React-Redux를 이용한 TodoList 만들기

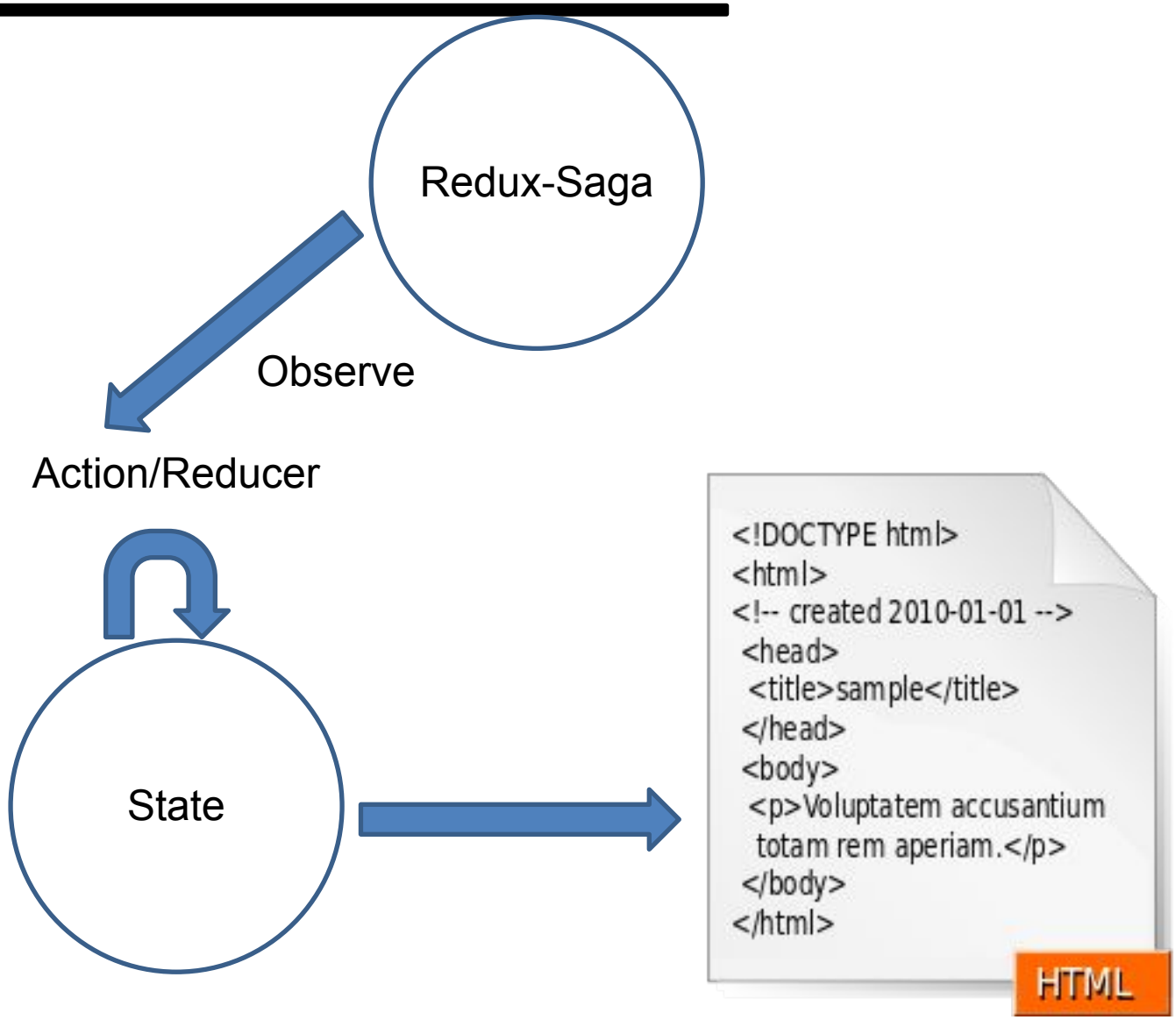
(2018 봄학기) 소프트웨어 개발의 원리 및 실제

2018. 04. 03 (화)

오늘 할일

- 이전에 했던 **React-Redux, Redux-saga** 예제의 추가 설명
  - 이전에 했던 **TodoList** 재구현(서버 **side**와 함께)
  - 오늘은 하나하나 같이 짜볼 것

구조 소개


Redux-Saga
Observe
Action/Reducer
State
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
<head>
<title>sample</title>
</head>
<body>
Voluptatem accusantium
totam rem aperiam.
</body>
</html>
HTML

사전 준비

- **“ARc”(Atomic React) 사용**
  - 기본적인 설정 제공
  - React의 component들을 여러 단계로 나눔

    (ex. Atoms, Molecules, organism, page, template)
  - 이전 단계의 component가 모여서 다음 단계의 component를 구성
- 프로젝트 시작하기
  - mkdir todo_list

    cd todo_list

    sudo npm install –g yo generator-arc

    yo arc . #redux선택

    npm install

    npm start

사전 준비

- **자동 생성된 주요 디렉토리 소개**
  - **src/store : state와 action, reducer가 들어감(Redux, Redux-saga)**
  - **src/components : 웹페이지에 그려질 그림을 구성할 component들이 들어감(React)**
  - **src/containers : component가 그림을 그릴 때 필요한 정보를 state에서 뽑아내서 component에 전달(React-Redux)**
  - **src/services : 외부 서버로 자료를 전송할 수 있는 fetch api등이 준비되어있는 부분**

TODO List만들기 1. 설계하기(1)

- **TODO List에 어떤 것이 필요할지 생각하여 설계**
  - **Redux쪽 : TODO list의 state를 설계**
  - **React쪽 : TODO list는 어떤 component로 이루어져야할지 생각**
  - **React-Redux : React의 component들이 그림을 그릴 때 state로부터 어떤 정보를 받아야 하는지 생각**

- **일단, TODO list의 state부터 생각해보자.**

TODO List만들기 1. 설계하기(2) : TODO list의 state

- **TODO list는 기본적으로 ”할 일”들의 List**
- **“할 일”들은 자신이 어떤 할일인지와 그 일의 진행 사항에 대한 정보를 가지고 있어야 한다.**
- **또한, 현재 state를 어떤식으로 변경해줄 수 있을지(Action)도 생각해야한다.**

TODO List만들기 1. 설계하기(2) : TODO list의 state

- **TODO_list state = [할 일1, 할 일2, 할 일3 , …]**
- **할 일 = { id : 1, contexts : 밥먹기, done : true }**
- **InitialState = []**
- **즉, TODO_list의 state는 할 일(TODO)들의 List이다.**
  할 일(TODO)들은 어떤 일을 할지(contexts)와 진행사항(done)을 가지고 있어야 한다.

TODO List만들기 1. 설계하기(2) : TODO list의 state

- **TODO list의 Action은 무엇이 있을지 생각해본다.**

- **TODO list의 Action은 두 가지를 생각해볼 수 있다.**

  - **TODO list에 새로운 TODO를 추가하는 Action**
  - **TODO list에 있는 TODO 중 의 하나의 진행 상황을 바꾸는 Action**
    - **Ex) TODO list = [TODO1, TODO2, …]일때,**
      TODO1의 done을 false에서 true로 바꾼다.

TODO List만들기 1. 설계하기(3) : TODO list의 components

- **ARc(Atomic React)의 component에는 계급이 있다.**

  Atom: 기본적인 html tag나 html tag를 작성하는 React component

  Ex. Input, Button, Todo..

  Molecule: Atom의 모임

  Ex. AddTodo, TodoList

  Organism: Molecule의 모임, 혹은 Organism들의 모임

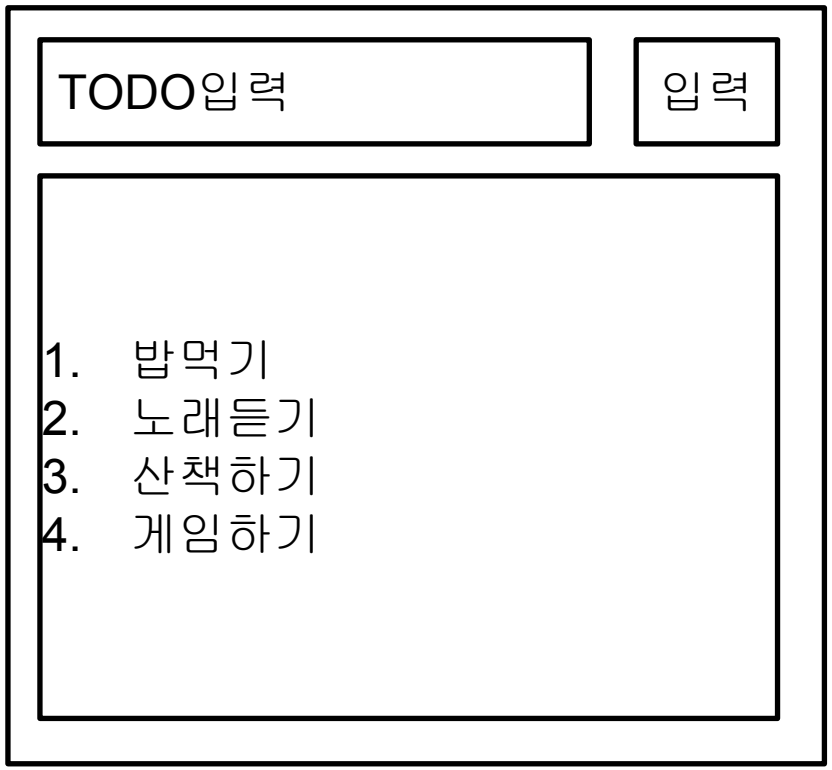
  Page: 그냥 페이지... 주로 Organism들이 들어간다

  Ex. homepage

  Template : Page에 쓰이는 layout

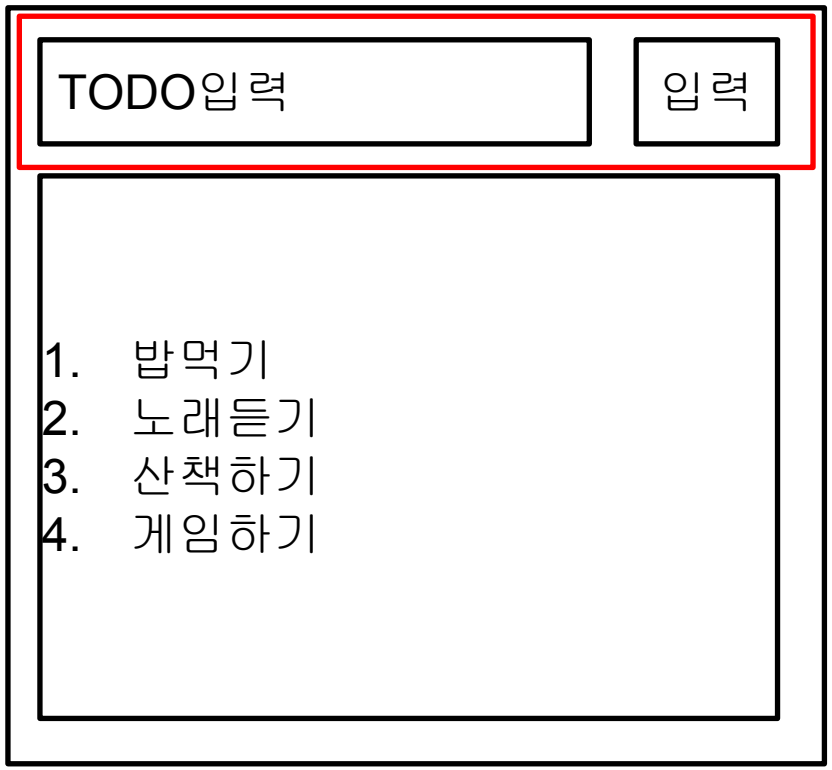
TODO List만들기 1. 설계하기(3) : TODO list의 components

- **TODO List는 TODO를 입력하는 부분과 TODO List를 보여주는 부분으로 나뉠 수 있다.**



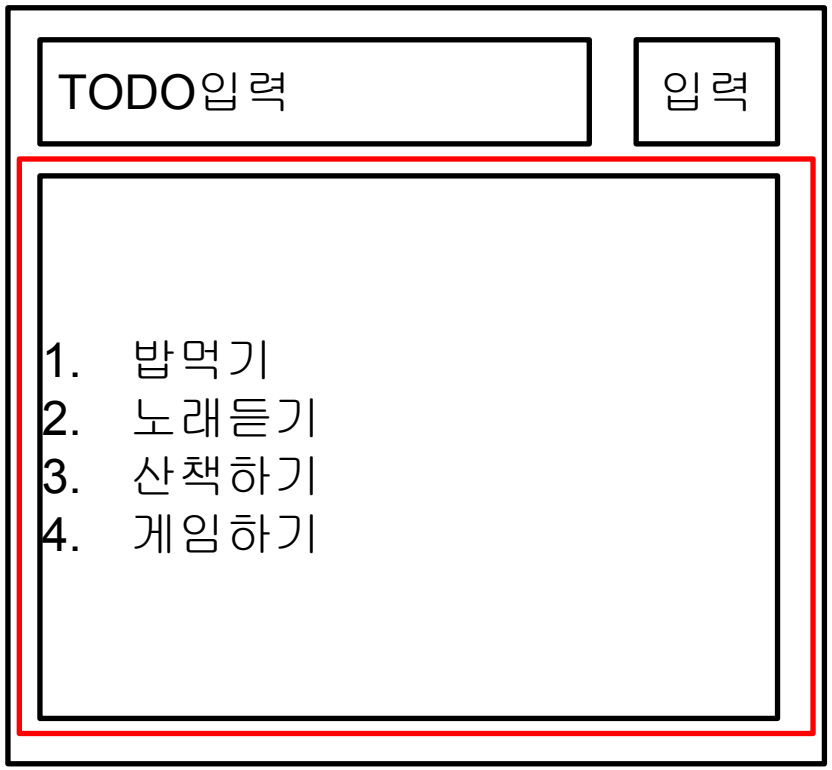
TODO List만들기 1. 설계하기(3) : TODO list의 components

- **TODO List는 TODO를 입력하는 부분과 TODO List를 보여주는 부분으로 나뉠 수 있다.(Molecule)**



TODO List만들기 1. 설계하기(3) : TODO list의 components

- **TODO List는 TODO를 입력하는 부분과 TODO List를 보여주는 부분으로 나뉠 수 있다. (Molecule)**

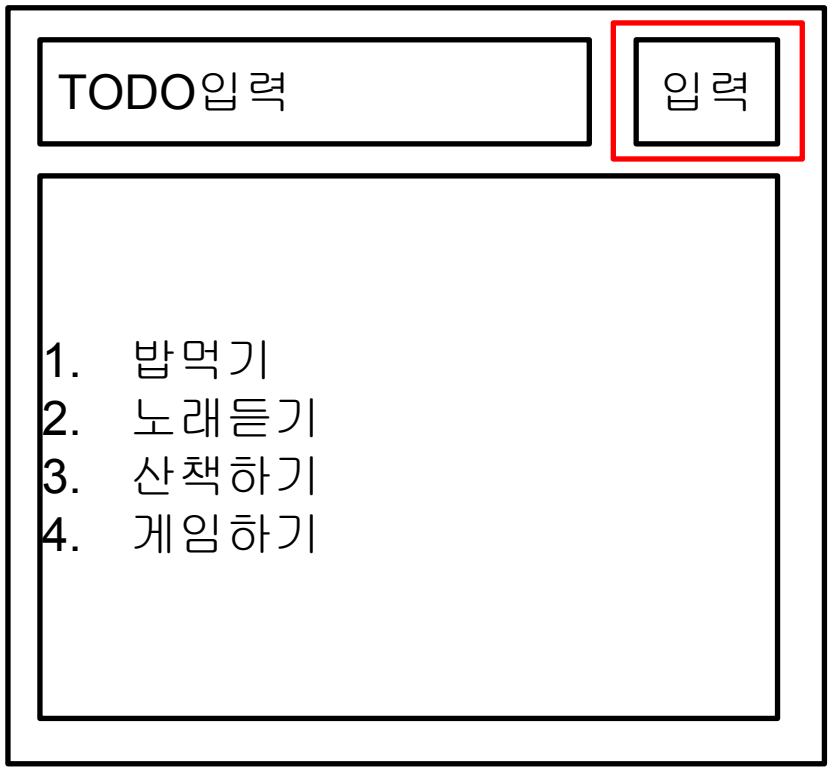


TODO List만들기 1. 설계하기(3) : TODO list의 components

- **TODO를 입력하는 부분은 글을 쓰는 부분과 입력버튼으로 이루어진다.(Atom)**

TODO List만들기 1. 설계하기(3) : TODO list의 components

- **TODO를 입력하는 부분은 글을 쓰는 부분과 입력버튼으로 이루어진다.(Atom)**



TODO List만들기 1. 설계하기(3) : TODO list의 components

- **TODO List를 보여주는 부분은 여러개의 TODO를 보여주는 부분으로 나뉜다.(Atom)**

TODO List만들기 1. 설계하기(3) : TODO list의 components

- **TODO를 추가하는 부분(AddTodo, Molecule)**
  - **TODO의 context를 입력하는 부분(Atom)**
  - **입력버튼(Atom)**
- **TODO를 보여주는 부분(TodoList, Molecule)**
  - **하나의 TODO를 보여주는 부분(Atom)**

TODO List만들기 2. State 구성하기

- 앞에서 했던 설계대로 **state**와 관련된 부분을 **src/store/todolist부분에 작성한다.(src/store에서 mkdir todolist)**

- 일단, **selectors.js**에 **initialState**를 정의한다.

src/store/todolist/selectors.js

```
1 export const initialState = []
2
3
```

TODO List만들기 2. State 구성하기

- 앞에서 말한대로 **Action**들을 **actions.js**에 정의한다.
  - **ADD_TODO : 할 일의 contexts를 받아서 새로운 TODO를 TODO list에 추가하는 Action**

src/store/todolist/actions.js : addTodo

```
1 let nextTodoId = 0
2 export const addTodo = (text) => {
3     return {
4   type: 'ADD_TODO',
5   id: nextTodoId++,
6   text
7     }
8 }
```

TODO List만들기 2. State 구성하기

- **TOGGLE_TODO : id를 받아서 해당하는 TODO의 진행상황(done)을 바꾸는 Action**

src/store/todolist/actions.js : toggleTodo

```
 9
10 export const toggleTodo = (id) => {
11     return {
12   type: 'TOGGLE_TODO',
13   id
14     }
15 }
16
```

TODO List만들기 2. State 구성하기

- **Action과 State를 받아서 새로운 State를 생성하는 Reducer를 reducers.js에 정의한다.**

- **Reducer는 다음과 같은 일을 해야한다.**
  - **ADD_TODO action을 받으면, 현재 TODO list에 새로운 TODO를 생성한다.**
  - **TOGGLE_TODO action을 받으면,**
    list 에 존재하는 TOGGLE_TODO action이 가진 id와 일치하는 id를 가진 TODO 진행사항(done)을 변경한다.

TODO List만들기 2. State 구성하기

src/store/todolist/reducer.js

```
13 const todolist_reducer = (state = initialState, action) => {
14   switch (action.type) {
15     case 'ADD_TODO':
16       return [
17         ...state,
18         {
19           id: action.id,
20           text: action.text,
21           completed: false
22         }
23       ]
24     case 'TOGGLE_TODO':
25       return state.map(t =>
26         toggleTodo(t, action)
27       )
28     default:
29       return state
30   }
31 }
32
33 export default todolist_reducer
```

TODO List만들기 2. State 구성하기

src/store/todolist/reducer.js

```
 1 import { initialState } from './selectors'
 2
 3 const toggleTodo = (todo, action) => {
 4   if (todo.id !== action.id) {
 5           return todo
 6   }
 7   return {
 8           ...todo,
 9           completed: !todo.completed
10   }
11 }
```

TODO List만들기 3. Component 구성하기

- **arc-generator로 필요한 components를 자동생성한다.**
  - **yo arc:component**
  - **src 디렉토리 바깥에서 해야함!**

- 자동생성 명령어 요약
  - **yo arc:component #Atom, Button, Enter**

    yo arc:component #Atom, Todo, Enter

    yo arc:component #Molecule, TodoList, Enter

    yo arc:component #Molecule, AddTodo, Enter

TODO List만들기 3. Component 구성하기

- 여기서 필요한 **components**들은 앞에서 요약한 것처럼 구성

- **TODO**를 추가하는 부분**(AddTodo, Molecule)**
  - **TODO**의 **context**를 입력하는 부분**(Atom)**
  - 입력버튼**(Atom)**
- **TODO**를 보여주는 부분**(TodoList, Molecule)**
  - 하나의 **TODO**를 보여주는 부분**(Atom)**

TODO List만들기 3. Component 구성하기

- **위에 언급한 Components들 중에 완벽하게 자동생성되지 않은 부분을 채워넣는다.**
  - **Todo**
  - **AddTodo**
  - **TodoList**

TODO List만들기 3. Component 구성하기

src/components/atoms/Todo/index.js

```
 1 import React, { PropTypes } from 'react'
 2 import styled from 'styled-components'
 3 import { font } from 'styled-theme'
 4
 5 const Styledli = styled.li`
 6   font-family: ${font('primary')};
 7 `
 8
 9 const Todo = ({ onClick, completed, text }) => (
10   <Styledli
11     onClick={onClick}
12     style={{
13   textDecoration: completed ? 'line-through' : 'none'
14     }}
15   >
16   {text}
17     </Styledli>
18 )
19
```

```
21
22 Todo.propTypes = {
23     onClick: PropTypes.func.isRequired,
24     completed: PropTypes.bool.isRequired,
25     text: PropTypes.string.isRequired,
26     reverse: PropTypes.bool
27 }
28
29 export default Todo
```

TODO List만들기 3. Component 구성하기

src/components/molecules/AddTodo/index.js

```
1 import React, { PropTypes } from 'react'
2 import styled from 'styled-components'
3 import { font, palette } from 'styled-theme'
4 import Button from '../../../components/atoms/Button'
5
6 const Wrapper = styled.div`
7 font-family: ${font('primary')};
8 color: ${palette('grayscale', 0)};
9 `
```

TODO List만들기 3. Component 구성하기

src/components/molecules/AddTodo/index.js

```
10
11 export const AddTodo = ({ statefunction, onAddTodo, onPostTodo }) => {
12     let input;
13     console.log(onAddTodo);
14     console.log('asdf')
15     const onSubmit = () => {
16   console.log('outer scope of if');
17   if (input != undefined) {
18       console.log('inner scope of if');
19       onAddTodo(input.value);
20       input.value = '';
21   }
22     };
23
24     const onPut = () => {
25   if (input != undefined) {
26       onPostTodo(input.value);
27       input.value = '';
28   }
29     };
30
```

TODO List만들기 3. Component 구성하기

src/components/molecules/AddTodo/index.js

```
30
31      return (
32        <div>
33          <input ref={node => {input = node;}} />
34          <Button type="submit" onClick={onSubmit}>ADD Todo</Button>
35        <Button type="submit" onClick={onPut}>POST Todo</Button>
36        </div>
37      );
38  };
39
40  AddTodo.propTypes = {
41      reverse: PropTypes.bool,
42      children: PropTypes.node,
43  }
44
```

TODO List만들기 3. Component 구성하기

src/components/molecules/TodoList/index.js

```
import React, { PropTypes } from 'react'
import styled from 'styled-components'
import { font } from 'styled-theme'
import Todo from '../../../components/atoms/Todo'


const Styledul = styled.ul`
  font-family: ${font('primary')};
`

export const TodoList = ({ todoliststate = [], onTodoClick }) => {
    return (
      <Styledul>
      {todoliststate.map(todo =>
            <Todo key={todo.id}
            {...todo}
            onClick={() => onTodoClick(todo.id)}
            />
          )}
      </Styledul>
  );
};

```

TODO List만들기 3. Component 구성하기

src/components/molecules/TodoList/index.js

```
24 TodoList.propTypes = {
25     todoliststate: PropTypes.arrayOf(PropTypes.shape({
26   id: PropTypes.number,
27   completed: PropTypes.bool,
28   text: PropTypes.string
29     })),
30     reverse: PropTypes.bool,
31 }
```

TODO List만들기 3. Component 구성하기

- **/src/components/pages/HomePage에 AddTodo와 TodoList를 등록한다.**

src/components/pages/HomePage/index.js

```
 1 import React from 'react'
 2 import AddTodo from '../../../containers/AddTodo'
 3 import TodoList from '../../../containers/TodoList'
 4
 5 const HomePage = () => {
 6   return (
 7     <div>
 8       <AddTodo/>
 9       <TodoList/>
10     </div>
11   )
12 }
13
14 export default HomePage
```

TODO List만들기 3. Component 구성하기 – 유의할 점

- **AddTodo**
  - 제출 버튼을 누르면, **Input**에 쓰여진 내용으로 **Todo**를 만들어서 **State**에 추가 **(ADD_TODO)**

- **TodoList**
  - **State**에서 **Todo**들의 **list**를 받아와서 **TODO list**를 그려줌
  - **TodoList**를 구성하는 **Todo**를 누르면, **Todo**의 진행사항**(done)**이 변경 **(TOGGLE_TODO)**

TODO List만들기 3. Store와 Component연결

- **src/containers**
- **Connect 함수를 통해서 component가 필요로 하는 내용을 전송**
  - **mapStateToProps : State의 정보를 뽑아냄**
  - **mapDispatchToProps : component가 일으키는 action을 dispatch 시킴**

- **AddTodo와 TodoList는 Store에서 정보를 가져오기도 하며, Store에 Action을 보내서(dispatch) State를 변경시키기도 한다.**

- **일단, AddTodo가 Store와 어떤 상호작용을 하는지 생각해보자.**

TODO List만들기 3. Store와 Component연결

- **src/containers/AddTodo.js**

- **AddTodo는 Input부분에 쓰여져 있는 내용으로 새로운 Todo를 만들어서 State에 추가하는 Action을 일으킬 수 있어야 한다.**
  - **mapDispatchToProps 로 ADD_TODO를 dispatch시킬 수 있는 함수를 보냄**

TODO List만들기 3. Store와 Component연결

src/containers/AddTodo.js

```
 1 import { connect } from 'react-redux'
 2 import { AddTodo } from '../components/molecules/AddTodo'
 3 import { addTodo, postTodoRequest } from '../store/todolist/actions'
 4
 5 const mapStateToProps = (state) => {
 6     return {
 7   statefunction : state
 8     }
 9 }
10
11 const mapDispatchToProps = (dispatch) => {
12     return {
13   onAddTodo: (text) => {
14       dispatch(addTodo(text))
15   },
16   onPostTodo: (text) => {
17       dispatch(postTodoRequest(text))
18   }
19     }
20 }
21
22 export default connect(mapStateToProps, mapDispatchToProps)(AddTodo)
```

TODO List만들기 3. Store와 Component연결

- **src/containers/TodoList.js**

- **State에서 Todo들의 list를 받아와서 TODO list를 그림**
  - **mapStateToProps로 state의 todolist를 뽑아서 보내줌**

- **TodoList를 구성하는 Todo를 click하면**

  해당 Todo의 진행사항(done)을 바꾸는 Action을 일으킴
  - **mapDispatchToProps로 TOGGLE_TODO Action을 dispatch시킬 수 있는 함수를 보냄**

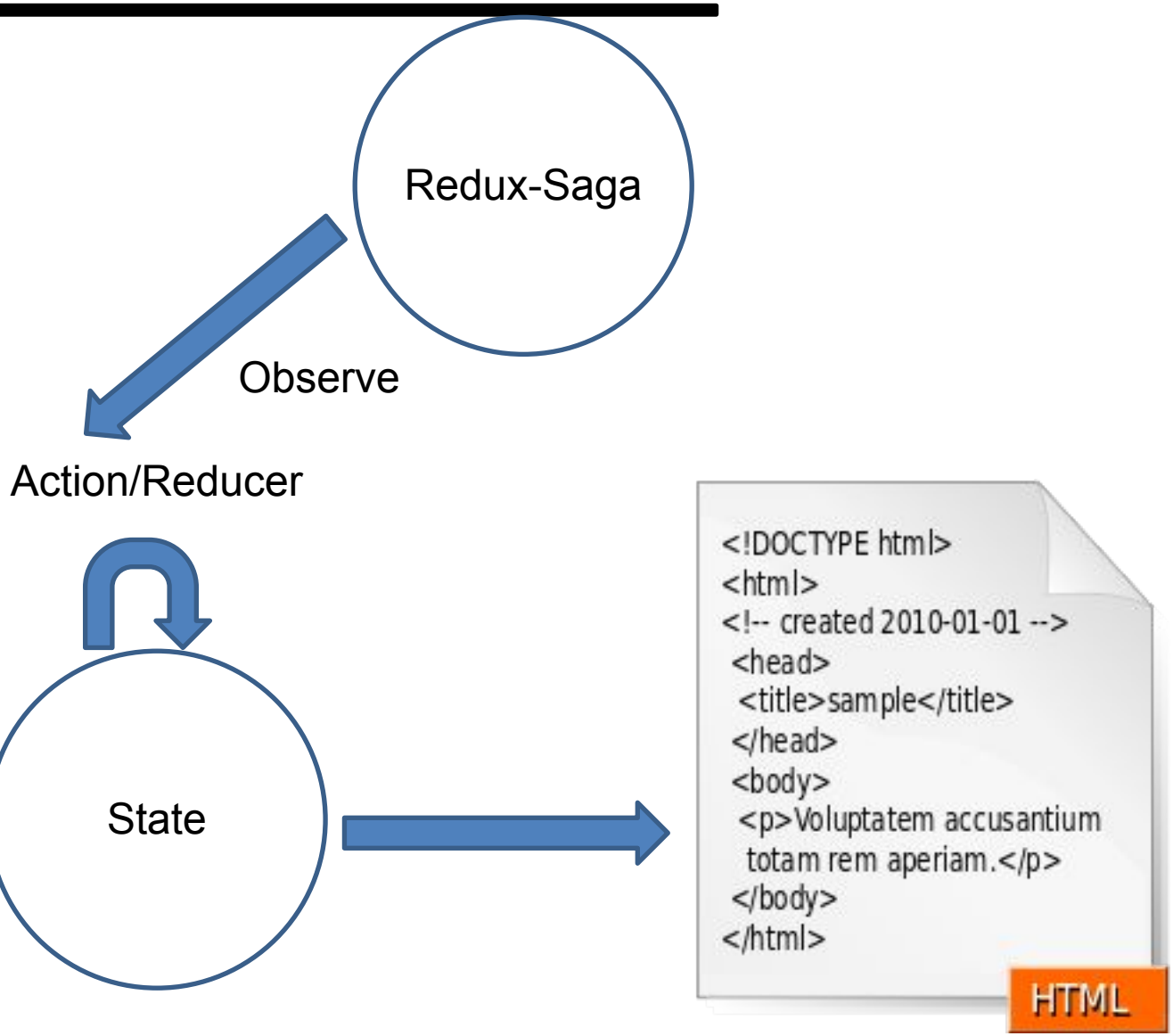
TODO List만들기 3. Store와 Component연결

src/containers/TodoList.js

```
 1 import { connect } from 'react-redux'
 2 import { toggleTodo } from '../store/todolist/actions'
 3 import { TodoList } from '../components/molecules/TodoList'
 4
 5 const mapStateToProps = (state) => {
 6   return {
 7     todoliststate: state.todolist
 8   }
 9 }
10
11
12 const mapDispatchToProps = (dispatch) => {
13   return {
14     onTodoClick: (id) => {
15       dispatch(toggleTodo(id))
16     }
17   }
18 }
19
20 export default connect(mapStateToProps, mapDispatchToProps)(TodoList)
```

TODO List만들기 : 중간과정

구조 소개 : 다시 한번


Redux-Saga
Observe
Action/Reducer
State
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
<head>
<title>sample</title>
</head>
<body>
Voluptatem accusantium
totam rem aperiam.
</body>
</html>
HTML

TODO List만들기 4. 서버와 통신하기

- **Redux-saga와 src/services/api에서 제공하는 fetch api를 통해서 서버와 통신**
- **앞에서 했던 action과 reducer를 만들었던 일과 거의 유사함**
- **단, saga에서는 주로 action을 기다리는 함수와 실제로 action을 처리해주는 함수를 따로 만듬**

TODO List만들기 4. 서버와 통신하기

- 서버에 **POST**명령을 내리는 **saga**를 만들어 볼 것

- 일단, 서버에 **POST**을 보낼 **Action**을 정의한다.
  - **src/store/todolist/actions.js**
  - **POST_TODO_REQUEST Action**

TODO List만들기 4. 서버와 통신하기

```
17 export const POST_TODO_REQUEST = 'POST_TODO_REQUEST'
18
19 export const postTodoRequest = (text) => {
20     return {
21   type: POST_TODO_REQUEST,
22   text
23     }
24
25 }
```

TODO List만들기 4. 서버와 통신하기

- **Sagas.js에도 reducer처럼 Action을 처리하는 함수가 있어야함**
  (saga가 가져오는 Action은 reducer가 처리할 수 없는 것이기 때문)

- **src/store/todolist/sagas.js**
  POST_TODO_REQUEST action을 처리하는 부분을 만들어야 함

- **Action이 dispatch되는 것을 기다리는 함수(saga)를 만든다.**
  - **watchPostTodoRequest()**
- **실제로 Action을 처리하는 함수(saga)를 만든다.**
  - **postTodo(text)**

TODO List만들기 4. 서버와 통신하기

```
1 import { take, put, call, fork } from 'redux-saga/effects'
2 import api from 'services/api'
3 import * as actions from './actions'
4
5
6 const url = 'http://127.0.0.1:8000/todos/'
7
8 export function* postTodo(text) {
9     console.log(text)
10    const data = yield call(api.post, url, {done: true, contents: text})
11 }
12
13 export function* watchPostTodoRequest() {
14    while (true) {
15   const { text } = yield take(actions.POST_TODO_REQUEST)
16   yield call(postTodo, text)
17    }
18 }
19
20
21 export default function* () {
22    yield fork(watchPostTodoRequest)
23 }
```

TODO List만들기 4. 서버와 통신하기

- **src/components/molecules/AddTodo**
  누르면 POST_TODO_REQUEST Action을 dispatch하는 버튼 생성

- **src/containers/AddTodo.js**
  connect를 통해 POST_TODO_REQUEST Action을 dispatch할 수 있는 함수를 component에게 넘겨준다.(mapDispatchToProps 사용)

TODO List만들기 4. 서버와 통신하기

src/components/molecules/AddTodo/index.js

```
30
31      return (
32        <div>
33          <input ref={node => {input = node;}} />
34          <Button type="submit" onClick={onSubmit}>ADD Todo</Button>
35        <Button type="submit" onClick={onPost}>POST Todo</Button>
36        </div>
37      );
```

TODO List만들기 4. 서버와 통신하기

src/containers/AddTodo.js

```
10
11 const mapDispatchToProps = (dispatch) => {
12     return {
13   onAddTodo: (text) => {
14       dispatch(addTodo(text))
15   },
16   onPostTodo: (text) => {
17       dispatch(postTodoRequest(text))
18   }
19     }
20 }
21
```

TODO List만들기 5. 질문시간

- **질문이 있으면 말씀해주세요.**