

[개인 과제] 프론트엔드 기초

훈련 과 정 명	[신한투자증권] 프로디지털아카데미 4기
강 의 과 목 명	프론트엔드 개발 기초 (JavaScript)
강 사 명	신윤수
과 제 명	<p>실전 자바스크립트 종합 과제</p> <ul style="list-style-type: none"> - 웹 분석하여 데이터 수집 및 저장하기 (HTML) - 웹 분석하여 데이터 수집 및 저장하기 고급 (JSON) - Bootstrap 이용하여 UI 구성하기. - React에서 통신 후 UI에 Rendering하기 - React에서 복잡한 Component 구성하기
제 출 기 한	<p>5월 27일 오후 23:59 까지 제출</p> <p>[과제 제출시 주의사항]</p> <ul style="list-style-type: none"> - 첨부파일을 다운받고 작업하시길 추천드립니다. - 패키지 설치하는 자유롭게 하셔도 되지만, 제출시에는 node_modules 디렉토리는 모두 제거하여 주십시오. - 과제 제출시 homework 디렉토리(가장 상위 디렉토리)의 이름을 제출자 이름으로 변경 후 압축하여 제출바랍니다. - 원활한 과제평가를 위하여 React UI(과제 그룹 2)는 산출 결과물을 캡처하여 이미지파일로 저장하여 주시기 바랍니다. - 결과물 캡처 이미지 파일은 가장 상위 폴더에 위치하며(제출자 이름) 파일이름은 세부과제번호로 해주십시오. <p>(예: 신윤수/2.1.1.jpg)</p>

자바스크립트 종합 과제로서 총 두개의 파트로 나뉘어집니다.

1. Node.js환경에서의 데이터 수집
2. React를 활용한 UI실습 및 통신

배운 것을 모두 활용하시고, 배우지 않은 것을 찾아서 활용하셔도 좋습니다.

[총 세부 과제 및 과제별 배점]

세부 과제의 총 개수는 7개이며, 각 세부과제별 내용과 배점은 다음과 같습니다.

- 1.1: 다음 검색(뉴스탭)에서 뉴스 데이터 수집하기 (15점)
- 1.2: 다음 주가차트에서 데이터 수집하기 (15점)
- 2.1.1. StockApp UI 완성하기 (20점)
- 2.1.2. StockApp 데이터 통신하여 Rendering하기 (10점)
- 2.2.1. UserApp UI완성하기 (20점)
- 2.2.2. UserApp 데이터 통신 및 Rendering (10점)
- 2.2.3. UserApp 데이터 통신 및 고급 응용 (10점)

1. Node.js환경에서의 데이터 수집

현 과제는 Node.js 환경에서 작업합니다.

1.1 다음 검색(뉴스탭)에서 뉴스 데이터 수집하기 (15점)

검색 URL: <https://search.daum.net/search?w=news&cluster=y&q=삼성전자>

삼성전자

✖

🔍

통합

뉴스

어학사전

지도

책

통합웹

이미지

쇼핑

..

관련

삼성전자 우

삼성전기

sk 하이닉스

삼성전자 주식

삼성전자 배당금

엘지전자

삼성전자 주가

삼성전자 우선주

삼성전자 서비스센터

카카오

삼성물산

삼성전자 물

정확도순

최신순

오래된순

오픈

뉴스제휴 언론사 검색결과 ①

뉴스검색 설정

기사 수 추이 ①

892건

5.7

5.15

파이낸셜뉴스

갤럭시Z 폴딩6 '두뇌'는 퀄컴칩만? 삼성전자 AP전략 바꿀지 관심

IT 틈새터 온리크스와 스마트프릭스가 제작한 갤럭시Z 폴딩6 예상 렌더링 삼성전자가 차세대 폴딩 스마트폰 '갤럭시Z 폴딩6'에서 두뇌에 해당하는 모바일 애플리...

1시간전

세계일보

애플 넘어설 삼성의 비밀병기, '여기'서 개발된다 [이동수는 이동중]

"이것만 잘해도 애플과 겨뤄볼 만하겠다는 생각이 듭니다." 지난달 16일(현지시간) 이탈리아 밀라노에서 취재진과 만난 한중회 삼성전자 디바이스경험(DX)부문장(...

11시간전

<그림 1>

기간

전체

1일

1주

1개월

6개월

1년

2024.4.1.~2024.4.5. ^

2024.4.1.

~

2024.4.5.

년	월	일
2021	2	1
2022	3	2
2023	4	3
2024	5	4

적용

유형

전체

포토

동영상

인포그래픽

자동생성기사

보도자료

탐사뉴스

팩트체크

언론사

전체

가나다순 ▾

직접입력 ▾

<그림2: 옵션 클릭 후 날짜 설정>



<그림3: 해당하는 날짜에 속하는 뉴스리스트>

[과제 내용]

1. <그림1: 검색 URL: <https://search.daum.net/search?w=news&cluster=y&q=삼성전자>> 에 접속하면 나오는 페이지에서 옵션 클릭
2. <그림2>에서 볼 수 있듯이 날짜 설정을 할 수 있다.

다음 검색에서 날짜 설정을 하여 총 3종류의 날짜범위(아래 [날짜범위] 참조)에 해당하는 "삼성전자"검색 결과 중에 제목과 뉴스 URL을 수집하여 **Json** 형태로 저장하는 코드를 작성하고 실제로 저장하시오. [파일이름: **news.json**] (하단 결과에서 확인)

- 저장데이터의 key값은 날짜범위의 종료날짜(%YYYY%MM%DD)로 하고 value는 뉴스내용({title: "~", "url": ~})의 배열로 한다.
- 검색결과는 1페이지만 조회하도록 한다.(뉴스의 개수 10개)

[파일 저장 경로]:

- <제출자이름>/homework-node/news.json 으로 저장

[저장할 Json Format 형태:]

```
{
  [endDate]: [{title: "title1...", url: "https://~~url1..." }]
}
```

[날짜범위]

1. 20240401 00시 00분 00초~20240405 23시59분59초
2. 20240408 00시 00분 00초~20240412 23시59분59초
3. 20240415 00시 00분 00초~20240419 23시59분59초

[결과 예시 - news.json]

```

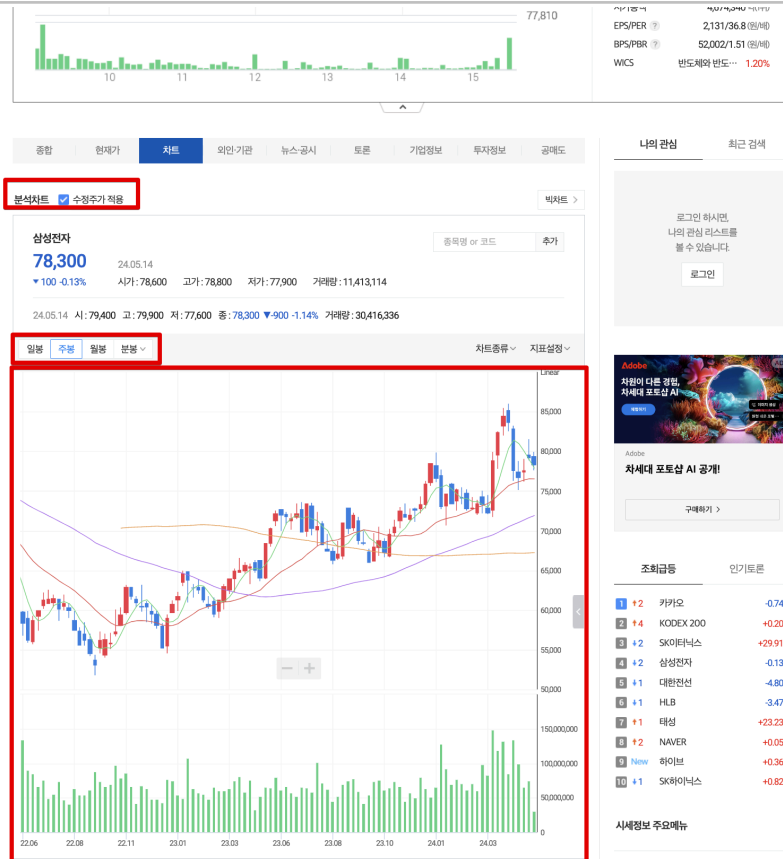
1  {
2    "2024-05-05": [
3      {
4        "title": "\"폴더블폰=삼성전자 갤럭시S\" 공식 깨는 화웨이와 황사바람",
5        "url": "http://v.daum.net/v/20240505093653690"
6      },
7    > { ...
10   },
11   > { ...
14   },
15   > { ...
18   },
19   > { ...
22   },
23   > { ...
26   },
27   > { ...
30   },
31   > { ...
34   },
35   > { ...
38   },
39   {
40     "title": "삼성전자, 가정의 달 맞아 '세상 편한 AI 라이프' 페스티벌 실시",
41     "url": "http://v.daum.net/v/20240503114117645"
42   }
43 ],
44 "2024-05-12": [
45   {
46     "title": "삼성전자, 베트남 총리에 \"수년간 연간 10억달러씩 투자\"",
47     "url": "http://v.daum.net/v/20240512210047696"
48   },
49   {

```

1.2 다음 주가차트에서 데이터 수집하기 (15점)

삼성전자 코드: 005930

daum.net 주가차트URL: <https://finance.daum.net/quotes/A005930#chart>



<그림 1>

[과제 내용]

daum.net 주가차트URL에 접속하고 스크롤을 조금 내리면 <그림 1> 과 같은 페이지를 확인할 수 있다.

현재 <그림 1>은 수정주가 적용이 체크되어 있는 상태이고, 삼성전자에 대한 주봉에 대한 내역을 볼 수 있는 상태이다.

2024년 4월 1일부터 2024년 4월 19일까지의 주봉 [총 3개의 데이터] 를 수집하여 Array<StockData>형태의 json포맷으로 저장하는 코드를 작성하고 실제로 저장하여라. [파일이름: **stock.json**] (하단 결과예시 확인)

StockData에는 반드시 [date, tradePrice, openingPrice, highPrice, lowPrice, candleAccTradePrice] key는 들어가야 하며,

해당 key가 의미하는 바는 다음과 같다.

(date: 종가 날짜, tradePrice: 종가, openingPrice: 시가, highPrice: 고가, lowPrice: 저가, candleAccTradePrice: 거래대금)

[파일 저장 경로]:

- <제출자이름>/homework-node/stock.json 으로 저장

[결과예시 - stock.json]

```
1  [
2  > { ...
17 } ,
18 > { ...
33 } ,
34 {
35   "symbolCode": "A005930",
36   "date": "2024-04-19",
37   "candleTime": "2024-04-15 00:00:00.0",
38   "tradePrice": 77600,
39   "openingPrice": 82900,
40   "highPrice": 83200,
41   "lowPrice": 76300,
42   "candleAccTradePrice": 10670676192269,
43   "candleAccTradeVolume": 133913001,
44   "tradeTime": "153010",
45   "timestamp": 1713736240069,
46   "change": null,
47   "changeRate": null,
48   "changePrice": null
49 }
50 ]
```

위 빨간 박스에 해당하는 **key**는 필수, 그 외는 자유(넣으셔도 되고 빼도 됨)

2. React 응용

본 과제는 **React**를 자유롭게 응용하시면 됩니다. 컴포넌트 분리는 하셔도 되고 하지 않으셔도 됩니다. 그에 따른 점수에 대한 영향은 없습니다. 요구사항만 충족시켜주시면 됩니다.

작업디렉토리: **homework-react**

현재 프로젝트는 **react-bootstrap**이 설치되어있는 환경이다.

- src/components/Root.jsx 에는 UserApp과 StockApp이 각각 Rendering되고 있다.
- public/data 디렉토리에는 각각 stock.json과 news.json이 있다.

다음 조건들에 맞게 UserApp과 StockApp을 완성하시오.

**** 사전지식 (react개발 서버를 실행한 후<npm run start 후>)**

- 해당 url(예:localhost:3000)에 public에 위치한 경로를 입력하면 public에 위치한 static file을 반환
⇒ localhost:3000/data/stock.json에 HTTP Request시 stock.json 내용 반환

2.1.1. StockApp UI 완성하기 (20점)

작업 Component: src/components/StockApp.jsx

- react-bootstrap table Component:
<https://react-bootstrap.netlify.app/docs/components/table>

[과제 내용]

react-bootstrap table을 사용하여 다음 <그림1>에 해당하는 UI를 완성하시오.

부트스트랩 Table Component에 다음 속성에 해당하는 props만 true로 설정하시고 나머지 스타일링은 자유입니다.

<true로 설정할 props>

- striped
- bordered
- hover

Users

Stocks

날짜	종가	시가	고가	저가	거래대금
----	----	----	----	----	------

<그림1>

[이미지 캡처 저장 경로]:

- <제출자 이름>/2.1.1.jpg 로 저장 (jpg, png 등 확장자는 상관없음)

2.1.2. StockApp 데이터 통신하여 Rendering하기 (10점)

작업 Component: src/components/StockApp.jsx

[과제 내용]

- `/data/stock.json` 에 요청을 보내어 주별 **stock Data**를 받아오고, 해당 내용을 **table-row**로 **Rendering** 하시오.
- 테이블에서 **th**(테이블 헤더)에 대응되는 `stock.json` 의 **key**는 다음과 같습니다.
 - [날짜: `date`, 종가: `tradePrice`, 시가: `openingPrice`, 고가: `highPrice`, 저가: `lowPrice`, 거래대금: `candleAccTradePrice`]
- 결과는 아래 [결과 예시] 확인

[결과 예시]

Users

Stocks

날짜	종가	시가	고가	저가	거래대금
2024-04-05	84500	83200	85500	82000	11098747991217
2024-04-12	83700	85200	86000	82500	7174699191474
2024-04-19	77600	82900	83200	76300	10670676192269

[이미지 캡처 저장 경로]:

- <제출자 이름>/2.1.2.jpg 로 저장 (jpg, png 등 확장자는 상관없음)

2.2.1. UserApp UI완성하기 (20점)

작업 Component: `src/components/UserApp.jsx`

- **react-bootstrap List Group Component:**
<https://react-bootstrap.netlify.app/docs/components/list-group#actionable-items>

[과제 내용]

1. **react-bootstrap ListGroup** 컴포넌트를 사용하여 다음 [결과 예시]에 해당하는 **UI**를 완성하시오.
 - 해당하는 데이터는 임의로 하셔도 무방합니다.
(ListGroup와 ListGroup.Item 만 사용하시면, 세부 스타일링은 과제점수에 반영 되지 않습니다.)

[결과 예시]

Users

0

1

Stocks

날짜	종가	시가	고가	저가	거래대금
2024-04-05	84500	83200	85500	82000	11098747991217
2024-04-12	83700	85200	86000	82500	7174699191474
2024-04-19	77600	82900	83200	76300	10670676192269

[이미지 캡처 저장 경로]:

- <제출자 이름>/2.2.1.jpg 로 저장 (jpg, png 등 확장자는 상관없음)

2.2.2. UserApp 데이터 통신 및 Rendering (10점)

작업 Component: `src/components/UserApp.jsx`

- 통신할 user URL: <https://jsonplaceholder.typicode.com/users>

[과제 내용]

- 위 링크(<https://jsonplaceholder.typicode.com/users>)에 요청을 보내면, 총 10명의 user에 대한 data를 불러온다.
위 링크에 HTTP요청을 하고 state에 저장한 후,
각각 <ListGroup.Item> 컴포넌트에서 다음형태로 Rendering하시오. ({user.id}. {user.name} - {user.email})
⇒ 데이터에서 [user.id, user.name, user.email] 속성이 필요
⇒ [결과 예시] 참조

[결과 예시]

Users	Stocks
1. Leanne Graham - Sincere@april.biz	날짜
2. Ervin Howell - Shanna@melissa.tv	종가
3. Clementine Bauch - Nathan@yesenia.net	시가
4. Patricia Lebsack - Julianne.OConner@kory.org	고가
5. Chelsey Dietrich - Lucio_Hettinger@annie.ca	저가
6. Mrs. Dennis Schulist - Karley_Dach@jasper.info	거래대금
7. Kurtis Weissnat - Telly.Hoeger@billy.biz	2024-04-05
8. Nicholas Runolfsson - Sherwood@rosamond.me	84500
9. Glenna Reichert - Chaim_McDermott@dana.io	83200
10. Clementina DuBuque - Rey.Padberg@karina.biz	85500
	83700
	85200
	86000
	82500
	7174699191474
	2024-04-12
	77600
	82900
	83200
	76300
	10670676192269
	2024-04-19

[이미지 캡처 저장 경로]:

- <제출자 이름>/2.2.2.jpg 로 저장 (jpg, png 등 확장자는 상관없음)

2.2.3. UserApp 데이터 통신 및 고급 응용 (10점)

작업 Component: src/components/UserApp.jsx

- React Bootstrap Modal Component:
<https://react-bootstrap.netlify.app/docs/components/modal>
- 통신할 URL:
 - <https://jsonplaceholder.typicode.com/todos>
 - <https://jsonplaceholder.typicode.com/posts>

[사전 지식]

- 위 링크는 모두 url parameter(query string)를 입력하여 세부 조건을 검색할 수 있다.
 - userId가 2에 해당하는 user의 todos정보를 가져오려면 다음과 같이 get 요청을 보내면 된다.
<https://jsonplaceholder.typicode.com/todos?userId=2>
 - userId가 8에 해당하는 user의 posts정보를 가져오려면 다음과 같이 get 요청을 보내면 된다.
<https://jsonplaceholder.typicode.com/posts?userId=8>

[과제 내용]

- (2.4)에서 작업한 User(<ListGroup.Item>)를 클릭하면 Modal창이 보이도록 작업하시오.
- 해당 Modal의 Title은 클릭한 user의 이름이 보이도록 하시오.

예: <Modal.Header><Modal.Title> {클릭한 user의 정보} </Modal.Title></Modal.Header>

- 본문에는 **user**가 작성한 **post**와 **user**의 **todo** 가 나열되도록 하시오. (스타일링은 신경쓰지 않으셔도 됩니다.)

예: (<Modal.Body>{본문내용}</Modal.Body>

- **Modal** 우상단 **X** 버튼을 클릭하면 **Modal**이 보이지 않도록 하시오.
- **[결과 예시]**를 참조하시오.

[결과 예시]

현재 3번 유저 클릭시 보이는 modal 창입니다.

The screenshot shows a web application with a modal window open. The modal is titled 'Clementine Bauch' and contains two columns: 'Posts' and 'Todos'. The 'Posts' column lists 10 items, and the 'Todos' column lists 10 items. The background shows a 'Users' table with 10 rows and a 'Stocks' table with 4 rows. The modal is highlighted with a red border.

시가	고가	저가	거래대금
83200	85500	82000	11098747991217
85200	86000	82500	7174699191474
82900	83200	76300	10670676192269

[이미지 캡처 저장 경로]:

- <제출자 이름>/2.2.3.jpg 로 저장 (jpg, png 등 확장자는 상관없음)