SSR Cookies

Setting & Getting Cookies in SvelteKit

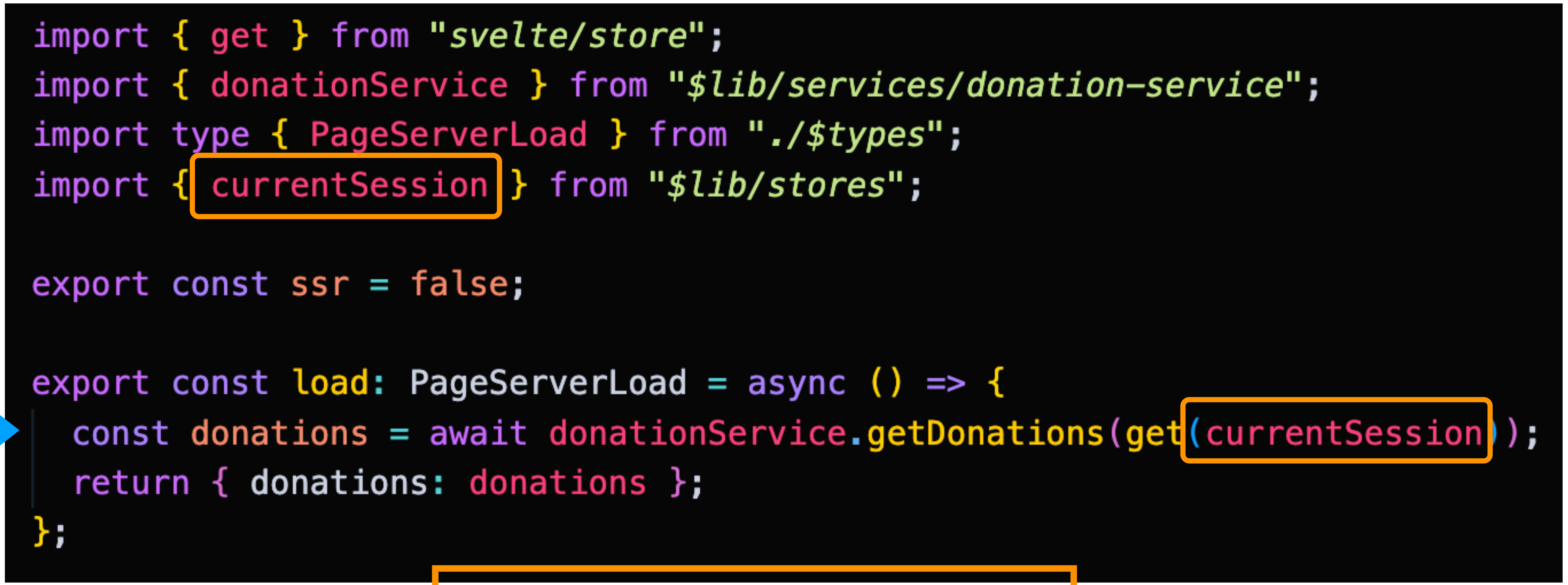
Sessions

- Svelte Stores are not automatically synchronised between client and server

- This will cause our current approach - keeping the session information in a store - will no longer work

- currentSession will not have the session information

- getDonations will fail

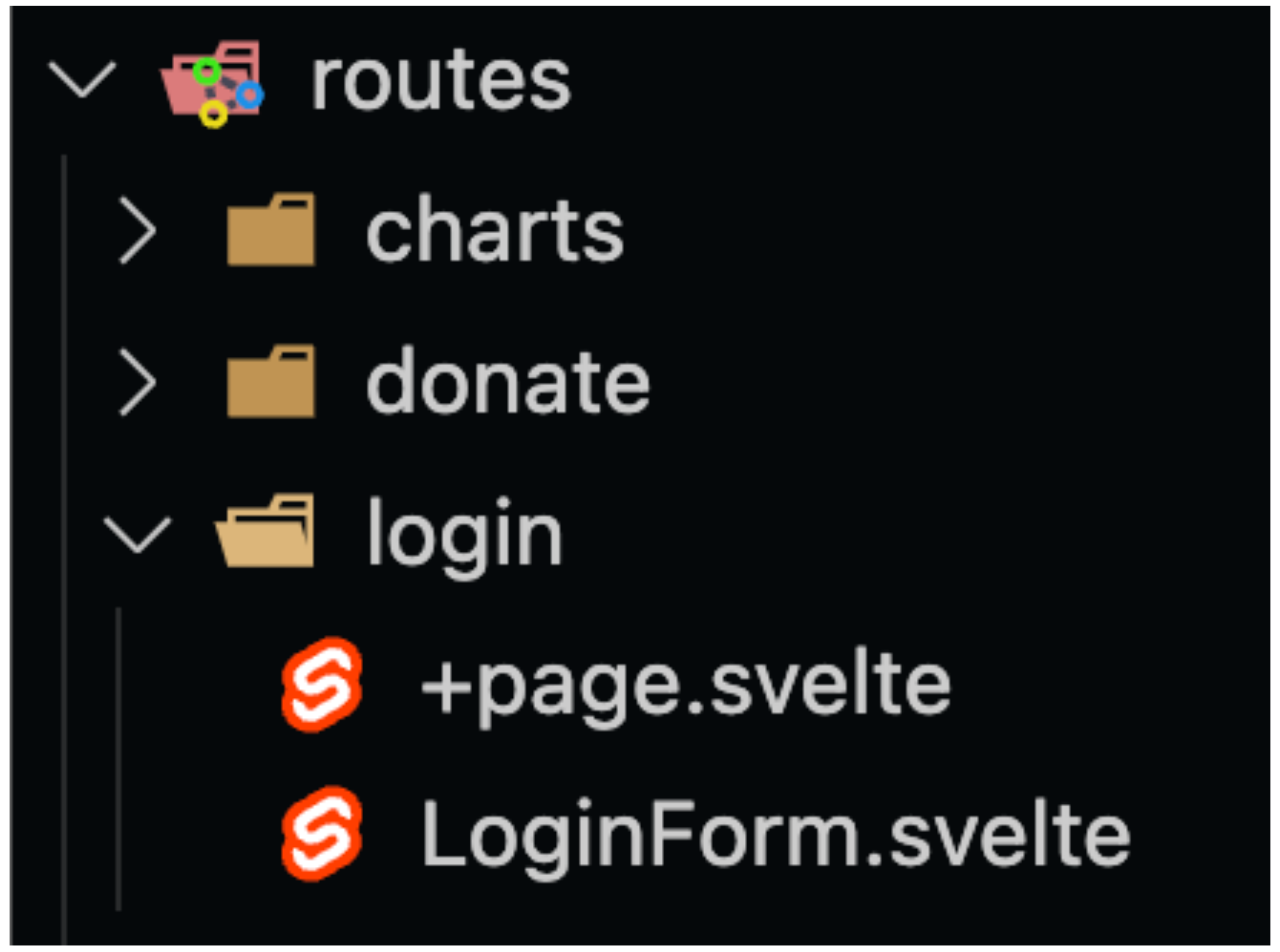


```
import { get } from "svelte/store";
import { donationService } from "$lib/services/donation-service";
import type { PageServerLoad } from "./$types";
import { currentSession } from "$lib/stores";

export const ssr = false;

export const load: PageServerLoad = async () => {
  const donations = await donationService.getDonations(get(currentSession));
  return { donations: donations };
};
```

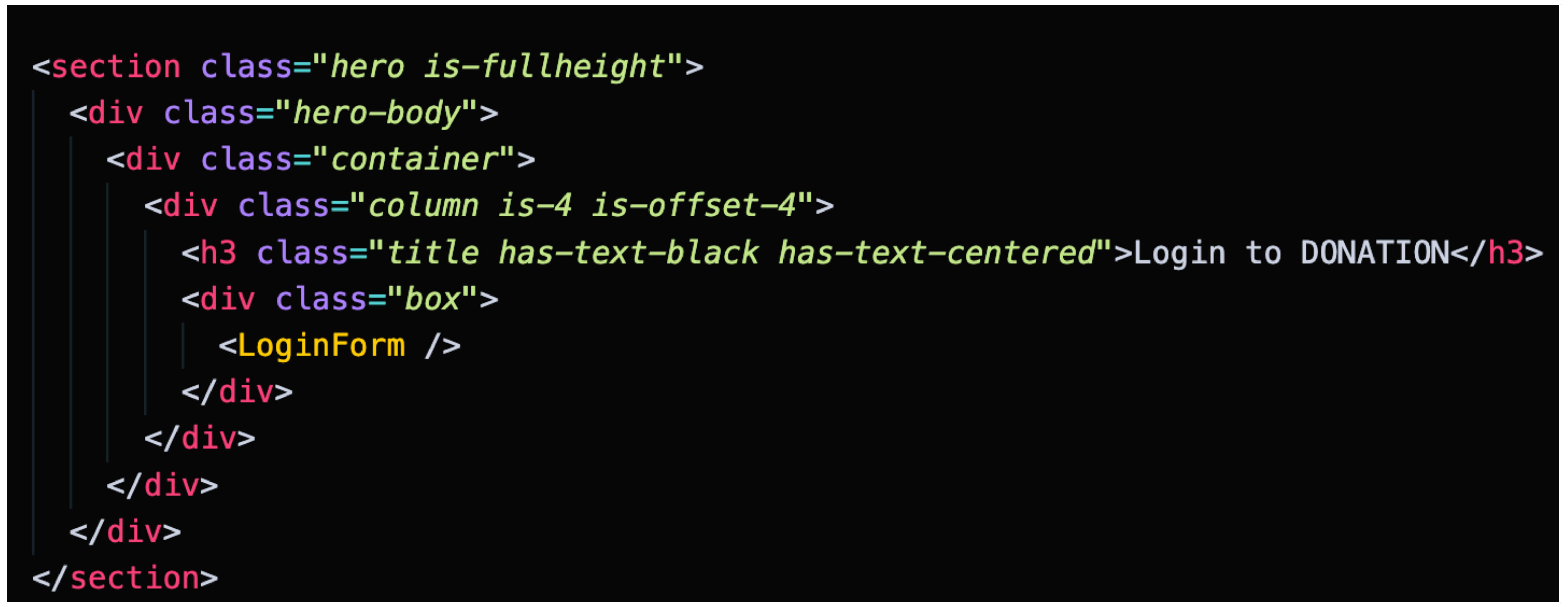
+page.server.ts

Current Login Route

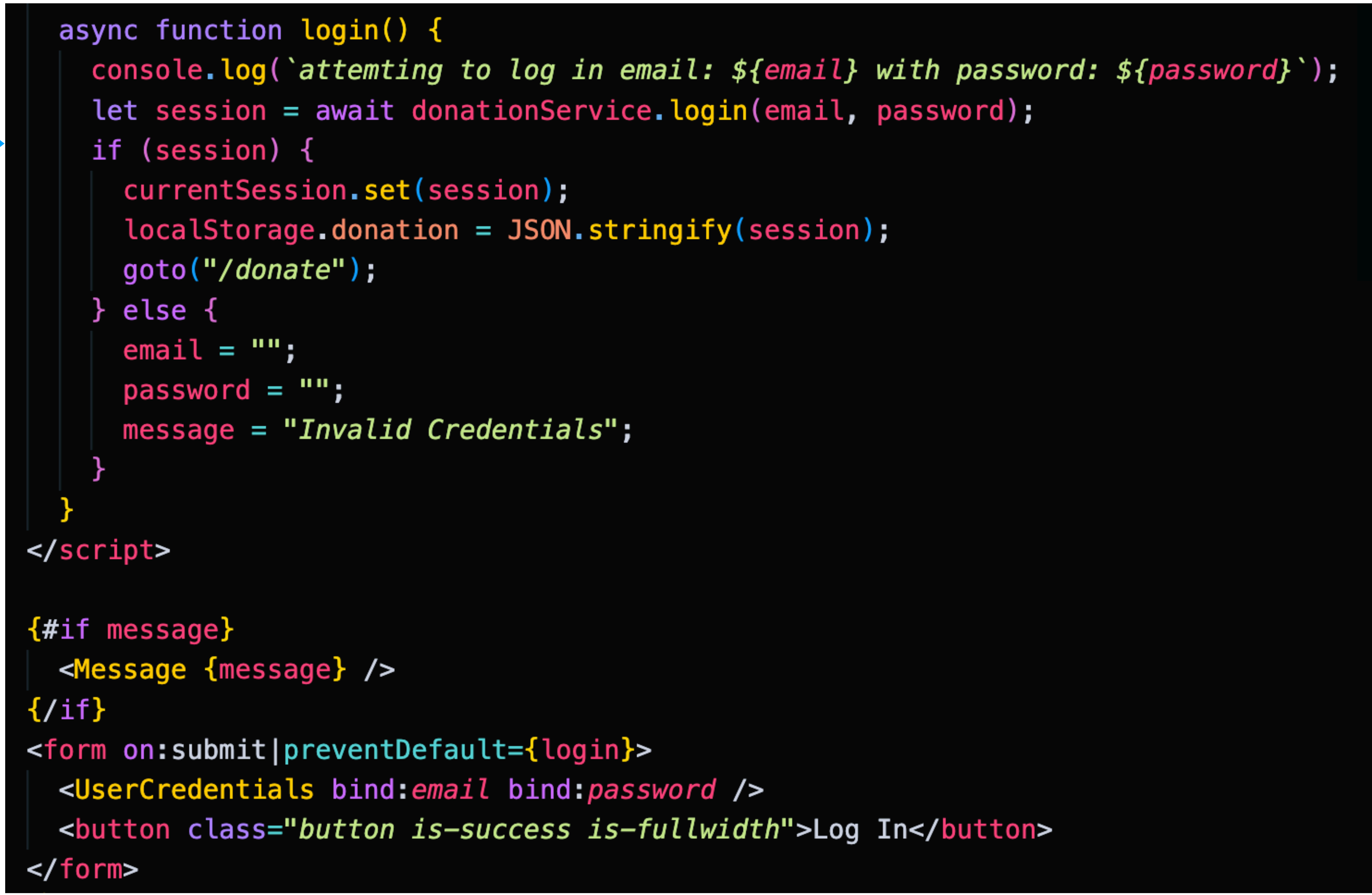
- Form handling handling in the client (browser)
- login() function make call to server



+page.svelte



```
<section class="hero is-fullheight">
  <div class="hero-body">
    <div class="container">
      <div class="column is-4 is-offset-4">
        <h3 class="title has-text-black has-text-centered">Login to DONATION</h3>
        <div class="box">
          <LoginForm />
        </div>
      </div>
    </div>
  </div>
</section>
```



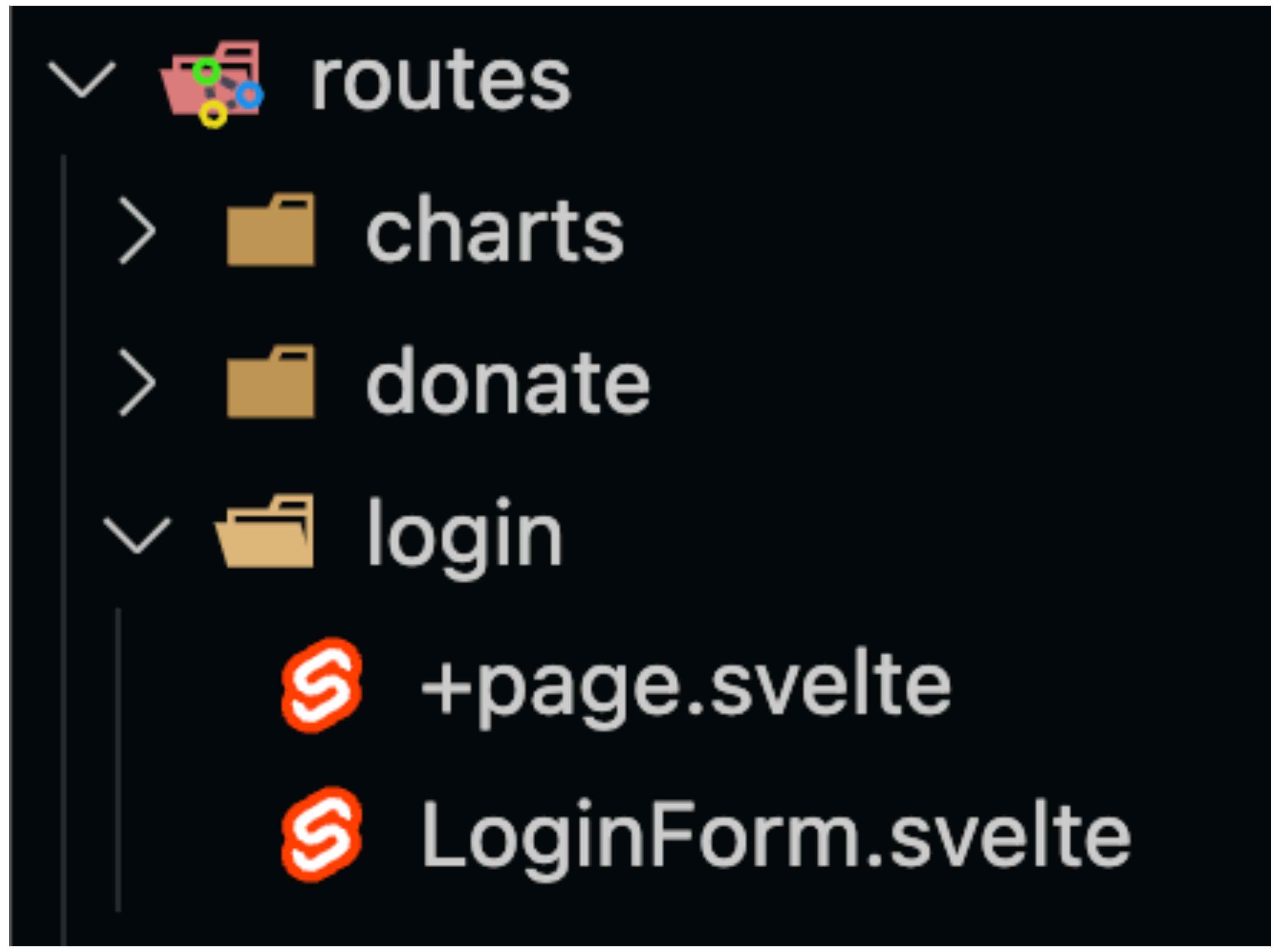
```
  async function login() {
    console.log(`attemting to log in email: ${email} with password: ${password}`);
    let session = await donationService.login(email, password);
    if (session) {
      currentSession.set(session);
      localStorage.donation = JSON.stringify(session);
      goto("/donate");
    } else {
      email = "";
      password = "";
      message = "Invalid Credentials";
    }
  }
</script>

{#if message}
  <Message {message} />
{/if}
<form on:submit|preventDefault={login}>
  <UserCredentials bind:email bind:password />
  <button class="button is-success is-fullwidth">Log In</button>
</form>
```

LoginForm.svelte

SSR - Form Handling - Authentication

- With SSR engaged, we can incorporate form handling in a similar manner to conventional node applications (Hapi)
- i.e. the Form data can be ‘Posted’ to the server, and processed there.
- Specifically, this processing can include conventional cookie based session creation + tracking, replacing the store/local storage approach

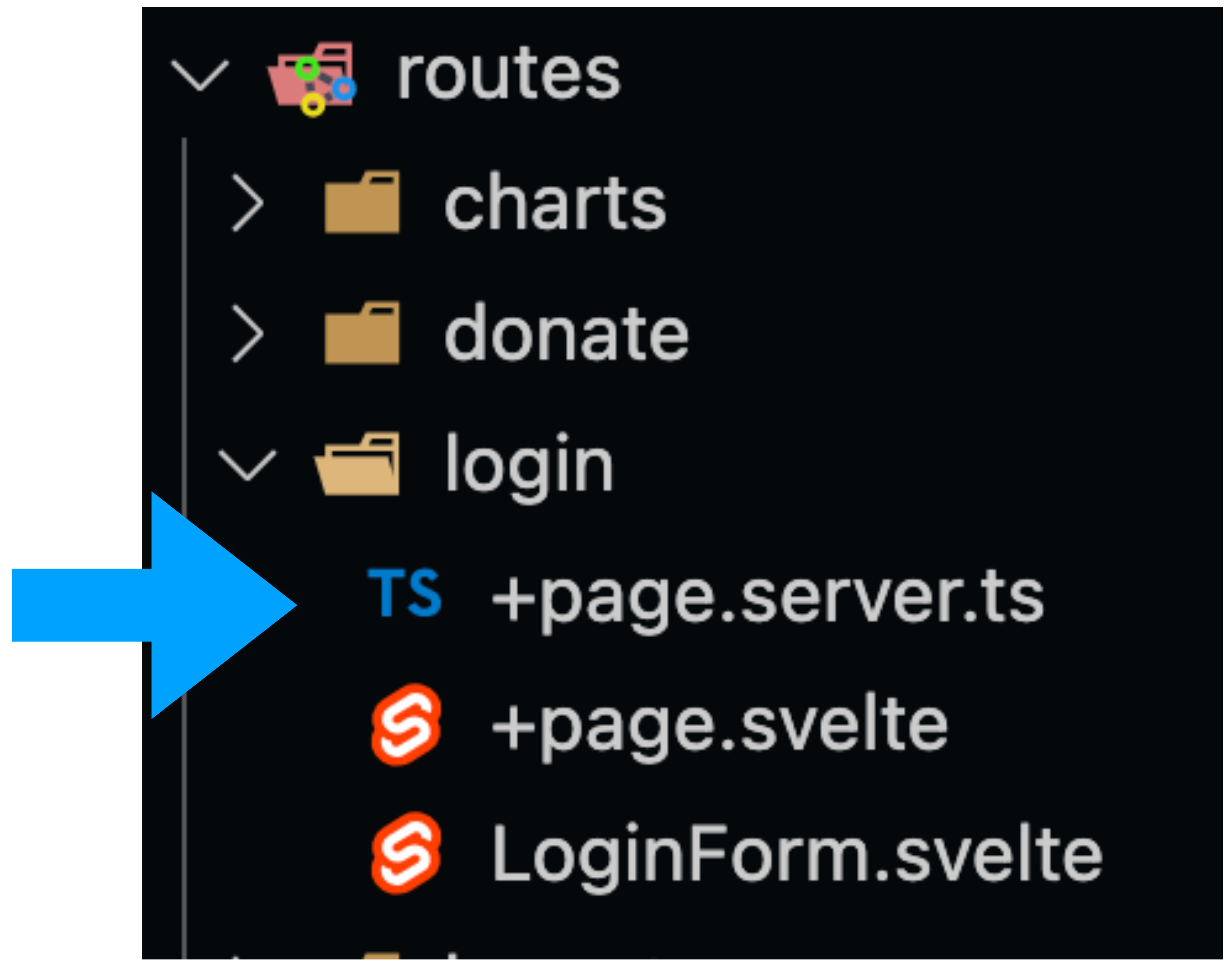
Server Page Component



```
routes
  charts
  donate
  login
    +page.svelte
    LoginForm.svelte
```

- Place +page.server.ts module in the login route

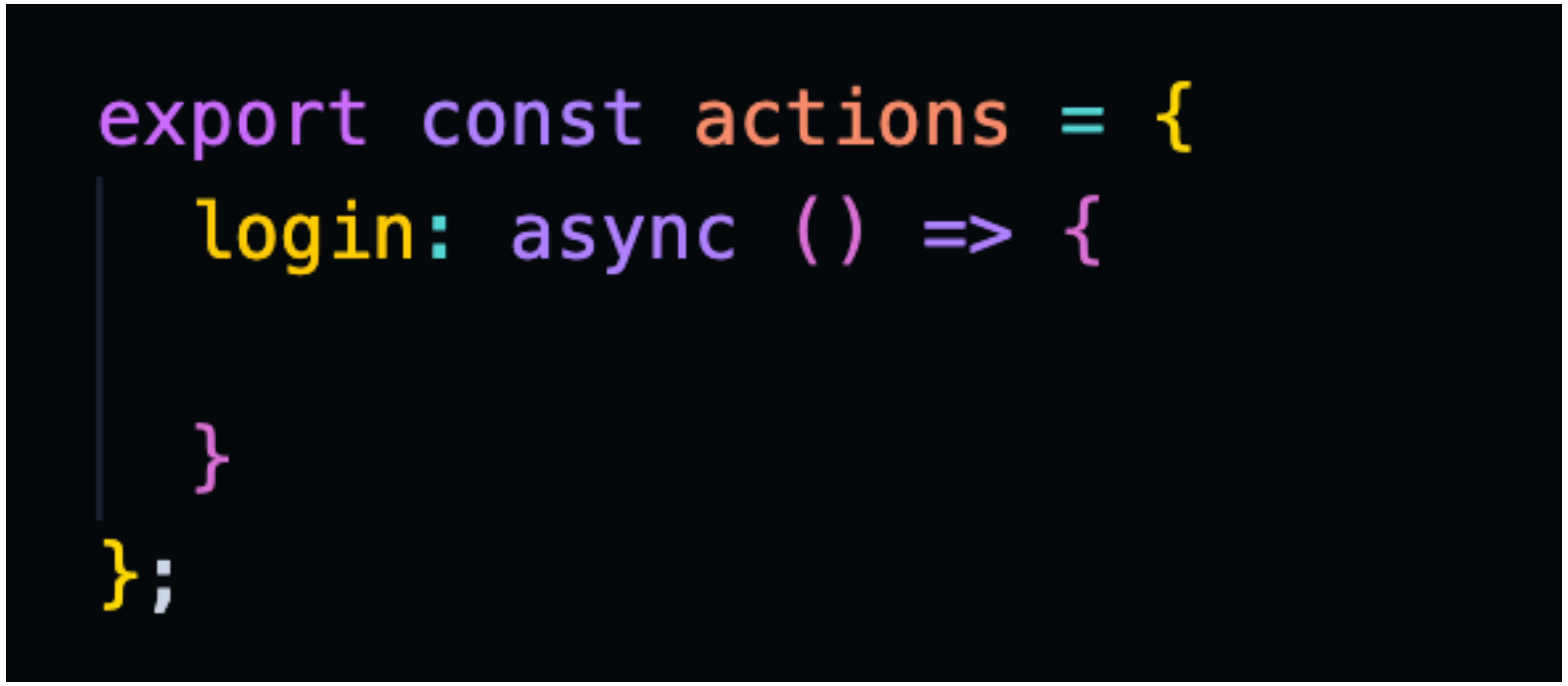
- This will always run on the server



```
routes
  charts
  donate
  login
    +page.server.ts
    +page.svelte
    LoginForm.svelte
```

+page.server.ts

- Can export named function called **actions**
- Equivalent to Node/Hapi controller methods
- I.e. run on the server
- May be associated with route associated with a form (POST route)



```
export const actions = {
  login: async () => {

  }
};
```

+page.svelte

- Instead of handling the login event locally (in the browser)…

- … POST the request to the server

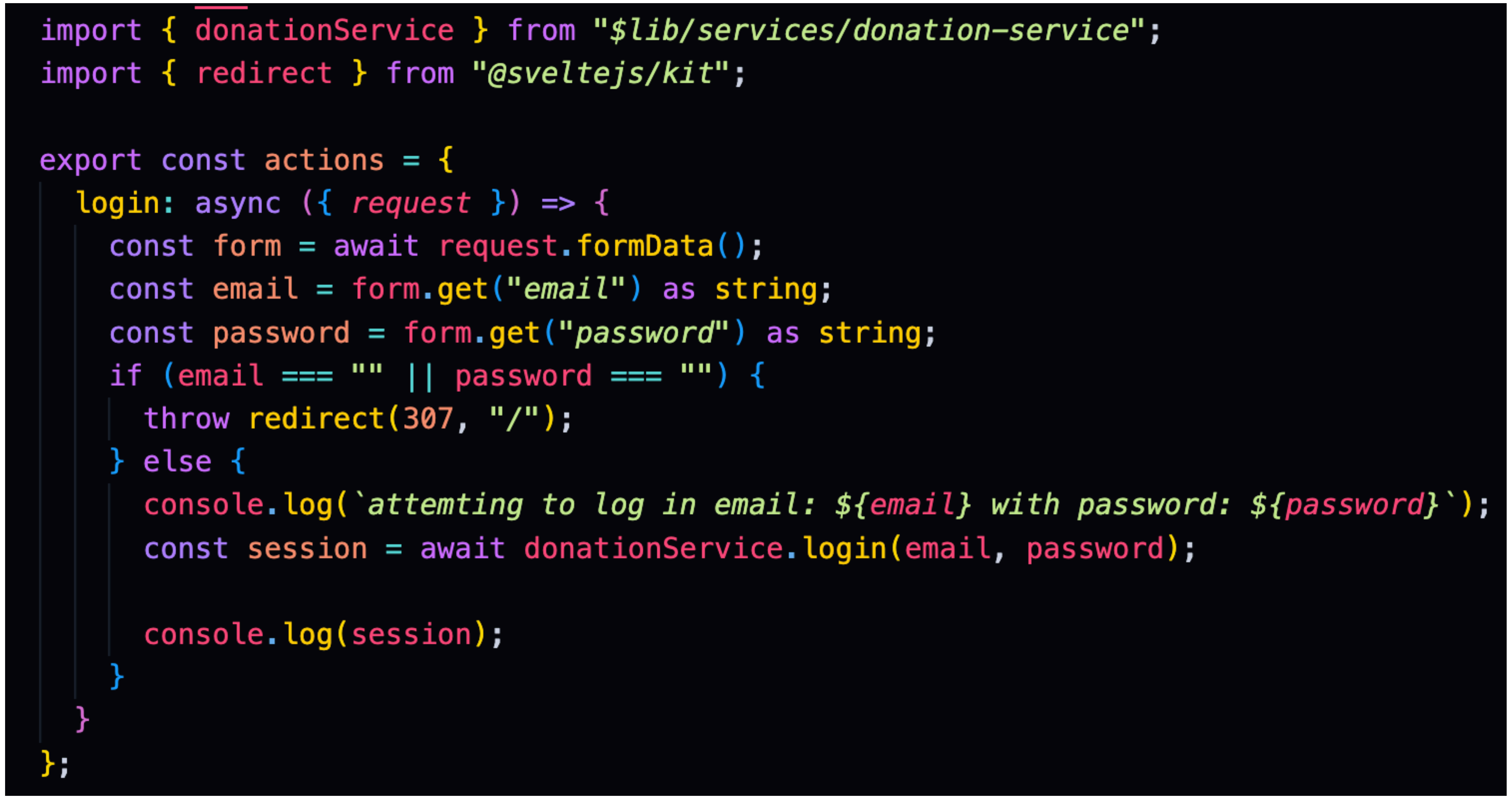
```
<form method="POST" action="?/login">
  <UserCredentials />
  <button class="button is-success is-fullwidth">Log In</button>
</form>
```

- This will to routed to +page.server.ts

- Executing always on the server

```
export const actions = {
  login: async () => {

  }
};
```

login server: action

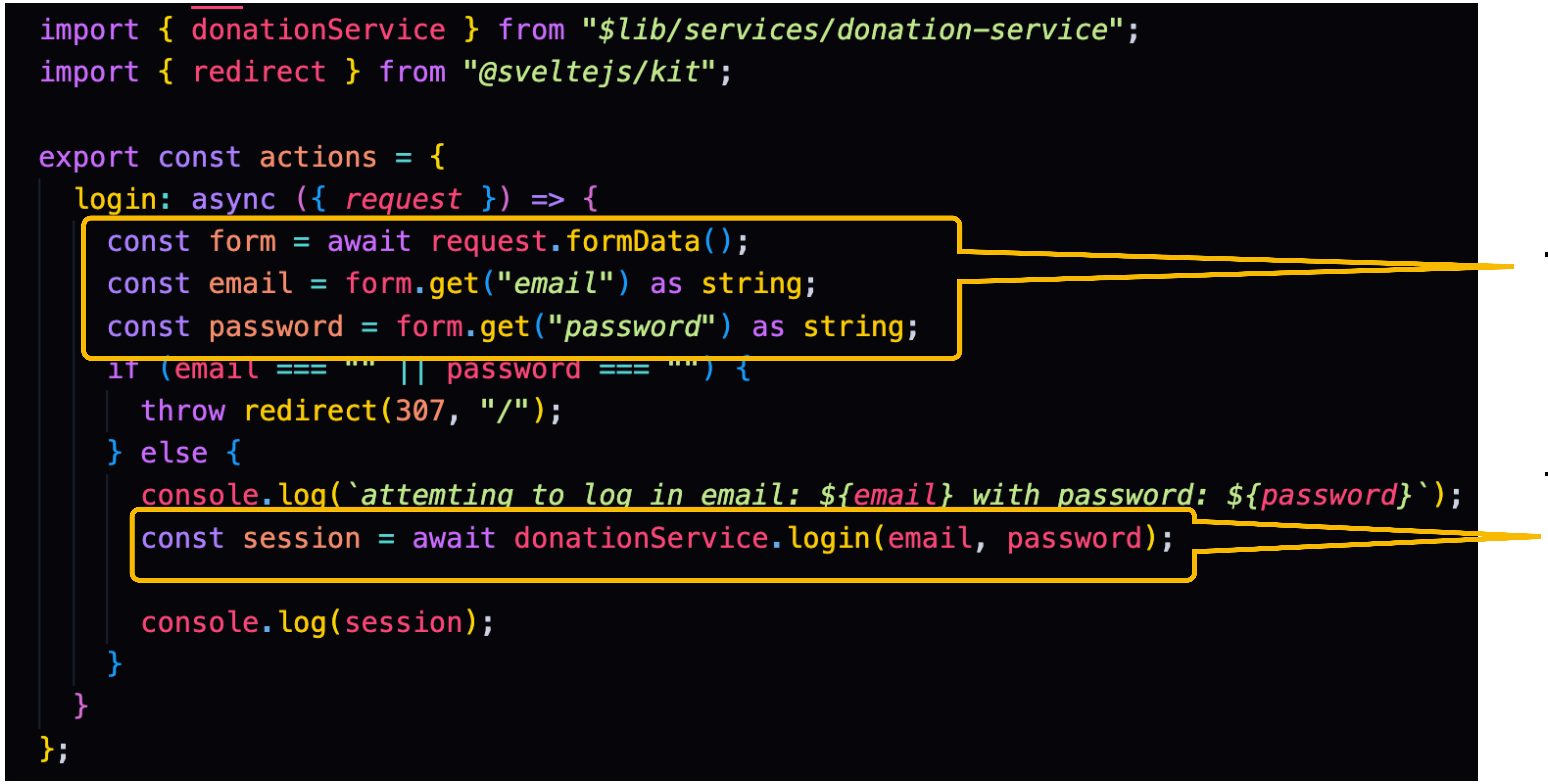


```
import { donationService } from "$lib/services/donation-service";
import { redirect } from "@sveltejs/kit";

export const actions = {
  login: async ({ request }) => {
    const form = await request.formData();
    const email = form.get("email") as string;
    const password = form.get("password") as string;
    if (email === "" || password === "") {
      throw redirect(307, "/");
    } else {
      console.log(`attemting to log in email: ${email} with password: ${password}`);
      const session = await donationService.login(email, password);

      console.log(session);
    }
  }
};
```

- Process the login form on the server

- Contact the database, verify user is valid + password matches.

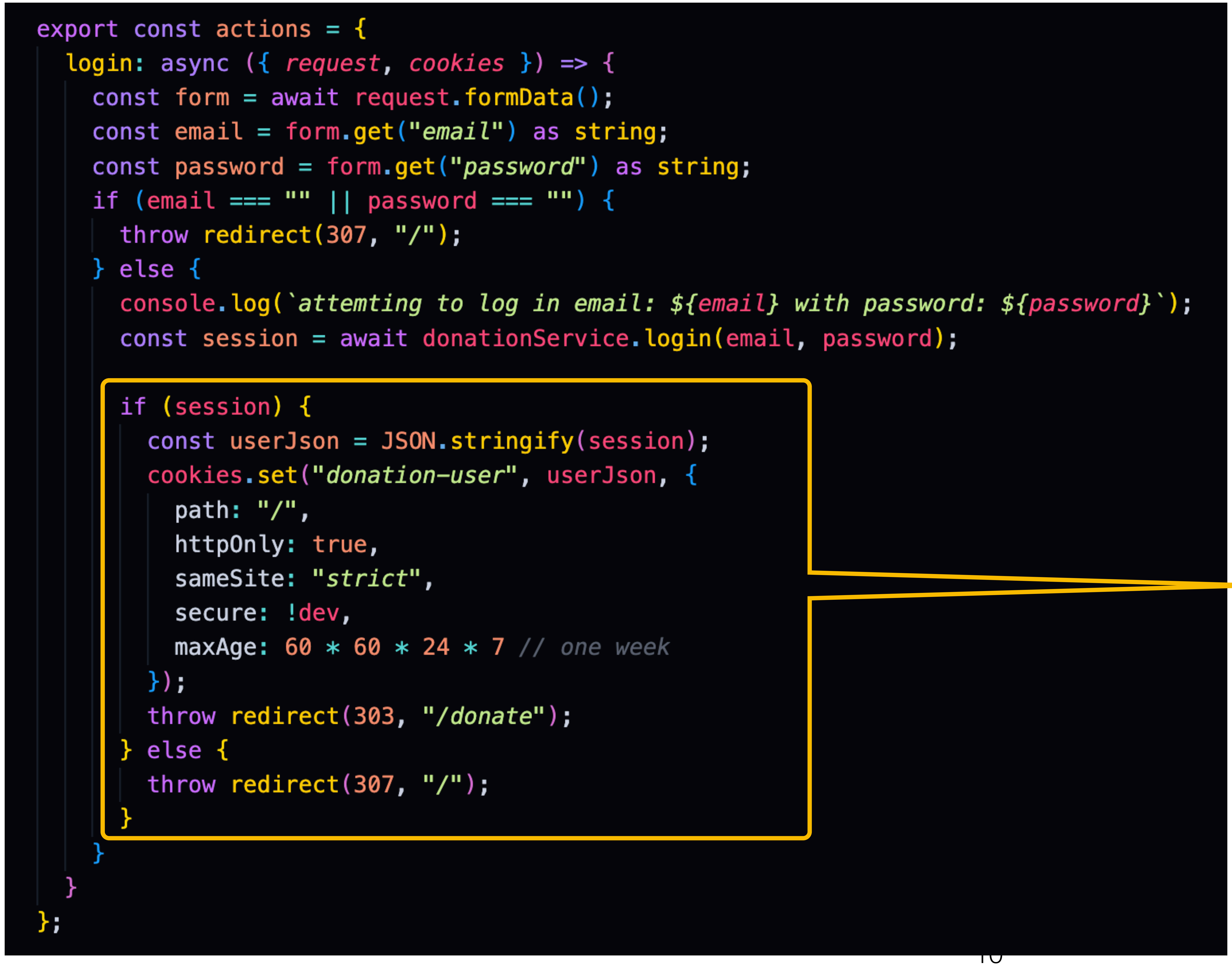
login server: form data



```
import { donationService } from "$lib/services/donation-service";
import { redirect } from "@sveltejs/kit";

export const actions = {
  login: async ({ request }) => {
    const form = await request.formData();
    const email = form.get("email") as string;
    const password = form.get("password") as string;
    if (email === "" || password === "") {
      throw redirect(307, "/");
    } else {
      console.log(`attemting to log in email: ${email} with password: ${password}`);
      const session = await donationService.login(email, password);

      console.log(session);
    }
  }
};
```

- Recover the form input fields
- Invoke the login api, generating a session

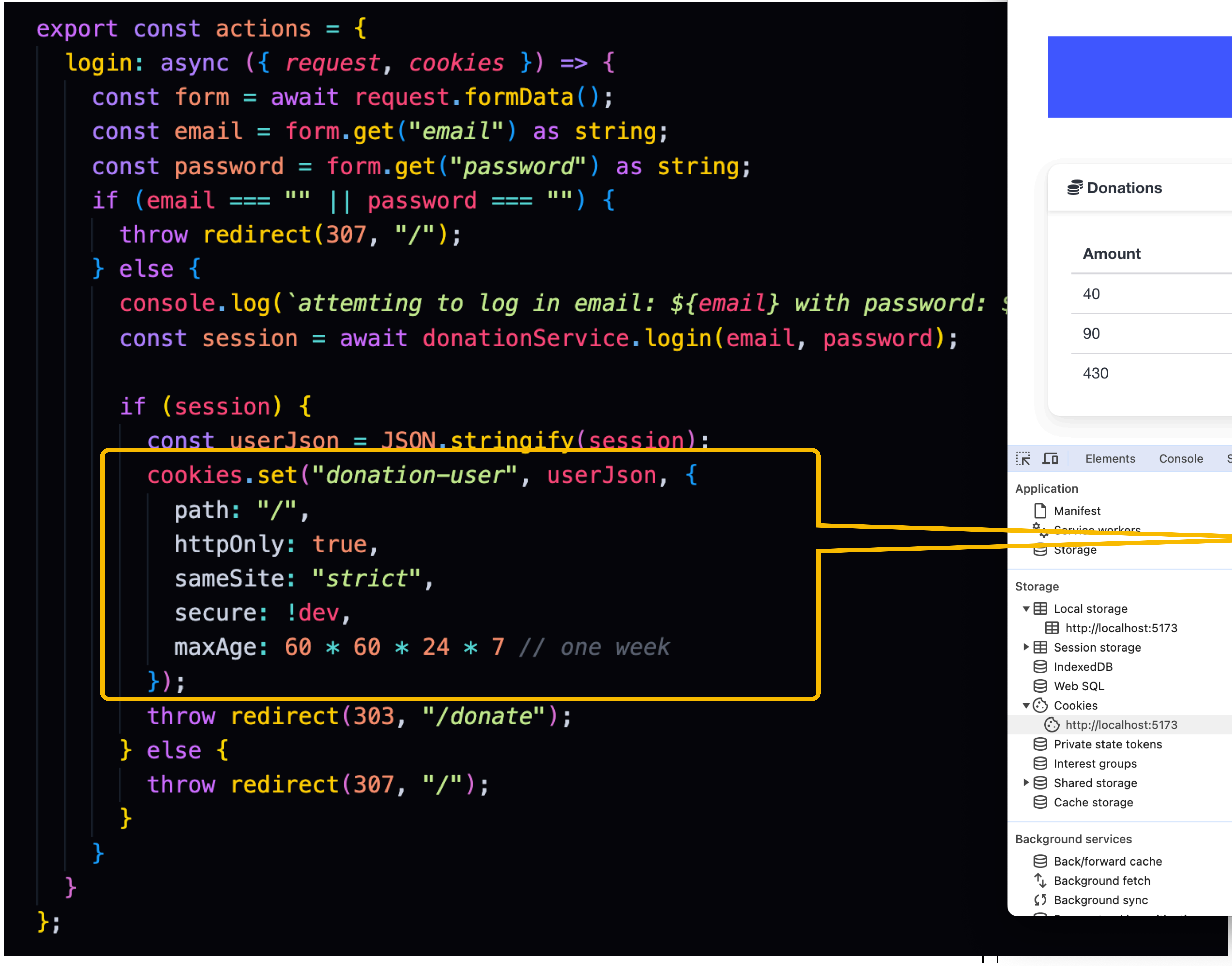
login server: cookies



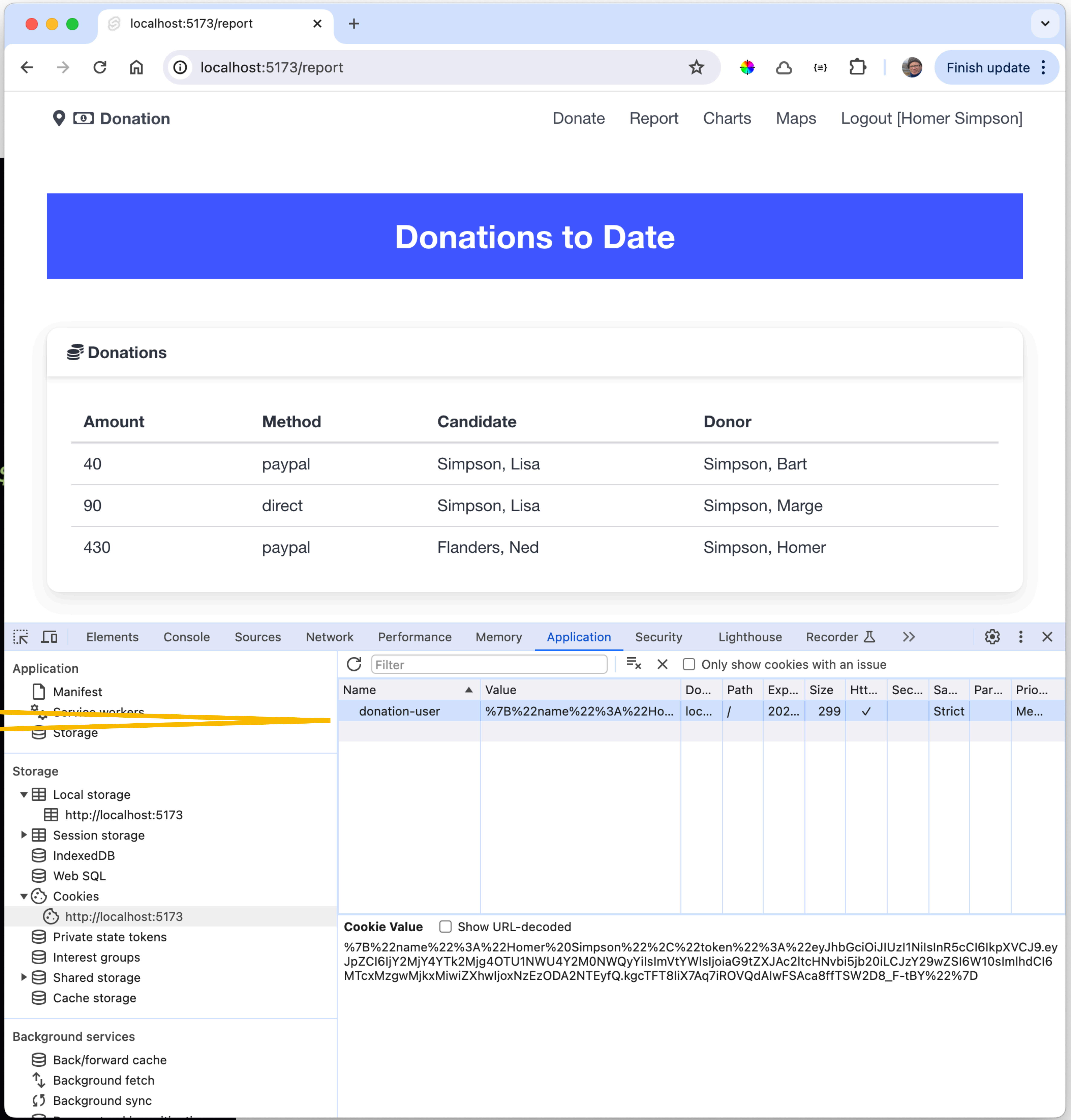
```
export const actions = {
  login: async ({ request, cookies }) => {
    const form = await request.formData();
    const email = form.get("email") as string;
    const password = form.get("password") as string;
    if (email === "" || password === "") {
      throw redirect(307, "/");
    } else {
      console.log(`attemting to log in email: ${email} with password: ${password}`);
      const session = await donationService.login(email, password);

      if (session) {
        const userJson = JSON.stringify(session);
        cookies.set("donation-user", userJson, {
          path: "/",
          httpOnly: true,
          sameSite: "strict",
          secure: !dev,
          maxAge: 60 * 60 * 24 * 7 // one week
        });
        throw redirect(303, "/donate");
      } else {
        throw redirect(307, "/");
      }
    }
  }
};
```

- If a session was created (i.e. name/password valid)
  - Create a JSON version of the session
  - Create a cookie “donation-user” with this value
  - Reroute to “/donate”

login server: cookies



```
export const actions = {
  login: async ({ request, cookies }) => {
    const form = await request.formData();
    const email = form.get("email") as string;
    const password = form.get("password") as string;
    if (email === "" || password === "") {
      throw redirect(307, "/");
    } else {
      console.log(`attemting to log in email: ${email} with password:
      const session = await donationService.login(email, password);

      if (session) {
        const userJson = JSON.stringify(session);
        cookies.set("donation-user", userJson, {
          path: "/",
          httpOnly: true,
          sameSite: "strict",
          secure: !dev,
          maxAge: 60 * 60 * 24 * 7 // one week
        });
        throw redirect(303, "/donate");
      } else {
        throw redirect(307, "/");
      }
    }
  }
};
```



localhost:5173/report

Donation — Donate | Report | Charts | Maps | Logout [Homer Simpson]

Donations to Date

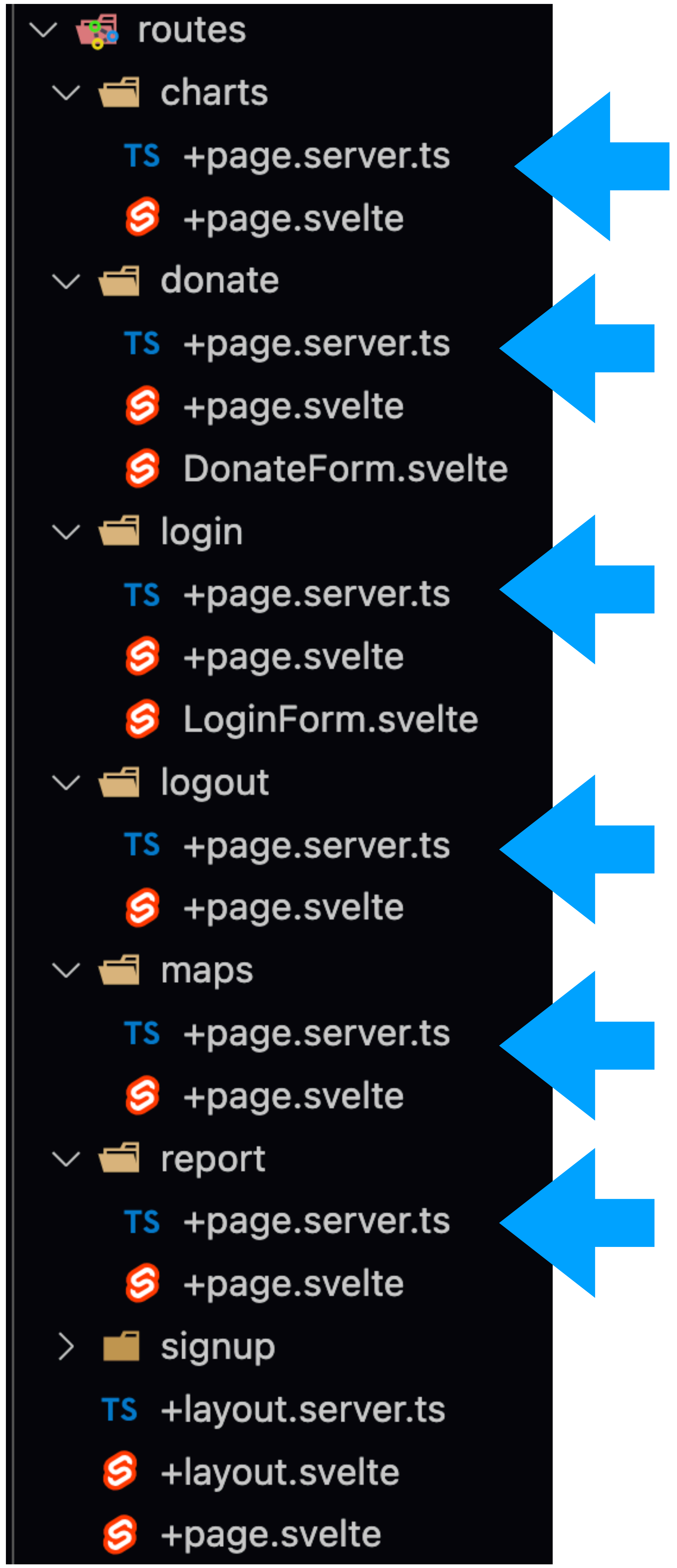
Donations

Amount	Method	Candidate	Donor
40	paypal	Simpson, Lisa	Simpson, Bart
90	direct	Simpson, Lisa	Simpson, Marge
430	paypal	Flanders, Ned	Simpson, Homer

Name	Value	Do...	Path	Exp...	Size	Htt...	Sec...	Sa...	Par...	Prio...
donation-user	%7B%22name%22%3A%22Ho...	loc...	/	202...	299	✓		Strict		Me...

Cookie Value ☐ Show URL-decoded

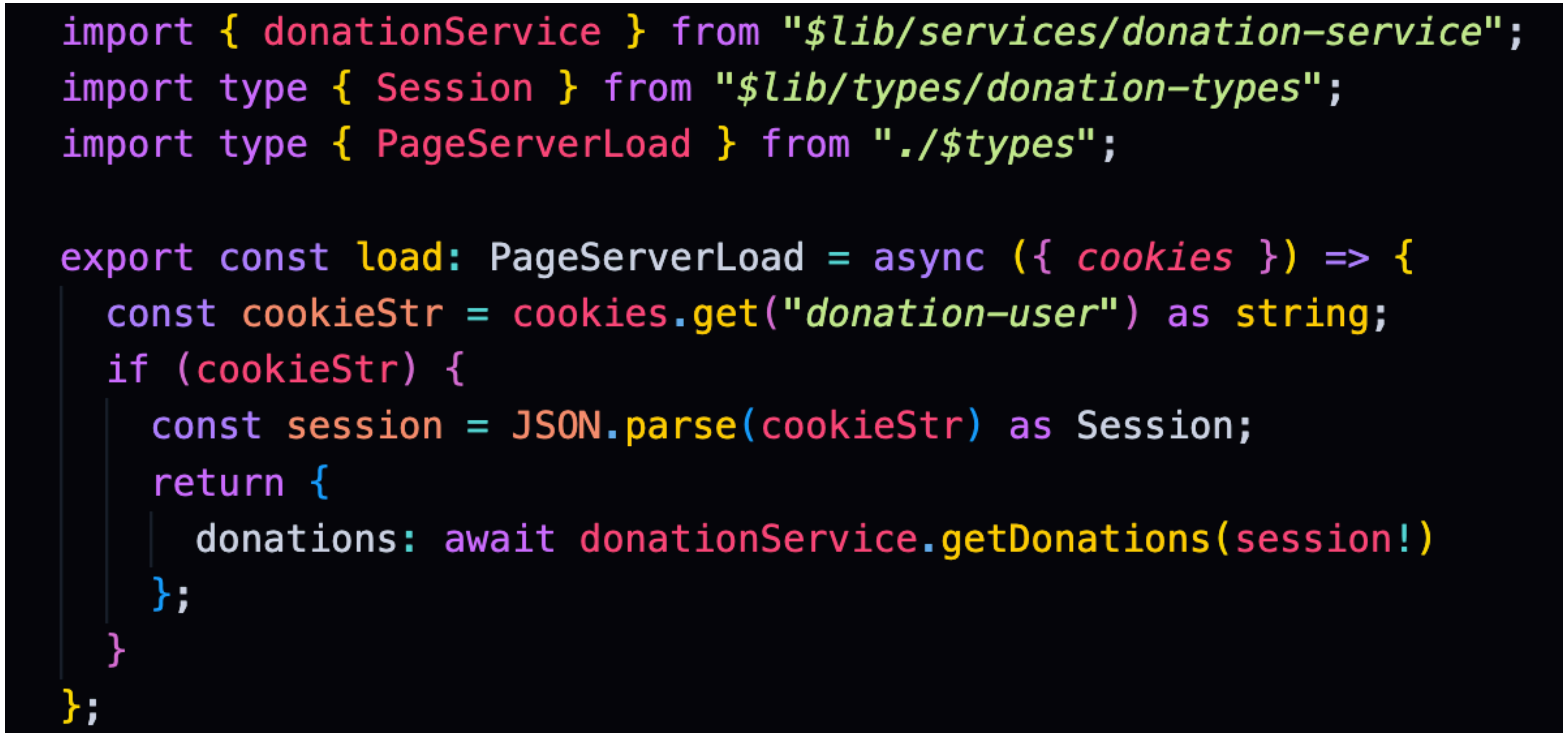
%7B%22name%22%3A%22Homer%20Simpson%22%2C%22token%22%3A%22eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2MjY4YTk2Mjg4OTU1NWU4Y2M0NWQyYiIsImVtYWlsIjoiaG9tZXJAc2ltcHNvbi5jb20iLCJzY29wZSI6W10sImlhdCI6MTcxMzgwMjkxMiwiZXhwIjoxNzEzODA2NTEyfQ.kgcTFT8liX7Aq7iROVQdAIwFSAca8ffTSW2D8_F-tBY%22%7D



```
routes
  charts
    +page.server.ts
    +page.svelte
  donate
    +page.server.ts
    +page.svelte
    DonateForm.svelte
  login
    +page.server.ts
    +page.svelte
    LoginForm.svelte
  logout
    +page.server.ts
    +page.svelte
  maps
    +page.server.ts
    +page.svelte
  report
    +page.server.ts
    +page.svelte
  signup
  +layout.server.ts
  +layout.svelte
  +page.svelte
```

Convert all routes to SSR

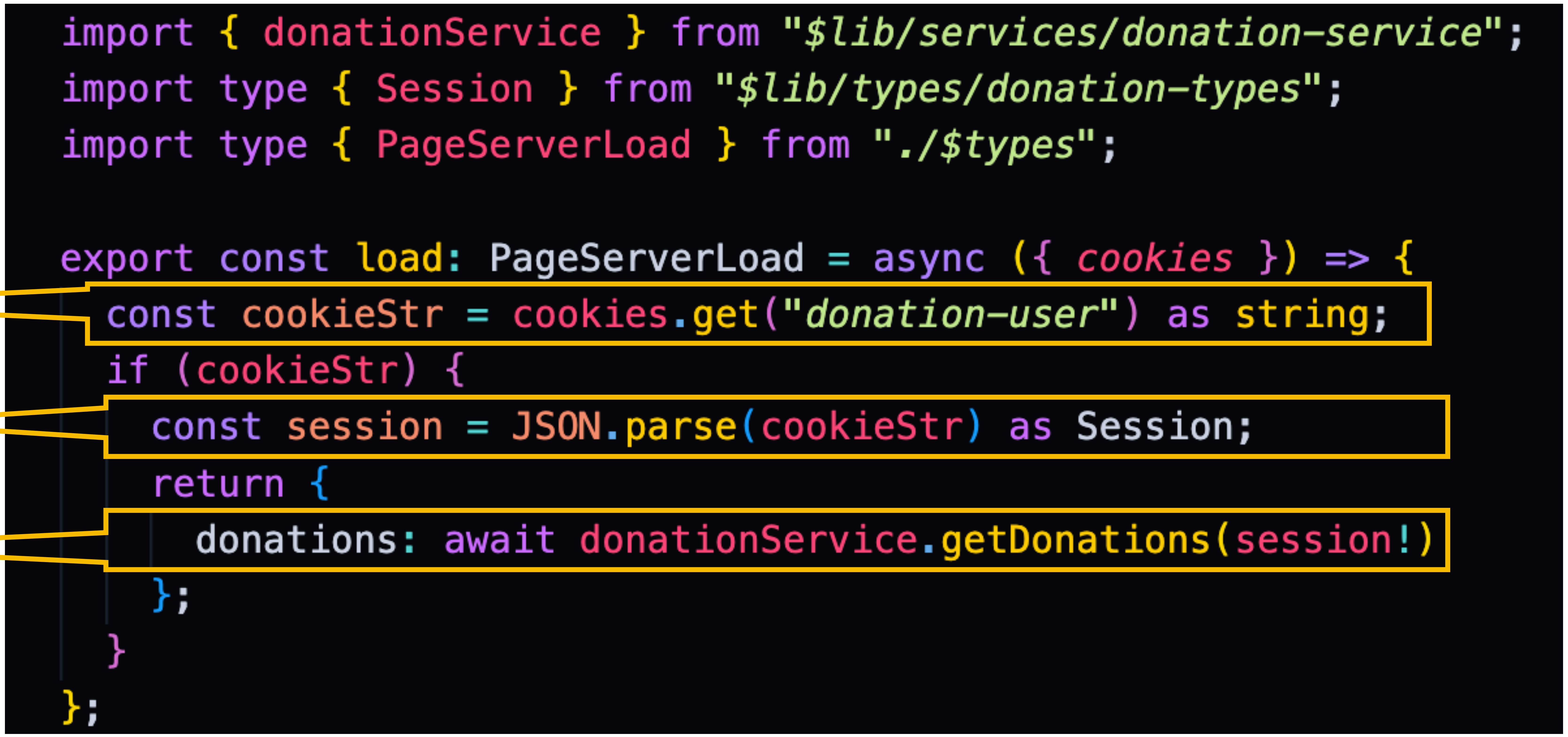
- Each route has +page.server.ts:



```
import { donationService } from "$lib/services/donation-service";
import type { Session } from "$lib/types/donation-types";
import type { PageServerLoad } from "./$types";

export const load: PageServerLoad = async ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      donations: await donationService.getDonations(session!)
    };
  }
};
```

Convert all routes to SSR

- Load the cookie
- Recover the Session
- Access the API

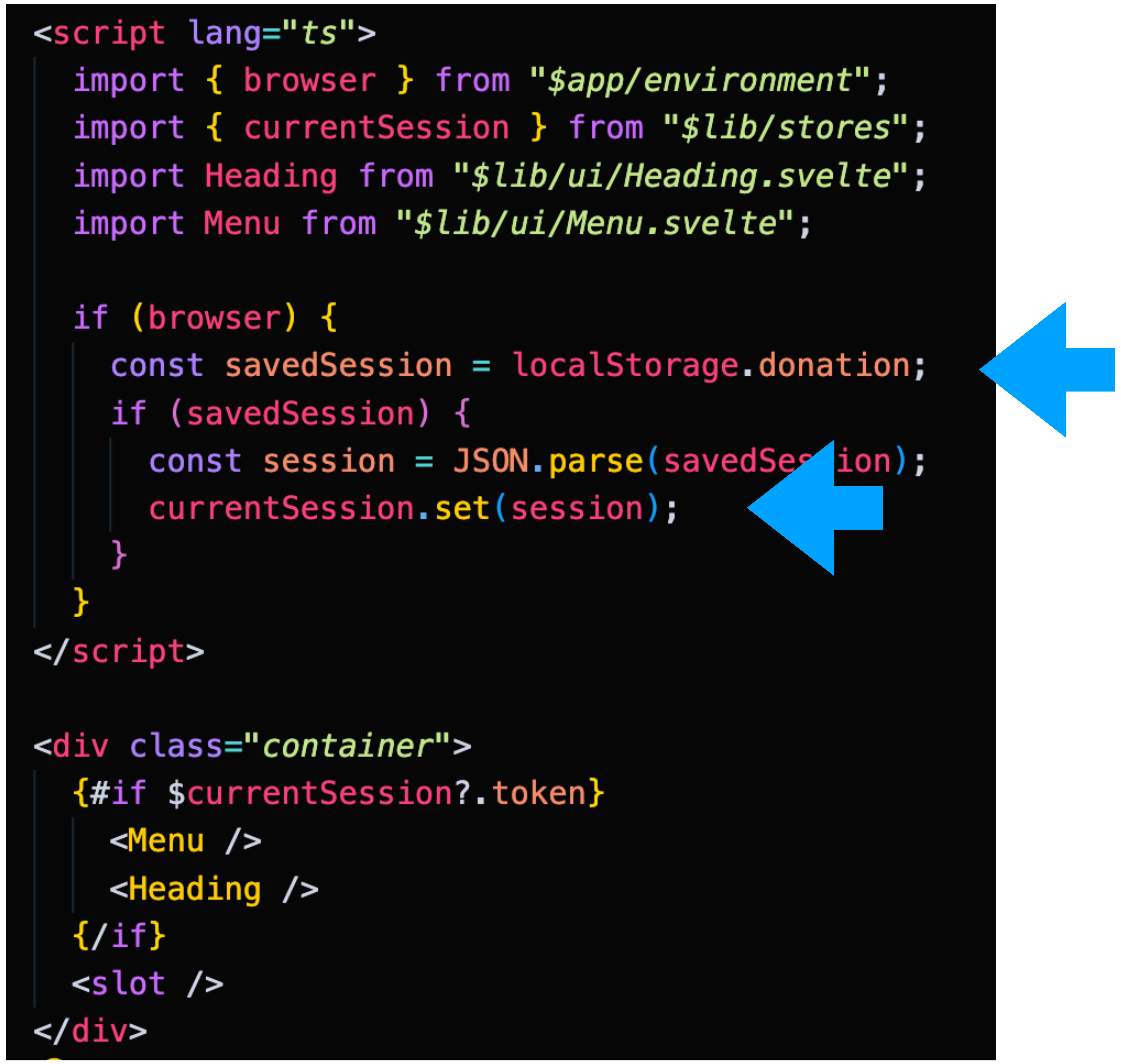


```
import { donationService } from "$lib/services/donation-service";
import type { Session } from "$lib/types/donation-types";
import type { PageServerLoad } from "./$types";

export const load: PageServerLoad = async ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      donations: await donationService.getDonations(session!)
    };
  }
};
```

layout.svelte

- Current version retrieves session from local storage

- Writes session to currentSession store

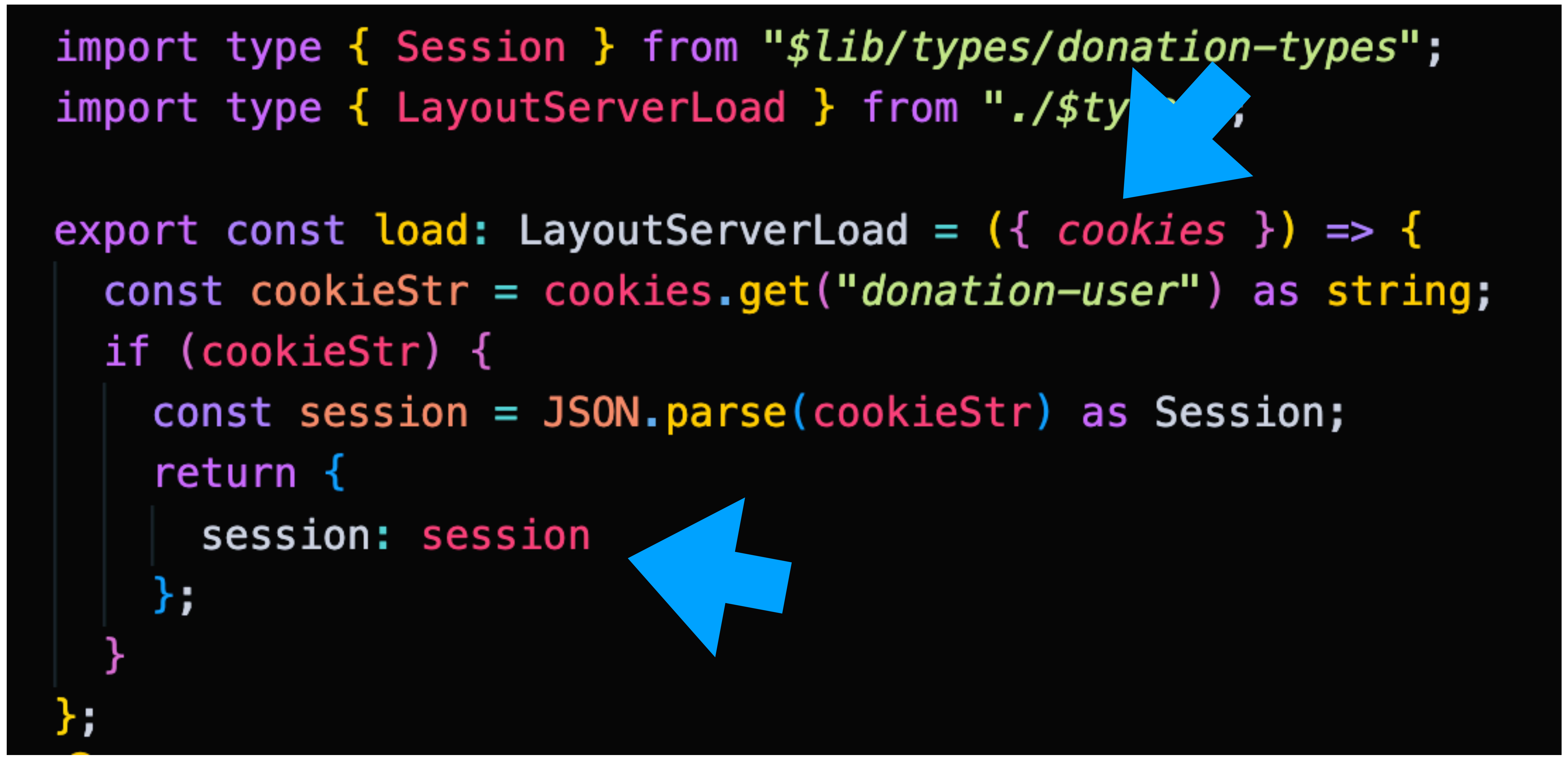


```
<script lang="ts">
  import { browser } from "$app/environment";
  import { currentSession } from "$lib/stores";
  import Heading from "$lib/ui/Heading.svelte";
  import Menu from "$lib/ui/Menu.svelte";

  if (browser) {
    const savedSession = localStorage.donation;
    if (savedSession) {
      const session = JSON.parse(savedSession);
      currentSession.set(session);
    }
  }
</script>

<div class="container">
  {#if $currentSession?.token}
    <Menu />
    <Heading />
  {/if}
  <slot />
</div>
```

- We are no longer using local storage so this will have to change

layout.server.ts

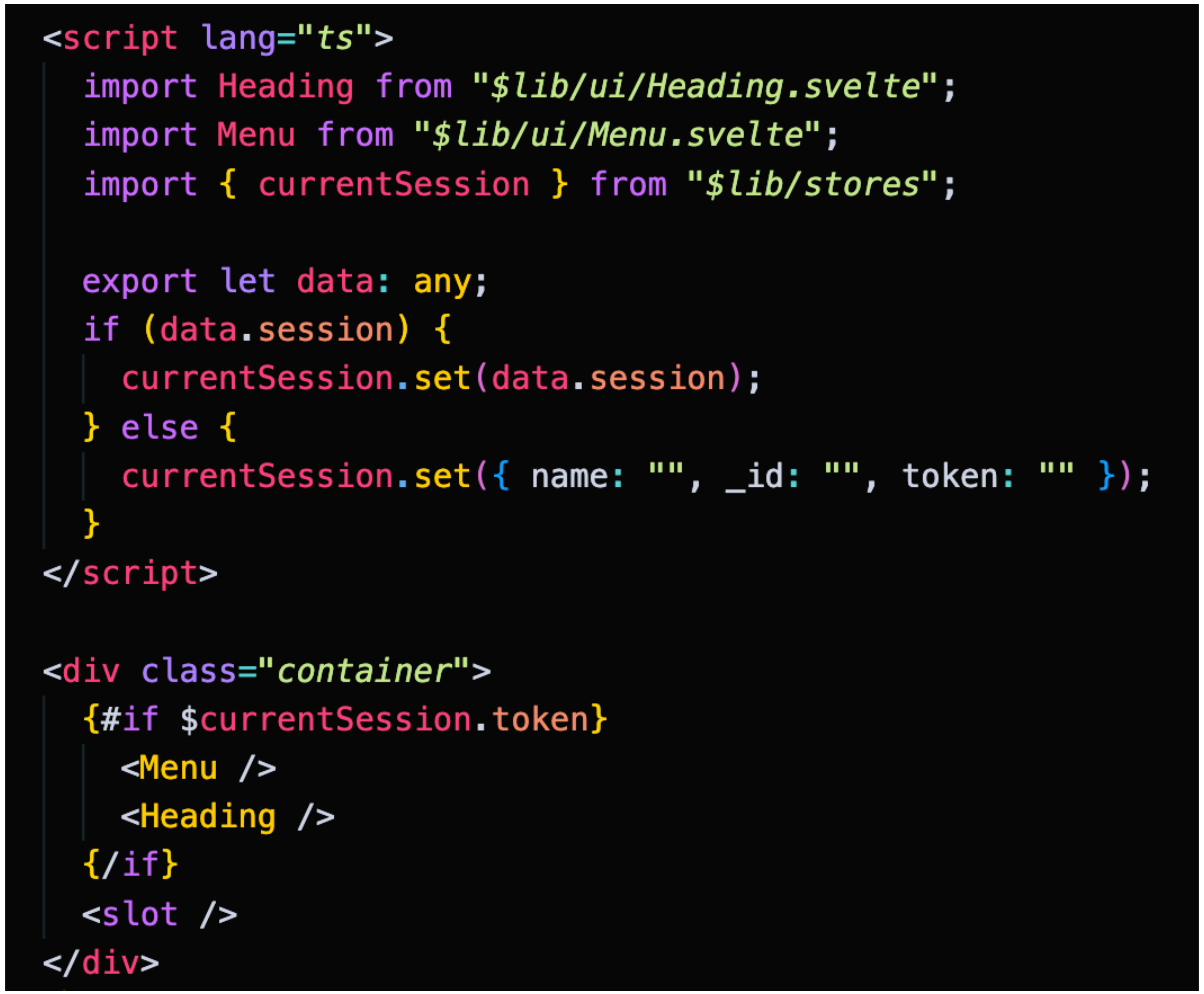
- Always runs on the server
- Has access to cookies interface



```
import type { Session } from "$lib/types/donation-types";
import type { LayoutServerLoad } from "./$ty   ;

export const load: LayoutServerLoad = ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      session: session
    };
  }
};
```

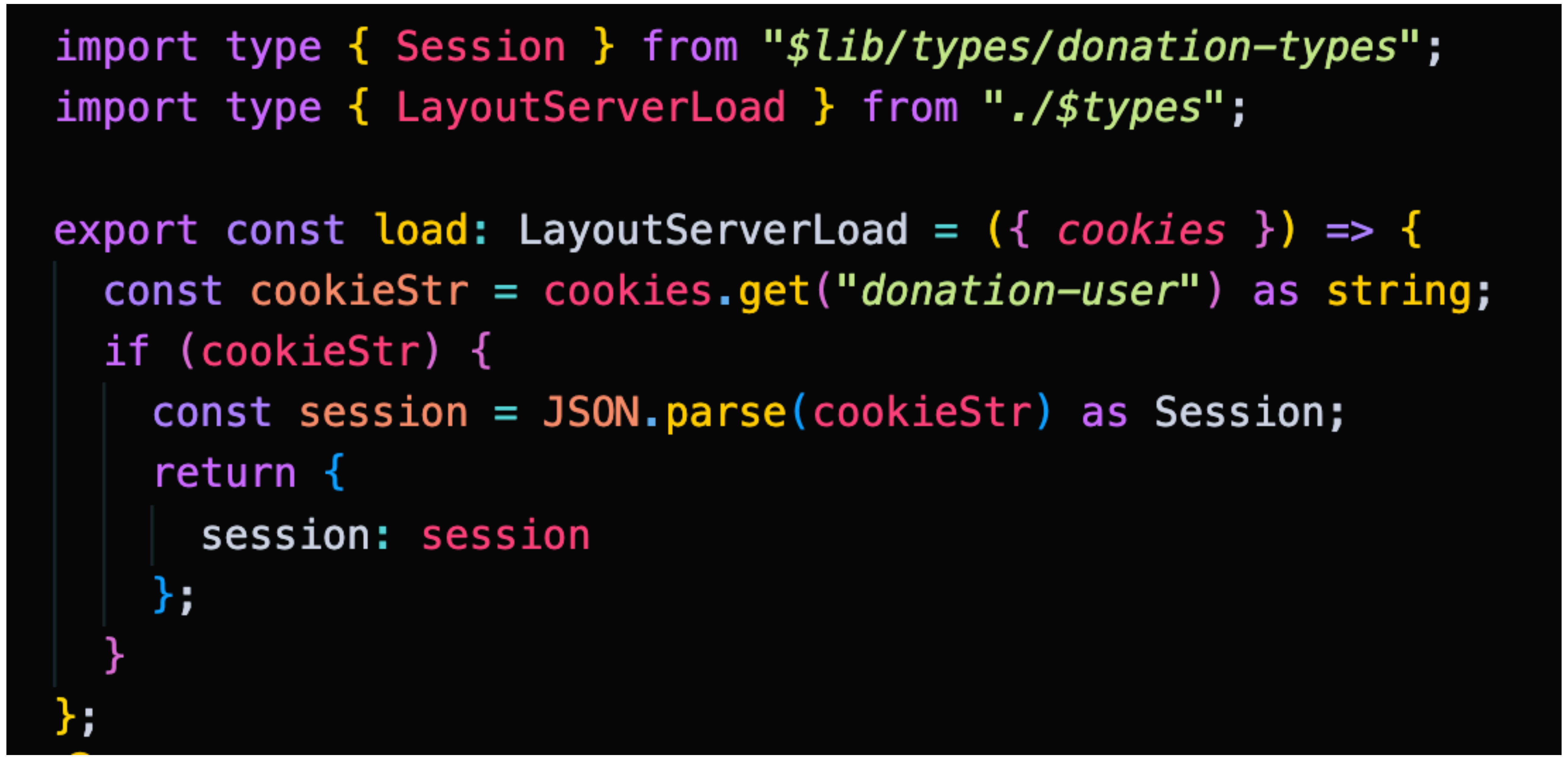
- Retrieve the cookie & return contents as session to the client

layout.svelte



```
<script lang="ts">
  import Heading from "$lib/ui/Heading.svelte";
  import Menu from "$lib/ui/Menu.svelte";
  import { currentSession } from "$lib/stores";

  export let data: any;
  if (data.session) {
    currentSession.set(data.session);
  } else {
    currentSession.set({ name: "", _id: "", token: "" });
  }
</script>

<div class="container">
  {#if $currentSession.token}
    <Menu />
    <Heading />
  {/if}
  <slot />
</div>
```

layout.server.ts



```
import type { Session } from "$lib/types/donation-types";
import type { LayoutServerLoad } from "./$types";

export const load: LayoutServerLoad = ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      session: session
    };
  }
};
```

- Read the cookie in a session object in the server

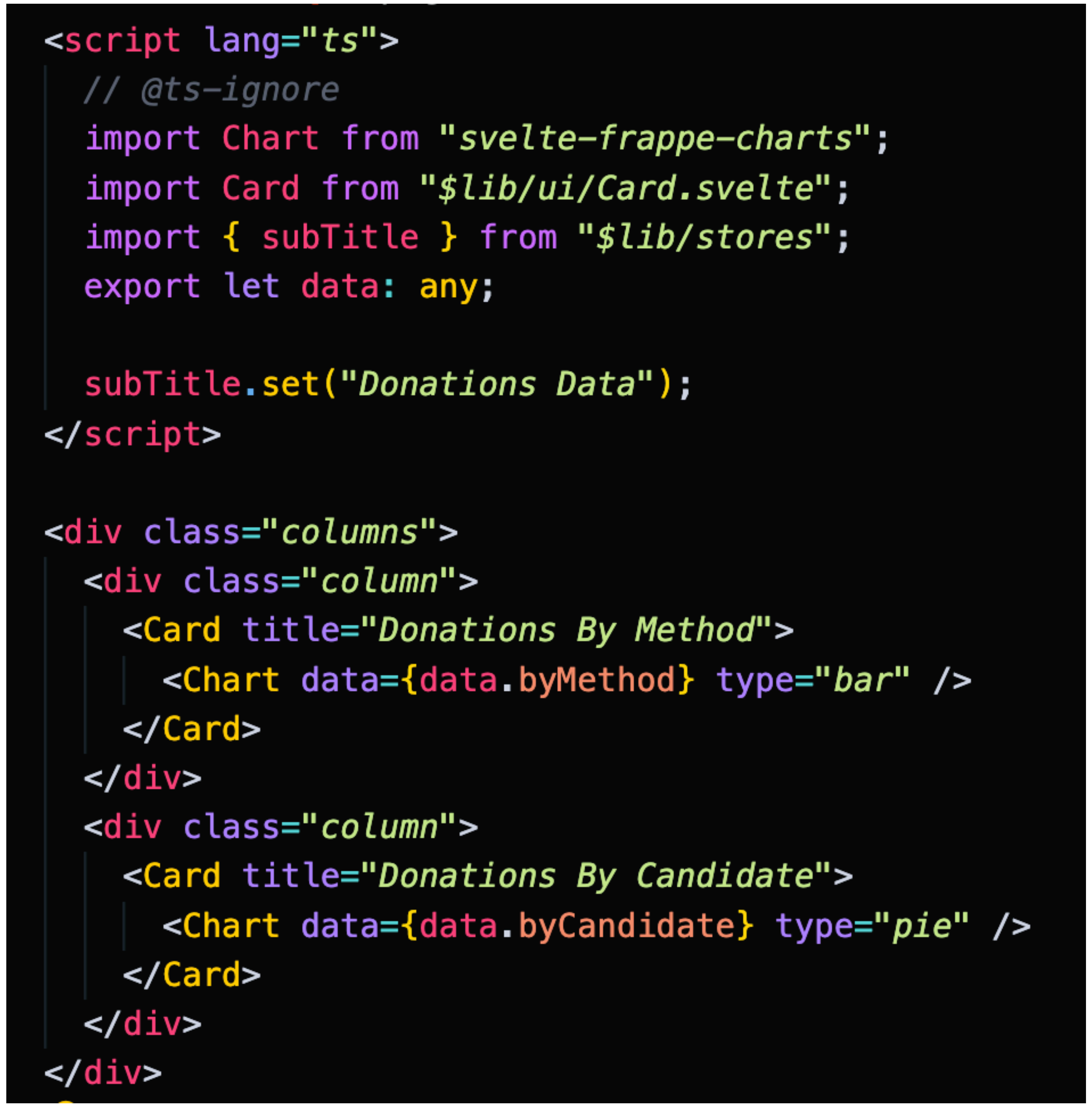
- Write the session into the currentSession store on the client.

- Pass data.byMethod & data.byCandidate to Chart components

- Recover the session

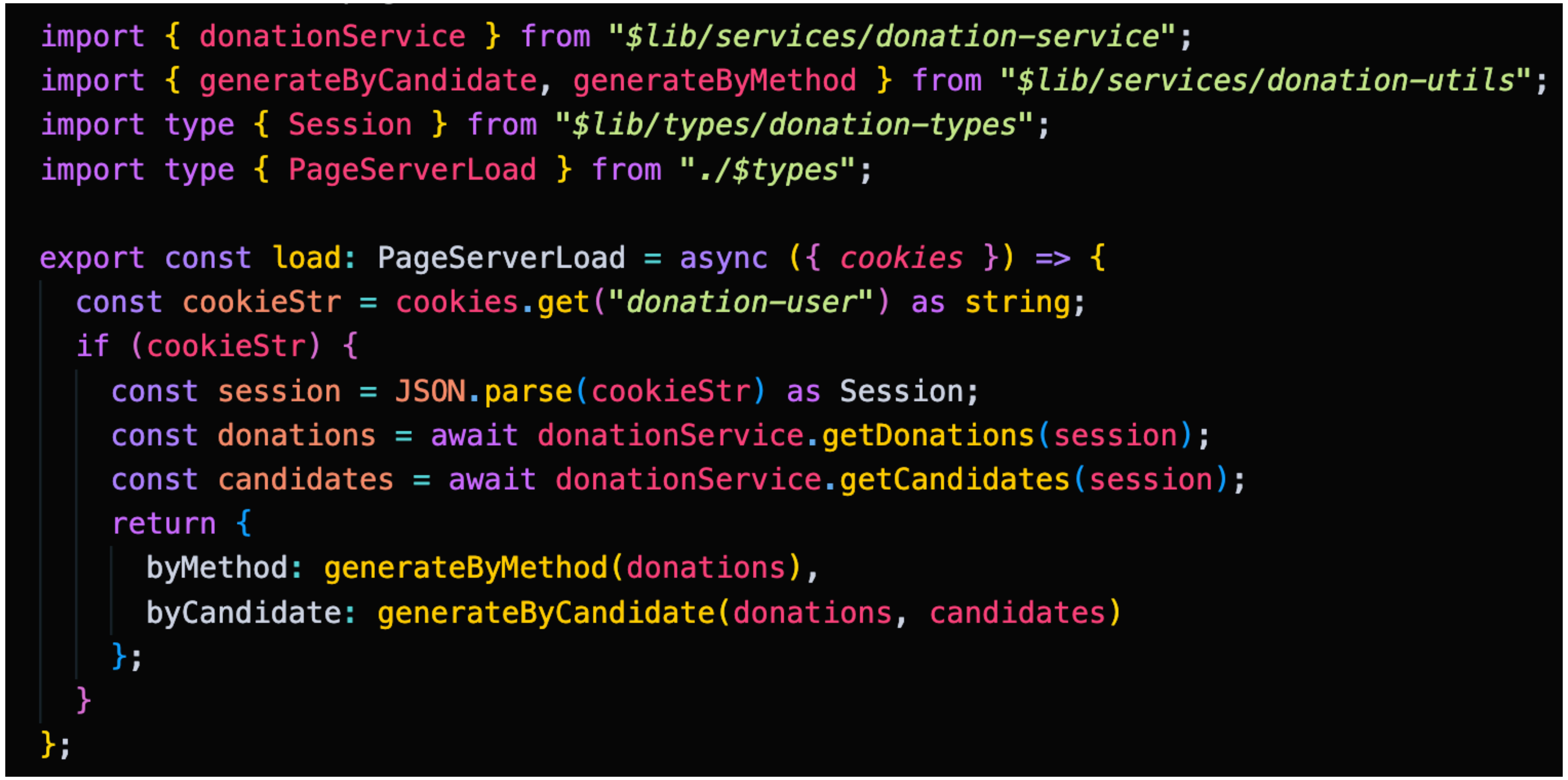
- Access the model

- Return relevant information to the +page.svelte



```
<script lang="ts">
  // @ts-ignore
  import Chart from "svelte-frappe-charts";
  import Card from "$lib/ui/Card.svelte";
  import { subTitle } from "$lib/stores";
  export let data: any;

  subTitle.set("Donations Data");
</script>

<div class="columns">
  <div class="column">
    <Card title="Donations By Method">
      <Chart data={data.byMethod} type="bar" />
    </Card>
  </div>
  <div class="column">
    <Card title="Donations By Candidate">
      <Chart data={data.byCandidate} type="pie" />
    </Card>
  </div>
</div>
```

routes/charts/+page.svelte

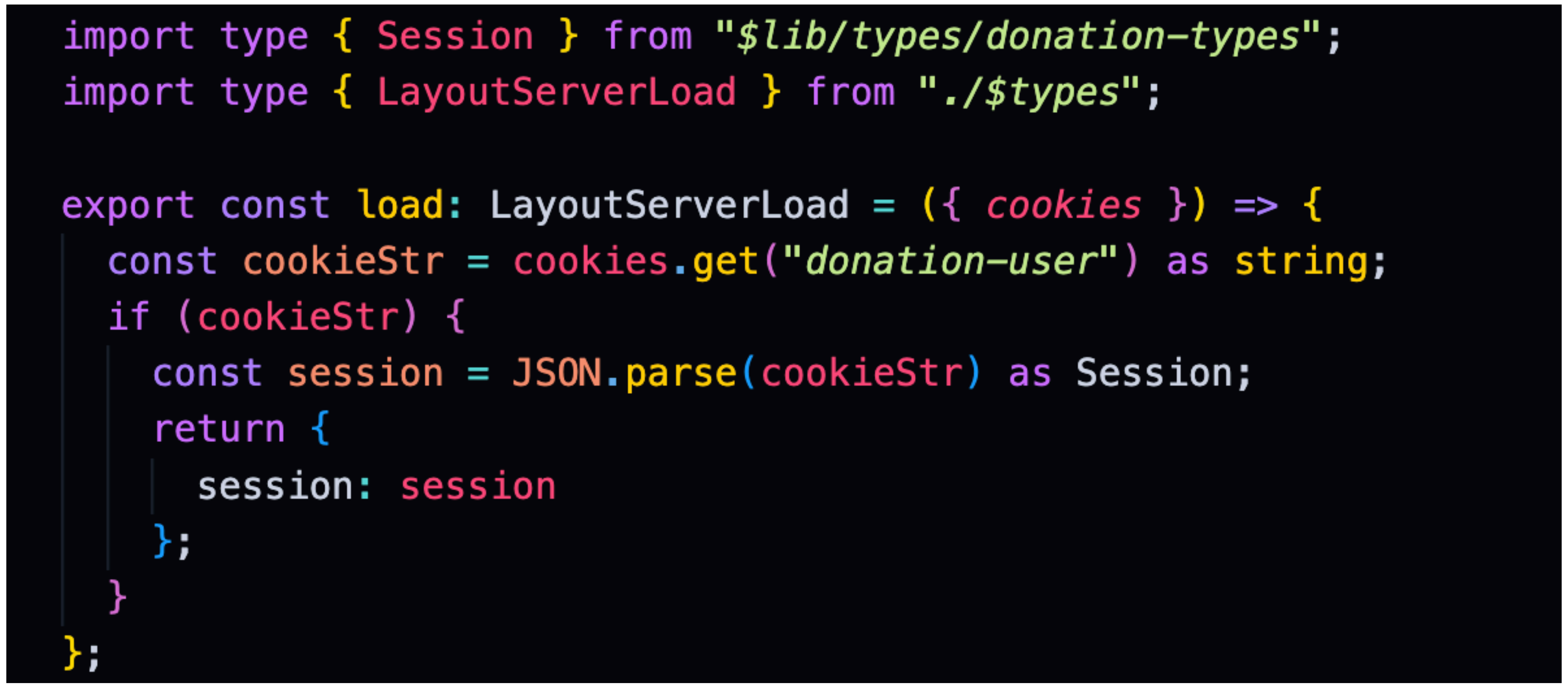


```
import { donationService } from "$lib/services/donation-service";
import { generateByCandidate, generateByMethod } from "$lib/services/donation-utils";
import type { Session } from "$lib/types/donation-types";
import type { PageServerLoad } from "./$types";

export const load: PageServerLoad = async ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    const donations = await donationService.getDonations(session);
    const candidates = await donationService.getCandidates(session);
    return {
      byMethod: generateByMethod(donations),
      byCandidate: generateByCandidate(donations, candidates)
    };
  }
};
```

routes/charts/+page.server.ts

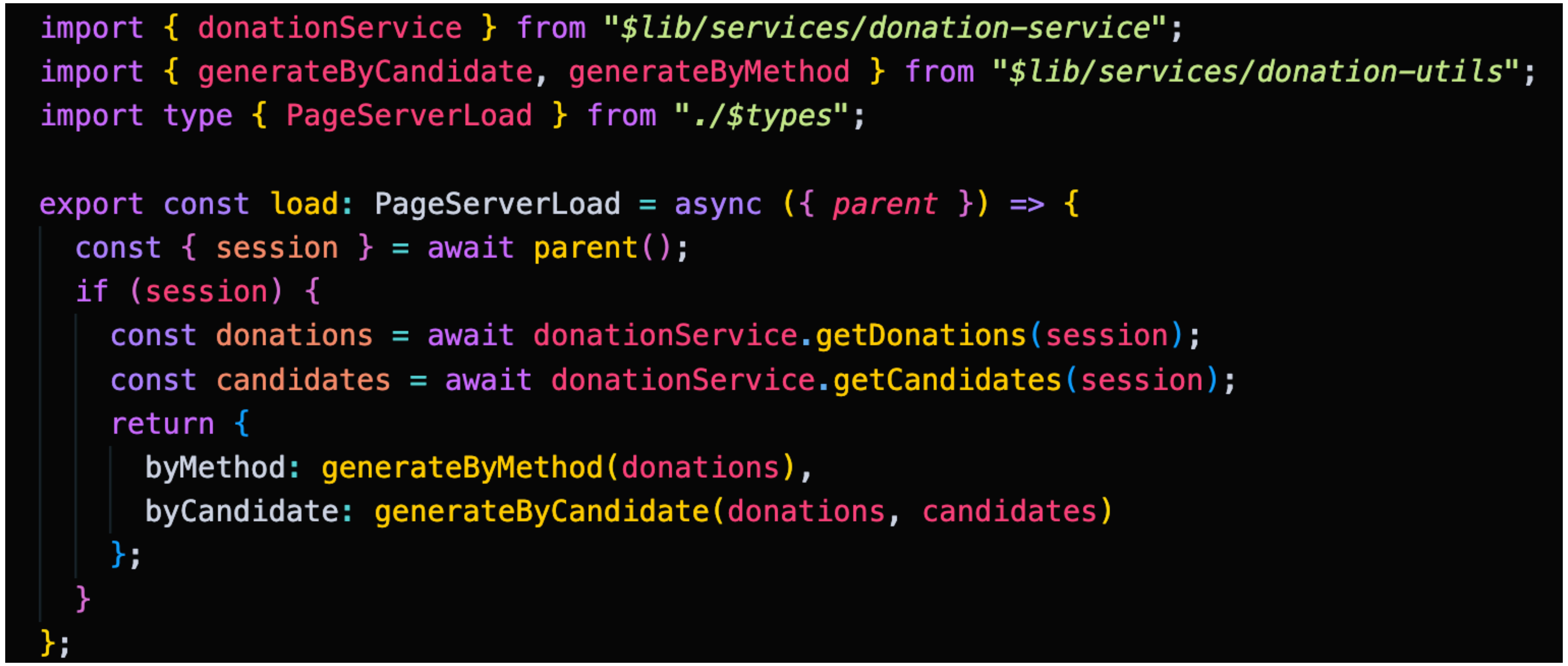
+layout.server.ts

- Recover the session
- Pass session to the view



```ts
import type { Session } from "$lib/types/donation-types";
import type { LayoutServerLoad } from "./$types";

export const load: LayoutServerLoad = ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      session: session
    };
  }
};
```

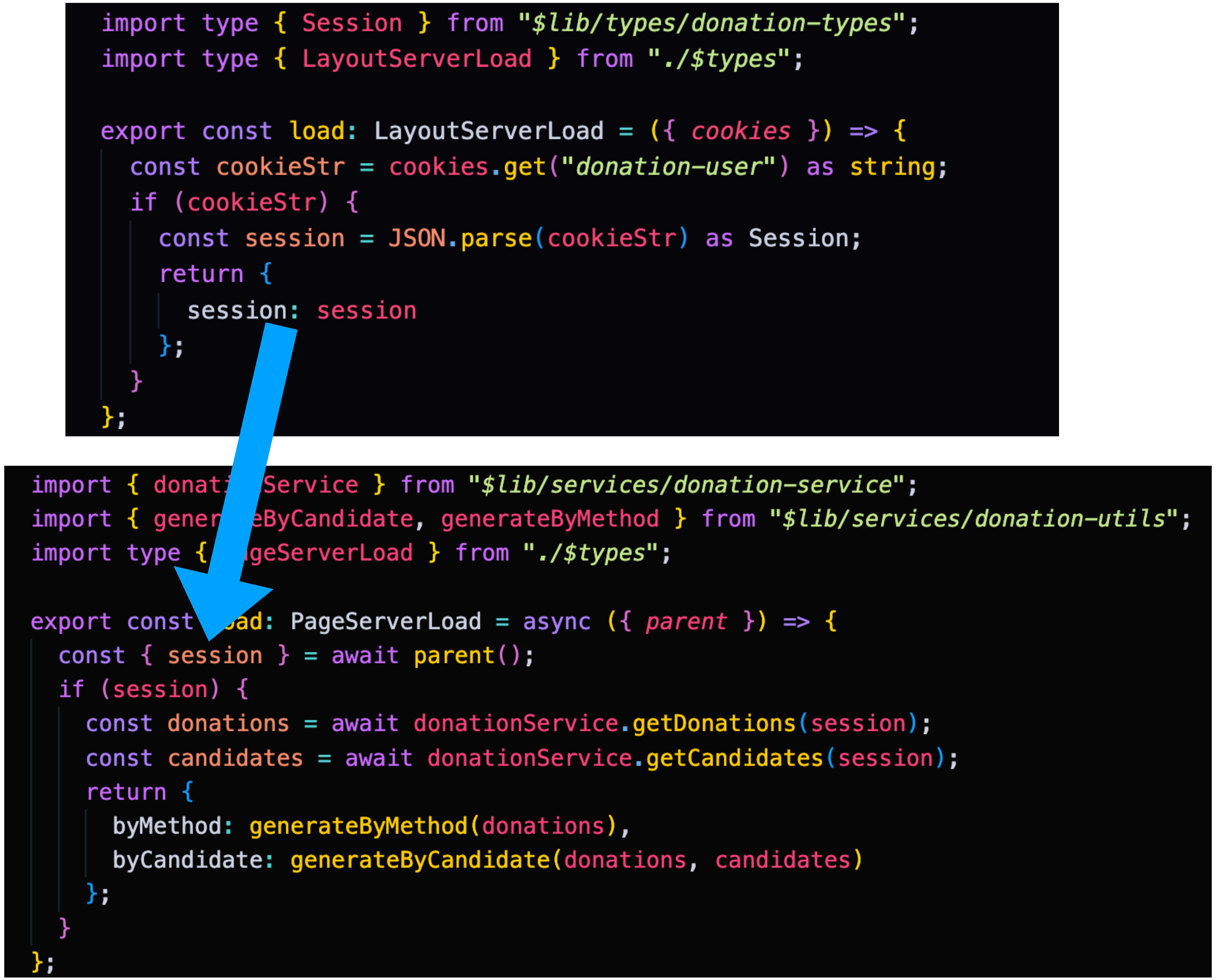
- Recover the session from the “parent”
- The parent (on the server) is the layout.server.ts



```ts
import { donationService } from "$lib/services/donation-service";
import { generateByCandidate, generateByMethod } from "$lib/services/donation-utils";
import type { PageServerLoad } from "./$types";

export const load: PageServerLoad = async ({ parent }) => {
  const { session } = await parent();
  if (session) {
    const donations = await donationService.getDonations(session);
    const candidates = await donationService.getCandidates(session);
    return {
      byMethod: generateByMethod(donations),
      byCandidate: generateByCandidate(donations, candidates)
    };
  }
};
```

routes/charts/+page.server.ts

+layout.server.ts



```
import type { Session } from "$lib/types/donation-types";
import type { LayoutServerLoad } from "./$types";

export const load: LayoutServerLoad = ({ cookies }) => {
  const cookieStr = cookies.get("donation-user") as string;
  if (cookieStr) {
    const session = JSON.parse(cookieStr) as Session;
    return {
      session: session
    };
  }
};
```

```
import { donati[illegible]Service } from "$lib/services/donation-service";
import { gener[illegible]eByCandidate, generateByMethod } from "$lib/services/donation-utils";
import type {[illegible]geServerLoad } from "./$types";

export const [illegible]oad: PageServerLoad = async ({ parent }) => {
  const { session } = await parent();
  if (session) {
    const donations = await donationService.getDonations(session);
    const candidates = await donationService.getCandidates(session);
    return {
      byMethod: generateByMethod(donations),
      byCandidate: generateByCandidate(donations, candidates)
    };
  }
};
```

- +page.server no longer responsible for reading cookie

- This is carried out in +layout.server.ts

routes/charts/+page.server.ts

SSR Cookies

Setting & Getting Cookies in SvelteKit