中test exec match search replace split方法的作用是什么

exec: 对string进行正则处理,并返回匹配结果.array[0]为原字符串,array[i]为匹配在整个被搜索字符串中的位置。

test: 测试string是否包含有匹配结果, 包含返回true, 不包含返回false。

match(pattern): 根据pattern进行正则匹配,如果匹配到,返回匹配结果,如匹配不到返回null

search(pattern): 根据pattern进行正则匹配,如果匹配到一个结果,则返回它的索引数;否则返回-1

replace(pattern,replacement): 根据pattern进行正则匹配,把匹配结果替换为replacement

split(pattern): 根据pattern进行正则分割,返回一个分割的数组

```
function checkForm() {  
    var u = document.form_name.check.value;  
  
    var s =  
/^[-a-zA-Z0-9_]+(\.[-a-zA-Z0-9_]+)*@[-a-zA-Z0-9_]+(\.[-a-zA-Z0-9_]+)*$/;  
  
    var a = s.exec(u);  
  
    var a = s.test(u);  
  
    var a = u.match(s);  
  
    var a = u.search(s);  
  
    alert(a);  
}
```

replace:

```
var u = "javascript is a good script language";
```

//在此我想将字母a替换成字母A

```
var s = /a/g;
```

```
var a = u.replace(s, "A");
```

split:

```
var str="How are you doing today?"
```

```
document.write(str.split(" ") + "<br />")
```

```
document.write(str.split("") + "<br />")
```

```
document.write(str.split(" ",3))
```

1. Js中怎么定义对象的构造函数,怎么体现js的继承关系,举例说明


```
function A( age, name ){
    this.age = age;
    this.name = name;
}

A.prototype.show = function(){
    alert('父级方法');
}

function B(age,name,job){
    A.apply( this, arguments );
    this.job = job;
}

var b = new A(14,'侠客行');
var a = new B(15,'狼侠','侠客');
a.show();
```

1. 请写出dom常用的节点操作

见附件脑图

1. 请写出常用的HTTP头信息并说明其作用

见附件《http头信息详解 对http1.0和http1.1的常用头做出详细解释》

1. 你常用的开发环境是怎样的

Sublime Text3 + Chrome Developer Tools + FireBug

1. 简单描述一下你制作一个网页的工作流程

现场发挥：切图文件命名，html结构搭建，css样式编写，注意松耦合，JS交互效果和性能，代码压缩，考虑使用框架……

1. 你更喜欢在哪个浏览器下进行开发？你使用哪些开发人员工具？

Chrome Developer Tools FireBug

1. 你如何对网站的文件和资源进行优化？

Css Sprite (CSS精灵)

Reset.css

资源服务器

seaJS应用

1. 为什么利用多个域名来存储网站资源会更有效?

加载DOM与加载资源是异步实现的，减轻单个服务器的负担

1. 你是否研究过你所使用的js库或者框架的源代码?

研究过，如jQuery

1. undefined变量和undeclared变量分别指什么?

undefined定义（或声明）了但未初始化值，undeclared为声明或为定义

1. 请举例出一个匿名函数的典型案例

```
;(function($) {  
  
    // Code goes here  
  
})(jQuery);
```

1. 请解释什么是js的模块模式，请举出实用实例

```
var Module = (function () {  
  
    var my={},  
  
        privateVar = 8;//私有属性  
  
    function privateFun() { //私有方法  
  
        return ++privateVar;  
  
    };  
  
    my.publicVar = 1;//公共属性  
  
    my.moduleFun = function () { //公共方法  
  
        return privateFun();  
  
    };  
  
    return my;  
  
})();  
  
console.log(Module.publicVar);//1  
  
console.log(Module.publicFun());//9
```

1. 冒泡排序

1. 事件绑定

Web前端笔试题（八）

1, 你做过的页面在那些浏览器测试过? 用过什么测试工具?

(1) Chrome, FireFox, Opera, IE6+

(2) FireBug, Chrome Developer Tools

2, 清除浮动?

(1) 为什么要清除浮动

```
<div style="background:#666;" class="clearfloat">  
  
<div style="float:left; width:30%; height:40px; ">Some Content</div>  
  
</div>
```

此时预览此代码, 我们会发现最外层的父元素float container, 并没有显示。这是因为子元素因进行了浮动, 而脱离了文档流, 导致父元素的height为零。

(2) 如何清除浮动

见附件: 《css清除浮动大全, 共8种方法》

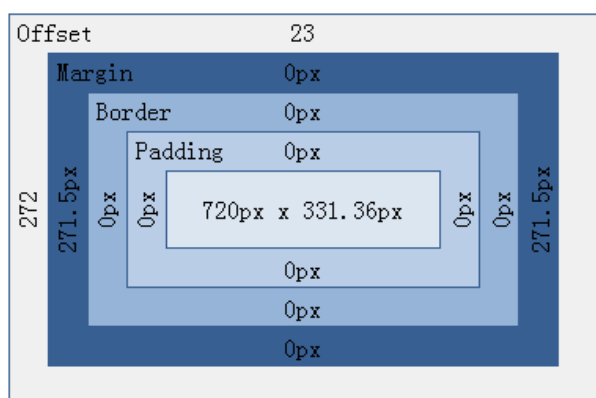
3, css3 新增属性有哪些?

- (1) 边框
- (2) 背景
- (3) 文本效果
- (4) 字体
- (5) 2D转换
- (6) 3D转换
- (7) 过渡
- (8) 动画
- (9) 多列布局
- (10) 弹性盒布局
- (11) Media Queries

(12) 选择器

参考<http://www.w3school.com.cn/css3/index.asp>

4, 画出css的盒模型



坐标: (0, 0)

Z-index = auto

5, 如何居中一个浮动元素? 假设一个绝对定位的容器, 宽500, 高300.

```
width:500px;
height:300px;
float:left;
margin-left:50%;
position:relative;
left:-250px;
```

6, 请实现下列效果, 需要兼容主流浏览器 (包括ie6): 一个div中背景色为黑色, 透明度为50%, div中的文字颜色为白色

```
div {
    background: #000;
    background: RGBA(0,0,0,0.5);
    filter:alpha(opacity=50);
    -moz-opacity:0.5;
    -khtml-opacity: 0.5;
    opacity: 0.5;
    color: #FFF;
}
```

7, 请列举你知道的关于ie6中的bug

参考附件《div+CSS浏览器兼容问题整理》

8, 列出display的值

参考《DIV CSS display (block none inline) 属性的用法》

9, Img的title和alt的异同

(1)、含义不同

alt: 使用alt属性是为了给那些不能看到你文档中图像的浏览者提供文字说明, 也就是图片显示不了的时候显示的文字。

title: 图片正常显示时, 鼠标悬停在图片上方显示的提示文字。

(2)、在浏览器中的表现不同

在firefox和ie8中, 当鼠标经过图片时title值会显示, 而alt的值不会显示; 只有在ie6中, 当鼠标经过图片时title和alt的值都会显示。

10, 都知道哪些css浏览器兼容问题

参考附件《div+CSS浏览器兼容问题整理》

11, 解决IE6的双倍边距BUG

参考附件《div+CSS浏览器兼容问题整理》

12, 如何让一个div垂直居中

参考第5题

13, 宽度自适应的三栏布局方式

参考《我熟知的三种三栏网页宽度自适应布局方法》

14, 列出五种触发ie hasLayout的属性及其值

参考《你了解IE的haslayout (拥有布局) 吗? 》

<http://www.nowamagic.net/librarys/veda/detail/1395>

15, 简述jpg , png-8, png-24, gif的区别, 分别适用场景

参考《jpg,Gif,png-8,png-24图片格式的区别》

16,xhtml和html有什么区别

HTML是一种基本的WEB网页设计语言，XHTML是一个基于XML的置标语言，看起来与HTML有些相象，只有一些小的但重要的区别，XHTML就是一个扮演着类似HTML的角色的XML，所以，本质上说，XHTML是一个过渡技术，结合了XML(有几分)的强大功能及HTML(大多数)的简单特性。

HTML 和 XHTML 的区别简单来说，XHTML 可以认为是 XML 版本的 HTML，为符合 XML 要求，XHTML 语法上要求更严谨些。

以下是 XHTML 相对 HTML 的几大区别：

XHTML 要求正确嵌套

XHTML 所有元素必须关闭

XHTML 区分大小写

XHTML 属性值要用双引号

XHTML 用 id 属性代替 name 属性

XHTML 特殊字符的处理

参见《HTML与XHTML的区别》

17,解释css sprites如何使用，并说出优缺点

参考《CSS Sprites的概念、原理、适用范围和优缺点》

18, 响应式布局有哪些优点和缺点？

参考《响应式布局这件小事有哪些优点和缺点该怎么设计》

19, 响应式布局该怎么设计？

参考《响应式布局这件小事有哪些优点和缺点该怎么设计》

20, CSS中的Media Queries（媒介查询）是什么？

参考《CSS3 Media Query实现响应布局》

1. CSS HACK如何书写

参考《css hack》

1. link与import的区别

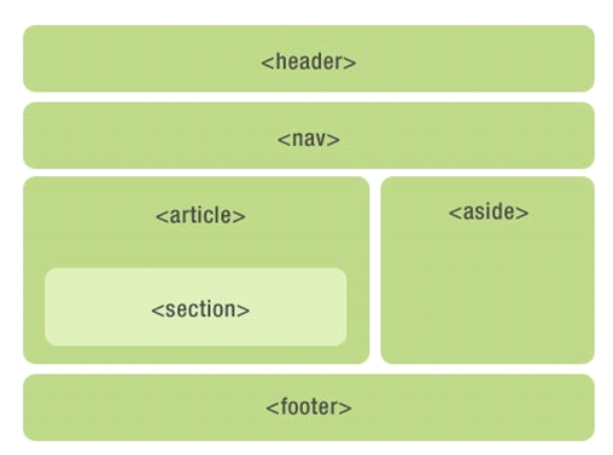
参考《link和@import的区别》

1. 解释下什么是web语义化，举出具体的实例，并说明语义化后有什么好处

HTML的每个标签都有其特定含义（语义），Web语义化是指使用语义恰当的标签，使页面有良好的结构，页

面元素有含义，能够让人和搜索引擎都容易理解。

使用这样结构写出的网页其语义显而易见。在有些面试的时候会问到类似**strong** 和 `font-weight: bold` 有什么区别，这时候就可以从语义化的角度解答了。



利于SEO

1. 请举例说明在css布局中对div自适应高度并且能兼容多浏览器的写法

```
<div style="width:auto;height:auto!important;min-height:200px;height:200px;background-color:#f4f4f4;font-size:12px">
```

1. 文档类型的作用是什么？你知道多少种文档类型？

参考《DOCTYPE与浏览器模式分析》

1. 浏览器标准模式和怪异模式的区别？

参考《DOCTYPE与浏览器模式分析》

1. 如何定义宽度很小的（如：1px）容器

```
div { background:red;overflow:hidden }
```

1. 如何使得文字不换行

```
<div style="height:35px;width:100px;white-space:nowrap;overflow-x:auto"></div>
```

1. 如何让鼠标变成手型，且兼容现代所有浏览器

```
cursor:pointer
```

1. 一个div为margin-bottom:10px，一个div为margin-top:5px，为什么2个div之间的间距是10px而不是15px？

外边距叠加问题。

在css中，两个或多个毗邻（父子元素或兄弟元素）的普通流中的块元素垂直方向上的 margin 会发生叠加。

这种方式形成的外边距即可称为外边距叠加。

margin叠加会发生在2种关系下，一种是父子元素，一种是兄弟元素。

避免外边距叠加，其实很简单：外边距发生叠加的4个必要条件（2个或多个、毗邻、垂直方向、普通流），破坏任一个即可。

参考《外边距叠加》

1. 如果一个页面上需要多种语言文字，那么网页需要什么编码格式

UTF-8

参考《网页编码就是那点事》

```
1. <style
type="text/css">#text{color:red;};p#text{color:brown};p.text{color:green
};p{color:white}</style><p id="text" class="text">请说出我的颜色</p> 请问p
中的文字颜色是什么
brown
```

1. strict和transitional这两种模式有何意义？如何声明html5？

参考《浅析网页Transitional和Strict的文档声明的区别》

1. display: none和visibility: hidden的区别是什么

display:none;

使用该属性后，HTML元素（对象）的宽度、高度等各种属性值都将“丢失”；

visibility:hidden;

使用该属性后，HTML元素（对象）仅仅是在视觉上看不见（完全透明），而它所占据的空间位置仍然存在，也即是说它仍具有高度、宽度等属性值。

Web前端笔试题（九）

1. css选择符有哪些？哪些属性可以继承？

(1) **css选择符**有哪些？

(a) 类型选择符

什么是类型选择符？指以网页中已有的标签类型作为名称的行径符。body是网页中的一个标签类型，div,p,span都是。

(b) 群组选择符

可以对一组同时进行了相同的样式指派。

使用逗号对选择符进行了分隔，这样书写的优点在于同样的样式只需要书写一次即可，减少代码量，改善CSS代码结构。


```
h1,h2,h6,p,span{}
```

(c) 包含选择符

对某对象中的子对象进行样式指点定，这样选择方式就发挥了作用。

需要注意的是，仅对此对象的子对象标签有效，对于其它单独存在或位于此对象以外的子对象，不应用此样式设置。

```
h2 span{}
```

(d) id选择符

```
#content{}
```

(e) class选择符

```
.he{}
```

(f) 标签指定式的选择符

如果想同时使用id和class，也想同时使用标签选择符

```
h1#content {} h1.p1 {}
```

(g) 组合选择符

```
h1 .p1,#content h1 {}
```

(2) css 哪些属性默认会继承，哪些不会继承？

不可继承的: display、margin、border、padding、background、height、min-height、max-height、width、min-width、max-width、overflow、position、left、right、top、bottom、z-index、float、clear、table-layout、vertical-align、page-break-after、page-break-before和unicode-bidi。

所有元素可继承: visibility和cursor。

内联元素可继承: letter-spacing、word-spacing、white-space、line-height、color、font、font-family、font-size、font-style、font-variant、font-weight、text-decoration、text-transform、direction。

终端块状元素可继承: text-indent和text-align。

列表元素可继承: list-style、list-style-type、list-style-position、list-style-image。

1. css优先级算法如何计算？

为了计算规则的特殊性，给每种选择器都分配一个数字值。然后，将规则的每个选择器的值加在一起，计算出规则的特殊性。可惜特殊性的计算不是以10为基数的，而是采用一个更高的未指定的基数。这能确保非常特殊的选择器（比如ID选择器）不会被大量一般选择器（比如类型选择器）所超越。但是，为了简化，如果在一个特定选择器中的选择器数量少于10个，那么可以以10为基数计算特殊性。

选择器的特殊性分成4个成分等级: a、b、c和d。

如果样式是行内样式，那么a=1。

b等于ID选择器的总数。

c等于类、伪类和属性选择器的数量。

d等于类型选择器和伪元素选择器的数量。

使用这些规则可以计算任何CSS选择器的特殊性。

选择器	特殊性	以10为基数的特殊性
<code>style=''</code>	1,0,0,0	1000
<code>#wrapper #content{}</code>	0,2,0,0	200
<code>#content .datePosted{}</code>	0,1,1,0	110
<code>div#content{}</code>	0,1,0,1	101
<code>#content{}</code>	0,1,0,0	100
<code>p.comment .dateposted{}</code>	0,0,2,1	21
<code>p.Comment</code>	0,0,1,1	11
<code>div p{}</code>	0,0,0,2	2
<code>p{}</code>	0,0,0,1	1

更多参考《CSS高级教程-[1]层叠、特殊性及继承.ppt》

1. 内联和!important哪个优先级高?

!important

1. 请描述一下cookies, sessionStorage 和 localStorage的区别?

参见: <http://devilswrwr.blog.163.com/blog/static/6583867020124822919358/>

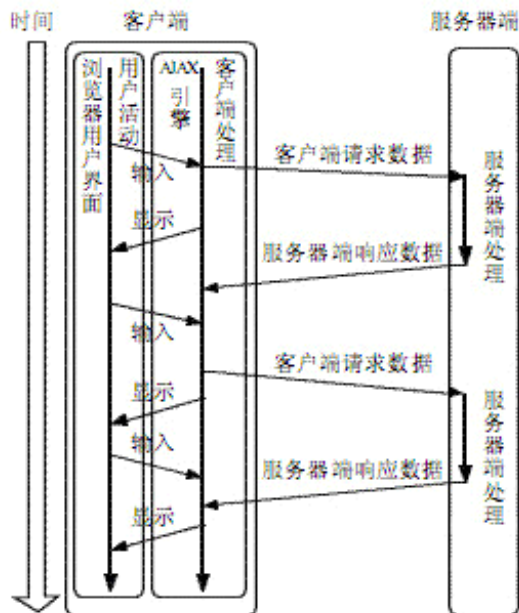
参考: 《sessionStorage、localStorage 和 cookie 之间的区别》

1. AJAX是什么? AJAX的交互模型(流程)? AJAX跨域的解决办法?

(1) AJAX是什么

Ajax是多种技术组合起来的一种浏览器和服务器交互技术,基本思想是允许一个互联网浏览器向一个远程页面/服务做异步的http调用,并且用收到的数据来更新一个当前web页面而不必刷新整个页面。

(2) AJAX的交互模型(流程)



(b) 基于 AJAX 的异步传输原理

同步：脚本会停留并等待服务器发送回复然后再继续

异步：脚本允许页面继续其进程并处理可能的回复

(3) AJAX跨域的解决办法？

跨域问题简单的理解就是因为JS同源策略的限制，a.com域名下的JS无法操作b.com或c.a.com下的对象，具体场景如下：

URL	说明	是否允许通信
http://www.kuqin.com/lab/a.js http://www.kuqin.com/script/b.js	同一域名下不同文件	允许
http://www.kuqin.com/a.js http://kuqin.com/b.js	同一域名下	允许
http://www.kuqin.com:8000/a.js http://www.kuqin.com/b.js	同一域名，不同端口	不允许
http://www.kuqin.com/a.js https://www.kuqin.com/b.js	同一域名，不同协议	不允许
http://www.kuqin.com/a.js http://70.32.92.74/b.js	域名和域名对应IP	不允许
http://www.kuqin.com/a.js http://script.kuqin.com/b.js	主域相同，子域不同	不允许
http://www.hao123.com/a.js http://www.kuqin.com/b.js	不同域名	不允许

前端对于跨域的解决办法：

- (a) document.domain+iframe
- (b) 动态创建script标签

6、编写一个方法，求一个字符串的字节长度？

```
new function(s)
{
    if(!arguments.length||!s) return null;
    if(""==s) return 0;
```

```

var l=0;
for(var i=0;i<s.length;i++)
{
    if(s.charCodeAt(i)>255) l+=2;
    else l++;
}
return l;
}("hello你好!");

```

7、如何控制alert中的换行?

```

alert("hello\nworld");

```

8、当点击按钮时，如何实现两个td的值互换?

```

<script>
function change(){
    var x,y
    x=document.getElementById("table1").rows[0].cells[0].innerHTML;
    y=document.getElementById("table1").rows[0].cells[1].innerHTML;
    document.getElementById("table1").rows[0].cells[0].innerHTML=y;
    document.getElementById("table1").rows[0].cells[1].innerHTML=x;
}
</script>
<table id="table1"><tr><td>第一个单元格</td><td>第二个单元格</td></tr>
</table>
<input type="button" value="改变" onclick="change()">

```

9、写一个简单form表单，当光标离开表单的时候表单的值发给后台?

```

<form action="index.php">
    <input type="text" name="txt" id="txt" value="abc" />
</form>
<script>
    window.onload = function(){
        var form = document.forms[0];
        var input = document.getElementById("txt");
        input.onblur = function(){
            form.submit();
        };
    };
</script>

```

10、如何获取表单<select>域的选择部分的文本?

```

<form action="index.php">
    <select name="" id="select">
        <option value="1">一</option>
        <option value="2">二</option>
        <option value="2">三</option>
    </select>

```

```

    </select>
</form>
<script>
    window.onload = function(){
        var select = document.getElementById("select");
        select.onchange = function(){
            console.log( this.options[ this.selectedIndex ].text );
        }
    };
</script>

```

11、在JavaScript中定时调用函数foo（）如何写？

方法一：

```
setInterval(foo,300);
```

方法二：

```

setInterval(function(){
    foo();
},300);

```

12、 table标签中border, cellpadding,td 标签中colspan , rowspan分别起什么作用？

table border:表格边框, table-cellpadding: 单元格填充

td colspan:单元格纵向合并, table-rowspan: 单元格横向合并

16、form中input可以设置readonly和disable, 请问这两项属性有什么区别？

从图上可以看出来，外观上有区别的。



disabled:

readonly:

disabled和readonly这两个属性有一些共同之处，比如都设为true，则form属性将不能被编辑，往往在写js代码的时候容易混合使用这两个属性，其实他们之间是有一定区别的：

如果一个输入项的disabled设为true，则该表单输入项不能获取焦点，用户的所有操作（鼠标点击和键盘输入等）对该输入项都无效，最重要的一点是当提交表单时，这个表单输入项将不会被提交。

而readonly只是针对文本输入框这类可以输入文本的输入项，如果设为true，用户只是不能编辑对应的文本，但是仍然可以聚焦焦点，并且在提交表单的时候，该输入项会作为form的一项提交。

17、JS中的三种弹出式消息提醒（警告窗口，确认窗口，信息输入窗口）的命令是什么？

```
alert(),confirm(),prompt()
```

18、实现三列布局，side左右两边宽度固定，main中同宽度自适应？

(1)、绝对定位法

这或许是三种方法里最直观，最容易理解的：左右两栏采用绝对定位，分别固定于页面的左右两侧，中间的主体栏用左右margin值撑开距离。于是实现了三栏自适应布局。

```
html,body{margin:0; height:100%;}
```

```
#left,#right{position:absolute; top:0; width:200px; height:100%;}
#left{left:0; background:#a0b3d6;}
#right{right:0; background:#a0b3d6;}
#main{margin:0 210px; background:#ffe6b8; height:100%;}
```

(2)、margin负值法

首先，中间的主体要使用双层标签。外层div宽度100%显示，并且浮动（本例左浮动，下面所述依次为基础），内层div为真正的主体内容，含有左右210像素的margin值。左栏与右栏都是采用margin负值定位的，左栏左浮动，margin-left为-100%，由于前面的div宽度100%与浏览器，所以这里的-100%margin值正好使左栏div定位到了页面的左侧；右侧栏也是左浮动，其margin-left也是负值，大小为其本身的宽度即200像素。

```
<div id="main">
    <div id="body"></div>
</div>
<div id="left"></div>
<div id="right"></div>
```

```
html,body{margin:0; height:100%;}
#main{width:100%; height:100%; float:left;}
#main #body{margin:0 210px; background:#ffe6b8; height:100%;}
#left,#right{width:200px; height:100%; float:left; background:#a0b3d6;}
#left{margin-left:-100%;}
#right{margin-left:-200px;}
```

(3) 自身浮动法

应用了标签浮动跟随的特性。左栏左浮动，右栏右浮动，主体直接放后面，就实现了自适应。

```
<div id="left"></div>
<div id="right"></div>
<div id="main"></div>
html,body{margin:0; height:100%;}
#main{height:100%; margin:0 210px; background:#ffe6b8;}
#left,#right{width:200px; height:100%; background:#a0b3d6;}
#left{float:left;}
#right{float:right;}
```

19、结果分别是什么？

```
function test(){
    tt = 'a';
    alert(tt);
    var tt = 'b';
    alert(tt);
}

test();

(function (){
    var tt = 'c';
    test();
})();//结果: a,b,a,b
```

21、描述一下冒泡，介绍件委托（事件代理）的实现方法，他的原理及优点？

22、给出下面a,b,c的输出结果

```
var a = parseInt("11",2)
var a = parseInt("02",10)
var a = parseInt("09/08/2009")
结果: 3,2,9
```

23、

```
function b(x , y , a ){
    arguments[2] = x;
    alert(a);
}
b(1,2,3);
结果: 1
```

26、分析程序执行结果

```
function b(){
    a=10;
    alert(arguments[1]);
}
b[1,2,3];
var a=10,
    b=20,
    c=10;
alert(a=b);
alert(a==b);
alert(a==c);
结果: 20, true, false
```

Web前端笔试题（十）

1、给定一个数组实现反转, 要求原地实现

```
function reversal( arr ){
    for (var i = 0; i < arr.length / 2; i++) {
        var temp = arr[i];
        arr[i] = arr[arr.length - i - 1];
        arr[arr.length - i - 1] = temp;
    }
}
var array = [1,"abc",3,4,5];
reversal(array);
alert(array);
```

2、如何判定iframe里面的资源都加载完毕

```

var iframe = document.createElement("iframe");
iframe.src = "http://www.sohu.com";
if (iframe.attachEvent){
    iframe.attachEvent("onload", function(){
        alert("Local iframe is now loaded.");
    });
} else {
    iframe.onload = function(){
        alert("Local iframe is now loaded.");
    };
}
document.body.appendChild(iframe);

```

3、 请写出两种以上FF有但IE没有的属性或函数

4、 简述IE、Firefox 的不同点 (css , javascript 等)

一、函数和方法差异

- (1) getYear() 方法
- (2) eval() 函数
- (3) const 声明

二、样式访问和设置

- (1) CSS的"float"属性
- (2) 访问<label>标签中的"for"
- (3) 访问和设置class属性
- (4) 对象宽高赋值问题

三、DOM方法及对象引用

- (1) getElementById
- (2) 集合类对象访问
- (3) frame的引用
- (4) parentElement
- (5) table操作
- (6) 移除节点removeNode() 和removeChild()
- (7) childNodes获取的节点
- (8) Firefox不能对innerText支持

四、事件处理

- (1) window.event
- (2) 键盘值的取得
- (3) 事件源的获取
- (4) 事件监听
- (5) 鼠标位置

五、其他差异的兼容处理

- (1) XMLHttpRequest
- (2) 模态和非模态窗口
- (3) input.type属性问题
- (4) 对select元素的option操作

- (5)img对象alt和title的解析
 - (6)img的src刷新问题
- 详细请参考《IE & FF 函数和方法差异.txt》

5、请写出下面javascript代码的运行结果

```
1. var a = 10;
   sayHi();
   function sayHi(){
       var a = 20;
       alert(a);
   }
   alert(a); //结果: 20,10
1. var a = 10;
   sayHi();
   function sayHi (){
       a = a + 10;
       alert(a);
       return a;
   }
   alert (a);
   alert( sayHi() + 10 );
   //结果: 20,20, 30,40
```

6、执行text()和new text()分别会有什么结果？

```
var a = 3;
function text(){
    var a = 0;
    alert(a);
    alert(this.a);
    var a;
    alert(a);
}
//结果: text()---0,3,0   new text()---0,undefined,0
```

7、在js中this关键字的使用场合和用法（如在构造函数中、setTimeout中等）

this是js的一个关键字，随着函数使用场合不同，this的值会发生变化。但是总有一个原则，那就是this指的是调用函数的那个对象。

this的引用规则：

- (1) 在最外层代码中，this引用的是全局对象。
- (2) 在函数内，this引用根据函数调用方式的不同而有所不同：

函数的调用方式	this应用的引用对象
构造函数调用	所生成的对象
方法调用	接收方对象
apply或是call调用	由apply或call的参数指定的对象
其他方式的调用	全局对象

(a) 纯粹函数调用

```

var x = 1;
function test() {
    alert(this.x);
}
test();//1
var x = 1;
function test() {
    this.x = 0;
}
test();
alert(x);//0

```

(b) 作为方法调用，那么this就是指这个上级对象。

```

function test() {
    alert(this.x);
}
var o = {};
o.x = 1;
o.m = test;
o.m(); //1

```

(c) 作为构造函数调用。所谓构造函数，就是生成一个新的对象。这时，这个this就是指这个对象。

```

function test() {
    this.x = 1;
}
var o = new test();
alert(o.x);//1

```

(d) apply调用

```

var x = 0;
function test() {
    alert(this.x);
}
var o = {};
o.x = 1;
o.m = test;
o.m.apply(); //0
o.m.apply(o); //1

```

(3) 通过点运算符或中括号运算符调用的对象的方法时，在运算符左侧所指定的对象

```

var obj = {
    x: 3,
    doit: function(){alert('method is called' + this.x);}
};
obj.doit(); //method is called 3
obj['doit'](); //method is called 3
-----
var obj = {
    x: 3,
    doit: function(){alert('method is called' + this.x);}
};
var fn = obj.doit;
fn(); //method is called undefined

```

```

var x = 5;
fn(); //method is called 5
var obj2 = {x:4,doit2:fn};
obj2.doit2(); //method is called 3
-----
var obj = {
  x: 3,
  doit: function(){alert('doit is called' + this.x); this.doit2(); },
  doit2: function(){alert('doit2 is called' + this.x); }
};
obj.doit();
//doit is called 3
//doit2 is called 3
(4)arguments对象的this
var foo = {
  bar: function () {
    return this.baz;
  },
  baz: 1
};
var a = (function () {
  return typeof arguments[0]();
})(foo.bar);
alert(a); //undefined,this指向了arguments

```

(5)js中setTimeout的this指向问题

```

function obj() {
  this.fn = function() {
    alert("ok");
    console.log(this);
    setTimeout(this.fn, 1000);//直接使用this引用当前对象
  }
}
var o = new obj();
o.fn();

```

解决方法一:

```

function obj() {
  this.fn = function() {
    alert("ok");
    console.log(this);
    setTimeout(this.fn.bind(this), 1000);//通过Function.prototype.bind 绑定当前对象
  }
}
var o = new obj();
o.fn();

```

解决方法二:

```

function obj() {
  this.fn = function() {
    alert("ok");
    setTimeout((function(a,b){

```

```

        return function(){
            b.call(a);
        }
    })(this,this.fn, 1000);//模拟Function.prototype.bind
}
var o = new obj();
o.fn();

```

解决方法三:

```

function obj() {
    this.fn = function() {
        var that = this;//保存当前对象this
        alert("ok");
        setTimeout(function(){
            that.fn();
        }, 1000);//通过闭包得到当前作用域，好访问保存好的对象that
    }
}
var o = new obj();
o.fn();

```

8、简述下cookie的操作，还有cookie的属性都知道那些？

(1) cookie操作：设置、获取及删除

```

function setCookie( name, value ){
    var date = new Date();
    date.setTime( date.getTime() + 30*24*60*60*1000 );
    document.cookie = name+'='+ escape(value) +';expires=' + date.toUTCString();
}
function getCookie( name ){
    var arr = document.cookie.match(new RegExp( "(^| )" + name + "=" + "([^;]*)" + "(;|$)" ));
    if( arr != null ) return unescape(arr[2]); return "";
}
function delCookie( name ){
    var date = new Date();
    date.setTime( date.getTime() - 1 );
    document.cookie = name + "=" + getCookie( name ) + ";expires=" + date.toUTCString();
}

```

(2) cookie的属性

- a) name名称
- b) value值
- c) expires 过期时间
- d) path路径
- e) domain域
- f) secure安全

详细参照《Cookies属性.txt》

9、DOM操作怎样添加、移除、替换、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()
```

(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  
getElementsByClassName() //通过元素class, 非浏览器兼容
```

10、口述：

- **如何提高前端性能，至少三点**

- (1) 尽可能的减少 HTTP 请求数
- (2) 使用CDN（内容分发网络）
- (3) 添加Expire/Cache-Control头
- (4) 启用Gzip压缩
- (5) 将css放在页面最上面
- (6) 将script放在页面最下面
- (7) 避免在CSS中使用Expressions
- (8) 把JavaScript和CSS都放到外部文件中
- (9) 减少DNS查询
- (10) 压缩 JavaScript 和 CSS
- (11) 避免重定向
- (12) 移除重复的脚本
- (13) 配置实体标签（ETag）
- (14) 使 AJAX 缓存
- (15) Yslow网站性能优化工具

- **当一个页面在你的电脑上没有任何问题，但在用户端有问题，你怎么办？**

如果是前端的问题，首先应考察用户的设备类型、屏幕分辨率及浏览器类型（版本），接着进行相同运行环境测试。

- **对于浏览器兼容性的看法，列举一些处理方法？**

- (1) CSS浏览兼容问题及解决方法
- (2) JS浏览器兼容问题及解决方法
- (3) HTML浏览器兼容问题及解决方法
- (4) navigator.userAgent

- **对响应式布局的看法？**

- (1) 什么是响应式布局

响应式布局是Ethan Marcotte在2010年5月份提出的一个概念，简而言之，就是一个网站能够兼容多个终端——而不是为每个终端做一个特定的版本。这个概念是为解决移动互联网浏览而诞生的。

响应式布局可以为不同终端的用户提供更加舒适的界面和更好的用户体验，而且随着目前大屏幕移动设备的普及，用大势所趋来形容也不为过。随着越来越多的设计师采用这个技术，我们不仅看到很多的创新，还看到了一些成形的模式。

（2）响应式布局的优缺点

优点：

面对不同分辨率设备灵活性强

能够快速解决多设备显示适应问题

缺点：

兼容各种设备工作量大，效率低下

代码累赘，会出现隐藏无用的元素，加载时间加长

其实这是一种折衷性质的设计解决方案，多方面因素影响而达不到最佳效果

一定程度上改变了网站原有的布局结构，会出现用户混淆的情况

（3）Media Query（媒介查询）

```
@media (min-device-width:1024px) and (max-width:989px), screen and (max-device-width:480px),
(max-device-width:480px) and (orientation:landscape), (min-device-width:480px) and
(max-device-width:1024px) and (orientation:portrait) {rules}
```

（4）viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

（5）弹性图片、弹性视频

```
width: 100%; max-width: 640px; 和height: auto
```

• 对HTML5和CSS3的看法，列举一些新属性？

JavaScript仍然称王 CSS3和HTML5在迅速崛起，浏览器兼容性问题越来越少，HTML5和JavaScript新功能给开发者带来不少烦恼，Web技术在移动领域的重要性愈加凸显。

html5：

- 新增标签和元素
- 表单与文件
- canvas
- video, audio
- web storage
- 离线应用程序
- web sockets
- web workers
- Geolocation API

CSS3：

- 选择器
- 文字和字体样式
- 盒相关样式
- 背景和边框
- transform
- transitions
- animations
- 多列布局与弹性盒布局
- Media Query

• 对DIV+CSS的看法，对语义化的看法？

DIV+CSS是WEB设计标准，它是一种网页的布局方法。与传统中通过表格（table）布局定位的方式不同，它可以实现网页页面内容与表现相分离。“DIV+CSS”其实是错误的叫法，而标准的叫法应是XHTML+CSS。因为DIV与Table都是XHTML或HTML语言中的一个标记，而CSS只是一种表现形式。

Div+css好处：

- 精简的代码，使用DIV+CSS布局，页面代码精简
- 提高访问速度、增加用户体验性
- div+css结构清晰，很容易被搜索引擎搜索到

11、分别使用2个3个5个div画出一个大的红十字

绝对定位

12、用JS实现一个选项卡效果，可以用任何你会的JS框架如jQuery等、

参见《tab.zip》

13、缩写下面样式：

代码一：{border-width:1px; border-color:#000; border-style:solide}
border:1px solid #000;

代码二：{background-position:0 0; background-repeat:no-repeat;
background-color:#f00;background-attachmend:fixed; background-images: url(background.gif) ;}

background:#f00 url(background.gif) no-repeat fixed 0 0;

代码三：{ font-style:italic; font-family:'Lucida Grande', sans-serif; font-size:1em;
font-weight:bold; font-variant:small-caps;line-height:140%; }

font:italic small-caps bold 1em/140% "Lucida Grande",sans-serif;

代码四：{list-style-position:inside; list-style-type:square; l
ist-style-image:url(image.gif);}

list-style:square inside url(image.gif);

代码五：{ margin-left:20px; margin-right:20px; margin-bottom:5px; margin-top:20px;}

margin:20px 20px 5px;

代码六：{color:#336699 ;color:#ffcc00;}

Color:#369;color:#fc0;

详细参见《常用CSS样式表缩写语法总结.txt》

1. 请使用css3的animation 创建一段动画，动画的过程为：100px*100px 的图片焦点，顺时针原地旋转3秒，然后开始水平滚动7秒，所有动画均为匀速，要求适配浏览器chrome和safari。

15、请简述如何通过css实现不同分辨率的手机适配，要求不可通过width=100%方式实现。

- (1) 弹性盒布局
- (2) Media Query (媒介查询)
- (3) viewport

(4) 弹性图片、弹性视频

16、关于数据的跨域问题请提出几个通过javascript实现的解决方案并附上简单代码。

- (1) document.domain+iframe的设置
- (2) 动态创建script
- (3) 利用iframe和location.hash
- (4) window.name实现的跨域数据传输
- (5) 使用HTML5 postMessage

详细参考《JavaScript跨域总结与解决办法.htm》

17、写一个方法完成身份证的验证

```
function checkID(text) {  
    var exp=new RegExp(/^[1-9]{15}|[1-9]{18}|[1-9]{17}[x|X]$/);  
    alert(exp.test(text));  
}
```

权威的身份证验证请参考《idCard.js》

18、请写出localStorage对象的常用方法

setItem(),getItem(),removeItem(),clear(),key()

19、js编程实现将10进制的数302转为二进制

```
parseInt(302,10).toString(2);
```

20、浏览器的缓存和本地存储相关内容有哪些？这些在什么环境下都各自能起什么作用？

(1) 浏览器的缓存

Expires、Cache-Control、Last-Modified、ETag是RFC2616（HTTP/1.1）协议中和网页缓存相关的几个字段。

前两个用来控制缓存的失效日期，浏览器可通过它来判定，需不需要发出HTTP请求；

后两个用来验证网页的有效性，服务器端利用它来验证这个文件是否需要重新返回

既然有了Last-Modified，为什么还要用ETag字段呢？因为如果在一秒钟之内对一个文件进行两次更改，Last-Modified就会不正确。

详细参考：《【总结】浏览器的缓存机制.html》<http://www.guojl.com/article/40/>

(2) 离线本地存储和传统的浏览器缓存有什么不同呢？

1)、浏览器缓存主要包含两类：

a.缓存协商：Last-modified, Etag

浏览器向服务器询问页面是否被修改过，如果没有修改就返回304，浏览器直接浏览本地缓存文件。否则服务器返回新内容。

b.彻底缓存：cache-control, Expires

通过Expires设置缓存失效时间，在失效之前不需要再跟服务器请求交互。

2)、离线存储为整个web提供服务，浏览器缓存只缓存单个页面；

3)、离线存储可以指定需要缓存的文件和哪些文件只能在线浏览，浏览器缓存无法指定；

4)、离线存储可以动态通知用户进行更新。

(3) 各自的作用

我们通常使用浏览器缓存在用户磁盘上存储web单页，在用户再次浏览的时候已节省带宽，但即便这样，依然无法在没有

Internet的情况下访问Web。为了让web应用程序在离线状态也能被访问。html5通过application cache API提供离线存储功能。前提是你需要访问的web页面至少被在线访问过一次。

详细参考: 《html5离线应用application cache》

http://www.silverlightchina.net/html/HTML_5/study/2011/1128/12118.html

Web前端笔试题 (十一)

- 请用代码写出<video>标签的使用方法

```
<!DOCTYPE html>
<html>
  <body>
    <div style="text-align:center">
      <button onclick="playPause()">
        播放/暂停
      </button>
      <button onclick="makeBig()">
        放大
      </button>
      <button onclick="makeSmall()">
        缩小
      </button>
      <button onclick="makeNormal()">
        普通
      </button>
      <br>
      <video id="video1" width="420">
        <source src="mov_bbb.mp4" type="video/mp4">
        <source src="mov_bbb.ogv" type="video/ogg">
        您的浏览器不支持 HTML5 video 标签。
      </video>
    </div>
    <script>
      var myVideo = document.getElementById("video1");
      function playPause() {
        if (myVideo.paused) myVideo.play();
        else myVideo.pause();
      }
      function makeBig() {
        myVideo.width = 560;
      }
      function makeSmall() {
        myVideo.width = 320;
      }
      function makeNormal() {
        myVideo.width = 420;
      }
    </script>
  </body>
```

</html>

- 请使用javascript创建video标签，并创建播放，暂停方法
- 请用HTML5标签写一个符合语义化的界面，页面中有导航栏，页眉、页脚，文字内容以及图片内容

```
<header>页眉</header>
<nav>
  <ul>
    <li>导航一</li>
    <li>导航二</li>
    <li>导航三</li>
    <li>导航四</li>
    <li>导航五</li>
  </ul>
</nav>
<article>
  <section>
    <p>文字内容一</p>
    
  </section>
  <section>
    <p>文字内容二</p>
    
  </section>
</article>
<footer>页脚</footer>
```

- 写出3个使用this的典型应用，哪些途径能改变this的取值。

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

```
var btn = document.getElementById("text");
btn.onclick = function() {
  alert(this.value); //此处的this是按钮元素
}
```

哪些途径能改变this的取值

(1) 闭包与this

```
var name = "The Window";
var object = {
  name : "My Object",
```

```

    getNameFunc : function(){
        return function(){
            return this.name;
        };
    }
};
alert(object.getNameFunc()()); //The Window

```

(2) arguments与this

```

Var
    foo={
bar:function(){
Return this.baz;
},
baz:1
};
var
a=(function(){
arguments.baz=34;
return typeof arguments[0]();
})(foo.bar);

alert(a);

```

(3) 表达式函数

```

var obj={};
obj.num=12;
obj.show=function (){
    alert(this.num);
};
(obj.show=obj.show)();

```

- 用过静态模板吗？那是什么模板或者是咋用的？

Jade: Jade是一款高性能简洁易懂的模板引擎, Jade是Haml的JavaScript实现在服务端 (NodeJS) 及客户端均有支持。

- data开头的这种属性有啥好处？

data-为前端开发者提供自定义的属性，这些属性集可以通过对象的dataset属性获取，不支持该属性的浏览器可以通过getAttribute方法获取。

```

<div data-author="david" data-time="2011-06-20" data-comment-num="10"
data-category="javascript"></div>
var post = document.getElementsByTagName('div')[0];
post.dataset; // DOMStringMap
post.dataset.commentNum; // 10

```

需要注意的是，data-之后的以连字符分割的多个单词组成的属性，获取的时候使用驼峰风格。
并不是所有的浏览器都支持.dataset属性，测试的浏览器中只有Chrome和Opera支持。

- HTTP协议的状态都有哪些？讲述你对HTTP缓存机制的了解，有没有CDN？

“100” : Continue（继续） 初始的请求已经接受，客户应当继续发送请求的其余部分。（HTTP 1.1新）

"101" : Switching Protocols (切换协议) 请求者已要求服务器切换协议, 服务器已确认并准备进行切换。(HTTP 1.1新)

"200" : OK (成功) 一切正常, 对GET和POST请求的应答文档跟在后面。

"201" : Created (已创建) 服务器已经创建了文档, Location头给出了它的URL。

"202" : Accepted (已接受) 服务器已接受了请求, 但尚未对其进行处理。

"203" : Non-Authoritative Information (非授权信息) 文档已经正常地返回, 但一些应答头可能不正确, 可能来自另一来源。(HTTP 1.1新)。

"204" : No Content (无内容) 未返回任何内容, 浏览器应该继续显示原来的文档。

"205" : Reset Content (重置内容) 没有新的内容, 但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容(HTTP 1.1新)。

"206" : Partial Content (部分内容) 服务器成功处理了部分 GET 请求。(HTTP 1.1新)

"300" : Multiple Choices (多种选择) 客户请求的文档可以在多个位置找到, 这些位置已经在返回的文档内列出。如果服务器要提出优先选择, 则应该在Location应答头指明。

"301" : Moved Permanently (永久移动) 请求的网页已被永久移动到新位置。服务器返回此响应(作为对 GET 或 HEAD 请求的响应)时, 会自动将请求者转到新位置。

"302" : Found (临时移动) 类似于301, 但新的URL应该被视为临时性的替代, 而不是永久性的。注意, 在HTTP1.0中对应的状态信息是"Moved Temporarily", 出现该状态代码时, 浏览器能够自动访问新的URL, 因此它是一个很有用的状态代码。注意这个状态代码有时候可以和301替换使用。例如, 如果浏览器错误地请求<http://host/~user> (缺少了后面的斜杠), 有的服务器返回301, 有的则返回302。严格地说, 我们只能假定只有当原来的请求是GET时浏览器才会自动重定向。请参见307。

"303" : See Other (查看其他位置) 类似于301/302, 不同之处在于, 如果原来的请求是POST, Location头指定的重定向目标文档应该通过GET提取(HTTP 1.1新)。

"304" : Not Modified (未修改) 自从上次请求后, 请求的网页未被修改过。原来缓冲的文档还可以继续使用, 不会返回网页内容。

"305" : Use Proxy (使用代理) 只能使用代理访问请求的网页。如果服务器返回此响应, 那么, 服务器还会指明请求者应当使用的代理。(HTTP 1.1新)

"307" : Temporary Redirect (临时重定向) 和 302 (Found) 相同。许多浏览器会错误地响应302应答进行重定向, 即使原来的请求是POST, 即使它实际上只能在POST请求的应答是303时才能重定向。由于这个原因, HTTP 1.1新增了307, 以便更加清除地区分几个状态代码: 当出现303应答时, 浏览器可以跟随重定向的GET和POST请求; 如果是307应答, 则浏览器只能跟随对GET请求的重定向。(HTTP 1.1新)

"400" : Bad Request (错误请求) 请求出现语法错误。

"401" : Unauthorized (未授权) 客户试图未经授权访问受密码保护的页面。应答中会包含一个WWW-Authenticate头, 浏览器据此显示用户名/密码对话框, 然后在填写合适的Authorization头后再次发出请求。

"403" : Forbidden (已禁止) 资源不可用。服务器理解客户的请求, 但拒绝处理它。通常由于服务器上文件或目录的权限设置导致。

"404" : Not Found (未找到) 无法找到指定位置的资源。

"405" : Method Not Allowed (方法禁用) 请求方法(GET、POST、HEAD、DELETE、PUT、TRACE等)禁用。(HTTP 1.1新)

"406" : Not Acceptable (不接受) 指定的资源已经找到, 但它的MIME类型和客户在Accept头中所指定的不兼容(HTTP 1.1新)。

"407" : Proxy Authentication Required (需要代理授权) 类似于401, 表示客户必须先经过代理服务器的授权。(HTTP 1.1新)

"408" : Request Time-out (请求超时) 服务器等候请求时超时。(HTTP 1.1新)

"409" : Conflict (冲突) 通常和PUT请求有关。由于请求和资源的当前状态相冲突, 因此请求不能成功。(HTTP 1.1新)

"410" : Gone (已删除) 如果请求的资源已被永久删除, 那么, 服务器会返回此响应。该代码与 404 (未找到) 代码类似, 但在资源以前有但现在已经不复存在的情况下, 有时会替代 404 代码出现。如果资源已被永久删除, 那么, 您应当使用 301 代码指定该资源的新位置。(HTTP 1.1新)

"411" : Length Required (需要有效长度) 不会接受包含无效内容长度标头字段的请求。(HTTP 1.1新)

"412" : Precondition Failed (未满足前提条件) 服务器未满足请求者在请求中设置的其中一个前提条件。(HTTP 1.1新)

"413" : Request Entity Too Large (请求实体过大) 请求实体过大, 已超出服务器的处理能力。如果服务器认为自己能够稍后再处理该请求, 则应该提供一个Retry-After头。(HTTP 1.1新)

"414" : Request-URI Too Large (请求的 URI 过长) 请求的 URI (通常为网址) 过长, 服务器无法进行处理。

"415" : Unsupported Media Type (不支持的媒体类型) 请求的格式不受请求页面的支持。

"416" : Requested range not satisfiable (请求范围不符合要求) 服务器不能满足客户在请求中指定的Range头。(HTTP 1.1新)

"417" : Expectation Failed (未满足期望值) 服务器未满足"期望"请求标头字段的要求。

"500" : Internal Server Error (服务器内部错误) 服务器遇到错误, 无法完成请求。

"501" : Not Implemented (尚未实施) 服务器不具备完成请求的功能。例如, 当服务器无法识别请求方法时, 服务器可能会返回此代码。

"502" : Bad Gateway (错误网关) 服务器作为网关或者代理时, 为了完成请求访问下一个服务器, 但该服务器返回了非法的应答。

"503" : Service Unavailable (服务不可用) 服务器由于维护或者负载过重未能应答。通常, 这只是一种暂时的状态。

"504" : Gateway Time-out (网关超时) 由作为代理或网关的服务器使用, 表示不能及时地从远程服务器获得应答。(HTTP 1.1新)

"505" : HTTP Version not supported (HTTP 版本不受支持) 不支持请求中所使用的 HTTP 协议版本。

- **test();结果分别是什么?为什么?**

```
var tt = 'a';
function test(){
    alert(tt);
    var tt = 'b';
    alert(tt);
}
test();
//undefined,b
```

- **以下结果是什么?为什么?**

```
var length=10;
function fn(){
    alert(this.length);
}
var obj={
    length:5,
    method: function(fn){
        fn();
        arguments[0]();
    }
}
obj.method(fn);
//10,1 第一次this指window对象, 第二次this指arguments
```

- **请写出下面javascript代码的运行结果!**

```
var a = 10;
sayHi();
function sayHi(){
    var a = 20;
    alert(a);
}
```

```

alert(a);
//20,10
-----
var a = 10;
(function(){
    alert(a);
    var a = 20;
})();
//undefined

```

- 请指出一下代码的性能问题，并进行优化。

```
var info = "四维图新是中国最大的数字地图生产商。"
```

```
info += "深交所股票代码：002405"。
```

```
info += "注重人才培养与开发，积极推动员工与企业共同发展。";
```

```
info += "公司现在中层管理者以及核心技术人员中，月60%为公司内部提拔晋升。";
```

```
info = info.split(",");
```

```
for(var i = 0; i < info.length; i++){
    alert(info[i]);
}
```

仔细观察info这个变量，发现它每次都要自加字符串，如果字符串很大的又很多的话会非常影响性能的。

对于js中的string类型，属于基本类型，因此一般情况下他们是存放在栈上的。如果字符串很大，info会每次变成一个很长的字符串，会很慢。

如果用引用类型数组来存放则好很多，如：

```

var temp = [];
temp.push("四维图新是中国最大的数字地图生产商。");
temp.push("深交所股票代码：002405");
temp.push("注重人才培养与开发，积极推动员工与企业共同发展。");
temp.push("公司现在中层管理者以及核心技术人员中，月60%为公司内部提拔晋升。");
temp.join("");
alert(temp);

```

- 请给出异步加载js方案，不少于两种。

默认情况javascript是同步加载的，也就是javascript的加载时阻塞的，后面的元素要等待javascript加载完毕后才能进行再加载，对于一些意义不是很大的javascript，如果放在页头会导致加载很慢的话，是会严重影响用户体验的。

异步加载方式：

(1) defer，只支持IE

(2) async：

(3) 创建script，插入到DOM中，加载完毕后callback，见代码：

```

function loadScript(url, callback){
    var script = document.createElement_x("script")
    script.type = "text/javascript";
    if (script.readyState){ //IE
        script.onreadystatechange = function(){
            if (script.readyState == "loaded" ||

```

```

        script.readyState == "complete"){
            script.onreadystatechange = null;
            callback();
        }
    };
} else { //Others: Firefox, Safari, Chrome, and Opera
    script.onload = function(){
        callback();
    };
}
script.src = url;
document.body.appendChild(script);
}

```

- **请设计一套方案，用于确保页面中JS加载完全。**

```

var n = document.createElement("script");
n.type = "text/javascript";
//以上省略部分代码
//ie支持script的readystatechange属性(IE support the readystatechange event for script and css
nodes)
if(ua.ie){
    n.onreadystatechange = function(){
        var rs = this.readyState;
        if('loaded' === rs || 'complete'===rs){
            n.onreadystatechange = null;
            f(id,url); //回调函数
        }
    };
//省略部分代码
//safari 3.x supports the load event for script nodes(DOM2)
    n.addEventListener('load',function(){
        f(id,url);
    });
//firefox and opera support onload(but not dom2 in ff) handlers for
//script nodes. opera, but no ff, support the onload event for link
//nodes.
}else{
    n.onload = function(){
        f(id,url);
    };
}

```

- **HTML5引入了应用程序缓存，如何启用应用程序缓存？**

(1) 应用程序缓存为应用带来三个优势：

离线浏览 - 用户可在应用离线时使用它们

速度 - 已缓存资源加载得更快

减少服务器负载 - 浏览器将只从服务器下载更新过或更改过的资源。

(2) 如何启用

1) Cache Manifest 基础

如需启用应用程序缓存，请在文档的 <html> 标签中包含 manifest 属性：

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

每个指定了 manifest 的页面在用户对其访问时都会被缓存。如果未指定 manifest 属性，则页面不会被缓存（除非在 manifest 文件中直接指定了该页面）。

manifest 文件的建议的文件扩展名是：".appcache"。

请注意，manifest 文件需要配置正确的 MIME-type，即 "text/cache-manifest"。必须在 web 服务器上进行配置。

2) Manifest 文件

manifest 文件是简单的文本文件，它告知浏览器被缓存的内容（以及不缓存的内容）。

manifest 文件可分为三个部分：

CACHE MANIFEST - 在此标题下列出的文件将在首次下载后进行缓存

NETWORK - 在此标题下列出的文件需要与服务器的连接，且不会被缓存

FALLBACK - 在此标题下列出的文件规定当页面无法访问时的回退页面（比如 404 页面）

CACHE MANIFEST

第一行，CACHE MANIFEST，是必需的：

CACHE MANIFEST

/theme.css

/logo.gif

/main.js

上面的 manifest 文件列出了三个资源：一个 CSS 文件，一个 GIF 图像，以及一个 JavaScript 文件。当 manifest 文件加载后，浏览器会从网站的根目录下载这三个文件。然后，无论用户何时与因特网断开连接，这些资源依然是可用的。

3) NETWORK

下面的 NETWORK 小节规定文件 "login.asp" 永远不会被缓存，且离线时是不可用的：

NETWORK:

login.asp

可以使用星号来指示所有其他资源/文件都需要因特网连接：

NETWORK:*

4) FALLBACK

下面的 FALLBACK 小节规定如果无法建立因特网连接，则用 "offline.html" 替代 /html5/ 目录中的所有文件：

FALLBACK:

/html5/ /404.html

注释：第一个 URI 是资源，第二个是替补。

5) 更新缓存

一旦应用被缓存，它就会保持缓存直到发生下列情况：

用户清空浏览器缓存

manifest 文件被修改（参阅下面的提示）

由程序来更新应用缓存

6) 实例 - 完整的 Manifest 文件

CACHE MANIFEST

2012-02-21 v1.0.0

/theme.css

/logo.gif

/main.js

NETWORK:

login.asp

FALLBACK:

/html5/ /404.html

重要的提示: 以 "##" 开头的是注释行, 但也可满足其他用途。应用的缓存会在其 manifest 文件更改时被更新。如果您编辑了一幅图片, 或者修改了一个 JavaScript 函数, 这些改变都不会被重新缓存。更新注释行中的日期和版本号是一种使浏览器重新缓存文件的办法。

- **什么是web worker? 创建一个简单的web worker实例。**

- **什么是 Web Worker?**

当在 HTML 页面中执行脚本时, 页面的状态是不可响应的, 直到脚本已完成。

web worker 是运行在后台的 JavaScript, 独立于其他脚本, 不会影响页面的性能。您可以继续做任何愿意做的事情: 点击、选取内容等等, 而此时 web worker 在后台运行。

- **web worker简单实例**

demo_workers.js

```
var i=0;

function timedCount(){
    i=i+1;
    postMessage(i);
    setTimeout("timedCount()",500);
}

timedCount();
<!DOCTYPE html>
<html>
<body>
<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>
<br /><br />
<script>
var w;
function startWorker(){
    if(typeof(Worker)!="undefined"){
        if(typeof(w)=="undefined") {
            w = new Worker("demo_workers.js");
        }
        w.onmessage = function (event) {
            document.getElementById("result").innerHTML = event.data;
        };
    }else{
        document.getElementById("result").innerHTML="Sorry, your browser does not support Web
Workers...";
    }
}

function stopWorker(){
    w.terminate();
}
</script>
</body>
</html>
```

- **请写出一个方法, 验证用户输入的是否是数字。**

```
function chk(str){
```

```

var pattern=/^\d{5,8}$/;
if(!pattern.test(str)){
    alert("请用5到8位数字");
}
}

```

- 如果你今年打算熟练掌握一项技术，那会是什么？

MEAN

- 在HTML5中如何使用XML？

通过Ajax, 代码如下:

```

function LoadXMLFile(xmlFile) {
    var xmlDoc = null;
    if (window.ActiveXObject) {
        xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
        //xmlDom.loadXML(xmlFile); //如果用的是XML字符串
        xmlDoc.load(xmlFile); //如果用的是xml文件。
    } else if (document.implementation && document.implementation.createDocument) {
        var xmlhttp = new window.XMLHttpRequest();
        xmlhttp.open("GET", xmlFile, false);
        xmlhttp.send(null);
        xmlDoc = xmlhttp.responseXML.documentElement; //一定要有根节点 (否则google浏览器读取不了)
    } else {
        xmlDoc = null;
    }
    return xmlDoc;
}

```

- 在HTML5的页面中可以使用XHTML的语法吗？

可以

- 判断以下两段代码的正误并阐述错误代码为何报错

<pre> sum(1); function sum(num){ alert(num); } </pre>	<pre> sum(1); var sum=function(){ alert(num); } </pre>
---	--

第二个报错: 函数声明提前

Web前端笔试题（十二）

- 请解释一下javascript的同源策略。

在JavaScript中，有一个很重要的安全性限制，被称为“Same-Origin Policy”（同源策略）。这一策略对于JavaScript代码能够访问的页面内容做了很重要的限制，即JavaScript只能访问与包含它的文档在同一域下的内容。

所谓同源是指，域名，协议，端口相同。

- 请解释一下事件代理？

JavaScript事件代理则是一种简单的技巧，通过它你可以把事件处理器添加到一个父级元素上，这样就避免了把事件处理器添加到多个子级元素上,提高性能。

当我们需要对很多元素添加事件的时候，可以通过将事件添加到它们的父节点而将事件委托给父节点来触发处理函数。这主要得益于浏览器的事件冒泡机制。

事件代理用到了两个在JavaScript事件中常被忽略的特性：事件冒泡以及目标元素。

```
function getEventTarget(e) {  
    e = e || window.event;  
    return e.target || e.srcElement;  
}
```

- 请描述判断一个数是否为超级素数的算法思路

超级素数：一个素数依次从低位去掉一位、两位。。。若所得的数依然是素数，如239就是超级素数。试求100~9999之内：超级素数的个数。

```
function prime() {  
    var i, j, n=0, t, f=0;  
    for(var t=100;t<10000;t++){  
        i=t;  
        f=1;  
        while(i!=0){  
            for(j=2;j<i;j++){  
                if(i%j==0) break;  
                if(j!=i){f=0;break;}  
            }  
            i/=10;  
        }  
        if(f==1){  
            console.log(t);  
            n++;  
        }  
    }  
    console.log(n);  
}  
prime()
```

- 请简述建设网站涉及的主要方面，当前开发使用的主流技术及趋势

(1) 建设网站涉及的主要方面

前端：

JS框架的选择，CSS框架选择，MVC框架选择，网站性能，网站安全性，网站用户体验及可用性，移动端，SEO等等；

后端：

开发平台，内容管理系统（CMS），网站安全性，带宽等等

(2) 开发使用的主流技术及趋势

JavaScript, jQuery, HTML5, CSS3, Zepto, iScroll, 微网页, HybridApp, ,Bootstrap, Less, PHP, MySQL, Ajax, JSON, AngularJS, NodeJS, Express, Jade, Socket.io, MongoDB, Mongoose

- 给出运行结果：

js中“5”+ 4 = ? //54

js中void(0) = ? //undefined

js中NaN * 4 = ? //NaN

[1,2] + [3,4] = ? //1,23,4

- 请写出下面JavaScript代码的运算结果是2还是undefined？请阐述原因

```
function show() {  
    var b = 1;  
    a = ++b;  
}  
show();  
alert(a);
```

2, 因为a是全局变量

- 简述 typeof 和 instanceof 的作用

typeof 是一个一元运算，放在一个运算数之前，运算数可以是任意类型。

它返回值是一个字符串，该字符串说明运算数的类型。typeof 一般只能返回如下几个结果：
number,boolean,string,function,object,undefined。

instanceof用于判断一个变量是否某个对象的实例

- 简述jQuery \$(document).ready()与window.onload的区别

jQuery中\$(document).ready()的作用类似于传统JavaScript中的window.onload方法，不过与window.onload方法还是有区别的。

(1).执行时间

window.onload必须等到页面内包括图片的所有元素加载完毕后才能执行。

\$(document).ready()是DOM结构绘制完毕后就执行，不必等到加载完毕。

(2).编写个数不同

window.onload不能同时编写多个，如果有多个window.onload方法，只会执行一个

\$(document).ready()可以同时编写多个，并且都可以得到执行

(3).简化写法

window.onload没有简化写法

\$(document).ready(function(){})可以简写成\$(function(){});

- 写出至少两种思路来实现轮播图片（无缝滚动）效果，建议用图文说明

（方法1）克隆图片组

（方法2）克隆首尾图片

（方法3）实时移动图片

- 请写出如下javascript代码的运行结果(10分)

```
var name = "zhangson";
function getName(){
    var arr = [123];
    var name = "lisi" + arr;
    document.write(name + "<br/>");
}
getName();
document.write(name);
lisi123
zhangson
```

- 如何在某个<textarea>中按下回车，不换行，而显示"<enter>"(注：不影响其它按键的正常输入)

```
var t = document.getElementsByTagName("textarea");
t[0].onkeyup = function(e){
    if(e.keyCode == 13){
        this.value = this.value.replace(/\n/, "");
        this.value += '<enter>';
    }
}
```

- };请写出如下javascript代码的运行结果

```
var my_arr=[];
for(var i=0;i<=5;i++){

    my_arr.push(i*(i+1)); //0, 2, 6, 12, 20, 30

}
var val=0;
while(val = my_arr.pop()){
    document.write(val+"");
}
//30,20,12,6,2
```

- 请列举css中position的参数和作用？

在CSS布局中，Position发挥着非常重要的作用，很多容器的定位是用Position来完成，CSS中Position属性有四个可选值，它们分别是：static、absolute、fixed、relative。

```
Position:static  无定位
Position:absolute 绝对定位
Position:fixed   相对于窗口的固定定位
Position:relative 相对定位
```

- 下面js中代码alert结果是多少？

```
var a =1;
function f(){
```

```

    alert(a);
    var a=2;
}
f();
//undefined

```

- 如何准确判断一个js对象是数组？

```

function isArray(o) {
    return Object.prototype.toString.call(o) === '[object Array]';
}

```

- jQuery中\$()都可以传哪些参数，分别实现什么功能？

- (1) \$(selector,[context]): 这个函数接收一个包含 CSS 选择器的字符串，然后用这个字符串去匹配一组元素
- (2) \$(html,[ownerDocument]): 根据提供的原始 HTML 标记字符串，动态创建由 jQuery 对象包装的 DOM 元素。同时设置一系列的属性、事件等。
- (3) \$(callback): 允许你绑定一个在DOM文档载入完成后执行的函数。

- php中，哪些地方用到"&"符号？

&这个是引用符号，他的作用是把一个变量指向另外一个变量，也就是说两个变量共用一块内存，不管操作那个变量都会影响另外一个变量。

```

$r = array();
$p = &$r;//此时$p和$r指向一块内存了
$p=array(0,1,2,3);//在这里操作$p和操作$r是一样的效果

```

- 作为一名前端开发工程师，你喜欢什么，最讨厌的又是什么？

参考答案：喜欢JavaScript，优雅的JS框架；讨厌兼容及适配的调试和DEBUG。

- js中给全部都是数字元素的数组排序的原生方法是（ ）其中使用的是()排序方法

```

sort(), 升序排序
[1,10,5].sort(function(x,y) {
    if(x>y) {return 1;}else{return -1;}
});

```

- js中调用某个函数之前，如何取得该函数最多可以传递多少个参数？该函数被调用时，如果知道传了多少个参数过来？

(1)假设函数名为fun，那个fun.length就是它最多能接受的参数个数；

(2)在fun函数里面，arguments就是用数组装着调用时传过来的所有参数，因此arguments.length就是已经传递过来的参数个数；

- 在窗口打开链接接的方法是（ ）

- 列举常用的浏览器类型及他们使用内核还有对应得调试工具

(1) 常用的浏览器类型及他们使用内核

- Trident (IE内核)
- Gecko (Firefox内核)
- Presto (Opera前内核, 已废弃, Opera现已改用Google Chrome的Blink内核)
- Webkit (Safari内核, Chrome内核原型, 开源)
- Blink (Google chrome的最新内核)

(2) 调试工具

- Chrome (Chrome Developer Tools)
- IE (开发者工具)
- 火狐 (firebug)
- Opera (<http://www.opera.com/dragonfly/>)

前端开发面试知识点

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 userData, 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 IE 7 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

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，就当它从来不存在。

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

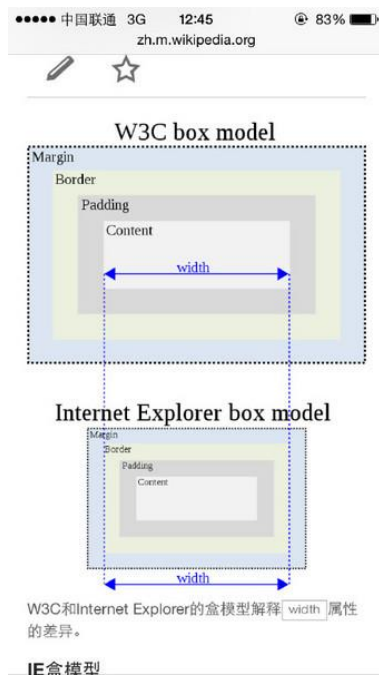
A：共同点：1.改变行内元素的呈现方式，display 被置为 block；2.让元素脱离普通流，不占据空间；3.默认会覆盖到非定位元素上

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 padding;
- 2) 盒模型：内容(content)、填充(padding)、边界(margin)、边框(border)。



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 生成绝对定位的元素，相对于浏览器窗口进行定位。（老 IE 不支持）

*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`。

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。

HTML 与 XHTML——二者有什么区别

区别 1.所有的标记都必须要有个相应的结束标记 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

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 浏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高宽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

```
getElementsByName()    //通过名称
```

`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风格:

* 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

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

*没有语义

如何实现浏览器内多个标签页之间的通信？

调用 localStorage、cookies 等本地存储方式

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

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

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。

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

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）。

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

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。

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。

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);
```

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'}
```

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

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](#)

document.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

.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](#)

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](#)

JavaScript 中的作用域与变量声明提升？

详见：[详解 JavaScript 函数模式](#)

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

其他问题？

你遇到过比较难的技术问题是？你是如何解决的？

常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？

列举 IE 与其他浏览器不一样的特性？

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）](#)

WEB 应用从服务器主动推送 Data 到客户端有那些方式？

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。

除了前端以外还了解什么其它技术么？你最最厉害的技能是什么？

你常用的开发工具是什么，为什么？

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。

你在现在的团队处于什么样的角色，起到了什么明显的作用？

你认为怎样才是全端工程师（ Full Stack developer ）？

介绍一个你最得意的作品吧？

项目中遇到什么问题？如何解决？

你的优点是什么？缺点是什么？

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](#))。

(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。

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>

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。

详情：[从输入 URL 到浏览器接收的过程中发生了什么事情？](#)

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

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

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

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](#)

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](#)

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。

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。

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

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

grunt，YUI compressor 和 google closure 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](#)

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

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起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。

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。

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

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 `firefox3.5` 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）。

闭包相关问题？

详情请见：[详解 js 闭包](#)

js 事件处理程序问题？

详情请见：[JavaScript 学习总结（九）事件详解](#)

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）。

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.removeEventListener) {

        element.removeEventListener(type, handler, false);

    } else if (element.detachEvent) {

        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;

    }

};
```

Node.js 的适用场景？

高并发、聊天、实时消息推送

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。

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。

WEB 应用从服务器主动推送 Data 到客户端有那些方式？

html5 websocket

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)

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`;

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](#)

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

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;

    }

}

```

AMD 和 CMD 规范的区别？

详情请见：[详解 JavaScript 模块化开发](#)

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

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>
```

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;  
}
```

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）。

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)客户端和服务器的时间不同步导致 `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。

前端开发面试知识点

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 userData，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 localStorage。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 IE 7 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

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，就当它从来不存在。

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

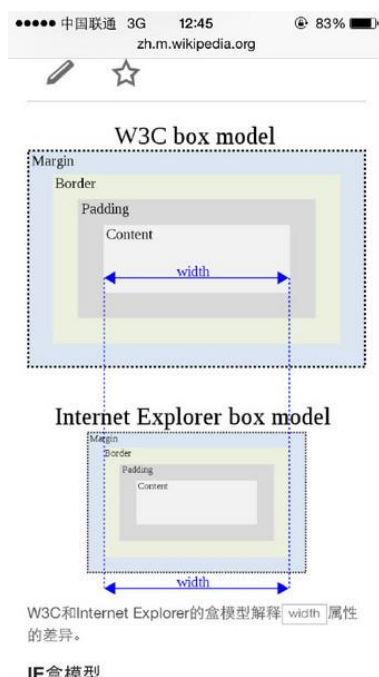
A : 共同点 : 1. 改变行内元素的呈现方式, `display` 被置为 `block`; 2. 让元素脱离普通流, 不占据空间; 3. 默认会覆盖到非定位元素上

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



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 生成绝对定位的元素，相对于浏览器窗口进行定位。（老 IE 不支持）

*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.

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`。

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。

HTML 与 XHTML——二者有什么区别

区别：1.所有的标记都必须要有个相应的结束标记 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用 "--" 8.图片必须有说明文字

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 浏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

getElementsByTagName()    //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风格：

* 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

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

如何实现浏览器内多个标签页之间的通信？

调用 localStorage、cookies 等本地存储方式

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

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。

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

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）。

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

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。

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。

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);
```

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'}
```

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`

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](#)

document.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

.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](#)

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](#)

JavaScript 中的作用域与变量声明提升？

详见：[详解 JavaScript 函数模式](#)

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

其他问题？

你遇到过比较难的技术问题是？你是如何解决的？

常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？

列举 IE 与其他浏览器不一样的特性？

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）](#)

WEB 应用从服务器主动推送 Data 到客户端有那些方式？

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。

除了前端以外还了解什么其它技术么？你最最厉害的技能是什么？

你常用的开发工具是什么，为什么？

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。

你在现在的团队处于什么样的角色，起到了什么明显的作用？

你认为怎样才是全端工程师（Full Stack developer）？

介绍一个你最得意的作品吧？

项目中遇到什么问题？如何解决？

你的优点是什么？缺点是什么？

如何管理前端团队？

最近在学什么？能谈谈你未来 3，5 年给自己的规划吗？

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](#)）。

（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。

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>

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。

详情：[从输入 URL 到浏览器接收的过程中发生了什么事情？](#)

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

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

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

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](#)

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](#)

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。

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。

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

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

grunt，YUI compressor 和 google closure 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](#)

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

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起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。

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。

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

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 `firefox3.5` 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）。

闭包相关问题？

详情请见：[详解 js 闭包](#)

js 事件处理程序问题？

详情请见：[JavaScript 学习总结（九）事件详解](#)

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）。

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.removeEventListener) {

        element.removeEventListener(type, handler, false);

    } else if (element.detachEvent) {

        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;  
  
    }  
  
};
```

Node.js 的适用场景？

高并发、聊天、实时消息推送

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。

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。

WEB 应用从服务器主动推送 Data 到客户端有那些方式？

html5 websocket

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)

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`；

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](#)

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

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;  
  
}  
  
}
```

AMD 和 CMD 规范的区别？

详情请见：[详解 JavaScript 模块化开发](#)

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

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>
```

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;

}

```

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）。

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)客户端和服务器的时间不同步导致 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。