

LIVE CODING SESIJA

KONVERTUOJAM HTML Į REACT





AKADEMIJA.IT

INFOBALT IR TECH CITY

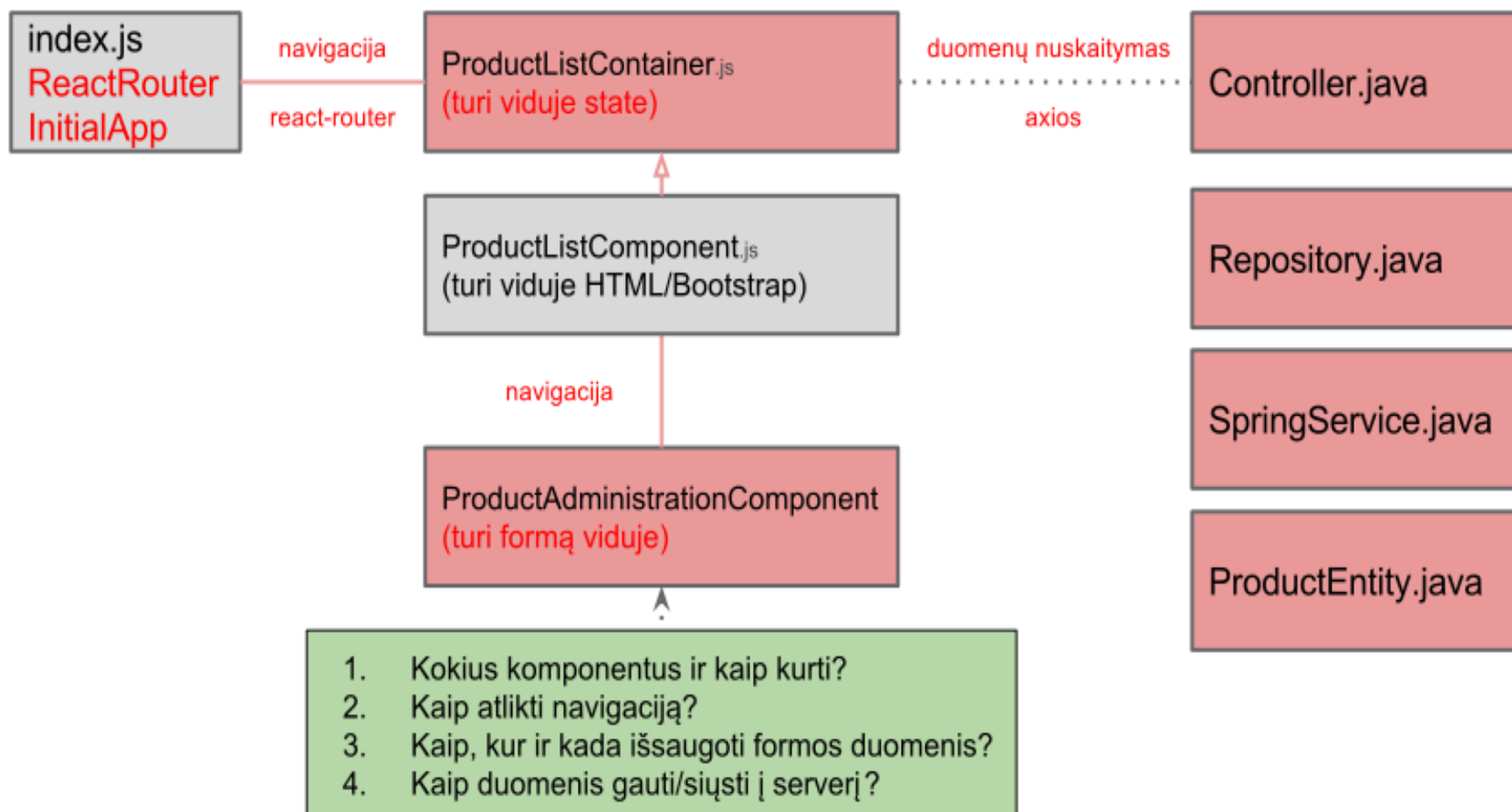
REACT KOMPONENTAI. NAVIGACIJA. GLOBALŪS KINTAMIEJI. DARBAS SU SERVERIU

Andrius Stašauskas

andrius@stasauskas.lt

<http://stasauskas.lt/itpro2018/>

KĄ JAU MOKAME IR KO DAR NE



TURINYS

- Komponentų būsenos ir gyvavimo ciklas
- Puslapio navigacija
- Globalūs kintamieji
- Darbas su serveriu



REACT KOMPONENTŲ BŪSENOS



KOMPONENTO BŪSENA

- Iki šiol komponentai neturėjo būsenos
- Vienintelis būdas atnaujinti komponentą, buvo jį perpiešti naudojant `render()` funkciją



BŪSENOS/DUOMENŲ VALDYMAS

- Viduje
 - Backbone: modeliai/kolekcijos
 - Ember: Ember Data
 - Angular 1: servisai/valdikliai-controllers
- React - lokali komponento būseną
 - "Flux" paternas (duomenys į vieną pusę)
 - Redux (vienas būsenos medis, nekeičiami atnaujinimai)
 - MobX (priklausomybės per observables)
- Angular 2
 - servisai/valdikliai
 - Redux; Observables

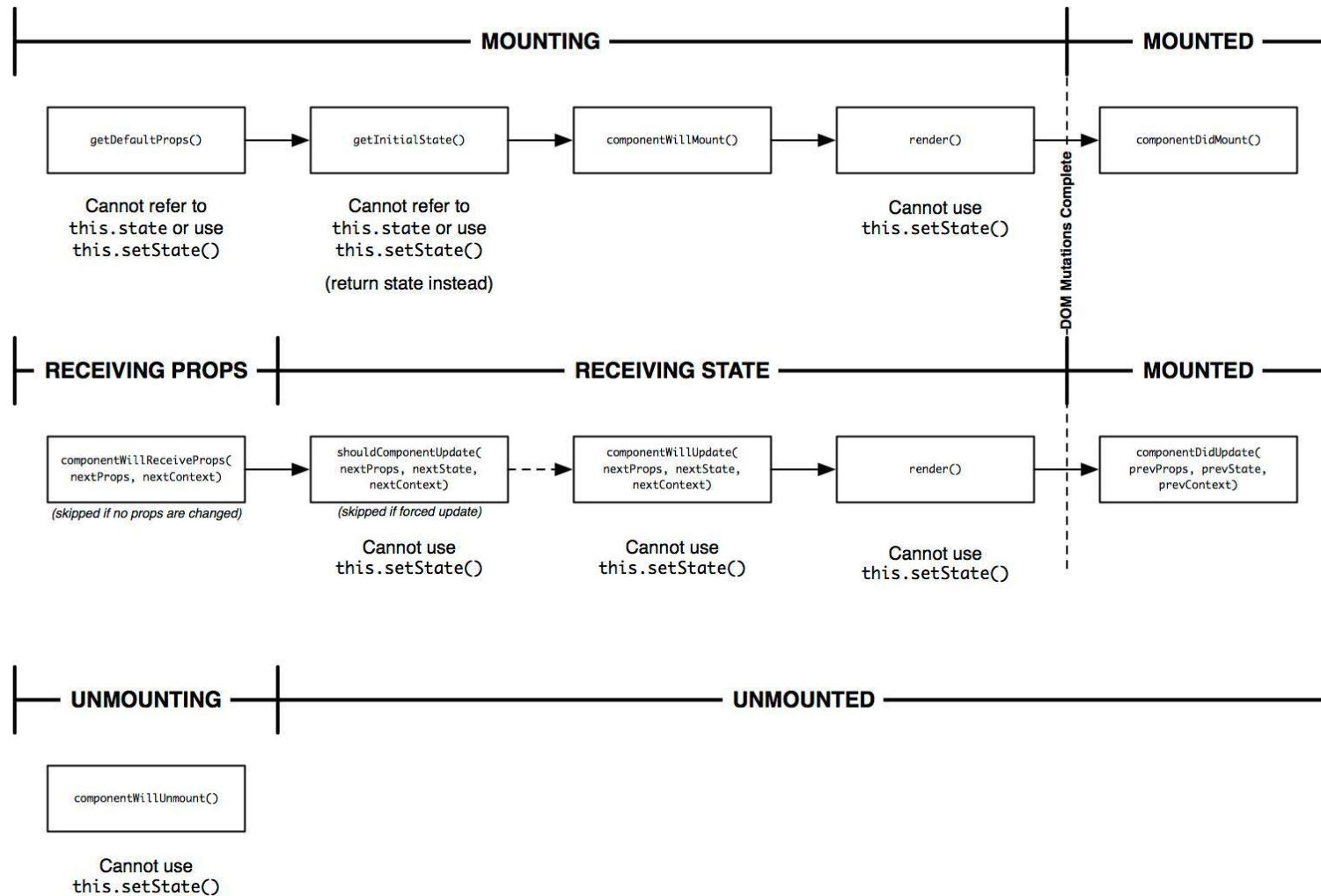


KOMPONENTO BŪSENA

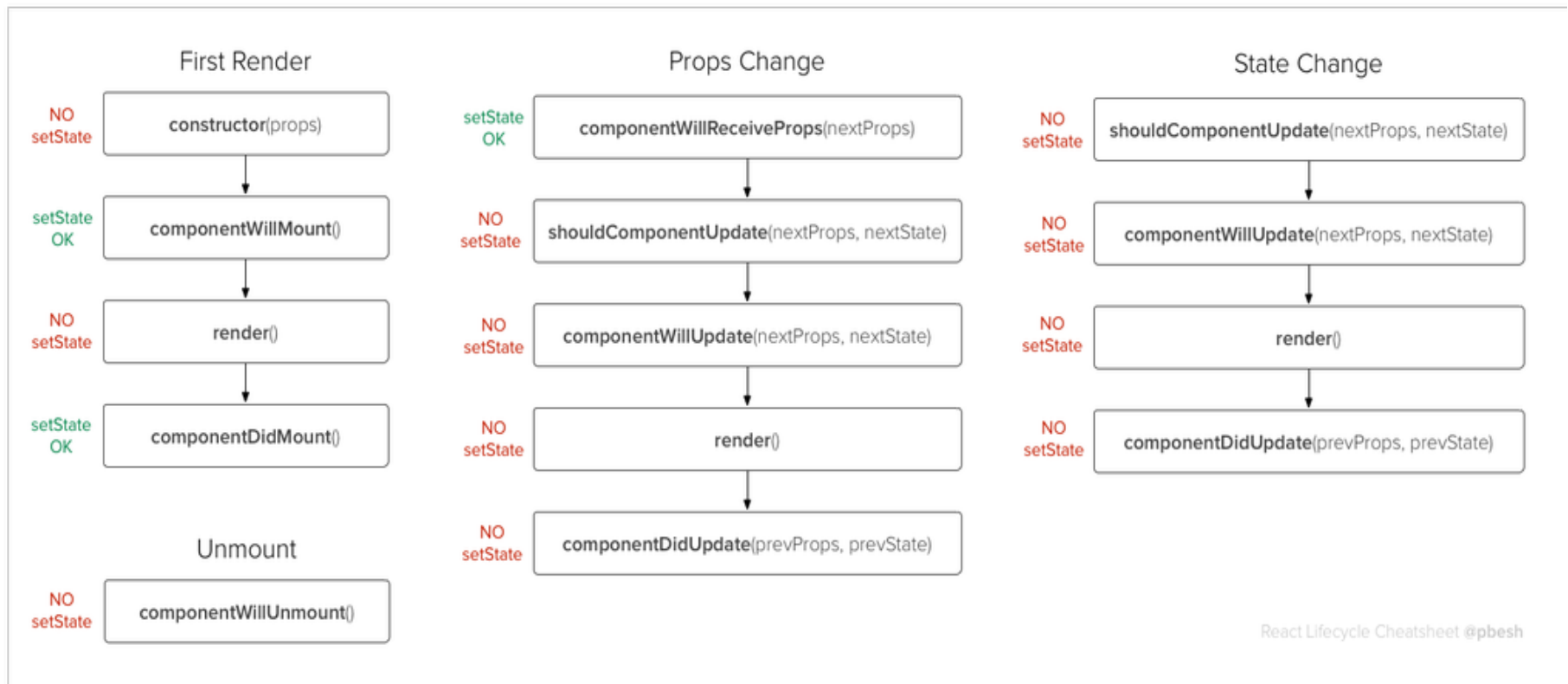
- Būsena saugoma kintamajame `this.state`
- Būsena atnaujinama kviečiant komponento funkciją `this.setState(newState)`
- React'as sulieja esamą būsenos objektą su `newState` būsenos objektu
- React'as automatiškai perpiešia komponentą, kai yra pakviečiama `setState` funkcija
- Dažniausiai būsena keičiama reaguojant į pelės paspaudimo arba formos lauko keitimo įvykius



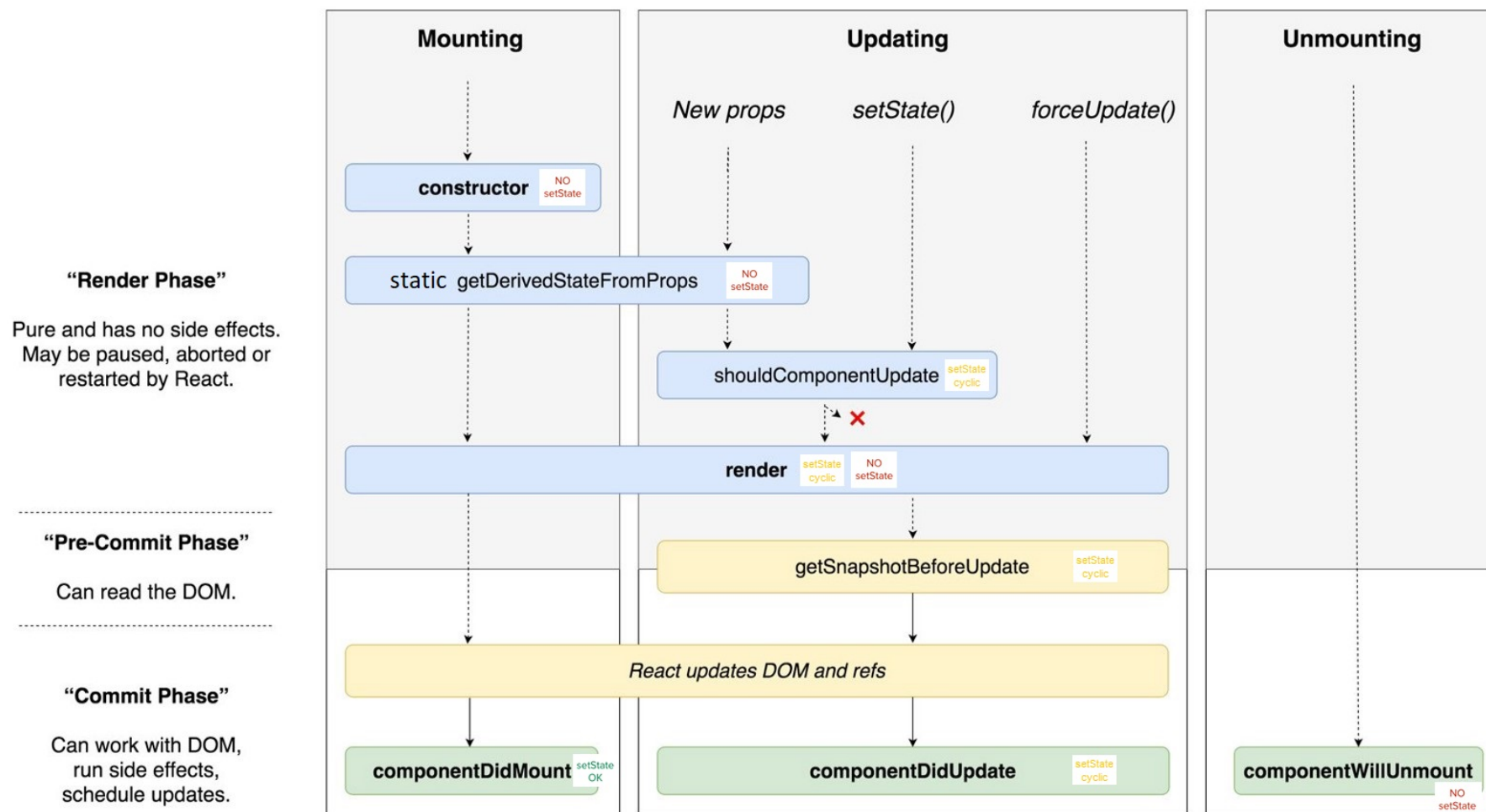
REACT KOMPONENTO GYVAVIMO CIKLAS ES5



REACT KOMPONENTO GYVAVIMO CIKLAS ES6



REACT 16+ KOMPONENTO GYVAVIMO CIKLAS ES6



UŽDUOTIS 1 - GYVAVIMO CIKLAS

- Paduoti props į konstruktorių ir sukurti pradinį `this.state`
- Perrašyti keletą metodų
 - bent jau `getDerivedStateFromProps`, `render`, `componentDidMount`
- Pastebėti, kas po ko ir kiek kartų kviečiasi
 - `console.log("getDerivedStateFromProps");`



DAŽNIAUSIAI NAUDOJAMOS BŪSENOS FUNKCIJOS

- `this.state = {}`
 - turi nustatyti objektą, kuris bus pradinė būsenos reikšmė
 - naudojamas konstruktoriuje
- `render()`
 - turi grąžinti komponento html'ą
 - pasikeitus būsenai, virtualus DOM palygins/apskaičiuos, kas keitėsi, ir iškvies `render`



DAŽNIAUSIAI NAUDOJAMOS BŪSENOS FUNKCIJOS

- `componentDidMount/componentWillUnmount`
 - skirtas sukurti/atšaukti globalius objektus, pvz `setInterval` ar `clearInterval`
 - pakviesti serverį užkraunant duomenis ir perduodant juos `setState`
 - `post-constructor/render` ir `pre-destructor`



PASPAUDIMO ĮVYKIAI

- React'js atributas `onClick` leidžia įvykdyti funkciją paspaudus mygtuką
- `onClick` atributas gali būti net tik ant mygtuko, bet ir ant bet kokio elemento.



PAVYZDYS

```
class IncreasingButtonComponent extends Component {
  constructor() { super(); this.state = { count: 0 }; }
  handleClick = (event) => {
    event.preventDefault(); // event.stopPropagation();
    this.setState({ count: this.state.count + 1 });
  }
  render() {
    return (
      <div>
        {this.state.count} &nbsp;
        <button className="btn btn-default"
          onClick={this.handleClick}>Increase</button>
      </div>
    );
  }
}
```



PAVYZDYS - PAAIŠKINIMAS

- `count`: 0 - paspaudimų skaičius bus saugomas būsenos `count` lauke. Pradžioje turime 0 paspaudimų.
- ` ` - taip html'e rašomas tarpas
- turime arrow funkciją `handleClick`, kuri naudoja `setState` ir padidina `count` skaitliuką vienetu
- `event.preventDefault()` blokuoja numatytąją `onClick` funkciją
- `handleClick` funkcija yra pakviečiama paspaudus mygtuką. Tai padaroma naudojant specialų ReactJs `onClick` atributą



UŽDUOTIS 2 - SUSINAIKINIMO SKAITLIUKAS

- Sukurkite komponentą SelfDestructTimerComponent
- Atidarius puslapį, skaitliukas turėtų pradėti skaičiuoti nuo 42 sekundžių iki 0
- Pasiekus 0, komponento fonas turėtų paraudonuoti



UŽDUOTIS 2 - UŽUOMINOS

- `setInterval` - <https://developer.mozilla.org/en-US/docs/Web/API/WindowTimers/setInterval>
- `clearInterval` - <https://developer.mozilla.org/en-US/docs/Web/API/WindowTimers/clearInterval>
- nepamirškite išvalyti `setInterval`
- `background` CSS atributas



KOMPONENTO ŠABLONAS

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';

class Komponentas extends Component {
  constructor(props, context) {
    super(props, context);
    this.state = { };
  }
  componentDidMount() {
    // užkrauti duomenis iš serverio
    this.setState({ /* čia išsisaugome duomenis iš serverio */ })
  }
  render() {
    return (<div> { /* Component view */ } </div>);
  }
}
```



REACT FORMOS

- Formos React'e išsiskiria iš kitų elementų dėl to, kad kiekvienas laukas savyje turi būseną
- React'e būseną laikoma state lauke
- Formos elemento atributai `onChange` ir `value` naudojami sinchronizuoti elemento būseną su state lauku
- `onChange` priima funkciją, kurią pakviečia pasikeitus formos lauko reikšmei
- `value` atspindi formos reikšmę
- jei norime pakeisti formos lauko reikšmę iš javascript'o, turime keisti state lauką, kuris yra `value` išraiška



REACT PAVYZDYS

```
class NameForm extends Component {
  constructor() { super(); this.state = { value: '' }; }
  handleChange = (event) => this.setState({value: event.target.value})
  handleSubmit = (event) => {
    this.setState({value: 'reset after submit'});
    event.preventDefault();
  }
  render() {
    return (<form onSubmit={this.handleSubmit}>
      Name ({this.state.value}):<br/>
      <input type="text" value={this.state.value}
        onChange={this.handleChange} />
      <input type="submit" value="Submit" />
    </form>);
  }
}
```



REACT PAVYZDYS - PAAIŠKINIMAS

- kaskart vedant reikšmę yra kviečiama `handleChange` funkcija, kuri naujai gautą reikšmę iš `event` parametro perkelia į `state.value`
- paspaudus `Submit` mygtuką į `console` atspausdinama reikšmė ir pakeičiama į `'reset after submit'`
- `event.preventDefault()` reikalingas tam, kad nebūtų vykdomas standartinis formos `submit`'as - formos duomenų siuntimas į serverį
- ateityje `handleSubmit` paskirtis būtų pakviesti serverį (REST Api) ir nusiųsti jam duomenis



UŽDUOTIS 3 - PRODUKTO SUKŪRIMO FORMA

- sukurkite ProductAdministrationComponent
- Produktą apibūdinantys laukai: title, imageUrl, description, price, quantity
- Mygtukas Save
- Paspaudus Save išspausdinkite visą produkto informaciją
- Naudokite Bootstrap'o formas:

<https://getbootstrap.com/docs/4.1/components/forms/>



KOMPONENTAI BE BŪSENOS

- Stateless Functional Components
- Komponentai, kurie turi tik render funkciją
 - tai yra pats komponentas ir yra render funkcija
- Užrašomi trumpiau, kaip paprastos JS funkcijos:

```
var Komponentas = (props) => { // arba ({atributas, kitas}) => {  
  var {atributas, kitas} = props; // destructuring assignment  
  return (<div>{atributas} {kitas}</div>);  
}
```

Panaudojimas

```
<Komponentas atributas="reiksme" kitas="nieko"/>
```



KOMPONENTŲ PASKIRTIS

- Dažniausia priimta turėti 2 tipų komponentus
 - Presentation (vaizdavimo)
 - Container (būsenos)
- React'o požiūriu tai yra tiesiog React'o komponentai
- Visada pora: ProductListComponent ir ProductListContainer



PRESENTATION KOMPONENTAI

- turi tik `render()` funkciją
- yra Stateless Functional Component todėl gali būti apibrėžti kaip funkcija
- atsakingi tik už piešimą
- visa piešimui reikalinga informacija yra perduodama per `props`



CONTAINER KOMPONENTAI

- turi `render()` funkciją kuri piešia Presentation componentą
 - daugiau nieko nepiešia
- gali turėti `state` ir kitus gyvavimo ciklo elementus
- atsakingas surinkti duomenis, registruoti `click` funkcijas ir valdyti būseną



PRESENTATION KOMPONENTO PAVYZDYS

```
var SomeComponent = (props) => {  
  return (  
    <div>  
      { /* A lot of html */ }  
    </div>  
  );  
}  
  
SomeComponent.propTypes = {  
  prop1: PropTypes.string.isRequired  
};
```



CONTAINER KOMPONENTO PAVYZDYS

```
class SomeContainer extends Component {  
  constructor(props) { super(props); this.state = {}; }  
  componentDidMount() {}  
  // Other functions  
  render() {  
    return <SomeComponent prop1={this.state.prop1} />  
  }  
})
```



METODŲ ATSKYRIMAS Į CONTAINER'Į

- Per props galima perduoti nebūtinai laukų reikšmes, galima perduoti ir pačias funkcijas
 - pvz. perduoti metodą iš container komponento į presentation galima tiesiog per komponento atributą

```
class FormaContainer extends React.Component {
  onReiksmeChange = (e) => this.setState({reiksme: e.target.value})
  <...>
  render() return <Forma onReiksmeChange={this.onReiksmeChange}

const Forma = ({onReiksmeChange}) => {
  return (<form>
    <input type="text" value={reiksme} onChange={onReiksmeChange}/>
  </form>);
}
```

- visas pavyzdys: <https://jsfiddle.net/w2hz4o5n/>



UŽDUOTIS 4 - KOMPONENTAS BE BŪSENOS

- pakeisti create-react-app paveiksluką taip, kad jis suktųsi tik užvedus pele
- sukurkite komponentą Basisukantis be būsenos
- komponentas turi gražinti logo paveiksluką su `className=""`
- prie `img` pridėkite `onMouseOver` ir `onMouseOut` atributus
- sukurkite `arrow` funkciją Basisukantis komponento viduje, kurių viena `event.target` nustato `className=""`, o kita `className="App-logo"`
- `img` pridėkite `style={{height : 70}}` dėl dydžio



PUSLAPIO NAVIGACIJA



PUSLAPIO NAVIGACIJA

- kol kas visi mūsų komponentai buvo viename puslapyje
- normalu, kad puslapio naudotojai norėtų naviguoti spausdami mygtukus, linkus, paveikslėlius ir t.t.
- šiam funkcionalumui įgyvendinti bus naudojama React Router biblioteka
- <https://github.com/reacttraining/react-router>



KLIENTO PUSĖS NAVIGAVIMAS

- routing/navigavimas: URL adreso priskyrimas elgsenai, taip pat išskiriant duomenų parametrus iš URL
- istoriškai buvo serverio atsakomybė
- kuriant modernią aplikaciją navigavimas kliento pusėje leidžia greičiau atnaujinti puslapį
- į package.json dependencies reikia pridėti:

```
"react-router": "^4.3.1",  
"react-router-dom": "^4.3.1"
```



NAVIGACIJOS KOMPONENTAI

- Navigacijos konfigūracijos
 - Switch
 - Redirect
 - Route
 - BrowserRouter
 - HashRouter
- Navigacijos: Link arba tiesiog `history.push()`



- Į index.js importuojame Router

```
import { Switch, Redirect, Route } from 'react-router';  
import { BrowserRouter, Link } from 'react-router-dom';
```

- komponentas AppContainer, kuriame bus navigacija

```
var AppContainer = (props) => {  
  return (<div>  
    <div>  
      <Link to='/'>Home</Link> |&nbsp;nbsp;nbsp;  
      <Link to='/products'>Products</Link> |&nbsp;nbsp;nbsp;  
      <Link to={`/${products}/${127}`}>Product by no</Link> |&nbsp;nb  
      <Link to='/help'>help</Link> |&nbsp;nbsp;nbsp;  
      <Link to='/non-existent'>Non Existent</Link>  
    </div>  
    {props.children}  
  </div>);  
};
```



- Komponentas, kurį rodysime neegzistuojančiam keliui

```
var NoMatch = (props) => {  
  var goApp = () => props.history.push("/");  
  return <div>Route did not match  
    <button onClick={goApp}>Go Home</button></div>;  
};
```

- Į App.js esantį komponentą įsidedame mygtuką navigavimui į nuorodą /products

```
goProducts = () => this.props.history.push("products");  
// o patį mygtuką kur nors į render() metoda  
<p><button onClick={this.goProducts}  
  className="btn btn-primary"  
  role="button">  
    Go to Products  
</button></p>
```



- Šis komponentas skirtas pademonstruoti navigaciją (index.js)

```
var DemonstruotiNavigacija = (props) => {  
  var goHome = () => props.history.push("/");  
  return (  
    <div>  
      At route: {props.location.pathname}  
      <button onClick={goHome}>Go Home</button>  
      <pre>  
        {JSON.stringify(props, null, 2)}  
      </pre>  
    </div>  
  );  
};
```



- Pakeisti ReactDOM.render iš <App/> į BrowserRouter:

```
ReactDOM.render((  
  <BrowserRouter>  
    <AppContainer>  
      <Switch>  
        <Route exact path="/" component={App}/>  
        <Route path="/products/:id" component={DemonstruotiNavigacija}/>  
        <Route path="/products" component={DemonstruotiNavigacija} />  
        <Route path="/help" component={DemonstruotiNavigacija} />  
        <Route path="*" component={NoMatch}/>  
        <Route component={NoMatch}/>  
      </Switch>  
    </AppContainer>  
  </BrowserRouter>  
, document.getElementById('root'));
```

- Dabar pradinis taškas bus BrowserRouter, o ne App



PAAIŠKINIMAS

- Router'io konfigūracija prasideda eilute `<BrowserRouter>`
- `path` pasako koks kelias url'e atitiks kokį komponentą
 - Tarkime, kai url'e bus `/products` bus piešiamas `DemonstruotiNavigacija`
- `path` atributas gali būti šablonas
- `/products/:id` atitiks visus kelius tokius kaip `products/1`, `products/new`
- `:id` reikšmė bus patalpinta `props.params.id`



BROWSERROUTER YRA ROUTER SU BROWSER HISTORY

- tai yra šis trumpesnis kodas

```
import { BrowserRouter } from 'react-router-dom';  
<BrowserRouter/>
```

- atlieka tą patį ką ir šis ilgesnis kodas

```
import { Router } from "react-router";  
import { createBrowserHistory } from "history";  
const history = createBrowserHistory(props);  
<Router history={history} />
```

- naudingas, jei reikia history perduoti toliau



NAVIGACIJA SU HISTORY

- Navigacija į /products

```
history.push('/products')
```

arba

```
history.push({pathname: '/products', search: '?p=1', state: {}})
```

- Dokumentacija

<https://github.com/ReactTraining/history#navigation>



NAVIGACIJA SU LINK

- Navigacija i /products

```
<Link to="/products">Go to Products</Link>
```

arba

```
<Link to={{pathname: '/products', query: {p: '1'}}}>Go to Products</Link>  
<Link to={`/${products}/${127}`}>Go to Products</Link>
```

- Dokumentacija <https://knowbody.github.io/react-router-docs/api/Link.html>



UŽDUOTIS 5 - NAVIGACIJA #1

- Naudokite prieš tai užduotyje sukurtus komponentus
 - ProductListComponent ir ProductAdministrationComponent
- / ir /products turi rodyti produktų sąrašą (ProductListComponent)
- /admin/products/new - ProductAdministrationComponent
- /admin/products/:id - ProductAdministrationComponent



UŽDUOTIS 5 - NAVIGACIJA #2

- papildykite ProductAdministrationComponent, kad jis rodytų priklausomai nuo kelio puslapio antraštę:
 - Kuriamas naujas produktas, jei url
`/admin/products/new`
 - Atnaujinamas produktas 'id', jei url
`/admin/products/:id`



GLOBALŪS KINTAMIEJI



GLOBALŪS KINTAMIEJI

- `index.js` nustatyti į props ir perduoti atributus toliau komponentams per props
 - `teisingas`, bet "užteršia" kodą
- variantas senuoju būdu (`nenaudotina`): susidėti servisu į `var window`
 - t.y. panaudoti globalų naršyklėje egzistuojantį kintamąjį `window`
- React 15 `experimental` static `.contextTypes` (`deprecated`)
 - jį pakeitė `teisingas` React 16 Context API



GLOBALŪS KINTAMIEJI

- React kiekvienam komponentui gali perduoti kontekstą, tada nereikia vis perduot per props. Konteksto sukūrimas

```
const ServicesContext = React.createContext(null);
```

- deklaravimas šakniniame komponente

```
<ServicesContext.Provider value={{userService: userService}}>  
  <AppContainer/> <!-- kontekstą gaus šis ir visi viduje -->  
</ServicesContext.Provider>
```

- ten, kur reikia panaudoti

```
<ServicesContext.Consumer>  
  ({userService}) => <span>{userService.name}</span>  
</ServicesContext.Consumer>
```

- dokumentacija <https://reactjs.org/docs/context.html>



REACT CONTEXT

- Iš dokumentacijos - kodėl nenaudoti context

Context is primarily used when some data needs to be accessible by many components at different nesting levels. Apply it sparingly because it makes component reuse more difficult. Context lets us pass a value deep into the component tree without explicitly threading it through every component.



UŽDUOTIS 6 - GLOBALŪS KINTAMIEJI

- sukurkite UserService implementaciją kad būtų įmanoma išsaugoti username
- įsidėkite UserService į globalų kontekstą
- kuriame nors paslapyje parašykite Labas ,
<username>!, kur username būtų panaudotas
UserService username
- kuriame nors kitame puslapyje onClick nustatykite
UserService username reikšmę
 - grįžkite į puslapį su pasisveikinimu ir username turi būti pasikeitęs



DARBAS SU SERVERIU



DARBAS SU SERVERIU

- Naudosime Axios (<https://github.com/mzabriskie/axios>)
- Sužinosime, kas yra Promise'as
- Pamatysime asinchroninį kodo vykdymą



PAVYZDINIS API

- Duomenis krausime iš
<https://itpro2017.herokuapp.com/swagger-ui.html#/>
- Tarkime šiuo metu parduodamų produktų sąrašas yra čia:
<https://itpro2017.herokuapp.com/api/products>
- Kaip tai pakrauti į mūsų React'o appą?



PAVYZDINIS API

- package.json

```
"axios" : "0.18.0"
```

- kaip naudoti: importuoti, įvykdyti užklausą, apdoroti atsakymą ir klaidas

```
import axios from 'axios';
axios.get('https://itpro2017.herokuapp.com/api/products')
.then( (response) => {
  console.log(response);
})
.catch( (error) => {
  console.log(error);
});
```



- `axios.get('URL ')` grąžina Promise tipo objektą
- Promise objektas yra kaip pažadas, kad kažkas į jo vidų kažkada įdės reikšmę
- `then(fn)` ir `catch(fn)` yra Promise objekto funkcijos priimančios funkcijas (callback'us) kurios bus įvykdytos, kai kas nors į promise'ą įdės rezultatą



- `then(fn)` - `fn` yra pakviečiamas, kai serveris grąžina sėkmingą rezultatą (HTTP 200 OK)
- `catch(fn)` - `fn` yra pakviečiamas, kai serveris grąžina nesėkmingą rezultatą (HTTP >400)
- serverio kvietimas vyksta asinchroniskai - kodas vykdomas toliau, nors serveris dar neatsakė



THEN(FN)

- callback funkcijai yra perduodamas response objektas
- <https://github.com/mzabriskie/axios#response-schema>
- data - serverio rezultato JSON'as
- status - serverio vykdymo kodas. 200 reiškia sėkmingą vykdymą.



- Kiekvienas HTTP method turi atitinkamą funkciją Axios bibliotekoje
- `axios.get`, `axios.post`, `axios.delete`, `axios.put`



INTEGRACIJA SU REACT'U

- Dažniausiai serveris kviečiamas kai:
 - užkraunamas komponentas ir jį reikia užpildyti duomenimis
 - įvyksta vartotojo veiksmas (formos išsaugojimas) ir reikia nusiųsti serveriui duomenis



UŽKRAUNANT PRADINĮ KOMPONENTO VAIZDĄ

```
componentDidMount() {  
  axios.get('https://itpro2017.herokuapp.com/api/products')  
    .then((response) => {  
      this.setState({ product: response.data });  
    })  
}  
}
```



UŽDUOTIS 7 - RODYKITE PRODUKTŲ SĄRAŠĄ IŠ SERVERIO

- Sukurkite ProductListContainer, kuris atsakingas už būsenos valdymą
- ProductListContainer container turi pakviesti serverį ir išpiešti ProductListComponent perduodamas jam produktų sąrašą
- Pakeiskite ProductListComponent taip, kad jis rodytu produktų sąrašą iš <https://itpro2017.herokuapp.com/api/products>
- Paveikslėliai nesimatys šiuo atveju.



UŽDUOTIS 8 - IŠSAUGOKITE NAUJĄ PRODUKTĄ SERVERYJE

- ProductAdministrationComponent iškelkite būseną į ProductAdministrationContainer komponentą
- Spaudžiant mygtuką Save padarykite POST <https://itpro2017.herokuapp.com/api/products>



KODO STRUKTŪRA

```
.
├── public
│   └── index.html
├── src
│   ├── index.js
│   └── components
│       ├── Navigation
│       │   └── NavigationComponent.js
│       ├── ProductList
│       │   ├── ProductCardComponent.js
│       │   └── ProductListContainer.js
│       └── ProductAdministration
│           ├── ProductAdministrationComponent.js
│           └── ProductAdministrationContainer.js
```

index.js importuotumēm taip:

```
import ProductListContainer
  from './components/ProductList/ProductListContainer';
```



KODO STRUKTŪRA - TASYKLĖS

- Iškelkite komponentus į atskirus failus
- `index.js` aprašykite React Router taisykles
- `NavigationComponent.js` aprašykite navigacijos meniu
- failo pavadinimas turi sutapti su komponento pavadinimu



UŽDUOTIS 9 - PERTVARKYKITE KODĄ

- Atskirkite komponentus į atskirus failus
- sukurkite components katalogą ir jame sukurkite katalogus skirtingiems komponentams
 - komponento container'is irgi yra komponento dalis
- index.js aprašykite React Router taisykles



UŽDUOTIS 10 - UŽBAIKITE EL. PARDUOTUVĘ

- <https://itpro2017.herokuapp.com>
- pabandykite patys sukurti tai, ko trūksta
- jei užstrigote, žiūrėkite į šalia skaidrių esančių dalies užduočių sprendimus
 - sprendimai ne visi
 - gali tekti persidaryti



UŽDUOTIS 11 - PRIJUNGTI SASS

- React nerekomenduoja naudoti SASS. Pvz. vietoj to, kad `.Button` klasę naudotumėte `<Ok>` ir `<Cancel>` mygtukuose, React rekomenduoja sukurti vieną `<Button>` mygtuką, kurį `<Ok>` ir `<Cancel>` galėtų nupiešti `render()` metode
- Dėl to SASS/LESS mažai naudingi, nes vietoj to naudojama komponentų kompozicija
- Kaip prijungti SASS
 - <https://facebook.github.io/create-react-app/docs/adding-a-sass-stylesheet>



SAVARANKIŠKAM PASIMOKYMU

- <https://hackr.io/tutorials/learn-react> - nemokami kursai
 - pvz <https://www.codecademy.com/learn/react-101>
- React Biblioteka ir jos dokumentacija
 - <https://reactjs.org/>
- dalies užduočių sprendimai
 - <http://stasauskas.lt/itpro2018/>



KITOJE PASKAITOJE

XML. Karkasai. Sistemų architektūros



AKADEMIJA.IT

INFOBALT IR TECH CITY