Nuxt 프로젝트 구조

Evan

nuxt.config.json

nuxt.config.json은 nuxt.js의 설정 정보가 들어 있는 파일임

package.json

해당 프로젝트의 의존성 모듈과 각종 정보, 스크립트를 포함하는 파일. **npm init** 명령어를 통해 생성 가능함

assets (폴더)

SASS, **LESS**나 빌드가 필요한 Javascript 파일이 포함되는 디렉터리임

components (폴더)

컴포넌트들을 포함하는 디렉터리. 화면을 구성하고 있는 요소를 컴포넌트. 모듈, 부품의 개념으로 이해할 수 있음.

layouts (폴더)

기본 레이아웃을 포함하는 디렉터리. 우리가 만드는 웹 애플리케이션의 대부분은 상단 탭, 하단 탭, 좌/우측 탭은 모든 페이지에서 동일하게 적용됨. 이런 것들은 페이지마다 만드는 것이 아니라 레이아웃을 통해 한 번만 정의해서 사용할 수 있음

middleware (폴더)

레이아웃, 페이지가 렌더링되기 전에 실행되는 파일이 정의되는 곳

pagers (폴더)

우리가 보는 일반적인 vue 파일을 포함하는 곳. pagers도 컴포넌트의 일종임.

plugins (폴더)

vue로 만든 웹 애플리케이션이 생성되기 전에 실행시키고 싶은 js파일을 포함.
외부에서 설치한 모듈이나 직접 만든 모듈 모두 포함 가능함.

static (폴더)

이미지 파일이나 각종 CSS, Javascript 같은 정적 파일이 포함됨

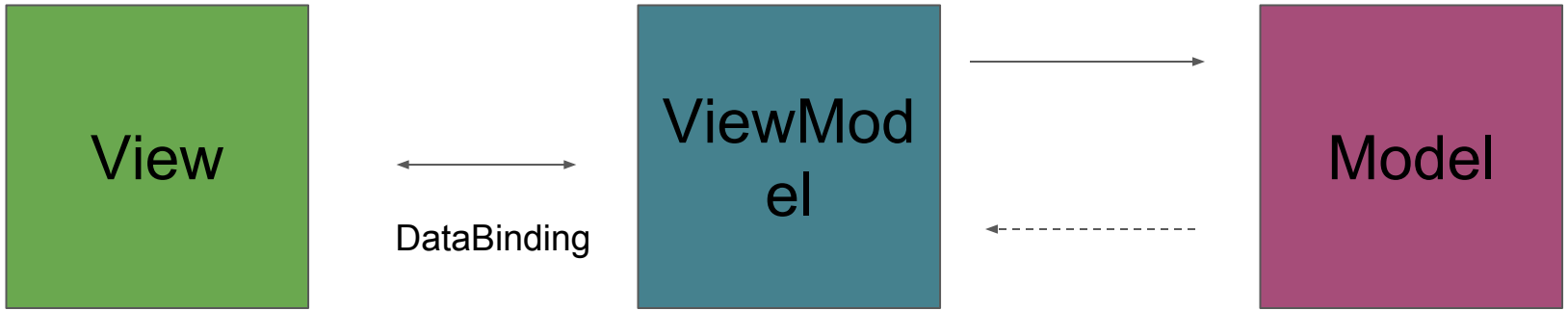
store (폴더)

상태(데이터) 관리를 도와주는 라이브러리인 vuex가 포함된 디렉토리

Vue 특징 : MVVM패턴 및 가상돔

Evan

MVVM패턴



Presentation and Presentation Logic

BusinessLogic and Data

화면 앞단의 화면 동작 관련 로직과 뒷단의 DB 데이터 처리 및 서버 로직을 분리하고, 뒷단에서 넘어온 데이터를 Model에 담아 View로 넘어주는 중간 지점이 ViewModel

웹페이지는 DOM과 Javascript의 조합으로 만들어지게 됨. DOM은 View 역할을 하고, Javascript가 Model 역할을 함. 뷰 모델이 없는 아키텍처는 getElementById 같은 돔 API를 직접 이용해서 모델과 뷰를 연결해야 함. 자바스크립트에서 컨트롤러 역할까지 하면서 코드양이 증가하는 단점이 생기게 됨

위와 같은 역할을 뷰모델이 대신 수행해 주는 것이 MVVM 모델.
뷰모델에 자바스크립트 객체와 돔을 연결해주면 뷰모델은 이 둘간의 동기화를 자동으로 처리함. 이것이 Vue.js가 하는 역할.

가상돔

DOM 요소가 많아지면 Javascript로 DOM을 핸들링 하는 일이 무거워짐. 그래서 DOM과 비슷한 구조로 Javascript를 만듬. 이것은 진짜 DOM과 달리 메모리에 올라가있는 것이기 때문에 비교적 매우 빠른 성능을 가짐. View는 Virtual DOM이 변경될 때마다 진짜 DOM과 비교해서 차이를 찾고 차이가 있는 부분의 DOM만 수정하는 동작을 함.

https://youtu.be/muc2ZF0QIO4