Vex Logo VEX

Introduction

Vex is a Creative, Responsive Material Design Admin Template built with Angular 8+ and the Angular-CLI. It extends the Material Design components built by the Angular team and it offers you everything you need to get started with your next CRM, CMS, Project Management, or other projects.

Vex has **no dependency on jQuery or similiar libraries**, Angular's functionality is completely used.

Support is available through E-Mail (themeforest@visurel.com). If you purchased the theme and love it, consider giving it a 5-star rating here on ThemeForest. It really helps pushing out more updates and adding more great features.

Getting Started

> In this section you will find the basic folder structure and everything you need to get the template up and running the first time to start developing.

Folder Structure

Name	Description
`angular.json`	Used for configuration of project specific settings. You can add external styles and scripts, change the output folder, add assets, add environment files and more.
`e2e`	Will be used for end-to-end tests to ensure functionality for users before deploying.
`node_modules`	All external modules used are here. Do not mess with this folder, as it is auto-generated by using `npm install`.
`package.json`	Contains all dependencies used for production and development.
`src`	Contains all Angular Typescript code, assets and basically everything the end user will have access to.
`tslint.json`	Angular-CLI includes an automatic Typescript-Linter, which can be configured with this file.

Installation

Angular-CLI allows you to create a new App in a matter of seconds and provides an awesome way to generate scaffolds for basically all Angular-Components. You can take a look at what commands are available here.

In this section, we are going to install Angular-CLI and it's prerequisites and then generate our first project with a few various components.

Prerequisites

> Before we can install Angular-CLI we will have to get a few thing set up first. To run Angular-CLI we will need to install this prerequisite first: * **NodeJS** v10 or newer

A detailed instruction on how to install NodeJS is available here.

Installing Angular-CLI

Installing Angular-CLI is as simple as running this simple command:

`npm install -g @angular/cli@latest`
or `sudo npm install -g @angular/cli@latest`

and the package manager `npm` will do the rest.

Install Vex Dependencies

Navigate to the Vex folder and run `npm install` to install all dependencies required by Vex.

Start Development Server

Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`. The app will automatically reload if you change any of the source files.

Build for Production

If you want to create a build for a production environment you can simply run `npm run build` or `ng build --prod` and you will get static HTML and JS files in the `/dist` folder ready to be uploaded to any server.

Customization

In this section you are going to learn how to customize Vex to be exactly the way you want it to be.

Configuration

Configuring the Vex Layout to your needs is as easy as setting a simple object with the values you want, here's an example configuration which handles everything for you:

You can change these values even at run-time and the page will adjust to the changes.

```
{
  id: 'vex-default',
  layout: 'horizontal',
  boxed: false,
  sidenav: {
    layout: 'expanded'
  },
  toolbar: {
    fixed: true
  },
  footer: {
    visible: true,
    fixed: true
  }
}
```

Changing Styling and CSS Variables

> Most of the styling (padding, colors, fonts, font-weights, ...) is build very modular and easily customizable. If you want to change any specific style globally you can just change the CSS Variable inside the `style.scss` file and it will update every single section of the page.

Here are just a few example variables, almost everything is done through variables:

```
 --container-width: 1200px;
--padding: 24px;
--font: Roboto, "Helvetica Neue", sans-serif;
--font-weight-semi-bold: 500;
--text-color: #{$dark-primary-text};
--sidenav-width: 280px;
--sidenav-background: #{$sidenav-background};
--sidenav-color: white;
// and much more...
```

Using Custom Colors for the Primary/Secondary/Warn Palettes

> Inside the `style.scss` find the section below. The values behind the names are just `rgb` colors, so `--color-primary-500` would be `rgb(92, 119, 255)` if you want to customize these colors you can do that easily by changing the colors for these variables.

The numbers behind the `--color-primary-` are called `hue`. Usually the `500-hue` is a strong colorful color e.g. a strong blue. The `400, 300, 200, 100, 50` are lighter versions of the base color and the `600, 700, 800, 900` are darker versions of the base color.

For each `primary color` there is a `contrast` variable. The contrast variable is used when you want to display something on the `primary color`.

Example: You use `--color-primary-900` as your background color (and the 900 hue should be pretty dark), and want to display text on that background, then you want the `--color-primary-contrast-900` to be `white` and use that as the text color.

This works the same for all the other colors too.

```
// Colors
  --color-primary-50: rgb(236, 239, 255);
  --color-primary-100: rgb(206, 215, 255);
  --color-primary-200: rgb(174, 188, 255);
  --color-primary-300: rgb(142, 161, 255);
  --color-primary-400: rgb(117, 140, 255);
  --color-primary-500: rgb(92, 119, 255);
  --color-primary-600: rgb(85, 112, 255);
  --color-primary-700: rgb(75, 101, 255);
  --color-primary-800: rgb(65, 91, 255);
  --color-primary-900: rgb(48, 72, 255);
  --color-primary-A100: rgb(128, 216, 255);
  --color-primary-A200: rgb(64, 196, 255);
  --color-primary-A400: rgb(219, 223, 255);
  --color-primary-A700: rgb(194, 200, 255);
  --color-primary-contrast-50: #{$dark-primary-text};
  --color-primary-contrast-100: #{$dark-primary-text};
  --color-primary-contrast-200: #{$dark-primary-text};
  --color-primary-contrast-300: #{$dark-primary-text};
  --color-primary-contrast-400: #{$dark-primary-text};
  --color-primary-contrast-500: #{$light-primary-text};
  --color-primary-contrast-600: #{$light-primary-text};
  --color-primary-contrast-700: #{$light-primary-text};
  --color-primary-contrast-800: #{$light-primary-text};
  --color-primary-contrast-900: #{$light-primary-text};
  --color-primary-contrast-A100: #{$dark-primary-text};
  --color-primary-contrast-A200: #{$dark-primary-text};
  --color-primary-contrast-A400: #{$dark-primary-text};
  --color-primary-contrast-A700: #{$dark-primary-text};
```

Adding Menu Items

Our Menu is generated completely dynamic and is therefore very easy to customize and use, the simplest way is to simply inject the NavigationService and just override the `items` instance variable. Navigation Items have the `NavigationItem` type which you can use for autocompletion and type-safety. (Generally, all our components have typings available!)

You can add 3 different types:**Subheading, Link, Dropdown**

Here's an example inside the AppComponent:

```
@Component({
  selector: 'vex-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'vex';

  constructor(private navigationService: NavigationService) {
    this.navigationService.items = [
      {
        label: 'Subheading'
      },
      {
        label: 'Item with a Link',
        route: '/my-link',
        icon: icAssigment
      },
      {
        label: 'Dropdown Item',
        icon: icAdd,
        children: [
          {
            label: 'Item with Link inside Dropdown',
            route: '/custom-link'
          }
        ]
      }
    ];
  }
}
```

Generating the first `Component`

> Now that we have installed all prerequisites it's time to start developing our app. Angular-CLI offers a lot of assistance and allows you to generate basically all Angular-Components there are. *(In a smart way!)*

To generate our first `component` we simply open up a terminal and navigate in our Angular-App. Now we simply run `ng g component client` and we get a new component in `/src/app/client` with the following files:

- `client.component.ts`
- `client.component.html`
- `client.component.css`
- `client.component.spec.ts`

The files `client.component.ts` and `client.component.spec.ts` contain the most code, the other files only contain placeholders.

client.component.ts

```
 import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-client',
  templateUrl: 'client.component.html',
  styleUrls: ['client.component.css']
})
export class ClientComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

By executing this short command, we just saved ourselves a lot of time creating all these `Component` files and boilerplate code.

Syntax for all commands are available here.

Generating a Service in a specific folder

Now we have our `component`, but what if we want to share some data between components and need to create a `service` to manage this all. Well, we probably would want the service to be in the correct folder, either right in the components folder or in the `shared` folder in `/src/app/`.

Either way, with Angular-CLI we can generate in any folder, wherever we want. Simply use the path *(relative to `/src/app/`)* and use it as the name of the generated component.

`ng g service shared/client`
or `ng g service client/shared/client`

Or anything you need. Afterward we will find two generated files in our specified folder `client.service.ts` and `client.service.spec.ts`.

client.service.ts

```
 import { Injectable } from '@angular/core';

@Injectable()
export class ClientService {

  constructor() { }

}
```

Running unit tests

Run `ng test` to execute the unit tests via Karma.

Running end-to-end tests

Run `ng e2e` to execute the end-to-end tests via Protractor.

Further help

> If you have any specific questions about the template, you can contact us anytime on our support email (themeforest@visurel.com) and we'll help you with any issues you may encounter. If you encounter any bugs or issues, feel free to send them over to us with a detailed description and we'll fix the issue ASAP.

Thanks for reading, if you have any pre-sale questions or just anything, don't hesitate to contact us! :)