## HTML

* 你有用过哪些前端性能优化的方法？  
  （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，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。（8） 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。
* Doctype作用？标准模式与兼容模式各有什么区别?  
  （1）、<!DOCTYPE>声明位于HTML文档中的第一行，处于 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模式呈现。（2）、标准模式的排版 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。
* 行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？  
  首先：CSS规范规定，每个元素都有display属性，确定该元素的类型，每个元素都有默认的display值，如div的display默认值为“block”，则为“块级”元素；span默认display属性值为“inline”，是“行内”元素。
  （1）行内元素有：a b span img input select strong（强调的语气）
  （2）块级元素有：div ul ol li dl dt dd h1 h2 h3 h4…p
  （3）常见的空元素：
  <br> <hr> <img> <input> <link> <meta>
  鲜为人知的是：
  <area> <base> <col> <command> <embed> <keygen> <param> <source> <track> <wbr>
  不同浏览器（版本）、HTML4（5）、CSS2等实际略有差异
  参考: http://stackoverflow.com/questions/6867254/browsers-default-css-for-html-elements
* 页面导入样式时，使用link和@import有什么区别？  
  （1）link属于XHTML标签，除了加载CSS外，还能用于定义RSS, 定义rel连接属性等作用；而@import是CSS提供的，只能用于加载CSS;
  （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是CSS2.1 提出的，只在IE5以上才能被识别，而link是XHTML标签，无兼容问题;
* 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。
* 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？\* 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。
  绘画 canvas;
  用于媒介回放的 video 和 audio 元素;
  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;
  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, websocket, Geolocation;
  移除的元素：
  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。 当然也可以直接使用成熟的框架、比如html5shim;
  <!--[if lt IE 9]>
  <script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>
  <![endif]-->
* 简述一下你对HTML语义化的理解？  
  用正确的标签做正确的事情。
  html语义化让页面的内容结构化，结构更清晰，便于对浏览器、搜索引擎解析;
  即使在没有样式CSS情况下也以一种文档格式显示，并且是容易阅读的;
  搜索引擎的爬虫也依赖于HTML标记来确定上下文和各个关键字的权重，利于SEO;
  使阅读源代码的人对网站更容易将网站分块，便于阅读维护理解。
* HTML5的离线储存怎么使用，工作原理能不能解释一下（不常用）  
  在用户没有与因特网连接时，可以正常访问站点或应用，在用户与因特网连接时，更新用户机器上的缓存文件。原理：HTML5的离线存储是基于一个新建的.appcache文件的缓存机制(不是存储技术)，通过这个文件上的解析清单离线存储资源，这些资源就会像cookie一样被存储了下来。之后当网络在处于离线状态下时，浏览器会通过被离线存储的数据进行页面展示。如何使用：1、页面头部像下面一样加入一个manifest的属性；2、在cache.manifest文件的编写离线存储的资源；
  CACHE MANIFEST
  #v0.11
  CACHE:
  js/app.js
  css/style.css
  NETWORK:
  resourse/logo.png
  FALLBACK:
  / /offline.html
  3、在离线状态时，操作window.applicationCache进行需求实现。
* 浏览器是怎么对HTML5的离线储存资源进行管理和加载的呢？  
  在线的情况下，浏览器发现html头部有manifest属性，它会请求manifest文件，如果是第一次访问app，那么浏览器就会根据manifest文件的内容下载相应的资源并且进行离线存储。如果已经访问过app并且资源已经离线存储了，那么浏览器就会使用离线的资源加载页面，然后浏览器会对比新的manifest文件与旧的manifest文件，如果文件没有发生改变，就不做任何操作，如果文件改变了，那么就会重新下载文件中的资源并进行离线存储。
  离线的情况下，浏览器就直接使用离线存储的资源。
* 请描述一下 cookies，sessionStorage 和 localStorage 的区别  
  cookie是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。
  cookie数据始终在同源的http请求中携带（即使不需要），记会在浏览器和服务器间来回传递。
  sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。
  存储大小：
  cookie数据大小不能超过4k。
  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。
  有期时间：
  localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动删除数据；
  sessionStorage 数据在当前浏览器窗口关闭后自动删除。
  cookie 设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭
* 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>
* HTML5的form如何关闭自动完成功能？  
  给不想要提示的 form 或某个 input 设置为 autocomplete=off。
* 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localstorge、cookies等本地存储方式；
  localstorge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？  
  1、map+area或者svg
  2、border-radius
  3、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
* 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="height:1px;overflow:hidden;background:red"></div>
* title与h1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。

## CSS

* 介绍一下标准的CSS的盒子模型？低版本IE的盒子模型有什么不同的？  
  （1）有两种， IE 盒子模型、W3C 盒子模型；
  （2）盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border)；
  （3）区 别： IE的content部分把 border 和 padding计算了进去;
* CSS选择符有哪些？哪些属性可以继承？  
   1.id选择器（ # myid）
  2.类选择器（.myclassname）
  3.标签选择器（div, h1, p）
  4.相邻选择器（h1 + p）
  5.子选择器（ul > li）
  6.后代选择器（li a）
  7.通配符选择器（ \* ）
  8.属性选择器（a[rel = "external"]）
  9.伪类选择器（a:hover, li:nth-child）
  \* 可继承的样式： font-size font-family color, UL LI DL DD DT;
  \* 不可继承的样式：border padding margin width height ;
* CSS优先级算法如何计算？\*  
  优先级就近原则，同权重情况下样式定义最近者为准;
  \* 载入样式以最后载入的定位为准;
  优先级为:
  同权重: 内联样式表（标签内部）> 嵌入样式表（当前文件中）> 外部样式表（外部文件中）。
  !important > id > class > tag
  important 比 内联优先级高
* CSS3新增伪类有那些？   
  举例：
  p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。
  p:last-of-type 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。
  p:only-of-type 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。
  p:only-child 选择属于其父元素的唯一子元素的每个 <p> 元素。
  p:nth-child(2) 选择属于其父元素的第二个子元素的每个 <p> 元素。
  ::after 在元素之前添加内容,也可以用来做清除浮动。
  ::before 在元素之后添加内容
  :enabled
  :disabled 控制表单控件的禁用状态。
  :checked 单选框或复选框被选中。
* 如何居中div(常用)  
  1.水平居中：给div设置一个宽度，然后添加margin:0 auto属性div{
  width:200px;
  margin:0 auto;
  }
    
    
  2.让绝对定位的div居中: div {
  position: absolute;
  width: 300px;
  height: 300px;
  margin: auto;
  top: 0;
  left: 0;
  bottom: 0;
  right: 0;
  background-color: pink;
  }
    
  水平垂直居中一确定容器的宽高 宽500 高 300 的层
  设置层的外边距
  div {
  position: relative; /\* 相对定位或绝对定位均可 \*/
  width:500px;
  height:300px;
  top: 50%;
  left: 50%;
  margin: -150px 0 0 -250px; /\* 外边距为自身宽高的一半 \*/
  background-color: pink; /\* 方便看效果 \*/
  }
    
    
  3.水平垂直居中二未知容器的宽高:利用 `transform` 属性
    
   div {
  position: absolute; /\* 相对定位或绝对定位均可 \*/
  width:500px;
  height:300px;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background-color: pink; /\* 方便看效果 \*/
  }
    
    
  4.水平垂直居中三利用 flex 布局
  : 实际使用时应考虑兼容性
    
  .container {
  display: flex;
  align-items: center; /\* 垂直居中 \*/
  justify-content: center; /\* 水平居中 \*/
  }
  .container div {
  width: 100px;
  height: 100px;
  background-color: pink; /\* 方便看效果 \*/
  }
* display有常用的几种值？说明他们的作用。   
   block 块类型。默认宽度为父元素宽度，可设置宽高，换行显示。
    
   none 缺省值。象行内元素类型一样显示。
    
   inline 行内元素类型。默认宽度为内容宽度，不可设置宽高，同行显示。
    
   inline-block 默认宽度为内容宽度，可以设置宽高，同行显示。
    
   list-item 象块类型元素一样显示，并添加样式列表标记。
    
   table 此元素会作为块级表格来显示。
    
   inherit 规定应该从父元素继承 display 属性的值。
* position的值relative和absolute定位原理是？   
   absolute
  生成绝对定位的元素，相对于值不为   
   static的第一个父元素进行定位。
    
   fixed （老IE不支持）
  生成绝对定位的元素，相对于浏览器窗口进行定位。
    
  relative
  生成相对定位的元素，相对于其正常位置进行定位。
    
  static
  默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right z-index 声明）。
    
   inherit
  规定从父元素继承 position 属性的值。
* CSS3有哪些新特性？   
   新增各种CSS选择器 （: not(.input)：所有 class 不是“input”的节点）
    
   圆角 （border-radius:8px）
    
   多列布局 （multi-column layout）
    
   阴影和反射 （Shadow\Reflect）
    
   文字特效 （text-shadow、）
    
   文字渲染 （Text-decoration）
    
   线性渐变 （gradient）
    
   旋转 （transform）
    
   缩放,定位,倾斜,动画,多背景
  例如:transform:\scale(0.85,0.90)\ translate(0px,-30px)\ skew(-9deg,0deg)\Animation:
* 请解释一下CSS3的Flexbox（弹性盒布局模型）,以及适用场景？  
   一个用于页面布局的全新CSS3功能，Flexbox可以把列表放在同一个方向（从上到下排列，从左到右），并让列表能延伸到占用可用的空间。
  较为复杂的布局还可以通过嵌套一个伸缩容器（flex container）来实现。
  采用Flex布局的元素，称为Flex容器（flex container），简称"容器"。
  它的所有子元素自动成为容器成员，称为Flex项目（flex item），简称"项目"。
  常规布局是基于块和内联流方向，而Flex布局是基于flex-flow流可以很方便的用来做局中，能对不同屏幕大小自适应。
  在布局上有了比以前更加灵活的空间。
* 用纯CSS创建一个三角形的原理是什么？  
  把上、左、右三条边隐藏掉（颜色设为 transparent）
    
  #demo {
  width: 0;
  height: 0;
  border-width: 20px;
  border-style: solid;
  border-color: transparent transparent red transparent;
  }
* css多列等高如何实现？  
   利用padding-bottom|margin-bottom正负值相抵；
    
  设置父容器设置超出隐藏（overflow:hidden），这样子父容器的高度就还是它里面的列没有设定padding-bottom时的高度，
    
  当它里面的任 一列高度增加了，则父容器的高度被撑到里面最高那列的高度，
  其他比这列矮的列会用它们的padding-bottom补偿这部分高度差。
* 经常遇到的浏览器的兼容性有哪些？原因，解决方法是什么，常用hack的技巧 ？\*  
   1. png24位的图片在iE6浏览器上出现背景，解决方案是做成PNG8.
    
   2.浏览器默认的margin和padding不同。解决方案是加一个全局的\*{margin:0;padding:0;}来统一。
    
   3. IE6双边距bug: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。
    
   4.浮动ie产生的双倍距离 #box{ float:left; width:10px; margin:0 0 0 100px;}
  这种情况之下IE会产生20px的距离，解决方案是在float的标签样式控制中加入 ——\_display:inline;将其转化为行内属性。(\_这个符号只有ie6会识别)
  渐进识别的方式，从总体中逐渐排除局部。
* 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： \* {padding: 0; margin: 0;} （强烈不建议）
    
  淘宝的样式初始化代码：
  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/1.5tahoma, arial, b8bf53; }
  h1, h2, h3, h4, h5, h6{ font-size:100%; }
  address, cite, dfn, em, var { font-style:normal; }
  code, kbd, pre, samp { font-family:couriernew, courier, monospace; }
  small{ font-size:12px; }
  ul, ol { list-style:none; }
  a { text-decoration:none; }
  a:hover { text-decoration:underline; }
  sup { vertical-align:text-top; }
  sub{ vertical-align:text-bottom; }
  legend { color:#000; }
  fieldset, img { border:0; }
  button, input, select, textarea { font-size:100%; }
  table { border-collapse:collapse; border-spacing:0; }
* absolute的containing block(容器块)计算方式跟正常流有什么不同？  
   无论属于哪种，都要先找到其祖先元素中最近的 position 值不为 static 的元素，然后再判断：
    
   1、若此元素为 inline 元素，则 containing block 为能够包含这个元素生成的第一个和最后一个 inline box 的 padding box (除 margin, border 外的区域) 的最小矩形；
    
   2、否则,则由这个祖先元素的 padding box 构成。
  如果都找不到，则为 initial containing block。
    
   补充：
  1. static(默认的)/relative：简单说就是它的父元素的内容框（即去掉padding的部分）
    
   2. absolute: 向上找最近的定位为absolute/relative的元素
    
   3. fixed: 它的containing block一律为根元素(html/body)，根元素也是initial containing block
* 对BFC规范(块级格式化上下文：block formatting context)的理解(范围广，可以聊很多)  
  （W3C CSS 2.1 规范中的一个概念,它是一个独立容器，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。）
    
   一个页面是由很多个 Box 组成的,元素的类型和 display 属性,决定了这个 Box 的类型。
  不同类型的 Box,会参与不同的 Formatting Context（决定如何渲染文档的容器）,因此Box内的元素会以不同的方式渲染,也就是说BFC内部的元素和外部的元素不会互相影响。
* css定义的权重以下是权重的规则：标签的权重为1，class的权重为10，id的权重为100，以下例子是演示各种定义的权重值：
  /\*权重为1\*/
  div{
  }
  /\*权重为10\*/
  .class1{
  }
  /\*权重为100\*/
  #id1{
  }
  /\*权重为100+1=101\*/
  #id1 div{
  }
  /\*权重为10+1=11\*/
  .class1 div{
  }
  /\*权重为10+10+1=21\*/
  .class1 .class2 div{
  }
  如果权重相同，则最后定义的样式会起作用，但是应该避免这种情况出现
* 请解释一下为什么需要清除浮动？  
   清除浮动的方式清除浮动是为了清除使用浮动元素产生的影响。浮动的元素，高度会塌陷，而高度的塌陷使我们页面后面的布局不能正常显示。  
   1、父级div定义height；
  2、父级div 也一起浮动；
  3、常规的使用一个class；
  .clearfix::before, .clearfix::after {
  content: " ";
  display: table;
  }
  .clearfix::after {
  clear: both;
  }
  .clearfix {
  \*zoom: 1;
  }
  4、SASS编译的时候，浮动元素的父级div定义伪类:after
  &::after,&::before{
  content: " ";
  visibility: hidden;
  display: block;
  height: 0;
  clear: both;
  }
  解析原理：
  1) display:block 使生成的元素以块级元素显示,占满剩余空间;
  2) height:0 避免生成内容破坏原有布局的高度。
  3) visibility:hidden 使生成的内容不可见，并允许可能被生成内容盖住的内容可以进行点击和交互;
  4）通过 content:"."生成内容作为最后一个元素，至于content里面是点还是其他都是可以的，例如oocss里面就有经典的 content:".",有些版本可能content 里面内容为空,一丝冰凉是不推荐这样做的,firefox直到7.0 content:”" 仍然会产生额外的空隙；
  5）zoom：1 触发IE hasLayout。
  通过分析发现，除了clear：both用来闭合浮动的，其他代码无非都是为了隐藏掉content生成的内容，这也就是其他版本的闭合浮动为什么会有font-size：0，line-height：0。
* 移动端的布局用过媒体查询吗？   
  CSS就是为文档提供在不同媒介上展示的适配方法
  <!-- link元素中的CSS媒体查询 -->
  当媒体查询为真时，相关的样式表或样式规则会按照正常的级联规被应用。
  当媒体查询返回假， <link> 标签上带有媒体查询的样式表 仍将被下载 （只不过不会被应用）。
    
  <link rel="stylesheet" media="(max-width: 800px)" href="example.css" />
  <!-- 样式表中的CSS媒体查询 -->
  包含了一个媒体类型和至少一个使用 宽度、高度和颜色等媒体属性来限制样式表范围的表达式。
    
  CSS3加入的媒体查询使得无需修改内容便可以使样式应用于某些特定的设备范围。
    
  <style>
  @media (min-width: 700px) and (orientation: landscape){
  .sidebar {
  display: none;
  }
  }
  </style>
* 使用 CSS 预处理器吗？喜欢那个？SASS (SASS、LESS没有本质区别，只因为团队前端都是用的SASS)
* CSS优化、提高性能的方法有哪些？关键选择器（key selector）。选择器的最后面的部分为关键选择器（即用来匹配目标元素的部分）；
  如果规则拥有 ID 选择器作为其关键选择器，则不要为规则增加标签。过滤掉无关的规则（这样样式系统就不会浪费时间去匹配它们了）；
  提取项目的通用公有样式，增强可复用性，按模块编写组件；增强项目的协同开发性、可维护性和可扩展性;
  使用预处理工具或构建工具（gulp对css进行语法检查、自动补前缀、打包压缩、自动优雅降级）；
* 怎么让Chrome支持小于12px 的文字？   
   1、用图片：如果是内容固定不变情况下，使用将小于12px文字内容切出做图片，这样不影响兼容也不影响美观。
    
   2、使用12px及12px以上字体大小：为了兼容各大主流浏览器，建议设计美工图时候设置大于或等于12px的字体大小，如果是接单的这个时候就需要给客户讲解小于12px浏览器不兼容等事宜。
    
   3、继续使用小于12px字体大小样式设置：如果不考虑chrome可以不用考虑兼容，同时在设置小于12px对象设置-webkit-text-size-adjust:none，做到最大兼容考虑。
    
  4、使用12px以上字体：为了兼容、为了代码更简单 从新考虑权重下兼容性。
* ::before 和 :after中双冒号和单冒号 有什么区别？解释一下这2个伪元素的作用。  
   单冒号(:)用于CSS3伪类，  
   双冒号(::)用于CSS3伪元素。（伪元素由双冒号和伪元素名称组成）
  双冒号是在当前规范中引入的，用于区分伪类和伪元素。  
   不过浏览器需要同时支持旧的已经存在的伪元素写法，
  比如:first-line、:first-letter、:before、:after等，
  而新的在CSS3中引入的伪元素则不允许再支持旧的单冒号的写法。
  想让插入的内容出现在其它内容前，使用::before，否者，使用::after；
  在代码顺序上，::after生成的内容也比::before生成的内容靠后。
  如果按堆栈视角，::after生成的内容会在::before生成的内容之上
* 设置元素浮动后，该元素的display值是多少？ 自动变成了 display:block
* 什么是Cookie 隔离？（或者说：请求资源的时候不要让它带cookie怎么做）  
  如果静态文件都放在主域名下，那静态文件请求的时候都带有的cookie的数据提交给server的，非常浪费流量，
  所以不如隔离开。
  因为cookie有域的限制，因此不能跨域提交请求，故使用非主要域名的时候，请求头中就不会带有cookie数据，
  这样可以降低请求头的大小，降低请求时间，从而达到降低整体请求延时的目的。
  同时这种方式不会将cookie传入Web Server，也减少了Web Server对cookie的处理分析环节，
  提高了webserver的http请求的解析速度。

**开放问题**

* 什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的IE？
* 你对line-height是如何理解的？
* style标签写在body后与body前有什么区别？
* png、jpg、gif 这些图片格式解释一下，分别什么时候用。有没有了解过webp？

## JavaScript

* 介绍js的基本数据类型。   
  Undefined、Null、Boolean、Number、String、
    
  ECMAScript 2015 新增:Symbol(创建后独一无二且不可变的数据类型 )
* 介绍js有哪些内置对象？  
   Object 是 JavaScript 中所有对象的父对象
    
   数据封装类对象：Object、Array、Boolean、Number 和 String
  其他对象：Function、Arguments、Math、Date、RegExp、Error
* 说几条写JavaScript的基本规范？  
  1.不要在同一行声明多个变量。
    
  2.请使用 ===/!==来比较true/false或者数值
    
  3.使用对象字面量替代new Array这种形式
    
  4.不要使用全局函数。
    
  5.Switch语句必须带有default分支
    
  6.函数不应该有时候有返回值，有时候没有返回值。
    
  7.For循环必须使用大括号
    
  8.If语句必须使用大括号
    
  9.for-in循环中的变量 应该使用var关键字明确限定作用域，从而避免作用域污染。
* JavaScript原型，原型链 ? 有什么特点？（开放问题）  
  每个对象都会在其内部初始化一个属性，就是prototype(原型)，当我们访问一个对象的属性时，
  如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，
  于是就这样一直找下去，也就是我们平时所说的原型链的概念。
    
  关系：instance.constructor.prototype = instance.\_\_proto\_\_
    
  特点：
  JavaScript对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。当我们修改原型时，与之相关的对象也会继承这一改变。
    
  当我们需要一个属性的时，Javascript引擎会先看当前对象中是否有这个属性， 如果没有的话，
  就会查找他的Prototype对象是否有这个属性，如此递推下去，一直检索到 Object 内建对象。
* JavaScript有几种类型的值？，你能画一下他们的内存图吗？栈：原始数据类型（Undefined，Null，Boolean，Number、String）
  堆：引用数据类型（对象、数组和函数）
  两种类型的区别是：存储位置不同；
  原始数据类型直接存储在栈(stack)中的简单数据段，占据空间小、大小固定，属于被频繁使用数据，所以放入栈中存储；
  引用数据类型存储在堆(heap)中的对象,占据空间大、大小不固定。如果存储在栈中，将会影响程序运行的性能；引用数据类型在栈中存储了指针，该指针指向堆中该实体的起始地址。当解释器寻找引用值时，会首先检索其在栈中的地址，取得地址后从堆中获得实体
* 如何将字符串转化为数字，  
   例如'12.3b'?\* parseFloat('12.3b');
    
   正则表达式，'12.3b'.match(/(\d)+(\.)?(\d)+/g)[0] \* 1, 但是这个不太靠谱，提供一种思路而已。
* 如何将浮点数点左边的数每三位添加一个逗号，如12000000.11转化为『12,000,000.11』?  
   function commafy(num){
    
   return num && num.toString().replace(/(\d)(?=(\d{3})+\.)/g, function($1, $2){
  return $2 + ',';
  });
  }
* 如何实现数组的随机排序？  
   方法一：
  var arr = [1,2,3,4,5,6,7,8,9,10];
    
   function randSort1(arr){
    
   for(var i = 0,len = arr.length;i < len; i++ ){
    
   var rand = parseInt(Math.random()\*len);
    
   var temp = arr[rand];
    
   arr[rand] = arr[i];  
  arr[i] = temp;
    
   }
    
   return arr;
    
   }
    
   console.log(randSort1(arr));
    
  方法二：
  var arr = [1,2,3,4,5,6,7,8,9,10];
  function randSort2(arr){
  var mixedArray = [];
  while(arr.length > 0){
  var randomIndex = parseInt(Math.random()\*arr.length);
  mixedArray.push(arr[randomIndex]);
  arr.splice(randomIndex, 1);
  }
  return mixedArray;
  }
  console.log(randSort2(arr));
    
  方法三：
  var arr = [1,2,3,4,5,6,7,8,9,10];
  arr.sort(function(){
  return Math.random() - 0.5;
  })
  console.log(arr);
* Javascript如何实现继承？  
   1、构造继承
  2、原型继承
  3、实例继承
  4、拷贝继承
  原型prototype机制或apply和call方法去实现较简单，建议使用构造函数与原型混合方式。
  function Parent(){
  this.name = 'wang';
  }
  function Child(){
  this.age = 28;
  }
  Child.prototype = new Parent();//继承了Parent，通过原型
  var demo = new Child();
  alert(demo.age);
  alert(demo.name);//得到被继承的属性
* Javascript作用链域?  
   全局函数无法查看局部函数的内部细节，但局部函数可以查看其上层的函数细节，直至全局细节。
  当需要从局部函数查找某一属性或方法时，如果当前作用域没有找到，就会上溯到上层作用域查找，
  直至全局函数，这种组织形式就是作用域链。
* 谈谈This对象的理解。  
   this总是指向函数的直接调用者（而非间接调用者）；
  如果有new关键字，this指向new出来的那个对象；
  在事件中，this指向触发这个事件的对象，特殊的是，IE中的attachEvent中的this总是指向全局对象Window；
* eval是做什么的？  
  它的功能是把对应的字符串解析成JS代码并运行；
  应该避免使用eval，不安全，非常耗性能（2次，一次解析成js语句，一次执行）。
  由JSON字符串转换为JSON对象的时候可以用eval，var obj =eval('('+ str +')');
* 什么是window对象?   
  什么是document对象?window对象是指浏览器打开的窗口。
  document对象是Documentd对象（HTML 文档对象）的一个只读引用，window对象的一个属性。
* null，undefined 的区别？  
  null 表示一个对象是“没有值”的值，也就是值为“空”；
  undefined 表示一个变量声明了没有初始化(赋值)；
  undefined不是一个有效的JSON，而null是；
  undefined的类型(typeof)是undefined；
  null的类型(typeof)是object；
  Javascript将未赋值的变量默认值设为undefined；
  Javascript从来不会将变量设为null。它是用来让程序员表明某个用var声明的变量时没有值的。
  typeof undefined
  //"undefined"
  undefined :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 undefined；
  例如变量被声明了，但没有赋值时，就等于undefined
  typeof null
  //"object"
  null : 是一个对象(空对象, 没有任何属性和方法)；
  例如作为函数的参数，表示该函数的参数不是对象；
  注意：
  在验证null时，一定要使用　=== ，因为 == 无法分别 null 和　undefined
  null == undefined // true
  null === undefined // false
  再来一个例子：
  null
  Q：有张三这个人么？
  A：有！
  Q：张三有房子么？
  A：没有！
  undefined
  Q：有张三这个人么？
  A：有！
  Q: 张三有多少岁？
  A: 不知道（没有被告诉）
* 写一个通用的事件侦听器函数。   
   markyun.Event = {
  // 页面加载完成后
  readyEvent : function(fn) {
  if (fn==null) {
  fn=document;
  }
  var oldonload = window.onload;
  if (typeof window.onload != 'function') {
  window.onload = fn;
  } else {
  window.onload = function() {
  oldonload();
  fn();
  };
  }
  },
  // 视能力分别使用dom0||dom2||IE方式 来绑定事件
  // 参数： 操作的元素,事件名称 ,事件处理程序
  addEvent : function(element, type, handler) {
  if (element.addEventListener) {
  //事件类型、需要执行的函数、是否捕捉
  element.addEventListener(type, handler, false);
  } else if (element.attachEvent) {
  element.attachEvent('on' + type, function() {
  handler.call(element);
  });
  } else {
  element['on' + type] = handler;
  }
  },
  // 移除事件
  removeEvent : function(element, type, handler) {
  if (element.removeEventListener) {
  element.removeEventListener(type, handler, false);
  } else if (element.datachEvent) {
  element.detachEvent('on' + type, handler);
  } else {
  element['on' + type] = null;
  }
  },
  // 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)
  stopPropagation : function(ev) {
  if (ev.stopPropagation) {
  ev.stopPropagation();
  } else {
  ev.cancelBubble = true;
  }
  },
  // 取消事件的默认行为
  preventDefault : function(event) {
  if (event.preventDefault) {
  event.preventDefault();
  } else {
  event.returnValue = false;
  }
  },
  // 获取事件目标
  getTarget : function(event) {
  return event.target || event.srcElement;
  },
  // 获取event对象的引用，取到事件的所有信息，确保随时能使用event；
  getEvent : function(e) {
  var ev = e || window.event;
  if (!ev) {
  var c = this.getEvent.caller;
  while (c) {
  ev = c.arguments[0];
  if (ev && Event == ev.constructor) {
  break;
  }
  c = c.caller;
  }
  }
  return ev;
  }
  };
* ["1", "2", "3"].map(parseInt) 答案是多少？  
  parseInt() 函数能解析一个字符串，并返回一个整数，需要两个参数 (val, radix)，
  其中 radix 表示要解析的数字的基数。【该值介于 2 ~ 36 之间，并且字符串中的数字不能大于radix才能正确返回数字结果值】;
    
  但此处 map 传了 3 个 (element, index, array),我们重写parseInt函数测试一下是否符合上面的规则。
  f  
  unction parseInt(str, radix) {
  return str+'-'+radix;
  };
  var a=["1", "2", "3"];
  a.map(parseInt); // ["1-0", "2-1", "3-2"] 不能大于radix
  因为二进制里面，没有数字3,导致出现超范围的radix赋值和不合法的进制解析，才会返回NaN
    
  所以["1", "2", "3"].map(parseInt)   
  答案也就是：[1, NaN, NaN]
  详细解析：
* 事件是？  
  IE与火狐的事件机制有什么区别？ 如何阻止冒泡？ 1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。
  2. 事件处理机制：IE是事件冒泡、Firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件；
  3. ev.stopPropagation();（旧ie的方法 ev.cancelBubble = true;）
* 什么是闭包（closure），为什么要用它？  
  包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。
  闭包的特性：
    
  1.函数内再嵌套函数
    
  2.内部函数可以引用外层的参数和变量
    
  3.参数和变量不会被垃圾回收机制回收
  //li节点的onclick事件都能正确的弹出当前被点击的li索引
  <ul id="testUL">
  <li> index = 0</li>
  <li> index = 1</li>
  <li> index = 2</li>
  <li> index = 3</li>
  </ul>
  <script type="text/javascript">
  var nodes = document.getElementsByTagName("li");
  for(i = 0;i<nodes.length;i+= 1){
  nodes[i].onclick = (function(i){
  return function() {
  console.log(i);
  } //不用闭包的话，值每次都是4
  })(i);
  }
  </script>
  执行say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在
  使得Javascript的垃圾回收机制GC不会收回say667()所占用的资源
  因为say667()的内部函数的执行需要依赖say667()中的变量
  这是对闭包作用的非常直白的描述
  function say667() {
  // Local variable that ends up within closure
  var num = 666;
  var sayAlert = function() {
  alert(num);
  }
  num++;
  return sayAlert;
  }
  var sayAlert = say667();
  sayAlert()//执行结果应该弹出的667
* javascript 代码中的"use strict";是什么意思 ?   
  使用它区别是什么？use strict是一种ECMAscript 5 添加的（严格）运行模式,这种模式使得 Javascript 在更严格的条件下运行,
  使JS编码更加规范化的模式,消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为。
  默认支持的糟糕特性都会被禁用，比如不能用with，也不能在意外的情况下给全局变量赋值;
    
  全局变量的显示声明,函数必须声明在顶层，不允许在非函数代码块内声明函数,arguments.callee也不允许使用；
  消除代码运行的一些不安全之处，保证代码运行的安全,限制函数中的arguments修改，严格模式下的eval函数的行为和非严格模式的也不相同;
    
  提高编译器效率，增加运行速度；
  为未来新版本的Javascript标准化做铺垫。
* 如何判断一个对象是否属于某个类？   
   使用instanceof （待完善）
  if(a instanceof Person){
  alert('yes');
  }
* new操作符具体干了什么呢?   
  1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。
  2、属性和方法被加入到 this 引用的对象中。
  3、新创建的对象由 this 所引用，并且最后隐式的返回 this 。
  var obj = {};
  obj.\_\_proto\_\_ = Base.prototype;
  Base.call(obj);
* Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？  
  hasOwnProperty
  javaScript中hasOwnProperty函数方法是返回一个布尔值，指出一个对象是否具有指定名称的属性。此方法无法检查该对象的原型链中是否具有该属性；该属性必须是对象本身的一个成员。
  使用方法：
  object.hasOwnProperty(proName)
  其中参数object是必选项。一个对象的实例。
  proName是必选项。一个属性名称的字符串值。
  如果 object 具有指定名称的属性，那么JavaScript中hasOwnProperty函数方法返回 true，反之则返回 false。
* JSON 的了解？  
  JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。
  它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小
  如：{"age":"12", "name":"back"}
  JSON字符串转换为JSON对象:
  var obj =eval('('+ str +')');
  var obj = str.parseJSON();
  var obj = JSON.parse(str);
  JSON对象转换为JSON字符串：
  var last=obj.toJSONString();
  var last=JSON.stringify(obj);
* [].forEach.call($$("\*"),function(a){a.style.outline="1px solid #"+(~~(Math.random()\*(1<<24))).toString(16)}) 能解释一下这段代码的意思吗？
* js延迟加载的方式有哪些？defer和async、动态创建DOM方式（用得最多）、按需异步载入js
* Ajax 是什么? 如何创建一个Ajax？ajax的全称：Asynchronous Javascript And XML。
  异步传输+js+xml。
  所谓异步，在这里简单地解释就是：向服务器发送请求的时候，我们不必等待结果，而是可以同时做其他的事情，等到有了结果它自己会根据设定进行后续操作，与此同时，页面是不会发生整页刷新的，提高了用户体验。
  (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
* Ajax 解决浏览器缓存问题？  
  1、在ajax发送请求前加上 anyAjaxObj.setRequestHeader("If-Modified-Since","0")。
    
  2、在ajax发送请求前加上 anyAjaxObj.setRequestHeader("Cache-Control","no-cache")。
    
  3、在URL后面加上一个随机数： "fresh=" + Math.random();。
          
  4、在URL后面加上时间戳："nowtime=" + new Date().getTime();。   
  5、如果是使用jQuery，直接这样就可以了 $.ajaxSetup({cache:false})。这样页面的所有ajax都会执行这条语句就是不需要保存缓存记录。
* 如何解决跨域问题?（开放问题）jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面
* 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
* DOM操作——怎样添加、移除、移动、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() //在已有的子节点前插入一个新的子节点
    
  （3）查找
  getElementsByTagName() //通过标签名称
  getElementsByName() //通过元素的Name属性的值(IE容错能力较强，会得到一个数组，其中包括id等于name值的)
  getElementById() //通过元素Id，唯一性
* 如何判断当前脚本运行在浏览器还是node环境中？（阿里）this === window ? 'browser' : 'node';
  通过判断Global对象是否为window，如果不为window，当前脚本没有运行在浏览器中
* 移动端最小触控区域是多大？
* 移动端的点击事件的有延迟，时间是多久，为什么会有？ 怎么解决这个延时？（click 有 300ms 延迟,为了实现safari的双击事件的设计，浏览器要知道你是不是要双击操作。）
* 检测浏览器版本版本有哪些方式？功能检测、userAgent特征检测
  比如：navigator.userAgent
  //"Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_2) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/537.36"

## 前端框架

* Vue 生命周期的理解。
    
  简单版本：罗列 beforeCreate、created、beforeMount、mounted、beforeUpdate、updated、beforeDestroy、destroyed 钩子函数。详细版本：从创建一个实例开始，也就是 new Vue() 的对象过程当中，介绍生命周期函数在什么情况下被调用
* Vue 双向数据绑定如何实现？  
  答到关键点就好了，通过 getter 和 setter 劫持了对对象赋值的过程，在这个过程中可以进行更新 dom 操作等等。
  双向绑定具体使用场景
* Vue 中对数组的操作，提供怎样的变异方法。
    
   push(),pop(),shift(),unshift(),splice(),sort(),reverse() 由于 JavaScript 的限制，Vue 不能检测以下变动的数组：
  当你利用索引直接设置一个项时，例如：vm.items[indexOfItem] = newValue
  当你修改数组的长度时，例如：vm.items.length = newLength
  以下两种方式都可以实现和 vm.items[indexOfItem] = newValue 相同的效果，同时也将触发状态更新：
  // Vue.set
  Vue.set(vm.items, indexOfItem, newValue)
  // Array.prototype.splice
  vm.items.splice(indexOfItem, 1, newValue)
* 使用过 vue-router，两种不同的路由模式?   
  自己如何实现类似的前端页面路由？把组件映射到对应的路由，通过改变url来渲染不同的页面
    
  vue-router两种模式:
    
   1.默认hash模式，每次url只会改变`#`后面对应的值，页面就不会重新加载，并且也不需要服务器端作任何配置。
    
   2.history模式，url就会正常http://yoursite.com/user/id，只需要添加配置mode:'history',同时需要后端配置，不然页面重新刷新，会匹配不到任何资源。   
  实现：
    
   window.location.href 打开新页面
    
   window.location.search 获取参数
    
   <a></a> 标签页面跳转，锚点链接
* vue 项目中，使用Axios库进行ajax 请求?  
   在哪实现统一的错误处理 Axios ，是一个基于 promise 的 HTTP 库，可以用在浏览器和 node.js 中。
    
   在拦截器中，进行统一的错误处理：
    
   ```
    
   axios.interceptors.request.use(function (config) {
    
   // 在发送请求之前做些什么
    
   return config;
    
   }, function (error) {
  // 对请求错误做些什么
  return Promise.reject(error);
  });
    
   // 添加响应拦截器
    
   axios.interceptors.response.use(function (response) {
    
   // 对响应数据做点什么
    
   return response;
  },  
   function (error) {
  // 对响应错误做点什么
    
   return Promise.reject(error);
  });
    
   ```
* 如何搭建整个vue项目？ 使用那些前端构建工具（webpack/gulp）? 是否使用了Vue-cli 脚手架进行快速构建项目，修改了里面的那些配置?
* React 使用场景？   
   逻辑复杂单页应用，偏中后台管理系统，纯展示性的UI页面不合适、
* 描述一下React 生命周期 渲染过程调用到的生命周期函数，  
   主要几个要知道；
  \* constructor
  \* getInitialState
  \* getDefaultProps
  \* componentWillMount
  \* render
  \* componentDidMount
    
  更新过程
  \* componentWillReceiveProps
  \* shouldComponentUpdate
  \* componentWillUpdate
  \* render
  \* componentDidUpdate
    
  卸载过程
  componentWillUnmount
* 实现组件有哪些方式？  
   React.createClass 使用API来定义组件
  React ES6 class component 用 ES6 的class 来定义组件
  Functional stateless component 通过函数定义无状态组件
* 应该在React生命周期的什么阶段发出ajax请求，为什么？   
   AJAX请求应在 componentDidMount函数 进行请求。
* shouldComponentUpdate函数有什么作用？   
   shouldComponentUpdate是一个允许我们自行决定某些组件（以及他们的子组件）是否进行更新的生命周期函数，reconciliation的最终目的是尽可能以最有效的方式去根据新的state更新UI，
  如果你已经知道UI的哪些状态无需进行改变，就没必要去让React去判断它是否该改变。 让shouldComponentUpdate返回falss, React就会让当前的组件和其子组件保持不变。
* 当组件的setState函数被调用之后，发生了什么？   
  会做的第一件事就是把你传递给setState的参数对象合并到组件原先的state。这个事件会导致一个“reconciliation”（调和）的过程。reconciliation的最终目标就是，
  尽可能以最高效的方法，去基于新的state来更新UI。为了达到这个目的，React会构建一个React元素树（你可以把这个想象成一个表示UI的一个对象）。一旦这个树构建完毕，
  React为了根据新的state去决定UI要怎么进行改变，它会找出这棵新树和旧树的不同之处。React能够相对精确地找出哪些位置发生了改变以及如何发生了什么变化，
  并且知道如何只通过必要的更新来最小化重渲染。
* 为什么循环产生的组件中要利用上key这个特殊的prop？   
  Keys负责帮助React跟踪列表中哪些元素被改变/添加/移除。React利用子元素的key在比较两棵树的时候，快速得知一个元素是新的还是刚刚被移除。没有keys，React也就不知道当前哪一个的item被移除了。
* 什么时候应该选择用class实现一个组件，什么时候用一个函数实现一个组件？   
  组件用到了state或者用了生命周期函数，那么就该使用Class component。其他情况下，应使用Functional component。