

Jul8

- Quick Start -

10살에 꿈을 잡을 뻔했던 김민규

이 자료에서 다루는 내용

- 들어가기 전에..
- 첫번째, 개발 환경 구축하기
- 두번째, TODO 앱 만들기



들여가기 전에..
왜 이런 자료를 남기죠?

웹 프론트엔드 개발에 관심 있거나 Jul8 프레임워크를 사용해보고 싶으신 분을 위한 자료
(사실.. 처음 접하는 환경에 당황해서.. 누군가는 이 과정에서 헤매지 말고 행복했으면.. 든든한 서포터)



무엇을 할 예정인가요?

Jul8을 다루기 위한 환경을 구축하고 TODO 앱을 만들어 볼 예정
(가장 만만한 예제는 역시 TODO 앱~)

사전에 알아야할 지식이 있을까요?

HTML/CSS, Typescript, jQuery

저는 Javascript 밖에 모르는데요?

Typescript == Javascript

1. 타입스크립트는 자바스크립트를 확장한 언어이며 자바스크립트의 ECMA 표준을 따르고 있다.
2. 자바스크립트 문법 그대로 사용할 수 있다.
3. 정적 타입을 지원하고 JS로 트랜스파일링 해준다.

(TypeScript를 무서워하지 않아도 되는 이유 : <http://han41858.tistory.com/14>)

Jul8

- 첫번째, 개발 환경 구축하기 -



Jul8

TypeScript



Visual Studio 2017, HTML/CSS, Jul8, Typescript, jQuery

환경 구축은 현재의 Jul8 버전에 맞춰서 진행합니다.

※ 주의사항

이 프로젝트의 구성은 실제 Jul8의 샘플과 다르며 [Quick Start]에 최적화 했습니다.

개발 환경 구축하기

1. Visual Studio 2017 Community 설치하기

(필수) <https://visualstudio.microsoft.com/ko/vs/community/>
기존에 사용 중이던 Visual Studio가 있으면 이 과정을 넘기세요.

(필수) .NET Framework 4.7 를 설치합니다.


(필수) ASP.NET 및 웹 개발을 설치합니다.

(필수) Typescript 2.8을 설치합니다.

타입스크립트(Typescript) 설치하기

타입스크립트 문서 보기: <https://www.typescriptlang.org/index.html>

타입스크립트에 대한 자세한 문서는 여기로



두 방법 중 하나를 선택하여 설치합니다.

1. TypeScript for Visual Studio 2017로 설치하기

아래 링크로 들어가서 [자세한 내용] - [Typescript 2.8.1] 이상으로 설치합니다.

<https://www.microsoft.com/ko-kr/download/details.aspx?id=55258>

2. Node.js Package로 설치하기

커맨드라인으로 설치 합니다.

```
npm install -g typescript
```

설치된 버전을 확인할 수 있습니다.

```
tsc -v
```


Github에서 Jul8 가져오기

1. Jul8 클론하기

Jul8-Quick-Start 클론을 만들어 가져옵니다. (처음 시작하시는 분들을 위해 기존 샘플을 제거한 포크 버전입니다.) - 오픈소스 굿!!

```
./> git clone https://github.com/simple-lab/Jul8-Quick-Start.git
```

샘플 버전(원본) 확인을 원하시면 다음과 같이 클론을 만들어 가져옵니다.

```
./> git clone https://github.com/devcat-studio/Jul8.git
```

2. Jul8-Quick-Start 디렉토리 구성

```
Jul8Compiler

TodoApplication
├─ js/                // 트랜스파일링 된 스크립트가 저장되는 공간
├─ templates/        // 템플릿이 저장되는 폴더
├─ ts/
│   └─ lib/           // typescript 라이브러리 폴더
│       └─ Main.ts
│       └─ TodoApp.ts
├─ index.html         // QuickStart의 홈 페이지
├─ jul8config.json    // jul8 컴파일 옵션 파일
├─ styls.css          // 기본 스타일 시트
└─ tsconfig.json      // typescript 컴파일 옵션 파일
```

j8config.json 구성

```
{
  "header": [
    "/// <reference path='lib/jul8.ts' />",
    "/// <reference path='lib/jquery.d.ts' />",
    ""
  ],
  "footer": [
  ],
  "build": [
    {
      "source": "templates/templates.html",
      "target": "ts/templates.g.ts"
    }
  ]
}
```

- 1) header - 자동 생성된 스크립트 문서 가장 상단에 들어가는 코드를 삽입합니다.
- 2) footer - 자동 생성된 스크립트 문서 가장 하단에 들어가는 코드를 삽입합니다.
- 3) build - 빌드 대상 템플릿(html) 소스와 자동 생성될 대상(ts)의 경로를 지정합니다.
 - source - 작성한 템플릿 소스입니다.
 - target - 코드젠으로 자동 생성된 템플릿 스크립트

※ source 대상의 파일에 아무런 내용이 없으면 빌드 에러가 발생합니다.

빌드 해보기 (1)

1. 시작 프로젝트 설정

프로젝트(TodoApplication) [우클릭 메뉴] - [시작 프로젝트 설정]

2. 프로젝트 빌드

프로젝트(TodoApplication) [우클릭 메뉴] - [빌드] (혹은 다시 빌드)

3. ts/ 폴더에 templates.g.ts 파일이 생성되면 성공!!

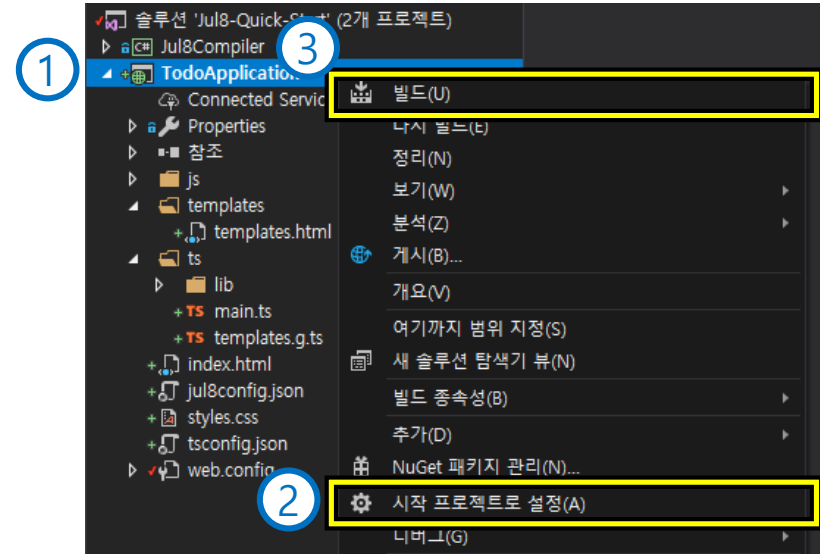


templates.g.ts

※ 빌드에 실패하면 아래 내용을 확인합니다.

프로젝트(TodoApplication) [우클릭 메뉴] - [속성] - [빌드 이벤트] - [빌드 이벤트 명령줄 대화 상자(R)] 에 다음과 같은 명령줄이 있는지 확인합니다.

```
$(SolutionDir)Compiler\bin\Jul8Compiler.exe $(ProjectDir)jul8config.json
```



4. 빌드 시 자동 생성된 스크립트 파일 내용

```
<div j8-template="ButtonGroup">
  <button type="button" j8-control="left">Left Button</button>
  <button type="button" j8-control="middle">Middle Button</button>
  <button type="button" j8-control="right">Right Button</button>
</div>
```

\$(ProjectDir)templates/templates.html



```
/// <reference path='lib/jul8.ts' />
/// <reference path='lib/jquery.d.ts' />
```

```
class ButtonGroup_d implements Jul8.View
{
```

```
  $: JQuery;
  left: JQuery;
  middle: JQuery;
  right: JQuery;
```

```
  constructor(templateHolder: Jul8.TemplateHolder, parentNode?: JQuery)
  {
```

```
    this.$ = templateHolder.cloneTemplate('ButtonGroup');
    if (parentNode) { parentNode.append(this.$); }
    let s = new Jul8.Scanner(this.$, false);
    this.left = s.C('left');
    this.middle = s.C('middle');
    this.right = s.C('right');
```

```
  }
}
```

\$(ProjectDir)ts/templates.g.ts

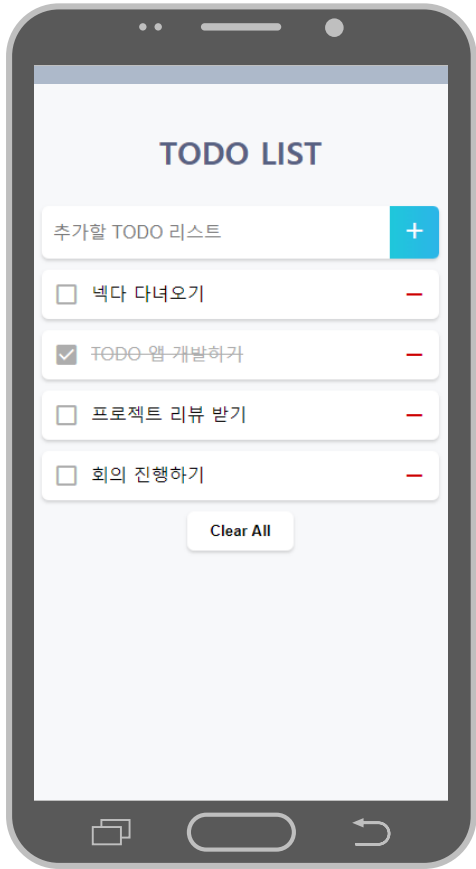
논리적인 요소를 정의해서
(템플릿)

스크립트로 자동 생성
(CodeGen)

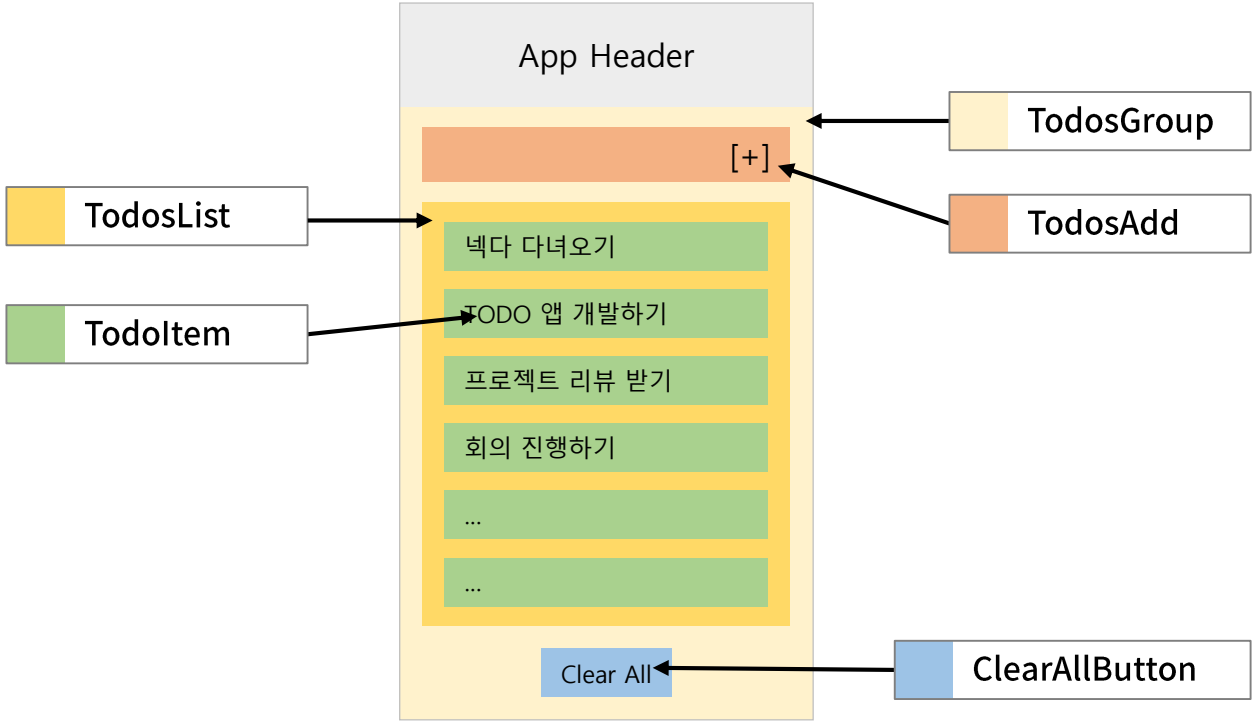
Jul8

- 두번째, TODO 앱 만들기 -

구현할 TODO 앱 미리보기



- 실행 화면 -



- 레이아웃 구성 -

HTML/CSS 구성 미리보기

※ index.html

[TODO Application] 프로젝트에 있는 예제 HTML 문서입니다.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0' name='viewport' />
  <title>Jul8-TODO-List-App</title>

  <link href="styles.css" rel="stylesheet" type="text/css" />
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  <script type="text/javascript" src="https://code.jquery.com/jquery-3.3.1.js"></script>
</head>
<body>
  <div id="app">

  </div>

  <script type="text/javascript" src="/js/all.js"></script>
</body>
</html>
```

예제에서 제공하는 스타일시트

Google Design - Material Icons

jQuery

앱의 구성 요소(Element)가 들어가는 공간

트랜스파일링 된 js파일

※ styles.css

[TODO Application] 프로젝트에 있는 styles.css를 그대로 사용하기에 자세한 설명은 생략합니다.

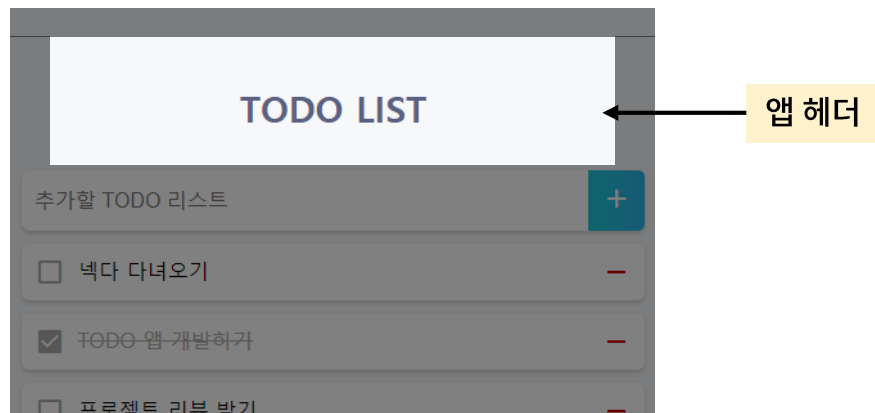
템플릿 만들기

1. [TODO 앱 만들기]에서는 모든 템플릿은 templates.html에 작성합니다.

```
TodoApplication
├ ...
└ templates/
  └ templates.html // 이 파일에 작성합니다.
```

2. [앱 헤더(타이틀) 템플릿]을 추가합니다.

```
<!-- 앱의 헤더 부분 -->
<div class="app-title" j8-template="AppTitle">
  TODO LIST
</div>
```



템플릿 만들기

3. [TODO 그룹 템플릿]을 추가합니다.

```
<!-- TODO 그룹 -->
<div class="todos-group" j8-template="TodosGroup">

  ① TodosAdd

  ② TodosList

  ③ ClearAllButton

</div>
```

※ 미리보기



템플릿 만들기

3-1. [TODO 그룹 템플릿] 안에 TODO 리스트를 추가하는 기능을 가진 컨트롤러를 추가합니다.

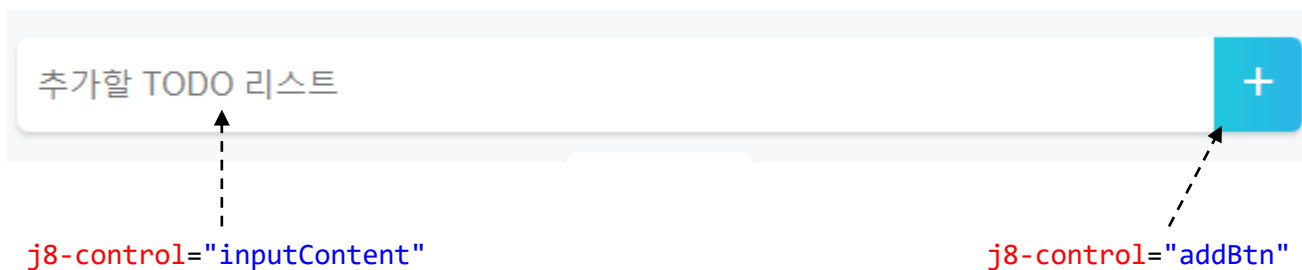
```
<!-- TODO 그룹 -->
<div class="todos-group" j8-template="TodosGroup">

  <!-- TODO 리스트를 추가하는 컨트롤러 -->
  <div class="todos-add">
    <input type="text" j8-control="inputContent" placeholder="추가할 TODO 리스트" />
    <button j8-control="addBtn"><i class="material-icons">add</i></button>
  </div>

  TodosList와 ClearAllButton 컨트롤러

</div>
```

※ 미리보기



템플릿 만들기

3-2. [TODO 그룹 템플릿] 안에 TODO 리스트 컨트롤러를 추가합니다.

```
<!-- TODO 그룹 -->
<div class="todos-group" j8-template="TodosGroup">

  TodosAdd 컨트롤러

  <!-- TODO 리스트 컨트롤러 -->
  <ul class="todos-list" j8-control="todosList">
    <li class="todo-item" j8-listItem="TodoItem" j8-model="todoModel">
      <div j8-control="todoCheckBtn">
        <input type="checkbox" j8-control="todoCheckState" />
        <i class="material-icons">check_box_outline_blank</i>
        <i class="material-icons" style="display: none;">check_box</i>
      </div>
      <div j8-control="todoContent">{{ content }}</div>
      <div j8-control="todoRemoveBtn">
        <i class="material-icons">remove</i>
      </div>
    </li>
  </ul>

  ClearAllButton 컨트롤러

</div>
```

※ 미리보기

☐ 벙다 다녀오기 —

☒ TODO 앱 개발하기 —

☐ 프로젝트 리뷰 받기 —

☐ 회의 진행하기 —

j8-listItem="TodoItem"



템플릿 만들기

3-3. [TODO 그룹 템플릿] 안에 전부 삭제하는 버튼 컨트롤러를 추가합니다.

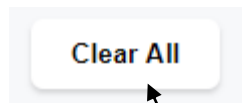
```
<!-- TODO 그룹 -->
<div class="todos-group" j8-template="TodosGroup">

  TodosAdd와 TodosList 컨트롤러

  <!-- TODO 리스트 모두 삭제하는 컨트롤러 -->
  <div class="flexbox">
    <button class="clear-all-btn" j8-control="clearAll">Clear All</button>
  </div>

</div>
```

※ 미리보기



j8-control="clearAll"

4. 완성된 템플릿 파일 (templates.html)

```
<!-- TODO 그룹 -->
<div class="todos-group" j8-template="TodosGroup">

  <!-- TODO 리스트를 추가하는 컨트롤러 -->
  <div class="todos-add">
    <input type="text" j8-control="inputContent" placeholder="추가할 TODO 리스트" />
    <button j8-control="addBtn"><i class="material-icons">add</i></button>
  </div>

  <!-- TODO 리스트 컨트롤러 -->
  <ul class="todos-list" j8-control="todosList">
    <li class="todo-item" j8-listItem="TodoItem" j8-model="todoModel">
      <div j8-control="todoCheckBtn">
        <input type="checkbox" j8-control="todoCheckState" />
        <i class="material-icons">check_box_outline_blank</i>
        <i class="material-icons" style="display: none;">check_box</i>
      </div>
      <div j8-control="todoContent">{{ content }}</div>
      <div j8-control="todoRemoveBtn">
        <i class="material-icons">remove</i>
      </div>
    </li>
  </ul>

  <!-- TODO 리스트 모두 삭제하는 컨트롤러 -->
  <div class="flexbox">
    <button class="clear-all-btn" j8-control="clearAll">Clear All</button>
  </div>
</div>
```

템플릿 파일을 Jul8Compiler을 사용해 CodeGen

5. [TodoApplication] 프로젝트를 빌드해 스크립트 코드를 자동생성 합니다. (ts/templates.g.ts)

```
class AppTitle_d implements Jul8.View
{
    $: JQuery;

    constructor(templateHolder: Jul8.TemplateHolder, parentNode?: JQuery)
    {
        this.$ = templateHolder.cloneTemplate('AppTitle');
        if (parentNode) { parentNode.append(this.$); }
        let s = new Jul8.Scanner(this.$, false);
    }
}
```

```
class TodosGroup_d implements Jul8.View
{
    $: JQuery;
    inputContent: JQuery;
    addBtn: JQuery;
    todosList: JQuery;
    clearAll: JQuery;
    listOf_TodoItem: Jul8.ViewList<TodosGroup_TodoItem_d>;

    constructor(templateHolder: Jul8.TemplateHolder, parentNode?: JQuery)
    {
        this.$ = templateHolder.cloneTemplate('TodosGroup');
        if (parentNode) { parentNode.append(this.$); }
        let s = new Jul8.Scanner(this.$, false);
        this.inputContent = s.C('inputContent');
        this.addBtn = s.C('addBtn');
        this.todosList = s.C('todosList');
        this.clearAll = s.C('clearAll');
        this.listOf_TodoItem = s.L<TodosGroup_TodoItem_d>('TodoItem');
    }
}
```

```
class TodosGroup_TodoItem_d implements Jul8.View
{
    $: JQuery;
    todoCheckBtn: JQuery;
    todoCheckState: JQuery;
    todoContent: JQuery;
    todoRemoveBtn: JQuery;

    constructor($: JQuery)
    {
        this.$ = $;
        let s = new Jul8.Scanner(this.$, true);
        this.j8fields = s.fields;
        this.todoCheckBtn = s.C('todoCheckBtn');
        this.todoCheckState = s.C('todoCheckState');
        this.todoContent = s.C('todoContent');
        this.todoRemoveBtn = s.C('todoRemoveBtn');
    }

    private j8fields: Jul8.Fields;

    set(data: todoModel): void
    {
        data.content;
        this.j8fields.set(data);
    }
}
```

6. [TODO 앱 만들기] 예제에서 기본으로 제공하는 Main.ts

※ 주의 - 이 예제에서는 Main.ts를 수정하지 않습니다.

```
/// <reference path='templates.g.ts' />
$(document).ready(() => {
    new Main();
});

class Main {

    public jul8: Jul8.TemplateHolder;
    public appNode: JQuery = $("#app");

    constructor() {

        // jul8config.json을 읽어 존재하는 템플릿을 확인합니다.
        $.get("/jul8config.json", (config: any) => {

            // Jul8-Quick-Start에서는 하나의 템플릿만 사용하여 흐름을 파악합니다.
            let templatePath: string = config.build[0].source;

            $.get(templatePath, (data) => {

                let virtualDOM: DocumentFragment = document.createDocumentFragment();
                $(virtualDOM).append(data);

                this.jul8 = new Jul8.TemplateHolder();
                this.jul8.addTemplateRoot($(virtualDOM));

                new TodoApp(this.jul8, this.appNode);
            });
        });
    }
}
```

← 스크립트 진입점

← <div id="app"></div>

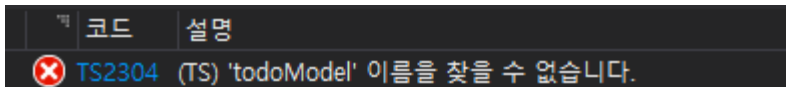
7. [TODO 앱 만들기] 예제에서 기본으로 제공하는 TodoApp.ts

모든 작업은 TodoApp.ts에서 이루어집니다.

```
class TodoApp {  
  
    private templateHolder: Jul8.TemplateHolder;  
    private parentNode: JQuery;  
  
    constructor(templateHolder: Jul8.TemplateHolder, parentNode: JQuery) {  
        this.templateHolder = templateHolder;  
        this.parentNode = parentNode;  
  
        // 이 곳에 뷰(View)를 추가합니다.  
    }  
}
```


8. j8-model을 Type Aliases로 선언하기

Jul8Compiler로 코드를 생성하면 (TS) 'todoModel' 이름을 찾을 수 없습니다. 오류가 발생합니다.



j8-model="todoModel" 을 작성하면 모델에 맞는 Type Aliases를 선언해줘야 합니다.
App.ts 가장 위에 추가 해줍니다.

```
// 반드시 j8-model마다 type aliases를 선언해줘야 합니다.
type todoModel = {
  content: string;
}

class TodoApp {

  private templateHolder: Jul8.TemplateHolder;
  private parentNode: JQuery;

  constructor(templateHolder: Jul8.TemplateHolder, parentNode: JQuery) {
    this.templateHolder = templateHolder;
    this.parentNode = parentNode;

    // 이 곳에 뷰(View)를 추가합니다.

  }
}
```

```
<!-- TODO 리스트 컨트롤러 -->
<ul class="todos-list" j8-control="todosList">
  <li class="todo-item" j8-listItem="TodoItem" j8-model="todoModel">
    <div j8-control="todoCheckBtn">
      <input type="checkbox" j8-control="todoCheckState" />
      <i class="material-icons">check_box_outline_blank</i>
      <i class="material-icons" style="display: none;">check_box</i>
    </div>
    <div j8-control="todoContent">{{ content }}</div>
    <div j8-control="todoRemoveBtn">
      <i class="material-icons">remove</i>
    </div>
  </li>
</ul>
```

9. 템플릿들을 TodoApp 클래스에 추가합니다.

```
// 반드시 j8-model마다 type aliases를 선언해줘야 합니다.
type todoModel = {
  content: string;
}

class TodoApp {

  private templateHolder: Jul8.TemplateHolder;
  private parentNode: JQuery;

  private appTitle: AppTitle_d;
  private todosGroup: TodosGroup_d;

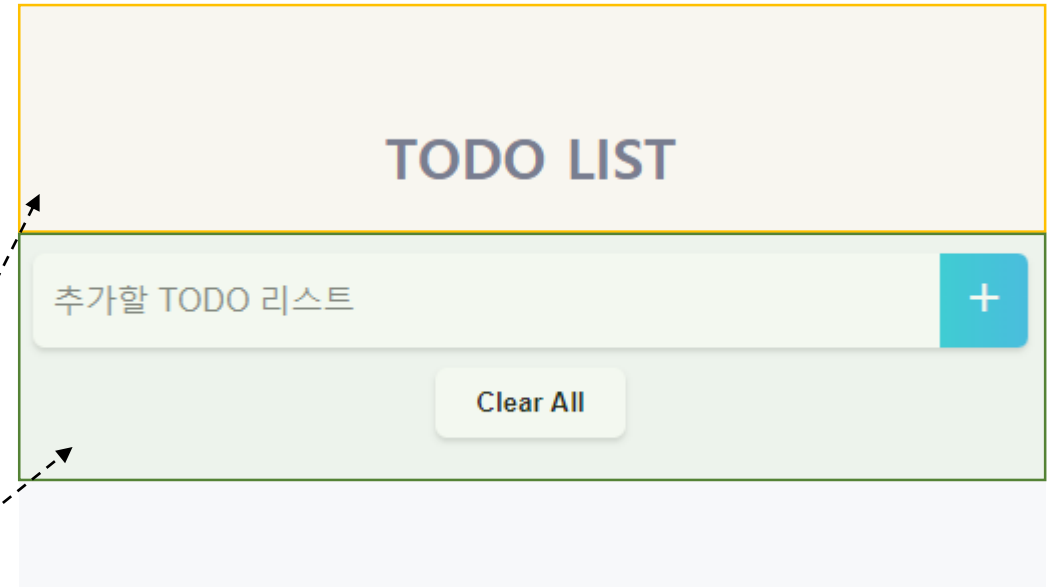
  constructor(templateHolder: Jul8.TemplateHolder, parentNode: JQuery) {
    this.templateHolder = templateHolder;
    this.parentNode = parentNode;

    // 앱 타이틀을 추가합니다.
    this.appTitle = new AppTitle_d(templateHolder, parentNode);

    // TodosGroup을 추가합니다.
    this.todosGroup = new TodosGroup_d(templateHolder, parentNode);
  }

  Event Logic Methods

}
```



10. TodoApp 클래스 내부에 이벤트 로직 addTodoList(), clearAll_TodoList() 을 작성한 함수를 추가합니다.

```
// TODO 리스트를 추가합니다.
addTodoList(): void {

    let todoStr: string = this.todosGroup.inputContent.val();
    if (todoStr.length <= 0) return;

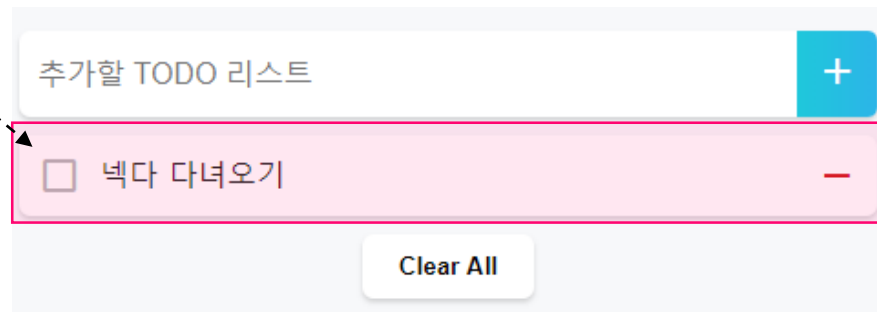
    let list = this.todosGroup.listOf_TodoItem.add(TodosGroup_TodoItem_d);

    this.todosGroup.inputContent.val("");
    list.set({ content: todoStr });

    // TODO 리스트의 체크박스를 클릭해 상태를 변경합니다.
    list.todoCheckBtn.click(() => {
        if (list.todoCheckState.prop("checked") == true) {
            list.todoCheckBtn.children().eq(1).show();
            list.todoCheckBtn.children().eq(2).hide();
            list.todoContent.removeClass("checkedTodo");
            list.todoCheckState.prop("checked", false);
        }
        else {
            list.todoCheckBtn.children().eq(1).hide();
            list.todoCheckBtn.children().eq(2).show();
            list.todoContent.addClass("checkedTodo");
            list.todoCheckState.prop("checked", true);
        }
    });

    // 현재 TODO 리스트를 제거합니다.
    list.todoRemoveBtn.click(() => {
        this.todosGroup.listOf_TodoItem.remove(list); // 혹은 list.$.remove();
    });
}
```

```
// 모든 TODO 리스트를 제거합니다.
clearAll_TodoList(): void {
    this.todosGroup.todosList.html("");
}
```



11. 작성한 이벤트를 클릭 이벤트로 바인딩 합니다.

```
class TodoApp {  
  
    private templateHolder: Jul8.TemplateHolder;  
    private parentNode: JQuery;  
  
    private appTitle: AppTitle_d;  
    private todosGroup: TodosGroup_d;  
  
    constructor(templateHolder: Jul8.TemplateHolder, parentNode: JQuery) {  
        this.templateHolder = templateHolder;  
        this.parentNode = parentNode;  
  
        // 앱 타이틀을 추가합니다.  
        this.appTitle = new AppTitle_d(templateHolder, parentNode);  
  
        // TodosGroup을 추가합니다.  
        this.todosGroup = new TodosGroup_d(templateHolder, parentNode);  
  
        // TodosGroup의 [+]버튼에 클릭 이벤트를 바인딩합니다.  
        this.todosGroup.addBtn.click(() => this.addTodoList());  
  
        // clearAll 버튼에 클릭 이벤트를 바인딩 합니다.  
        this.todosGroup.clearAll.click(() => this.clearAll_TodoList());  
    }  
  
    Event Logic Methods  
  
}
```





마지막 단계!!!

여기까지 오셨다면 TODO 앱을 위한 모든 절차가 끝났습니다.

빌드를 시도하고 페이지를 열어봅니다.

완성된 TODO 앱 실행 화면

※ Github Pages에 업로드 했습니다.

[Demo] https://simple-lab.github.io/Jul8-Quick-Start/completed_version/

