Supplying Data

Web Dev, Spring 2021

Last time

We introduced fetch as a way to pull data from a web server via JavaScript

Web servers are just glorified document providers

- HTML documents are special kind of documents that
  web browsers just know how to display nicely

(We'll refine this picture of web servers later)

Introduction to web servers

A web server listens to requests on a port

- every network-aware app listens on a port
- Web servers listen on port 80 (HTTP) or 443 (HTTPS) by default

Browsers assume those ports when not supplied explicitly

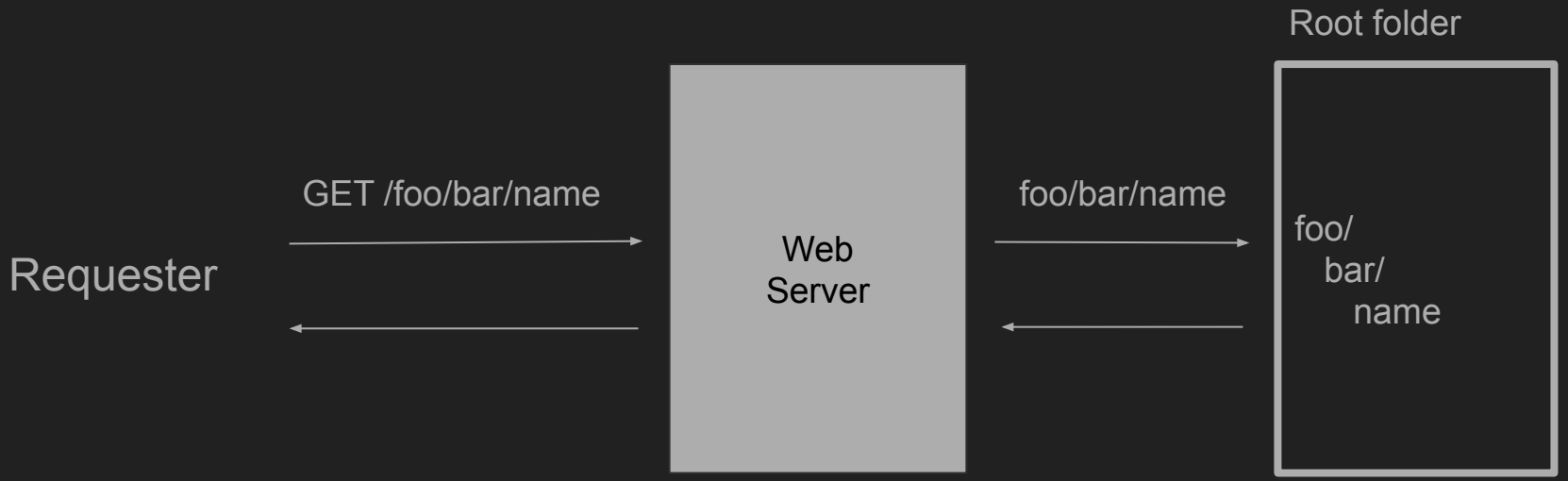
URL: <protocol>://<hostname>:<port>/<route>

- Protocol usually http or https
- hostname is the name of the machine running the server
- port is the port (can be dropped)
- route is the "location" of the file on the server

Classic Web Server

Respond to HTTP GET requests on a route:

- treat route as a path on the filesystem

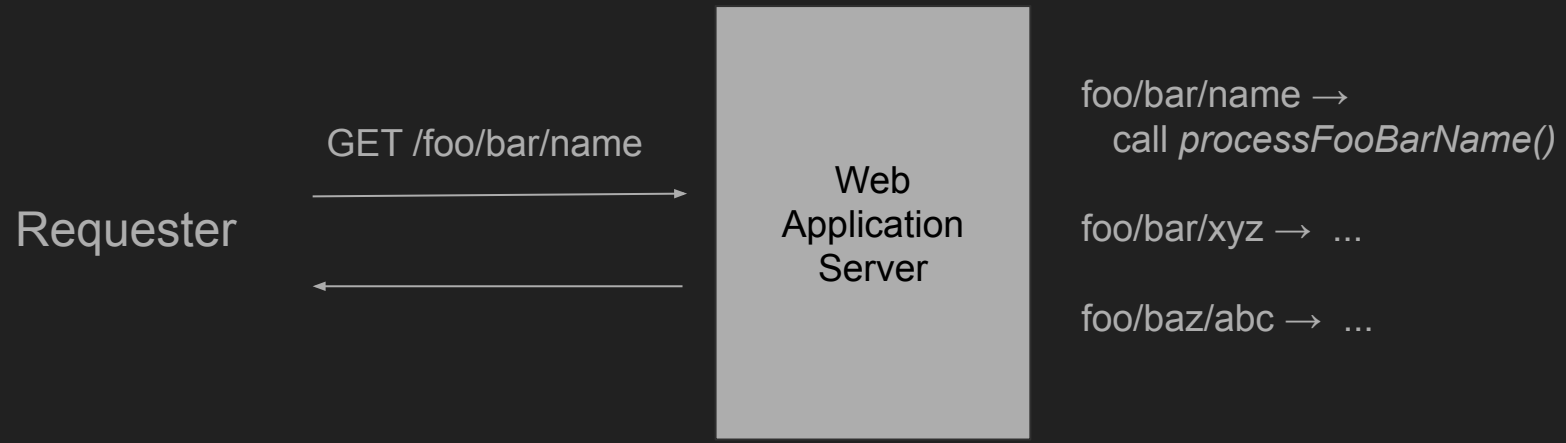


Demo — classic web server

Web Application Server

Respond to HTTP GET requests on a route:

- execute some code associated with the route that creates a response



Flask

How do you program Web Application Servers?

- Any programming language with a networking / HTTP library can be used
- Web Frameworks are libraries dedicated to creating Web Application Servers
- Different levels of scalability and "provide-X-out-of-the-box"
- Java → Spring
  JS → Express
  Python → Django, Flask, …
  PHP → Symfony

We're going to use Flask because it's lightweight and doesn't hide too much

Flask

Tasks:

- create the server and run it on a port
- associate functions with routes

```
import flask

app = flask.Flask(__name__)
app.run(port=8080)
```

```
@app.route('/foo/bar/name')
def processFooBarName():
    return '<html>...</html>'
```

Demo — Flask