sebastian.greiner@aformatik.de
sebsgr
sebs_gr

Sebastian Greiner
aformatik Training & Consulting GmbH & CO. KG
Java Forum Stuttgart 2022

TypeScript

StarkeTypen für JavaScript

Typescript (TS)

- 2012 von Microsoft veröffentlicht
- OpenSource auf GitHub
- veröffentlicht unter der Apache-Lizenz

Typescript (TS)

Features

- Transpiliert TS Code zu JS
- statische Typisierung mit Checks zur Compilezeit
- Transpiliert in verschiede ECMA Script Versionen
- Objektorientierte Sprache
- Unterstützt alle JS Bibliotheken
- Superset von ECMA Script
  → valider JS Code ist valider TS Code

Und warum das Ganze?

```
1 Array(8).join('foo' - 1) + ' Batman!';
2
3 // NaNNaNNaNNaNNaNNaNNaN Batman!
```

Try on StackBlitz ⚡

Typescript installieren

via NPM installieren (-g => global)

```
1  npm install -g typescript
```

Try on StackBlitz ⚡

Typescript File transpilieren

```ts
// Filename: demo2.ts

const hello: string = "Hello World";
console.log(hello);
```

```
tsc demo2.ts
```

```js
// Filename: helloWorld.js

var hello = "Hello World";
console.log(hello);
```

Try on StackBlitz ⚡

Duck-Typing

When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.

```
1  class Duck {
2      sound = "quack";
3      swim() { console.log("Duck Swim!"); }
4  }
5  class Pinguin {
6      sound = "...";
7      swim() { console.log("Pinguin Swim!"); }
8  }
9  class Eagle {
10     sound = "...";
11 }
12
13 let duck2: Duck = new Pinguin();
14 let duck3: Duck = new Eagle();  // Error
15 console.log(duck2.swim());  // "Pinguin
   Swim!"
```

Try on StackBlitz ⚡

Typen

boolean	number	string	
enum	object	Function	
Array	Tuple		
null	undefined		

---

any	unknown	never	void

Typen

```typescript
let foo: unknown = "";
let bar: any = foo;

foo = 1;                    // OK
let s1: string = foo;       // Error
let s2: string = bar;       // OK

let v1: void = null;        // Abhängig von Compiler Option "strictNullChecks"
let v2: void = undefined;   // OK

function keepProcessing(): never {
    while (true) {
        console.log('Infinity Loop');
    }
}


keepProcessing();
keepProcessing(); // Unreachable code detected. (Compiler Option: allowUnreachableCode=false)
```

Try on StackBlitz ⚡

Type Inference

```
let x = 3;                                          // x: number

function bar(n: number, s: string) {                // bar(n: number, s: string): boolean
    return false;
}

class Foo {
    doSomething() { }                               // doSomething() : void
}
let foo = new Foo();                                // foo: Foo
foo.doSomething();
```

Try on StackBlitz ⚡

Object Types

```
1 var point: {x: number; y: number;};
2
3 function foo(point: {x: number; y: number;}) { }
4
5 function bar(): {x: number; y: number;}{ }
```

Try on StackBlitz ⚡

Type Aliases

```ts
type Test = string

type Point = { x: number, y: number };

type Cat = { meow: () => void };

type StringPropertyGetter = (obj: any) => string;

function printStringProperty(obj: any, getter: StringPropertyGetter) {
    console.log(getter(obj))
}

printStringProperty({ name: "Peter" }, obj => obj.peter)
```

Try on StackBlitz ⚡

Property Modifiers

Optional | Readonly

```
 1 type Point = {
 2     x: number;
 3     y: number;
 4     z?: number;
 5     readonly color: string
 6     readonly name?: string
 7 };
 8
 9 const point : Point = {x: 5, y: 10, color: "red", name: "P1" };
10 point.z;              // Returns: undefined
11 point.name = "P2";   // Error: Cannot assign to 'name' because it is a read-only property
```

Try on StackBlitz ⚡

Union Types

OR Verknüpfung von Typen

```
1 declare const foo: number | string;
2
3 foo.length;              // Error
4 foo.toExponential();     // Error
5
6 foo.toLocaleString();
```

Try on StackBlitz ⚡

Type Predicates

```ts
type Square = { a: number };
type Circle = { radius: number };
type Shape = Square | Circle;

function isSquare(shape: Shape): shape is Square {
    return ...;
}

declare const shape: Shape;

if (isSquare(shape)) {
    shape.a;
} else {
    shape.radius;
}
```

Try on StackBlitz ⚡

Intersection Types

AND Verknüpfung von Typen

```
1 type Colorful = { color: string };
2 type Circle = { radius: number };
3
4 type ColorfulCircle = Colorful & Circle;
5
6 const t: ColorfulCircle = {
7     color: "red",
8     radius: 4
9 }
```

Try on StackBlitz ⚡

String/Number Literal Types

```ts
type PrimeNumber = 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19;

type pets = "Dog" | "Cat" | "Horse";

type Yes = "yes" | "y" | 1 | true;
type No = "no" | "n" | 0 | false;

type YesOrNo = Yes | No;
```

Try on StackBlitz ⚡

Nullable Types

```
1 // required strictNullChecks=true
2
3 let s = "foo";
4 s = null;                          // error, 'null' is not assignable to 'string'
5
6 let sn: string | null = "bar";
7 sn = null;                         // ok
8 sn = undefined;                    // error, 'undefined' is not assignable to 'string | null'
```

Try on StackBlitz ⚡

Generics

```
function log<T>(arg: T): T {
    console.log(arg);
    return arg;
}

type Colorful = { color: string }

function logColor<T extends Colorful>(colorful: T) {
    console.log(colorful.color);
}

class ReadOnlyArray<T> {
    constructor(private innerArray: T[]) { }

    public getAtIndex(index : number): T{

        return this.innerArray[index];
    }
}
```

Try on StackBlitz ⚡

Mapped types

```
type Person = {
    readonly id: number,
    name: string,
    age: number,
    city?: string
};

type OptionalPerson = {
    [P in keyof Person]?: Person[P];
}

type StringPerson = {
    [P in keyof Person]: string;
}
```

Try on StackBlitz ⚡

Vorhandene Mapped Types

Try on StackBlitz ⚡

```
type Partial<T> = {
    [P in keyof T]?: T[P];
}
```

```
type Required<T> = {
    [P in keyof T]-?: T[P];
}
```

```
type Readonly<T> = {
    readonly  [P in keyof T]: T[P];
}
```

Conditional Types

```
1 type Foo<T> = T extends string ? number : boolean;
```

Try on StackBlitz ⚡

Conditional Types - Type Inferring

```ts
type Flatten<T> = Type extends Array<infer I> ? I : T;
declare var t1: Flatten<number[]>;  // number
declare var t2: Flatten<number>;    // number

type FullFlatten<T> = Type extends Array<infer I> ? FullFlatten<I> : T;
declare var t3: FullFlatten<number[][][][]>;  // number

type ReturnType<T extends (...args: any) => any> = T extends (...args: any) => infer R ? R : never;
declare var foo: ReturnType<() => string>;    // string
```

Try on StackBlitz ⚡

Und was ist jetzt mit Batman?

```
1 Array(16).join('foo' - 1) + ' Batman!';    // Error
```

Try on StackBlitz ⚡

Warum wird TS nicht zum neuen JS?

A Proposal For Type Syntax in JavaScript

sebastian.greiner@aformatik.de
sebsgr
sebs_gr

Sebastian Greiner
aformatik Training & Consulting GmbH & CO. KG
Java Forum Stuttgart 2022

TypeScript

StarkeTypen für JavaScript