
*http://www.react-cursus.nl*

MH

FRONTEERS

# Welkom bij Fronteers!

We zijn een vakvereniging voor front-end developers in Nederland en België. Vanaf 2007 zetten wij ons in voor de goede naam van front-end developers. We organiseren meetups om kennis te delen en mensen met elkaar in contact te brengen. Dát is onze missie: dat jij met meer plezier je werk doet.

WORD LID

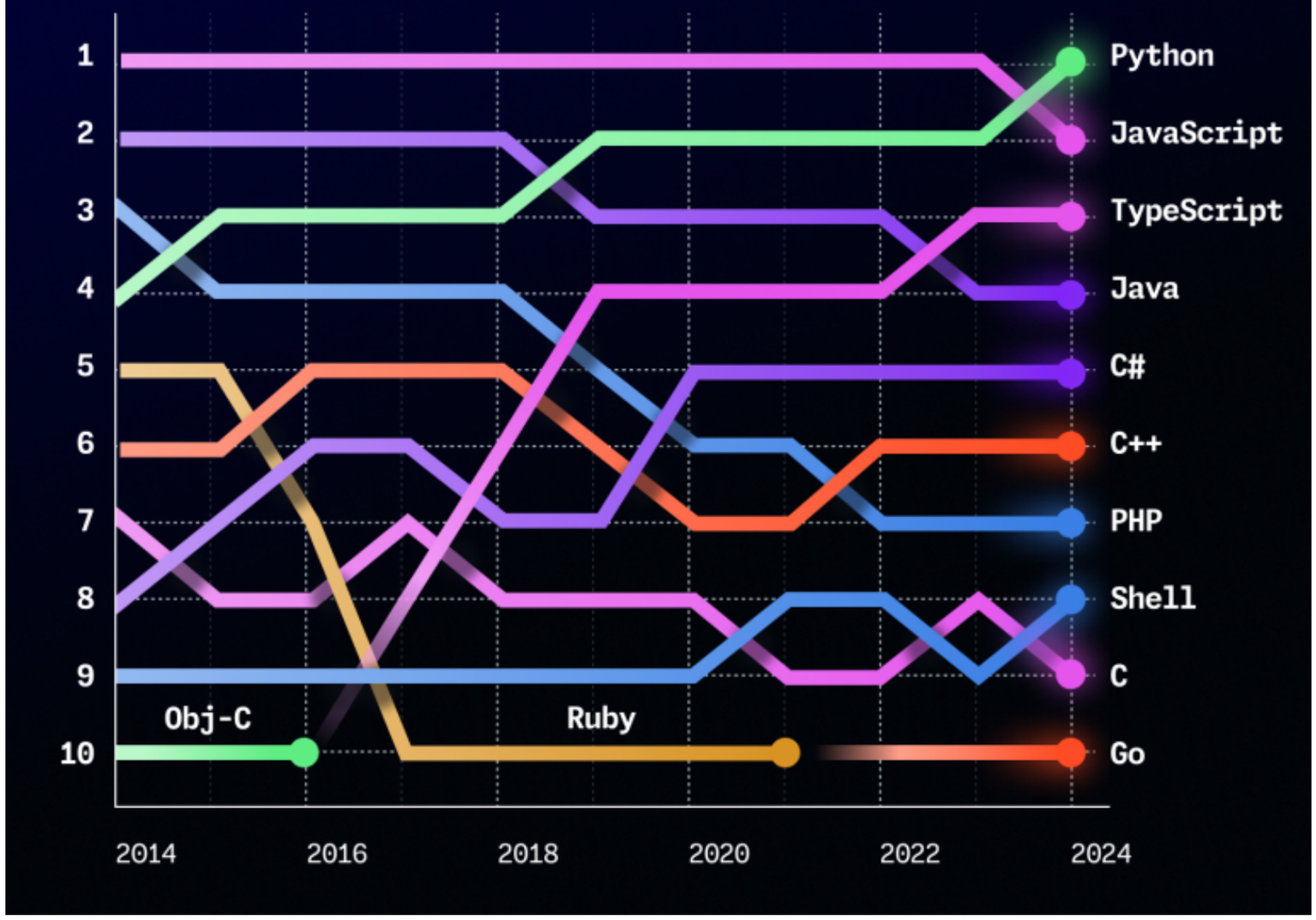
# TypeScript

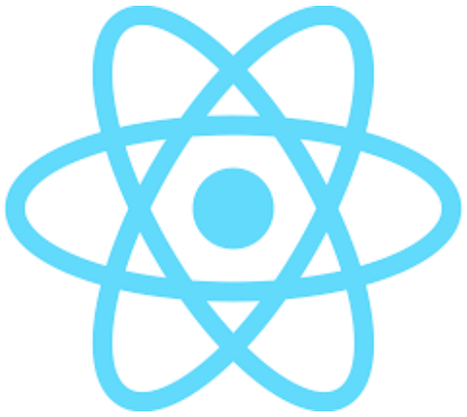
*https://www.typescriptlang.org/*

MH

# Top programming languages on GitHub

RANKED BY COUNT OF DISTINCT USERS CONTRIBUTING TO PROJECTS OF EACH LANGUAGE.






# Anders Hejlsberg

Microsoft

C#

BORLAND TURBO

TS

MH

# TypeScript


*https://www.typescriptlang.org/*

MH

# Wat doet TypeScript

```ts
let isValid: boolean = false;
let age: number = 12;
let firstName: string = "String";
let prime: null = null;
let galaxy: void = undefined;
```

MH

# TypeScript compiled naar js

```
let isValid = false;
let age = 12;
let firstName = "String";
let prime = null;
let galaxy = undefined;
```

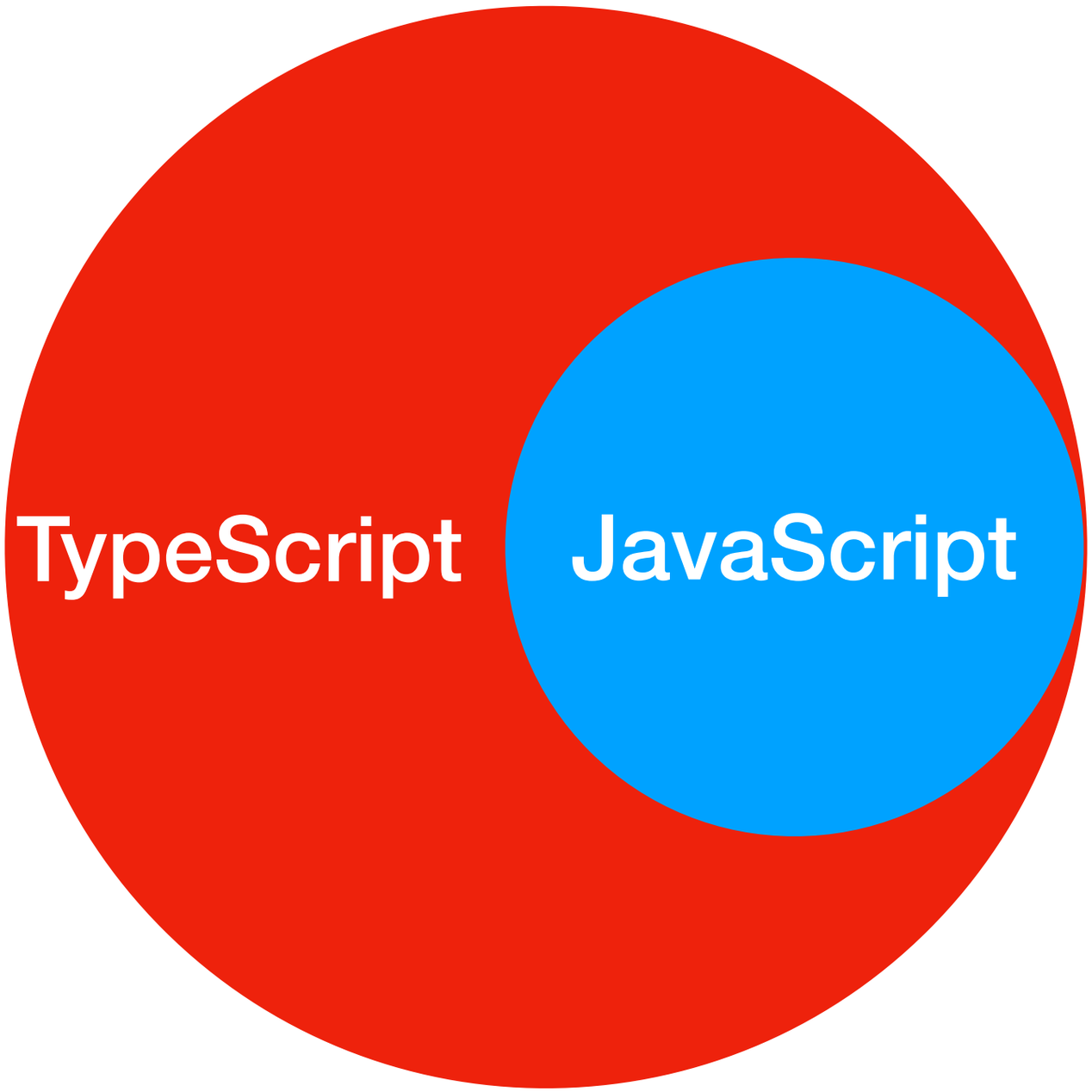
MH

# Oefening

001-typescript-primitive-variables

MH

# Superset



# Voordelen

- Een "typechecker" die zorgt dat onze code klopt. Dit maakt code handelbaarder.

- Autocompletion: jouw text editor snapt jouw code veel beter, de autocomplete is veel fijner.

MH

# Nadelen

- Extra dingen die je moet leren, en JavaScript en TypeScript.

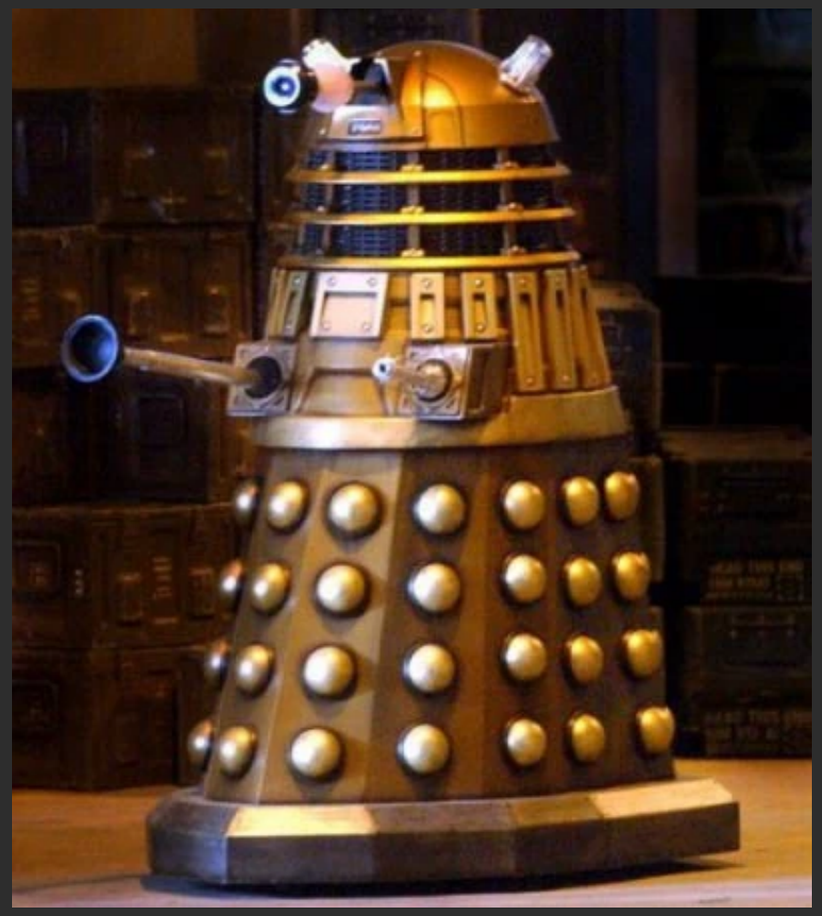
- TypeScript kan niet door de browser worden uitgevoerd en zal dus moeten worden gecompileerd.

MH

# TypeScript imago

## Love


## Hate



MH

# TypeScript is complex

as const

unions

structural typing

variadic

bivariance

guards

intrinsic

infer

covariance

intersections

literal

contravariance

variadic

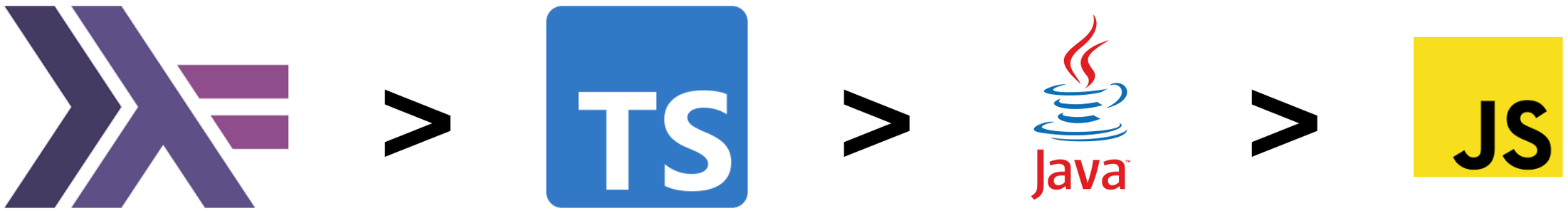
narrowing

readonly

inference

MH

# Maartens powerscale™




"In TypeScript kan je programmeren met types: dus types van andere types maken, dit is enorm krachtig"

MH

# TypeScript is complex

# Arrays

```ts
const prices: number[] = [1, 2, 3, 4, 5];

const ages: Array<number> = [42, 1337];
```

MH

# Tuples

```ts
const maarten: [string, number] = ["maarten", 28];
```

MH

# named tuples

```ts
const coord: [x: number, y: number] = [42, 42];
```

MH

# Oefening

## 002-typescript-array

MH

# functions

```
function sum(a: number, b: number): number {
  return a + b;
}
```

MH

# Oefening

003-typescript-functions

MH

# objects

```ts
const person: {
  name: string,
  age: number
} = {
  name: 'Maarten',
  age: 28
};
```

MH

# Type

```
type Person = {
    name: string;
    age: number;
}
```

# Oefening

004-typescript-object-part-one

MH

# Oefening

005-typescript-object-part-two

MH

# Type Aliassen

```ts
type PhoneNumber = string;

function formatPhoneNumber(phoneNumber: PhoneNumber): PhoneNumber {
  return `+31 ${phoneNumber}`;
}
```

MH

# Nesten

```ts
type PhoneNumber = string;

type Person = {
  name: string;
  phoneNumber: PhoneNumber;
  age: number;
}
```

MH

# Oefening

006-typescript-nesting

MH

# objects with maybe

```ts
type Person = {
  name: string;
  age?: number;
}
```

# Oefening

007-typescript-maybe-part-one

MH

# Oefening

008-typescript-maybe-part-two

MH

# Method definen

```ts
type Props = {
  value: number;
  onChange: (value: number) => void;
}
```

MH

# Method definen BAD

```ts
type Props = {
  value: number;
  onChange(value: number): void;
}
```

MH

# Optional Method

```tsx
type Props = {
  value: number;
  onChange?: (value: number) => void;
}
```

MH

# Oefening

## 009-methods

MH

# Type inference

```
let isValid = false;
let age = 12;
let firstName = "String";
let prime = null;
let galaxy = undefined;
```

MH

# Type inference

```ts
export const names = ['Maarten', 'Tosca', 'Owen', 'Jane'];

export const person = {
  name: 'Maarten',
  age: 33,
};

export const persons = [
  {
    name: 'Jan',
    age: 33,
  },
  {
    name: 'Dirk',
    age: 66,
  },
  {
    name: 'Phoebe',
    age: 40,
  },
];
```

MH

# Oefening

010-type-inference

MH

# types definen #5

```
type Awesome = 42;
```

MH

# types definen #6

```ts
type Color = "red" | "white" | "blue";
type Size = "small" | "medium" | "large";
```

MH

# Oefening

011-union-part-one

MH

# types definen #7

```ts
type NumberOrString = number | string;
```

MH

# Oefening

012-union-part-two

MH

# generics

```ts
type List<T> = {
  content: T[]
}

const listOfNumbers: List<number> = {
  content: [1, 2, 3]
};

const listOfStrings: List<string> = {
  content: ["A", "B"]
};
```

MH

# Oefening

[013-generics]

# Casting

```ts
const button = document.getElementById('button') as HTMLButtonElement

const x = "string" as any as number;
```

# Oefening

## 014-casting

# Einde

*http://www.react-cursus.nl*

# Bonus ronde

MH

# Fun with types #0

```ts
type Entity = {
  id: number;
}

function logId<T extends Entity>(entity: T) {
  console.log(entity.id);
}
```

MH

# Fun with types #1

```ts
type Size = 'sm' | 'md' | 'lg';

type Space = '1' | '2' | '3' | '4' | '5';

type Position = 'mt' | 'mb' | 'ml' | 'mr' | 'mx' | 'my';

type Margin = `${Position}-${Space}-${Size}`;

const a: Margin = 'mt-5-sm';
```

MH

# Fun with types #2

```ts
type Person {
  name: string
};
```

```ts
type PersonWithId = Person & { id: number }

const maarten: PersonWithId = {
  id: 42,
  name: 'maarten'
}
```

MH

# Fun with types #3

```ts
type IsAllowedCallback = (name: string, age: number) => boolean

const f: IsAllowedCallback = (name, age) => {
  return name === "Maarten" && age === 29;
}
```

MH

# Fun with types #4

```ts
type Person = {
  name: string;
  phoneNumber: string;
  age: number;
}

type PersonKey = keyof Person;

// Zelfde als
// type PersonKey = 'name' | 'phoneNumber' | 'age';

const key: PersonKey = "name";
```

MH

# Fun with types #5

```ts
type Person = {
  id: number;
  name: string;
}

type PersonWithoutId = Omit<Person, 'id'>;

const maarten: PersonWithoutId = {
  name: "maarten"
}
```

https://www.typescriptlang.org/docs/handbook/utility-types.html

MH

# Fun with types #6

```ts
export const directions = [
  'north',
  'east',
  'south',
  'west'
] as const;

type Direction = typeof directions[number];

const a: Direction = 'north';
```

MH

# Fun with types #7

```ts
type Person = {
  readonly name: string;
  age: number;
}

const jane: Person = {
  name: "Jane",
  age: 1
}

jane.name = 'Owen';
```

MH

# Fun with types #9

```ts
type DaysOfTheWeek =
  "sunday" |
  "monday" |
  "tuesday" |
  "wednesday" |
  "thursday" |
  "friday" |
  "saturday";

// "SUNDAY" | "MONDAY" | "TUESDAY" etc etc
type CapitalizedDaysOfTheWeek = Uppercase<DaysOfTheWeek>
```

MH

# Fun with types #9

```ts
type A = number;

// Wanneer A een nummer is dan is het
// een boolean anders een string
type B = A extends number
  ? boolean
  : string;
```

MH

# Fun with types #10

```ts
type Person = {
  firstName: string;
  lastName: string;
  age: number;
};

// Dit type is net een functie hij accepteert een argument
// T wat een type moet zijn.
type StringProperties<T> = {
  // Loopt over alle properties van T heen via "in keyof"
  [K in keyof T]: // Als een key van T een string is
  T[K] extends string
    ? // Dan returnen we K
      K
    : // Is het geen string dan returnen we niets
      never;
}[keyof T];

// "firstName" | "lastName"
type PersonStringProps = StringProperties<Person>;

// { firstName: string, lastName: string}
type PersonOnlyStringProps = Pick<Person, PersonStringProps>;
```

MH

# Fun with types #11

```ts
type Person = {
  firstName: string;
  lastName: string;
};
```

```ts
type Getterize<T> = {
  [K in keyof T as `get${Capitalize<string & K>}`]: () => T[K]
}

type PersonWithGetters = Getterize<Person>;

const person: PersonWithGetters = {
  getFirstName() {
    return "Maarten";
  },
 getLastName() {
    return "Hus";
  },
};
```

MH

# classes #1

```
class Person {
  name: string;
  age: number;

  constructor(name: string, age: number) {
    this.name = name;
    this.age = age;
  }
}

const maarten = new Person("Maarten", 28);
```

MH

# classes #2

```
class Person {
  name?: string;
  age?: number;
}

const maarten = new Person();
maarten.name = 'Maarten';
maarten.age = 28;
```

MH

# interfaces

```
interface Serializable {
  serialize(): string
}

class User implements Serializable {
  serialize() {
    return 'User';
  }
}
```

MH

# Interface lijkt op type

```ts
interface Person {
  name: string;
};

interface Entity {
  id: number;
}

interface PersonWithId extends Person, Entity {}

const maarten: PersonWithId = {
  id: 42,
  name: 'maarten'
}
```

MH

# Gebruik geen interface

- Type heeft minder karakters.

- Type kan alles wat interface kan, andersom niet. Een type kan namelijk een alias maken, interfaces niet.

- Type kan niet worden aangepast na aanmaken interface wel, je kan hem uitbreiden na de definitie.

MH

# Interface uitbreiden

```ts
interface Person {
  name: string;
};
```

```ts
interface Person {
  id: number;
}

const person: Person = {
  id: 1,
  name: "Jane"
}
```

MH

# @ts-expect-error

“Een escape hatch om TypeScript het zwijgen op te leggen”

MH

# Special types #1

```ts
// Betekent dat de waarde "alles" kan zijn.
const a: any;

// Broertje van "any" verschil is dat hij
// je niets laat doen met de waarde tot
// je bewijst wat het is.
const b: unknown;
```

MH

# Special types #2

```
// De return value van een functie die niets
// teruggeeft, dus als een functie geen
// return heeft.
const c: void;

// Wanneer een functie altijd een error gooit
// dan is het return type "never" hij geeft
// dus nooit iets terug.
const d: never;
```

MH

# @ts-expect-error

```ts
let a: number = 42;

// @ts-expect-error "This part is the reason"
a = 'string';
```

MH

# @ts-expect-error

```ts
// @ts-expect-error allow for reading username
const USERNAME: string = env.STANDARD_USERNAME;


// @ts-expect-error test mock
authentication.useIsLoggedIn = () => false;
```

MH

# @ts-expect-error

- Liegen en bedriegen met types moet je echt alleen gebruiken als laatste redmiddel.

- Meestal in "normale" code niet veel nodig, zie het echt als een code smell bij een review.

MH

# @ts-ignore

- Niet gebruiken is oude variant van @ts-expect-errror
- Verschil is dat wanneer een @ts-expect-error onnodig is, na bijv een TypeScript update die het wel snapt, je een warning krijgt dat de @ts-expect-error niet meer nodig is.

MH