



TypeScript

Neues aus 2.0 und 2.1

TypeScript



Philipp Burgmer

<> burgmer@w11k.de | [@philippburgmer](https://twitter.com/philippburgmer)

Software-Entwickler, Trainer

Web-Technologien, Sicherheit

TypeScript, Angular </>

theCodeCampus.de - Weiter.Entwickeln.

- <> Schulungen seit 2007
- theCodeCampus seit 2013
- Projekt-Kickoffs & Unterstützung im Projekt </>

W11K GmbH - The Web Engineers

- <> Gegründet 2000
- Auftrags-Entwicklung / Consulting
- Web / Java
- Esslingen / Siegburg </>

<> 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

```
1 function doSomethingHeavy(): never {  
2   while(true) {  
3     // ...  
4   }  
5 }
```

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](#) (which is lighter than [immutable.js](#))
- Just check immutability at compile time

```
1 interface DataFromServer {  
2   readonly prop1: number;  
3   readonly prop2: string;  
4 }
```

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

```
1 let x: number;
2 x; // Error, reference not preceded by assignment
3 x = 1;
4 x; // Ok
5
6 let z: number | undefined;
7 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 }
```

<> 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

```
1  async function doSomething() {  
2    await doSomethingAsync();  
3    console.log("finished");  
4  }
```

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