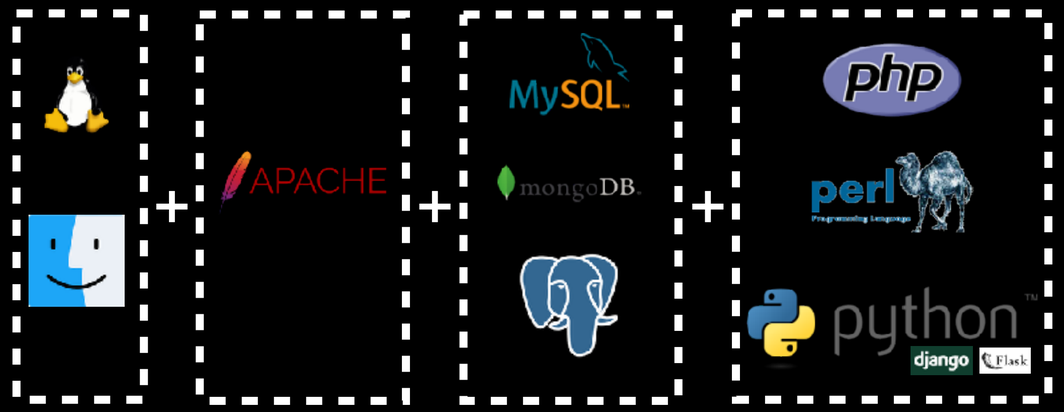
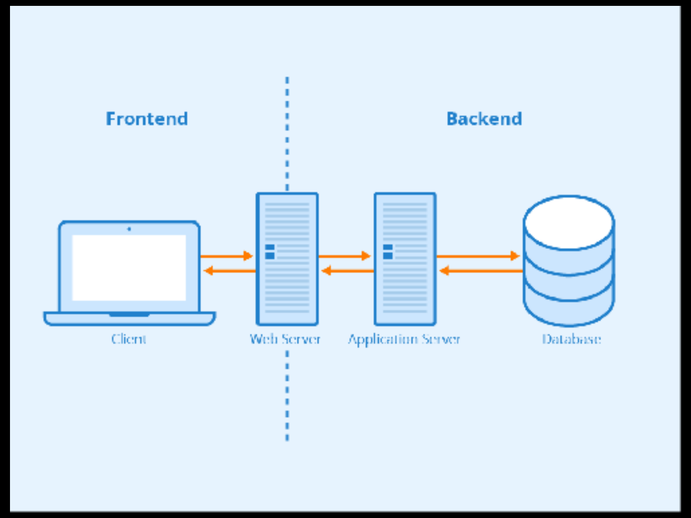


网站是如何实现的?

前端： HTML, CSS, JavaScript

后端： LAMP/MAMP

协议： HTTP/HTTPS



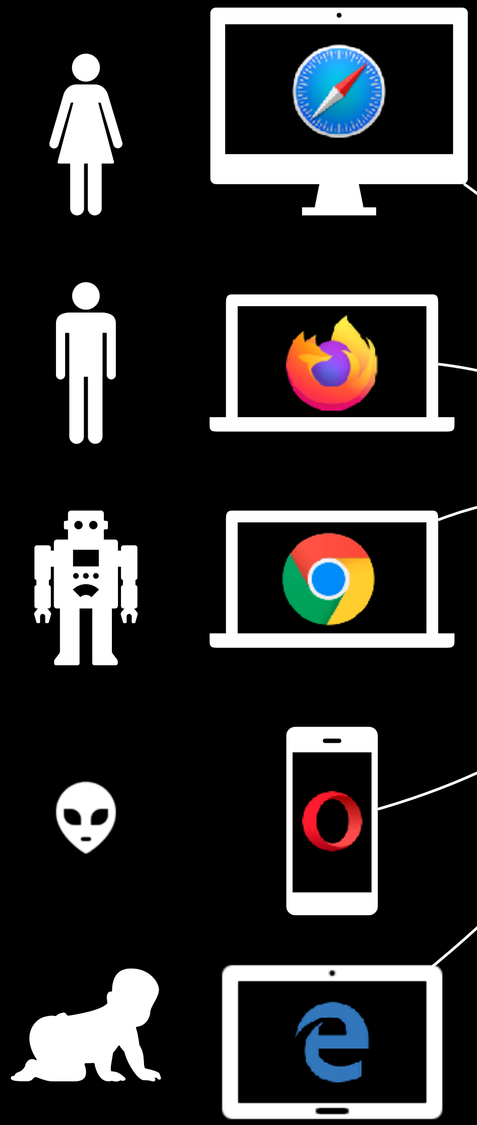
@theGeekDuck极客鸭

客户端（前端）：用户通过浏览器访问网站

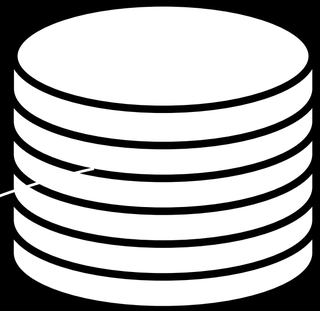
front-end：client side

服务器端（后端）：用户通过浏览器访问网站

back-end：server side



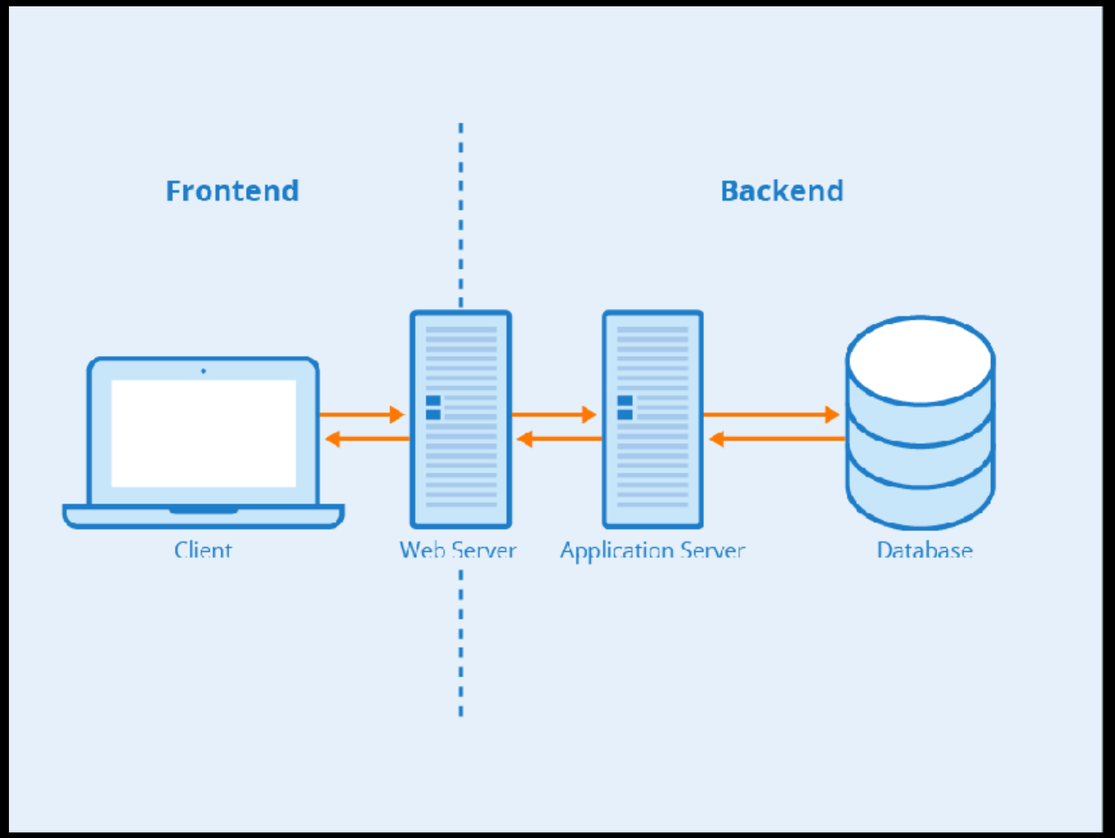
网络



硬盘

服务器

简化结构



图片引用：https://www.seobility.net/en/wiki/Frontend

Web Server与Application Server的区别?

658

Most of the times these terms Web Server and Application server are used interchangeably.

Following are some of the key differences in features of Web Server and Application Server:

- Web Server is designed to serve HTTP Content. App Server can also serve HTTP Content but is not limited to just HTTP. It can be provided other protocol support such as RMI/RPC
- Web Server is mostly designed to serve static content, though most Web Servers have plugins to support scripting languages like Perl, PHP, ASP, JSP etc. through which these servers can generate dynamic HTTP content.
- Most of the application servers have Web Server as integral part of them, that means App Server can do whatever Web Server is capable of. Additionally App Server have components and features to support Application level services such as Connection Pooling, Object Pooling, Transaction Support, Messaging services etc.
- As web servers are well suited for static content and app servers for dynamic content, most of the production environments have web server acting as reverse proxy to app server. That means while servicing a page request, static contents (such as images/Static HTML) are served by web server that interprets the request. Using some kind of filtering technique (mostly extension of requested resource) web server identifies dynamic content request and transparently forwards to app server

Example of such configuration is Apache Tomcat HTTP Server and Oracle (formerly BEA) WebLogic Server. Apache Tomcat HTTP Server is Web Server and Oracle WebLogic is Application Server.

In some cases the servers are tightly integrated such as IIS and .NET Runtime. IIS is web server. When equipped with .NET runtime environment, IIS is capable of providing application services.

https://stackoverflow.com/questions/936197/what-is-the-difference-between-application-server-and-web-server

后端组成四大件：LAMP/MAMP

Linux OS

Mac OS

Operating System
操作系统

+

APACHE

Web Server
网络服务器（处理访问请求）

+

MySQL

mongoDB.

Postgresql

Database
数据库（大规模储存，查询数据）

+

php

perl
Programming Language

python™

django

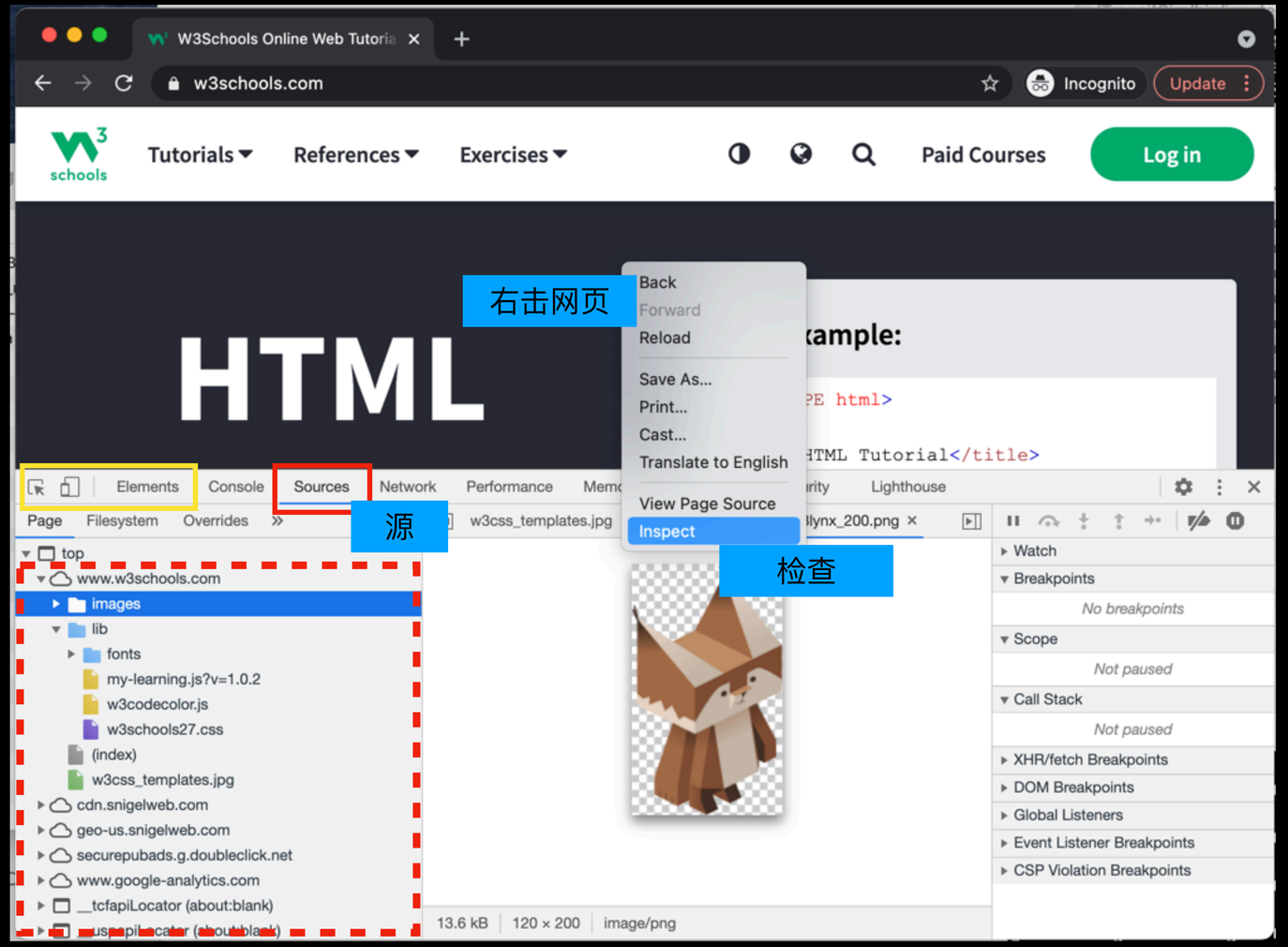
Flask

Programming Language
编程语言（实现后端逻辑任务）

前端三大件：HTML，CSS，JavaScript

打开网页，以下文件会从服务器传送到客户端浏览器（web browser）

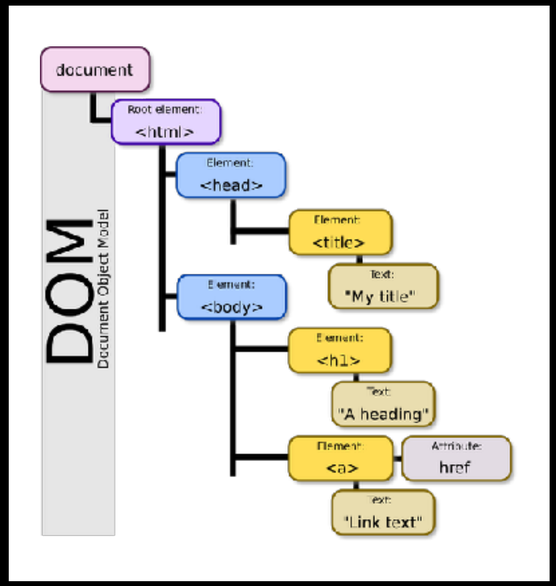
- HTML文件：网页的结构
- CSS文件（cascading style sheet）：个性化字体，格式
- JavaScript（JS）脚本文件：加载前端脚本，浏览器本地运行脚本
- 其他文件，如：图片（svg/png）



前端三大件之：数据模型（Data Object Model）

HTML, CSS, JavaScript

- DOM：树结构，统一网页格式
  - 元素（Element）父元素“嵌套”子元素
  - 可给定元素类型（class），id等
- HTML：描述基本DOM，网页的基本结构
- CSS：指定元素（根据id，类型，元素名称等）调整DOM中的风格(颜色，像素，大小，字体等等)
- JavaScript：前端编程改变DOM（交互）
  - TypeScript: 限制性的JavaScript
  - ECMAScript: JavaScript的国际标准



DOM

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>My title</title>
    <link rel="stylesheet" href="style.css">
    <script src="script.js"></script>
  </head>
  <body>
    <!-- page content —>
     <h1>A heading</h1>
     <a href=“www.bilibili.com”>Link text</a>
  </body>
</html>
```

前端三大件之：前端脚本JavaScript

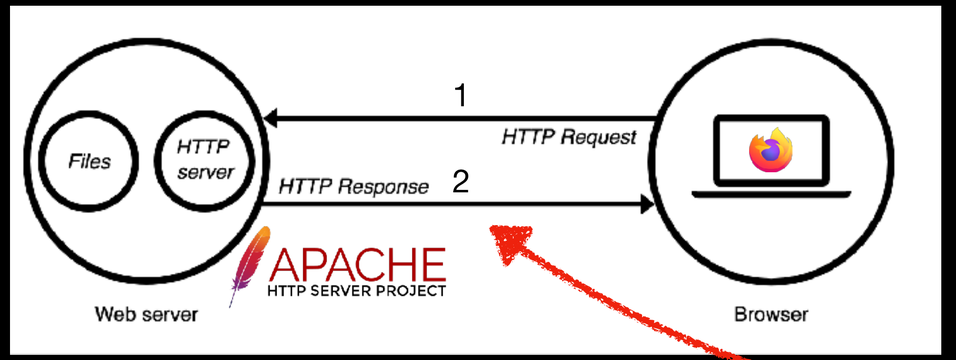
JS包名称	用途
JQuery	快速方便检索DOM，信息传输（JS <—> HTML）(Serialization, JSON)
BootStrap	提供好看的UI组件，方便适配不同屏幕
d3.js	JS可视化鼻祖
…	还有上千种包，最活跃的开源社区（动画，渲染，机器学习，地图，等等） • 博客“2021 40大包”：https://kinsta.com/blog/javascript-libraries/
JS前端“三大架构”（Framework）（相互竞争，不同互联网公司采用不同的架构）	
React	• 制作复杂的UI • 简化编程，区分数据模型(data model)与视图模型(view model) • 降低延迟，尽量在前端解决问题，减少与服务器沟通次数
Vue	
Angular	
JS后端拓展	
Node.js	• 使用JavaScript作为语言，在本地电脑上使用（而非浏览器），完成服务器编程 • 即：掌握JavaScript一种可以完成所有的网站编写
Node.js==》npm	• 包管理器，管理后端/本地安装的包（类似Python中的pip，conda） • 现在也用来管理前端包（博客教程：https://www.impressivewebs.com/npm-for-beginners-a-guide-for-front-end-developers/）

端对端协议：HTTP/HTTPS

Communication between Front-end and Back-end

- HTTP (HyperText Transfer)
- HTTP Request Methods
  - GET：用户申请浏览内容（例如：使用搜索引擎）
    - 直接在网址URL(uniform resource locator) 后面填写变量
  - POST：用户提交内容改变服务端数据（例子：修改密码）(create new)
    - POST变量单独存储，并有加密措施。
  - 其他方法: PUT, HEAD, DELETE, PATCH

流程永远是：客户端先请求—> 服务器再回应

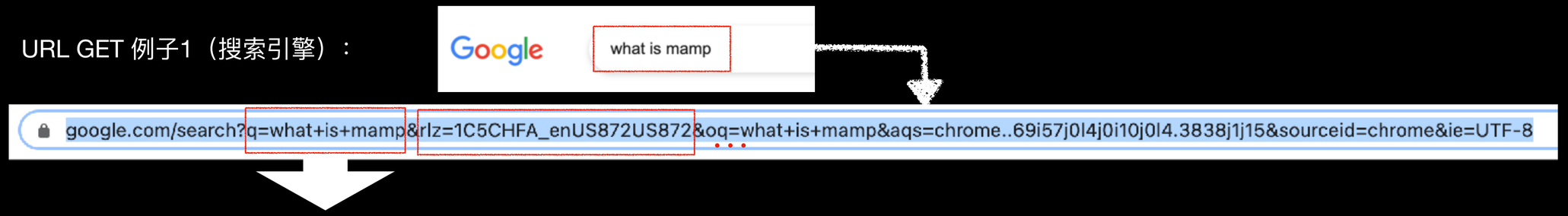


服务器返回值

	含义
2xx	正常
3xx	Redirect重新导航
4xx	客户端错误
5xx	服务器错误

Python中的POST请求例子

URL GET 例子1（搜索引擎）：



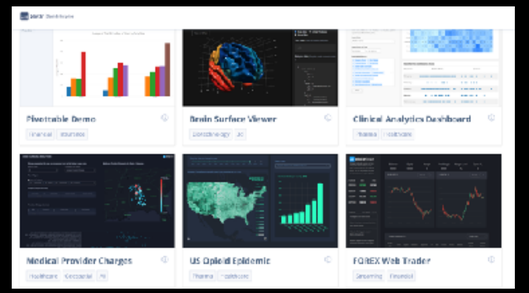
网址?变量名1=值1&变量名2=值2&变量名3=值3…

例子2: 应用URL作为网站的接口 API（Application Programming Interface）

- 商用API接口（例子：地图导航/天气服务等）
- 一些有趣的API（参考：https://betterprogramming.pub/a-curated-list-of-100-cool-and-fun-public-apis-to-inspire-your-next-project-7600ce3e9b3）
  - 疫情数据：https://covid19api.com/
  - 宝可梦数据库：https://pokeapi.co/api/v2/pokemon/charizard
  - AI图像识别API：https://docs.clarifai.com/api-guide/api-overview
  - 股票数据：https://polygon.io/stocks?gclid=EAIaIQobChMIto7e_Z6I8QIV221vBB3Yuwa1EAAYASAAEgK9t_D_BwE

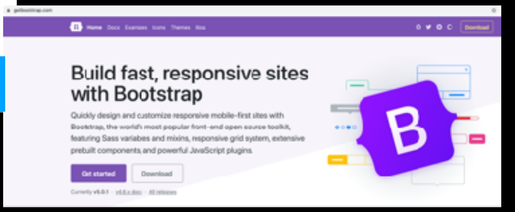
实践与发展趋势

- 开发网站后端的成本越来越低：
  - Python后端快速编写程序（也会被大公司采用）：Flask，Django
    - Flask: Reddit, Airbnb, Uber(Flask, Node, Go, https://eng.uber.com/uber-tech-stack-part-two/), Lyft, etc.
    - Django: Spotify, Dropbox, Pinterest, etc. (https://www.geeksforgeeks.org/top-10-django-apps-and-why-companies-are-using-it/)
  - JavaScript后端 (Node.js): https://trio.dev/blog/companies-use-node-js
- 可快速开发部署的应用网站（Apps）（Python/R/Julia）
  - 项目减少代码量，快速搭建简易网站（仍需要服务器）
  - Streamlit: 类似于Jupyter notebook，但是是网页后端，很快实现网页（https://streamlit.io/gallery）
  - Dash: 配合Plotly（https://dash-gallery.plotly.host/Portal/）
  - R Shiny
- 全前端0成本开发部署（个人网页，项目展示等）
  - GitHub Pages项目https://pages.github.com/
- 云服务
  - 客户租用硬件设备，服务商提供相关的各种服务方便客户（例：亚马逊AWS）成本降低，资源可拓展，不用自己维护硬件



一些数据App网站实例

整个网页写在GitHub项目里



参考资料
References

后端

- MAMP: https://en.wikipedia.org/wiki/MAMP
- Front-end vs. Back-end(技术栈对比): https://fullscale.io/blog/top-5-tech-stacks/#:~:text=The%20frontend%20tech%20stack%20is,interface%2C%20and%20straightforward%20internal%20structures.
- HTTP server: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server
- Web server vs. Application server: https://stackoverflow.com/questions/936197/what-is-the-difference-between-application-server-and-web-server
- Node.js解释是如何作为后端应用的：https://www.youtube.com/watch?v=YSyFSnisip0

前端

- Vue JS 3介绍: https://www.youtube.com/watch?v=YrxBCBibVo0
- JS包/架构介绍：https://newrelic.com/blog/best-practices/best-javascript-libraries-frameworks
- BootStrap介绍：https://getbootstrap.com/docs/4.0/getting-started/introduction/
- NPM包：https://www.npmjs.com/
- 使用NPM管理前端包：https://www.impressivewebs.com/npm-for-beginners-a-guide-for-front-end-developers/

前后端数据传输，HTTP/HTTPs

- https://developer.mozilla.org/en-US/docs/Learn/Forms/Sending_and_retrieving_form_data
- POST vs. GET methods: https://techdifferences.com/difference-between-get-and-post-method-in-html.html
- POST vs. PUT vs. PATCH methods: https://stackoverflow.com/questions/107390/whats-the-difference-between-a-post-and-a-put-http-request#:~:text=A%20POST%20request%20creates%20a,that%20URI%20to%20the%20client.&text=A%20POST%20request%20can%20also,or%20updates%20an%20existing%20resource.
- POST-Redirect-Get Pattern: https://en.wikipedia.org/wiki/Post/Redirect/Get
- API: https://en.wikipedia.org/wiki/Query_string
- Restful API: https://www.redhat.com/en/topics/api/what-is-a-rest-api