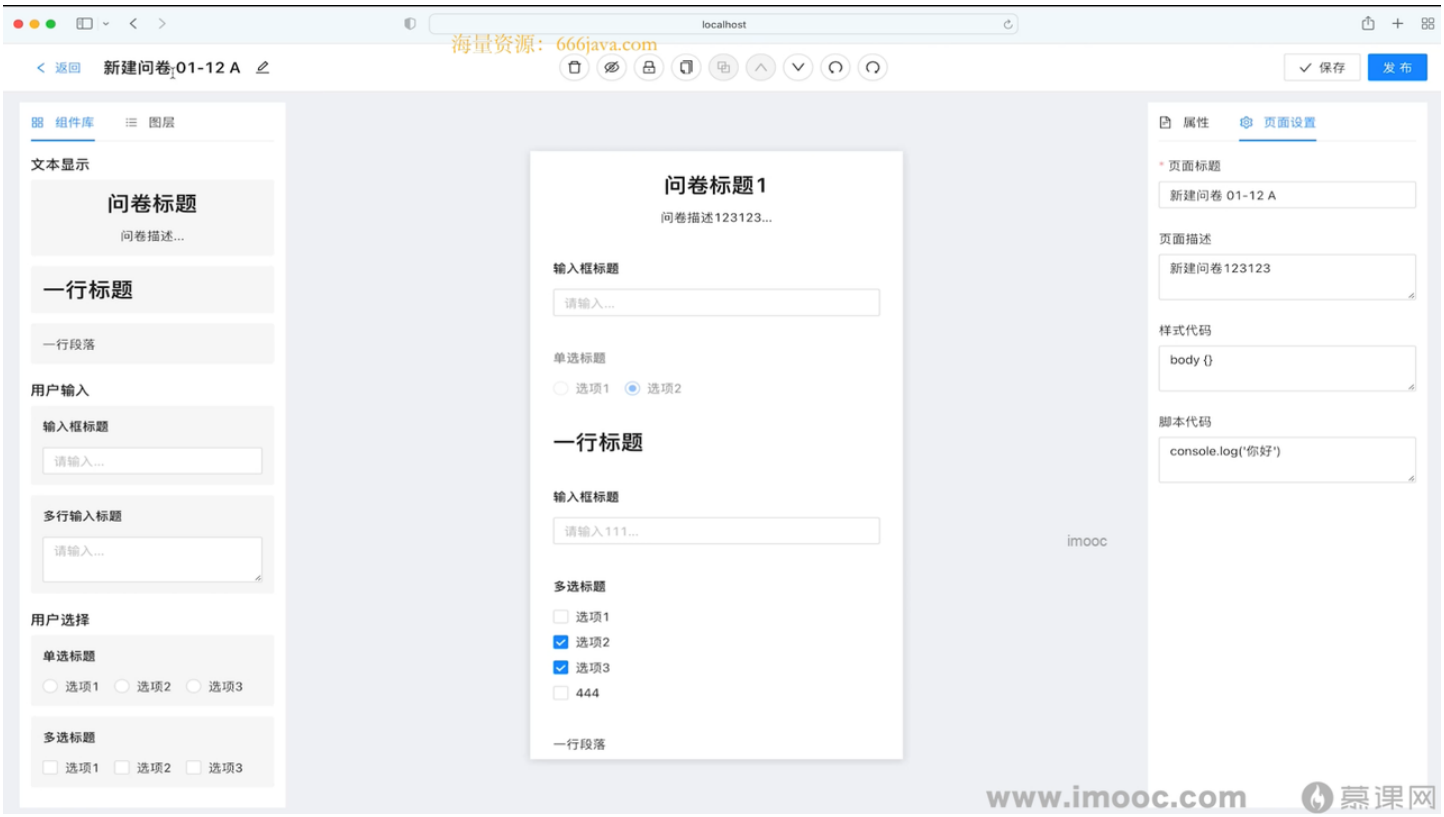
12 问卷编辑器的开发

需求分析



页面模块

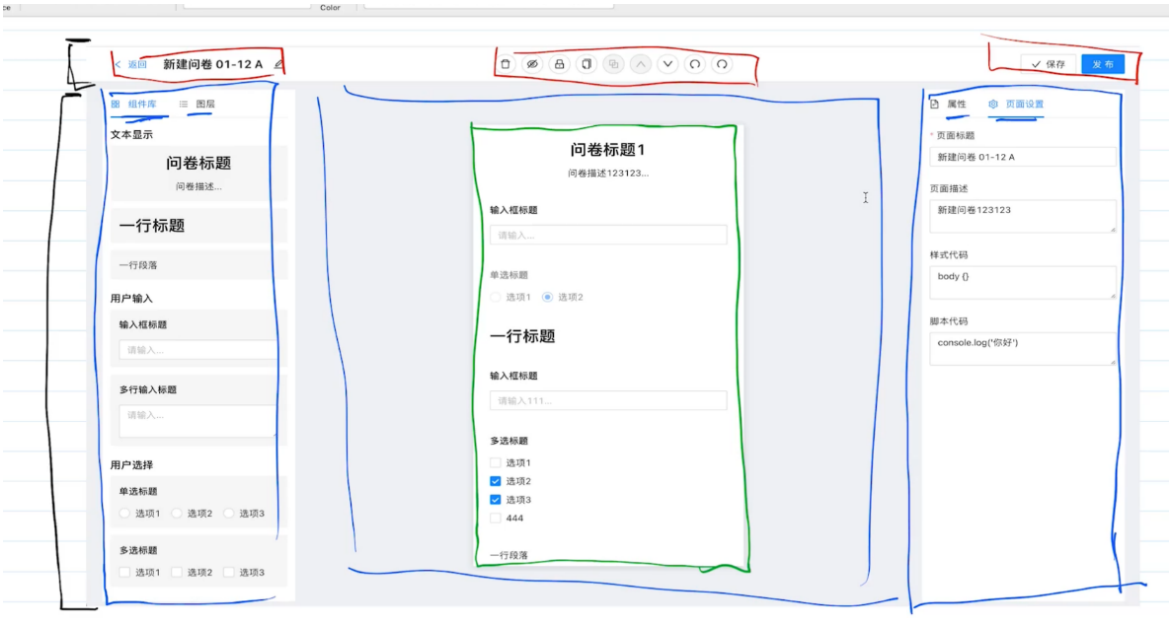
- 顶部 - **顶部栏**
- 左侧 - **组件库** + **图层**
- 中间 - **画布**
- 右侧 - **属性** + **页面设置**

功能描述

1. **顶部栏**
    - 返回
    - 显示标题，修改标题
    - 工具栏（删除、隐藏、锁定、复制、粘贴、上移，下移、撤销，重做）
    - 保存，自动保存，ctrl + s 快捷键
    - 发布

2. 左侧-**组件库**
    - 显示组件列表（各个组件，看系统）
    - 点击添加组件到画布
3. 左侧-**图层**
    - 显示图层列表
    - 拖拽排序
    - 单击，选中
    - 双击，修改标题
    - 隐藏、锁定
4. 中间-**画布**
    - 展示组件列表
    - Y 滚动条
    - 拖拽排序
    - 单击，选中
    - 快捷键
        - delete backspace
        - up
        - down
        - ctrl + c , v
        - ctrl + z , ctrl + shift + z
        - ctrl + s ，保存
5. 右侧-**属性**
    - 修改属性
6. 右侧-**页面设置**
    - 标题，描述
    - JS CSS 代码
    - tab 自动切换：点击组件，切换到属性，取消点中，切换到页面设置

UI 设计



这里设计的组件位于 QuestionLayout.tsx 中，同时使用 flex 进行组件的拆分和布局。需要注意的是，为了让路由组件充满页面，需要设置 QuestionLayout.tsx 的最外层元素的样式为 `{height: \`calc(100vh - 16px)\`}`。上图是对编辑页面的 UI 的拆分，参考以设计问卷编辑器（位于 src/pages/Question/Edit/index.tsx 中）。

页面结构

```tsx
<div className={styles.container}>
  {/* Header - 工具栏 */}
  <div className={styles.header}>Header</div>
  {/* Content */}
  <div className={styles['content-wrapper']}>
    <div className={styles.content}>
      {/* 左侧（组件库 + 图层） */}
      <div className={styles.left}>Left</div>
      {/* 中间（画布） */}
      <div className={styles.main}>
        <div className={styles['canvas-wrapper']}>
          <div className={styles.canvas}>Canvas</div>
        </div>
      </div>
      {/* 右侧（属性 + 页面设置） */}
      <div className={styles.right}>Right</div>
    </div>
  </div>
</div>
```

页面样式（布局）

```scss
$componentBgColor: #fff;

.container {

    height: 100%;
    background-color: #f0f2f5;

    display: flex;
    flex-direction: column;

    /* 顶部工具栏 */
    .header {
        height: 40px;
        background-color: $componentBgColor;
    }

    .content-wrapper {
        /* 关于 flex: auto 的解释
            - flex 是 flex-grow flex-shrink flex-basis 的复合属性
            - flex: auto <==> flex: 1 1 auto <==> 允许拉伸、允许压缩、不设置基准长度（即默认以伸缩项目的宽或高为准，如果设置了的话） */
        flex: auto;
        /* 关于 padding: <length> <length> 的解释
            - 第一个长度值表示 padding-top/bottom
            - 第二个长度值表示 padding-left/right */
        padding: 12px 0;
        /* margin 设置两个值与 padding 同义 */
        margin: 0 24px;

        .content {
            height: 100%;
            display: flex;

            /* 左侧组件库&图层 */
            .left {
                width: 285px;
                background-color: $componentBgColor;
                padding: 0 12px;
            }

            /* 中间画布 */
            .main {
                /* 关于 flex: 1 的解释
                    - flex: 1 <==> flex: 1 1 0 <==> 允许拉伸、允许压缩、基准长度为 0
                    - 由于 content 下的三个伸缩项目只有 main 的 flex-grow 为 1，其他都是默认值 0 ==> main 占据剩余主轴空间
```

```
                    - 简而言之，flex: 1 是一种响应式处理 ==> 让元素能够动态地根据容器的可用空间扩展或收缩。 */
                flex: 1;
                /* 关于使用定位 position 居中的解释
                    - 父元素 relative 子元素 absolute ==> 此时父元素是子元素的包含块，同时子元素脱离文档流
                    - 子元素 top: 50%; left: 50%，再通过位移向 X、Y 轴的负方向移动自身的百分之五十 ==> 子元素在父元素中完成居中 */
                position: relative;
                overflow: hidden;

                .canvas-wrapper {
                    width: 400px;
                    // height: 712px;
                    height: calc(100% - 20px);
                    background-color: #fff;
                    /* 关于 box-shadow 的使用
                        - box-shadow: h-shadow(阴影的水平位置) v-shadow(阴影的垂直位置) blur(模糊距离) spread(阴影的外延值) color(阴影的颜色) inset(将外部阴影变为内部阴影)
                        - h-shadow 和 v-shadow 都是必选的，其余是可选的。数值默认值为 0。*/
                    box-shadow: 0 2px 10px #0000001f;

                    /* 子元素绝对定位配合父元素的相对定位实现水平垂直居中 */
                    position: absolute;
                    top: 50%;
                    left: 50%;
                    transform: translateX(-50%) translateY(-50%);

                    /* 实现画布的 y 轴滚动 */
                    overflow: auto;

                    .canvas {
                        height: 900px;
                    }
                }
            }

            /* 右侧属性&页面设置 */
            .right {
                width: 300px;
                background-color: $componentBgColor;
                padding: 0 12px;
            }
        }
    }
```

```
}
/*
flex: 1 Vs. flex: auto
- 初始尺寸
    flex: 1; 的元素初始尺寸为 0%，意味着它会尽可能地扩展以填充容器。
    flex: auto; 的元素初始尺寸基于内容的大小，意味着它会先考虑内容的自然尺寸，然后再考虑扩展或收缩。
- 行为
    flex: 1; 的元素始终试图扩展以填满容器，而不管内容的实际大小。
    flex: auto; 的元素首先尝试保持其内容的固有尺寸，然后根据需要扩展或收缩。*/
```

点击联动效果实现

用户点击画布中的组件，页面左侧和右侧区域展现联动效果 → 记录点击的 id，然后存储在 Redux 中，形成联动效果。

- 初始加载数据时，设置 selectedId 为问卷信息列表的第一个组件的 id，即**默认选中**效果
- 用户每次点击组件时，设置 selectedId 为所选中的组件的 id，即**点击选中**效果
- 用户点击画布外的空白区域时，设置 selectedId 为 ''，表示清空选中，即**取消选中**效果

组件库的设计

组件分组

1. TS 类型

```ts
type GroupElementType = {
  groupId: string;
  groupName: string;
  components: FC[];
};
export const componentGroup: GroupElementType[] = [
  /* x */ ];
```

2. 分组信息

groupId	text	input	select
groupName	文本显示	用户输入	用户选择

显示到组件库

根据 `componentGroup` 简单遍历渲染即可，

```
componentGroup.map((group, gIndex) => {
    const { groupId, groupName, components } = group;
    return (
      <div key={groupId}>
        <Title
            level={5}
            style={{ marginTop: gIndex > 0 ? '20px' : '0px' }}>
          {groupName}
        </Title>
        {components.map((Component, cIndex) => (
          <div className={styles.wrapper} key={cIndex}>
            <div className={styles.component}>
              <Component />
            </div>
          </div>
        ))}
      </div>
    );
})
```

点击添加组件到画布

- 若未选中画布组件，点击组件库组件时，将其添加为画布的最后一个组件，并选中它。
- 若选中画布组件，点击组件库组件时，将其添加在选中组件下方，并选中它。

> 关于 id 的设计：对于组件，使用 `fe_id` 做唯一标识，而每个问卷，使用 `_id` 做唯一标识（与 MongoDB 配合）。

组件属性的设计

为每一个组件设计一个属性组件

- 通过 `useEffect` 实现 `props` 变化时，属性组件的重新渲染
- 通过 `onChange` + Redux 中用于更新组件数据的 `action` 实现表单数据更新时，将数据同步到 Redux，同时画布组件的重新渲染

  > 这里的 `action`，基于 `selectedId` 和接收到的 `payload`，在 `componentList` 中查找对应的组件信息，进一步更新其属性数据

点击画布组件时，显示对应的属性组件

通过 `useSelector` 获取 `components` 状态数据，根据 `selectedId` 查找到相应的组件信息，根据信息中的 `type` 和 `props` 字段，渲染出相应的属性组件。

没有画布组件被选中时，显示页面设置

工具栏设计

1. 相关功能：删除、隐藏/显示、锁定/解锁、复制/粘贴、上移/下移、撤销/重做
2. 快捷键：ahooks - useKeyPress
3. ...

拖拽排序

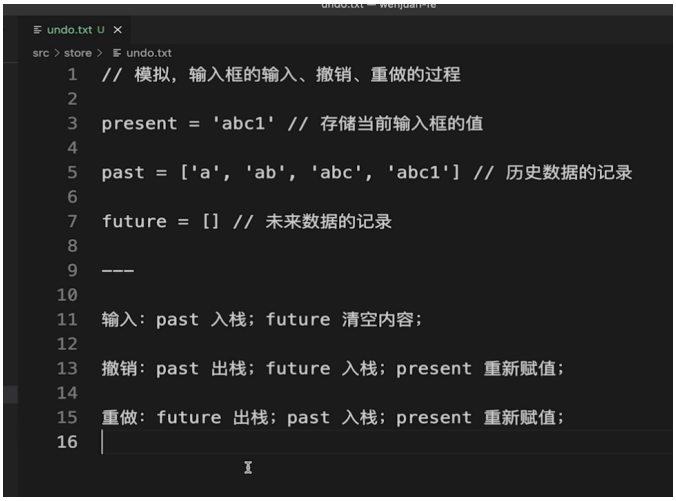
1. React 常见用于拖拽排序的库（这里选择 Dnd-kit）

- React-dnd
- React-sortable-hoc
- React-beautiful-dnd
- Dnd-kit
- Sortable.js

撤销重做

实现原理

- present - 保存当前数据
- past (undo-stack) - 历史数据列表
- future (redo-stack) - 未来数据列表



redux-undo 的简单使用

redux-toolkit 2.0 后内置 immer