1.Doctype作用？标准(严格)模式与兼容(混杂)模式各有什么区别?

1>!DOCTYPE声明位于位于HTML文档中的第一行，处于html 标签之前。告知浏览器的解析器用什么文档标准解析这个文档。DOCTYPE不存在或格式不正确会导致文档以兼容(混杂)模式呈现。

2>标准(严格)模式的排版 和JS运作模式都是以该浏览器支持的最高标准运行。在兼容(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。

2.html5有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何区分 HTML 和HTML5？

1>HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加。

2>绘画 canvas, 用于媒介回放的 video 和 audio 元素

3>本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失 sessionStorage 的数据在浏览器关闭后自动删除

4>语意化更好的内容元素，比如 article、footer、header、nav、section

* 表单控件，calendar、date、time、email、url、search
* 新的技术webworker, websockt, Geolocation
* 纯表现的元素：basefont，big，center，font, s，strike，tt，u；
* 对可用性产生负面影响的元素：frame，frameset，noframes；  
  支持HTML5新标签：
* IE8/IE7/IE6支持通过document.createElement方法产生的标签，
* 可以利用这一特性让这些浏览器支持HTML5新标签，
* 浏览器支持新标签后，还需要添加标签默认的样式：

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

* cookie在浏览器和服务器间来回传递。 sessionStorage和localStorage不会
* sessionStorage和localStorage的存储空间更大；
* sessionStorage和localStorage有更多丰富易用的接口；
* sessionStorage和localStorage各自独立的存储空间；

9.如何实现浏览器内多个标签页之间的通信?

* 调用localstorge、cookies等本地存储方式

****1、Web语义化 和 SEO****  
html 常规标签有html,head,body,div,span,table,ul,ol,dl,p,b,h1~h6,strong,form,input,img,em,i 等等，另外html5 还新增了很多语义化的标签，比如header,acticle,aside,section,footer,audio,radio 等等。  
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。  
SEO是指在了解搜索引擎自然排名机制的基础之上，对网站进行内部及外部的调整优化，改进网站在搜索引擎中关键词的自然排名，获得更多的展现量，吸引更多目标客户点击访问网站，从而达到互联网营销及品牌建设的目标。  
搜索引擎通过爬虫技术获取的页面就是由一堆html标签组成的代码,，人可以通过可视化的方式来判断页面上哪些内容是重点，而机器做不到。 但搜索引擎会根据标签的含义来判断内容的权重，因此，在合适的位置使用恰当的标签，使整个页面的语义明确，结构清晰，搜索引擎才能正确识别页面中的重要内容，并予以较高的权值。比如h1~h6这几个标签在SEO中的权值非常高，用它们作页面的标题就是一个简单的SEO优化。

****2、页面渲染机制****  
页面渲染就是浏览器的渲染引擎将html代码根据CSS定义的规则显示在浏览器窗口中的过程。大致工作原理如下：

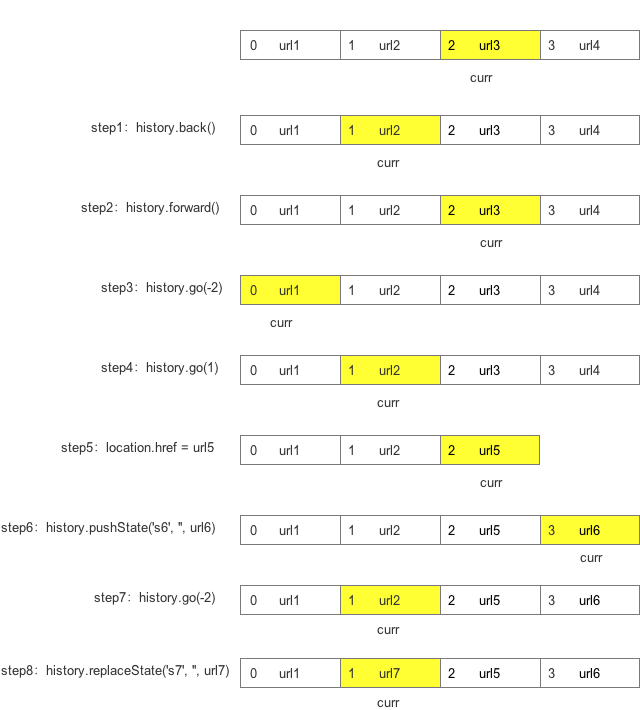
1. 用户输入网址，浏览器向服务器发出请求，服务器返回html文件；
2. 渲染引擎开始载入html代码，并将HTML中的标签转化为DOM节点，生成DOM树；
3. 如果<head>中引用了外部css文件，则发出css文件请求，服务器返回该文件；
4. 如果<head>中引用了外部js文件，则发出js文件请求，服务器返回该文件后开始运行；
5. 渲染引擎继续载入html中的<body>部分的代码，并开始解析前面返回的css文件，然后根据css选择器计算出节点的样式，创建渲染树；
6. 从根节点递归调用，计算每一个元素的大小、位置等，给每个节点所应该出现在屏幕上的精确坐标；
7. 如果body中的<img>引用了图片资源，则立即向服务器发出请求，此时渲染引擎不会等待图片下载完毕，而是继续渲染后面的代码；
8. 服务器返回图片文件，由于图片占用了一定面积，影响了后面段落的排版，因此引擎需要回过头来重新渲染这部分代码；
9. 如果此时js脚本中运行了style.display="none", 布局被改变，引擎也需要重新渲染这部分代码；
10. 直到</html>为止，页面渲染完毕。

****3、重绘和回流****  
当渲染树中的一部分(或全部)因为元素的规模尺寸，布局，隐藏等改变而需要重新构建。这就称为回流。比如上面的img文件加载完成后就会引起回流，每个页面至少需要一次回流，就是在页面第一次加载的时候。  
当渲染树中的一些元素需要更新属性，而这些属性只是影响元素的外观，风格，而不会影响布局的，比如background-color。则就叫称为重绘。  
从上面可以看出，回流必将引起重绘，而重绘不一定会引起回流。  
会引起重绘和回流的操作  
添加、删除元素(回流+重绘)  
隐藏元素，display:none(回流+重绘)，visibility:hidden(只重绘，不回流)  
移动元素，比如改变top,left,transform的值，或者移动元素到另外一个父元素中。(重绘+回流)  
对style的操作(对不同的属性操作，影响不一样)  
还有一种是用户的操作，比如改变浏览器大小，改变浏览器的字体大小等(回流+重绘)

****4、本地存储****  
本地存储最原始的方式就是 cookie,cookie 是存放在本地浏览器的一段文本，数据以键值对的形式保存，可以设置过期时间。 但是 cookie 不适合大量数据的存储，因为每请求一次页面，cookie 都会发送给服务器，这使得 cookie 速度很慢而且效率也不高。因此cookie的大小被限制为4k左右(不同浏览器可能不同,分HOST)，如下所示：  
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。  
Internet Explorer允许cookie多达4095个字节，包括：名（name）、值（value）和等号。  
在所有浏览器中，任何cookie大小超过限制都被忽略，且永远不会被设置。  
html5提供了两种在客户端存储数据的新方法：localStorage 和 sessionStorage, 它们都是以key/value 的形式来存储数据，前者是永久存储，后者的存储期限仅限于浏览器会话(session),即当浏览器窗口关闭后，sessionStorage中的数据被清除。  
localStorage的存储空间大约5M左右(不同浏览器可能不同，分 HOST)，这个相当于一个5M大小的前端页面的数据库，相比于cookie可以节约带宽，但localStorage在浏览器隐私模式下是不可读取的，当存储数据超过了localStorage的存储空间后会抛出异常。  
此外，H5还提供了逆天的websql和indexedDB,允许前端以关系型数据库的方式来存储本地数据，相对来说，这个功能目前应用的场景比较少，此处不作介绍。

****5、浏览器缓存机制****  
浏览器缓存机制是指通过 HTTP 协议头里的 Cache-Control（或 Expires）和 Last-Modified（或 Etag）等字段来控制文件缓存的机制。  
Cache-Control 用于控制文件在本地缓存有效时长。最常见的，比如服务器回包：Cache-Control:max-age=600 表示文件在本地应该缓存，且有效时长是600秒（从发出请求算起）。在接下来600秒内，如果有请求这个资源，浏览器不会发出 HTTP 请求，而是直接使用本地缓存的文件。  
Last-Modified 是标识文件在服务器上的最新更新时间。下次请求时，如果文件缓存过期，浏览器通过 If-Modified-Since 字段带上这个时间，发送给服务器，由服务器比较时间戳来判断文件是否有修改。如果没有修改，服务器返回304告诉浏览器继续使用缓存；如果有修改，则返回200，同时返回最新的文件。  
Cache-Control 通常与 Last-Modified 一起使用。一个用于控制缓存有效时间，一个在缓存失效后，向服务查询是否有更新。  
Cache-Control 还有一个同功能的字段：Expires。Expires 的值一个绝对的时间点，如：Expires: Thu, 10 Nov 2015 08:45:11 GMT，表示在这个时间点之前，缓存都是有效的。  
Expires 是 HTTP1.0 标准中的字段，Cache-Control 是 HTTP1.1 标准中新加的字段，功能一样，都是控制缓存的有效时间。当这两个字段同时出现时，Cache-Control 是高优化级的。  
Etag 也是和 Last-Modified 一样，对文件进行标识的字段。不同的是，Etag 的取值是一个对文件进行标识的特征字串。在向服务器查询文件是否有更新时，浏览器通过 If-None-Match 字段把特征字串发送给服务器，由服务器和文件最新特征字串进行匹配，来判断文件是否有更新。没有更新回包304，有更新回包200。Etag 和 Last-Modified 可根据需求使用一个或两个同时使用。两个同时使用时，只要满足基中一个条件，就认为文件没有更新。  
另外有两种特殊的情况：  
手动刷新页面（F5)，浏览器会直接认为缓存已经过期（可能缓存还没有过期），在请求中加上字段：Cache-Control:max-age=0，发包向服务器查询是否有文件是否有更新。  
强制刷新页面（Ctrl+F5)，浏览器会直接忽略本地的缓存（有缓存也会认为本地没有缓存），在请求中加上字段：Cache-Control:no-cache（或 Pragma:no-cache），发包向服务重新拉取文件。

****6、HistoryAPI和**Vue router的本质**

********

展示了初始长度为4的history栈，在不同操作方法下的变化情况，下面对每步进行分析：

step1～step4：均为history的访问方法，获取相应的历史记录中的url；

step5：通过location.href创建一个新的url记录，其将当前url2之后的记录清空，并在其后新增url5；history长度由4变成3；history的每次新增操作，都会将其后记录清空，在其后新增记录。

step6：通过pushState方法创建一个新的url记录，其也是清空、再新增记录；

step8：通过replaceState方法修改一个url记录，其不会产生新记录，而是将当前记录进行修改。

值得注意的是，通过pushState新增或者修改的history记录，被访问时，当前页面不刷新。而locaiton.href生成的记录被访问时，页面将进行刷新。

　　vue router其实质是在HTML5 historyAPI、window.onpopstate、window.onhashchange上做的封装，通过一定的规则映射到对应的方法上，同时通过监听变化，从而实现单页路由。

****7、HTML5离线缓存****  
HTML5离线缓存又叫Application Cache，是从浏览器的缓存中分出来的一块缓存区，如果要在这个缓存中保存数据，可以使用一个描述文件（manifest file），列出要下载和缓存的资源。  
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）  
离线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。

****8、Canvas 和 SVG****  
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。  
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。你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和SVG相比，canvas更依赖于分辨率，不支持事件处理器，文本渲染能力弱，比较适合密集型游戏，其中的许多对象会被频繁绘制，而svg则比较适用于类似谷歌地图带有大型渲染区域的应用程序。