How have you split your application up into components? Are these components independent and reusable? How do you handle data flow and state management between your components?

Components in my code are Javascript classes that contain the functionality of my weather app. To decide which functionality should be its own component I used the Single Responsibility Principle, i.e., a component should do one thing only. I decided to have exactly three components: Weather, Search and HowToDress. Weather component accepts input (props) and returns weather data. It makes sure that headings are not displayed until the weather data is retrieved. HowToDress displays different pictures; it accepts const temperature and makes a decision about which picture to display based on that. *render()* will display a default picture when no weather is received. Search component is used for displaying a search window description.
Data flow in React is a manner in which data is passed to components. In my weather app, I have a master component App.js that gives *recieveWeatherForecast()* function to a Search component (which serves as an HTML form). When the form is submitted it executes the *recieveWeatherForecast()* with the parameters in the form (city and country names). The function execution is asynchronous so that the browser will not freeze while waiting for the data to be retrieved. When this function completes it sets the state of App.js component. App.js passes its state variables down to Weather and HowToDress. Those two components then accept the variables as props and render them as plain HTML. Data flow besides Search (which sends back information to App.js.) is a one-way flow - Weather and HowToDress don't send back data to their parent component. App.js expects parameters only from Search.
Breaking the app into a component hierarchy makes reusing components in different sections of the application much easier. However, because I declare each of the components in one location only, my app doesn't really reuse any of them. I can't see a place where I could use them in a different way that I described above. On the other hand, most of them execute independently from one another. Only the interaction between App.js and Search might arguably be called dependent since the data flow between them is two-way.