프론트엔드 개발자를 위한 Vue.js 시작하기

Vue.js 시작하기

Router

한국기술교육대학교
온라인평생교육원

학습내용

- Single Page Application
- vue-router

학습목표

- Single Page Application에 대해 설명할 수 있다.
- vue-router 사용 방법에 대해 설명할 수 있다.

Single Page Application

Single Page Application

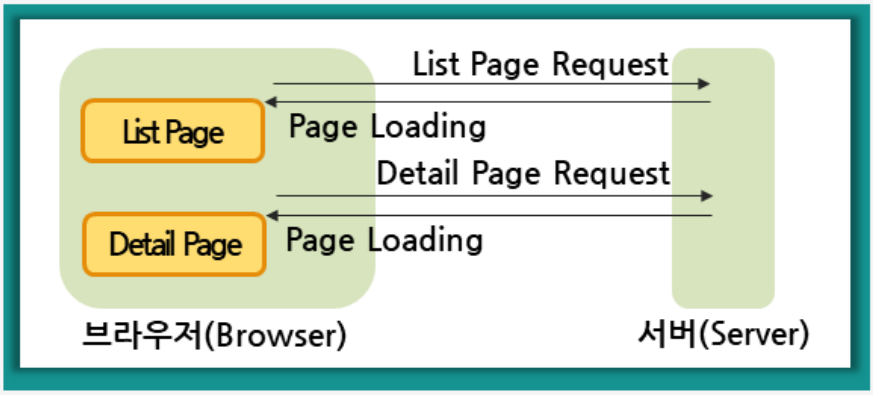
1 Single Page Application

- Single Page Application(SPA)는 하나의 페이지에서 애플리케이션의 여러 화면을 제공하는 웹 애플리케이션을 의미함
- Multi Page Application의 반대 개념으로 이용됨

Single Page Application

1 Single Page Application

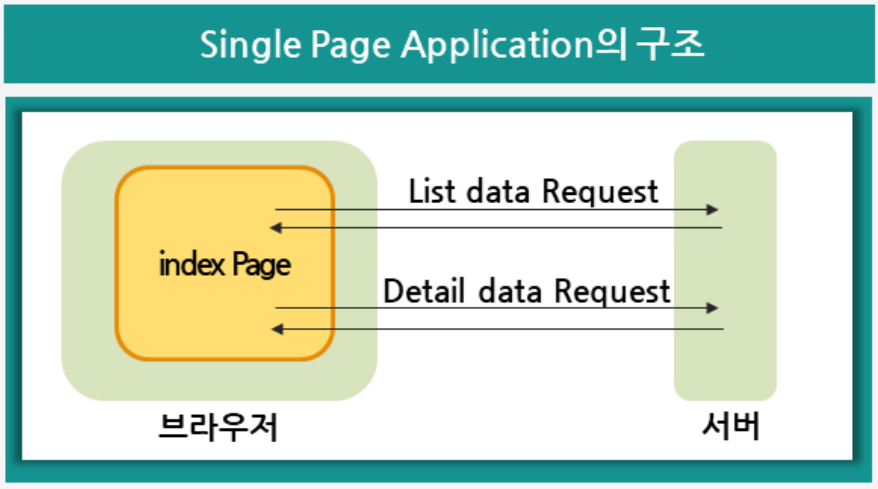
Multi Page Application의 구조



Single Page Application

Single Page Application

1 Single Page Application



Single Page Application

2 Routing

Routing

- 특정 화면에서 다른 화면으로의 이동을 의미함
- Single Page Application이 되려면 Routing이 물리적인 파일 로딩에 의한 화면전환이 아닌 하나의 페이지내에서 논리적인 화면전환이 되어야 함

Single Page Application

Single Page Application

2 Routing

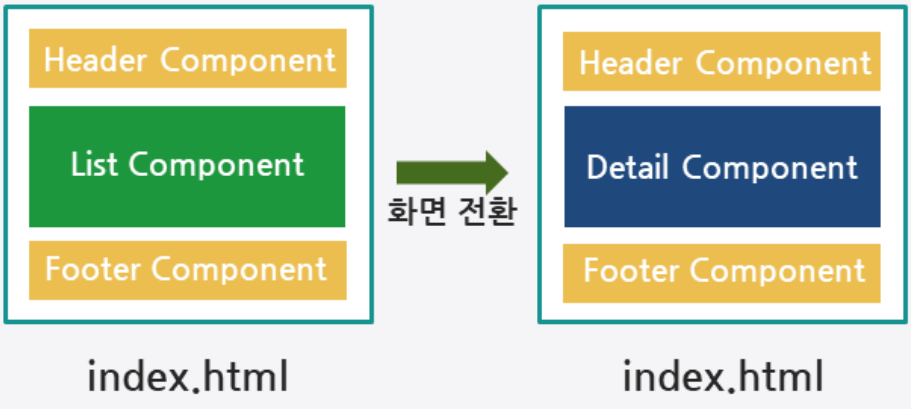
Routing

- 각 화면(혹은 화면의 구성요소)이 Component로 개발되어 있어야 함

Single Page Application

2 Routing

- Client Side(Javascript)에서 어떤 조건에서 어떤 화면(Component)이 출력되어야 하는지 결정되어야 함
- 이 부분을 지원하기 위한 라이브러리가 Router임



vue-router

vue-router

1 vue-router 기본

① Router 설치

- Router를 이용하기 위해서는 vue-router를 설치해야 함
- vue-router는 Vue.js의 공식 라이브러리임

```
>npm install vue-router@4
```

vue-router 4 버전	Vue.js 3 버전을 위한 Router 라이브러리
vue-router 3 버전	Vue.js 2 버전을 위한 Router 라이브러리

vue-router

1 vue-router 기본

② Routing 조건

- URL의 Path 조건
- http://localhost:8080/이라면 Routing조건은 /임
- http://localhost:8080/login이라면 Routing 조건은 /login임

vue-router

vue-router

1 vue-router 기본

② Routing 조건

⋯› Routing 조건을 명시한 배열을 준비함

path	Routing 조건
component	Routing 조건이 맞을 때 출력할 Component

vue-router

1 vue-router 기본

② Routing 조건

```
const routes = [
    { path: '/', component: Home },
    { path: '/login', component: LoginPage },
    { path: '/detail', component: DetailPage },
]
```

vue-router

vue-router

1 vue-router 기본

③ Router Instance 생성

- createRouter()함수를 이용해 Router Instance를 생성함
- routes option에 Routing 조건을 등록해 줌

```
const router = createRouter({
    history: createWebHistory(),
    routes
})
```

vue-router

1 vue-router 기본

④ Router Instance 사용 선언

- application instance의 use() 함수로 router instance 등록

```
app.use(router);
```

vue-router

vue-router

1 vue-router 기본

⑤ router-view component 등록

⋯› Router에 의해 출력되는 Component는 <router-view/>에 출력됨

```
<router-view></router-view>
```

vue-router

1 vue-router 기본

⑥ router-link

⋯› 화면에 출력되는 라우팅을 위한 링크

```
<router-link to="/login">Go to Login</router-link>
```

vue-router

vue-router

1 vue-router 기본

⑦ $router.push

- Routing에 의해 출력되는 Component에서는 $router 객체로 Routing과 관련된 다양한 데이터 및 함수 이용
- $router.push() 함수로 특정 path 조건의 Component로 이동

```
this.$router.push('/login')
```

vue-router

2 vue-router parameter

① parameter 선언

- parameter 값을 routing 조건에 추가할 때는 path에 콜론(:)을 추가해 명시함

```
{ path: '/detail/:title', component: DetailPage }
```

vue-router

vue-router

2 vue-router parameter

① parameter 선언

- path 조건의 콜론(:)은 임의의 문자열 하나를 의미함
- 콜론(:) 뒷 문자열은 임의 문자열이며 전달되는 데이터의 key 값으로 사용됨
- 콜론(:)에 의한 조건은 하나의 path 내에 여러 번 나올 수 있음

vue-router

2 vue-router parameter

① parameter 선언

path	Matched path	전달 데이터
/users/:username	vue-router parameter	vue-router parameter
/users/:username/posts/:postId	/users/kim/posts/123	{username: 'kim', postId: 123}

vue-router

vue-router

2 vue-router parameter

② parameter 획득

··› Routing 조건에 의해 실행되는 Component에서는 parameter 값을 $route.params 객체로 획득함

```
title : {{$route.params.title}}
```

vue-router

3 vue-router 중첩

① Nested Routing

··› Routing 조건이 복잡한 경우 Routing 조건을 중첩에 의해 선언이 가능함

··› Routing 조건에 의해 실행된 Component부터 다시 Routing 조건을 판단함

vue-router

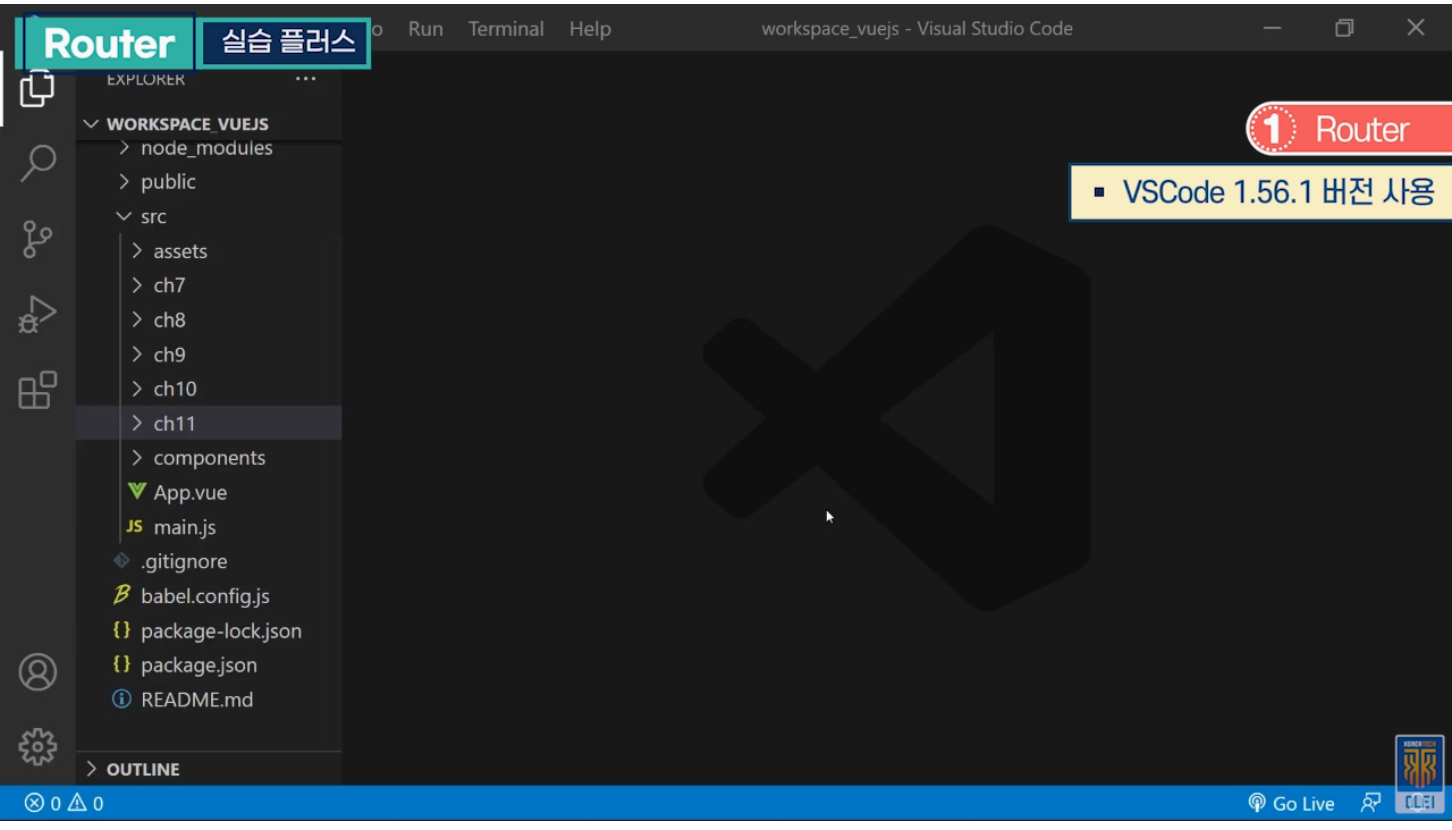
vue-router

3 vue-router 중첩

① Nested Routing

```
const router = new VueRouter({
  routes: [
    { path: '/user/:id', component: User,
      children: [
        {
          ///user/:id/profile is matched
          path: 'profile',
          component: UserProfile
        },
        {
          ///user/:id/posts is matched
          path: 'posts',
          component: UserPosts
        }
      ]
    }
  ]
})
```

실습 플러스



실습단계
src → ch12, Router를 테스트하고자 하는 것이 주목적임
Router 설치, npm CLI 명령어를 통하여 Router 모듈 추가 설치
프로젝트에서 npm 명령을 내려주면 됨
ch12 폴더에 Component 정의, Router : Component를 교체하면서 화면을 전환하고자 하는 것이 주목적
New → File에서 첫 화면으로 쓰고자 하는 Component, Home.vue 선언
template, script 정의 및 작성
배열 데이터로 목록 화면 출력, 목록에 각각의 문자열을 출력시키기 위해서 작성한 것
이벤트 발생에 의해서 호출되며, Router로 화면 전환
화면을 꾸미기 위해 template 정의
배열 데이터를 v-for로 나열, 항목이 나열되고 클릭할 때 event 함수가 호출되게 화면 구성
ch12 폴더에 Login.vue 정의
Component의 이름 정도만 화면에 출력되게끔 작성
ch12 폴더에 Detail.vue 작성

실습 플러스

실습단계
화면에 식별을 위해 title 작성
Home.vue의 Component가 나오고 항목을 눌렀을 때 Detail 화면이 나오도록 함 → 이전 화면에서 전달된 데이터를 받을 수 있도록 테스트함
Router의 내장객체 : 내장객체를 이용하여 전달한 데이터를 가지고 있는 변수 params로 데이터 획득
세 개의 Component를 Router에 등록, main.js에서 작업
이전에 생성한 ch11의 index.html, main.js, vue.config.js 파일 복사하여 이용
Router에서 제공되는 API를 import 받음, Component도 import하여 Home으로 정의
Component 간 화면관리, 화면전환을 Router로 함
Router에 등록할 정보 준비
첫 번째 path 정보, 어떤 path가 들어왔을 때 Component를 실행시킬 것인지에 대한 정보
'/'로 들어왔을 때 실행될 Component
/login으로 path가 들어오면 Login Component 출력
Detail path의 ':' 뒤에 임의의 단어 하나는 title이란 변수로 저장, Component는 DetailPage
정보가 Router에 추가되어야 하므로 API로 Router 생성
Router에 정보가 포함되므로 Router에 의해 정보에 맞는 화면이 출력됨
Router 객체가 애플리케이션에 등록되어 있는 상태
index.html에 Router가 나와야 할 위치 지정
테스트를 위해 링크 준비
화면에 go home이 나옴, 클릭했을 때 path 정보 값이 '/'로 바뀌고 Router가 캐치하여 '/'에 의해 Component 출력
Vue 명령으로 실행
브라우저에 Component 출력
Router에 의한 Component가 출력된 부분, Router에 의해서 Detail Page Component로 전환됨