KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

vue-router를 이용한 라우팅 1

[KB] IT's Your Life

요청이 왔을 때 담당하는 주체를 찾아주는 것을 라우팅이라 한다.

네트워크에서의 라우팅과 약간 차이가 있죠?

요청 url을 어느 컴포넌트가 처리하느냐?
그 컴포넌트한테 처리를 위임함

KB 국민은행

1 vue-router란

SPA(단일 페이지 애플리케이션: Single Page Application)

- 하나의 페이지 안에서 데스크톱 애플리케이션과 같은 사용자 경험을 제공
- 여러 화면을 하나의 페이지 안에서 제공, 화면을 별도로 로딩하지 않음

→ 화면마다 고유의 식별자(URI)를 기반으로 화면을 렌더링해야 함

- 요청한 URI 경로에 따라 각각 다른 화면이 렌더링되도록 함

→ vue-router 라이브러리 이용

이 기능은 vue router라는 추가적인 라이브러리가 필요하다

createRouter()

createRouter 라우터를 생성

- router 객체를 생성
  - URI와 이 경로를 처리할 컴포넌트 매핑

```
import { createRouter, createWebHistory } from 'vue-router'
...
const router = createRouter({
    history: createRouter(),
    routes: [
        { path: '/', components: Home },
        { path: '/about', components: About },
        { path: '/members', components: Members },
        { path: '/videos', components: Videos},
    ]
});
```

라우팅 테이블
경로와 도착지정보를 모아놓은 테이블 자료구조

이 정보들로 라우터 만들겠다

라우팅 테이블 정보

하나의 라우트 형식 :
path: URL, componentes: 담당하는 컴포넌트.

router 객체의 등록

```
const app = createApp(App)
app.use(router)
app.mount('#app')
```

app의 use메소드를 통해 만든 router 등록

2 vue-router의 기본 사용법

<RouterView>

- 각 경로별 컴포넌트를 렌더링할 위치를 지정하는 컴포넌트

```
<template>
  <div class="container">
    <Header />
    <RouterView/>
  </div>
</template>
```

app.vue에서 활용됨
app.vue의 주역할이
레이아웃 잡는 역할

이 부분이 계속 동적으로
바뀌는 부분

나머지 영역을 URL에 따라서 교체하는 컴포넌트

화면 전환시 페이지 이동<a> 을 하지 않고 index.html만 사용.
ajax활용해서 화면 전환

<RouterLink to="경로">

a태그와 비슷한 역할을하지만 페이지 이동하지 않는

<RouterLink> 태그를 활용

- 화면 전환 을 위한 링크 생성

```
<router-link to="[등록시킬 URI 경로]">[링크 텍스트]</router-link>
```

```
<RouterLink to="/">Home</RouterLink>
```

변수를 연결할 때는 바인딩하여 :to="~"로 표현하면 돼

실습 프로젝트 생성

npm init vue router-test

Vue.js - The Progressive JavaScript Framework

✓ Add TypeScript? ... No / Yes
✓ Add JSX Support? ... No / Yes
✓ Add Vue Router for Single Page Application development? ... No / Yes
✓ Add Pinia for state management? ... No / Yes
✓ Add Vitest for Unit Testing? ... No / Yes
✓ Add an End-to-End Testing Solution? » No
✓ Add ESLint for code quality? ... No / Yes
...

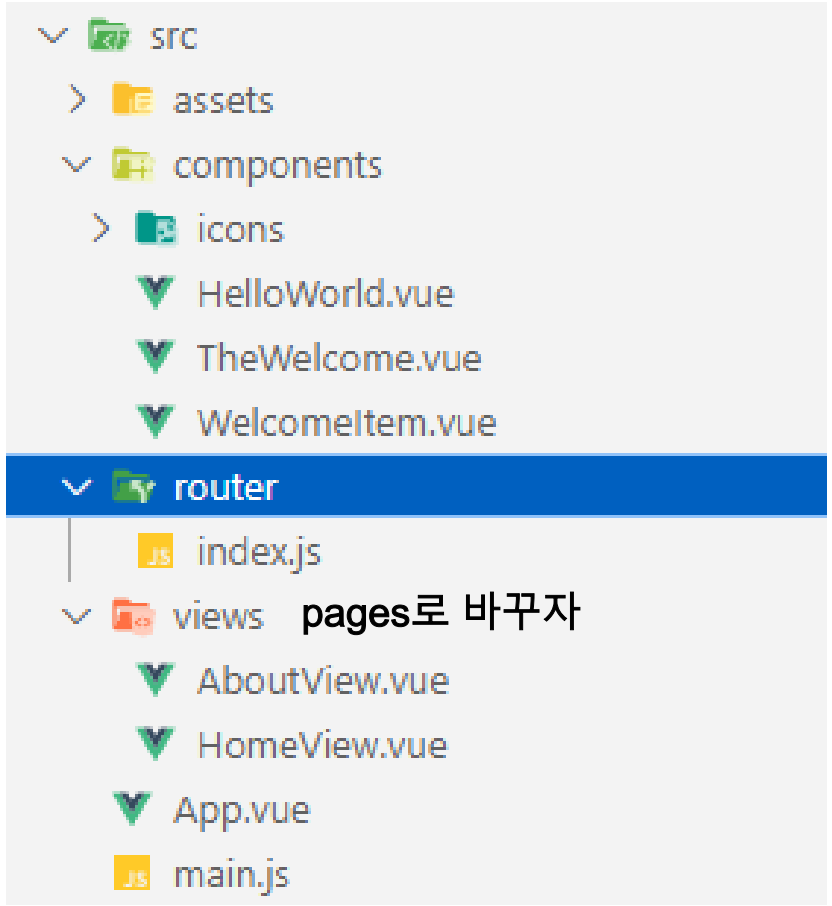
cd router-test
npm install

package.json

```
{
  "name": "router-test",
  "version": "0.0.0",
  "private": true,
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "vue": "^3.3.11",
    "vue-router": "^4.2.5"
  },
  "devDependencies": {
    "@vitejs/plugin-vue": "^4.5.2",
    "vite": "^5.0.10"
  }
}
```

2 vue-router의 기본 사용법

기본 골격



components에는 그냥 사용되는 컴포넌트

views==
pages에는 라우터태그에 들어갈 페이지 컴포넌트들

router 모듈
어떤 모듈은 여러가지 모듈들의 모음일 수 있음
이럴 땐 보통 디렉토리 명이 모듈명.
그리고 이럴 땐 보통 index.js 파일이 있어야 한다.
해당 모듈을 얻고 싶을 땐
import ~~ from ' 디렉토리명 ' 할 때 index.js를 읽는다

view아래 있는 것들은 페이지 컴포넌트다.
/about 페이지 담당
/home 페이지 담당

아까 위에서 봤떤 RouterView태그 있는 곳을 간다.

2 vue-router의 기본 사용법

src/router/index.js 자동으로 만들어진 기본골격

```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [ 수정대상
    {
      path: '/',
      name: 'home',
      component: HomeView
    },
    {
      path: '/about',
      name: 'about',
      // route level code-splitting
      // this generates a separate chunk (About.[hash].js) for this route
      // which is lazy-loaded when the route is visited.
      component: () => import('../views/AboutView.vue')
    }
  ]
})
export default router
```

이름은 home인 home페이지를 담당하는 HomeView 컴포넌트

여러개의 라우트 정보를 담고있는 라우트 테이블

이렇게 표현하면 지금 당장 로드해라라는 뜻

라우트 레벨에서 코드가 분리된다. 빌드 단계에서.

페이지별로 js가 생성된다.

컴포넌트를 이런식으로 로드

이렇게 표현하면 나중에 함수를 호출하여 로드해라 라는 뜻.
바로 하지말고 해당 url이 필요할 때.

빌드를 하게 되면

여러개였던 파일들이

index.html
xxx.js
xxxx.css 만 남아서 돌아간다.
하나 하나가 크기가
굉장히 커질 수 있다.

그렇게 된다면 하나의
페이지만 들어가야해도
다운 받아야하고
초기 로딩 시간이
너무 오래 걸리는 문제점이
있었다.

=> 큰 덩어리의 파일들을
분리하여 처음 로딩할 때는
필요한 부분만 먼저 로드.
라우팅 단위로 짜르겠다는 말

src/main.js

```
import './assets/main.css'

import { createApp } from 'vue'
import App from './App.vue'
import router from './router'

const app = createApp(App)

app.use(router)
app.mount('#app')
```

앱의 미들웨어로 등록

src/App.vue

```
<script setup>
import { RouterLink, RouterView } from 'vue-router'
import HelloWorld from './components/HelloWorld.vue'
</script>

<template>
  <header>
    <img alt="Vue logo" class="logo" src="@/assets/logo.svg" width="125" height="125" />

    <div class="wrapper">
      <HelloWorld msg="You did it!" />

      <nav>
        <RouterLink to="/">Home</RouterLink>
        <RouterLink to="/about">About</RouterLink>
      </nav>
    </div>
  </header>

  <RouterView />
</template>
<style scoped>
…
</style>
```

헤더

메뉴

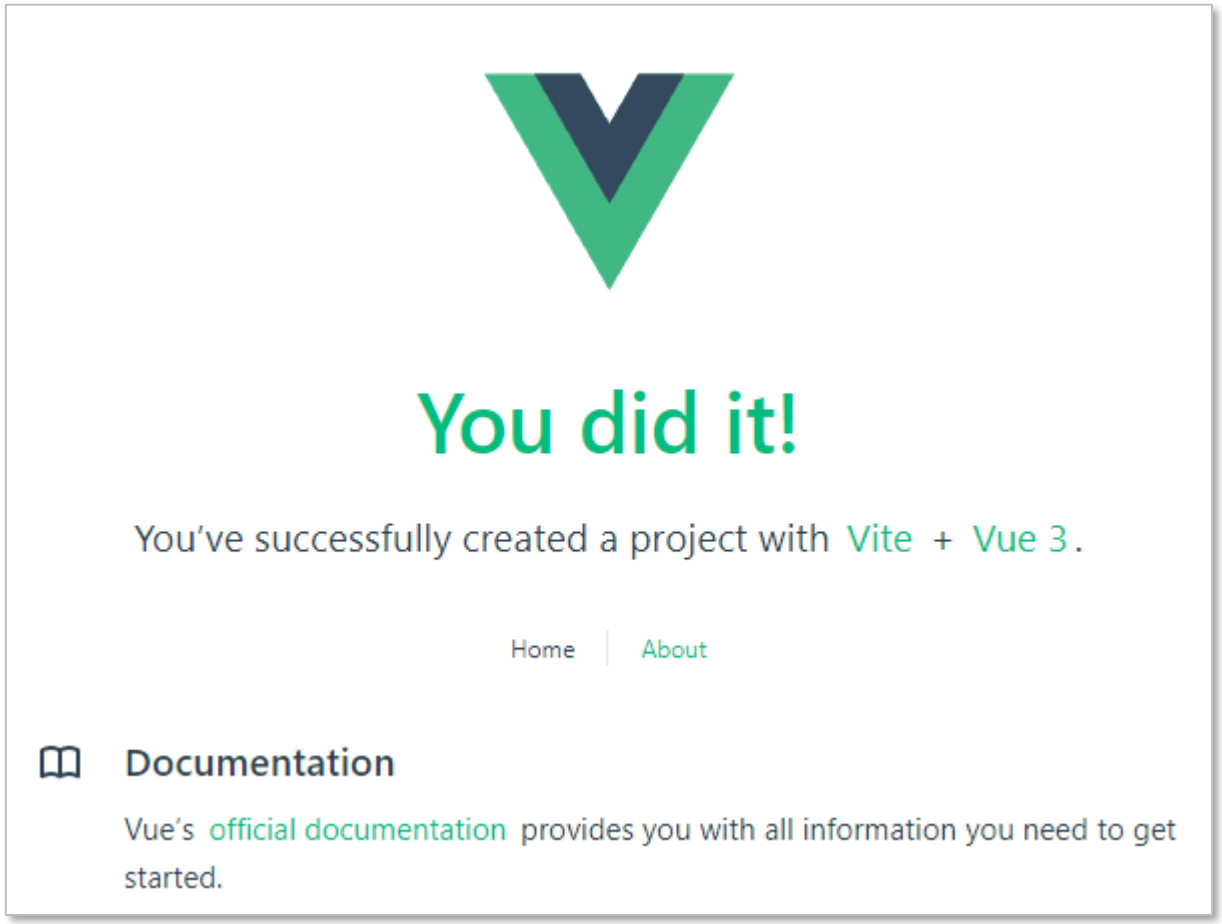
본문

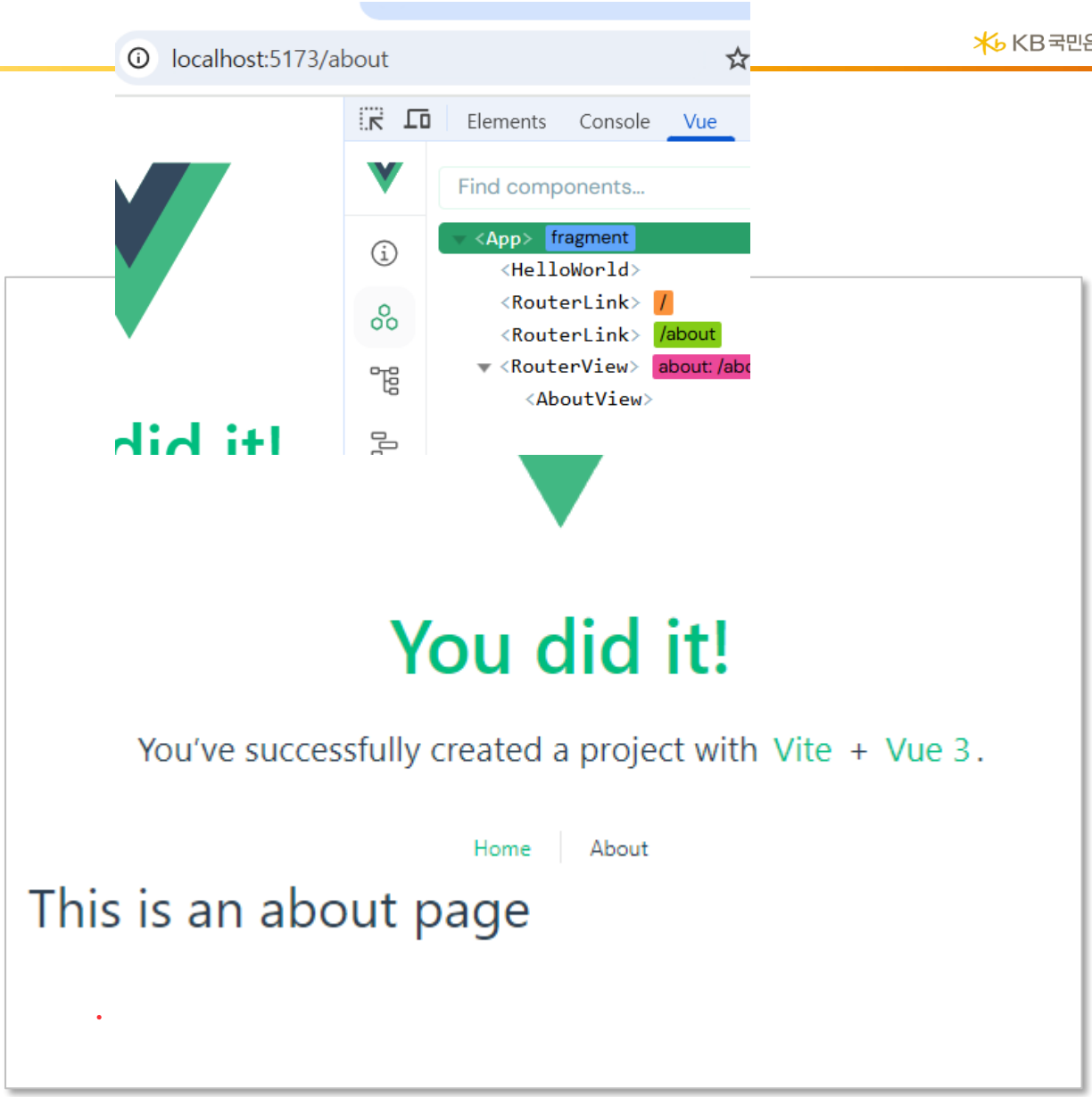
최종 파일에서는 여러 속성이나 링크가 붙은 a태그로 변환된다.
절대로 그냥 a태그를 사용하면 안된다

:to가 아니기에
값은 그냥
문자열

2 vue-router의 기본 사용법

run dev 하여 브라우저에 올려보자.
디버깅 뷰 탭으로 구성이나 url이 바뀌는 것을 관찰

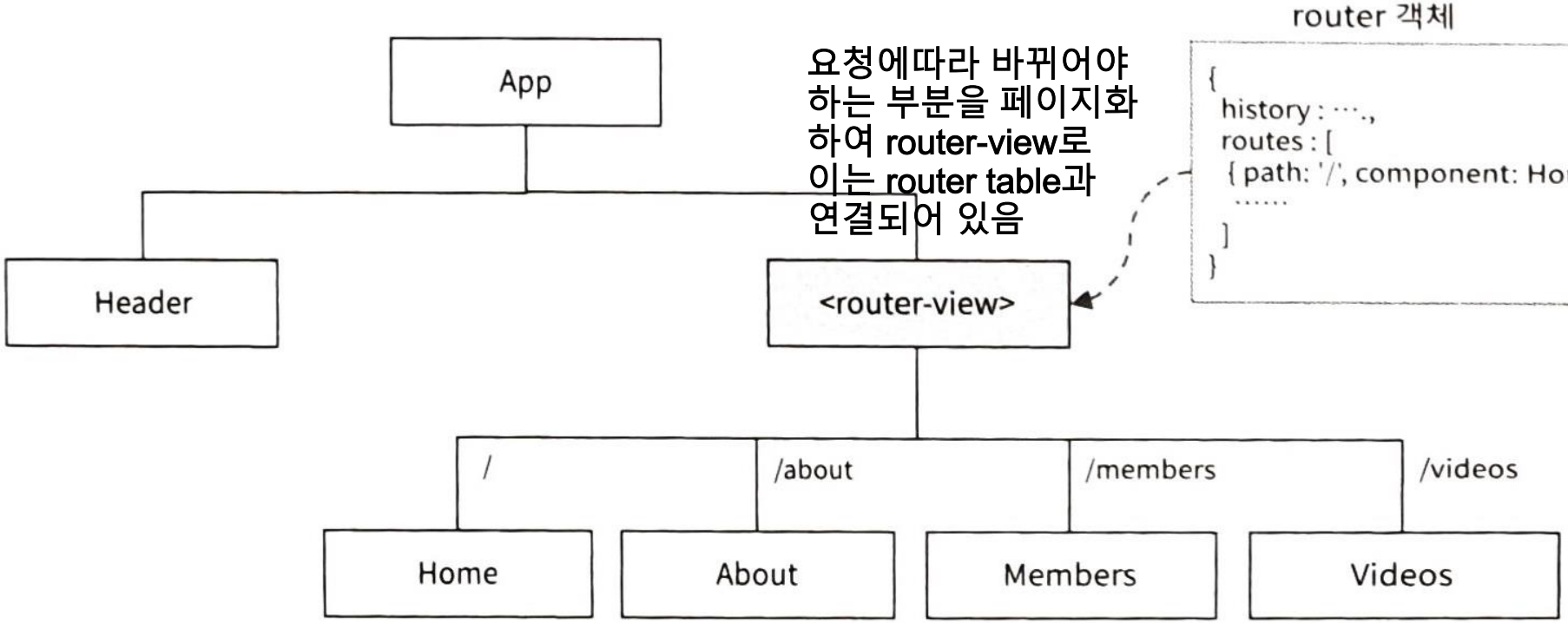




npm run build 명령어로 빌드 해보자. 페이지 별로 js가 생기는 것을 확인할 수 있다. 파일 분리된 것을 볼 수 잇따.

작업할 프로젝트 컴포넌트 구조

컴포넌트나 페이지나 다 컴포넌트.
components폴더 안에 있으면 컴포넌트
pages폴더 안에 잇으면 페이지.
이들은 컴포넌트로 부르는지 라우터로 부르는지 차이가 있다.

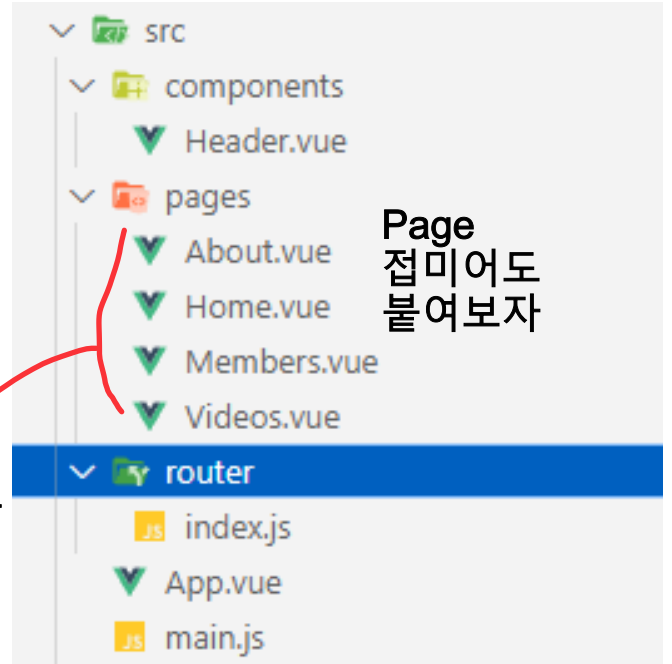


요청에따라 바뀌어야 하는 부분을 페이지화 하여 router-view로 이는 router table과 연결되어 있음

router 객체

```
{
 history : ···,
 routes : [
  { path: '/', component: Home },
  ......
 ]
}
```

pages폴더 아래 있는 것들을 라우터로 부르면 router-view 아래로 배치됨



Page 접미어도 붙여보자

페이지 컴포넌트라고 부르자.

부트스트랩5 설치

npm install bootstrap@5

page 컴포넌트 정의

- src/pages
  - Home.vue
  - About.vue
  - Members.vue
  - Videos.vue

- name과 타이틀을 각 페이지마다 지정

```
<template>
    <div class="card card-body">
        <h2>Home</h2>
    </div>
</template>

<script>
export default {
  name : "Home"
}
</script>
```

src/router/index.js

```
import { createRouter, createWebHistory } from 'vue-router'

import Home from '@/pages/Home.vue'
import About from '@/pages/About.vue'
import Members from '@/pages/Members.vue'
import Videos from '@/pages/Videos.vue'

const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', component: Home },
        { path: '/about', component: About },
        { path: '/members', component: Members },
        { path: '/videos', component: Videos },
    ]
})

export default router;
```

임포트하고 각 페이지컴포넌트들을

해당 path로 접근시 해당 컴포넌트 렌더링

라우팅 테이블에 등록

URL 패쓰와 담당컴포넌트 지정.

src/main.js

```
import { createApp } from 'vue'
import 'bootstrap/dist/css/bootstrap.css'
import App from './App.vue'
import router from './router'

const app = createApp(App)
app.use(router)
app.mount('#app')
```

src/components/Header.vue

```
<template>
  <nav class="navbar navbar-expand-md bg-dark navbar-dark mt-2">
    <span class="navbar-brand">이날치(LeeNalChi)</span>
    <button class="navbar-toggler" type="button" @click="changeIsNavShow">
      <span class="navbar-toggler-icon"></span>
    </button>

    <div :class="navClass">
      <ul class="navbar-nav">
        <li class="nav-item">
          <router-link class="nav-link" to="/">홈</router-link>
        </li>
        <li class="nav-item">
          <router-link class="nav-link" to="/about">소개</router-link>
        </li>
        <li class="nav-item">
          <router-link class="nav-link" to="/members">멤버</router-link>
        </li>
        <li class="nav-item">
          <router-link class="nav-link" to="/videos">영상</router-link>
        </li>
      </ul>
    </div>
  </nav>
</template>
```

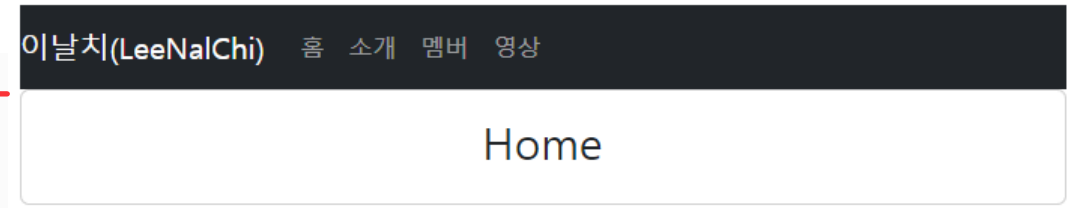
router link는 to경로로 이동. router/index.js에 명시된 to경로와 연결된 페이지 컴포넌트로 이동

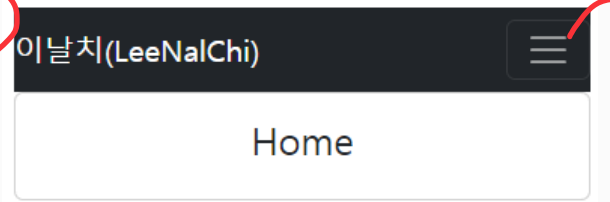
a태그 대신에 라우터링크

src/components/Header.vue

```
<script>
import { reactive, computed } from 'vue';

export default {
  setup() {
    const state = reactive({ isNavShow: false });

    const navClass = computed(() =>
      state.isNavShow
        ? 'collapse navbar-collapse show'
        : 'collapse navbar-collapse'
    );

    const changeIsNavShow = () => { state.isNavShow = !state.isNavShow; };

    return { state, changeIsNavShow, navClass };
  },
};
</script>
```

css media query 에 의해서 크기 전환됨





클릭행위 구현

src/App.vue

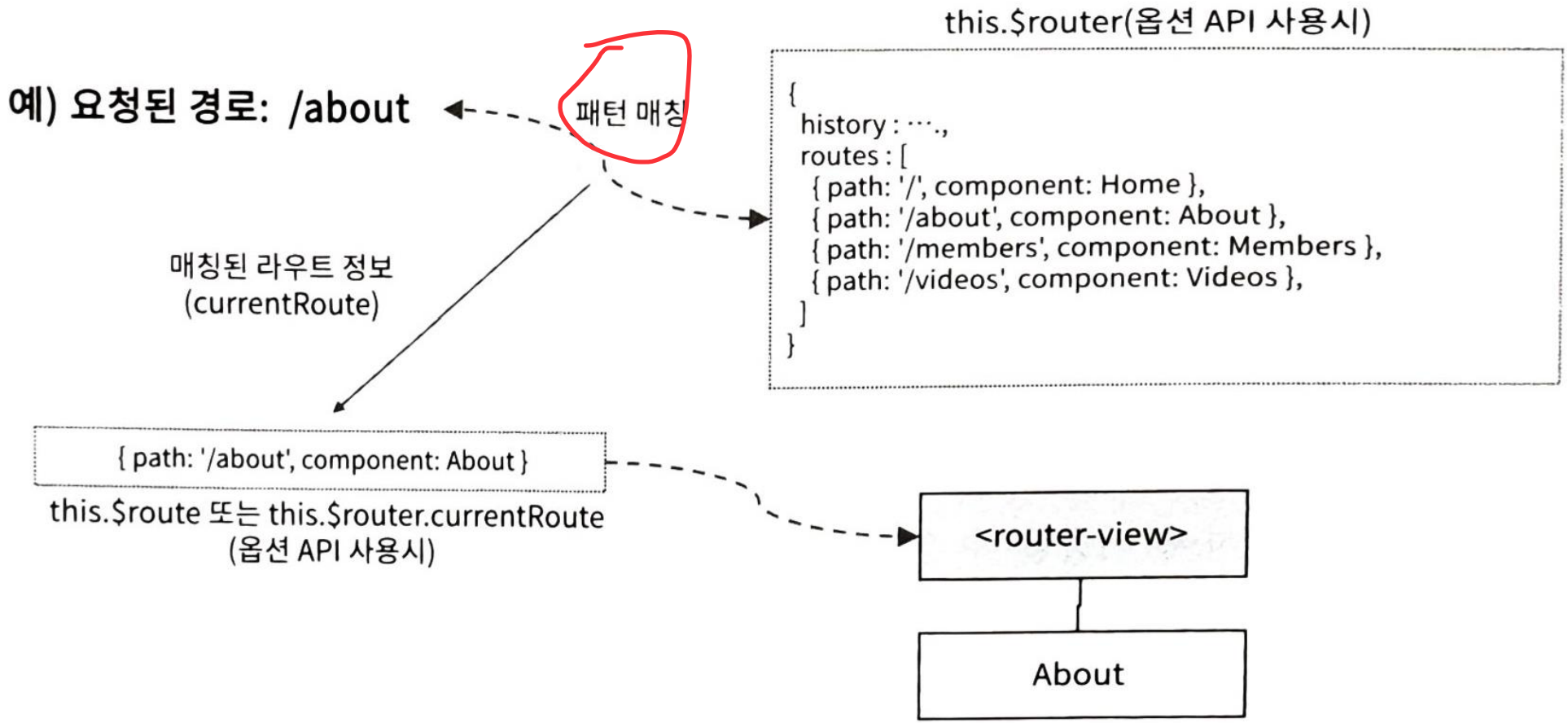
```
<template>
  <div class="container">
    <Header />
    <router-view></router-view>
  </div>
</template>

<script>
import Header from '@/components/Header.vue'

export default {
  name : "App",
  components : { Header },
}
</script>
<style>
.container {
  text-align: center;
  margin-top:10px;
}
</style>
```

routerlink태그를
가지고 있는
컴포넌트

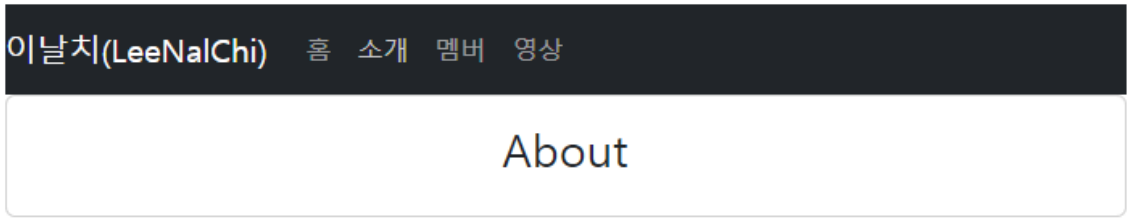
구조 정리

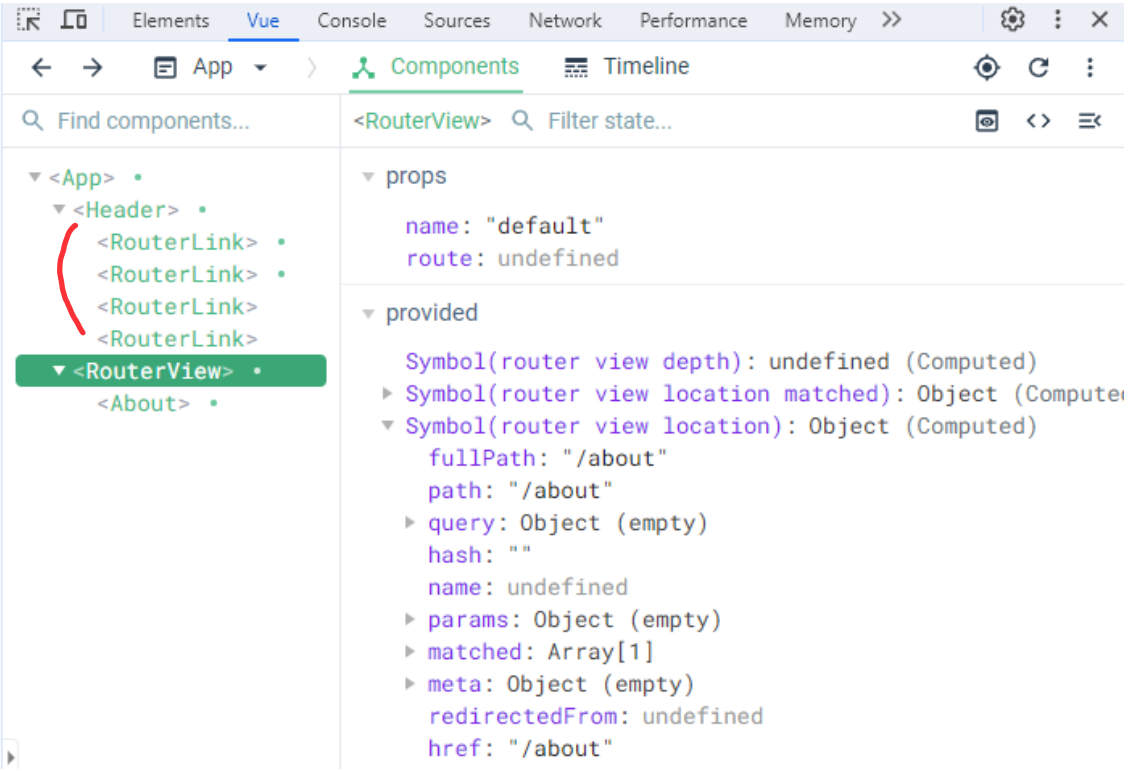
main.js
ㄴApp.vue. router(from router index.js)
| ㄴ routes 라우팅테이블
| ㄴpages/페이지 컴포넌트들
ㄴ Header.vue (Header tag). router-view tag
ㄴ router-link tag

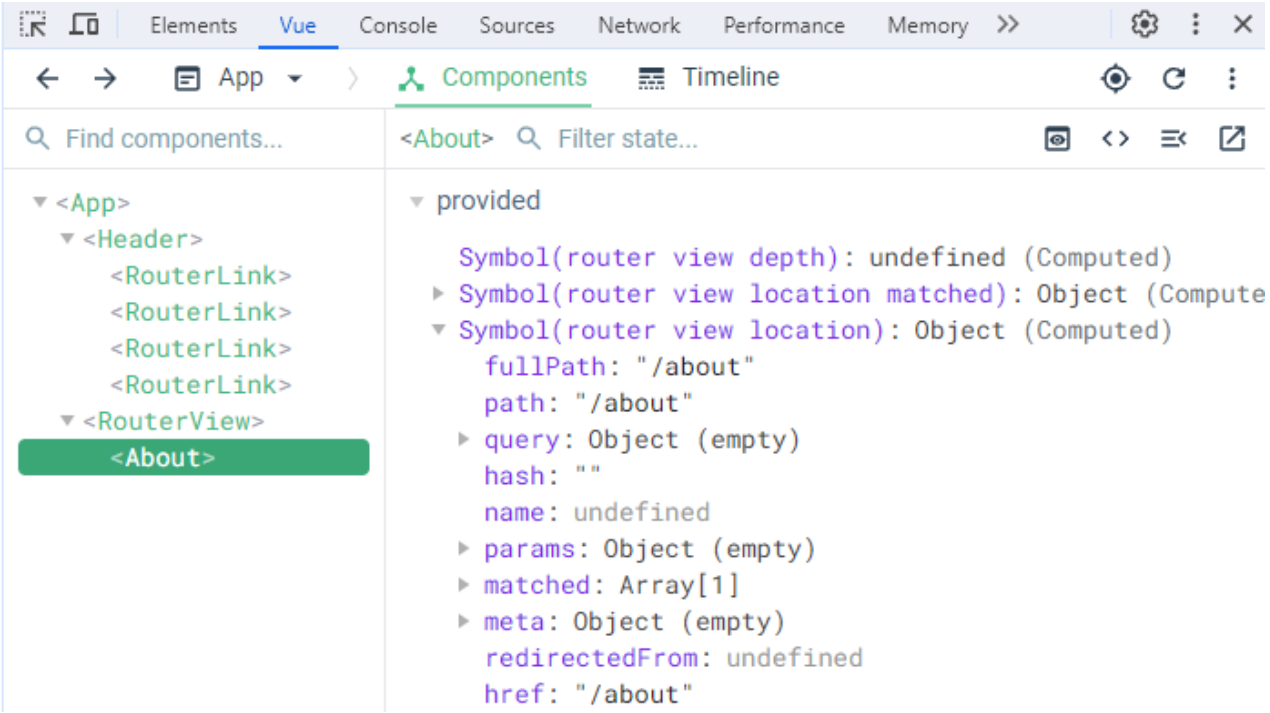
<router-view>의 컴포넌트 마운트 방법



vue-router 기본 적용 방법 예제 실행







컴포넌트에서 router 객체 접근하기

~~~? name= ...& dfasd:...
쿼리 스트링을
얻으려해도 라우팅이 필요함

this.$router

- src/router/index.js에서 작성한 router 객체
- 전체 라우트 정보 파악
- push(), replace(), go() 등의 메서드 이용가능

Router 관련 정보 접근

라우터는 모든 경로를 갖고잇는 라우팅
테이블을 갖고 있다

스크립트에 의해서
페이지 이동 가능해짐

구분	Options API	Composition API
라우터 객체	this.$router	const router = useRouter()
매칭된 라우트 (currentRoute)	this.$route – 또는 – this.$router.currentRoute	const currentRoute = useRoute()

원하는 라우터 객체 반환

현재 라우트 경로만
갖고 잇는 객체

함수명 다르니까
햇갈리지 말자
~~~

컴포넌트에서 router 객체 접근하기

currentRoute 객체의 주요 속성

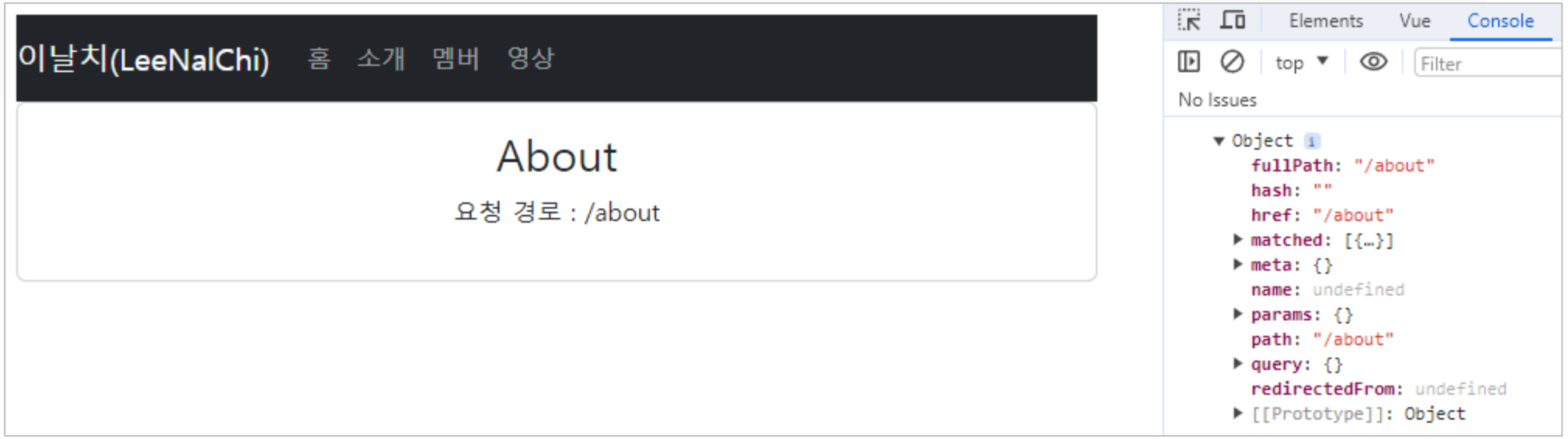
구분	설명
fullPath	전체 요청 경로. 쿼리문자열까지 포함(ex: /about?a=1&b=2)
matched	vue-router 객체의 routes 배열의 라우트 중 매칭된 라우트
params	URI 경로에 동적으로 전달된 파라미터 정보. 자세한 내용은 다음 절(동적 라우트)에서 다룸
path	요청 URI 경로
query	쿼리 문자열 정보 ?a=1&b=2로 요청되었다면 this.$route.query는 { a:1, b:2 }와 같은 객체임
redirectedFrom	다른 경로에서 리디렉트된 경우 리디렉트시킨 URI 경로 정보를 포함

router 객체와 currentRoute 객체

src/pages/About.vue

```
<template>
    <div class="card card-body">
        <h2>About</h2>
        <p>요청 경로 : {{$route.fullPath}}</p>
    </div>
</template>

<script>
export default {
  name : "About",
  created() {
    console.log(this.$route)
  }
}
</script>
```

$를 써서 접근한거 보니
composition api가 아닌
옵션 api네
예날거네



src/pages/Members.vue

```
<template>
    <div class="card card-body">
        <h2>Members</h2>
        <p>요청 경로 : {{currentRoute.fullPath}}</p>
    </div>
</template>

<script>
import { useRoute } from 'vue-router'

export default {
  name : "Members",
  setup() {
    const currentRoute = useRoute();
    return { currentRoute };
  }
}
</script>
```