

Let's read the Julia documentation in your preferred language

もう一つの「two language problem」を解決する

Satoshi Terasaki (@AtelierArith) | Julia Japan 2025

「two language problem」は解決された ...はず？

- Julia は C の速度と Python の書きやすさを両立
- しかし、私たちには「もう一つの壁」が
- **英語のドキュメント vs 母国語**
- 専門用語が多い英語のドキュメントは、非ネイティブにとって大きな認知的負荷です。



Solution: DocstringTranslation.jl

ドキュメントを「読む」体験を変える

OpenAI API を用いてREPL 上で、関数のドキュメントを
母国語に自動翻訳

- OPENAI_API_KEY を用意
- `julia> using DocstringTranslation; @switchlang! :ja`
- Julia のヘルプモード "?" とシームレスに統合



```
> julia
```

```
      _      _ _ _ _ _      | Documentation: https://docs.julialang.org
  _ _      _ _ _ _ _      |
  _ _      _ _ _ _ _      | Type "?" for help, "]"? for Pkg help.
  | | | | | | | / _ ` |
  | | | | | | | ( _ | |
  _/ | \ _ _ ' _ | _ | \ _ _ ' _ | Official https://julialang.org release
  | _/ |
```

```
help?> sin
```

```
search: sin sinh asin sind sign sinc min in isinf using asind
```

```
sin(x::T) where {T <: Number} -> float(T)
```

Compute sine of x , where x is in radians.

Throw a [DomainError](#) if `isinf(x)`, return a `T(NaN)` if `isnan(x)`.

See also [sind](#), [sinpi](#), [sincos](#), [cis](#), [asin](#).

Examples

```
=====
```

```
julia> round.(sin.(range(0, 2pi, length=9)'), digits=3)
1x9 Matrix{Float64}:
 0.0  0.707  1.0  0.707  0.0 -0.707 -1.0 -0.707 -0.0
```

```
julia> sind(45)
0.7071067811865476
```

```
julia> sinpi(1/4)
0.7071067811865475
```

```
julia> round.(sincos(pi/6), digits=3)
(0.5, 0.866)
```

```
julia> round(cis(pi/6), digits=3)
0.866 + 0.5im
```

```
> julia
```

```
      _      _ _ _ _ _      | Documentation: https://docs.julialang.org
  _ _      _ _ _ _ _      |
  _ _      _ _ _ _ _      | Type "?" for help, "]"? for Pkg help.
  | | | | | | | / _ ` |
  | | | | | | | ( _ | |
  _/ | \ _ _ ' _ | _ | \ _ _ ' _ | Official https://julialang.org release
  | _/ |
```

```
julia> using DocstringTranslation
```

```
julia> @switchlang! :ja
"ja"
```

```
help?> sin
```

```
search: sin sinh asin sind sign sinc min in isinf using asind
```

```
sin(x::T) where {T <: Number} -> float(T)
```

x の正弦を計算します。 x はラジアンで指定されます。

`isinf(x)`の場合は[DomainError](#)をスローし、`isnan(x)`の場合は`T(NaN)`を返します。

他に[sind](#)、[sinpi](#)、[sincos](#)、[cis](#)、[asin](#)も参照してください。

例

```
=====
```

```
julia> round.(sin.(range(0, 2pi, length=9)'), digits=3)
1x9 Matrix{Float64}:
 0.0  0.707  1.0  0.707  0.0 -0.707 -1.0 -0.707 -0.0
```

```
julia> sind(45)
0.7071067811865476
```

```
julia> sinpi(1/4)
0.7071067811865475
```

```
julia> round.(sincos(pi/6), digits=3)
(0.5, 0.866)
```


https://atelierarith.github.io/UnofficialJuliaDoc-ja/

Julia Documentation

Julia 1.12 Documentation

Welcome to the documentation for Julia 1.12.

Please read the [release notes](#) to see what has changed since the last release.

Note

The documentation is also available in PDF format: [julia-1.12.2.pdf](#).

Important Links

Below is a non-exhaustive list of links that will be useful as you learn and use the Julia programming language.

- [Julia Homepage](#)
- [Install Julia](#)
- [Discussion forum](#)
- [Julia YouTube](#)
- [Find Julia Packages](#)
- [Learning Resources](#)
- [Read and write blogs on Julia](#)

Introduction

Scientific computing has traditionally required the highest performance, yet domain experts have largely moved to slower dynamic languages for daily work. We believe there are many good reasons to prefer dynamic languages for these applications, and we do not expect their use to diminish. Fortunately, modern language design and compiler techniques make it possible to mostly eliminate the performance trade-off and provide a single environment productive enough for prototyping and efficient enough for deploying performance-intensive applications. The Julia programming language fills this role: it is a flexible dynamic language, appropriate for scientific and numerical computing, with performance comparable to traditional statically-typed languages.

Because Julia's compiler is different from the interpreters used for languages like Python or R, you may find that Julia's performance is unintuitive at first. If you find that something is slow, we highly recommend reading through the [Performance Tips](#) section before trying anything else. Once you understand how Julia works, it is easy to write code that is nearly as fast as C.

Julia Documentation

Julia 1.12 Documentation

Welcome to the documentation for Julia 1.12.

[release notes](#)を読んで、前回のリリース以降に何が変更されたかを確認してください。

Note

The documentation is also available in PDF format: [julia-1.12.1.pdf](#).

Important Links

以下は、Juliaプログラミング言語を学び、使用する際に役立つリンクの非網羅的なリストです。

- [Julia Homepage](#)
- [Install Julia](#)
- [Discussion forum](#)
- [Julia YouTube](#)
- [Find Julia Packages](#)
- [Learning Resources](#)
- [Read and write blogs on Julia](#)

Introduction

科学計算は伝統的に最高のパフォーマンスを要求してきましたが、ドメインの専門家は日常の作業のために主に遅い動的言語に移行しています。私たちは、これらのアプリケーションに動的言語を好む多くの良い理由があると考えており、その使用が減少することはないと予想しています。幸いなことに、現代の言語設計とコンパイラ技術により、パフォーマンスのトレードオフをほぼ排除し、プロトタイピングに十分生産的で、パフォーマンス集約型アプリケーションの展開に十分効率的な単一の環境を提供することが可能になりました。Juliaプログラミング言語はこの役割を果たします：それは科学計算および数値計算に適した柔軟な動的言語であり、従来の静的型付け言語と同等のパフォーマンスを持っています。


JuliaのコンパイラはPythonやRのような言語で使用されるインタプリタとは異なるため、最初はJuliaのパフォーマンスが直感的でないと感じるかもしれません。何かが遅いと感じた場合は、他のことを試す前に[Performance Tips](#)セクションを読むことを強くお勧めします。Juliaの動作を理解すれば、Cとほぼ同じくらい速いコードを書くのは簡単です。



Juliaを、すべての人に。

母国語でドキュメントを読み、より深く、より速く Julia を理解しましょう。

 [AtelierArith/DocstringTranslation.jl](#)

 [AtelierArith/UnofficialJuliaDoc-ja](#)

Julia アドベントカレンダー 2025 の 8 日目の記事

Image Sources



https://upload.wikimedia.org/wikipedia/commons/1/1f/Julia_Programming_Language_Logo.svg

Source: it.wikipedia.org



https://gurmentor.com/wp-content/uploads/2020/04/gurmentor.com-what-are-the-challenges-in-learning-a-new-language-2020-04-20_19-55-23_945100.jpg

Source: gurmentor.com



https://img.freepik.com/premium-vector/futuristic-robot-reading-book-artificial-intelligence-learning_831490-4650.jpg

Source: www.freepik.com