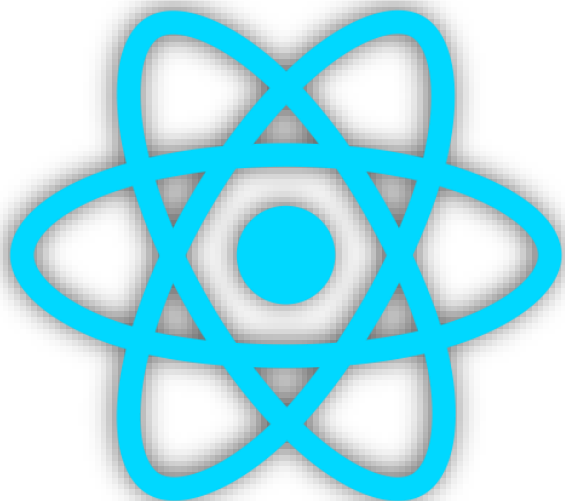


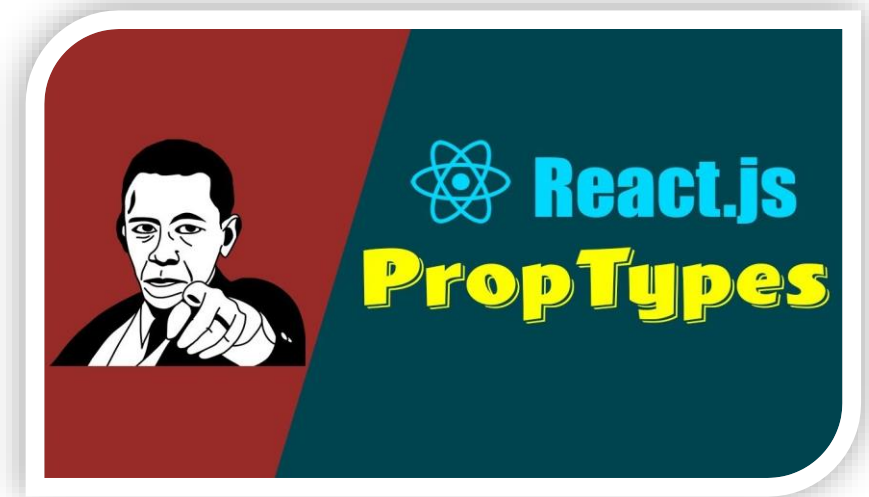
FRONT-END *Academy*

by Present Connection



Prop types

- ◆ Patikrina komponento parametrų tipus
- ◆ Padeda išvengti klaidų, kurios gali atsirasti aplikacijai augant
- ◆ Tipai yra tikrinami tik development mode
- ◆ Pranešimai apie blogus tipus yra parodomi konsolėje



Prop types

```
</tr>  
</thead>  
<tbody>  
  {props.data.map(item => (  
    <tr>
```

'data' is missing in props validation eslint(react/prop-types)
any

```
Persons.propTypes = {  
  data: PropTypes.array,  
};
```

```
<Persons data="Test" />
```

✖ ▶ Warning: Failed prop type: Invalid prop `data` of type `string` supplied to `Persons`, expected `array`.

Prop types

<https://reactjs.org/docs/typechecking-with-proptypes.html>

```
// You can declare that a prop is a specific JS type. By default, these
// are all optional.
optionalArray: PropTypes.array,
optionalBool: PropTypes.bool,
optionalFunc: PropTypes.func,
optionalNumber: PropTypes.number,
optionalObject: PropTypes.object,
optionalString: PropTypes.string,
optionalSymbol: PropTypes.symbol,
```

```
// A React element.
optionalElement: PropTypes.element,
```

```
// An object that could be one of many types
optionalUnion: PropTypes.oneOfType([
  PropTypes.string,
  PropTypes.number,
  PropTypes.instanceOf(Message)
]),
```

```
// An array of a certain type
optionalArrayOf: PropTypes.arrayOf(PropTypes.number),
```

Prop types

<https://reactjs.org/docs/typechecking-with-proptypes.html>

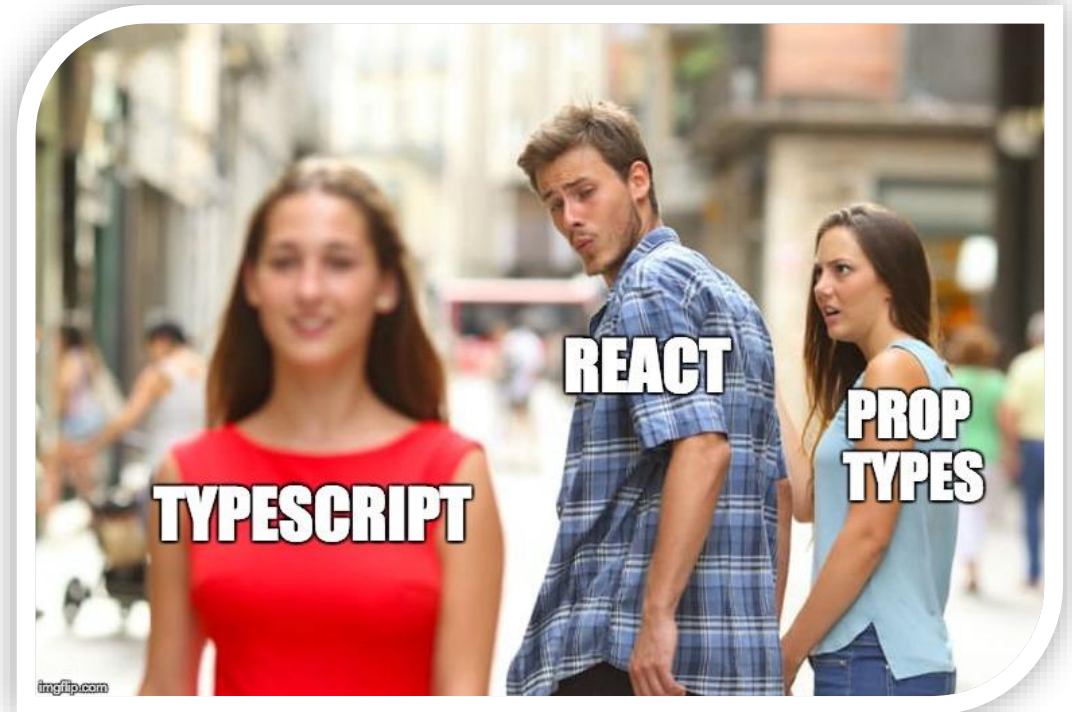
```
// An object taking on a particular shape
optionalObjectWithShape: PropTypes.shape({
  color: PropTypes.string,
  fontSize: PropTypes.number
}),
```

```
// A value of any data type
requiredAny: PropTypes.any.isRequired,
```

```
// You can chain any of the above with `isRequired` to make sure a warning
// is shown if the prop isn't provided.
requiredFunc: PropTypes.func.isRequired,
```

Prop Types vs TypeScript

- ◆ TypeScript parodo tipų klaidas iš karto (nereikia paleisti projekto)
- ◆ PropTypes parodo klaidas jau kai yra paleistas projektas (naršyklės konsolėje)



Default props

- ◆ Skirtas nustatyti default'inėms komponento parametrų reikšmėms
- ◆ Nenaudojamas kai parametro tipas nustatytas į required
- ◆ Tipų tikrinimas galioja ir default props

```
Persons.defaultProps = {  
  data: [],  
};
```

List & Keys


- ◆ Key visada reikalingas element array (90% kai naudojate map).
- ◆ Key padeda React'ui nustatyti kuris elementas buvo pakeistas (pašalintas, atnaujintas, pakeista jo vieta ir t.t)
- ◆ Key turi būti unikalus savo masyvo scope

✖ ▶ react.development.js?72d0:207
Warning: Each child in an array
or iterator should have a unique
"key" prop.


List & Keys

<https://goo.gl/6cCy3k>

- ◆ Venkite naudoti index kaip key, nes tai gali neigiamai paveikti performance ir sukelti problemas su state'u
- ◆ React by default naudoja index kaip key
- ◆ Index galima naudoti kai:
 - ◆ Masyvas bus statiškas (nesikeis nei tvarka nei patys elementai)
 - ◆ Kai masyvas nebus filtruojamas
 - ◆ Kai neturit unikalaus lauko nustatyti masyvo elementui



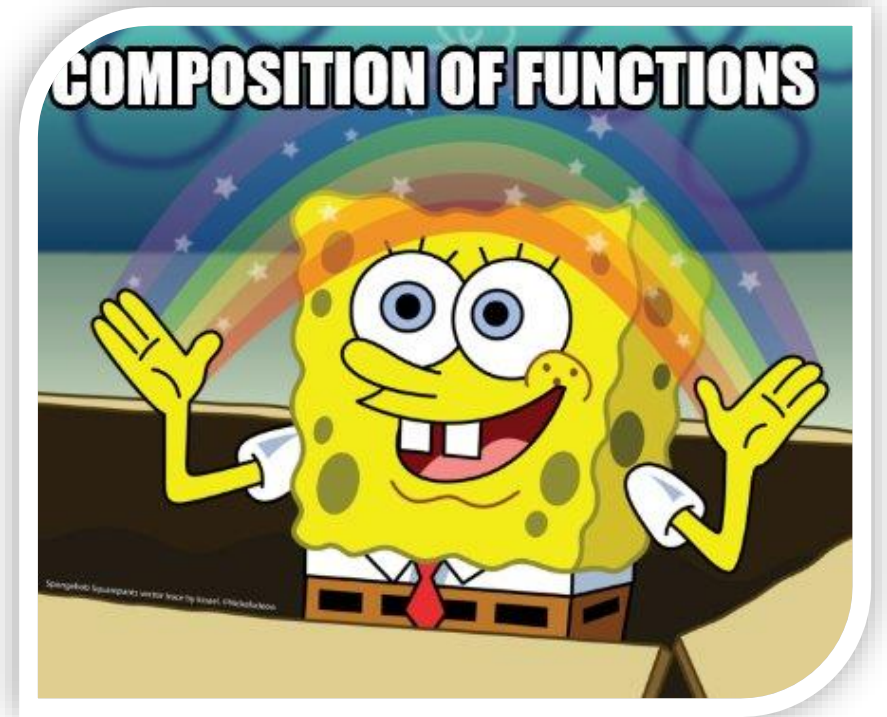
```
{props.data.map(item => (  
  <tr key={item.id}>
```



```
{props.data.map((item, index) => (  
  <tr key={index}>
```

Composition

- ◆ Būdas sukurti kompleksinius komponentus derinant iš kitų mažesnių komponentų
- ◆ Padeda sukurti labiau perpanaudojamus komponentus
- ◆ Panaikina kodo duplikavimą
- ◆ Palengvina testavimą
- ◆ Sumažina klaidų atsiradimą



Composition

- ◆ Komponentai nežino apie vienas kito egzistavimą
- ◆ Specialus children parametras
- ◆ Puikiai tinka dialog arba sidebar komponentams nes juose gali būti bet kas

```
export function FancyBorder(props) {  
  return <div className="FancyBorder">  
    {props.children}  
  </div>;  
}
```

```
export function WelcomeDialog() {  
  return (  
    <FancyBorder color="blue">  
      <h1 className="Dialog-title">  
        Welcome  
      </h1>  
      <p className="Dialog-message">  
        Thank you for visiting!  
      </p>  
    </FancyBorder>  
  );  
}
```

Composition

Jeigu reikia daugiau vietų kur idėsite komponentą (specialus header'is ir content dialog'e) galima naudoti savo props.

```
function SplitPane(props) {  
  return (  
    <div className="SplitPane">  
      <div className="SplitPane-left">  
        {props.left}  
      </div>  
      <div className="SplitPane-right">  
        {props.right}  
      </div>  
    </div>  
  );  
}
```

```
function App() {  
  return (  
    <SplitPane  
      left={  
        <Contacts />  
      }  
      right={  
        <Chat />  
      } />  
  );  
}
```

Composition

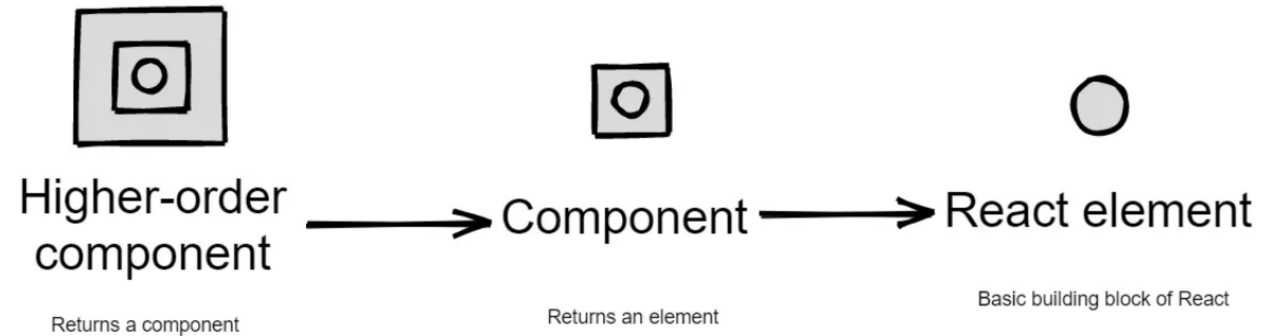
- ◆ Kartais nutinka kad vieni komponentai yra „special case“ kitų komponentų kaip pvz: dialog ir welcomeDialog
- ◆ Iš esmės konkretesnis komponentas viduje render'ina labiau abstraktų komponentą

```
function Dialog(props) {  
  return (  
    <FancyBorder color="blue">  
      <h1 className="Dialog-title">  
        {props.title}  
      </h1>  
      <p className="Dialog-message">  
        {props.message}  
      </p>  
    </FancyBorder>  
  );  
}
```

```
function WelcomeDialog() {  
  return (  
    <Dialog  
      title="Welcome"  
      message="Thank you!" />  
  );  
}
```

Higher-Order Components

- ◆ Funkcija kuri gauna komponentą ir grąžina komponentą
- ◆ Tikslas -> perpanaudoti tą pačią logiką
- ◆ Hoc'ui nėra žinoma kaip bus panaudoti duomenys
- ◆ Wrapp'intui komponentui nesvarbu iš kur tie duomenys atėjo
- ◆ Nemodifikuokite komponento keisdami jo funkcijų
- ◆ **with**HigherOrderComponent



Higher-Order Components

```
class CommentList extends React.Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
    this.state = {
      // "DataSource" is some global data source
      comments: DataSource.getComments()
    };
  }

  componentDidMount() {
    // Subscribe to changes
    DataSource.addChangeListener(this.handleChange);
  }

  componentWillUnmount() {
    // Clean up listener
    DataSource.removeChangeListener(this.handleChange);
  }

  handleChange() {
    // Update component state whenever the data source changes
    this.setState({
      comments: DataSource.getComments()
    });
  }

  render() {
    return (
      <div>
        {this.state.comments.map((comment) => (
          <Comment comment={comment} key={comment.id} />
        ))}
      </div>
    );
  }
}
```

```
class BlogPost extends React.Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
    this.state = {
      blogPost: DataSource.getBlogPost(props.id)
    };
  }

  componentDidMount() {
    DataSource.addChangeListener(this.handleChange);
  }

  componentWillUnmount() {
    DataSource.removeChangeListener(this.handleChange);
  }

  handleChange() {
    this.setState({
      blogPost: DataSource.getBlogPost(this.props.id)
    });
  }

  render() {
    return <TextBlock text={this.state.blogPost} />;
  }
}
```

Higher-Order Components

```
// This function takes a component...
function withSubscription(WrappedComponent, selectData) {
  // ...and returns another component...
  return class extends React.Component {
    constructor(props) {
      super(props);
      this.handleChange = this.handleChange.bind(this);
      this.state = {
        data: selectData(DataSource, props)
      };
    }

    componentDidMount() {
      // ... that takes care of the subscription...
      DataSource.addChangeListener(this.handleChange);
    }

    componentWillUnmount() {
      DataSource.removeChangeListener(this.handleChange);
    }

    handleChange() {
      this.setState({
        data: selectData(DataSource, this.props)
      });
    }

    render() {
      // ... and renders the wrapped component with the fresh data!
      // Notice that we pass through any additional props
      return <WrappedComponent data={this.state.data} {...this.props} />;
    }
  };
}
```

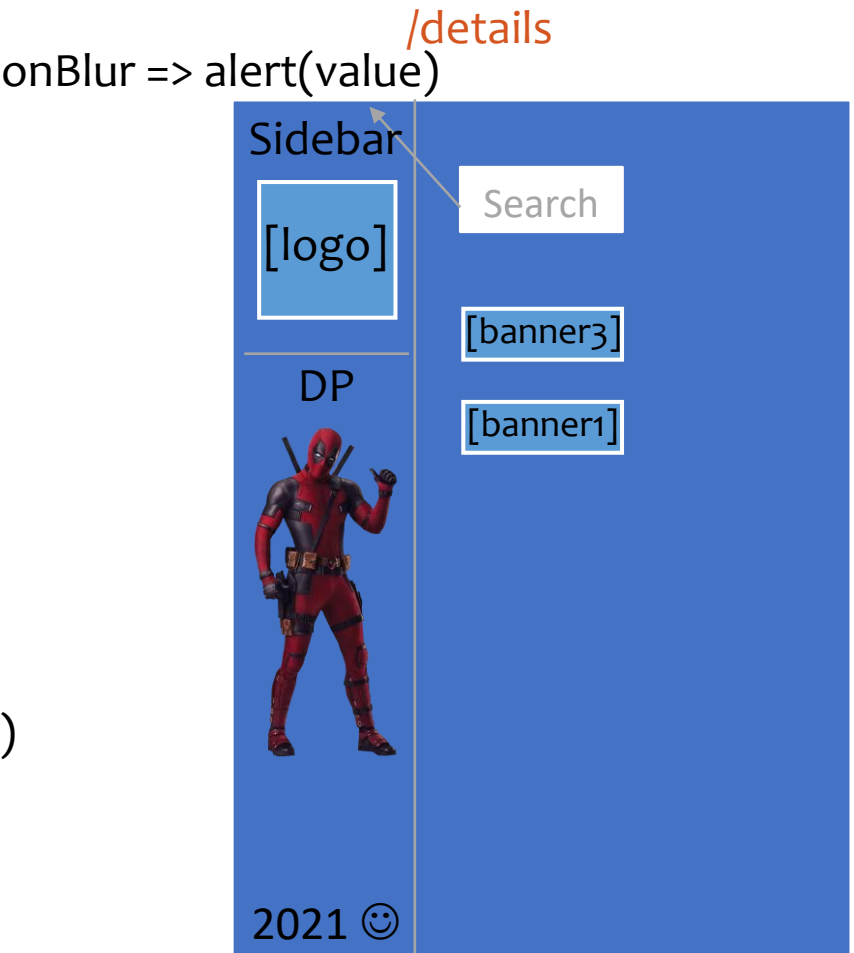
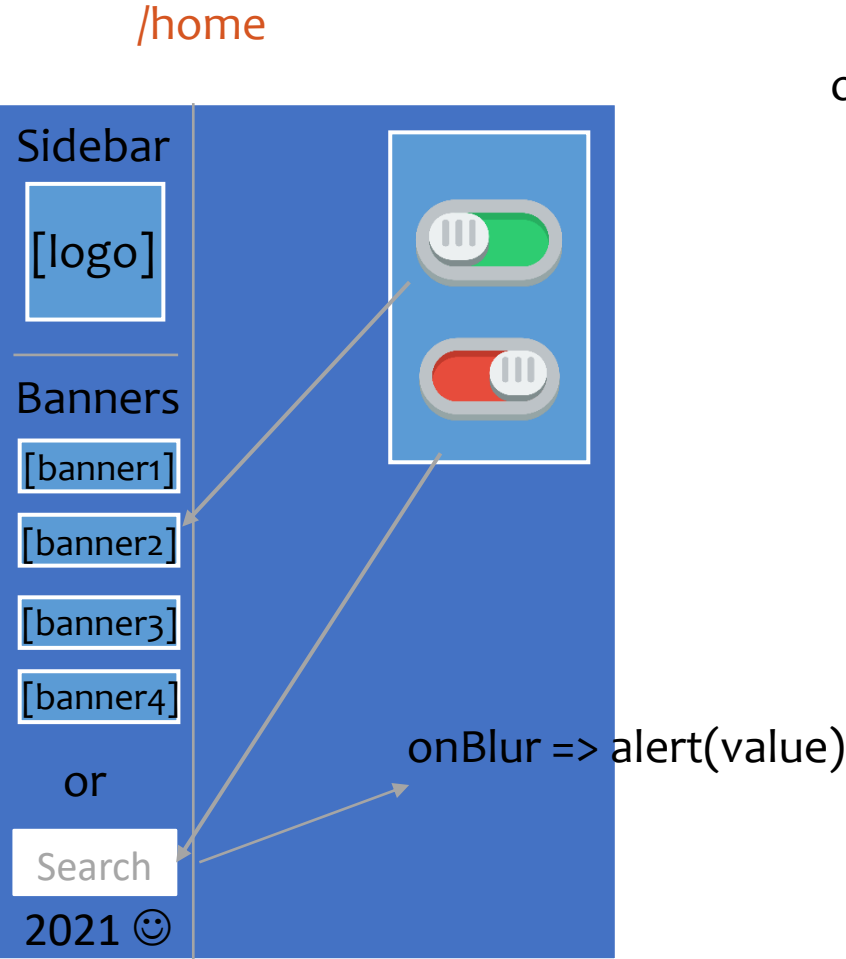
```
const CommentListWithSubscription = withSubscription(
  CommentList,
  (DataSource) => DataSource.getComments()
);

const BlogPostWithSubscription = withSubscription(
  BlogPost,
  (DataSource, props) => DataSource.getBlogPost(props.id)
);
```


DEMO



Workshop



Namų darbai

- ◆ Atsižvelgiam į feedback pasitaisom projektus ir toliau dirbam
- ◆ Pritaikom composition ir HOC's, jeigu yra projekte tam vietų kur galim panaudoti
- ◆ Įvedam Prop Types jeigu nenaudojate Types

