**Actions**

action是NgRx中的主要构建块之一。action表示应用程序中发生的唯一事件。从用户与页面的交互，通过网络请求的外部交互，以及与设备api的直接交互，这些以及更多的事件都用action来描述。

**Introduction**

在NgRx的许多领域都使用了action。action是NgRx中许多系统的输入和输出。动作帮助您理解在应用程序中如何处理事件。本指南提供了在应用程序中编写操作的一般规则和示例。

**The Action interface**

NgRx中的一个动作由一个简单的接口组成:

interface Action { type: string; }

该接口有一个单独的字符串属性，表示为type。type属性用于描述将在应用程序中分派的Action 。type的值以[事件源]事件名称的形式出现，用于提供action类别的上下文，以及action从何处分派。向action添加属性以为action提供额外的上下文或元数据。

下面列出的是用普通旧JavaScript对象(pojo)编写的动作示例:

{ type: '[Auth API] Login Success' }

此action描述与后端API交互后由成功身份验证触发的事件。

{

type: '[Login Page] Login',

username: string;

password: string;

}

此action描述用户从登录页面单击登录按钮以尝试对用户进行身份验证所触发的事件。用户名和密码被定义为登录页面提供的附加元数据。

在应用程序中编写好的action有一些规则。

Upfront - 在开发特性之前编写action，以理解和获得要实现的特性的共享知识。

Divide - 根据事件源对action进行分类。

Many - action的编写成本不高，因此编写的action越多，在应用程序中表达流就越好。

Event-Driven - 捕获事件而不是命令，因为您正在分离事件的描述和事件的处理。

Descriptive - 提供针对特定事件的上下文，以及可以用于帮助使用开发人员工具进行调试的更详细信息。

遵循这些指导原则可以帮助您了解这些action在应用程序中是如何流动的。

让我们看一个初始化登录请求的示例action。

import { createAction, props } from '@ngrx/store';

export const login = *createAction*(

'[Login Page] Login',

props<{ username: string; password: string }>()

);

createAction函数返回一个函数，该函数被调用时返回一个动作接口形状的对象。props方法用于定义action处理所需的任何额外元数据。action创建者提供了一种一致的、类型安全的方法来构造要分派的action。

在调度时使用action创建器返回动作。

onSubmit(username: string, password: string) {

store.dispatch(login({ username: username, password: password }));

}

登录操作创建者接收用户名和密码对象，并返回一个普通的JavaScript对象，其类型属性type为[login Page] login，用户名和密码作为附加属性。

返回的action有关于action来自哪里和发生了什么事件的非常具体的上下文

action的类别在方括号[]中捕获。

类别用于对特定区域的操作进行分组，无论该区域是组件页面、后端API还是浏览器API。

类别之后的登录文本是关于此action发生的事件的描述。在本例中，用户单击登录页面中的登录按钮，尝试使用用户名和密码进行身份验证。

*您还可以使用基于类的操作创建器来编写操作，这是在NgRx引入操作创建器之前定义的方法。如果您正在寻找基于类的操作创建器的示例，请访问版本7的文档。x和之前。*