9. 리스트 렌더링(1) _ 리스트 렌더링

> 리스트를 state 값과 연동하기
>
> - 추가, 삭제

1. 추가

: 결제 추가를 하면 아래 리스트에 값이 추가되게 만들어 보기

```
const getPaymentFormData = (data) => {
    setExpenses(
        [
            {
                id: Math.random().toString(),
                title: data.name,
                amount: data.price,
                data: new Daae(data.today),
            },
            ...expenses
        ],
    );
};
```

2. 삭제

: 삭제 버튼을 리스트에 넣어서 버튼 클릭 시 해당 리스트의 아이템이 삭제되게 만들어 보기

(1) 삭제 버튼 만들기 (`ExpensesItem.js`)

```
<button onClick={()=> props.deleteExpenseItem.props.id}>삭제하기</button>
```

(2) 삭제 로직은 `App.js` 에 구현하는 것을 권장

```
const getPaymentFormData = (data) => {
    setExpenses(
        [
            {
                id: Math.random().toString(),
                title: data.name,
                amount: data.price,
                data: new Daae(data.today),
            },
            ...expenses
        ],
    );
    const deleteExpensesItem = (id) => {

    };

    return (
```

```
        <>
        <PaymentForm getPaymentFormData={getPaymentFormData} />
        <Expenses item={expenses} deleteExpenseItem={deleteExpensesItem} />
        </>
    )
};
```

```
//Expenses.js
<Card className="expenses">
    {props.items.map((itme) => (
        <ExpenseItem
            id={item.id}
            title={itme.title}
            amout={itme.amout}
            date={itme.date}
            deleteExpenseItem={props.deleteExpenseItem}
    ))}
</Card>
```

- 삭제 기능 구현 (1. filter, 2. slice)

```
  const deleteExpensesItem = (id) => {
      console.log(id);
      //1. filter
      const newFilteredArray = expenses.filter(item => item.id !== id);
      setExpenses(newFilteredArray);
      //2. slice (id 값이 아닌 index 값이 필요)
      // [0, 1, 2, ..., index, index+1, ..., n-1]
      // [0, 1, 2, ..., index-1, index+1, ..., n-1]
      // [0, 1, 2, ..., index-1] [index+1, ..., n-1]
      const beforeArray = expenses.slice(0, index);
      const afterArray = expensees.slice(index+1);
      setExpenses([...beforeArray, ...afterArray]);
    };
```

> Array.prototype.filter() ===> id 값
>
> Array.prototype.slice() ===> index 값

부모 컴포넌트의 state 를 자식 리스트 컴포넌트에 연동하면 해당 리스트에 값이 추가되고 삭제가 될 때마다 바로 화면이 업데이트 된다.