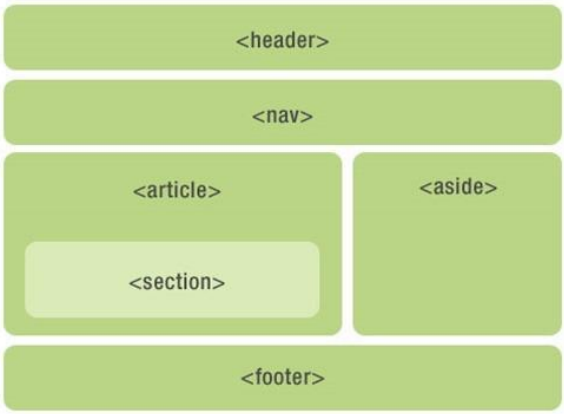
1、Img 的 title 和 alt 的异同
(1)、含义不同
alt：使用 alt 属性是为了给那些不能看到你文档中图像的浏览者提供文字说明，也就是图片显示不了的时候显示的文字。
title：图片正常显示时，鼠标悬停在图片上方显示的提示文字。

(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。
2、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
3、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
4、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">
5、如何使得文字不换行

6、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。
7、如果一个页面上需要多种语言文字，那么网页需要用什么编码格式
UTF-8
8、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。
9、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];//1
              arr[i] = arr[arr.length - i - 1];
              arr[arr.length - i - 1] = temp;
        }
    }
    var array = [1,"abc",3,4,5];
    reversal(array);
    alert(array);
```

10、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);
```

11、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

12、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

 2）var a = 10;
 sayHi();
 function sayHi (){
            a = a + 10;
            alert(a);
            return  a;
 }
 alert (a);
 alert( sayHi() + 10 );

 //结果：20,20，30,40
```

13、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);
}
New Text()
//结果：text()---0,3,0    new text()---0,undefined,0
```

14、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();
```

15、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

16、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
17、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
18、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。
19、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
20、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
能够快捷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） {srules}
（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
21、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
22、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
23、缩写下面样式：
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; list-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;
24、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

25、请写出 localstorage 对象的常用方法

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

26、js 编程实现将 10 进制的数 302 转为二进制

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

27、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。

28、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.ogg" 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>
```

29、请使用 javascript 创建 video 标签，并创建播放，暂停方法

见第 28 题

30、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。

31、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。

32、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 Temporatily”，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字/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 Accpet 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。

33、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
```

34、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
```

35、

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

36、

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

37、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);

38、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);
}
```

39、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);
   };
}
```

40、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。

41、什么是 web worker？创建一个简单的 web worker 实例。

(1) 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。

(2) 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>
```

42、在 HTML5 中如何使用 XML?

通过 Ajax,代码如下：

```
function LoadXMLFile(xmlFile) {
  var xmlDom = null;
  if (window.ActiveXObject) {
      xmlDom = new ActiveXObject("Microsoft.XMLDOM");
      //xmlDom.loadXML(xmlFile);//如果用的是 XML 字符串
      xmlDom.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);
      xmlDom = xmlhttp.responseXML.documentElement;//一定要有根节点
(否则 google 浏览器读取不了)
  } else {
      xmlDom = null;
  }
  return xmlDom;
```

43、判断以下两段代码的正误并阐述错误代码为何报错

```
    sum(1);
     sum(1);
    function sum(num){                                  var
    sum=function(){
```

```
            alert(num);
      alert(num);
     }
                                                                    }
```

第二个报错：函数声明提前

44、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
style= ‘’	1,0,0,0	1000
#wrapper #content{}	0,2,0,0	200
#content .datePosted{}	0,1,1,0	110
div#content{}	0,1,0,1	101
#content{}	0,1,0,0	100
p.comment .dateposted{}	0,0,2,1	21
p.Comment	0,0,1,1	11
div p{}	0,0,0,2	2
p{}	0,0,0,1	1

45、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!");
```

46、如何控制 alert 中的换行？

```
alert(“hello\nworld”);
```

47、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()">
```

48、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>
```

49、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>
</form>
<script>
      window.onload = function(){
            var select = document.getElementById("select");
            select.onchange = function(){

      console.log( this.options[ this.selectedIndex ].text );
                  }
            };
</script>
```

50、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);
```

51、 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
52、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()
53、实现三列布局，side 左右两边宽度固定，main 中同宽度自适度？
(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;}
```

54、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
```

55、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

57、

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

58、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

59、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;
          }
  }
```

60、列出 ie 和火狐事件对象的不同点

```
Var getEvent = function( event ){
   return event ? event : window.event;
}
```

61、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。

62、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。

63、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

64、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() );
```

65、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。

66、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.

67、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，而后一个是会覆盖前一个的，这个时候当然要指定同步方式：flase。

68、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。
69、请举例出一个匿名函数的典型案例
```
;(function($)  {
    //  Code  goes  here
})(jQuery);
```
70、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。
71、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 JavaSciprt 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;
}
```
72、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。。。若所得的书已然是素数，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()
```
73、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
74、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
```
75、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
76、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
77、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 $(function(){});
78、写出至少两种思路来实现轮播图片（无缝滚动）效果，建议用图文说明
（方法 1）克隆图片组
（方法 2）克隆首尾图片
（方法 3）实时移动图片
79、请写出如下 javascript 代码的运行结果(10 分)
```
var name = "zhangson";
function getName(){
      var arr = [123];
      var name = "lisi" + arr;
      document.write(name + "<br/>");
}
getName();
document.write(name);
```
```
lisi123
zhangson
```
80、如何使在某个<textarea>中按下回车，不换行，而显示“<enter>”(注：不影响其它按键的正常输入)
```
var t = document.getElementsByTagName("textarea");
t[0].onkeyup = function(e){
 if(e.keyCode == 13){
      this.value = this.value.replace(/\n/,"");
      this.value += '<enter>';
 }
}
```
81、请写出如下 javascript 代码的运行结果
```
var my_arr=[];
for(var i=0;i<=5;i++){
      my_arr.push(i*(i+1)); //0, 2, 6, 12, 20, 30
}
var val=0;
while(val = my_arr.pop()){
      document.write(val+" ");
}
//30,20,12,6,2
```
82、下面 js 中代码 alert 结果是多少？
```
var a =1;
function f(){
      alert(a);
      var a=2;
}
f();
```
//undefined
83、如何准确判断一个 js 对象是数组？
```
function  isArray(o)  {
    return  Object.prototype.toString.call(o)  ===  '[object Array]';
}
```
84、jQuery 中$()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。
85、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
```

86、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;}
    });
```

87、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；

88、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;
```

88、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);
        }
```

89、完成下列 JS 面向对象案例

（1） 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

（2） 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。

（3） 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() );*/
```

90、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);
    }
});
```

91、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();*/
```

92、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;
}
```

93、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);
         }
    }
}
```

94、编写 js 函数，用于计算 a+(a-1)+(a-2)+….+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));
```

95、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") );
```

96、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]

97、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);
```

98、写出 jQuery 事件绑定方法（至少三种）

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

99、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

100、写出选择选择器~、>、+、“ “对应的四个方法

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

101、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})
```

102、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');
```

103、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');
  }
}
```

104、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");
```

105、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/>";
}
```

106、写出 SQL 语句的格式 : 插入 , 更新 , 删除

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

表名：User

表字段和数据如下：

(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='张三'
```

107、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') );
```

108、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>
```

109、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");
```

110、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(vari = 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);
}
```

112、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 + "( |$)" );
            var aEle = this.getEleByTag( tag, parent );
            for( var i = 0, len = aEle.length; i < len; i++ ){
                reg.test(    aEle[i].className    )    && aClass.push( aEle[i] );
            }
            return aClass;
        }
    },
    getEleByTag: function( ele, obj ){
        return ( obj || document ).getElementsByTagName( ele );
    },
    hasClass: function( ele, cls ){
        return ele.className.match( new RegExp( "(^|\\s)" + cls + "(\\s|$)" ));
    },
    addClass: function( ele, cls ){
        if( !$.hasClass( ele, cls ) ){
            ele.className += " " + cls;
        }
    },
    removeClass: function( ele, cls ){
        if( $.hasClass( ele, cls ) ){
            var reg = new RegExp( "(^|\\s)" + cls + "(\\s|$)" );
            ele.className = ele.className.replace( reg, " " );
        }
    }
};
```

113、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"));*/
```

114、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+="<tr><td>"+value.cid+"</td><td>"+value.oname+"</td></tr>"
            });
        }

     });
     $('table').html(str);
```

115、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(typeof(needle) =='string'||typeof(needle) =='number') {
            for(var i in haystack) {
                if(haystack[i] == needle) {
                    Return true;
                }
            }
```

```
        }
      return false;
}
```

116、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()
```

117、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;
          }
     }
```

118、判断 input 中用户输入的字符串中是否含有”shianyun”

```
/shianyun/i.test()
```

119、显示 input 中用户输入的字符串中含有几个 a（不区分大小写）

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

120、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');
     }
}
```

121、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()
```

122、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>
```

123、请用 javascript 找出所有 ClassName 包含 text 的标签<li>，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 +
"\\b"))!=-1){//用正则表达式来判断是不是包含此类名
          eles[i].style.backgroundColor="yellow";
          }
}
```

124、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 第七只鞋是高跟并且有鞋带

125、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

126、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) ;
          }
}
```

127、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;
}
```

128、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)
}
```

129、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.randow()*o+start);}
        var arr=new array();
        for(var i=0;i<10;i++){
                arr.push(select(10,100));
        }
alert(arr);
```

130、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();
```

知识点

1、JavaScript 跨域总结与解决办法

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
6、利用 flash

2、cssHack

CSS Hack 大致有 3 种表现形式,CSS 类内部 Hack、选择器 Hack 以及 HTML 头部引用(if IE)Hack，CSS Hack 主要针对类内部 Hack：比如 IE6 能识别下划线"_"和星号"*",IE7 能识别星号"*",但不能识别下划线"_",而 firefox 两个都不能认识。

" #demo {width:100px;} "为例;
#demo {width:100px;} /*被 FIREFOX,IE6,IE7 执行.*/
* html #demo {width:120px;} /*会被 IE6 执行,之前的定义会被后来的覆盖,所以#demo 的宽度在 IE6 就为 120px; */
*+html #demo {width:130px;} /*会被 IE7 执行*/
所以最后,#demo 的宽度在三个浏览器的解释为: FIREFOX:100px; ie6:120px; ie7:130px;
IE8 最新 css hack：
"\9" 例:"border:1px \9;".这里的"\9"可以区别所有 IE 和 FireFox.
"\0" IE8 识别，IE6、IE7 不能.
"*" IE6、IE7 可以识别.IE8、FireFox 不能.
"_" IE6 可以识别"_",IE7、IE8、FireFox 不能.

3、CSS Sprites 的概念、原理、适用范围和优缺点

概念

CSSSprites 在国内很多人叫 css 精灵，是一种网页图片应用处理方式。它允许你将一个页面涉及到的所有零星图片都包含到一张大图中去，这样一来，当访问该页面时，载入的图片就不会像以前那样一幅一幅地慢慢显示出来了。

原理

在需要用到图片的时候，现阶段是通过 CSS 属性 background-image 组合 background-repeat, background-position 等来实现

适用范围：

1，需要通过降低 http 请求数完成网页加速。

2，网页中含有大量小图标。或者，某些图标通用性很强。

3，网页中有需要预载的图片。主要是 a 与 a:hover 背景图这种关系的。如果 a 与 a:hover 的背景图分别加载，那么，就会出现用户鼠标移到某个按钮上， 按钮的背景突然消失再出来，产生“闪烁”，如果按钮文字色与大背景相同或相近，就更囧了，有可能让人产生按钮“消失”了的错觉。

优点

我们从前面了解到，CSS Sprites 为什么突然跑火，跟能够提升网站性能有关。显而易见，这是它的巨大优点之一。普通制作方式下的大量图片，现在合并成一个图片，大大减少了 HTTP 的连接数。HTTP 连接数对网站的加载性能有重要影响。

缺点

至于可维护性，这是一般双刃剑。可能有人喜欢，有人不喜欢，因为每次的图片改动都得往这个图片删除或添加内容，显得稍微繁琐。而且算图片的位置（尤其是这种上千 px 的图）也是一件颇为不爽的事情。当然，在性能的口号下，这些都是可以克服的。

由于图片的位置需要固定为某个绝对数值，这就失去了诸如 center 之类的灵活性。

前 面我们也提到了，必须限制盒子的大小才能使用 CSS Sprites，否则可能会出现出现干扰图片的情况。这就是说，在一些需要非单向的平铺背景和需要网页缩放的情况下，CSS Sprites 并不合适。YUI 的解决方式是，加大图片之间的距离，这样可以保持有限度的缩放。

4、CSS3 Media Query 实现响应布局

一、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。

二、响应式布局有哪些优点和缺点？

优点：

面对不同分辨率设备灵活性强

能够快捷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

三、响应式布局该怎么设计？

CSS 中的 Media Query（媒介查询）是什么？

通过不同的媒体类型和条件定义样式表规则。媒体查询让 CSS 可以更精确作用于 不同的媒体类型和同一媒体的不同条件。媒体查询的大部分媒体特性都接受 min 和 max 用于表达”大于或等于”和”小与或等于”。如：width 会有 min-width 和 max-width 媒体查询可以被用在 CSS 中的@media 和@import 规则上，也可以被用在 HTML 和 XML 中。通过这个标 签属性，我们可以很方便的在不同的设备下实现丰富的界面，特别是移动设备，将会运用更加的广泛。

media query 能够获取哪些值？

设备的宽和高 device-width，device-heigth 显示屏幕/触觉设备。

渲染窗口的宽和高 width，heigth 显示屏幕/触觉设备。

设备的手持方向，横向还是竖向 orientation（portrait|lanscape）和打印机等。

画面比例 aspect-ratio 点阵打印机等。

设备比例 device-aspect-ratio-点阵打印机等。

对象颜色或颜色列表 color，color-index 显示屏幕。

设备的分辨率 resolution。

语法结构及用法

在 link 中使用@media：

<link rel=“stylesheet” type=“text/css” media=“only screen and（max-width：480px），only screen and（max-device-width：480px）” href=“link.css”/>

在样式表中内嵌@media：

@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） {srules}

四、css 清除浮动大全，共 8 种方法

1，父级 div 定义 height

原理：父级 div 手动定义 height，就解决了父级 div 无法自动获取到高度的问题。

优点：简单，代码少，容易掌握

缺点：只适合高度固定的布局，要给出精确的高度，如果高度和父级 div 不一样时，会产生问题

建议：不推荐使用，只建议高度固定的布局时使用

2、结尾处加空 div 标签 clear:both

原理：添加一个空 div，利用 css 提高的 clear:both 清除浮动，让父级 div 能自动获取到高度

优点：简单，代码少，浏览器支持好，不容易出现怪问题

缺点：不少初学者不理解原理；如果页面浮动布局多，就要增加很多空 div，让人感觉很不爽

建议：不推荐使用，但此方法是以前主要使用的一种清除浮动方法

3、父级 div 定义 伪类:after 和 zoom

/*清除浮动代码*/

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

.clearfloat{zoom:1}

原理：IE8 以上和非 IE 浏览器才支持:after，原理和方法 2 有点类似，zoom(IE 转有属性)可解决 ie6,ie7 浮动问题

优点：浏览器支持好，不容易出现怪问题（目前：大型网站都有使用，如：腾迅，网易，新浪等等）

缺点：代码多，不少初学者不理解原理，要两句代码结合使用，才能让主流浏览器都支持。

建议：推荐使用，建议定义公共类，以减少 CSS 代码。

4、父级 div 定义 overflow:hidden

原理：必须定义 width 或 zoom:1，同时不能定义 height，使用 overflow:hidden 时，浏览器会自动检查浮动区域的高度

优点：简单，代码少，浏览器支持好

缺点：不能和 position 配合使用，因为超出的尺寸的会被隐藏。

建议：只推荐没有使用 position 或对 overflow:hidden 理解比较深的朋友使用。

5，父级 div 定义 overflow:auto

原理：必须定义 width 或 zoom:1，同时不能定义 height，使用 overflow:auto 时，浏览器会自动检查浮动区域的高度

优点：简单，代码少，浏览器支持好

缺点：内部宽高超过父级 div 时，会出现滚动条。

建议：不推荐使用，如果你需要出现滚动条或者确保你的代码不会出现滚动条就使用吧。

6，父级 div 也一起浮动

原理：所有代码一起浮动，就变成了一个整体

优点：没有优点

缺点：会产生新的浮动问题。

建议：不推荐使用，只作了解。

7，父级 div 定义 display:table

原理：将 div 属性变成表格

优点：没有优点

缺点：会产生新的未知问题。

建议：不推荐使用，只作了解。

评分：★☆☆☆☆

8，结尾处加 br 标签 clear:both

原理：父级 div 定义 zoom:1 来解决 IE 浮动问题，结尾处加 br 标签 clear:both

建议：不推荐使用，只作了解。

五、<u>div+CSS 浏览器兼容问题整理</u>

1.div 的垂直居中问题

vertical-align:middle; 将行距增加到和整个 DIV 一样高 line-height:200px; 然后插入文字，就垂直居中了。缺点是要控制内容不要换行

2. margin 加倍的问题

设置为 float 的 div 在 ie 下设置的 margin 会加倍。这是一个 ie6 都存在的 bug。解决方案是在这个 div 里面加上 display:inline;

3.浮动 ie 产生的双倍距离

#box{ float:left; width:100px; margin:0 0 0 100px; //这种情况之下 IE 会产生 200px 的距离 display:inline; //使浮动忽略}

这里细说一下 block 与 inline 两个元素：block 元素的特点是,总是在新行上开始,高度,宽度,行高,边距都可以控制(块元素);Inline 元素的特点是,和其他元素在同一行上,不可控制(内嵌元素);

#box{ display:block; //可以为内嵌元素模拟为块元素

display:inline; //实现同一行排列的效果 diplay:table;

4 IE 与宽度和高度的问题

IE 不认得 min-这个定义，但实际上它把正常的 width 和 height 当作有 min 的情况来使。这样问题就大了，如果只用宽度和高度，正常的浏览器里这两 个值就不会变，如果只用 min-width 和 min-height 的话，IE 下面根本等于没有设置宽度和高度。
比如要设置背景图片，这个宽度是比较重要的。要解决这个问题，可以这样：
#box{ width: 80px; height: 35px;}html>body #box{ width: auto; height: auto; min-width: 80px; min-height: 35px;}

5.页面的最小宽度

min -width 是个非常方便的 CSS 命令，它可以指定元素最小也不能小于某个宽度，这样就能保证排版一直正确。但 IE 不认得这个，而它实际上把 width 当 做最小宽度来使。为了让这一命令在 IE 上也能用，可以把一个 放到 <body> 标签下，然后为 div 指定一个类，然后 CSS 这样设计：
#container{ min-width: 600px;
width:expression(document.body.clientWidth < 600? "600px": "auto");}
第一个 min-width 是正常的；但第 2 行的 width 使用了 Javascript，这只有 IE 才认得，这也会让你的 HTML 文档不太正规。它实际上通过 Javascript 的判断来实现最小宽度。

6.DIV 浮动 IE 文本产生 3 象素的 bug

左边对象浮动，右边采用外补丁的左边距来定位，右边对象内的文本会离左边有 3px 的间距.
#box{ float:left; width:800px;}
#left{ float:left; width:50%;}
#right{ width:50%;}
*html #left{ margin-right:-3px; //这句是关键}

7.IE 捉迷藏的问题

当 div 应用复杂的时候每个栏中又有一些链接，DIV 等这个时候容易发生捉迷藏的问题。
有些内容显示不出来，当鼠标选择这个区域是发现内容确实在页面。 解决办法：对#layout 使用 line-height 属性 或者给#layout 使用固定高和宽。页面结构尽量简单。
8.高度不适应
高度不适应是当内层对象的高度发生变化时外层高度不能自动进行调节，特别是当内层对象使用 margin 或 paddign 时。
例：
#box {background-color:#eee; }
#box p {margin-top: 20px;margin-bottom: 20px;
text-align:center; }

p 对象中的内容

解决方法：在 P 对象上下各加 2 个空的 div 对象 CSS 代码：.1{height:0px;overflow:hidden;}或者为 DIV 加上 border 属性。
9.IE6 下为什么图片下有空隙产生

解 决这个 BUG 的方法也有很多,可以是改变 html 的排版,或者设置 img 为 display:block 或者设置 vertical-align 属性为 vertical-align:top | bottom |middle |text-bottom 都可以解决.
10.如何对齐文本与文本输入框
加上 vertical-align:middle;
11.web 标准中定义 id 与 class 有什么区别吗

一.web 标准中是不容许重复 ID 的,比如 div id="aa" 不容许重复 2 次,而 class 定义的是类,理论上可以无限重复, 这样需要多次引用的定义便可以使用他.

二.属性的优先级问题
ID 的优先级要高于 class,看上面的例子

三.方便 JS 等客户端脚本,如果在页面中要对某个对象进行脚本操作,那么可以给他定义一个 ID,否则只能利用遍历页面元素加上指定特定属性来找到它,这是相对浪费时间资源,远远不如一个 ID 来得简单.
12. LI 中内容超过长度后以省略号显示的方法
li {
width:200px;
white-space:nowrap;
text-overflow:ellipsis;
-o-text-overflow:ellipsis;
overflow: hidden;
}

13.为什么 web 标准中 IE 无法设置滚动条颜色了

解决办法是将 body 换成 html
html {
scrollbar-face-color:#f6f6f6;
scrollbar-highlight-color:#fff;
scrollbar-shadow-color:#eeeeee;
scrollbar-3dlight-color:#eeeeee;
scrollbar-arrow-color:#000;
scrollbar-track-color:#fff;
scrollbar-darkshadow-color:#fff;
}
14、为什么无法定义 1px 左右高度的容器
IE6 下这个问题是因为默认的行高造成的,解决的方法也有很多,例如:overflow:hidden | zoom:0.08 | line-height:1px
15、怎么样才能让层显示在 FLASH 之上呢

解决的办法是给 FLASH 设置透明
<param name="wmode" value="transparent" />
16、怎样使一个层垂直居中于浏览器中

这里我们使用百分比绝对定位,与外补丁负值的方法,负值的大小为其自身宽度高度除以二
六、DOCTYPE 与浏览器模式详解（标准模式&混杂模式）
模式可以分为两类：标准模式和混杂模式，其中，标准模式又可更严格的分为近似标准模式、标准模式、超级标准模式。
什么是 DOCTYPE：

DOCTYPE，或者称为 Document Type Declaration（文档类型声明，缩写 DTD）。

通常情况下，DOCTYPE 位于一个 HTML 文档的最前面的

位置，位于根元素 HTML 的起始标签之前。这样一来，在浏览器解析 HTML 文

档正文之前就可以确定当前文档的类型，以决定其需要采用的

渲染模式（不同的渲染模式会影响到浏览器对于 CSS 代码甚至 JavaScript 脚本的解析）。

用 JS 判断浏览器当前的模式：

document.write(document.compatMode == "CSS1Compat" ? "当前处于标准模式" : "当前处于混杂模式");

七、HTML 与 XHTML 的区别
XHTML 文档必须具有良好完整的排版(well-formed)
元素和属性名必须小写
对非空元素,必须使用结束标签
属性值必须在引号中
属性最小化
空元素
属性值中的空白字符处理
Script and Style 元素
SGML 排斥
具有'id'和'name'属性的元素

八、jpg,Gif,png-8,png-24 图片格式的区别
GIF
GIF 最突出的地方就是他支持动画,同时 GIF 也是一种无损的图片格式,也就是说你在修改图片之后,图片质量并没有损失.再者 GIF 支持半透明(全透明或是全不透明).根据 Google 的说法,GIF 适用于很小或是较简单的图片(10×10 以下或是 3 种颜色以下的图片).

PNG
首先,PNG 包括了 PNG-8 跟真彩色-PNG(PNG-24 or PNG-32).那 PNG 相对于 GIF 最大的优势是:
通常体积会更小
支持 alpha(全透明)
但是我们知道 PNG 是不支持动画的.
同时需要留意 IE6 是可以支持 PNG-8 的,但是在处理 PNG-24 的透明时会显示会灰色.相关例子可以参考 sitepoint.
通常图片保存为 PNG-8 会在同等质量下获得比 GIF 更小的体积,而全透明的图片我们现在只能使用 PNG-24.但是请留意在保存图片在 PNG-8 与 GIF 中进行比较.因为定律并不一直正确.

JPG
JPG 所能显示的颜色比 GIF,PNG 要多的多,同时得到很好的压缩,所以 JPG 很适用于保存数码照片.但是注意它是一种失真压缩,这意味着你每次修改图片都会造成像素失真.
看了上面的介绍你应该对使用哪种格式保存哪种图片有了大概的了解.简单的说就是:小图片或网页基本元素(如按钮),考虑 PNG-8 或 GIF.照片则考虑 JPG.

九、link 和@import 的区别

两者都是外部引用 CSS 的方式，但是存在一定的区别：

区别 1：link 是 XHTML 标签，除了加载 CSS 外，还可以定义 RSS 等其他事务；@import 属于 CSS 范畴，只能加载 CSS。

区别 2：link 引用 CSS 时，在页面载入时同时加载；@import 需要页面网页完全载入以后加载。

区别 3：link 是 XHTML 标签，无兼容问题；@import 是在 CSS2.1 提出的，低版本的浏览器不支持。

区别 4：ink 支持使用 Javascript 控制 DOM 去改变样式；而@import 不支持。

补充：@import 最优写法

@import 的写法一般有下列几种：

@import 'style.css' //Windows IE4/ NS4, Mac OS X IE5, Macintosh IE4/IE5/NS4 不识别

@import "style.css" //Windows IE4/ NS4, Macintosh IE4/NS4 不识别

@import url(style.css) //Windows NS4, Macintosh NS4 不识别

@import url('style.css') //Windows NS4, Mac OS X IE5, Macintosh IE4/IE5/NS4 不识别

@import url("style.css") //Windows NS4, Macintosh NS4 不识别

由上分析知道，@import url(style.css) 和@import url("style.css")是最优的选择,兼容的浏览器最多。从字节优化的角度来看@import url(style.css)最值得推荐。

十、浅析网页 Transitional 和 Strict 的文档声明的区别

在 Strict DOCTYPEs 下不支持的标签

- center
- font
- iframe
- srike
- u

在 Strict DOCTYPEs 下不支持的属性

- align (表格相关的支持：col, colgroup, tbody, td, tfoot, th, thead, and tr)
- language
- background
- bgcolor
- border (table 支持)
- height (img 和 object 支持)
- hspace
- name (在 HTML 4.01 Strict 中支持，XHTML 1.0 Strict 中的 form 和 img 不支持)
- noshade
- nowrap
- target
- text, link, vlink, 和 alink
- vspace
- width (img, object, table, col, 和 colgroup 都支持)

内容模型的区别

元素类型的模型描述了什么样的元素类型实例可以被包含。这一点上，两种文档声明的最大区别在于 blockquote, body, 和 form 元素仅能够包含块级元素，如：

- 文本和图像不允许直接包含在 body 中，必须被 p 或者 div 等块级元素包含
- input 元素不能直接是 form 元素的下一层
- blockquote 元素内的文本，必须被 p 或者 div 等块级元素包含

将所有的表现都交给 CSS，恪守 Strict 标准

十、sessionStorage 、localStorage 和 cookie 之间的区别(转)

sessionStorage 和 localStorage 是 HTML5 Web Storage API 提供的，可以方便的在 web 请求之间保存数据。有了本地数据，就可以避免数据在浏览器和服务器间不必要地来回传递。

sessionStorage、 localStorage、cookie 都是在浏览器端存储的数据，其中 sessionStorage 的概念很特别，引入了一个“浏览器窗口”的概念。sessionStorage 是在同源的同窗口（或 tab）中，始终存在的数据。也就是说只要这个浏览器窗口没有关闭，即使刷新页面或进入同源另一页面， 数据仍然存在。关闭窗口后，sessionStorage 即被销毁。同时“独立”打开的不同窗口，即使是同一页面，sessionStorage 对象也是不同的。

Web Storage 带来的好处：减少网络流量：一旦数据保存在本地后，就可以避免再向服务器请求数据，因此减少不必要的数据请求，减少数据在浏览器和服务器间不必要地来回传递。快速显示数据：性能好，从本地读数据比通过网络从服务器获得数据快得多，本地数据可以即时获得。再加上网页本身也可以有缓存，因此整个页面和数据都在本地的话，可以立即显示。临时存储：很多时候数据只需要在用户浏览一组页面期间使用，关闭窗口后数据就可以丢弃了，这种情况使用 sessionStorage 非常方便。

浏览器本地存储与服务器端存储之间的区别其实数据既可以在浏览器本地存储，也可以在服务器端存储。

浏览器端可以保存一些数据，需要的时候直接从本地获取，sessionStorage、localStorage 和 cookie 都由浏览器存储在本地的数据。

服 务器端也可以保存所有用户的所有数据，但需要的时候浏览器要向服务器请求数据。1.服务器端可以保存用户的持久数据，如数据库和云存储将用户的大量数据保 存在服务器端。2.服务器端也可以保存用户的临时会话数据。

服务器端的 session 机制，如 jsp 的 session 对象，数据保存在服务器上。实现上，服务器和浏览器之间仅需传递 session id 即可，服务器根据 session id 找到对应用户的 session 对象。会话数据仅在一段时间内有效，这个时间就是 server 端设置的 session 有效期。

服务器端保存所有的用户的数据，所以服务器端的开销较大，而浏览器端保存则把不同用户需要的数据分布保存在用户各自的浏览器中。浏览器端一般只用来存储小数据，而服务器可以存储大数据或小数据。服务器存储数据安全一些，浏览器只适合存储一般数据。

sessionStorage 、localStorage 和 cookie 之间的区别

共同点：都是保存在浏览器端，且同源的。区别：cookie 数据始终在同源的 http 请求中携带（即使不需要），即 cookie 在浏览器和服务器间来回传递。而 sessionStorage 和 localStorage 不会自动把数据发给服务器，仅在本地保存。cookie 数据还有路径（path）的概念，可以限制 cookie 只属于某个路径 下。存储大小限制也不同，cookie 数据不能超过 4k，同时因 为每次 http 请求都会携带 cookie，所以 cookie 只适合保存很小的数据，如会话标识。sessionStorage 和 localStorage 虽然也有存储大小的限制，但比 cookie 大得多，可以达到 5M 或更大。数据有效期不同，sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；localStorage：始终有效，窗口或浏览器关闭也一直保存,因此用作持久数据;cookie 只在设置的 cookie 过期时间之前一直有效，即使窗口或浏览器关闭。作用域不同，

sessionStorage 不在不同的浏览器窗口中共享，即使是同一个页面；localStorage 在所有同源窗口中都是共享的；cookie 也是在所有同源窗口中都是共享的。Web Storage 支持事件通知机制，可以将数据更新的通知发送给监听者。Web Storage 的 api 接口使用更方便。

sessionStorage 和 localStorage 之间的区别见上面的区别 3、4

sessionStorage 与页面 js 数据对象的区别页面中一般的 js 对象或数据的生存期是仅在当前页面有效，因此刷新页面或转到另一页面这样的重新加载页面的情况，数据就不存在了。而 sessionStorage 只要同源的同窗口（或 tab）中，刷新页面或进入同源的不同页面，数据始终存在。也就是说只要这个浏览器窗口没有关闭，加载新页面或重新加载，数据仍然存在。

cookie，容量 4kb，默认各种浏览器都支持，缺陷就是每次请求，浏览器都会把本机存的 cookies 发送到服务器，无形中浪费带宽。

userdata，只有 ie 支持，单个容量 64kb，每个域名最多可存 10 个共计 640k 数据。默认保存在 C:\Documents and Settings\Administrator\UserData\目录下，保存格式为 xml。关于 userdata 更多资料参考 http://msdn.microsoft.com/library/default.asp?url=/workshop/author/behaviors/reference/behaviors/userdata.asp

sessionStorage 与 localStorage

Web Storage 实际上由两部分组成：sessionStorage 与 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。

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。

为什么选择 Web Storage 而不是 Cookie?

与 Cookie 相比，Web Storage 存在不少的优势，概括为以下几点：

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

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

3. 更多丰富易用的接口：Web Storage 提供了一套更为丰富的接口，使得数据操作更为简便。

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

兼容性如何?

接下来的各种测试是在以下浏览器中进行的：IE8、Firefox3.6、Chrome5、Safari4、Opera10，事实证明各个浏览器在 API 方面的实现基本上一致，存在一定的兼容性问题，但不影响正常的使用。

十一、javascript 数据类型强制转换和隐式转换

强制转换

一、转换为数值类型

Number(参数) 把任何的类型转换为数值类型（全局函数，直接使用，注意首字母是大写！）

A.如果是布尔值，false 为 0，true 为 1

B.如果是数字，转换成为本身。将无意义的后导 0（小数点后面的 0）和前导 0 去掉。

C.如果 Null 转换为 0

D.如果是 undefined 转换为 NaN not a number

E.如果对象则会先调用对象的 valueOf(),如果 valueOf()返回的是 NaN,然后再调用对象的 toString()（后面学^^^）

F.如果是字符串

1.如果字符串当中只有数字，转换为 10 进制(忽略前导 0 和后导 0)

2.如果是有效的规范的浮点型，转换为浮点值(忽略前导 0 和后导 0)

3.如果是空字符串，则转换为 0

4.如果是其他的值，返回 NaN

parseInt(参数 1，参数 2) 将字符串转换为整数，常用于输出结果转换，它忽略后面的小数；

A.如果一个字符串只包含数字，则以 10 进制的方式转换为整型。如果该字符串包含的数字是浮点型数值，那么他会直接去掉浮点后面的数（不会四舍五入）。

B.他会自动忽略字符串前面的空格，直到找到第一个非空的数值字符串，然后解析它，如果再遇见一个空格就结束。

如：

```
var a=parseInt("   79   875");
var a=parseInt("   79   875");
alert(a)//（这个地方弹出的是 79）
```

C.如果字符串的第一个字符不是空格、数字、—（负号），那么返回 NaN。如果字符串开头是数字，后面是非数值型字符串。它就会只解析到数值型字符串。

D.参数 1:可以是八进制，十六进制，十进制等数值。

八进制：0 开头，后面的数字不能超过 7 十六进制：0x

开头，后面的数字是 0-9 或 a-f。

参数 2，控制输出模式 可添 2-32，

例：

var a=0×000000

alert(parseInt(a,8))（把十六进制转换为八进制。）

parseFloat() 将字符串转换为浮点数

*此处说的字符串是带数值的字符串。

A.字符串当中的.只有第一个有效，其他的都是无效的。

B.如果字符串是一个有效的整数，他返回的是整数，不会返回浮点数。

二、转换为字符串类型

1.String(参数)（注意 s 要大写)可以将任何的类型转换为字符串

null 和 undefined: 也都会转换为字符串，分别是 null 和 undefined

布尔类型:会返回 true 和 false

数值类型:本身的字符串

2.toString()

每个对象都有一个 toString 的方法。

调用的格式 对象.toString()

作用是将对象以字符串的方式来表示

array.toString()（array:数组。数组调用字符串） 返回由’，’分割的字符串

例：

```
var arr=[1,2,3,4];
alert(arr.toString())//（弹出“1,2,3,4”）；
Boolean.toString()  // 两个值  true false
String.toString()   // 返回本身
Number.toString(参数)     //返回本身的字符串形式
```

注意：null 和 undefined 没有 toString()方法

三、转换为布尔类型

Boolean() 可以将任何类型的值转换为布尔值

转换为假：“”、 0、 NaN 、undefined、 false

其他的全部都转换为真。

隐式转换

一、函数类

isNaN()

该函数会对参数进行隐式的 Number()转换，如果转换不成功则返回 true;

```
var a="abc";
alert(isNaN(a))//(转换不成功，弹出的是 true)
```

alert()

输出的内容隐式的转换为字符串（引用输出的内容就要做相应的转换）

二、运算符类

1.算数运算符

- * / %

如果操作数不是数值，将会隐式的调用 Number()函数，按照这个函数的转换规则进行转换，

```
var b=89;
var a=false;
alert(a-b)//（把 a 按照 Number（）函数的规则转为 0，然后再减去 89）
如果转换不成功，整个表达式返回 NaN）
var b=89;
var a=” zhouxiaog”
alert(a-b)//(把 a 按照 Number（）函数的规则转为 nan，然后再减去 89，然后按照运算符的规则，最终弹出的也是 NAN)
```

+

如果操作数都是数值，就进行相加

任何数据类型和字符串相加，都会隐式的调用他们的 toString()方法，然后返回他们拼接的结果。

如果操作数都是布尔值，那么进行 Number()转换，false 为 0，true 为 1，进行相加。

2.关系运算符

关系运算符的操作数可以是任何类型，如果操作数不是数值类型，将会隐式的转换

(1)他运算的结果都是布尔值

(2)都是字符串的时候，他会先隐式转换成 ASCII 码然后进行比较他们的第一个字母。

(3)都是数值的时候，他会正常比较

(4)当一个是字符串，另一个是数值的时候，把字符串尝试转换成 Number()数值类型，然后进行比较，如果不能转换成数值类型，则会返回 NaN,然后返回假

(5) undefined == null

(6)如果两个都是数值型字符串，那只比较第一个数。

(7)如果一个数值和布尔值进行比较，会把布尔值隐式转换为数值再进行比较，true 为 1，false 为 0。

3.等性运算符 == !=

会对操作数隐式的转换后再比较值

(1)如果其中至少有一个是布尔值，那么会隐式的调用 Number()进行转换，然后比较。

(2)如果一个为字符串。另一个为数值，那么会隐式的调用 Number()对字符串进行转换，如果转换不成功，则返回 false;

(3) undefined == null

(a) 比较字符串的时候是比较他们的 ASCII 码是否相等

(b) 比较两个数值的时候是比较他们的数值是否相等

(c) 比较函数的时候，判断他们的位置是否相等。

4.逻辑运算符

A. 放在表达式里面用于判断。

B. 给变量赋值

var a= b&&c

如果一个运算数是对象，另一个是隐式的调用 Boolean()函数，返回该对象。

如果两个运算数都是对象，返回第二个对象。

如果某个运算数是 null，返回 null。

如果某个运算数是 NaN，返回 NaN。

如果某个运算数是 undefined，发生错误。

var a=b||c

如果一个运算数是对象，并且该对象左边的运算数隐式的调用 Boolean()函数值为 false，则返回该对象。

如果两个运算数都是对象，返回第一个对象。

如果最后一个运算数是 null，并且其他运算数值均为 false，则返回 null。

如果最后一个运算数是 NaN，并且其他运算数值均为 false，则返回 NaN。

如果某个运算数是 undefined，发生错误。

三、语句类

```
if(表达式){
}else{
}
var 变量= Boolean expression?真值:假值
while(表达式){（里面不管输入什么都会隐式的转换为
boolean 值，然后 true 就执行语句块内的语句，false
就不执行。）
}
```

if 语句和三元表达式里面的表达式会隐式的调用 Boolean()函数，按照这个函数的转换规则，转换为相应的布尔值.

十二、Ajax

1、ajax 是什么？

AJAX 是一种用于创建快速动态网页的技术。

AJAX 即“Asynchronous Javascript And XML”（异步 JavaScript 和 XML），是指一种创建交互式网页应用的网页开发技术。

AJAX = 异步 JavaScript 和 XML（标准通用标记语言的子集）。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页页面。

2、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 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.

1、创建 XMLHttpRequest 对象

不同的浏览器使用的异步调用对象也有所不同,在 IE 浏览器中异步调用使用的是 XMLHTTP 组件中的 XMLHttpRequest 对象,而在 Netscape、Firefox 浏览器中则直接使用 XMLHttpRequest 组件。因此,在不同浏览器中创建 XMLHttpRequest 对象的方 式都有所不同.

在 IE 浏览器中创建 XMLHttpRequest 对象的方式如下所示:

```
var xmlHttpRequest = new ActiveXObject("Microsoft.XMLHTTP");
```

在 Netscape 浏览器中创建 XMLHttpRequest 对象的方式如下所示:

var xmlHttpRequest = new XMLHttpRequest();

由于无法确定用户使用的是什么浏览器,所以在创建 XMLHttpRequest 对象时,最好将以上两种方法都加上.如以下代码所示:

```
<html>
  <head>
  <title>创建 XMLHttpRequest 对象</title>
<script language = "javascript" type = "text/javascript">
  <!--
var xmlHttpRequest;   //定义一个变量,用于存放 XMLHttpRequest 对象
function createXMLHttpRequest()        //创建 XMLHttpRequest 对象的方法
{
      if(window.ActiveXObject)      //判断是否是 IE 浏览器
      {
         xmlHttpRequest                    =                    new
ActiveXObject("Microsoft.XMLHTTP");    // 创 建  IE  浏 览 器 中 的 XMLHttpRequest 对象
        }else if(window.XMLHttpRequest) //判断是否是 Netscape 等其他支持 XMLHttpRequest 组件的浏览器{
X          mlHttpRequest = new XMLHttpRequest();   //创建其他浏览器上的 XMLHttpRequest 对象
      }
}
-->
</script>
createXMLHttpRequst();      //调用创建对象的方法
</head>
<body>
</body>
</html>
```

"if(window.ActiveXObject)"用来判断是否使用 IE 浏览器.其中 ActiveXOject 并不是 Windows 对象的标准属性, 而是 IE 浏览器中专有的属性,可以用于判断浏览器是否支持 ActiveX 控件.通常只有 IE 浏览器或以 IE 浏览器为核心的浏览器才能支持 Active 控 件.

"else if(window.XMLHttpRequest)"是为了防止一些浏览器既不支持 ActiveX 控件,也不支持 XMLHttpRequest 组件而进 行的判断.其中 XMLHttpRequest 也不是 window 对象的标准属性,但可以用来判断浏览器是否支持 XMLHttpRequest 组件.

如果浏览器既不支持 ActiveX 控件,也不支持 XMLHttpRequest 组件,那么就不会对 xmlHttpRequest 变量赋值.

2、创建 HTTP 请求

创建了 XMLHttpRequest 对象之后,必须为 XMLHttpRequest 对象创建 HTTP 请求,用于说明 XMLHttpRequest 对象要从哪里获取数据.通常可以是网站中的数据,也可以是本地中其他文件中的数据.

创建 HTTP 请求可以使用 XMLHttpRequest 对象的 open()方法,其语法代码如下所示:

XMLHttpRequest.open(method,URL,flag,name,password)

代码中的参数解释如下所示:

method:该参数用于指定 HTTP 的请求方法,一共有 get、post、head、put、delete 五种方法,常用的方法为 get 和 post。

URL:该参数用于指定 HTTP 请求的 URL 地址,可以是绝对 URL,也可以是相对 URL。

flag:该参数为可选参数,参数值为布尔型。该参数用于指定是否使用异步方式。true 表示异步方式、false 表示同步方式,默认为 true。

name:该参数为可选参数,用于输入用户名。如果服务器需要验证,则必须使用该参数。

password:该参数为可选参数,用于输入密码。如果服务器需要验证,则必须使用该参数。通常可以使用以下代码来访问一个网站文件的内容。

xmlHttpRequest.open("get","http://www.aspxfans.com/BookSupport/JavaScript/ajax.htm",true);

或者使用以下代码来访问一个本地文件内容:

xmlHttpRequest.open("get","ajax.htm",true);

注意:如果 HTML 文件放在 Web 服务器上,在 Netscape 浏览器中的 JavaScript 安全 机制不允许与本机之外的主机进行通信。也就是说,使用 open()方法只能打开与 HTML 文件在同一个服务器上的文件。而在 IE 浏览器中则无此限制(虽然 可以打开其他服务器上的文件,但也会有警告提示)。

3、设置响应 HTTP 请求状态变化的函数

创建完 HTTP 请求之后,应该就可以将 HTTP 请求发送给 Web 服务器了。然而,发送 HTTP 请求的目的是为了接收从服务器中返回的数据。从创建 XMLHttpRequest 对象开始,到发送数据、接收数据、XMLHttpRequest 对象一共会经历以下 5 中状态。

⑴未初始化状态。在创建完 XMLHttpRequest 对象时,该对象处于未初始化状态,此时 XMLHttpRequest 对象的 readyState 属性值为 0。

⑵初始化状态。在创建完 XMLHttpRequest 对象后使用 open()方法创建了 HTTP 请求时,该对象处于初始化状态。此时 XMLHttpRequest 对象的 readyState 属性值为 1。

⑶发送数据状态。在初始化 XMLHttpRequest 对象后,使用 send()方法发送数据时,该对象处于发送数据状态,此时 XMLHttpRequest 对象的 readyState 属性值为 2。

⑷接收数据状态。Web 服务器接收完数据并进行处理完毕之后,向客户端传送返回的结果。此时,XMLHttpRequest 对象处于接收数据状态,XMLHttpRequest 对象的 readyState 属性值为 3。

⑸完成状态。XMLHttpRequest 对象接收数据完毕后,进入完成状态,此时 XMLHttpRequest 对象的 readyState 属性值为 4。 此时接收完毕后的数据存入在客户端计算机的内存中,可以使用 responseText 属性或 responseXml 属性来获取数据。

只有在 XMLHttpRequest 对象完成了以上 5 个步骤之后,才可以获取从服务器端返回的数据。因此,如果要获得从服务器端返回的数据,就必须要先判断 XMLHttpRequest 对象的状态。

XMLHttpRequest 对象可以响应 readystatechange 事件,该事件在 XMLHttpRequest 对象状态改变时(也就是 readyState 属性值改变时)激发。因此,可以通过该事件调用一个函数,并在该函数中判断 XMLHttpRequest 对象的 readyState 属性值。如果 readyState 属性值为 4 则使用 responseText 属性或 responseXml 属性来获取数据。具体代码如下所示:

```
//设置当 XMLHttpRequest 对象状态改变时调用的函数,注意函数名后面不要添加小括号
```

```
xmlHttpRequest.onreadystatechange = getData;
//定义函数
function getData(){
    //判断 XMLHttpRequest 对象的 readyState 属性值是否为 4，如果为 4 表示异步调用完成
    if(xmlHttpRequest.readyState == 4){
        //设置获取数据的语句
    }
}
```

4、设置获取服务器返回数据的语句

如果 XMLHttpRequest 对象的 readyState 属性值等于 4，表示异步调用过程完毕，就可以通过 XMLHttpRequest 对象的 responseText 属性或 responseXml 属性来获取数据。

但是，异步调用过程完毕，并不代表异步调用成功了，如果要判断异步调用是否成功,还要判断 XMLHttpRequest 对象的 status 属性值,只有该 属性值为 200，才表示异步调用成功，因此，要获取服务器返回数据的语句，还必须要先判断 XMLHttpRequest 对象的 status 属性值是否等于 200，如以下代码所示：

```
if(xmlHttpRequst.status == 200){
    //使用以下语句将返回结果以字符串形式输出
    document.write(xmlHttpRequest.responseText);
    //或者使用以下语句将返回结果以 XML 形式输出
    //document.write(xmlHttpRequest.responseXML);
 }
```

注意：如果 HTML 文件不是在 Web 服务器上运行，而是在本地运行，则 xmlHttpRequest.status 的返回值为 0。因此，如果该文件在本地运行，则应该加上 xmlHttpRequest.status == 0 的判断。

通常将以上代码放在响应 HTTP 请求状态变化的函数体内，如以下代码所示：

```
//设置当 XMLHttpRequest 对象状态改变时调用的函数，注意函数名后面不要添加小括号
xmlHttpRequest.onreadystatechange = getData;
//定义函数
function getData(){
    //判断 XMLHttpRequest 对象的 readyState 属性值是否为 4，如果为 4 表示异步调用完成
    if(xmlHttpRequest.readyState==4){
        //设置获取数据的语句
            if(xmlHttpRequest.status == 200 || xmlHttpRequest.status == 0){
            //使用以下语句将返回结果以字符串形式输出
            document.write(xmlHttpRequest.responseText);
            //或者使用以下语句将返回结果以 XML 形式输出
            //docunment.write(xmlHttpRequest.responseXML);
        }
    }
}
```

5、发送 HTTP 请求

在经过以上几个步骤的设置之后，就可以将 HTTP 请求发送到 Web 服务器上去了。发送 HTTP 请求可以使用 XMLHttpRequest 对象的 send()方法，其语法代码如下所示：

```
XMLHttpRequest.send(data)
```

其中 data 是个可选参数，如果请求的数据不需要参数，即可以使用 null 来替代。data 参数的格式与在 URL 中传递参数的格式类似，以下代码为一个 send()方法中的 data 参数的示例：

```
name=myName&value=myValue
```

只有在使用 send()方法之后，XMLHttpRequest 对象的 readyState 属性值才会开始改变，也才会激发 readystatechange 事件，并调用函数。

6、局部更新

在通过 Ajax 的异步调用获得服务器端数据之后，可以使用 JavaScript 或 DOM 来将网页中的数据进行局部更新。常用的局部更新的方式有以下 3 种：

⑴表单对象的数据更新

表单对象的数据更新，通常只要更改表单对象的 value 属性值，其语法代码如下所示：

```
FormObject.value = "新数值"
```

有关表单对象的数据更新的示例如以下代码所示：

```
<html>
  <head>
  <title>局部更新</title>
  <script language = "javascript" type = "text/javascript">
  <!--
  function changeData(){
      document.myForm.myText.value = "更新后的数据"
  }
-->
</head>
<body>
<form name = "myForm">
    <input type = "text" value = "原数据" name = "myText">
    <input type = "button" value = "更新数据" onclick = "changeData()">
</form>
</body>
</html>
```

⑵IE 浏览器中标签间文本的更新

在 HTML 代码中，除了表单元素之外，还有很多其他的元素，这些元素的开始标签与结束标签之间往往也会有一点文字（如以下代码所示），对这些文字的更新，也是局部更新的一部分。

```
<p>文字</p>
 <span>文字</span>
 <div>文字</div>
 <label>文字</label>
<b>文字</b>
<i>文字</i>
```

在 IE 浏览器中，innerText 或 innerHTML 属性可以用来更改标签间文本的内容。其中 innerText 属性用于更改开始标签与结束标签之间的纯文本内容，而 innerHTML 属性用于更改 HTML 内容。如以下代码所示：

```
<html>
  <head>
  <title>局部更新</title>
  <script language = "javascript" type = "text/javascript">
<!--
  function changeData(){
        myDiv.innerText = "更新后的数据";
  }
-->
</script>
    </head>
    <body>
        <div id = "myDive">原数据</div>
        <input type = "button" value = "更新数据" onclick = "changeData()">
    </body>
</html>
```

⑶DOM 技术的局部刷新

innerText 和 innerHTML 两个属性都是 IE 浏览器中的属性，在 Netscape 浏览器中并不支持该属性。但无论是 IE 浏览器还是 Netscape 浏览器，都支持 DOM。在 DOM 中，可以修改标签间的文本内容。

在 DOM 中，将 HTML 文档中的每一对开始标签和结束标签都看成是一个节点。例如 HTML 文档中有一个标签如下所示，那么该标签在 DOM 中称之为一个“节点”。

```
        <div id = "myDiv">原数据</div>
```

在 DOM 中使用 getElementById()方法可以通过 id 属性值来查找该标签（或者说是节点），如以下语句所示：

```
    var node = document.getElementById("myDiv");
```

注意：在一个 HTML 文档中，每个标签中的 id 属性值是不能重复的。因此，使用 getElementById()方法获得的节点是唯一的。

在 DOM 中，认为开始标签与结束标签之间的文本是该节点的子节点，而 firstChild 属性可以获得一个节点下的第 1 个子节点。如以下代码可以获 得 节点下的第 1 个子节点，也就是 标签与 标签之间的文字节点。

```
    node.firstChild
```

注意，以上代码获得的是文字节点，而不是文字内容。如果要获得节点的文字内容，则要使用节点的 nodeValue 属性。通过设置 nodeValue 属性值，可以改变文字节点的文本内容。完整的代码如下所示：

```
<html>
  <head>
  <title>局部更新</title>
  <script language = "javascript" type = "text/javascript">
  <!--
  function changeData(){
        //查找标签（节点）
        var node = document.getElementById("myDiv");
        //在 DOM 中标签中的文字被认为是标签中的子节点
        //节点的 firstChild 属性为该节点下的第 1 个子节点
        //nodeValue 属性为节点的值，也就是标签中的文本值
        node.firstChild.nodeValue = "更新后的数据";
  }
   -->
</script>
</head>
</html>
```

注意：目前主流的浏览器都支持 DOM 技术的局部刷新。

7、完整的 AJAX 实例

```
  <html>
  <head>
      <title>AJAX 实例</title>
<script language="javascript" type="text/javascript">
<!--
var xmlHttpRequest;   //定义一个变量用于存放 XMLHttpRequest 对象
/定义一个用于创建 XMLHttpRequest 对象的函数
function createXMLHttpRequest(){
      if(window.ActiveXObject){
          //IE 浏览器的创建方式
          xmlHttpRequest = new ActiveXObject("Microsoft.XMLHTTP");
      }else if(windew.XMLHttpRequest){
          //Netscape 浏览器中的创建方式
          xmlHttpRequest = new XMLHttpRequest();
     }
}
//响应 HTTP 请求状态变化的函数
function httpStateChange(){
      //判断异步调用是否完成
      if(xmlHttpRequest.readyState == 4){
            //判断异步调用是否成功,如果成功开始局部更新数据
            if(xmlHttpRequest.status == 200||xmlHttpRequest.status
== 0){
                  //查找节点
                  var node = document.getElementById("myDIv");
                  //更新数据
                  node.firstChild.nodeValue                    =
xmlHttpRequest  .responseText;
            }else{
                //如果异步调用未成功,弹出警告框,并显示出错信息
              alert("异步调用出错/n 返回的 HTTP 状态码为:"+xmlHttpRequest.status + "/n 返回的 HTTP 状态信息为:" +
xmlHttpRequest.statusText);
            }
      }
}
//异步调用服务器段数据
function getData(name,value){
```

```
    //创建 XMLHttpRequest 对象
    createXMLHttpRequest();
    if(xmlHttpRequest!=null){
        //创建 HTTP 请求
        xmlHttpRequest.open("get","ajax.text",true)
        //设置 HTTP 请求状态变化的函数
        xmlHttpRequest.onreadystatechange = httpStateChange;
        //发送请求
        xmlHttpRequest.send(null);
    }
}
-->
</script>
</head>
<body>
    <div id="myDiv">原数据</div>
      <input type = "button" value = "更新数据" onclick = "getData()">
</body>
</html>
```

三、ajax 原理

首先我们 ajax 体现的是异步关系，但什么是异步呢，我们在刷新一个页面的时候，一般浏览器下面都会有一个绿色或者蓝色的进度条，刷一个页面就刷满了，这个就叫同步，同步的意思就是我每次刷新一下页面，刷出了一个请求，服 务器那边接到我们的请求，这个数据进行一系列的处理，处理完之后，然后返回到我的浏览器这里，到浏览器这边就成了一段 html5,或者是一段 html,css3，js 这样的一段代码，我的浏览器认识这些代码，然后把这些代码解析成为丰富多彩的页面了。这就是同步：当我整张页面刷新完的时候，页 面中所加载到的数据就都是同步的，

异步是什么：当我在页面上触发一些动作的时候，比如说鼠标移动的这么个动作，比如说点击的动作，比如说上下滑屏的动作，这个时候一定是动作触发的，当你触 发一些动作的时候，举一个最简单的例子，页面中的 tab 切换，当你从男装切换到女装的时候，女装的信息再次就会单独的向服务器发出请求，其它的地方都不 变，就是当我的男装切换到女装的时候，这个女装的请求就会单独的向服务器发出请求 比如说 我就要女装，她就会从服务器再次接到你这个请求的时候，它会同 它那边的数据当中摘选出来关于女装的相关信息，再次给你返回到页面中，记住，就是刷新女装这一块，底下的什么母音用品都没关系，这就是异步。一样的道理， 从女装移动到儿童用品上，那儿童的数据传过来。远程单独的去请求，

四、优点

1.较少的请求

2.较小的带宽

3.减少载入时间

4.更强的交互

5.更快的响应

6.不刷新

7、跨浏览器的兼容性好

五、缺点(Disadvantages)

1。一个 Ajax 请求往往没有浏览器浏览历史。

2。它禁用了网页状态书签。

3。它很难被搜索引擎抓取 Ajax 的内容。

4。AJAX 请求不能跨网域

5。Ajax 不容易测试。

六、ajax 特点

1、改善了表单的验证方式，不再需要打开新的页面，也不需要把这个页面数据提交

2、不需刷新页面，就可以改变页面的内容，减少用户等待时间

3、按需获取数据、每次只从服务器端获取需要的数据

4、读取外部的数据，进行数据处理整合

5、异步与服务器进行交互，在交互过程中无需等待，仍可继续操作

七、ajax 包含的技术

1、使用 DOM 进行动态的显示和交互

2、使用 XHTML 和 css 基于标准的表示技术

3、使用 XML 和 XSLT 进行数据的交换和处理

4、使用 xmlHttpRequest 进行异步的数据检索

5、使用 javascript 将以上技术融合在一块

十三、html5 离线应用 application cache

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。

二、如何实现

离线存储是通过 manifest 文件来管理的，需要服务器端的支持，不同的服务器开启支持的方式也是不同的。

CACHE 指定需要缓存的文件;NETWORK 指定只有通过联网才能浏览的文件，*代表除了在 CACHE 中的文件;FALLBACK 每行分别指定在线和离线时使用的文件

要让 manifest 管理存储，还需要在 html 标签中定义 manifest 属性

三、通过 JS 动态控制更新

四、浏览器与服务器的交互

曾经有面试题是这样的："描述在浏览器的地址栏中输入:http://www.baidu.com 后发生了什么?"。

1、服务端返回 baidu 页面资源，浏览器载入 html

2、浏览器开始解析

3、发现 link，发送请求载入 css 文件

4、浏览器渲染页面

5、发现图片，发送请求载入图片，并重新渲染

6、发送请求 js 文件，阻塞渲染。如果 js 对 dom 进行了操作，则会进行 rerender

对于支持离线存储的页面，浏览器和服务器的交互又是如何呢?

首次载入页面：

1-6 ： 同上

7：请求页面中需要缓存的页面和数据，就算在之前的步骤中已经请求过(这是个耗能的地方)

8：服务器返回所有请求文件，浏览器进行本地存储

再次载入页面：

1：发送请求

2：使用本地存储的离线文件

3：解析页面

4：请求服务端的 manifest 文件，判断是否有改变，返回 304 则表示没有改变进入步骤 5，否则进入步骤 6

5：进入首次载入页面的 7-8

6：使用本地存储，不重新请求

十四、IE & FF 函数和方法差异

一、函数和方法差异

1. getYear()方法

【分析说明】先看一下以下代码：

```
var year= new Date().getYear();
document.write(year);
```

在 IE 中得到的日期是"2010"，在 Firefox 中看到的日期是"110"，主要是因为在 Firefox 里面 getYear 返回的是 "当前年份-1900" 的值。

【兼容处理】

加上对年份的判断，如:

```
var year= new Date().getYear();
year = (year<1900?(1900+year):year);document.write(year);
```

也可以通过 getFullYear getUTCFullYear 去调用:

```
var year = new Date().getFullYear();
document.write(year);
```

2. eval()函数

【分析说明】在 IE 中,可以使用 eval("idName")或 getElementById("idName")来取得 id 为 idName 的 HTML 对象；Firefox 下只能使用 getElementById("idName")来取得 id 为 idName 的 HTML 对象。

【兼容处理】统一用 getElementById("idName")来取得 id 为 idName 的 HTML 对象。

3. const 声明

【分析说明】在 IE 中不能使用 const 关键字。如：

```
const constVar = 32;
```

在 IE 中这是语法错误。

【兼容处理】不使用 const ，以 var 代替。

4. var

【分析说明】请看以下代码：

```
echo=function(str){
document.write(str);
}
```

这个函数在 IE 上运行正常，Firefox 下却报错了。

【兼容处理】而在 echo 前加上 var 就正常了，这个就是我们提到 var 的目的。

5. const 问题

【分析说明】在 IE 中不能使用 const 关键字。如 const constVar = 32; 在 IE 中这是语法错误。

【解决方法】不使用 const ，以 var 代替。

二、样式访问和设置

1. CSS 的"float"属性

【分析说明】Javascript 访问一个给定 CSS 值的最基本句法是：object.style.property，但部分 CSS 属性跟 Javascript 中的保留字命名相同，如"float"，"for"，"class"等，不同浏览器写法不同。

在 IE 中这样写：

```
document.getElementById("header").style.styleFloat = "left";
```

在 Firefox 中这样写：

```
document.getElementById("header").style.cssFloat = "left";
```

【兼容处理】在写之前加一个判断，判断浏览器是否是 IE：

```
if(document.all){     document.getElementById("header").style.styleFloat = "left";}
else{          document.getElementById("header").style.cssFloat   = "left";}
```

2. 访问<label>标签中的"for"

【分析说明】和"float"属性一样，同样需要使用不现的句法区分来访问<label>标签中的"for"。

在 IE 中这样写：

```
var myObject = document.getElementById("myLabel");
var myAttribute = myObject.getAttribute("htmlFor");
```

在 Firefox 中这样写：

```
var myObject = document.getElementById("myLabel");
var myAttribute = myObject.getAttribute("for");
```

【兼容处理】解决的方法也是先 判断浏览器类型。

3. 访问和设置 class 属性

【分析说明】同样由于 class 是 Javascript 保留字的原因，这两种浏览器使用不同的 JavaScript 方法来获取这个属性。

IE8.0 之前的所有 IE 版本的写法：

```
var myObject = document.getElementById("header");
var myAttribute = myObject.getAttribute("className");
```

适用于 IE8.0 以及 firefox 的写法：

```
var myObject = document.getElementById("header");
var myAttribute = myObject.getAttribute("class");
```

另外，在使用 setAttribute()设置 Class 属性的时候，两种浏览器也存在同样的差异。

```
setAttribute("className",value);
```

这种写法适用于 IE8.0 之前的所有 IE 版本，注意：IE8.0 也不支持"className"属性了。

```
setAttribute("class",value);
```

适用于 IE8.0 以及 firefox。

【兼容处理】

方法一，两种都写上：

```
var myObject = document.getElementById("header");
myObject.setAttribute("class","classValue");
myObject.setAttribute("className","classValue"); //设置 header 的 class 为 classValue
```

方法二，IE 和 FF 都支持 object.className，所以可以这样写：

```
var myObject = document.getElementById("header");
myObject.className="classValue";//设置 header 的 class 为 classValue
```

方法三，先判断浏览器类型，再根据浏览器类型采用对应的写法。

4. 对象宽高赋值问题

【分析说明】FireFox 中类似 obj.style.height = imgObj.height 的语句无效。

【兼容处理】统一使用 obj.style.height = imgObj.height + 'px';

三、DOM 方法及对象引用

1. getElementById

【分析说明】先来看一组代码:

```
<!-- input 对象访问 1 -->
<input id="id" type="button"
value="click me" ōnclick="alert(id.value)"/>
```

在 Firefox 中，按钮没反应，在 IE 中，就可以，因为对于 IE 来说，一个 HTML 元素的 ID 可以直接在脚本中当作变量名来使用，而 Firefox 中不可以。

【兼容处理】尽量采用 W3C DOM 的写法，访问对象的时候，用 document.getElementById("id") 以 ID 来访问对象，且一个 ID 在页面中必须是唯一的，同样在以标签名来访问对象的时候，用 document.getElementsByTagName("div") [0] 。该方式得到较多浏览器的支持。

```
<!-- input 对象访问 2 -->
<input id="id" type="button" value="click me"
onclick="alert(document.getElementById('id').value)" />
```

2. 集合类对象访问

【分析说明】IE 下，可以使用()或[]获取集合类对象；Firefox 下，只能使用[]获取集合类对象。如:

```
document.write(document.forms("formName").src);//该写法在 IE 下能访问到 Form 对象的 scrc 属性
```

【兼容处理】将 document.forms("formName") 改为 document.forms["formName"]。统一使用[]获取集合类对象。

3. frame 的引用

【分析说明】IE 可以通过 id 或者 name 访问这个 frame 对应的 window 对象，而 Firefox 只可以通过 name 来访问这个 frame 对应的 window 对象。

例如如果上述 frame 标签写在最上层的 window 里面的 htm 里面，那么可以这样访问:

IE: window.top.frameId 或者 window.top.frameName 来访问这个 window 对象;

Firefox: 只能这样 window.top.frameName 来访问这个 window 对象。

【兼容处理】使用 frame 的 name 来访问 frame 对象，另外，在 IE 和 Firefox 中都可以使用 window.document.getElementById("frameId")来访问这个 frame 对象。

4. parentElement

【分析说明】IE 中支持使用 parentElement 和 parentNode 获取父节点。而 Firefox 只可以使用 parentNode。

【兼容处理】因为 firefox 与 IE 都支持 DOM，因此统一使用 parentNode 来访问父节点。

5. table 操作

【分析说明】IE 下 table 中无论是用 innerHTML 还是 appendChild 插入<tr>都没有效果，而其他浏览器却显示正常。

【兼容处理】解决的方法是，将<tr>加到 table 的<tbody>元素中，如下面所示:

```
var row = document.createElement("tr");
var cell = document.createElement("td");
var cell_text = document.createTextNode("插入的内容");
cell.appendChild(cell_text);
row.appendChild(cell);
document.getElementsByTagName("tbody")[0].appendChild(row);
```

6. 移除节点 removeNode()和 removeChild()

【分析说明】appendNode 在 IE 和 Firefox 下都能正常使用，但是 removeNode 只能在 IE 下用。

removeNode 方法的功能是删除一个节点，语法为 node.removeNode（false）或者 node.removeNode（true），返回值是被删除的节点。

removeNode（false）表示仅仅删除指定节点，然后这个节点的原孩子节点提升为原双亲节点的孩子节点。

removeNode（true）表示删除指定节点及其所有下属节点。被删除的节点成为了孤立节点，不再具有有孩子节点和双亲节点。

【兼容处理】Firefox 中节点没有 removeNode 方法，只能用 removeChild 方法代替，先回到父节点，在从父节点上移除要移除的节点。

```
node.parentNode.removeChild(node);  // 为了在 ie 和 firefox 下都能正常使用，取上一层的父结点，然后 remove。
```

7. childNodes 获取的节点

【分析说明】childNodes 的下标的含义在 IE 和 Firefox 中不同，看一下下面的代码:

```
<ul id="main">
<li>1</li>
<li>2</li>
<li>3</li>
</ul>
<input type=button value="clickme!"
onclick="alert(document.getElementById('main').childNodes.length)"
>
```

分别用 IE 和 Firefox 运行，IE 的结果是 3，而 Firefox 则是 7。Firefox 使用 DOM 规范，"#text"表示文本（实际是无意义的空格和换行等）在 Firefox 里也会被解析成一个节点，在 IE 里只有有实际意义的文本才会解析成"#text"。

【兼容处理】

方法一，获取子节点时，可以通过 node.getElementsByTagName()来回避这个问题。但是 getElementsByTagName 对复杂的 DOM 结构遍历明显不如用 childNodes，因为 childNodes 能更好的处理 DOM 的层次结构。

方法二，在实际运用中，Firefox 在遍历子节点时，不妨在 for 循环里加上:

```
if(childNode.nodeName=="#text") continue;//或者使用 nodeType == 1。
```

这样可以跳过一些文本节点。

延伸阅读

《IE 和 FireFox 中的 childNodes 区别》

8. Firefox 不能对 innerText 支持

【分析说明】Firefox 不支持 innerText，它支持 textContent 来实现 innerText，不过 textContent 没有像 innerText 一样考虑元素的 display 方式，所以不完全与 IE 兼容。如果不用 textContent，字符串里面不包含 HTML 代码也可以用 innerHTML 代替。也可以用 js 写个方法实现，可参考《为 firefox 实现 innerText 属性》一文。

【兼容处理】通过判断浏览器类型来兼容：

```
if(document.all){
document.getElementById('element').innerText = "my text";
} else{
document.getElementById('element').textContent = "my text";
}
```

四、事件处理

如果在使用 javascript 的时候涉及到 event 处理，就需要知道 event 在不同的浏览器中的差异，主要的 JavaScript 的事件模型有三种（参考《Supporting Three Event Models at Once》），它们分别是 NN4、IE4+和 W3C/Safar。

1. window.event

【分析说明】先看一段代码

```
function et()
{
alert(event);//IE: [object]
}
```

以上代码在 IE 运行的结果是[object]，而在 Firefox 无法运行。

因为在 IE 中 event 作为 window 对象的一个属性可以直接使用，但是在 Firefox 中却使用了 W3C 的模型，它是通过传参的方法来传播事件的，也就是说你需要为你的函数提供一个事件响应的接口。

【兼容处理】添加对 event 判断，根据浏览器的不同来得到正确的 event：

```
function et()
{
evt=evt?evt:(window.event?window.event:null);        //兼容 IE 和 Firefox
alert(evt);
}
```

2. 键盘值的取得

【分析说明】IE 和 Firefox 获取键盘值的方法不同，可以理解，Firefox 下的 event.which 与 IE 下的 event.keyCode 相当。关于彼此不同，可参考《键盘事件中 keyCode、which 和 charCode 的兼容性测试》

【兼容处理】

```
function myKeyPress(evt){ //兼容 IE 和 Firefox 获得 keyBoardEvent 对象
    evt = (evt) ? evt : ((window.event) ? window.event : "")    //兼容 IE 和 Firefox 获得 keyBoardEvent 对象的键值
    var key = evt.keyCode?evt.keyCode:evt.which;  if(evt.ctrlKey && (key == 13 || key == 10)){                //同时按下了 Ctrl 和回车键
        //do something;
    }
}
```

3. 事件源的获取

【分析说明】在使用事件委托的时候，通过事件源获取来判断事件到底来自哪个元素，但是，在 IE 下，event 对象有 srcElement 属性，但是没有 target 属性；Firefox 下，even 对象有 target 属性，但是没有 srcElement 属性。

【兼容处理】

```
ele=function(evt){ //捕获当前事件作用的对象
evt=evt||window.event;
                                                                return (obj=event.srcElement?event.srcElement:event.target;);
}
```

4. 事件监听

【分析说明】在事件监听处理方面，IE 提供了 attachEvent 和 detachEvent 两个接口，而 Firefox 提供的是 addEventListener 和 removeEventListener。

【兼容处理】最简单的兼容性处理就是封装这两套接口：

```
function addEvent(elem, eventName, handler) {
    if (elem.attachEvent) {
       elem.attachEvent("on"+eventName, function(){   handler.call(elem)});            //此处使用回调函数 call()，让 this 指向 elem
    } else if (elem.addEventListener) {
        elem.addEventListener(eventName, handler, false);
    }}
function removeEvent(elem, eventName, handler) {
    if (elem.detachEvent) {
         elem.detachEvent("on"+eventName, function(){handler.call(elem)});          //此处使用回调函数 call()，让 this 指向 elem
    } else if (elem.removeEventListener) {
        elem.removeEventListener(eventName, handler, false);
    }
}
```

需要特别注意，Firefox 下，事件处理函数中的 this 指向被监听元素本身，而在 IE 下则不然,可使用回调函数 call，让当前上下文指向监听的元素。

5. 鼠标位置

【分析说明】IE 下，even 对象有 x，y 属性，但是没有 pageX，pageY 属性；Firefox 下，even 对象有 pageX，pageY 属性，但是没有 x，y 属性。

【兼容处理】使用 mX(mX = event.x ? event.x : event.pageX;)来代替 IE 下的 event.x 或者 Firefox 下的 event.pageX。复杂点还要考虑绝对位置。

```
function getAbsPoint(e){
    var x = e.offsetLeft, y = e.offsetTop;
    while (e = e.offsetParent) {
        x += e.offsetLeft;
```

```
        y += e.offsetTop;
    }
    alert("x:" + x + "," + "y:" + y);
}
```

五、其他差异的兼容处理

1. XMLHttpRequest

【分析说明】new ActiveXObject("Microsoft.XMLHTTP");只在 IE 中起作用，Firefox 不支持，但支持 XMLHttpRequest。

【兼容处理】

```
function createXHR() {
    var xhr=null;
    if(window.XMLHttpRequest){
        xhr=new ActiveXObject("Msxml2.XMLHTTP");
    }else{
        try {
            xhr=new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch() {
        xhr=null;
    }
}
if(!xhr)return;
    return xhr;
}
```

2. 模态和非模态窗口

【分析说明】IE 中可以通过 showModalDialog 和 showModelessDialog 打开模态和非模态窗口，但是 Firefox 不支持。

【解决办法】直接使用 window.open(pageURL,name,parameters)方式打开新窗口。 如果需要传递参数，可以使用 frame 或者 iframe。

3. input.type 属性问题

IE 下 input.type 属性为只读，但是 Firefox 下可以修改

4. 对 select 元素的 option 操作

设置 options，IE 和 Firefox 写法不同：

Firefox:可直接设置

```
option.text = 'fooooooo';
```

IE:只能设置

```
option.innerHTML = 'fooooooo';
```

删除一个 select 的 option 的方法：

Firefox:可以

```
select.options.remove(selectedIndex);
```

IE7:可以用

```
select.options[i] = null;
```

IE6:需要写

```
select.options[i].outerHTML = null;
```

5. img 对象 alt 和 title 的解析

【分析说明】img 对象有 alt 和 title 两个属性，区别在于，alt：当照片不存在或者 load 错误时的提示。

title：照片的 tip 说明，在 IE 中如果没有定义 title，alt 也可以作为 img 的 tip 使用，但是在 Firefox 中，两者完全按照标准中的定义使用

在定义 img 对象时。

【兼容处理】最好将 alt 和 title 对象都写全，保证在各种浏览器中都能正常使用 。

6. img 的 src 刷新问题

【分析说明】先看一下代码：

```
<img id="pic" onclick= "this.src= 'a.jpg'" src="aa.jpg" style="cursor: pointer"/>
```

在 IE 下，这段代码可以用来刷新图片，但在 FireFox 下不行。主要是缓存问题。

【兼容处理】在地址后面加个随机数就解决了:

```
<img id="pic" onclick= "javascript:this.src=this.src+'?'+Math.random()"src="a.jpg" style="cursor: pointer"/>
```

总结

IE 和 Firefox 的 Javascript 方面存在着不少的差异，要做到兼容，我觉得很有必要把一些常见的整理成一个 js 库，如 DOM 的操 作，事件的处理，XMLHttpRequest 请求等，或者也可以选择使用现有的一些库(如 jQuery，YUI，ExtJs 等)，不过我觉得还是有必要 了解一下这些差异，这样对于我们参加兼容性和可用性代码很有帮助。

办法总比问题多，无论浏览器兼容如何折腾人，做前端开发的总能迎刃而解的!

一、什么是 deferred 对象?
开发网站的过程中，我们经常遇到某些耗时很长的 javascript 操作。其中，既有异步的操作（比如 ajax 读取服务器数据），也有同步的操作（比如遍历一个大型数组），它们都不是立即能得到结果的。
通常的做法是，为它们指定回调函数（callback）。即事先规定，一旦它们运行结束，应该调用哪些函数。
deferred 对象就是 jQuery 的回调函数解决方案
它解决了如何处理耗时操作的问题，对那些操作提供了更好的控制，以及统一的编程接口。
二、ajax 操作的链式写法

```
$.ajax("data.json")
.done(function(){consolel.log("ok")})
.fail(function(){console.log("error")})
```

done()相当于 success 方法，fail()相当于 error 方法。采用链式写法以后，代码的可读性大大提高。
三、指定同一操作的多个回调函数

```
$.ajax("test.html")
.done(function(){ alert("哈哈，成功了！");} )
.fail(function(){ alert("出错啦！"); } )
.done(function(){ alert("第二个回调函数！");} );
```

回调函数可以添加任意多个，它们按照添加顺序执行。

四、为多个操作指定回调函数
deferred 对象的另一大好处，就是它允许你为多个事件指定一个回调函数

```
$.when($.ajax("test1.html"), $.ajax("test2.html")
      .done(function(){ alert("哈哈，成功了！"); })
      .fail(function(){ alert("出错啦！"); }));
```

这段代码的意思是，先执行两个操作$.ajax("test1.html")和$.ajax("test2.html")，如果都成功了，就运行 done()指定的回调函数；如果有一个失败或都失败了，就执行 fail()指定的回调函数。

五、普通操作的回调函数接口（上）

deferred 对象的最大优点，就是它把这一套回调函数接口，从 ajax 操作扩展到了所有操作。也就是说，任何一个操作----不管是 ajax 操作还是本地操作，也不管是异步操作还是同步操作----都可以使用 deferred 对象的各种方法，

指定回调函数。

我们来看一个具体的例子。假定有一个很耗时的操作 wait：

```
var wait = function(){
    var tasks = function(){
        alert("执行完毕！");
    };
    setTimeout(tasks,5000);
};
```

我们为它指定回调函数，应该怎么做呢？

很自然的，你会想到，可以使用$.when()：

```
$.when(wait())
.done(function(){ alert("哈哈，成功了！"); })
.fail(function(){ alert("出错啦！"); });
```

（运行代码示例 5）

但是，这样写的话，done()方法会立即执行，起不到回调函数的作用。原因在于$.when()的参数只能是 deferred 对象，所以必须对 wait()进行改写：

```
var dtd = $.Deferred(); // 新建一个 deferred 对象
var wait = function(dtd){
    var tasks = function(){
        alert("执行完毕！");
        dtd.resolve(); // 改变 deferred 对象的执行状态
    };
    setTimeout(tasks,5000);
    return dtd;
};
```

现在，wait()函数返回的是 deferred 对象，这就可以加上链式操作了。

```
$.when(wait(dtd))
.done(function(){ alert("哈哈，成功了！"); })
.fail(function(){ alert("出错啦！"); });
```

（运行代码示例 6）

wait()函数运行完，就会自动运行 done()方法指定的回调函数。

六、deferred.resolve()方法和 deferred.reject()方法

deferred 对象有三种执行状态----未完成，已完成和已失败。如果执行状态是"已完成"（resolved）,deferred 对象立刻调用 done()方法指定的回调函数；如果执行状态是"已失败"，调用 fail()方法指定的回调函数；如果执行状态是"未完成"，则继续等待，或者调用 progress()方法指定的回调函数

七、deferred.promise()方法

上面这种写法，还是有问题。那就是 dtd 是一个全局对象，所以它的执行状态可以从外部改变。

```
var wait = function(dtd){
    var dtd = $.Deferred(); //在函数内部，新建一个 Deferred 对象
    var tasks = function(){
       alert("执行完毕！");
       dtd.resolve(); // 改变 Deferred 对象的执行状态
    };
    setTimeout(tasks,5000);
    return dtd.promise(); // 返回 promise 对象
};

$.when(wait())

.done(function(){ alert("哈哈，成功了！"); })
.fail(function(){ alert("出错啦！"); });
```

十、小结：deferred 对象的方法

（1） $.Deferred() 生成一个 deferred 对象。

（2） deferred.done() 指定操作成功时的回调函数

（3） deferred.fail() 指定操作失败时的回调函数

（4） deferred.promise() 没有参数时，返回一个新的 deferred 对象，该对象的运行状态无法被改变；接受参数时，作用为在参数对象上部署 deferred 接口。

（5） deferred.resolve() 手动改变 deferred 对象的运行状态为"已完成"，从而立即触发 done()方法。

（6）deferred.reject() 这个方法与 deferred.resolve()正好相反，调用后将 deferred 对象的运行状态变为"已失败"，从而立即触发 fail()方法。

（7） $.when() 为多个操作指定回调函数。

（8）deferred.then()

有时为了省事，可以把 done()和 fail()合在一起写，这就是 then()方法。

```
$.when($.ajax( "/main.php" ))

.then(successFunc, failureFunc );
```

如果 then()有两个参数，那么第一个参数是 done()方法的回调函数，第二个参数是 fail()方法的回调方法。如果 then()只有一个参数，那么等同于 done()。

（9）deferred.always()

这个方法也是用来指定回调函数的，它的作用是，不管调用的是 deferred.resolve()还是 deferred.reject()，最后总是执行。

```
$.ajax( "test.html" )

.always( function() { alert("已执行！");} );
```

十一、Javascript 异步编程的 4 种方法

javascript 语言的执行环境是“单线程”

所谓的单线程，每次执行只能一次任务。如果有多个任务的时候，必须等到前一个任务执行完毕的时候，在执行下一个任务。

好处：实现起来比较简单，执行的环境也相对比较简单

缺点：只要一个任务的耗时特别长，后面的任务都的必须排队等着，会拖延整个程序的执，,往往就是因为某一段 Javascript 代码长时间运行（比如死循环），导致整个页面卡在这个地方，其他任务无法执行。

为了解决单线程，Javascript 语言将任务的执行模式分成两种：同步（Synchronous）和异步（Asynchronous）。

"异步模式"非常重要。在浏览器端，耗时很长的操作都应该异步执行，避免浏览器失去响应，最好的例子就是 Ajax 操作。在服务器端，"异步模式"甚至是唯一的模式，因为执行环境是单线程的，如果允许同步执行所有 http 请求，服务器性能会急剧下降，很快就会失去响应。

方法：一、回调函数

```
f1() f2();
f1(f2());
```

采用这种方式，我们把同步操作变成了异步操作，f1 不会堵塞程序运行，相当于先执行程序的主要逻辑，将耗时的操作推迟执行。

回调函数的优点是简单、容易理解和部署，缺点是不利于代码的阅读和维护，各个部分之间高度耦合（Coupling），流程会很混乱，而且每个任务只能指定一个回调函数。

二、事件监听

另一种思路是采用事件驱动模式。任务的执行不取决于代码的顺序，而取决于某个事件是否发生。

还是以 f1 和 f2 为例。首先，为 f1 绑定一个事件（这里采用的 jQuery 的写法）。

```
f1.on('done', f2);
```

上面这行代码的意思是，当 f1 发生 done 事件，就执行 f2。然后，对 f1 进行改写：

```
function f1(){
    setTimeout(function () {
        // f1 的任务代码
        f1.trigger('done');
    }, 1000);
}
```

f1.trigger('done')表示，执行完成后，立即触发 done 事件，从而开始执行 f2。

这种方法的优点是比较容易理解，可以绑定多个事件，每个事件可以指定多个回调函数，而且可以"去耦合"（Decoupling），有利于实现模块化。缺点是整个程序都要变成事件驱动型，运行流程会变得很不清晰。

三、发布/订阅

观察者模式又叫发布订阅模式（Publish/Subscribe），它定义了一种一对多的关系，让多个观察者对象同时监听某一个主题对象，这个主题对象的状态发生变化时就会通知所有的观察者对象，使得它们能够自动更新自己。

观察者的使用场合就是：当一个对象的改变需要同时改变其它对象，并且它不知道具体有多少对象需要改变的时候，就应该考虑使用观察者模式。

总的来说，观察者模式所做的工作就是在解耦，让耦合的双方都依赖于抽象，而不是依赖于具体。从而使得各自的变化都不会影响到另一边的变化。

四、Promises 对象

Promises 对象是 CommonJS 工作组提出的一种规范，目的是为异步编程提供统一接口。

每一个异步任务返回一个 Promise 对象，该对象有一个 then 方法，允许指定回调函数。比如，f1 的回调函数 f2,可以写成：

```
f1().then(f2);

function f1(){
    var dfd = $.Deferred();
    setTimeout(function () {
        // f1 的任务代码
        dfd.resolve();
    }, 500);
    return dfd.promise;
}
```

这样写的优点在于，回调函数变成了链式写法，程序的流程可以看得很清楚，而且有一整套的配套方法，可以实现许多强大的功能。

比如，指定多个回调函数：

```
f1().then(f2).then(f3);
```

再比如，指定发生错误时的回调函数：

```
f1().then(f2).fail(f3);
```

而且，它还有一个前面三种方法都没有的好处：如果一个任务已经完成，再添加回调函数，该回调函数会立即执行。所以，你不用担心是否错过了某个事件或信号。

这种方法的缺点就是编写和理解，都相对比较难。

十二、

琐碎

1、js 有那些数据类型

javascript 中包含 6 种数据类型：undefined、null、string、number、boolean 和 object。其中，前 5 种是原始数据类型，object 是对象类型。object 类型中包括 Object、Function、 String、Number、Boolean、Array、Regexp、Date、Globel、Math、Error，以及宿主环境提供的 object 类型。

2、规避 javascript 多人开发函数重名问题

(1) 可以开发前规定命名规范， 根据不同开发人员开发的功能在函数前加前缀

(2) 将每个开发人员的 函数封装到类中，调用的时候就调用类的函数，即使函数重名只要类名不重复就 ok

3，FF 下面实现 outerHTML

FF 不支持 outerHTML，要实现 outerHTML 还需要特殊处理

思路如下：

在页面中添加一个新的元素 A，克隆一份需要获取 outerHTML 的元素，将这个元素 append 到新的 A 中，然后获取 A 的 innerHTML 就可以了。

view source

print?

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>获取 outerHMTL</title>
<style>
div{ background:#0000FF;width:100px;height:100px;}
span{ background:#00FF00;width:100px;height:100px;}
p{ background:#FF0000;width:100px;height:100px;}
</style>
</head>
<body>
<div id="a"><span>SPAN</span>DIV</div>
<span>SPAN</span>
<p>P</p>
<script type="text/javascript">
function getOuterHTML(id){
var el = document.getElementById(id);
var newNode = document.createElement("div");
document.appendChild(newNode);
var clone = el.cloneNode(true);
newNode.appendChild(clone);
alert(newNode.innerHTML);
document.removeChild(newNode);
}
getOuterHTML("a");
</script>
</body>
</html>
```

4、编写一个方法 求一个字符串的字节长度

假设：

一个英文字符占用一个字节，一个中文字符占用两个字节

view source

print?

```
function GetBytes(str){
var len = str.length;
var bytes = len;
for(var i=0; i<len; i++){
if (str.charCodeAt(i) > 255) bytes++;
```

```
6 }
7 return bytes;
8 }
9 alert(GetBytes("你好,as"));
```

5，写出 3 个使用 this 的典型应用

（1）在 html 元素事件属性中使用，如

view source

print?

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

（2）构造函数

view source

print?

```
1 function Animal(name, color) {
2 this.name = name;
3 this.color = color;
4 }
```

（3）

view source

print?

```
1 <input type="button" id="text" value="点击一下" />
2 <script type="text/javascript">
3 var btn = document.getElementById("text");
4 btn.onclick = function() {
5 alert(this.value); //此处的 this 是按钮元素
6 }
7 </script>
```

（4）CSS expression 表达式中使用 this 关键字

view source

print?

```
1 <table width="100px" height="100px">
2 <tr>
3 <td>
4 <div style="width:expression(this.parentNode.width);">div element</div>
5 </td>
6 </tr>
7 </table>
```

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

String 类型有两种生成方式：

view source

print?

```
01 (1)Var str = “hello world”;
02
03
04 (2)Var str2 = new String(“hello world”);
05
06
07 function IsString(str){
08 return (typeof str == "string" || str.constructor == String);
09 }
10 var str = "";
11 alert(IsString(1));
12 alert(IsString(str));
13 alert(IsString(new String(str)));
```

14，网页中实现一个计算当年还剩多少时间的倒数计时程序，要求网页上实时动态显示“××年还剩××天××时××分××秒”

view source

print?

```
01 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://ww
02 <html>
03 <head>
04 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
05 <title>倒计时</title>
06 </head>
07 <body>
08 <input type="text" value="" id="input" size="1000"/>
09 <script type="text/javascript">
10 function counter() {
11 var date = new Date();
12 var year = date.getFullYear();
13 var date2 = new Date(year, 12, 31, 23, 59, 59);
14 var time = (date2 - date)/1000;
15 var day =Math.floor(time/(24*60*60))
16 var hour = Math.floor(time%(24*60*60)/(60*60))
17 var minute = Math.floor(time%(24*60*60)%(60*60)/60);
18 var second = Math.floor(time%(24*60*60)%(60*60)%60);
19 var str = year + "年还剩"+day+"天"+hour+"时"+minute+"分"+second+"秒";
20 document.getElementById("input").value = str;
21 }
22 window.setInterval("counter()", 1000);
23 </script>
24 </body>
25 </html>
```

15，补充代码，鼠标单击 Button1 后将 Button1 移动到 Button2 的后面

view source

print?

```
01 <div><input type=”button” id=”button1” value=”1” onclick=”???”><i
   value=”2” /></div>
02
03
04
05 <div>
06 <input type="button" id ="button1" value="1" onclick="moveBtn(this);">
07 <input type="button" id ="button2" value="2" />
08 </div>
09 <script type="text/javascript">
10 function moveBtn(obj) {
11 var clone = obj.cloneNode(true);
12 var parent = obj.parentNode;
13 parent.appendChild(clone);
14 parent.removeChild(obj);
15 }
16 </script>
```

16、JavaScript 中如何对一个对象进行深度 clone

view source

print?

```
01  function cloneObject(o) {
02  if(!o || 'object' !== typeof o) {
03  return o;
04  }
05  var c = 'function' === typeof o.pop ? [] : {};
06  var p, v;
07  for(p in o) {
08  if(o.hasOwnProperty(p)) {
09  v = o[p];
10  if(v && 'object' === typeof v) {
11  c[p] = Ext.ux.clone(v);
12  }
13  else {
14  c[p] = v;
15  }
16  }
17  }
18  return c;
19  };
```

26，js 中如何定义 class,如何扩展 prototype?

Ele.className = “***”; //***在 css 中定义，形式如下：.*** {…}
A.prototype.B = C;
A 是某个构造函数的名字
B 是这个构造函数的属性
C 是想要定义的属性的值

27，如何添加 html 元素的事件,有几种方法.
（1） 为 HTML 元素的事件属性赋值
（2） 在 JS 中使用 ele.on*** = function() {…}
（3） 使用 DOM2 的添加事件的方法 addEventListener 或 attachEvent

28，documen.write 和 innerHTML 的区别
document.write 只能重绘整个页面
innerHTML 可以重绘页面的一部分

29，多浏览器检测通过什么？
（1） navigator.userAgent
（2） 不同浏览器的特性，如 addEventListener

30，js 的基础对象有那些，window 和 document 的常用的方法和属性列出来
String,Number,Boolean
Window:
方法：
setInterval,setTimeout,clearInterval,clearTimeout,alert,confirm,open
属性：name,parent,screenLeft,screenTop,self,top,status
Document
方法：
createElement,execCommand,getElementById,getElementsByName,getElementByTagName,write,writeln
属性：cookie,doctype,domain,documentElement,readyState,URL,

31，前端开发的优化问题
（1） 减少 http 请求次数：css spirit,data uri
（2） JS，CSS 源码压缩
（3） 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
（4） 用 innerHTML 代替 DOM 操作，减少 DOM 操作次数，优化 javascript 性能
（5） 用 setTimeout 来避免页面失去响应
（6） 用 hash-table 来优化查找
（7） 当需要设置的样式很多时设置 className 而不是直接操作 style
（8） 少用全局变量
（9） 缓存 DOM 节点查找的结果
（10） 避免使用 CSS Expression
（11） 图片预载
（12） 避免在页面的主体布局中使用 table，table 要等其中的内容完全下载之后才会显示出来，显示比 div+css 布局慢

32，如何控制网页在网络传输过程中的数据量
启用 GZIP 压缩
保持良好的编程习惯，避免重复的 CSS，JavaScript 代码，多余的 HTML 标签和属性
33，Flash、Ajax 各自的优缺点，在使用中如何取舍？
Ajax 的优势
（1） 可搜索型
（2） 开放性
（3） 费用
（4） 易用性
（5） 易于开发
Flash 的优势
（1） 多媒体处理
（2） 兼容性
（3） 矢量图形 比 SVG，Canvas 优势大很多
（4） 客户端资源调度，比如麦克风，摄像头
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
一、你能描述一下渐进增强和优雅降级之间的不同吗?
如果提到了特性检测，可以加分。
检测浏览器，渐进增强就是让牛 b 的浏览器的效果更好，优雅降级就是让 2b 的浏览器在功能 ok 的情况下效果一般。
二、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。
三、请解释一下什么是“语义化的 HTML”。
语义化的好处：1：去掉或样式丢失的时候能让页面呈现清晰的结构：
html 本身是没有表现的，我们看到例如<h1>是粗体，字体大小 2em，加粗；是加粗的，不要认为这是 html 的表现，这些其实 html 默认的 css 样式在起作用，所以去掉或样式丢失的时候能让页面呈现清晰的结构不是语义化的 HTML 结构的优点，但是浏览器都有有默认样式，默认样式的目的也是为了更好的表达 html 的语义，可以说浏览器的默认样式和语义化的 HTML 结构是不可分割的。
2.屏幕阅读器（如果访客有视障）会完全根据你的标记来“读”你的网页.
3.PDA、手机等设备可能无法像普通电脑的浏览器一样来渲染网页（通常是因为这些设备对 CSS 的支持较弱）.
4.搜索引擎的爬虫也依赖于标记来确定上下文和各个关键字的权重.
5.你的页面是否对爬虫容易理解非常重要,因为爬虫很大程度上会忽略用于表现的标记,而只注重语义标记.
6.便于团队开发和维护

语义化的 HTML 就是：标题用 h1-h6，文字段落用 p，列表用 ul li，大致如此
四、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

五、为什么利用多个域名来提供网站资源会更有效?
浏览器同一时间可以从一个域名下载多少资源?

你的浏览器能同时保持对一个域名的多少连接?

显示当前主要浏览的这个参数限制：

更多详情：浏览器默认的并发数限制

三个最主流的原因:
1. CDN 缓存更方便
2. 突破浏览器并发限制（你随便挑一个 G 家的 url:
https://lh4.googleusercontent.com/-si4dh2myPWk/T81YkSi__AI/AAAAAAAAQ5o/L1wbBRpp58Q/w497-h373/IMG_20120603_163233.jpg, 把前面的 lh4 换成 lh3,lh6 啥的，都照样能够访问，像地图之类的需要大量并发下载图片的站点，这个非常重要。）
3. Cookieless, 节省带宽，尤其是上行带宽 一般比下行要慢。。。
还有另外两个非常规原因:
4. 对于 UGC 的内容和主站隔离，防止不必要的安全问题(上传 js 窃取主站 cookie 之类的) 。
正是这个原因要求用户内容的域名必须不是自己主站的子域名，而是一个完全独立的第三方域名。
5. 数据做了划分，甚至切到了不同的物理集群，通过子域名来分流比较省事. ^_^ 这个可能被用的不多。
PS: 关于 Cookie 的问题，带宽是次要的，安全隔离才是主要的。
关于多域名，也不是越多越好，虽然服务器端可以做泛解释，浏览器做 dns 解释也是耗时间的，而且太多域名，如果要走 https 的话，还有要多买证书和部署的问题，^_^。
六、请说出三种减少页面加载时间的方法。（加载时间指感知的时间或者实际加载时间）
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）。
七、如果你参与到一个项目中，发现他们使用 Tab 来缩进代码，但是你喜欢空格，你会怎么做？
1.建议这个项目使用像 EditorConfig (http://editorconfig.org/) 之类的规范
2.为了保持一致性，接受项目原有的风格
3.直接使用 VIM 的 retab 命令
八、请写一个简单的幻灯效果页面
如果不使用 JS 来完成，可以加分。（如：纯 CSS 实现的幻灯片效果）
九、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
十、如果今年你打算熟练掌握一项新技术，那会是什么？
nodejs，html5，css3，less
十一、请谈一下你对网页标准和标准制定机构重要性的理解。
(google)w3c 存在的意义就是让浏览器兼容性问题尽量小，首先是他们对浏览器开发者的约束，然后是对开发者的约束。
十二、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。
HTML 相关问题：
十三、doctype（文档类型）的作用是什么？你知道多少种文档类型？
此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范。
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。
十四、浏览器标准模式和怪异模式之间的区别是什么？
W3C 标准推出以后，浏览器都开始采纳新标准，但存在一个问题就是如何保证旧的网页还能继续浏览，在标准出来以前，很多页面都是根据旧的渲染方法编写的，如果用的标准来渲染，将导致页面显示异常。为保持浏览器渲染的兼容性，使以前的页面能够正常浏览，浏览器都保留了旧的渲染方法（如：微软的 IE）。这样浏览器渲染上就产生了 Quircks mode 和 Standars mode，两种渲染方法共存在一个浏览器上。
IE 盒子模型和标准 W3C 盒子模型：ie 的 width 包括：padding\border。 标准的 width 不包括：padding\border
在 js 中如何判断当前浏览器正在以何种方式解析？
document 对象有个属性 compatMode ,它有两个值：
BackCompat 对应 quirks mode
CSS1Compat 对应 strict mode
十五、使用 XHTML 的局限有哪些？
xhtml 要求严格，必须有 head、body 每个 dom 必须要闭合。
如果页面使用'application/xhtml+xml'会有什么问题吗？
一些老的浏览器并不兼容。
十六、如果网页内容需要支持多语言，你会怎么做？
编码 UTF-8，空间域名需要支持多浏览地址。
在设计和开发多语言网站时，有哪些问题你必须要考虑？
1、应用字符集的选择 2、语言书写习惯&导航结构 3、数据库驱动型网站
十七、data-属性的作用是什么？
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.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。
十八、如果把 HTML5 看作做一个开放平台，那它的构建模块有哪些？
<nav>, <header>,<section>, <footer>
相关阅读：必须知道的七个 HTML5 新特性
十九、请描述一下 cookies，sessionStorage 和 localStorage 的区别？
sessionStorage 和 localStorage 是 HTML5 Web Storage API 提供的，可以方便的在 web 请求之间保存数据。有了本地数据，就可以避免数据在浏览器和服务器间不必要地来回传递。
sessionStorage、localStorage、cookie 都是在浏览器端存储的数据，其中 sessionStorage 的概念很特别，引入了一个“浏览器窗口”的概念。
sessionStorage 是在同源的同窗口（或 tab）中，始终存在的数据。也就是说只要这个浏览器窗口没有关闭，即使刷新页面或进入同源另一页面，数据仍然存在。关闭窗口后，sessionStorage 即被销毁。同时“独立”打开的不同窗口，即使是同一页面，sessionStorage 对象也是不同的
cookies 会发送到服务器端。其余两个不会。
Microsoft 指出 InternetExplorer8 增加 cookie 限制为每个域名 50 个，但 IE7 似乎也允许每个域名 50 个 cookie。
Firefox 每个域名 cookie 限制为 50 个。
Opera 每个域名 cookie 限制为 30 个。
Firefox 和 Safari 允许 cookie 多达 4097 个字节，包括名(name)、值(value)和等号。
Opera 允许 cookie 多达 4096 个字节，包括：名（name）、值（value）和等号。
InternetExplorer 允许 cookie 多达 4095 个字节，包括：名（name）、值（value）和等号。
CSS 相关问题：
二十、描述下 “reset” CSS 文件的作用和使用它的好处。
因为浏览器的品种很多，每个浏览器的默认样式也是不同的，所以定义一个 css reset 可以使各浏览器的默认样式统一。
二十一、解释下浮动和它的工作原理。

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。
二十二、列举不同的清除浮动的技巧，并指出它们各自适用的使用场景。

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；二、content 属性是必须的，但其值可以为空，蓝色理想讨论该方法的时候 content 属性的值设为”.”，但我发现为空亦是可以的。

view source

print?

```
1 *{margin:0;padding:0;}
2 body{font:36px bold; color:#F00; text-align:center;}
3 #layout{background:#FF9;}
4 #layout:after{display:block;clear:both;content:"";visibility:hidden;height:0;}
5 #left{float:left;width:20%;height:200px;background:#DDD;line-height:200px;}
6 #right{float:right;width:30%;height:80px;background:#DDD;line-height:80px;}
<div id=" layout" >
 <div id=" left" >Left</div>
 <div id=" right" >Right</div>
 </div>
```

此三种方法各有利弊，使用时应择优选择，比较之下第二种方法更为可取

二十三、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。
二十四、你最喜欢的图片替换方法是什么，你如何选择使用。
<h2> <span 图片丢这里>Hello World </h2> 把 span 背景设成文字内容，这样又可以保证 seo，也有图片的效果在上面。
 一般都是：alt，title，onerror
二十五、讨论 CSS hacks，条件引用或者其他。
background-color:blue; 各个浏览器都认识，这里给 firefox 用：
 background-color:red\9;\9 所有的 ie 浏览器可识别；
 background-color:yellow\0; \0 是留给 ie8 的
 +background-color:pink; + ie7 定了；
 _background-color:orange; _专门留给神奇的 ie6；
 :root #test { background-color:purple\9; } :root 是给 ie9 的，
 @media all and (min-width:0px){ #test {background-color:black\0;} }
这个是老是跟 ie 抢着认\0 的神奇的 opera，必须加个\0,不然 firefox，chrome，safari 也都认识。。。
 @media screen and (-webkit-min-device-pixel-ratio:0){ #test {background-color:gray;} }最后这个是浏览器新贵 chrome 和 safari 的。
二十六、如何为有功能限制的浏览器提供网页？你会使用哪些技术和处理方法？
百度 谷歌 SO SOGOU Bing
二十七、如何视觉隐藏网页内容，只让它们在屏幕阅读器中可用？
1.display:none;的缺陷
 搜索引擎可能认为被隐藏的文字属于垃圾信息而被忽略
 屏幕阅读器（是为视觉上有障碍的人设计的读取屏幕内容的程序）会忽略被隐藏的文字。
 2. visibility: hidden ;的缺陷
 这个大家应该比较熟悉就是隐藏的内容会占据他所应该占据物理空间
 3.overflow:hidden;一个比较合理的方法
 .texthidden { display:block;/*统一转化为块级元素*/ overflow: hidden; width: 0; height: 0; }
 就像上面的一段 CSS 所展示的方法，将宽度和高度设定为 0，然后超过部分隐藏，就会弥补上述一、二方法中的缺陷，也达到了隐藏内容的目的。
二十八、你用过栅格系统吗？如果使用过，你最喜欢哪种？
比如：Bootstrap，流式栅格系统，http://960.gs/ ，栅格系统延续美学
29、你用过媒体查询，或针对移动端的布局/CSS 吗？
view source

print?

```
1 @media screen and (min-width: 400px) and (max-width: 700px) { … }
2 @media handheld and (min-width: 20em), screen and (min-width: 20em) { … }
```

媒体查询，就是响应式布局。

30、你熟悉 SVG 样式的书写吗？
view source

print?

```
1 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999
2 <circle cx="40" cy="40" r="24" style="stroke:#006600; fill:#00cc00"/>
3 <text x="250" y="150" font-family="Verdana" font-size="10px" fill="blue">He
4 <defs><style type="text/css"> <![CDATA[.sample{stroke:blue;fill:#0080FF;opa
5 <use xlink:href="#sample1" class="sample"/>
6 </svg>
```

31、如何优化网页的打印样式？
 <link rel="stylesheet" type="text/css" media="screen" href="xxx.css" />
 其中 media 指定的属性就是设备，显示器上就是 screen，打印机则是 print，电视是 tv，投影仪是 projection。
 <link rel="stylesheet" type="text/css" media="print" href="yyy.css" />
 设计打印样式表也应有些注意事项：
 1、打印样式表中最好不要用背景图片，因为打印机不能打印 CSS 中的背景。如要显示图片，请使用 html 插入到页面中。
 2、最好不要使用像素作为单位，因为打印样式表要打印出来的会是实物，所以建议使用 pt 和 cm。
 3、隐藏掉不必要的内容。（@print div{display:none;}）
 4、打印样式表中最好少用浮动属性，因为它们会消失。
 如果想要知道打印样式表的效果如何，直接在浏览器上选择打印预览就可以了。
32、在书写高效 CSS 时会有哪些问题需要考虑？
1.样式是：从右向左的解析一个选择器
 2.ID 最快，Universal 最慢 有四种类型的 key selector，解析速度由快到慢依次是：ID、class、tag 和 universal
 3.不要 tag-qualify （永远不要这样做 ul#main-navigation { } ID 已经是唯一的，不需要 Tag 来标识，这样做会让选择器变慢。）
 4.后代选择器最糟糕（换句话说，下面这个选择器是很低效的： html body ul li a { }）
 5.想清楚你为什么这样写
 6.CSS3 的效率问题（CSS3 选择器（比如 :nth-child）能够漂亮的定位我们想要的元素，又能保证我们的 CSS 整洁易读。但是这些神奇的选择器会浪费很多的浏览器资源。）
 7.我们知道#ID 速度是最快的，那么我们都用 ID，是不是很快。但是我们不应该为了效率而牺牲可读性和可维护性
33.使用 CSS 预处理器的优缺点有哪些？
(SASS，Compass，Stylus，LESS)
 描述下你曾经使用过的 CSS 预处理的优缺点
34.如果设计中使用了非标准的字体，你该如何去实现？
 Webfonts（字体服务例如：Google Webfonts，Typekit 等等。）
35.解释下浏览器是如何判断元素是否匹配某个 CSS 选择器？
36.解释一下你对盒模型的理解，以及如何在 CSS 中告诉浏览器使用不同的盒模型来渲染你的布局。
相关阅读：前端人员必须知道的 10 个 CSS3 属性（附例子）
JS 相关问题：
37、解释下事件代理。
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。
 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 JavaSciprt 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;
 }
```

38、解释下 JavaScript 中 this 是如何工作的。

this 永远指向函数运行时所在的对象，而不是函数被创建时所在的对象。
匿名函数或不处于任何对象中的函数指向 window
1.如果是 call，apply,with，指定的 this 是谁，就是谁
2.普通的函数调用，函数被谁调用，this 就是谁
39、解释下原型继承的原理。

以下代码展示了 JS 引擎如何查找属性：

view source

print?

```
1 function getProperty(obj, prop) {
2 if (obj.hasOwnProperty(prop))
3 return obj[prop]
4 else if (obj.__proto__ !== null)
5 return getProperty(obj.__proto__, prop)
6 else
7 return undefined
8 }
```

ios 和安卓兼容问题

一、搜索确定问题

添加 form 元素，在提交的时候是 input 失去焦点

二、时间框选择问题

添加 form 元素

三、多图上传问题

安卓上不能多图上传，无法解决

四、浮动问题

尽量用盒子模型布局

五、音频自动播放问题，ios 默认不自动播放

在 document 上添加点击事件播放音频

六、浮动高度撑开盒子

```
.clearfix:after{
content:".";
display:block;
height:0;
clear:both;
visibility:hidden;
}
.clearfix{
display:inline-block;
}
* html .clearfix{
height:1%;
}
.clearfix{
display:block;
}
.clear{
clear:both;
height:0;
font:0/0 Arial;
visibility:hidden;
}
```

七、 Css 在 ios 中动画闪屏问题

IOS 下 Safari 渲染 Transition 时页面闪动 Bug

http://classjs.com/category/technology/css/

环境：IOS 的 Safari 环境

问题：在做移动端左右滑动的时候，用到了 CSS3 的 Transition 属性来进行动画变换，结果每次渲染 Transition 属性时，出现闪屏现象。

解决办法：在网上查了一下，有一下两种解决办法，

方法一：

```
1 -webkit-transform-style: preserve-3d;
2 /*设置内嵌的元素在 3D 空间如何呈现：保留 3D*/
```

方法二：

view source print?

```
1 -webkit-backface-visibility: hidden;
2 /*（设置进行转换的元素的背面在面对用户时是否可见：隐藏）*/
```

八、单屏网页滑动问题

添加

```
$( document ).on("touchmove",function(e){
    e.stopPropagation();
    return false;
})
```

九、键盘弹出问题

解决办法：无，安卓键盘弹出整个页面的 window 层的高度减小，ios 无影响；

十：伪类 active 失效

要 CSS 伪类 :active 生效，只需要给 document 绑定 touchstart 或 touchend 事件

使用 css3 动画用了 transition 或者 animation 后会有闪白的现象

-webkit-backface-visibility: hidden;

尽量写成 3d 使页面运行更流畅

-webkit-transform-style: preserve-3d;

ios 横屏时会重置字体大小

text-size-adjust：none

手机上最好是用 tap 事件 click 事件会有 300ms 的延迟

禁止 ios 弹出各种操作窗口

-webkit-touch-callout:none

禁止用户选中文字

-webkit-user-select:none

js 动态生成的下拉菜单在安卓 2.0 中不起作用

解决方法：删除了 overflow-x:hidden;然后在 js 生成下拉菜单之后聚焦 focus

消除 IE10 里面的那个叉号

input:-ms-clear{display:none;}

ios 中文输入法输入英文时会有小空格 解决办法 用正则

this.value = this.value.replace(/\u2006/g, '');

三星 I9100 （Android 4.0.4）不支持 display:-webkit-flex 这种写法的弹性布局，

但支持 display:-webkit-box 这种写法的布局，

相关的属性也要相应修改，如-webkit-box-pack: center;

移动端采用弹性布局时，建议直接写 display:-webkit-box 系列的布局

touchmove 事件在 Android 部分机型(如 LG Nexus 5 android4.4，小米 2 android 4.1.1)上只会触发一次

解决方案是在触发函数里面加上 e.preventDefault(); 记得将 e 也传进去。

图片圆角是显示不正常 在图片外面包裹 一个元素 然后给那个元素设置圆角 让图片变成这个元素的背景图片

在 android4.2 背景会溢出，使用 background-clip:padding-box;

ios 不会弹出键盘 必须满足下面几点

① 文本框获取焦点

② 手指触屏（网页区域，混合开发触屏 app 头不能让 webview 弹出键盘）

③ 没有延迟（不会 ajax 回调，不会延迟）

禁止链接高亮

-webkit-tap-highlight-color:rgba(0,0,0,0)

禁止链接长按弹出选项菜单

-webkit-touch-callout:none;

阻止屏幕旋转时自动调整字体大小

-webkit-text-size-adjust:none;

田俊杰 19:04:16

div 的垂直居中问题

设置行高和 div 一样的高度 vertical-align:middle;

margin 加倍的问题

设置 display：inline 或者设置宽度的一半

浮动 ie 产生的双倍距离

display:inline;

图片产生 3px 的距离

父元素设置 font-size:0

内容不显示

给元素固定的宽高

显示多的内容出来

删除注释

无法定义 1px 左右高度的容器

line-height:1px; overflow:hidden;

超链接访问过后 hover 样式就不再出现的问题

改 变 css 的 排 列 顺序 l-v-h-a link-visited-hover-active

ul 在火狐浏览器有 padding 值 在 IE 中有 margin 值

margin：0；padding：0；

ff 无法撑开容器的高度

去掉高度 设置最小高度为 min-height:200px;

自动换行：word-wrap:break-word

内容显示在一行：write-space:nowrap

为什么 IE6 下容器的宽高和 ff 不同

因为 IE 把 border 也算在内 ff 没算 border 的值

1.ajax 原理

它相当于在用户和服务器之间加了一个中间层，使用户操作和服务器响应异步化。这样就会把之前的一些服务器负担的工作转嫁到客户端，减轻服务器的负担！简单地说：就是通过 XmlHttpRequest 对象来向服务器发送异步请求，从服务器获得并下载数据到客户端，然后用 JavaScript 技术来操作 DOM 而更新页面。其核心是 JavaScript 对象 XmlHttpRequest，它是一种支持异步请求的技术，使您使用 JavaScript 向服务器提出请求并处理响应，而不至于阻塞用户！

2.渲染

页面渲染就是浏览器将请求返回的页面资源（如 html 文本，图像，动画等）基于一定的规则（css 语句，js 语句）完成页面布局及绘制的过程。

用户通过在浏览器中执行某种动作，向服务器发送请求，服务器对发送的请求进行处理并解析，把相应的数据提取出来，加载到客户端页面中。实现无刷新更新页面功能。在这个过程中，主要借助于 ajax 技术进行渲染数据，操作的对象主要是 json 数据。

常用到的技术有: $.ajax() $.getJSON();

常用到的转换方法：

将一个对象转化为 json 对象：json.encode() eval()

其中 eval()函数，把字符串转换为 JavaScript 功能代码

Json 对象转换为 json 字符串：json.stringify();

Json 字符串转换为 json 对象：json.parse();

3.h4 和 h5 的区别

- <canvas>标签替代 Flash
- 新增加了一些语义化标签

  Header footer aside section article nav addresstime
- 新增了表单属性

  如 url,date,range,tel,num, color,email...
- 简化的语法

4.闭包

简单地说，JavaScript 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他们所在的外部函数中声明的所有局部变量、参数和声明的其它内部函数。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。它是一个定义在函数中的函数。它可以访问别的函数的变量和属性。

使用闭包的优点：

① 可以读取函数内部变量

② 使变量始终保持在内存中

③ 不增加额外的全局变量

④ 执行过程中所有变量都在函数内部

闭包的缺点：

i. 变量始终保存在内存中，内存消耗很大

ii. Ie 中会导致内存泄露

iii. 会在父函数外部改变函数内部变量的值

5.数据类型

基本数据类型：

数值型：number 字符型：string 布 尔 型 ：boolean

未定义：undefined 空值：null 对象：object

6.循环

- for(var i=0;i<100;i++){ 遍历对象中的每一项
  }
- do{ 先执行一次代码，再进行判断
  }while()
- While(){ //先进行判断
  }
- for(var i in arr){ // i 代表索引
  }
- foreach(arr as $k=>$v){ //php 中进行的遍历
  }

7.DOM 节点操作

- 节点属性

  Nodetype 元素 1 属性 2文本 3 页面 9

  Nodename

  Nodevalue
- 查找节点

  Document.getElementById()

  Document.getElementsByClassName()

  Document.getElementByTagName()

  Document.getElementByName()

  Document.forms 获取所有 form 元素

  Document.querySelector(“.class”) //查找第一个

  Document.querySelectorAll(“.class”) //查找所有

  Document.body //查找 body

  Document.documentElement //查找 html 元素
- 遍历节点

  父节点.children //查找所有元素子节点

  子节点：childNodes //查找所有子节点，包括文本节点

  firstchild lastchild

  兄弟节点：previousSibling nextSibling

  父节点：parentNode
- 操作节点

  添加节点

  添加节点：父节点.appendChild() 末尾添加

  任意位置添加：父节点.insertBefore(新节点，指定节点)

  创建节点

  document.createElement(“标签名”)

  document.createTextNode(“文本内容”)

  删除节点

  父节点.removeChild()

  替换节点

  父节点.replaceChild(新节点，旧节点)

  复制节点

  节点.cloneNode(true/false)

8.DOM 属性操作

- 标准属性

  获取：对象.属性名

  设置：对象.属性名=值

  注意：class 属性必须通过“对象.classname=值”来操作
- 自定义属性 data_属性名=””

  获取：对象.getAttribute(“属性名”)

  设置：对象.setAttribute(“属性名”,”属性值”)

  删除：对象.removeAttribute(“属性名”)

  注意：以上方法也适用于标准属性

9.DOM 表格操作

- 获取表格元素

  表格对象.tBodies

  Table/thead/tbody/tfoot.rows

  Table/thead/tbody/tfoot/tr.cells
- 表格的添加删除

  添加：

  行：insertRow(下标)

  列:insertCell(下标)

  删除：

  deleteRow(下标)

  deleteCell(下标)

10.尺寸和位置

元素大小： padding+border+width/height

offsetWidth

offsetHeight

元素位置：相对于父元素

offsetLeft (定位)left+marginLeft

offsetTop ((定位)top+marginTop)

客户端尺寸

clientWidth=width+padding

clientHeight=height+padding

解决兼容：

document.documentElement.clientWidth||document.body.clientWidth

滚动尺寸 （整个文档的尺寸）

Scrollwidth scrollheight

滚动距离

ScrollTop scrollLeft

document.documentElement.ScrollTop||document.body.ScrollTop

窗口位置：兼容

Window.screenX window.screenY “”+px

window.screenLeft window.screenTop number

处理兼容：

Var leftpo=(typeof window.screenLeft== ”number”)?screenLeft:screenX

窗口移动：

moveTo(x,y)

moveBy(x,y)

窗口大小：

Window.outerHeight

window.outerWidth

可视区大小：

window.innerHeight;

window.innerWidth

可视区的宽度和高度：

document.documentElement.clientWidth

document.documentElement.clientHeight

document.body.clientWidth

Document.body.clientHeight

窗口的打开关闭

ss=Window.open(“url”,”名字”,”宽，高，left’，top”)

ss.Window.close() //关闭弹出框

Ss.close() //关闭本身窗口

Screen.width

screen.height

Location.href=”” 超链接跳转

History.go(1) history.foward() history.go(-1) history.back()

11.表单的操作

获取表单元素

Form.elements //获取所有表单元素

表单的 name.表单元素的 name

组合式

提交表单

提交按钮

图像按钮

自定义提交按钮<button></button>

脚本实现：form 对象.submit()

重置表单

重置按钮

自定义提交按钮<button></button>

脚本实现：form 对象.reset()

下拉菜单

获取所有 option：select 对象.options

通过 multiple 和 size 来指定多选和行数

添加 option

Select 对象.Add(newoption,reloption)

Select 对象.Add(newoption,undefined) //末端插入

删除 option

单个：

Remove(index)

Select 对象.options[i]=null/””

所有：

Options.length=0

Select 对象.innerHTML=””/null

也可以考虑使用 text 属性

事件:change()

获取选中项的值：

Select 对象.value

获取选中项的索引值：

Select 对象.selectedIndex

12.事件

- 事件流的 3 个阶段：

  捕获阶段

  处于目标阶段

  冒泡阶段

- 事件处理程序

Html 事件处理程序

直接写在 html 结构中

如：点击

DOM0 级

绑定：

对象.on 事件名=function(){.....}

删除：

对象.on 事件名=null

DOM2 级　　可添加多个事件

添加：

对象.addEventListener(“事件名”,”处理函数”,布尔值)

注意：事件名不加 on，true 捕获，false 冒泡

删除:

removeEventListener(“事件名”,”处理函数”,布尔值)

IE 事件处理函数 没有布尔值

添加：

attachEvent(“on 事件名”,事件处理函数)

删除:

detachEvent(“on 事件名”,事件处理函数)

- 事件对象

标准浏览器：作为事件的第一个参数访问

低版本浏览器：作为 window 的 event 的属性访问

解决办法：var event=event||window.event

阻止事件冒泡：

事件对象.stopPropagation()

事件对象.cancelBubble=true

取消默认行为：

事件对象.preventDefaul()

事件对象.returnValue=false

事件委托技术：给层次较高的元素添加事件

获取事件目标:

事件对象.target

事件对象.srcElement　　IE

事件类型：

事件对象.type

获取键值：返回 ASCII 码值

事件对象.keyCode

光标位置:

事件对象.clientX

事件对象.clientY

13.面向对象

- 创建函数

函数声明:

Function 函数名(){......}

函数表达式：

Var 函数名=function(){.....}

- 递归函数

例如：function fn(n){

If(n==1){

Return 1;

}else{

Return n*fn(n-1);

}

}

- 创建对象　　属性和方法的集合

1. 构造函数

Var obj=new Object();

Obj.属性名=值；

Obj.方法名=function(){

}

2.字面量法

Var obj={

Name：“”,

Sex:””

}

3.工厂模式

将创建对象的语句封装在一个函数中，有 return

Function 函数名(参数列表){

Var obj=new Object();

Obj.name=”” ;

Obj.fn=function(){.....}

Return obj;

}

4.构造函数 必须通过 new 运算符，无 return 语句

Function 函数名(){

This.name=”” ;

This.fn=function(){.....}

}

Var obj1=new 函数名();

5.原型模式

Function user(n){

User.prototype.name=”” ;

User.prototype.fn=function(){.....}

}

6.组合模式 构造函数+原型模式

```
Function user(){
    This.name=””
}
User.prototype={
    Work:function(){},
    Fn:function(){}
}
```

14.继承 子类继承父类里的属性和方法

- 原型链继承

定义父类：function super(){
This.house=”房子”;
This.job=function(){}
}

定义子类：function sub(){...}

引用原型链：

Sub.prototype=new Super();
Sub.job();

- 借用构造函数

通过 call()和 apply()来调用超类

函数名.call();

Call():用来更改 this 指向，参数以列表形式展现

Apply():用来改变 this 指向,参数以数组形式传参

定义父类：

```
Function super(){
    This.house=””;
    This.fn=function(){....}
}
```

定义子类：

```
Function sub(){
    Super.call(this);     //实现继承
}
```

- 原型式继承
- 寄生式继承
- 寄生组合式继承

15.数组常用的方法：

concat()连接两个或更多的数组，并返回结果。

join()把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。

pop()删除并返回数组的最后一个元素

push()向数组的末尾添加一个或更多元素，并返回新的长度。

reverse()颠倒数组中元素的顺序。

shift()删除并返回数组的第一个元素

slice(n,m)从某个已有的数组查找返回选定的元素

sort()对数组的元素进行排序

splice(开始，长度，替换字符) 删除元素，并向数组添加新元素。

toSource()返回该对象的源代码

toString()把数组转换为字符串，并返回结果。

toLocaleString()把数组转换为本地字符串，并返回结果。

unshift()向数组的开头添加一个或更多元素，并返回新的长度。

valueOf()返回数组对象的原始值

16.字符串常用的方法:

concat() - 将两个或多个字符的文本组合起来，返回一个新的字符串。

indexOf() -返回字符串中一个子串第一处出现的索引。如果没有匹配项，返回 -1 。

lastindexOf() - 返回字符串中一个子串从后往前第一处出现的索引。如果没有匹配项，返回 -1 。

charAt() - 返回指定位置的字符。

charCodeAt() - 返回指定位置的字符 ascii 编码。

lastIndexOf() - 返回字符串中一个子串最后一处出现的索引，如果没有匹配项，返回 -1 。

match() - 检查一个字符串是否匹配一个正则表达式。

substring() - 返回字符串的一个子串。传入参数是起始位置和结束位置。

replace() - 用来查找匹配一个正则表达式的字符串，然后使用新字符串代替匹配的字符串。

search() - 执行一个正则表达式匹配查找。如果查找成功，返回字符串中匹配的索引值。否则返回 -1 。

slice(n,m) - 提取字符串的一部分，并返回一个新字符串。

Substring(n,m) 返回位于 String 对象中指定位置的子字符串，子字符串中包括 start 位置字符，不包括end 位置上的字符。

Substr(n.len)返回一个从指定位置（start）开始的指定长度(length)的子字符串;

split() - 通过将字符串划分成子串，将一个字符串做成一个字符串数组。

length - 返回字符串的长度,所谓字符串的长度是指其包含的字符的个数。

toLowerCase() - 将整个字符串转成小写字母。

toLocalLowerCase() - 将整个字符串转成小写字母。

toUpperCase() - 将整个字符串转成大写字母。

ToLocalUpperCase() - 将整个字符串转成大写字母。

Trim()去掉前后空格

Json() - 转换字符串对象的方法