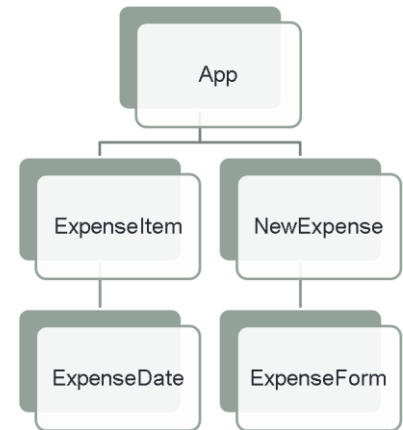


USING FORM FOR INPUT

Adding form inputs

```
import "../ExpenseForm.css"
const ExpenseForm = () => {
  return(
    <form>
      <div className="new-expense__controls">
        <div className="new-expense__control">
          <label>Title</label>
          <input />
        </div>
        <div className="new-expense__control">
          <label>Amount</label>
          <input type="number"/>
        </div>
        <div className="new-expense__control">
          <label>Date</label>
          <input type="date" min="2019-01-01"
            max="2022-12-31" />
        </div>
      </div>
      <div className="new-expense__actions">
        <button type="submit">Add Expense</button>
      </div>
    </form>
  );
}
export default ExpenseForm;
```

```
const NewExpense = () => {
  return(
    <div className="new-expense">
      <ExpenseForm />
    </div>
  );
}
export default NewExpense;
```



The screenshot displays the 'NewExpense' form at the top, which has a purple header and a light purple body. It contains three input fields: 'Title', 'Amount', and 'Date' (with a date picker). Below the inputs is an 'Add Expense' button. Below the form is a list of three existing expenses, each with a date, title, and amount, and a 'Change Title' button.

Date	Title	Amount	Action
August 14 2020	Groceries	Rs 900	Change Title
March 12 2021	New TV	Rs 34000	Change Title
March 28 2021	SofaSet	Rs 25000	Change Title

Listening to user input

```
const ExpenseForm = () => {  
  
  const titleChangeHandler = (event) => {  
    console.log(event.target.value)  
  }  
  
  return(  
    <form>  
      <div className="new-expense__controls">  
        <div className="new-expense__control">  
          <label>Title</label>  
          <input onChange={titleChangeHandler}/>  
        </div>  
        ...  
      </div>  
      <div className="new-expense__actions">  
        <button type="submit">Add Expense</button>  
      </div>  
    </form>  
  );  
}  
export default ExpenseForm;
```

The image shows a web application interface for adding an expense. The form is purple and contains three input fields: 'Title', 'Amount', and 'Date'. The 'Title' field has the text 'table' entered. A green arrow points from the 'Title' input field to the 'titleChangeHandler' function in the code above. Another green arrow points from the 'titleChangeHandler' function to a dropdown menu on the right side of the form, which shows a list of suggestions: 't', 'ta', 'tab', 'tabl', 'table'. The dropdown menu is also highlighted with a red box.

Storing input into state – working with multiple states

```
import React, {useState} from 'react';
const ExpenseForm = () => {
  const [inputTitle, setInputTitle] = useState('')
  const [inputAmount, setInputAmount] = useState('')
  const [inputDate, setInputDate] = useState('')

  const titleChangeHandler = (event) => {
    setInputTitle(event.target.value)
  }
  const amountChangeHandler = (event) => {
    setInputAmount(event.target.value)
  }
  const dateChangeHandler = (event) => {
    setInputDate(event.target.value)
  }
  return(
    <form>
      <div className="new-expense__controls">
        <div className="new-expense__control">
          <label>Title</label> <input onChange={titleChangeHandler}/>
        </div>
        <div className="new-expense__control">
          <label>Amount</label>
          <input type="number" onChange={amountChangeHandler}/>
        </div> ...
      </div>
    </form>
  )
}
```

Diagram illustrating the flow of data from the input field to the state:

- 1. The `titleChangeHandler` function is called when the input field's value changes.
- 2. The `titleChangeHandler` function calls `setInputTitle` to update the state.
- 3. The `setInputTitle` function updates the `inputTitle` state.

Form submission

```
const ExpenseForm = () => {  
  
  const [inputTitle, setInputTitle] = useState('')  
  const [inputAmount, setInputAmount] = useState('')  
  const [inputDate, setInputDate] = useState('')  
  
  const titleChangeHandler = (event) => { ...  
  }  
  const amountChangeHandler = (event) => { ...  
  }  
  const dateChangeHandler = (event) => { ...  
  }  
  const submitHandler = (event) => {  
    event.preventDefault();  
  }  
  return(  
    <form onSubmit={submitHandler}>  
      <div className="new-expense__controls"> ...  
      </div>  
      <div className="new-expense__actions">  
        <button type="submit">Add Expense</button>  
      </div>  
    </form>  
  );  
}
```

OnSubmit default behaviour is to reload

Form submission — 1 — extracting data, 2-way binding

```
const ExpenseForm = () => {
```

3 : setting values onchange

```
  const [inputTitle, setInputTitle] = useState('')
  const [inputAmount, setInputAmount] = useState('')
  const [inputDate, setInputDate] = useState('')
```

4 : on submit passing values to object

```
  const titleChangeHandler = (event) => {...}
  const amountChangeHandler = (event) => {...}
  const dateChangeHandler = (event) => {...}
```

```
  const submitHandler = (event) => {
    event.preventDefault();
```

```
    const expenseData = {
      title:inputTitle,
      amount:inputAmount,
      date:inputDate
    }
```

```
    console.log(expenseData)
```

```
    setInputAmount('')
    setInputDate('')
    setInputTitle('')
```

```
  }
```

7 : reset again

2 : Default value

```
  return(
```

```
    <form onSubmit={submitHandler}>
```

```
      <div className="new-expense_controls">
```

```
        <div className="new-expense_control">
```

```
          <label>Title</label>
```

```
          <input value={inputTitle}
            onChange={titleChangeHandler}/>
```

```
        </div>
```

```
        ...
```

```
      </form>
```

```
    );
```

```
  }
```

Title
Mouse

Amount
900

Date
01-12-2021

Add Expense

Title
Amount
Date
dd-mm-yyyy

Add Expense

[HMR] Waiting for update signal from WDS...

```
{title: 'Mouse', amount: '900', date: '2021-12-01'}
```

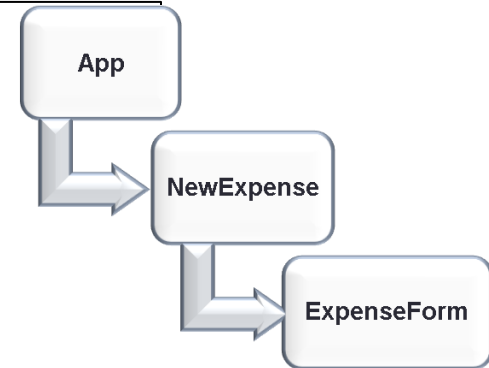
Passing data from child to parent component

```
const NewExpense = () => {
```

```
  const saveExpenseDataHandler = (inputExpenseData) => {  
    const expenseData = {  
      ...inputExpenseData,  
      id: Math.random().toString()  
    }  
    console.log("In NewExpense ", expenseData)  
  }  
}
```

```
  return(  
    <div className="new-expense">  
      <ExpenseForm onSaveExpenseData={saveExpenseDataHandler} />  
    </div>  
  );  
}
```

```
export default NewExpense;
```



```
const ExpenseForm = (props) => {
```

```
  ...  
  const submitHandler = (event) => {  
    event.preventDefault();  
    const expenseData = {  
      title: inputTitle,  
      amount: inputAmount,  
      date: new Date(inputDate)  
    }  
  }
```

```
  //console.log(expenseData)  
  props.onSaveExpenseData(expenseData);  
  ...  
}
```

Passing data from child to parent

```
function App() {  
  const expenses = [...];  
  
  const addExpenseHandler = (expense) => {  
    console.log("In App component ", expense)  
  }  
}
```

```
return (  
  <div className="App">  
    <h2>Welcome to React!</h2>  
    <NewExpense onAddExpense={addExpenseHandler} />  
    ...  
  );  
}  
export default App;
```

Title
Mouse

Amount
900

Date
03-12-2021

Add Expense

In App component [App.js:20](#)
▶ {title: 'Mouse', amount: '900', date: '2021-12-03', id: '0.25759054649698765'}

```
const NewExpense = (props) => {  
  const saveExpenseDataHandler = (inputExpenseData) => {  
    const expenseData = {  
      ...inputExpenseData,  
      id: Math.random().toString()  
    }  
    //console.log("In NewExpense ", expenseData)  
    props.onAddExpense(expenseData)  
  }  
  return(...);  
}  
export default NewExpense;
```