

# PostgreSQL

The RDBMS with the Elephant Logo

# Database Management System (DBMS)

- How to define records (Data Definition)
- CRUD, in a performant way (B-Trees, etc.)
- Administration

# Progression of Databases

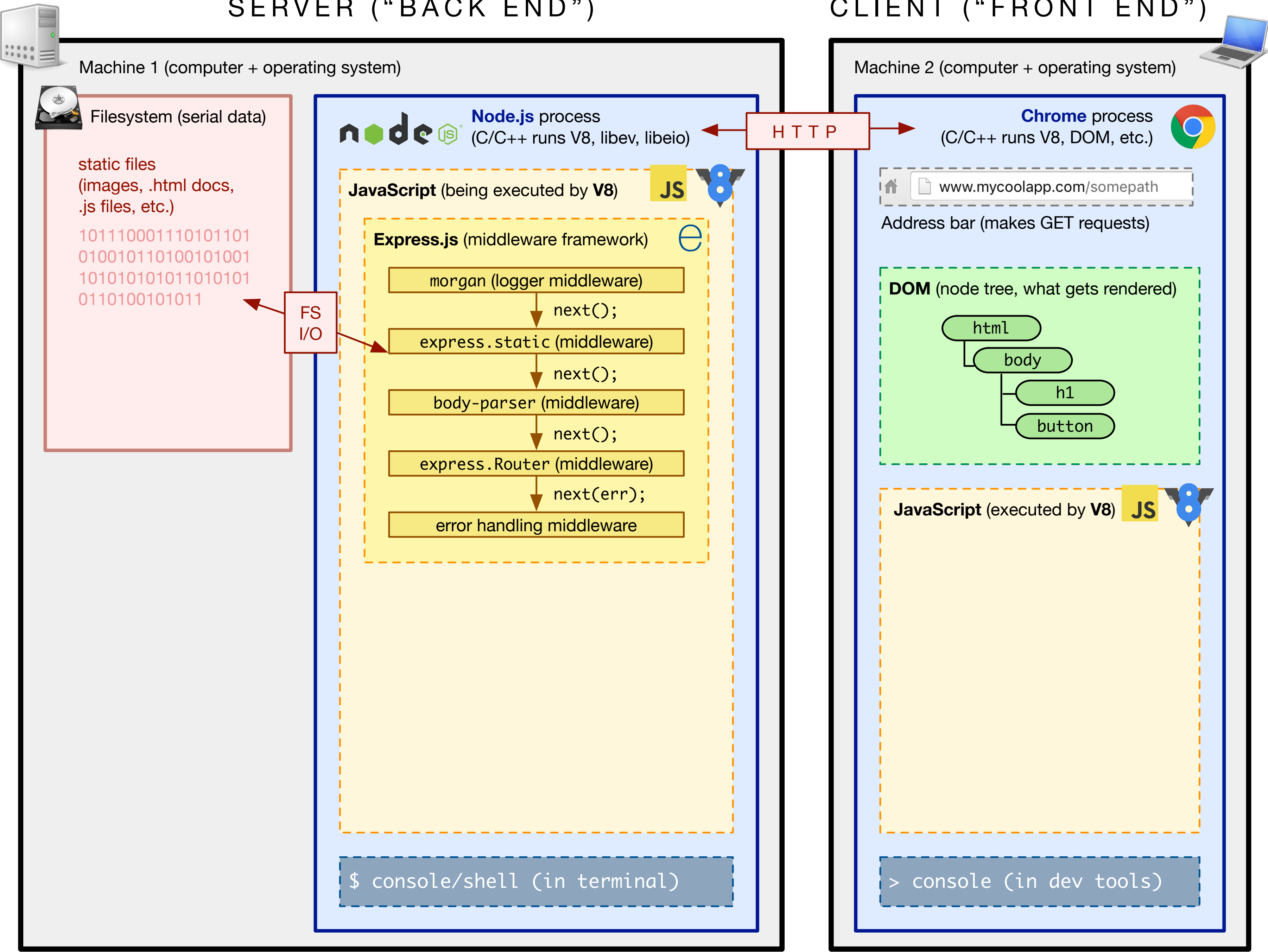
- Navigational (< 1970s)
- Relational (> 1970s)
- NoSQL (> 2000s)

# History of Postgres

- 1970s at UC Berkley:  
**I**Nteractive **G**raphics **R**etrieval System (INGRES)
- 1980s: POSTGRES ("Post-Ingres")
- 1995: POSTQUEL and Postgres95. monitor -> psql
- 1996: Open source community adopts it
- Ongoing: stability, testing, documentation, new features
- PostgreSQL

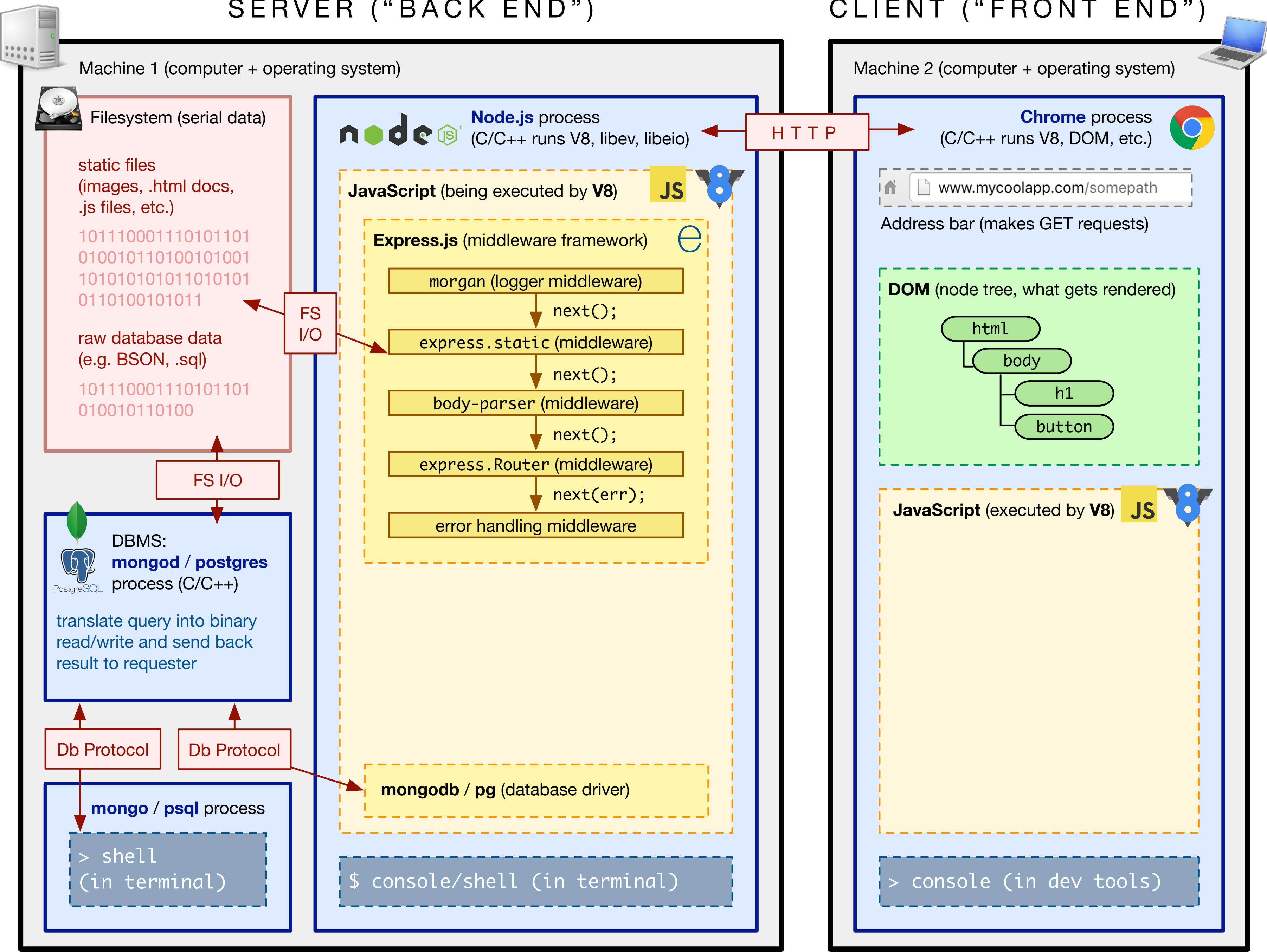
SERVER ("BACK END")

CLIENT ("FRONT END")



SERVER ("BACK END")

CLIENT ("FRONT END")



# postgres process

- Waiting for incoming SQL
- Knows how to read/write to disk in an organized, performant way
- Sends back results

Where does the "incoming SQL" come from?



# Query sources

- psql CLI (human input as text)
- GUI like Postico, Datazenit (human actions turned into SQL queries)
- ...and other applications

# Tooling

- psql
- CLI

```
1. psql (psql)
X .../hello-world (zsh) 1 X ...s/class-libs (zsh) 2 X psql (psql) 3 X postgres (postgres) 4

psql [local]:5432 glebec # ~ \l
                                List of databases

```

Name	Owner	Encoding	Collate	Ctype	Access privileges
assessmentexpresssequelize	fullstack	UTF8	en_US.UTF-8	en_US.UTF-8	
auther	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	
checkpoint_angular	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	
checkpoint_express_review	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	
glebec	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	
juke	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	
postgres	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	
sequelizecheckpoint	fullstack	UTF8	en_US.UTF-8	en_US.UTF-8	
template0	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	=c/glebec ↵ glebec=CTc/glebec
template1	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	=c/glebec ↵ glebec=CTc/glebec
tripplanner	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	
twitterdb	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	
wikistack	glebec	UTF8	en_US.UTF-8	en_US.UTF-8	

```

(13 rows)

psql [local]:5432 glebec # ~ \c auther
You are now connected to database "auther" as user "glebec".

psql [local]:5432 glebec # auther \dt
                                List of relations

```

Schema	Name	Type	Owner
public	stories	table	glebec
public	users	table	glebec

```

(2 rows)

psql [local]:5432 glebec # auther
```

# Tooling

- Postico
- Free

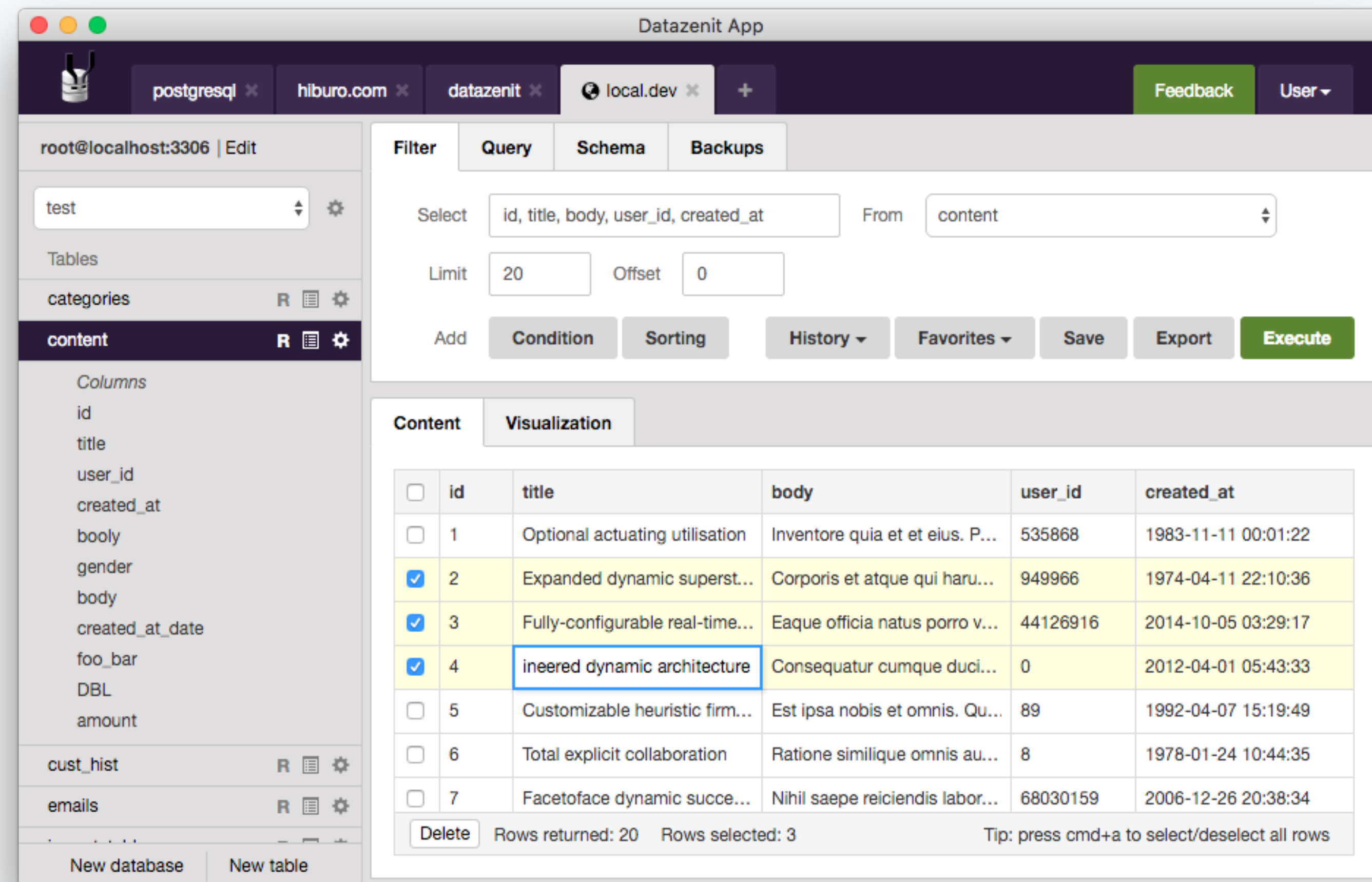
The screenshot shows the Postico application interface. At the top, there's a navigation bar with tabs for 'Reporting', 'reporting', and 'forex'. The 'forex' tab is selected. Below the navigation bar, there's a status bar showing 'Connected.' and 'PostgreSQL 9.4.5'. The main area displays a table with columns: currency, base, rate, date, and source\_id. The table is filtered to show data for the date '2014-07-25'. The 'forex' table is selected in the left sidebar. On the right side, there are filters for 'currency' (set to 'MULTIPLE'), 'base' (set to 'USD'), 'rate' (set to 'MULTIPLE'), and 'date' (set to '2014-07-25'). Below these filters, there's a section for 'source\_id' with a dropdown menu showing '1' and 'Open Exchange R...'. At the bottom, there's a search bar and a pagination bar showing 'Page 1 of 342'.

currency	base	rate	date	source_id
AED	USD	3.67291	2014-07-25	1
AFN	USD	56.485726	2014-07-25	1
ALL	USD	103.5838	2014-07-25	1
AMD	USD	410.086	2014-07-25	1
ANG	USD	1.787	2014-07-25	1
AOA	USD	96.952626	2014-07-25	1
ARS	USD	8.169642	2014-07-25	1
AUD	USD	1.062844	2014-07-25	1
AWG	USD	1.79	2014-07-25	1
AZN	USD	0.784067	2014-07-25	1
BAM	USD	1.453024	2014-07-25	1
BBD	USD	2	2014-07-25	1
BDT	USD	77.61971	2014-07-25	1
BGN	USD	1.452543	2014-07-25	1



# Tooling

- Datazenit
- \$\$\$



Etc.

How to transmit SQL text to app?  
How can postgres be "waiting for SQL"?  
And how do the results get "sent back"?

# Postgres is a TCP server!

- Listening on a TCP port (5432 by default) for *requests*
- Does disk access
- Sends back a TCP *response* to the *client* that made the requests

OK, Postgres is a TCP server. Is it... HTTP?



# Postgres uses the postgres:// protocol

	Transport Protocol	Message Protocol	Content Type
Node + Express	TCP/IP	http://	Anything: HTML, JSON, XML, TXT, etc.
Postgres	TCP/IP	postgres://	SQL

For HTTP clients, the TCP/IP was handled for you by the browser or Node. How can our JS app communicate with the postgres server?

“Let's implement the postgres protocol in  
JavaScript ourselves!”

-Ambitious McOverkill

<https://www.postgresql.org/docs/current/static/protocol.html>

"On second thought...  
has anyone done this for us?"

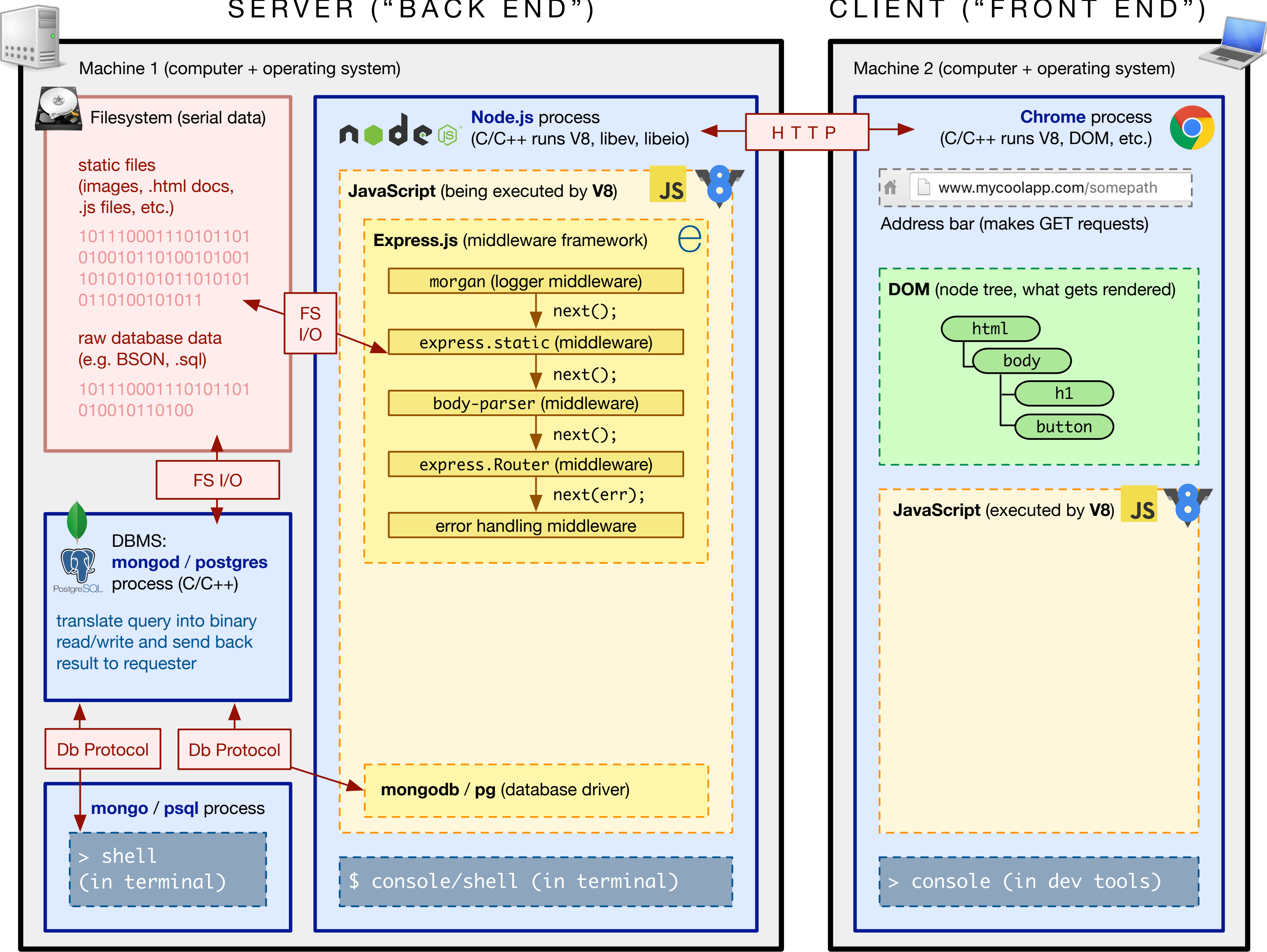
-Saney McReasonable

# Node-postgres

- npm library: `npm install pg --save`
- *database driver*
- implements the postgres protocol in a Node module (JS!)
- Gives us a `client` object that we can pass SQL to
- Asynchronously talks via postgres protocol over TCP to postgres
- gives us a callback with `rows` array of resulting table

SERVER ("BACK END")

CLIENT ("FRONT END")



# Final note: `returning`

- SQL comes in slightly different "dialects" depending on your RDBMS of choice (SQLite, MySQL, PostgreSQL etc.)
- PostgreSQL has a very convenient keyword `returning`
- Used during INSERT, UPDATE
- Returns the row(s) inserted/updated
- May come in handy during workshop, so check it out!