

2018-2
웹 시스템 설계
실습

Week 10

Vue.js (router)

Composed by:
WISE Research Lab Ajou University



Preliminary

■ 실습 목표

- Vue.js router 사용법을 실습하고, 게시글 열람 Web App 을 개발한다.

■ Routing in SPA

- Single Page Application(SPA)는 하나의 정적 페이지 파일(HTML, JS, CSS)을 기반으로, 사용자 요청에 따라 동적으로 자료를 받아와 보여주는 방식의 Application 이다.
 - ✓ 하나의 정적 페이지에서 새로운 페이지 요청 없이 다양한 자료들을 동적으로 보여주어, 더욱 연속적인 사용자 경험을 제공할 수 있다.
- 기존의 Backend REST Server 를 통해서 view 를 받아오는 방식을 사용하지 않고, Frontend 에서 view 를 URI 에 따라 변경하기 위해, routing 을 사용한다.
 - ✓ REST Server 는 동적으로 자료를 보여주기 위한 REST 요청만 처리한다.

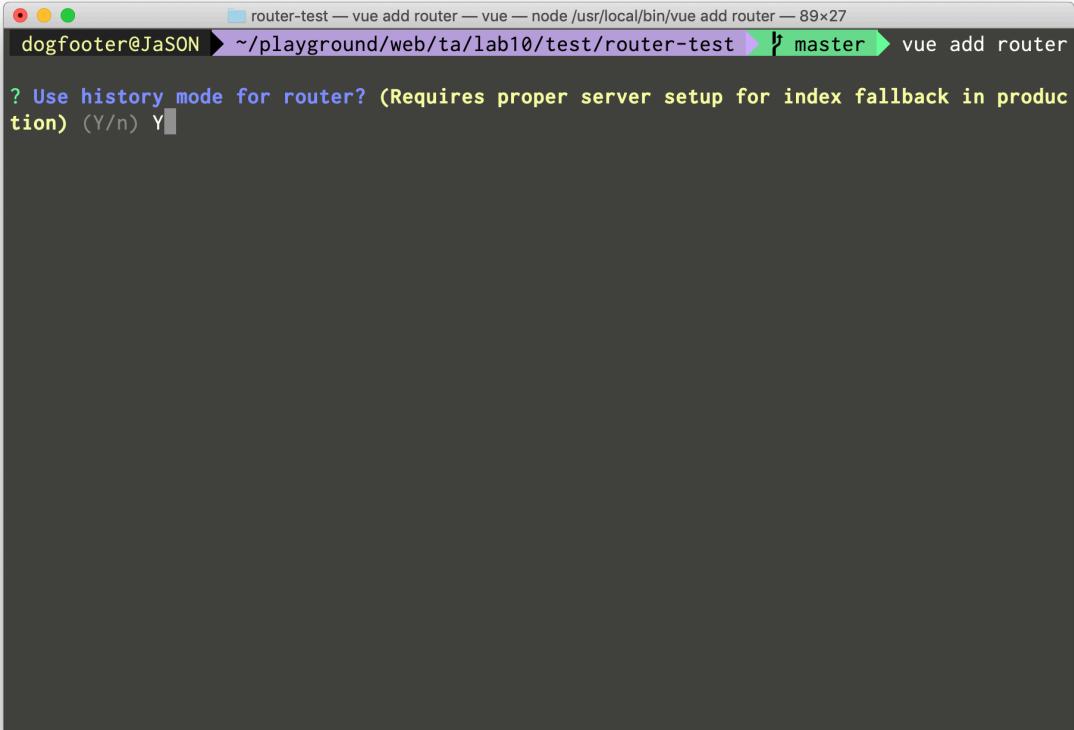
■ Vue.js router

- Vue.js 는 vue-router 를 통해 routing 을 구현할 수 있다.
- Router 객체를 예, 사용자 요청 path 와 해당 요청을 통해 보여줄 view(컴포넌트)를 정의한다.
- template 에 <router-link :to="/path/to/route">TEXT</route-link> 를 통해 routing 할 link 를 생성할 수 있다.
- 위 router-link 를 통해 요청한 routing 의 결과(view, 컴포넌트)는 <router-view/>에 rendering 된다.

■ vue-router 설치 및 적용

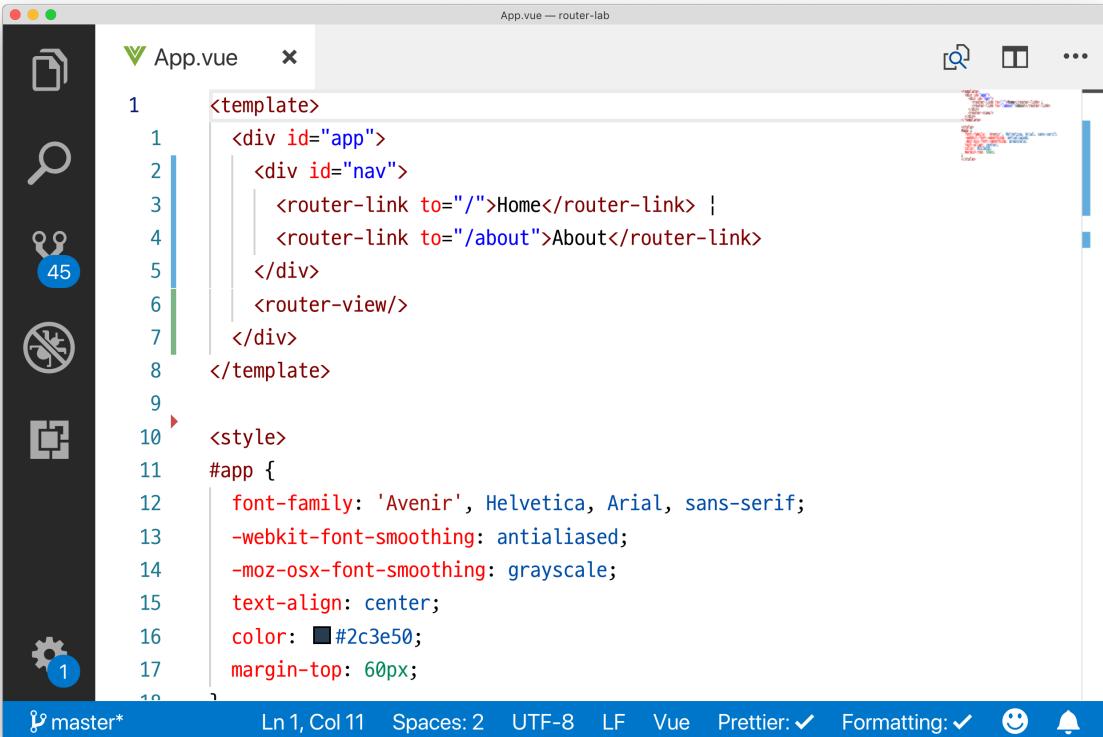
- vue-cli 를 통해서 쉽게 설치 및 적용할 수 있다.
- vue-cli 로 만든 프로젝트 디렉토리에서 아래의 console 명령을 입력한다.
 - ✓ vue add router

- 해당 명령을 통해 Router를 위해 필요한 router.js가 생성되고, main.js에 router를 위한 코드들이 추가된다.
- App.vue의 template에는 router-link와 router-view가 추가된다.



```
router-test — vue add router — vue — node /usr/local/bin/vue add router — 89x27
dogfooter@JaSON ~ playground/web/ta/lab10/test/router-test master ➜ vue add router
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) Y
```

```
src — dogfooter@JaSON — ..uter-test/src — -zsh — 89x27
dogfooter@JaSON ~ /playground/web/ta/lab10/test/router-test/src ↵ master • ? ➜ 11
total 24
-rw-r--r-- 1 dogfooter staff 415B Nov 28 00:14 App.vue
drwxr-xr-x 3 dogfooter staff 96B Nov 28 00:12 assets
drwxr-xr-x 3 dogfooter staff 96B Nov 28 00:12 components
-rw-r--r-- 1 dogfooter staff 175B Nov 28 00:14 main.js
-rw-r--r-- 1 dogfooter staff 581B Nov 28 00:14 router.js
drwxr-xr-x 4 dogfooter staff 128B Nov 28 00:14 views
dogfooter@JaSON ~ /playground/web/ta/lab10/test/router-test/src ↵ master • ? ➜
```



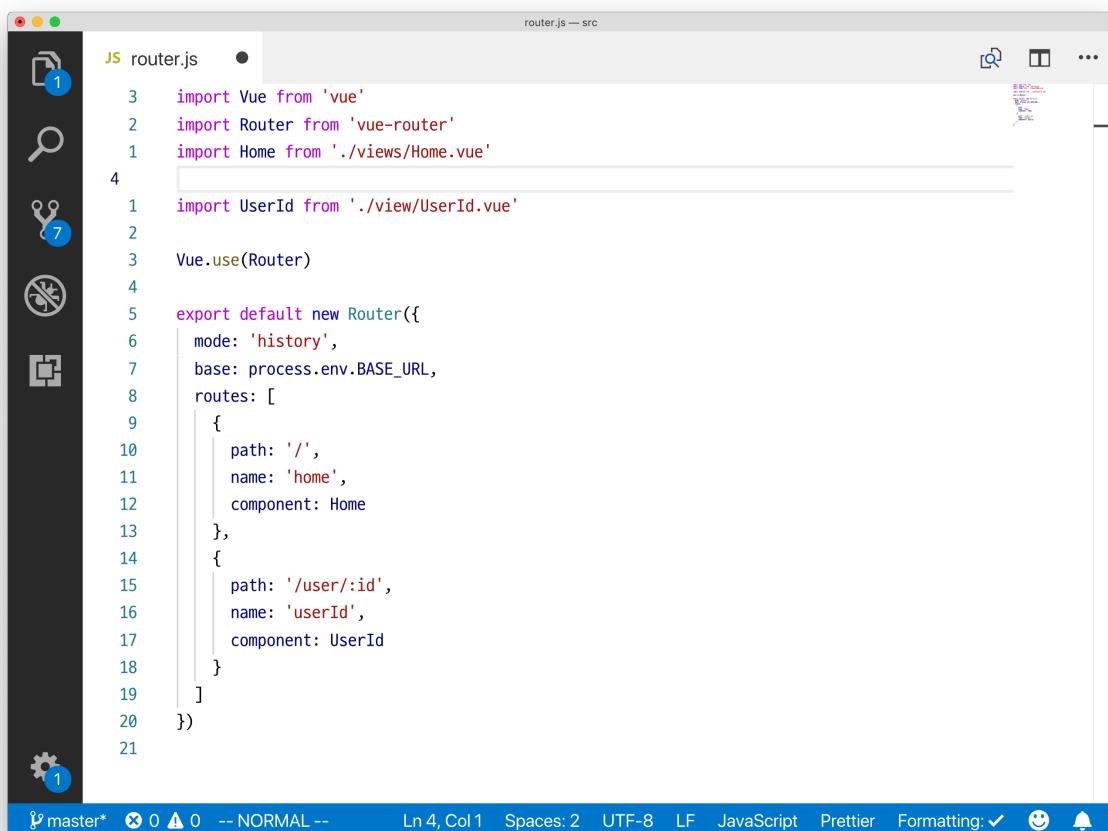
```
App.vue — router-lab
template>
<div id="app">
  <div id="nav">
    <router-link to="/">Home</router-link> |
    <router-link to="/about">About</router-link>
  </div>
  <router-view/>
</div>
</template>

<style>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
```

The screenshot shows a code editor window with the file "App.vue" open. The code defines a Vue component with a template section containing a navigation bar and a router view, and a corresponding style section with CSS rules for the "app" element. The editor interface includes a sidebar with icons for file operations, a status bar at the bottom, and a terminal tab above the code area.

■ Routing 선언

- src/router.js 에 routing 을 할 path 와 해당 path 의 결과로 반환할 컴포넌트를 선언한다.
- 하나의 새로운 routing 을 선언할 때 필요한 값들은 다음과 같다.
 - ✓ path: routing 할 path
 - ✓ name: 해당 routing 의 이름
 - ✓ component: 해당 routing 이 반환할 component



The screenshot shows a code editor window with the file 'router.js' open. The code defines a Router instance with two routes: one for the home page ('/home') and one for a user profile ('/user/:id'). The 'home' route uses the 'Home' component, and the 'user/:id' route uses the 'UserId' component.

```
router.js — src
JS router.js ●
1 1
3 import Vue from 'vue'
2 import Router from 'vue-router'
1 import Home from './views/Home.vue'
4
1 import UserId from './view/UserId.vue'
2
3 Vue.use(Router)
4
5 export default new Router({
6   mode: 'history',
7   base: process.env.BASE_URL,
8   routes: [
9     {
10       path: '/',
11       name: 'home',
12       component: Home
13     },
14     {
15       path: '/user/:id',
16       name: 'userId',
17       component: UserId
18     }
19   ]
20 })
21
21 1
```

master* 0 0 0 -- NORMAL -- Ln 4, Col 1 Spaces: 2 UTF-8 LF JavaScript Prettier Formatting: ✓ 😊 🚨

■ Routing link 와 view

- <router-link :to="/path/to/route">TEXT</route-link> 를 통해 routing 할 link 를 생성할 수 있다.
- 위 router-link 를 통해 요청한 routing 의 결과(view, 컴포넌트)는 <router-view/>에 rendering 된다.
- 아래는 App.vue 에서 router-link 와 router-view 를 사용한 예시이다.

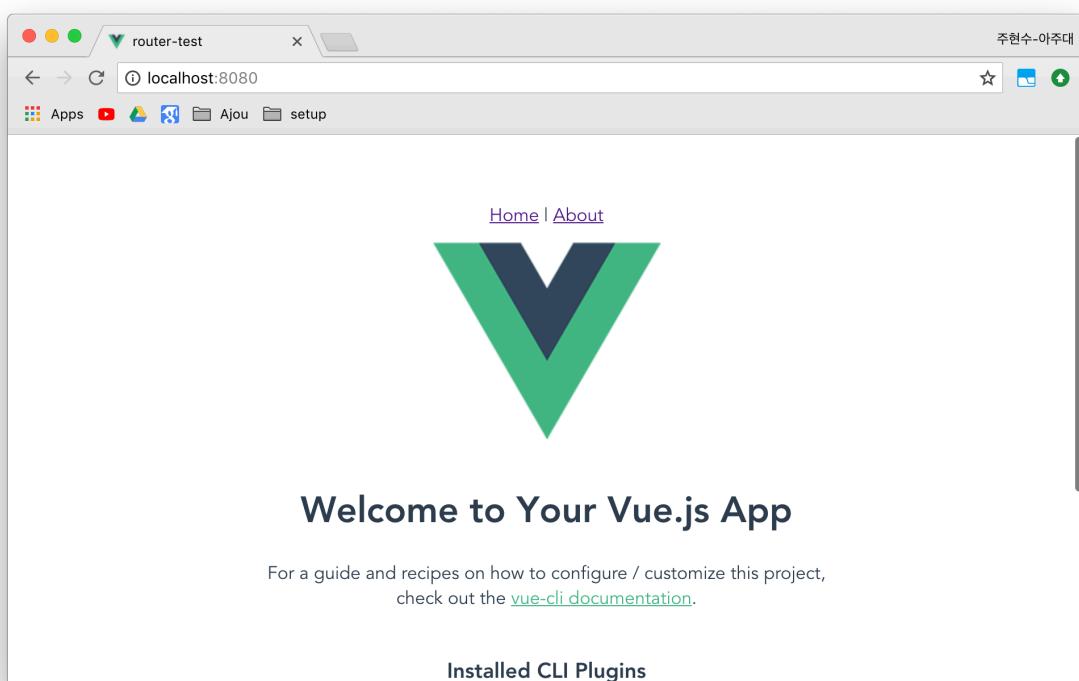
The screenshot shows the VS Code interface with the following details:

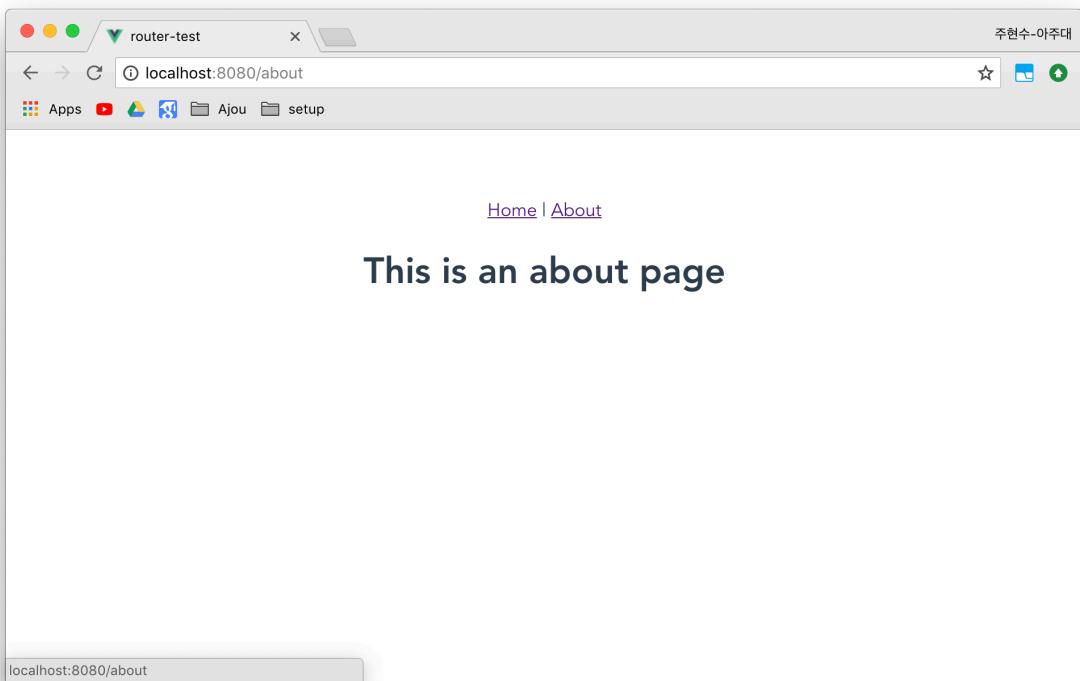
- EXPLORER** sidebar: Shows files in the project structure: OPEN EDITORS (App.vue), SRC (assets, components, views), and JS files (main.js, router.js).
- App.vue** editor tab: The active file contains the following Vue template and CSS code.
- Code Content:**

```
<template>
<div id="app">
  <div id="nav">
    <router-link to="/">Home</router-link> |
    <router-link to="/about">About</router-link>
  </div>
  <router-view/>
</div>
</template>

<style>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

- Bottom Status Bar:** Shows the current branch (master*), file status (0 changes), normal mode, line (Ln 10), column (Col 1), spaces (Spaces: 2), encoding (UTF-8), line separator (LF), Vue mode, Prettier status (✓), and other icons.





■ Dynamic Routing

- 같은 컴포넌트에 다양한 데이터가 들어올 수 있다(가령 게시글을 보여주는 컴포넌트). 이때, 해당 routing path 를 처리할 수 있는 routing 방법이다.
- Routing 을 선언할 때, path 값에 /user/:id 처럼 가변하는 parameter 앞에 :을 붙인다.
- Dynamic Routing 의 <routing-link>를 정의하는 방법은 다음과 같다.
 - ✓ :to 에 routing 의 name 과 parameter 에 들어갈 값을 정의한다.
(아래 그림 참조)
- Routing 의 결과로 반환된 component 의 script 에서 해당 parameter 의 값을 받아올 수 있다.
 - ✓ this.\$router.params 의 property 로 해당 parameter 를 받아올 수 있다.
 - ✓ /user/:id 의 경우는 this.\$router.params.id 로 받아올 수 있다.

- 아래는 router-link 를 정의한 코드와 해당 routing 을 선언한 코드이다.

```

App.vue
1 <template>
2   <div>
3     <div v-if="hasObjs">
4       <ul>
5         <li v-for="obj in objs" :key="obj.id">
6           <router-link :to="{name: 'OBJSID', params: {id: obj.id}}>{{obj.title}}</router-link>
7         </li>
8       </ul>
9     </div>
10    </template>

```

Ln 2, Col 7 Spaces: 2 UTF-8 LF Vue Prettier Formatting: ✓

```

JS router.js
1 const routes = [
2   {
3     name: "OBJSID",
4     component: OBJSID
5   },
6   {
7     path: "/objs/:id",
8     name: "OBJSID",
9     component: OBJSCON
10   }
11 );

```

Ln 34, Col 24 Spaces: 2 UTF-8 LF JavaScript Prettier Formatting: ✓

- 아래는 script 에서 Dynamic routing parameter 값을 참조하는 코드이다

```

OBJSCON.vue
1 <template>
2
3 <script>
4 export default {
5   name: "OBJSCON",
6   data: function() {
7     return {
8       obj: { id: -1 },
9       id: this.$route.params.id
10     };
11   },
12   computed: {
13     ...
14   }
15 }

```

Ln 31, Col 14 (1 selected) Spaces: 2 UTF-8 LF Vue Prettier: ✓ Formatting: ✓

Lab Assignment

■ 실습 과제 목표

- Vue-router 를 활용하여 게시글을 열람할 수 있는, SPA 기반 Web App 을 개발한다.

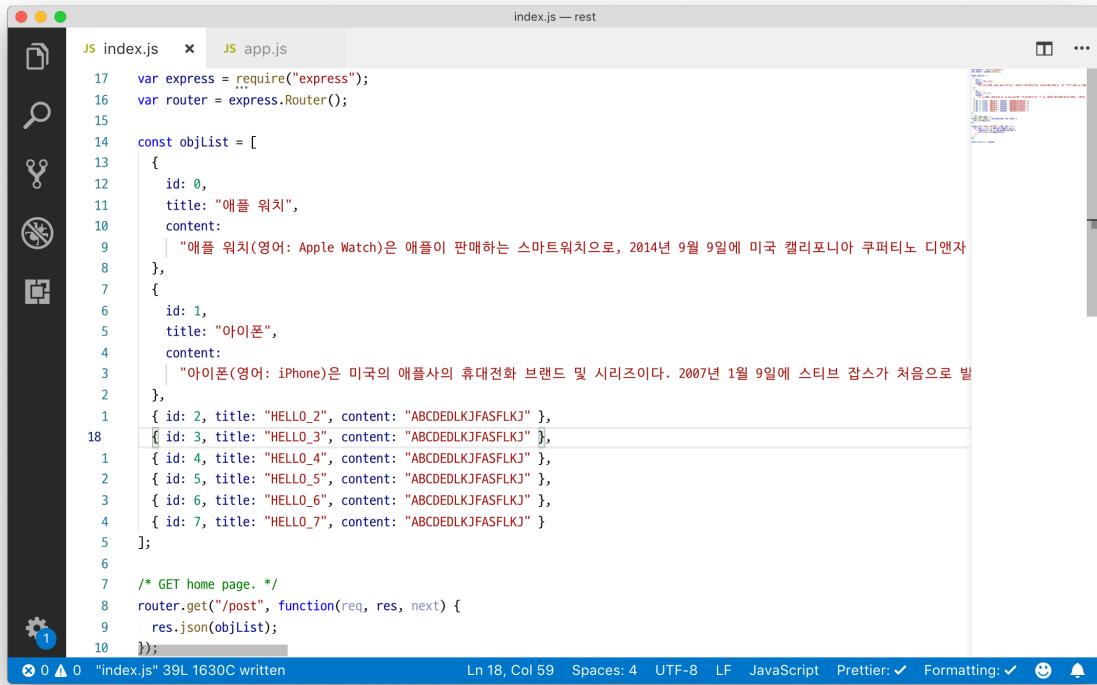
■ 프로젝트 아키텍쳐

- Web App 을 구성하기 위해, SPA Frontend 를 위한 Vue 서버 (vue-cli: npm run serve)와 동적 데이터 제공을 위해 Node.js 기반 REST 서버를 구축한다.
- 과제 2 와 같이 Frontend 와 Backend 두 가지 프로젝트가 따로 존재한다.
- Frontend
 - ✓ @vue/cli v3.0 이상에서 프로젝트를 생성한다.
 - 프로젝트 이름은 lab10-201820000-front 이다.
 - ✓ REST 서버로부터 **게시글 리스트**를 받아와 게시글을 열람할 수 있는 **게시글 제목** 리스트를 보여준다.
 - ✓ 하나의 **게시글 제목**을 클릭하면, 해당 게시글의 제목과 내용을 보여준다.
 - ✓ REST 서버와의 통신은 Ajax 를 사용한다.
- Backend 는 다음과 같은 RESTful API 를 처리한다.
 - ✓ Node.js + Express 프로젝트이다.
 - 프로젝트 폴더 이름은 lab10-201820000-REST 이다.
 - ✓ GET /post
 - 전체 **게시글 리스트**를 반환한다.
 - ✓ GET /post/:postId
 - postId 를 id 로 갖는 **게시글을** 반환한다.

■ 게시글

- REST 서버에서 게시글 리스트를 변수로 따로 관리한다.
- 게시글 하나의 형태는 다음과 같다.
 - ✓ { id: Number, title: String, content: String }

- REST 서버에 다음과 같은 방식으로 변수를 추가하여, 게시글 리스트나 특정 id 를 갖는 게시글을 반환할 때, 이 변수를 참조하여 처리한다.



The screenshot shows a code editor window with the file 'index.js' open. The code defines an array of objects representing posts. Each post has an 'id', 'title', and 'content'. The content includes some Korean text and English text about Apple Watch and iPhone. The code editor interface includes a sidebar with icons, a status bar at the bottom, and a floating terminal window on the right.

```

JS index.js x JS app.js
17 var express = require("express");
16 var router = express.Router();
15
14 const objList = [
13 {
12   id: 0,
11   title: "애플 워치",
10   content:
9     "| 애플 워치(영어: Apple Watch)은 애플이 판매하는 스마트워치으로, 2014년 9월 9일에 미국 캘리포니아 쿠퍼ти노 디엔자
8   },
7   {
6     id: 1,
5     title: "아이폰",
4     content:
3       "| 아이폰(영어: iPhone)은 미국의 애플사의 휴대전화 브랜드 및 시리즈이다. 2007년 1월 9일에 스티브 잡스가 처음으로 빌
2   },
1   { id: 2, title: "HELLO_2", content: "ABCDEDLKJFASFLKJ" },
18   { id: 3, title: "HELLO_3", content: "ABCDEDLKJFASFLKJ" },
1   { id: 4, title: "HELLO_4", content: "ABCDEDLKJFASFLKJ" },
2   { id: 5, title: "HELLO_5", content: "ABCDEDLKJFASFLKJ" },
3   { id: 6, title: "HELLO_6", content: "ABCDEDLKJFASFLKJ" },
4   { id: 7, title: "HELLO_7", content: "ABCDEDLKJFASFLKJ" }
5 ];
6
7 /* GET home page. */
8 router.get("/post", function(req, res, next) {
9   res.json(objList);
10 });

```

Ln 18, Col 59 Spaces: 4 UTF-8 LF JavaScript Prettier: ✓ Formatting: ✓ 😊 🔔

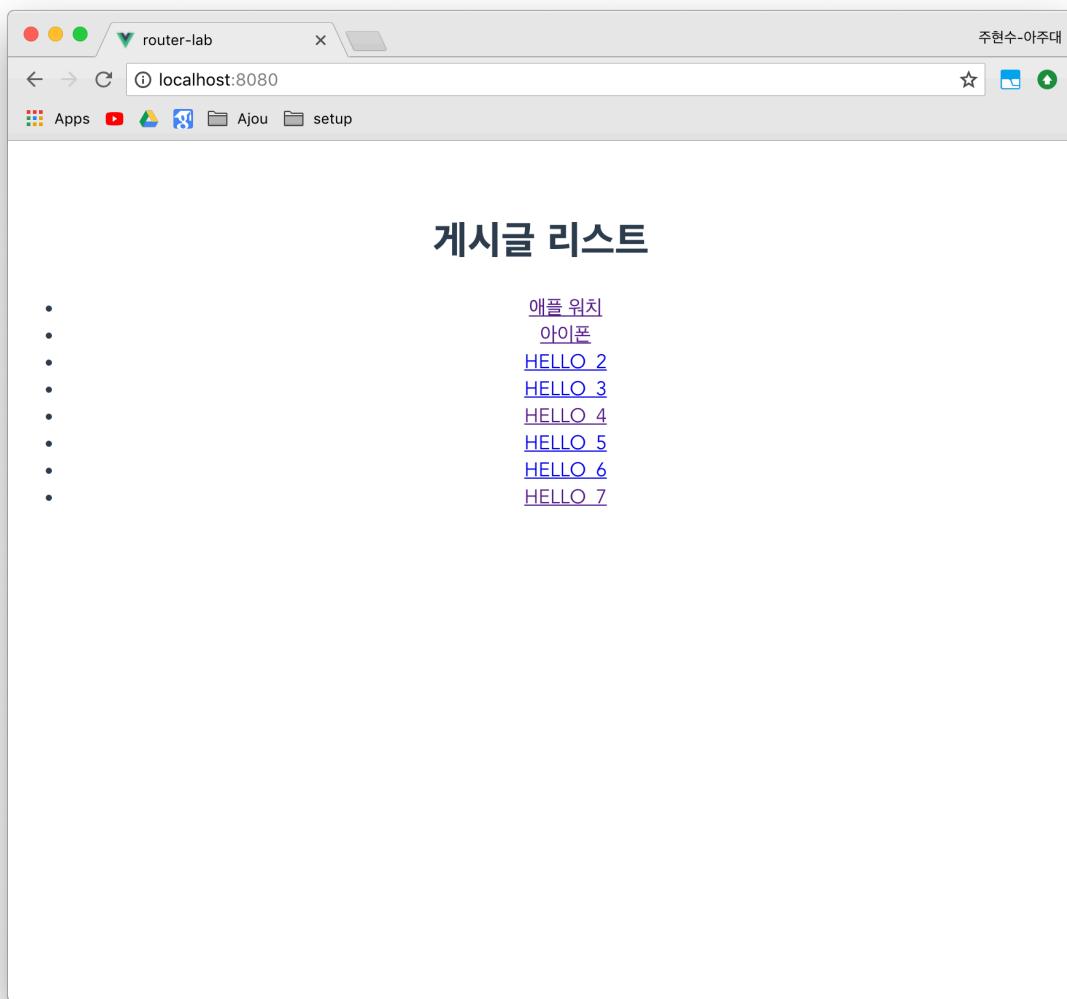
■ Vue component 구현

- App
 - ✓ Template
 - Ajax 를 통해 게시글 리스트를 받아오면, 게시글의 개수만큼 **게시글 제목 리스트를 rendering 한다.**
 - 각 제목은 <router-link>를 통하여 /post/:postId 로 routing 한다.
 - Routing 된 결과로 반환될 component 를 rendering 할 <router-view>를 갖고 있다.
 - ✓ Script
 - Ajax 를 통해 게시글 리스트를 받아온다.
- Post
 - ✓ /post/:postId 를 통해 routing 되어 반환되는 component 이다.
 - ✓ Template
 - Ajax 를 통해 게시글을 받아오면 해당 **게시글의 제목과 내용을 rendering 한다.**
 - 게시글 제목을 보여줄 h1

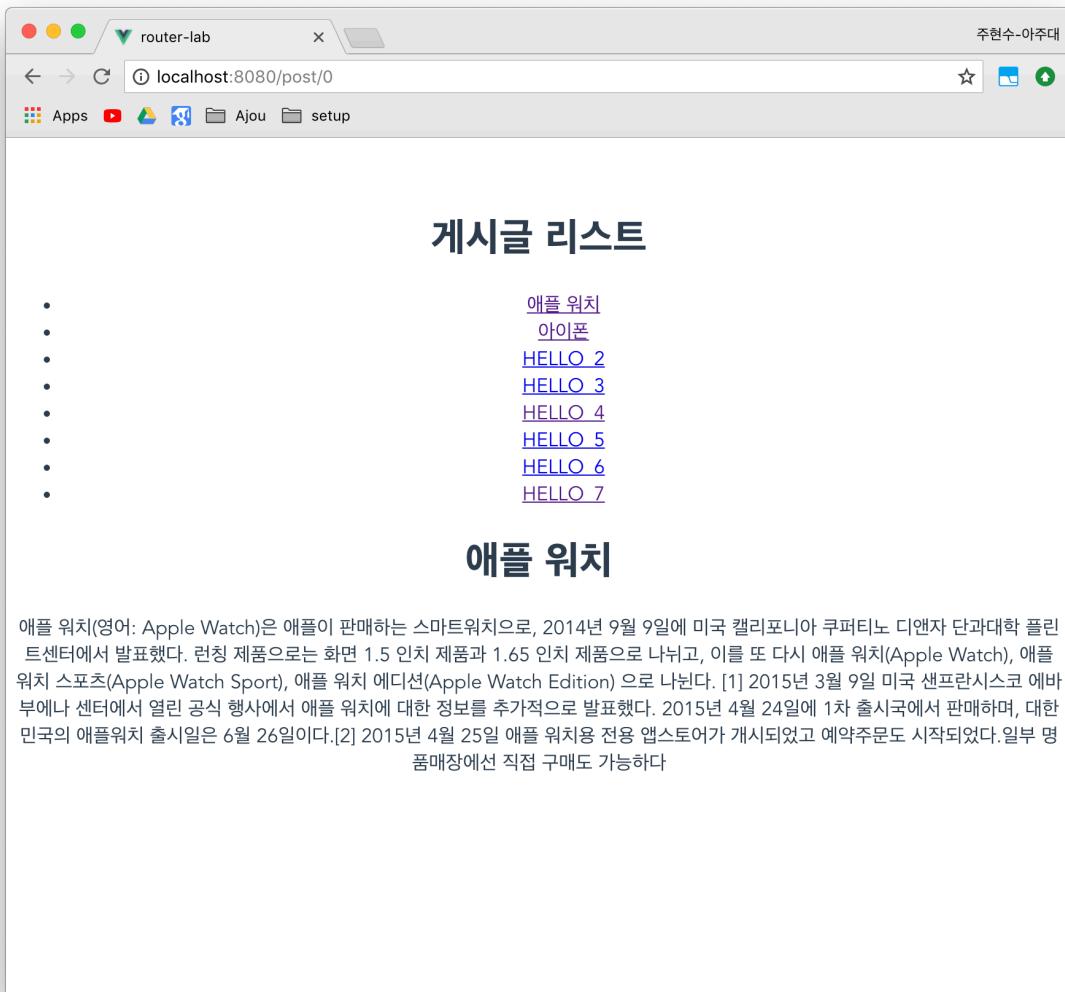
- 게시글 내용을 보여줄 p
- ✓ Script
 - Ajax를 통해 postId를 id로 갖는 게시글을 받아온다.

■ 실행 예시

- 처음 실행 시 보이는 화면



- 게시글 제목을 하나 클릭했을 때 화면



■ [필수] Backend(Node.js) code 추가 사항

- Node.js + Express로 서버를 구성할 때, 아래 코드를 꼭 추가시켜야 한다.
- Express app 객체가 사용되는 코드(보통 app.js에 Express app을 선언한다)에 다음과 같은 코드를 추가한다.

```
1. app.use((req, res, next) => {
2.   res.header("Access-Control-Allow-Origin", "*");
3.   res.header("Access-Control-Allow-Methods", "GET, PUT, POST, DELETE, OPTIONS");
4.   res.header("Access-Control-Allow-Headers", "X-Requested-With, Content-Type");
5.   next();
6. });
```

The screenshot shows a code editor window titled "app.js — rest". The editor has tabs for "index.js" and "app.js", with "app.js" currently active. The code in "app.js" is as follows:

```
var app = express();
app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Methods", "GET, PUT, POST, DELETE, OPTIONS");
  res.header("Access-Control-Allow-Headers", "X-Requested-With, Content-Type");
  next();
});
app.use(logger("dev"));
```

The code editor interface includes a sidebar with icons for file, search, and settings, and a status bar at the bottom showing "Ln 17, Col 1" and other file details.

■ 제출양식

- REST 서버 프로젝트와 Vue 프로젝트 폴더를 **lab10-{학번}** 폴더에 넣는다.
 - ✓ Ex) lab10-201820000
- 제출하는 압축파일의 이름은 **lab10-{학번}.zip** 으로 한다
 - ✓ Ex) lab10-2018200000.zip
- 명시된 파일명, 폴더명을 반드시 지키기 바랍니다. (감점요인)

■ 제출기한

- 금요일 5:59 am 까지 제출: 100% 점수인정
- 금요일 5:59 pm 까지 제출: 70% 점수인정
- 그 이후 제출 시 0%

■ 주의사항

- 모든 과제는 본인이 수행한 결과물만 제출
- 처음 Copy 적발 시 해당 과제 0점 처리
- Copy 재적발 시 해당 과제 전체점수 0점 처리