Starting React with Env

How can I start a React app using a .env file?

Starting a React app using a `.env` file involves several steps. The `.env` file is used to define environment variables that can be accessed within your React application. Here's a step-by-step guide:

1. Create a React App

If you haven't already, create a new React app using Create React App:

```sh
npx create-react-app my-app
cd my-app
```

2. Create a .env File

Create a `.env` file in the root directory of your React project (i.e., at the same level as `package.json`).

3. Define Environment Variables

Inside the `.env` file, define your environment variables. Note that environment variables must be prefixed with `REACT_APP_` to be accessible in the React app:

```dotenv
REACT_APP_API_URL=https://api.example.com
REACT_APP_API_KEY=your-api-key
```

4. Access Environment Variables in React

You can access these variables in your React components using `process.env`:

```jsx
import React from 'react';

const App = () => {
  const apiUrl = process.env.REACT_APP_API_URL;
  const apiKey = process.env.REACT_APP_API_KEY;

  console.log('API URL:', apiUrl);
  console.log('API Key:', apiKey);
```

```
    return (
      <div>
        <h1>React App with Environment Variables</h1>
        <p>API URL: {apiUrl}</p>
        <p>API Key: {apiKey}</p>
      </div>
    );
  };

  export default App;
```

5. Run the React App

Start your React app as usual:

```sh
npm start
```

The environment variables defined in the `.env` file will be available in your React app. Make sure to restart the development server if you add or modify the `.env` file while the server is running.

6. Additional Tips

- **Security:** Environment variables in React are embedded into the build, meaning they are not truly secret and will be visible in the browser's developer tools. Do not store sensitive information like passwords or secret keys in environment variables.
- **Environment-Specific Files:** You can create environment-specific files like `.env.development`, `.env.production`, etc., to handle different environments.

Example of Environment-Specific Files

- `.env.development`:

  ```dotenv
  REACT_APP_API_URL=https://dev-api.example.com
  REACT_APP_API_KEY=dev-api-key
  ```

- `.env.production`:

  ```dotenv
  REACT_APP_API_URL=https://prod-api.example.com
  REACT_APP_API_KEY=prod-api-key
  ```

When you run your build or start your server, Create React App will automatically pick the right `.env` file based on the environment.

Conclusion

Using a `.env` file in a React application is straightforward with Create React App. Just remember to prefix your variables with `REACT_APP_` and be cautious about the information you store in these files due to their visibility in the build output.

I have a .env file called ".env.development" and I'm using the below code to try and read from it:

const mostRecentURL = process.env.RECENT_APPS_URL;

Assuming RECENT_APPS_URL is in the .env file, why does my React App not find the enviroment file when I start it with the "npm start" command?

When using Create React App, the `.env` files must follow certain conventions for them to be recognized correctly. Here are some potential reasons why your environment variables might not be getting picked up and some troubleshooting steps:

1. Naming Conventions

Ensure that your environment variable is correctly prefixed. Only variables prefixed with `REACT_APP_` will be loaded by Create React App. For example:

```dotenv
REACT_APP_RECENT_APPS_URL=https://example.com/api/recent
```

2. Correct File Naming

Make sure your `.env.development` file is correctly named and placed in the root of your project directory (i.e., at the same level as `package.json`).

3. Restart the Development Server

If you add or modify the `.env` file while the development server is running, you need to restart the server for the changes to take effect:

```sh
npm start
```

4. Accessing the Variable

Ensure you are accessing the variable correctly in your React code. For example:

```jsx
const mostRecentURL = process.env.REACT_APP_RECENT_APPS_URL;
console.log('Recent Apps URL:', mostRecentURL);
```

5. Verify the Environment

Make sure you are running in the correct environment. The `.env.development` file will only be used if the `NODE_ENV` is set to `development`. The `npm start` command should set this automatically, but you can verify it:

```sh
echo $NODE_ENV
```

In your React app, you can log the environment to ensure it's correct:

```jsx
console.log('Environment:', process.env.NODE_ENV);
```

Example `.env.development` File

Here is a complete example of what your `.env.development` file might look like:

```dotenv
REACT_APP_RECENT_APPS_URL=https://example.com/api/recent
```

Example React Component

Here's how you might access and use the environment variable in your React component:

```jsx
import React from 'react';

const App = () => {
  const mostRecentURL = process.env.REACT_APP_RECENT_APPS_URL;

  console.log('Recent Apps URL:', mostRecentURL);

  return (
    <div>
      <h1>React App with Environment Variables</h1>
      <p>Recent Apps URL: {mostRecentURL}</p>
    </div>
  );
};
```

```
export default App;
```

Additional Checks

- **Environment Variable Availability:** Ensure no typos in the variable name and check for its availability in the browser console.
- **.env File Encoding:** Ensure your `.env` file is encoded in UTF-8 without BOM (Byte Order Mark).

Summary

By prefixing your environment variables with `REACT_APP_`, placing your `.env.development` file correctly, and ensuring you restart your development server, your environment variables should be accessible within your React application.