

hxm-vscode-download-icon插件

Presentation slides for developers

Press Space for next page →



Background

Daily Work On Downloading Icon

- 打开 设计中台
- 搜索图标 -> 下载图标
- Repeat Step2
- 从杂乱的`Downloads`目录中找到刚刚下载的图标
- 将图标传输至内网
- 拷贝到项目的文件夹内
- Reapeat All



Crazy! 😣



hxmui什么时候能出一个icon组件，现在每次都是从icon平台找icon拉到内网非常麻烦

08-06 上午 10:47

沈佳棋0254



Icon组件其实线上倒是有。但是目前是把常用的十几个icon打包成了svg-symbols文件去使用的。把icon平台的icon都内置到组件库我觉得肯定不现实的。所以还是得想办法让使用者能够比较简单地去获取想要的icon然后生成svg-symbols在项目中引入。

已读 上午 10:53 08-06

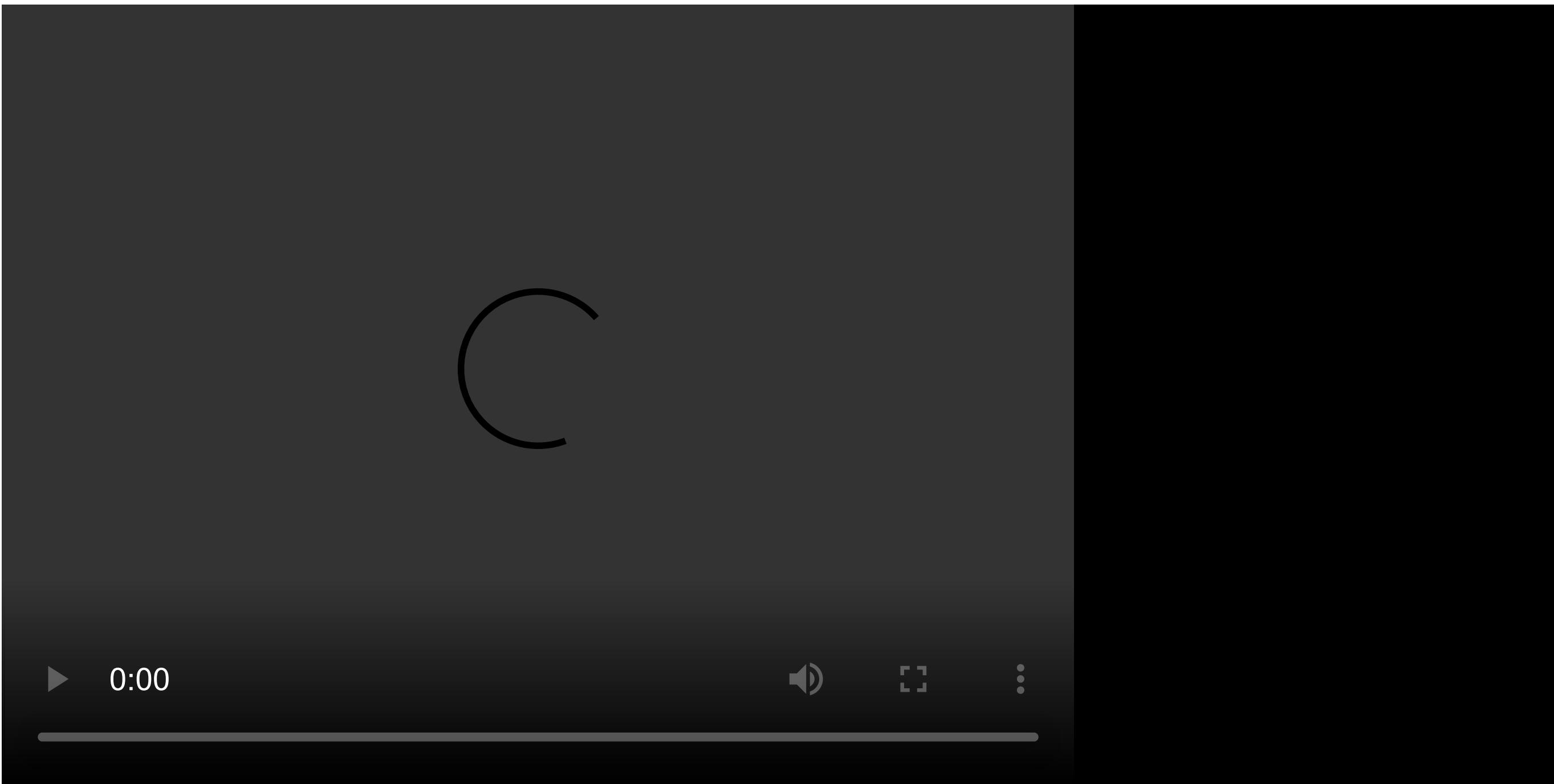


沈佳棋0254

沈佳棋

What is hxm-vscode-download-icon?

Demo



Main Feature



hxm-vscode-download-icon

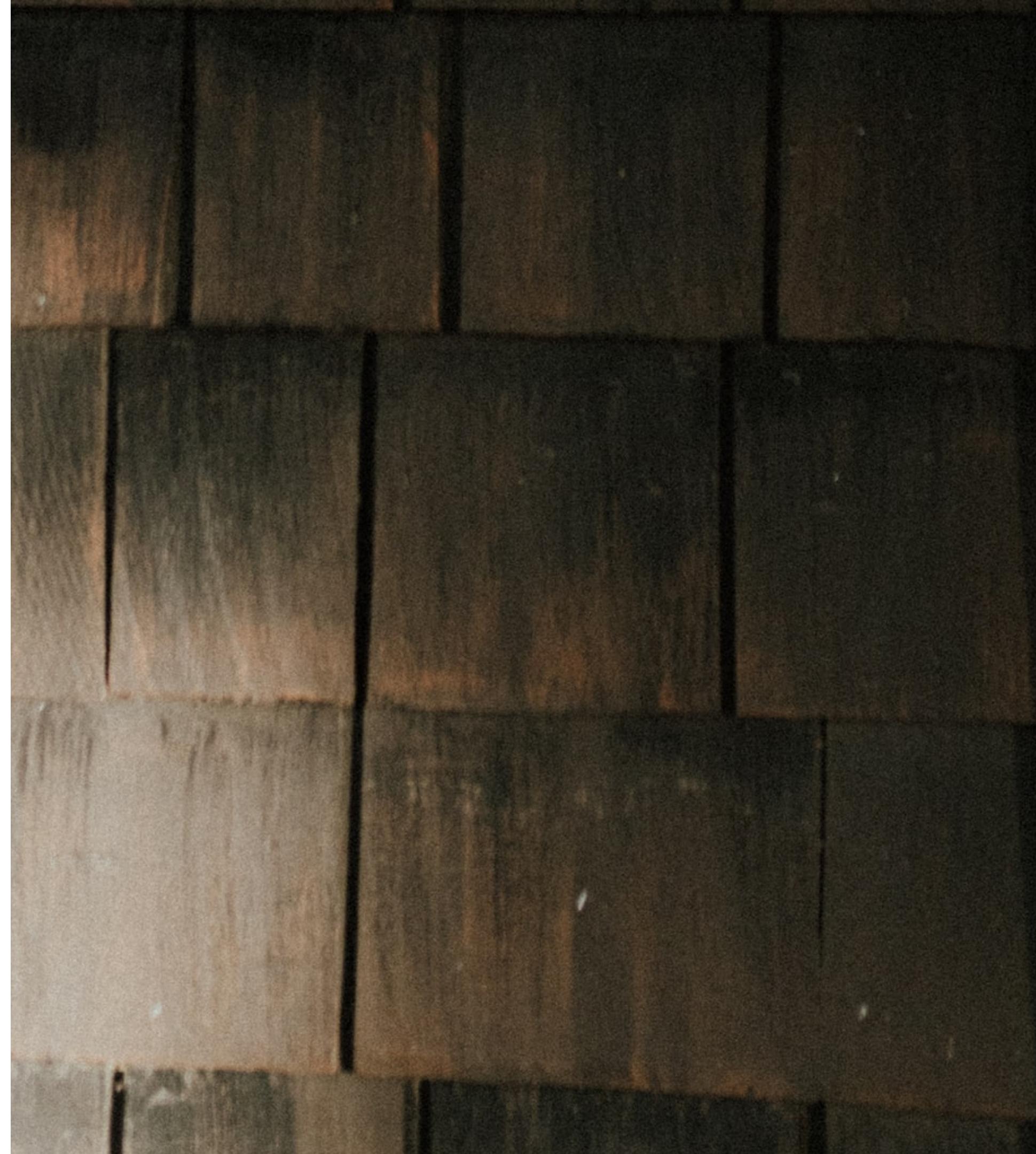
1. 将Icon平台功能集成于vscode插件内，随开随用
2. 支持通过图标英文名搜索后进行批量选择
3. 支持剔除已选择的图标
4. 支持批量导出svg文件夹
5. 支持批量打包输出为svg-symbols.js 结合hxm-icon组件直接使用
6. 支持缓存功能，下次打开页面的时候依然展示退出页面前的搜索及选择结果
7. 内外网通用

Plugin FrameWork

FrameWork

- 一、vscode插件基础框架搭建
- 二、侧边栏图标及目录树配置
- 三、在vscode中创建webview
- 四、核心功能 Core:
 - 工具选择
 - 搜索图标功能
 - 搜索图标列表展示与联动
 - 已选择图标列表展示
 - vscode插件与webview的通信
 - 数据缓存
 - 内外网兼容
- 五、插件发布

Plugin Todos: XMind文档下载地址



Code Implements

1. VSCode插件基础教程

2. 侧边栏配置

2.1 package.json 配置

```
{  
  // 注册事件  
  "activationEvents": [  
    "onView: download_svg",  
  ],  
  // 注册视图  
  "contributes": {  
    "viewsContainers": {  
      "activitybar": [  
        {  
          "id": "hxmui",  
          "title": "SVG图标管理平台"  
          "icon": "resources/sideIcon.svg"  
        }  
      ]  
    },  
    "views": {  
      "hxmui": [  
        {  
          "id": "download_svg",  
          "name": "同花顺Hux平台图标下载"  
        },  
      ]  
    }  
  }
```

2.2 sidebarTree.js 目录树配置

```
module.exports = class SideBarTree {
    // 类中必须要实现的两个接口
    getTreeItem(element) {
        return element;
    }
    getChildren() {
        // 在这里面注册children
        const r_cates = [
            {
                title: 'Home',
                icon: 'home.svg'
            }
        ];
        const fin_items = [];
        for (let i = 0; i < c_rates.length; i++) {
            fin_items.push(
                new DataItem(
                    /** 实例化 */
                )
            )
        }
    }
}

class DataItem extends vscode.TreeItem {}
```

2.3 usage: extension.js

```
function active(context) {
    vscode.window.registerTreeDataProvider('download_svg', new SidebarTree(context));

    let openSite = vscode.commands.registerCommand('download_svg.openSite', () => {
        /**
         * 注册完事件的回调
         * 这里可以创建webview
         */
    })
}
```

3. 创建webview

3.1 注册webview panel

```
const panel = vscode.window.createWebviewPanel(  
    'webview',  
    '同花顺Hxmui-Icon',  
    vscode.ViewColumn.One,  
    {  
        enableScripts: true,  
        retainContextWhenHidden: true  
    }  
);  
  
panel.iconPath = vscode.Uri.file(join(__dirname, 'resources', 'home.svg'));  
panel.webview.html = getWebViewContent(context, 'views/index.html');
```

`window.createWebviewPanel`

3.2 getWebViewContent 方法

```
function getWebViewContent(context, templatePath) {
    const resourcePath = join(context.extensionPath, templatePath);
    const dirPath = dirname(resourcePath);
    let html = fs.readFileSync(resourcePath, 'utf-8');

    html = html.replace(/(<link.+?href=)|(script.+?src=)|(<img.+?src=)(.+?)/g, (m, $1, $2) => {
        if ($2.indexOf("https://") < 0) {
            return $1 + vscode.Uri.file(resolve(dirPath, $2)).with({ scheme: 'vscode-resource' }).toString() + '';
        } else {
            return $1 + $2 + '';
        }
    });

    return html;
}
```

4. VS Code插件与Webview的通信

场景：用户在ui界面点击按钮的时候，文件/系统层面的工作需要由vscode去处理

- 生成svg-symbols.js
- 生成svg文件夹
- 设置本地缓存
- 获取本地缓存
- 清除本地缓存

webview中触发事件调用`vscode.postMessage`发送事件通知和参数

index.html

```
vscode.postMessage({  
  command: 'exportSvgSymbolsFile',  
  data: this.choosedList  
})
```

vscode插件一侧调用`panel.webview.onDidReceiveMessage`监听webview事件

extension.js

```
panel.webview.onDidReceiveMessage(  
  async message => {  
    switch(message.command) {  
      case 'exportSvgSymbolsFile':  
        break;  
      case 'exportSvgFile':  
        break;  
      // 其他事件...  
    }  
  }  
)
```

5. 生成svg-symbols

生成svg-symbols

1. `vscode.window.showSaveDialog` 调起保存弹窗，获取保存路径
2. 根据 `webview` 中返回的svg数组生成svg文件夹
3. 使用 `gulp-svg-symbols` 和 `gulp-svg-symbols2js` 输出 `svg-symbols.js`

```
function genSvgSymbolsJs(svgPath, outPath, callback = () => {}) {  
    removeSync(outPath);  
    return gulp  
        .src(`.${svgPath}/*.svg`)  
        .pipe(svgSymbols())  
        .pipe(svgSymbols2js())  
        .pipe(gulp.dest(`.${outPath}`))  
        .on('end', () => { callback() })  
}
```

6. 本地缓存

这里缓存用的是lowdb

1. 初始化

```
const adapter = new FileSync(join(__dirname, 'db.json'));
const db = low(adapter);

db.data = { posts: {} }; // 设置默认值
```

2. 设置缓存

```
db.data.posts[message.key] = message.value;
```

3. 获取缓存 - 这里涉及到和webview的通信

```
panel.webview.postMessage({ command: 'onGetStorageSuccess', data: db.data.posts[message.key] });
```

4. 清除缓存

```
db.data.posts[message.key] = '';
```

7. 插件发布

插件发布的两种方式

- vsce package 发布vsix本地包
- vsce publish 发布到应用市场

有个坑需要注意一下：

当我们在`package.json`中配置了`icon`属性，即应用的图标，但是没有指定repository仓库时，发布的命令需要增加额外的参数：

```
vsce publish --baseContentUrl https://none/ --baseImagesUrl https://none/
```

```
vsce package --baseContentUrl https://none/ --baseImagesUrl https://none/
```

References



参考资料

- [vscode官方文档](#)
- [vscode webview](#)
- [vscode TreeView](#)
- [vscode插件与webview相互通信](#)
- [lowdb 仓库](#)
- [gulp-svg-symbols 仓库](#)
- [gulp-svg-symbols2js 仓库](#)
- [VS Code 插件开发入门教程](#)
- [VSCode插件开发全攻略](#)
- [VSCode WebView插件（扩展）开发实战](#)
- [小霸王插件 仓库](#)
- [我爱掘金插件 仓库](#)

Thank You!