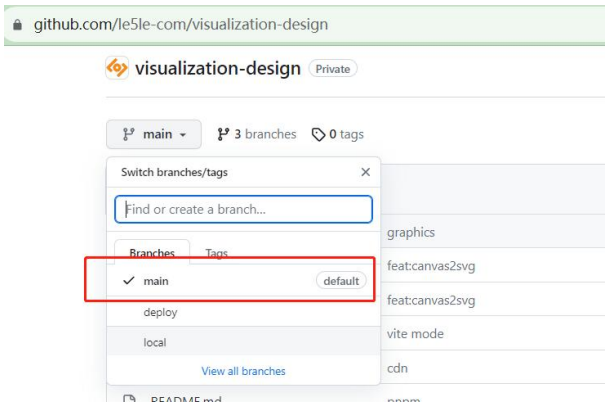
visualization-design 源码使用手册

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：




```
Administrator@LAPTOP-UKGGNDEF MINGW64 [redacted]/大屏/test
$ git clone https://github.com/le5le-com/visualization-design.git
Cloning into 'visualization-design'...
remote: Enumerating objects: 2489, done.
remote: Counting objects: 100% (662/662), done.
remote: Compressing objects: 100% (254/254), done.
remote: Total 2489 (delta 485), reused 568 (delta 405), pack-reused 1827
Receiving objects: 100% (2489/2489), 5.21 MiB | 1.34 MiB/s, done.
Resolving deltas: 100% (1861/1861), done.

Administrator@LAPTOP-UKGGNDEF MINGW64 [redacted]/大屏/test
$ cd visualization-design/

Administrator@LAPTOP-UKGGNDEF MINGW64 [redacted]/大屏/test/visualization-
design (main)
$
```

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

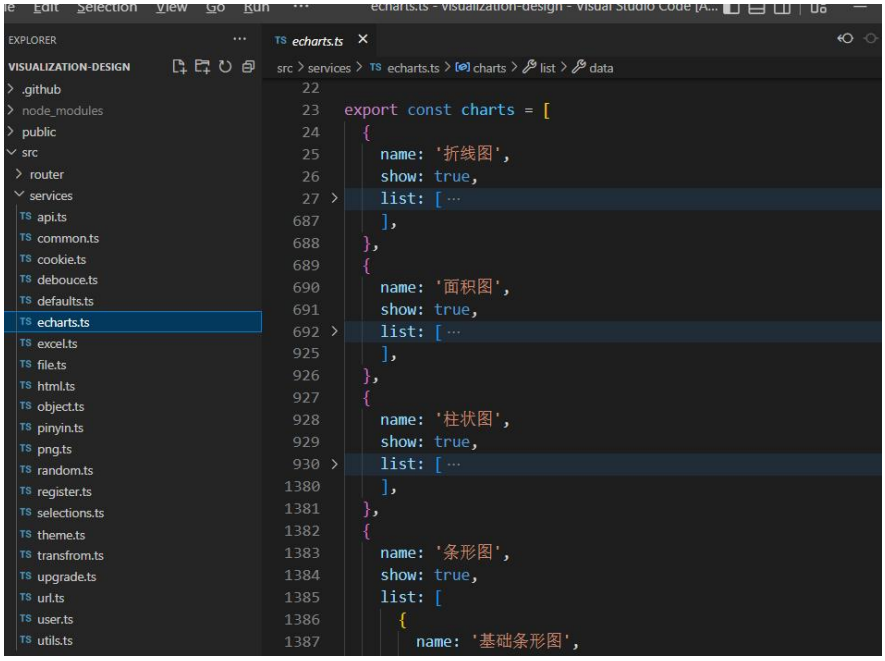
通过/api/data/v.component/list 接口请求，通过 systemFlag=1 区分于非系统组件。

```
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%20%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

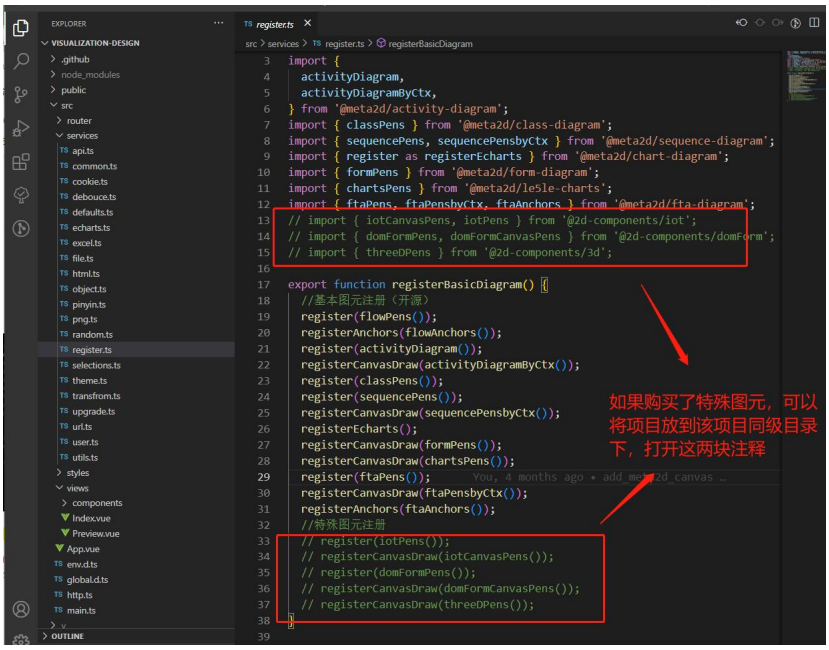
图元主要是指不同行业/场景下的 svg/png 组件库，接口请求同素材，传入参数 path 分别是”svg”/”png”

```
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;
```

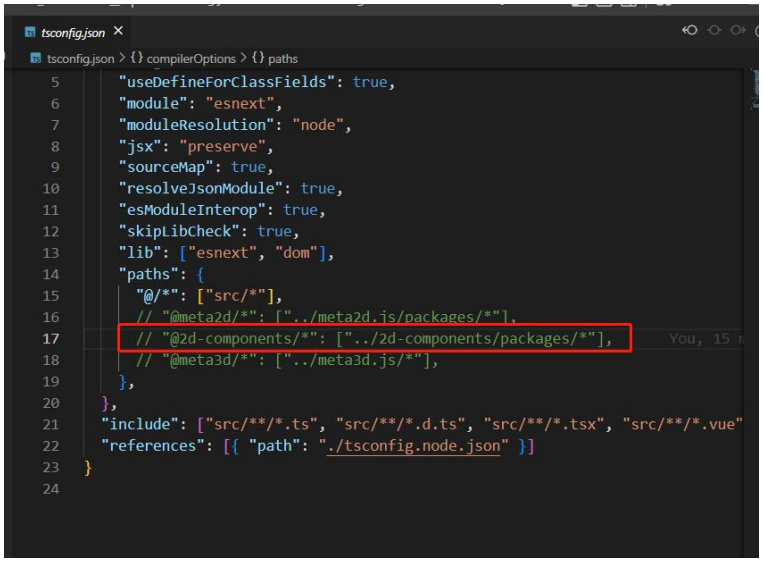
3.7 控件

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

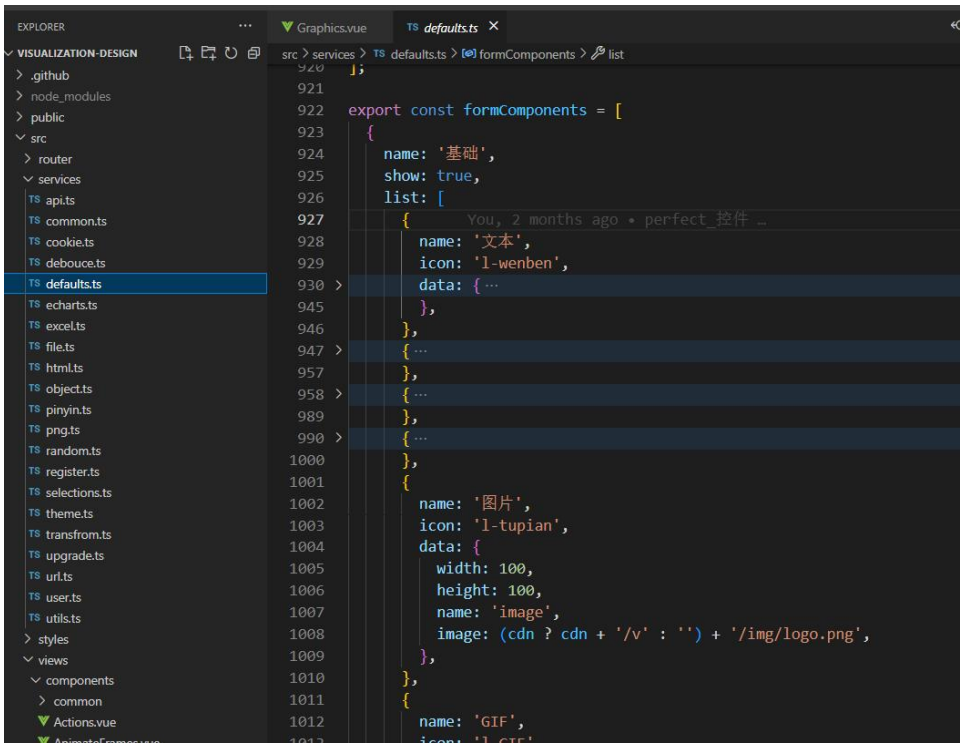
图元的注册是在 src/services/register.ts 文件里



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：



图形库的配置是在 src/services/defaults 文件里。



3.8 图形

图形主要是核心库开源的基础图元，具体可见文档：
https://doc.le5le.com/document/119754049
注册同控件，配置文件如下：

```
export const shapes = [
  {
    name: '基本形状',
    show: true,
    list: [ ···
    ],
  },
  {
    name: '脑图',
    show: true,
    list: [ ···
    ],
  },
  {
    name: '流程图',
    show: true,
    list: [ ···
    ],
  },
  {
    name: '活动图',
    show: true,
    list: [
      {
```

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.将数据对应到云盘文件夹
  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);      You, 5 months ago • feat:我的资源
    }
  }
  return await Promise.all(···
  );
};
```

获取云盘下所有图片资源。

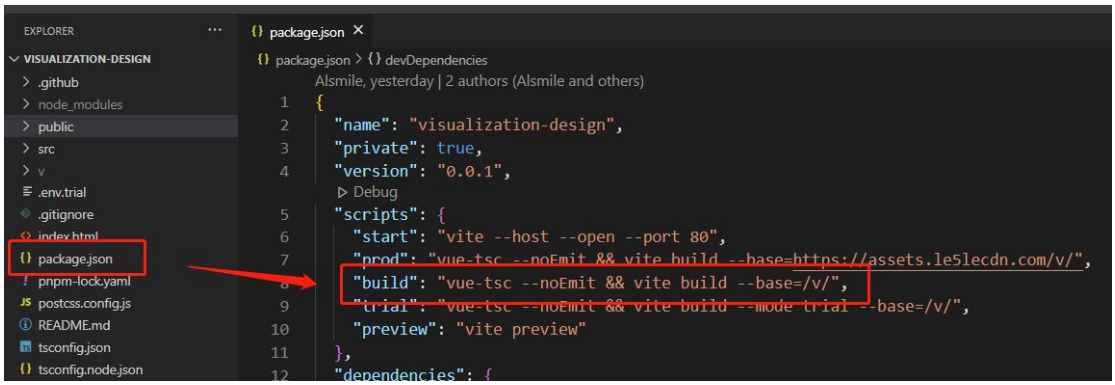
3.9.5 3D

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

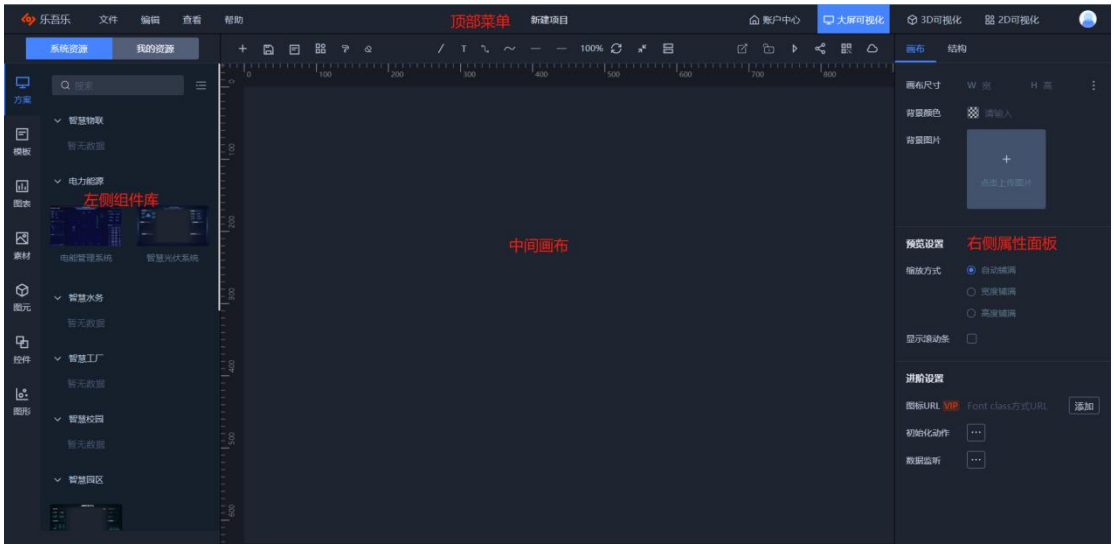
整个项目分为两个页面：①Index.vue 大屏编辑页面 ②Preview.vue 大屏预览页面（运行页面）

```
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>      Alsmile, 4 months ago • in
      <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">
```

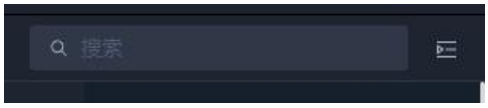
③ 右侧是一些导航链接，包括账户中心、乐吾乐其他产品、登录/用户菜单

5.1.2 左侧组件库

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

```vue
<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

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

```
const onSearch = () => {      Alsmile, last month • search ing
  debounce(searchGraphics, 300);
};

const searchGraphics = async () => {
  if (search.value) {
    activedPanels[activedGroup.value].splice(
      0,
      activedPanels[activedGroup.value].length
    );
  }

  for (const group of subGroups.value) {
    for (const item of group.list) {
      if (search.value) {
        item.visible = searchObjectPinyin(item, 'name', search.value);
      } else {
        item.visible = true;
      }
    }

    if (search.value) {
      activedPanels[activedGroup.value].push(group.name);
    }
  }
};
```

右侧按钮通过控制当前活动面板 actived Panels 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 (!activedPanels[activedGroup.value]) {
    return;
  }

  if (activedPanels[activedGroup.value].length) {
    activedPanels[activedGroup.value] = [];
  } else {
    activedPanels[activedGroup.value] = [];
    for (const item of subGroups.value) {
      activedPanels[activedGroup.value].push(item.name);
    }
  }
};      Alsmile, last month • 图元
```

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) => {
  activedGroup.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 = () => {        You,
  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;       You, 3 months ago • perfect_vie
    meta2d.store.options.disableAnchor = true;
  }
};
```

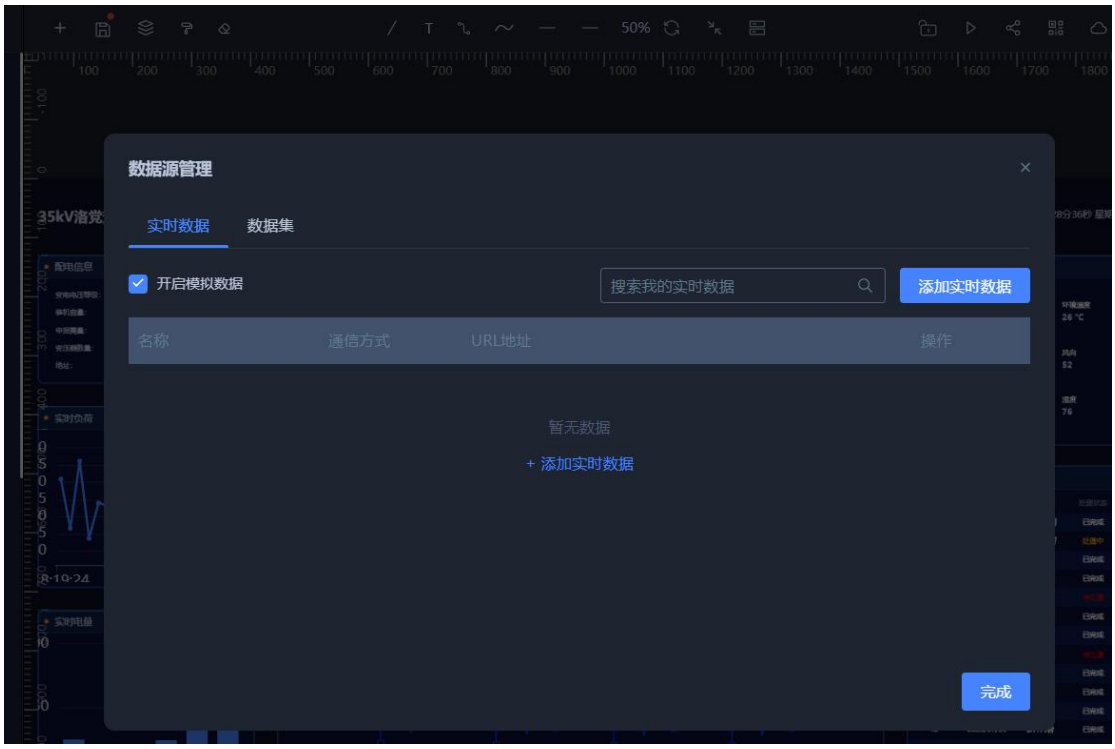
通过修改 options 配置默认连线样式，下图代码表示修改连线类型。

```
const changeLineType = (value: string) => {
  currentLineType.value = value;
  if (meta2d) {
    meta2d.store.options.drawingLineName = value;       You, 3 mont
    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();
  };
});
```

③ 右键菜单（components/ ContextMenu.vue）


```
const onMenu = (val: string) => {
  switch (val) {
    case 'addAnchorHand':
      meta2d.addAnchorHand();
      break;
    case 'removeAnchorHand':
      meta2d.removeAnchorHand();
      break;
    case 'toggleAnchorHand':
      meta2d.toggleAnchorHand();
      break;
    case 'top':
    case 'bottom':
    case 'up':
    case 'down':
    case 'undo':
    case 'redo':
    case 'cut':
    case 'copy':
    case 'paste':
      (meta2d as any)[val]();
      break;
    case 'group':
      meta2d.combine(meta2d.store.active);
      break;
    case 'states':
      meta2d.combine(meta2d.store.active, 0);
      break;
    case 'unGroup':
      meta2d.uncombine();
      break;
    case 'lock':···
```

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）


```
const changeValue = (e: any, key: string) => {
  // @ts-ignore
  meta2d.store.data[key] = e;
  if (key === 'background') {
    meta2d.store.patchFlagsBackground = true;
  }
  meta2d.render();
  openData();
};
```

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

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

```
const fileSuccessed = async (content: any) => {
  // meta2d.store.patchFlagsBackground = true;
  meta2d.setBackgroundImage(content.response.url);
  setTimeout(() => {
    meta2d.store.patchFlagsBackground = true;
    meta2d.render();
  }, 1000);
};

const fileRemoved = () => {
  meta2d.setBackgroundImage('');
  meta2d.store.patchFlagsBackground = true;
  meta2d.render();
  data.background = [];
};
```

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。


```
const opened = () => {
  let fit: any =
    (meta2d.store.data as any).scaleMode === '3'
      ? 'height'
      : (meta2d.store.data as any).scaleMode === '2'
      ? 'width'
      : true;
  meta2d.fitSizeView(fit, 10);
  if ((meta2d.store.data as any).scroll) {
    meta2d.setOptions({
      scroll: true,
    });
  }
};
```

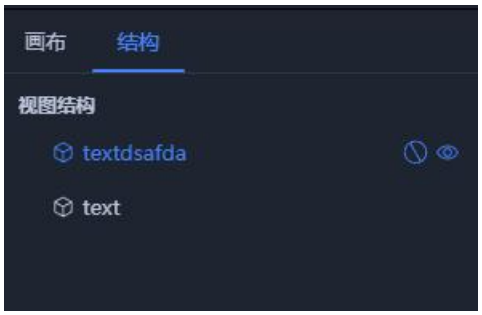
进阶设置首先可以配置图标地址，输入字体图标地址后，会加载对应的图标库，在选中画笔进行图标设置时可以在图标抽屉中展示请求到的图标库。

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：

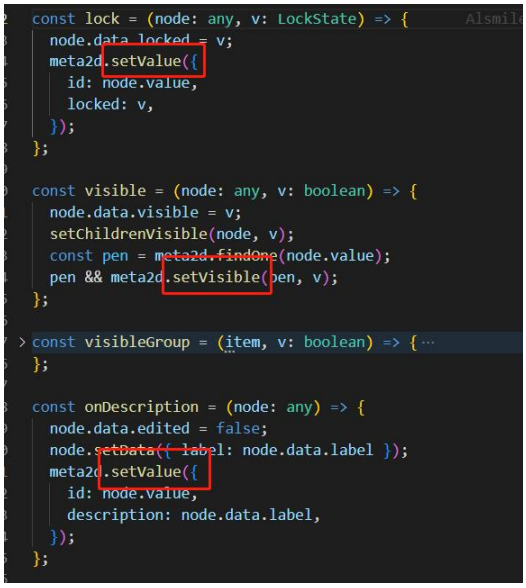
https://doc.le5le.com/document/119620524#%E8%A7%A3%E6%9E%90%E8%87%AA%E5%AE%9A%E4%B9%89%E6%A0%BC%E5%BC%8F%E6%95%B0%E6%8D%AE

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：



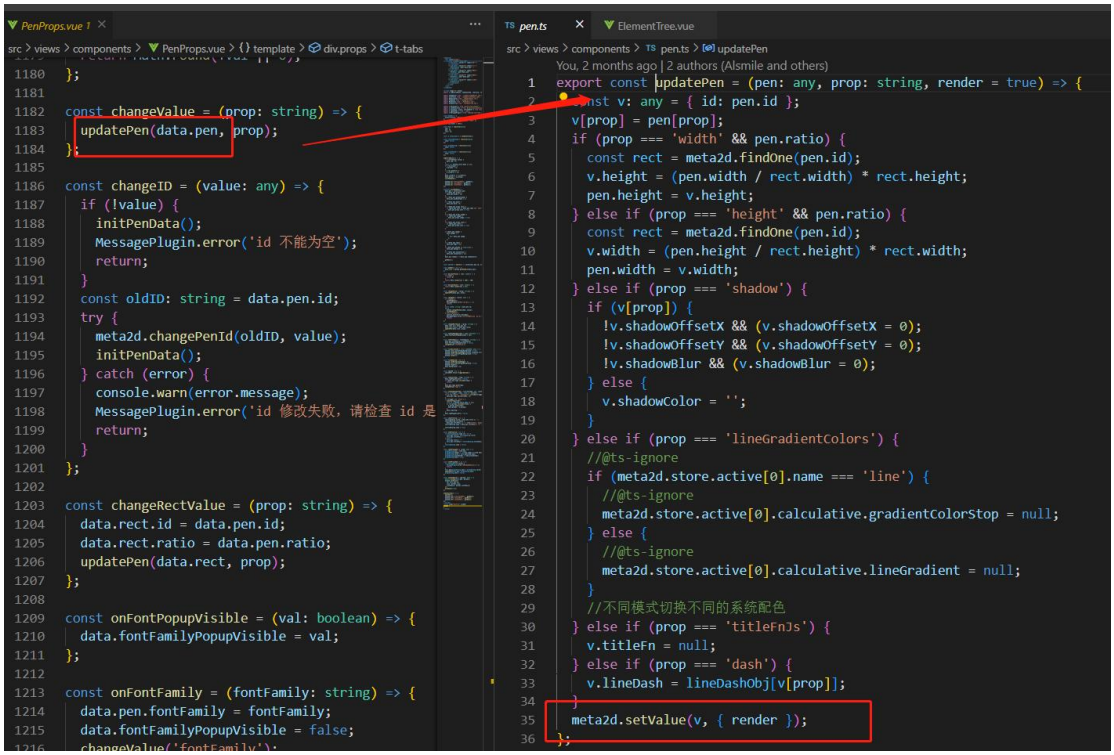
在分组中，可以新增/删除自定义分组名称，可以批量控制该分组对应图元的显示/隐藏。

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。

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

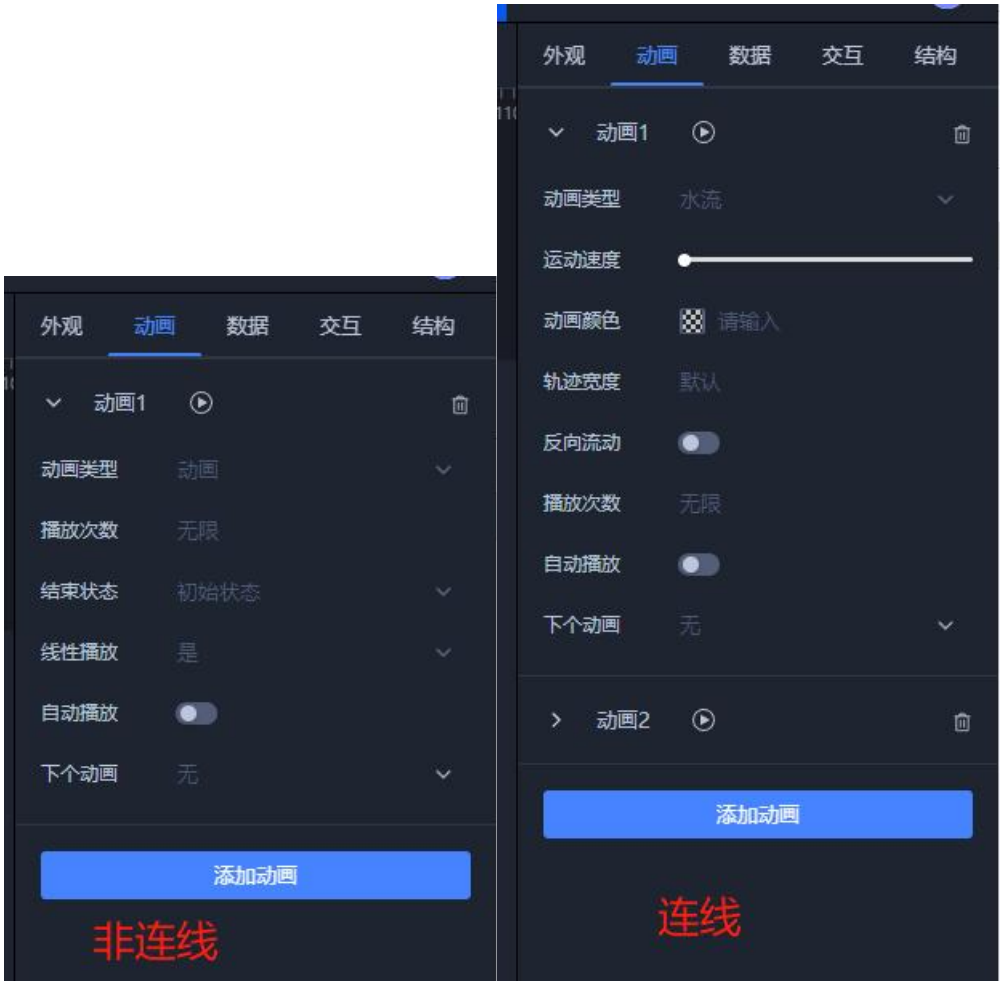


大部分属性更改都是通过核心库 setValue 方法去修改，特殊的属性提前做处理。具体的属性名介绍可看官方文档：

https://doc.le5le.com/document/119895613#%E5%B1%9E%E6%80%A7

2. 动画（components/PenAnimates.vue）

动画需要区分连线节点和非连线节点，节点支持设置多个动画，点击播放可以执行你想要播放的动画。




```
const addAnimate = () => {
  openedCollapses.value.push(props.pen.animations.length);
  props.pen.animations.push({
    name: '动画' + (props.pen.animations.length + 1),
  });
};      Alsmile, 3 months ago • animate

const changeAnimate = (item: any) => {
  const animate: any = animateList.find((elem: any) => {
    return elem.value === item.animate;
  });

  if (!animate) {
    return;
  }

  item.frames = deepClone(animate.data);
};

const play = (i: number) => {
  meta2d.startAnimate(props.pen.id, i);
  isPlaying.value = i;
};

const stop = () => {
  meta2d.stopAnimate(props.pen.id);
  isPlaying.value = -1;
};
```

动画执行

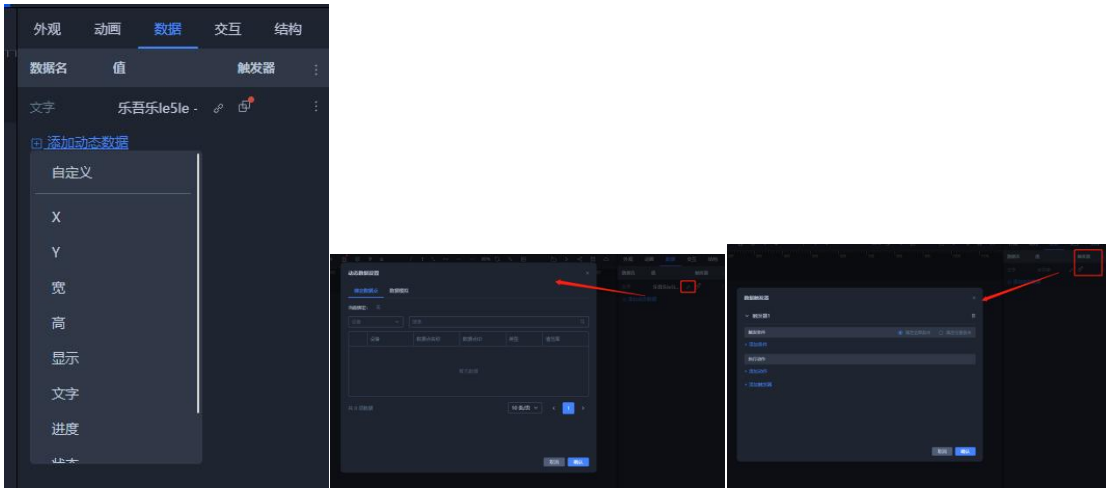
动画停止

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：

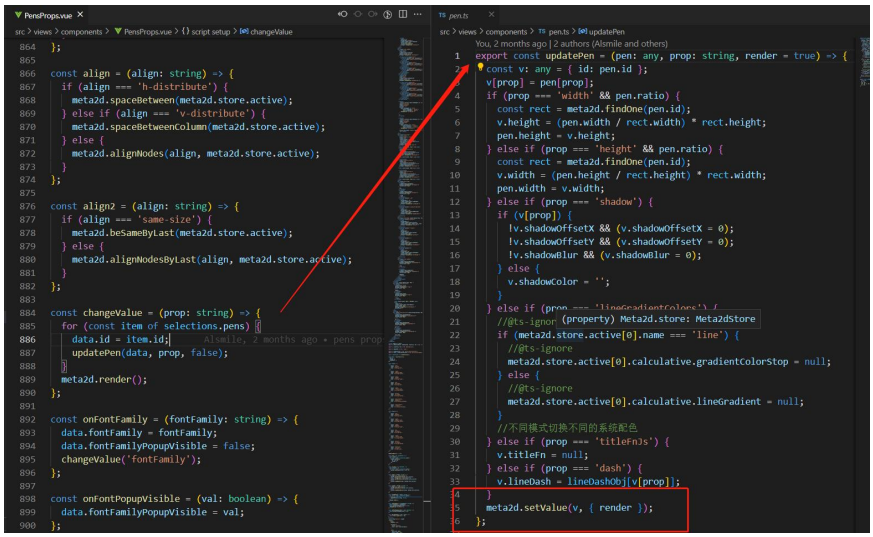
外观主要包括对多个图元的统一锁定、显示状态的修改，对齐操作以及外观、文字等一些公共样式的统一修改。
锁定/显示主要通过调用核心库 setValue、setVisible 方法。

```
const lock = (v: LockState) => {       Alsmile, 2
  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

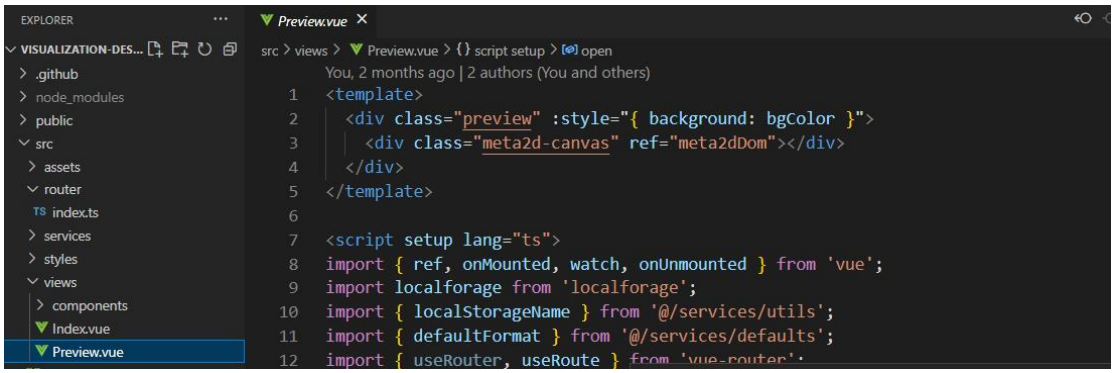
```
const align = (align: string) => {
  if (align === 'h-distribute') {
    meta2d.spaceBetween(meta2d.store.active);
  } else if (align === 'v-distribute') {
    meta2d.spaceBetweenColumn(meta2d.store.active);
  } else {
    meta2d.alignNodes(align, meta2d.store.active);
  }
};

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

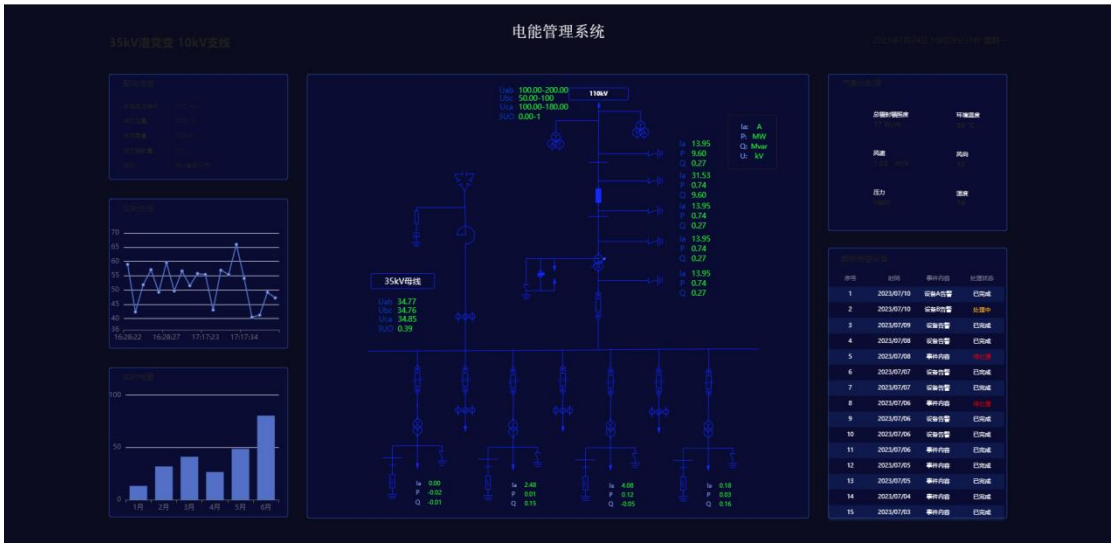


5.2 预览页面

预览页面只有中心画布用于展示大屏页面。



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