

实验报告

—— 日历的开发

陈广庆

2016年10月28日

目录

1 前言	3
2 设计思路	4
3 设计过程	5
3.1 index.html	5
3.2 main.js	5
3.3 React&Flux	6
3.3.1 View	6
3.3.1.1 app.js	6
3.3.1.2 main-panel.js	7
3.3.1.3 lunar-panel.js	9
3.3.1.4 date-panel.js	12
3.3.1.5 date-choose-panel.js	13
3.3.1.6 date-controller.js	17
3.3.2 Dispatcher	21
3.3.2.1 dispatcher.js	21
3.3.3 Action	22
3.3.3.1 calendar-actions.js	22
3.3.3.2 config-actions.js	23
3.3.3.3 dialog-actions.js	23
3.3.4 Store	24
3.3.4.1 calendar-database.js	24
3.3.4.2 calendar-store.js	26

3.3.4.3 dialog-store.js	29
3.3.4.4 config-store.js	30
4 测试情况	31
4.1 启动界面	31
4.2 工具栏界面	31
4.3 记事本界面	32
4.4 设置菜单	32
4.5 关于菜单	33
4.6 占用资源	33
5 优点与改进	33
5.1 优点	33
5.2 缺点	33
6 参考文献	34
6.1 Q&A	34
6.2 Electron	34
6.3 Menubar	34
6.4 React	34
6.5 Flux	34
6.6 browserify	34
6.7 gulp	34
6.8 Material Design	34
6.9 Material-ui	34

1 前言

这次实验作业，我选择了做日历

通过Electron，只用一份源代码，就能同时生成Windows、Linux、maxOS三个系统的应用，经过简单的修改，还能生成网页版(goushi.me/yun/lunar-calendar/lunar-calendar.html)

同时实现了以下功能：

- 按周、月的方式显示日历
- 显示农历和节假日
- 记事本功能，可记录每日的活动
- 点击跳转到上一年/月
- 点击跳转到下一年/月
- 通过输入，跳转到指定年/月份
- 返回今天
- 退出日历
- 总在最上

开发备注

```
1 # 安装依赖包（在此之前，需要先安装Node V6）
2 npm install
3 # 打包并测试
4 npm start
5 # 生成Windows 64位的应用
6 npm run build-win-64
7 # 生成Windows 32位的应用
8 npm run build-win-32
9 # 生成Linux 64位的应用
10 npm run build-linux-64
11 # 生成Linux 32位的应用
12 npm run build-linux-32
13 # 生成Mac 64位的应用
14 npm run build-darwin-64
15 # 生成Mac 32位的应用
16 npm run build-darwin-32
```

2 设计思路

Electron可以看做一个由JavaScript控制的Chromium浏览器，可以把在其之上开发的网站封装成桌面应用。

Electron可以使用绝大多数的npm包，里面有个Menubar可以帮你处理好工具栏应用的系统底层细节。

所以接下来只需要写一个农历网页。

网页主要由React&Flux实现

前端界面元素由'material-ui'提供

后端农历数据由'lunar-calendar-zh'提供

在Electron里，需要把React写的js文件转码，因此还用了browserify把它打包成SPA(Single Page Application)

手动打包再构建的话，不利于开发，因此还用了gulp实现一键自动化打包、构建

3 设计过程

3.1 index.html

首先用个简单的HTML文件写个网页框架，该网页只有一个ID为'app'的div元素，由React渲染

```
index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <link rel="stylesheet" type="text/css" href=
6     fonts.css">
7   <title>Lunar-Calendar</title>
8 </head>
9 <body style="margin: 0">
10 <div id="app"></div>
11 <!-- reason: https://github.com/electron/electron/
12   issues/1611 -->
13 <script>var electronRequire = require;</script>
14 <script src="build/app.js"></script>
15 </html>
```

3.2 main.js

然后是Electron的main process，主要是创建个Menubar的对象，并在其中加载index.html

同时绑定了“退出”，“重启”，“隐藏”，“显示”等事件

```
main.js
1 var mb = menubar({
2   alwaysOnTop: conf.get('alwaysOnTop') || false,
3   preloadWindow: true,
4   dir: __dirname,
5   icon: __dirname + '/icon.png',
6   tooltip: '新世界的大门',
7   width: 800,
8   height: 436
9});
```

```

10    mb.on('ready', function () {
11      console.log('app is ready');
12
13
14      ipc.on('app-quit', function () {
15        mb.app.quit();
16      });
17
18      ipc.on('app-restart', function () {
19        mb.app.relaunch();
20        mb.app.quit();
21      });
22
23      mb.on('hide', function () {
24        mb.window.webContents.send('app-hide')
25      });
26
27      mb.on('show', function () {
28        mb.window.webContents.send('app-refresh')
29      });
30    });

```

3.3 React&Flux

3.3.1 View

3.3.1.1 app.js

react部分的出口，主要是用MainPanel组件渲染ID为'app'的div元素

```

react/app.js
1  var React = require('react');
2  var ReactDOM = require('react-dom');
3  var MainPanel = require('./components/main-panel');
4  import InjectTapEventPlugin from
  'react-tap-event-plugin';
5
6 // Needed for onTouchTap
7 // http://stackoverflow.com/a/34015469/988941
8 InjectTapEventPlugin();
9
10 ReactDOM.render(<MainPanel/>,
  document.getElementById('app'));

```

3.3.1.2 main-panel.js

MainPanel组件由LunarPanel组件与DatePanel组件组成，通过`<GridList>`组件布局

同时注意要监听CalendarStore的事件'changeDay'，CalendarStore的属性'activeDay'一改变，MainPanel就要重新渲染组件

react/components/main-panel.js

```
1 var React = require('react');
2 var ipc = electronRequire('electron').ipcRenderer;
3 var DatePanel = require('./date-panel');
4 var LunarPanel = require('./lunar-panel');
5 var CalendarStore = require('../stores/
    calendar-store');
6 var CalendarAction = require('../actions/
    calendar-actions');
7 var COLOR = require('../constants/calendar-color');

8
9 import MuiThemeProvider from 'material-ui/styles/
    MuiThemeProvider';
10 import {GridList, GridTile} from 'material-ui/
    GridList';

11
12 var MainPanel = React.createClass({
13     getInitialState: function () {
14         return {
15             today: CalendarStore.getToday(),
16             activeDay: CalendarStore.getActiveDay(),
17             activeMonth:
18                 CalendarStore.getActiveMonth()
19         };
20     },
21     componentDidMount: function () {
22         CalendarStore.addChangeDayListener(
23             this._onChange);
24         ipc.on('app-refresh', function () {
25             CalendarAction.refresh();
26         });
27     },
28     _onChange: function () {
```

```
29         this.setState({
30             today: CalendarStore.getToday(),
31             activeDay: CalendarStore.getActiveDay(),
32             activeMonth:
33                 CalendarStore.getActiveMonth()
34         );
35     },
36
37     render: function () {
38         var styles = {
39             mainPanel: {
40                 border: '2px solid ' +
41                     COLOR.mainPanel.border
42             };
43
44             return (
45                 <MuiThemeProvider>
46                 <GridList cols={10} cellHeight={432}
47                     padding={0} style={styles.mainPanel}>
48                     <GridTile cols={4}>
49                         <LunarPanel activeDay={
50                             this.state.activeDay}>
51                         </GridTile>
52                         <GridTile cols={6}>
53                             <DatePanel
54                                 today={this.state.today}
55                                 activeDay={this.state.activeDay}
56                                 activeMonth={this.state.activeMonth}>
57                         </GridTile>
58                     </GridList>
59                 </MuiThemeProvider>
60             );
61     }
62 };
```

module.exports = MainPanel;

3.3.1.3 lunar-panel.js

LunarPanel即左边的显示详细信息的面板

这里主要是前端的设计，通过`<Transitive/>`组件实现动画效果

react/components/lunar-panel.js

```
29         background:
30             COLOR.ganZhiPanel.background,
31             color: COLOR.ganZhiPanel.color,
32             lineHeight: '60px',
33             textAlign: 'center'
34     },
35     ...
36     ...
37     ...
38     h1Panel: {
39         background: COLOR.h1Panel.background
40             ,
41             color: COLOR.h1Panel.color,
42             textOverflow: 'ellipsis',
43             overflow: 'hidden',
44             whiteSpace: 'nowrap'
45     }
46 };
47
48
49     return (
50         <GridList cols={10} cellHeight={31}
51             padding={0}>
52             <GridTile cols={10} rows={2} style={
53                 styles.ganZhiPanel}>
54                 <Transitive>{activeDay.GanZhiYear +
55                     '年' }</Transitive>
56                 <Transitive>{' ' + +
57                     activeDay.GanZhiMonth + '月' }</
58                     Transitive>
59                 <Transitive>{' ' + +
60                     activeDay.GanZhiDay + '日' }</
61                     Transitive>
62             </GridTile>
63             <GridTile cols={10} rows={1} style={
64                 styles.dayPanel}>
65             </GridTile>
66             <GridTile cols={3} rows={6} style={
67                 styles.yearPanel}>
68                 <br/><Transitive>{activeDay.year
69                     }</Transitive><br/>
70                 <Transitive>{activeDay.month + +
71                     '月' }</Transitive>
```

```
58         </GridTile>
59         <GridTile cols={4} rows={6} style={
60             styles.dayPanel}>
61             <Paper zDepth={2} style={
62                 styles.dayDetail}>
63                 <Transitive>{activeDay.day}<
64                     /Transitive>
65             </Paper>
66         </GridTile>
67         <GridTile cols={3} rows={6} style={
68             styles.festivalPanel}>
69             <br/>{festivalElements}
70         </GridTile>
71         <GridTile cols={10} rows={1} style={
72             styles.monthPanel}>
73             <GridTile cols={10} rows={2} style={
74                 styles.monthPanel}>
75                 <Paper zDepth={1} style={
76                     styles.monthDetail}>
77                     <Transitive>{
78                         activeDay.lunarMonthName
79                         + ' ' }</Transitive>
80                     <Transitive>{
81                         activeDay.lunarDayName}<
82                         /Transitive>
83                     </Paper>
84                 </GridTile>
85             </GridList>
86         )
87     }
88 );
89 }
90
91 module.exports = LunarPanel;
```

3.3.1.4 date-panel.js

DatePanel组件即右边的面板，由DateController组件与DateChoosePanel组件组成

```
react/components/date-panel.js
1 var React = require('react');
2 var DateController = require('./date-controller');
3 var DateChoosePanel = require('./date-choose-panel')
4 ;
5 var DatePanel = React.createClass({
6   render: function () {
7     return (
8       <div>
9         <DateController activeDay={
10           this.props.activeDay} />
11         <DateChoosePanel
12           today={this.props.today}
13           activeDay={this.props.activeDay}
14           activeMonth={this.props.activeMonth}
15         />
16       </div>
17     )
18   }
19 });
20 module.exports = DatePanel;
```

3.3.1.5 date-choose-panel.js

DateChoosePanel组件即右下方的选择日子的面板，通过`<Table/>`组件实现表格布局

这里每格都是一个按钮，按下后，会触发'changeDay'这个action

```
react/components/date-choose-panel.js
1 var React = require('react');
2 var CalendarAction = require('../actions/
3   calendar-actions');
4 var COLOR = require('../constants/calendar-color');
5
6 import {Table, TableBody, TableHeader,
7   TableHeaderColumn, TableRow, TableRowColumn}
8   from 'material-ui/Table';
9 import {FlatButton} from 'material-ui'
10
11 var DateChoosePanel = React.createClass({
12   render: function () {
13     var today = this.props.today;
14     var activeDay = this.props.activeDay;
15     var activeMonth = this.props.activeMonth;
16
17     var table = [];
18     var tableBody = [];
19     var TableRow = [];
20
21     var styles = {
22       tableHeader: {
23         background:
24           COLOR.tableHeader.background
25     },
26     tableBody: {
27       background:
28           COLOR.tableBody.background
29     },
30     orangeLabel: {
31       lineHeight: 0,
32       fontSize: 10,
33       color: COLOR.orangeLabel.color
34     },
35     greyLabel: {
```

```
31         lineHeight: 0,
32         fontSize: 10,
33         color: COLOR.greyLabel.color
34     },
35     tableButton: {
36         minWidth: 66,
37         height: 50
38     }
39 };
40
41 table.push(
42     <TableHeader key='tableHeader'
43         adjustForCheckbox={false}
44         displaySelectAll={false}
45         enableSelectAll={false}
46         style={styles.tableHeader}
47     >
48     <TableRow>
49         <TableHeaderColumn>日</
50             TableHeaderColumn>
51         <TableHeaderColumn>一</
52             TableHeaderColumn>
53         <TableHeaderColumn>二</
54             TableHeaderColumn>
55         <TableHeaderColumn>三</
56             TableHeaderColumn>
57         <TableHeaderColumn>四</
58             TableHeaderColumn>
59         <TableHeaderColumn>五</
60             TableHeaderColumn>
61         <TableHeaderColumn>六</
62             TableHeaderColumn>
63     </TableRow>
64     </TableHeader>
65 );
66
67 for (var i = 0; i < activeMonth.length; i++)
68 {
69     var date = activeMonth[i];
70
71     var label = '';
72     if (date.lunarFestival != undefined)
```

```
65         label = <div style={  
66             styles.orangeLabel}>{  
67                 date.lunarFestival}</div>;  
68     else if (date.term != undefined)  
69         label = <div style={  
70             styles.orangeLabel}>{date.term}  
71                 </div>;  
72     else if (date.lunarDay == 1)  
73         label = <div style={  
74             styles.orangeLabel}>{  
75                 date.lunarMonthName}</div>;  
76     else  
77         label = <div style={styles.greyLabel  
78             }>{date.lunarDayName}</div>;  
79  
80         var styles_tableCell = {  
81             background:  
82                 COLOR.tableCell.background.normal  
83                 ,  
84                 paddingLeft: 0  
85             };  
86  
87         ...  
88         ...  
89         ...  
90         ...  
91  
92         tableRow.push(  
93             <TableRowColumn style={  
94                 styles_tableCell} key={'  
95                     TableRowColumn' + i}>  
96                 <FlatButton  
97                     label={date.day}  
98                     labelStyle={  
99                         styles_numberLabel}  
100                     labelPosition={'before'}  
101                     rippleColor={  
102                         COLOR.numberLabel.color.ripple  
103                         }  
104                     onTouchTap={  
105                         this._onClick.bind(this,  
106                         date)}  
107                     style={styles.tableButton}
```

```
91      >
92          {label}
93      </FlatButton>
94  </TableRowColumn>
95 );
96
97 if ((i+1) % 7 == 0) {
98     tableBody.push(
99         <TableRow key={'tableRow' + i}>
100            {tableRow}
101        </TableRow>
102    );
103    tableRow = [];
104 }
105
106
107 table.push(
108     <TableBody
109       displayRowCheckbox={false}
110       style={styles.tableBody}
111       key='tableBody'
112     >
113       {tableBody}
114     </TableBody>
115   );
116
117 return (
118   <Table selectable={false}>
119     {table}
120   </Table>
121 )
122 },
123
124 _onClick: function (date) {
125   CalendarAction.changeDay(date);
126 }
127 );
128
129 module.exports = DateChoosePanel;
```

3.3.1.6 date-controller.js

DateController组件即右上方的toolbar
只有这里需要从DialogStore与ConfigStore中获取数据（对话框、设置选项的状态），因此也要监听DialogStore的'closeAllDialog'事件
每一个按钮都会触发相应的action

```
react/components/date-controller.js
1  var DateController = React.createClass({
2      getInitialState: function () {
3          return {
4              noteDialog: DialogStore.getNote(),
5              optionDialog: DialogStore.getOption(),
6              aboutDialog: DialogStore.getAbout(),
7              registerDialog: DialogStore.getRegister
8                  (),
9              note: this.props.activeDay.note,
10             alwaysOnTop: ConfigStore.getAlwaysOnTop
11                 ()
12         };
13     },
14
15     componentDidMount: function () {
16         DialogStore.addCloseDialogListener(
17             this.closeAllDialog);
18         ipc.on('app-hide', function () {
19             DialogAction.closeDialog();
20         });
21
22         openDialog: function (name) {
23             if (name == 'note')
24                 this.setState({
25                     note: this.props.activeDay.note,
26                     [name + 'Dialog']: true
27                 });
28             else if (name == 'option')
29                 this.setState({
30                     alwaysOnTop:
31                         ConfigStore.getAlwaysOnTop(),
32                     [name + 'Dialog']: true
33                 });
34         };
35     }
36
37     closeAllDialog: function () {
38         DialogStore.removeCloseDialogListener(this.closeAllDialog);
39         ipc.removeListener('app-hide', this.closeAllDialog);
40     }
41
42     render() {
43         return (
44             <div>
45                 <button>Note</button>
46                 <button>Option</button>
47                 <button>About</button>
48                 <button>Register</button>
49             </div>
50         );
51     }
52
53     closeAllDialog() {
54         DialogStore.closeAllDialog();
55     }
56
57     handleHide() {
58         ipc.send('app-hide');
59     }
60
61     handleShow() {
62         ipc.send('app-show');
63     }
64
65     handleAlwaysOnTop() {
66         ConfigStore.toggleAlwaysOnTop();
67     }
68
69     handleAbout() {
70         DialogAction.openAbout();
71     }
72
73     handleOption() {
74         DialogAction.openOption();
75     }
76
77     handleNote() {
78         DialogAction.openNote();
79     }
80
81     handleRegister() {
82         DialogAction.openRegister();
83     }
84
85     handleCloseAllDialog() {
86         DialogStore.closeAllDialog();
87     }
88
89     handleCloseDialog(name) {
90         DialogStore.closeDialog(name);
91     }
92
93     handleOpenDialog(name) {
94         DialogStore.openDialog(name);
95     }
96
97     handleToggleAlwaysOnTop() {
98         ConfigStore.toggleAlwaysOnTop();
99     }
100
101     handleToggleCloseAllDialog() {
102         DialogStore.toggleCloseAllDialog();
103     }
104
105     handleToggleCloseDialog(name) {
106         DialogStore.toggleCloseDialog(name);
107     }
108
109     handleToggleOpenDialog(name) {
110         DialogStore.toggleOpenDialog(name);
111     }
112
113     handleToggleOption() {
114         DialogAction.toggleOption();
115     }
116
117     handleToggleNote() {
118         DialogAction.toggleNote();
119     }
120
121     handleToggleAbout() {
122         DialogAction.toggleAbout();
123     }
124
125     handleToggleRegister() {
126         DialogAction.toggleRegister();
127     }
128
129     handleToggleCloseAllDialog() {
130         DialogStore.toggleCloseAllDialog();
131     }
132
133     handleToggleCloseDialog(name) {
134         DialogStore.toggleCloseDialog(name);
135     }
136
137     handleToggleOpenDialog(name) {
138         DialogStore.toggleOpenDialog(name);
139     }
140
141     handleToggleOption() {
142         DialogAction.toggleOption();
143     }
144
145     handleToggleNote() {
146         DialogAction.toggleNote();
147     }
148
149     handleToggleAbout() {
150         DialogAction.toggleAbout();
151     }
152
153     handleToggleRegister() {
154         DialogAction.toggleRegister();
155     }
156
157     handleToggleCloseAllDialog() {
158         DialogStore.toggleCloseAllDialog();
159     }
160
161     handleToggleCloseDialog(name) {
162         DialogStore.toggleCloseDialog(name);
163     }
164
165     handleToggleOpenDialog(name) {
166         DialogStore.toggleOpenDialog(name);
167     }
168
169     handleToggleOption() {
170         DialogAction.toggleOption();
171     }
172
173     handleToggleNote() {
174         DialogAction.toggleNote();
175     }
176
177     handleToggleAbout() {
178         DialogAction.toggleAbout();
179     }
180
181     handleToggleRegister() {
182         DialogAction.toggleRegister();
183     }
184
185     handleToggleCloseAllDialog() {
186         DialogStore.toggleCloseAllDialog();
187     }
188
189     handleToggleCloseDialog(name) {
190         DialogStore.toggleCloseDialog(name);
191     }
192
193     handleToggleOpenDialog(name) {
194         DialogStore.toggleOpenDialog(name);
195     }
196
197     handleToggleOption() {
198         DialogAction.toggleOption();
199     }
200
201     handleToggleNote() {
202         DialogAction.toggleNote();
203     }
204
205     handleToggleAbout() {
206         DialogAction.toggleAbout();
207     }
208
209     handleToggleRegister() {
210         DialogAction.toggleRegister();
211     }
212
213     handleToggleCloseAllDialog() {
214         DialogStore.toggleCloseAllDialog();
215     }
216
217     handleToggleCloseDialog(name) {
218         DialogStore.toggleCloseDialog(name);
219     }
220
221     handleToggleOpenDialog(name) {
222         DialogStore.toggleOpenDialog(name);
223     }
224
225     handleToggleOption() {
226         DialogAction.toggleOption();
227     }
228
229     handleToggleNote() {
230         DialogAction.toggleNote();
231     }
232
233     handleToggleAbout() {
234         DialogAction.toggleAbout();
235     }
236
237     handleToggleRegister() {
238         DialogAction.toggleRegister();
239     }
240
241     handleToggleCloseAllDialog() {
242         DialogStore.toggleCloseAllDialog();
243     }
244
245     handleToggleCloseDialog(name) {
246         DialogStore.toggleCloseDialog(name);
247     }
248
249     handleToggleOpenDialog(name) {
250         DialogStore.toggleOpenDialog(name);
251     }
252
253     handleToggleOption() {
254         DialogAction.toggleOption();
255     }
256
257     handleToggleNote() {
258         DialogAction.toggleNote();
259     }
260
261     handleToggleAbout() {
262         DialogAction.toggleAbout();
263     }
264
265     handleToggleRegister() {
266         DialogAction.toggleRegister();
267     }
268
269     handleToggleCloseAllDialog() {
270         DialogStore.toggleCloseAllDialog();
271     }
272
273     handleToggleCloseDialog(name) {
274         DialogStore.toggleCloseDialog(name);
275     }
276
277     handleToggleOpenDialog(name) {
278         DialogStore.toggleOpenDialog(name);
279     }
280
281     handleToggleOption() {
282         DialogAction.toggleOption();
283     }
284
285     handleToggleNote() {
286         DialogAction.toggleNote();
287     }
288
289     handleToggleAbout() {
290         DialogAction.toggleAbout();
291     }
292
293     handleToggleRegister() {
294         DialogAction.toggleRegister();
295     }
296
297     handleToggleCloseAllDialog() {
298         DialogStore.toggleCloseAllDialog();
299     }
300
301     handleToggleCloseDialog(name) {
302         DialogStore.toggleCloseDialog(name);
303     }
304
305     handleToggleOpenDialog(name) {
306         DialogStore.toggleOpenDialog(name);
307     }
308
309     handleToggleOption() {
310         DialogAction.toggleOption();
311     }
312
313     handleToggleNote() {
314         DialogAction.toggleNote();
315     }
316
317     handleToggleAbout() {
318         DialogAction.toggleAbout();
319     }
320
321     handleToggleRegister() {
322         DialogAction.toggleRegister();
323     }
324
325     handleToggleCloseAllDialog() {
326         DialogStore.toggleCloseAllDialog();
327     }
328
329     handleToggleCloseDialog(name) {
330         DialogStore.toggleCloseDialog(name);
331     }
332
333     handleToggleOpenDialog(name) {
334         DialogStore.toggleOpenDialog(name);
335     }
336
337     handleToggleOption() {
338         DialogAction.toggleOption();
339     }
340
341     handleToggleNote() {
342         DialogAction.toggleNote();
343     }
344
345     handleToggleAbout() {
346         DialogAction.toggleAbout();
347     }
348
349     handleToggleRegister() {
350         DialogAction.toggleRegister();
351     }
352
353     handleToggleCloseAllDialog() {
354         DialogStore.toggleCloseAllDialog();
355     }
356
357     handleToggleCloseDialog(name) {
358         DialogStore.toggleCloseDialog(name);
359     }
360
361     handleToggleOpenDialog(name) {
362         DialogStore.toggleOpenDialog(name);
363     }
364
365     handleToggleOption() {
366         DialogAction.toggleOption();
367     }
368
369     handleToggleNote() {
370         DialogAction.toggleNote();
371     }
372
373     handleToggleAbout() {
374         DialogAction.toggleAbout();
375     }
376
377     handleToggleRegister() {
378         DialogAction.toggleRegister();
379     }
380
381     handleToggleCloseAllDialog() {
382         DialogStore.toggleCloseAllDialog();
383     }
384
385     handleToggleCloseDialog(name) {
386         DialogStore.toggleCloseDialog(name);
387     }
388
389     handleToggleOpenDialog(name) {
390         DialogStore.toggleOpenDialog(name);
391     }
392
393     handleToggleOption() {
394         DialogAction.toggleOption();
395     }
396
397     handleToggleNote() {
398         DialogAction.toggleNote();
399     }
400
401     handleToggleAbout() {
402         DialogAction.toggleAbout();
403     }
404
405     handleToggleRegister() {
406         DialogAction.toggleRegister();
407     }
408
409     handleToggleCloseAllDialog() {
410         DialogStore.toggleCloseAllDialog();
411     }
412
413     handleToggleCloseDialog(name) {
414         DialogStore.toggleCloseDialog(name);
415     }
416
417     handleToggleOpenDialog(name) {
418         DialogStore.toggleOpenDialog(name);
419     }
420
421     handleToggleOption() {
422         DialogAction.toggleOption();
423     }
424
425     handleToggleNote() {
426         DialogAction.toggleNote();
427     }
428
429     handleToggleAbout() {
430         DialogAction.toggleAbout();
431     }
432
433     handleToggleRegister() {
434         DialogAction.toggleRegister();
435     }
436
437     handleToggleCloseAllDialog() {
438         DialogStore.toggleCloseAllDialog();
439     }
440
441     handleToggleCloseDialog(name) {
442         DialogStore.toggleCloseDialog(name);
443     }
444
445     handleToggleOpenDialog(name) {
446         DialogStore.toggleOpenDialog(name);
447     }
448
449     handleToggleOption() {
450         DialogAction.toggleOption();
451     }
452
453     handleToggleNote() {
454         DialogAction.toggleNote();
455     }
456
457     handleToggleAbout() {
458         DialogAction.toggleAbout();
459     }
460
461     handleToggleRegister() {
462         DialogAction.toggleRegister();
463     }
464
465     handleToggleCloseAllDialog() {
466         DialogStore.toggleCloseAllDialog();
467     }
468
469     handleToggleCloseDialog(name) {
470         DialogStore.toggleCloseDialog(name);
471     }
472
473     handleToggleOpenDialog(name) {
474         DialogStore.toggleOpenDialog(name);
475     }
476
477     handleToggleOption() {
478         DialogAction.toggleOption();
479     }
480
481     handleToggleNote() {
482         DialogAction.toggleNote();
483     }
484
485     handleToggleAbout() {
486         DialogAction.toggleAbout();
487     }
488
489     handleToggleRegister() {
490         DialogAction.toggleRegister();
491     }
492
493     handleToggleCloseAllDialog() {
494         DialogStore.toggleCloseAllDialog();
495     }
496
497     handleToggleCloseDialog(name) {
498         DialogStore.toggleCloseDialog(name);
499     }
500
501     handleToggleOpenDialog(name) {
502         DialogStore.toggleOpenDialog(name);
503     }
504
505     handleToggleOption() {
506         DialogAction.toggleOption();
507     }
508
509     handleToggleNote() {
510         DialogAction.toggleNote();
511     }
512
513     handleToggleAbout() {
514         DialogAction.toggleAbout();
515     }
516
517     handleToggleRegister() {
518         DialogAction.toggleRegister();
519     }
520
521     handleToggleCloseAllDialog() {
522         DialogStore.toggleCloseAllDialog();
523     }
524
525     handleToggleCloseDialog(name) {
526         DialogStore.toggleCloseDialog(name);
527     }
528
529     handleToggleOpenDialog(name) {
530         DialogStore.toggleOpenDialog(name);
531     }
532
533     handleToggleOption() {
534         DialogAction.toggleOption();
535     }
536
537     handleToggleNote() {
538         DialogAction.toggleNote();
539     }
540
541     handleToggleAbout() {
542         DialogAction.toggleAbout();
543     }
544
545     handleToggleRegister() {
546         DialogAction.toggleRegister();
547     }
548
549     handleToggleCloseAllDialog() {
550         DialogStore.toggleCloseAllDialog();
551     }
552
553     handleToggleCloseDialog(name) {
554         DialogStore.toggleCloseDialog(name);
555     }
556
557     handleToggleOpenDialog(name) {
558         DialogStore.toggleOpenDialog(name);
559     }
560
561     handleToggleOption() {
562         DialogAction.toggleOption();
563     }
564
565     handleToggleNote() {
566         DialogAction.toggleNote();
567     }
568
569     handleToggleAbout() {
570         DialogAction.toggleAbout();
571     }
572
573     handleToggleRegister() {
574         DialogAction.toggleRegister();
575     }
576
577     handleToggleCloseAllDialog() {
578         DialogStore.toggleCloseAllDialog();
579     }
580
581     handleToggleCloseDialog(name) {
582         DialogStore.toggleCloseDialog(name);
583     }
584
585     handleToggleOpenDialog(name) {
586         DialogStore.toggleOpenDialog(name);
587     }
588
589     handleToggleOption() {
590         DialogAction.toggleOption();
591     }
592
593     handleToggleNote() {
594         DialogAction.toggleNote();
595     }
596
597     handleToggleAbout() {
598         DialogAction.toggleAbout();
599     }
600
601     handleToggleRegister() {
602         DialogAction.toggleRegister();
603     }
604
605     handleToggleCloseAllDialog() {
606         DialogStore.toggleCloseAllDialog();
607     }
608
609     handleToggleCloseDialog(name) {
610         DialogStore.toggleCloseDialog(name);
611     }
612
613     handleToggleOpenDialog(name) {
614         DialogStore.toggleOpenDialog(name);
615     }
616
617     handleToggleOption() {
618         DialogAction.toggleOption();
619     }
620
621     handleToggleNote() {
622         DialogAction.toggleNote();
623     }
624
625     handleToggleAbout() {
626         DialogAction.toggleAbout();
627     }
628
629     handleToggleRegister() {
630         DialogAction.toggleRegister();
631     }
632
633     handleToggleCloseAllDialog() {
634         DialogStore.toggleCloseAllDialog();
635     }
636
637     handleToggleCloseDialog(name) {
638         DialogStore.toggleCloseDialog(name);
639     }
640
641     handleToggleOpenDialog(name) {
642         DialogStore.toggleOpenDialog(name);
643     }
644
645     handleToggleOption() {
646         DialogAction.toggleOption();
647     }
648
649     handleToggleNote() {
650         DialogAction.toggleNote();
651     }
652
653     handleToggleAbout() {
654         DialogAction.toggleAbout();
655     }
656
657     handleToggleRegister() {
658         DialogAction.toggleRegister();
659     }
660
661     handleToggleCloseAllDialog() {
662         DialogStore.toggleCloseAllDialog();
663     }
664
665     handleToggleCloseDialog(name) {
666         DialogStore.toggleCloseDialog(name);
667     }
668
669     handleToggleOpenDialog(name) {
670         DialogStore.toggleOpenDialog(name);
671     }
672
673     handleToggleOption() {
674         DialogAction.toggleOption();
675     }
676
677     handleToggleNote() {
678         DialogAction.toggleNote();
679     }
680
681     handleToggleAbout() {
682         DialogAction.toggleAbout();
683     }
684
685     handleToggleRegister() {
686         DialogAction.toggleRegister();
687     }
688
689     handleToggleCloseAllDialog() {
690         DialogStore.toggleCloseAllDialog();
691     }
692
693     handleToggleCloseDialog(name) {
694         DialogStore.toggleCloseDialog(name);
695     }
696
697     handleToggleOpenDialog(name) {
698         DialogStore.toggleOpenDialog(name);
699     }
700
701     handleToggleOption() {
702         DialogAction.toggleOption();
703     }
704
705     handleToggleNote() {
706         DialogAction.toggleNote();
707     }
708
709     handleToggleAbout() {
710         DialogAction.toggleAbout();
711     }
712
713     handleToggleRegister() {
714         DialogAction.toggleRegister();
715     }
716
717     handleToggleCloseAllDialog() {
718         DialogStore.toggleCloseAllDialog();
719     }
720
721     handleToggleCloseDialog(name) {
722         DialogStore.toggleCloseDialog(name);
723     }
724
725     handleToggleOpenDialog(name) {
726         DialogStore.toggleOpenDialog(name);
727     }
728
729     handleToggleOption() {
730         DialogAction.toggleOption();
731     }
732
733     handleToggleNote() {
734         DialogAction.toggleNote();
735     }
736
737     handleToggleAbout() {
738         DialogAction.toggleAbout();
739     }
740
741     handleToggleRegister() {
742         DialogAction.toggleRegister();
743     }
744
745     handleToggleCloseAllDialog() {
746         DialogStore.toggleCloseAllDialog();
747     }
748
749     handleToggleCloseDialog(name) {
750         DialogStore.toggleCloseDialog(name);
751     }
752
753     handleToggleOpenDialog(name) {
754         DialogStore.toggleOpenDialog(name);
755     }
756
757     handleToggleOption() {
758         DialogAction.toggleOption();
759     }
760
761     handleToggleNote() {
762         DialogAction.toggleNote();
763     }
764
765     handleToggleAbout() {
766         DialogAction.toggleAbout();
767     }
768
769     handleToggleRegister() {
770         DialogAction.toggleRegister();
771     }
772
773     handleToggleCloseAllDialog() {
774         DialogStore.toggleCloseAllDialog();
775     }
776
777     handleToggleCloseDialog(name) {
778         DialogStore.toggleCloseDialog(name);
779     }
780
781     handleToggleOpenDialog(name) {
782         DialogStore.toggleOpenDialog(name);
783     }
784
785     handleToggleOption() {
786         DialogAction.toggleOption();
787     }
788
789     handleToggleNote() {
790         DialogAction.toggleNote();
791     }
792
793     handleToggleAbout() {
794         DialogAction.toggleAbout();
795     }
796
797     handleToggleRegister() {
798         DialogAction.toggleRegister();
799     }
800
801     handleToggleCloseAllDialog() {
802         DialogStore.toggleCloseAllDialog();
803     }
804
805     handleToggleCloseDialog(name) {
806         DialogStore.toggleCloseDialog(name);
807     }
808
809     handleToggleOpenDialog(name) {
810         DialogStore.toggleOpenDialog(name);
811     }
812
813     handleToggleOption() {
814         DialogAction.toggleOption();
815     }
816
817     handleToggleNote() {
818         DialogAction.toggleNote();
819     }
820
821     handleToggleAbout() {
822         DialogAction.toggleAbout();
823     }
824
825     handleToggleRegister() {
826         DialogAction.toggleRegister();
827     }
828
829     handleToggleCloseAllDialog() {
830         DialogStore.toggleCloseAllDialog();
831     }
832
833     handleToggleCloseDialog(name) {
834         DialogStore.toggleCloseDialog(name);
835     }
836
837     handleToggleOpenDialog(name) {
838         DialogStore.toggleOpenDialog(name);
839     }
840
841     handleToggleOption() {
842         DialogAction.toggleOption();
843     }
844
845     handleToggleNote() {
846         DialogAction.toggleNote();
847     }
848
849     handleToggleAbout() {
850         DialogAction.toggleAbout();
851     }
852
853     handleToggleRegister() {
854         DialogAction.toggleRegister();
855     }
856
857     handleToggleCloseAllDialog() {
858         DialogStore.toggleCloseAllDialog();
859     }
860
861     handleToggleCloseDialog(name) {
862         DialogStore.toggleCloseDialog(name);
863     }
864
865     handleToggleOpenDialog(name) {
866         DialogStore.toggleOpenDialog(name);
867     }
868
869     handleToggleOption() {
870         DialogAction.toggleOption();
871     }
872
873     handleToggleNote() {
874         DialogAction.toggleNote();
875     }
876
877     handleToggleAbout() {
878         DialogAction.toggleAbout();
879     }
880
881     handleToggleRegister() {
882         DialogAction.toggleRegister();
883     }
884
885     handleToggleCloseAllDialog() {
886         DialogStore.toggleCloseAllDialog();
887     }
888
889     handleToggleCloseDialog(name) {
890         DialogStore.toggleCloseDialog(name);
891     }
892
893     handleToggleOpenDialog(name) {
894         DialogStore.toggleOpenDialog(name);
895     }
896
897     handleToggleOption() {
898         DialogAction.toggleOption();
899     }
900
901     handleToggleNote() {
902         DialogAction.toggleNote();
903     }
904
905     handleToggleAbout() {
906         DialogAction.toggleAbout();
907     }
908
909     handleToggleRegister() {
910         DialogAction.toggleRegister();
911     }
912
913     handleToggleCloseAllDialog() {
914         DialogStore.toggleCloseAllDialog();
915     }
916
917     handleToggleCloseDialog(name) {
918         DialogStore.toggleCloseDialog(name);
919     }
920
921     handleToggleOpenDialog(name) {
922         DialogStore.toggleOpenDialog(name);
923     }
924
925     handleToggleOption() {
926         DialogAction.toggleOption();
927     }
928
929     handleToggleNote() {
930         DialogAction.toggleNote();
931     }
932
933     handleToggleAbout() {
934         DialogAction.toggleAbout();
935     }
936
937     handleToggleRegister() {
938         DialogAction.toggleRegister();
939     }
940
941     handleToggleCloseAllDialog() {
942         DialogStore.toggleCloseAllDialog();
943     }
944
945     handleToggleCloseDialog(name) {
946         DialogStore.toggleCloseDialog(name);
947     }
948
949     handleToggleOpenDialog(name) {
950         DialogStore.toggleOpenDialog(name);
951     }
952
953     handleToggleOption() {
954         DialogAction.toggleOption();
955     }
956
957     handleToggleNote() {
958         DialogAction.toggleNote();
959     }
960
961     handleToggleAbout() {
962         DialogAction.toggleAbout();
963     }
964
965     handleToggleRegister() {
966         DialogAction.toggleRegister();
967     }
968
969     handleToggleCloseAllDialog() {
970         DialogStore.toggleCloseAllDialog();
971     }
972
973     handleToggleCloseDialog(name) {
974         DialogStore.toggleCloseDialog(name);
975     }
976
977     handleToggleOpenDialog(name) {
978         DialogStore.toggleOpenDialog(name);
979     }
980
981     handleToggleOption() {
982         DialogAction.toggleOption();
983     }
984
985     handleToggleNote() {
986         DialogAction.toggleNote();
987     }
988
989     handleToggleAbout() {
990         DialogAction.toggleAbout();
991     }
992
993     handleToggleRegister() {
994         DialogAction.toggleRegister();
995     }
996
997     handleToggleCloseAllDialog() {
998         DialogStore.toggleCloseAllDialog();
999     }
1000
1001     handleToggleCloseDialog(name) {
1002         DialogStore.toggleCloseDialog(name);
1003     }
1004
1005     handleToggleOpenDialog(name) {
1006         DialogStore.toggleOpenDialog(name);
1007     }
1008
1009     handleToggleOption() {
1010         DialogAction.toggleOption();
1011     }
1012
1013     handleToggleNote() {
1014         DialogAction.toggleNote();
1015     }
1016
1017     handleToggleAbout() {
1018         DialogAction.toggleAbout();
1019     }
1020
1021     handleToggleRegister() {
1022         DialogAction.toggleRegister();
1023     }
1024
1025     handleToggleCloseAllDialog() {
1026         DialogStore.toggleCloseAllDialog();
1027     }
1028
1029     handleToggleCloseDialog(name) {
1030         DialogStore.toggleCloseDialog(name);
1031     }
1032
1033     handleToggleOpenDialog(name) {
1034         DialogStore.toggleOpenDialog(name);
1035     }
1036
1037     handleToggleOption() {
1038         DialogAction.toggleOption();
1039     }
1040
1041     handleToggleNote() {
1042         DialogAction.toggleNote();
1043     }
1044
1045     handleToggleAbout() {
1046         DialogAction.toggleAbout();
1047     }
1048
1049     handleToggleRegister() {
1050         DialogAction.toggleRegister();
1051     }
1052
1053     handleToggleCloseAllDialog() {
1054         DialogStore.toggleCloseAllDialog();
1055     }
1056
1057     handleToggleCloseDialog(name) {
1058         DialogStore.toggleCloseDialog(name);
1059     }
1060
1061     handleToggleOpenDialog(name) {
1062         DialogStore.toggleOpenDialog(name);
1063     }
1064
1065     handleToggleOption() {
1066         DialogAction.toggleOption();
1067     }
1068
1069     handleToggleNote() {
1070         DialogAction.toggleNote();
1071     }
1072
1073     handleToggleAbout() {
1074         DialogAction.toggleAbout();
1075     }
1076
1077     handleToggleRegister() {
1078         DialogAction.toggleRegister();
1079     }
1080
1081     handleToggleCloseAllDialog() {
1082         DialogStore.toggleCloseAllDialog();
1083     }
1084
1085     handleToggleCloseDialog(name) {
1086         DialogStore.toggleCloseDialog(name);
1087     }
1088
1089     handleToggleOpenDialog(name) {
1090         DialogStore.toggleOpenDialog(name);
1091     }
1092
1093     handleToggleOption() {
1094         DialogAction.toggleOption();
1095     }
1096
1097     handleToggleNote() {
1098         DialogAction.toggleNote();
1099     }
1100
1101     handleToggleAbout() {
1102         DialogAction.toggleAbout();
1103     }
1104
1105     handleToggleRegister() {
1106         DialogAction.toggleRegister();
1107     }
1108
1109     handleToggleCloseAllDialog() {
1110         DialogStore.toggleCloseAllDialog();
1111     }
1112
1113     handleToggleCloseDialog(name) {
1114         DialogStore.toggleCloseDialog(name);
1115     }
1116
1117     handleToggleOpenDialog(name) {
1118         DialogStore.toggleOpenDialog(name);
1119     }
1120
1121     handleToggleOption() {
1122         DialogAction.toggleOption();
1123     }
1124
1125     handleToggleNote() {
1126         DialogAction.toggleNote();
1127     }
1128
1129     handleToggleAbout() {
1130         DialogAction.toggleAbout();
1131     }
1132
1133     handleToggleRegister() {
1134         DialogAction.toggleRegister();
1135     }
1136
1137     handleToggleCloseAllDialog() {
1138         DialogStore.toggleCloseAllDialog();
1139     }
1140
1141     handleToggleCloseDialog(name) {
1142         DialogStore.toggleCloseDialog(name);
1143     }
1144
1145     handleToggleOpenDialog(name) {
1146         DialogStore.toggleOpenDialog(name);
1147     }
1148
1149     handleToggleOption() {
1150         DialogAction.toggleOption();
1151     }
1152
1153     handleToggleNote() {
1154         DialogAction.toggleNote();
1155     }
1156
1157     handleToggleAbout() {
1158         DialogAction.toggleAbout();
1159     }
1160
1161     handleToggleRegister() {
1162         DialogAction.toggleRegister();
1163     }
1164
1165     handleToggleCloseAllDialog() {
1166         DialogStore.toggleCloseAllDialog();
1167     }
1168
1169     handleToggleCloseDialog(name) {
1170         DialogStore.toggleCloseDialog(name);
1171     }
1172
1173     handleToggleOpenDialog(name) {
1174         DialogStore.toggleOpenDialog(name);
1175     }
1176
1177     handleToggleOption() {
1178         DialogAction.toggleOption();
1179     }
1180
1181     handleToggleNote() {
1182         DialogAction.toggleNote();
1183     }
1184
1185     handleToggleAbout() {
1186         DialogAction.toggleAbout();
1187     }
1188
1189     handleToggleRegister() {
1190         DialogAction.toggleRegister();
1191     }
1192
1193     handleToggleCloseAllDialog() {
1194         DialogStore.toggleCloseAllDialog();
1195     }
1196
1197     handleToggleCloseDialog(name) {
1198         DialogStore.toggleCloseDialog(name);
1199     }
1200
1201     handleToggleOpenDialog(name) {
1202         DialogStore.toggleOpenDialog(name);
1203     }
1204
1205     handleToggleOption() {
1206         DialogAction.toggleOption();
1207     }
1208
1209     handleToggleNote() {
1210         DialogAction.toggleNote();
1211     }
1212
1213     handleToggleAbout() {
1214         DialogAction.toggleAbout();
1215     }
1216
1217     handleToggleRegister() {
1218         DialogAction.toggleRegister();
1219     }
1220
1221     handleToggleCloseAllDialog() {
1222         DialogStore.toggleCloseAllDialog();
1223     }
1224
1225     handleToggleCloseDialog(name) {
1226         DialogStore.toggleCloseDialog(name);
1227     }
1228
1229     handleToggleOpenDialog(name) {
1230         DialogStore.toggleOpenDialog(name);
1231     }
1232
1233     handleToggleOption() {
1234         DialogAction.toggleOption();
1235     }
1236
1237     handleToggleNote() {
1238         DialogAction.toggleNote();
1239     }
1238
1239     handleToggleAbout() {
1240         DialogAction.toggleAbout();
1241     }
1242
1243     handleToggleRegister() {
1244         DialogAction.toggleRegister();
1245     }
1246
1247     handleToggleCloseAllDialog() {
1248         DialogStore.toggleCloseAllDialog();
1249     }
1250
1251     handleToggleCloseDialog(name) {
1252         DialogStore.toggleCloseDialog(name);
1253     }
1254
1255     handleToggleOpenDialog(name) {
1256         DialogStore.toggleOpenDialog(name);
1257     }
1258
1259     handleToggleOption() {
1260         DialogAction.toggleOption();
1261     }
1262
1263     handleToggleNote() {
1264         DialogAction.toggleNote();
1265     }
1266
1267     handleToggleAbout() {
1268         DialogAction.toggleAbout();
1269     }
1270
1271     handleToggleRegister() {
1272         DialogAction.toggleRegister();
1273     }
1274
1275     handleToggleCloseAllDialog() {
1276         DialogStore.toggleCloseAllDialog();
1277     }
1278
1279     handleToggleCloseDialog(name) {
1280         DialogStore.toggleCloseDialog(name);
1281     }
1282
1283     handleToggleOpenDialog(name) {
1284         DialogStore.toggleOpenDialog(name);
1285     }
1286
1287     handleToggleOption() {
1288         DialogAction.toggleOption();
1289     }
1290
1291     handleToggleNote() {
1292         DialogAction.toggleNote();
1293     }
1294
1295     handleToggleAbout() {
1296         DialogAction.toggleAbout();
1297     }
1298
1299     handleToggleRegister() {
1300         DialogAction.toggleRegister();
1301     }
1302
1303     handleToggleCloseAllDialog() {
1304         DialogStore.toggleCloseAllDialog();
1305     }
1306
1307     handleToggleCloseDialog(name) {
1308         DialogStore.toggleCloseDialog(name);
1309     }
1310
1311     handleToggleOpenDialog(name) {
1312         DialogStore.toggleOpenDialog(name);
1313     }
1314
1315     handleToggleOption() {
1316         DialogAction.toggleOption();
1317     }
1318
1319     handleToggleNote() {
1320         DialogAction.toggleNote();
1321     }
1322
1323     handleToggleAbout() {
1324         DialogAction.toggleAbout();
1325     }
1326
1327     handleToggleRegister() {
1328         DialogAction.toggleRegister();
1329     }
1330
1331     handleToggleCloseAllDialog() {
1332         DialogStore.toggleCloseAllDialog();
1333     }
1334
1335     handleToggleCloseDialog(name) {
1336         DialogStore.toggleCloseDialog(name);
1337     }
1338
1339     handleToggleOpenDialog(name) {
1340         DialogStore.toggleOpenDialog(name);
1341     }
1342
1343     handleToggleOption() {
1344         DialogAction.toggleOption();
1345     }
1346
1347     handleToggleNote() {
1348         DialogAction.toggleNote();
1349     }
1350
1351     handleToggleAbout() {
1352         DialogAction.toggleAbout();
1353     }
1354
1355     handleToggleRegister() {
1356         DialogAction.toggleRegister();
1357     }
1358
1359     handleToggleCloseAllDialog() {
1360         DialogStore.toggleCloseAllDialog();
1361     }
1362
1363     handleToggleCloseDialog(name) {
1364         DialogStore.toggleCloseDialog(name);
1365     }
1366
1367     handleToggleOpenDialog(name) {
1368         DialogStore.toggleOpenDialog(name);
1369     }
1370
1371     handleToggleOption() {
1372         DialogAction.toggleOption();
1373     }
1374
1375     handleToggleNote() {
1376         DialogAction.toggleNote();
1377     }
1378
1379     handleToggleAbout() {
1380         DialogAction.toggleAbout();
1381     }
1382
1383     handleToggleRegister() {
1384         DialogAction.toggleRegister();
1385     }
1386
1387     handleToggleCloseAllDialog() {
1388         DialogStore.toggleCloseAllDialog();
1389     }
1390
1391     handleToggleCloseDialog(name) {
1392         DialogStore.toggleCloseDialog(name);
1393     }
1394
1395     handleToggleOpenDialog(name) {
1396         DialogStore.toggleOpenDialog(name);
1397     }
1398
1399     handleToggleOption() {
1400         DialogAction.toggleOption();
1401     }
1402
1403     handleToggleNote() {
1404         DialogAction.toggleNote();
1405     }
1406
1407     handleToggleAbout() {
1408         DialogAction.toggleAbout();
1409     }
1410
1411     handleToggleRegister() {
1412         DialogAction.toggleRegister();
1413     }
1414
1415     handleToggleCloseAllDialog() {
1416         DialogStore.toggleCloseAllDialog();
1417     }
1418
1419     handleToggleCloseDialog(name) {
1420         DialogStore.toggleCloseDialog(name);
1421     }
1422
1423     handleToggleOpenDialog(name) {
1424         DialogStore.toggleOpenDialog(name);
1425     }
1426
1427     handleToggleOption() {
1428         DialogAction.toggleOption();
1429     }
1430
1431     handleToggleNote() {
1432         DialogAction.toggleNote();
1433     }
1434
1435     handleToggleAbout() {
1436         DialogAction.toggleAbout();
1437     }
1438
1439     handleToggleRegister() {
1440         DialogAction.toggleRegister();
1441     }
1442
1443     handleToggleCloseAllDialog() {
1444         DialogStore.toggleCloseAllDialog();
1445     }
1446
1447     handleToggleCloseDialog(name) {
1448         DialogStore.toggleCloseDialog(name);
1449     }
1450
1451     handleToggleOpenDialog(name) {
1452         DialogStore.toggleOpenDialog(name);
1453     }
1454
14
```

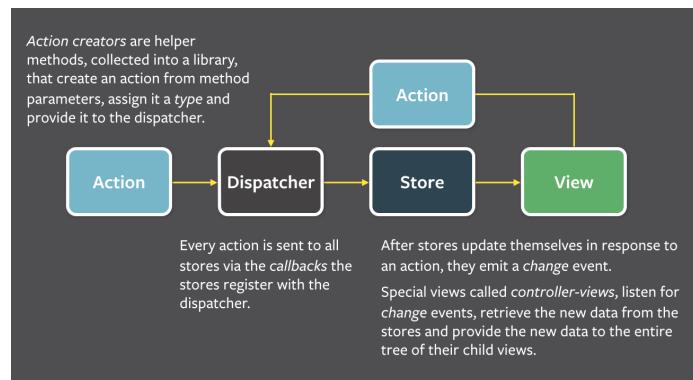
```
31     else
32         this.setState({
33             [name + 'Dialog']: true
34         });
35     },
36
37     closeDialog: function (name) {
38         ...
39         ...
40         ...
41     },
42
43     closeAllDialog: function () {
44         this.setState({
45             noteDialog: false,
46             optionDialog: false,
47             aboutDialog: false,
48             registerDialog: false
49         });
50     },
51
52     handleToggle: function (event, toggled) {
53         this.setState({
54             [event.target.name]: toggled
55         });
56     },
57
58     changeNote: function (event) {
59         this.setState({
60             note: event.target.value
61         });
62     },
63
64     render: function () {
65         var activeDay = this.props.activeDay;
66
67         var noteDialogActions = [
68             <RaisedButton
69                 label='保存'
70                 primary={true}
71                 icon={<Save/>}
72                 onTouchTap={this._saveNote}
```

```
73         />,
74         <FlatButton
75             label='關閉'
76             primary={true}
77             onTouchTap={this.closeDialog.bind(
78                 this, 'note')}
79         />
80     ];
81     ...
82     ...
83     ...
84
85     var styles = {
86         dateController: {
87             background:
88                 COLOR.dateController.background
89             },
90             yearField: {
91                 width: 40
92             },
93             monthField: {
94                 width: 18
95             }
96         };
97
98         return (
99             <Toolbar style={styles.dateController}>
100             ...
101             ...
102             ...
103             </Toolbar>
104         );
105
106         _minusYear: function () {
107             CalendarAction.minusYear();
108         },
109
110         _addYear: function () {
111             CalendarAction.addYear();
112         },
113     };
114 }
```

```
113
114     _changeYear: function (event) {
115         var value = parseInt(event.target.value || 0);
116         if (value < RANGE.MIN_YEAR || value > RANGE.MAX_YEAR)
117             return;
118         var activeDay = this.props.activeDay;
119         activeDay.year = value;
120         CalendarAction.changeDay(activeDay);
121     },
122
123     _minusMonth: function () {
124         CalendarAction_MINUSMonth();
125     },
126
127     _addMonth: function () {
128         CalendarAction.addMonth();
129     },
130
131     _refresh: function(){
132         CalendarAction.refresh();
133     },
134
135     _saveNote: function () {
136         CalendarAction.changeNote(
137             this.props.activeDay,
138             this.state.note.substr(0,140));
139     },
140
141     _saveOption: function () {
142         ConfigAction.setAlwaysOnTop(
143             this.state.alwaysOnTop);
144     },
145
146     _quitApp: function(){
147         CalendarAction.quitApp();
148     }
149 );
150
151 module.exports = DateController;
```

3.3.2 Dispatcher

3.3.2.1 dispatcher.js



Dispatcher是Flux模型里的核心，主要是以下两点：

- View分发actions给Dispatcher，即“触发事件”
- Store在Dispatcher上注册相应action，即“响应事件”

这两者的实现一般放在相应View或者Store的文件里，所以dispatcher.js本身很简单：

react/dispatchers/dispatcher.js

```

1  /**
2   * Created by gou4shi1 on 16-8-16.
3   */
4
5  var Dispatcher = require('flux').Dispatcher;
6
7  module.exports = new Dispatcher();

```

3.3.3 Action

3.3.3.1 calendar-actions.js

这里定义了日历相关的一系列操作的函数，函数执行后会分发相应的action到Dispatcher上

```
react/actions/calendar-actions.js
1  var Dispatcher = require('../dispatchers/dispatcher');
2
3  var TYPE = require('../constants/action-type');
4  var ipc = electronRequire('electron').ipcRenderer;
5  var CalendarAction = {
6      changeDay: function (activeDay) {
7          Dispatcher.dispatch({
8              actionType: TYPE.CHANGE_DAY,
9              activeDay: activeDay
10         });
11     },
12     changeNote: function (activeDay, note) {
13         Dispatcher.dispatch({
14             actionType: TYPE.CHANGE_NOTE,
15             note: note,
16             activeDay: activeDay
17         });
18     },
19     addMonth: function () {
20         Dispatcher.dispatch({
21             actionType: TYPE.ADD_MONTH
22         });
23     },
24     ...
25     ...
26     refresh: function () {
27         Dispatcher.dispatch({
28             actionType: TYPE.REFRESH
29         });
30     },
31     quitApp: function () {
32         ipc.send('app-quit');
33     }
34 };
module.exports = CalendarAction;
```

3.3.3.2 config-actions.js

目前，设置选项里只有“总在最上”一项可供设置：

```
react/actions/config-actions.js
1 | var Dispatcher = require('../dispatchers/dispatcher');
2 | var TYPE = require('../constants/action-type');
3 |
4 | var ConfigAction = {
5 |   setAlwaysOnTop: function (value) {
6 |     Dispatcher.dispatch({
7 |       actionType: TYPE.ALWAYS_ON_TOP,
8 |       value: value
9 |     });
10 |   },
11 | };
12 |
13 | module.exports = ConfigAction;
```

3.3.3.3 dialog-actions.js

根据需求，只有“关闭所有对话框”这一操作需要与Store交互

```
react/actions/dialog-actions.js
1 | var Dispatcher = require('../dispatchers/dispatcher');
2 | var TYPE = require('../constants/action-type');
3 |
4 | var DialogAction = {
5 |   closeDialog: function () {
6 |     Dispatcher.dispatch({
7 |       actionType: TYPE.CLOSE_DIALOG
8 |     });
9 |   },
10 | };
11 |
12 | module.exports = DialogAction;
```

3.3.4 Store

3.3.4.1 calendar-database.js

农历相关的后端数据由NPM里的'lunar-calendar-zh'包提供，calendar-database把'Lunar-calendar-zh'包、黄历、记事本封装在一起

```
react/stores/calendar-database.js
1  var RANGE = require('../constants/calendar-range');
2  var LC = require('lunar-calendar-zh');
3  var HL = require('../huangli/huangli');
4  var Assign = require('object-assign');
5  var _find = require('lodash').find;
6
7  var CalendarDatabase = {
8      noteData : localStorage,
9
10     isInRange: function (year, month, day) {
11         if (year < RANGE.MIN_YEAR || year >
12             RANGE.MAX_YEAR)
13             return false;
14         if (month < RANGE.MIN_MONTH || month >
15             RANGE.MAX_MONTH)
16             return false;
17         if (day < 1 || day > LC.calendar(year, month
18             , false).monthDays)
19             return false;
20         return true;
21     },
22
23     changeNote: function (year, month, day, note) {
24         if (!this.isInRange(year, month, day))
25             return false;
26         year = year.toString();
27         month = (month < 10 ? '0' : '') + month;
28         day = (day < 10 ? '0' : '') + day;
29         this.noteData.setItem(year + month + day,
30             note);
31     },
32
33     getLunarByDay: function (year, month, day) {
34         if (!this.isInRange(year, month, day))
35             return {};
36     }
37 }
```

```
32     var monthData = LC.calendar(year, month,
33         false).monthData;
34     var dayData = _find(monthData, { 'day': day
35         });
36     year = year.toString();
37     month = (month < 10 ? '0' : '') + month;
38     day = (day < 10 ? '0' : '') + day;
39     var hl = HL['hl' + year]['d'+month+day];
40     //encapsulation:
41     return Assign(dayData, {
42         hl_y: hl.y || '',
43         hl_j: hl.j || '',
44         note: this.noteData.getItem(year + month
45             + day) || ''
46     },
47     getLunarByMonth: function (year, month) {
48         if (!this.isInRange(year, month, 1))
49             return {};
50         var monthData = LC.calendar(year, month,
51             true).monthData;
52         //encapsulation:
53         monthData.forEach(function (dayData, index)
54         {
55             monthData[index] = this.getLunarByDay(
56                 dayData.year, dayData.month,
57                 dayData.day);
58             }.bind(this));
59         return monthData;
60     },
61     getMonthDays: function (year, month) {
62         if (!this.isInRange(year, month, 1))
63             return {};
64         return LC.calendar(year, month, false).
65             monthDays;
66     }
67 };
68 module.exports = CalendarDatabase;
```

3.3.4.2 calendar-store.js

有了calendar-database的封装，calendar-store的实现就很简洁了
它本身只有一个属性'activeDay'，即当前选中的日子，默认是今天
然后是一系列get操作与change操作
因为要监听'changeDay'这一事件，所以calendar-store以EventEmitter为
原型

同时也要在Dispatcher上注册相应action

react/stores/calendar-store.js

```
1 var CalendarDatabase = require('./calendar-database');
2 );
3 var Dispatcher = require('../dispatchers/dispatcher');
4 );
5 var TYPE = require('../constants/action-type');
6 var RANGE = require('../constants/calendar-range');
7 var EventEmitter = require('events').EventEmitter;
8 var Assign = require('object-assign');
9
10 var CalendarStore = Assign({},
11     EventEmitter.prototype, {
12         activeDay: false,
13
14         getToday: function () {
15             var date = new Date();
16             return CalendarDatabase.getLunarByDay(
17                 date.getFullYear(), date.getMonth() + 1,
18                 date.getDate());
19         },
20
21         getActiveDay: function () {
22             return this.activeDay || this.getToday();
23         },
24
25         getActiveMonth: function () {
26             var activeDay = this.getActiveDay();
27             return CalendarDatabase.getLunarByMonth(
28                 activeDay.year, activeDay.month);
29         },
30
31         changeDay: function (year, month, day) {
32             //over year
```

```
27         if (month == 13) {
28             year++;
29             month = 1;
30         }
31         if (month == 0) {
32             year--;
33             month = 12;
34         }
35         if (year > RANGE.MAX_YEAR || year <
36             RANGE.MIN_YEAR)
37             return false;
38         if (day > CalendarDatabase.getMonthDays(year,
39             , month))
40             day = 1;
41         this.activeDay =
42             CalendarDatabase.getLunarByDay(year,
43             month, day);
44         this.emit(TYPE.CHANGE_DAY);
45         return true;
46     },
47
48     changeNote: function (activeDay, note) {
49         CalendarDatabase.changeNote(activeDay.year,
50             activeDay.month, activeDay.day, note);
51         this.activeDay =
52             CalendarDatabase.getLunarByDay(
53                 activeDay.year, activeDay.month,
54                 activeDay.day);
55         this.emit(TYPE.CHANGE_DAY);
56     },
57
58     addChangeDayListener: function (callback) {
59         this.on(TYPE.CHANGE_DAY, callback);
60     }
61 );
62
63 Dispatcher.register(function (action) {
64     var activeDay = CalendarStore.getActiveDay();
65
66     switch (action.actionType) {
67         case TYPE.CHANGE_DAY:
68             activeDay = action.activeDay;
```

```
61     CalendarStore.changeDay(activeDay.year,
62                             activeDay.month, activeDay.day);
63     break;
64   case TYPE.ADD_MONTH:
65     CalendarStore.changeDay(activeDay.year,
66                             activeDay.month + 1, activeDay.day);
67     break;
68   ...
69   ...
70
71   case TYPE.REFRESH:
72     activeDay = CalendarStore.getToday();
73     CalendarStore.changeDay(activeDay.year,
74                             activeDay.month, activeDay.day);
75     break;
76   case TYPE.CHANGE_NOTE:
77     CalendarStore.changeNote(
78       action.activeDay, action.note);
79     break;
80   default:
81     break;
82   }
83 });
84 module.exports = CalendarStore;
```

3.3.4.3 dialog-store.js

dialog-store只需要维护对话框的状态，套路与calendar-store类似

```
react/stores/dialog-store.js
1  var Dispatcher = require('../dispatchers/dispatcher');
2  var TYPE = require('../constants/action-type');
3  var EventEmitter = require('events').EventEmitter;
4  var Assign = require('object-assign');
5
6  var DialogStore = Assign({}, EventEmitter.prototype,
7  {
8    note: false,
9    option: false,
10   about: false,
11   register: false,
12
13   getNote: function () {
14     return this.note;
15   },
16   ...
17   ...
18   closeAllDialog: function () {
19     this.note = false;
20     this.option = false;
21     this.about = false;
22     this.register = false;
23     this.emit(TYPE.CLOSE_DIALOG);
24   },
25   addCloseDialogListener: function (callback) {
26     this.on(TYPE.CLOSE_DIALOG, callback);
27   }
28 });
29
30 Dispatcher.register(function (action) {
31   if (action.actionType == TYPE.CLOSE_DIALOG)
32     DialogStore.closeAllDialog();
33 });
34
35 module.exports = DialogStore;
```

3.3.4.4 config-store.js

通过NPM里的'ConfigStore'包把设置选项保存在本地
注意，设置为“总在最上”后，要向ipc传输信号，使应用重启

```
react/stores/config-store.js
1  var Configstore = electronRequire('configstore');
2  var pkg = require('../package.json');
3  var ipc = electronRequire('electron').ipcRenderer;
4  var Dispatcher = require('../dispatchers/dispatcher');
5  var TYPE = require('../constants/action-type');
6
7  var conf = new Configstore(pkg.name);
8
9  var ConfigStore = {
10    alwaysOnTop: conf.get('alwaysOnTop'),
11
12    getAlwaysOnTop: function () {
13      return this.alwaysOnTop;
14    },
15
16    setAlwaysOnTop: function (value) {
17      var restart = (value != conf.get(
18        'alwaysOnTop'));
19      conf.set('alwaysOnTop', value);
20      if (restart)
21        ipc.send('app-restart');
22    }
23  };
24  Dispatcher.register(function (action) {
25    if (action.actionType == TYPE.ALWAYS_ON_TOP)
26      ConfigStore.setAlwaysOnTop(action.value);
27  });
28
29 module.exports = ConfigStore;
```

4 测试情况

4.1 启动界面

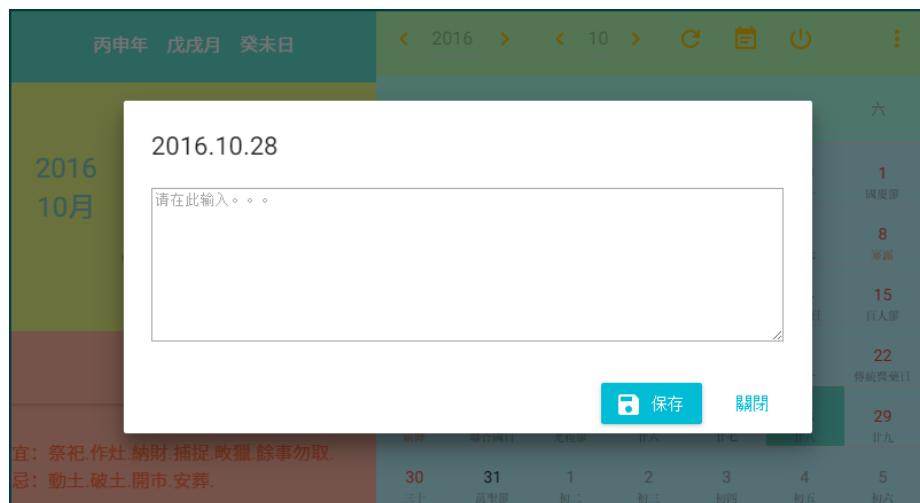
在Windows右下角的托盘栏里单击图标即可启动



4.2 工具栏界面



4.3 记事本界面



4.4 设置菜单



4.5 关于菜单



4.6 占用资源

总共约占用100MB的内存，但是CPU消耗很小

进程名	% CPU	ID	内存
◆ Lunar-Calendar --type=renderer --no-sandbox --primordial-pipe-toker	0	4040	85.7 MiB
◆ Lunar-Calendar --type=gpu-process --channel=4004.0.384429965 --no-sandbox	0	4031	13.5 MiB
◆ Lunar-Calendar --type=zygote --no-sandbox	0	4006	3.1 MiB
◆ Lunar-Calendar	0	4004	18.2 MiB

5 优点与改进

5.1 优点

- 功能强大，实现了一般农历需要实现的所有功能
- 界面优美，采用了Google的Material设计标准

5.2 缺点

- 程序占用硬盘空间过大，多达一百多MB
- 程序占用内存空间较大，约一百MB

6 参考文献

6.1 Q&A

<https://stackoverflow.com>

6.2 Electron

<http://electron.atom.io/>

6.3 Menubar

<https://github.com/maxogden/menubar>

6.4 React

<https://facebook.github.io/react/docs/getting-started.html>

6.5 Flux

<https://facebook.github.io/flux/docs/overview.html>

6.6 browserify

<http://browserify.org/>

6.7 gulp

<http://gulpjs.com/>

6.8 Material Design

<https://material.google.com/>

6.9 Material-ui

<http://www.material-ui.com/>