# Introduction

## Google's Material Design

All libraries and custom made components are following **Google's Material Design Specifications.**

## React

**React** is the core of our template. Fuse React is NOT a traditional admin template, it's an React app. If you don't know what React is or don't know how to use it, we strongly recommend checking the React before start doing anything with Fuse.

## Material-UI

**Material-UI** is a react ui library that implement Google's Material Design specification.

## Create React App (CLI)

**Create React App** is a tool built by developers at Facebook to help you build React applications. It saves you from time-consuming setup and configuration.

# Prerequisites

This section will give you some information about what tools you will need. You can skip to the Installation section to start installing the template. We already mentioned all the prerequisites and how to install them in the Installation section.

## Node.js

To install and use Fuse React, you will need **Node.js** installed to your computer. We won't get into too much detail about Node.js as it's out of the scope of this documentation. Also you won't need to actually use Node.js, it's only required for the development process.

## Git

To be able to install and use Fuse, you will also need **Git** installed to your computer. Git is required for npm/yarn to work correctly.

## Yarn - Package Manager

Fuse React uses **yarn** package manager to install and manage 3rd party components and libraries.

# Installation

1. Unzip the zip file that you have downloaded from Themeforest. Inside the zip file, you will find the Skeleton Project **(Fuse-react-x.x.x-skeleton.zip)** along with the Demo Project **(Fuse-react-x.x.x-demo.zip)**, PSD designs and a readme file.

2. Extract the contents of the zip file **(Fuse-react-x.x.x-skeleton.zip)** into a folder that you will work within. For this documentation, we will refer that as "your work folder".

3. Open your favorite console application (Terminal, Command Prompt etc.), navigate into your work folder, run the following command and wait for it to finish:

   ```
   yarn
   ```

   This command will install all the required Node.js modules into the node_modules directory inside your work folder. And now, you are ready to run the Fuse React for the first time.

# Working with Fuse

While still in your work folder, run the following command in the console application:

And that's it. Angular CLI will take care everything and start the Fuse.

You can check out your console application to get further information about the server. By default, it will run on **http://localhost:300** but it might change depending on your setup.

```
yarn start
```

# Production

```
yarn run build
```

compiles the application into `/build` directory

# Default Settings

You can set default settings of your app at `fuse-configs/FuseSettings.js`

```
const fuseSettingsConfig = {
    layout          : {
        navbar      : 'left', // 'right', 'left', 'top', 'none'
        navbarFolded: false, // true, false
        toolbar     : 'below', // 'above', 'below', 'none'
        footer      : 'below', // 'above', 'below', 'none'
        mode        : 'fullwidth' // 'boxed', 'fullwidth'
    },
    customScrollbars: true,
    theme           : 'default',
    navbarTheme     : 'currentThemeDark',
    toolbarTheme    : 'default',
    footerTheme     : 'currentThemeDark'
};

export default fuseSettingsConfig;
```

# Fuse Routing

Fuse React routing system based on **react-router** and its package **react-router-config**

For the modular approach and route based Fuse settings, we are using config files and generate routes from that files.

For example, have a look at `MailAppConfig.js` file under `src/main/content/apps/mail`. Whenever we navigate to `/apps/mail`, the navigation sidebar position will be changed to right.

```
import MailApp from './MailApp';
import React from 'react';
import {Redirect} from 'react-router-dom';

export const MailAppConfig = {
    settings: {
         layout          : {
            navbar              : 'right', // 'right', 'left', 'top', 'none'
            navbarFolded        :  false, // true, false
            toolbar             : 'below', // 'above', 'below', 'none'
            footer              : 'below', // 'above', 'below', 'none'
            mode                : 'fullwidth' // 'boxed', 'fullwidth'
         },
         customScrollbars: true,
         theme           : 'default'
    },
    routes  : [
        {
            path     : '/apps/mail/label/:labelHandle/:mailId?',
            component: MailApp
        },
        {
            path     : '/apps/mail/filter/:filterHandle/:mailId?',
            component: MailApp
        },
        {
            path     : '/apps/mail/:folderHandle/:mailId?',
            component: MailApp
        },
        {
            path     : '/apps/mail',
            component: () => <Redirect to="/apps/mail/inbox"/>
        }
    ]
};
```

# Fuse Routing

Then we import and generate routes from that file in `fuse-configs/fuseRoutes`

```
import {appsRoutes} from 'main/content/apps/mail/MailAppConfig.js';
import {FuseUtils} from '@fuse/index';
import {Redirect} from 'react-router-dom';
import React from 'react';

const routeConfigs = [
    MailAppConfig
];

export const routes = [
    ...FuseUtils.generateRoutesFromConfigs(routeConfigs),
    {
        path     : '/',
        component: () => <Redirect to="/pages/errors/error-404"/>
    }
];
```

# FuseAuth

`FuseAuth` is authorization component of the Fuse React. It allows to block routes based on user roles. It should wraps the FuseTheme component.

```
<FuseAuth routes={routes}>
     <FuseTheme>
         <FuseLayout
             routes={routes}
             toolbar={
                 <MainToolbar/>
             }
             navbarHeader={
                 <MainNavbarHeader/>
             }
             navbarContent={
                 <MainNavbarContent/>
             }
             footer={
                 <MainFooter/>
             }
         />
         <FuseSettings/>
         <QuickPanel/>
     </FuseTheme>
 </FuseAuth>
```

## Configuration

You can define authorization roles in route config files.

```
export const AdminRoleExampleConfig = {
    settings: {
        layout: {}
    },
    auth    : authRoles.admin,//['admin']
    routes  : [
        {
            path     : '/auth/admin-role-example',
            component: AdminRoleExample
        }
    ]
};
```

You can also hide navigation item/group/collapse by adding auth property in `fuse-configs/fuseNavigationConfig.js` .

```
{
  'id'   : 'only-admin-navigation-item',
  'title': 'Nav item only for Admin',
  'type' : 'item',
  'auth' : authRoles.admin,
  'url'  : '/auth/admin-role-example',
  'icon' : 'verified_user'
},
```

# FuseLayout

`FuseLayout` is the main layout component of the Fuse React.

The component has layout areas to easily enter the contents of the app.

Routes should be assigned. It makes changing layout with route configuration possible.

```
<FuseLayout
    routes={routes}
    toolbar={
        <MainToolbar/>
    }
    navbarHeader={
        <MainNavbarHeader/>
    }
    navbarContent={
        <MainNavbarContent/>
    }
    footer={
        <MainFooter/>
    }
/>
```

# FuseNavigation

`FuseNavigation` is a custom built Fuse component allows you to create a multi-level collapsable navigation.

## Usage

```
<FuseNavigation navigation={navigation}/>
```

### [navigation]

`FuseNavigation` uses a array to populate the entire navigation. It supports four different navigation items; Group, Collapse, Item. and Divider. These items can be mixed and matched to create unique and complex navigation layouts.

#### Group

```
{
   'id'       : 'applications',
   'title'    : 'Applications',
   'type'     : 'group',
   'icon'     : 'apps',
   'children' : [
        {
           'id'       : 'calendar',
           'title'    : 'Calendar',
           'type'     : 'item',
           'icon'     : 'today',
           'url'      : '/apps/calendar'
       }
   ]
}
```

#### Collapse

```
{
   'id'       : 'dashboards',
   'title'    : 'Dashboards',
   'type'     : 'collapse',
   'icon'     : 'dashboard',
   'children' : [
       {
           'id'   : 'project',
           'title': 'Project',
           'type' : 'item',
           'url'  : '/apps/dashboards/project'
       }
   ]
 }
```

## Item

```
{
    'id'   : 'project',
    'title': 'Project',
    'type' : 'item',
    'url'  : '/apps/dashboards/project'
}
```

## Divider

```
{
    'id'   : 'project',
    'title': 'Project',
    'type' : 'item',
    'url'  : '/apps/dashboards/project'
}
{
    'type': 'divider
},
{
    'id'   : 'project',
    'title': 'Project',
    'type' : 'item',
    'url'  : '/apps/dashboards/project'
}
```

# FusePageCarded

`FusePageCarded` is the carded page layout component of the Fuse React.

The component has layout areas to easily enter the contents of the app.

You can override the class names injected by the classes property

```
<FusePageCarded
    classes={{
        root: classes.layoutRoot
    }}
    header={
        Header
    }
    contentToolbar={
        Content Toolbar
    }
    content={
        Content
    }
    leftSidebarHeader={
        Left Sidebar Header
    }
    leftSidebarContent={
        Left Sidebar Content
    }
    rightSidebarHeader={
        Right Sidebar Header
    }
    rightSidebarContent={
        Right Sidebar Content
    }
    onRef={instance => {
        this.pageLayout = instance;
    }}
    singleScroll
/>
```

# Demos

- Full Width
- Full Width Tabbed
- Full Width 2
- Full Width 2 Tabbed
- Left Sidebar
- Left Sidebar Tabbed
- Left Sidebar 2
- Left Sidebar 2 Tabbed
- Right Sidebar
- Right Sidebar Tabbed
- Right Sidebar 2
- Right Sidebar 2 Tabbed

# FusePageSimple

`FusePageSimple` is the simple page layout component of the Fuse React.

The component has layout areas to easily enter the contents of the app.

You can override the class names injected by the classes property

```
<FusePageSimple
     classes={{
         root: classes.layoutRoot
     }}
     header={
         Header
     }
     contentToolbar={
         Content Toolbar
     }
     content={
         Content
     }
     leftSidebarHeader={
         Left Sidebar Header
     }
     leftSidebarContent={
         Left Sidebar Content
     }
     rightSidebarHeader={
         Right Sidebar Header
     }
     rightSidebarContent={
         Right Sidebar Content
     }
     onRef={instance => {
         this.pageLayout = instance;
     }}
     singleScroll
     sidebarInner
 />
```

# Demos

- Full Width
- Left Sidebar
- Left Sidebar 2
- Left Sidebar 3
- Right Sidebar
- Right Sidebar 2
- Right Sidebar 3
- Tabbed

# FuseScrollbars

`FuseScrollbars` is a simple **perfect-scrollbar** component for react.

It can be disabled globally by Fuse Settings.

```
<FuseScrollbars className={classes.content}>
    Content
</FuseScrollbars>
```

## Props

```
FuseScrollbars.defaultProps = {
    className    : '',
    enable       : true,
    option       : undefined,
    containerRef : () => {
    },
    onScrollY    : undefined,
    onScrollX    : undefined,
    onScrollUp   : undefined,
    onScrollDown : undefined,
    onScrollLeft : undefined,
    onScrollRight: undefined,
    onYReachStart: undefined,
    onYReachEnd  : undefined,
    onXReachStart: undefined,
    onXReachEnd  : undefined
};
```

# FuseCountdown

`FuseCountdown` is a custom built Fuse component allows you to create countdowns.

## Usage

```
<FuseCountdown endDate="2019-07-28" className="my-48"/>
```

## Preview

469 days 6 hours 30 minutes 13 seconds

## Demos

- Coming Soon

# FuseHighlight

`FuseHighlight` is a custom built Fuse component allows to show syntax highlighted codes with **PrismJS**.

## Usage

```
<FuseHighlight component="pre" className="language-html">
  <div className="title">
       <span>Example Title</span>
   </div>
</FuseHighlight>
```

## Preview

```
<div className="title">
    <span>Example Title</span>
</div>
```