Nuxt.js vs Vue.js - SSR 시작하기

bluestragglr · 2020년 4월 28일

SSR nuxt.js seo vue.js

> 본 포스트는 Vue 프로젝트의 구조를 바탕으로 Nuxt.js를 사용해 보기 위해 기존 코드와의 차이점을 비교하며 정리한 포스트입니다.

시작하며

Nuxt.js는 Vue.js 프레임워크를 기반으로 SSR(Server Side Rendering) 웹 페이지를 만들 수 있도록 해 주는 라이브러리입니다. SEO 등의 문제로 CSR이 아닌 SSR 웹을 구축해야 하는 경우에 유용하게 사용할 수 있습니다.

바쁜 현대인을 위한 요약

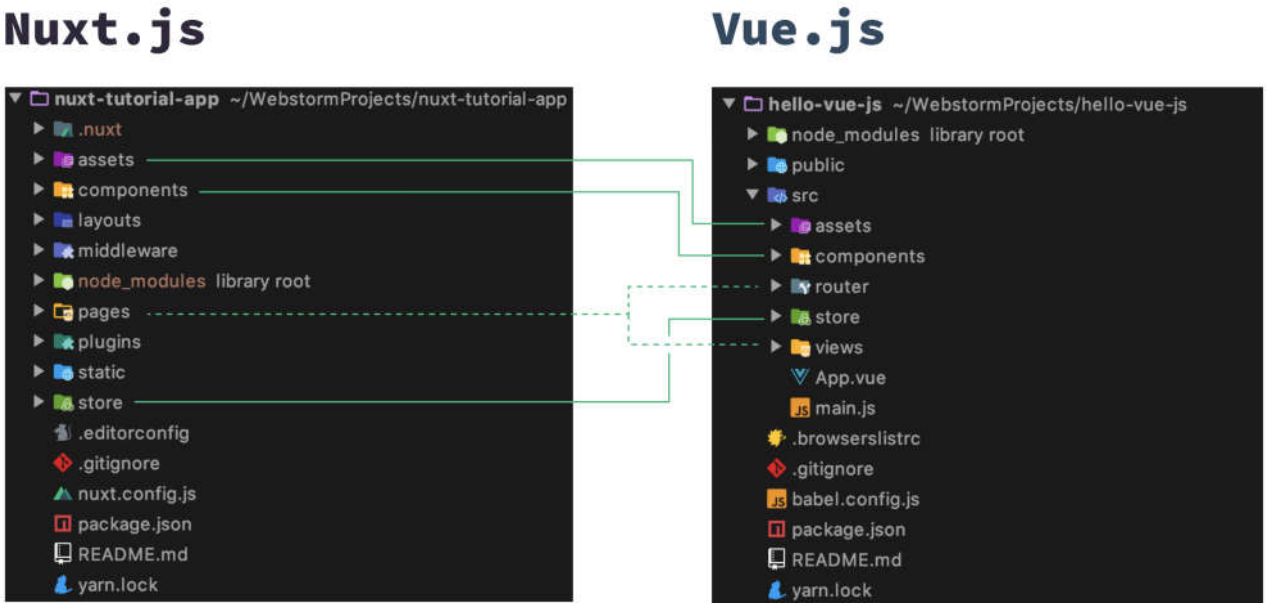
- 프로젝트 전반적인 폴더 구조가 조금 다르지만 대체로 비슷하게 동작합니다.
- 라우터 셋업이 달라집니다. 디렉토리 구조에 따라 자동생성하는 것을 디폴트로 합니다.
- 개발 및 배포 환경이 조금 달라집니다.
- Layout을 컴포넌트의 프로퍼티로써 사용할 수 있습니다.

목차

구조적 차이

npx로 생성한 Nuxt 프로젝트와 vue-cli3으로 생성한 Vue.js 프로젝트의 구조를 비교해 보겠습니다.

- Nuxt의 경우에는 Universal(non-SPA) 앱이면서 vuex를 사용하도록 설정하였습니다.
- Vue.js의 경우에는 vue-router와 vuex를 사용하도록 설정하였습니다.



전반적으로 Vue.js 프로젝트의 src 폴더 아래에 있던 코드들이 루트 레벨로 올라왔습니다. 이름이 그대로 유지되는 디렉토리는 기능 또한 그대로 유지되는 것으로 보입니다. (Asset, Component, Store 등)

router의 경우 Vue.js에서는 선택적으로 사용하는 기능인 반면 nuxt에는 항상 포함되어 있는 기능으로 변화하였습니다. vue-cli로 프로젝트를 생성하면 router의 사용 여부를 묻고 사용하는 것으로 선택한 경우 `router` 및 `views` 폴더가 생성됩니다. 반면, npx로 Nuxt 프로젝트를 생성하는 경우에는 `pages` 폴더가 항상 존재하게 되며, `router`와 `views` 폴더를 대체합니다. 이 때, 기존에 `router/index.js` 에 직접 라우터를 등록해 주던 것과 달리 Nuxt에서는 빌드시 자동으로 `pages` 폴더의 구조대로 라우터를 생성해 줍니다.

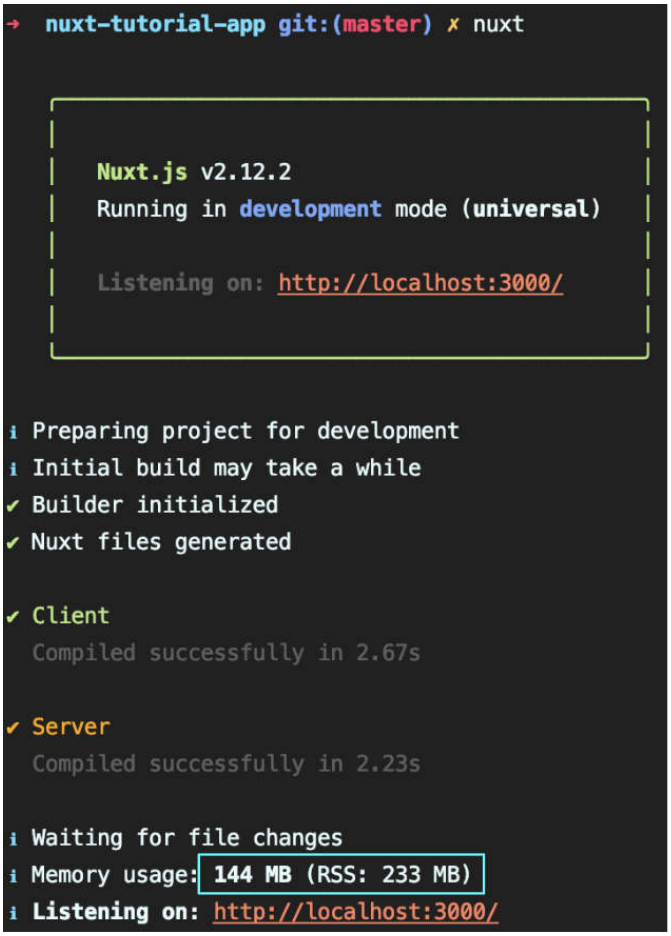
프로젝트를 빌드했을 때 `dist` 폴더 안에 생성되던 파일들은 `.nuxt` 안으로 이동했습니다. 물론 CSR이 아닌 SSR이 되었기에 빌드 결과물의 형태는 조금 달라졌습니다. 빌드에 대해서는 바로 아래에서 좀 더 자세히 이야기해 봅시다.

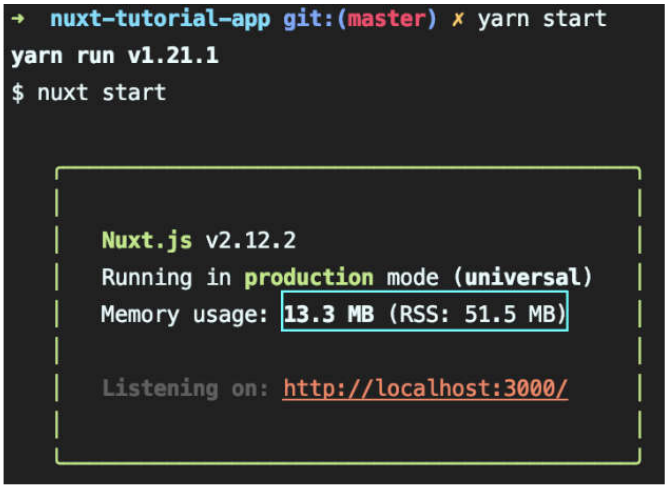
개발 환경 및 배포에서의 차이

핫 리로드(코드 수정에 따라 실시간으로 반영되는 환경)만 가능한 Vue.js와는 달리, Nuxt의 경우 핫 리로드와 로컬 빌드 및 서빙 두 가지가 모두 가능합니다.

```
// For hot-reloading localhost
$ yarn dev
// Build & start at localhost
$ yarn build && yarn start
```

당연하게도 핫 리로드 방식을 선택하는 경우에는 메모리를 보다 많이 소모합니다.

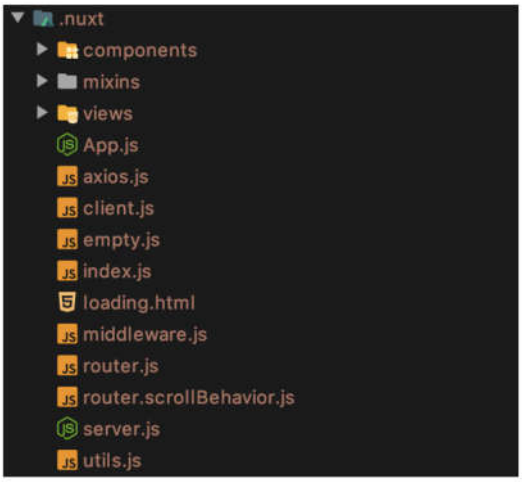




물론 계속 새로 빌드하고 start 하는 것에 비해서는 핫 리로드 방식을 사용해 개발하는 쪽이 훨씬 유리합니다.

빌드 결과물에도 차이가 있습니다. 기존 Vue.js에는 정적 배포가 가능한 파일이 생성되는 반면, Nuxt.js의 경우에는 서버 코드를 포함한 빌드 결과물이 나옵니다.

Nuxt.js

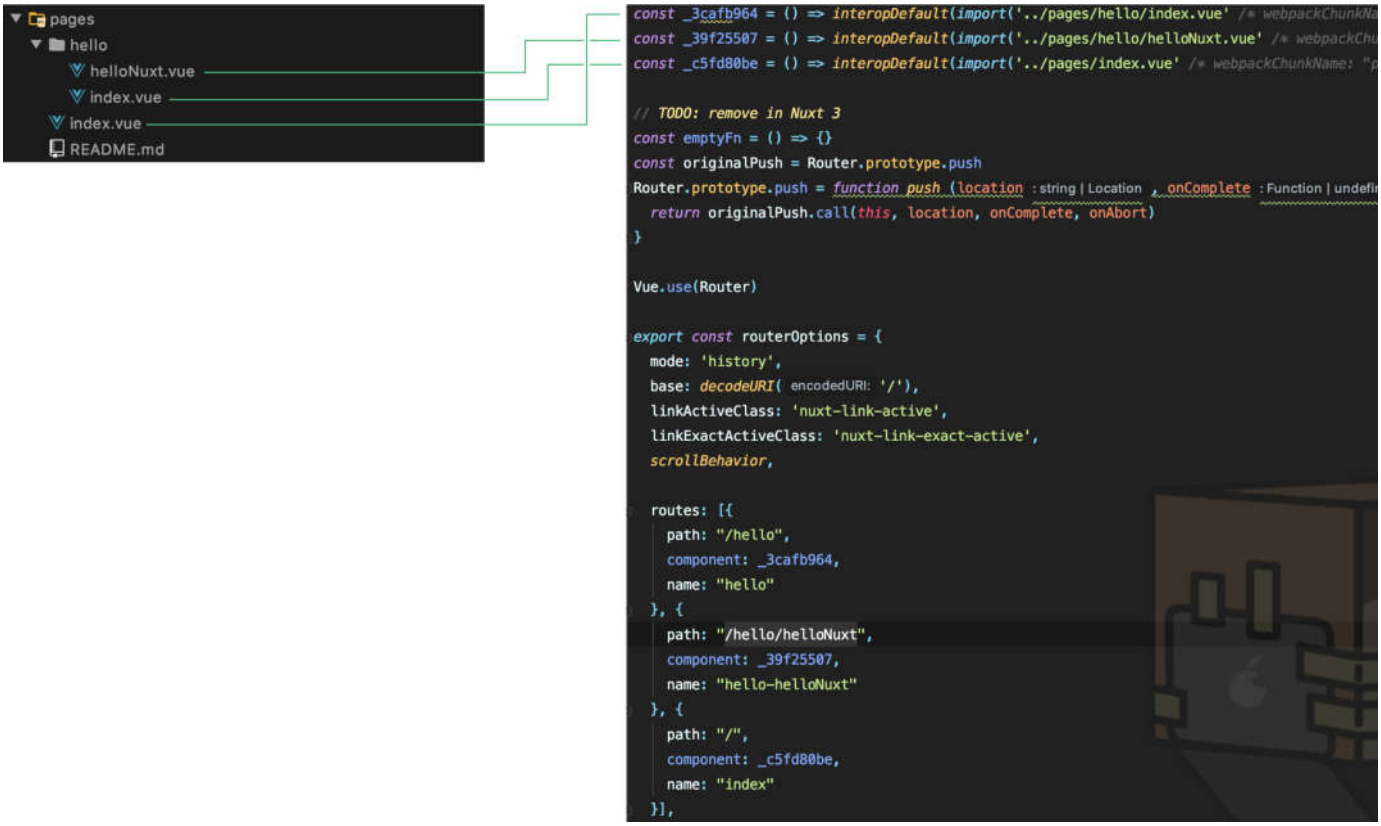


Vue.js

Vue.js 프로젝트의 경우 즉시 정적배포가 가능한 형태였기 때문에 S3 등을 이용해 바로 배포하는 것이 가능하지만, Nuxt는 EC2 인스턴스 등으로 별도 환경을 구축해 주어야 할 것으로 보입니다. 서버사이드에서 렌더링 연산을 수행하기 위해서는 당연한 결과로 보이기는 합니다. 이러한 구조적 특성으로 인해 Nuxt는 프론트엔드에서 백엔드까지의 작업을 묶어 하나의 인스턴스에 빌드하는 방식에 좀 더 효율적입니다.

라우팅 방식의 차이

가장 큰 변화중의 하나로써, 라우터를 꼽을 수 있을 것 같습니다. 기존의 Vue.js 프로젝트에서는 `router` 를 이용하여 직접 컴포넌트를 path에 바인딩하고 폴백 등을 설정할 수 있었습니다. Nuxt에서는 `views` 폴더 안에 있는 .vue 파일들을 폴더 내 구조에 맞게 자동으로 라우터에 추가해 줍니다.



폴백이나 동적 라우터 사용 등도 여전히 사용이 가능하지만, 기존에 직접 코드를 작성하던 방식과 달리 특수문자(주로 _)가 이름에 들어가는 디렉토리를 생성하는 등의 방식을 사용합니다. 자세한 방법은 공식 문서를 참고하면 좋을 것 같습니다.

메타 태그

SSR의 가장 중요한 목적 중의 하나인 SEO에 필요한 메타태그를 훨씬 편하게 달 수 있게 되었습니다. 기존 Vue 프로젝트에서 메타 태그를 달기 위해서는 vue-meta 등의 외부

라이브러리를 이용해야 했지만 nuxt에서는 별다른 추가작업 없이 메타 태그를 작성할 수 있습니다.

```
head () {
  return {
    title: this.message,
    meta: [
      {
        name: 'viewport',
        content: 'width=device-width,initial-scale=1.0,minimum-scale=1.0,maximum-scale=1.0,user-scalable=no'
      }
    ]
  }
}
```

*구체적인 작성 방법에 대해서는 직접 코드를 작성해 확인해 보지 않아 확신이 없습니다. 다만, 별도의 라이브러리를 직접 추가하지 않고도 meta 값을 적용할 수 있는 것을 확인하였습니다.

Layout

Layout이라는 폴더가 추가되어, Nuxt를 통해 라우팅되는 컴포넌트를 감싸는 레이아웃을 별도로 만들어줄 수 있습니다. 기존 프로젝트에서 App.vue나 nested router를 사용할 때 감싸는 것과 비슷한 방식입니다. `<router-view/>` 를 감싸는 대신 `<nuxt/>` 를 감싸주는 레이아웃을 생성합니다.

Layout은 여러 개를 생성할 수 있습니다. 특정 레이아웃을 사용할 컴포넌트 안에 layout 프로퍼티를 설정하여 layout을 사용할 수 있습니다.

```
<template>
<!-- Your template -->
</template>
<script>
export default {
  layout: 'yourLayout'
  // page component definitions
}
</script>
```

기타

이외에도 `middleware`, `plugin` 같은 폴더가 추가되었습니다. middleware의 경우 렌더링 이전 단계에 수행하는 함수 등을 선언하고 관리할 수 있으며 plugin의 경우 렌더링 이전 단계에서 외부 라이브러리(i.e. Axios)를 불러오는 등의 기능을 수행할 수 있습니다.

정리하며

렌더링 방식이 변화한 만큼 프로젝트의 구조도 다르고, 빌드의 결과물 및 동작 방식 등에 차이가 있습니다. 아직 프로젝트를 진행 해 보지 않아서 확실히 와닿지는 않지만, SSR과 CSR의 차이를 프레임워크 기본 설정에서부터 느낄 수 있는 것 같습니다.

Sunghwan Shin

디자인하는 프론트개발자

이전 포스트

Vue.js Mixin: 기능 캡슐화하기

1개의 댓글

댓글을 작성하세요

댓글 작성

katanazero86

3일 전

헷 놀러왔습니닷

답글 달기