inovex

SolidJS

Reaktivität einfach gemacht

Tilman Adler, Bernd Kaiser

Tilman Adler

- Software Developer at inovex Erlangen
- Webdev of passion

https://linkedin.com/in/tilman-adler

Bernd Kaiser

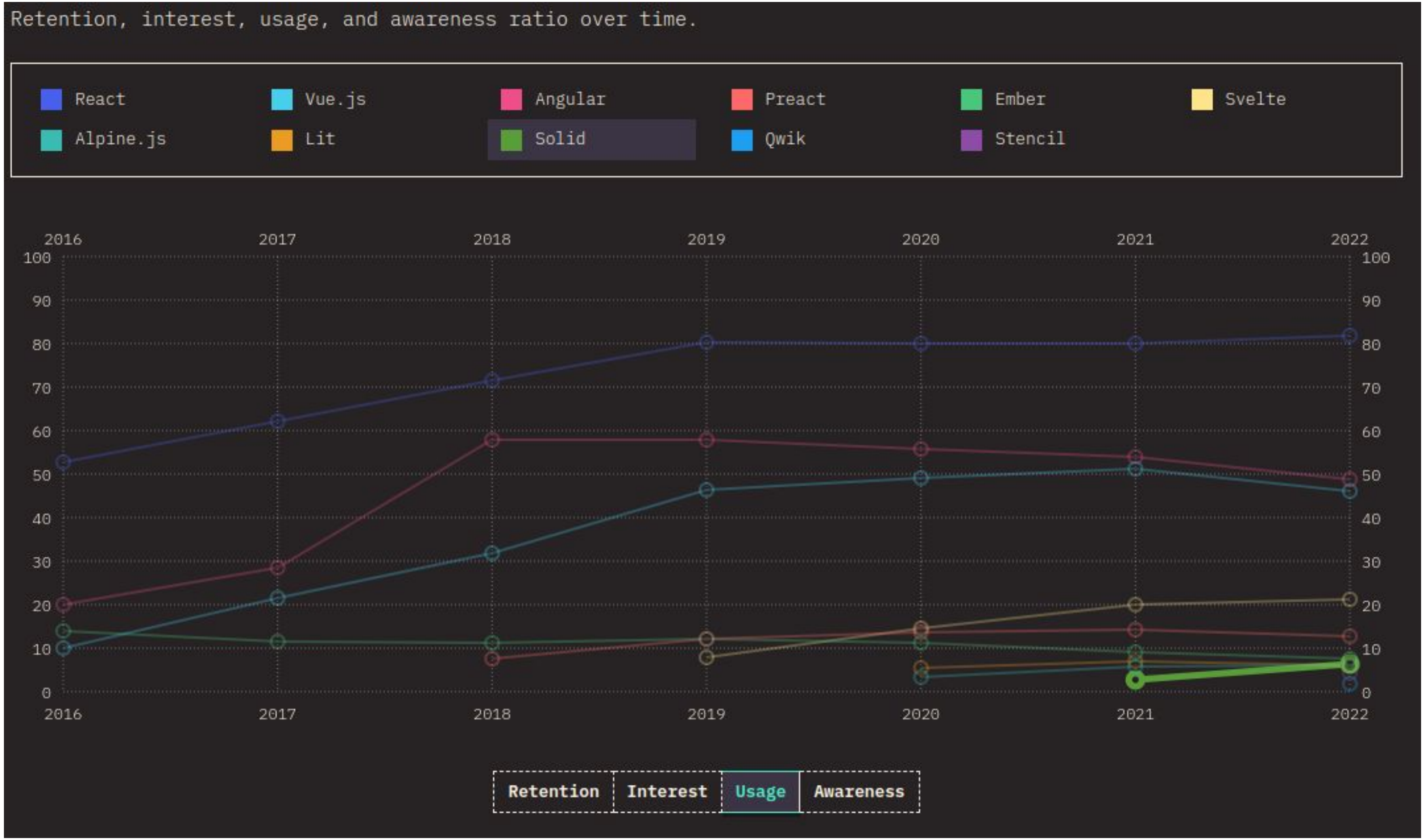
- Software Developer at inovex Erlangen
- CS Master with Focus on IT Security
- JS developer for over 20 years

https://linkedin.com/in/bernd-kaiser

Agenda

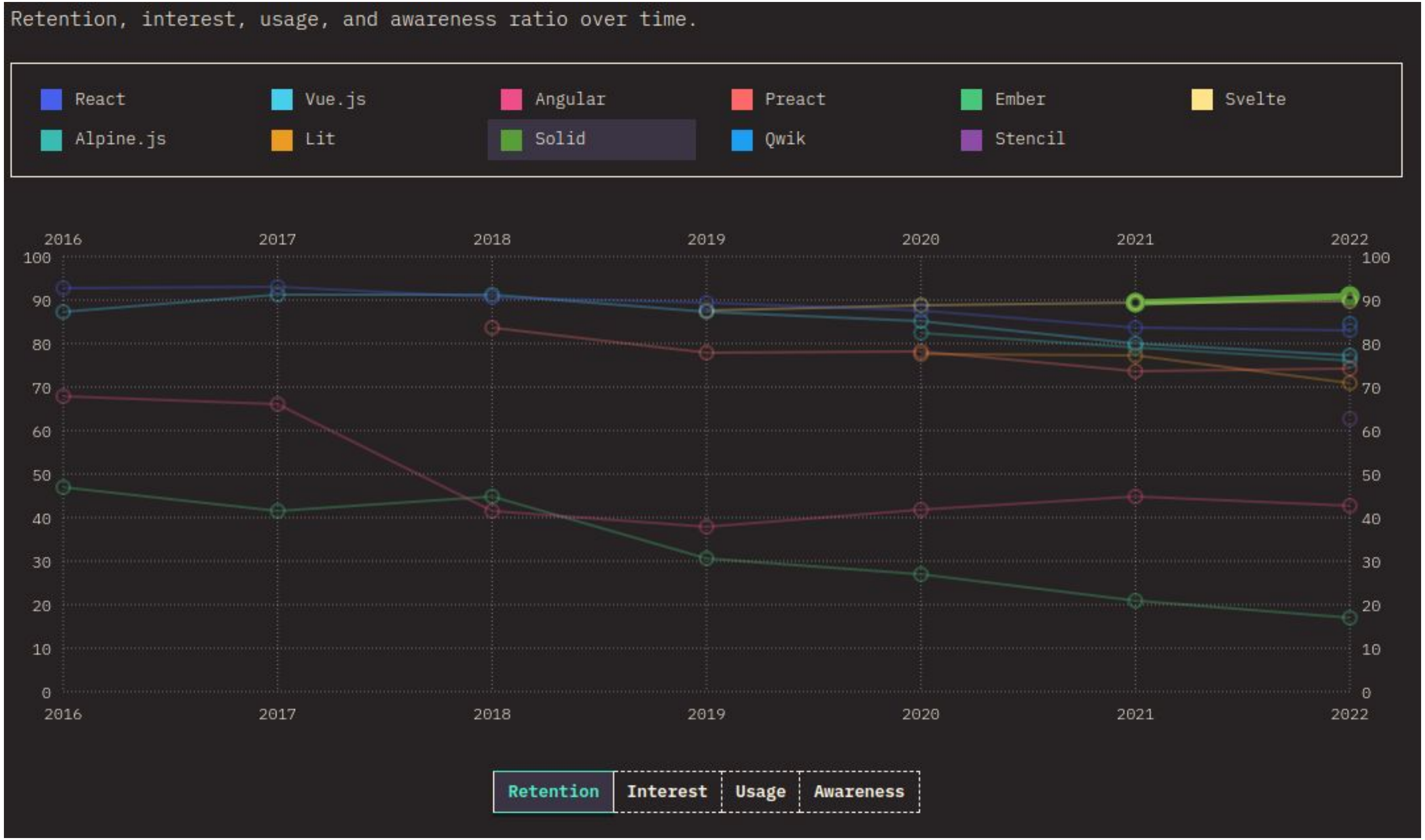
- Key facts & Comparison
- Reactivity
- Components, Props & Bindings
- Control Flow
- Solid Start

State of JS 2022



Retention, interest, usage, and awareness ratio over time.

State of JS 2022



Retention, interest, usage, and awareness ratio over time.

Key Facts

- **JSX**
- Functional components
- Vite
- **Typescript**
- Built around **reactivity**
  - **Surgical** DOM updates
  - Observables/**RxJs** compatibility*

➜ Quite similar to React

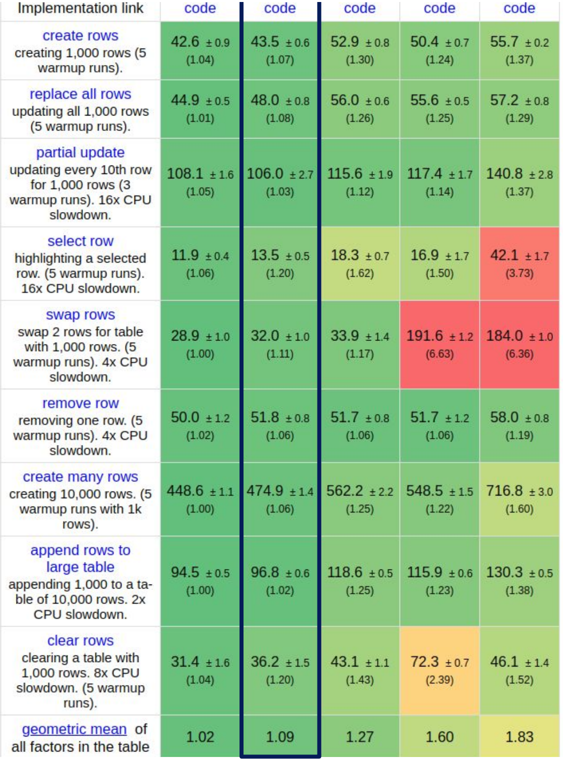
```
function MyButton() {
 return (
   <button>
     I'm a button
   </button>
 );
}

export default function MyApp() {
 return (
   <div>
     <h1>Welcome to my app</h1>
     <MyButton />
   </div>
 );
}
```

Comparison

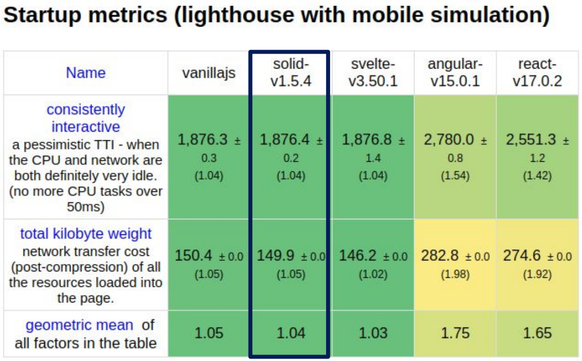
Rendering	Through reactivity	Buildtime	Runtime (virtual DOM)
Runtime	Minimal	Minimal	Full-blown
State Management	Built-in	Built-in	BYO
Reactivity	**Core principle**	Built-in	-
Styling	BYO (vite)	Built-in	BYO
Syntax	Plain JS(X)	DSL	Plain JS(X)

Comparison - Performance

🌳 🏭

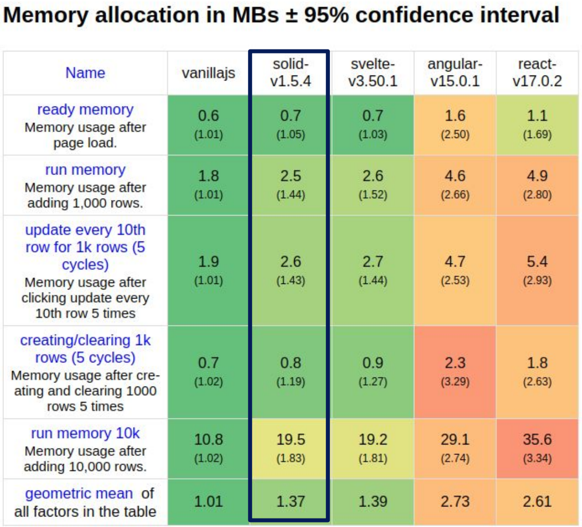


Implementation link	code	code	code	code	code
create rows creating 1,000 rows (5 warmup runs).	42.6 ± 0.9 (1.04)	43.5 ± 0.6 (1.07)	52.9 ± 0.8 (1.30)	50.4 ± 0.7 (1.24)	55.7 ± 0.2 (1.37)
replace all rows updating all 1,000 rows (5 warmup runs).	44.9 ± 0.5 (1.01)	48.0 ± 0.8 (1.08)	56.0 ± 0.6 (1.26)	55.6 ± 0.5 (1.25)	57.2 ± 0.8 (1.29)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	108.1 ± 1.6 (1.05)	106.0 ± 2.7 (1.03)	115.6 ± 1.9 (1.12)	117.4 ± 1.7 (1.14)	140.8 ± 2.8 (1.37)
select row highlighting a selected row. (5 warmup runs). 16x CPU slowdown.	11.9 ± 0.4 (1.06)	13.5 ± 0.5 (1.20)	18.3 ± 0.7 (1.62)	16.9 ± 1.7 (1.50)	42.1 ± 1.7 (3.73)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	28.9 ± 1.0 (1.00)	32.0 ± 1.0 (1.11)	33.9 ± 1.4 (1.17)	191.6 ± 1.2 (6.63)	184.0 ± 1.0 (6.36)
remove row removing one row. (5 warmup runs). 4x CPU slowdown.	50.0 ± 1.2 (1.02)	51.8 ± 0.8 (1.06)	51.7 ± 0.8 (1.06)	51.7 ± 1.2 (1.06)	58.0 ± 0.8 (1.19)
create many rows creating 10,000 rows. (5 warmup runs with 1k rows).	448.6 ± 1.1 (1.00)	474.9 ± 1.4 (1.06)	562.2 ± 2.2 (1.25)	548.5 ± 1.5 (1.22)	716.8 ± 3.0 (1.60)
append rows to large table appending 1,000 to a table of 10,000 rows. 2x CPU slowdown.	94.5 ± 0.5 (1.00)	96.8 ± 0.6 (1.02)	118.6 ± 0.5 (1.25)	115.9 ± 0.6 (1.23)	130.3 ± 0.5 (1.38)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown. (5 warmup runs).	31.4 ± 1.6 (1.04)	36.2 ± 1.5 (1.20)	43.1 ± 1.1 (1.43)	72.3 ± 0.7 (2.39)	46.1 ± 1.4 (1.52)
geometric mean of all factors in the table	1.02	1.09	1.27	1.60	1.83



Startup metrics (lighthouse with mobile simulation)

Name	vanillajs	solid-v1.5.4	svelte-v3.50.1	angular-v15.0.1	react-v17.0.2
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	1,876.3 ± 0.3 (1.04)	1,876.4 ± 0.2 (1.04)	1,876.8 ± 1.4 (1.04)	2,780.0 ± 0.8 (1.54)	2,551.3 ± 1.2 (1.42)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	150.4 ± 0.0 (1.05)	149.9 ± 0.0 (1.05)	146.2 ± 0.0 (1.02)	282.8 ± 0.0 (1.98)	274.6 ± 0.0 (1.92)
geometric mean of all factors in the table	1.05	1.04	1.03	1.75	1.65



Memory allocation in MBs ± 95% confidence interval

Name	vanillajs	solid-v1.5.4	svelte-v3.50.1	angular-v15.0.1	react-v17.0.2
ready memory Memory usage after page load.	0.6 (1.01)	0.7 (1.05)	0.7 (1.03)	1.6 (2.50)	1.1 (1.69)
run memory Memory usage after adding 1,000 rows.	1.8 (1.01)	2.5 (1.44)	2.6 (1.52)	4.6 (2.66)	4.9 (2.80)
update every 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	1.9 (1.01)	2.6 (1.43)	2.7 (1.44)	4.7 (2.53)	5.4 (2.93)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	0.7 (1.02)	0.8 (1.19)	0.9 (1.27)	2.3 (3.29)	1.8 (2.63)
run memory 10k Memory usage after adding 10,000 rows.	10.8 (1.02)	19.5 (1.83)	19.2 (1.81)	29.1 (2.74)	35.6 (3.34)
geometric mean of all factors in the table	1.01	1.37	1.39	2.73	2.61

https://krausest.github.io/js-framework-benchmark/2023/table_chrome_109.0.5414.87.html

inovex

Component Basics

- Functions
  - **executed once**
  - Return JSX
- Nesting is possible (duh)
- Structuring code
  - × State
  - × Life-cycle*
  - ✓ JS Scope (duh)

```
function MyButton() {
 return (
   <button>
     I'm a button
   </button>
 );
}

export default function MyApp() {
 return (
   <div>
     <h1>Welcome to my app</h1>
     <MyButton />
   </div>
 );
}
```

Reactivity: Signals

- **setter** and **getter**
- **all modification** must use setter
- **all access** must use getter

```
type Getter<T> = () => T
type Setter<T> = (value: T) => void

function createSignal<T>(
  initial: T,
  options?: {
    equals: ((prev: T, next: T) => boolean)
  }
): [Getter<T>, Setter<T>]
```

🙀 Data without signal ➜ No reactivity

Reactivity: Effects

- reactive scope
- for **side-effects**
  - internally for DOM-updates
  - also for custom behavior

```
function createEffect<T>(
      fn: (v: T) => T,
      initial?: T
): void

function createEffect(
      fn: () => void
): void
```

🙀 Setting signals from effects is dangerous

Reactivity: Resource

- **asynchronous** resource loading
- returns data, errors, loading state…
- optionally react to changes in signals

```
const [data, { mutate, refetch }] = createResource(fetchData)
const [data, { mutate, refetch }] = createResource(sourceSignal, fetchData)
```

Components: Props

- pass data to children
- reactive through getters
  - must not be destructured by consumer
  - JSX destructuring

```
// before compilation
<MyComp dynamic={mySignal()}>
 <Child />
</MyComp>

// after compilation
MyComp({
 get dynamic() { return mySignal() },
 get children() { return Child() }
});
```

🙀 Destructuring props prevents DOM updates

Control flow

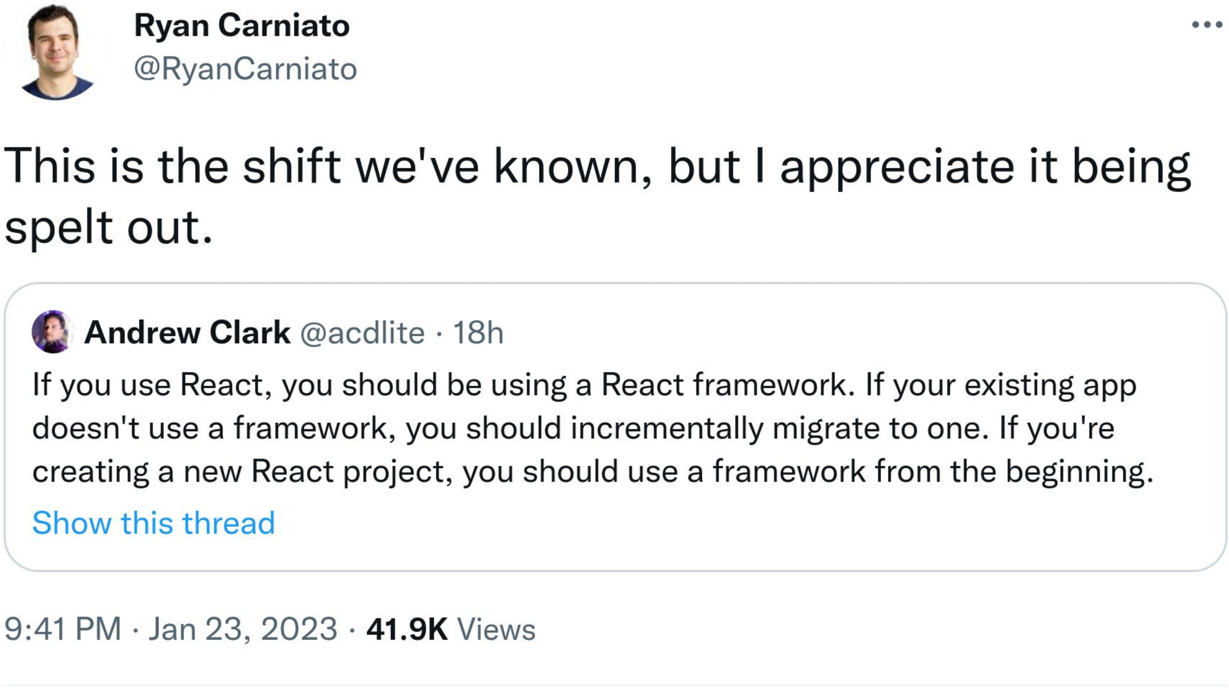
- Components **executed once**
  - Records observer/observable relationship
  - DOM updates through effects
- JS conditions and loops conflict with recording,
- `<Show>`,`<For>`,`<Switch>`/`<Match>`,...

🙀 JS control flow prevents DOM updates

SolidStart

https://start.solidjs.com

Meta Frameworks - JS’s current thing



Ryan Carniato
@RyanCarniato

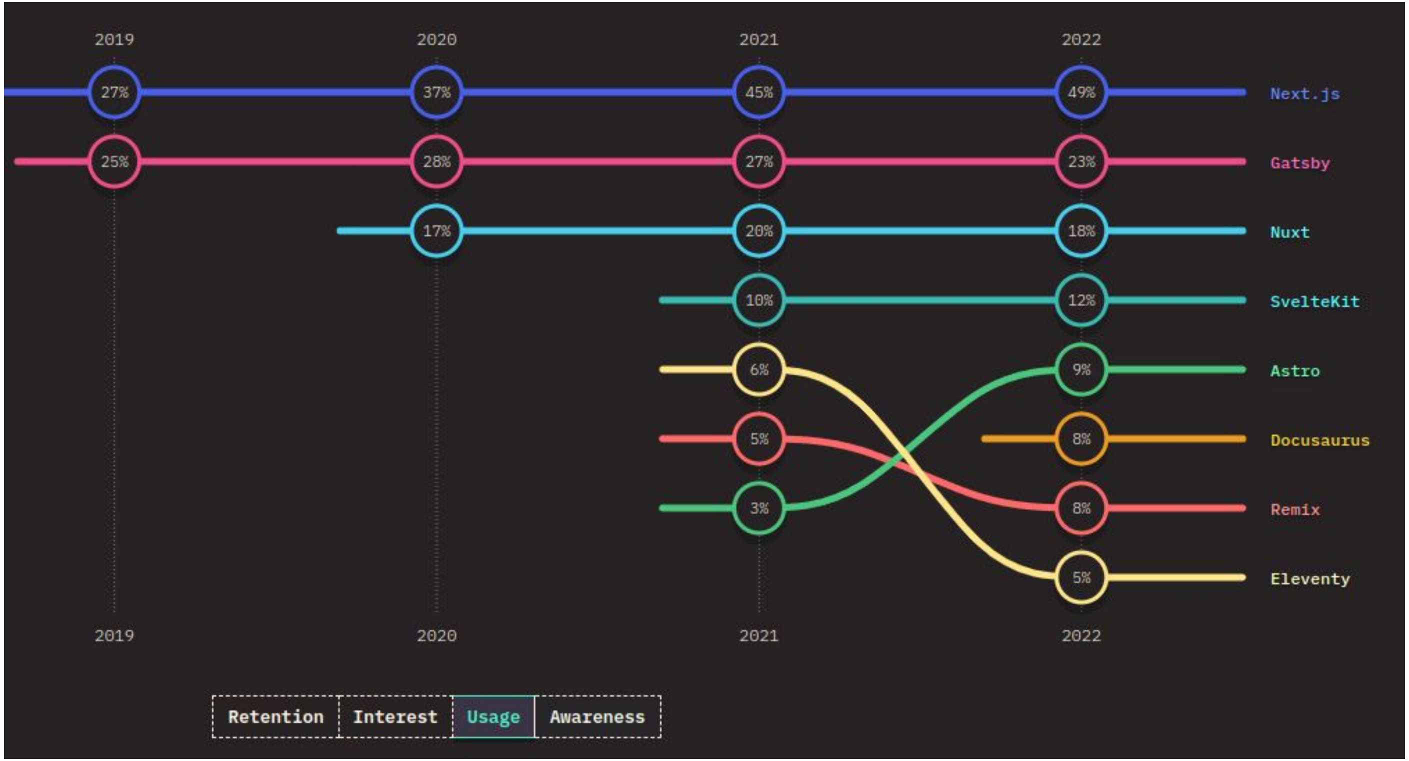
This is the shift we've known, but I appreciate it being spelt out.

> **Andrew Clark** @acdlite · 18h
> If you use React, you should be using a React framework. If your existing app doesn't use a framework, you should incrementally migrate to one. If you're creating a new React project, you should use a framework from the beginning.
> Show this thread

9:41 PM · Jan 23, 2023 · **41.9K** Views

https://twitter.com/RyanCarniato/status/1617623647803539456

Meta Frameworks - Usage



https://2022.stateofjs.com/en-US/libraries/rendering-frameworks/

SolidStart Meta Framework

- Client-side rendering (CSR)
- Server-side rendering (SSR)
- Streaming SSR
- Static site generation (SSG)

Isomorphic code approach

SolidStart Project Setup

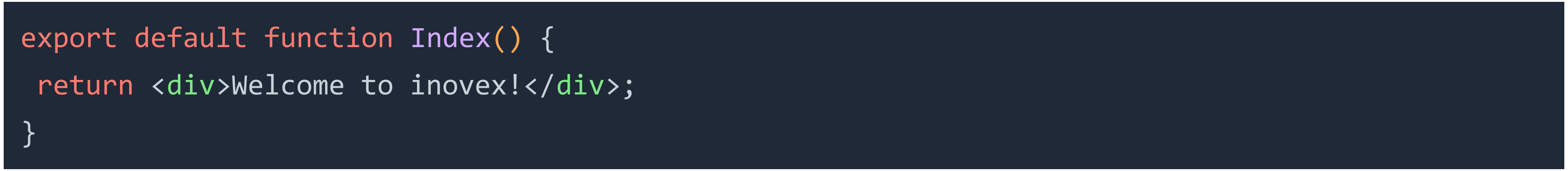
`npm create solid`

```
node_modules/
public/
src/
├── routes/
│   ├── index.tsx
├── entry-client.tsx
├── entry-server.tsx
├── root.tsx
```

File Based Routing

- inovex.de/ ➜ /routes/index.tsx
- inovex.de/blog ➜ /routes/blog.tsx
- inovex.de/standorte/er ➜ /routes/standorte/er.tsx
- inovex.de/jobs/:id/:lang ➜ /routes/jobs/[id]/[lang].tsx

Default Export Component:



```
export default function Index() {
 return <div>Welcome to inovex!</div>;
}
```

Navigating

`<A>` component:

```
import { A } from 'solid-start';

export default function Index() {
 return (
   <div>
     <A href="/about">About</A>
   </div>
 );
}
```

`useNavigate`:

```
const navigate = useNavigate();
if (unauthorized) {
 navigate("/login", {
   replace: true,
   scroll: true,
 });
}
```

API Routes

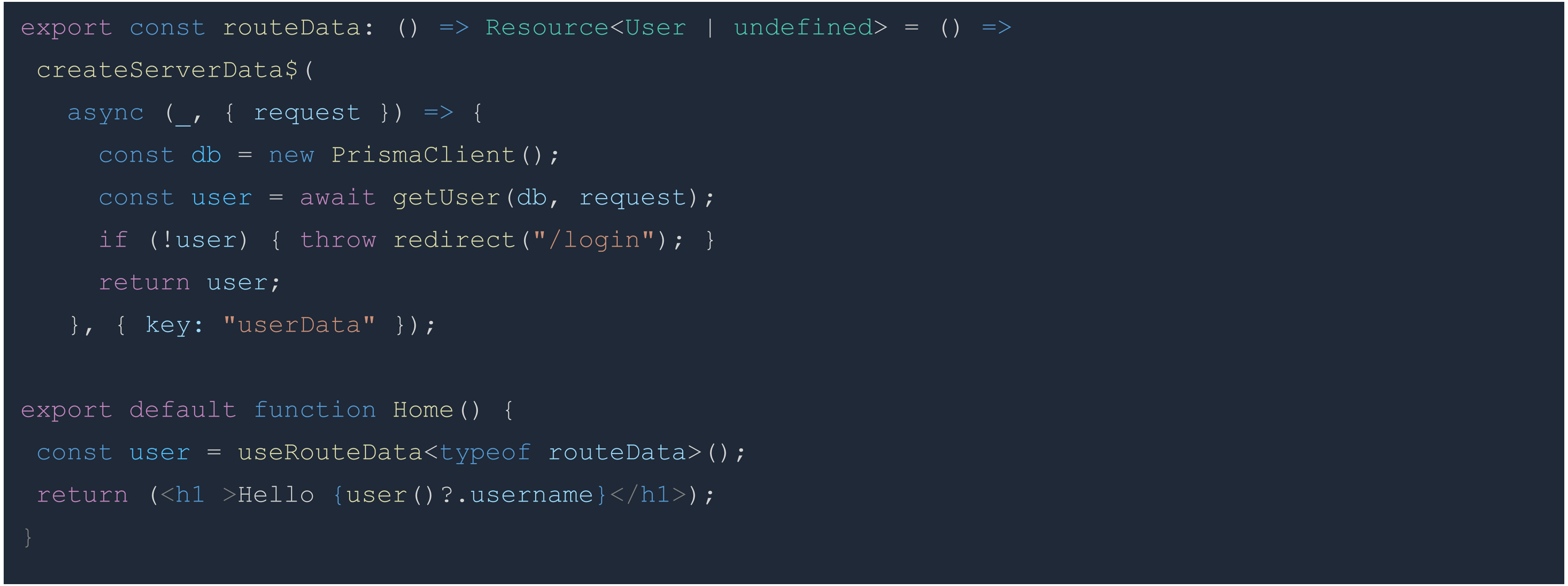
```
// handles HTTP GET requests to /api/cat-facts
export function GET() {
 return new Response("Cats are awesome 😸");
}
export function POST() {}
export function PATCH() {}
export function DELETE() {}
```

WARNING
A route can only export either a default UI component or a `GET` handler. You cannot export both.

State with Cookie Sessions

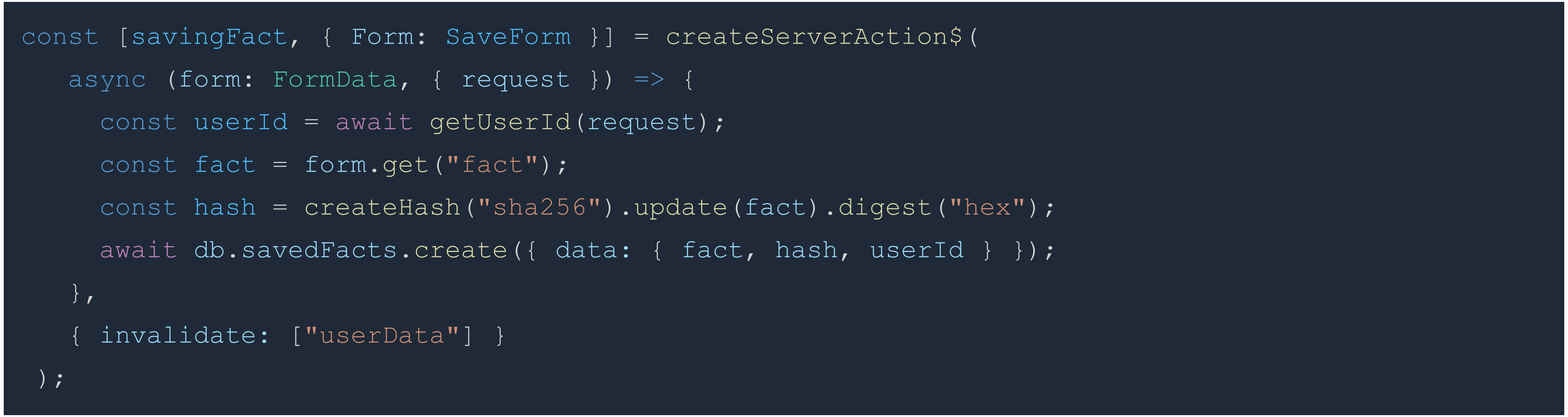
- Solid “borrowed” Remix’s session cookie storage
- User IDs in encrypted **http only** cookies
- SSR uses the cookie during initial page loads

useRouteData



```
export const routeData: () => Resource<User | undefined> = () =>
 createServerData$(
   async (_, { request }) => {
     const db = new PrismaClient();
     const user = await getUser(db, request);
     if (!user) { throw redirect("/login"); }
     return user;
   }, { key: "userData" });

export default function Home() {
 const user = useRouteData<typeof routeData>();
 return (<h1 >Hello {user()?.username}</h1>);
}
```

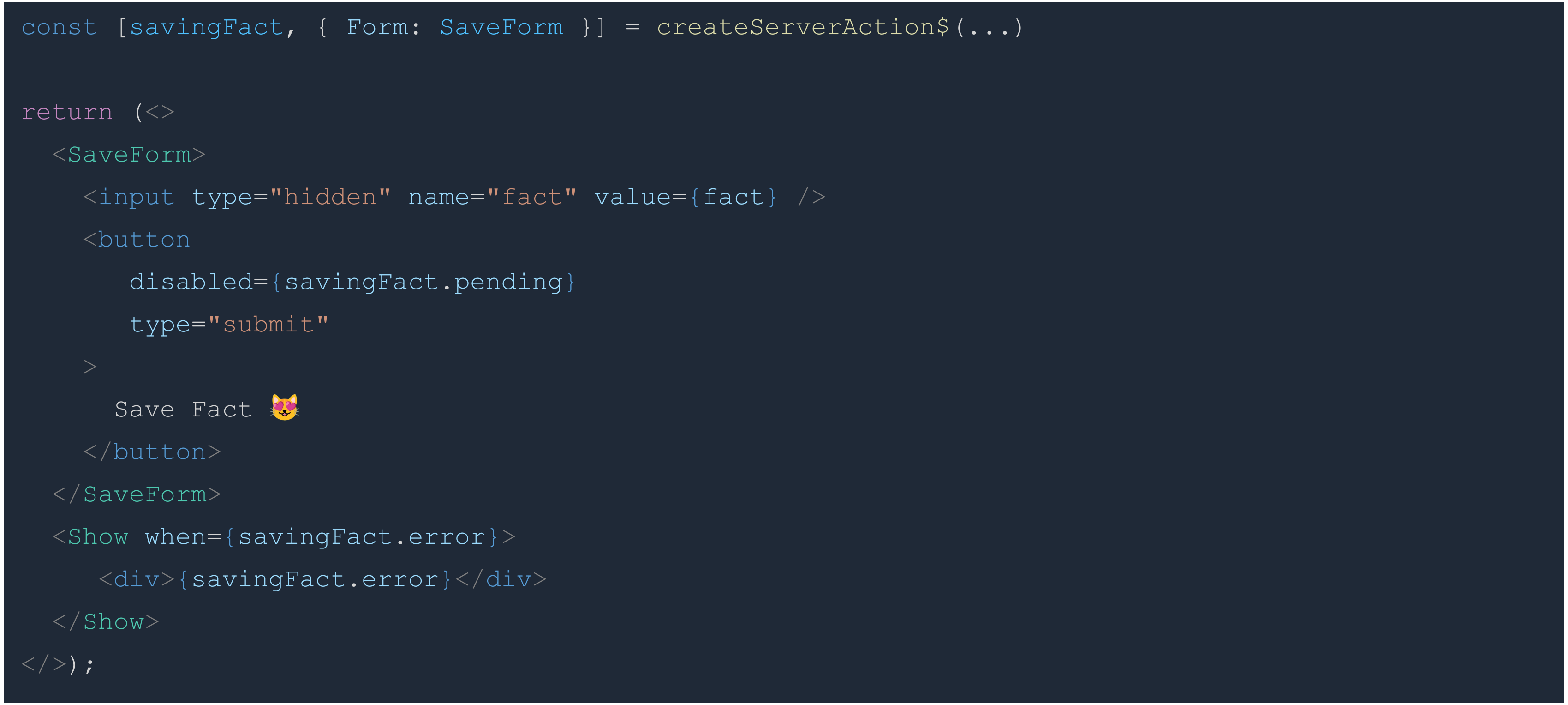
createServerAction$ - create



```
const [savingFact, { Form: SaveForm }] = createServerAction$(
   async (form: FormData, { request }) => {
     const userId = await getUserId(request);
     const fact = form.get("fact");
     const hash = createHash("sha256").update(fact).digest("hex");
     await db.savedFacts.create({ data: { fact, hash, userId } });
   },
   { invalidate: ["userData"] }
 );
```

inovex

createServerAction$ - call



```
const [savingFact, { Form: SaveForm }] = createServerAction$(...)

return (<>
  <SaveForm>
    <input type="hidden" name="fact" value={fact} />
    <button
       disabled={savingFact.pending}
       type="submit"
    >
      Save Fact 😻
    </button>
  </SaveForm>
  <Show when={savingFact.error}>
     <div>{savingFact.error}</div>
  </Show>
</>);
```

DEMO TIME

https://github.com/meldron/cat-facts

Solid Start Summary

- 😻 Easy Routing & APIs
- 😻 Solid Isomorphic Integration
- 😻 Data Loading & Form Submission
- 😻 Great Community

- 😿 Sparse Documentation
- 😿 Few tutorials
- 😿 Buggy Examples
- 😿 Beta: everything can & will change (Hybrid Routing + Minimal Hydration)

Resources

- SolidJS: Tutorial, API, solid-primitives
- https://start.solidjs.com/
- **Discord**
- Videos:
  - @ryansolid
  - The World Beyond Components
  - Fireship: Solid in 100 Seconds, a solid start
- Podcasts
  - LogRocket: SolidJS & SolidStart with Ryan Carniato
  - Modern Web: Introduction to SolidJS

Thanks!

Tilman Adler

tilman.adler@inovex.de

Bernd Kaiser

bernd.kaiser@inovex.de

inovex