从浏览器输入URL到看见页面发生了什么

* 1. 解析URL并生成HTTP请求
  2. DNS域名解析
  3. 委托协议栈发送消息
  4. 发起TCP连接（三次握手）
  5. HTTP发起请求
  6. 服务器处理请求
  7. 渲染页面：构建DOM树
  8. 关闭TCP连接：四次挥手

1. URL：统一资源定位符。
   1. 标准格式：protocol://[userinfo@]host[:port][/path][?query][#fragment]
      * 1. protocol【通信协议】、userinfo【格式为user：password，已弃用】、host【主机（域名）】、port【端口号，http默认端口为80，https默认为443】、path【路径，由任意个/分隔，开头的/表示根目录】、query【参数，键值对形式，通过&分隔】、fragment【锚点，#开始到最后，常见于链接锚点】
      1. 路径：
         1. 路径不以 / 结尾，会先将路径当做指向文件处理，若找不到该文件。再将路径当做指向目录处理
         2. 路径以 / 结尾，表示访问该目录下的默认文件【一般为index.html或default.htm】
         3. 路径省略不写，表示访问根目录下的默认文件
2. 同源策略：两个URL协议、域名、端口号必须完全相同。违背同源策略就是跨域，默认禁止跨域读写。跨域是浏览器拦截了返回数据，而非服务器
   1. 跨域解决方案：
      1. JSONP：非官方的跨域解决方案，利用script标签可跨域访问脚本的特性，只支持get请求。
         1. 创建自定义函数，用于处理返回的数据
         2. 动态创建一个script标签，设置src属性为接口地址，并附上callback参数，值为自定义函数名
         3. 服务端使用自定义函数来响应数据
      2. CORS：官方的跨域解决方案，支持get和post，在express服务端中设置
         1. response.setHeader('Access-Control-Allow-Origin','\*') 设置响应头，设置允许跨域
            1. 第二个属性为允许的域名，\*表示所有网页
3. HTTP协议：hypertext transport protocol，超文本传输协议，规定了浏览器和万维网服务器之间互相通信的规则约定
   1. 状态码：
      1. 1\*\*：信息，服务器收到请求，需要请求者继续执行操作
      2. 2\*\*：成功，操作被成功接收并处理
         1. 200【OK】
      3. 3\*\*：重定向，需要进一步的操作以完成请求
         1. 301【请求的资源被永久移动到新的URI】、302【临时移动】、303【查看其他地址】、304【所请求的资源未修改，服务器返回此状态码时，不会返回任何资源】、307【临时重定向】
      4. 4\*\*：客户端错误，请求包含语法错误或无法完成请求
         1. 400【客户端请求语法错误】、401【要求用户的身份认证】、403【服务器拒绝请求】、404【服务器无法找到资源】、408【服务端等待客户端发送的请求时间过长，超时】
      5. 5\*\*：服务端错误，服务器在处理请求的过程中发生了错误
         1. 500【服务器内部错误，无法完成请求】、501【服务器不支持请求的功能】、502【网关或代理服务器尝试执行请求时，从远程服务器接收到了一个无效响应】
   2. 请求：
      1. 若要发送自定义响应头属性，需在后端设置
      2. 请求报文：
         1. 请求行：格式为“请求方法 URL 协议版本 换行符”
         2. 请求头：键值对形式，说明请求体类型。常用的请求头有：
            1. Accept：浏览器能处理的内容类型、Accept-Charset：浏览器能显示的字符集、Accept-Encoding：浏览器能处理的压缩编码、
            2. Accept-Language：浏览器当前设置的语言、Connection：浏览器与服务器之间连接的类型、Cookie：当前页面设置的任何Cookie
            3. Host：发出请求的页面所在的域、Referer：发出请求的页面的URL、User-Agent：浏览器的用户代理字符串
         3. 空行：必须有
         4. 请求体：post put请求携带的数据，如username=admin&password=admin
      3. 传递参数：
         1. url直接传递：在url后面写上？参数名=值&参数名=值
         2. 请求体传递：将参数放入请求体中
      4. 常见的请求方法：
         1. GET【向服务器获取数据】、POST【将实体提交到指定资源】、PUT【上传文件，更新数据】、DELETE【删除服务器上的对象】、
         2. HEAD【获取报文首部，相比GET不返回报文主体部分】、OPTIONS【询问支持的请求方法，检查访问权限，如跨域权限】、
         3. CONNECT【要求在与代理服务器通信时建立隧道，使用隧道进行TCP通信】、TRACE【回显服务器收到的请求，用于测试或诊断】
      5. GET请求与POST请求的区别：
         1. GET请求是一个幂等【多次操作是否对结果有影响】的请求，POST请求不是一个幂等的请求
         2. 浏览器一般会对GET请求缓存，但一般不对POST请求缓存
         3. GET请求的请求体一般为空，POST请求的请求体一般为向服务器发送的数据
         4. Get请求把参数放入url传递，Post可将参数放入请求体。get更快，post更安全
         5. get提交的参数受限于url长度；post可以提交大量数据，支持更多的数据类型，受服务器限制
   3. 响应：
      1. 响应报文：
         1. 响应行：“协议版本 状态码 状态码原因短语”
         2. 响应头：键值对形式，对响应体的描述，常用的响应头有：
            1. Date：消息发送的时间，事件的描述格式由rfc822定义、server：服务器名称、Connection：浏览器与服务器之间连接的类型、
            2. Cache-Control：控制HTTP缓存、Content-Type：表示后面的文档属于什么MIME类型、Content-length：。。。
         3. 空行
         4. 响应体
   4. HTTP1.0和HTTP1.1的区别：
      1. 1.0默认使用非持久连接；1.1默认使用持久连接，使多个http请求复用一个TCP连接
      2. 1.0存在带宽浪费，需要某对象的一部分时，会传送整个对象，且不支持断点续传；1.1可以只请求资源的某个部分
      3. 1.0主要使用If-Modified-Since、Expires作为缓存判断标准；1.1引入了更多的缓存控制策略
      4. 1.0不传递主机名（hostname）；1.1新增了host字段
      5. 1.1新增了PUT、HEAD、OPTIONS等请求方法
   5. HTTP1.1和HTTP2.0的区别：
   6. HTTP协议的优点：简单快速、无连接、无状态、灵活【可传输任意类型的数据】
   7. HTTP协议的缺点：无状态、明文传输、不安全
4. HTTPS协议：超文本传输安全协议（Hypertext Transfer Protocol Secure）
   1. HTTP和HTTPS协议的区别：
      1. HTTPS协议需要CA证书；HTTP不需要
      2. HTTP协议是超文本传输协议，信息是明文传输的；HTTPS是具有安全性的SSL加密传输协议
      3. HTTP的端口是80；HTTPS的端口是443
      4. HTTP协议连接简单，没有状态；HTTPS协议可进行加密传输、身份认证，比HTTP更加安全
   2. HTTPS优点：可以认证用户和服务器、可以加密传输、是现行架构下最安全的解决方案
   3. HTTPS缺点：加密和解密耗费资源、协议的握手阶段费时，增加页面加载时间、SSL证书收费、不能在同一IP绑定多个域名
5. DNS协议：域名系统（Domain Name System），提供主机名到IP地址的转换服务
   1. DNS占用53号端口，同时使用TCP和UDP协议
   2. 完整查询过程：
      1. 首先会在浏览器的缓存中查找对应的IP地址，如果查找到直接返回，若找不到继续下一步
      2. 将请求发送给**本地DNS服务器**，在本地域名服务器缓存中查询，如果查找到，就直接将查找结果返回，若找不到继续下一步
      3. 本地DNS服务器向**根域名服务器**发送请求，根域名服务器会返回一个所查询域的顶级域名服务器地址
      4. 本地DNS服务器向**顶级域名服务器**发送请求，接受请求的服务器查询自己的缓存，如果有记录，就返回查询结果，如果没有就返回相关的下一级的权威域名服务器的地址
      5. 本地DNS服务器向**权威域名服务器**发送请求，域名服务器返回对应的结果
      6. 本地DNS服务器将返回结果保存在缓存中，便于下次使用
      7. 本地DNS服务器将返回结果返回给浏览器
6. 网络模型：
   1. 对等通信：模型的每一层都必须与目的端的对等层进行通信
   2. OSI七层模型：自上而下分别为应用层、表示层、会话层、传输层、网络层、数据链路层、物理层
      1. 应用层：为用户提供应用接口和网络服务。应用层的网络服务协议有：HTTP、HTTPS、FTP【文件传输】、POP3、SMTP【邮件传输】等
      2. 表示层：提供各种应用层数据的编码和转换功能，确保一个系统的应用层发送的数据能被另一个系统的应用层识别。base64
      3. 会话层：负责建立、管理和终止表示层实体之间的通信会话
      4. 传输层：建立了主机端到端【端口】的链接，为上层协议提供端到端的数据传输服务【规定传输方式】。TCP、UDP
      5. 网络层：通过IP寻址来建立两个节点之间的连接，将分组从源端传输层正确地送到目的端的传输层【规定传输路线】。IP协议
      6. 数据链路层：将数据合成为帧，使用链路层地址（MAC地址）访问介质，并进行差错检测【传输路线】
      7. 物理层：通过物理介质传输比特流
   3. TCP/IP五层协议：自上而下分别为应用层、传输层、网络层、数据链路层、物理层
      1. TCP/IP将OSI的应用层、表示层、会话层合并到了应用层，更简洁
7. TCP与UDP协议：都是传输层协议，属于TCP/IP协议族
   1. UDP：用户数据报协议
      1. 特点：面向无连接、有单播，多播，广播的功能、面向报文【仅对报文添加首部，不合并拆分】、不可靠性【不需要建立连接，接收端不对报文进行确认】、首部开销小【8字节，高效】
      2. 头部包含：两个十六位端口号【源端口和目标端口】、整个数据报文的长度、整个数据报文的校验和
      3. 使用场景：效率要求高，准确性要求低，如QQ、网络语音电话、广播通信
   2. TCP：传输控制协议
      1. 特点：面向连接、仅支持单播传输、面向字节流【不保留报文边界以字节流传输】、可靠传输、有拥塞控制、提供全双工通信、首部开销大【最小20字节，最大60字节】
      2. 使用场景：效率要求相对低，准确性要求高，如文件传输、远程登录
      3. 重传机制：发送数据后开启一个定时器，若没有收到ACK确认报文，则对报文进行重传
      4. 三次握手：建立一个TCP连接时，客户端和服务器总共需要发送3个包
8. WebSocket：
   1. 理解：是HTML5提供的一种浏览器与服务器进行全双工通讯的网络技术，属于应用层协议，基于TCP传输协议，并复用HTTP的握手通道。
   2. - 浏览器和服务器只需完成一次握手，两者之间就能创建持久性连接，进行双向数据传输
   3. 特点：支持双向通信、可以发送文本及二进制数据、建立在TCP协议上服务端实现容易、数据格式轻量性能开销小、没有同源限制、协议标识
   4. - 符为ws（加密则为wss），服务器网址为URL、与HTTP协议有良好兼容性，默认端口也为80和443
   5. 使用：
      1. let ws = new WebSocket（“ws：//localhost：9999”）；
      2. ws.onopen = function（）｛console.log（“open”；ws.send（‘hello’）；）｝ 连接建立后触发
      3. ws.onmessage = function（res）｛console.log（res.data）；｝ 服务端给客户端发送消息时触发
      4. ws.onclose = function（evt）｛console.log（“close”）；｝ 连接关闭时触发
9. 客户端与服务端的即时通讯：
   1. 短轮询：浏览器每隔一段时间向服务端发送http请求，若有数据更新，则响应。严重浪费资源
   2. 长轮询：客户端向服务端发送请求，服务端不直接进行响应，将请求挂起，有更新再响应。仍然造成资源浪费
   3. SSE：基于http协议，服务器向客户端声明，发送流信息，节约了资源
   4. WebSocket：允许双向通信