React File Organization Tips

how to show from array of object just name that end with .pdf it can be inside object inside object in react

8/8

To filter and display names that end with `.pdf` from nested objects within an array in React, you can use recursion to handle the nested structure and filter out the names accordingly. Here's a step-by-step approach:

Example

Suppose you have an array of objects like this:

```javascript
const data = [
  {
    id: 1,
    name: "document.pdf",
    files: [
      { id: 2, name: "notes.docx" },
      { id: 3, name: "presentation.pdf" },
    ],
  },
  {
    id: 4,
    name: "image.jpg",
    details: {
      subfiles: [
        { id: 5, name: "report.pdf" },
        { id: 6, name: "summary.txt" },
      ],
    },
  },
];
```

Solution

You can create a recursive function to search through all levels and extract names ending with `.pdf`.

1. **Define a Recursive Function** to traverse nested objects and arrays, filtering for names ending with `.pdf`.
2. **Render the Result** in your component.

Here's the code for this approach:

```javascript
import React from "react";

// Recursive function to find .pdf files
const getPdfNames = (obj) => {
  let pdfNames = [];

  // Iterate through each key in the object or array
  for (const key in obj) {
    const value = obj[key];

    // Check if value is an object or array to apply recursion
    if (typeof value === "object" && value !== null) {
      pdfNames = pdfNames.concat(getPdfNames(value));
    } else if (key === "name" && typeof value === "string" &&
value.endsWith(".pdf")) {
      pdfNames.push(value);
    }
  }

  return pdfNames;
};

const PdfList = ({ data }) => {
  const pdfNames = getPdfNames(data);

  return (
    <div>
      <h2>PDF Files:</h2>
      <ul>
        {pdfNames.map((name, index) => (
          <li key={index}>{name}</li>
        ))}
      </ul>
    </div>
  );
};

// Usage
const App = () => (
  <div>
    <PdfList data={data} />
  </div>
);

export default App;
```

Explanation

1. `getPdfNames` **Function**: It iterates over each item. If it finds an object or array, it calls itself recursively. If it finds a `name` ending with `.pdf`, it adds it to the `pdfNames` array.
2. `PdfList` **Component**: Uses `getPdfNames` to get all `.pdf` names from the `data` prop, then renders each in a list.

This approach will handle any depth of nested objects and arrays.

traverse nested objects and arrays, filtering for names ending with .pdf. give me 3 solution

Here are three different ways to achieve this, each with slightly different techniques for handling nested structures in JavaScript. All these solutions aim to find and return all names ending with `.pdf` from nested objects and arrays.

1. Recursive Traversal (Depth-First Search)

This solution uses recursion to traverse the nested structure. It's straightforward and works well for deeply nested objects.

javascript

```javascript
const getPdfNamesDFS = (obj) => {
  let pdfNames = [];

  for (const key in obj) {
    const value = obj[key];

    if (typeof value === "object" && value !== null) {
      pdfNames = pdfNames.concat(getPdfNamesDFS(value));
    } else if (key === "name" && typeof value === "string" &&
value.endsWith(".pdf")) {
      pdfNames.push(value);
    }
  }

  return pdfNames;
};

// Usage
const pdfNames = getPdfNamesDFS(data);
console.log(pdfNames); // ["document.pdf", "presentation.pdf", "report.pdf"]
```

2. Iterative Stack-Based Traversal

Using an iterative stack approach, this version avoids recursion and uses a stack to handle nested structures, which can be useful for very deeply nested structures where recursion might cause a stack overflow.

javascript

```javascript
const getPdfNamesIterative = (data) => {
  const stack = [data];
  const pdfNames = [];

  while (stack.length > 0) {
    const current = stack.pop();

    for (const key in current) {
      const value = current[key];

      if (typeof value === "object" && value !== null) {
        stack.push(value);
      } else if (key === "name" && typeof value === "string" &&
value.endsWith(".pdf")) {
        pdfNames.push(value);
      }
    }
  }

  return pdfNames;
};

// Usage
const pdfNames = getPdfNamesIterative(data);
console.log(pdfNames); // ["document.pdf", "presentation.pdf", "report.pdf"]
```

3. Using a Helper Function with JSON.stringify and Regex Matching

This method serializes the entire data structure into a JSON string and uses a regular expression to match `.pdf` file names. It's a quick and concise method but has the downside of being less robust for complex object types or functions.

```javascript
const getPdfNamesWithRegex = (data) => {
  const jsonString = JSON.stringify(data);
  const regex = /"name":"(.*?\.pdf)"/g;
  const pdfNames = [];
  let match;

  while ((match = regex.exec(jsonString)) !== null) {
    pdfNames.push(match[1]);
  }

  return pdfNames;
};

// Usage
const pdfNames = getPdfNamesWithRegex(data);
console.log(pdfNames); // ["document.pdf", "presentation.pdf", "report.pdf"]
```

Summary

- **Recursive Traversal:** Simple and effective, especially for moderate nesting levels.
- **Iterative Stack-Based**: Avoids recursion and may work better for very deep nesting levels.
- **Regex-Based on JSON.stringify**: Concise but may not work well with certain complex data structures.

Choose the approach that best fits your project's requirements and data complexity.