KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

vue-router를 이용한 라우팅 3

[KB] IT's Your Life

스크립트 파트에서
라우팅하는 방법

KB국민은행

라우터 객체의 메서드

라우터 객체 접근

[Options API 적용]

```
const router = this.$router;
```

[Composition API 적용]

```
import { useRouter } from 'vue-router'
......
const router = useRouter();
```

useRouter함수로 라우터객체 받기

라우터 객체의 메서드

라우터는 라우팅 테이블을 가지고 있죠?

메서드	설명
addRoute(parentRouter, route)	실행시에 동적으로 부모 라우트에 새로운 라우트 정보를 추가합니다.
removeRoute(name)	실행시에 동적으로 라우트 정보를 삭제합니다.
go(n)	n만큼 브라우저 히스토리를 이용해 이동합니다. go(-1)를 호출하면 이전 방문 경로로 이동합니다.
back()	go(-1)과 같습니다.
forward()	go(1)과 같습니다.
push(to)	지정된 경로로 이동하고 브라우저 히스토리에 이동 경로를 추가합니다.
replace(to)	지정된 경로로 이동하지만 브라우저 히스토리에 새롭게 추가하지 않고 현재의 히스토리를 새로운 경로로 교체합니다.
getRoutes()	현재 설정된 라우트 정보를 조회할 수 있습니다

근데 잘 안씀

이동 관련

양수 앞으 음수 뒤로

히스토리 관계로 묶임

to는 어디로갈지 에대한 라우트정보

얘는 스택쌓지 않고 탑에 있는 것을 to로 교환

라우트들 리턴

얘를 통해 접근하여 동적으로 조작가능

라우터 객체의 메서드

동적 라우트 생성

[router 객체 생성 직후에...]

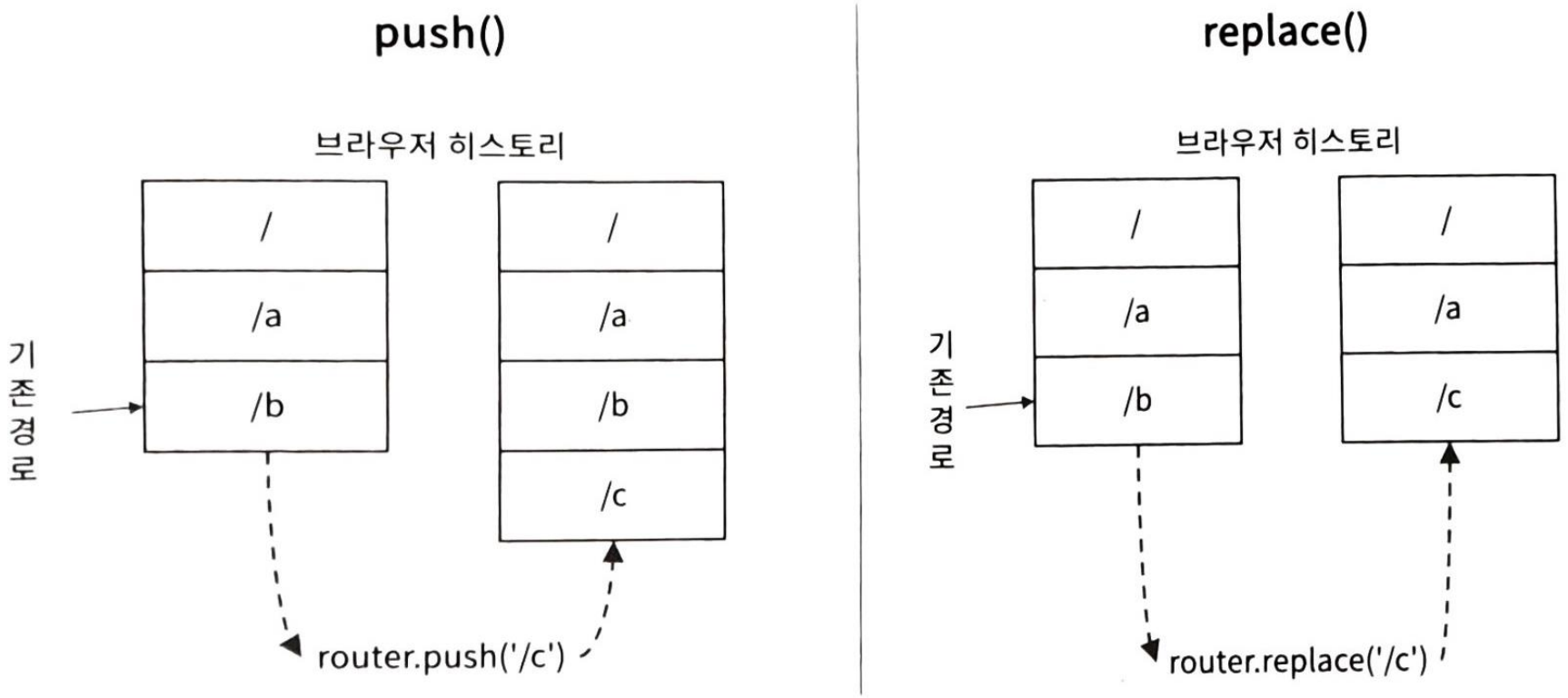
```
import { defineAsyncComponent } from 'vue'
......(생략)
const router = createRouter({
    history: createWebHistory(),
    routes : [
      ......(생략)
    ]
})

router.addRoute({
    path:'/repeat', name:'repeat',
    component: defineAsyncComponent(()=>import('@/pages/Repeat'))
})

export default router;
```

라우터 객체의 메서드

push()와 replace() 비교



라우터 객체의 메서드

push() 메서드

- 이동 경로를 정적인 문자열을 인자로 전달 가능
- 경로 정보를 담은 객체를 인자로 전달 가능

```
// 문자열 직접 전달
router.push('/home')
// 객체 정보로 전달
router.push({ path: '/about' })
// 명명된 라우트 사용
router.push({ name: 'members/id', params: { id: 1 } })
// 쿼리 문자열 전달 ex) /board?pageno=1&pagesize=5
router.push({ path: '/board', query: { pageno: 1, pagesize: 5 }})
```

라우트 패쓰로 추가 이동

객체를 통해서

패쓰에다가 쿼리스트링도

내비게이션 가드(Navigation Guard) 증잉에서 관리해야하는 상황에서.

- 라우팅이 일어날 때 프로그래밍 방식으로 내비게이션을 취소하거나, 다른 경로로 리디렉션
→ 내비게이션을 안전하게 보호하는 기능을 수행

- 예)
  - 인증 받은 사용자만 해당 페이지 접근
  - 인증 받지 않는 사용자인 경우 로그인 경로로 이동

- 라우트하는 경로(path)가 바뀔 때 반응
- 동일한 경로에서 파라미터나 쿼리 문자열이 변경될 때는 작동하지 않음

- 사용 수준
  - 전역 수준
  - 라우트 수준
  - 컴포넌트 수준

전역 수준 내비게이션 가드

- 모든 경로에 대해 내비게이션할 때 가드 함수가 호출됨
- 라우트 객체를 이용해서 등록
- beforeEach()는 내비게이션이 일어나기 전,
- afterEach()는 내비게이션이 일어난 후에 실행

전역 수준 내비게이션 가드

```
import { createRouter, createWebHistory, isNavigationFailure } from 'vue-router'
......
const router = createRouter({ ... })

router.beforeEach((to, from) => {
  // 내비게이션을 취소하려면 명시적으로 false를 리턴합니다.
  return false
})

router.afterEach((to, from, failure)=> {
  // 내비게이션을 실패했을 때 failure 정보를 이용해 실패 처리를 할 수 있습니다.
  if (isNavigationFailure(failure)) {

  }
})
```

beforeEach(함수)

함수의 리턴값

- 내비게이션을 정상적으로 진행 → 리턴하지 않거나, true 리턴
- 내비게이션 취소 → false 리턴
- 리다이렉트 하는 경우
  - 이동할 경로 문자열 또는 route 객체 리턴

```
예1) return '/videos/1'

예2) return { path: '/' }

예3) return { name: 'members/id', params: { id: 2 } }
```

- Error 객체 throw → router.onError()에 등록된 콜백 함수 호출

참고

beforeResolve 가드

- 컴포넌트 수준의 가드, 라우트 수준의 가드가 모두 실행된 후 내비게이션이 수행되기 직전에 실행
- 사용자가 특정 경로로 내비게이션하기 직전에 해당 경로의 화면에서 실행 시
  - 필요한 데이터를 가져오거나 필수적인 전처리 작업을 수행할 수 있음

라우트 수준의 내비게이션 가드

- 각 라우트 단위로 설정

```
const router = createRouter({
    ......
    routes : [
        ......
        {
            path: '/members/:id', name:'members/id', component: MemberInfo,
            beforeEnter : (to, from) => {
                //false를 리턴하면 내비게이션이 중단됩니다.
            }
        },
        ......
    ]
})
```

인증 가드
라고도 불림

라우트 수준의 내비게이션 가드

- 여러 개의 함수 등록

```
const guard1 = (to, from) => {
}

const guard2 = (to, from) => {
}

const router = createRouter({
    ......
    routes : [
        ......
        {
            path: '/members/:id', name:'members/id', component: MemberInfo,
            beforeEnter : [ guard1, guard2 ]
        },
        ......
    ]
})
```

가드가 여러개 필요할때

컴포넌트 수준의 내비게이션 가드

이 페이지에 들어가려면 이 데이터가 필요해 라는 상황일때

잘 사용 안됨

내비게이션 가드	설명
beforeRouteEnter	컴포넌트가 렌더링하는 경로가 확정되기 전에 호출됩니다. Options API를 사용하는 경우 이 시점에서는 인스턴스가 생성되지 않았기 때문에 this를 이용할 수 없습니다.
beforeRouteUpdate	컴포넌트를 렌더링하는 경로가 바뀔 때 호출됩니다. 새로운 경로이지만 기존 컴포넌트가 재사용됩니다. 즉 새롭게 마운트되지 않고 컴포넌트는 업데이트됩니다. 예를 들어 앞서 이미 작성한 예제와 같이 /videos/:id 경로에 의해 VideoPlayer 컴포넌트가 마운트되도록 라우트를 설정했을 때, /videos/abc에서 /videos/def로 이동하면 이 이벤트 훅이 실행됩니다. 하지만 VideoPlayer 컴포넌트는 이미 마운트되어 있기 때문에 재사용됩니다.
beforeRouteLeave	현재의 경로에서 다른 경로로 벗어날 때 호출됩니다.

컴포넌트 수준의 내비게이션 가드

Options API에서 내비게이션 가드

```
export default {
  beforeRouteEnter(to, from) {
  },
  beforeRouteUpdate(to, from) {
  },
  beforeRouteLeave(to, from) {
  },
}
```

컴포넌트 수준의 내비게이션 가드

Composition API에서 내비게이션 가드

Options API	Composition API
beforeRouteEnter	setup() 메서드 내부의 코드로 대체
beforeRouteUpdate	onBeforeRouteUpdate
beforeRouteLeave	onBeforeRouteLeave

내비게이션 가드 실행 순서

1. 내비게이션 시작됨
2. 비활성화되는 컴포넌트에서 beforeRouteLeave 가드가 호출됨
3. 전역 수준의 beforeEach 가드가 호출됨
4. 재사용되는 컴포넌트에서 beforeRouteUpdate 가드가 호출됨
5. 라우트 정보 수준의 beforeEnter 가드가 호출됨
6. 비동기 라우트 컴포넌트가 분석되고 로딩됨
7. 활성화된 컴포넌트에서 beforeRouteEnter 가드가 호출됨
8. 전역 수준의 beforeResolve 가드가 호출됨
9. 내비게이션이 수행되고 확정됨
10. 전역 수준의 afterEach 가드가 호출됨
11. DOM이 업데이트 됨
12. 인스턴스에서 beforeRouteEnter 가드에 인자가 전달된 next 콜백 함수를 호출함

내비게이션 가드 적용하기

- 요청 경로에 혹시라도 쿼리 스트링 파라미터가 있다면 모두 제거하도록 기능 구현
  → 전역 수준의 내비게이션 가드를 적용

- 이전 경로가 /members, members/:id인 경우만 members/:id 경로로 이동하도록 변경
  → 라우트 수준의 내비게이션 가드 적용

- /videos/:id 경로에 의해 마운트, 렌더링하는 VideoPlayer 컴포넌트에서 이전, 다음 버튼을 클릭하면 플레이할 영상을 onBeforeRouteUpdate를 사용하도록 코드를 변경

→ 컴포넌트 수준의 내비게이션 가드를 사용

src/router/index.js

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

src/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 }
            ]
        },
    ]
})
```

src/router/index.js

```
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;
```

src/pages/VideoPlayer.vue

```
…
<script>
import { reactive, ref, inject } from 'vue'
import { useRoute, useRouter, onBeforeRouteUpdate } 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, 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;
        }
```

src/pages/VideoPlayer.vue

```
        //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 };
    }
}
</script>
```

테스트

- http://localhost:5173/about?a=1&b=2 을 직접 입력해서 요청
  → 쿼리 파라미터가 제거됨

- /members를 거치지 않고 /member/2 경로로 직접 이동해 봄
  → 네비게이션 중단 로드가 콘솔로 나타냄

- /videos/:id로 이동하여 영상을 플레이하고, 이전, 다음 버튼을 클릭하여 정상 작동 여부를 확인