Navigation & Astro API

Navigieren

Ersteinmal ist das Navigieren denkbar einfach. Wir setzen einfach einen Link.

```
<a href="/impressum">Impressum</a>
```

Easy peasy lemon squeezy.

Mit allen Links wäre das:

```
<nav>
    <a href="/">Home</a>
    <a href="/programm">Programm</a>
    <a href="/programm/speaker">Speakers</a>
    <a href="/impressum">Impressum</a>
</nav>
```

Manuelle Pflege aller Links? Mögen wir das?

Natürlich nicht!

Ausserdem müssen wir das auf jeder Seite einbauen! So geht das nicht!

Auslagern der Navigation in eine Komponente

Zuerst lagern wir die Navigation in eine Komponente aus.

Wir erstellen eine neue Navigation Component und nutzen diese dann in allen Seiten (mehr dazu später)

```
---
import Navigation from "../../components/Navigation.astro";
---

<Navigation />
```

Dann setzten wir ggf. noch eine active klasse auf die Elemente. Und fangen an zu automatisieren.

Automatisieren der Navigation

Astro bietet uns eine kleine API mit funktionen die man oft benötigt.

In unserem Fall ist das die `Astro.glob()` funktion.

```
---
const allpages = await Astro.glob("../pages/**/*.astro");
---
```

Glob ist die Implementierung die ihr aus vielen Sprachen und Tools ggf. schon kennt. Auf diese Syntax gehen wir aber nicht in die Tiefe ein.

- Alle .astro Dateien aus pages.
- Inklusive aller Unterverzeichnisse.
- Kann in größeren Projekten natürlich komplexer aussehen
- Erfordert organisation

Automatisieren der Navigation II

Nachdem wir die Liste an Dateien erlangt haben müssen wir diese auch ausgeben.

```
---
const allpages = await Astro.glob("../pages/**/*.astro");
---
<nav>
    {allpages.map(({ url, title }) => {
        if (!url) { url = "/"; } // Home URL
        return <a href={url}>{title}</a>;
    })}
</nav>
```

Typisierung

Damit astro bzw. Typescript sich später nicht beschwert fügen wir noch eine Typisierung hinzu.

```
import type { AstroInstance } from "astro";

let allpages = await Astro.glob<AstroInstance & { title: string }>(
    "../pages/**/*.astro",
);
```

So ist der korrekte Typ auf der Seite bekannt.

Logik einbauen

Links brauchen ggf. eine active CSS Klasse. Wir nutzen den aktuellen Pfad um das zu ermitteln.

```
const pathname = new URL(Astro.request.url).pathname;
...
return (
    <a class={pathname === url ? "active" : ""} href={url}>
        {title}
    </a>
)
...
```

Mit JS nach Bedarf weitere Verfeinerungen, Sortierungen… vornehmen.

Aufgaben

1. Probiere ob es bei deiner Seite funktioniert
2. Filtere das Impressum aus der Navigation heraus