

# **THE REMIX PHILOSOPHY IS NOT JUST FOR REMIX**



<https://conf42.com>

Ken Snyder, Conf42  
November 2023



# What we'll cover

1. Why should you care
2. Let's create an app!
3. Applying its philosophy

# A little background

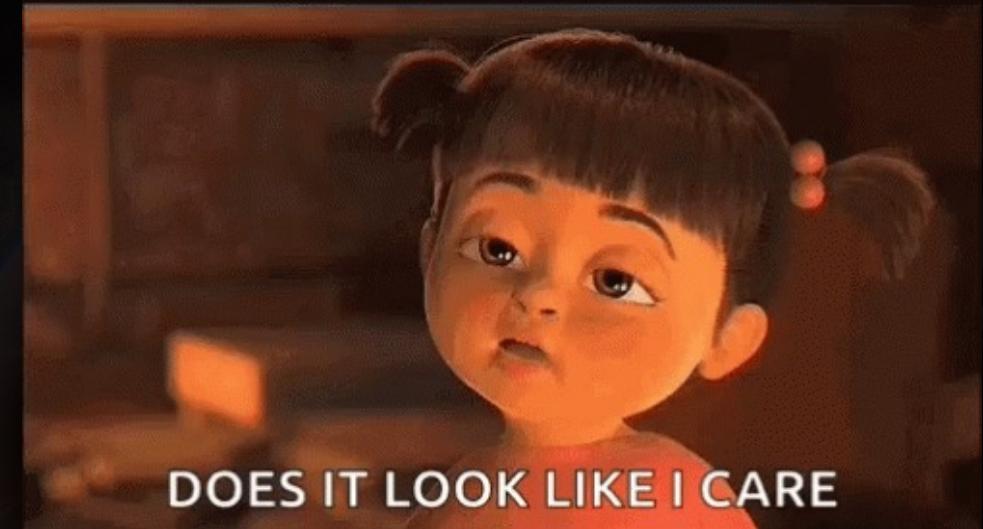
- Shoreline **streamlines patient education** management by centralizing resources and making them easy to share via text, email, and EHR.
- UtahJS is a **JavaScript developer group** with monthly meetups and yearly conferences.
- Remix is a **meta-framework** for web applications running on Node\* and React

\*Or any other JavaScript or TypeScript runtime

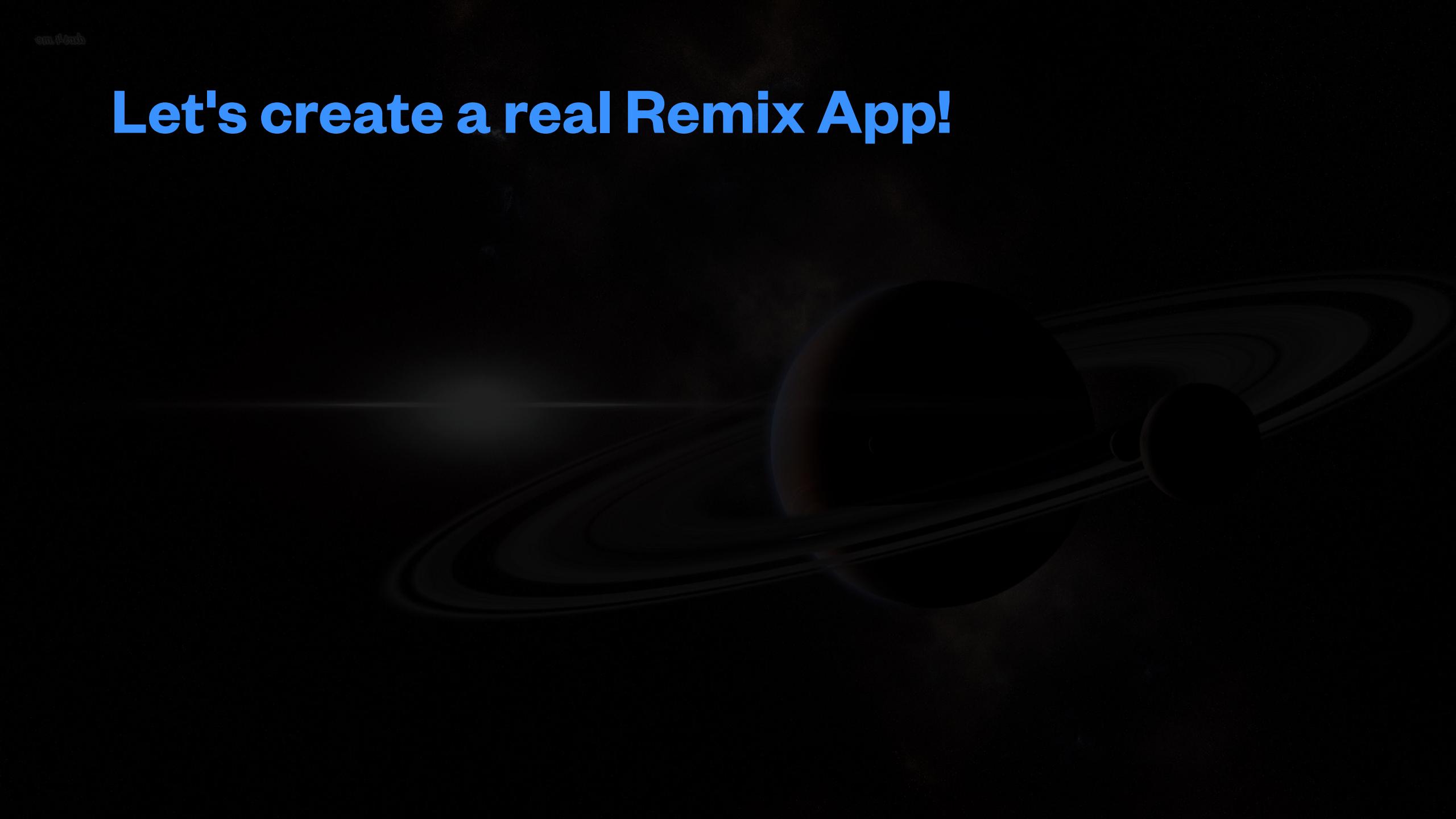


# Why should you care?

- First-render performance
- Immediate interactivity
- React Router
- Part of Shopify



# Let's create a real Remix App!





# Try it on GitHub



<https://github.com/kensnyder/remix-demo-shopping-list>

# <Outlet />

example.com/projects/1234

Dashboards

Projects

About

Project  
Menu

Projects  
Outlet

# More complex forms

Shoreline Beta KS

## Manage Users

Search FILTER

<input type="checkbox"/>	First name <span>▲</span>	Last name <span>▲</span>	Email address <span>▲</span>	Organization <span>▲</span>	Last login <span>▲</span>	Actions
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]	[REDACTED]	10/24/2023 9:12 AM	<span>edit</span> <span>user</span> <span>profile</span>
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]	[REDACTED]	10/24/2023 8:23 AM	<span>edit</span> <span>user</span> <span>profile</span>
<input type="checkbox"/>	BM	[REDACTED]	[REDACTED]	[REDACTED]	N/A	<span>edit</span> <span>user</span> <span>profile</span>
<input type="checkbox"/>	BH	[REDACTED]	[REDACTED]	[REDACTED]	10/27/2023 11:19 AM	<span>edit</span> <span>user</span> <span>profile</span>
<input type="checkbox"/>	BD	[REDACTED]	[REDACTED]	[REDACTED]	N/A	<span>edit</span> <span>user</span> <span>profile</span>

Items per page: 10 ▾ < 1 2 3 4 5 ... 8 >

# More complex forms

Shoreline Beta KS

## Manage Users

[ADD USER](#)

<input type="checkbox"/>	First name	Last name	Email address	Organization	Last login	Actions
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]	[REDACTED]	10/24/2023 9:12 AM	
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]	[REDACTED]	10/24/2023 8:23 AM	
<input type="checkbox"/>	BM	[REDACTED]	[REDACTED]	[REDACTED]	N/A	
<input type="checkbox"/>	BH	[REDACTED]	[REDACTED]	[REDACTED]	10/27/2023 11:19 AM	
<input type="checkbox"/>	BD	[REDACTED]	[REDACTED]	[REDACTED]	N/A	

Items per page: 10 ▾ 1 2 3 4 5 ... 8 >

# More complex forms

Shoreline Beta KS

## Manage Users

+ ADD USER

	First name	Last name	Email address	Organization	Last login	Actions
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]		10/24/2023 9:12 AM	
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]		10/24/2023 8:23 AM	
<input type="checkbox"/>	BM	[REDACTED]	[REDACTED]		N/A	
<input type="checkbox"/>	BH	[REDACTED]	[REDACTED]		10/27/2023 11:19 AM	
<input type="checkbox"/>	BD	[REDACTED]	[REDACTED]		N/A	

Items per page: 10 ▾

1 2 3 4 5 ... 8

# More complex forms

Shoreline Beta KS

## Manage Users

FILTER <button type="submit">Filter</button>

<input type="checkbox"/>	First name ▾	Last name ▾	Email address ▾	Organization ▾	Last login ▾	Actions
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]	[REDACTED]	10/24/2023 9:12 AM	  
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]	[REDACTED]	10/24/2023 8:23 AM	  
<input type="checkbox"/>	BM	[REDACTED]	[REDACTED]	[REDACTED]	N/A	  
<input type="checkbox"/>	BH	[REDACTED]	[REDACTED]	[REDACTED]	10/27/2023 11:19 AM	  
<input type="checkbox"/>	BD	[REDACTED]	[REDACTED]	[REDACTED]	N/A	  

Items per page: 10 ▾ 1 2 3 4 5 ... 8 >

# More complex forms

The screenshot shows a user interface for managing users. At the top left is the Shoreline logo and the word "Beta". On the top right is a circular profile icon with the letters "KS". Below the header is a breadcrumb navigation with a back arrow and the text "Manage Users". To the right of the breadcrumb is a green button with a plus sign and the text "ADD USER".

Below the header is a search bar with a placeholder "Search" and a magnifying glass icon. To the right of the search bar is a green "FILTER" button. To the right of the search bar are two buttons for sorting the email address column: one for升序 (UpIcon) and one for降序 (DownIcon).

The main area is a table with the following columns: a checkbox column, First name (with a dropdown arrow), Last name (with a dropdown arrow), Email address (with a dropdown arrow), Organization (with a dropdown arrow), Last login (with a dropdown arrow), and Actions (with three icons: edit, user, and person). There are six rows of data, each with a checkbox and a small circular badge indicating initials (AB, AB, BM, BH, BD) followed by redacted names and email addresses.

At the bottom of the table are pagination controls: "Items per page: 10" with a dropdown arrow, a page number "1" in a highlighted circle, and arrows for navigating between pages (1, 2, 3, 4, 5, ..., 8, >).

Action	First name	Last name	Email address	Organization	Last login	Actions
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]		10/24/2023 9:12 AM	
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]		10/24/2023 8:23 AM	
<input type="checkbox"/>	BM	[REDACTED]	[REDACTED]		N/A	
<input type="checkbox"/>	BH	[REDACTED]	[REDACTED]		10/27/2023 11:19 AM	
<input type="checkbox"/>	BD	[REDACTED]	[REDACTED]		N/A	

# More complex forms

The screenshot shows a user management interface for 'Shoreline Beta'. At the top right is a user icon with 'KS'. On the right side of the header is a green button with a plus sign and 'ADD USER'. Below the header is a search bar with a magnifying glass icon and a 'FILTER' button. The main area is a table titled 'Manage Users' with columns: a checkbox, First name (with a dropdown arrow), Last name (with a dropdown arrow), Email address (with a dropdown arrow), Organization (with a dropdown arrow), Last login (with a dropdown arrow), and Actions. There are six rows of data, each with a checkbox and a small circular icon containing initials (AB, AB, BM, BH, BD). The last row's organization and last login fields show 'N/A'. The actions column contains three icons: a pencil, a person, and a person with a gear. At the bottom left is a 'Items per page: 10' dropdown. The bottom center shows a navigation bar with page numbers 1, 2, 3, 4, 5, ..., 8, >. A pink box highlights the page number '2'.

<input type="checkbox"/>	First name	Last name	Email address	Organization	Last login	Actions
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]		10/24/2023 9:12 AM	
<input type="checkbox"/>	AB	[REDACTED]	[REDACTED]		10/24/2023 8:23 AM	
<input type="checkbox"/>	BM	[REDACTED]	[REDACTED]		N/A	
<input type="checkbox"/>	BH	[REDACTED]	[REDACTED]		10/27/2023 11:19 AM	
<input type="checkbox"/>	BD	[REDACTED]	[REDACTED]		N/A	

`<button type="submit" name="page" value="2">2</button>`

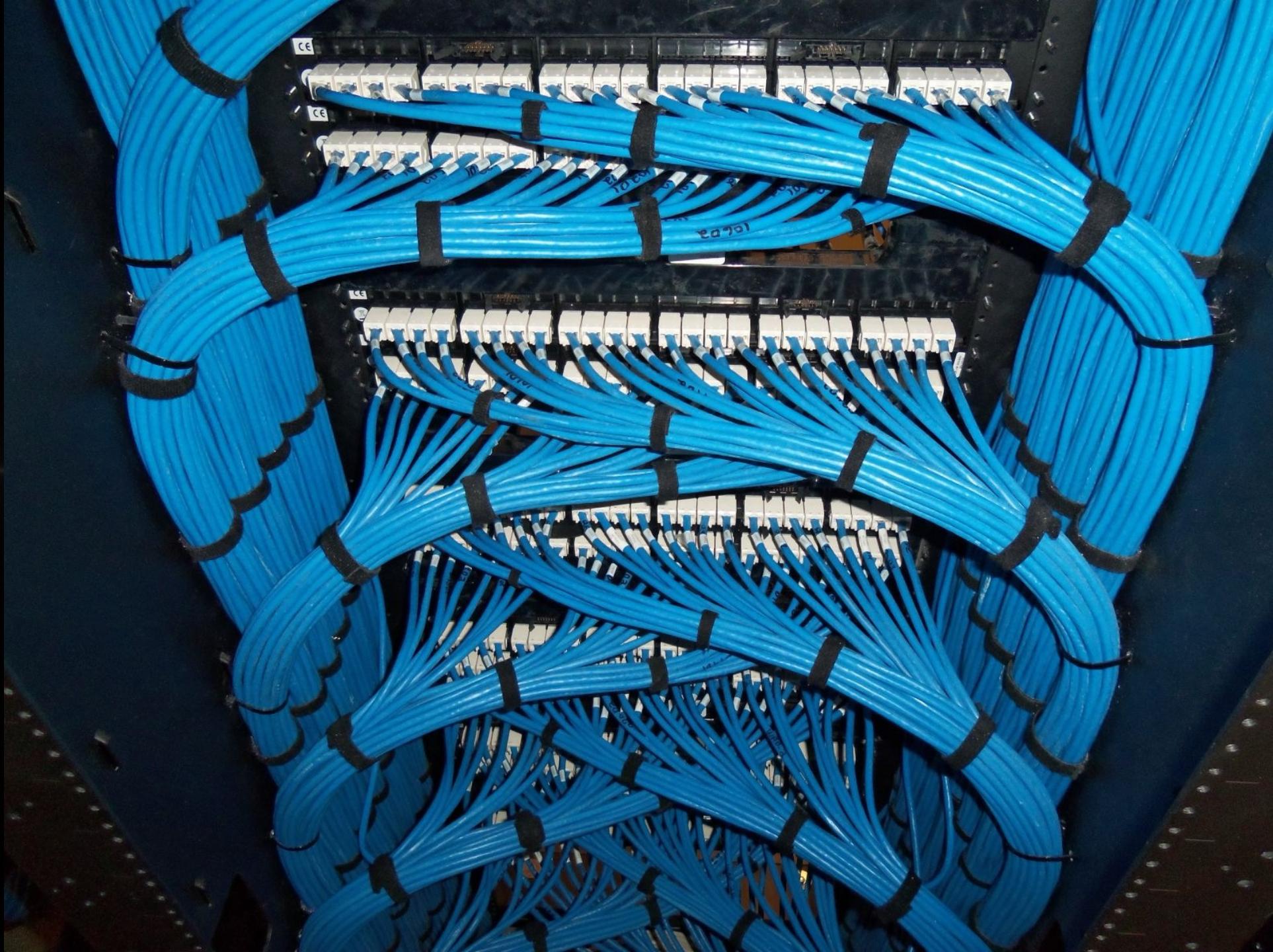
# Philosophy

1. Embrace server/client model
2. Work with web foundations
3. Work without JavaScript
4. Roadmap & future flags

# Philosophy

1. Embrace server/client model
2. Work with web foundations
3. Work without JavaScript
4. Roadmap & future flags

# Bundles



# Bundles

Server  
Bundle

```
4 > export const items = [...  
10 ];  
11  
12 export async function loader({ request }: LoaderFunctionArgs) {  
13   return json({ items });  
14 }  
15  
16 export default function Items() {  
17   const { items } = useLoaderData<typeof loader>();  
18   return (  
19     <main>  
20       <ul>  
21 >       {items.map(item => (  
22         <li>{item}</li>  
23       )}  
24     </ul>  
25     <Form method="POST" key={items.length}>  
26       <input name="name" autoFocus />  
27       <button>Add item</button>  
28     </Form>  
29   </main>  
30 );  
31 }  
32  
33 export async function action({ request }: ActionFunctionArgs) {  
34   const formData = await request.formData();  
35   const name = formData.get('name') as string;  
36   items.push(  
37     {  
38       id: +new Date(),  
39       name,  
40       inCart: false,  
41     }  
42   );  
43   return null;  
44 }
```

Browser  
Bundle

# Single page app



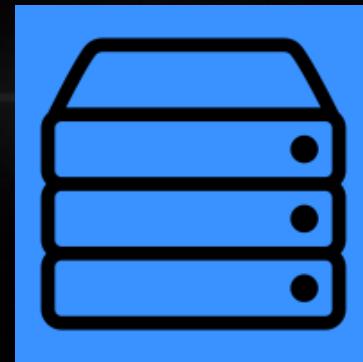
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>...
7 </head>
8 <body>
9   <noscript>Please enable JavaScript</noscript>
10  <script src="bundle.js"></script>
11 </body>
12 </html>
```

```
fetch('/todos')
```

```
  "name": "Example ALARM", "date": "07/17/2017 10:58", "username": "user1", "id": "00134558", "device": "001B2D3104E551F0", "serial": "RAJ-E-12345", "deviceType": "RF", "refreshTime": "00", "display": "0", "interval": "300", "grid": "interval", "version": "1.0", "status": "OK", "lastConfig": "2017-07-17T10:58:00Z", "config": "[]", "timezones": "[]", "formart": "0", "display": "0", "daylight_saving_in": "0", "sensor": "[]"}, {"label": "Internal", "id": "00134559", "temp": "20.43", "humidity": "70.55", "light": "66.77", "lowTemp": "18.43", "highTemp": "22.55", "lowHumidity": "60.00", "highHumidity": "70.00", "lowLight": "60.00", "highLight": "70.00", "alarm": "[]"}, {"label": "Ext Sensor", "id": "00134560", "temp": "32.00", "humidity": "0.00", "light": "32.00", "lowTemp": "30.00", "highTemp": "34.00", "lowHumidity": "0.00", "highHumidity": "0.00", "lowLight": "0.00", "highLight": "0.00", "alarm": "[]"}, {"label": "Motion Sen", "id": "00134561", "temp": "0.00", "humidity": "0.00", "light": "0.00", "lowTemp": "0.00", "highTemp": "0.00", "lowHumidity": "0.00", "highHumidity": "0.00", "lowLight": "0.00", "highLight": "0.00", "alarm": "[]"}]
```

```
15 <main>
16   >   <ul>...
17   </ul>
18   </main>
19   <form>
20     <input ... />
21     <button ... />
22   </form>
23 </main>
```

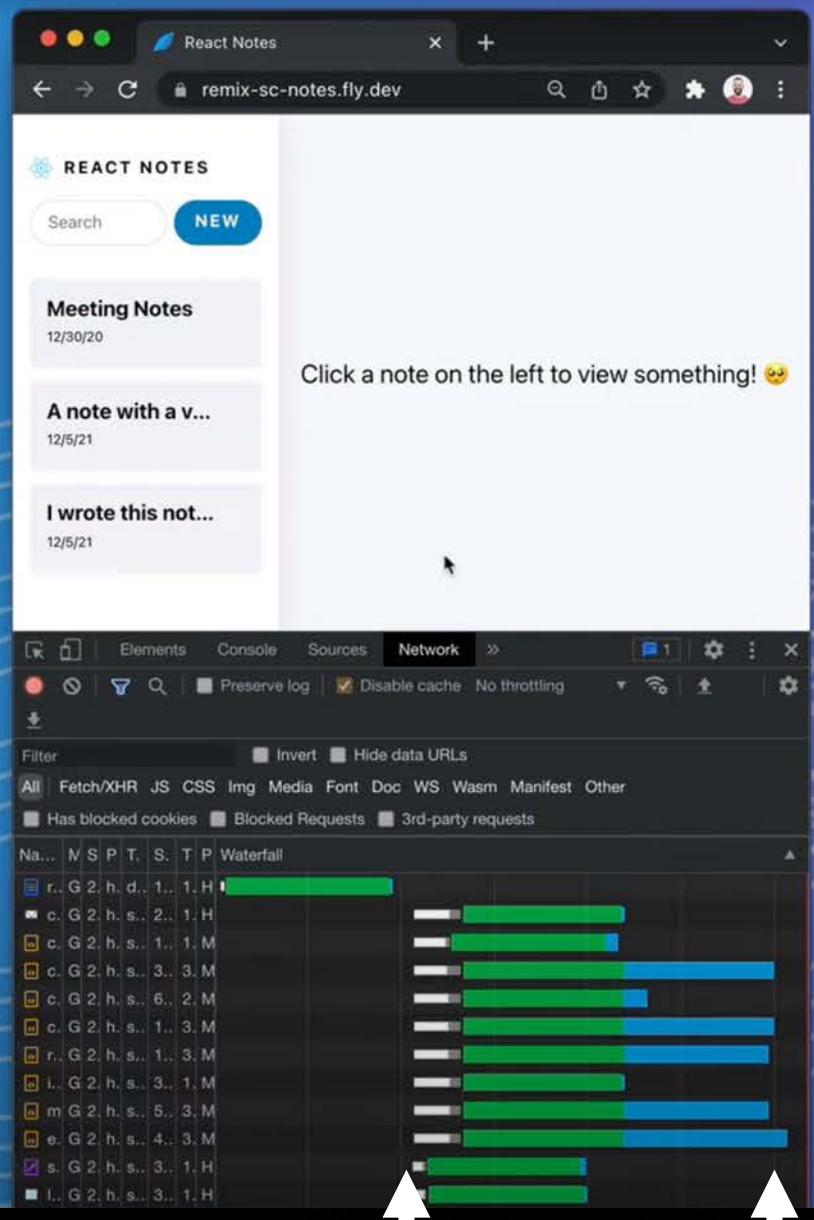
# Remix app



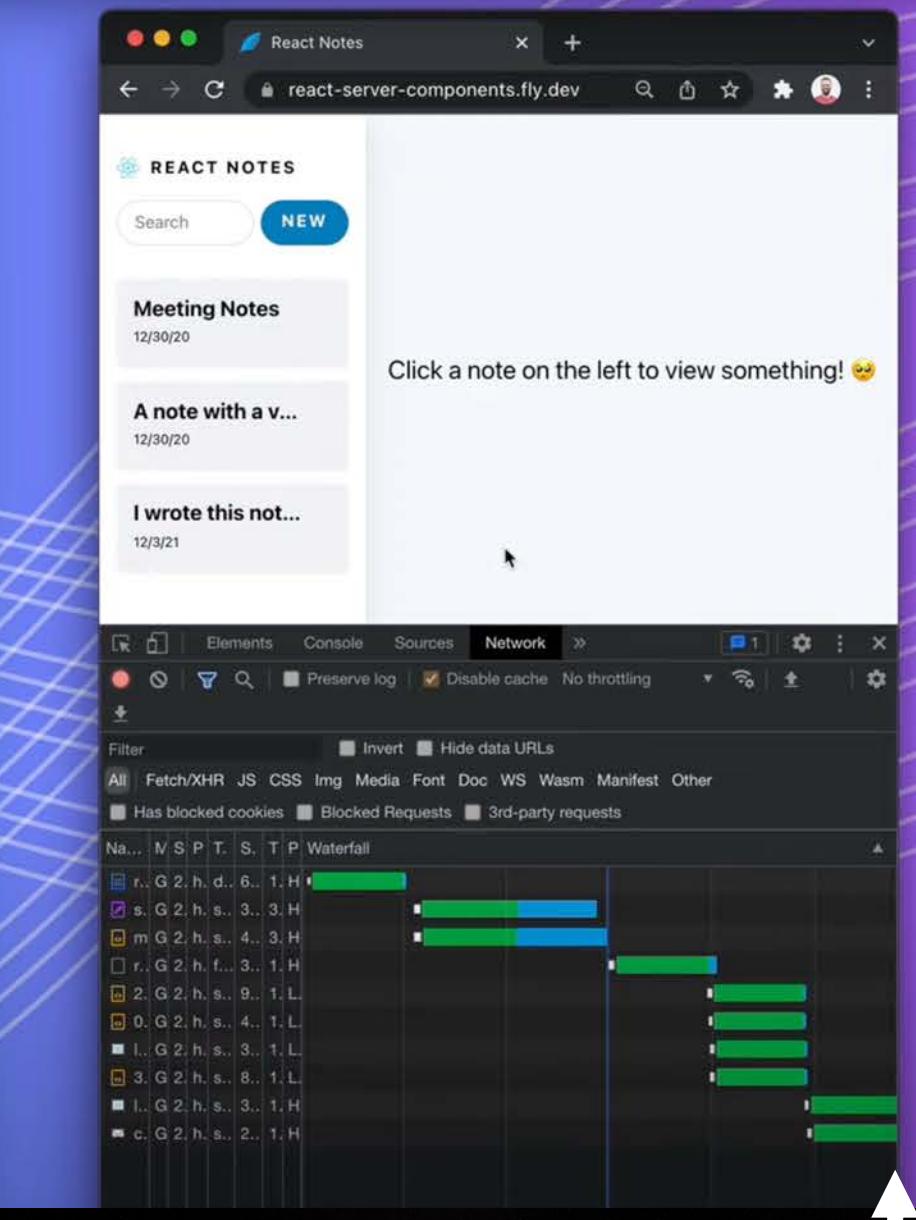
```
1  <!DOCTYPE html>
2  <html lang="en">
3  > <head>...
7  </head>
8  <body>
9  > <main>
10 >   <ul>...
12 </ul>
13 <form>
14   <input ... />
15   <button ... />
16 </form>
17 </main>
18 <script src="hydrator.js" type="module"></script>
19 </body>
20 </html>
```

```
function addInteractivity() {
    const hydrator = document.querySelector('hydrator');
    hydrator.addEventListener('load', () => {
        const hydrate = hydrator.querySelector('hydrate');
        hydrate.addEventListener('click', () => {
            hydrator.innerHTML = hydrator.getAttribute('data-hydrate');
        });
    });
}
```

addInteractivity()



## Fully painted



# Fully interactive

# Server & runtime adapters

- Cloudflare Workers
- Deno
- Bun
- Netlify
- Vercel
- Azure Functions
- AWS Lambda
- Google Cloud Functions
- Express
- Fastify
- Many more



# Server & runtime adapters

- Cloudflare Workers
- Deno
- Bun
- Netlify
- Vercel
- Azure Functions
- AWS Lambda
- Google Cloud Functions
- Express
- Fastify
- Many more



# Server & runtime adapters

- Cloudflare Workers
- Deno
- Bun
- Netlify
- Vercel
- Azure Functions
- AWS Lambda
- Google Cloud Functions
- Express
- Fastify
- Many more



# Official list

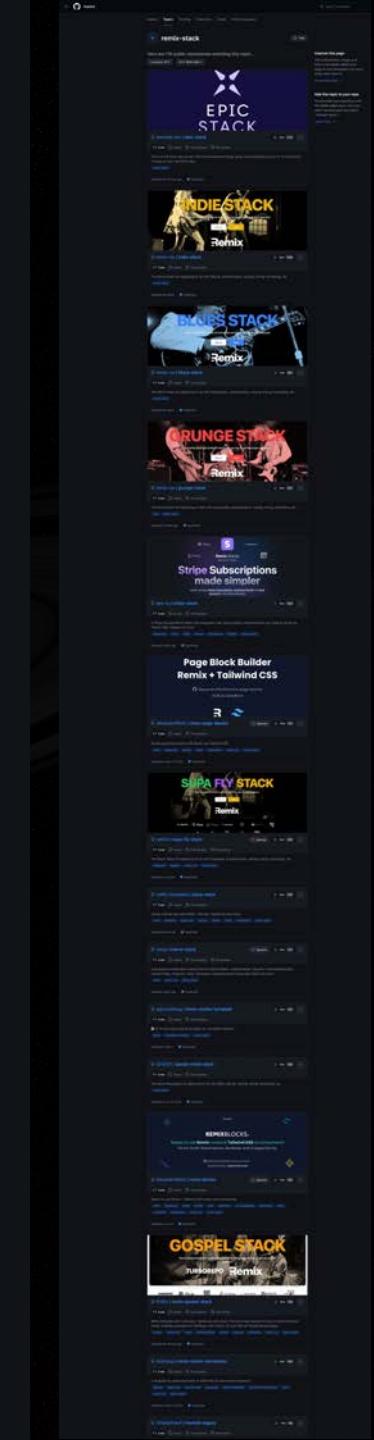
# remix-stack

Star

Here are 116 public repositories matching this topic...

Language: All ▾ Sort: Most stars ▾

The screenshot shows the GitHub search results for the query "# remix-stack". At the top, there's a large button for "Star" and a header with the search term "# remix-stack". Below the header, a message says "Here are 116 public repositories matching this topic...". There are two dropdown menus: "Language: All ▾" and "Sort: Most stars ▾". The main content area features a large banner for the "EPIC STACK" repository. The banner has a dark blue background with a purple starburst icon at the top. Below the icon, the words "EPIC STACK" are written in large, white, sans-serif capital letters. Underneath the banner, the repository card for "epicweb-dev / epic-stack" is visible. It includes the repository name, a star icon with "Star 3.1k", and tabs for "Code", "Issues", "Pull requests", and "Discussions". A description below the tabs reads: "This is a Full Stack app starter with the foundational things setup and configured for you to hit the ground running on your next EPIC idea." A "remix-stack" tag is present. At the bottom of the card, it says "Updated 25 minutes ago" and "TypeScript".





# Philosophy

1. Embrace server/client model
2. Work with web foundations
3. Work without JavaScript
4. Roadmap & future flags

# The Elders of the Internet



# Standards

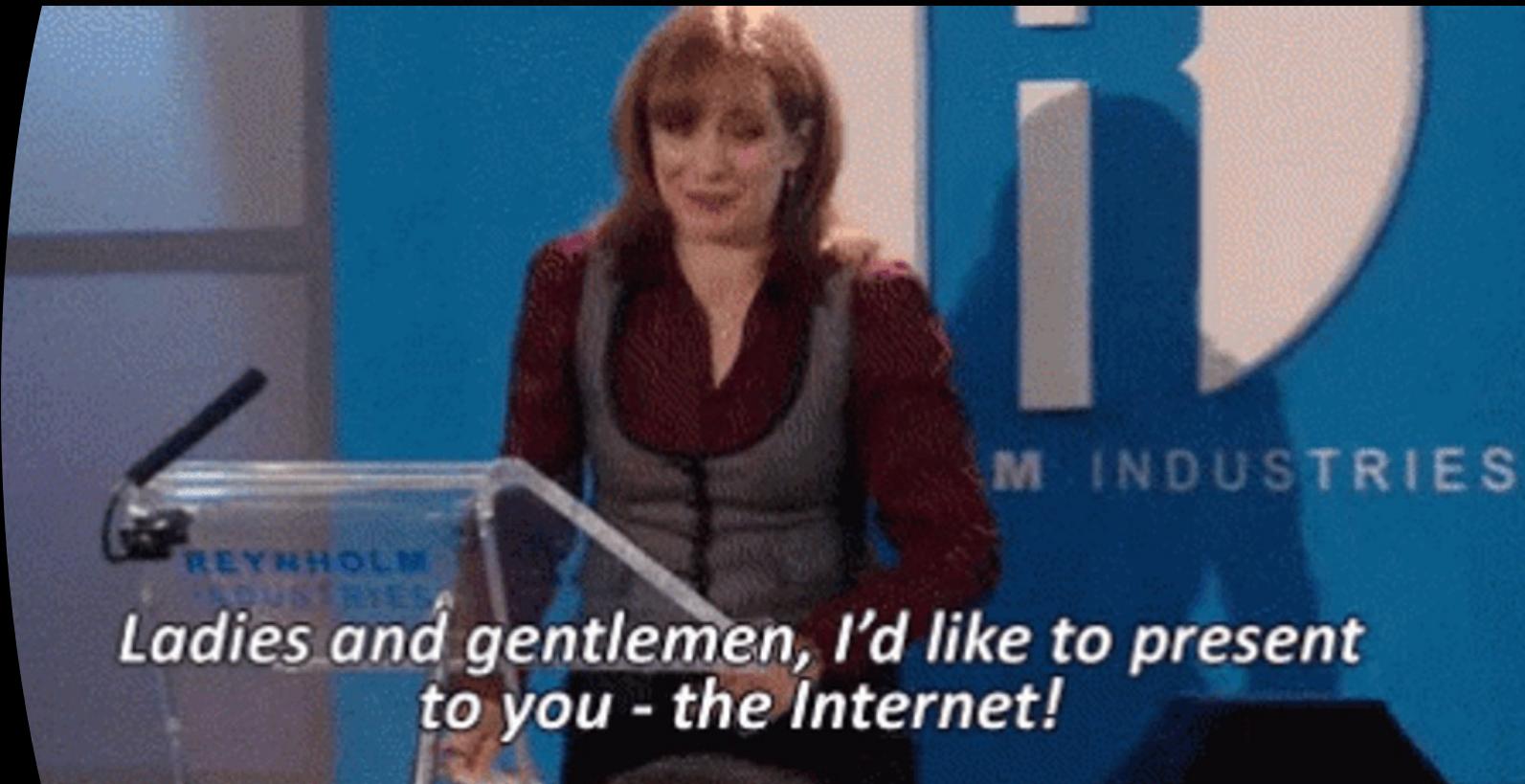
- Request
- Response
- fetch
- FormData
- Headers



MakeAGIF.com

# Standards

- URL
- URLSearchParams
- HTTP Prefetching
- HTTP Caching
- HTML Forms



# Request, Response

```
4 // Bun
5 Bun.serve({
6   port: 80,
7   fetch: async (request: Request) => {
8     return new Response('Hello World');
9   },
10 });
11
12 // Deno
13 Deno.serve(
14   {
15     port: 80,
16   },
17   async (request: Request) => {
18     return new Response('Hello World');
19   }
20 );
21
22 // Cloudflare Workers
23 export default {
24   fetch: async (request: Request) => {
25     return new Response('Hello World');
26   },
27 };
28
```

# fetch loves Request & Response

```
75 // Same signatures
76 await fetch(url, init);
77 new Request(url, init);
78
79 // Or you can directly use a Request object:
80 const request = new Request(url, init);
81 await fetch(request);
```

```
85 // And note that fetch returns a Response promise
86 > function fetch(request: Request) : Promise<Response> { ...
89 }
```

# Request, Response, fetch

```
75 // Same signatures
76 await fetch(url, init);
77 new Request(url, init);
78
79 // Or you can directly use a Request object:
80 const request = new Request(url, init);
81 await fetch(request);
```

```
85 // And note that fetch returns a Response promise
86 > function fetch(request: Request) : Promise<Response> { ...
89 }
```

# Request, Response, fetch

```
75 // Same signatures
76 await fetch(url, init);
77 new Request(url, init);
78
79 // Or you can directly use a Request object:
80 const request = new Request(url, init);
81 await fetch(request);
```

```
85 // And note that fetch returns a Response promise
86 > function fetch(request: Request) : Promise<Response> { ...
89 }
```

# Headers (Request & Response)

```
60 request.headers instanceof Headers; // true      29 new Response('Hello World', {  
61  
62 request.headers.get('Cookie')  
30   headers: new Headers({  
31     'Content-type': 'text/plain',  
32   }),  
33 });  
34  
35 new Response('Hello World', {  
36   headers: new Headers([  
37     ['Content-type', 'text/plain'],  
38     ['Set-Cookie', '< cookie 1 >'],  
39     ['Set-Cookie', '< cookie 2 >'],  
40   ]),  
41 });  
42
```

# URLSearchParams

```
43 new URLSearchParams('filter=Hello&sort=name');  
44  
45 new URLSearchParams({  
46   filter: 'Hello',  
47   sort: 'name',  
48 });  
49  
50 new URLSearchParams([  
51   ['filter', 'Hello'],  
52   ['sort', 'name'],  
53   ['tags[]', 'JavaScript'],  
54   ['tags[]', 'TypeScript'],  
55 ]);  
56
```

# Web standards in plain React code

# Traditional React component

```
84 function AddUserPageJson() {
85   const addUser = useCallback(async (evt) => {
86     evt.preventDefault();
87     const formData = new FormData(evt.target);
88     const { first, last, email } = Object.fromEntries(formData);
89     await fetch('/users', {
90       method: 'POST',
91       headers: { 'Content-type': 'application/json' },
92       body: JSON.stringify({ first, last, email }),
93     });
94     navigate('/users')
95   }, []);
96   return (
97     <form onSubmit={addUser}>
98       <input type="text" name="first" placeholder="First name" />
99       <input type="text" name="last" placeholder="Last name" />
100      <input type="email" name="email" placeholder="Email" />
101      <button>Submit</button>
102    </form>
103  );
104}
```

# Traditional React component

```
84 function AddUserPageJson() {
85   const addUser = useCallback(async (evt) => {
86     evt.preventDefault();
87     const formData = new FormData(evt.target);
88     const { first, last, email } = Object.fromEntries(formData);
89     await fetch('/users', {
90       method: 'POST',
91       headers: { 'Content-type': 'application/json' },
92       body: JSON.stringify({ first, last, email }),
93     });
94     navigate('/users')
95   }, []);
96   return (
97     <form onSubmit={addUser}>
98       <input type="text" name="first" placeholder="First name" />
99       <input type="text" name="last" placeholder="Last name" />
100      <input type="email" name="email" placeholder="Email" />
101      <button>Submit</button>
102    </form>
103  );
104}
```

# Traditional React component

```
84 function AddUserPageJson() {
85   const addUser = useCallback(async (evt) => {
86     evt.preventDefault();
87     const formData = new FormData(evt.target);
88     const { first, last, email } = Object.fromEntries(formData.entries());
89     await fetch('/users', {
90       method: 'POST',
91       headers: { 'Content-type': 'application/json' },
92       body: JSON.stringify({ first, last, email })
93     });
94     navigate('/users')
95   }, []);
96   return (
97     <form onSubmit={addUser}>
98       <input type="text" name="first" placeholder="First name" />
99       <input type="text" name="last" placeholder="Last name" />
100      <input type="email" name="email" placeholder="Email" />
101      <button>Submit</button>
102    </form>
103  );
104}
105
```

```
107 function AddUserPageFormData() {
108   const addUser = useCallback(async (evt) => {
109     evt.preventDefault();
110     const formData = new FormData(evt.target);
111     await fetch('/users', {
112       method: 'POST',
113       headers: { 'Content-type': 'application/x-www-form-urlencoded' },
114       body: formData,
115     });
116     navigate('/users')
117   }, []);
118   return (
119     <form onSubmit={addUser}>
120       <input type="text" name="first" placeholder="First name" />
121       <input type="text" name="last" placeholder="Last name" />
122       <input type="email" name="email" placeholder="Email" />
123       <button>Submit</button>
124     </form>
125   );
126 }
```

# More traditional React page

```
84 function AddUserPageJson() {          107 function AddUserPageFormData() {  
85   const addUser = useCallback(async (evt) => { 108   const addUser = useCallback(async (evt) => {  
86     evt.preventDefault();                      109     evt.preventDefault();  
87     const formData = new FormData(evt.target); 110     const formData = new FormData(evt.target);  
88     const { first, last, email } = Object.fromEntries(await fetch('/users', { 111     await fetch('/users', {  
89       method: 'POST',                         112       method: 'POST',  
90       headers: { 'Content-type': 'application/x-www-form-urlencoded' }, 113       headers: { 'Content-type': 'application/x-www-form-urlencoded' },  
91       body: JSON.stringify({ first, last, email }) 114       body: formData,  
92     });                                         115     });  
93   });                                         116   navigate('/users')  
94   navigate('/users')                         117 }, []);  
95 }, []);                                     118 return (  
96   return (  
97     <form onSubmit={addUser}>                119     <form onSubmit={addUser}>  
98       <input type="text" name="first" placeholder="First name" /> 120       <input type="text" name="first" placeholder="First name" />  
99       <input type="text" name="last" placeholder="Last name" /> 121       <input type="text" name="last" placeholder="Last name" />  
100      <input type="email" name="email" placeholder="Email" /> 122       <input type="email" name="email" placeholder="Email" />  
101      <button>Submit</button>                 123       <button>Submit</button>  
102    </form>                                124   </form>  
103  );                                         125 );  
104 }                                         126 }
```

# ...Or just use Remix

```
128 import { Form } from '@remix-run/react';
129
130 function AddUserPageRemix() {
131   return (
132     <Form action="/users">
133       <input type="text" name="first" placeholder="First name" />
134       <input type="text" name="last" placeholder="Last name" />
135       <input type="email" name="email" placeholder="Email" />
136       <button>Submit</button>
137     </Form>
138   );
139 }
```

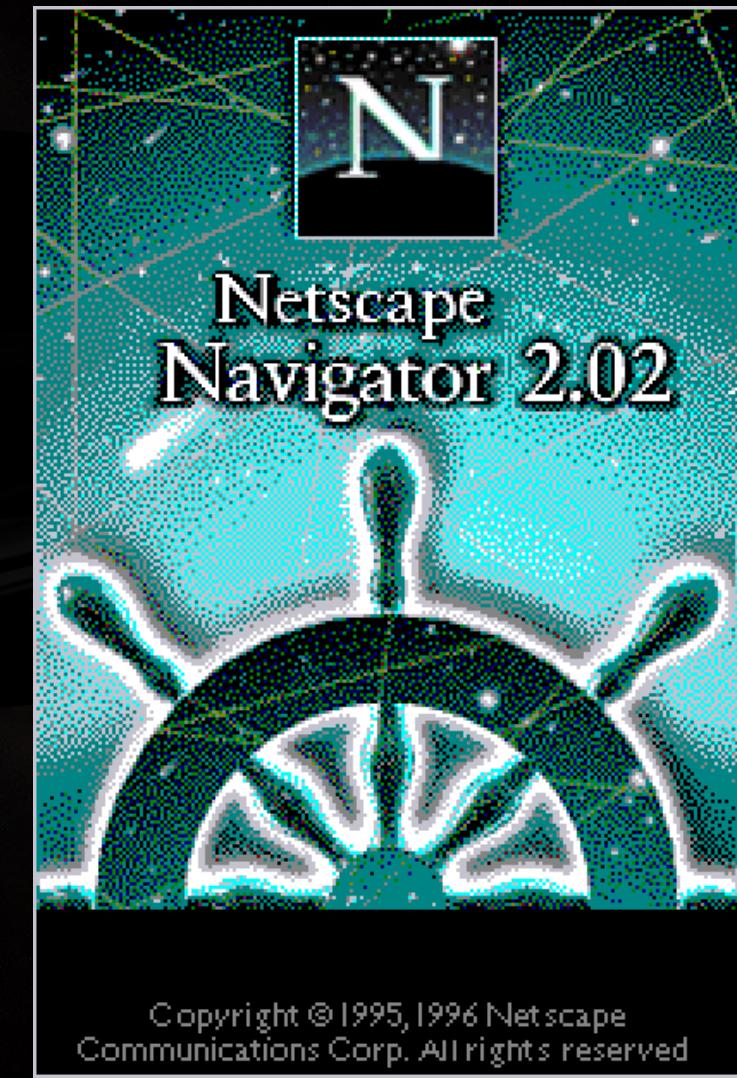
# Philosophy

1. Embrace server/client model
2. Work with web foundations
3. Work without JavaScript
4. Roadmap & future flags

# Work without JavaScript



- Forms still work before hydration
- Support for IE11



Copyright © 1995, 1996 Netscape  
Communications Corp. All rights reserved

The screenshot displays a developer's environment with three main panes:

- Left Pane (Code Editor):** Shows the file `TS root.tsx` open in the editor. The code defines a `App` component that returns an `<html>` element with `<head>` and `<body>` sections containing `<Outlet>`, `<ScrollRestoration>`, `<Scripts>`, and `<LiveReload>`. The file has 34 lines of code.
- Middle Pane (Browser Preview):** A web browser window titled "localhost" shows a simple "Shop" page. It features a list of items with a single item currently selected: "Bread". Below the list is a button labeled "Add item".
- Bottom Pane (Terminal):** A terminal window titled "node" shows the output of the `remix dev` command. It includes the message "building...", "built (301ms)", and the URL "[remix-serve] http://localhost:3000 (http://192.168.1.140:3000)". The terminal also shows a GET request for `/shop/items` with a response status of 200 and a duration of 33.465 ms.

# Philosophy

1. Embrace server/client model
2. Work with web foundations
3. Work without JavaScript
4. Roadmap & future flags

# Roadmap & RFCs are published on GitHub

The image shows a GitHub Roadmap interface with the following columns:

- Backlog**: Contains one item: "Warn when remix packages are out of sync".
- Planned**: Contains seven items:
  - remix #7639: Remix SPA Target
  - remix #7641: Single Fetch
  - remix #7643: Middleware
  - remix #7645: Server Context
  - remix #7646: Migrate Integration tests to @remix-run/testing
  - Draft: Make build and watch from remix-dev public API for hydrogen
  - remix #7824: Remix Package
- In Progress**: Contains two items:
  - remix #7635: Client Data
  - remix #7633: Vite
- Merged**: Contains three items:
  - ✓ remix #7710: Fetcher Tweaks
  - Draft: Build error popup
  - Draft: View Transitions
- Released**: Contains zero items.

At the top of the board, there are navigation tabs: Board (selected), List, Gantt, and a search/filter bar. On the right side, there are three buttons: Discard, ... (three dots), and a copy icon.

# Roadmap Highlights

- Vite build system
- Loader/action middleware
- Optimistic update tools

# Future flags

```
1  /**  
2   * @type {import('@remix-run/dev').AppConfig}  
3   */  
4  module.exports = {  
5    future: {  
6      v2_dev: true,  
7      v2_headers: true,  
8      v2_errorBoundary: true,  
9      v2_meta: true,  
10     v2_normalizeFormMethod: true,  
11     v2_routeConvention: true,  
12   },  
13   /* other configuration options here */  
14};
```

# Applying the philosophy outside Remix

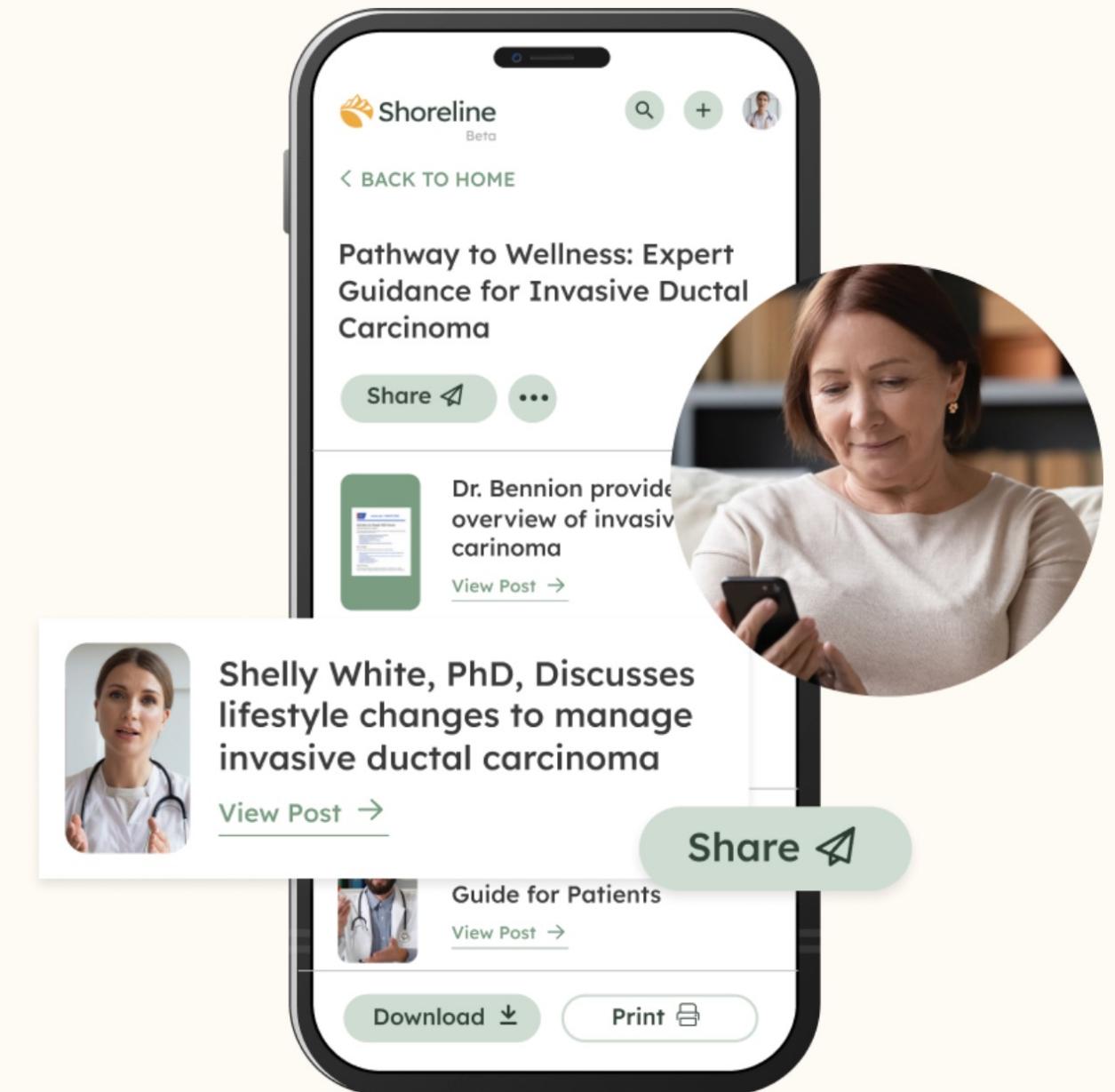
- Directly use fetch, FormData, URL
- Learn web standards
- Learn runtime standards
- Progressive enhancement

# We chose a new mindset

- I don't worry about:
  - Where my code runs
  - Routing, submission and rendering target
- I am liberated from:
  - Server-side rendering duplication
  - Worry about first-render performance

# Hello Shoreline

<https://shoreline.health>



# THANK YOU

Remix

Ken Snyder, Conf42  
November 2023

<https://conf42.com>

