### HTML5新特性：

1：拖拽释放（Drag  drop）

2:语义化更好的内容标签（header,nav,footer,aside,article,section）

语义化作用：页面也能呈现出很好地内容结构，有利于[SEO](http://baike.baidu.com/view/1047.htm" \t "https://www.cnblogs.com/freeyiyi1993/p/_blank)，便于团队开发和维护

3:音频，视频的API：（audio,video）

4:画布（Canvas） API

5:地理：（Geolocation）API

6:本地离线存储:localStorage长期存储数据，浏览器关闭后数据不丢失，

7：sessionStorage的数据在浏览器关闭后自动删除

8：表单控件：calendar，date, time, email ,url ,search

9:新的技术：webworker,websocket, Geolocation

#### canvas与svg的区别：

1canvas时h5提供的新的绘图方法

svg已经有了十多年的历史

2、canvas画图基于像素点，是位图，如果进行放大或缩小会失真

svg基于图形，用html标签描绘形状，放大缩小不会失真

3、canvas需要在js中绘制

svg在html正绘制

4、canvas支持颜色较svg多

5、canvas无法对已经绘制的图像进行修改操作

svg可以获取到标签进行操作

### 1 说一下meta标签有什么用？

元素可提供有关页面的元信息（meta-information），比如针对搜索引擎、更新频度、cookie的描述和关键词。元数据总是以名称/值对的形式表示，名称有两种类型：name和http-equiv。其中当名称为http-equiv，会将值关联到HTTP头部

### 2 HTML文档有哪几种文档类型？为什么要进行类型声明？

严格型（strict）、过渡型（transitional）和框架型（Frameset）  
严格型不包括框架集、一些废弃的元素标签，过渡型包含废弃的元素标签，但不允许框架集。而框架型等同于过渡型，且允许框架。  
作用：只有进行正确的类型声明，浏览器才能正确的解析html文档。

### 3 bootstrap栅格系统的原理是什么？

媒体查询 + 百分比宽度

@media (min-width: 768px) {/\*当宽度大于768px时触发\*/

.container {

width: 750px;

}

}

@media (min-width: 992px) {/\*当宽度大于992px时触发\*/

.container {

width: 970px;

}

}

@media (min-width: 1200px) {/\*当宽度大于1200px时触发\*/

.container {

width: 1170px;

}

}

### 4 回流和重绘

**重绘：**

当元素样式发生改变，但不影响布局时，浏览器将使用重绘进行元素更新，由于此时只需要UI层面的绘制，因此损耗较小

**回流：**

当元素尺寸、结构或者触发某些属性的时候，浏览器会重新渲染页面，这就叫回流。此时，浏览器需要重新计算，重新进行页面布局，所以损耗较大

一般有以下几种操作：

页面初次渲染

浏览器窗口大小改变

元素尺寸、位置、内容改变

元素字体大小改变

添加或删除可见的dom元素

触发CSS伪类，如:hover

查询某些属性或者调用某些方法

clientWidth, clientHeight, clientTop, clientLeft

offsetWidth, offsetHeight, offsetTop, offsetLeft

scrollWidth, scrollHeight, scrollTop, scrollLeft

getComputedStyle()

getBoundingClientRect()

scrollTo()

**回流必定触发重绘，重绘不一定触发回流，重绘代价小，回流代价大**

### 5 如何避免回流和重绘

CSS:

* 避免使用table布局
* 尽可能再dom树的末端修改class
* 避免使用多层内联样式
* 将动画效果应用到position: absolute || fixed上
* 避免使用css表达式（例如calc）
* CSS3硬件加速（GPU加速）

JavaScript:

* 避免频繁操作样式，最好一次性修改style属性，或者将样式列表定义成class，并一次性更改class属性
* 避免频繁操作dom，创建一个documentFragment，在他上面应用所有的dom操作，最后再把他添加到文档中
* 也可以先为元素设置display: none，操作结束后再把它显示出来，因为再display为none的元素上进行dom操作不会引发重绘和回流
* 避免频繁读取会引发重绘回流的属性，如果需要多次使用，就用一个变量缓存起来
* 对具有复杂动画的元素使用绝对定位，使他脱离文档流，否则会引起父元素及后续元素频繁回流

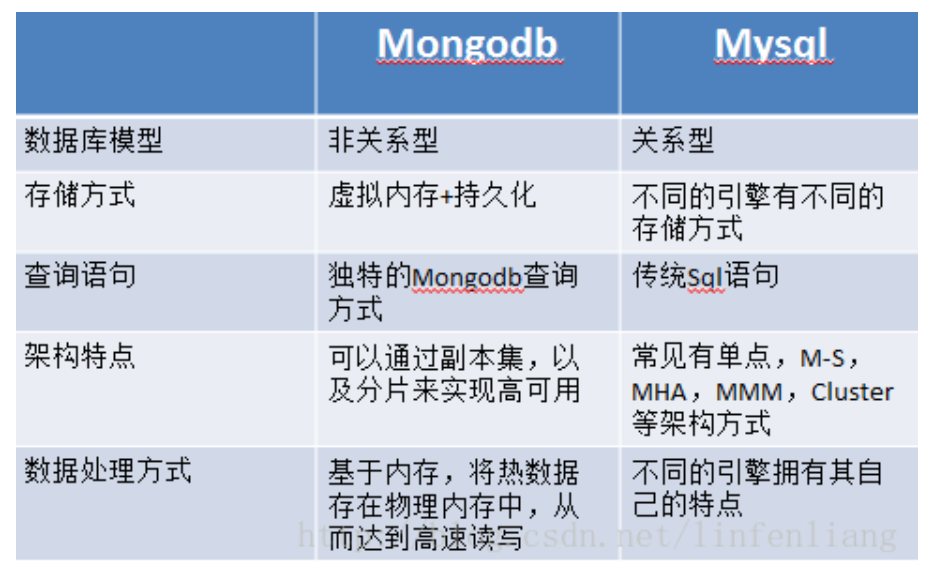
### 6 为什么使用MongoDB

由于MongoDB独特的数据处理方式，可以将热点数据加载到内存，故而对查询来讲，会非常快（当然也会非常消耗内存）；同时由于采用了BSON的方式存储数据，故而对JSON格式数据具有非常好的支持性以及友好的表结构修改性，文档式的存储方式，数据友好可见

缺点：支持的特性不够丰富

现有产品的不够成熟

### 7 关系型数据库和非关系型数据库区别



1.成本：Nosql数据库简单易部署，基本都是开源软件，不需要像使用Oracle那样花费大量成本购买使用，相比关系型数据库价格便宜。  
2.查询速度：Nosql数据库将数据存储于缓存之中，而且不需要经过SQL层的解析，关系型数据库将数据存储在硬盘中，自然查询速度远不及Nosql数据库。  
3.存储数据的格式：Nosql的存储格式是key,value形式、文档形式、图片形式等等，所以可以存储基础类型以及对象或者是集合等各种格式，而数据库则只支持基础类型。  
4.扩展性：关系型数据库有类似join这样的多表查询机制的限制导致扩展很艰难。Nosql基于键值对，数据之间没有耦合性，所以非常容易水平扩展。  
5.持久存储：Nosql不使用于持久存储，海量数据的持久存储，还是需要关系型数据库  
6.数据一致性：非关系型数据库一般强调的是数据最终一致性，不像关系型数据库一样强调数据的强一致性，从非关系型数据库中读到的有可能还是处于一个中间态的数据，  
Nosql不提供对事务的处理。

### 8 强缓存和协商缓存

浏览器缓存包含两种类型，即强缓存（也叫本地缓存）和协商缓存，浏览器在第一次请求发生后，再次请求时：

浏览器在请求某一资源时，会先获取该资源缓存的header信息，判断是否命中强缓存（cache-control和expires信息），若命中直接从缓存中获取资源信息，包括缓存header信息；本次请求根本就不会与服务器进行通信

如果没有命中强缓存，浏览器会发送请求到服务器，请求会携带第一次请求返回的有关缓存的header字段信息（Last-Modified/If-Modified-Since和Etag/If-None-Match），由服务器根据请求中的相关header信息来比对结果是否协商缓存命中；若命中，则服务器返回新的响应header信息更新缓存中的对应header信息，但是并不返回资源内容，它会告知浏览器可以直接从缓存获取；否则返回最新的资源内容

### 9 强缓存相关的Header字段

1. **expires**，这是http1.0时的规范；它的值为一个绝对时间的GMT格式的时间字符串，如Mon, 10 Jun 2015 21:31:12 GMT，如果发送请求的时间在expires之前，那么本地缓存始终有效，否则就会发送请求到服务器来获取资源
2. **cache-control：max-age=number**，这是http1.1时出现的header信息，主要是利用该字段的max-age值来进行判断，它是一个相对值；资源第一次的请求时间和Cache-Control设定的有效期，计算出一个资源过期时间，再拿这个过期时间跟当前的请求时间比较，如果请求时间在过期时间之前，就能命中缓存，否则就不行；cache-control除了该字段外，还有下面几个比较常用的设置值：
   1. no-cache：不使用本地缓存。需要使用缓存协商，先与服务器确认返回的响应是否被更改，如果之前的响应中存在ETag，那么请求的时候会与服务端验证，如果资源未被更改，则可以避免重新下载。
   2. no-store：直接禁止游览器缓存数据，每次用户请求该资源，都会向服务器发送一个请求，每次都会下载完整的资源。
   3. public：可以被所有的用户缓存，包括终端用户和CDN等中间代理服务器。
   4. private：只能被终端用户的浏览器缓存，不允许CDN等中继缓存服务器对其缓存。

　　注意：如果cache-control与expires同时存在的话，cache-control的优先级高于expires

### 10 协商缓存

如果缓存过期了，我们就可以使用协商缓存来解决问题。协商缓存需要请求，如果缓存有效会返回 304。

协商缓存需要客户端和服务端共同实现，和强缓存一样，也有两种实现方式。

#### Last-Modified 和 If-Modified-Since

Last-Modified 表示本地文件最后修改日期，If-Modified-Since 会将 Last-Modified 的值发送给服务器，询问服务器在该日期后资源是否有更新，有更新的话就会将新的资源发送回来。

但是如果在本地打开缓存文件，就会造成 Last-Modified 被修改，所以在 HTTP / 1.1 出现了 ETag 。

#### ETag 和 If-None-Match

ETag 类似于文件指纹，If-None-Match 会将当前 ETag 发送给服务器，询问该资源 ETag 是否变动，有变动的话就将新的资源发送回来。并且 ETag 优先级比 Last-Modified 高。

### 11 前端性能优化

减少http请求

1 选择合适的缓存策略

2 使用HTTP2.0：因为浏览器会有并发请求限制，在 HTTP / 1.1 时代，每个请求都需要建立和断开，消耗了好几个 RTT 时间，并且由于 TCP 慢启动的原因，加载体积大的文件会需要更多的时间。

在 HTTP / 2.0 中引入了多路复用，能够让多个请求使用同一个 TCP 链接，极大的加快了网页的加载速度。并且还支持 Header 压缩，进一步的减少了请求的数据大小。

3 懒加载

4 图片优化：**计算图片大小，**减少像素点，减少每个像素点能够显示的颜色，图片多的话使用雪碧图

5 静态资源尽量使用 CDN 加载

6 CSS放head中，JS放body后

7 减少DOM操作，多个操作尽量合并在一起执行

8 路由懒加载

9 开启 gzip 压缩

10 使用Webpack优化

**html性能优化：**

1:语意化标签 (利于搜索引擎收录 也提高了代码阅读性)

2:减少标签的嵌套 (利于渲染引擎的加速渲染)

3:把script标签移到HTML文件末尾，因为JS会阻塞后面的页面的显示

4:减少iframe的使用，因为iframe会增加一条http请求，阻止页面加载，即使内容为空，加载也需要时间

5:减少注释，因为过多注释不光占用空间，如果里面有大量关键词会影响搜索引擎的搜索。上线的时候最好清空注释

6:html结构的复用性

**css性能优化：**

1:尽量不使用通配符(＊)

2:把页面公用的样式给封装到css文件里面(减少代码量、复用性强、便于维护) 3:不要用内联样式和行类样式(比较乱 维护性比较差)

4:尽量使用最具体的类别、避免后代选择器、属于标签类别的规则永远不要包含子选择器 //因为从左到右解析关系，在CSS选择器中后代选择器非但没有帮我们加快CSS查找。反而查找很慢

5:能用css实现的样式就不要用图片。减少http请求

6:合并压缩css文件 减少http请求

**js性能优化:**

1:可以多使用事件委托 减少事件注册

2:多使用if语句多做判断 减少不必要的操作

3:创建标签的时候尽量通过字符串创建，不要通过createElement。因为通过 createElement创建完之后还要再次操作这个dom去设置属性和文本等。而直接创建字符串只需要一步设置好。

4:设置样式的时候 尽量不要用style去设置样式 维护性 可读性差 最好添加通过class去设置样式

5:尽量少使用eval函数。(使用eval相当于在运行时再次调用解释引擎对内容进行运行，需要消耗大量时间，而且使用Eval带来的安全性问题也是不容忽视的。)

6:使用三目运算符替代条件分支可以减少解释器对条件的探测次数

7:提高代码复用性使用封装。减少代码量、维护性好。

8:合并压缩js 减少http请求

### 12 Express 优点

线性逻辑，通过中间件形式把业务逻辑细分、简化，一个请求进来经过一系列中间件处理后再响应给用户，清晰明了。 缺点：基于 callback 组合业务逻辑，业务逻辑复杂时嵌套过多，异常捕获困难。

### 13 PWA

PWA 的全称是 Progressive Web App，是 Google 在2015年提出，今年才推广开来的想法。具体的意思就是在移动端利用提供的标准化框架，在网页应用中实现和原生应用相近的用户体验。总而言之，它说到底还是一个网页应用，只不过通过最新的技术手段实现了某些以前原生应用才有的功能。

### 14 什么是模块化开发

我们要写一个实现A功能的JS代码，这个功能在项目其他位置也需要用到，那么我们就可以把这个功能看成一个模块采用一定的方式进行模块化编写，既能实现复用还可以分而治之，同理在写样式的时候，如果我们需要某种特殊的样式，会在很多地方应用，那么我们也可以采用一定的方式进行CSS的模块化，具体说来，JS模块化方案很多有AMD/CommonJS/UMD/ES6 Module等，CSS模块化开发大多是在less、sass、stylus等预处理器的import/mixin特性支持下实现的，总结来说模块化就是进行功能的封装

### 15 组件化

组件化将页面视为一个容器，页面上各个独立部分例如：头部、导航、焦点图、侧边栏、底部等视为独立组件，不同的页面根据内容的需要，去盛放相关组件即可组成完整的页面。

模块化和组件化一个最直接的好处就是复用，同时我们也应该有一个理念，模块化和组件化除了复用之外还有就是分治，我们能够在不影响其他代码的情况下按需修改某一独立的模块或是组件

### 16 如何计算首屏时间

MutationObserver接口提供了监视对DOM树所做更改的能力，如果通过 MutationObserver 监听页面body，当页面body元素变化最剧烈并达到最大时就是首屏初次渲染完成的时间

如果是img的话，我们可以使用img标签下的src属性获取属性值即可，如果是div的化可以使用 window.getComputedStyle(dom) 方式获取它的属性值，然后通过正则获取图片的链接即可

然后通过 performance.getEntriesByName(element)[0].responseEnd 的方式可以获取到图片的下载时间，与我们计算的dom响应时间相比取最大值。

### 17 如何设计组件

* 从业务上看，当业务达到一定规模后，很多地方需要复用
* 从设计上看，产品要遵循一定的设计规范来保持统一性
* 从开发上看，对开发效率要求高，需要快速迭代和响应开发需求
* 从维护上看，需要统一代码管理，需要达到更改一处全局响应的高可维护性