고급 소프트웨어 실습(CSE4152)

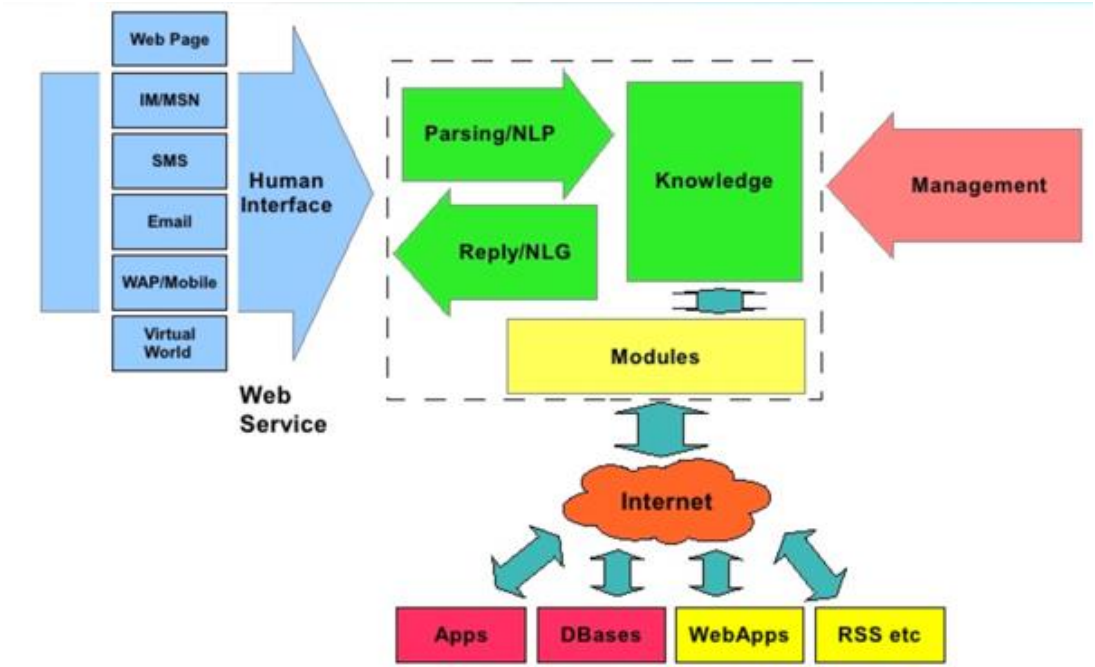
8주차
React – Chatting App(1)

목차

- Chat Bot
- JSX tag elements
- CSS 소개
- CSS-in-JSX
- Javascript – map()함수
- 실습 설명
- 과제 설명

Chat Bot

- 사람과 의사소통을 수행하는 프로그램
- 동작 방식에 따라 규칙 기반 / 기계학습 기반 챗봇으로 분류될 수 있다.

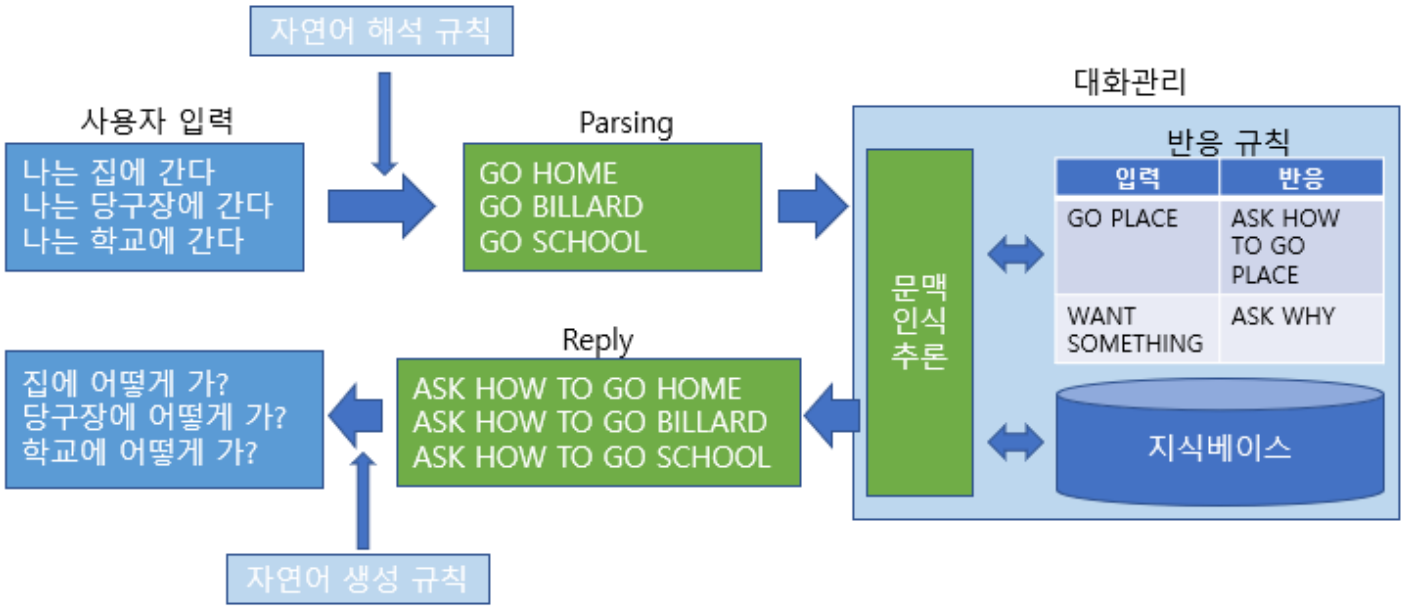


규칙 기반 Chat Bot

- 규칙
  - 자연어 해석 규칙
  - 자연어 생성 규칙
  - 반응 규칙
- 장점
  - 규칙을 기반으로 동작
  - 구현이 쉬움
  - 규칙이 잘 정의되면, 높은 품질의 대화 서비스 가능
- 단점
  - 규칙을 잘 정의 및 작성 하기가 어려움(시간, 노동 집약적)

규칙 기반 Chat Bot

- Parsing 시 사용자가 입력한 문장을 규칙 기반으로 해석
- 대화관리 시 규칙을 기반으로 답변의 규칙을 작성하며, 경우에 따라 지식 베이스를 이용하여 반응을 생성하기도 함
- Reply 시 규칙을 기반으로 답변 규칙에서 자연어 작성

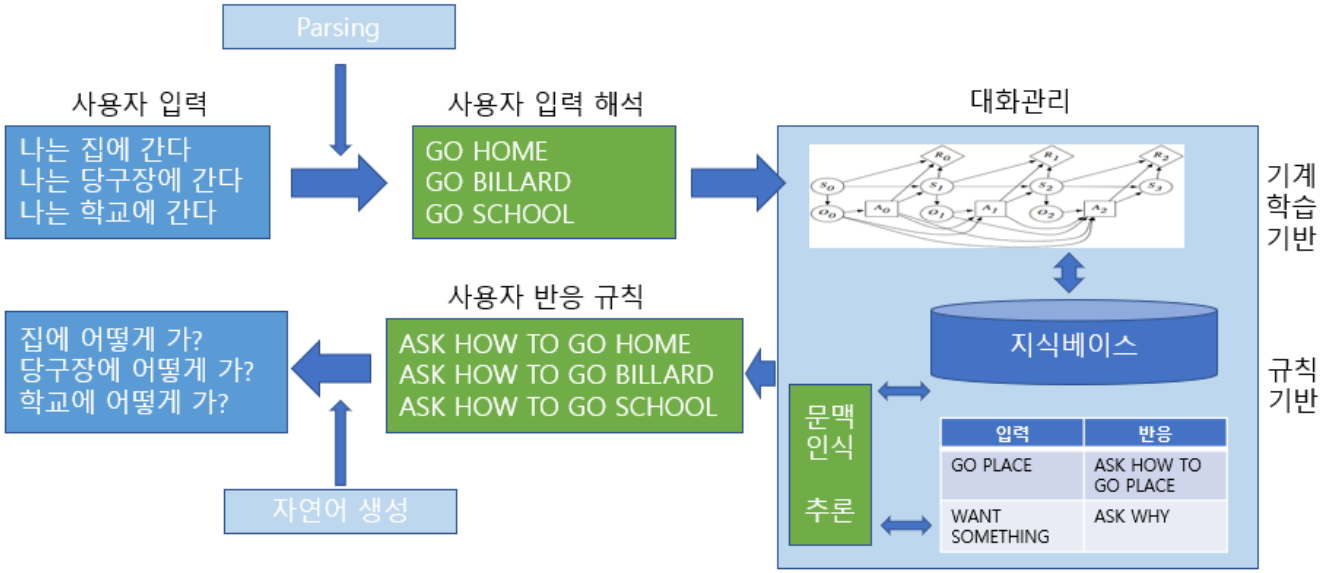


기계 학습 기반 Chat Bot

- 기계학습
  - 자연어 이해
  - 자연어 생성
  - 대화 관리
- 장점
  - 자연어 처리 기반으로 구현, 높은 수준의 대화가 가능.
- 단점
  - 자연어 이해/생성 및 대화 관리 모듈 구현상의 어려움
  - 자연어 관련 기계학습 모듈은 많은 기반 지식 및 툴을 필요
    - 말뭉치, 형태소 사전, 구문분석 트리 사전, 의미역 사전
  - 적용 되는 분야에 따라서 기존의 결과 재활용이 힘든 경우도 있음

기계 학습 기반 Chat Bot

- Parsing 시 사용자가 입력한 문장을 기계학습(자연어처리) 기반으로 해석 (구문 분석, 의도 분석)
- 대화 관리 시 사용자의 의도를 입력 받아 기계 학습을 통해 반응 수행 (강화학습 기반, POMDP)
- Reply 시 대화관리 모듈의 결과(의도)를 자연어로 생성



JSX – Tag Elements

- JSX 코드에서 HTML과 유사한 형태로 코드가 구현된다.
- HTML은 태그의 형태로 표현되는 element(요소)를 사용하여 웹 페이지의 구조적 설계, 즉 뼈대를 구성하기 위한 언어이다.

JSX – Tag Elements

-   : 웹 페이지의 논리적 구분을 정의하는 태그
- <h1></h1> : 페이지 문서의 제목 등의 역할을 하는 텍스트를 나타내는 태그

(글씨 크기는 <h1></h1> , <h2></h2>, <h3></h3> 순으로 작아진다.)

- <form></form> : 받은 정보를 전달하는 태그
- <TextField/> : 사용자에게 텍스트 값을 입력 받는 태그
- <button></button> : 클릭할 수 있는 버튼을 정의
- <{컴포넌트이름}/> : 작성한 React 컴포넌트를 표현하는 태그

(ex: <TodoTemplate />)

JSX – Tag Elements

- 태그 안에 필요한 속성을 정의해서 기능을 구현할 수 있다.
- Ex) `<div className="name-field">`
- 속성 종류
  - className : 지정된 태그의 클래스 값을 저장하는 속성
  - onSubmit : form이 전송되기 전에 실행되는 함수를 저장하는 속성
  - name : 입력 값을 지칭하는 이름을 지정하는 속성
  - onChange : 태그의 값이 변경될 때 실행되는 함수를 저장하는 속성
  - value: TextField 등에서 입력 값을 저장하는 속성
  - id: 지정된 태그의 id 값 저장하는 속성
  - variant: 태그의 스타일 나타내는 속성
  - label: TextField 등에서 값이 입력되기 전에 입력 창에 나타나는 텍스트를 저장하는 속성

CSS란

- HTML이 웹 페이지의 뼈대와 구조를 담당한다면, CSS는 페이지의 스타일과 레이아웃 등을 담당하여 웹 페이지를 꾸밀 때 사용되는 언어
- 웹 개발 시 정보(HTML)와 디자인(CSS)을 분리하여 개발 및 관리 가능

CSS 문법

- 선택자: 원하는 스타일을 적용하려는 대상, 범위
- 속성: 대상에서 어떤 스타일을 바꿀 것인지를 명시
- 속성 값: 명시한 속성에 대한 값, 속성에 어떻게 스타일을 입힐 것인지를 명시
- 사용: 속성: 속성 값; 의 형태로 ‘;’(세미콜론)을 이용하여 각 스타일 선언문을 구분

CSS 문법

: **div 요소**에 대해서 **폰트 크기를 18 픽셀로 설정**하고,

글씨 색은 **파란색**으로 설정

CSS의 주요 속성들

- height: 요소의 높이 조절
- width: 요소의 너비 조절
- margin: 요소와 요소 간의 여백 크기 조절
- padding: 요소의 테두리(border)와 컨텐츠 사이의 여백 크기 조절
- Background-color: 배경 색 설절
- Border-radius: 요소에 둥근 모서리를 만들 때 사용
- color: 텍스트 색 설정
- font-size: 텍스트의 크기 조절
- text-align: 텍스 정렬(값: left, right, center …)

Styled component란

- CSS-in-JS: 웹 페이지의 스타일을 CSS 파일이 아닌 컴포넌트가 작성된 JavaScript 파일에 바로 삽입하여 작성하는 스타일 기법
- Styled Components: CSS-in-JS를 구현하기 위해 사용되는 라이브러리
- styled-components 패키지 설치 후 원하는 jsx 파일에서 styled 함수 import
- 원하는 HTML element / React component의 태그명을 사용해서 사용자 지정 스타일 정의

CSS-in-JSX

- CSS-in-JSX의 일반적인 적용 방식은 다음과 같다.
  1. HTML 요소에 정의하고자 하는 스타일을 적용
  2. 적용한 스타일을 원하는 React 컴포넌트 변수에 저장

CSS-in-JSX 예시

```
import React from "react";
import styled from "styled-components";

const StyledContainer = styled.div`
  padding: 10px 20px;
  font-size: 18px;
  border: 1px solid lightgray;
  color: gray;
`;

function Container({ children }) {
  return <StyledContainer>{children}</StyledContainer>;
}
```

- styled-components에서 style를 import
- div 요소에 CSS 선언문 형태로 스타일 적용
- StyledContainer 컴포넌트 변수에 스타일 적용한 div 요소를 저장
- JSX 문에서 스타일 저장한 컴포넌트 사용

JavaScript – 화살표 함수

- function 키워드를 사용하지 않고 화살표('=>')를 사용하여 간단하게 함수를 표현
- 함수 표현식을 통해 나타내며, 항상 익명 함수로 사용
- Call back 함수로 사용 가능
- 화살표 함수의 기본 문법:

```
// 일반 함수
var foo = function () { console.log("foo") }; // foo

// 매개변수가 없는 화살표 함수
var foo = () => console.log("foo"); // foo
// 매개변수가 2개인 화살표 함수
var foo = (a, b) => {
  var c = 3;
  return a + b + c;
} // a+b+c를 return
```

JavaScript – map 함수

- 배열 내 각각의 요소에 대해 주어진 함수를 수행하고 그 결과를 모아 새로운 배열 생성
- 형태: array.map(callback (currentValue, index, array), thisArg)
  - currentValue : 배열 내 현재의 값
  - index : 배열 내 현재 값의 인덱스
  - array: 현재 배열
  - thisArg: callback 함수에서 this로 사용될 값

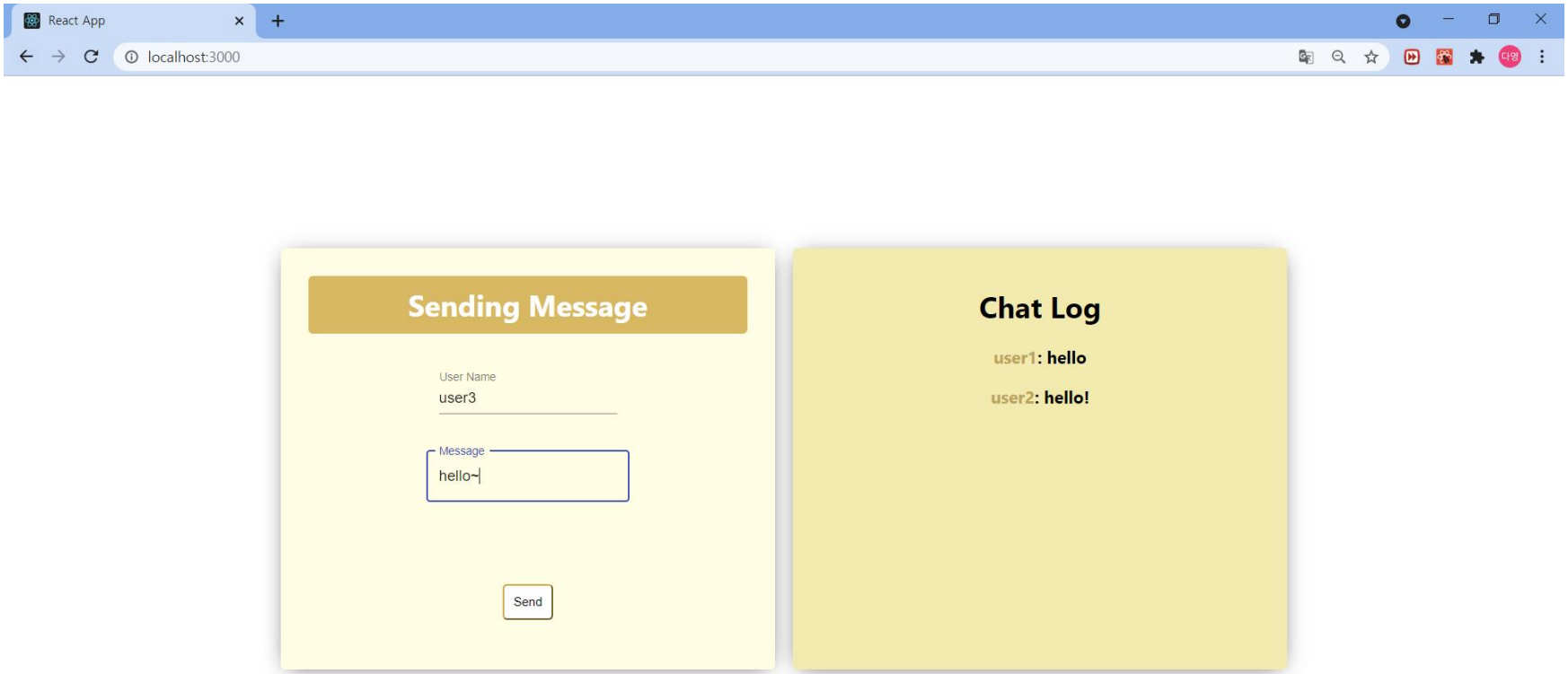
```
const arr = [1, 2, 3];
const marr = arr.map((curVal) => curVal + 1);
const marr2 = arr.map(function addOne(curVal){
  return curVal + 1;
})

console.log(marr); // [2, 3, 4]
console.log(marr2); // [2, 3, 4]
```

실습

- 채팅 프로그램 프론트엔드 및 기본 틀 구현
- Send 버튼 클릭 시 입력한 user name, message가 오른쪽 Chat Log에 추가되어 나타나도록 구현
- (새로 고침 시 ChatLog는 초기화됨)

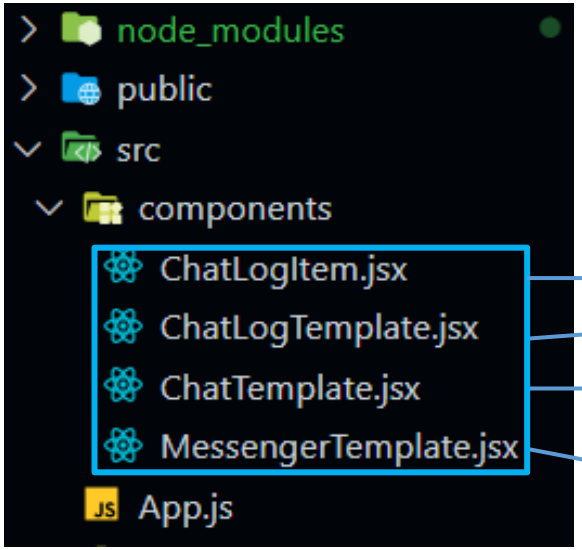
실습

- 실습 완료 후 실행 결과



실습 파일 구조

8주차: 채팅 프로그램 프론트엔드 구현



- 채팅 한 줄을 표현하는 컴포넌트
- 채팅 기록이 표현되는 컴포넌트(8주차 구현)
- 채팅 프로그램 페이지의 두 컴포넌트를 감싸는 템플릿 컴포넌트(8주차 구현)
- 사용자의 채팅 내용을 입력 받는 폼 컴포넌트(8주차 구현)

실습

- 스타일 구현
  - src > components > ChatLogTemplate.jsx, ChatTemplate.jsx, MessengerTemplate.jsx의 styled components 부분의 주석을 보고 실행 결과와 최대한 비슷하게 구현
- 기능 구현
  - ChatLogTemplate.jsx
    - 해당 컴포넌트가 입력한 채팅 객체들을 차례대로 표시하도록 return 함수를 구현
  - ChatTemplate.jsx
    - getChatLog 함수: 새로운 채팅 객체를 생성해서 기존 chats 배열에 추가
  - MessengerTemplate.jsx
    - onMsgSubmit 함수
    - 메시지 입력 창에서 빈 경우에는 '메시지를 입력하세요'라는 내용의 경고 창 출력 후 리턴
    - 사용자의 이름 입력 창이 빈 경우에는 사용자 이름을 '이름없음'으로 저장
    - getChatLog 함수 사용하여 새로운 채팅 추가 후 MsgState의 속성들을 빈 문자열로 초기화
    - Form 컴포넌트 완성
    - Title, UserName, MessageContents, Button 컴포넌트

과제: 보고서 – 프로젝트 분석

- 각 주요 컴포넌트들의 주요 기능 및 구조에 대하여 설명하고 컴포넌트 간의 상호 관계에 대해 설명

참고

- JSX 레퍼런스 : https://reactjs.org/docs/jsx-in-depth.html
- CSS 레퍼런스(MDN) : https://developer.mozilla.org/en-US/docs/Learn/CSS
- Styled Components 레퍼런스 : https://styled-components.com/