

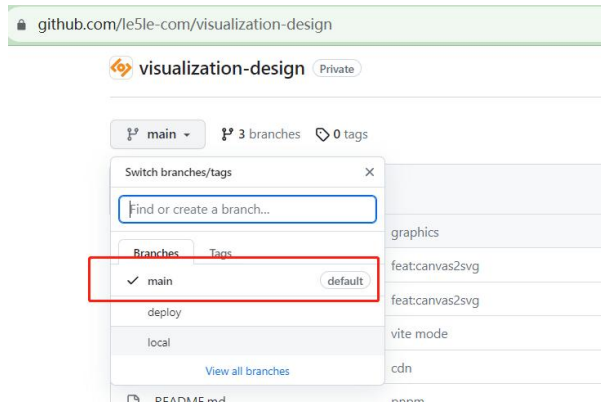
visualization-design 源码使用手册

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 控件	3
3.4 设备	5
3.5 图表	9
3.6 素材	10
3.7 图形	12
3.8 组件	12
3.9 我的资源	13
3.9.1 方案	13
3.9.2 模版	13
3.9.3 组件	14
3.9.4 图片	14
3.9.5 3D	14
四 编译打包	15
五 源码结构说明	15
5.1 编辑器页面	15
5.1.1 头部菜单	16
5.1.2 左侧组件库	17
5.1.3 中间画布	24
5.1.3 右侧属性面板	28
5.1.3.1 不选中图元	28
5.1.3.2 单选图元	30
5.1.3.3 多选图元	36
5.2 预览页面	38
六 目录介绍	39
6.1 Public 公共静态资源目录	39
6.2 Src 开发目录	39
6.3 其他	39
七 运行流程	39
八 代码中实现登录链接	40
8.1 完全自己实现后端	40
8.2 购买了乐吾乐后端	40
九 部署集成	40

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 ~/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 ~/大屏/test
$ cd visualization-design/

Administrator@LAPTOP-UKGGNDEF MINGW64 ~/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 system 表示为系统方案。

```
const getCaseProjects = async (name: string, system:boolean, template:boolean, current = 1, pageSize = 1000) => {
  const query: any = { tags: name };
  let collection = name == '系统组件' ? 'v.component' : 'v';
  let data = { system, template };
  if(!data.system){
    delete data.system;
  }
  const ret: any = await axios.post(
    `/api/data/${collection}/list`,
    data,
    {
      // {
      //   // query: {
      //   //   tags: "系统方案"
      //   // },
      //   //shared: true,
      //   // projection: "id,id,name,image,price,case",
      //   // sort: { createdAt: 1 },
      //   // systemFlag
      //   system,
      //   template
      // },
    }
  );
  if (!ret) { ... }
  for (const item of ret.list) { ... }
  return ret.list;
}
```

3.2 模版

模板同场景，通过 template 参数表示为模版。

将交付的文件中，“模版”文件夹中 json 文件，在大屏编辑器平台-文件-导入文件后保存即可。再到运营中心设置为系统模版。

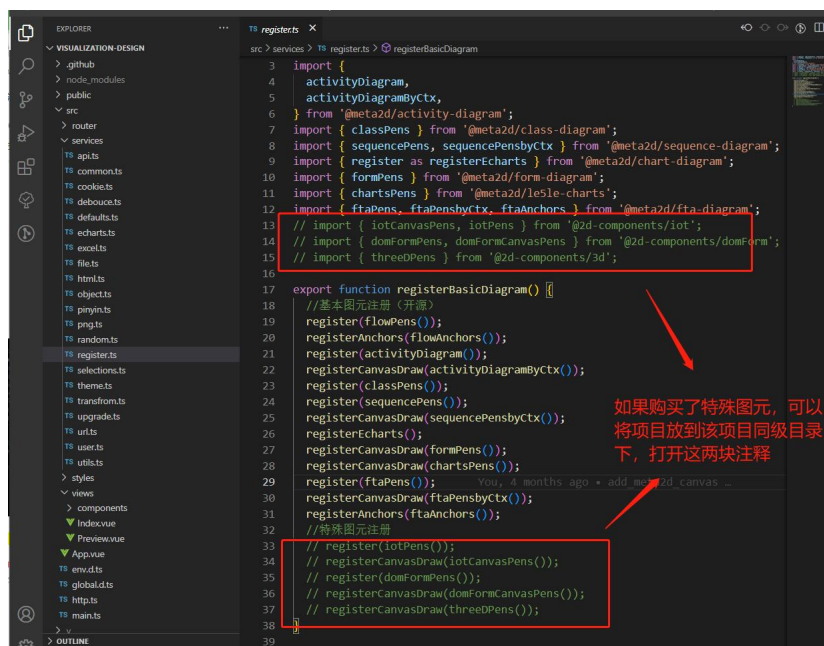
乐吾乐大屏可视化系统V1.0交付文件 > 08 模版 >

名称	修改日期
布局	2024/8/2
主题	2024/8/2

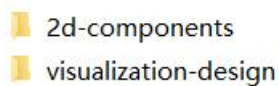
3.3 控件

控件主要是指一些控件图元，包含基础图元和特殊图元，对应交付文件中“控件源码”。控件的详细介绍可以查看文档：<https://doc.le5le.com/document/146738139>

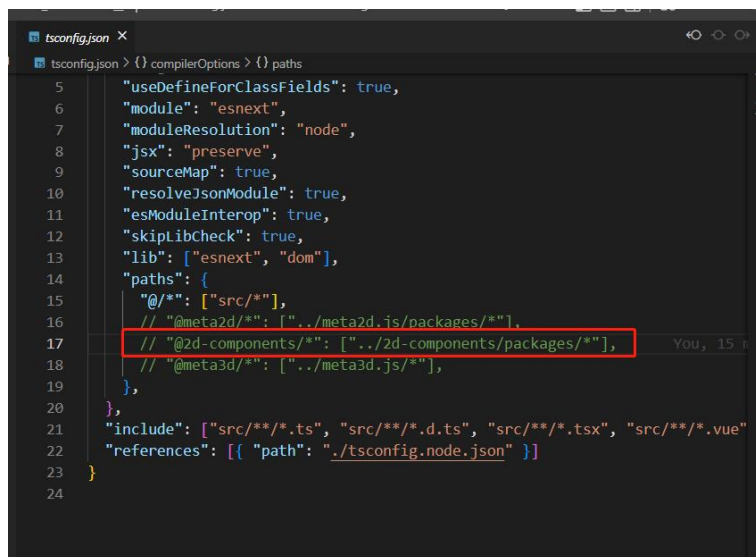
图元的注册是在 src/services/register.ts 文件里

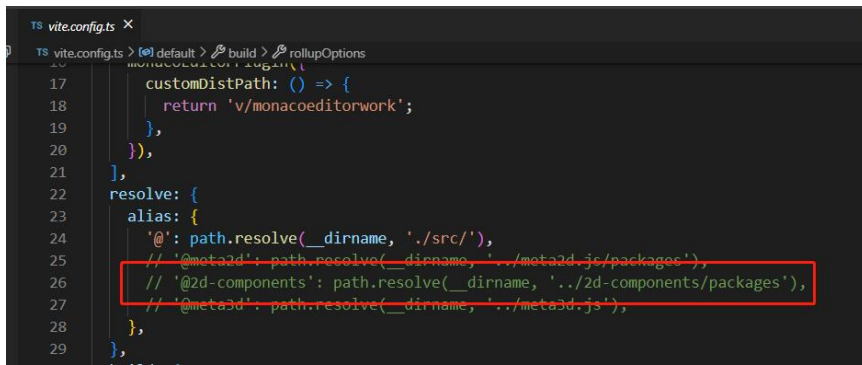


需要将控件项目(2d-components)放到此项目同级目录下，注意如果是 2d-components-main，文件夹重命名为 2d-components

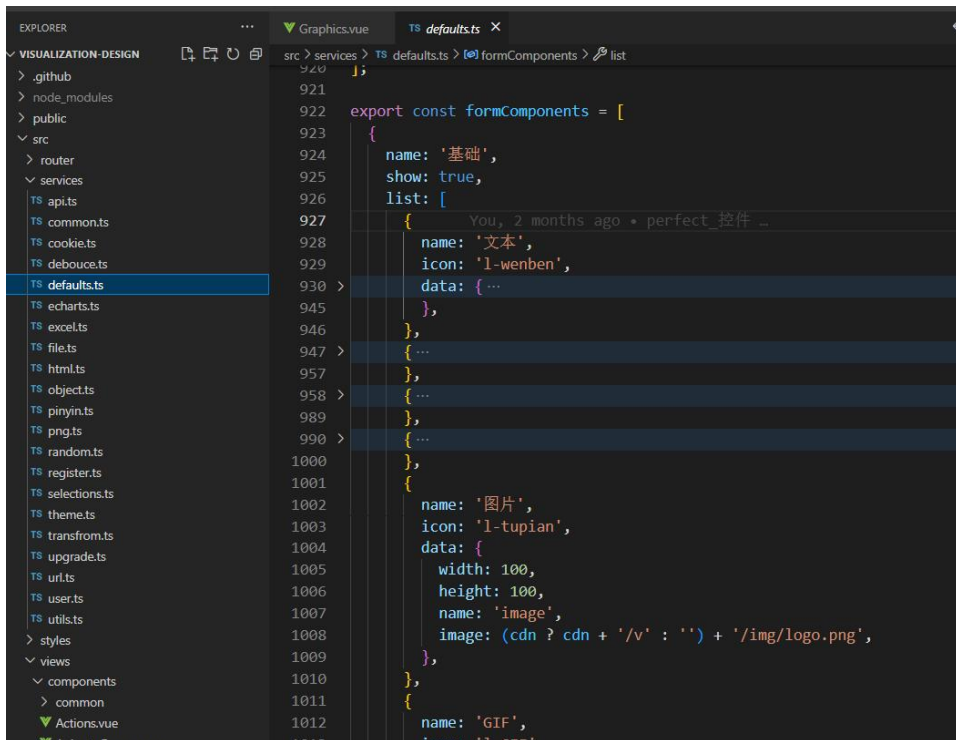


并取消下面框选位置的注释：





图形库的配置是在 src/services/defaults 文件里。



3.4 设备

设备主要是指不同行业/场景下的 svg/png/js/icon 图形库，对应资源在交付文件的“设备”文件夹下。

乐吾乐大屏可视化系统V1.0交付文件 > 07 设备 >		
名称		修改日期
icon图形库		2024/9/3
js图形库		2024/8/2
png图形库		2024/8/2
svg图形库		2024/8/2

1. png 和 svg 图形库通过 /api/assets/folders 和 /api/assets/files 两个接口获取得到，传入参数 path 分别是“svg”/“png”，代码地址如下

```
Graphics.vue X
src > views > components > Graphics.vue > {} script setup > groupChange

719     }
720     subGroups.value = materials;
721     lastName = name;
722     break;
723   case '设备':
724     if (!pngs.length) {
725       loading.value = true;
726       pngs.push(...(await getFolders('png')));
727       pngs.push(...(await getFolders('svg', true)));
728       pngs.push(...jsDiagrams);
729       loading.value = false;
730       const _pngs: string = await localforage.getItem('le5leV-pngs');
731       if (_pngs) {
732         Object.assign(pngs, JSON.parse(_pngs));
733       }
734     }
735     subGroups.value = pngs;
736     lastName = name;
737     break;
738   case '图形':
```

```
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 [];
  }

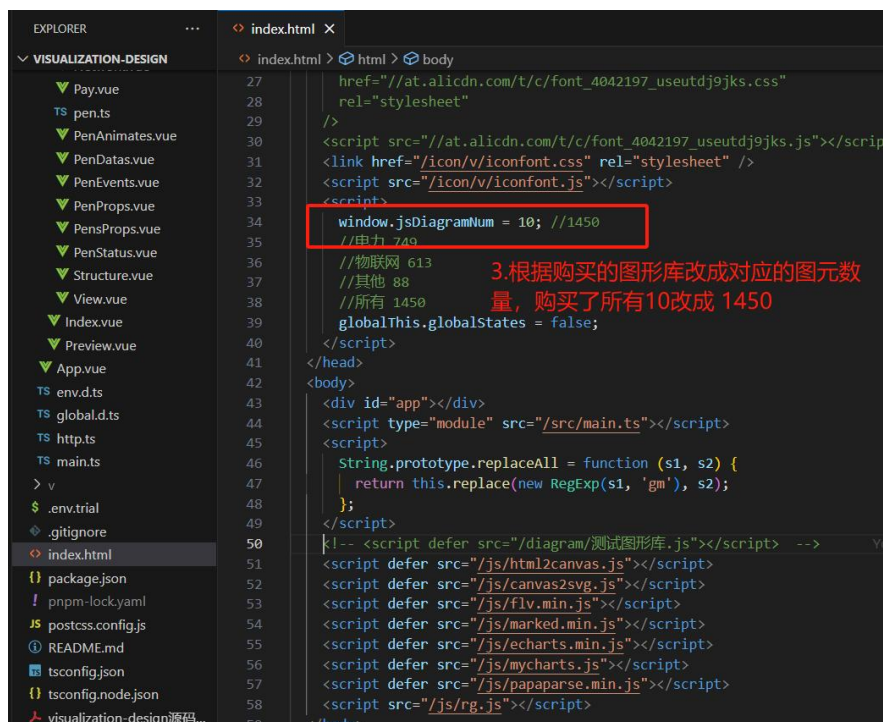
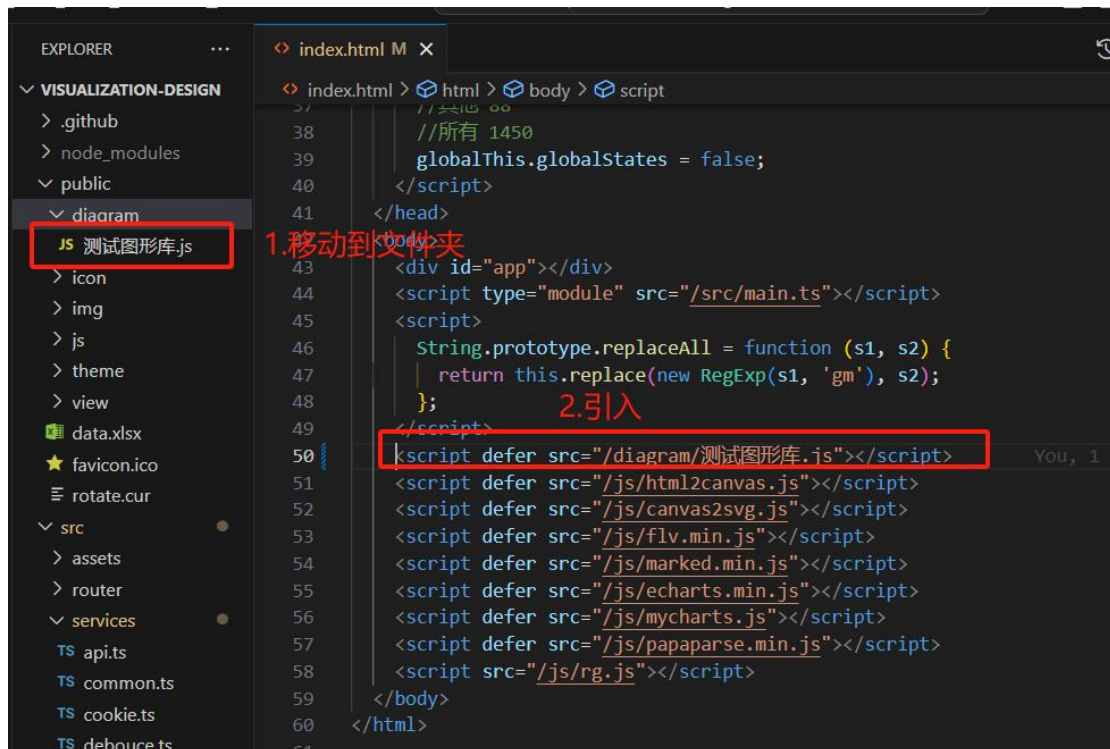
  const files: any = await axios.post('/api/assets/files', {
    path,
  });

  const results = [];
  for (const item of folders.list) { ...
  }
  return results;
}
```

将 png/svg 文件夹及其文件夹下资源放到服务器 app_web_assets_root（查看后端使用手册）目录下即可。前端 path 参数传入“png”，后端接口就会返回 app_web_assets_root 下的 png 下文件夹及文件名。

2. Js 图形库对应 设备/js 图形库。将 js 图形库中所有的 js 文件放到项目的 public/diagram 文件夹下，在 index.html 文件中，引入所有这些 js 文件。

以测试图形库.js 为例：



Js 图形库的注册代码如下：

```

function getJsDiagram() {
  if (globalThis.registerToolsNew) {
    (window as any).registerToolsNew();
    const temJSClass = [];
    const temJSMaterial = [];
    globalThis.meta2dTools.forEach((item: any) => {
      if (temJSClass.indexOf(item.subClassName) === -1) {
        temJSClass.push(item.subClassName);
        temJSMaterial.push({
          name: item.subClassName,
          show: true,
          list: [],
        });
      }
      temJSMaterial.forEach((_class: any) => {
        if (_class.name === item.subClassName) {
          _class.list.push(item);
        }
      });
    });
    jsDiagrams.push(...temJSMaterial);
    // 切换回页面仍需要注册, 所以此处不清空
    // window.meta2dTools = undefined;
  }
}

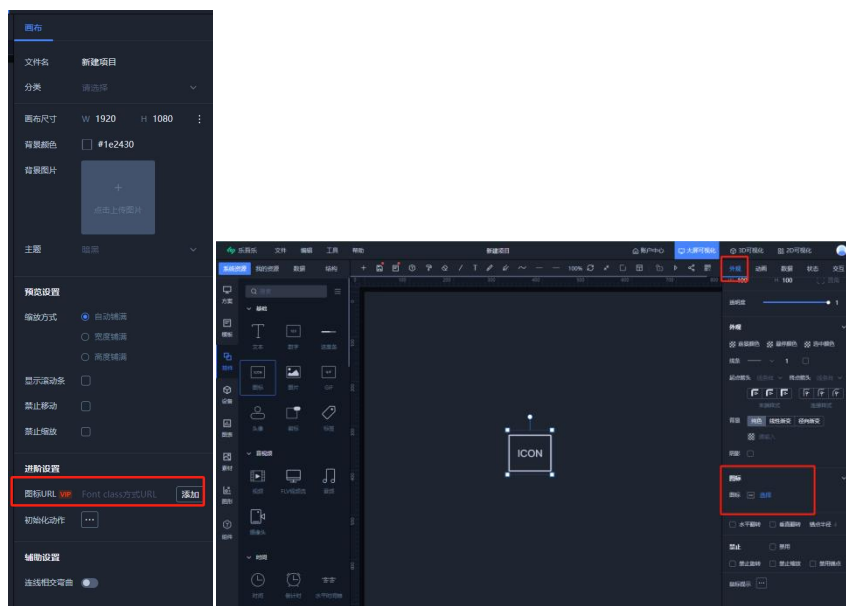
```

You, 3 months ago • feat:js线性图元

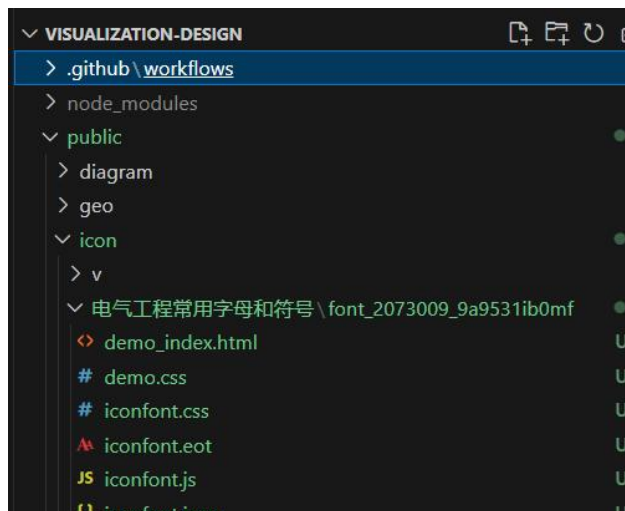
3. Icon 图形库

这个文件夹下的国家电网和电气工程图库，大屏已经不再使用，已经被 svg 图形库替代。但是客户需要的话，可以将 icon 下文件放到公共资源目录，然后在大屏可视化编辑器-点击画布-进阶设置-图标 URL 中，直接添加 iconfont.css 地址即可。

从控件中可以拖拽一个图标图元，右侧-外观-图标-可选择配置好的图标库。



例如：
图形库放到



图标 URL 中填写的地址就是：
/icon/电气工程常用字母和符号/font_2073009_9a9531b0mf/iconfont.css

3.5 图表

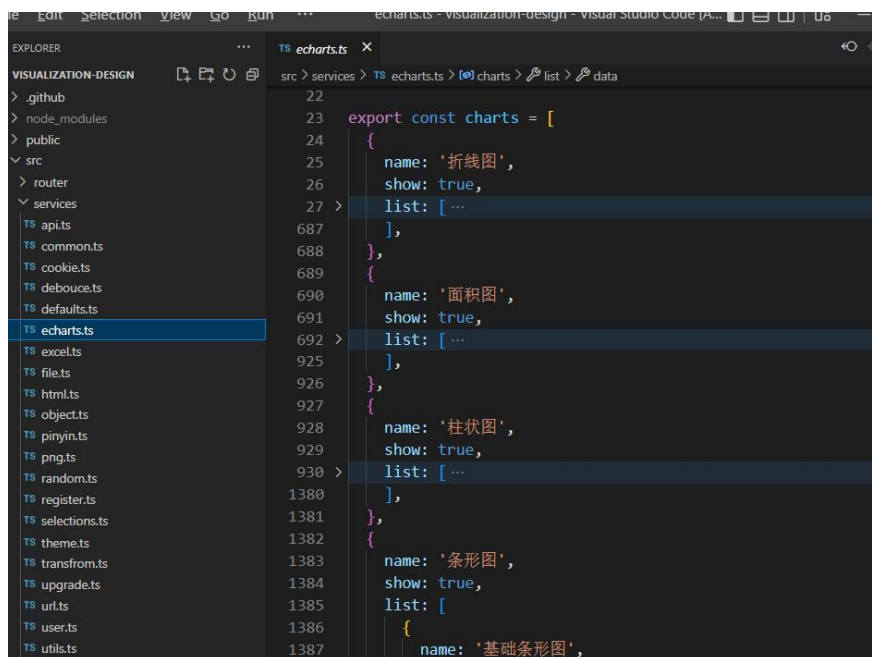
图表主要包括 echarts 图表和乐吾乐图表
图形库使用对应的文档介绍：

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

```
export function registerTheme() {
  fetch((cdn ? cdn + '/v/' : '') + 'theme/dark.json')
    .then((r) => r.json())
    .then((theme) => {
      echarts.registerTheme('le-dark', theme);
    });
}

export function changeTheme(theme: string) {
  charts.forEach((item) => {
    item.list.forEach((chart) => {
      if (chart.data.name === 'echarts') {
        chart.data.echarts && (chart.data.echarts.theme = theme);
      }
    });
  });
}
```

3.6 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，对应交付文件中 “素材” 文件夹。

乐吾乐大屏可视化系统V1.0交付文件 > 09 素材 >

名称	修改日期
标题	2024/8/2
常用图标	2024/8/2
电信机房	2024/8/2
面板	2024/8/2
智慧楼宇	2024/8/2
智能家居	2024/8/2
装饰	2024/8/2

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'



```
break;
case '素材': // Alsmile, 15 months ago via PR #1 * 场景
  groupType.value = 1;
  if (!materials.length) {
    loading.value = true;
    materials.push(...(await getFolders('v/material')));
    loading.value = false;
    const _materials: string = await localforage.getItem(
      'leSlev-materials'
    );
    if (_materials) {
      Object.assign(materials, JSON.parse(_materials));
    }
  }
  subGroups.value = materials;
  lastName = name;
  break;
case '设备':

5
6 export async function getFolders(name: string, isSvg?: boolean) { // A
7   const path = name;
8   const folders: any = await axios.post('/api/assets/folders', {
9     path,
10  });
11  if (!folders || !folders.list) {
12    return [];
13  }
14
15  const files: any = await axios.post('/api/assets/files', {
16    path,
17  });
18 }
```

将素材下的文件夹及其文件夹下资源放到服务器 `app_web_assets_root`（查看后端使用手册）目录下的 `/v/material` 文件夹下即可。前端 `path` 参数传入“`v/material`”，后端接口就会返回 `app_web_assets_root` 下的 `/v/material` 下文件夹及文件名。

3.7 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：

```
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 }
```

3.8 组件

通过/api/data/v.component/list 接口请求，通过 system 表示是系统组件。

```
break;
case '组件':
  if (activeAssets.value === 'system') {
    if (!componentCaches.length) {
      loading.value = true;
      componentCaches.push(...(await getCaseProjects('系统组件', true, false)));
      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);
        }
      }
    }
  }
  groupType.value = 1;
  subGroups.value = components;
  lastName = name;
```

3.9 我的资源

3.9.1 方案

```
//用户方案
subGroups.value = await getCollectionImageList('方案', 'v', false, false);
moveGroups['方案'] = [];
subGroups.value.forEach((item) => { if (item.name !== '默认') { moveGroups['方案'].push({ name: item.name }); });
groupType.value = 1;
userLastName = name;
}
```

1. /api/directory/list 获取文件夹列表
2. /api/data/v/list 获取用户所有图纸数据，system 不传，template 为 false
3. 再将图纸数据放入对应到文件夹

```
//获取方案文件夹
const getCollectionImageList = async (name?: string, collection?: string, system?: boolean, template?: boolean) => {
  //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) { ...
  }
  let list = [];
  for (let i of ret.list) { ...
  }
  const data = { ...
  };
  if (!data.system) { ...
  }
  const config = {
    params: {
      current: 1,
      pageSize: 1000,
    },
  };
  //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) { ...
  }
  for (const item of res.list) { ...
  }
  for (const item of list) { ...
  }
  results.push({
    name: '默认',
    list: resultsMap['默认'],
  });
};
```

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

```
// userLastName = name;
subGroups.value = await getCollectionImageList('模板', 'v', false, true);
moveGroups['模板'] = [];
subGroups.value.forEach((item) => { if (item.name !== '默认') { moveGroups['模板'].push({ name: item.name }); });
groupType.value = 1;
userLastName = name;
```

同方案, collection 参数对应 '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，需使用到乐吾乐 3d 可视化，可以咨询商务购买。

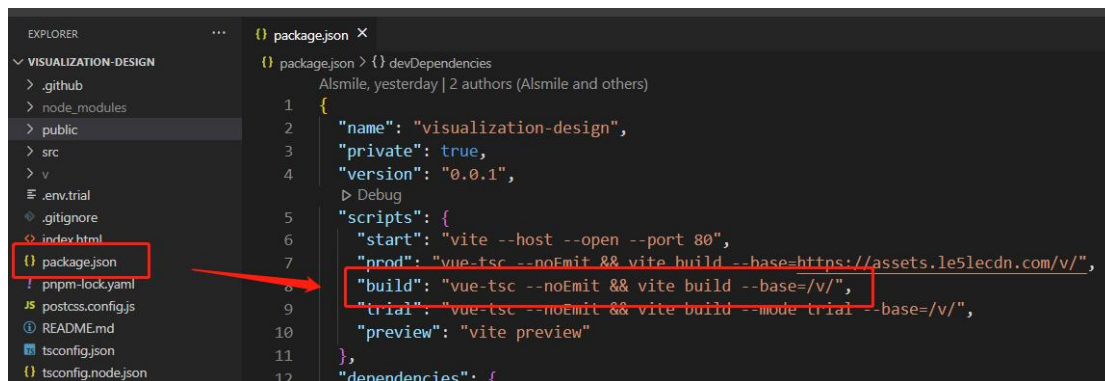
```

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：

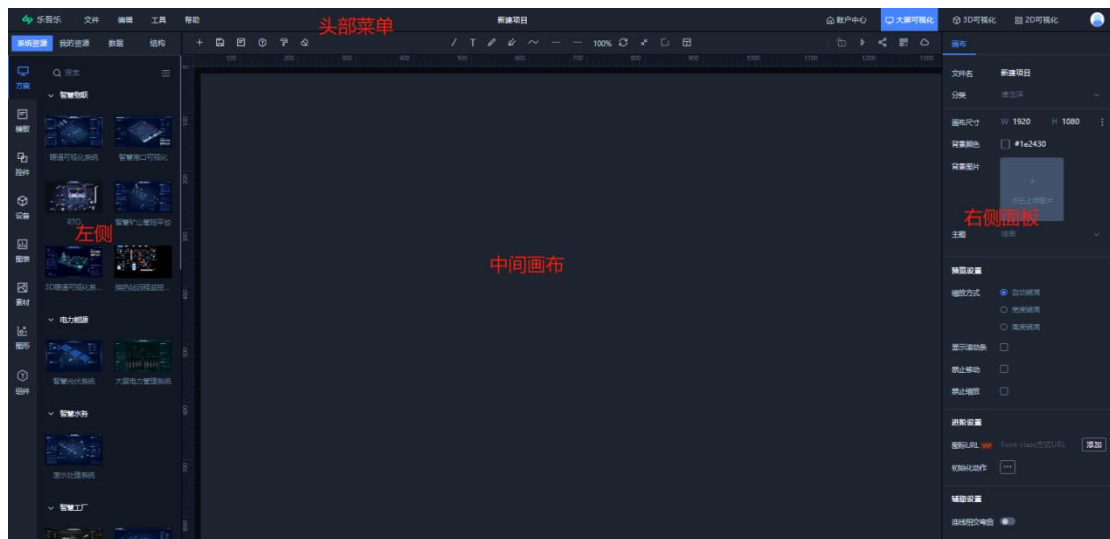
```

You, last month | 3 authors (Alsmile and others)
<template>
  <div class="app-page" @contextmenu.prevent>
    <Header />
    <div class="design-body">
      <Graphics />
      <View />
      <div style="border-left: 1px solid var(--color-border);">
        <FitProps v-if="selections.mode === SelectionMode.Fit" />
        <FileProps v-else-if="selections.mode === SelectionMode.File" />
        <PenProps v-else-if="selections.mode === SelectionMode.Pen" />
        <PensProps v-else />
      </div>
    </div>
  </div>
</template>

```

头部菜单
 左侧 (系统资源、我的资源、数据、结构)
 中间主画布
 自适应布局
 单击画布
 单选图元
 多选图元

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="group-asset">
  <t-radio-group
    v-model="activeAssets"
    @change="assetsChange"
    variant="primary-filled"
  >
    <t-radio-button value="system">系统资源</t-radio-button>
    <t-radio-button value="user">我的资源</t-radio-button>
    <t-radio-button value="data">数据</t-radio-button>
    <t-radio-button value="structure">结构</t-radio-button>
  </t-radio-group>

```

切换不同的按钮，展示不同的内容。

2.1 系统资源/我的资源

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 (!activatedPanels[activatedGroup.value]) {
    return;
  }

  if (activatedPanels[activatedGroup.value].length) {
    activatedPanels[activatedGroup.value] = [];
  } else {
    activatedPanels[activatedGroup.value] = [];
    for (const item of subGroups.value) {
      activatedPanels[activatedGroup.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': ...
  }
}
```

2.2 数据

数据包含列表创建、获取和监听。对应 view/components/Data.vue 组件。



主要涉及到增删改查，对应的接口/api/data/datasource/xx 接口。

2.3 结构

结构可以查看当前大屏的完整的图元组成结构。对应 view/components/Structure.vue 组件。



通过监听内置的消息，去触发页面更新。

```
onMounted(() => {
  meta2d.on('opened', getTree);
  meta2d.on('add', getTree);
  meta2d.on('undo', getTree);
  meta2d.on('redo', getTree);
  meta2d.on('delete', getTree);
  meta2d.on('combine', getTree);
  meta2d.on('click', getActivated);
  meta2d.on('paste', getActivated);
  meta2d.on('layer', layerChange);
  meta2d.on('active', getActivated);

  if (inTreePanel.timer) {
```

可以点击定位、显示/隐藏、锁定图元，对应调用了核心库 `api`。

```

const onActive = (e,value: any) => {
  if (!value || value.endsWith('Layer')) {
    return;
  }

  if(e.ctrlKey){
    if(data.active.includes(value)){
      data.active = data.active.filter((item) => item !== value);
    }else{
      data.active.push(value);
    }
  }else{
    data.active = [value];
  }
  getActiveFlag = true;
  if(data.active.length > 1){
    let pens = [];
    data.active.forEach((item) => {
      pens.push(meta2d.store.pens[item]);
    });
    meta2d.active(pens, true);
  }else{
    const pen = meta2d.store.pens[value];
    meta2d.active([pen], true);
    if (!pen.calculative?.inView) {
      meta2d.gotoView(pen);
      meta2d.resize();
    }
  }
}

meta2d.render();
};

```

```

const lock = (node: any, v: LockState) => {
  node.data.locked = v;
  meta2d.setValue({
    id: node.value,
    locked: v,
  });
};

const visible = (node: any, v: boolean) => {
  node.data.visible = v;
  if (node.data.value.endsWith('Layer')) {
    if (node.data.value === 'imageLayer') {
      meta2d.canvas.canvasImage.canvas.style.display = v ? 'block' : 'none';
    } else if (node.data.value === 'mainLayer') {
      meta2d.canvas.canvas.style.display = v ? 'block' : 'none';
    } else if (node.data.value === 'imageBottomLayer') {
      meta2d.canvas.canvasImageBottom.canvas.style.display = v
        ? 'block'
        : 'none';
    } else if (node.data.value === 'templateLayer') {
      meta2d.canvas.canvasTemplate.canvas.style.display = v ? 'block' : 'none';
    }
  }
  return;
}

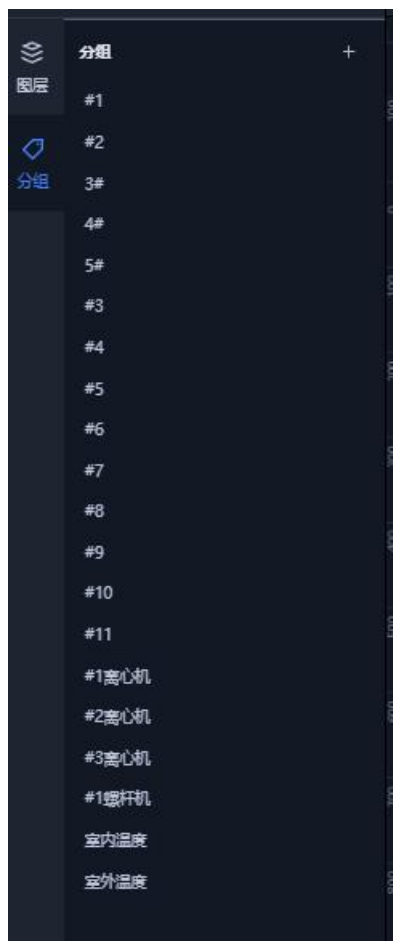
setChildrenVisible(node, v);
const pen = meta2d.findOne(node.value);
pen && meta2d.setVisible(pen, v);
meta2d.render();
};

```

锁定

显示/隐藏

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 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、自适应设置相关操作

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.
};
```

开启自适应布局模式后，会出现布局容器面板，具体操作说明见文档：

<https://doc.le5le.com/document/60>



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>
<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();
  };
});
```

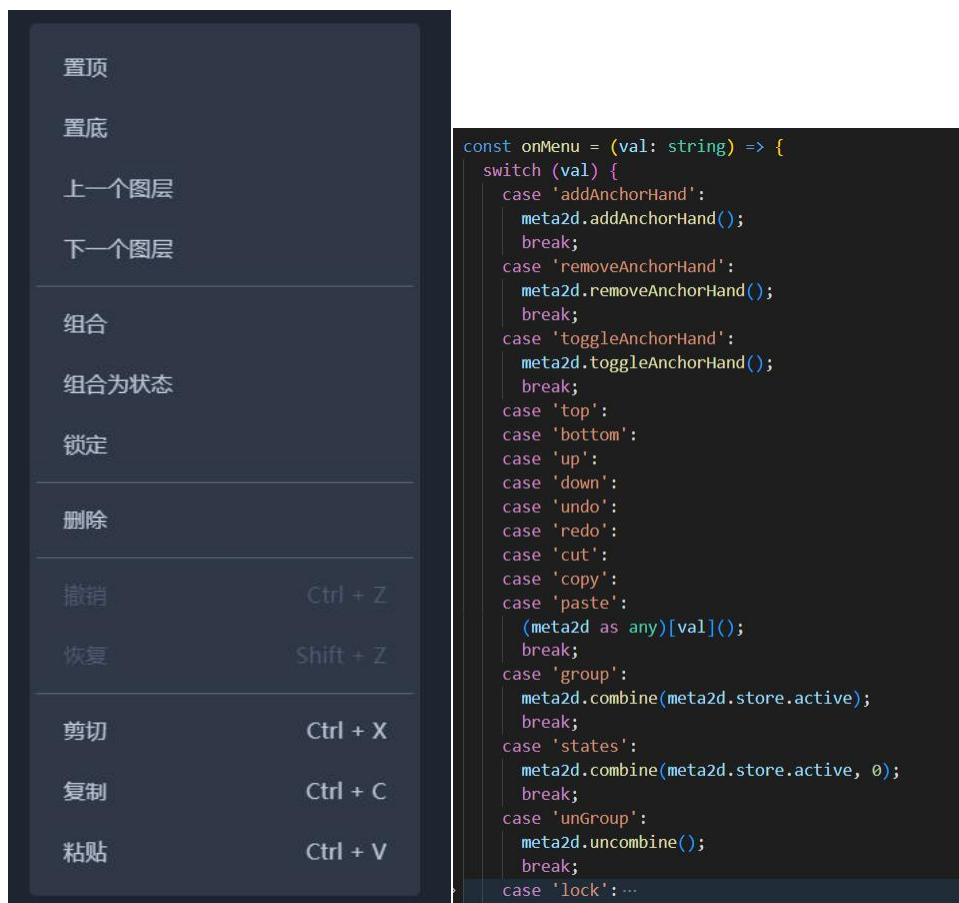
创建meta2d实例

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）

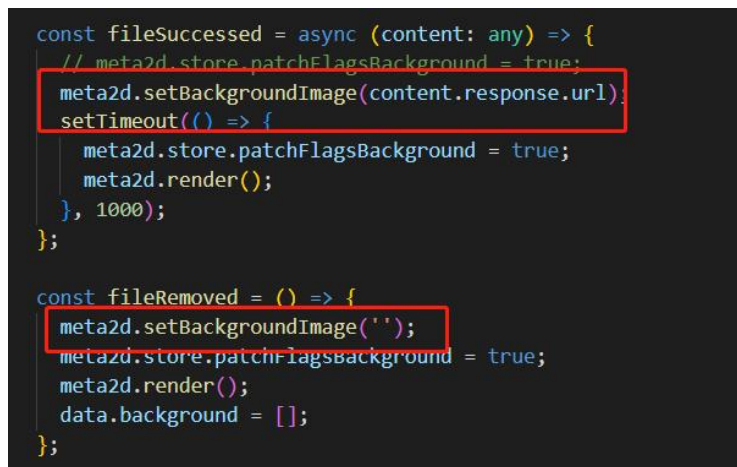


基本设置这一块，主要是通过修改 `meta2d.store.data` 下面的属性，具体说明可查看文档：

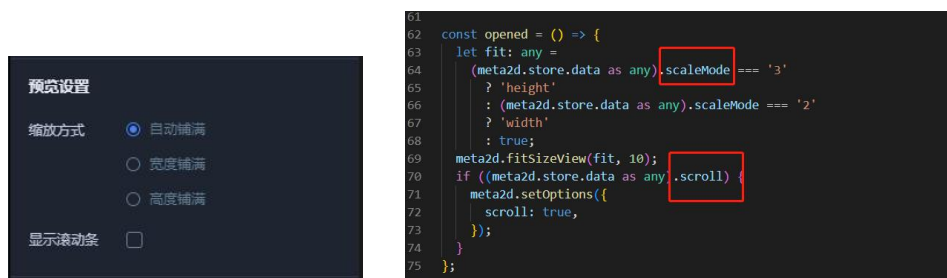
<https://doc.le5le.com/document/119882449#%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84>

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>

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。

```
<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" @tabchange="tabChange" />
  </t-tab-panel>
  <t-tab-panel :value="5" label="状态">
    <PenStatus :pen="data.pen" ref="status" />
  </t-tab-panel>
  <t-tab-panel :value="4" label="交互">
    <PenEvents :key="data.key" :pen="data.pen" />
  </t-tab-panel>
  <!-- <t-tab-panel :value="5" label="结构">
    <ElementTree />
  </t-tab-panel> -->
</t-tabs>
</div>
```

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

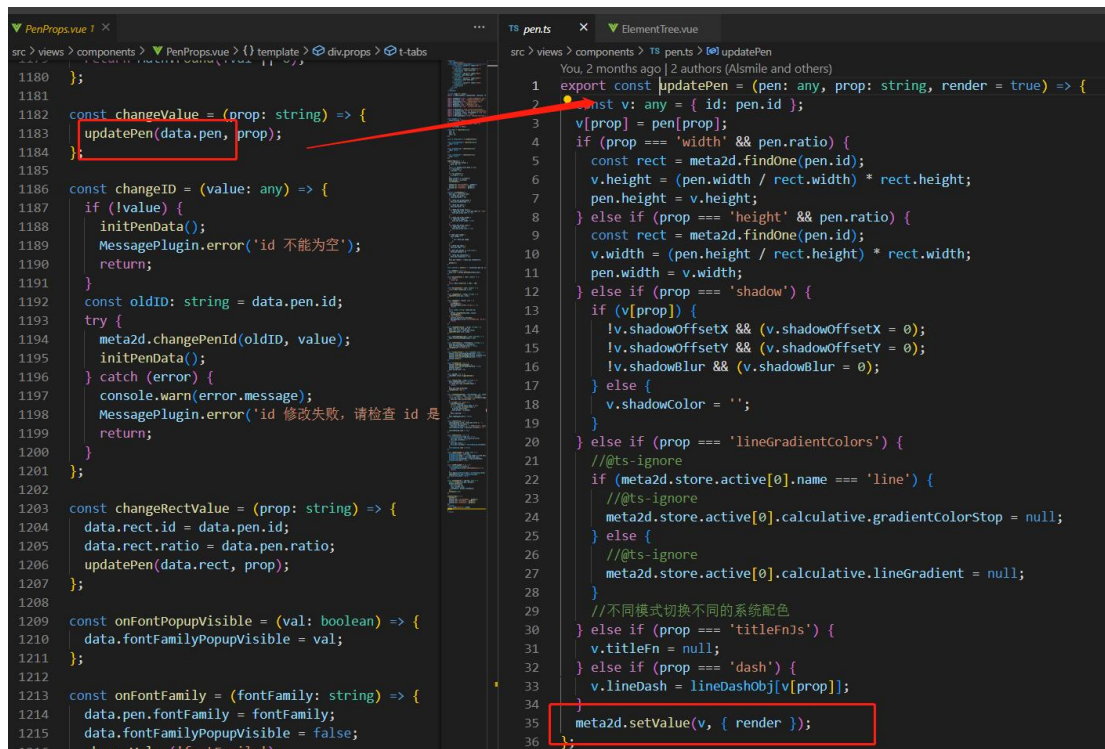
锚点半径 4

禁止旋转

禁止缩放

禁用锚点

鼠标提示





```

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>

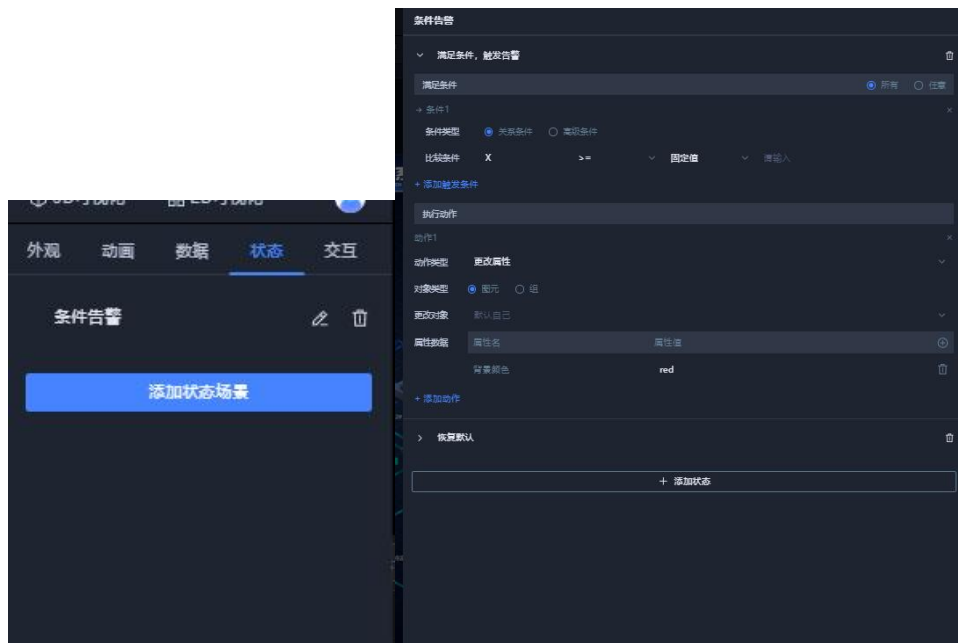
3. 数据（components/PenDatas.vue）

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. 状态（components/PenStatus.vue）

状态可以用作监听数据的值变化，做状态判断，执行动作行为。



5. 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>



5.1.3.3 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，设置分组，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) Meta2d.store: 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/diagram.js 图形库

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/views 主要的页面，首页和预览页面

Src/views/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

九 部署集成

因为大屏编辑器代码比较重，推荐直接独立部署，通过域名访问或者 iframe 嵌入。