Redux?
Redux is a predictable state container for javascript applications.

- **Why for Javascript application?**
  Redux is not tied to React. It can be used with any javascript-based UI library like React, Angular, Vue, and even with vanilla javascript. It doesn't matter what UI library we are using
- **State container?**
  Redux stores and manages the application state.

  What is the application state?
  In React, each component has its own state stored locally inside the component.

  Login Form component,
  ```
  state = {
        “name”: “jhon”,
        “Age”: 28,
        “email”: “jhon@gmail.com”
        “isSubmitted”: false
  }
  ```

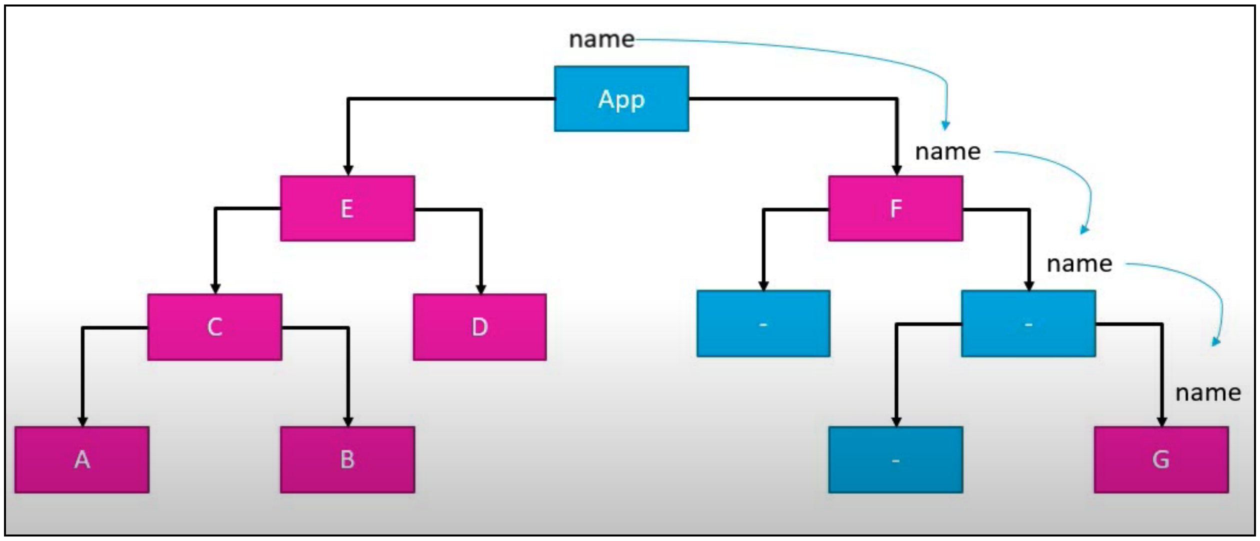
  Users component,
  ```
  state = {
        “users”: [],
  }
  ```

  The state of an application is represented by the state of each individual components

- **Predictable - predictable in which way?**
  In React, each component has a state and states always changes.
  Ex. ToDo list
  Task (init) → Task (pending) → Task (completed)
  In React, the state transition is transparent and predictable.

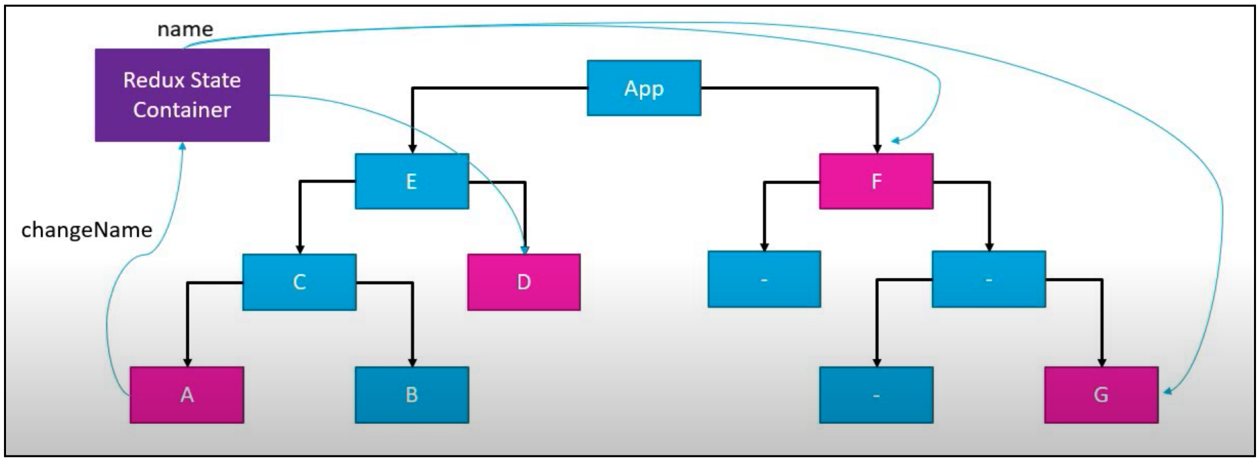
Why is Redux?
If we want to manage the application state in a predictable way then Redux can help.

Why React Redux?
Redux solves the problem of Props drilling and helps to manage the application state in a predictable way.



Requirements/cons:
- State lifting,
- Props passing
- Managing the state is not so easy in React
- Need to pass state through all the intermediate components.

But In Redux, the Application state is stored and managed outside of the component scope. When any component wants to update any state it communicates with the Redux State container (Store) and updates the state.
Once any state is updated the updates will pass to the components that only require it. No need to drill props.

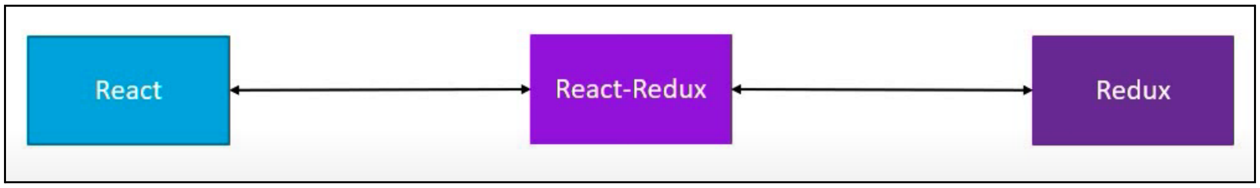


Context API vs Redux?
Context API + useReducer also provides the same functionalities that Redux offers then why do we need Redux?

Pros:
- Redux is an effect for larger-scale applications,
- Manage old application - we can think of React v1.0 in 2015 when Context Api was not there - Many application was built with Redux and currently exists. Many companies still hire based on their Redux skill.
- Elegance
- Performance

Cons:
- Lots of Boilerplate code,
- High learning curve.

What is React-Redux?
React Redux is the official Redux UI binding library for React. React-Redux offers some functionalities to connect Redux with React in an efficient way.
Using Redux directly in React is very tedious and difficult. So, to use Redux in React we will also need React-Redux.



Three key concepts related to Redux:
- **Store:** Store is a javascript object that contains application state.
- **Action:** Action defines the intension or tells what should performed to the application state,
- **Reducer:** Reducer actually caries out the action based on previous state and action. Reducer actually ties Action and Reducer.

Redux Principles:
- (Store) Application state should be stored in a object tree within a store
  That means we should maintain application state inside a single object that will be managed by the Redux.
- (Action) State transaction should only performed by emitting an action
  Changes in application state will be possible only by dispatching an action. Without sending an action on state transaction will be performed.
- (Reducer) The state transformation process should be define through a Pure Reducer.

  Pure Reducer / Pure function is a function that takes previous state and action as input and returns a New state.

```
Reducer (previousState, action) → New State
```

Being a pure function, instead of mutate the previous state directly, it will return a new state instead.

References:

- https://www.youtube.com/watch?v=9boMnm5X9ak&list=PLC3y8-rFHvwheJHvseC3I0HuYI2f46oAK&index=2&ab_channel=Codevolution
- https://www.youtube.com/watch?v=poQXNp9ItL4&ab_channel=ProgrammingwithMosh