<WA/>
2025

Web Architecture

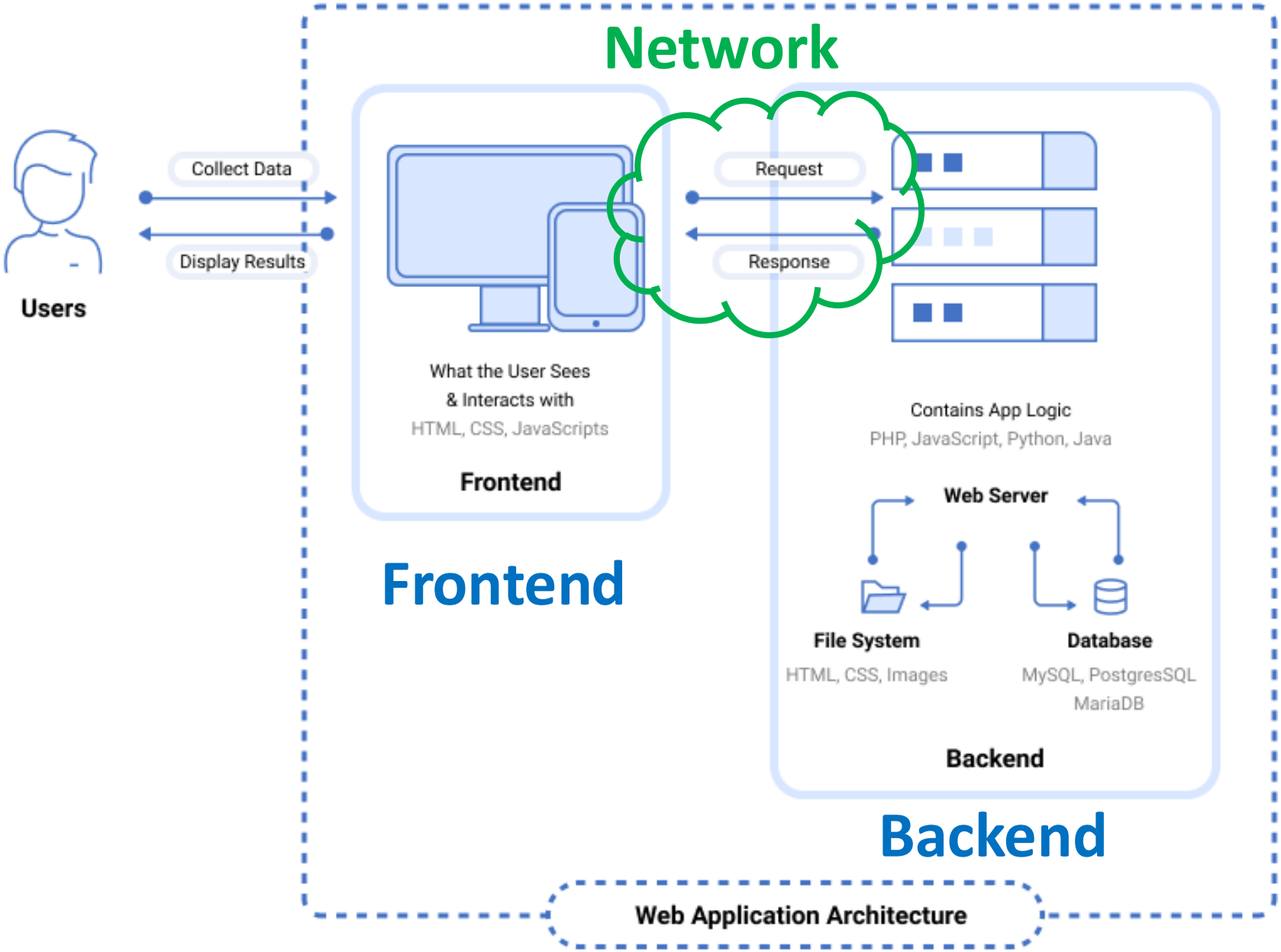
Layers, Languages, Protocols

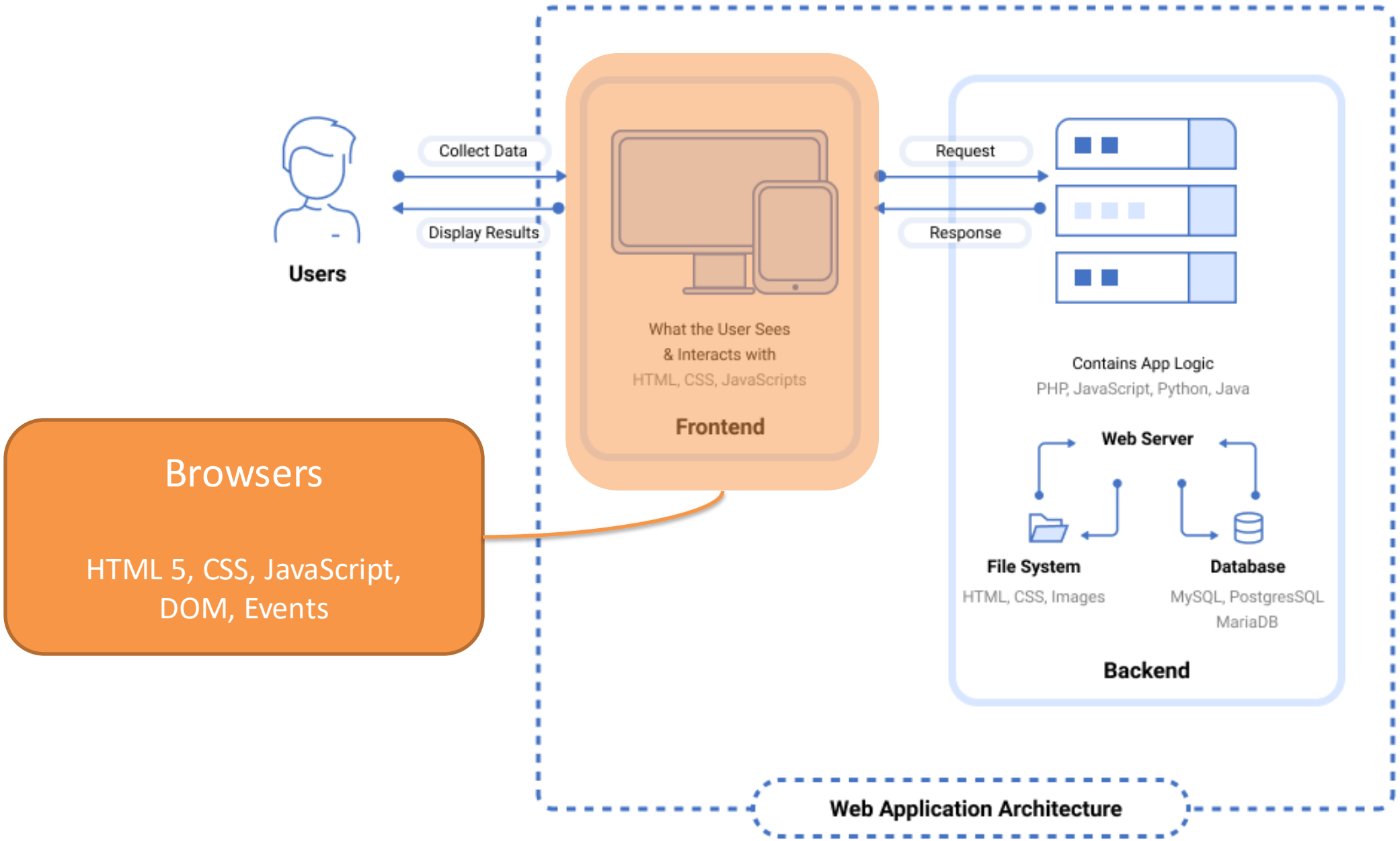
Fulvio Corno
Luigi De Russis
Enrico Masala

Goal

- Understand what is the Web and its architecture
  - main (logical) components
  - main network protocols
  - existing architectural patterns and languages
- Know the interaction and communication across components

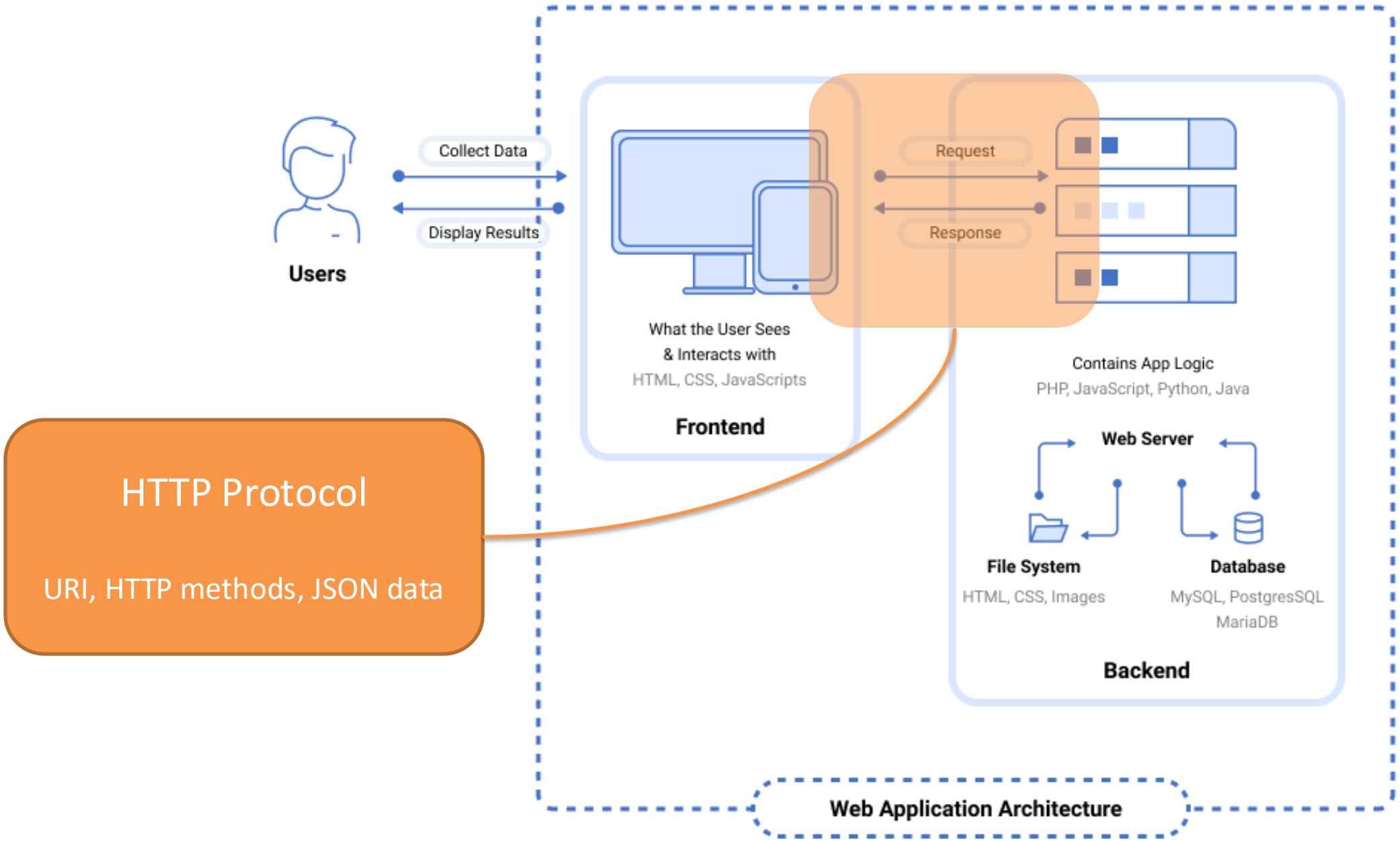
- *NOTE: All the topics mentioned here will be presented in more details in the next lectures*


Network
Collect Data
Display Results
Users
Request
Response
What the User Sees
& Interacts with
HTML, CSS, JavaScripts
Frontend
Frontend
Contains App Logic
PHP, JavaScript, Python, Java
Web Server
File System
HTML, CSS, Images
Database
MySQL, PostgresSQL
MariaDB
Backend
Backend
Web Application Architecture



Browsers

HTML 5, CSS, JavaScript,
DOM, Events



HTTP Protocol

URI, HTTP methods, JSON data

HTTP protocol

RFC 2616, RFC 2617
http://www.w3.org/Protocols

```
GET / HTTP/1.1
Host: www.polito.it
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
DNT: 1
Connection: keep-alive
Cookie: __utma=55042356.701936439.1606736391.1615238467.1615289682.230; __utmz=55042356. [...]
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
```

HTTP Request

HTTP protocol

RFC 2616, RFC 2617
http://www.w3.org/Protocols

```
GET / HTTP/1.1
Host: www.polito.it
User-Agent: Mozilla/5.0 (Windo
Accept: text/html,application/
Accept-Language: en-US,en;q=0
Accept-Encoding: gzip, defla
DNT: 1
Connection: keep-alive
Cookie: __utma=55042356.7
Upgrade-Insecure-Request
Pragma: no-cache
Cache-Control: no-cache
```

HTTP Response

```
HTTP/1.1 200 OK
Date: Tue, 09 Mar 2021 14:21:35 GMT
Server: Apache
Strict-Transport-Security: max-age=31536000
Content-Security-Policy: script-src 'self' 'unsafe-inline' 'unsafe-eval' [...]
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Referrer-Policy: no-referrer-when-downgrade
Feature-Policy: accelerometer 'none'; camera 'none'; geolocation 'none'; [...]
Last-Modified: Tue, 09 Mar 2021 14:03:41 GMT
Cache-Control: no-cache, must-revalidate
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 11905
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<!doctype html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="it">
<head>
          <meta charset="UTF-8">
          <title>Politecnico di Torino</title>
. . .
```

Header

Blank line

Body

HTTP Response Body

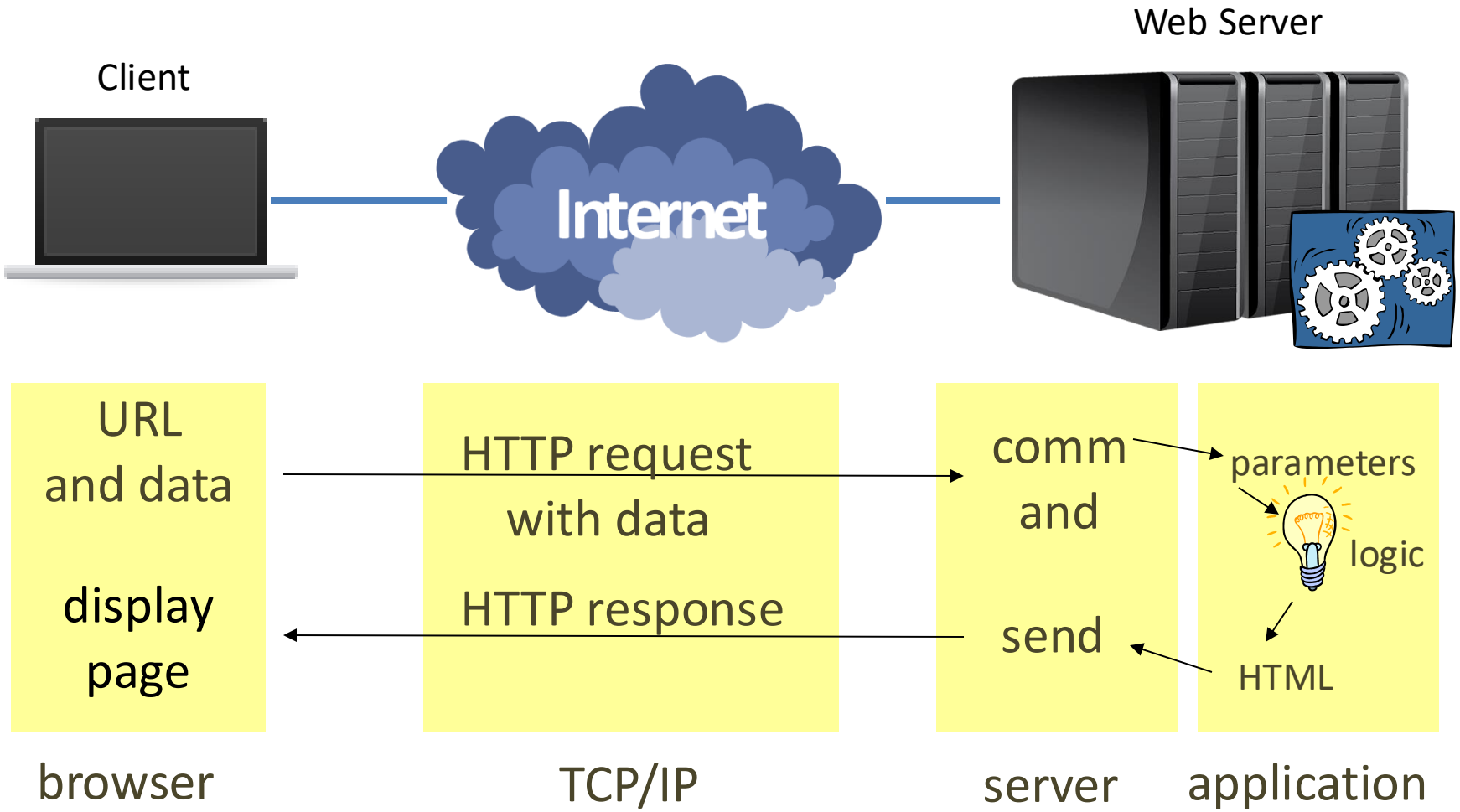
Generation

- Empty Response Body
  - When returning errors
- Static content (exists in the server)
  - HTML (seldom)
  - Images, JavaScript, CSS, …
- Dynamically generated on-the-fly by the server
  - HTML (generated with templates)
  - JSON data

File and Content Type

- HTTP does not care about the meaning of the payload
- Web content
  - HTML, CSS, JS
  - Used by the browser
- Data content (API)
  - JSON, XML, binary data, …
  - Used by JavaScript code

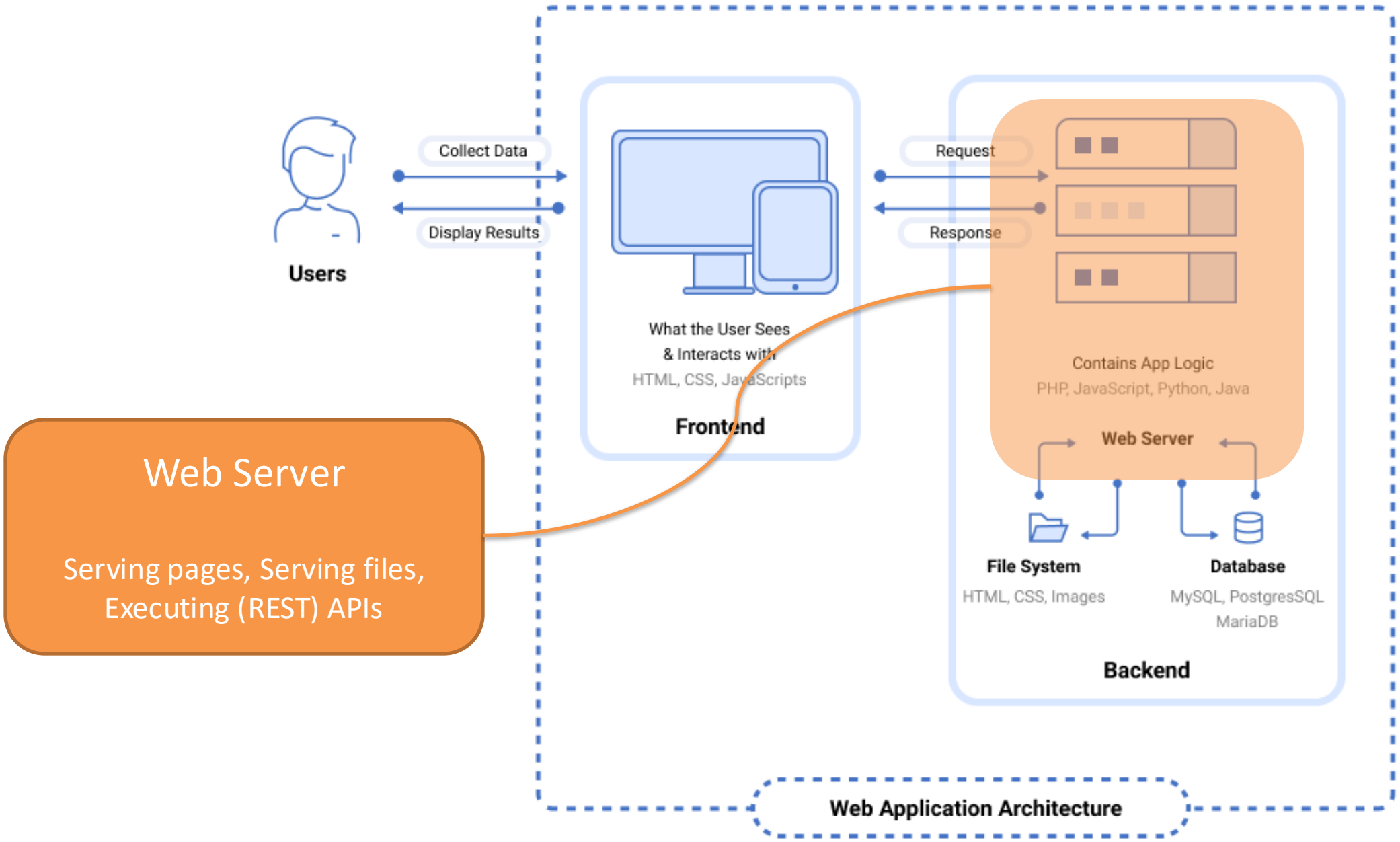
Dynamic Web Transaction



HTTP Methods

HTTP method	RFC	Request has Body	Response has Body	Safe	Idempotent	Cacheable
GET	RFC 7231	Optional	Yes	Yes	Yes	Yes
HEAD	RFC 7231	Optional	No	Yes	Yes	Yes
POST	RFC 7231	Yes	Yes	No	No	Yes
PUT	RFC 7231	Yes	Yes	No	Yes	No
DELETE	RFC 7231	Optional	Yes	No	Yes	No
CONNECT	RFC 7231	Optional	Yes	No	No	No
OPTIONS	RFC 7231	Optional	Yes	Yes	Yes	No
TRACE	RFC 7231	No	Yes	Yes	Yes	No
PATCH	RFC 5789	Yes	Yes	No	No	No

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods



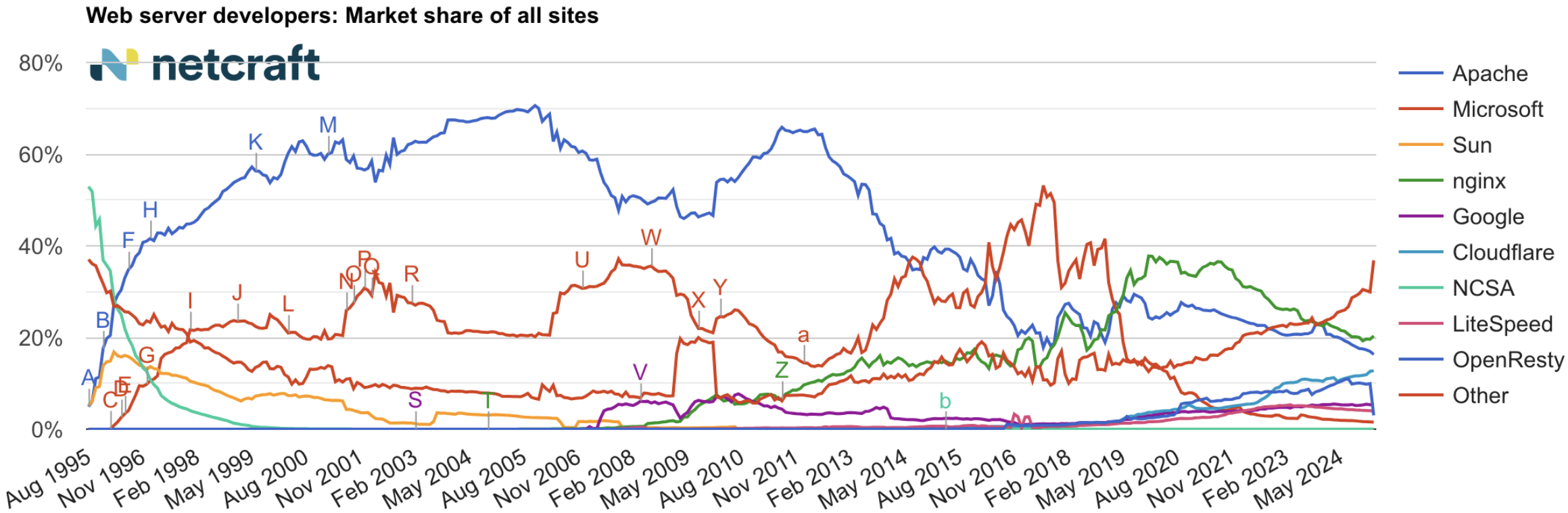
Web Server

Serving pages, Serving files,
Executing (REST) APIs

Web Server

- A web server delivers web resources in response to a request
  - manages the HTTP protocol to handle requests and provide responses
- It either reads or generates a web page
  - receives client requests
  - reads *static pages* from the filesystem
  - asks the application server to generate *dynamic pages* (server-side)
  - provides a file (HTML, CSS, JS, JSON, …) back to the client
- One HTTP connection for each request
- Multi-process, multi-threaded or process pool

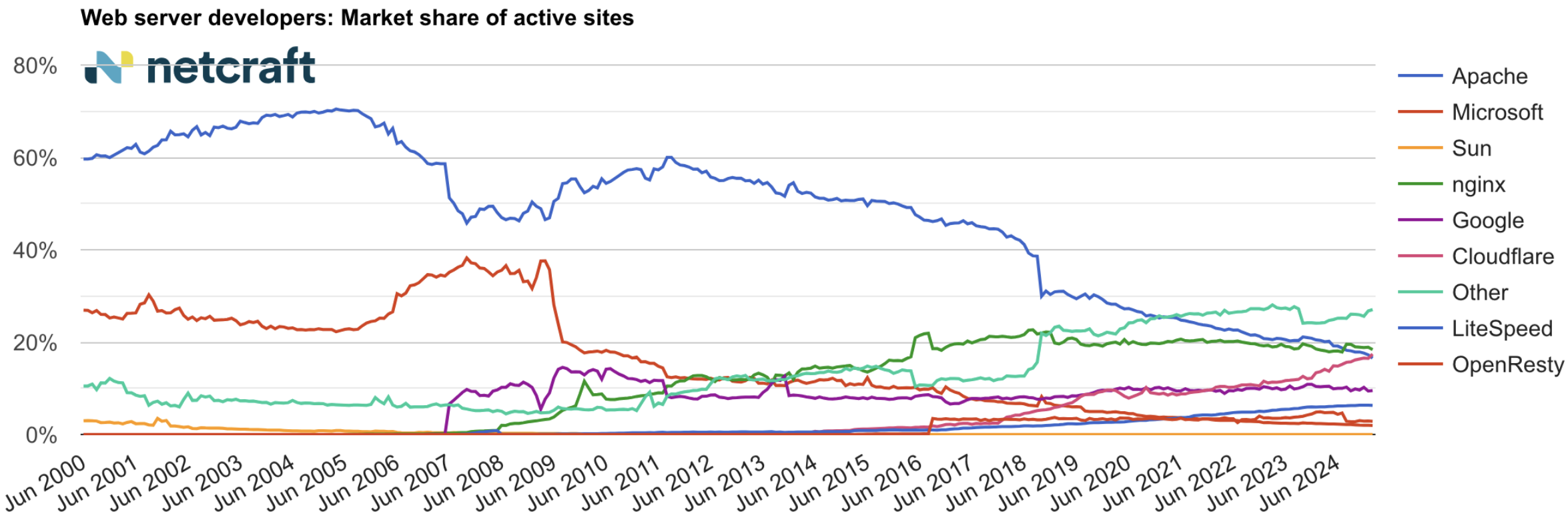
Web Server



Source: http://news.netcraft.com/

https://www.netcraft.com/blog/february-2025-web-server-survey/

Web Server



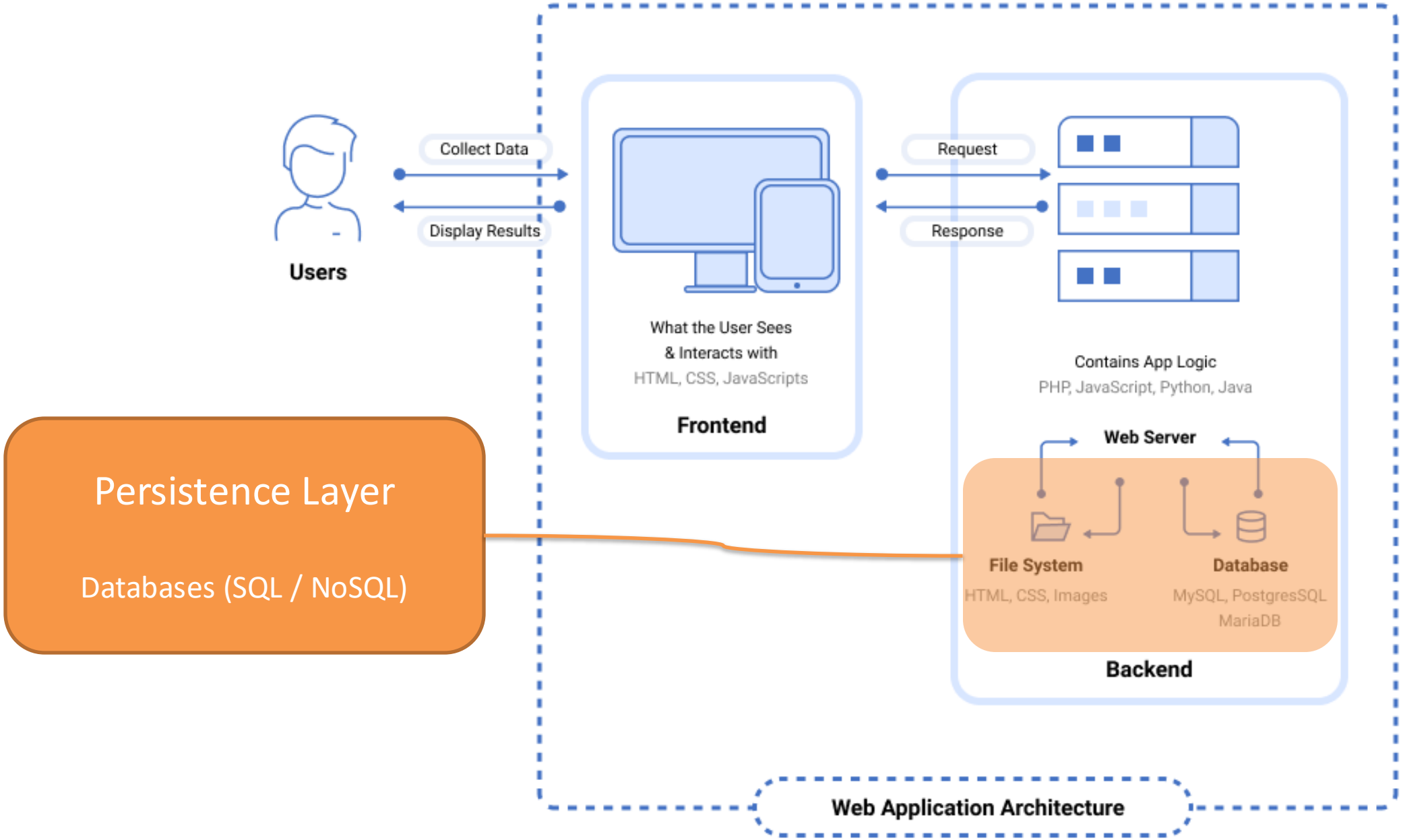
Source: http://news.netcraft.com/

https://www.netcraft.com/blog/february-2025-web-server-survey/

Web Server

- We adopt a web server running in the Node.js environment because:
  - No need to learn a new language to write code on the server side
  - JavaScript is very well suited for asynchronous operations
  - Very good packages exist for this purpose
- `Express`: a simple and extensible web server, easy to extend with many available extensions - http://expressjs.com/

Very important: this is just a choice for our course, many other technologies, frameworks and languages available to develop the backend parts


Users
Collect Data
Display Results
What the User Sees
& Interacts with
HTML, CSS, JavaScripts
Frontend
Request
Response
Contains App Logic
PHP, JavaScript, Python, Java
Web Server
File System
HTML, CSS, Images
Database
MySQL, PostgresSQL
MariaDB
Backend
Persistence Layer
Databases (SQL / NoSQL)
Web Application Architecture

Web Architecture

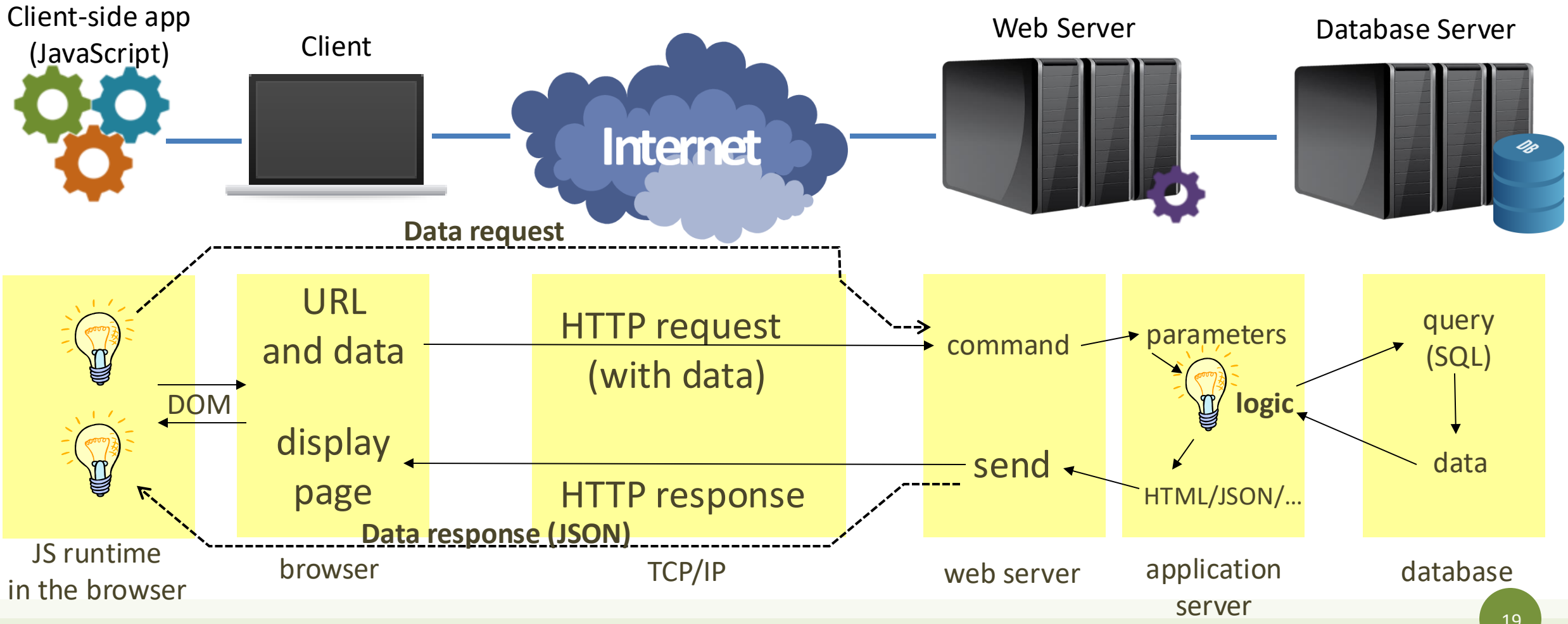
ARCHITECTURAL PATTERNS

"Traditional" Architectural Pattern

- The server sends a new content (HTML) page for each request it receives
  - with related resources (i.e., images, CSS, …)
  - some parts of those pages can then be dynamically updated with asynchronous JavaScript requests (so-called "Rich-Client")

- A web application is doing **server-side rendering**, and a ***multi-page*** web application is created

All The Layers At Work…



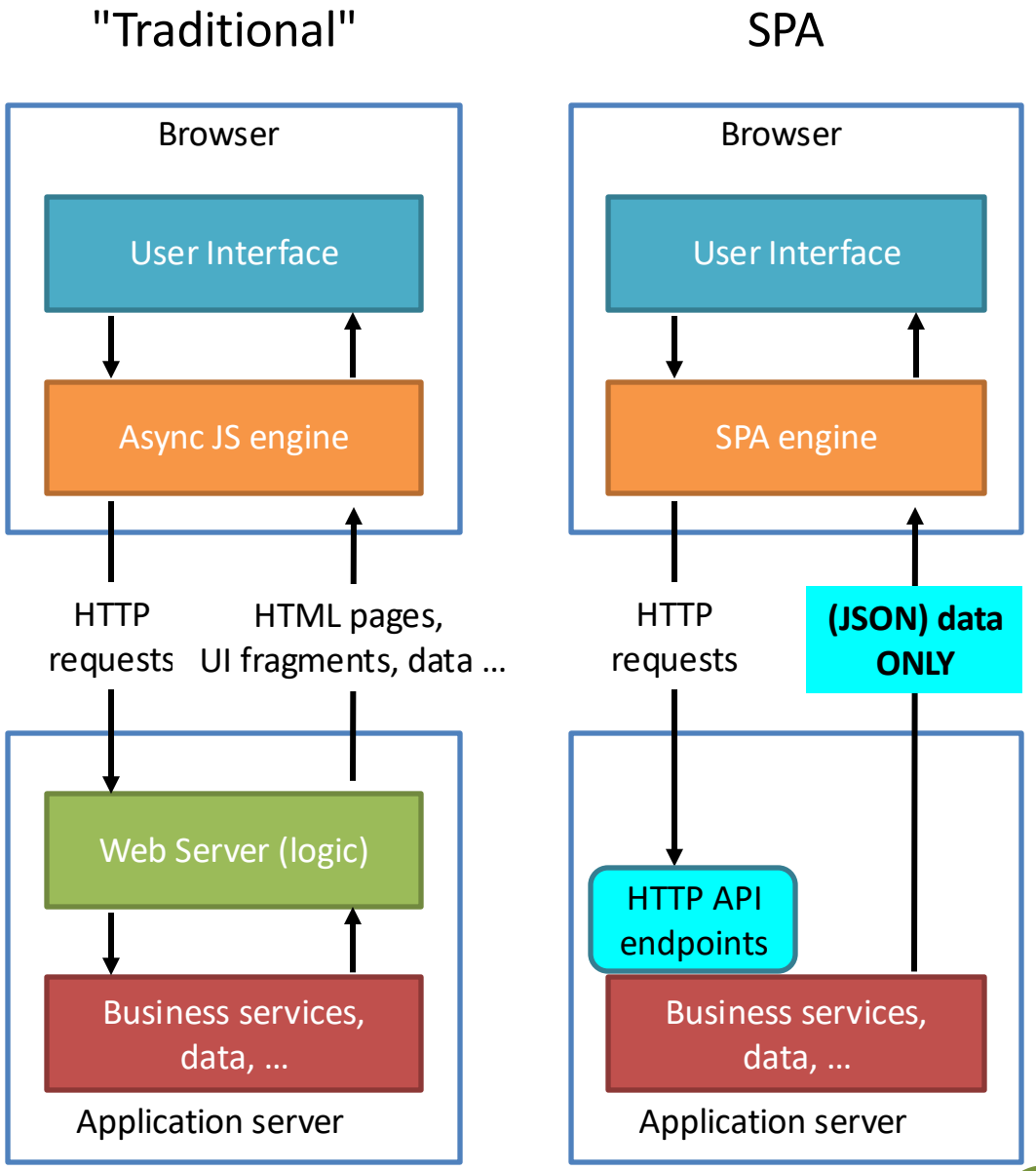
Modern Patterns

Other three patterns to architect a web application exist, roughly

1. **Single-Page Application** (SPA)
   - the server sends the **exact same** web page for every unique URL
   - the page runs JavaScript to change the content and the aspect
   - by querying another (logical) server which provides “raw” information

Single-Page Application

- An *evolution* of the "traditional" approach
  - JavaScript starts with an (almost empty) HTML
  - add all the content dynamically
  - instead of asking for data to update some parts of an already well-formed page
- Goal: to serve an outstanding User Experience with **no page reloading** and **no extra time waiting**
- Examples: Google Docs, Trello, …



SPA: Disadvantages

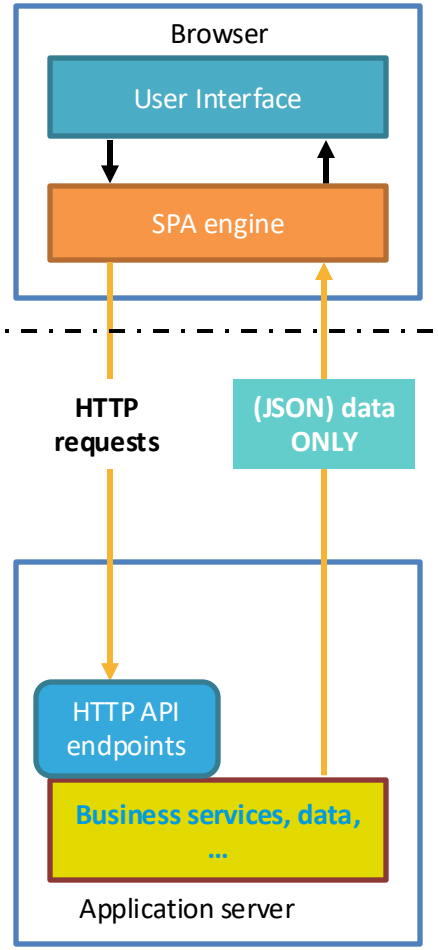
- Search Engine Optimization (SEO) is hard
  - Google launched a new scheme to increase single-page app SEO optimization, but this means extra work for the developer
- Browser history is not working
  - Web History API exists to tackle this problem and to allow a developer to emulate the back-and-forth action
- Security issues
  - Given that "all the logic is in the client", **special care should be taken when handling access control**. Cross-Site Scripting (XSS) is a problem as well.
- Client-side rendering can be slow!

Modern Patterns

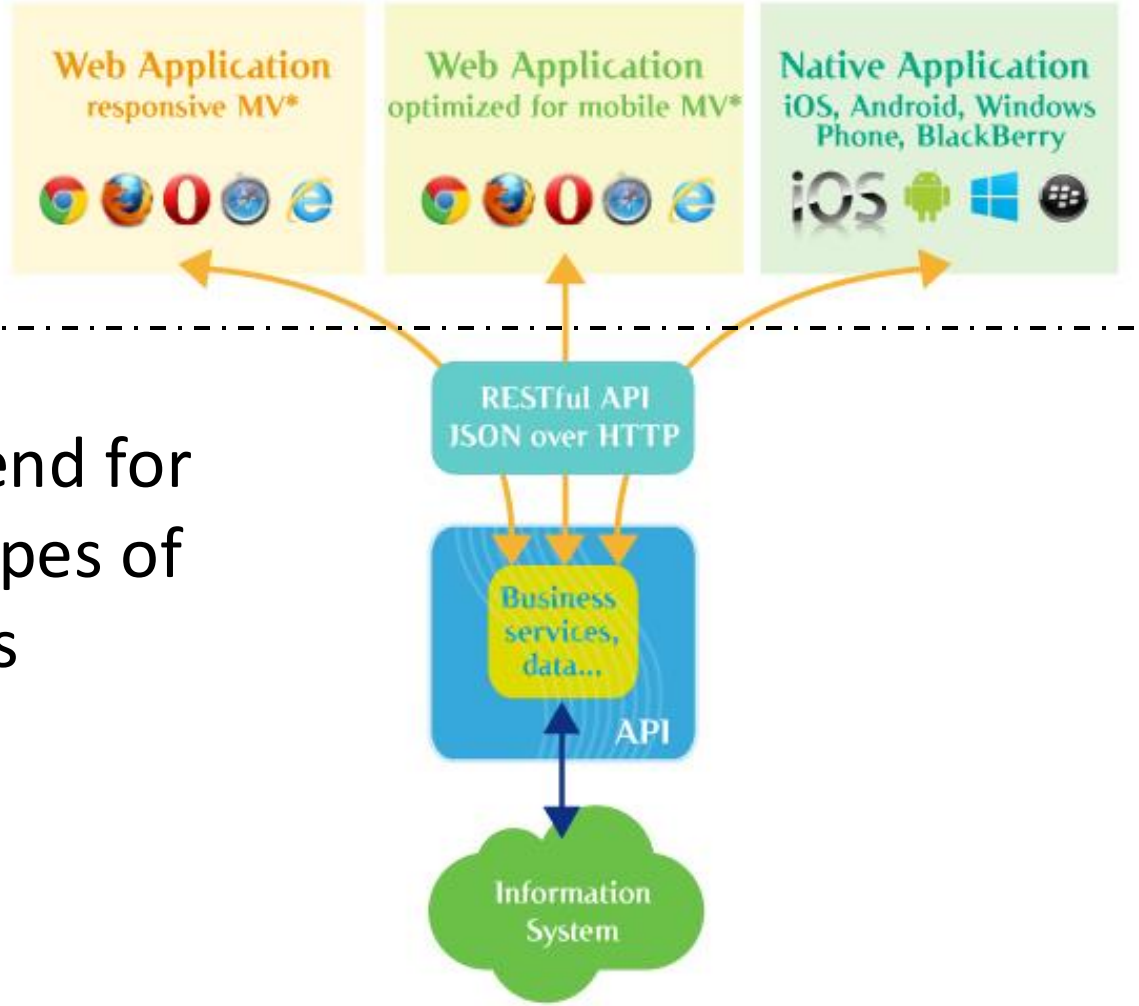
Other three patterns to architect a web application exist, roughly

1. **Single-Page Application** (SPA)
   - the server sends the exact same web page for every unique URL
   - the page runs JavaScript to change the content and the aspect
   - by querying another (logical) server which provides "raw" information
2. **Isomorphic Application**
   - Combination of SPA with server-side rendering
3. **Progressive Web App (PWA)**
   - Web applications that emulate "native" apps

Supporting Mobile Development



Same backend for different types of clients



https://blog.octo.com/new-web-application-architectures-and-impacts-for-enterprises-2

Client-side, server-side, databases

Websites	Popularity (unique visitors per month)[1]	Front-end (Client-side)	Back-end (Server-side)	Database
Google[2]	2,800,000,000	JavaScript, TypeScript	C, C++, Go,[3] Java, Python, Node	Bigtable,[4] MariaDB[5]
Facebook	1,120,000,000	JavaScript, Typescript, Flow	Hack/HHVM, Python, C++, Java, Erlang, D,[6] Haskell[7]	MariaDB, MySQL,[8] HBase, Cassandra[9]
YouTube	1,100,000,000	JavaScript, TypeScript	Python, C, C++, Java,[10] Go[11]	Vitess, BigTable, MariaDB[5]
Yahoo	750,000,000	JavaScript	PHP	PostgreSQL, HBase, Cassandra, MongoDB,[12]
Etsy	516,000,000 (Total, not unique)[13]	JavaScript	PHP[14][15]	MySQL, Redis[16]
Amazon	2,400,000,000[17]	JavaScript	Java, C++, Perl[18]	DynamoDB, RDS/ Aurora, Redshift[19]
Wikipedia	475,000,000	JavaScript	PHP	MariaDB[20]
Fandom	315,000,000[21]	JavaScript	PHP	MySQL

JavaScript everywhere

https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites

References

- Overview of Web Applications: https://www.robinwieruch.de/web-applications/
- How the Web works: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works

- Summary about HTTP key aspects: https://code.tutsplus.com/http-the-protocol-every-web-developer-must-know-part-1--net-31177t
- HTTP/1.x vs. HTTP/2 – The Difference Between the Two Protocols Explained https://cheapsslsecurity.com/p/http2-vs-http1/

License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/