
Table of Contents

Introduction	1.1
Technology We Use	1.2
Installation	1.3
Overview	1.4
Non Monorepo Structure Overview	1.5
Monorepo Packages Overview	1.6
isomorphic-cra	1.6.1
Isomorphic-next	1.6.2
isomorphic-boilerplate	1.6.3
isomorphic-servers	1.6.4
Isomorphic-boilerplate-single	1.6.5
Isomorphic-boilerplate-graphql	1.6.6
Getting Started	1.7
Quick Tips	1.7.1
Styling	1.7.2
AsyncComponent	1.8
Components	1.9
Shop(Algolia)	1.9.1
Authentication	1.9.2
Firebase	1.9.2.1
Auth0	1.9.2.2
UI Elements	1.9.3
widgets	1.9.4
Map	1.9.5
leaflet Map	1.9.6
Isotope	1.9.7
Pages	1.9.8

Sidebar & Routing	1.9.9
Forms	1.9.10
Authentication	1.9.11
Contact	1.9.12
Calendar	1.9.13
Ant Table	1.9.14
leaflet Map	1.9.15
React Trend Chart	1.9.16
Recharts	1.9.17
ReactVIs Chart	1.9.18
react-chart-2	1.9.19
Uppy	1.9.20
chat-firebase	1.9.21
RTL (CSS) transformation	1.10
Multi Language Support	1.11
Express JWT Implementation	1.12
Deployment	1.13
Spark JWT Implementation	1.14
Flask JWT	1.15
Django JWT	1.16

Isomorphic

React Redux Admin Dashboard.

Demo : <https://isomorphic.redq.io/>

Credits:

- Create React App
- React
- Redux
- Redux-Saga
- React Router 4
- Webpack 3
- ImmutableJS
- [Ant Design](#)
- Google Map
- React Big Calendar
- React Flip Move
- React Google Charts
- Recharts
- React Vis
- React Chart 2
- React Trend
- Echart
- React Grid Layout
- Authentication Firebase
- Authentication Auth0
- Algolia Search

Monorepo

The idea behind a monorepo is to store all code in a single version control system (VCS) repository. The alternative, of course, is to store code split into many different VCS repositories, usually on a service/application/library basis.

Welcome to Isomorphic new version Built on mono-repo structure. Check the below link if you want to learn more about yarn workspaces.

1. [Introducing-workspaces/](#)
2. [Workspaces.](#)

Why we use monorepo

1. To store all code in a single version control system (VCS) repository.
2. **Easier collaboration and code sharing**
3. Code refactors are easy / atomic commits

NB: A Non Monorepo dashboard version is also available.

Technology Credits

For Isomorphic [Dashboard]

- Create React App
- React
- Redux
- Redux-Saga
- React Router 5
- [Ant Design](#)
- Google Map
- React Big Calendar
- React Flip Move
- React Google Charts

- Recharts
- React Vis
- React Chart 2
- React Trend
- Echart
- React Grid Layout
- Firebase CRUD
- Authentication Firebase
- Authentication Auth0
- Algolia Search etc

adsf;alskdjf;la

Installation

- Install Node JS
- Install yarn
- yarn
- yarn start:iso-cra

Installing Node & Yarn:

To work with `Isomorphic` the first thing you need is to have [Node](#) install on your system. To make sure you have already Node js installed on your system you may follow the below instructions :-

As Node will make sure you have node and npm commands are available via command line, just run the below command on your terminal

```
node -v
```

Installing YARN:

You will need to Install [Yarn](#) for the Fast, Reliable, and Secure Dependency Management. Before you start using [Yarn](#), you'll first need to install it on your system. And to make sure it running on your system with latest version run the below command

```
yarn -version
```

or

```
yarn -v
```

On successful installation, it will print out the version.

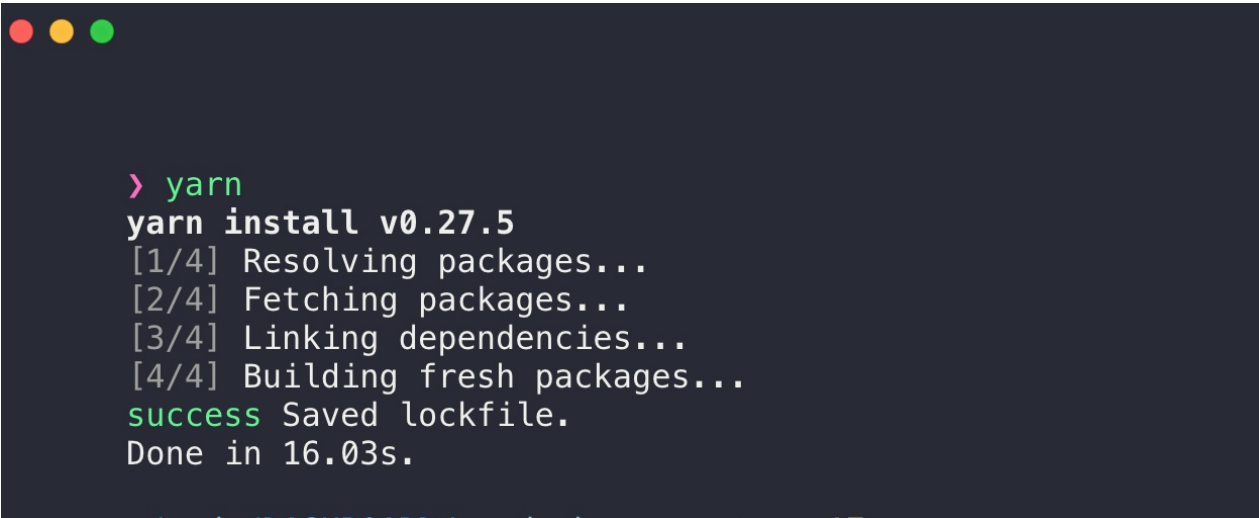
```
> yarn -v
yarn install v0.27.5
[1/4] Resolving packages...
success Already up-to-date.
Done in 0.06s.
```

Installing Packages & Dependencies:

After Installing Yarn, now open the `Isomorphic` app in your terminal. Now at your terminal In the root directory of `Isomorphic` app just run

```
yarn
```

it will download all the necessary packages and dependencies in the `node_modules` folder.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The terminal shows the command 'yarn' being executed, followed by 'yarn install v0.27.5'. The progress shows four steps: [1/4] Resolving packages..., [2/4] Fetching packages..., [3/4] Linking dependencies..., and [4/4] Building fresh packages... The output ends with 'success Saved lockfile.' and 'Done in 16.03s.'

```
> yarn
yarn install v0.27.5
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
Done in 16.03s.
```

yarn start:iso-cra

Now to start the `Isomorphic` app all you need to do is to run the below command in you terminal root directory of the `Isomorphic` app.

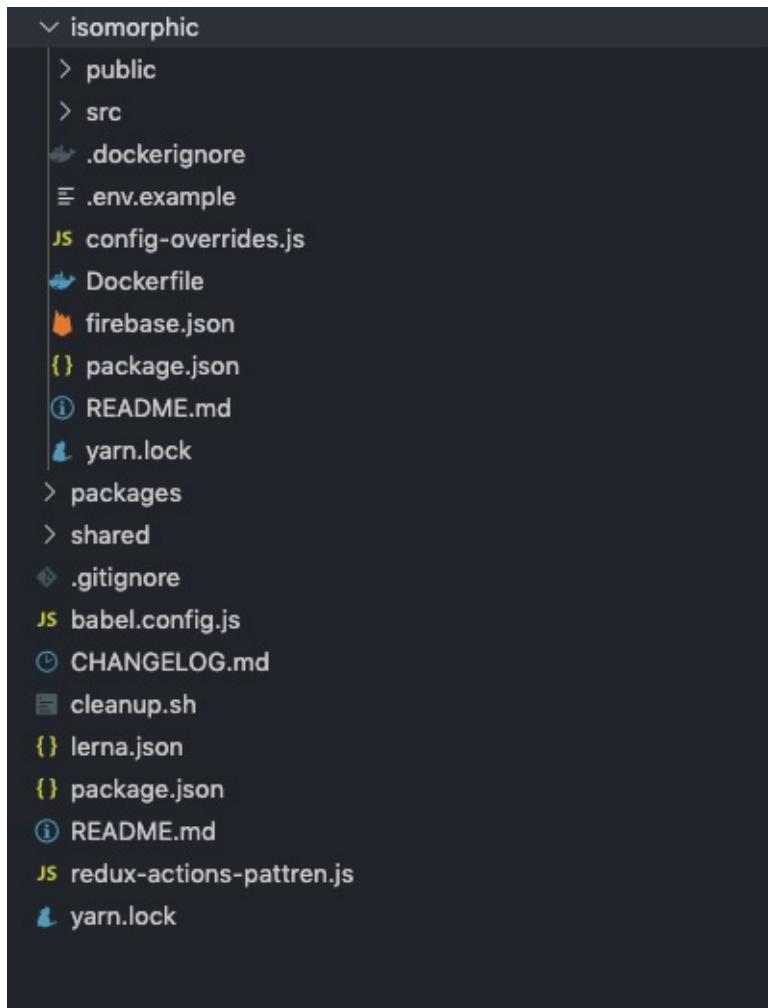
```
yarn start:iso-cra
```

after the compiled process completed successfully, it will show the below success commands & redirect to the `http://localhost:3002/` of your browser where you will find the login screen of the `Isomorphic` app.

NOTE: The above installation guide uses our monorepo structure

Non Monorepo:

If you want to use a non monorepo dashboard then you can directly go to the isomorphic directory



and run the below command,

1. yarn
2. yarn start
3. yarn build

the non monorepo dashboard is bootstrapped with [Create React App](#)

Overview

Isomorphic Comes With multiple boilerplate and configurations for different use cases.

1. isomorphic (build with custom CRA)
2. isomorphic-next (SSR supported by next.js)
3. isomorphic-boilerplate (if Anyone want to use with monorepo and custom CRA)
4. isomorphic-boilerplate-single (If Anyone want to use isomorphic without monorepo)
5. isomorphic-boilerplate-graphql (CRA with GraphQL server and client)
6. isomorphic-servers (example: of different backend integration)

How we utilize monorepo structure

By using monorepo we simplified `import`

```
import Component from '../../../components/Component'
import ContainerComponent from '../../../containers/ContainerComponent'

to

import Component from '@iso/components/Component'
import ContainerComponent from '@iso/containers/ContainerComponent'
```

Reuse `components` , `containers` , `redux` , `assets` , `config` between all our `packages`. By moving them in `shared` folder and setup with monorepo. to import things from there you just prefix with `@iso/folderName`

example:

```
shared/assets ==> @iso/assets
```

```
shared/components ==> @iso/components/required_component_path
```

We also move common things like library and ui within `shared/UI` and `shared/library`

To use them you just need to import things from

```
shared/library ==> @iso/lib
```

```
shared/UI ==> @iso/ui
```

If you don't want to use monorepo in next version

Note: React dashboard version non monorepo is already available in the isomorphic directory check the next section for more information

we didn't provide any without monorepo support for isomorphic-next but you can easily make your own by adding some configurations within `next.config.js` file like:

```
config.resolve.alias = {
  ...config.resolve.alias,
  // Will make webpack look for these modules in parent directories
  '@iso/assets': require.resolve('shared/assets'),
  '@iso/config': require.resolve('shared/config'),
  '@iso/components': require.resolve('shared/components'),
  '@iso/containers': require.resolve('shared/containers'),
  '@iso/lib': require.resolve('shared/library'),
  '@iso/ui': require.resolve('shared/UI'),
};
```

Also

you need to move all the folders from `shared` within your project folder .

Non Monorepo Structure:

NOTE: please note that currently we are providing non monorepo only for the Create React App Dashboard and not for other packages.

After downloading Isomorphic from themeforest, you will find a zip. If you extract this, there is a Isomorphic folder for you which contains all of our codes. inside isomorphic directory there is another isomorphic directory which is basically a non monorepo version and bootstrapped with Create React App.

The folder structure of Isomorphic is following like that.



Build: All the Build files are available on this folder.



config: It contains all the files regarding webpack. All the build configurations (develop, production) is inside that folder.

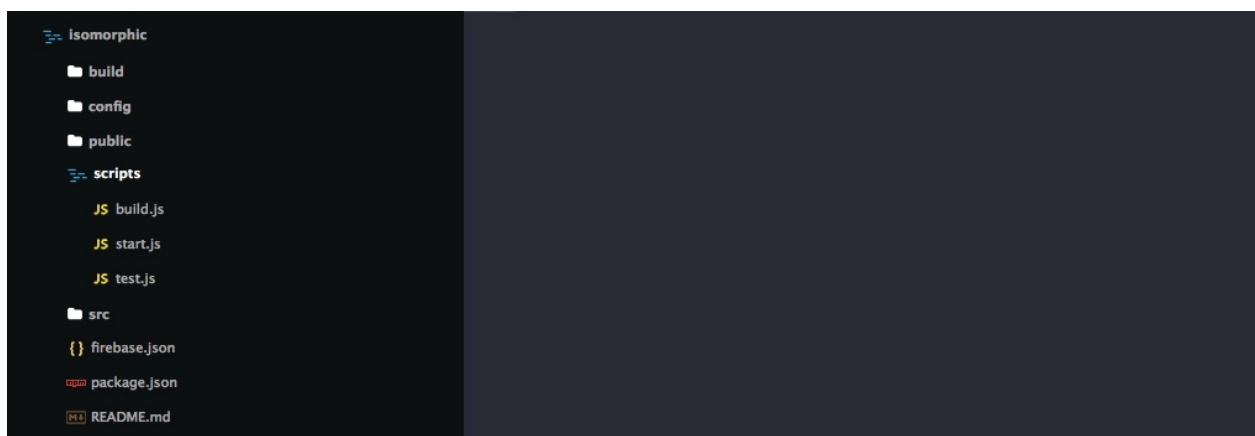


node_modules: It contains all the npm packages that is used on this projects.

public: Contains public files used on the projects like menifest file, index.html file, icon files.



scripts: Contains files that fires up the build file on the config for the node environment.



src: Contains all the codes including js, less and the image files. It has some folders inside. They are:

- components: Reusable react components
- containers: Constains all the files of the react component of the project.
- config: General config files.

- helpers: Utility codes for the projects.
- image: Images used in the project.
- reducers: Contains the functional code of redux.
- sagas: React sagas for handling async request.
- selectors: React selectors
- store: Redux Stores
- styles: Less code files.



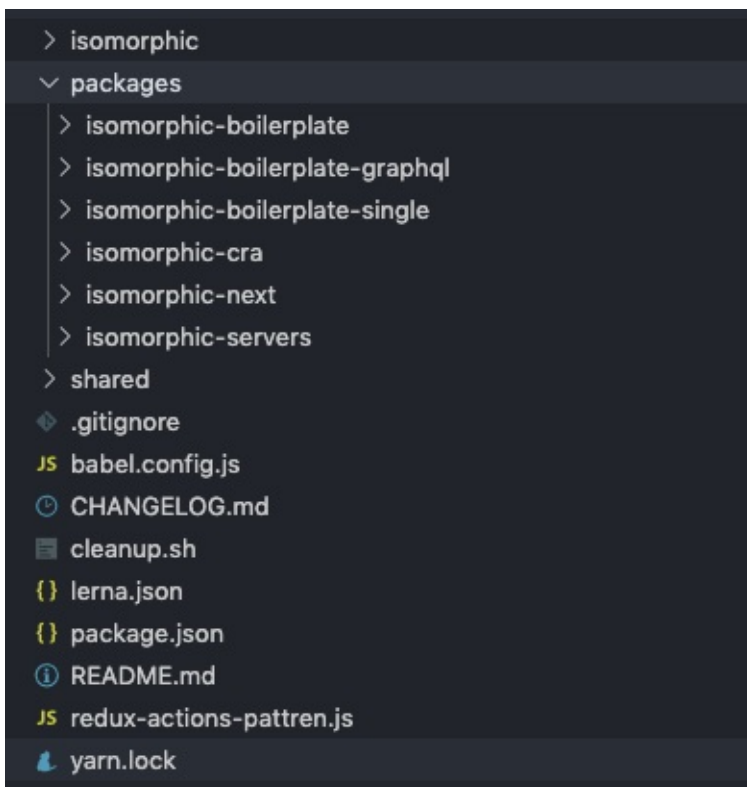
package.json: Contains all the informations about the project like third party packages, scripts etc .

server.js: The file fires up the node server.

Monorepo Packages Overview

This section will help you to understand the folder structures monorepo package. Inside the Root Isomorphic directory there you will see a packages directory where you will find the below packages

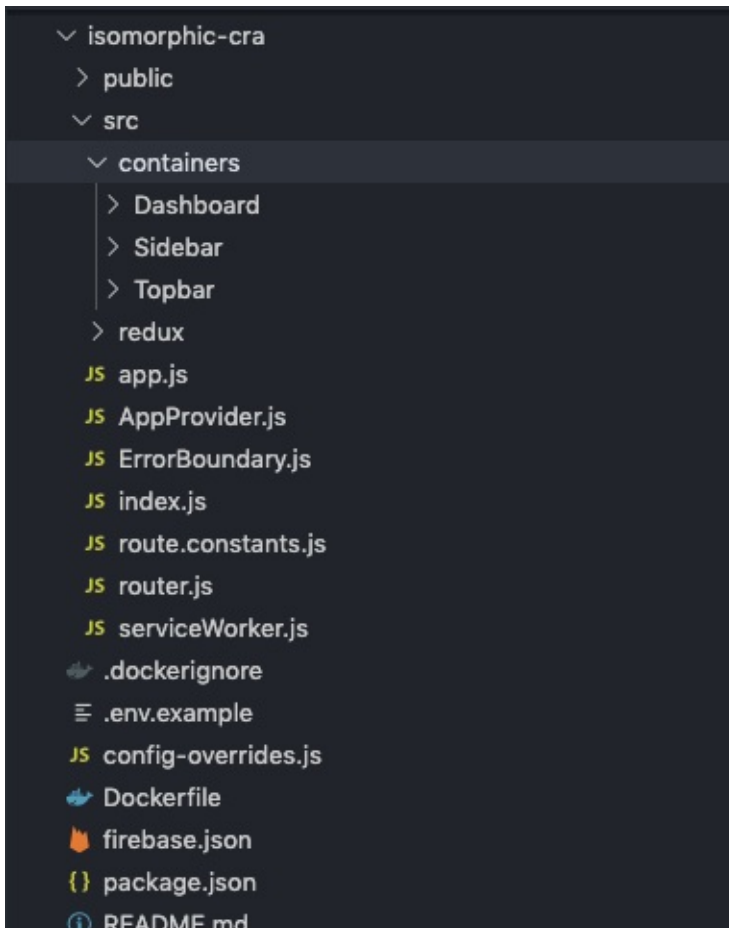
1. isomorphic-cra
2. isomorphic-next
3. isomorphic-boilerplate
4. isomorphic-boilerplate-graphql
5. isomorphic-boilerplate-single
6. isomorphic-servers



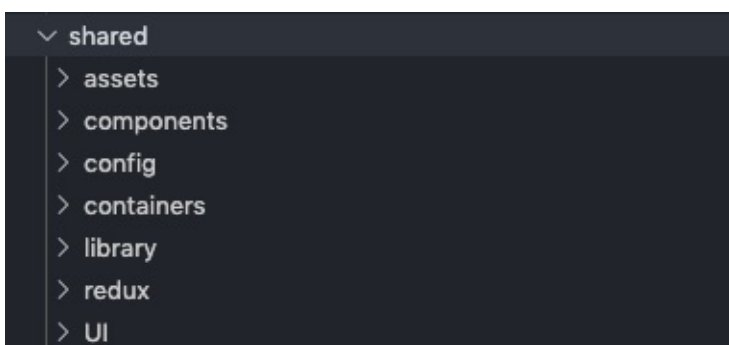
Lets go to the next section to find out details about them.

isomorphic-cra

isomorphic-cra is bootstrapped create react app. You can see the isomorphic-cra directory from the below screenshot,



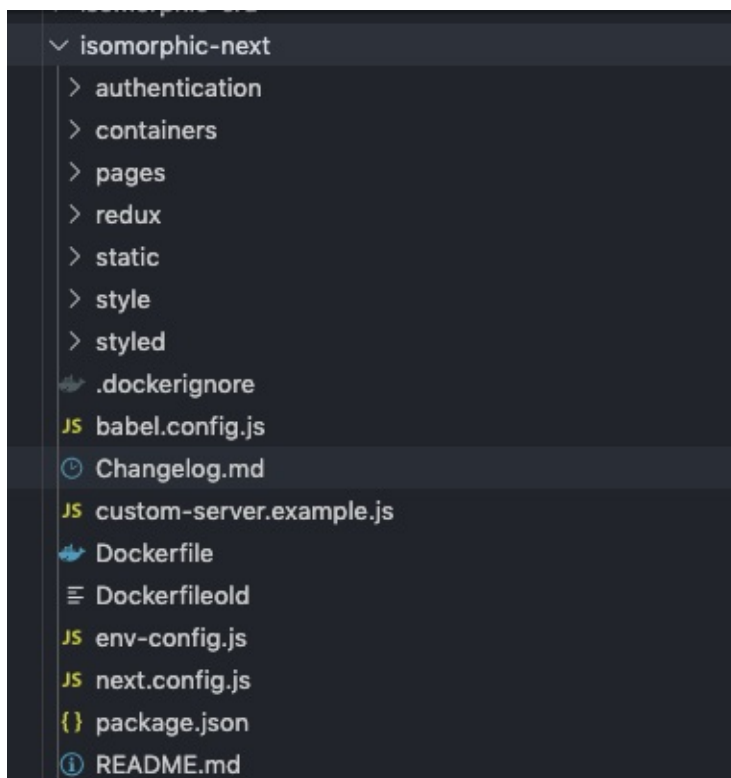
and most of the components of isomorphic cra is coming from the shared directory,



please note that shared is also another monorepo package.

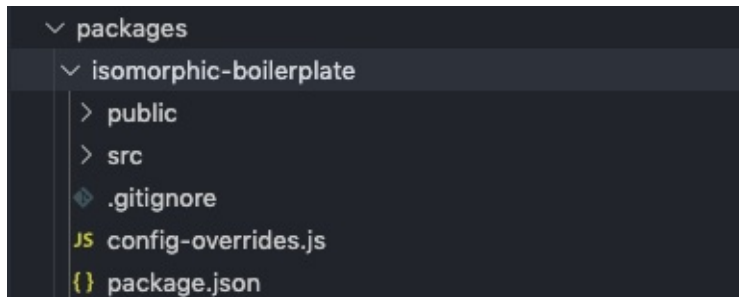
isomorphic-next

isomorphic next is bootstrapped with next js and in below screenshot you can see the folder structure,



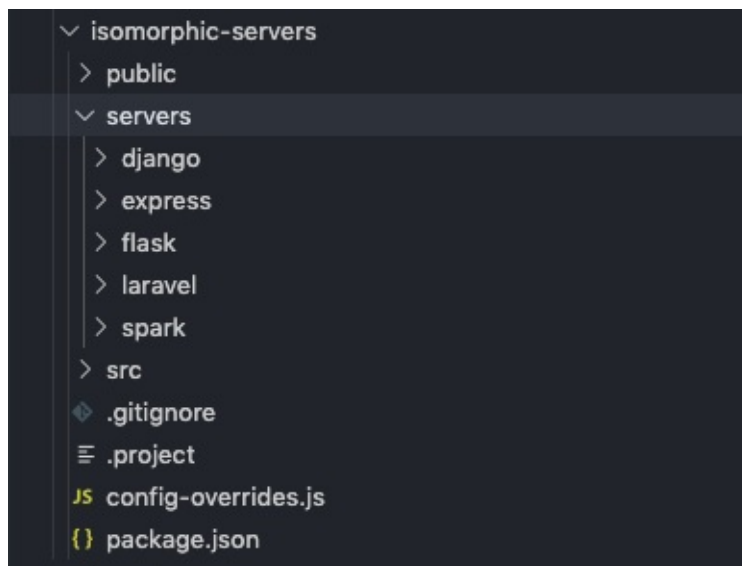
isomorphic-boilerplate

if Anyone want to use with monorepo and custom CRA



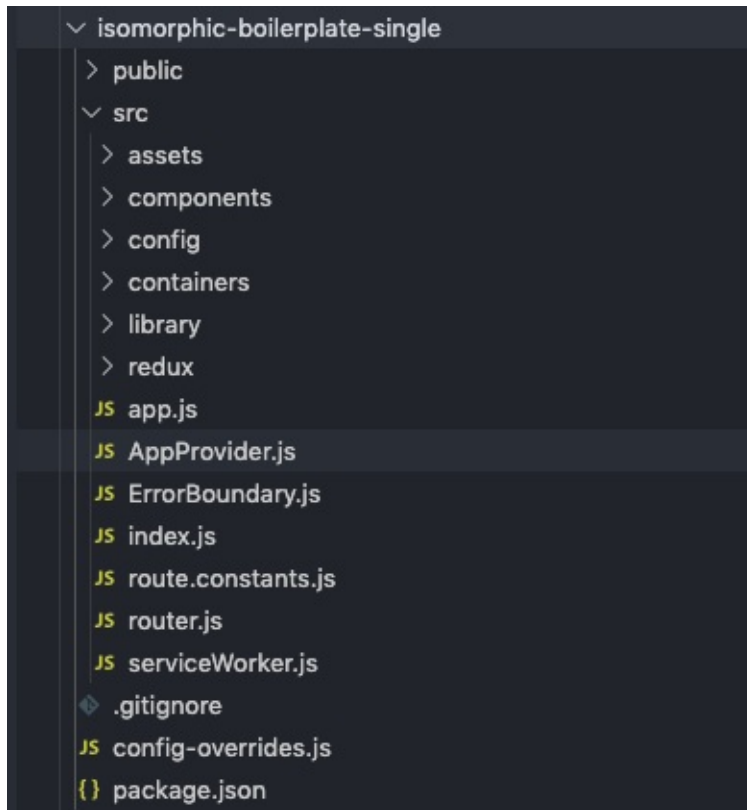
Isomorphic-servers.zip

Some server side integration is provided in Isomorphic. Three server side support is already integrated like Laravel, Spark, express. We have some upcoming features in the queue also.



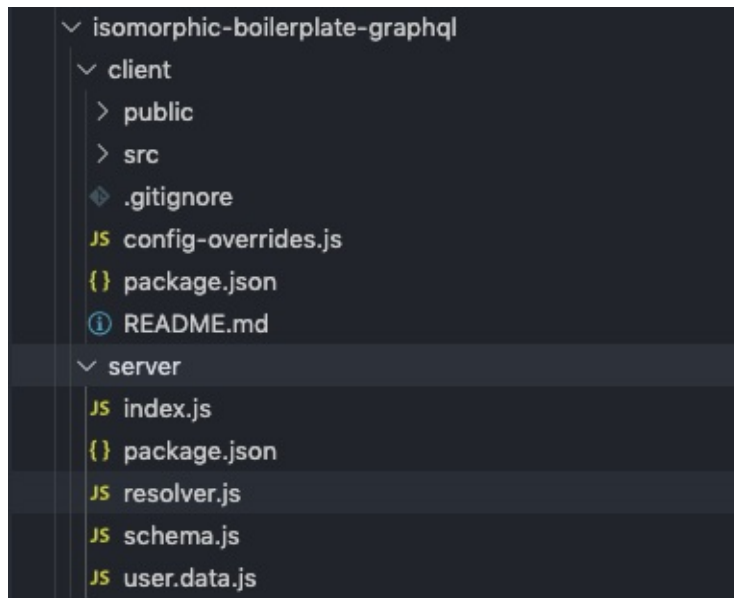
isomorphic-boilerplate-single

If Anyone want to use isomorphic boilerplate without monorepo,



Isomorphic-boilerplate-graphql

CRA with GraphQL server and client



In This section We are giving some hint where to start from depending on your purpose and some basic things of Isomorphic that you can be benefited with.

1. Use Boilerplate

If you are working on the demo that is Isomorphic.zip, Please try to code on the CustomApp folder. So that on the future updates you can replace your component and container and redux folders and it will not mess up with your work.

2. Basic JWT Auth

If you want to get stated JWT Auth, just go to the Isomorphic-servers.zip. We have implemented the basic jwt auth with 3 server side implementation.

3. Firebase Authentication and Auth0 Authentication

Check the Documentation in the section Components/Authentication/

4. Starting From a Blank Page

If you want to start working on a blank page just try to use the Isomorphic boilerplate. You will have the chance to start from the scratch.

We have used styled component in our Isomorphic. Credit goes to the libraries like [styled-components](#) and [styled-theme](#).

AsyncComponent

Isomorphic supports `AsyncComponent` . All the component being used in this app are based on `React Router 4` . Using `asyncComponent` provide you the facility to `Asynchronously` load components and feed them into `Match on route changes` .

you will find the `asyncComponent` related function and necessary code from the below file

To find out the code of `asyncComponent` related router, please go to your-apps-root-path/src/containers/App/AppRouter.js

To find out the code of `asyncComponent` related function, please go to your-apps-root-path/src/helpers/AssyncFunc.js

The Basic Route Component for The Widgets,

```
<Route
  exact
  path={`${url}/`}
  component={asyncComponent(() => import('../widgets/index.js'))
}
/>
```

The `asyncComponent` function

```
export default function asyncComponent(importComponent) {
  class AsyncFunc extends Component {
    constructor(props) {
      super(props);
      this.state = {
        component: null,
      };
    }
    componentWillMount() {
      Nprogress.start();
    }
    async componentDidMount() {
      const { default: Component } = await importComponent();
      Nprogress.done();
      this.setState({
        component: <Component {...this.props} />,
      });
    }

    render() {
      console.log(this.state.component, 'insided');
      const configs = {
        Component: this.state.component,
        props: this.props,
        holderComponent: 'asyncFunc',
      };
      return <HolderComponent {...configs} />;
    }
  }
  return AsyncFunc;
}
```


Components

We are using [Ant Design](#) components for Ui and layouts. Currently we are using Ant Design version 3.24.2 react javascript version.

Algolia

We have provided shop page in **Isomorphic**. A searching system is also provided with the help of [Algolia](#).

Why Algolia

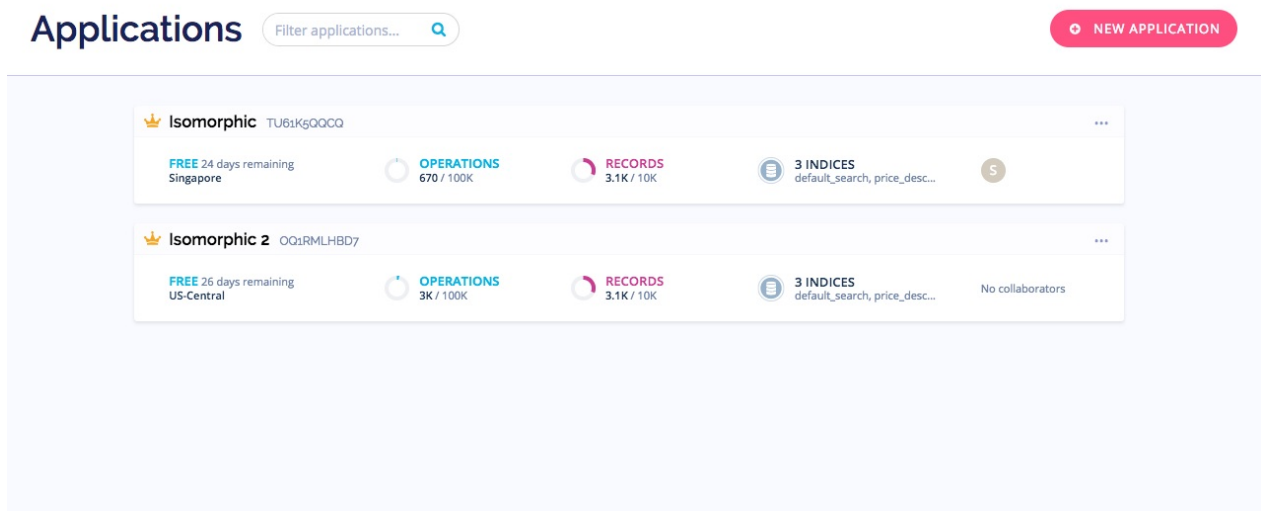
- Its always better to search from the JSON data instead of searching from the DB.
- Get Result in milliseconds.
- Easier Integration

How to Use

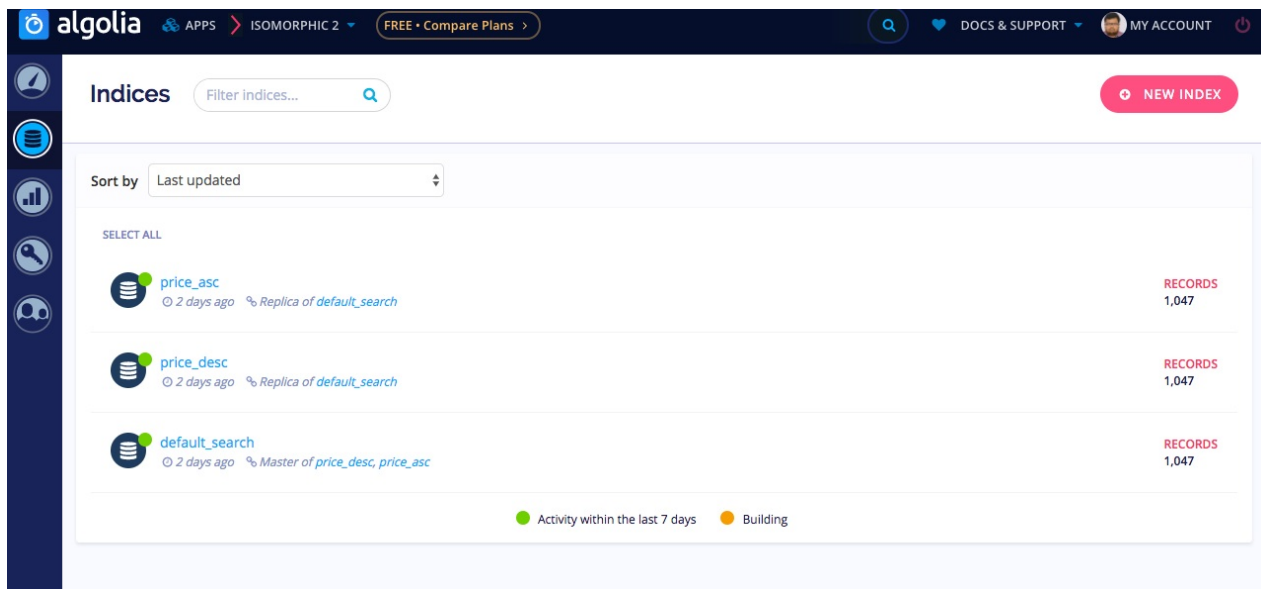
Algolia Dashboard Part

To Use aloglia, first you need to go to algolia official website and open an app and provide Necessary indices and data and configure as per as their [documentation](#).

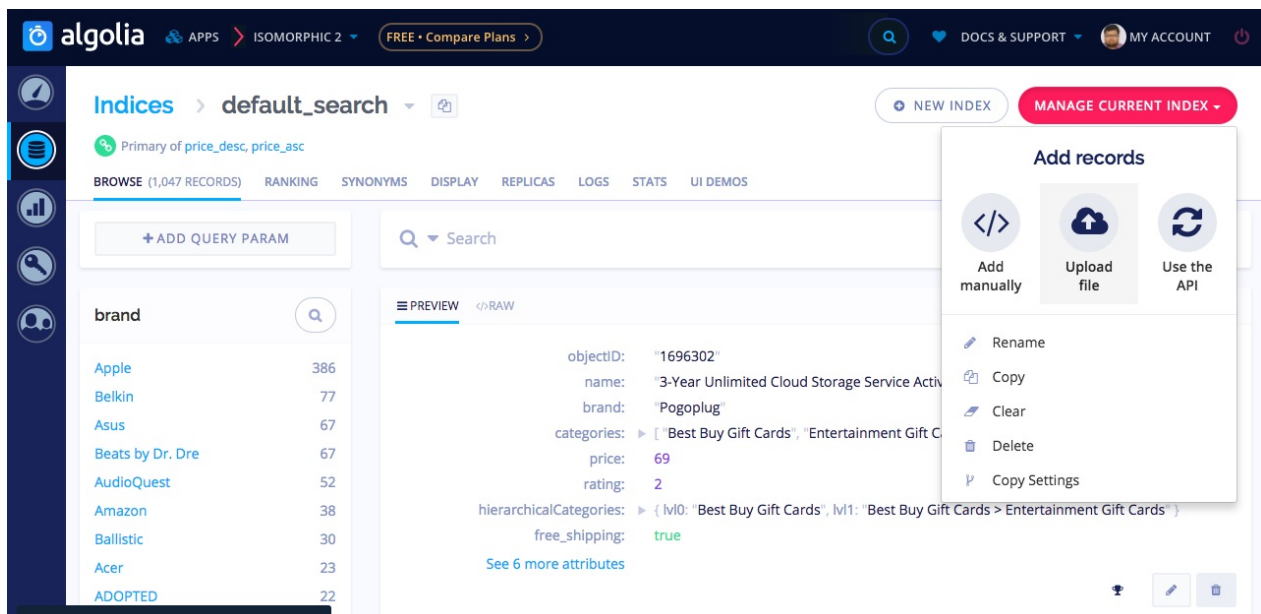
First you need to create an app after signing in to algolia.



Then entering into the application that is created, the admin dashboard will look like the following.



In algolia the search is truly dependent on the indices. So its time to create a indices. And after that the indices should be modified in order to use. So, the arrays of data on which the search operations should be done, should be uploaded. A JSON file can serve the purpose.



More details on configuring the Algolia for using from Web SDK can be found [here](#).

On Isomorphic Project

The data we have worked on is simple like the following

```
{
  "name": "3-Year Unlimited Cloud Storage Service Activation Card - Other",
  "description": "Enjoy 3 years of unlimited Cloud storage service with this activation card, which allows you to remotely access your favorite music, movies and other media via a compatible device and enables private file sharing with loved ones.",
  "brand": "Pogoplug",
  "categories": [
    "Best Buy Gift Cards",
    "Entertainment Gift Cards"
  ],
  "hierarchicalCategories": {
    "lvl0": "Best Buy Gift Cards",
    "lvl1": "Best Buy Gift Cards > Entertainment Gift Cards"
  },
  "type": "Online data backup",
  "price": 69,
  "price_range": "50 - 100",
  "image": "https://cdn-demo.algolia.com/bestbuy/1696302_sc.jpg",
  "url": "http://www.bestbuy.com/site/3-year-unlimited-cloud-storage-service-activation-card-other/1696302.p?id=1219066776306&skuId=1696302&cmp=RMX&ky=1uWSHMdQqBeVJB9cXgEke60s5EjfS6M1W",
  "free_shipping": true,
  "popularity": 10000,
  "rating": 2,
  "objectID": "1696302"
}
```

The Total Project is structured in such a systematic way that can be easily understandable by any developer.

File Path	Uses
src/config.js	App Credentials like appId and apiKey
src/containers/Ecommerce/algolia/instantSearch.js	Entry point for Ecommerce Shop Page
src/containers/Ecommerce/algolia/desktopview.js or src/containers/Ecommerce/algolia/mobileview.js	Entry point for Algolia Components based on device (Mobile or Desktop)
src/components/algolia/sidebar	Side bar component that holds the search elements
src/components/algolia/contents	The Grid part that shows the Product overview
src/components/algolia/footer	Footer of the Shop page

In our project you will find the algolia config on the path `src/config.js` . You should provide two options from that is essential.

```
const AlgoliaSearchConfig = {
  appId: '',
  apiKey: ''
};
```

In the Entry point for algolia components a config should be provided indicating the available options like URL sync, indices name etc.

```
const searchInfo = {
  ...AlgoliaSearchConfig,
  indexName: 'default_search',
  searchState: this.state.searchState,
  urlSync: true,
  onSearchStateChange: searchState => {
    this.setState({ searchState });
    setUrl(searchState);
  },
};
```

and this config should be passed to a react component([react-instantsearch/dom](#)) like the following.

```
<InstantSearch {...searchInfo}>
  <div className="isoAlgoliaMainWrapper">
    <Sidebar setVoice={this.setVoice} />
    <Content />
  </div>
  <Footer />
</InstantSearch>
```

Sidebar component holds the search elements. For example for a rangebox the following code should be placed in the sidebar.

```
<MultiRange
  attributeName="price"
  items={[
    { end: 10, label: '<$10' },
    { start: 10, end: 100, label: '$10-$100' },
    { start: 100, end: 500, label: '$100-$500' },
    { start: 500, label: '>$500' }
  ]}
/>
```

Search here

Start Speaking

Multi Range

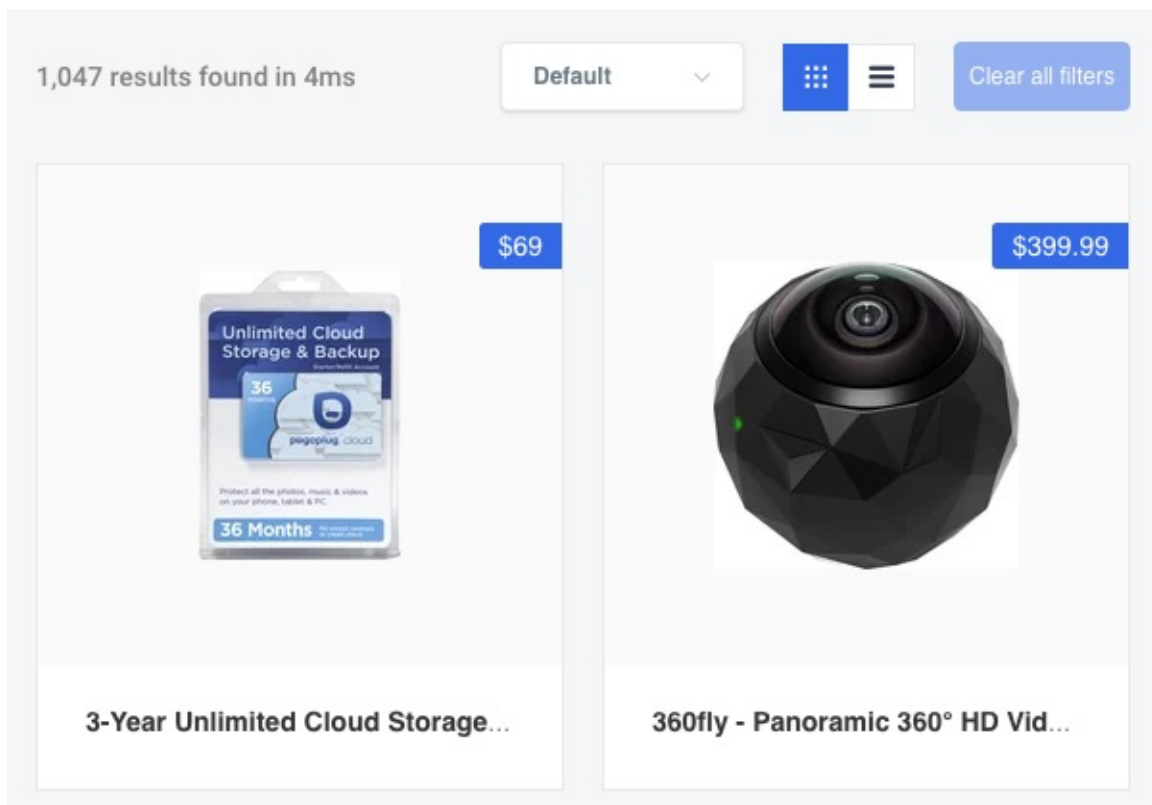
- ☐ <\$10
- ☐ \$10-\$100
- ☐ \$100-\$500
- ☐ >\$500
- ☒ All

Slider 2.99 - 3199.99

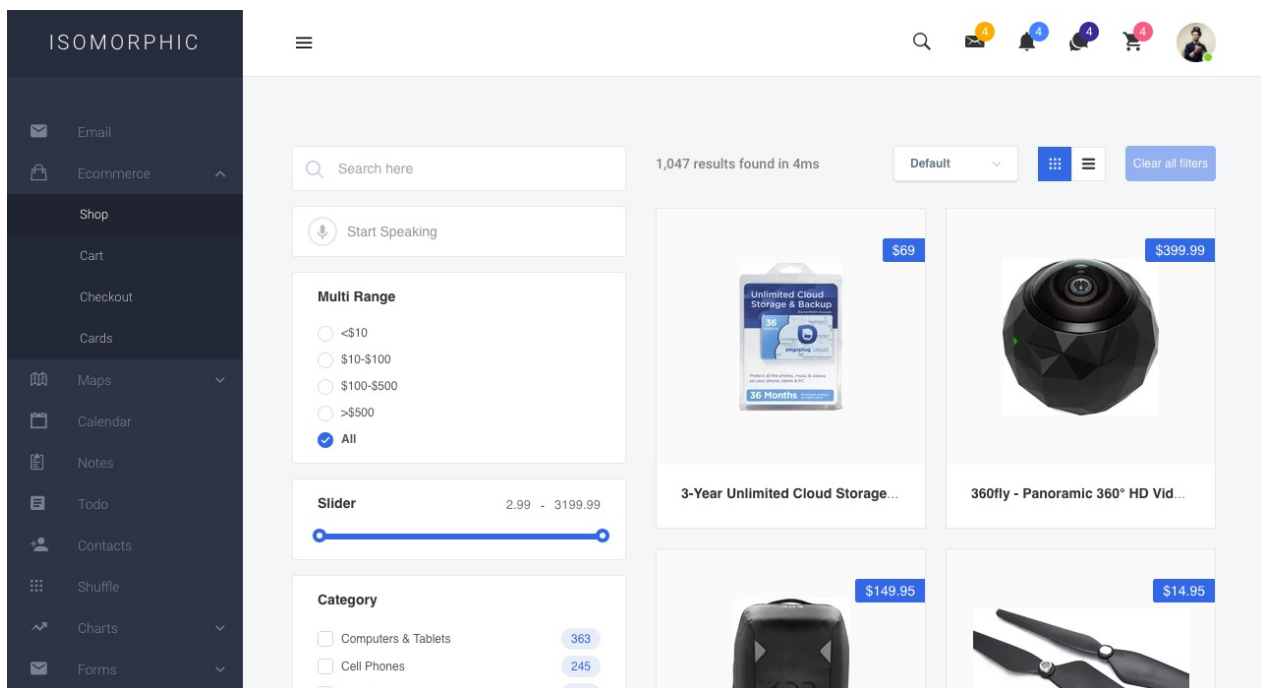
Category

<input type="checkbox"/> Computers & Tablets	363
<input type="checkbox"/> Cell Phones	245
<input type="checkbox"/> Audio	120

Content components hold the Product overview part and the pagination . in other word we can call it as grid. Single Grid path is `src/components/algolia/hit.js`



So the total outlook of our shop page will be something like the following.



Authentication

We provided two third party Auth integration in our Project. There are

1. Firebase
2. Auth0

A brief description about them is given bellow.

Firestore

Firestore-Doc

Folder path: /src/components/firebase/

If you want a login button like the following image given bellow.



On the `onclick` function of the button You can render a form with two input fields(one for Email and one for Password). And passing those credentials through the Firestore Api, a user can be get signed up or signed in. The following code is connects the User inputs to the Firestore Api

```
Firestore.login(Firebase.EMAIL, { email, password })
```

And you can use basic Javascript promise methods like `then()` or `catch()` methods and use the Firestore returned data to do your desired operations like the following code

```
Firestore.login(Firebase.EMAIL, { email, password })
  .then(result => {
    // Do something
  })
  .catch(error => {
    // Handle your error precisely
  })
```

To Use the Firestore Api you need to configure your app to the [Firestore Official Website](#) first. And put your app credentials to the config file of our app.

Path to the config file: /src/config.js

The following are the important Credentials you must provide in order to make Firestore Authentication work.

Keys
apiKey
authDomain
databaseURL
projectId
storageBucket
messagingSenderId

Currently We are providing Firestore authentication through Email and password only. You can also integrate Social Logins (Facebook, Google, Github, Twitter) with Firestore. The code will be found on the following Path: `Folder path: /src/helpers/firestore/`

And the code is like the following.

Authentication Type	Helper function	Manual Co
Basic(Email, Password)	firebaseAuth().signInWithEmailAndPassword()	Firestore.log email, pass
Facebook	firebaseAuth().FacebookAuthProvider()	Firestore.log
Google	firebaseAuth().GoogleAuthProvider()	Firestore.log
Github	firebaseAuth().GithubAuthProvider()	Firestore.log
Twitter	firebaseAuth().TwitterAuthProvider()	Firestore.log

After all those Works clicking the Login with Firestore button A prompt Window like the following will open

Sign in with Firebase

Email

demo@gmail.com

Password

Reset Password

Cancel

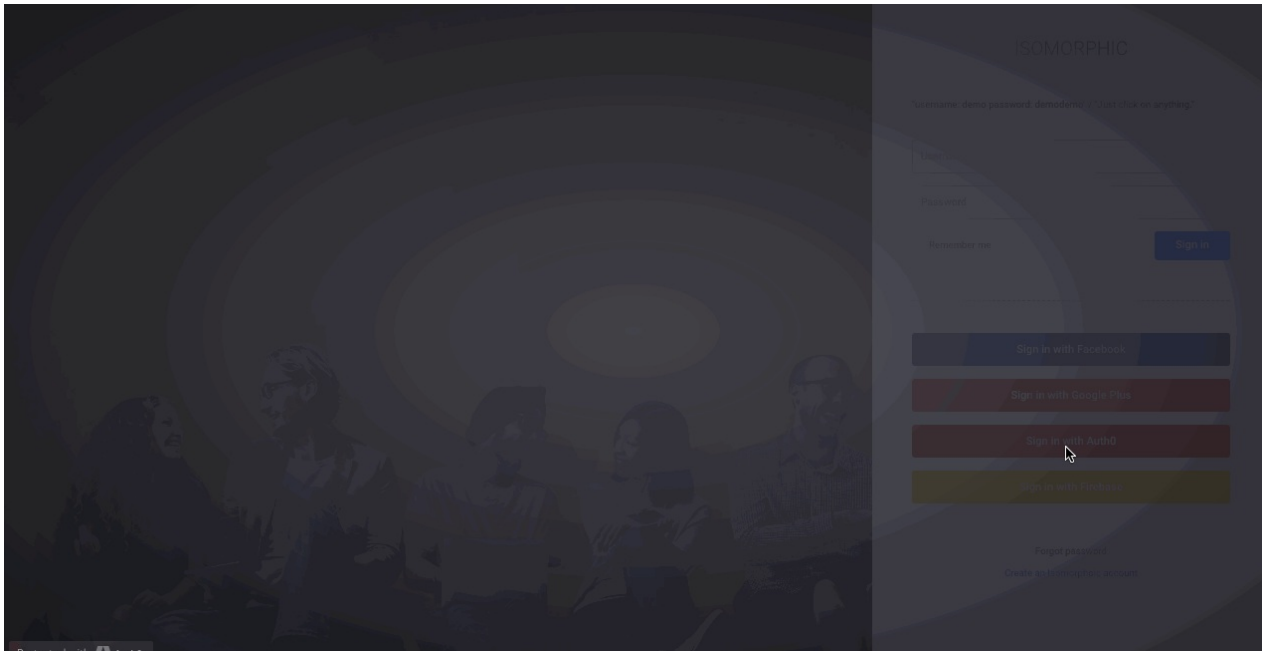
Login

Auth0

[Auth0-official-website](#)

Folder path: `/src/helpers/auth0/`

If you want a login button like the following image given bellow.



Just need to Place a button and On the onclick function of the button You can render the following functions provided by Auth0 itself.

Function	Details
<code>new Auth0Lock()</code>	Instantiating Lock
<code>getUserInfo()</code>	Obtaining the profile of a logged in user
<code>show()</code>	Showing the lock widget
<code>on()</code>	Listening for events
<code>logout()</code>	Log out the user

We are using the firebase [Lock](#) widget.

In the folder path `/src/helpers/auth0/` there is a `Auth0Helper` class that uses all the functions.

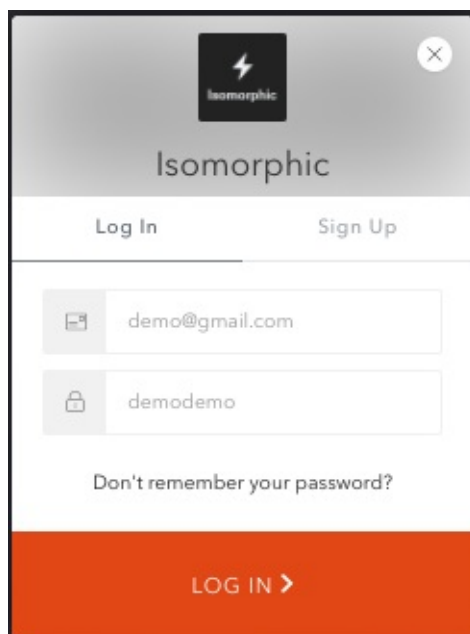
To Use the Firebase Api you need to configure your app to the Auth0-Official documentation first. And put your app credentials to the config file of our app.

Path to the config file: `src/settings/index.js`

The following are the important Credentials you must provide in order to make Firebase Authentication work.

Keys
clientID
domain

After all those Works clicking the Login with Auth0 button A prompt Window like the following will open

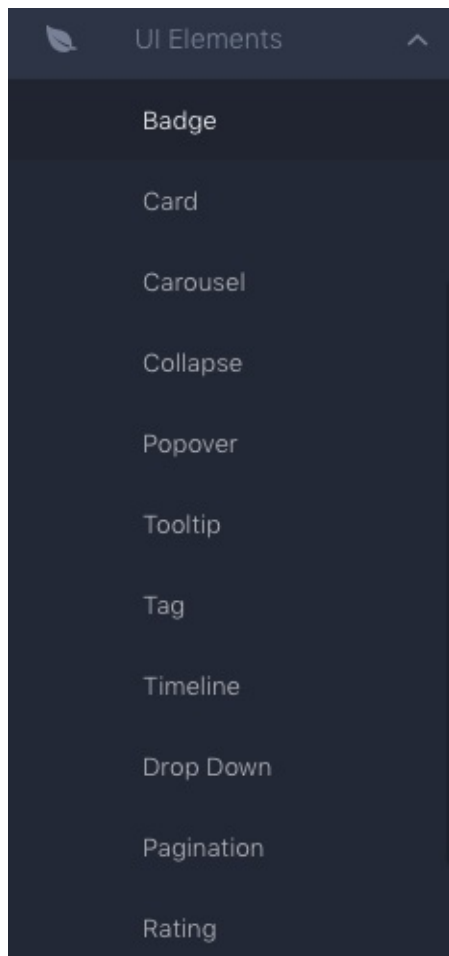


UI Elements

11 unique UI Elements are available with this template.

- Badge
- Card
- Carousel
- Collapse
- Popover
- Tooltip
- Tag
- Timeline
- Dropdown
- Pagination
- Rating

You will find this in the left sidebar menu. The screenshot is given below.



Badge

4 types of badge design are available inside badge menu. You can use any of this. To use this follow the instruct code example given below.

Basic Example

Basic Example

Simplest Usage. Badge will be hidden when count is 0, but we can use showZero to show it.



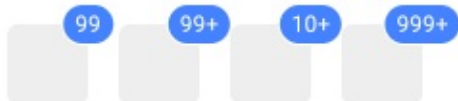
To show this type of badge you will need this code in down here.

```
<Badge count={5}>
  <a className="isoBadgeLink"> </a>
</Badge>
```

Overflow Count

Overflow Count

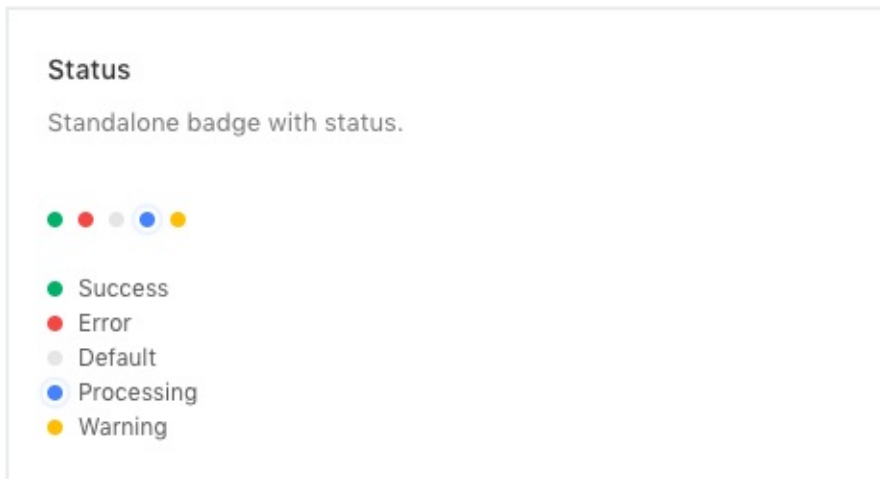
OverflowCount is displayed when count is larger than overflowCount. The default value of overflowCount is 99.



If your number of badge is too large then use the overflow count example code.

```
<Badge count={1000} overflowCount={999}>
  <a className="isoBadgeLink"> </a>
</Badge>
```

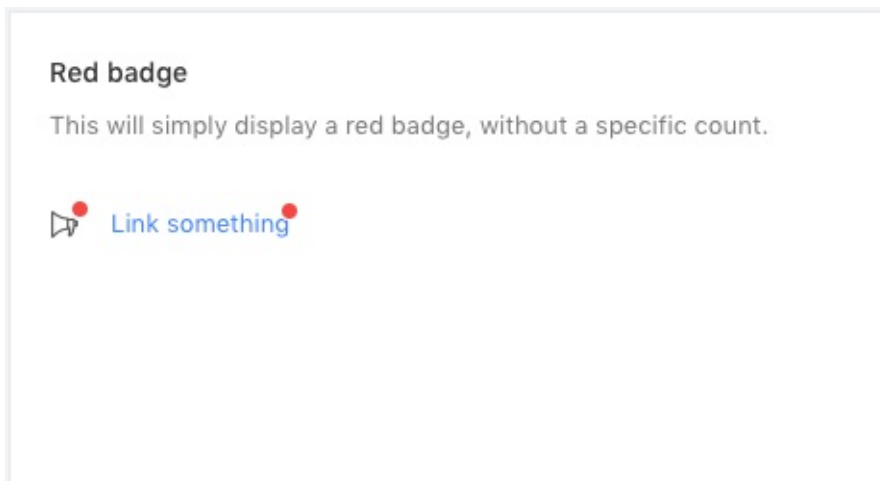

Status



Available status colors are here.

```
<Badge status="success" />
<Badge status="error" />
<Badge status="default" />
<Badge status="processing" />
<Badge status="warning" />
```

Red Badge



You can add it to any link or some icon.

```
<Badge dot>
  <Icon type="notification" />
</Badge>
<Badge dot>
  <a href=".">Link something</a>
</Badge>
```

Available parameters, type and descriptions are down below.

Parameter	Type	Description
count	integer	give the no you want to show as notification
overflowCount	integer	maximum threshold value for showing the counter display
status	string	success, error, default, processing, warning are available for choosing color option.
dot	null	only show a red dot

You will find the example code inside

```
src/containers/Uielements/Badge/index.js
```

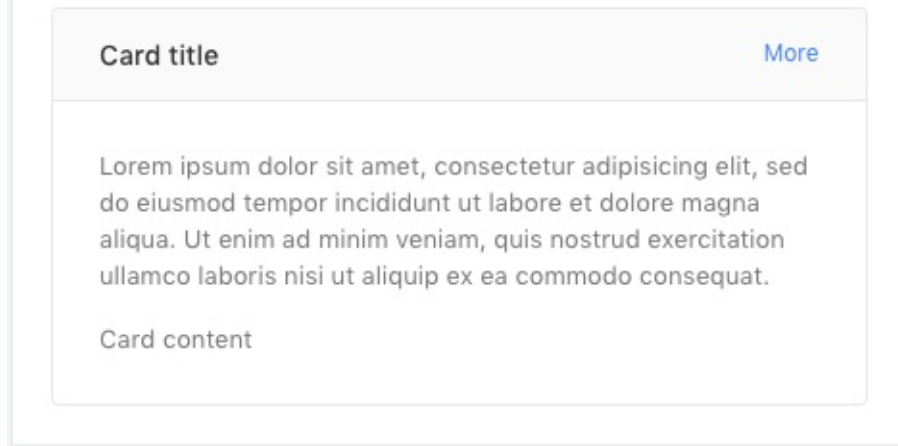
Card

Multiple types of cards are available with this isomorphic template.

Basic Card

Basic card

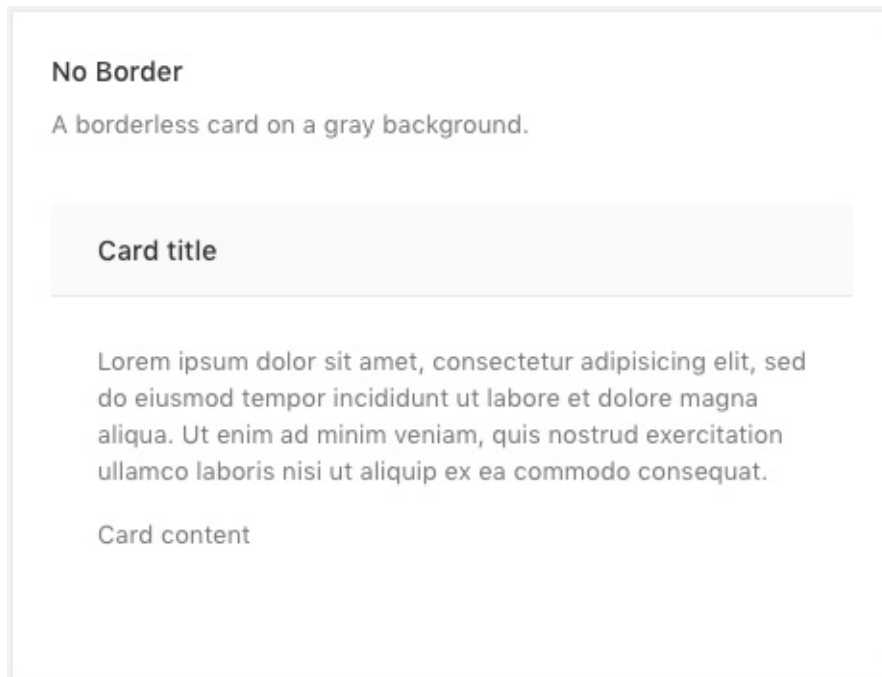
A basic card containing a title, content and an extra corner content.



Simple basic card preview.

```
<Card
  title="Card title"
  extra={<a>More</a>}
  style={{ width: '100%' }}
>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, se
    d do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco labo
    ris nisi ut aliquip ex ea commodo consequat.
  </p>
  <p>Card content</p>
</Card>
```

No border



Card without border preview screenshot will look like this, and the example code you need:

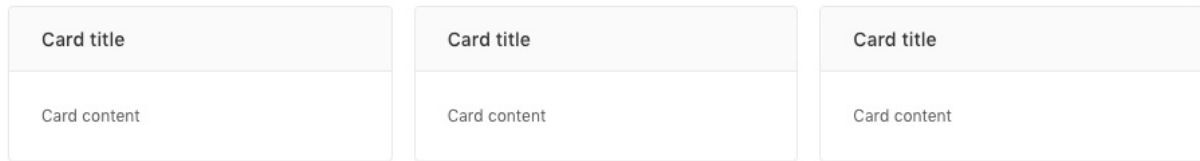
```
<Card
  title="Card title"
  bordered={false}
  style={{ width: '100%' }}
>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, se
    d do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco labo
    ris nisi ut aliquip ex ea commodo consequat.
  </p>
  <p>Card content</p>
</Card>
```

Grid Card

You can preview your card in grid system. To use grid follow the steps.

Grid card

Cards usually cooperate with grid layout in overview page.

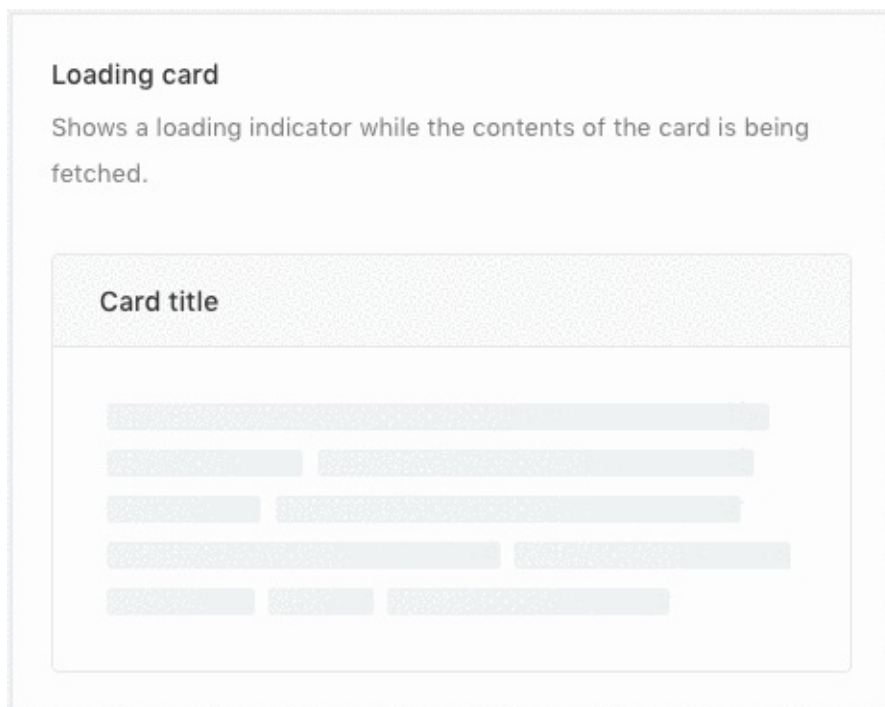


```
<Row>
  <Col span="8">
    <Card title="Card title">Card content</Card>
  </Col>
  <Col span="8">
    <Card title="Card title">Card content</Card>
  </Col>
  <Col span="8">
    <Card title="Card title">Card content</Card>
  </Col>
</Row>
```

Loading Card

Loading card

Shows a loading indicator while the contents of the card is being fetched.



```
<Card loading title="Card title" style={{ width: '100%' }}>
  Whatever content
</Card>
```

Customized Content

Customized Content

Shows a loading indicator while the contents of the card is being fetched.

Europe Street beat
www.instagram.com

You can customize your card any type of design you want.

```
<Card bodyStyle={{ padding: 0 }}>
  <div className="custom-image">
    
  </div>
  <div className="custom-card">
    <h3>Europe Street beat</h3>
    <p>www.instagram.com</p>
  </div>
</Card>
```

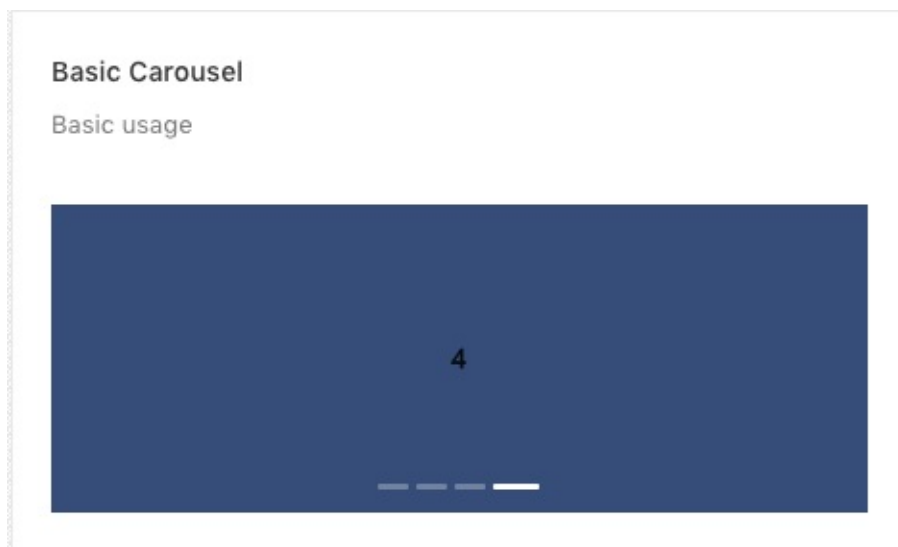
Available parameters, type and descriptions are down below.

Parameter	Type	Description
title	string	enter your card title
extra	html	you can add HTML content
style	object	use custom css style
bordered	boolean	show/hide card border
loading	null	it will show animated loading card
bodyStyle	object	use custom css body style

Carousel

Simple but effective carousel are added with this template. Using the carousel is very easy and just need some simple step.

Basic Carousel



```
<Carousel afterChange={this.onChange}>
  <div><h3>1</h3></div>
  <div><h3>2</h3></div>
  <div><h3>3</h3></div>
  <div><h3>4</h3></div>
</Carousel>
```

Vertical Carousel

Fade In Transition

Slides use fade for transition. {effect='fade'}



```
<Carousel vertical="true">
  <div><h3>1</h3></div>
  <div><h3>2</h3></div>
  <div><h3>3</h3></div>
  <div><h3>4</h3></div>
</Carousel>
```

Scroll Automatically

Scroll Automatically

Timing of scrolling to the next card/picture. autoplay




```
<Carousel autoplay>
  <div><h3>1</h3></div>
  <div><h3>2</h3></div>
  <div><h3>3</h3></div>
  <div><h3>4</h3></div>
</Carousel>
```

Available parameters, type and descriptions are down below.

Parameter	Type	Description
afterChange	function	you can use the callback function after change
vertical	boolean	enable to show vertical carousel
autoplay	null	scroll automatically

Collapse

Basic Collapse / Accordion

Collapse

More than one panel can be expanded at a time, the first panel is initialized to be active in this case. use {defaultActiveKey= ['keyNum']}

This is panel header 1



This is panel header 2



This is panel header 3



```
<Collapse accordion>
  <Panel header={'This is panel header 1'} key="1">
    <p>{text}</p>
  </Panel>
  <Panel header={'This is panel header 2'} key="2">
    <p>{text}</p>
  </Panel>
  <Panel header={'This is panel header 3'} key="3">
    <p>{text}</p>
  </Panel>
</Collapse>
```

Nested Example

Nested Example

Collapse is nested inside the Collapse.

This is panel header 1



This is panel header 2



This is panel header 3



```
<Collapse onChange={this.callback}>
  <Panel header={'This is panel header 1'} key="1">
    <Collapse defaultActiveKey="1">
      <Panel header={'This is panel nest panel'} key="1">
        <p>{text}</p>
      </Panel>
    </Collapse>
  </Panel>
  <Panel header={'This is panel header 2'} key="2">
    <p>{text}</p>
  </Panel>
  <Panel header={'This is panel header 3'} key="3">
    <p>{text}</p>
  </Panel>
</Collapse>
```

Borderless Example

Borderless Example

A borderless style of Collapse. use {bordered={false}}

This is panel header 1



A dog is a type of domesticated animal. Known for its loyalty and faithfulness, it can be found as a welcome guest in many households across the world.

This is panel header 2



This is panel header 3



```
<Collapse bordered={false} defaultActiveKey={['1']}>
  <Panel header="This is panel header 1" key="1">
    <p>{text}</p>
  </Panel>
  <Panel header="This is panel header 2" key="2">
    <p>{text}</p>
  </Panel>
  <Panel header="This is panel header 3" key="3">
    <p>{text}</p>
  </Panel>
</Collapse>
```

Available parameters on **Collapse**, type and descriptions are down below.

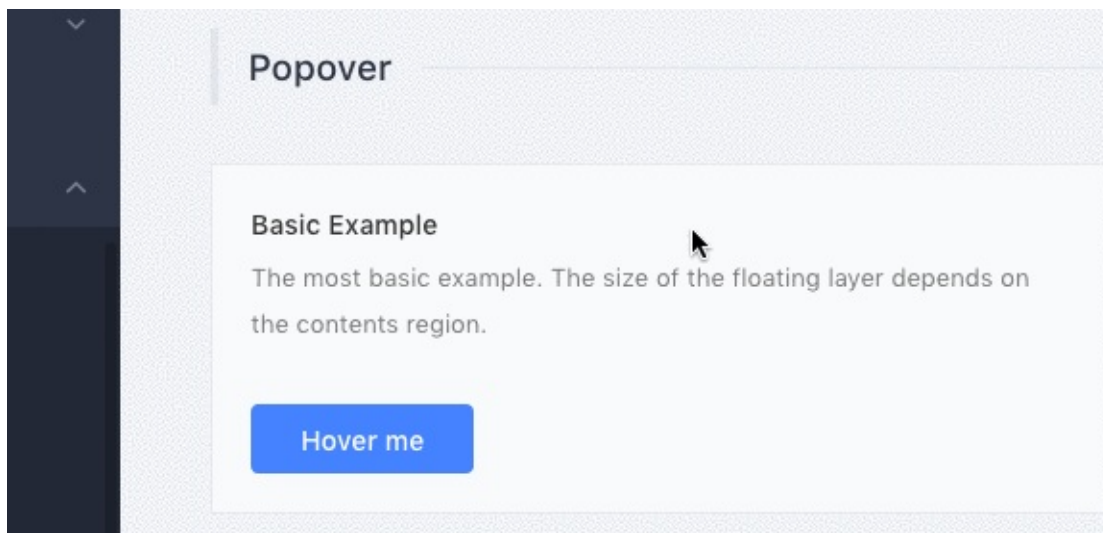
Parameter	Type	Description
onChange	function	callback function
accordion	null	accordion type
bordered	boolean	show / hide border
defaultActiveKey	object	default active

Available parameters on **Panel**, type and descriptions are down below.

Parameter	Type	Description
header	string	enter panel title
key	integer	add unique identifier for each panel

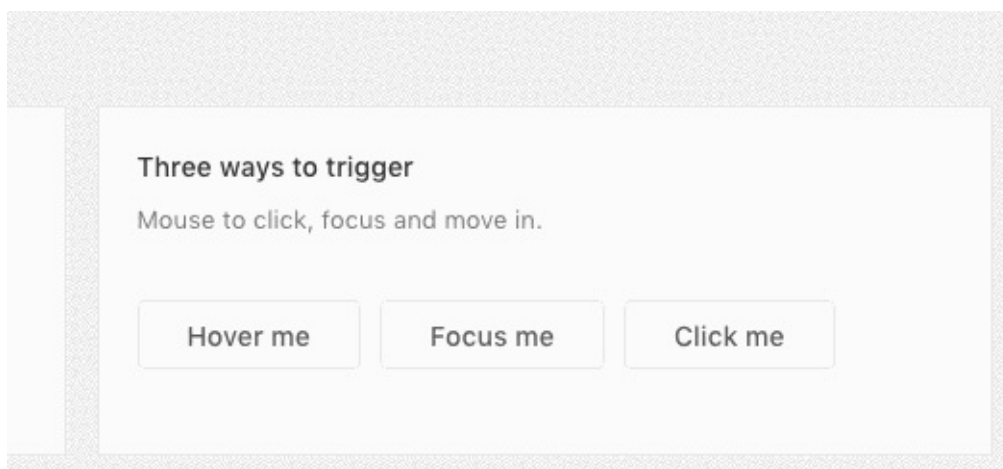
Popover

Basic Example



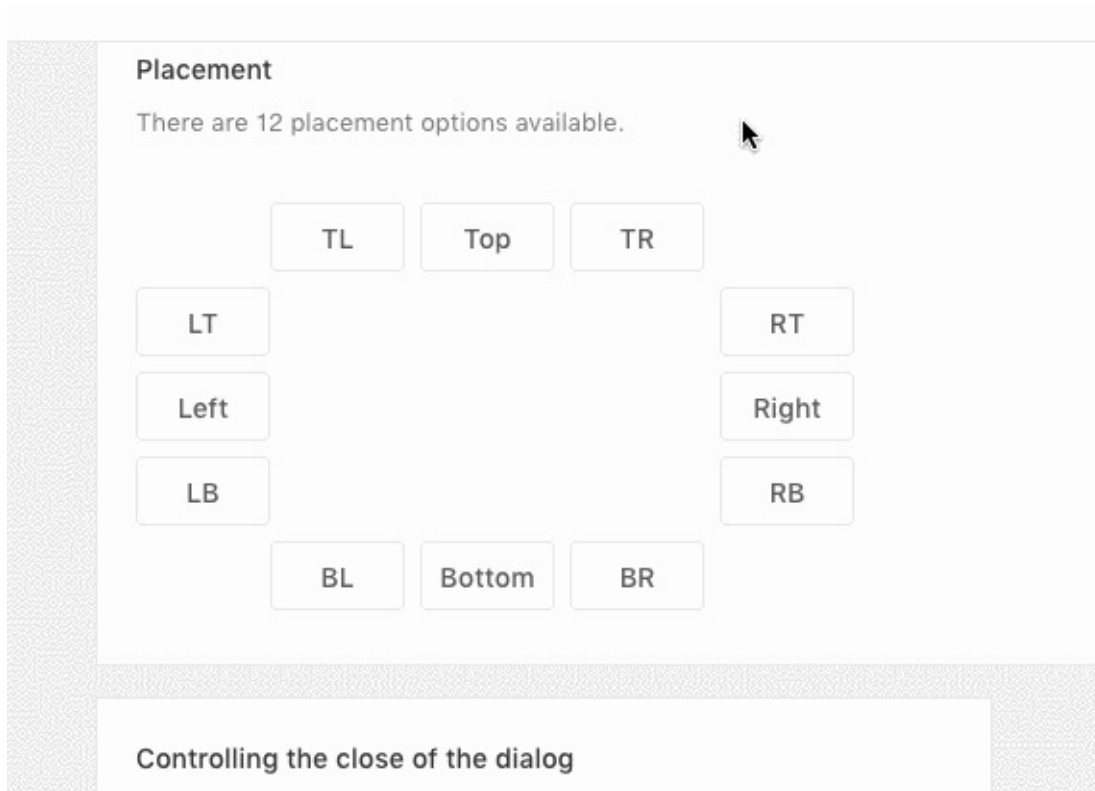
```
<Popover content={content} title="Title">
  <Button type="primary">Hover me</Button>
</Popover>
```

Three ways to trigger



```
<Popover content={content} title="Title" trigger="hover">
  <Button className="demoBtn">Hover me</Button>
</Popover>
<Popover content={content} title="Title" trigger="focus">
  <Button className="demoBtn">Focus me</Button>
</Popover>
<Popover content={content} title="Title" trigger="click">
  <Button className="demoBtn">Click me</Button>
</Popover>
```

Placement Options



```
<Popover
  placement="topLeft"
  title="Top Left"
  content={content}
  trigger="click"
>
  <Button className="demoPosBtn">TL</Button>
</Popover>
<Popover
  placement="top"
  title="Top"
  content={content}
  trigger="click"
>
  <Button className="demoPosBtn">Top</Button>
</Popover>
<Popover
  placement="topRight"
  title="Top Right"
  content={content}
  trigger="click"
```

```
>
  <Button className="demoPosBtn">TR</Button>
</Popover>
</div>
<div
className="demoBtnsWrapper"
style={{ width: buttonWidth, float: 'left' }}
>
  <Popover
    placement="leftTop"
    title="Left Top"
    content={content}
    trigger="click"
  >
    <Button className="demoPosBtn">LT</Button>
  </Popover>
  <Popover
    placement="left"
    title="Left"
    content={content}
    trigger="click"
  >
    <Button className="demoPosBtn">Left</Button>
  </Popover>
  <Popover
    placement="leftBottom"
    title="Left Bottom"
    content={content}
    trigger="click"
  >
    <Button className="demoPosBtn">LB</Button>
  </Popover>
  <Popover
    placement="rightTop"
    title="Right Top"
    content={content}
    trigger="click"
  >
    <Button className="demoPosBtn">RT</Button>
  </Popover>
```

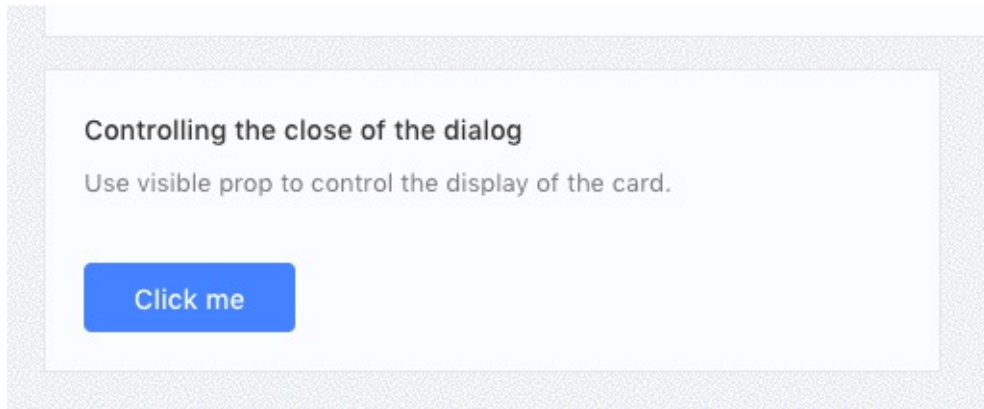
```
<Popover
  placement="right"
  title="Right"
  content={content}
  trigger="click"
>
  <Button className="demoPosBtn">Right</Button>
</Popover>
<Popover
  placement="rightBottom"
  title="Right Bottom"
  content={content}
  trigger="click"
>
  <Button className="demoPosBtn">RB</Button>
</Popover>

<Popover
  placement="bottomLeft"
  title="Bottom Left"
  content={content}
  trigger="click"
>
  <Button className="demoPosBtn">BL</Button>
</Popover>
<Popover
  placement="bottom"
  title="Bottom"
  content={content}
  trigger="click"
>
  <Button className="demoPosBtn">Bottom</Button>
</Popover>
<Popover
  placement="bottomRight"
  title="Bottom Right"
  content={content}
  trigger="click"
>
  <Button className="demoPosBtn">BR</Button>
```



```
</Popover>
```

Controlling the close of the dialog



```
<Popover
  content={<a onClick={this.hide}>Close</a>}
  title="Title"
  trigger="click"
  visible={this.state.visible}
  onVisibleChange={this.handleVisibleChange}
>
  <Button type="primary">Click me</Button>
</Popover>
```

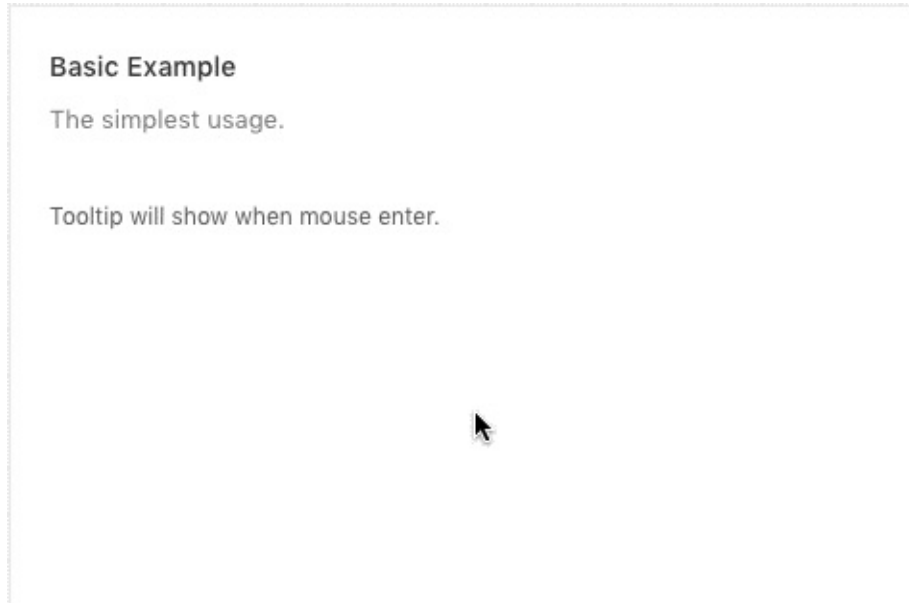
Available parameters, type and descriptions are down below.

Parameter	Type	Description
title	string	popover title
content	html	popover content
trigger	options	hover, focus, click
visible	boolean	visible state change
onVisibleChange	function	callback function
placement	options	topLeft, top, topRight, leftTop, left, leftBottom, rightTop, right, rightBottom, bottomLeft, bottom, bottomRight

Tooltip

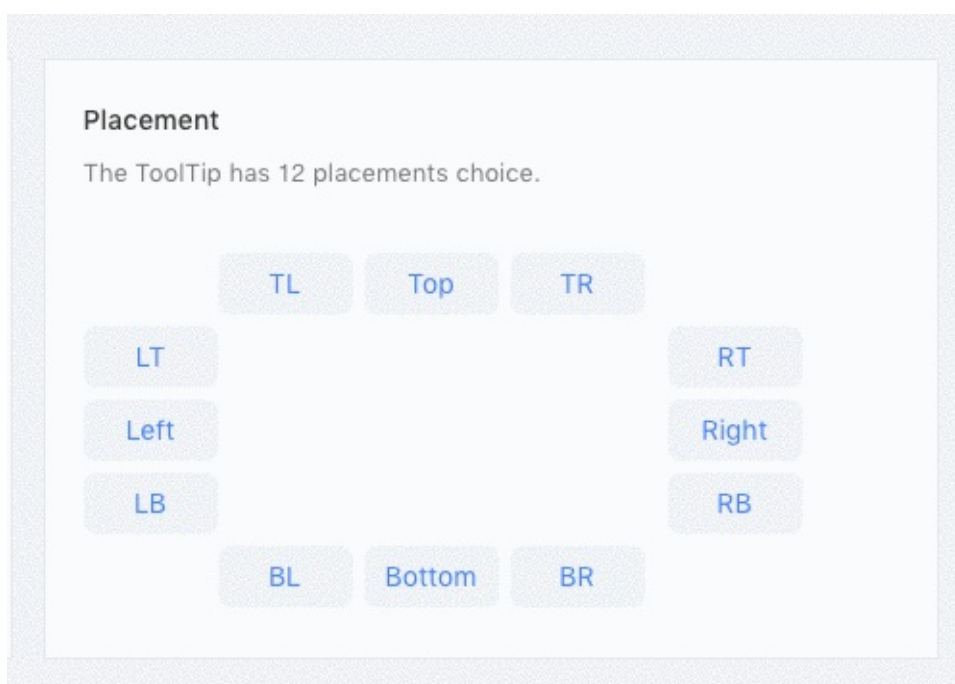
Every templates needs some basic tooltip in some places to focus the content more precisely. Here we have some tooltip options available for you.

Basic Example



```
<Tooltip title="Tooltip Content">  
  <span>Tooltip will show when mouse enter.</span>  
</Tooltip>
```

Placement Options



Multiple placement options are available with the tooltip.

```
<Tooltip placement="topLeft" title={text}>
  <a className="tooltipBtn">TL</a>
</Tooltip>
<Tooltip placement="top" title={text}>
  <a className="tooltipBtn">Top</a>
</Tooltip>
<Tooltip placement="topRight" title={text}>
  <a className="tooltipBtn">TR</a>
</Tooltip>
<Tooltip placement="leftTop" title={text}>
  <a className="tooltipBtn">LT</a>
</Tooltip>
<Tooltip placement="left" title={text}>
  <a className="tooltipBtn">Left</a>
</Tooltip>
<Tooltip placement="leftBottom" title={text}>
  <a className="tooltipBtn">LB</a>
</Tooltip>
<Tooltip placement="rightTop" title={text}>
  <a className="tooltipBtn">RT</a>
</Tooltip>
<Tooltip placement="right" title={text}>
  <a className="tooltipBtn">Right</a>
</Tooltip>
<Tooltip placement="rightBottom" title={text}>
  <a className="tooltipBtn">RB</a>
</Tooltip>
<Tooltip placement="bottomLeft" title={text}>
  <a className="tooltipBtn">BL</a>
</Tooltip>
<Tooltip placement="bottom" title={text}>
  <a className="tooltipBtn">Bottom</a>
</Tooltip>
<Tooltip placement="bottomRight" title={text}>
  <a className="tooltipBtn">BR</a>
</Tooltip>
```

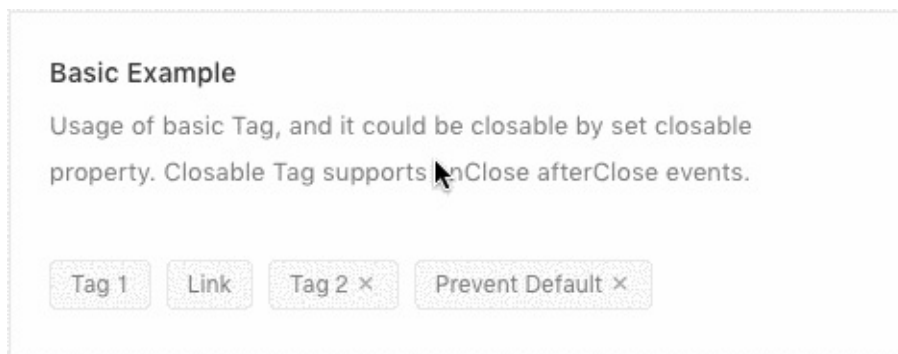
Available parameters, type and descriptions/options are down below.

Parameter	Type	Description
title	string	title for tooltip
placement	options	topLeft, top, topRight, leftTop, left, leftBottom, rightTop, right, rightBottom, bottomLeft, bottom, bottomRight

Tag

Basic tag systems are included with this template.

Basic Example



```
<Tag>Tag 1</Tag>
<Tag>
  <a
    href="https://redq.io"
  >
    Link
  </a>
</Tag>
<Tag closable onClose={this.log}>Tag 2</Tag>
<Tag closable onClose={this.preventDefault}>
  Prevent Default
</Tag>
```

Colorful Tag

Colorful Tag



```
<Tag color="#f50">#f50</Tag>
<Tag color="#2db7f5">#2db7f5</Tag>
<Tag color="#87d068">#87d068</Tag>
<Tag color="#108ee9">#108ee9</Tag>
```

Hot Tags

Hot Tags

Select your favourite topics.

Hots: Movie Books Music

```
const tagsFromServer = ['Movie', 'Books', 'Music'];
{tagsFromServer.map(tag => (
  <CheckableTag
    key={tag}
    checked={selectedTags.indexOf(tag) > -1}
    onChange={checked => this.handleChange(tag, checked)}
  >
    {tag}
  </CheckableTag>
))}
```

Add & Remove Dynamically

Add & Remove Dynamically

Generating a set of Tags by array, you can add and remove dynamically. Its based on afterClose event, which will be triggered while the close animation end.

Unremovable

Tag 2 ×

Tag 3 ×

+ New Tag

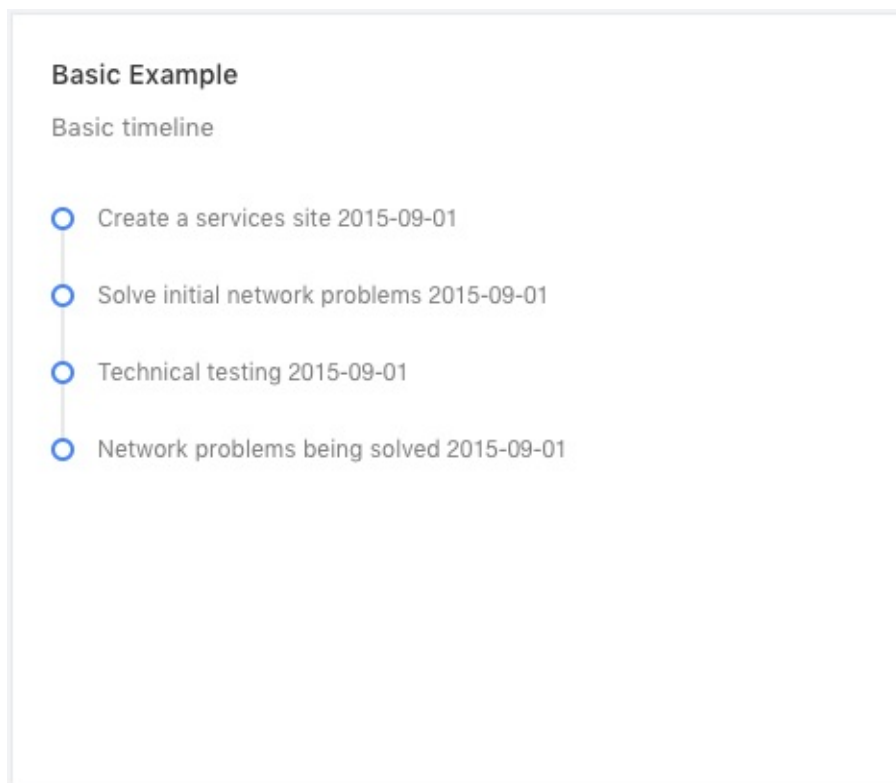
```
// in the state
state = {
  selectedTags: [],
  tags: ['Unremovable', 'Tag 2', 'Tag 3'],
  inputVisible: false,
  inputValue: '',
};
// and inside the render method
{tags.map((tag, index) => {
  const isLongTag = tag.length > 20;
  const tagElem = (
    <Tag
      key={tag}
      closable={index !== 0}
      afterClose={() => this.handleClose(tag)}
    >
      {isLongTag ? `${tag.slice(0, 20)}...` : tag}
    </Tag>
  );
  return isLongTag
    ? <Tooltip title={tag}>{tagElem}</Tooltip>
    : tagElem;
})}
{inputVisible &&
  <Input
    ref={this.saveInputRef}
    type="text"
    size="small"
    style={{ width: 78 }}
    value={inputValue}
    onChange={this.handleInputChange}
    onBlur={this.handleInputConfirm}
    onPressEnter={this.handleInputConfirm}
  />
  &&
  {!inputVisible &&
    <Button size="small" type="dashed" onClick={this.showInput}>
      + New Tag
    </Button>
  }
}
```

Available parameters, type and descriptions/options are down below.

Parameter	Type	Description
key	integer	unique identifier for tag
closable	null	removable tag
onClose	function	on close callback function
color	hex color	put hex color value e.g. #2db7f5
checked	function	on checked callback function
afterClose	function	callback function

Timeline

Basic Example




```
<Timeline>
  <Timeline.Item>
    Create a services site 2015-09-01
  </Timeline.Item>
  <Timeline.Item>
    Solve initial network problems 2015-09-01
  </Timeline.Item>
  <Timeline.Item>Technical testing 2015-09-01</Timeline.Item>
  <Timeline.Item>
    Network problems being solved 2015-09-01
  </Timeline.Item>
</Timeline>
```

Color Example

Color Example

Set the color of circles. green means completed or success status, red means warning or error, and blue means ongoing or other default status.

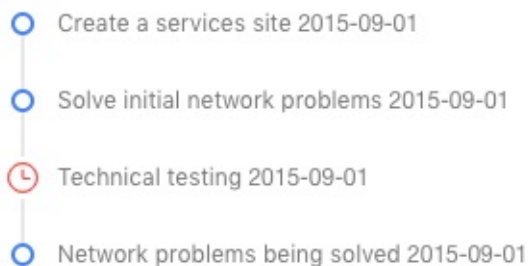


```
<Timeline>
  <Timeline.Item color="green">
    Create a services site 2015-09-01
  </Timeline.Item>
  <Timeline.Item color="green">
    Create a services site 2015-09-01
  </Timeline.Item>
  <Timeline.Item color="red">
    <p>Solve initial network problems 1</p>
    <p>Solve initial network problems 2</p>
    <p>Solve initial network problems 3 2015-09-01</p>
  </Timeline.Item>
  <Timeline.Item>
    <p>Technical testing 1</p>
    <p>Technical testing 2</p>
    <p>Technical testing 3 2015-09-01</p>
  </Timeline.Item>
</Timeline>
```

Custom Timeline

Custom

Set a node as an icon or other custom element.

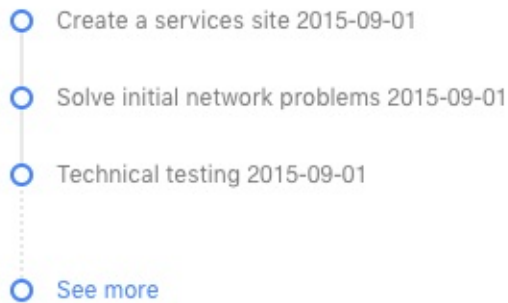
- 
- Create a services site 2015-09-01
 - Solve initial network problems 2015-09-01
 - 🕒 Technical testing 2015-09-01
 - Network problems being solved 2015-09-01

```
<Timeline>
  <Timeline.Item>
    Create a services site 2015-09-01
  </Timeline.Item>
  <Timeline.Item>
    Solve initial network problems 2015-09-01
  </Timeline.Item>
  <Timeline.Item
    dot={
      (
        <Icon
          type="clock-circle-o"
          style={{ fontSize: '16px' }}
        />
      )
    }
    color="red"
  >
    Technical testing 2015-09-01
  </Timeline.Item>
  <Timeline.Item>
    Network problems being solved 2015-09-01
  </Timeline.Item>
</Timeline>
```

Last Node

Last Node

When the timeline is incomplete and ongoing, put a ghost node at last. set {pending={true}} or {pending={a React Element}}



```
<Timeline pending={<a>See more</a>}>
  <Timeline.Item>
    Create a services site 2015-09-01
  </Timeline.Item>
  <Timeline.Item>
    Solve initial network problems 2015-09-01
  </Timeline.Item>
  <Timeline.Item>Technical testing 2015-09-01</Timeline.Item>
</Timeline>
```

Available parameters, type and descriptions for Timeline are down below.

Parameter	Type	Description
pending	data content	add some content as last node

Available parameters, type and descriptions for **Timeline.Item** are down below.

Parameter	Type	Description
color	color value	add color value here
dot	custom content	you can add custom content inside dot parameter

Dropdown

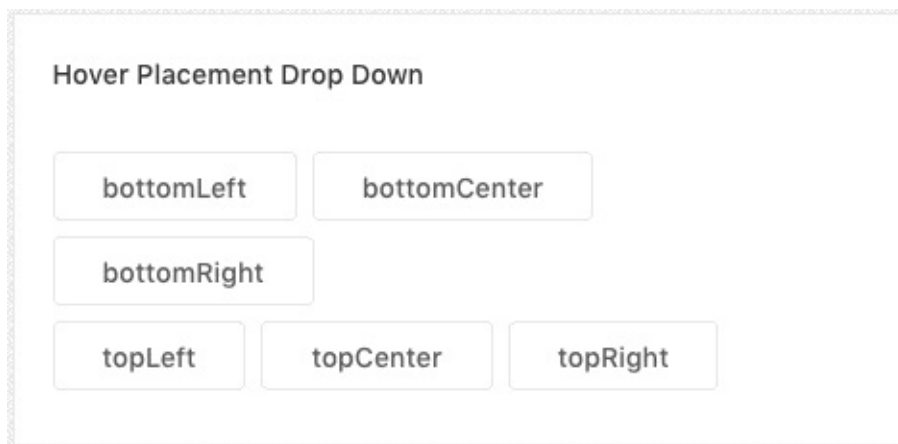
Hover Dropdown



```
// Menu component
const menuHover = (
  <Menu>
    <Menu.Item>
      <a target="_blank" rel="noopener noreferrer" href="http://
redq.io/">
        1st menu item
      </a>
    </Menu.Item>
    <Menu.Item>
      <a target="_blank" rel="noopener noreferrer" href="http://
redq.io/">
        2nd menu item
      </a>
    </Menu.Item>
    <Menu.Item>
      <a target="_blank" rel="noopener noreferrer" href="http://
redq.io/">
        3d menu item
      </a>
    </Menu.Item>
  </Menu>
);

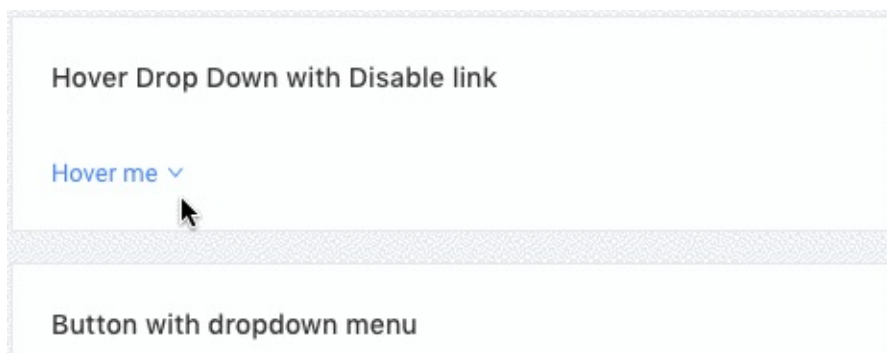
// In the render method
<Dropdown overlay={menuHover}>
  <a className="ant-dropdown-link">
    Hover me <Icon type="down" />
  </a>
</Dropdown>
```

Hover Placement Drop Down



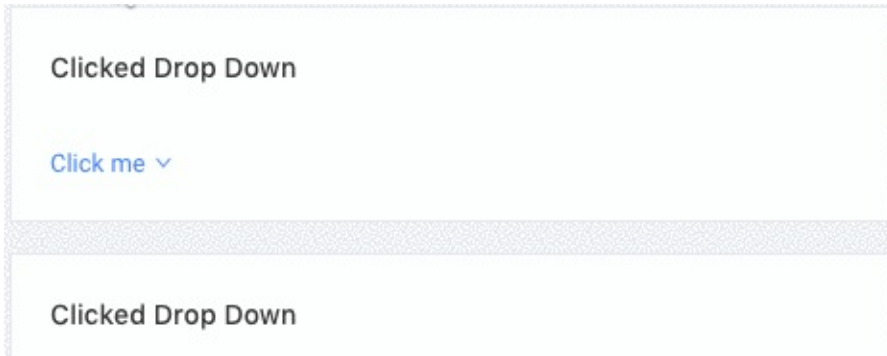
```
<Dropdown overlay={menuHover} placement="bottomLeft">
  <Button style={demoStyle}>bottomLeft</Button>
</Dropdown>
<Dropdown overlay={menuHover} placement="bottomCenter">
  <Button style={demoStyle}>bottomCenter</Button>
</Dropdown>
<Dropdown overlay={menuHover} placement="bottomRight">
  <Button style={demoStyle}>bottomRight</Button>
</Dropdown>
<br />
<Dropdown overlay={menuHover} placement="topLeft">
  <Button style={demoStyle}>topLeft</Button>
</Dropdown>
<Dropdown overlay={menuHover} placement="topCenter">
  <Button style={demoStyle}>topCenter</Button>
</Dropdown>
<Dropdown overlay={menuHover} placement="topRight">
  <Button style={demoStyle}>topRight</Button>
</Dropdown>
```

Hover Drop Down with Disable link



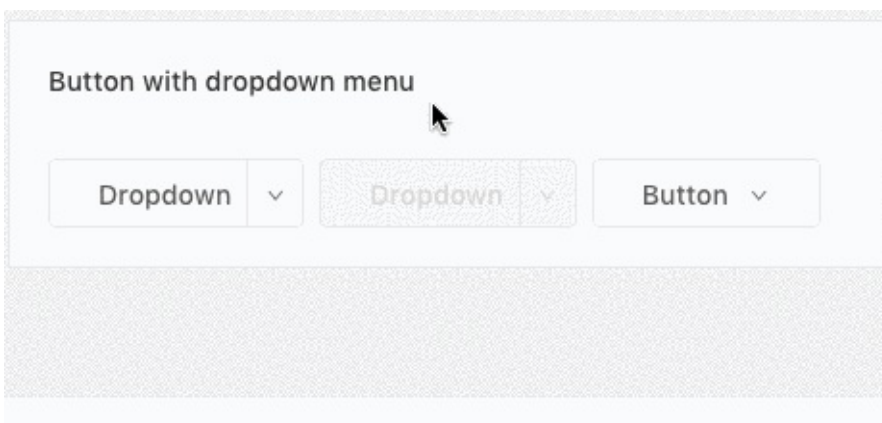
```
<Dropdown overlay={menuHoverDisable}>
  <a className="ant-dropdown-link">
    Hover me <Icon type="down" />
  </a>
</Dropdown>
```

Clicked Drop Down



```
<Dropdown overlay={menuHover} trigger={['click']}>
  <a className="ant-dropdown-link">
    Click me <Icon type="down" />
  </a>
</Dropdown>
```

Button with dropdown menu

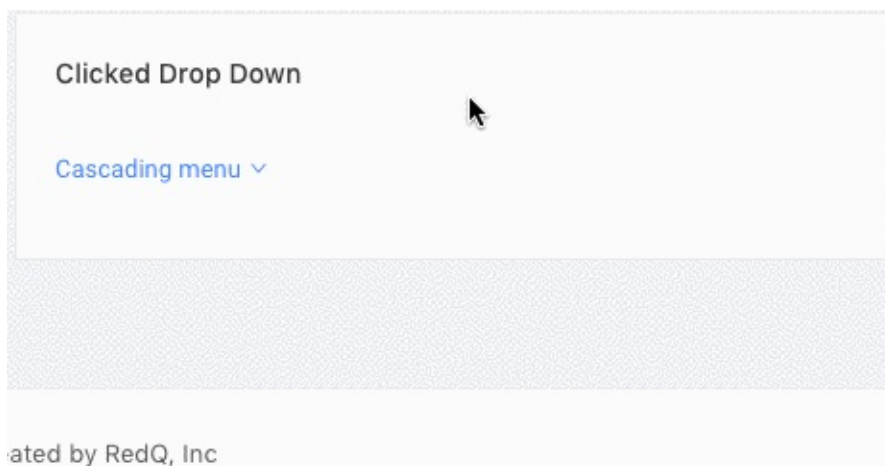



```

<Dropdown.Button
  onClick={this.handleButtonClick}
  overlay={menuClicked}
>
  Dropdown
</Dropdown.Button>
<Dropdown.Button
  onClick={this.handleButtonClick}
  overlay={menuClicked}
  disabled
  style={{ marginLeft: 8 }}
>
  Dropdown
</Dropdown.Button>
<Dropdown overlay={menuClicked}>
  <Button style={{ marginLeft: 8 }}>
    Button <Icon type="down" />
  </Button>
</Dropdown>

```

Clicked Drop Down



```

<Dropdown overlay={menuSubmenu}>
  <a className="ant-dropdown-link">
    Cascading menu <Icon type="down" />
  </a>
</Dropdown>

```

Available parameters, type and descriptions for **Dropdown** are down below.

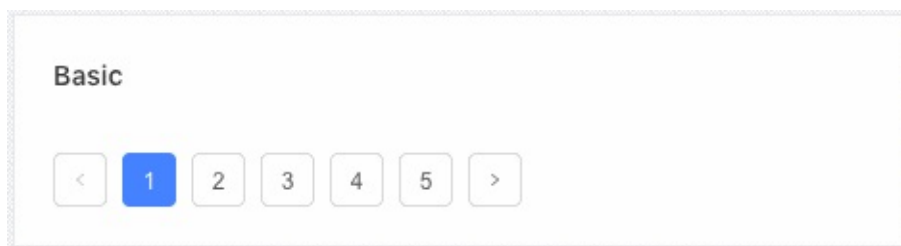
Parameter	Type	Description
overlay	component	dropdown menu component
placement	options	bottomLeft, bottomCenter, bottomRight, topLeft, topCenter, topRight
trigger	options	['click']

Available parameters, type and descriptions for **Dropdown.Button** are down below.

Parameter	Type	Description
overlay	component	dropdown menu component
style	object	css style object
disabled	null	disable button
onClick	function	callback function

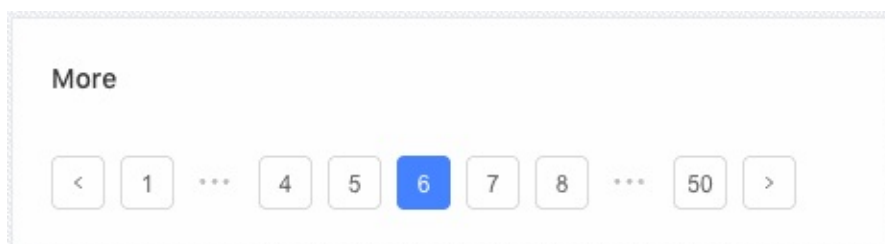
Pagination

Basic



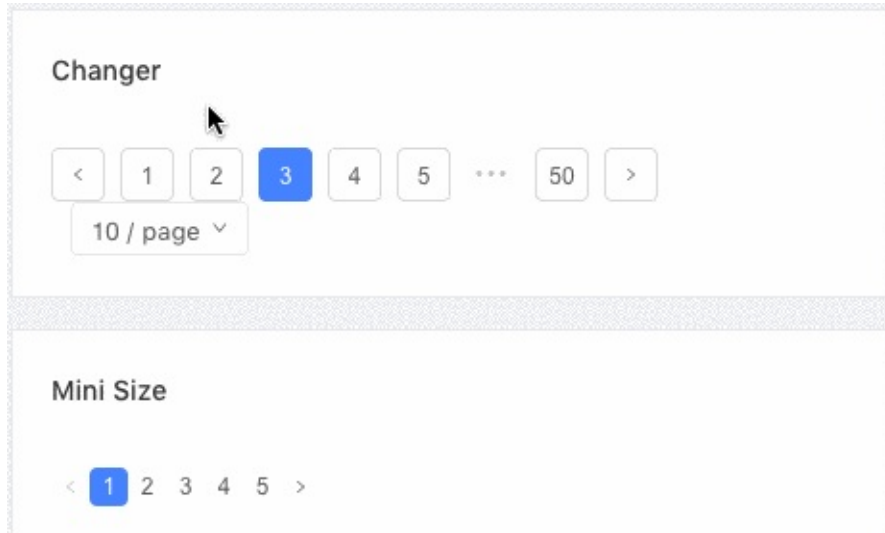
```
<Pagination defaultCurrent={1} total={50} />
```

More



```
<Pagination defaultCurrent={6} total={500} />
```

Changer



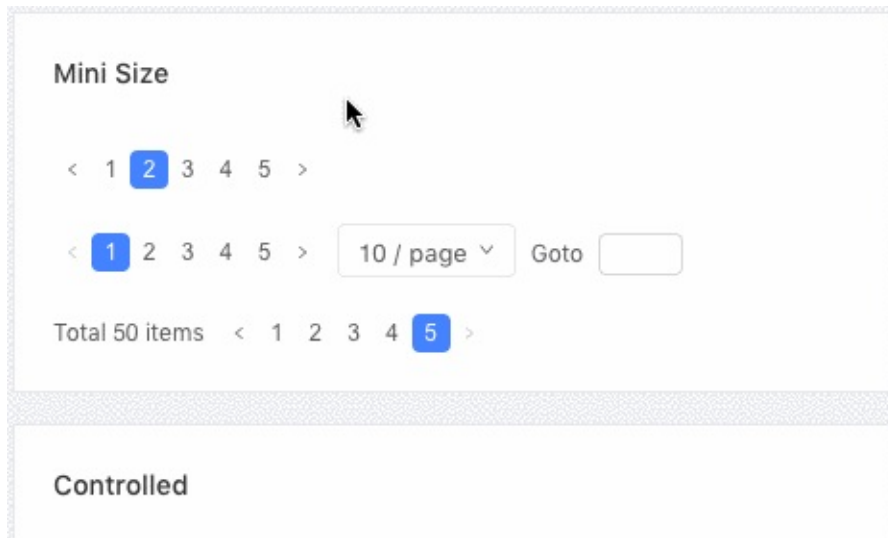
```
<Pagination
  showSizeChanger
  onShowSizeChange={this.onShowSizeChange}
  defaultCurrent={3}
  total={500}
/>
```

Jumper



```
<Pagination
  showQuickJumper
  defaultCurrent={2}
  total={500}
  onChange={this.onChange}
/>
```

Mini Size



```
<Pagination size="small" total={50} />

<Pagination
  size="small"
  total={50}
  showSizeChanger
  showQuickJumper
/>

<Pagination
  size="small"
  total={50}
  showTotal={this.showTotal}
/>
```

Simple Mode



```
<Pagination simple defaultCurrent={2} total={50} />
```

Controlled



```
<Pagination
  current={this.state.current}
  onChange={this.onChangeControlled}
  total={50}
/>
```

Total Number



```
<Pagination
  total={85}
  showTotal={total => `Total ${total} items`}
  pageSize={20}
  defaultCurrent={1}
/>

<Pagination
  total={85}
  showTotal={({total, range}) =>
    `${range[0]}-${range[1]} of ${total} items`}
  pageSize={20}
  defaultCurrent={1}
/>
```

Available parameters, type and descriptions for are down below.

Parameter	Type	Description
defaultCurrent	integer	current page number
total	integer	total no of pages
showSizeChanger	null	show/hide size changer
onShowSizeChange	function	callback function
showQuickJumper	null	show/hide quick jumper input option
size	options	pagination size. e.g. small

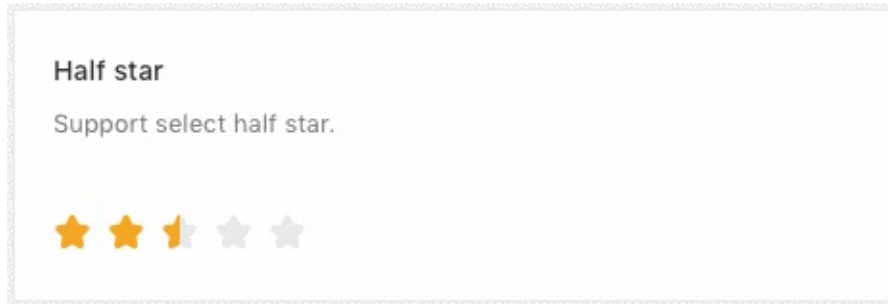
Rating

Basic Example



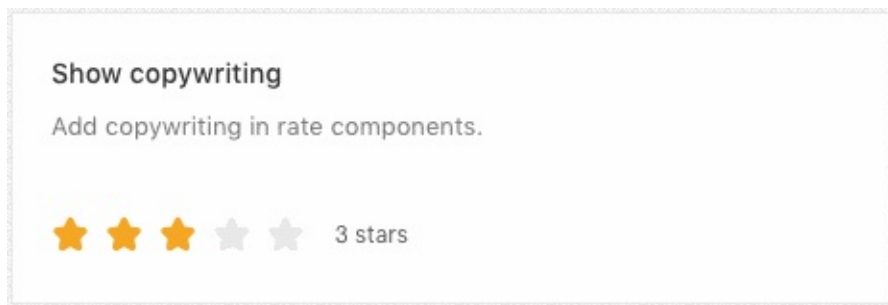
```
<Rate />
```

Half Star



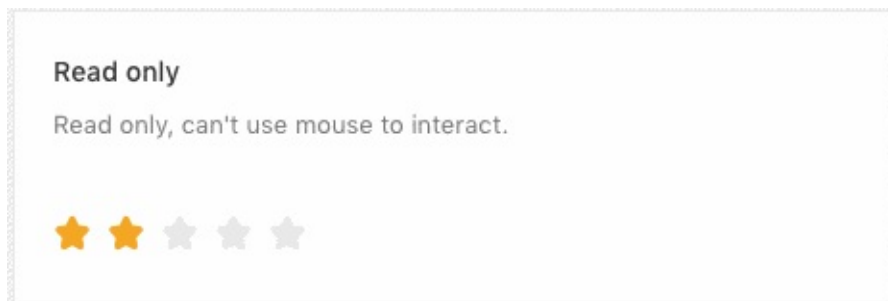
```
<Rate allowHalf defaultValue={2.5} />
```

Show copywriting



```
<Rate onChange={this.handleChange} value={value} />
```

Read Only



```
<Rate disabled defaultValue={2} />
```

Other Character

Other Character

Replace the default star to other character like alphabet, digit, iconfont or even Chinese word.



A A A A A

```
<Rate character={<Icon type="heart" />} allowHalf />
```

```
<Rate character="A" allowHalf style={{ fontSize: 36 }} />
```

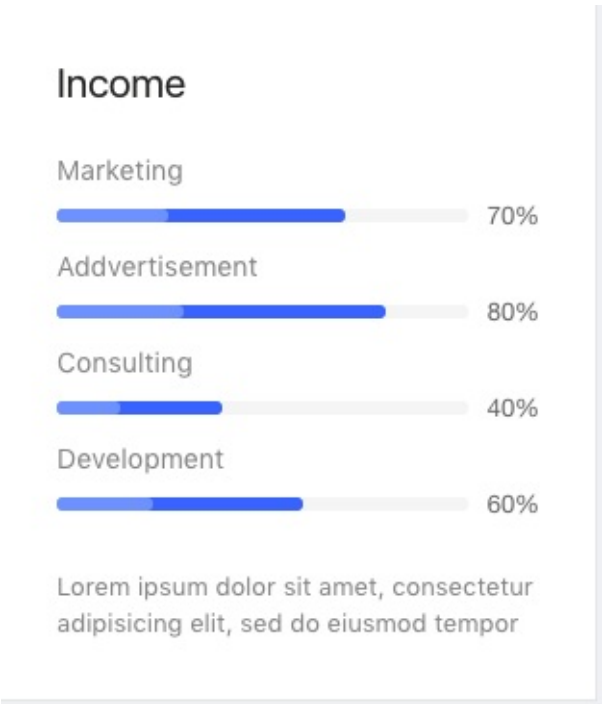
Available parameters, type and descriptions for are down below.

Parameter	Type	Description
defaultValue	float	show rating default value
allowHalf	null	allow half rating input
onChange	function	callback function
value	float	value for rating
disabled	null	read only mode
character	component	icon component
style	style object	put css object if you want to styling

Widgets

To find out the code of widgets, please go to your-apps-root-path/src/containers/widgets

ReportsWidget



You will find the code at your-apps-root-path/src/containers/widgets/report/report-widget.js.

Then, the file is imported in your-apps-root-path/src/containers/widgets/index.js file.

Code:

```
<IsoWidgetsWrapper>
  {/* Report Widget */}
  <ReportsWidget
    label="Income"
    details="Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor"
  >
    <SingleProgressWidget
      label="Marketing"
      percent={70}
      barHeight={7}
      status="active"
      info={true}
    />
    <SingleProgressWidget
      label="Addvertisement"
      percent={80}
      barHeight={7}
      status="active"
      info={true}
    />
    <SingleProgressWidget
      label="Consulting"
      percent={40}
      barHeight={7}
      status="active"
      info={true}
    />
    <SingleProgressWidget
      label="Development"
      percent={60}
      barHeight={7}
      status="active"
      info={true}
    />
  </ReportsWidget>
</IsoWidgetsWrapper>
```

Table Widget

First Name	Last Name	City	Street
Emelia	Gislason	Lake Zelda	Kulas Shoals
Cloyd	Armstrong	East Pierce	Lyla Heights
Rahul	Funk	Sibylside	Jolie Shoals
Hilbert	Langosh	Anaishshire	Sim Station
Cloyd	Wilderman	North Brad	Ruecker Turnpike

Code:

```
<IsoWidgetsWrapper>
  <IsoWidgetBox>
    {/* TABLE */}
    <TableViews.SimpleView
      tableInfo={tableInfos[0]}
      dataList={tableDataList}
    />
  </IsoWidgetBox>
</IsoWidgetsWrapper>
```

StickerWidget**Code:**

```
<IsoWidgetsWrapper>
  {/* Sticker Widget */}
  <StickerWidget
    number="210"
    text="Unread Email"
```

```
        icon="ion-email-unread"
        fontColor="#ffffff"
        bgColor="#7266BA"
    />
</IsoWidgetsWrapper>
</Col>

<Col md={6} sm={12} xs={24} style={colStyle}>
    <IsoWidgetsWrapper>
        {/* Sticker Widget */}
        <StickerWidget
            number="1749"
            text="Image Upload"
            icon="ion-android-camera"
            fontColor="#ffffff"
            bgColor="#42A5F6"
        />
    </IsoWidgetsWrapper>
</Col>

<Col md={6} sm={12} xs={24} style={colStyle}>
    <IsoWidgetsWrapper>
        {/* Sticker Widget */}
        <StickerWidget
            number="3024"
            text="Total Message"
            icon="ion-chatbubbles"
            fontColor="#ffffff"
            bgColor="#7ED320"
        />
    </IsoWidgetsWrapper>
</Col>

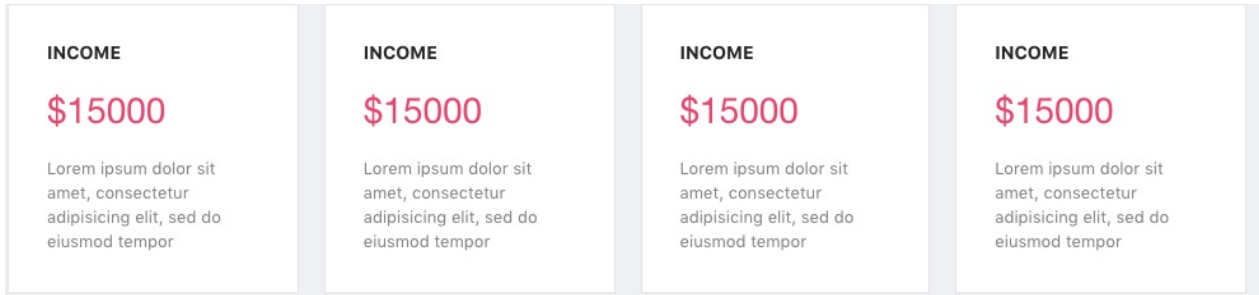
<Col md={6} sm={12} xs={24} style={colStyle}>
    <IsoWidgetsWrapper>
        {/* Sticker Widget */}
        <StickerWidget
            number="54"
            text="Orders Post"
            icon="ion-android-cart"
```

```

        fontColor="#ffffff"
        bgColor="#F75D81"
    />
</IsoWidgetsWrapper>

```

SaleWidget



Code:

```

<IsoWidgetsWrapper>
  {/* Sale Widget */}
  <SaleWidget
    label="Income"
    price="$15000"
    fontColor="#F75D81"
    details="Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor"
  />
</IsoWidgetsWrapper>
</Col>

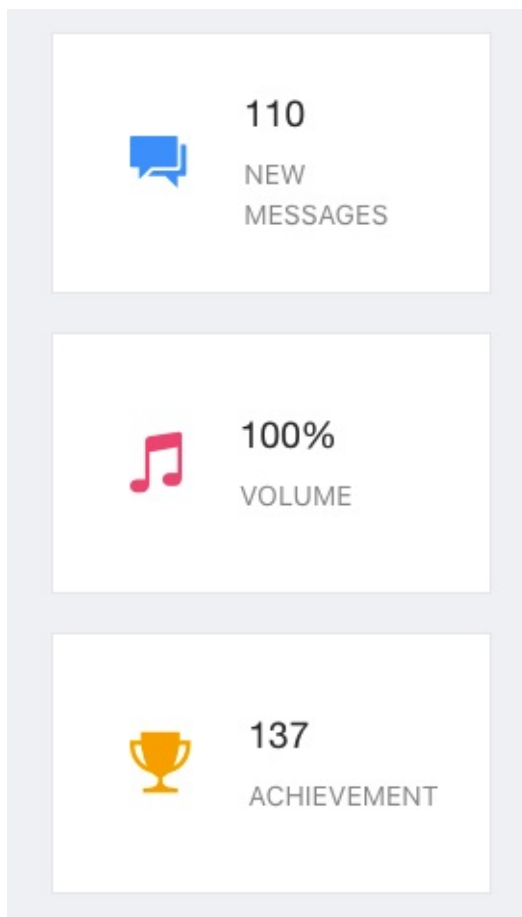
<Col md={6} sm={12} xs={24} style={colStyle}>
  <IsoWidgetsWrapper>
    {/* Sale Widget */}
    <SaleWidget
      label="Income"
      price="$15000"
      fontColor="#F75D81"
      details="Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor"
    />
  </IsoWidgetsWrapper>
</Col>

```

```
<Col md={6} sm={12} xs={24} style={colStyle}>
  <IsoWidgetsWrapper>
    {/* Sale Widget */}
    <SaleWidget
      label="Income"
      price="$15000"
      fontColor="#F75D81"
      details="Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor"
    />
  </IsoWidgetsWrapper>
</Col>

<Col md={6} sm={12} xs={24} style={colStyle}>
  <IsoWidgetsWrapper>
    {/* Sale Widget */}
    <SaleWidget
      label="Income"
      price="$15000"
      fontColor="#F75D81"
      details="Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor"
    />
  </IsoWidgetsWrapper>
```

Card Widget



Code:


```
<IsoWidgetsWrapper gutterBottom={20}>
  {/* Card Widget */}
  <CardWidget
    icon="ion-android-chat"
    iconcolor="#42A5F5"
    number="110"
    text="New Messages"
  />
</IsoWidgetsWrapper>

<IsoWidgetsWrapper gutterBottom={20}>
  {/* Card Widget */}
  <CardWidget
    icon="ion-music-note"
    iconcolor="#F75D81"
    number="100%"
    text="Volume"
  />
</IsoWidgetsWrapper>

<IsoWidgetsWrapper>
  {/* Card Widget */}
  <CardWidget
    icon="ion-trophy"
    iconcolor="#FEAC01"
    number="137"
    text="Achievement"
  />
</IsoWidgetsWrapper>
```


ProgressWidget

Download




90% Responsive

Support



90% Responsive

Download



90% Responsive

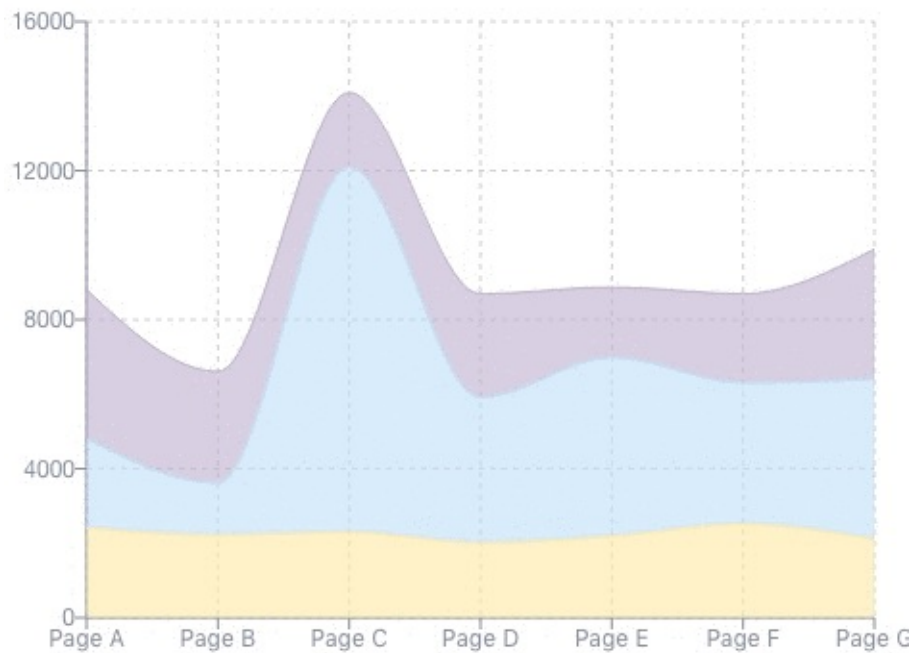
Code:

```
<IsoWidgetsWrapper gutterBottom={20}>
  {/* Progress Widget */}
  <ProgressWidget
    label="Download"
    icon="ion-archive"
    iconcolor="#222222"
    details="90% Responsive"
    percent={50}
    barHeight={7}
    status="active"
  />
</IsoWidgetsWrapper>
```

```
<IsoWidgetsWrapper gutterBottom={20}>
  {/* Progress Widget */}
  <ProgressWidget
    label="Support"
    icon="ion-pie-graph"
    iconcolor="#222222"
    details="90% Responsive"
    percent={80}
    barHeight={7}
    status="active"
  />
</IsoWidgetsWrapper>
```

```
<IsoWidgetsWrapper>
  {/* Progress Widget */}
  <ProgressWidget
    label="Download"
    icon="ion-android-download"
    iconcolor="#222222"
    details="90% Responsive"
    percent={40}
    barHeight={7}
    status="active"
  />
</IsoWidgetsWrapper>
```

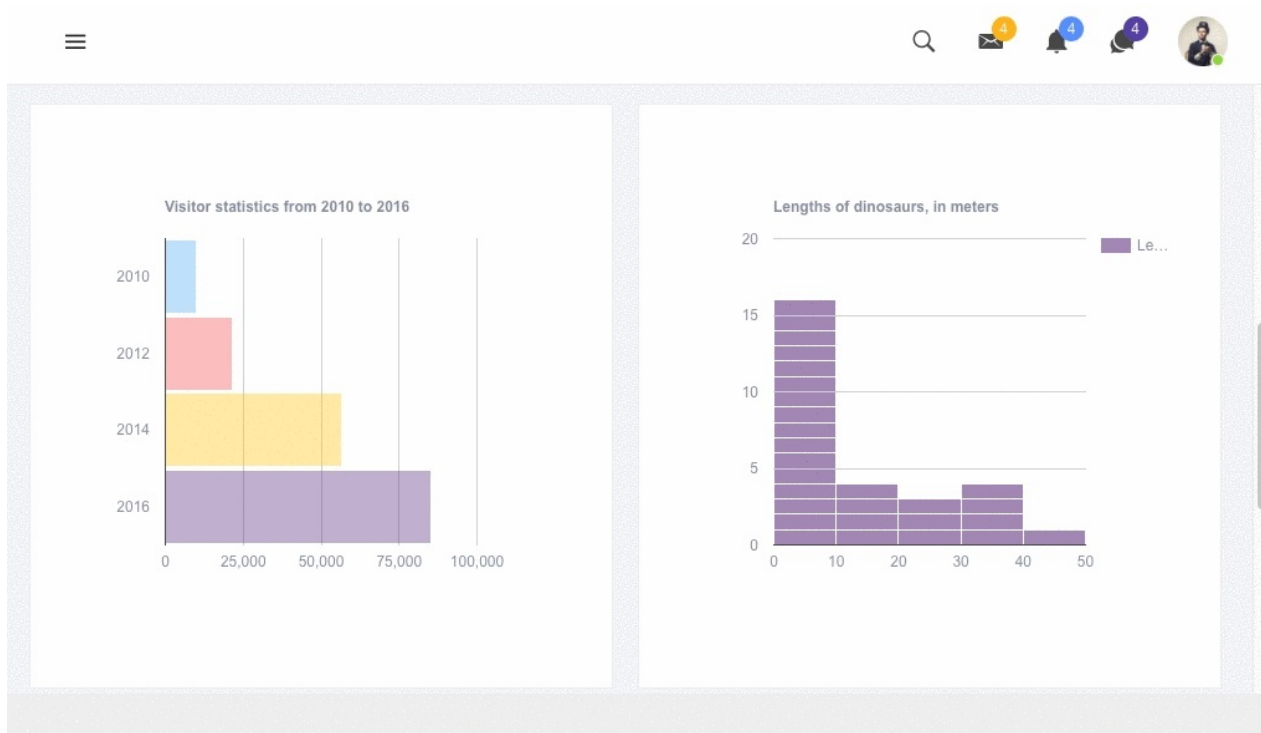
StackedAreaChart



Code:

```
<IsoWidgetsWrapper>  
  <IsoWidgetBox height={455}>  
    <StackedAreaChart {...stackConfig} />  
  </IsoWidgetBox>  
</IsoWidgetsWrapper>
```

GoogleChart

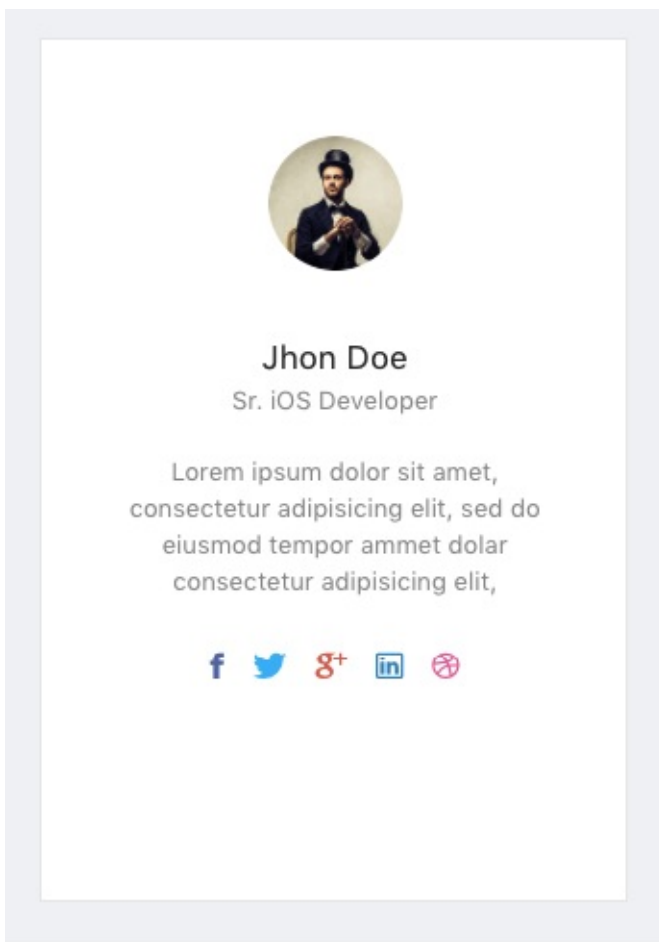


Code:

```
<IsoWidgetsWrapper>
  <IsoWidgetBox height={470}>
    <GoogleChart
      {...googleChartConfigs.BarChart}
      chartEvents={chartEvents}
    />
  </IsoWidgetBox>
</IsoWidgetsWrapper>

<IsoWidgetsWrapper>
  <IsoWidgetBox height={470}>
    <GoogleChart {...googleChartConfigs.Histogram} /
  >
  </IsoWidgetBox>
</IsoWidgetsWrapper>
```

VCard

**Code:**

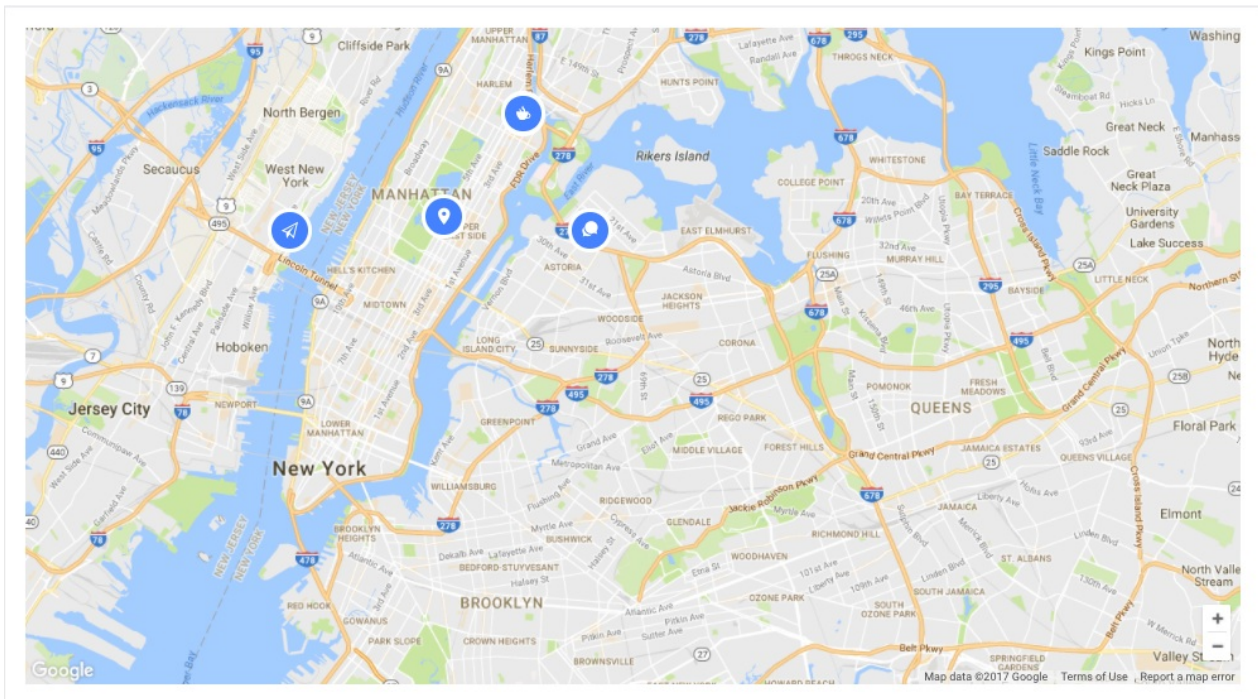
```
<IsoWidgetsWrapper>
    { /* VCard Widget */ }
    <VCardWidget
        style={{ height: '450px' }}
        src={userpic}
        alt="Jhon"
        name="Jhon Doe"
        title="Sr. iOS Developer"
        description="Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor ammet dolar consectetur adipisicing elit,"
    >
        <SocialWidget>
            <SocialProfile
                url="#"
                icon="ion-social-facebook"
                iconcolor="#3b5998"
            />
        </SocialWidget>
    </VCardWidget>
</IsoWidgetsWrapper>
```

```
        <SocialProfile
            url="#"
            icon="ion-social-twitter"
            iconcolor="#00aced"
        />
        <SocialProfile
            url="#"
            icon="ion-social-googleplus"
            iconcolor="#dd4b39"
        />
        <SocialProfile
            url="#"
            icon="ion-social-linkedin-outline"
            iconcolor="#007bb6"
        />
        <SocialProfile
            url="#"
            icon="ion-social-dribbble-outline"
            iconcolor="#ea4c89"
        />
    </SocialWidget>
</VCardWidget>
</IsoWidgetsWrapper>
```

Google map

Folder path: /src/containers/map/GoogleMap

If you want to render a map like the following image



Then The code should be like this.

```
<GoogleMapReact
  defaultZoom={this.props.zoom}
  bootstrapURLKeys={{
    key: API_KEY,
  }}
  onBoundsChange={this.boundsChange}
  onChildClick={this.childClick}
  center={this.state.center}
  distanceToMouse={this.distanceToMouse}
>
  {this.state.posts.length ? this.state.posts.map(this
.allMarkers) : null}
</GoogleMapReact>
```

Where,

Parameter	Type	Keys/Parameters	Description
center	Object.	lat and lng	Center of the Map
defaultZoom	Integer		Default zoom level
onChildClick	Function	key, childProps	Marker click callback functionbootstrapURLKe
bootstrapURLKeys	Object	API_KEY	Google Map API Key
onBoundsChange	Function	key, childProps	Callback function for changing map bound

For Custom marker design, You can edit the following part of code of `marker.js`

```
<div className="isoMarkerInfowindow">
  <div className="">
    <div>
      <i className={` ${markerIcon}`}></i>
    </div>
  </div>
  <div>
    {props.infoWindow !== null ? openInfowindow() : null}
  </div>
</div>
```

For Custom Infowindow Design you can edit the following code portion of `marker.js`.


```
const openInfoWindow = () => {
  const { infoWindow, item } = props;
  if (item.ID !== infoWindow.ID)
    return;
  return (
    <div className="isoInfoWindow">
      <div className="isoInfoWindowImage">
        <img alt="#" src={infoWindow.img}/>
      </div>
      <div className="isoInfoWindowDetails">
        <h3 className="isoHeading">{infoWindow.title}</h3>
        <p className="isoLocation">{infoWindow.location}</p>
      </div>
      <button className="windowCloseBtn" onClick={() => props
.closeInfoWindow()}>
        <i className="ion-android-close"></i>
      </button>
    </div>
  )
};
```

You can insert post data in the file name config.js

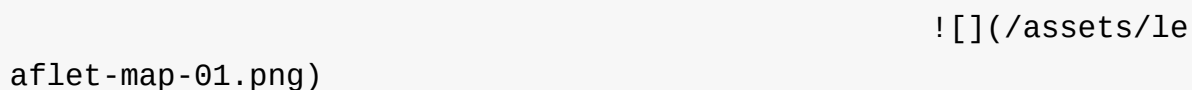
An array of posts will have to be provided. A single Post will have the following keys.

Keys	Type	Parameters	Descriptio
ID	Integer		
title	String		Title Show on Info Window
location	String		Location shown on Info Windo
img	String		Image U
lat	Integer		Latitude
lng	Integer		Longitude
marker	Object	borderStyle,borderColor,fontFamily,iconClass	Styling of each marker

Leaflet map

Folder path: /src/containers/map/Leaflet

If you want to render a map like the following image



Then The code should be like this.

```
const { L } = window;
const map = L.map(element).setView(
  mapboxConfig.center,
  !isNaN(mapboxConfig.defaultZoom) ? mapboxConfig.defaultZoom : 13
);
const osmAttr = '&copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors';
L.tileLayer(mapboxConfig.tileLayer, {
  maxZoom: !isNaN(mapboxConfig.maxZoom) ? mapboxConfig.maxZoom : 18,
  attribution: osmAttr
}).addTo(map);

basicMarkers.map(singleMarker => {
  return L.marker(singleMarker.position)
    .addTo(map)
    .bindPopup(singleMarker.popupText);
});
```

Where,

Parameter	Type	Keys/Parameters	Description
center	Object.	lat and lng	Center of the Map
defaultZoom	Integer		Default zoom level
osmAttr	String		default footer text

For Custom html marker design, You can edit the following part of code of

```
const { L } = window;
const map = L.map(element).setView(
  mapboxConfig.center,
  !isNaN(mapboxConfig.defaultZoom) ? mapboxConfig.defaultZoom : 13
);
const osmAttr = '&copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors';

L.tileLayer(mapboxConfig.tileLayer, {
  maxZoom: !isNaN(mapboxConfig.maxZoom) ? mapboxConfig.maxZoom : 18,
  attribution: osmAttr
}).addTo(map);

customHtmlMarker.map(singleMarker => {
  const htmlIcon = L.divIcon({
    className: singleMarker.className,
    popupAnchor: [15, -15], // point from which the popup should open relative to the iconAnchor
    html: singleMarker.html,
  });
  return L.marker(singleMarker.position, { icon: htmlIcon })
    .addTo(map)
    .bindPopup(singleMarker.popupText);
});
```

For Custom ICON marker design, You can edit the following part of code of

```

    const { L } = window;
    const map = L.map(element).setView(
        mapboxConfig.center,
        !isNaN(mapboxConfig.defaultZoom) ? mapboxConfig.defaultZoom : 13
    );
    const osmAttr = '&copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors';

    L.tileLayer(mapboxConfig.tileLayer, {
        maxZoom: !isNaN(mapboxConfig.maxZoom) ? mapboxConfig.maxZoom : 18,
        attribution: osmAttr
    }).addTo(map);

    customIconMarkers.map(singleMarker => {
        var customIcon = L.icon({
            iconUrl: singleMarker.iconUrl,
            shadowUrl: singleMarker.shadowUrl,
            iconSize: [40, 40], // size of the icon
            shadowSize: [40, 40], // size of the shadow
            // iconAnchor: [22, 94], // point of the icon which will
            // correspond to marker's location
            shadowAnchor: [12, 20], // the same for the shadow
            popupAnchor: [0, -20], // point from which the popup should
            // open relative to the iconAnchor
        });
        return L.marker(singleMarker.position, { icon: customIcon })
            .addTo(map)
            .bindPopup(singleMarker.popupText);
    });

```

For marker cluster, You can edit the following part of code of

```

const { L } = window;
const map = L.map(element).setView(
  mapboxConfig.center,
  !isNaN(mapboxConfig.defaultZoom) ? mapboxConfig.defaultZoom : 13
);
const osmAttr = '&copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors';
L.tileLayer(mapboxConfig.tileLayer, {
  maxZoom: !isNaN(mapboxConfig.maxZoom) ? mapboxConfig.maxZoom : 18,
  attribution: osmAttr
}).addTo(map);

let markers = L.markerClusterGroup();

markerCluster.map(singleMarker => {
  var customIcon = L.icon({
    iconUrl: singleMarker.iconUrl,
    iconSize: [40, 40], // size of the icon
    popupAnchor: [0, -20], // point from which the popup should open relative to the iconAnchor
  });
  return markers.addLayer(
    L.marker(singleMarker.position, { icon: customIcon })
      .bindPopup(singleMarker.popupText)
      .openPopup()
  );
});
map.addLayer(markers);

```

If you want to show shortest path between two makers. you can use the following code

```

const { L } = window;
const map = L.map(element).setView(
  mapboxConfig.center,
  !isNaN(mapboxConfig.defaultZoom) ? mapboxConfig.defaultZoom : 13
);
const osmAttr = '&copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors';
L.tileLayer(mapboxConfig.tileLayer, {
  maxZoom: !isNaN(mapboxConfig.maxZoom) ? mapboxConfig.maxZoom : 18,
  attribution: osmAttr
}).addTo(map);
try {
  L.Routing
    .control({
      waypoints: [
        L.latLng(40.72143, -74.05729),
        L.latLng(40.6943, -74.074201),
      ],
      routeWhileDragging: true,
    })
    .addTo(map);
} catch (e) {}

```

The following are the screen shot of demos containing a custom ICON marker and the other is custom HTML marker.

```




)

```

Isotope

To find out the code of Isotope, please go to your-apps-root-path/src/containers/Isotope

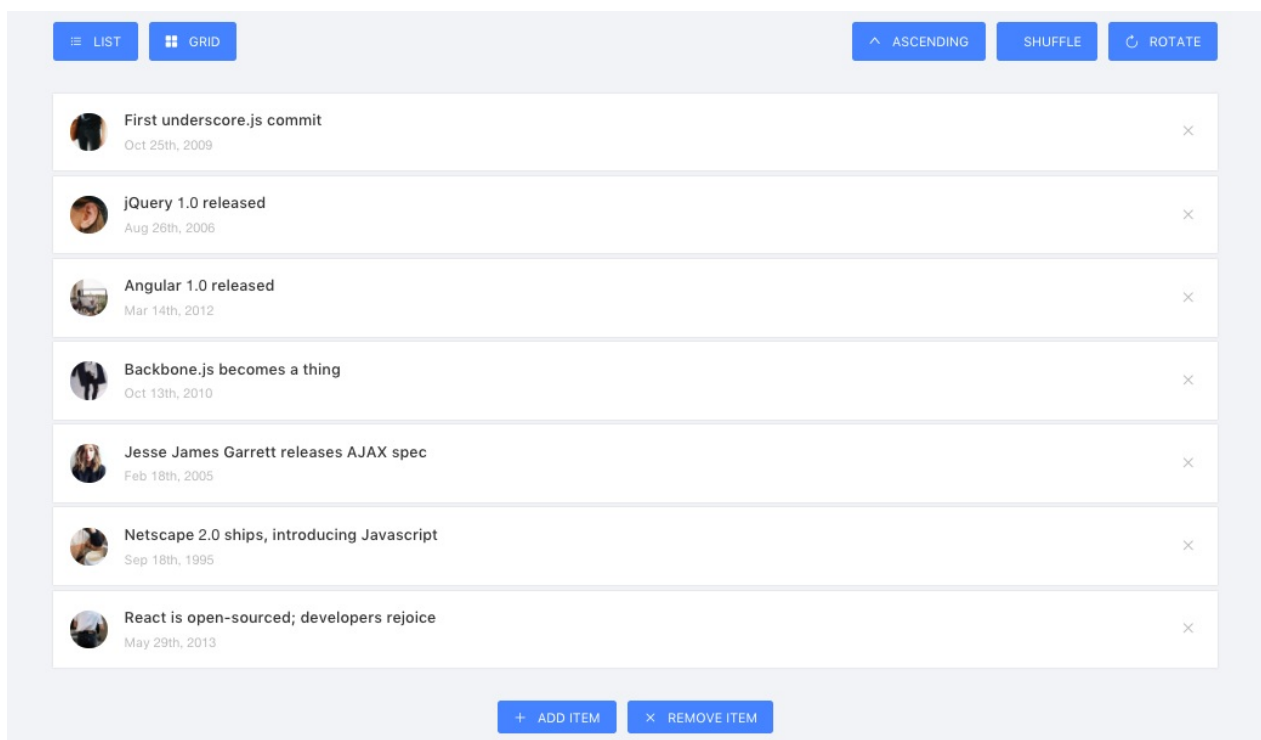
View:

There are two kind of view for Isotope .

- List View
- Grid View

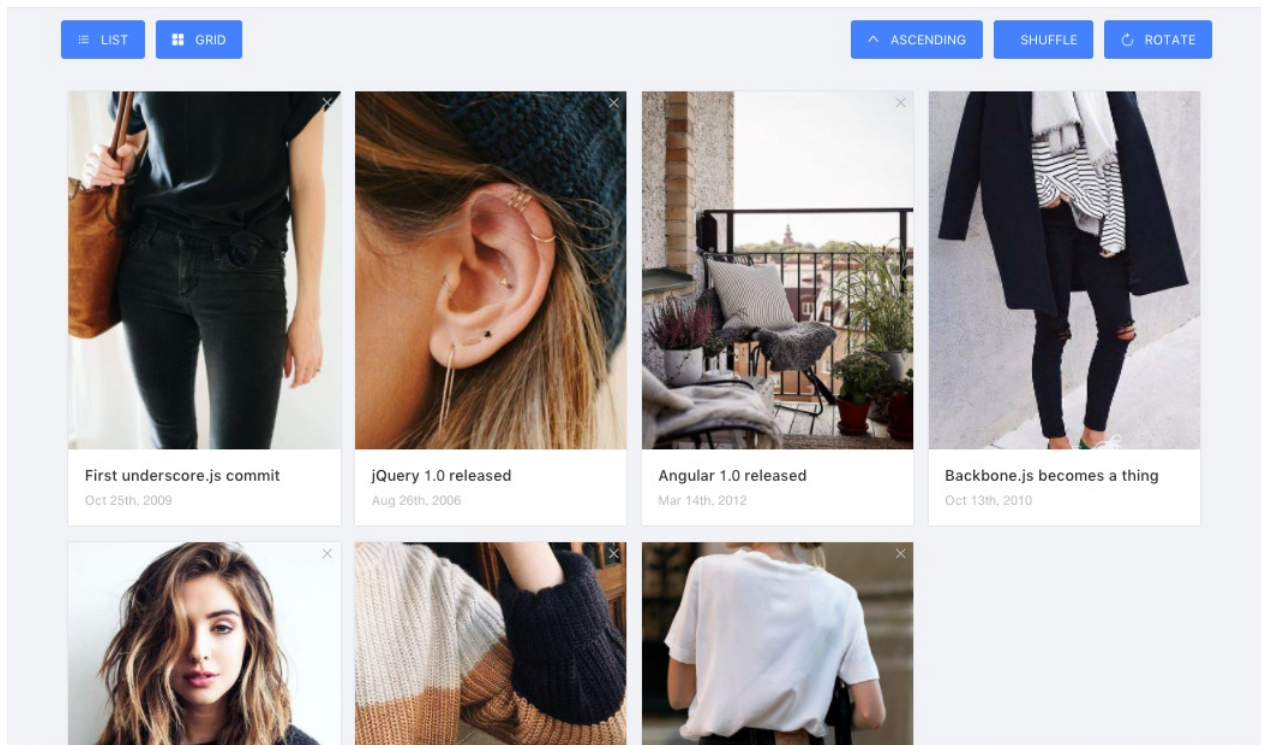
List View:

This is the List view for Isotope .



Grid View:

This is the Grid View of Isotope .



Code:

The Isotope folder is divided into three parts, to make it understand better follow the below steps,

1. index.js (main file)
2. config.js (data config)
3. Toggle.js (toggle component)

In the `index.js` file you will get all the necessary code.

You can get the code for `Grid` or `List` item of the `Isotope` from the `renderArticles()` function.

and here is the function code,

```
<ListItem
  key={article.id}
  view={this.state.view}
  index={i}
  clickHandler={throttle(
    () => this.moveArticle('articles', 'removedArticles', article.id),
    800,
  )}
  {...article}
/>
```

to get the code for ListItem copy the below code

```
class ListItem extends Component {
  render() {
    const listClass = `isoSingleCard card ${this.props.view}`;
    const style = { zIndex: 100 - this.props.index };

    return (
      <li id={this.props.id} className={listClass} style={style}
    >
      <div className="isoCardImage">
        <img alt="#" src={process.env.PUBLIC_URL + this.props.
img} />
      </div>
      <div className="isoCardContent">
        <h3 className="isoCardTitle">{this.props.desc}</h3>
        <span className="isoCardDate">
          {moment(this.props.timestamp).format('MMM Do, YYYY')}
        </span>
      </div>
      <button className="isoDeleteBtn" onClick={this.props.cli
ckHandler}>
        <Icon type="close" />
      </button>
    </li>
    );
  }
}
```

Options:

In the Isotope you will get several Necessary Options.

1. List or Grid Toggle.
2. Ascending or Descending
3. Shuffle
4. Rotate
5. Add Item

6. Remove Item
7. Animation

List or Grid Toggle:

The List or Grid Toggle uses the `<Toggle/>` component from the `** Toggle.js **` file. and here is the code for List or Grid Toggle from `** index.js **` file`

```
<Toggle
  clickHandler={this.toggleList}
  text="List"
  icon="bars"
  active={this.state.view === 'list'}
/>

<Toggle
  clickHandler={this.toggleGrid}
  text="Grid"
  icon="appstore"
  active={this.state.view === 'grid'}
/>
```

the respective handler function can be found from `index.js` file.

Ascending or Descending, Shuffle & Rotate:

Same as the previous List or Grid Component, Ascending or Descending, Shuffle & Rotate also uses the Toggle component from `index.js` file.

```
<Toggle
  clickHandler={this.toggleSort}
  text={this.state.order === 'asc' ? 'Ascending' : 'Descending'}
  icon={this.state.order === 'asc' ? 'up' : 'down'}
  active={this.state.sortingMethod === 'chronological'}
/>

<Toggle
  clickHandler={this.sortShuffle}
  text="Shuffle"
  icon="random"
  active={this.state.sortingMethod === 'shuffle'}
/>

<Toggle
  clickHandler={this.sortRotate}
  text="Rotate"
  icon="reload"
  active={this.state.sortingMethod === 'rotate'}
/>
```

Add Item or Remove Item:

At the end of the `Isotope` we have added the Item add or Remove functionality in the `index.js` file

```
<Toggle
  clickHandler={() =>
    this.moveArticle('removedArticles', 'articles')}
  text="Add Item"
  icon="plus"
  active={this.state.removedArticles.length > 0}
/>

<Toggle
  clickHandler={() =>
    this.moveArticle('articles', 'removedArticles')}
  text="Remove Item"
  icon="close"
  active={this.state.articles.length > 0}
/>
```

Animation:

Flipmove Component been used to create the Animation in the Isotope Layout.

```
<FlipMove
  staggerDurationBy="30"
  duration={500}
  enterAnimation={this.state.enterLeaveAnimation}
  leaveAnimation={this.state.enterLeaveAnimation}
  typeName="ul"
/>
```

Page Components

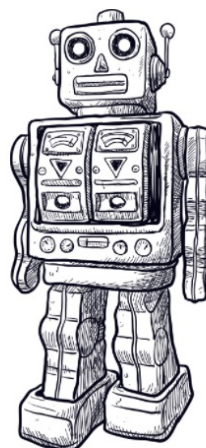
1. 404 Page

404

Looks like you've got lost

The page you're looking for doesn't exist or has been moved.

[BACK HOME](#)



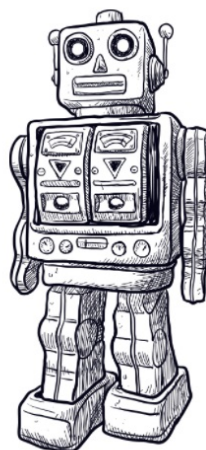
2. 500 Page

500

Internal Server Error

Something went wrong. Please try again later.

[BACK HOME](#)



3. Sign-in Page



ISOMORPHIC

☐ Remember me

Sign in

Sign in with Facebook

Sign in with Google Plus

Forgot password

[Create an Isomorphic account](#)

4. Sign-up Page



ISOMORPHIC

☐ I agree with terms and conditons

Sign up

[Already have an account? Sign in.](#)

5. Forget Password Page



ISOMORPHIC

Forgot Password?
Enter your email and we send you a reset link.

Email

Send request

6. Reset Password



ISOMORPHIC

Reset Password
Enter new password and confirm it.

New Password

Confirm Password

Save

7. Invoice Page

Invoice

Print Invoice

Invoice Info

#1942784

Order Status: Pending

Order date: June 23, 2017

Bill From

REDQ Inc.
redq@company.com

405 Mulberry Rd, Mc Grady,
NC, 28649

Fax: +0(863) 228-7064
Phone: +(740) 927-9284

Bill To

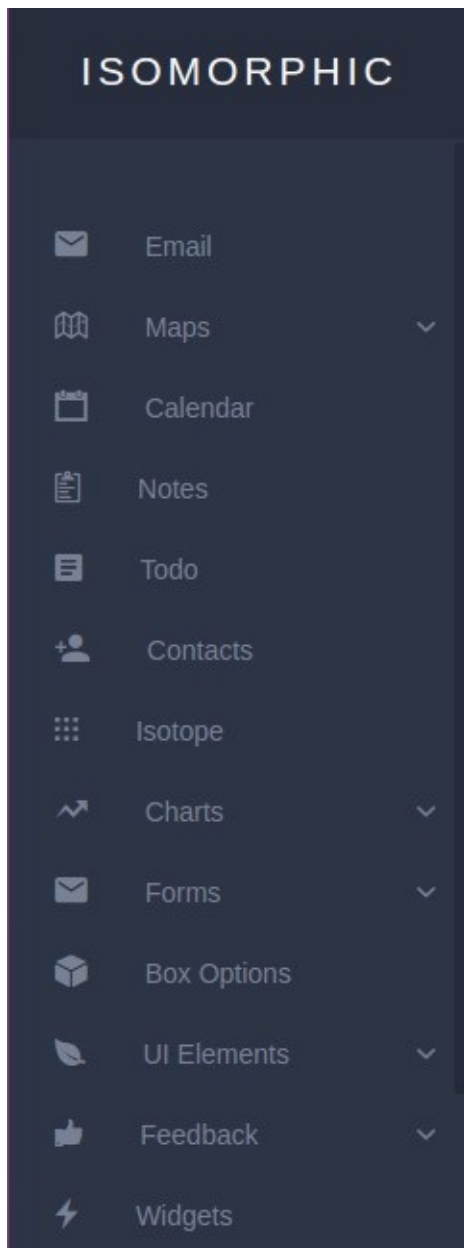
Pineapple Inc.
pineapple@company.com

86781 547th Ave, Osmond,
NE, 68765

Phone: +(402) 748-3970

#	Item Name	Costs	Qty	Price
1	A box of happiness	200	14	\$2800

Sidebar:



Code: Add the code below in `src/containers/Sidebar/Sidebar.js` in between `<Menu>` Component. to add a new menu item in the sidebar.

```
// Add New Menu
<Menu.Item key="email">
  <Link to={`${url}/newmenu`} >
    <span className="isoMenuHolder">
      <i className="ion-android-mail" />
      <span className="nav-text">New Menu</span>
    </span>
  </Link>
</Menu.Item>
```

For Routing the menu now add the below code to the

`src/containers/App/AppRouter.js` file. and add a New Component which will render when you click on the menu.

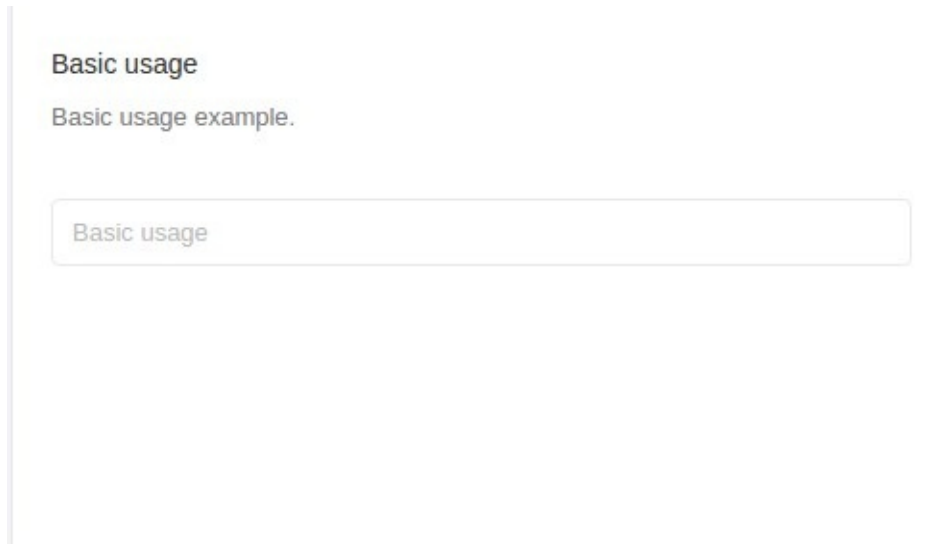
Code:

```
//menu routes
<Route
  exact
  path={`${url}/mailbox`}
  component={asyncComponent(() => import('path to NewMenuComponent'))}
/>
```

Form Elements

Input Form:

Basic Form:



Basic usage

Basic usage example.

Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box title="Basic usage" subtitle="Basic usage example.">
    <ContentHolder>
      <Input placeholder="Basic usage" />
    </ContentHolder>
  </Box>
</Col>
```

File Path: `your-dash-app-root/src/containers/Forms/Input/index.js`

Form with Different Size:

Three sizes of Input

There are three sizes of an Input box: large (42px), default (35px) and small (30px). Note: Inside of forms, only the large size is used.

Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box
    title="Three sizes of Input"
    subtitle="There are three sizes of an Input box: large (42px)
) `default (35px) and small (30px). Note: Inside of forms, only
the large size is used."
  >
    <ContentHolder>
      <Input
        size="large"
        placeholder="large size"
        style={{ marginBottom: '15px' }}
      />
      <Input
        placeholder="default size"
        style={{ marginBottom: '15px' }}
      />
      <Input size="small" placeholder="small size" />
    </ContentHolder>
  </Box>
</Col>
```

File Path: `your-dash-app-root/src/containers/Forms/Input/index.js`

Input Group:

Input Group

Input.Group example Note: You dont need Col to control the width in the compact mode.

0571

26888888

0571

26888888

Zhejiang

Xihu District, Hangzhou

Option1

input content

input content

Select date

Option1-1

Option2-2

Between

Minimum

~

Maximum

Sign Up

Email

Code:

```
<Row style={rowStyle} gutter={gutter} justify="start">
  <Col md={24} sm={24} xs={24} style={colStyle}>
    <Box
      title="Input Group"
      subtitle="Input.Group example Note: You dont need Col to c
ontrol the width in the compact mode."
    >
      <ContentHolder>
        <InputGroup size="large" style={{ marginBottom: '15px' }
}>
          <Col span="4">
            <Input defaultValue="0571" />
          </Col>
          <Col span="8">
            <Input defaultValue="26888888" />
          </Col>
        </InputGroup>

        <InputGroup compact style={{ marginBottom: '15px' }}>
          <Input style={{ width: '20%' }} defaultValue="0571" />
          <Input style={{ width: '30%' }} defaultValue="26888888
" />
```

```

</InputGroup>

<InputGroup compact style={{ marginBottom: '15px' }}>
  <Select defaultValue="Zhejiang">
    <Option value="Zhejiang">Zhejiang</Option>
    <Option value="Jiangsu">Jiangsu</Option>
  </Select>
  <Input
    style={{ width: '50%' }}
    defaultValue="Xihu District, Hangzhou"
  />
</InputGroup>

<InputGroup compact style={{ marginBottom: '15px' }}>
  <Select defaultValue="Option1" style={{ width: '33%' }}
    <Option value="Option1">Option1</Option>
    <Option value="Option2">Option2</Option>
  </Select>
  <Input
    style={{ width: '33%' }}
    defaultValue="input content"
  />
  <InputNumber style={{ width: '33%' }} />
</InputGroup>

<InputGroup compact style={{ marginBottom: '15px' }}>
  <Input
    style={{ width: '50%' }}
    defaultValue="input content"
  />
  <DatePicker />
</InputGroup>

<InputGroup compact style={{ marginBottom: '15px' }}>
  <Select defaultValue="Option1-1">
    <Option value="Option1-1">Option1-1</Option>
    <Option value="Option1-2">Option1-2</Option>
  </Select>
  <Select defaultValue="Option2-2">

```



```

        <Option value="Option2-1">Option2-1</Option>
        <Option value="Option2-2">Option2-2</Option>
    </Select>
</InputGroup>

<InputGroup compact style={{ marginBottom: '15px' }}>
    <Select defaultValue="1">
        <Option value="1">Between</Option>
        <Option value="2">Except</Option>
    </Select>
    <Input
        style={{ width: 100, textAlign: 'center' }}
        placeholder="Minimum"
    />
    <Input
        style={{ width: 24, borderLeft: 0, pointerEvents: 'none' }}
        placeholder="~"
    />
    <Input
        style={{ width: 100, textAlign: 'center', borderLeft: 0 }}
        placeholder="Maximum"
    />
</InputGroup>

<InputGroup compact style={{ marginBottom: '15px' }}>
    <Select defaultValue="Sign Up">
        <Option value="Sign Up">Sign Up</Option>
        <Option value="Sign In">Sign In</Option>
    </Select>
    <AutoComplete
        dataSource={this.state.dataSource}
        style={{ width: 200 }}
        onChange={this.handleChange}
        placeholder="Email"
    />
</InputGroup>
</ContentHolder>
</Box>

```

```
</Col>  
</Row>
```

File Path: `your-dash-app-root/src/containers/Forms/Input/index.js`

Autosizing the height to fit the content:

Autosizing the height to fit the content

`autosize` prop for a `textarea` type of `Input` makes the height to automatically adjust based on the content. An options object can be provided to `autosize` to specify the minimum and maximum number of lines the `textarea` will automatically adjust.

Autosize height based on content lines

Autosize height with minimum and maximum number of lines

Code:

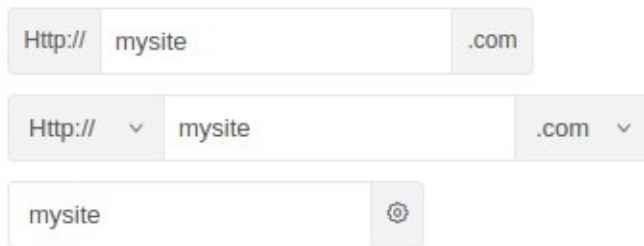
```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box
    title="Autosizing the height to fit the content"
    subtitle="autosize prop for a textarea type of Input makes the height to automatically adjust based on the content. An options object can be provided to autosize to specify the minimum and maximum number of lines the textarea will automatically adjust."
  >
    <ContentHolder>
      <Input
        type="textarea"
        placeholder="Autosize height based on content lines"
        autosize
        style={{ marginBottom: '15px' }}
      />
      <Input
        type="textarea"
        placeholder="Autosize height with minimum and maximum number of lines"
        autosize={{ minRows: 2, maxRows: 6 }}
      />
    </ContentHolder>
  </Box>
</Col>
```

File Path: `your-dash-app-root/src/containers/Forms/Input/index.js`

Pre / Post tab:

Pre / Post tab

Using pre & post tabs example..



Three examples of form input fields with pre and post tabs:

- Example 1: Pre tab "Http://", input "mysite", post tab ".com".
- Example 2: Pre tab "Http://" with a dropdown arrow, input "mysite", post tab ".com" with a dropdown arrow.
- Example 3: Pre tab "mysite", input "mysite", post tab with a settings icon.

Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box
    title="Pre / Post tab"
    subtitle="Using pre & post tabs example.."
  >
    <ContentHolder>
      <Input
        addonBefore="Http://"
        addonAfter=".com"
        defaultValue="mysite"
      />
      <Input
        addonBefore={selectBefore}
        addonAfter={selectAfter}
        defaultValue="mysite"
      />
      <Input
        addonAfter={<Icon type="setting" />}
        defaultValue="mysite"
      />
    </ContentHolder>
  </Box>
</Col>
```

File Path: `your-dash-app-root/src/containers/Forms/Input/index.js`

Search:

Search

Example of creating a search box by grouping a standard input with a search button

Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box
    title="Search"
    subtitle="Example of creating a search box by grouping a standard input with a search button"
  >
    <ContentHolder>
      <Search placeholder="input search text" />
    </ContentHolder>
  </Box>
</Col>
```

File Path: `your-dash-app-root/src/containers/Forms/Input/index.js`

Editor:

Normal ▾ Sans Serif ▾ **B** *I* U          

Write Something

Code:

```
const Editor = (props) => <Async load={import(/* webpackChunkNam
```

```
e: "forms-editor" */ '../.../components/uielements/editor')}} componentProps={props} />;
```

```
function uploadCallback(file) {
  return new Promise(
    (resolve, reject) => {
      const xhr = new XMLHttpRequest();
      xhr.open('POST', 'https://api.imgur.com/3/image');
      xhr.setRequestHeader('Authorization', 'Client-ID 8d26ccd12712fca');
      const data = new FormData();
      data.append('image', file);
      xhr.send(data);
      xhr.addEventListener('load', () => {
        const response = JSON.parse(xhr.responseText);
        resolve(response);
      });
      xhr.addEventListener('error', () => {
        const error = JSON.parse(xhr.responseText);
        reject(error);
      });
    }
  );
}
```

```
export default class AntdTreeSelect extends Component {
  constructor(props) {
    super(props);
    this.state = {
      editorState: null,
      loading: false,
      iconLoading: false,
    };
  }
  render() {
    const onEditorStateChange = (editorState) => {
      this.setState({ editorState });
    }
    const editorOption = {
      style: { width: '90%', height: '70%' },

```

```

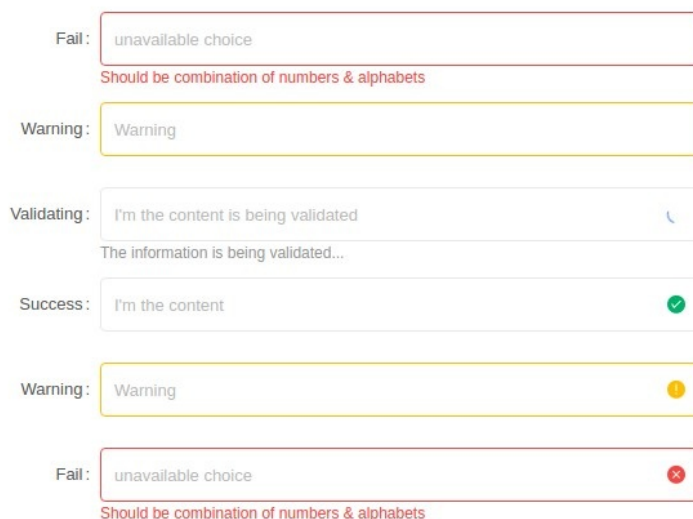
    editorState: this.state.editorState,
    toolbarClassName: 'home-toolbar',
    wrapperClassName: 'home-wrapper',
    editorClassName: 'home-editor',
    onEditorStateChange: onEditorStateChange,
    uploadCallback: uploadCallback,
    toolbar: { image: { uploadCallback: uploadCallback } },
  };

  return (<LayoutWrapper>
    <PageHeader>Editor</PageHeader>
    <Box>
      <ContentHolder>
        <Editor {...editorOption} />
      </ContentHolder>
    </Box>
  </LayoutWrapper>);
}
}

```

File Path: `your-dash-app-path/src/containers/Forms/editor/index.js`

Customized Validation Form:



Fail: unavailable choice
Should be combination of numbers & alphabets

Warning: Warning

Validating: I'm the content is being validated
The information is being validated...

Success: I'm the content

Warning: Warning

Fail: unavailable choice
Should be combination of numbers & alphabets

Code:

```
const formItemLayout = {
```

```
labelCol: {
  xs: { span: 24 },
  sm: { span: 5 },
},
wrapperCol: {
  xs: { span: 24 },
  sm: { span: 12 },
},
};

export default class FormsWithValidation extends Component {
  render() {
    return (
      <LayoutWrapper>
        <PageHeader>Customized Validation Form</PageHeader>
        <Box>
          <Form>
            <FormItem
              {...formItemLayout}
              label="Fail"
              validateStatus="error"
              help="Should be combination of numbers & alpha
            </FormItem>
            <FormItem
              {...formItemLayout}
              label="Warning"
              validateStatus="warning"
            </FormItem>
            <FormItem
              {...formItemLayout}
              label="Validating"
              hasFeedback
```



```
        validateStatus="validating"
        help="The information is being validated..."
    >
    <Input
        placeholder="I'm the content is being validated"
        id="validating"
    />
</FormItem>

<FormItem
    {...formItemLayout}
    label="Success"
    hasFeedback
    validateStatus="success"
>
    <Input placeholder="I'm the content" id="success"
/>

</FormItem>

<FormItem
    {...formItemLayout}
    label="Warning"
    hasFeedback
    validateStatus="warning"
>
    <Input placeholder="Warning" id="warning" />
</FormItem>

<FormItem
    {...formItemLayout}
    label="Fail"
    hasFeedback
    validateStatus="error"
    help="Should be combination of numbers & alpha
    bets"
>
    <Input placeholder="unavailable choice" id="error"
/>

</FormItem>
</Form>
```

```
        </Box>
      </LayoutWrapper>
    );
  }
}
```

File Path: `your-dash-app-patha/src/containers/Forms/FormsWithValidation/index.js`

Progress Bar:

Standard Progress:

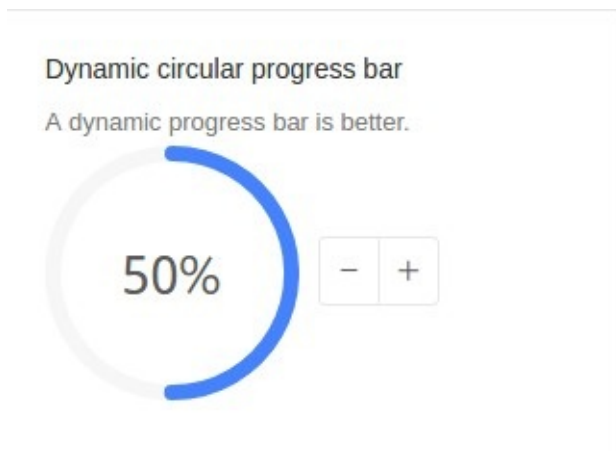


Code:

```
<Col md={12} xs={24} style={colStyle}>
  <Box title="Progress bar" subtitle="A standard progress bar.">
    <Progress percent={30} style={marginStyle} />
    <Progress percent={50} status="active" style={marginStyle} /
  >
    <Progress percent={70} status="exception" style={marginStyle
  } />
    <Progress percent={100} style={marginStyle} />
    <Progress percent={50} showInfo={false} style={marginStyle}
  />
  </Box>
</Col>
```

File Path: `your-dash-app-patha/src/containers/Forms/Progress/index.js`

Dynamic circular progress bar:

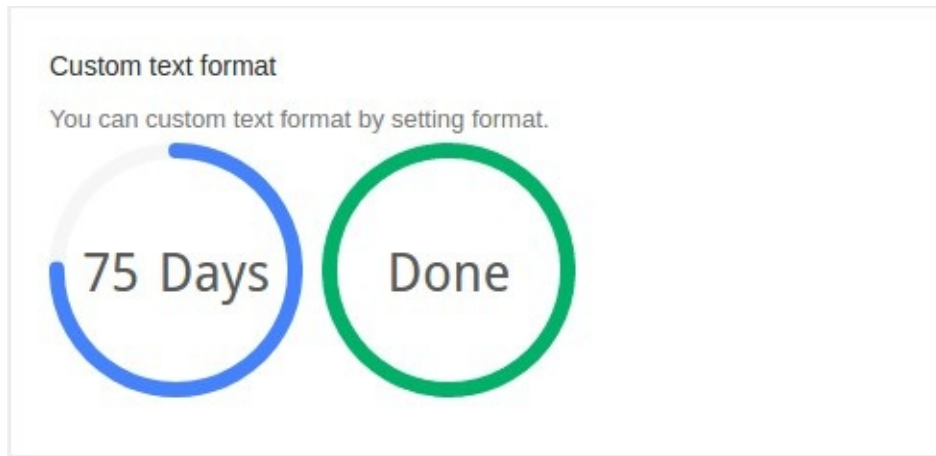


Code:

```
<Col md={8} xs={24} style={colStyle}>
  <Box
    title="Dynamic circular progress bar"
    subtitle="A dynamic progress bar is better."
  >
    <Progress
      type="circle"
      percent={this.state.percent}
      style={marginStyle}
    />
    <ButtonGroup>
      <Button onClick={this.decline} icon="minus" />
      <Button onClick={this.increase} icon="plus" />
    </ButtonGroup>
  </Box>
</Col>
```

File Path: `your-dash-app-patha/src/containers/Forms/Progress/index.js`

Custom text format:



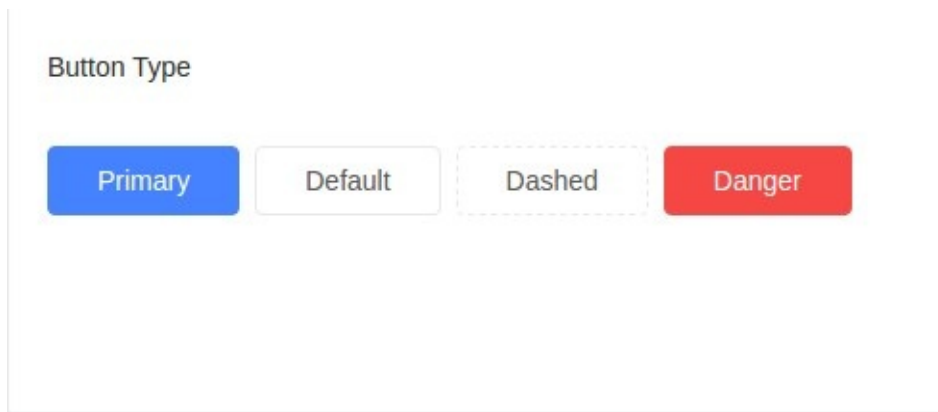
Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box
    title="Custom text format"
    subtitle="You can custom text format by setting format."
  >
    <Progress
      type="circle"
      percent={75}
      format={percent => `${percent} Days`}
      style={marginStyle}
    />
    <Progress
      type="circle"
      percent={100}
      format={() => 'Done'}
      style={marginStyle}
    />
  </Box>
</Col>
```

File Path: your-dash-app-patha/src/containers/Forms/Progress/index.js

Button:

Button Type:

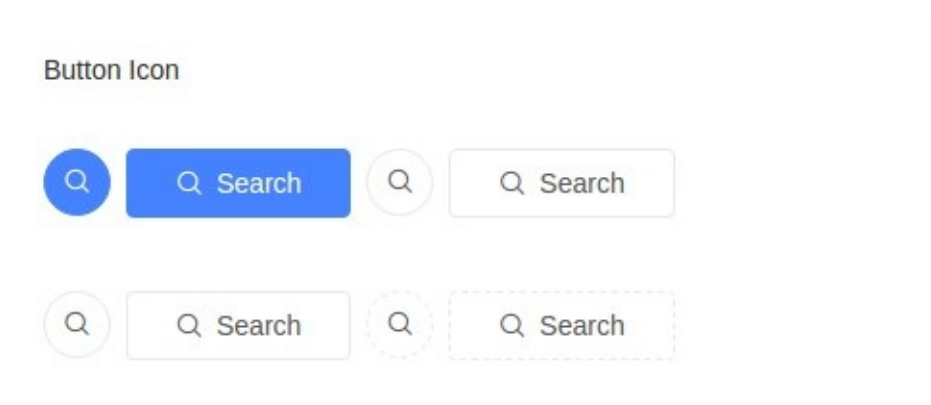


Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box title="Button Type">
    <ContentHolder>
      <Button type="primary" style={margin}>Primary</Button>
      <Button style={margin}>Default</Button>
      <Button type="dashed" style={margin}>Dashed</Button>
      <Button type="danger">Danger</Button>
    </ContentHolder>
  </Box>
</Col>
```

File Path: `src/containers/Forms/Button/index.js`

Button Icon:



Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box title="Button Icon">
    <ContentHolder>
      <Button
        type="primary"
        shape="circle"
        icon="search"
        style={margin}
      />
      <Button type="primary" icon="search" style={margin}>
        Search
      </Button>
      <Button shape="circle" icon="search" style={margin} />
      <Button icon="search">Search</Button>
    </ContentHolder>



    <ContentHolder>
      <Button shape="circle" icon="search" style={margin} />
      <Button icon="search" style={margin}>Search</Button>
      <Button
        type="dashed"
        shape="circle"
        icon="search"
        style={margin}
      />
      <Button type="dashed" icon="search">Search</Button>
    </ContentHolder>
  </Box>
</Col>
```

File Path: `src/containers/Forms/Button/index.js`

Button Size:

Button Size

Large Default Small

  Download Normal

< Backward Forward >

Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box title="Button Size">
    <ContentHolder>
      <Radio.Group value={size} onChange={this.handleSizeChange}
>
        <Radio.Button value="large">Large</Radio.Button>
        <Radio.Button value="default">Default</Radio.Button>
        <Radio.Button value="small">Small</Radio.Button>
      </Radio.Group>
    </ContentHolder>

    <ContentHolder>
      <Button
        type="primary"
        shape="circle"
        icon="download"
        size={size}
        style={margin}
      />
      <Button
        type="primary"
        icon="download"
        size={size}
```

```
        style={margin}
      >
        Download
      </Button>
      <Button type="primary" size={size}>Normal</Button>
    </ContentHolder>

    <ContentHolder>
      <ButtonGroup size={size}>
        <Button type="primary">
          <Icon type="left" />Backward
        </Button>
        <Button type="primary">
          Forward<Icon type="right" />
        </Button>
      </ButtonGroup>
    </ContentHolder>
  </Box>
</Col>
```

File Path: `src/containers/Forms/Button/index.js`

Button Loading:

Button Loading



Code:


```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box title="Button Loading">
    <ContentHolder>
      <Button type="primary" loading style={margin}>
        Loading
      </Button>
      <Button type="primary" size="small" loading>
        Loading
      </Button>
    </ContentHolder>

    <ContentHolder>
      <Button
        type="primary"
        loading={this.state.loading}
        onClick={this.enterLoading}
        style={margin}
      >
        Click me!
      </Button>
      <Button
        type="primary"
        icon="poweroff"
        loading={this.state.iconLoading}
        onClick={this.enterIconLoading}
      >
        Click me!
      </Button>
    </ContentHolder>

    <ContentHolder>
      <Button shape="circle" loading style={margin} />
      <Button type="primary" shape="circle" loading />
    </ContentHolder>
  </Box>
</Col>
```

File Path: `src/containers/Forms/Button/index.js`

Button Group:

Button Group

Basic

Cancel

OK

L

M

R

L

M



M

R

With Icon

< Go back

Go forward >



Code:

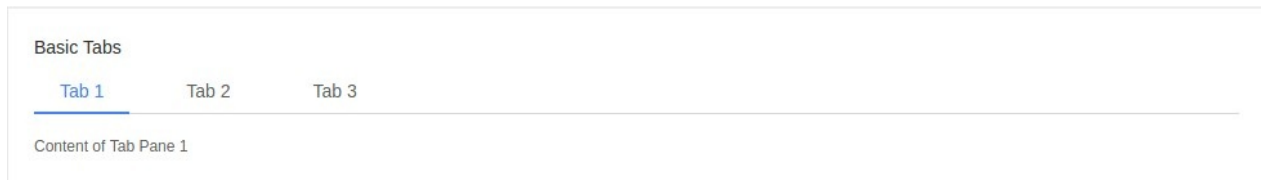
```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box title="Button Group">
    <ContentHolder>
      <h4>Basic</h4>
      <ButtonGroup style={margin}>
        <Button>Cancel</Button>
        <Button type="primary">OK</Button>
      </ButtonGroup>
      <ButtonGroup style={margin}>
        <Button disabled>L</Button>
        <Button disabled>M</Button>
        <Button disabled>R</Button>
      </ButtonGroup>
      <ButtonGroup style={margin}>
        <Button type="primary">L</Button>
        <Button>M</Button>
        <Button>M</Button>
        <Button type="dashed">R</Button>
      </ButtonGroup>
    </ContentHolder>

    <ContentHolder>
      <h4>With Icon</h4>
      <ButtonGroup style={margin}>
        <Button type="primary">
          <Icon type="left" />Go back
        </Button>
        <Button type="primary">
          Go forward<Icon type="right" />
        </Button>
      </ButtonGroup>
      <ButtonGroup>
        <Button type="primary" icon="cloud" />
        <Button type="primary" icon="cloud-download" />
      </ButtonGroup>
    </ContentHolder>
  </Box>
</Col>
```

File Path: `src/containers/Forms/Button/index.js`

Tabs:

Basic Tab:

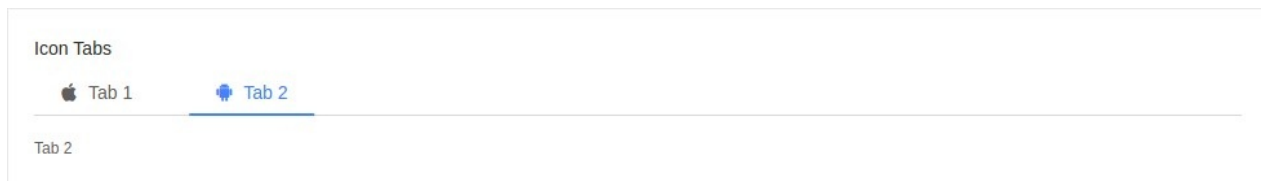


Code:

```
<Box title="Basic Tabs">
  <Tabs defaultActiveKey="1" onChange={callback}>
    <TabPane tab="Tab 1" key="1">Content of Tab Pane 1</TabPane>
    <TabPane tab="Tab 2" key="2">Content of Tab Pane 2</TabPane>
    <TabPane tab="Tab 3" key="3">Content of Tab Pane 3</TabPane>
  </Tabs>
</Box>
```

File Path: `src/containers/Forms/Tab/index.js`

Icon Tab:



Code:

```
<Box title="Icon Tabs">
  <Tabs defaultActiveKey="2">
    <TabPane tab={<span><Icon type="apple" />Tab 1</span>} key="
1">
      Tab 1
    </TabPane>
    <TabPane tab={<span><Icon type="android" />Tab 2</span>} key
="2">
      Tab 2
    </TabPane>
  </Tabs>
</Box>
```

File Path: `src/containers/Forms/Tab/index.js`

Position Tab:

Position

Tabs's position: left, right, top or bottom

Tab position :

Tab 1

Tab 2

Tab 3

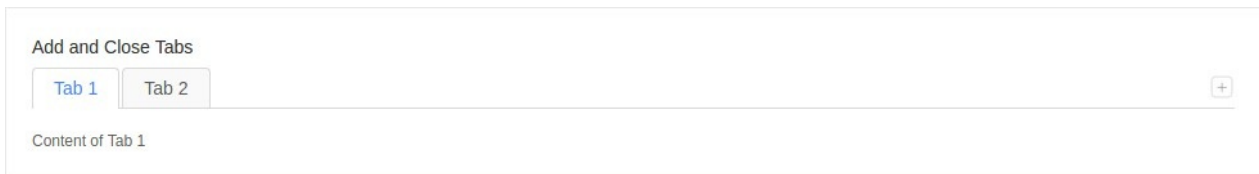
Content of Tab 1

Code:

```
<Box
  title="Position"
  subtitle="Tabs's position: left, right, top or bottom"
>
<div style={{ marginBottom: 16 }}>
  Tab position :
  <Select
    value={this.state.tabPosition}
    onChange={this.changeTabPosition}
    dropdownMatchSelectWidth={false}
  >
    <Option value="top">top</Option>
    <Option value="bottom">bottom</Option>
    <Option value="left">left</Option>
    <Option value="right">right</Option>
  </Select>
</div>
<Tabs tabPosition={this.state.tabPosition}>
  <TabPane tab="Tab 1" key="1">Content of Tab 1</TabPane>
  <TabPane tab="Tab 2" key="2">Content of Tab 2</TabPane>
  <TabPane tab="Tab 3" key="3">Content of Tab 3</TabPane>
</Tabs>
</Box>
```

File Path: `src/containers/Forms/Tab/index.js`

Add and Close Tabs:

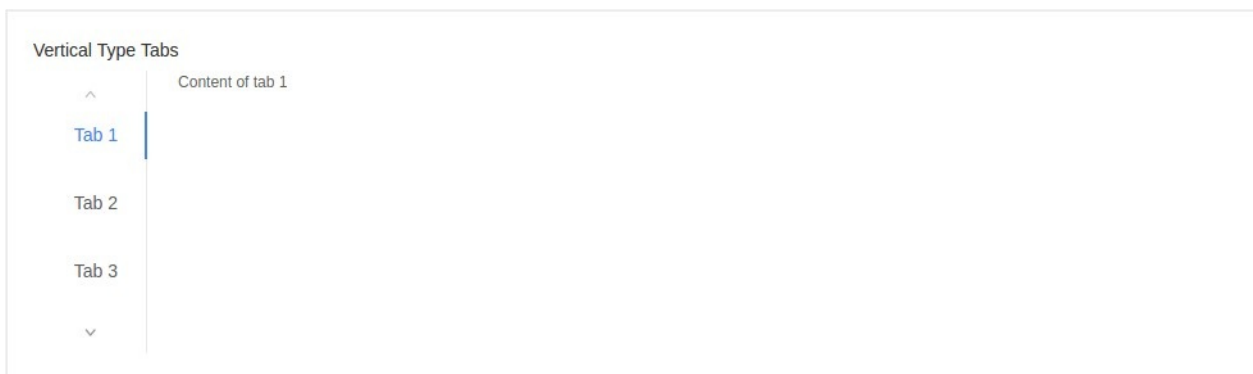


Code:

```
<Box title="Add and Close Tabs">
  <Tabs
    onChange={this.onChange}
    activeKey={this.state.activeKey}
    type="editable-card"
    onEdit={this.onEdit}
  >
    {this.state.panes.map(pane => (
      <TabPane tab={pane.title} key={pane.key} closable={pane.closable}>
        {pane.content}
      </TabPane>
    ))}
  </Tabs>
</Box>
```

File Path: `src/containers/Forms/Tab/index.js`

Vertical Type Tabs:



Code:

```
<Box title="Vertical Type Tabs">
  <div className="card-container">
    <Tabs
      defaultActiveKey="1"
      tabPosition="left"
      style={{ height: 220 }}
    >
      <TabPane tab="Tab 1" key="1">Content of tab 1</TabPane>
      <TabPane tab="Tab 2" key="2">Content of tab 2</TabPane>
      <TabPane tab="Tab 3" key="3">Content of tab 3</TabPane>
      <TabPane tab="Tab 4" key="4">Content of tab 4</TabPane>
      <TabPane tab="Tab 5" key="5">Content of tab 5</TabPane>
      <TabPane tab="Tab 6" key="6">Content of tab 6</TabPane>
      <TabPane tab="Tab 7" key="7">Content of tab 7</TabPane>
      <TabPane tab="Tab 8" key="8">Content of tab 8</TabPane>
      <TabPane tab="Tab 9" key="9">Content of tab 9</TabPane>
      <TabPane tab="Tab 10" key="10">Content of tab 10</TabPane>
      <TabPane tab="Tab 11" key="11">Content of tab 11</TabPane>
    </Tabs>
  </div>
</Box>
```

File Path: `src/containers/Forms/Tab/index.js`

Basic Checkbox:

Basic Checkbox

Basic usage of checkbox.

☐ Checkbox

Code:


```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box title="Basic Checkbox" subtitle="Basic usage of checkbox."
">
    <ContentHolder>
      <Checkbox onChange={this.handleOnChange}>Checkbox</Checkbo
x>
    </ContentHolder>
  </Box>
</Col>
```

File Path: `src/containers/Forms/Checkbox/index.js`

Checkbox Group:

Checkbox Group

Generate a group of checkboxes from an array. Use `disabled` to disable a checkbox.

☒ Apple ☐ Pear ☐ Orange

☐ Apple ☒ Pear ☐ Orange

☒ Apple ☐ Pear ☐ Orange

Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box
    title="Checkbox Group"
    subtitle="Generate a group of checkboxes from an array. Use
disabled to disable a checkbox."
  >
    <ContentHolder>
      <CheckboxGroup
        options={plainOptions}
        defaultValue={['Apple']}
        onChange={this.handleOnChange}
      />
      <br />
      <CheckboxGroup
        options={options}
        defaultValue={['Pear']}
        onChange={this.handleOnChange}
      />
      <br />
      <CheckboxGroup
        options={optionsWithDisabled}
        disabled
        defaultValue={['Apple']}
        onChange={this.handleOnChange}
      />
    </ContentHolder>
  </Box>
</Col>
```

File Path: `src/containers/Forms/Checkbox/index.js`

Checkbox Group 2:

Checkbox Group

Generate a group of checkboxes from an array. Use disabled to disable a checkbox.

☒ Check all

☒ Apple ☐ Pear ☒ Orange

Code:

```
<Row style={rowStyle} gutter={gutter} justify="start">
  <Col md={12} sm={12} xs={24} style={colStyle}>
    <Box
      title="Checkbox Group"
      subtitle="Generate a group of checkboxes from an array. Use disabled to disable a checkbox."
    >
      <ContentHolder>
        <div>
          <div
            style={{
              borderBottom: '1px solid #E9E9E9',
              paddingBottom: '15px',
            }}
          >
            <Checkbox
              indeterminate={this.state.indeterminate}
              onChange={this.onCheckAllChange}
              checked={this.state.checkAll}
            >
              Check all
            </Checkbox>
          </div>
          <br />
          <CheckboxGroup
            options={plainOptions}
            value={this.state.checkedList}
            onChange={this.onChange}
          />
        </div>
      </ContentHolder>
    </Box>
  </Col>
</Row>
```

File Path: `src/containers/Forms/Checkbox/index.js`

Basic Radio:

Basic Radio

The simplest use. Use disabled to disable a radio.

- ☐ Radio
- ☐ Disabled
- ☒ Disabled

Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box
    title="Basic Radio"
    subtitle="The simplest use. Use disabled to disable a radio."
  >
    <ContentHolder>
      <Radio>Radio</Radio>
      <br />
      <Radio defaultChecked={false} disabled>Disabled</Radio>
      <br />
      <Radio defaultChecked disabled>Disabled</Radio>
    </ContentHolder>
  </Box>
</Col>
```

File Path: `src/containers/Forms/Radiobox/index.js`

Vertical RadioGroup:

Vertical RadioGroup

Vertical RadioGroup, with more radios.

- ☒ Option A
- ☐ Option B
- ☐ Option C
- ☐ More...

Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box
    title="Vertical RadioGroup"
    subtitle="Vertical RadioGroup, with more radios."
  >
    <ContentHolder>
      <RadioGroup onChange={this.onChange} value={this.state.value}>
        <Radio style={radioStyle} value={1}>Option A</Radio>
        <Radio style={radioStyle} value={2}>Option B</Radio>
        <Radio style={radioStyle} value={3}>Option C</Radio>
        <Radio style={radioStyle} value={4}>
          More...
          {this.state.value === 4
            ? <Input style={{ width: 100, marginLeft: 10 }} />
            : null}
        </Radio>
      </RadioGroup>
    </ContentHolder>
  </Box>
</Col>
```

File Path: `src/containers/Forms/Radiobox/index.js`

RadioGroup:

RadioGroup

A group of radio components.



Code:

```
<Col md={12} sm={12} xs={24} style={colStyle}>
  <Box title="RadioGroup" subtitle="A group of radio components."
  ">
    <ContentHolder>
      <RadioGroup
        options={plainOptions}
        onChange={this.onChange1}
        value={this.state.value1}
        style={{ marginBottom: '10px' }}
      />
      <RadioGroup
        options={options}
        onChange={this.onChange2}
        value={this.state.value2}
        style={{ marginBottom: '10px' }}
      />
      <RadioGroup
        options={optionsWithDisabled}
        onChange={this.onChange3}
        value={this.state.value3}
      />
    </ContentHolder>
  </Box>
</Col>
```

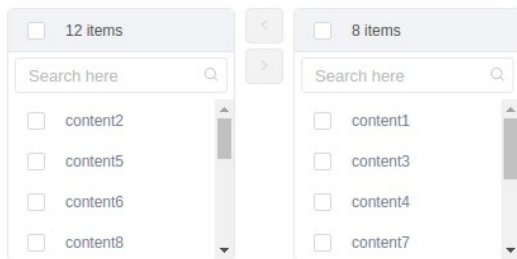
File Path: `src/containers/Forms/Radiobox/index.js`

Transfer:

Transfer with a search box:

Search

Transfer with a search box.



Code:

```
export default class IsomorphicTransfer extends Component {
  state = {
    mockData: [],
    targetKeys: [],
  };
  componentDidMount() {
    this.getMock();
  }
  getMock = () => {
    const targetKeys = [];
    const mockData = [];
    for (let i = 0; i < 20; i++) {
      const data = {
        key: i.toString(),
        title: `content${i + 1}`,
        description: `description of content${i + 1}`,
        chosen: Math.random() * 2 > 1,
      };
      if (data.chosen) {
        targetKeys.push(data.key);
      }
      mockData.push(data);
    }
    this.setState({ mockData, targetKeys });
  };
  filterOption = (inputValue, option) => {
    return option.description.indexOf(inputValue) > -1;
  };
}
```



```
};
handleChange = targetKeys => {
  this.setState({ targetKeys });
};
render() {
  return (
    <LayoutWrapper>
      <PageHeader>Transfer</PageHeader>

      <Box title="Search" subtitle="Transfer with a search box
.">
        <ContentHolder>
          <Transfer
            dataSource={this.state.mockData}
            showSearch
            filterOption={this.filterOption}
            targetKeys={this.state.targetKeys}
            onChange={this.handleChange}
            render={item => item.title}
            className="isomorphicTransfer"
          />
        </ContentHolder>
      </Box>
    </LayoutWrapper>
  );
}
}
```

File Path: `src/containers/Forms/Transfer/index.js`

Autocomplete:

Customized

You could pass `AutoComplete.Option` as children of `AutoComplete`, instead of using `dataSource`

Code:

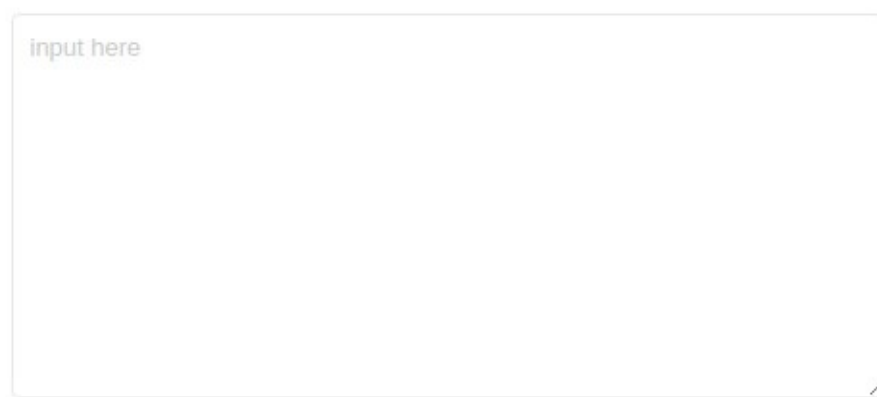
```
<Col md={12} xs={24} style={colStyle}>
  <Box
    title="Customized"
    subtitle="You could pass AutoComplete.Option as children of
AutoComplete, instead of using dataSource"
  >
    <ContentHolder>
      <AutoComplete
        onChange={this.handleCustomizedChange}
        placeholder="input here"
      >
        {children}
      </AutoComplete>
    </ContentHolder>
  </Box>
</Col>
```

File Path: `src/containers/Forms/AutoComplete/index.js`

Customize Input Component:

Customize Input Component

Customize Input Component



Code:

```
<Col md={12} xs={24} style={colStyle}>
  <Box
    title="Customize Input Component"
    subtitle="Customize Input Component"
  >
    <ContentHolder>
      <AutoComplete
        dataSource={dataSource}
        style={{ height: 200 }}
        onChange={this.handleChange}
        placeholder="input here"
      >
        <textarea style={{ height: 200 }} />
      </AutoComplete>
    </ContentHolder>
  </Box>
</Col>
```

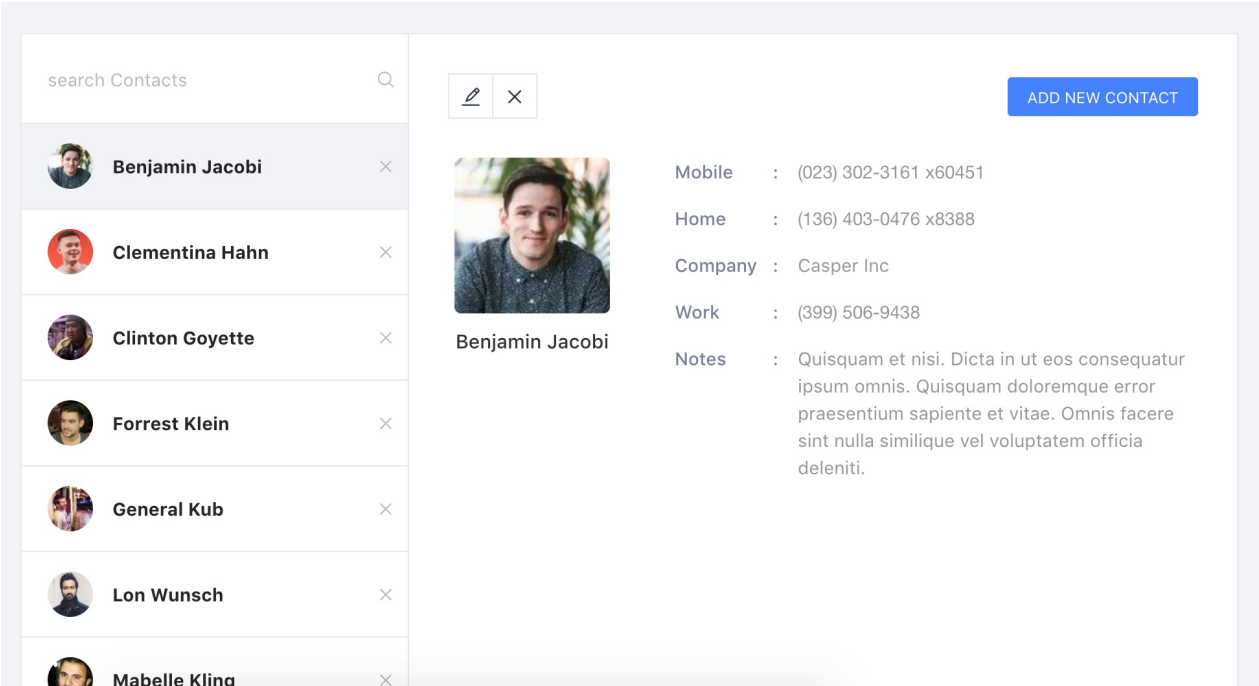
File Path: src/containers/Forms/AutoComplete/index.js

Contact

Folder path: /src/containers/contacts

If you want to render `Contact` component like the the following images for different Views


Desktop View



Tab View


search Contacts

Q




Benjamin Jacobi

X




Clementina Hahn

X




Clinton Goyette

X




Forrest Klein

X




General Kub

X




Lon Wunsch

X




Mabelle Kling

X



X

ADD NEW CONTACT



Benjamin Jacobi

Mobile : (023) 302-3161 x60451

Home : (136) 403-0476 x8388

Company : Casper Inc


Work : (399) 506-9438

Notes : Quisquam et nisi. Dicta in ut eos consequatur ipsum omnis. Quisquam doloremque error praesentium sapiente et vitae. Omnis facere sint nulla similique vel voluptatem officia deleniti.

Mobile View


161

search Contacts




Benjamin Jacobi

×




Clementina Hahn

×




Clinton Goyette

×




Forrest Klein


×



General Kub

×





ADD NEW CONTACT

```
<ContactList
  contacts={contacts}
  seectedId={seectedId}
  changeContact={changeContact}
  deleteContact={deleteContact} />
```

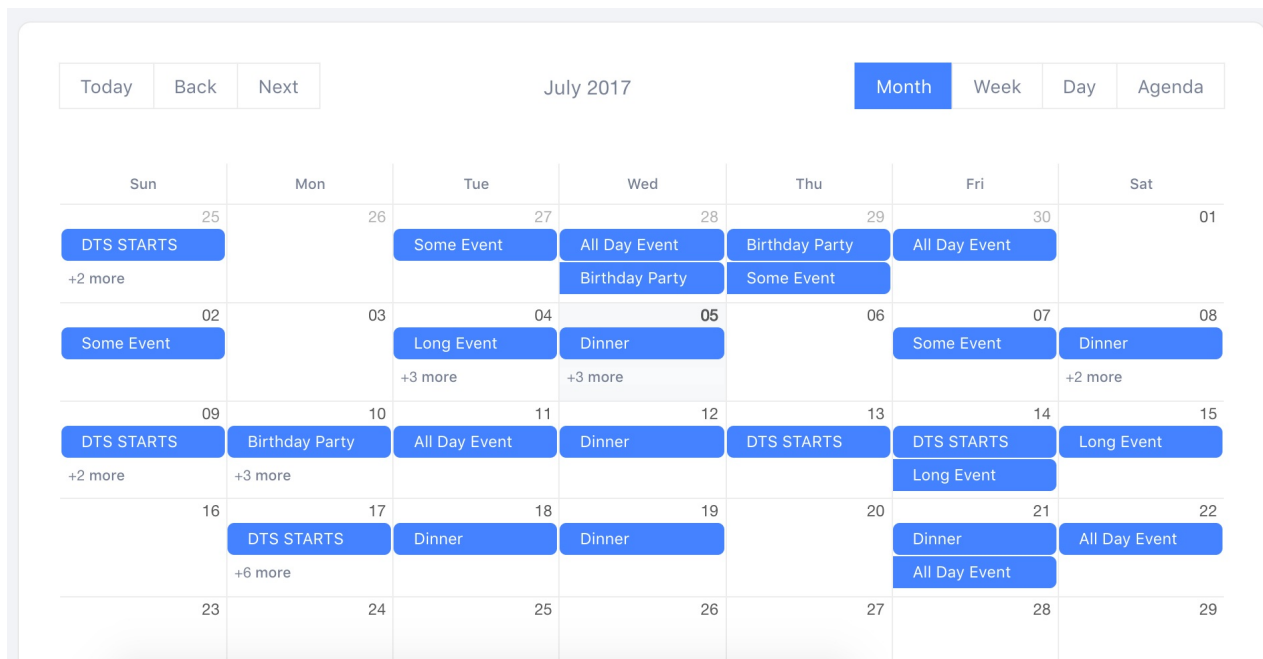
There are some main components

Component	Description
contacts	Carry all contacts
deleteContact	action use to delete contact
changeContact	action use to edit contact

Calendar

Folder path: /src/containers/Calendar

If you want to render Calendar component like the following images for different Views



The event prototypes are

Name	Description
id	Id of the event
allDay	Is the event all day
desc	Description about the event
start	Start time
end	End time

Ant Table

Folder path: /src/containers/Tables/antTables

Simple Table

It is a simple table it is in /tableviews/simpleView

First Name	Last Name	City	Street
Emelia	Gislason	Lake Zelda	Kulas Shoals
Cloyd	Armstrong	East Pierce	Lyla Heights
Rahul	Funk	Sibylside	Jolie Shoals
Hilbert	Langosh	Anaishshire	Sim Station
Cloyd	Wilderman	North Brad	Ruecker Turnpike
Jonatan	Gutmann	Goyetteside	Donnelly Mountains
Verdie	O'Conner	West Terrence	Windler Mountains
Elza	Hoeger	Dietrichfort	Howe Stravenue

Sortable Table

It is a sortable table it is in /tableviews/sortView

Ant Table

First Name ▴ ▾	Last Name ▴ ▾	City ▴ ▾	Street ▴ ▾
Verdie	O'Conner	West Terrence	Windler Mountains
Rahul	Funk	Sibylside	Jolie Shoals
Jonatan	Gutmann	Goyetteside	Donnelly Mountains
Hilbert	Langosh	Anaishshire	Sim Station
Gennaro	Waters	Kaitlynmouth	O'Hara Radial
Erling	Armstrong	Hammeschester	Fanny Lights
Emelia	Gislason	Lake Zelda	Kulas Shoals
Elza	Hoeger	Dietrichfort	Howe Stravenue






Search Text

It is a search table. Anyone can search columns it is in
/tableviews/filterView

Q First Name	Last Name	City	Street
<div><input type="text" value="Search name"/> <button>SEARCH</button></div>		Sibylside	Jolie Shoals
Jonatan	Gutmann	Goyetteside	Donnelly Mountains
Gennaro	Waters	Kaitlynmouth	O'Hara Radial
Emelia	Gislason	Lake Zelda	Kulas Shoals
Elza	Hoeger	Dietrichfort	Howe Stravenue

Editable View

It is a editable table. Anyone can edit or delete columns it is in
/tableviews/editView

First Name		Last Name	City	Street	operat
<div>Verdie</div>	✓	O'Conner	West Terrence	Windler Mountains	Delete
Rahul		Funk	Sibylside	Jolie Shoals	Delete
Jonatan		Gutmann	Goyetteside	Donnelly Mountains	Delete
Hilbert		Langosh	Anaishshire	Sim Station	Delete
Gennaro		Waters	Kaitlynmouth	O'Hara Radial	Delete
Erling		Armstrong	Hammeschester	Fanny Lights	Delete

Grouping View

It is a group table. Anyone can search columns it is in `/tableviews/groupView`

image	Name		Address	
	First Name	Last Name	City	Street
	Verdie	O'Conner	West Terrence	Windler Mountains
	Rahul	Funk	Sibylside	Jolie Shoals
	Jonatan	Gutmann	Goyetteside	Donnelly Mountains
	Hilbert	Langosh	Anaishshire	Sim Station
	Gennaro	Waters	Kaitlynmouth	O'Hara Radial

Customized View

You can also customize various options like `Bordered` , `Loading` , `Pagination` , `Title` , `Show Header` , `Footer` , `Expanded Row Render` , `Checkbox` , `Scrollable` etc.

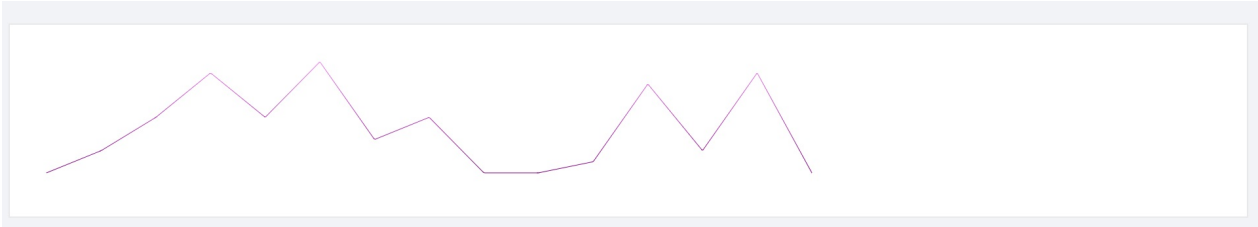
Bordered : ☐ Loading : ☐ Pagination : ☒ Title : ☒ Show Header : ☒ Footer : ☒
Expanded Row Render : ☒ Checkbox : ☒ Scrollable : ☐

Here is title								
	<input type="checkbox"/>	image	First Name	Last Name	City	Street	Email	DOB
<input type="checkbox"/>	<input type="checkbox"/>		Verdie	O'Conner	West Terrence	Windler Mountains	Tina.Stehr66@hotmail.com	2017-02-10
<input type="checkbox"/>	<input type="checkbox"/>		Rahul	Funk	Sibylside	Jolie Shoals	Ransom.Bergstrom@gmail.com	2017-02-10
<input type="checkbox"/>	<input type="checkbox"/>		Jonatan	Gutmann	Goyetteside	Donnelly Mountains	Tamia.Abbott98@hotmail.com	2016-10-10
<input type="checkbox"/>	<input type="checkbox"/>		Hilbert	Langosh	Anaishshire	Sim Station	Loyce.Upton@hotmail.com	2017-02-10
<input type="checkbox"/>	<input type="checkbox"/>		Gennaro	Waters	Kaitlynmouth	O'Hara Radial	Jefferey33@hotmail.com	2017-02-10

React Trend Chart

Folder path: /src/containers/charts/reactTrend

If you want to render React trend chart component like the following image.



Then The code should be like this.

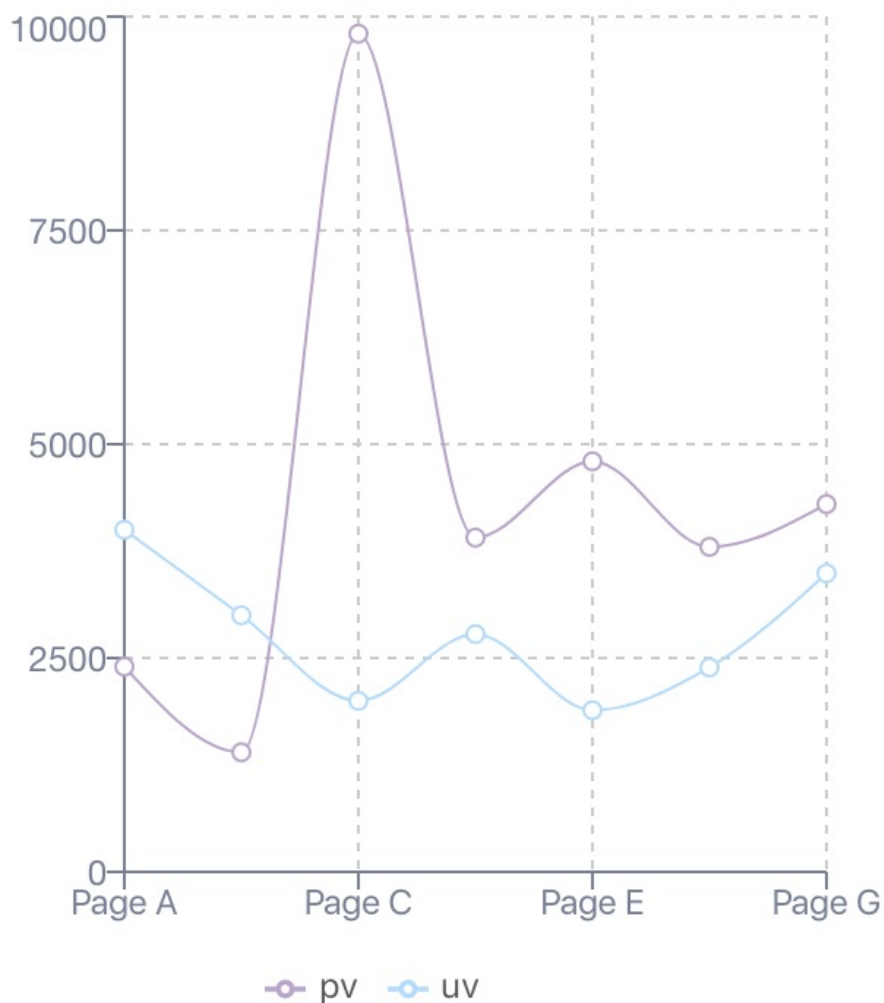
```
<Trend
  smooth={smooth}
  autoDraw={autoDraw}
  autoDrawDuration={parseInt(autoDrawDuration, 10)}
  autoDrawEasing={autoDrawEasing}
  height={100}
  width={600}
  data={data}
  gradient={gradient}
  radius={parseInt(radius, 10)}
  strokeWidth={strokeWidth}
  strokeLinecap={strokeLinecap} />
```

Recharts

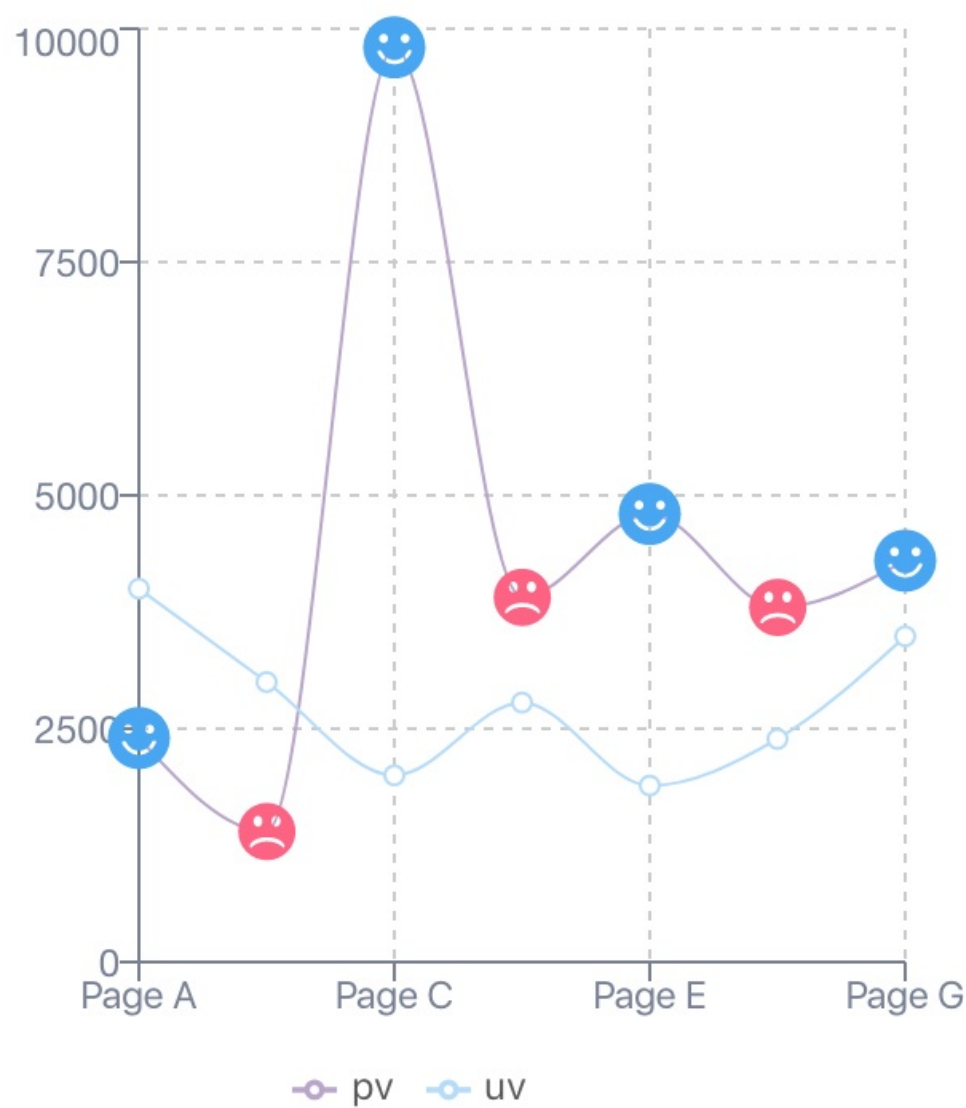
Folder path: /src/containers/charts/recharts

If you want to render Recharts component like the following images. The configurations are given in the `config` file

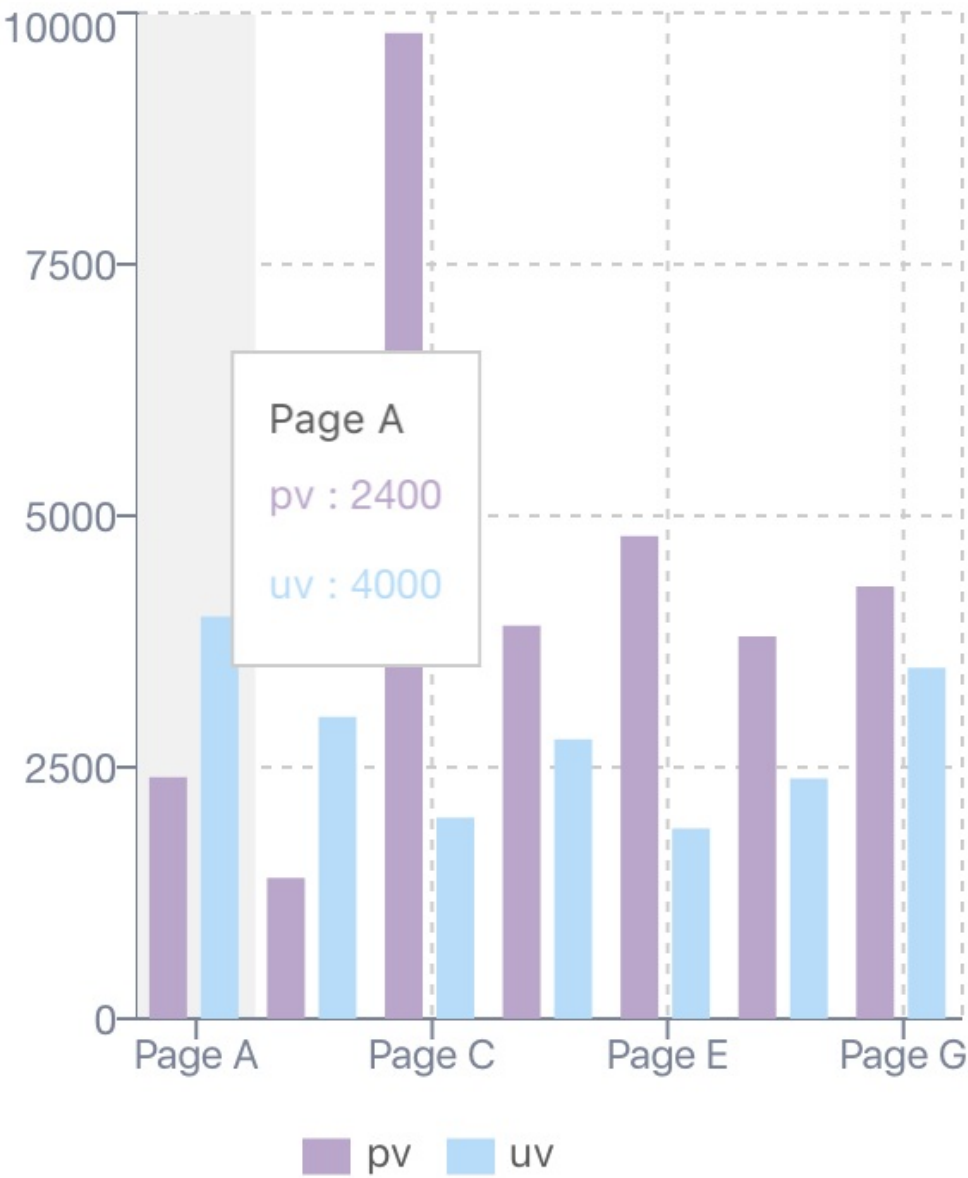
SimpleLineCharts



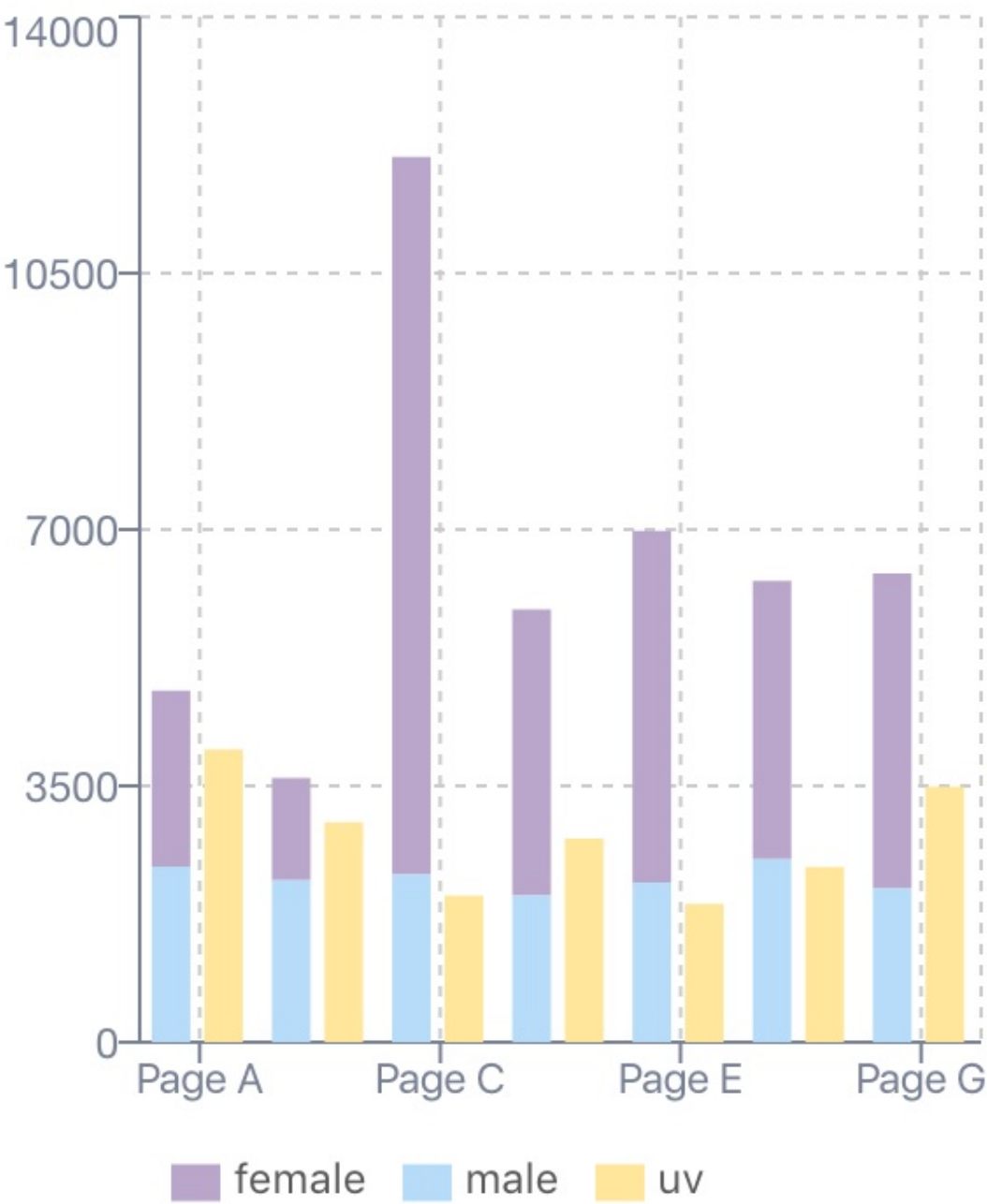
CustomizedDotLineChart



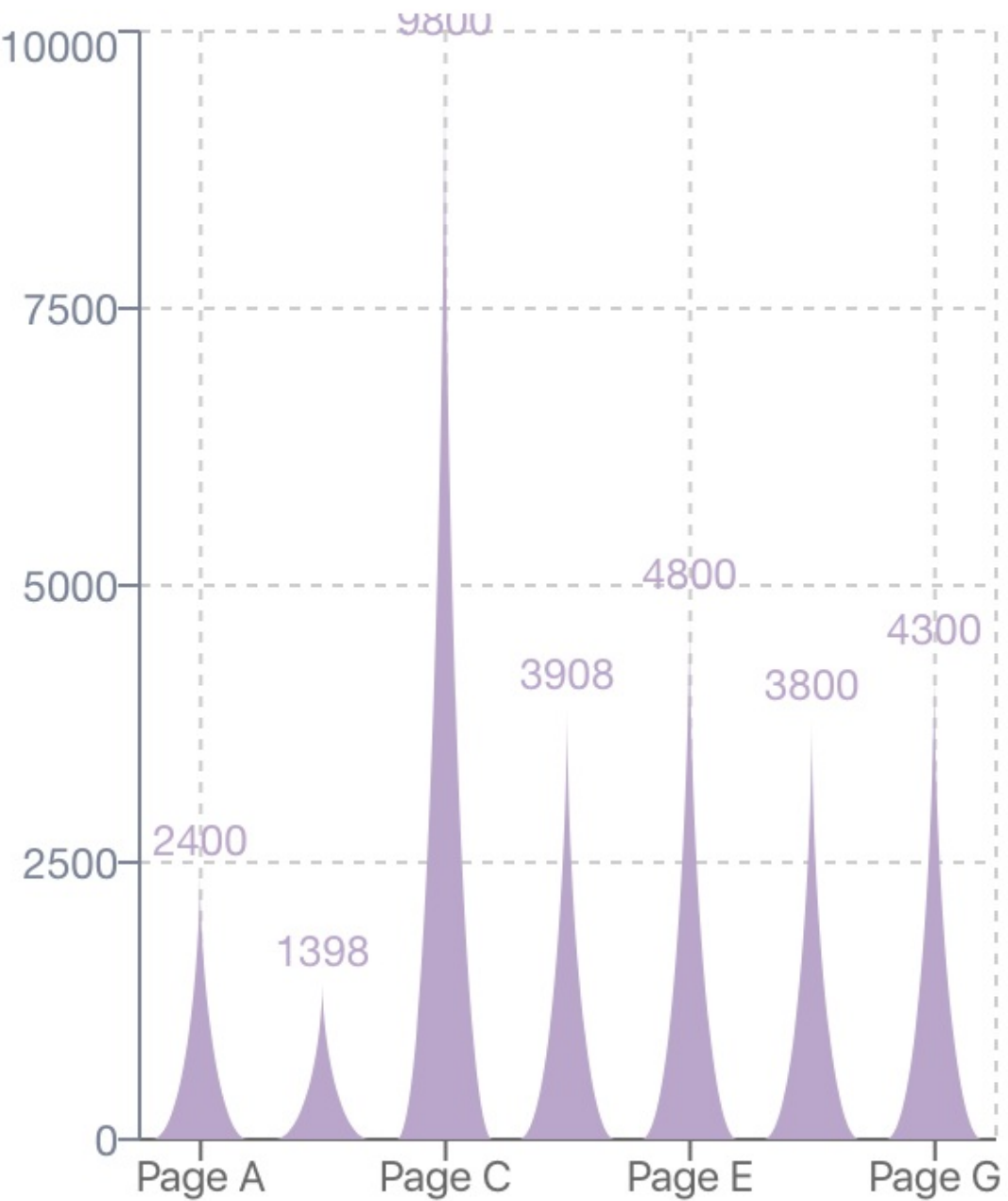
SimpleBarChart



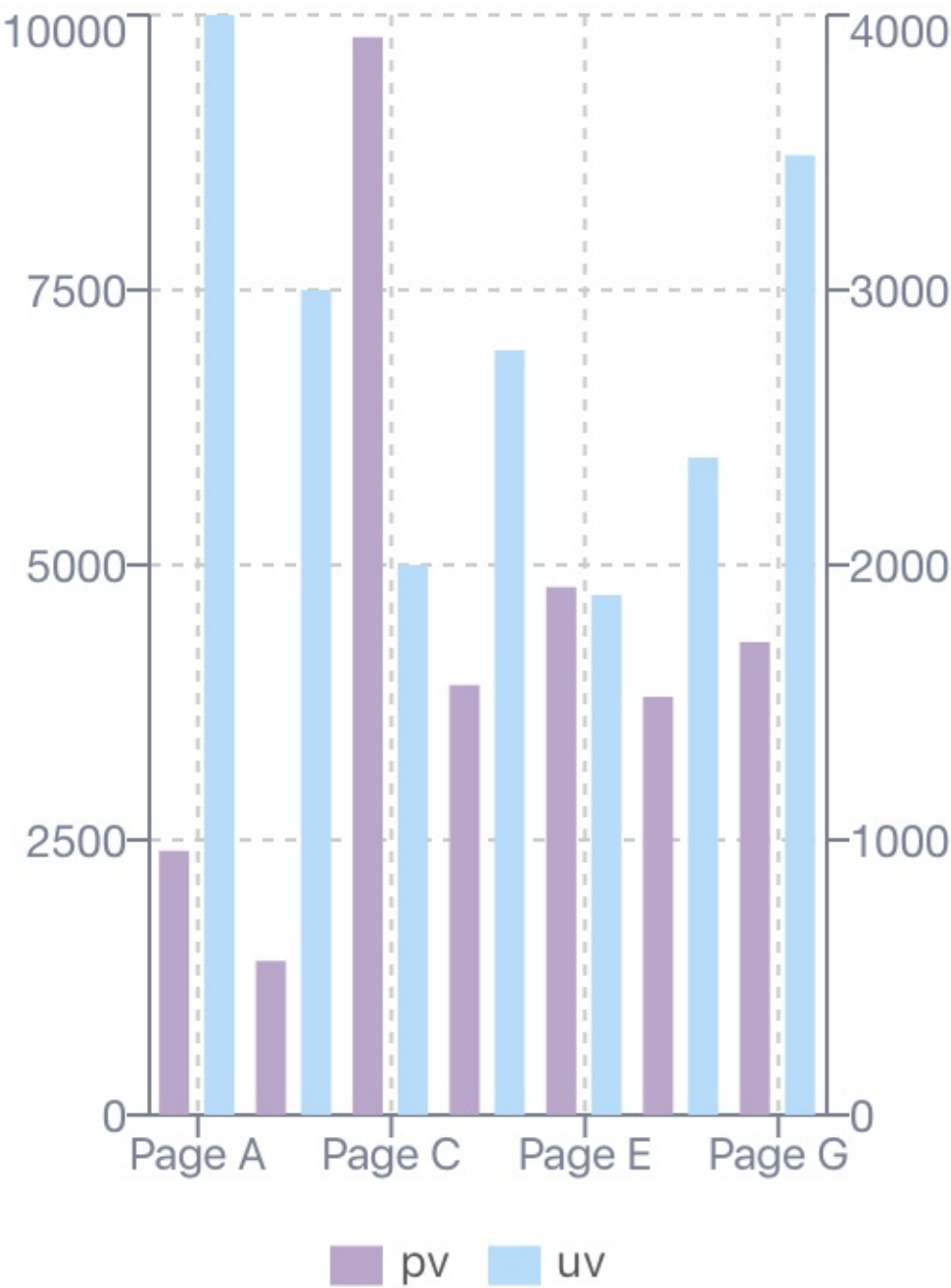
MixBarChart



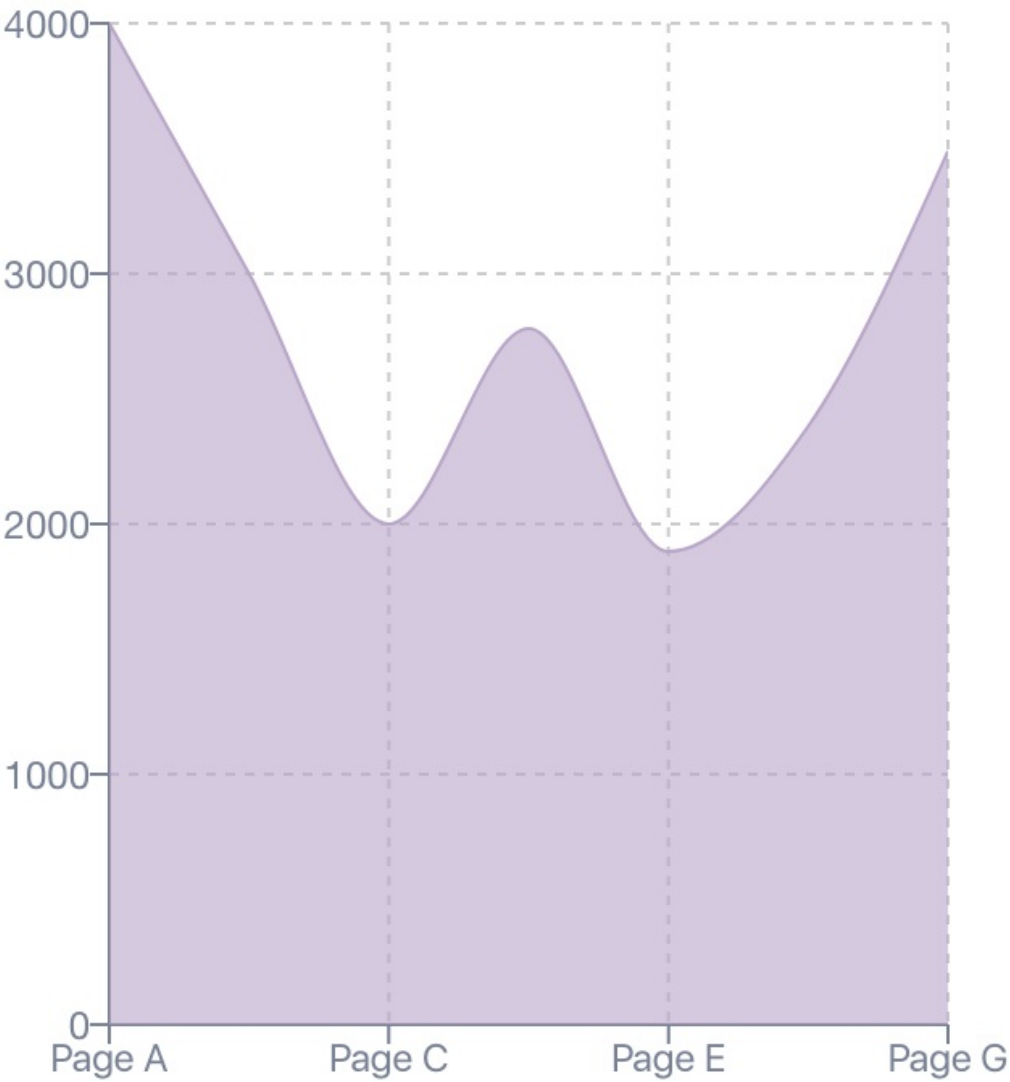
CustomShapeBarChart



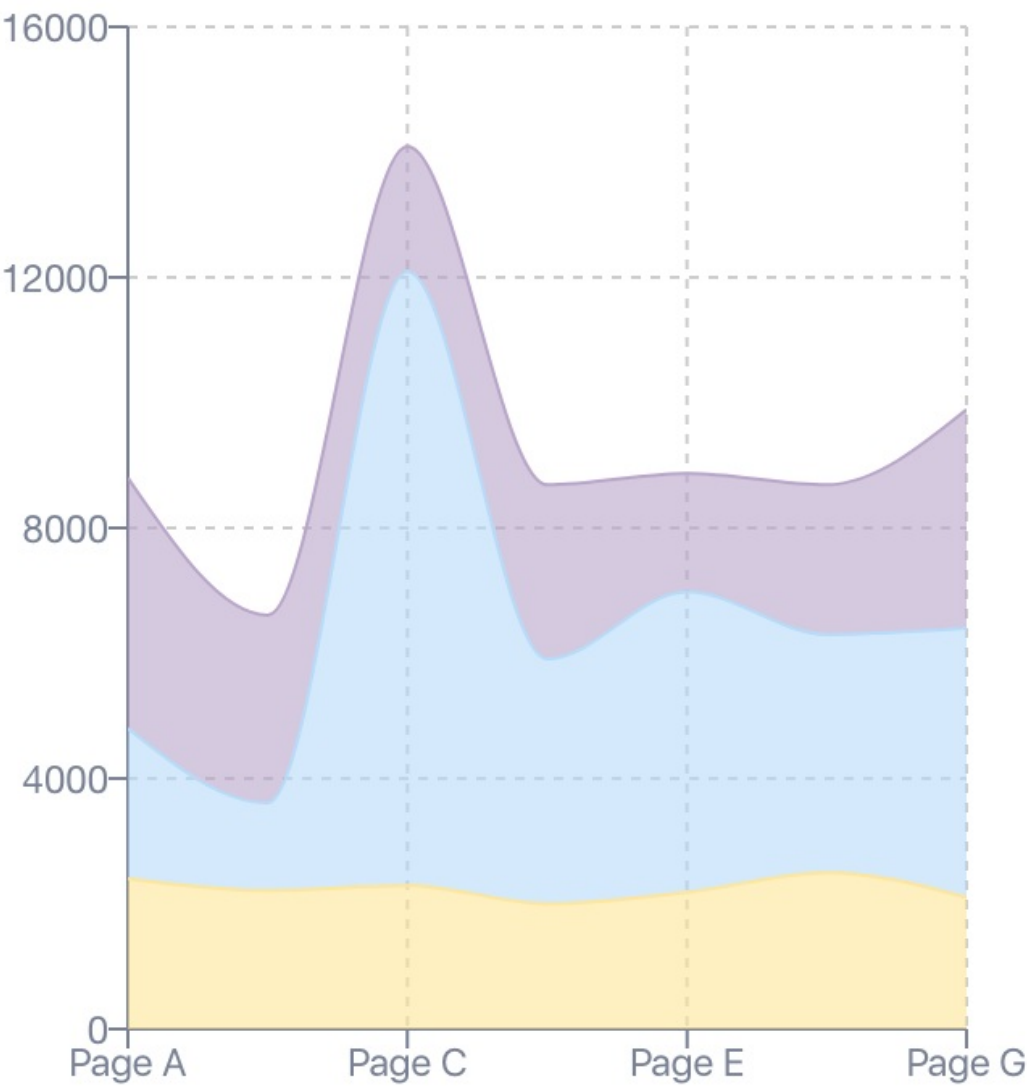
BiaxialBarChart



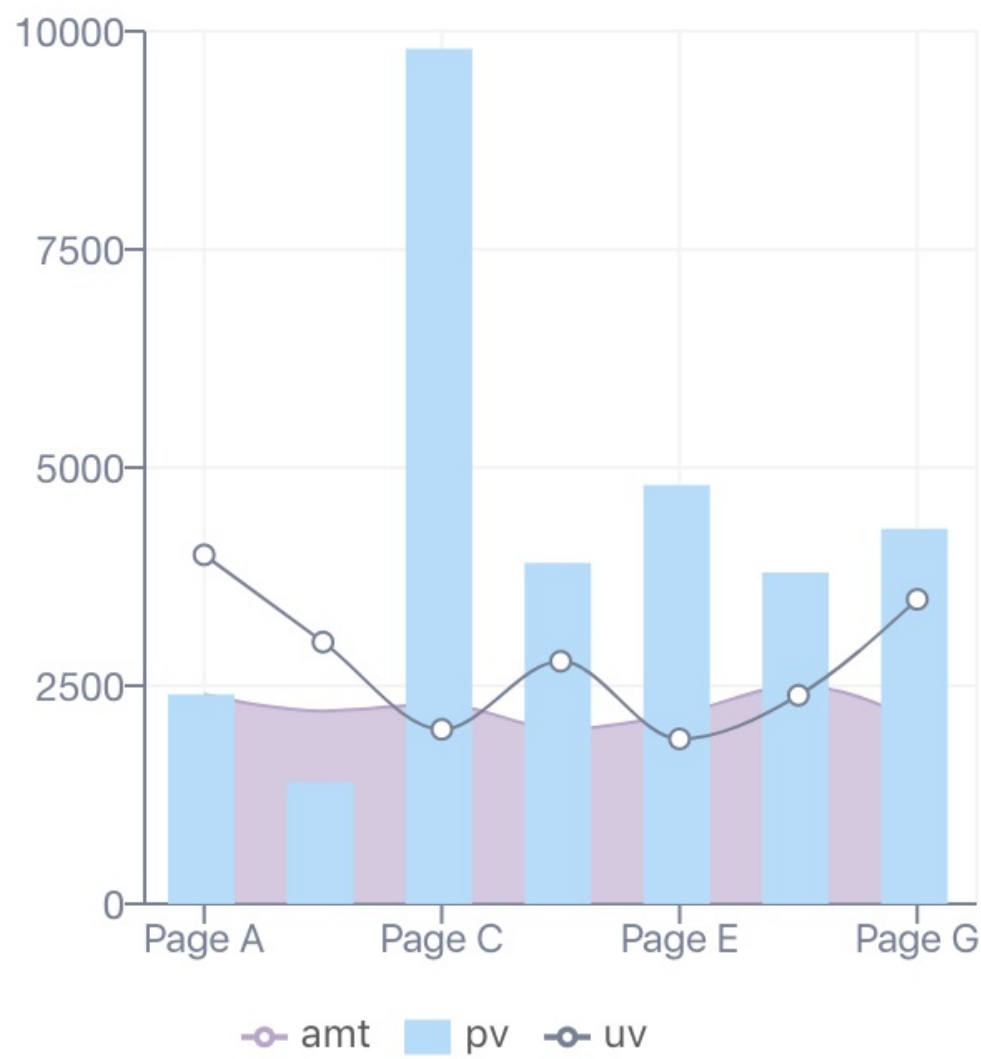
SimpleAreaChart



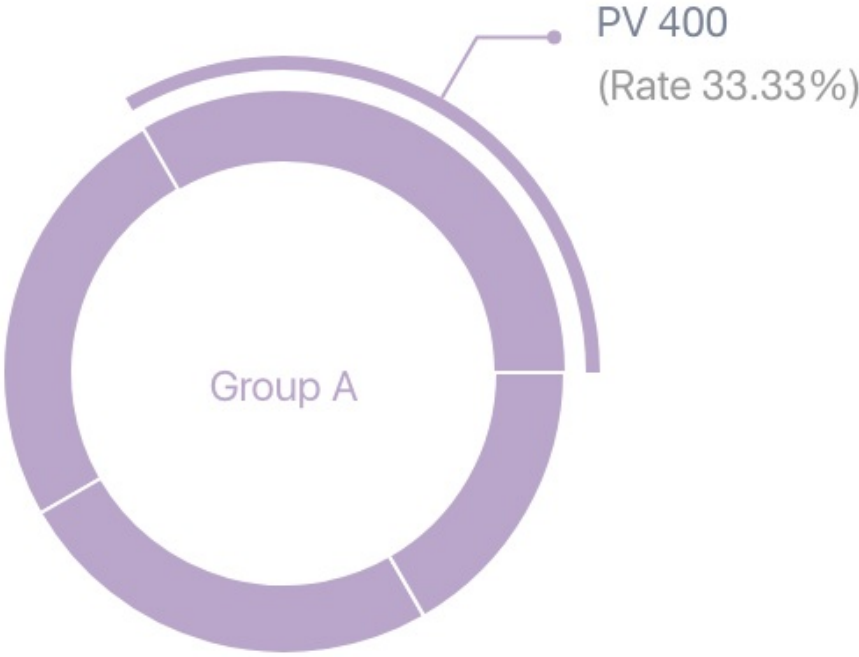
StackedAreaChart



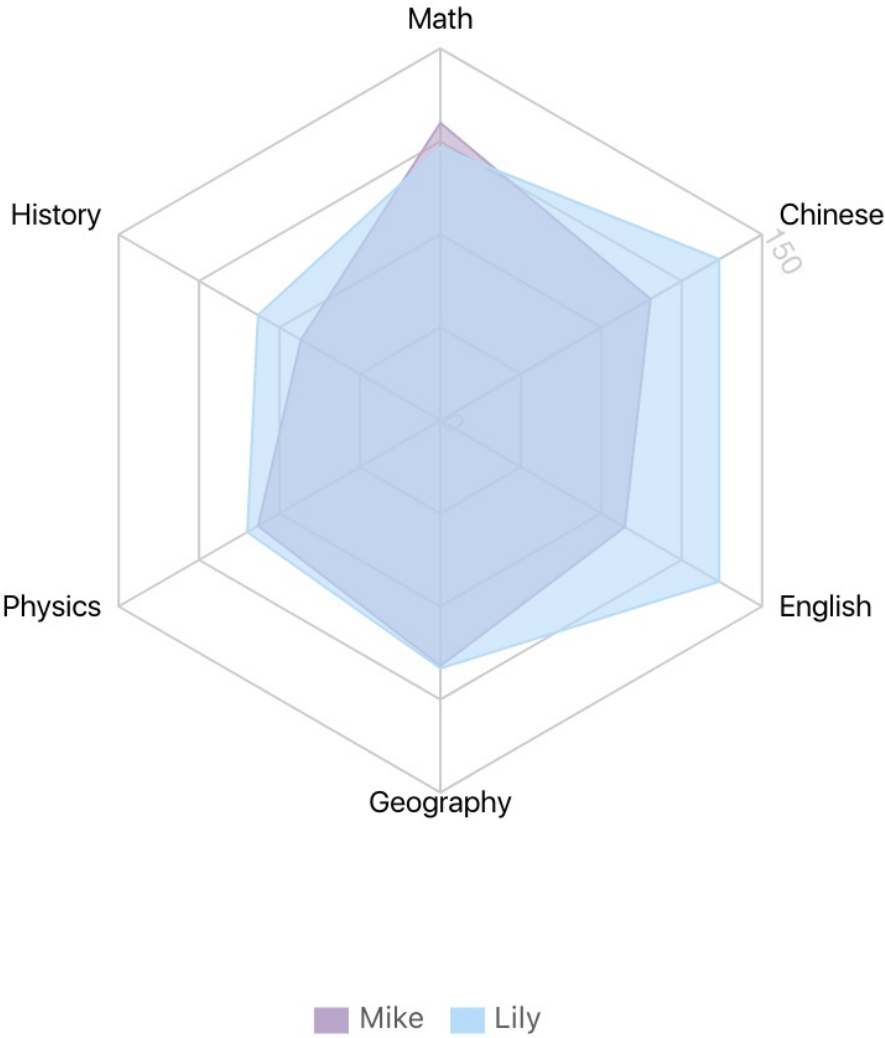
LineBarAreaComposedChart



CustomActiveShapePieChart



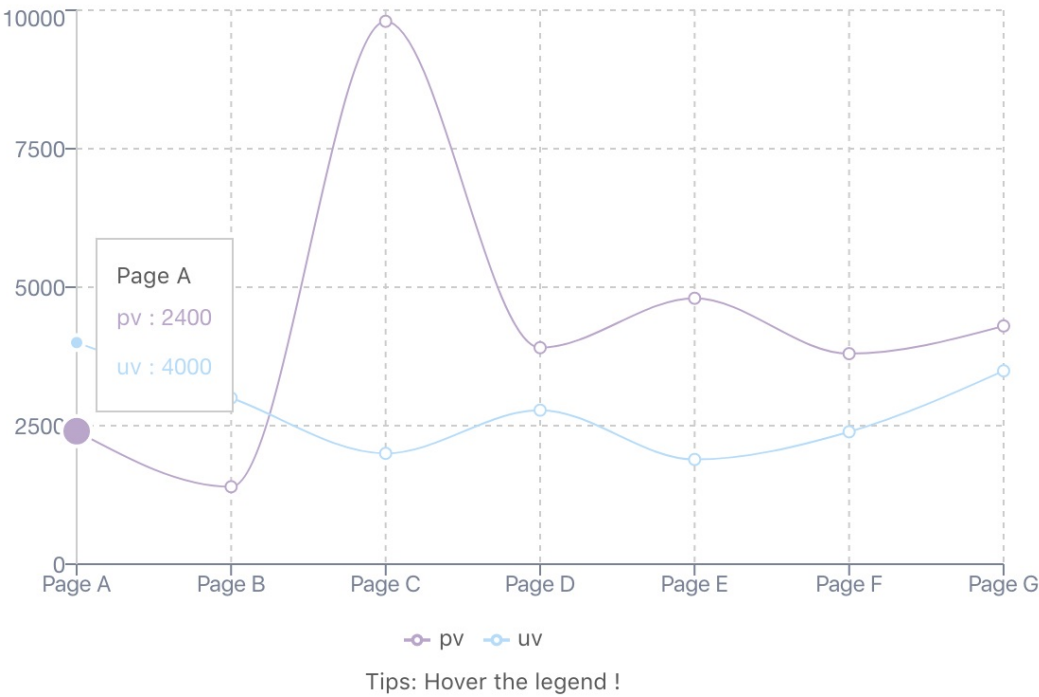
SpecifiedDomainRadarChart



SimpleRadialBarChart



LegendEffectOpacity



ReactVis Charts

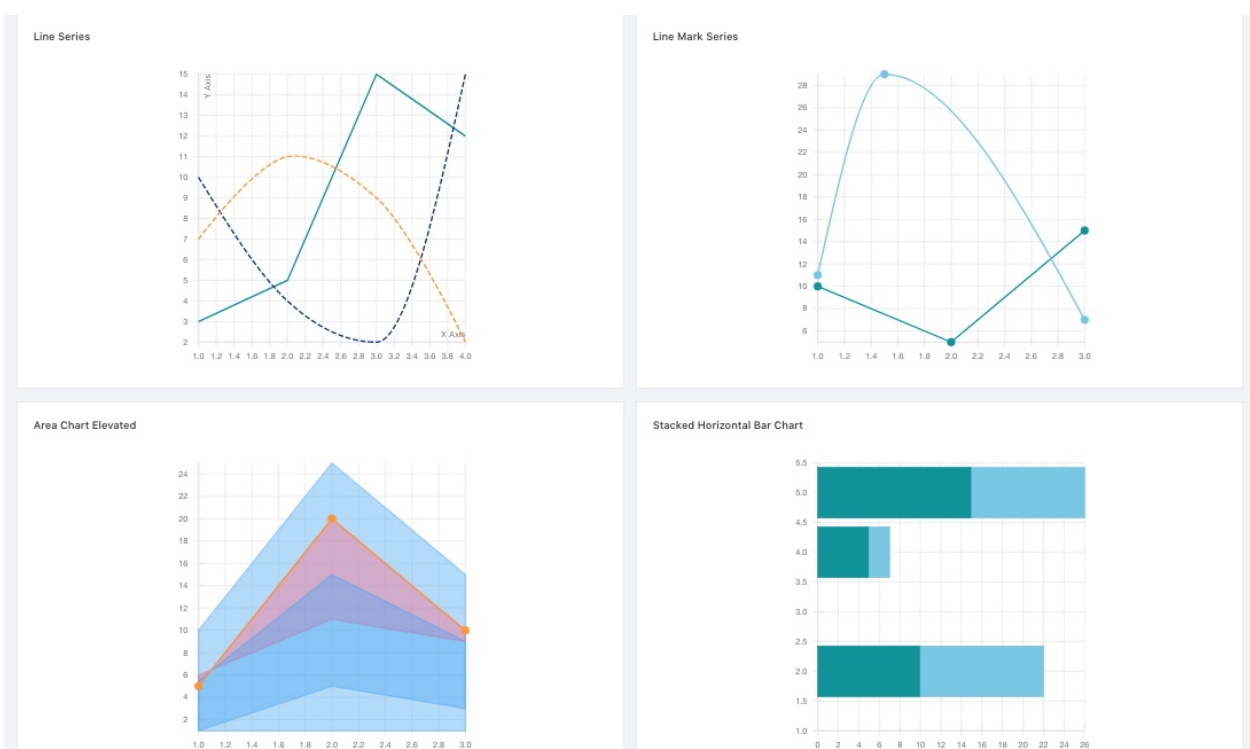
Folder path: `/src/containers/charts/reactVis`

API Link: <https://github.com/uber/react-vis>

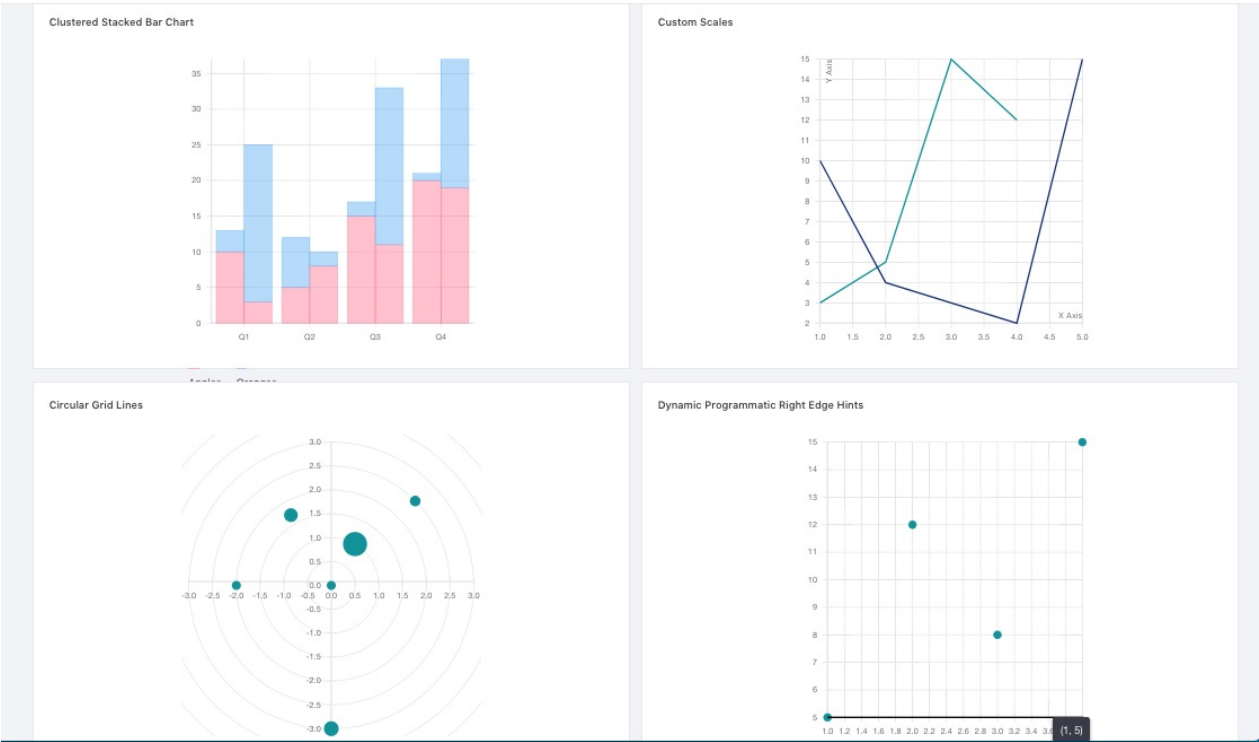
Configuration should be supplied from

`/src/containers/charts/reactVis/config.js`

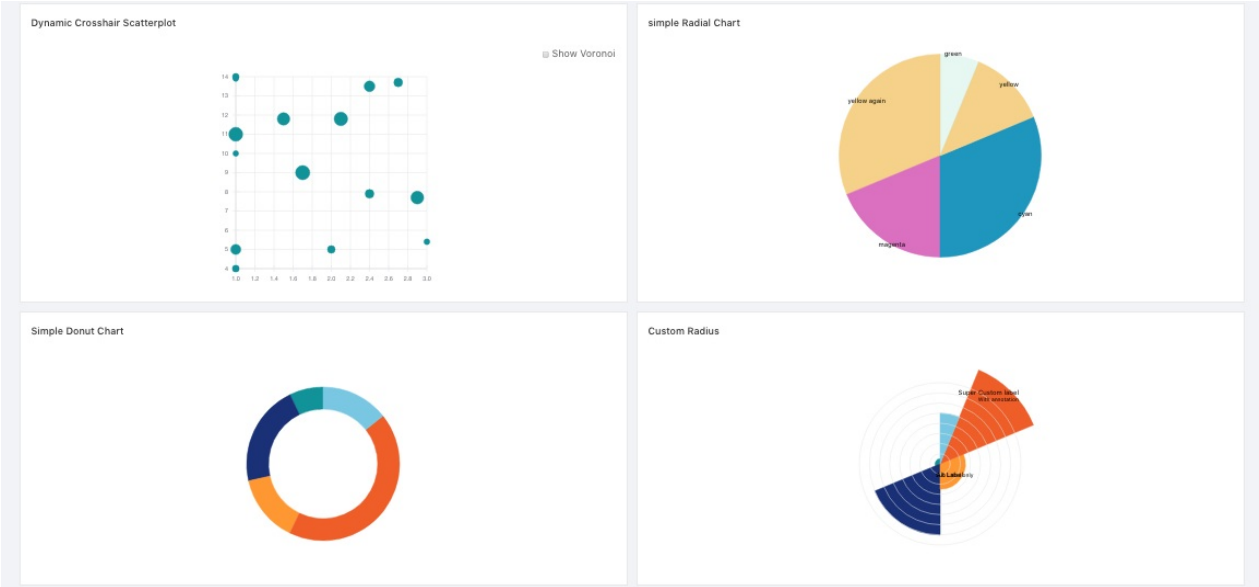
The followings are the images of the reactVis chart.



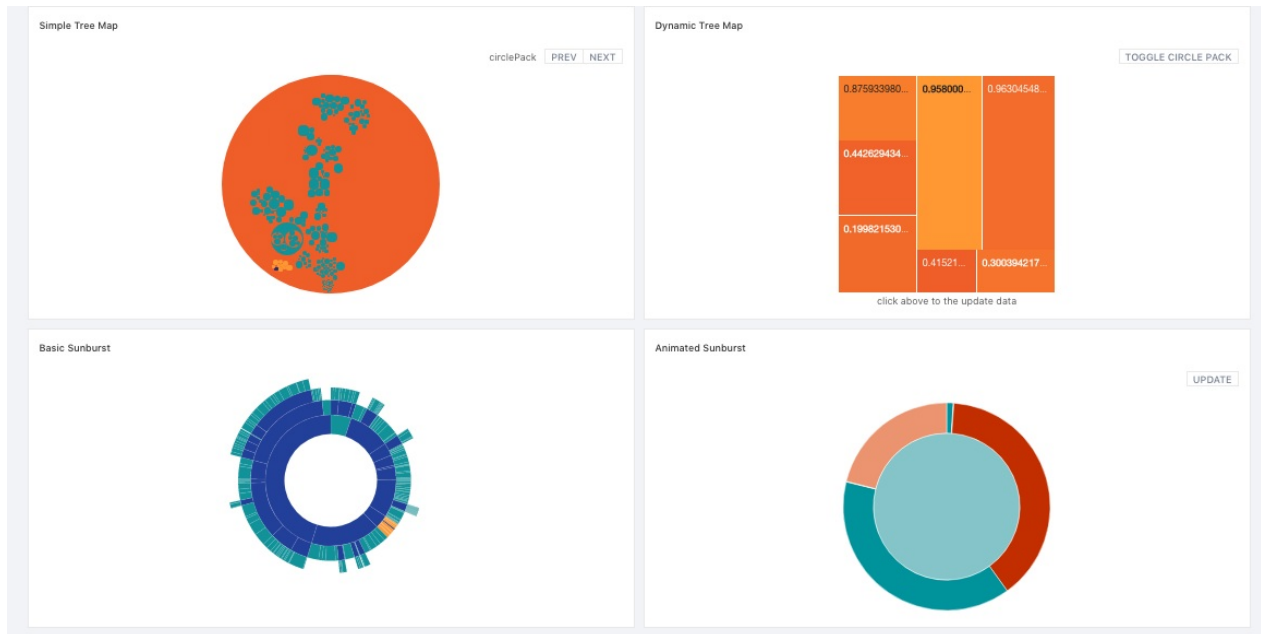
These are the charts of Line Series, Line Mark Series, Area Chart Elevated and Stacked Horizontal Bar Chart.



These are the Charts of Clustered Stacked Bar Chart, Custom Scales, Circular Grid Lines and Dynamic Programmatic Right Edge Hints.



These are the Charts of Dynamic Crosshair Scatterplot, simple Radial Chart, Simple Donut Chart and Custom Radius.



These are the Charts of Simple Tree Map, Dynamic Tree Map, Basic Sunburst and Animated Sunburst.



These are the Charts of Candle Stick, Complex Chart and Stream Graph.

All the charts have the same format.

```
<ReactVisChartType {...config} />
```

Example:

```
<LineSeries {...configs.LineSeries} />
```

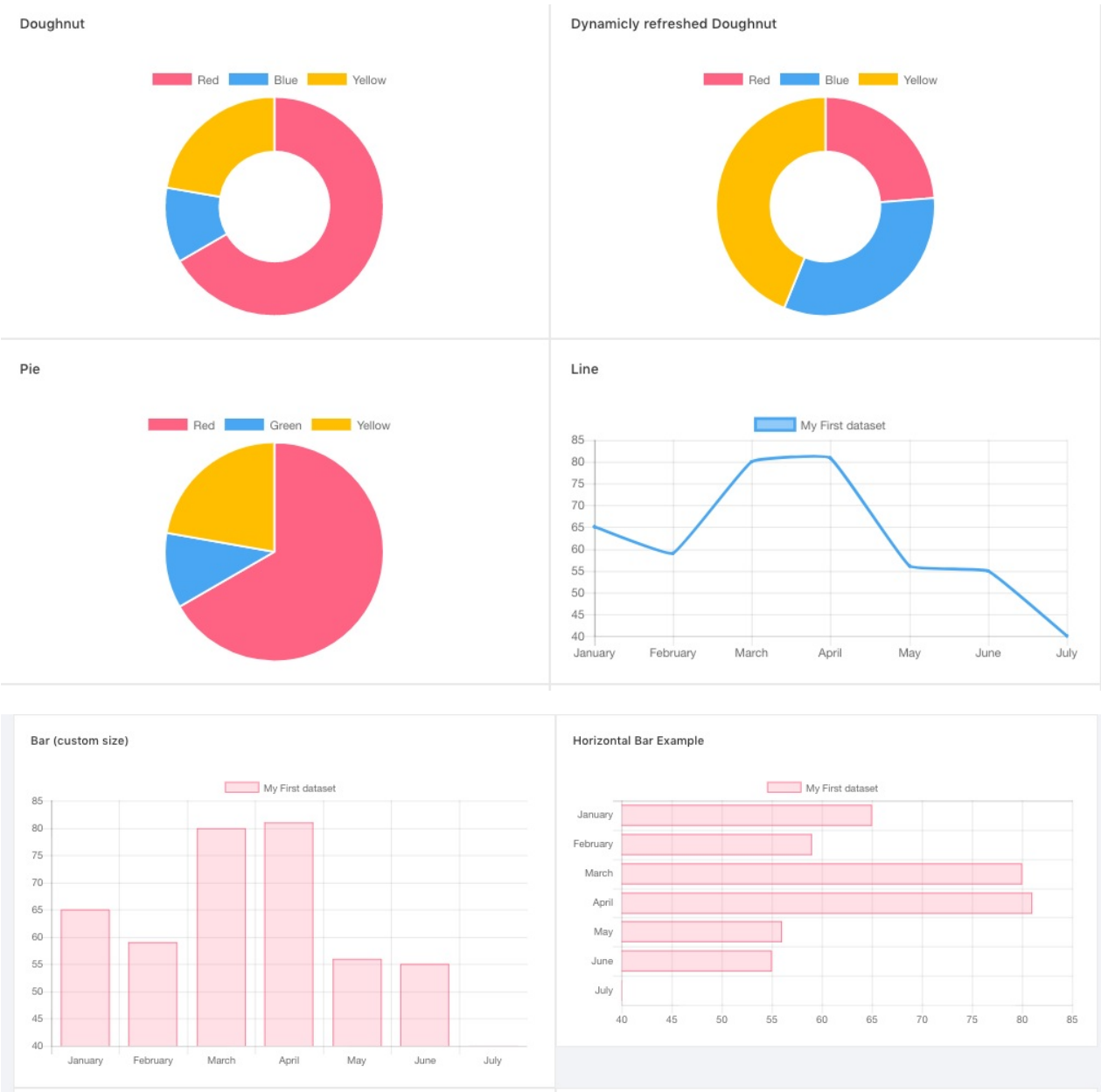
Will be found on `/src/containers/charts/reactVis/config.js`

React-Chart-2

Folder path: /src/containers/charts/reactChart2

API documentation of React-Chart-2: <https://github.com/gor181/react-chartjs-2>

If you want to render React trend chart component like the following image.





All the charts have the same format.

```
<ReactChart2Type {...config} />
```

Example:

```
<Pie data={...} />
```

Will be found on

```
/src/containers/charts/reactChart2/{ChartType}/{ChartTypeconfig}.js
```

Uppy

Uppy is file uploader solution which work can be integrated with any framework. It has functionalities to upload files from local disk, Google Drive, Dropbox, Instagram, remote URLs, cameras etc.

For more introduction visit their website <https://uppy.io/>

Now, Let's move on with our Isomorphic app and Uppy Integration.

We divide our Uppy Integration into 3 steps.

1. UI Layer
2. Configuration
3. Scripting

UI Layer :

Code location : *newdashapp/src/containers/AdvancedUI/uppy/index.js*

Code Integration :

```
<LayoutWrapper>
  <PageHeader>Uppy</PageHeader>
  <Box>
    <ContentHolder>
      <div id="uppyHolder" />
    </ContentHolder>
  </Box>
</LayoutWrapper>
```

Here, **<div id="uppyHolder" />** is the main concern. To integrate Uppy in any UI section you need to declare a div with ID which contains the Uppy UI Holder.

Configuration :

Code Location : *newdashapp/src/containers/AdvancedUI/uppy/config.js*

Code Integration :


```
const config = {
  target: '#uppyHolder',
  // trigger: '#UppyModalOpenerBtn',
  endpoint: 'fakeServer',
  DashboardInline: true,
  Webcam: true,
  GoogleDrive: true,
  Dropbox: true,
  Instagram: true,
  autoProceed: false,
  restrictions: {
    maxFileSize: 1000000,
    maxNumberOfFiles: 3,
    minNumberOfFiles: 2,
    allowedFileTypes: ['image/*', 'video/*']
  },
  metaFields: [
    {
      id: 'resizeTo',
      name: 'Resize to',
      value: 1200,
      placeholder: 'specify future image size'
    },
    {
      id: 'description',
      name: 'Description',
      value: 'none',
      placeholder: 'describe what the file is for'
    }
  ]
};
export default config;
```

Code Explanation :

Code (Key)	Parameter Type	Description
target	string	id of the <div> which render the Uppy UI Layer
endpoint	string	server connection url
trigger	string	Modal View Opener
DashboardInline	boolean	To show the Uppy Dashboard in current page or not
Webcam	boolean	To use file uploader from webcam
GoogleDrive	boolean	To use file upload from GoogleDrive
Dropbox	boolean	To use file upload from Dropbox
Instagram	boolean	To use file upload from instagram
autoProceed	boolean	Control the direct upload or not
maxFileSize (restrictions)	numeric value	Control the maximum file upload size
maxNumberOfFile (restrictions)	numeric value	control the maximum number of file upload at a time
minNumberOfFile (restrictions)	numeric value	Control the minimum number of file upload at a time
allowedFileTypes	array	contains the file types allowed to be uploaded. Example: image, video, xml etc
id (metaFields)	string	id of the field after uploading, during editing the images
name (metaFields)	string	name of the field during editing
value (metaFields)	numeric	Resize to value
placeholder (metaFields)	string	placeholder for the metaFields

Now, export the config object.

Scripting :

Code location : *newdashapp/src/containers/ui/elements/uppy.js*

Code Integration :

```
const SERVER = null;
```

All the JS code is done here. User just need to integrate the server in this line.

Chat

Folder path: `/src/containers/Chat`

If you want to render `Chat` component for different Views

Desktop View

Folder path: `/src/containers/Chat/destopView.js`

Tab and Mobile View

Folder path: `/src/containers/Chat/mobileView.js`

Major Components

Component	Description
Chatroom	list of chats
Messeges	list of messages of single chatroom
ComposeMessage	to compose message
ViewProfile	to view sender and receiver profil
ComposeMessage	to compose message

Firestore Credential

To Use the Firestore Api you need to configure your app to the [Firestore Official Website](#) first. And put your app credentials to the config file of our app.

Path to the config file: `/src/settings/index.js`

The following are the important Credentials you must provide in order to make Firestore Authentication work.

Keys
apiKey
authDomain
databaseURL
projectId
storageBucket
messagingSenderId

firebase data structure

```
{
  users: [
    {
      key: 'wt4TiasxgPrQ3dNwVZ55',
      data: {
        dob: '06-Apr-1993',
        gender: 'Male',
        language: 'Burmese',
        mobileNo: '5726784596',
        name: 'Zondra Kulic',
        profileImageUrl: 'https://s3.amazonaws.com/redqteam.com/mateadmin/support-male-zonra.png'
      },
    },
    ...
  ],
  chatRooms: [
    {
      key: '-L2ZNtIAFMPGa_Me56YN',
      data: {
        id: '-L2ZNtIAFMPGa_Me56YN',
        lastMessage: '',
        lastMessageTime: 0,
        otherUserId: '3M9ySG4N5RBGYmmiopy5',
        userId: '0lAR4PcX71m7MPOTRIFF'
      }
    },
  ],
}
```

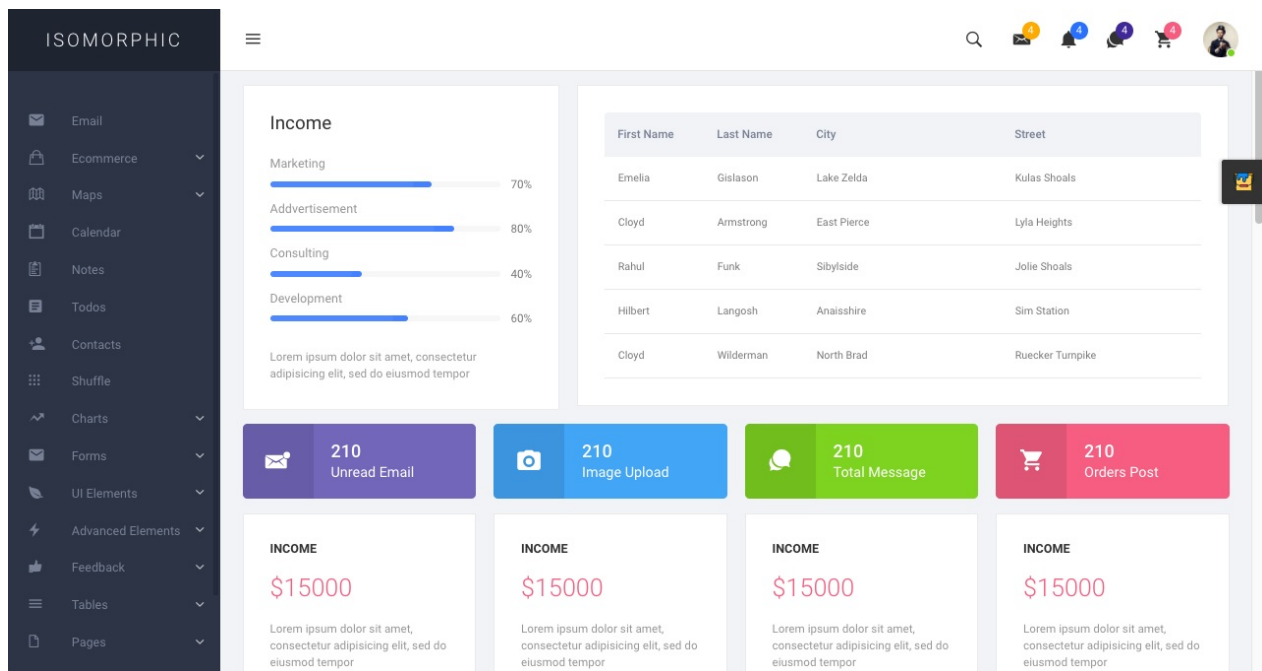
```
    ...
  ],
  messages: [
    {
      key: '-L2Z0XGKzP_GJypCLMor',
      data: {
        chatRoomId: '-L2ZNtIMQzKe_P5i5wRE',
        messageTime: 1515660387407,
        sender: '3M9ySG4N5RBGYmmiopy5',
        text: 'hello there'
      }
    },
    ...
  ]
}
```

LTR -> RTL (css) transformation

Isomorphic supports LTR (left to right) view to RTL (right to left) transformation.

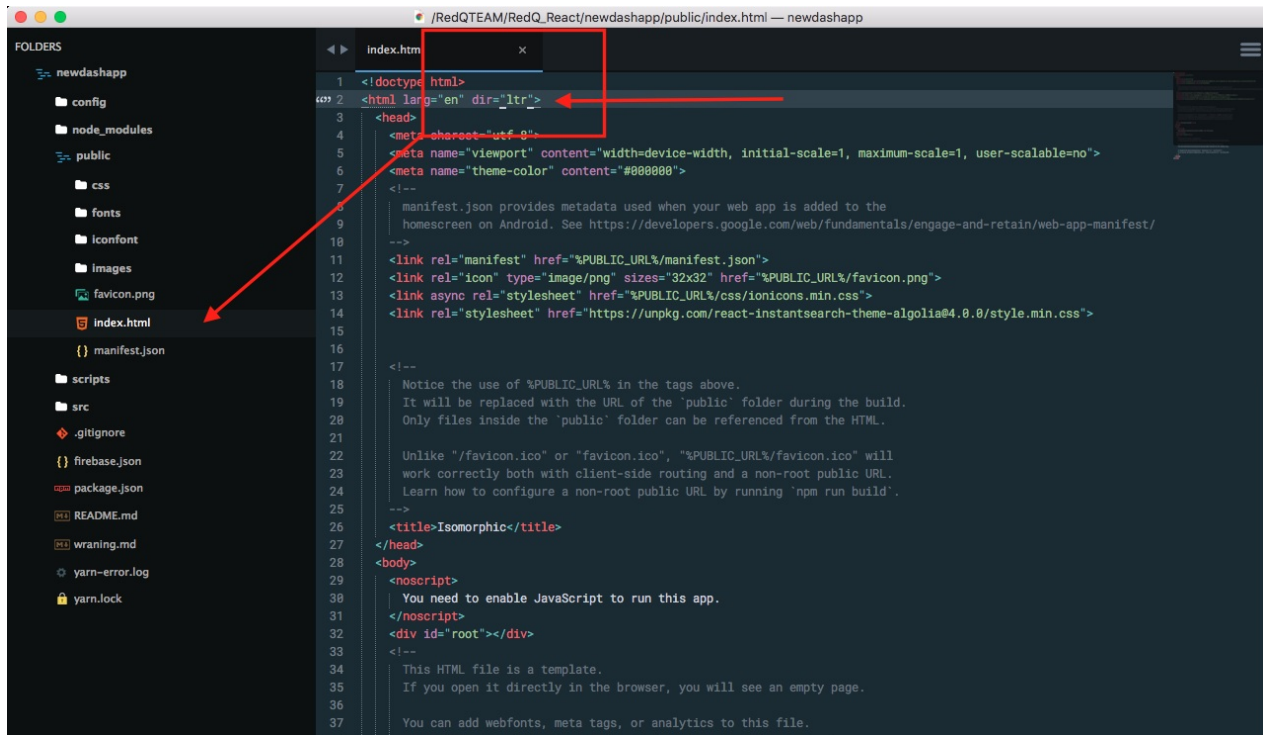
The transformation process is very easy to use.

Here is the dashboard overview in **LTR** mood :

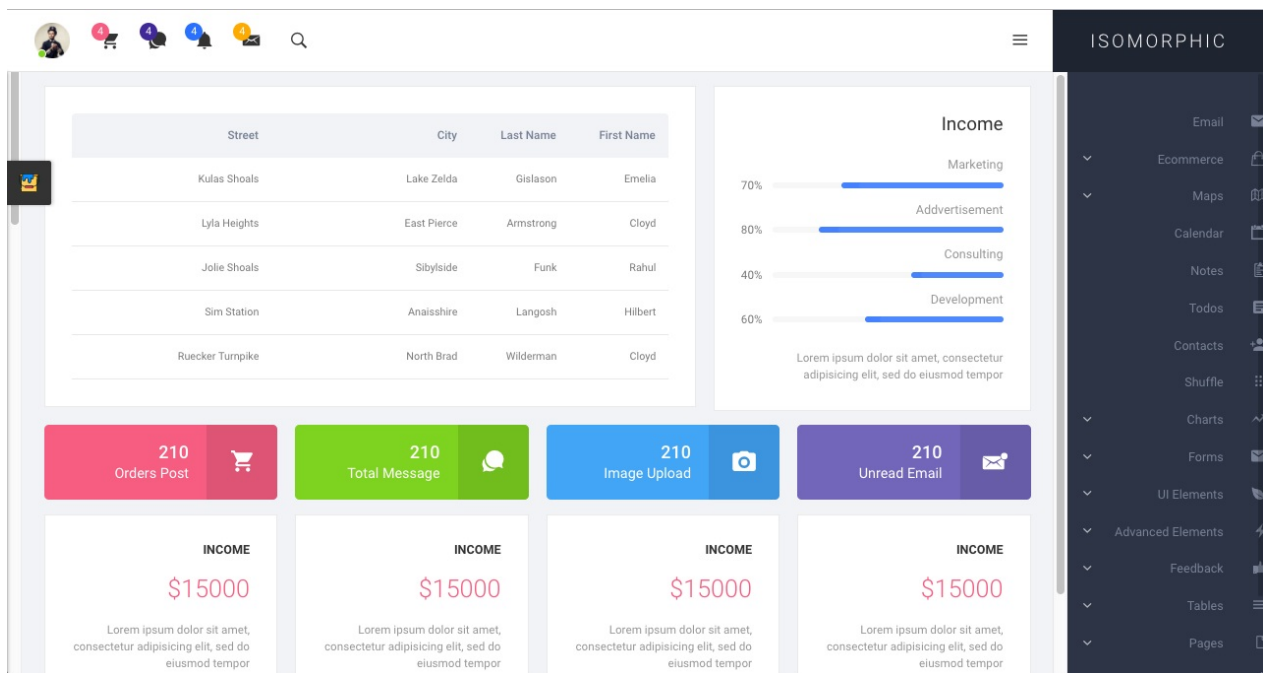


To transform dashboard into **RTL** view the process is :

1. Go to public folder
2. index.html file
3. change the `dir="ltr"` to `dir="rtl"` (please see the screenshot)



Now the dashboard view will be like this.



Thanks.

Multi Language Support.

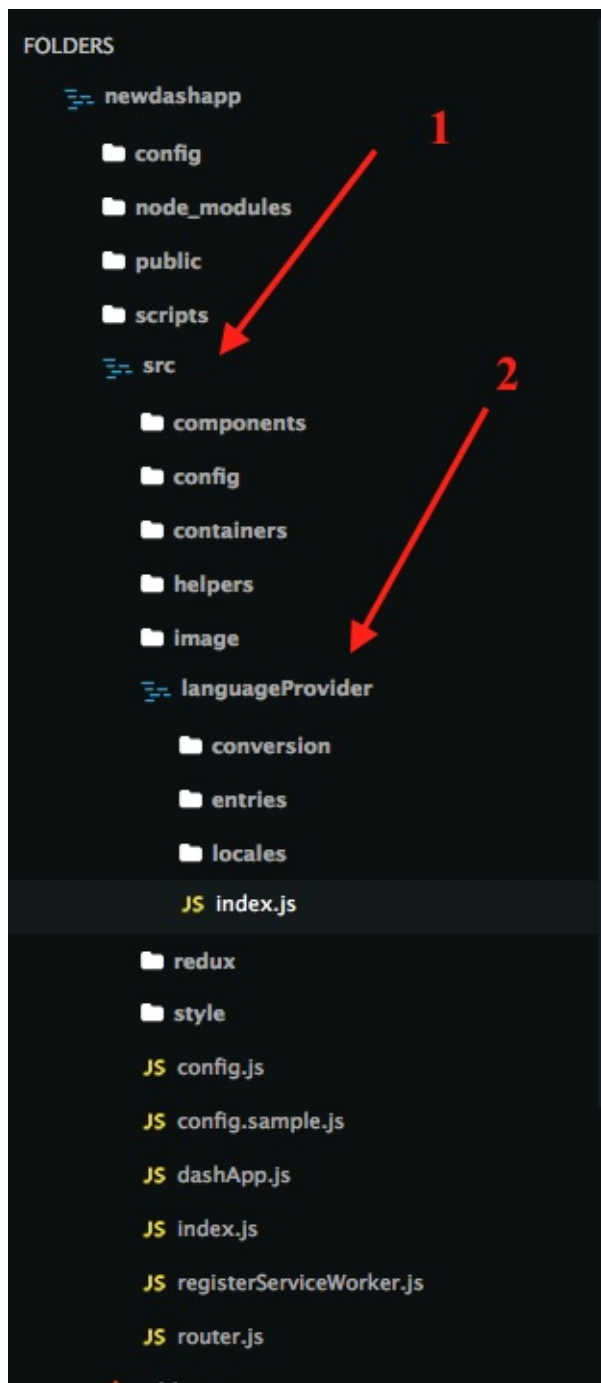
Isomorphic supports multi language. It became very useful to people from around the world.

For multi-language conversion Isomorphic use <https://github.com/yahoo/react-intl> library.

Lets discuss the procedure step by step of how to Add/Convert into new languages.

Step 1 :

Go to Project Root folder > `src` folder > `languageProvider` folder



Open `index.js` file and Add the language name into `const AppLocale` which you want to add into Isomorphic App.

```
const AppLocale = {  
  en: Enlang,  
  zh: Zhlang,  
  sa: Salang,  
  it: Itlang,  
  es: Eslang,  
  fr: Frlang  
};
```

for example you want to Add/Convert German language.

then add `de` language code into AppLocale constant like this.

```
const AppLocale = {  
  en: Enlang,  
  zh: Zhlang,  
  sa: Salang,  
  it: Itlang,  
  es: Eslang,  
  fr: Frlang,  
  de: GermanLang  
};
```

and Add this code `addLocaleData(AppLocale.de.data);`

Step 2 :

create a new file in `src > languageProvider > entries > de_DE.js` file and paste this code.

```
import antdSA from 'antd/lib/locale-provider/de_DE';
import appLocaleData from 'react-intl/locale-data/de';
import deMessages from '../locales/de_DE.json';

const deLang = {
  messages: {
    ...deMessages
  },
  antd: antdDE,
  locale: 'de-DE',
  data: appLocaleData
};
export default deLang;
```

Step 3 :

Now, create a JSON file named `de_DE.json` in `src > languageProvider > locales` folder.

Step 4 :

This is the big step for the language conversion. To change language in every Isomorphic components this is the general procedure. Let's discuss an **Alert box** component inside the **FeedBack** section.

Step 4.1

copy all the element from *en_US.json _file and paste this into de_DE.json _file*. Now all the text strings are listed there. Just translate every json element's value into German language.

Example :

`"sidebar.formsWithValidation": "Forms With Validation"` will be convert like this

`"sidebar.formsWithValidation": "Formulare mit Validierung"`

Step 5 :

To add this new language into Sidebar switcher option, follow this file path `src > containers > languageSwitcher > config.js` file.

add the new icon image from image folder in the top of the file like this `import italianLang from '../..../image/flag/german.svg';`

Then add this new Language into config -> option array by this

```
{
  languageId: 'german',
  locale: 'de',
  text: 'German',
  icon: germanLang,
},
```

Step 6 (additional step) :

To chose the default language of Isomorphic go to this file `src > config.js` and change this code `const language = 'english';` into `const language = 'german';`

This will initiate german language as default language.

Ta-Da!!

Thanks for reading & understanding the process. It's not that hard. For any inquiries or issues you can contact with us via our support portal

<https://redqsupport.ticksy.com/>

Express JWT Implementation

Server Side

In this section we will show you how to implement JSON Web Token (JWT) using Node Express framework.

First, Open the `isomorphic-express` folder in your favorite editor. where you will find the below folder structure



You can see we have provided an extra `server` folder where we have implemented our **Express JWT Authentication**.

To Run the Server follow the below steps,

i) Run `yarn` at the terminal in the `isomorphic-express -> server` directory. (This will bring all the necessary node modules)

ii) Then Run the `yarn start` command

if you have done the above steps correctly then you will see the below screen,



ok, now our server is running successfully at 9000 port,

Lets see what's inside the `server` folder & what's happening there:



From the above snap you can see that theres a `server.js` & `middleware.js` file where we have implemented the necessary server & middleware coding for the JWT implementation.

If you open the `server.js` file in the first part you will see we have declared some necessary requirement,

```
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const jsonwebtoken = require('jsonwebtoken');
4 var cors = require('cors');
5 const Config = require('./config');
6 const Middlewares = require('./middleware');
7
8 const { port, secretKey, expiredAfter } = Config;
9 const { authenticate, authError } = Middlewares;
10 const app = express();
11 app
12   .use(bodyParser.urlencoded({ extended: true }))
13   .use(bodyParser.json())
14   .use(cors())
15   .use('/api', authenticate, authError);
```

at line 5 & 6 we have required the configuration from the `config.js` file where have declared the `port` , `secret key` , `expiration time` & `middleware.js` file where we have implemented the **Token Authorization**, **Token Expiration** and **Error checking**.

let's have a look at the `middleware.js` file

```
1 const jwt = require('jsonwebtoken');
2 const Config = require('./config');
3
4 const { secretKey } = Config;
5
6 const authenticate = (req, res, next) => {
7   const token = req.headers.authorization || '';
8   jwt.verify(token, secretKey, (error, decoded) => {
9     if (error) {
10       next({ error: 'token varified failed' });
11     } else {
12       const { expiredAt } = decoded;
13       if (expiredAt > new Date().getTime()) {
14         next();
15       } else {
16         next({ error: 'token expired' });
17       }
18     }
19   });
20 };
21
22 const authError = (err, req, res, next) => {
23   res.json(err);
24 };
25
26 module.exports = { authenticate, authError };
```

The `authenticate` function checks the `token authorization` & `token expiration` . while the `authError` function is doing the error checking if something went wrong.

From the `server.js` line no 13 you can see that the two middlewere `authenticate` & `authError` is used by our `app` at the `\api` route. so whenever you want to access this route you have to generate the valid token.

Now, let's have a look at the later part of the of the `server.js` file


```
17 app.post('/login', (req, res) => {
18   const { username, password } = req.body;
19   const response = {};
20   if (username === 'demo@gmail.com' && password === 'demodemo') {
21     response.token = jwt.sign(
22       {
23         expiredAt: new Date().getTime() + expiredAfter,
24         username,
25         password,
26         id: 1
27       },
28       secretKey
29     );
30   } else {
31     response.error = 'Not found';
32   }
33   res.json(response);
34 });
35
36 app.post('/api/demoTesting', (req, res) => {
37   res.json({
38     status: 200,
39     message: 'succesful'
40   });
41 });
42 app.listen(port, () => {
43   console.log('Isomorphic JWT login ' + port);
44 });
```

you can see here we have implemented the `/login` related functionality at the beginning of this part and later we have created a

demo testing post request at `/api/demoTesting/` route.

Let's see what we have done at the `/login`

Here we have generated the `token` in the response when the login is successful.

Client Side

For the client part if you are already familiar with our Isomorphic codebase than all you have to do is to check the below file,

`isomorphic-express/src/helpers/authHelper.js` file ,

where you can find the necessary client side coding,

you can also check the,

`isomorphic-express/src/redux`

where we have done the reducer & saga related code for the jwt authentication.

If you are not familiar with the Isomorphic codebase than we suggest you to check our previous section of this documentation.

Deployment

`yarn build` or `npm run build` creates a `build` directory with a production build of your app. Set up your favourite HTTP server so that a visitor to your site is served `index.html`, and requests to static paths like `/static/js/main.<hash>.js` are served with the contents of the `/static/js/main.<hash>.js` file.

For more information visit [create-react-app](#)

For deploying your site in firebase visit [Firebase Deployment](#) page

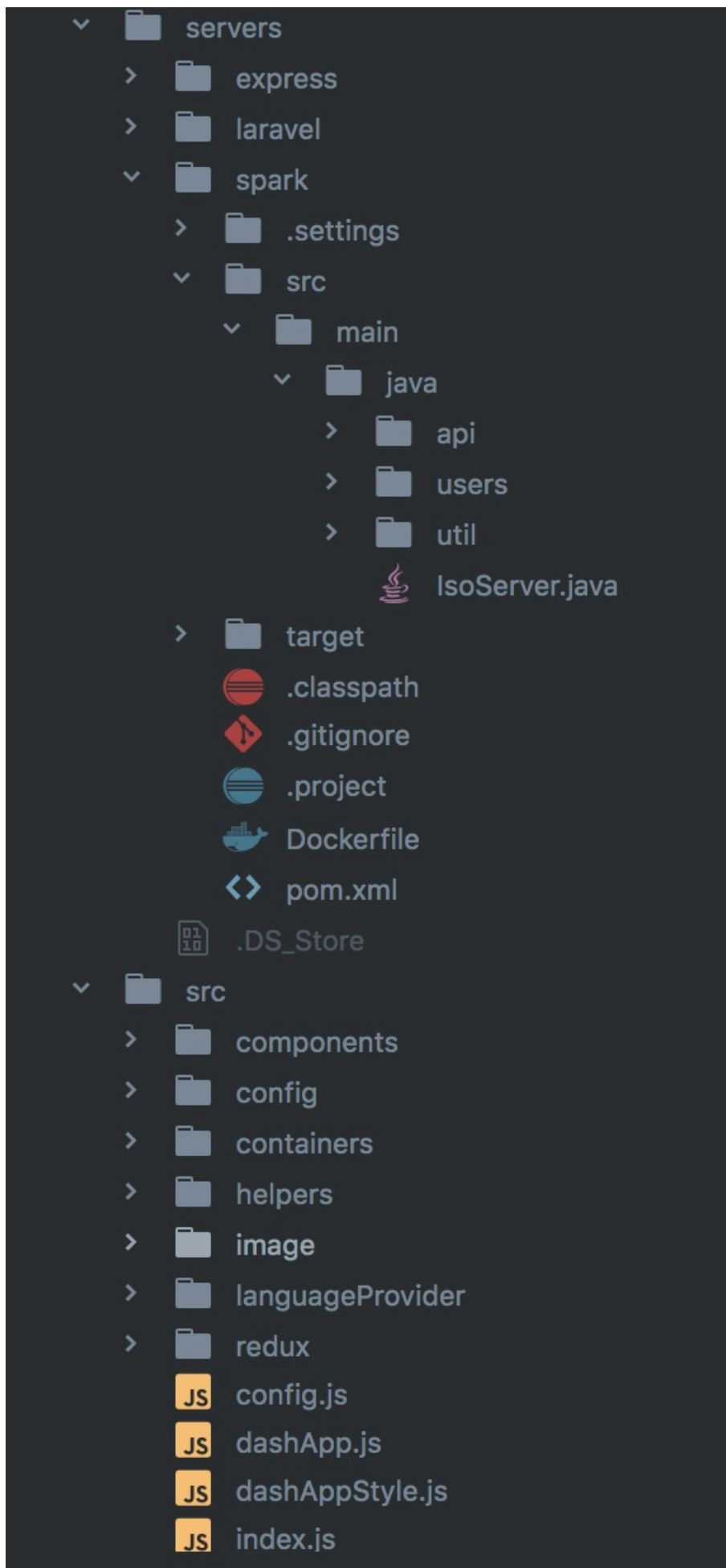
Spark JWT Implementation

Server Side

In this section we will show you how to implement JSON Web Token (JWT) using Spark java framework.

To implement the framework you have to download [Eclipse](#) or [IntelliJ](#) . Here implementation is done by Eclipse.

First, Open the `isomorphic-servers` folder in your favorite editor. where you will find the below folder structure



You can see we have provided an extra `servers/spark` folder where we have implemented our **Spark JWT Authentication**.

Docker Implementation

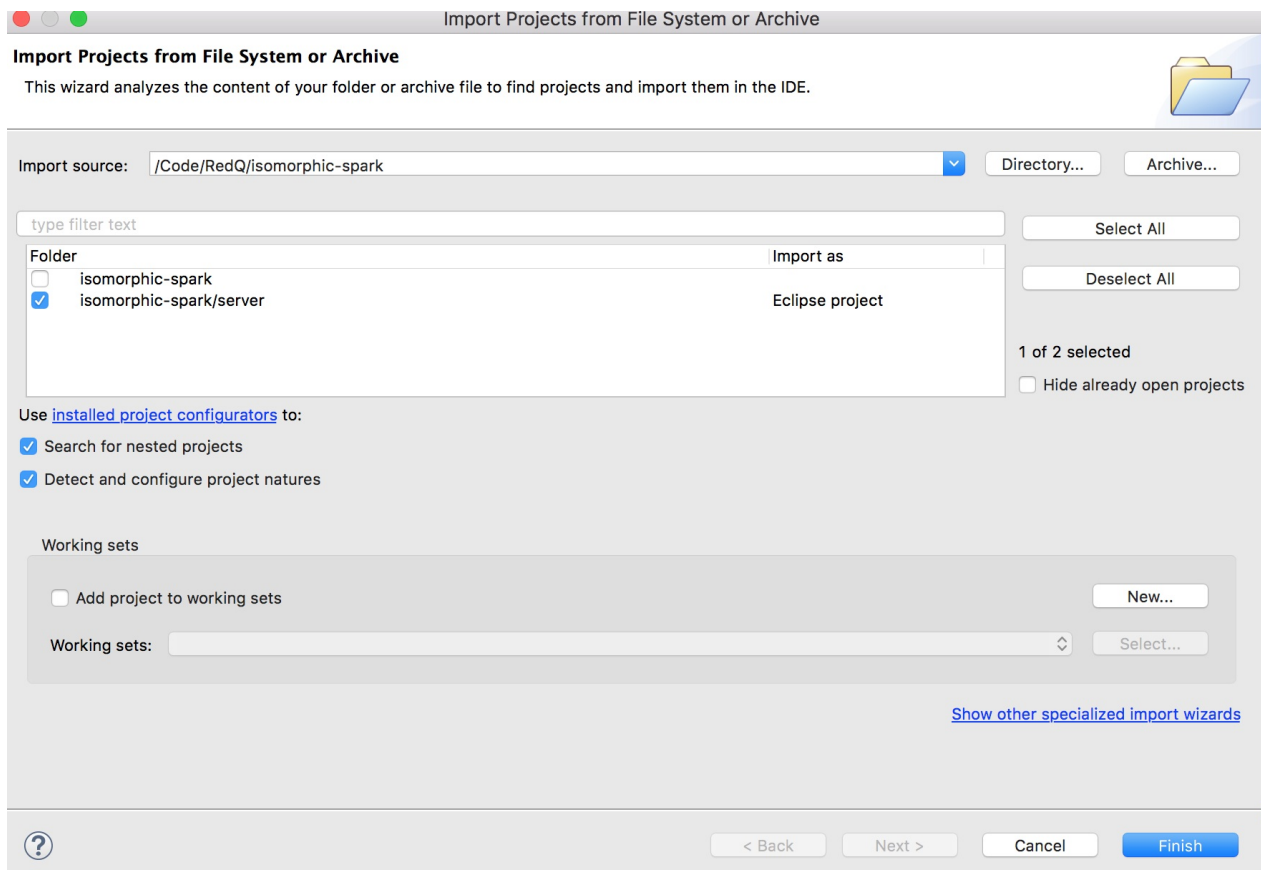
You can directly run it on docker. For that go to `isomorphic-servers/spark` location in terminal and write following codes

```
docker build -t isoserver .  
docker run -d -p 9000:9000 isoserver
```

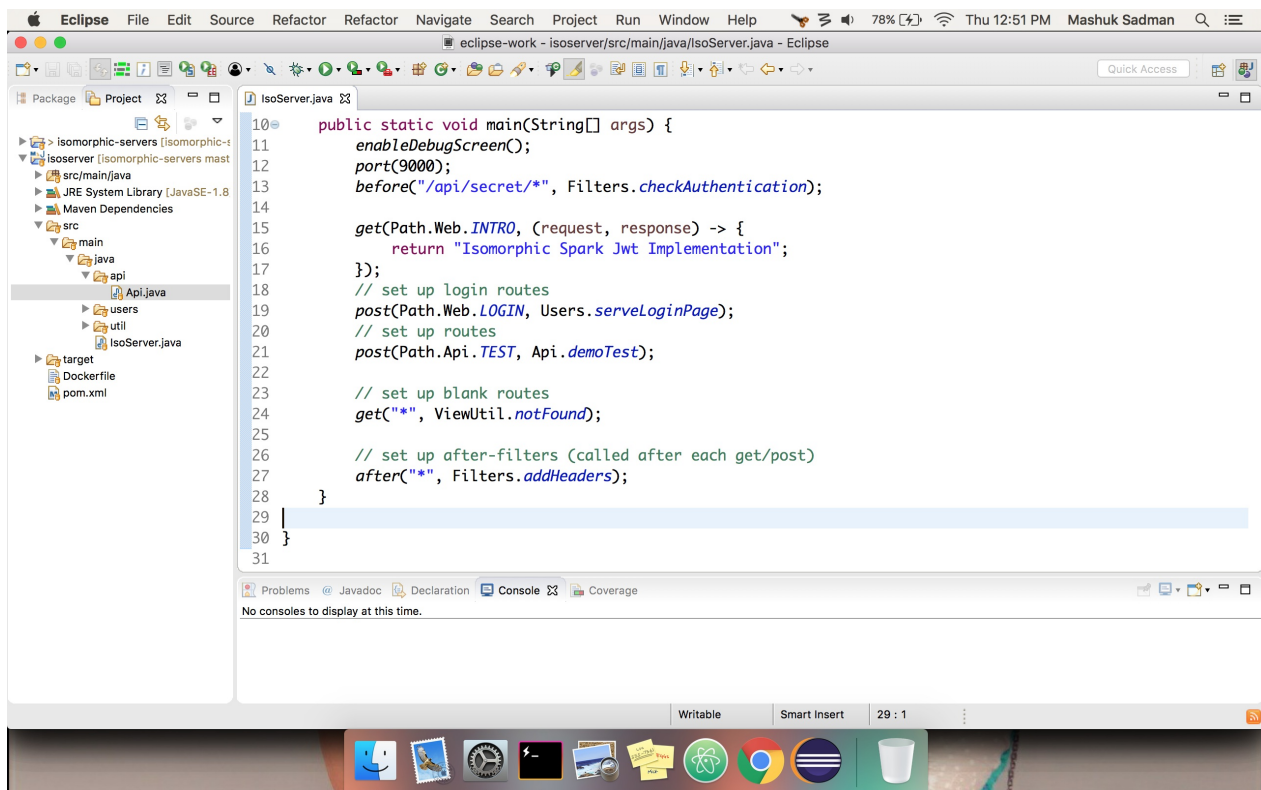
IdE Implementations

To Run the Server follow the below steps,

i) Open your IdE(Here Eclipse. Go to `File->Open Projects File From File Systems` and select `isomorphic-servers/spark` location.



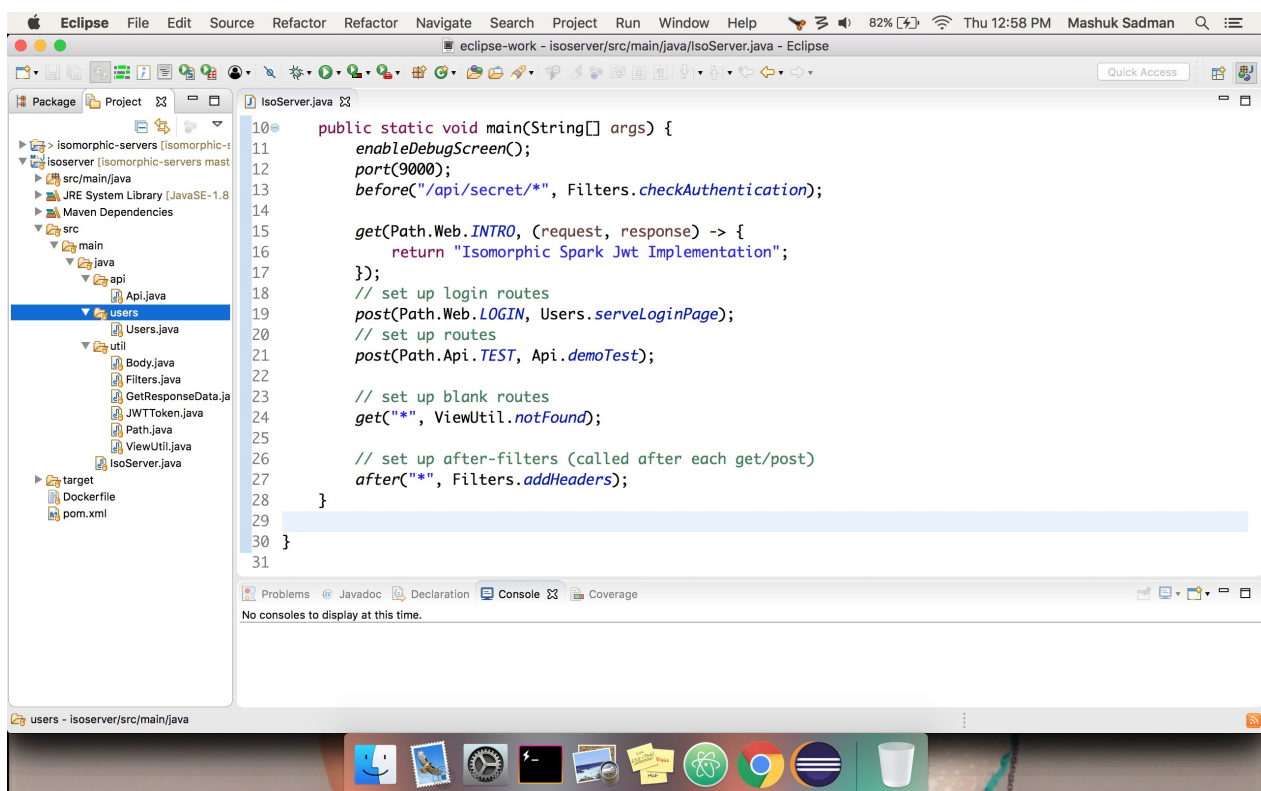
ii) In your editor you will see the project



iii) At last run the application

ok, now our server is running successfully at 9000 port,

Lets see what's inside the **server folder** & what's happening there:



From the above snap you can see that theres a **src/main/java** folder where all the Java classes are kept.

The Application class holds all the routes

```
public static void main(String[] args) {
    enableDebugScreen();
    port(9000);
    before("/api/secret/*", Filters.checkAuthentication);

    get(Path.Web.INTRO, (request, response) -> {
        return "Isomorphic Spark Jwt Implementation";
    });
    // set up login routes
    post(Path.Web.LOGIN, Users.serveLoginPage);
    // set up routes
    post(Path.Api.TEST, Api.demoTest);

    // set up blank routes
    get("/*", ViewUtil.notFound);

    // set up after-filters (called after each get/post)
    after("/*", Filters.addHeaders);
}
```

A basic routing api app is given in Api class(`java/api/api.java`)

```
public static Route demoTest = (Request request, Response response) -> {
    Map<String, String> model = new HashMap<>();
    model.put("message", "succcesful");
    return ViewUtil.testResponse(request, response, model);
};
```

The User(`java/User`), JWTToken(`java/util/JWTToken`) and Filter(`java/util/Filter`) classes are used for authorizing and generating data

The dependencies are kept in `pom.xml`

For more Spark integration see Spark [documentation](#) and [tutorials](#)

Client Side

For the client part if you are already familiar with our Isomorphic codebase than all you have to do is to check the below file,

`isomorphic-servers/src/helpers/jwtAuthentication.js` file,

where you can find the necessary client side coding,

you can also check the,

`isomorphic-servers/src/redux`

where we have done the reducer & saga related code for the jwt authentication.

If you are not familiar with the Isomorphic codebase than we suggest you to check our previous section of this documentation.

Flask JWT

In the root folder you will get a folder named `servers` in the `servers` folder you will get a folder named `flask`. The structure of the folder is below

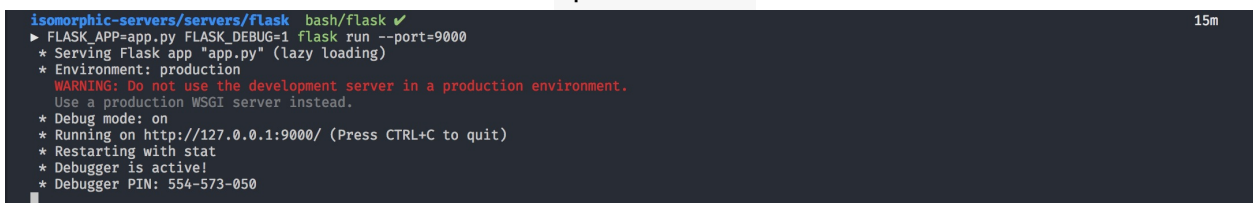


Here You can see the folder the structure where the `app.py` is the file where all the routes are defined and in the `resources.py` file the functionality of the routes.

To start the server run the below command

```
FLASK_APP=app.py FLASK_DEBUG=1 flask run --port=9000
```

It will run the server in the localhost port 9000



We have the below routes

```
api.add_resource(resources.UserLogin, '/api/login')
api.add_resource(resources.TokenRefresh, '/refresh')
# Secret Route
api.add_resource(resources.SecretResource, '/api/secret/test')
```

Login route generate the token.

To modify how you want your user to get their token modify the below function in the `resources.py`

```
class UserLogin(Resource):
    def post(self):
        data = parser.parse_args()
        # Just add any model and you can then check database Use
r
        # current_user = UserModel.find_by_username(data['userna
me'])

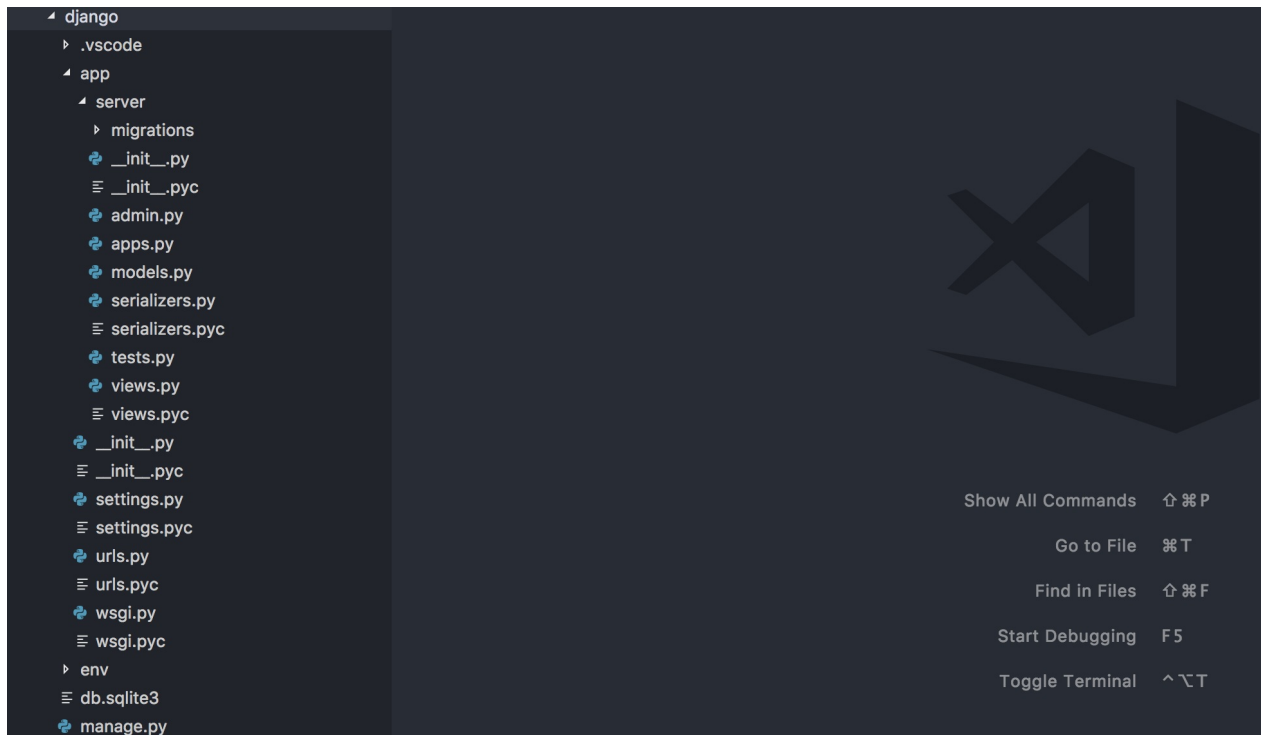
        # static user
        current_user = {
            'username': 'demo@gmail.com',
            'password': 'demodemo'
        }

        if not current_user:
            return {'message': 'User {} doesn\'t exist'.format(d
ata['username'])}

        if data['username'] == current_user['username'] and data
['password'] == current_user['password']:
            access_token = create_access_token(identity=data['us
ername'])
            return {
                'token': access_token,
            }
        else:
            return {'message': 'Wrong credentials'}
```

Django JWT

In the root folder you will get a folder named `servers` in the `servers` folder you will get a folder named `django`. The structure of the folder is below



To start the server run the below command

```
python manage.py runserver 9000
```

```
isomorphic-servers/servers/django bash/flask ✓ 22m
> python manage.py runserver 9000
Performing system checks...

System check identified no issues (0 silenced).
June 27, 2018 - 12:59:35
Django version 1.11.13, using settings 'app.settings'
Starting development server at http://127.0.0.1:9000/
Quit the server with CONTROL-C.
```

All the routes are in `app->urls.py` file. You will get below routes in this file

```
# Auth Route
url(r'^api/secret/test', include(router.urls)),
# No Authentcation required
url(r'^api/login', obtain_jwt_token),
url(r'^api/refresh', refresh_jwt_token),
url(r'^api/verify', verify_jwt_token),
```

We have given a user model you test it. Just check the `app->server->serializer.py` and the views `app->server->views.py`

`serializers.py` User Serializer below

```
class UserSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = User
        fields = ('url', 'username', 'email')
```

User Views we Have used Django default viewset `app->server->views.py`

```
class UserViewSet(viewsets.ModelViewSet):
    """
    API endpoint that allows users to be viewed or edited.
    """
    queryset = User.objects.all().order_by('-date_joined')
    serializer_class = UserSerializer
```

MOdify all of these to your need.

Important Notice:

For testing django server you need to change a few code as we have tested the server for GET method. So in the root directory

`isomorphic-servers/src/helpers/authHelper.js` Replace the below code

```
return await SuperFetch.post('secret/test', { token })
```

With below code

```
return await SuperFetch.get('secret/test', { token })
```

Here `post` has been replaced with `get` .

Another important thing is understanding the below settings. which resides in `isomorphic-servers/servers/django/app/settings.py` path.

Here the below settings can be changed on your need

```
JWT_AUTH = {
    'JWT_ENCODE_HANDLER':
        'rest_framework_jwt.utils.jwt_encode_handler',

    'JWT_DECODE_HANDLER':
        'rest_framework_jwt.utils.jwt_decode_handler',

    'JWT_PAYLOAD_HANDLER':
        'rest_framework_jwt.utils.jwt_payload_handler',

    'JWT_PAYLOAD_GET_USER_ID_HANDLER':
        'rest_framework_jwt.utils.jwt_get_user_id_from_payload_handler',

    'JWT_RESPONSE_PAYLOAD_HANDLER':
        'rest_framework_jwt.utils.jwt_response_payload_handler',

    'JWT_SECRET_KEY': 'secretKey',
    'JWT_GET_USER_SECRET_KEY': None,
    'JWT_PUBLIC_KEY': None,
    'JWT_PRIVATE_KEY': None,
    'JWT_ALGORITHM': 'HS256',
    'JWT_VERIFY': True,
    'JWT_VERIFY_EXPIRATION': True,
    'JWT_LEEWAY': 0,
    # 'JWT_EXPIRATION_DELTA': datetime.timedelta(seconds=300),
    'JWT_AUDIENCE': None,
    'JWT_ISSUER': None,

    'JWT_ALLOW_REFRESH': False,
    # 'JWT_REFRESH_EXPIRATION_DELTA': datetime.timedelta(days=7)

    'JWT_AUTH_HEADER_PREFIX': 'Bearer',
    'JWT_AUTH_COOKIE': None,
}
```

Here `JWT_SECRET_KEY` is the most important part. You have to use same secret key for both frontend and back end. The frontend config can be found in path `isomorphic-servers/src/settings/index.js` under below config

```
const jwtConfig = {  
  fetchUrl: '/api/',  
  secretKey: 'secretKey',  
};
```