Axure扩展库

Axure的前端组件库,在RP9.x版本通过测试

组件

- ## axlib-v3

  核心库，包含多项实用的axure操作等函数

  ### 加载方式

  - 在Axure中添加 *“加载时-打开外部链接”* 的交互，然后输入以下代码：

    ```
    $axure.utils.loadJS('https://ax.minicg.com/axlib-v3.min.js', ()=>{
      console.log(axlib);
    });
    ```

  - 也可以通过添加网络字体的方式加载，代码如下：

    ```
    "><script src="https://ax.minicg.com/axlib-v3.min.js"></script><link href="
    ```

  ### 功能扩展

  - **中继器 - Repeater**

    ```
    /*
     * 获取中继器数据（实时）
     * format:
     * default - 返回最原始的 axure 中继器数据格式
     * auto - 按照每行获取数据，类型为数组，数组内每个元素都是对象，并标记每列的值
     * row - 按照每行获取数据，类型为对象，键值为首列每行的值
     * col - 按照每列获取数据，类型为对象，键值为列名（小写）
     * tree - 返回树状数据结构，另外配合参数：trim 可移除空值，表头必须命名为C1,C2,C3...
     */

    // 获取中继器数据（仅获取初始化时的数据，用axure交互、或用代码操作过的数据都不会被获取到）
    $axure('@中继器名称').getRepeaterDataOnce({ format: 'row' });

    // 插入中继器数据（仅接受axure原始数据格式）
    $axure('@中继器名称').addRepeaterData(
      [
        {
          'col1': { type: 'text', text: 'r1c1' },
          'col2': { type: 'text', text: 'r1c2' },
        },
        {
    ```

```
      'col1': { type: 'text', text: 'r2c1' },
      'col2': { type: 'text', text: 'r2c2' },
    }
  ]
);

// 删除n行中继器数据
let rows = 3;
$axure('@中继器名称').deleteRepeaterData(rows);

// 刷新中继器（插入、更新、删除、清空数据后需要刷新才能获取实时数据）
// 会发射一个 REPEATER_UPDATED 的事件，可通过 repeater.$().on('REPEATER_UPDATED', callback) 侦听
$axure('@中继器名称').refreshRepeater();

// 覆盖设置中继器数据（仅接受axure原始数据格式）
$axure('@中继器名称').setRepeaterData(data);

// 清空中继器数据
$axure('@中继器名称').clearRepeaterData();

// 侦听中继器更新（也支持add、delete等）
$axure('@中继器名称').on('REPEATER_DATA_UPDATED', (e, res) => {
  console.log(res);
});
```

- **动态面板 - Dynamic Panel**

```
// 获取动态面板所有状态
$axure('@动态面板名称').getPanelStates();

// 获取动态面板当前状态
$axure('@动态面板名称').getCurrentPanelState();

// 动态面板跳转到指定名称的状态
let name = 'State 1';
$axure('@动态面板名称').setPanelStateByName(name);

// 动态面板跳转到指定索引
$axure('@动态面板名称').setPanelStateByIndex(1);
```

- **页面跳转**

```
// 注入方法到页面跳转方法（原生交互也可触发）
$axure.beforeNav((options,stop)=>{
  console.log(options); /* 这里其实可以加入转场效果（实测） */
  stop(); /* 执行这句会阻断原生跳转 */
});

// 手动跳转
$axure.navigate({
  url: 'page_1.html',
  target: 'frame',
  includeVariables: true,
  frame: $iframe.get(0)
});
```

- ##### 简化获取ID与节点的方法
  ```js
  // 返回第一个同名元素的 id
  $axure('@元素名称').getId();

  // 返回第一个同名元素的DOM节点
  $axure('@元素名称').getEl();
  ```

- ##### 设置和获取全局变量
  ```js
  // 设置全局变量（需要先在 Axure 里添加该全局变量名称才能生效）
  axlib.setGlobalVar('varName', 'value');

  // 获取全局变量的值
  axlib.getGlobalVar('varName');
  ```

- ##### 设置鼠标指针样式
  ```js
  // 设置元素的指针样式为手形
  $axure('@元素名称').buttonMode(true);

  // 设置元素不感应任何鼠标事件
  $axure('@元素名称').ghostMode(true);
  ```

附加功能

- ##### 加载插件

  ```js
  const { toast, swal2 } = axlib.plugins;
  yepnope({
  ```

```
    load: [...toast.files, ...swal2.files],
    complete: ()=>{ console.log($.toast, Swal) }
  });
  ```

- ##### 改变界面布局

  ```js
  /*
   * 参数:
   * 0 ~ 默认布局（保留折叠）
   * 1 ~ 隐藏顶栏，默认缩放
   * 2 ~ 隐藏顶栏，按宽度缩放
   * 3 ~ 隐藏顶栏，按高度缩放
   */
  axlib.layout(2);
  ```

- ##### 在线读取与保存数据（新）
  ```js
  // 初始化 （参数：仓库id，可在 https://json.minicg.com 获得）
  let bin = new jsonbin('64a6db339d312622a37b3ebc');

  // 读取数据
  bin.read().then(res=>console.log(res));

  // 更新数据（数据覆盖）
  bin.update({foo:'bar'});

  // 创建新仓库（需要先设置 API-KEY）
  bin.create();

  // 删除仓库（需要先设置 API-KEY）
  bin.delete();

  // 设置 API-KEY
  bin.apiKey = 'YOUR_API_KEY';

  // 设置 ACCESS-KEY（可单独设置create/delete/read/update权限）
  bin.accessKey = 'YOUR_ACCESS_KEY';
  ```

- ##### 插入加载动画
  ```js
  let container = $axure('@target').$().get(0);
  let show = true;
  let options = {
    color:'#fff',
    opacity:0.2,
```

```
    scale:1.2,
    delay:0,
    dur:1,
    blending:'difference'
  };
  axlib.loading(container, show, options);
  ```

- ##### 获取多个元件
  ```js
  /*
   * 范例代码等同于：
   * let sym1 = $axure('@sym1');
   * let sym2 = $axure('@sym2');
   * let sym3 = $axure('@sym3');
   */
  let { sym1, sym2, sym3 } = axlib.find('sym1', 'sym2', 'sym3');
  ```

- ##### 保存文本文件
  ```js
  // 浏览器会提示下载文件
  let text = 'Hello Axlib!';
  let fileName = 'hello.txt';
  axlib.saveTextFile(text, fileName);
  ```

- ##### 测试表达式
  ```js
  let fn = function(){};
  axlib.testExp({
    exp: 'swal',    // 检查表达式 window.swal 是否不等于 undefined
    success: fn,    // 如果表达式为真，则执行此函数
    fail: fn,       // 如果表达式在超时后仍然为不为真，则执行此函数
    interval: 200,  // 检测的时间间隔
    timeout: 5000,  // 超时上限
    debug: false    // 启用 debug 模式（打印检测过程）
  });
  ```

- ##### 隐藏 Axure 默认打印的 dispatchMessage
  ```js
  //axlib.hideMsg(); 有bug，已弃用
  ```

- # axlib-jsbox (v2.x)

可直接在 Axure 里写JS代码的中继器组件，可方便设定需要先加载的 js 和 css 地址，自动预加载 axlib，编辑器内的代码会等待预加载的文件成功载入后再执行，且变量作用域与外部隔离，内置`THIS`变量可获取自身的各种属性。

效果预览 & 组件下载

https://axhub.im/ax9/203a5819ebe3e37b

使用方法

- 打开范例 rp 文件，将组件（一个特制的中继器）复制到你的文件中，双击组件打开编辑器输入代码即可；
- 若需要先加载外部css或js，在 **“载入时-添加行”** 中将文件的链接添加到第一列（注意：如果列名后期修改过，需要点击一下这个交互打开编辑窗口以刷新数据，随后不用做任何操作关闭窗口即可）；
- 可在交互中通过 **“启用/禁用”** 和预先设置的 **“禁用的样式”** 来改变盒子外观；
- 可在交互中通过 **“设置尺寸”** 修改盒子的宽高；

编辑器内置

> 双击 jsbox 组件进入中继器内部，在文本域中自由编写 JavaScript，请继续阅读以下代码范例：

```
// 加载外部代码（以 "//import" 开头，注意import前没有空格，import后有一个英文空格）：
// 注意：这种方式引入的代码无法访问 "THIS"
//import your_css_or_js_url
```

```
/*
 * 获取组件产生的 视图容器 节点
 *
 * THIS.view 返回的是 axure 元素
 * 若要获取该节点的 jquery 元素，需要写成 THIS.$view
 * 若要转为原生的 DOM 节点，需要写成 THIS.$view.get(0)，或直接写成 THIS.view.getEl()
 *
 * ！注意：此节点从 v2.2.0 开始从中继器内部移出到外部，以避免中继器数据变化时被 Axure 清理掉
 * 但是此改动也会导致获取的元素存在时效性，如果要获取中继器内最新的节点，可以通过以下新增的方法：
 * THIS.getView() / THIS.getRepeater() / THIS.getTextarea() /
THIS.getAllTextareas()
 */
console.log(THIS.view);

// 获取本体预加载的文件列表
console.log(THIS.files);

// 获取本体的中继器的 axure对象 和 jquery对象
console.log(THIS.repeater, THIS.$repeater);

// 获取 中继器id 和 视图id
console.log(THIS.rid, THIS.vid);
```

```
// 获取中继器的 位置 和 尺寸
console.log(THIS.size, THIS.position);

// 清除 view 上的样式
THIS.setViewStyle('none');

// 设置文本域的文本内容
THIS.setText(string);

// 获取文本域的文本内容
THIS.getText();

// 设置文本域的串联字符串用于返回js结果给axure（会在字符串开头增加一个分隔符）
// marker默认值为"/", 假设数组为：["值1","值2","值3"], 返回值则为："/值1/值2/值3"
THIS.setValues(array, marker);

// 显示 view （可以同时控制通明度和是否感应鼠标事件）
let alpha = 1, pointEvent = 'none';
THIS.showView(alpha, pointEvent);

// 隐藏 view （当仅作为插入代码的媒介，不需要显示本体时可以用这个）
THIS.hideView();

// 在当前容器内追加插入一段文本：“Hello Axlib!”
let html = '<h1 style="color:#000;">Hello Axlib!</h1>';
THIS.view.$().append(html);
THIS.appendHTML(html);

// 设置当前容器的html内容（清除原内容）
THIS.setHTML('<h2>新的内容<h2>');

// 插入全局 CSS 样式
THIS.appendCSS(`
  .your-class-1 { background: #0CF; }
  .your-class-2 { background: #FC0; }
`);

// 传入ax、jq或原生节点对象，转为原生节点
THIS.getNode(el);

// 获取自身中继器上通过样式面板配置的初始数据，[详见](#功能扩展)
THIS.getData({ format: 'row' });

// 同上，该方法只是一个别名
THIS.getDataOnce({ format: 'row' });

// 获取自身中继器除了样式面板配置数据以外的数据
THIS.getExtraData({ format: 'row' });
```

```
// 获取自身中继器的“实时最新的数据”
THIS.getLatestData({ format: 'row' });

// 删除中继器N行数据，会自动刷新中继器
THIS.deleteData(3);

// 清空中继器所有数据
THIS.clearData();

// 删除中继器非样式面板配置的数据，会自动刷新中继器
THIS.clearExtraData();

// 显示和隐藏 loading 动画
THIS.showLoading();
THIS.hideLoading();

/*
 * 侦听中继器数据变化，接受：
 * REPEATER_DATA_ADDED - 添加数据
 * REPEATER_DATA_DELETED - 删除数据
 * REPEATER_DATA_UPDATED - 数据重写
 * REPEATER_REFRESH - 刷新中继器
 */
THIS.$repeater.on('事件名称', e=>{ THIS.getLatestData({format:'col'}); });

// 文本域发射一个移动事件，用于触发原生交互
THIS.emit('moved');

// 指定Axure元件发射一个事件，目前支持：moved(移动时), selected(选中时), unselected(未选中时), toggleSelect(切换选择)
THIS.emit('moved', $axure('@元件名'));

// 外置脚本，等同于 eval('https://127.0.0.1/script.js')，可方便用外部 IDE（如 VSCode）编写代码
// 注意：外置脚本也可以访问变量"THIS"，但"//import"语法糖还是要写在jsbox里，不能够放在外部脚本里
THIS.runScript('url');

// 清空中继器内所有文本域标签内的内容
THIS.clearAllTextareas();

// 获取用户添加到中继器的元素（除 .ax-jsbox* 外）
THIS.getUserElements();
```