REACT

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

REACT – REDUX

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- REDUX
  - O **Redux** é um contêiner de estado previsível projetado para ajudá-lo a escrever aplicativos JavaScript que se comportam de maneira consistente em ambientes cliente, servidor e nativo e são fáceis de testar.

É usado principalmente como uma ferramenta de gerenciamento de estado com **React**,

O Redux permite que você gerencie o estado de seu aplicativo em um único local e mantenha as alterações em seu aplicativo mais previsíveis e rastreáveis.;

REACT – REDUX

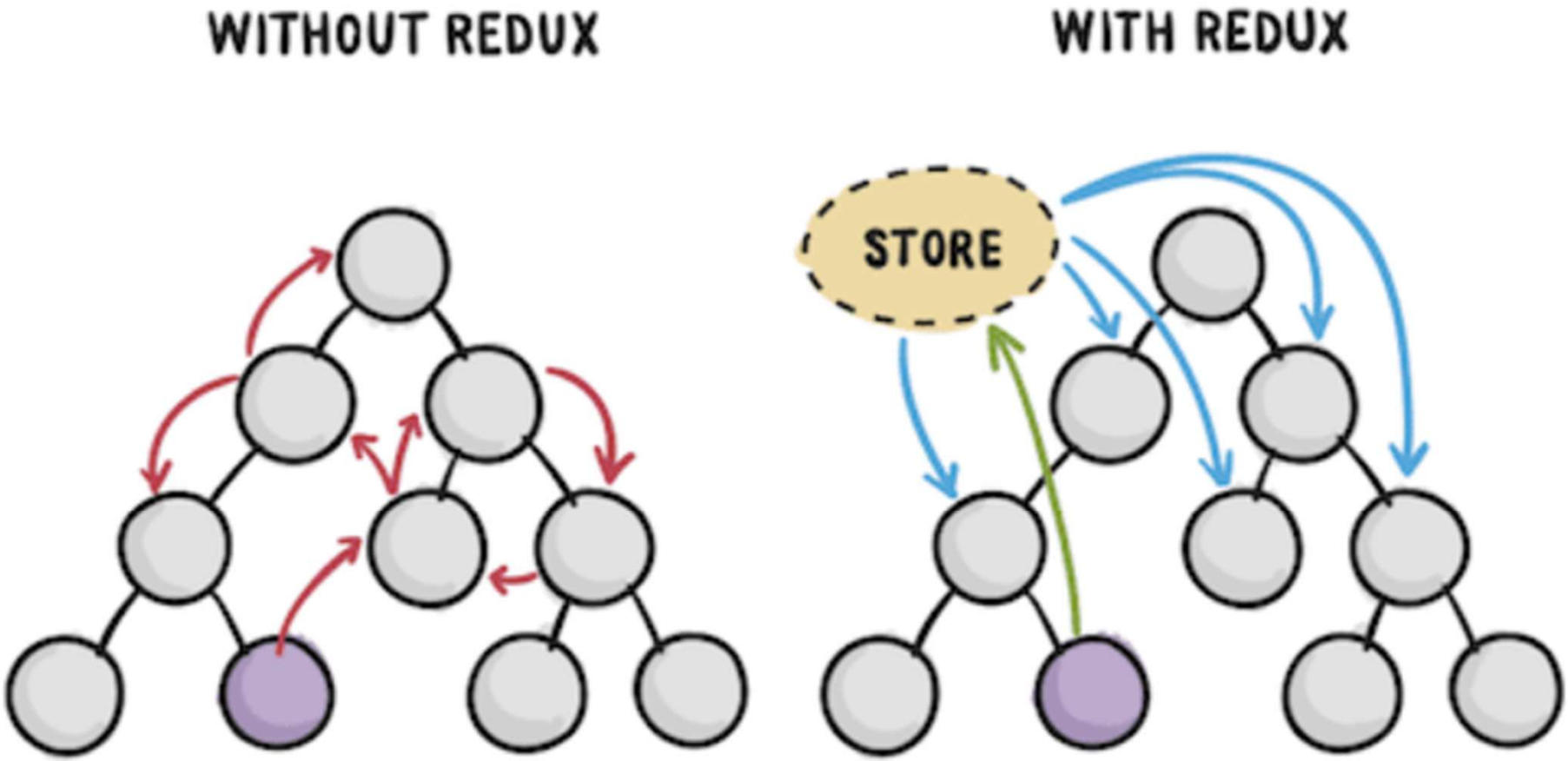
2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

- REDUX
  - O Redux é baseado nos três princípios a seguir:

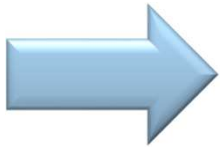
- **A store é a única fonte confiável**. Existe uma única store onde seu aplicativo mantém o **state**.
- **O state é somente leitura**. Quando uma ação é emitida, a função redutora não atualiza, mas clona o estado atual e o atualiza com base na ação.
- **Mudanças de state são feitas com funções puras**. Você escreve as funções redutoras que executam uma ação no objeto de estado atual e retorna um novo estado.

REACT

2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207



REACT – REDUX



2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

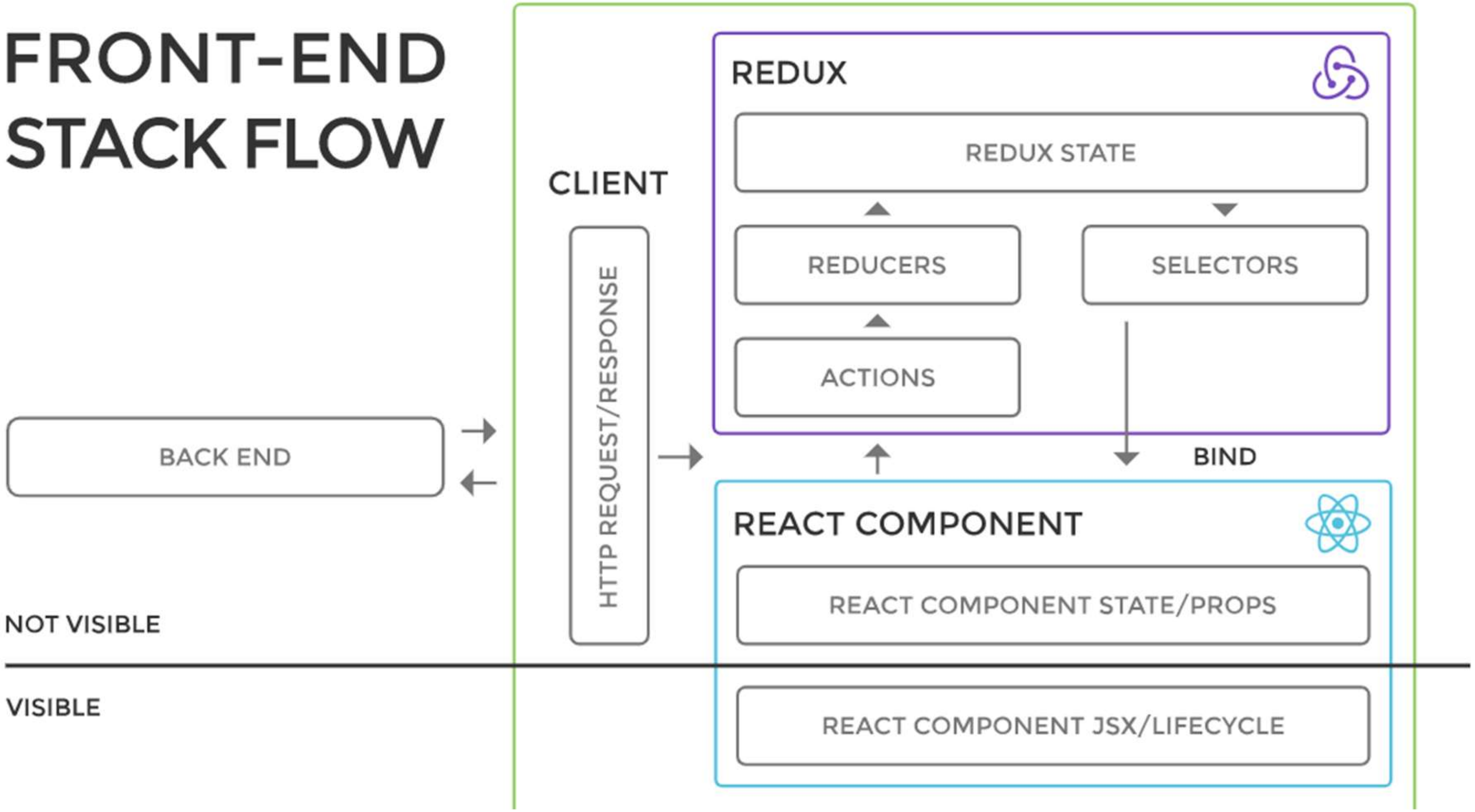
- Instalação

npm install react-redux @reduxjs/toolkit

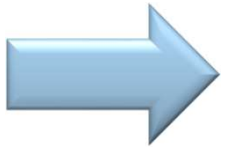
yarn add react-redux @reduxjs/toolkit

REACT – REDUX

- REACT com REDUX



REACT – REDUX



2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- REACT sem REDUX

```
import React, { useState } from 'react';

const Counter = () => {
  const [count, setCount] = useState(0);

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

export default Counter;
```

REACT – REDUX

2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f48696768207

- REACT com REDUX

import React, { useState } from 'react';

```
1   import { Provider, useSelector, useDispatch } from "react-redux";
2   import { createStore } from "redux";
3
4   // Tipos
5   type Action = { type: "INCREMENT" } | { type: "DECREMENT" };
6
7   // Reducer
8 > const counterReducer = (state = 0, action: Action): number => {···
17  };
18
19  // Store
20  const store = createStore(counterReducer);
21
22  // Componente
23 > function Counter() {···
34  }
35
36  // App principal
37  function App() {
38    return (
39      <Provider store={store}>
40        <Counter />
41      </Provider>
42    );
43  }
44
45  export default App;
```

export default connect(mapStateToProps, mapDispatchToProps)(Counter);

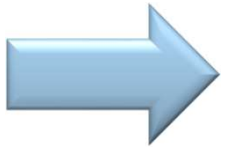
Provider: componente que conecta o Redux à árvore do React.

useSelector: hook que permite acessar dados da store.

useDispatch: hook para despachar ações que modificam o estado.

createStore: função (hoje obsoleta) que cria a store Redux.

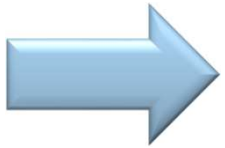
REACT – REDUX



2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

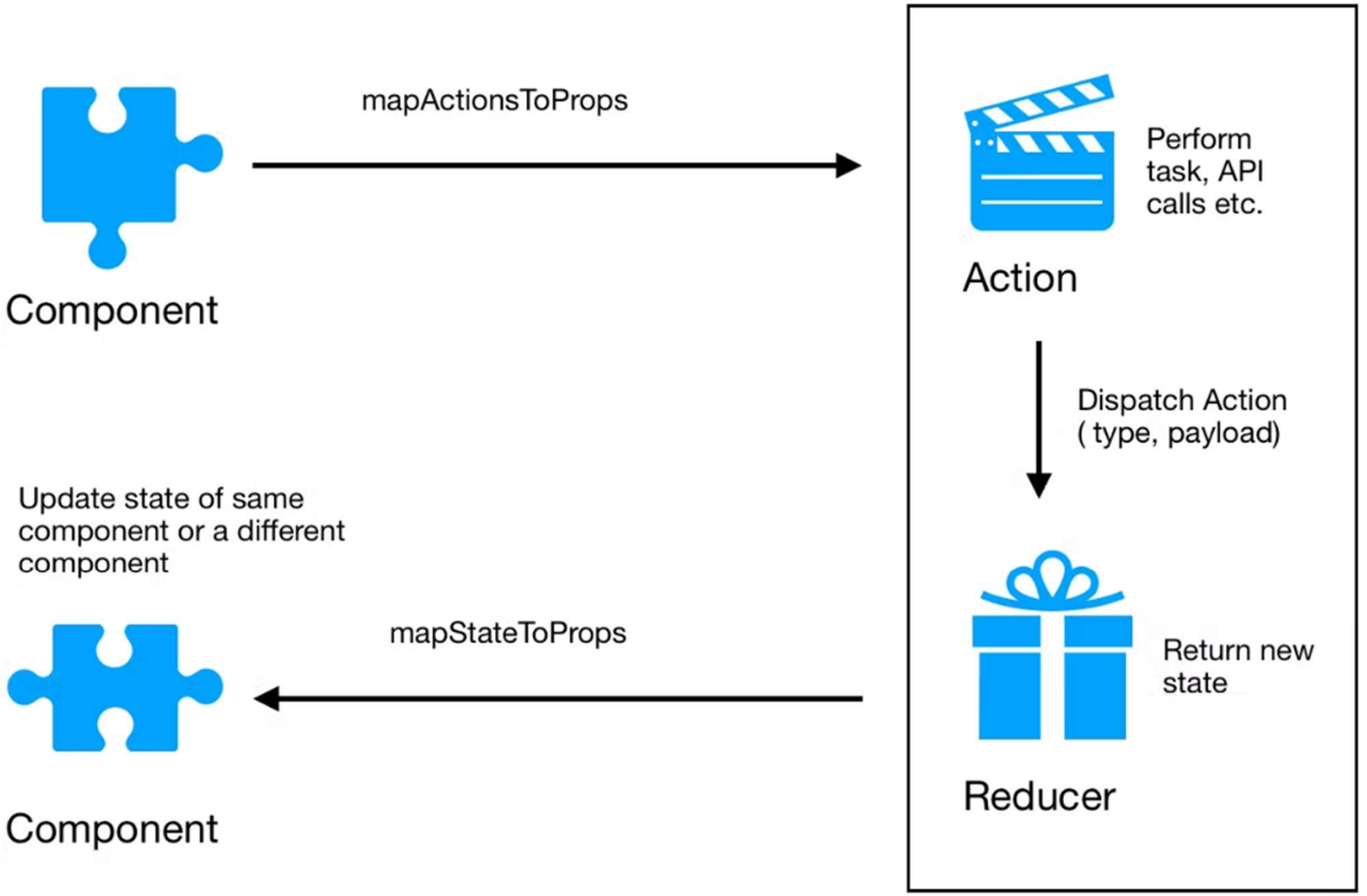
```
6
7   // Reducer
8   const counterReducer = (state = 0, action: Action): number => {
9     switch (action.type) {
10      case "INCREMENT":
11        return state + 1;
12      case "DECREMENT":
13        return state - 1;
14      default:
15        return state;
16    }
17  };
```

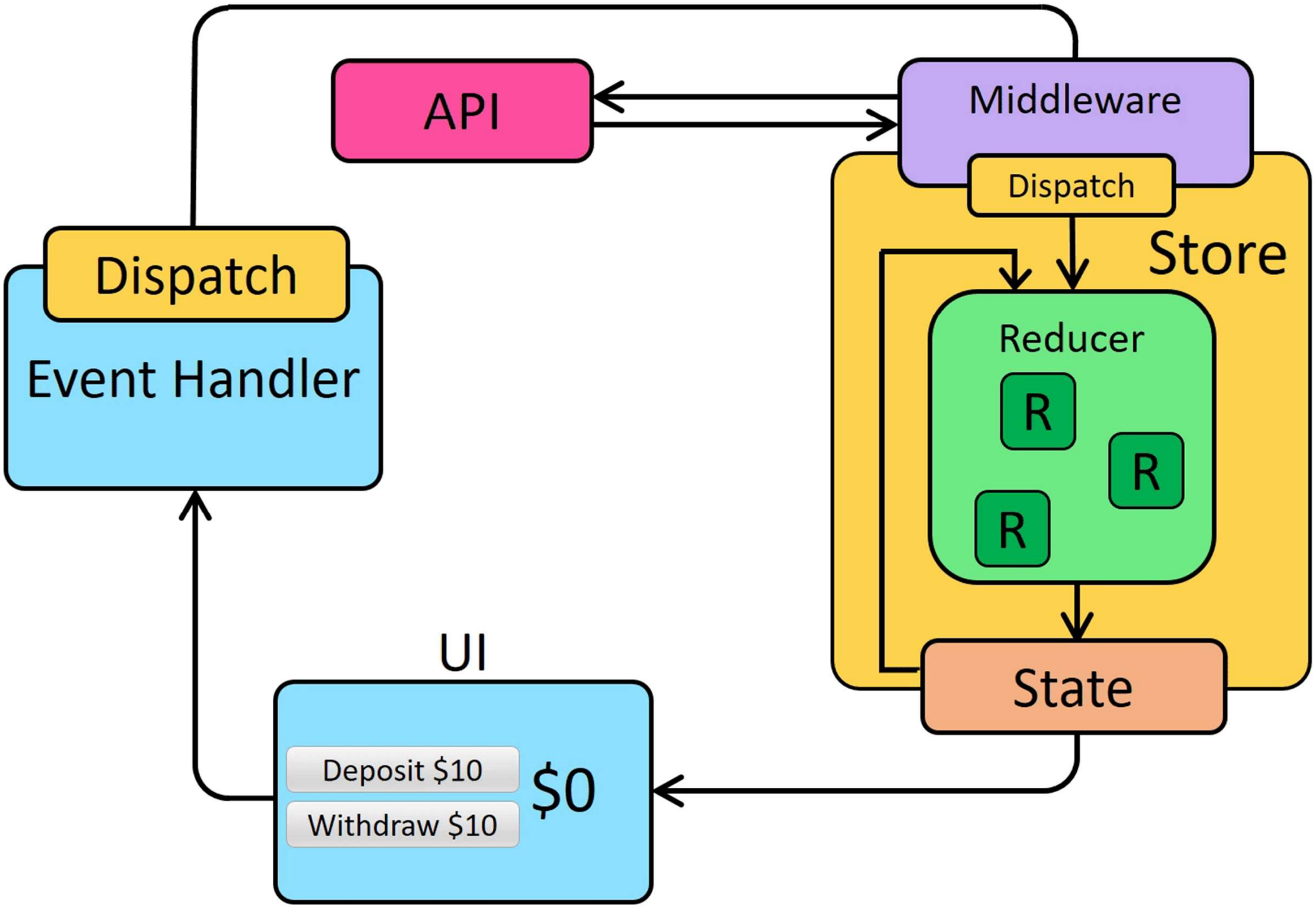
REACT – REDUX



2f486967682070657266f726d616e63652e2044656e697665726564e2f486967682070657266f726d616e63652e2044656e697665726564e2f486967682070657266f726d616e63652e2044656e697665726564e2f486967682070657266f726d616e63652e2044656e697665726564e2f486967682070657266f726d616e63652e2044656e697665726564e2f486967682070657266f726d616e63652e2044656e697665726564e2f48696768207

React-Redux Flow

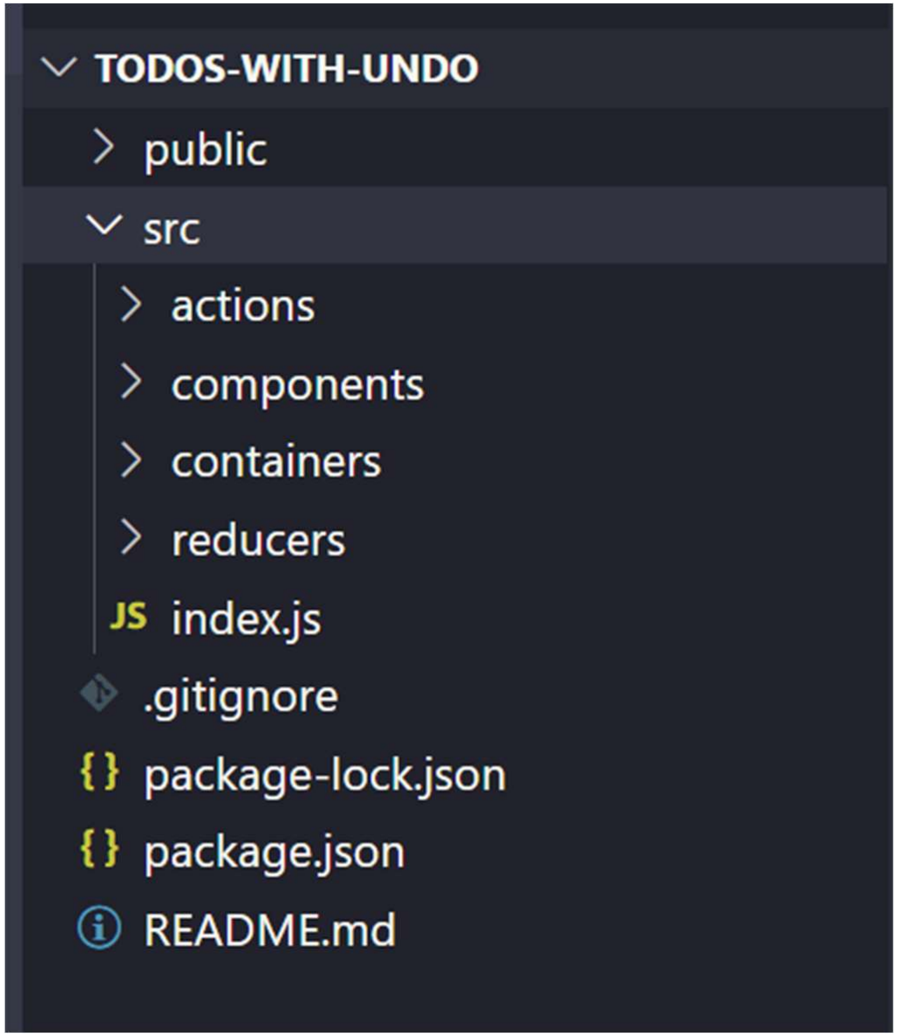



API
Middleware
Dispatch
Store
Reducer
R
R
R
State
Dispatch
Event Handler
UI
Deposit $10
Withdraw $10
$0

REACT – REDUX

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

- REACT com REDUX



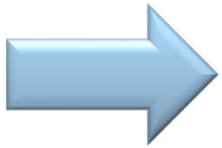
STORE
É um container imutável, isto é, não há alteração dele, e sim evolução, que armazena e centraliza o estado global da aplicação. Com isso, podemos dizer que é o conjunto de estados da aplicação centralizados/reunidos em um apenas um lugar.

ACTION
Tecnicamente, uma action é um objeto que possui, obrigatoriamente, um atributo nomeado ***type*** que indica que ação é. Pode ter dados associados ou não, mas geralmente possuem. Esses dados são reunidos em único atributo chamado de ***payload***.

REDUCERS
São funções puras (funções que não geram efeitos colaterais, isto é, para a mesma entrada, temos a mesma saída) com a capacidade de disparar eventos e que podem alterar um atributo da store, evoluindo o estado global da aplicação.

SELECTORS
Uma "função seletora" é qualquer função que aceita o estado da store Redux (ou parte do estado) como um argumento e retorna dados baseados nesse estado.

REACT – REDUX



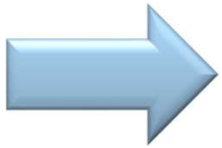
2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070

REDUCE

- O **reduce** é uma função que sempre retorna um novo estado, nunca modificando o estado existente.
- Seu resultado depende apenas dos argumentos fornecidos (estado anterior e ação).

```
const counterReducer = (state = initialState, action) =>
                                    switch (action.type) {
                                                case 'INCREMENT':
                                                        return {
                                                         ...state,
                                                            count: state.count + 1
                                                             };
                                                case 'DECREMENT':
                                                        return {
                                                         ...state,
                                                            count: state.count – 1
                                                          };
                                                default:
                                                       return 1
                                    }
}
```

REACT – REDUX



2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f48696768207

ACTION

- Define que ações são aceitas pelo reducer.

```
type Action = { type: 'INCREMENT' } | { type: 'DECREMENT' };
```

REACT – Atividade

- Desafio “To do”
  - Adicionar os botões desfazer (undo) e refazer (redo).
  - Ao clicar no botão “desfazer” a tarefa incluída será excluída.
  - Ao clicar no botão “refazer” a tarefa excluída volta à aparecer.

Questions and Comments

2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207065726666f726d616e63652e2044656c6976657265642e2f48696768207

- What questions or comments do you have?