Mobile

TypeScript

HO
GENT

TypeScript

HO
GENT

TypeScript

- **Ontworpen** door **Microsoft (2012)** en het is **bedoeld** voor het **ontwikkelen** van **complexe JavaScript toepassingen**.
- **Voorbeeld** van grote projecten die TypeScript gebruiken nl. **Visual Studio Code** (300.000 regels code).
- **Statische code-analyse**, **compile-time type checking**, **code-level documentatie**, ...
- Helpt bij het **vangen** van **fouten tijdens** de **ontwikkelingsfase**
- **Vereenvoudigt** het **beheer** van **grote codebases**
- **Verbetert** de **leesbaarheid** en **onderhoudbaarheid** van de **code**

Voorbeeld

```
// JavaScript

const message = "Hello, World!";
console.log(message);

// TypeScript

const message: string = "Hello, World!";
console.log(message);
```

TypeScript

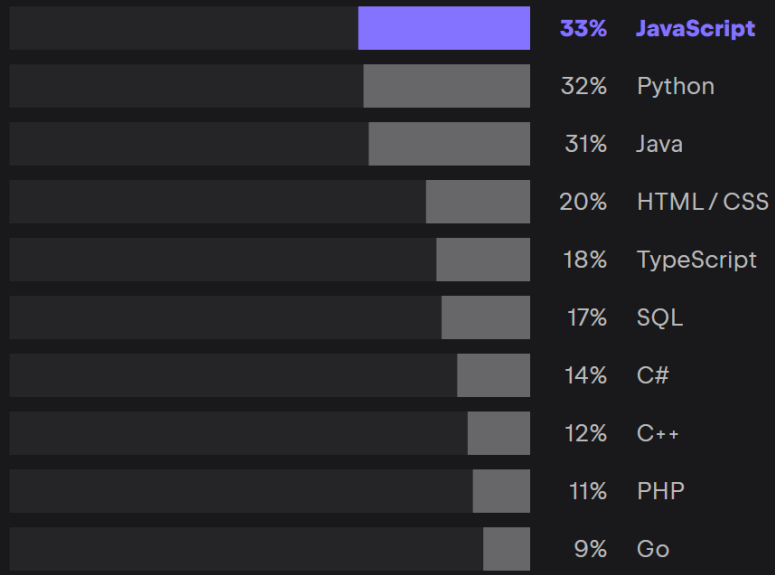
Which programming languages have you used in the last 12 months?

	2017	2018	2019	2020	2021	2022
JavaScript	65%	64%	69%	70%	69%	65%
HTML / CSS	60%	55%	61%	61%	60%	54%
Java	47%	51%	50%	54%	49%	48%
SQL	42%	47%	56%	56%	54%	49%
Python	32%	41%	49%	55%	52%	53%
PHP	30%	26%	29%	27%	32%	20%
C#	20%	22%	24%	22%	21%	23%
C++	17%	18%	20%	27%	23%	25%
C	15%	16%	17%	23%	19%	20%
TypeScript	12%	17%	25%	28%	29%	34%
Ruby	10%	8%	11%	8%	6%	5%
Swift	9%	8%	11%	9%	7%	7%
Go	8%	12%	18%	19%	17%	19%
Objective-C	7%	5%	6%	4%	3%	3%
Scala	7%	5%	6%	5%	3%	3%
Kotlin	2%	9%	16%	17%	14%	16%
Rust	–	2%	5%	7%	6%	9%
Shell	–	29%	40%	39%	37%	34%

0 70%

What are your primary programming languages?

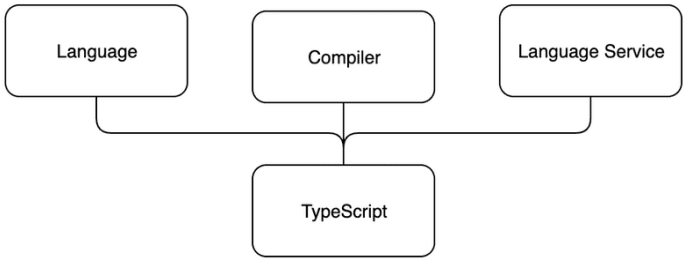
Choose no more than three languages.



Visual Studio Magazine - 2022: https://visualstudiomagazine.com/articles/2023/02/02/jetbrains-survey.aspx

Onderdelen

- **Superset** van **JavaScript**, wat betekent dat het **alle functies** van **JavaScript** omvat, evenals **aanvullende functies**. Met andere woorden, alle bestaande JavaScript-code is geldige TypeScript
- Bestaat uit **3 afzonderlijke delen**: de **taal** zelf, de **compiler**, en de **taaldienst** (language service).



HO GENT

Taal

- Bestaat uit de **syntax**, de **keywords** en **type-annotaties**
- **Syntax** lijkt op, maar is **niet hetzelfde** als **JavaScript-syntax**
- **Ontwikkelaars** hebben het meeste **contact** met de **taal** van TypeScript

HO
GENT

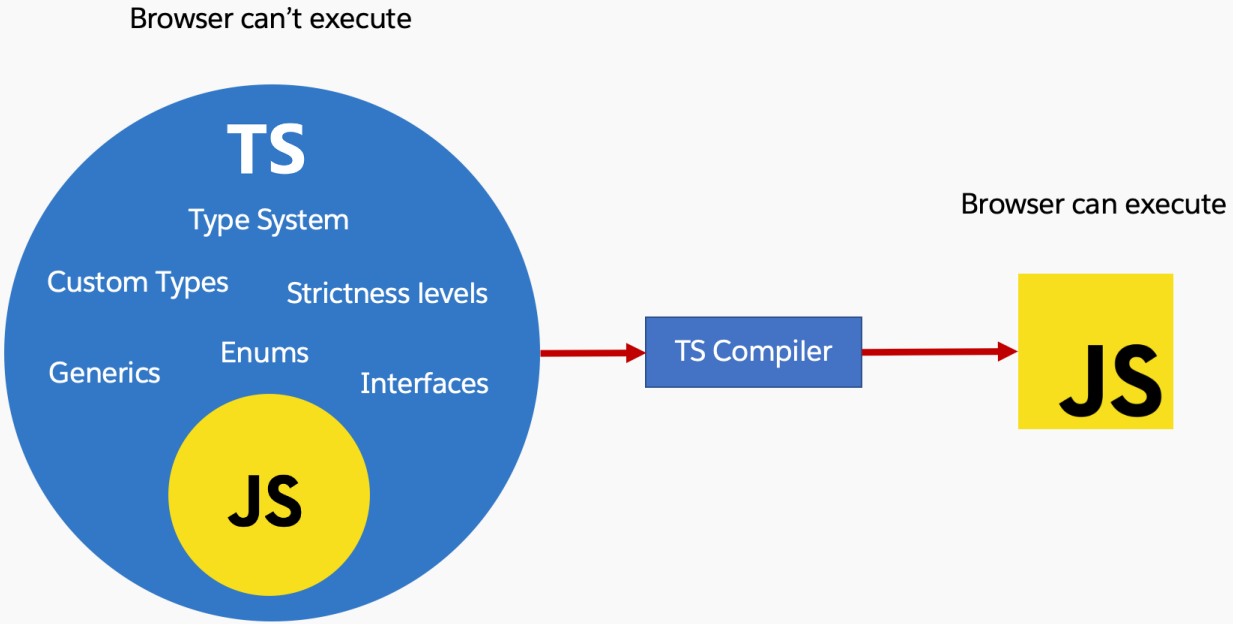
Compiler

- Verantwoordelijk voor het **verwijderen** van **type-informatie** (type-annotaties) en het **uitvoeren** van **code-transformaties**
- **Code-transformaties** stellen **TypeScript-code** in staat om te worden **omgezet** in **uitvoerbare JavaScript code**
- **Traditioneel** wordt **compileren** gebruikt om **code om** te **zetten** van een **leesbaar format** voor **mensen** naar een **leesbaar format** voor **machines**, maar in **TypeScript** wordt **leesbare broncode** omgezet in een **andere leesbare broncode**. Dit wordt vaak **transpilen** genoemd.
- **Statische code-analyse** die **waarschuwingen** of **fouten genereren**

Taaldienst (Language Service)

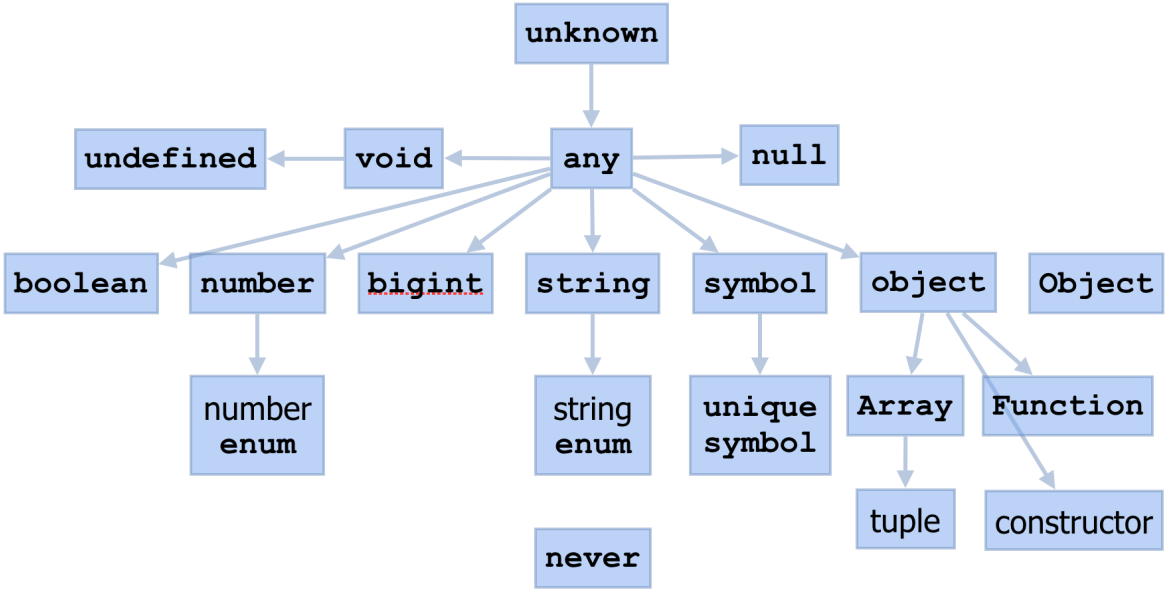
- **Verzamelt type-informatie** uit **broncode**
- **Ontwikkeltools** kunnen deze **type-informatie** gebruiken voor **functies** zoals **intellisense**, **typehints** en **mogelijke refactoring-opties**.

HO GENT


Browser can't execute
TS
Type System
Custom Types
Strictness levels
Generics
Enums
Interfaces
JS
TS Compiler
Browser can execute
JS

Types

- **Verschillende basistypes** die we kunnen gebruiken



HO GENT

Types (2)

```ts
// TypeScript types

// number
const age: number = 25;

// string
const name: string = "John Doe";

// boolean
const isActive: boolean = true;

// Array
const list: number[] = [1, 2, 3];
const secondList: Array<number> = [1, 2, 3];
```

HO
GENT

Types (3)

```typescript
// TypeScript types

// Tuple
const x: [string, number] = ["hello", 10];

// Enum
enum Color { Red, Green, Blue };
const c: Color = Color.Green;

// any
const notSure: any = 4;

// void
const warnUser = (): void => {
    console.log("Dit is een waarschuwing!");
}
```

HO GENT

Types & Type aliases

- **type** is een **keyword** in **TypeScript** waarmee we de **vorm** van **data** kunnen **definiëren**.
- **Type aliases** zijn in principe een “**naam voor elk mogelijk type**”.

```ts
// Types & type aliases

type MyNumber = number;

type User = {
  id: number;
  name: string;
  email: string;
}

// Union types

type ErrorCode = string | number;
type Answer = string | number;
```

HO GENT

Interfaces

- **Een interface** in TypeScript **definieërt** een **contract** waar een **object moet** aan **voldoen**
- **Lijkt gelijk** aan het **type keyword MAAR NIET ALTIJD ZO**

```
// Interfaces

interface Client {
  name: string;
  address: string;
}

// OF

type Client = {
  name: string;
  address: string;
}
```

HO GENT

Type vs interface

- In **sommige scenario's niet mogelijk** om **interface** te **gebruiken**.
- Bvb. primitieve type aliases, union types

```
// Primitieve types - type aliases

type Address = string;
    // Geen alternatief met het interface keyword

// Union types

type Transport = 'Bus' | 'Car' | 'Bike' | 'Walk';
    // Geen alternatief met het interface keyword
```

```
// Union types met interfaces

interface CarBattery {
  power: number;
}

interface Engine {
  type: string;
}

type HybridCar = Engine | CarBattery;
```

Type vs interface (2)

- In **sommige scenario's niet aan te raden** om **interface** te **gebruiken**.
- Bvb. function types

```typescript
// Function types

type AddFn = (num1: number, num2: number) => number;

// Kan wel gebruik maken van het interface keyword maar onduidelijker

interface IAdd {
  (num1: number, num2: number): number;
}
```

HO GENT

Type vs interface (3)

- In **sommige scenario's niet aan te raden** om **interface** te **gebruiken**.
- Bvb. function types

```typescript
// Function types

type Car = 'ICE' | 'EV';
type ChargeEV = (kws: number) => void;
type FillPetrol = (type: string, liters: number) => void;
type RefillHandler<A extends Car> = A extends 'ICE' ? FillPetrol : A extends 'EV' ? ChargeEV : never;

const chargeTesla: RefillHandler<'EV'> = (power) => {
  // Implementatie
}

const refillToyota: RefilHandler<'ICE'> = (fuelType, amount) => {
  // Implementatie
}
```

HO GENT