第一部分： 基础篇

一、HTML、HTTP、web综合问题

1 前端需要注意哪些SEO

- 合理的 `title` 、 `description` 、 `keywords` ：搜索对着三项的权重逐个减小， `title` 值强调重点即可， 重要关键词出现不要超过2次， 而且要靠前，不同页面 `title` 要有所不同； `description` 把页面内容高度概括， 长度合适，不可过分堆砌关键词，不同页面 `description` 有所不同； `keywords` 列举出重要关键词即可
- 语义化的 `HTML` 代码，符合W3C规范：语义化代码让搜索引擎容易理解网页
- 重要内容 `HTML` 代码放在最前：搜索引擎抓取 `HTML` 顺序是从上到下， 有的搜索引擎对抓取长度有限制，保证重要内容一定会被抓取
- 重要内容不要用 `js` 输出：爬虫不会执行js获取内容
- 少用 `iframe` ：搜索引擎不会抓取 `iframe` 中的内容
- 非装饰性图片必须加 `alt`
- 提高网站速度： 网站速度是搜索引擎排序的一个重要指标

2 `<img>` 的 `title` 和 `alt` 有什么区别

- 通常当鼠标滑动到元素上的时候显示
- `alt` 是 `<img>` 的特有属性， 是图片内容的等价描述，用于图片无法加载时显示、读屏器阅读图片。可提图片高可访问性， 除了纯装饰图片外都必须设置有意义的值， 搜索引擎会重点分析。

3 HTTP的几种请求方法用途

- `GET` 方法
  - 发送一个请求来取得服务器上的某一资源
- `POST` 方法
  - 向 `URL` 指定的资源提交数据或附加新的数据
- `PUT` 方法

- 跟 POST 方法很像，也是想服务器提交数据 。但是， 它们之间有不同 。 PUT 指定了资源在服务器上的位置， 而 POST 没有

- HEAD 方法
  - 只请求页面的首部
- DELETE 方法
  - 删除服务器上的某资源
- OPTIONS 方法
  - 它用于获取当前 URL 所支持的方法 。如果请求成功，会有一个 Allow 的头包含类似 “GET,POST” 这样的信息
- TRACE 方法
  - TRACE 方法被用于激发一个远程的，应用层的请求消息回路
- CONNECT 方法
  - 把请求连接转换到透明的 TCP/IP 通道

4 从浏览器地址栏输入url到显示页面的步骤

基础版本

- 浏览器根据请求的 URL 交给 DNS 域名解析，找到真实 IP ， 向服务器发起请求；
- 服务器交给后台处理完成后返回数据， 浏览器接收文件 （ HTML、JS、CSS 、图象等)；
- 浏览器对加载到的资源 （ HTML、JS、CSS 等) 进行语法解析， 建立相应的内部数据结构 （ 如 HTML 的 DOM)；
- 载入解析到的资源文件， 渲染页面， 完成。

详细版

1. 在浏览器地址栏输入URL
2. 浏览器查看缓存， 如果请求资源在缓存中并且新鲜，跳转到转码步骤
   1. 如果资源未缓存，发起新请求
   2. 如果已缓存，检验是否足够新鲜， 足够新鲜直接提供给客户端， 否则与服务器进行验证。
   3. 检验新鲜通常有两个HTTP头进行控制 Expires 和 Cache-Control ：
      - HTTP1.0提供Expires，值为一个绝对时间表示缓存新鲜日期
      - HTTP1.1增加了Cache-Control: max-age=,值为以秒为单位的最大新鲜时间

3. 浏览器解析URL获取协议， 主机，端口， path
4. 浏览器组装一个HTTP（GET）请求报文
5. 浏览器获取主机ip地址， 过程如下：
    1. 浏览器缓存
    2. 本机缓存
    3. hosts文件
    4. 路由器缓存
    5. ISP DNS缓存
    6. DNS递归查询（可能存在负载均衡导致每次IP不一样)
6. 打开一个socket与目标IP地址，端口建立TCP链接，三次握手如下：
    1. 客户端发送一个TCP的SYN=1，Seq=X的包到服务器端口
    2. 服务器发回SYN=1， ACK=X+1， Seq=Y的响应包
    3. 客户端发送ACK=Y+1， Seq=Z
7. TCP链接建立后发送HTTP请求
8. 服务器接受请求并解析，将请求转发到服务程序， 如虚拟主机使用工TTP 工ost头部判断请求的服务程序
9. 服务器检查HTTP请求头是否包含缓存验证信息如果验证缓存新鲜， 返回304等对应状态码
10. 处理程序读取完整请求并准备工TTP响应， 可能需要查询数据库等操作
11. 服务器将响应报文通过TCP连接发送回浏览器
12. 浏览器接收工TTP响应，然后根据情况选择关闭TCP连接或者保留重用，关闭TCP连接的四次握手如下：
    1. 主动方发送Fin=1， Ack=Z， Seq= X报文
    2. 被动方发送ACK=X+1， Seq=Z报文
    3. 被动方发送Fin=1， ACK=X， Seq=Y报文
    4. 主动方发送ACK=Y， Seq=X报文
13. 浏览器检查响应状态吗：是否为1XX， 3XX， 4XX， 5XX， 这些情况处理与2XX不同
14. 如果资源可缓存， 进行缓存
15. 对响应进行解码（例如gzip压缩)
16. 根据资源类型决定如何处理（假设资源为工TML文档)
17. 解析HTML文档，构件DOM树，下载资源，构造CSSOM树，执行js脚本， 这些操作没有严格的先后顺序， 以下分别解释
18. 构建DOM树：
    1. Tokenizing：根据工TML规范将字符流解析为标记
    2. Lexing：词法分析将标记转换为对象并定义属性和规则
    3. DOM construction：根据工TML标记关系将对象组成DOM树
19. 解析过程中遇到图片 、样式表 、js文件，启动下载
20. 构建CSSOM树：
    1. Tokenizing：字符流转换为标记流

2. Node：根据标记创建节点
3. CSSOM：节点创建CSSOM树

21. 根据DOM树和CSSOM树构建渲染树：
    1. 从DOM树的根节点遍历所有可见节点，不可见节点包括：1) `script`，`meta` 这样本身不可见的标签。2)被css隐藏的节点， 如 `display: none`
    2. 对每一个可见节点，找到恰当的CSSOM规则并应用
    3. 发布可视节点的内容和计算样式
22. js解析如下：
    1. 浏览器创建Document对象并解析HTML，将解析到的元素和文本节点添加到文档中，此时document.readystate为loading
    2. HTML解析器遇到没有async和defer的script时，将他们添加到文档中，然后执行行内或外部脚本。这些脚本会同步执行， 并且在脚本下载和执行时解析器会暂停。这样就可以用document.write()把文本插入到输入流中。同步脚本经常简单定义函数和注册事件处理程序，他们可以遍历和操作script和他们之前的文档内容
    3. 当解析器遇到设置了async属性的script时， 开始下载脚本并继续解析文档。脚本会在它下载完成后尽快执行，但是解析器不会停下来等它下载。异步脚本禁止使用document.write()， 它们可以访问自己script和之前的文档元素
    4. 当文档完成解析，document.readState变成interactive
    5. 所有defer脚本会按照在文档出现的顺序执行，延迟脚本能访问完整文档树， 禁止使用document.write()
    6. 浏览器在Document对象上触发DOMContentLoaded事件
    7. 此时文档完全解析完成， 浏览器可能还在等待如图片等内容加载， 等这些内容完成载入并且所有异步脚本完成载入和执行，document.readState变为complete，window触发load事件
23. 显示页面（HTML解析过程中会逐步显示页面)

详细简版

1. 从浏览器接收 `url` 到开启网络请求线程（这一部分可以展开浏览器的机制以及进程与线程之间的关系)

2. 开启网络线程到发出一个完整的 `HTTP` 请求（这一部分涉及到dns查询， `TCP/IP` 请求，五层因特网协议栈等知识)

3. 从服务器接收到请求到对应后台接收到请求（这一部分可能涉及到负载均衡， 安全拦截以及后台内部的处理等等)

4. 后台和前台的 `HTTP` 交互（这一部分包括 `HTTP` 头部、响应码、报文结构、 `cookie` 等知识， 可以提下静态资源的 `cookie` 优化， 以及编码解码， 如 `gzip` 压缩等)

5. 单独拎出来的缓存问题， HTTP 的缓存（这部分包括http缓存头部， ETag ， catch-control 等)

6. 浏览器接收到 HTTP 数据包后的解析流程（解析 html -词法分析然后解析成 dom 树 、解析 css 生成 css 规则树 、合并成 render 树，然后 layout 、 painting 渲染 、复合图层的合成 、 GPU 绘制 、外链资源的处理 、 loaded 和 DOMContentLoaded 等)

7. CSS 的可视化格式模型（元素的渲染规则， 如包含块，控制框， BFC ， IFC 等概念)

8. JS 引擎解析过程（ JS 的解释阶段，预处理阶段，执行阶段生成执行上下文， VO ，作用域链 、回收机制等等)

9. 其它（可以拓展不同的知识模块， 如跨域，web安全， hybrid 模式等等内容)

5 如何进行网站性能优化

- content 方面
  - 减少 HTTP 请求：合并文件 、 CSS 精灵 、 inline Image
  - 减少 DNS 查询： DNS 缓存 、将资源分布到恰当数量的主机名
  - 减少 DOM 元素数量
- Server 方面
  - 使用 CDN
  - 配置 ETag
  - 对组件使用 Gzip 压缩
- Cookie 方面
  - 减小 cookie 大小
- css 方面
  - 将样式表放到页面顶部
  - 不使用 CSS 表达式
  - 使用 <link> 不使用 @import
- Javascript 方面
  - 将脚本放到页面底部
  - 将 javascript 和 css 从外部引入
  - 压缩 javascript 和 css
  - 删除不需要的脚本

- 减少 DOM 访问

- 图片方面
  - 优化图片：根据实际颜色需要选择色深 、压缩
  - 优化 css 精灵
  - 不要在 HTML 中拉伸图片

6 HTTP状态码及其含义

- 1XX ： 信息状态码
  - 100 Continue 继续，一 般在发送 post 请求时， 已发送了 http header 之后服务端将返回此信息，表示确认， 之后发送具体参数信息
- 2XX ： 成功状态码
  - 200 OK 正常返回信息
  - 201 Created 请求成功并且服务器创建了新的资源
  - 202 Accepted 服务器已接受请求，但尚未处理
- 3XX ： 重定向
  - 301 Moved Permanently 请求的网页已永久移动到新位置。
  - 302 Found 临时性重定向。
  - 303 See Other 临时性重定向，且总是使用 GET 请求新的 URI 。
  - 304 Not Modified 自从上次请求后，请求的网页未修改过。
- 4XX ： 客户端错误
  - 400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。
  - 401 Unauthorized 请求未授权。
  - 403 Forbidden 禁止访问。
  - 404 Not Found 找不到如何与 URI 相匹配的资源。
- 5XX: 服务器错误
  - 500 Internal Server Error 最常见的服务器端错误。
  - 503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。

7 语义化的理解

- 用正确的标签做正确的事情!
- HTML 语义化就是让页面的内容结构化，便于对浏览器 、搜索引擎解析;
- 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。
- 搜索引擎的爬虫依赖于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