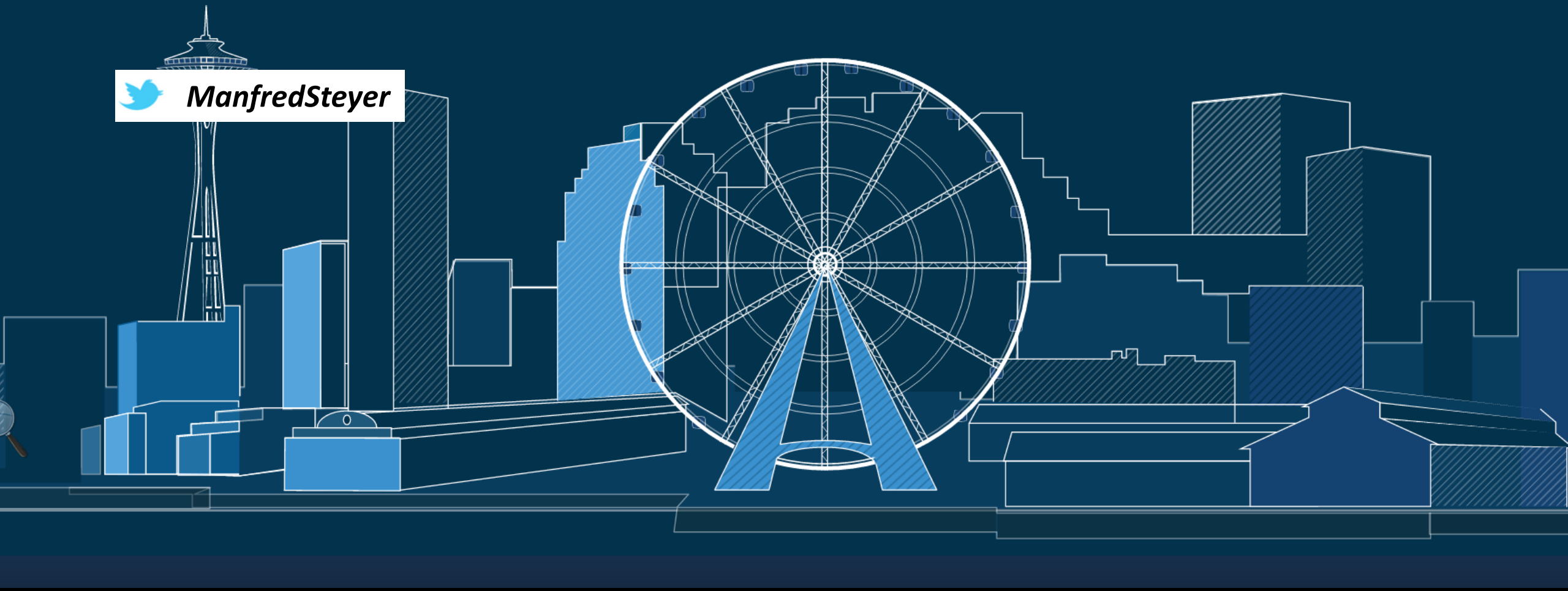


Einführung in TypeScript

Manfred Steyer

SOFTWAREarchitekt.at

Inhalt

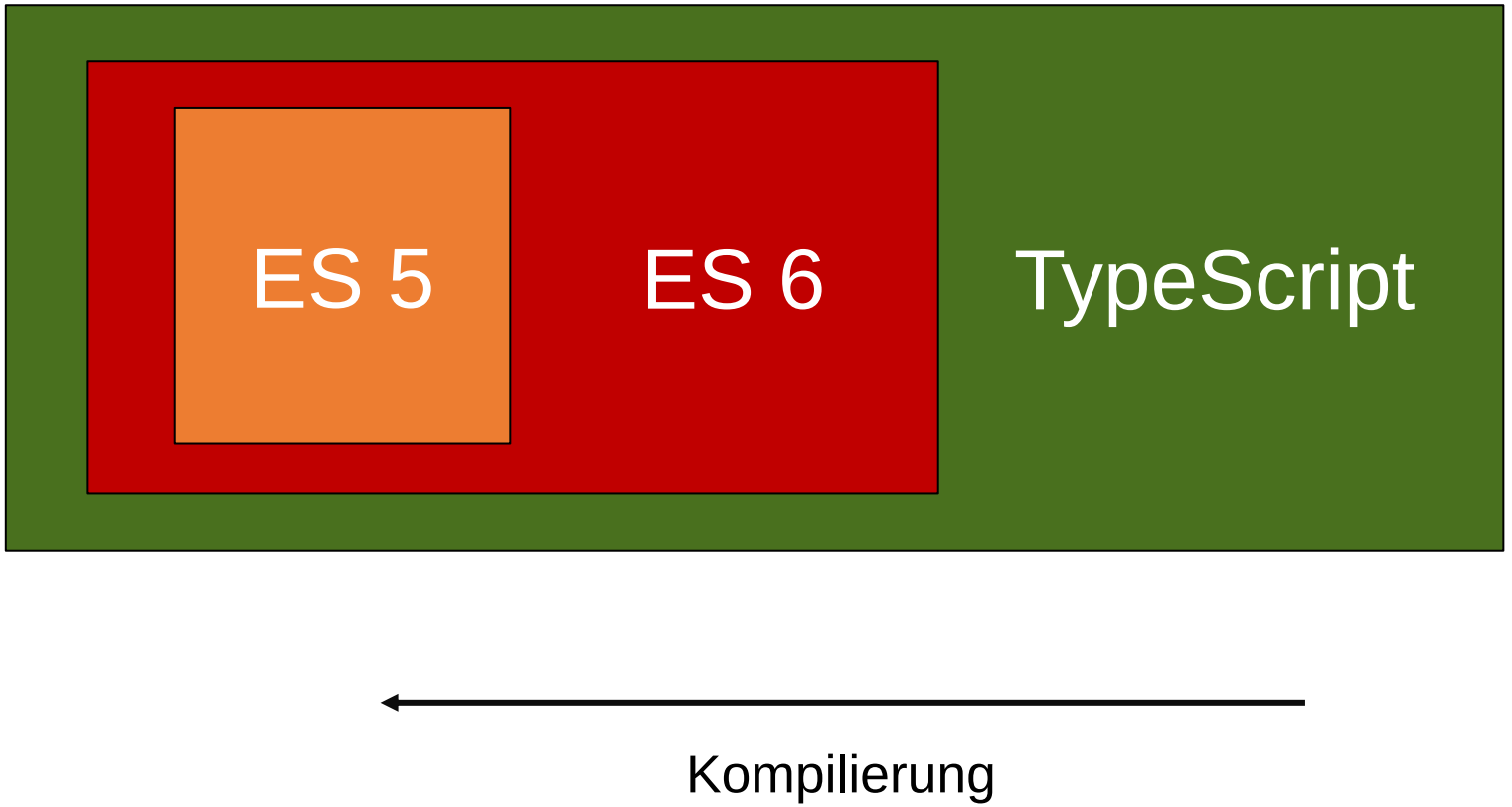
- Überblick zu TypeScript
- Ein erstes Beispiel
- Projektsetup (JavaScript-Ecosystem)
- Weitere Details

Überblick

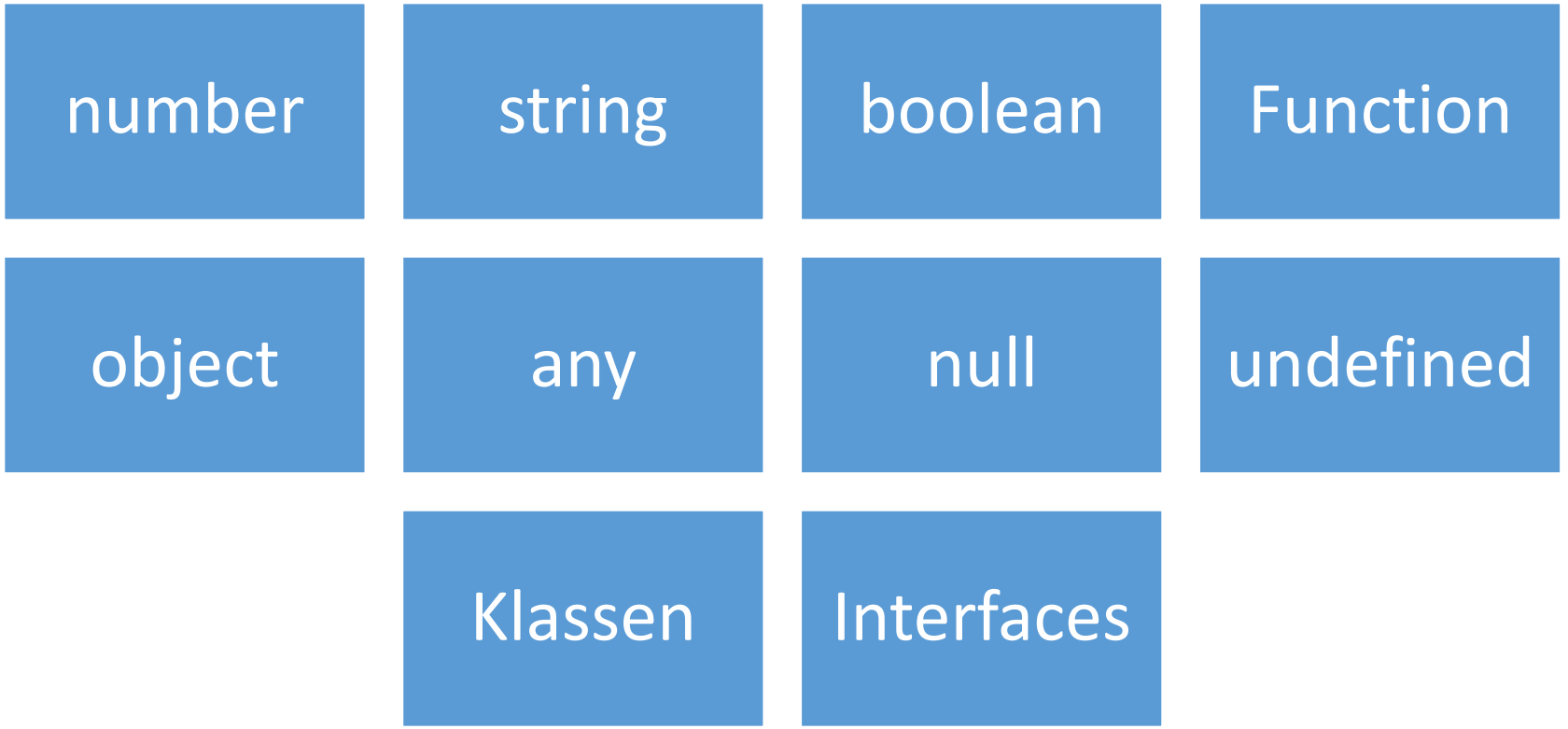
Was ist TypeScript?

- Superset von EcmaScript 6+ (2015+)
- Kompilierung nach EcmaScript 6 (2015), 5 oder 3
- Bringt statisches Typsystem
- Erlaubt Codevervollständigung und Refactoring

TypeScript und ES6



Datentypen


number
string
boolean
Function
object
any
null
undefined
Klassen
Interfaces

Ein erstes Beispiel

Ein erstes Beispiel

```
export class Flug {

    constructor(id: number) {
        this.id = id;
    }

    public id: number; // int + double
    public von: string;
    public nach: string;
    public datum: string; // ISO-Datum
}
```

Ein erstes Beispiel

```
export class Flug {

    constructor(public id: number) {
       this.id = id;
    }

    public id: number; // int + double
    public von: string;
    public nach: string;
    public datum: string; // ISO-Datum
}
```

Ein erstes Beispiel

```
export class Flug {

    constructor(public id: number) {
    }

    public von: string;
    public nach: string;
    public datum: string; // ISO-Datum
}
```

Access Modifier

public | protected | private | readonly

public ist standard

Ein erstes Beispiel

```
// flug-manager.ts
import { Flug } from './flug';

export class FlugManager {

    constructor(private cache: Array<Flug>) {
    }

    search(von: string, to: string): Array<Flug> { [...] }
}
```

Ein erstes Beispiel

```
// flug-manager.ts
import { Flug } from './flug';

export class FlugManager {

    constructor(private cache: Array<Flug>) {
    }

    search(von: string, to: string): Array<Flug> { [...] }

    get count() {
       return this.cache.length;
    }
}
```

Ein erstes Beispiel

```
// flug-manager.ts
import { Flug } from './flug';

export class FlugManager {

    constructor(private cache: Array<Flug>) {
    }

    search(von: string, to: string): Array<Flug> { [...] }

    get count() { return this.cache.length; }
    get flights() { return this.cache; }
    set flights(c: Flug[]) {  this.cache = c; }
}
```

Ein erstes Beispiel

```
// flug-manager.ts
import { Flug } from './flug';

export class FlugManager {

    constructor(private cache: Array<Flug>) {
    }

    search(von: string, to: string): Array<Flug> { [...] }

    get count(): number { return this.cache.length; }
    get flights(): Flug[] { return this.cache; }
    set flights(c: Flug[]): void {  this.cache = c; }
}
```

Ein erstes Beispiel

```
// main.ts
import { FlugManager } from './flug-manager';
import { Flug } from './flug';

let cache: Flug[] = [
    new Flug(…), new Flug(…)
];

let fm = new FlugManager(cache);
let fluege = fm.search("Graz", "Hamburg");

console.debug(fluege);
```

Vererbung

```
export class ExtendedFlugManager extends FlugManager {

    constructor(cache: Array<Flug>) {
        super(cache);
    }

    // Überschreiben
    search() {
        […]
        return super.search();
    }

}
```

Kann auch entfallen: Kein Konstruktor => Konstruktor wird geerbt

DEMO

Projektsetup und Ecosystem

Typische Projektstruktur

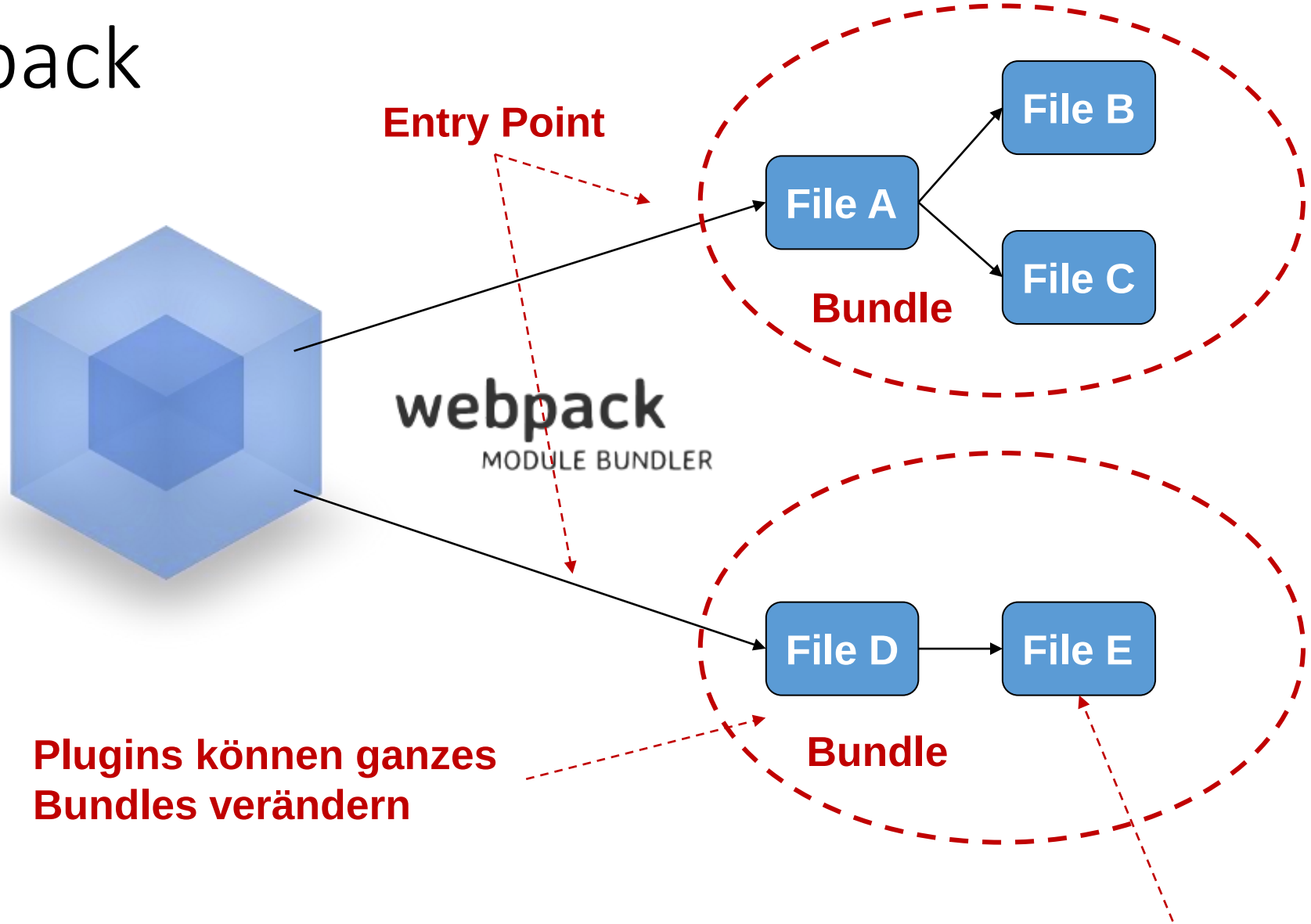
src	• Quellcode (TypeScript, HTML, …)
dist	• Kompilate
node_modules	• Bibliotheken
package.json	• Verweis auf Bibliotheken und Skripte
tsconfig.json	• Konfiguration für TypeScript-Compiler
webpack.config.js	• Konfiguration für Bundler (webpack)

Webpack

Was ist Webpack?

- Build-Tool, Bundling-Lösung
- Schnell
- De-facto Standard
  - (Angular CLI, React, Angular)
- Deklarativ anstatt imperativ
- Unterstützt Node-Pakete (→ defacto Standard)

Webpack


Entry Point
File A
File B
File C
Bundle
webpack
MODULE BUNDLER
File D
File E
Bundle
Plugins können ganzes Bundles verändern
Loader (z.B. .ts → .js)

Webpack-Konfiguration

```
module.exports = {
    entry: {
      'vendor':    './src/vendor.browser.ts',
      'main':      './src/main.browser.ts'
    },
    […]
}
```

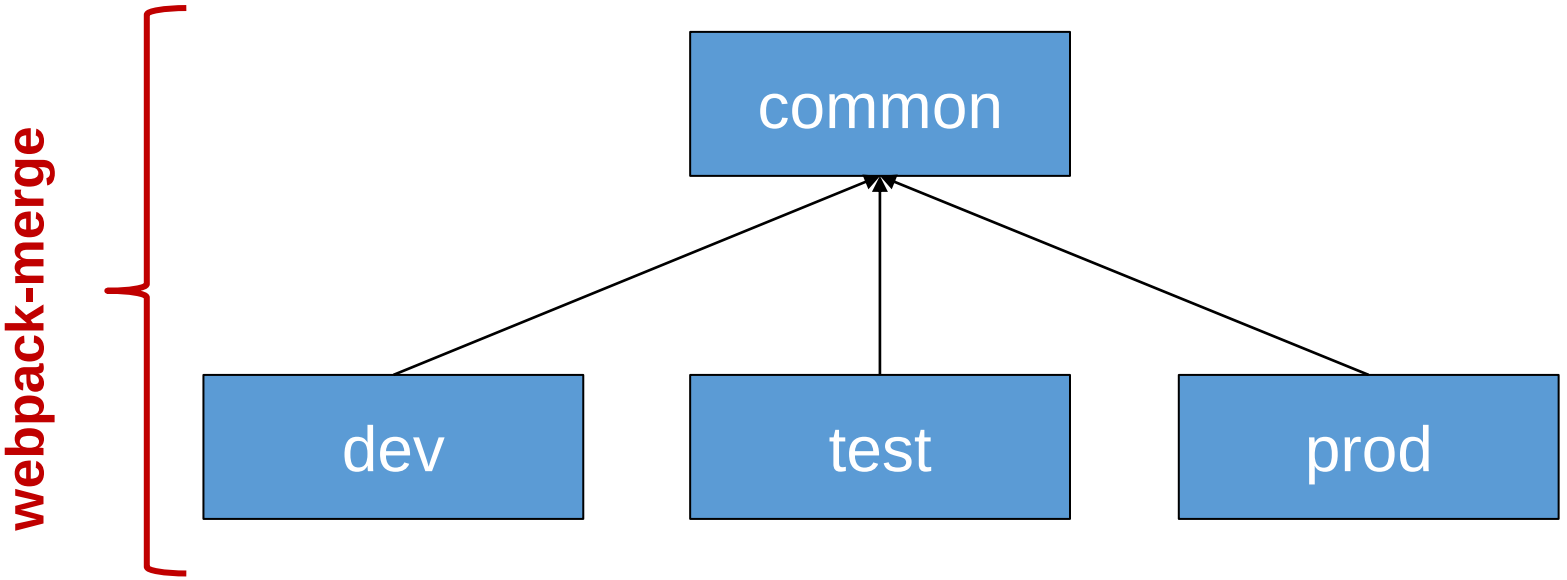
Webpack-Konfiguration

```
module.exports = {
    entry: {
      'vendor':    './src/vendor.browser.ts',
      'main':      './src/main.browser.ts'
    },
    module: {
      rules: [{
          test: /\.ts$/,
          use: ['typescript-loader', 'angular2-template-loader', …]
      }]
    },
    […]
}
```

Webpack-Konfiguration

```
module.exports = {
    entry: {
      'vendor':    './src/vendor.browser.ts',
      'main':      './src/main.browser.ts'
    },
    module: {
      rules: [{
          test: /\.ts$/,
          use: ['typescript-loader', 'angular2-template-loader', …]
      }]
    },
    plugins: [
      new HtmlWebpackPlugin({ template: 'src/index.html' })
    ]
}
```

Webpack-Konfiguration



webpack.dev.config

```
const commonConfig = require('./webpack.common.js');

module.exports = webpackMerge(commonConfig, {

    devtool: 'cheap-module-source-map',

    plugins: [
        new DefinePlugin({
            'ENV': '"development"'
        })
    ],

    devServer: {
      port: 8080
    }

});
```

Webpack ausführen

- webpack --config webpack.dev.js
- Standard-Konfiguration: webpack.config.js

DEMO

NPM und package.json

package.json

```
{
    "dependencies": {
        "@angular/common": "2.0.0",
        […]
    },
    […]
}
```

package.json

```
{
    "dependencies": {
        "@angular/common": "~2.0.0",
        […]
    },
    […]
}
```

package.json

```
{
    "dependencies": {
        "@angular/common": "~2.0.0",
        […]
    },
    […]
}
```

package.json

```
{
    "dependencies": {
        "@angular/common": "^2.0.0",
        […]
    },
    […]
}
```

package.json

```
{
    "dependencies": {
        "@angular/common": "^2.0.0",
        […]
    },
    […]
}
```

package.json

```
{
    "dependencies": {
        "@angular/common": "^2.0.0",
        […]
    },
    "devDependencies": {
       "webpack": "^1.12.9",
       "webpack-dev-server": "^1.14.0",
        […]
    },
    […]
}
```

package.json

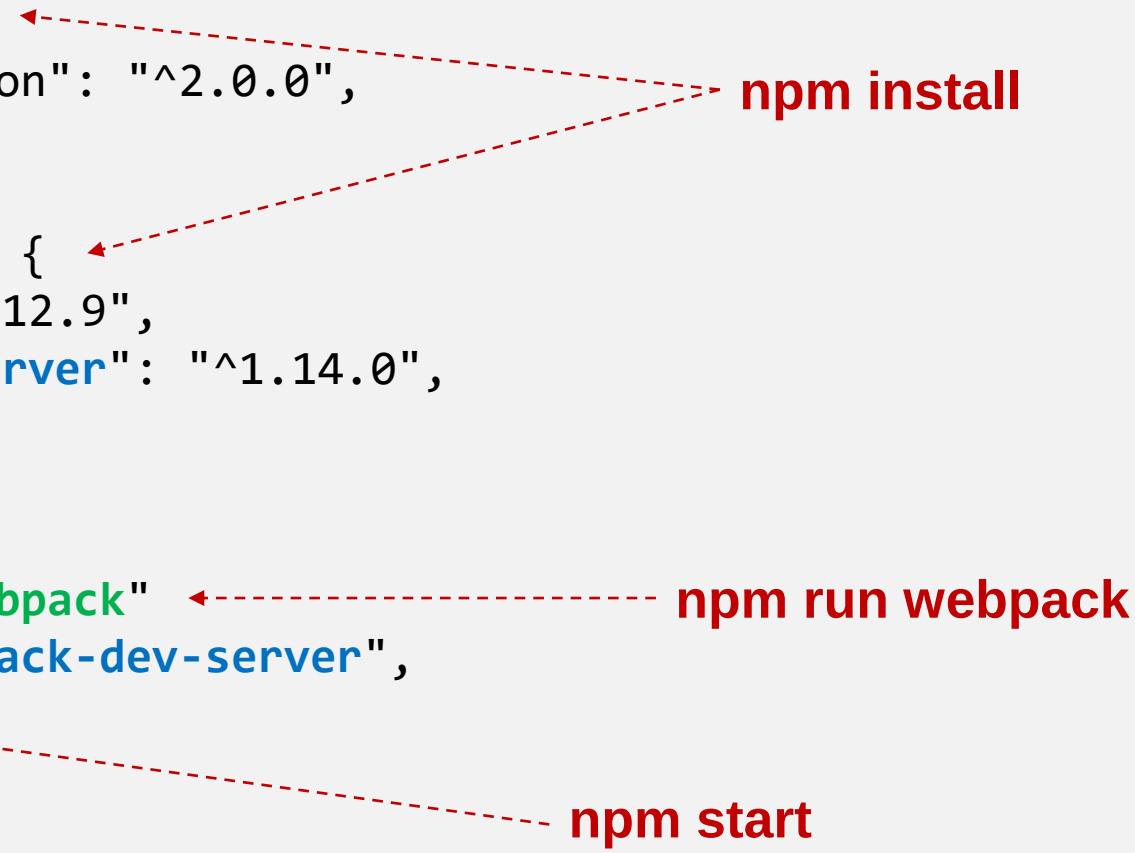
```
{
    "dependencies": {
        "@angular/common": "^2.0.0",
        […]
    },
    "devDependencies": {
       "webpack": "^1.12.9",
       "webpack-dev-server": "^1.14.0",
        […]
    },
    "scripts": {
        "webpack": "webpack"
        "start": "webpack-dev-server",
    },
    […]
}
```

package.json

```
{
    "dependencies": {
        "@angular/common": "^2.0.0",
        […]
    },
    "devDependencies": {
       "webpack": "^1.12.9",
       "webpack-dev-server": "^1.14.0",
        […]
    },
    "scripts": {
        "webpack": "webpack --config webpack.dev.js"
        "start": "webpack-dev-server",
    },
    […]
}
```

package.json

```
{
    "dependencies": {
        "@angular/common": "^2.0.0",
        […]
    },
    "devDependencies": {
       "webpack": "^1.12.9",
       "webpack-dev-server": "^1.14.0",
        […]
    },
    "scripts": {
        "webpack": "webpack"
        "start": "webpack-dev-server",
    },
    […]
}
```



Pakete per npm installieren

- Option 1
  - Paket in package.json eintragen
  - npm install
- Option 2
  - npm install *bibliothek* --save
  - npm install *bibliothek* --save-dev

DEMO

TypeScript: Speedrun ;-)

Typen und Typherleitung

Arbeiten mit Typen

```
let name: string = "Max Muster";

let plz: number = 45257;

let autor: boolean = true;
```

Typherleitung

```
let website1 = "http://www.ix.de"; // Typ: string

website1 = 123; // falsch !!!
```

Verwendung von any

```
let website2: any = "http://www.softwarearchitekt.at";

website2 = 1;
```

Implizite Verwendung von any

```
let website3;

website3 = "http://www.typescript.org"

website3 = 4;
```

any ist immer zuweisungskompatibel

```
let website4: string;

let test: any = "www.softwarearchitekt.at";

website4 = test;
```

Union-Types

```
let nameOrNumber: string | number;

nameOrNumber = "Max";

nameOrNumber = 17;
```

Abstrakte Klassen

Abstrakte Klasse

```
abstract class Kontakt {

    […]

    firstName: string;
    lastName: string;

    public abstract sendeNachricht(msg: string);

    get fullName(): void {
        return this.firstName + " " + this.lastName;
    }

    […]
}
```

Vererbung

```
class Kunde extends Kontakt {

    public kundenArt: string;

    public sendeNachricht(msg: string) {
        […]
    }

}
```

Instanziierung

```
let k1 = new Kunde(123, "Max Muster", "Essen");


let ok1: boolean = k1 instanceof Kunde;
let ok2: boolean = k1 instanceof Kontakt;
```

Type Assertions

Type Assertions

```
let k: Kontakt = new Kunde(123, "Max Muster", "Essen");
```

Type Assertions

```
let k: Kontakt = new Kunde(123, "Max Muster", "Essen");


[…]
let art = k.kundenArt;  // Funktioniert nicht!
```

Type Assertions

```
let k: Kontakt = new Kunde(123, "Max Muster", "Essen");


[…]
let kunde = k as Kunde;
let art = kunde.kundenArt;  // OK
```

Type Assertions (Alternative)

```
let k: Kontakt = new Kunde(123, "Max Muster", "Essen");


[…]
let kunde = <Kunde>k;
let art = kunde.kundenArt;  // OK
```

DEMO

AbstractFlightManager

Interfaces

```
interface IKontakt {
        id: number;
        name: string;
        ort?: string;
        plz: number;
        erfassungsdatum: any;
        getInfo(): string;
}
```

```
class Kontakt implements IKontakt {
        […]
}
```

Sub-Typing

```
interface IKontakt {
        id: number;
        name: string;
}

class Kontakt implements IKontakt {
         id: number;
         name: string;
}

let k: IKontakt = new Kontakt();
```

Strukturelles Sub-Typing

```
interface IKontakt {
        id: number;
        name: string;
}

class Kontakt /* implements IKontakt */ {
        id: number;
        name: string;
}

let k: IKontakt = new Kontakt();
```

Strukturelles Sub-Typing

```
interface IKontakt {
        id: number;
        name: string;
}

class Kontakt /* implements IKontakt */ {
         id: number;
         name: string;
}

let k: IKontakt = new Kontakt();

let k: IKontakt = { id: 17, name: "Max Muster"};
```

DEMO

IFlight

Generics

```
class ReadOnly<T> {
    private data: T;
    constructor(data: T) {
        this.data = data;
    }
    public getData(): T {
        return this.data;
    }
}

let readOnlyNumber = new ReadOnly<number>(42);
console.debug(readOnlyNumber.getData());
```

Typparameter mit Einschränkungen

```
class Dienstvertrag<T extends Kontakt> {

    constructor(private angestellter: T) { }

    public get dienstnehmerName(): string {
        return this.angestellter.fullName;
    }

    […]

}
```

DEMO

Invoice<T>

Funktionen

Funktionen

```
function sayHello(name: string = "noname"): void {
    console.debug("Hallo " + name);
}

sayHello("Max");
sayHello();
```

Optionale Parameter

```
function sayHello(name?: string): void {
    if (name) {
        console.debug("Hallo " + name);
    }
    else {
        console.debug("Hallo!");
    }
}
```

Rest Parameter

```
function sayHello(...names: string[]): void {
    for (let name of names) {
        console.debug("Hello " + name);
    }
}

sayHello("Max", "Susi", "Anna");
```

Rest Parameter

```
function sayHello(...names: string[]): void {
    for (let name of names) {
        console.debug("Hello " + name);
    }
}

sayHello("Max", "Susi", "Anna");
```

Union-Types

```
function sayHello(namesOrId: string | number): void {
    […]
}

sayHello("Max");
sayHello(0);
```

Funktions-Typen

```
let berechnung: (a: number, b: number) => number;

function add(x: number, y: number): number {
    return x + y;
}

berechnung = add;
console.debug("Ergebnis: " + berechnung(17, 2));
```

Anonyme Funktionen

```
let berechnung: (a: number, b: number) => number;

berechnung = function (a, b) {
    return a + b;
}

console.debug("Ergebnis: " + berechnung(17, 2));
```

Lambda-Ausdrücke (Arrow-Functions)

```
let berechnung: (a: number, b: number) => number;

berechnung = (a, b) => {
    return a + b;
}

console.debug("Ergebnis: " + berechnung(17, 2));
```

Lambda-Ausdrücke (Arrow-Functions)

```
let berechnung: (a: number, b: number) => number;

berechnung = (a, b) => a + b;

console.debug("Ergebnis: " + berechnung(17, 2));
```

Lambda-Ausdruck bindet this!

this außerhalb des Ausdrucks == this im Ausdruck

DEMO

find(f => f.from == 'Graz')

Enums

Enums

- enum Direction { UP, DOWN, LEFT, RIGHT }
  let d = Direction.UP;
- enum Direction { UP = 7, DOWN, LEFT, RIGHT }
- type Direction = 'UP' | 'DOWN' | 'LEFT' | 'RIGHT';
  let d: Direction = 'UP';

DEMO

FlightType

Ambiente Deklarationen

Ambiente Deklarationen

- Typangaben für bereits bestehende JavaScript-Variablen
- Notwendig, um Typen für bestehende JavaScript-Frameworks zu definieren
  - Beispiele
    - Globale Variable *angular* bei AngularJS
    - Globale Variable *$* bei jQuery

Beispiel: jQuery

```
$("#someElement").show();

$("#someElement").hide('slow');

$("#someElement").show(300);
```

Ambiente Deklarationen

```
interface JQueryStatic {
    (selector: string): JQuery
}

interface JQuery {
   show(speed?: number | string);
   hide(speed?: number | string);
}

declare let $: JQueryStatic;

$("#someElement").show();
```

Ambiente Deklarationen

- Zahlreiche Typings-Dateien (mit ambienten Deklarationen) online

Installation jQuery + Typings

- npm install jquery --save
- npm install @types/jquery --save-dev

DEMO

jQuery-based UI