.NETの上で
お手製の言語を動かす技術

Kyoto.cs#1

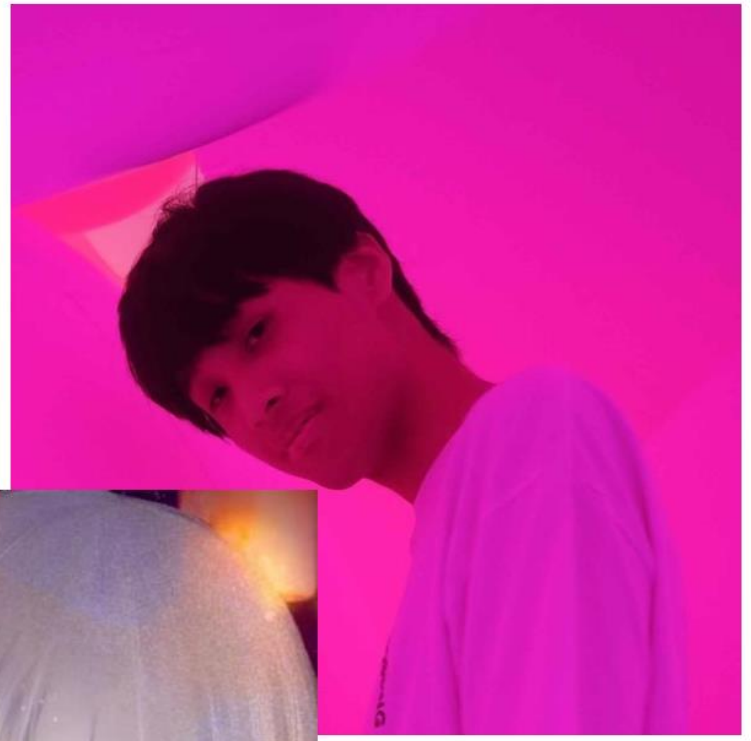
2025/4/6

京都産業大学 上村太成

WSOFT

自己紹介

- taiseiue
  - @taiseiue
  - id:taiseiue
  - taiseiue.jp
  - C#/TypeScript/Perl
  - WSOFT



Taskbar Hiddener

タスクバー表示切替

タスクバーを隠したり表示したりするシンプルなソフトウェア

What's this?

もしあなたがタスクバーの存在に頭を悩ませているなら、このソフトウェアが役に立つでしょう。 このソフトウェアを使うと、Windowsの「タスクバーを自動的に隠す」オプションとは異なり、自分のタイミングでタスクバーを隠したり、表示したりできます。

このソフトウェアは、 内部でWin32APIを呼び出し、タスクバーのウィンドウハンドルを取得することで自在に隠したり、表示したりしています。このソフトウェアは.Net Framework 4.5.2以降の環境で動作します。(つまり、

mukaikun

向井くん

Moodleのリソース表示を改善するくん

これは何

- Moodleで掲示されているPDFなどのファイルを、ダウンロードせずにブラウザ内で読み込むChrome拡張機能です。
- Moodle以外のサイトでも、パターンを書くことでブラウザ内で読み込めます。
- 初めましての人は、まず下の設定欄にMoodleのUrlを入力しよう。

Welcome to Lantana!

Lantana は、シンプルで軽量なMKDocsのテーマです。LantanaとMkDocsを使用すると、簡単に自分のサイトを作成できます。

Lantanaは、Bootstrapを使用した軽量なMkDocsのテーマです。 LantanaとMkDocsを使用すると、簡単に自分のサイトを作成できます。

Lantanaはシンプルながらも、多言語に対応しており、フリーソフトかつオープンソースプロジェクトで開発されています。LantanaはMITライセンスのもとで頒布されているので、オープンソースのプロジェクトとプロプライエタリのプロジェクトの両方で使用できます。

使用法

- 作業開始

サポート

Lantanaの使用に際してサポートが必要な場合は、お気軽にお問合せください。

- 些細な質問でも、お気軽にGitHubのDiscussionsに投稿してください！
- 不具合を報告したり、新機能をリクエストするには、GitHubでIssueを開いてください。

貢献

C#の話をします

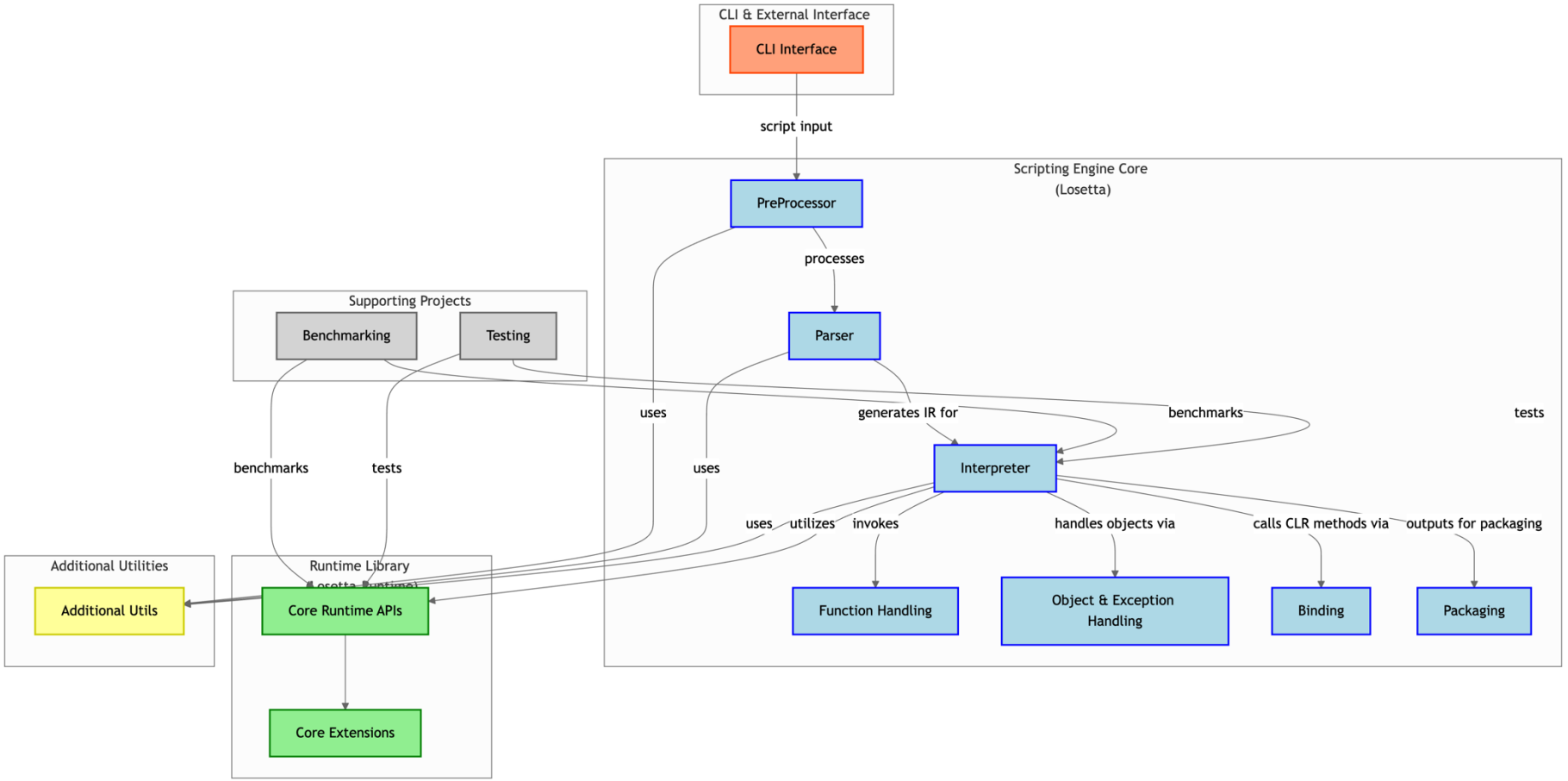
AliceScript

Powerd by AliceProject

```
using Alice.IO;
using Alice.Shell;
using Alice.Random;
using Alice.Diagnostics;
using Alice.Environment;

namespace WSOFT.AliceScript.Installer;

private string GetRepositoryName(string address)
{
    var parts = address.Split('/');
    return parts[^1];
}
private string GetRepositoryTag(string address)
{
```

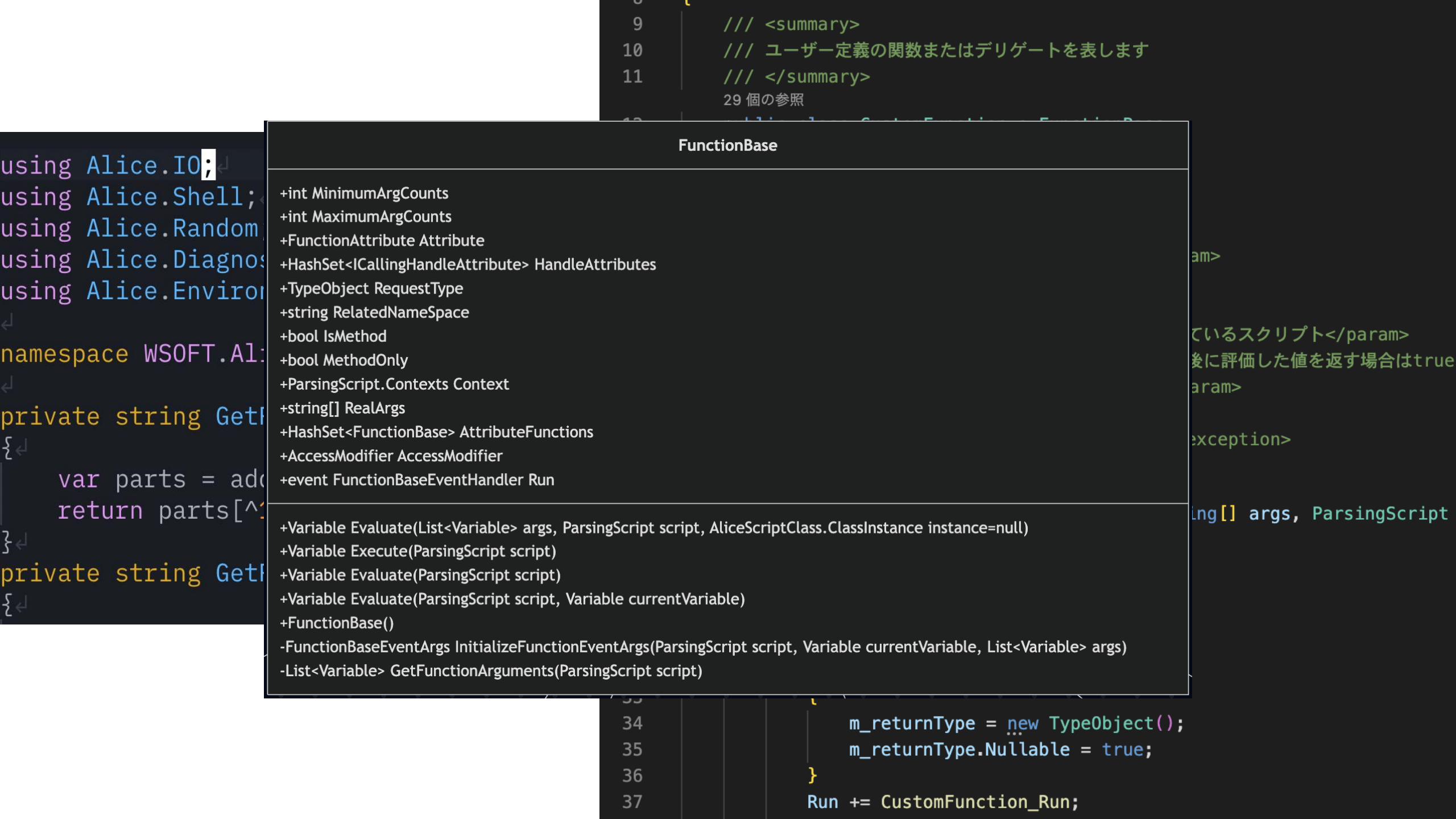
```
$ brew tap wsoft-ws/losetta
$ brew install alice
```

AliceScriptの実装: Losetta

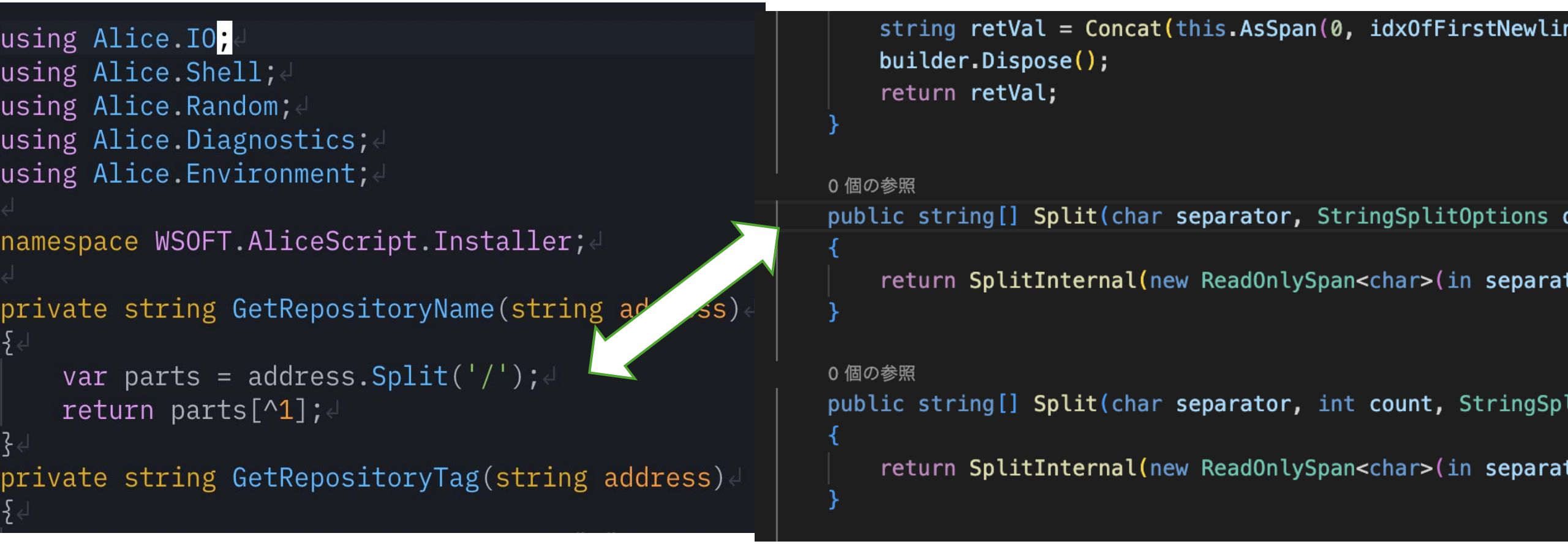


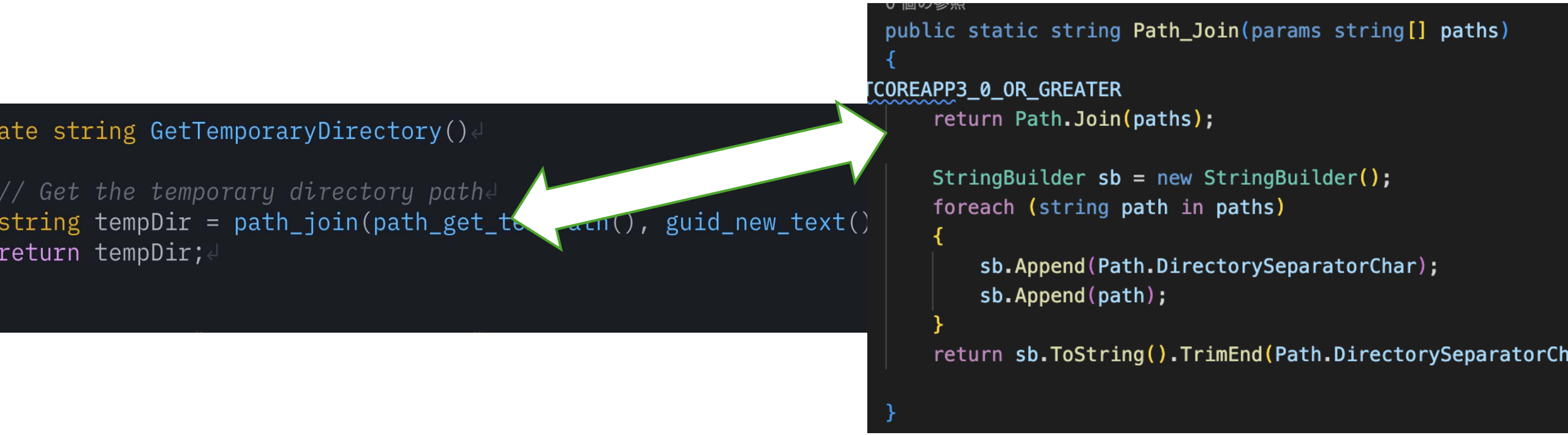
```
using Alice.IO;
using Alice.Shell;
using Alice.Random;
using Alice.Diagnostics;
using Alice.Environment;

namespace WSOFT.AliceScript.Installer;

private string GetRepositoryName(string address)
{
    var parts = address.Split('/');
    return parts[^1];
}
private string GetRepositoryTag(string address)
{
```

```
9       /// <summary>
10      /// ユーザー定義の関数またはデリゲートを表します
11      /// </summary>
        29 個の参照
12      public class CustomFunction : FunctionBase
        {
            /// <summary>
            /// 新しいユーザー定義関数を作成します
            /// </summary>
            /// <param name="funcName">関数の名前</param>
            /// <param name="body">関数の本文</param>
            /// <param name="args">引数の一覧</param>
            /// <param name="script">この関数が定義されているスクリプト</param>
            /// <param name="forceReturn">この関数が最後に評価した値を返す場合はtrue
            /// <param name="returnType">戻り値の型</param>
            /// <param name="nullable"></param>
            /// <exception cref="ScriptException"></exception>
                参照
            pu  CustomFunction(string funcName,
                                        string body, string[] args, ParsingScript
            {
                Name = funcName;
                m_body = body;
                m_forceReturn = forceReturn;
                m_returnType = returnType;
                if (m_returnType is null)
                {
                    m_returnType = new TypeObject();
                    m_returnType.Nullable = true;
                }
                Run += CustomFunction_Run;
31
32
33
34
35
36
37
```


FunctionBase
+int MinimumArgCounts
+int MaximumArgCounts
+FunctionAttribute Attribute
+HashSet<ICallingHandleAttribute> HandleAttributes
+TypeObject RequestType
+string RelatedNameSpace
+bool IsMethod
+bool MethodOnly
+ParsingScript.Contexts Context
+string[] RealArgs
+HashSet<FunctionBase> AttributeFunctions
+AccessModifier AccessModifier
+event FunctionBaseEventHandler Run
+Variable Evaluate(List<Variable> args, ParsingScript script, AliceScriptClass.ClassInstance instance=null)
+Variable Execute(ParsingScript script)
+Variable Evaluate(ParsingScript script)
+Variable Evaluate(ParsingScript script, Variable currentVariable)
+FunctionBase()
-FunctionBaseEventArgs InitializeFunctionEventArgs(ParsingScript script, Variable currentVariable, List<Variable> args)
-List<Variable> GetFunctionArguments(ParsingScript script)

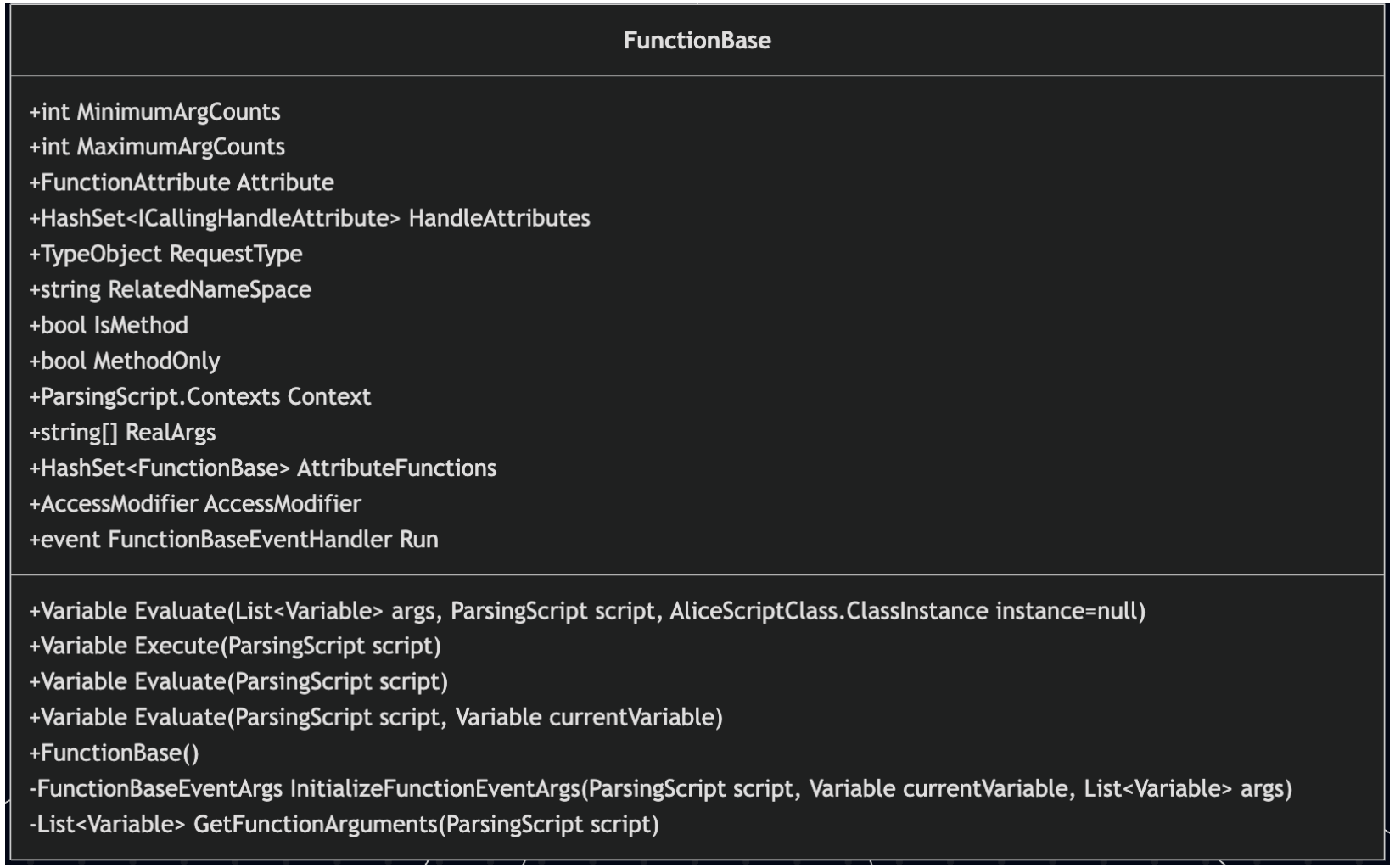
じゃあ
AliceScriptの外の関数は?


using Alice.IO;
using Alice.Shell;
using Alice.Random;
using Alice.Diagnostics;
using Alice.Environment;
namespace WSOFT.AliceScript.Installer;
private string GetRepositoryName(string address)
{
var parts = address.Split('/');
return parts[^1];
}
private string GetRepositoryTag(string address)
{
string retVal = Concat(this.AsSpan(0, idxOfFirstNewlin
builder.Dispose();
return retVal;
}
0 個の参照
public string[] Split(char separator, StringSplitOptions
{
return SplitInternal(new ReadOnlySpan<char>(in separat
}
0 個の参照
public string[] Split(char separator, int count, StringSpl
{
return SplitInternal(new ReadOnlySpan<char>(in separat
}



```
ate string GetTemporaryDirectory()

// Get the temporary directory path
string tempDir = path_join(path_get_te      ath(), guid_new_text()
return tempDir;
```

```
public static string Path_Join(params string[] paths)
{
TCOREAPP3_0_OR_GREATER
    return Path.Join(paths);

    StringBuilder sb = new StringBuilder();
    foreach (string path in paths)
    {
        sb.Append(Path.DirectorySeparatorChar);
        sb.Append(path);
    }
    return sb.ToString().TrimEnd(Path.DirectorySeparatorCh

}
```


FunctionBase
+int MinimumArgCounts
+int MaximumArgCounts
+FunctionAttribute Attribute
+HashSet<ICallingHandleAttribute> HandleAttributes
+TypeObject RequestType
+string RelatedNameSpace
+bool IsMethod
+bool MethodOnly
+ParsingScript.Contexts Context
+string[] RealArgs
+HashSet<FunctionBase> AttributeFunctions
+AccessModifier AccessModifier
+event FunctionBaseEventHandler Run
+Variable Evaluate(List<Variable> args, ParsingScript script, AliceScriptClass.ClassInstance instance=null)
+Variable Execute(ParsingScript script)
+Variable Evaluate(ParsingScript script)
+Variable Evaluate(ParsingScript script, Variable currentVariable)
+FunctionBase()
-FunctionBaseEventArgs InitializeFunctionEventArgs(ParsingScript script, Variable currentVariable, List<Variable> args)
-List<Variable> GetFunctionArguments(ParsingScript script)

こういう関数を考える

```
number pow(number x, number y);
```

```
class PowFunction : FunctionBase
{
    0 個の参照
    public PowFunction()
    {
        Name = "pow";
        Run += OnRun;
        MinimumArgCounts = 2;
    }
    1 個の参照
    public void OnRun(object sender, FunctionBaseEventArgs e)
    {
        double x = Utils.GetSafeInt(e.Args, 0);
        double y = Utils.GetSafeInt(e.Args, 1);

        double result = Math.Pow(x, y);

        e.Return = Variable.From(result);
    }
}
```

しんどい

```
0 個の参照
public static double Pow(double x, double y)
{
    return Math.Pow(x, y);
}
```

こういうのにしたい

```
0 個の参照
public static double Pow(double x, double y)
{
    return Math.Pow(x, y);
}
```

```
class PowFunction : FunctionBase
{
    0 個の参照
    public PowFunction()
    {
        Name = "pow";
        Run += OnRun;
        MinimumArgCounts = 2;
    }
    1 個の参照
    public void OnRun(object sender, FunctionBaseEvent
    {
        double x = Utils.GetSafeInt(e.Args, 0);
        double y = Utils.GetSafeInt(e.Args, 1);

        double result = Math.Pow(x, y);

        e.Return = Variable.From(result);
    }
}
```

いるもの

- Variable型から指定した型への変換
- .NETのメソッドをFunctionBaseにする

Variableか.NETの型への変換

```
/// <summary>
/// この変数を指定した型に変換します
/// </summary>
/// <typeparam name="T">変換先の型</typeparam>
/// <returns>変換されたオブジェクト</returns>
22 個の参照
public T As<T>()
{
    return (T)ConvertTo(typeof(T));
}
/// <summary>
/// この変数を指定した型に変換できるか試みます
/// </summary>
/// <param name="result">変換されたオブジェクト</param>
/// <returns>変換に成功した場合はTrue、それ以外の場合はfalse</returns>
12 個の参照
public bool Is<T>(out T result)
{
    bool r = TryConvertTo(typeof(T), out object o);
    result = r ? (T)o : default;
    return r;
}
```

```
case VarType.NUMBER:
    {
        if (type == typeof(byte))
        {
            result = (byte)Value;
            return Utils.TestNumInRange(this, true, byte.MinValue, byte.MaxValue);
        }
        if (type == typeof(sbyte))
        {
            result = (sbyte)Value;
            return Utils.TestNumInRange(this, true, sbyte.MinValue, sbyte.MaxValue);
        }
        if (type == typeof(short))
        {
            result = (short)Value;
            return Utils.TestNumInRange(this, true, short.MinValue, short.MaxValue);
        }
        if (type == typeof(ushort))
        {
            result = (ushort)Value;
            return Utils.TestNumInRange(this, true, ushort.MinValue, ushort.MaxValue);
        }
        if (type == typeof(int))
        {
            result = (int)Value;
            return Utils.TestNumInRange(this, true, int.MinValue, int.MaxValue);
        }
        if (type == typeof(uint))
        {
            result = (uint)Value;
            return Utils.TestNumInRange(this, true, uint.MinValue, uint.MaxValue);
        }
```

いるもの

- ✅Variable型から指定した型への変換
- .NETのメソッドをFunctionBaseにする

```
Console.WriteLine("Hello,World!");

Type type = typeof(Console);

MethodInfo method = type.GetMethod("WriteLine");

method.Invoke(null, ["Hello,World!"]);
```

```
{
    /// <summary>
    /// .NETのメソッドと対応するAliceScriptの関数
    /// </summary>
    25 個の参照
    public class BindFunction : FunctionBase
    {
        /// <summary>
        /// BindFunctionを初期化します
        /// </summary>
        2 個の参照
        public BindFunction()
        {
            HandleAttributes = new HashSet<ICallingHandleAttribute>();
            Run += BindFunction_Run;
        }

        1 個の参照
        private void BindFunction_Run(object sender, FunctionBaseEventArgs e)
        {
            bool wantMethod = e.CurentVariable is not null;
            foreach (var load in Overloads)
            {
                if ((!wantMethod || load.IsMethod) && load.TryConvertParameters(e, this, out var args))
                {
```

BindFunctionには対応する
.NET のメソッドの
MethodInfoを持たせておく

BindFunctionには対応する
.NET のメソッドの
MethodInfoを持たせておく

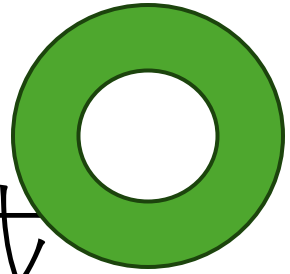
遅い!

MethodInfo.Invokeの欠点

- 遅い
  - リフレクション
  - 呼び出し毎に遅延バインディングが起きる

リフレクション

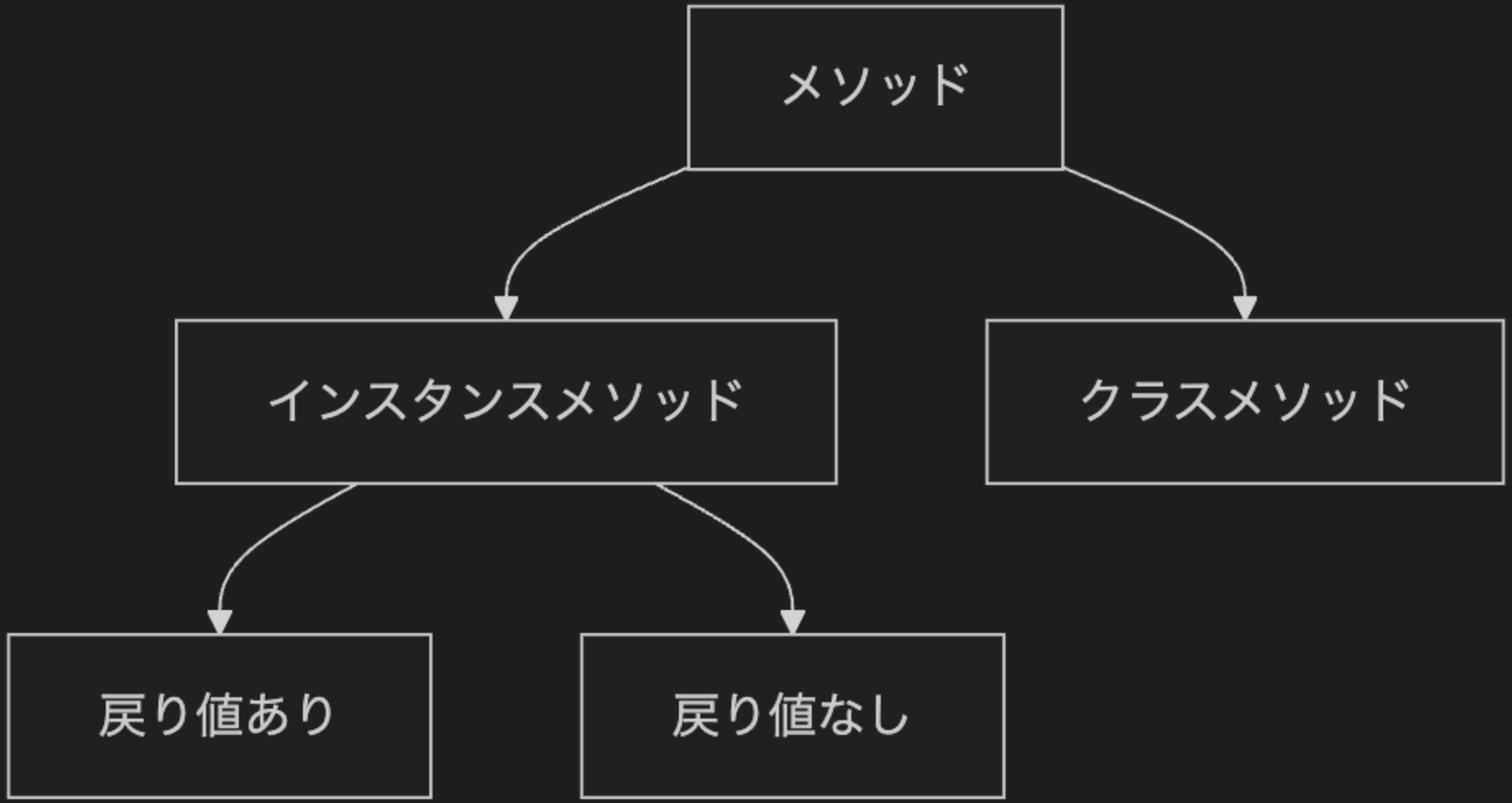
動的コード生成



```
var args = Expression.Parameter(typeof(object[]), "args");
var instance = Expression.Parameter(typeof(object), "instance");
var parameters = load.TrueParameters.Select((x, index) =>
    Expression.Convert(
        Expression.ArrayIndex(args, Expression.Constant(index)),
        Utils.GetTrueParametor(x.ParameterType)
        )).ToArray();
var f = Expression.Lambda<Func<object[], object>>(
    Expression.Convert(
            Expression.Call(Expression.Convert(instance, methodInfo.DeclaringType), methodInfo, parameters),
            typeof(object)),
        instance, args
).Compile();
```

```
var args = Expression.Parameter(typeof(object[]), "args");
var instance = Expression.Parameter(typeof(object), "instance");
var parameters = load.TrueParameters.Select((x, index) =>
    Expression.Convert(
        Expression.ArrayIndex(args, Expression.Constant(index)),
        Utils.GetTrueParametor(x.ParameterType)
        )).ToArray();
var f = Expression.Lambda<Func<object[], object>>(
    Expression.Convert(
            Expression.Call(Expression.Convert(instance, methodInfo.DeclaringType), methodInfo, parameters),
            typeof(object)),
        instance, args
).Compile();
```

```
Func<object[], object> f = args => instance.Method(args);
```


メソッド
インスタンスメソッド
クラスメソッド
戻り値あり
戻り値なし

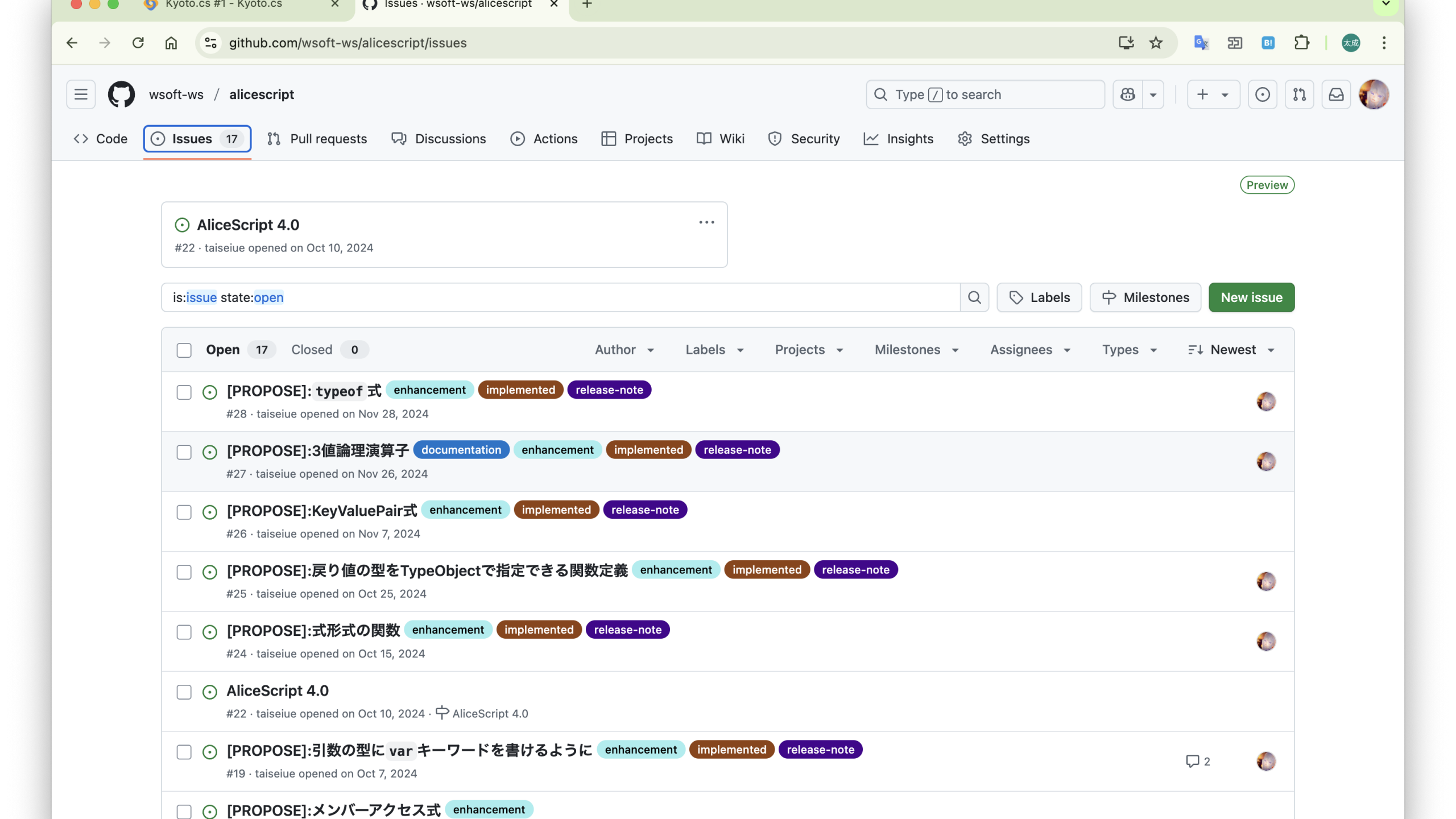
```
if (methodInfo.ReturnType == typeof(void))
{
    if (methodInfo.IsStatic)
    {
        load.VoidFunc = Expression.Lambda<Action<object[]>>(
        Expression.Convert(
            Expression.Call(methodInfo, parameters),
            typeof(void)),
        args).Compile();
    }
    else
    {
        load.InstanceVoidFunc = Expression.Lambda<Action<object, object[]>>(
        Expression.Convert(
            Expression.Call(Expression.Convert(instance, methodInfo.DeclaringType), methodInfo
            typeof(void)),
        instance, args).Compile();
        load.IsInstanceFunc = true;
    }
    load.IsVoidFunc = true;
}
```

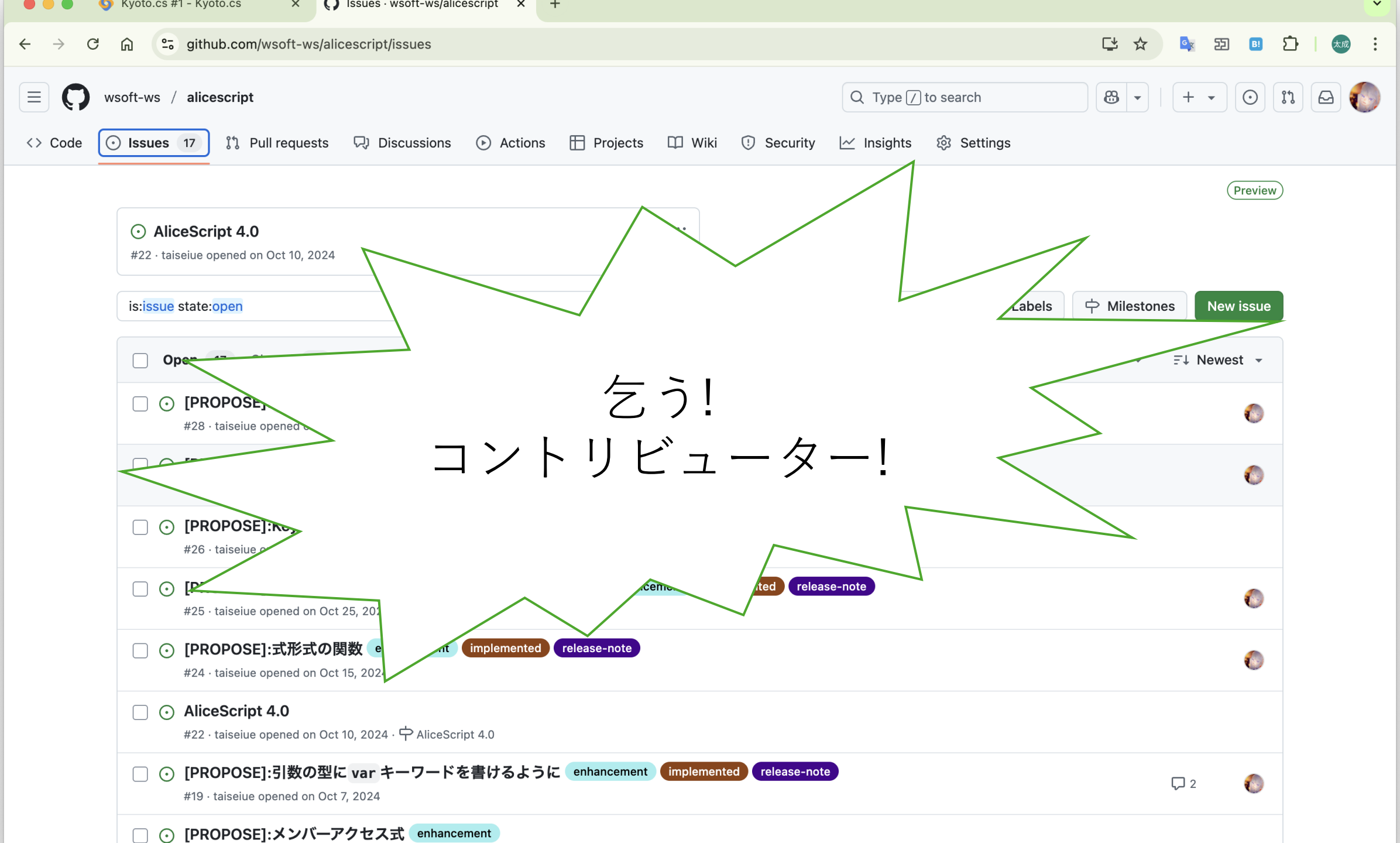
Native C Functionは?

```
string moduleName = Path.GetFileNameWithoutExtension(libraryFile.ToUpper());
AssemblyBuilder asmBld = AssemblyBuilder.DefineDynamicAssembly(
    new AssemblyName("Invoke_Asm" + moduleName), AssemblyBuilderAccess.Run);

ModuleBuilder modBld = asmBld.DefineDynamicModule(
    "Invoke_Mod" + moduleName);

TypeBuilder typBld = modBld.DefineType(
    "Invoke_Class" + moduleName,
    TypeAttributes.Public | TypeAttributes.Class);

MethodBuilder methodBuilder = typBld.DefinePInvokeMethod(
    procName, libraryFile, entryPoint ?? procName,
    MethodAttributes.Public | MethodAttributes.Static | MethodAttributes.PinvokeImpl | Method
    Constants.InvokeStringToType(returnType), Constants.InvokeStringToType(parameterTypes),
    CallingConvention.StdCall,
    useUnicode.HasValue ? useUnicode.Value ? CharSet.Unicode : CharSet.Ansi : CharSet.Auto);
methodBuilder.SetImplementationFlags(MethodImplAttributes.PreserveSig);

typBld.CreateType().GetMethod(procName);
```

まとめ

- リフレクションでメソッドを呼ぶときには
- デリゲートにキャッシュしよう


Kyoto.cs #1 - Kyoto.cs
Issues · wsoft-ws/alicescript
github.com/wsoft-ws/alicescript/issues
wsoft-ws / alicescript
Type / to search
Code
Issues 17
Pull requests
Discussions
Actions
Projects
Wiki
Security
Insights
Settings
Preview
AliceScript 4.0
#22 · taiseiue opened on Oct 10, 2024
is:issue state:open
Labels
Milestones
New issue
Open 17
Closed 0
Author
Labels
Projects
Milestones
Assignees
Types
Newest
[PROPOSE]: typeof 式 enhancement implemented release-note
#28 · taiseiue opened on Nov 28, 2024
[PROPOSE]:3値論理演算子 documentation enhancement implemented release-note
#27 · taiseiue opened on Nov 26, 2024
[PROPOSE]:KeyValuePair式 enhancement implemented release-note
#26 · taiseiue opened on Nov 7, 2024
[PROPOSE]:戻り値の型をTypeObjectで指定できる関数定義 enhancement implemented release-note
#25 · taiseiue opened on Oct 25, 2024
[PROPOSE]:式形式の関数 enhancement implemented release-note
#24 · taiseiue opened on Oct 15, 2024
AliceScript 4.0
#22 · taiseiue opened on Oct 10, 2024 · AliceScript 4.0
[PROPOSE]:引数の型に var キーワードを書けるように enhancement implemented release-note
#19 · taiseiue opened on Oct 7, 2024
2
[PROPOSE]:メンバーアクセス式 enhancement


Kyoto.cs #1 - Kyoto.cs
Issues · wsoft-ws/alicescript
github.com/wsoft-ws/alicescript/issues
wsoft-ws / alicescript
Type / to search
Code
Issues 17
Pull requests
Discussions
Actions
Projects
Wiki
Security
Insights
Settings
Preview
AliceScript 4.0
#22 · taiseiue opened on Oct 10, 2024
is:issue state:open
Labels
Milestones
New issue
Newest
[PROPOSE]
#28 · taiseiue opened
[PROPOSE]:
#26 · taiseiue
release-note
#25 · taiseiue opened on Oct 25, 202
[PROPOSE]:式形式の関数
implemented
release-note
#24 · taiseiue opened on Oct 15, 202
AliceScript 4.0
#22 · taiseiue opened on Oct 10, 2024 · AliceScript 4.0
[PROPOSE]:引数の型に var キーワードを書けるように
enhancement
implemented
release-note
#19 · taiseiue opened on Oct 7, 2024
2
[PROPOSE]:メンバーアクセス式
enhancement
乞う!
コントリビューター!

完