**REACT 11: REDUX HOOK**

A – LÝ THUYẾT

I – REDUX

**1. Truyền State theo cách thông thường**

Thông thường để truyền State tới các Component trong App thì chúng ta sẽ truyền tuần tự và lần lượt, các bạn có thể theo dõi hình dưới để hiểu rõ hơn về cơ chế truyền qua lại các State giữa các Component

App

Com C

Com B

Com A

**2. Một số vấn đề khi truyền State theo cách thông thường**

Khi ứng dụng được mở rộng, số lượng Component dần nhiều hơn thì việc truyền State đến các Component cũng trở nên phức tạp, tuy nhiên đó chưa phải vấn đề lớn, vấn đề quan trọng muốn nhấn mạnh ở đây đó là bản chất của truyền State thông thường đó là sự tuần tự và lần lượt, do đó nếu như chúng ta muốn bổ sung hay loại bỏ một Component nào trong chuỗi Component trước đó thì đều gây ra đứt gãy quá trình truyền Component. Các bạn có thể theo dõi hình dưới để hiểu rõ hơn về vấn đề gặp phải trong thực tế này

1. Trường hợp bổ sung thêm Com New

Com New

Com B

Com A

Com C

App

1. Trường hợp loại bỏ đi Com B

App

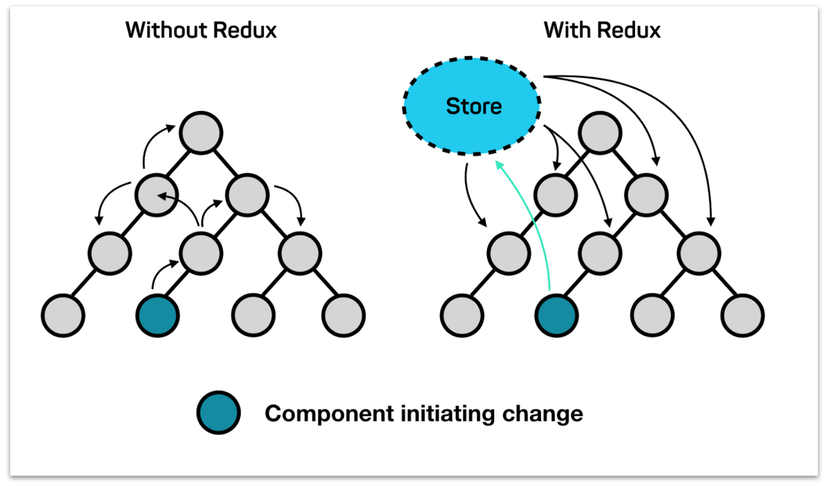
Com A

Com C

**3. Khắc phục vấn đề với giải pháp sử dụng Redux**

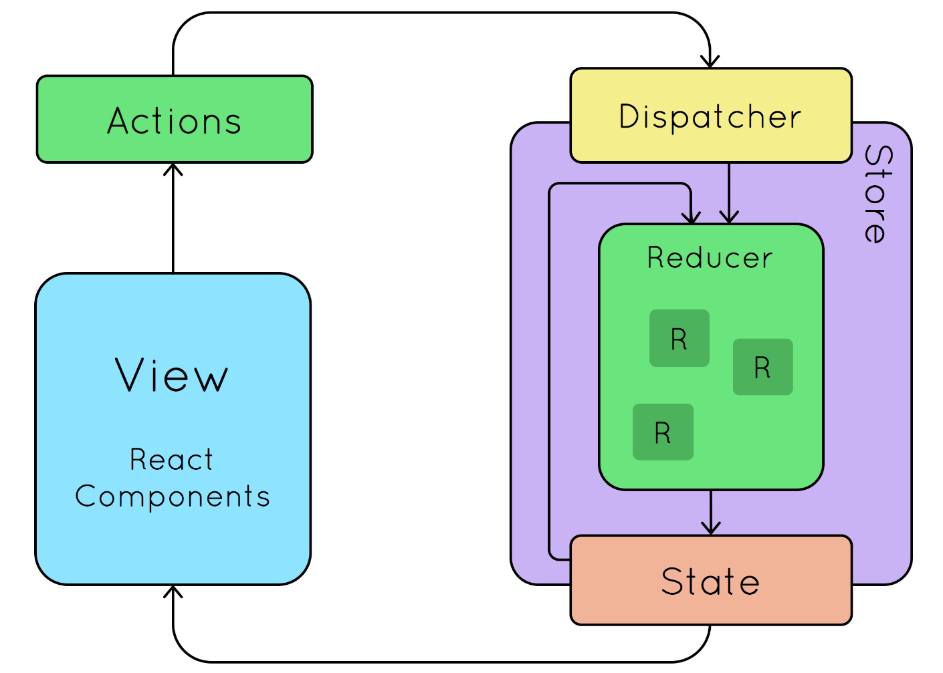
Redux không phải là thư viện, cũng không phải là công nghệ gì mới mẻ, nó đơn giản chỉ là một giải pháp để khắc phục cách truyền State qua lại giữa các Compoennt theo phương pháp thông thường trước đây

Đơn giản nó lưu trữ State như một biến toàn cục ở một nơi gọi là Store, từ đó co thể cấp phát State trực tiếp tới các Component cần sử dụng mà không phải thông qua bất kỳ một Component trung gian nào khác



II – REDUX HOOK

**1. Kiến trúc của một ứng dụng Redux**



**Trong đó:**

* **Action**: Tương tác của người dùng lên trên giao diện làm phát sinh một sự kiện nào đó (Click, Hover, Reload,…)
* **Dispatcher**: Xử lý sự kiện được kích hoạt từ người dùng (tương tác trực tiếp với Reducer, nhằm giúp Reducer cho ra các giá trị mới tương ứng cho State)
* **Reducer**: Nơi tiếp nhận Action và phân tích xem Action đó đã làm gì, từ đó cho ra các giải thuật để tạo ra các giá trị mới cho State (đây là State có khả năng cung cấp giá trị cho mọi Component muốn sử dụng)
* **State**: Nơi lưu trữ các giá trị của một ứng dụng như của React thông thường, tuy nhiên State của Redux có khả năng cung cấp giá trị một cách trực cho mọi Component nào cần mà không cần phải truyền qua lại gián tiếp thông qua Props
* **Store**: Nơi quản lý tất cả các State và kiểm soát giá trị của State thông qua Reducer

**2. Redux Hook**

Như chúng ta đã biết, từ phiên bản 16.8 thì React đã đưa React Hook vào sử dụng và đã mang lại hiệu quả tuyệt vời. Do Redux đi với React nên khi cách làm việc với React they đổi theo chiều hướng sử dụng Hook thì Redux cũng thay đổi theo. Và từ phiên bản 7.1.0 thì React-Redux đã chính thức cung cấp các API Hook để chúng ta xử lý thay thế cho các phương pháp cũ

**3. Các API Hook khi làm với Redux**

* useSelector()
* useDispatch()
* useStore()

III – CÁC BƯỚC TÍCH HỢP REDUX CHO MỘT ỨNG DỤNG THỰC TẾ

Để hiểu rõ hơn về bản chất của Redux cũng như Redux Hook thì chúng ta cùng nhau nghiên cứu một ứng dụng đơn giản có tích hợp Redux và sử dụng API Hook, đó là ứng dụng **“Hiển thị số ngẫu nhiên từ 1-10 khi bấm vào nút Button”**

**1. Cấu trúc và code cơ bản ban đầu của các file trong ứng dụng**

**Code index.html**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Vietpro React</title>  <meta charset="utf-8">  <style>  .App{  text-align: center;  }  </style>  </head>  <body>  <div id="root"></div>  </body>  </html> |

**Code index.js**

|  |
| --- |
| import React from "react"  import ReactDom from "react-dom"  import App from "./App"  ReactDom.render(<App />, document.getElementById("root")) |

**Code App.js**

|  |
| --- |
| import React from "react"  import Number from "./components/Number"  import Button from "./components/Button"  function App() {  return (  <div className="App">  <Number />  <Button />  </div>  )  }  export default App |

**Code Component Number.js**

|  |
| --- |
| import React from "react"  function Number() {  return (  <p>  <b>8</b>  </p>  )  }  export default Number |

**Code Component Button.js**

|  |
| --- |
| import React from "react"  function Button() {  return <button>Click</button>  }  export default Button |

**2. Các bước tích hợp Redux và sử dụng Redux Hook cho ứng dụng**

**Bước 1: Cài đặt các Package cần thiết**

**Code:**

|  |
| --- |
| npm install redux --save  npm install react-redux --save |

**Trong đó:**

* **redux**: module khởi tạo Redux
* **react-redux**: module để kết nối React với Redux

**Bước 2: Khởi tạo Store**

Tạo một file store.js nằm trong thư mục redux theo đường dẫn sau **src/redux/store.js**

**Code JS:**

|  |
| --- |
| import { createStore } from "redux"  const defaultState = {  num: 0  }  const reducers = (state = defaultState, action) => {  switch (action.type) {  case "ACTION":  return { num: Math.ceil(Math.random() \* 10) }  default:  return state  }  }  const store = createStore(reducers)  export default store |

**Trong đó:**

* **defaultState**: là một State với giá trị mặc định ban đầu (nó thay thế cho state khai báo trong Constructor trước đây)
* **reducer**: là một phương thức xử lý các Action (tiếp nhận các Action và trả về state mới với các giá trị mới)
* **createStore** là một phương thức có sẵn của Redux, nó nhận vào reducer để tạo ra một đối tượng Store

**Bước 3: Tích hợp Store cho tất cả các Component**

**Code App.js**

|  |
| --- |
| import React from "react"  import { Provider } from "react-redux"  import store from "./redux/store"  import Number from "./components/Number"  import Button from "./components/Button"  function App() {  return (  <Provider store={store}>  <div className="App">  <Number/>  <Button/>  </div>  </Provider>  )  }  export default App |

**Trong đó:**

* **Provider**: là đối tượng được trích xuất từ React-Redux nhằm tạo ra một đối tượng toàn cục giúp có thể gửi Props tới toàn bộ các Component trong Component App
* **Thuộc tính store**: chính là việc truyền Store tới tất cả các Component thông qua một Props tên bắt buộc phải là store

**Bước 4: Nhận State từ Store Redux**

**Code Number.js**

|  |
| --- |
| import React from "react"  import { useSelector } from "react-redux"  function Number(props) {  const number = useSelector(function (state) {  return state.num;  })  return (  <p>  <b>{number}</b>  </p>  )  }  export default Number |

**Trong đó:**

* Hook “useSelector” cho phép trích xuất dữ liệu từ State của redux.
* Tham số nhận vào là 1 function selector với đối số là state của redux.

**Bước 5: Gửi hành động bằng Dispatch**

**Code Buuton.js**

|  |
| --- |
| import React from "react"  import { useDispatch } from "react-redux"  function Button(props) {  const dispatch = useDispatch();  const onClickButton = (e) => {  dispatch({ type: "ACTION" })  }  return <button onClick={onClickButton}>Click</button>  }  export default Button |

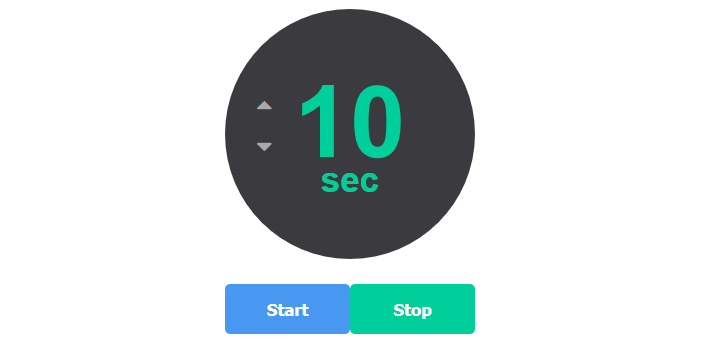
**Trong đó:**

* Hook “useDispatch” trả về tham chiếu tới dispatch của Redux.
* Có thể sử dụng hook để gửi hành động khi cần thiết

B – BÀI TẬP

I – BÀI TẬP THỰC HÀNH

**1. Xây dựng ứng dụng Pomodoro Clock như hình dưới**



**Trong đó**

* Khi bấm nút Start thì ứng dụng bắt đầu đếm lùi mỗi lần 1 đơn vị sau mỗi giây, bộ đếm lùi đến khi về 0 thì dừng lại
* Trong quá trình thời gian chạy lùi, chúng ta có thể bấm vào nút Stop để tạm dừng việc chạy thời gian
* Ngoài ra có 2 nút Control ngay cạnh phần hiển thị thời gian
  + Nút tăng sẽ cho phép chúng ta tăng thời gian lên 1 đơn vị mỗi lần bấm
  + Nút giảm sẽ cho phép chúng ta giảm thời gian đi 1 đơn vị mỗi lần bấm

II – BÀI TẬP VỀ NHÀ

**1. Hoàn thiện ứng dụng Pomodoro Clock trên lớp**

* Fix lỗi double click vào nút Start khiến bộ đếm chạy bị nhanh và sai
* Khi bộ đếm chạy về 0 thì chuyển trạng thái của nút Start thành Reset
* Khi bấm nút Reset thì bộ đếm trở về giá trị mặc định ban đầu là 10
* Các nút control tăng/giảm thời gian chỉ sử dụng được khi bộ đếm ở trạng thái Stop

***Chú ý:***

* *Trong bài tập sẽ sử dụng toán tử “…” của ES6, toán tử này cho phép copy lại toàn bộ tất cả các phần tử của một đối tượng bất kỳ*

**Ví dụ:**

|  |
| --- |
| const obj1 = {  course: "ES6"  }  const obj2 = {  ...obj  }  console.log(obj1);  console.log(obj2);  // Object { course: "ES6" }  // Object { course: "ES6" } |