Fresh: a full stack web framework for Deno

About Me

- Senior Software Engineer at [OpenSauced]
- From Montreal, Quebec, Canada
- I'm @nickytonline everywhere
- Also find me at nickyt.co
- Streaming at nickyt.live
- Not a big fan of spiders

What We’ll Cover

What We'll Cover

- What is Fresh?
- Web Standards
- Features
- Demo
- Questions

What is Fresh?

What is Fresh?

Hold on! First, we need to talk about Deno.

What is Deno?

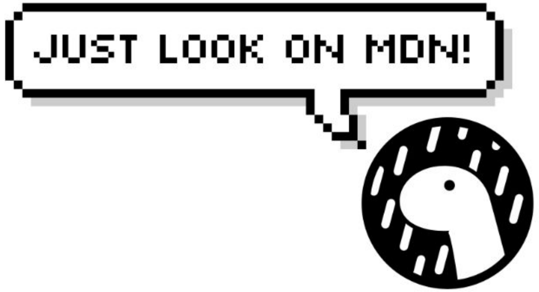
- Runtime for JavaScript, TypeScript and Web Assembly (WASM) that uses V8
- For the web, it runs on the edge
- Create command line interfaces (CLIs)
- Built-in linter
- Built-in code formatter
- Built-in test runner
- Node.js interoperability via node specifiers
- npm interoperability via npm specifiers and CDNs

Web Standards

Web Standards

Deno uses web standards

- import maps
- fetch
- Request and Response
- etc.



Web Standards

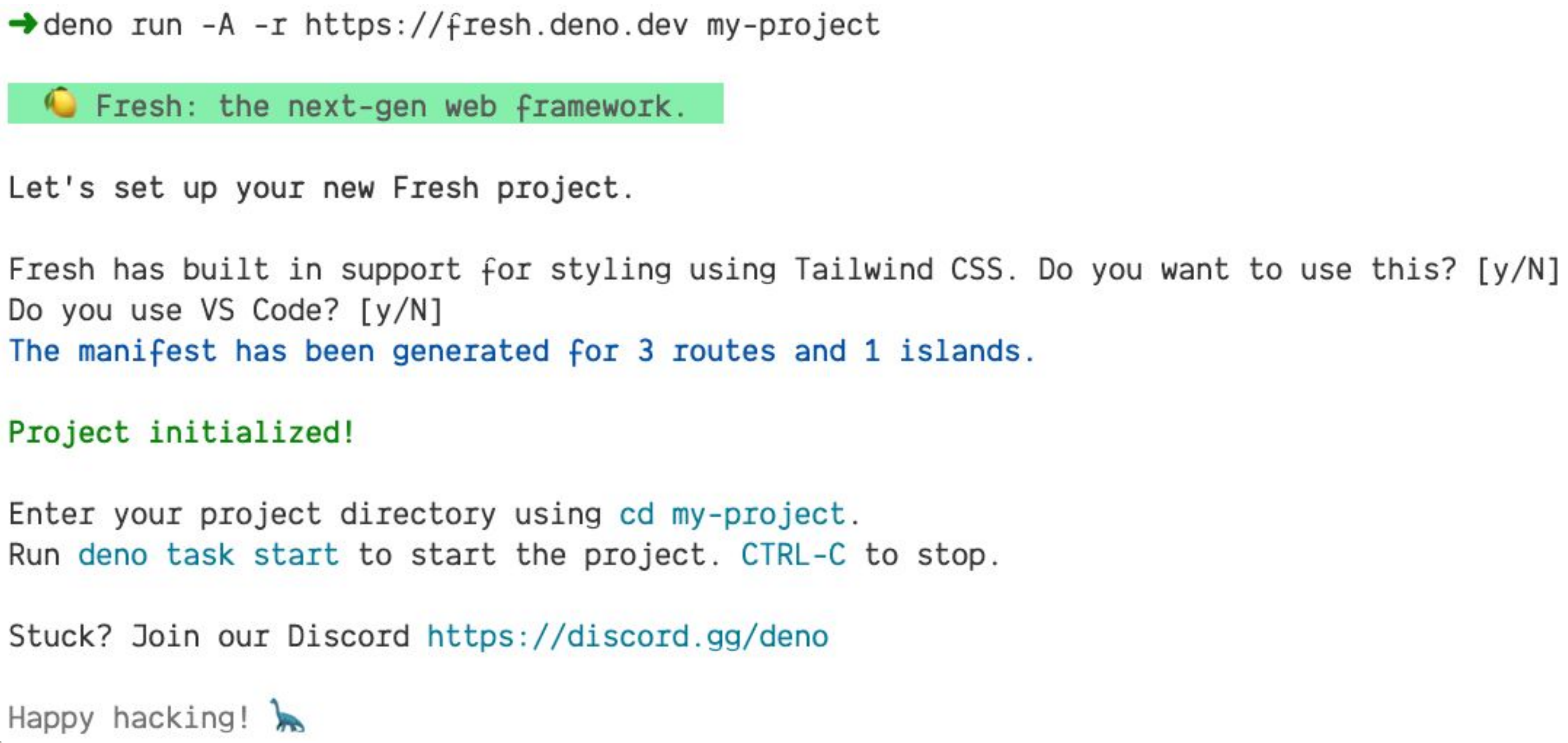
Deno is a part of the Web-interoperable Runtimes Community Group (WinterCG)

- The group provides a space for JavaScript runtimes to collaborate on API interoperability

Where were we? 😎

```
deno run -A -r https://fresh.deno.dev my-project
```

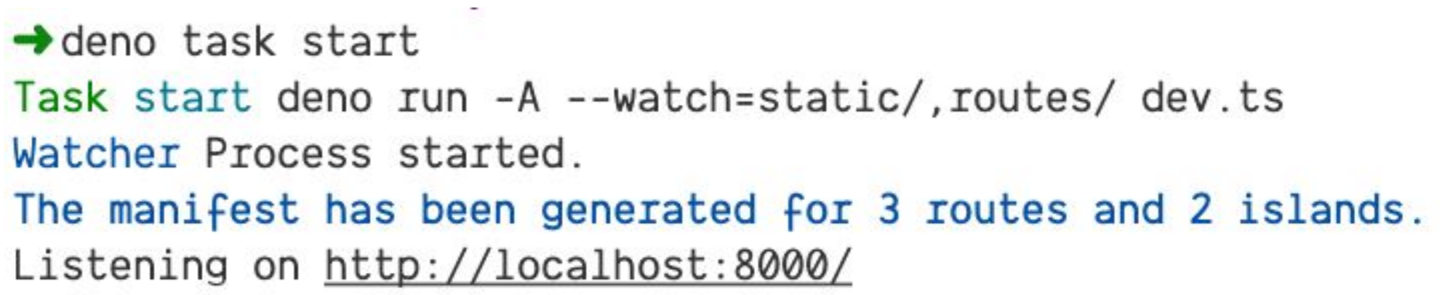
Fresh installation



```
➜ deno run -A -r https://fresh.deno.dev my-project

 🍋 Fresh: the next-gen web framework.

Let's set up your new Fresh project.

Fresh has built in support for styling using Tailwind CSS. Do you want to use this? [y/N]
Do you use VS Code? [y/N]
The manifest has been generated for 3 routes and 1 islands.

Project initialized!

Enter your project directory using cd my-project.
Run deno task start to start the project. CTRL-C to stop.

Stuck? Join our Discord https://discord.gg/deno

Happy hacking! 🦕
```

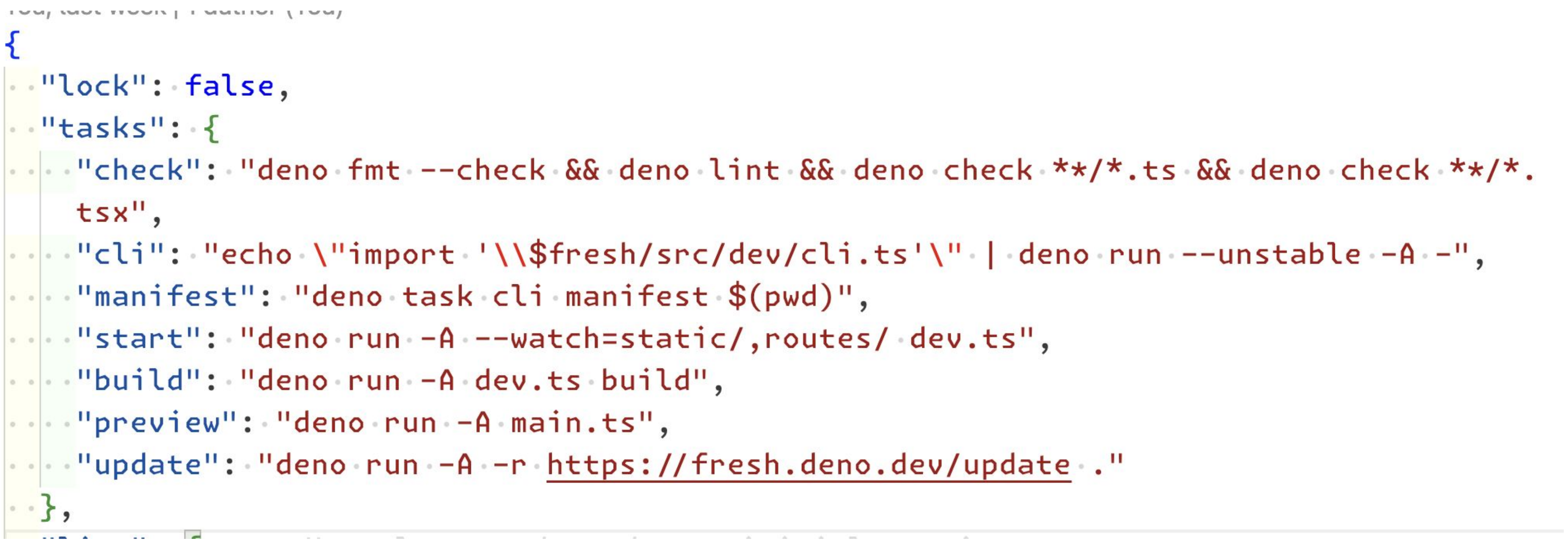
nickyt.co

Let’s start Fresh



```
➜ deno task start
Task start deno run -A --watch=static/,routes/ dev.ts
Watcher Process started.
The manifest has been generated for 3 routes and 2 islands.
Listening on http://localhost:8000/
```

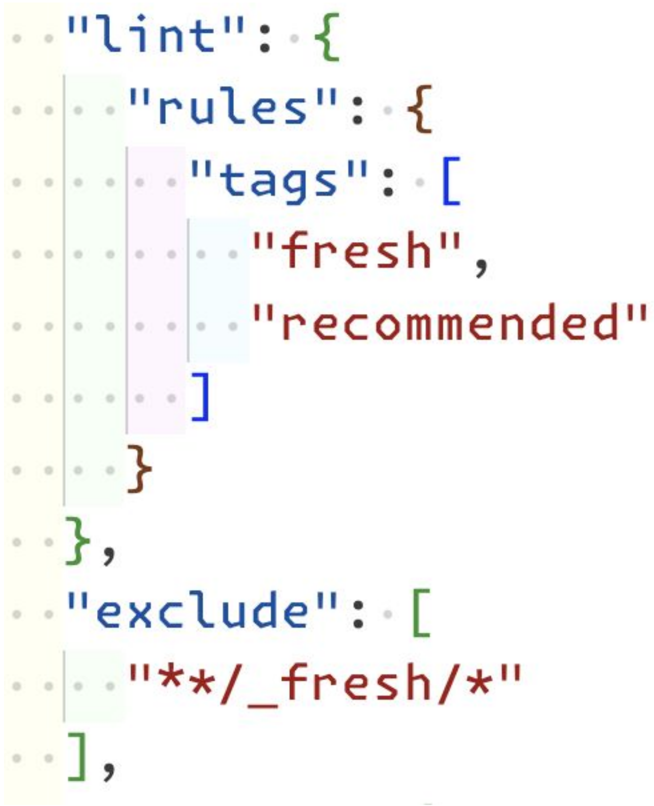
Let’s start Fresh



```
You, last week | 1 author (You)
{
  "lock": false,
  "tasks": {
    "check": "deno fmt --check && deno lint && deno check **/*.ts && deno check **/*.
    tsx",
    "cli": "echo \"import '\\$fresh/src/dev/cli.ts'\" | deno run --unstable -A -",
    "manifest": "deno task cli manifest $(pwd)",
    "start": "deno run -A --watch=static/,routes/ dev.ts",
    "build": "deno run -A dev.ts build",
    "preview": "deno run -A main.ts",
    "update": "deno run -A -r https://fresh.deno.dev/update ."
  },
```

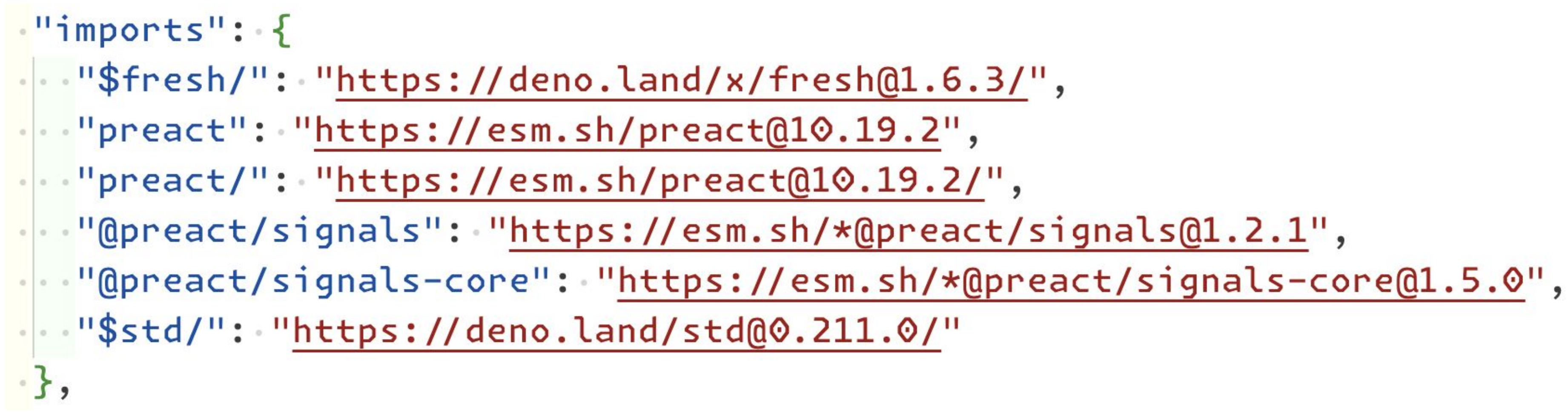
nickyt.co

Let’s start Fresh



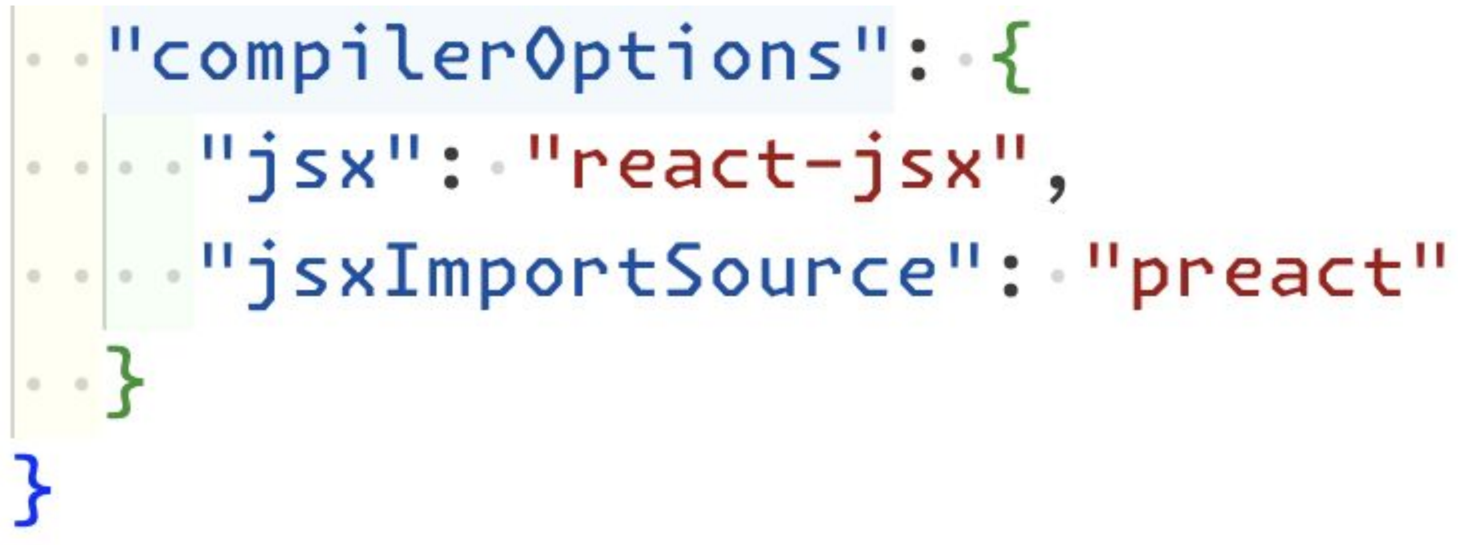
```
  "lint": {
    "rules": {
      "tags": [
        "fresh",
        "recommended"
      ]
    }
  },
  "exclude": [
    "**/_fresh/*"
  ],
```

Let’s start Fresh



```
"imports": {
  "$fresh/": "https://deno.land/x/fresh@1.6.3/",
  "preact": "https://esm.sh/preact@10.19.2",
  "preact/": "https://esm.sh/preact@10.19.2/",
  "@preact/signals": "https://esm.sh/*@preact/signals@1.2.1",
  "@preact/signals-core": "https://esm.sh/*@preact/signals-core@1.5.0",
  "$std/": "https://deno.land/std@0.211.0/"
},
```

Let’s start Fresh



```
  "compilerOptions": {
    "jsx": "react-jsx",
    "jsxImportSource": "preact"
  }
}
```

nickyt.co

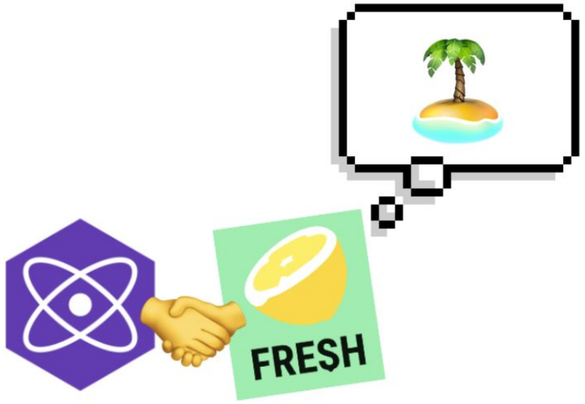
What is Fresh?

Fresh is a full-stack web framework that runs on Deno.

- Server-side rendered (SSR) framework
- Just-in-time (JIT) rendering on the edge
- Provides TypeScript support out of the box
- Nothing to configure to get up and running

What is Fresh?

- No JavaScript delivered by default
- Islands Architecture for client interactivity
- Preact (server-side and client-side)
- JSX support (thanks Preact and TypeScript!)



Features

Features

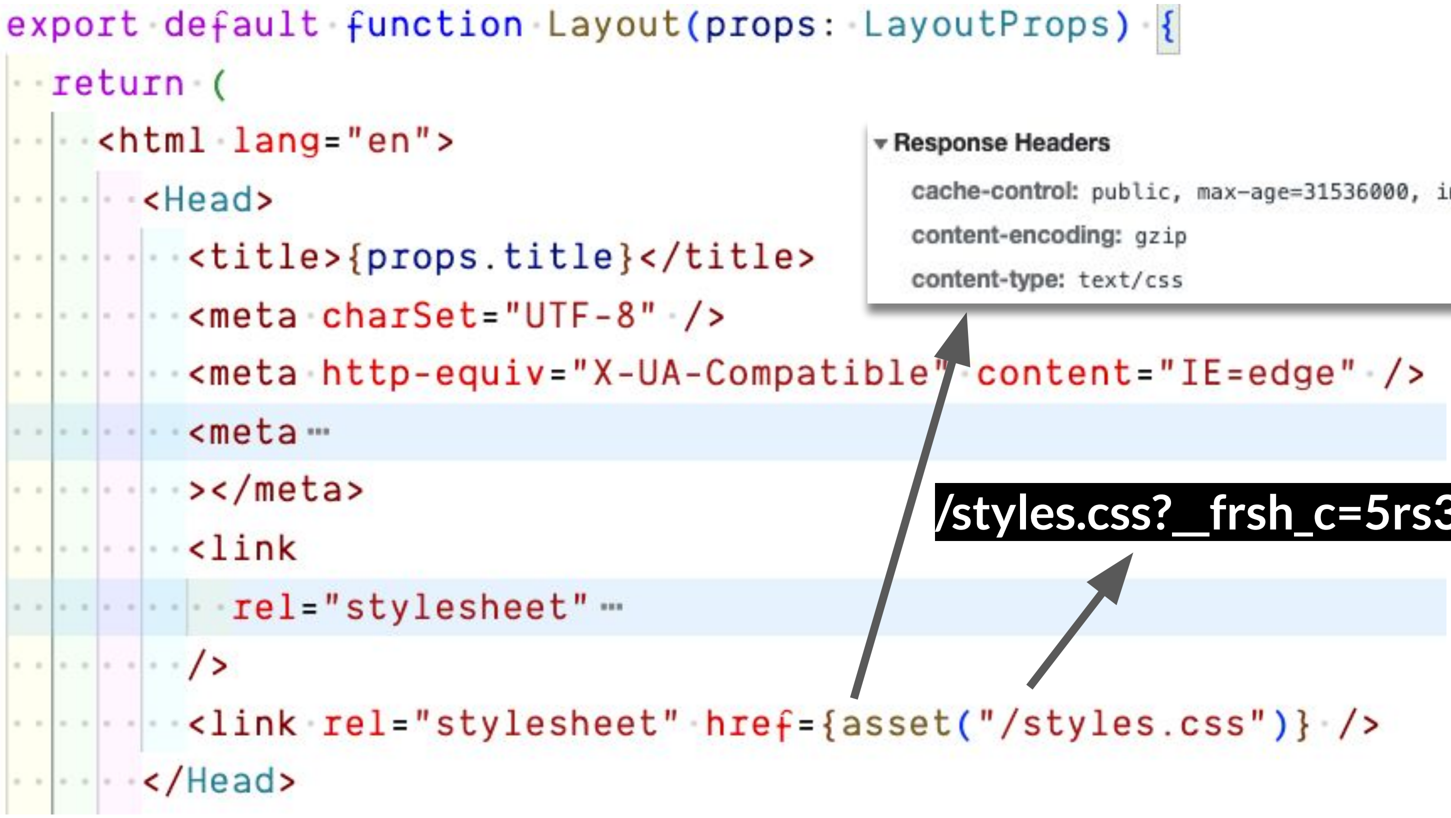
- Static files
- Routes & Routing
- Data Fetching
- Middleware
- Error pages
- Styling
- Islands
- Signals
- Partials

Static Files

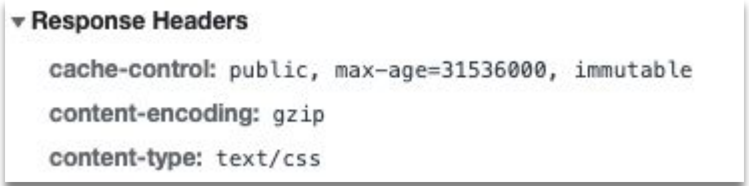
Static Files

- All static assets reside in the `/static` folder
- No caching headers by default for static assets
    - Exceptions: **src** and **srcset** attributes on **<img />** and **<source/>** elements
- Use the **asset** helper to automatically cache for a year

Static Files



```
export default function Layout(props: LayoutProps) {
  return (
    <html lang="en">
      <Head>
        <title>{props.title}</title>
        <meta charSet="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta ⋯
        ></meta>
        <link
          rel="stylesheet" ⋯
        />
        <link rel="stylesheet" href={asset("/styles.css")} />
      </Head>
```



```
▾ Response Headers
   cache-control: public, max-age=31536000, immutable
   content-encoding: gzip
   content-type: text/css
```

/styles.css?_frsh_c=5rs329q7fe2g

nickyt.co

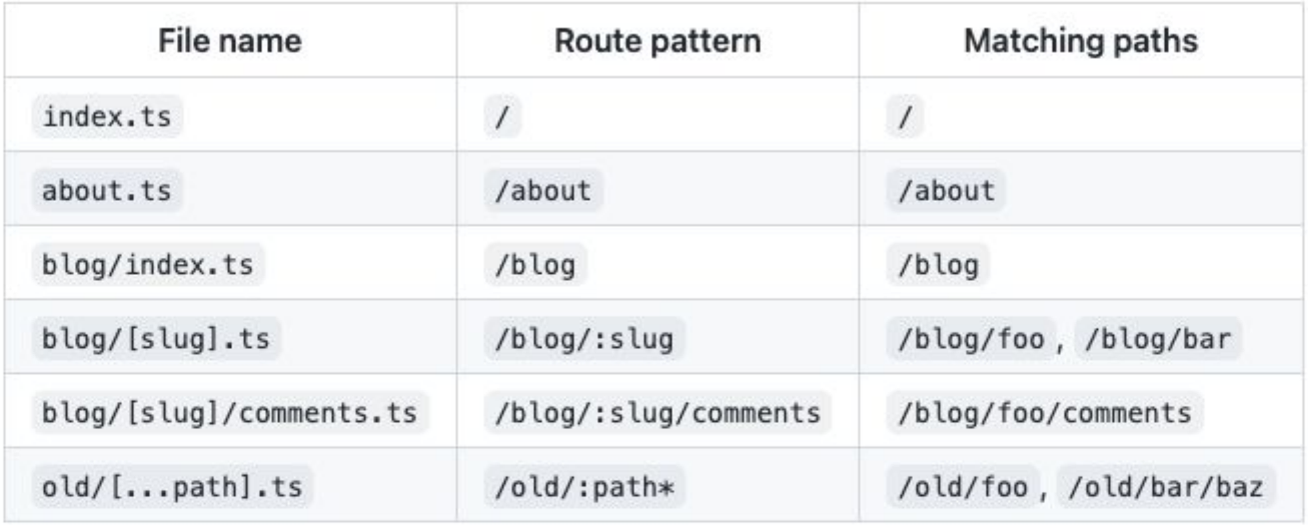
Routes & Routing

Routes & Routing

- 3 kinds of routes:
    - Handler route (API route)
    - Component route (Pages)
    - Hybrid route (handler and component)
        - form that POSTs back,
        - search page that GETs search results

Routes & Routing

- Routes can be dynamic
  - Table from https://fresh.deno.dev/docs/concepts/routing



File name	Route pattern	Matching paths
`index.ts`	`/`	`/`
`about.ts`	`/about`	`/about`
`blog/index.ts`	`/blog`	`/blog`
`blog/[slug].ts`	`/blog/:slug`	`/blog/foo`, `/blog/bar`
`blog/[slug]/comments.ts`	`/blog/:slug/comments`	`/blog/foo/comments`
`old/[...path].ts`	`/old/:path*`	`/old/foo`, `/old/bar/baz`

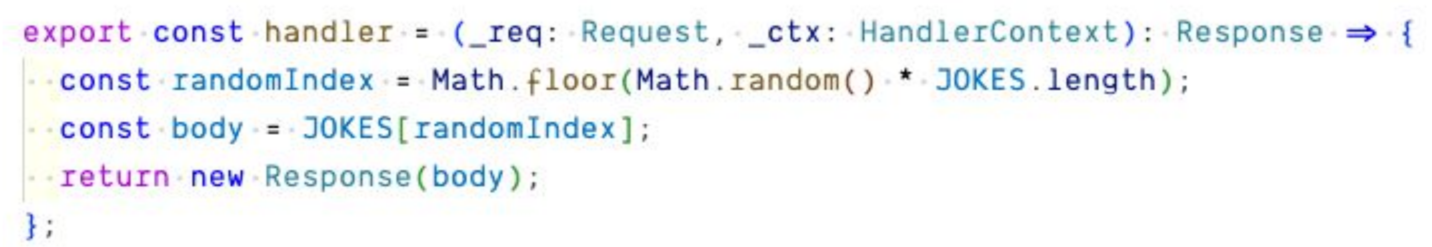
Data Fetching

Data Fetching

Handler routes and Hybrid routes handle data fetching

Data Fetching

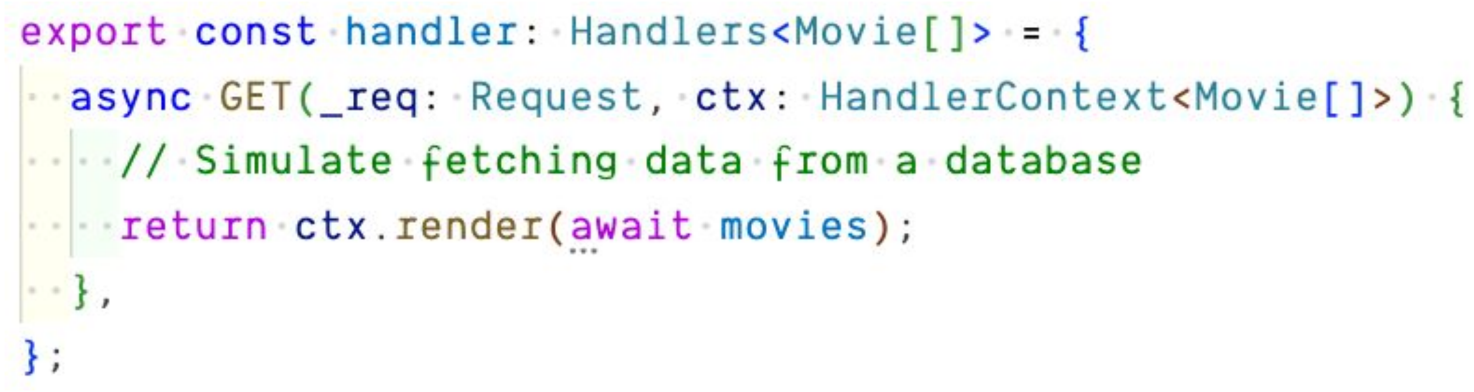
- For handler routes (APIs), export function that returns a response



```
export const handler = (_req: Request, _ctx: HandlerContext): Response => {
  const randomIndex = Math.floor(Math.random() * JOKES.length);
  const body = JOKES[randomIndex];
  return new Response(body);
};
```

Data Fetching

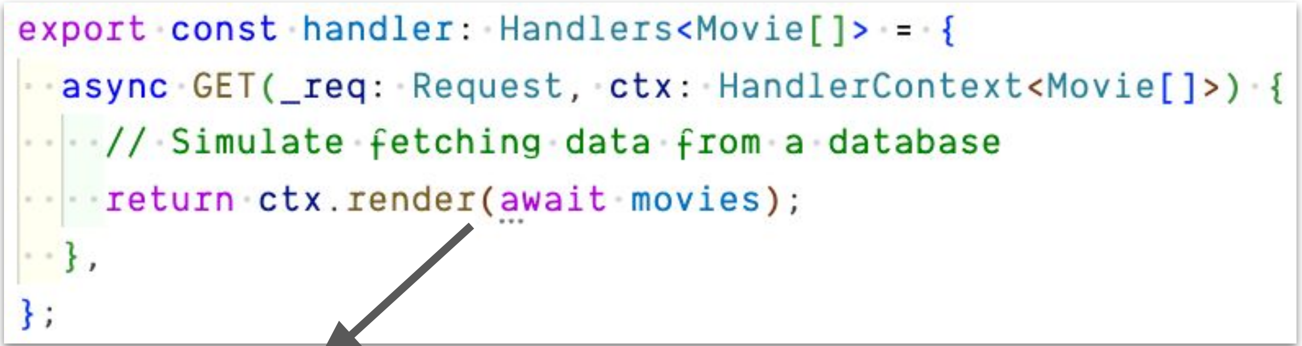
- For hybrid routes, define functions for actions in an exported variable called **handler**
- Name async functions after HTTP verbs: **GET**, **POST** etc.



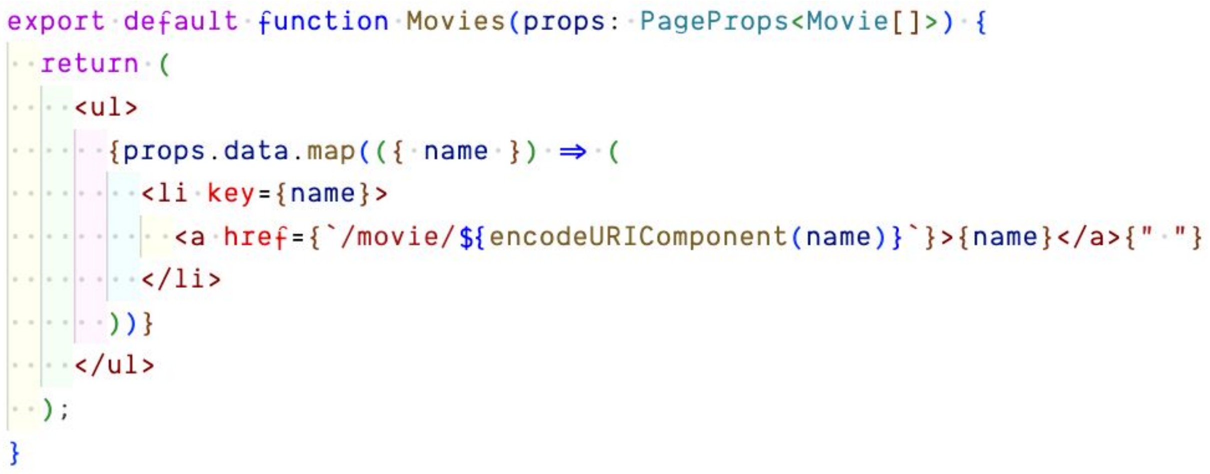
```
export const handler: Handlers<Movie[]> = {
  async GET(_req: Request, ctx: HandlerContext<Movie[]>) {
    // Simulate fetching data from a database
    return ctx.render(await movies);
  },
};
```

nickyt.co

Data Fetching



```
export const handler: Handlers<Movie[]> = {
  async GET(_req: Request, ctx: HandlerContext<Movie[]>) {
    // Simulate fetching data from a database
    return ctx.render(await movies);
  },
};
```



```
export default function Movies(props: PageProps<Movie[]>) {
  return (
    <ul>
      {props.data.map(({ name }) => (
        <li key={name}>
          <a href={`/movie/${encodeURIComponent(name)}`}>{name}</a>{" "}
        </li>
      ))}
    </ul>
  );
}
```

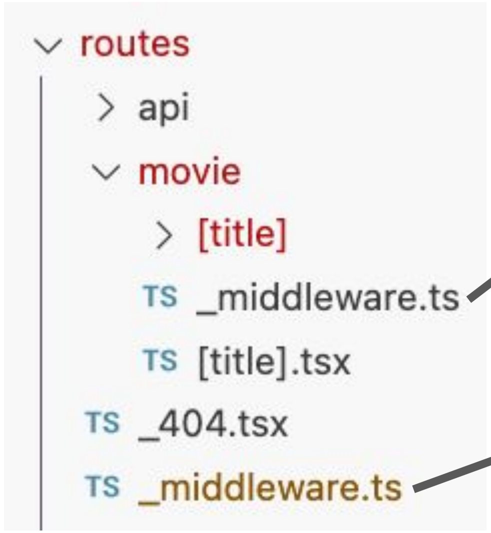
nickyt.co

Middleware

Middleware

- Needs to be named _middleware.ts.
- Resides in the /routes folder
- Multiple middlewares are supported

Middleware



Movie subfolder middleware runs 2nd

```
const response = await ctx.next();
response.headers.set("x-movie-page", "true");
```

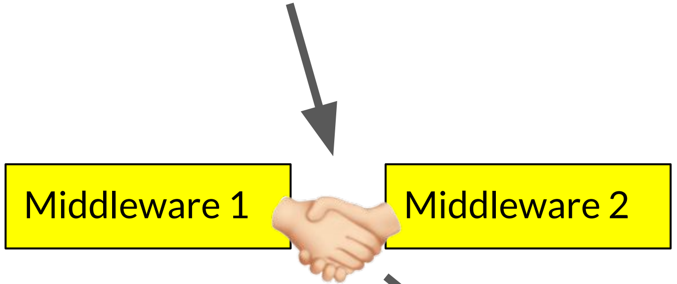
Root middleware runs 1st

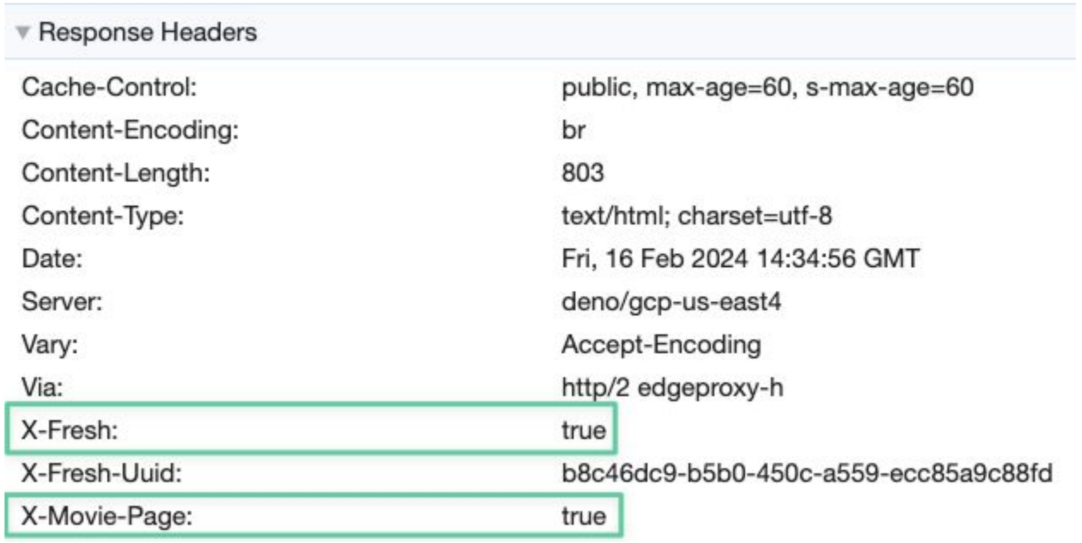
```
const response = await ctx.next();
response.headers.set("x-fresh", "true");
```

nickyt.co

Middleware

http://localhost:8000/movies/1





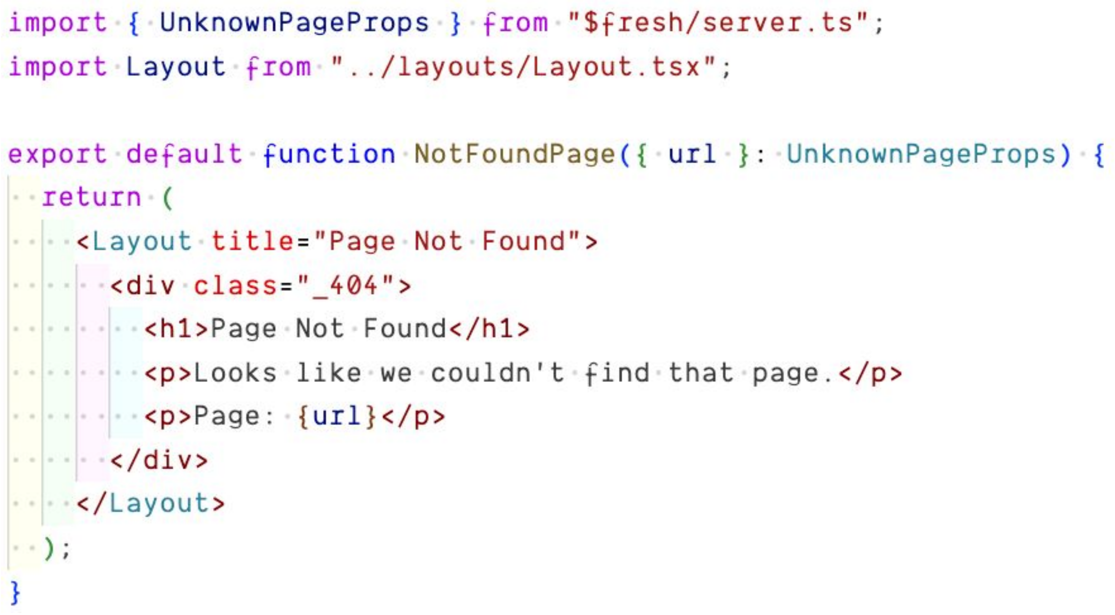
▼ Response Headers

Cache-Control:	public, max-age=60, s-max-age=60
Content-Encoding:	br
Content-Length:	803
Content-Type:	text/html; charset=utf-8
Date:	Fri, 16 Feb 2024 14:34:56 GMT
Server:	deno/gcp-us-east4
Vary:	Accept-Encoding
Via:	http/2 edgeproxy-h
X-Fresh:	true
X-Fresh-Uuid:	b8c46dc9-b5b0-450c-a559-ecc85a9c88fd
X-Movie-Page:	true

Error Pages

Error Pages

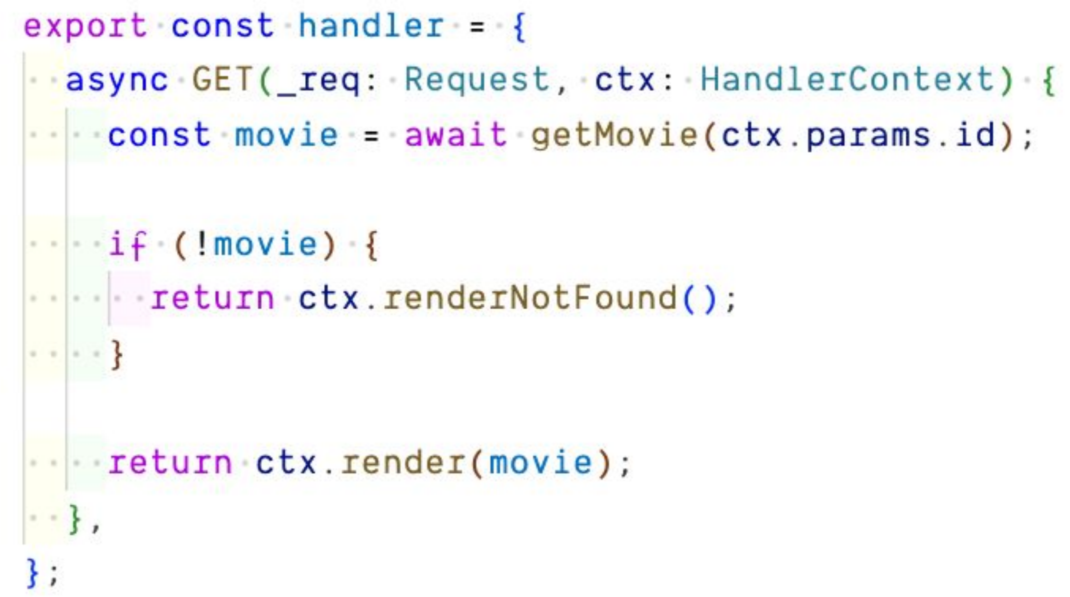
- Create custom error pages: 404, 500 etc.
- Must be in the root of the **/routes** folder



```
import { UnknownPageProps } from "$fresh/server.ts";
import Layout from "../layouts/Layout.tsx";

export default function NotFoundPage({ url }: UnknownPageProps) {
  return (
    <Layout title="Page Not Found">
      <div class="_404">
        <h1>Page Not Found</h1>
        <p>Looks like we couldn't find that page.</p>
        <p>Page: {url}</p>
      </div>
    </Layout>
  );
}
```

Error Pages

- For a dynamic route, call **ctx.renderNotFound()** to render the 404 page if page does not exist



```
export const handler = {
  async GET(_req: Request, ctx: HandlerContext) {
    const movie = await getMovie(ctx.params.id);

    if (!movie) {
      return ctx.renderNotFound();
    }

    return ctx.render(movie);
  },
};
```

nickyt.co

Styling

Styling

- Fresh can use Twind, a server-side rendered implementation of Tailwind
- **Modern CSS is pretty awesome**

  <link rel=”stylesheet” href=”/styles.css” />
- Expect innovation here from userland

Islands

🏝️

Islands

- Let’s talk islands architecture
- Pockets of interactivity

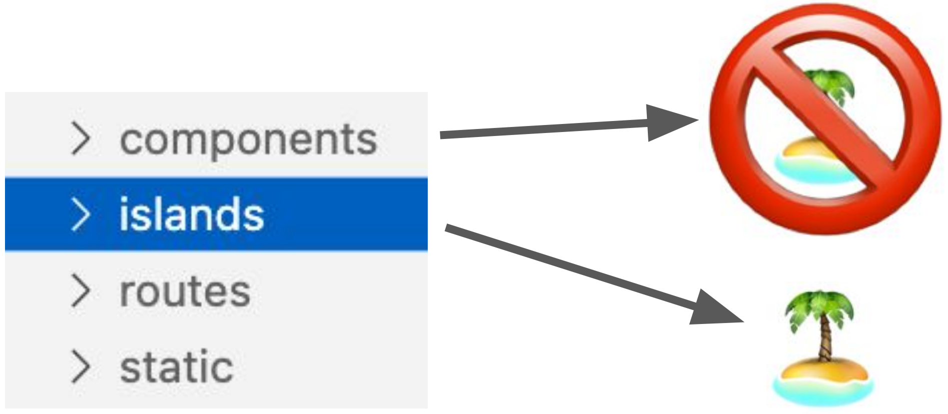
Islands

“The general idea of an “Islands” architecture is deceptively simple: render HTML pages on the server, and inject placeholders or slots around highly dynamic regions. These placeholders/slots contain the server-rendered HTML output from their corresponding widget. They denote regions that can then be "hydrated" on the client into small self-contained widgets, reusing their server-rendered initial HTML.”

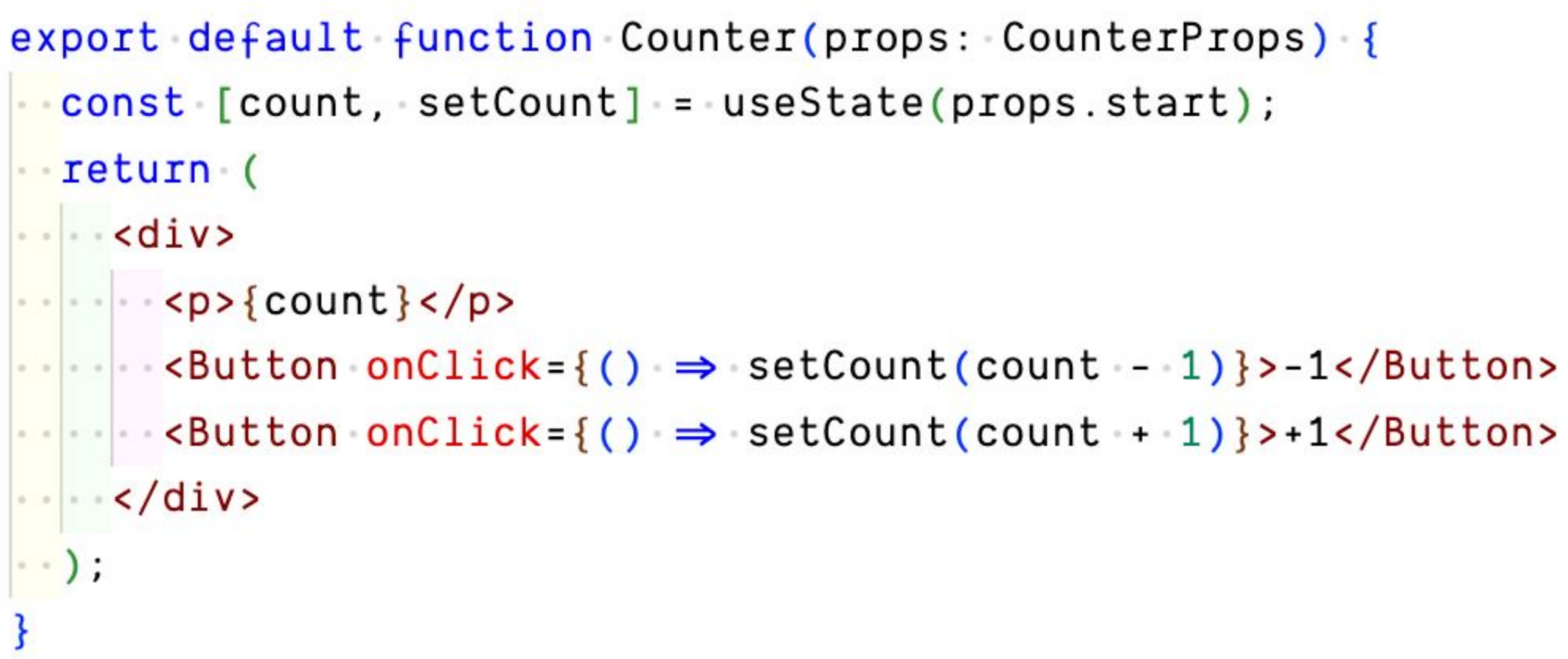
https://jasonformat.com/islands-architecture – Jason Miller

Islands

- Must reside in **/islands** folder



Islands

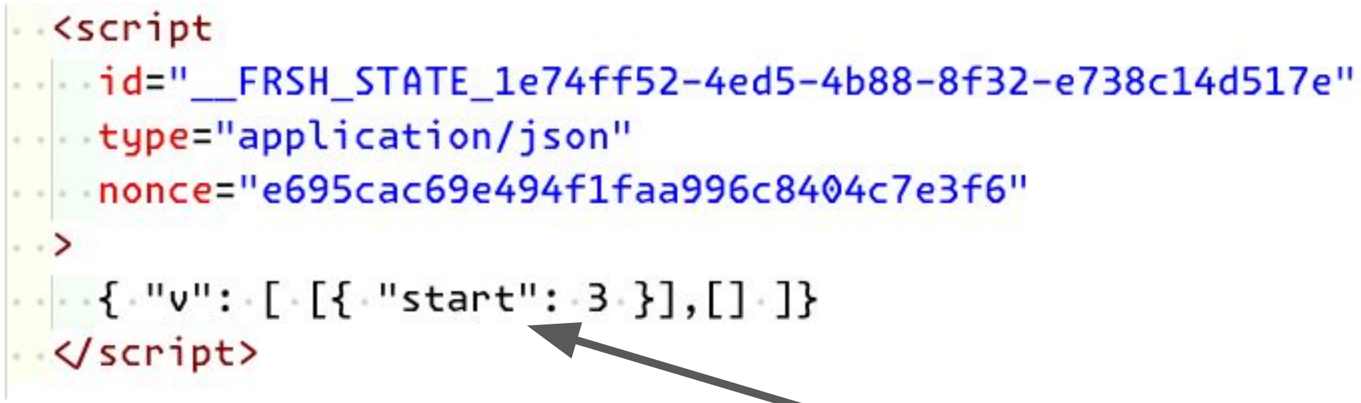


```
export default function Counter(props: CounterProps) {
  const [count, setCount] = useState(props.start);
  return (
    <div>
      <p>{count}</p>
      <Button onClick={() => setCount(count - 1)}>-1</Button>
      <Button onClick={() => setCount(count + 1)}>+1</Button>
    </div>
  );
}
```

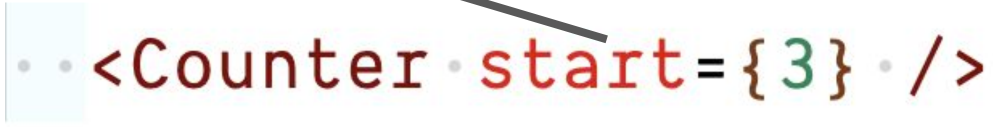
How do we revive an island?

How do we revive an island?

- A script of type application/json holds the initial state of any islands



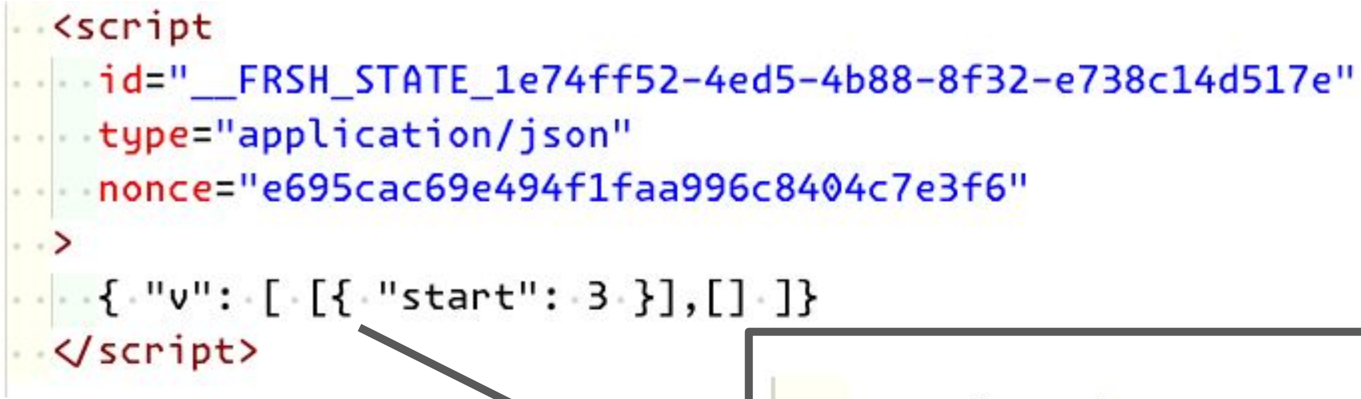
```
<script
  id="__FRSH_STATE_1e74ff52-4ed5-4b88-8f32-e738c14d517e"
  type="application/json"
  nonce="e695cac69e494f1faa996c8404c7e3f6"
>
  { "v": [ [{ "start": 3 }],[] ]}
</script>
```



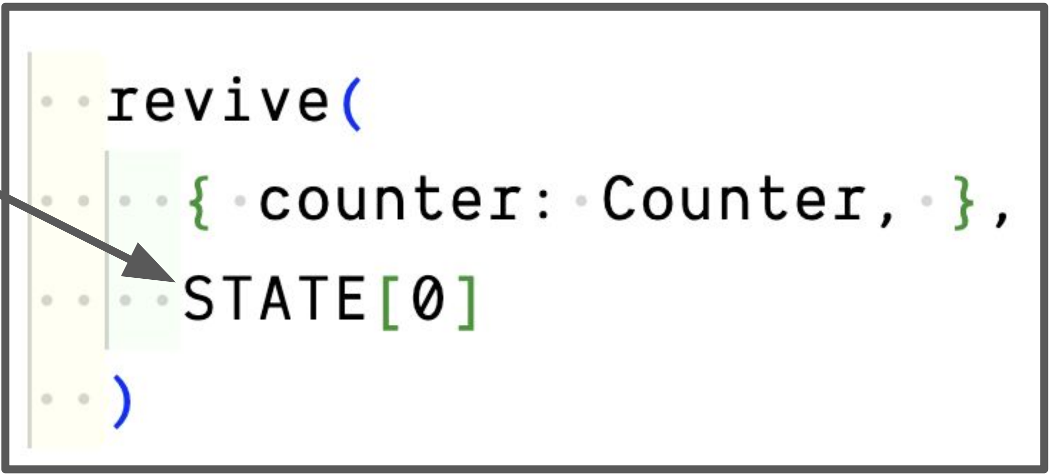
```
<Counter start={3} />
```

nickyt.co

How do we revive an island?



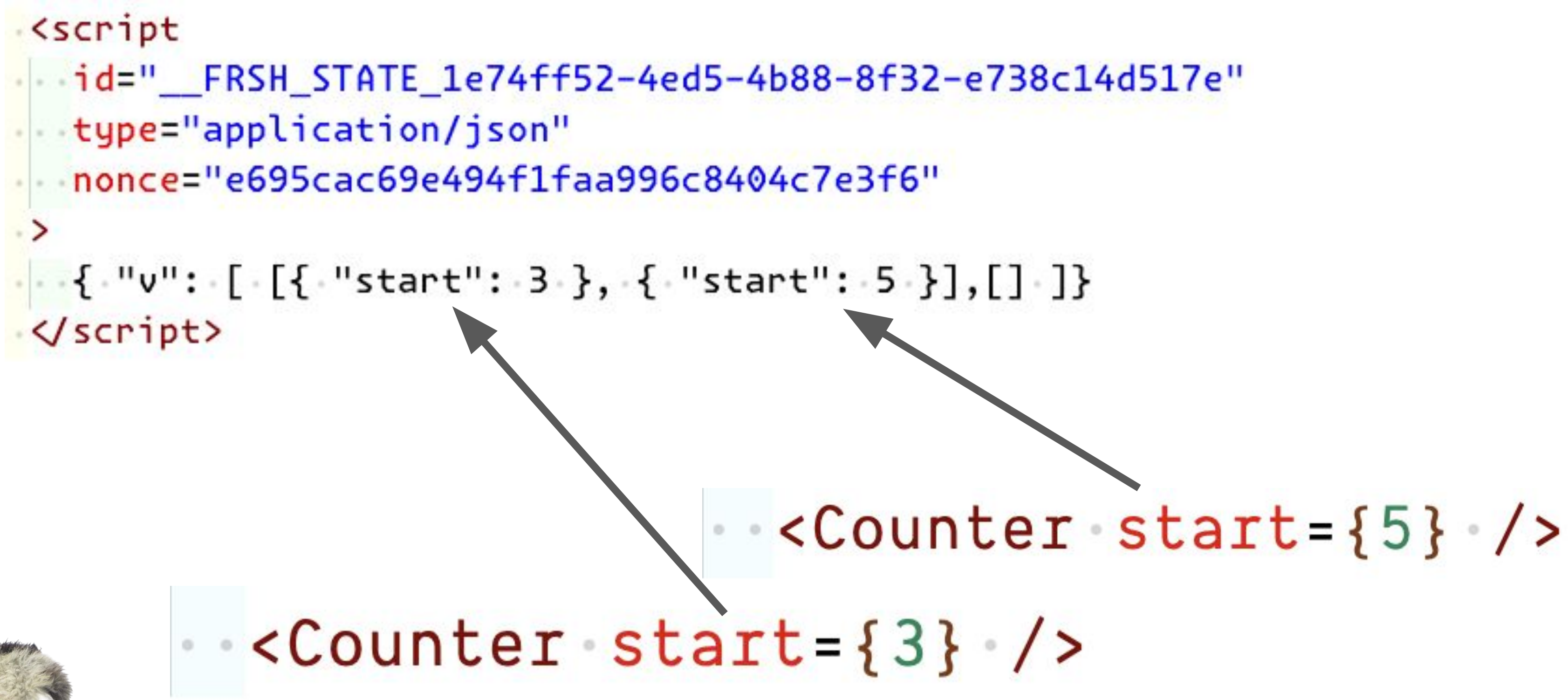
```
<script
  id="__FRSH_STATE_1e74ff52-4ed5-4b88-8f32-e738c14d517e"
  type="application/json"
  nonce="e695cac69e494f1faa996c8404c7e3f6"
>
  { "v": [ [{ "start": 3 }],[] ]}
</script>
```



```
revive(
  { counter: Counter, },
  STATE[0]
)
```

nickyt.co

How do we revive an island?

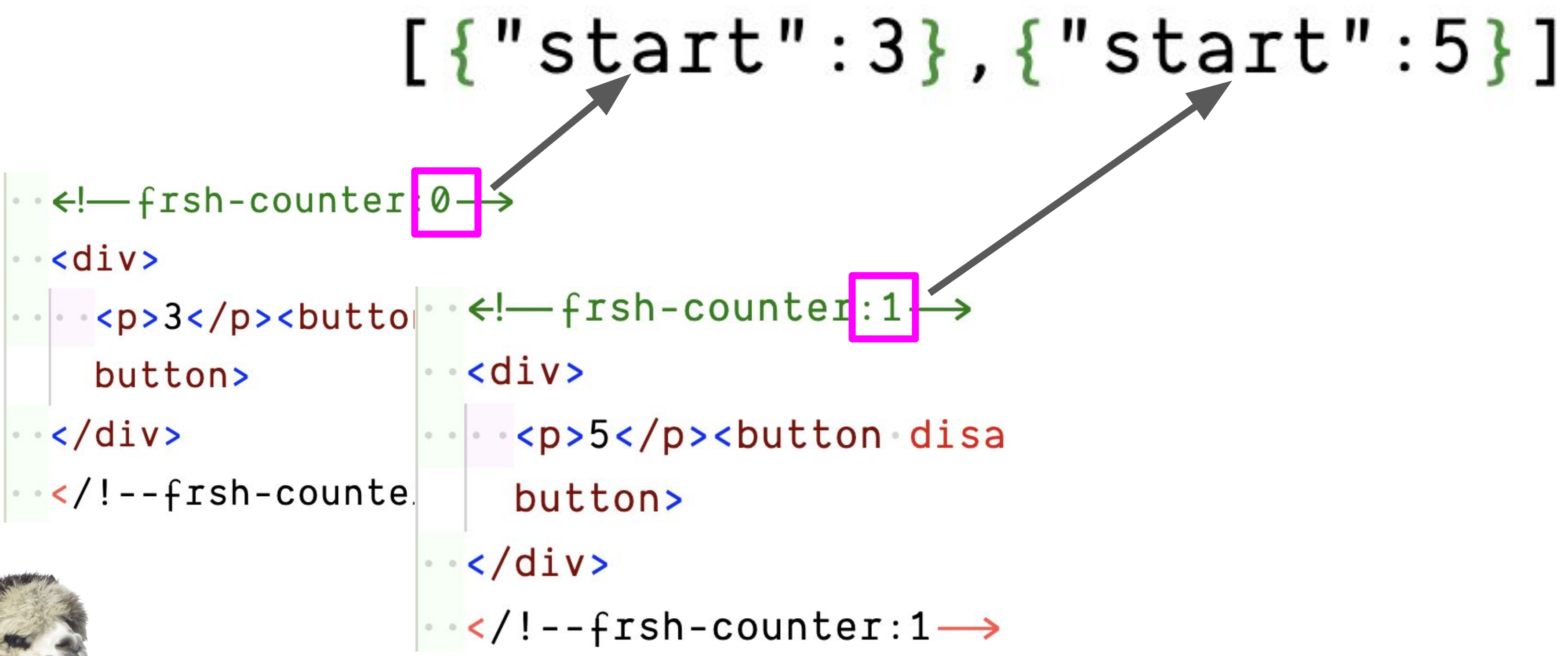


```
<script
  id="__FRSH_STATE_1e74ff52-4ed5-4b88-8f32-e738c14d517e"
  type="application/json"
  nonce="e695cac69e494f1faa996c8404c7e3f6"
>
  { "v": [ [{ "start": 3 }, { "start": 5 }],[] ]}
</script>
```

```
  <Counter start={5} />
```

```
  <Counter start={3} />
```

How do we revive an island?



```
[{"start":3},{"start":5}]
```

```
<!—frsh-counter:0—>
<div>
  <p>3</p><butto
  button>
</div>
</!--frsh-counte
```

```
<!—frsh-counter:1—>
<div>
  <p>5</p><button disa
  button>
</div>
</!--frsh-counter:1—>
```

nickyt.co

Signals

- Since Fresh uses Preact, signals are available on the client-side.
- Alternative to useState, useReducer…

nickyt.co

Signals

"At its core, a signal is an object with a **.value** property that holds a value. This has an important characteristic: a signal's value can change, but the signal itself always stays the same"

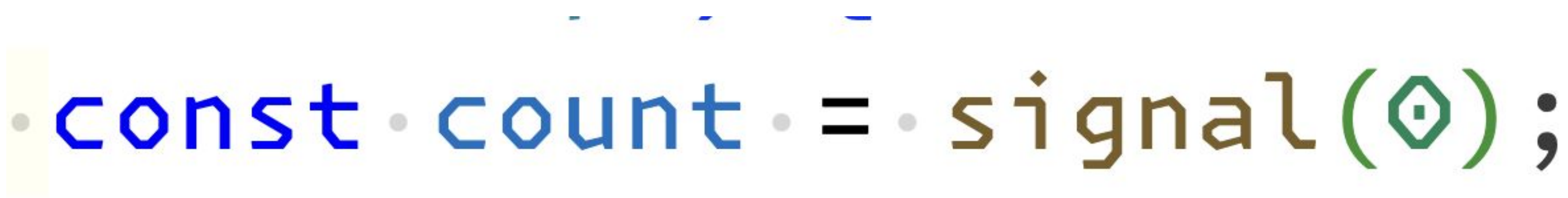
– Preact Signals documentation

Signals

```
import { signal } from "@preact/signals";
```

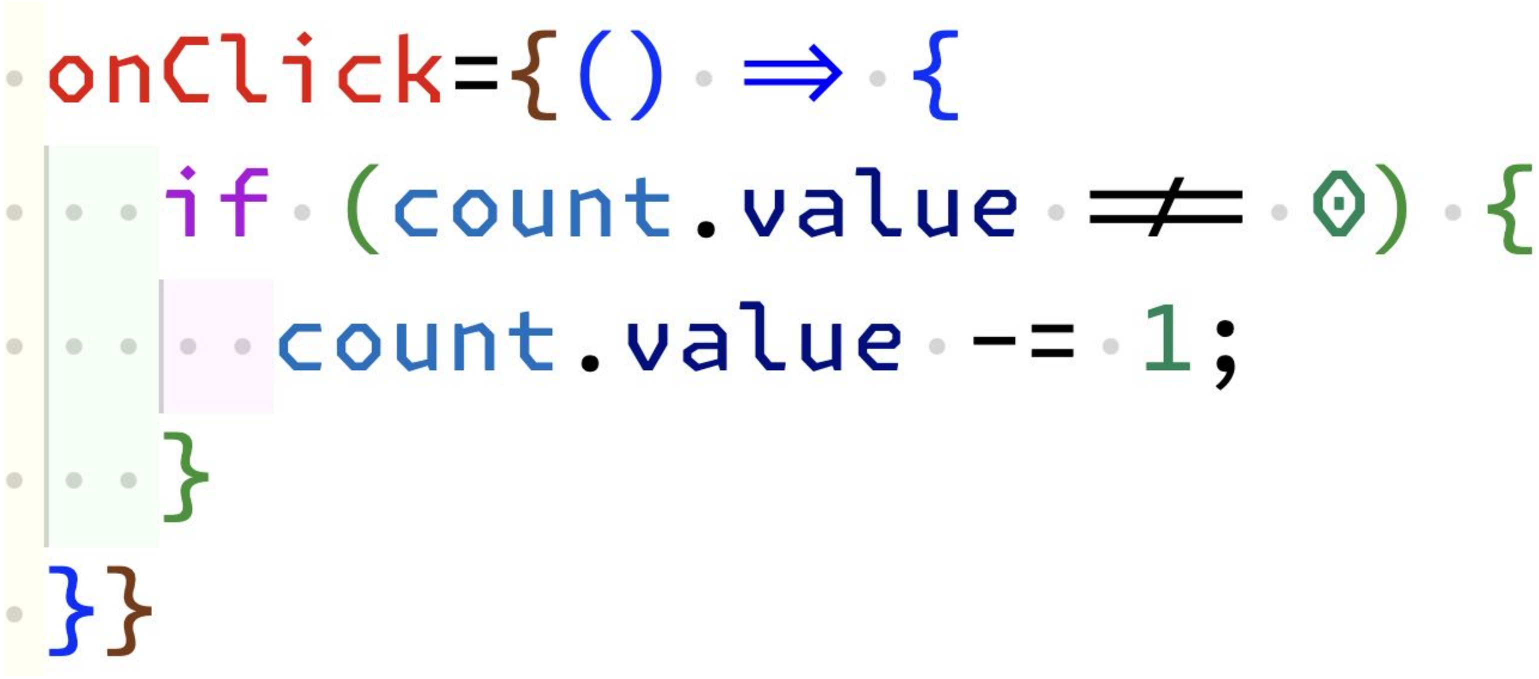
nickyt.co

Signals



```
const count = signal(0);
```

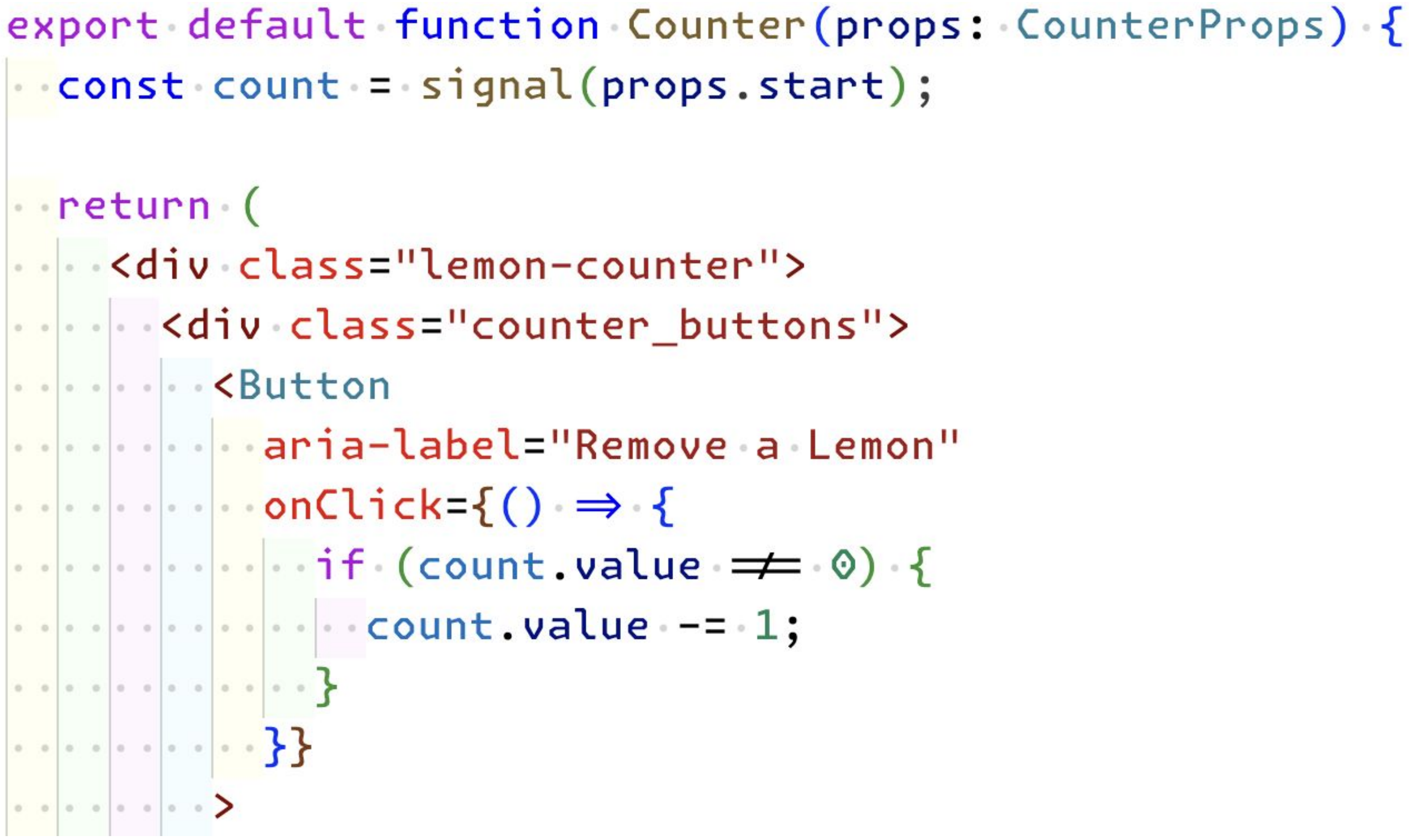
Signals



```
 onClick={() => {
   if (count.value !== 0) {
     count.value -= 1;
   }
 }}
```

nickyt.co

Signals



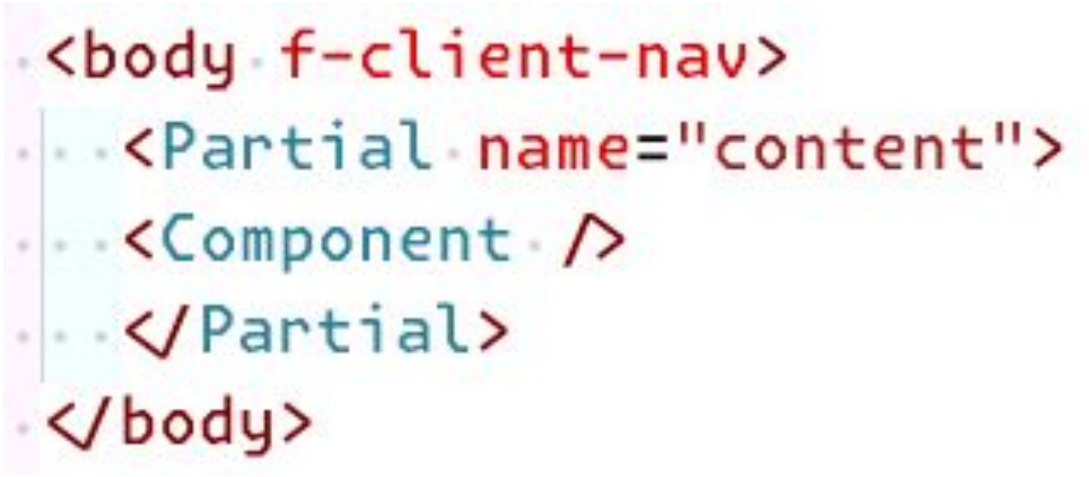
```
export default function Counter(props: CounterProps) {
  const count = signal(props.start);

  return (
    <div class="lemon-counter">
      <div class="counter_buttons">
        <Button
          aria-label="Remove a Lemon"
          onClick={() => {
            if (count.value !== 0) {
              count.value -= 1;
            }
          }}
        >
```

nickyt.co

Partials

- Enables client-side routing to give a snappier feel to your app
- Requires an HTML attribute ***f-client-nav*** on a container element, e.g. ***<body />***
- Wrap main content with a ***<Partial />*** component

Partials



```
<body f-client-nav>
  <Partial name="content">
  <Component />
  </Partial>
</body>
```

Partials

- Can do more granular partials if you want to
- See the documentation on [Partials]

Demo time

demo deployed at

nickyt.co/demos/fresh

demo source code

github.com/nickytonline/fresh-demo

Resources

Resources

- Fresh Website
- Fresh: a new full stack web framework for Deno with Luca Casonato
- Fresh 1.5: Partials, client side navigation and more
- Islands Architecture
- Installing Deno
- Preact
- Preact Signals
- WTF is JSX?
- What is Deno?

Resources

- [TypeScript Language]
- Import maps
- Fetch API
- Request
- Response
- Twind
- Deno: node: specifiers
- Deno: npm via CDNs
- Deno: npm: specifiers
- WinterCG
- revive() function

nickyt.co

Slide deck available at
nickyt.co/slides/confoo-fresh

That's all folks!

Thank you!

Stay in touch!

- @nickytonline everywhere
- nickyt.co
- nickyt.live
- nickyt.tube