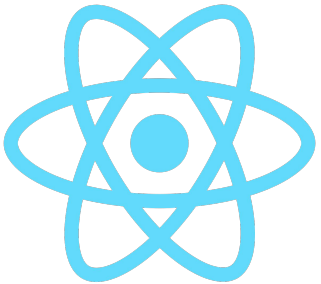
React Context vs Redux



Tomasz Fiechowski

tfiechowski

Agenda

Dlaczego Context? 🤔

Context 🤷
- wprowadzenie,
- sposoby użycia,

Obsługa wielu Context'ów 👫

Redux 📊
- porównanie i zestawienie funkcjonalności

Testowanie 🛠️
- Wzorzec `InnerComponent`, API jako `prop`

Przykłady z działających aplikacji 💪

Rota codility

Dlaczego context? 🤔

Dan Abramov @dan_abramov · Sep 11

When 10 people say “I dislike Redux” they might mean 10 completely different things. And they often have nothing to do with Redux itself, but with how the code they worked with is structured, or with the examples they learned from.

18 19 192

Dan Abramov @dan_abramov · Sep 11

If you copy paste some action creators and reducers handling FETCH actions over and over and over again, you’re probably using Redux in a different way than I imagined people would do. I’m sorry for all the repetitive code you felt you needed to write. That’s my fault.

17 25 204

Dan Abramov @dan_abramov · Sep 11

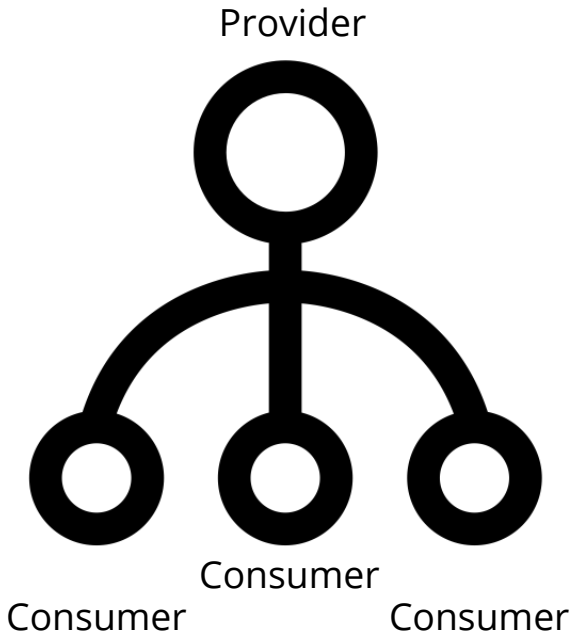
Same goes for having 20 actions with types like SET_A, SET_B, SET_C, and dispatching them from a long-ass async action creator. Also not how I imagined people would do it. Again, sorry I didn’t explain this clearly.

React Context 🤦

React Context

= Provider + Consumers

React Context

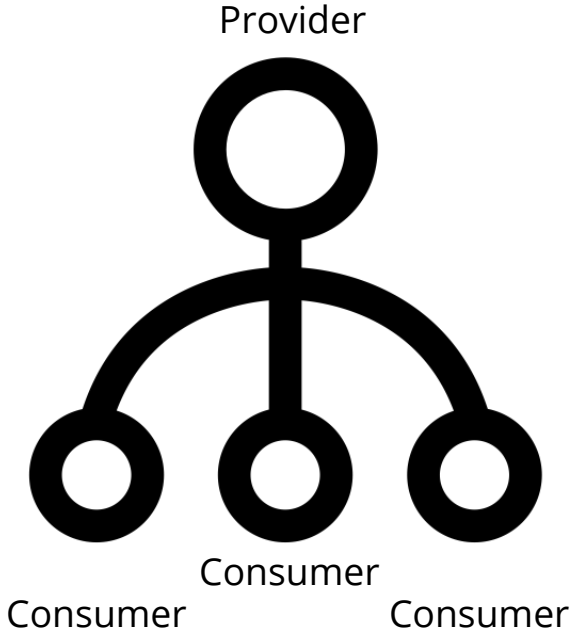


Rota

codility

React Context

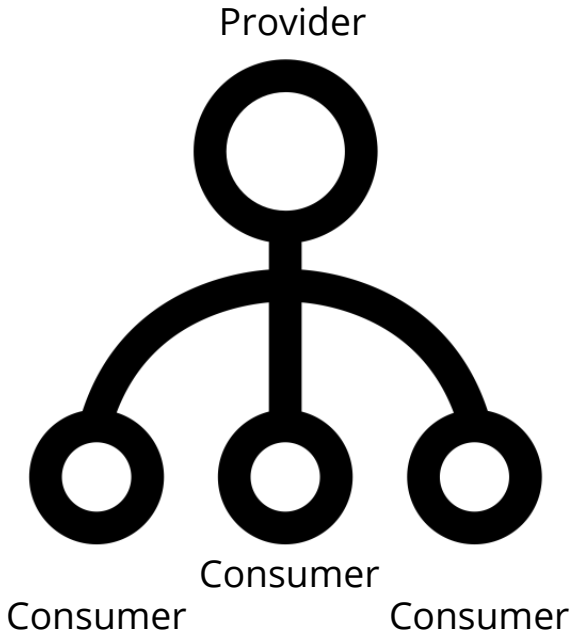
```
import React, { Component, createContext } from "react";

const Context = createContext({});
```



Rota

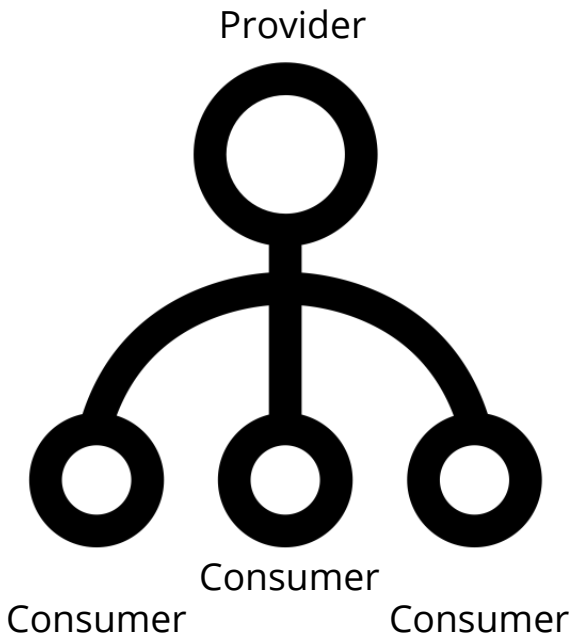
codility

React Context




```
import React, { Component, createContext } from "react";

const Context = createContext({});

export class PhotosProvider extends Component {
  constructor(props) {
    super(props);

    this.state = {
      photos: [],
    };
  }

  render() {
    return (
      <Context.Provider value={this.state}>
        {this.props.children}
      </Context.Provider>
    );
  }
}

export default Context;
```

React Context + actions




```
import React, { Component, createContext } from "react";
import api from "./api";

const Context = createContext({});

export class PhotosProvider extends Component {
  constructor(props) {
    super(props);

    this.state = {
      photos: [],
      fetchPhotos: this.fetchPhotos
    };
  }

  render() {
    return (
      <Context.Provider value={this.state}>
        {this.props.children}
      </Context.Provider>
    );
  }

  fetchPhotos = async () => {
    const photos = await api.fetchPhotos();
    this.setState(state => ({ photos: [state.photos, ...photos] }));
  };
}

export default Context;
```

Rota codility

Provider

```
import { PhotosProvider } from "./contexts/PhotosContext/PhotosContext";

ReactDOM.render(
  <PhotosProvider>
    <App />
  </PhotosProvider>,
  document.getElementById("root")
);
```

Co na to konsumenci? 

React Context - PhotosList helper

```
export default function PhotosList({ photos, fetchPhotos }) {
  return (
    <div>
      {photos.map(photo => (
        <div key={photo.id}>{photo.title}</div>
      ))}

      <button onClick={fetchPhotos}>Fetch more photos</button>
    </div>
  );
```

Rota codility

React Context - Context.Consumer

```
import React from "react";
import PhotosList from "../components/PhotoList";
import PhotosContext from "../contexts/PhotosContext";

export default function PhotoListContextConsumer() {
  return (
    <PhotosContext.Consumer>
      {({ photos, fetchPhotos }) => (
        <PhotosList photos={photos} fetchPhotos={fetchPhotos} />
      )}
    </PhotosContext.Consumer>
  );
}
```

Rota codility

React Context - contextType

```
import React, { Component } from "react";
import PhotosList from "../components/PhotoList";
import PhotosContext from "../contexts/PhotosContext";

export default class PhotoListContextType extends Component {
  static contextType = PhotosContext;

  render() {
    const { photos, fetchPhotos } = this.context;

    return <PhotosList photos={photos} fetchPhotos={fetchPhotos} />;
  }
}
```

React Context - useContext (hook)

```
import React, { useContext } from "react";
import PhotosList from "../components/PhotoList";
import PhotosContext from "../contexts/PhotosContext";

export default function PhotoListUseContext() {
  const { photos, fetchPhotos } = useContext(PhotosContext);

  return <PhotosList photos={photos} fetchPhotos={fetchPhotos} />;
}
```

React Context - podsumowanie

1. Inicjalizacja context'u
```
const Context = createContext({});
```

2. Renderujemy Provider
```
ReactDOM.render(
 <PhotosProvider><App /></PhotosProvider>,
 document.getElementById("root")
);
```

3. Dostęp do Contextu poprzez: `Consumer`, `contextType` lub `useContext`.
4. Profit!

Rota codility

React Context - podsumowanie

```
  constructor(props) {
    super(props);

    this.state = {
      photos: [],
      fetchPhotos: this.fetchPhotos
    };
  }

  render() {
    return (
      <Context.Provider value={this.state}>
        {this.props.children}
      </Context.Provider>
    );
  }

export default function PhotoListUseContext() {
  const { photos, fetchPhotos } = useContext(PhotosContext);

  return <PhotosList photos={photos} fetchPhotos={fetchPhotos} />;
}
```

Rota codility

React Context - podsumowanie

```
render() {
  return (
    <Context.Provider value={{ photos: [] }}>
      {this.props.children}
    </Context.Provider>
  );
}
```

https://reactjs.org/docs/context.html#caveats

Rota codility

Obsługa wielu Context'ów 

React Context - multiple contexts

Providers

```
render(
  <BrowserRouter>
    <ChangeRequestProvider>
      <AvailabilityProvider>
        <DeadlineProvider api={deadlineApi}>
          <ColumnProvider>
            <ShiftProvider>
              <UserProvider>
                <Page />
              </UserProvider>
            </ShiftProvider>
          </ColumnProvider>
        </DeadlineProvider>
      </AvailabilityProvider>
    </ChangeRequestProvider>
  </BrowserRouter>,
  window.document.getElementById('root'),
);
```

Consumers

```
return (
  <UserConsumer>
    {({ currentTeam, user, isAdminInCurrentTeam, getUser }) => (
      <ColumnConsumer>
        {({ addColumns, columns, getColumn }) => (
          <DeadlineConsumer>
            {({ fetchDeadlines, deadlines }) => (
              <ShiftConsumer>
                {({
                  fetchShifts,
                  giveAwayShift,
                  modifyShifts,
                  shifts,
                  suggestAdjustment,
                  removeAdjustment,
                  acceptAdjustment,
                  rejectAdjustment,
                  removeAdjustmentSuggestion,
                  saveAdjustment,
                  takeShift,
                  undoGiveAwayShift,
                }) => (
                  <AvailabilityConsumer>
                    {({ availabilities, fetchAvailabilities }) => (
```

Rota codility

React Context - Consumer nesting

```
return (
  <ThemeConsumer>
    {({ theme }) => (
      <PhotosConsumer>
        {({ photos, fetchPhotos }) => (
          <PhotosPage photos={photos} fetchPhotos={fetchPhotos} theme={theme} />
        )}
      </PhotosConsumer>
    )}
  </ThemeConsumer>
);
```

Rota codility

React Context - Consumer nesting

Helper:

```
function ContextConsumer({ children }) {
  return (
    <PhotosContext.Consumer>
      {photos => (
        <ThemeContext.Consumer>
          {theme => children({ photos, theme })}
        </ThemeContext.Consumer>
      )}
    </PhotosContext.Consumer>
  );
}
```

```
export default function PhotoListContextConsumer() {
  return (
    <ContextConsumer>
      {({ photos: { photos, fetchPhotos }, theme: { theme } }) => (
        <PhotosList
          photos={photos}
          fetchPhotos={fetchPhotos}
          theme={theme}
        />
      )}
    </ContextConsumer>
  );
}
```

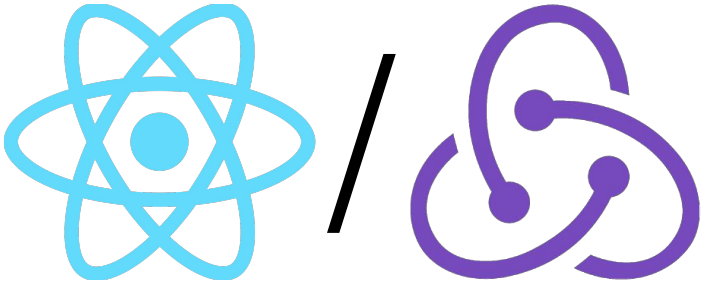
Rota codility

React Context - Consumer nesting

```
export default function PhotoListUseContext() {
  const { photos, fetchPhotos } = useContext(PhotosContext);
  const { theme } = useContext(ThemeContext);

  return <PhotosList photos={photos} fetchPhotos={fetchPhotos} theme={theme} />;
}
```

Rota codility

React Context/Redux



React Context/Redux

```jsx
import React, { Component, createContext } from "react";
import api from "./api";

const Context = createContext({});

export class PhotosProvider extends Component {
  constructor(props) {
    super(props);

    this.state = {
      photos: [],
      fetchPhotos: this.fetchPhotos
    };
  }

  render() {
    return (
      <Context.Provider value={this.state}>
        {this.props.children}
      </Context.Provider>
    );
  }

  fetchPhotos = async () => {
    const photos = await api.fetchPhotos();
    this.setState(state => ({ photos: [state.photos, ...photos] }));
  };
}

export default Context;
```

```jsx
import api from "../contexts/PhotosContext/api";

export function addPhotos(photos) {
  return {
    type: "photos/ADD_PHOTOS",
    payload: {
      photos
    }
  };
}

export function fetchPhotos() {
  return async dispatch => {
    const photos = await api.fetchPhotos();
    dispatch(addPhotos(photos));
  };
}

function getInitialState() {
  return {
    photos: []
  };
}

export default function photosReducer(state = getInitialState(), action) {
  const { type, payload } = action;
  switch (type) {
    case "photos/ADD_PHOTOS":
      return Object.assign({}, state, {
        photos: [...state.photos, payload.photos]
      });
    default:
      return state;
  }
}
```

React Context/Redux + components

```
import React, { useContext } from "react";
import PhotosList from "../components/PhotoList";
import PhotosContext from "../contexts/PhotosContext";

export default function PhotoListUseContext() {
  const { photos, fetchPhotos } = useContext(PhotosContext);

  return <PhotosList photos={photos} fetchPhotos={fetchPhotos} />;
}
```

```
import connect from "react-redux";
import PhotosList from "../components/PhotoList";
import { fetchPhotos } from "../redux/Photos";

const mapStateToProps = state => ({ photos: state.photos });

const mapDispatchToProps = () => ({ fetchPhotos });

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(PhotosList);
```

Rota

Testowanie! 

Context recap

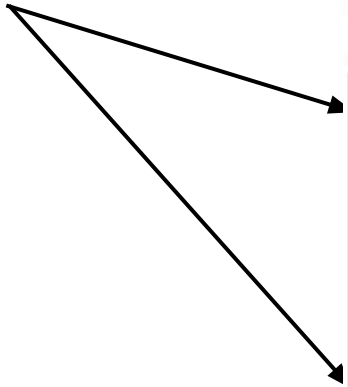
`import api from "./api"`

```
import React, { Component, createContext } from "react";
import api from "./api";

const Context = createContext({});

export class PhotosProvider extends Component {
  constructor(props) {
    super(props);

    this.state = {
      photos: [],
      fetchPhotos: this.fetchPhotos
    };
  }

  render() {
    return (
      <Context.Provider value={this.state}>
        {this.props.children}
      </Context.Provider>
    );
  }

  fetchPhotos = async () => {
    const photos = await api.fetchPhotos();
    this.setState(state => ({ photos: [state.photos, ...photos] }));
  };
}

export default Context;
```

Rota codility

Testing: Inner Component

`api` as a prop



```
import React, { Component, createContext } from "react";
import api from "./api";

const Context = createContext({});

export class PhotosProviderInner extends Component {
  static propTypes = {
    api: PropTypes.object
  };

  constructor(props) {…
  }

  render() {…
  }

  fetchPhotos = async () => {
    const photos = await this.props.api.fetchPhotos();
    this.setState(state => ({ photos: [state.photos, ...photos] }));
  };
}

export function PhotosProvider(props) {
  return <PhotosProviderInner api={api} {...props} />;
}

export default Context;
```

Rota codility

Testing: Redux

```
import configureMockStore from 'redux-mock-store'
import thunk from 'redux-thunk'
import * as actions from '../../actions/TodoActions'
import * as types from '../../constants/ActionTypes'
import fetchMock from 'fetch-mock'
import expect from 'expect' // You can use any testing library

const middlewares = [thunk]
const mockStore = configureMockStore(middlewares)

describe('async actions', () => {
  afterEach(() => {
    fetchMock.restore()
  })

  it('creates FETCH_TODOS_SUCCESS when fetching todos has been done', () => {
    fetchMock.getOnce('/todos', {
      body: { todos: ['do something'] },
      headers: { 'content-type': 'application/json' }
    })

    const expectedActions = [
      { type: types.FETCH_TODOS_REQUEST },
      { type: types.FETCH_TODOS_SUCCESS, body: { todos: ['do something'] } }
    ]
    const store = mockStore({ todos: [] })

    return store.dispatch(actions.fetchTodos()).then(() => {
      // return of async actions
      expect(store.getActions()).toEqual(expectedActions)
    })
  })
})
```

Rota codility

Testing: Context

```
import { mount } from "enzyme";
import React from "react";
import { PhotosProviderInner } from "./PhotosContext";

describe("PhotosContext", () => {
  it("Should properly update photos after fetching", async () => {
    const api = {
      fetchPhotos: () => [{ id: 37, title: "Back in Black" }]
    };

    const wrapper = mount(<PhotosProviderInner api={api} />);
    await wrapper.instance().fetchPhotos();
    const { photos } = wrapper.state();

    expect(photos).toMatchSnapshot();
  });
});
```

Rota codility

Projekt + prezentacja (PDF):

`github.com/tfiechowski/meetjs-wdi-context`

plzzz: bit.ly/meetjs-wdi-context

tomasz.fiechowski@outlook.com

Rota codility