This work is licensed under CC BY 4.0

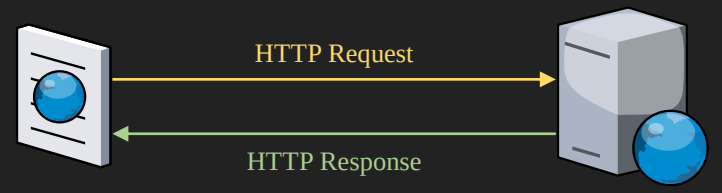
Express – A Node.js Web Application Framework

Web Application Architecture



Why You Need a Web Application Architecture

- ✓ As applications grow, you need structure.
- ✓ Architectures help to provide the application structure, both physical and logical.
- ✓ At least two parts, the client and the server.
  - The code that is on the web browser and responds to user actions.
  - The code that is on the server and responds to HTTP requests.
- ✓ It is up to the developer to decide what the code on the client and the server should do.
  - Client-side code
    - HTML, CSS, JavaScript
  - Server-side code
    - Ruby, C#, Python, PHP, Java, JavaScript
    - Never seen by the client, persistent data, creates dynamic content, etc.



Minimal “hello, world”

```
import http from 'node:http'

const server = http.createServer((req, res) => {
  res.setHeader('Content-Type', 'text/plain')
  res.end('hello, world\n')
})

server.listen(8080, () =>
  console.log('Server running at http://localhost:8080/')
)
```

Handles HTTP requests.

- ✓ The build in HTTP module http allows Node.js to transfer data over HTTP.
  - Express has only build-in support for HTTP/1.1.
- ✓ Event-driven programing.
  - The event being handled is an HTTP request.
- ✓ The `createServer` method takes a function as an argument; that function will be invoked every time an HTTP request is made.
  - Creates an HTTP server that listens to a server port and gives a response back to the client.

Minimal Web Application with Simple Routing

```
import http from 'node:http'

const server = http.createServer((req, res) => {
  const path = req.url.replace(/\/?(?:\?.*)?$/, '').toLowerCase()

  res.setHeader('Content-Type', 'text/plain')

  switch (path) {
    case '':
      res.end(‘Start Page')
      break

    case '/about':
      res.end('About')
      break

    default:
      res.statusCode = 404
      res.end('Not Found')
      break
  }
})

server.listen(8080, () =>
  console.log('Server running at http://localhost:8080/')
)
```

- ✓ Routing is the mechanism for serving the client with the requested content.
  - It is possible to browse to `http://localhost:8080` and `http://localhost:8080/about` (query strings will be ignored).
  - Any other URL will get a response with status code 404.
- ✓ The complexity increases rapidly with the number of routes!
  - It would help if you had something that makes it somewhat easier.



Some JavaScript Back-end Frameworks

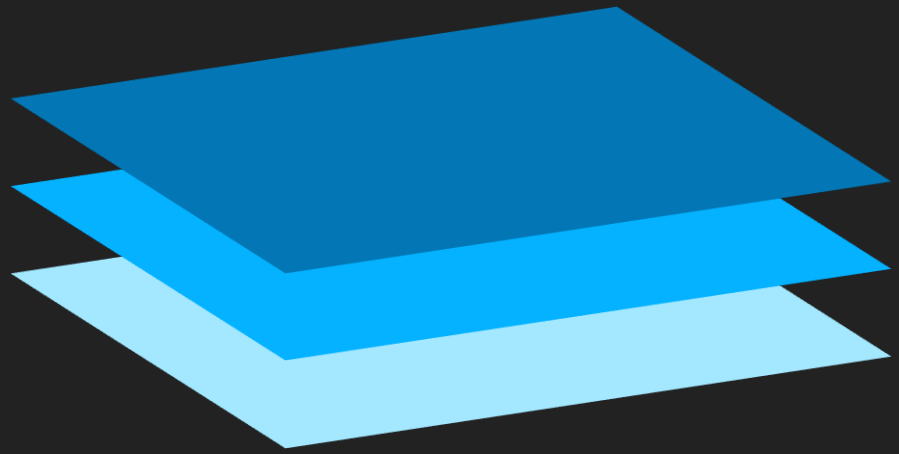
- ✓ Established options

npm (2023-12-11), https://www.npmjs.com/

Framework	Weekly Downloads	Version	Last Published
Express	27,546,008	4.18.2	2022-10-08
Koa	1,456,660	2.14.2	2023-04-12
Fastify	1,407,524	4.23.3	2023-10-19
Hapi	624,782	21.3.2	2023-04-25
SvelteKit (Svelte)	206,983	1.28.0	2023-12-11
Next.js (CLI tool, React)	96,058	14.0.4	2023-12-09
FeathersJS	58,181	5.0.12	2023-11-28
Sails	25,532	1.5.8	2023-09-01

Full Stack Models

- A software stack is a set of software subsystems or components needed to create a complete platform where an application can be run on.
  - A web application depends on multiple pieces of technology such as web server, database server and framework.
- Acronyms describe "stacks", development environments.
  - LAMP (Linux, Apache, MySQL, PHP)
  - WAMP (Windows, Apache, MySQL, PHP)
  - …
- Full stack JavaScript Development
  - Both client- and server-side code should be written in JavaScript.
    - MERN (MongoDB, Express, React, Node.js)
    - MEAN (MongoDB, Express, Angular, Node.js)
    - JAMStack (JavaScript, API:s, Markup (HTML and CSS))
    - …



Full Stack JavaScript Development

- ✓ During the course you will use...
  - Front-end
    - Non-framework, HTML, CSS and vanilla JavaScript.
  - Back-end
    - Node.js + Express, with EJS as the view engine.
  - Database system
    - MongoDB as document database.



N

Node.js
Back-end runtime environment



E

Express
back-end web framework



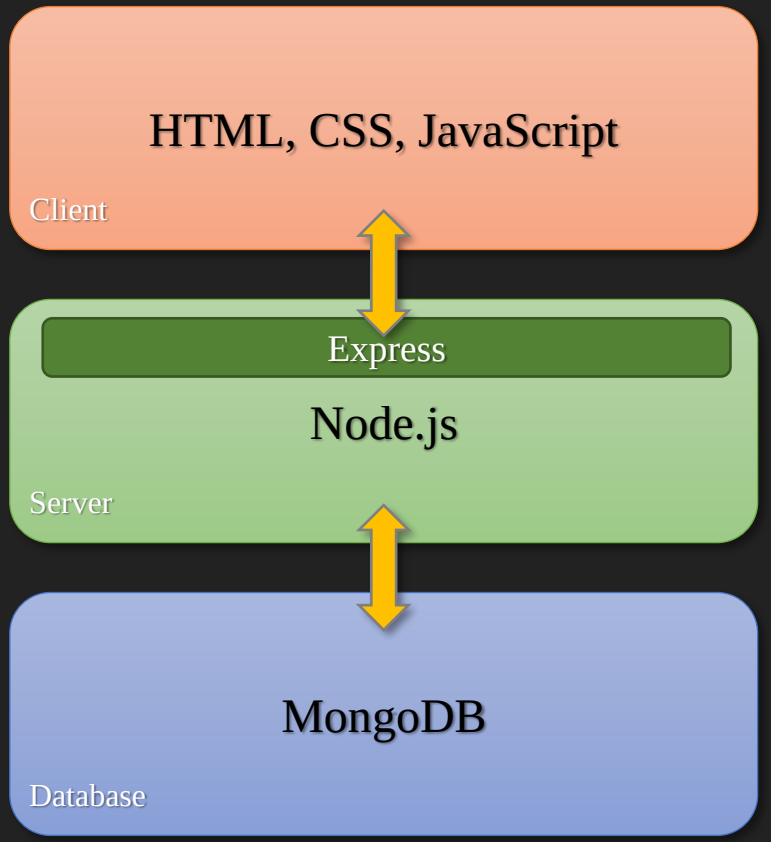
J

JavaScript
front-end non-framework (pure vanilla JavaScript)



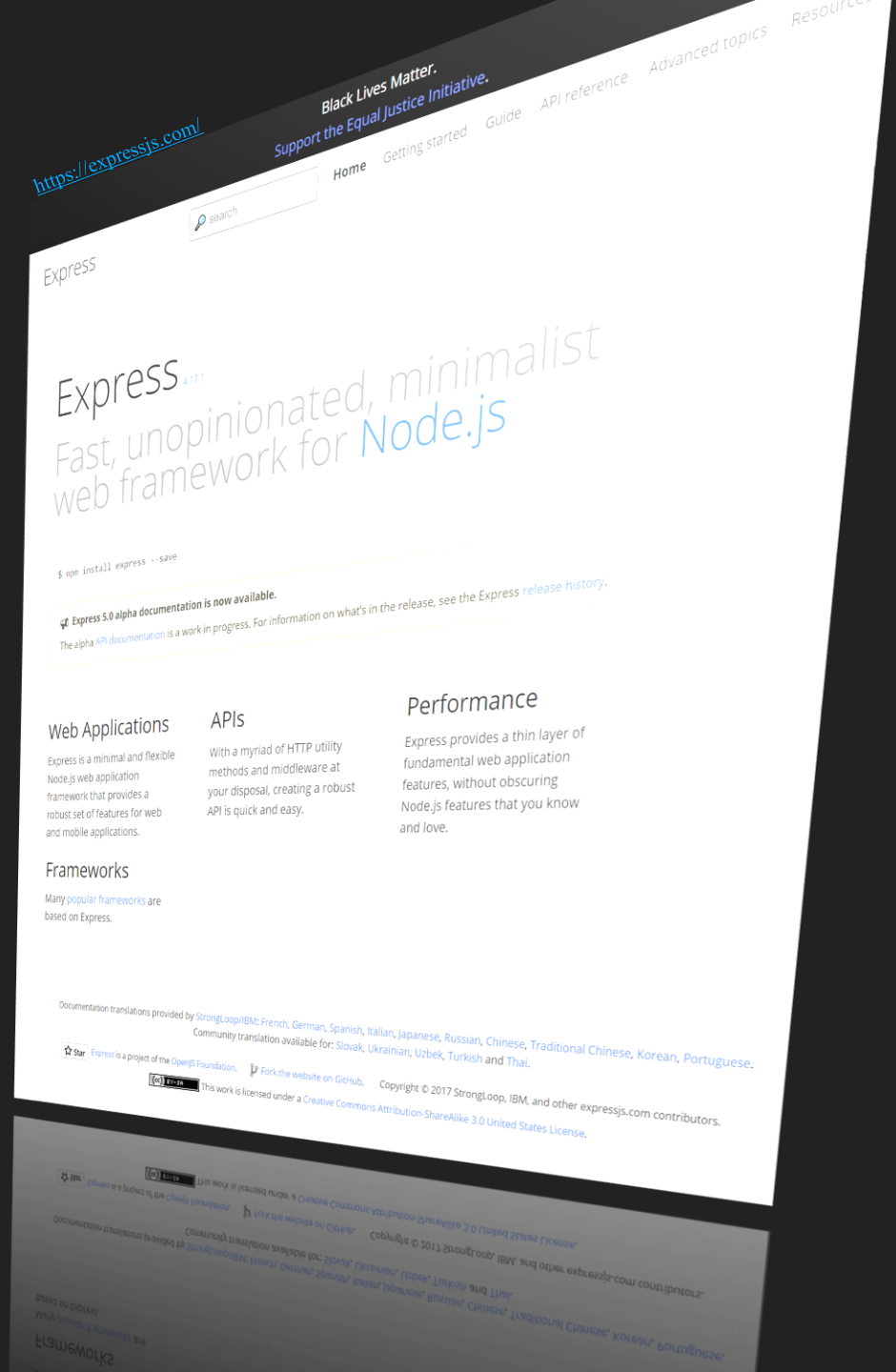
M

MongoDB
database system



What is Express?

- ✓ "A minimal and flexible Node.js web application framework."
  - Node.js has a lot of common with other web servers, but most interesting is how it differs.
    - Easy to setup and configure.
    - Node.js is single threaded.
    - Platform independent.
- ✓ Inspired by Sinatra, a web application framework in Ruby.
- ✓ Intertwined with Connect, a pluggable Node module that can handle web requests (a.k.a. "middleware").



Minimal Express Web Application

```
import express from 'express'

const app = express()

app.get('/', (req, res) => res.send('Start Page'))
app.get('/about', (req, res) => res.send('About'))

app.listen(8080, () =>
  console.log('Server running at http://localhost:8080/')
)
```

- ✓ The application starts a server and listens on port 8080 for connections.
- ✓ The framework will handle the request.
  - The application responds with “Start Page" for requests to the root URL (`/`).
  - The application responds with "About" for requests to the about URL (`/about`).
  - For any other request, it will respond with a 404 Not Found.

