

# State

دیزاین پترن هادر React

Behavioral Patterns

به زبان ساده همراه با مثال های عملی  
کاربرد دیزاین پترن ها برای برنامه نویسان React



# Behavioral Patterns

## State



الگوی State به یک شی اجازه می‌دهد رفتار خود را بر اساس تغییر وضعیت داخلی خود تغییر دهد. این الگو زمانی مفید است که یک شی چندین حالت داشته باشد و نیاز باشد که رفتار آن با تغییر وضعیت تغییر کند.

فرض کنید می خواهیم یک API را Call کنیم، همانطوری که می دانید نتیجه Call کردن API می تواند یک از سه وضعیت زیر باشد.

- در حال بارگذاری (Loading)
- با موفقیت بارگذاری شد (Success)
- بنا به دلایل مختلف (قطعی اینترنت، مشکل در سرور و...) به درستی بارگذاری نشد (Error).

با استفاده از این الگو می خواهیم حالات فوق را پیاده سازی کنیم.



# بخش StateProps

در ابتدای Props به شکل زیر می سازیم که دارای یک Field به نام data است که دربرگیرنده اطلاعات دریافتی از API می باشد و یک Method به نام retry برای تلاش مجدد در صورتی که دریافت اطلاعات با مشکل مواجه شود.



## StateProps



```
export default interface StateProps<T> {  
  data?: T;  
  retry?: () => void;  
}
```

# بخش Post

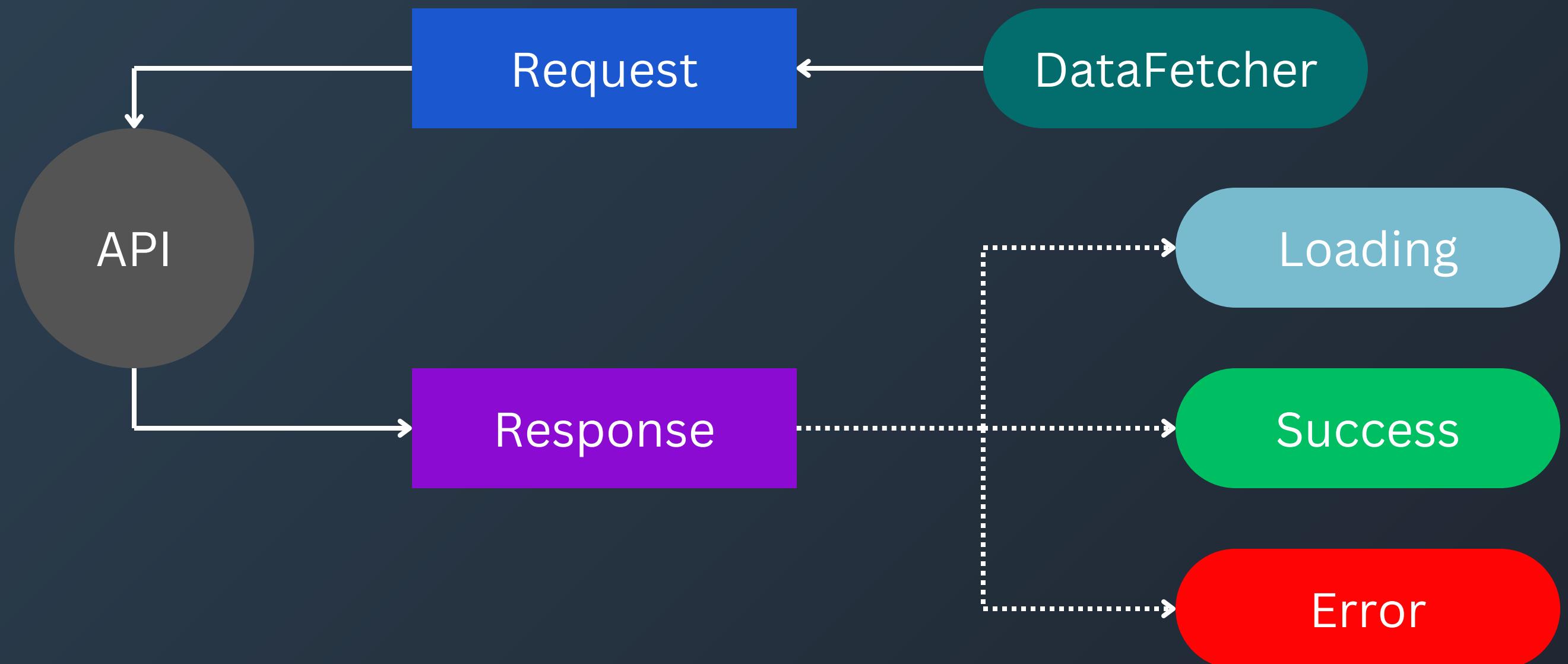
یک Interface برای اطلاعات دریافتی از API به شکل زیر می سازیم.



```
export default interface Post {  
  userId: number;  
  id: number;  
  title: string;  
  body: string;  
}
```

# نتایج API Call کردن

همانطور که اشاره کردیم ممکن است یکی از سه حالت زیر در نتیجه Call API کردن حاصل شود.



از props هیچ استفاده ای نمی کند.

از data برای نمایش اطلاعات استفاده می کند.

از retry برای تلاش مجدد استفاده می کند.

# بخش LoadingState

در این بخش حالت Loading را پیاده سازی می کنیم و همانطوری که مشاهده می کنید از هیچ استفاده ای نکرده ایم.



## LoadingState

```
import StateProps from '../state.props';

const LoadingState: React.FC<StateProps<never>> = () => {
  return <div>Loading...</div>;
};

export default LoadingState;
```

# بخش SuccessState

در این بخش حالت Success را پیاده سازی می کنیم و از data برای نمایش اطلاعات استفاده می کنیم.



## SuccessState

```
import Post from '../post.interface';
import StateProps from '../state.props';

const SuccessState: React.FC<StateProps<Post>> = ({ data }) => {
  return (
    <div>
      <dl>
        <dt>UserIs</dt>
        <dd>{data?.userId}</dd>
        <dt>Id</dt>
        <dd>{data?.id}</dd>
        <dt>Title</dt>
        <dd>{data?.title}</dd>
        <dt>Body</dt>
        <dd>{data?.body}</dd>
      </dl>
      <p>Data loaded successfully!</p>
    </div>
  );
};

export default SuccessState;
```

# بخش ErrorState

در این بخش حالت Error را پیاده سازی می کنیم که ممکن است به دلایل مختلف رخداد و از استفاده می کنیم که کاربر بتواند با کلیک بر روی دکمه Retry برای دریافت اطلاعات تلاش مجدد کند.



## ErrorState

```
import StateProps from '../state.props';

const ErrorState: React.FC<StateProps<never>> = ({ retry }) => {
  return (
    <div>
      Something went wrong. <button onClick={retry}>Retry</button>
    </div>
  );
};

export default ErrorState;
```

# بخش DataFercher

در این بخش یک Component به شکل زیر می سازیم که وظیفه آن Call API و در نهایت نمایش یکی از حالت سه گانه ای که مطرح کردیم، می باشد.



## DataFetcher

```
import { useState, useEffect } from 'react';
import LoadingState from './pattern/states>LoadingState';
import SuccessState from './pattern/states/SuccessState';
import ErrorState from './pattern/states>ErrorState';

const DataFetcher: React.FC = () => {
  const [currentState, setCurrentState] = useState<JSX.Element>(<LoadingState />);

  const fetchData = async () => {
    setCurrentState(<LoadingState />);
    try {
      const response = await fetch('https://jsonplaceholder.typicode.com/posts/1');
      const result = await response.json();
      setCurrentState(<SuccessState data={result} />);
    } catch (error) {
      setCurrentState(<ErrorState retry={fetchData} />);
      console.log(error);
    }
  };

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

  return (
    <div>
      {currentState}
    </div>
  );
};

export default DataFetcher;
```

# طرز استفاده از DataFetcher

در پایان به شکل زیر از DataFetcher استفاده می کنیم.



## StateDesignPattern



```
import DataFetcher from './DataFetcher';

const StateDesignPattern = () => {
  return (
    <div>
      <DataFetcher/>
    </div>
  );
};

export default StateDesignPattern;
```