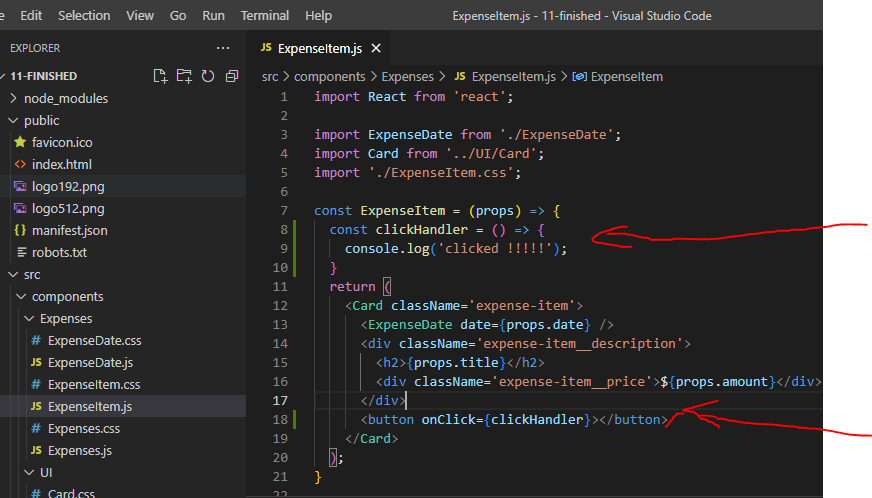
React State & Working With Events

+ Handling events

+ Updating the UI & Working with “State”

Source if you need: <https://github.com/academind/react-complete-guide-code/tree/04-react-state-events> (choose branch 03-react-basics-working-with-components and select folder 11-finished)

47. Listening to events & working with event handlers



result

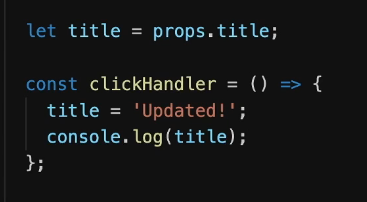


48. How Component Functions Are Executed

+It will search until last component only have html

49. Working with “State”

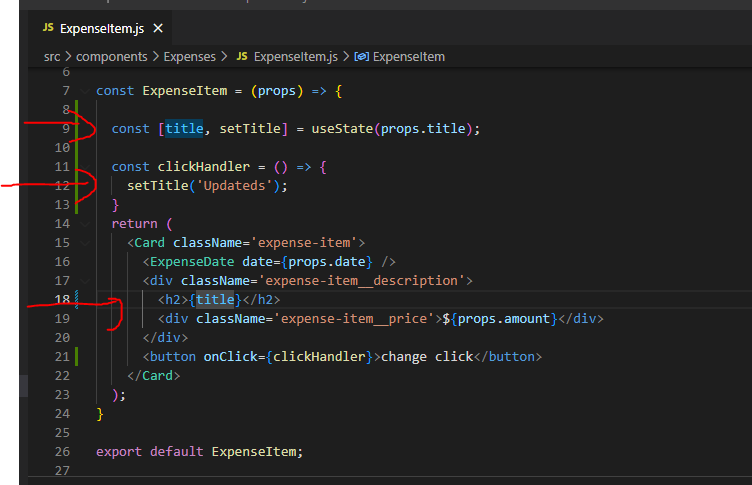
Before



* Will not working

Use userState like this

+Manage data somewhere in memory react then set value to you desired



Note: when every time you add more ExpenseItem, it will create new userState, with use different memory for save data props.title.

52. Adding Form Input

52.1 Event.preventDefault(), Event.stopPropagation(), Event.stopImmediatePropagation()

* **preventDefault**: Huỷ bỏ event nếu nó có thể huỷ mà không dừng sự lan rộng(propagation) của event tới phần khác.
* **stopPropagation** Ngăn chặn sự lan rộng của sự kiện hiện tại tới thằng khác.
* **stopImmediatePropagation** ngăn chặn những listeners cũng đang đang lắng nghe cùng event được gọi.

**Detail**

**Event.preventDefault**

Hãy nhìn vào mãi ví dụ ở dưới. Chúng ta sẽ thấy việc click vào submit button trên form sẽ gửi nội dung của form tới chỗ xử lý. Event.preventDefault là cách hoàn hảo để không cho gửi form khi nhấn vào button submit.

<form id="myForm" action="/my-handling-form-page" method="post">

<div>

<label for="name">Name:</label>

<input type="text" id="name" />

</div>

<div>

<label for="mail">E-mail:</label>

<input type="email" id="mail" />

</div>

<div>

<label for="msg">Message:</label>

<textarea id="msg"></textarea>

</div>

<div class="button">

<button type="submit">Send your message</button>

</div>

</form>

$('#myForm').on('submit', function(e) {

e.preventDefault(); // Now nothing will happen

});

Event.preventDefault sẽ đảm bảo rằng form không bao giờ được gửi, và nó đã giành được quyền kiểm soát và ngăn chặn sự kiện đó khi click. Đó là những gì chúng ta đã làm.

**Event.stopPropagation**

stopPropagation để đảm bảo chắc chắn rằng event không lan rộng nữa. Hãy xem ví dụ ở dưới:

<div class="container">

<a href="#" class="element">Click Me!</a>

</div>

$('.container').on('click', function(e) {

console.log('container was clicked');

});

$('.element').on('click', function(e) {

e.preventDefault(); // Now link won't go anywhere

console.log('element was clicked');

});

Bây giờ nếu bạn click link và trước đó đã mở console bạn sẽ thấy:

"element was clicked"

"container was clicked"

Bây giờ hãy thêm Event.stopPropagation:

$('.container').on('click', function(e) {

console.log('container was clicked');

});

$('.element').on('click', function(e) {

e.preventDefault(); // Now link won't go anywhere

e.stopPropagation(); // Now the event won't bubble up

console.log('element was clicked');

});

Và click lại. Đây sẽ là cái bạn thấy:

"element was clicked"

**Event.stopImmediatePropagation**

Với 2 methods ở trên đã giúp bạn khoảng 90% các trường hợp cần thiết phải xử lí với events. Nhưng hãy tìm hiểu tiếp method cuối này nhé

Chúng ta sẽ bắt đầu với markup tương tự ở trên, và thêm vào một class. Một cái là item mà tất cả các anchors sẽ nhận, và một cái riêng chỉ cho markup ở đây. Nó rất quan trọng để giúp code ví dụ này hoặt động.

<div class="container">

<a href="#" class="item element">Click Me!</a>

</div>

Và chúng sẽ thêm Event.stopPropagation mothod mà chúng ta đã tìm hiểu ở phần trước.

$('.item').on('click', function(e) {

console.log('an item was clicked');

});

$('.element').on('click', function(e) {

e.preventDefault(); // Now link won't go anywhere

e.stopPropagation(); // Now the event won't bubble up

console.log('element was clicked');

});

Bây giờ hãy hãy xem có gì ở console khi click nhé.

"an item was clicked"

"element was clicked"

Vấn đề ở đây là .item và .element được xếp đồng hạng trên DOM. Nó không lan tràn lên phần tử trên như ví dụ trước đó. Và vì khi click cả 2 action gắn trên .element và .item sẽ chạy cùng lúc, bạn không thể dừng lan rộng(propagation) như bạn mong muốn.

Đây chính là cơ hội để xử dụng stopImmediatePropagation!

$('.element').on('click', function(e) {

e.preventDefault(); // Now link won't go anywhere

e.stopImmediatePropagation(); // Now item on click won't fire

console.log('element was clicked');

});

$('.item').on('click', function(e) {

console.log('an item was clicked');

});

Điêu quan trọng ở đây để ngăn chặn sự lan rộng tới cùng hạng của event trong DOM là đặt stopImmediatePropagation ở ngay khai báo lệnh đầu tiên về action lắng nghe event click trong code của bạn.

52 Adding Form Input

Step 1 create new NewExpense/ExpenseForm.js

import React, { useState } from 'react';

import './ExpenseForm.css';

const ExpenseForm = () => {

  const [enteredTitle, setEnteredTitle] = useState('');

  const [enteredAmount, setEnteredAmount] = useState('');

  const [enteredDate, setEnteredDate] = useState('');

  // const [userInput, setUserInput] = useState({

  //   enteredTitle: '',

  //   enteredAmount: '',

  //   enteredDate: '',

  // });

  const titleChangeHandler = (event) => {

    setEnteredTitle(event.target.value);

    // setUserInput({

    //   ...userInput,

    //   enteredTitle: event.target.value,

    // });

    // setUserInput((prevState) => {

    //   return { ...prevState, enteredTitle: event.target.value };

    // });

  };

  const amountChangeHandler = (event) => {

    setEnteredAmount(event.target.value);

    // setUserInput({

    //   ...userInput,

    //   enteredAmount: event.target.value,

    // });

  };

  const dateChangeHandler = (event) => {

    setEnteredDate(event.target.value);

    // setUserInput({

    //   ...userInput,

    //   enteredDate: event.target.value,

    // });

  };

  return (

    <form>

      <div className='new-expense\_\_controls'>

        <div className='new-expense\_\_control'>

          <label>Title</label>

          <input type='text' onChange={titleChangeHandler} />

        </div>

        <div className='new-expense\_\_control'>

          <label>Amount</label>

          <input

            type='number'

            min='0.01'

            step='0.01'

            onChange={amountChangeHandler}

          />

        </div>

        <div className='new-expense\_\_control'>

          <label>Date</label>

          <input

            type='date'

            min='2019-01-01'

            max='2022-12-31'

            onChange={dateChangeHandler}

          />

        </div>

      </div>

      <div className='new-expense\_\_actions'>

        <button type='submit'>Add Expense</button>

      </div>

    </form>

  );

};

export default ExpenseForm;

Step 2 add NewExpense/ExpenseForm.css

.new-expense\_\_controls {

  display: flex;

  flex-wrap: wrap;

  gap: 1rem;

  margin-bottom: 1rem;

  text-align: left;

}

.new-expense\_\_control label {

  font-weight: bold;

  margin-bottom: 0.5rem;

  display: block;

}

.new-expense\_\_control input {

  font: inherit;

  padding: 0.5rem;

  border-radius: 6px;

  border: 1px solid #ccc;

  width: 20rem;

  max-width: 100%;

}

.new-expense\_\_actions {

  text-align: right;

}

Step 3 add NewExpense/NewExpense.js

import React from 'react';

import ExpenseForm from './ExpenseForm';

import './NewExpense.css';

const NewExpense = () => {

  return (

    <div className='new-expense'>

      <ExpenseForm />

    </div>

  );

};

export default NewExpense;

Step 4 add NewExpense/NewExpense.css

.new-expense {

  background-color: #a892ee;

  padding: 1rem;

  margin: 2rem auto;

  width: 50rem;

  max-width: 95%;

  border-radius: 12px;

  text-align: center;

  box-shadow: 0 1px 8px rgba(0, 0, 0, 0.25);

}

.new-expense button {

  font: inherit;

  cursor: pointer;

  padding: 1rem 2rem;

  border: 1px solid #40005d;

  background-color: #40005d;

  color: white;

  border-radius: 12px;

  margin-right: 1rem;

}

.new-expense button:hover,

.new-expense button:active {

  background-color: #510674;

  border-color: #510674;

}

.new-expense button.alternative {

  color: #220131;

  border-color: transparent;

  background-color: transparent;

}

.new-expense button.alternative:hover,

.new-expense button.alternative:active {

  background-color: #ddb3f8;

}

Step 5 then in App.js

import React from 'react';

import NewExpense from './components/NewExpense/NewExpense';

import Expenses from './components/Expenses/Expenses';

const App = () => {

  const expenses = [

    {

      id: 'e1',

      title: 'Toilet Paper',

      amount: 94.12,

      date: new Date(2020, 7, 14),

    },

    { id: 'e2', title: 'New TV', amount: 799.49, date: new Date(2021, 2, 12) },

    {

      id: 'e3',

      title: 'Car Insurance',

      amount: 294.67,

      date: new Date(2021, 2, 28),

    },

    {

      id: 'e4',

      title: 'New Desk (Wooden)',

      amount: 450,

      date: new Date(2021, 5, 12),

    },

  ];

  // return React.createElement(

  //   'div',

  //   {},

  //   React.createElement('h2', {}, "Let's get started!"),

  //   React.createElement(Expenses, { items: expenses })

  // );

  return (

    <div>

      <NewExpense />

      <Expenses items={expenses} />

    </div>

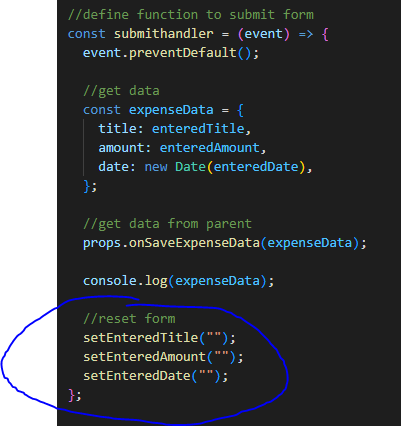
  );

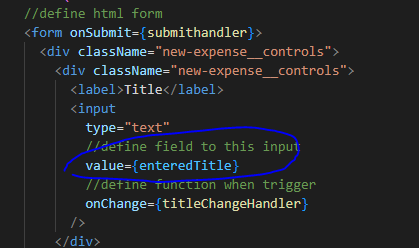
}

export default App;

58. Adding Two-Way Binding

Reset form when submit, bind between input by parameter “value” and useState





Step 1 in ExpenseForm.js change like this

import React, { useState } from 'react';

import './ExpenseForm.css';

const ExpenseForm = () => {

  const [enteredTitle, setEnteredTitle] = useState('');

  const [enteredAmount, setEnteredAmount] = useState('');

  const [enteredDate, setEnteredDate] = useState('');

  // const [userInput, setUserInput] = useState({

  //   enteredTitle: '',

  //   enteredAmount: '',

  //   enteredDate: '',

  // });

  const titleChangeHandler = (event) => {

    setEnteredTitle(event.target.value);

    // setUserInput({

    //   ...userInput,

    //   enteredTitle: event.target.value,

    // });

    // setUserInput((prevState) => {

    //   return { ...prevState, enteredTitle: event.target.value };

    // });

  };

  const amountChangeHandler = (event) => {

    setEnteredAmount(event.target.value);

    // setUserInput({

    //   ...userInput,

    //   enteredAmount: event.target.value,

    // });

  };

  const dateChangeHandler = (event) => {

    setEnteredDate(event.target.value);

    // setUserInput({

    //   ...userInput,

    //   enteredDate: event.target.value,

    // });

  };

  const submitHandler = (event) => {

    event.preventDefault();

    const expenseData = {

      title: enteredTitle,

      amount: enteredAmount,

      date: new Date(enteredDate),

    };

    console.log(expenseData);

    setEnteredTitle('');

    setEnteredAmount('');

    setEnteredDate('');

  };

  return (

    <form onSubmit={submitHandler}>

      <div className='new-expense\_\_controls'>

        <div className='new-expense\_\_control'>

          <label>Title</label>

          <input

            type='text'

            value={enteredTitle}

            onChange={titleChangeHandler}

          />

        </div>

        <div className='new-expense\_\_control'>

          <label>Amount</label>

          <input

            type='number'

            min='0.01'

            step='0.01'

            value={enteredAmount}

            onChange={amountChangeHandler}

          />

        </div>

        <div className='new-expense\_\_control'>

          <label>Date</label>

          <input

            type='date'

            min='2019-01-01'

            max='2022-12-31'

            value={enteredDate}

            onChange={dateChangeHandler}

          />

        </div>

      </div>

      <div className='new-expense\_\_actions'>

        <button type='submit'>Add Expense</button>

      </div>

    </form>

  );

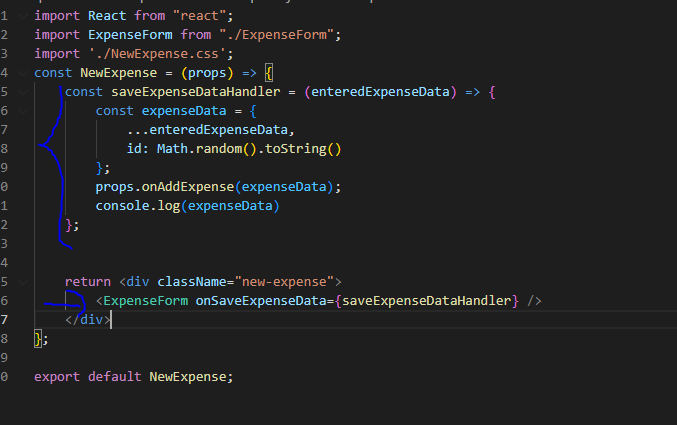
};

export default ExpenseForm;

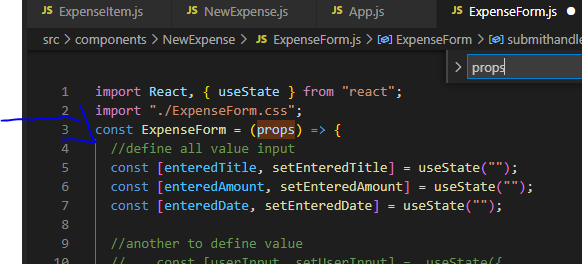
59. Child-To-Parent Component Communications (Bottom-up)

Pass data from Child to Parent

In Parent



Then in child just use props to call that function



Text

Description automatically generated

Step 1 in New Expense.js

import React from 'react';

import ExpenseForm from './ExpenseForm';

import './NewExpense.css';

const NewExpense = (props) => {

  const saveExpenseDataHandler = (enteredExpenseData) => {

    const expenseData = {

      ...enteredExpenseData,

      id: Math.random().toString()

    };

    props.onAddExpense(expenseData);

  };

  return (

    <div className='new-expense'>

      <ExpenseForm onSaveExpenseData={saveExpenseDataHandler} />

    </div>

  );

};

export default NewExpense;

Step 2 in ExpenseForm.js

import React, { useState } from 'react';

import './ExpenseForm.css';

const ExpenseForm = (props) => {

  const [enteredTitle, setEnteredTitle] = useState('');

  const [enteredAmount, setEnteredAmount] = useState('');

  const [enteredDate, setEnteredDate] = useState('');

  // const [userInput, setUserInput] = useState({

  //   enteredTitle: '',

  //   enteredAmount: '',

  //   enteredDate: '',

  // });

  const titleChangeHandler = (event) => {

    setEnteredTitle(event.target.value);

    // setUserInput({

    //   ...userInput,

    //   enteredTitle: event.target.value,

    // });

    // setUserInput((prevState) => {

    //   return { ...prevState, enteredTitle: event.target.value };

    // });

  };

  const amountChangeHandler = (event) => {

    setEnteredAmount(event.target.value);

    // setUserInput({

    //   ...userInput,

    //   enteredAmount: event.target.value,

    // });

  };

  const dateChangeHandler = (event) => {

    setEnteredDate(event.target.value);

    // setUserInput({

    //   ...userInput,

    //   enteredDate: event.target.value,

    // });

  };

  const submitHandler = (event) => {

    event.preventDefault();

    const expenseData = {

      title: enteredTitle,

      amount: enteredAmount,

      date: new Date(enteredDate),

    };

    props.onSaveExpenseData(expenseData);

    setEnteredTitle('');

    setEnteredAmount('');

    setEnteredDate('');

  };

  return (

    <form onSubmit={submitHandler}>

      <div className='new-expense\_\_controls'>

        <div className='new-expense\_\_control'>

          <label>Title</label>

          <input

            type='text'

            value={enteredTitle}

            onChange={titleChangeHandler}

          />

        </div>

        <div className='new-expense\_\_control'>

          <label>Amount</label>

          <input

            type='number'

            min='0.01'

            step='0.01'

            value={enteredAmount}

            onChange={amountChangeHandler}

          />

        </div>

        <div className='new-expense\_\_control'>

          <label>Date</label>

          <input

            type='date'

            min='2019-01-01'

            max='2022-12-31'

            value={enteredDate}

            onChange={dateChangeHandler}

          />

        </div>

      </div>

      <div className='new-expense\_\_actions'>

        <button type='submit'>Add Expense</button>

      </div>

    </form>

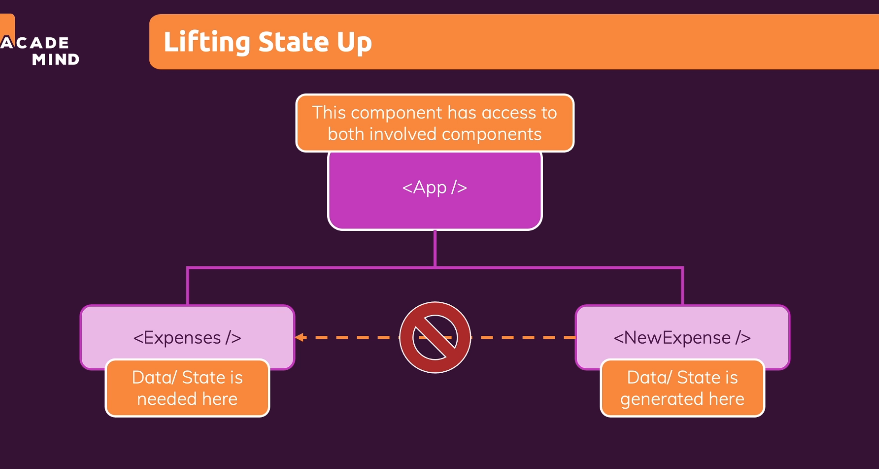
  );

};

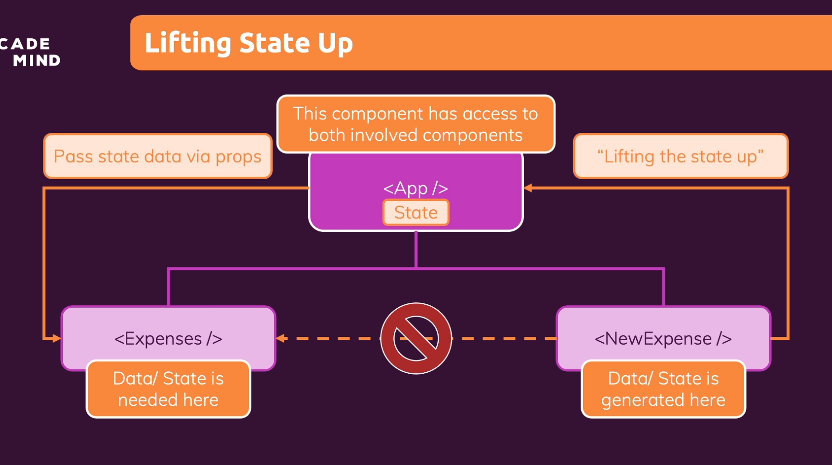
export default ExpenseForm;

60. Lifting the state up (about 59 Child-To-Parent)

Problem

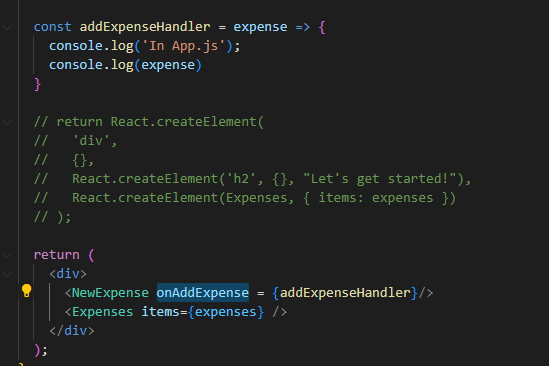


Solved

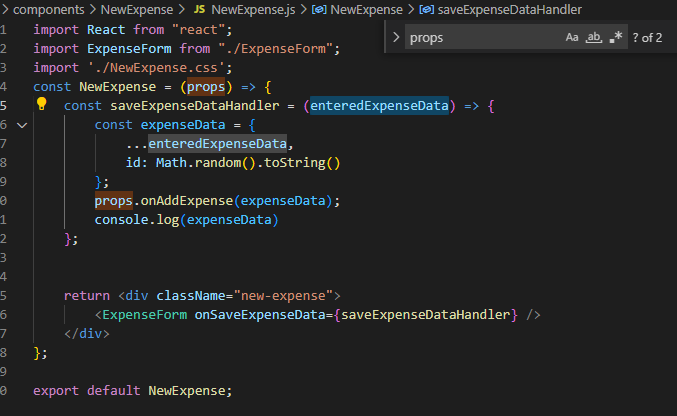


🡺 App want list data (expenseForm will load data then tranfer to NewExpense then tranfer to app)

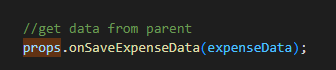
App



NewExpense



ExpenseForm



61. Controlled and Uncontrolled components & stateless and stateful component

Mean some data only print data but not manage by submit or anything else

62. Module about input form

<https://github.com/academind/react-complete-guide-code/tree/04-react-state-events>

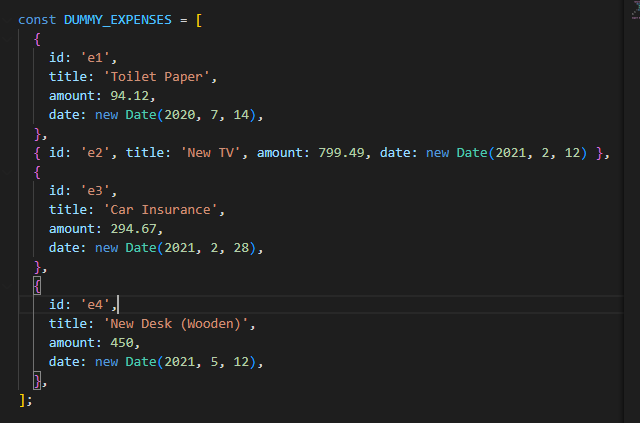
Section 5: Rendering Lists & Conditional Content

(Working with dynamic content)

64. rendering Lists of data

Need checkout branch 05-rendering-lists-conditional-content with folder 01-using-stateful-lists

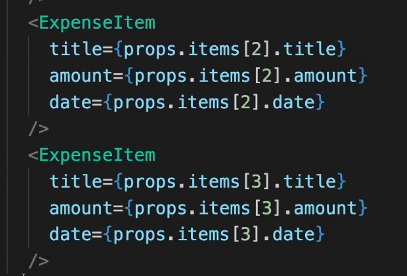
Step 1 in App.js (inparent)



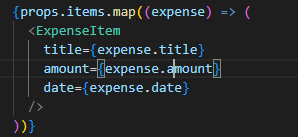


Step 2 in Expenses.js

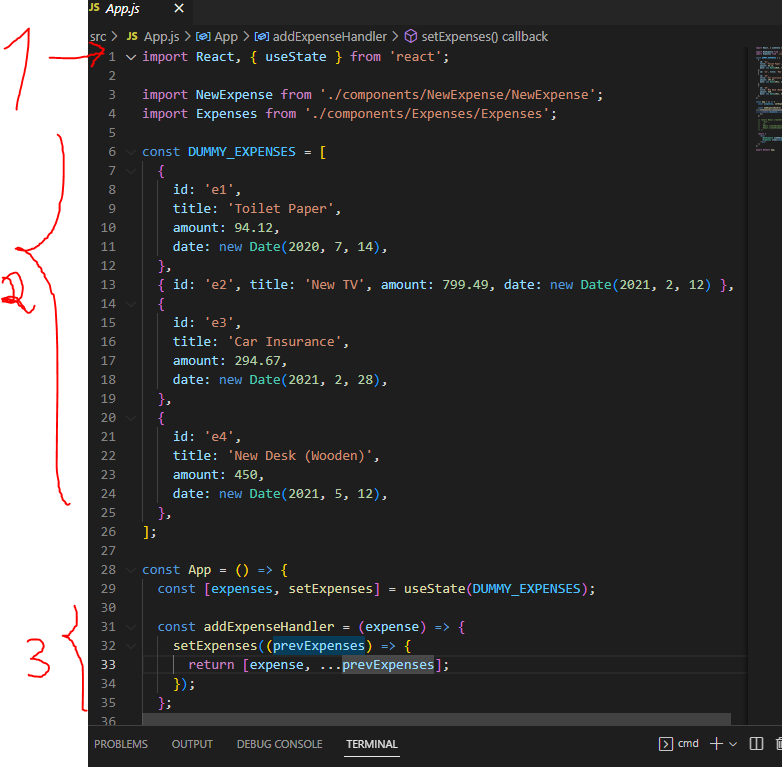
Before with hard code number

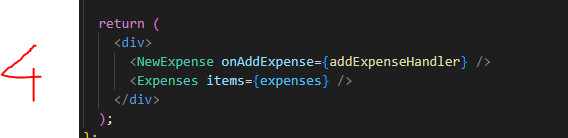


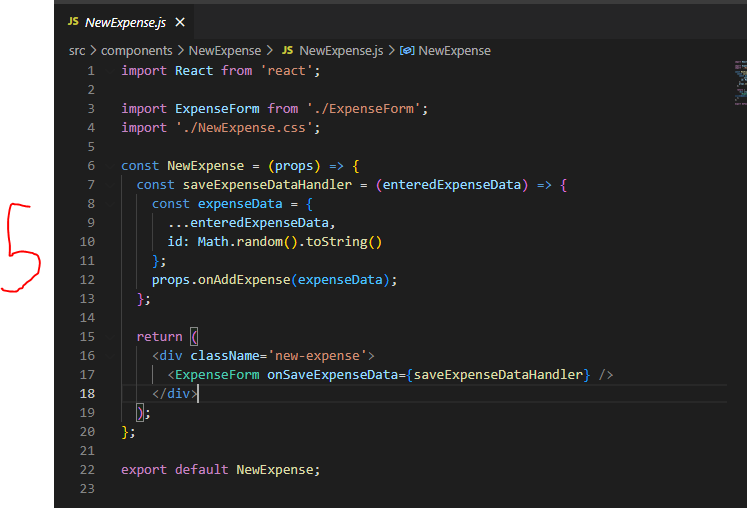
After



65. Using stateful lists







66.Understanding ‘keys’

Chính xác thì có ảnh hưởng gì của mảng mà ReactJS cần có key props?

Giả sử, chúng ta render một mảng mà không thêm key props:

<li>Devnote 1</li>

<li>Devnote 2</li>

Sau đó, Devnote 3 được thêm vào. Lúc này:

<li>Devnote 1</li>

<li>Devnote 2</li>

<li>Devnote 3</li>

ReactJS bắt đầu so sánh 2 trees này để tìm ra điểm khác biệt. Để thực hiện điều này, nó sẽ cùng-lặp-qua-lần lượt-tất-cả các phần tử con của cả 2 mảng; generate ra chỗ cần cập nhật mỗi khi nhận ra được điểm nào đó khác nhau.

Như vậy thì ở ví dụ trên, phần tử đầu tiên và phần tử thứ 2, okie, giống nhau, phần tử thứ 3 là chỗ được thay đổi, cập nhật thôi ! Nom có vẻ ngon lành cành đào nhỉ 😸😸

Bây giờ, chúng ta lại thêm Devnote 0 vào trước mảng đó:

<li>Devnote 0</li>

<li>Devnote 1</li>

<li>Devnote 2</li>

<li>Devnote 3</li>

Một lần nữa, ReactJS lại lặp, lại so sánh:

* Phần tử đầu của old tree(*<li>Devnote 1</li>*) với phần tử đầu của new tree(*<li>Devnote 0</li>*) ⇒ khác nhau ⇒ cập nhật.
* Phần tử thứ 2 của old tree(*<li>Devnote 2</li>*) với phần tử thứ 2 của new tree(*<li>Devnote 1</li>*) ⇒ khác nhau ⇒ cập nhật.
* ...

Ồ, có vẻ như có gì đó phát sinh rồi đây !

Cứ vậy thì ReactJS sẽ cập nhật lại hết tất cả các phần tử con thay vì nhận ra được *<li>Devnote 1</li>, <li>Devnote 2</li>, <li>Devnote 3</li>* không thay đổi. Điều này sẽ ảnh hưởng tới performance của ứng dụng.

Lúc này, key props sinh ra cho đời bớt khổ (J4F).

Theo *Trang chủ ReactJS*:

Keys help React identify which items have changed, are added, or are removed. Keys should be given to the elements a stable identity.

Như vậy, khi chúng ta sửa lại:

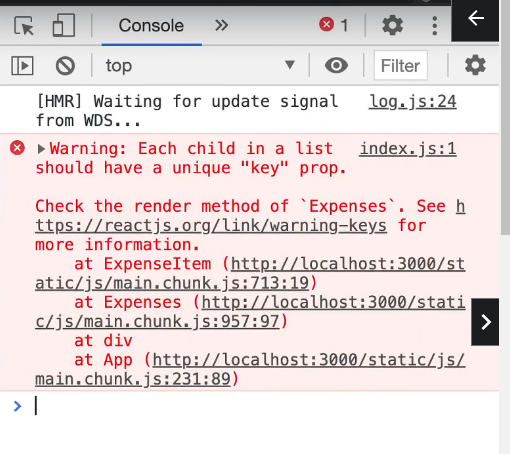
<li key={ 0 }>Devnote 0</li>

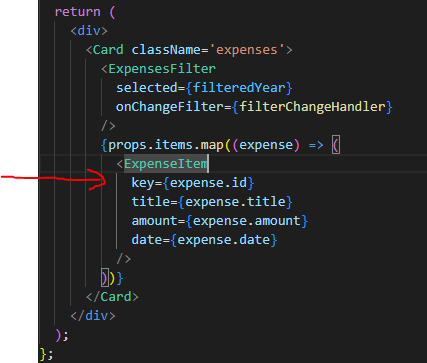
<li key={ 1 }>Devnote 1</li>

<li key={ 2 }>Devnote 2</li>

<li key={ 3 }>Devnote 3</li>

* ReactJS sẽ sử dụng key này trong quá trình reconciliation, so sánh tree cũ và tree mới thông qua key của từng phần tử, và kết quả là ReactJS sẽ nhận ra được phần tử mới là <li key={ 0 }>Devnote 0</li> được thêm vào phía trước, các phần tử còn lại chỉ là dịch xuống vị trí kế tiếp mà thôi.





Then in ExpenseItem.js

import React from 'react';

import ExpenseDate from './ExpenseDate';

import Card from '../UI/Card';

import './ExpenseItem.css';

const ExpenseItem = (props) => {

  return (

    <Card className='expense-item'>

      <ExpenseDate date={props.date} />

      <div className='expense-item\_\_description'>

        <h2>{props.title}</h2>

        <div className='expense-item\_\_price'>${props.amount}</div>

      </div>

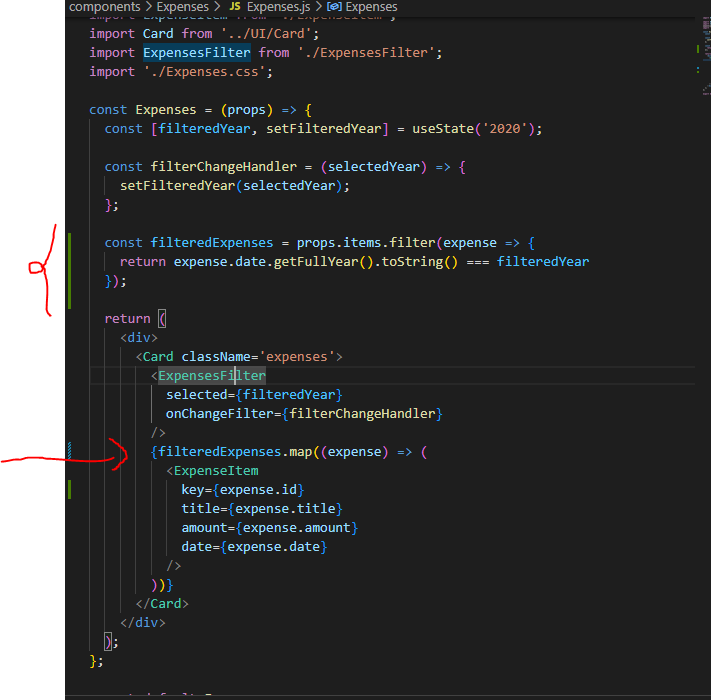
    </Card>

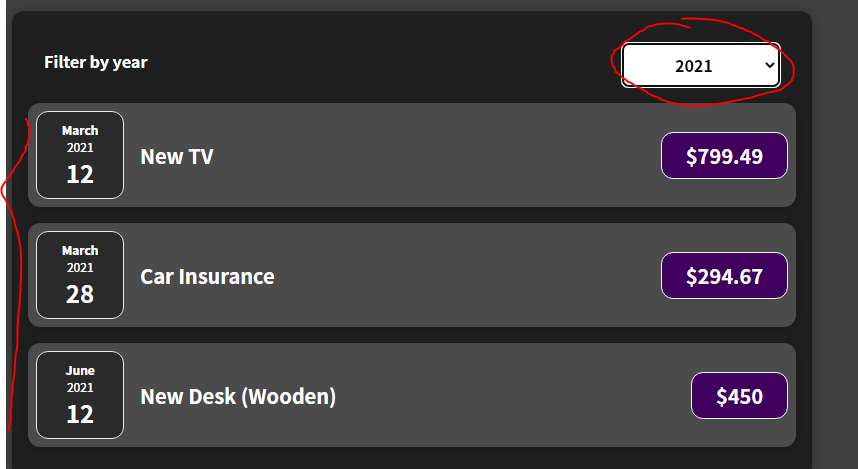
  );

}

export default ExpenseItem;

66.2 Filter year





68. Adding Conditional Return Statements

Create another ExpenseList like this

import React from "react";

import ExpenseItem from "./ExpenseItem";

import './ExpensesList.css';

const ExpensesList = props => {

    if(props.items.length === 0){

     return <h2 className="expenses-list\_\_fallback">Found no expenses</h2>;

    }

    return <ul className="expenses-list">

        {props.items.map((expense) => (

          <ExpenseItem

            key={expense.id}

            title={expense.title}

            amount={expense.amount}

            date={expense.date}

          />

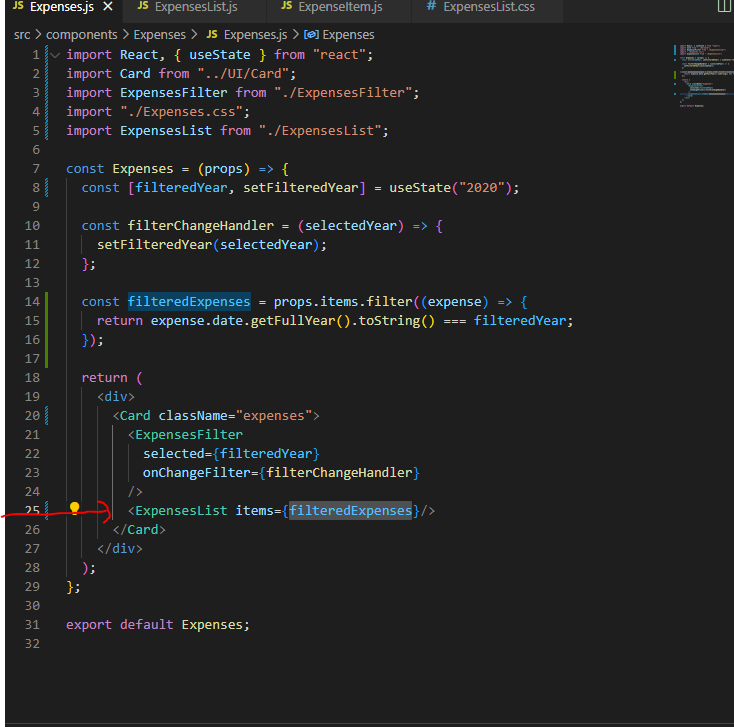
        ))}

    </ul>

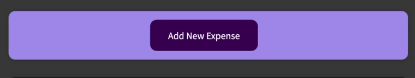
};

export default ExpensesList;

Then in Expense.js



Assignment 4 add new Expense



Solved

In NewExpense.js

import React, { useState } from "react";

import ExpenseForm from "./ExpenseForm";

import "./NewExpense.css";

const NewExpense = (props) => {

  const [isEditing, setIsEditing] = useState(false);

  const saveExpenseDataHandler = (enteredExpenseData) => {

    const expenseData = {

      ...enteredExpenseData,

      id: Math.random().toString(),

    };

    props.onAddExpense(expenseData);

    setIsEditing(false);

  };

  const startEditingHandler = () => {

    setIsEditing(true);

  };

  const stopEditingHandler = () => {

    setIsEditing(false);

  };

  return (

    <div className="new-expense">

      {!isEditing && (

        <button onClick={startEditingHandler}>Add New Expense</button>

      )}

      {isEditing && (

        <ExpenseForm

          onSaveExpenseData={saveExpenseDataHandler}

          onCancel={stopEditingHandler}

        />

      )}

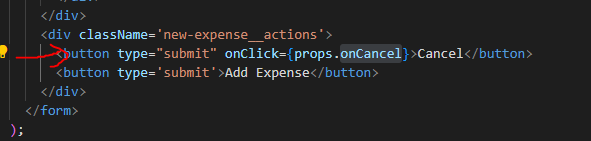
    </div>

  );

};

export default NewExpense;

then in ExpenseForm.js



69. Demo App: Adding a Chart

Section 6. Styling React Components

(clone source branch 06 first)

75. Setting Dynamic Inline Style

Set red when input is empty



Step 1 Go to CourseInput.js and change like this

import React, { useState } from "react";

import Button from "../../UI/Button/Button";

import "./CourseInput.css";

const CourseInput = (props) => {

  const [enteredValue, setEnteredValue] = useState("");

  const [isValid, setIsValid] = useState(true);

  const goalInputChangeHandler = (event) => {

    if(event.target.value.trim().length > 0){

      setIsValid(true);

    }

    setEnteredValue(event.target.value);

  };

  const formSubmitHandler = (event) => {

    event.preventDefault();

    if (enteredValue.trim().length === 0) {

      setIsValid(false);

      return;

    }

    props.onAddGoal(enteredValue);

  };

  return (

    <form onSubmit={formSubmitHandler}>

      <div className="form-control">

        <label style={{ color: !isValid ? "red" : "black" }}>Course Goal</label>

        <input

          style={{

            borderColor: !isValid ? "red" : "black",

            background: !isValid ? "salmon" : "transparent",

          }}

          type="text"

          onChange={goalInputChangeHandler}

        />

      </div>

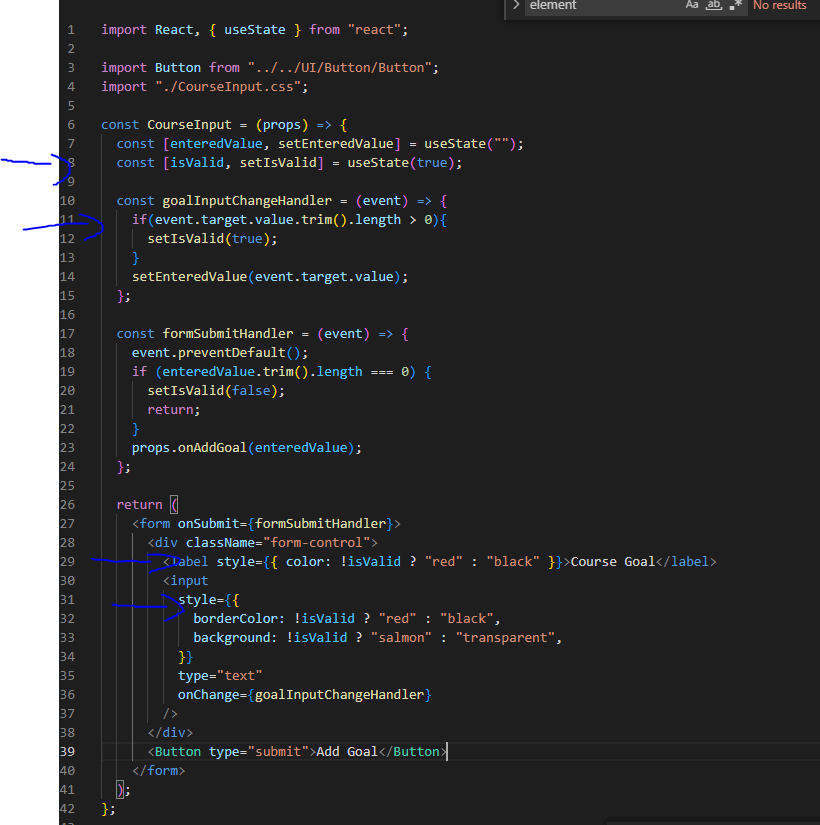
      <Button type="submit">Add Goal</Button>

    </form>

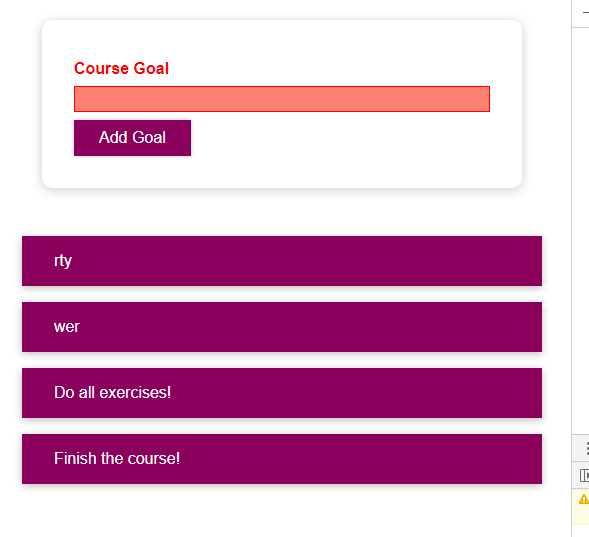
  );

};

export default CourseInput;



Result



* Note: bad code, because hard code, when color red need change, all other element also need edit (hard to maintain)

Step 1 Go to CourseInput.css add new css like this

.form-control.invalid input {

  border-color: red;

  background: #ffd7d7;

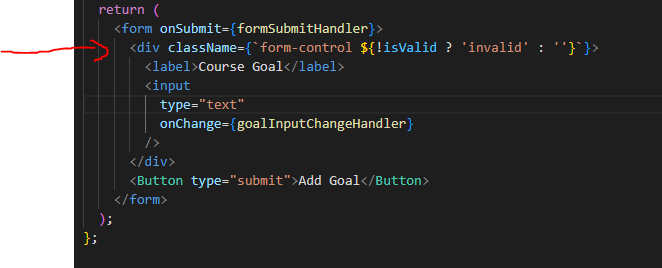
}

.form-control.invalid label {

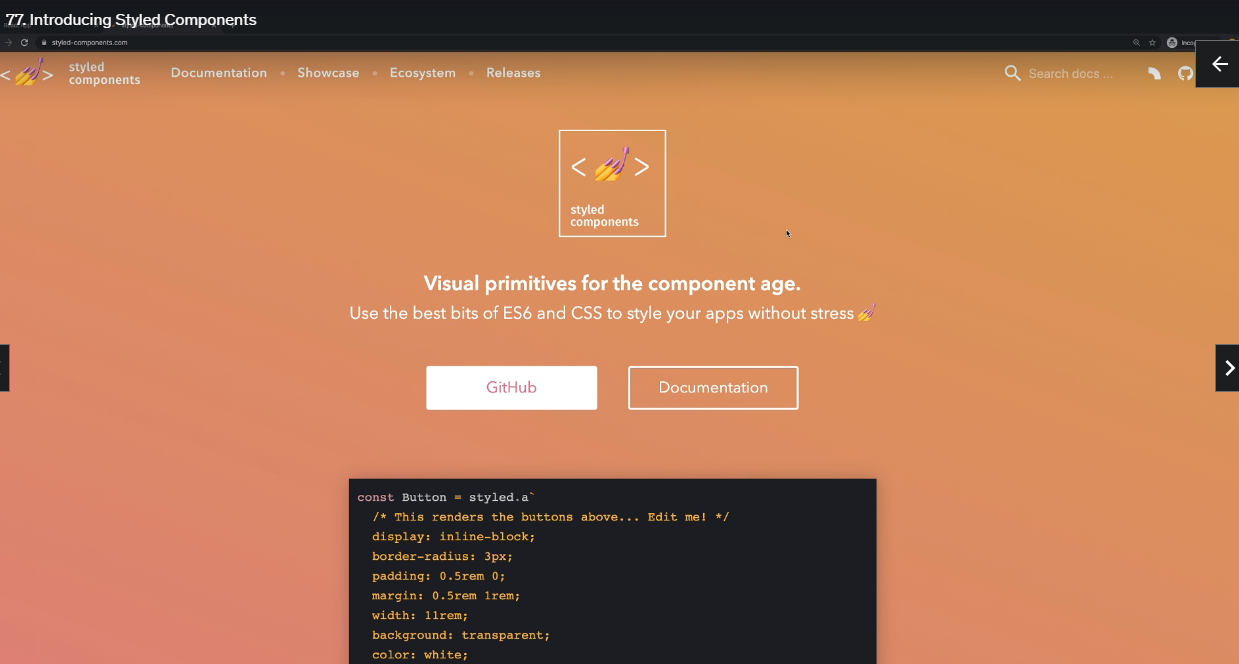
  color: red;

}

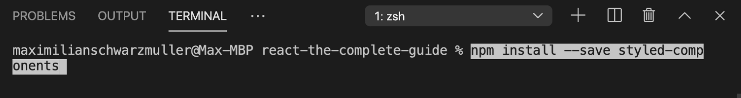
Then in CourseInput.js change like this



77. Introducing Styled Components

[Styled-Components](https://www.styled-components.com/) là một thư viện giúp bạn có thể tổ chức và quản lý code CSS một cách dễ dàng trong các project React. Nó được xây dựng với mục tiêu giữ cho các styles của các components trong React gắn liền với chính các components đó 

Run



Copy all Button.css and paste Button.js like this

import React from 'react';

import './Button.css';

import styled from 'styled-components';

const Button = styled.button`

    font: inherit;

    padding: 0.5rem 1.5rem;

    border: 1px solid #8b005d;

    color: white;

    background: #8b005d;

    box-shadow: 0 0 4px rgba(0, 0, 0, 0.26);

    cursor: pointer;

  .&:focus {

    outline: none;

  }

  .&:hover,

  .&:active {

    background: #ac0e77;

    border-color: #ac0e77;

    box-shadow: 0 0 8px rgba(0, 0, 0, 0.26);

  }

`;

// const Button = props => {

//   return (

//     <button type={props.type} className="button" onClick={props.onClick}>

//       {props.children}

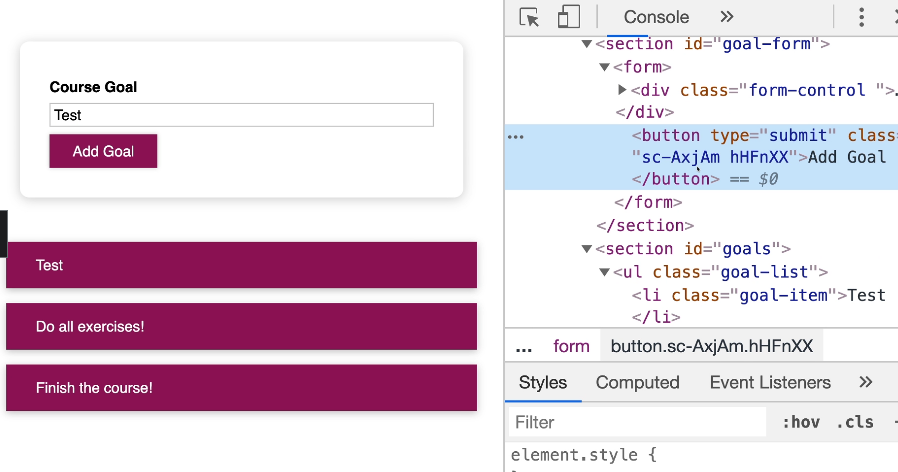
//     </button>

//   );

// };

export default Button;

Result



* You will Notice class different in style components

78. Styled components & dynamic Props

Use Props in Style Components

In CourseInput.js change like this

import React, { useState } from "react";

import styled from "styled-components";

import Button from "../../UI/Button/Button";

import "./CourseInput.css";

const FormControl = styled.div`

  margin: 0.5rem 0;

& label {

  font-weight: bold;

  display: block;

  margin-bottom: 0.5rem;

  color: ${props => props.invalid ? 'red' : 'black'}

}

& input {

  display: block;

  width: 100%;

  border: 1px solid ${props => (props.invalid ? 'red' :'#ccc')};

  background: ${props => props.invalid ? '#ffd7d7' : 'transparent'}

  font: inherit;

  line-height: 1.5rem;

  padding: 0 0.25rem;

}

& input:focus {

  outline: none;

  background: #fad0ec;

  border-color: #8b005d;

}

`

const CourseInput = (props) => {

  const [enteredValue, setEnteredValue] = useState("");

  const [isValid, setIsValid] = useState(true);

  const goalInputChangeHandler = (event) => {

    if(event.target.value.trim().length > 0){

      setIsValid(true);

    }

    setEnteredValue(event.target.value);

  };

  const formSubmitHandler = (event) => {

    event.preventDefault();

    if (enteredValue.trim().length === 0) {

      setIsValid(false);

      return;

    }

    props.onAddGoal(enteredValue);

  };

  return (

    <form onSubmit={formSubmitHandler}>

      <FormControl invalid={!isValid}>

        <label>Course Goal</label>

        <input

          type="text"

          onChange={goalInputChangeHandler}

        />

      </FormControl>

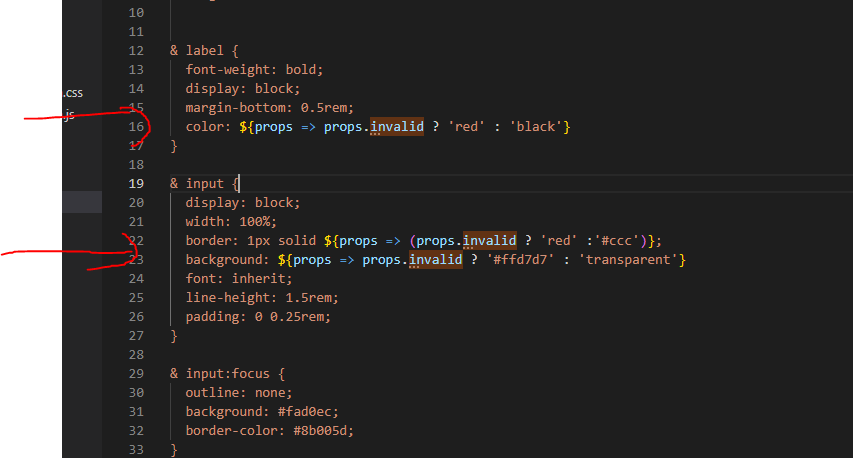
      <Button type="submit">Add Goal</Button>

    </form>

  );

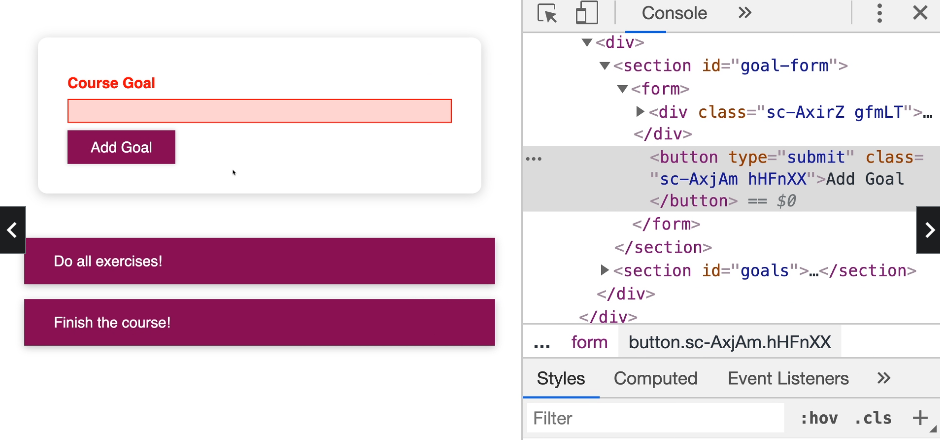
};

export default CourseInput;



Result

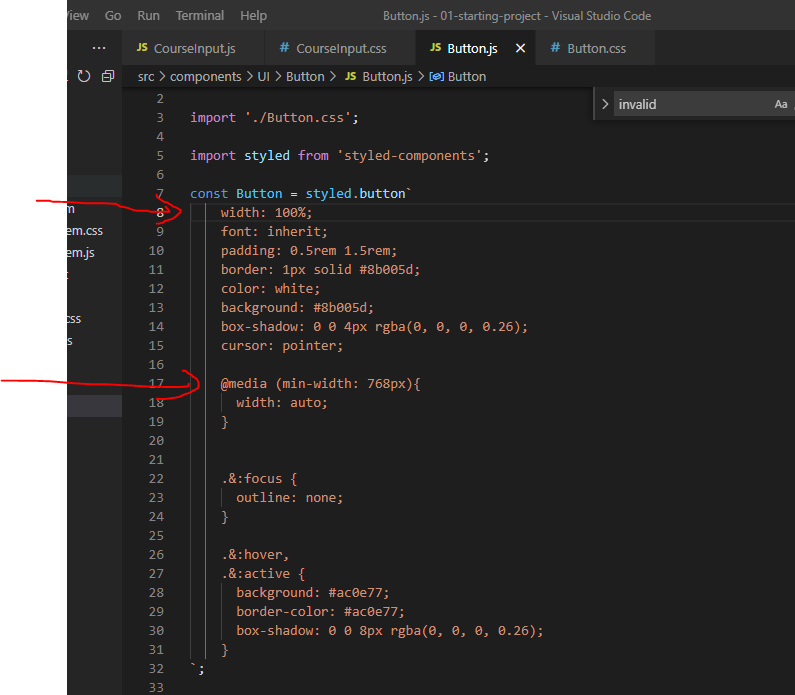
You will notice different class in button



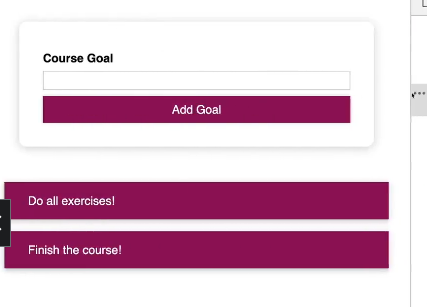
79. Styled Components & Media Queries

You want button full px when in mobile or ipad

Step 1 in Button.js Fix like this

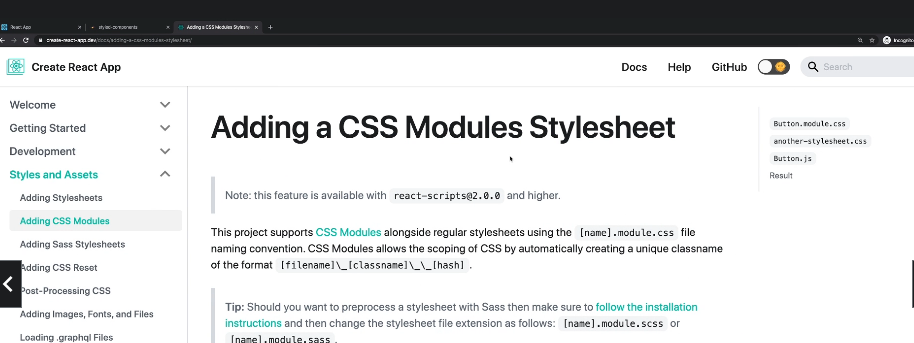


Result



80. Using CSS Moules

Make sure css only affect to js have import from that module



Step 1 Change Button.css to Button.module

Step 2 uncomment button.module.css

.button {

  font: inherit;

  padding: 0.5rem 1.5rem;

  border: 1px solid #8b005d;

  color: white;

  background: #8b005d;

  box-shadow: 0 0 4px rgba(0, 0, 0, 0.26);

  cursor: pointer;

}

.button:focus {

  outline: none;

}

.button:hover,

.button:active {

  background: #ac0e77;

  border-color: #ac0e77;

  box-shadow: 0 0 8px rgba(0, 0, 0, 0.26);

}

Step 3 comment sytle and paster like this

import React from 'react';

import styles from './Button.module.css';

// import styled from 'styled-components';

// const Button = styled.button`

//     width: 100%;

//     font: inherit;

//     padding: 0.5rem 1.5rem;

//     border: 1px solid #8b005d;

//     color: white;

//     background: #8b005d;

//     box-shadow: 0 0 4px rgba(0, 0, 0, 0.26);

//     cursor: pointer;

//     @media (min-width: 768px){

//       width: auto;

//     }

//     .&:focus {

//       outline: none;

//     }

//     .&:hover,

//     .&:active {

//       background: #ac0e77;

//       border-color: #ac0e77;

//       box-shadow: 0 0 8px rgba(0, 0, 0, 0.26);

//     }

// `;

const Button = props => {

  return (

    <button type={props.type} className={styles.button} onClick={props.onClick}>

      {props.children}

    </button>

  );

};

export default Button;

81. Dynamic Styles with CSS Modules (use CSS Module in CourseInput)

Step 1 change CourseInput.css to CourseInput.module.css

.form-control {

  margin: 0.5rem 0;

}

.form-control label {

  font-weight: bold;

  display: block;

  margin-bottom: 0.5rem;

}

.form-control input {

  display: block;

  width: 100%;

  border: 1px solid #ccc;

  font: inherit;

  line-height: 1.5rem;

  padding: 0 0.25rem;

}

.form-control input:focus {

  outline: none;

  background: #fad0ec;

  border-color: #8b005d;

}

.form-control.invalid input {

  border-color: red;

  background: #ffd7d7;

}

.form-control.invalid label {

  color: red;

}

Step 2 copy CourseInput.js like this

import React, { useState } from "react";

// import styled from "styled-components";

import Button from "../../UI/Button/Button";

import "./CourseInput.module";

// const FormControl = styled.div`

//   margin: 0.5rem 0;

// & label {

//   font-weight: bold;

//   display: block;

//   margin-bottom: 0.5rem;

//   color: ${props => props.invalid ? 'red' : 'black'}

// }

// & input {

//   display: block;

//   width: 100%;

//   border: 1px solid ${props => (props.invalid ? 'red' :'#ccc')};

//   background: ${props => props.invalid ? '#ffd7d7' : 'transparent'}

//   font: inherit;

//   line-height: 1.5rem;

//   padding: 0 0.25rem;

// }

// & input:focus {

//   outline: none;

//   background: #fad0ec;

//   border-color: #8b005d;

// }

// `

const CourseInput = (props) => {

  const [enteredValue, setEnteredValue] = useState("");

  const [isValid, setIsValid] = useState(true);

  const goalInputChangeHandler = (event) => {

    if (event.target.value.trim().length > 0) {

      setIsValid(true);

    }

    setEnteredValue(event.target.value);

  };

  const formSubmitHandler = (event) => {

    event.preventDefault();

    if (enteredValue.trim().length === 0) {

      setIsValid(false);

      return;

    }

    props.onAddGoal(enteredValue);

  };

  return (

    <form onSubmit={formSubmitHandler}>

      <div

        className={`styles['form-control']} ${!isValid && StyleSheet.invalid}`}

      >

        <label>Course Goal</label>

        <input type="text" onChange={goalInputChangeHandler} />

      </div>

      <Button type="submit">Add Goal</Button>

    </form>

  );

};

export default CourseInput;

82. Module Resource

For that, you find **multiple code snapshots** for this module here in this Github repository: <https://github.com/academind/react-complete-guide-code/tree/06-styling>