Project 3 - Dice Game | 10 React Projects for Beginners

Generated on February 3, 2024

Summary

Notes	Screenshots	Bookmarks
28	0	0

now second page is being started

▷ 23:45

now i will create another component GamePlay. then we will use ternary operator for logic.

logic:

```
{
      isGameStarted ?  &lt;GamePage/&gt;  : &lt;MainPage /&gt;
     }
```

▷ 25:33

now, we will pass the toggleGamePlay function in the mainPage as a prop. then write {toggle} in the argument of mainPage and add onClick={toggle} in the button tag.

since button is a saperate component, we will pass the onClick in destructuring. also, add the folowing in the button tag

onClick={onClick}

▷ 26:40

now, i want to create GamePage. i want that when user click on the play now button, GamePage component should appear where MainPage is appearing. for this, i will use a useState Hook and a function toggleGamePlay.

```
 const [isGameStarted, setisGameStarted] = useState(false);

  const toggleGamePlay = () =&gt;{

    setisGameStarted((prev) =&gt; !prev)

    console.log("play btn clicked")

  }
```

we will set the state of hook by

```
 setisGameStarted((prev) =&gt; !prev)
```

this will set the previous value to its reciprocal. we are doing it in reverse order. as you can see, the default value is false, setState will turn its value to true.

▷ 26:51

now, we will create the number boxes. we will use the following code:

```
{arrNumber.map((value, i) =&gt; (

  &lt;div key={i} className={styles.box}&gt;

    {value}

  &lt;/div&gt;
```

explaination:

- **arrNumber.map((value, i) =&gt; (**: This part initiates a **map** function on the **arrNumber** array. The **map** function iterates over each element of the **arrNumber** array and executes the provided callback function for each element.
- **(value, i)**: Inside the callback function, **value** represents the current element being processed in the array, and **i** represents the index of the current element.
- **&lt;div key={i} className={styles.box}&gt;**: For each element in the **arrNumber** array, a **&lt;div&gt;** element is created. The **key** prop is set to the index **i**, which helps React

identify each element uniquely and optimize rendering performance. The **className** prop is set to **styles.box**, which applies CSS styles defined in the **box** class from the **styles** object.

• **{value}**: Inside the **&lt;div&gt;** element, the value of the current element in the **arrNumber** array is displayed. This is the content of each **&lt;div&gt;** element.

• **))}**: This closing parenthesis and curly brace close the **map** function and the outer JSX expression.

In summary, this code dynamically creates **&lt;div&gt;** elements for each element in the **arrNumber** array. Each **&lt;div&gt;** displays the value of the corresponding element from the array, and the elements are styled using CSS styles defined in the **box** class.

▷ 37:21

to select any number by clicking, i will create a state.

const [selectedNumber, setselectedNumber] = useState(1);

▷ 39:18

whenever we want to pass any value in the function, we have to give the callback. in the above code, callback is given in onClick.

▷ 40:24

we will create the following function to set state

```
const handleClick = (value) =&gt; {
      setselectedNumber(value);
  };
```

▷ 40:28

we will use it in the div like this

&lt;div onClick={() =&gt; handleClick(value)} key={i} className={styles.box}&gt;

{value}

&lt;/div&gt;

▷ 40:28

now, we will change the color of box and number color when selected.

▷ 42:28

To achieve the desired functionality where clicking on a number changes its background color to black and the text color to white, you can conditionally apply CSS classes based on whether the number is selected. Here's how you can modify your code to accomplish this:

className={`${styles.box} ${selectedNumber === value ? styles.selected : ''}`}

explaination:

- When a number is clicked, the **handleClick** function updates the **selectedNumber** state with the clicked value.
- Each **.box** div has both the **styles.box** class and, if it's the selected number, the **styles.selected** class. This will change the background color to black and the text color to white for the selected number.
- The **onClick** event is updated to call **handleClick(value)** when a number is clicked.
- CSS is added for the **.selected** class to style the selected number differently.

▷ 43:23

i will create a saperate component dice. place dice image inside it. now i want that when i click on the dice img, they should shuffle. total dice pics are 6.

▷ 56:33

first of all, we will import a useState hook, then create a function to create a random number between 1 and 6.

const [cureentDice, setCurrentDice] = useState();

    const GenerateRandomNumber = () =&gt; Math.floor(Math.random() * 6) + 1;

▷ 57:43

1. Pass the **GenerateRandomNumber** function as a callback to the **onClick** event handler.
2. Call the **GenerateRandomNumber** function inside the **onClick** event handler.

const handleClick = () =&gt; { const randomNum = GenerateRandomNumber(); console.log(randomNum); };

▷ 1:02:20

remove the console.log and set the state

 setCurrentDice((prev) =&gt; randomNum);

(prev) =&gt; randomNum is an arrow function used as a callback. This function takes the previous state (**prev**) as an argument and returns a new state value, in this case, **randomNum**.

▷ 1:02:20

update the image address like this:

{`./images/dice_${currentDice}.png`}

${currentDice} is a placeholder that will be replaced with the value of the variable set**currentDice**.

▷ 1:02:25

now, by clicking on the dice img, images are changing

▷ 1:02:41

now, i have to check the dice number and box number and then update the total score.

▷ 1:02:54

for this, we will place the both states state in the dice component and state in the GamePlay component at the same place.

we will do this to easily handle both states.

▷ 1:03:00

we will shift the state of dice component in the GamePlay component. and pass the both values as props in the dice component, destructure in the dice compoenent.

▷ 1:05:12

now, i want that when the dice is clicked, selected number should not appear. (mean the selected button gets dark and number gets white)

▷ 1:09:44

i will add the following at the end of handleClick function

setselectedNumber(undefined);

▷ 1:09:44

now, dice is rolling even without selecting the number box.

i wan to show the error if user rolls the dice without selecting the number in box.

add if(!selectedNumber) return; at the start of handleClick function. without selecting the box, even random number will not generate.

▷ 1:10:27

actually, i want to show the error if the dice is rolled without selecting the box number.

following is the useState hook

```
    const [error, seterror] = useState("Select the Number");
```

and added this on the top of handleClick function.

```
if (!selectedNumber) {

            seterror("You have not selected any number");

            return;

        }
```

▷ 1:11:43

now, i want that when user select a number, a text should appear that says number has been selected. for this, i will create a function:

handleNumberSelection, and set the state in it, then pass it to the box div

```
const handleNumberSelection = (value) =&gt; {

        setselectedNumber(value);

        seterror("You have selected the number");

    };
```

▷ 1:12:38

i want that when error appear, its color should be red. other text color should be black.

```
className={`${error === "You have not selected any number" ? styles.errorText : styles.normalText}`}&gt;{error}&lt;/b&gt;
```

&lt;/p&gt;

1:15:21

In React's `useState` hook, when you use the functional form of the state setter function, such as `setState(prev =&gt; ...)`, `prev` represents the previous state value.

Here's how it works:

1. When you call `setState` with a function, React calls that function and passes the previous state value as an argument. This previous state value is commonly referred to as `prev`.

2. You can then use `prev` inside the function to access the previous state value.

3. You should return the new state value from this function, based on the previous state value. React will then update the state using this new value.

For example, consider the following code:

```javascript

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

// Using the functional form of setState

setCount(prev =&gt; prev + 1);

```

In this code:

- `prev` represents the previous state value of `count`.

- Inside the function, we're returning `prev + 1`, which means we're incrementing the previous state value by 1.

- React will then update the state of `count` to the new value returned from this function, effectively incrementing it by 1.

Using `prev` in this manner ensures that the state updates correctly, especially in scenarios where multiple state updates may be happening concurrently. It helps avoid issues like stale state and ensures that the updates are based on the most recent state value.

▷ 1:18:23

now, shift the GenerateRandomNumber and handleClick from dice.js to GamePlay.js. and add the following code in the function

```
if (randomNum === selectedNumber) {

    setscore(prev =&gt; prev + randomNum);

  } else {

    setscore(prev =&gt; prev - 2);

  }
```

explaination:

- We're checking if the **randomNum** generated from rolling the dice is equal to the **selectedNumber** chosen by the user.
- If they are equal, it means the user guessed correctly. In this case, we update the **score** state by adding **randomNum** to the previous value of **score**. We're using the functional form of **setState** to update the state based on the previous state value (**prev =&gt; prev + randomNum**).
- If **randomNum** is not equal to **selectedNumber**, it means the user guessed incorrectly. In this case, we subtract 2 from the **score**. Again, we're using the functional form of **setState** to update the state based on the previous state value (**prev =&gt; prev - 2**).

This logic ensures that the **score** state is updated correctly based on the user's guesses. If they guess correctly, their score increases by the value of the dice roll; if they guess incorrectly, their score decreases by 2 points.

▷ 1:18:23