6. Component Iteration

6.1 map()

- 자바스크립트 배열객체의 내장함수 `map()함수를 사용하여 반복되는 컴퍼넌트를 랜더링 할 수 있다.`
- `map함수는 전달될 함ㅅ를 사용해서 배열 내 각 요소를 원하는 규칙으로 변환후 새로운 배열을 생성` 한다.

6.1.1 문법

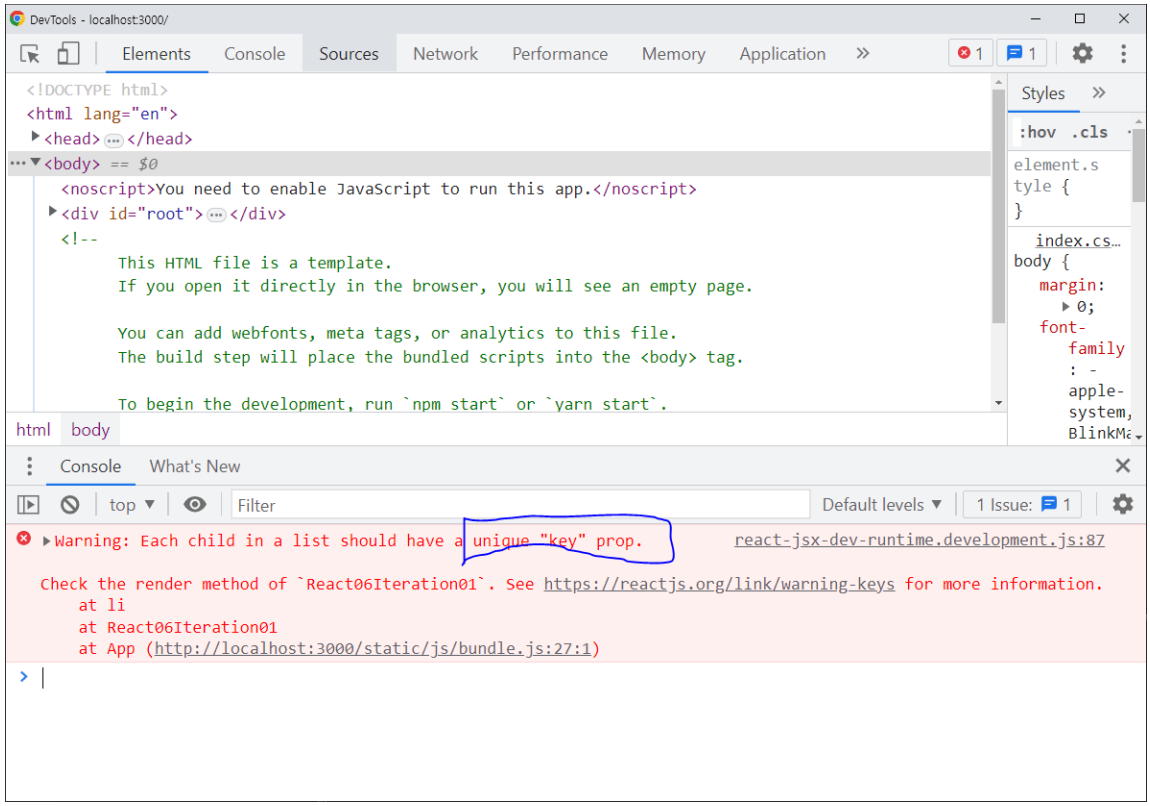
> `배열.map(callback, [thisArg])` 이 함수의 파라미터
>
> > - callbaek : 새로운 배열의 요소를 생서하는 함수, 3개의 파라미터를 갖는다.
> >   - currentValue : 현재 처리하고 있는 요소
> >   - index : 현재 처리하는 요소의 index
> >   - array : 현재 처리하고 있는 원본배열
> > - thisArg(선택) : callback함수 내부에서 사용할 this 래퍼런스

6.2 배열을 컴퍼넌트배열로 변환하기

```
src/App.js
import React, { Component } from 'react';
import React06Iteration01 from './mysrc/React06Iteration01';

class App extends Component {

  render() {
    return (
      <>
        <React06Iteration01 />
        <hr/>
      </>
    );
  }
}
export default App;

src/mysrc/React06Iteration01.js
const React06Iteration01 = () => {

  const names =['🥝','🍇','🍏','🍐','🍑','🍒','🍓','🍅'];
  const nameList = names.map(name => <li>{name}</li>);
    return (
      <ul>
        {nameList}
```

```
        </ul>
      );
    }

    export default React06Iteration01;
```

key가 없다는 경고메시지



6.3 key

- 리액트에서 key는 컴퍼넌트 배열을 랜더링했을 때 어떤 원소에 변동이 있었는지 알아낼 때 사용한다.
- 원소의 생성, 삭제, 추가등 작업을 할 때 key가 없을 때는 Virtual DOM을 비교하는 과정에서 순차적으로 비교한다.
- 하지만, key가 있을 경우 더욱 빠르게 알아낼 수가 있게 된다.
- key값을 알아낼 경우, `map()`함수에 전달되는 콜백삼수의 인수인 `index`값을 사용하면 된다.
- `index`를 `key`로 전달하면 더이상 경고메시지는 나타나지 않는다.
- 고유한 값이 없을 때만 index값을 key로 사용해야 한다. `index`를 `key`로 사용하면 배열이 변경될 때 효율적으로 리랜더링을 하지 못한다.

```
src/App.js
import React, { Component } from 'react';
import React06Iteration01 from './mysrc/React06Iteration01';
import React06Iteration02 from './mysrc/React06Iteration02';

class App extends Component {

  render() {
    return (
      <>
```

```
        <React06Iteration01 />
        <hr/>
        <React06Iteration02 />
        <hr/>
      </>
    );
  }
}
export default App;
```

src/mysrc/React06Iteration02.js

```
const React06Iteration02 = () => {

  const names =['🥝','🍇','🍏','🍋','🍑','🍒','🍓','🍅'];
  const nameList = names.map((name,index) => <li key={index}>
{name}</li>);
    return (
      <ul>
        {nameList}
      </ul>
    );
}

export default React06Iteration02;
```

6.4 동적배열 랜더링

6.4.1 CRUD

- 리액트에서 상태를 업데이트할 때는 기존 상태를 그대로 두면서 새로운 값을 상태로 설정해야 한다.
- 이를 `불변성 유지`라고 한다. 불변성 유지를 해 주어야 나중에 `리액트 컴퍼넌트의 성능을 최저화` 할 수 있다.
- 불변성을 유지하면서 `배열의 특정항목을 지울 때는 배열의 내장함수 filter를 사용` 한다.

src/mysrc/React06Iteration03.js

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

const React06Iteration03 = () => {

  const [names, setNames] = useState([
    {id: 1, text: '🍑'},
    {id: 2, text: '🍒'},
    {id: 3, text: '🍓'},
    {id: 4, text: '🍅'}]);

  const [inputText, setInputText] = useState('');
  const [nextId, setNextId] = useState(5);

  const onChange = e => setInputText(e.target.value);

  const onClick = () => {
    // 배열을 추가할 때 push를 사용하지 않고 concat를 사용
```

```
    // push는 기존 배열을 변경, concat는 새로운 배열을 생성
    const nextNames = names.concat({
      id: nextId,
      text: inputText
    });
    setNextId(nextId + 1);
    setNames(nextNames);
    setInputText('');
  }

  const onRemove = id => {
    const nextNames = names.filter(name => name.id !== id);
    setNames(nextNames);
  }

  const nameList = names.map(name => (
      <li key={name.id} onDoubleClick={() => onRemove(name.id)}>
{name.text}</li>));
    return (
      <ul>
        <input value={inputText} onChange={onChange} />
        <button onClick={onClick}>추가</button>
        <ul>{nameList}</ul>
      </ul>
    );
}

export default React06Iteration03;
```

6.5 정리

- 컴퍼넌트 배열을 랜더링할 때는 key값 설정에 주의해야 한다. 또 key값은 언제나 유일해야 한다.
- key값이 중복되면 랜더링과정에서 오류가 발생한다.
- 상태안에서 배열을 변경할 때는 배열에 직접 접근하여 수정하는 것이 아니라 `concat, filter등의 배열 내장함수를 사용` 하여
- 새로운 배열을 만든 후에 이를 새로운 상태로 설정해 주어야 한다.