데이터 바인딩

* html의 어트리뷰트 값을 수정할 때, v-bind라는 디렉티브를 이용해서 데이터 값을 브라우저 화면에 자동으로 반영하는 것을 의미
* 변수값을 html 어트리뷰트에 연결함으로써 변수가 새로운 값으로 바뀔 때마다 html 엘리먼트에 실시간으로 렌더링

v-bind 디렉티브

* 콜론(:) 을 중심으로 왼쪽에는 v-bind를 기입하고 오른쪽에는 html 엘리먼트의 어트리뷰트를 선언해서 바인딩으로 만들어진 값을 대입하는 식
* 콜론(:) 만 사용하면 'v-bind’라는 명칭은 생략

<input v-bind:value="variableName”> // html, 어트리뷰트 값 생성

양방향 데이터 바인딩

* 입력과 동시에 데이터가 동기화되어 표시되는 기능을 수행

v-model 디렉티브

* html 어트리뷰트에 양뱡향 데이터 바인딩을 수행
* 콜론(:) 을 중심으로 왼쪽에는 v-model를 기입하고 오른쪽에는 html 엘리먼트의 어트리뷰트 이름을 작성

조건 판단 / v-if, v-else 디렉티브

* v-if 디렉티브

조건이 충족될때만 렌더링을 수행하도록 돕는 어트리뷰트

<p v-if:”bFlag == true”>앞면</p>

<p v-else>뒷면</p>

반복 / v-for 디렉티브

* v-for 디렉티브

html 안에서 반복문으로 배열값을 읽어서 목록의 항목과 인덱스 변수에 저장하도록 돕는 어트리뷰트

<li v-for="item in fruits”>

fruits 안의 fruitsName 항목을 하나씩 가져와 html로 렌더링

{{ item.fruitsName }}

이벤트 핸들러 실행과 v-on 디렉티브

* v-on 디렉티브

뷰에서 발생하는 이벤트를 지켜보면서 DOM 엘리먼트를 제어할 수있도록 도와주는 어트리뷰트

뷰 안에서 사용자가 키보드나 마우스를 제어하는 형식의 이벤트가 발생했을 때 이를 실행하는 함수와 바인딩 할 때 사용

< 버튼을 누르면 fnChangeTitle 메소드로 이벤트 핸들러 수행 >

<button v-on:click=”fnChangeTitle”>CLICK</buuton>

* 이벤트 핸들러

이벤트가 발생했을 때 실행되는 함수

* 메소드(Methods)

이벤트 핸들러 함수를 제작해야 할 때 미리 정의해 두고 사용

뷰 객체의 행위를 정의하는 용도, 즉 로직을 실행하는 함수가 필요할 때 이곳에 정의해 두고 사용

computed

* 머스태시를 이용해 html 엘리먼트 값이 어떻게 변경되는지 살펴보면서 필요한 연산 작업을 도움
* 머스태시

변수, 함수 명령어를 조합하여 사용할 수 있지만 내용이 복잡할 경우 머스태시에 간단히 바인딩하여 사용하기에는 부담

* computed 속성

머스태시에 넣는 자바스크립트 로직이 크고 복잡할 때 이곳에 위치

자바스크립트의 내용을 함수로 바꾸어 그 함수명을 실행하는 형식으로 머스태시에 사용

캐시하여 메모리에 저장하는 것이 큰 장점 > 반환값 동일할 경우 계산 없이 바로 전달, 빠르게 실행

/\* 변수안에서 함수 실행 시\*/

methods 속성

* 이벤트핸들러를 사용해 마우스 클릭과 같은 이벤트가 발생했을 때 실행되는 로직에 활용
* 메서드는 뷰 인스턴스에 포함해 사용하는 함수를 의미

차이점

* computed : 머스태시에 작성할 로직이 복잡하면 함수로 정의할 때

계산량이 많아 캐시가 필요할 때

* methods : 뷰의 이벤트핸들러 로직을 함수로 정의할 때

component html 엘리먼트 만들기

* 기본 엘리먼트 외의 자신만의 엘리먼트를 만들어 쓰는 모듈
* 미리 만들어둔 이름을 가지고 태그(<>) (</>) 에 적용해 사용

템플릿(template)

* 컴포넌트 안에서 화면에 표시될 부분을 처리하는 속성
* html, css, 자바스크립트를 적용하여 표시될 내용의 구조화 표현, 사용자와의 상호 작용 기능을 구현
* 백틱(`) 키를 사용하여 문자열을 선언하면 줄바꿈이 있어도 html 문서로 자동으로 인식

컴포넌트 속성(props)

* 컴포넌트에서 전달되는 어트리뷰트의 값으로써, 문자열이나 객체의 배열 형식을 사용

뷰 컴포넌트 장점

* html의 일반 엘리먼트처럼 사용할 수 있기 때문에 기능을 무한대로 확장할 수 있는 잠재력
* 다양한 주제별로 그에 맞는 엘리먼트를 설계하여 사용
* 컴팩트한 개발이 가능하므로 유지보수, 관점에서도 편리

컴포넌트 등록

* 컴포넌트는 html 엘리먼트와 동일하게 사용하며 어트리뷰트도 동일하게 사용

컴포넌트를 등록할 때 props: 속성을 선언

선언된 이름을 다른 속성에서 전달받아 매개변수처럼 사용

ex> <favorite-fruits fruit=”사과"></favorite-fruits>

상태 관리와 Vuex

* Veux란?

뷰의 코어 중 상태 관리를 위한 패턴 & 라이브러리

하나의 뷰 또는 복수의 화면 사이에서 여러 개의 컴포넌트를 사용하게 되면 각 컴포넌트 간에 상태값 전달이나 공유 (컴포넌트 간 공유할 수 있는 데이터 관리)

localStorage에 저장되지 않고 메모리에 저장되므로 새로고침 시 데이터 초기화

상태를 새로고침 하더라도 데이터를 유지하기 위해서는 별도의 라이브러리 필요

* Vuex의 작동 원리 이해하기

Stats, Mutations, Getters, Actions - 4개의 속성

actions :비동기 실행을 위해 준비

API의 fnDecrement() 함수

* 외부 API의 함수를 살행하는 것을 흉내 내기 위해서 API 모듈을 별도로 제작

네이게이션과 라우터

* 라우터란?

페이지간 이동을 위한 라이브러리

경로와 컴포넌트를 등록하면 싱글 페이지 앱에서 사용자가 클릭한 경로로 화면이 이동하도록 도와줌

주요기능

* 중첩된 route/view mapping
* 모듈화된, 컴포넌트 기반의 라우터 설정
* 라우터 파라미터, 쿼리, 와일드카드
* vue.js의 transition sysyem을 이용한 트랜지젼 효과
* 세밀한(Fine-grained) 네이게이션 컨트롤
* active CSS 클래스를 자동으로 추가해주는 링크
* html5 history 모드 또는 hash 모드
  + auto-fallback un IE9
  + vue route에서 default는 hash 모드
* 사용자 정의 가능한 스크롤 동작
* 라우터 사용

utl에 따른 컴포넌트가 화면에 그려지는(view) 영역

현재 라우터 위치의 컴포넌트 렌더링

<router-view></router-view>

router-link

* to 어트리뷰트

to 속성 값의 경로로 이동

v-bind와 함께 사용하여 동적으로 경로 생성

to="test/path” 처럼 붙이면 현재 url에 path 추가

to=”/test/detail” 처럼 붙이면 default url에 붙음 (대표적)

        <p>라우터 사용:

            <router-link to="/main">메인페이지로</router-link>

            <router-link to="/sub">서브페이지로</router-link>

* styling

router-link-exact-active 등 class를 통해 스타일 적용

router / index.js

import Vue from 'vue'

import VueRouter from 'vue-router'

import Read from '@/components/Read.vue'

import Create from '@/components/Create.vue'

import Detail from '@/components/Detail.vue'

import NotFound from '@/components/NotFound.vue'

const routes = [

  {

    path: '/',

    name: 'Read',

    component: Read

  },

  {

    path: '/create/:contentId?',

    name: 'Create',

    component: Create

  },

  {

    path: '/detail/:contentId',

    name: 'Detail',

    component: Detail

  },

  {

    path: '\*',

    name: 'NotFound',

    component: NotFound

  }

]

router 객체

vuetify init setting

<!DOCTYPE html>

<html>

<head>

  <link href="https://fonts.googleapis.com/css?family=Roboto:100,300,400,500,700,900|Material+Icons" rel="stylesheet">

  <link href="https://cdn.jsdelivr.net/npm/vuetify@1.x/dist/vuetify.min.css" rel="stylesheet">

  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no, minimal-ui">

</head>

<body>

  <div id="app">

    <v-app>

      <v-content>

        <v-container>Hello world</v-container>

      </v-content>

    </v-app>

  </div>

  <script src="https://cdn.jsdelivr.net/npm/vue@2.x/dist/vue.js"></script>

  <script src="https://cdn.jsdelivr.net/npm/vuetify@1.x/dist/vuetify.js"></script>

  <script>

    new Vue({ el: '#app' })

  </script>

</body>

</html>

    <!-- vuetify  CSS -->

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/vuetify@2.6.13/dist/vuetify.min.css">

    <!-- Vue CDN -->

    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>

    <!-- vuetify CDN-->

    <script src="https://cdn.jsdelivr.net/npm/vuetify@2.6.13/dist/vuetify.js"></script>

<script>

    new Vue({

        el: "#app",

        vuetify: new Vuetify()

    })

Vuetify

* 뷰티파이란?

뷰 자바스크립트 프레임워크에 머티리얼 디자인을 사용할 수 있는 컴포넌트 프레임 워크

구글의 머티리얼 디자인 스펙2를 충실하게 표현

현대 웹앱에 필요한 컴포넌트를 편리하게 사용

* 뷰티파이가 주목받는 4가지 특징

1. 구글 머티리얼 스펙의 충실한 지원
2. 80개 이상의 시맨틱 머티리얼 디자인 컴포넌트
3. 빠른속도
4. 쉬운 학습 곡선

* 기본 레이아웃 만들기

삼단 레이아웃 실습

* 모바일 UI의 가장 기본이 되는 툴바,본문, 바닥글의 삼단 레이아웃을 구성

핵심 내용

* 툴바 영역 : v-toolbar 엘리먼트 사용
* 본문 영역 : v-content 엘리먼트 사용
* 여백 자동 설정 : v-container 엘리먼트 사용
* 바닥글 영역 : v-footer 엘리먼트 사용

jsDelivr

@mdi/font 검색

[/npm/@mdi/font@7.1.96/**css**](mailto:/npm/@mdi/font@7.1.96/css)

[/npm/@mdi/font@7.1.96/**css/materialdesignicons.css**](https://cdn.jsdelivr.net/npm/@mdi/font@7.1.96/css/materialdesignicons.css)

copy to clipboard > URL

@vuetify/dist

<https://cdn.jsdelivr.net/npm/vuetify@2.6.13/dist/vuetify.css>

<https://cdn.jsdelivr.net/npm/vuetify@2.6.13/dist/vuetify.js>

앱바 스타일

* v-appbar 엘리먼트

앱바를 넣고 싶을 때 사용

* dark 어트리뷰트 지정

글자 색을 흰색으로 바꾸고 템와 글자의 명도를 자동으로 조정하여 가독성 향상

* fixed 어트리뷰트 지정

스크롤에 영향 받지 않고 툴바의위치를 고정

테마 컬러

{

primary: '#1976D2',

secondary: '#424242',

accent: '#82B1FF',

error: '#FF5252',

info: '#2196F3',

success: '#4CAF50',

warning: '#FFC107',

}

아이콘 버튼, 제목 등의 추가

* 메뉴 아이콘 버튼의 v-app-bar-nav-icon 엘리먼트를 사용
* 제목은 v-toolbar-title 엘리먼트를 사용해 값에 제목명 기입

아이콘과 텍스트 오른쪽 정렬

* v-spacer 엘리먼트를 추가하여 구분
* v-spacer 엘리먼트 사용 전에는 왼쪽 정렬로 나타남

아이콘 변경

* 구글 폰트사이트 / icons = Material icons에서 사용할 아이콘 선택
* <span classs=”아이콘 클래스명”> 사용할 아이콘 이름</span>  
  MDI 아이콘 변경
* https://materialdesignicons.com/

사용할 아이콘에 마우스 오버

<v-icon>mid-아이콘 이름<v-icon>

v-main

* 본문의 영역을 정함
* 사용 가능한 모든 영역을 본문 영역 지정

v-container

* v-container 안에 모든 엘리먼트에게 화면 크기에 맞는 적절한 여백을 자동으로 부여
* v-main을 먼저 사용하고 그 다음에 v-container를 레이아웃에 맞춰서 사용

카드 UI

(color="white(배경색) white--text(--텍스트)“

* 카드 영역 안에 이미지

v-img 엘리먼트 사용

src 어트리뷰트를 사용 이미지 경로 지정

aspect-ratio 어트리뷰트로 이미지의 가로 세로 비율 지정

* 카드 안에 제목과 본문을 넣는 방법

v-card-text 엘리먼트 사용

* v-card-text로 글자로 디자인 하고 싶은 영역 지정

타이포그래피의 크기 :title 로 설정

제모그이 여백 :mb-2으로 bottom 여백을 2(8px) 로 설정

* 버튼 스타일

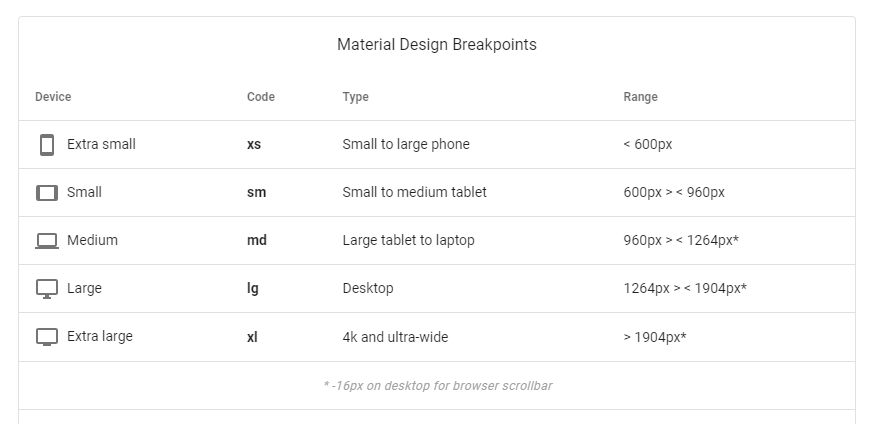
<v-btn>버튼</v-btn>

그리드 기본 원리

* 그리드는 화면 레이아웃 작업에서 가장 핵심

반응형 웹 디자인을 하려면 그리드 사용법은 필수

v-row, v-col 엘리먼트를 통해 그리드 기능을 충실히 제공



CLI

PWA

npm install -g serve (serve) 설치 / npm run build 빌드 / serve dist (실행)

Hosting

npm istall -g firebase-tools 설치 / firebase login (로그인) / 계정 연동(구글)

뷰 파이어

뷰와 파이어베이스를 연결하여 손쉬운 코드만으로 실시간 데이터 기반의 동적인 웹 UI를 빠르게 렌더링할 수 있도록 해주는 최적의 파이어베이스 데이터바인딩 라이브러리

파이어베이스를 이용해서 최신 웹 애플리케이션의 데이터 관리의 CRUD(create, read, update, delete)를 쉽고 빠르게 작업하도록 도와주는 솔루션