领域驱动的FRP复合范式在复杂前端应用的实践

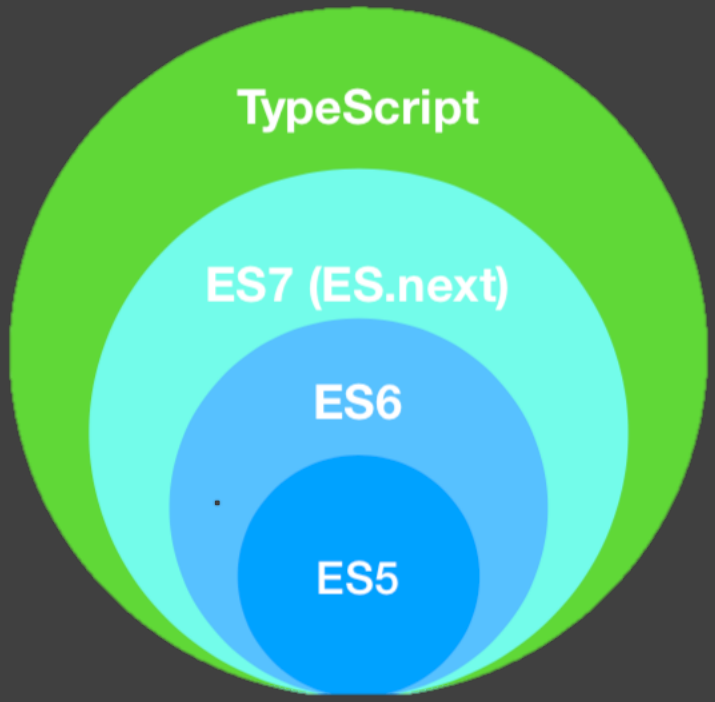
中电六所

胡戎

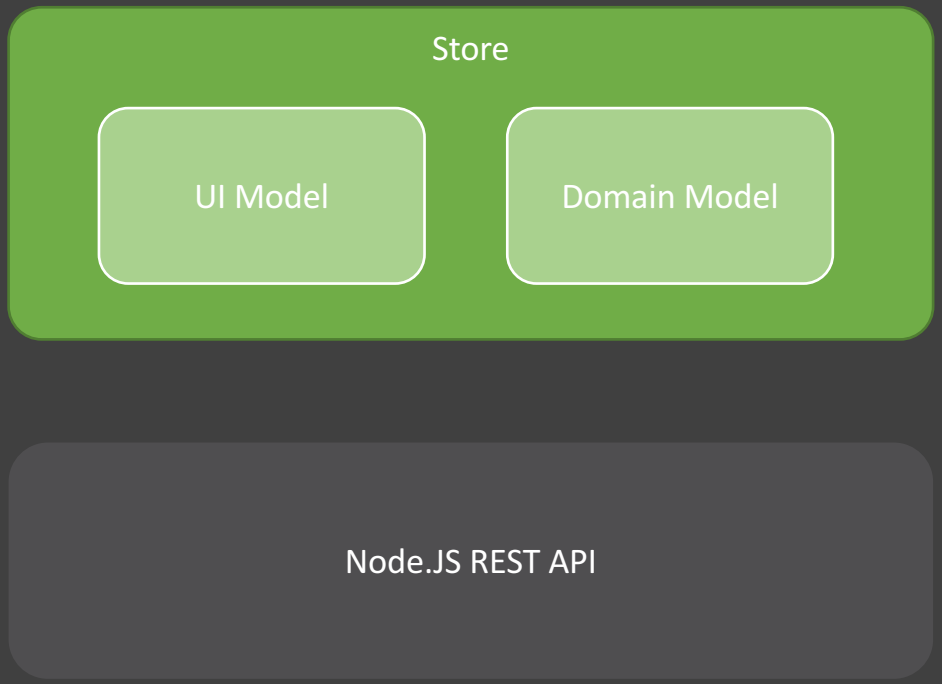
主题

- 前端应用发展的现状
- 函数响应式编程在前端的应用
- DDD在前端开发中的应用

前端应用发展现状



DDD在前端开发中的应用



DDD在前端开发中的应用

```
export const GetProjects=(action$,store) => {
    return action$.ofType( key: 'ON_GET_PROJECTS')
        .switchMap(()=>{
            return Observable.fromPromise(dao.getProject())
                .map((res)=>{
                    const Projects:Array<IProject> = res.data;
                    console.log(Projects[0].somefield)
                    return getProjectSuccess(Projects)
                }).catch(error=>{
                    return Observable.of({
                        type: 'GET_PROJECT_REJECTED',
                        payload: error,
                        error: true
                    })
                })
        })
}
```

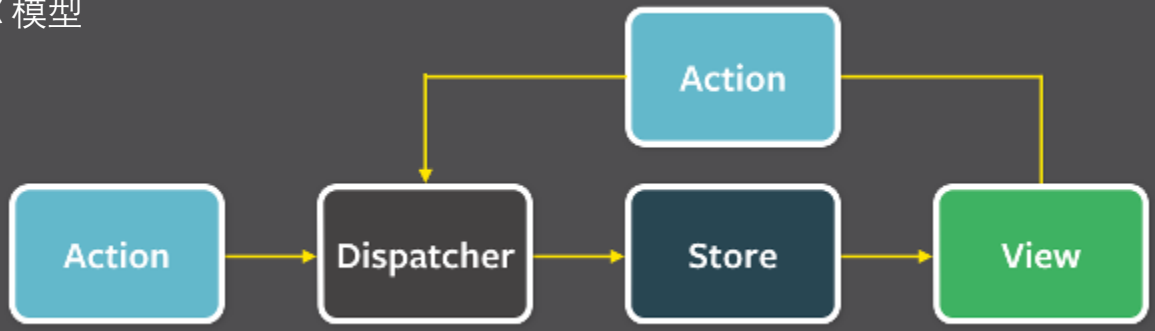
```
export interface IProject{
    id:number,//项目ID
    pname:string,
    start_time:string,
    responserID:number,
    groupID:Array<number>,
}
```

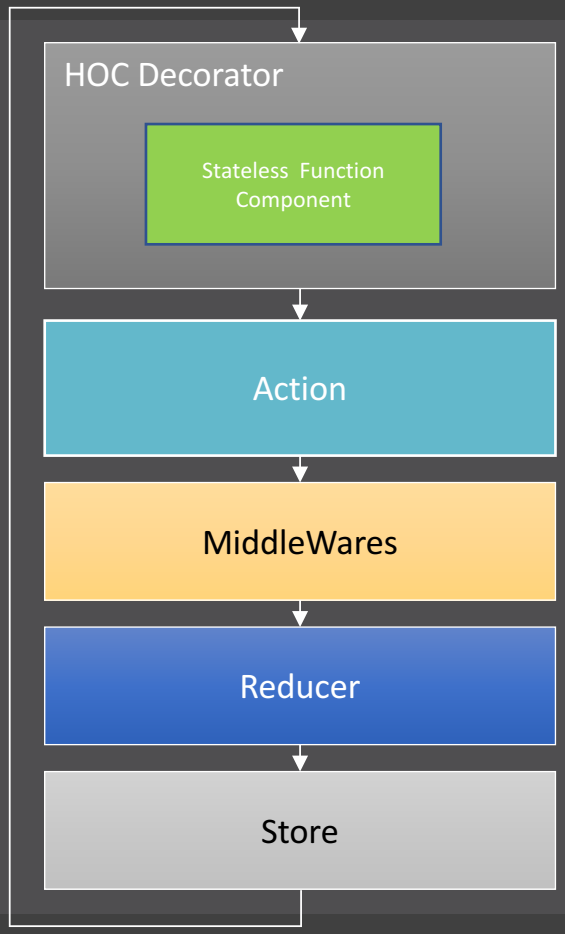
```
@Post("/project/:id")
@ContentType("application/json")
async PostProject(@Req() request: Request,@Body() project: IProject)
    try{
        console.log(project.someField)
        const [Projects]= await connection.execute(InseatProject);
        return Projects
    }catch (e){
        throw new BadRequestError(e)
    }
}
```

```
/工作项目/人力资源项目/人力资源项目源码/src/Client/epics/ComplexSearchPlan/SearchPlansEpic.ts
  Error:(82, 49)  TS2339:Property 'somefield' does not exist on type 'IProject'.
/工作项目/人力资源项目/人力资源项目源码/src/Server/Controller/ProjectController.ts
  Error:(42, 33)  TS2339:Property 'someField' does not exist on type 'IProject'.
```

前端应用发展现状

FLUX 模型



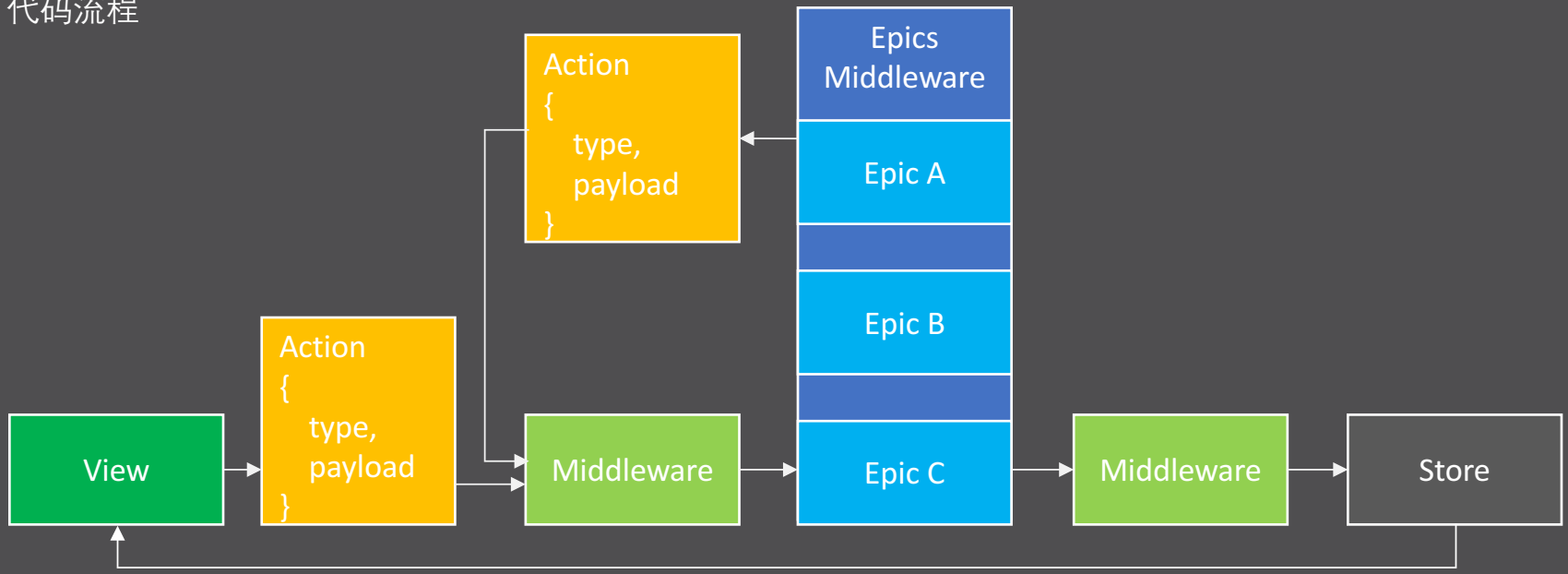



```
export interface orderProps {
    plan?:ISearchPlan
    fieldNames?: Array<filedObject>
    tableNames?: Array<string>
    onTabelSelected?: () => void,
    onFieldSelected?: () => void,
    onDeleteRuleItem?:(ruleIndex)=>{
    }
    onPlanRuleADD:(selectedPlanID)=>any
}
const EmployeeComplicatedRulesTablePanel: React.SFC<orderProps>
    const {plan} = props;
    const columns : TableColumnConfig<IRules>[] = [...];
    return (<Table columns={columns} dataSource={plan?plan.rules
     return (<div className={'Complex-title'}>
         <div className="ComplexTitle-text">{plan?plan.planName
         <div className="ComplexTitle-AddBtn">

             <Button icon={'save'} type={'primary'}  onClick={()
```

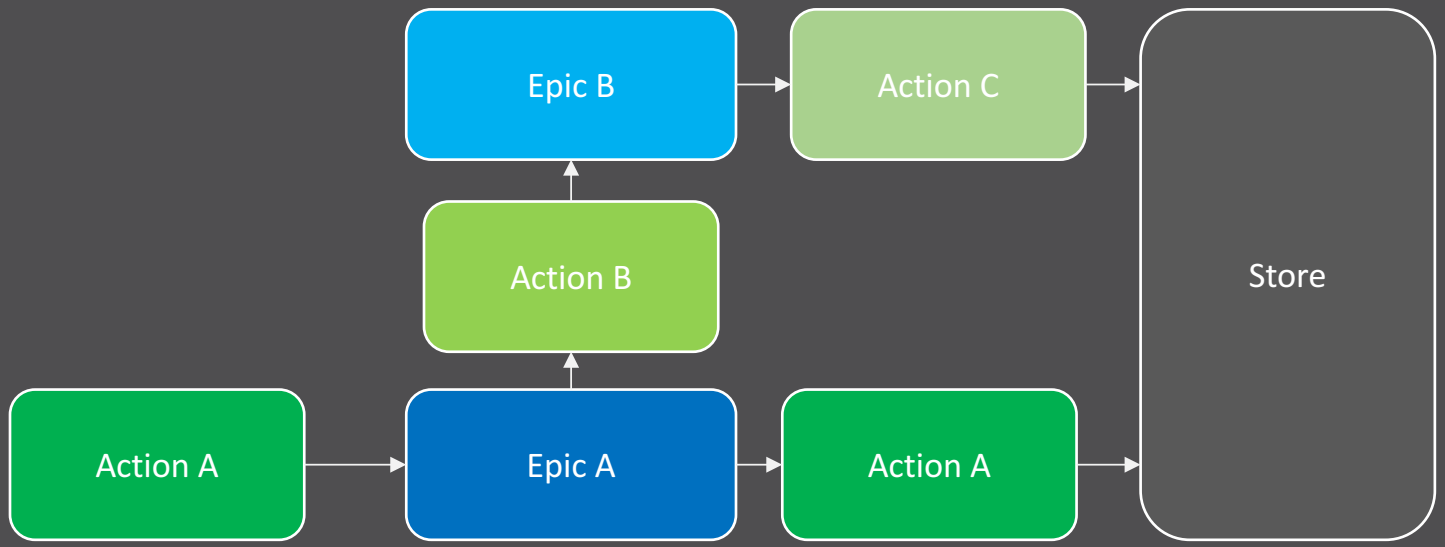
函数响应式编程在前端中的应用

代码流程



函数响应式编程在前端中的应用

业务流逻辑视图



1传统模式

```
export function initPersonal(workNum:string){
    return async (dispatch)=>{
        try{
            const res:any = await dao.initPersonalInfoDao(workNum);
            dispatch(onInitPersonalSuccess(res.data));
        }catch(err) {
            dispatch(onInitPersonalError(err));
        }
    }
}
```

1 Race Condition

```
export const initComplexSearchEpic = action$ => {
    return action$.ofType(OnSearchPlanInfoInit.type)
        .switchMap(()=>{
            return Observable.forkJoin(
                [Observable.fromPromise(SearchPlandao.getAllSearchPlans()),
                 Observable.fromPromise(SearchPlandao.GetEmployeeTablesName())
                ])
                .map(([searchPlans,EmployeeTables])=>{
                    return OnSearchPlanInfoInitSuccess(searchPlans.data,EmployeeTables.data)
                }).catch(e=>{
                    return ActionsObservable.of(OnSearchPlanInfoInitError(e))
                })
        })
}
```

2 渐进式查询

```
rx.Observable.merge(
    rx.Observable.fromPromise(orgSearch),
    rx.Observable.fromPromise(EmployeeSearch))
    .scan((acc,value)=>{
        switch (value.type){
            case 'org': return acc.org=value;
            case 'Employee': return acc.Employee=value;
        }
    },{})
```