

```

1 // ===== Connecting a Backend & Database =====
2 // Sending Http Requests
3 /*
4     API = application program Interface
5
6     1) How do React Apps Interact with Database ?
7     2) Sending Http Requests & Using Responses
8     3) Handling Errors & Loading State
9 */
10
11 /*
12     Browser-side Apps Don't Directly Talk to Databases
13
14     React App -----X ----- Database (SQL, NoSQL...)
15
16     But by the help of [Backend App, (Node js App, PHP, Asp.net)] connect the sever and get the data
17
18 */
19
20 /* ***** fetch('url API or server', {pass object Method: "post", header, body chance on this object})
21     we can use fetch() method to resuest and send data to in server
22
23     fetch('url').then(response => response.json();).then(data => useData);
24 */
25
26 // ***** fetch() data using API *****
27
28 const [movies, setMovies] = useState([]);
29
30
31 const fetchMoviesHandler = () => {
32
33     fetch('https://swapi.dev/api/films/').then(response => {
34         return response.json(); // data simple formate json or js object by json() function
35     }).then(data => {
36         // data tranform according to ours need change title, id, others things..
37         const transformedMovies = data.results.map(movieData => {
38             return {
39                 id: movieData.episode_id,
40                 title: movieData.title,
41                 openingText: movieData.opening_crawl,
42
43             };
44         });
45         setMovies(transformedMovies);
46     });
47 }
48
49 // ***** END *****
50
51 // ===== async and await =====
52
53 // sometimes we get data After few milliseconds later that time programm must not block that why we use async and await
54
55 // ***** now code look like this it is better way....
56 const [isLoading, setIsLoading] = useState(false);
57
58 function App() {
59     const [movies, setMovies] = useState([]);
60     async function fetchMoviesHandler() {
61
62         // when data is coming from server we can show that loading data .....
63         // by showing one components as yours wish
64
65         setIsLoading(true);
66
67         const response = await fetch('https://swapi.dev/api/films/');
68         const data = await response.json();
69         const transformedMovies = data.results.map(movieData => {
70             return {
71                 id: movieData.episode_id,
72                 title: movieData.title,
73                 openingText: movieData.opening_crawl,
74                 releaseDate: movieData.release_date,
75             }

```

```

76     });
77     setMovies(transformedMovies);
78
79     setIsLoading(false);
80     // After loading date make it is false and remove the loding components
81
82 }
83
84 return (
85   <React.Fragment>
86     {isLoading && movies.length > 0 && <MovieList movies={movies} />}
87     {isLoading && <p>loading ....</p>}
88   </React.Fragment>
89 )
90 }
91
92
93
94
95 // ===== END =====
96
97
98 // ===== full code fetch data by API url=====
99
100 // ***** APP.js START *****
101
102 import React, { useState } from 'react';
103
104 import MoviesList from './components/MoviesList';
105 import './App.css';
106
107 function App() {
108
109   const [movies, setMovies] = useState([]);
110   /* const dummyMovies = [
111     {
112       id: 1,
113       title: 'Some Dummy Movie',
114       openingText: 'This is the opening text of the movie',
115       releaseDate: '2021-05-18',
116     },
117     {
118       id: 2,
119       title: 'Some Dummy Movie 2',
120       openingText: 'This is the second opening text of the movie',
121       releaseDate: '2021-05-19',
122     },
123   ];
124   */
125   // we can use fetch() method to resuest and send data to in server
126   // and fetch return a promise....js term
127
128   const fetchMoviesHandler = () => {
129     fetch('https://swapi.dev/api/films/').then(response => {
130       return response.json(); // data simple formate json or js object by json() function
131     }).then(data => {
132       // data tranform acording to ours need
133       const transformedMovies = data.results.map(movieData => {
134         return {
135           id: movieData.episode_id,
136           title: movieData.title,
137           openingText: movieData.opeating_crawl,
138
139         };
140       });
141       setMovies(transformedMovies);
142     });
143   }
144
145   return (
146     <React.Fragment>
147       <section>
148         <button onClick={fetchMoviesHandler}>Fetch Movies</button>
149       </section>
150       <section>
151         <MoviesList movies={movies} />

```

```

152     </section>
153   </React.Fragment>
154 );
155 }
156
157 export default App;
158
159
160 // ***** END *****
161
162 // ***** MoviesList.js START *****
163
164 import React from 'react';
165
166 import Movie from './Movie';
167 import classes from './MoviesList.module.css';
168
169 const MovieList = (props) => {
170   return (
171     <ul className={classes['movies-list']}>
172       {props.movies.map((movie) => (
173         <Movie
174           key={movie.id}
175           title={movie.title}
176           releaseDate={movie.release}
177           openingText={movie.openingText}
178         />
179       ))}
180     </ul>
181   );
182 };
183
184 export default MovieList;
185
186 // ***** END *****
187
188 // ***** Movie.js *****
189
190 import React from 'react';
191
192 import classes from './Movie.module.css';
193
194 const Movie = (props) => {
195   return (
196     <li className={classes.movie}>
197       <h2>{props.title}</h2>
198       <h3>{props.releaseDate}</h3>
199       <p>{props.openingText}</p>
200     </li>
201   );
202 };
203
204 export default Movie;
205
206 // ***** END *****
207
208
209
210
211 // ===== Handling Http Errors =====
212
213 function App() {
214   const [movies, setMovies] = useState([]);
215   const [isLoading, setIsLoading] = useState(false);
216   const [error, setError] = useState(null);
217
218   async function fetchMoviesHandler() {
219
220     setIsLoading(true);
221     // after running the if it has some error that will remove by calling that...below setError(null);
222     setError(null);
223
224     // when we use async and await the to handle the error by using => try {} catch{} way
225
226     try {
227       // try some code and find error or generating error and use to handel that...below

```

```

228     const response = await fetch('https://swapi.dev/api/films');
229
230
231     if(!response.ok) {
232         throw new Error('something wrong there');
233     }
234
235     const data = response.json();
236
237     // we are actually generating error .. if the response not ok then we have to handel this by won your won
238
239     // some API will provide some json after the error ocured then .. we have to manage the After the data or berore the data get ....
240
241     /*
242     if(!response.ok) {
243         throw new Error('something wrong there');
244     }
245     */
246
247     const transformedMovies = data.results.map(movieData => {
248         return {
249             id: movieData.episode_id,
250             title: movieData.title,
251             release: movieData.releaseDate,
252         };
253     });
254     setMovies(transformedMovies);
255
256     setIsLoading(false);
257 }
258 catch (error){
259     setError(error.message);
260 }
261
262
263 }
264
265 return (
266
267     <React.Fragment>
268         {isLoading && movies.length > 0 && <MoviesList movies={movies} />}
269         {isLoading && <p>loading....</p> }
270         { !isLoading && error && <p>{error}</p>}
271     </React.Fragment>
272 );
273 }
274
275
276 // ===== END =====
277
278 // ***** using variables components using *****
279
280 function App(){
281
282     let content = <p>Found no data</p>;
283
284     if(movies.length > 0){
285         content = <MoviesList movies={movies}/>;
286     }
287
288     if(error){
289         content = <p>{error}</p>;
290     }
291
292     if(isLoading){
293         content = <p>Loading...</p>;
294     }
295
296
297     return (
298         <React.Fragment>
299             {content}
300         </React.Fragment>
301     );
302 }
303

```

```

304 //***** END *****/
305
306
307 // ===== useEffect() and callBack() =====
308
309 // in some Applicaton when user Enter that automitically loading data form database
310 // that time we can use useEffect() hook to get data imidiately from database
311
312 import React, {useState, useEffect, callBack} from 'react';
313
314 function App() {
315     const [movies, setMovies] = useState([]);
316     const [isLoading, setIsLoading] = useState(false);
317     const [error, setError] = useState(null);
318
319     // ***** here callBack *****
320
321     // callBackk theke the hole function because we want to make inside only one reference of that fetchMoviesHandler function...
322
323     const fetchMoviesHandler = callBack(async () => {
324
325         setIsLoading(true);
326         setError(null);
327
328         try {
329
330             const response = await fetch('https://swapi.dev/api/films');
331
332             if(!response.ok) {
333                 throw new Error('something wrong there');
334             }
335
336             const data = response.json();
337
338             const transformedMovies = data.results.map(movieData => {
339                 return {
340                     id: movieData.episode_id,
341                     title: movieData.title,
342                     release: movieData.releaseDate,
343                 };
344             });
345             setMovies(transformedMovies);
346
347             setIsLoading(false);
348         }
349         catch (error){
350             setError(error.message);
351         }
352     });
353
354
355     // if we use A hole function as a dependence then it will create infinity loop cos function is a object it sometime some value will bechnage ..
356     // components will call again and gain....that why we give the function inside the calback() to create only one reference of that function ....
357     useEffect(()=>{
358         fetchMoviesHandler();
359     }, [fetchMoviesHandler]);
360
361
362     return (
363
364         <React.Fragment>
365             {isLoading && movies.length > 0 && <MoviesList movies={movies} />}
366             { isLoading && <p>loading....</p> }
367             { !isLoading && error && <p>{error}</p>}
368         </React.Fragment>
369     );
370 }
371
372 // ===== END =====
373
374
375 // ===== get and post on firebase Realtime server =====
376
377 /*
378 1) create database
379 2) name the database what you want

```

```

380 3) select realtime database
381 4) copy the link (rest API)
382 5) fetch('https://react-http-2b87c-default-rtdb.firebaseio.com/AddFoder.json');
383 */
384
385
386
387
388 // ***** Post inside the database *****
389
390 async function addMovieHandler() {
391   const response = await fetch('https://react-http-2b87c-default-rtdb.firebaseio.com/AddFoder.json', {
392
393     // by default method: 'GET',
394
395     method: 'POST',
396     body: JSNO.stringify(movie),
397     header: {
398       'content-Type': 'application/json',
399     }
400   });
401
402   const data = await response.json();
403   console.log(data);
404 }
405
406
407 // ***** get data form server
408
409 // when firebase feedback the data that will a object....now we have to moififie like this way ...so that we can show the data....
410 // on our website.... how???
411
412 const fetchMoviesHandler = callBack(async ()=> {
413
414   setIsLoading(true);
415   setError(null);
416
417   try {
418
419     const response = await fetch('https://react-http-2b87c-default-rtdb.firebaseio.com/AddFoder.json');
420
421     if(!response.ok) {
422       throw new Error('something wrong there');
423     }
424
425     const data = response.json();
426
427     const loadedMovies = [];
428
429     for(const key in data){
430       loadedMovies.push({
431         id: key,
432         title: data[key].title,
433         openingText: data[key].openingText,
434         releaseDate: data[key].releaseDate,
435       });
436     }
437
438
439     setMovies(loadedMovies);
440
441     setIsLoading(false);
442   }
443   catch (error){
444     setError(error.message);
445   }
446 });
447
448
449
450 useEffect(()=>{
451   fetchMoviesHandler();
452 }, [fetchMoviesHandler]);

```