View, add and edit your notes in the app

Project 2 - Contact Us | 10 React Projects for Beginners

Generated on January 28, 2024

Summary

Notes	Screenshots	Bookmarks
31	5	0

previously, we were using the universal css, now

we will use the module css.

module css is the dedicated css which we use to write

to avoid css classes conflicts.

1:05

type: npm create vite@latest

in the powershell to create new project

2:34

npx create-react-app in the powrshell

3:35

import the css modules in the navigation and add in the main tag like this

<nav className={styles.navigation}>

module css works like an object of javascript and its classes like navigation will work like class of an object, and code inside it will work like values of class

17:27

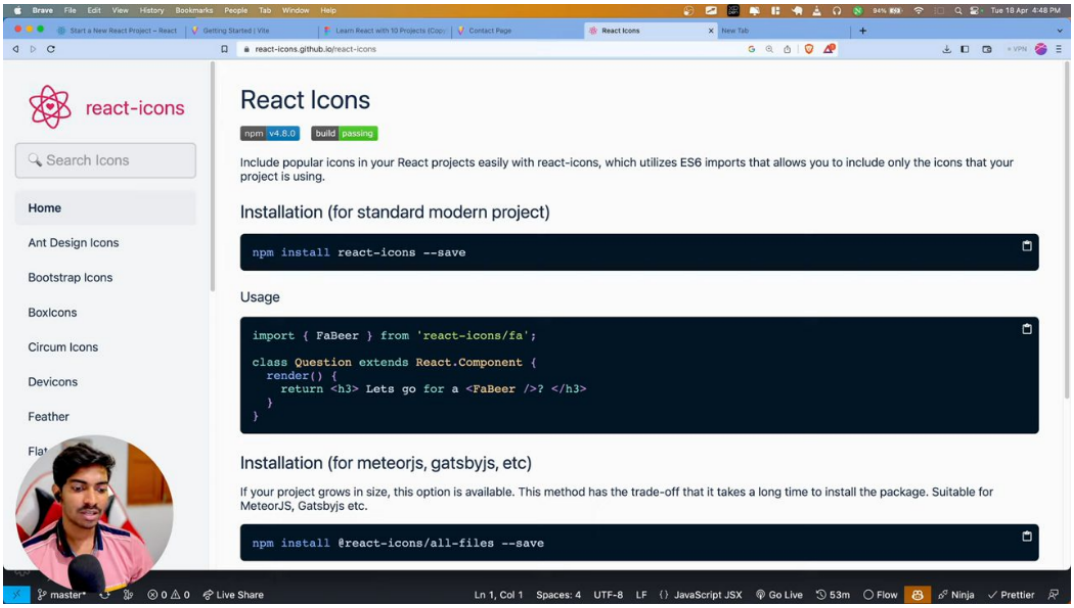
you can also convert that code in curly braces in template literals

18:24

run the following command in terminal to install the icon package

```
npm i react-icons
```

29:54



To install the package, use the following command: [appreciation] If you are able to successfully install the package, then like I wrote this command, then if you install the package Jaisalmer.

29:54

you can verify the installation of icons package by navigating to the package.json > “dependencies“. you can see the installed version

30:12

write the class of module.css like this

<button className={styles.primary_btn}>

34:14

you can increase the size of icon by writing fontSize in the icon tag

35:55

now the button is ready. i have to create another button so will i have to write the whole button again? no, react gives you a power to use the same component again and again with modification

36:27

pass the props like this.

in the form component:

<Button text="VIA SUPPORT CHAT" icon={<MdOutlineMessage/>}/>

<Button text="VIA CALL" icon={<IoCall/>}/>

in the button component:

<button className={styles.primary_btn} fontSize="24px"> {props.icon}{props.text}</button>

40:14

In React, `isOutLine={true}` is a prop passed to the `Button` component. This prop is likely used to specify whether the button should have an outline style or not. When `isOutLine` is `true`, the button would typically have an outline style applied.

Here's the breakdown:

1. `&lt;Button isOutLine={true} text="VIA EMAIL FORM" icon={&lt;CiMail /&gt;} /&gt;`:

- This line is passing the prop `isOutLine` with a value of

`true` to the `Button` component. It indicates that the button should have an outline style.

- The `text` prop specifies the text to be displayed on the button.

- The `icon` prop likely specifies an icon component (such as `&lt;CiMail /&gt;`) to be displayed alongside the text on the button.

2. `&lt;button className={props.isOutLine ? styles.outline_btn : styles.primary_btn}&gt;&lt;/button&gt;`:

- This line is using a ternary operator within a JSX expression to conditionally apply different CSS classes to the button element based on the value of the `isOutLine` prop.

- If `props.isOutLine` evaluates to `true`, it applies the CSS class `styles.outline_btn`.

- If `props.isOutLine` evaluates to `false` (or `undefined`), it applies the CSS class `styles.primary_btn`.

- This allows the button to have different styles based on the value of the `isOutLine` prop, providing flexibility in styling based on the component's props.

45:48

now, you can use object DE structuring in button.js

in this procedure, we add the props like this before the return

const {isOutLine, icon, text } = props;

and then use them below without this.

47:05

you can also directly destrucure the props in the function argument

47:21

to align the label of input box on the line of box, use the following css class

form_control label {

font-size: 14px;

position: absolute;

top: -12px;

left: 10px;

background-color: white;

padding: 0 10px;

}

53:18

now, i have to modify the submit button. i added the following logic

<button className={`${isOutLine ? styles.outline_btn : (text === "Submit" ? styles.submit_btn : styles.primary_btn)}`}>

explaination:

Certainly! Let's break down the line of code:

```jsx

&amp;lt;button className={`${isOutLine ? styles.outline_btn : (text === "Submit" ? styles.submit_btn : styles.primary_btn)}`}&amp;gt;

```

1. **`${...}`**: This part of the code utilizes template literals in JavaScript. It allows us to embed expressions inside strings by using `${...}`. In this case, it's used to dynamically generate the className string based on certain conditions.

2. **`isOutLine ? styles.outline_btn : (text === "Submit" ? styles.submit_btn : styles.primary_btn)`**:

- This is a ternary operator, which is a concise way of writing conditional expressions. It consists of three parts:

- The condition (`isOutLine`): If this condition evaluates to `true`, the first expression (`styles.outline_btn`) is returned.

- If the condition (`isOutLine`) is `false`, the second part of

the ternary operator is evaluated.

- The second part is another ternary operator: `text === "Submit" ? styles.submit_btn : styles.primary_btn`.

  - If `text === "Submit"` evaluates to `true`, `styles.submit_btn` is returned.

  - If `text === "Submit"` evaluates to `false`, `styles.primary_btn` is returned.

3. **`className={`...`}`**: This JSX attribute allows us to set the CSS class of the `<button>` element. The value inside the curly braces is evaluated as a JavaScript expression, which generates a string containing the CSS class name based on the conditions specified.

So, the entire line of code dynamically assigns a CSS class to the `<button>` element based on the values of `isOutLine` and `text`. If `isOutLine` is `true`, it applies the `styles.outline_btn` class. If `isOutLine` is `false` and `text` is "Submit", it applies the `styles.submit_btn` class. Otherwise, it applies the `styles.primary_btn` class.

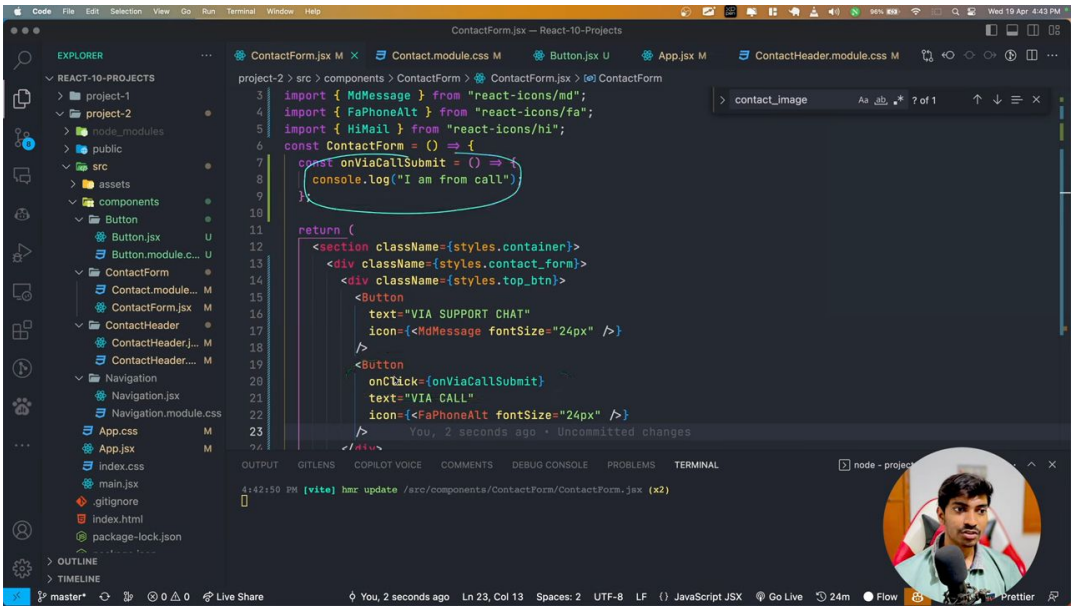
56:13

give form tag position: relative and add postion: absolute and right: 0 in submit_btn class. this will position the button to the right end.

57:08

add ..rest in the destructuring of button.js to run onClick on buttons

1:04:27



A mysterious event is set to unfold, as A prepares to embark on a new journey.

1:04:28

In the context of React, **...rest** is commonly used in functional components to capture any additional props that are passed to a component, but not explicitly destructured or handled by other props. This allows you to pass down any additional props to child components without explicitly specifying them.

1:06:02

In React, **...rest** is a JavaScript feature known as the "rest operator" or "rest parameter". It is used in function declarations to collect all remaining arguments into a single array parameter.

1:06:02

you can find all button events by pressing space along with ctrl inside the button tag in button.js

1:06:13

create an onSubmit function and console.log its event. pass this function in the form tag.

1:07:39

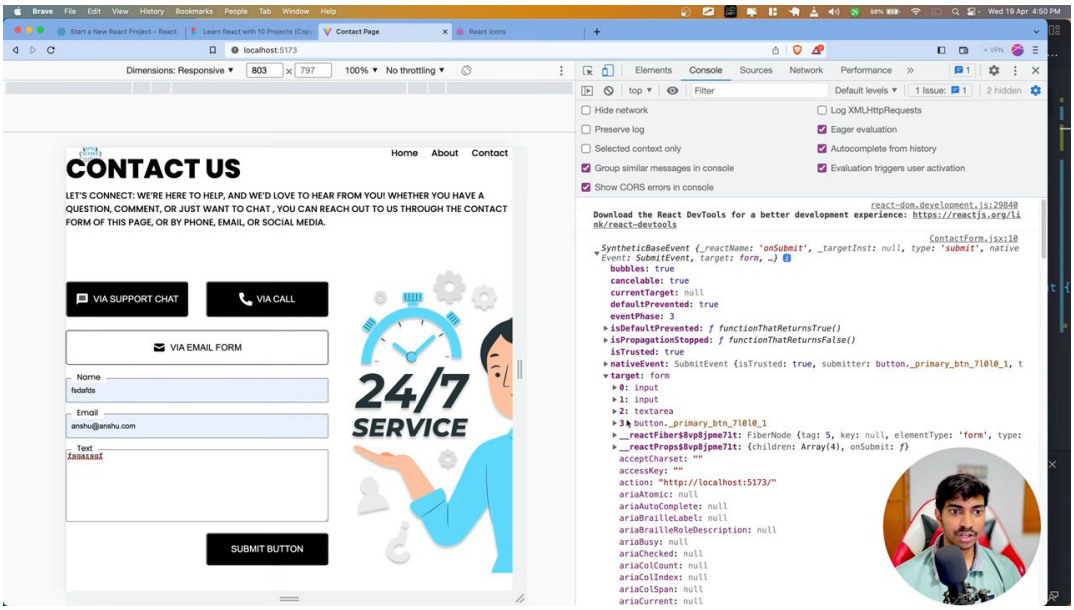
now, after filling the input boxes, when i click on submit button, all entered data is appearing on the above url

1:07:54

well, this is the default behaviour. we have to stop reloading the page.

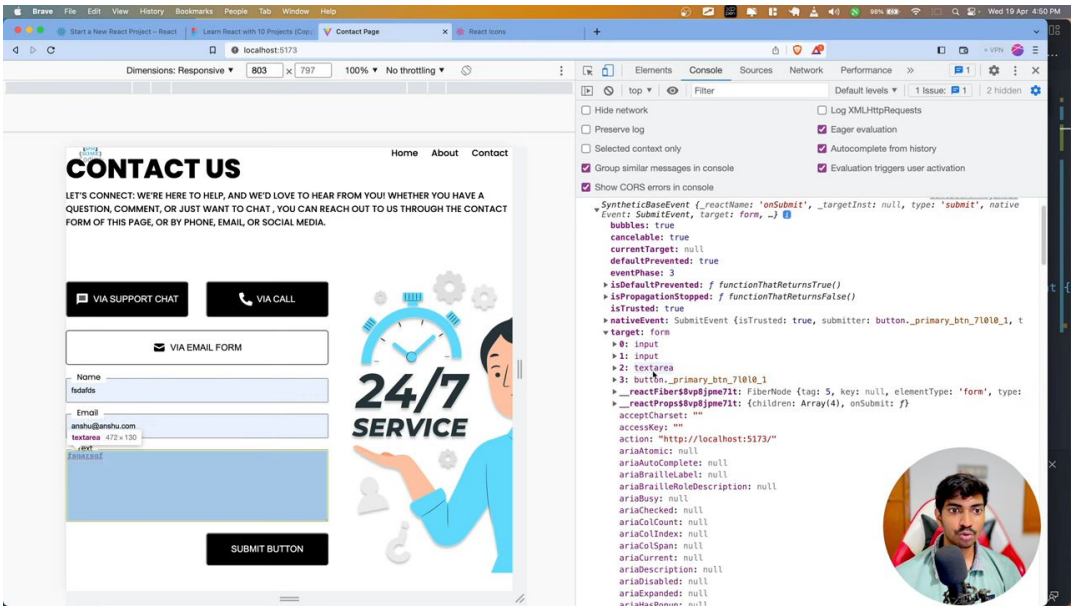
add event.preventDefault(); in the onSubmit function. to prevent default reloading after clicking submit button.

1:08:19



New form-filling technology has been developed to extract and store data from input fields, revolutionizing the way information is collected and utilized.

1:08:54



1:08:56

now, the page is not reloaded. navigating to the console, click on target. there there are inputs. these are the inputs of form input boxes. now, we will fetch the data from these inputs and store them somewhere else.

1:09:05

now, add

console.log("name:",event.target[0].value);

console.log("email:",event.target[1].value);

console.log("text:",event.target[2].value);

in onSubmit function. the entered data in boxes will appear in console.

1:10:40

now, we will show the collected data below the form

1:11:14

now, initialize 3 variables name, email, and text outside the onSubmit function. give them dummy value.

then asign the values of console.log to each variable like this

name = event.target[0].value;

and below the form display the values like this

<div>{name + " " + email + " " + text}</div>

now if you enter the values in the form they should appear below the form. but they are not appearing. they are changing as you can see in the console, but not appearing below the form. you have to use a useState hook to solve this problem.

1:14:35

use useState hook.

const [name, SetName] = useState("usman");

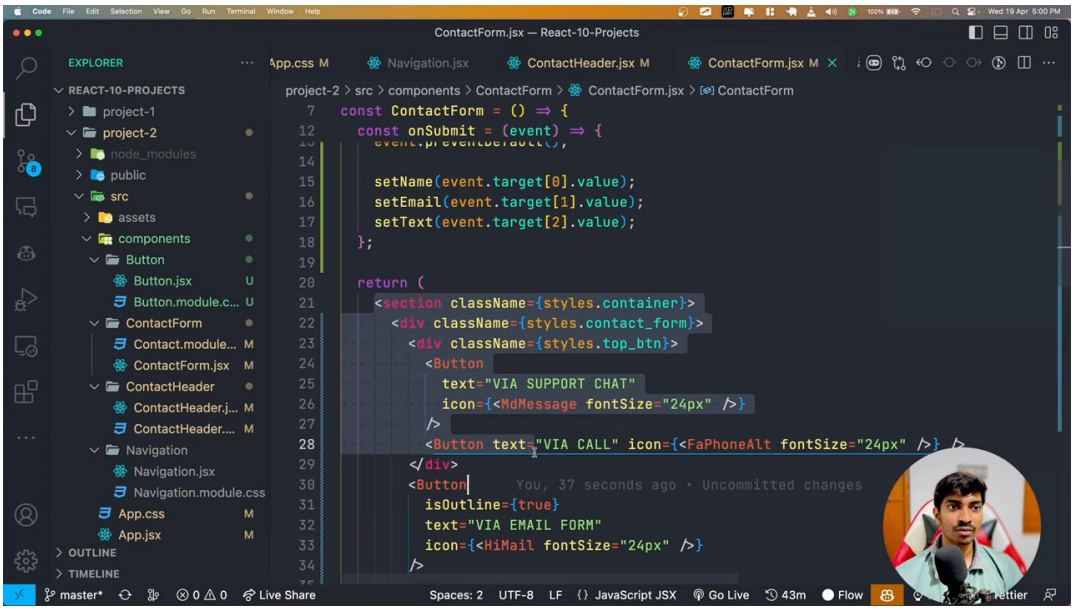
assign setName new value like this

SetName(event.target[0].value);

1:16:59

i can remove the default values

1:17:34



1:18:35

now, i want that when submit button is clicked, input fields should empty.

add the following code in the onSubmit button

// Clear input fields

```
event.target[0].value = '';

event.target[1].value = '';

event.target[2].value = '';
```

1:20:28