UIBuilder 模板使用指南

背景

本模板会使用 `UIBuilder` 将复杂的 UI 渲染简化成一行命令调用的渲染框架，以降低搭建 UI 的成本。

使用方式

由于初始化 `UIBuilder` 需要若干步骤，所以建议直接 Fork UIBuilder Template（或者github地址），然后在 `src/runUIBuilder.tsx` 文件的 `main` 函数内调用 `UIBuilder` 的方法。

> 🚂 你可以把 `UIBuilder` 的使用看作是组装一列火车。你的工作就是添加车厢（组件），当用户点击表单按钮时，`UIBuilder` 会自动卸掉这个表单按钮后面的所有车厢，所以你无需关心如何卸载车厢（组件）。

示例

查找替换 UI

```
import { bitable, UIBuilder } from "@lark-base-open/js-sdk";

export default async function main(uiBuilder: UIBuilder) {
  uiBuilder.form(form => ({
    formItems: [
        form.tableSelect('table', {label:'选择数据表'}),
        form.fieldSelect('field', {label: '选择字段', sourceTable: 'table', mode: 'multiple'}),
        form.input('findText', {label: '输入查找的文本'}),
        form.input('replaceText', {label: '替换为'}),
    ],
    buttons: ['确认'],
  }), async ({ values }) => {
    const { table, field, findText, replaceText } = values;
    // 省略实现查找逻辑的代码
    uiBuilder.text(`查找到 n 条记录`);
    uiBuilder.buttons('是否替换', ['确定'], () => {
        // 省略实现替换逻辑的代码
    });
```

```
  });
}
```

预览视图：

> 由于不在多维表格内，所以数据表选择组件和字段选择组件为空，可以将 https://ext.baseopendev.com/ext/uibuilder-template-find-replace/9d818e66cb85a55412a361c96de79fc43d13c5d3/index.html粘贴到多维表格自定义插件中进行体验

https://ext.baseopendev.com/ext/uibuilder-template-find-replace/9d818e66cb85a55412a361c96de79fc43d13c5d3/index.html

URL 转附件 UI

```
import { bitable, UIBuilder, FieldType } from "@lark-base-open/js-sdk";

export default async function main(uiBuilder: UIBuilder) {
  uiBuilder.form(form => ({
    formItems: [
      form.tableSelect('table', { label: '选择数据表' }),
      form.fieldSelect('urlField', { label: '要转换的 URL 字段', sourceTable: 'table', filterByTypes: [FieldType.Url] }),
      form.fieldSelect('attachmentField', { label: '转换到的附件字段', sourceTable: 'table', filterByTypes: [FieldType.Attachment] }),
    ],
    buttons: ['批量转换'],
  }), async ({ values }) => {
    const { table, urlField, attachmentField } = values;
    // 省略转换逻辑
  });
}
```

预览视图：

> 由于不在多维表格内，所以数据表选择组件和字段选择组件为空，可以将
>
> https://ext.baseopendev.com/ext/uibuilder-template-url-to-attachment/c78162f4aa76f355f8ad5738b159520e005dd59e/index.html粘贴到多维表格自定义插件中进行体验

https://ext.baseopendev.com/ext/uibuilder-template-url-to-attachment/c78162f4aa76f355f8ad5738b159520e005dd59e/index.html

`uiBuilder.form(formBuilder, callback)`

在页面绘制一个表单

参数	类型	是否必填	说明
formBuilder	`(form) => { formItems: Array<FormItem>, buttons: Array<string> }`	是	在 `formItems` 中描述表单项，在 `buttons` 和描述按钮的文案
callback	{key: string, values: {[key: string]: unkown}}	是	点击表单按钮的回调事件，其中 1. `key` 为用户所点击按钮的 `key` ，你可以根据不同的key做不同的操作 2. `values` 为表单值

示例代码

```
import { bitable, UIBuilder } from "@lark-base-open/js-sdk";

export default async function main(uiBuilder: UIBuilder) {
  uiBuilder.markdown(`
  ## 这是一个 UIBuilder 的演示插件
  你可以在 \`uiBuilder.markdown\` 或者 \`uiBuilder.text\` 中输出交互内容
  `);
  uiBuilder.form((form) => ({
    formItems: [
      form.tableSelect('table', { label: '选择数据表' }),
      form.viewSelect('view', { label: '选择视图', sourceTable: 'table' }),
      form.fieldSelect('field', { label: '选择字段', sourceTable: 'table', multiple: true }),
      form.input('text', { label: '输入文本', defaultValue: '文本默认值' }),
      form.inputNumber('number', { label: '输入数字', defaultValue: 28 }),
      form.textArea('textArea', { label: '输入多行文本' }),
```

```
16        form.checkboxGroup('checkbox', { label: '选择水果', options: ['Apple',
    'Orange'], defaultValue: ['Apple'] }),
17        form.select('select', { label: '下拉选择器', options: [{ label: 'Apple',
    value: 'Apple' }, { label: 'Orange', value: 'Orange' }], defaultValue: 'Apple'
     }),
18      ],
19      buttons: ['确定', '取消'],
20    }), async ({ key, values }) => {
21      const { table, view, field, text, number, textArea, checkbox, select } =
    values;
22      uiBuilder.markdown(`你点击了**${key}**按钮`);
23    });
24  }
```

预览视图：

https://ext.baseopendev.com/ext/uibuilder-template-form/9a47b2e2b147ddb646ac11bd3947bdbe991a2526/index.html

https://ext.baseopendev.com/ext/uibuilder-template-form/9a47b2e2b147ddb646ac11bd3947bdbe991a2526/index.html

`form.tableSelect(key, args)`

在表单内声明一个数据表选择组件，如果在多维表格内会自动获取当前文档数据表列表，在用户点击按钮后返回 `table` 对象

参数	类型	是否必填	说明
key	string	是	该表单项的id，需要在表单内唯一
args	object	是	表单项的描述信息
args.label	string	是	表单项的label
args.placeholder	string	否	占位文案

`form.viewSelect(key, args)`

在表单内声明一个视图选择组件，如果在多维表格内会自动获取当前文档的视图列表，在用户点击按钮后返回 `view` 对象

参数	类型	是否必填	说明
key	string	是	该表单项的id，需要在表单内唯一
args	object	是	表单项的描述信息
args.label	string	是	表单项的label
args.placeholder	string	否	占位文案
args.sourceTable	string	是	数据表选择组件的id，用户选择数据表后，框架会将对应数据表的视图作为本组件的选项
args.multiple	boolean	否	设置为多选模式
args.filterByTypes	Array<ViewType>	否	过滤视图的类型，不填则为全类型展示
args.filter	(option: IViewMeta) => boolean	否	自定义过滤选项的逻辑

示例代码

```
import { bitable, UIBuilder } from "@lark-base-open/js-sdk";

export default async function main(uiBuilder: UIBuilder) {
  uiBuilder.form((form) => ({
    formItems: [
      form.tableSelect('table', { label: '选择数据表' }),
      form.viewSelect('view', { label: '选择视图', sourceTable: 'table' }),
      form.viewSelect('multipleView', { label: '多选选择视图', sourceTable: 'table', multiple: true }),
      form.tableSelect('table2', { label: '选择数据表2' }),
      // 视图选择组件 view2 会随 table2 数据表选择组件的选项变化而更新
      form.viewSelect('view2', { label: '选择视图2', sourceTable: 'table2' }),
    ],
    buttons: ['确定', '取消'],
  }), async ({ values }) => {
    const { table, view, multipleView, table2, view2 } = values;
  });
}
```

预览视图：

> 由于不在多维表格内，所以数据表选择组件和视图选择组件为空。如需深度体验，可将 https://ext.baseopendev.com/ext/uibuilder-template-view/3dfc35e8b8b173723dc2571f0b990090e1fb3d42/index.html粘贴到多维表格自定义插件中运行。

https://ext.baseopendev.com/ext/uibuilder-template-view/3dfc35e8b8b173723dc2571f0b990090e1fb3d42/index.html

`form.fieldSelect(key, args)`

在表单内声明一个字段选择组件，如果在多维表格内会自动获取当前文档的字段列表。

参数	类型	是否必填	说明
key	string	是	该表单项的id，需要在表单内唯一
args	object	是	表单项的描述信息
args.label	string	是	表单项的label
args.placeholder	string	否	占位文案
args.sourceTable	string	是	数据表选择组件的id，用户选择数据表后，框架会将对应数据表的字段作为本组件的选项
args.multiple	boolean	否	设置为多选模式
args.filterByTypes	Array<FieldType>	否	过滤字段的类型，不填则为全类型展示
args.filter	(option: IFieldMeta) => boolean	否	自定义过滤选项的逻辑

示例代码

```
import { FieldType, UIBuilder } from '@lark-base-open/js-sdk';

export default async function main(uiBuilder: UIBuilder) {
  uiBuilder.form(
    (form) => ({
      formItems: [
        form.tableSelect('table', { label: '选择数据表' }),
```

```
        form.fieldSelect('multipleField', { label: '多选模式', sourceTable: 'table', multiple: true }),
        form.fieldSelect('urlField', {
          label: '要转换的 URL 字段',
          sourceTable: 'table',
          // 只展示类型为链接的字段
          filterByTypes: [FieldType.Url],
        }),
        form.fieldSelect('selectField', {
          label: '单选或多选字段',
          sourceTable: 'table',
          // 只展示名字包含单选或者多选的字段
          filter: ({ name }) => name.includes('单选') || name.includes('多选'),
        }),
        form.fieldSelect('peopleField', {
          label: '工作人员字段',
          sourceTable: 'table',
          // 只展示名字包含人员且类型为人员的字段
          filter: ({ name, type }) => name.includes('人员') && type === FieldType.User,
        }),
      ],
      buttons: ['确定'],
    }),
    async ({ values }) => {
      const { table, multipleField, urlField, selectField, peopleField } = values;
    }
  );
}
```

预览视图：

> 由于不在多维表格内，所以数据表选择组件和字段选择组件为空。如需深度体验，可将 https://ext.baseopendev.com/ext/uibuilder-template-filed/5b9f806636579fb2a24e535e2c99b00c1d776da8/index.html粘贴到多维表格自定义插件中运行。

https://ext.baseopendev.com/ext/uibuilder-template-filed/5b9f806636579fb2a24e535e2c99b00c1d776da8/index.html

`form.select(key, args)`

在表单内声明一个下拉选择器组件

参数	类型	是否必填	说明
key	string	是	该表单项的id，需要在表单内唯一
args	object	是	表单项的描述信息
args.label	string	是	表单项的label
args.options	Array<{label: string, value: string}>	是	下拉选择器的可选项
args.defaultValue	Array<string | number>	否	默认选中的选项
args.multiple	boolean	否	是否开启多选模式
args.tags	boolean	否	是否开启允许用户自定义新选项的标签模式

示例代码

```
export default async function main(uiBuilder) {
  uiBuilder.form((form) => ({
    formItems: [
      form.select('select', { label: '基本使用', options: [{ label: 'Apple', value: 'Apple' }, { label: 'Orange', value: 'Orange' }], defaultValue: 'Apple' }),
      form.select('multipleSelect', { label: '多选模式', options: [{ label: 'Apple', value: 'Apple' }, { label: 'Orange', value: 'Orange' }], multiple: true, defaultValue: ['Apple', 'Orange'] }),
      form.select('tags', { label: '标签模式（允许用户输入新选项）', options: [{ label: 'Apple', value: 'Apple' }, { label: 'Orange', value: 'Orange' }], tags: true, defaultValue: ['Apple', 'Orange'] }),
    ],
    buttons: ['确定'],
  }), async ({ values }) => {
    const { select, multipleSelect, tags } = values;
  });
}
```

预览视图：

https://ext.baseopendev.com/ext/uibuilder-template-select/a0fcf485bd7370fb8d143a3c38741aed7208fc1e/index.html

`form.input(key, args)`

在表单内声明一个文本输入组件

参数	类型	是否必填	说明
key	string	是	该表单项的id，需要在表单内唯一
args	object	是	表单项的描述信息
args.label	string	是	表单项的label
args.placeholder	string	否	占位文案
args.defaultValue	string	否	输入框默认内容

示例代码

```
export default async function main(uiBuilder) {
  uiBuilder.form((form) => ({
    formItems: [
      form.input('text', { label: '基本使用', placeholder: 'Basic usage' }),
      form.input('text2', { label: '设置默认值', defaultValue: '文本默认值' }),
    ],
    buttons: ['确定', '取消'],
  }), async ({ values }) => {
    const { text, text2 } = values;
  });
}
```

https://ext.baseopendev.com/ext/uibuilder-template-input/4056ad202a71725ce2511764dde124524259f99d/index.html

`form.inputNumber(key, args)`

在表单内声明一个数字输入组件

参数	类型	是否必填	说明

key	string	是	该表单项的id，需要在表单内唯一
args	object	是	表单项的描述信息
args.label	string	是	表单项的label
args.placeholder	string	否	占位文案
args.defaultValue	string	否	输入框默认内容
args.prefix	string	否	前缀内容

示例代码

```
export default async function main(uiBuilder) {
  uiBuilder.form((form) => ({
    formItems: [
      form.inputNumber('number', { label: '基本使用', defaultValue: '28' }),
      form.inputNumber('number2', { label: '添加前缀', prefix: '￥' }),
    ],
    buttons: ['确定', '取消'],
  }), async ({ values }) => {
    const { number, number2 } = values;
  });
}
```

https://ext.baseopendev.com/ext/uibuilder-template-input-number/52508b96b0b07694d4d7866cbfea41e18b16a8b7/index.html

`form.textArea(key, args)`

在表单内声明一个多行文本组件

参数	类型	是否必填	说明
key	string	是	该表单项的id，需要在表单内唯一
args	object	是	表单项的描述信息

args.label	string	是	表单项的label
args.placeholder	string	否	占位文案
args.defaultValue	string	否	输入框默认内容
args.autoSize	boolean | object	否	自适应内容高度，可设置为 true | false 或对象：{ minRows: 2, maxRows: 6 }

示例代码

```
export default async function main(uiBuilder) {
  uiBuilder.form((form) => ({
    formItems: [
      form.textArea('textarea', { label: '基本使用', placeholder: 'Basic usage' }),
      form.textArea('textarea2', { label: '设置高度', autoSize: { minRows: 4 } }),
    ],
    buttons: ['确定', '取消'],
  }), async ({ values }) => {
    const { textarea, textarea2 } = values;
  });
}
```

预览视图：

https://ext.baseopendev.com/ext/uibuilder-template-textarea/534c14b480b794f99328ae68ad599b9c31b6fae5/index.html

https://ext.baseopendev.com/ext/uibuilder-template-textarea/534c14b480b794f99328ae68ad599b9c31b6fae5/index.html

`form.checkboxGroup(key, args)`

在表单内声明一个多选框组件

参数	类型	是否必填	说明
key	string	是	

			该表单项的id，需要在表单内唯一
args	object	是	表单项的描述信息
args.label	string	是	表单项的label
args.options	Array<string | number| {label: string, value: string}>	是	Checkbox可选项
args.defaultValue	Array<string | number>	否	默认选中的选项

示例代码

```
export default async function main(uiBuilder) {
  uiBuilder.form((form) => ({
    formItems: [
      form.checkboxGroup('checkbox', { label: '选择水果', options: ['Apple', 'Orange'], defaultValue: ['Apple'] }),
    ],
    buttons: ['确定', '取消'],
  }), async ({ values }) => {
    const { checkbox } = values;
  });
}
```

预览视图：

https://ext.baseopendev.com/ext/uibuilder-template-checkbox/3db59f581e8e8044e15dadea45c82819653ebeec/index.html

https://ext.baseopendev.com/ext/uibuilder-template-checkbox/3db59f581e8e8044e15dadea45c82819653ebeec/index.html

`form.radio(key, args)`

在表单内声明一个单选框组件

参数	类型	是否必填	说明
key	string	是	

			该表单项的id，需要在表单内唯一
args	object	是	表单项的描述信息
args.label	string	是	表单项的label
args.options	Array<string | number| {label: string, value: string}>	是	Radio可选项
args.defaultValue	Array<string | number>	否	默认选中的选项

> 💡 你可修改左侧 Script 面板下的代码，然后点击右侧「运行」按钮，修改会实时生效（该修改只对你生效）

https://code.juejin.cn/pen/7287475543237197883

`uiBuilder.buttons(label, args, callback)`

在页面绘制一个按钮，在回调里返回用户点击结果

参数	类型	是否必填	说明
label	string	是	用于向用户解释应该点击哪个按钮
args	Array<string | {label: string, key: string}>	是	按钮的描述信息
args.label	string	是	按钮的label
args.key	string	否	按钮的id
callback	(key: string) => void	否	用户的点击回调

示例代码

直接传入字符串数组

```
export default async function main(uiBuilder) {
  uiBuilder.buttons('Cat or Dog?', ['Cat', 'Dog'], catOrDog => {
    uiBuilder.text(`You click ${catOrDog}`);
  });
```

```
5  }
```

预览视图：

https://ext.baseopendev.com/ext/uibuilder-template-button/9c835b51ec8f65893188156e9bdc815f258f133a/index.html

https://ext.baseopendev.com/ext/uibuilder-template-button/9c835b51ec8f65893188156e9bdc815f258f133a/index.html

`uiBuilder.text(source)`

在页面上输出纯文本

参数	类型	是否必填	说明
source	string	是	希望输出的字符串文本

示例代码

```
1  export default async function main(uiBuilder) {
2    uiBuilder.text('hello world');
3  }
```

`uiBuilder.markdown(source)`

在页面上输出 Markdown 信息

参数	类型	是否必填	说明
source	string	是	希望输出的Markdown文本

示例代码

```
1  export default async function main(uiBuilder) {
2    uiBuilder.markdown(`# A demo of \`uiBuilder.markdown\`
```

```
    如果你想了解某些语法元素的更多信息，请参阅更详细的 [基本语法](https://markdown.com.cn/basic-syntax/) 和 [扩展语法](https://markdown.com.cn/extended-syntax/)
    ## Markdown 语法速查表
    | 元素 | Markdown语法 |
    | --- | --- |
    | 标题（Heading） | \`# H1\` \`## H2\` \`### H3\`|
    | 粗体（Bold） | \`**bold text**\` |
    | 斜体（Italic）） | \`*italicized text*\` |
    | 引用块（Blockquote） | \`> blockquote\` |
    | 有序列表（Ordered List） | \`1. First item\` |
    | 无序列表（Unordered List） | \`- First item\` |
    | 代码（Code） | \`\` \`code\` \`\` |
    | 分隔线（Horizontal Rule） | \`---\` |
    | 链接（Link） | \`[title](https://www.example.com)\` |
    | 图片（Image） | \`![alt text](image.jpg)\` |
    | 表格（Table） | ✅ |
    | 删除线（Strikethrough） | \`~~The world is flat.~~\` |
    | 任务列表（Task List） | \`- [x] Write the press release\` |
    | 代码块（Fenced Code Block） | ❌（暂不支持） |
  `);
  }
```

预览视图：

https://ext.baseopendev.com/ext/uibuilder-template-markdown/d1d3310ee091b276491981d4c526641093875128/index.html

https://ext.baseopendev.com/ext/uibuilder-template-markdown/d1d3310ee091b276491981d4c526641093875128/index.html

`uiBuilder.message(content, [duration], onClose)`

在页面上弹出反馈信息，支持以下类型的 message

- `message.success(content, [duration], onClose)`
- `message.error(content, [duration], onClose)`
- `message.info(content, [duration], onClose)`
- `message.warning(content, [duration], onClose)`
- `message.loading(content, [duration], onClose)`

参数	类型	是否必填	说明
content	string	是	提示内容
duration	number	否	自动关闭的延时，单位秒。设为 0 时不自动关闭
onClose	function	否	关闭时触发的回调函数

示例代码

```
export default async function main(uiBuilder) {
  uiBuilder.buttons('', ['Show message'], () => {
    uiBuilder.message.success('Hello world!')
  });
}
```

预览视图：

https://ext.baseopendev.com/ext/uibuilder-template-message/97917ec2ddf199d861a11a836f4a2a0da52aa6bf/index.html

https://ext.baseopendev.com/ext/uibuilder-template-message/97917ec2ddf199d861a11a836f4a2a0da52aa6bf/index.html

`uiBuilder.showLoading(message)`

显示 loading 提示框，需要主动调用 `uiBuilder.hideLoading` 才能关闭提示框

参数	类型	是否必填	说明
`message`	string	否	提示的内容

示例代码

```
export default async function main(uiBuilder) {
  uiBuilder.buttons('', ['Show loading'], () => {
    uiBuilder.showLoading('Loading...');
    // 1000 毫秒后隐藏 loading
```

```
    setTimeout(() => {
      uiBuilder.hideLoading();
    }, 1000);
  });
}
```

预览视图：

https://ext.baseopendev.com/ext/uibuilder-template-loading/18b9fb9a539ad5912b740d7555e9866c8f594601/index.html

https://ext.baseopendev.com/ext/uibuilder-template-loading/18b9fb9a539ad5912b740d7555e9866c8f594601/index.html

`uiBuilder.reload()`

UIBuilder 会在用户交互后，重新渲染交互组件之后的所有元素，如果你想重新渲染之前的元素可以使用 `reload` 方法。

> 💡 你可修改左侧 Script 面板下的代码，然后点击右侧「运行」按钮，修改会实时生效（该修改只对你生效）

https://code.juejin.cn/pen/7282688523059576890

常见问题

根据不同条件，渲染不同的表单组件

目前尚不支持根据条件选择性渲染表单内的组件，一个可选的解决方案是根据条件渲染整个表单。

> 💡 你可修改左侧 Script 面板下的代码，然后点击右侧「运行」按钮，修改会实时生效（该修改只对你生效）

https://code.juejin.cn/pen/7275602202109394978

支持国际化

UIBuilder 已内置国际化方案，你只需提供国际化的文案即可

1. 在 `src/App.tsx` 文件取消注释以引入已存在的 `i18n.ts` 文件

```
import './App.css';
import { useEffect } from 'react';
import { bitable, UIBuilder } from "@lark-base-open/js-sdk";
import callback from './runUIBuilder';
import { useTranslation } from 'react-i18next';
import './i18n'; // 取消注释以启用国际化
```

2. 在 `src/i18n/resources.ts` 文件配置你的国际化文案，比如指定 `Welcome to UIBuilder` 在中文环境下文案为「欢迎使用 UIBuilder」，在英文环境下为「Welcome to UIBuilder」

```
const resources = {
  zh: {
    translation: {
      // 定义你的中文文案
      'Welcome to UIBuilder': '欢迎使用 UIBuilder',
    },
  },
  en: {
    translation: {
      // Define your English text
      'Welcome to UIBuilder': 'Welcome to UIBuilder',
    },
  },
};

export default resources;
```

3. 在 `src/runUIBuilder.tsx` 中通过 `t` 函数国际化 `key` 如 `Welcome to UIBuilder`

```
import { bitable, UIBuilder } from "@lark-base-open/js-sdk";
import { UseTranslationResponse } from 'react-i18next';

export default async function(uiBuilder: UIBuilder, { t }:
UseTranslationResponse<'translation', undefined>) {
  uiBuilder.markdown(`## ${translation.t('Welcome to UIBuilder')}`);
}
```

4. 至此，插件在中文语言环境下会展示中文「欢迎使用 UIBuilder」，在英文语言环境下就会展示「Welcome to UIBuilder」，你可将 https://uibuilder-i18n.lark-base.repl.co 粘贴到多维表格插件的 URL中，将多维表格语言切换为英文体验

关于国际化的更多用法可见 i18next 文档。

单复数

UIBuilder内置的国际化方案支持通过 `zero` 、 `one` 、 `other` 等关键词指定文案的单复数形式，详情可见i18next Plurals 文档。

> 💡 注意变量名必须为 `count`

```
// 声明国际化 key 时加上关键字
{
  "key_zero": "{{count}} item",
  "key_one": "{{count}} item",
  "key_other": "{{count}} items",
}
```

```
// 使用时指定 count 的值，i18next 会自动渲染对应的单复数文案
t('key', {count: 0}); // -> 0 item
t('key', {count: 1}); // -> 1 item
t('key', {count: 2}); // -> 2 items
```

升级指南

API 的最新版本是

```
@lark-base-open/js-sdk@0.3.2
```

如果你 `package.json` 文件中不存在 `@lark-base-open/js-sdk` ，或 `@lark-base-open/js-sdk` 是低于这个版本，推荐的方式是 Fork UIBuilder Template 然后将你项目中 `src/main.ts` 或 `src/runUIBuilder.ts` 文件的内容拷贝到新项目的 `src/runUIBuilder.tsx` 文件中，或参考以下步骤升级：

将依赖包由 `@base-open/web-api` 切换成 `@lark-base-open/js-sdk`

1. 在控制台执行以下命令安装最新版的 `@lark-base-open/js-sdk` 和相关依赖

```
npm i @lark-base-open/js-sdk@latest i18next i18next-browser-languagedetector
react-i18next -S
```

2. 新增 `src/i18n/index.ts` 文件，并将以下内容添加到文件中

```
import i18n from 'i18next';
import { initReactI18next } from 'react-i18next';
import LanguageDetector from 'i18next-browser-languagedetector';
import { bitable } from '@lark-base-open/js-sdk';
import resources from './resources';

i18n
  .use(LanguageDetector)
  .use(initReactI18next)
  .init({
    resources,
    fallbackLng: 'en', // 指定降级文案为英文
    interpolation: {
      escapeValue: false,
    },
  });

bitable.bridge.getLanguage().then((lng) => {
  if (i18n.language !== lng) {
    i18n.changeLanguage(lng);
  }
});

export default i18n;

```

3. 新增 `src/i18n/resources.ts` 文件，并将一下内容添加到文件中

```
const resources = {
  zh: {
    translation: {
      // 定义你的中文文案
      'Welcome to UIBuilder': '欢迎使用 UIBuilder',
    },
  },
  en: {
    translation: {
```

```
        // Define your English text
        'Welcome to UIBuilder': 'Welcome to UIBuilder',
      },
    },
};

export default resources;
```

4. 将以下内容覆盖到你的 `src/App.tsx` 文件中

```
import './App.css';
import { useEffect } from 'react';
import { bitable, UIBuilder } from "@lark-base-open/js-sdk";
import callback from './runUIBuilder';
import { useTranslation } from 'react-i18next';
import './i18n';

export default function App() {
  const translation = useTranslation();
  useEffect(() => {
    const uiBuilder = new UIBuilder(document.querySelector('#container') as HTMLElement, {
      bitable,
      callback,
      translation,
    });
    return () => {
      uiBuilder.unmount();
    };
  }, [translation]);
  return (
    <div id='container'></div>
  );
}
```

适配部分API的更新

1. tableSelect、viewSelect、fieldSelect组件的回调值由字符串变为对象

```
import { bitable } from "@lark-base-open/js-sdk";
export default async function main(uiBuilder: UIBuilder) {
  uiBuilder.form(
    (form) => ({
```

```
      formItems: [
-       form.tableSelect('tableId', { label: '选择数据表' }),
-       form.viewSelect('viewId', { label: '选择视图', sourceTable: 'tableId' }),
-       form.fieldSelect('fieldId', { label: '选择字段', sourceTable: 'tableId' }),
        // 将 key 由 tableId/viewId/fieldId 改为 table/view/field
+       form.tableSelect('table', { label: '选择数据表' }),
+       form.viewSelect('view', { label: '选择视图', sourceTable: 'table' }),
+       form.fieldSelect('field', { label: '选择字段', sourceTable: 'table' }),
      ],
      buttons: ['确定', '取消'],
    }),
    async ({ values }) => {
      const { table, view, field } = values;
-     const table = await bitable.base.getTableById(table);
-     const field = await table.getFieldById(field);
-     const view = await table.getViewById(view);

      // 无需再调用方法获取数据表/字段/视图，如有需要，访问 ".id" 即可获取数据表id、字段id、视图id
+     const tableId = table.id;
+     const fieldId = field.id;
+     const viewId = view.id;
    }
  );
}
```

2. viewSelect、fieldSelect组件的 `filter` 参数变为IViewMeta和IFieldMeta

```
import { FieldType, UIBuilder } from '@lark-base-open/js-sdk';

export default async function main(uiBuilder: UIBuilder) {
  uiBuilder.form(
    (form) => ({
      formItems: [
        form.tableSelect('table', { label: '选择数据表' }),
        form.fieldSelect('peopleField', {
          label: '工作人员字段',
          sourceTable: 'table',
          // 字段名从 label 改为 name 属性
-         filter: ({ label, type }) => label.includes('人员') && type === FieldType.User,
```

```
13  +         filter: ({ name, type }) => name.includes('人员') && type === FieldType.User,
14            }),
15          ],
16          buttons: ['确定'],
17        }),
18        async ({ values }) => {
19          const { table, multipleField, urlField, selectField, peopleField } = values;
20        }
21      );
22    }
```

Roadmap

在使用中如遇到功能缺失，欢迎大家评论补充👋

功能优化

p0

- [ ] 增加 `uiBuilder.auth` 方法，方便服务端插件获取 personalBaseToken
- [ ] 增加配置文件如 config.ts
- [ ] 数据表选择组件监听多维表格的onTableAdd、onTableDelete事件，以兼容字段/视图的实时更新
- [ ] 数据表/视图/字段选择器选项
  - [ ] 支持设置默认值
  - [ ] 支持搜索
- [ ] 第一个表单组件自动 autoFocus
- [ ] 支持条件渲染

p1

- [ ] 在非多维表格内打开则显示顶部提示栏，并支持配置关闭
- [ ] 字段选择器选项
  - [ ] 展示字段图标
  - [ ] 希望支持sourceView
  - [ ] 希望支持异步过滤，参数为field实例，这样就可以过滤出代理了多行文本的公式字段
- [ ] 数据表/视图/字段选择组件
  - [ ] 增加一个固定选项为新建数据表/视图/字段

- [ ] Select组件
  - [ ] 支持一键全选
  - [ ] 支持搜索
- [ ] 预览使用所见即所得的编辑器如 CodePen
- [ ] 表单支持声明必填/可选项
- [ ] 表单支持定义校验规则
- [ ] 表单支持未通过校验规则时按钮置灰

p2

- [ ] 支持关闭自动清除后续 UI 的开关
- [ ] 支持指定 Button 类型
- [ ] 支持自定义表单控件、支持自定义 UI 控件
- [ ] 默认支持暗黑模式
- [ ] 增加 `uiBuilder.table` 方法，以表格方式反馈信息
- [ ] 支持关闭默认选中当前数据表
- [ ] 增加重置API用于重新渲染当前页面，如 `uiBuilder.reload`
- [ ] 补全内部的类型
- [ ] 支持紧凑模式
- [ ] 底层切换为 Semi Design

更多组件支持

p1

- [ ] 增加弹窗组件
- [ ] uiBuilder.tableSelect、uiBuilder.viewSelect 等独立的组件
- [ ] 文件上传组件
- [ ] 日期选择组件
- [ ] 时间选择组件

p2

- [ ] Switch组件
- [ ] Radio组件
- [ ] Rate组件
- [ ] 颜色选择组件

- [ ] Slider组件