NILS HARTMANN | HTTPS://NILSHARTMANN.NET

Let's type!

A practical introduction to TypeScript

Slides: http://bit.ly/oose-react

NILS HARTMANN

Software Developer from Hamburg

JavaScript, Java Trainings, Workshops "JavaScript that scales"

TypeScript

BACKGROUND: TYPESCRIPT

TypeScript: Superset of JavaScript with Type System

- Every JavaScript Code is valid TypeScript code (should be...)
- Compiler compiles TypeScript into JavaScript-Code
 - Compiler Supports JSX (for React Apps)
- Very good IDE Support
 - esp. VS Code and IDEA

```
workspace - TypeScript - hello-world/src/HelloWorld.tsx - Eclipse
                                                                                           SE Outline $3
import * as React from 'react';
                                                                                              # "HelloWorld"

▼ 

→ HelloWorld

  3 type HelloWorldProps = {
                                                                                                   render
                                                                                                 ① HelloWorldProps
             name: string,
                                                                                                 @ React
             greeting: string
 6 }
  80 export default class HelloWorld
         extends React.Component<HelloWorldProps, {}> {
         render() {
             return (
                  <div>
 129
$13
                       {this.props.name}
                       {this.props.}
                                                                         (property) children: React ReactNode
15
                  </div>
                                     greeting
             );
                                                                                                                   V - -
Problems 🔗 Search 23
2 matches - done in 572 ms.
▼ Phello-world
  W Src
      HelloWorld.tsx (2 matches)
          3: name: string.
         12: (this props name)
                                               Press '^Space' to show Template Proposals
                             Writable
                                                                                                            20 m = 7 6
```

Using Types

Variables

let foo: string; // built-in types, for example: string, number, boolean

Using Types

```
Variables
let foo: string; // built-in types, for example: string, number, boolean

Functions
function sayIt(what: string) {
   return `Saying: ${what}`;
}
```

Using Types

```
Variables
let foo: string; // built-in types, for example: string, number, boolean

Functions
function sayIt(what: string) {
    return `Saying: ${what}`;
}

Specifying Types is optional, Types will then be inferred:
let result = 7; inferred Type: number
result = sayIt('Lars') // Error (inferred type of sayIt: string)
```

Using Types

```
Variables
let foo: string; // built-in types, for example: string, number, boolean

Functions
function sayIt(what: string) {
    return `Saying: ${what}`;
}

Specifying Types is optional, Types will be inferred:
let result = 7; inferred Type: number
result = sayIt('Lars') // Error (inferred type of sayIt: string)
```

Defining own Types

```
interface Person {
    firstName: string,
    lastName: string|null,
    age?: number
}
// alternative: type
// nullable Type ("a String or null")
// optional type (might be undefined)
}
```

Defining own Types - Usage

```
firstName: string,
 lastName: string|null, // nullable Type ("a String <u>or</u> null")
 age?: number
                   // optional type (might be undefined)
function sayHello(p: Person) {
 console.log(`Hello, ${p.lastName}`);
 p.lastName.toUpperCase(); // Error: Object is possibly null
sayHello({firstName: 'Klaus', lastName: null}); // OK
sayHello({firstName: 'Klaus', lastName: 777}); // Error: lastName not a string
sayHello({firstName: 'Klaus', lastName: 'Mueller', age: 32}); // OK
```

Generics

```
interface Person { name: string };
interface Movie { title: string };

let persons:Array<Person> = [];
let movies:Array<Movie> = [];

persons.push({name: 'Klaus'}); // OK
movies.push({title: 'Batman'}); // OK
persons.push({title: 'Casablanca'}) // error ('title' not in Person)
```

Type Alias

Union Types and Type Guards

Advanced Types

Thank you!

Slides: http://bit.ly/oose-react

Questions?