

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 </body>
12 <!-- reason: https://github.com/electron/electron/
13     issues/1611 -->
14 <script>var electronRequire = require;</script>
15 <script src="build/app.js"></script>
16 </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
11 mb.on('ready', function () {
12     console.log('app is ready');
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     render: function () {
37         var styles = {
38             mainPanel: {
39                 border: '2px solid ' +
40                     COLOR.mainPanel.border
41             };
42
43             return (
44                 <MuiThemeProvider>
45                 <GridList cols={10} cellHeight={432}
46                     padding={0} style={styles.mainPanel}
47                     >
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={
56                                 this.state.activeDay}
57                             activeMonth={
58                                 this.state.activeMonth}
59                             />
60                     </GridTile>
61                 </GridList>
62                 </MuiThemeProvider>
63             );
64         }
65     });
66
67     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
1  var React = require('react');
2  var Transitive = require('react-transitive-number');
3  var COLOR = require('../constants/calendar-color');
4
5  import {GridList, GridTile} from 'material-ui/
   GridList';
6  import Paper from 'material-ui/Paper';
7
8  var LunarPanel = React.createClass({
9    render: function () {
10     var activeDay = this.props.activeDay;
11     var festivalElements = [];
12     var key_index = 0;
13
14     if (activeDay.term) {
15       festivalElements.push(<br key={'festival
16         ' + (key_index++)}/>);
17       festivalElements.push(
18         <Transitive key={'festival' + (
19           key_index++)}>{activeDay.term}</
20         Transitive>
21       );
22     }
23     if (activeDay.lunarFestival) {
24       festivalElements.push(<br key={'festival
25         ' + (key_index++)}/>);
26       festivalElements.push(
27         <Transitive key={'festival' + (
28           key_index++)}>{
29           activeDay.lunarFestival}</
30         Transitive>
31       );
32     }
33
34     var styles = {
35       ganZhiPanel: {
```

```
29         background:
30             COLOR.ganZhiPanel.background,
31         color: COLOR.ganZhiPanel.color,
32         lineHeight: '60px',
33         textAlign: 'center'
34     },
35     ...
36     ...
37     hlPanel: {
38         background: COLOR.hlPanel.background
39         ,
40         color: COLOR.hlPanel.color,
41         textOverflow: 'ellipsis',
42         overflow: 'hidden',
43         whiteSpace: 'nowrap'
44     }
45 };
46
47 return (
48     <GridList cols={10} cellHeight={31}
49     padding={0}>
50     <GridTile cols={10} rows={2} style={
51     styles.ganZhiPanel}>
52     <Transitive>{activeDay.GanZhiYear +
53     '年'}</Transitive>
54     <Transitive>{' ' +
55     activeDay.GanZhiMonth + '月'}</
56     Transitive>
57     <Transitive>{' ' +
58     activeDay.GanZhiDay + '日'}</
59     Transitive>
60     </GridTile>
61     <GridTile cols={10} rows={1} style={
62     styles.dayPanel}>
63     </GridTile>
64     <GridTile cols={3} rows={6} style={
65     styles.yearPanel}>
66     <br/><Transitive>{activeDay.year
67     }</Transitive><br/>
68     <Transitive>{activeDay.month + '
69     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>
74         <GridTile cols={10} rows={2} style={
75             styles.monthPanel}>
76             <Paper zDepth={1} style={
77                 styles.monthDetail}>
78                 <Transitive>{
79                     activeDay.lunarMonthName
80                     + ' '}</Transitive>
81                 <Transitive>{
82                     activeDay.lunarDayName}<
83                     /Transitive>
84             </Paper>
85         </GridTile>
86         <GridTile cols={10} rows={2} style={
87             styles.hlPanel}>
88             '宜' :{activeDay.hl_y}<br/>
89             '忌' :{activeDay.hl_j}
90         </GridTile>
91     </GridList>
92 )
93 }
94 });
95
96 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={
15             this.props.activeMonth}
16         />
17       </div>
18     );
19   }
20 });
21 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/
   calendar-actions');
3  var COLOR = require('../constants/calendar-color');
4
5  import {Table, TableBody, TableHeader,
   TableHeaderColumn, TableRow, TableRowColumn}
   from 'material-ui/Table';
6  import {FlatButton} from 'material-ui'
7
8  var DateChoosePanel = React.createClass({
9    render: function () {
10     var today = this.props.today;
11     var activeDay = this.props.activeDay;
12     var activeMonth = this.props.activeMonth;
13
14     var table = [];
15     var tableBody = [];
16     var tableRow = [];
17
18     var styles = {
19       tableHeader: {
20         background:
21           COLOR.tableHeader.background
22       },
23       tableBody: {
24         background:
25           COLOR.tableBody.background
26       },
27       orangeLabel: {
28         lineHeight: 0,
29         fontSize: 10,
30         color: COLOR.orangeLabel.color
31       },
32       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     tableRow.push(
92         <TableRowColumn style={
93             styles_tableCell} key={'
94             tableRowColumn' + i}>
95         <FlatButton
96             label={date.day}
97             labelStyle={
98                 styles_numberLabel}
99             labelPosition={'before'}
100             rippleColor={
101                 COLOR.numberLabel.color.ripple
102             }
103             onTouchTap={
104                 this._onClick.bind(this,
105                     date)}
106             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   componentDidMount: function () {
15     DialogStore.addCloseDialogListener(
16       this.closeAllDialog);
17     ipc.on('app-hide', function () {
18       DialogAction.closeDialog();
19     });
20   },
21   openDialog: function (name) {
22     if (name == 'note')
23       this.setState({
24         note: this.props.activeDay.note,
25         [name + 'Dialog']: true
26       });
27     else if (name == 'option')
28       this.setState({
29         alwaysOnTop:
30           ConfigStore.getAlwaysOnTop(),
31         [name + 'Dialog']: true
32       });
33   }
34 });
```

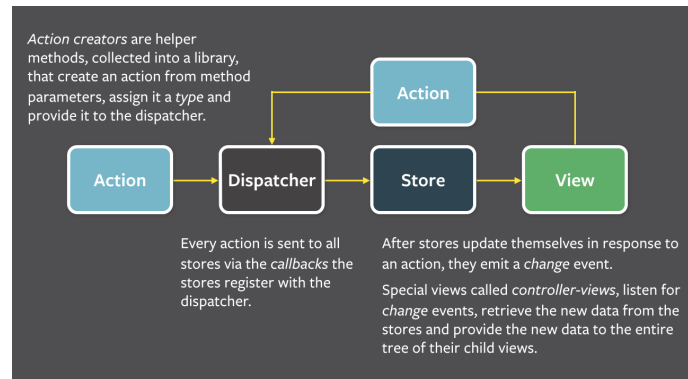
```
31         else
32             this.setState({
33                 [name + 'Dialog']: true
34             });
35     },
36
37     closeDialog: function (name) {
38         ...
39         ...
40         ...
41     },
42
43     closeAllDialog: function () {
44         this.setState({
45             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             onTap={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         </Toolbar>
103     );
104 },
105
106 _minusYear: function () {
107     CalendarAction.minusYear();
108 },
109
110 _addYear: function () {
111     CalendarAction.addYear();
112 },
```

```
113
114   _changeYear: function (event) {
115       var value = parseInt(event.target.value ||
116           0);
117       if (value < RANGE.MIN_YEAR || value >
118           RANGE.MAX_YEAR)
119           return;
120       var activeDay = this.props.activeDay;
121       activeDay.year = value;
122       CalendarAction.changeDay(activeDay);
123   },
124
125   _minusMonth: function () {
126       CalendarAction.minusMonth();
127   },
128
129   _addMonth: function () {
130       CalendarAction.addMonth();
131   },
132
133   _refresh: function(){
134       CalendarAction.refresh();
135   },
136
137   _saveNote: function () {
138       CalendarAction.changeNote(
139           this.props.activeDay,
140           this.state.note.substr(0,140));
141   },
142
143   _saveOption: function () {
144       ConfigAction.setAlwaysOnTop(
145           this.state.alwaysOnTop);
146   },
147
148   _quitApp: function(){
149       CalendarAction.quitApp();
150   }
151 });
152
153 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 | var TYPE = require('../constants/action-type');
3 | var ipc = electronRequire('electron').ipcRenderer;
4 | var CalendarAction = {
5 |   changeDay: function (activeDay) {
6 |     Dispatcher.dispatch({
7 |       actionType: TYPE.CHANGE_DAY,
8 |       activeDay: activeDay
9 |     });
10 |   },
11 |   changeNote: function (activeDay, note) {
12 |     Dispatcher.dispatch({
13 |       actionType: TYPE.CHANGE_NOTE,
14 |       note: note,
15 |       activeDay: activeDay
16 |     });
17 |   },
18 |   addMonth: function () {
19 |     Dispatcher.dispatch({
20 |       actionType: TYPE.ADD_MONTH
21 |     });
22 |   },
23 |   ...
24 |   ...
25 |   refresh: function () {
26 |     Dispatcher.dispatch({
27 |       actionType: TYPE.REFRESH
28 |     });
29 |   },
30 |   quitApp: function () {
31 |     ipc.send('app-quit');
32 |   }
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 |       actionTypes: 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 |       actionTypes: 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 },
48
49 getLunarByMonth: function (year, month) {
50     if (!this.isInRange(year, month, 1))
51         return {};
52     var monthData = LC.calendar(year, month,
53         true).monthData;
54     //encapsulation:
55     monthData.forEach(function (dayData, index)
56     {
57         monthData[index] = this.getLunarByDay(
58             dayData.year, dayData.month,
59             dayData.day);
60     }).bind(this));
61     return monthData;
62 },
63
64 getMonthDays: function (year, month) {
65     if (!this.isInRange(year, month, 1))
66         return {};
67     return LC.calendar(year, month, false).
68         monthDays;
69 }
70 };
71
72 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  var Dispatcher = require('../dispatchers/dispatcher')
    );
3  var TYPE = require('../constants/action-type');
4  var RANGE = require('../constants/calendar-range');
5  var EventEmitter = require('events').EventEmitter;
6  var Assign = require('object-assign');
7
8  var CalendarStore = Assign({},
    EventEmitter.prototype, {
9    activeDay: false,
10
11    getToday: function () {
12      var date = new Date();
13      return CalendarDatabase.getLunarByDay(
14        date.getFullYear(), date.getMonth() + 1,
15        date.getDate());
16    },
17
18    getActiveDay: function () {
19      return this.activeDay || this.getToday();
20    },
21
22    getActiveMonth: function () {
23      var activeDay = this.getActiveDay();
24      return CalendarDatabase.getLunarByMonth(
25        activeDay.year, activeDay.month);
26    },
27
28    changeDay: function(year, month, day) {
29      //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 addChangeListener: 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 });
85 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   note: false,
8   option: false,
9   about: false,
10  register: false,
11
12  getNote: function () {
13    return this.note;
14  },
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 );
6 var TYPE = require('../constants/action-type');
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

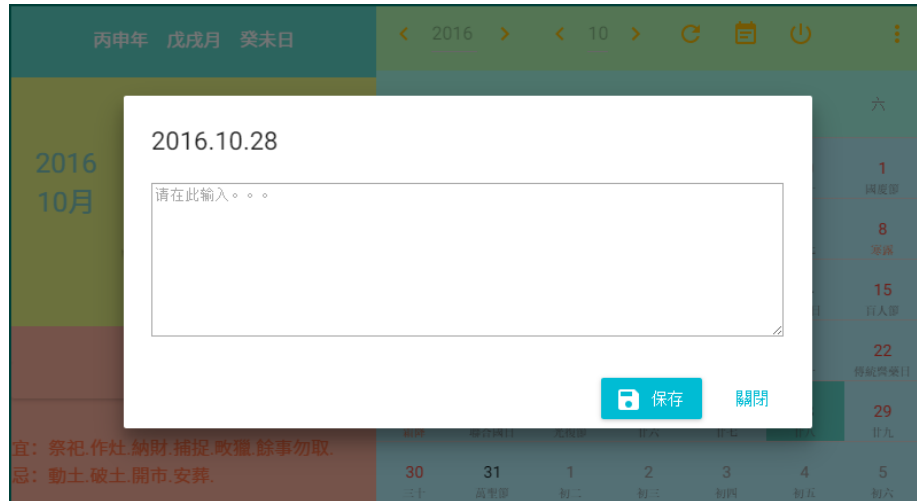
在Windows右下角的托盘栏里单击图标即可启动



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 -mo	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/>