

TYPESCRIPT-FUNKTIONEN DEKORIEREN

NICHT NUR ZU WEIHNACHTEN!

ZIEL DIESER SESSION

- ◆ Bewusstsein für Decorators erhöhen
- ◆ Zeigen, wie man sie schreibt und verwendet
- ◆ Einige (hoffentlich) interessante Beispiele präsentieren
- ◆ Neugierde und Lust wecken, sie selbst auszuprobieren

WAS ERWARTET EUCH?

- ◆ Decorators im Allgemeinen
- ◆ Decorators in TypeScript
- ◆ Vorteile in Advent of Code
- ◆ Code, den ihr nie so verwenden solltet!
- ◆ Unsere eigenes kleines Dependency Injection Framework

WER BIN ICH?



Marco Sieben

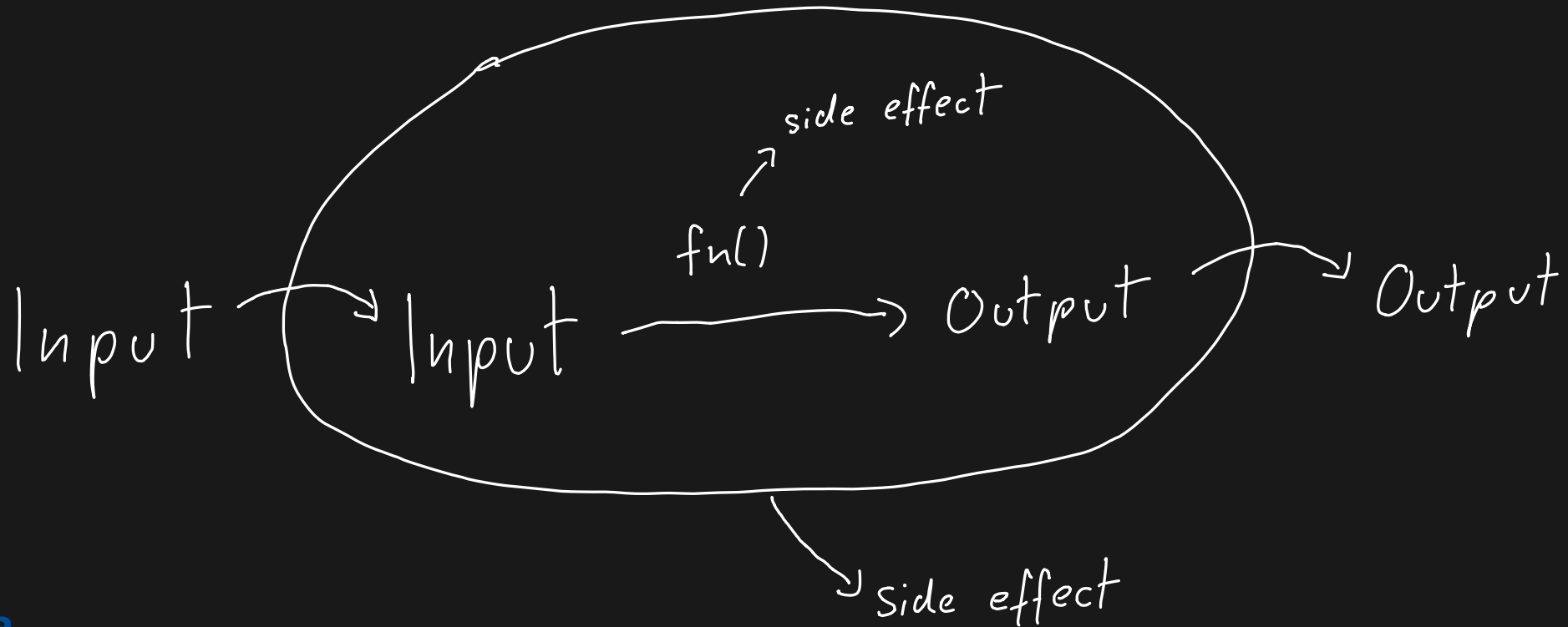
- ◆ Fullstack-Entwickler mit einem besonderen Herzen fürs Frontend
- ◆ TypeScript-Enthusiast
- ◆ Liebt den jährlichen Advent of Code (=> im Dezember immer besonders müde)
- ◆ Hat dort Decorators erfolgreich eingesetzt

Danke an meinen Arbeitgeber andrena objects ❤️

DECORATORS

DECORATORS

THEORIE

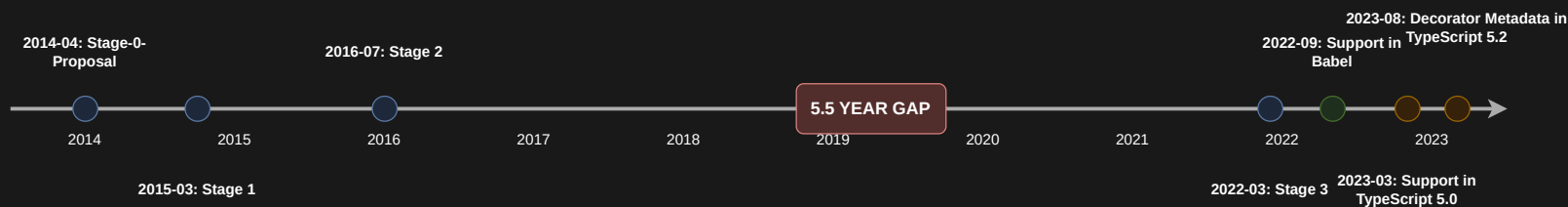


DECORATORS

EINFACHES BEISPIEL

DECORATORS IN TYPESCRIPT

- ◆ Stufe-3-Vorschlag für ECMAScript => Irgendwann nativ in JavaScript
- ◆ Unterstützung in TypeScript seit TypeScript 5.0
- ◆ Ergänzungen/Anpassungen in 5.2 und 5.5



DECORATORS IN TYPESCRIPT

- ◆ Class method decorators
- ◆ Class decorators
- ◆ Class getter/setter decorators
- ◆ Class field decorators
- ◆ Class auto-accessor decorators

CLASS METHOD DECORATORS

CLASS METHOD DECORATORS

BEISPIEL: LOGGED

CLASS METHOD DECORATORS

INTERFACE

```
type ClassMethodDecorator = (  
  value: Function,  
  context: {  
    kind: 'method';  
    name: string | symbol;  
    metadata: object;  
    static: boolean;  
    private: boolean;  
    access: { get: () => unknown };  
    addInitializer(initializer: () => void): void;  
  }  
) => Function | void;
```

ADVENT OF CODE

[Advent of Code](#) [\[About\]](#) [\[Events\]](#) [\[Shop\]](#) [\[Settings\]](#) [\[Log Out\]](#) [StNimmerlein \(AoC++\)](#)
[y\(2024\)](#) [\[Calendar\]](#) [\[AoC++\]](#) [\[Sponsors\]](#) [\[Leaderboard\]](#) [\[Stats\]](#)

Hi! I'm [Eric Wastl](#). I make Advent of Code. I hope you like it! I also made [Vanilla JS](#), [PHP Sadness](#), and [lots of other things](#). You can find me on [Twitter](#), [Mastodon](#), and [GitHub](#).

Advent of Code is an [Advent calendar](#) of small programming puzzles for a variety of skill sets and skill levels that can be solved in [any](#) programming language you like. People use them as [interview prep](#), [company training](#), [university coursework](#), [practice problems](#), a [speed contest](#), or to [challenge each other](#).

You don't need a computer science background to participate - just a little programming knowledge and some [problem solving skills](#) will get you pretty far. Nor do you need a fancy computer; every problem has a solution that completes in at most 15 seconds on ten-year-old hardware.

If you'd like to support Advent of Code, you can do so indirectly by helping to [\[Share\]](#) it with others or directly via [AoC++](#).

--- General Tips ---

If you get stuck, try your solution against the [examples](#) given in the puzzle; you should get the same answers. If not, re-read the description. Did you misunderstand something? Is your program doing something you don't expect? After the examples work, if your answer still isn't correct, [build some test cases](#) for which you can verify the answer by hand and see if those work with your program. Make sure you have the entire puzzle input. If you're still stuck, maybe [ask a friend](#) for help, or come back to the puzzle later. You can also ask for hints in the [subreddit](#).

--- Frequently Asked Questions ---

Is there an easy way to select entire code blocks? You should be able to triple-click code blocks to select them. You'll need JavaScript enabled.

```
#!/usr/bin/env perl
use warnings;

andrena
OBJECTS
```

- ◆ Tägliche Herausforderungen (1. Dezember bis 25. Dezember)
- ◆ In eine durchgehende Geschichte eingebunden
- ◆ Keine Programmiersprache vorgegeben
- ◆ Globale und private Leaderboards
- ◆ Komplette kostenlos

Unbedingt angucken unter
<https://adventofcode.com>

CLASS METHOD DECORATORS

BEISPIEL 2: MEMOIZATION

CLASS DECORATORS

CLASS DECORATORS

INTERFACE

```
type ClassDecorator = (  
  value: Function,  
  context: {  
    kind: 'class';  
    name: string | undefined;  
    metadata: object;  
    addInitializer(initializer: () ⇒ void): void;  
  }  
) ⇒ Function | void;
```

CLASS DECORATORS

BEISPIEL: SINGLETON

CLASS DECORATORS

BEISPIEL 2: DEPENDENCY INJECTION

WEITERE IDEEN

- ◆ Serialization
- ◆ Event Producer und Consumer
- ◆ Reactivity einführen
- ◆ ...

AUSBLICK

VERWANDTE THEMEN

- ◆ Andere Decorator-Typen (field, getter/setter, auto accessor)
- ◆ Decorator metadata
- ◆ auto accessors

LESEMATERIAL

- ◆ Release notes von TypeScript 5.0 (Decorators)
- ◆ Release notes von TypeScript 5.2 (Decorator metadata)
- ◆ Release notes von TypeScript 5.5 (Strikteres Parsing von Decorators)
- ◆ Der Decorator-Vorschlag für ECMAScript
- ◆ Der Decorator-Metadata-Vorschlag für ECMAScript
- ◆ Ausführlicher Blogpost von Axel Rauschmayer über den Decorators-Vorschlag

VIELEN DANK

UND VIEL SPASS BEIM DEKORIEREN



Repository mit Folien und
Beispielen:



Marco Sieben

 me@m7.codes
 m7.codes
 <https://www.linkedin.com/in/stnimmerlein>

andrena
OBJECTS