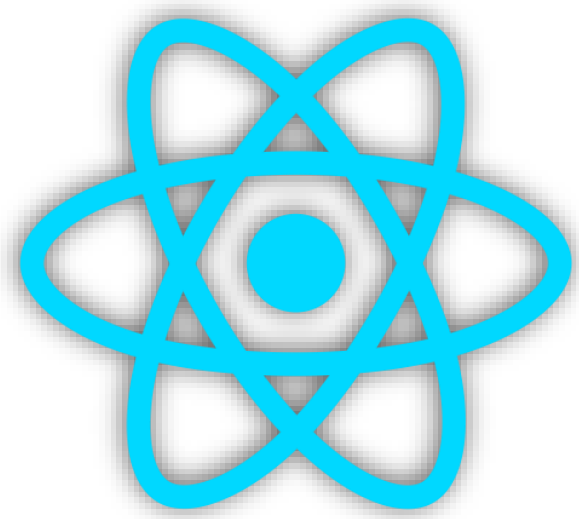
FRONT-END Academy

by Present Connection



Multilanguage

- **react-intl** - galimi vertimai tiek statiniam tiek dinaminiam turiniui, kuris keičiasi priklausomai nuo state
- **npm i -S react react-intl**

Panaudojimas

```
<IntlProvider messages={translations[locale]} locale={locale}>
  <Your_APP></Your_APP>
</IntlProvider>
```

en.json

```
{
    "header.tournaments": "Tournaments",
    "header.login": "Login"
}
```

lt.json

```
{
    "header.tournaments": "Turnyrai",
    "header.login": "Prisijungti"
}
```

Panaudojimas

```
import { FormattedMessage } from 'react-intl';
```

Sugeneruoja išverstą tekstą

```
<FormattedMessage id={YOUR_TRANSLATION_ID}></FormattedMessage>
```

Panaudojimas

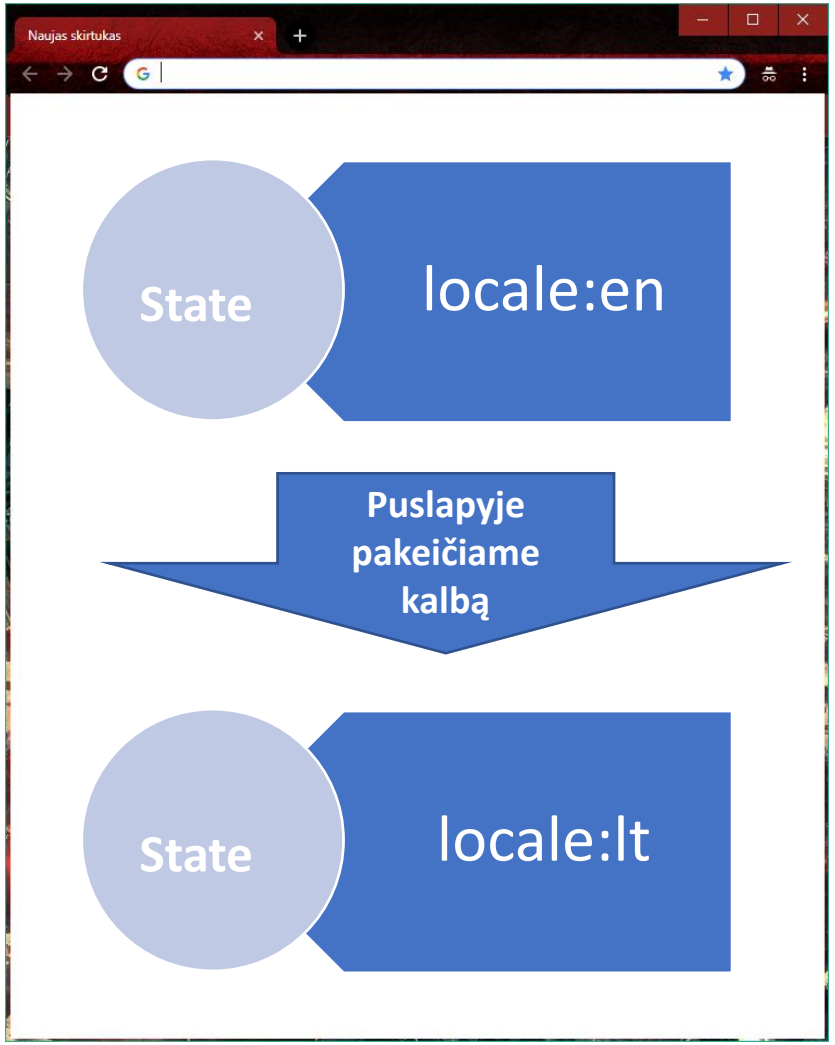
```
import { useIntl } from 'react-intl';
```

```
const intl = useIntl();
```

```
YOUR_PROP={intl.formatMessage({ id: TRANSLATION_ID })}
```

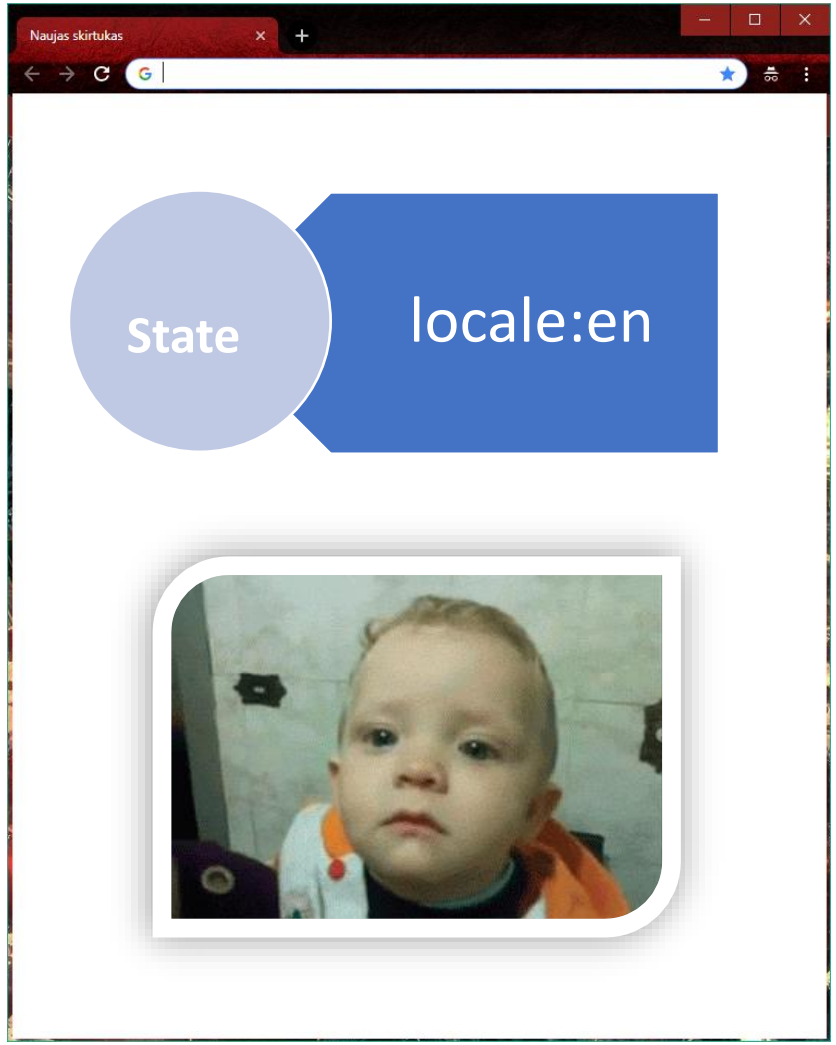
Inject'inam intl kai mums reikia teksto, o ne sugeneruoto elemento (span). Kaip pvz: paduoti label (kaip props) komponentui kuris tikisi string'o, o ne kito komponento

Įprasta state

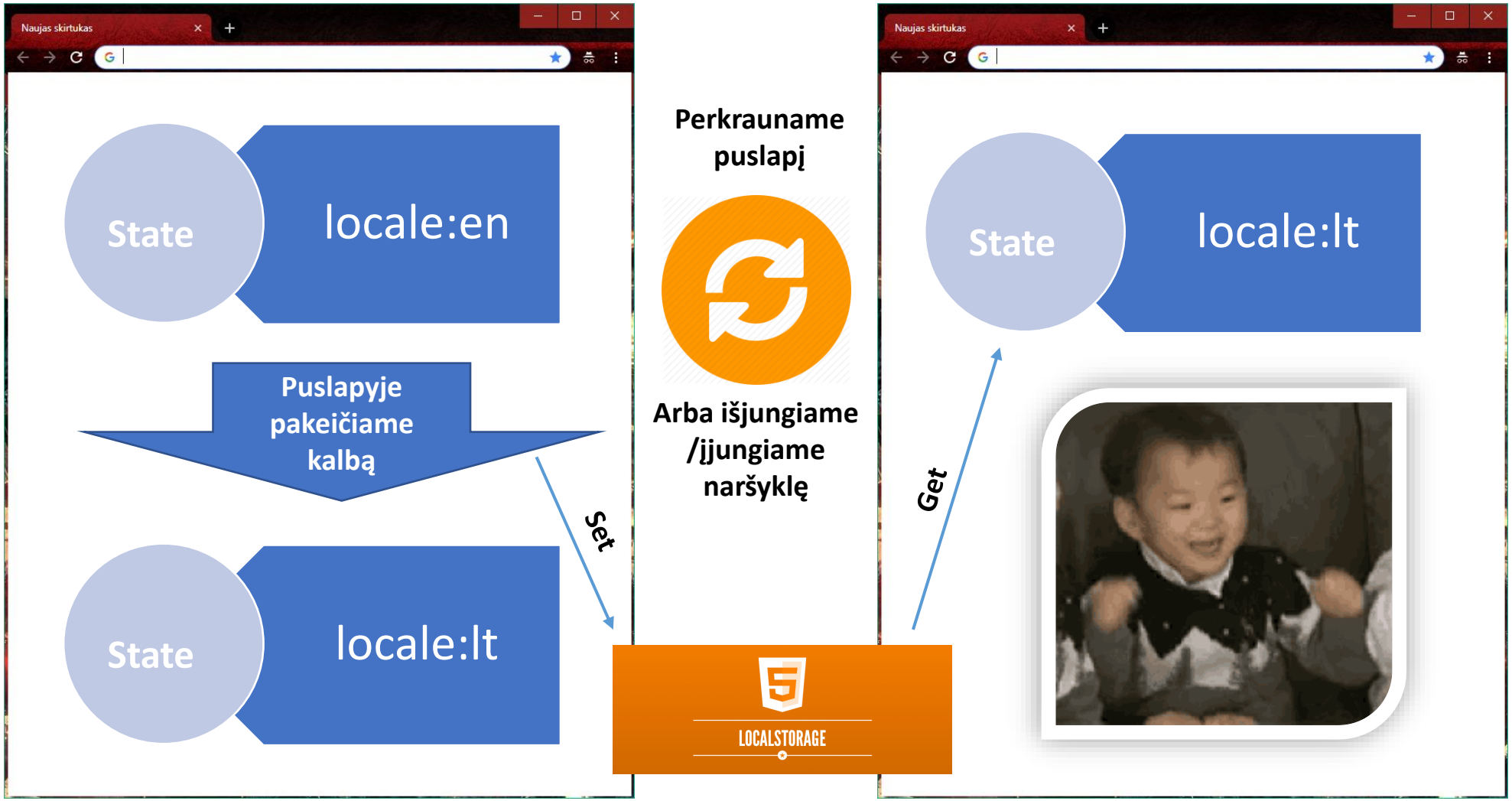


Perkrauname puslapį

Arba išjungiame /įjungiame naršyklę



Išliekanti state



Error handling

- **JavaScript** error neturi nubreak'inti viso app
- **Error boundary,** komponentas kuris pagauna klaidas esančias komponentuose kuriuos jis „apgaubia“, panašiai kaip **try-catch**.
- Pagauna klaidas esančias renderinimo metu, lifecycle metoduose ir kontsruktoriuje.
- Puiki vieta (single point) loginti visos aplikacijos klaidas ir jas handlinti

Error handling

- Error boundary nepagauna klaidų esančių:
  - Event handler‘iuose
  - Async kode
  - Server side rendering
  - Jeigu klaida įvyksta pačiam error boundary komponente

Error handling: implementacija

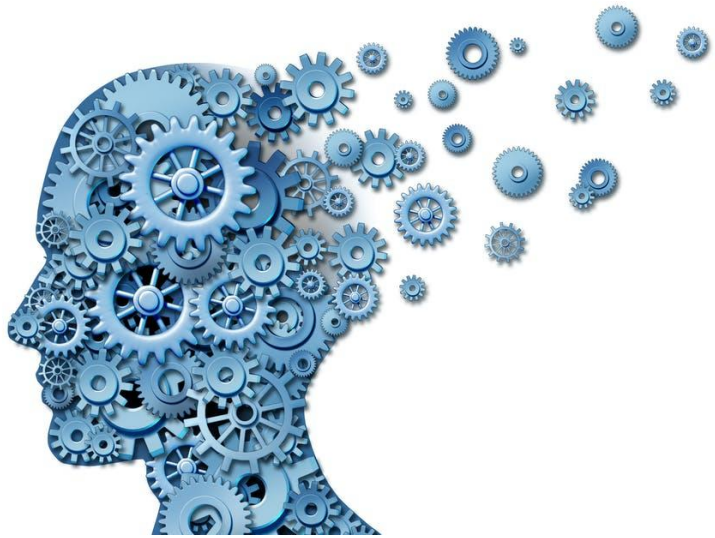
```
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }

  static getDerivedStateFromError(error) {
    // Update state so the next render will show the fallback UI.
    return { hasError: true };
  }

  componentDidCatch(error, errorInfo) {
    // You can also log the error to an error reporting service
    logErrorToMyService(error, errorInfo);
  }

  render() {
    if (this.state.hasError) {
      // You can render any custom fallback UI
      return <h1>Something went wrong.</h1>;
    }

    return this.props.children;
  }
}
```

```
<ErrorBoundary>
  <MyWidget />
</ErrorBoundary>
```

useMemo

```
const memoizedValue = useMemo(() => computeExpensiveValue(a, b), [a, b]);
```

- Grąžina „memoized“ reikšmę
- Perskaičiuoja reikšmę tik tada, jeigu paduodame parametrą kuriam pasikeitus reiktų perskaičiavimo, taip pat kaip useEffect
- Galima naudoti tiek funkcijoms tiek ir komponentams prisiminti



ref

- Suteikią prieigą prie DOM elementų
- Kai prireikia modifikuoti/gauti DOM elementų info išeinant iš standartinio flow.
- Avoid using refs for anything that can be done declaratively.
- Kada naudoti
  - Valdant elemento focus, teksto pasirinkimą
  - Trigger‘inant animacijas
  - Integracijai su third-party DOM bibiliotekomis

```
const onButtonClick = () => {
  // `current` points to the mounted text input element
  inputEl.current.focus();
};
```

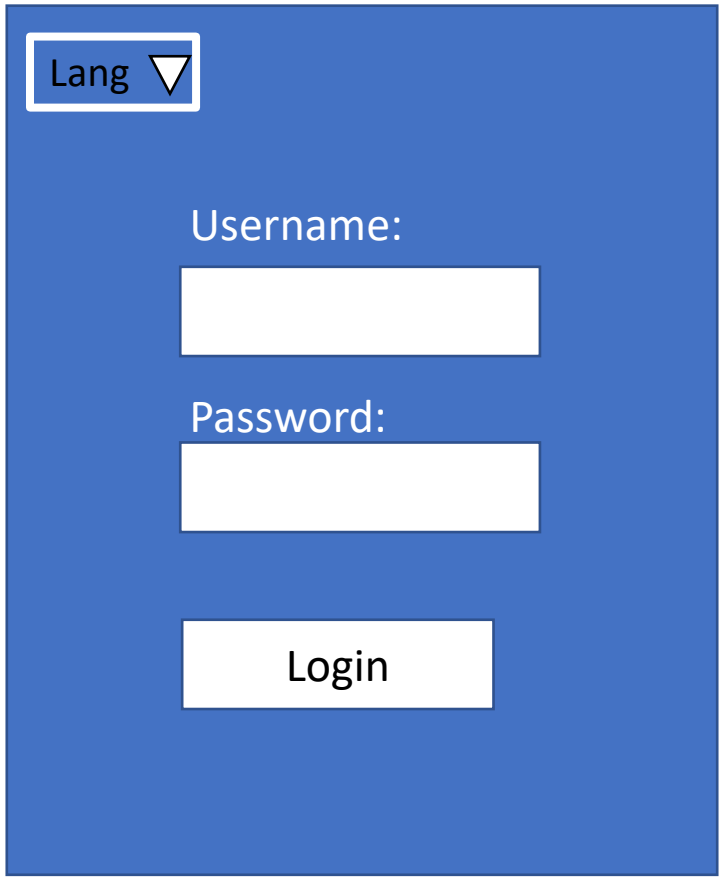
```
const refContainer = useRef(initialValue);
```

DEMO

Workshop

1. Turi būti multilanguage (3 kalbos)
2. Perkrovus arba išjungus naršyklę pasirinkta kalba išlieka, username taip pat
3 Paspaudus login patenkame į home
4. Jeigu vartotojas neprisijungęs, eidamas į home yra nukreipiamas į login
5. Jeigu vartotojas prisijungęs, eidamas į login yra nukreipiamas į home

/login



/home

Namų darbai

- Toliau tęsiam darbą su projektais (atsižvelgdami į pastabas)