1、浏览器页面有哪三层构成，分别是什么，作用什么？

构成：结构层、表示层、行为层

查看更多

分别是：HTML、CSS、JavaScript

作用：HTML实现页面结构、CSS完成页面的表现与风格、JavaScript实现一些客户端的功能与业务。

2、HTML5的优点和缺点：

优点： a、网络标准统一、HTML5本身是由W3C推荐出来的。意味着每一个浏览器或者每一个平台都会去实现

b、多设备、跨平台。例如可以轻松地移植到UC的开放平台、Opera的游戏中心等。自适应调整布局。

c、即时更新。不用像端游那样更新客户端。

d、提高可用性和改进用户的友好体验；

e、有几个新的标签，这将有助于开发人员定义重要的内容；

f、可以给站点带来更多的多媒体元素(视频和音频)；

g、可以很好的替代Flash和Silverlight；

h、涉及到网站的抓取和索引的时候，对于SEO很友好；

i、被大量应用于移动应用程序和游戏。

缺点： a、安全：像之前Firefox4的web socket和透明代理的实现存在严重的安全问题，同时web storage、web socket 这样的功能很容易被黑客利用，来盗取用户的信息和资料。

b、完善性：许多特性各浏览器的支持程度也不一样。

c、技术门槛：HTML5简化开发者工作的同时代表了有许多新的属性和API需要开发者学习，像web worker、web socket、web storage 等新特性，后台甚至浏览器原理的知识，机遇的同时也是巨大的挑战

d、性能：某些平台上的引擎问题导致HTML5性能低下。

e、浏览器兼容性：最大缺点，IE9以下浏览器几乎全军覆没。

3、Doctype的作用？严格模式与混杂模式如何区分？它们有什么意义？

（1）、 声明位于文档中的最前面，处于标签之前。告知浏览器的解析器，用什么文档类型规范来解析这个文档。(使用哪种html或xhtml规范)

（2）、严格模式的排版和JS 运作模式是以该浏览器支持的最高标准运行。

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

（4）、DOCTYPE不存在或格式不正确会导致文档以混杂模式呈现。

html4、xhtml(EXtensible HyperText Markup Language)和html5：
html一直发展到html4都是先实现后有标准的，导致html标准不是很规范，浏览器对html页面中的错误相当宽容，导致html作者写出了大量含有错误的html页面。（严格模式、过度模式、框架模式）
随后W3C为了规范html，结合XML指定了XHTML，按照XML的要求来规范XHTML，定义了心的MIME type（application/xhtml+xml）。新的MIME type进行强错误检查，如果有HTML错误，就要现实错误信息，所以没什么人用。
后来有了HTML5，将向后兼容作为了一个很重要的原则。不会break已有的网页，只要将第一行改成<!DOCTYPE html>。
在html4.01中，<!DOCTYPE>声明引用DTD，因为HTML4.01基于SGML。DTD规定了标记语言的规则，这样浏览器才能正确地呈现内容。
HTML不基于SGML，所以不需要引用DTD。
SGML：标准通用标记语言，由三部分组成：语法定义、文件类型定义（DTD）、文件实例。
SGML、HTML是XML的先驱。

4、HTML有哪些新特征，移除了哪些元素?
css复习笔记中有

5、浏览器内核有哪些：
IE：trident
Firefox：gecko
Safari：webkit
Opera：presto，现已改用Google Chrome的blink
Chrome：blink（基于webkit，Google与Opera Software共同开发）

6、对HTML5的认识：
HTML5指的是包括 HTML 、 CSS 和 JavaScript 在内的一套技术组合。它希望能够减少网页浏览器对于需要插件的丰富性网络应用服务（ Plug-in-Based Rich Internet Application， RIA ），例如： AdobeFlash 、 Microsoft Silverlight 与 Oracle JavaFX 的需求，并且提供更多能有效加强网络应用的标准集（audio，video）。 HTML5 是 HTML 最新版本， 2014 年 10 月由万维网联盟（ W3C ）完成标准制定。目标是替换 1999 年所制定的 HTML 4.01和 XHTML 1.0 标准，以期能在互联网应用迅速发展的时候，使网络标准达到匹配当代的网络需求。
为什么：

HTML4陈旧不能满足日益发展的互联网需要，特别是移动互联网。为了增强浏览器功能 Flash 被广泛使用，但安全与稳定堪忧，不适合在移动端使用（耗电、触摸、不开放）。
HTML5增强了浏览器的原生功能，符合 HTML5 规范的浏览器功能将更加强大，减少了 Web 应用对插件的依赖，让用户体验更好，让开发更加方便，另外 W3C 从推出 HTML4.0 到 5.0 之间共经历了 17 年， HTML 的变化很小，这并不符合一个好产品的演进规则。

7、什么是WebGL，它有什么优点?
WebGL（全写 Web Graphics Library ）是一种 3D 绘图标准，这种绘图技术标准允许把 JavaScript 和 OpenGL ES 2.0 结合在一起，通过增加OpenGL ES 2.0 的一个 JavaScript 绑定， WebGL 可以为 HTML5 Canvas 提供硬件 3D 加速渲染，这样 Web 开发人员就可以借助系统显卡来在浏览器里更流畅地展示 3D 场景和模型了，还能创建复杂的导航和数据视觉化。显然， WebGL 技术标准免去了开发网页专用渲染插件的麻烦，可被用于创建具有复杂 3D 结构的网站页面，甚至可以用来设计 3D 网页游戏等等。
WebGL完美地解决了现有的 Web 交互式三维动画的两个问题：
第一，它通过HTML脚本本身实现 Web 交互式三维动画的制作，无需任何浏览器插件支持；
第二，它利用底层的图形硬件加速功能进行的图形渲染，是通过统一的、标准的、跨平台的OpenGL接口实现的。
通俗说WebGL中 canvas 绘图中的 3D 版本。因为原生的 WebGL 很复杂，我们经常会使用一些三方的库，如 three.js 等，这些库多数用于 HTML5游戏开发。

8、cookie、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 Internet Explorer 8 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 许 cookie 多达 4096 个字节，包括：名（

name ）、值（ value ）和等号。 Internet Explorer 允许 cookie 多达 4095 个字节，包括：名（ name ）、值（ value ）和等号。
区别：
- Cookie
+ 每个域名存储量比较小（各浏览器不同，大致 4K ）
+ 所有域名的存储量有限制（各浏览器不同，大致 4K ）
+ 有个数限制（各浏览器不同）
+ 会随请求发送到服务器
- LocalStorage
+ 永久存储
+ 单个域名存储量比较大（推荐 5MB ，各浏览器不同）
+ 总体数量无限制
- SessionStorage
+ 只在 Session 内有效
+ 存储量更大（推荐没有限制，但是实际上各浏览器也不同）

9、HTML语义化的理解：
(1)什么是 HTML 语义化?
<基本上都是围绕着几个主要的标签，像标题（ H1~H6 ）、列表（ li ）、强调（ strong em ）等等 >
根据内容的结构化（内容语义化），选择合适的标签（代码语义化）便于开发者阅读和写出更优雅的代码的同时让浏览器的爬虫和机器很好地解析。（table表格，p段落）
(2)为什么要语义化?
1、为了在没有CSS的情况下，页面也能呈现出很好地内容结构、代码结构 : 为了裸奔时好看；（table，ul，ol，select等）
2、用户体验：例如title、 alt 用于解释名词或解释图片信息、 label 标签的活用；
3、有利于SEO（搜索引擎优化）：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：爬虫依赖于标签来确定上下文和各个关键字的权重；
4、方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；
5、便于团队开发和维护，语义化更具可读性，是下一步网页的重要动向，遵循W3C标准的团队都遵循这个标准，可以减少差异化。
(3) 语义化标签（以下这些都是H5中的）
<header></header>
<footer></footer>
<nav></nav>
<section></section>
<article></article> SM:用来在页面中表示一套结构完整且独立的内容部分
<aslde></aside> SM:主题的附属信息（ 用途很广，主要就是一个附属内容 ），如果 article 里面为一篇文章的话，那么文章的作者以及信息内容就是这篇文章的附属内容了

<figure></figure>SM:媒体元素，比如一些视频，图片啊等等
<datalist></datalist>
SM:选项列表，与 input 元素配合使用，来定义 input 可能的值
<details></details>
SM:用于描述文档或者文档某个部分的细节 ~ 默认属性为 open~
ps:配合 summary 一起使用

10、link和@import的区别：
都可以在页面中引用。
<link rel='stylesheet' rev='stylesheet' href='CSS文件 ' type='text/css' media='all' />
<style type='text/css' media='screen'>
@import url('CSS文件 ');
</style>
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 。
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。
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。
4、 link 支持使用 Javascript 控制 DOM 去改变样式；而 @import 不支持。

11、SVG的理解：
SVG可缩放矢量图形（ Scalable Vector Graphics ）是基于可扩展标记语言（ XML ），用于描述二维矢量图形的一种图形格式。 SVG 是W3C('World Wide Web ConSortium' 即 ' 国际互联网标准组织 ') 在 2000 年 8 月制定的一种新的二维矢量图形格式，也是规范中的网络矢量图形标准。 SVG 严格遵从 XML 语法，并用文本格式的描述性语言来描述图像内容，因此是一种和图像分辨率无关的矢量图形格式。 SVG 于 2003 年 1 月14 日成为 W3C 推荐标准。
特点：
(1)任意放缩
用户可以任意缩放图像显示，而不会破坏图像的清晰度、细节等。
(2)文本独立
SVG图像中的文字独立于图像，文字保留可编辑和可搜寻的状态。也不会再有字体的限制，用户系统即使没有安装某一字体，也会看到和他们制作时完全相同的画面。
(3)较小文件
总体来讲，SVG文件比那些 GIF 和 JPEG 格式的文件要小很多，因而下载也很快。
(4)超强显示效果

SVG图像在屏幕上总是边缘清晰，它的清晰度适合任何屏幕分辨率和打印分辨率。
(5)超级颜色控制
SVG图像提供一个 1600 万种颜色的调色板，支持 ICC 颜色描述文件标准、 RGB 、线 X 填充、渐变和蒙版。
(6)交互 X 和智能化。 SVG 面临的主要问题一个是如何和已经占有重要市场份额的矢量图形格式 Flash 竞争的问题，另一个问题就是 SVG 的本地运行环境下的厂家支持程度。
浏览器支持：
Internet Explorer9，火狐，谷歌 Chrome ， Opera 和 Safari 都支持 SVG 。
IE8和早期版本都需要一个插件 - 如 Adobe SVG 浏览器，这是免费提供的。

12、HTML全局属性有哪些：（全局属性是指：所有的元素都拥有的属性，设置是html5未定义的标签）
accesskey：设置快捷键，提供快速访问元素的快捷键。在Mac上，对chrome，使用alt+control+accesskey定义的键，可以快速激活对应元素。（a标签就是访问对应地址。）
class：为元素设置类标识
contenteditable（H5）：表示这个标签可以被用户编辑。true或者false。
contextmenu（H5）：为元素自定义一个上下文菜单，在鼠标右键的时候弹出菜单内容。（contextmenu的值跟右键的标签的id一致）,大多数浏览器不支持
data-：自定义属性，名字不能以xml开头，不能含有分号，不能含有大写字母。
dir：表明标签的文本方向，ltr（从左往右）、rtl（从右往左）、auto
traggable（H5）：决定一个标签是否可以被拖动。true或者false。
dropzone（H5）：被拖动的项目被拖放到元素中时会发生什么。没有浏览器支持。

- copy, 表示丢放时会创建一个被拖拽element的副本；
- move, 表示被拖拽的element被移动到这个新位置；
- link, 将会给拖拽的数据（dragged data）创建一个链接；

hidden（H5）：隐藏元素
id：唯一的标识
lang：定义元素中内容的语言
spellcheck（H5）：是否对元素进行拼写或者语法检查。true或者false。
style：定义css样式。
tabindex：规定元素的tab键控制次序

- 负数意味这element不可以获得焦点，也不可以通过排序的键盘导航到达；
- 0意味着element可以通过排序的键盘导航到达，但是相对顺序取决于平台惯例；
- 一个正数意味着可以通过排序的键盘导航获得焦点并到达。相对顺序去决议该属性的值，按照tabindex的增值排序。如果几个element有相同的tabindex,他们的相对顺序取决于他们在document中的位置。

title：有关元素的额外信息。（img有alt属性，在图像无法显示时显示）

13、超链接target属性的取值和作用：
target定义所链接的页面在浏览器窗口中的打开方式：
_blank：在新浏览器窗口中打开
_self：在当前窗口中打开
_parent：将链接的文件载入含有该链接框架的父框架集或父窗口中。如果含有该链接的框架不是嵌套的，则在浏览器全屏窗口中载入链接的文件，就象 _self 参数一。
_top：在当前的整个浏览器窗口中打开所链接的文档，因为会删除所有框架。
window.open(url,name,features,replace)当中，name可以是上述某一个值，也可以是自定义名字。

14、data-属性的作用是什么：
是H5新增的属性，为前端开发者提供自定义的属性，当没有合适的属性和元素的时候，自定义的data属性是能够存储页面或者App的私有的自定义数据。
取值的时候，使用element.dataset或者element.getAttribute(key).
关于属性获取：
1、属性选择器，不能用于自定义属性（IE中可以）
例如：element.style.backgroundColor，element.className；可以获得对应的类型，如element.onclick，获得的就是一个函数
2、get/setAttribute():可以用于任何属性，包括自定义属性，不区分大小写
例如：element.getAttribute("class")，引号中key的值与html中定义的一致。返回的都是字符串。
element.getAttibute("data-id")
3、dataset：用于自定义属性
DOMStringMap {id: "hell"}
例如：div.dataset.xxx，xxx只包含data-后面的内容，如果后面的内容还有连字符，则使用驼峰形式
低版本的浏览器不支持，可以用getAttribute做兼容
4、attributes：可以用于任何属性，包括自定义属性，不区分大小写
NamedNodeMap {0: data-id, 1: style, length: 2}
可以通过div.attributes['data-id']来获取，返回：data-id="id"；div.attributes['data-id'].nodeValue，返回id；div.attributes['data-id'].nodeName，返回：data-id

关于样式的操作：

1、 **document.styleSheets**[0].cssRules[1].style.color

document.styleSheets[0].cssRules[1].style.selectorText：对应选择器

document.styleSheets：内联样式表和外联样式表。

document.styleSheets[0]：内联样式表和外联样式表中的第一个表

document.styleSheets[0].cssRules[1]：内联样式表的第二行

document.styleSheets[0].cssRules[1].style：内联样式表的第二行内的css内容

这种方法是针对内联和外联

2、 **element.style** = ""

element.style.cssText="输入多条css语句"

element.style.borderColor

这种方法是针对行内样式

3、 **element.className** = ""

element.className += ""追加类名

4、**element（link标签）.href** = ""

5、var style= **window.getComputedStyle**(element[,pseudoElt])

var style= window.getComputedStyle(element[,pseudoElt]).color，获取页面中的元素的颜色。

如果使用element.style.只能获取到行内设置的，不一定是页面真正的样式。

ie9以下不支持，ie9以下使用element.currentStyle来做兼容

15、对浏览器内核的理解

主要分成两部分：渲染引擎(layout engineer或 Rendering Engine) 和 JS 引擎。

渲染引擎：负责取得网页的内容（HTML、 XML 、图像等等）、整理讯息（例如加入 CSS 等），以及计算网页的显示方式，然后会输出至显示器或打印机。浏览器的内核的不同对于网页的语法解释会有不同，所以渲染的效果也不相同。所有网页浏览器、电子邮件客户端以及其它需要编辑、显示网络内容的应用程序都需要内核。

JS引擎则：解析和执行 javascript 来实现网页的动态效果。

最开始渲染引擎和JS引擎并没有区分的很明确，后来 JS 引擎越来越独立，内核就倾向于只指渲染引擎。

16、常见的浏览器内核：

1.

Trident内核： IE,MaxThon,TT,The World,360, 搜狗浏览器等。[又称 MSHTML]

2.

Gecko内核： Netscape6 及以上版本， FF,MozillaSuite/SeaMonkey 等
3.
Presto内核： Opera7 及以上。 ［Opera 内核原为： Presto ，现为： Blink;］
4.
Webkit内核： Safari,Chrome 等。 ［ Chrome 的： Blink （ WebKit 的分支） ］

17、iframe的缺点
1、iframe会阻塞主页面的 Onload 事件；
2、搜索引擎的检索程序无法解读这种页面，不利于 SEO;
3、iframe和主页面共享连接池，而浏览器对相同域的连接有限制，所以会影响页面的并行加载。
使用iframe之前需要考虑这两个缺点。如果需要使用 iframe ，最好是通过 javascript 动态给iframe添加 src 属性值，这样可以绕开以上两个问题。

18、label标签的作用，如何使用：
label标签来定义表单控制间的关系，当用户选择该标签时，浏览器会自动将焦点转到和标签相关的表单控件上。
<label for='Name'>Number:</label>
<input type=“ text “ name='Name' id='Name'/>
<label>Date:<input type='text' name='B'/></label>

注意:label的for属性值要与后面对应的input标签id属性值相同
<label for='Name'>Number:</label>
<input type=“ text “ name='Name' id='Name'/>
另一种用法，是与表单配合使用，在form属性中写表单的ID，代表来自哪个表单

19、如何实现浏览器内多个标签页之间的通信
1、**cookie+setInterval**：
将要传递的信息存储在cookie中，每隔一定时间读取cookie信息，即可随时获取要传递的信息。
2、localStorage的**storage事件**：
```
 window.addEventListener("storage", function(e){
    console.log(e);
    document.write("oldValue: "+ e.oldValue + " newValue:" + e.newValue)
```

});

3、利用HTML5中webWorker的postMessage方法

在Web Worker中，postMessage是MessagePort对象的方法之一，用来推送数据(与之相对的为onmessage，用来接收数据)。

postMessage(data,origin)

参数:

data:要传递的数据，html5规范该参数可以是JavaScript的任意基本类型或可复制的对象，然而部分浏览器只能处理字符串参数，所以传递参数的时候需要使用JSON.stringify()方法对对象参数序列化，在低版本IE中引用json2.js可以实现类似效果。

origin:字符串参数，指明目标窗口的源，协议+主机+端口号[+URL]，URL会被忽略，所以可以不写，这个参数是为了安全考虑，postMessage()方 法只会将message传递给指定窗口，当然如果愿意也可以建参数设置为"*"，这样可以传递给任意窗口，如果要指定和当前窗口同源的话设置为"/"。

使用这种方法，需要借助于iframe。可以从父窗口发送给子窗口，也可以从子窗口发送给父窗口。具体可见学习内容－html－标签页通信。

postMessage方法和onmessage事件不仅只有Worker线程拥有。

在页面中，一些Window对象拥有postMessage方法，例如parent（Window {stop: function, open: function, alert: function, confirm: function, prompt: function...}）、（iframe）.contentWindow。这些对象可以直接调用postMessage方法。可以写：`document.getElementById('child').contentWindow.postMessage(document.getElementById('message').value,'/');(发送给document.getElementById('child')这个iframe)`

同时，window对象上有onmessage事件。可以写：`window.addEventListener('message', function(event){})`

postMessage会被onmessage检测到。

在这里，谁来调用postMessage，就代表发给哪个子窗口。

19、如何在页面上实现一个圆形的可点击区域

1、map+area

```
<body>
    <img src="t.jpg" width="1366" height="768" border="0" usemap="#Map">
    <map name="Map" id="Map">
        <area shape="circle" coords="180,139,100" href="http://www.baidu.com" target="_blank" />
    </map>
</body>
```

a、img 元素中的 "usemap" 属性引用 map 元素中的 "id" 或 "name" 属性（根据浏览

器），所以我们同时向 map 元素添加了 "id" 和 "name" 属性。
b、area 元素永远嵌套在 map 元素内部。area 元素可定义图像映射中的区域。
c、shape和coords配合使用，
圆形：shape="circle"，coords="x,y,z"
这里的 x 和 y 定义了圆心的位置（"0,0" 是图像左上角的坐标），r 是以像素为单位的圆形半径。
多边形：shape="polygon"，coords="x1,y1,x2,y2,x3,y3,..."
每一对 "x,y" 坐标都定义了多边形的一个顶点（"0,0" 是图像左上角的坐标）。定义三角形至少需要三组坐标；高纬多边形则需要更多数量的顶点。
矩形：shape="rectangle"，coords="x1,y1,x2,y2"
第一个坐标是矩形的一个角的顶点坐标，另一对坐标是对角的顶点坐标，"0,0" 是图像左上角的坐标。
d、href：点击要打开的网页
2、border-radius（H5）

```
.circle{
    display: block;
    width: 100px;
    height: 100px;
    background-color: red;
    border-radius: 100px;
    cursor: pointer;
    text-decoration: none;
    line-height: 100px;
    text-align: center;
}
<div><a class="circle" href="http://www.baidu.com" target="_blank">点击</a>
</div>
```

3、js实现
圆心在(100,100)，半径为50

```
document.addEventListener('click',function(e){
    var r = 50,
        x1 = 100,
        y1 = 100,
        x2 = e.clientX,
        y2 = e.clientY;
    var len = Math.abs(Math.sqrt(Math.pow(x2-x1,2)+Math.pow(y2-y1,2)));
    if (len < 50) {
```

```
                console.log('success');
            }
        });
```

4、svg实现：

```
        <svg width="100" height="100">
            <circle cx="50" cy="50" r="50" fill="red"></circle>
        </svg>
```

cx 和 cy 属性定义圆点的 x 和 y 坐标。如果省略 cx 和 cy，圆的中心会被设置为 (0, 0)

r 属性定义圆的半径。

20、title与h3的区别、b与strong的区别、i与em的区别

title属性没有明确意义只表示是个标题， H1 则表示层次明确的标题，对页面信息的抓取也有很大的影响；

strong是标明重点内容，有语气加强的含义，使用阅读设备阅读网络时： 会重读；而 <B> 是展示强调内容。

i内容展示为斜体， em 表示强调的文本；

Physical Style Elements -- 自然样式标签

b, i, u, s, pre（起到展示效果）

Semantic Style Elements -- 语义样式标签

strong, em, ins, del, code（有语义作用）

应该准确使用语义样式标签, 但不能滥用 , 如果不能确定时首选使用自然样式标签。

21、不使用 border 画出1px高的线，在不同浏览器的标准模式与怪异模式下都能保持一致的效果?

使用div的宽度

```
<div style='height:1px;overflow:hidden;background:red'></div>
```

22、THML5标签的作用

a、使web页面的内容更加有序和规范

b、使搜索引擎更加容易按照HTML5规则识别出有效的内容
c、使web页面更接近于一种数据字段和表

23、src和href的区别：
1、src用于替换当前元素， href 用于在当前文档和引用资源之间确立联系。
2、src是 source 的缩写，指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求 src 资源时会将其指向的资源下载并应用到文档内，例如 js 脚本，img 图片和 frame 等元素。
<script src ='js.js'></script>
当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行完毕，图片和框架等元素也如此，类似于将所指向资源嵌入当前标签内。这也是为什么将js脚本放在底部而不是头部。
href是 Hypertext Reference 的缩写，指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的链接，如果我们在文档中添加
<link href='common.css' rel='stylesheet'/>
那么浏览器会识别该文档为css文件，就会并行下载资源并且不会停止对当前文档的处理。这也是为什么建议使用 link 方式来加载 css ，而不是使用@import 方式。

24、对canvas的理解
canvas是HTML5中新增一个HTML5标签与操作canvas的javascript API，它可以实现在网页中完成动态的2D与3D图像技术。标记和 SVG以及 VML 之间的一个重要的不同是，有一个基于 JavaScript 的绘图 API，而 SVG 和 VML 使用一个 XML 文档来描述绘图。SVG 绘图很容易编辑与生成，但功能明显要弱一些。 canvas可以完成动画、游戏、图表、图像处理等原来需要Flash完成的一些功能。

25、WebSocket与消息推送
B/S架构的系统多使用HTTP协议，
HTTP协议的特点： 1 无状态协议；2 用于通过 Internet 发送请求消息和响应消息；3 使用端口接收和发送消息，默认为80端口 底层通信还是使用Socket完成
HTTP协议决定了服务器与客户端之间的连接方式，无法直接实现消息推送（ F5 已坏），一些变相的解决办法：
1、轮询：客户端定时向服务器发送Ajax请求，服务器接到请求后马上返回响应信息并关闭连接。
优点：后端程序编写比较容易。

缺点：请求中有大半是无用，浪费带宽和服务器资源。
实例：适于小型应用。
2、长轮询：客户端向服务器发送Ajax请求，服务器接到请求后 hold 住连接，直到有新消息才返回响应信息并关闭连接，客户端处理完响应信息后再向服务器发送新的请求。
得到响应后会断开连接
优点：在无消息的情况下不会频繁的请求，耗费资小。
缺点：服务器hold连接会消耗资源，返回数据顺序无保证，难于管理维护。 Comet 异步的 ashx ，
实例：WebQQ、 Hi 网页版、 Facebook IM 。
3、长连接：在页面里嵌入一个隐藏iframe，将这个隐藏 iframe 的 src 属性设为对一个长连接的请求或是采用 xhr 请求，服务器端就能源源不断地往客户端输入数据。(建立一次http连接)
得到响应后不会断开连接，等待客户端再次向服务器发送http请求
优点：消息即时到达，不发无用请求；管理起来也相对便。
缺点：服务器维护一个长连接会增加开销。
实例：Gmail聊天
4、Flash Socket：在页面中内嵌入一个使用了 Socket 类的 Flash 程序 JavaScript 通过调用此 Flash 程序提供的 Socket 接口与服务器端的 Socket 接口进行通信， JavaScript 在收到服务器端传送的信息后控制页面的显示。
优点：实现真正的即时通信，而不是伪即时。
缺点：客户端必须安装Flash插件；非 HTTP 协议，无法自动穿越防火墙。
实例：网络互动游戏。

Websocket:
WebSocket是 HTML5 开始提供的一种浏览器与服务器间进行全双工通讯的网络技术。依靠这种技术可以实现客户端和服务器端的长连接，双向实时通信。只需要发送一次http请求，便可以双向通信。
特点:
a、事件驱动
b、异步
c、使用 ws 或者 wss 协议的客户端 socket
d、能够实现真正意义上的推送功能
缺点：少部分浏览器不支持，浏览器支持的程度与方式有区别。
WebSocket 的出现使得浏览器提供对 Socket 的支持成为可能，从而在浏览器和服务器之间提供了一个基于 TCP 连接的双向通道。Web 开发人员可以非常方便地使用 WebSocket 构建实时 web 应用。
Web 应用的信息交互过程通常是客户端通过浏览器发出一个请求，服务器端接收和审核完请求后进行处理并返回结果给客户端，然后客户端浏览器将信息呈现出来，这种机制对于信息变化不是特别频繁的应用尚能相安无事，但是对于那些实时要求比较高的应用来说，比如说在线游戏、在线证券、设备监控、新闻在线播报、RSS 订阅推送等等，当客户端浏览器准备呈现这些信息的时候，这些信息在服务器端可能已经过时了。所以保持客户端和服务器端的信息同步是实时 Web 应用的关键要素，对 Web 开发人员来说

也是一个难题。
轮询：
这是最早的一种实现实时 Web 应用的方案。客户端以一定的时间间隔向服务端发出请求，以频繁请求的方式来保持客户端和服务器端的同步。这种同步方案的最大问题是，当客户端以固定频率向服务器发起请求的时候，服务器端的数据可能并没有更新，这样会带来很多无谓的网络传输，所以这是一种非常低效的实时方案。
长轮询：
长轮询是对定时轮询的改进和提高，目地是为了降低无效的网络传输。当服务器端没有数据更新的时候，连接会保持一段时间周期直到数据或状态改变或者时间过期，通过这种机制来减少无效的客户端和服务器间的交互。当然，如果服务端的数据变更非常频繁的话，这种机制和定时轮询比较起来没有本质上的性能的提高。
流：
流技术方案通常就是在客户端的页面使用一个隐藏的窗口向服务端发出一个长连接的请求。服务器端接到这个请求后作出回应并不断更新连接状态以保证客户端和服务器端的连接不过期。通过这种机制可以将服务器端的信息源源不断地推向客户端。这种机制在用户体验上有一点问题，需要针对不同的浏览器设计不同的方案来改进用户体验，同时这种机制在并发比较大的情况下，对服务器端的资源是一个极大的考验。
综合这几种方案，您会发现这些目前我们所使用的所谓的实时技术并不是真正的实时技术，它们只是在用 Ajax 方式来模拟实时的效果，在每次客户端和服务器端交互的时候都是一次 HTTP 的请求和应答的过程，而每一次的 HTTP 请求和应答都带有完整的 HTTP 头信息，这就增加了每次传输的数据量，而且这些方案中客户端和服务器端的编程实现都比较复杂，在实际的应用中，为了模拟比较真实的实时效果，开发人员往往需要构造两个 HTTP 连接来模拟客户端和服务器之间的双向通讯，一个连接用来处理客户端到服务器端的数据传输，一个连接用来处理服务器端到客户端的数据传输，这不可避免地增加了编程实现的复杂度，也增加了服务器端的负载，制约了应用系统的扩展性。
HTML5 WebSocket 设计出来的目的就是要取代轮询和 Comet 技术，使客户端浏览器具备像 C/S 架构下桌面系统的实时通讯能力。 浏览器通过 JavaScript 向服务器发出建立 WebSocket 连接的请求，连接建立以后，客户端和服务器端就可以通过 TCP 连接直接交换数据。因为 WebSocket 连接本质上就是一个 TCP 连接，所以在数据传输的稳定性和数据传输量的大小方面，和轮询以及 Comet 技术比较，具有很大的性能优势。
WebSocket 协议本质上是一个基于 TCP 的协议。为了建立一个 WebSocket 连接，客户端浏览器首先要向服务器发起一个 HTTP 请求，这个请求和通常的 HTTP 请求不同，包含了一些附加头信息，其中附加头信息”Upgrade: WebSocket”表明这是一个申请协议升级的 HTTP 请求，服务器端解析这些附加的头信息然后产生应答信息返回给客户端，客户端和服务器端的 WebSocket 连接就建立起来了，双方就可以通过这个连接通道自由的传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连接。

26、img的title和alt有什么区别

alt用于图片无法加载时显示；title为该属性提供信息，通常当鼠标滑动到元素上的时候显示

27、表单的基本组成部分有哪些，表单的主要用途是什么
组成：表单标签、表单域、表单按钮
a、表单标签：这里面包含了处理表单数据所用 CGI 程序的 URL, 以及数据提交到服务器的方法。
b、表单域：包含了文本框、密码框、隐藏域、多行文本框、复选框、单选框、下拉选择框、和文件上传框等。
c、表单按钮：包括提交按钮，复位按钮和一般按钮；用于将数据传送到服务器上的 CGI 脚本或者取消输入，还可以用表单按钮来控制其他定义了处理脚本的处理工作。
主要用途：表单在网页中主要负责数据采集的功能，和向服务器传送数据。

28、表单提交中Get和Post的区别：
(1)、 get 是从服务器上获取数据， post 是向服务器传送数据。
(2)、 get 是把参数数据队列加到提交表单的 ACTION 属性所指的 URL 中，值和表单内各个字段一一对应，在 URL 中可以看到。 post 是通过 HTTP post 机制，将表单内各个字段与其内容放置在 HTML HEADER 内一起传送到 ACTION 属性所指的 URL 地址 , 用户看不到这个过程。
(3)、对于 get 方式，服务器端用 Request.QueryString 获取变量的值，对于 post 方式，服务器端用 Request.Form 获取提交的数据。
(4)、 get 传送的数据量较小，不能大于 2KB 。 post 传送的数据量较大，一般被默认为不受限制。但理论上， IIS4 中最大量为 80KB ， IIS5 中为100KB 。
(5)、 get 安全性低， post 安全性较高。

29、HTML5新增表单元素：
datalist
keygen
output

30、HTML5放弃了哪些HTML4标签：
frame
frameset
noframe
applet
big
center
basefront

31、HTML5提供了哪些新的API：
HTML5 提供的应用程序 API 主要有：
· Media API
· Text Track API
· Application Cache API
· User Interaction
· Data Transfer API
· Command API
· Constraint Validation API
· History API

32、HTML5的存储类型有什么区别：
localStorage 用于持久化的本地存储，数据永远不会过期，关闭浏览器也不会丢失。
· sessionStorage 同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储

33、HTML5应用程序缓存和浏览器缓存有什么区别：
应用程序缓存是 HTML5 的重要特性之一，提供了离线使用的功能，让应用程序可以获取本地的网站内容，例如 HTML 、 CSS 、图片以及 JavaScript 。这个特性可以提高网

站性能，它的实现借助于 manifest 文件，如下：
example.manifest:
CACHE MANIFEST
#comment
file.js
xxxx.js
xxxx.html
列出要缓存的文件
<!doctype html>
<html manifest=”example.manifest”>
.....
</html>

与传统浏览器缓存相比，它不强制用户访问的网站内容被缓存。
应用程序缓存是会预加载的，保证齐全地供应和保存。浏览器缓存没有这些控制，不能作为程序缓存使用。不幸地，应用程序缓存过於简单，导致效率不彰，预期将会被 Service Worker 取代。

HTML5 引入了应用程序缓存，这意味着 web 应用可进行缓存，并可在没有因特网连接时进行访问。
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。

HTML5应用程序缓存和浏览器缓存的区别。
（有些）浏览器会主动保存自己的缓存文件以加快网站加载速度。但是要实现浏览器缓存必须要满足一个前提，那就是网络必须要保持连接。如果网络没有连接，即使浏览器启用了对一个站点的缓存，依然无法打开这个站点。只会收到一条错误信息。而使用离线web应用，我们可以主动告诉浏览器应该从网站服务器中获取或缓存哪些文件，并且在网络离线状态下依然能够访问这个网站。
如何实现HTML5应用程序缓存。
实现HTML5应用程序缓存非常简单，只需三步，并且不需要任何API。只需要告诉浏览器需要离线缓存的文件，并对服务器和网页做一些简单的设置即可实现。
•创建一个 cache.manifest 文件，并确保文件具有正确的内容
•在服务器上设置内容类型
•所有的HTML文件都指向 cache.manifest

应用程序缓存：
chrome://appcache-internals/
浏览器缓存

chrome://cache
chrome://view-http-cache

34、canvas的作用：
Canvas 元素用于在网页上绘制图形，该元素标签强大之处在于可以直接在 HTML 上进行图形操作，
<canvas id=”canvas1“ width=”300“ height=”100“>
</canvas>
svg也可以在html页面中绘制图形。

canvas和svg的区别：
SVG
SVG 是一种使用 XML 描述 2D 图形的语言。
SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。
在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。
Canvas
Canvas 通过 JavaScript 来绘制 2D 图形。
Canvas 是逐像素进行渲染的。
在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。

35、除了 audio 和 video，HTML5 还有哪些媒体标签
<**embed**> 标签定义嵌入的内容，比如插件。
<embed type=”video/quicktime” src=”Fishing.mov”>

<**source**> 对于定义多个数据源很有用。
<video width=”450“ height=”340“ controls>
    <source src=”jamshed.mp4“ type=”video/mp4“>
    <source src=”jamshed.ogg” type=”video/ogg”>
</video>

<**track**> 标签为诸如 video 元素之类的媒介规定外部文本轨道（字幕等？）。用于规定字幕文件或其他包含文本的文件，当媒介播放时，这些文件是可见的。
<video width=”450“ height=”340“ controls>
    <source src=”jamshed.mp4“ type=”video/mp4“>

```
    <source src=” jamshed.ogg ” type= ” video/ogg ” >
    <track kind=” subtitles ” label= ” English ” src= ” jamshed_en.vtt ” srclang= ” en ” default></track>
     <track kind=” subtitles ” label= ” Arabic ” src= ” jamshed_ar.vtt ” srclang= ” ar ” ></track>
</video>
```

36、HTML5中嵌入视频：

和音频类似，HTML5 支持 MP4 、 WebM 和 Ogg 格式的视频，下面是简单示例：

```
<video width=” 450 “ height= ” 340 “ controls>
  <source src=” jamshed.mp4 “ type= ” video/mp4 “ >
   Your browser does’ nt support video embedding feature.
</video>
```

37、HTML5中嵌入音频：

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

```
<audio controls>
    <source src=” jamshed.mp3 “ type= ” audio/mpeg ” >
    Your browser does’ nt support audio embedding feature.
</audio>
```

38、HTML5文档类型和字符集

文档类型：<DOCTYPE html>

字符集：<meta charset="utf-8">

39、css盒子模型：

40、选择器：

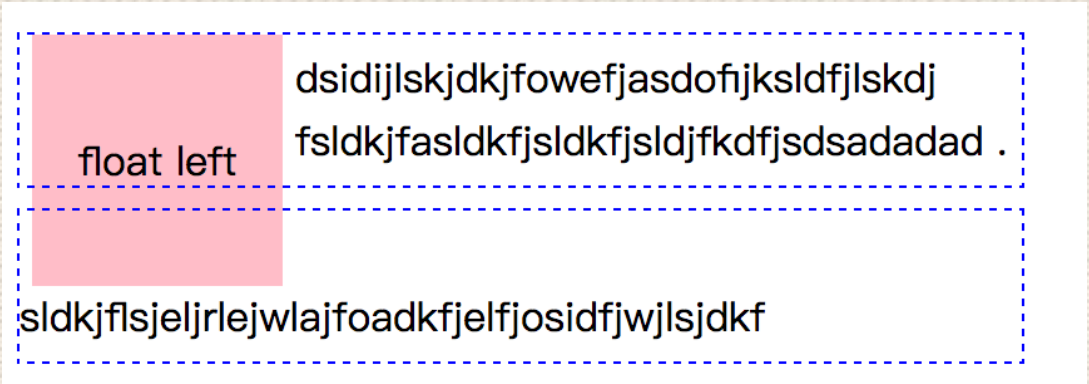
41、css特殊性（优先级、计算特殊值）
优先级
(1)、同类型，同级别的样式后者先于前者
(2))、ID > 类样式 > 标签 > *
(3)、内联>ID选择器>伪类＝属性选择器＝类选择器>标签选择器>通用选择器(*)>继承的样式
(4)、具体 > 泛化的，特殊性即css优先级
(5)、近的 > 远的 (内嵌样式 > 内部样式表 = 外联样式表)
内嵌样式：内嵌在元素中，span
内部样式表：在页面中的样式，写在<style></style>中的样式
外联样式表：单独存在一个css文件中，通过link引入或import导入的样式
(6)、!important 权重最高，比 inline style 还要高
计算特殊性值
important > 内嵌 > ID > 类 > 标签 | 伪类 | 属性选择 > 伪对象 > 继承 > 通配符
权重、特殊性计算法：
CSS样式选择器分为4个等级，a、b、c、d
(1)、如果样式是行内样式（通过Style=“”定义），那么a=1，1,0,0,0
(2)、b为ID选择器的总数 0,1,0,0
(3)、c为属性选择器，伪类选择器和class类选择器的数量。0,0,1,0
(4)、d为标签、伪元素选择器的数量 0,0,0,1
(5)、!important 权重最高，比 inline style 还要高

42、动态改变网页内容：
innerHTML、innerText
document.write和innerHTML的区别：
document.write：document.write("")，会导致整个页面重绘，写入内容是字符串的html（不会在ODM树中显示，检查元素查看不到）
innerHTML：element.innerHTML = ""，可以控制只刷新页面的一部分，精确到某一个具体的元素。

45、css中link和@import的区别是什么：
1、link属于HTML标签，而@import是CSS提供的;

2、页面被加载的时，link会同时被加载，而@import引用的CSS会等到页面被加载完再加载;
3、import只在IE5以上才能识别，而link是HTML标签，无兼容问题;
4、优先级还是看引入的顺序
@import需要放在style里面的第一行，否则不起作用

46、清除浮动的方式：
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。
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。
3、父级div定义 伪类:after 和 zoom
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。 clearfix
4、父级div定义 overflow:hidden 或者 overflow:auto
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。
5、在浮动元素的后面一个元素加上clearfix，这时候格式的混乱发生在浮动元素的后面一个元素的后面一个元素。



其实，也就是，哪两个元素之间发生了混乱，就对前面那个元素加clearfix

47、block，inline和inlinke-block细节对比
- **display:block**

a、block元素会独占一行，多个block元素会各自新起一行。默认情况下，block元素宽度自动填满其父元素宽度。
b、block元素可以设置width,height属性。块级元素即使设置了宽度,仍然是独占一行。
c、block元素可以设置margin和padding属性。

- **display:inline**

a、inline元素不会独占一行，多个相邻的行内元素会排列在同一行里，直到一行排列不下，才会新换一行，其宽度随元素的内容而变化。
b、inline元素设置width,height属性无效。
c、inline元素的margin和padding属性，水平方向的padding-left, padding-right, margin-left, margin-right都产生边距效果；但竖直方向的padding-top, padding-bottom, margin-top, margin-bottom不会产生边距效果。

- **display:inline-block**

a、简单来说就是将对象呈现为inline对象，但是对象的内容作为block对象呈现。之后的内联对象会被排列在同一行内。比如我们可以给一个link（a元素）inline-block属性值，使其既具有block的宽度高度特性又具有inline的同行特性。
IE（低版本IE）本来是不支持inline-block的，所以在IE中对内联元素使用display:inline-block，理论上IE是不识别的，但使用display:inline-block在IE下会触发layout，从而使内联元素拥有了display:inline-block属性的表象。

48、优雅降级和渐进增强
优雅降级： Web站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，则代码会检查以确认它们是否能正常工作。由于IE独特的盒模型布局问题，针对不同版本的IE的hack实践过优雅降级了,为那些无法支持功能的浏览器增加候选方案，使之在旧式浏览器上以某种形式降级体验却不至于完全失效.
渐进增强： 从被所有浏览器支持的基本功能开始，逐步地添加那些只有新式浏览器才支持的功能,向页面增加无害于基础浏览器的额外样式和功能。当浏览器支持时，它们会自动地呈现出来并发挥作用。

49、浮动元素会引起的问题和解决办法
问题：
（1）父元素的高度无法被撑开，影响与父元素同级的元素
（2）与浮动元素同级的非浮动元素会跟随其后
（3）若非第一个元素浮动，则该元素之前的元素也需要浮动，否则会影响页面显示的结构
解决方法：
使用CSS中的clear:both;属性来清除元素的浮动可解决问题(2)、(3)，对于问题(1)，添

加如下样式，给父元素添加clearfix样式：

```
.clearfix:after{content: ".";display: block;height: 0;clear: both;visibility: hidden;}
.clearfix{display: inline-block;} /* for IE/Mac */
```

50、性能优化的方法

1、减少http请求次数：CSS Sprites, JS、CSS源码压缩、图片大小控制合适；网页Gzip（使用gzip压缩内容），CDN托管，data缓存 ，图片服务器。
2、前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数；
3、用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。
即尽量少用document.createElement，而先将语句拼接好，然后使用innerHTML。
4、当需要设置的样式很多时设置className而不是直接操作style
5、少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。
6、避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)
7、图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳
8、避免空的src和href
9、为文件头指定Expires
10、将CSS和JS放到外部文件中
11、避免跳转
12、使用GET来完成AJAX请求

网易笔记：
为什么优化：
提升网页响应速度
对搜索引擎、屏幕阅读器友好
提高可读性、可维护性
优化的方式：
1、减少请求
2、减小文件大小
3、提高可读性、可维护性
减少请求：
图片合并
css文件合并
多个css文件合并为一个
少量css样式内联
避免使用import的方式引入css文件
每个import语句产生一个请求，且同步

减小文件大小：

选择合适的图片格式

－PNG：半透明，例如小图标

－JPG：色彩比较绚丽，尺寸比较大

压缩图片

－ImageOption

－ImageAlpha

－JPEGmini

－……　　以上都是无损压缩

css值缩写

省略值为0的单位

颜色值最短表示

CSS选择器合并

文件压缩：

－YUI Compressor

－cssmin

提高页面性能：

加载顺序

建议css文件放在head里面。放在底部第一次看到的可能是没有样式的页面；

JS脚本放在body底部，因为JS的加载和执行会阻塞其他资源的加载和页面的执行加载，而

且js很多处理逻辑也是等整个页面加载完成以后再处理。

减少标签数量

选择器长度

避免耗性能的属性

图片设置宽高：

设置宽高且图片的实际宽高和设置的一致

所有表现用CSS表现

可读性、可维护性

规范：页面中的缩进的空格，变量名规范，文件名规范

语义化

尽量避免Hack

模块化

注释

51、初始化CSS样式
因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对CSS初始化往往会出现浏览器之间的页面显示差异。
当然，初始化样式会对SEO有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化。
每种浏览器都有一套默认的样式表，即user agent stylesheet，网页在没有指定的样式时，按浏览器内置的样式表来渲染。这是合理的，像word中也有一些预留样式，可以让我们的排版更美观整齐。不同浏览器甚至同一浏览器不同版本的默认样式是不同的。但这样会有很多兼容问题。
a、最简单的办法：（不推荐使用）*{margin: 0;padding: 0;}。
b、使用CSSReset可以将所有浏览器默认样式设置成一样。
c、normalize：也许有些cssreset过于简单粗暴，有点伤及无辜，normalize是另一个选择。bootstrap已经引用该css来重置浏览器默认样式，比普通的cssreset要精细一些，保留浏览器有用的默认样式，支持包括手机浏览器在内的超多浏览器，同时对HTML5元素、排版、列表、嵌入的内容、表单和表格都进行了一般化。
天猫 使用的css reset重置浏览器默认样式：
@charset "gb2312";body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt, dd, ul, ol, li, pre, form, fieldset, legend, button, input, textarea, th, td {margin: 0;padding: 0}body, button, input, select, textarea {font: 12px "microsoft yahei";line-height: 1.5;-ms-overflow-style: scrollbar}h1, h2, h3, h4, h5, h6 {font-size: 100%}ul, ol {list-style: none}a {text-decoration: none;cursor:pointer}a:hover {text-decoration: underline}img {border: 0}button, input, select, textarea {font-size: 100%}table {border-collapse: collapse;border-spacing: 0}.clear {clear:both}.fr {float:right}.fl {float:left}.block {display:block;text-indent:-999em}
SEO（Search Engine Optimization），汉译为搜索引擎优化，为近年来较为流行的网络营销方式，主要目的是增加特定关键字的曝光率以增加网站的能见度，进而增加销售的机会。分为站外SEO和站内
SEO两种。SEO的主要工作是通过了解各类搜索引擎如何抓取互联网页面、如何进行索引以及如何确定其对某一特定关键词的搜索结果排名等技术，来对网页进行相关的优化，使其提高搜索引擎排名，从
而提高网站访问量，最终提升网站的销售能力或宣传能力的技术。

52、浮动和它的工作原理？清除浮动的技巧?
浮动元素脱离文档流，不占据空间。浮动元素碰到包含它的边框或者浮动元素的边框停留。
(1)、使用空标签清除浮动。
这种方法是在所有浮动标签后面添加一个空标签 定义css clear:both. 弊端就是增加了无意义标签。
(2)、使用overflow。
给包含浮动元素的父标签添加css属性 overflow:auto; zoom:1; zoom:1用于兼容IE6。
(3)、使用after伪对象清除浮动。
该方法只适用于非IE浏览器。具体写法可参照以下示例。使用中需注意以下几点。一、该方法中必须为需要清除浮动元素的伪对象中设置 height:0，否则该元素会比实际高出若干像素；

53、CSS样式表根据所在网页的位置，可分为哪几种样式表
行内样式表，内嵌样式表，外部样式表

54、CSS中的刻度
a、特殊值0可以省略单位。例如：margin:0px可以写成margin:0
b、一些属性可能允许有负长度值，或者有一定的范围限制。如果不支持负长度值，那应该变换到能够被支持的最近的一个长度值。
c、长度单位包括：相对单位和绝对单位。
相对长度单位有： em, ex, ch, rem, vw, vh, vmax, vmin
绝对长度单位有： cm, mm, q, in, pt, pc, px
绝对长度单位：1in = 2.54cm = 25.4 mm = 72pt = 6pc = 96px
文本相对长度单位：em
相对长度单位是相对于当前对象内文本的字体尺寸，如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。(相对父元素的字体大小倍数)
body { font-size: 14px; }
h1 { font-size: 16px; }
.size1 p { font-size: 1em; }
.size2 p { font-size: 2em; }
.size3 p { font-size: 3em; }
文本相对长度单位：rem
rem是CSS3新增的一个相对单位（root em，根em），相对于根元素(即html元素)font-size计算值的倍数
只相对于根元素的大小
浏览器的默认字体大小为16像素，浏览器默认样式也称为user agent stylesheet，就是

所有浏览器内置的默认样式，多数是可以被修改的，但chrome不能直接修改，可以被用户样式覆盖。

55、em与rem的区别

rem

rem是CSS3新增的一个相对单位（root em，根em），相对于根元素(即html元素)font-size计算值的倍数

只相对于根元素的大小

rem（font size of the root element）是指相对于根元素的字体大小的单位。简单的说它就是一个相对单位。

作用：利用rem可以实现简单的响应式布局，可以利用html元素中字体的大小与屏幕间的比值设置font-size的值实现当屏幕分辨率变化时让元素也变化，以前的天猫tmall就使用这种办法

em

文本相对长度单位。相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸(默认16px)。(相对父元素的字体大小倍数)

em（font size of the element）是指相对于父元素的字体大小的单位。它与rem之间其实很相似，区别在。（相对是的HTML元素的字体大，默认16px）

em与rem的重要区别： 它们计算的规则一个是依赖父元素另一个是依赖根元素计算

56、box-sizing属性的的用法

设置或检索对象的盒模型组成模式

a、box-sizing:content-box： padding和border不被包含在定义的width和height之内。对象的实际宽度等于设置的width值和border、padding之和，即 (Element width = width + border + padding，但占有页面位置还要加上margin) 此属性表现为标准模式下的盒模型。

b、box-sizing:border-box： padding和border被包含在定义的width和height之内。对象的实际宽度就等于设置的width值，即使定义有border和padding也不会改变对象的实际宽度，即 (Element width = width) 此属性表现为怪异模式下的盒模型。IE盒模型。

57、浏览器标准模式和怪异模式之间的区别

从IE6开始，引入了Standards模式，标准模式中，浏览器尝试给符合标准的文档在规范

上的正确处理达到在指定浏览器中的程度。
在IE6之前CSS还不够成熟，所以IE5等之前的浏览器对CSS的支持很差， IE6将对CSS提供更好的支持，然而这时的问题就来了，因为有很多页面是基于旧的布局方式写的，而如果IE6 支持CSS则将令这些页面显示不正常，如何在即保证不破坏现有页面，又提供新的渲染机制呢?
在写程序时我们也会经常遇到这样的问题，如何保证原来的接口不变，又提供更强大的功能，尤其是新功能不兼容旧功能时。遇到这种问题时的一个常见做法是增加参数和分支，即当某个参数为真时，我们就使用新功能，而如果这个参数 不为真时，就使用旧功能，这样就能不破坏原有的程序，又提供新功能。IE6也是类似这样做的，它将DTD当成了这个“参数”，因为以前的页面大家都不会去写DTD，所以IE6就假定 如果写了DTD，就意味着这个页面将采用对CSS支持更好的布局，而如果没有，则采用兼容之前的布局方式。这就是Quirks模式（怪癖模式，诡异模式，怪异模式）。
区别：总体会有布局、样式解析和脚本执行三个方面的区别。

在“标准模式”(Standards Mode) 页面按照 HTML 与 CSS 的定义渲染，而在“怪异模式”(Quirks Mode)就是浏览器为了兼容很早之前针对旧版本浏览器设计、并未严格遵循 W3C 标准的网页而产生的一种页面渲染模式。浏览器基于页面中文件类型描述的存在以决定采用哪种渲染模式；如果存在一个完整的`DOCTYPE`则浏览器将会采用标准模式，而如果它缺失则浏览器将会采用怪异模式。
强烈建议阅读加深理解：[怪异模式（Quirks Mode）对 HTML 页面的影响](http://www.ibm.com/developerworks/cn/web/1310_shatao_quirks/)，这里列下浏览器标准模式和怪异模式的区别：
（1）盒模型：
在怪异模式下，盒模型为IE盒模型而非标准模式下的W3C 盒模型：在 IE 盒模型中，
box width = content width + padding left + padding right + border left + border right，
box height = content height + padding top + padding bottom + border top + border bottom。
而在 W3C 标准的盒模型中，box 的大小就是 content 的大小。
（2）图片元素的垂直对齐方式:
对于`inline`元素和`table-cell`元素，在 IE Standards Mode 下 vertical-align 属性默认取值为`baseline`。而当`inline`元素的内容只有图片时，如`table`的单元格`table-cell`。在 IE Quirks Mode 下，`table`单元格中的图片的 `vertical-align` 属性默认为 `bottom`，因此，在图片底部会有几像素的空间。
（3）`<table>`元素中的字体:
CSS 中，描述`font`的属性有`font-family`，`font-size`，`font-style`，`font-weigh`,上述属性都是可以继承的。而在 IE Quirks Mode 下，对于`table` 元素，字体的某些属性将不会从`body`或其他封闭元素继承到`table`中，特别是 `font-size`属性。
（4） 内联元素的尺寸:
在 IE Standards Mode 下，non-replaced inline 元素无法自定义大小，而在 IE Quirks Mode 下，定义这些元素的`width`和`height` 属性，能够影响该元素显示的大小尺

寸。
（5） 元素的百分比高度:
a、CSS 中对于元素的百分比高度规定如下，百分比为元素包含块的高度，不可为负值。如果包含块的高度没有显式给出，该值等同于“auto”（即取决于内容的高度）。所以百分比的高度必须在父元素有声明高度时使用。
b、当一个元素使用百分比高度时，在 IE Standards Mode 下，高度取决于内容的变化，而在 Quirks Mode 下，百分比高度则被正确应用。
（6） 元素溢出的处理:
在 IE Standard Mode 下，`overflow`取默认值 `visible`，即溢出可见，这种情况下，溢出内容不会被裁剪，呈现在元素框外。而在 Quirks Mode 下，该溢出被当做扩展 `box`来对待，即元素的大小由其内容决定，溢出不会被裁剪，元素框自动调整，包含溢出内容。
（7）使用margin:0 auto在standards模式下可以使元素水平居中，但在quirks模式下却会失效。

58、边距折叠
外边距折叠： 相邻的两个或多个外边距 (margin) 在垂直方向会合并成一个外边距（margin）
相邻： 没有被非空内容、padding、border 或 clear 分隔开的margin特性. 非空内容就是说这元素之间要么是兄弟关系或者父子关系
垂直方向外边距合并计算:
a、参加折叠的margin都是正值：取其中 margin 较大的值为最终 margin 值。
b、参与折叠的 margin 都是负值：取的是其中绝对值较大的，然后，从 0 位置，负向位移。
c、参与折叠的 margin 中有正值，有负值：先取出负 margin 中绝对值中最大的，然后，和正 margin 值中最大的 margin 相加。
注意，边距折叠不仅仅在设置两个边距的时候会进行折叠。且，在子元素上设置margin-top的时候，父元素和子元素一起下移。
例如:

```
<div style="width: 400px;background-color: red;height: 100px;">
    <div style="margin-top:auto;background-color: green;width: 100px;height: 50px">sss</div>
    <!-- <span style="background: green">sdfs</span> -->
</div>

<div align="center" style="width: 400px;background-color: red;height: 100px;">
```

```
        <div style="background-color: green;margin-top: 20px">sdfs</div>
    </div>
```

如果给下面的父div增加的样式是border，就不会出现这种情况，下移的只是子div

59、隐藏元素的方式
a、使用CSS的display:none，不会占有原来的位置
b、使用CSS的visibility:hidden，会占有原来的位置
c、使用HTML5中的新增属性**hidden="hidden"**，不会占有原来的位置

60、BFC与IFC
(1)、什么是BFC与IFC
a、BFC（Block Formatting Context）即“块级格式化上下文”， IFC（Inline Formatting Context）即行内格式化上下文。常规流（也称标准流、普通流）是一个文档在被显示时最常见的布局形态。一个框在常规流中必须属于一个格式化上下文，你可以把BFC想象成一个大箱子，箱子外边的元素将不与箱子内的元素产生作用。
b、BFC是W3C CSS 2.1 规范中的一个概念，它决定了元素如何对其内容进行定位，以及与其他元素的关系和相互作用。当涉及到可视化布局的时候，Block Formatting Context提供了一个环境，HTML元素在这个环境中按照一定规则进行布局。一个环境中的元素不会影响到其它环境中的布局。比如浮动元素会形成BFC，浮动元素内部子元素的主要受该浮动元素影响，两个浮动元素之间是互不影响的。也可以说BFC就是一个作用范围。

c、在普通流中的 Box(框) 属于一种 formatting context(格式化上下文)，类型可以是 block，或者是 inline，但不能同时属于这两者。并且， Block boxes(块框) 在 block formatting context(块格式化上下文) 里格式化， Inline boxes(块内框) 则在 Inline Formatting Context(行内格式化上下文) 里格式化。
(2)、如何产生BFC
当一个HTML元素满足下面条件的任何一点，都可以产生Block Formatting Context：
a、float的值不为none
b、overflow的值不为visible
c、display的值为table-cell, table-caption, inline-block中的任何一个
d、position的值不为relative和static
CSS3触发BFC方式则可以简单描述为：在元素定位非static，relative的情况下触发，float也是一种定位方式。
(3)、BFC的作用与特点
a、不和浮动元素重叠，清除外部浮动，阻止浮动元素覆盖
如果一个浮动元素后面跟着一个非浮动的元素，那么就会产生一个重叠的现象。常规流（也称标准流、普通流）是一个文档在被显示时最常见的布局形态，当float不为none时，position为absolute、fixed时元素将脱离标准流。

BFC（Block Formatting Context）叫做“块级格式化上下文”。BFC的布局规则例如以下：
1.内部的盒子会在垂直方向，一个个地放置；
2.盒子垂直方向的距离由margin决定，属于同一个BFC的两个相邻Box的上下margin会发生重叠；
3.每一个元素的左边，与包括的盒子的左边相接触，即使存在浮动也是如此；
4.BFC的区域不会与float重叠；
5.BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。反之也如此；
6.计算BFC的高度时，浮动元素也参与计算。

IFC布局规则：

1. 框会从包含块的顶部开始，一个接一个地水平摆放。
2. 摆放这些框的时候，它们在水平方向上的外边距、边框、内边距所占用的空间都会被考虑在内。在垂直方向上，这些框可能会以不同形式来对齐：它们可能会把底部或顶部对齐，也可能把其内部的文本基线对齐。能把在一行上的框都完全包含进去的一个矩形区域，被称为该行的行框。水平的margin、padding、border有效，垂直无效。不能指定宽高。
3. 行框的宽度是由包含块和存在的浮动来决定。行框的高度由行高计算这一章所描述的规则来决定。

61、页面中使用定位(position)
使用css布局position非常重要，语法如下：
position：static | relative | absolute | fixed | center | page | sticky
默认值：static；center、page、sticky是CSS3中新增加的值。
(1)、static
可以认为静态的，默认元素都是静态的定位，对象遵循常规流。此时4个定位偏移属性不会被应用，也就是使用left，right，bottom，top将不会生效。
(2)、relative
相对定位，对象遵循常规流，并且参照自身在常规流中的位置通过top，right，bottom，left这4个定位偏移属性进行偏移时不会影响常规流中的任何元素。其他元素还在原来的位置。
(3)、absolute
a、绝对定位，对象脱离常规流，此时偏移属性参照的是离自身最近的定位祖先元素，如果没有定位的祖先元素，则一直回溯到body元素。盒子的偏移位置不影响常规流中的任何元素，其margin不与其他任何margin折叠。其他元素会往上重叠在一起。
b、元素定位参考的是离自身最近的定位祖先元素，要满足两个条件，第一个是自己的祖先元素，可以是父元素也可以是父元素的父元素，一直找，如果没有则选择body为对照对象。第二个条件是要求祖先元素必须定位，通俗说就是position的属性值为非static都行。
(4)、fixed
固定定位，与absolute一致，但偏移定位是以窗口为参考。当出现滚动条时，对象不会随着滚动。
(5)、center
与absolute一致，但偏移定位是以定位祖先元素的中心点为参考。盒子在其包含容器垂直水平居中。（CSS3）
(6)、page
与absolute一致。元素在分页媒体或者区域块内，元素的包含块始终是初始包含块，否则取决于每个absolute模式。（CSS3）
(7)、sticky
对象在常态时遵循常规流。它就像是relative和fixed的合体，当在屏幕中时按常规流排版，当卷动到屏幕外时则表现如fixed。该属性的表现是现实中你见到的吸附效果。
（CSS3）

62、多个元素重叠问题
使用z-index属性可以设置元素的层叠顺序
z-index属性
语法：z-index: auto | <integer>
默认值：auto

适用于：定位元素。即定义了position为非static的元素
取值：
auto： 元素在当前层叠上下文中的层叠级别是0。元素不会创建新的局部层叠上下文，除非它是根元素。
整数： 用整数值来定义堆叠级别。可以为负值。 说明：
检索或设置对象的层叠顺序。
z-index用于确定元素在当前层叠上下文中的层叠级别，并确定该元素是否创建新的局部层叠上下文。
当多个元素层叠在一起时，数字大者将显示在上面。

63、overfloa:hidden是否形成新的块级格式化上下文
会形成，触发BFC的条件有：
- float的值不为none。
- overflow的值不为visible。
- display的值为table-cell, table-caption, inline-block 中的任何一个。
- position的值不为relative 和static。

64、CSS实现宽高1：1

首先需要知道，一个元素的 `padding`，如果值是一个百分比，那这个百分比是相对于其父元素的宽度而言的，即使对于 `padding-bottom` 和 `padding-top` 也是如此。

另外，在计算 Overflow 时，是将元素的内容区域（即 `width` / `height` 对应的区域）和 Padding 区域一起计算的。换句话说，即使将元素的 `overflow` 设置为 `hidden`，“溢出”到 Padding 区域的内容也会照常显示。

综上两条所述，我们可以使用 `padding-bottom` 来代替 `height` 来实现高度与宽度成比例的效果。因为 item 元素的宽度是其父元素宽度的 100%，所以我们将 `padding-bottom` 设置为它的 1 倍，即 100%。同时将其 `height` 设置为 `0` 以使元素的“高度”等于 `padding-bottom` 的值，从而实现需要的效果。

```
<div style="width: 50%;background-color: green;height: 500px;">
    <div style="width: 100%;height: 0;background-color: red;padding-bottom: 100%;"></div>
</div>
```