React Design Patterns

أول حاجة وهي الـ Render Props واللي بتستخدم مصطلح اسمه Inversion Of Control(IOC)

RENDER PROPS: **the render prop pattern is all about passing in a prop called render, which is a function that a component uses, to know what it should render and how to do it.**

<ul className="list">

  {displayItems.map((product) => (

    <ProductItem key={product.productName} product={product} />

  ))}

</ul>

دلوقتي انا بداخل الـ List Function Component بعمل loop لعرض المنتجات بالـ map بالتالي دي النتيجة

A screenshot of a computer

Description automatically generated

طيب لو انا عايز اعمل reuse لنفس الـ map دا لعرض المنتجات برضو هنا بقا هستخدم الـ IOC

export default function App() {

  return (

    <div>

      <h1>Render Props Demo</h1>

      <div className="col-2">

        <List

          title="Products"

          items={products}

          render={(product) => (

            <ProductItem key={product.productName} product={product} />

          )}

        />

      </div>

    </div>

  );

}

بالتالي داخل الـ app دا هعمل props وجواها هعمل الـ loop و الـ render props دي هحطها هي نفسها كدا داخل الـ map function الموجودة في الـ List Component function

<ul className="list">{displayItems.map(render)}</ul>

وبناءا عليه اقدر استخدم الـ List Component function دي تاني داخل الـ app component بشكل مختلف

        <List

          title="companies"

          items={companies}

          render={(company) => (

            <CompanyItem

              key={company.companytName}

              company={company}

              defaultVisibility={false}

            />

          )}

        />

A screenshot of a computer

Description automatically generatedلاني في الـ map مباصي حاجة اسمها render وبناءا بقا ع الـ render دا وما يحتويه هيتم استخدام الـ map في عرضه برضو

بس علي كل حال الـ react hooks حلت الموضوع دا اصلا انت بس بتاخد فكرة مش اكتر عن الـ patterns

365- A Look at Higher-Order Components (HOC)

تاني pattern معايا الـ HOC

HIGHER ORDER COMPONENTS: **a Higher Order Component is simply a component that takes in another component and then returns a new component that is better, so an enhanced version of the initial component.**

الـ HOC بختصار بتمكنك انك لو عندك third part library مش هتقدر تعدل اي حاجة فيها حاجة انك تدخلها في component تانيه فيها التعديل اللي انت عايزو وتطلعلي بـ Component تالتة جديدة جايبة من الأخر بقا

يعني علي سبيل المثال انا عندي الـ component دي بتعرض المنتجات

// LATER: Let's say we got this component from a 3rd-party library, and can't change it. But we still want to add the 2 toggle functionalities to it

function ProductList({ title, items }) {

  return (

    <ul className="list">

      {item.map((product) => (

        <ProductItem key={product.productName} product={product} />

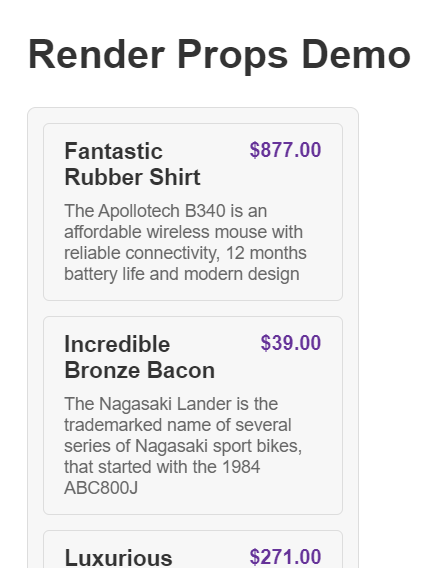
      ))}

    </ul>

  );

}

وهستدعيها دلوقتي في الـ App component



فظهرت بالشكل دا من غير الـ Toggles اللي بتفتحها

انا بقا عندي component تانية

import { useState } from "react";

export default function withToggles(WrappedComponent) {

  return function List(props) {

    const [isOpen, setIsOpen] = useState(true);

    const [isCollapsed, setIsCollapsed] = useState(false);

    const displayItems = isCollapsed ? props.items.slice(0, 3) : props.items;

    function toggleOpen() {

      setIsOpen((isOpen) => !isOpen);

      setIsCollapsed(false);

    }

    return (

      <div className="list-container">

        <div className="heading">

          <h2>{props.title}</h2>

          <button onClick={toggleOpen}>

            {isOpen ? <span>&or;</span> : <span>&and;</span>}

          </button>

        </div>

        {isOpen && <WrappedComponent {...props} items={displayItems} />}

        <button onClick={() => setIsCollapsed((isCollapsed) => !isCollapsed)}>

          {isCollapsed ? `Show all ${props.items.length}` : "Show less"}

        </button>

      </div>

    );

  };

}

هتاخد الـ ProductList component اللي فوق تضيف عليها مميزات الـ toggles دي وتطلعلي بميكس بيهم بقا يبقي دا Component تالت وجديد اسمه withToggles

const ProductListWithToggels = withToggles(ProductList);

استدعيت الـ Component بتاعتي وبصيت الـ argument جواها اللي هي الـ ProductList القديمة اللي هتاخدها تضيف عليها الـ toggles وترجعهالي اخزنها في المتغير اللي اسمها ProductListWithToggles

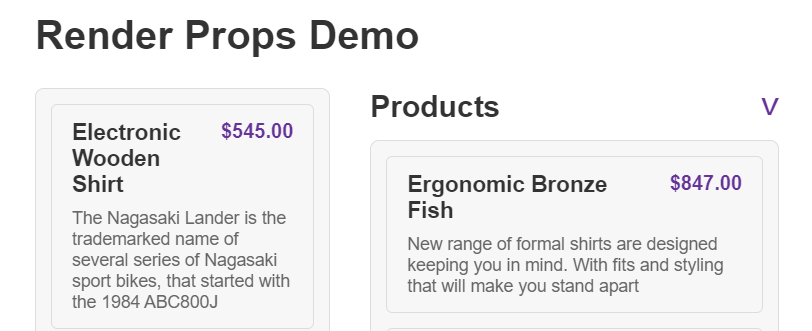
<div className="col-2">

  <ProductList items={products} title="Products" />

  <ProductListWithToggels items={products} title="Products" />

</div>

فلما اجي استخدم الـ version القديم و استخدم الـ HOC بقا تحته اللي ضيفت فيه الـ Toggles تبقي دي النتيجة



وبكدا اكون طبقت مبدأ الـ HOC

بس علي كل حال الـ react hooks حلت الموضوع دا اصلا انت بس بتاخد فكرة مش اكتر عن الـ patterns

366- The Compound Component Pattern

Compound Component: **the idea of a Compound Component is that we can create a set of related components that together achieve a common and useful task,**

import { createContext, useState, useContext } from "react";

// 1) Create a context

const CounterContext = createContext();

// 2) Create a parent component

function Counter({ children }) {

  const [count, setCount] = useState(0);

  const increase = () => setCount(count + 1);

  // const increase = () => setCount((count) => count + 1);

  const decrease = () => setCount(count - 1);

  // const decrease = () => setCount((count) => count - 1);

  return (

    <CounterContext.Provider value={{ count, increase, decrease }}>

      <span>{children}</span>

    </CounterContext.Provider>

  );

}

// 3) Create a child component that uses the context

function Count() {

  const { count } = useContext(CounterContext);

  return <span>{count}</span>;

}

function Label({ children }) {

  return <span>{children}</span>;

}

function Increase({ icon }) {

  const { increase } = useContext(CounterContext);

  return <button onClick={increase}>{icon}</button>;

}

function Decrease({ icon }) {

  const { decrease } = useContext(CounterContext);

  return <button onClick={decrease}>{icon}</button>;

}

// 4) Add child components as proeprties to parent component

Counter.Count = Count;

Counter.Label = Label;

Counter.Increase = Increase;

Counter.Decrease = Decrease;

export default Counter;

   <Counter>

        <Counter.Label>My super flexible counter</Counter.Label>

        <Counter.Decrease icon="-" />

        <Counter.Increase icon="+" />

        <Counter.Count />

      </Counter>

367- Building a Modal Window Using a React Portal

Modal.jsx

import { HiXMark } from "react-icons/hi2";

import styled from "styled-components";

const StyledModal = styled.div`

  position: fixed;

  top: 50%;

  left: 50%;

  transform: translate(-50%, -50%);

  background-color: var(--color-grey-0);

  border-radius: var(--border-radius-lg);

  box-shadow: var(--shadow-lg);

  padding: 3.2rem 4rem;

  transition: all 0.5s;

`;

const Overlay = styled.div`

  position: fixed;

  top: 0;

  left: 0;

  width: 100%;

  height: 100vh;

  background-color: var(--backdrop-color);

  backdrop-filter: blur(4px);

  z-index: 1000;

  transition: all 0.5s;

`;

const Button = styled.button`

  background: none;

  border: none;

  padding: 0.4rem;

  border-radius: var(--border-radius-sm);

  transform: translateX(0.8rem);

  transition: all 0.2s;

  position: absolute;

  top: 1.2rem;

  right: 1.9rem;

  &:hover {

    background-color: var(--color-grey-100);

  }

  & svg {

    width: 2.4rem;

    height: 2.4rem;

    /\* Sometimes we need both \*/

    /\* fill: var(--color-grey-500);

    stroke: var(--color-grey-500); \*/

    color: var(--color-grey-500);

  }

`;

const Modal = ({ children, onClose }) => {

  return (

    <Overlay>

      <StyledModal>

        <Button onClick={onClose}>

          <HiXMark />

        </Button>

      <div>{children}</div>

      </StyledModal>

    </Overlay>

  );

};

export default Modal;

AddCabin.jsx

import { useState } from "react";

import Button from "../../ui/Button";

import Modal from "../../ui/Modal";

import CreateCabinForm from "./CreateCabinForm";

const AddCabin = () => {

  const [isOpenModal, setIsOpenModal] = useState(false);

  return (

    <div>

      <Button size="custom" onClick={() => setIsOpenModal(!isOpenModal)}>

        Add new Cabin

      </Button>

      {isOpenModal && (

        <Modal onClose={() => setIsOpenModal(false)}>

          <CreateCabinForm onCloseModal={() => setIsOpenModal(false)} />

        </Modal>

      )}

    </div>

  );

};

export default AddCabin;

**Cabin.jsx**

function Cabins() {

  return (

    <>

      <Row type="horizontal">

        <Heading as="h1">All cabins</Heading>

        <p>Filter / Sort </p>

      </Row>

      <Row>

        <CabinTable />

        <AddCabin />

      </Row>

    </>

  );

}

CreateCabinForm.jsx

  <Form onSubmit={handleSubmit(onSubmit)} type={onCloseModal ? "modal" : "regular"} >

<FormRow>

        {/\* type is an HTML attribute! \*/}

        <Button

          variation="secondary"

          type="reset"

          onClick={() => onCloseModal?.()}

        >

          Cancel

        </Button>

        <Button disabled={isWorking}>

          {isEditSession ? "Edit cabin" : "Create new cabin"}

        </Button>

      </FormRow>

    </Form>

ليه السطر دا كدا ؟!   
عشان الـ form دا ممكن يكون مش داخل الـ Modal في مكان تاني بالتالي الزرار دا مش هيستلم الـ Props بتاعت الـ onCloseModal دي !   
بالتالي وجودها خارج الـ modal في الفورم في مكان تاني ممكن يعمل error لذلك عملت الـ optional chain دي

From.jsx

import styled, { css } from "styled-components";

const Form = styled.form`

  ${(props) =>

    props.type === "regular" &&

    css`

      padding: 2.4rem 4rem;

      /\* Box \*/

      background-color: var(--color-grey-0);

      border: 1px solid var(--color-grey-100);

      border-radius: var(--border-radius-md);

    `}

  ${(props) =>

    props.type === "modal" &&

    css`

      width: 80rem;

    `}

  overflow: hidden;

  font-size: 1.4rem;

`;

Form.defaultProps = {

  type: "regular",

};

export default Form;

REACT PORTAL: **React portal is a feature that essentially allows us to render an element outside of the parent component's DOM structure while still keeping the element in the original position of the component tree. So in other words, with a portal we can basically render a component in any place that we want inside the DOM tree but still leave the component at the same place in the React component tree. And so then things like props keep working normally. And so this is great and generally used for all elements that we want to stay on top of other elements. So things like modal windows, tool tips, menus and so on.**

import { createPortal } from "react-dom";

const Modal = ({ children, onClose }) => {

  return createPortal(

    <>

      <Overlay onClick={onClose} />

      <StyledModal>

        <Button onClick={onClose}>

          <HiXMark />

        </Button>

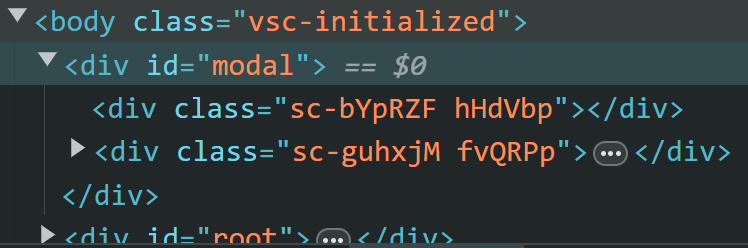
        <div>{children}</div>

      </StyledModal>

    </>,

    document.getElementById("modal")

  );

};

وبكدا اكون حطيت الـ Overlay و الـ Modal برا الـ Root element

368 - Converting the Modal to a Compound Component

import Button from "../../ui/Button";

import Modal from "../../ui/Modal";

import CabinTable from "./CabinTable";

import CreateCabinForm from "./CreateCabinForm";

function AddCabin() {

  return (

Children   
inside   
Modal

    <Modal>

      <Modal.Open opens="cabin-form">

        <Button>Add new Cabin</Button>

      </Modal.Open>

      <Modal.Window name="cabin-form">

        <CreateCabinForm />

      </Modal.Window>

      <Modal.Open opens="table">

        <Button>Show table</Button>

      </Modal.Open>

      <Modal.Window name="table">

        <CabinTable />

      </Modal.Window>

    </Modal>

  );

}

export default AddCabin;

انا جوا الـ Modal عملت اتنين مختلفين عن بعض كل واحد ليه prop معينة وكل واحد ليه name تخصه

عشان ابدأ استخدم الـ Compound Component لازم اولا استخدم الـ useContext

//1. Create Context

const ModalContext = createContext();

// 2. Create parent component

function Modal({ children }) {

  const [openName, setOpenName] = useState("");

  const close = () => setOpenName("");

  const open = setOpenName;

  return (

    <ModalContext.Provider value={{ openName, close, open }}>

      {children}

    </ModalContext.Provider>

  );

}

الـ useState هنا هتبقي string فاضي لان بناءا علي اسم الـ props بتاعت open فوق هيتحدد مين الـ Window  
اللي هتتفتح اصلا

1. عملت close function لما يحصلها trigger ترجع الـ openName state بـ string فاضي تاني
2. الـ open بقا هتبقي بقيمة الـ setOpenName
3. عملت provide للداتا دي كلها للـ Children اللي جوا الـ Modal سواء الـ Open أو الـ Window

// 3. Create child components to help implementing the common task

function Open({ children, opens: opensWindowName }) {

  const { open } = useContext(ModalContext);

  return cloneElement(children, { onClick: () => open(opensWindowName) });

}

أول Helper function عندي اللي هي الـ Open .. اللي فيما بعد هباصيها للـ Modal.Open تحت ...

هي خدت الـ Children اللي جوا الـ Modal.Open من الـ AddCabin.jsx Component  
وكمان اسم الـ props اللي كان اسمها Opens وبس يعني عملنا ليها alias او nickname مميز

جواها بقا استخرجنا الـ open function من الـ ModalContext اللي هي دي

 const open = setOpenName;

وقولتله رجعلي بالـ cloneElment دي نفس الـ children اللي جواها وضيف عليه الـ onClick دي اللي هيستخدم الـ open function وياخد الـ اسم props اللي فيه Modal.Open كـ openName

const Window = ({ children, name }) => {

  const { close, openName } = useContext(ModalContext);

  if (name !== openName) return null;

  return createPortal(

    <Overlay>

      <StyledModal>

        <Button onClick={close}>

          <HiXMark />

        </Button>

        <div>{cloneElement(children, { onCloseModal: close })}</div>

      </StyledModal>

    </Overlay>,

    document.getElementById("modal")

  );

};

بعد كدا الـ Window

وفيه بطلع الـ close function اللي هي دي

  const close = () => setOpenName("");

وبطلع الـ openName بناءا علي قيمته هل هو form ولا cabin بناءا برضو ع الـ Props اللي موجودة في   
الـ AddCabin.jsx

وبقوله لو الـ name لا يساوي الـ openName يبقي رجعلي null اقف لحد هنا متعملش حاجة   
طيب الـ name دي جت منين جت برضو من الـ props اللي في الـ AddCabin.jsx

      <Modal.Open opens="cabin-form">

        <Button>Add new Cabin</Button>

      </Modal.Open>

      <Modal.Window name="cabin-form">

        <CreateCabinForm />

      </Modal.Window>

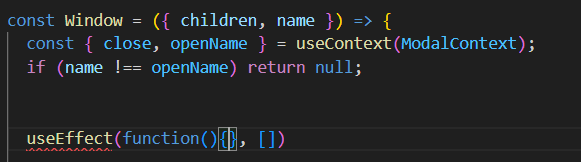
بعدين بنفس الكيفية بالظبط عملت الـ portal بتاعي وشكرا

369- Detecting a Click Outside the Modal

دلوقتي انا عايز اطبق الـ close للـ modal بمجرد ما ادوس برا الـ modal لما ادوس في الـ overlay يعني

عشان اعمل دا محتاج اطبق شوية حاجات بالترتيب من ضمنها اني هستخدم الـ DOM تاني و الـ events

أول حاجة مدام هستخدم الـ Dom و الـ Events يبقي جوا الـ useEffect



A screenshot of a computer program

Description automatically generated

عملي المشكلة دي لان استخدمته بعد الـ return null لازم يبقي قبله

  const ref = useRef();

<Overlay>

      <StyledModal ref={ref}>

انا بالـ ref دي كدا مسكت الـ Modal Window بتاعتي

useEffect(

    function () {

      function handleClick(e) {

        if (ref.current && !ref.current.contains(e.target)) {

          console.log("Clicked OutSide Modal Window");

          close();

        }

      }

      document.addEventListener("click", handleClick, true);

      return () => document.removeEventListener("click", handleClick, true);

    },

    [close]

  );

اي بقا اللي حصل هنا بالترتيب كدا جوا الـ useEffetct انا عملت handleClick function

وظيفتها تـ call الـ close()

  const close = () => setOpenName("");

بس بشوية شروط

اولا لو الـ ref.current اللي هو الـ Modal بشكل عام   
 و الـ ref.current.contains(e.traget) دي مهمة جدا ... لان الـ e.traget دا هو المكان اللي هيتم الضغط عليه في الصفحة

فانا بقولة بالـ ref.current دي ان لو الـ Modal موجود

كمان بقوله لو الـ click حصل وكان الـ current دا اي عنصر جوا الـ modal يعني الـ click دا محصلش جوا عناصر الـ Modal ولكن شرط اساسي انه يكون حصل براها ساعتها call الـ close()

في مرحلة الـ unMount phase بقا بعد ما الـ click يحصل لغيته تاني باستخدام الـ removeEventListener

اما الـ true بقا اللي في الـ DOM دي عشان ولا اقولك هشرحها بعد الفطار

الـ true اللي في نهاية الـ Dom line دي بتحل مشكلة عندي وهي انه بدونها انا كدا اصلا لوجيكالي برا الـ Modal  
وبالتالي بيحصل ان الـ Modal تفتح وتقفل في لحظة

لذا الـ Ture دي غرضها انك تطبق الـ Click Event في الـ capturing phase وليس في الـ phase bubbling

useOutsideClick.js – Custom Hook

import { useEffect, useRef } from "react";

export const useOutsideClick = (handler, listenCapturing = true) => {

  const ref = useRef();

  useEffect(

    function () {

      function handleClick(e) {

        if (ref.current && !ref.current.contains(e.target)) {

          console.log("Clicked OutSide Modal Window");

          handler();

        }

      }

      document.addEventListener("click", handleClick, listenCapturing);

      return () =>

        document.removeEventListener("click", handleClick, listenCapturing);

    },

    [handler, listenCapturing]

  );

  return ref;

};

انا بعتله الـ close

  const ref = useOutsideClick(close);

370- Confirming Cabin Deletions

CabinRow.jsx

  <Modal.Open opens="delete">

  <button>

  <HiTrash />

  </button>

</Modal.Open>

  <Modal.Window name="delete">

  <ConfirmDelete

    resourceName="cabins"

      disabled={isDeleting}

      onConfirm={() => deleteCabin(cabinId)}

     />

   </Modal.Window>

ConfirmDelete.jsx

import styled from "styled-components";

import Button from "./Button";

import Heading from "./Heading";

const StyledConfirmDelete = styled.div`

  width: 40rem;

  display: flex;

  flex-direction: column;

  gap: 1.2rem;

  & p {

    color: var(--color-grey-500);

    margin-bottom: 1.2rem;

  }

  & div {

    display: flex;

    justify-content: flex-end;

    gap: 1.2rem;

  }

`;

function ConfirmDelete({ resourceName, onConfirm, disabled, onCloseModal }) {

  return (

    <StyledConfirmDelete>

      <Heading as="h3">Delete {resourceName}</Heading>

      <p>

        Are you sure you want to delete this {resourceName} permanently? This

        action cannot be undone.

      </p>

      <div>

        <Button

          variation="secondary"

          disabled={disabled}

          onClick={onCloseModal}>

          Cancel

        </Button>

        <Button variation="danger" disabled={disabled} onClick={onConfirm}>

          Delete

        </Button>

      </div>

    </StyledConfirmDelete>

  );

}

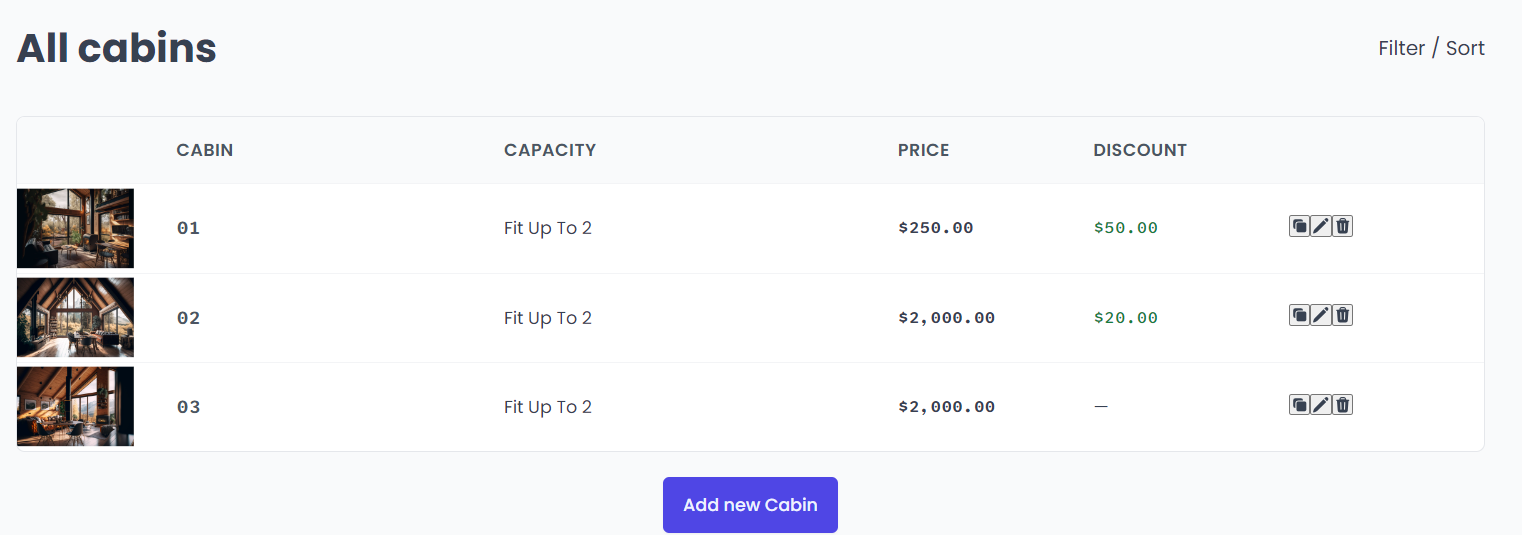
export default ConfirmDelete;

A screenshot of a computer

Description automatically generated

371 - Building a Reusable Table

دلوقتي الـ Table اللي عندي دي مش Reusable ودا بسبب اني كاتب هارد كود فيه مقدرش اعيد استخدامه



  grid-template-columns: 0.6fr 1.8fr 2.2fr 1fr 1fr 1fr;

فانا عايز اعمل Component كدا اباصيله اي مقاسات يتسخدمها بحيث اني اقدر اعيد استخدام الـ Table دا اكتر من مرة نفس الكود ولكن بمقاسات مختلفة عن بعض

فانا عندي الـ Styles دي مكتوبة بالفعل في Component اسمه Table.jsx

import styled from "styled-components";

const StyledTable = styled.div`

  border: 1px solid var(--color-grey-200);

  font-size: 1.4rem;

  background-color: var(--color-grey-0);

  border-radius: 7px;

  overflow: hidden;

`;

const CommonRow = styled.div`

  display: grid;

  grid-template-columns: ${(props) => props.columns};

  column-gap: 2.4rem;

  align-items: center;

  transition: none;

`;

const StyledHeader = styled(CommonRow)`

  padding: 1.6rem 2.4rem;

  background-color: var(--color-grey-50);

  border-bottom: 1px solid var(--color-grey-100);

  text-transform: uppercase;

  letter-spacing: 0.4px;

  font-weight: 600;

  color: var(--color-grey-600);

`;

const StyledRow = styled(CommonRow)`

  padding: 1.2rem 2.4rem;

  &:not(:last-child) {

    border-bottom: 1px solid var(--color-grey-100);

  }

`;

const StyledBody = styled.section`

  margin: 0.4rem 0;

`;

const Footer = styled.footer`

  background-color: var(--color-grey-50);

  display: flex;

  justify-content: center;

  padding: 1.2rem;

  /\* This will hide the footer when it contains no child elements. Possible thanks to the parent selector :has 🎉 \*/

  &:not(:has(\*)) {

    display: none;

  }

`;

const Empty = styled.p`

  font-size: 1.6rem;

  font-weight: 500;

  text-align: center;

  margin: 2.4rem;

`;

الـ Component دا اللي هيكون عبارة عن الـ Compound Component ودي كدا أول خطوة

//1- Create Context

const TableContext = createContext();

//2- Create Parent

function Table({ children }) {

  return <TableContext.Provider>{children}</TableContext.Provider>;

}

//3- Create Child

function Header() {}

function Row() {}

//4- Pass Child as Props To Parent

Table.Header = Header;

Table.Row = Row;

export default Table;

//1- Create Context

const TableContext = createContext();

//2- Create Parent

function Table({ children, columns }) {

  return (

    <TableContext.Provider value={{ columns }}>

      <StyledTable role="table">{children}</StyledTable>

    </TableContext.Provider>

  );

}

//3- Create Child

function Header({ children }) {

  const { columns } = useContext(TableContext);

  return (

    <StyledHeader role="row" as="header" columns={columns}>

      {children}

    </StyledHeader>

  );

}

function Row({ children }) {

  const { columns } = useContext(TableContext);

  return (

    <StyledRow role="row" columns={columns}>

      {children}

    </StyledRow>

  );

}

function Body({ children }) {}

//4- Pass Child as Props To Parent

Table.Header = Header;

Table.Row = Row;

Table.Body = Body;

Table.Footer = Footer;

export default Table;

1. انا عملت الـ Context عادي
2. عملت الـ Parent وباصيت جواه الـ columns كـ value اللي انا اصلا هبعتها للـ Table دا من   
   الـ Component التاني

CabinTable.jsx ==> sending columns as props to Table.jsx

    <Table columns="0.6fr 1.8fr 2.2fr 1fr 1fr 1fr">

1. عملت محاوطة للـ children اللي هيبقو جوا الـ Table دا بالـ StyledTable بالـ role = ‘table’
2. بعد كدا الـ children اللي جوا الـ Table اللي هما الـ Header و الـ Row ففي الـ Header استملت الـ columns  
   اللي كنت عملت provide بيها للـ children وبعدين باصيته للـ component اللي اسمه StyledHeader
3. نفس الفكرة في الـ Row باصيت او عملت provide للـ Row دا بالـ columns دي

طبعا الاتنين Row و الـ Header هيكون جواها children بس خلينا نفهم الاول قصة الـ columns اللي انا باصيتها جو الـ Row و الـ Header دول خصوصا ان الـ styled components ليها دور في الموضوع

بص يا عم

const CommonRow = styled.div`

  display: grid;

  grid-template-columns: ${(props) => props.columns};

  column-gap: 2.4rem;

  align-items: center;

  transition: none;

`;

الـ commonRow دا مجرد styledComponent ليه بعض التنسيقات بس اهم حاجة فيه ان   
الـ grid-templates-columns دي خاصية هتستلم جواها props او قيم من برا

تاني حاجة

const StyledHeader = styled(CommonRow)`

  padding: 1.6rem 2.4rem;

  background-color: var(--color-grey-50);

  border-bottom: 1px solid var(--color-grey-100);

  text-transform: uppercase;

  letter-spacing: 0.4px;

  font-weight: 600;

  color: var(--color-grey-600);

`;

const StyledRow = styled(CommonRow)`

  padding: 1.2rem 2.4rem;

  &:not(:last-child) {

    border-bottom: 1px solid var(--color-grey-100);

  }

`;

الـ styledHeader و الـ StyledRow بيستلمو تنسيقات الـ commonRow دا وزي ما تقول كدا بيحصله apply جواها يعني التنسيقات دي بتندمج مع الـ StyledHeader

1. اخر حاجة بباصي الـ Child components دي كـ properties للـ Parent

بروح بعد كدا للـ CabinTable.jsx استخدم الـ Table بتاع الـ Compound وجواه باستخدم الـ Header

return (

    <Table columns="0.6fr 1.8fr 2.2fr 1fr 1fr 1fr">

      <Table.Header>

        <div></div>

        <div>cabin</div>

        <div>capacity</div>

        <div>price</div>

        <div>Discount</div>

        <div></div>

      </Table.Header>

      {cabins.map((cabin) => (

        <CabinRow cabin={cabin} key={cabin.id} />

      ))}

    </Table>

  );

ثم الـ CabinRow.jsx

 return (

 <Table.Row>

      <Img src={image} />

      <Cabin>{name}</Cabin>

      <div>Fit Up To {maxCapacity}</div>

      <Price>{formatCurrency(regularPrice)}</Price>

      {discount ? (

        <Discount>{formatCurrency(discount)}</Discount>

      ) : (

        <span>&mdash;</span>

      )}

      <div>

        <button disabled={isCreating} onClick={handleDuplicate}>

          <HiSquare2Stack />

        </button>

        <Modal>

          <Modal.Open opens="edit">

            <button>

              <HiPencil />

            </button>

          </Modal.Open>

          <Modal.Window name="edit">

            <CreateCabinForm cabinToEdit={cabin} />

          </Modal.Window>

          <Modal.Open opens="delete">

            <button>

              <HiTrash />

            </button>

          </Modal.Open>

          <Modal.Window name="delete">

            <ConfirmDelete

              resourceName="cabins"

              disabled={isDeleting}

              onConfirm={() => deleteCabin(cabinId)}

            />

          </Modal.Window>

        </Modal>

      </div>

</Table.Row>

  );

    <Table columns="0.6fr 1.8fr 2.2fr 1fr 1fr 1fr">

دلوقتي اقدر العب في القيم دي برحتي واكستميز الـ table علي كيفي

372- Applying the Render Props Pattern

  return (

    <Table columns="0.6fr 1.8fr 2.2fr 1fr 1fr 1fr">

      <Table.Header>

        <div></div>

        <div>cabin</div>

        <div>capacity</div>

        <div>price</div>

        <div>Discount</div>

        <div></div>

      </Table.Header>

    <Table.Body

        data={cabins}

        render={(cabin) => <CabinRow cabin={cabin} key={cabin.id} />}

    />

    </Table>

function Body({ render, data }) {

  if (!data.length) return <Empty>No data to show at this moment</Empty>;

  return <StyledBody>{data.map(render)}</StyledBody>;

}

Table.Body = Body;

A screenshot of a computer

Description automatically generated

373- Building a Reusable Context Menu