Zod

https://nathankrasney.com/

What is zod

- Validation library
- TypeScript-first schema declaration and validation library.
- The goal is to eliminate duplicative type declarations (probably by inferring the interface from the schema and use it e.g. in ZodError)
- With Zod, you declare a validator *once* and Zod will automatically infer the static TypeScript type

https://nathankrasney.com/

Motivation for zod

- Simple
- Designed to be as developer-friendly as possible
- Popular - from 600K weekly downloads in june 2022 to 3.5M in june 2023
- Works in node and browsers
- Works also in javascript not just typescript
- The error thrown are super informative

https://nathankrasney.com/

Motivation for zod for me

- Use all ready defined interface from types directory to define the zod schema - is it possible ??
- Simple validation library

https://nathankrasney.com/

Use cases

Client

- Validation of form input
- Dynamic type check of json received from a server (check my video for motivation but validation here with json schema)

Server

- Validate info from client via http request
- Validate data into schema based database SQL

Simple string validation - safeParse

```
function validateStringSafeParse(
  val: any
): SafeParseReturnType<string, string> {
  const schema = z.string();
  return schema.safeParse(val);
}
…..
console.log(validateStringSafeParse("111")); //  { success: true, data: "111" }
console.log(validateStringSafeParse(111)); // { success: false, error: Getter }
```

Define the schema

Check value against the schema
safeParse **return an object** with
success : true\false and data | error

string-validators.ts (ver 0.1)

https://nathankrasney.com/

Simple string validation - parse

```
function validateStringParse(val: any): void {
  const schema = z.string();
  schema.parse(val);
}
……
validateStringParse("111"); // ok
validateStringParse(111); // throw
```

Define the schema

Check value against the schema
Throw if not valid

```
ZodError: [
  {
    "code": "invalid_type",
    "expected": "string",
    "received": "number",
    "path": [],
    "message": "Expected string, received number"
  }
]
```

Very informative info thrown

string-validators.ts (ver 0.1)

https://nathankrasney.com/

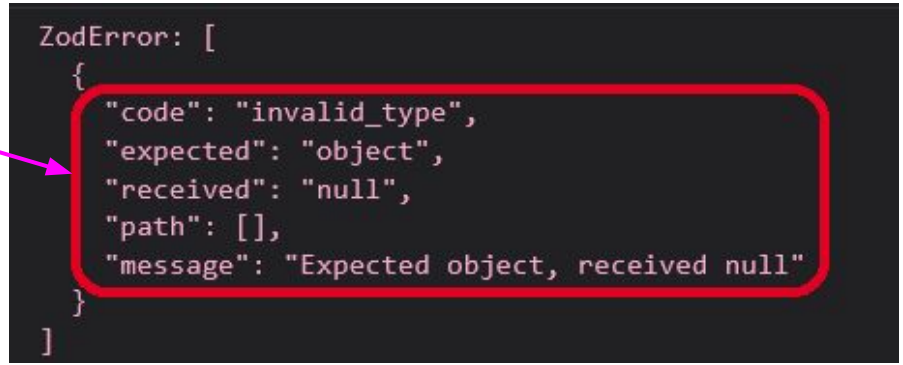
Validate object - sample 1

```
function validatePerson(person: any): void {
  const schemaUser = z.object({name: z.string(),age: z.number(),});
  schemaUser.parse(person);
}
```

object-validators.ts (ver 0.3)

Check value against the schema
Throw if not valid

```
……
const personOk: IPerson = {name: "John Doe",age: 10,};
validatePerson(personOk); // do not throw
validatePerson(null); // throw
```



https://nathankrasney.com/

Validate object - sample 2

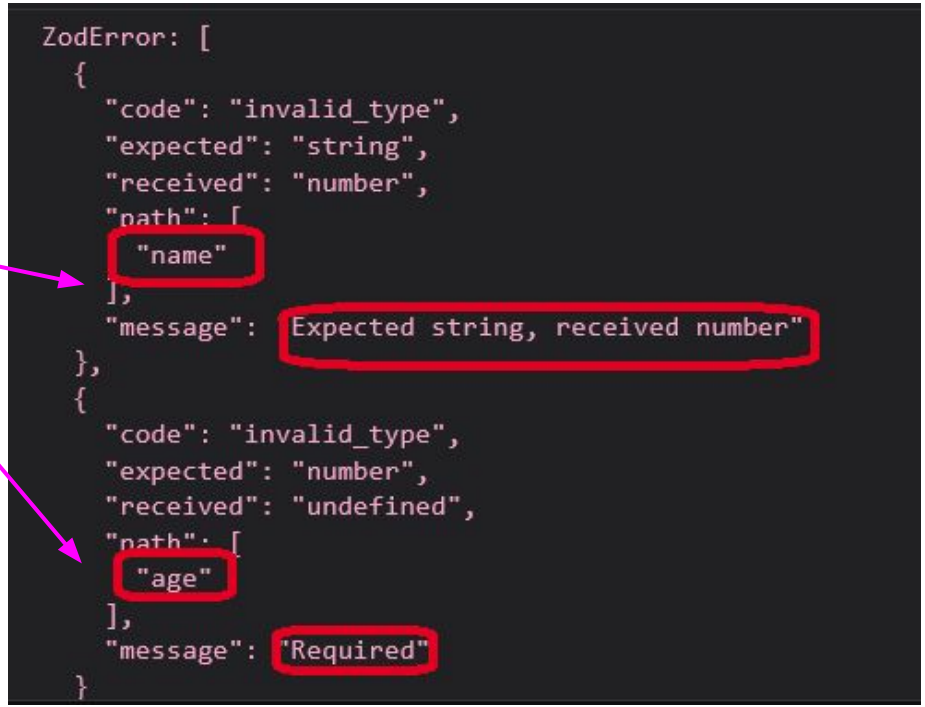
validatePerson({name : 11}); // should throw

Same function from last slide

……

The error are super informative !!!!!!!

Path allow to refer to the object properties



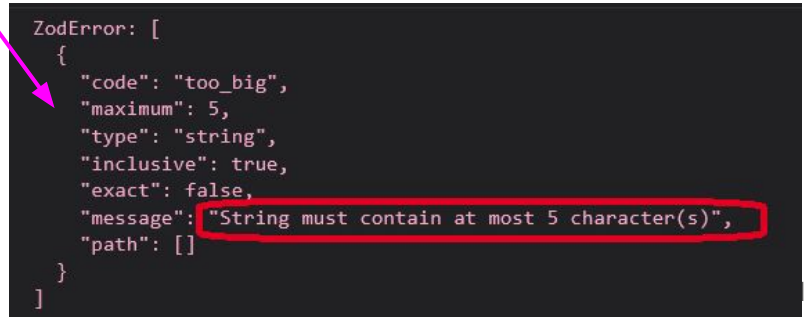
https://nathankrasney.com/

Validate string with limits

```
function validateStringMinMax(val: any): void {
  const schema = z.string().min(3).max(5);
  schema.parse(val);
}
....
validateStringMinMax("ab12"); // ok , not throw
validateStringMinMax("ab1222");; // throw
```

Define the schema

Check value against the schema
Throw if not valid



https://nathankrasney.com/

Validate array

```
export function validateLinkedinProfilesUrl(
  profileUrls: any
): z.SafeParseReturnType<string[], string[]> {
  const schema = z
    .string()
    .startsWith("https://www.linkedin.com/in/")
    .url()
    .array()
    .nonempty();
  return schema.safeParse(profileUrls);
}
```

Verify that the input is non empty array of strings which are linkedin profile url

Code here (ver 0.32)

https://nathankrasney.com/

Error handling

zod provides a sub class for error handling called ZodError - check in one of the following slides

```
if (!validationResult.success) {
…
    console.log(validationResult.error.errors);
    const formatted = validationResult.error.format();
    getNameError().innerText = formatted.name?._errors.join(" , ") ?? "";
    getEmailError().innerText = formatted.email?._errors.join(" , ") ?? "";
    getAgeError().innerText = formatted.age?._errors.join(" , ") ?? "";
…
}
```

Get all errors from all a schema properties

Get error per schema property

https://nathankrasney.com/

display errors concepts

You can use the `.format()` method to convert this error into a nested object.

```
const result = z
  .object({
    name: z.string(),
  })
  .safeParse({ name: 12 });

if (!result.success) {
  const formatted = result.error.format();
  /* {
    name: { _errors: [ 'Expected string, received number' ] }
  } */

  formatted.name?._errors;
  // => ["Expected string, received number"]
}
```

name is a key in the schema !!

https://nathankrasney.com/

Validate form sample 1/2

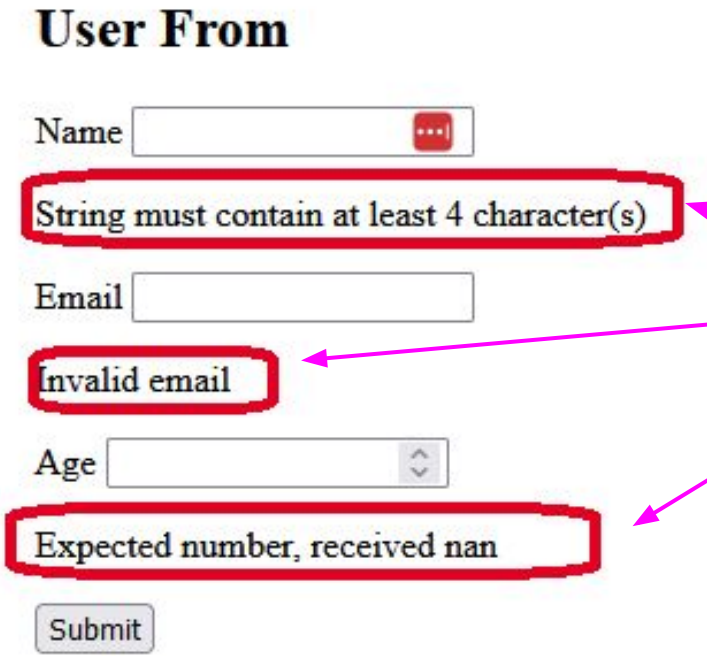
```
const validationResult = formZodSchema.safeParse({
    name: getNameVal(),
    email: getEmailVal(),
    age: getAgeVal(),
  });

  if (validationResult.success) {
    getNameError().innerText = getEmailError().innerText = getAgeError().innerText ="";
  } else {
    const formatted = validationResult.error.format();
    getNameError().innerText = formatted.name?._errors.join(" , ") ?? "";
    getEmailError().innerText = formatted.email?._errors.join(" , ") ?? "";
    getAgeError().innerText = formatted.age?._errors.join(" , ") ?? "";
  }
```

Check validateFormWithZod
form-validation.ts (ver 0.3)

ZodError

https://nathankrasney.com/

validate form sample 2/2



Following the code from prev slide - these are the errors we get for the schema

```
const formZodSchema = z.object({
  name: z.string().min(4).max(7),
  email: z.string().email(),
  age: z.number().min(18).max(65),
});
```

https://nathankrasney.com/

Important classes

```
type SafeParseReturnType<Input, Output> = SafeParseSuccess<Output> |
SafeParseError<Input>;
```

On success

```
type SafeParseSuccess<Output> = {
    success: true;
    data: Output;
};
```

On error

```
type SafeParseError<Input> = {
    success: false;
    error: ZodError<Input>;
};
```

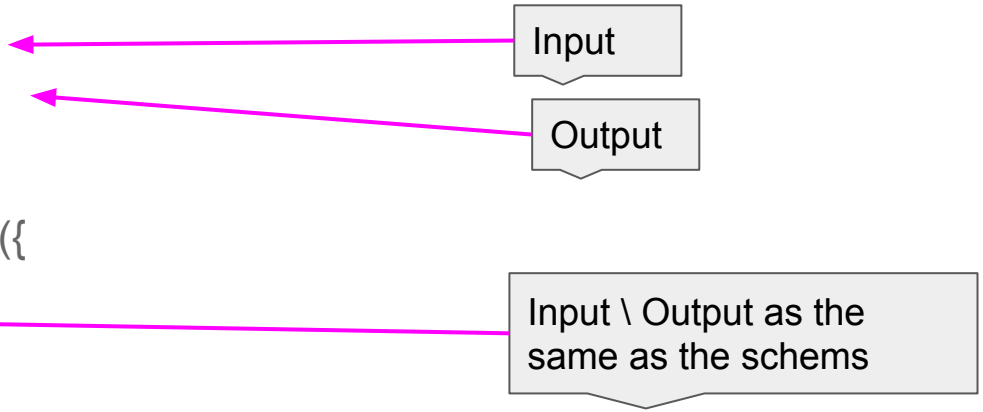
Return type of **parse** for success. Error throw ….

Return type of **safeParse**

https://nathankrasney.com/

Input \ Output example

```
function validatePersonSafeParse(person: any): SafeParseReturnType<
  { name: string;age: number;},
  { name: string;age: number;}
> {
  const schemaUser = z.object({
    name: z.string(),
    age: z.number(),
  });
  return schemaUser.safeParse(person);
}
```



https://nathankrasney.com/

ZodError

```
class ZodError<T = any> extends Error {
  issues: ZodIssue[];
  get errors(): ZodIssue[];
  constructor(issues: ZodIssue[]);
  format(): ZodFormattedError<T>;
  format<U>(mapper: (issue: ZodIssue) => U): ZodFormattedError<T, U>;
  static create: (issues: ZodIssue[]) => ZodError<any>;
  toString(): string;
  get message(): string;
  get isEmpty(): boolean;
  addIssue: (sub: ZodIssue) => void;
  addIssues: (subs?: ZodIssue[]) => void;
  flatten(): typeToFlattenedError<T>;
  flatten<U>(mapper?: (issue: ZodIssue) => U): typeToFlattenedError<T, U>;
  get formErrors(): typeToFlattenedError<T, string>;
}
```

T is the interface inferred from the schema **this is super powerful because you can access the schema properties !!!!**

errors is used to show all errors used in formatValidationError() here (ver 0.34)

format() is used my form sample (ver 0.32)

https://nathankrasney.com/

My repository

zod-validation-playground

https://nathankrasney.com/

questions

- Can i create schema using already existing interface?? This is the most important question for me - check here , check also ts-to-zod

references

Official docs

Zod Makes TypeScript Even Better - nov 2022

Fixing TypeScript's Blindspot: Runtime Typechecking - sep 2021

https://nathankrasney.com/