Kurs med Online, våren 2025

Hvordan fungerer React egentlig?

Kjøreplan 🕐

- 17:00: Oppmøte
- 17:15: Kurset begynner
- 18:30: Pause med Pizza 🍕
- 20:00: Kurs ferdig
- 20:00 - 21:00: Mingling og hygge på det nye Varianthuset 🏠

Variant

Jakob Endrestad Kielland

Driver med frontend og ansvarlig for studentrekruttering

Variant

Mål: At dere ikke skal lære noe nytt

Variant

Variant

En kjapp gjennomgang av en Reactkomponent

Variant

```tsx
interface MyComponentProps {
  title: string;
  subtitle: string;
}

const MyComponent = ({ title, subtitle }: MyComponentProps) => {
  return (
    <>
      <h1>{title}</h1>
      <p>{subtitle}</p>
    </>
  );
};

export default MyComponent;
```

```tsx
import { useState } from "react";

...

const MyComponent = ({ title, initialSubtitle }: MyComponentProps) => {
  const [subtitle, setSubtitle] = useState(initialSubtitle);

  return (
    <>
      <h1>{title}</h1>
      <p>{subtitle}</p>
    </>
  );
};
```

Variant

```tsx
import { useState } from "react";

...

const MyComponent = ({ title, initialSubtitle }: MyComponentProps) => {
  const [subtitle, setSubtitle] = useState(initialSubtitle);

  function changeSubtitle() {
    setSubtitle("New subtitle!");
  }

  return (
    <>
      <h1>{title}</h1>
      <p>{subtitle}</p>
      <button onClick={changeSubtitle}>Change subtitle</button>
    </>
  );
};
```

Del 1

Hva er egentlig JSX?

Variant

Variant

Variant

Reference

`createElement(type, props, ...children)`

Call `createElement` to create a React element with the given `type`, `props`, and `children`.

Variant

```
const Greeting = ({ name }) => {
  return <h1 className='greeting'>Hello</>
}
```

Variant

```jsx
const Greeting = ({ name }) => {
  return <h1 className='greeting'>Hello</>
}
```

```jsx
import { createElement } from 'react';

const Greeting = ({ name }) => {
  return createElement(
    'h1', { className: 'greeting' }, 'Hello'
  );
}
```

Variant

Variant

2013

Variant

Oppgave 1 - Tilbake til steinalderen

Wifi: varianthuset-guest

PW: thom10guest

Før vi kan starte:

1. Gå til `https://github.com/itzjacki/kurs-online-react`, klon repoet
2. Få det til å kjøre opp lokalt ved å følge Readme

Når vi er oppe og kjører:

1. Lag en ny fil, LogoHeader.ts, ved siden av LogoHeader.tsx
2. Skriv kode i .ts-fila så den gjør akkurat det samme som LogoHeader.tsx
3. I App.tsx, bytt fra å importere .tsx-fila til å importere .ts-fila. Endrer noe seg?
4. Gjør det samme med UsefulLinks.tsx

Variant

Hvorfor diamantene?

```
return (
  <>
    <p>Ting 1</p>
    <p>Ting 2</p>
    <p>Ting 3</p>
  </>
);
```

Variant

Hvorfor diamantene?

```
return (
  <p>Ting 1</p>
  <p>Ting 2</p>
  <p>Ting 3</p>
);
```

Variant

Hvorfor diamantene?

```
return (
  React.createElement('p', null, 'Ting 1'),
  React.createElement('p', null, 'Ting 2'),
  React.createElement('p', null, 'Ting 3')
);
```

Variant

Hvorfor diamantene?

```
return React.createElement(
  React.fragment,
  null,
  React.createElement('p', null, 'Ting 1'),
  React.createElement('p', null, 'Ting 2'),
  React.createElement('p', null, 'Ting 3')
);
```

Variant

Del 2

State

Variant

State = dynamisk data

Variant

```jsx
const CounterFunction = () => {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
};
```

Variant

```
class CounterClass extends Component {
  constructor(props: {}) {
    super(props);
    this.state = { count: 0 };
  }

  render() {
    return (
      <div>
        <p>Count: {this.state.count}</p>
        <button onClick={() => this.setState({ count: this.state.count + 1 })}>
          Increment
        </button>
      </div>
    );
  }
}
```

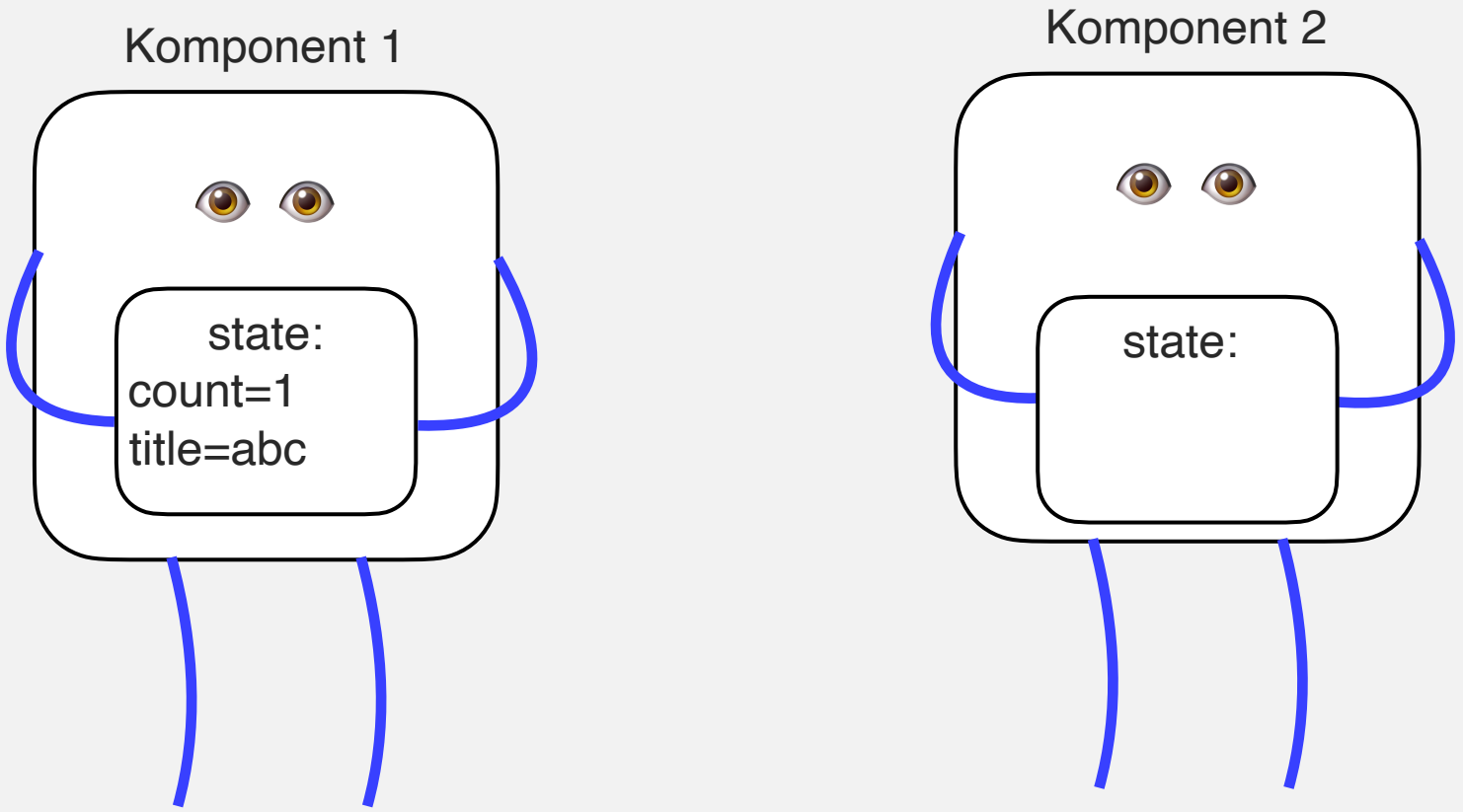
Variant

Kjapt sidespor om hooks

Variant

useState()
useEffect()
useContext()

Variant



Variant

Oppgave 2: Statefulness

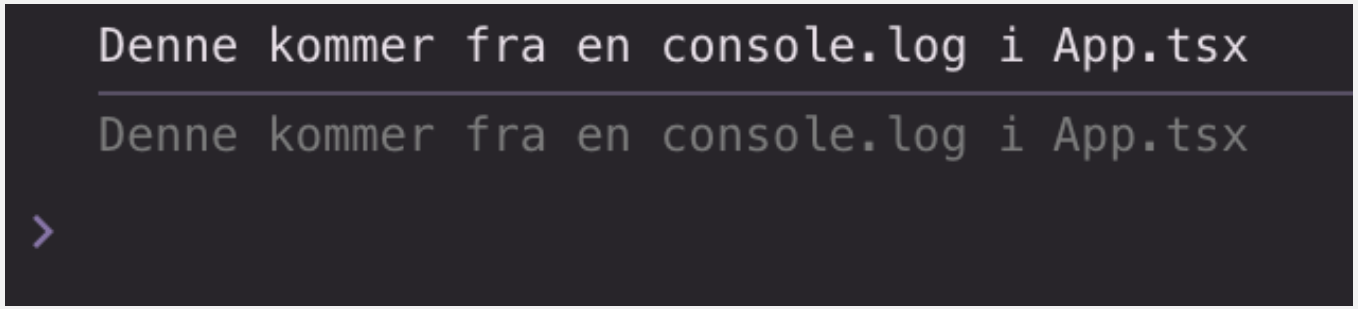
1. Skriv om FeedbackForm.tsx så input-feltene blir **controlled**, altså at de oppbevarer verdien sin i React state. Bruk `value` og `onChange`-attributtene på input-elementene
2. Skriv om `handleSubmit`-funksjonen så den bruker data fra state i stedet å hente det fra en formEvent
3. Lag en character counter til feedback-feltet, det er mye lettere nå som verdien finnes i state

Variant

Del 3

Rendring

Variant


Denne kommer fra en console.log i App.tsx
Denne kommer fra en console.log i App.tsx
>

Variant

React strict mode

Variant

React strict mode

```
createRoot(document.getElementById("root")!).render(
  <StrictMode>
    <App />
  </StrictMode>
);
```

Variant

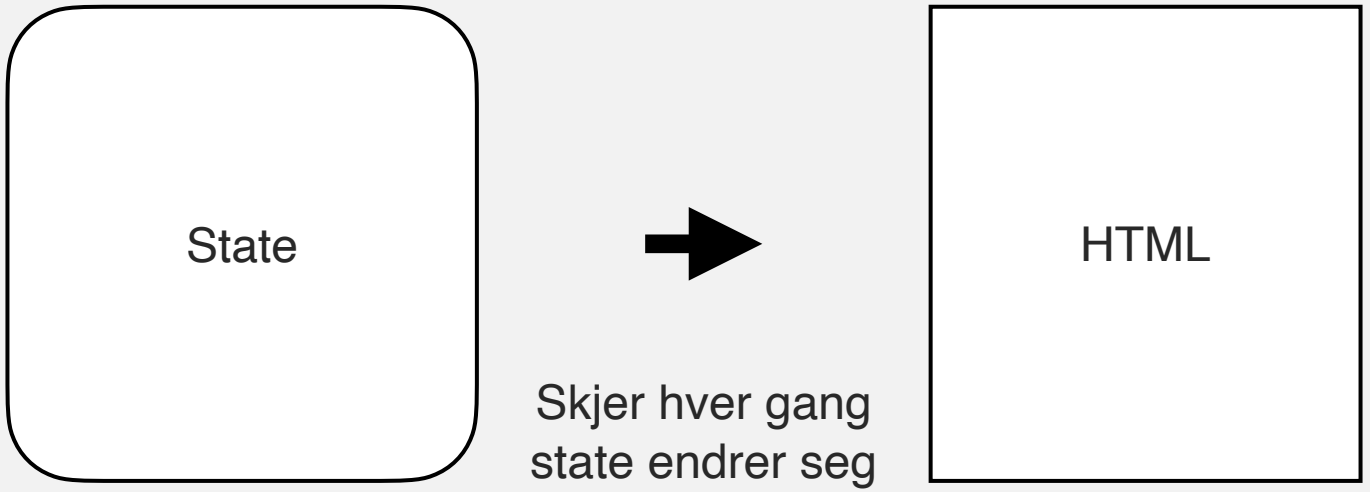
- Rendrer komponenten to ganger
- Kjører useEffects i komponenten to ganger

Variant

Rendring i React

Variant

Variant



Variant

Men dette er treigt!

Variant

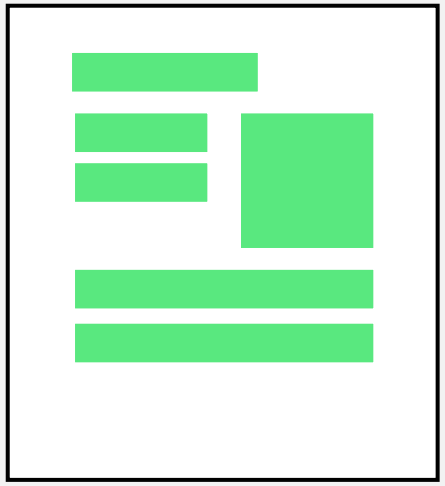
1. Bruk av **Virtual DOM** for å gjøre rendering mye kjappere
2. React rendrer bare komponentene som **må re-rendres**

Variant

Virtual DOM

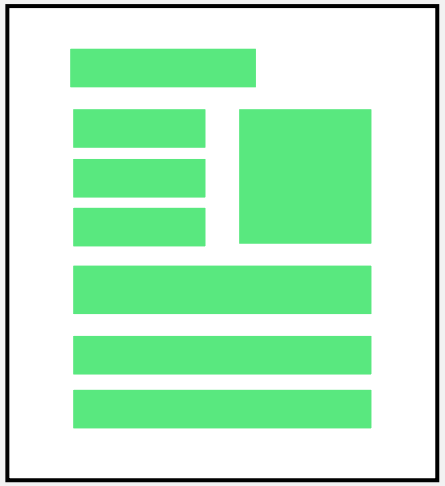
Variant

Gammel state



Virtual DOM 1

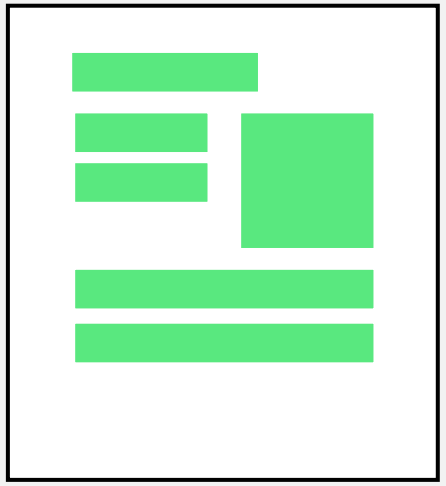
Ny state



Virtual DOM 2

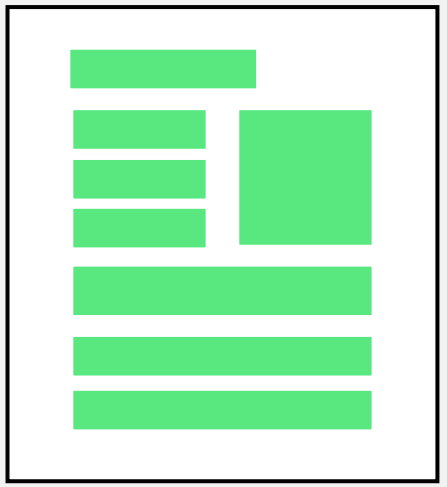
Variant

Gammel state



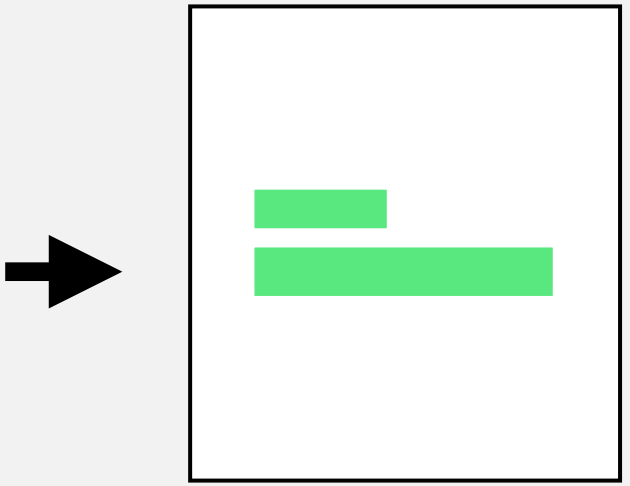
Virtual DOM 1

Ny state



Virtual DOM 2

Endringer som må gjøres med den ekte DOMen

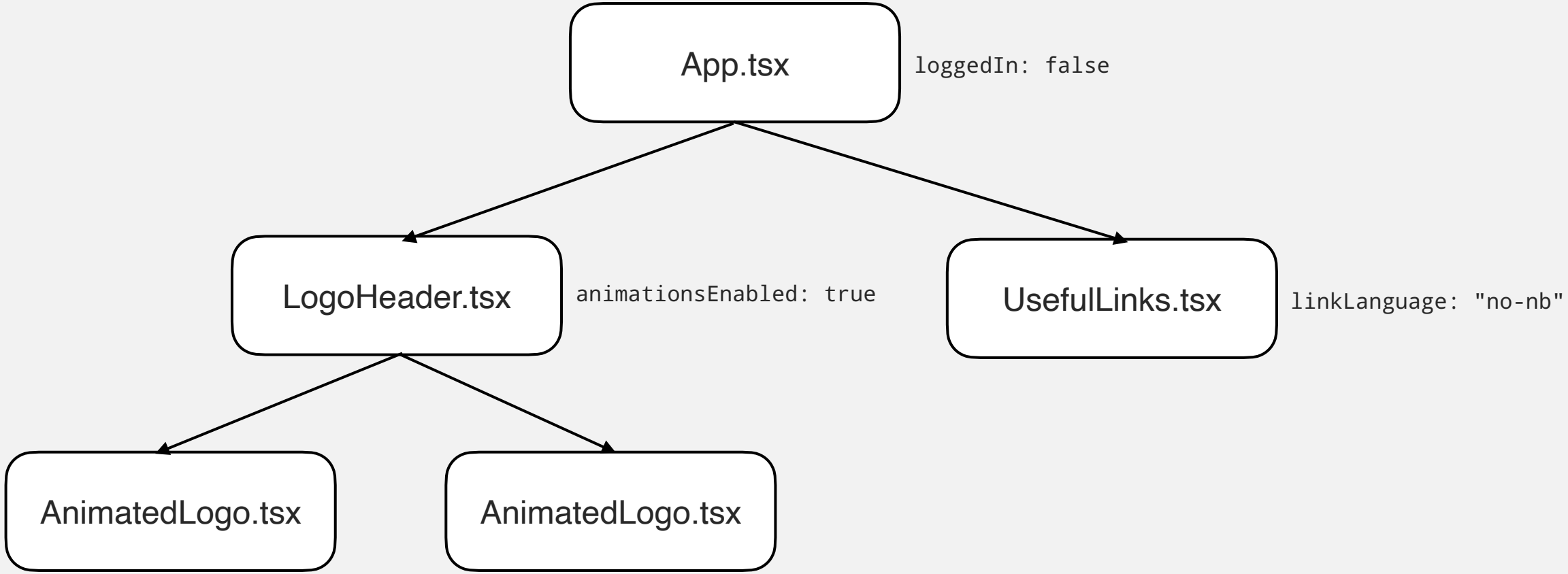


Diff

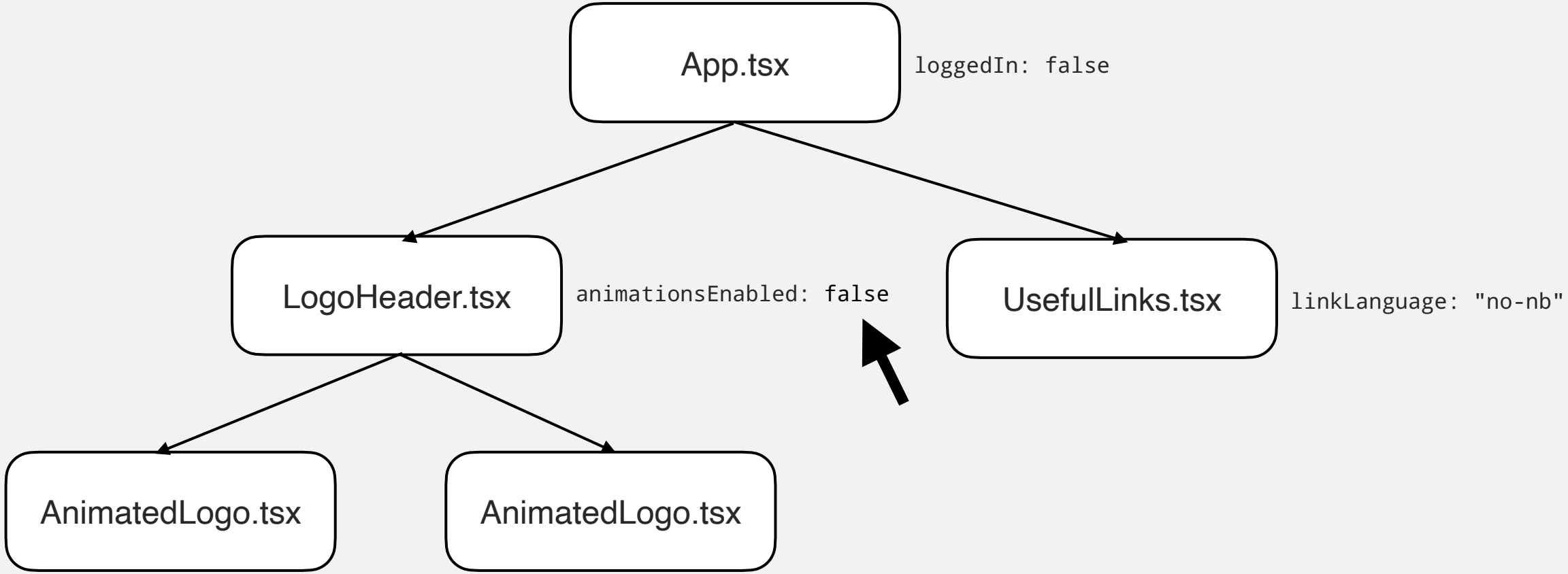
Variant

Begrensa re-rendring

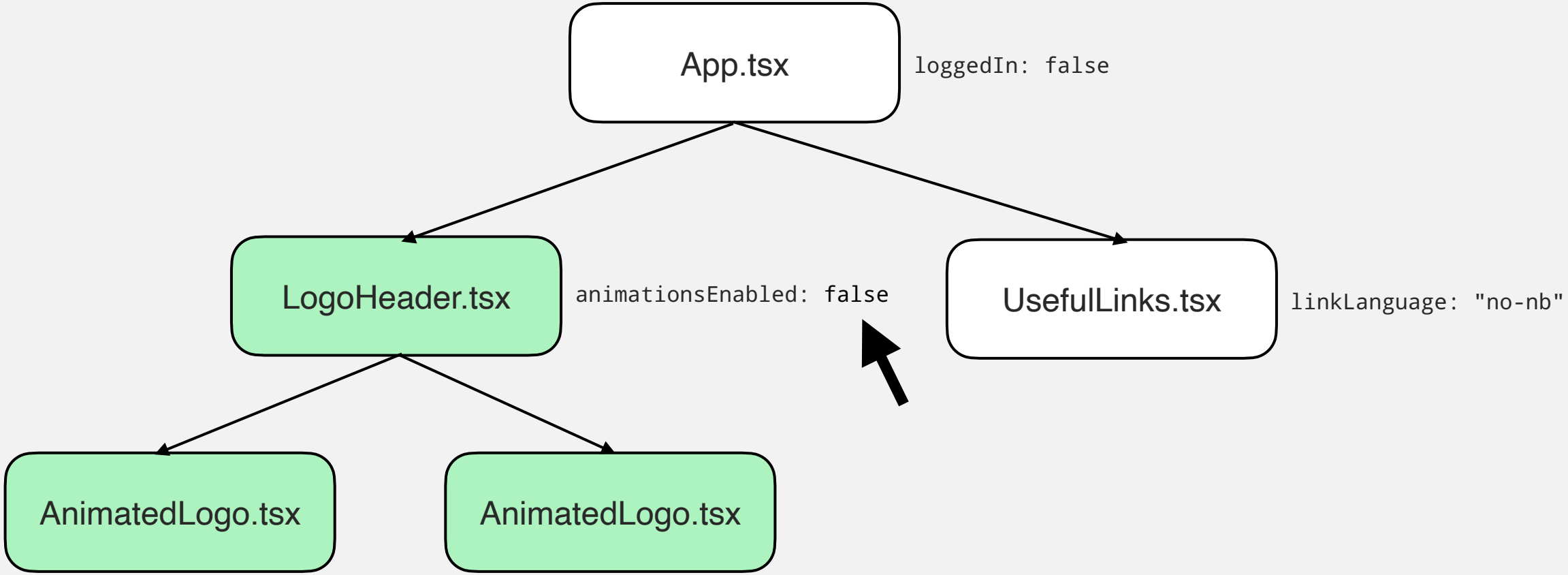
Variant


App.tsx
loggedIn: false
LogoHeader.tsx
animationsEnabled: true
UsefulLinks.tsx
linkLanguage: "no-nb"
AnimatedLogo.tsx
AnimatedLogo.tsx

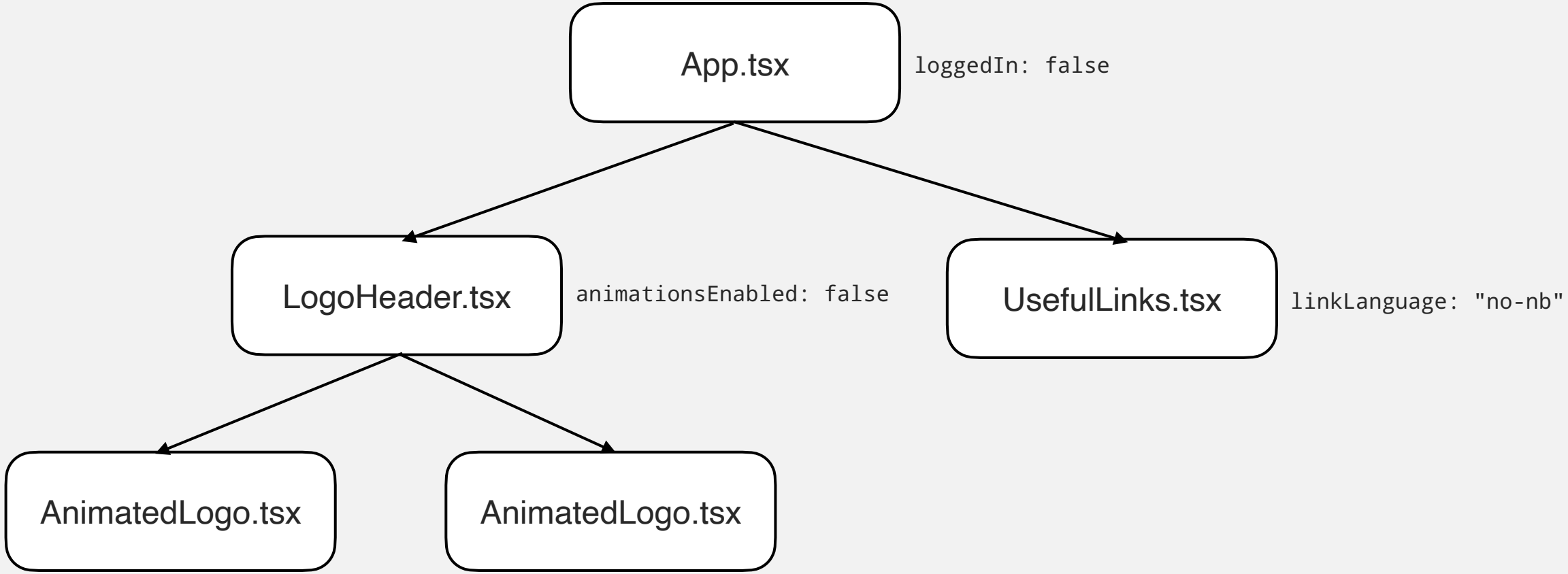
Variant


App.tsx
loggedIn: false
LogoHeader.tsx
animationsEnabled: false
UsefulLinks.tsx
linkLanguage: "no-nb"
AnimatedLogo.tsx
AnimatedLogo.tsx

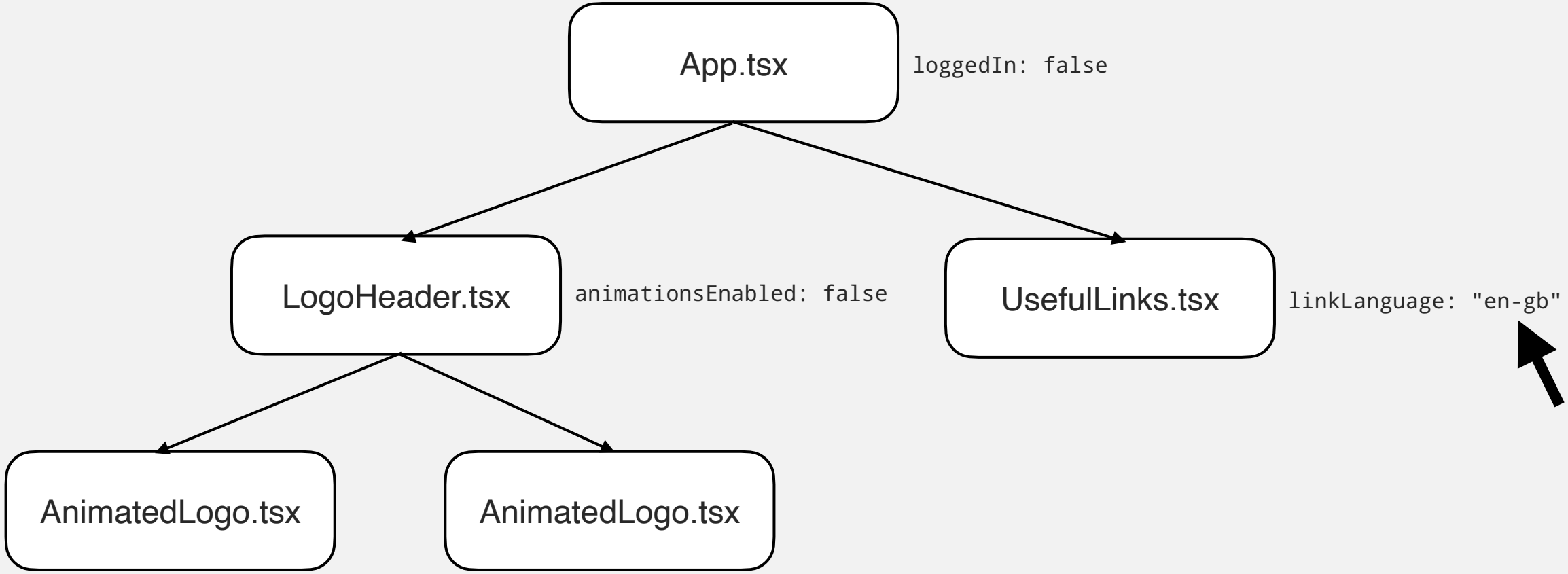
Variant


App.tsx
loggedIn: false
LogoHeader.tsx
animationsEnabled: false
UsefulLinks.tsx
linkLanguage: "no-nb"
AnimatedLogo.tsx
AnimatedLogo.tsx

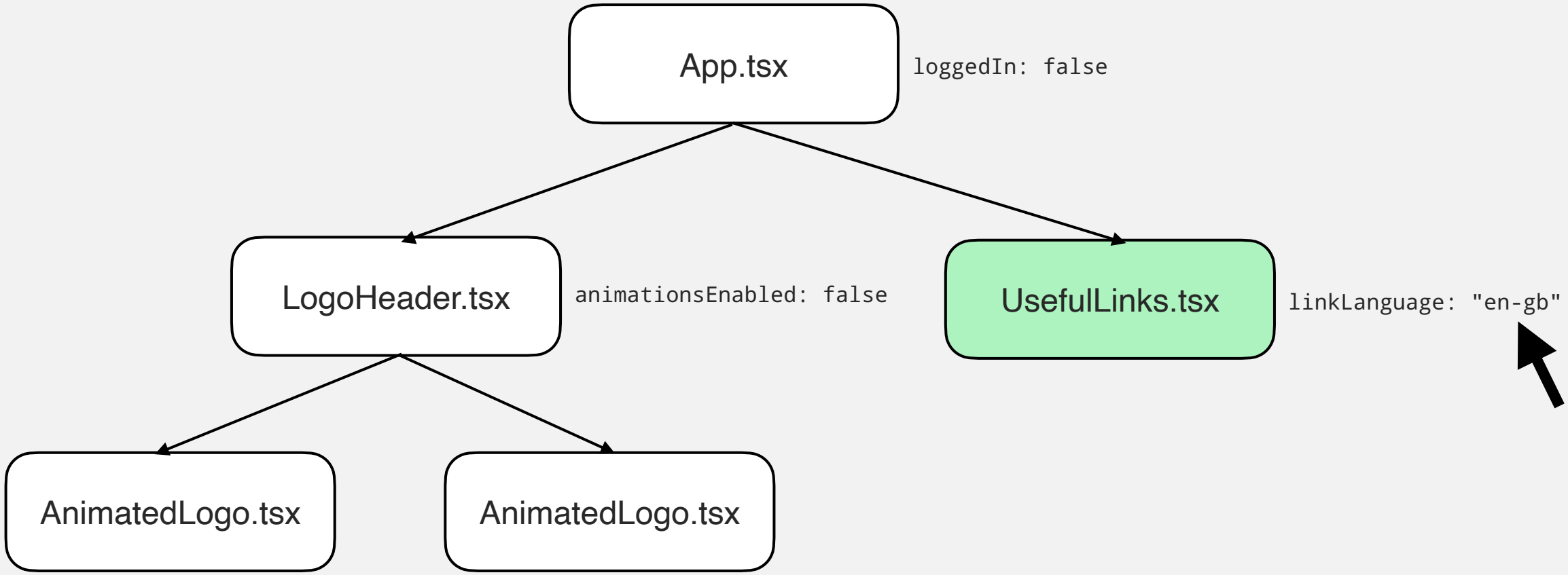
Variant


App.tsx
loggedIn: false
LogoHeader.tsx
animationsEnabled: false
UsefulLinks.tsx
linkLanguage: "no-nb"
AnimatedLogo.tsx
AnimatedLogo.tsx

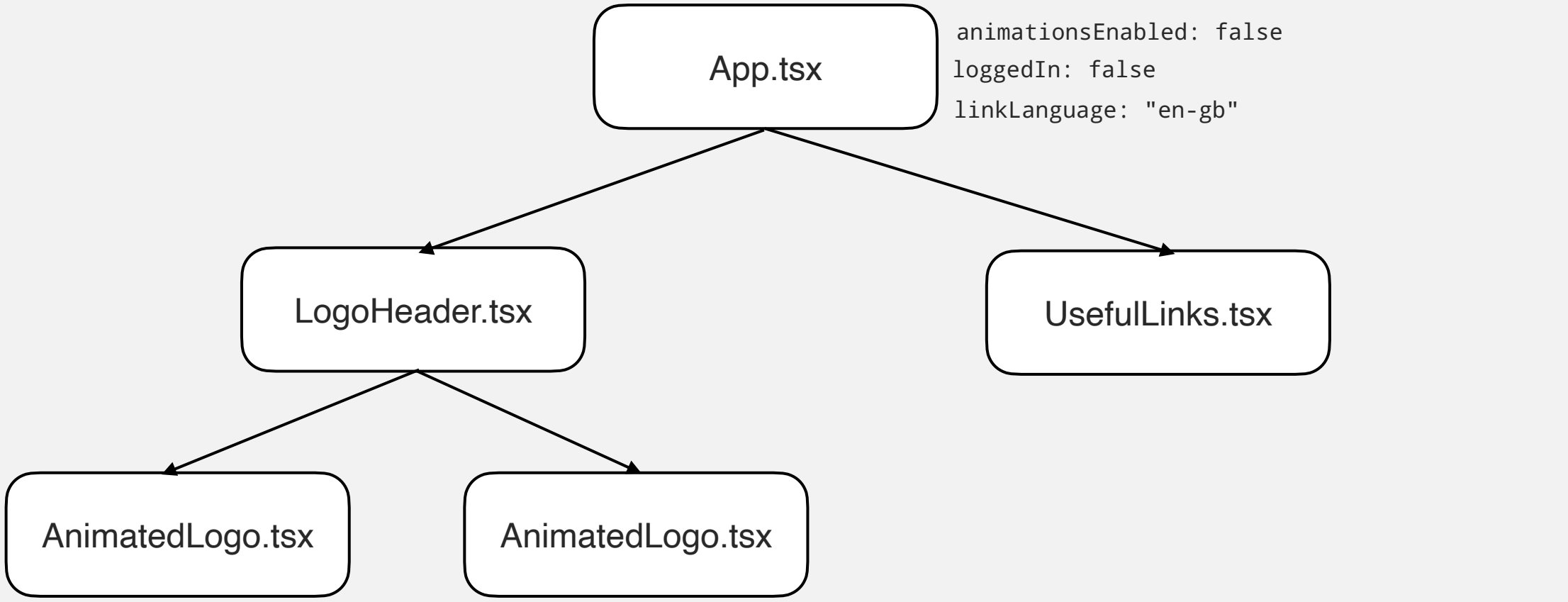
Variant


App.tsx
loggedIn: false
LogoHeader.tsx
animationsEnabled: false
UsefulLinks.tsx
linkLanguage: "en-gb"
AnimatedLogo.tsx
AnimatedLogo.tsx

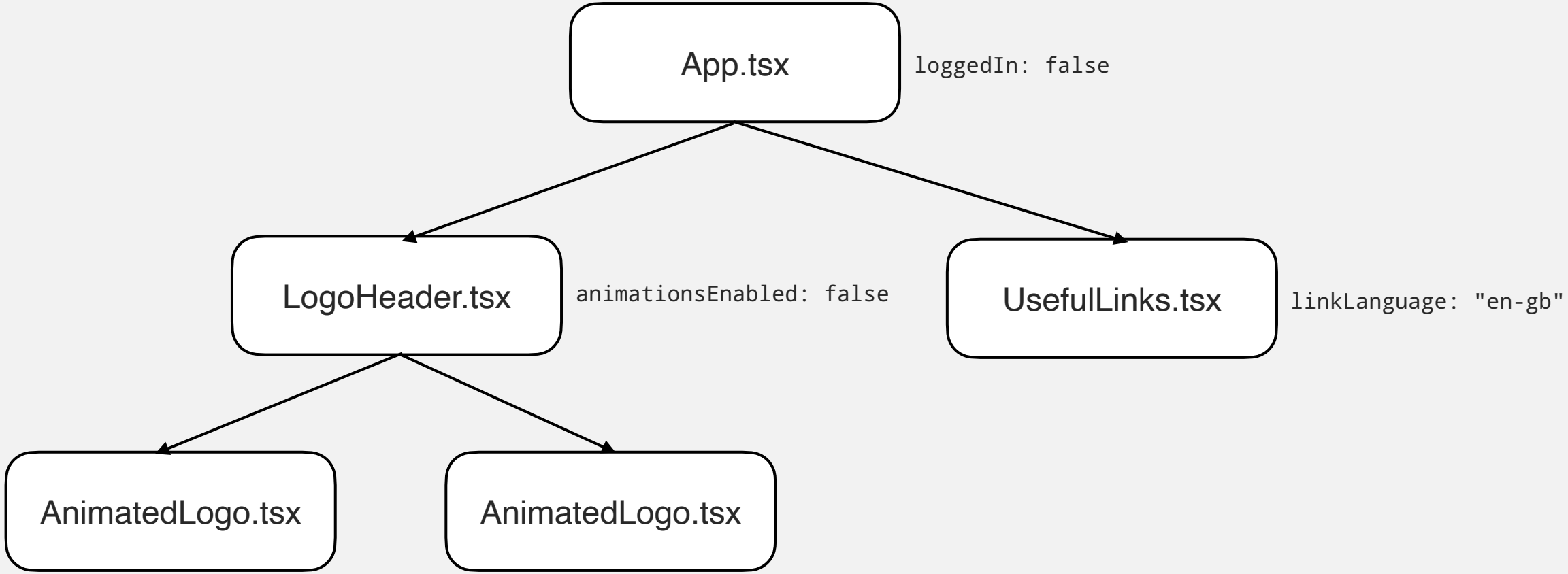
Variant


App.tsx
loggedIn: false
LogoHeader.tsx
animationsEnabled: false
UsefulLinks.tsx
linkLanguage: "en-gb"
AnimatedLogo.tsx
AnimatedLogo.tsx

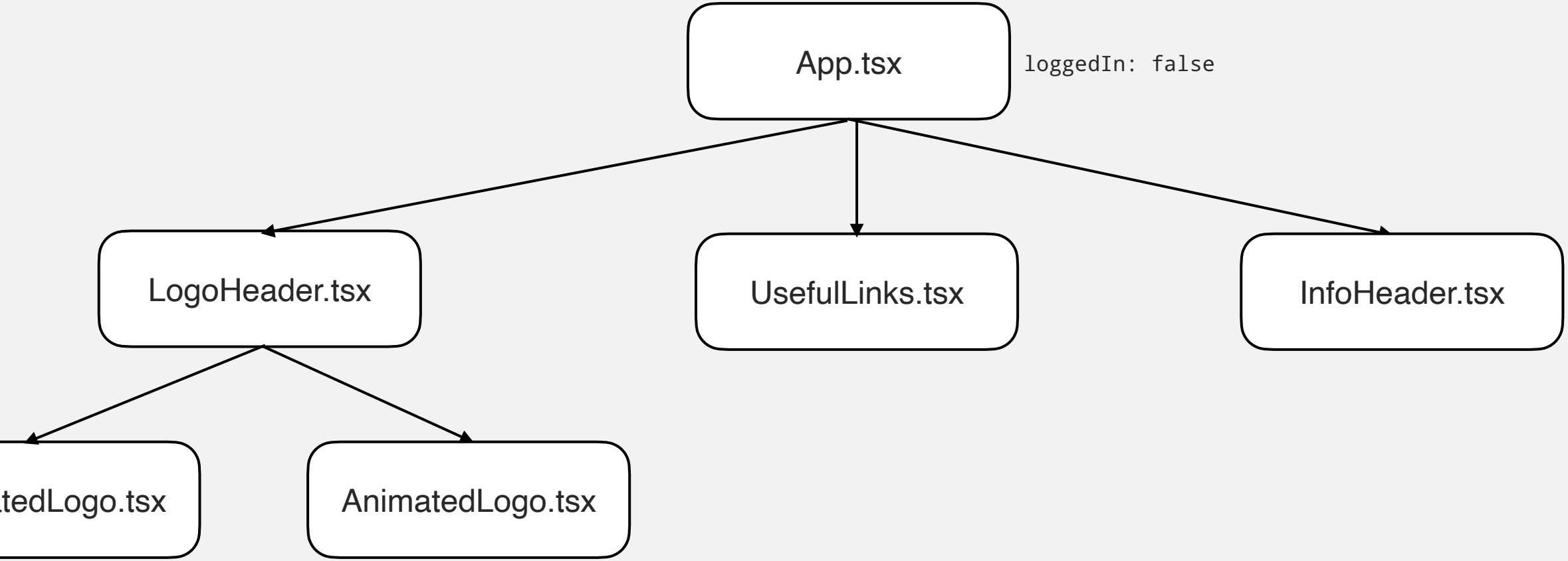
Variant


App.tsx
animationsEnabled: false
loggedIn: false
linkLanguage: "en-gb"
LogoHeader.tsx
UsefulLinks.tsx
AnimatedLogo.tsx
AnimatedLogo.tsx

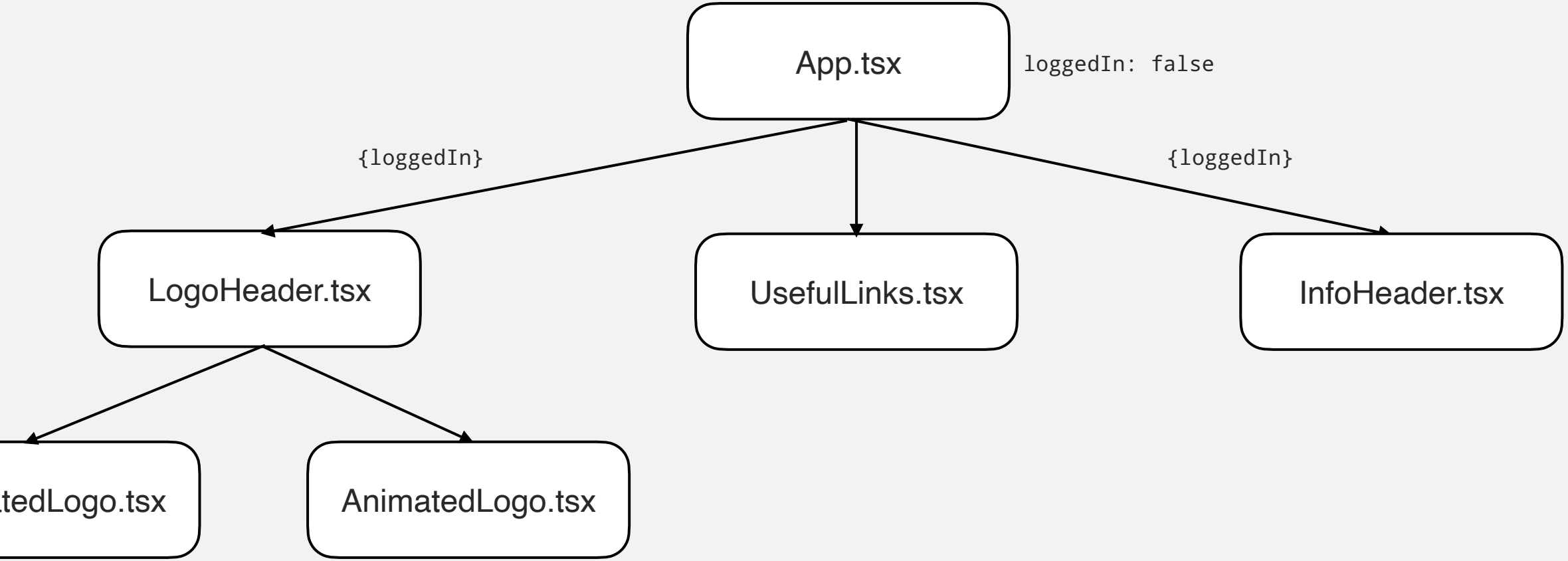
Variant


App.tsx
loggedIn: false
LogoHeader.tsx
animationsEnabled: false
UsefulLinks.tsx
linkLanguage: "en-gb"
AnimatedLogo.tsx
AnimatedLogo.tsx

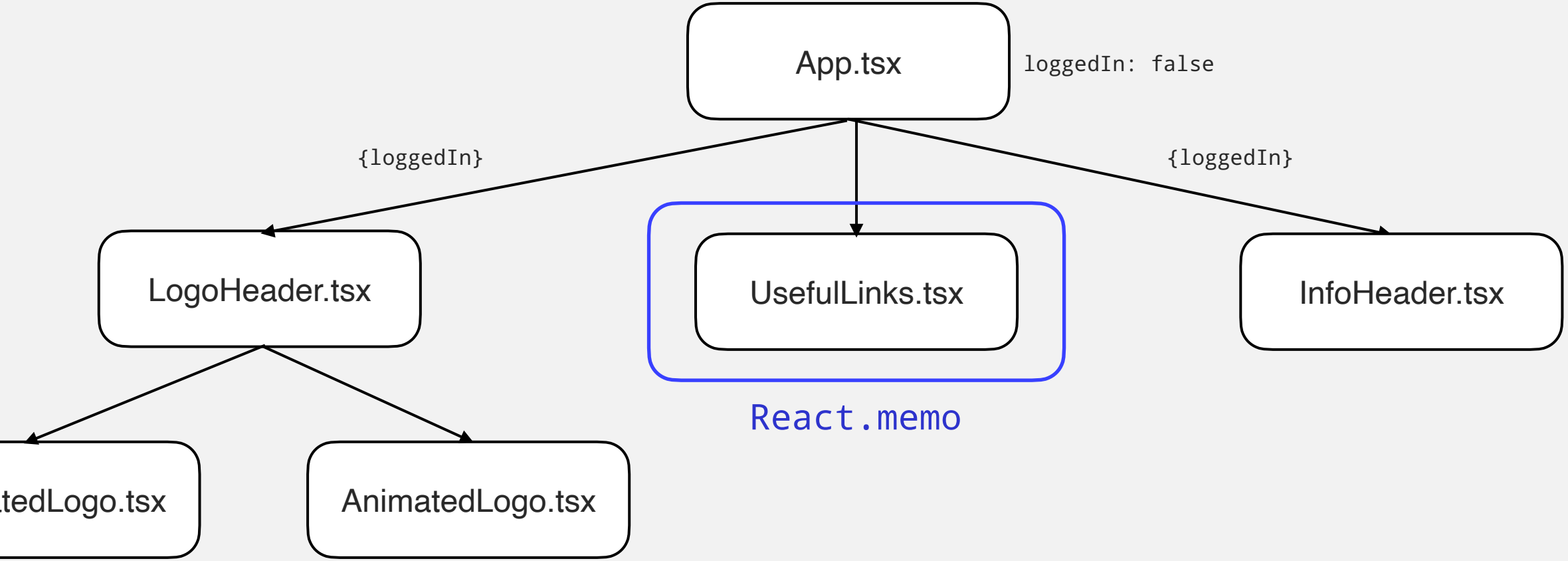
Variant


App.tsx
loggedIn: false
LogoHeader.tsx
UsefulLinks.tsx
InfoHeader.tsx
tedLogo.tsx
AnimatedLogo.tsx

Variant


App.tsx
loggedIn: false
{loggedIn}
{loggedIn}
LogoHeader.tsx
UsefulLinks.tsx
InfoHeader.tsx
tedLogo.tsx
AnimatedLogo.tsx

Variant


App.tsx
loggedIn: false
{loggedIn}
{loggedIn}
LogoHeader.tsx
UsefulLinks.tsx
InfoHeader.tsx
React.memo
tedLogo.tsx
AnimatedLogo.tsx

Variant

```
import { memo } from "react";

...

export default memo(UsefulLinks);
```

Variant

Oppgave 3: Unødvendige renders

1. Eksperimenter litt med applikasjonen, når re-rendres de forskjellige komponentene? (Se Readme for mer detaljer)
2. Bruk teknikkene vi så på til å optimalisere applikasjonen så den ikke rendrer flere komponenter enn nødvendig

Variant

Del 4

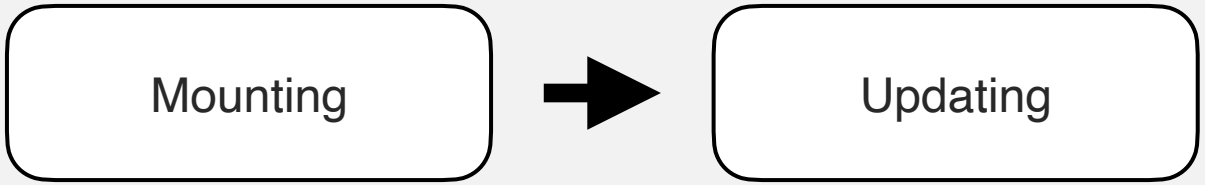
The circle of life

Variant

Variant

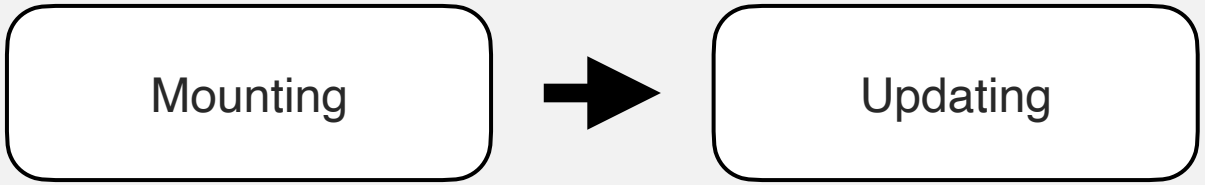
Komponenten hentes inn
i DOMen for første gang

Variant



Komponenten hentes inn
i DOMen for første gang

Variant



Komponenten hentes inn i DOMen for første gang

Komponenten re-rendres

Variant

Komponenten hentes inn i DOMen for første gang

Komponenten re-rendres

Variant

Komponenten hentes inn i DOMen for første gang

Komponenten re-rendres

Komponenten fjernes fra DOMen

Variant

```
class TheCircleOfLife extends Component {
  constructor(props: {}) {
    super(props);
  }

  componentDidMount() {
    console.log("✅ componentDidMount: Component has mounted");
  }

  componentDidUpdate() {
    console.log("♻️ componentDidUpdate: Component updated");
  }

  componentWillUnmount() {
    console.log("❌ componentWillUnmount: Component is being removed");
  }

  render() {
    return (
      <div>Howdy</div>
    );
  }
}
```

Komponenten hentes inn i DOMen for første gang

Komponenten re-rendres

Komponenten fjernes fra DOMen

Variant

Komponenten hentes inn i DOMen for første gang

`componentDidMount()`

Komponenten re-rendres

Komponenten fjernes fra DOMen

Variant

Komponenten hentes inn i DOMen for første gang

`componentDidMount()`

Komponenten re-rendres

`componentDidUpdate()`

Komponenten fjernes fra DOMen

Variant

Komponenten hentes inn i DOMen for første gang

`componentDidMount()`

Komponenten re-rendres

`componentDidUpdate()`

Komponenten fjernes fra DOMen

`componentWillUnmount()`

Variant

```
useEffect(() => {
  // Gjør disse tingene
}, [/* Hver gang disse endrer seg */]);
```

Variant

```
useEffect(() => {
  // Hvis deps array er tom gjøres dette når komponenten mountes
}, []);
```

Variant

```
useEffect(() => {
  // Hvis deps array er tom gjøres dette når komponenten mountes
  return () => {
    // Returnerer man en funksjon gjøres den når komponenten unmountes
  }
}, []);
```

Variant

```
useEffect(() => {
  // Dropper man deps array gjøres dette hver gang komponenten rendres
});
```

Variant

```
class TheCircleOfLife extends Component {
  constructor(props: {}) {
    super(props);
  }

  componentDidMount() {
    console.log("✅ componentDidMount: Component has mounted");
  }

  componentDidUpdate() {
    console.log("♻️ componentDidUpdate: Component updated");
  }

  componentWillUnmount() {
    console.log("❌ componentWillUnmount: Component is being removed");
  }

  render() {
    return (
      <div>Howdy</div>
    );
  }
}
```

```jsx
const TheCircleOfLife = () => {
  useEffect(() => {
    console.log("✅ useEffect (empty deps array): Component has mounted");

    return () => {
      console.log("❌ Cleanup: Component is being unmounted");
    };
  }, []);

  useEffect(() => {
    console.log("♻️ useEffect (no deps array): Component rendered");
  },);

  return (
    <div>Howdy</div>
  );
};
```

Oppgave 4: The Circle of Life

1. Eksperimenter med useEffect i CounterButton.
    1. Skriv forskjellige ting til loggen når komponenten mounter, unmounter og re-rendres
2. Følg instruksene i Readme for å fremprovosere uendelige re-renders, som er en relativt vanlig bug når man kombinerer useEffects og useState uten å tenke på hva som skjer i bakgrunnen. Hva er grunnen til at dette oppstår?

Variant

Takk for at dere kom hit! 🍾