

Composite

دیزاین پترن ها در React

Structural Patterns

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



Structural Patterns

Composite



الگوی Composite برای ایجاد ساختارهای درختی استفاده می شود، جایی که اشیاء تکی و گروهی به طور یکنواخت رفتار می کنند. این الگو برای نمایش ساختارهای تو در تو (مثل منوها، درخت فایل‌ها، یا نظرات تو در تو) بسیار مفید است.

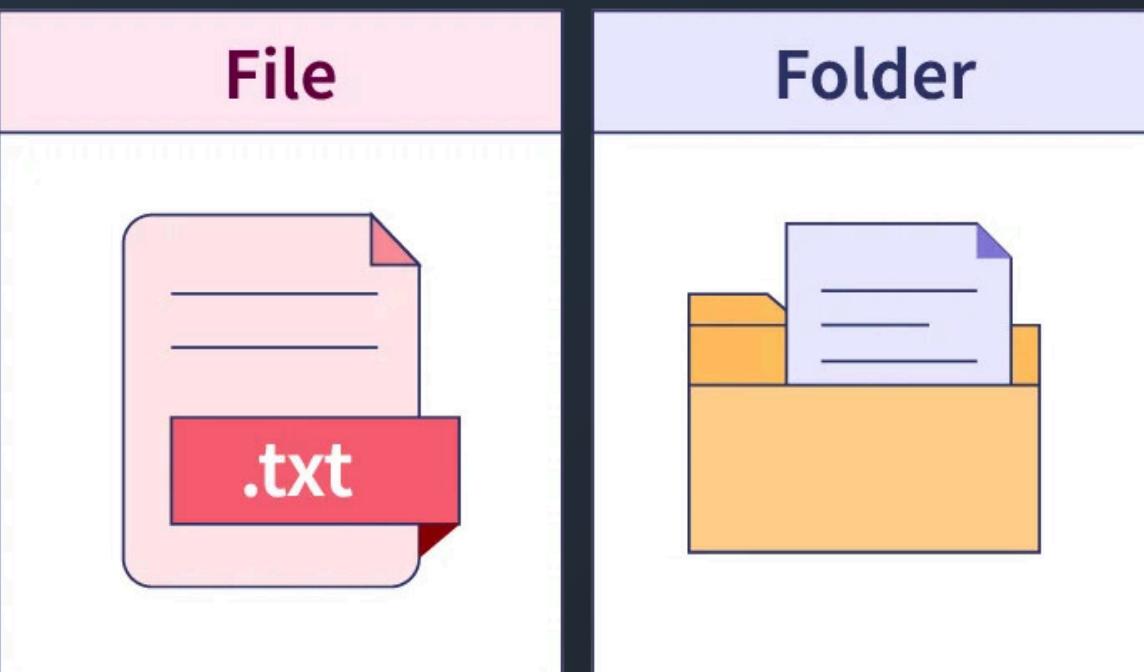
فرض کنید می خواهیم یک کامپونت مانند FileSystem بسازیم که وظیفه آن نمایش فolderها و فایل‌های موجود باشد، در این کامپونت فایل‌ها می توانند داخل فolderها باشند و همچنین به صورت درختی فolderهایی می توانند داخل فolder قرار گیرند.

بخش Types

ابتدا نوع های File و Folder را به شکل زیر تعریف می کنیم.
همچنین نوع FileSystemItem که می تواند File یا Folder باشد.



```
● ● ●  
  
type File = {  
    id: number;  
    name: string;  
    type: 'FILE';  
};  
  
type Folder = {  
    id: number;  
    name: string;  
    type: 'FOLDER';  
    children?: FileSystemItem[];  
};  
  
export type FileSystemItem = File | Folder;
```



بخش FileSystemProps

در این بخش Props مربوط به FileSystem را می سازیم.



FileSystemProps

```
import { FileSystemItem } from "./types";

export interface FileSystemProps {
  item: FileSystemItem;
};
```

بخش FileSystem

در این بخش Component مد نظر خود را پیاده سازی می کنیم.



FileSystem

```
import { FileSystemProps } from "./FileSystem.props";

const FileSystem: React.FC<FileSystemProps> = ({ item }) => {
  if (item.type === 'FILE') {
    return <div style={{ borderLeft: '1px solid #fff' }}>📄 {item.name}</div>;
  }

  return (
    <div>
      <div style={{ fontWeight: 'bold' }}>📁 {item.name}</div>
      <div style={{ paddingLeft: '20px' }}>
        {item.children?.map((child) => (
          <FileSystem key={child.id} item={child} />
        )));
      </div>
    </div>
  );
};

export default FileSystem;
```

ایجاد یک نمونه فرضی از FileSystem

یک نمونه از ساختاری که قرار است به عنوان ورودی به Component بدهیم را به شکل زیر می‌سازیم. این ساختار در برنامه‌های تحت وب توسط سرویس دهنده و در برنامه‌های تحت سیستم عامل می‌تواند توسط شما با دسترسی‌های لازم به دست آید.



```
import { FileSystemItem } from './pattern/FileSystem/types';

const sample: FileSystemItem = {
  id: 1,
  name: 'root',
  type: 'FOLDER',
  children: [
    { id: 2, name: 'file-1.txt', type: 'FILE' },
    {
      id: 3,
      name: 'subFolder-1',
      type: 'FOLDER',
      children: [
        { id: 4, name: 'file-2.txt', type: 'FILE' },
        { id: 5, name: 'file-3.txt', type: 'FILE' },
      ],
    },
  ],
};

export default sample;
```

FileSystem

حال می توانیم به شکل زیر نمونه (Sample) خود را به FileSystem بدهیم و خروجی را مشاهده کنیم.



CompositeDesignPattern

```
import FileSystem from './pattern/FileSystem/FileSystem';
import sample from './sample';

const CompositeDesignPattern = () => {
  return (
    <div>
      <FileSystem item={sample} />
    </div>
  );
};

export default CompositeDesignPattern;
```

