Here are the files for the provided dropdown list code:

**index.html**

html

<!DOCTYPE html>

<html>

<head>

<title>Dropdown List</title>

<link rel="stylesheet" type="text/css" href="app.css" />

</head>

<body>

<div id="root"></div>

<script src="index.js"></script>

</body>

</html>

**index.js**

javascript

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

ReactDOM.render(<App />, document.getElementById('root'));

**App.js**

javascript

import React from 'react';

import Dropdown from './Dropdown';

function App() {

return (

<div className="App">

<h1>Dropdown List</h1>

<Dropdown />

</div>

);

}

export default App;

**Dropdown.js**

javascript

import React, { useState } from 'react';

const Dropdown = () => {

const [continent, setContinent] = useState('');

const [fruit, setFruit] = useState('');

const [vegetable, setVegetable] = useState('');

const continents = ['', 'asia', 'europe', 'africa'];

const fruits = {

asia: ['Mango', 'Banana', 'Apple'],

europe: ['Orange', 'Strawberry', 'Pear'],

africa: ['Pineapple', 'Watermelon', 'Grapes'],

};

const vegetables = {

mango: ['Carrot', 'Cabbage', 'Broccoli'],

banana: ['Tomato', 'Cucumber', 'Bell Pepper'],

apple: ['Lettuce', 'Spinach', 'Kale'],

orange: ['Potato', 'Onion', 'Garlic'],

strawberry: ['Celery', 'Eggplant', 'Zucchini'],

pear: ['Radish', 'Mushroom', 'Cauliflower'],

pineapple: ['Pumpkin', 'Sweet Potato', 'Beetroot'],

watermelon: ['Green Bean', 'Asparagus', 'Brussels Sprouts'],

grapes: ['Corn', 'Peas', 'Artichoke'],

};

const handleContinentChange = (e) => {

setContinent(e.target.value);

setFruit('');

setVegetable('');

};

const handleFruitChange = (e) => {

setFruit(e.target.value);

setVegetable('');

};

const handleVegetableChange = (e) => {

setVegetable(e.target.value);

};

const handleSubmit = (e) => {

e.preventDefault();

console.log('Continent:', continent);

console.log('Fruit:', fruit);

console.log('Vegetable:', vegetable);

setContinent('');

setFruit('');

setVegetable('');

};

return (

<form onSubmit={handleSubmit}>

<select value={continent} onChange={handleContinentChange}>

{continents.map((continent) => (

<option key={continent} value={continent}>

{continent ? continent.toUpperCase() : 'Select'}

</option>

))}

</select>

<select value={fruit} onChange={handleFruitChange}>

<option value="">Second Select</option>

{fruits[continent] &&

fruits[continent].map((fruit) => (

<option key={fruit} value={fruit.toLowerCase()}>

{fruit}

</option>

))}

</select>

<select value={vegetable} onChange={handleVegetableChange}>

<option value="">Select Vegetable</option>

{vegetables[fruit] &&

vegetables[fruit].map((vegetable) => (

<option key={vegetable} value={vegetable.toLowerCase()}>

{vegetable}

</option>

))}

</select>

<button type="submit">Submit</button>

</form>

);

};

export default Dropdown;

**app.css**

css

h1 {

text-align: center;

}

select {

display: block;

margin: 20px auto;

padding: 1px;

width: 200px;

}

.highlight {

background-color: lightblue;

}

Make sure to have the necessary React dependencies installed and configure the build system to bundle the JavaScript files using Babel and a bundler like Webpack or Parcel. Then, you can run the application and see the dropdown list in action.

why isn't app test needed

ChatGPT

I apologize for the oversight. The app.test.js file is typically used for writing tests to ensure the functionality and behavior of your application. In the case of the provided dropdown list code, if you want to write tests to verify its functionality, you can create a separate app.test.js file.

Here's a basic example of how the app.test.js file could look like for testing the dropdown functionality using a testing library like Jest:

javascript

import React from 'react';

import { render, screen, fireEvent } from '@testing-library/react';

import App from './App';

describe('Dropdown', () => {

test('selects continent, fruit, and vegetable', () => {

render(<App />);

const continentSelect = screen.getByLabelText('Select');

const fruitSelect = screen.getByLabelText('Second Select');

const vegetableSelect = screen.getByLabelText('Select Vegetable');

const submitButton = screen.getByRole('button', { name: 'Submit' });

// Select continent

fireEvent.change(continentSelect, { target: { value: 'asia' } });

expect(continentSelect.value).toBe('asia');

// Select fruit

fireEvent.change(fruitSelect, { target: { value: 'mango' } });

expect(fruitSelect.value).toBe('mango');

// Select vegetable

fireEvent.change(vegetableSelect, { target: { value: 'carrot' } });

expect(vegetableSelect.value).toBe('carrot');

// Submit form

fireEvent.click(submitButton);

// Assert the selected values were logged

expect(console.log).toHaveBeenCalledWith('Continent:', 'asia');

expect(console.log).toHaveBeenCalledWith('Fruit:', 'mango');

expect(console.log).toHaveBeenCalledWith('Vegetable:', 'carrot');

});

});

This is just a simple example to demonstrate how you can write a test for the dropdown functionality. You can expand on this and write more comprehensive tests based on your specific requirements.