thecodecampus</>

WMK

TypeScript

What's new in 2.0 and 2.1

Über mich

Philipp Burgmer
<> burgmer@w11k.de | @philippburgmer
Software-Developer, Trainer
Web-Technologies, Security
TypeScript, Angular </>

Über uns

theCodeCampus.de - Weiter.Entwickeln.

<> Trainings since 2007
theCodeCampus since 2013
Project-Kickoffs & Support </>

W11K GmbH - The Web Engineers

<> Founded 2000
Development / Consulting
Web / Java
Esslingen / Siegburg </>

Overview

- Return Type `never`
- `readonly` Properties
- Null- and undefined-aware types
- async/await

Return Type never

<> Different to `void`

<> Represents the type of values that never occur

<> Return type for functions that never return

```
function doSomethingHeavy(): never {
  while(true) {
    // ...
  }
}
```

readonly Properties

<> **New keyword** `readonly`
- Compiler catches write access
- Even within nested structures

<> **Pure TypeScript Feature**
- Does not generate properties with `Object.defineProperty` and `writable: false`
- Lightweight alternative to icepick (with is lighter than immutable.js)
- Just check immutability at compile time

```
interface DataFromServer {
  readonly prop1: number;
  readonly prop2: string;
}
```

Null- and undefined-aware types

<> `null`: The Billion Dollar Mistake

<> Compiler Flag `--strictNullChecks`

- Types `undefined` and `null`
- Other types does not include `null` or `undefined` values anymore

```
1 let x: number = undefined // error
```

- Union Types

```
1 let x: number | undefined = undefined // ok
```

Null- and undefined-aware types

<> Allows assigned-before-use checking

```
let x: number;
x;  // Error, reference not preceded by assignment
x = 1;
x; // Ok

let z: number | undefined;
z;  // Ok
```

Null- and undefined-aware types

<> What does `document.querySelector("foo")` return?

<> Do you check for `null`?

<> Or do you check for `undefined`?

<> Or do you check both?

Null- and undefined-aware types

```
1  var element = document.querySelector("foo");
2
3  element.classList.add("bar"); // error: element can be null
4
5  if (element !== null) {
6    element.classList.add("bar"); // ok
7  }
```

async/await

<> Should make async programming easy
- Less code nesting, flat code again
- Readable code!

<> Didn't make it to ES2016, hopefully 2017

<> Already supported by TypeScript, but only for `target: "es6"`

<> Now supported for ES3 and ES5 also

```
async function doSomething() {
  await doSomethingAsync();
  console.log("finished");
}
```

async/await

```
1  function check() {
2    return new Promise<boolean>(resolve => resolve(false));
3  }
4
5  function doNotGiveIn() {
6    return new Promise<void>(resolve => { console.log("Do it now!"); resolve(); });
7  }
8
9  async function askForCleanup() {
10   console.log("Please clean up your room");
11   while(await !check()) { await doNotGiveIn(); }
12   console.log("Thanks!");
13 }
14
15 askForCleanup();
```

Philipp Burgmer
burgmer@w11k.de
Twitter: @philippburgmer
GitHub: pburgmer

www.thecodecampus.de
@theCodeCampus