

# .NET Conf China 2025

改变世界 改变自己

2025 年 11 月 30 日 | 中国 上海





# 鸿蒙上的 Avalonia 和 C#

01

Avalonia 和 C# 在鸿蒙上的运行方式、限制、性能瓶颈、实机演示。

02

OpenHarmony NDK API 绑定的构建方式、代码组织方式、综合进度。

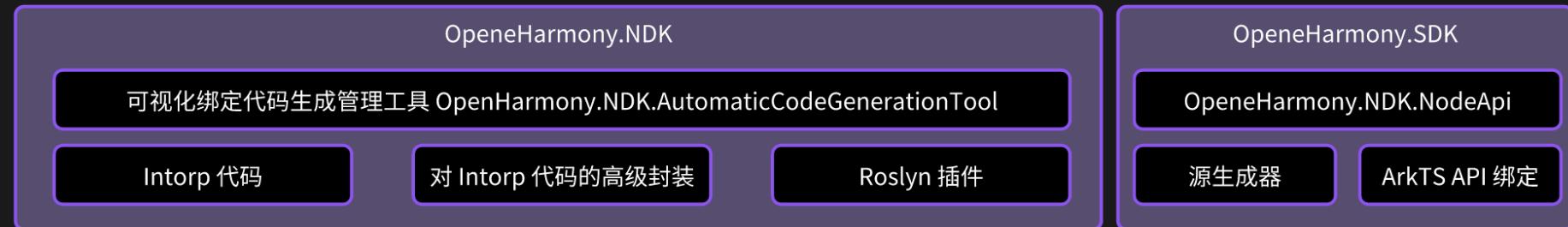
# 一览当前的 C# 与鸿蒙

.NET 程序



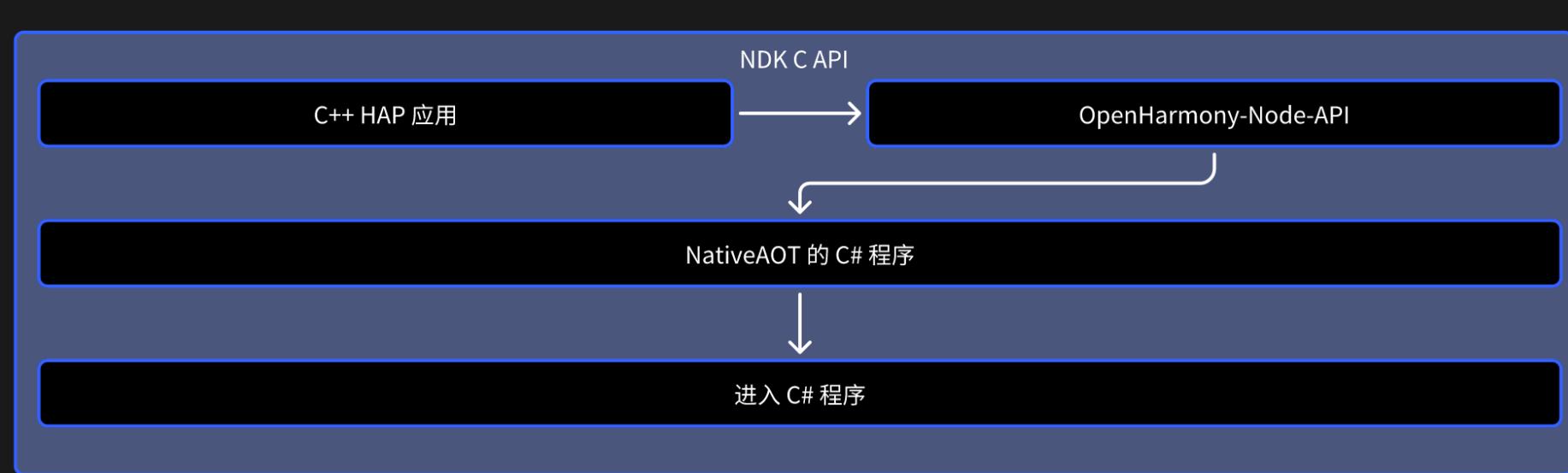
其它 .NET 程序

访问介质



.NET 运行时

原生平台



# 前情提要：Avalonia 和 C# 是如何在鸿蒙上运行的？

## C# 发布鸿蒙可用程序之常规流程：

- 将程序编译为NativeAOT库；
- 将所得二进制库导入HAP工程；
- 按需调用C#程序。

## Avalonia 应用于鸿蒙系统运行的一般流程：

- 利用 XComponent 模块功能注册应用生命周期。
- 除需留意更严格的权限限制与文件访问要求外，与常规 Avalonia 开发无其他异同。

```
build() {
  Row() {
    XComponent({ type: XComponentType.SURFACE, id: 'AvaloniaRootComponent', libraryname: "entry" })
      .focusable(true)
      .focusOnTouch(true)
      .defaultFocus(true)
    }
    .height('100%')
    .onClick(() => {
      let context = this.getUIContext().getHostContext() as common.UIAbilityContext;
    })
}
```

```
extern "C" __attribute__((constructor)) void RegisterEntryModule(void) {
  auto handle = dlopen("libavalonia.so", RTLD_NOW);
  assert(handle != nullptr);
  auto func = (void (*)())dlsym(handle, "RegisterEntryModule");
  assert(func != nullptr);
  func();
  return;
```

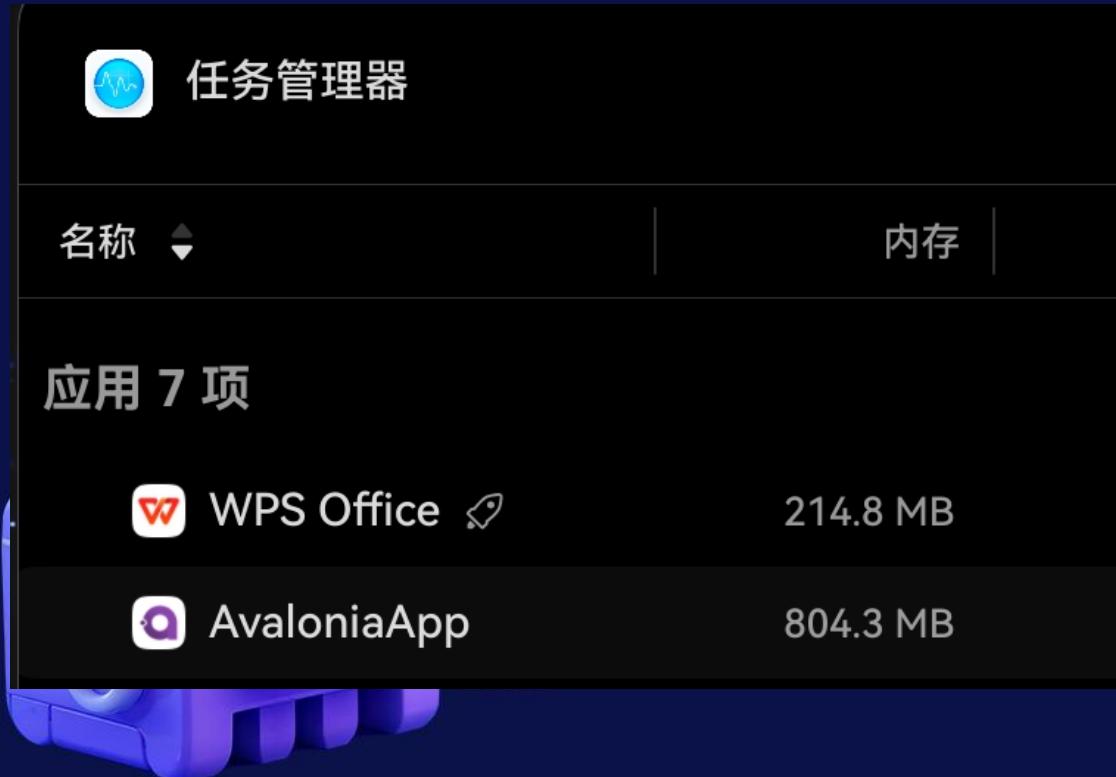
```
[UnmanagedCallersOnly(CallConvs = [typeof(CallConv.Cdecl)], EntryPoint = "RegisterEntryModule")]
public static void RegisterEntryModule() {...}

// [UnmanagedCallersOnly(CallConvs = [typeof(CallConv.Cdecl)])]
public static unsafe napi_value Init(napi_env env, napi_value exports)
{
  try
  {
    if (NodeApi.napi_get_named_property(env, exports, utf8name:"__NATIVE_XCOMPONENT_OBJ__",
      result: out var exportInstance:napi_value) ==
      napi_status.Ok)
    {
      if (NodeApi.napi_unwrap(env, exportInstance, out var result:IntPtr) ==
          napi_status.Ok)
      {
        var nativeXComponent = Unsafe.BitCast<IntPtr, OH_NativeXComponent>(result);
```

# 当前最为突出性能问题

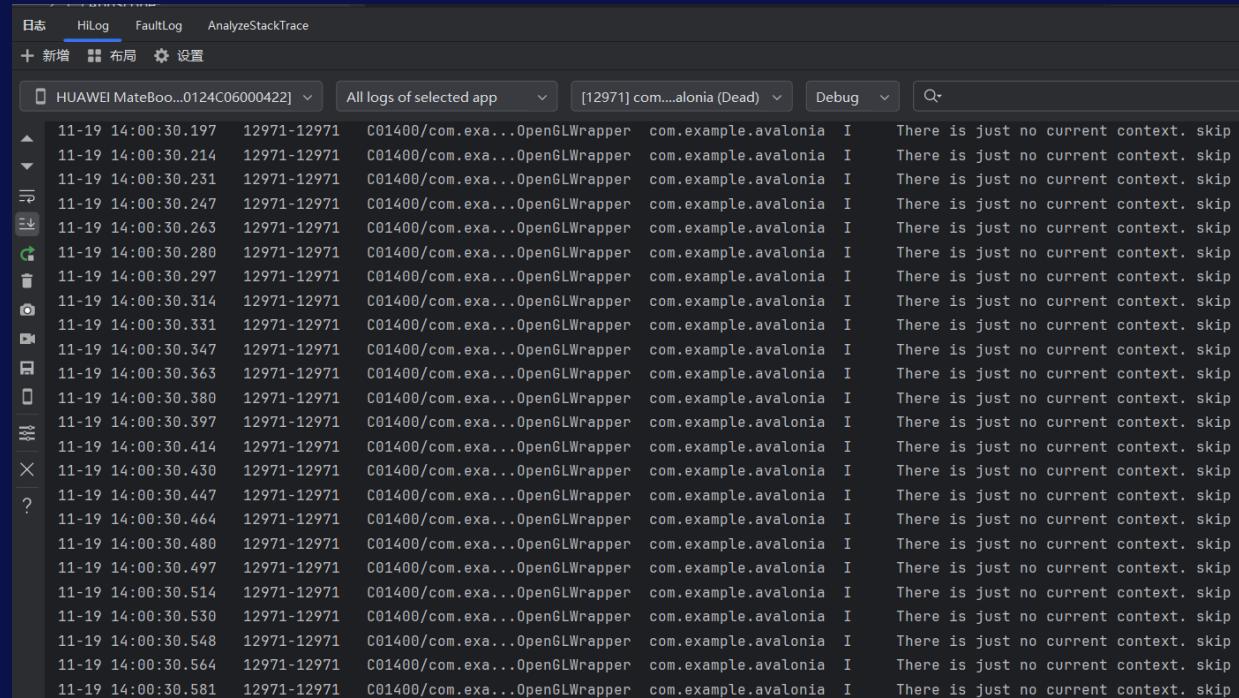
C#

- 超高的内存占用。



Avalonia

- 频繁的GL上下文错误致使渲染性能不稳定。



# 发展史简要

2024 China .NET Conf  
孙策首次分享.NET运行  
于鸿蒙平台

25年第一季度中旬，团队正式成立

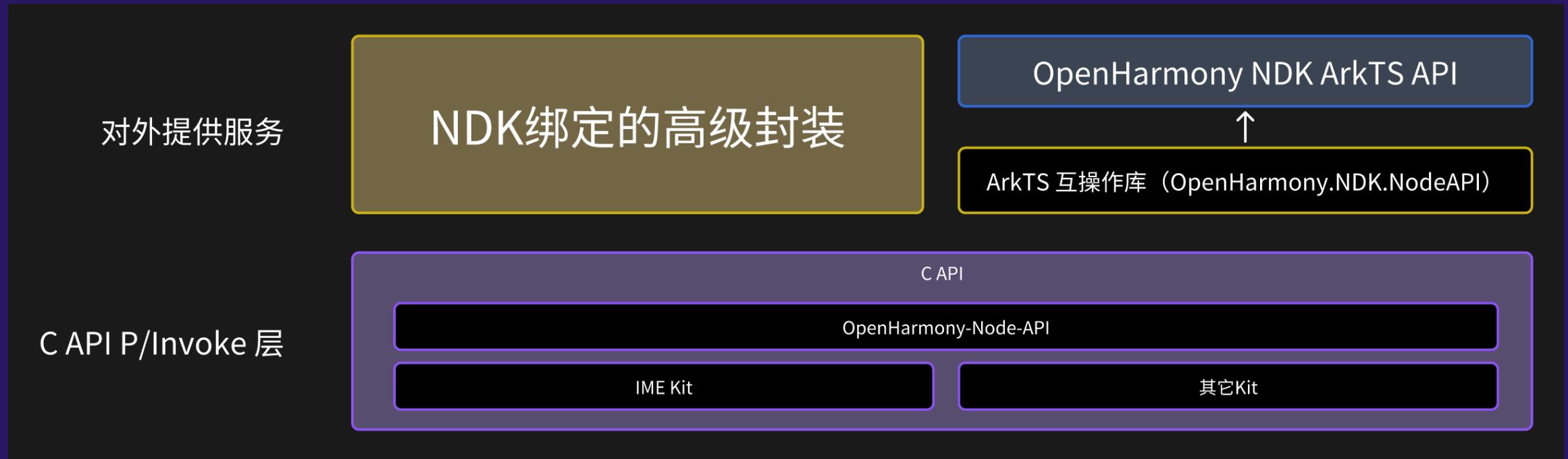
华为开发者大会2025 (HDC2025)

25年第三季度初

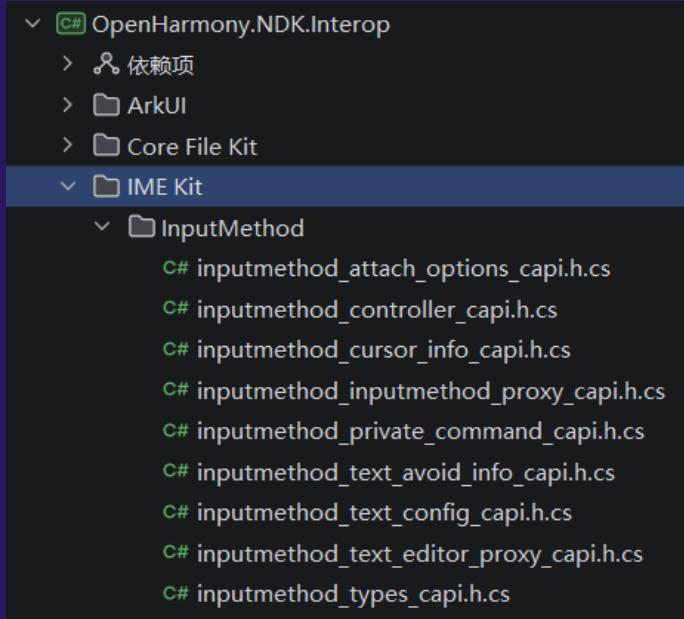
团队首次集体会面，商讨项目发展方向。  
寻找专家解答我们的疑惑。

Presented with xmind

# OpenHarmony.NDK 项目架构



# 代码组织方式、P/Invoke 风格和代码生成工具

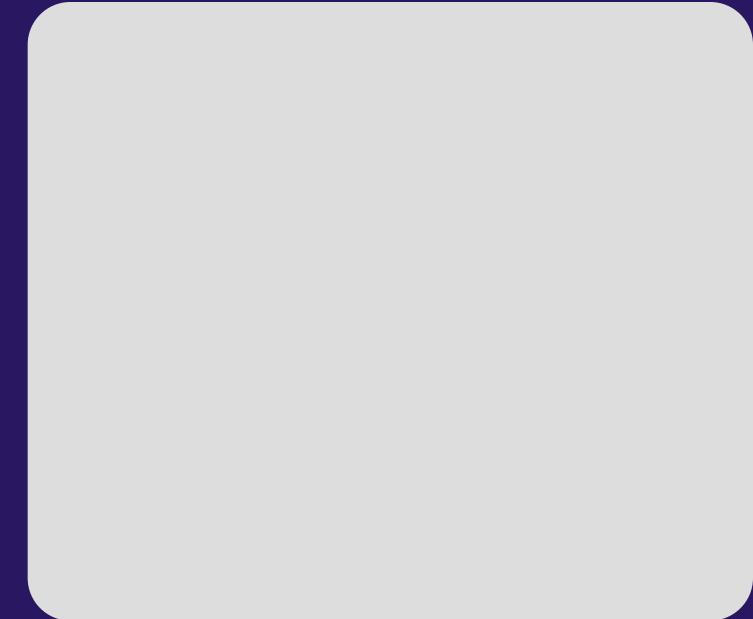


```
[DllImport( libraryName: "libohinputmethod.so",
    StringMarshalling = StringMarshalling.Utf16)]
[UnmanagedCallConv(CallConvs = [typeof(CallConvCdecl)])]
public static partial InputMethod_ErrorCode
    &1 用法 &重置空优 *
    OH_InputMethodProxy_NotifySelectionChange(
        InputMethod_InputMethodProxy inputMethodProxy,
        string text,
        ulong length,
        int start,
        int end);

[DllImport( libraryName: "libohinputmethod.so")]
[UnmanagedCallConv(CallConvs = [typeof(CallConvCdecl)])]
public static partial InputMethod_ErrorCode
    &1 用法 &新 *
    OH_InputMethodProxy_SendPrivateCommand(
        InputMethod_InputMethodProxy inputMethodProxy,
        [In] InputMethod_PrivateCommand[] privateCommand,
        ulong size);
```

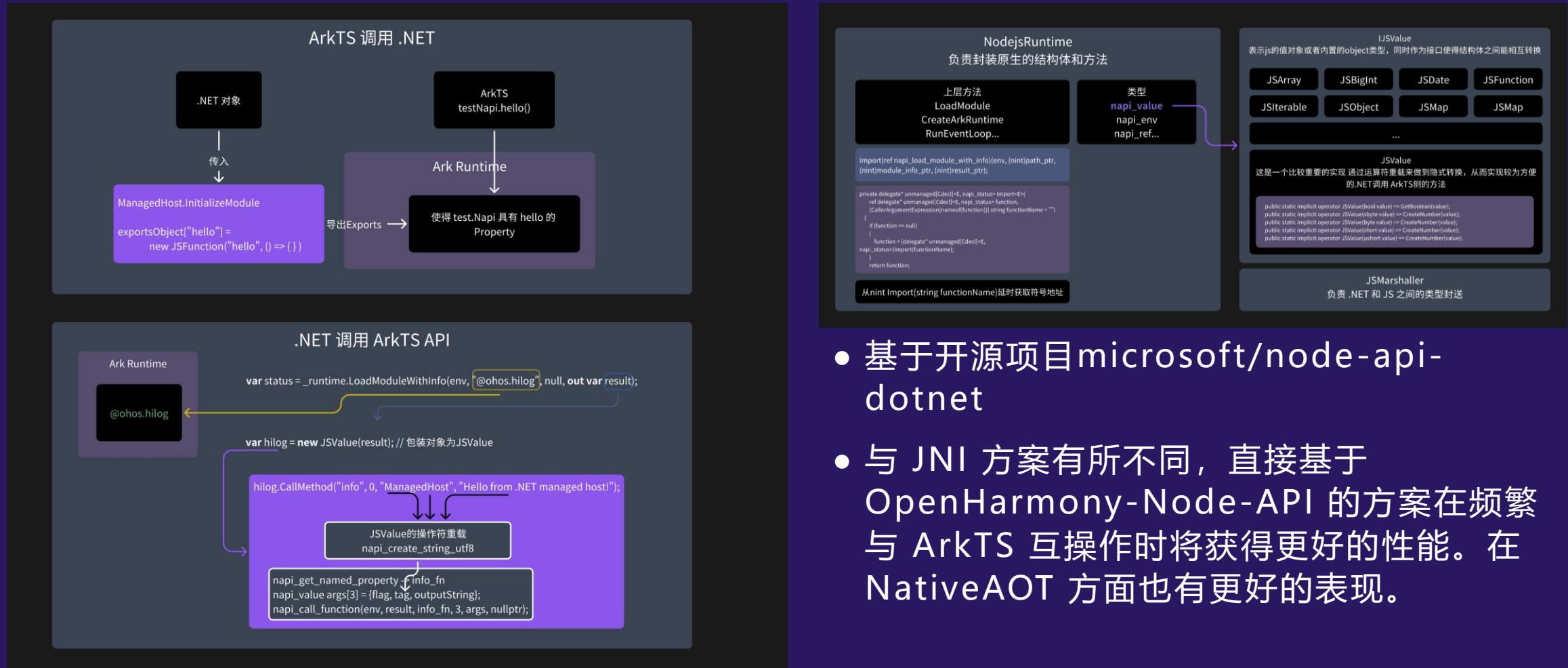
- 每一个存放绑定代码的 C# 文件都与实际的 C 头文件对应。
- 文件夹与 Kit 和模块对应。

我们善用 DllImport 和 LibraryImport，这使得开发者可以方便地使用Interop代码而无需过多关注封送方面的问题。



这是一个基于Avalonia构建的可视化工具。旨在更高效地处理NDK API的绑定与具有特定规律代码的高级封装。

# C# 与 ArkTS互操作



- 基于开源项目 `microsoft/node-api-dotnet`
- 与 JNI 方案有所不同，直接基于 OpenHarmony-Node-API 的方案在频繁与 ArkTS 互操作时将获得更好的性能。在 NativeAOT 方面也有更好的表现。

# 综合进度

## .NET SDK

类似 CodeArts 那样，借助 HNP 方案让第三方生产力框架运行在鸿蒙平台上。

我们已做过一些尝试并存有技术方向。

## OpenHarmony.NDK

工具开发进度过半，预计年末完成关键 Kit 的高级封装开发以使其他工作重新启动。

## .NET Runtime

仍然只支持 NativeAOT，修复了 Marshal.GetFunctionForDelegate。

## OpenHarmony.SDK

正在进行方案验证。

## Avalonia

新增一些平台功能的实现。

# THANK YOU

特别鸣谢（不分先后）  
孙策、董彬（铱泓科技）、钟应斌（赣东学院）