Obelix Group
obelixswe@gmail.com

Manuale utente

Versione	*v2_0_0*
Data creazione	2017-08-01
Redattori	Tomas Mali
	Emanuele Crespan
	Riccardo Saggese
Verificatori	Federica Schifano
	Nicolò Rigato
Approvazione	Silvio Meneguzzo
Stato	Approvato
Uso	Esterno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin

Sommario

Manuale utente per l'SDK Monolith e per le bolle di default e della Demo

Indice

1 Introduction

1.1 intro

The following document explain how to install and utilize Monolith SDK. It provide code snippet and example on how to create a custom bubble on Rocket.chat using the SDK.

1.2 How to install

To install Monolith SDK on Rocket.chat you need to follow the following steps:

1. git clone https://github.com/RocketChat/Rocket.Chat.git
2. cd Rocket.Chat
3. meteor npm start
   when finished close meteor
4. meteor add templating blaze-html-templates react-meteor-data maxharris9:classnames react-template-helper
5. meteor npm i react react-dom bluebird simpl-schema react-addons-pure-render-mixin money request request-promise --save
6. copy monolith-sdk folder inside the packages folder on Rocket.chat
7. meteor add monolith-sdk
8. meteor

If you want to use an existing Rocket.chat application start from step 4.

2 Use requirements

2.1 Requirements

In order to use Monolith the user needs to access the internet either using a laptop or desktop computer, or using a mobile cellphone or tablet. Other than that, the device which will be used to use Monolith must have a browser which supports JavaScript and has it enabled.

2.1.1 Desktop requirements

To access Monolith using a desktop or laptop computer, one of the following browsers is required:

- Google Chrome v59+
- Firefox v54+
- Internet Explorer v11+
- Safari v10+

2.1.2 Javascript

In order to enable JavaScript inside the different browsers' versions the next steps must be followed:

- **Google Chrome**
  - Next to the navigation bar, click on "Customize and control Google Chrome", and then "Settings";
  - Inside the "Settings" section, click on "Show advanced settings" at the bottom of the page;
  - Search for the "Privacy" section and click on "Content settings";
  - Under the "JavaScript" section inside the dialog popup, select "Allow all sites to run JavaScript (reccomended)";
  - Click "Done";
  - Close the "Section" tab;
  - **Note**: If you were on the application homepage while enabling JavaScript, please reload the page to make sure the changes take action.
- **Mozilla Firefox**
  - On the address bar, type `about:config` and hit "Enter";
  - Click on "I accept the risk!" if an alert message shows up;
  - Inside the search bar, search for `javascript.enabled`;
  - If the value below the "value" field is set to "false", right click on "false" and click "Toggle";
  - Close the "about:config" tab;
  - **Note**: If you were on the application homepage while enabling JavaScript, please reload the page to make sure the changes take action.
- **Safari**
  - Click on "Safari" on the menu bar, and then select "Preferences";
  - Inside the "Preferences" window, select on "Security";
  - Search for the "Web contents" section, and check "Enable JavaScript";
  - Close the "Preferences" menu;
  - **Note**: If you were on the application homepage while enabling JavaScript, please reload the page to make sure the changes take action.

2.2 Access to the application

To access Monolith, the only things required are the following:

1. Connect to the following Rocket.chat server from the device used to start the server:

   `http://localhost:3000`

2. Either:
   (a) Register a new account if you do not have one.
      i. Click on "Register a new account";
      ii. Fill all the required fields with your information;
      iii. Click on "Register a new account".
   (b) Login with you credentials.
      i. Insert your credentials inside the required fields
      ii. Click on "Login"

2.3 Error and bug reporting

If you encounter any error or bug during the normal use of the application, please report it to us by opening an issue on our GitHub repository at the following URL: `https://github.com/ObelixSWE/monolith-sdk/issues`.
We will work to continuously improve our application and fix all the bugs and error that will be reported.

3 Class Description

This section explains how to use the Monolith library classes.

3.1 SingleComponents

Singlecomponent classes represent the components that can be rendered.

CheckButton CheckButton represent a HTML <checkbox> tag.

```
<CheckButton
    id="HTML id"
    classes="CSS classes"
    getCheck={this.functionName}
    value="checkbox value"
/>
```

getCheck is a props that hold a function called when the checkbox onChange event is called and passes to the function a variable containing the state of the checkbox.

```
functionName(m){...}

m={id:"id", value:"value", check:[true/false]};
```

CheckBoxList CheckBoxList represent a group of CheckButton.
CheckBoxList needs to have an array passed like this:

```
let opt=[{id: 1, value: 'Hello World'},{id: 2, value: 'Installation'}];

<CheckBoxList
```

```
    classes="CSS classes"
    options={opt}
    getCheck={this.functionName}
/>
```

getCheck like the CheckButton getCheck.

```
functionName(m){...}

m={id:"id", value:"value", check:[true/false]};
```

ComboBox ComboBox represent a HTML <select> tag.

```
<ComboBox
    id="HTML id"
    classes="CSS classes"
    options={["a","b","c"]}
    getSelection={this.functionName}
/>
```

getSelection is a props that hold a function called when the select onChange event is called and passes to the function a variable containing the selected option.

```
functionName(m){...}

m="selected option";
```

Image Image represent a HTML <img> tag.

```
<Image
    id="HTML id"
    classes="CSS classes"
    src="img source location"
    alt="image description"
    width="image width"
    height="image height"
/>
```

ImageButton ImageButton represent a button with an image.

```
<ImageButton
    id="HTML id"
    src="img source location"
    alt="image description"
    width="image width"
    height="image height"
    handleClick={this.functionName}
/>
```

handleClick is a props that hold a function called when the ImageButton is clicked.

```
functionName(id){...}
id="id of the clicked button"
```

LineEdit LineEdit represent a HTML text <input> tag.

```
<LineEdit
    id="HTML id"
    classes="CSS classes"
    updateState={this.functionName}
    value="default value"
 />
```

updateState is a props that hold a function called when onChange event of the text input is called.

```
updateState(text,id){...}

text="text of the text input"
id="id of the text input"
```

LineEditComboBox LineEditComboBox represent a HTML text <input> and a HTML <select> tag.

```
<LineEditComboBox
    idle="LineEdit HTML id"
    idcb="ComboBox  HTML id"
    classesle="LineEdit CSS classes"
    classescb="ComboBox CSS classes"
    textUpdate={this.functionName1}
    options={["a","b","c"]}
    comboUpdate={this.functionName2}
/>
```

textUpdate is a props that hold a function passed to the LineEdit.
comboUpdate is a props that hold a function passed to le ComboBox.

```
functionName1(text,id){...}

text="text of the text input"
id="id of the text input"

functionName2(m){...}

m="selected option";
```

PushButton PushButton represent a HTML <button>.

```
<PushButton
    id="HTML id"
    classes="CSS classes"
    handleClick={this.functionName}
    buttonName="button name"
/>
```

handleClick is a props that hold a function called when the button is clicked.

```
functionName(id){...}

id="id of the button"
```

LineEditPushButton LineEditPushButton represent LineEdit and a PushButton.

```
<LineEditPushButton
    idle="LineEdit HTML id"
    idpb="PushButton HTML id"
    classesle="LineEdit CSS classes"
    classespb="PushButton CSS classes"
    getText={this.functionName}
    buttonName="button name"
/>
```

getText is a props that hold a function called when the PushButton is clicked and passes to the function a variable containing the text of the LineEdit.

```
functionName(text){...}

text="text of the LineEdit"
```

RadioButtonGroup RadioButtonGroup represent a group of HTML radio button <input>.

```
<RadioButtonGroup
    classes="CSS classes"
    options={["a","b","c"]}
    getValue={this.functionName}
/>
```

getValue is a props that hold a function called when a radio button is clicked and passes to the function a variable containing the selected radio information.

```
functionName(value){...}

value="value of the selected radio button"
```

TextAreaButton TextAreaButton represent a HTML <textarea> tag and a PushButton.

```
<TextAreaButton
    idta="textArea HTML id"
    classesta="textArea CSS classes"
    idpb="PushButton HTML id"
    classespb="PushButton CSS classes"
    getText={this.functionName}
    width="textarea width"
    height="textarea height"
    buttonName="button name"
/>
```

getText is a props that hold a function called when the PushButton is clicked and passes to the function a variable containing the text of the textarea.

```
functionName(text){...}

text="text of the textarea"
```

TextAreaComboBox TextAreaComboBox represent a HTML <textarea> tag and a ComboBox.

```
<TextAreaComboBox
    idtx="textArea HTML id"
    classestx="textArea CSS classes"
    idcb="combobox HTML id"
    classescb="combobox CSS classes"
    width="textarea width"
    height="textarea height"
    textUpdate={this.functionName1}
    options={["a","b","c"]}
    comboUpdate={this.functionName2}
/>
```

textUpdate is a props that hold a function called when onChange event of the textarea is called.
comboUpdate is a props that hold a function called when the select onChange event is called and passes to the function a variable containing the selected option.

```
functionName1(text){...}

text="text of the textarea"

functionName2(m){...}

m="selected option";
```

3.2 Layout

Layout are classes that represent are containers that place the elements conteined in a certain way.

VerticalLayout VerticalLayout represent a container that place the elements contained in a vertically.

```
<VerticalLayout hide={"visibility state(true or false)"}>
    <Children/>
    <Children/>
    .
    .
    .
</VerticalLayout>
```

HorizontalLayout HorizontalLayout represent a container that place the elements contained in a horizontally.
The maximum number of element that can be displayed horizontally is 12.

```
<HorizontalLayout hide={"visibility state(true or false)"}>
    <Children/>
    <Children/>
    .
    .
    .
</HorizontalLayout>
```