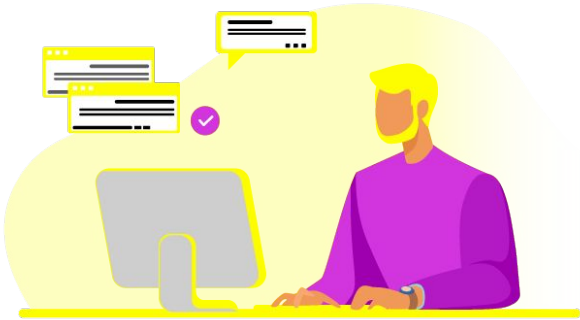
REACT REDUX

OBJETIVOS DE LA LECTURE

Al finalizar esta lecture estarás en la capacidad de...

- Conectar React y Redux
- Gestionar el estado global desde los componentes
- Renderizar información del estado global en tu componentes

RECAPITULACIÓN



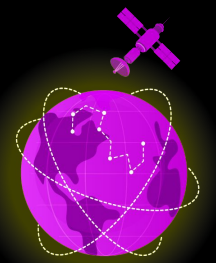
Agenda de esta clase

- **Componentes y Redux**
- **Conexión**
- **Resumen ideas centrales de la clase**
- **Explicación de la HW**

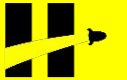
¿Dónde estamos?

1. DOM Avanzado
2. ES6
3. AJAX
4. Bundlers
5. React Intro
6. React Estados LifeCycle I
7. React Estados LifeCycle II
8. React Routing
9. React Forms
10. Redux

12. React Redux

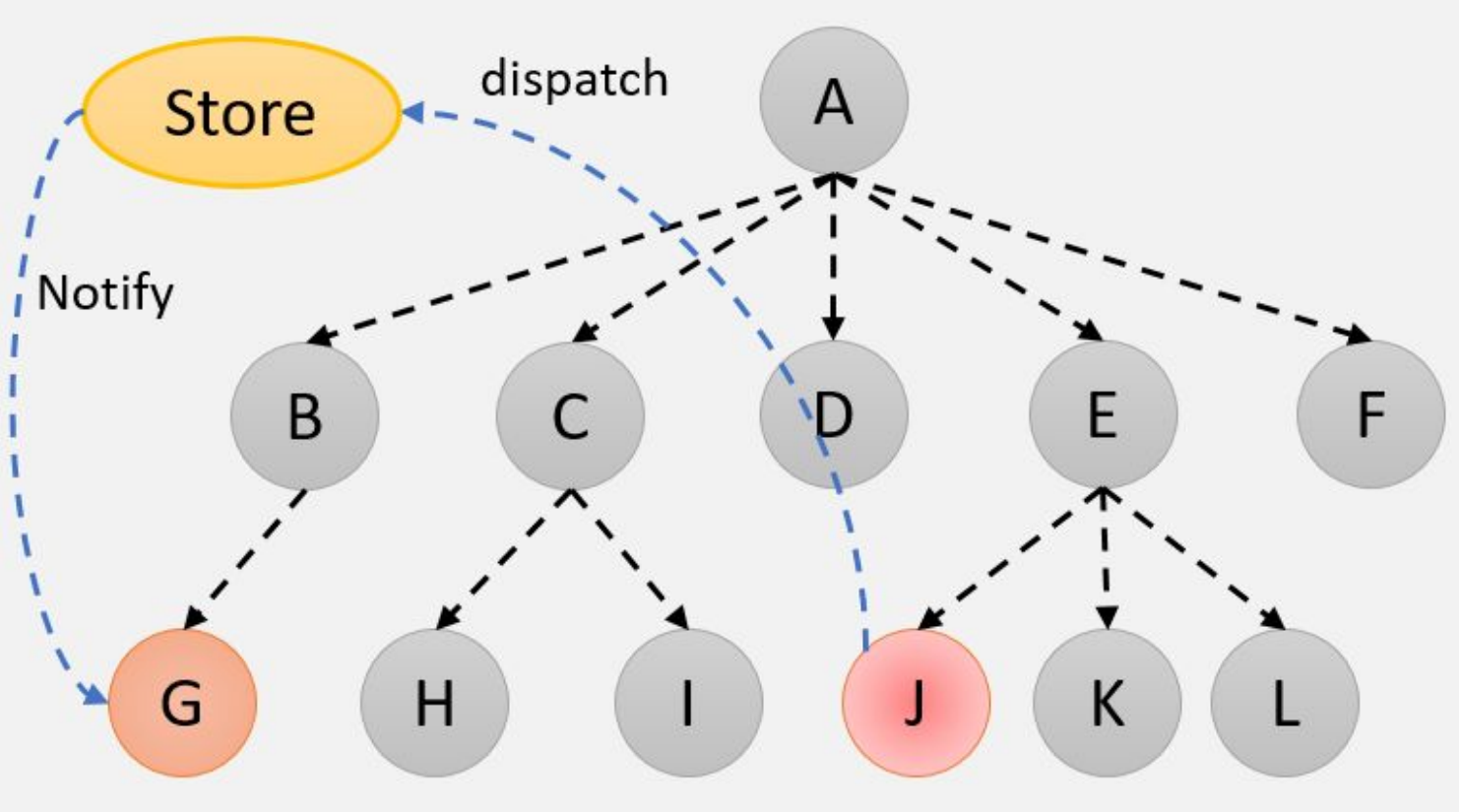


¡COMENCEMOS!



COMPONENTES Y REDUX

Suscripción



Presentacionales
dumb

- ✔ Cómo se ven las cosas.
- ✔ Trabaja con sus propiedades.
- ✔ Generalmente no tienen estado propio.

Containers
smart

- ✔ Cómo funcionan las cosas.
- ✔ Poco o anda del DOM.
- ✔ Sin estilos.
- ✔ Provee datos.
- ✔ Invoca acciones.

Presentacionales vs. Containers

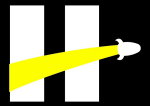
Separarlos de esta manera trae varias ventajas

- ✔ Separa los problemas de lógica de lo presentacional.
- ✔ Obtenemos componentes reutilizables.
- ✔ Localizamos la complejidad en los containers.

Presentacionales vs. Containers

Comparemos

CARACTERÍSTICA	PRESENTACIONALES	CONTAINERS
Propósito	Muestran cómo se ven las cosas	Hacen funcionar cosas (buscan datos, actualizan estados, etc...)
Sbe de Redux	NO	SI
Para leer datos	Lee propiedades	Se suscribe al estado de Redux
Para cambiar datos	Invoca callbacks en sus propiedades	Envía acciones a Redux



<DEMO />

¡Vayamos a codear!

¿ P R E G U N T A S ?

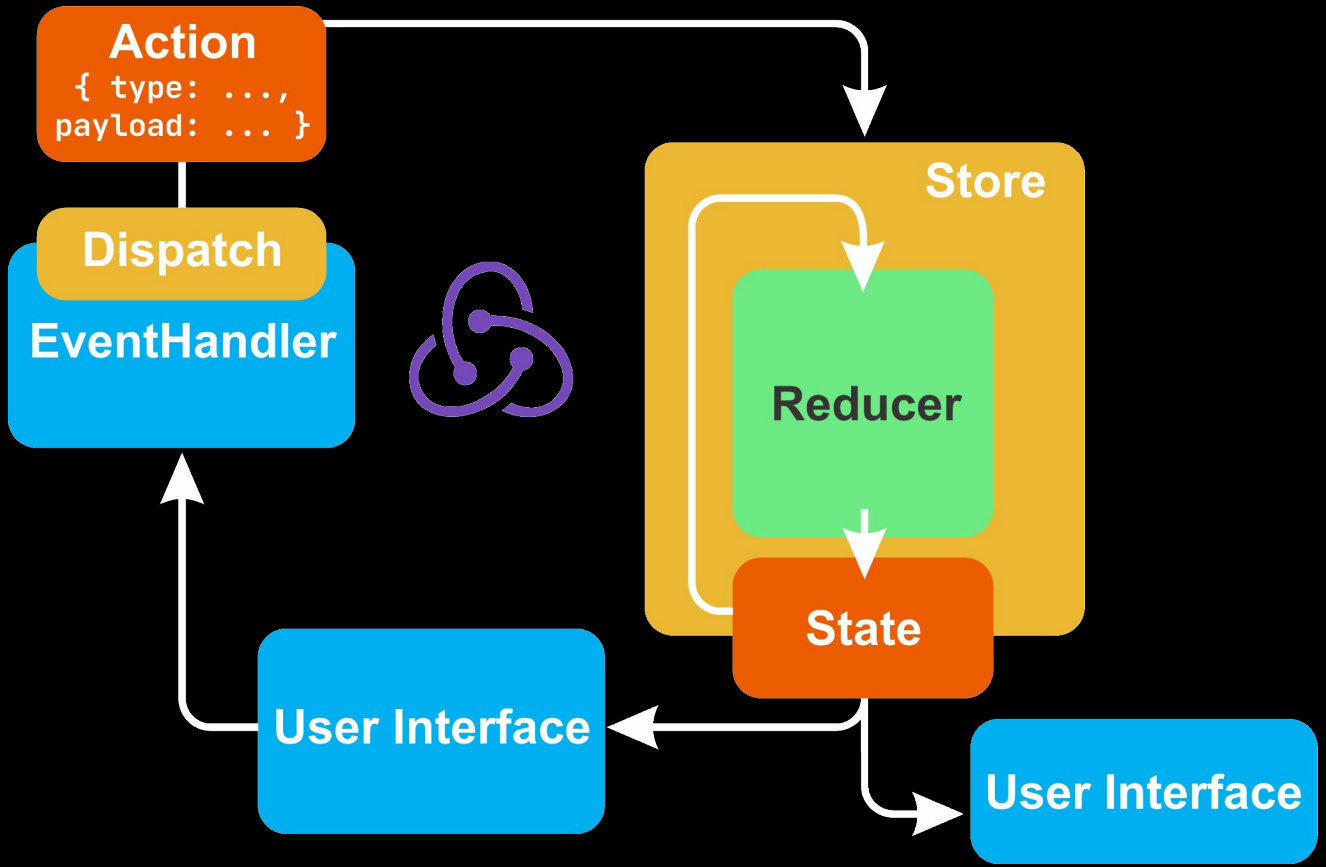
CONEXIÓN

Provider

Lo que nos recomienda Redux es usar un *<Provider>* que permite que el Store esté disponible para todos los Container de nuestra App, sin pasarla explícitamente.

```
import { Provider } from 'react-redux';
import store, { history } from './store.js';

const router = (
  <Provider store={store}>
      <Route path='/' element={<App />} />
      <Route path='/picture' element={<PhotoGrid />} />
      <Route path='view/:postId' element={<Single />}/>
  </Provider>
);
```


Action
{ type: ...,
payload: ... }
Dispatch
EventHandler
Store
Reducer
State
User Interface
User Interface

Conectar un componente a Redux

Disponemos de la función ***connect***, la cual incluirá al componente en el flujo de datos de Redux.

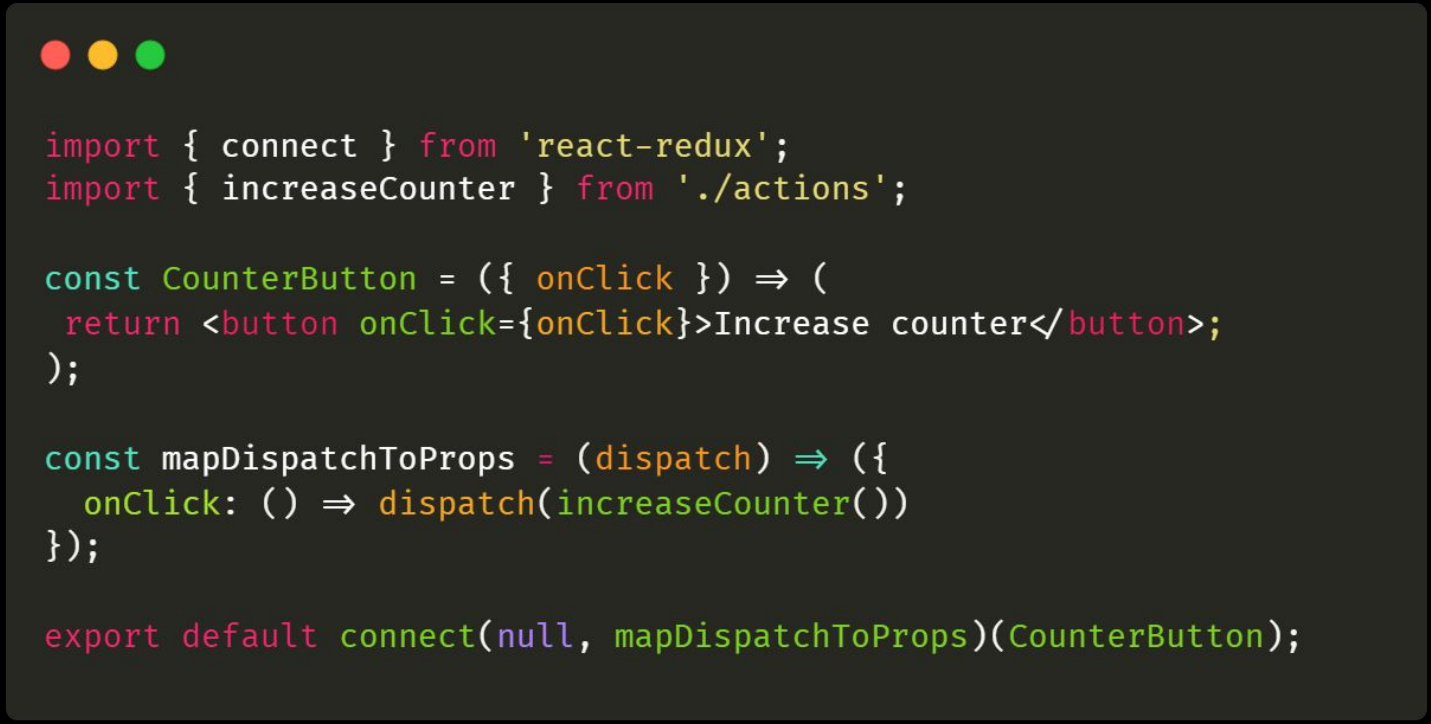
```
import { connect } from 'react-redux';

function Card(props) {
  return (<div>
        <h1>{props.name}<h1/>
        <h1>{props.lastName}<h1/>
        <img src={props.image} alt='' />
    <div/>);
}

export default connect(null, null)(Card);
```

mapDispatchToProps

Es una función que conecta acciones de Redux con componentes de React como propiedades.



```
import { connect } from 'react-redux';
import { increaseCounter } from './actions';

const CounterButton = ({ onClick }) => (
 return <button onClick={onClick}>Increase counter</button>;
);

const mapDispatchToProps = (dispatch) => ({
  onClick: () => dispatch(increaseCounter())
});

export default connect(null, mapDispatchToProps)(CounterButton);
```

mapStateToProps

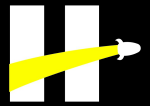
Es una función que conecta el estado de Redux con las props de un componente, permitiendo que éste acceda y utilice el estado.

```
import { connect } from 'react-redux';

const CounterDisplay = ({ counter }) => (
 return <p>The current counter value is: {counter}</p>
);

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

export default connect(mapStateToProps, null)(CounterDisplay);
```



<DEMO />

¡Vayamos a codear!

¿ P R E G U N T A S ?

Resumen

- **Presentacional |** Es aquel componente que se encarga de mostrar la información y la interfaz de usuario, pero no tiene lógica de negocio ni interactúa directamente con el estado de la aplicación.

- **Container |** Es aquel componente que se encarga de conectar un componente presentacional al store de Redux y proporcionarle los datos y las acciones necesarias para interactuar con el estado de la aplicación.

Resumen

- **Suscribir |** Suscribir un componente de React al estado de Redux significa que el componente estará conectado al store de Redux y recibirá automáticamente actualizaciones del estado cuando este cambie.

- **Connect |** Es una función utilizada para conectar un componente de React al store de Redux y mapear el estado y las acciones necesarias como propiedades del componente.

Resumen

- **mapDispatchToProps |** Es una función que se utiliza para asignar las acciones de Redux a las propiedades del componente, lo que permite al componente interactuar con el estado de la aplicación.

- **mapStateToProps |** Es una función que se utiliza para asignar una parte del estado de Redux a las propiedades del componente, lo que permite al componente acceder a los datos del estado de la aplicación.

- **Provider |** El componente **Provider** en React-Redux es utilizado para proporcionar el store de Redux a la aplicación React, lo que permite que los componentes conectados a Redux tengan acceso al estado de la aplicación y a las acciones.

PRÓXIMA LECTURE

REACT HOOKS

¿Alguien dijo **homework**?


HENRY