# React Framework Showdown🥊

# DISCLAIMER 😬

Every framework has its use cases and a lot of work goes into making web better 🕸️

# GOAL 😉

Help you assess the pros and cons of each framework so you can feel empowered to make that decision yourself

# OATH TO BE UNBIASED 🤟

# HI, I'M ANKITA! 👋

**Ankita Kulkarni**

Educator | Senior Engineering Leader | Developer

bit.ly/reactrallylinks

@kulkarniankita9 @kulkarniankita

# SLIDES & MORE...

👋

Slides and code snippets at:
bit.ly/reactrallylinks

# RAISE YOUR HAND ✋

How many of you have used Next.js before?

# RAISE YOUR HAND ✋

How many of you have used Remix before?

# RAISE YOUR HAND ✋

How many of you have used Astro before?

# We'll be comparing Astro, Remix and Next.js

Your tool choice 🔨 matters much less than your skill at using the tool to accomplish your desired outcome (a great user experience) 👌.

- Kent C Dodds

# WHAT IS A GOOD DEVELOPER EXPERIENCE?

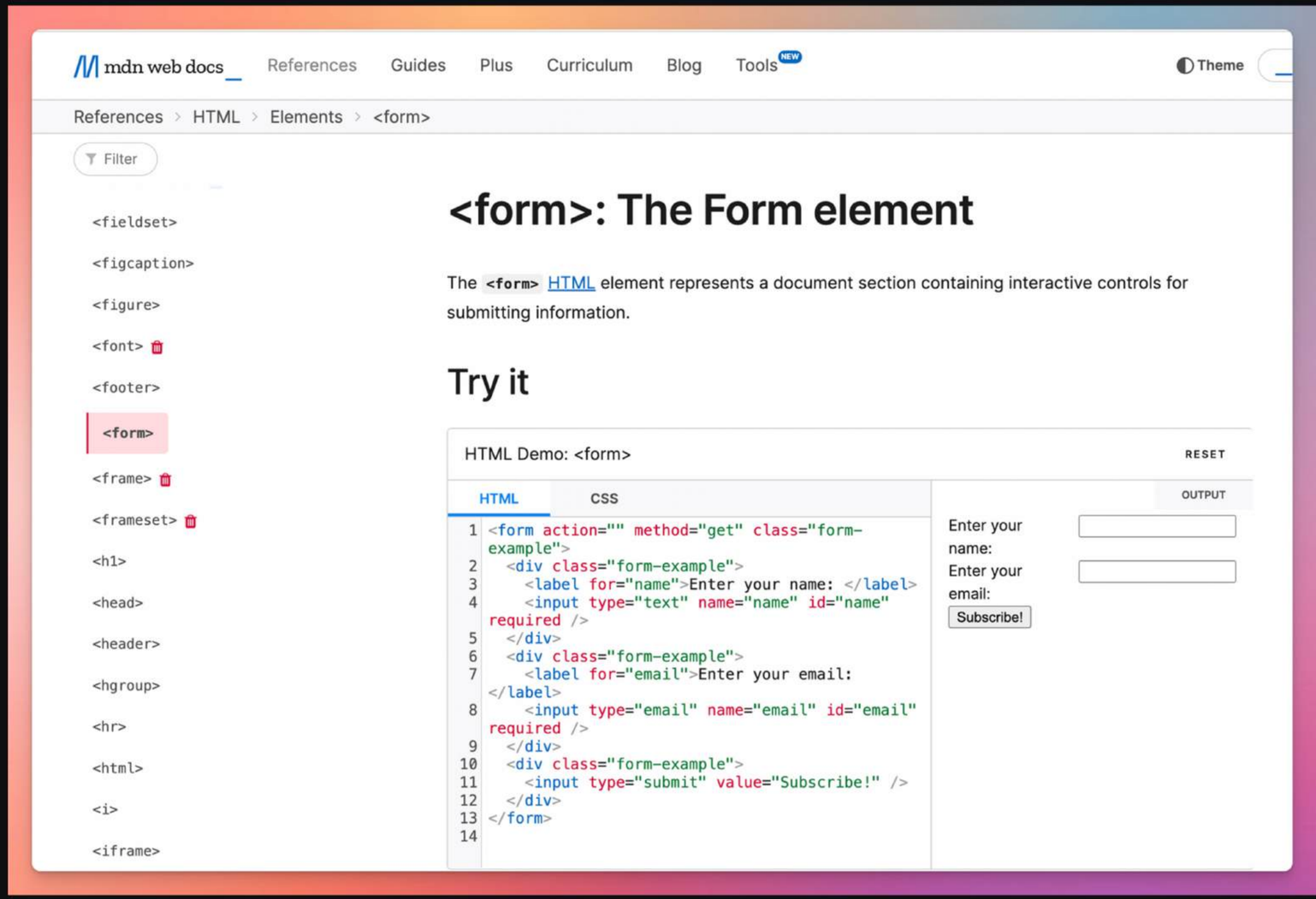
# What is Remix? 

# WHAT MAKES REMIX SPECIAL

It unifies the client and server with web standards so you can think less about code and more about your product.

# When you google

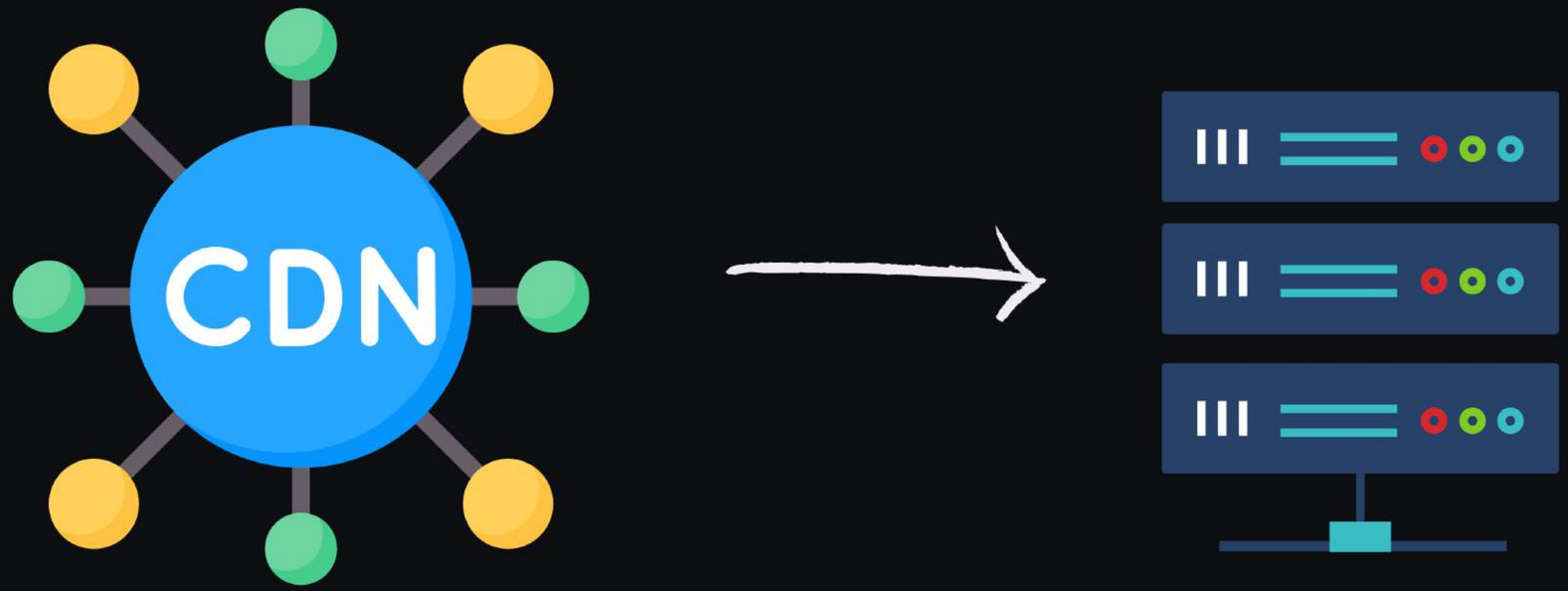
## You go to MDN docs vs docs created by Remix




bit.ly/reactrallylinks

@kulkarniankita9 @kulkarniankita

# Flexibility with Remix



When you bet on Remix, you bet on web standards, Vite eco-system etc.

# What is Next.js? 🚀

# NEXT.JS

It is a framework built on top of React that gives you the flexibility of building scalable apps by allowing you to render content on the server

# WHAT MAKES NEXT.JS SPECIAL?

Data fetching mechanisms

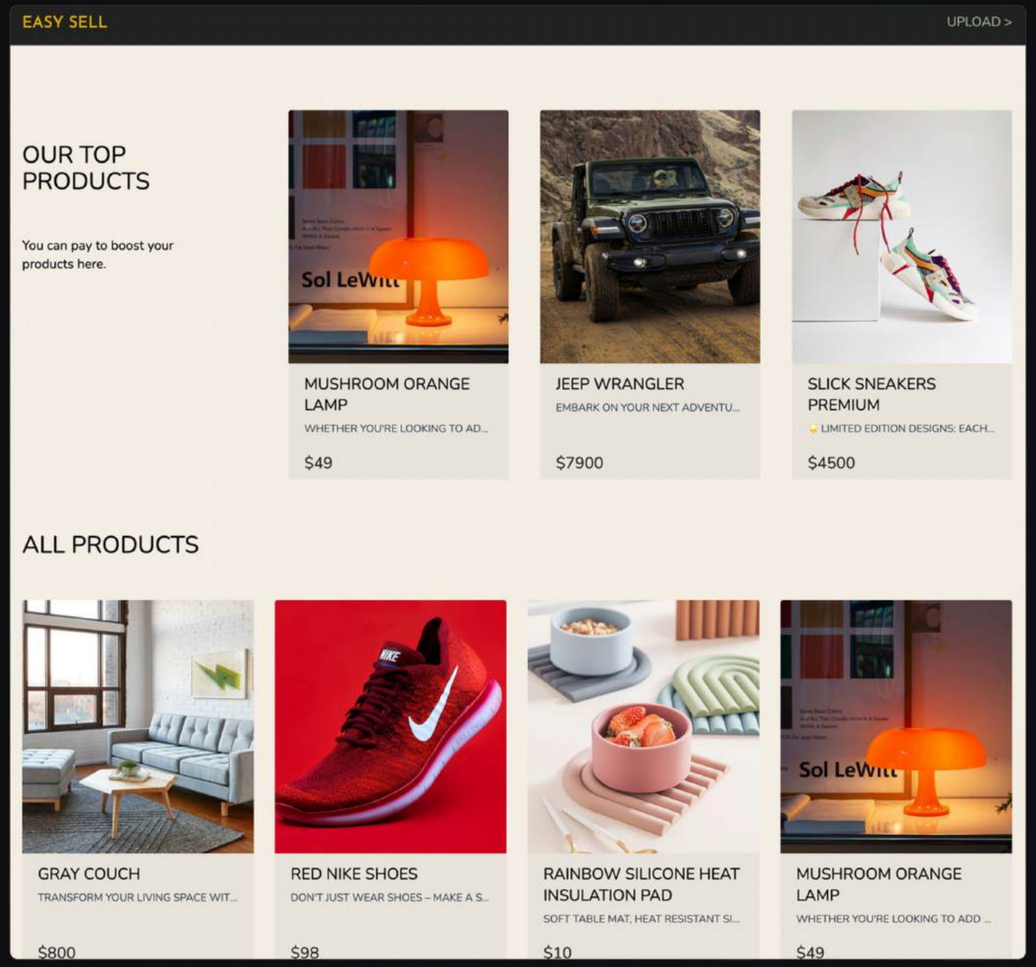
Custom components

Server Components

Automatic Code Splitting

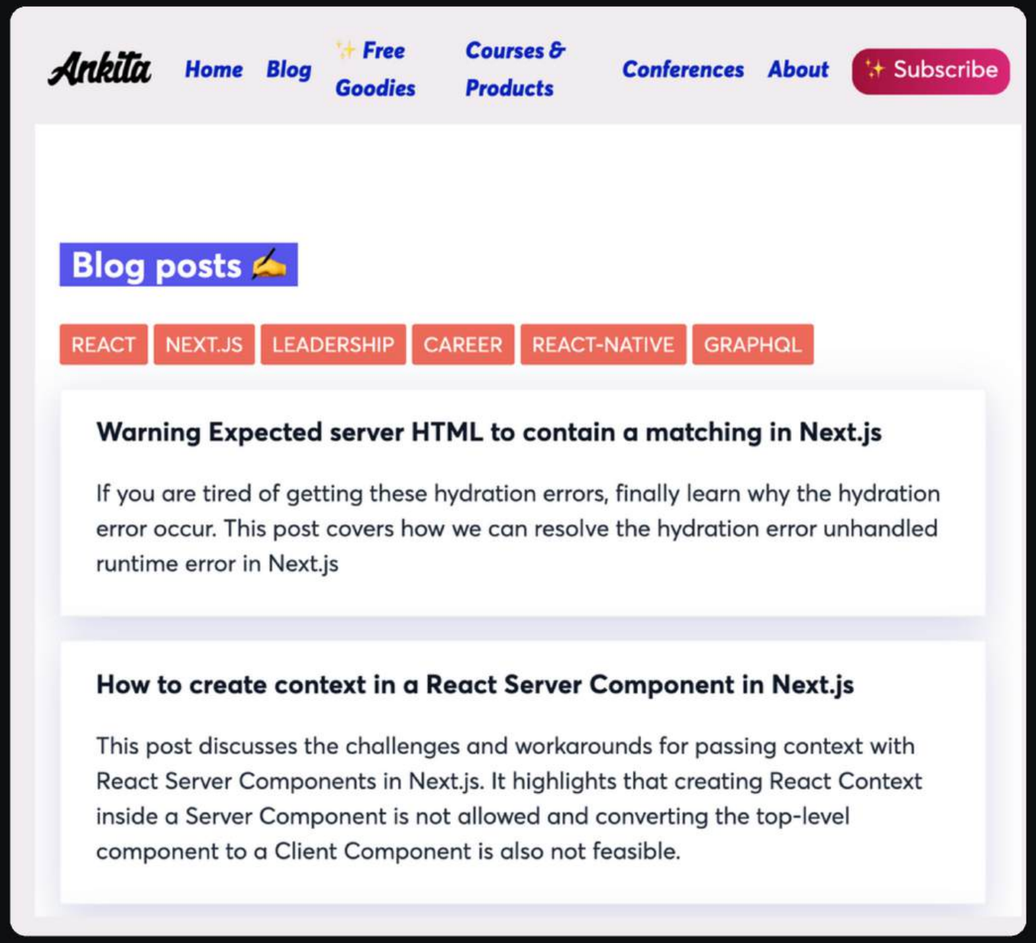
A layer on top

# BUILDING STATIC & DYNAMIC CONTENT

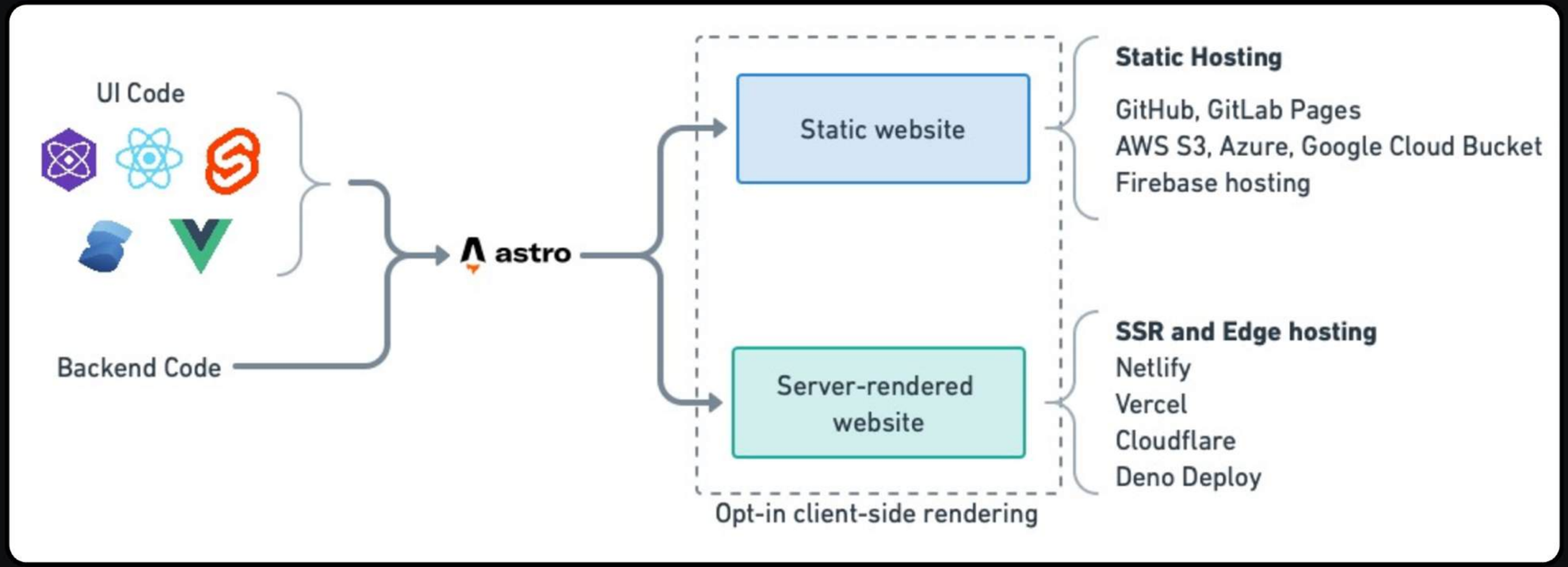


# What is Astro? 

# BUILDING STATIC CONTENT

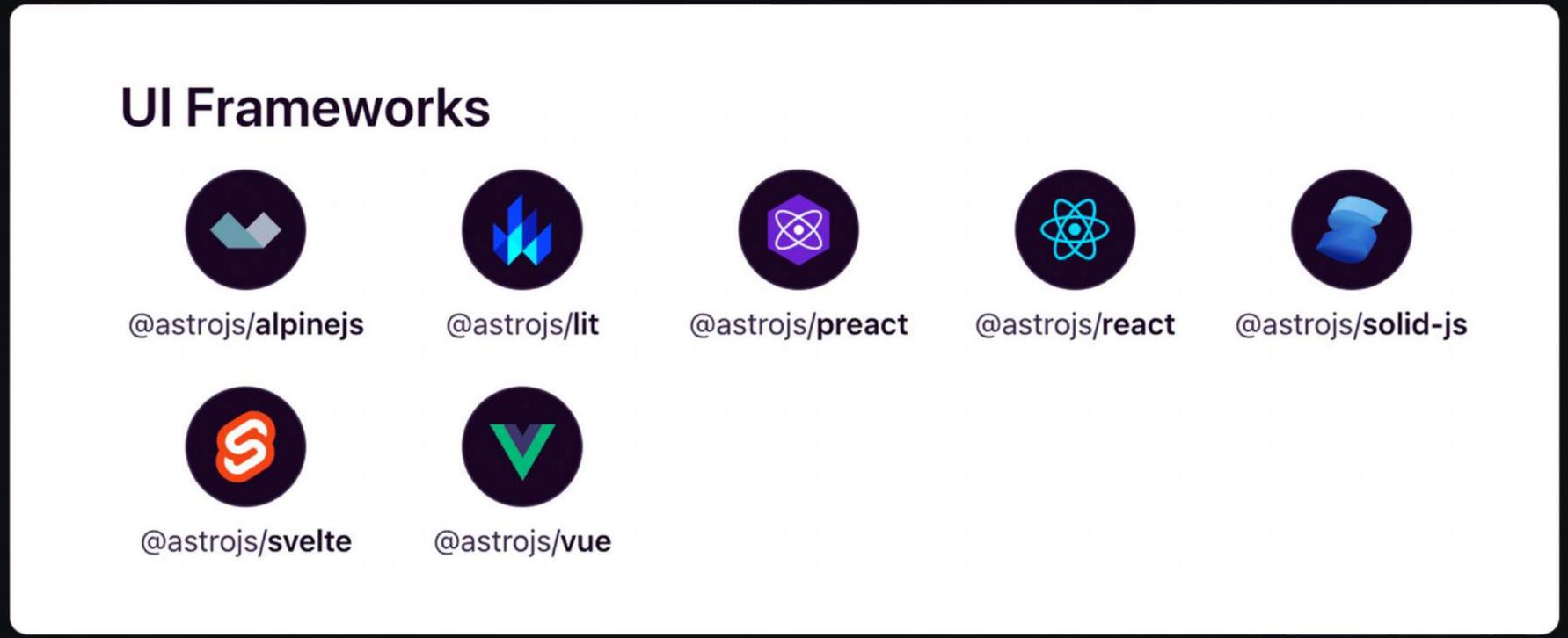


# ASTRO




Reference

# Astro works with any framework



# You don't need React or Vue to get started

about.astro

```
---
import BaseLayout from "../layouts/BaseLayout.astro";

const pageTitle = "About Me";
const skills = ["HTML", "CSS", "JavaScript", "React", "Astro", "Writing Docs"];

const skillColor = "navy";
---

<style define:vars={{ skillColor, fontWeight, textCase }}>
  .skill {
    color: var(--skillColor);
  }
</style>

<BaseLayout pageTitle={pageTitle}>
  <h2>... and my new Astro site!</h2>

  <p>My skills are:</p><p></p><ul>
    {skills.map((skill) => <li class="skill">{skill}</li>)}
  </ul>
</BaseLayout>
```

bit.ly/reactrallylinks

@kulkarniankita9 @kulkarniankita

# But you can get React if you'd like

# 6 PILLARS 💪

- Adoption
- Developer Experience
- Performance
- Routing
- Data Fetching
- Deployment

# Adoption

Learning Curve with Next.js

Learning Curve with Remix

bit.ly/reactrallylinks

@kulkarniankita9 @kulkarniankita

# THINGS TO LEARN

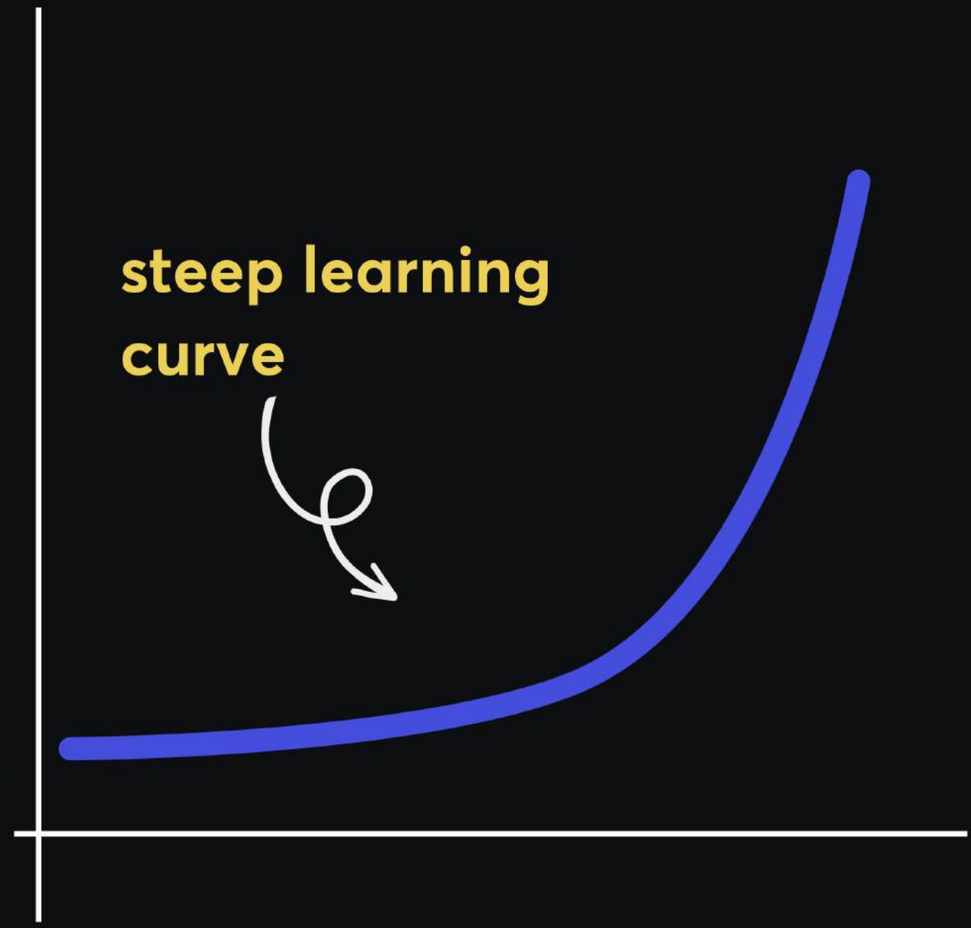
App & Pages Router

Server Actions

Server Components

Different behaviours in versions

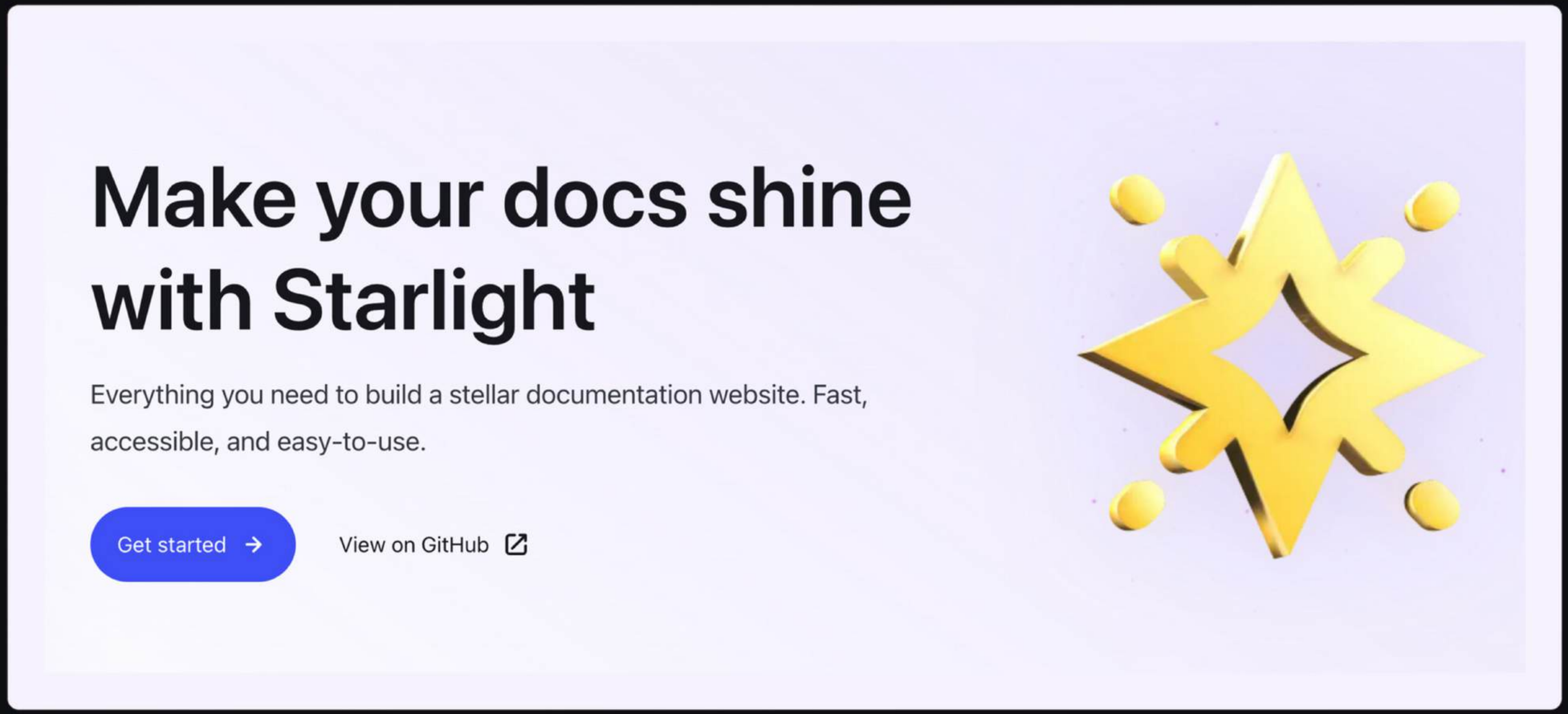
Different rendering methods



# ASTRO

# UNTIL YOU INTRODUCE SSR

# Course Platform Build Story 📙




bit.ly/reactrallylinks

@kulkarniankita9 @kulkarniankita

# REMIX

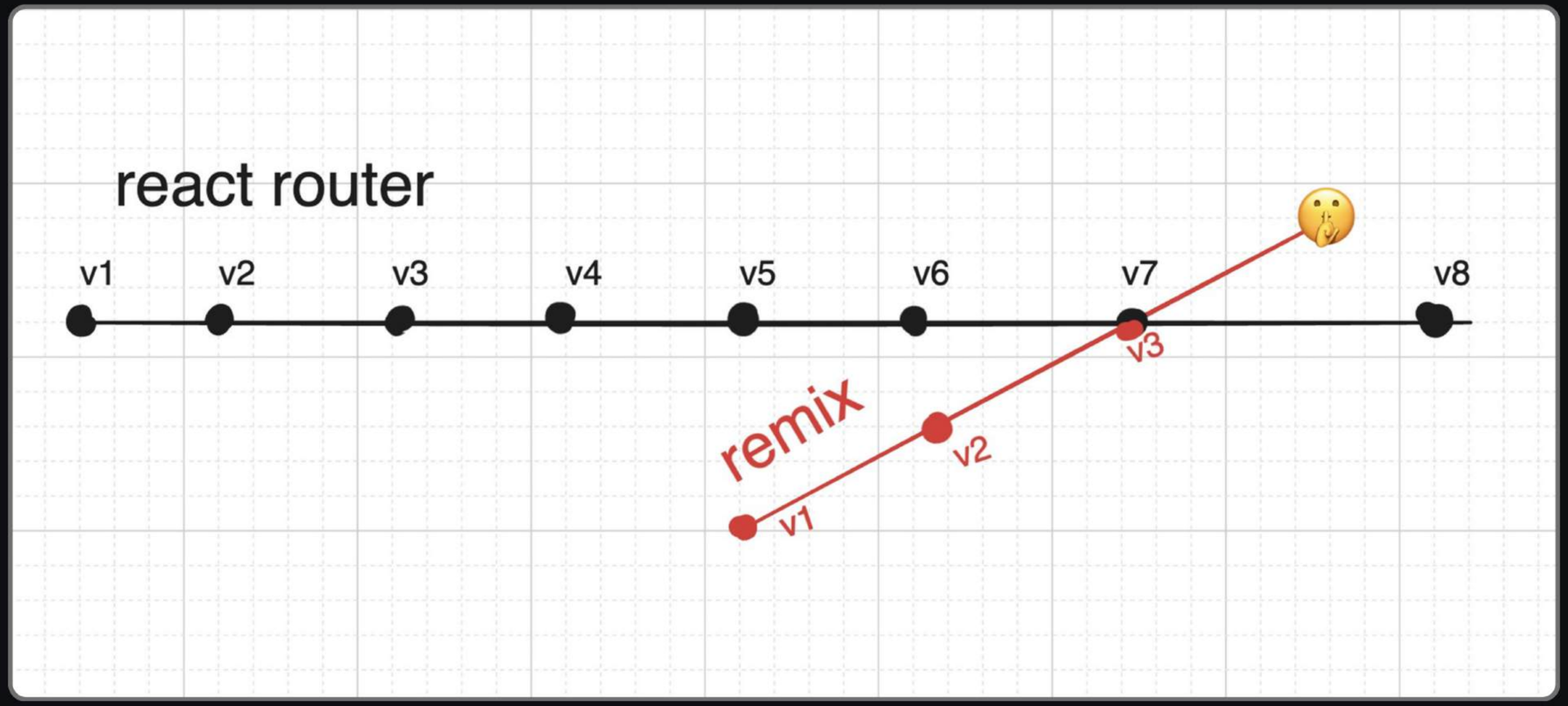
## React Router V7 🤝 Remix

There are millions of projects using React Router, many built on top of Create React App (CRA). These days CRA is no longer recommended and the React docs recommend using a framework. Since Remix has always been effectively **"React Router: The Framework"**, we wanted to create a bridge for all these React Router projects to be able to upgrade to Remix.

Turns out we made that bridge a little too well, specifically with the introduction of our Vite plugin and SPA Mode. We found ourselves looking at Remix, then looking at React Router, then looking back at Remix, and we could no longer meaningful tell the difference.


So we're remixing React Router (again).

# REMIX



Ecosystem around Next.js

🎂 7 yrs ago

Ecosystem around Remix

🎂 3 yrs ago

# Routing

# Next.js

**Pages Router**

***Before Next.js 13***

**App Router**

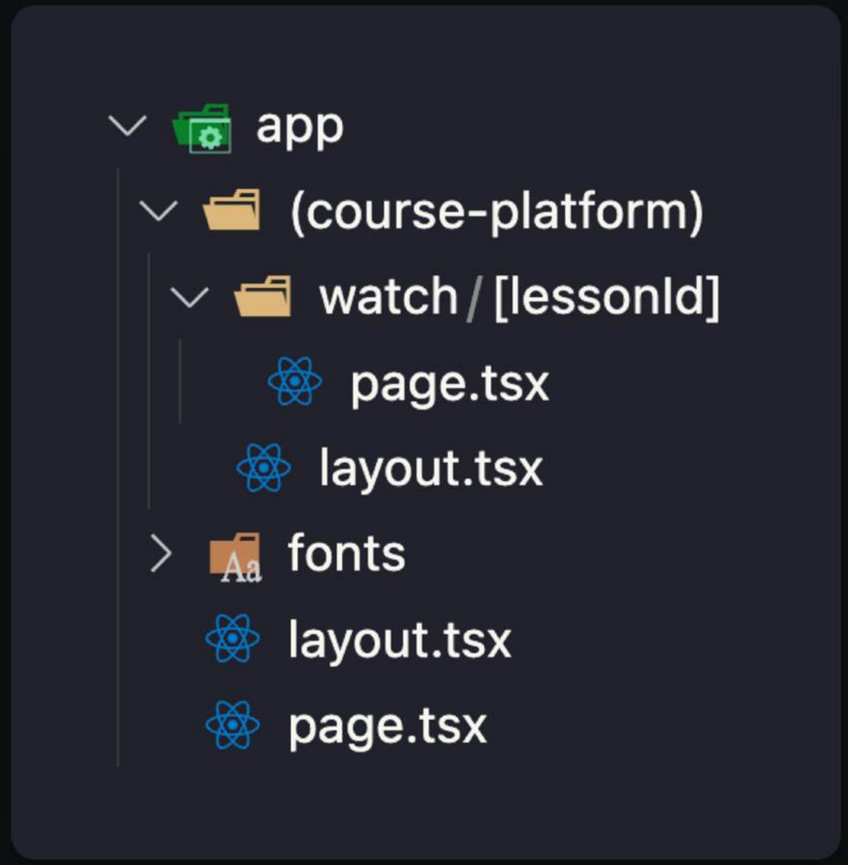
***After Next.js 13***

# Next.js

**Pages Router**

***Before Next.js 13***

# Next.js



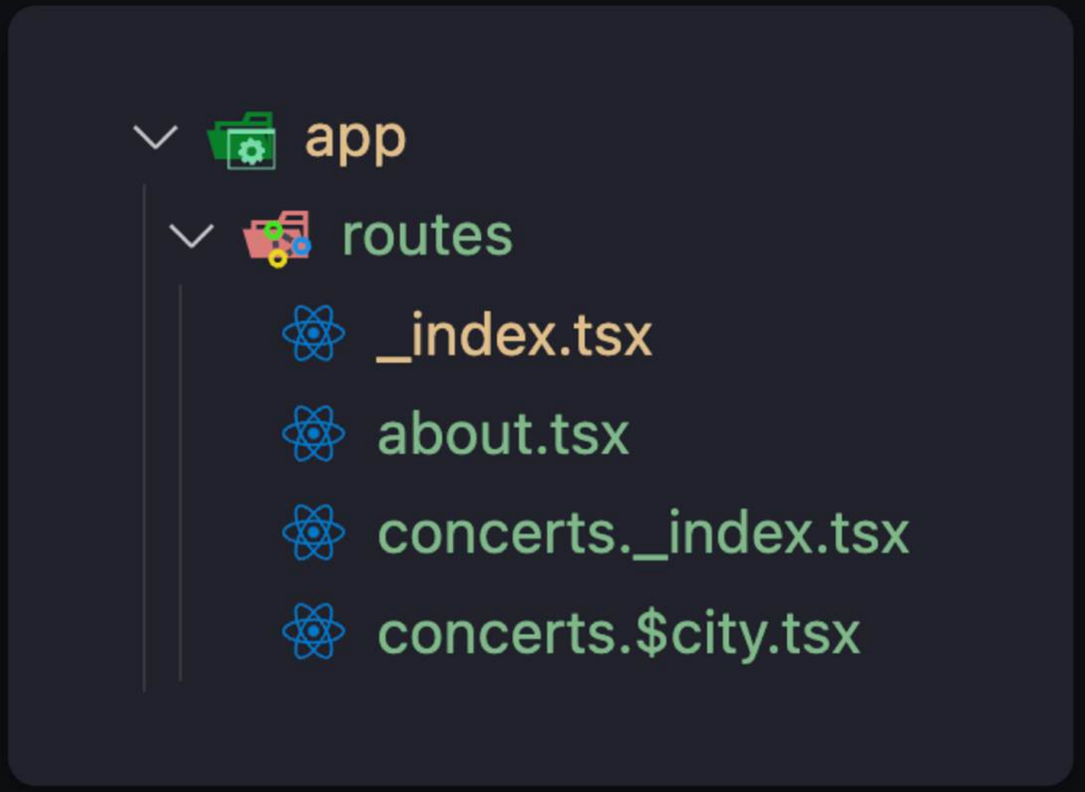

**App Router**

***After Next.js 13***

# Remix

**Pages Router**

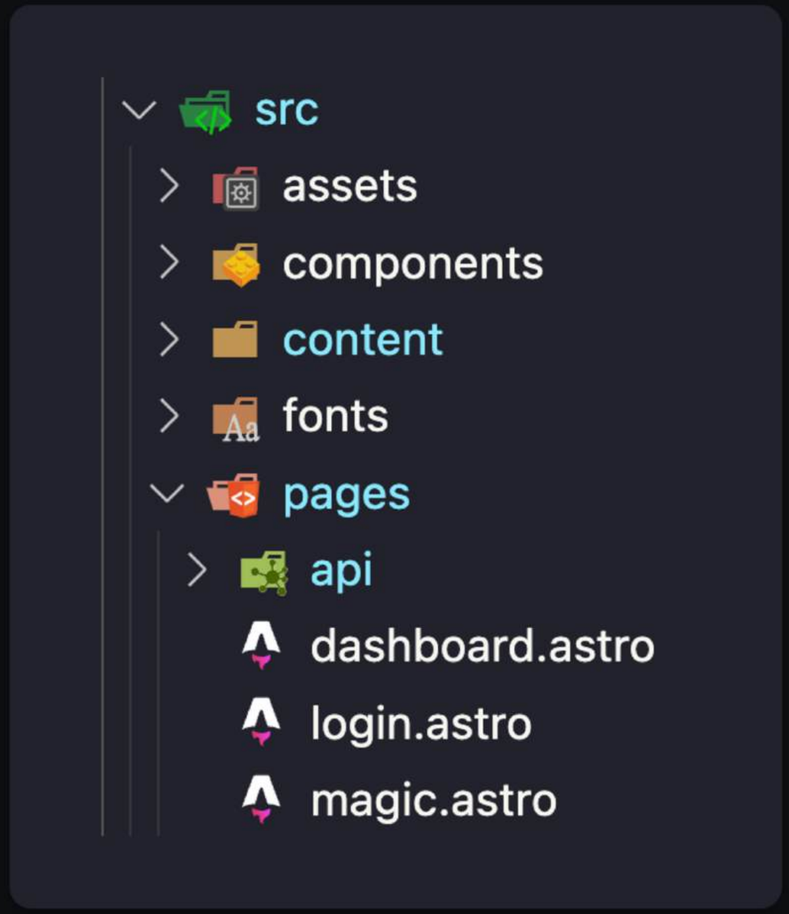
***Before Next.js 13***



# Astro

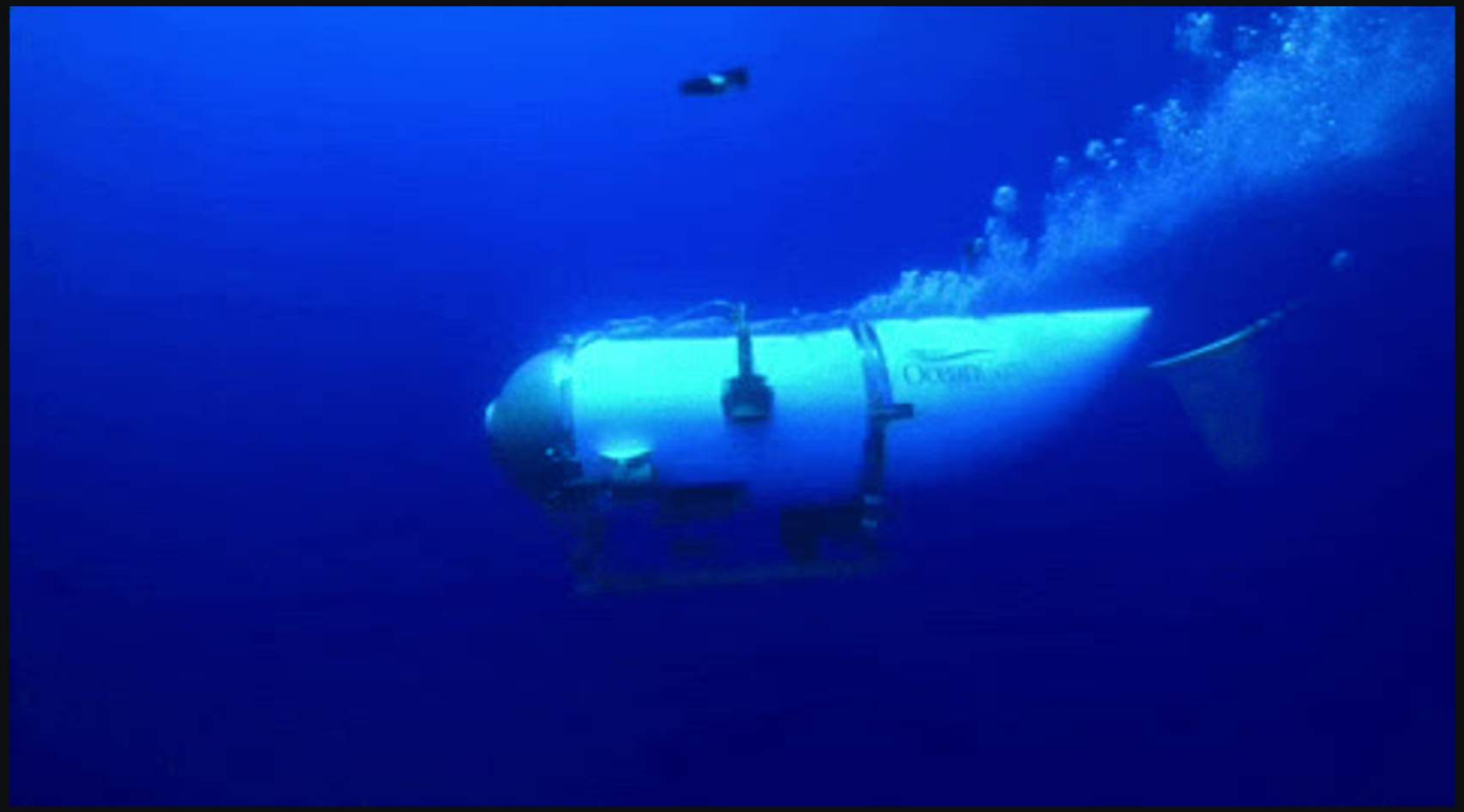
**Pages Router**

***Before Next.js 13***



# Developer Experience

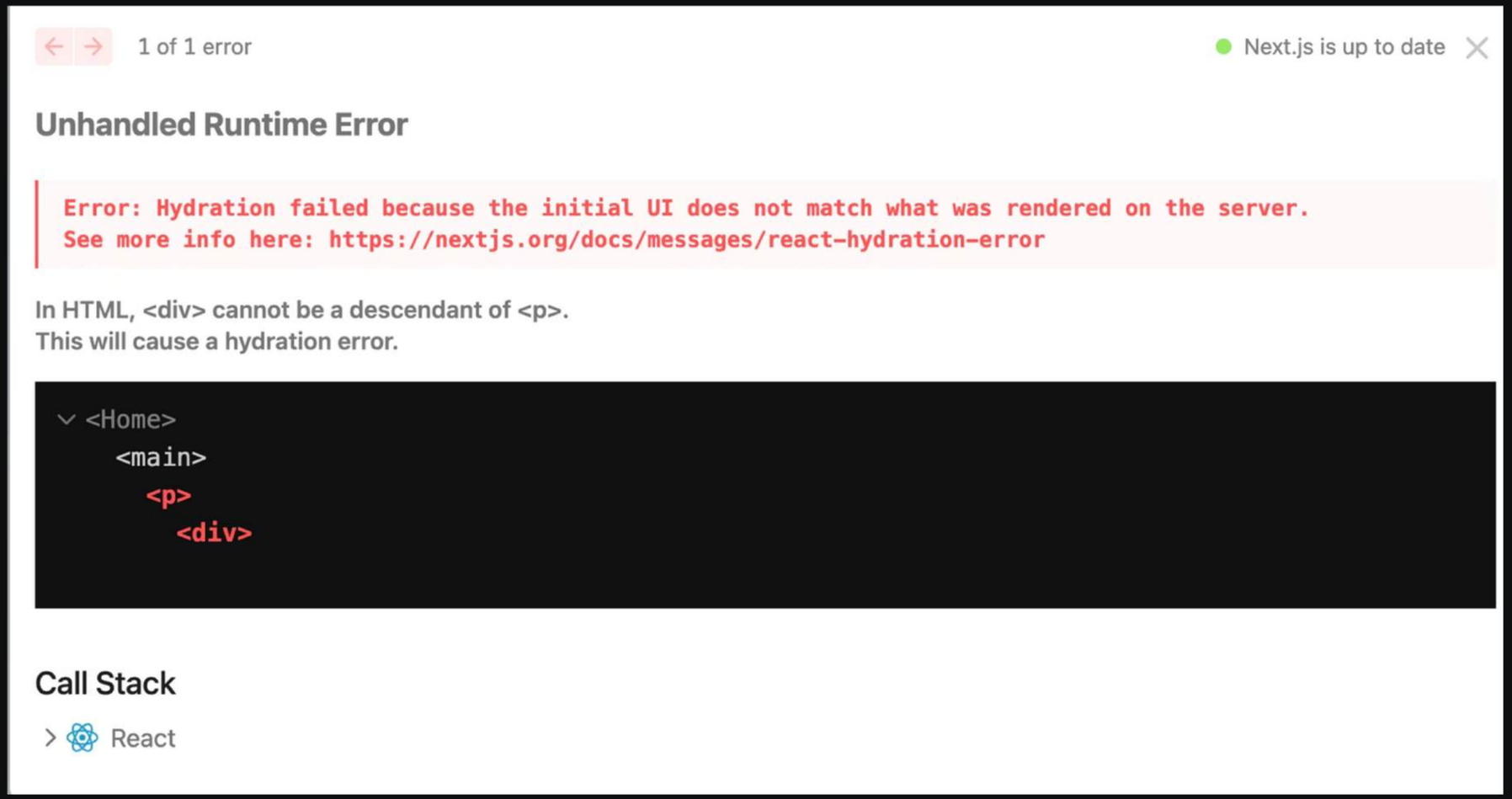
# NEXT.JS 13 APP ROUTER



bit.ly/reactrallylinks

@kulkarniankita9 @kulkarniankita

# Things are way better in Next 14



# Remix is way better

# Remix itself is just a Vite plugin 👉

vite.config.ts

```ts
import { vitePlugin as remix } from "@remix-run/dev";
import { defineConfig } from "vite";
import tsconfigPaths from "vite-tsconfig-paths";

export default defineConfig({
  plugins: [
    remix({
      ignoredRouteFiles: ["**/*.css"],
    }),
    tsconfigPaths(),
  ],
});
```

Astro is just incredible from onboarding, to docs, helping developers etc.

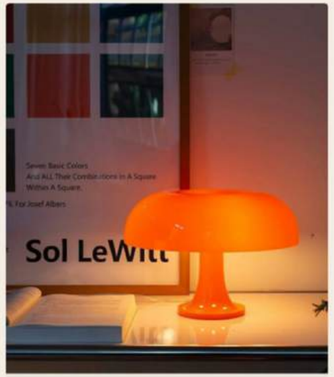
# Performance

# E-COMMERCE APP

EASY SELL UPLOAD >

## OUR TOP PRODUCTS

You can pay to boost your products here.




MUSHROOM ORANGE LAMP

WHETHER YOU'RE LOOKING TO AD...

$49

JEEP WRANGLER

EMBARK ON YOUR NEXT ADVENTU...

$7900

SLICK SNEAKERS PREMIUM

✨ LIMITED EDITION DESIGNS: EACH...

$4500

## ALL PRODUCTS

GRAY COUCH

TRANSFORM YOUR LIVING SPACE WIT...

$800


RED NIKE SHOES

DON'T JUST WEAR SHOES – MAKE A S...

$98

RAINBOW SILICONE HEAT INSULATION PAD

SOFT TABLE MAT, HEAT RESISTANT SI...

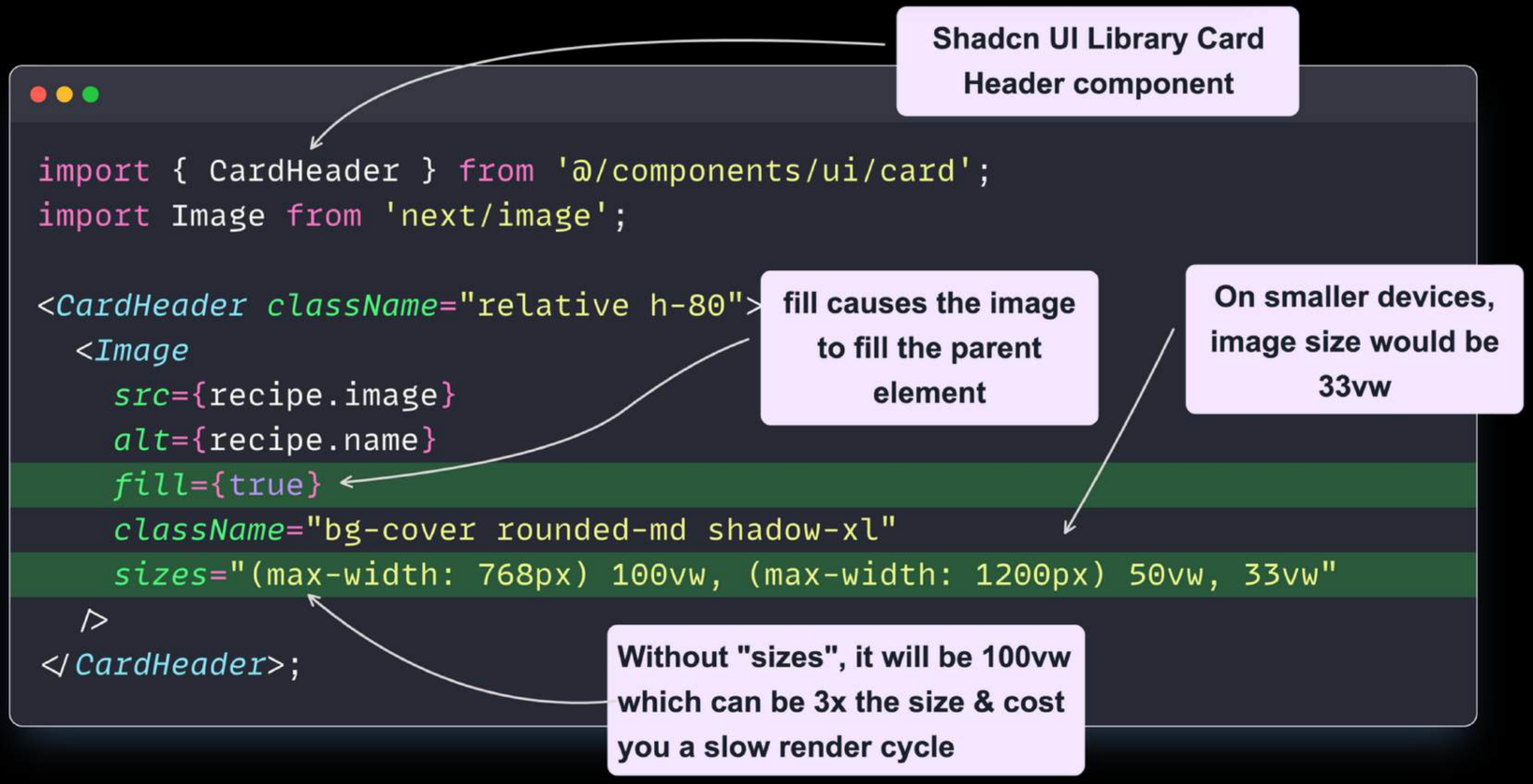
$10


MUSHROOM ORANGE LAMP

WHETHER YOU'RE LOOKING TO ADD ...

$49

# <Image /> in Next.js



# There is more...

*Image Component* 🌠

*Script* 📜

*Link* 🔗

*Fonts* 🖍️

*Granular caching controls* 🚀

*Automatic code splitting* 🧩

bit.ly/reactrallylinks

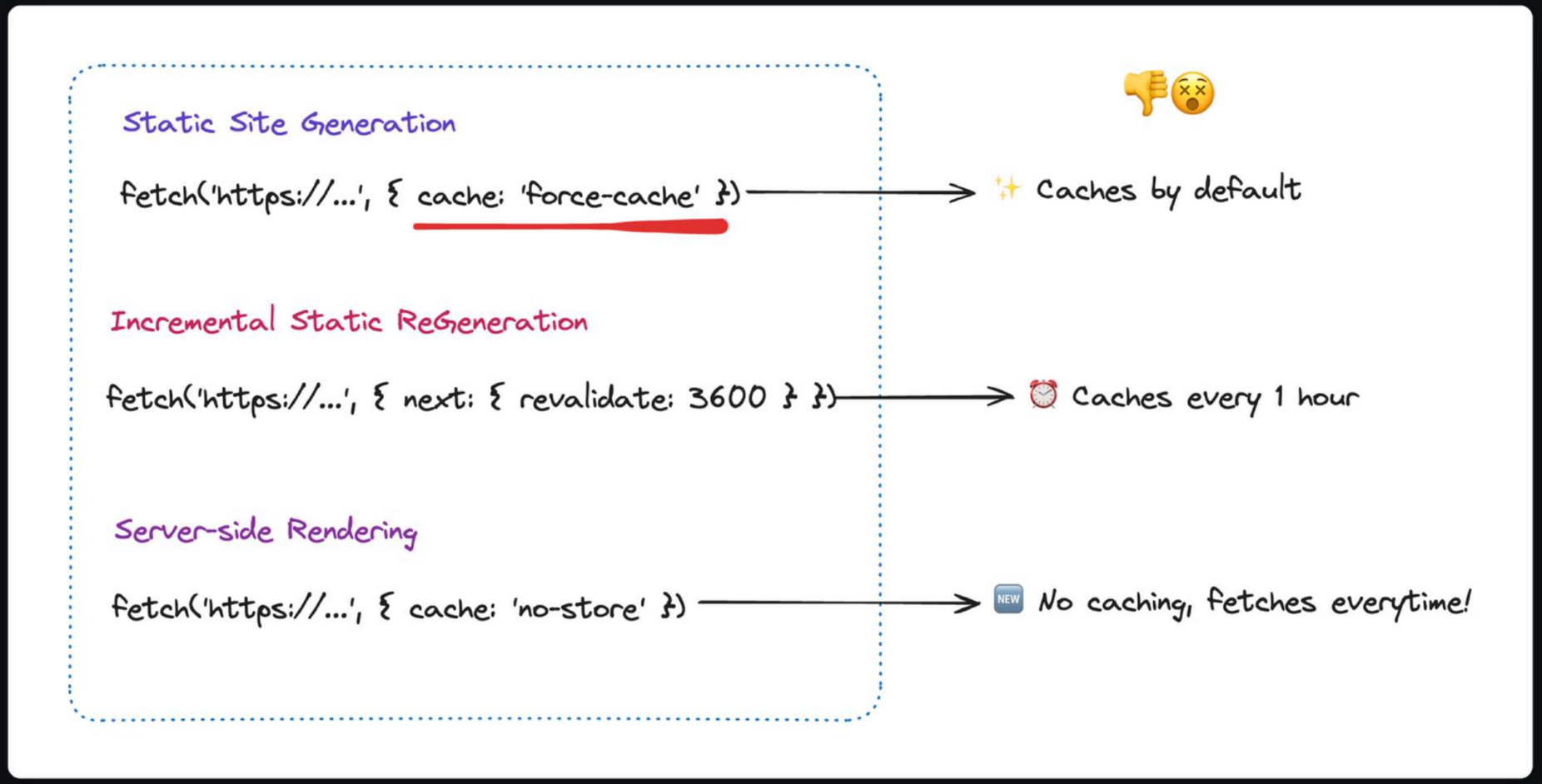
@kulkarniankita9 @kulkarniankita

# Link Pre-fetching

## All 3 frameworks enable instant transitions

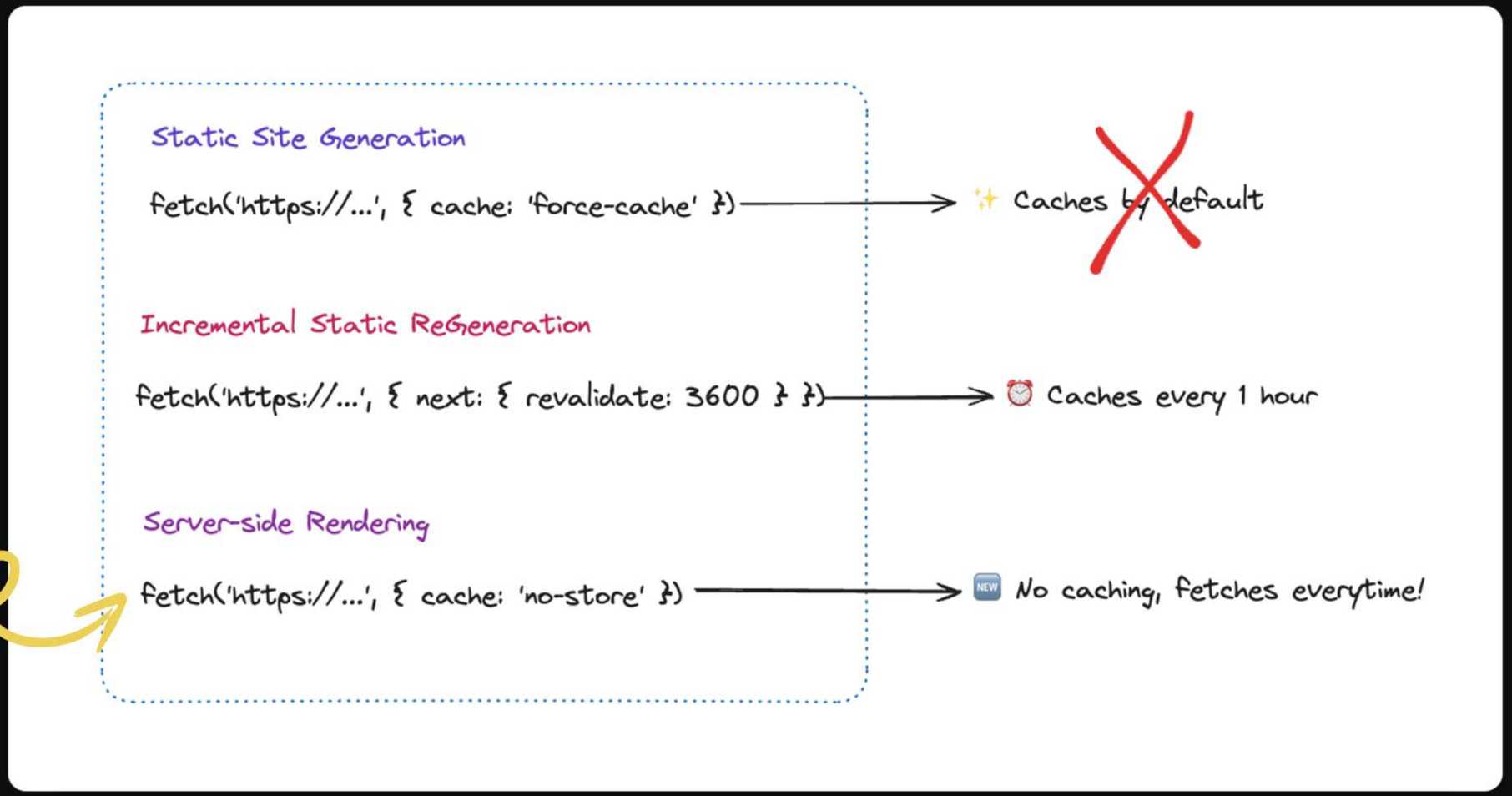
# Data Fetching

# NEXT.JS HAS EXTENDED FETCH




they are changing this in Next.js 15 🎉

# NEXT.JS HAS EXTENDED FETCH

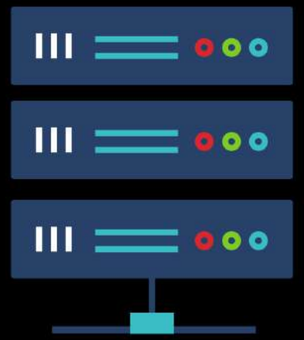



new default

they are changing this in Next.js 15

# NEXT.JS

Server components for fetch ⭐



Server actions for mutations

**Next.js**

page.tsx

```tsx
async function getData() {
  const res = await fetch('https://nextjscourse.dev/invoices',
  { ...authHeaders });
  return res.json();
}

export default async function Page() {
  const { invoices } = await getData()
  return <main>Num of invoices:{invoices.length}</main>
}
```

page.tsx

```tsx
export default function Page() {
  async function invoiceAction(formData: FormData) {
    'use server'
    const amount = formData.get('amount');
    await createInvoiceInDb({ amount }); //mutate data
    revalidateData('/invoices'); // invalidate cache
  }
  return <form action={invoiceAction}>...</form>
}
```

# REMIX


MJ
@mjackson · Follow
Replying to @mjackson
You may be thinking: that's cheating, Michael! You should've built an abstraction around `wget` and taunted it as a new feature of Remix. 😤
But `wget` is *the original* static site generator! With logging, cookies, base URL paths, progress... it's pretty nice! 🙌
9/12
12:50 AM · Oct 28, 2022

routes/projects.ts **Component**

```tsx
// Component
export default function Projects() {
  const projects = useLoaderData<typeof loader>();
  const actionData = useActionData<typeof action>();

  return (
    <div>
      {projects.map((project) => (
        <Link key={project.slug} to={project.slug}>
          {project.title}
        </Link>
      ))}
      <Form method="post">
        <input name="title" />
        <button type="submit">Create New Project</button>
      </Form>
      {actionData?.errors ? (
        <ErrorMessages errors={actionData.errors} />
      ) : null}
    </div>
  );
}
```

routes/projects.ts **Loader**

```tsx
// Loaders only run on the server and provide
// data to your component on GET requests
export async function loader() {
  return json(await db.projects.findAll());
}

// Previous Project Component
```

routes/projects.ts **Actions**

```tsx
// Actions only run on the server and handle POST
// PUT, PATCH, and DELETE. They can also provide data
// to the component
export async function action({
  request,
}: ActionFunctionArgs) {
  const form = await request.formData();
  const errors = validate(form);
  if (errors) {
    return json({ errors });
  }
  await createProject({ title: form.get("title") });
  return json({ ok: true });
}
```

routes/projects.ts **Component**

```tsx
// Component
export default function Projects() {
  const projects = useLoaderData<typeof loader>();
  const actionData = useActionData<typeof action>();

  return (
    <div>
      {projects.map((project) => (
        <Link key={project.slug} to={project.slug}>
          {project.title}
        </Link>
      ))}
      <Form method="post">
        <input name="title" />
        <button type="submit">Create New Project</button>
      </Form>
      {actionData?.errors ? (
        <ErrorMessages errors={actionData.errors} />
      ) : null}
    </div>
  );
}
```

routes/projects.ts **Loader**

```tsx
// Loaders only run on the server and provide
// data to your component on GET requests
export async function loader() {
  return json(await db.projects.findAll());
}

// Previous Project Component
```

routes/projects.ts **Actions**

```tsx
// Actions only run on the server and handle POST
// PUT, PATCH, and DELETE. They can also provide data
// to the component
export async function action({
  request,
}: ActionFunctionArgs) {
  const form = await request.formData();
  const errors = validate(form);
  if (errors) {
    return json({ errors });
  }
  await createProject({ title: form.get("title") });
  return json({ ok: true });
}
```

routes/projects.ts **Component**

```tsx
// Component
export default function Projects() {
  const projects = useLoaderData<typeof loader>();
  const actionData = useActionData<typeof action>();

  return (
    <div>
      {projects.map((project) => (
        <Link key={project.slug} to={project.slug}>
          {project.title}
        </Link>
      ))}
      <Form method="post">
        <input name="title" />
        <button type="submit">Create New Project</button>
      </Form>
      {actionData?.errors ? (
        <ErrorMessages errors={actionData.errors} />
      ) : null}
    </div>
  );
}
```

routes/projects.ts **Loader**

```tsx
// Loaders only run on the server and provide
// data to your component on GET requests
export async function loader() {
  return json(await db.projects.findAll());
}

// Previous Project Component
```

routes/projects.ts **Actions**

```tsx
// Actions only run on the server and handle POST
// PUT, PATCH, and DELETE. They can also provide data
// to the component
export async function action({
  request,
}: ActionFunctionArgs) {
  const form = await request.formData();
  const errors = validate(form);
  if (errors) {
    return json({ errors });
  }
  await createProject({ title: form.get("title") });
  return json({ ok: true });
}
```

routes/projects.ts **Component**

```tsx
// Component
export default function Projects() {
  const projects = useLoaderData<typeof loader>();
  const actionData = useActionData<typeof action>();

  return (
    <div>
      {projects.map((project) => (
        <Link key={project.slug} to={project.slug}>
          {project.title}
        </Link>
      ))}
      <Form method="post">
        <input name="title" />
        <button type="submit">Create New Project</button>
      </Form>
      {actionData?.errors ? (
        <ErrorMessages errors={actionData.errors} />
      ) : null}
    </div>
  );
}
```

routes/projects.ts **Loader**

```tsx
// Loaders only run on the server and provide
// data to your component on GET requests
export async function loader() {
  return json(await db.projects.findAll());
}

// Previous Project Component
```

routes/projects.ts **Actions**

```tsx
// Actions only run on the server and handle POST
// PUT, PATCH, and DELETE. They can also provide data
// to the component
export async function action({
  request,
}: ActionFunctionArgs) {
  const form = await request.formData();
  const errors = validate(form);
  if (errors) {
    return json({ errors });
  }
  await createProject({ title: form.get("title") });
  return json({ ok: true });
}
```

# ASTRO


Theo - t3.gg
@t3dotgg
Rebuilt my (next.js) blog using @astrodotbuild out of curiosity...holy shit the difference in bundle size.
Home route: 138kb -> 7.6kb
"All posts": 570kb -> 100kb (85kb was images)
Seriously considering moving my static sites to this
T3
Theo Browne - T3 Stack, Ping.gg, YouTuber
From astro.t3.gg

# ASTRO

## Uses fetch Web API's

about.astro

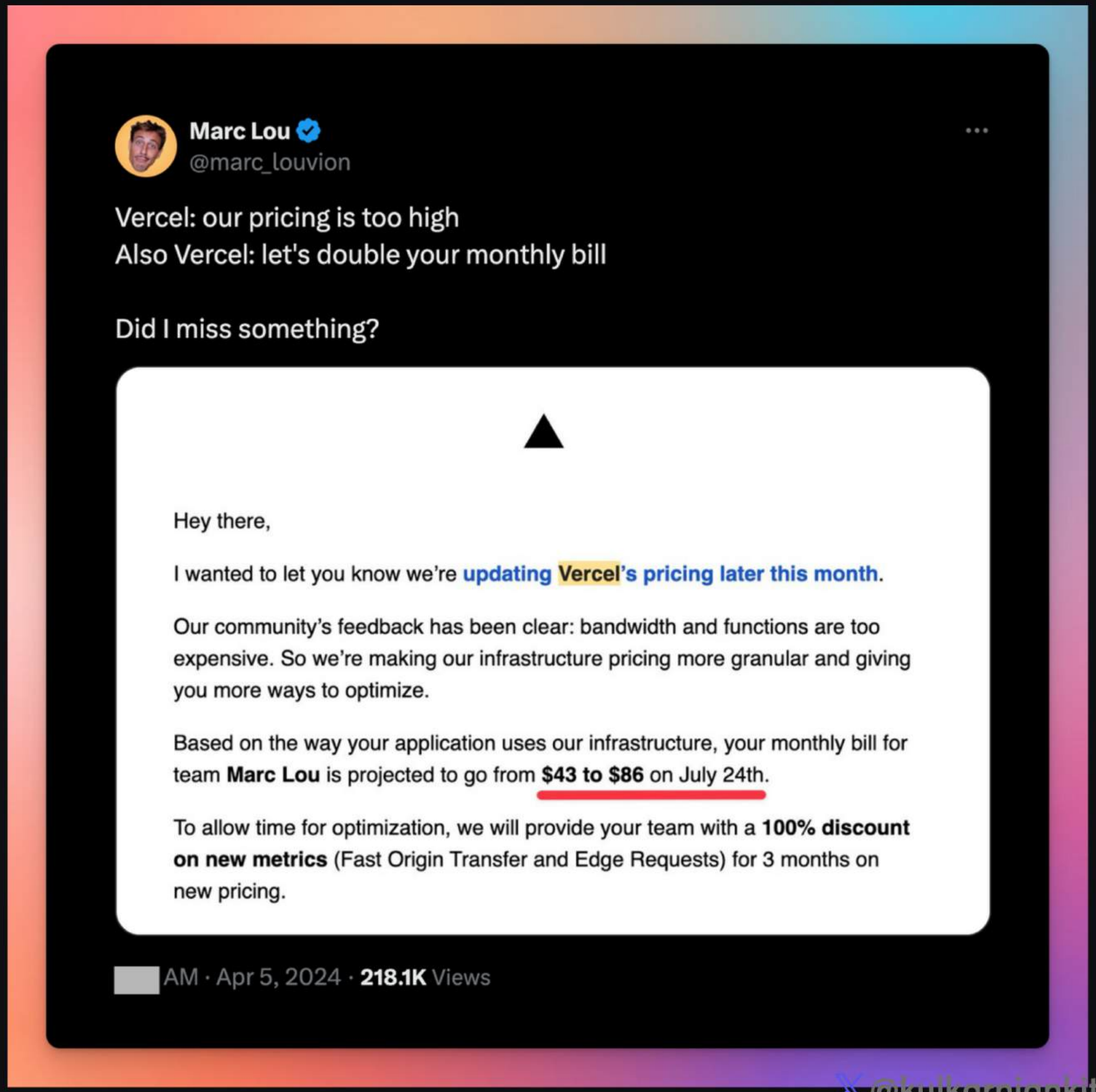
```
---
import Contact from '../components/Contact.jsx';

const response = await fetch('https://randomuser.me/api/');
const data = await response.json();
const randomUser = data.results[0];
---
<!— Data fetched at build can be rendered in HTML —>
<h1>User</h1>
<h2>{randomUser.name.first} {randomUser.name.last}</h2>

<!— Data fetched at build can be passed to components as prop
<Contact client:load email={randomUser.email} />
```

# Deployment

# NEXT.JS

# NEXT.JS


tightly coupled


Marc Lou
@marc_louvion
Vercel: our pricing is too high
Also Vercel: let's double your monthly bill
Did I miss something?
Hey there,
I wanted to let you know we're updating Vercel's pricing later this month.
Our community's feedback has been clear: bandwidth and functions are too expensive. So we're making our infrastructure pricing more granular and giving you more ways to optimize.
Based on the way your application uses our infrastructure, your monthly bill for team Marc Lou is projected to go from $43 to $86 on July 24th.
To allow time for optimization, we will provide your team with a 100% discount on new metrics (Fast Origin Transfer and Edge Requests) for 3 months on new pricing.
AM · Apr 5, 2024 · 218.1K Views

# But, all my products are on Vercel...

# IT JUST WORKS!!

# REMIX: CAN RUN ANYWHERE ✨

fly.io

netlify

cloudflare

vercel

others

# ASTRO

## Static


## Dynamic

# ASTRO

## Static


## Dynamic


bit.ly/reactrallylinks

@kulkarniankita9 @kulkarniankita

# CHOOSE FRAMEWORK WHEN

# CHOOSE NEXT.JS WHEN

01

BUILD-IN OPTIMIZATIONS, USE SERVER COMPONENTS

02

NEED GRANULAR CACHING CONTROLS

03

NO CUSTOM WEBPACK BUILDS

# CHOOSE REMIX WHEN

01

LOTS OF DYNAMIC PAGES

02

SUPPORT FOR WEB APIS

03

NO LIMITATION FOR INTEGRATION TO CLOUD PLATFORMS

# CHOOSE ASTRO WHEN

01

BUILDING CONTENT-DRIVEN WEBSITES

02

LOADS FAST & GREAT SEO

03

EASILY ACCESSIBLE TO ALL DEVELOPERS IRRESPECTIVE OF LIBRARIES

# NEXT.JS

ADOPTION

DEVELOPER EXPERIENCE

PERFORMANCE

ROUTING

DATA FETCHING

DEPLOYMENT

# REMIX

ADOPTION

DEVELOPER EXPERIENCE

PERFORMANCE

ROUTING

DATA FETCHING

DEPLOYMENT

# ASTRO

ADOPTION

DEVELOPER EXPERIENCE

PERFORMANCE

ROUTING

DATA FETCHING

DEPLOYMENT

WINNER 🥇

# Remix

WINNER 🥇

# it depends....

# BIT.LY/REACTRALLYLINKS

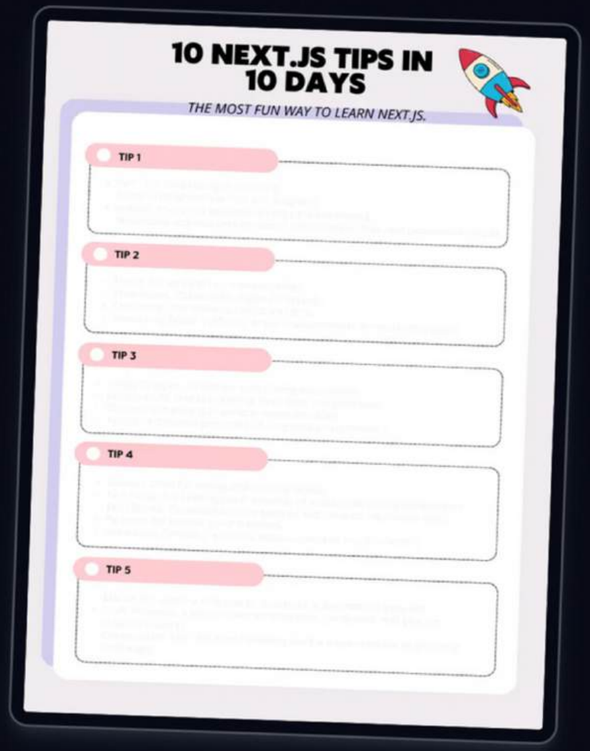
🔥 FREE: Email Course

## Level up your Next.js skills w/ 10 digestible tips

*The most fun way to learn Next.js. 10 easy-to-follow tips in 10 days!*

Get Lesson 1

Bonus: FREE access to Next.js code repository with examples.




*Plus, get a downloadable guide at the end of the course with all 10 tips.*

# Thank you! 🙌

ALL MY SLIDES ARE AT
BIT.LY/REACTRALLYLINKS