UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

React 04

Two way binding

Two way binding

Permite “enlazar” un input a una propiedad del estado, permitiendo que al modificar la propiedad se modifique el input, y al modificar el input se modifique la propiedad

Two way binding

```
state = {
    nombre: ''
}

constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
}


handleChange(e) {
    this.setState({nombre: e.target.value});
}

render() {
    return (
        <div className="row">
            <input type="form" value={this.state.nombre} onChange={this.handleChange} />
        </div>
    )
}
```

Actualización del estado (lista)

Agregar un elemento en la lista

```
this.setState({
    listado: [
        ...this.state.listado,
        nuevoElemento
    ]
})
```

nuevoElemento es el elemento a agregar

...this.state.listado crea una copia de la lista (Ver documentación)

Borrar un elemento de la lista

```
const posicion = this.state.listado.findIndex(item => {
    return item.id === id;
})
this.setState({
    listado: [
        ...this.state.listado.slice(0, posicion),
        ...this.state.listado.slice(posicion+1),
    ]
})
```

slice retorna una copia de la lista (Ver documentación)

Actualizar un elemento de la lista

```
const posicion = this.state.listado.findIndex(item => {
    return item.id === id;
})
this.setState({
    listado: [
        ...this.state.listado.slice(0, posicion),
        copiarElementoModificado,
        ...this.state.listado.slice(posicion+1),
    ]
})
```

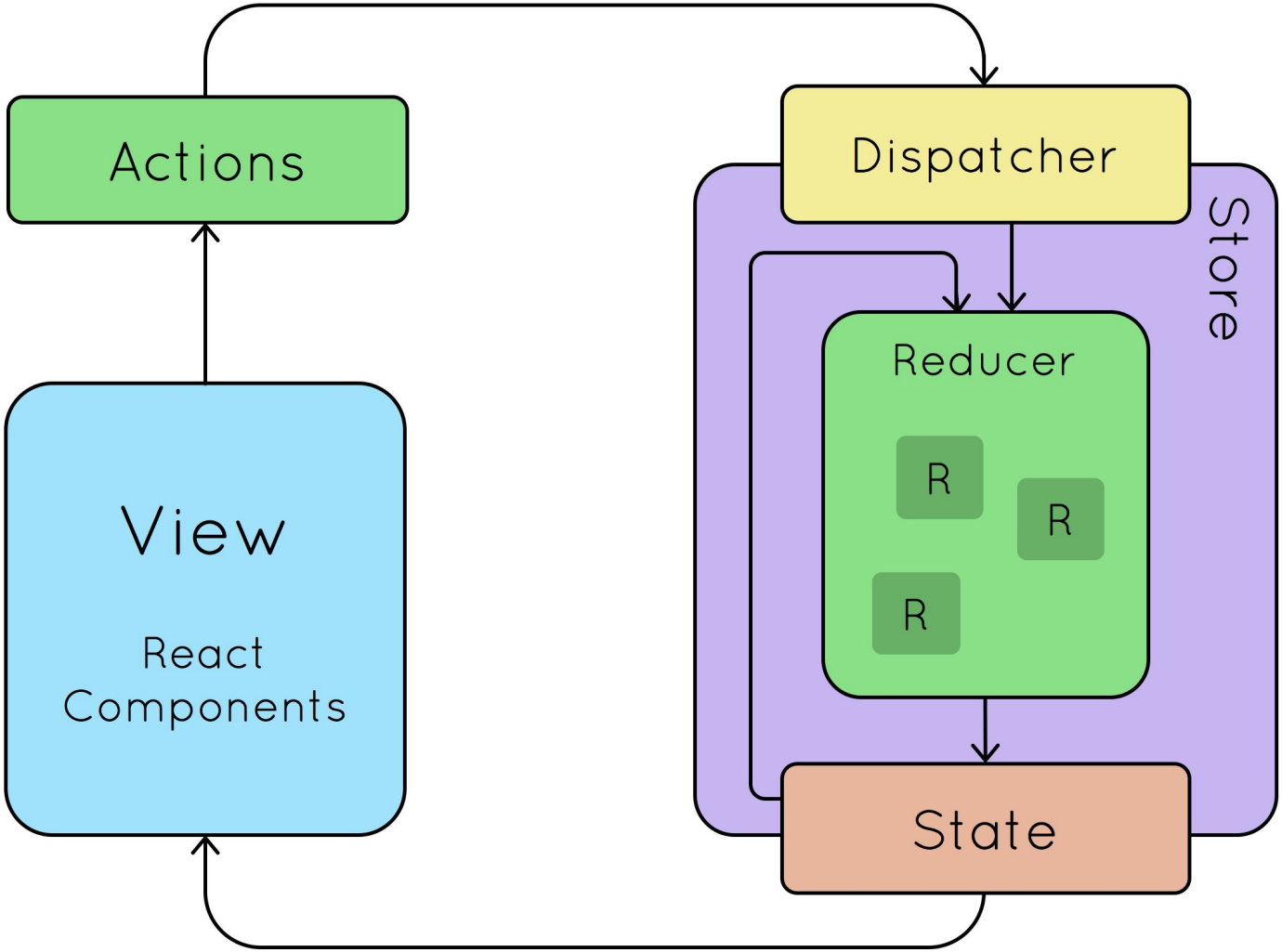
Manejo de estado

Manejo del estado

Cuando la aplicación comienza a ser más grande y compleja, cada vez es más difícil poder manejar el estado

Manejo del estado

Redux es un paquete que nos facilita el manejo del estado en React


Actions
Dispatcher
Store
Reducer
R
R
R
View
React
Components
State

Principios

- Todo el estado de la aplicación está guardado es un store
- El estado es de solo lectura (solo se puede actualizar por medio de acciones)
- Los cambios de estados se realizar en funciones (reducers)

Estado guardado en un store

Permite mantener todo el estado en un lugar

Permite persistir el estado en el servidor

Permite restaurar un estado de la aplicación desde el servidor

Actualización del estado

Para actualizar el estado debemos emitir (dispatch) una acción

Generalmente indicamos que acción deseamos realizar y los datos necesarios para realizar dicha acción

Reducers

Son las funciones que reciben

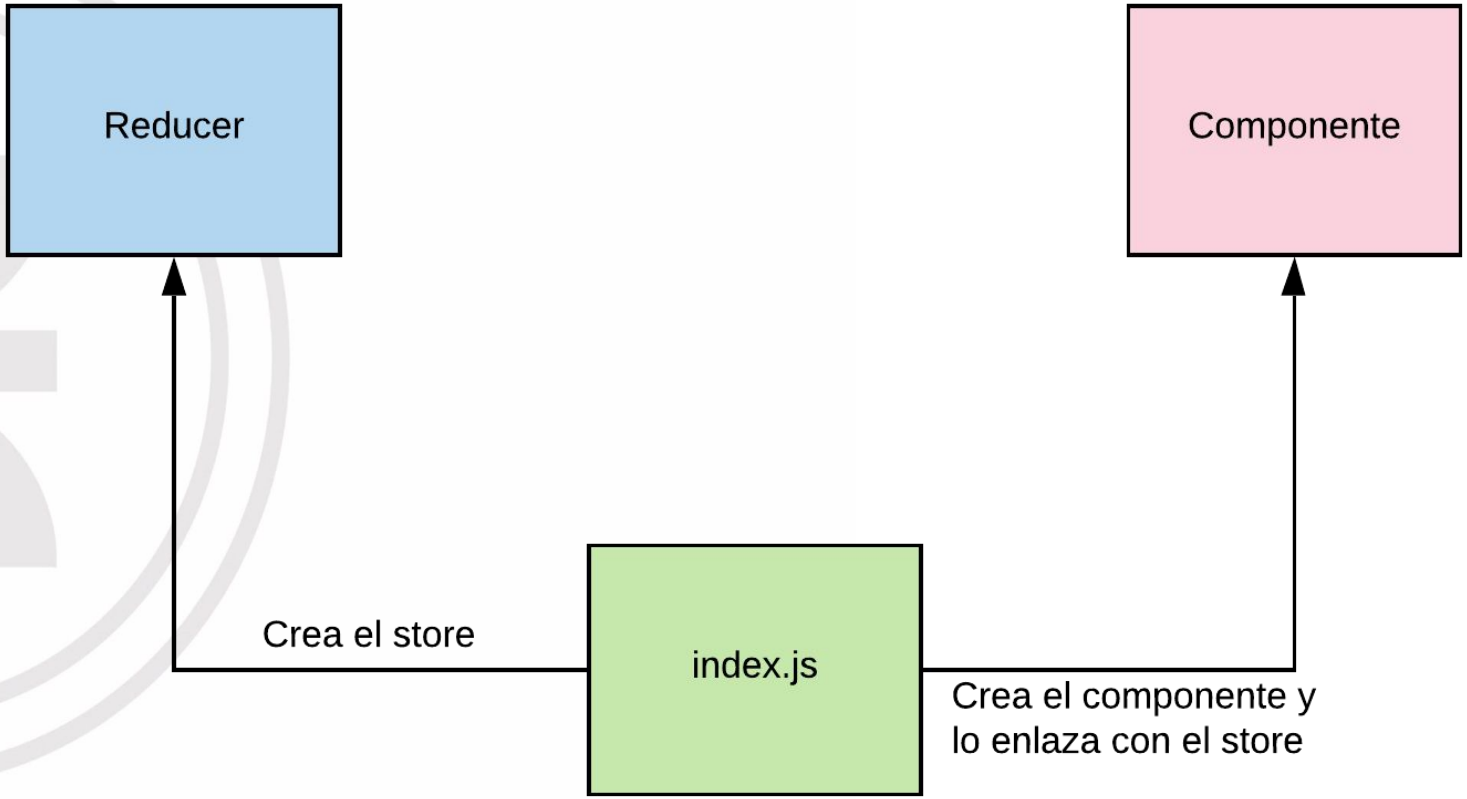
- El estado anterior
- La acción a realizar
- Datos necesarios para realizar la acción

Y actualizan el estado

Instalación

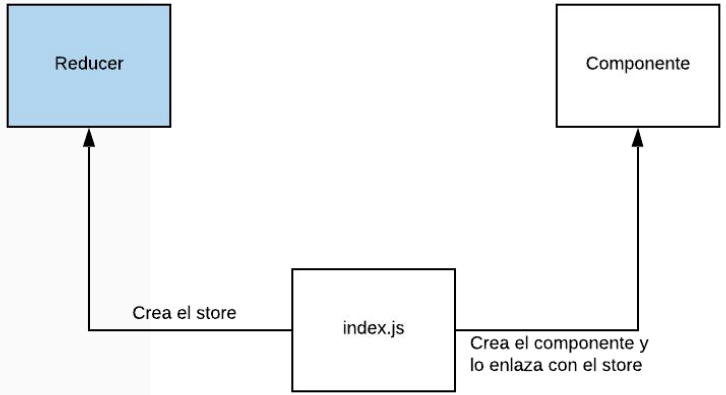
```
npm install --save redux
npm install --save react-redux
```

Pasos


Reducer
Componente
Crea el store
index.js
Crea el componente y
lo enlaza con el store

Pasos - Crear reducer

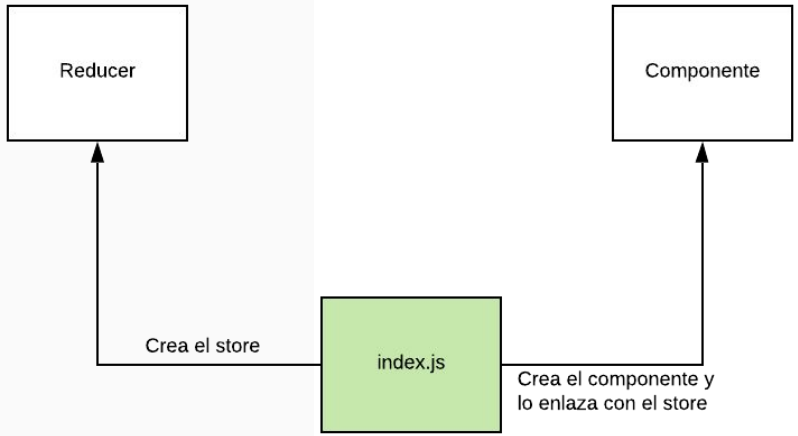
```
export default function(estado=0, accion) {
    switch (accion.type) {
        case 'INCREMENTAR':
            return estado+1
        case 'DECREMENTAR':
            return estado-1;
        default:
            return estado;
    }
}
```



src/ContadorReducer.js

Pasos - Componente inicial

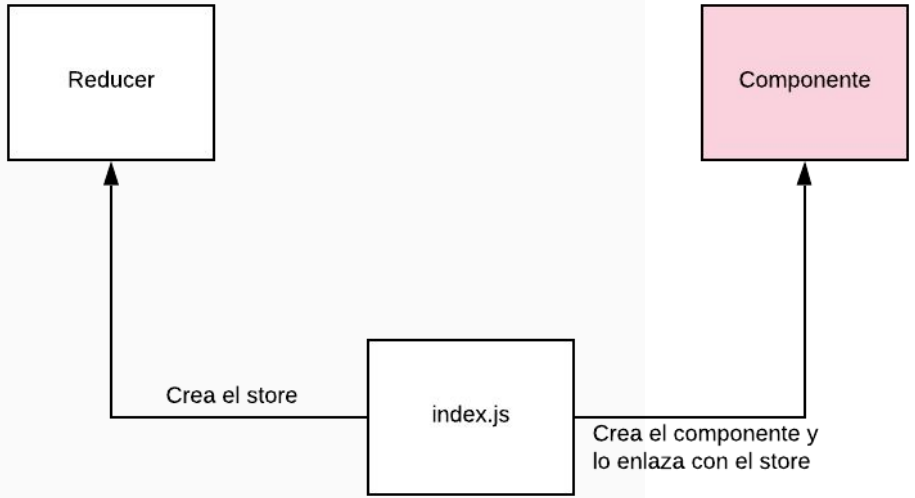
```
// REDUX
import {Provider} from 'react-redux';
import  { createStore } from 'redux';
import ContadorReducer from './reducers/ContadorReducer'

var store = createStore(ContadorReducer);

ReactDOM.render(
    <Provider store={store}>
        <Contador />
    </Provider>,
document.getElementById('root'));
```



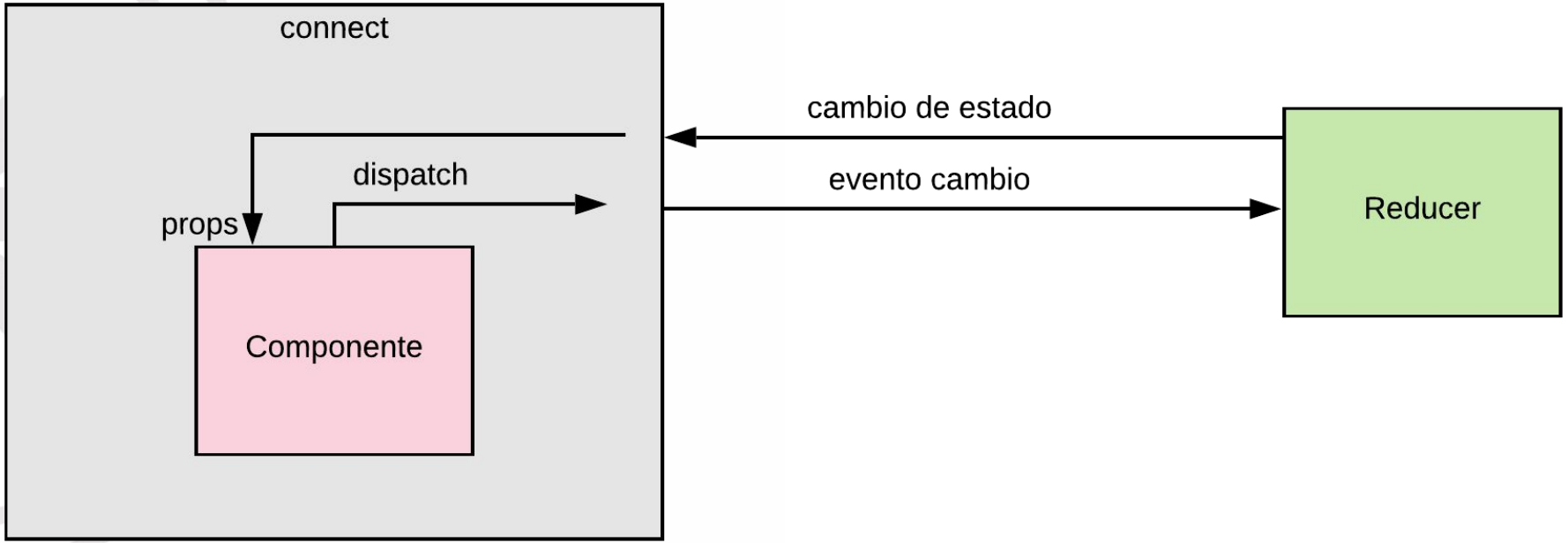
index.js

Pasos - Componente

```
import { connect } from 'react-redux';
class Contador extends React.Component {
...
}

const mapEstadoAProps = (estado) => {
    return {
        estado: estado
    }
}

const mapAccionesAProps = (dispatch) => {
    return {
        onIncrementar: () => dispatch({ type: 'INCREMENTAR' }),
        onDecrementar: () => dispatch({type: 'DECREMENTAR'}),
    }
}

export default connect(mapEstadoAProps, mapAccionesAProps)(Contador);
```



Comunicación Componente - Store



Ejemplo

Contador en React con Redux

https://github.com/cursos-utn/react_ejemplos/tree/master/06-redux-contador

Ruteo

Ruteo

Permite “enlazar” URLs a componentes

Las URLs no son pedidas al servidor

Ruteo

Para poder utilizar el ruteo, debemos instalar el componente react-router y react-router-dom

```
npm install --save react-router react-router-dom
```

Uso

```
import { BrowserRouter as Router, Route } from 'react-router-dom'
...
<Router>
    <Route exact path="/" component={App} />
    <Route path="/dos" component={AppDos} />
    <Route path="/tres" component={AppTres} />
</Router>
```

Ejemplo

Ruteo

https://github.com/cursos-utn/react_ejemplos/tree/master/07-rutas

Redux + Ruteo

Redux + Ruteo

Podemos incorporar Redux al ruteo

```
<Provider store={store}>
    <Router>
        <Route exact path="/" component={Home} />
        <Route exact path="/todo" component={ToDo} />
    </Router>
</Provider>
```

index.js

Redirección desde código

```
irAPaginaToDo() {
    this.props.history.push('/todo');
}
```

No hace falta hacer un map del estado history

Ejemplo

Ruteo + Redux

https://github.com/cursos-utn/react_ejemplos/tree/master/05-todo-step-by-step/06