

(自作の)
プログラミング言語ができるまで

ランチタイムトーク

2024/5/9

B1 上村太成

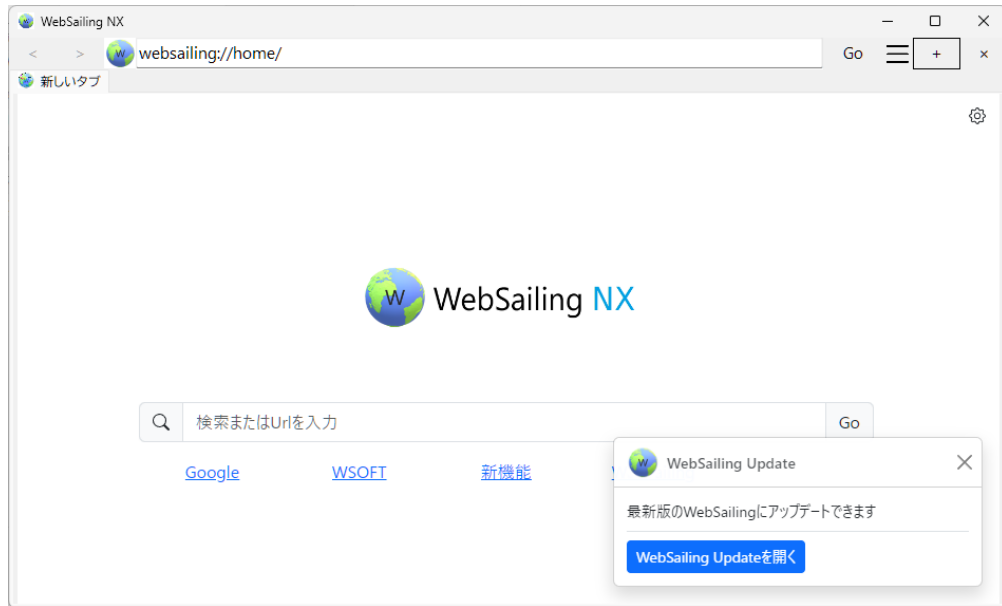
本日の話

- 作ったプログラミング言語の話
 - ふんわりしか説明されてこなかった部分に少しでもスポットライトを
 - 自分の言語がこうできてるのであって他の言語がそうとは限らない
- 何を作ったかを中心に
 - 背景
 - 作ったもの
 - 作ったものの特徴
 - これから

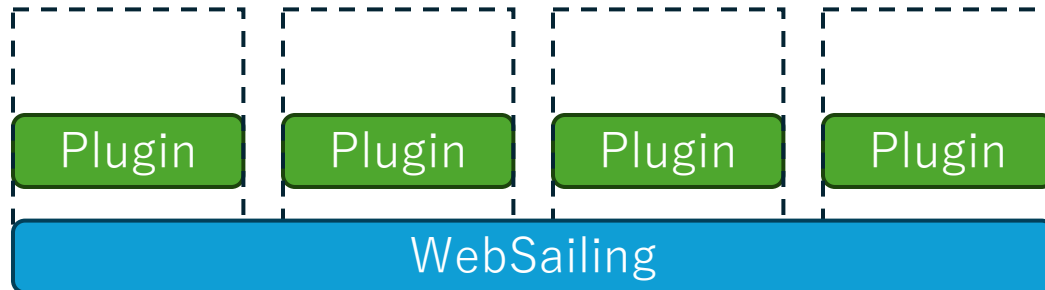
背景

どういうものを作っていて、どういう要求があったかというと

当時作っていたもの



- .NET上で動くChromiumベースのブラウザ
- ユーザーが欲しい機能をプラグインとして追加できる
- メーカーはプラグインを介して独自のアプリを作れる
- 独自のミニアプリを作るためのフレームワーク



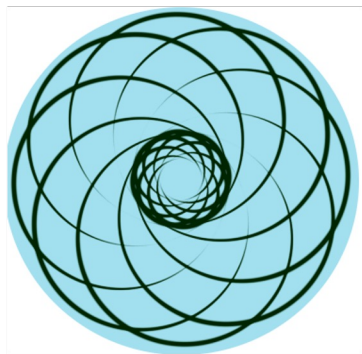
ユーザーには点線の部分が
個別のアプリに見える

前提:作っていたプラグインの性質

- 開発コスト高すぎ
 - がっつりプラグインの仕組みを理解していないと開発できない
 - デバッグが難しい
- バイナリに依存しまくり
 - ブラウザを更新するたびプラグインも更新してもらう必要がある



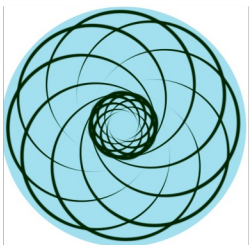
簡単に開発でき、コンパイルせずに動かせるマクロ言語が欲しい！



AliceScript

こんな言語が出来上がりました

こんな言語ができあがりました



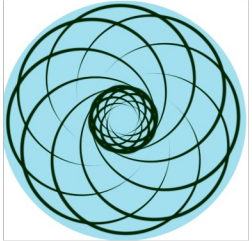
AliceScript

```
using Alice.Diagnostics;

namespace YTDownloader
{
    public const YTDLP = "yt-dlp.exe";
    public bool Download(string url, bool audioOnly = false, bool embedMetadata = false,
        bool useCookies = false, string prefix = null)
    {
        string args = GetCommandLine(url, audioOnly, embedMetadata, useCookies, prefix);
        var p = exec(YTDLP, args, true, true);
        return p.ExitCode == 0;
    }

    private string GetCommandLine(string url, bool audioOnly, bool embedMetadata, bool
        useCookies, string prefix)
    {
        string args = "";
        if(prefix != null)
        {
            prefix += "_";
        }
        string args = "-o \"" + prefix + "%(upload_date)s_%(title)s_%(id)s.%(ext)s" + "\" "
;
        if(audioOnly)
        {
            args += "-x --audio-format mp3 ";
        }
        else
        {
            args += "-f \"bestvideo[ext=mp4]+bestaudio[ext=m4a]/best[ext=mp4]/best\"
--embed-sub "
;
        }
        if(embedMetadata)
        {
            args += "--embed-metadata ";
        }
    }
}
```

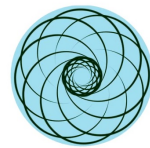
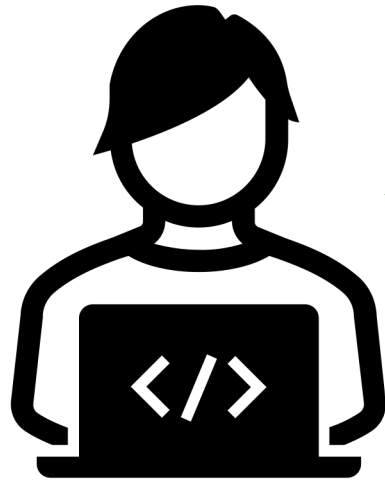
こんな言語ができあがりました



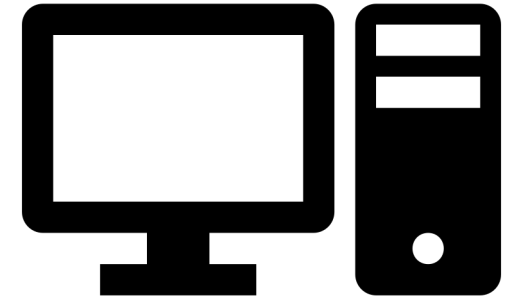
AliceScript

- 元々は.NETアプリ向けのマクロ言語
- 気に入ったので汎用言語化
- WindowsやmacOS、Linuxなど
様々な環境で動作
- たくさんのAPIが標準で使える

何を作ったの？



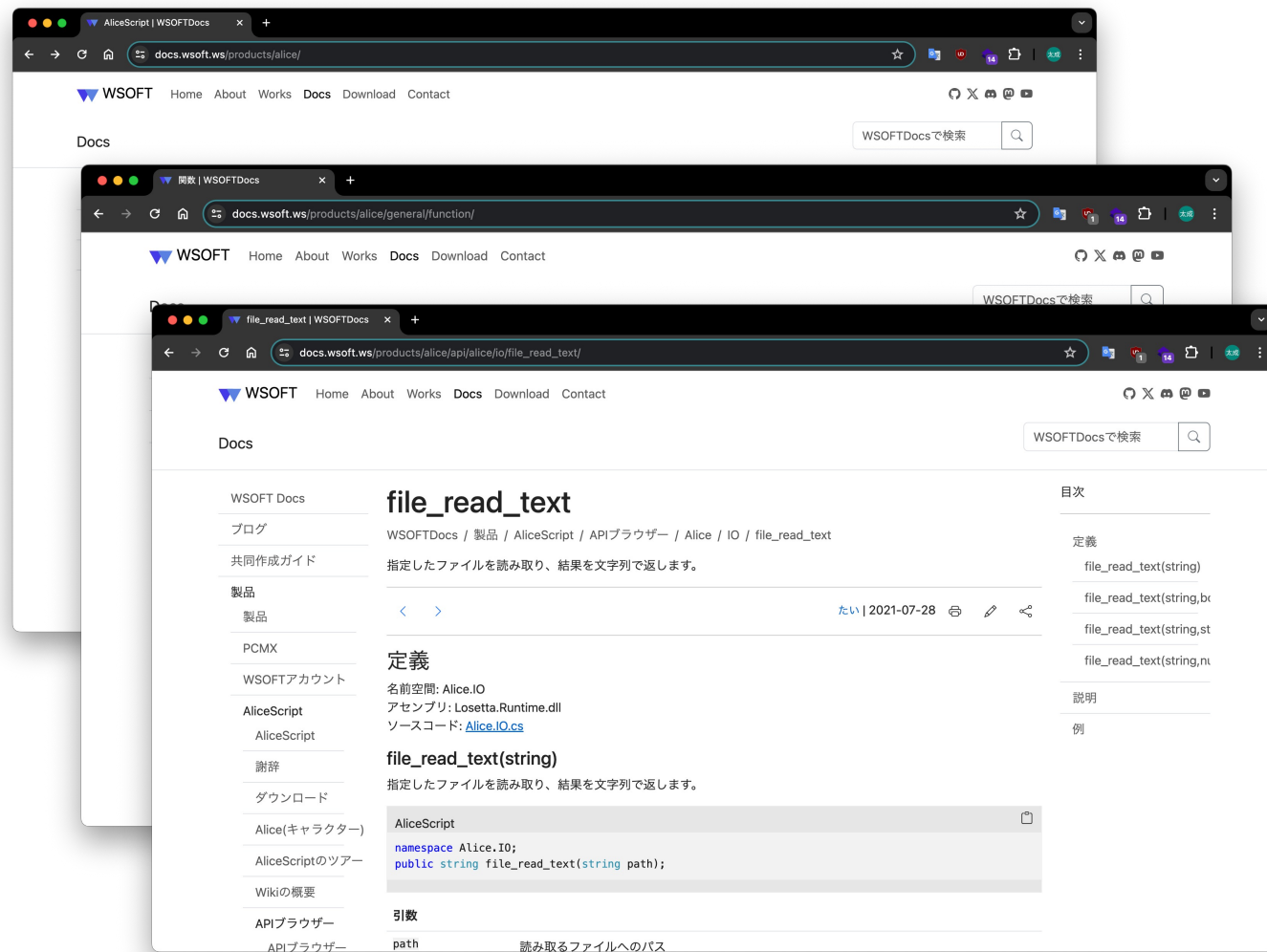
Alice Script



AliceScript Docs
プログラミング言語の説明

Losetta
プログラミング言語の実装

AliceScript Docs



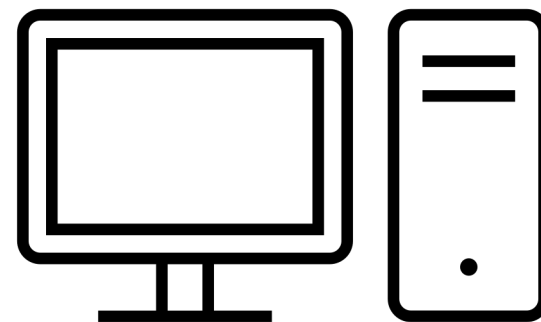
- 「AliceScript」って
- 調べると出てきます
- APIや概念について説明
- クラウド上で100%動作

Losetta The AliceScript Interpreter Platform

- よくいわれる例え



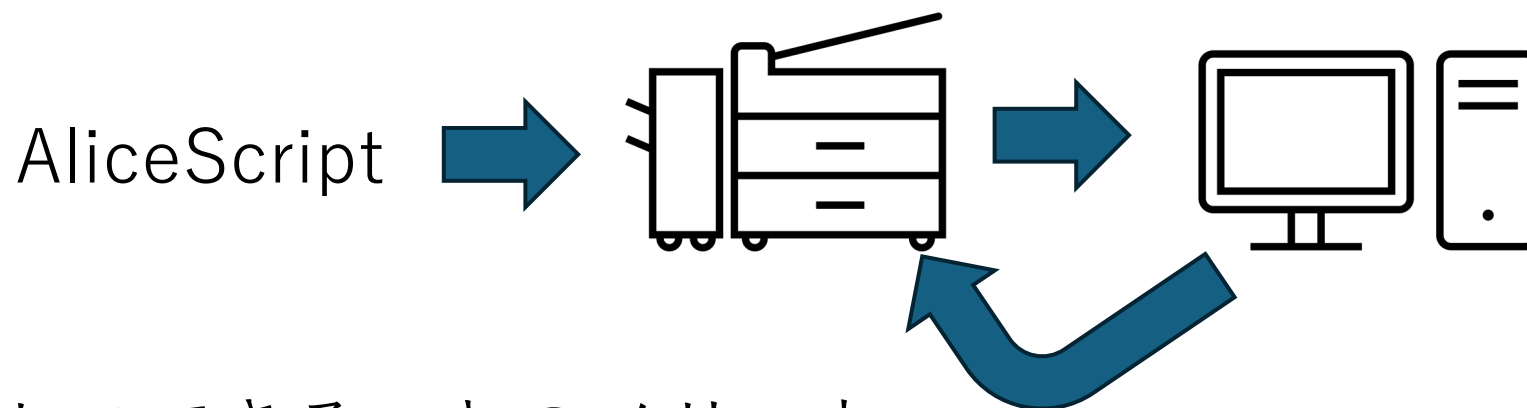
読んでそのたび
コンピューターに命令する



Losetta

The AliceScript Interpreter Platform

- インタプリターにアクセスできるように



- アクセスできることのメリット
 - AliceScriptから、コードを書き換えられる
 - 簡単に言語機能を拡張できる

AliceScriptの特徴

長いのでざっと3つ

特徴:名前空間

```
namespace MyLibrary
{
    public void Func()
    {
        //...
    }
    public string Message = "Hello,World!";

    private number Count = 0;
}

MyLibrary.Func();

print(MyLibrary.Message);

//これはエラー
print(MyLibrary.Count);
```

- C++やC#と同じ概念
- ファイルをフォルダで整理するように関数や変数を名前空間で整理
- アクセス修飾子を使うことで使わせたいものと使わせたくないものを分離

特徴: 契約プログラミング

```
// F : a / x の関数でxは0であってははいけない
public number F(number a, number x)
    requires(x != 0)
{
    return a / x;
}
```

- 関数の定義時に制約をつけることができる
- D言語を参考に導入

```
Alice /Users/taiseiue>>F(1,4);
0.25
Alice /Users/taiseiue>>F(1,0);
ASSERTION_ERROR(0x04c): この呼び出しは、関数が表明した事前条件を満たしませんでした
詳細情報: https://a.wsoft.ws/alice/exceptions/0x04c
スタックトレース
  場所 .structure function x();
  場所 .custom function F(a,x);
```

特徴: 契約プログラミング

```
// F : a / x の関数でxは0であってははいけない
public number F(number a, number x)
    requires(x != 0)
{
    return a / x;
}
```

- 関数の定義時に制約をつけることができる
- D言語を参考に導入
- 実行時コード書き換えで実現



```
public number F(number a, number x)
{
    //ここが生成コード
    Alice.Diagnostics.Assert(x != 0,
        "この呼び出しは、関数が表明した事前条件を満たしていません");

    //ここが処理本体
    return a / x;
}
```


特徴:プラットフォーム呼び出し

Beep(Int32, Int32)

ソース: [Console.cs](#)

周波数と時間を指定して、コンソールのスピーカーからビーブ音を出します。

C#

コピー

```
[System.Runtime.Versioning.SupportedOSPlatform("windows")]  
public static void Beep (int frequency, int duration);
```

パラメーター

frequency Int32

ビーブ音の周波数。37 Hz から 32767 Hz の範囲で指定できます。

duration Int32

ビーブ音の再生時間。ミリ秒単位で指定します。

属性 [SupportedOSPlatformAttribute](#)

例外

- .NETの静的メソッドをAliceScriptで定義して呼び出せる
- @netimportでどこにある関数かを伝える
- "extern"は実装が外部にある意
- 引数リストには変換先の型を書く

```
@netimport("System.Console","System.Console")  
extern void Beep(int frequency, int duration);
```

特徴: プラットフォーム呼び出し

Learn / Windows / アプリ / Win32 / API / Utilapiset.h /

Beep 関数 (utilapiset.h)

[アーティクル] • 2024/03/03 [フィードバック](#)

この記事の内容

- 構文
- パラメーター
- 戻り値
- 解説

さらに 2 個を表示

単純な音色をスピーカーから出します。関数は同期です。警告可能な待機を実行し、サウンドが終了するまで呼び出し元に制御を返しません。

構文

```
C++  
BOOL Beep(  
    [in] DWORD dwFreq,  
    [in] DWORD dwDuration  
);
```

- OSや他のライブラリで公開されている関数も呼び出せる
- @libimportでどこにある関数かを伝える
- "extern"は実装が外部にある意
- 引数リストには変換先の型を書く

```
@libimport("kernel32.dll")  
extern bool Beep(DWORD dwFreq, DWORD dwDuration);
```

特徴: プラットフォーム呼び出し

```
/// <summary>
/// 指定された式が真と評価されたときに、本文を実行します
/// </summary>
/// <param name="script">このブロックがあるスクリプト</param>
/// <param name="func">この関数がバインドされるFunctionBase</param>
/// <param name="condition">本文を実行するかどうかを決める条件</param>
/// <returns>本文の実行結果</returns>
0 個の参照
public static Variable If(ParsingScript script, BindFunction func, bool condition)
{
    Variable result = Variable.EmptyInstance;

    if (condition)
    {
        result = script.ProcessBlock();

        if (result is not null && (result.IsReturn ||
            result.Type == Variable.VarType.BREAK ||
            result.Type == Variable.VarType.CONTINUE))
        {
            // if文中で早期リターンしたからブロックごと飛ばす
            script.SkipBlock();
        }
        script.SkipRestBlocks();

        //return result;
    }
    else
    {
        // elseブロックのためifを飛ばす
        script.SkipBlock();
    }

    ParsingScript nextData = new ParsingScript(script);
    nextData.ParentScript = script;

    string nextToken = Utils.GetNextToken(nextData, false, true);
```

```
if(condition)
{
    //...
}
```

- 標準APIも構文もプラットフォーム呼び出しでできている
- マーシャリングと
バインディングで実現

特徴:プラットフォーム呼び出し

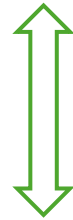
- マーシャリング(Marshalling)



特徴: プラットフォーム呼び出し

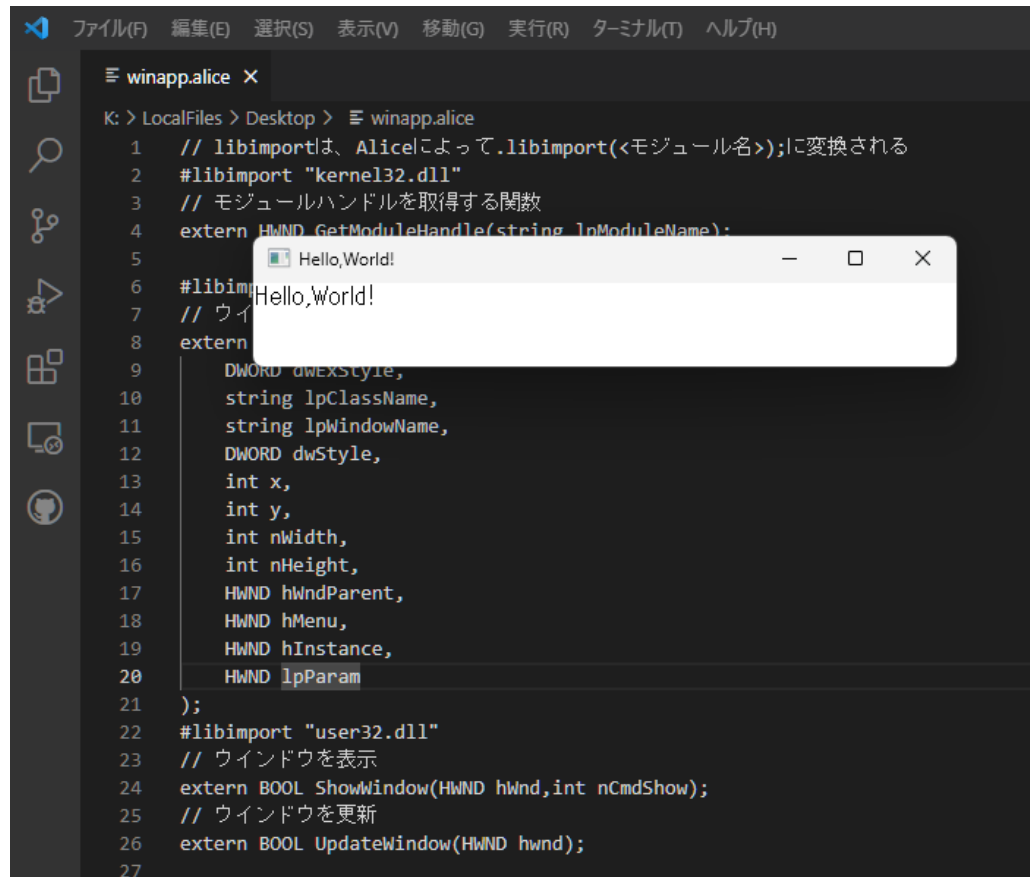
- バインディング(Binding)

```
print("Hello,World!");
```



```
public static void Print(string text)
{
    AddOutput(text);
}
```

特徴: プラットフォーム呼び出し



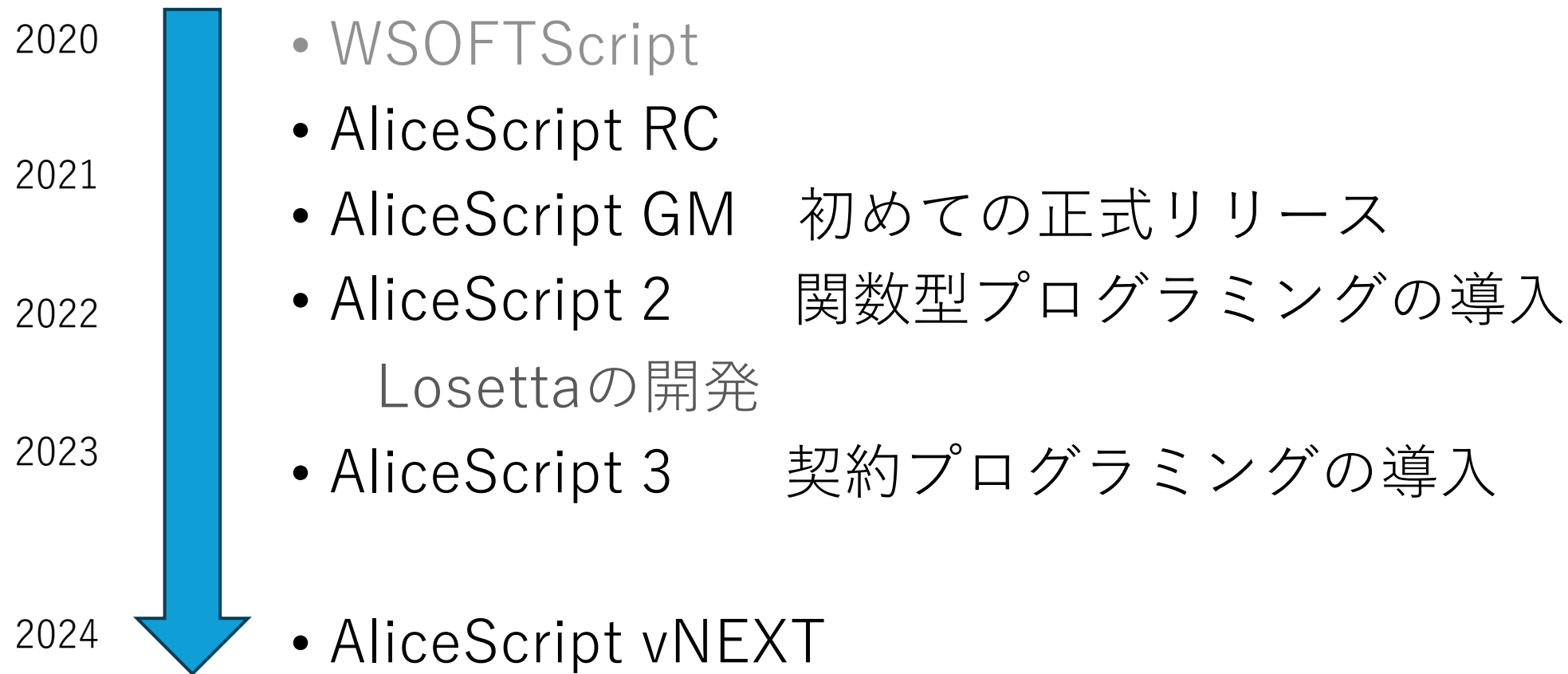
```
ファイル(F) 編集(E) 選択(S) 表示(V) 移動(G) 実行(R) ターミナル(T) ヘルプ(H)
≡ winapp.alice X
K: > LocalFiles > Desktop > ≡ winapp.alice
1 // libimportは、Aliceによって.libimport(<モジュール名>);に変換される
2 #libimport "kernel32.dll"
3 // モジュールハンドルを取得する関数
4 extern HMODULE GetModuleHandle(string lpModuleName);
5
6 #libimport "user32.dll"
7 // ウィンドウを表示
8 extern BOOL ShowWindow(HMODULE hModule, int nCmdShow);
9
10 DWORD dwExStyle,
11 string lpClassName,
12 string lpWindowName,
13 DWORD dwStyle,
14 int x,
15 int y,
16 int nWidth,
17 int nHeight,
18 HWND hWndParent,
19 HWND hMenu,
20 HWND hInstance,
21 HWND lpParam
22 );
23 #libimport "user32.dll"
24 // ウィンドウを表示
25 extern BOOL ShowWindow(HMODULE hModule, int nCmdShow);
26 // ウィンドウを更新
27 extern BOOL UpdateWindow(HWND hWnd);
```

- 上記を駆使すれば、理論上大規模なアプリだって作れる

AliceScriptのこれから

いままでの歴史とこれからの計画と

これまでのAliceScript



言語開発を通して

- 自分の知らないことが知れる
 - “使い方は知ってる”のに“作り方は知らない”ことはよくある
 - 言語機能が“なぜ”あるか、考える時間ができる
- 自分でも技術基盤が作れる
 - 自作の基盤の上で何かが動くのは楽しい
 - リバースエンジニアリングが得意になる

AliceScriptのこれから

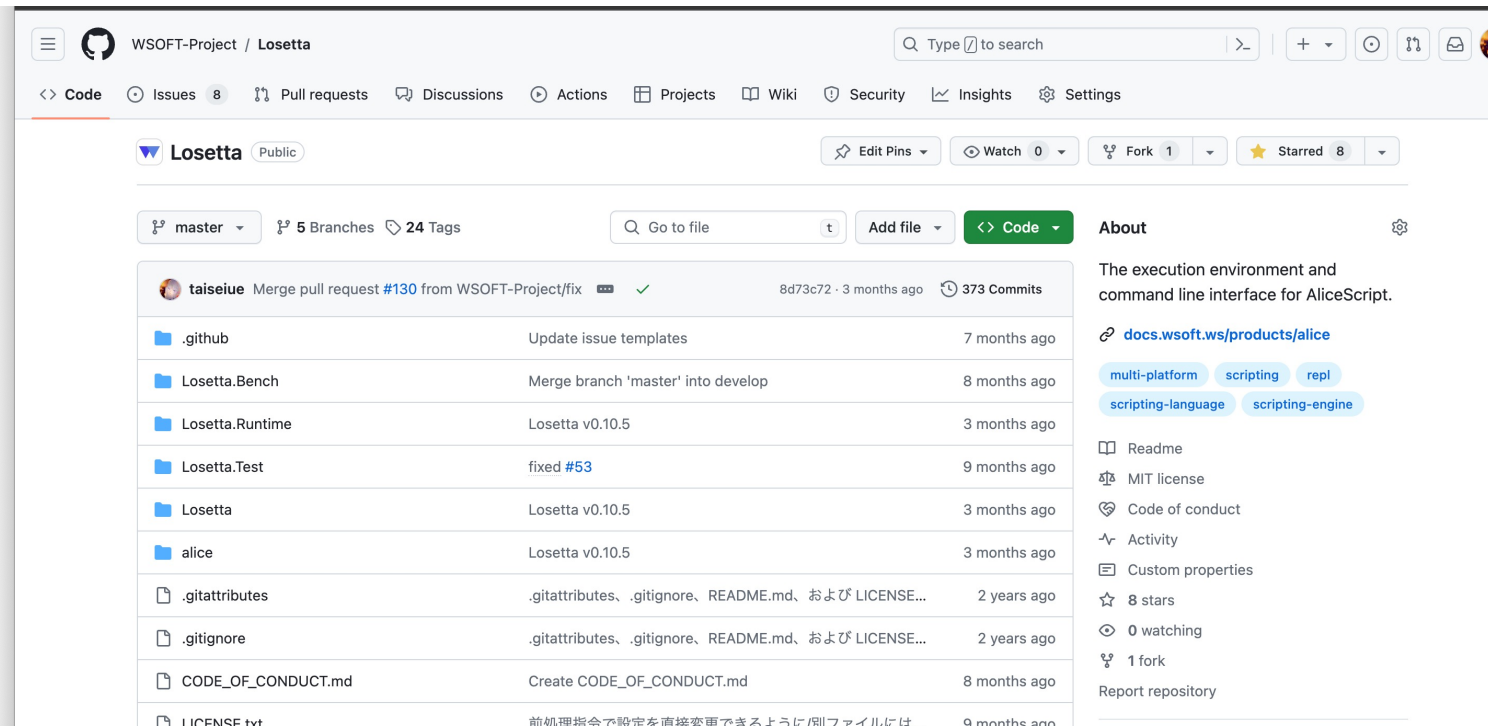
- 例年通りできれば新機能を10月に締め切り、
トリアージ後AliceScript 4を冬ごろに公開予定
- 事前コンパイルやIDEのサポートも構想中
- メディア化企画「Aliceたん」も進行中



a.wsoft.ws/alice/me

コメント募集中

- 新機能のアイデアや質問など…
- X(Twitter): @taiseiue
- github.com/WSOFT-Project/Losetta



コメント募集中

- 新機能のアイデアや質問など…
- X(Twitter): @taiseiue
- github.com/WSOFT-Project/alicescript

WSOFT-Project / alicescript

Search: Type to search

Code Issues 1 Pull requests Discussions Actions Projects Wiki Security Insights Settings

[PROPOSE]:注釈(名称未定) #1

Open taiseiue opened this issue yesterday · 2 comments

taiseiue commented yesterday · edited Member

注釈(名称未定)

概要

JavaのアノテーションやC#の属性のように、インタプリタやスクリプト内である式に追加のデータを通知できる機能が欲しい。

動機

デバッグモードのみ実行できる関数を定義したり、非推奨であることを通知したりすることは、コメントでは不十分で、修飾子では過剰で引数が使用できない。例えば、この機能によって関数定義時にもう使用して欲しくないことを通知するために、次のようなコードを書ける。

```
#obsolete "func_newを使用してください"
public virtual void func_old();
```

Assignees: No one—[assign yourself](#)

Labels: **enhancement**

Projects: None yet

Milestone: No milestone

Development: [Create a branch](#) for this issue or link a pull request.

Notifications: [Customize](#)