고급소프트웨어실습1

Lecture 8 과제

반 : 4

학번 : 20171669

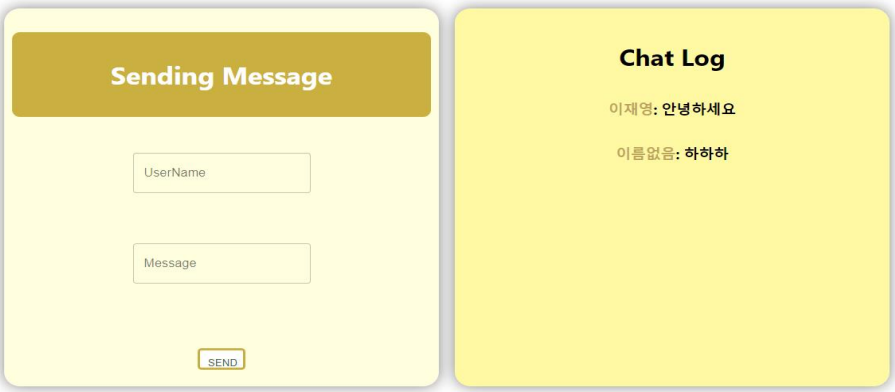
이름 : 이재영

1. 실습 설명

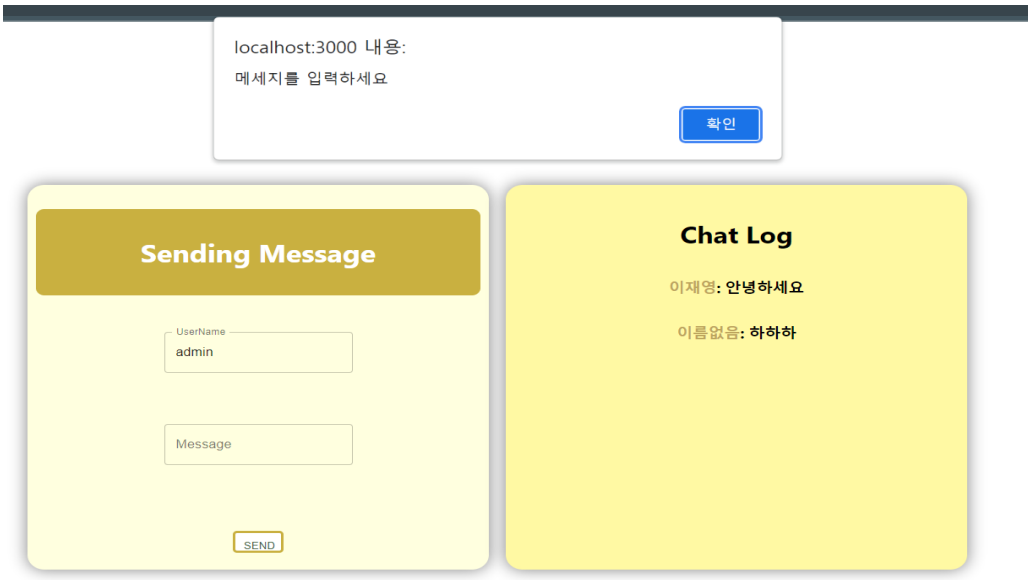
8주차 실습에서는 chat bot을 구현하는 것을 목표로 한다. Chat bot이란 간단히 parsing할 때, 사용자가 입력한 문장을 규칙 기반으로 해석을 하는 것으로 자연어 처리 등 학습 기반의 chat bot이 존재한다. 이번 실습에서는 css에서 채팅 프로그램 프론트엔드 및 기본 frame을 구현하는 것으로 user name과 Message를 입력하고 send버튼을 누르면 chat log로 결과가 나오게 하는 실습을 구현한다.

2. 실습 내용

Username과 Message를 입력하고 send를 누르면 chat log의 첫번째 줄과 같이 나타나고 username없이 message만 입력하면 2번째 줄과 같이 이름없음과 message내용이 나타난다.



만약 username만 있고 message내용이 없다면 메시지를 입력하세요 라는 콘솔창이 나타난다.



3. 실습 분석

실습에서 사용되는 컴포넌트들은 채팅 한 줄을 표현하는 ChatLogItem.jsx, 채팅 기록이 Chat Log 부분에 표현되는 ChatLogTemplate.jsx, 채팅 프로그램 페이지의 컴포넌트를 감싸는 ChatTemplate.jsx와 User의 Message 내용을 입력 받는 폼을 구현하는 MessengerTemplate.jsx 가 존재한다.

- ChatLogTemplate.jsx

```
function ChatLogTemplate({chats}) {

  // chats 배열을 받아 저장해 둔 채팅 로그를 보여줌
  // ChatLogItem 컴포넌트 사용
  return (
    <ChatContainer>
        {/* 8주차 실습 구현 */}
        <h1 style={{ color: '#000'}}>Chat Log</h1>
        {chats.map((chat_content) => {
            return <ChatLogItem chat={chat_content} />
        })}

    </ChatContainer>
  );

}
```

- Chat Log에 해당되는 Container 에서 Chat Log에 대한 색깔을 검은색 (color : #000)으로 설정하고 chats.map을 사용한다. 이 때 사용되는 chat_content는 ChatTemplate.jsx 에서 정의된 것으로 ChatLogTemplate 컴포넌트와 ChatTemplate 컴포넌트는 변수가 사용되는 관계를 가진다.

  다음은 Chat Log container에 해당되는 부분의 스타일을 구현한 것이다.

```
const ChatContainer = styled.div`
  width: 500px;
  margin: 10px;
  padding: 20px;
  // 해당 영역 모서리를 둥글게
  // 해당 영역 모서리에 그림자
  //background-color: #fff; // 해당 영역의 배경색 변경
  // 8주차 실습 구현

  display: flex;
  flex-direction: column;
  border: 3px solid #fff9a3;
  border-radius: 20px;
  background-color: #fff9a3;
  box-shadow : 0px 0px 20px gray;
;
```

- ChatTemplate.jsx

```jsx
// 전체를 감싸는 컴포넌트
function ChatTemplate() {
    const [ chats, setChats ] = useState([])

    // chats 배열에 채팅 추가
    const getChatLog = (username, message) => {
        // 8주차 실습 구현
        const chat_content = {
            username,
            message
        };
        setChats(chats.concat(chat_content));
    }

    return (
        <Container>
            <MessenagerTemplate getChatLog={getChatLog} />
            <ChatLogTemplate chats={chats} />
        </Container>
    );
}
```

ChatTemplate에서는 username과 message에 해당되는 chat_content를 선언해놓고 그것을 setChats(chats.concat(chat_content))를 선언한다.

ChatTemplate 컴포넌트는 전체를 감싸기 때문에 Chat Log와 관련된 ChatLogTemplate 컴포넌트와 Messenger와 관련된 MessengerTemplate 컴포넌트와 관련되어있다.

- MessengerTemplate.jsx

```
const onMsgSubmit = (e) => {
    e.preventDefault();
    // 8주차 실습 구현
    if (!MsgState.username) MsgState.username = "이름없음";
    if (!MsgState.message) {
        alert("메세지를 입력하세요");
        return;
    }
    getChatLog(MsgState.username, MsgState.message);
    setMsgState({ message: "", username: "" });
}
```

Messenger에서 message를 send하는 부분을 구현하는 함수로 username이 없으면 "이름없음"으로 설정하고 message가 없는 경우는 메시지를 입력하세요라는 경고창을 표출한다. 그리고 getChatLog로 send할 것을 저장한다. 후에 message와 username을 초기화한다.

```
return (
    <Form onSubmit={onMsgSubmit}>
        {/* 8주차 실습 구현 */}
        <Title>
            <h1>Sending Message</h1>
        </Title>

        <UserName>
            <TextField
                variant="outlined"
                label="UserName"
                name="username"
                onChange={onMsgChange}
                value={MsgState.username}
            />
        </UserName>
        <MessageContents>
            <TextField
                variant="outlined"
                label="Message"
                name = "message"
                onChange={onMsgChange}
                value={MsgState.message}
            />
        </MessageContents>
        <Button
            type="submit"
            onClick={onMsgSubmit}
            onKeyPress={onMsgChange}
            style={{ color: '#415e3e'}}
        >
            SEND
        </Button>
    </Form>
);
}
```

MessengerTemplate 컴포넌트에서 입력한 내용들을 저장해서 ChatLogTemplate 컴포넌트에서 사용될 수 있고 MessengerTemplate 컴포넌트는 ChatTemplate 컴포넌트에 관계가 되어있다.

```
const Button = styled.button`
  margin-top: 60px;
  padding: 10px;
  // 해당 영역 모서리를 둥글게
  // 해당 영역 모서리색 변경
  // 해당 영역의 배경색 변경
  // 8주차 실습 구현

  width : 60px;
  height : 30px;
  display: flex;
  flex-direction: column;
  border: 3px solid #c9b040;
  border-radius: 5px;
  align-self : center;
  background-color: #ffff;
  cursor : pointer;
  &:focus {
    outline: none;
  }
`;
```

```
const Title = styled.h1`
  margin: 10px;
  padding: 10px;
  // 해당 영역 모서리를 둥글게
  // 해당 영역의 배경색 변경
  // 해당 영역 안 텍스트 폰트색 변경
  // 8주차 실습 구현

  width : 500px;
  font-size : 17px;
  color : #ffff;
  align-self: center;
  flex-direction: column;
  border: 3px solid #c9b040;
  border-radius: 10px;
  background-color: #c9b040;
`;

// 채팅 메시지 입력 form
const Form = styled.form`
  width: 500px;
  margin: 10px;
  padding: 20px;
  // 해당 영역 모서리를 둥글게
  // 해당 영역 모서리에 그림자
  // 해당 영역의 배경색 변경
  // 8주차 실습 구현

  display: flex;
  flex-direction: column;
  border: 3px solid #ffffdf;
  border-radius: 20px;
  background-color: #ffffdf;
  box-shadow : 0px 0px 20px gray;
`;
```

다음은 send버튼과 title, 채팅 메시지 form 모양에 대한 style에 대한 설정을 보여준다.