



**THANK YOU  
ORGANIZERS!**

# Ego Slide

- Push buttons
- Type things
- Break stuff
- Wear sweatshirt as cape
- This picture is not stack safe

**We ❤️ TS**

**The outside  
world is scary**



# Data Validation Libraries

```
import * as z from "zod"

const Todo = z.object({
  id: z.number(),
  name: z.string(),
  isCompleted: z.boolean(),
  createdAt: z.string().datetime()
})

export type Todo = z.infer<typeof Todo>
/* ^?
type Todo = {
  id: number;
  name: string;
  isCompleted: boolean;
  createdAt: string;
}
*/
```





# Manual encoding

```
import * as z from "zod"

const Todo = z.object({
  id: z.number(),
  name: z.string(),
  isCompleted: z.boolean(),
  createdAt: z.date()
})

export type Todo = z.infer<typeof Todo>

export function encode(todo: Todo){
  return {
    id: todo.id,
    name: todo.name,
    isCompleted: todo.isCompleted,
    createdAt: todo.createdAt.toString()
  }
}
```

# The cost of manual encoding

```
import * as z from "zod"

const Todo = z.object({
  id: z.number(),
  name: z.string(),
  isCompleted: z.boolean(),
  createdAt: z.date()
})

export type Todo = z.infer<typeof Todo>

export function encode(todo: Todo){
  return {
    id: todo.id,
    name: todo.name,
    isCompleted: todo.isCompleted,
    createdAt: todo.createdAt.toString() // ← BUG!
  }
}
```

# Property based testing

```
import * as fc from "fast-check"

const ArbitraryTodo = fc.record({
  id: fc.integer(),
  name: fc.string(),
  isCompleted: fc.boolean(),
  createdAt: fc.date()
})

const randomTodos = fc.sample(ArbitraryTodo, { seed: 42 })

for(const todo of randomTodos){
  const encodedTodo = encode(todo)
  const decodedTodo = decode(encodedTodo)
  expect(decodedTodo).toEqual(todo)
}
```

# This does not end here

- Decoding
- Encoding
- Assertion
- Fastcheck
- Equality
- JSONSchema

# One definition to rule them all

```
interface Todo {  
  id: number  
  name: string  
  isCompleted: boolean  
  createdAt: Date  
}
```

# Unfortunately won't work...

```
interface Todo {  
    id: number  
    name: string  
    isCompleted: boolean  
    createdAt: Date  
}  
  
decode<Todo>({id: 1, name: "Buy milk", isCompleted: false, createdAt: new Date()})
```

...gets compiled to...

```
decode({id: 1, name: "Buy milk", isCompleted: false, createdAt: new Date()})
```







# **@effect/schema**

A Library for defining and using schemas

# Giulio Canti

Author of other impressive TypeScript libraries

- `tcomb`
- `fp-ts`
- `io-ts`

```
import * as Schema from "@effect/schema/Schema"

export const Todo = Schema.Struct({
  id: Schema.Number,
  name: Schema.String,
  isCompleted: Schema.Boolean,
  completedAt: Schema.Date
})

/* ^? Schema.Struct<{
  id: typeof Schema.Number;
  name: typeof Schema.String;
  isCompleted: typeof Schema.Boolean;
  completedAt: typeof Schema.Date;
}> */

export type Todo = Schema.Schema.Type<typeof Todo>
// ^? { id: number; name: string; isCompleted: boolean; completedAt: Date; }
```



```
import * as Schema from "@effect/schema/Schema"

export const Todo = Schema.Struct({
  id: Schema.Number,
  name: Schema.String,
  isCompleted: Schema.Boolean,
  completedAt: Schema.Date
})

export type Todo = Schema.Schema.Type<typeof Todo>

console.log(Todo.ast)
```

```
import * as Schema from "@effect/schema/Schema"

const Todo = Schema.Struct({
  id: Schema.Number,
  name: Schema.String
})

const decodeTodoOrThrow = Schema.decodeUnknownSync(Todo)
/* ^? (u: unknown) => {
  readonly id: number;
  readonly name: string;
} */

const firstTodo = decodeTodoOrThrow({ id: 1, name: "Buy milk"}) // decode successfully
console.log("firstTodo", firstTodo)

const secondTodo = decodeTodoOrThrow({ id: 2, name: true }) // throws
```

```
import * as Schema from "@effect/schema/Schema"

const Todo = Schema.Struct({
  id: Schema.Number,
  name: Schema.String
})

const decodeTodo = Schema.decodeUnknownEither(Todo)
/* ^? (u: unknown, overrideOptions?: ParseOptions) => Either<{
  readonly id: number;
  readonly name: string;
}, ParseError> */

const successTodo = decodeTodo({ id: 1, name: "Buy milk" })
// ^? {_tag: "Right", right: { id: 1, name: "Buy milk" } }

const failureTodo = decodeTodo({ id: 2, name: true })
console.log("failureTodo", failureTodo)
// ^? {_tag: "Left", left: { message: "... ", ... } }
```

```
import * as Schema from "@effect/schema/Schema"

const Todo = Schema.Struct({
  id: Schema.Number,
  name: Schema.String
}).annotations({ identifier: "Todo" })

const decodeTodo = Schema.decodeUnknownEither(Todo)

const decodedTodo = decodeTodo({ id: 2, name: true })

if(decodedTodo._tag === "Left"){
  console.error(decodedTodo.left.message)
}else{
  console.log("success", decodedTodo.right)
}
```





decode: (input: unknown)  $\Rightarrow$  Todo

encode: (input: Todo)  $\Rightarrow$  unknown

```
import * as Schema from "@effect/schema/Schema"

const Todo = Schema.Struct({
  id: Schema.Number,
  name: Schema.String,
  isCompleted: Schema.Boolean,
  createdAt: Schema.Date
}).annotations({ identifier: "Todo" })

const encodeTodo = Schema.encodeEither(Todo)

const encodedTodo = encodeTodo({ id: 2, name: "Buy milk", isCompleted: false, createdAt: new Date() })

if(encodedTodo._tag === "Left"){
  console.error(encodedTodo.left.message)
}else{
  console.log("success", encodedTodo.right)
}
```

Schema<A, I>





# Schema



# Effect

```
import * as Schema from "@effect/schema/Schema"
import * as FastCheck from "@effect/schema/FastCheck"
import * as Arbitrary from "@effect/schema/Arbitrary"

const Todo = Schema.Struct({
  id: Schema.Int,
  name: Schema.String,
  isCompleted: Schema.Boolean,
  createdAt: Schema.Date
}).annotations({ identifier: "Todo" })

const ArbitraryTodo = Arbitrary.make(Todo)

console.log(FastCheck.sample(ArbitraryTodo, 2))
```



```
import * as Schema from "@effect/schema/Schema"
import * as JSONSchema from "@effect/schema/JSONSchema"

const Todo = Schema.Struct({
  id: Schema.Int,
  name: Schema.String,
  isCompleted: Schema.Boolean,
  createdAt: Schema.Date.annotations({ jsonSchema: { type: "string", format: "date-time" } })
}).annotations({ identifier: "Todo" })

const finalSchema = JSONSchema.make(Todo)

console.log(JSON.stringify(finalSchema, null, 2))
```

# Built-in @effect/Schema interpreters

- Decoding
- Encoding
- Asserting
- Arbitraries
- JSON Schema
- Equivalence
- Pretty Print

# Schema applications

- HTTP RPC Server (@effect/rpc)
- HTTP Api (@effect/platform)
- Generate static TS Typings
- Support AI training in outputting structured data

**Something is  
not in the list?**

# Happy Schem-ing!

Thanks for your time!

*[x.com/mattiamanzati](https://x.com/mattiamanzati)*