# Fullstack Development

# Data Fetching in React / NextJS

https://github.com/fullstack-66/data-fetching

# Setup time server

- Clone https://github.com/fullstack-66/time-server
- `npm install`
- `npm start`

# Setup NextJS project

```
npx create-next-app@latest
```

# Configure import

`tsconfig.json`

```
{
  "baseUrl": ".",
  "paths": {
    "@app/*": ["app/*"],
    "@components/*": ["components/*"]
  }
}
```

- Note, if you don't define `baseUrl`, you need to prefix the path with `./`.

# Create additional page

`./app/another/page.tsx`

```tsx
export default function AnotherPage() {
  return <div>Another Page</div>;
}
```

# Add navigation

./app/layout.tsx

```tsx
import Link from "next/link";
// ...
export default function RootLayout(...) {
  return (
    <html lang="en">
      <body className={inter.className}>
        <div className="flex gap-2 mb-4">
          <Link href="/">Home</Link> 👈
          <Link href="/another">another</Link> 👈
        </div>
        <div className="m-4">{children}</div>
      </body>
    </html>
  );
}
```

# Fetching #1: Server component

- `./components/t1_serverComponent/index.tsx`
- Notice the caching behavior.

# Also used

- `DisplayTime` component from `./components/utils/displayTime.tsx`
- Type from `./components/uitls/types.ts`
  - Generated using `Paste as Code` extension.

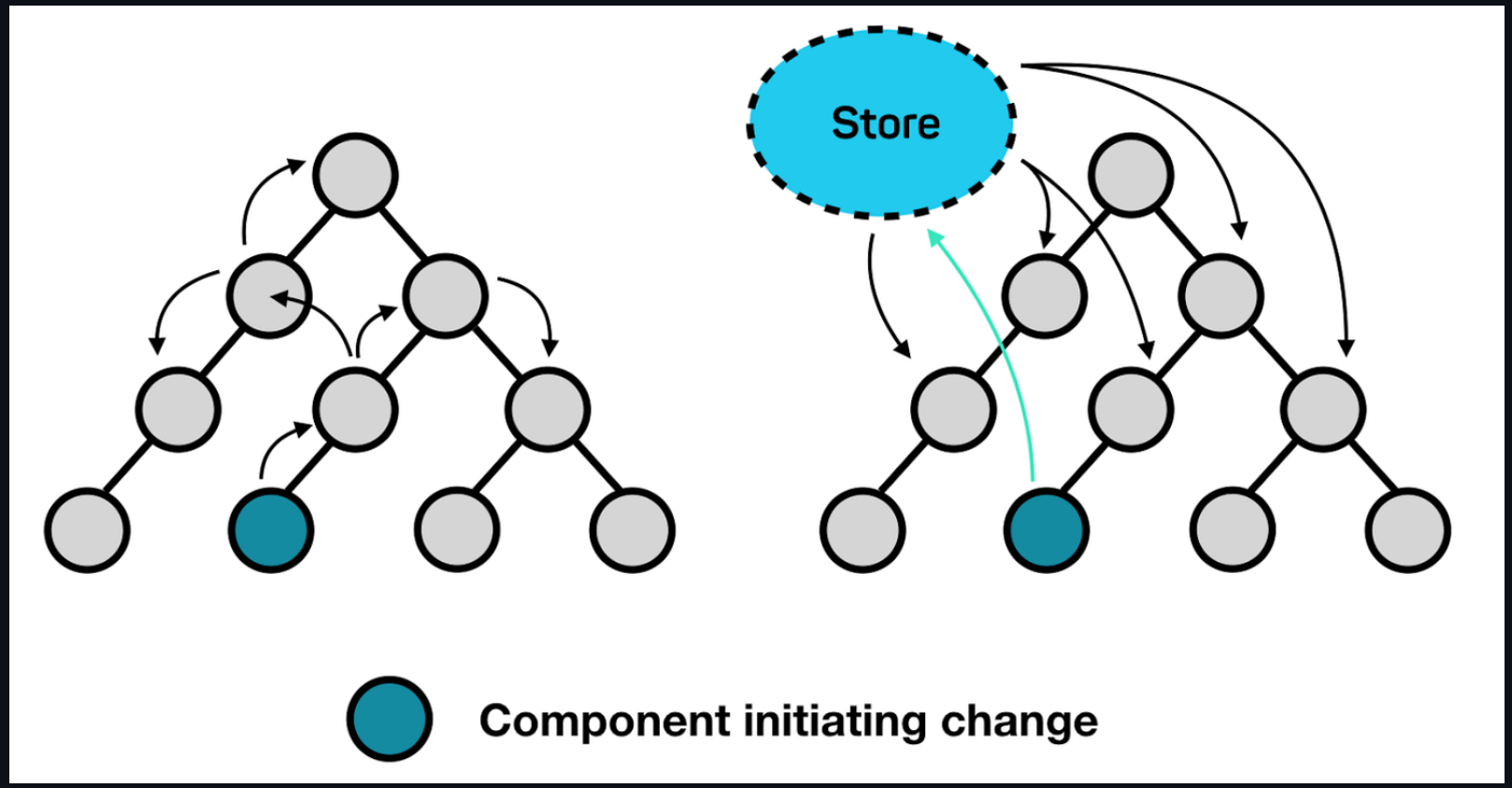
# Fetching #2: `useEffect` way

`./t2_clientUseEffect/index.ts`

# Global store pattern

What does using a global store solve?

- Prop drilling
- Unncessary re-render


Store
Component initiating change

# Global store libraries / API

- `React Context`
- `Redux`
- `Jotai`
- `Zustand`
  - We will use this one for now.

# Zustant

- `npm install zustant`

./components/utils/store.ts

```ts
import { create } from "zustand";
import { type Time } from "./types";

interface Store {
  time: Time | null;
  setTime: (time: Time) => void;
}

const useStore = create<Store>((set) => ({
  time: null,
  setTime: (time) => set({ time }),
}));

export default useStore;
```

# Fetching #3: Global store

- `./components/t3_clientGlobalStore/index.ts`
- `./components/t4_clientGlobalStoreUpdate/index.ts`
  - Notice how the data change in both components.

# Fetching #4: React Query

- Data-fetching + state management library
- Highly recommended!

# Installation

- `npm install @tanstack/react-query`
- `npm install -D @tanstack/react-query-devtools`

# Provider

- `./components/uitls/reactQueryProvider.tsx`
- `./app/layout.tsx`

```
import ReactQueryProvider from "@components/utils/reactQueryProvider";
...
<ReactQueryProvider>
  <div className="m-4">{children}</div>
</ReactQueryProvider>
...
```

# Fetching with React Query

- `./components/t5_clientReactQuery/index.tsx`
- `./components/t6_clientReactQueryTwo/index.tsx`
  - Notice how the data is cached and refetched.

# Extra: use custom hook

- `./components/util/reactQueryData.ts`
- Refactor
  - `./components/t5_clientReactQuery/index.tsx`
  - `./components/t6_clientReactQueryTwo/index.tsx`