1、web 前端优化机制？

1）内容上的优化：

1）尽量减少 HTTP 请求

有几种常见的方法能切实减少 HTTP 请求：

1）合并文件，比如把多个 CSS 文件合成一个；

2）CSS Sprites 利用 CSS background 相关元素进行背景图绝对定位；

3）图像地图

4）内联图象 使用 data: URL scheme 在实际的页面嵌入图像数据.

2）减少 DNS 查找

3）避免重定向：比如对 Web 站点子目录的后面添加个 /，就能有效避免一次重定向。 http://www.dbanotes.net/arch 与 http://www.dbanotes.net/arch/ 二者之间是有差异 的。如果是 Apache 服务器，通过配置 Alias 或 mod_rewrite 或 是 DirectorySlash 能消除

1）使得 Ajax 可缓存：响应时间对 Ajax 来说至关重要，否则用户体验绝对好不到哪里去。提高响应时间的有效手段 就 是 Cache 。其它的一些优化规则对这一条也是有效的。

5）延迟载入组件 (Post-load Components)

6）预载入组件 (Preload Components)

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

8）切分组件到多个域 (Split Components Across Domains).主要的目的是提高页面组件并行下载能力。但不要跨太多域名，否则就和第二条有些冲突了。

9）最小化 iframe 的数量 (Minimize the Number of iframes)

10）杜绝 http 404 错误

2）面向 Server 端：

1）使用 CDN (Use a Content Delivery Network)

2） 添加 Expires 或 Cache-Control 信息头

3）压缩内容 (Gzip Components)

4）设置 Etags (Configure ETags)

5）尽早刷新 Buffer (Flush the Buffer Early)

6）对 AJAX 请求使用 GET 方法 (Use GET for AJAX Requests)

XMLHttpRequest POST 要两步，而 GET 只需要一步。但要注意的是在 IE 上 GET 最大能处理 的 URL 长度是 2K。

3）面向 cookie

1）缩小 Cookie (Reduce Cookie Size)

根据 RFC 2109 的描述，每个客户端最多保持 300 个 Cookie，针对每个域名最多 20 个 Cookie （实 际上多数浏览器现在都比这个多，比如 Firefox 是 50 个），每个 Cookie 最多 4K，注 意这里的 4K 根据不同的浏览器可能不是严格的 4096 。别扯远了，对于 Cookie 最重要的就 是，尽量控制 Cookie 的大小，不要塞入一些无用的信息。

2）针对 Web 组件使用域名无关性的 Cookie (Use Cookie-free Domains for Components)

这个话题在此前针对 Web 图片服务器的讨论中曾经提及。这里说的 Web 组件(Component)，多 指静态文件，比如图片 CSS 等，Yahoo! 的静态文件都在 yimg.com 上，客户端请求静态文件的时候，减少了 Cookie 的反复传输对主域名 (yahoo.com) 的影响。

2、javascript 闭包？

Closure，所谓“闭包”，指的是一个拥有许多变量和绑定了这些变量的环境的表达式，因而这些变量也是该表达式的一部分。

简单来讲，ECMAScript 允许使用内部函数－－即函数定义和函数表达式位于另一个函数的函数体内。而且，这些内部函数可以访问它们所在的外部函数中声明的所有局部变量、参数和声明的其他内部函数。当其中一个这样的内部函数在包含它们的外部函数之外被调用时，就会形成闭包。也就是说，内部函数会在外部函数返回后被执行。而当这个内部函数执行时，它仍然必需访问其外部函数的局部变量、参数以及其他内部函数。这些局部变量、参数和函数声明（最初时）的值是外部函数返回时的值，但也会受到内部函数的影响。

闭包是通过在对一个函数调用的执行环境中返回一个函数对象构成的。比如，在对函数调用的过程中，将一个对内部函数对象的引用指定给另一个对象的属性。或者，直接将这样一个（内部）函数对象的引用指定给一个全局变量、或者一个全局性对象的属性，或者一个作为参数以引用方式传递给外部函数的对象。保护函数内的变量安全。以最开始的例子为例，函数 a 中 i 只有函数 b 才能访问，而无法通过其他途径访问到，因此保护了 i 的安全性。

闭包使用：

1）在内存中维持一个变量。依然如前例，由于闭包，函数 a 中 i 的一直存在于内存中，因此每次执行 c()，都会给 i 自加 1。

2）通过保护变量的安全实现 JS 私有属性和私有方法（不能被外部访问）

3）私有属性和方法在 Constructor 外是无法被访问的。

```
function exampleClosureForm(arg1, arg2){
```

```
    var localVar = 8;
    function exampleReturned(innerArg){
        return ((arg1 + arg2)/(innerArg + localVar));
    }
    /* 返回一个定义为 exampleReturned 的内部函数的引用 -:-
*/
    return exampleReturned;
}
var globalVar = exampleClosureForm(2, 4);
```

3、javascript 跑马灯

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="gb2312">
<title>RandomRunHouse</title>
<script language="javascript">
  msgArray = new Array(4);
  msgArray[0] = "欢迎光临";
  msgArray[1] = "白日依山静，黄河入海流";
  msgArray[2] = "人道是，三国周郎赤壁";
  msgArray[3] = "EmailTo: kof_he@163.com";

  var msg = "";
  var flag = true;
  var addOne = 1;
  var space = 50;
  var addTwo = 0;

  function randomMessage()
  {
     if ( flag == true ){  // 判断次句跑马灯是否跑完，跑完则
重新随机获得语句
           msg  =  msgArray[Math.floor(  Math.random()  *
msgArray.length )];
          flag = false;// 表示跑马灯已开始，下次调用此函数则
不在随机获得语句，直到跑马灯结束
   }
      if ( addOne < msg.length ){
```

```
            // 在状态栏中先一个字一个字递加的把 msg 字符串显示出来
            window.status = msg.substring( 0, addOne );
            addOne++;
        }else{
          var scroller = "";
            for ( i = 0; i < space; i++ ){// 添加状态栏前的空格
               scroller += " ";
            }
            scroller += msg;
            window.status = scroller.substring( addTwo, space + msg.length );
            addTwo++;
            // 表示跑马灯结束，初始化数据
            if ( addTwo > space + msg.length ){
               addOne = 0;
            addTwo = 0;
            flag = true;
            }
    }
    setTimeout( 'randomMessage()', 100 );
 }
 </script>
</head>
<body onLoad="randomMessage();">
</body>
</html>
```

16，javascript 浅复制与深度复制

浅复制(影子克隆):只复制对象的基本类型,对象类型仍属于原来的引用.

深复制(深度克隆):不仅复制对象的基本类,同时也复制原对象中的对象.就是说完全是新对象产生的.

例子：

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
   <TITLE> New Document </TITLE>
   <meta charset="gb2312">
```

```
   <META NAME="Generator" CONTENT="Wawa Editor 1.0">
   <META NAME="Author" CONTENT="八神奄">
   <META NAME="Keywords" CONTENT="Javascript, Java, XML,
net,C#,C++,Database">
   <META NAME="Description" CONTENT="不及格的程序员,无所不在
">
</HEAD>
<BODY>
   <SCRIPT LANGUAGE="JavaScript">
   <!--
   function Object.prototype.clone(){
      var newObj = new Object();
      for(elements in this){
         newObj[elements] = this[elements];
      }
      return newObj;
   }
   function Object.prototype.cloneAll(){
      function clonePrototype(){}
      clonePrototype.prototype = this;
      var obj = new clonePrototype();
      for(var ele in obj){
         if(typeof(obj[ele])=="object")      obj[ele]      =
obj[ele].cloneAll();
      }
      return obj;
   }
      var obj1       = new Object();
      obj1.Team         = "First";
      obj1.Powers       = new Array("Iori","Kyo");
      obj1.msg       = function(){alert()};
      obj1.winner       = new Object();
      obj1.winner.name  = obj1.Powers[0];
      obj1.winner.age      = 23;
      obj1.winner.from  = "Japan"
      var obj1_clone       = obj1.cloneAll();
      obj1_clone.Team         = "Second";
      obj1_clone.Powers    = new Array("Jimmy","Anndy");
```

```
        obj1_clone.winner.name  = obj1_clone.Powers[1];
        obj1_clone.winner.age   = 22;
        obj1_clone.winner.from  = "USA";
        msg  = "2003 界拳皇单打独斗杯,拳皇挑战赛:  \n\n   A 组
对战形式:\n\n"
        msg += obj1.Team+" 组 ,人员名单:"+obj1.Powers+"\n";
        msg += "第一轮过后,胜利者为:"+obj1.winner.name +" ,参
赛者年龄:"+obj1.winner.age+
           " ,来自岛国: "+obj1.winner.from+"\n";
        msg += "\n\n   B 组 对战形式:\n\n"
        msg   +=   obj1_clone.Team+"   组     ,   人   员   名
单:"+obj1_clone.Powers+"\n";
        msg += "第一轮过后,胜利者为:"+obj1_clone.winner.name +
           " ,参赛者年龄:"+obj1_clone.winner.age+" ,来自国际警
        察
           部队: "+obj1_clone.winner.from+"\n";
        alert(msg);
    //-->
    </SCRIPT>
</BODY>
</HTML>
```

深度 clone 方法 1：

```
object.prototype.clone=function(){
        if(typeOf(this)!='object'){
        return this;
    }
        if(this==null){
        return this;
    }
        var newObj=new Object();
        for(var i in this){
           newObj[i]=this[i].clone;
    }
    return(newObj);
}
```

深度 clone 方法 2：

```
function clone(obj){
    function clone(){};
```

```
    clone.prototype=obj;
    return new clone();
}
```

4、浏览器中用 javascript 如何取得用户系统当前的时间？

```
<script type='text/javascript'>
    var now=new Date();
    var hour=now.getHours();
    var min=now.getMinutes();
    var sec=now.getSeconds();
    document.write('当前时间是:'+hour+':'+min+':'+sec);
</script>
```

5、JavaScript 脚本为 Array 对象添加一个 indexOf 方法。

```
<script type='text/javascript'>
    Array.prototype.indexOf=function(item,i){
          i||i=0;
          var length=this.length;
          if(i<0){
          i=length+i;
       }
          for(;i<length;i++){
          if(this[i]==item){
             return(i);
          }
       }
          return(-1);
    }
</script>
```

6、写一个网页版的 comboBox,即可以下拉选择又可以文本框输入的组合文本框。

```
<input type='text' class='combobox' item='aaa,bbb,ccc,ddd' />
<script type='text/javascript'>
    var inputs=document.getElementsByTagName('input');
    for (var i=0;i<inputs.length;i++){
          if
(inputs[i].type=='text'&&inputs[i].className='combobox') {
          var input=inputs[i];
       }
    }
```

```
        var items=input.item.split(',');
        var select=document.createElement('select');
    for(i=0;i<items.length;i++){
        option=document.createElement('option') ;
            option.value=i;
            option.text=items[i];
            select.options.add(option);
    }
    select.style.position='absolute';
    select.style.width=input.clientWidth+20+'px';
    select.style.clip='rect(auto          auto          auto
'+input.clientWidth+'px)' ;
    select.onchange=function(){
          var input=this.nextSibling;
          input.text=this.text;
          input.select();
          input.focus();
    }
    var parent=input.parentNode;
    parent.insertBefore(select,input);
</script>
```

7、ActionScript 与 JavaScript 的交互?

ActionScript 提供了外部 API，那就是 **ExternalInterface 类**，通过 ExternalInterface 类可以实现 ActionScript 和 Flash Player 容器之间的直接通信。

1）在 JavaScript 中调用 ActionScript 方法

在 Flash Player 中，可以使用 HTML 页中的 JavaScript 来调用 ActionScript 函数。 ActionScript 函数可以返回一个值，JavaScript 会立即接收它作为该调用的返回值。

第一步:在 ActionScript 中调用 addCallback()把 ActionScript 注册为可从容器调用。 成功调用 addCallBack() 后，容器中的 JavaScript 代码可以调用在 Flash Player 中注册的 ActionScript 方法。

addCallBack 定义如下：

addCallback(functionName:String, closure:Function):void

functionName 参数就是在 Html 页面中脚本调用的方法名。closure 参数是要调用的本地方法，这个参数可以是一个方法也可以是对象实例。

```
<?xml version=”1.0″ encoding=”utf-8″?>
<mx:Application xmlns:mx=”http://www.adobe.com/2006/mxml”
layout=”absolute”
    initialize=”this.initApp()”>
   <mx:Script>
       <![CDATA[
        import mx.controls.Alert;
        import flash.external. ExternalInterface;
        public function sayHelloWorld(name:String):String {
            var msg:String=name+",hello world!";
             Alert.show(msg);
             return msg;
        }
        public function initApp():void {

ExternalInterface.addCallback("sayHelloWorld",sayHelloWorld)
;
        }
       ]]>
    </mx:Script>
</mx:Application>
```

第二步：那么在 Html 页面中，先获得 SWF 对象的引用，也就是用<object …/>声明的 Swf 的对象，比如说是 testJs。然后就可以用以下方式调用 ActionScript 中的方法。

```
function callActionScript() {

alert(document.getElementById(“testJs”).sayHelloWorld(“奎跃翔“));
}
```

2）在 ActionScript 中调用 JavaScript

在 ActionScript 中调用 JavaScript 最简单的方法是使用 ExternalInterface()，可以使用此 API 调用任意 JavaScript，传递参数，获得返回值，如果调用失败，ActionScript 抛出一个异常。

```
<?xml version=”1.0″ encoding=”utf-8″?>
<mx:Application xmlns:mx=”http://www.adobe.com/2006/mxml”
layout=”absolute“>
    <mx:Script>
       <![CDATA[
```

```
            import mx.controls.Alert;
            import flash.external. ExternalInterface;
            public function callJavaScript():void {

Alert.show(ExternalInterface.call("sayHelloWorld","奎 跃 翔
"));
          }
        ]]>
    </mx:Script>
    <mx:Button x=”290” y=”10” label=”Button” click=”
callJavaScript()” />
</mx:Application>
```

Html 页面中有如下函数定义：

```
function sayHelloWorld(name) {
    var msg=name+”,hello world!”
    alert(msg);
    return msg;
}
```

8、web 跨域访问解决方案

1）Web 代理的方式。即用户访问 A 网站时所产生的对 B 网站的跨域访问请求均提交到 A 网站的指定页面，由该页面代替用户页面完成交互，从而返回合适的结果。此方案可以解决现阶段所能够想到的多数跨域访问问题，但要求 A 网站提供 Web 代理的支持，因此 A 网站与 B 网站之间必须是紧密协作的，且每次交互过程，A 网站的服务器负担增加，且无法代用户保存 session 状态。

2）on-Demand 方式。MYMSN 的门户就用的这种方式，不过 MYMSN 中不涉及跨域访问问题。动态控制 script 标记的生成，通过修改 script 标记的 src 属性完成对跨域页面的调用。此方案存在的缺陷是，script 的 src 属性完成该调用时采取的方式时 get 方式，如果请求时传递的字符串过大时，可能会无法正常运行。不过此方案非常适合聚合类门户使用。

3）iframe 方式。查看过醒来在 javaeye 上的一篇关于跨域访问的帖子，他提到自己已经用 iframe 的方式解决了跨域访问问题。数据提交跟获取，采用 iframe 这种方式的确可以了，但由于父窗口与子窗口之间不能交互（跨域访问的情况下，这种交互被拒绝），因此无法完成对父窗口效果的影响。

4）用户本地转储方式：IE 本身依附于 windows 平台的特性为我们提供了一种基于 iframe，利用内存来“绕行”的方案，即两个 window

之间可以在客户端通过 windows 剪贴板的方式进行数据传输，只需要在接受数据的一方设置 Interval 进行轮询，获得结果后清除 Interval 即可。FF 的平台独立性决定了它不支持剪贴板这种方式，而以往版本的 FF 中存在的插件漏洞又被 fixed 了，所以 FF 无法通过内存来完成暗渡陈仓。而由于文件操作 FF 也没有提供支持（无法通过 Cookie 跨域完成数据传递），致使这种技巧性的方式只能在 IE 中使用。

9、js 中如何定义 class?

1）工厂方式

```
function Car(){
    var ocar = new Object;
    ocar.color = "blue";
    ocar.doors = 4;
    ocar.showColor = function(){
     document.write(this.color)
    };
    return ocar;
}
var car1 = Car();
var car2 = Car();
```

调用此函数时将创建新对象，并赋予它所有的属性和方法。使用此函数可以创建 2 个属性完全相同的对象。当然也可以通过给它传递参数来改版这种方式。

```
function Car(color,door){
    var ocar = new Object;
    ocar.color = color;
    ocar.doors = door;
    ocar.showColor = function(){
     document.write(this.color)
    };
    return ocar;
}
var car1 = Car("red",4);
var car2 = Car("blue",4);
car1.showColor()  //output:"red"
car2.showColor()  //output:"blue"
```

2）构造函数方式

```
function Car(color,door)...{
```

```
    this.color = color;
    this.doors = door;
    this.showColor = function()...{
     alert(this.color)
    };
}
var car1 = new Car("red",4);
var car2 = new Car("blue",4);
```

可以看到构造函数方式在函数内部没有创建对象，是用 this 关键字。因为在调用构造函数时已经创建了对象，而在函数内部只能用 this 来访问对象属性。现在用 new 来创建对象，看起来像那么回事了！但是它同工厂方式一样。每次调用都会为对象创建自己的方法。

3）原型方式

该方式利用了对象的 **prototype 属性**。首先用空函数创建类名，然后所有的属性和方法都被赋予 prototype 属性。

```
  function Car(){
  }
  Car.prototype.color = "red";
  Car.prototype.doors = 4;
  Car.prototype.showColor = function()...{
   alert(this.color);
  }
  var car1 = new Car();
  var car2 = new Car();
```

在这段代码中，首先定义了一个空函数，然后通过 prototype 属性来定义对象的属性。调用该函数时，原型的所有属性都会立即赋予要创建的对象，所有该函数的对象存放的都是指向 showColor()的指针，语法上看起来都属于同一个对象。 但是这个函数没有参数，不能通过传递参数来初始化属性，必须要在对象创建后才能改变属性的默认值。 原型方式有个很严重的问题就是当属性指向的是对象时，如数组。

```
  function Car(){
  }
  Car.prototype.color = "red";
  Car.prototype.doors = 4;
  Car.prototype.arr = new Array("a","b");
  Car.prototype.showColor = function(){
   alert(this.color);
```

```
}
var car1 = new Car();
var car2 = new Car();
car1.arr.push("cc");
alert(car1.arr);  //output:aa,bb,cc
alert(car2.arr);  //output:aa,bb,cc
```

这里由于数组的引用值，Car 的两个对象指向的都是同一个数组，所以当在 car1 添加值后，在 car2 中也可以看到。联合是用构造函数/原型方式就可以像其他程序设计语言一样创建对象，是用构造函数定义对象的非函数属性，用原型方式定义对象的方法。

```
function Car(color,door){
 this.color = color;
 this.doors = door;
 this.arr = new Array("aa","bb");
}
Car.prototype.showColor(){
 alert(this.color);
}
var car1 = new Car("red",4);
var car2 = new Car("blue",4);
car1.arr.push("cc");
alert(car1.arr);  //output:aa,bb,cc
alert(car2.arr);  //output:aa,bb
```

4）动态原型方式

动态原型的方式同混合的构造函数/原型方式原理相似。唯一的区别就是赋予对象方法的位置。

```
function Car(color,door){
   this.color = color;
   this.doors = door;
   this.arr = new Array("aa","bb");
   if(typeof Car._initialized == "undefined"){
    Car.prototype.showColor = function(){
     alert(this.color);
    };
    Car._initialized = true;
   }
}
```

动态原型方式是使用一个标志来判断是否已经给原型赋予了方法。

这样可以保证该方法只创建一次

5）混合工厂方式

它的目的师创建假构造函数，只返回另一种对象的新实例。

```
function Car(){
    var ocar = new Object();
    ocar.color = "red";
    ocar.doors = 4;
    ocar.showColor = function(){
     alert(this.color)
    };
    return ocar;
}
```

与工厂方式所不同的是，这种方式使用 new 运算符。

以上就是全部的创建对象方法。目前使用最广泛的就是混合构造函数/原型方式，此外，动态原型方式也很流行。在功能上与构造函数/原型方式等价。

10、javascript 判断浏览器类型？

1）方法一：

```
function CheckBrowser(){
    var cb = "Unknown";
    if(window.ActiveXObject){
        cb = "IE";
    }else
if(navigator.userAgent.toLowerCase().indexOf("firefox")  !=
-1){
        cb = "Firefox";
    }else if((typeof document.implementation != "undefined")
&&
     (typeof    document.implementation.createDocument    !=
"undefined") &&
    (typeof HTMLDocument != "undefined")){
        cb = "Mozilla";
    }else
if(navigator.userAgent.toLowerCase().indexOf("opera")    !=
-1){
        cb = "Opera";
    }
    return cb;
```

```
}
```
2）方法二：
```
if (window.XMLHttpRequest) { //Mozilla, Safari,...IE7
   alert('Mozilla, Safari,...IE7 ');
   if(!window.ActiveXObject){// Mozilla, Safari,...
          alert('Mozilla, Safari');
   } else {
          alert('IE7');
   }
}else{
       alert('IE6');
}
```
11、使用 javascript 校验登录名：只能输入 5-20 个以字母开头、可带数字、“_”、“.”的字串
```
Function isLegalUserName(s){
   Var patern = /^[a-zA-Z]{1}([a-zA-Z0-9] | [._]){4,19}$/;
   If(!patern.Exec(s))return false;
   Return true;
}
```
12、使用 javascript 校验用户姓名：只能输入 1-30 个以字母开头的字串,密码：只能输入 6-20 个字母、数字、下划线
```
Function isUserName(s){
   Var patern = /^[a-zA-Z]{1,30}$/;
   If(!patern.Exec(s))return false;
   Return true;
}
Function isPassword(s){
   Var patern = /^(\w){6,20}$/;
   If(!patern.Exec(s))return false;
   Return true;
}
```
13、使用 javascript 校验普通电话、传真号码：可以“+”开头，除数字外，可含有“-”
```
Function isTel(s){
   Var                                                patern=
/^[+]{0,1}(\d){1,3}[ ]?([-]?((\d)|[ ]){1,12})$/;
   If(!patern.Exec(s))return false;
   Return true;
```

```
}
```

14、正则表达式编程使用方式：

1）Java：java.util.Regex

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class GetParen1 {
  public static void main(String[] args) {
    Pattern patt = Pattern.compile("(\\w+)\\s(\\d+)");
    Matcher matcher = patt.matcher("Bananas 123");
    matcher.lookingAt();
    System.out.println("Name: " + matcher.group(1));
    System.out.println("Number: " + matcher.group(2));
  }
}
```

2）Javascript

```
Function isXXX(s){
   Var pattern=/^(\d){3}$/;
   If(!pattern.Exec(s))return false;
   Return true;
}
```

3）C++

```
#include "deelx.h"
int test_all_number(const char * string)
{
    static CRegexpT <char> regexp("\\d+");
    MatchResult result = regexp.MatchExact(string);
    return result.IsMatched();
}
int main(int argc, char * argv[])
{
    char * str1 = "12345";
    char * str2 = "12345 abcde";
    printf("'%s' => %s\n", str1, (test_all_number(str1) ?
"yes" : "no"));
    printf("'%s' => %s\n", str2, (test_all_number(str2) ?
"yes" : "no"));
    return 0;
}
```

15、javascript 的数组 sort 方法使用？

arrayObject.Sort()，表示按字母顺序排序。

arrayObject.sort(sortby)，其中 sortby 为自定义的比较函数的名字。

1）我们将创建一个数组，并按字母顺序进行排序：

```
<script type="text/javascript">
   var arr = new Array(6)
   arr[0] = "George"
   arr[1] = "John"
   arr[2] = "Thomas"
   arr[3] = "James"
   arr[4] = "Adrew"
   arr[5] = "Martin"
   document.write(arr + "<br />")
   document.write(arr.sort())
</script>
```

输出：

George,John,Thomas,James,Adrew,Martin

Adrew,George,James,John,Martin,Thomas

2）我们将创建一个数组，并按字母顺序进行排序：

```
<script type="text/javascript">
   var arr = new Array(6)
   arr[0] = "10"
   arr[1] = "5"
   arr[2] = "40"
   arr[3] = "25"
   arr[4] = "1000"
   arr[5] = "1"
   document.write(arr + "<br />")
   document.write(arr.sort())
</script>
```

输出：

10,5,40,25,1000,1

1,10,1000,25,40,5

请注意，上面的代码没有按照数值的大小对数字进行排序，要实现这一点，就必须使用一个排序函数：

```
<script type="text/javascript">
   function sortNumber(a,b)
```

```
    {
        return a - b
    }
    var arr = new Array(6)
    arr[0] = "10"
    arr[1] = "5"
    arr[2] = "40"
    arr[3] = "25"
    arr[4] = "1000"
    arr[5] = "1"
    document.write(arr + "<br />")
    document.write(arr.sort(sortNumber))
</script>
```

输出：

10,5,40,25,1000,1

1,5,10,25,40,1000

16、完成 foo()函数的内容，要求能够弹出对话框提示当前选中的是第几个单选框。

Body 里面的内容为：

```
<form   name="form1"   onsubmit="return   foo();">
<input   type="radio"   name="radioGroup"/>
<input   type="radio"   name="radioGroup"/>
<input   type="radio"   name="radioGroup"/>
<input   type="radio"   name="radioGroup"/>
<input   type="radio"   name="radioGroup"/>
<input   type="radio"   name="radioGroup"/>
<input   type="submit"/>
</form>
```

答案：

```
<script>
    function foo() {
        var                          rg                          =
    document.getElementsByName("radioGroup");
        for(var i=0;i<rg.length;i++) {
              if(rg[i].checked) {
                 alert("你选择了第" +(i+1)+ "个单选框");
              }
        }
```

```
        return   false;
    }
</script>
```

17、填充注释部分的函数体，使得 foo()函数调用弹出”成功”的对话框。代码应尽量简短。

```
<script type="application/javascript">
    function   foo(){
        var   str   =   reverse('a,b,c,d,e,f,g');
        alert(str);
        if(str   ==   'g,f,e,d,c,b,a')   alert('成功');
        else   alert('失败');
    }
    function   reverse(str){
        var  spl=str.split(",");
        var  newstr="";
        for(i=spl.length-1;i>=0;i--) {
            newstr+=spl[i]+',';
        }
        newstr=newstr.substring(0,newstr.length-1)
        return   newstr;
    }
</script>
```

18、用 javascript 编写一个方法 求一个字符串的字节长度。

```
function byteLength(s){
    var  clen= s.length;
    var  blen=0;
    for(var i=0;i<clen;i++){
        blen += isTwoBytes(s.charAt(i))
    }
    return blen;
}
function isTwoBytes(c){
    var patern= /^[\u4e00-\u9fa5]{1}$/;
    if(!patern.Exec(s))return 1;
    4eturn 2;
}
```

方法二：

```
    If(s.charCodeAt(i) >255) len+=2;
```

```
   Else len++;
```

19、javascript 日期格式化？

```
<script language="javascript" type="text/javascript">
   <!--
   /**
    * 对 Date 的扩展，将 Date 转化为指定格式的 String
    * 月(M)、日(d)、12 小时(h)、24 小时(H)、分(m)、秒(s)、周(E)、季度(q) 可以用 1-2 个占位符
    * 年(y)可以用 1-4 个占位符，毫秒(S)只能用 1 个占位符(是 1-3 位的数字)
    * eg:
    * (new Date()).pattern("yyyy-MM-dd hh:mm:ss.S") ==> 2006-07-02 08:09:04.423
    * (new Date()).pattern("yyyy-MM-dd E HH:mm:ss") ==> 2009-03-10 二 20:09:04
    * (new Date()).pattern("yyyy-MM-dd EE hh:mm:ss") ==> 2009-03-10 周二 08:09:04
    * (new Date()).pattern("yyyy-MM-dd EEE hh:mm:ss") ==> 2009-03-10 星期二 08:09:04
    * (new Date()).pattern("yyyy-M-d h:m:s.S") ==> 2006-7-2 8:9:4.18
    */
   Date.prototype.pattern=function(fmt) {
       var o = {
       "M+" : this.getMonth()+1, //月份
       "d+" : this.getDate(), //日
       "h+" : this.getHours()%12 == 0 ? 12 : this.getHours()%12, //小时
       "H+" : this.getHours(), //小时
       "m+" : this.getMinutes(), //分
       "s+" : this.getSeconds(), //秒
       "q+" : Math.floor((this.getMonth()+3)/3), //季度
       "S" : this.getMilliseconds() //毫秒
       };
       var week = {
       "0" : "\u65e5",
       "1" : "\u4e00",
       "2" : "\u4e8c",
```

```
        "3" : "\u4e09",
        "4" : "\u56db",
        "5" : "\u4e94",
        "6" : "\u516d"
        };
        if(/(y+)/.test(fmt)){
            fmt=fmt.replace(RegExp.$1,
           (this.getFullYear()+"").substr(4                    -
       RegExp.$1.length));
        }
        if(/(E+)/.test(fmt)){
            fmt=fmt.replace(RegExp.$1,
           ((RegExp.$1.length>1)   ?   (RegExp.$1.length>2   ?
       "\u661f\u671f" : "\u5468") :
         "")+week[this.getDay()+""]);
        }
        for(var k in o){
            if(new RegExp("("+ k +")").test(fmt)){
                fmt          =          fmt.replace(RegExp.$1,
(RegExp.$1.length==1) ? (o[k]) :
                (("00"+ o[k]).substr((""+ o[k]).length)));
            }
        }
        return fmt;
    }

    var date = new Date();
    window.alert(date.pattern("yyyy-MM-dd hh:mm:ss"));
    // -->
</script>
```

方法二：

```
<script language="javascript" type="text/javascript">
    new function(){
        with(new Date()){
              var t=function(a){return a<10?"0"+a:a;}
              alert(getFullYear()+" 年 "+t(getMonth()+1)+" 月
"+t(getDate())+"日"+
              t(getHours())+"   时   "+t(getMinutes())+"   分
```

```
            "+t(getSeconds())+"秒");
        }
    }
</script>
```

20、javascript 去掉 Array 中的重复元素？

```
<script language="javascript" type="text/javascript">
    Array.prototype.strip=function(){
        if(this.length<2) return [this[0]]||[];
        var arr=[];
        for(var i=0;i<this.length;i++){
            arr.push(this.splice(i--,1));
                for(var j=0;j<this.length;j++){
                if(this[j]==arr[arr.length-1]){
                    this.splice(j--,1);
                }
                }
        }
        return arr;
    }
    var arr=["abc",85,"abc",85,8,8,1,2,5,4,7,8];
    alert(arr.strip());
</script>
```

21、什么是 FOUC？如何避免？

Flash Of Unstyled Content：用户定义样式表加载之前浏览器使用默认样式显示文档，用户样式加载渲染之后再从新显示文档，造成页面闪烁。解决方法：把样式表放到文档的 head

如何创建块级格式化上下文（block formatting context）？有什么用

创建规则：

根元素

浮动元素（float 不是 none）

绝对定位元素（position 取值为 absolute 或 fixed）

display 取值为 inline-block,table-cell, table-caption,flex, inline-flex 之一的元素

overflow 不是 visible 的元素

作用：可以包含浮动元素、不被浮动元素覆盖、阻止父子元素的 margin 折叠

display, float, position 的关系

如果 display 为 none，那么 position 和 float 都不起作用，这种情况下

元素不产生框
否则，如果 position 值为 absolute 或者 fixed，框就是绝对定位的，float 的计算值为 none，display 根据下面的表格进行调整。
否则，如果 float 不是 none，框是浮动的，display 根据下表进行调整
否则，如果元素是根元素，display 根据下表进行调整
其他情况下 display 的值为指定值
总结起来：绝对定位、浮动、根元素都需要调整 display display 转换规则
外边距折叠（collapsing margins)
毗邻的两个或多个 margin 会合并成一个 margin，叫做外边距折叠。
规则如下：
两个或多个毗邻的普通流中的块元素垂直方向上的 margin 会折叠
浮动元素/inline-block 元素/绝对定位元素的 margin 不会和垂直方向上的其他元素的 margin 折叠
创建了块级格式化上下文的元素，不会和它的子元素发生 margin 折叠
元素自身的 margin-bottom 和 margin-top 相邻时也会折叠

22、如何确定一个元素的包含块（containing block)

根元素的包含块叫做初始包含块，在连续媒体中他的尺寸与 viewport 相同并且 anchored at the canvas origin；对于 paged media，它的尺寸等于 page area。初始包含块的 direction 属性与根元素相同。
position 为 relative 或者 static 的元素，它的包含块由最近的块级（display 为 block,list-item, table）祖先元素的内容框组成
如果元素 position 为 fixed。对于连续媒体，它的包含块为 viewport；对于 paged media，包含块为 page area
如果元素 position 为 absolute，它的包含块由祖先元素中最近一个 position 为 relative,absolute 或者 fixed 的元素产生，规则如下：
如果祖先元素为行内元素，the containing block is the bounding box around the padding boxes of the first and the last inline boxes generated for that element.
其他情况下包含块由祖先节点的 padding edge 组成
stacking context，布局规则
z 轴上的默认层叠顺序如下（从下到上）：
根元素的边界和背景
常规流中的元素按照 html 中顺序
浮动块
positioned 元素按照 html 中出现顺序

23、如何创建 stacking context：

根元素
z-index 不为 auto 的定位元素
a flex item with a z-index value other than 'auto'
opacity 小于 1 的元素
在移动端 webkit 和 chrome22+，z-index 为 auto，position: fixed 也将创建新的 stacking context

24、客户端存储 localStorage 和 sessionStorage

localStorage 有效期为永久，sessionStorage 有效期为顶层窗口关闭前
同源文档可以读取并修改 localStorage 数据，sessionStorage 只允许同一个窗口下的文档访问，如通过 iframe 引入的同源文档。
Storage 对象通常被当做普通 javascript 对象使用：通过设置属性来存取字符串值，也可以通过 setItem(key, value)设置，getItem(key)读取，removeItem(key)删除，clear()删除所有数据，length 表示已存储的数据项数目，key(index)返回对应索引的 key

```
localStorage.setItem('x', 1); // storge x->1
localStorage.getItem('x); // return value of x
//  枚举所有存储的键值对
for (var i = 0, len = localStorage.length; i < len; ++i ) {
        var name = localStorage.key(i);
        var value = localStorage.getItem(name);
}
localStorage.removeItem('x'); // remove x
localStorage.clear();    // remove all data
```

25、cookie 及其操作

cookie 是 web 浏览器存储的少量数据，最早设计为服务器端使用，作为 HTTP 协议的扩展实现。cookie 数据会自动在浏览器和服务器之间传输。
通过读写 cookie 检测是否支持
cookie 属性有名，值，有效期，作用域，secure；
cookie 默认有效期为浏览器会话，一旦用户关闭浏览器，数据就丢失，通过设置 max-age=seconds 属性告诉浏览器 cookie 有效期
cookie 作用域通过文档源和文档路径来确定，通过 path 和 domain 进行配置，web 页面同目录或子目录文档都可访问
通过 cookie 保存数据的方法为：为 document.cookie 设置一个符合目标的字符串如下
读取 document.cookie 获得'; '分隔的字符串，key=value,解析得到结果

```
document.cookie = 'name=qiu; max-age=9999; path=/; domain=domain; secure';
document.cookie = 'name=aaa; path=/; domain=domain; secure';
```

// 要改变 cookie 的值，需要使用相同的名字、路径和域，新的值
// 来设置 cookie，同样的方法可以用来改变有效期
// 设置 max-age 为 0 可以删除指定 cookie
//读取 cookie，访问 document.cookie 返回键值对组成的字符串，
//不同键值对之间用'; '分隔。通过解析获得需要的值
cookieUtil.js：自己写的 cookie 操作工具