React 程序开发

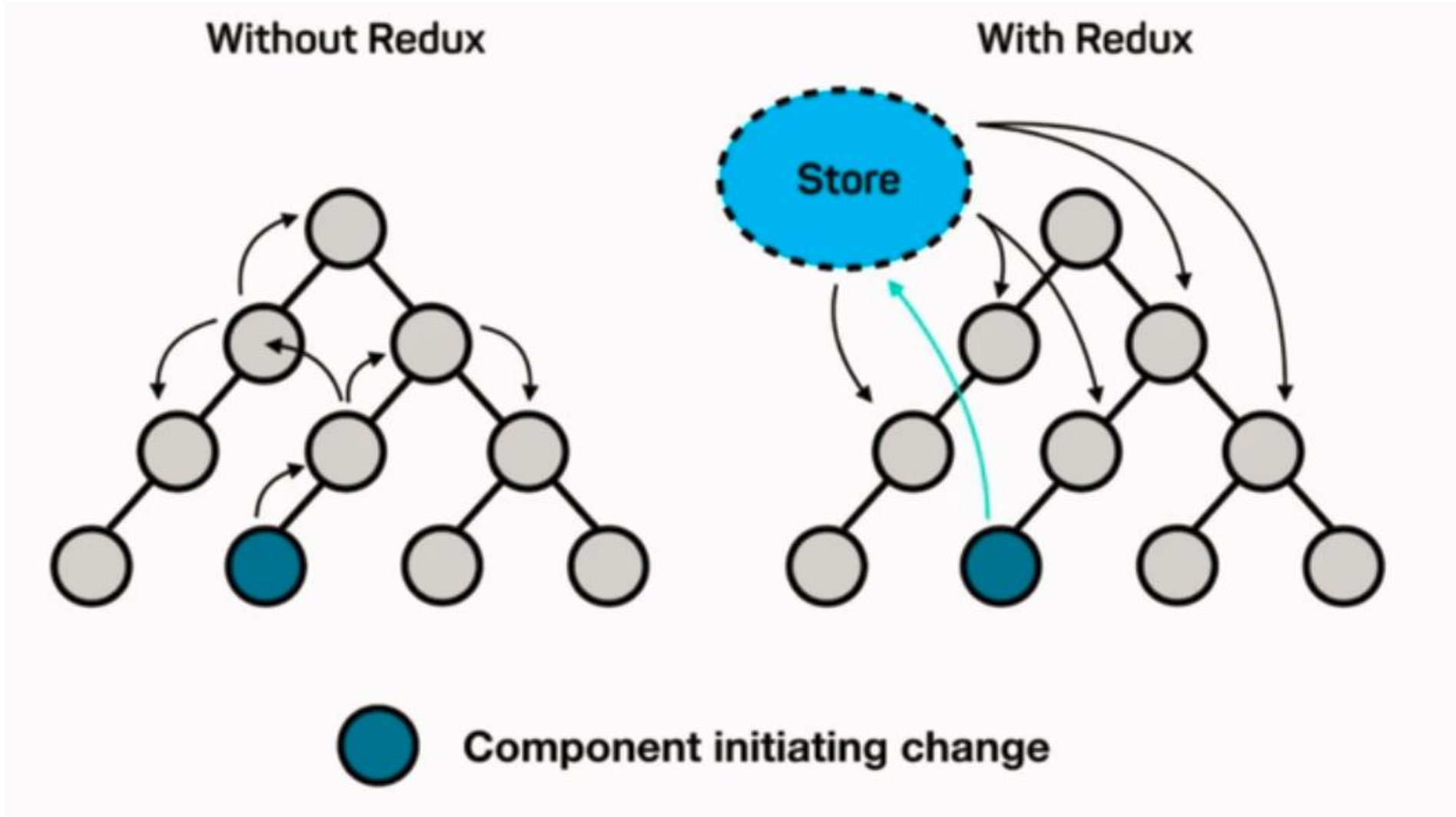
--- Redux

河北师范大学软件学院
Software College of Hebei Normal University

内容提纲

- **Redux 简介**
- **Redux 基础**
- **Reducer 拆分**

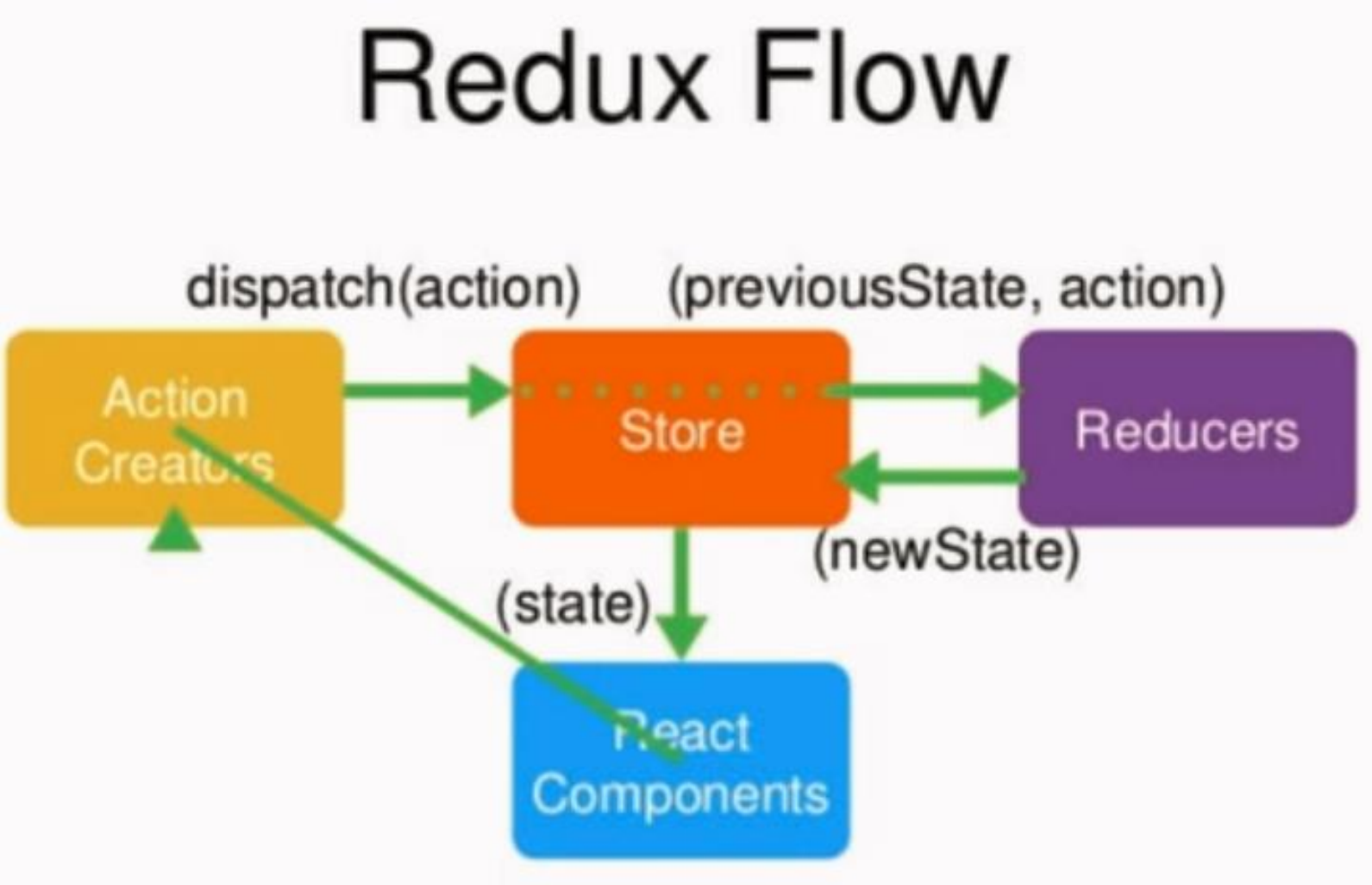
Redux 简介


Without Redux
With Redux
Store
Component initiating change

Redux 简介

- Redux
  - Redux 是 JavaScript 状态容器，提供可预测化的状态管理
- 目的
  - 随着 JavaScript 单页应用开发日趋复杂，JavaScript 需要管理比任何时候都要多的 state（状态），管理不断变化的 state 非常困难
- 核心概念
  - Action、Reducer、Store

Redux 简介


Redux Flow
dispatch(action)
(previousState, action)
Action Creators
Store
Reducers
(newState)
(state)
React Components

内容提纲

- Redux 简介
- Redux 基础
- Reducer 拆分

Redux 基础

- Action
  - Action 是把数据从应用传到 store 的有效载荷
  - 是 store 数据的**唯一**来源
  - 一般会通过 store.dispatch( ) 将 action 传到 store
  - Action 本质上是 JavaScript 普通对象
  - Action 内必须使用一个字符串类型的 type 字段来表示将要执行的动作
  - 一般 type 会被定义成字符串常量
  - 当应用规模越来越大时，建议使用单独的模块或文件来存放 action

Redux 基础

- Action 代码展示

```
// 普通对象
let action = { type : ’add_todo_item’ , inputValue:’hello’ }

//  利用字符串常量
const ADD_TODO = ‘add_todo'
let  action = { type: ADD_TODO, text: 'Build my first Redux app' }

// 利用单独的模块或者文件来存放 Action
import { ADD_TODO, REMOVE_TODO } from '../actionTypes'
```

Redux 基础

- Reducer
  - 指定了应用状态的变化如何响应 action 并发送到 store 的
  - reducer 就是一个纯函数，接收旧的 state 和 action，返回新的 state
  - (previousState, action) => newState
  - 禁止在 Reducer 里做这些操作
    - 修改传入参数；
    - 执行有副作用的操作，如 API 请求和路由跳转；
    - 调用非纯函数，如 Date.now() 或 Math.random( )

Redux 基础

- Reducer 代码展示

```
// 一个纯函数
…
function todoApp(state = initialState, action) {
    switch (action.type) {
        case ‘add_todo_item’:
                let newState = JSON.parse(JSON.stringify(state));
                newState.inputValue = action.value;
                return newState;
         default:
                return state
    }
}
```

Redux 基础

- Store
  - Store 是把 Action 和 Reducer 联系到一起的对象
  - Redux 应用只有一个单一的 store
  - 作用：
    - 维持应用的 state；
    - 提供 getState( ) 方法获取 state；
    - 提供 dispatch(action) 方法更新 state；
    - 通过 subscribe(listener) 注册监听器;
    - 通过 subscribe(listener) 返回的函数注销监听器

Redux 基础

- Store 代码展示

```
// 创建 store
import  { createStore }  from  'redux';
import  todoApp  from  './reducers' ;
let store = createStore( todoApp );
export default store
```

Redux 基础

- Store 代码展示

```
import  store  from './store' ;
…
class TodoList extends Component {
        constructor( props){
                super( props );
                this.state = store.getState(  );
                store.subscribe( (  )=>{
                        this.setState(store.getState())
                } )
        }
}
```

Redux 基础

- Action 代码展示

```
class TodoList extends Component {
        constructor( props){
                …
        }
        this.handleChange = ( e )=>{
                let action = { type : ‘change’ , inputValue: e.target.value }
                store.dispatch( action )
        }
}
```

demo01

Redux 三大原则

- 单一数据源
  - 整个应用的 state 被储存在一棵 object tree 中
  - 这个 object tree 只存在于唯一一个 store 中
- State 是只读的
  - 唯一改变 state 的方法就是触发 action(一个用于描述已发生事件的普通对象)
- 使用纯函数来执行修改
  - 为了描述 action 如何改变 state tree，你需要编写 reducers
  - Reducer 只是一些纯函数，它接收先前的 state 和 action，并返回新的 state

内容提纲

- Redux 简介
- Redux 核心概念
- **Reducer 拆分**

Reducer 拆分

- Action 代码展示

```
//  引入 combineReducers
import { combineReducers } from 'redux‘;
//  引入或者声明 inputChange 和 addTodo 函数
const todoApp = combineReducers({
      inputChange,
      addTodo
})
export default todoApp
```

Thank You

河北师范大学软件学院
Software College of Hebei Normal University