Is **Remix** your **next framework**

Remix

Blog Docs GitHub

Focused on web standards and modern web app UX, you're simply going to build better websites

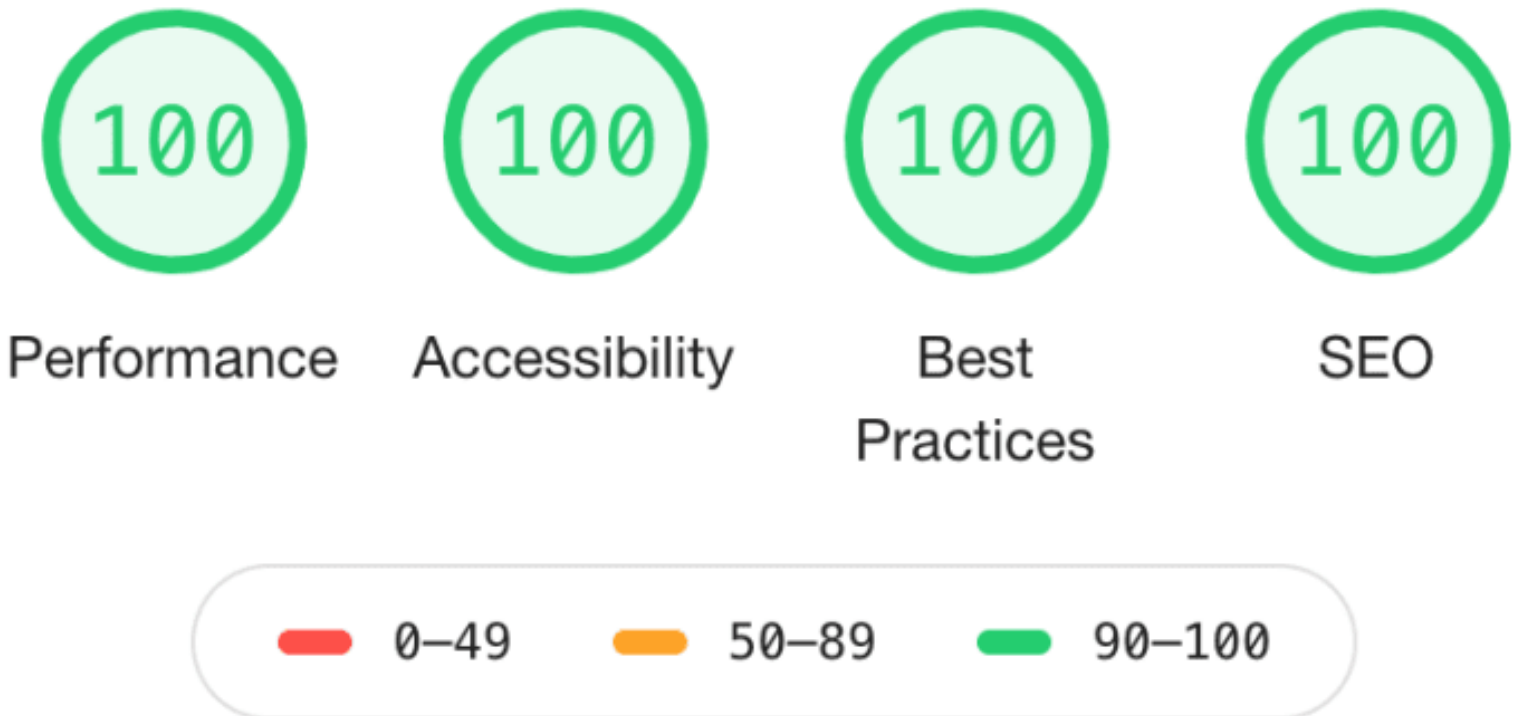
Remix is a full stack web framework that lets you focus on the user interface and work back through web standards to deliver a fast, slick, and resilient user experience. People are gonna love using your stuff.

Read the Docs

Get Started

```
export async function loader({ request }) {
  return getProjects();
}

export async function action({ request }) {
  const form = await request.formData();
  return createProject({ title: form.get("title") });
}

export default function Projects() {
  const projects = useLoaderData();
  const { state } = useTransition();
  const busy = state === "submitting";

  return (
    <div>
      {projects.map((project) => (
        <Link to={project.slug}>{project.title}</Link>
      ))}

      <Form method="post">
        <input name="title" />
        <button type="submit" disabled={busy}>
          {busy ? "Creating..." : "Create New Project"}
        </button>
      </Form>
    </div>
  );
}
```

Reaching the **goal**



SAME OLD PROBLEMS

NEW SOLUTIONS

SSR

SSG

ISR

Remix vs NextJS

Search Page, Cached, Virginia, Cable

https://remix.run/blog/remix-vs-next
https://demo.vercel.store/

```
4   export default function Projects() {
5     const projects = useLoaderData();
6     const { state } = useTransition();
7     const busy = state === "submitting";
8
9     return (
10      <div>
11        {projects.map((projects) => (
12          <Link to={projects.slug} key={projects.id}>
13            {projects.title}
14          </Link>
15        ))}
16        <Form method="post">
17          <input name="title" />
18          <button type="submit" disabled={busy}>
19            {busy ? "Saving..." : "Save"}
20          </button>
21        </Form>😀
22      </div>
23    )
24  }
25
```

Using what we **know**

Without
the **hard parts**

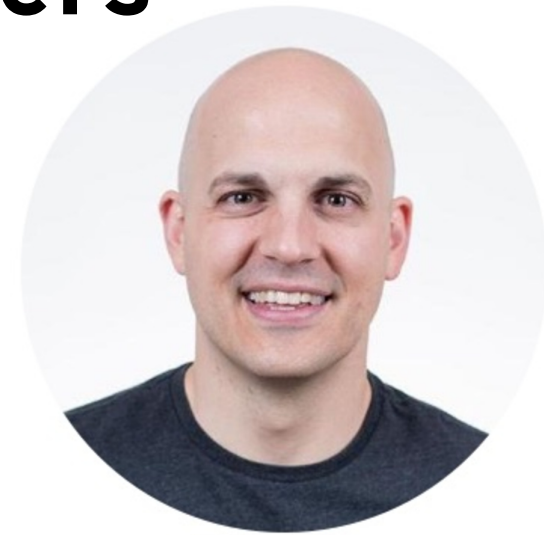
React without the **hard parts**

Learn **Remix**,
Accidentally Learn the **Web**

Remix is driven by **community leaders**

Kent C Dodds

Ryan Florence



Michael Jackson

Remix is about

- Developer friendliness
- Developer productivity
- Performance & speed
- Using web standard

Lets dive in Remix

ROUTING

Defining routes

```
app
├── root.jsx
└── routes
    ├── accounts.jsx
    ├── dashboard.jsx
    ├── expenses.jsx
    ├── index.jsx
    ├── reports.jsx
    ├── sales
    │   ├── customers.jsx
    │   ├── deposits.jsx
    │   ├── index.jsx
    │   ├── invoices
    │   │   ├── $invoiceId.jsx
    │   │   └── index.jsx
    │   ├── invoices.jsx
    │   └── subscriptions.jsx
    └── sales.jsx
```

```
/sales/invoices/102000

- root.jsx
- routes/sales.jsx
- routes/sales/invoices.jsx
- routes/sales/invoices/$invoiceId.jsx
```

```
<Root>
  <Sales>
    <Invoices>
      <InvoiceId />
    </Invoices>
  </Sales>
</Root>
```

DATA LOADING

How can we describe performance

Time to First Byte (TTFB)

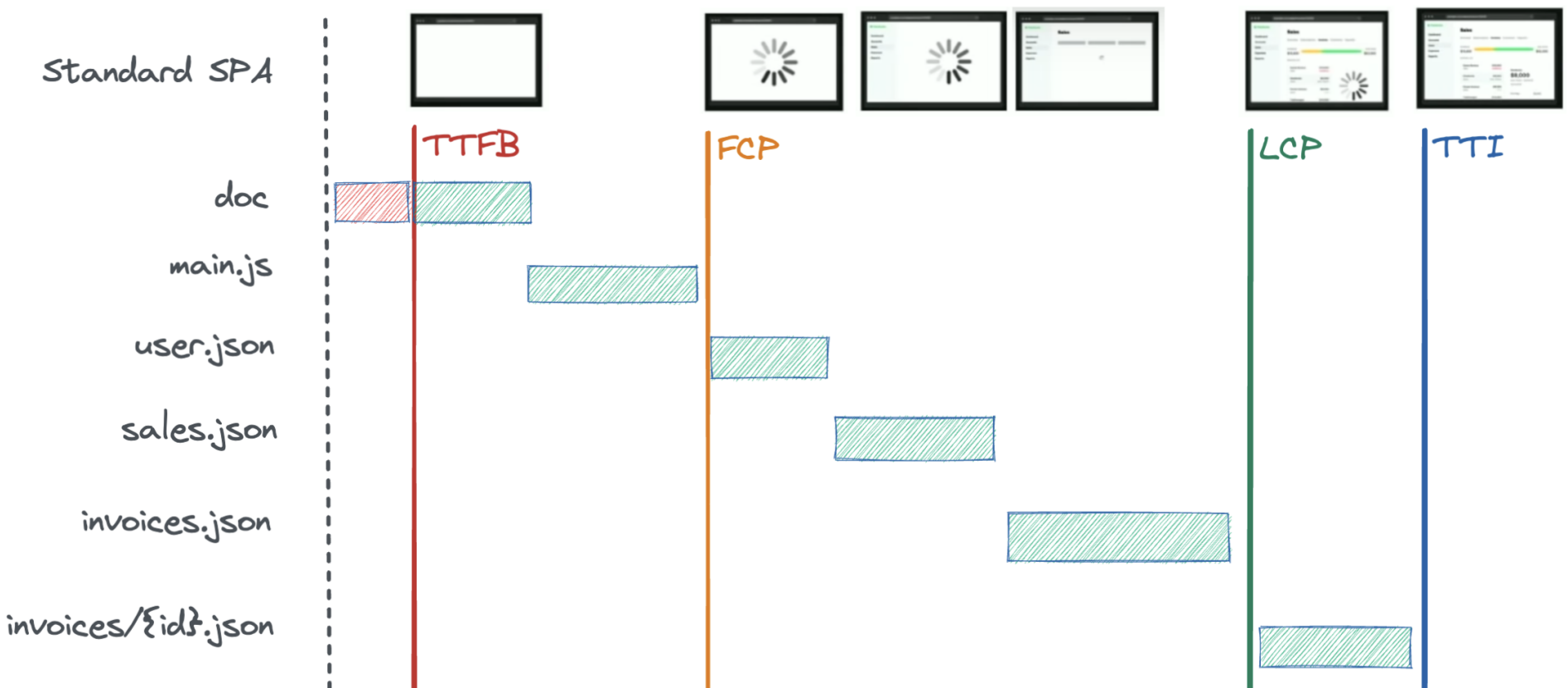
First Contentful Paint (FCP)

Cumulative Layout Shift (CLS)

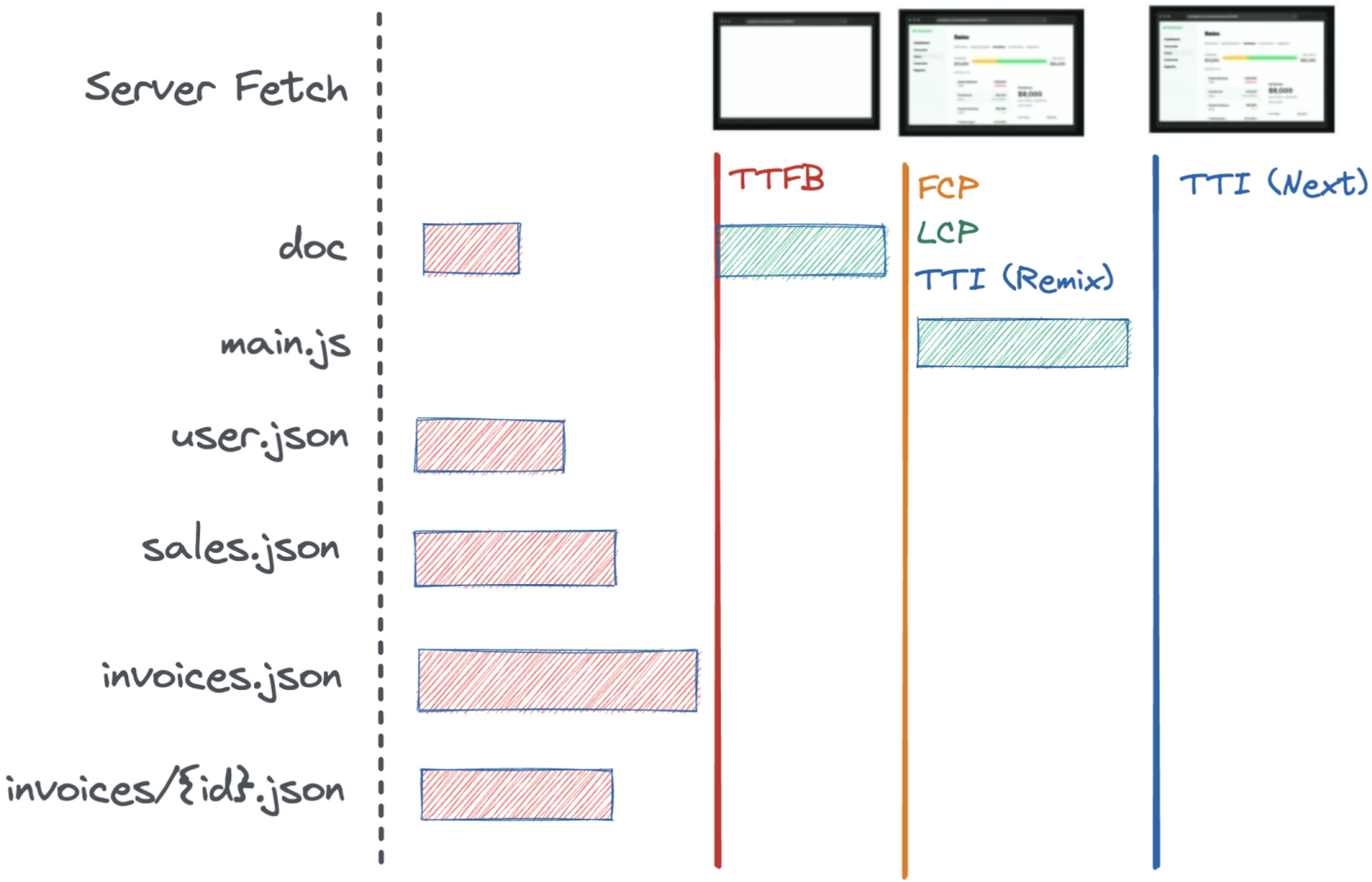
Largest Contentful Paint (LCP)

Time To Interactive (TTI)

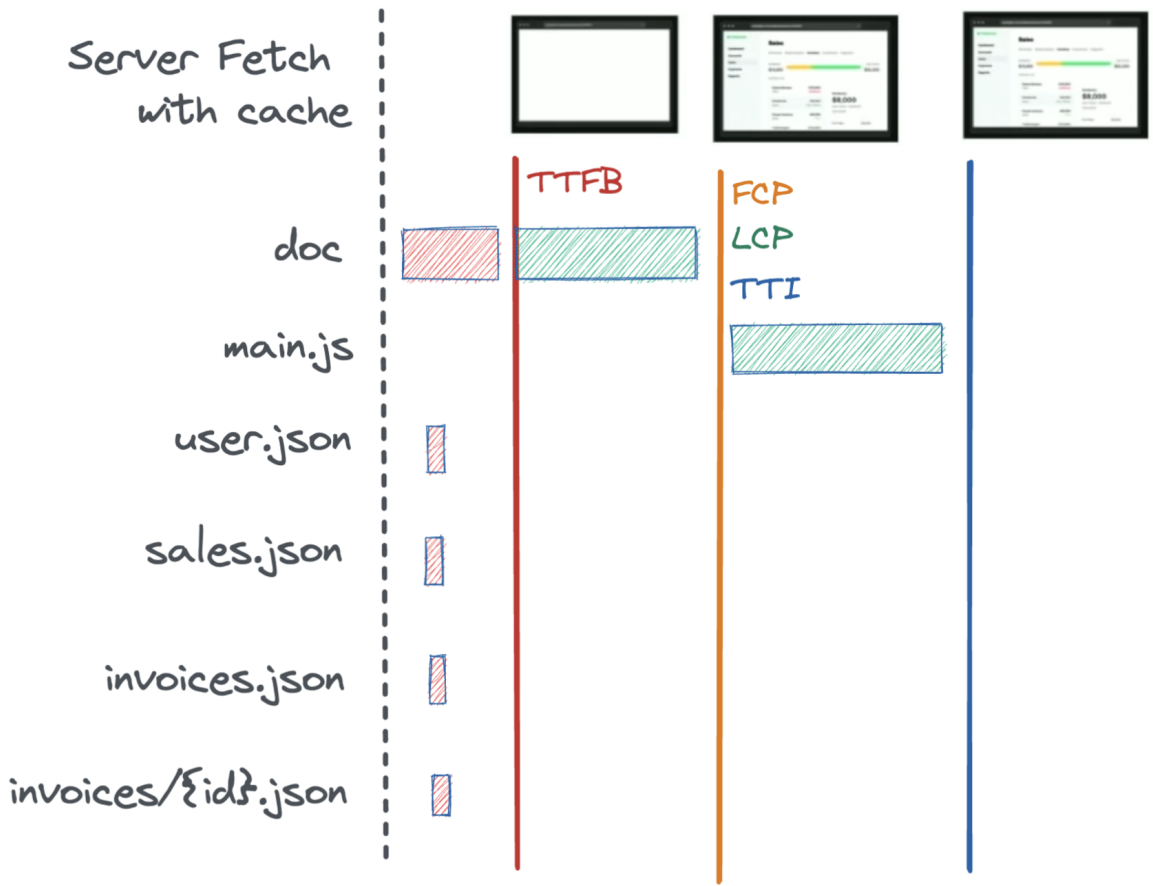
Rendering of a standard SPA



Server side rendering



Server side rendering with caching



Route with data fetching

```tsx
import type { LoaderFunction } from "@remix-run/node"; // or cloudflare/deno
import { json } from "@remix-run/node"; // or cloudflare/deno
import { useLoaderData } from "@remix-run/react";
import { db } from "~/db.server";

export const loader: LoaderFunction = async ({ params }) => {
  return json(
    await db.product.findMany({
      where: { categoryId: params.categoryId },
    })
  );
};

export default function ProductCategory() {
  const products = useLoaderData();
  return (
    <div>
      <p>{products.length} Products</p>
      {/* ... */}
    </div>
  );
}
```

FORM ACTIONS

React Form

```
function MyForm() {
  const [inputs, setInputs] = useState({});

  const handleChange = (event) => {
    const name = event.target.name;
    const value = event.target.value;
    setInputs(values => ({...values, [name]: value}))
  }

  const handleSubmit = (event) => {
    event.preventDefault();
    console.log(inputs);
  };

  return (
    <form onSubmit={handleSubmit}>
      <label>Enter your name:
        <input type="text"
               value={inputs.username || ""}
               onChange={handleChange}
        />
      </label>
    </form>
  )
}
```

The **power** of **Remix**

GET /users

```
<a href="/users">Users</a>
```

GET /users?name=peter

```
<form action="/users">Users</form>
```

Query (form get)

```tsx
import type { LoaderArgs } from "@remix-run/node"; // or cloudflare/deno
import { redirect } from "@remix-run/node"; // or cloudflare/deno

export const loader = async ({ request }: LoaderArgs) => {
  const url = new URL(request.url);
  const search = new URLSearchParams(url.search);
  const query = search.get('query') || '';
  const data = [....];
  return json({ query, data });
};

export default function Users() {
  const { data, query } = useLoaderData();
  return (
    <h1>Search page</h1>
    <form method='get'>
      <input type='text' name='query' defaultValue={query}/>
    </form>
  )
}
```

Mutation (form post)

```tsx
import type { ActionArgs } from "@remix-run/node"; // or cloudflare/deno
import { redirect } from "@remix-run/node"; // or cloudflare/deno

export const action = async ({ request }: ActionArgs) => {
  const formData = await request.formData();
  const user = await db.save.user(formData);
  return redirect(`/users/${user.id}`);
};

export default function NewUser() {
  return (
    <form method="post" action="/search">
      <input name="name" type="text" />
      <input name="email" type="email" />
      <button type="submit">Submit</button>
    </form>
  )
}
```

Form validation - action

```
// Zod validation schema
export const formSchema = z.object({
  name: z.string(),
  email: z.string().email().optional()
});

export const action = async ({ request }) => {
  const formPayload = Object.fromEntries(await request.formData());
  try {
    const formValues = formSchema.parse(formPayload);
    const user = await db.save.user(formValues);
    return redirect(`/users/${user.id}`);
  } catch (error) {
    return {
      formPayload,
      errors: (error as ZodError).flatten().fieldErrors,
    };
  }
```

Form validation - handle errors

```
1  export default function NewUser() {
2    const actionData = useActionData();
3    return (
4      <form method="get" action="/search">
5        <input name="name" type="text" />
6        {actionData?.errors.name && (
7           <span className="text-red-600">{actionData?.errors.name[0]}</span>
8        )}
9        <input name=email" type="email" />
10       {actionData?.email.name && (
11          <span className="text-red-600">{actionData?.errors.email[0]}</span>
12       )}
13       <button type="submit">Submit</button>
14     </form>
15   )
16 }
```

CACHING

Caching

```
// data cache
export const loader: LoaderFunction = async ({ params }) => {
  const products = await db.product.findMany({
    where: { categoryId: params.categoryId },
  }
  return json(products, {
    "Cache-Control": "public, s-maxage=600",
  });
};
```

```
// route cache
export let headers: HeadersFunction = () => {
  return {
    "Cache-Control": "public, s-maxage=600",
  };
};
```

ERROR HANDLING

Global Error handling

```
1  // root.aspx
2  export function ErrorBoundary({ error }) {
3    return (
4      <html>
5        <head>
6          <title>Oh no!</title>
7          <Meta />
8          <Links />
9        </head>
10       <body className='m-4'>
11         <h1 className='text-2xl'>Something went wrong!</h1>
12         <p>{error.message}</p>
13         <Scripts />
14       </body>
15     </html>
16   );
17 }
```

Route Error handling

```
1  // my route component
2  export default () => {
3    return (
4      <div>...</div>
5    )
6  }
7
8  // Error Boundery Component
9  export function ErrorBoundary({ error }) {
10   return (
11     <div>
12       <h1>Error</h1>
13       <p>{error.message}</p>
14       <p>The stack trace is:</p>
15       <pre>{error.stack}</pre>
16     </div>
17   );
18 }
```

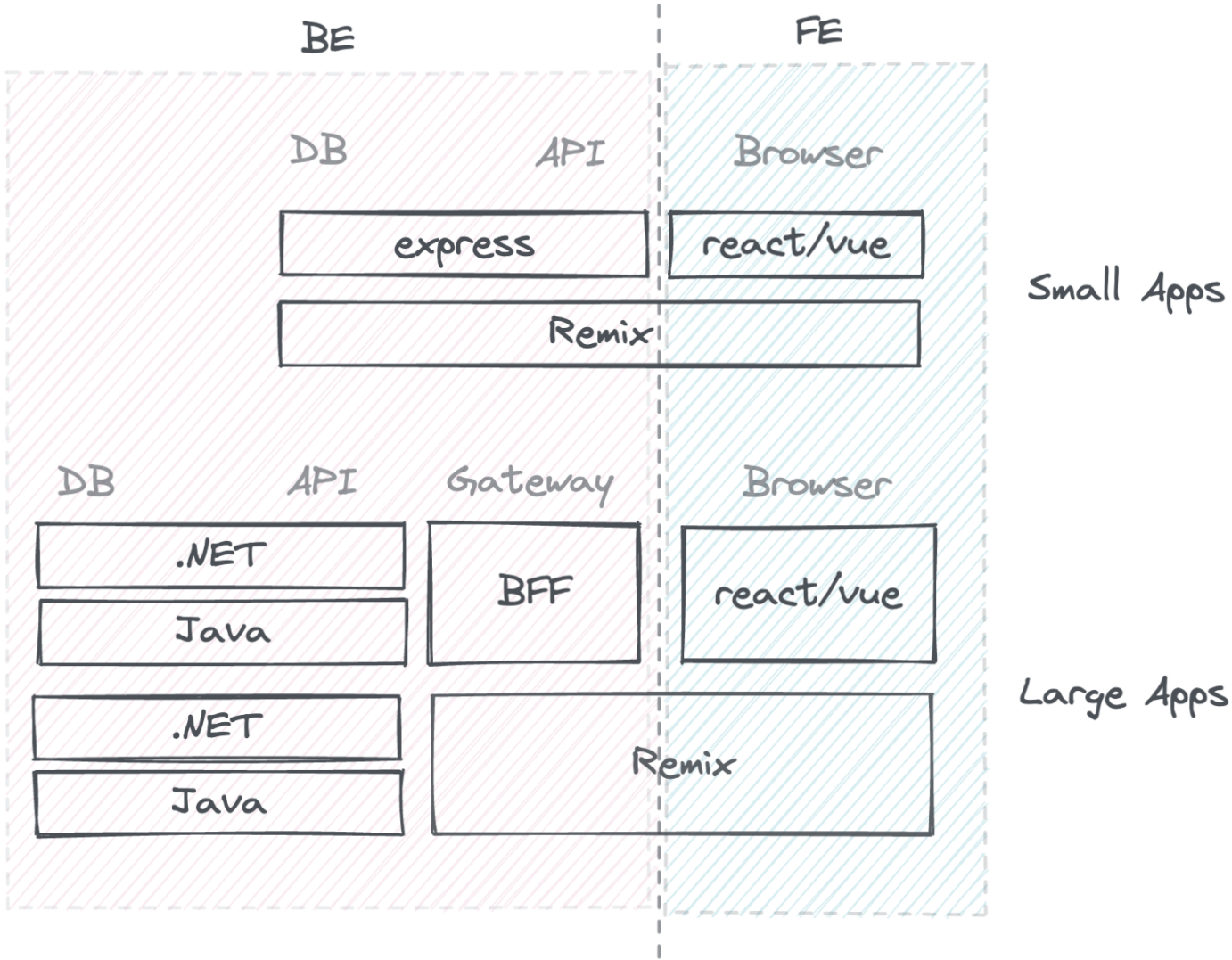
Catch Error (http status errors)

```
// We can throw a 404 error
export const loader = async ({ params }: LoaderArgs) => {
  const id = params.id;
  const category = await getDb().categories.findOne({ _id: new SafeObjectId(id) });
  if (!category) {
    throw json({ status: 404, statusText: 'Not Found' });
  }

  return json({ category });
};
```

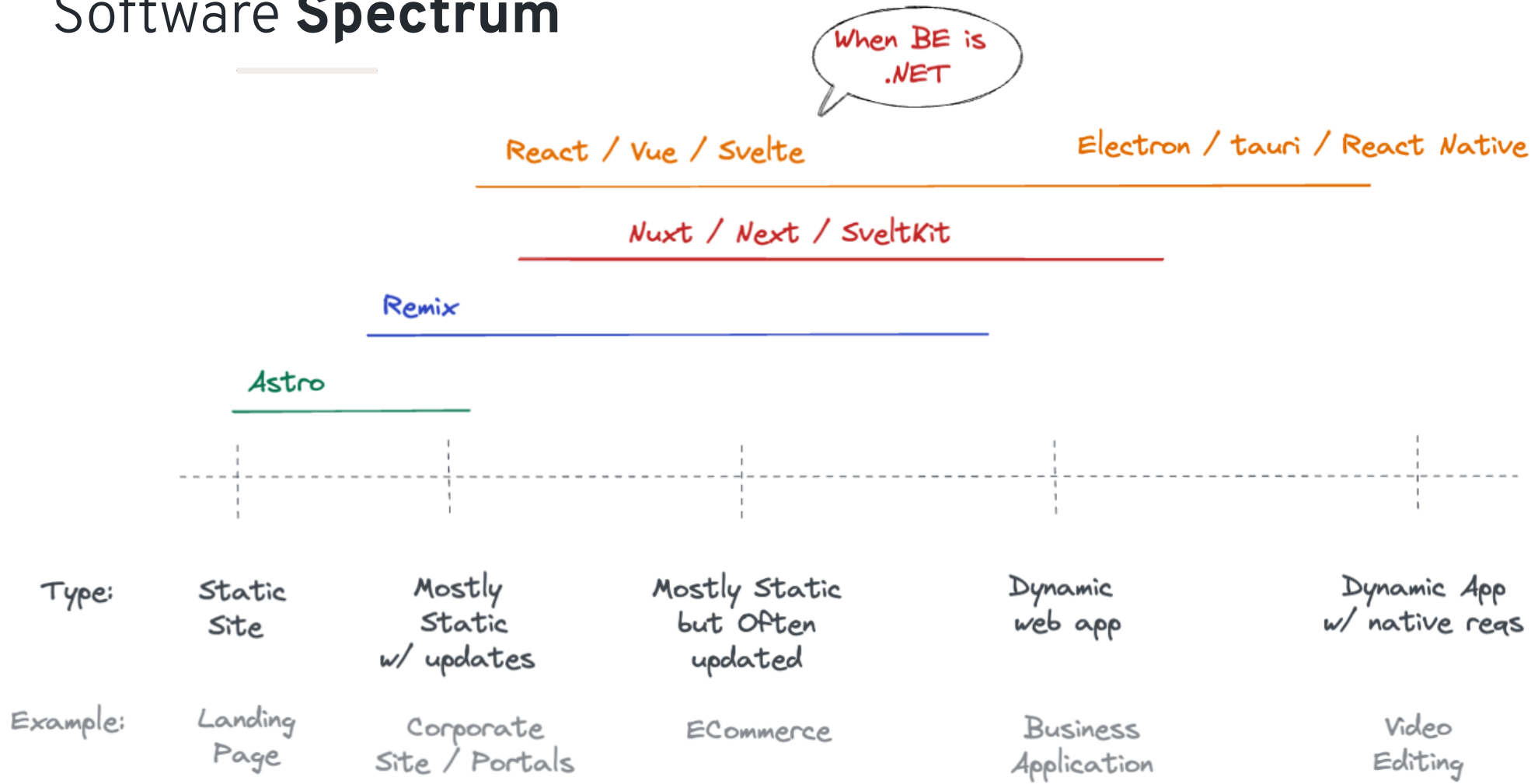
```
// Status Catch Boundery
export function CatchBoundary() {
  const { data } = useCatch();
  return (
    <div>
      <p>Resources not found [{data.status}]</p>
    </div>
  );
}
```

STACK

Software Stack



Software **Spectrum**




So less code
faster UX
better DX

Is Remix your next **framework?**

Want to learn more?

- Remix Guide
- Remix Conf Europ
- awesome-remix
- React-Router 6.4
- Learn Remix