Welcome

What is Remix?

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.

Why another JS framework?

Web dev history

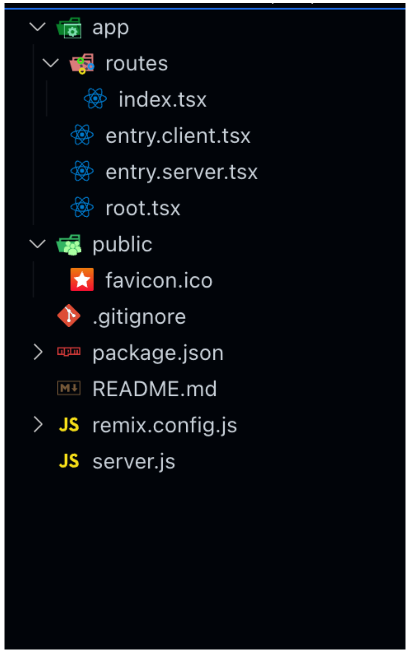
- …
- Backend MVC
- SPA
- SSR
- SSG
- ISR
- DPR
- …

Philosophy of Remix

- Server/Client Model
- Web Standards, HTTP, and HTML
- Progressive Enhancement
- Don’t Over Abstract

Let's start

- app/routes 👉 File based routes
- app/entry.[client/server].ts 👉 Browser / Server render's entry point
- app/root.tsx 👉 Root route, other route will render under this layout
- remix.config.js 👉 Remix configuration file
- server.js 👉 Express.js adapter, different for each adapter



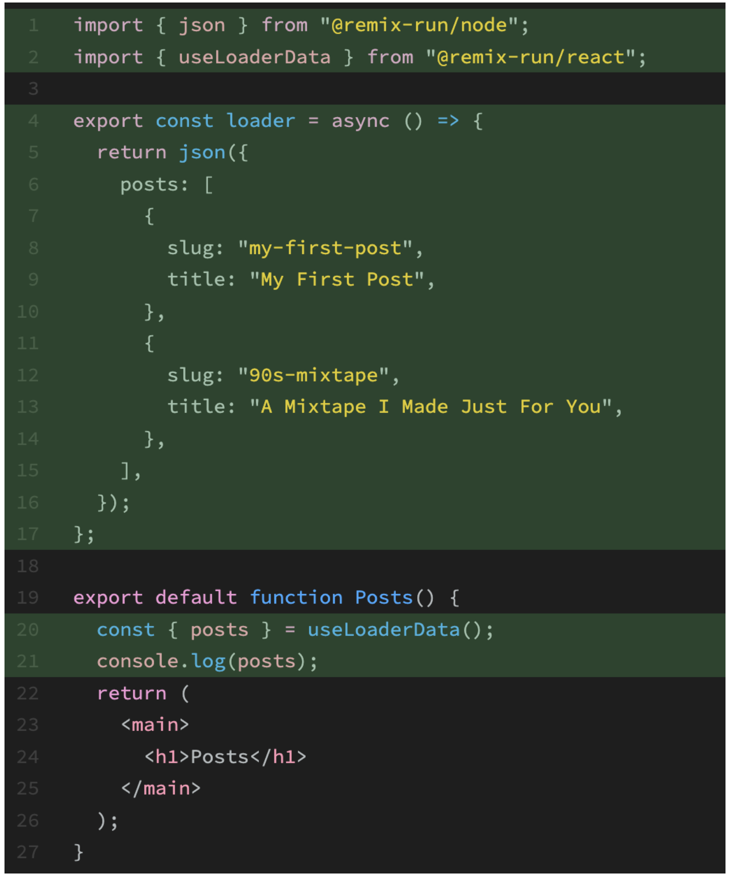
```
app
  routes
    index.tsx
  entry.client.tsx
  entry.server.tsx
  root.tsx
public
  favicon.ico
.gitignore
package.json
README.md
remix.config.js
server.js
```

Elements in a typical page

- Page react component
- loader function
- action function
- links function
- meta function

Fetch data

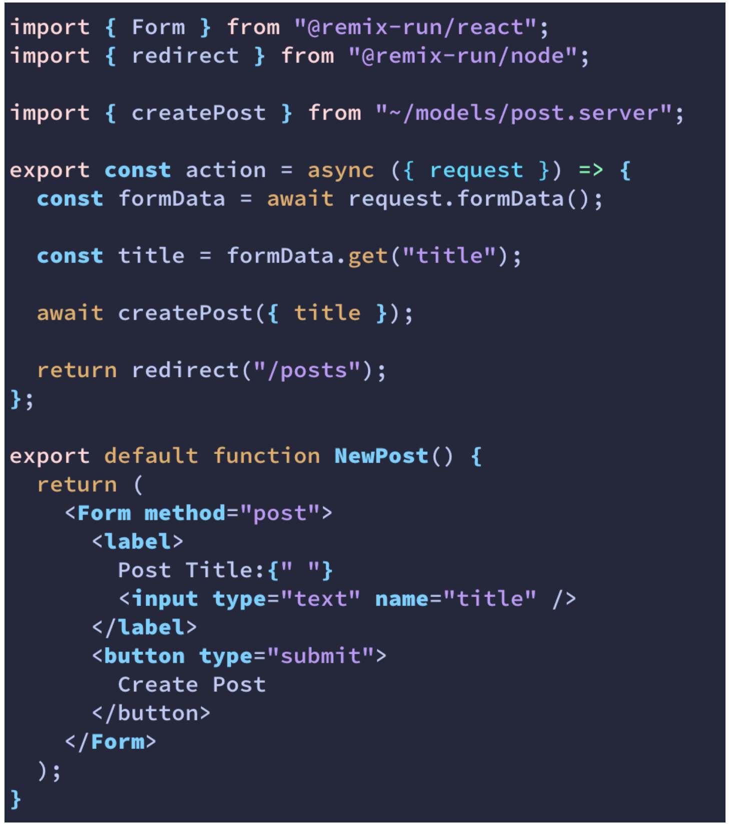
- loader is running on Server side
- useLoaderData is a hook to get the data returned by the loader function



```
import { json } from "@remix-run/node";
import { useLoaderData } from "@remix-run/react";

export const loader = async () => {
  return json({
    posts: [
      {
        slug: "my-first-post",
        title: "My First Post",
      },
      {
        slug: "90s-mixtape",
        title: "A Mixtape I Made Just For You",
      },
    ],
  });
};

export default function Posts() {
  const { posts } = useLoaderData();
  console.log(posts);
  return (
    <main>
      <h1>Posts</h1>
    </main>
  );
}
```

Mutation

- Form is an enhancement of form tag
- action is running after a POST request received
- redirect returns a header in response

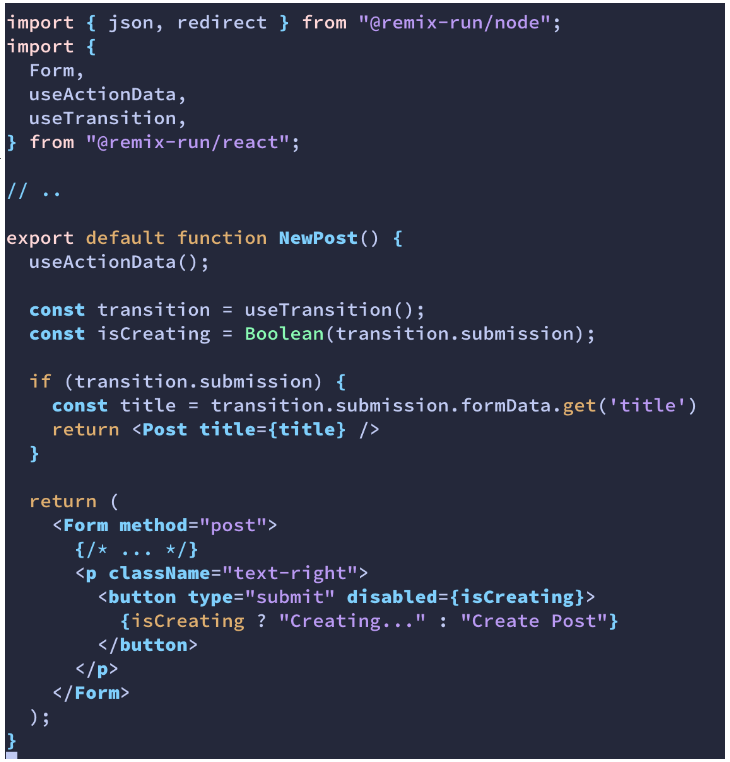


```
import { Form } from "@remix-run/react";
import { redirect } from "@remix-run/node";

import { createPost } from "~/models/post.server";

export const action = async ({ request }) => {
  const formData = await request.formData();

  const title = formData.get("title");

  await createPost({ title });

  return redirect("/posts");
};

export default function NewPost() {
  return (
    <Form method="post">
      <label>
        Post Title:{" "}
        <input type="text" name="title" />
      </label>
      <button type="submit">
        Create Post
      </button>
    </Form>
  );
}
```

It still could work if the browser disabled javascript. 😆

Progressive Enhancement

- useTransition understand the state of current action
- useTransition/useActionData/useLoaderData are hooks
- Optimistic UI

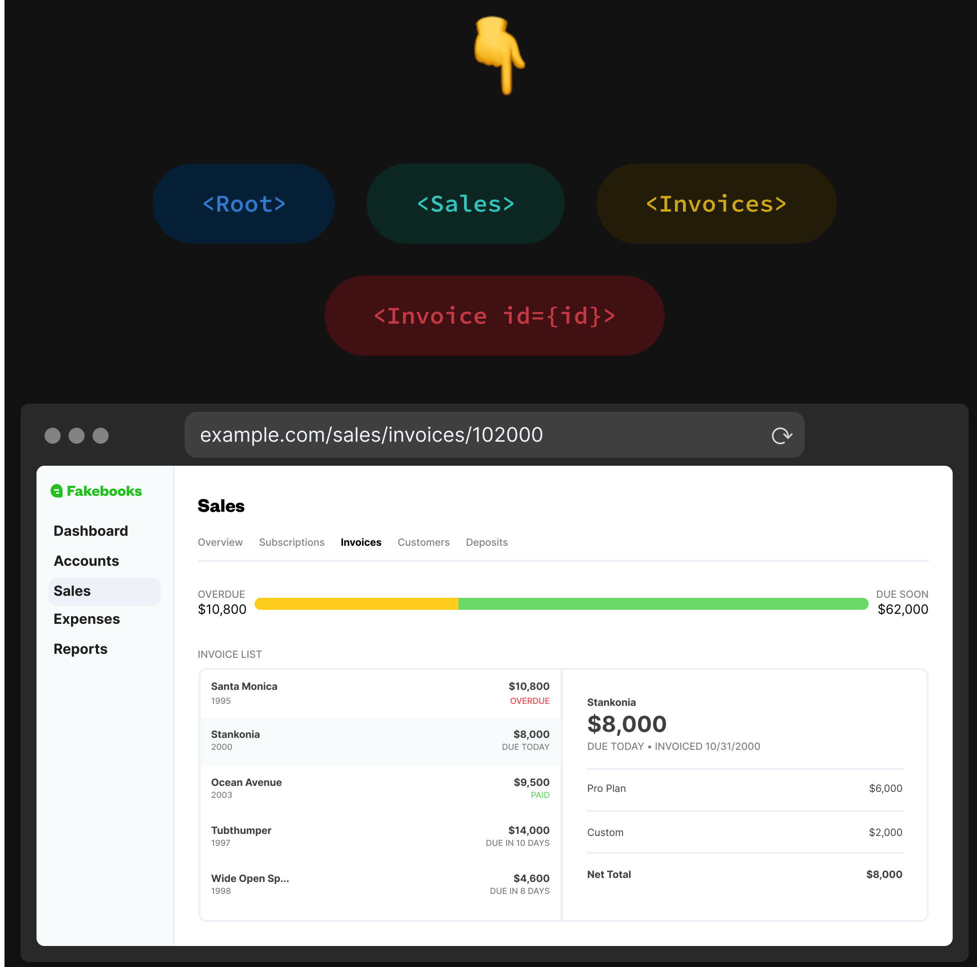


```
import { json, redirect } from "@remix-run/node";
import {
  Form,
  useActionData,
  useTransition,
} from "@remix-run/react";

// ..

export default function NewPost() {
  useActionData();

  const transition = useTransition();
  const isCreating = Boolean(transition.submission);

  if (transition.submission) {
    const title = transition.submission.formData.get('title')
    return <Post title={title} />
  }

  return (
    <Form method="post">
      {/* ... */}
      <p className="text-right">
        <button type="submit" disabled={isCreating}>
          {isCreating ? "Creating..." : "Create Post"}
        </button>
      </p>
    </Form>
  );
}
```

Nested Routing

```
app
├── root.jsx
└── routes
    ├── sales
    │   ├── invoices
    │   │   └── $invoiceId.jsx
    │   └── invoices.jsx
    └── sales.jsx
```

```
<Outlet /> like React's children prop
```



example.com/sales/invoices/102000

Fakebooks

Dashboard
Accounts
Sales
Expenses
Reports

Sales

Overview Subscriptions Invoices Customers Deposits

OVERDUE $10,800 — DUE SOON $62,000

INVOICE LIST

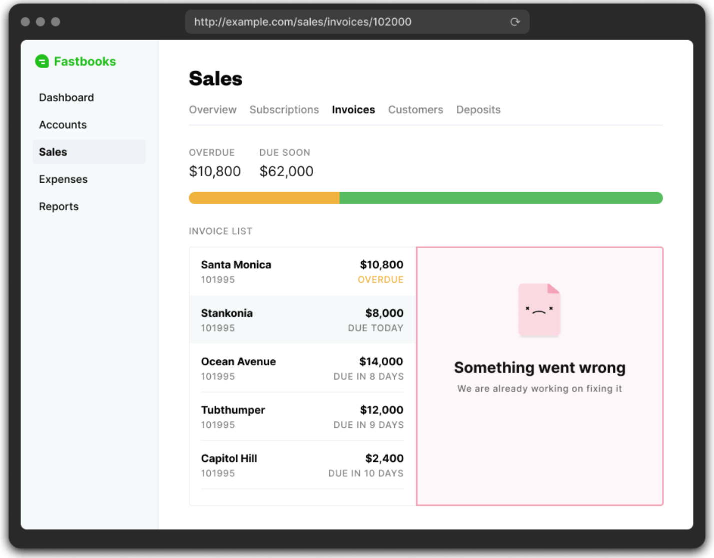
Santa Monica 1995	$10,800 OVERDUE
Stankonia 2000	$8,000 DUE TODAY
Ocean Avenue 2003	$9,500 PAID
Tubthumper 1997	$14,000 DUE IN 10 DAYS
Wide Open Sp... 1998	$4,600 DUE IN 8 DAYS

Stankonia
$8,000
DUE TODAY • INVOICED 10/31/2000

Pro Plan	$6,000
Custom	$2,000
Net Total	$8,000

Error Boundary

```
export function ErrorBoundary({ error }) {
  return (
    <div>
      <h1>Error</h1>
      <p>{error.message}</p>
      <p>The stack trace is:</p>
      <pre>{error.stack}</pre>
    </div>
  );
}
```



OVERDUE $10,800 DUE SOON $62,000

INVOICE LIST

Santa Monica 101995	$10,800 OVERDUE
Stankonia 101995	$8,000 DUE TODAY
Ocean Avenue 101995	$14,000 DUE IN 8 DAYS
Tubthumper 101995	$12,000 DUE IN 9 DAYS
Capitol Hill 101995	$2,400 DUE IN 10 DAYS

Something went wrong

We are already working on fixing it

MISC

- Link
- useFetcher
- cookie and session
- PrefetchPageLinks
- Remix Stacks
- ...

Remix vs Next.js

Remix has a better set of tradeoffs than Next.js.

- Remix is as fast or faster than Next.js at serving static content
- Remix is faster than Next.js at serving dynamic content
- Remix enables fast user experiences even on slow networks
- Remix automatically handles errors, interruptions, and race conditions, Next.js doesn’t
- Next.js encourages client side JavaScript for serving dynamic content, Remix doesn’t
- Next.js requires client side JavaScript for data mutations, Remix doesn’t
- Next.js build times increase linearly with your data, Remix build times are nearly instant and decoupled from data
- Next.js requires you to change your application architecture and sacrifice performance when your data scales
- We think Remix’s abstractions lead to better application code

https://remix.run/blog/remix-vs-next

Why Remix fast than Next.js

- all fast for static content
- SSR can be cached on CDN
- The rewrite one caches data at the edge in Redis

Remix Rewrite
Remix Port
Next.js
0.0
0.0
0.0

https://remix.run/blog/remix-vs-next

Why Remix fast than Next.js

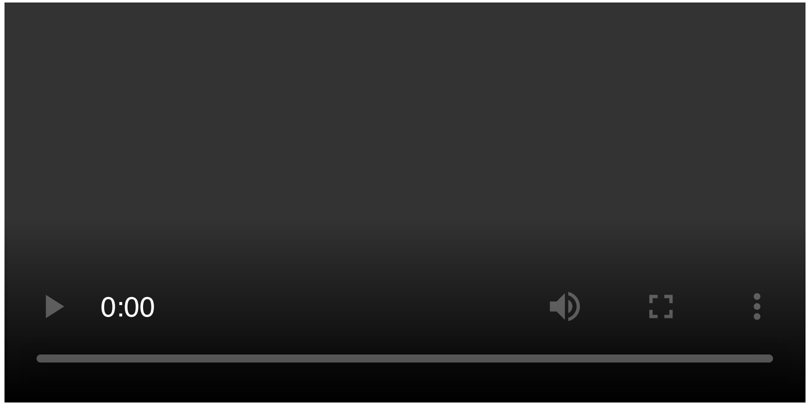
- all fast for dynamic content
- Next.js have a network request waterfall

https://remix.run/blog/remix-vs-next

Why Remix fast than Next.js

PrefetchPageLinks

```
import { Form, PrefetchPageLinks } from "@remix-run/react";

function Search() {
  let [query, setQuery] = useState("");
  return (
    <Form>
      <input type="text" name="q" onChange={(e) => setQuery(e.target.value)} />
      {query && <PrefetchPageLinks page={`/search?q=${query}`} />}
    </Form>
  );
}
```



https://remix.run/blog/remix-vs-next

What is Remix!

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.

Thank you

- https://remix.run/docs/en/v1
- https://kentcdodds.com/blog
- https://www.youtube.com/playlist?list=PLXoynULbYuEDG2wBFSZ66b85EIspy3fy6
- https://github.com/remix-run/remix/blob/main/docs/pages/contributing.md