

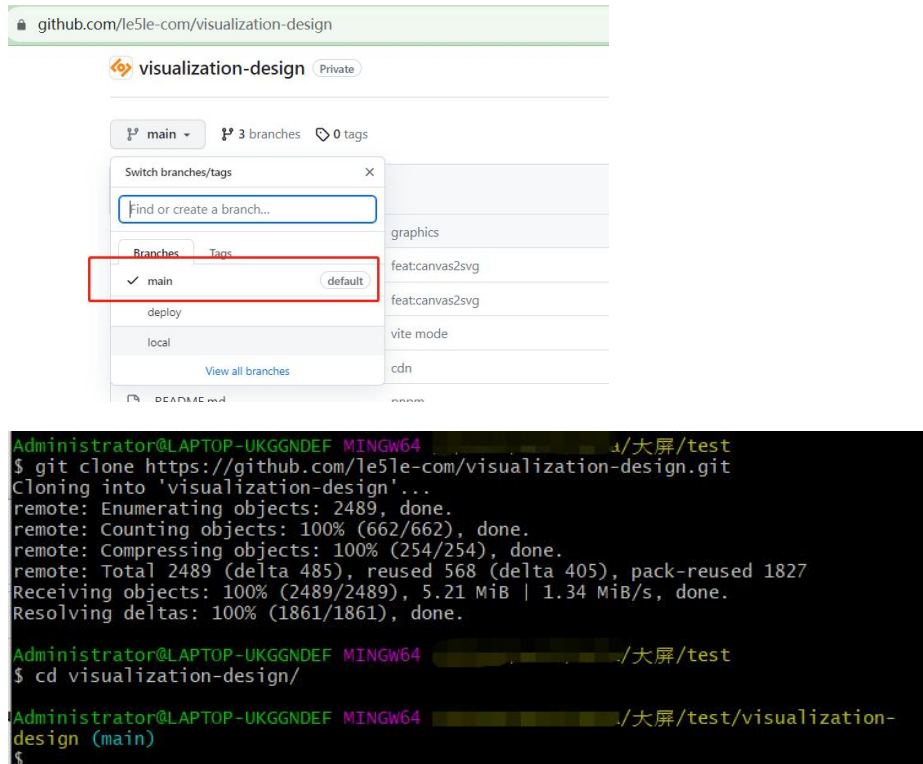
# visualization-design 源码使用手册

visualization-design 源码使用手册 .....	1
一 下载代码 .....	2
二 安装依赖包（快速运行） .....	2
三 加载图形库 .....	3
3.1 方案 .....	3
3.2 模版 .....	3
3.3 组件 .....	3
3.4 图表 .....	4
3.5 素材 .....	5
3.6 图元 .....	5
3.7 控件 .....	6
3.8 图形 .....	7
3.9 我的资源 .....	8
3.9.1 方案 .....	8
3.9.2 模版 .....	9
3.9.3 组件 .....	9
3.9.4 图片 .....	9
3.9.5 3D .....	10
四 编译打包 .....	10
五 源码结构说明 .....	10
5.1 编辑器页面 .....	11
5.1.1 头部菜单 .....	11
5.1.2 左侧组件库 .....	12
5.1.3 中间画布 .....	16
5.1.3 右侧属性面板 .....	20
5.1.3.1 不选中图元 .....	20
5.1.3.2 单选图元 .....	23
5.1.3.3 多选图元 .....	28
5.2 预览页面 .....	30
六 目录介绍 .....	31
6.1 Public 公共静态资源目录 .....	31
6.2 Src 开发目录 .....	31
6.3 其他 .....	31
七 运行流程 .....	31
八 代码中实现登录链接 .....	32
8.1 完全自己实现后端 .....	32
8.2 购买了乐吾乐后端 .....	32

## 一 下载代码

项目地址: <https://github.com/le5le-com/visualization-design>

1. 拉取代码: `git clone https://github.com/le5le-com/visualization-design.git`
2. 进入项目文件: `cd visualization-design/`
3. 确认当前是 **main 分支**:



说明:

- ① **main 分支**的核心库是通过引入最新发布的稳定的 **npm 包(meta2d 版本)**: **【推荐】**

[不推荐使用下面方式]

- ② **local 分支**引用的是非稳定状态下的源码包。需要拉取核心库并将其放到该项目的同级目录下, 核心库地址: <https://github.com/le5le-com/meta2d.js>  
(若使用 **local 分支**, 需全局搜索 **2d-components**, 并注释)

## 二 安装依赖包 (快速运行)

进入到 **visualization-design** 项目, 终端运行命令

`pnpm install` //安装依赖

`pnpm start` //启动项目

**pnpm** 下载地址: <https://pnpm.io/installation> (中文: <https://www.pnpm.cn/installation>)

## 三 加载图形库

### 3.1 方案

方案即图纸，通过/api/data/v/list 接口请求，通过 systemFlag=1 区分于非系统图纸。

```
const getCaseProjects = async (name: string, systemFlag = 1, current = 1, pageSize = 1000) => {
  const query: any = { tags: name };
  let collection = name == '系统组件' ? 'v.component' : 'v';
  const ret: any = await axios.post(
    `/api/data/${collection}/list`,
    {
      systemFlag
    },
    {
      params: {
        current,
        pageSize,
      },
    }
  );
  if (!ret) {
    return [];
  }
  for (const item of ret.list) { ...
  }
  return ret.list;
};
```

### 3.2 模版

模板同场景，通过 systemFlag=2 区分于非系统图纸。

乐吾乐提供场景和模板的后台管理平台，需要可以单独购买。

### 3.3 组件

通过/api/data/v.component/list 接口请求，通过 systemFlag=1 区分于非系统组件。

```
case '组件':
  if (activeAssets.value === 'system') {
    if (!componentCaches.length) {
      loading.value = true;
      componentCaches.push(...(await getCaseProjects('系统组件', 1)));
      loading.value = false;
      for (const component of componentCaches) {
        if (component.case) {
          let group = components.filter((item) => { item.name === component.case });
          if (group && group.length) {
            group[0].list.push(component);
          } else {
            components.push({
              name: component.case,
              list: [component]
            });
          }
        } else {
          components[0].list.push(component);
        }
      }
    }
  }
```

## 3.4 图表

图表主要包括 echarts 图表和乐吾乐图表

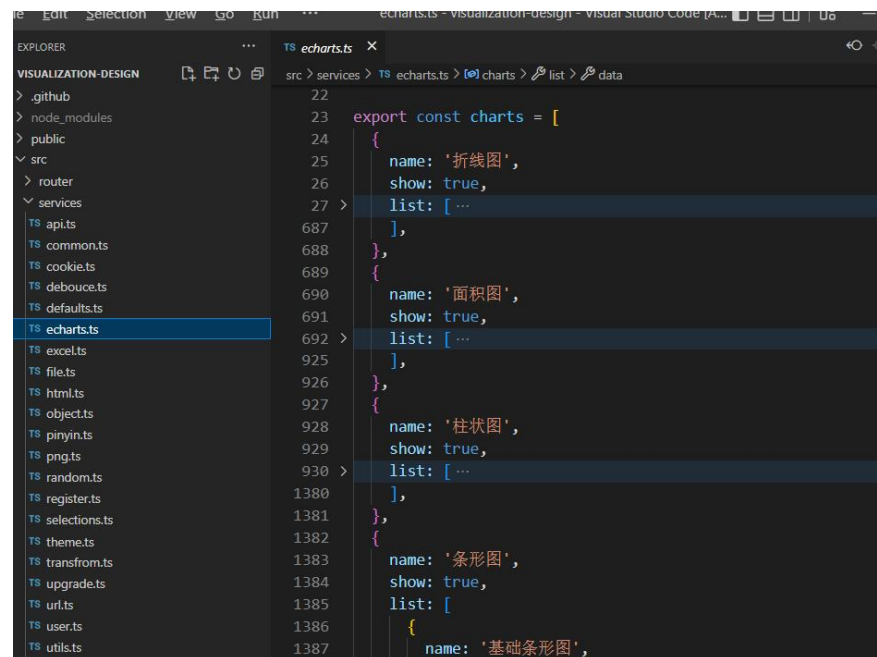
图形库使用对应的文档介绍：

Echarts 图表：

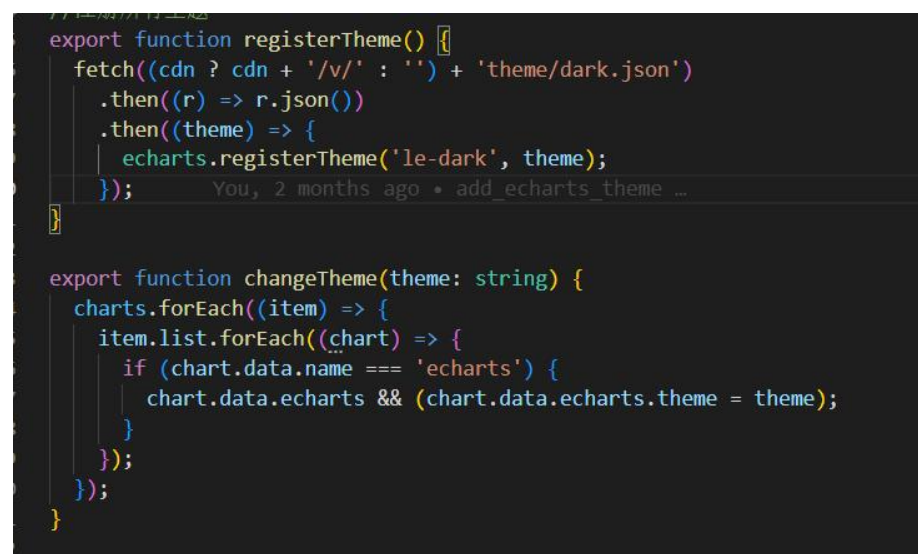
<https://doc.le5le.com/document/119754049#Echarts%E5%9B%BE%E8%A1%A8>

乐吾乐图表：<https://doc.le5le.com/document/119826189>

图表配置文件在 src/services/echarts.ts 文件里面。



这里有 echarts 图形库主题的注册及使用



## 3.5 素材

素材主要是图片资源，包括一些图片图标、装饰、标题和面板等。通过 `/api/assets/folders` 和 `/api/assets/files` 接口分别请求对应文件夹和文件夹下的文件。传入参数 `path:'v/material'`



```
case '素材':
  groupType.value = 1;
  if (!materials.length) {
    loading.value = true;
    materials.push(...(await getFolders('v/material')));
    loading.value = false;
  }
  subGroups.value = materials;
  break;
```

```
export async function getFolders(name: string, isSvg?: boolean) {
  const path = name;
  const folders: any = await axios.post('/api/assets/folders', {
    path,
  });
  if (!folders || !folders.list) {
    return [];
  }
  const files: any = await axios.post('/api/assets/files', {
    path,
  });
}
```

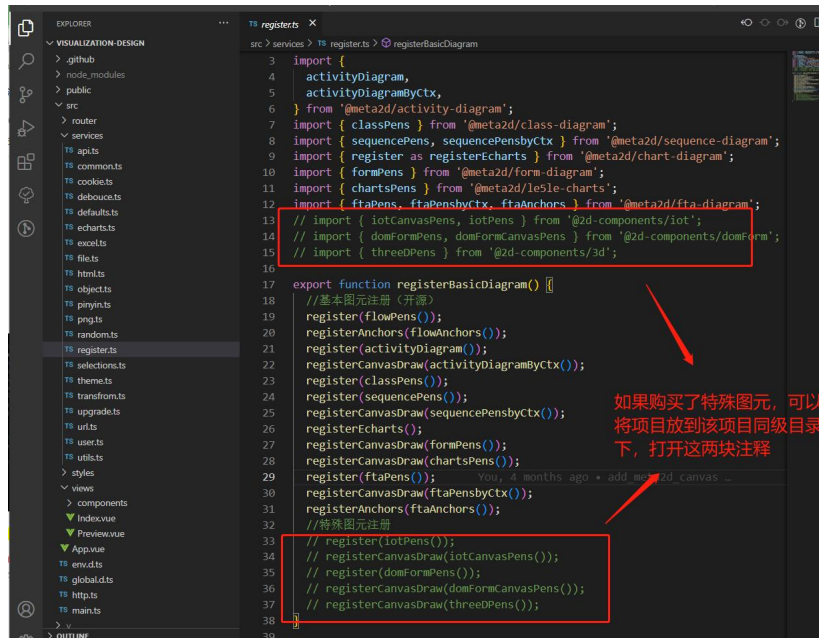
## 3.6 图元

图元主要是指不同行业/场景下的 `svg/png` 组件库，接口请求同素材，传入参数 `path` 分别是 `"svg"` / `"png"`

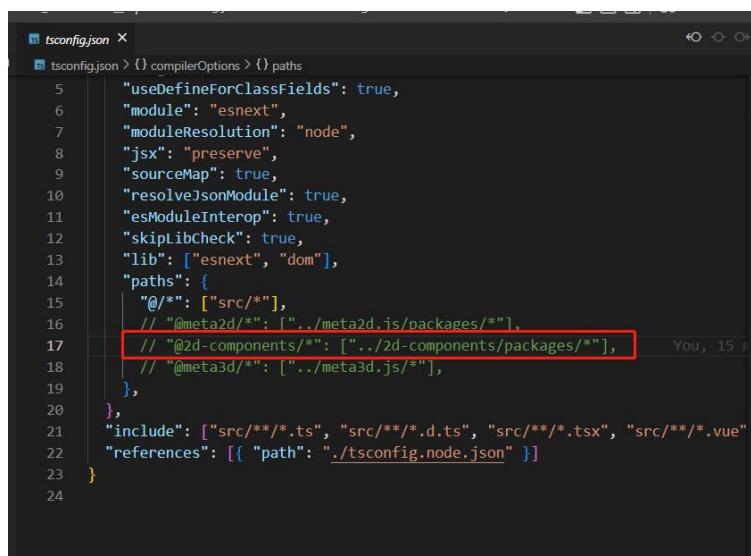
```
case '图元':
  if (!pngs.length) {
    loading.value = true;
    pngs.push(...(await getFolders('png')));
    pngs.push(...(await getFolders('svg', true)));
    loading.value = false;
  }
  subGroups.value = pngs;
  break;
```

## 3.7 控件

控件主要是指一些控件图元，包含基础图元和特殊图元（需要单独购买）。图形库属性的详细介绍可以查看文档：<https://doc.le5le.com/document/119754049>  
图元的注册是在 `src/services/register.ts` 文件里



如果购买了特殊图元，需要将特殊图元项目放到此项目同级目录下，并取消下面框选位置的注释：







```

9 export const shapes = [
10   {
11     name: '基本形状',
12     show: true,
13     list: [ ...
14   ],
15   },
16   {
17     name: '脑图',
18     show: true,
19     list: [ ...
20   ],
21   },
22   {
23     name: '流程图',
24     show: true,
25     list: [ ...
26   ],
27   },
28   {
29     name: '活动图',
30     show: true,
31     list: [
32       {
33         name: '活动图',
34         show: true,
35         list: [
36           {
37             name: '活动图',
38             show: true,
39             list: [
40               {
41                 name: '活动图',
42                 show: true,
43                 list: [
44                 ]
45               }
46             ]
47             }
48           ]
49         }
50       ]
51     }
52   ]
53 }

```

## 3.9 我的资源

### 3.9.1 方案

```

//用户方案
subGroups.value = await getCollectionImageList('方案', 'v',1);
groupType.value = 1;
userLastName = name;

```

1. /api/directory/list 获取文件夹列表
2. /api/data/v/list 获取所有图纸数据
3. 再将图纸数据放入对应到文件夹

```

const getCollectionImageList = async (name?: string, collection?: string, userFlag?: number) => {
  //1. 获取网盘文件夹
  const fullpath = `大屏/${name}`;
  let ret: { list: any[] } = await axios.post('/api/directory/list', {
    fullpath,
  });
  if (!ret) {
    // You, 4 months ago • feat:我的资源请求方式规范 ...
  }
  let list = [];
  for (let i of ret.list) { ...
  }
  const data = { ...
  };
  const config = { ...
  };
  //2. 请求所有图纸/组件数据
  const res: any = await getCollectionList(collection, data, config);
  //3. 将数据对应到网盘文件夹
  const results = [];
  const resultsMap = { ...
  };
  for (const item of list) { ...
  }
}

```



### 3.9.2 模版

```
subGroups.value = await getCollectionImageList('模板', 'v', 2);
groupType.value = 1;
userLastName = name;
```

同方案，通过 userFlag=1/2 区分是方案还是模版

### 3.9.3 组件

```
subGroups.value = await getCollectionImageList(
  '组件',
  'v.component',
  1
);
groupType.value = 1;
userLastName = name;
}
```

同方案, collection 对应 'v.component'。

### 3.9.4 图片

```
case '图片':
  loading.value = true;
  subGroups.value = await getImageList();
  loading.value = false;
  userLastName = name;
  break;
```

```
const getImageList = async () => {
  let ret: { list: any[] } = await axios.post('/api/directory/list', {
    fullpath: '/大屏/图片',
  });
  if (!ret) {
    return [];
  }
  let list = [];
  for (let i of ret.list) {
    if (i.fullpath.split('/').length === 4) {
      //不取当前文件夹
      list.push(i);
    }
  }
  return await Promise.all(...);
};
```

获取云盘下所有图片资源。

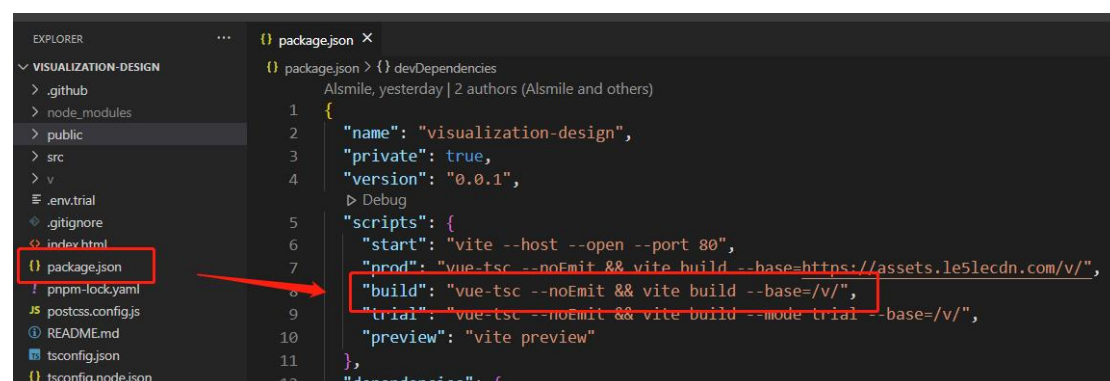
### 3.9.5 3D

自定义的 3d 场景，通过/api/data/3d/list 接口请求 3d 场景

```
case '3D':
  subGroups.value = [
    {
      name: '3D',
      list: [],
    },
  ];
  groupType.value = 1;
  await getPrivateGraphics();
  userLastName = name;
  break;
```

## 四 编译打包

运行命令：pnpm run build



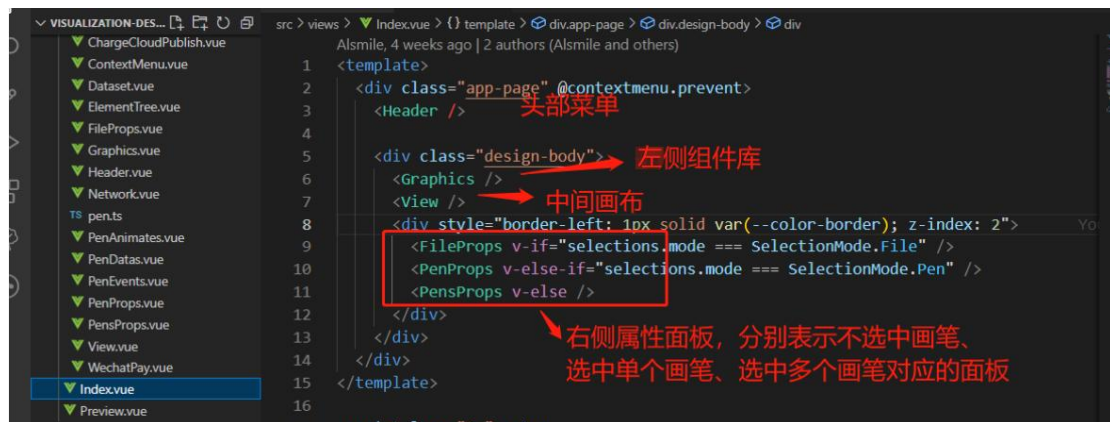
## 五 源码结构说明

整个项目分为两个页面：①Index.vue 大屏编辑页面 ②Preview.vue 大屏预览页面（运行页面）

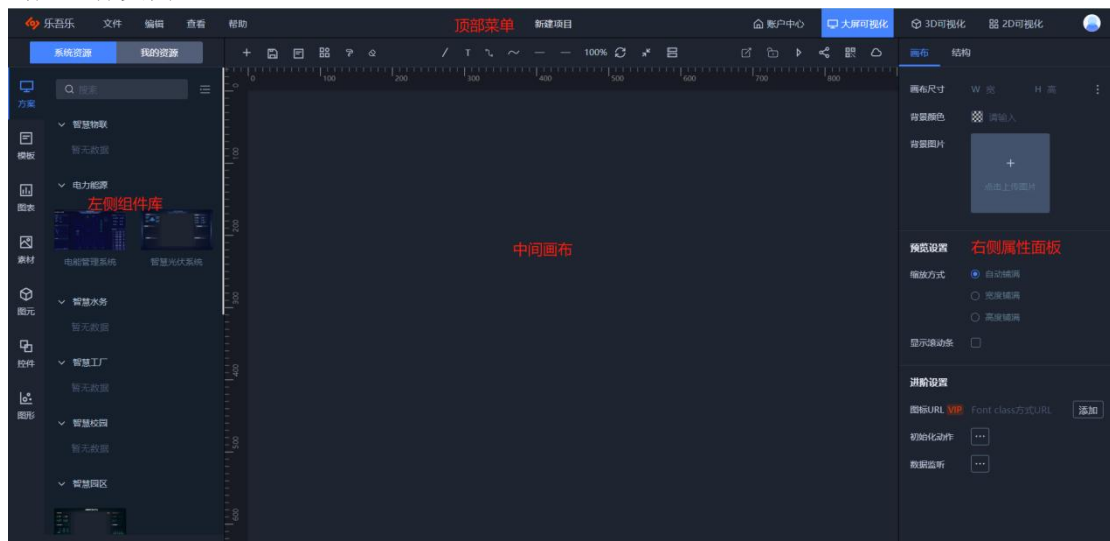
```
const routes = [
  { path: '/', component: () => import('@views/Index.vue') },
  { path: '/preview', component: () => import('@views/Preview.vue') },
];
```

## 5.1 编辑器页面

编辑器页面整体结构如下：



对应运行页面：



### 5.1.1 头部菜单

1. 目录：components/Header.vue
2. 源码说明：



- ① 右侧包含 logo 及公司名、文件、编辑、查看、帮助,都是通过下拉组件实现以文件为例,可以在每个选项的 click 事件定位到对应执行的代码内容。

```

<t-dropdown
  :minColumnWidth="200"
  :maxHeight="560"
  :delay2="[10, 150]"
  overlayClassName="header-dropdown"
  trigger1="click"
>
  <a> 文件 </a>
  <t-dropdown-menu>
    <t-dropdown-item @click="newFile">
      <a>新建文件</a>
    </t-dropdown-item>
    <t-dropdown-item @click="load(true)">
      <a>打开文件</a>
    </t-dropdown-item>
    <t-dropdown-item divider="true" @click="load">
      <a>导入文件</a>
    </t-dropdown-item>
    <t-dropdown-item>
      <a @click="save()">保存</a>
    </t-dropdown-item>
    <t-dropdown-item>
      <a @click="save(SaveType.SaveAs)">另保存</a>
    </t-dropdown-item>
    <t-dropdown-item divider="true">
      <a @click="downloadJson">下载JSON文件</a>
    </t-dropdown-item>
    <t-dropdown-item>

```

方法中可能用到了一些开源库（例如：下载文件用到了 `file-saver` 库）、调用了核心库方法，具体开发者自行阅读执行逻辑。

② 中间输入大屏图纸的名称

```

<div style= width: 148px; flex-shrink: 0 ></div>
<input v-model="data.name" @input="onInputName" />
Alsmile, 4 months ago • init
<a href= "assets/account" target= "blank">

```

③ 右侧是一些导航链接，包括账户中心、乐吾乐其他产品、登录/用户菜单

## 5.1.2 左侧组件库

1. 目录：components/Graphics.vue
2. 源码说明：

```

<div class="input-search">
  <div class="btn">
    <t-icon name="search" />
  </div>
  <t-input
    v-model="search"
    @change="onSearch"
    @enter="onSearch"
    placeholder="搜索"
  />

  <div class="ml-16">
    <t-tooltip content="展开/折叠">
      <t-icon
        name="menu-fold"
        class="hover"
        style="font-size: 16px"
        @click="onFold"
      />
    </t-tooltip>
  </div>
</div>

```

① 顶部的左侧搜索框用于从下面选中的组件中搜索内容，右侧按钮控制下面选中的组件整体的展开/折叠



搜索框执行代码如图：主要通过 visible 属性控制组件的显示/隐藏

```

<t-input
  v-model="search"
  @change="onSearch"
  @enter="onSearch"
  placeholder="搜索"
/>

```

```

5  const onSearch = () => {      Alsmile, last month • search ing
6  |   debounce(searchGraphics, 300);
7  | };
8
9  const searchGraphics = async () => {
10 |   if (search.value) {
11 |     activePanels[activeGroup.value].splice(
12 |       0,
13 |       activePanels[activeGroup.value].length
14 |     );
15 |   }
16
17 |   for (const group of subGroups.value) {
18 |     for (const item of group.list) {
19 |       if (search.value) {
20 |         item.visible = searchObjectPinyin(item, 'name', search.value);
21 |       } else {
22 |         item.visible = true;
23 |       }
24 |     }
25
26 |     if (search.value) {
27 |       activePanels[activeGroup.value].push(group.name);
28 |     }
29 |   }
30 | };
31

```

右侧按钮通过控制当前活动面板 activePanels key 的内容控制下面面板的展开/折叠。

```

<t-tooltip content="展开/折叠">
  <t-icon
    name="menu-fold"
    class="hover"
    style="font-size: 16px"
    @click="onFold"
  />
</t-tooltip>

```



```
const onFold = () => {
  if (!activePanels[activeGroup.value]) {
    return;
  }

  if (activePanels[activeGroup.value].length) {
    activePanels[activeGroup.value] = [];
  } else {
    activePanels[activeGroup.value] = [];
    for (const item of subGroups.value) {
      activePanels[activeGroup.value].push(item.name);
    }
  }
};
```

② 下面组件库，不同的类型对应不同的组件库/场景/模版内容。



通过监听点击选中，根据 name 去请求不同的数据，展示对应的内容。



```
const groupChange = async (name: string) => {
  activatedGroup.value = name;
  groupType.value = 0;
  switch (name) {
    > case '方案': ...
    > case '模板': ...
    > case '图表': ...
    > case '控件': ...
    > case '素材': ...
    > case '图元': ...
    > case '图形': Alsmile, 3 months ago • 场景
    > case '组件': ...
    > case '图片': ...
    > case '3D': ...
  }
}
```

### 5.1.3 中间画布

#### ① 顶部二级菜单



右侧是快捷按钮 新建、保存为大屏、保存为我的组件、格式刷和清除格式。

可以通过点击事件跳转到对应方法的执行，例如格式刷主要调用了核心库方法。

```
const oneFormat = () => {
  if (one.value) {
    one.value = false;
  } else {
    one.value = true;
    meta2d.setFormatPainter();
  }
  if (always.value) {
    always.value = false;
    one.value = false;
  }
};

const alwaysFormat = () => {
  always.value = true;
};

const clearFormat = () => {
  always.value = false;
  one.value = false;
  meta2d.clearFormatPainter();
};
```

中间是连线、视图、数据源管理相关操作

可以通过调用核心库方法将图纸切换到连线状态，下图代码表示控制关闭/打开连线状态

```

const oneDraw = () => {
  if (oneD.value) {
    oneD.value = false;
    if (!alwaysD.value) {
      meta2d.finishDrawLine();
      meta2d.drawLine();
      meta2d.store.options.disableAnchor = true;
    }
  } else {
    oneD.value = true;
    meta2d.drawLine(meta2d.store.options.drawingLineName);
    meta2d.store.options.disableAnchor = false;
  }
  if (alwaysD.value) {
    meta2d.finishDrawLine();
    meta2d.drawLine();
    oneD.value = false;
    alwaysD.value = false;
    meta2d.store.options.disableAnchor = true;
  }
};

```

通过修改 options 配置默认连线样式，下图代码表示修改连线类型。

```

const changeLineType = (value: string) => {
  currentLineType.value = value;
  if (meta2d) {
    meta2d.store.options.drawingLineName = value;
    meta2d.canvas.drawingLineName && (meta2d.canvas.drawingLineName
    meta2d.store.active?.forEach((pen) => {
      meta2d.updateLineType(pen, value);
    });
  }
};

```

通过调用核心库方法改变当前视图大小

```

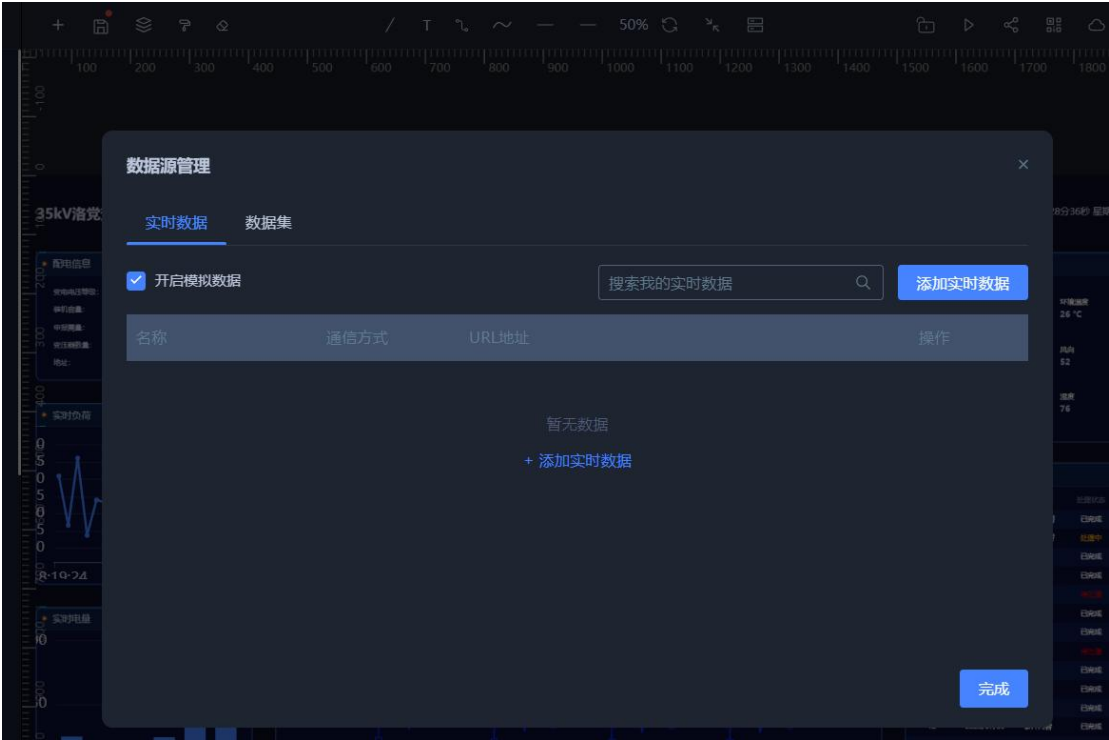
export const onScaleWindow = () => {
  // meta2d.fitView();
  meta2d.fitSizeView(true, 32);
};

export const onScaleFull = () => {
  meta2d.scale(1);
  // meta2d.centerView();
  const { x, y, origin, center } = meta2d.store.data;

  meta2d.translate(-x - origin.x, -y - origin.y);
  meta2d.translate(meta2d.store.options.x || 0, meta2d.store.
};

```

数据源管理主要包括实时数据和数据集管理



右侧包含锁定画布、运行、分享、云发布等。  
主要是项目业务内容，涉及到大屏的正式发布，需要结合部署人员一起探讨。

## ② 核心画布

```

<div id="meta2d"></div>

```

```

onMounted(() => {
  meta2d = new Meta2d('meta2d', meta2dOptions);
  registerBasicDiagram();
  open(true);
  meta2d.on('active', active);
  meta2d.on('inactive', inactive);
  meta2d.on('scale', scaleSubscriber);
  meta2d.on('add', lineAdd);
  meta2d.on('opened', openedListener);

  meta2d.on('undo', patchFlag);
  meta2d.on('redo', patchFlag);
  meta2d.on('add', patchFlag);
  meta2d.on('delete', patchFlag);
  meta2d.on('rotatePens', patchFlag);
  meta2d.on('translatePens', patchFlag);

  // 所有编辑栏所做修改
  meta2d.on('components-update-value', patchFlag);
  meta2d.on('contextmenu', onContextMenu);
  meta2d.on('click', canvasClick);

  timer = setInterval(autoSave, 60000);

  window.onbeforeunload = () => {
    autoSave();
  };
});

```

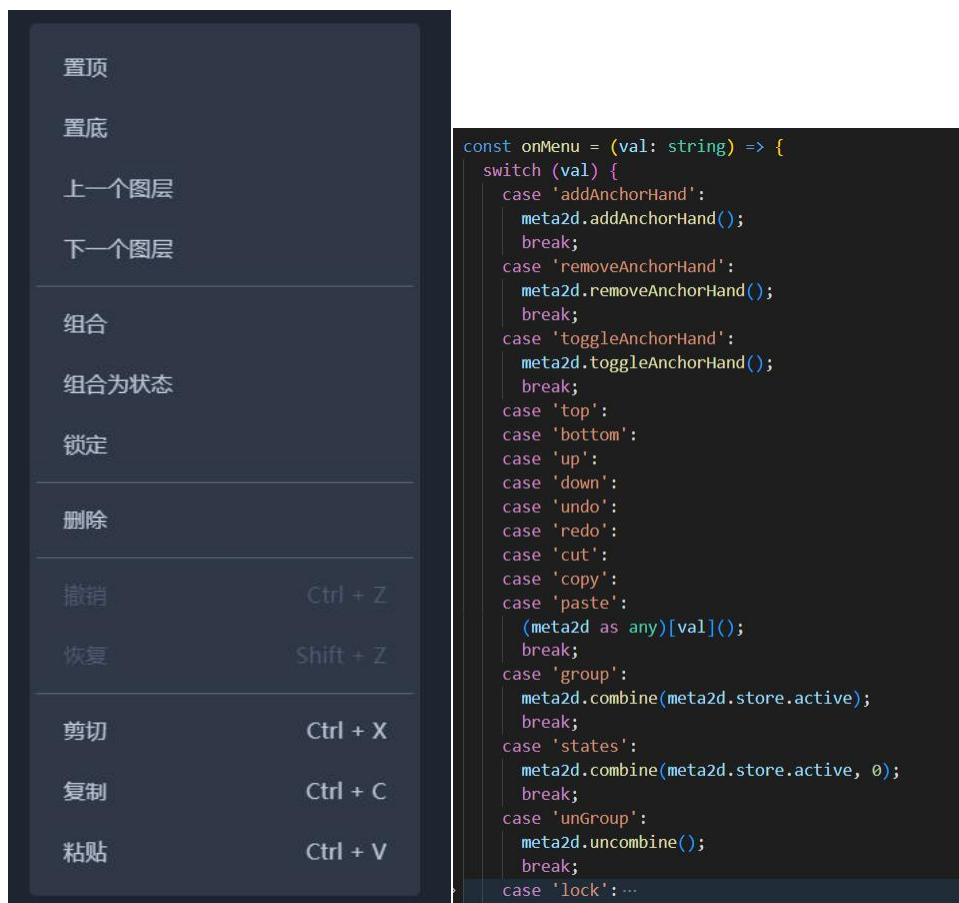
创建meta2d实例

注册图形库

监听画布消息

自动保存图纸

③ 右键菜单 (components/ ContextMenu.vue)



主要是通过调用核心库方法，具体方法说明可查看官方文档：

<https://doc.le5le.com/document/119882449#%E5%87%BD%E6%95%B0>

## 5.1.3 右侧属性面板

### 5.1.3.1 不选中图元

① 目录 components/FileProps.vue

② 源码说明：

#### 1. 画布板块

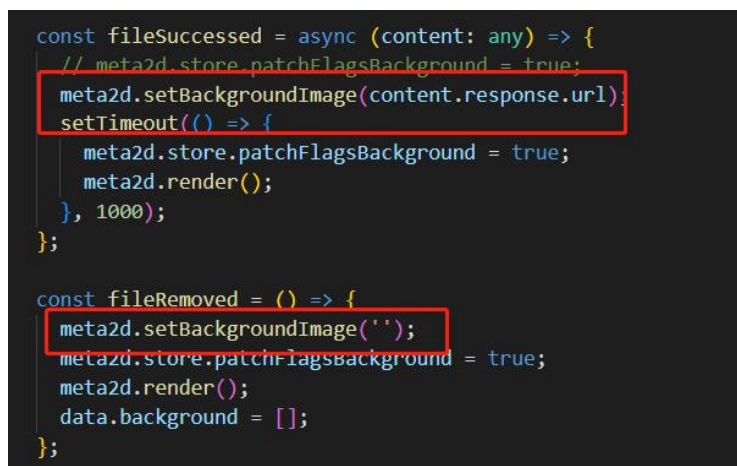
画布板块主要分为，基本设置（包括大屏尺寸、背景颜色和背景图片）、预览设置（设置预览时/运行时的缩放方式、是否显示滚动条）、进阶设置（配置图片库、初始化和数据监听）



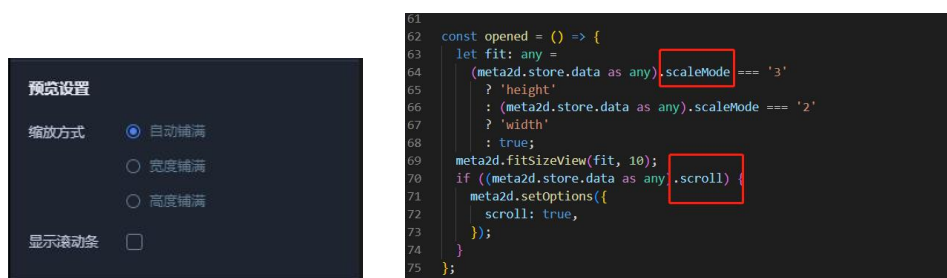


基本设置这一块，主要是通过修改 `meta2d.store.data` 下面的属性，具体说明可查看文档：  
<https://doc.le5le.com/document/119882449#%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84>

4  
 一些特殊属性：例如背景图片的设置，调用核心库 `setBackgroundImage` 方法。



预览设置这一块主要是配置预览时，大屏页面的显示方式，属于业务属性，使用可以查看 `Preview.vue` 页面。



进阶设置首先可以配置图标地址，输入字体图标地址后，会加载对应的图标库，在选中画笔进行图标设置时可以在图标抽屉中展示请求到的图标库。

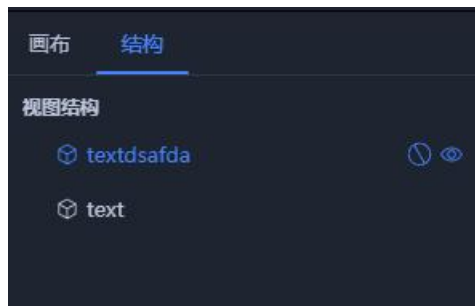


其次可以配置初始化动作，是指首次打开图纸后会执行的脚本。最后数据监听，即通信建立后获取数据的处理脚本，具体可查看文档：

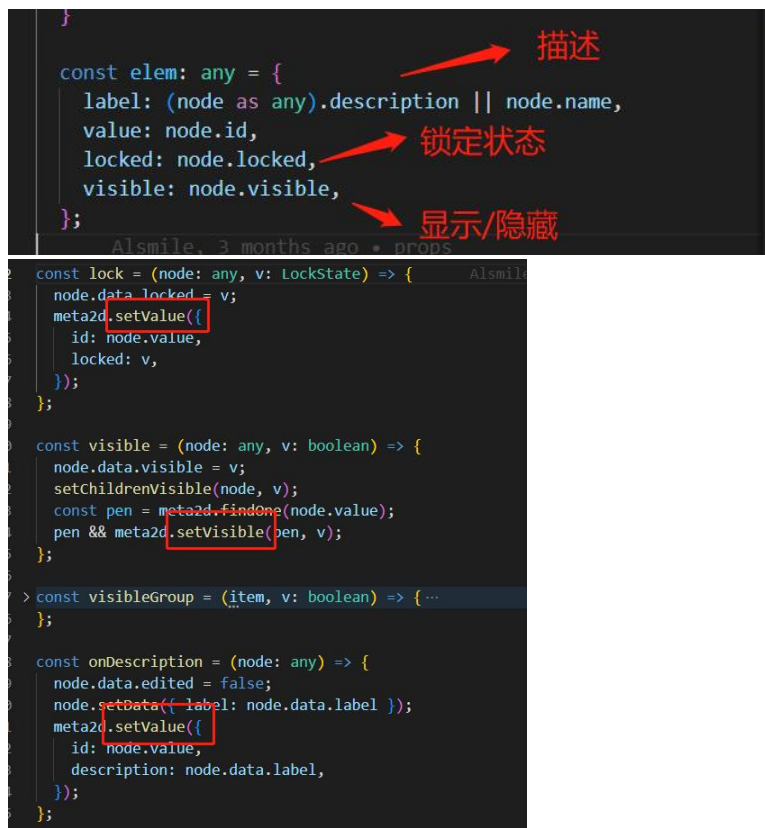
<https://doc.le5le.com/document/119620524#%E8%A7%A3%E6%9E%90%E8%87%AA%E5%AE%9A%E4%B9%89%E6%A0%BC%E5%BC%8F%E6%95%B0%E6%8D%AE>

## 2. 结构板块（components/ElementTree.vue）

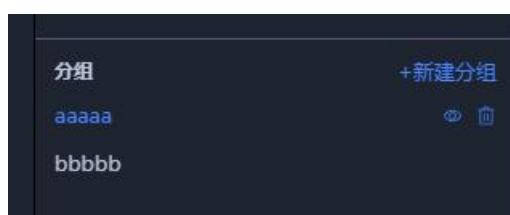
结构主要包括视图结构和分组，在视图结构可以设置画布中每个图元的描述名称、控制图元锁定状态和控制视图显示/隐藏，



主要是通过调用核心库的 setValue 和 setVisible 方法，对应的属性如下：



在分组中，可以新增/删除自定义分组名称，可以批量控制该分组对应图元的显示/隐藏。



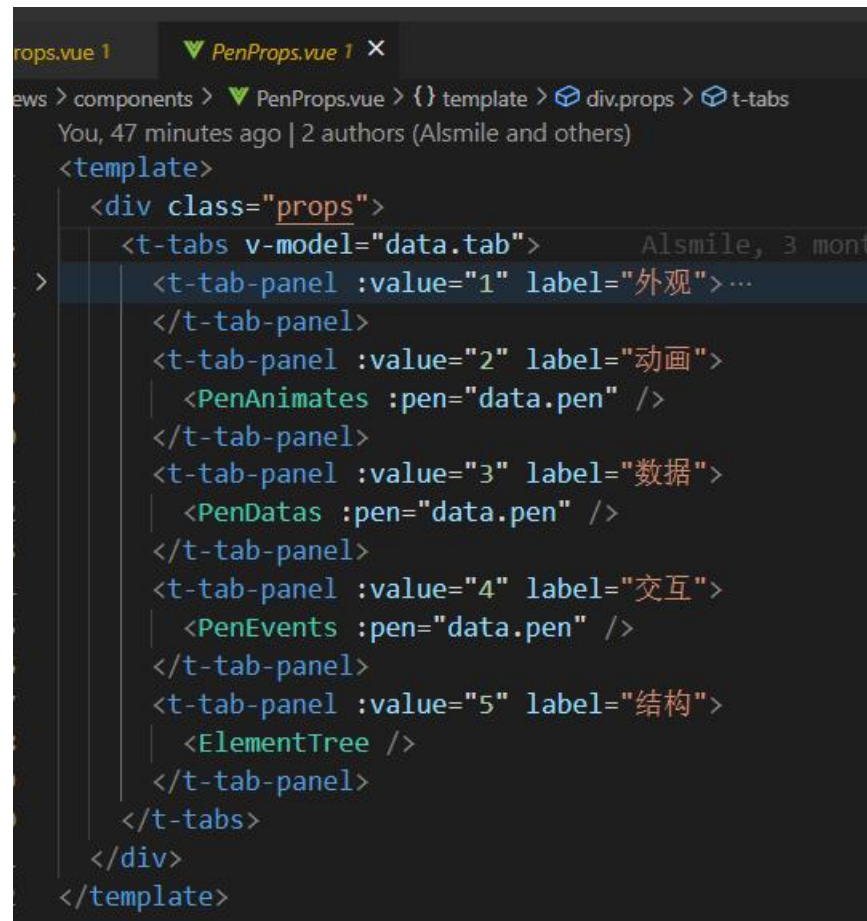


### 5.1.3.2 单选图元

① 目录 components/PenProps.vue

② 源码说明：

单选图元板块主要包括外观、动画、数据、交互和结构。



```
<template>
  <div class="props">
    <t-tabs v-model="data.tab">
      <t-tab-panel :value="1" label="外观">...
    </t-tab-panel>
    <t-tab-panel :value="2" label="动画">
      <PenAnimates :pen="data.pen" />
    </t-tab-panel>
    <t-tab-panel :value="3" label="数据">
      <PenDatas :pen="data.pen" />
    </t-tab-panel>
    <t-tab-panel :value="4" label="交互">
      <PenEvents :pen="data.pen" />
    </t-tab-panel>
    <t-tab-panel :value="5" label="结构">
      <ElementTree />
    </t-tab-panel>
    </t-tabs>
  </div>
</template>
```

#### 1. 外观

外观主要内容是设置图元的样式、包括 id 位置等基本设置、文字、图片/图标、自定义属性等。

外观

动画

数据

交互

结构

ID

116fa54

名称

text

分组

aaaaa

X

217.25

Y

77.25

0

W

160

H

30

圆角

不透明度

1

外观

前景颜色

悬停颜色

选中颜色

线条

1

末端样式

连接样式

背景

纯色

线性渐变

径向渐变

请输入

阴影

文字

属性

水平翻转

垂直翻转

锚点半径 4

禁止旋转

禁止缩放

禁用锚点

鼠标提示

```
1180 };
1181
1182 const changeValue = (prop: string) => {
1183   updatePen(data.pen, prop);
1184 };
1185
1186 const changeID = (value: any) => {
1187   if (!value) {
1188     initPenData();
1189     MessagePlugin.error('id 不能为空');
1190     return;
1191   }
1192   const oldID: string = data.pen.id;
1193   try {
1194     meta2d.changePenId(oldID, value);
1195     initPenData();
1196   } catch (error) {
1197     console.warn(error.message);
1198     MessagePlugin.error('id 修改失败, 请检查 id 是否');
1199     return;
1200   }
1201 };
1202
1203 const changeRectValue = (prop: string) => {
1204   data.rect.id = data.pen.id;
1205   data.rect.ratio = data.pen.ratio;
1206   updatePen(data.rect, prop);
1207 };
1208
1209 const onFontPopupVisible = (val: boolean) => {
1210   data.fontFamilyPopupVisible = val;
1211 };
1212
1213 const onFontFamily = (fontFamily: string) => {
1214   data.pen.fontFamily = fontFamily;
1215   data.fontFamilyPopupVisible = false;
1216   changeValue('fontFamily');
1217 };
1218
1219 export const updatePen = (pen: any, prop: string, render = true) => {
1220   const v: any = { id: pen.id };
1221   v[prop] = pen[prop];
1222   if (prop === 'width' && pen.ratio) {
1223     const rect = meta2d.findOne(pen.id);
1224     v.height = (pen.width / rect.width) * rect.height;
1225     pen.height = v.height;
1226   } else if (prop === 'height' && pen.ratio) {
1227     const rect = meta2d.findOne(pen.id);
1228     v.width = (pen.height / rect.height) * rect.width;
1229     pen.width = v.width;
1230   } else if (prop === 'shadow') {
1231     if (v[prop]) {
1232       lv.shadowOffsetX && (v.shadowOffsetX = 0);
1233       lv.shadowOffsetY && (v.shadowOffsetY = 0);
1234       lv.shadowBlur && (v.shadowBlur = 0);
1235     } else {
1236       v.shadowColor = '';
1237     }
1238   } else if (prop === 'lineGradientColors') {
1239     // @ts-ignore
1240     if (meta2d.store.active[0].name === 'line') {
1241       // @ts-ignore
1242       meta2d.store.active[0].calculative.gradientColorStop = null;
1243     } else {
1244       // @ts-ignore
1245       meta2d.store.active[0].calculative.lineGradient = null;
1246     }
1247     // 不同模式切换不同的系统配色
1248   } else if (prop === 'titleFnJs') {
1249     v.titleFn = null;
1250   } else if (prop === 'dash') {
1251     v.lineDash = lineDashObj[v[prop]];
1252   }
1253   meta2d.setValue(v, { render });
1254 };
1255
1256 export const updateRect = (rect: any, prop: string, render = true) => {
1257   const v: any = { id: rect.id };
1258   v[prop] = rect[prop];
1259   if (prop === 'width' && rect.ratio) {
1260     const pen = meta2d.findOne(rect.id);
1261     v.height = (rect.width / pen.width) * pen.height;
1262     rect.height = v.height;
1263   } else if (prop === 'height' && rect.ratio) {
1264     const pen = meta2d.findOne(rect.id);
1265     v.width = (rect.height / pen.height) * pen.width;
1266     rect.width = v.width;
1267   } else if (prop === 'shadow') {
1268     if (v[prop]) {
1269       lv.shadowOffsetX && (v.shadowOffsetX = 0);
1270       lv.shadowOffsetY && (v.shadowOffsetY = 0);
1271       lv.shadowBlur && (v.shadowBlur = 0);
1272     } else {
1273       v.shadowColor = '';
1274     }
1275   } else if (prop === 'lineGradientColors') {
1276     // @ts-ignore
1277     if (meta2d.store.active[0].name === 'line') {
1278       // @ts-ignore
1279       meta2d.store.active[0].calculative.gradientColorStop = null;
1280     } else {
1281       // @ts-ignore
1282       meta2d.store.active[0].calculative.lineGradient = null;
1283     }
1284     // 不同模式切换不同的系统配色
1285   } else if (prop === 'titleFnJs') {
1286     v.titleFn = null;
1287   } else if (prop === 'dash') {
1288     v.lineDash = lineDashObj[v[prop]];
1289   }
1290   meta2d.setValue(v, { render });
1291 };
1292
1293 export const updateText = (text: any, prop: string, render = true) => {
1294   const v: any = { id: text.id };
1295   v[prop] = text[prop];
1296   if (prop === 'fontSize') {
1297     v.fontSize = text.fontSize;
1298   } else if (prop === 'fontWeight') {
1299     v.fontWeight = text.fontWeight;
1300   } else if (prop === 'fontStyle') {
1301     v.fontStyle = text.fontStyle;
1302   } else if (prop === 'fontFamily') {
1303     v.fontFamily = text.fontFamily;
1304   } else if (prop === 'textColor') {
1305     v.textColor = text.textColor;
1306   } else if (prop === 'textAlign') {
1307     v.textAlign = text.textAlign;
1308   } else if (prop === 'textBaseline') {
1309     v.textBaseline = text.textBaseline;
1310   } else if (prop === 'textDirection') {
1311     v.textDirection = text.textDirection;
1312   } else if (prop === 'textOverflow') {
1313     v.textOverflow = text.textOverflow;
1314   } else if (prop === 'textWrap') {
1315     v.textWrap = text.textWrap;
1316   } else if (prop === 'textLineHeight') {
1317     v.textLineHeight = text.textLineHeight;
1318   } else if (prop === 'textLetterSpacing') {
1319     v.textLetterSpacing = text.textLetterSpacing;
1320   } else if (prop === 'textWordSpacing') {
1321     v.textWordSpacing = text.textWordSpacing;
1322   } else if (prop === 'textFontWeight') {
1323     v.textFontWeight = text.textFontWeight;
1324   } else if (prop === 'textFontStyle') {
1325     v.textFontStyle = text.textFontStyle;
1326   } else if (prop === 'textFontFamily') {
1327     v.textFontFamily = text.textFontFamily;
1328   } else if (prop === 'textColor') {
1329     v.textColor = text.textColor;
1330   } else if (prop === 'textAlign') {
1331     v.textAlign = text.textAlign;
1332   } else if (prop === 'textBaseline') {
1333     v.textBaseline = text.textBaseline;
1334   } else if (prop === 'textDirection') {
1335     v.textDirection = text.textDirection;
1336   } else if (prop === 'textOverflow') {
1337     v.textOverflow = text.textOverflow;
1338   } else if (prop === 'textWrap') {
1339     v.textWrap = text.textWrap;
1340   } else if (prop === 'textLineHeight') {
1341     v.textLineHeight = text.textLineHeight;
1342   } else if (prop === 'textLetterSpacing') {
1343     v.textLetterSpacing = text.textLetterSpacing;
1344   } else if (prop === 'textWordSpacing') {
1345     v.textWordSpacing = text.textWordSpacing;
1346   } else if (prop === 'textFontWeight') {
1347     v.textFontWeight = text.textFontWeight;
1348   } else if (prop === 'textFontStyle') {
1349     v.textFontStyle = text.textFontStyle;
1350   } else if (prop === 'textFontFamily') {
1351     v.textFontFamily = text.textFontFamily;
1352   }
1353   meta2d.setValue(v, { render });
1354 };
1355
1356 export const updateImage = (image: any, prop: string, render = true) => {
1357   const v: any = { id: image.id };
1358   v[prop] = image[prop];
1359   if (prop === 'width') {
1360     v.width = image.width;
1361   } else if (prop === 'height') {
1362     v.height = image.height;
1363   } else if (prop === 'x') {
1364     v.x = image.x;
1365   } else if (prop === 'y') {
1366     v.y = image.y;
1367   } else if (prop === 'angle') {
1368     v.angle = image.angle;
1369   } else if (prop === 'opacity') {
1370     v.opacity = image.opacity;
1371   } else if (prop === 'filter') {
1372     v.filter = image.filter;
1373   } else if (prop === 'src') {
1374     v.src = image.src;
1375   } else if (prop === 'alt') {
1376     v.alt = image.alt;
1377   } else if (prop === 'title') {
1378     v.title = image.title;
1379   } else if (prop === 'description') {
1380     v.description = image.description;
1381   } else if (prop === 'copyright') {
1382     v.copyright = image.copyright;
1383   } else if (prop === 'license') {
1384     v.license = image.license;
1385   } else if (prop === 'author') {
1386     v.author = image.author;
1387   } else if (prop === 'created') {
1388     v.created = image.created;
1389   } else if (prop === 'updated') {
1390     v.updated = image.updated;
1391   } else if (prop === 'deleted') {
1392     v.deleted = image.deleted;
1393   } else if (prop === 'status') {
1394     v.status = image.status;
1395   } else if (prop === 'category') {
1396     v.category = image.category;
1397   } else if (prop === 'tags') {
1398     v.tags = image.tags;
1399   } else if (prop === 'keywords') {
1400     v.keywords = image.keywords;
1401   } else if (prop === 'description') {
1402     v.description = image.description;
1403   } else if (prop === 'copyright') {
1404     v.copyright = image.copyright;
1405   } else if (prop === 'license') {
1406     v.license = image.license;
1407   } else if (prop === 'author') {
1408     v.author = image.author;
1409   } else if (prop === 'created') {
1410     v.created = image.created;
1411   } else if (prop === 'updated') {
1412     v.updated = image.updated;
1413   } else if (prop === 'deleted') {
1414     v.deleted = image.deleted;
1415   } else if (prop === 'status') {
1416     v.status = image.status;
1417   } else if (prop === 'category') {
1418     v.category = image.category;
1419   } else if (prop === 'tags') {
1420     v.tags = image.tags;
1421   } else if (prop === 'keywords') {
1422     v.keywords = image.keywords;
1423   }
1424   meta2d.setValue(v, { render });
1425 };
1426
1427 export const updateGroup = (group: any, prop: string, render = true) => {
1428   const v: any = { id: group.id };
1429   v[prop] = group[prop];
1430   if (prop === 'width') {
1431     v.width = group.width;
1432   } else if (prop === 'height') {
1433     v.height = group.height;
1434   } else if (prop === 'x') {
1435     v.x = group.x;
1436   } else if (prop === 'y') {
1437     v.y = group.y;
1438   } else if (prop === 'angle') {
1439     v.angle = group.angle;
1440   } else if (prop === 'opacity') {
1441     v.opacity = group.opacity;
1442   } else if (prop === 'filter') {
1443     v.filter = group.filter;
1444   } else if (prop === 'src') {
1445     v.src = group.src;
1446   } else if (prop === 'alt') {
1447     v.alt = group.alt;
1448   } else if (prop === 'title') {
1449     v.title = group.title;
1450   } else if (prop === 'description') {
1451     v.description = group.description;
1452   } else if (prop === 'copyright') {
1453     v.copyright = group.copyright;
1454   } else if (prop === 'license') {
1455     v.license = group.license;
1456   } else if (prop === 'author') {
1457     v.author = group.author;
1458   } else if (prop === 'created') {
1459     v.created = group.created;
1460   } else if (prop === 'updated') {
1461     v.updated = group.updated;
1462   } else if (prop === 'deleted') {
1463     v.deleted = group.deleted;
1464   } else if (prop === 'status') {
1465     v.status = group.status;
1466   } else if (prop === 'category') {
1467     v.category = group.category;
1468   } else if (prop === 'tags') {
1469     v.tags = group.tags;
1470   } else if (prop === 'keywords') {
1471     v.keywords = group.keywords;
1472   }
1473   meta2d.setValue(v, { render });
1474 };
1475
1476 export const updateLayer = (layer: any, prop: string, render = true) => {
1477   const v: any = { id: layer.id };
1478   v[prop] = layer[prop];
1479   if (prop === 'width') {
1480     v.width = layer.width;
1481   } else if (prop === 'height') {
1482     v.height = layer.height;
1483   } else if (prop === 'x') {
1484     v.x = layer.x;
1485   } else if (prop === 'y') {
1486     v.y = layer.y;
1487   } else if (prop === 'angle') {
1488     v.angle = layer.angle;
1489   } else if (prop === 'opacity') {
1490     v.opacity = layer.opacity;
1491   } else if (prop === 'filter') {
1492     v.filter = layer.filter;
1493   } else if (prop === 'src') {
1494     v.src = layer.src;
1495   } else if (prop === 'alt') {
1496     v.alt = layer.alt;
1497   } else if (prop === 'title') {
1498     v.title = layer.title;
1499   } else if (prop === 'description') {
1500     v.description = layer.description;
1501   } else if (prop === 'copyright') {
1502     v.copyright = layer.copyright;
1503   } else if (prop === 'license') {
1504     v.license = layer.license;
1505   } else if (prop === 'author') {
1506     v.author = layer.author;
1507   } else if (prop === 'created') {
1508     v.created = layer.created;
1509   } else if (prop === 'updated') {
1510     v.updated = layer.updated;
1511   } else if (prop === 'deleted') {
1512     v.deleted = layer.deleted;
1513   } else if (prop === 'status') {
1514     v.status = layer.status;
1515   } else if (prop === 'category') {
1516     v.category = layer.category;
1517   } else if (prop === 'tags') {
1518     v.tags = layer.tags;
1519   } else if (prop === 'keywords') {
1520     v.keywords = layer.keywords;
1521   }
1522   meta2d.setValue(v, { render });
1523 };
1524
1525 export const updateCanvas = (canvas: any, prop: string, render = true) => {
1526   const v: any = { id: canvas.id };
1527   v[prop] = canvas[prop];
1528   if (prop === 'width') {
1529     v.width = canvas.width;
1530   } else if (prop === 'height') {
1531     v.height = canvas.height;
1532   } else if (prop === 'x') {
1533     v.x = canvas.x;
1534   } else if (prop === 'y') {
1535     v.y = canvas.y;
1536   } else if (prop === 'angle') {
1537     v.angle = canvas.angle;
1538   } else if (prop === 'opacity') {
1539     v.opacity = canvas.opacity;
1540   } else if (prop === 'filter') {
1541     v.filter = canvas.filter;
1542   } else if (prop === 'src') {
1543     v.src = canvas.src;
1544   } else if (prop === 'alt') {
1545     v.alt = canvas.alt;
1546   } else if (prop === 'title') {
1547     v.title = canvas.title;
1548   } else if (prop === 'description') {
1549     v.description = canvas.description;
1550   } else if (prop === 'copyright') {
1551     v.copyright = canvas.copyright;
1552   } else if (prop === 'license') {
1553     v.license = canvas.license;
1554   } else if (prop === 'author') {
1555     v.author = canvas.author;
1556   } else if (prop === 'created') {
1557     v.created = canvas.created;
1558   } else if (prop === 'updated') {
1559     v.updated = canvas.updated;
1560   } else if (prop === 'deleted') {
1561     v.deleted = canvas.deleted;
1562   } else if (prop === 'status') {
1563     v.status = canvas.status;
1564   } else if (prop === 'category') {
1565     v.category = canvas.category;
1566   } else if (prop === 'tags') {
1567     v.tags = canvas.tags;
1568   } else if (prop === 'keywords') {
1569     v.keywords = canvas.keywords;
1570   }
1571   meta2d.setValue(v, { render });
1572 };
1573
1574 export const updatePage = (page: any, prop: string, render = true) => {
1575   const v: any = { id: page.id };
1576   v[prop] = page[prop];
1577   if (prop === 'width') {
1578     v.width = page.width;
1579   } else if (prop === 'height') {
1580     v.height = page.height;
1581   } else if (prop === 'x') {
1582     v.x = page.x;
1583   } else if (prop === 'y') {
1584     v.y = page.y;
1585   } else if (prop === 'angle') {
1586     v.angle = page.angle;
1587   } else if (prop === 'opacity') {
1588     v.opacity = page.opacity;
1589   } else if (prop === 'filter') {
1590     v.filter = page.filter;
1591   } else if (prop === 'src') {
1592     v.src = page.src;
1593   } else if (prop === 'alt') {
1594     v.alt = page.alt;
1595   } else if (prop === 'title') {
1596     v.title = page.title;
1597   } else if (prop === 'description') {
1598     v.description = page.description;
1599   } else if (prop === 'copyright') {
1600     v.copyright = page.copyright;
1601   } else if (prop === 'license') {
1602     v.license = page.license;
1603   } else if (prop === 'author') {
1604     v.author = page.author;
1605   } else if (prop === 'created') {
1606     v.created = page.created;
1607   } else if (prop === 'updated') {
1608     v.updated = page.updated;
1609   } else if (prop === 'deleted') {
1610     v.deleted = page.deleted;
1611   } else if (prop === 'status') {
1612     v.status = page.status;
1613   } else if (prop === 'category') {
1614     v.category = page.category;
1615   } else if (prop === 'tags') {
1616     v.tags = page.tags;
1617   } else if (prop === 'keywords') {
1618     v.keywords = page.keywords;
1619   }
1620   meta2d.setValue(v, { render });
1621 };
1622
1623 export const updatePageGroup = (pageGroup: any, prop: string, render = true) => {
1624   const v: any = { id: pageGroup.id };
1625   v[prop] = pageGroup[prop];
1626   if (prop === 'width') {
1627     v.width = pageGroup.width;
1628   } else if (prop === 'height') {
1629     v.height = pageGroup.height;
1630   } else if (prop === 'x') {
1631     v.x = pageGroup.x;
1632   } else if (prop === 'y') {
1633     v.y = pageGroup.y;
1634   } else if (prop === 'angle') {
1635     v.angle = pageGroup.angle;
1636   } else if (prop === 'opacity') {
1637     v.opacity = pageGroup.opacity;
1638   } else if (prop === 'filter') {
1639     v.filter = pageGroup.filter;
1640   } else if (prop === 'src') {
1641     v.src = pageGroup.src;
1642   } else if (prop === 'alt') {
1643     v.alt = pageGroup.alt;
1644   } else if (prop === 'title') {
1645     v.title = pageGroup.title;
1646   } else if (prop === 'description') {
1647     v.description = pageGroup.description;
1648   } else if (prop === 'copyright') {
1649     v.copyright = pageGroup.copyright;
1650   } else if (prop === 'license') {
1651     v.license = pageGroup.license;
1652   } else if (prop === 'author') {
1653     v.author = pageGroup.author;
1654   } else if (prop === 'created') {
1655     v.created = pageGroup.created;
1656   } else if (prop === 'updated') {
1657     v.updated = pageGroup.updated;
1658   } else if (prop === 'deleted') {
1659     v.deleted = pageGroup.deleted;
1660   } else if (prop === 'status') {
1661     v.status = pageGroup.status;
1662   } else if (prop === 'category') {
1663     v.category = pageGroup.category;
1664   } else if (prop === 'tags') {
1665     v.tags = pageGroup.tags;
1666   } else if (prop === 'keywords') {
1667     v.keywords = pageGroup.keywords;
1668   }
1669   meta2d.setValue(v, { render });
1670 };
1671
1672 export const updatePageLayer = (pageLayer: any, prop: string, render = true) => {
1673   const v: any = { id: pageLayer.id };
1674   v[prop] = pageLayer[prop];
1675   if (prop === 'width') {
1676     v.width = pageLayer.width;
1677   } else if (prop === 'height') {
1678     v.height = pageLayer.height;
1679   } else if (prop === 'x') {
1680     v.x = pageLayer.x;
1681   } else if (prop === 'y') {
1682     v.y = pageLayer.y;
1683   } else if (prop === 'angle') {
1684     v.angle = pageLayer.angle;
1685   } else if (prop === 'opacity') {
1686     v.opacity = pageLayer.opacity;
1687   } else if (prop === 'filter') {
1688     v.filter = pageLayer.filter;
1689   } else if (prop === 'src') {
1690     v.src = pageLayer.src;
1691   } else if (prop === 'alt') {
1692     v.alt = pageLayer.alt;
1693   } else if (prop === 'title') {
1694     v.title = pageLayer.title;
1695   } else if (prop === 'description') {
1696     v.description = pageLayer.description;
1697   } else if (prop === 'copyright') {
1698     v.copyright = pageLayer.copyright;
1699   } else if (prop === 'license') {
1700     v.license = pageLayer.license;
1701   } else if (prop === 'author') {
1702     v.author = pageLayer.author;
1703   } else if (prop === 'created') {
1704     v.created = pageLayer.created;
1705   } else if (prop === 'updated') {
1706     v.updated = pageLayer.updated;
1707   } else if (prop === 'deleted') {
1708     v.deleted = pageLayer.deleted;
1709   } else if (prop === 'status') {
1710     v.status = pageLayer.status;
1711   } else if (prop === 'category') {
1712     v.category = pageLayer.category;
1713   } else if (prop === 'tags') {
1714     v.tags = pageLayer.tags;
1715   } else if (prop === 'keywords') {
1716     v.keywords = pageLayer.keywords;
1717   }
1718   meta2d.setValue(v, { render });
1719 };
1720
1721 export const updatePageCanvas = (pageCanvas: any, prop: string, render = true) => {
1722   const v: any = { id: pageCanvas.id };
1723   v[prop] = pageCanvas[prop];
1724   if (prop === 'width') {
1725     v.width = pageCanvas.width;
1726   } else if (prop === 'height') {
1727     v.height = pageCanvas.height;
1728   } else if (prop === 'x') {
1729     v.x = pageCanvas.x;
1730   } else if (prop === 'y') {
1731     v.y = pageCanvas.y;
1732   } else if (prop === 'angle') {
1733     v.angle = pageCanvas.angle;
1734   } else if (prop === 'opacity') {
1735     v.opacity = pageCanvas.opacity;
1736   } else if (prop === 'filter') {
1737     v.filter = pageCanvas.filter;
1738   } else if (prop === 'src') {
1739     v.src = pageCanvas.src;
1740   } else if (prop === 'alt') {
1741     v.alt = pageCanvas.alt;
1742   } else if (prop === 'title') {
1743     v.title = pageCanvas.title;
1744   } else if (prop === 'description') {
1745     v.description = pageCanvas.description;
1746   } else if (prop === 'copyright') {
1747     v.copyright = pageCanvas.copyright;
1748   } else if (prop === 'license') {
1749     v.license = pageCanvas.license;
1750   } else if (prop === 'author') {
1751     v.author = pageCanvas.author;
1752   } else if (prop === 'created') {
1753     v.created = pageCanvas.created;
1754   } else if (prop === 'updated') {
1755     v.updated = pageCanvas.updated;
1756   } else if (prop === 'deleted') {
1757     v.deleted = pageCanvas.deleted;
1758   } else if (prop === 'status') {
1759     v.status = pageCanvas.status;
1760   } else if (prop === 'category') {
1761     v.category = pageCanvas.category;
1762   } else if (prop === 'tags') {
1763     v.tags = pageCanvas.tags;
1764   } else if (prop === 'keywords') {
1765     v.keywords = pageCanvas.keywords;
1766   }
1767   meta2d.setValue(v, { render });
1768 };
1769
1770 export const updatePagePageGroup = (pagePageGroup: any, prop: string, render = true) => {
1771   const v: any = { id: pagePageGroup.id };
1772   v[prop] = pagePageGroup[prop];
1773   if (prop === 'width') {
1774     v.width = pagePageGroup.width;
1775   } else if (prop === 'height') {
1776     v.height = pagePageGroup.height;
1777   } else if (prop === 'x') {
1778     v.x = pagePageGroup.x;
1779   } else if (prop === 'y') {
1780     v.y = pagePageGroup.y;
1781   } else if (prop === 'angle') {
1782     v.angle = pagePageGroup.angle;
1783   } else if (prop === 'opacity') {
1784     v.opacity = pagePageGroup.opacity;
1785   } else if (prop === 'filter') {
1786     v.filter = pagePageGroup.filter;
1787   } else if (prop === 'src') {
1788     v.src = pagePageGroup.src;
1789   } else if (prop === 'alt') {
1790     v.alt = pagePageGroup.alt;
1791   } else if (prop === 'title') {
1792     v.title = pagePageGroup.title;
1793   } else if (prop === 'description') {
1794     v.description = pagePageGroup.description;
1795   } else if (prop === 'copyright') {
1796     v.copyright = pagePageGroup.copyright;
1797   } else if (prop === 'license') {
1798     v.license = pagePageGroup.license;
1799   } else if (prop === 'author') {
1800     v.author = pagePageGroup.author;
1801   } else if (prop === 'created') {
1802     v.created = pagePageGroup.created;
1803   } else if (prop === 'updated') {
1804     v.updated = pagePageGroup.updated;
1805   } else if (prop === 'deleted') {
1806     v.deleted = pagePageGroup.deleted;
1807   } else if (prop === 'status') {
1808     v.status = pagePageGroup.status;
1809   } else if (prop === 'category') {
1810     v.category = pagePageGroup.category;
1811   } else if (prop === 'tags') {
1812     v.tags = pagePageGroup.tags;
1813   } else if (prop === 'keywords') {
1814     v.keywords = pagePageGroup.keywords;
1815   }
1816   meta2d.setValue(v, { render });
1817 };
1818
1819 export const updatePagePageLayer = (pagePageLayer: any, prop: string, render = true) => {
1820   const v: any = { id: pagePageLayer.id };
1821   v[prop] = pagePageLayer[prop];
1822   if (prop === 'width') {
1823     v.width = pagePageLayer.width;
1824   } else if (prop === 'height') {
1825     v.height = pagePageLayer.height;
1826   } else if (prop === 'x') {
1827     v.x = pagePageLayer.x;
1828   } else if (prop === 'y') {
1829     v.y = pagePageLayer.y;
1830   } else if (prop === 'angle') {
1831     v.angle = pagePageLayer.angle;
1832   } else if (prop === 'opacity') {
1833     v.opacity = pagePageLayer.opacity;
1834   } else if (prop === 'filter') {
1835     v.filter = pagePageLayer.filter;
1836   } else if (prop === 'src') {
1837     v.src = pagePageLayer.src;
1838   } else if (prop === 'alt') {
1839     v.alt = pagePageLayer.alt;
1840   } else if (prop === 'title') {
1841     v.title = pagePageLayer.title;
1842   } else if (prop === 'description') {
1843     v.description = pagePageLayer.description;
1844   } else if (prop === 'copyright') {
1845     v.copyright = pagePageLayer.copyright;
1846   } else if (prop === 'license') {
1847     v.license = pagePageLayer.license;
1848   } else if (prop === 'author') {
1849     v.author = pagePageLayer.author;
1850   } else if (prop === 'created') {
1851     v.created = pagePageLayer.created;
1852   } else if (prop === 'updated') {
1853     v.updated = pagePageLayer.updated;
1854   } else if (prop === 'deleted') {
1855     v.deleted = pagePageLayer.deleted;
1856   } else if (prop === 'status') {
1857     v.status = pagePageLayer.status;
1858   } else if (prop === 'category') {
1859     v.category = pagePageLayer.category;
1860   } else if (prop === 'tags') {
1861     v.tags = pagePageLayer.tags;
1862   } else if (prop === 'keywords') {
1863     v.keywords = pagePageLayer.keywords;
1864   }
1865   meta2d.setValue(v, { render });
1866 };
1867
1868 export const updatePagePageCanvas = (pagePageCanvas: any, prop: string, render = true) => {
1869   const v: any = { id: pagePageCanvas.id };
1870   v[prop] = pagePageCanvas[prop];
1871   if (prop === 'width') {
1872     v.width = pagePageCanvas.width;
1873   } else if (prop === 'height') {
1874     v.height = pagePageCanvas.height;
1875   } else if (prop === 'x') {
1876     v.x = pagePageCanvas.x;
1877   } else if (prop === 'y') {
1878     v.y = pagePageCanvas.y;
1879   } else if (prop === 'angle') {
1880     v.angle = pagePageCanvas.angle;
1881   } else if (prop === 'opacity') {
1882     v.opacity = pagePageCanvas.opacity;
1883   } else if (prop === 'filter') {
1884     v.filter = pagePageCanvas.filter;
1885   } else if (prop === 'src') {
1886     v.src = pagePageCanvas.src;
1887   } else if (prop === 'alt') {
1888     v.alt = pagePageCanvas.alt;
1889   } else if (prop === 'title') {
1890     v.title = pagePageCanvas.title;
1891   } else if (prop === 'description') {
1892     v.description = pagePageCanvas.description;
1893   } else if (prop === 'copyright') {
1894     v.copyright = pagePageCanvas.copyright;
1895   } else if (prop === 'license') {
1896     v.license = pagePageCanvas.license;
1897   } else if (prop === 'author') {
1898     v.author = pagePageCanvas.author;
1899   } else if (prop === 'created') {
1900     v.created = pagePageCanvas.created;
1901   } else if (prop === 'updated') {
1902     v.updated = pagePageCanvas.updated;
1903   } else if (prop === 'deleted') {
1904     v.deleted = pagePageCanvas.deleted;
1905   } else if (prop === 'status') {
1906     v.status = pagePageCanvas.status;
1907   } else if (prop === 'category') {
1908     v.category = pagePageCanvas.category;
1909   } else if (prop === 'tags') {
1910     v.tags = pagePageCanvas.tags;
1911   } else if (prop === 'keywords') {
1912     v.keywords = pagePageCanvas.keywords;
1913   }
1914   meta2d.setValue(v, { render });
1915 };
1916
1917 export const updatePagePagePageGroup = (pagePagePageGroup: any, prop: string, render = true) => {
1918   const v: any = { id: pagePagePageGroup.id };
1919   v[prop] = pagePagePageGroup[prop];
1920   if (prop === 'width') {
1921     v.width = pagePagePageGroup.width;
1922   } else if (prop === 'height') {
1923     v.height = pagePagePageGroup.height;
1924   } else if (prop === 'x') {
1925     v.x = pagePagePageGroup.x;
1926   } else if (prop === 'y') {
1927     v.y = pagePagePageGroup.y;
1928   } else if (prop === 'angle') {
1929     v.angle = pagePagePageGroup.angle;
1930   } else if (prop === 'opacity') {
1931     v.opacity = pagePagePageGroup.opacity;
1932   } else if (prop === 'filter') {
1933     v.filter = pagePagePageGroup.filter;
1934   } else if (prop === 'src') {
1935     v.src = pagePagePageGroup.src;
1936   } else if (prop === 'alt') {
1937     v.alt = pagePagePageGroup.alt;
1938   } else if (prop === 'title') {
1939     v.title = pagePagePageGroup.title;
1940   } else if (prop === 'description') {
1941     v.description = pagePagePageGroup.description;
1942   } else if (prop === 'copyright') {
1943     v.copyright = pagePagePageGroup.copyright;
1944   } else if (prop === 'license') {
1945     v.license = pagePagePageGroup.license;
1946   } else if (prop === 'author') {
1947     v.author = pagePagePageGroup.author;
1948   } else if (prop === 'created') {
1949     v.created = pagePagePageGroup.created;
1950   } else if (prop === 'updated') {
1951     v.updated = pagePagePageGroup.updated;
1952   } else if (prop === 'deleted') {
1953     v.deleted = pagePagePageGroup.deleted;
1954   } else if (prop === 'status') {
1955     v.status = pagePagePageGroup.status;
1956   } else if (prop === 'category') {
1957     v.category = pagePagePageGroup.category;
1958   } else if (prop === 'tags') {
1959     v.tags = pagePagePageGroup.tags;
1960   } else if (prop === 'keywords') {
1961     v.keywords = pagePagePageGroup.keywords;
1962   }
1963   meta2d.setValue(v, { render });
1964 };
1965
1966 export const updatePagePagePageLayer = (pagePagePageLayer: any, prop: string, render = true) => {
1967   const v: any = { id: pagePagePageLayer.id };
1968   v[prop] = pagePagePageLayer[prop];
1969   if (prop === 'width') {
1970     v.width = pagePagePageLayer.width;
1971   } else if (prop === 'height') {
1972     v.height = pagePagePageLayer.height;
1973   } else if (prop === 'x') {
1974     v.x = pagePagePageLayer.x;
1975   } else if (prop === 'y') {
1976     v.y = pagePagePageLayer.y;
1977   } else if (prop === 'angle') {
1978     v.angle = pagePagePageLayer.angle;
1979   } else if (prop === 'opacity') {
1980     v.opacity = pagePagePageLayer.opacity;
1981   } else if (prop === 'filter') {
1982     v.filter = pagePagePageLayer.filter;
1983   } else if (prop === 'src') {
1984     v.src = pagePagePageLayer.src;
1985   } else if (prop === 'alt') {
1986     v.alt = pagePagePageLayer.alt;
1987   } else if (prop === 'title') {
1988     v.title = pagePagePageLayer.title;
1989   } else if (prop === 'description') {
1990     v.description = pagePagePageLayer.description;
1991   } else if (prop === 'copyright') {
1992     v.copyright = pagePagePageLayer.copyright;
1993   } else if (prop === 'license') {
1994     v.license = pagePagePageLayer.license;
1995   } else if (prop === 'author') {
1996     v.author = pagePagePageLayer.author;
1997   } else if (prop === 'created') {
1998     v.created = pagePagePageLayer.created;
1999   } else if (prop === 'updated') {
2000     v.updated = pagePagePageLayer.updated;
2001   } else if (prop === 'deleted') {
2002     v.deleted = pagePagePageLayer.deleted;
2003   } else if (prop === 'status') {
2004     v.status = pagePagePageLayer.status;
2005   } else if (prop === 'category') {
2006     v.category = pagePagePageLayer.category;
2007   } else if (prop === 'tags') {
2008     v.tags = pagePagePageLayer.tags;
2009   } else if (prop === 'keywords') {
2010     v.keywords = pagePagePageLayer.keywords;
2011   }
2012   meta2d.setValue(v, { render });
2013 };
2014
2015 export const updatePagePagePageCanvas = (pagePagePageCanvas: any, prop: string, render = true) => {
2016   const v: any = { id: pagePagePageCanvas.id };
2017   v[prop] = pagePagePageCanvas[prop];
2018   if (prop === 'width') {
2019     v.width = pagePagePageCanvas.width;
2020   } else if (prop === 'height') {
2021     v.height = pagePagePageCanvas.height;
2022   } else if (prop === 'x') {
2023     v.x = pagePagePageCanvas.x;
2024   } else if (prop === 'y') {
2025     v.y = pagePagePageCanvas.y;
2026   } else if (prop === '
```



```

4
5 const addAnimate = () => {
6   openedCollapses.value.push(props.pen.animations.length);
7   props.pen.animations.push({
8     name: '动画' + (props.pen.animations.length + 1),
9   });
10 };
11
12 const changeAnimate = (item: any) => {
13   const animate: any = animateList.find((elem: any) => {
14     return elem.value === item.animate;
15   });
16   if (!animate) {
17     return;
18   }
19   item.frames = deepClone(animate.data);
20 };
21
22 const play = (i: number) => {
23   meta2d.startAnimate(props.pen.id, i); 动画执行
24   isPlaying.value = i;
25 };
26
27 const stop = () => {
28   meta2d.stopAnimate(props.pen.id); 动画停止
29   isPlaying.value = -1;
30 };

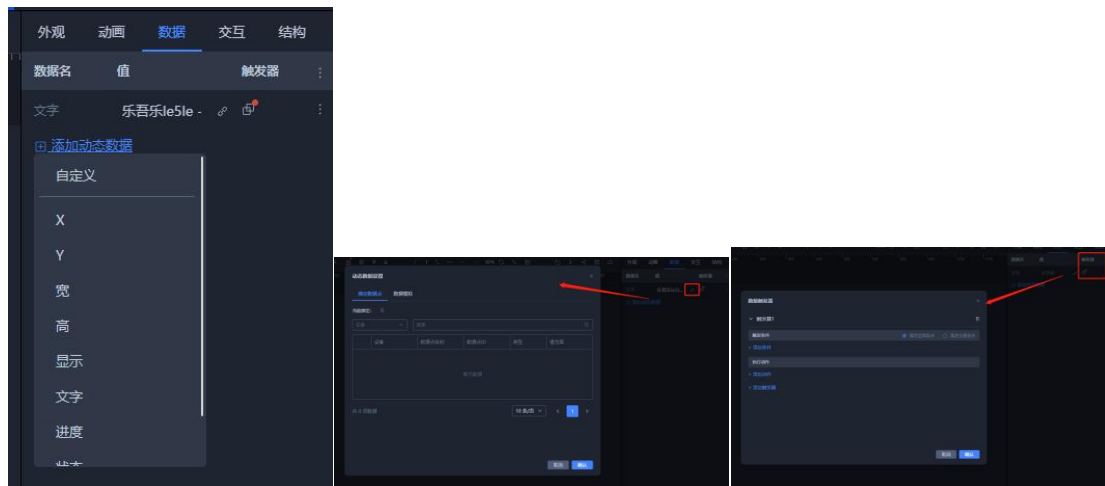
```

animations 属性用于存放该节点配置的多个动画，最终动画的执行是由 frames 属性控制的，通过 startAnimate/stopAnimate 方法控制动画的执行/停止。动画相关属性介绍可查看文档：

<https://doc.le5le.com/document/119895613>

### 3. 数据（components/PenDatas.vue）

数据主要是绑定数据点和设置（值变化）触发器，主要是修改 pen 的 realTimes 属性，具体可以查看文档：<https://doc.le5le.com/document/135786389>



### 4. 交互（components/PenEvents.vue）

交互主要是配置节点的交互事件，主要修改的是 pen 的 events 属性，具体可以查看文档：<https://doc.le5le.com/document/119627190>



## 5. 结构

同 不选中图元 结构板块

### 5.1.3.3 多选图元

① 目录 components/PensProps.vue

② 源码说明：



外观主要包括对多个图元的统一锁定、显示状态的修改，对齐操作以及外观、文字等一些公共样式的统一修改。

锁定/显示主要通过调用核心库 `setValue`、`setVisible` 方法。

```
const lock = (v: LockState) => {
  data.locked = v;
  for (const item of selections.pens) {
    meta2d.setValue({
      id: item.id,
      locked: v,
    });
  }
};

const visible = (v: boolean) => {
  data.visible = v;
  for (const item of selections.pens) {
    meta2d.setVisible(item as any, v);
  }
};
```

对齐操作也是直接调用核心库开源方法，具体方法说明可见文档：

<https://doc.le5le.com/document/119882449#%E5%87%BD%E6%95%B0>

```
const align = (align: string) => {
  if (align === 'h-distribute') {
    meta2d.spaceBetween(meta2d.store.active);
  } else if (align === 'v-distribute') {
    meta2d.spaceBetweenColumn(meta2d.store.active);
  } else {
    meta2d.alignNodes(align, meta2d.store.active);
  }
};

const align2 = (align: string) => {
  if (align === 'same-size') {
    meta2d.beSameByLast(meta2d.store.active);
  } else {
    meta2d.alignNodesByLast(align, meta2d.store.active);
  }
};
```

修改公共样式和上面修改单个图元样式一样，调用核心库 `setValue` 方法，特殊属性提前处理。注意这里遍历所有 `pen` 进行 `setValue` 是没有直接 `render` 的，最后再统一 `render`。具体可以参考文档：

<https://doc.le5le.com/document/119882449#setValue>



```

854 };
855
856 const align = (align: string) => {
857   if (align === 'h-distribute') {
858     meta2d.spaceBetween(meta2d.store.active);
859   } else if (align === 'v-distribute') {
860     meta2d.spacebetweenColumn(meta2d.store.active);
861   } else {
862     meta2d.alignNodes(align, meta2d.store.active);
863   }
864 };
865
866 const align2 = (align: string) => {
867   if (align === 'same-size') {
868     meta2d.besameByLast(meta2d.store.active);
869   } else {
870     meta2d.alignNodesByLast(align, meta2d.store.active);
871   }
872 };
873
874
875
876 const changeValue = (prop: string) => {
877   for (const item of selections.pens) {
878     data.id = item.id;
879     updatePen(data, prop, false);
880   }
881   meta2d.render();
882 };
883
884
885 const onFontFamily = (fontfamily: string) => {
886   data.fontfamily = fontfamily;
887   data.fontfamilypopupVisible = false;
888   changeValue('fontfamily');
889 };
890
891
892 const onFontPopupVisible = (val: boolean) => {
893   data.fontfamilypopupVisible = val;
894 };
895
896
897
898
899
900

```

```

1  export const updatePen = (pen: any, prop: string, render = true) => {
2    const v: any = { id: pen.id };
3    v[prop] = pen[prop];
4    if (prop === 'width' && pen.ratio) {
5      const rect = meta2d.findOne(pen.id);
6      v.height = (pen.width / rect.width) * rect.height;
7      pen.height = v.height;
8    } else if (prop === 'height' && pen.ratio) {
9      const rect = meta2d.findOne(pen.id);
10     v.width = (pen.height / rect.height) * rect.width;
11     pen.width = v.width;
12   } else if (prop === 'shadow') {
13     if (v[prop]) {
14       lv.shadowOffsetX && (v.shadowOffsetx = 0);
15       lv.shadowOffsetY && (v.shadowOffsety = 0);
16       lv.shadowBlur && (v.shadowblur = 0);
17     } else {
18       v.shadowColor = '';
19     }
20   } else if (prop === 'linegradientcolor') {
21     // @ts-ignore (property) Meta2dStore: Meta2dStore
22     if (meta2d.store.active[0].name === 'line') {
23       // @ts-ignore
24       meta2d.store.active[0].calculative.gradientcolorStop = null;
25     } else {
26       // @ts-ignore
27       meta2d.store.active[0].calculative.linegradient = null;
28     }
29     // 不同模式切换不同的系统配色
30   } else if (prop === 'titlefnjs') {
31     v.titlefn = null;
32   } else if (prop === 'dash') {
33     v.lineDash = lineDashObj[v[prop]];
34   }
35   meta2d.setValue(v, { render });
36 };

```

## 5.2 预览页面

预览页面只有中心画布用于展示大屏页面。

```

EXPLORER
  src > views > Preview.vue
  src > views > Preview.vue > {} script setup > open
  You, 2 months ago | 2 authors (You and others)
  1 <template>
  2   <div class="preview" :style="{ background: bgColor}">
  3     <div class="meta2d-canvas" ref="meta2dDom"></div>
  4   </div>
  5 </template>
  6
  7 <script setup lang="ts">
  8   import { ref, onMounted, watch, onUnmounted } from 'vue';
  9   import localforage from 'localforage';
 10   import { localStorageName } from '@services/utils';
 11   import { defaultFormat } from '@services/defaults';
 12   import { useRouter, useRoute } from 'vue-router';

```

对应运行某个大屏图纸：



## 六 目录介绍

### 6.1 Public 公共静态资源目录

public/icon 项目图标库

public/img 项目图片资源存放

public/js 项目需要的一些离线资源包，例如 echarts 包（echarts.min.js）

public/theme echarts 图形库主题文件存放位置，具体可以查看：

<https://echarts.apache.org/zh/theme-builder.html>

public/view 编辑器“下载离线部署包”/“组件”等功能所需要的运行环境文件

Public/data.xlsx 数据集 Excel 示例

Public/favicon.ico 左上角 logo

Public/rotate.cur 图形节点旋转时鼠标样式

### 6.2 Src 开发目录

Src/assets 静态资源，fonts 下是系统字体

Src/router 路由配置

Src/services 公用服务（公用方法）

Src/styles 公用样式文件

Src/view 主要的页面，首页和预览页面

Src/view/components 组件

Src/App.vue 主组件

Src/global.d.ts 可以从全局范围访问的库

Src/http.ts axios 配置和网络请求/响应拦截器

Src/main.ts 入口 ts 文件

### 6.3 其他

Index.html 入口 html 文件

Package.json 项目依赖描述

postcss.config.js postcss 配置文件

Tsconfig.json ts 配置文件

Vite.config.ts vite 配置文件

## 七 运行流程

Vue 项目运行流程：index.html > App.vue 的 export 外的 js 代码 > main.js > App.vue 的 export 里面的 js 代码

想了解更多请学习 vue: <https://v3.cn.vuejs.org/>

快速修改代码：用 VS code 搜索关键字

## 八 代码中实现登录链接

### 8.1 完全自己实现后端

可以参考后端 API 接口文档：<https://doc.le5le.com/document/7>

### 8.2 购买了乐吾乐后端

参考后端使用手册运行部署后端。然后通过前后端分离模式部署即可