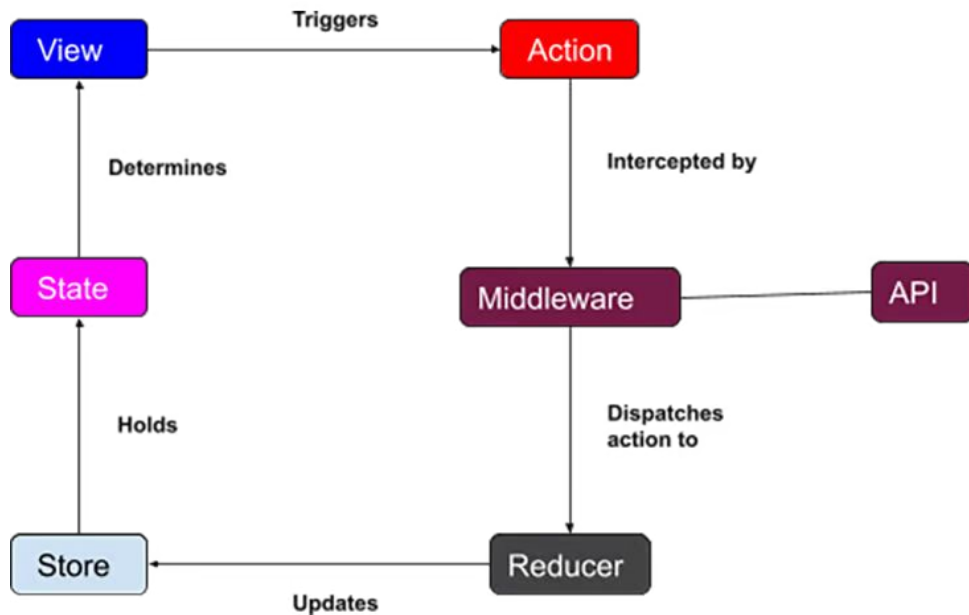
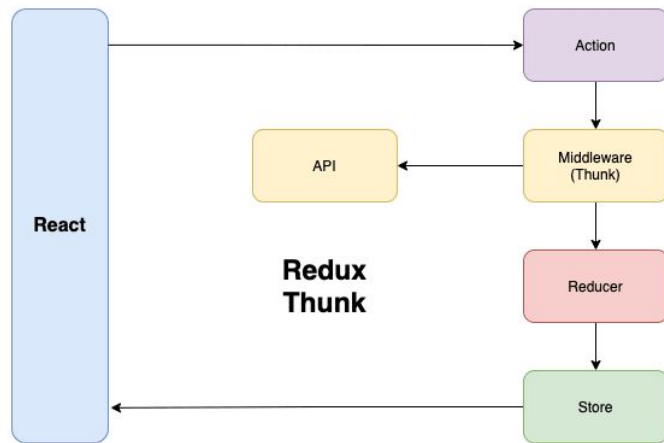


Тема занятия:
Redux Thunk

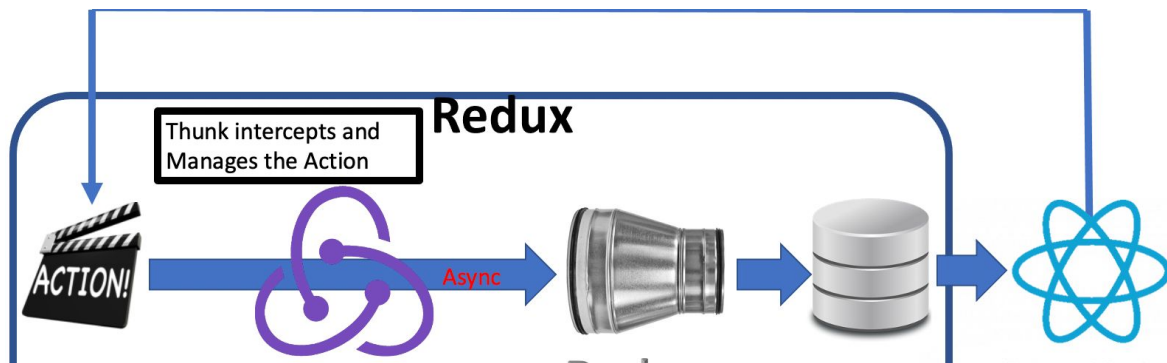
Middleware - промежуточная функция, которое берет входные данные, делает с ними что-то (например асинхронное действие - отправку запроса на сервер) и передаёт дальше.



Redux Thunk — это middleware для библиотеки управления состоянием Redux. Он предоставляет возможность создавать и обрабатывать асинхронные действия (actions) в Redux.



В **Redux Toolkit** существует удобная функция **create.asyncThunk**, которая упрощает создание асинхронных действий, и она уже использует **Redux Thunk** внутри себя.



Шаг 1

Создаём slice

```
import { PayloadAction } from '@reduxjs/toolkit';
import { createAppSlice } from "store/createAppSlice"

interface UserSliceState {
  data: UserData | null
  status: 'not sent' | 'loading' | 'success' | 'failed'
  error: string | null;
}

const userInitialState: UserSliceState = {
  data: null,
  status: 'idle',
  error: null,
};
```



продолжение

Шаг 1

Создаём slice

↓
продолжение

```
export const userSlice = createAppSlice
({
  name: 'USER',
  initialState: userInitialState,
  reducers: create => ({
    getUser: create.asyncThunk(
      async () => {
        //fetchDataFromServer функция написанная заранее, которая делает какой-то запрос

        const response = await fetchDataFromServer()
        return response
      },
      {
        pending: state => {
          state.status = "loading"
        },
        fulfilled: (state, action) => {
          state.status = "success"
          state.data = response
        },
        rejected: state => {
          state.status = "failed"
          state.error = response.error
        },
      },
    ),
  }),
})
```

Шаг 1

Создаём slice

```
selectors: {  
  user: state => state,  
},  
});  
export const userSliceActions = userSlice.actions;  
export const userSliceSelectors = userSlice.selectors;
```

Шаг 2

Используем в
компоненте

```
import { useAppDispatch, useAppSelector } from 'store/hook'
import { userSliceSelectors, userSliceActions } from
'store/redux/user/userSlice'

const UserData= () => {
  const dispatch = useAppDispatch();
  //эти значения можно использовать в дальнейшем
  const {data, status, error} = useAppSelector(userSliceSelectors);

  return (
    <div>
      <button onClick={() => {dispatch(userSliceActions.getUser())}}>
        Get user
      </button>
    </div>
  );
};

export default UserData
```