

2일차

React

컴포넌트 이해

발표자 : 노나연

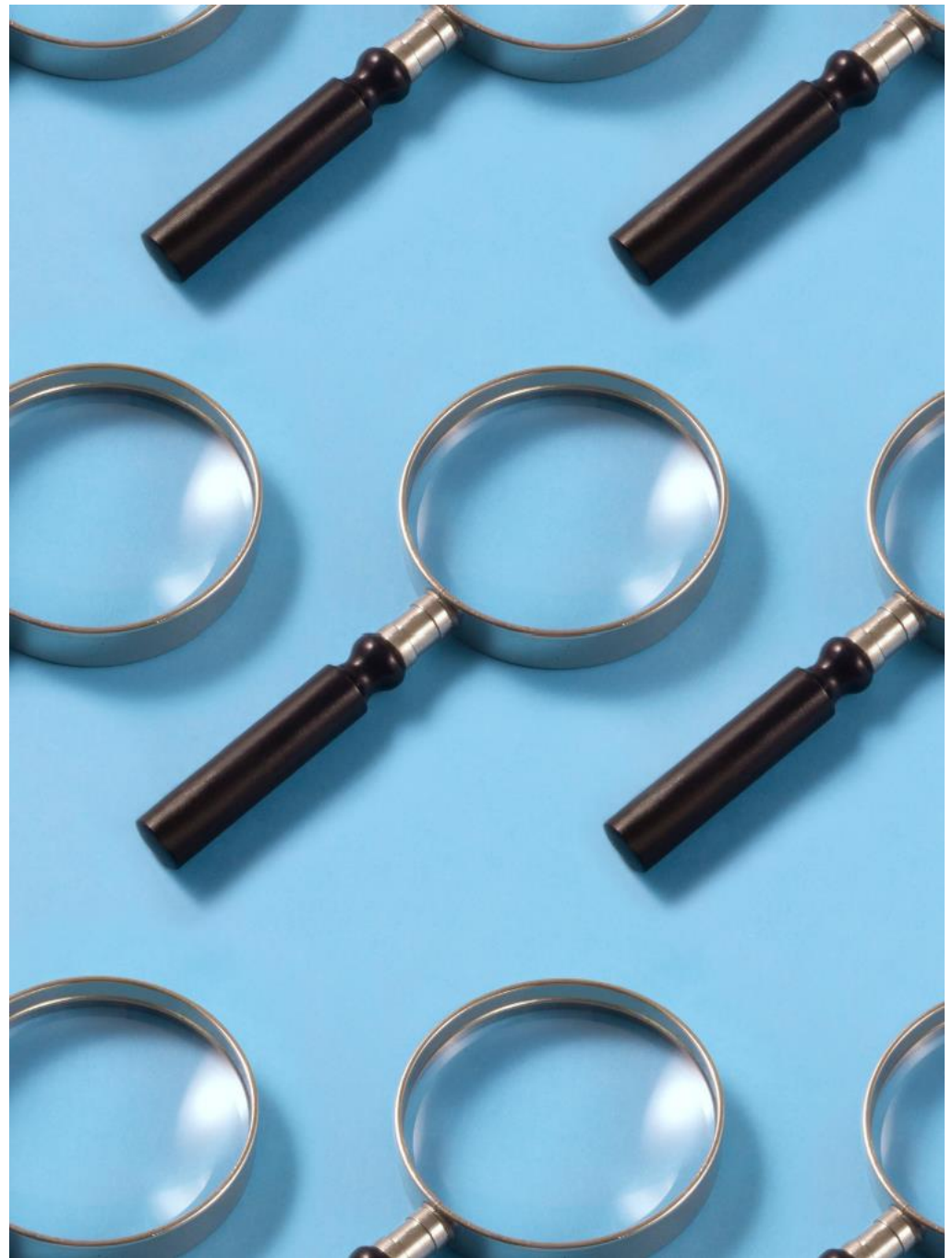
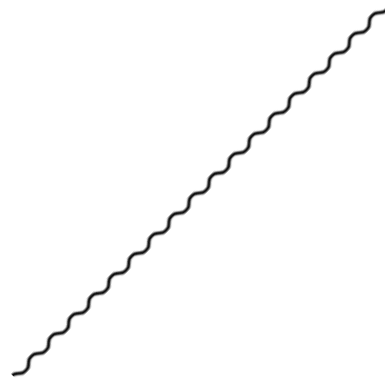


Table of
Contents



목차



- 01. 컴포넌트
- 02. 컴포넌트의 종류 (클래스/ 함수)
- 03. 컴포넌트 설계
- 04. 실습

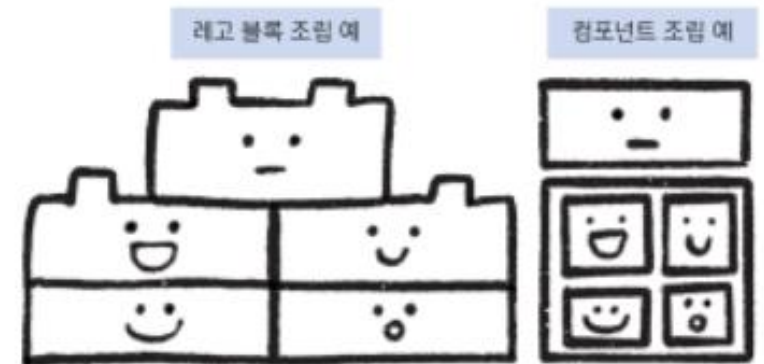
교육 준비

- <https://github.com/nkiateam/react-component-tutorial>
- git 소스 다운 받고 싶은 폴더 선택
- `git clone` <https://github.com/nkiateam/react-component-tutorial>
- `cd react-component-tutorial`
- `npm install`

- 혹시 `git`을 설치하지 않으신 분들은 꼭 설치해주세요!
- <https://git-scm.com/>

01. 컴포넌트란

- 독립적이고 재사용 가능하도록 UI를 나눈 조각
- 컴포넌트는 레고 블록에 비유할 수 있다.
- 컴포넌트 개념을 잘 적용한 소프트웨어란 부품만 바꾸어 주었을 시, 오류 없이 잘 작동 되는 것을 의미합니다.



컴포넌트 예제

- NAVER 웹툰 페이지

장르별 추천웹툰 >

에피소드 오니버스 스토리 | 일상 **개그** 판타지 액션 드라마 순정 감성 스릴러 시대극 스포츠

인기순 **업데이트순** 조회순 별점순



모쪼의 일지 **웹툰**
121화. 집순이 연구 보고서
모쪼



후덜덜덜 남극전자
47화 아빠 펭귄도 아랫배로 알을 품는다
김민혁



가슴털 로맨스
54화
가오모



오늘의 순정망화
시즌2 12화
손하기



갑자기 커피
15화 전설의 맛집
우 / 리지



만물의 영장
52화 엄마 X 가를 운동회 <끝>
보민

컴포넌트 한 개의 코드

```

▼<div class="genreRecomImg2">
  ▼<a onclick="nclk_v2(event,'rcn*g.thum','728015','1')" href="/webtoon/list.nhn?titleId=728015">
    <span class="mask"></span>
    
  </a>
</div>
▼<div class="genreRecomInfo2">
  ▼<h6 class="title">
    <span class="ico_cut">컷툰</span>
    ▼<a onclick="nclk_v2(event,'rcn*g.title','728015','1')" href="/webtoon/list.nhn?titleId=728015">
      <span>모쥼의 일지</span>
    </a>
  </h6>
  ▼<p>
    ▼<a onclick="nclk_v2(event,'rcn*g.sub','728015','1')" href="/webtoon/detail.nhn?titleId=728015&no=121">
      <span>121화. 집순이 연구 보고서</span>
    </a>
  </p>
  ▼<span class="user">
    <a href="#" onclick="return goArtist('728015', '1', this, event);">

      모쥼

    </a>
  </span>
</div>

```

6개를 나열한다면?

굉장히 깊니다...

컴포넌트 한 개의 코드

```

▼<div class="genreRecomImg2">
  ▼<a onclick="nclk_v2(event,'rcn*g.thum','728015','1')" href="/webtoon/list.nhn?titleId=728015">
    <span class="mask"></span>
    
  </a>
</div>
▼<div class="genreRecomInfo2">
  ▼<h6 class="title">
    <span class="ico_cut">컷툰</span>
    ▼<a onclick="nclk_v2(event,'rcn*g.title','728015','1')" href="/webtoon/list.nhn?titleId=728015">
      <span>모조의 일지</span>
    </a>
  </h6>
  ▼<p>
    ▼<a onclick="nclk_v2(event,'rcn*g.sub','728015','1')" href="/webtoon/detail.nhn?titleId=728015&no=121">
      <span>121화. 집순이 연구 보고서</span>
    </a>
  </p>
  ▼<span class="user">
    <a href="#" onclick="return goArtist('728015', '1', this, event);">
      모조
    </a>
  </span>
</div>

```

props state

코드로 본다면..

```
import React, {Component} from 'react';
import CardItem from './CardItem';

class Webtoon extends Component {
  render() {
    return (
      <div>
        <ul>
          <li><CardItem title="모쪼의 일지" subTitle="121화 집순이 연구 보고서" writer="모쪼" /></li>
          <li><CardItem title="후덜덜덜 남극전자" subTitle="47화 아빠 펭귄도 아랫배로 알을 품는다" writer="김민혁" /></li>
          <li><CardItem title="가슴털 로맨스" subTitle="54화" writer="갸오오" /></li>
          <li><CardItem title="오늘의 순정만화" subTitle="시즌2 12화" writer="손하기" /></li>
          <li><CardItem title="갑자기 커피" subTitle="15화 전설의 맛집" writer="우/리지" /></li>
          <li><CardItem title="만물의 영장" subTitle="52화 엄마 X 가을 운동회<끝>" writer="보민" /></li>
        </ul>
      </div>
    );
  }
}
```

더 간단히 쓴다면..

```
class Webtoon extends Component {  
  render() {  
    const { list } = this.props;  
    return (  
      <div>  
        <ul>  
          {list.map(({ title, subTitle, writer }) =>(  
            <li>  
              <CardItem title={title} subTitle={subTitle} writer={writer}/>  
            </li>  
          ))}  
        </ul>  
      </div>  
    );  
  }  
}
```

장르별 추천웹툰 >

에피소드 음니버스 스토리 | 일상 **개그** 판타지 액션 드라마 순정 감성 스릴러 시대극 스포츠

인기순 업데이트순 조회순 별점순



모조의 일지 **첫화**
121화. 집순이 연구 보고서
모조



후덜덜달 남극전자
47화 아빠 펭귄도 아랫배로 알을 품는다
김민혁



가슴털 로망스
54화
가오모



오늘의 순정망화
시즌2 12화
손하기



갑자기 커피
15화 전설의 맛집
우 / 리지



만물의 엉장
52화 엄마 X 가을 운동회<끝>
보민

컴포넌트의 장점

- 가독성 : 복잡한 코드를 한 눈에 볼 수 있다.
- 재사용성 : 필요한 부분에 재사용이 가능하다.
- 유지보수가 용이하다.

02. 컴포넌트의 종류

- 클래스형 컴포넌트

```
1  class App extends React.Component {  
2      render() {  
3          return (  
4              <div>  
5                  <h2>클래스 컴포넌트</h2>  
6              </div>  
7          )  
8      }  
9  }  
10
```

- Render 함수가 반드시 존재해야 한다.
- State의 사용이 가능하다.
- 라이프 사이클 API의 사용이 가능하다.

02. 컴포넌트의 종류

- 함수형 컴포넌트 (Stateless Functional Component)

```
1 function App(props) {  
2   return (  
3     <div>  
4       <h2>함수형 컴포넌트</h2>  
5     </div>  
6   );  
7 }  
8  
9
```

- 클래스형 컴포넌트에 비해 선언하기가 편리하다.
- 메모리 자원을 클래스형 컴포넌트에 비해 덜 차지한다.
- 빌드 후 배포시에 결과물의 크기가 작다.

But,

React 의 버전 업데이트(v.16.8) 이후 Hooks라는 기능을 통해 상태관리가 가능해지게 되었습니다.

(Hooks은 뒤에서!)

03. 컴포넌트 설계

- <https://ko.reactjs.org/docs/thinking-in-react.html>
- UI를 컴포넌트 계층 구조로 나누기
 - 단일 책임의 원칙
- props / state 찾기
 - 최소한의 state 찾기
- 이벤트 핸들러 추가하기

컴포넌트 계층구조로 나누기

NKIA-TODOLIST

TodosCompleted

☐ 2일차 컴포넌트 설계Remove

☐ 컴포넌트 란?Remove

☐ 클래스형 컴포넌트와 함수형 컴포넌트Remove

☐ 함수형 컴포넌트에서 쓰이는 hooksRemove

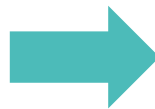
☐ HOC로 컴포넌트 재사용하기Remove

☐ 성능 최적화Remove

☐ 그외Remove



Container 컴포넌트



Header 컴포넌트



Contents 컴포넌트



List 컴포넌트

컴포넌트 역할 정의

- Container 컴포넌트 : 전체 영역입니다.
- Header 컴포넌트 : header 영역입니다.
 - Title: todoList 제목을 보여줍니다.
 - Input: todoList 내용을 입력합니다.
- Contents 컴포넌트 : Tab을 분리하여 TodoList 내용을 보여줍니다.
 - List 컴포넌트 : List 한 줄을 구성합니다.
 - checkbox : 체크 기능을 제공합니다.
 - Button : 삭제 버튼

실습 시간

- <https://github.com/nkiateam/react-component-tutorial>
- git 소스 다운 받고 싶은 폴더 선택
- `git clone` <https://github.com/nkiateam/react-component-tutorial>
- `cd react-component-tutorial`
- `npm install`

아직 다운받지 않은 분이 있다면, 지금 다운받아주세요~

실습 전에 미리 알아두어야 할 것들

1. 이벤트 핸들러

- onChange
- onClick
- onKeyUp

2. 기본문법

- Map
- Filter

1. 이벤트 핸들러

- 이벤트 핸들러
 - onChange : 변화가 일어났을 때 발생합니다.
 - onKeyUp : 키보드 이벤트, 키를 눌렀다 뗐을 때 발생합니다.
 - onClick : 클릭했을 때 발생합니다.
- <https://ant.design/docs/react/introduce>

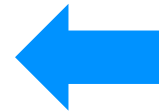
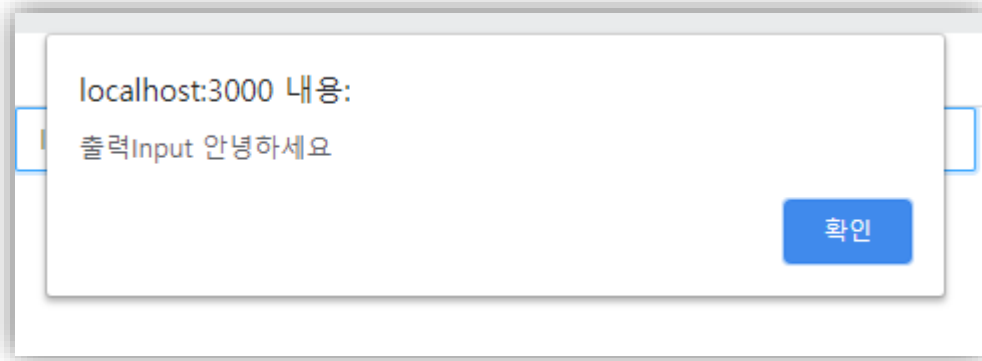
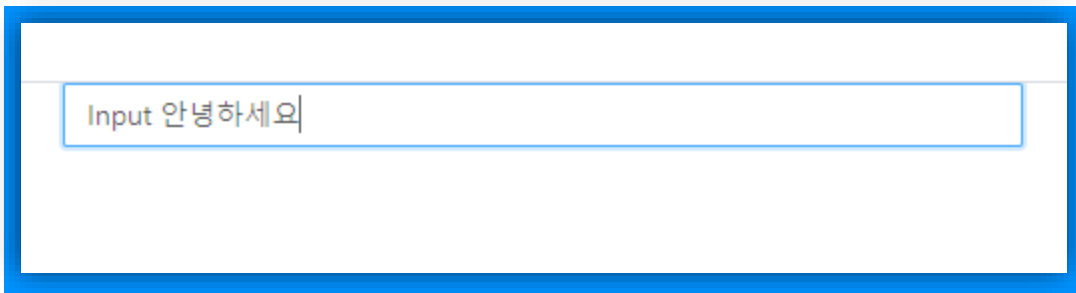
API

To get a customized button, just set `type` / `shape` / `size` / `loading` / `disabled` .

Property	Description	Type	Default	Version
disabled	disabled state of button	boolean	false	
ghost	make background transparent and invert text and border colors	boolean	false	
href	redirect url of link button	string	-	
htmlType	set the original html <code>type</code> of <code>button</code> , see: MDN	string	<code>button</code>	

InputBox 예제

- src/todo/01_todoList/01_input/InputBox.js



- 터미널에서 : npm start

```
class InputBox extends Component {
  state = {
    text : '',
  };
  onChangeText = (e) => {
    this.setState({
      text : e.target.value,
    })
  };
  render() {
    return (
      <div className="center">
        <Input
          placeholder="입력하세요"
          onChange={e => this.onChangeText(e)}
          onKeyUp={e => {
            if(e.keyCode === 13){
              alert('출력'+this.state.text);
            }
          }}
        />
      </div>
    );
  }
}
```

2. 기본 문법

- `map` 메서드는 배열 내의 모든 요소 각각에 대하여 주어진 함수를 호출한 결과를 모아 새로운 배열을 반환합니다.

```
1 const array1 = [1, 4, 9, 16];  
2  
3 // pass a function to map  
4 const map1 = array1.map(x => x * 2);  
5 |  
6 console.log(map1);  
7 // expected output: Array [2, 8, 18, 32]  
8
```

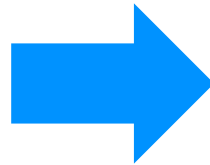
Map 예제

- src/todo/01_todoList/02_map/MapList.js

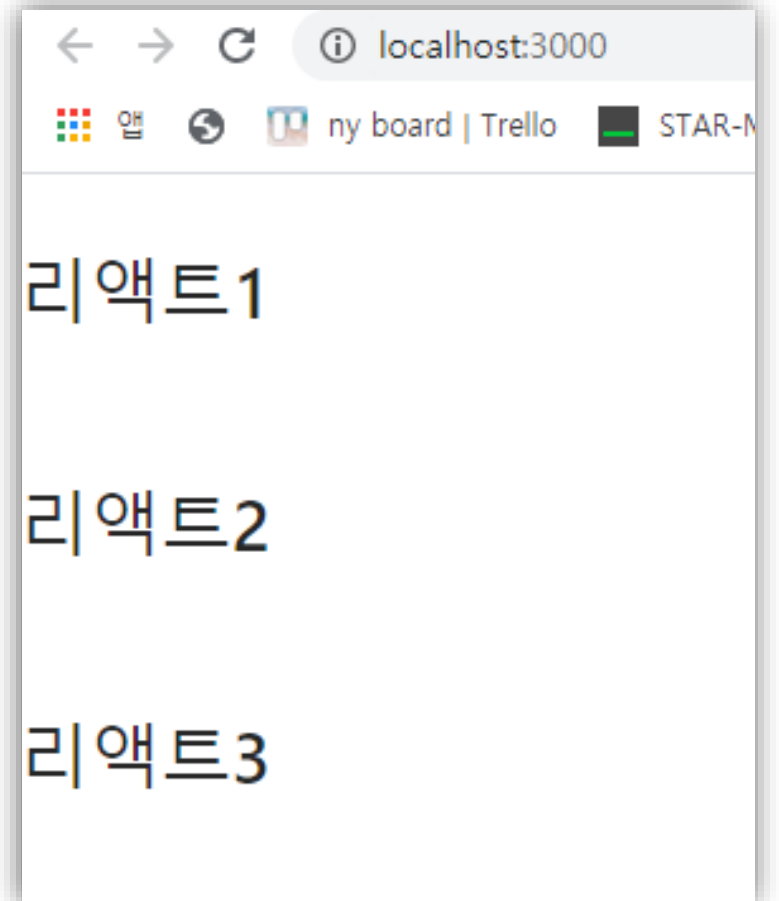
```
import React, {Component} from 'react';
let list = [];
list.push({ index:1, text:'리액트1'});
list.push({ index:2, text:'리액트2'});
list.push({ index:3, text:'리액트3'});
```

```
class MapList extends Component {

  render() {
    return (
      <div>
        {list.map((val) => (
          <h1>{val.text}</h1>
        ))}
      </div>
    );
  }
}
```



```
class App extends Component {
  render () {
    return (
      // <InputBox />
      <MapList/>
      // <Fillter />
      // <ContainerComponent />
      // <Main />
      // <Counter />
    )
  }
}
```



2. 기본 문법

- `filter` 메서드는 주어진 함수의 테스트를 통과하는 모든 요소를 모아 새로운 배열로 반환합니다.

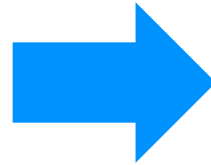
```
1 const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];
2
3 const result = words.filter(word => word.length > 6);
4
5 console.log(result);
6 // expected output: Array ["exuberant", "destruction", "present"]
7
```

Filter 예제

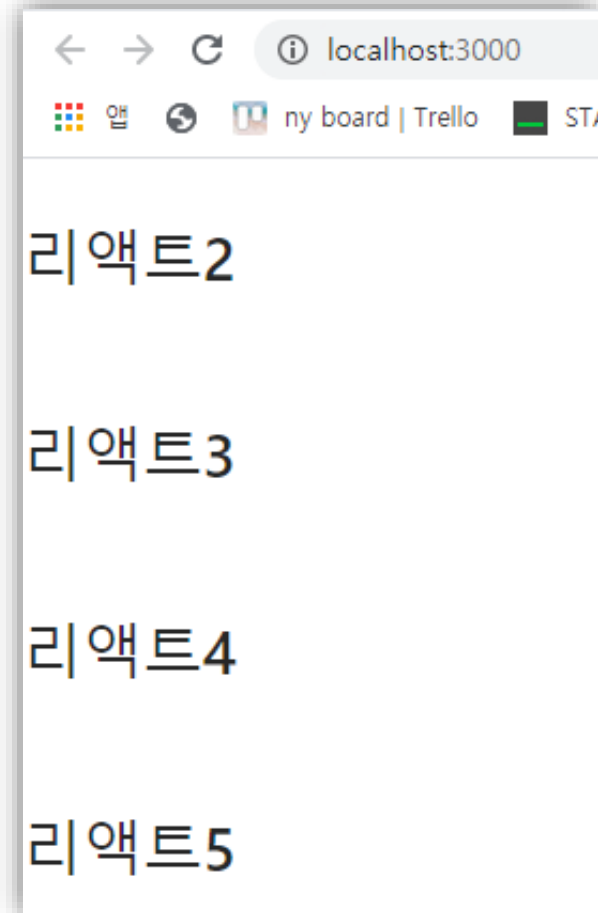
- src/todo/01_todoList/03_fillter/Fillter.js

```
import React, {Component} from 'react';
let list = [];
list.push({ index:1, text:'리액트1'});
list.push({ index:2, text:'리액트2'});
list.push({ index:3, text:'리액트3'});
list.push({ index:4, text:'리액트4'});
list.push({ index:5, text:'리액트5'});

class Fillter extends Component {
  render() {
    return (
      <div>
        {list
          .filter(num => num.index > 1)
          .map(val =>(
            <h1>{val.text}</h1>
          ))
        }
      </div>
    );
  }
}
```



```
class App extends Component {
  render () {
    return (
      // <InputBox />
      // <MapList/>
      <Fillter />
      // <ContainerComponent />
      // <Main />
      // <Counter />
    )
  }
}
```



컴포넌트 역할 정의

- Container 컴포넌트 : 전체 영역입니다.
- Header 컴포넌트 : header 영역입니다.
 - Title: todoList 제목을 보여줍니다.
 - Input: todoList 내용을 입력합니다.
- Contents 컴포넌트 : Tab을 분리하여 TodoList 내용을 보여줍니다.
 - List 컴포넌트 : List 한 줄을 구성합니다.
 - checkbox : 체크 기능을 제공합니다.
 - Button : 삭제 버튼

구조잡기

- src/todo/01_todoList/ContainerComponent.js

```
class ContainerComponent extends Component {  
  
  render() {  
    return (  
      <div className="APP">  
        <div className="wrapper">  
          <HeaderComponent/>  
          <ContentsComponent/>  
        </div>  
      </div>  
    );  
  }  
}  
  
export default ContainerComponent;
```

```
.App {  
  font-family: sans-serif;  
}  
  
.wrapper {  
  margin: auto;  
  max-width: 480px;  
  width: calc(100vw - 40px);  
}
```

HeaderComponent

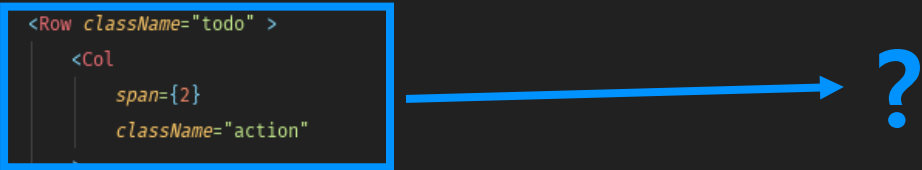
```
class HeaderComponent extends Component {  
  render(){  
    const { onChangeText, onSubmit, text } = this.props;  
    return(  
      <div>  
        <h1>NKIA-TODOLIST</h1>  
        <div id="add">  
          <Input  
            placeholder="입력하세요"  
            value={text}  
            onChange={onChangeText}  
            onKeyDown={e => {  
              if(e.keyCode === 13){  
                onSubmit()  
              }  
            }}  
          />  
        </div>  
      </div>  
    )  
  }  
}
```

- 역할

- 타이틀을 보여줍니다.
- 내용을 입력합니다.

ListComponent

```
class ListComponent extends Component {  
  render() {  
    const { todo, onDelete, onChangeCheckbox } = this.props;  
    return (  
      <Row className="todo" >  
        <Col  
          span={2}  
          className="action"  
        >  
          <Checkbox  
            defaultChecked={todo.complete}  
            onChange={() => onChangeCheckbox(todo.index)}  
          />  
        </Col>  
        <Col span={18}>  
          <span style={{ textDecoration : todo.complete ? "line-through" : "none"}}>  
            {todo.text}  
          </span>  
        </Col>  
        <Col  
          span={4}  
          className="action"  
        >  
          <Button type="link" onClick={() => onDelete(todo.index)}>  
            Remove  
          </Button>  
        </Col>  
      </Row>  
    );  
  }  
}
```



• 역할

- todoList 한 줄을 구성합니다
- Check기능
- 삭제 button

Row / Col



- Row 는 한 줄입니다.
- Col 은 Row를 등분한 것입니다. (24개)
- Span은 등분한 col을 얼마큼 점유할 것인지 명시합니다.



2 18

4

```
<Row className="todo" >
  <Col
    span={2}
    className="action"
  >
</Col>
  <Col span={18}>
</Col>
  <Col
    span={4}
    className="action"
  >
</Col>
</Row>
```

ContentsComponent

```
class ContentsComponent extends Component {
  render(){
    const { list, ...other } = this.props;
    return(
      <Row >
        <Col span={24}>
          <Tabs defaultActiveKey="1">
            <TabPane tab="Todos" key="1">
              {list
                .filter(todo => !todo.complete)
                .map(
                  todo => (
                    <ListComponent
                      key={todo.index}
                      todo={todo}
                      {...other}
                    />
                  )
                )
              }
            </TabPane>
            <TabPane tab="Completed" key="2">
              {list
                .filter(todo => todo.complete)
                .map(
                  todo => (
                    <ListComponent
                      key={todo.index}
                      todo={todo}
                      {...other}
                    />
                  )
                )
              }
            </TabPane>
          </Tabs>
        </Col>
      </Row>
    );
  }
}
```

• 역할

- Tab을 분리하여 리스트를 뿌려준다.
- Tab 분리기준: complete = true/false
- Complete는 체크 할때마다 변경된다.

ContainerComponent

```
class ContainerComponent extends Component {  
  state = {  
    text: '',  
    list: [],  
    index: 0,  
  };  
  
  onChangeText = (e) => {  
    this.setState({  
      text: e.target.value,  
    });  
  };  
  
  onSubmit = () => {  
    this.setState({  
      list: this.state.list.concat({  
        index: this.state.index++,  
        text: this.state.text,  
        complete: false,  
      }),  
      text: '',  
    })  
  };  
  
  onDelete = (index) => {  
    this.setState({  
      list: this.state.list.filter(todo => todo.index !== index)  
    });  
  };  
  
  onChangeCheckbox = (index) => {  
    this.setState({  
      list: this.state.list.map(todo => todo.index === index ? { ...todo, complete: !todo.complete } : todo),  
    });  
  };  
}
```

```
render() {  
  const { text, list } = this.state;  
  return (  
    <div className="APP">  
      <div className="wrapper">  
        <HeaderComponent  
          onChangeText={this.onChangeText}  
          onSubmit={this.onSubmit}  
          text={text}  
        />  
        <ContentsComponent  
          list={list}  
          onDelete={this.onDelete}  
          onChangeCheckbox={this.onChangeCheckbox}  
        />  
      </div>  
    </div>  
  );  
}
```

THANK
YOU