注：实体首部字段是包含在请求报文和响应报文中的实体部分所使用的首部。

https://blog.csdn.net/Newbie___/article/details/107212575

1. 地址栏输入url
2. 浏览器查看是否命中强缓存，如果命中直接提供给客户端，否则与服务器进行验证。

- 如果资源未缓存，发起请求
- 如果已缓存，验证是否过期
- 验证过期通过http头部 cache-control(通用首部字段 http1.1)，和expires（http1.0 实体首部字段）验证

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. 浏览器获取域名ip地址，过程如下： 浏览器缓存 操作系统缓存 本地域名服务器查询 DNS查询
6. 与对应的ip地址建立TCP连接 三次握手：

- 第一次握手：客户端发送一个含有同步序列号的标志位的数据段给服务器，请求建立连接，服务端收到得出结论，自身接收没问题，客户端发送没问题
- 第二次握手：服务端收到客户端的请求，发送确定应答ACK标志和同步序列号SYN标志位的数据段响应。客户端收到得出结论：自己的发送和接收没问题，服务端的接收和发送没问题。
- 第三次握手：客户端收到数据段之后，再发送一个确认应答，表示已经收到。服务端收到，得出结论：客户端的接收没问题，自身的发送没问题。

7. 建立TCP连接后发送HTTP请求
8. 服务器接收请求并解析，将请求转发到服务程序，根据http头host判断请求的地址
9. 服务器检查请求头的if-none-macthed和if-modified-since，验证是否命中协商缓存，如果命中，返回304状态码，浏览器从缓存中读取，如果没有命中，服务器处理请求，返回数据
10. 服务器将响应报文发送到服务器
11. 浏览器接收http响应，然后根据情况选择关闭TCP连接或保留重用，关闭TCP需要四次挥手

- 第一次挥手：主动发发送控制位FIN，提出停止连接的请求
- 第二次挥手：被动方收到请求，发送确认位，表示已经收到了
- 第三次挥手：被动方发送控制位FIN,也提出停止连接
- 第四次挥手：主动发发送确认ACK，服务端收到确认即关闭请求，主动方等待2MSL（2个报文最大生存时间）时间之后关闭请求

12. 浏览器检查响应状态码，200继续解析，400、500的话就报错，300的话就重定向，这里会有个重定向计数器，避免多次的重定向，超过次数也会报错
13. 查看响应头，根据浏览器缓存字段进行缓存
14. 如果是gzip格式的话，首先解码
15. 根据资源类型决定如何处理

16. 根据html构建DOM树，有css的话构建CSSOM树，如果遇到script标签会判断async或defer，async会并行下载，然后阻塞执行，defer会并行下载，等待html解析完顺序执行 如果没有defer或async，那么就会阻塞渲染直到js执行完毕 DOM树和CSSOM树构建完成之后开始生成render树，确定页面元素的布局，样式

17. 在生成render树的过程中，浏览器开始调用GUI绘制，合成图层，将内容显示在屏幕上

详细简版：

- 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）
- 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）
- 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）
- 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）
- 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 ， catchcontrol 等）
- 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）
- 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）
- 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）
- 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）