## 浏览器页面有哪三层构成，分别是什么，作用是什么?

构成：结构层、表示层、行为层

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。

## HTML5的优点与缺点？

优点：

1. 网络标准统一、HTML5本身是由W3C推荐出来的。
2. 多设备、跨平台
3. 即时更新。
4. 提高可用性和改进用户的友好体验；
5. 有几个新的标签，这将有助于开发人员定义重要的内容；
6. 可以给站点带来更多的多媒体元素(视频和音频)；
7. 可以很好的替代Flash和Silverlight；
8. 涉及到网站的抓取和索引的时候，对于SEO很友好；
9. 被大量应用于移动应用程序和游戏。

缺点：

1. 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。
2. 完善性：许多特性各浏览器的支持程度也不一样。
3. 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
4. 性能：某些平台上的引擎问题导致HTML5性能低下。
5. 浏览器兼容性：最大缺点，IE9以下浏览器几乎全军覆没。

## Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?

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。
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。

## HTML5有哪些新特性、移除了哪些元素？

Html5新增了 27 个元素，废弃了 16 个元素，根据现有的标准规范，把 HTML5 的元素按优先级定义为结构性属性、级块性元素、行内语义性元素和交互性元素 4 大类。

结构性元素主要负责web上下文结构的定义

section：在 web 页面应用中，该元素也可以用于区域的章节描述。

header：页面主体上的头部， header 元素往往在一对 body 元素中。

footer：页面的底部（页脚），通常会标出网站的相关信息。

nav：专门用于菜单导航、链接导航的元素，是 navigator 的缩写。

article：用于表现一篇文章的主体内容，一般为文字集中显示的区域。

级块性元素主要完成web页面区域的划分，确保内容的有效分割。

aside：用于表达注记、贴士、侧栏、摘要、插入的引用等作为补充主体的内容。

figure：是对多个元素进行组合并展示的元素，通常与 ficaption 联合使用。

code：表示一段代码块。

dialog：用于表达人与人之间的对话，该元素包含 dt 和 dd 这两个组合元素， dt 用于表示说话者，而 dd 用来表示说话内容。

行内语义性元素主要完成web页面具体内容的引用和描述，是丰富内容展示的基础。

meter：表示特定范围内的数值，可用于工资、数量、百分比等。

time：表示时间值。

progress：用来表示进度条，可通过对其 max 、 min 、 step 等属性进行控制，完成对进度的表示和监事。

video：视频元素，用于支持和实现视频文件的直接播放，支持缓冲预载和多种视频媒体格式。

audio：音频元素，用于支持和实现音频文件的直接播放，支持缓冲预载和多种音频媒体格式。

交互性元素主要用于功能性的内容表达，会有一定的内容和数据的关联，是各种事件的基础。

details：用来表示一段具体的内容，但是内容默认可能不显示，通过某种手段（如单击）与 legend 交互才会显示出来。

datagrid：用来控制客户端数据与显示，可以由动态脚本及时更新。

menu：主要用于交互菜单（曾被废弃又被重新启用的元素）。

command：用来处理命令按钮。

## 你做的网页在哪些浏览器测试过,这些浏览器的内核分别是什么?

a、 IE: trident 内核

b、 Firefox ： gecko 内核

c、 Safari:webkit 内核

d、 Opera: 以前是 presto 内核， Opera 现已改用 Google Chrome 的 Blink 内核

e、 Chrome:Blink( 基于 webkit ， Google 与 Opera Software 共同开发 )

## 每个HTML文件里开头都有个很重要的东西，Doctype，知道这是干什么的吗？

声明位于文档中的最前面的位置，处于标签之前。此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范。（重点：告诉浏览器按照何种规范解析页面）

## 说说你对HTML5认识?（是什么,为什么）

是什么：

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。 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。

## 对WEB标准以及W3C的理解与认识?

标签闭合、标签小写、不乱嵌套、提高搜索机器人搜索几率、使用外 链css和 js 脚本、结构行为表现的分离、

文件下载与页面速度更快、内容能被更多的用户所访问、内容能被更广泛的设备所访问、更少的代码和组件，

容易维 护、改版方便，不需要变动页面内容、提供打印版本而不需要复制内容、提高网站易用性。

## HTML5行内元素有哪些,块级元素有哪些, 空元素有哪些?

(1)行内元素

a - 锚点

\* abbr - 缩写

\* acronym - 首字

\* b - 粗体 ( 不推荐 )

\* bdo - bidi override

\* big - 大字体

\* br - 换行

\* cite - 引用

\* code - 计算机代码 ( 在引用源码的时候需要 )

\* dfn - 定义字段

\* em - 强调

\* font - 字体设定 ( 不推荐 )

\* i - 斜体

\* img - 图片

\* input - 输入框

\* kbd - 定义键盘文本

\* label - 表格标签

\* q - 短引用

\* s - 中划线 ( 不推荐 )

\* samp - 定义范例计算机代码

\* select - 项目选择

\* small - 小字体文本

\* span - 常用内联容器，定义文本内区块

\* strike - 中划线

\* strong - 粗体强调

\* sub - 下标

\* sup - 上标

\* textarea - 多行文本输入框

\* tt - 电传文本

\* u - 下划线

\* var - 定义变量

(2)块元素 (block element)

\* address - 地址

\* blockquote - 块引用

\* center - 居中对齐块

\* dir - 目录列表

\* div - 常用块级容易，也是 css layout 的主要标签

\* dl - 定义列表

\* fieldset - form控制组

\* form - 交互表单

\* h1 - 大标题

\* h2 - 副标题

\* h3 - 3级标题

\* h4 - 4级标题

\* h5- 5级标题

\* h6 - 6级标题

\* hr - 水平分隔线

\* isindex - input prompt

\* menu - 菜单列表

\* noframes - frames可选内容，（对于不支持 frame 的浏览器显示此区块内容

\* noscript - ）可选脚本内容（对于不支持 script 的浏览器显示此内容）

\* ol - 排序表单

\* p - 段落

\* pre - 格式化文本

\* table - 表格

\* ul - 非排序列表

可变元素

可变元素为根据上下文语境决定该元素为块元素或者内联元素。

\* applet - java applet

\* button - 按钮

\* del - 删除文本

\* iframe - inline frame

\* ins - 插入的文本

\* map - 图片区块 (map)

\* object - object对象

\* script - 客户端脚本

(3)空元素 ( 在 HTML[1] 元素中，没有内容的 HTML 元素被称为空元素 )

<br/> //换行

<hr> //分隔线

<input> //文本框等

<img> //图片

<link> <meta>

## 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。

## 请你描述一下 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 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）

## 说说你对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。

(2)为什么要语义化？

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；

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；

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

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；

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

<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

## link和@import的区别?

XML/HTML代码

<link rel='stylesheet' rev='stylesheet' href='CSS文件 ' type='text/css' media='all' />

XML/HTML代码

<style type='text/css' media='screen'>

@import url('CSS文件 ');

</style>

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

## 说说你对SVG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。

## HTML全局属性(global attribute)有哪些?

accesskey 规定激活元素的快捷键；

class 规定元素的一个或多个类名（引用样式表中的类）；

contenteditable 规定元素内容是否可编辑；

contextmenu 规定元素的上下文菜单。上下文菜单在用户点击元素时显示。

data-\* 用于存储页面或应用程序的私有定制数据。

dir 规定元素中内容的文本方向。

draggable 规定元素是否可拖动。

dropzone 规定在拖动被拖动数据时是否进行复制、移动或链接。

hidden  样式上会导致元素不显示，但是不能用这个属性实现样式。

id 规定元素的唯一 id。

lang 规定元素内容的语言。

spellcheck 规定是否对元素进行拼写和语法检查。

style 规定元素的CSS行内元素。

tabindex 规定元素的tab键次序。

title 规定有关元素的额外信息。

translate 规定是否应该翻译元素内容。

## 说说超链接target属性的取值和作用？

target这个属性指定所链接的页面在浏览器窗口中的打开方式。

它的参数值主要有：

a、 \_blank ：在新浏览器窗口中打开链接文件

b、 \_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。

c、 \_self ：在同一框架或窗口中打开所链接的文档。此参数为默认值，通常不用指定。但是我不太理解。

d、 \_top ：在当前的整个浏览器窗口中打开所链接的文档，因而会删除所有框架。

## `data-`属性的作用是什么？

`data-`为H5新增的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 :

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。 所有主流浏览器都支持 data-\* 属性。

即：当没有合适的属性和元素时，自定义的 data 属性是能够存储页面或 App 的私有的自定义数据。

## 介绍一下你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。

## 常见的浏览器内核有哪些？

view sourceprint?

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） ]

## iframe有那些缺点？

\*iframe会阻塞主页面的 Onload 事件；

\*搜索引擎的检索程序无法解读这种页面，不利于 SEO;

\*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。

## Label的作用是什么，是怎么用的？

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'/>

## 如何实现浏览器内多个标签页之间的通信?

WebSocket、 SharedWorker ；

也可以调用localstorage、 cookies 等本地存储方式；

localstorage另一个浏览上下文里被添加、修改或删除时，它都会触发一个事件，

我们通过监听事件，控制它的值来进行页面信息通信；

注意quirks： Safari 在无痕模式下设置 localstorge 值时会抛出 QuotaExceededError 的异常；

## 如何在页面上实现一个圆形的可点击区域？

a、 map+area 或者 svg

b、 border-radius

c、纯 js 实现 需要求一个点在不在圆上简单算法、获取鼠标坐标等等

## 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： <strong> 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

Semantic Style Elements -- 语义样式标签

strong, em, ins, del, code

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。

## 实现不使用 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？

<div style="width:100%;height:1px;background-color:black"></div>

## HTML5标签的作用?(用途)

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

## 简述一下src与href的区别？

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。

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。

## 谈谈你对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。

## 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：

双向通信与消息推送

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。

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。

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 。

长连接：在页面里嵌入一个隐蔵iframe，将这个隐蔵 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。

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

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。

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。

## 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.

## 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。

## 表单提交中Get和Post方式的区别？

(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。

## HTML5 有哪些新增的表单元素？

HTML5 新增了很多表单元素让开发者构建更优秀的 Web 应用程序，主要有：

datalist

keygen

output

## HTML5 存储类型有什么区别？

HTML5 能够本地存储数据，在之前都是使用 cookies 使用的。 HTML5 提供了下面两种本地存储方案：

· localStorage 用于持久化的本地存储，数据永远不会过期，关闭浏览器也不会丢失。

· sessionStorage 同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储

## 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：

<!doctype html>

<html manifest=”example.appcache”>

…..

</html>

与传统浏览器缓存相比，它不强制用户访问的网站内容被缓存。

## HTML5 Canvas 元素有什么用？

Canvas 元素用于在网页上绘制图形，该元素标签强大之处在于可以直接在 HTML 上进行图形操作，

<canvas id=” canvas1 ″ width= ” 300 ″ height= ” 100 ″ >

</canvas>

## 除了 audio 和 video，HTML5 还有哪些媒体标签？

HTML5 对于多媒体提供了强有力的支持，除了 audio 和 video 标签外，还支持以下标签：

<embed> 标签定义嵌入的内容，比如插件。

<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。 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>

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

## HTML5 中如何嵌入视频？

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>

## HTML5 中如何嵌入音频？

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

<audio controls>

<source src=” jamshed.mp3 ″ type= ” audio/mpeg ” >

Your browser does’ nt support audio embedding feature.

</audio>

## 新的 HTML5 文档类型和字符集是？

HTML5 文档类型很简单：

<!doctype html>

HTML5 使用 UTF-8 编码示例：

<meta charset=” UTF-8 ″ >

## 解释一下CSS的盒子模型？

a、标准的css盒子模型：宽度=内容的宽度+边框的宽度+加上内边具的宽度

b、网页设计中常听的属性名：内容(content)、填充(padding)、边框(border)、边界(margin)， CSS盒子模式都具备这些属性。

c、这些属性我们可以把它转移到我们日常生活中的盒子（箱子）上来理解，日常生活中所见的盒子也就是能装东西的一种箱子，也具有这些属性，所以叫它盒子模式。CSS盒子模型就是在网页设计中经常用到的CSS技术所使用的一种思维模型。

回答二：

标准的盒模型：width = content

IE盒模型：width = content+padding-Left+padding-right+border-left + border-right

## 要动态改变层中内容可以使用的方法？

innerHTML，innerText

## 常见浏览器兼容性问题与解决方案？

(1)浏览器兼容问题一：不同浏览器的标签默认的外补丁和内补丁不同

问题症状：随便写几个标签，不加样式控制的情况下，各自的margin 和padding差异较大。

碰到频率:100%

解决方案：CSS里 \*{margin:0;padding:0;}

备注：这个是最常见的也是最易解决的一个浏览器兼容性问题，几乎所有的CSS文件开头都会用通配符\*来设置各个标签的内外补丁是0。

(2)浏览器兼容问题二：块属性标签float后，又有横行的margin情况下，在IE6显示margin比设置的大

问题症状:常见症状是IE6中后面的一块被顶到下一行

碰到频率：90%（稍微复杂点的页面都会碰到，float布局最常见的浏览器兼容问题）

解决方案：在float的标签样式控制中加入 display:inline;将其转化为行内属性

备注：我们最常用的就是div+CSS布局了，而div就是一个典型的块属性标签，横向布局的时候我们通常都是用div float实现的，横向的间距设置如果用margin实现，这就是一个必然会碰到的兼容性问题。

(3)浏览器兼容问题三：设置较小高度标签（一般小于10px），在IE6，IE7，遨游中高度超出自己设置高度

问题症状：IE6、7和遨游里这个标签的高度不受控制，超出自己设置的高度

碰到频率：60%

解决方案：给超出高度的标签设置overflow:hidden;或者设置行高line-height 小于你设置的高度。

备注：这种情况一般出现在我们设置小圆角背景的标签里。出现这个问题的原因是IE8之前的浏览器都会给标签一个最小默认的行高的高度。即使你的标签是空的，这个标签的高度还是会达到默认的行高。

(4)浏览器兼容问题四：行内属性标签，设置display:block后采用float布局，又有横行的margin的情况，IE6间距bug

问题症状：IE6里的间距比超过设置的间距

碰到几率：20%

解决方案 ： 在display:block;后面加入display:inline;display:table;

备注：行内属性标签，为了设置宽高，我们需要设置display:block;(除了input标签比较特殊)。在用float布局并有横向的margin后，在IE6下，他就具有了块属性float后的横向margin的bug。不过因为它本身就是行内属性标签，所以我们再加上display:inline的话，它的高宽就不可设了。这时候我们还需要在display:inline后面加入display:talbe。

(5) 浏览器兼容问题五：图片默认有间距

问题症状：几个img标签放在一起的时候，有些浏览器会有默认的间距，加了问题一中提到的通配符也不起作用。

碰到几率：20%

解决方案：使用float属性为img布局

备注 ： 因为img标签是行内属性标签，所以只要不超出容器宽度，img标签都会排在一行里，但是部分浏览器的img标签之间会有个间距。去掉这个间距使用float是正道。（我的一个学生使用负margin，虽然能解决，但负margin本身就是容易引起浏览器兼容问题的用法，所以我禁止他们使用）

(6) 浏览器兼容问题六：标签最低高度设置min-height不兼容

问题症状：因为min-height本身就是一个不兼容的CSS属性，所以设置min-height时不能很好的被各个浏览器兼容

碰到几率：5%

解决方案：如果我们要设置一个标签的最小高度200px，需要进行的设置为：{min-height:200px; height:auto !important; height:200px; overflow:visible;}

备注：在B/S系统前端开时，有很多情况下我们又这种需求。当内容小于一个值（如300px）时。容器的高度为300px；当内容高度大于这个值时，容器高度被撑高，而不是出现滚动条。这时候我们就会面临这个兼容性问题。

(7)浏览器兼容问题七：透明度的兼容CSS设置

一般在ie中用的是filter:alpha(opacity=0);这个属性来设置div或者是块级元素的透明度，而在firefox中，一般就是直接使用opacity:0,对于兼容的，一般的做法就是在书写css样式的将2个都写上就行，就能实现兼容

## 列出display的值并说明他们的作用？

display： none | inline | block | list-item | inline-block | table | inline-table | table-caption | table-cell | table-row | table-row-group | table-column | table-column-group | table-footer-group | table-header-group | run-in | box | inline-box | flexbox | inline-flexbox | flex | inline-flex

默认值：inline

none： 隐藏对象。与visibility属性的hidden值不同，其不为被隐藏的对象保留其物理空间

inline： 指定对象为内联元素。

block： 指定对象为块元素。

list-item： 指定对象为列表项目。

inline-block： 指定对象为内联块元素。（CSS2）

table： 指定对象作为块元素级的表格。类同于html标签<table>（CSS2）

inline-table： 指定对象作为内联元素级的表格。类同于html标签<table>（CSS2）

table-caption： 指定对象作为表格标题。类同于html标签<caption>（CSS2）

table-cell： 指定对象作为表格单元格。类同于html标签<td>（CSS2）

table-row： 指定对象作为表格行。类同于html标签<tr>（CSS2）

table-row-group： 指定对象作为表格行组。类同于html标签<tbody>（CSS2）

table-column： 指定对象作为表格列。类同于html标签<col>（CSS2）

table-column-group： 指定对象作为表格列组显示。类同于html标签<colgroup>（CSS2）

table-header-group： 指定对象作为表格标题组。类同于html标签<thead>（CSS2）

table-footer-group： 指定对象作为表格脚注组。类同于html标签<tfoot>（CSS2）

run-in： 根据上下文决定对象是内联对象还是块级对象。（CSS3）

box： 将对象作为弹性伸缩盒显示。（伸缩盒最老版本）（CSS3）

inline-box： 将对象作为内联块级弹性伸缩盒显示。（伸缩盒最老版本）（CSS3）

flexbox： 将对象作为弹性伸缩盒显示。（伸缩盒过渡版本）（CSS3）

inline-flexbox： 将对象作为内联块级弹性伸缩盒显示。（伸缩盒过渡版本）（CSS3）

flex： 将对象作为弹性伸缩盒显示。（伸缩盒最新版本）（CSS3）

inline-flex： 将对象作为内联块级弹性伸缩盒显示。（伸缩盒最新版本）（CSS3）

## 如何居中div, 如何居中一个浮动元素?

(1)、非浮动元素居中：可以设置 margin:0 auto 令其居中, 定位 ,父级元素text-align:center等等

(2)、浮动元素居中:

方法一:设置当前div的宽度，然后设置margin-left:50%; position:relative; left:-250px;其中的left是宽度的一半。

方法二:父元素和子元素同时左浮动，然后父元素相对左移动50%，再然后子元素相对左移动-50%。

方法三:position定位等等。

## 请列举几种清除浮动的方法(至少两种)?

(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，让人感觉很不好

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

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。

## 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。

补充说明

a、一般我们会用display:block，display:inline或者display:inline-block来调整元素的布局级别，其实display的参数远远不止这三种，仅仅是比较常用而已。

b、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。

## 什么叫优雅降级和渐进增强？

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

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

## 说说浮动元素会引起的问题和你的解决办法

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 \*/

清除浮动的几种方法：

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

(2)、使用after伪类

#parent:after{

  content:" ";

  height:0;

  visibility:hidden;

  display:block;

  clear:both;

}

(3)、浮动外部元素

(4)、设置`overflow`为`hidden`或者auto

## 你有哪些性能优化的方法？

回答一：

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

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

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

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

（5）、少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。

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

（7）、图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

回答二：

(1)、减少HTTP请求次数

(2)、使用CDN

(3)、避免空的src和href

(4)、为文件头指定Expires

(5)、使用gzip压缩内容

(6)、把CSS放到顶部

(7)、把JS放到底部

(8)、避 免使用CSS表达式

(9)、将CSS和JS放到外部文件中

(10)、避免跳转

(11)、可缓存的AJAX

(12)、使用GET来完成AJAX请求

## 为什么要初始化CSS样式？

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

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

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

## 解释下浮动和它的工作原理？清除浮动的技巧？

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

(1)、使用空标签清除浮动。

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

(2)、使用overflow。

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

(3)、使用after伪对象清除浮动。

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

## 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

## 谈谈你对CSS中刻度的认识？

在CSS中刻度是用于设置元素尺寸的单位。

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。

## 请你说说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

## 请你说说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。

## 浏览器标准模式和怪异模式之间的区别是什么？

所谓的标准模式是指，浏览器按W3C标准解析执行代码；怪异模式则是使用浏览器自己的方式解析执行代码，因为不同浏览器解析执行的方式不一样，所以我们称之为怪异模式。浏览器解析时到底使用标准模式还是怪异模式，与你网页中的DTD声明直接相关，DTD声明定义了标准文档的类型（标准模式解析）文档类型，会使浏览器使用相应的方式加载网页并显示，忽略DTD声明,将使网页进入怪异模式(quirks mode)。

## 怪异Quirks模式是什么，它和标准Standards模式有什么区别？

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。

盒模型： 在W3C标准中，如果设置一个元素的宽度和高度，指的是元素内容的宽度和高度，而在Quirks 模式下，IE的宽度和高度还包含了padding和border。

设置行内元素的高宽： 在Standards模式下，给<span>等行内元素设置wdith和height都不会生效，而在quirks模式下，则会生效。

设置百分比的高度： 在standards模式下，一个元素的高度是由其包含的内容来决定的，如果父元素没有设置百分比的高度，子元素设置一个百分比的高度是无效的用margin:0 auto设置水平居中：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。

（还有很多，答出什么不重要，关键是看他答出的这些是不是自己经验遇到的，还是说都是看文章看的，甚至完全不知道。）

## 说说你对边距折叠的理解?

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。

## 内联与块级标签有何区别？

Html中的标签默认主要分为两大类型，一类为块级元素，另一类是行内元素，许多人也把行内称为内联，所以叫内联元素，其实就是一个意思。为了很好的布局，必须理解它们间的区别。

## 说说隐藏元素的方式有哪些？

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

## 为什么重置浏览器默认样式，如何重置默浏览器认样式？

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}

## 谈谈你对BFC与IFC的理解？(是什么，如何产生，作用)

(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。

## 说说你对页面中使用定位(position)的理解？

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。

(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。

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）

## 如何解决多个元素重叠问题？

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。

## 页面布局的方式有哪些？

方式：双飞翼、多栏、弹性、流式、瀑布流、响应式布局 （1）、双飞翼布局 经典三列布局，也叫做圣杯布局【Holy Grail of Layouts】是Kevin Cornell在2006年提出的一个布局模型概念，在国内最早是由淘宝UED的工程师传播开来，在中国也有叫法是双飞翼布局，它的布局要求有几点： a、三列布局，中间宽度自适应，两边定宽； b、中间栏要在浏览器中优先展示渲染； c、允许任意列的高度最高； d、要求只用一个额外的DIV标签； e、要求用最简单的CSS、最少的HACK语句； 在不增加额外标签的情况下，圣

方式：双飞翼、多栏、弹性、流式、瀑布流、响应式布局

（1）、双飞翼布局

经典三列布局，也叫做圣杯布局【Holy Grail of Layouts】是Kevin Cornell在2006年提出的一个布局模型概念，在国内最早是由淘宝UED的工程师传播开来，在中国也有叫法是双飞翼布局，它的布局要求有几点：

a、三列布局，中间宽度自适应，两边定宽；

b、中间栏要在浏览器中优先展示渲染；

c、允许任意列的高度最高；

d、要求只用一个额外的DIV标签；

e、要求用最简单的CSS、最少的HACK语句；

在不增加额外标签的情况下，圣杯布局已经非常完美，圣杯布局使用了相对定位，以后布局是有局限性的，而且宽度控制要改的地方也多。在淘宝UED（User Experience Design）探讨下，增加多一个div就可以不用相对布局了，只用到了浮动和负边距，这就是我们所说的双飞翼布局。

(2)、多栏布局

a、栏栅格系统：就是利用浮动实现的多栏布局，在bootstrap中用的非常多。

b、多列布局：栅格系统并没有真正实现分栏效果（如word中的分栏），CSS3为了满足这个要求增加了多列布局模块

(3)、弹性布局（Flexbox）

CSS3引入了一种新的布局模式——Flexbox布局，即伸缩布局盒模型（Flexible Box），用来提供一个更加有效的方式制定、调整和分布一个容器里项目布局，即使它们的大小是未知或者动态的，这里简称为Flex。

Flexbox布局常用于设计比较复杂的页面，可以轻松的实现屏幕和浏览器窗口大小发生变化时保持元素的相对位置和大小不变，同时减少了依赖于浮动布局实现元素位置的定义以及重置元素的大小。

Flexbox布局在定义伸缩项目大小时伸缩容器会预留一些可用空间，让你可以调节伸缩项目的相对大小和位置。例如，你可以确保伸缩容器中的多余空间平均分配多个伸缩项目，当然，如果你的伸缩容器没有足够大的空间放置伸缩项目时，浏览器会根据一定的比例减少伸缩项目的大小，使其不溢出伸缩容器。

综合而言，Flexbox布局功能主要具有以下几点：

a、屏幕和浏览器窗口大小发生改变也可以灵活调整布局；

b、可以指定伸缩项目沿着主轴或侧轴按比例分配额外空间（伸缩容器额外空间），从而调整伸缩项目的大小；

c、可以指定伸缩项目沿着主轴或侧轴将伸缩容器额外空间，分配到伸缩项目之前、之后或之间；

d、可以指定如何将垂直于元素布局轴的额外空间分布到该元素的周围；

e、可以控制元素在页面上的布局方向；

f、可以按照不同于文档对象模型（DOM）所指定排序方式对屏幕上的元素重新排序。也就是说可以在浏览器渲染中不按照文档流先后顺序重排伸缩项目顺序。

(4)、瀑布流布局

瀑布流布局是流式布局的一种。是当下比较流行的一种网站页面布局，视觉表现为参差不齐的多栏布局，随着页面滚动条向下滚动，这种布局还会不断加载数据块并附加至当前尾部。最早采用此布局的网站是Pinterest，逐渐在国内流行开来。

优点

a、有效的降低了界面复杂度，节省了空间：我们不再需要臃肿复杂的页码导航链接或按钮了。

b、对触屏设备来说，交互方式更符合直觉：在移动应用的交互环境当中，通过向上滑动进行滚屏的操作已经成为最基本的用户习惯，而且所需要的操作精准程度远远低于点击链接或按钮。

c、更高的参与度：以上两点所带来的交互便捷性可以使用户将注意力更多的集中在内容而不是操作上，从而让他们更乐于沉浸在探索与浏览当中。

缺点

a、有限的用例：

无限滚动的方式只适用于某些特定类型产品当中一部分特定类型的内容。

例如，在电商网站当中，用户时常需要在商品列表与详情页面之间切换，这种情况下，传统的、带有页码导航的方式可以帮助用户更稳妥和准确的回到某个特定的列表页面当中。

b、额外的复杂度：

那些用来打造无限滚动的JS库虽然都自称很容易使用，但你总会需要在自己的产品中进行不同程度的定制化处理，以满足你们自己的需求;另外这些JS库在浏览器和设备兼容性等方面的表现也参差不齐，你必须做好充分的测试与调整工作。

c、再见了，页脚：

如果使用了比较典型的无限滚动加载模式，这就意味着你可以和页脚说拜拜了。

最好考虑一下页脚对于你的网站，特别是用户的重要性;如果其中确实有比较重要的内容或链接，那么最好换一种更传统和稳妥的方式。

千万不要耍弄你的用户，当他们一次次的浏览到页面底部，看到页脚，却因为自动加载的内容突然出现而无论如何都无法点击页脚中的链接时，他们会变的越发愤怒。

d、集中在一页当中动态加载数据，与一页一页的输出相比，究竟那种方式更利于SEO，这是你必须考虑的问题。对于某些以类型网站来说，在这方面进行冒险是很不划算的。

e、关于页面数量的印象：

其实站在用户的角度来看，这一点并非负面;不过，如果对于你的网站来说，通过更多的内容页面展示更多的相关信息(包括广告)是很重要的策略，那么单页无限滚动的方式对你并不适用。

(5)、流式布局（Fluid）

固定布局和流式布局在网页设计中最常用的两种布局方式。固定布局能呈现网页的原始设计效果，流式布局则不受窗口宽度影响，流式布局使用百分比宽度来限定布局元素，这样可以根据客户端分辨率的大小来进行合理的显示。

(6)、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。

优点

a、面对不同分辨率设备灵活性强

b、能够快捷解决多设备显示适应问题

缺点

a、兼容各种设备工作量大，效率低下

b、代码累赘，会出现隐藏无用的元素，加载时间加长

c、其实这是一种折中性质的设计解决方案，多方面因素影响而达不到最佳效果

d、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

杯布局已经非常完美，圣杯布局使用了相对定位，以后布局是有局限性的，而且宽度控制要改的地方也多。在淘宝UED（User Experience Design）探讨下，增加多一个div就可以不用相对布局了，只用到了浮动和负边距，这就是我们所说的双飞翼布局。 (2)、多栏布局 a、栏栅格系统：就是利用浮动实现的多栏布局，在bootstrap中用的非常多。 b、多列布局：栅格系统并没有真正实现分栏效果（如word中的分栏），CSS3为了满足这个要求增加了多列布局模块 (3)、弹性布局（Flexbox） CSS3引入了一种新的布局模式——Flexbox布局，即伸缩布局盒模型（Flexible Box），用来提供一个更加有效的方式制定、调整和分布一个容器里项目布局，即使它们的大小是未知或者动态的，这里简称为Flex。 Flexbox布局常用于设计比较复杂的页面，可以轻松的实现屏幕和浏览器窗口大小发生变化时保持元素的相对位置和大小不变，同时减少了依赖于浮动布局实现元素位置的定义以及重置元素的大小。 Flexbox布局在定义伸缩项目大小时伸缩容器会预留一些可用空间，让你可以调节伸缩项目的相对大小和位置。例如，你可以确保伸缩容器中的多余空间平均分配多个伸缩项目，当然，如果你的伸缩容器没有足够大的空间放置伸缩项目时，浏览器会根据一定的比例减少伸缩项目的大小，使其不溢出伸缩容器。 综合而言，Flexbox布局功能主要具有以下几点： a、屏幕和浏览器窗口大小发生改变也可以灵活调整布局； b、可以指定伸缩项目沿着主轴或侧轴按比例分配额外空间（伸缩容器额外空间），从而调整伸缩项目的大小； c、可以指定伸缩项目沿着主轴或侧轴将伸缩容器额外空间，分配到伸缩项目之前、之后或之间； d、可以指定如何将垂直于元素布局轴的额外空间分布到该元素的周围； e、可以控制元素在页面上的布局方向； f、可以按照不同于文档对象模型（DOM）所指定排序方式对屏幕上的元素重新排序。也就是说可以在浏览器渲染中不按照文档流先后顺序重排伸缩项目顺序。 (4)、瀑布流布局 瀑布流布局是流式布局的一种。是当下比较流行的一种网站页面布局，视觉表现为参差不齐的多栏布局，随着页面滚动条向下滚动，这种布局还会不断加载数据块并附加至当前尾部。最早采用此布局的网站是Pinterest，逐渐在国内流行开来。 优点 a、有效的降低了界面复杂度，节省了空间：我们不再需要臃肿复杂的页码导航链接或按钮了。 b、对触屏设备来说，交互方式更符合直觉：在移动应用的交互环境当中，通过向上滑动进行滚屏的操作已经成为最基本的用户习惯，而且所需要的操作精准程度远远低于点击链接或按钮。 c、更高的参与度：以上两点所带来的交互便捷性可以使用户将注意力更多的集中在内容而不是操作上，从而让他们更乐于沉浸在探索与浏览当中。 缺点 a、有限的用例： 无限滚动的方式只适用于某些特定类型产品当中一部分特定类型的内容。 例如，在电商网站当中，用户时常需要在商品列表与详情页面之间切换，这种情况下，传统的、带有页码导航的方式可以帮助用户更稳妥和准确的回到某个特定的列表页面当中。 b、额外的复杂度： 那些用来打造无限滚动的JS库虽然都自称很容易使用，但你总会需要在自己的产品中进行不同程度的定制化处理，以满足你们自己的需求;另外这些JS库在浏览器和设备兼容性等方面的表现也参差不齐，你必须做好充分的测试与调整工作。 c、再见了，页脚： 如果使用了比较典型的无限滚动加载模式，这就意味着你可以和页脚说拜拜了。 最好考虑一下页脚对于你的网站，特别是用户的重要性;如果其中确实有比较重要的内容或链接，那么最好换一种更传统和稳妥的方式。 千万不要耍弄你的用户，当他们一次次的浏览到页面底部，看到页脚，却因为自动加载的内容突然出现而无论如何都无法点击页脚中的链接时，他们会变的越发愤怒。 d、集中在一页当中动态加载数据，与一页一页的输出相比，究竟那种方式更利于SEO，这是你必须考虑的问题。对于某些以类型网站来说，在这方面进行冒险是很不划算的。 e、关于页面数量的印象： 其实站在用户的角度来看，这一点并非负面;不过，如果对于你的网站来说，通过更多的内容页面展示更多的相关信息(包括广告)是很重要的策略，那么单页无限滚动的方式对你并不适用。 (5)、流式布局（Fluid） 固定布局和流式布局在网页设计中最常用的两种布局方式。固定布局能呈现网页的原始设计效果，流式布局则不受窗口宽度影响，流式布局使用百分比宽度来限定布局元素，这样可以根据客户端分辨率的大小来进行合理的显示。 (6)、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。 优点 a、面对不同分辨率设备灵活性强 b、能够快捷解决多设备显示适应问题 缺点 a、兼容各种设备工作量大，效率低下 b、代码累赘，会出现隐藏无用的元素，加载时间加长 c、其实这是一种折中性质的设计解决方案，多方面因素影响而达不到最佳效果 d、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

## overflow :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。