14. 외부API 연동 News Viewer만들기

- https://newsapi.org에서 제공하는 API를 이용하여 데이터를 조회
- styled-components를 활용하여 styling 실습

14.1 비동기작업의 이해

14.1.1 콜백함수

콜백지옥

- 10~...까지의 결과를 순차적으로 출력하고잛 할때 콜백함수를 중첨하여 구현하는 경우
- 콜백함수에 콜백함수를 넣어 구현할 수 있지만 코드의 가독성이 나빠졌다 이러한 형태의 코드를 `콜백지옥`이라고 한다.

src/example/callback.js - 콜백지옥 예제

```
function increase(number, callback) {
  setTimeout(() => {
    const result = number + 10;
    if(callback) {
      callback(result)
    }
  }, 1000);
}

console.log('작업시작...');
increase(0, result => {
  console.log(result);
  increase(result, result => {
    console.log(result);
    increase(result, result => {
      console.log(result);
      increase(result, result => {
        console.log(result);
      });
    });
  });
});
```

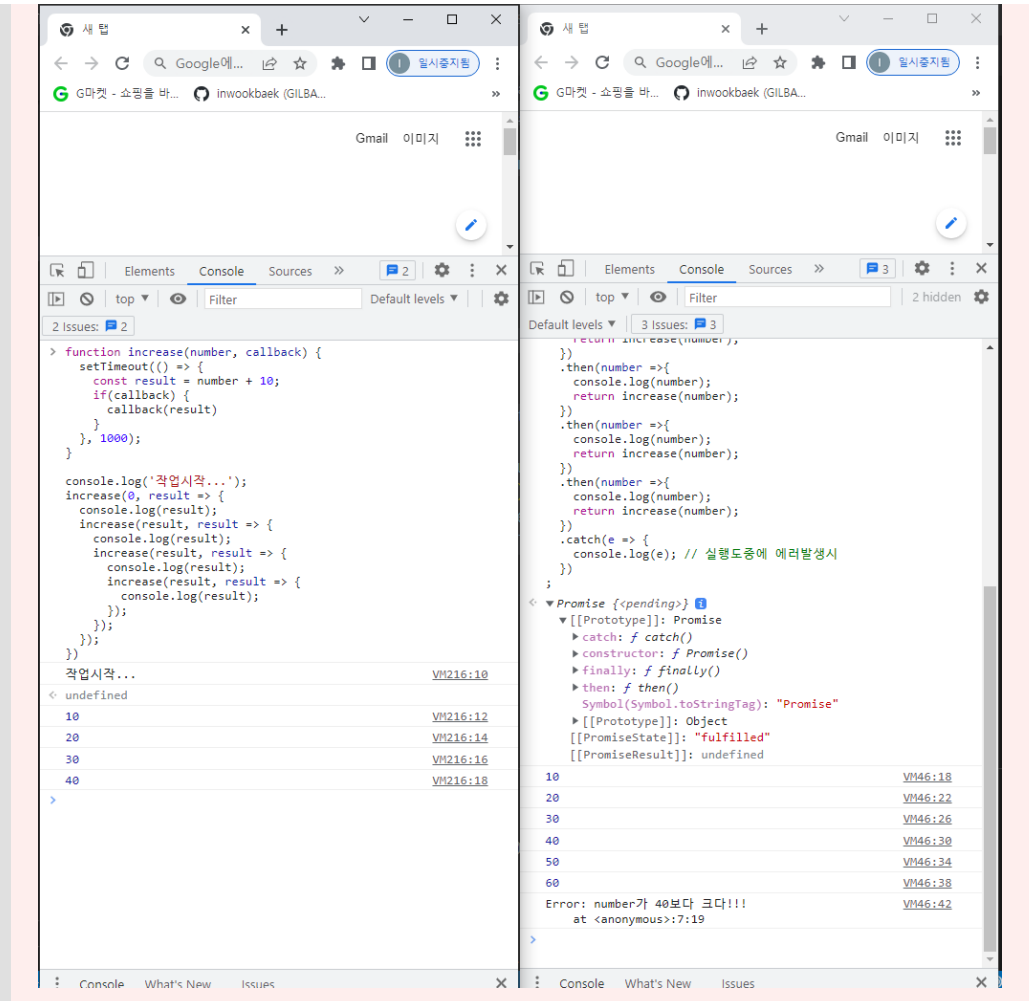
14.1.2 `Promise`

- `Promise는 콜백지옥같은 코드를 해결하는 방안으로 ES6에 도입된 기능`이다.

src/example/promise.js - Promise 적용 예제

```
function increase(number) {
  const promise = new Promise((resolve, reject) => {
    // resolve = success, reject = fail
    setTimeout(() => {
      const result = number + 10;
```

```
        if(number>50) {
          const e = new Error("number가 40보다 크다!!!");
          return reject(e);
        }
        resolve(result)
      }, 1000);
    });
    return promise;
}

increase(0)
  .then(number =>{ // Promise로 return된 값은 .then으로 받아올 수 있음
    console.log(number);
    return increase(number);  // promise를 retun
  })
  .then(number =>{
    console.log(number);
    return increase(number);
  })
  .then(number =>{
    console.log(number);
    return increase(number);
  })
  .then(number =>{
    console.log(number);
    return increase(number);
  })
  .catch(e => {
    console.log(e); // 실행도중에 에러발생시
  })
```



14.2 axio

- axios는 자바스크립트 HTTP 클라이언트이다.
- 이 라이브러리는 `HTTP요청을 Promise기반으로 처리` 한다는 점이다.
- 설치 : `yarn add axios`

.prettierrc

- Prettier로 소스코드를 정리하려면 최상위폴더에 .prettierrc 파일설정

```
{
  "singleQuote": true,
  "semi": true,
  "useTabs": false,
  "tabWidth": 2,
  "trailingComma": "all",
  "printWidth": 80
}
```

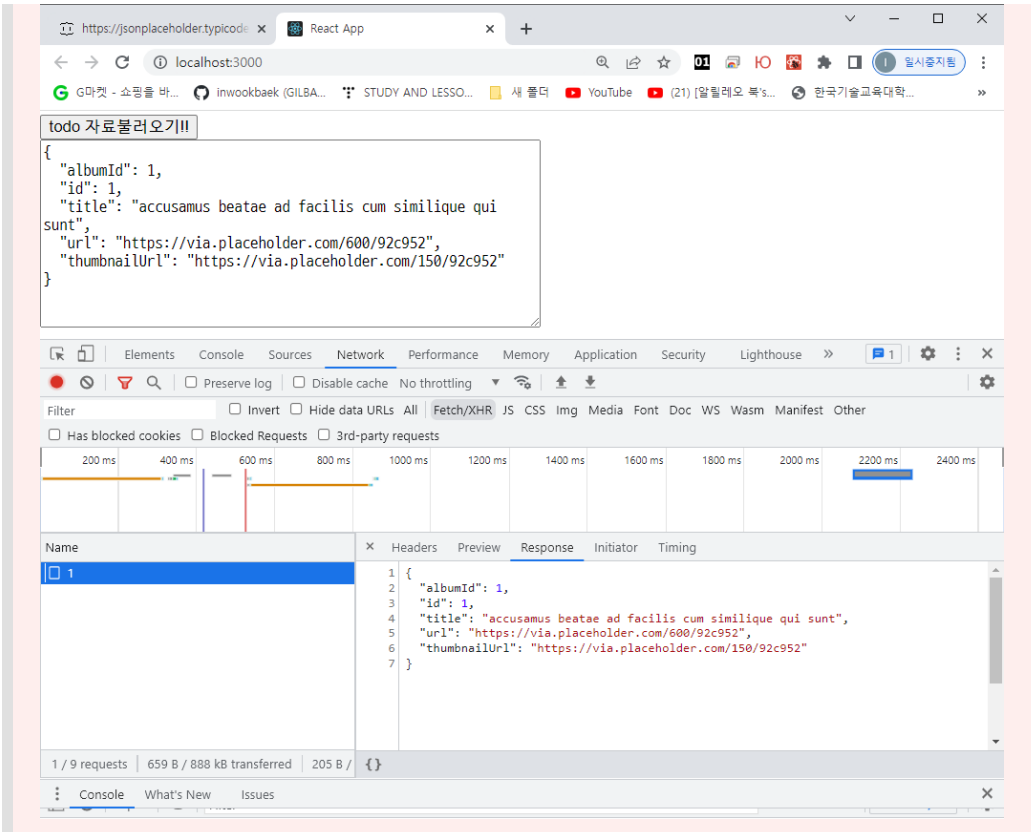
jsconfig.json

- VS Code에서 파일 자동불러오기 기능 적용하려면 최상위폴더에 jsconfig.json 설정

```
{
  "compilerOptions": {
    "target": "es6"
  }
}
```

src/App.js

```
import { useState } from 'react';
import { axios } from 'axios';

const App = () => {

  const [data, setData] = useState(null);
  const onClick = () => {
    axios.get('https://jsonplaceholder.typicode.com/photos/1')
      .then(response => setData(response.data))
  }

  return (
    <div className="App">
      <div>
        <button onClick={onClick}>todo 자료불러오기!!</button>
      </div>
      {data && <textarea rows={7} value={JSON.stringify(dat, null, 2)}
readOnly={true} />}
    </div>
  );
}

export default App;
```



async/await 적용

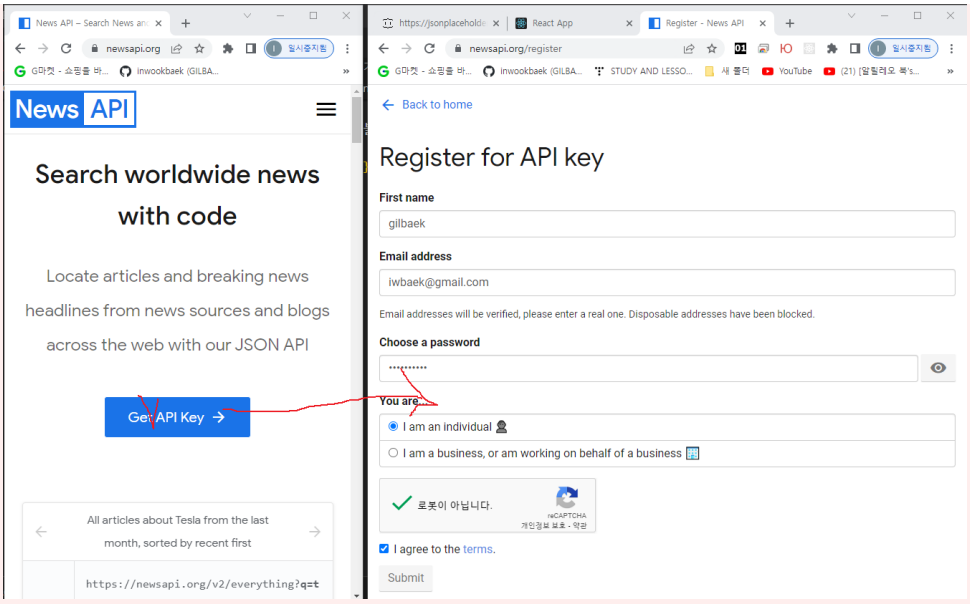
src/App.js

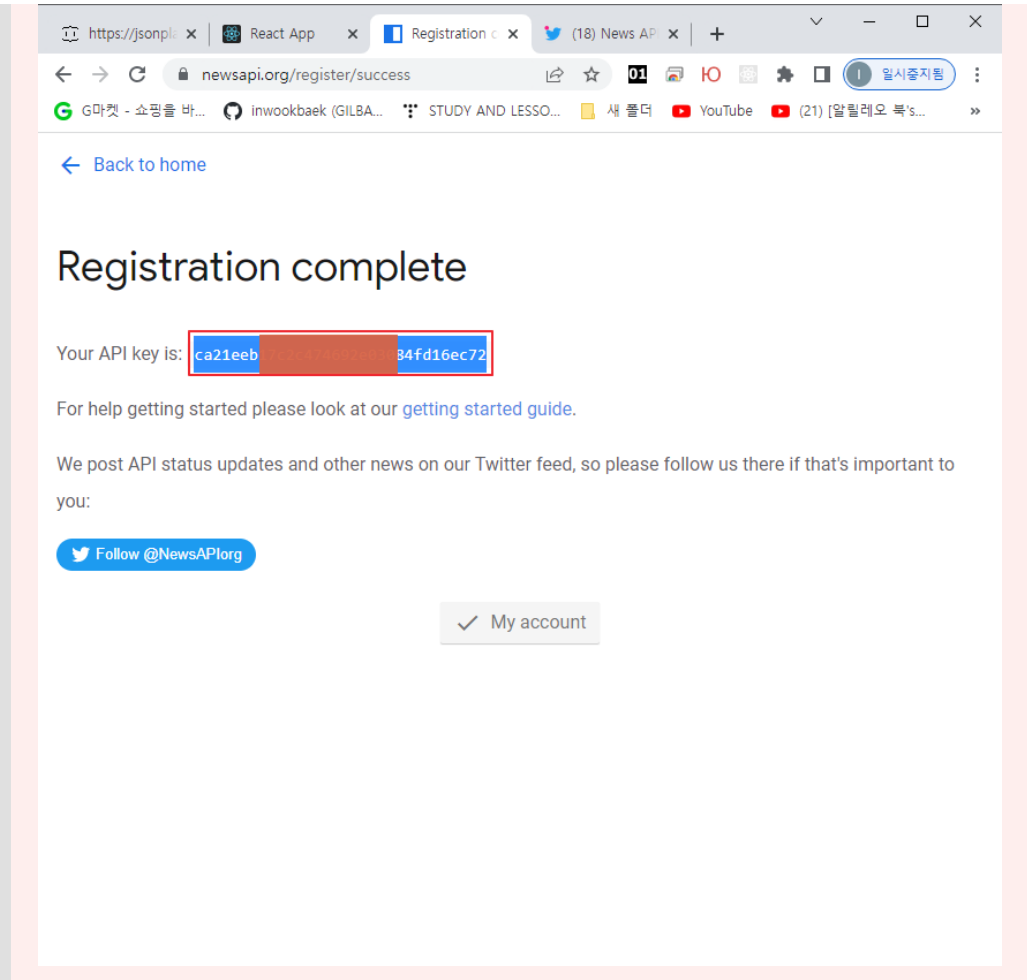
```
import { useState } from 'react';
import axios from 'axios';

const App = () => {
```

```
  const [data, setData] = useState(null);
  const onClick = async () => {
    try {
      const response = await
axios.get('https://jsonplaceholder.typicode.com/photos/1');
      setData(response.data);
    } catch (error) {
      console.log(error)
    }
  }

  return (
    <div className="App">
      <div>
        <button onClick={onClick}>todo 자료불러오기!!</button>
      </div>
      {data && <textarea rows={10} cols={60} value={JSON.stringify(data, null,
2)} readOnly={true} />}
    </div>
  );
}

export default App;
```

14.3 newsapi API키 발급받기

- YOUR KEY : `ca21eeb17c2c474692e03084fd16ec72`





- 전체뉴스 불러오기 : https://newsapi.org/v2/top-headlines?country=kr&apiKey=ca21eeb17c2c474692e03084fd16ec72
- 특정카테고리뉴스 불러오기 : https://newsapi.org/v2/top-headlines?country=kr&category=business&apiKey=ca21eeb17c2c474692e03084fd16ec72
- axios.get(url)을 newsapi로 대체 -> `status: 'ok'`

14.4 News Viewer UI 만들기

- styled-components 설치 : yarn add styled-components
- src/components 폴더 생성
- 해당 폴더에 NewsItem.js, NewsList.js 생성

14.4.1 NewsItem 만들기

src/components/NewsItem.js

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

const NewsItemBlock = styled.div`
  display: flex;

  .thumbnail {
    margin-right: 1rem;
    img {
      display: block;
      width: 160px;
```

```
      height: 100px;
      object-fit: cover;
    }
  }
  .contents {
    h4 {
      margin: 0;
      a {
        color: black;
      }
    }
    p {
      margin: 0;
      line-height: 1.5;
      margin-top: 0.5rem;
      white-space: normal;
    }
  }
  & + & {
    margin-top: 3rem;
  }
`;
const NewsItem = ({ article }) => {
  const { title, description, url, urlToImage } = article;
  return (
    <NewsItemBlock>
      {urlToImage && (
        <div className="thumbnail">
          <a href={url} target="_blank" rel="noopener noreferrer">
            <img src={urlToImage} alt="thumbnail" />
          </a>
        </div>
      )}
      <div className="contents">
        <h4>
          <a href={url} target="_blank" rel="noopener noreferrer">
            {title}
          </a>
        </h4>
        <p>{description}</p>
      </div>
    </NewsItemBlock>
  );
};

export default NewsItem
```

14.4.1 NewsList 만들기

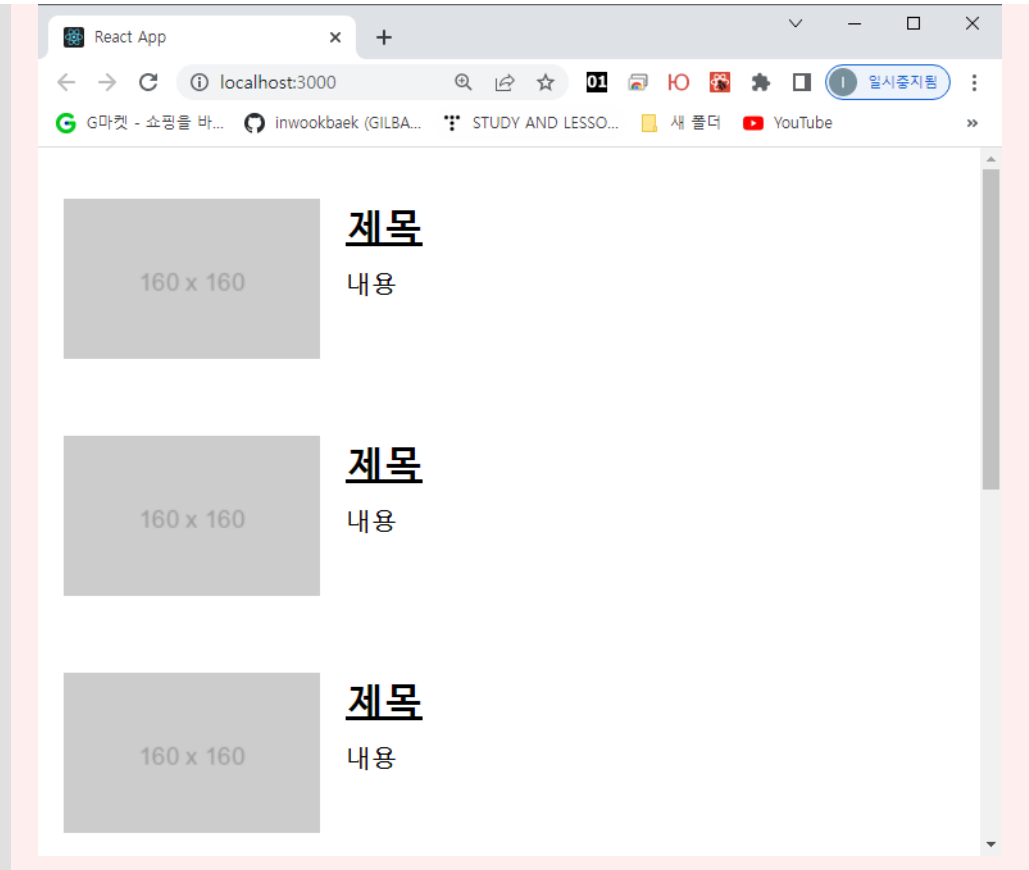
src/components/NewsList.js

```
import styled from 'styled-components';
import NewsItem from './NewsItem';

const NewsListBlock = styled.div`
  box-sizing: border-box;
  padding-bottom: 3rem;
```

```
  width: 768px;
  margin: 0 auto;
  margin-top: 2rem;
  @media screen and (max-width: 768px) {
    width: 100%;
    padding-left: 1rem;
    padding-right: 1rem;
  }
`;

const sampleArticle = {
  title: '제목',
  description: '내용',
  url: 'https://google.com',
  urlToImage: 'https://via.placeholder.com/160'
}

const NewsList = () =>{
  return (
    <NewsListBlock>
      <NewsItem article={sampleArticle} />
      <NewsItem article={sampleArticle} />
      <NewsItem article={sampleArticle} />
      <NewsItem article={sampleArticle} />
      <NewsItem article={sampleArticle} />
      <NewsItem article={sampleArticle} />
    </NewsListBlock>
  );
}

export default NewsList;
```

src/App.js

```
import NewsList from "./components/NewsList";

const App = () => {
  return <NewsList />
}

export default App;
```



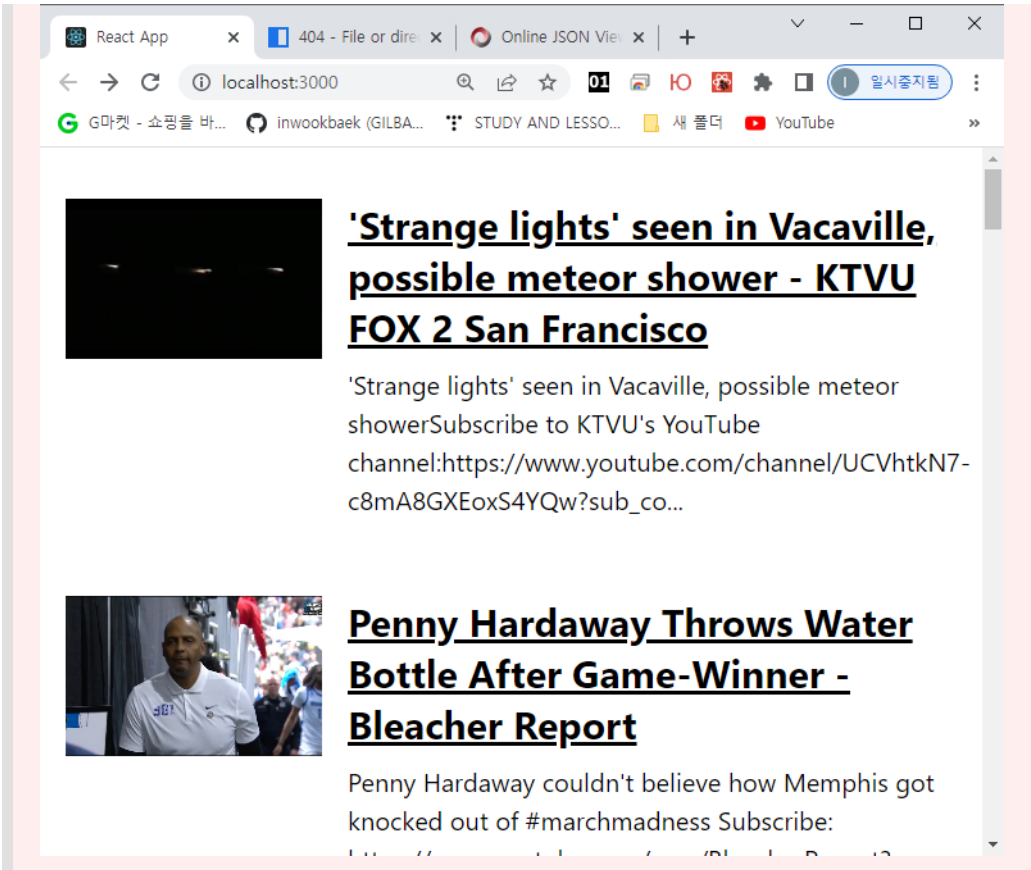
14.5 데이터연동하기

- useEffect hook을 이용하여 처음 랜더링하는 시점에 API를 요청
- 여기서 주의할 점은 useEffect에서 반환하는 값이 뒷정리함수이기 때문에 useEffect에 async로 선언하면 않된다.
- useEffect 내부에서 async/await를 사용하고 싶다면 `함수 내부에 async로 선언된 다른 함수를 만들어서 사용` 해야 한다.
- 또한, 뉴스 데이터배열을 map함수를 사용하기 전에 꼭 `!articles를 조회하여 해당 값의 null여부를 검사` 하지 않으면
- 데이터가 없을 때 렌더링 에러가 발생한다.

components/NewsList.js

```
import { useState, useEffect } from 'react';
import styled from 'styled-components';
import NewsItem from './NewsItem';
import axios from 'axios';

const NewsListBlock = styled.div`
  box-sizing: border-box;
  padding-bottom: 3rem;
  width: 768px;
  margin: 0 auto;
  margin-top: 2rem;
  @media screen and (max-width: 768px) {
    width: 100%;
    padding-left: 1rem;
    padding-right: 1rem;
  }
`;
```

```
const NewsList = () => {

  const [articles, setArticles] = useState(null);
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    // async를 사용하는 함수 선언
    const fetchData = async () => {
      setLoading(true);
      try {
        const response = await axios.get('https://newsapi.org/v2/top-
headlines?country=us&apiKey=ca21eeb17c2c474692e03084fd16ec72');
        setArticles(response.data.articles);
      } catch (error) {
        console.log(error);
      }
      setLoading(false);
    }
    fetchData();
  }, []);

  // 대기 중일 때
  if(loading) {
    return <NewsListBlock>대기 중...</NewsListBlock>
  }

  // articles값이 설정되지 않았을 때
  if(!articles) return null;

  // artticles값이 유효할 때
  return (
    <NewsListBlock>
      {articles.map(aricle => (
        <NewsItem key={aricle.url} article={aricle}/>
      ))}
    </NewsListBlock>
  );
}

export default NewsList;
```



14.6 카테고리 기능 구현

14.6.1 카테고리 선택 UI 만들기

components/Categories.js

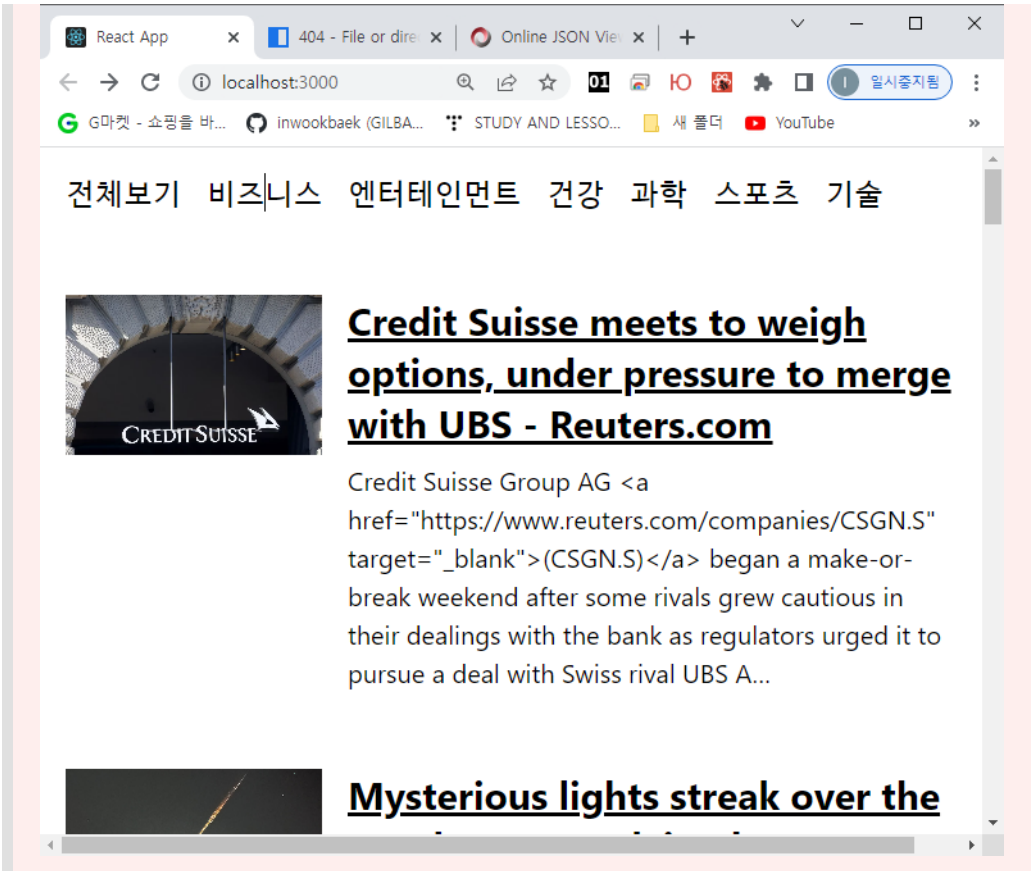
```
import styled from 'styled-components';

const categories = [
  {
    name: 'all',
    text: '전체보기',
  },
  {
    name: 'business',
    text: '비즈니스',
  },
  {
    name: 'entertainment',
    text: '엔터테인먼트',
  },
  {
    name: 'health',
    text: '건강',
  },
  {
    name: 'science',
    text: '과학',
  },
  {
    name: 'sports',
```

```
    text: '스포츠',
  },
  {
    name: 'technology',
    text: '기술',
  },
];

const CategoriesBlock = styled.div`
  display: flex;
  padding: 1rem;
  width: 768px;
  margin: 0 auto;
  @media screen and (max-width: 768px) {
    width: 100%;
    overflow-x: auto;
  }
`;

const Category = styled.div`
  font-size: 1.125rem;
  cursor: pointer;
  white-space: pre;
  text-decoration: none;
  color: inherit;
  padding-bottom: 0.25rem;

  &:hover {
    color: #495057;
  }

  & + & {
    margin-left: 1rem;
  }
`;
const Categories = () => {
  return (
    <CategoriesBlock>
      {categories.map(c => (
        <Category key={c.name}>{c.text}</Category>
      ))}
    </CategoriesBlock>
  );
};

export default Categories;
```

src/App.js

```
import Categories from "./components/Categories";
import NewsList from "./components/NewsList";

const App = () => {

  return (
  <>
    <Categories />
```

```
      <NewsList />
    </>
    );
}

export default App;
```



src/App.js - useState로 category 상태관리

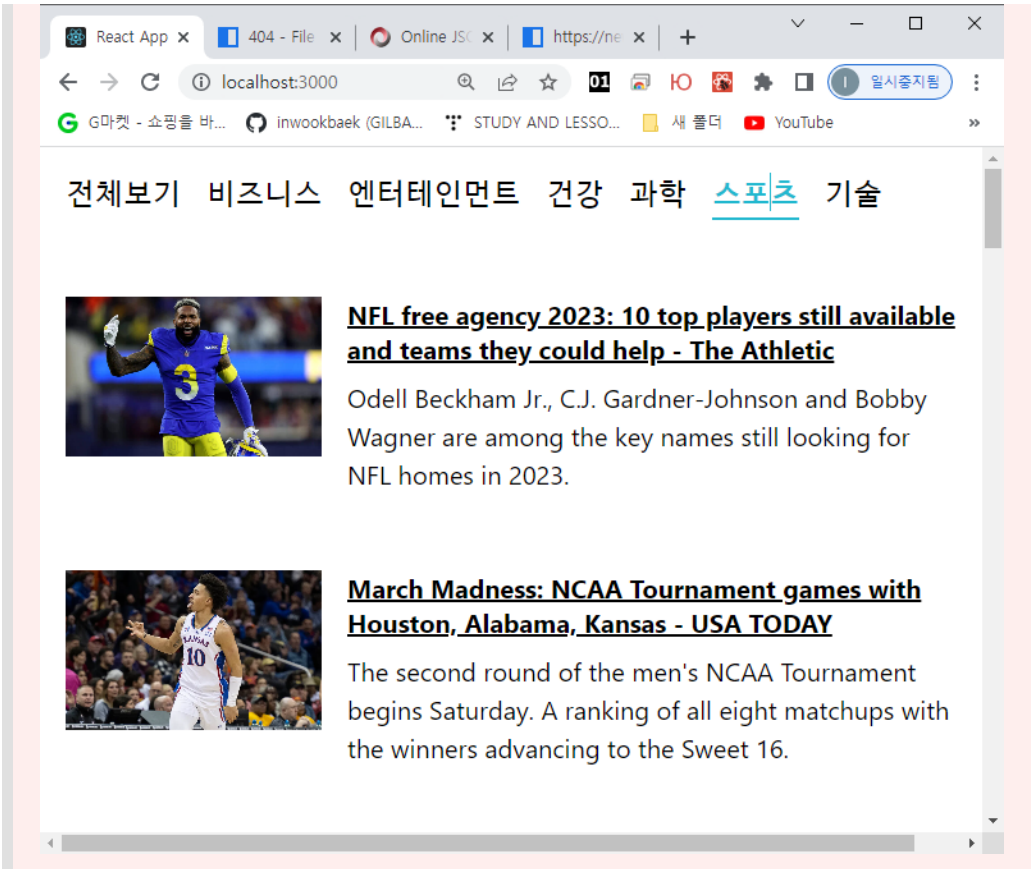
```
import Categories from "./components/Categories";
import NewsList from "./components/NewsList";
import { useState, useCallback } from 'react';

const App = () => {

  const [category, setCategory] = useState('all');
  const onSelect = useCallback(category => setCategory(category), []);

  return (
  <>
    <Categories category={category} onSelect={onSelect}/>
    <NewsList category={category}/>
  </>
  );
}

export default App;
```

components/Categories.js

- category값이 변경될 때 마다 새로운 news를 조회하기 위해서 useEffect의 의존 배열(두 번째 파라미터)에 category를 넣어 주어야 한다.

```
import styled, { css } from 'styled-components';

const categories = [
  {
    name: 'all',
    text: '전체보기',
  },
  {
    name: 'business',
    text: '비즈니스',
  },
  {
    name: 'entertainment',
    text: '엔터테인먼트',
  },
  {
    name: 'health',
    text: '건강',
  },
  {
    name: 'science',
    text: '과학',
  },
  {
    name: 'sports',
    text: '스포츠',
  },
  {
    name: 'technology',
    text: '기술',
  },
];

const CategoriesBlock = styled.div`
  display: flex;
  padding: 1rem;
  width: 768px;
  margin: 0 auto;
  @media screen and (max-width: 768px) {
    width: 100%;
    overflow-x: auto;
  }
`;

const Category = styled.div`
  font-size: 1.125rem;
  cursor: pointer;
  white-space: pre;
  text-decoration: none;
  color: inherit;
  padding-bottom: 0.25rem;

  &:hover {
    color: #495057;
  }

  ${props =>
```

```
      props.active && css`
      font-weight: 600;
      border-bottom: 2px solid #22b8cf;
      color: #22b8cf;
      &:hover {
        color: #3bc9db;
      }
    `}

    & + & {
      margin-left: 1rem;
    }
  `;

const Categories = ({onSelect, category}) => {
  return (
    <CategoriesBlock>
      {categories.map(c => (
        <Category
          key={c.name}
          active={category === c.name}
          onClick={() => onSelect(c.name)}
        >{c.text}</Category>
      ))}
    </CategoriesBlock>
  );
};

export default Categories;
```



14.7 react-router 적용하기

14.7.1 react-router 설치 및 적용

- 설치 : `yarn add react-router-dom`

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { BrowserRouter } from 'react-router-dom';


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);
```

14.7.2 NewsPage 생성

- pages폴더 생성 및 NewsPage.js 생성

- src/pages/NewsPage.js

```
import { useParams } from 'react-router-dom';
import Categories from '../components/Categories';
import NewsList from './../components/NewsList';

const NewsPage = () => {

  const params = useParams();

  // category가 선택되지 않았다면 기본값 'all'
  const category = params.category || 'all';

  return (
    <>
      <Categories />
      <NewsList category={category} />
    </>
  )
}

export default NewsPage
```

src/app.js

- category를 URL의 파라미터로 전달할 것이기 때무네 Categories 컴퍼넌트에서 category값과 onSelect함수를 전달할 필요가 없다.

```
import { Route, Routes } from 'react-router-dom';
import NewsPage from "./pages/NewsPage";

const App = () => {
```

```
  return (
    <Routes>
      <Route path="/" element={ <NewsPage />} />
      <Route path="/:category" element={ <NewsPage />} />
    </Routes>
  );
}

export default App;
```

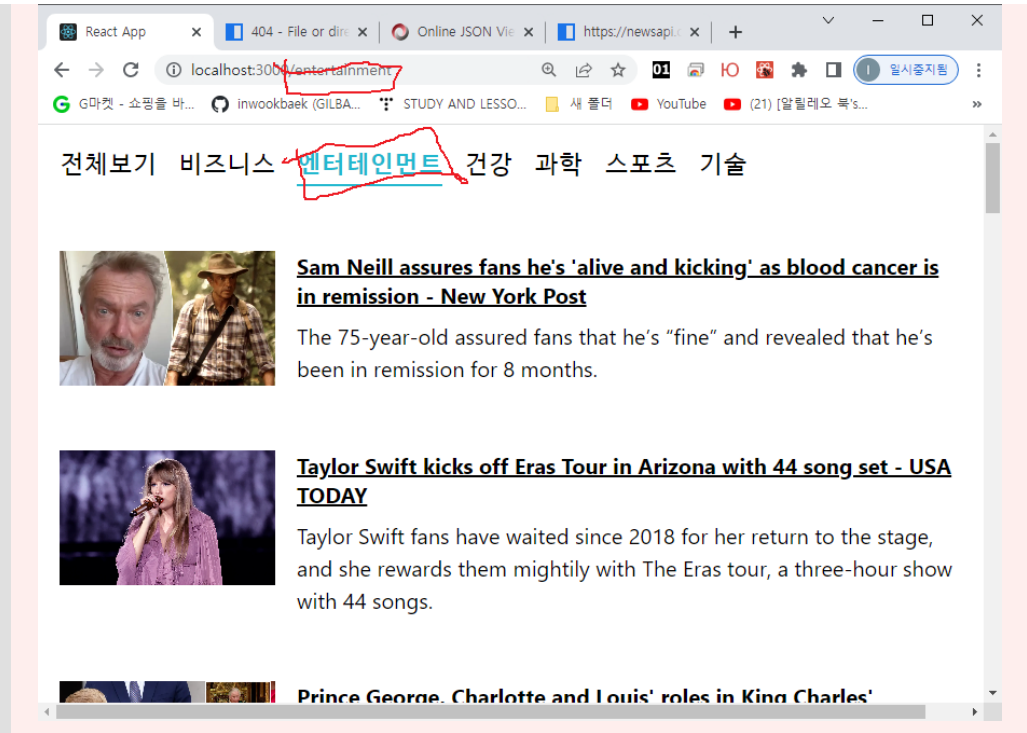
14.7.3 Categories에서 NavLink 사용하기

- Categories에서 onSelect함수로 카테고리를 선택하고, 다른 스타일을 주는 기능을 NavLink로 대체
- `HTML요소가 아닌 특정 컴퍼넌트에서 styled-components를 사용할 때는 'styled(컴퍼넌트이름)``'을 사용`

components/Categories.js

```
import styled from 'styled-components';
import { NavLink } from 'react-router-dom';

const categories = [
  {
    name: 'all',
    text: '전체보기',
  },
  {
    name: 'business',
    text: '비즈니스',
  },
  {
    name: 'entertainment',
    text: '엔터테인먼트',
  },
  {
    name: 'health',
    text: '건강',
  },
  {
    name: 'science',
    text: '과학',
  },
  {
    name: 'sports',
    text: '스포츠',
  },
  {
    name: 'technology',
    text: '기술',
  },
];

const CategoriesBlock = styled.div`
```

```
  display: flex;
  padding: 1rem;
  width: 768px;
  margin: 0 auto;
  @media screen and (max-width: 768px) {
    width: 100%;
    overflow-x: auto;
  }
`;

const Category = styled(NavLink)`
  font-size: 1.125rem;
  cursor: pointer;
  white-space: pre;
  text-decoration: none;
  color: inherit;
  padding-bottom: 0.25rem;

  &:hover {
    font-weight: 600;

  }

  &.active {
    font-weight: 600;
    border-bottom: 2px solid #22b8cf;
    color: #22b8cf;
    &:hover {
      color: #3bc9db;
    }
  }

  & + & {
    margin-left: 1rem;
  }
`;

const Categories = ({onSelect}) => {
  return (
    <CategoriesBlock>
      {categories.map(c => (
        <Category
          key={c.name}
          className={({isActive}) => (isActive ? 'active' : undefined)}
          to={c.name === 'all' ? '/' : `/${c.name}`}
        >{c.text}
      </Category>
      ))}
    </CategoriesBlock>
  );
};

export default Categories;
```



14.8 usePromise 커스텀 HOOK 만들기

- 컴퍼넌트에서 API호출처럼 Promise를 사용하는 경우 간결한 코드작성을 위해 커스텀HOOK을 작성
- `usePromise라는 이름으로 커스텀 HOOK을 작성` - src/lib/usePromise.js 생성
- 유틸함수는 보토 src/lib안에 작성하는 것이 관례
- `usePromise 커스텀HOOK은 대기중, 완료, 에러에 대한 상태관리 하며 의존배열인 deps를 파라미터로 받아 온다`
- deps배열은 의존배열로 설정되는 데 `ESLint의 경고메시지`가 나타난다.
- 이 경고를 무시하려면 `특정 줄에서만 ESLint 규칙을 무시하도록 주석을 작성`해야 한다.
  - 초록색 경고줄이 있을 경우 커서를 올리면 QuickFix클릭후 `Disable ... this line`을 선택하면
  - `// eslint-disable-next-line react-hooks/exhaustive-deps` 처럼 주석처리

src/lib/usePromise.js

```
import { useState, useEffect } from 'react';

export default function usePromise(promiseCreator, deps) {

  // 대기중/완료/실패에 대한 상태관리
  const [loading, setLoading] = useState(false);
  const [resolved, setResolved] = useState(null);
  const [error, setError] = useState(null);

  useEffect(() => {
    const process = async () => {
      setLoading(true);
      try {
        const resolved = await promiseCreator();
        setResolved(resolved);
```

```
      } catch (e) {
        setError(e);
      }
      setLoading(false);
    };
    process();
  // eslint-disable-next-line react-hooks/exhaustive-deps
  }, deps);

  return [loading, resolved, error];
}
```

usePromise을 NewsList컴퍼넌트에 적용하기

- `useState, useEffect대신에 usePromise 적용`

components/NewsList.js

```
import styled from 'styled-components';
import NewsItem from './NewsItem';
import axios from 'axios';
import usePromise from '../lib/usePromise';

const NewsListBlock = styled.div`
  box-sizing: border-box;
  padding-bottom: 3rem;
  width: 768px;
  margin: 0 auto;
  margin-top: 2rem;
  @media screen and (max-width: 768px) {
    width: 100%;
    padding-left: 1rem;
    padding-right: 1rem;
  }
`;

const NewsList = ({ category }) => {

  const [loading, response, error] = usePromise(() => {
    const query = category === 'all' ? '' : `&category=${category}`;
    return axios.get(`https://newsapi.org/v2/top-headlines?
country=us${query}&apiKey=ca21eeb17c2c474692e03084fd16ec72`);
  }, [category]);

  // 대기 중일 때
  if(loading) {
    return <NewsListBlock>대기 중...</NewsListBlock>
  }

  // response 값이 설정되지 않았을 때
  if(!response) return null;

  if(error) return <NewsListBlock>에러발생!!!</NewsListBlock>

  // response 값이 유효할 때
  const { articles } = response.data;
```

```
  return (

    <NewsListBlock>
      {articles.map(aricle => (
        <NewsItem key={aricle.url} article={aricle}/>
      ))}
    </NewsListBlock>
  );
}

export default NewsList;
```

- usePromise를 적용하면 상태관리와 useEffect설정을 하지 않아도 되므로 코드가 훨씬 간결해 진다.
- 무조건 custom hook을 만들어서 사용해야 하는 것은 아니나 적절히 사용하면 좋은 코드를 작성할 수 있다.