KB금융그룹 | 국민의 평생 금융파트너

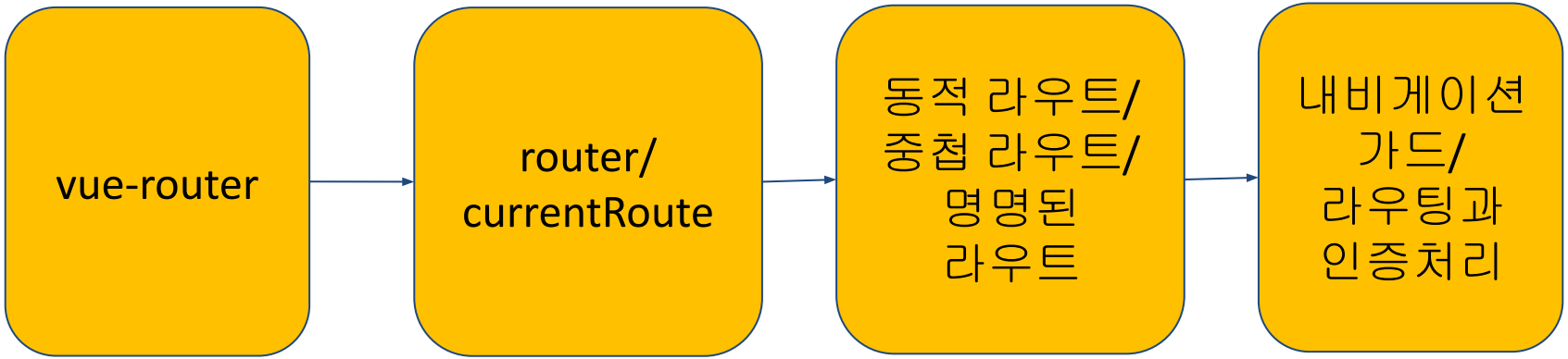
2024년 상반기 K-디지털 트레이닝

Router

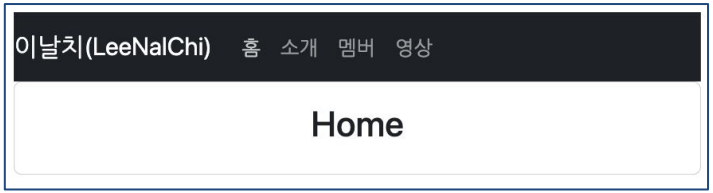
[KB] IT's Your Life

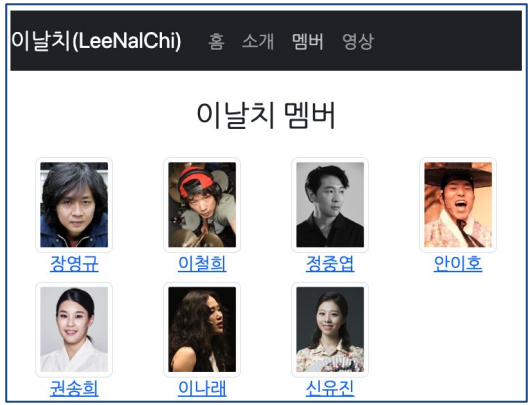
KB 국민은행

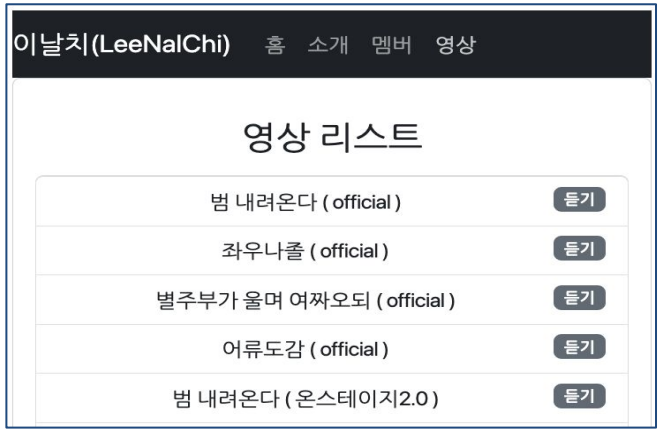
커리큘럼


vue-router
router/
currentRoute
동적 라우트/
중첩 라우트/
명명된
라우트
내비게이션
가드/
라우팅과
인증처리

오늘의 목표(10-vue-router-최종예제.mov)







vue-Router

vue-router

- 클라이언트가 요청한 주소에 따라 서버가 주소를 분석하여 처리할 수 있는 함수로 연결하는 것
- vue에서는 vue-router라는 라우팅 라이브러리를 사용
  - 요청한 주소 처리 함수 매핑 기능
  - 처리 후 결과를 응답하게 연결하는 기능
  - vueJS의 전환 효과 지원
  - 히스토리, 해시 모두 지원
  - 다양한 쿼리 스트링, 파라메터, 와일드 카드를 사용하여 라우팅 구현
- router객체 생성, createRouter()함수 사용
- routers 속성의 값인 배열에 정의되어 있는 요청 주소에 따라 처리할 수 있는 컴포넌트로 연결(랜더링)

vue-router

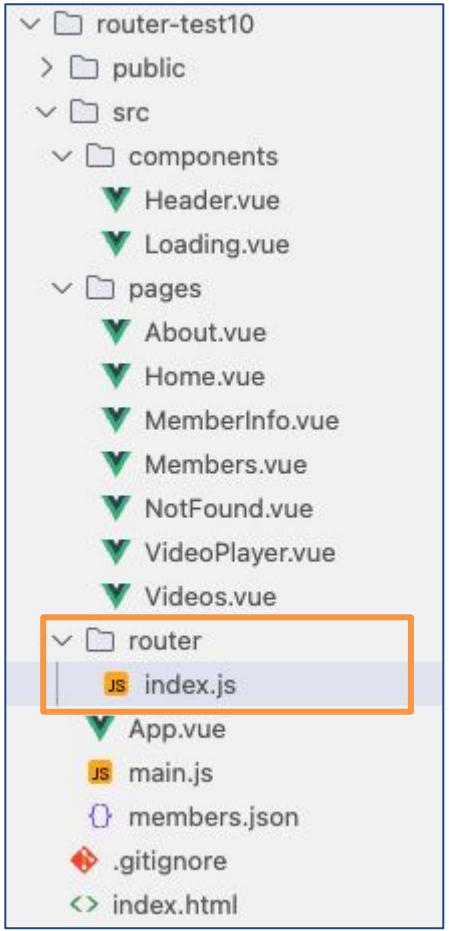
```
import { createRouter, createWebHistory } from 'vue-router'

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;
```

router/index.js

[
{주소, 담당 컴포넌트},
{주소, 담당 컴포넌트},
——----
]

vue-router

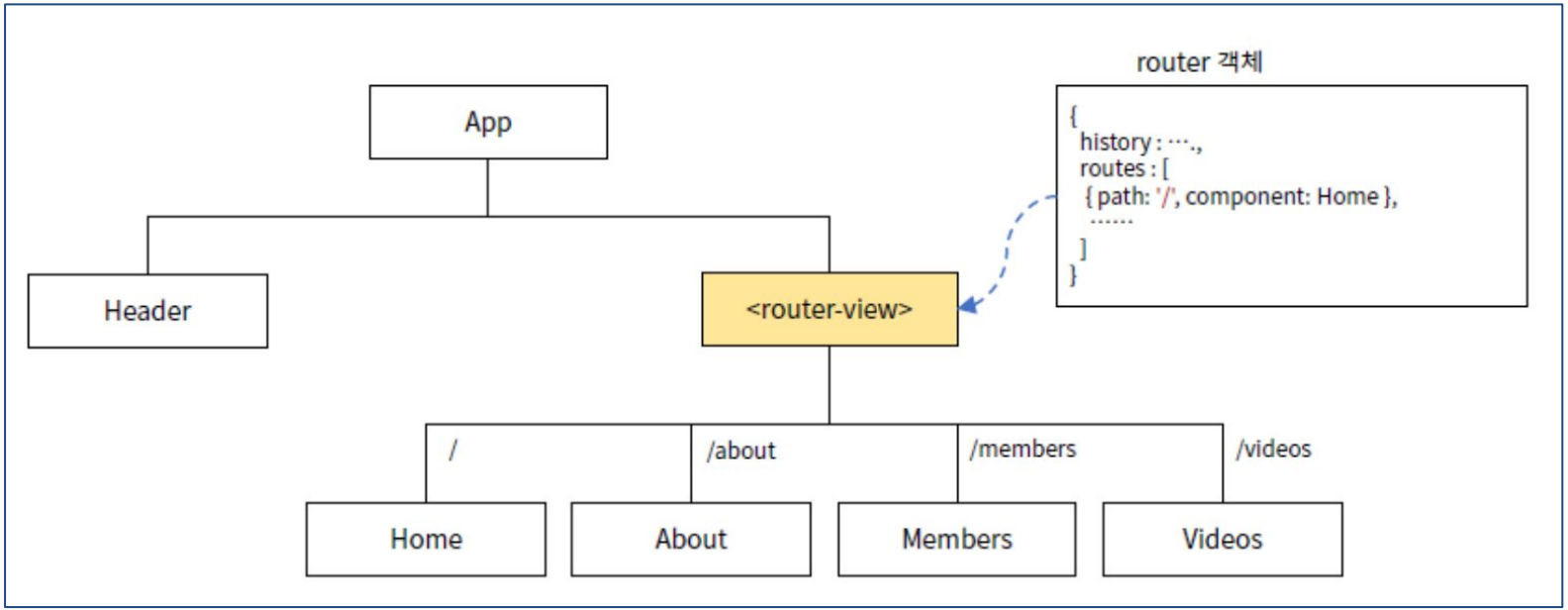
```
administrator@MacBook-Pro myNode % npm init vue router-test10

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
✔ Add Vue DevTools 7 extension for debugging? (experimental) … No / Yes

Scaffolding project in /Users/administrator/Documents/myNode/router-test10...

Done. Now run:

  cd router-test10
  npm install
  npm run dev
```



vue-router

```
administrator@MacBook-Pro router-test10 % npm install
npm WARN deprecated sourcemap-codec@1.4.8: Please use @jridgewell/sourcemap-codec instead

added 150 packages, and audited 151 packages in 6s

45 packages are looking for funding
  run `npm fund` for details

3 vulnerabilities (2 moderate, 1 high)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
administrator@MacBook-Pro router-test10 % npm install vue-router@4 bootstrap@5

changed 4 packages, and audited 151 packages in 6s

45 packages are looking for funding
  run `npm fund` for details

3 vulnerabilities (2 moderate, 1 high)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
```

- **<router-view>기본 객체 위치**



vue-router

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')
```

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>
```

랜더링처리 후
넣을 위치 지정

vue-router

Home.vue

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

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

About.vue

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

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

vue-router

Members.vue

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

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

Videos.vue

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

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

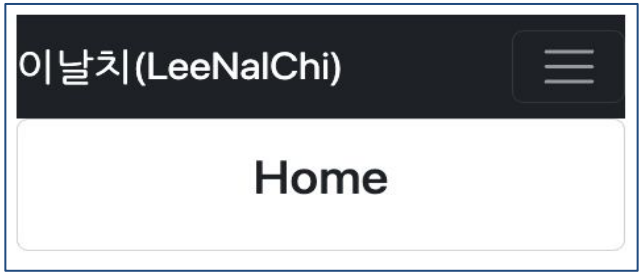
vue-router

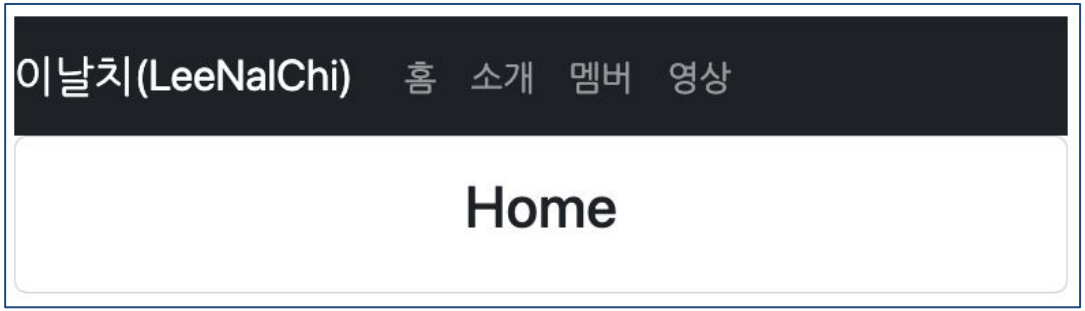
- **<router-link to="주소"> 클릭할 문자</router-link> - <a>태그와 같은 역할**

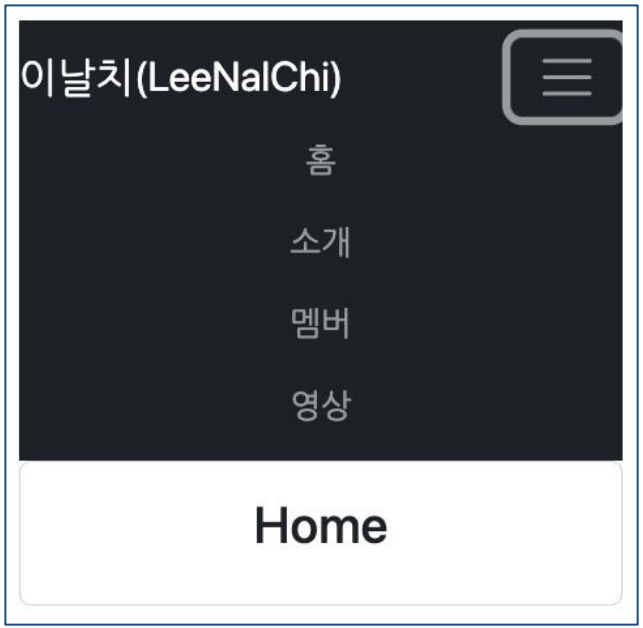
```
<template>
    <nav class="navbar navbar-expand-md bg-dark navbar-dark mt-2">
        <span class="navbar-brand">이날치(LeeNalChi)</span>
        <button class="navbar-toggler" type="button" @click="changeIsNavShow">
            <span class="navbar-toggler-icon"></span>
        </button>

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

Haeder.vue

vue-router


이날치(LeeNalChi)
Home

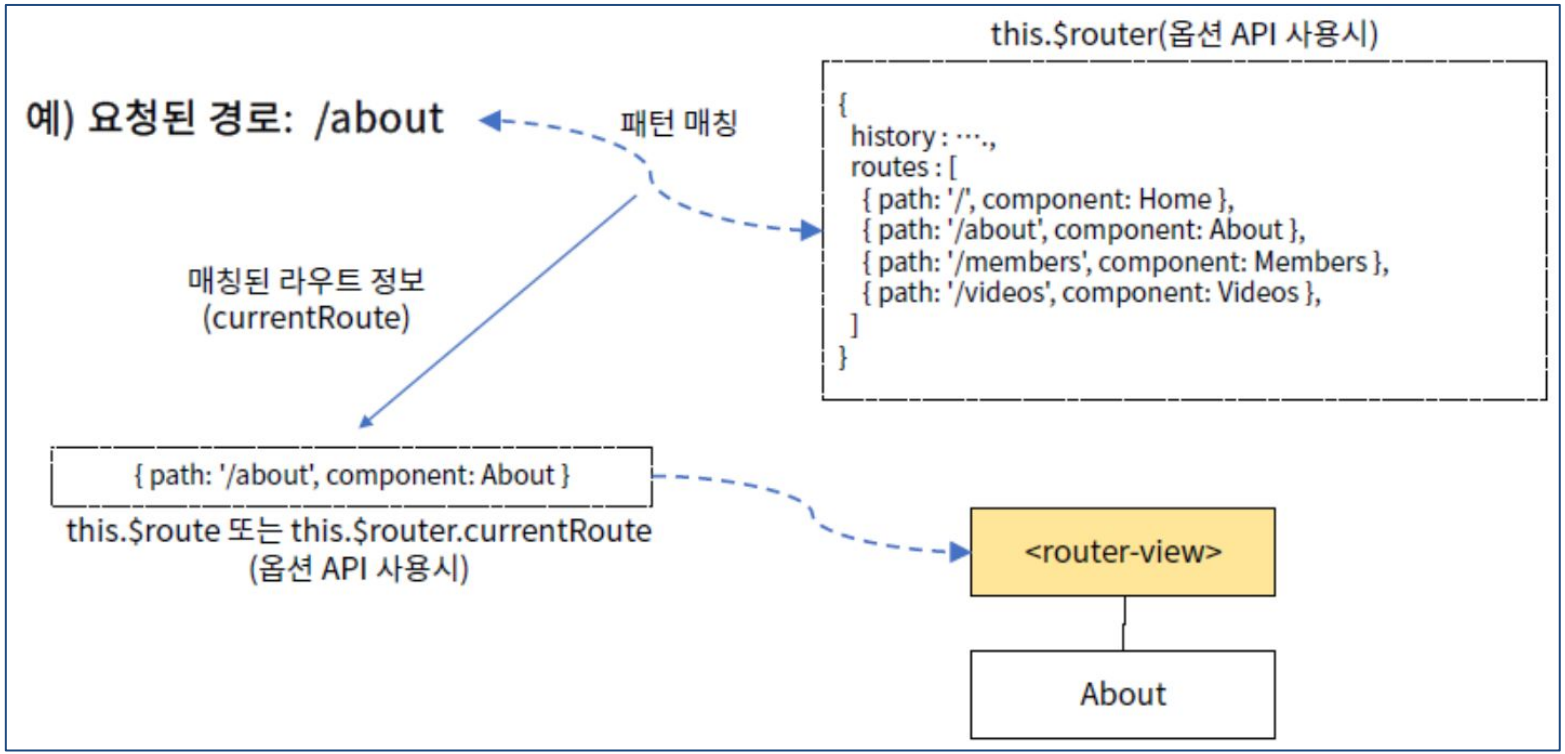

이날치(LeeNalChi)
홈
소개
멤버
영상
Home


이날치(LeeNalChi)
홈
소개
멤버
영상
Home

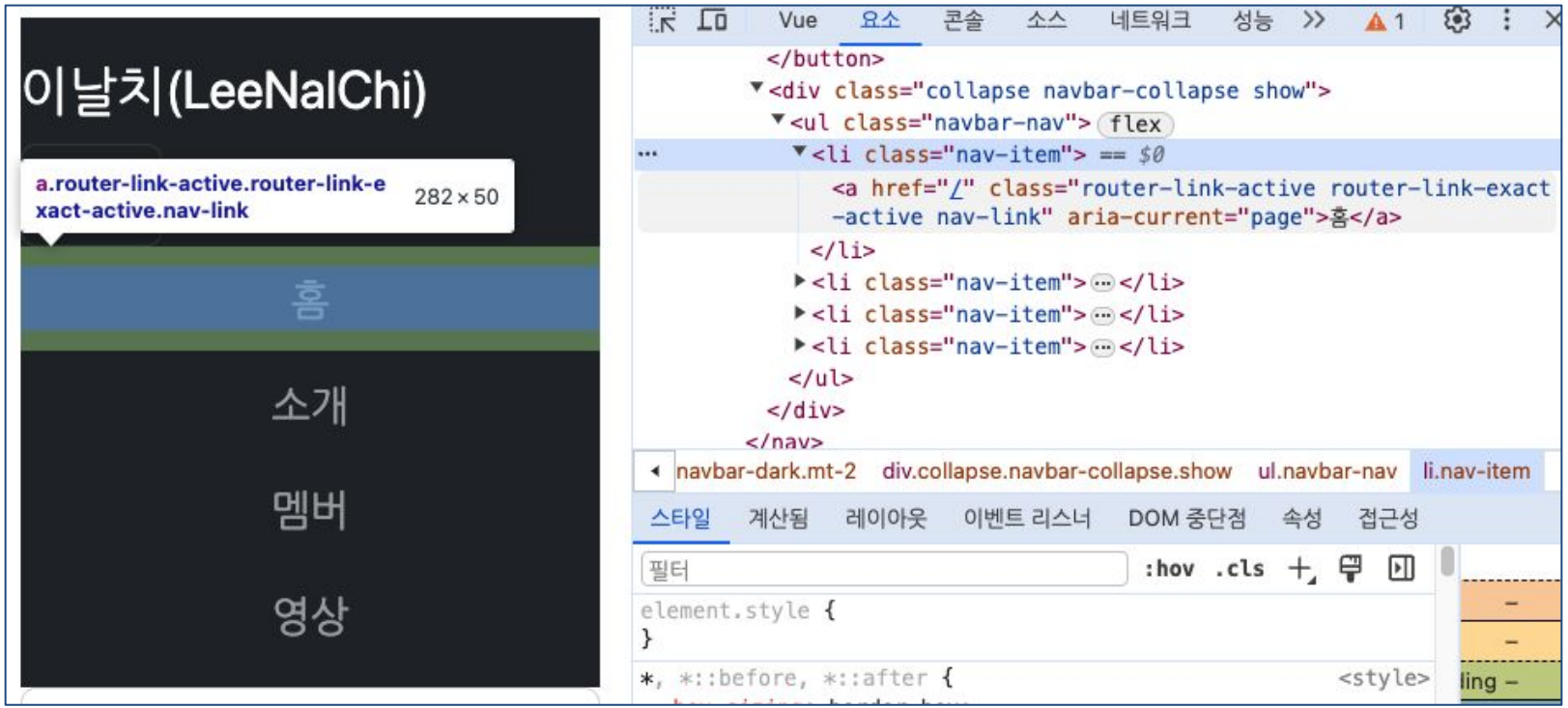

이날치(LeeNalChi)
Home

vue-router

- **App.vue의 <router-view></router-view>**
  - 라우팅되어 요청된 데이터가 들어갈 자리 설정



vue-router


이날치(LeeNalChi)
a.router-link-active.router-link-exact-active.nav-link
282 × 50
홈
소개
멤버
영상
Vue
요소
콘솔
소스
네트워크
성능
1
</button>

<ul class="navbar-nav">
flex
<li class="nav-item"> == $0
홈

<li class="nav-item">…
<li class="nav-item">…
<li class="nav-item">…

</nav>
navbar-dark.mt-2
div.collapse.navbar-collapse.show
ul.navbar-nav
li.nav-item
스타일
계산됨
레이아웃
이벤트 리스너
DOM 중단점
속성
접근성
필터
:hov
.cls
element.style {
}
*, *::before, *::after {
<style>

router
currentRouter

router/currentRouter

- 라우팅된 특정한 페이지의 라우팅과 관련된 데이터를 얻고 싶은 경우
  - this.$router, this.$route, this.$router.currentRoute
  - useRouter(), useRoute()

	Option API	Composition API
라우터 객체	this.$router	const router = useRouter()
CurrentRoute (매칭된 객체)	this.$route this.$router.currentRoute	const currentRoute = useRoute()

- 주요 currentRoute객체의 속성
  - fullPath - 전체 경로
  - matched - routes배열 중 매칭된 라우트
  - params - URI 경로와 함께 전달된 데이터들
  - path - 요청 URI
  - query - URI 경로와 함께 전달된 쿼리 스트링
  - redirectFrom - 리다이렉트된 경우 이전 URI

router/currentRouter

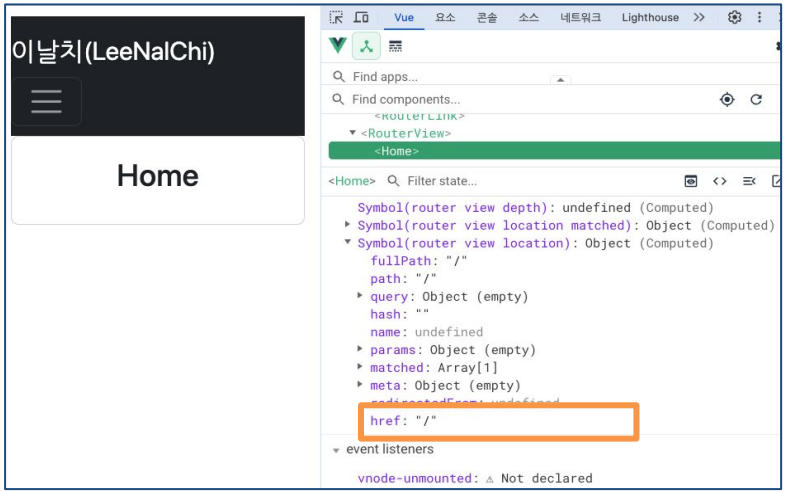
About.vue

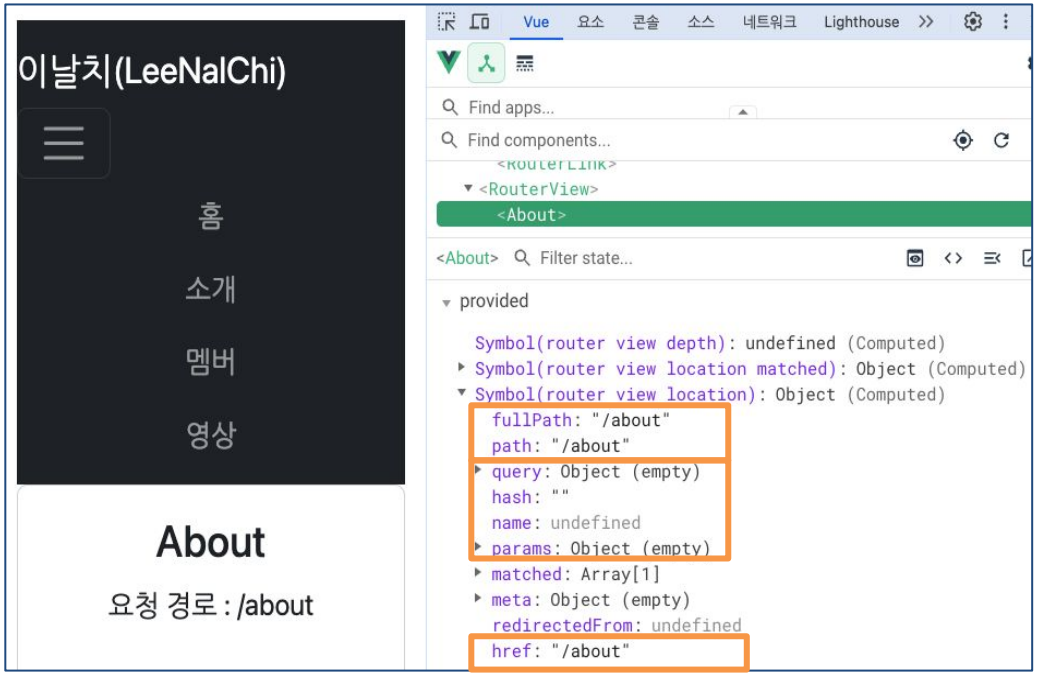
```vue
<template>
    <div class="card card-body">
        <h2>About</h2>
        <p>요청 경로 : {{$route.fullPath}}</p>
    </div>
</template>

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

Members.vue

```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>
```

router/currentRouter





동적 라우트/중첩 라우트/명명된 라우트

동적 라우트

- URI에 패턴을 설정하고 패턴에 따른 경로를 따라 라우팅
- 파라메터 값을 전달 가능(Restful)

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

주소부분에 패턴은
정규표현식으로
- 교재참조

예) 요청된 경로:
/members/6

{ path: '/members/:id', component: MemberInfo }

MemberInfo 컴포넌트

```
const currentRoute = useRoute()
const id = currentRoute.parms.id
```

```
export default {
    name : "MemberInfo",
    setup() {
        const currentRoute = useRoute()
        const id = parseInt(currentRoute.params.id, 10);
        const member = members.find((m)⇒m.id === id)
        return { member }
    }
}
```

동적 라우트

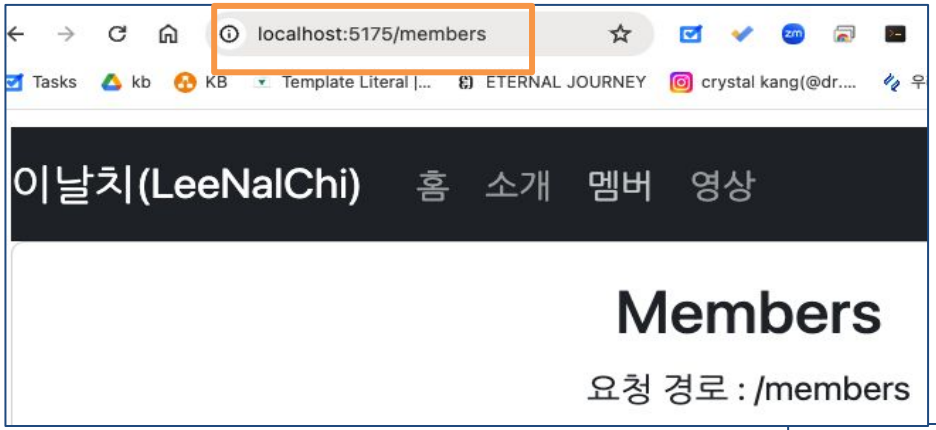
router/index.js

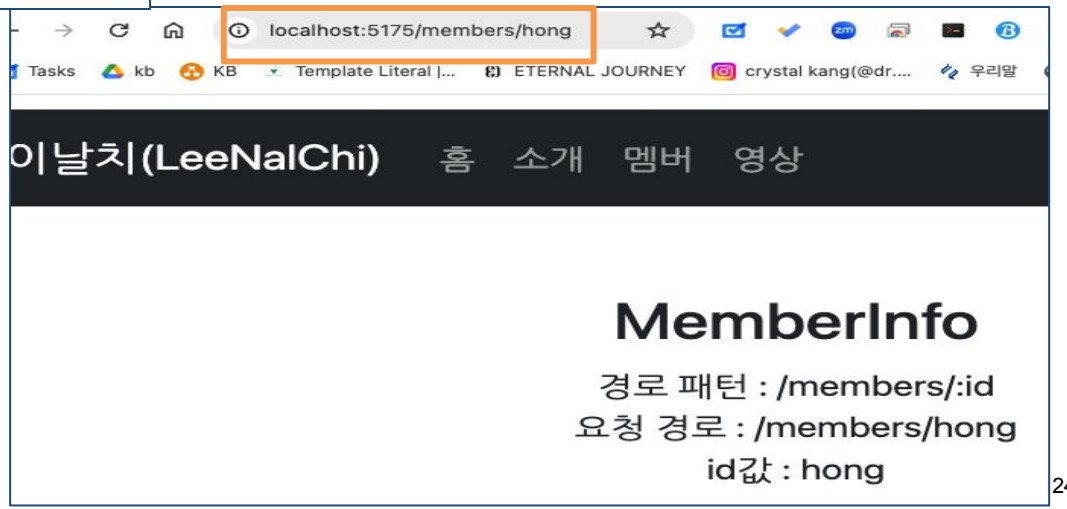
```
import { create                    vue-router'

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

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

export default router;
```

pages/MemberInfo.vue

```
<temp
    <div className="mt-5">
        <h2>MemberInfo</h2>
        <div>
            경로 패턴 : /members/:id<br>
            요청 경로 : {{ currentRoute.path }}<br>
            id값 : {{ currentRoute.params.id }}<br>

        </div>
    </div>
</template>

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

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

동적 라우트


localhost:5175/members
Tasks
kb
KB
Template Literal |...
ETERNAL JOURNEY
crystal kang(@dr....
이날치(LeeNalChi)
홈
소개
멤버
영상
Members
요청 경로 : /members


localhost:5175/members/hong
Tasks
kb
KB
Template Literal |...
ETERNAL JOURNEY
crystal kang(@dr....
우리말
이날치(LeeNalChi)
홈
소개
멤버
영상
MemberInfo
경로 패턴 : /members/:id
요청 경로 : /members/hong
id값 : hong

동적 라우트

- 데이터는 **json**으로 준비**(member.json)**

member.json

동적 라우트

- member.json을 import
- member별 페이지 라우팅해줄 수 있도록 <router-link>변경

```
<template>
    <div>
      <h2 class="m-4">이날치 멤버</h2>
      <div class="container">
        <div class="row">
          <div v-for="m in members" :key="m.id"
            class="col-6 col-xs-6 col-sm-4 col-md-3 col-lg-2">
            <router-link :to="'/members/'+m.id">
              <img class="img-thumbnail"
                style="width:90px; height:110px;"
                :src="m.photo" :title="m.name" /><br/>
              <h6 class="display-7">{{m.name}}</h6>
            </router-link>
          </div>
        </div>
      </div>
    </div>
</template>
```

```
<script>
import members from '@/members.json'

export default {
    name : "Members",
    setup() {
        return { members };
    }
}
</script>
```

동적 라우트

pages/MemberInfo.vue

```vue
<template>
    <div className="mt-5">
      <img :src="member.photo" class="img" />
      <h4 class="mt-2">{{member.name}}({{member.role}})</h4>
      <p>{{member.desc}} </p>
      <a v-if="member.insta && member.insta !== ''"
         class="fa fa-instagram m-1" :href="member.insta"></a>
      <a v-if="member.facebook && member.facebook !== ''"
         class="fa fa-facebook m-1" :href="member.facebook"></a>
      <a v-if="member.youtube && member.youtube !== ''"
         class="fa fa-youtube m-1" :href="member.youtube"></a>
      <br /><br />
      <router-link to="/members">멤버 목록으로</router-link>
    </div>
</template>
```

```vue
<script>
import { useRoute } from 'vue-router'
import members from '@/members.json'

export default {
    name : "MemberInfo",
    setup() {
        const currentRoute = useRoute()
        const id = parseInt(currentRoute.params.id, 10);
        const member = members.find((m)=>m.id === id)
        return { member }
    }
}
</script>
```

동적 라우트

이날치(LeeNalChi) 홈 소개 멤버 영상

Home

이날치(LeeNalChi) 홈 소개 멤버 영상

Videos

이날치(LeeNalChi) 홈 소개 멤버 영상

이날치 멤버


장영규


이철희


정중엽


안이호

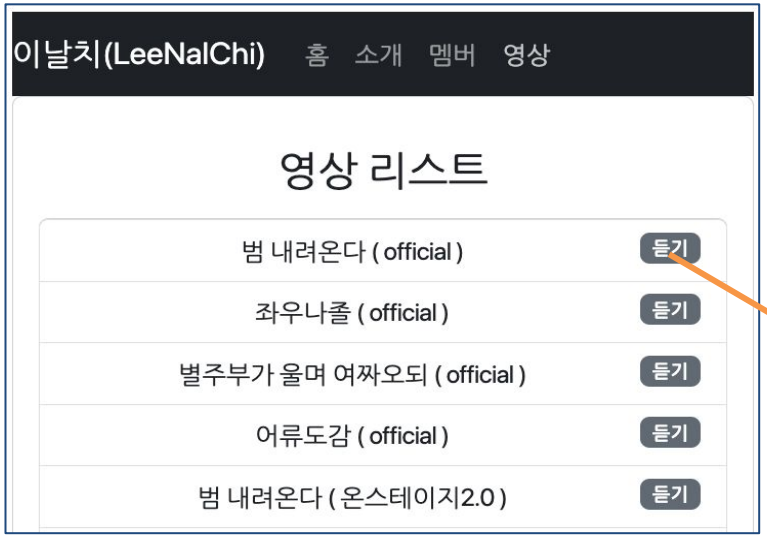

권송희

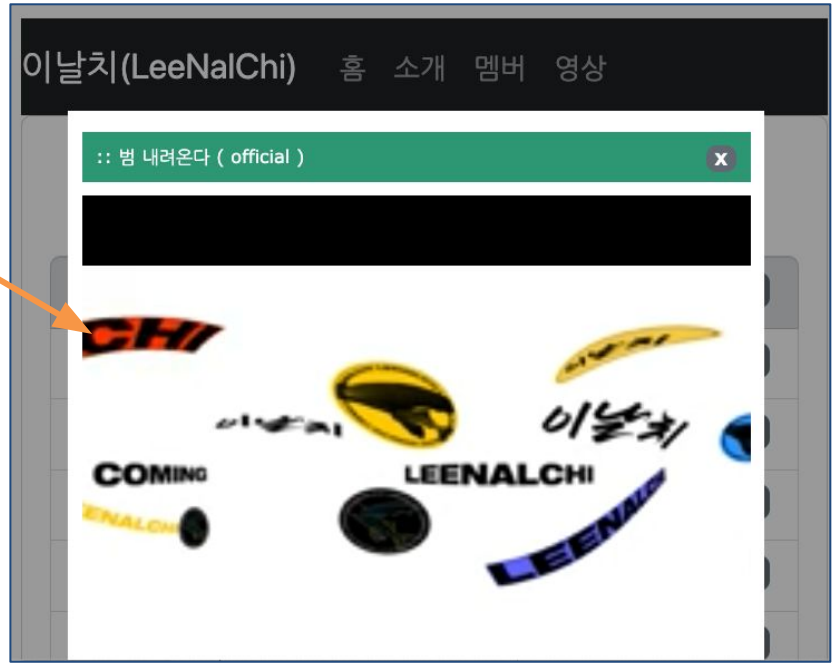

이나래


신유진

중첩 라우트

- 화면을 중첩해야 하는 경우 사용



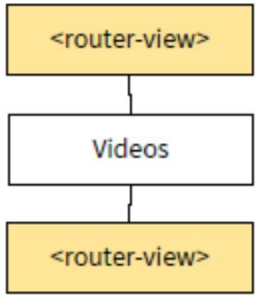


중첩 라우트

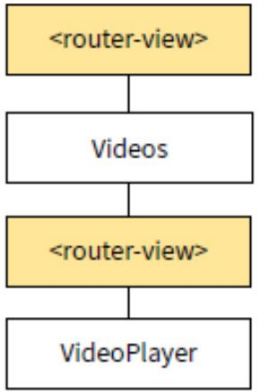
```
const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', component: Home },
        { path: '/about', component: About },
        { path: '/members', component: Members },
        { path: '/members/:id(\\d+)', component: MemberInfo },
        {
            path: '/videos', component: Videos,
            children : [
                { path:":id", component: VideoPlayer }
            ]
        },
    ]
})
```

중첩되게
router-view객체를
정의

요청 경로 : /videos



요청 경로 : /videos/:id



중첩 라우트

router/index.js

```
import { createRouter, creat
import Home from '@/pages/Home.vue'
import About from '@/pages/About.vue'
import Members from '@/pages/Members.vue'
import Videos from '@/pages/Videos.vue'
import MemberInfo from '@/pages/MemberInfo.vue'
import VideoPlayer from '@/pages/VideoPlayer.vue'

const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', component: Home },
        { path: '/about', component: About },
        { path: '/members', component: Members },
        { path: '/members/:id(\\d+)', component: MemberInfo },
        {
            path: '/videos', component: Videos,
            children : [
                { path:":id", component: VideoPlayer }
            ]
        },
    ]
})

export default router;
```

App.vue

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

<script>
import Header from '@/components/Header.vue'
import { provide } from 'vue';

export default {
  name : "App",
  components : { Header
  setup() {
    provide('videos', [
      { "id":"t0BHhqw_Ecc", "title":"범 내려온다", "category":"official" },
      { "id":"FrCkLMxnlMI", "title":"좌우나졸", "category":"official" },
      { "id":"7O0hIrgMcCg", "title":"별주부가 울며 어짜오되", "category":"official" },
      { "id":"MJD_fAdqNQc", "title":"어류도감", "category":"official" },
      { "id":"SmTRaSg2fTQ", "title":"범 내려온다", "category":"온스테이지2.0" },
      { "id":"B_X7n0AaLqA", "title":"범 내려온다(서울)", "category":"관광공사" },
      { "id":"sV1jq6RFSXc", "title":"어류도감(부산)", "category":"관광공사" },
      { "id":"dInPs_VHqSM", "title":"좌우나졸(전주)", "category":"관광공사" }
    ])
  }
}
</script>
```

provide로 설정해두면 inject하여 원하는 곳에서 사용 가능

중첩 라우트

Videos.vue

```
<template>
    <div class="card card-body">
        <h2 class="m-3">영상 리스트</h2>
        <ul class="list-group">
          <li v-for="v in videos" :key="v.id" class="list-group-item text-left"
            :class="playingVideo(v.id)">
            {{v.title}} ( {{v.category}} )
            <router-link :to="'/videos/'+v.id">
              <span class="float-end badge bg-secondary">듣기</span>
            </router-link>
          </li>
        </ul>
        <router-view></router-view>
    </div>
</template>
```

```
export default {
  name : "Videos",
  setup() {
    const videos = inject('videos')
    const currentRoute = useRoute();
    const playingVideo = (id) => {
      return id === currentRoute.params.id ? "list-group-item-secondary" : "";
    }
    return { playingVideo, videos }
  }

}
</script>
```

```
npm install --save youtube-vue3
```

중첩 라우트

- **구현 기능**
  - 듣기를 누르면 중첩된 모달 화면에 영상이 플레이
    - 모달화면 띄우기
    - 영상 플레이
  - 플레이 중 다음 영상 누르면 다음 영상 플레이
  - 해당 영상이 끝나면 다음 영상 플레이
  - 플레이 영상 중지/닫기 버튼 클릭하면 플레이어 닫기

중첩 라우트

VideoPlayer.vue

```
<template>
    <div class="modal">
        <div class="box">
            <div class="heading">
                <span class="title">:: {{videoInfo.video.title}}</span>
                <span class="category"> ( {{videoInfo.video.category}} ) </span>
                <span class="float-end badge bg-secondary pointer" @click="stopVideo">
                    X
                </span>
            </div>
            <div class="player">
                <YoutubeVue3 ref="playerRef" :videoid="videoInfo.video.id"
                    :autoplay="1" :controls="1" @ended="playNext" />
            </div>
            <div>
                <div>
                    <i class="fa fa-backward ml-2 pointer" @click="playPrev"></i>
                    <i class="fa fa-stop ml-2 pointer" @click="stopVideo"></i>
                    <i class="fa fa-forward ml-2 pointer" @click="playNext"></i>
                </div>
            </div>
        </div>
    </div>
</template>
```

중첩 라우트

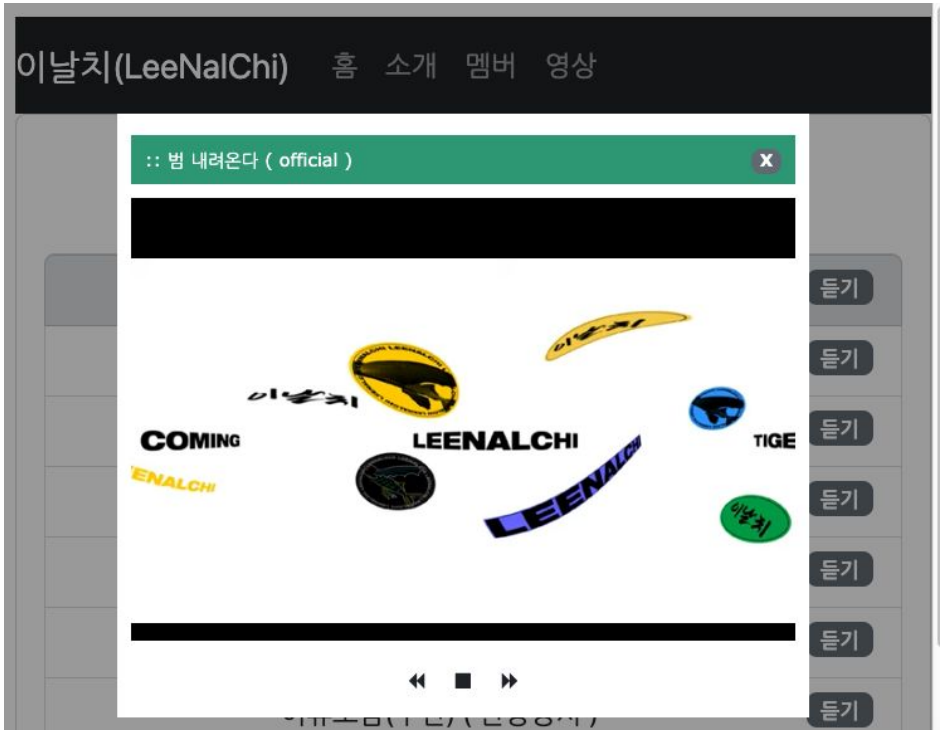
VideoPlayer.vue

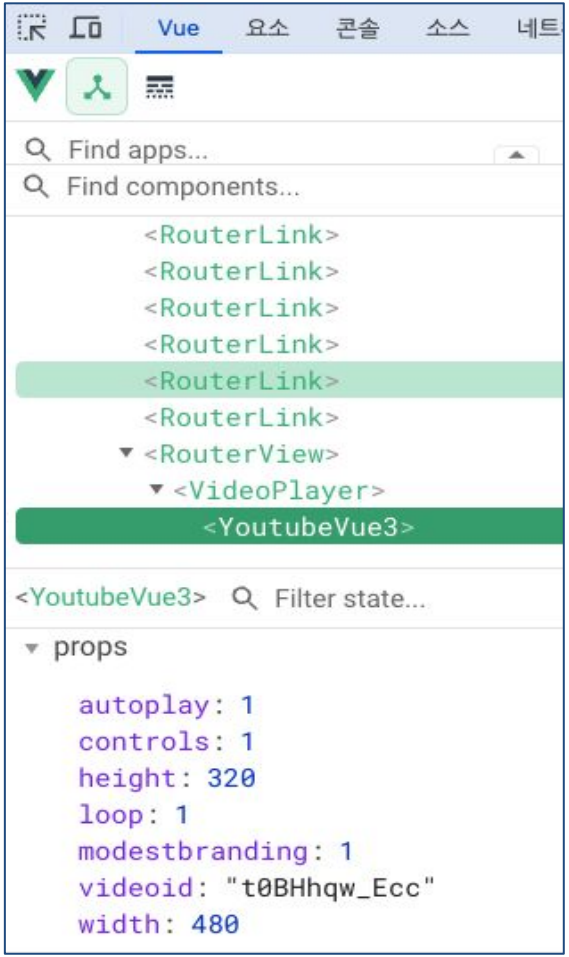
```vue
<script>
import { reactive, ref, inject } from 'vue';
import { useRoute, useRouter } from 'vue-router';
import { YoutubeVue3} from 'youtube-vue3';

export default {
    name : "VideoPlayer",
    components : { YoutubeVue3 },
    setup() {
        const videos = inject('videos');
        const playerRef = ref(null);
        const currentRoute = useRoute();
        const router = useRouter();

        let videoInfo =
            reactive({ video: videos.find((v)=>v.id === currentRoute.params.id) });
        const stopVideo = () => {
            playerRef.value.player.stopVideo();
            router.push('/videos');
        }
        const playNext = () => {
            const index = videos.findIndex((v)=>v.id === videoInfo.video.id);
            const nextVideo = videos[index+1];
            if (nextVideo) {
                videoInfo.video = nextVideo;
                router.push('/videos/' + nextVideo.id);
            } else {
                videoInfo.video = videos[0];
                router.push('/videos/' + videos[0].id);
            }
        }
```

라우팅 실행
<a>태그 클릭

```vue
        const playPrev = () => {
            const index = videos.findIndex((v)=>v.id === videoInfo.video.id);
            const prevVideo = videos[index-1];
            if (prevVideo) {
                videoInfo.video = prevVideo;
                router.push('/videos/' + prevVideo.id);
            }
        }

        return { videoInfo, playerRef, playNext, stopVideo,  playPrev };
    }
}
</script>

<style scoped>
.modal { display: block; position: fixed; z-index: 1;
    left: 0; top: 0; width: 100%; height: 100%;
    overflow: auto;
    background-color: rgba(0,0,0,0.4); }
.box { background-color: white; margin:80px auto;
    max-width: 500px; min-width: 100px; min-height: 350px;
    font: 13px "verdana";
    padding: 10px 10px 10px 10px;  }
.box div { padding: 0;  display: block;  margin: 10px 0 0 0; }
.box .heading { background: #33A17F; font-weight: 300;
    text-align: left; color: #fff;
```

중첩 라우트


이날치(LeeNalChi)
홈
소개
멤버
영상
:: 범 내려온다 (official)
COMING
LEENALCHI
TIGE
듣기


Vue
요소
콘솔
소스
Find apps...
Find components...
<RouterLink>
<RouterLink>
<RouterLink>
<RouterLink>
<RouterLink>
<RouterLink>
<RouterView>
<VideoPlayer>
<YoutubeVue3>
<YoutubeVue3>
Filter state...
props
autoplay: 1
controls: 1
height: 320
loop: 1
modestbranding: 1
videoid: "t0BHhqw_Ecc"
width: 480

명명된 라우트

- 라우팅 정보를 이름으로 간단하게 설정하여 하이퍼 링크 설정시 용이

router/index.js

```
const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', name:'home', component: Home },
        { path: '/about', name:'about', component: About },
        { path: '/members', name:'members', component: Members },
        { path: '/members/:id', name:'members/id', component: MemberInfo },
        {
            path: '/songs', name:'videos', component: Videos,
            children : [
                { path: ':id', name:'videos/id', component: VideoPlayer }
            ]
        },
    ]
})
```

명명된 라우트

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="{ name:'home' }">홈</router-link>
                </li>
                <li class="nav-item">
                    <router-link class="nav-link" :to="{ name:'about' }">소개</router-link>
                </li>
                <li class="nav-item">
                    <router-link class="nav-link" :to="{ name:'members' }">멤버</router-link>
                </li>
                <li class="nav-item">
                    <router-link class="nav-link" :to="{ name:'videos' }">영상</router-link>
                </li>
            </ul>
        </div>
    </nav>
</template>
```

명명된 라우트

Members.vue

```
<div class="row">
  <div v-for="m in members" :key="m.id"
    class="col-6 col-xs-6 col-sm-4 col-md-3 col-lg-2">
    <router-link :to="{ name:'members/id', params: { id: m.id } }">
      <img class="img-thumbnail"
        style="width:90px; height:110px;"
        :src="m.photo" :title="m.name" /><br/>
      <h6 class="display-7">{{m.name}}</h6>
    </router-link>
  </div>
</div>
```

/member/:id →
<router-link :to = "{ name:member/id', params : {id : 1} }" >

/member?a=1&b=2 →
<router-link :to = "{ name: 'members', query: {a:1, b:2} }" >

MemberInfo.vue

```
<template>
    <div className="mt-5">
      <img :src="member.photo" class="img" />
      <h4 class="mt-2">{{member.name}}({{member.role}})</h4>
      <p>{{member.desc}} </p>
      <a v-if="member.insta && member.insta =/= ''"
          class="fa fa-instagram m-1" :href="member.insta"></a>
      <a v-if="member.facebook && member.facebook =/= ''"
          class="fa fa-facebook m-1" :href="member.facebook"></a>
      <a v-if="member.youtube && member.youtube =/= ''"
          class="fa fa-youtube m-1" :href="member.youtube"></a>
      <br /><br />
      <router-link :to="{ name:'members' }">멤버 목록으로</router-link>
    </div>
</template>
```

명명된 라우트

Videos.vue

```
<template>
    <div class="card card-body">
        <h2 class="m-3">영상 리스트</h2>
        <ul class="list-group">
          <li v-for="v in videos" :key="v.id" class="list-group-item text-left"
            :class="playingVideo(v.id)">
            {{v.title}} ( {{v.category}} )
            <router-link :to="{ name:'videos/id', params: { id: v.id } }">
              <span class="float-end badge bg-secondary">듣기</span>
            </router-link>
          </li>
        </ul>
        <router-view></router-view>
    </div>
</template>
```

명명된 라우트

VideoPlayer.vue

```
<script>
export default {
    setup() {

        let videoInfo =
            reactive({ video: videos.find((v)⇒v.id === currentRoute.params.id) });
        const stopVideo = () ⇒ {
            playerRef.value.player.stopVideo();
            router.push({ name:'videos' });
        }
        const playNext = () ⇒ {
            const index = videos.findIndex((v)⇒v.id === videoInfo.video.id);
            const nextVideo = videos[index+1];
            if (nextVideo) {
                videoInfo.video = nextVideo;
                router.push({ name:'videos/id', params: { id: nextVideo.id } });
            } else {
                videoInfo.video = videos[0];
                router.push({ name:'videos/id', params: { id: videos[0].id } });
            }
        }
```

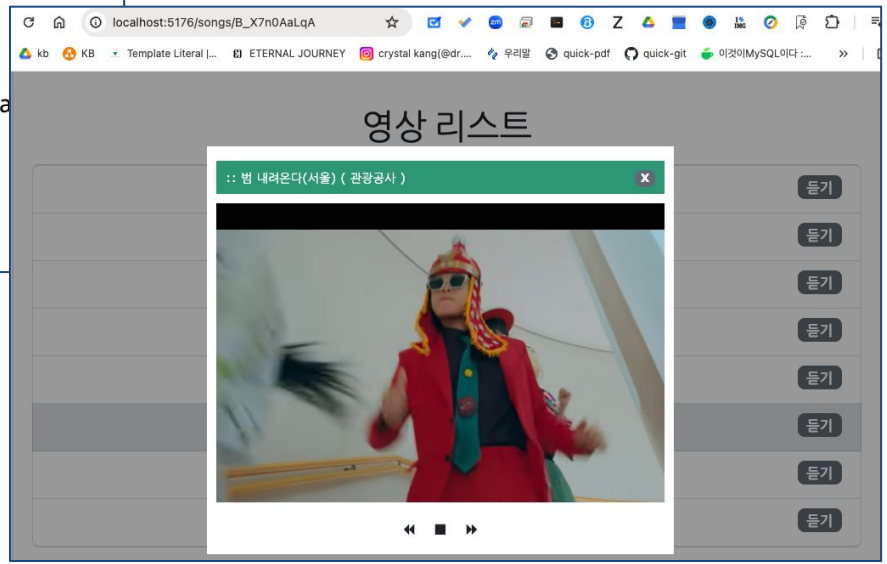
명명된 라우트

VideoPlayer.vue

```
        const playPrev = () => {
            const index = videos.findIndex((v)=>v.id === videoInfo.video.id);
            const prevVideo = videos[index-1];
            if (prevVideo) {
                videoInfo.video = prevVideo;
                router.push({ name:'videos/id', params: { id: prevVideo.id } });
            }
        }

        return { videoInfo, playerRef, playNext, stopVideo,  playPrev };
    }
}
</script>
```

명명된 라우트

```
const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', name:'home', component: Home },
        { path: '/about', name:'about', component: About },
        { path: '/members', name:'members', component: Members },
        { path: '/members/:id', name:'members/id', component: MemberInfo },
        {
            path: '/songs', name:'videos', component: Videos,
            children : [
                { path: ':id', name:'videos/id', component: VideoPla
            ]
        },
    ]
})
```



명명된 뷰

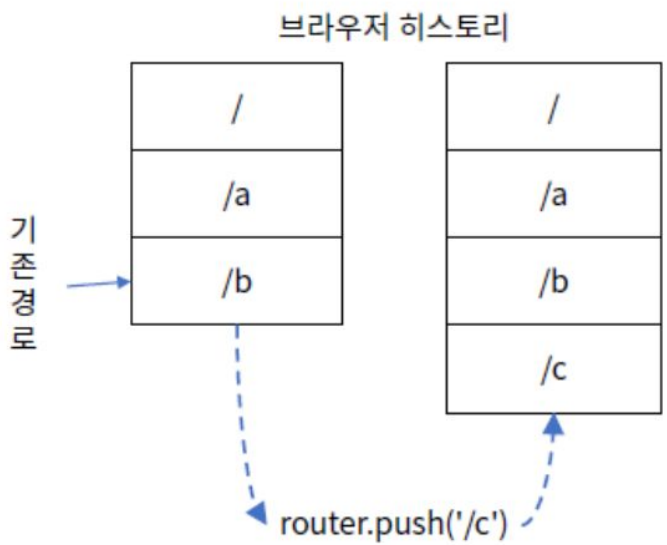
```
const router = createRouter({
    history: createWebHistory(),
    routes : [
        {
            path: '/',
            components: {
                default:Home,
                left:HomeLeft,
            }
        },
        {
            path: '/members',
            components: {
                default: Members,
                left: MembersLeft,
                footer: MembersFooter,
            }
        },
    ]
})
```

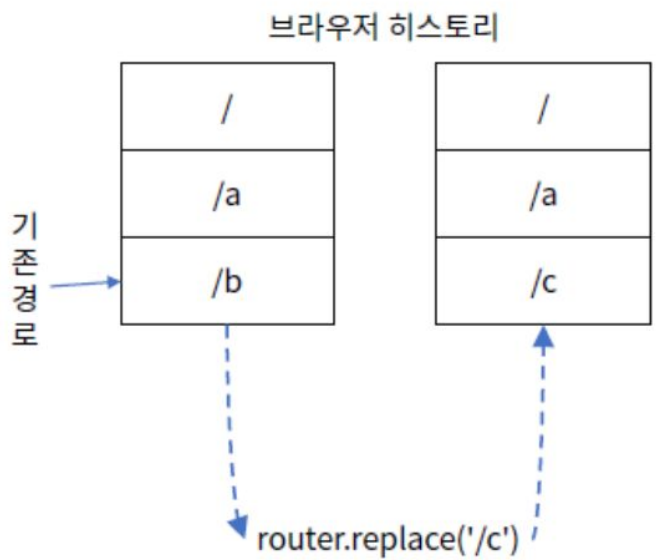
Header	
left	default
footer	

라우터 객체의 주요 메서드

메서드	설명
addRoute(parentParent, route)	실행시에 동적으로 부모 라우트에 새로운 라우트 추가
removeRoute(name)	실행시에 동적으로 라우트 정보 삭제
go(n)	history.go(n)을 실행, go(-1)는 뒤로
back()	go(-1)
forward()	go(1)
push(to)	지정된 경로로 이동
replace(to)	현재의 경로는 history에 추가하지 않고, 대체되는 경로를 추가
getRoutes()	현재 설정된 라우트 정보 조회

라우터 객체의 주요 메서드





내비게이션 가드/ 라우팅과 인증

내비게이션 가드

- **라우팅이 일어날 때 프로그래밍 방식으로 취소하거나 다른 경로로 리디렉션 시켜서 내비게이션을 안전하게 보호**
  - 인증된 사용자만이 접근할 수 있는 화면
    - 인증 토큰을 가지고 있으면 접근 가능하도록 처리
    - 인증 토큰이 없거나 파기되면 로그인화면으로 강제 이동 처리
  - 라우트하는 경로가 바뀔 때 반응성 작동
    - 동일한 경로에서 파라메터나 쿼리 문자열이 변경될 때는 작동하지 않아야 함.
- **적용 수준**
  - 전역 수준
  - 라우트 수준
  - 컴포넌트 수준

내비게이션 가드

- **전역 수준**
  - 라우터 객체 수준에서 등록
  - 모든 경로에 적용
  - 사용 가능한 가드
    - beforeEach
    - afterEach
    - beforeResolve - 라우트/컴포넌트 수준 가드가 모두 실행 후 실행
  - 내비게이션 정상 완료는 리턴이 없거나 true리턴
  - 내비게이션 취소는 false를 리턴
  - 리디렉션은 경로 문자열이나 Route객체 리턴
    - return '/videos/1', return {path: '/'}, return {name: 'member/id', params: {id:2}
- **라우트 수준**
  - 각 라우트 마다 가드를 설정
  - 내비게이션 가드를 여러 개 설정 가능
    - beforeEnter : [guard1, guard2]
- **컴포넌트 수준**
  - 생명주기 이벤트 훅과 동일한 방법

내비게이션 가드

- 전역수준, 라우트수준

```
const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', name:'home', component: Home },
        { path: '/about', name:'about', component: About },
        { path: '/members', name:'members', component: Members, },
        {
            path: '/members/:id', name:'members/id',
            beforeEnter:membersIdGuard
        },
        {
            path: '/videos', name:'videos',
            children : [
                { path: ':id', name:'videos/id', com
            ]
        },
    ]
})
```

```
const membersIdGuard = (to, from) => {
    // members/:id 경로는 반드시 이전 경로가
    // /members, /members:id 인 경우만 내비게이션 허용함
    if (from.name !== "members" && from.name !== "members/id") {
        return false;
```

라우트수준
false를 리턴했으므로,
내비게이션 취소

```
router.beforeEach((to)=> {
    //to.query에 속성이 존재할 경우 제거된 경로로 재이동
    if (to.query && Object.keys(to.query).length > 0) {
        return { path:to.path, query:{}, params: to.params };
    }
})

router.afterEach((to, from, failure)=> {
    if (isNavigationFailure(failure)) {
        console.log("@@ 내비게이션 중단 : ", failure)
        return { name:"home" };
    }
})

export default router;
```

전역수준
네비게이션
취소했을 때

전역수준
내비게이션
실패했을 때

내비게이션 가드

내비게이션 가드	설명
beforeRouteEnter	컴포넌트가 렌더링하는 경로가 확정되기 전에 호출 Options API를 사용하는 경우 아직 인스턴스가 생성되지 않았기 때문에 this 사용 불가
beforeRouteUpdate	컴포넌트를 렌더링하는 경로가 변경될 때 호출 새로운 경로이기는 하지만 기존 컴포넌트가 재사용 → 새롭게 마운트 되지 않음.
beforeRouteLeave	다른 경로로 벗어날 때 호출

내비게이션 가드 비교

Options API	Composition API
beforeRouteEnter	setup() 내부 코드 사용
beforeRouteUpdate	onBeforeRouteUpdate
beforeRouteLeave	onBeforeRouteLeave

내비게이션 가드 실행 순서

1. 내비게이션 시작
2. 비활성화된 컴포넌트에서 beforeRouteLeave가드 호출
3. 전역수준의 beforeEach가드 호출
4. 재사용되는 컴포넌트에서 beforeRouteUpdate가드 호출
5. 라우트 정보 수준의 beforeEnter가드 호출
6. 비동기 라우트 컴포넌트 분석후 로딩
7. 활성화된 컴포넌트에서 beforeRouteEnter가드 호출
8. 전역수준의 beforeResolve가드 호출
9. 내비게이션 수행 후 확정
10. 전역수준의 afterEach가드 호출
11. DOM 업데이트
12. 인스턴스에서 beforeRouteEnter가드에 인자로 전달된 next콜백함수 호출

내비게이션 가드

- 요청 경로에 쿼리 스트링이 있는 경우 제거
  - 전역 수준
- 이전 경로가 **/members, /members/:id**인 경우에만 **members/:id**로 이동
  - 라우트 수준
- **/videos/:id**에 의해 마운트, 렌더링하는 **VideoPlayer**컴포넌트에서 이전, 다음 버튼을 클릭하면 플레이할 영상을 **onBeforeRouteUpdate** 사용
  - 컴포넌트 수준

내비게이션 가드


```
const membersIdGuard = (to, from) => {
    // members/:id 경로는 반드시 이전 경로가
    // /members, /members:id 인 경우만 내비게이션 허용함
    if (from.name !== "members" && from.name !== "members/id") {
        return false;
    }
}
```

router/index.js

```
const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', name:'home', component: Home },
        { path: '/about', name:'about', component: About },
        { path: '/members', name:'members', component: Members, },
        {
            path: '/members/:id', name:'members/id', component: MemberInfo,
            beforeEnter:membersIdGuard
        },
        {
            path: '                    Videos,
            children
                { path:                    , component: VideoPlayer }
            ]
        },
    ]
```

라우트 수준

전역 수준

```
router.beforeEach((to)=> {
    //to.query에 속성이 존재할 경우 제거된 경로로 재이동
    if (to.query && Object.keys(to.query).length > 0) {
        return { path:to.path, query:{}, params: to.params };
    }
})

router.afterEach((to, from, failure)=> {
    if (isNavigationFailure(failure)) {
        console.log("@@ 내비게이션 중단 : ", failure)
        return { name:"home" };
    }
})

export default router;
```

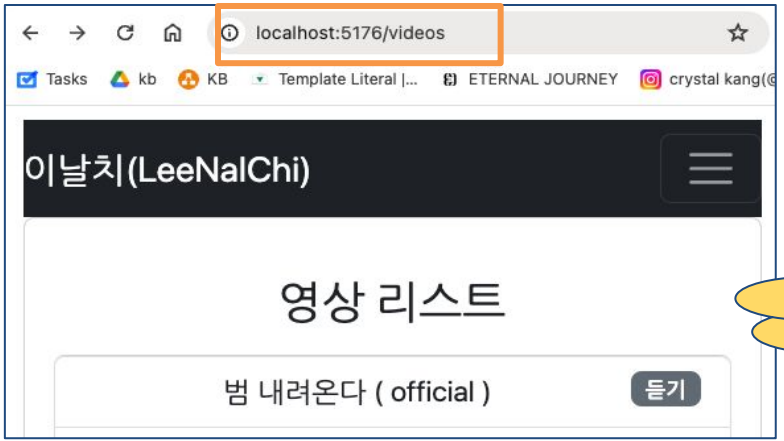
내비게이션 가드

```
export default {
    setup() {
        let videoInfo, currentIndex, prevVideoId, nextVideoId;
        videoInfo=reactive({ video: videos.find((v)=>v.id == currentRoute.params.id) })

        const getNavId = (to) => {
            videoInfo.video = videos.find((v)=>v.id == to.params.id);
            currentIndex = videos.findIndex((v)=>v.id == videoInfo.video.id)
            prevVideoId = videos[currentIndex-1] ? videos[currentIndex-1].id : null;
            nextVideoId = videos[currentIndex+1] ? videos[currentIndex+1].id : null;
        }
        //마운트되었을 때 현재의 라우트 정보를 이용해 이전, 다음 ID 획득
        getNavId(currentRoute)
        const stopVideo = () => {
            playerRef.value.player.stopVideo()
            router.push({ name:'videos' });
        }
        const playNext = () => {
            if (nextVideoId)
                router.push({ name:'videos/id', params: { id: nextVideoId } })
            else
                router.push({ name:'videos/id', params: { id: videos[0].id } })
        }
        const playPrev = () => {
            if (prevVideoId)
                router.push({ name:'videos/id', params: { id: prevVideoId } })
        }

        onBeforeRouteUpdate((to) => {
            getNavId(to)
        })

        return { videoInfo, playerRef, playNext, stopVideo,  playPrev };
    }
```

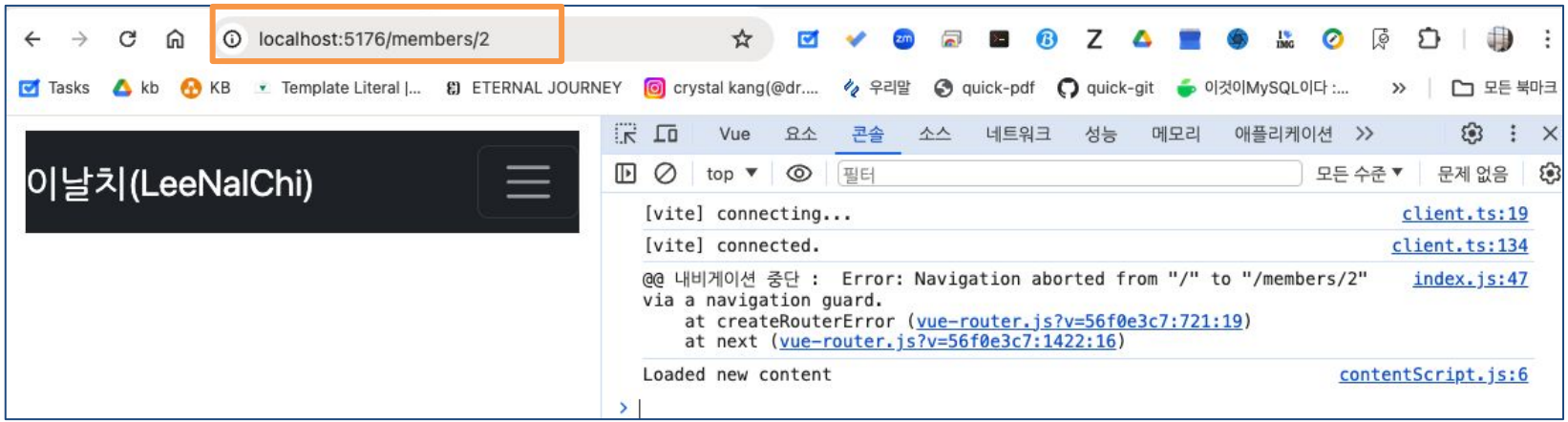
VideoPlayer.vue

컴포넌트 수준

내비게이션 가드

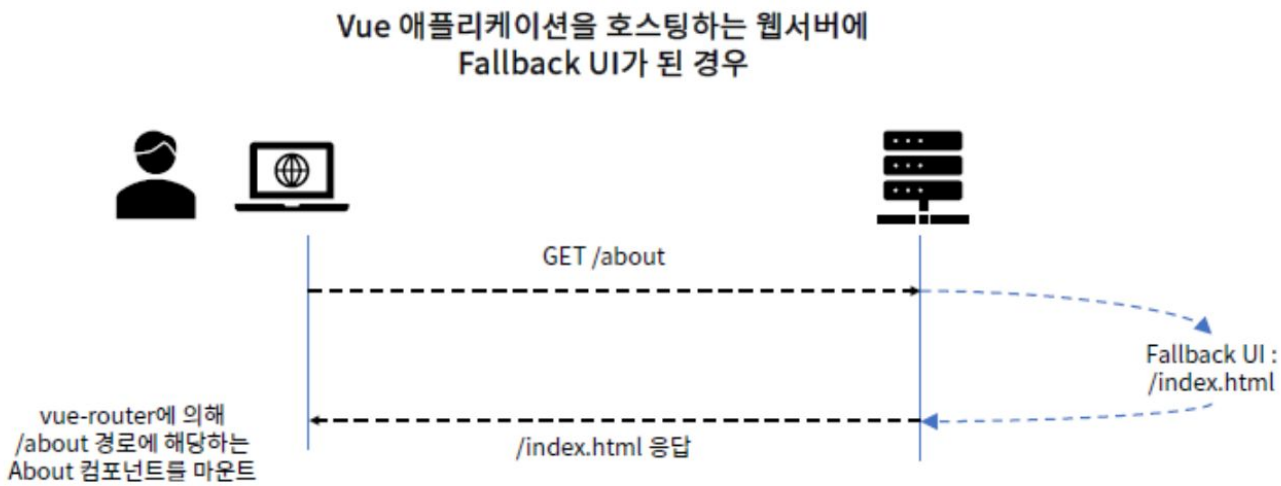


주소부분에 순서대로 호출



히스토리 모드

- 라우팅 모드와 동일
- **폴백 UI**
  - 요청한 경로의 문서가 존재하지 않을 때 응답할 기본 UI HTML
  - vite에는 내장되어있으나 배포할 때 웹서버에 별도로 폴백 UI설정해야함.



404라우트

- 응답 코드가 **404**번인 경우 특정한 화면이 나타나도록 함.

router/index.js

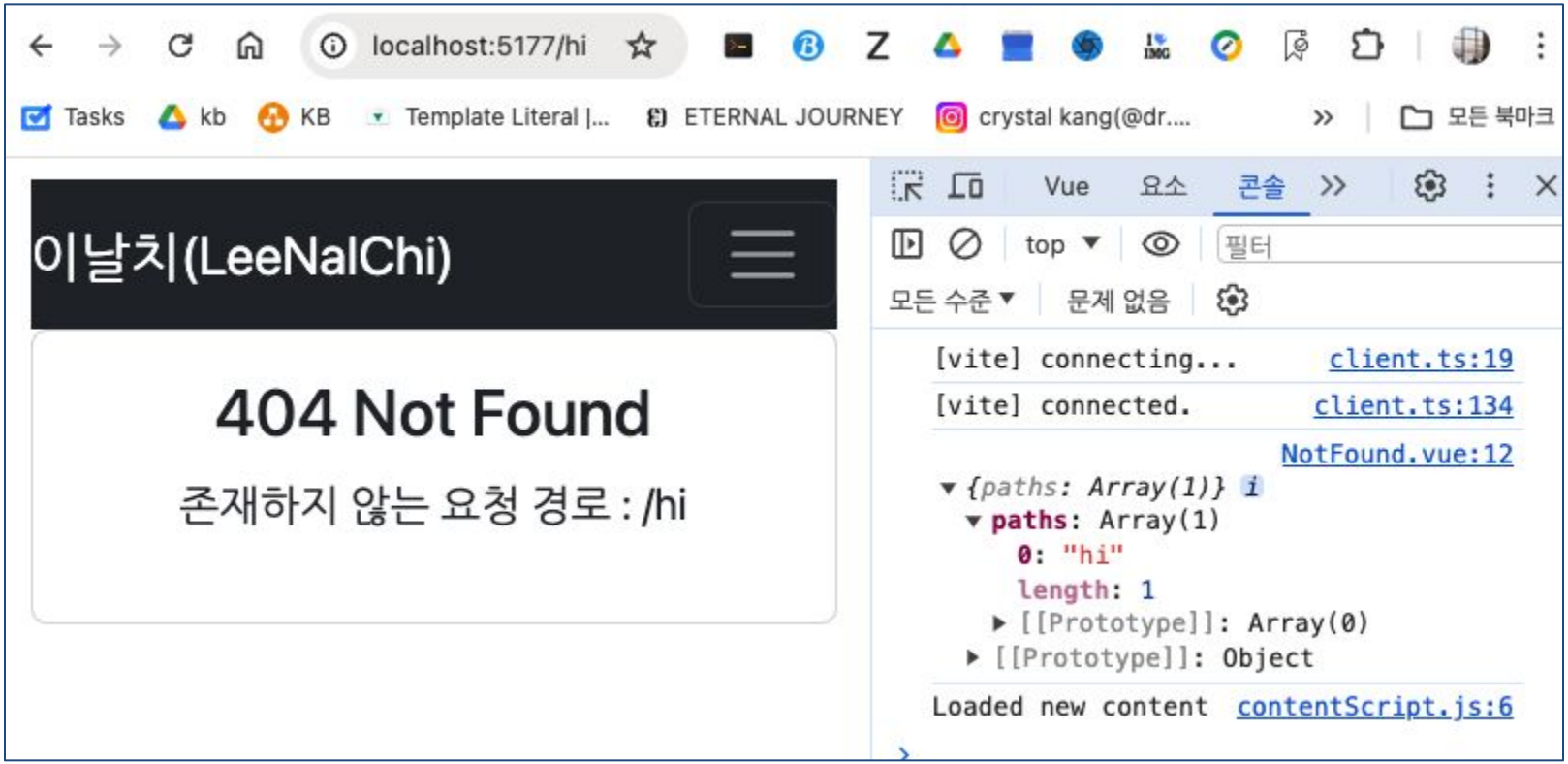
```
const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', name:'home', component: Home },
        { path: '/about', name:'about', component: About },
        { path: '/members', name:'members', component: Members, },
        {
            path: '/members/:id', name:'members/id', component: MemberInfo,
            beforeEnter:membersIdGuard
        },
        {
            path: '/videos', name:'videos', component: Videos,
            children : [
                { path: ':id', name:'videos/id', component: VideoPlayer }
            ]
        },
        { path: '/:paths(.*)*', name: 'NotFound', component: NotFound },
    ]
})
```

NotFound.vue

```
<template>
    <div class="card card-body">
        <h2>404 Not Found</h2>
        <p>존재하지 않는 요청 경로 : {{$route.fullPath}}</p>
    </div>
</template>

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

마지막 마우트로 등록
정규식으로 패턴 매핑

404라우트


localhost:5177/hi
Tasks
kb
KB
Template Literal |...
ETERNAL JOURNEY
crystal kang(@dr....
모든 북마크
이날치(LeeNalChi)
404 Not Found
존재하지 않는 요청 경로 : /hi
Vue
요소
콘솔
top
필터
모든 수준
문제 없음
[vite] connecting...
client.ts:19
[vite] connected.
client.ts:134
NotFound.vue:12
{paths: Array(1)}
paths: Array(1)
0: "hi"
length: 1
[[Prototype]]: Array(0)
[[Prototype]]: Object
Loaded new content
contentScript.js:6

라우트 정보 속성 연결

- **컴포넌트에서 router, currentRoute 관련 정보에 접근하는 경우**
  - $route, useRoute() 훅 사용
  - vue-route 하위 종속적인 컴포넌트로 설정
  - 동적 파라메터, 쿼리 스트링 정보 등을 전달 가능
- **routes 속성에 props: ture를 추가**
  - 동적 파라메터를 동일한 이름 속성으로 전달 가능

라우트 정보 속성 연결

index.js

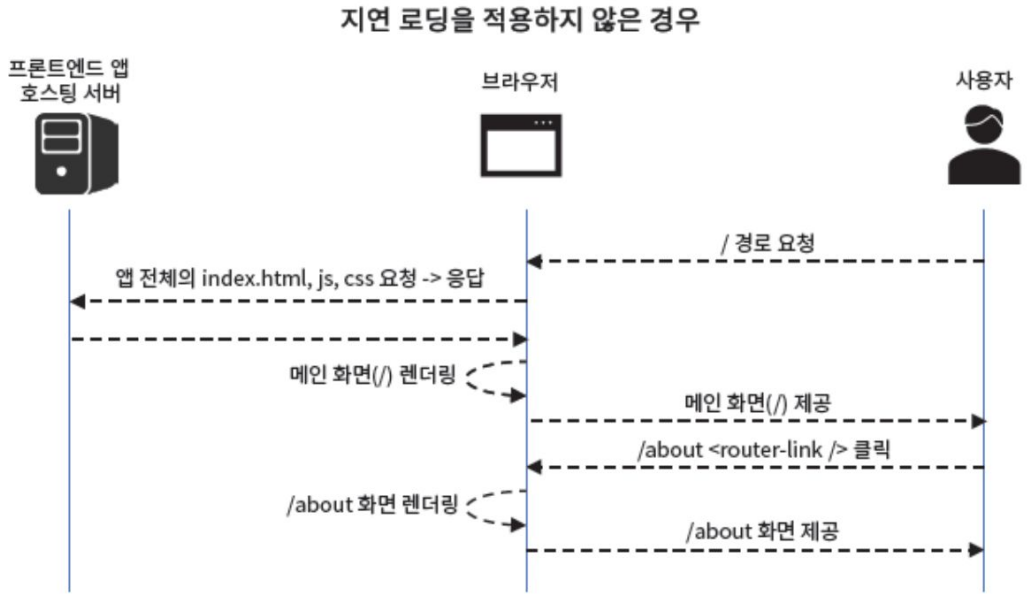
```
const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', name:'home', component: Home },
        { path: '/about', name:'about', component: About },
        { path: '/members', name:'members', component: Members, },
        {
            path: '/members/:id', name:'members/id', component: MemberInfo,
            beforeEnter:membersIdGuard, props:true
        },
```

memberInfo.vue

```
<script>
//import { useRoute } from 'vue-router'
import members from '@/members.json'

export default {
    name : "MemberInfo",
    props : ['id',],
    setup(props) {
        // const currentRoute = useRoute()
        // const id = parseInt(currentRoute.params.id, 10);
        // const member = members.find((m)=>m.id === id)
        const member = members.find((m)=>m.id === parseInt(props.id, 10))

        return { member }
    }
}
</script>
```

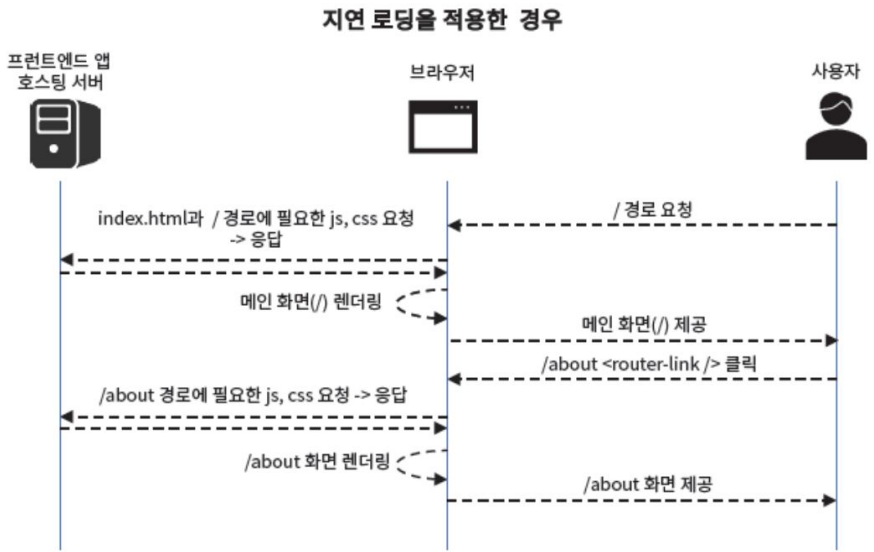
지연 로딩

- 프로젝트 빌드 시점이 아닌 컴포넌트가 이용되는 시점에 특정 컴포넌트와 모듈이 웹서버로부터 로딩
- **webpack, rollup** 등으로 모듈을 번들링하면 전체 코드가 몇 개의 **js**파일로 생성됨.
- 첫 화면 로딩시 미리 생성된 **is**파일들을 모두 웹서버로부터 다운로드 받아야 함.



지연 로딩

- **vue-router는 지원로딩 지원**
  - 컴포넌트 또는 화면 단위로 js파일을 분리시켜 빌드
  - 라우팅에 해당하는 컴포넌트가 사용될 때 js파일을 다운로드
  - 초기 화면의 로딩 속도 향상

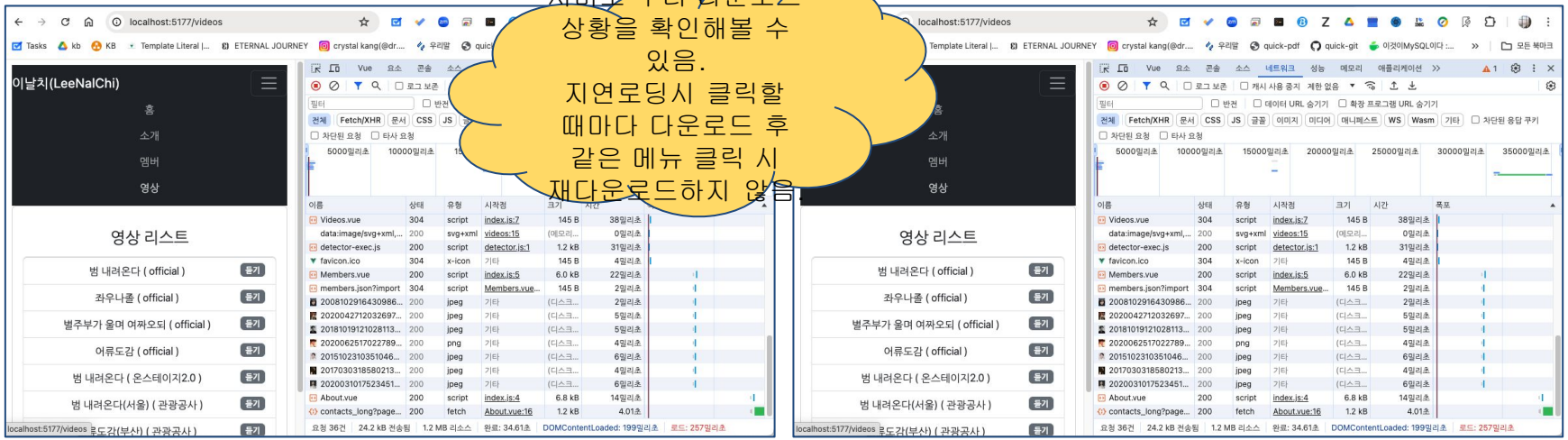


지연 로딩

index.js

```
const Home = () => import('@/pages/Home.vue');
const About = () => import('@/pages/About.vue');
const Members = () => import('@/pages/Members.vue');
const MemberInfo = () => import('@/pages/MemberInfo.vue');
const Videos = () => import('@/pages/Videos.vue');
const VideoPlayer = () => import('@/pages/VideoPlayer.vue');
const NotFound = () => import('@/pages/NotFound.vue');
```

서버로 부터 다운로드 상황을 확인해볼 수 있음.
지연로딩시 클릭할 때마다 다운로드 후 같은 메뉴 클릭 시 재다운로드하지 않음.



Suspense 컴포넌트

- 컴포넌트 로딩 시 지연시간 동안 로딩중임을 알리는 화면을 보여줄 때 사용

Loading.vue

```
<template>
    <VueCsspin message="Loading" spin-style="cp-flip" />
</template>

<script>
import { VueCsspin } from 'vue-csspin'
import 'vue-csspin/dist/vue-csspin.css'

export default {
    name : "Loading",
    components : { VueCsspin },
}
</script>
```

```
npm install vue-csspin
```

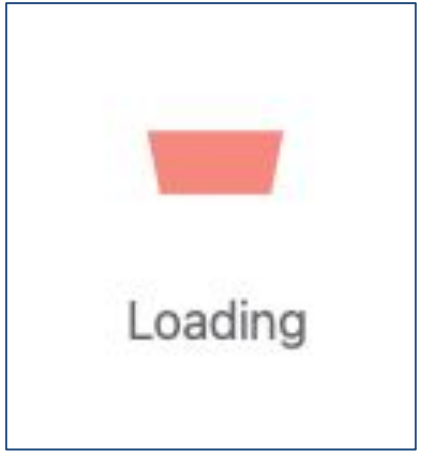
App.vue

```
<template>
  <div class="container">
    <Header />
    <router-view v-slot="{ Component }">
      <Suspense timeout="0">
        <Component :is="Component"></component>
        <template #fallback>
          <Loading />
        </template>
      </Suspense>
    </router-view>
  </div>
</template>

<script>
import Header from '@/components/Header.vue'
import Loading from '@/components/Loading.vue'
import { provide } from 'vue';

export default {
  name : "App",
  components : { Header, Loading },
```

Suspense 컴포넌트

About.vue

```vue
<script>
import { reactive } from "vue";

export default {
  async setup() {
    const user = reactive({ no:0, name:"", tel:"", address:"" });
    const url = "https://contactsvc.bmaster.kro.kr/contacts_long?pageno=1";
    const response = await fetch(url);
    const contactList = await response.json();
    user.no = contactList.contacts[0].no;
    user.name = contactList.contacts[0].name;
    user.tel = contactList.contacts[0].tel;
    user.address = contactList.contacts[0].address;

    return { user };
  }
}
</script>
```



청크 스플릿팅

- 한 프로젝트의 파일들을 여러 개의 청크로 쪼개서 빌드
- **vite 프로젝트**
  - 지연로딩 불적용 - 단 하나의 js로 빌드
  - 지연로딩 적용 - 컴포넌트 / 로딩 시점에 따라 여러 개 js로 빌드
- **몇 개의 컴포넌트를 묶어서 하나의 js로 빌드**
  - 기준 - 관련성, 로딩 시점이 동일??
- **webpackChuckName 라이브러리 사용**
  - vite.config.js에 추가 설정

```
npm install -D vite-plugin-webpackchunkname
```

vite.config.js

```
import { fileURLToPath, URL } from

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import { manualChunksPlugin } from 'vite-plugin-webpackchunkname'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [vue(), manualChunksPlugin() ],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url))
    }
  },
})
```

청크 스플릿팅

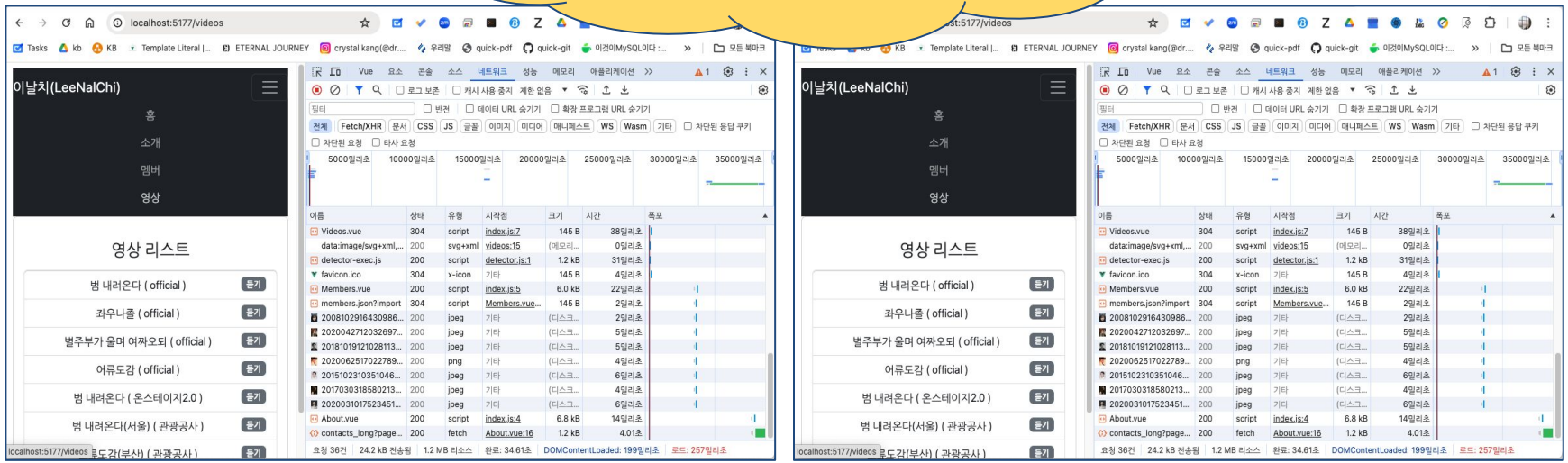
index.js

```
const Home = () => import(/* webpackChunkName: "home" */ '@/pages/Home.vue');
const About = () => import(/* webpackChunkName: "home" */ '@/pages/About.vue');
const Members = () => import(/* webpackChunkName: "members" */'@/pages/Members.vue');
const MemberInfo = () => import(/* webpackChunkName: "members" */'@/pages/MemberInfo.vue');
const Videos = () => import(/* webpackChunkName: "videos" */'@/pages/Videos.vue');
const VideoPlayer = () => import(/* webpackChunkName: "videos" */'@/pages/VideoPlayer.vue');
const NotFound = () => import(/* webpackChunkName: "home" */ '@/pages/NotFound.vue');
```

chunkName에 따라
그룹별로 js가 생성됨.

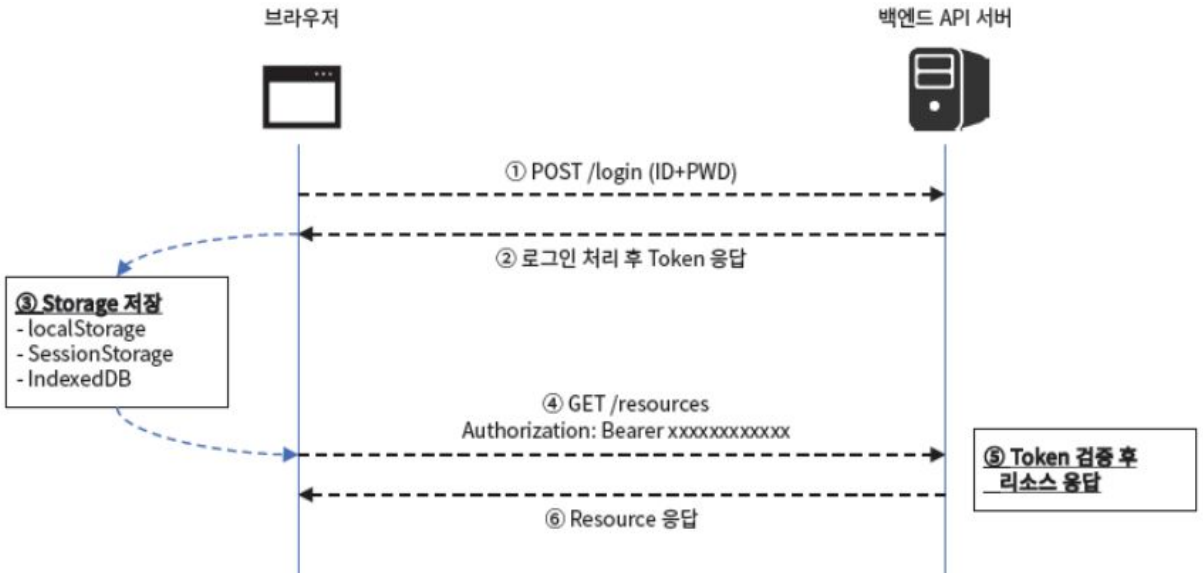
청크 스플릿팅

서버로 부터 다운로드 상황을
확인해볼 수 있음.
청크별로 다운로드 받는 파일을
확인할 수 있음.



라우팅과 인증 처리

- 토큰 기반 인증과 내비게이션 가드를 이용한 로그인 화면 제어
- 보통 인증 처리를 하려면
  - 인증 처리 백엔드 서버 필요
  - 외부 서버 필요 - Google Firebase

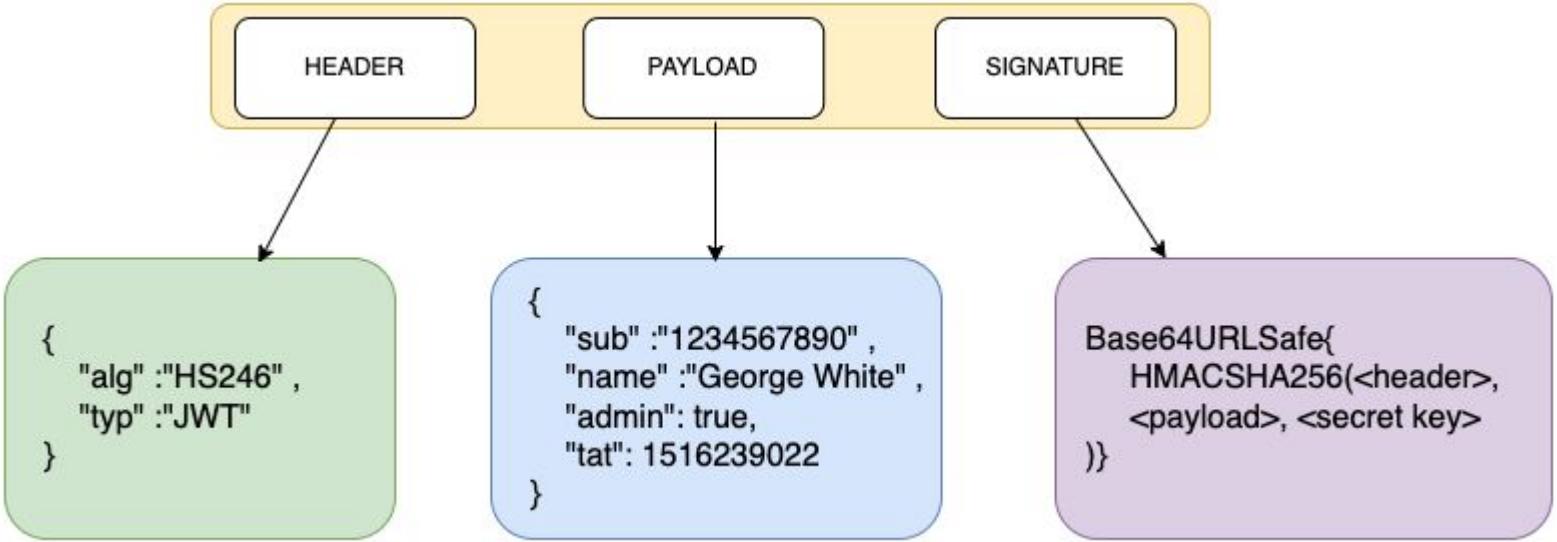


라우팅과 인증 처리

- **JWT(JSON Web Token)**
  - 가장 많이 사용되는 토큰 형식
  - 사용자 정보증명 정보과 역할 정보 등을 JSON 정보를 생성
  - 생성된 JSON정보를 웹 전송시 위변조가 불가능하도록 무결정 검증을 위한 서명을 첨부한 토큰

라우팅과 인증 처리

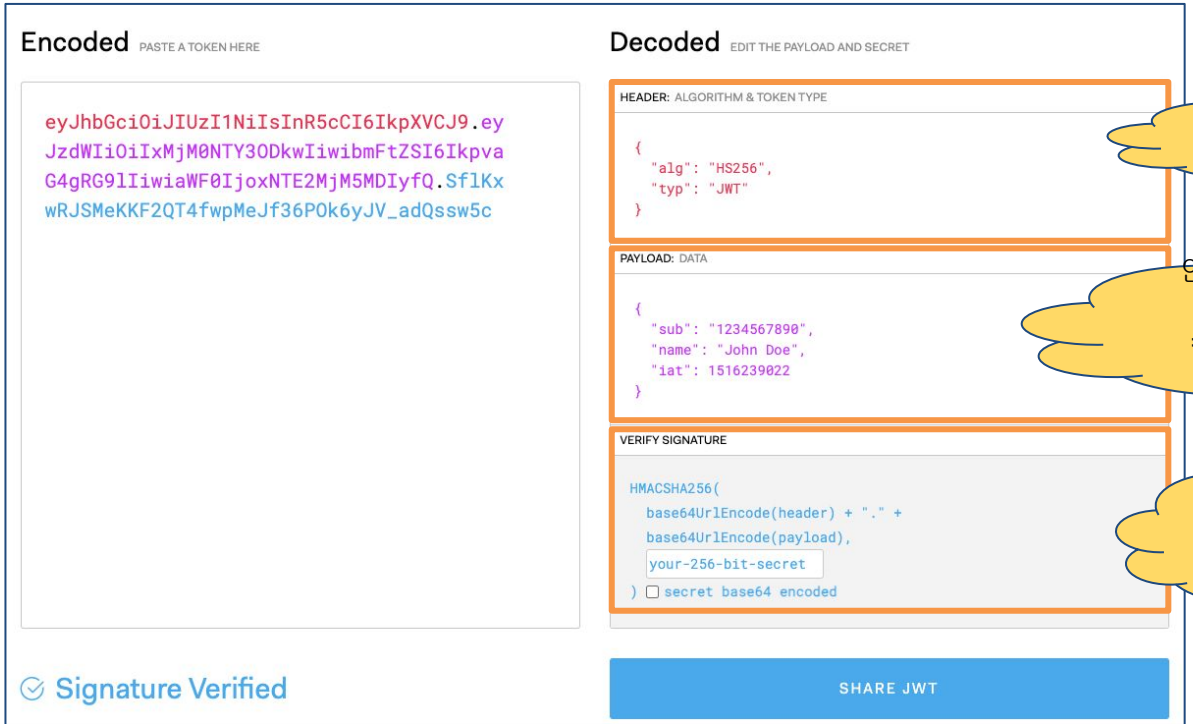
- **JWT**
  - JWT구조 = Header + Payload + Signature



라우팅과 인증 처리

- **JWT**
  - JWT구조 = Header + Payload + Signature
  - jwt.io

```
Signature = HMAC( ( Base64(Header) + "." + Base64(Payload) ), Secret)
세 번째 영역 서명 = Base64(Signature)
```



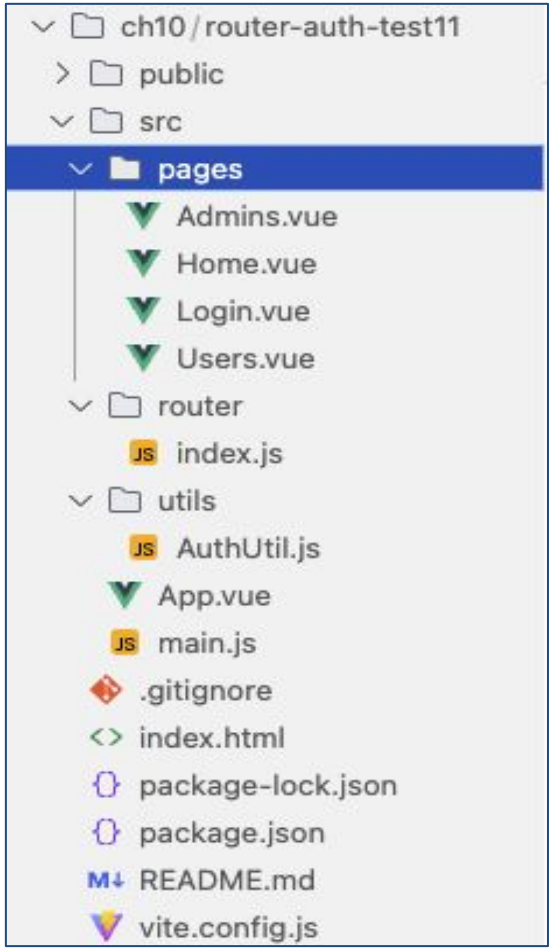
인코딩 알고리즘

인코딩할 데이터 → 알고리즘
디코딩
⇒ 변경 불가(비밀키가 없어
변경해도 무의미)

인증서버만이
가지는 비밀키로
인코딩데이터 검증

라우팅과 인증 처리

```
administrator@MacBook-Pro kb_ws_front % npm init vue auth-app

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
✔ Add Vue DevTools 7 extension for debugging? (experimental) … No / Yes

Scaffolding project in /Users/administrator/Documents/kb_ws_front/auth-app...

Done. Now run:

  cd auth-app
  npm install
  npm run dev

administrator@MacBook-Pro kb_ws_front % cd auth-app
administrator@MacBook-Pro auth-app % npm install

added 27 packages, and audited 28 packages in 10s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
administrator@MacBook-Pro auth-app % npm i vue-router$4

added 3 packages, and audited 31 packages in 1s

5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



라우팅과 인증 처리

- 주요 라우팅 경로

경로	라우트명	컴포넌트	설명
/	home	Home	홈 화면. 누구나 접근 가능(everybody)
/login	login	Login	로그인 화면. 누구나 접근 가능(everybody)
/users	users	Users	사용자 전용 화면. users 역할(role)을 가진 사용자만 접근 가능
/admins	admins	Admins	관리자 전용 화면. adimins 역할(role)을 가진 사용자만 접근 가능

- 내비게이션 가드
  - 브라우저에 보유한 토큰이 있는지 확인
  - 토큰이 없거나 파기되었으면 로그인 화면으로 이동
  - 로그인 후 로그인 전 접근했던 경로로 재이동

라우팅과 인증 처리

AuthUtil.js
→ 인증처리용 Util

User와
role정의

```
const staticUsers = [
    { userid:"user1", password:"1234", roles: ["users"] },
    { userid:"user2", password:"1234", roles: ["users"] },
    { userid:"admin", password:"1234", roles: ["users","admins"] },
]

const pathsToRoles = [
    { path:"/", roles: ["everybody"] },
    { path:"/login", roles: ["everybody"] },
    { path:"/users", roles: ["users","admins"] },
    { path:"/admins", roles: ["admins"] },
]
```

라우팅과 인증 처리

AuthUtil.js

```
//userInfo가 null이면 로컬 스토리지 삭제
const setUserInfo = (userInfo) => {
    if (userInfo && userInfo.authenticated) {
        window.localStorage.setItem("userInfo", btoa(JSON.stringify(userInfo)))
    } else {
        window.localStorage.removeItem("userInfo")
    }
}

const getUserInfo = () => {
    let strUserInfo = window.localStorage.getItem("userInfo");
    if (!strUserInfo) {
        return { authenticated: false};
    } else {
        return JSON.parse(window.atob(strUserInfo));
    }
}

const loginProcess = (userid, password, successCallback, failCallback) => {
    //이 부분은 백엔드 API 인증 서버와 HTTP로 통신하여 인증 처리해야 함.
    const user= staticUsers.find((u) => u.userid==userid && u.password==password);
    if (user) {
        let userInfo = { authenticated:true, userid:user.userid, roles: user.roles };
        setUserInfo(userInfo);
        successCallback();
    } else {
        if (failCallback) failCallback();
    }
}

const logoutProcess = (callback) => {
    setUserInfo(null);      //로컬 스토리지 삭제
    callback();
}
```

userInfo가 null아니면 로그인 정보 로컬 스토리지에 저장, null이면 로컬 스토리지 삭제

user에 대한 정보를 로컬 스토리지에서 획득

로그인 처리 → 로그인 성공하면 userInfo 정보를 로컬 스토리지에 저장

로그아웃 처리 → userInfo가 null이면 로컬 스토리지 삭제 요청

라우팅과 인증 처리

AuthUtil.js

```
//경로와 사용자 정보의 role을 기반으로 접근 허가 여부 결정(true/false)
const isMatchToRoles = (reqPath) => {
    //{ path:"/", roles: ["everybody"] }
    const path = pathsToRoles.find((pr)=>pr.path === reqPath);
    //경로가 없다면 접근 불가
    if (!path)  return false;

    const userInfo = getUserInfo();
    //인증되지 않았다면 everybody 가 지정된 경로만 접근 가능
    if (userInfo.authenticated === false) {
        return path.roles.find((p)=>p==="everybody") ? true : false;
    } else {
        //인증이 되었다면 userInfo의 roles와 path.roles에 동일한 것이 있어야 함.
        let isAccessible = false;
        if (path.roles.indexOf('everybody') > -1) {
            isAccessible = true;
        } else {
            for (let i=0; i < userInfo.roles.length; i++) {
                let role = userInfo.roles[i];
                const index = path.roles.indexOf(role);
                if (index >= 0) {
                    isAccessible = true;
                    break;
                }
            }
        }

        return isAccessible;
    }
}

export { isMatchToRoles, loginProcess, logoutProcess, getUserInfo };
```

user에 대한 정보를 이용해 접근 허가 여부 결정

라우팅과 인증 처리

router/index.js

```
import { createRouter, createWebHistory } from 'vu
import { isMatchToRoles } from '@/utils/AuthUtil.js';

import Home from '@/pages/Home.vue';
import Users from '@/pages/Users.vue';
import Admins from '@/pages/Admins.vue';
import Login from '@/pages/Login.vue';

const router = createRouter({
    history: createWebHistory(),
    routes : [
        { path: '/', name:'home', component: Home },
        { path: '/login', name:'login', component: Login },
        { path: '/users', name:'users', component: Users },
        { path: '/admins', name:'admins', component: Admins },
    ]
})

router.beforeEach((to)=>{
    if (!isMatchToRoles(to.path)) {
        return { name:'login', query: { fromname:to.name } };
    }
})

export default router;
```

내비게이션 가드 이용 → 로그인으로 전환되었던 사이트로 이동

라우팅과 인증 처리

App.vue

```
<template>
  <div class="container">
    <div>
      <router-link to="/">Home</router-link> 
      <router-link to="/login">Login</router-link> 
      <router-link to="/users">Users</router-link> 
      <router-link to="/admins">Admins</router-link>&nbs
    </div>
    <hr />
    <div>
      <router-view />
    </div>
  </div>
</template>

<script>
export default {
  name : "App"
}
</script>

<style>
.container { margin:10px; }
</style>
```

main.js

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

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

라우팅과 인증 처리

Login.vue

```vue
<template>
    <div>
        <h2>로그인</h2>
        사용자 : <input type="text" v-model="info.userid" /><br />
        암호  : <input type="password" v-model="info.password" /><br />
        <br />
        <button @click="login">로그인</button>
    </div>
</template>

<script>
import { reactive } from 'vue';
import { useRoute, useRouter } from 'vue-router';
import { loginProcess } from '@/utils/AuthUtil.js';

export default {
    name : "Login",
    setup() {
        const router = useRouter();
        const currentRoute = useRoute();
        const fromname = currentRoute.query.fromname;

        const info = reactive({ userid:"", password:"" });

        const successCallback = () => {
            if (fromname) router.push({ path: fromname })
            else router.push({ name:'home' })
        }
        const failCallback = () => {
            alert('로그인 실패');
        }

        const login = ()=> {
            loginProcess(info.userid, info.password, successCallback, failCallback)
        }

        return { info, login };
    }
}
</script>
```

라우팅과 인증 처리

Home.vue

```vue
<template>
    <div>
        <h2>Home</h2>
        <p>인증되지 않아도 접근 가능한 페이지</p>
        <div v-if="data.userInfo.authenticated">
            <p>사용자 : {{data.userInfo.userid}}</p>
            <p>사용자의 역할 : [ {{data.userInfo.roles.join(', ')}} ]</p>
            <button @click="logout">로그아웃</button>
        </div>
    </div>
</template>

<script>
import { getUserInfo, logoutProcess } from '@/utils/AuthUtil.js'
import { useRouter } from 'vue-router';
import { reactive } from 'vue';

export default {
    name : "Home",
    setup() {
        const router = useRouter();
        const data = reactive({ userInfo: getUserInfo() })
        const logout = () => {
            logoutProcess(()=>{
                data.userInfo = {};
                router.push({ name:'home' });
            })
        }
        return { data, logout }
    }
}
</script>
```

로그아웃 프로세스 처리

라우팅과 인증 처리

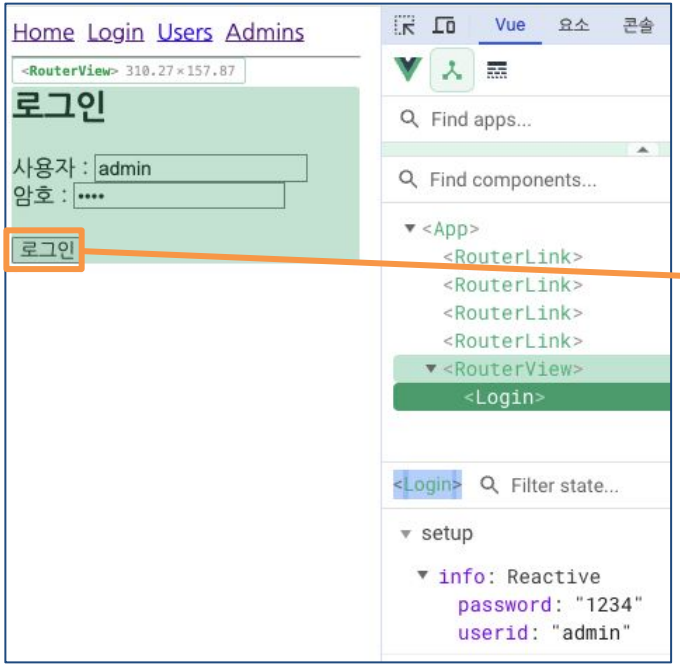
Users.vue

```vue
<template>
    <div>
        <h2>Users</h2>
        <p>users 역할이 있어야만 접근할 수 있는 페이지</p>
        <p>사용자 : {{userInfo.userid}}</p>
        <p>사용자의 역할 : [ {{userInfo.roles.join(', ')}} ]</p>
        <button @click="logout">로그아웃</button>
    </div>
</template>

<script>
import { getUserInfo, logoutProcess } from '@/utils/AuthUtil.js'
import { useRouter } from 'vue-router';

export default {
    name : "Users",
    setup() {
        const router = useRouter();
        const userInfo = getUserInfo();
        const logout = () => {
            logoutProcess(()=>{
                router.push({ name:'home' });
            })
        }
        return { userInfo, logout }
    }
}
</script>
```

Admins.vue

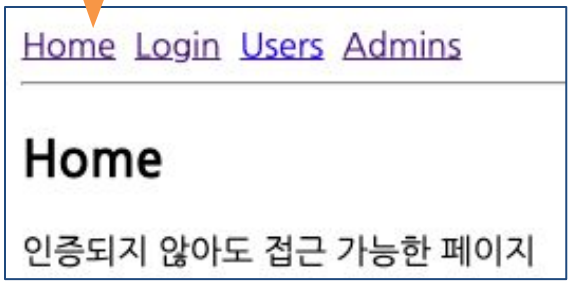
```vue
<template>
    <div>
        <h2>Admins</h2>
        <p>admins 역할이 있어야만 접근할 수 있는 페이지</p>
        <p>사용자 : {{userInfo.userid}}</p>
        <p>사용자의 역할 : [ {{userInfo.roles.join(', ')}} ]</p>
        <button @click="logout">로그아웃</button>
    </div>
</template>

<script>
import { getUserInfo, logoutProcess } from '@/utils/AuthUtil.js'
import { useRouter } from 'vue-router';

export default {
    name : "Admins",
    setup() {
        const router = useRouter();
        const userInfo = getUserInfo();
        const logout = () => {
            logoutProcess(()=>{
                router.push({ name:'home' });
            })
        }
        return { userInfo, logout }
    }
}
</script>
```

라우팅과 인증 처리

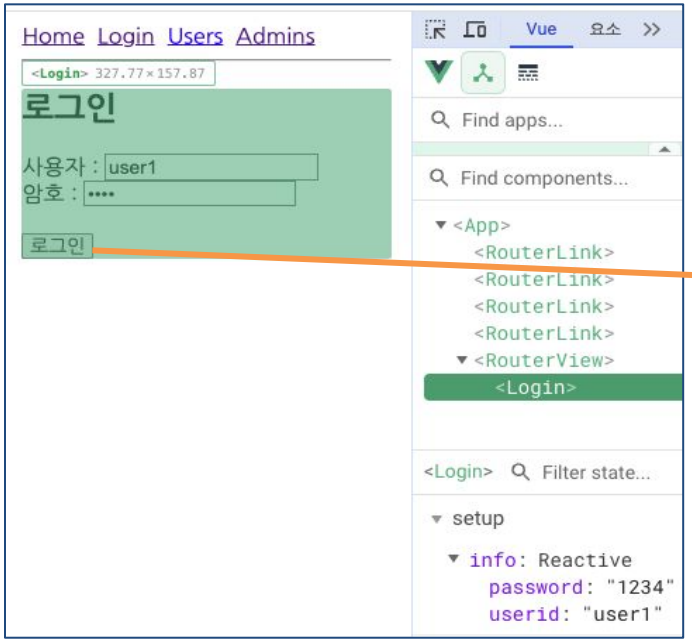

Home Login Users Admins
<RouterView> 310.27×157.87
로그인
사용자 : admin
암호 : ••••
로그인
Vue
요소
콘솔
Find apps...
Find components...
<App>
<RouterLink>
<RouterLink>
<RouterLink>
<RouterLink>
<RouterView>
<Login>
<Login>
Filter state...
setup
info: Reactive
password: "1234"
userid: "admin"


Home Login Users Admins
Home
인증되지 않아도 접근 가능한 페이지
사용자 : admin
사용자의 역할 : [users, admins]
로그아웃


Home Login Users Admins
Admins
admins 역할이 있어야만 접근할 수 있는 페이지
사용자 : admin
사용자의 역할 : [users, admins]
로그아웃

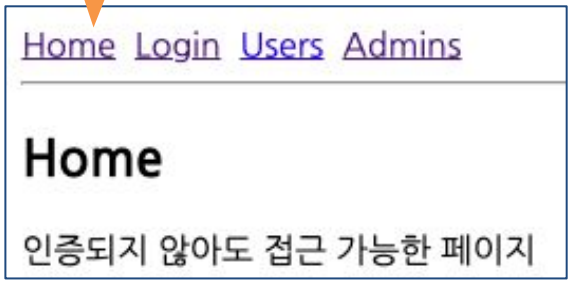

Home Login Users Admins
Home
인증되지 않아도 접근 가능한 페이지

라우팅과 인증 처리


Home Login Users Admins
<Login> 327.77×157.87
로그인
사용자 : user1
암호 : ••••
로그인
Vue
요소
Find apps...
Find components...
<App>
<RouterLink>
<RouterLink>
<RouterLink>
<RouterLink>
<RouterView>
<Login>
<Login> Filter state...
setup
info: Reactive
password: "1234"
userid: "user1"


Home Login Users Admins
Home
인증되지 않아도 접근 가능한 페이지
사용자 : user1
사용자의 역할 : [users]
로그아웃


Home Login Users Admins
Users
users 역할이 있어야만 접근할 수 있는 페이지
사용자 : user1
사용자의 역할 : [users]
로그아웃


Home Login Users Admins
Home
인증되지 않아도 접근 가능한 페이지

핵심 정리

- **vue-router** 기본 사용법
- **router, currentRoute** 객체
- 동적 라우트
- 중첩 라우트
- 명명된 라우트
- 명명된 뷰
- 라우터 객체 메서드
- 네비게이션 가드
- 히스토리 모드
- **404** 라우트, 지연 로딩
- **Suspense** 컴포넌트, 청크 스플릿팅
- 라우터와 인증 처리