# ReactJS + Redux + ES6 Completo

* ¿Por qué elegir React JS?
  + Manejar de forma eficiente la interfaz
  + Gran ecosistema, su estabilidad y perfomance
  + SPA
* Ciclo de vida en react.

Montaje:

* + Constructor:
    - Es el primer método llamado por React y sucede antes de que se monte el componente.
  + Render:
    - 3Es el único método requerido, sirve para generar los elementos de React que van a ser mostrados en la pantalla.
  + ComponentDidMount

Actualización:

* Render
* ComponentDidUpdate:
  + Inmediatamente después de qué ocurre una actualización.

Desmontaje:

* ComponentWillUnmount
* Redux – Framework
  + Manipular el estado de la aplicación, es su única función.
  + Es estado es único y global
  + Es estado se modifica por acciones
  + El estado es inmutable, se generan copias del mismo.
* react-redux
* connect es una función para unir ambos fremeworks que espera dos parámetros funciones y retornara otra función.
* Smart Components - Containers => Componentes que tienen acceso al estado de la aplicación (**connect**).
* Dumb Components - Presentational Components
* **Reducer**
  + Funciones puras, depende solamente de los parámetros que le pasamos.
  + Evitar efectos colaterales.
  + No alterar el estado => state.prop = ‘Nuevo estado’.
  + Hacer una copia del estado => { …state, prop: ‘Nuevo valor’}.
  + **Connect (1,2)(Component)**
    - **1 Values => Inyecta los values como propiedades, se ponen nuevos valores de propiedades.**
    - **2 Func => Inyecta las funciones como propiedades, acciones para alterar el estado de la aplicación.**
  + Mantener un estado global y alterarlo por medio de las acciones.
  + Y una vez se altera el estado de la aplicación se ejecuta la renderización.
* **Redux**

1. npm install –save redux
2. npm install –save react-redux
3. store / index.js
   * import {createStore} from ‘redux’;
   * const reducers = state => state;
   * export const store = createStore(reducers, {});
4. src /index.js
   * import {Provider} from ‘react-redux’;
   * import {store} from ‘. / store’;
   * Envolver a <App/> en las etiquetas <Provider store={store}/>
5. actions / acción
   * export const acción = () => ({ type:’TIPO\_DE\_ACCION’, value: null });

6 – 8.

Contendor donde se va a importar la acción

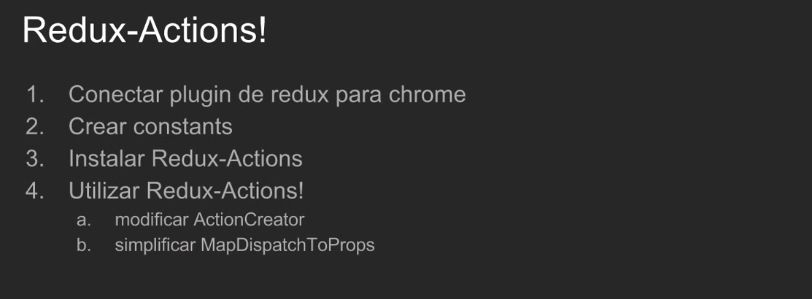
* + import {connect} form ‘react-redux’;
  + import {accion} from '. / .. /actions/accion;
  + const mapDispatchToProps = dispatch => (

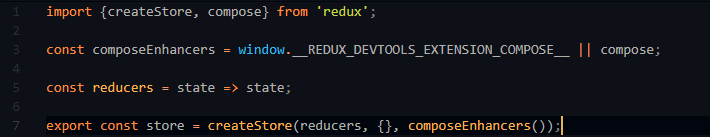
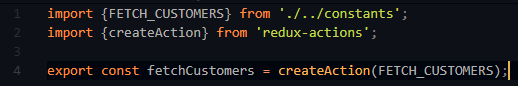
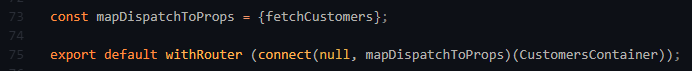
{

acción: () => dispatch(accion())

}

)

* + export default (connect(null, mapDispatchToProps)(componente)
  + agregar la acción al ComponentDidMount y validarla en los propTypes.
* Redux-actions

1. store / index.js
2. constants / index.js
   * const tipo\_de\_accion = ‘tipo\_de\_accion’;
3. npm install –save redux-actions
4. actions / accion
   * actionCreator
   * mapDispatchToProps

ó

* JSON-SERVER