

# What is Standard Schema?




@macchiitaka at Mita.ts #4 (2025-02-26)

2024 年 12 月某日、`Zod` のリリースノートにて `@standard-schema/spec` なる仕様に遭遇

<https://github.com/colinhacks/zod/releases/tag/v3.24.0>

## v3.24.0

[Compare](#)

 github-actions released this Dec 10, 2024 · 20 commits to refs/heads/main since this release  v3.24.0  b333f96

### Implement `@standard-schema/spec`

This is the first version of Zod to implement the [Standard Schema](#) spec. This is a new community effort among several validation library authors to implement a common interface, with the goal of simplifying the process of integrating schema validators with the rest of the ecosystem. Read more about the project and goals [here](#).

@standard-schema/spec ?



作成者曰く...

これは、Zod (👋)、Valibot、ArkType の作成者が、相互運用性を促進するために共同で作成した、すべて TypeScript スキーマライブラリが実装する「共通インターフェース」の仕様です。



Colin McDonnell



@colinhacks

...

BIG DAY. Introducing Standard Schema 1.0!

It's a specification for a "common interface" to be implemented by all TypeScript schema libraries, written collaboratively by the the creators of Zod (👋), Valibot, and ArkType to promote interoperability.

[ポストを翻訳](#)

<https://x.com/colinhacks/status/1883907825384190418>

≡ TypeScript スキーマライブラリの「共通の型」を定義した

≡ Standard Schema は TypeScript の型定義

Standard Schema が生まれる前の世界



スキーマライブラリと組み合わせて使うライブラリやフレームは  
APIの違いを吸収するための `Resolver` や `Plugin` を提供している

- React Hook Form
- Hono
- tRPC
- TanStack Router
- ...



react-hook-form/resolvers を覗いてみよう

<https://github.com/react-hook-form/resolvers>

## Links

---

- [React-hook-form validation resolver documentation](#)

## Supported resolvers

- [Install](#)
- [Links](#)
  - [Supported resolvers](#)
- [API](#)
- [Quickstart](#)
  - [Yup](#)
  - [Zod](#)
  - [Superstruct](#)
  - [Joi](#)
  - [Vest](#)
  - [Class Validator](#)
  - [io-ts](#)
  - [Nope](#)
  - [computed-types](#)
  - [typanion](#)
  - [Ajv](#)
  - [TypeBox](#)
    - [With](#) `ValueCheck`
    - [With](#) `TypeCompiler`
  - [ArkType](#)
  - [Valibot](#)
  - [TypeSchema](#)
  - [effect-ts](#)
  - [VineJS](#)
  - [fluentvalidation-ts](#)
  - [standard-schema](#)
- [Backers](#)
  - [Sponsors](#)
- [Contributors](#)

## API

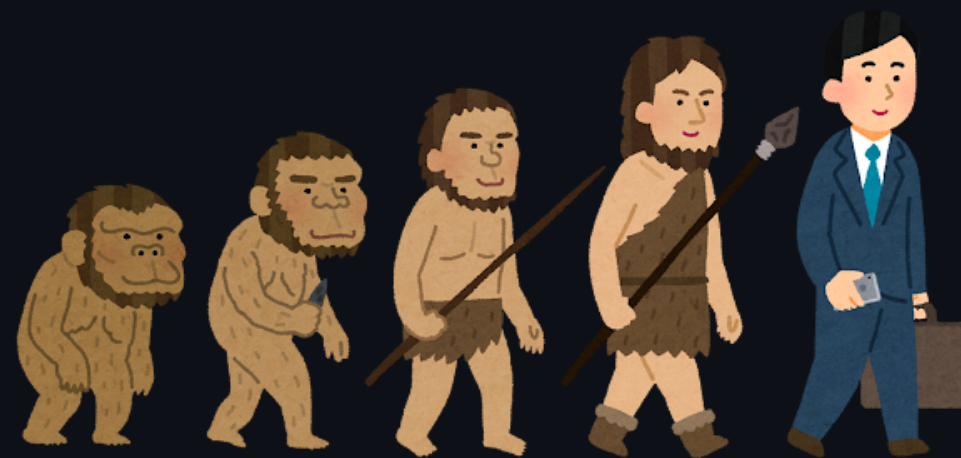
---

スキーマライブラリ毎に Resolver を提供している

ツライ



Standard Schema がある世界



スキーマライブラリが Standard Schema に対応していれば...



連携するライブラリは Standard Schema 向けの

`Resolver` や `Plugin` だけ提供すれば OK

簡単に連携できる！



つまり嬉しいのはだれ？



スキーマライブラリを利用するライブラリやフレームワークの作者

具体のスキーマ定義と実装サンプルを見てみよう

スキーマ

<https://github.com/standard-schema/standard-schema/blob/main/packages/spec/src/index.ts>

```

1  /** The Standard Schema interface. */
2  export interface StandardSchemaV1<Input = unknown, Output = Input> {
3      /** The Standard Schema properties. */
4      readonly "<standard>": StandardSchemaV1.Props<Input, Output>;
5  }
6
7  export declare namespace StandardSchemaV1 {
8      /** The Standard Schema properties interface. */
9      export interface Props<Input = unknown, Output = Input> {
10         /** The version number of the standard. */
11         readonly version: 1;
12         /** The vendor name of the schema library. */
13         readonly vendor: string;
14         /** Validates unknown input values. */
15         readonly validate: (
16             value: unknown
17         ) => Result<Output> | Promise<Result<Output>>;
18         /** Inferred types associated with the schema. */
19         readonly types?: Types<Input, Output> | undefined;
20     }
21
22     /** The result interface of the validate function. */
23     export type Result<Output> = SuccessResult<Output> | FailureResult;
24
25     /** The result interface if validation succeeds. */
26     export interface SuccessResult<Output> {
27         /** The typed output value. */
28         readonly value: Output;
29         /** The non-existent issues. */
30         readonly issues?: undefined;
31     }
32
33     /** The result interface if validation fails. */
34     export interface FailureResult {
35         /** The issues of failed validation. */
36         readonly issues: ReadonlyArray<Issue>;
37     }
38
39     /** The issue interface of the failure output. */
40     export interface Issue {
41         /** The error message of the issue. */
42         readonly message: string;
43         /** The path of the issue, if any. */
44         readonly path?: ReadonlyArray<PropertyKey | PathSegment> | undefined;
45     }
46
47     /** The path segment interface of the issue. */
48     export interface PathSegment {
49         /** The key representing a path segment. */
50         readonly key: PropertyKey;
51     }
52
53     /** The Standard Schema types interface. */
54     export interface Types<Input = unknown, Output = Input> {
55         /** The input type of the schema. */
56         readonly input: Input;
57         /** The output type of the schema. */
58         readonly output: Output;
59     }
60
61     /** Infers the input type of a Standard Schema. */
62     export type InferInput<Schema extends StandardSchemaV1> = NonNullable<
63         Schema["<standard>"]["types"]
64     >["input"];
65
66     /** Infers the output type of a Standard Schema. */
67     export type InferOutput<Schema extends StandardSchemaV1> = NonNullable<
68         Schema["<standard>"]["types"]
69     >["output"];
70
71     // biome-ignore lint/complexity/noUselessEmptyExport: needed for granular visibility control of TS namespace
72     export {};
73 }
74

```

- 100 行未満のコード
- シンプル
- バリデーターの呼び出し方、実行結果、エラーメッセージと位置
- `~standard` プロパティ配下の実装する
  - `~` で始まるプロパティは自動補完の優先順位が低い

スキーマライブラリの実装サンプル

<https://github.com/standard-schema/standard-schema/blob/main/packages/examples/implement.ts>



```
1  import type { StandardSchemaV1 } from "@standard-schema/spec";
2
3  // Step 1: Define the schema interface
4  interface StringSchema extends StandardSchemaV1<string> {
5    type: "string";
6    message: string;
7  }
8
9  // Step 2: Implement the schema interface
10 function string(message = "Invalid type"): StringSchema {
11   return {
12     type: "string",
13     message,
14     "~standard": {
15       version: 1,
16       vendor: "valizod",
17       validate(value) {
18         return typeof value === "string"
19           ? { value }
20           : { issues: [{ message, path: [] }] };
21       },
22     },
23   };
24 }
25
26 const schema = string();
27 schema["~standard"].validate("hello");
28
```

スキーマライブラリを呼び出すフレームワークの実装サンプル

<https://github.com/standard-schema/standard-schema/blob/main/packages/examples/integrate.ts>





```
1  /**
2   * This is an example showing how to accept Standard Schemas in a generic way.
3   */
4
5  import type { StandardSchemaV1 } from "@standard-schema/spec";
6
7  import * as z from "zod";
8  import * as v from "valibot";
9  import { type } from "arktype";
10
11 export async function standardValidate<T extends StandardSchemaV1>({
12   schema: T,
13   input: StandardSchemaV1.InferInput<T>
14 }): Promise<StandardSchemaV1.InferOutput<T>> {
15   let result = schema["~standard"].validate(input);
16   if (result instanceof Promise) result = await result;
17
18   // if the `issues` field exists, the validation failed
19   if (result.issues) {
20     throw new Error(JSON.stringify(result.issues, null, 2));
21   }
22
23   return result.value;
24 }
25
26 const zodResult = await standardValidate(z.string(), "hello");
27 const valibotResult = await standardValidate(v.string(), "hello");
28 const arktypeResult = await standardValidate(type("string"), "hello");
29
30 console.log({
31   zodResult,
32   valibotResult,
33   arktypeResult,
34 });
35
```

Zod、Valibot、ArkType とひとつの Resolver で連携できる

幸せな世界 🥳🥳🥳

## Appendix

- Standard Schema <https://standardschema.dev/>
- standard-schema/standard-schema: A standard interface for TypeScript schema validation libraries <https://github.com/standard-schema/standard-schema>
- Standard Schema リリース  
<https://x.com/colinhacks/status/1883907825384190418>
- Standard Schema Proposal <https://x.com/colinhacks/status/1634284724796661761>
- Release v3.24.0 · colinhacks/zod  
<https://github.com/colinhacks/zod/releases/tag/v3.24.0>
- Release v4.7.0 · honojs/hono <https://github.com/honojs/hono/releases/tag/v4.7.0>