Client-Server Architecture: Fullstack Applications

SERVER ("BACK END")

Machine 1 (computer + operating system)

Filesystem (serial data)

static files
(images, .html docs,
.js files, etc.)

10111000111010110101
01001011010010100101
10101010101101010101
0110100101011

raw database data
(e.g. BSON, .sql)

10111000111010110101
010010110100

FS I/O

FS I/O

DBMS:
mongod / **postgres**
process (C/C++)

translate query into binary
read/write and send back
result to requester

Db Protocol   **Db Protocol**

mongo / **psql** process

```
> shell
  (in terminal)
```

Node.js process
(C/C++ runs V8, libev, libeio)

JavaScript (being executed by **V8**) `JS`

Express.js (middleware framework)

morgan (logger middleware)

next();

express.static (middleware)

next();

body-parser (middleware)

next();

express.Router (middleware)

next(err);

error handling middleware

Mongoose (ODM) / **Sequelize** (ORM) `m`

schema

model → instances

mongodb / **pg** (database driver)

```
$ console/shell (in terminal)
```

CLIENT ("FRONT END")

Machine 2 (computer + operating system)

HTTP

Chrome process
(C/C++ runs V8, DOM, etc.)

🏠 📄 www.mycoolapp.com/somepath

Address bar (makes GET requests)

DOM (node tree, what gets rendered)

html
  └ body
      └ h1
      └ button

JavaScript (executed by **V8**) `JS`

```
> console (in dev tools)
```