

Ubuntu 22.04
CPU x86_64
Electron **git** **Electron**

aki

1 git aki

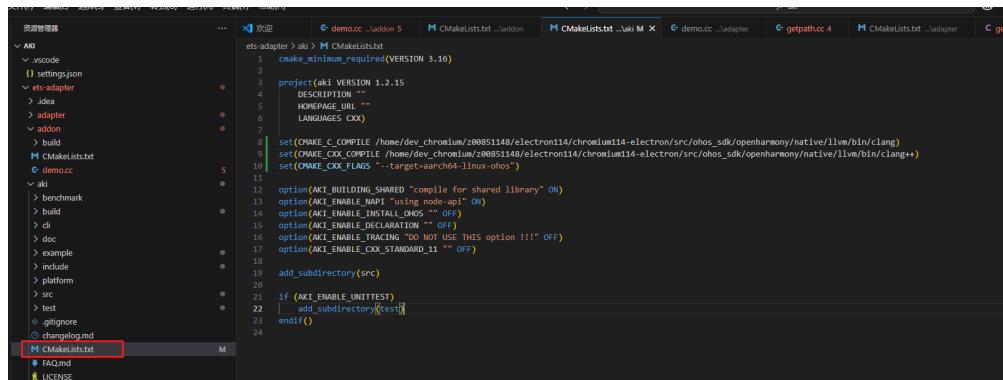
```
git clone https://gitee.com/openharmony-sig/aki.git
```

2 cmake

```

aki      cmake           clang  clang++
set(CMAKE_C_COMPILE /home/dev_chromium/z00851148/electron114/chromium114-electron/src/ohos_sdk/openharmony/native/llvm/bin/clang)
set(CMAKE_CXX_COMPILE /home/dev_chromium/z00851148/electron114/chromium114-electron/src/ohos_sdk/openharmony/native/llvm/bin/clang)
set(CMAKE_CXX_FLAGS "-target=aarch64-linux-ohos")

```



3 clang clang++

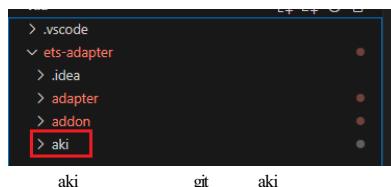
```
export CC="/home/dev_chromium/z00851148/electron114/chromium114-electron/src/ohos_sdk/openharmony/native/llvm/bin/clang --target=a  
export CXX="/home/dev_chromium/z00851148/electron114/chromium114-electron/src/ohos_sdk/openharmony/native/llvm/bin/clang++ --targe
```

4

```
build/src libaki_jsbind.so  
cd aki  
mkdir build  
cd build  
cmake .. -DCMAKE_BUILD_TYPE=Release  
make
```

adapter

1



2. adapter demo

```

// 欢迎 X demo.cc ..\addon M CMakeLists.txt M demo.cc ..\adapter X getPatch.cc
ets-adapter > adapter > demo.cc > getPatch.cc > getNativeContext(napi_env, napi_callback_info)
1 #include "third_party/electron_node/src/node.h"
2
3 #include "openharmony/native/sysroot/usr/include/napi/native_api.h"
4 #include "openharmony/native/sysroot/usr/include/js_native_apitypes.h"
5 #include "aki/jbind.h"
6 #include <string>
7 #include <cstring>
8 #define DECLARE_NAPI_METHOD(name, func) { name, 0, func, 0, 0, 0, napi_default, 0 }
9
10 using v8::FunctionCallbackInfo;
11 using v8::Isolate;
12 using v8::Local;
13 using v8::Message;
14 using v8::Object;
15 using v8::String;
16 using v8::Value;
17
18 static napi_value getNativeContext(napi_env env, napi_callback_info info) {
19     napi_value exports;
20     if (napi_create_object(env, &exports) != napi_ok) {
21         napi_throw_type_error(env, nullptr, "napi_create_object failed");
22         return exports;
23     }
24
25     aki::JSBind::BindSymbols(env, exports);
26     return exports;
27 }
28
29
30 static napi_value Initialize(napi_env env, napi_value exports) {
31     if (env == nullptr || exports == nullptr) {
32         return exports;
33     }
34     napi_property_descriptor desc[] = {
35         DECLARE_NAPI_METHOD("getNativeContext", getNativeContext)
36     };
37     if (napi_define_properties(env, exports, sizeof(desc) / sizeof(desc[0]),
38                               desc) != napi_ok) {
39         return exports;
40     }
41     return exports;
42 }
43
44 static napi_module AdapterTestModule = {
45     .nm_version = 1,
46     .nm_flags = 0,
47     .nm_filename = nullptr,
48     .nm_register_func = Initialize,
49     .nm_modname = "AdapterTest",
50     .nm_rev = ((void*)nullptr),
51     .nm_reserved = (void*)nullptr,
52 };
53
54 extern "C" __attribute__((constructor)) void RegisterAdapterModule(void) {
55     napi_module_register(&AdapterTestModule);
56 }

```

3. getPatch.cc

```

// 欢迎 X demo.cc ..\addon M CMakeLists.txt M demo.cc ..\adapter X getPatch.cc 4 C getPatch.h 2
ets-adapter > adapter > demo.cc > getPatch.cc > PathAdapter > GetDir(std::string&)
1 // 同步方式
2 if (auto getDirFunc = aki::JSBind::GetJSFunction(getDirFuncName)) {
3     auto path = getDirFunc->Invoke<std::string>();
4     return path;
5 }
6
7 return "";
8
9 // 异步方式
10 // std::promise<std::string> promise;
11 // std::function<void(std::string)> callback = [&promise](std::string path) {
12 //     promise.set_value(path);
13 // };
14
15 // if (auto jsFunc = aki::JSBind::GetJSFunction(getDirFuncName)) {
16 //     jsFunc->Invoke<void>(callback);
17 // } else {
18 //     return "";
19 // }
20
21 // auto future = promise.get_future();
22 // auto status = future.wait_for(std::chrono::seconds(5));
23 // if (status == std::future_status::timeout) {
24 //     return "timeout";
25 // }
26
27 // return future.get();
28 }
29
30
31
32
33
34
35

```

4. CMakeLists

```

# Project Name
SET(TARGETFILENAME "adaptertest")
PROJECT(${TARGETFILENAME})

# CMake minimum version requirement setting
cmake_minimum_required(VERSION 3.8)

# set electron path
set(ELE_PATH /home/dev_chromium/z00851148/electron114/chromium114-electron)

# ohos
set(CMAKE_C_COMPILER ${ELE_PATH}/src/ohos_sdk/openharmony/native/llvm/bin/clang)
set(CMAKE_CXX_COMPILER ${ELE_PATH}/src/ohos_sdk/openharmony/native/llvm/bin/clang++)
set(CMAKE_CXX_FLAGS "--target=aarch64-linux-ohos")

#
# ohos
include_directories(${ELE_PATH}/src/)
include_directories(${ELE_PATH}/src/third_party/electron_node/deps/v8/include/)
include_directories(${ELE_PATH}/src/ohos_sdk/)
include_directories(${CMAKE_CURRENT_SOURCE_DIR}../aki/include)

message(${CMAKE_CURRENT_SOURCE_DIR})

set(src demo.cc getpatch.cc)
set(headers getpatch.h)

#
#add_executable(${TARGETFILENAME} demo.cpp)

#
#ADD_LIBRARY(${TARGETFILENAME} STATIC demo.cpp)

#

```

```

ADD_LIBRARY(${TARGETFILENAME} SHARED
    ${src}
)
target_compile_features(${TARGETFILENAME} PUBLIC cxx_std_17)
target_compile_definitions(${TARGETFILENAME} PUBLIC JSBIND_USING_NAPI=1)
target_compile_definitions(${TARGETFILENAME} PUBLIC AKI_BUILDING_SHARED=1)

#           -ohos
target_link_libraries(${TARGETFILENAME} PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}/../aki/build/src/libaki_jsbind.so)
target_link_libraries(${TARGETFILENAME} PUBLIC ${ELE_PATH}/src/out/musl_64/libelectron.so)

#           .node
# SET_TARGET_PROPERTIES(${TARGETFILENAME} PROPERTIES SUFFIX ".node")

```

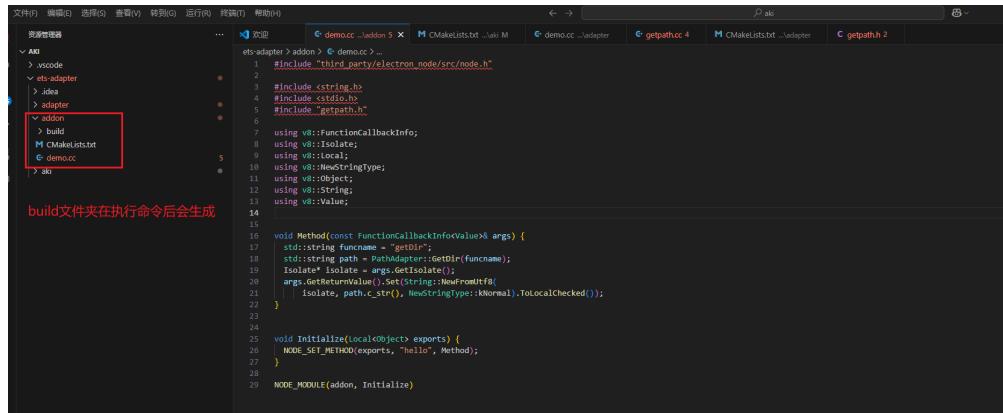
5.

libadaptertest.so

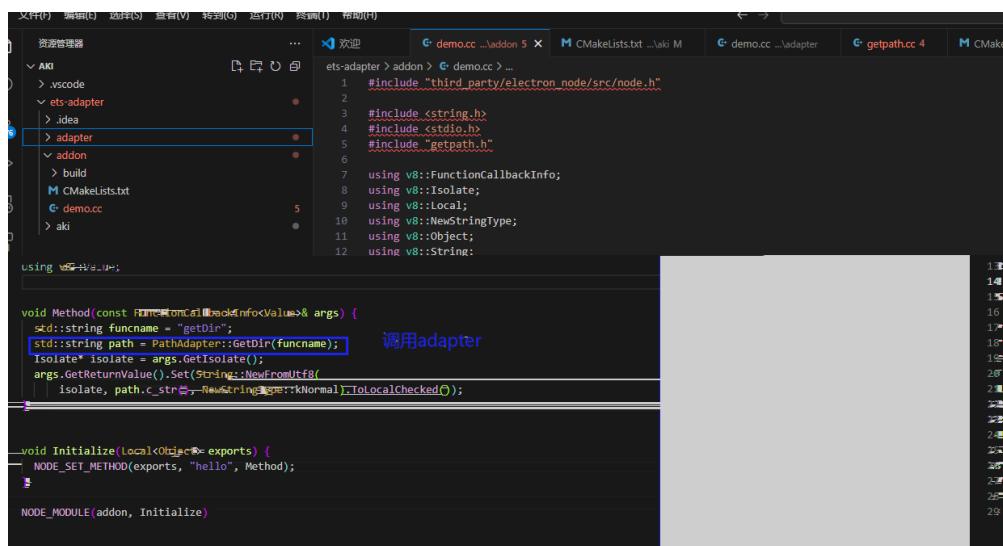
```
mkdir build  
cd build  
cmake ../  
make
```

addon

1



2 adapter



3 CMakeLists

```
# Project Name
SET(TARGETFILENAME "addon")
PROJECT(${TARGETFILENAME})

# CMake minimum version requirement setting
cmake_minimum_required(VERSION 3.8)

# set electron path
set(ELE_PATH /home/dev_chromium/z00851148/electron114/chromium114-electron)

# ohos
set(CMAKE_C_COMPILER ${ELE_PATH}/src/ohos_sdk/openharmony/native/llvm/bin/clang)
set(CMAKE_CXX_COMPILER ${ELE_PATH}/src/ohos_sdk/openharmony/native/llvm/bin/clang++)
set(CMAKE_CXX_FLAGS "--target=aarch64-linux-ohos")

#
# ohos

include_directories(${ELE_PATH}/src/)
include_directories(${ELE_PATH}/src/third_party/electron_node/deps/v8/include/)
```

```

include_directories(${ELE_PATH}/src/ohos_sdk/)
include_directories(${CMAKE_CURRENT_SOURCE_DIR}/../adapter/)

message(${CMAKE_CURRENT_SOURCE_DIR})
set(src demo.cc)
#
#add_executable(${TARGETFILENAME} demo.cpp)
#
#
#ADD_LIBRARY(${TARGETFILENAME} STATIC demo.cpp)

#
ADD_LIBRARY (${TARGETFILENAME} SHARED
${src}
)
#
#           -ohos
target_link_libraries(${TARGETFILENAME} PUBLIC ${ELE_PATH}/src/out/musl_64/libelectron.so)
target_link_libraries(${TARGETFILENAME} PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}/../adapter/build/libadaptertest.so)
target_compile_definitions(${TARGETFILENAME} PUBLIC NODE_MODULE_VERSION=116)

#
#           .node
SET_TARGET_PROPERTIES(${TARGETFILENAME} PROPERTIES SUFFIX ".node")

```

4

```

addon.node

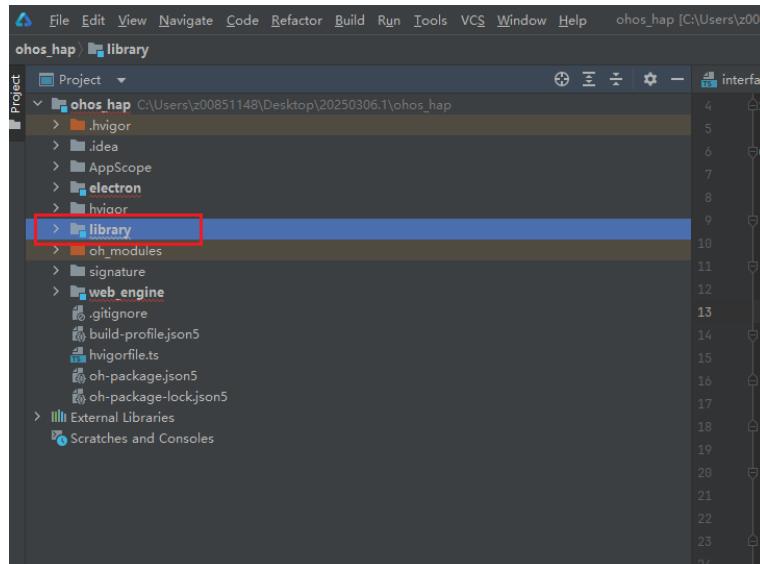
mkdir build
cd build
cmake ../
make

```

ets har

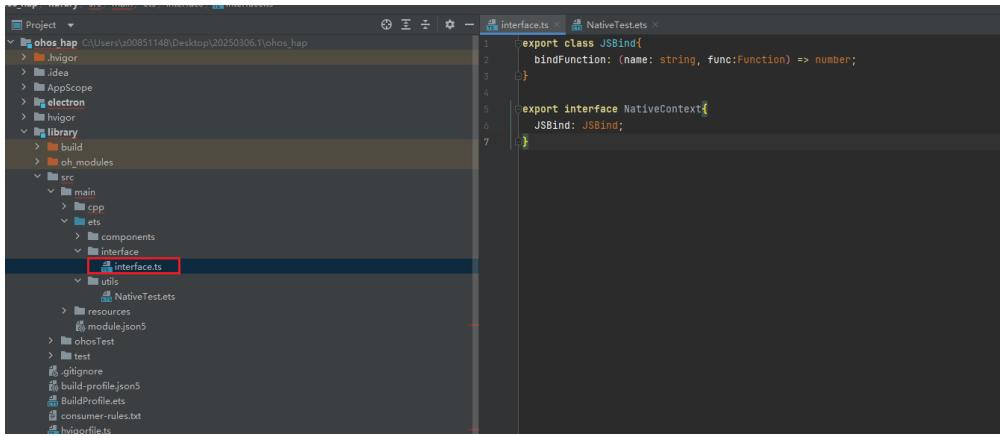
1

deveco File ==> New ==> Moudle ==> Static Library Library

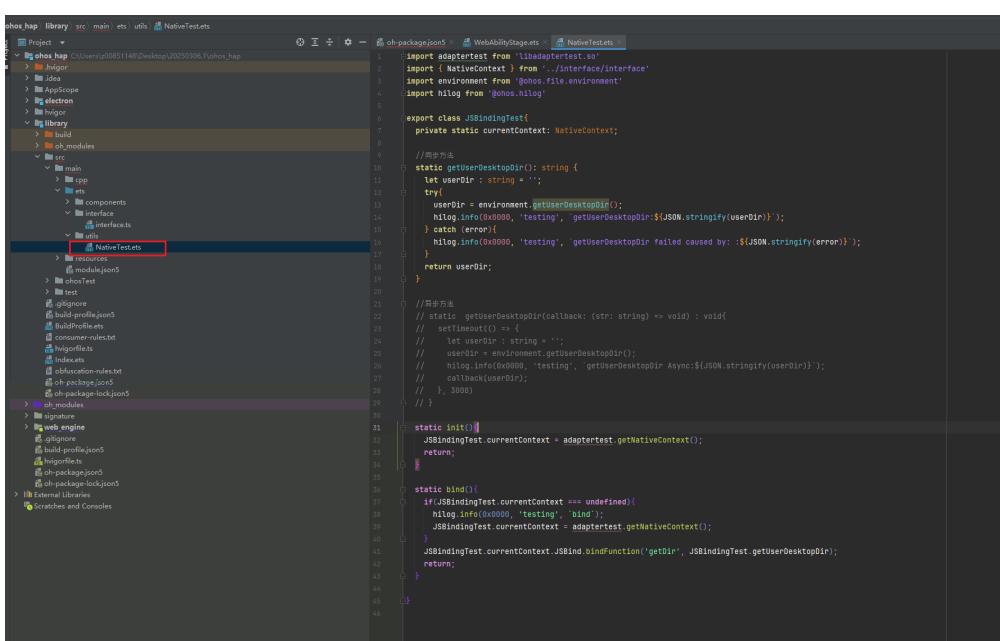


2 adapter

2.1 adapter



2.2 bind



```

import adaptertest from 'libadaptertest.so'
import { NativeContext } from '../interface/interface'
import environment from '@ohos.file.environment'
import hilog from '@ohos.hilog'

export class JSBindingTest{
  private static currentContext: NativeContext;

  // 同步方法
  static getUserDesktopDir(): string {
    let userDir : string = '';
    try{
      userDir = environment.getUserDesktopDir();
      hilog.info(0x0000, 'testing', `getUserDesktopDir:${JSON.stringify(userDir)})`);
    } catch (error){
      hilog.info(0x0000, 'testing', `getUserDesktopDir failed caused by: ${JSON.stringify(error)})`);
    }
    return userDir;
  }

  // 异步方法
  static getUserDesktopDir(callback: (str: string) => void) : void{
    // setTimeout(() => {
    //   let userDir : string = '';
    //   userDir = environment.getUserDesktopDir();
    //   hilog.info(0x0000, 'testing', `getUserDesktopDir Async:${JSON.stringify(userDir)})`);
    //   callback(userDir);
    // }, 3000)
    // }

    static init(){
      JSBindingTest.currentContext = adaptertest.getNativeContext();
      return;
    }

    static bind(){
      if(JSBindingTest.currentContext === undefined){
        hilog.info(0x0000, 'testing', `bind`);
        JSBindingTest.currentContext = adaptertest.getNativeContext();
      }
    }
}

```

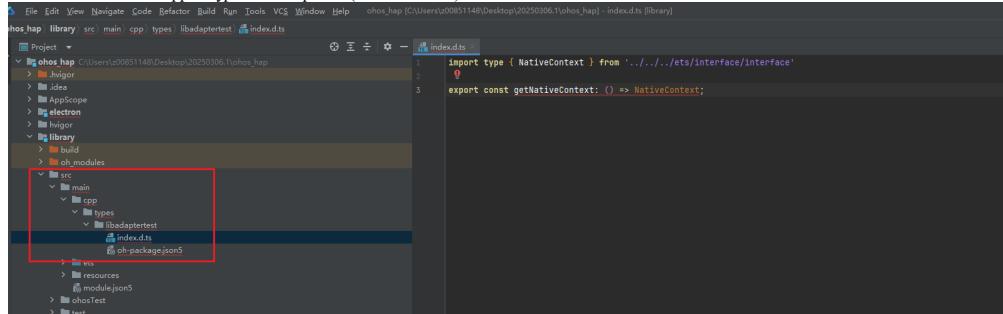
```

        }
        JSBindingTest.currentContext.JSBind.bindFunction('getDir', JSBindingTest.getUserDesktopDir);
        return;
    }
}

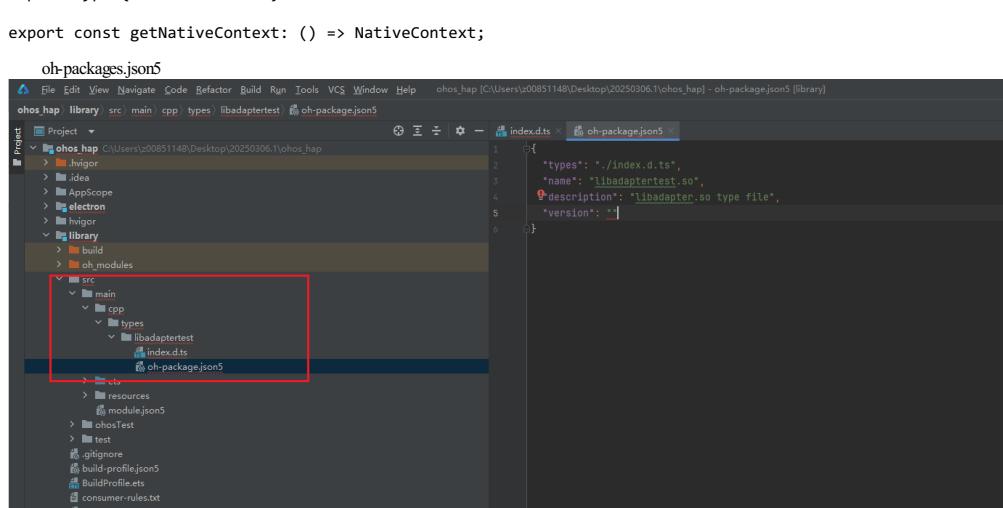
```

2.2.3

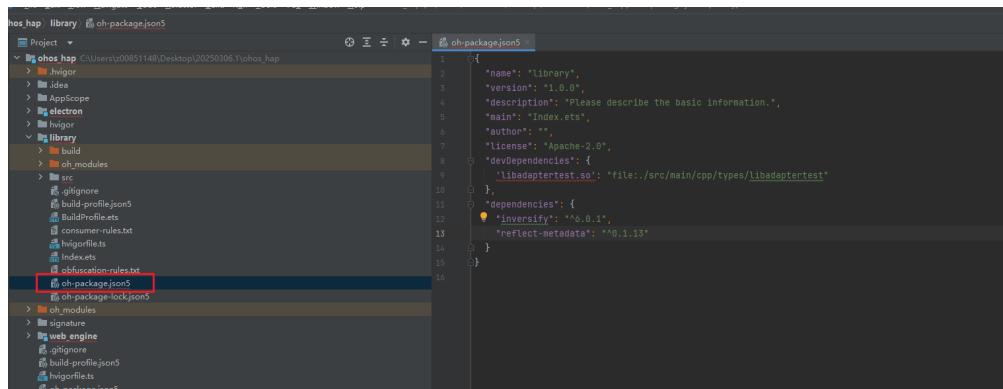
src/main cpp -> types ->libadAPTERtest()



oh-packages.json5



2.2.4 library oh-packages.json5



2.2.5 library Index.ets

```

export { MainPage } from './src/main/ets/components/MainPage';
export { JSBindingTest } from './src/main/ets/utils/NativeTest'

```

2.2.6 har

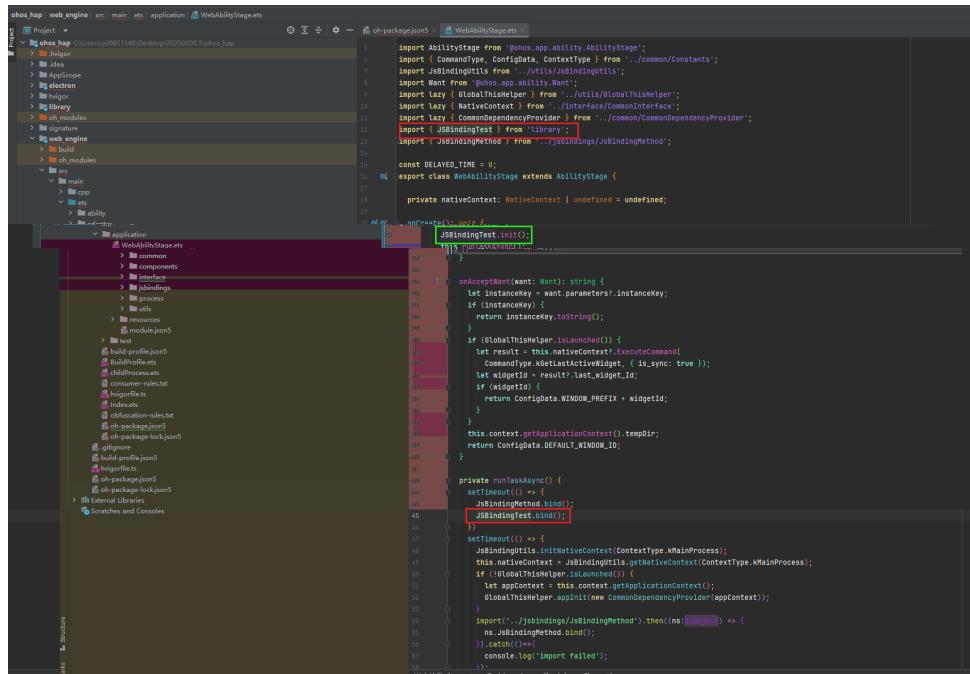
1. web_engine HAR webgine/oh-packages.json5

```

{
  "name": "web_engine",
  "version": "1.0.0",
  "description": "Please describe the basic information.",
  "main": "Index.ets",
  "author": "",
  "license": "Apache-2.0",
  "devDependencies": {
    "libadapter.so": "file:./src/main/cpp/types/libadapter",
  },
  "dependencies": {
    "inversify": "^6.0.1",
    "reflect-metadata": "^0.1.13",
    "library": "file:../library"
  }
}

```

2. WebAbilityStage.ets



```
import AbilityStage from '@ohos.app.ability.AbilityStage';
import { CommandType, ConfigData, ContextType } from '../common/Constants';
import JsBindingUtils from '../utils/JsBindingUtils';
import Want from '@ohos.app.ability.Want';
import lazy { GlobalThisHelper } from '../utils/GlobalThisHelper';
import lazy { NativeContext } from '../interface/CommonInterface';
import lazy { CommonDependencyProvider } from '../common/CommonDependencyProvider';
import { JSBindingTest } from 'library';
import { JsBindingMethod } from './jsbindings/JsBindingMethod';

const DELAYED_TIME = 0;
export class WebAbilityStage extends AbilityStage {
    private nativeContext: NativeContext | undefined = undefined;

    onAcceptWant(want: Want): string {
        let instanceKey = want.parameters?.instanceKey;
        if (instanceKey) {
            return instanceKey.toString();
        }
        if (GlobalThisHelper.isLaunched()) {
            let result = this.nativeContext?.ExecuteCommand(
                CommandType.kGetLastActiveWidget, { is_sync: true });
            let widgetId = result?.last_widget_Id;
            if (widgetId) {
                return ConfigData.WINDOW_PREFIX + widgetId;
            }
        }
        this.context.getApplicationContext().tempDir;
        return ConfigData.DEFAULT_WINDOW_ID;
    }

    private runTaskAsync() {
        setTimeout(() => {
            JsBindingMethod.bind();
            JSBindingTest.bind();
        })
        setTimeout(() => {
            this.nativeContext?.JsBindingUtils.getNativeContext(ContextType.kMainProcess);
            this.nativeContext?.JsBindingUtils.getNativeContext(ContextType.kMainProcess);
            if (!GlobalThisHelper.isLaunched()) {
                let appContext = this.context.getApplicationContext();
                GlobalThisHelper.initializeApp(new CommonDependencyProvider(appContext));
            }
            import('../jsbindings/JsBindingMethod').then(ns=>{
                ns.JsBindingMethod.bind();
            }).catch((e) => {
                console.log(`Import failed`);
            });
        })
    }
}
```

```
// Copyright (c) 2024 Huawei Device Co., Ltd. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.
```

```
import AbilityStage from '@ohos.app.ability.AbilityStage';
import { CommandType, ConfigData, ContextType } from '../common/Constants';
import JsBindingUtils from '../utils/JsBindingUtils';
import Want from '@ohos.app.ability.Want';
import lazy { GlobalThisHelper } from '../utils/GlobalThisHelper';
import lazy { NativeContext } from '../interface/CommonInterface';
import lazy { CommonDependencyProvider } from '../common/CommonDependencyProvider';
import { JSBindingTest } from 'library';
import { JsBindingMethod } from './jsbindings/JsBindingMethod';

const DELAYED_TIME = 0;
export class WebAbilityStage extends AbilityStage {

    private nativeContext: NativeContext | undefined = undefined;

    onAcceptWant(want: Want): string {
        let instanceKey = want.parameters?.instanceKey;
        if (instanceKey) {
            return instanceKey.toString();
        }
        if (GlobalThisHelper.isLaunched()) {
            let result = this.nativeContext?.ExecuteCommand(
                CommandType.kGetLastActiveWidget, { is_sync: true });
            let widgetId = result?.last_widget_Id;
            if (widgetId) {
                return ConfigData.WINDOW_PREFIX + widgetId;
            }
        }
        this.context.getApplicationContext().tempDir;
        return ConfigData.DEFAULT_WINDOW_ID;
    }

    private runTaskAsync() {
        setTimeout(() => {
            JsBindingMethod.bind();
            JSBindingTest.bind();
        })
        setTimeout(() => {
            this.nativeContext?.JsBindingUtils.getNativeContext(ContextType.kMainProcess);
            this.nativeContext?.JsBindingUtils.getNativeContext(ContextType.kMainProcess);
            if (!GlobalThisHelper.isLaunched()) {
                let appContext = this.context.getApplicationContext();
                GlobalThisHelper.initializeApp(new CommonDependencyProvider(appContext));
            }
            import('../jsbindings/JsBindingMethod').then(ns=>{
                ns.JsBindingMethod.bind();
            }).catch((e) => {
                console.log(`Import failed`);
            });
        })
    }
}
```

```

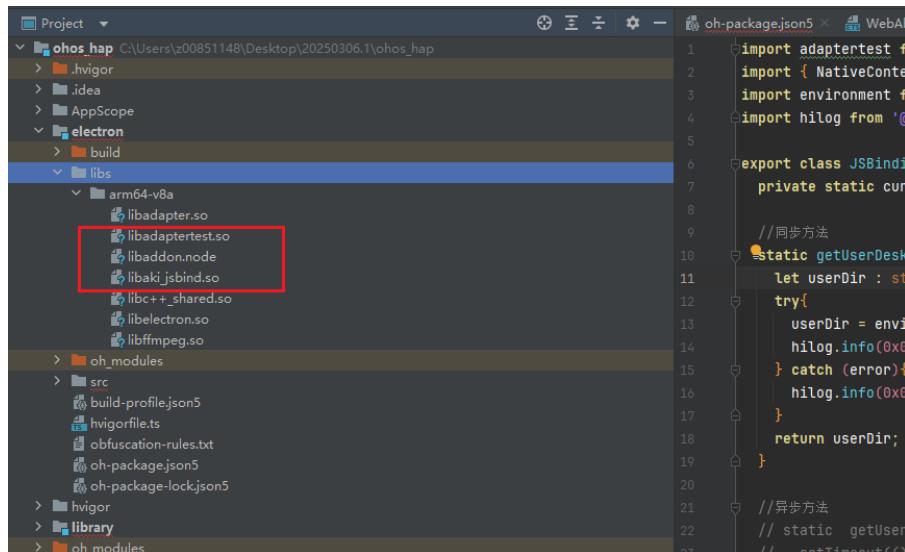
        GlobalThisHelper.appInit(new CommonDependencyProvider(getApplicationContext()));
    }
    import('../jsbindings/JsBindingMethod').then((ns:ESObject) => {
        ns.JsBindingMethod.bind();
    }).catch(()=>{
        console.log('import failed');
    });
}, DELAYED_TIME)
}
}

```

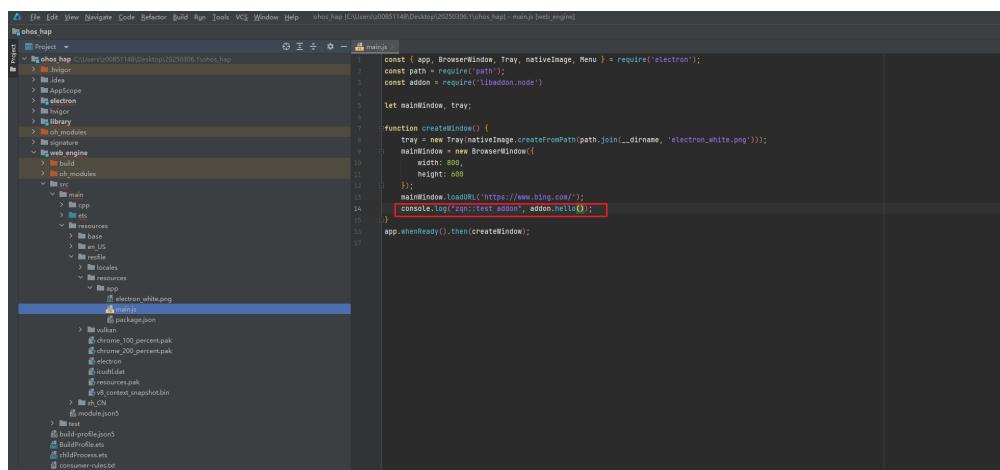
2.2.7

- **libs:** 用于存放.so文件。
- **src > main > cpp > types:** 用于存放C++ API描述文件，子目录按照so维度进行划分。
- **src > main > cpp > types > liblibrary > Index.d.ts:** 描述C++接口的方法名、入参、返回参数等信息。
- **src > main > cpp > types > liblibrary > oh-package.json5:** 描述so三方包声明文件入口和so包名信息。
- **src > main > cpp > CMakeLists.txt:** CMake配置文件，提供CMake构建脚本。
- **src > main > cpp > napi_init.cpp:** 共享包C++代码源文件。
- **Index.ets:** 共享包导出声明的入口。

1 HAP



2 JS



3

