$HTML， HTTP，web综合问题

前端需要注意哪些SEO

1. 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
2. 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
3. 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
4. 重要内容不要用js输出：爬虫不会执行js获取内容
5. 少用iframe：搜索引擎不会抓取iframe中的内容
6. 非装饰性图片必须加alt
7. 提高网站速度：网站速度是搜索引擎排序的一个重要指标

web开发中会话跟踪的方法有哪些

1. cookie
2. session
3. url重写
4. 隐藏input
5. ip地址

`<img>`的`title`和`alt`有什么区别

1. `title`是global attributes之一，用于为元素提供附加的advisory information。通常当鼠标滑动到元素上的时候显示。
2. `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。

doctype是什么,举例常见doctype及特点

1. `<!doctype>`声明必须处于HTML文档的头部，在`<html>`标签之前，HTML5中不区分大小写
2. `<!doctype>`声明不是一个HTML标签，是一个用于告诉浏览器当前HTMl版本的指令
3. 现代浏览器的html布局引擎通过检查doctype决定使用兼容模式还是标准模式对文档进行渲染，一些浏览器有一个接近标准模型。
4. 在HTML4.01中`<!doctype>`声明指向一个DTD，由于HTML4.01基于SGML，所以DTD指定了标记规则以保证浏览器正确渲染内容
5. HTML5不基于SGML，所以不用指定DTD

常见dotype:

1. **HTML4.01 strict**：不允许使用表现性、废弃元素（如font）以及frameset。声明：`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
2. **HTML4.01 Transitional**:允许使用表现性、废弃元素（如font），不允许使用frameset。声明：`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
3. **HTML4.01 Frameset**:允许表现性元素，废气元素以及frameset。声明：`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">`
4. **XHTML1.0 Strict**:不使用允许表现性、废弃元素以及frameset。文档必须是结构良好的XML文档。声明：`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">`
5. **XHTML1.0 Transitional**:允许使用表现性、废弃元素，不允许frameset，文档必须是结构良好的XMl文档。声明：`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`
6. **XHTML 1.0 Frameset**:允许使用表现性、废弃元素以及frameset，文档必须是结构良好的XML文档。声明：`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">`
7. **HTML 5**: `<!doctype html>`

HTML全局属性(global attribute)有哪些

参考资料：MDN: html global attribute或者W3C HTML global-attributes

- `accesskey`:设置快捷键，提供快速访问元素如aaa在windows下的firefox中按`alt + shift + a`可激活元素
- `class`:为元素设置类标识，多个类名用空格分开，CSS和javascript可通过class属性获取元素
- `contenteditable`: 指定元素内容是否可编辑
- `contextmenu`: 自定义鼠标右键弹出菜单内容
- `data-*`: 为元素增加自定义属性
- `dir`: 设置元素文本方向
- `draggable`: 设置元素是否可拖拽
- `dropzone`: 设置元素拖放类型： copy, move, link
- `hidden`: 表示一个元素是否与文档。样式上会导致元素不显示，但是不能用这个属性实现样式效果
- `id`: 元素id，文档内唯一
- `lang`: 元素内容的的语言
- `spellcheck`: 是否启动拼写和语法检查
- `style`: 行内css样式
- `tabindex`: 设置元素可以获得焦点，通过tab可以导航
- `title`: 元素相关的建议信息
- `translate`: 元素和子孙节点内容是否需要本地化

什么是web语义化,有什么好处

web语义化是指通过HTML标记表示页面包含的信息，包含了HTML标签的语义化和css命名的语义化。
HTML标签的语义化是指：通过使用包含语义的标签（如h1-h6）恰当地表示文档结构
css命名的语义化是指：为html标签添加有意义的class，id补充未表达的语义，如Microformat通过添加符合规则的class描述信息
为什么需要语义化：

- 去掉样式后页面呈现清晰的结构
- 盲人使用读屏器更好地阅读
- 搜索引擎更好地理解页面，有利于收录

- 便团队项目的可持续运作及维护

HTTP method

1. 一台服务器要与HTTP1.1兼容，只要为资源实现**GET**和**HEAD**方法即可
2. **GET**是最常用的方法，通常用于**请求服务器发送某个资源**。
3. **HEAD**与GET类似，但**服务器在响应中值返回首部，不返回实体的主体部分**
4. **PUT**让服务器**用请求的主体部分来创建一个由所请求的URL命名的新文档，或者，如果那个URL已经存在的话，就用干这个主体替代它**
5. **POST**起初是用来向服务器输入数据的。实际上，通常会用它来支持HTML的表单。表单中填好的数据通常会被送给服务器，然后由服务器将其发送到要去的地方。
6. **TRACE**会在目的服务器端发起一个环回诊断，最后一站的服务器会弹回一个TRACE响应并在响应主体中携带它收到的原始请求报文。TRACE方法主要用于诊断，用于验证请求是否如愿穿过了请求/响应链。
7. **OPTIONS**方法请求web服务器告知其支持的各种功能。可以查询服务器支持哪些方法或者对某些特殊资源支持哪些方法。
8. **DELETE**请求服务器删除请求URL指定的资源

从浏览器地址栏输入url到显示页面的步骤(以HTTP为例)

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HTTP Host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HTTP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HTTP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HTML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HTML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HTML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）

HTTP request报文结构是怎样的

rfc2616中进行了定义：

1. 首行是**Request-Line**包括：**请求方法**，**请求URI**，**协议版本**，**CRLF**
2. 首行之后是若干行**请求头**，包括**general-header**，**request-header**或者**entity-header**，每个一行以CRLF结束
3. 请求头和消息实体之间有一个**CRLF分隔**
4. 根据实际请求需要可能包含一个**消息实体**
   一个请求报文例子如下：

```
GET /Protocols/rfc2616/rfc2616-sec5.html HTTP/1.1
Host: www.w3.org
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36
Referer: https://www.google.com.hk/
Accept-Encoding: gzip,deflate,sdch
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Cookie: authorstyle=yes
If-None-Match: "2cc8-3e3073913b100"
If-Modified-Since: Wed, 01 Sep 2004 13:24:52 GMT

name=qiu&age=25
```

HTTP response报文结构是怎样的

rfc2616中进行了定义：

1. 首行是状态行包括：**HTTP版本，状态码，状态描述**，后面跟一个CRLF
2. 首行之后是**若干行响应头**，包括：**通用头部，响应头部，实体头部**
3. 响应头部和响应实体之间用**一个CRLF空行**分隔
4. 最后是一个可能的**消息实体**
   响应报文例子如下：

```
HTTP/1.1 200 OK
Date: Tue, 08 Jul 2014 05:28:43 GMT
Server: Apache/2
Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT
ETag: "40d7-3e3073913b100"
Accept-Ranges: bytes
Content-Length: 16599
Cache-Control: max-age=21600
Expires: Tue, 08 Jul 2014 11:28:43 GMT
P3P: policyref="http://www.w3.org/2001/05/P3P/p3p.xml"
Content-Type: text/html; charset=iso-8859-1

{"name": "qiu", "age": 25}
```

如何进行网站性能优化

雅虎Best Practices for Speeding Up Your Web Site:

- content方面
    1. 减少HTTP请求：合并文件、CSS精灵、inline Image
    2. 减少DNS查询：DNS查询完成之前浏览器不能从这个主机下载任何任何文件。方法：DNS缓存、将资源分布到恰当数量的主机名，平衡并行下载和DNS查询
    3. 避免重定向：多余的中间访问
    4. 使Ajax可缓存
    5. 非必须组件延迟加载
    6. 未来所需组件预加载
    7. 减少DOM元素数量
    8. 将资源放到不同的域下：浏览器同时从一个域下载资源的数目有限，增加域可以提高并行下载量
    9. 减少iframe数量
    10. 不要404
- Server方面
    1. 使用CDN
    2. 添加Expires或者Cache-Control响应头

3. 对组件使用Gzip压缩
4. 配置ETag
5. Flush Buffer Early
6. Ajax使用GET进行请求
7. 避免空src的img标签

- Cookie方面
  1. 减小cookie大小
  2. 引入资源的域名不要包含cookie

- css方面
  1. 将样式表放到页面顶部
  2. 不使用CSS表达式
  3. 使用不使用@import
  4. 不使用IE的Filter

- Javascript方面
  1. 将脚本放到页面底部
  2. 将javascript和css从外部引入
  3. 压缩javascript和css
  4. 删除不需要的脚本
  5. 减少DOM访问
  6. 合理设计事件监听器

- 图片方面
  1. 优化图片：根据实际颜色需要选择色深、压缩
  2. 优化css精灵
  3. 不要在HTML中拉伸图片
  4. 保证favicon.ico小并且可缓存

- 移动方面
  1. 保证组件小于25k
  2. Pack Components into a Multipart Document

什么是渐进增强

渐进增强是指在web设计时强调可访问性、语义化HTML标签、外部样式表和脚本。保证所有人都能访问页面的基本内容和功能同时为高级浏览器和高带宽用户提供更好的用户体验。核心原则如下:

- 所有浏览器都必须能访问基本内容
- 所有浏览器都必须能使用基本功能
- 所有内容都包含在语义化标签中
- 通过外部CSS提供增强的布局
- 通过非侵入式、外部javascript提供增强功能
- end-user web browser preferences are respected

HTTP状态码及其含义

参考RFC 2616

- 1XX：信息状态码
  - **100 Continue**：客户端应当继续发送请求。这个临时相应是用来通知客户端它的部分请求已经被服务器接收，且仍未被拒绝。客户端应当继续发送请求的剩余部分，或者如果请求已经完成，忽略这个响应。服务器必须在请求万仇向客户端发送一个最终响应
  - **101 Switching Protocols**：服务器已经理解力客户端的请求，并将通过Upgrade消息头通知客户端采用不同的协议来完成这个请求。在发送完这个响应最后的空行后，服务器将会切换到Upgrade消息头中定义的那些协议。

- 2XX：成功状态码
  - **200 OK**：请求成功，请求所希望的响应头或数据体将随此响应返回
  - **201 Created**:
  - **202 Accepted**:
  - **203 Non-Authoritative Information**:
  - **204 No Content**:
  - **205 Reset Content**:
  - **206 Partial Content**:

- 3XX：重定向
  - **300 Multiple Choices**:
  - **301 Moved Permanently**:
  - **302 Found**:

    - **303 See Other**:
    - **304 Not Modified**:
    - **305 Use Proxy**:
    - **306 （unused）**：
    - **307 Temporary Redirect**:

- 4XX：客户端错误
    - **400 Bad Request**:
    - **401 Unauthorized**:
    - **402 Payment Required**:
    - **403 Forbidden**:
    - **404 Not Found**:
    - **405 Method Not Allowed**:
    - **406 Not Acceptable**:
    - **407 Proxy Authentication Required**:
    - **408 Request Timeout**:
    - **409 Conflict**:
    - **410 Gone**:
    - **411 Length Required**:
    - **412 Precondition Failed**:
    - **413 Request Entity Too Large**:
    - **414 Request-URI Too Long**:
    - **415 Unsupported Media Type**:
    - **416 Requested Range Not Satisfiable**:
    - **417 Expectation Failed**:

- 5XX: 服务器错误
    - **500 Internal Server Error**:
    - **501 Not Implemented**:
    - **502 Bad Gateway**:
    - **503 Service Unavailable**:
    - **504 Gateway Timeout**:
    - **505 HTTP Vers