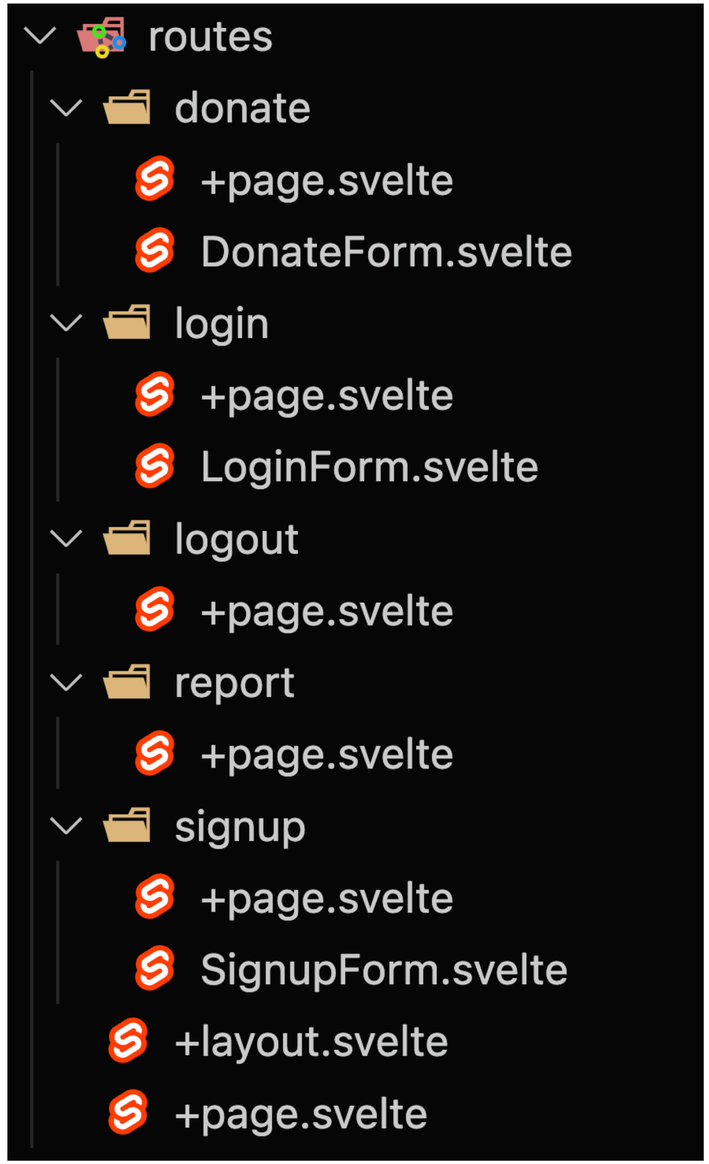
PageLoad in SSR

PageLoad & PageServer
Load functions

SvelteKit Page Components

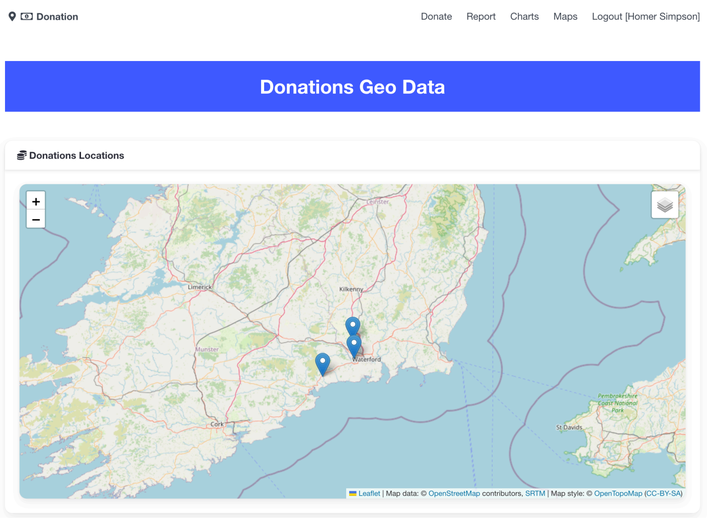
- At the heart of SvelteKit is a filesystem-based router.
- The routes of your app — i.e. the URL paths that users can access — are defined by the directories in your codebase:
  - src/routes is the root route
  - src/routes/report creates a /report route

Each route directory contains one or more route files, which can be identified by their + prefix.



+page.svelte

- A +page.svelte component defines a page of your app.

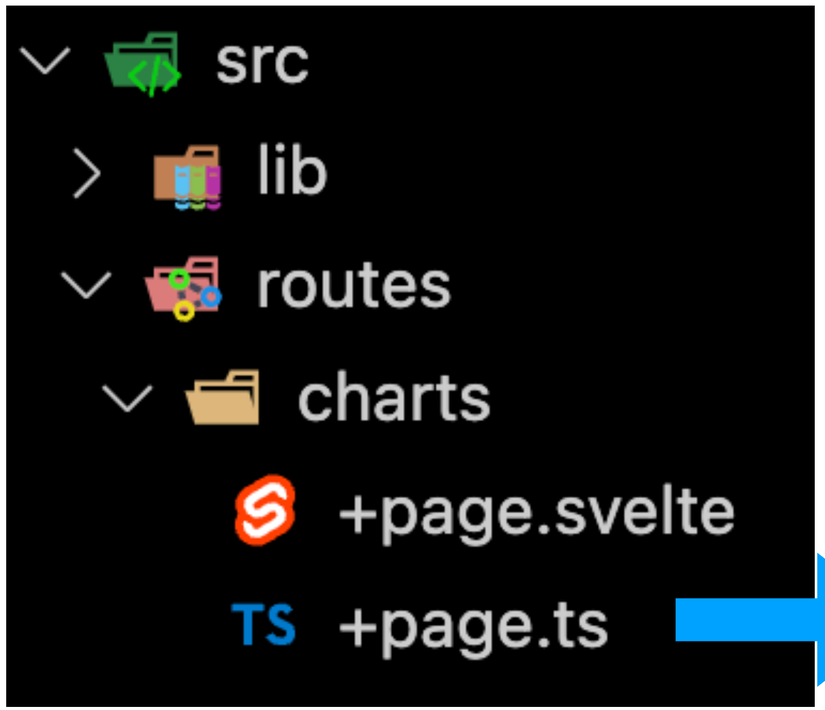



```svelte
<script lang="ts">
  import { subTitle } from "$lib/runes.svelte";
  import { refreshDonationMap } from "$lib/services/donation-utils";
  import Card from "$lib/ui/Card.svelte";
  import LeafletMap from "$lib/ui/LeafletMap.svelte";
  import { onMount } from "svelte";

  subTitle.text = "Donations Geo Data";
  let map: LeafletMap;

  onMount(async () => {
    await refreshDonationMap(map);
  });
</script>

<Card title="Donations Locations">
  <LeafletMap height={60} bind:this={map} />
</Card>
```

By default, pages are rendered both on the server (SSR) for the initial request and in the browser (CSR) for subsequent navigation

+page.ts

- Often, a page will need to load some data before it can be rendered.
- For this, we add a +page.ts module that exports a load function:




```ts
import { loggedInUser } from "$lib/runes.svelte";
import { donationService } from "$lib/services/donation-service";
import type { PageLoad } from "./$types";

export const load: PageLoad = async () => {
  return {
    donations: await donationService.getDonations(loggedInUser.token),
    candidates: await donationService.getCandidates(loggedInUser.token)
  }
}
```

This function runs alongside +page.svelte, which means it runs on the server during server-side rendering and in the browser during client-side navigation

- **load** function reads donations from the donationService (API)

+page.ts

```ts
import { loggedInUser } from "$lib/runes.svelte";
import { donationService } from "$lib/services/donation-service";
import type { PageLoad } from "./$types";

export const load: PageLoad = async () => {
  return {
    donations: await donationService.getDonations(loggedInUser.token),
    candidates: await donationService.getCandidates(loggedInUser.token)
  }
}
```

```svelte
<script lang="ts">
  import { curentDataSets } from "$lib/runes.svelte";
  // @ts-ignore
  import Chart from "svelte-frappe-charts";
  import Card from "$lib/ui/Card.svelte";
  import type { PageProps } from "./$types";
  import { computeByCandidate, computeByMethod } from "$lib/services/donation-utils";

  let { data }: PageProps = $props();

  computeByMethod(data.donations);
  computeByCandidate(data.donations, data.candidates);
</script>

<div class="columns">
  <div class="column">
    <Card title="Donations By Method">
      <Chart data={curentDataSets.donationsByMethod} type="bar" />
    </Card>
  </div>
  <div class="column has-text-centered">
    <Card title="Donations By Method">
      <Chart data={curentDataSets.donationsByCandidate} type="pie" />
    </Card>
  </div>
</div>
```

+page.svelte

- **data** object contains values returned from the **load** function

- **load** function reads donations from the donationService (API)

+page.ts

```ts
import { loggedInUser } from "$lib/runes.svelte";
import { donationService } from "$lib/services/donation-service";
import type { PageLoad } from "./$types";

export const load: PageLoad = async () => {
  return {
    donations: await donationService.getDonations(loggedInUser.token),
    candidates: await donationService.getCandidates(loggedInUser.token)
  };
};
```

```svelte
<script lang="ts">
  import { curentDataSets } from "$lib/runes.svelte";
  // @ts-ignore
  import Chart from "svelte-frappe-charts";
  import Card from "$lib/ui/Card.svelte";
  import type { PageProps } from "./$types";
  import { computeByCandidate, computeByMethod } from "$lib/services/donation-utils";

  let { data }: PageProps = $props();

  computeByMethod(data.donations);
  computeByCandidate(data.donations, data.candidates);
</script>

<div class="columns">
  <div class="column">
    <Card title="Donations By Method">
      <Chart data={curentDataSets.donationsByMethod} type="bar" />
    </Card>
  </div>
  <div class="column has-text-centered">
    <Card title="Donations By Method">
      <Chart data={curentDataSets.donationsByCandidate} type="pie" />
    </Card>
  </div>
</div>
```

+page.svelte

- **data** object contains values returned from the **load** function

Customise Rendering Behaviour

- As well as **load,** +page.ts can export values that configure the page's behaviour.
- SSR set to false forces all rendering to take place in the browser.
- i.e. page.ts will always run win the browser

```
import { loggedInUser } from "$lib/runes.svelte";
import { donationService } from "$lib/services/donation-service";
import type { PageLoad } from "./$types";

export const ssr = false;

export const load: PageLoad = async ({ }) => {
  return {
    donations: await donationService.getDonations(loggedInUser.token),
    candidates: await donationService.getCandidates(loggedInUser.token)
  }
}
```

export const ssr = false;

+page.ts

- If the load function can only run on the server rename:
  - +page.ts to +**page.server.ts**
  - PageLoad type to PageServerLoad.

```ts
import { loggedInUser } from "$lib/runes.svelte";
import { donationService } from "$lib/services/donation-service";
import type { PageLoad } from "./$types";

export const load: PageLoad = async () => {
  return {
    donations: await donationService.getDonations(loggedInUser.token),
    candidates: await donationService.getCandidates(loggedInUser.token)
  }
}
```

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

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

+page.ts

- This may be appropriate if the app needs to fetch data from a database

- Or you need to access private environment variables like API keys

```ts
import { loggedInUser } from "$lib/runes.svelte";
import { donationService } from "$lib/services/donation-service";
import type { PageLoad } from "./$types";

export const load: PageLoad = async () => {
  return {
    donations: await donationService.getDonations(loggedInUser.token),
    candidates: await donationService.getCandidates(loggedInUser.token)
  }
}
```

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

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

PageLoad in SSR

PageLoad & PageServer
Load functions