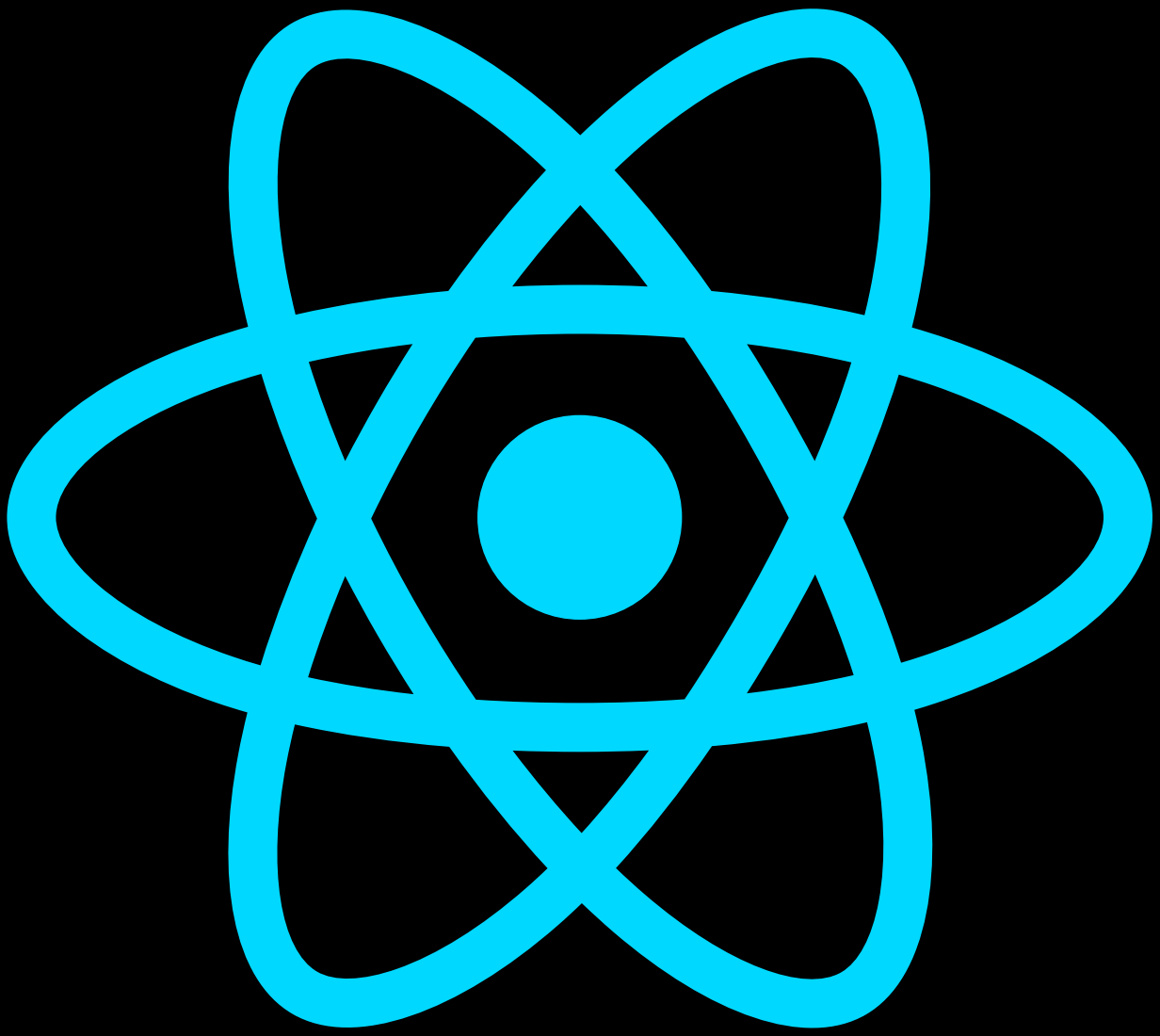
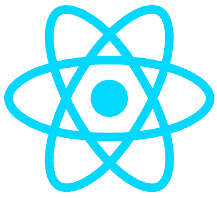


React Js Notes



React Introduction

The library for web and native user interfaces

Watch On YouTube

✍️👋 React Introduction

- 1. React is a JavaScript library for building user interfaces.
- 2. React is used to build single-page applications.
- 3. React allows us to create reusable UI components.

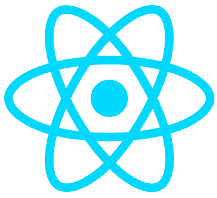
🔗 What is React?

- 1. React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.
- 2. React is a tool for building UI components.

🔗 How does React Work?

- 1. React creates a VIRTUAL DOM in memory.
- 2. Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.
- 4. React only changes what needs to be changed!

- 5. React finds out what changes have been made, and changes only what needs to be changed.



React Setup

The library for web and native user interfaces

Watch On YouTube

✍️👋 React Setup

React has been designed from the start for gradual adoption. You can use as little or as much React as you need. Whether you want to get a taste of React, add some interactivity to an HTML page, or start a complex React-powered app, this section will help you get started.

🔗 Start a New React Project

If you want to build a new app or a new website fully with React, we recommend picking one of the React-powered frameworks popular in the community. Frameworks provide features that most apps and sites eventually need, including routing, data fetching, and generating HTML.

Note

You need to install Node.js for local development. You can also choose to use Node.js in production, but you don't have to. Many React frameworks support export to a static HTML/CSS/JS folder.

🔗 Installation React without a framework

This is recommended for beginners

With NPM:

```
npm create vite@latest
```

```
cd my-project

npm install
npm run dev
```

With Yarn:

```
yarn create vite
```


JavaScript Refresher

The library for web and native user interfaces

Watch On YouTube

✍️👋 JavaScript Refresher

🔗 What is ES6?

ES6 stands for ECMAScript 6.

ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript, it was published in 2015, and is also known as ECMAScript 2015.

🔗 Why Should I Learn ES6?

React uses ES6, and you should be familiar with some of the new features like:

```
- React ES6 Arrow Functions
- React ES6 Variables
- React ES6 Array Methods
- React ES6 Array Methods
- React ES6 Spread Operator
```

```
- React ES6 Modules
- React ES6 Ternary Operator
```

✍️👋 React ES6 Arrow Functions

🔗 Arrow Functions

Arrow functions allow us to write shorter function syntax:

Example

Before:

```
function() {
    return "Hello World!";
}
```

With Arrow Function:

```
const hello = () => {
  return "Hello World!";
}
```

It gets shorter! If the function has only one statement, and the statement returns a value, you can remove the brackets and the return keyword:

```
const hello = () => return "Hello World!";
```

Note

This works only if the function has only one statement.

🔗 If you have parameters, you pass them inside

the parentheses:

Example

Arrow Function With Parameters:

```
const hello = (val) => "Hello " + val;
```

Arrow Function Without Parentheses:

```
const hello = val => "Hello " + val;
```

✍️👋 React ES6 Variables

🔗 Variables

Before ES6 there was only one way of defining your variables: with the var keyword. If you did not define them, they would be assigned to the global object. Unless you were in strict mode, then you would get an error if your variables were undefined.

Now, with ES6, there are three ways of defining your variables: `var`, `let`, and `const`.

🔗 var

```
var x = 5.6;
```

- 1. If you use `var` outside of a function, it belongs to the global scope.
- 2. If you use `var` inside of a function, it belongs to that function.
- 3. If you use `var` inside of a block, i.e. a for loop, the variable is still available outside of that block.

Note

var has a function scope, not a block scope.

🔗 let

```
let x = 5.6;
```

- 1. `let` is the block scoped version of var, and is limited to the block (or expression) where it is defined.
- 2. If you use `let` inside of a block, i.e. a for loop, the variable is only available inside of that loop.

Note

let has a block scope.

🔗 const

```
const x = 5.6;
```

- 1. `const` is a **variable** that **once it has been created**, its **value** can **never change**.

Note

const has a block scope.

✍️👋 React ES6 Array Methods

🔗 Array Methods

There are many JavaScript array methods.

One of the most useful in React is the .map() array method.

The `.map()` method allows you to run a function on each item in the array, returning a new array as the result.

In React, `.map()` can be used to generate lists.

```
const myArray = ['apple', 'banana', 'orange'];
const myList = myArray.map((item) ⇒ console.log
```

```
const myArray = ['apple', 'banana', 'orange'];

const myList = myArray.map((item)⇒{
    return(
        console.log(item)
    )
})
```

Output

```
apple
banana
orange
```

✍️👋 React ES6 Destructuring

🔗 Destructuring

Destructuring makes it easy to extract only what is needed.

Example

Before:

```
const person = {
 firstName: "Kamal Nayan",
 lastName: "Upadhyay",
 age: 21,
 sex: "M"
}
```

```
const first = person.firstName;
const age = person.age;
const city = person.city || "Patna";

console.log(first) // "Kamal Nayan"
console.log(age) // 21
console.log(city) // "Patna" because person.city
```

Here is the new way of assigning array items to a variable:

Example

With destructuring:

```
const person = {
 firstName: "Kamal Nayan",
 lastName: "Upadhyay",
 age: 21,
 sex: "M"
}

const { firstName: first, age, city = "Patna" }

console.log(age) // 21 -- A new variable age is
console.log(first) // "Kamal" -- A new variable
console.log(firstName) // Undefined -- person.fi
console.log(city) // "Patna" -- A new variable c
```

🔗 Destructure with array:

```
function calculate(a, b) {
  const add = a + b;
  const subtract = a - b;
  const multiply = a * b;
  const divide = a / b;
```

```
  return [add, subtract, multiply, divide];
}

const [add, subtract, multiply, divide] = calcul

console.log("Sum: " + add);
console.log("Difference " + subtract);
console.log("Product: " + multiply);
console.log("Quotient " + divide);
```

Output

```
Sum: 11
Difference -3
Product: 28
Quotient 0.5714285714285714
```

✍️👋 React ES6 Spread Operator

🔗 Spread Operator

The JavaScript spread operator `(...)` allows us to quickly copy all or part of an existing array or object into another array or object.

Example:

```
const numbersOne = [1, 2, 3];
const numbersTwo = [4, 5, 6];
const numbersCombined = [ ... numbersOne,  ... numbe
console.log(numbersCombined);
```

Output

```
[1,2,3,4,5,6]
```

🔗 We can use the spread operator with objects too:

Example:

Combine these two objects:

```
const myVehicle = {
  brand: 'Ford',
  model: 'Mustang',
  color: 'red'
}

const updateMyVehicle = {
  type: 'car',
  year: 2021,
  color: 'yellow'
}

const myUpdatedVehicle = { ... myVehicle,  ... updat
```

Output:

```
{brand: 'Ford', model: 'Mustang', color: 'yellow
```

Note

Notice the properties that did not match were combined, but the property that did match, color, was overwritten by the last object that was passed, updateMyVehicle. The resulting color is now yellow.

✍️👋 React ES6 Modules

🔗 Modules

JavaScript modules allow you to break up your code into separate files.

This makes it easier to maintain the code-base.

ES Modules rely on the `import` and `export` statements.

🔗 Export

You can export a function or variable from any file.

Let us create a file named `person.js`, and fill it with the things we want to export.

There are two types of exports: Named and Default.

🔗 Named Exports

You can create **named exports** two ways. In-line individually, or all at once at the bottom.

Example

In-line individually:

person.js

```
export const name = "Jesse"
export const age = 40
```

All at once at the bottom:

person.js

```
const name = "Jesse"
const age = 40

export { name, age }
```

🔗 Default Exports

Let us create another file, named `message.js`, and use it for demonstrating default export.

You can only have one default export in a file.

Example

message.js

```
const message = () => {
  const name = "Jesse";
  const age = 40;
  return name + ' is ' + age + 'years old.';
};

export default message;
```

🔗 Import

You can **import modules** into a **file** in two ways, based on if they are **named exports** or **default exports**.

Named exports must be **destructured** using **curly braces**. **Default exports** do not.

Example

Import **named exports** from the file **person.js**:

```
import { name, age } from "./person.js";
```

Example

Import **default export** from the file **message.js**:

```
import message from "./message.js";
```

✍️👋 React ES6 Ternary Operator

🔗 Ternary Operator

The ternary operator is a simplified conditional operator like `if /`

`else`.

Syntax: `condition ? <expression if true> : <expression if false>`

Here is an example using `if / else`:

Example:

Before:

```
if (authenticated) {
  console.log('home');
} else {
  console.log('Login Page');
}
```

Here is the **same example** using a **ternary operator**:

```
authenticated ?  console.log('home') : console.l
```

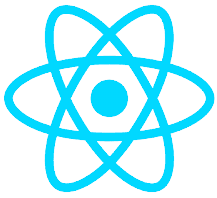
LIVE EDITOR

```
function Clock(props) {
  const [date, setDate] = useState(new Date());
  useEffect(() => {
    var timerID = setInterval(() => tick(), 1000);

    return function cleanup() {
      clearInterval(timerID);
    };
  });

  function tick() {
    setDate(new Date());
  }

  return (
    <div>
      <h2>It is {date.toLocaleTimeString()}.</h2>
    </div>
```

```
    );
}
```

RESULT

It is 18:03:48.



React JSX

The library for web and native user interfaces

Watch On YouTube

✍️👋 React JSX

🔗 What is JSX?

```
1. JSX stands for JavaScript XML.

2. JSX allows us to write HTML in React.

3. JSX makes it easier to write and add HTML in
```

🔗 Rules of write Jsx

```
1. Html code must wrap into one top level elemen

2. Element must be closed.

3. Attribute class = className.
```

```
4. No if else Condition inside jsx ternary opera

5. Js expression  in Jsx must be wrapped in {}.
```

🔗 Rule no :- 1

Html code must wrap into one top level element (Parents element)

LIVE EDITOR

```
function App(){
     return(
          <>
          <h1>Hello world</h1>
          <p>Rule no :- 1</p>
          </>
     )
 }
```

RESULT

Hello world
Rule no :- 1

🔗 Rule no :- 2

Element must be closed.

LIVE EDITOR

```
function App(){
     return(
          <>
          <h1>Hello world</h1>
          <p>Rule no :- 2</p>
          </>
     )
 }
```

RESULT

Hello world
Rule no :- 2

🔗 Rule no :- 3

`class = className`

class not use in Jsx because class already present in jsx in the form of reserve keyword.

LIVE EDITOR

```
function App(){
     return(
          <>
          <h1>Hello world</h1>
          <p>Rule no :- 3</p>
          </>
     )
 }
```

RESULT

Hello world
Rule no :- 3

🔗 Rule no :- 4

No if else condition inside Jsx but ternary operator is okay

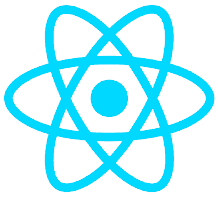
LIVE EDITOR

```
function App(){
    const x = 10;
    return(
         <>
         <h1>Hello world!</h1>
         <p>Rule no:-4</p>
         <p>{(x)>15? "Greater": "smaller"} </p>
         </>
    )
 }
```

RESULT

Hello world!
Rule no:-4
smaller

Devknus



React Components

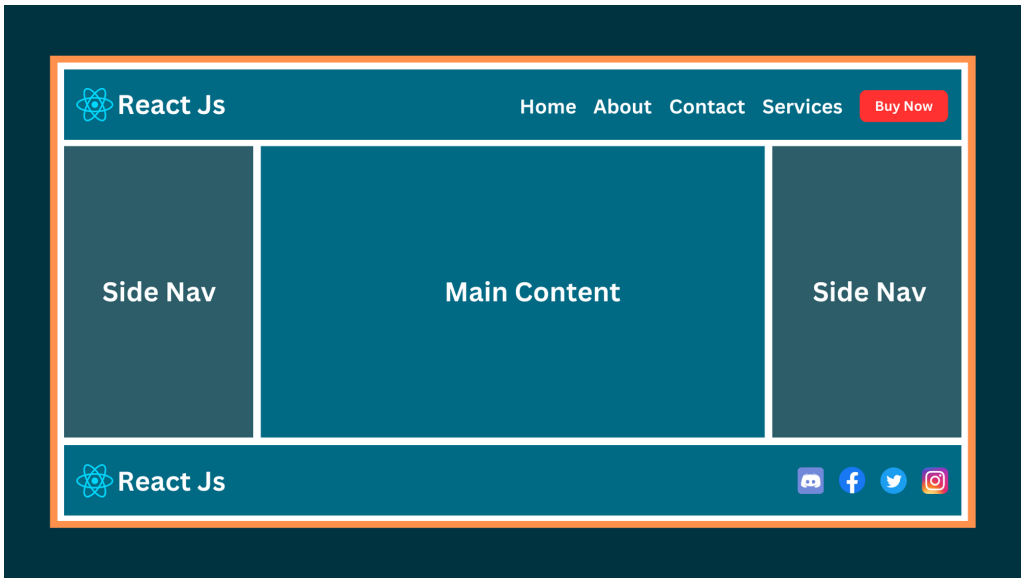
The library for web and native user interfaces

Watch On YouTube

✍️👋 React Components

🔗 What is component

```
1. React component is a type of function that re

2. React component is called the building block
creates userfrinedly and attractive div, section
any content in a proper format.

3. React components are used to create different
resuable sections and UI (User Interface).
```



🔗 Types Of Components

```
1. Class Component
2. Function Components
```

🔗 1. Class Component

Class component is a user defined class of javascript. Which is made with the render() method to return the HTML content.

```
//Create a Class component called ClassComponent
class ClassComponent extends React.Component {
render() {
    return <h2>Hi, I am a Class Component!</h2>;
 }
}

// Hi, I am a Class Component!
```

🔗 2. Function Component

A functional component is a user friend function of JavaScript that returns HTML content.

```
A Function component also returns HTML, and beha
but Function components can be written using muc
```

🔗 How To Create Components

```
React is all about re-using code, and it is reco

To do that, create a new file with a `.jsx` file
```

> **Note**
>
> Filename must start with an uppercase character.

```jsx
import React from "react";

function Navbar(){
     return(
          <>
          <h1>Hello world</h1>
          <p>How are you </p>
          <Car/>
          </>
     )
 }
export default Navbar;
```

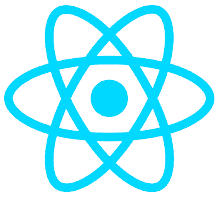
🔗 Rendering a Component

We made a component, now we want to render/use it. Syntax for using a component is:

```
<ComponentName />
```

🔗 Components in Components

```
import React from "react";

function Navbar(){
     return(
          <>
          <h1>Hello world</h1>
          <p>How are you </p>
          <Car/>
          </>
     )
 }
export default App;

// second components
function Car() {
  return <h2>I am a Car!</h2>;
}
```

React Props

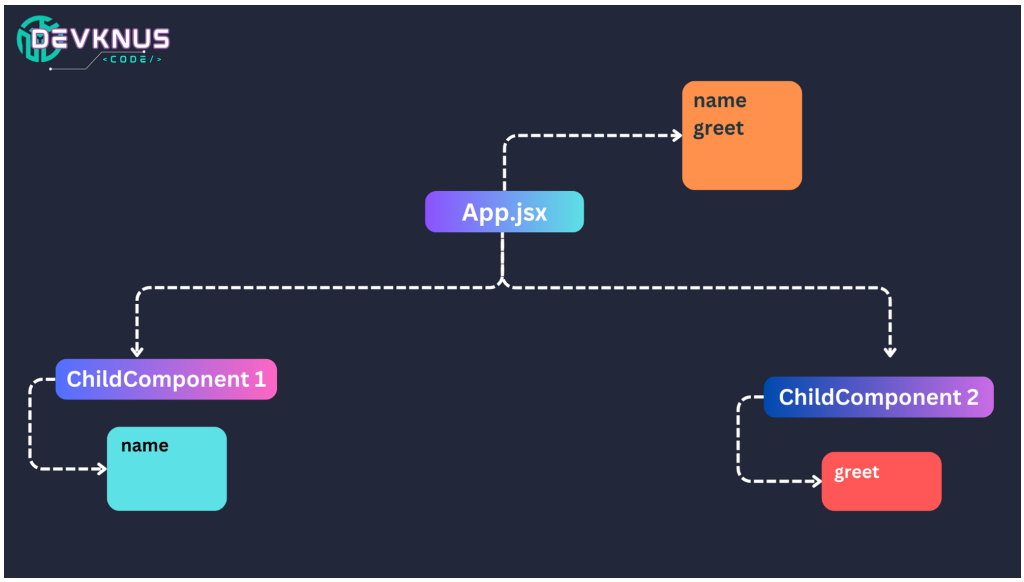
The library for web and native user interfaces

Watch On YouTube

✍️👋 React Props

🔗 Props in React

```
1. Props stand for properties.

2. Props are used to transfer data from one comp

3. Props are read only.

4. Props are just like a function in Jsx.
```



🔗 Example 1 :

🔗 Step 1: Create a components ChildComponent1.jsx

```
- components
  - ChildComponents1
```

🔗 Step 2: App.jsx

Send Props Parent to child

```
import React from 'react'
import ChildComponent1 from './components/ChildC

function App() {
  const name = "Kamal Nayan Upadhyay";

  return (
    <div className='main'>
      <div classNames="">
      <ChildComponent1 name={name}/>
      </div>
    </div>
  )
```

```
}

export default App
```

🔗 Step 3: ChildComponents1.jsx

Receive Props

There are Two way to receive props:

🔗 Example 1:

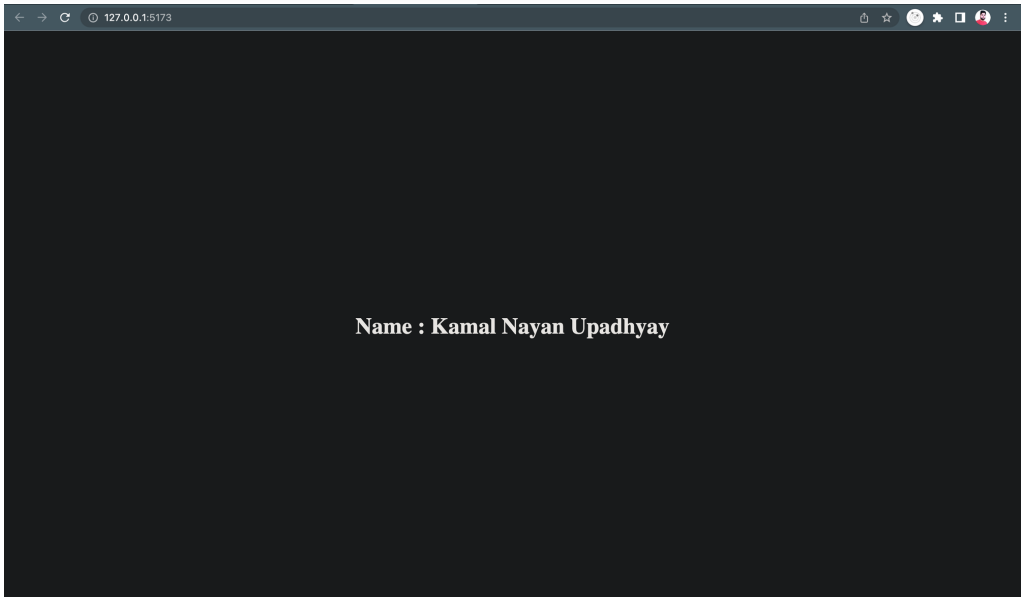
Way 1:

```
import React from 'react'

function ChildComponent1(props) {
    // console.log(props)
    return (
        <div>
            <div className="text">
            <h1>Name : {props.name}</h1>
            </div>
        </div>
    )
}

export default ChildComponent1
```

Way 2:

```
import React from 'react'

function ChildComponent1({name}) {
    // console.log(props)
    return (
        <div>
            <div className="text">
```

```
            <h1>Name : {name}</h1>
            </div>
        </div>
    )
}

export default ChildComponent1
```



🔗 Example 2:

🔗 Step 1: Create a components ChildComponent2.jsx

```
- components
  - ChildComponents1.jsx
  - ChildComponents2.jsx
```

🔗 Step 2: App.jsx

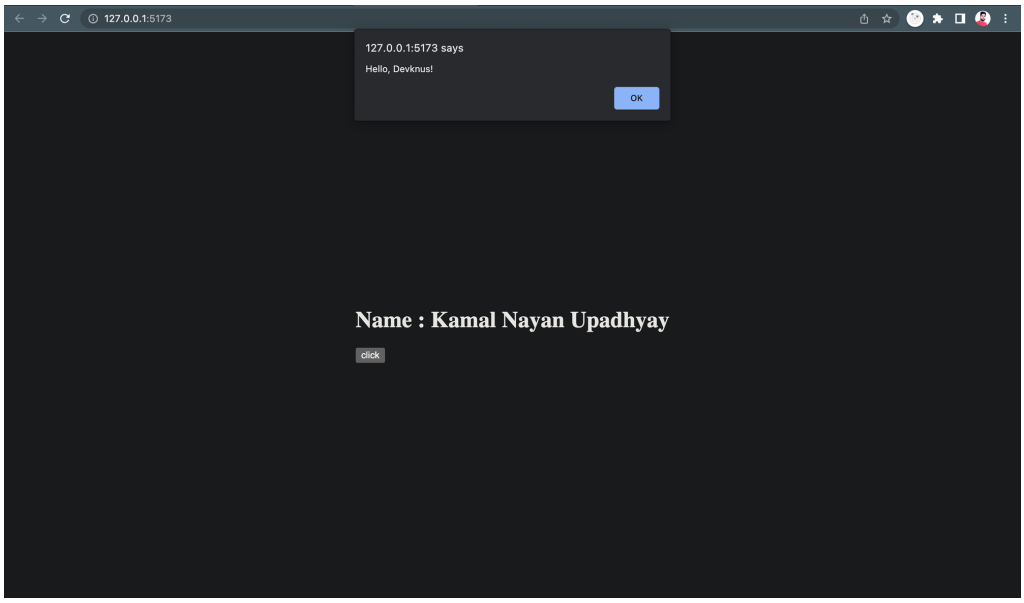
Send Props Parent to child

```
import React from 'react'
import ChildComponent1 from './components/ChildC
import ChildComponents2 from './components/Child
```

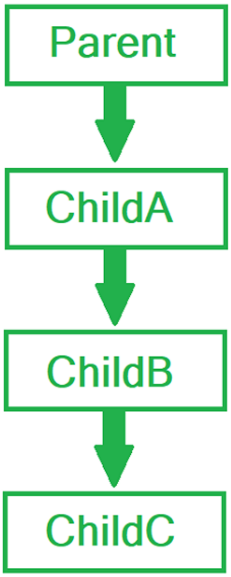
```
function App() {
  const name = "Kamal Nayan Upadhyay";
  const greet = ( ) =>{
    alert("Hello, Devknus!");
  }
  return (
    <div className='main'>
      <div className="">
      <ChildComponent1 name={name}/>
      <ChildComponents2 greet={greet}/>
      </div>
    </div>
  )
}

export default App
```

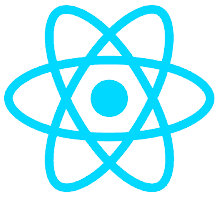
🔗 Step 3: ChildComponents2.jsx

```
import React from 'react'

function ChildComponents2({greet}) {
  return (
    <div>
        <button onClick={greet}>click</button>
    </div>
  )
}

export default ChildComponents2
```



🔗 Problems with Props





React Events

The library for web and native user interfaces

Watch On YouTube

✍️👋 React Events

You have to learn React Event handling and event handler if you need to perform events like click, change hover, key press, key down, key up & more in your web application.

React has the same events as HTML: click, change, mouseover etc.

🔗 Adding Events

React events are written in `camelCase` syntax:

`onClick` instead of `onclick`.

HTML:

```
onclick
```

React:

```
onClick
```

🔗 React event handlers are written inside curly braces:

```
onClick={shoot} instead of onClick="shoot()".
```

HTML:

```
onclick="shoot()"
```

React:

```
onClick={shoot}
```

HTML:

```
<button onclick="shoot()">Take the Shot!</button
```

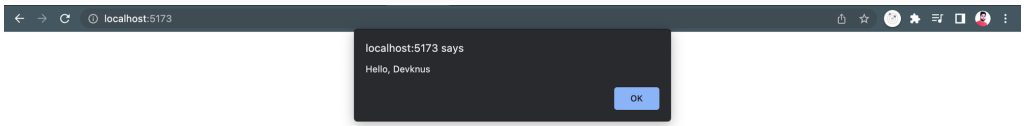
React:

```
<button onClick={shoot}>Take the Shot!</button>
```

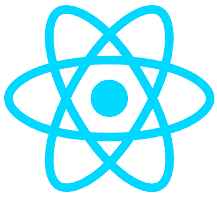
🔗 How To Use onClick event

```
import React from 'react'

function App() {
```

```
  const greet = () =>{
    alert('Hello, Devknus');
  }
  return (
    <div>
      <button onClick={greet}>click</button>
    </div>
  )
}

export default App
```



click



React Conditional Rendering

The library for web and native user interfaces

Watch On YouTube

✍️👋 React Conditional Rendering

In React, you can conditionally render components.

There are several ways to do this.

```
1. if Statement
2. Logical && Operator
3. Ternary Operator
```

🔗 if Statement

We can use the `if` JavaScript operator to decide which component to render.

Example:

We'll use these two Pages:

```
function Login() {
  return <h1>Login Page</h1>;
}

function Home() {
  return <h1>Home Page</h1>;
}
```

Example:

Now, we'll create another component that chooses which component to render based on a condition:

Try changing the `isAuth` attribute to `false` :

```
import React from 'react'

function App(props) {
  const isAuth = false;
  if (isAuth) {
    return <Login />;
  }
  return <Home />;
}

export default App
```

🔗 Logical && Operator

Another way to conditionally render a React component is by using the `&&` operator.

```
import React from 'react'

function App() {
 const fruits = ['Apple', 'Banana', 'Orange'];
```

```
  return (
    <>
     <div>Fruits</div>
     {fruits.length > 0 &&
       <h2>You have {fruits.length} fruits in you
    </>
  )
}

export default App
```

🔗 Ternary Operator

Another way to conditionally render elements is by using a ternary operator.

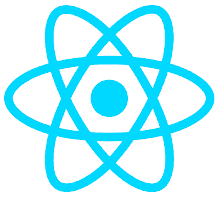
```
condition ? true : false
```

Example:

if `isAuth` is `true`, otherwise return the `Home` Page:

```
import React from 'react'
import Home from './pages/Home'
import Login from './pages/Login'

function App() {
  const isAuth = true;

  return (
    <>
    <h1>Welcome To devknus</h1>
    {isAuth ? <Home/> : <Login/>}
    </>
  )
}
```

```
export default App
```



React Lists Rendering

The library for web and native user interfaces

Watch On YouTube

✍️👋 React Lists Rendering

```
1. Lists in React are same as the lists in JavaS

2. We can use lists to show multiple items in a

3. We can use lists for displaying menu, navigat

4. To display a list, we can use map() method of
```

🔗 Basic HTML/JS List

```
<ul>
     <li>Item 1</li>
     <li>Item 2</li>
```

```
    <li>Item 3</li>
</ul>
```

🔗 React List

```
import React from 'react'

function App() {
  const items = ["Item 1", "Item 2", "Item 3"];
  return (
    <div>
      <ul>
        {items.map((value)⇒{
          console.log(value)
          return  <li>{value}</li>
        })}
      </ul>
    </div>
  )
}
export default App
```

🔗 map()

An example of map:

```
function App() {
  const languages = ['JavaScript', 'Python', 'Ja

  return (
    <div className="p-2">
       <div className="App">
      {languages.map((language) ⇒ {
        return <div>I learn {language}</div>
      })}
```

```
    </div>
    </div>
  )
}

// Output

// I learn JavaScript
// I learn Python
// I learn Java
// I learn C
// I learn C++
// I learn C#
```

🔗 Lists

When we run this React List code, we will be given a warning that

```
"Each child in a list should have a unique 'key'
```

🔗 Keys

```
1. A "key" is a special string attribute you nee
creating lists of elements.

2. Keys help React identify which items have cha

3. If you choose not to assign an explicit key t
default to using indexes as keys. You will get a
```

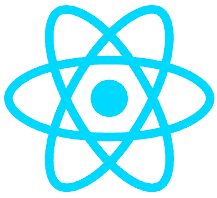
```
function App() {
  const languages = ['JavaScript', 'Python', 'Ja

  return (
```

```
    <div className="p-2">
       <div className="App">
      {languages.map((language, index) => {
        return <div key={index}>I learn {languag
      })}
    </div>
    </div>
  )
}

export default App
```

Example:

```
import React from 'react'

const skills = [
  { id: 1, skill: 'JavaScript' },
  { id: 2, skill: 'Python' },
  { id: 3, skill: 'Java' },
  { id: 4, skill: 'C' },
  { id: 5, skill: 'C++' },
  { id: 6, skill: 'C#' },
  { id: 7, skill: 'Html' },
  { id: 8, skill: 'Css' },
  { id: 9, skill: 'Bootstrap' },
  { id: 10, skill: 'Tailwind Css' },
  { id: 11, skill: 'Material ui' },
  { id: 12, skill: 'React Js' },
  { id: 13, skill: 'Next Js' },
  { id: 14, skill: 'Angular Js' },
  { id: 15, skill: 'Vue Js' },
  { id: 16, skill: 'React Native' },
  { id: 17, skill: 'Dart' },
  { id: 18, skill: 'Flutter' },
  { id: 19, skill: 'Github' },
  { id: 20, skill: 'Electron Js' },
```

```
];

function App() {
  return (
    <div>
      {skills.map((value,index)⇒{
        // console.log(value)
        const {skill,id}= value
        return(
          <h1 key={index}>{id}. {skill}</h1>
        )
      })}
    </div>
  )
}

export default App
```



React Router v6

The library for web and native user interfaces

Watch On YouTube

✍️👋 React Router

🔗 Step 1 : Add React Router

To install React router, run this in your terminal:

```
npm i react-router-dom
```

🔗 Step 2: Creating Multiple Pages

We'll create the following `Home`, `About`, `Contact`, and `Nopage` pages like this:

`src/pages/Home.jsx`

```
function Home() {
  return (
    <div>
      <h1>This is the home page</h1>
    </div>
```

```
    );
}

export default Home;
```

src/pages/About.jsx

```
import React from 'react'

function About() {
    return (
        <div>
            <h1>This is the about page</h1>
        </div>
    )
}

export default About
```

src/pages/Contact.jsx

```
import React from 'react'

function Contact() {
    return (
        <div>
            <h1>This is the contact page</h1>
        </div>
    )
}

export default Contact
```

src/pages/NoPage.jsx

```
import React from 'react'
```

```
function NoPage() {
    return (
        <div>
            <h1>This is the noPage page</h1>
        </div>
    )
}

export default NoPage
```

🔗 Step 3: Define routes in App.jsx

- 1. Import `BrowserRouter as Router`, `Route`, `Routes`, from **react-router-dom**

```
import {
  BrowserRouter as Router,
  Route,
  Routes,
} from "react-router-dom";
```

🔗 Complete Code

```
import {
  BrowserRouter as Router,
  Route,
  Routes,
} from "react-router-dom";

import Home from "./pages/Home"
import About from "./pages/About"
import Contact from "./pages/Contact"
import NoPage from "./pages/NoPage"


function App() {
```

```
  return (
    <div className="App">
      <Router>
       <Routes>
        <Route path="/" element={ <Home/> } />
        <Route path="/about" element={ <About/>
        <Route path="/contact" element={ <Contac
        <Route path="/*" element={ <NoPage/> } /
       </Routes>
      </Router>
    </div>
  )
}

export default App
```

🔗 Step 4: Create a Navbar

```
- components
  - Navbar.jsx
```

```
import React from 'react'
function Navbar() {
  return (
    <div className='main'>
      <div className="left">
        <h1 className='logo_text'>Navbar</h1>
      </div>
      <div className="right">
        <ul>
          <li>Home</li>
          <li>About</li>
          <li>Contact</li>
        </ul>
      </div>
```

```
    </div>
  )
}

export default Navbar
```

🔗 Css Code For Navbar

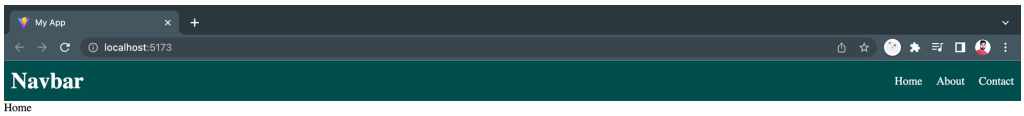
```
* {
    margin: 0;
    padding: 0;
}

.main {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px;
    background-color: rgb(0, 78, 78);
}

.logo_text{
    color: white;
}

ul {
    display: flex;
    gap: 20px;
    color: white;
}

li {
    list-style: none;
}
a{
    text-decoration: none;
```

```
    color: white;
}
a:hover{
    color: wheat;
}
```

🔗 Step 5: Import Navbar.jsx in App.jsx

```
import {
  BrowserRouter as Router,
  Route,
  Routes,
} from "react-router-dom";

import Home from "./pages/Home";
import About from "./pages/About";
import Contact from "./pages/Contact";
import NoPage from "./pages/NoPage";
import Navbar from "./components/Navbar";

function App() {
  return (
    <div className="App">
      <Router>
       <Navbar/>
```

```
        <Routes>
         <Route path="/" element={ <Home/> } />
         <Route path="/about" element={ <About/>
         <Route path="/contact" element={ <Contac
         <Route path="/*" element={ <NoPage/> } /
        </Routes>
       </Router>
     </div>
   )
}

export default App
```



Home

🔗 Step 6: Use Link to navigate to routes

go to Navbar.jsx

The `Link` component is similar to the anchor element `(<a>)` in HTML. Its to attribute specifies which path the link takes you to. Always remember to `import` `Link` from `react-router-dom` before using it.

```
import React from 'react'
import { Link } from "react-router-dom";

function Navbar() {
```

```
  return (
    <div className='main'>
      <div className="left">
        <h1 className='logo_text'>Navbar</h1>
      </div>
      <div className="right">
        <ul>
          <Link to={'/home'}><li>Home</li></Link
          <Link to={'/about'}><li>About</li></Li
          <Link to={'/contact'}><li>Contact</li>
        </ul>
      </div>

    </div>
  )
}

export default Navbar
```

🔗 Step 7: Move One Page To Another (useNavigate hook)

In this example, we will use `useNavigate()` hook to `navigate` to the `about` page and to go back to the `home` page. From the following code, the user can go back to the `Home page` from the `About page` on the `button` click.

`src/pages/Home.jsx`

```
import React from 'react';
import {useNavigate} from "react-router-dom"

const Home = () => {
const navigate = useNavigate();

return (
    <>
        <h1>Home Page</h1>
```

```
            <button onClick={()⇒navigate("/about")}
        </>
    )
};

export default Home;
```

`src/pages/About.jsx`

```
import React from 'react';
import {useNavigate} from "react-router-dom"

const About = () ⇒ {
  const navigate = useNavigate();

  return (
  <>
     <h1>About Page</h1>
     <button onClick={()⇒navigate(-1)}>Go Back
  </>
  )
};

export default About;
```

🔗 Step 8: ReactJS useParams Hook

The `useParams()` hook is a React Router hook that `allows you to access the parameters of the current URL.` This can be useful if you want to dynamically render content based on the URL parameters. For example, if you have a blog application, you may want to render different articles based on the article ID in the URL.

1.Create a Blog Page

```
import React from 'react'

function Blog() {
  return (
    <div>App</div>
  )
}

export default Blog
```

2. Define Route

```
import {
  BrowserRouter as Router,
  Route,
```

```
  Routes,
} from "react-router-dom";

import Home from "./pages/Home"
import About from "./pages/About"
import Contact from "./pages/Contact"
import Blog from "./pages/Blog"
import NoPage from "./pages/NoPage"

function App() {
  return (
    <div className="App">
      <Router>
       <Routes>
        <Route path="/" element={ <Home/> } />
        <Route path="/about" element={ <About/>
        <Route path="/contact" element={ <Contac
        <Route path="/blog/:id" element={ <Blog/
        <Route path="/*" element={ <NoPage/> } /
       </Routes>
      </Router>
    </div>
  )
}

export default App
```

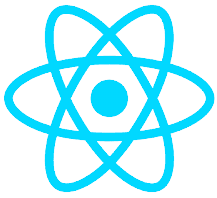
3. Navbar.jsx

```
import React from 'react'
import { Link } from "react-router-dom";

function Navbar() {
  return (
    <div className='main'>
      <div className="left">
        <h1 className='logo_text'>Navbar</h1>
```

```
        </div>
        <div className="right">
          <ul>
            <Link to={'/home'}><li>Home</li></Link
            <Link to={'/about'}><li>About</li></Li
            <Link to={'/contact'}><li>Contact</li>
            <Link to={'/blog'}><li>Blog</li></Link
          </ul>
        </div>

    </div>
  )
}

export default Navbar
```

Receive Parameter

```
import React from 'react'
import { useParams } from 'react-router-dom';

function Blog() {
    const { id } = useParams();
  return (
    <div>Blog Id : {id}</div>
  )
}

export default Blog
```



React CSS Styling

The library for web and native user interfaces

Watch On YouTube

✍️👋 React CSS Styling

```
1. Inline Styling
2. CSS Stylesheets
3. CSS Modules
```

🔗 Inline Styling

To `inline` style an element, we can make a Javascript Object, like this:

```
function App() {
  return (
    <div>
      <h2 style={{ backgroundColor: "#7d5fff", c
    </div>
```

```
    )
}
```

In this first curly brace is to write javascript and second is to make a javascript object. We can also write it like:

```
function App() {
  const h2Style = {
    backgroundColor: '#7158e2',
    color: 'white'
  }
  return (
    <div>
      <h2 style={h2Style}>WebKnuDocs</h2>
    </div>
  )
}
```

Note

CSS property name must be camelCase. background-color would be backgroundColor

🔗 CSS Stylesheets

You can save the whole CSS in a separate file with file extension .css and import it in your application.

`src/App.css`

```
.heading{
    background-color: red;
    font-size: 90px;
}
```

Here we are writing CSS, so we don't need to make JS Object or do it in camelCase.

App.js

Just import it like this:

```
import React from 'react'
import './App.css';

function App() {
  return (
    <div>App</div>
  )
}

export default App
```

🔗 CSS Modules

In this you don't have to worry about `name` conflicts as it is component specific. The CSS is available only for the file in which it is imported.

Make a file with extension `.module.css`, example: `index.module.css`

Make a new file `index.module.css` and insert some CSS into it, like this:

`src/index.module.css`

```
.button {
  background-color: 'purple';
  color: 'white';
}
```

Import it in component like this:

```
import styles from './index.module.css';

const App = () => {
    return (
        <button className={styles.button}>Click
    )
};

export default App;
```

✍️👋 Tailwind Css in React Js

🔗 Install tailwind Css

- 1. Install tailwindcss and its peer dependencies, then generate your tailwind.config.js and postcss.config.js files.

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

- 2. Configure your template paths

Add the paths to all of your template files in your tailwind.config.js file.

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

- 3. Add the Tailwind directives to your CSS

Add the @tailwind directives for each of Tailwind's layers to your ./src/index.css file.

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

- 4. Run your build process with npm run dev.

```
npm run dev
```

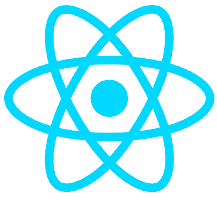
🔗 Bootstrap in React Js

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/boo
  </head>
```

```
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bo
  </body>
</html>
```

Devknus

Devknus



React Hook Introduction

The library for web and native user interfaces

Watch On YouTube

What is Hook

```
1. Hook are basically functions.
2. Hooks were added to React in version 16.8.
3. Hooks cannot work in React Class Components.
4. Hooks allow us to "hook" into React feature s
5. Hooks are the functions which "hook into" Rea
```

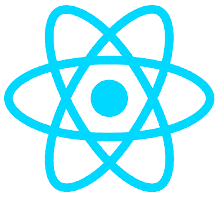
Hooks Rules

```
1. Hooks cannot work in React Class Components.
2. You must import hook first
3. Hooks can only be called inside React functio
4. We should not call Hooks inside loops, condit
```

```
// import statements
// Can't call here
const Blogs = () => {
    // Can call here
    return <h1>Blogs</h1>;
};

export default Blogs;
```

🔗 Types of Hook

1. useState
2. useEffect
3. useContext
4. useRef



useState Hook

The library for web and native user interfaces

Watch On YouTube

✍️👋 UseState Hook

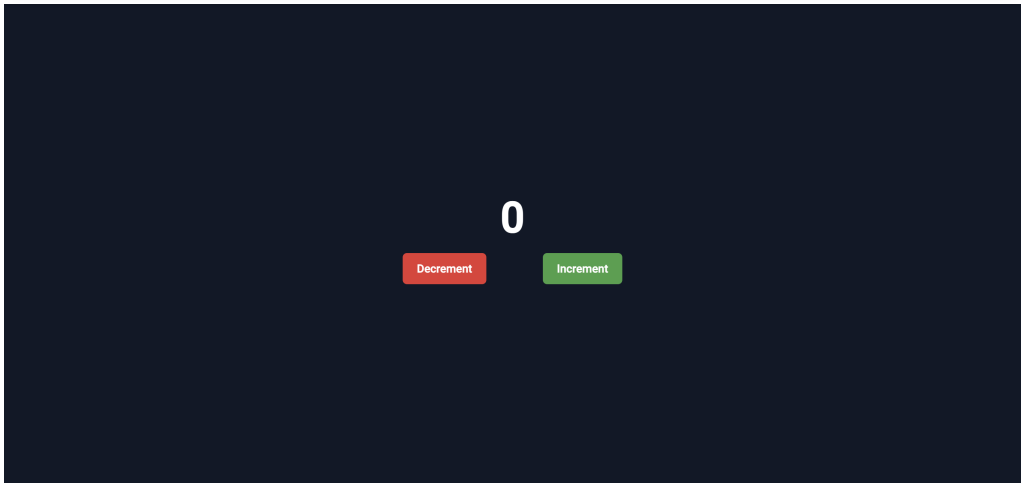
`useState` is React `Hook` that `allows` you to add `state` to a functional component

🔗 What is State

```
1. The state is a object that is used to contain
2. A component's state can change over time; whe
```

```
import React from 'react'

function App() {
    let count = 0;

    const increment = () => {
        count++;
```

```
            console.log(count)
        }
        const decrement = () => {
            count--;
            console.log(count)
        }

        return (
            <div className=' flex space-x-3 justify-
                <div className="">
                    <p className=' text-6xl font-bol
                    <button className=' bg-red-600 t
                    <button className=' ml-20 bg-gre
                </div>
            </div>
        )
    }

    export default App
```



🔗 Syntax

```
const [state, setState] = useState(initialState)
```

Image 1

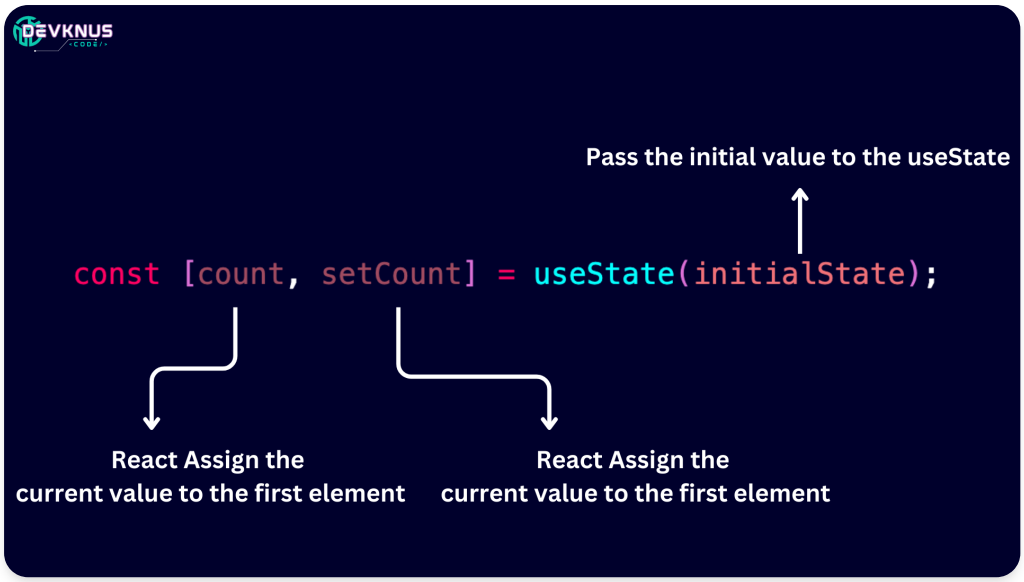
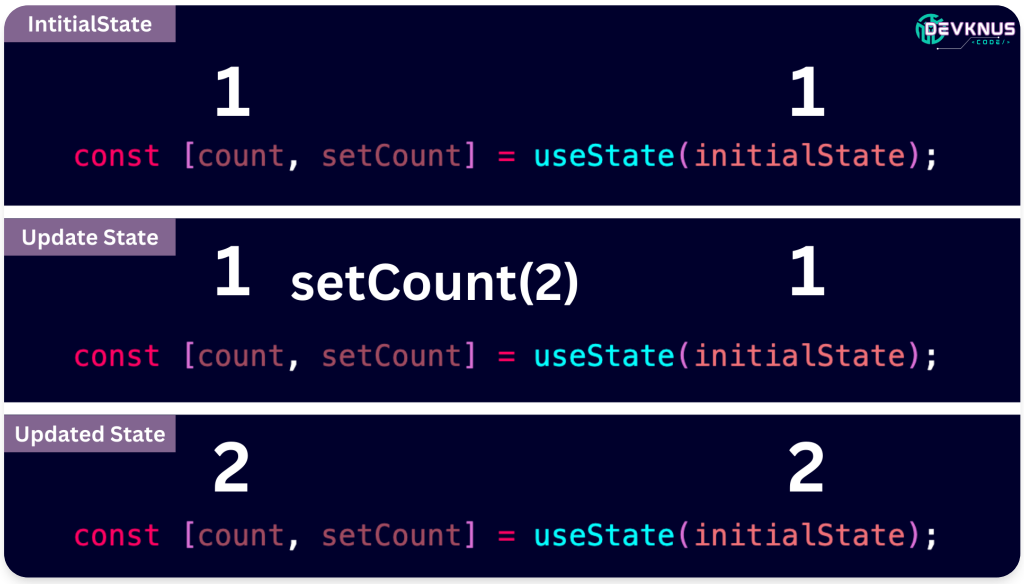


Image 2



🔗 Importing useState

To use `useState`, first we need to `import` `useState` and `initialize` it, you can import it from react like this:

```
import { useState } from "react";
```

🔗 Initializing useState

You can initialize `state` like this:

```
import React from 'react'

function App() {
    const [count, setCount] = useState(1);
  return (
    <div>
        <h1>useState</h1>
    </div>
  )
}

export default App
```

🔗 Reading a state

As mentioned earlier, it returns a state and a function to change/update that state. Hence, everything is `stored` in `count`. We can read `states` just like variables:

LIVE EDITOR

```
function App() {
    const [count, setCount] = useState(1);
  return (
    <div>
        <h1 className="text-4xl text-center">
{count}</h1>
    </div>
  )
}
```

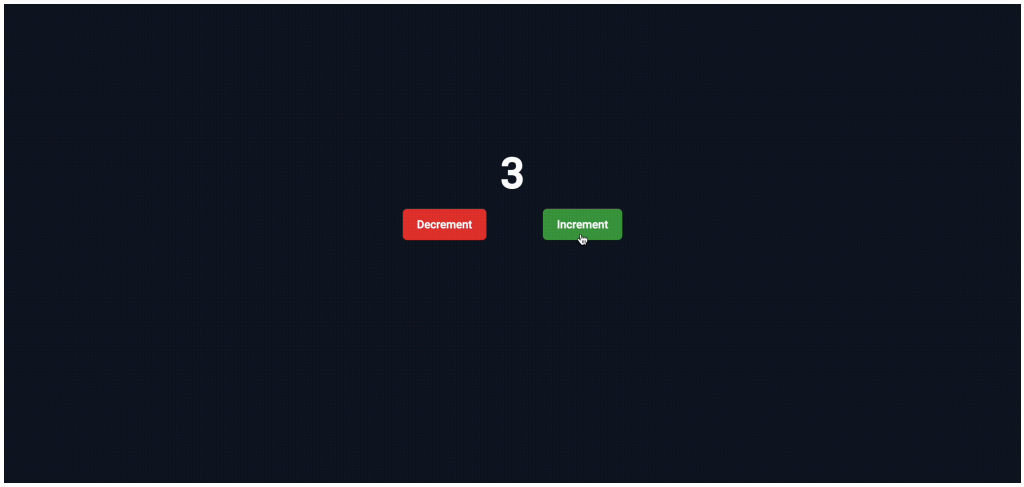
RESULT

1

🔗 Updating a state

To update `state` we use the `function` it `returns` to update state, in this case: setCount. State can be `updated` like this:

```
import React, { useState } from 'react'

function App() {
    const [count, setCount] = useState(1);
    const increment = () => {
        setCount(count + 1);
    }
    const decrement = () => {
        setCount(count - 1)
    }
    return (
        <div className=' flex space-x-3 justify-
            <div className="">
                <p className=' text-6xl font-bol
                <button className=' bg-red-600 t
                <button className=' ml-20 bg-gre
            </div>
        </div>
    )
}

export default App
```



🔗 What can state hold?

state can hold any datatype like:

```
strings,
numbers,
booleans,
arrays,
objects,
objects in arrays,
arrays in objects
```

```
import React, { useState } from 'react'

function App() {
    const [data, setData] = useState({
        name: 'Kamal Nayan',
        age: 21
    });
    return (
        <div className="">
            <h1 className='text-3xl font-bold'>
                My name is <span className=' tex
                and my age is <span className='
            </h1>
        </div>
    )
}

export default App
```

LIVE EDITOR

```
function App() {
    const [data, setData] = useState({
        name: 'Kamal Nayan',
        age: 21
    });
```

```
    return <div className="p-5">
            <h1 className='text-3xl font-bold'>
                My name is <span className=' text-
pink-500'>{data.name}</span>
                and my age is <span className='
text-blue-500'>{data.age}</span>
            </h1>
        </div>
}
```

RESULT

My name is Kamal Nayanand my age is 21

🔗 Updating Objects and Arrays in State

```
import React, { useState } from 'react'

function App() {
    const [data, setData] = useState({
        name: 'Kamal Nayan ',
        age: 21
    });

    const updateData = () => {
        console.log({...data})
        setData({...data, name: 'Devknus '})
    }
    return (
        <div className="">
            <h1 className='text-3xl font-bold mb
                My name is <span className=' tex
                and my age is <span className='
            </h1>
            <button onClick={updateData} classNa
```

```
            </div>
        )
    }

    export default App
```

LIVE EDITOR

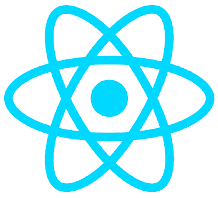
```
function App() {
    const [data, setData] = useState({
        name: 'Kamal Nayan ',
        age: 21
    });

    const updateData = () => {
        console.log({ ...data}) // {name: 'Kamal Nayan', age: 21}
        setData({ ...data, name: 'Devknus '})
    }
    return <div className="p-5">
            <h1 className='text-3xl font-bold mb-3'>
                My name is <span className=' text-pink-500'>{data.name}</span>
                and my age is <span className='text-blue-500'>{data.age}</span>
            </h1>
            <button onClick={updateData} className='bg-green-500 text-white py-2.5 px-10 rounded-md
            font-bold'>Update</button>
        </div>
}
```

RESULT

My name is Kamal Nayan and my age is 21

Update

Devknus



useEffect Hook

The library for web and native user interfaces

Watch On YouTube

✍️👋 useEffect Hooks

The `useEffect` Hook allows you to perform side effects in your components.

Some examples of side effects are: fetching data, directly updating the DOM, and timers.

`useEffect` accepts two arguments. The second argument is optional.

```
useEffect(<function>, <dependency>)
```

```
useEffect(() => {
    // Mounting

    return () => {
        // Cleanup function
    }
}, [//Updating])
```

🔗 Example

Let's use a timer as an example.

```
import React, { useEffect, useState } from 'reac

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

  useEffect(() => {
    setTimeout(() => {
      setCount((count) => count + 1);
    }, 1000);
  });
  return (
    <div>
      <h2>I have rendered {count} times!</h2>
    </div>
  )
}

export default App
```

LIVE EDITOR

```
function App() {
  const [count, setCount] = useState(0);
```

```
  useEffect(() => {
    setTimeout(() => {
      setCount((count) => count + 1);
    }, 1000);
  });

  return <h2 className='text-4xl font-bold text-
center'>I have rendered <span className='text-
pink-500'>{count}</span> times!</h2>
}
```

RESULT

I have rendered 2 times!

But wait!! It keeps counting even though it should only count once!

`useEffect` runs on every render. That means that when the count changes, a render happens, which then triggers another effect.

There are several ways to control when side effects run.

1. No dependency passed:

```
useEffect(() => {
  //Runs on every render
});
```

2. An empty array:

```
useEffect(() => {
  //Runs only on the first render
}, []);
```

3. Props or state values:

```
useEffect(() => {
  //Runs on the first render
```

```
    //And any time any dependency value changes
  }, [prop, state]);
```

So, to fix this issue, let's only run this effect on the initial render.

Example:

Only run the effect on the initial render:

```
import React, { useEffect, useState } from 'reac

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

  useEffect(() => {
    setTimeout(() => {
      setCount((count) => count + 1);
    }, 1000);
  },[]); // ← add empty brackets here
  return (
    <div>
      <h2>I have rendered {count} times!</h2>
    </div>
  )
}

export default App
```

LIVE EDITOR

```
function App() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    setTimeout(() => {
      setCount((count) => count + 1);
    }, 1000);
  }, []); // ← add empty brackets here
```

```
  return <h2 className='text-4xl font-bold text-
center'>
  I have rendered <span className='text-pink-500'>
  {count}</span> times!</h2>
}
```

RESULT

I have rendered 1 times!

Example

Here is an example of a useEffect Hook that is dependent on a variable. If the count variable updates, the effect will run again:

```
import React, { useEffect, useState } from 'reac

function App() {
  const [count, setCount] = useState(0);
  const [calculation, setCalculation] = useState

  const update = () => {
    setCount(count + 1);
  }

  useEffect(() => {
    setCalculation(()=> count * 2)
  }, [count]);

  return (
    <div>
      <div className='flex justify-center items-
        <div className=' border-2 border-gray-20
          <p className='text-center font-bold mb
          <button onClick={update} className=' f
          <p className='text-lg font-bold'>Calcu
        </div>
      </div>
    </div>
```

```
    )
  }

  export default App
```

If there are multiple dependencies, they should be included in the useEffect dependency array.

LIVE EDITOR

```
function App() {
  const [count, setCount] = useState(0);
  const [calculation, setCalculation] =
useState(0);

  const update = () => {
    setCount(count + 1);
  }

  useEffect(() => {
    setCalculation(() => count * 2);
  }, [count]); // ← add the count variable here

  return <div className='flex justify-center
items-center h-screen'>
      <div className=' border-2 border-gray-200 p-
3 rounded-xl shadow-md bg-gray-50'>
      <p className='text-center font-bold mb-2
text-3xl'>Count: {count}</p>
      <button className=' font-bold w-full
rounded-lg mb-2 text-white bg-gray-400 px-20 py-
1.5' onClick={update}>+</button>
      <p className='text-lg font-
bold'>Calculation: {calculation}</p>
      </div>
    </div>

}
```

RESULT

Count: 0

+

Calculation: 0

Context Api

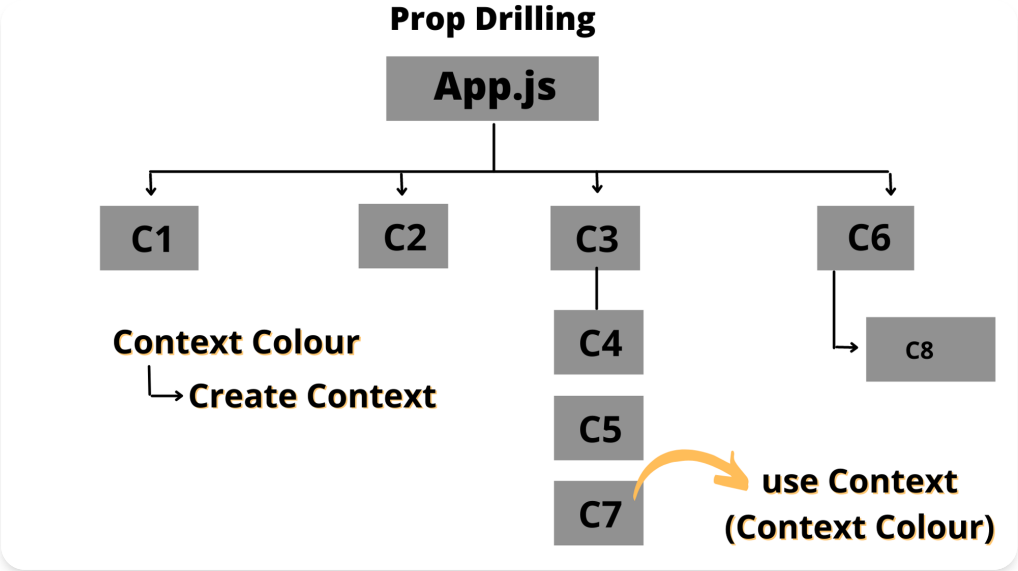
The library for web and native user interfaces

Watch On YouTube

✍️👋 Context Api

🔗 Understanding Context API

The `Context API` can be used to `share` data with multiple `components`, without having to `pass` data through props `manually`. Let's deal with the similar prop drilling issue with the help of `context API` as shown below:



🔗 Getting Started - Context API

```
- context
  - data
    - myContext.jsx
    - myState.jsx
```

🔗 Creating Context - In myContext.jsx

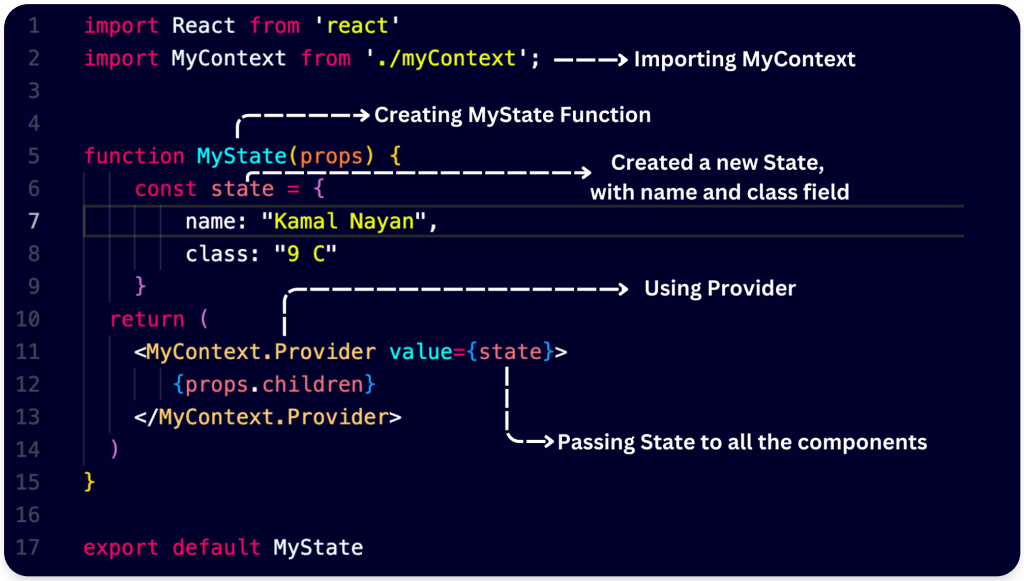
First of all, we have to import `createContext` in the file by using the import command. After that, we will be creating a new context with the help of a predefined syntax and finally, we will export the `myContext` as shown in the below figure:

```
import {createContext} from 'react';

const myContext = createContext();
export default myContext;
```

🔗 Creating State - In myState.jsx

```
import React from 'react'
import MyContext from './myContext';

function MyState(props) {
    const state = {
        name: "Kamal Nayan",
        rollNumber: 15
    }
  return (
    <MyContext.Provider value={state}>
       {props.children}
    </MyContext.Provider>
  )
```

```
}

export default MyState
```



🔗 Create Some Components

```
- components
  - ComponentOne.jsx
  - ComponentTwo.jsx
```

🔗 Import MyState in App.Js

```
import React from 'react'
import Home from "./component/ComponentOne";
import Home from "./component/ComponentTwo";
import MyState from "./context/data/myState";

function App() {
  return (
    <MyState>
      <ComponentOne />
      <ComponentTwo />
    </MyState>
```

```
    )
}

export default App
```

🔗 Using the Created Context -In ComponentOne.jsx and ComponentTwo.jsx

Let's check if the created context is working or not. In our case, we are using the `ComponentOne` component for testing purposes.

`ComponentOne.jsx`

```
import React, { useContext } from 'react'
import myContext from '../context/data/myContext

function ComponentOne() {
    const context = useContext(myContext);
    console.log(context);
    const {name, rollNumber, color} = context;
  return (
    <div className=' bg-red-300 p-2'>
         ComponentOne
        <h2>Name : {name}</h2>
        <h2>Roll Number : {rollNumber}</h2>
        <h2>My Color : {color}</h2>
    </div>
  )
}

export default ComponentOne
```

`ComponentTwo.jsx`

```
import React, { useContext } from 'react'
import myContext from '../context/data/myContext

function ComponentTwo() {
```

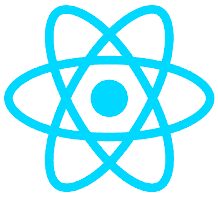
```
    const context = useContext(myContext);
    const {name, rollNumber, color} = context
  return (
    <div className=' bg-green-300 p-2'>
         ComponentTwo
        <h1>Name: {name}</h1>
        <h1>Roll Number : {rollNumber}</h1>
        <h1>My Color : {color}</h1>
    </div>
  )
}

export default ComponentTwo
```

🔗 Multiple State

```
import React from 'react'
import MyContext from './myContext'

function MyState(props) {
    const state = {
        name : 'Kamal Nayan Upadhyay',
        rollNumber : 15
    }
    const color = 'red'
  return (
```

```
      <MyContext.Provider value={{state, color}}>
          {props.children}
      </MyContext.Provider>
    )
}
export default MyState
```

Devknus



UseRef Hook

The library for web and native user interfaces

Watch On YouTube

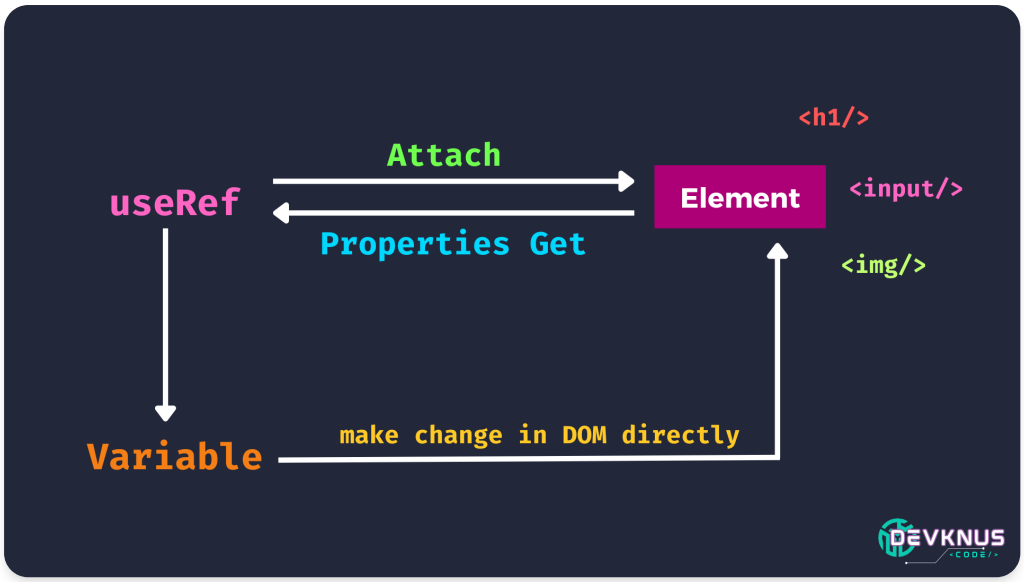
✍️👋 useRef Hook

The useRef Hook allows you to persist values between renders.

It can be used to store a mutable value that does not cause a re-render when updated.

It can be used to access a DOM element directly.

```
const ref = useRef(initialValue);
```

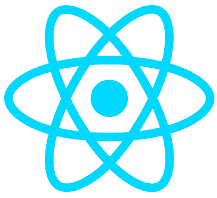


LIVE EDITOR

```
function App() {
    const [inputValue, setInputValue] =
useState("");
    const count = useRef(0);

    useEffect(() => {
      count.current = count.current + 1;
    });

    return (
      <div className="flex justify-center">
       <div>
       <input
       type="text"
       value={inputValue}
       onChange={(e) =>
setInputValue(e.target.value)}
     />
     <h2>Render Count: {count.current}</h2>
       </div>
      </div>
    );
  }
```

RESULT

Render Count: 0



Fetching Data From Api

The library for web and native user interfaces

Watch On YouTube

✍️👋 Fetching Data From Api

We can fetch data by using the Javascript fetch() method. The Fetch API is a new standard to server requests with Promises, it also includes additional features.

```
import React, { useEffect, useState } from 'reac

function App() {
  //* create product state
  const [products, setProducts] = useState([]);

  //* fetch data by using the Javascript fetch()
  const getProduct = async () => {
    const res = await fetch('https://myfirstapi-
    const productData = await res.json();
    console.log(productData.products);
    setProducts(productData.products)
```

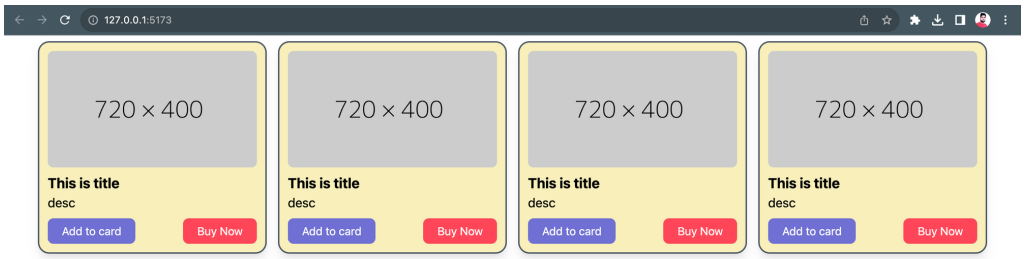
```
  }
  //* get Product automatically
  useEffect(() => {
    getProduct();
  }, []);
  return (
    <div>
      <div className='flex flex-wrap px-4 lg:px-
        {products.map((item, index) => {
          const { title, price, image, descripti
          return (
            <div key={index} className="p-2 md:w
              <div className="bg-[#F8EFBA] p-3 r
                <img className='rounded-lg w-ful
                <h2 className='text-xl text-blac
                  {title.substr(0, 20)}
                </h2>
                <h2 className='text-xl text-blac
                  ₹ {price}
                </h2>
                <h2 className='text-lg text-blac
                  {description}
                </h2>
                <div className=" flex  space-x-2
                  <button className='bg-[#706fd3
                     Add to card
                  </button>
                  <button className='bg-[#ff4757
                     Buy Now
                  </button>
                </div>
              </div>
            </div>
          )
        })}
      </div>
    </div>
  )
}
```

```
export default App
```

🔗 Product Card

```
<div>
      <div className='flex flex-wrap px-4 lg:px-
        <div className="p-2 md:w-1/4 w-full">
          <div className="bg-[#F8EFBA] p-3 round
            <img className='rounded-lg w-full mb
            <h2 className='text-xl text-black fo
            <h2 className='text-lg text-black mb
            <div className=" flex  space-x-2 jus
              <button className='bg-[#706fd3] px
               Add to card
              </button>
              <button className='bg-[#ff4757] px
               Buy Now
              </button>
            </div>
          </div>
        </div>

        <div className="p-2 md:w-1/4 w-full">
          <div className="bg-[#F8EFBA] p-3 round
```

```
            <img className='rounded-lg w-full mb
            <h2 className='text-xl text-black fo
            <h2 className='text-lg text-black mb
            <div className=" flex  space-x-2 jus
              <button className='bg-[#706fd3] px
              <button className='bg-[#ff4757] px
            </div>
          </div>
        </div>

        <div className="p-2 md:w-1/4 w-full">
          <div className="bg-[#F8EFBA] p-3 round
            <img className='rounded-lg w-full mb
            <h2 className='text-xl text-black fo
            <h2 className='text-lg text-black mb
            <div className=" flex  space-x-2 jus
              <button className='bg-[#706fd3] px
              <button className='bg-[#ff4757] px
            </div>
          </div>
        </div>

        <div className="p-2 md:w-1/4 w-full">
          <div className="bg-[#F8EFBA] p-3 round
            <img className='rounded-lg w-full mb
            <h2 className='text-xl text-black fo
            <h2 className='text-lg text-black mb
            <div className=" flex  space-x-2 jus
              <button className='bg-[#706fd3] px
              <button className='bg-[#ff4757] px
            </div>
          </div>
        </div>

      </div>
    </div>
```

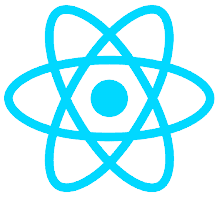


🔗 Two Projects Related To Fetch Api

Movie Searching : https://youtu.be/lu_v7KPFcWU?si=-MUjjyn6LjYm_J7z

Food Searching : https://youtu.be/KS2ch0pn1wk?si=gCADRlZRq9BLEOQQ

Devknus © 2023 Devknus — @devknus



Fetch Data using Axios

The library for web and native user interfaces

Watch On YouTube

✍️👋 Fetch Data using Axios

🔗 What is Axios?

Axios is lightweight package and use to make HTTP requests in Any Javascript Library like React, Angular or Vue. Axios makes it easy to send asynchronous HTTP requests to REST endpoints and perform CRUD operations. If you use Fetch method in Javascript, Axios is the “Easy to use” Version of Fetch.

🔗 Advantages of using Axios

1. Axios by default Work in JSON format.So no more JSON parsing.

2. Make all types of HTTP requests (GET, POST, PUT, DELETE)

🔗 How to install Axios in React App

Like any other npm package, you have to simply install Axios package in your React Application and import axios from the axios package.

Install Axios https://www.npmjs.com/package/axios

```
npm install axios
or
yarn add axios
```

🔗 Use Axios in React with Promises and Error Handling

```
import axios from 'axios';
import React, { useEffect, useState } from 'reac

function App() {
  const [products, setProducts] = useState([]);

  //* By Axios using Promises
  const getProduct = () => {
    axios.get('https://myfirstapi-data.vercel.ap
      .then((response) => setProducts(response.d
      .catch((error) => console.log(error))
  }

  //* Get Product Automatically
  useEffect(() => {
    getProduct();
  }, []);
  return (
    <div>
      <div>
        <div className='flex flex-wrap px-4 lg:p
          {products.map((item, index) => {
            const { title, price, image, descrip
            return (
              <div key={index} className="p-2 md
                <div className="bg-[#F8EFBA] p-3
                  <img className='rounded-lg w-f
                  <h2 className='text-xl text-bl
```

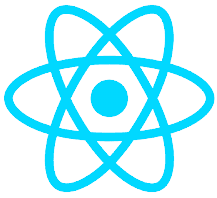
```
                    <h2 className='text-xl text-bl
                    <h2 className='text-lg text-bl
                    <div className=" flex  space-x
                      <button className='bg-[#706f
                         Add to card
                      </button>
                      <button className='bg-[#ff47
                         Buy Now
                      </button>
                    </div>
                  </div>
                </div>
              )
            })}
          </div>
        </div>
      </div>
    )
}

export default App
```

🔗 By Axios using Async & Await Also Error Handling woth Try Catch

```
import axios from 'axios';
import React, { useEffect, useState } from 'reac

function App() {
  const [products, setProducts] = useState([]);

  // //* By Axios using Promises
  // const getProduct = () => {
  //   axios.get('https://myfirstapi-data.vercel
  //     .then((response) => setProducts(respons
  //     .catch((error) => console.log(error))
```

```
// }

// useEffect(() => {
//   getProduct();
// }, []);

//* By Axios using Async & Await Also Error Ha
const getProduct = async () => {
  const res = await axios.get('https://myfirst
  // console.log(res.data.products);
  setProducts(res.data.products);
}

//* Get Product Automatically
useEffect(() => {
  getProduct();
}, []);


return (
  <div>
    <div>
      <div className='flex flex-wrap px-4 lg:p
        {products.map((item, index) => {
          const { title, price, image, descrip
          return (
            <div key={index} className="p-2 md
              <div className="bg-[#F8EFBA] p-3
                <img className='rounded-lg w-f
                <h2 className='text-xl text-bl
                <h2 className='text-xl text-bl
                <h2 className='text-lg text-bl
                <div className=" flex  space-x
                  <button className='bg-[#706f
                     Add to card
                  </button>
                  <button className='bg-[#ff47
                     Buy Now
                  </button>
```

```
                    </div>
                  </div>
                </div>
              )
            })}

          </div>
        </div>
      </div>
    )
}

export default App
```



React Animation

The library for web and native user interfaces

Watch On YouTube

✍️👋 React Awesome Reveal

React Awesome Reveal is a library for React apps that provides a set of curated animated primitives. They can be used to add revealing animations to your components when they enter the browser viewport as the user scrolls the page.

🔗 Installation

Inside your React project directory, run the following:

```
yarn add react-awesome-reveal @emotion/react
```

Or, if you are using `npm`:

```
npm install react-awesome-reveal @emotion/react
```

Or, if you are a `pnpm` user (🙌 if this is your case):

🔗 Quick Start

Import any of the animated components ✨ and wrap your elements:

```
import { Fade } from "react-awesome-reveal";

function App() {
  return (
    <Fade>
      <p>I am an animated text</p>
    </Fade>
  );
}
```

🔗 Example

React Animation : https://reactanimation-two.vercel.app

🔗 React Animation Github Link

React Animation : https://github.com/88kamal/reactanimation

🔗 React Awesome Reveal

React Awesome Reveal : https://react-awesome-reveal.morello.dev

React Redux Tookit

Redux is a state management tool that works as a “centralized store,”

Watch On YouTube

✍️👋 React Redux ToolKit

🔗 What is Redux

Redux is a state management tool that works as a “centralized store,” which means that it is the only place to access the state, Also known as **“The single source of truth.”**

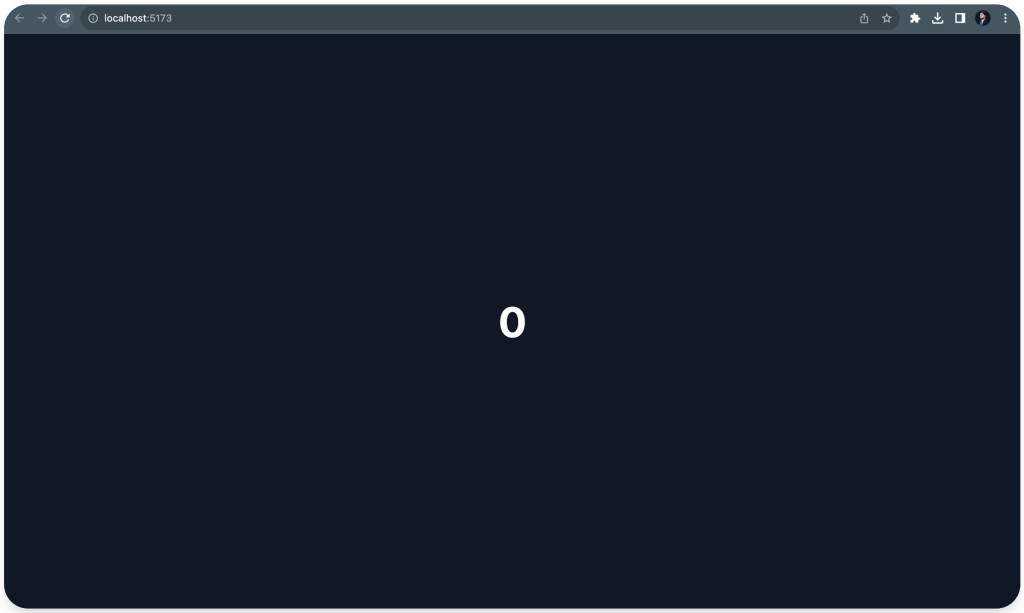
The **single source of truth** is the only place where the application’s state lives and can be accessed.

🔗 To understand Redux, you must understand the “one-way data flow” known as unidirectional data flow.

```
1  import React, { useState } from 'react'
2
3  function App() {
4    const [number, setNumber] = useState(0);
5    return (
6      <div>
7        <span onClick={() => setNumber(number + 1)}>
8          {number}
9        </span>
10     </div>
11   )
12 }
13
14 export default App
```

Step 1 : You have a number as a `state` with an initial value: 0 (line 4)

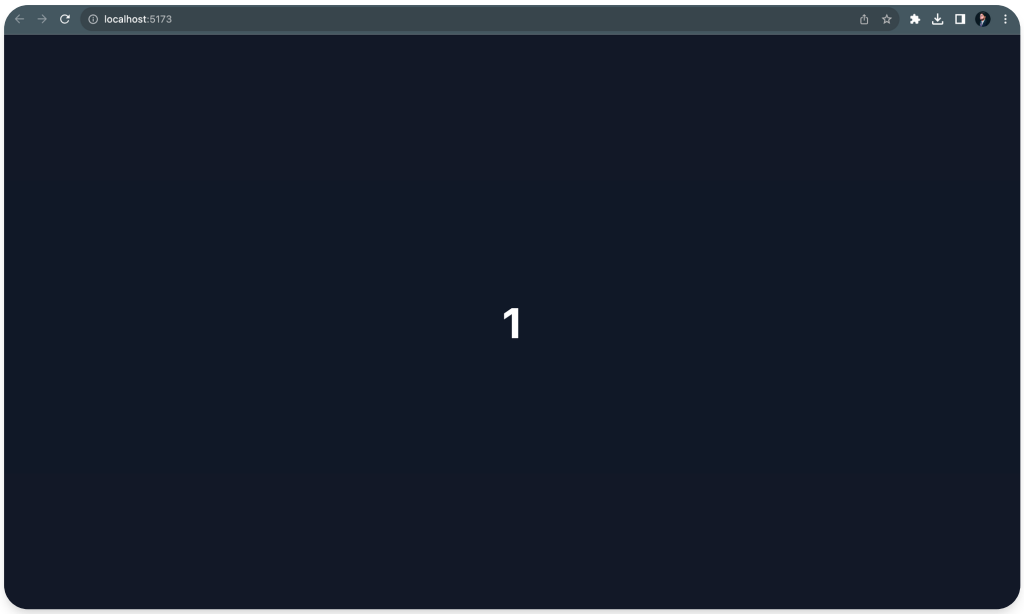
Step 2 : You render this `state` in the view (line 8)



Step 3 : You trigger an action to change/update the `state` (line 7)

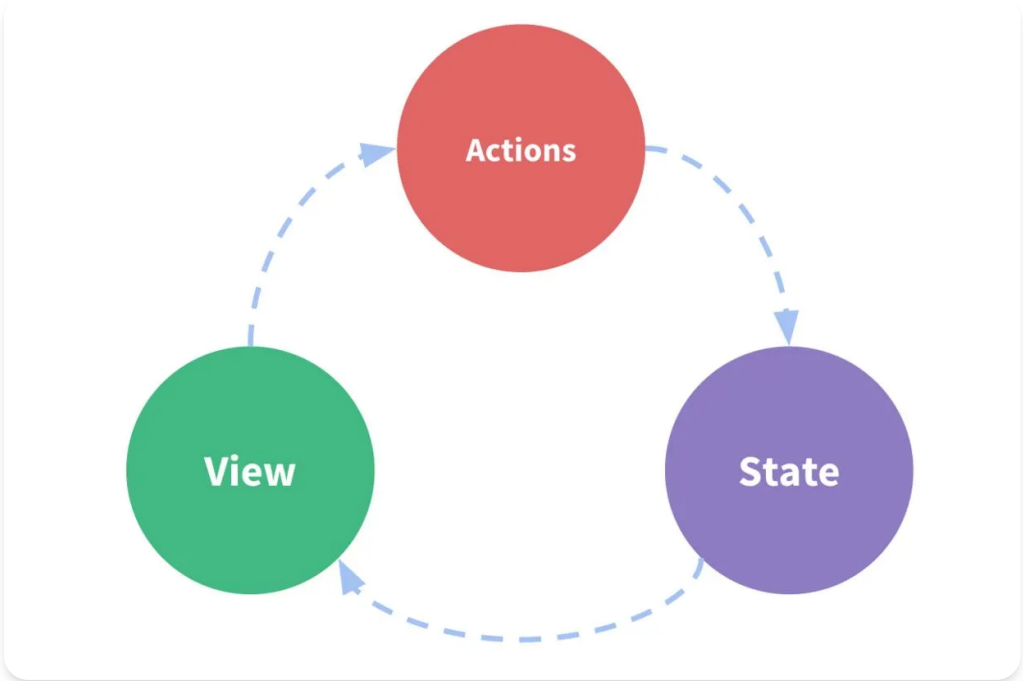
Step 4 : The `state` notifies the subscribing component(s) using the state that it got changed.

Step 5 : The component(s) re-render themselves to display the new state.

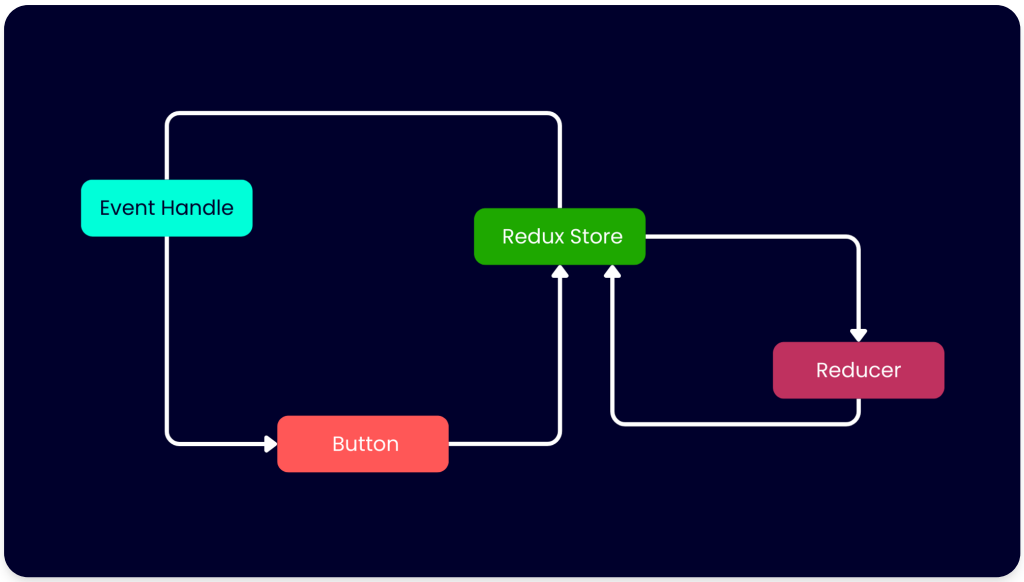


🔗 Pattern

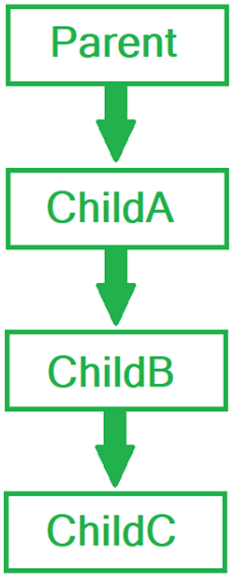
You have a state that gets rendered in a `view` and the view trigger an `action` to change the `state` which will eventually change the `view`.



This is exactly how Redux conceptually works.



🔗 Why Use Redux



🔗 1. Install Redux Toolkit and React-Redux

Add the Redux Toolkit and React-Redux packages to your project:

```
npm install @reduxjs/toolkit react-redux
```

🔗 2. Create a Redux Store

Add the Redux Toolkit and React-Redux packages to your project:

Create a file named `src/app/store.jsx`. Import the `configureStore` API from Redux Toolkit. We'll start by creating

an empty Redux store, and exporting it:

```
1. Store is a state container which contain the

2. It is an object.

3. Every Applicationcan have single store.

4. And in store need to specify the reducer
```

`src/app/store.jsx`

```
import { configureStore } from '@reduxjs/toolkit

export const store = configureStore({
  reducer: {},
})
```

This creates a Redux store, and also automatically configure the Redux DevTools extension so that you can inspect the store while developing

🔗 3. Provide the Redux Store to React

Once the store is created, we can make it available to our React components by putting a React-Redux `<Provider>` around our application in src/index.js. Import the Redux store we just created, put a `<Provider>` around your `<App>`, and pass the store as a prop:

`src/main.jsx`

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'
import App from './App'
import { store } from './app/store'
import { Provider } from 'react-redux'
```

```
ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
)
```

🔗 4. Create a Redux State Slice

Add a new file named `src/features/counter/counterSlice.jsx`.

In that file, import the `createSlice` API from Redux Toolkit.

Creating a slice requires a string name to identify the slice, an initial state value, and one or more reducer functions to define how the state can be updated. Once a slice is created, we can export the generated Redux action creators and the reducer function for the whole slice.

`src/features/counter/counterSlice.jsx`

```
import { createSlice } from '@reduxjs/toolkit'

const initialState = {
  value: 0,
}

export const counterSlice = createSlice({
  name: 'counter',
  initialState,
  reducers: {
    increment: (state) => {
      state.value += 1
    },
    decrement: (state) => {
      state.value -= 1
    },
    incrementByAmount: (state, action) => {
      state.value += action.payload
```

```
    },
  },
})

// Action creators are generated for each case r
export const { increment, decrement, incrementBy

export default counterSlice.reducer
```

🔗 5. Add Slice Reducers to the Store

Next, we need to import the reducer function from the counter slice and add it to our store. By defining a field inside the reducer parameter, we tell the store to use this slice reducer function to handle all updates to that state.

`app/store.jsx`

```
import { configureStore } from '@reduxjs/toolkit
import counterSlice from '../features/counter/co

export const store = configureStore({
  reducer: {
    counter: counterSlice,
  },
})
```

🔗 6. Use Redux State and Actions in React Components

Now we can use the React-Redux hooks to let React components interact with the Redux store. We can read data from the store with useSelector, and dispatch actions using useDispatch. Create a src/components/Counter.js file with a `<Counter>` component inside, then import that component into App.js and render it inside of `<App>`.

🔗 Get And Update state in Redux Store

`UseSelector`

```
The useSelector hook is used to extract the stat
```

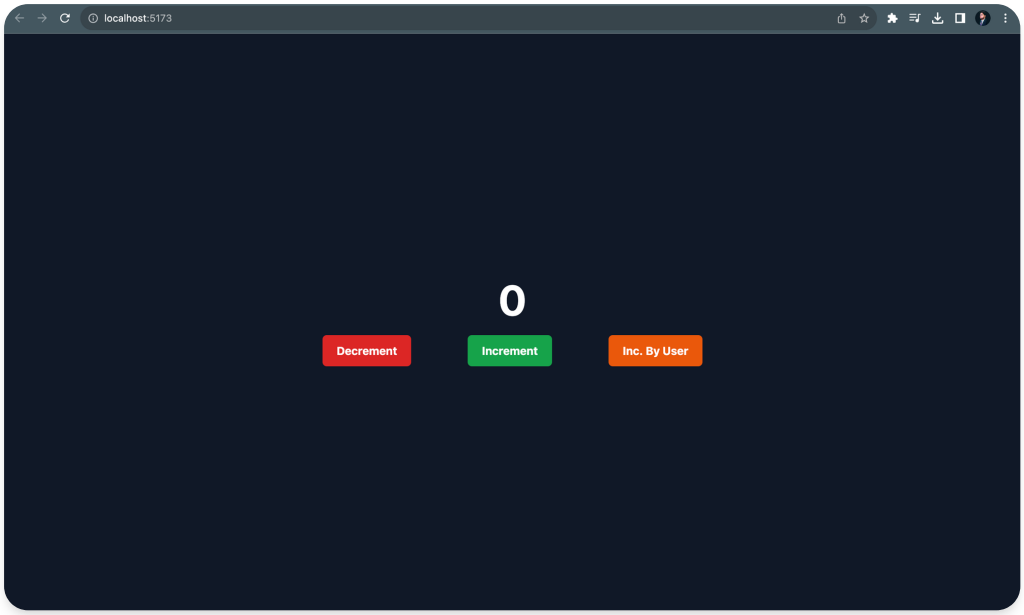
`useDispatch`

```
The useDispatch hook is used to update the state
```

Counter

```
<div className=' flex space-x-3 justify-center i
      <div className="">

        {/* Read  */}
        <p className=' text-6xl font-bold text-c

        {/* Increment Button  */}
        <button
          className=' bg-red-600 text-white py-2
        >
          Increment
        </button>

        {/* Decrement Button  */}
        <button
          className=' ml-20 bg-green-600 text-wh
        >
          Decrement
        </button>

        {/* Increment By User */}
        <button
          className=' ml-20 bg-orange-600 text-w
        >
          Inc. By User
        </button>
```

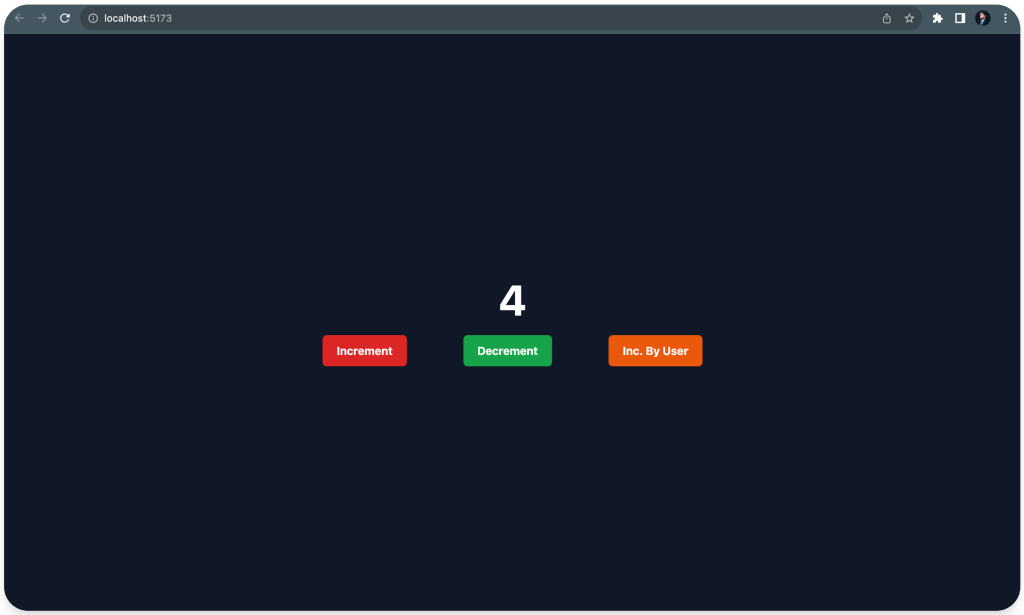
```
      </div>
    </div>
```



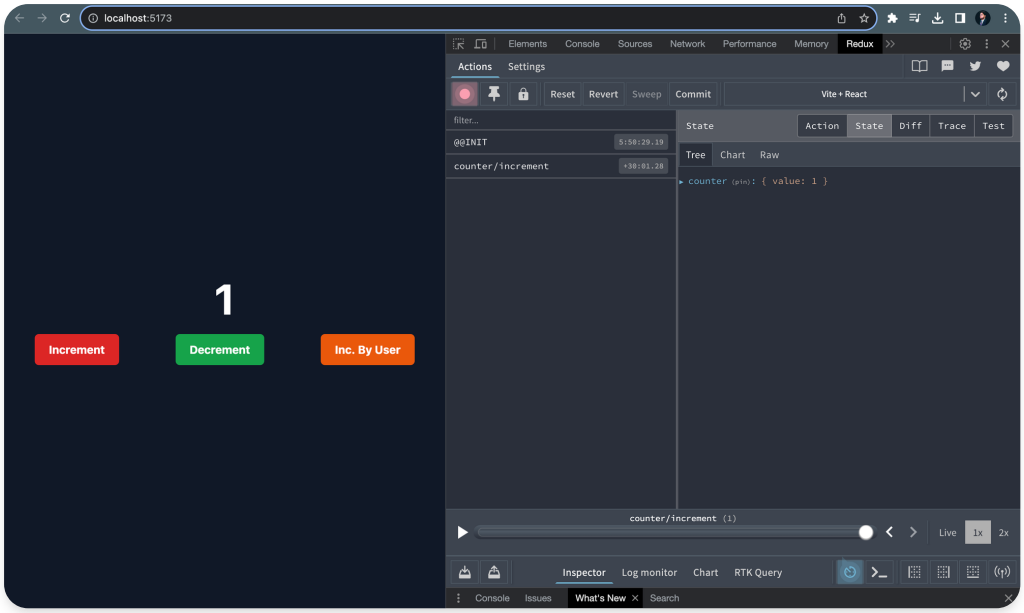
🔗 src/App.jsx

```
import React from 'react'
import { useSelector, useDispatch } from 'react-
import { decrement, increment,incrementByAmount


export function App() {
  const count = useSelector((state) => state.cou
  const dispatch = useDispatch()

  return (
    <div className=' flex space-x-3 justify-cent
      <div className="">

        {/* Read  */}
        <p className=' text-6xl font-bold text-c

        {/* Increment Button  */}
        <button
          className=' bg-red-600 text-white py-2
          onClick={() => dispatch(increment())}
```

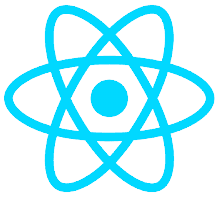
```
          >
            Increment
          </button>

          {/* Decrement Button  */}
          <button
            className=' ml-20 bg-green-600 text-wh
            onClick={() => dispatch(decrement())}
          >
            Decrement
          </button>

          {/* Increment By User */}
          <button
            className=' ml-20 bg-orange-600 text-w
            onClick={() =>dispatch(incrementByAmou
          >
            Inc. By User
          </button>
        </div>
      </div>
    )
}

export default App
```



🔗 Redux Dev Tool



Devknus



Creating a Protected Routes In React Js

The library for web and native user interfaces

Watch On YouTube

✍️👋 Creating a Protected Routes In React Js

🔗 Step 1 : Create a Pages

```
- pages
  - home
    - Home.jsx
  - product
    - Product.jsx
  - dashboard
    - Dashboard.jsx
```

`Home.jsx`

```
import React from 'react'

function Home() {
```

```
    return (
        <div className='flex justify-center item
            Home
        </div>
    )
}

export default Home
```

`Product`

```
import React from 'react'

function Product() {
    return (
        <div className='flex justify-center item
            Product
        </div>
    )
}

export default Product
```

`Dashboard.jsx`

```
import React from 'react'

function Dashboard() {
    return (
        <div className='flex justify-center item
            Dashboard
        </div>
    )
}

export default Dashboard
```

🔗 Step 2 : Install React Router Dom

```
npm i react-router-dom
```

🔗 Step 3 : App.jsx

```
import React from 'react';
import {
  BrowserRouter as Router,
  Route,
  Routes,
} from "react-router-dom";
import Home from './pages/home/Home';
import Product from './pages/product/Product';
import Dashboard from './pages/dashboard/Dashboa

function App() {
  return (
    <>
      <Router>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/product" element={<Produ
          <Route path="/dashboard" element={<Das
        </Routes>
      </Router>
    </>
  )
}

export default App
```

🔗 Step 4 : Create a Components

```
- components
  - navbar
    - Navbar.jsx
```

Navbar.jsx

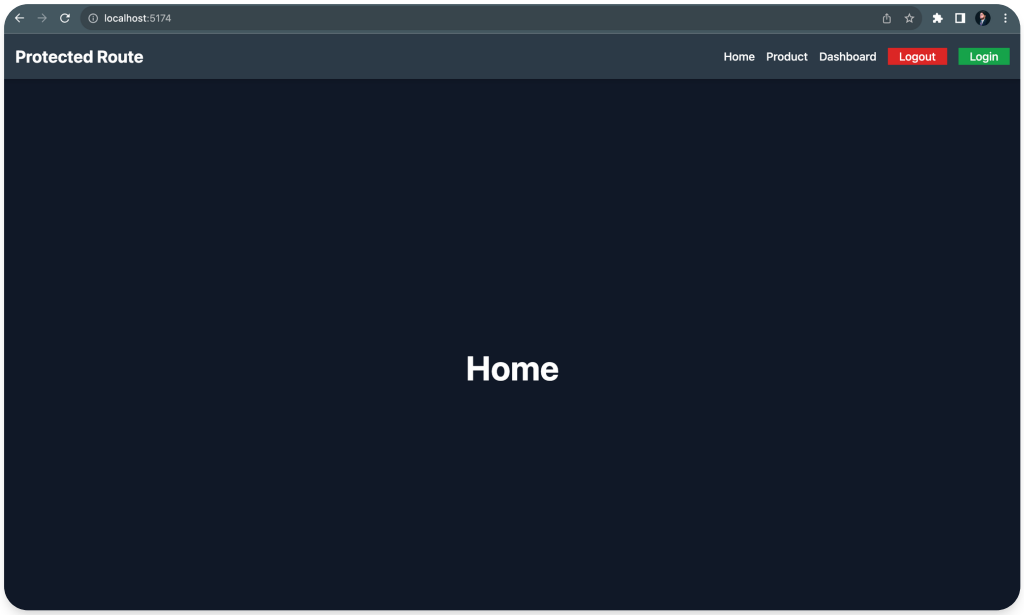
```
import React from 'react'
import { Link } from 'react-router-dom'

function Navbar() {
    return (
        <div className='main lg:flex md:flex fle
        items-center px-4 bg-[#2C3A47] py-4 shad
            <div className="left">
                <Link to={'/'}>
                    <div className="logo font-bo
                        Protected Route
                    </div>
                </Link>
            </div>
            <div className="right">
                <ul className='flex space-x-4 te
                    <Link to={'/'}>
                        <li className='cursor-po
                    </Link>
                    <Link to={'/product'}>
                        <li className='cursor-po
                    </Link>
                    <Link to={'/dashboard'}>
                        <li className='cursor-po
                    </Link>
                    <li className='cursor-pointe
                        Logout
                    </li>
                    <li className='cursor-pointe
                        Login
                    </li>
```

```
                </ul>
            </div>
        </div>
    )
}

export default Navbar
```

🔗 Step 5 : Use Navbar in App.jsx

```
import React from 'react';
import {
  BrowserRouter as Router,
  Route,
  Routes,
} from "react-router-dom";
import Home from './pages/home/Home';
import Product from './pages/product/Product';
import Dashboard from './pages/dashboard/Dashboa
import Navbar from './components/navbar/Navbar';

function App() {
  return (
    <>
      <Router>
        <Navbar/>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/product" element={<Produ
          <Route path="/dashboard" element={<Das
        </Routes>
      </Router>
    </>
  )
}
```

```
export default App
```



🔗 Step 6 : Create Protected Route

Create One useState

```
const [user, setUser] = useState(false);
```

Create Two Function Login And Logout

`Login`

```
const login = () =>{
    setUser(true);
}
```

`Logout`

```
const logout = () =>{
    setUser(false);
}
```

Create Protected Route Function in App.jsx

```
export const ProtectedRoute = ({ user, children
  if (user) {
    return children
  }
  else {
    return <Navigate to={'/'} />
  }
}
```

App.jsx

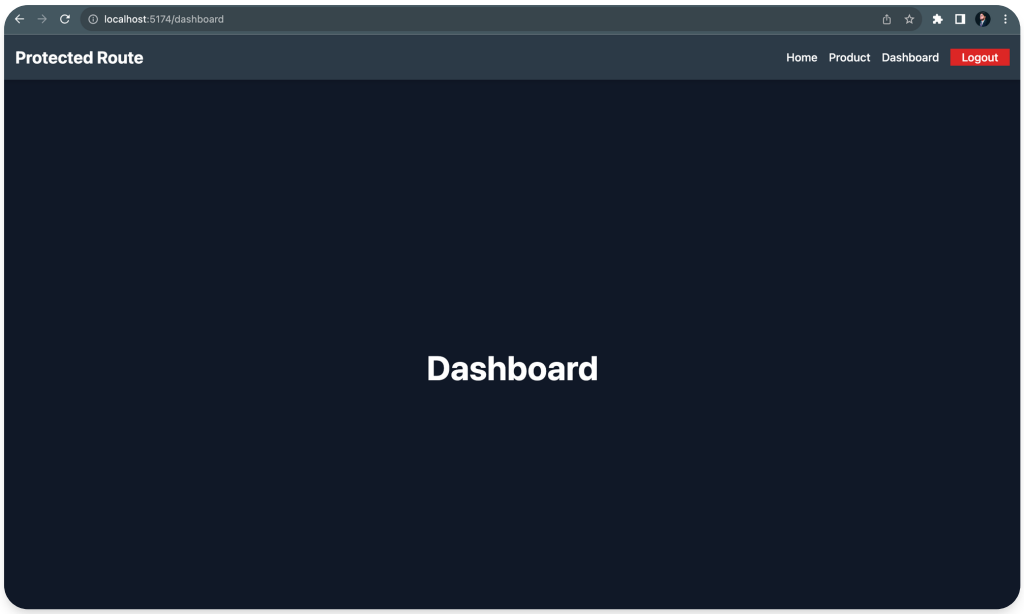
```
import React, { useState } from 'react';
import {
  BrowserRouter as Router,
  Route,
  Routes,
  Navigate,
} from "react-router-dom";
import Home from './pages/home/Home';
import Product from './pages/product/Product';
import Dashboard from './pages/dashboard/Dashboa
import Navbar from './components/navbar/Navbar';

function App() {
  const [user, setUser] = useState(false);

  //* create Login Funtion
  const login = () =>{
    setUser(true);
  }

  //* create logout Funtion
  const logout = () =>{
    setUser(false);
  }
```

```
  return (
    <>
      <Router>
        <Navbar user={user} login={login} logout
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/product" element={<Produ
          <Route path="/dashboard" element={
            <ProtectedRoute user={user}>
              <Dashboard />
            </ProtectedRoute>
          } />
        </Routes>
      </Router>
    </>
  )
}

export default App

//* Create Protected Route Function
export const ProtectedRoute = ({ user, children
  if (user) {
    return children
  }
  else {
    return <Navigate to={'/'} />
  }
}
```

Navbar.jsx

```
import React from 'react'
import { Link } from 'react-router-dom'

function Navbar({ user, login, logout }) {
```

```
    return (
        <div className='main lg:flex md:flex fle
        items-center px-4 bg-[#2C3A47] py-4 shad
            <div className="left">
                <Link to={'/'}>
                    <div className="logo font-bo
                        Protected Route
                    </div>
                </Link>
            </div>
            <div className="right">
                <ul className='flex space-x-4 te
                    <Link to={'/'}>
                        <li className='cursor-po
                    </Link>
                    <Link to={'/product'}>
                        <li className='cursor-po
                    </Link>
                    <Link to={'/dashboard'}>
                        <li className='cursor-po
                    </Link>
                    {user
                        ?
                        <li onClick={logout} cla
                            Logout
                        </li>
                        :
                        <li onClick={login} clas
                            Login
                        </li>
                    }
                </ul>
            </div>
        </div>
    )
}

export default Navbar
```



Devknus

How to Upload Images to Cloudinary in React JS

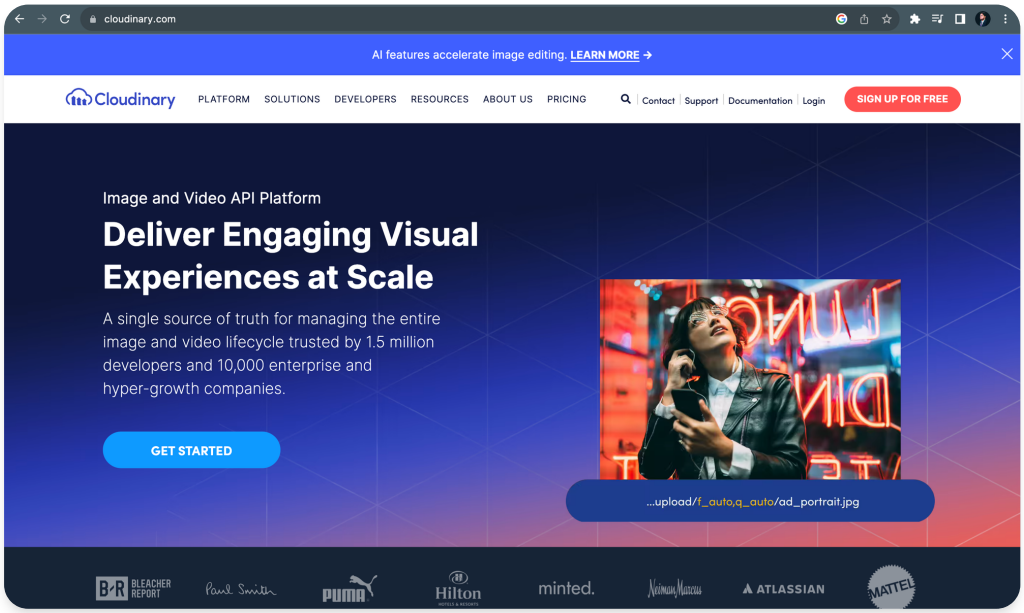
The library for web and native user interfaces

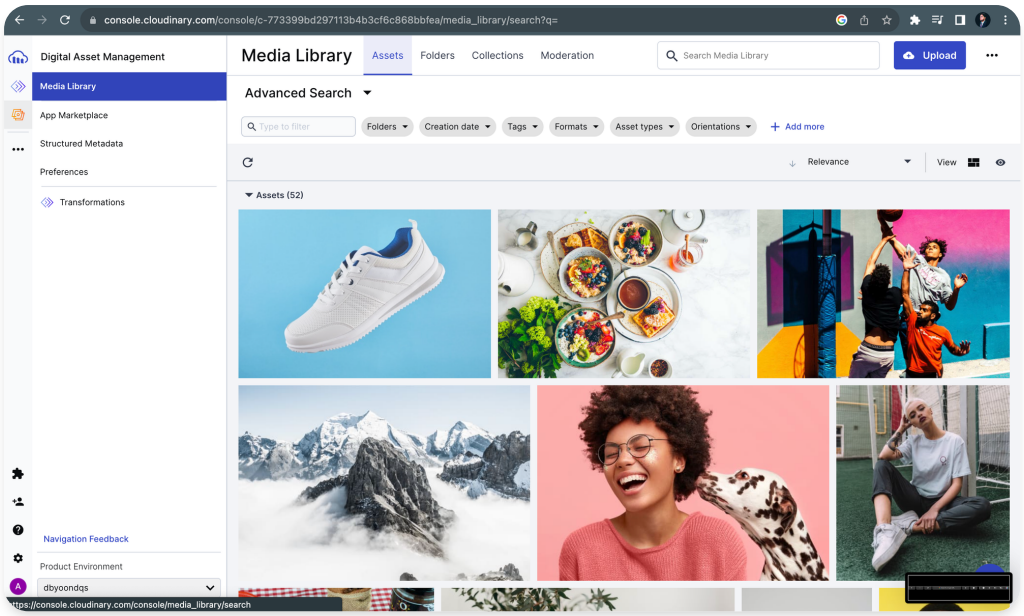
Watch On YouTube

✍️👋 How to Upload Images to Cloudinary in React JS

🔗 1. Go To Cloudinary

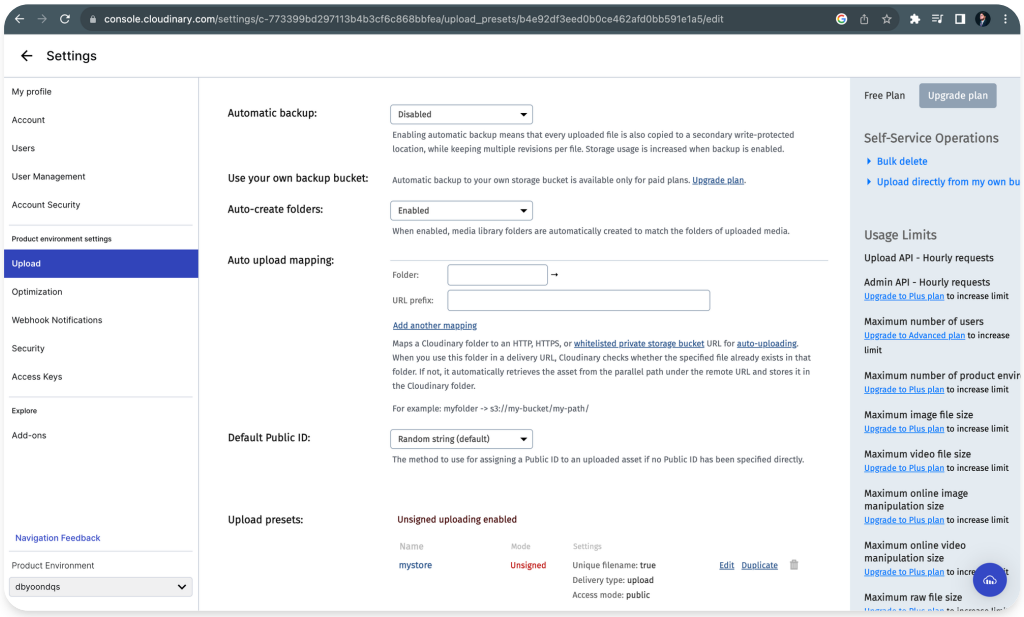
https://cloudinary.com



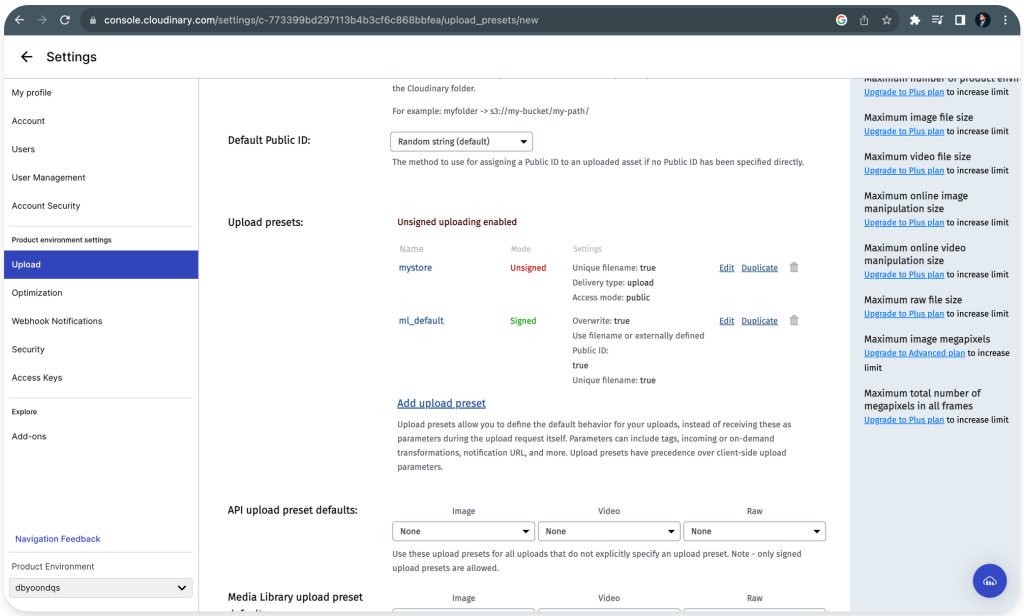


🔗 2. Go To Setting

click on upload



🔗 3. Add upload preset



🔗 4. Index.css

```css
body{
  background-color: #111827;
}

#file-upload {
  display: none;
}
.custom-file-upload {
  display: inline-block;
  cursor: pointer;
}
```

🔗 5. Create Upload Component

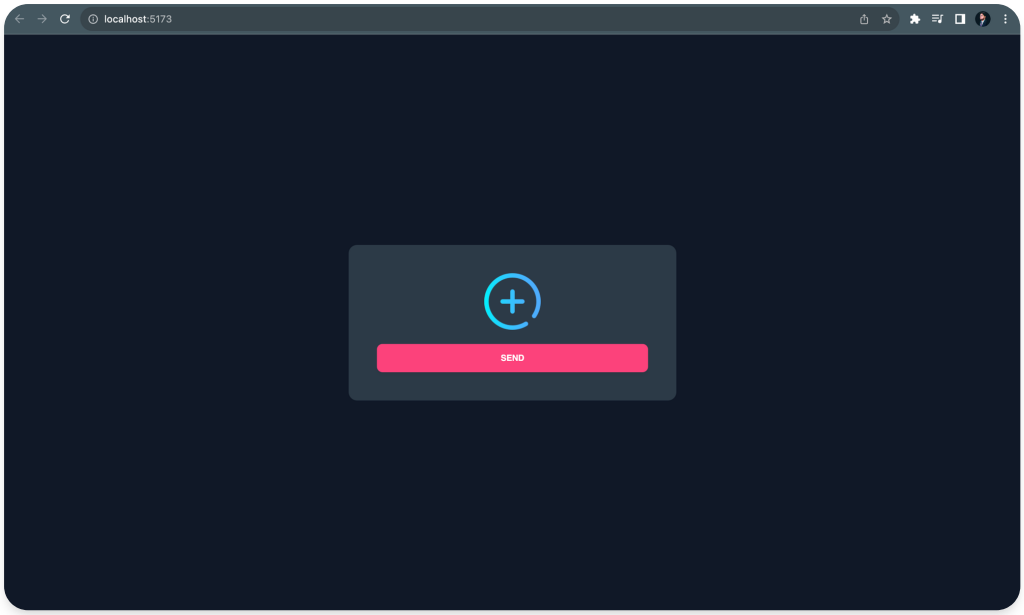
```jsx
<div className='flex justify-center items-center
      {/* Image Upload Section  */}
      <div className=" bg-[#2C3A47] p-10 rounded

        {/* Upload Input And Image Section  */}
        <div className="input flex justify-cente
          <label
```

```
            for="file-upload"
            class="custom-file-upload">
            <img
              src="https://cdn-icons-png.flatico
              className="h-20 w-20"
            />
          </label>

          {/* Image Upload Input */}
          <input
            id="file-upload"
            className=' text-white'
            type="file"
          />
        </div>

        {/* Send Button  */}
        <div className="">
          <Button
            className=' w-72 lg:w-96  bg-[#FC427
          >
             Send
          </Button>
        </div>
      </div>
</div>
```

🔗 6. Go To App.jsx

```
import { Button } from '@material-tailwind/react
import React from 'react'

function App() {
  return (
    <div className='flex justify-center items-ce
      {/* Image Upload Section  */}
      <div className=" bg-[#2C3A47] p-10 rounded
```

```
          {/* Upload Input And Image Section  */}
          <div className="input flex justify-cente
            <label
              for="file-upload"
              class="custom-file-upload">
              <img
                src="https://cdn-icons-png.flatico
                className="h-20 w-20"
              />
            </label>

           {/* Image Upload Input */}
            <input
              id="file-upload"
              className=' text-white'
              type="file"
            />
          </div>

          {/* Send Button  */}
          <div className="">
            <Button
              className=' w-72 lg:w-96  bg-[#FC427
            >
               Send
            </Button>
          </div>
        </div>
      </div>
    )
}

export default App
```



🔗 In App.jsx

Create Two useState

```
const [image, setImage] = useState('');
const [url, setUrl] = useState('');
```

Create saveImage Function

```
const saveImage = async () => {
    const data = new FormData();
    data.append("file", image);
    data.append("upload_preset", "mystore");
    data.append("cloud_name", "dbyoondqs");

    try {
      if(image === null){
        return toast.error("Please Upload image"
      }

      const res = await fetch('https://api.cloud
        method : "POST",
        body : data
      })
```

```
      const cloudData = await res.json();
      setUrl(cloudData.url);
      console.log(cloudData.url);
      toast.success("Image Upload Successfully")
    } catch (error) {

    }
  }
```

App.jsx

```
import { Button } from '@material-tailwind/react
import React, { useState } from 'react'
import toast, { Toaster } from 'react-hot-toast'

function App() {
  const [image, setImage] = useState(null);
  const [url, setUrl] = useState('');

  const saveImage = async () => {
    const data = new FormData();
    data.append("file", image);
    data.append("upload_preset", "mystore");
    data.append("cloud_name", "dbyoondqs");

    try {
      if(image === null){
        return toast.error("Please Upload image"
      }

      const res = await fetch('https://api.cloud
        method : "POST",
        body : data
      })

      const cloudData = await res.json();
```

```
      setUrl(cloudData.url);
      console.log(cloudData.url);
      toast.success("Image Upload Successfully")
    } catch (error) {

    }
  }

  console.log(url)
  return (
    <div className='flex justify-center items-ce
      <div className=" bg-[#2C3A47] p-10 rounded
        <div className="input flex justify-cente
          <label
            for="file-upload"
            class="custom-file-upload">
            {image
              ? <img
                classNam=" w-72 lg:w-96  rounde
                src={image ? URL.createObjectURL
                alt="img"
              />
              : <img
                src="https://cdn-icons-png.flati
                className="h-20 w-20"
              />}
          </label>
          <input
            id="file-upload"
            className=' text-white'
            type="file"
            onChange={(e) => setImage(e.target.f
        </div>
        <div className="">
          <Button
            className=' w-72 lg:w-96  bg-[#FC427
            onClick={saveImage}
          >
             Send
```

```
            </Button>
            <Toaster />
          </div>
        </div>
      </div>


    )
}

export default App
```