The qwik workshop

Another front-end framework?

- 
- SVELTE
- 
- ANGULAR
- qwik
-
- 
- astro

The selling points

- Resumability
- Familiar dx

Time to get your hands dirty

- We'll be building a small app for listing the past Knowit JS guild meetings
- The app has already been built using Next.js
- Your job: transfer that app to Qwik
- https://qwik.builder.io/docs/
- https://github.com/cybercom-finland/code-forward-23

1. Clone the repo `` `git clone git@github.com:cybercom-finland/code-forward-23.git` ``
2. Go to `` `./qwik/next/` `` and run `` `npm install && npm run build && npm start` ``

Step 1: Skaffold Qwik and load data

```
# Go to ./qwik

npm create qwik@latest
# use e.g. qwik-yourname as the project name and  select "Emp
# Install these additional depencies,
npm install @amcharts/amcharts5 rambda amcharts capitalize @t

# Alternatively: use ./qwik/qwik-empty
```

```
export const useGuildList = routeLoader$(async () => {
  const data = // await  fetch etcd
  return data;
});

export default component$(() => {
  const guildListSignal = useGuildList();
  // guildListSignal.value
  return <div></div>
}
```

- Edit routes/src/index.tsx so that it displays the guild list found at `./qwik/common/guildlist.json` as a json string
- Treat the loading of guildlist.json as server-side data fetching in the Next.js version
- Boilerplate can be copied from `./qwik/samples/index.tsx` --> `./qwik/qwik-yourname/src/routes/index.tsx`

5 min	Start

Step 2: Port GuildList and GuildItem to qwik

- `next/src/components/GuildList.tsx` --> `qwik/qwik-juho/src/components/GuildList.tsx`
- `next/src/components/GuildItem.tsx` --> `qwik/qwik-juho/src/components/GuildItem.tsx`
- `next/src/app/globals.css` --> `qwik/qwik-juho/src/global.css`
- Render GuildList with the prefetched data in `routes/index.tsx`
- `component$`
- `onClick$`
- `class`
- `GuildItem` needs some trimming at this point, we'll add the missing logic later:
  - comment out `useStates` and `useEffects`
  - make onClick-handlers just return null
  - hard-code `showStats` to `false`

10 min | Start

```tsx
// ./src/routes/index.tsx

<main><GuildList guilds={guildListSignal.value}/></main>
```

```tsx
// next/components/GuildList
export const GuildList: FC<Props> = (props) => {
  if (!Array.isArray(props.guilds)) return;
  return (
    <ul>
      {props.guilds.map((repo) =>  <li key={repo.id} />)}
    </ul>
  );
};
```

```tsx
// ./qwik/components/GuildList
export default component$<Props>((props) => {
  if (!Array.isArray(props.guilds)) return null;
  return (
    <ul>
      {props.guilds.map((repo) =>  <li key={repo.id} />)}
    </ul>
  );
});
```

Step 3: Port SideBar and add the interactions

- `next/src/components/SideBar.tsx` --> `qwik/qwik-juho/src/components/SideBar.tsx`
- Render `SideBar` besides GuildList at `routes/index.tsx`
- Convert the `useContext` + `useReducer` combo to a context + store combo
  - start by just:

```
//const state = useContext(MainContext);
//const dispatch = useContext(MainDispatchContext);
const state = { favouriteGuild: undefined, showAllStats: false };
```

- Use Qwik's version of context combined with *a store* to set the logic inside side bar and guildItem (we'll do this together soon, but feel free to advance by yourself)

5 min | Start

Let’s deal with that store thing together...

1. Create a context in `routes/index.tsx`

```
export const CTX = createContextId<AppState>("some.id");
```

2. Create a store

```
const appData = useStore({
    favouriteGuild: "",
    showAllStats: false
})
```

3. Set the created store as the value of the created context

```
useContextProvider(CTX, appData);
```

Inside `./components/SideBar.tsx`

1. use the context

```
const state = useContext(CTX);
```

2. Change the state:

```
  <button
    onClick$={() => {
      state.showAllStats = !state.showAllStats;
    }}
  >
```

3. Your job: adjust the logic for setting a favourite guild inside `./src/components/GuildListItem.tsx`

3 min	Start

Step 4: Add stats to GuildItem and finalize

- Use the sample: `./qwik/samples/RepoStats.tsx` --> `./qwik-yourname/src/components/RepoStats.tsx`
- In GuildItem: Add logic for showing and hiding based on either local or global state

```
const showStats = useSignal(false);
useTask$(({ track }) => {
  track(() => state.showAllStats);
  showStats.value = state.showAllStats;
});
```

- ...and render the RepoStats component

```
<div>
  {showStats.value ? <RepoStats name={repo.name} /> : <p>Stats hidden</p>}
</div>
```

Testing and comparing

- Stop the `npm run dev` of your quick app
- Start again by running `npm run preview`
- Open your network tab, disable cache and set throttling to slow 3g