Router

Router - 1

- **Router**

SPA에서의 라우팅은 클라이언트(브라우저)에서 처리되며 URL에 해당하는 내용을
클라이언트에서 비동기 요청을 보내어 데이터를 응답받고 렌더링함

즉, 페이지의 새로고침 없이 URL을 이동하여 뷰에 접근할 수 있음

- **SPA(Single Page Application)**

단일 페이지로 사용 가능한 애플리케이션을 의미하며 페이지 이동 등의 행동이
발생할 경우 웹 페이지 전체가 바뀌는게 아닌, 처음 로딩된 페이지 중에서
변경이 필요한 부분만 바뀜

그로 인해 페이지 전환 속도가 굉장히 빠르고, 이미 로딩이 완료된걸 서버로부터
다시 받아올 필요가 없기 때문에 서버의 자원을 효율적으로 사용할 수 있음

* 웹에서 이동 가능한 모든 페이지에 대한 파일을 받아와서 js에 의해
  자체적으로 페이지가 로딩됨

Router - 2

- **셋팅**

VSCode 터미널에서 아래 명령어 입력

npm install vue-router

Router - 3

- **src/router/index.js**

router를 사용하기 위한 초기 import

```
import { createRouter, createWebHistory } from 'vue-router'
import First from '../views/FirstPage.vue'
import Second from '../views/SecondPage.vue'

const routes = [
  { path: '/First', name:"first", component: First},
  { path: '/Second', name:"second", component: Second}
];

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

export default router;
```

path : URL
name : 지정할 이름
component : import한 파일

router를 사용할 수 있도록 생성

생성된 router를 외부에서 사용할 수 있도록 내보냄

Router - 4

- src/App.vue

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

import { createApp } from 'vue'
import App from './App.vue'
import router from './router/index'
const app = createApp(App)
app.use(router)
app.mount('#app')
```

가져올 파일의 경로 및 이름

사용할 이름

router 사용 및 id가 app인 공간에 마운트

Router - 5

- **src/main.js**

app.js에서 mount를 정의한 id

```
<template>
  <div id="app">
    App.vue 페이지
    <nav>
      <ul>
        <li>
          <router-link to="/first">First</router-link>
        </li>
        <li>
          <router-link to="/Second">Second</router-link>
        </li>
      </ul>
    </nav>
    <hr><br>
    <router-view></router-view>
  </div>
</template>
```

이동할 URL (당장은 a태그라고 생각해도 됨)

가져온 페이지 정보를 보여줄 공간

Router - 6

- 뷰 파일

src/views/FirstPage.vue

```vue
<template>
    <div id="btn">
        <button @click="count++">
            First : {{ count }}번 클릭되었습니다
        </button>
    </div>
</template>
<script>
export default {
    data() {
        return {
            count: 0
        }
    }
}
</script>
```

해당 페이지의 vue를 외부에서 사용할 수 있도록 내보내줌

src/views/SecondPage.vue

```vue
<template>
    <div id="btn">
        <button @click="count++">
            Second : {{ count }}번 클릭되었습니다
        </button>
    </div>
</template>
<script>
export default {
    data() {
        return {
            count: 0
        }
    }
}
</script>
```

Router 활용 - 1
(404 예외 처리)

Router 활용 – 404 예외 처리

- **404 에러**

  사용자가 사이트에 존재하지 않는 URL에 접근했을 때 발생하는 에러

- **HTML Error Code**

  - 1xx (조건부 응답)
    → 일반적으로 보기 힘든 에러 코드이기 때문에 생략
  - 2xx (성공)
    → 요청이 성공했을 때 발생하는 200번대 코드
  - 3xx (리다이렉션)
    → 요청한 경로가 변경되었을 때 발생하는 300번대 코드
  - 4xx (요청 오류)
    → 사용자가 잘못된 요청을 했을 때 발생하는 400번대 코드
  - 5xx (서버 오류)
    → 서버의 문제로 응답할 수 없을 때 발생하는 500번대 코드

Router 활용 – 404 예외 처리

- **자주 발생하는 4XX, 5XX 코드**

HTTP CODE	설명
400(Bad Request)	잘못된 요청. 문법상 오류가 있어서 서버가 해석하지 못함
401(Unauthorized)	요청에 대한 인증 권한이 없음
403(Forbidden)	요청을 거절함. 해당 파일에 접근하기 위한 권한이 없음
404(Not Found)	요청한 페이지를 찾을 수 없음
405 (Method Not Allowed)	요청한 HTTP Method를 허용하지 않음
501(Internal Server Error)	서버가 요청받은 기능을 수행할 수 없음
502(Bad Geteway)	서버가 게이트웨이 또는 프록시 역할을 하고 있거나 업스트림 서버에서 잘못된 응답을 받음
503(Service Unavailable)	서버가 중단되었거나 과부하로 인해 요청할 수 없는 상태
504(Gateway Timeout)	서버의 응답이 매우 느려 통신할 수 없는 상태

Router 활용 – 404 예외 처리

- **뷰 파일**

  src/views/NotFound.vue

```
<template>
    <h1>404</h1> 찾을
    <p>페이지를 수 없습니다. ☹</p>
</template>
```

Router 활용 – 404 예외 처리

- **src/router/index.js**

```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../App.vue'
import First from '../views/FirstPage.vue'
import Second from '../views/SecondPage.vue'
import NotFound from '../views/NotFound.vue'

const routes = [
  { path: '/', name:"Home", component: Home},
  { path: '/First', name:"first", component: First},
  { path: '/Second', name:"second", component: Second},
  { path: '/404', name: "notFound", component: NotFound},
  { path: '/:pathMatch(.*)*', redirect: "/404"}
];

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

export default router;
```

404 발생 시 리다이렉트할 페이지 불러오기

URL입력 없을 때 불러올 페이지

404입력 시 불러올 페이지

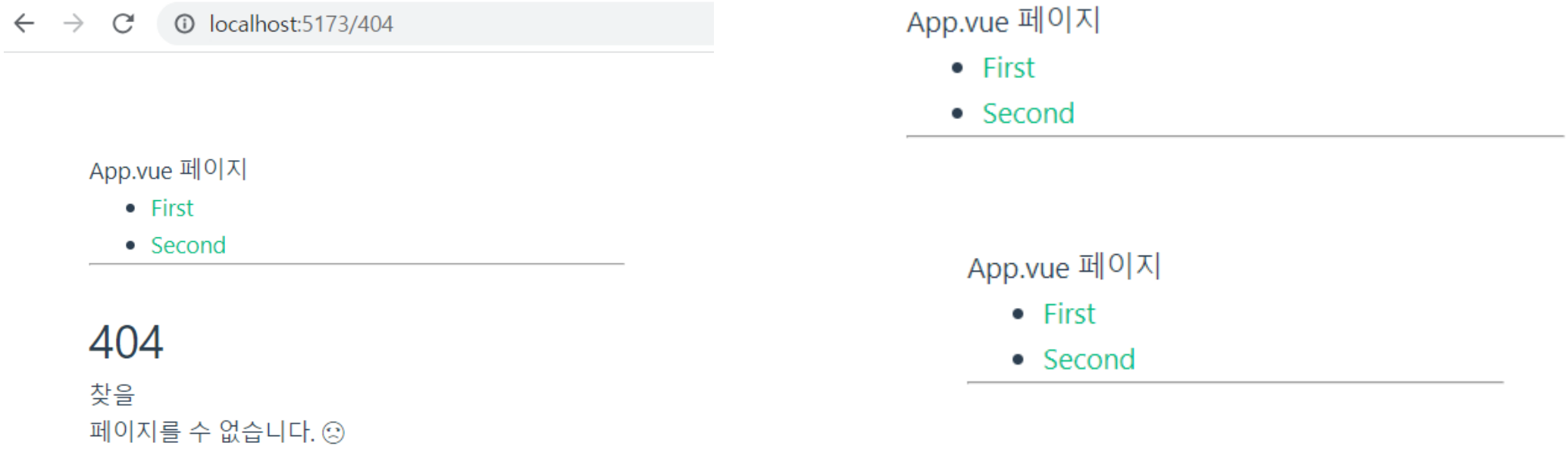
path에 맞을 경우 /404 URL로 리다이렉트

`/:` 경로의 첫번째 세그먼트(루트 /)
`patchMatch(.*)` 0개 이상의 문자를 의미
`*` 모든 문자를 의미

Router 활용 – 404 예외 처리

- **적용 후**

URL에 아무런 값을 입력할 경우 정상적으로 404 페이지를 가져 옴
하지만, 홈 페이지가 2번 로딩되는 등의 문제가 발생



Router 활용 – 404 예외 처리

- 뷰 파일

src/views/Home.vue

```
<template>
    <nav>
        <ul>
        <li>
            <router-link to="/first">First</router-link>
        </li>
        <li>
            <router-link to="/Second">Second</router-link>
        </li>
        </ul>
    </nav>
</template>
```

src/App.vue

```
<template>
  <div id="app">
    App.vue 페이지
    <router-view></router-view>
  </div>
</template>
```

Router 활용 – 404 예외 처리

- **적용 후**

각 파일의 내용을 나눠서 사용하여 중복되는 내용 제거

App.vue 페이지

404

찾을
페이지를 수 없습니다. ☹

App.vue 페이지

- First
- Second

Router 활용 - 2
('홈으로 돌아가기' 버튼 만들기)

Router 활용 – '홈으로 돌아가기'버튼 만들기

- **src/views/FirstPage.vue**

v-on:click 디렉티브 사용하는 방법

```
<template>
    <div id="btn">
        <!-- <button @click="$router.push('/')">홈으로 돌아가기</button> -->
        <button @click="goHome()">홈으로 돌아가기</button>
        <br>
        <button @click="count++">
            First : {{ count }}번 클릭되었습니다
        </button>
    </div>
</template>
<script>
export default {
    data() {
        return {
            count: 0
        }
    },
    methods: {
        goHome() {
            this.$router.push('/');
        }
    }
}
</script>
```

클릭 시 goHome 함수 호출

Router 활용 – '홈으로 돌아가기'버튼 만들기

- **src/views/FirstPage.vue**

v-on:click 디렉티브 사용하는 방법

```
<template>
    <div id="btn">
        <!-- <button @click="$router.push('/')">홈으로 돌아가기</button> -->
        <button @click="goHome()">홈으로 돌아가기</button>
        <br>
        <button @click="count++">
            First : {{ count }}번 클릭되었습니다
        </button>
    </div>
</template>
<script>
export default {
    data() {
        return {
            count: 0
        }
    },
    methods: {
        goHome() {
            this.$router.push('/');
        }
    }
}
</script>
```

클릭 시 goHome 함수 호출

Router 활용 – '홈으로 돌아가기'버튼 만들기

- **src/views/FirstPage.vue**

v-on:click 디렉티브에 직접 입력하는 방법

```
<template>
    <div id="btn">
        <!-- <button @click="$router.push('/')">홈으로 돌아가기</button> -->
        <button @click="goHome()">홈으로 돌아가기</button>
        <br>
        <button @click="count++">
            First : {{ count }}번 클릭되었습니다
        </button>
    </div>
</template>
<script>
export default {
    data() {
        return {
            count: 0
        }
    },
    methods: {
        goHome() {
            this.$router.push('/');
        }
    }
}
</script>
```

클릭 시 goHome 함수 호출하는 방법

현재 나의 인스턴스 객체의 $router 접근 후 URL / 을 Push함