Advanced React Patterns

# Reusable Code - basic

There are 2 things that we can reuse in React:

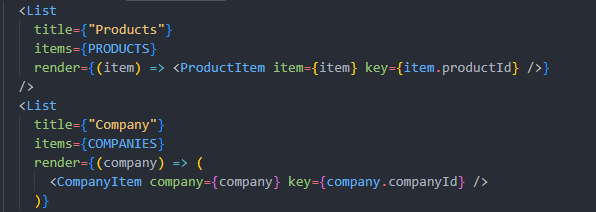
* UI
  + Component and Props
  + Children Prop
* Stateful Logic (logic with hooks)
  + Custom Hooks

# Render Props Pattern – advanced reuse of code

* Complete control by passing a **function** as a **PROP**
* The **Function** tells the component what and how to render

## How to:

1. Use the **render** prop to pass the function we want to use in the **List** component
   1. We pass the function for the **map** logic



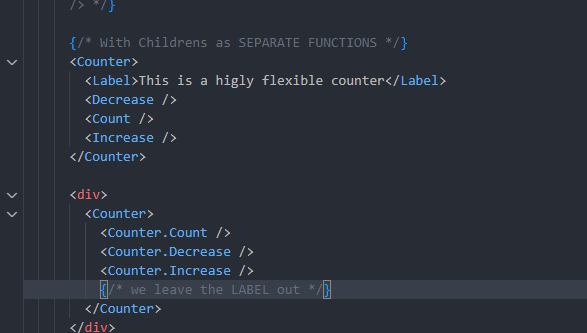
# Compound Component Pattern – advanced reuse of code

* For very self-contained components that need to manage their own state
* Fancy super-components
* We create a **Parent** component and then some **Child** components that will make sense only in use with the **Parent** component
* This will allow us to create **highly usable components** with an expressive API

## How

1. The **Parent component** will keep the **STATE**
2. We will use the **CONTEXT API** to pass **STATE** to children Components





## Modal in Wild Oasis App

1. We create a new component AddCabin
2. Use the Modal component
3. Use the **children** prop on the Modal to pass the FORM component
   1. We add a new prop to the Form in CreateCabinForm
      1. Type
      2. To know when the form is rendered in a modal
      3. We use the onCloseModal prop to check if we are in a modal or not
      4. A screen shot of a computer

         Description automatically generated
      5. To **ADJUST the form** we use the StyledComponents power to adjust based on props

## React Portal

* React Feature to use with components that we want to be **OVER** other components
* The Component **TREE** structure remains the same, only the physical **DOM changes**
* Generally used with **MODALS, tooltips, menus etc.**
* **Usage**
  + **createPortal()** hook

A screen shot of a computer program

Description automatically generated

## Compound Component – Modal

1. The Modal Component will hold **ITS OWN STATE**
   1. It will **DISPLAY THE OPEN BUTTON**
   2. It will take care of when it’s open or closed
   3. It will **not** use OUTSIDE state like previous AddCabin
2. Create the Parent **Modal**
   1. Create the state
   2. Create the handlers
   3. Create the Context
3. Create the Children Components
   1. Open
      1. This will have a button that will tell the Modal what Window to Open
      2. We use **cloneElement** from React to **add props to childrens**
      3. We pass the onClick prop and the setWindowName to the prop of the **Open** button
      4. A screen shot of a computer code

         Description automatically generated
   2. Window
      1. If the name of the window is different that what we want to Open, we return null
      2. The window will contain the **FORM** as a **cloned element**  so that we can pass the  **onCloseModal prop**
      3. A screen shot of a computer program

         Description automatically generated
      4. A screen shot of a computer program

         Description automatically generated