



{mfs.js}

Modern Full-Stack JavaScript

Cody Zuschlag



 **codyzus**

Cody Zuschlag

Developer Relations Engineer @NearForm
University Instructor @ Université Savoie Mont Blanc

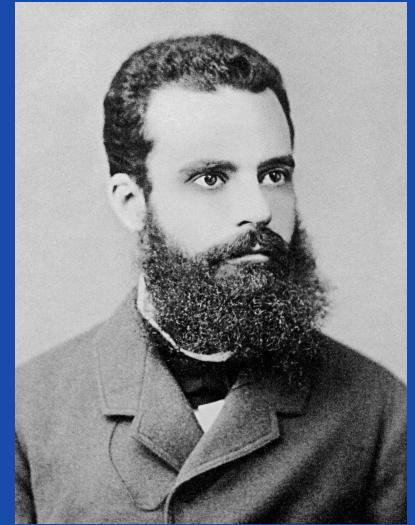
France

How we imagine our next project...



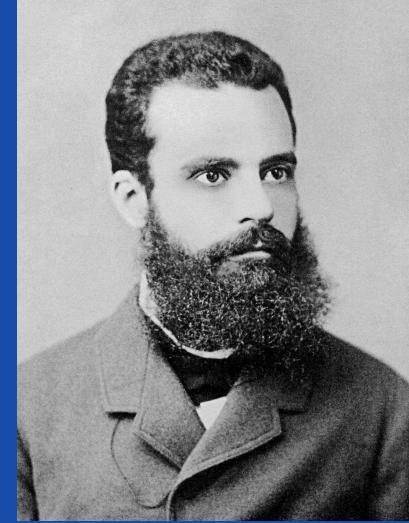


...reality



**For many outcomes, roughly
80% of consequences come from
20% of causes (the "vital few")**

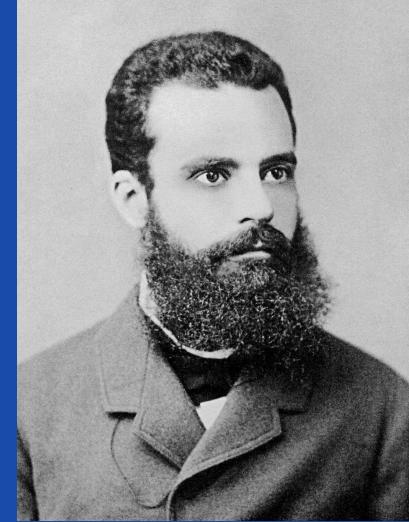
- Pareto Principle



**Vilfredo
Pareto**

0%

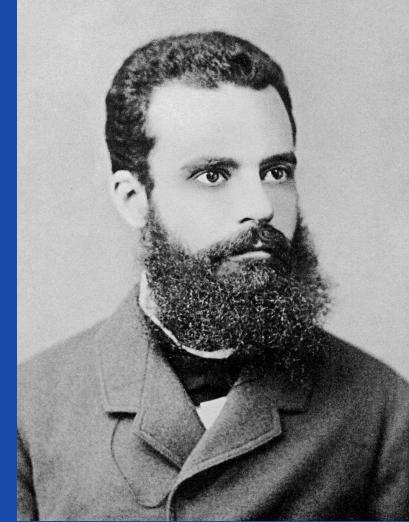
**For many outcomes, roughly
80% of consequences come from
20% of causes (the "vital few")**
- Pareto Principle



**Vilfredo
Pareto**



**For many outcomes, roughly
80% of consequences come from
20% of causes (the "vital few")**
- Pareto Principle

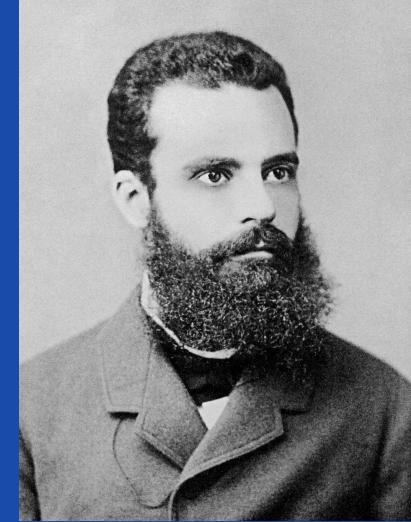


**Vilfredo
Pareto**



**For many outcomes, roughly
80% of consequences come from
20% of causes (the "vital few")**

- Pareto Principle



**Vilfredo
Pareto**



Steve Jobs returns to Apple in 1996...



20%

80%

Steve Jobs returns to Apple in 1996...

**...and cuts the product models in
all categories down to 1 (or none)**



20%

80%

Focus on the hard 20%...

 **20%****80%**

...gain 80% in productivity?

20%

80%

How can we build a stack that works for the real world?





Modularity wins



DevEx matters





Community first





Transparency counts



The stack

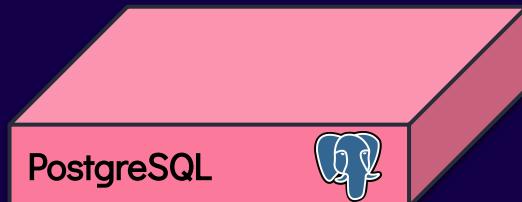






Database Layer

PostgreSQL



Developer Knowledge

Devs know it today and can design systems with it now



Super powers

Best RDB + NoSQL ready
Supports connection pooling (massive scaling)



DevEx

Run anywhere, locally or otherwise 🌎
SQL is crazy powerful

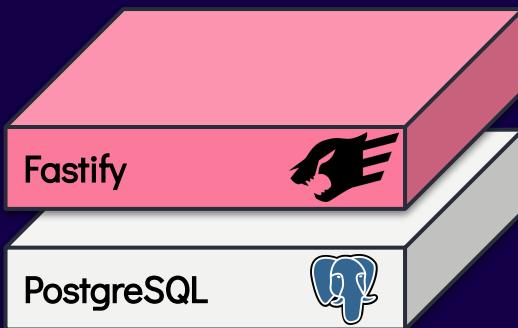






Server Layer

Fastify



Developer Knowledge

Inspired by hapi and familiar to express devs



Super powers

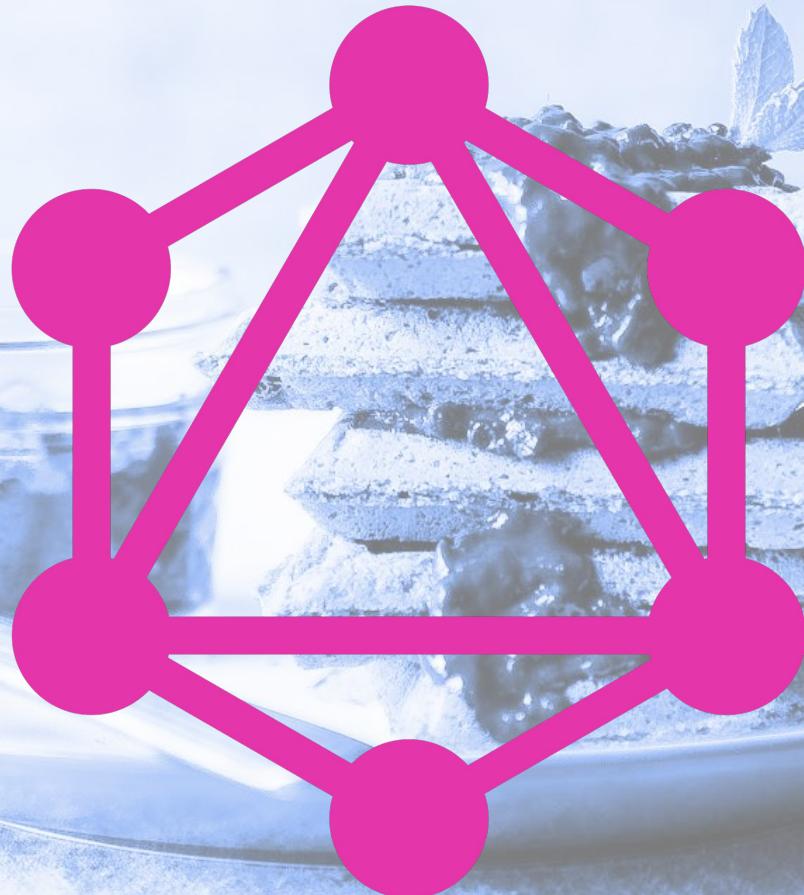
Monolith today, services tomorrow (Conway)
Insane scaling



DevEx

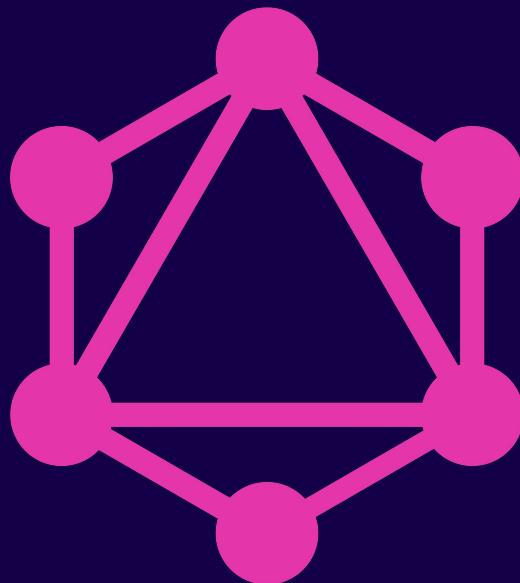
Run anywhere, locally or otherwise 🌎
Plugin architecture is programming zen





API Layer

GraphQL



Developer Knowledge

Real world -> data is a graph, can model business domain



Super powers

Eliminate global state, efficiency, federation, performance & caching

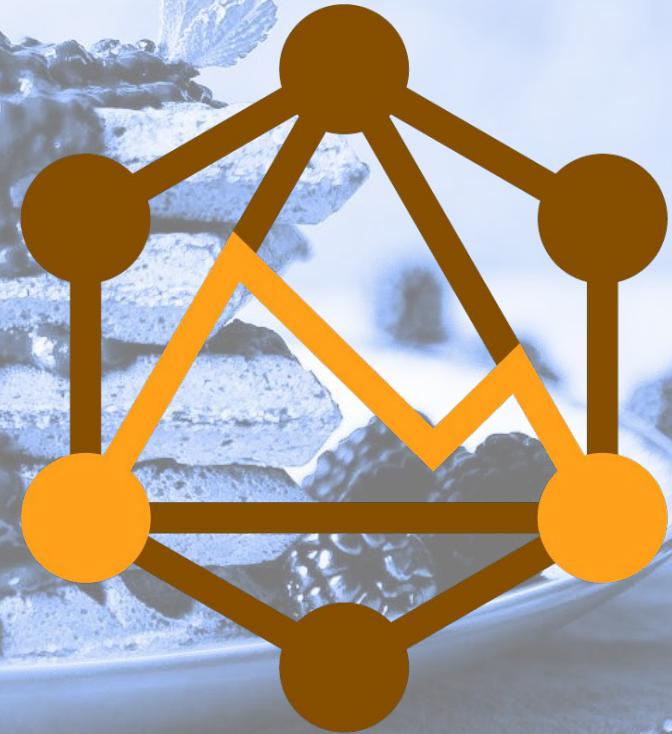


DevEx

Graphql
Entity depiction



Mercurius



API Layer

Mercurius



Developer Knowledge

Solves real-world graphQL challenges out of the box



Super powers

Caching, federation, authentication, subscription, file uploads 



DevEx

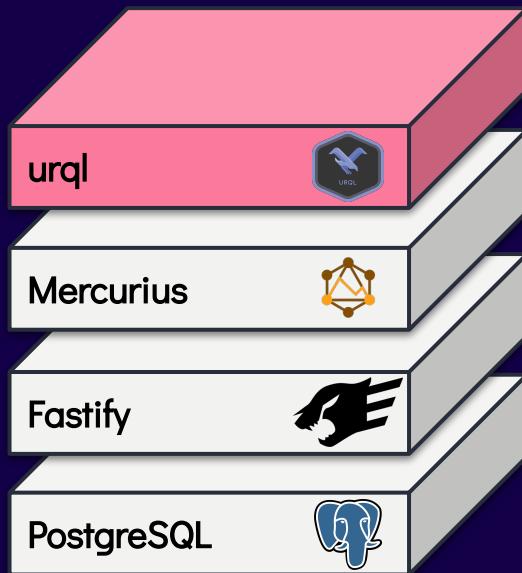
Integrates graphql
Loaders support built-in





API Client

urql



Developer Knowledge

Works with the major front-end libs



Super powers

Caching, offline support, subscriptions, uploads, exchanges architecture, SSR



DevEx

Hooks API 🧙 Devtools

```
const pizzasQuery = `query {
  pizzaList {
    id
    name
    toppings {
      id
      name
    }
  }
}`;
```

```
export default function PizzaList() {
  const [result, reexecuteQuery] = useQuery({
    query: pizzasQuery,
  });

  const {data, fetching, error} = result;

  if (error) {
    return <span>something went wrong: {error.message}</span>;
  }

  if (fetching) {
    return <span>Loading...</span>;
  }

  return (
    <Container>
      <Row>
        {data.pizzaList.map((pizza) => (
          <Col key={pizza.id}>
            <Pizza {...pizza} />
            <Link to={`/pizza/${pizza.id}`}>Details</Link>
          </Col>
        )));
      </Row>
    </Container>
  );
}
```

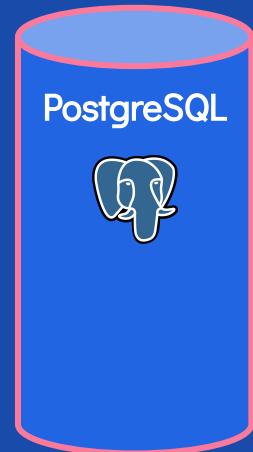
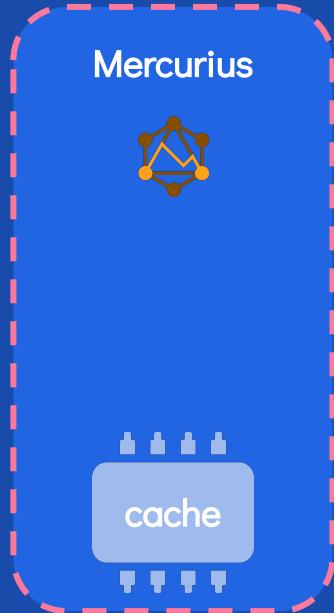
```
const pizzaQuery = `query($pizzaId: Int!) {  
  pizza(id: $pizzaId) {  
    id  
    name  
    toppings {  
      id  
      name  
    }  
  }  
}`;
```

```
export default function PizzaDetails({pizzaId}) {  
  const [result, reexecuteQuery] = useQuery({  
    query: pizzaQuery,  
    variables: {pizzaId},  
  });  
  
  const {data, fetching, error} = result;  
  
  if (error) {  
    return <span>something went wrong: {error.message}</span>;  
  }  
  
  if (fetching) {  
    return <span>Loading...</span>;  
  }  
  
  return <Pizza {...data.pizza} />;  
}
```

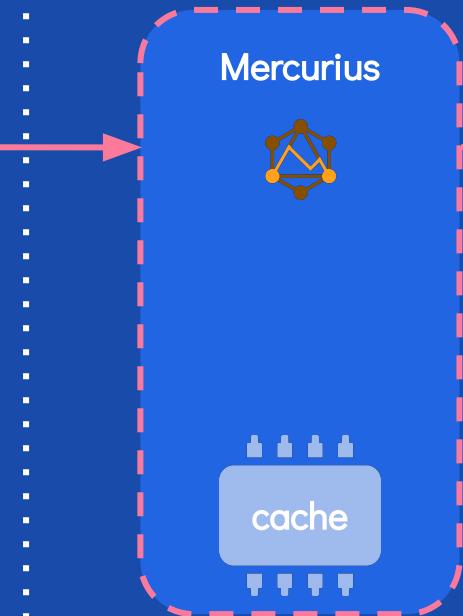
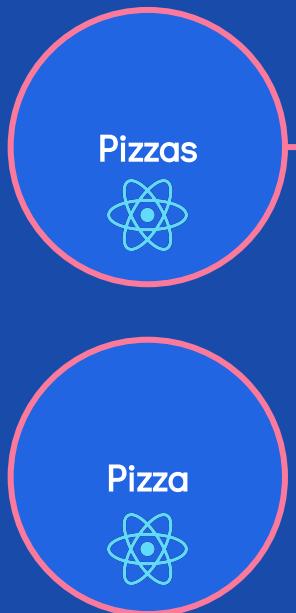
client



server



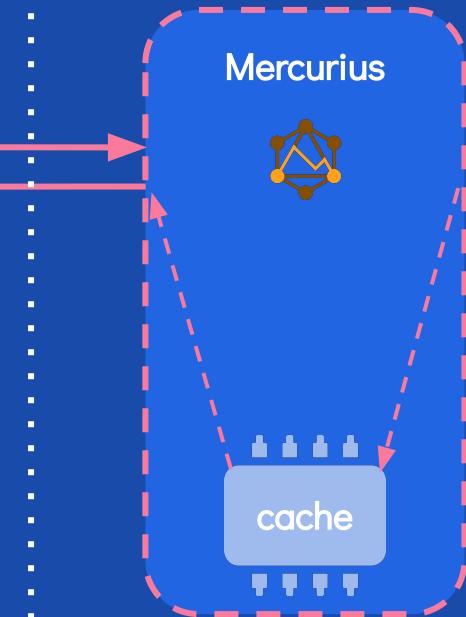
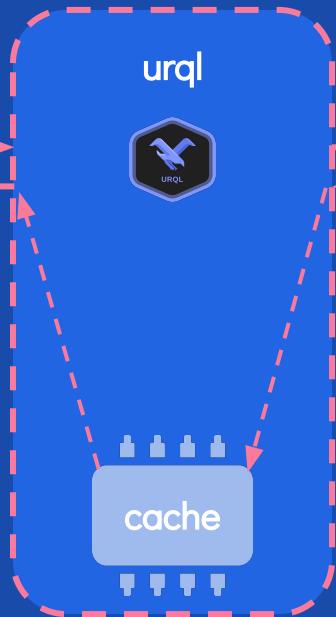
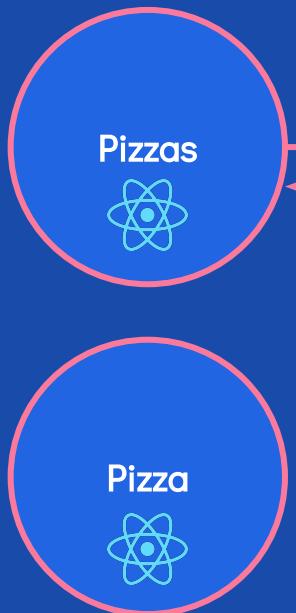
client



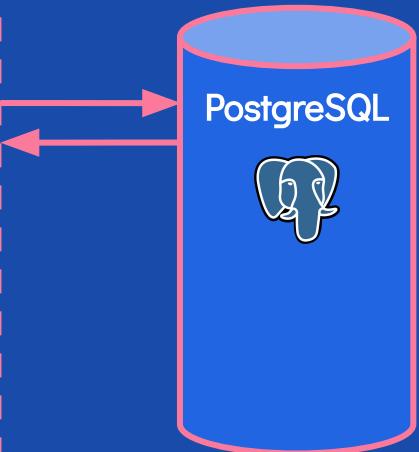
server



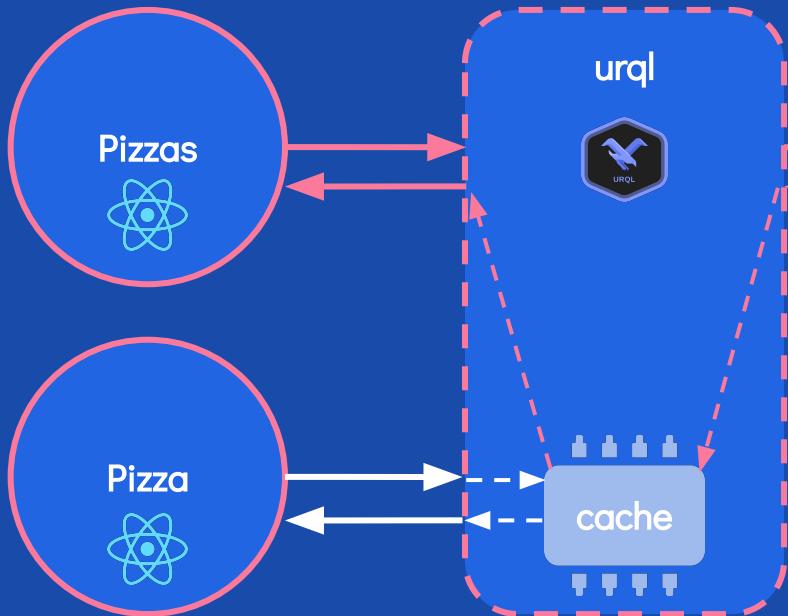
client



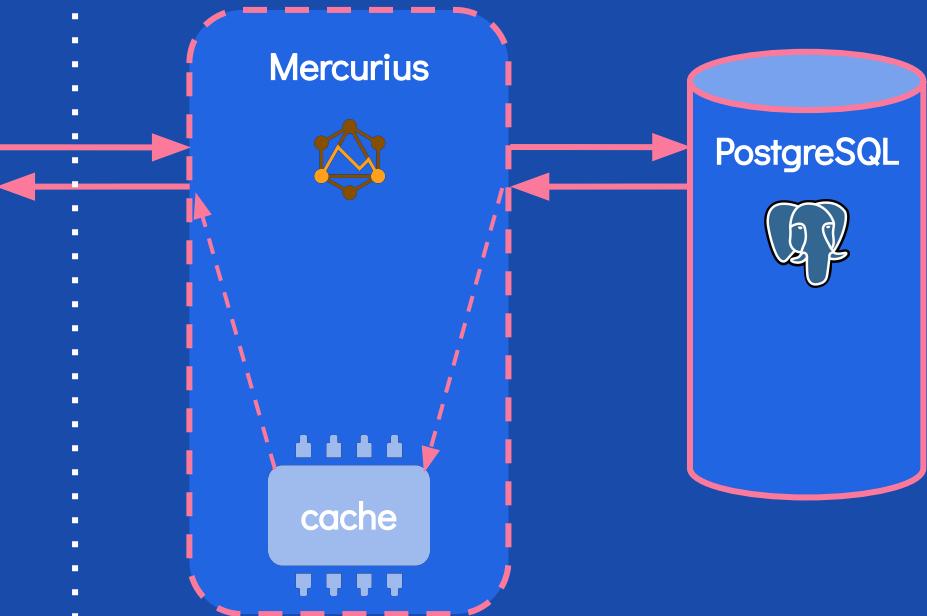
server



client



server



**graphQL APIs that
match your business
domain...**

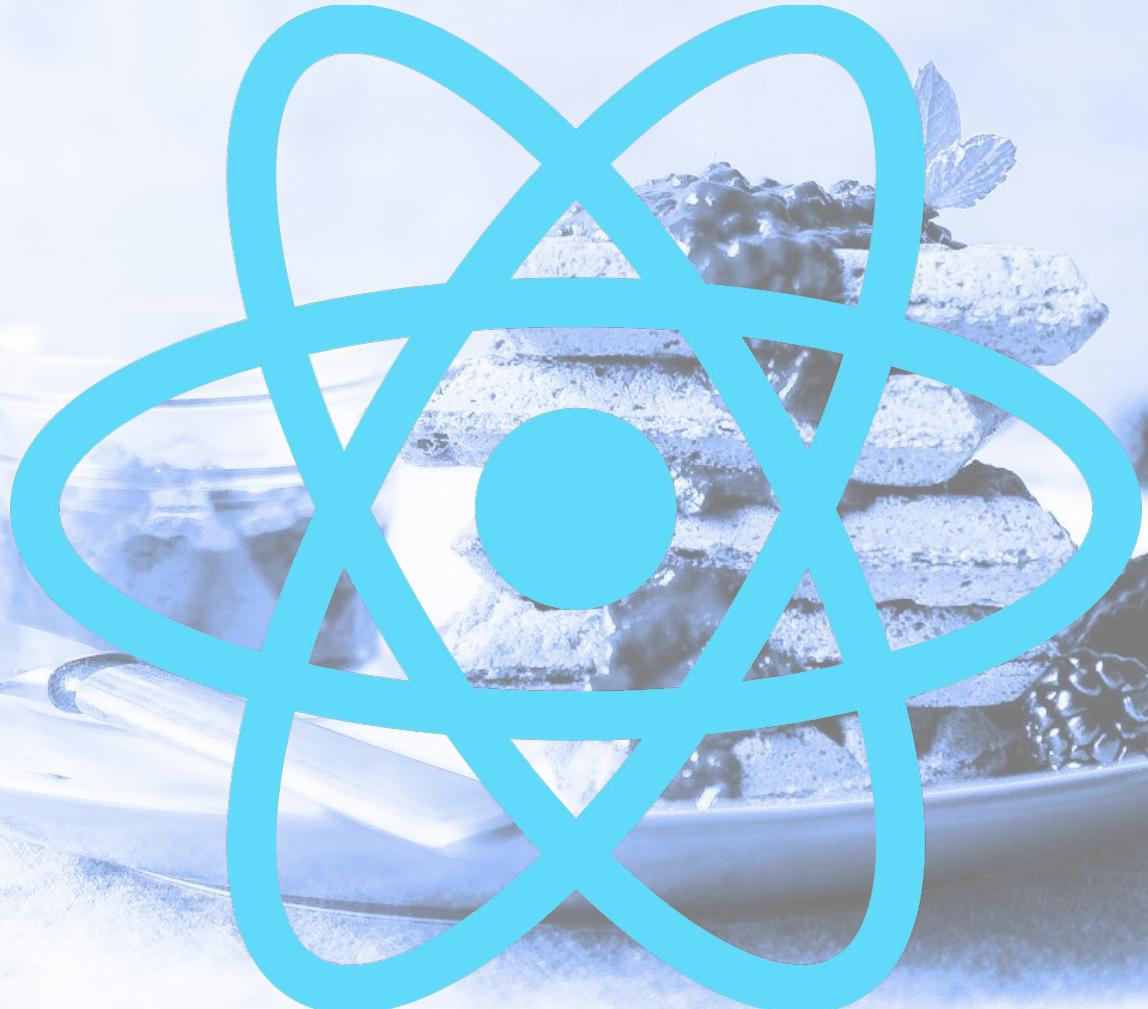
BAD ASS



MOTHER FCKER

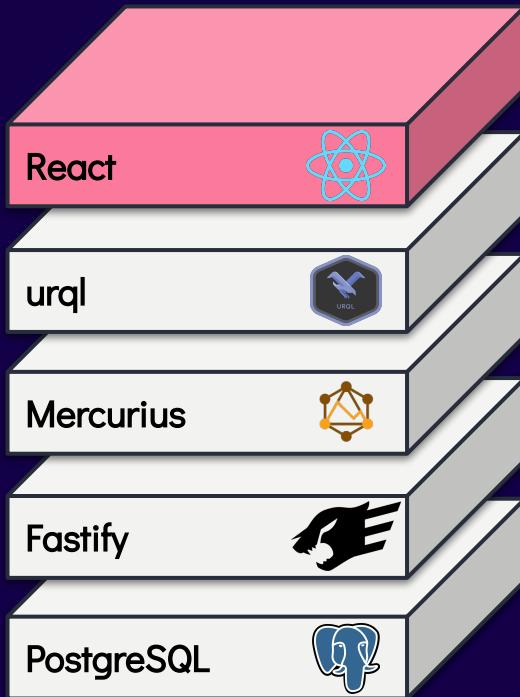
makeameme.org





UI Layer

React



Developer Knowledge

Umm... its React!
Devs know it



Super powers

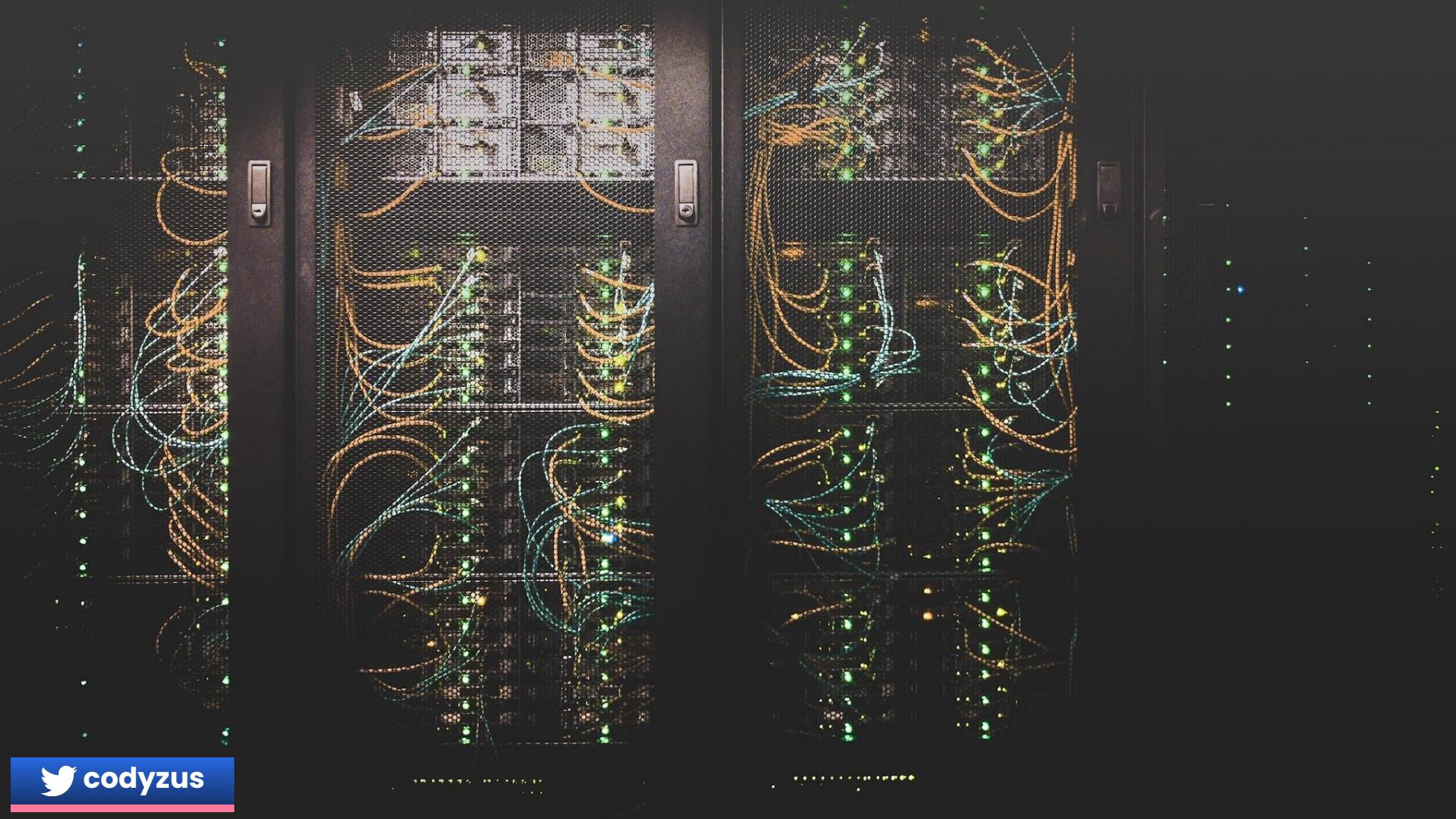
Evolving yet stable, components first, hooks



DevEx

Devtools
Vite! Just use it.





SSR



SSR

fastify-vite



Developer Knowledge

Works with major front-end libs



Super powers

Handles page level templating
Integrates client routing into fastify server routes



DevEx

Highly customizable
No magic

Welcome to Fastify DX!

localhost:3000

fastify
DX

Welcome to Fastify DX for React!

- [`/using-data`](#) demonstrates how to leverage the `getData()` function and `useRouteContext()` to retrieve server data for a route.
- [`/using-store`](#) demonstrates how to leverage the automated `Valtio` store to retrieve server data for a route and maintain it in a global state even after navigating to another route.
- [`/client-only`](#) demonstrates how to set up a route for rendering on the client only (disables SSR).
- [`/server-only`](#) demonstrates how to set up a route for rendering on the server only (sends no JavaScript).
- [`/streaming`](#) demonstrates how to set up a route for SSR in streaming mode.



Why this stack?

04 Community supported

Open source, community supported projects security and constant evolution.

02 Highly customizable

The stack can be customized to your business logic.

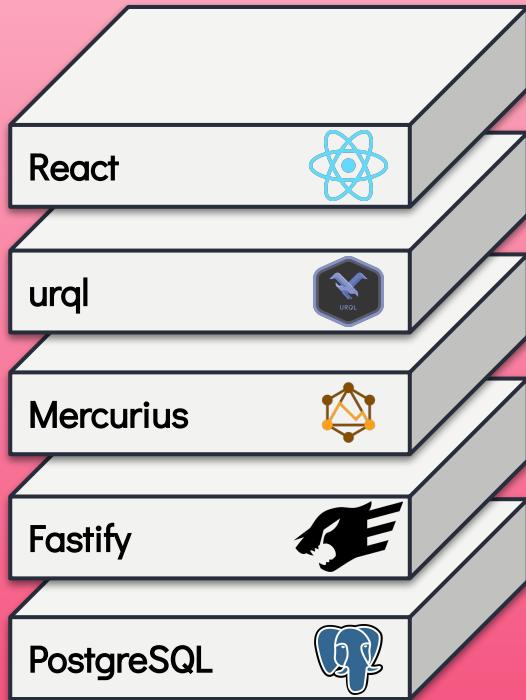
01 Built for complex functionality

Simple from the start, but built with that last, hard, 20% of functionality in mind.

03 No magic

Transparent abstractions provide the common use case, but don't hide the details.

LEAN, MEAN, JAM, ...



???



WE'RE BOLD
WE'RE FLEXIBLE
WE'RE OPEN
WE'RE EMPOWERING

Global Delivery Org with 300+ and counting

We are hiring!

**Major Contributors to the
Open Source Web Platform**

NPM monthly downloads

1B

Represents modules used globally

8%

Questions?

