**Fetch Data from Real World Movies API — Part 2**

Learn how you can filter out movies based on different categories.

**[Fetch Data from a Real API — React / JS](https://blog.devgenius.io/fetch-data-from-a-real-api-react-js-f962da8af24a" \t "_blank)**

[Learn how we can fetch real data from real-world API using React/JS](https://blog.devgenius.io/fetch-data-from-a-real-api-react-js-f962da8af24a" \t "_blank)

[blog.devgenius.io](https://blog.devgenius.io/fetch-data-from-a-real-api-react-js-f962da8af24a" \t "_blank)

**Let’s Start…**

Graphical user interface, text, application, chat or text message

Description automatically generated

Movies

**Creating Movies Web Application with Filters and Categories using React.**

We have just learned how to fetch data from the real-world API. Now we are going to add Filters and Categories to the project to make our project Awesome.

*Creating Filter Component :****Filter.jsx***

With reference to our project, we now need to create a **Filter**component. Here are going to create just 3 filters for our project. viz All, Comedy and Action.

import React from 'react';const Filter = () => {  
 return (  
 <div className="filter-container">  
 <button>All</button>  
 <button>Comedy</button>  
 <button>Action</button>  
 </div>  
 );  
};export default Filter;

Now if we go to App.js we have got the **popular**movies using the first hook:  
const [popular, setPopular] = useState([]);

Now you might wonder how to get this functionality, This hook currently holds the set of **popular**movies(we have looped over it using **map()**) see **App.js**, right? So we will create a new state that copies everything from there but it filters out elements or movies based on our click.

Now we will add **setFiltered()** to **fetchPopular()**in **App.js** as below:

Text

Description automatically generated

set the filter…

Text

Description automatically generated

Now if see check the **console**we will get two arrays, right? It means it has copied data from **popular**to **filtered**. Awesome!

Text

Description automatically generated

2 objects were found.

Also if you installed React DevTools extension in Chrome Browser, you can there are 2 pieces of state available i.e First is the original one, and the second is Filtered one.

Graphical user interface, application

Description automatically generated

2 states were created.

Now we will pass props to Filter.js to copy and modify the data. The **setFiltered={setFiltered}** will actually modify a copy of **filtered**. i.e const [filtered, setFiltered] = useState([]);

Now are going to create another piece of state called **ActiveGenre**to know which button is Active, const [activeGenre, setActiveGenre] = useState(0); Now you think why it starts from Zero ‘0’, right?

The reason we don’t know the number of Genres where it starts and ends there, and suppose if it were a string then might have passed ‘**all**’ there e.g. const [activeGenre, setActiveGenre] = useState('all');

Text

Description automatically generated

genre\_ids array varies to any number.

Again you might be confused about what is 28, 12, 878 in the array, so unfortunately we need to visit the website of [**movies**](https://developers.themoviedb.org/3)API.

Graphical user interface, application

Description automatically generated

Copy URL

Copy the above URL and paste it into a browser and get data returned in JSON format.

A picture containing diagram

Description automatically generated

list of genres id.

**→ App.jsx**

Text

Description automatically generated

passing props to Filter.jsx

→ Now we will pass props of Genres to **Filter.jsx**

Text

Description automatically generated

Now if check devTools Components, we have got **State:28** after clicking on **Action**Filter, Cool! Similarly, we will get:  
All → 0, Comedy → 35, and Action → 28

A screenshot of a computer

Description automatically generated with medium confidence

We got state 28 for the Action filter.

Now we will grab the duplicate array and based on this number (say 28, 25, etc), we are going to filter out the movies and again this will really be quite simple by using a **useEffect()** hook.  
This means runs this function when **activeGenre**changes.

useEffect(() => {  
   
},[activeGenre]);

Now grab the original movies from ‘**popular**’, and filter each individual movie with reference to genre ids and make sure it contains id 28, 35, etc.

**→ Filter the movies**.

Text

Description automatically generated

**→ You can see we list of movies from Comedy Filter.**

Graphical user interface, website

Description automatically generated

the filter is working.

There is one problem here, we are always looping over to **popular.map()** in **App.js**which is the original arrays so nothing changes on our homepage, right? i.e are we are getting our filter work.

→ Just change code from **popular**to **filtered**in a map() and you can see our images are also changing on the homepage.

Existing:  
<div className=”popular-movies”>  
{popular.map((movie) => {  
return <Movie key={movie.id} movie={movie} />;  
})}

New:  
<div className=”popular-movies”>  
{filtered.map((movie) => {  
return <Movie key={movie.id} movie={movie} />;  
})}

→ Add class name to**<Filter />** in order to show which is the current active Filter.

**Final Codes for references:**

**index.js**

import React from "react";  
import ReactDOM from "react-dom";  
import "./index.css";  
import App from "./App";ReactDOM.render(<App />, document.getElementById("root"));

**App.js**

import React, { useEffect, useState } from "react";  
import "./App.css";  
import Movie from "./Movie";  
import Filter from "./Filter";const App = () => {  
 const url =  
 "<https://api.themoviedb.org/3/movie/popular?api_key=19dedc791dc255982eaf84be8a93012a&language=en-US&page=1>";const [popular, setPopular] = useState([]);  
 const [filtered, setFiltered] = useState([]);   
 const [activeGenre, setActiveGenre] = useState(0);useEffect(() => {  
 fetchPopular();  
 }, []);const fetchPopular = async () => {  
 const data = await fetch(url);  
 const movies = await data.json();  
 console.log(movies);  
 setPopular(movies.results);  
 setFiltered(movies.results);  
 };return (  
 <div className="App">  
 <h1>Movies</h1>  
 <Filter popular={popular} setFiltered={setFiltered} activeGenre={activeGenre} setActiveGenre={setActiveGenre} />  
 <div className="popular-movies">  
 {filtered.map((movie) => {  
 return <Movie key={movie.id} movie={movie} />;  
 })}  
 </div>  
 </div>  
 );  
};export default App;

**Movie.jsx**

import React from "react";const Movie = ({movie}) => {  
 return (  
 <div>  
 <h5>{movie.title}</h5>  
 <img src={"<https://image.tmdb.org/t/p/w500>" + movie.backdrop\_path} alt={movie.path} />  
 </div>  
 );  
};export default Movie;

**Filter.jsx**

import React, { useEffect } from "react";const Filter = ({ popular, setFiltered, activeGenre, setActiveGenre }) => {  
 useEffect(() => {  
 if (activeGenre === 0) {  
 setFiltered(popular); //Checks- do nothing, return all  
 return;  
 }  
 const filtered = popular.filter((movie) =>  
 movie.genre\_ids.includes(activeGenre),  
 );  
 setFiltered(filtered);  
 }, [activeGenre]);return (  
 <div className="filter-container">  
 <button  
 onClick={() => setActiveGenre(0)}  
 className={activeGenre === 0 ? "active" : ""}  
 >  
 All  
 </button>  
 <button  
 onClick={() => setActiveGenre(35)}  
 className={activeGenre === 35 ? "active" : ""}  
 >  
 Comedy  
 </button>  
 <button  
 onClick={() => setActiveGenre(28)}  
 className={activeGenre === 28 ? "active" : ""}  
 >  
 Action  
 </button>  
 </div>  
 );  
};export default Filter;

**App.css**

.App {  
 text-align: center;  
}.popular-movies {  
 display: grid;  
 grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));  
 grid-column-gap: 1rem;  
 grid-row-gap: 2rem;  
}img {  
 width: 100%;  
 height: 30vh;  
 object-fit: cover;  
 border-radius: 1rem;  
 margin-bottom: 1rem;  
}.filter-container button {  
 margin-right: 2rem;  
 min-width: 5rem;  
 padding: 0.5rem 1rem;  
 border: none;  
 background: white;  
 color: rgb(65, 98, 168);  
 border-radius: 1rem;  
 border: 2px solid rgb(28, 36, 53);  
 font-weight: bold;  
 cursor: pointer;  
}button.active {  
 background: rgb(65, 98, 168);  
 color: white;  
}

**index.css**

body {  
 margin: 5% 20%;  
}

**Final output:**

Graphical user interface, website

Description automatically generated

final output with active filters.

**Conclusion:**

You just now learn how we can use filter the movie from real-world movies API.