# Angular

Mit Github und Visual Studio Code

Vorbereitung einschließlich Github

1. Node.js muss auf dem Rechner installiert sein
2. In der Kommandozeile mit **npm install –g @angular/cli** die letzte Version von Angular installieren (-g = global auf dem Rechner)
3. In den Ordner wechseln, wo das neue Projekt erstellt werden soll
4. In der Kommandozeile **ng new ProjektName** eingeben
5. In das Projekt wechseln und mit **ng serve** den Server starten
6. Im Browser über **localhost:4200** steht das Projekt zur Verfügung
7. Projekt für Github vorbereiten
   1. Auf der Kommandozeile im Projektordner **git init** eingeben
   2. Auf Github neues Repository anlege
   3. In der Kommandozeile   
      **git remote add ProjektName https://github.com/Benutzername/ProjektName.git** eingeben (github.com/BirgitFl/learnAngular.git)

Visual Studio Code

Nicht benötigte Ordner ausblenden:

* **Strg+Umschalten+p** und dort nach Benutzereinstellungen suchen
* In der **settings.json** die Einstellungen unter **file.exclude** anpassen

Projektstruktur

* Dateien unterhalb des Projektordners dienen der Konfiguration (CLI, Git usw.)
* Alle wichtigen Dateien und Ordner liegen innerhalb des Ordners src
* Dateien direkt unter src dienen der Projektkonfiguration.
  + Main.ts ist die Datei die Angular als aller erstes startet
  + Index.html – einzige Datei der Webseite
  + Styles.css – die externen Stylesheets
  + Sowie einige Konfigurationsdateien
* Der Ordner e2e enthält die End-to-End Tests
* Assets beinhaltet eingebundene Bilder
* Environments teilt Angular mit, ob das die Umgebung eine Entwicklungs- oder Produktionsumgebung ist
* Der Ordner apps enthält alle Dateien für das entsprechende Projekt. Hier werden die Components, die Module und zugehörigen html bzw. css Dateien abgelegt

Typescript

* Erweiterung zu Javascript, ergänzt es mit eigenen Features (Typen, Klassen, Module…)
* Möglichkeit mit expliziten Typenangaben zu arbeiten um Fehler zu reduzieren
* Beinhaltet next gen JS-Features
* Stellt auch in Zukunft fehlende JS-Features zur Verfügung (Interfaces, Generics…)

Typescript installieren: **npm install typescript –g**

Bootstrap

Bootstrap installieren**: npm install –save bootstrap**

- in angular-cli.json unter **styles** eintragen: **"../node\_modules/bootstrap/dist/css/bootstrap.min.css",**

Wichtigste Dateien zum Starten

main.ts

import { enableProdMode } from '@angular/core';

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';

import { environment } from './environments/environment';

if (environment.production) {

enableProdMode();

}

platformBrowserDynamic().bootstrapModule(AppModule);

// ruft die Datei app.module.ts auf

app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms';

import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';

import { OtherComponent } from './other/other.component';

@NgModule({

declarations: [AppComponent, OtherComponent],

// innerhalb der [] kann auch jede Component in einer eigenen Zeile stehen

// hier hintereinander weg geschrieben um Platz zu sparen

imports: [BrowserModule, FormsModule, HttpModule],

providers: [],

bootstrap: [AppComponent]

//teilt Angular mit, dass die Datei app.component.ts geladen werden soll

})

export class AppModule { }

//dieser Name sagt der main.ts dass es genau diese Datei laden soll

app.component.ts

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

// teilt Angular mit, an welcher Stelle diese Component in der Index.html geladen werden soll

templateUrl: './app.component.html',

//enthält den Html-code damit der Inhalt dieser Component ausgegeben wird

styleUrls: ['./app.component.css']

// ein Array für die Aufnahme der CSS-Dateien

/\*\* weitere Möglichkeit das Template und die Styles direkt in dieser Datei zu schreiben

\* mehrere Zeilen werden innerhalb eines Backticks geschrieben (links neben der Backshift-Taste)

\* template:`

\* <h1>{{title}}</h1>

\* `,

\* styles:[`

\* h1{color:red;}

\* `]

\*/

})

export class AppComponent {

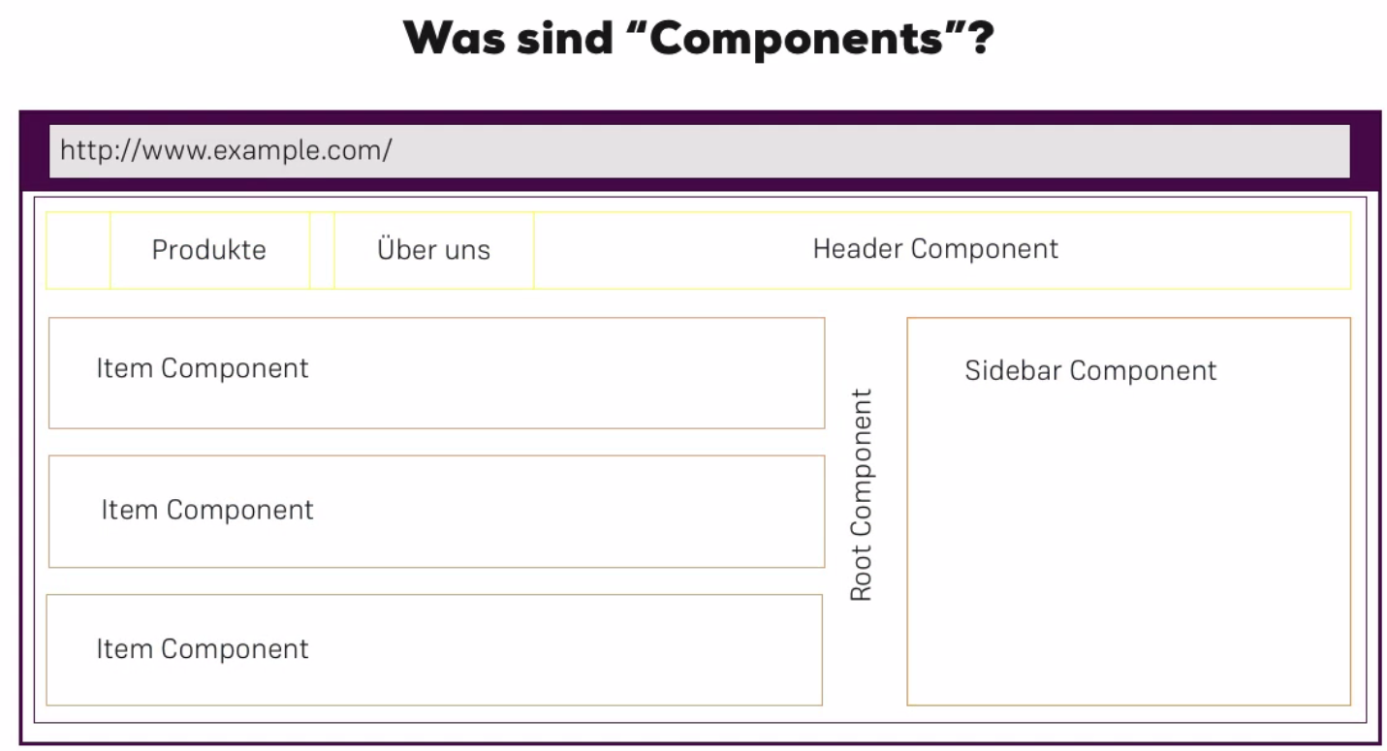
// der Name welcher zur Identifizierung der Datei nötig ist

title = 'yeah app works!';

}

Was sind Components?

* am Beispiel einer normalen Webseite:



Inhalte einer Component

* Components sind Klassen mit einem Decorator(@)
* Müssen mindestens ein Template enthalten
* Beinhalten den Selector über welchen Angular erkennt an welcher Stelle was geladen werden soll
* Selector ist ein HTML-Element und muss einen eindeutigen Namen haben (kein HTML-Namen), dazu einen Identifier vor den Namen (hier app-)
* Der Selector wird standardmäßig als normales HTML-Element verwendet
  + Bsp.: **selector:“app-root“** 🡪 **<app-root>** in der HTML-Datei
  + Bsp.: **selector:“#app-root“ 🡪 <div id=“app-root“>**
  + **Bsp.: selector:“.app-root“ 🡪 <div class=“app-root“>**

Neue Component erstellen

**ng g c name 🡪 ng generate component ComponentName**

* Erstellt einen Ordner mit drei Dateien (ComponentName.component.ts, ComponentName.component.html und ComponentName.component.spec.ts)

**ng g c name --flat -it -is --spec false 🡪erstellt die Component im Ordner wo der Befehl aufgerufen wird (--flat), nutzt Inline-Template (-it) und Inline-Style (-is) und erstellt keine \*.spec.ts**

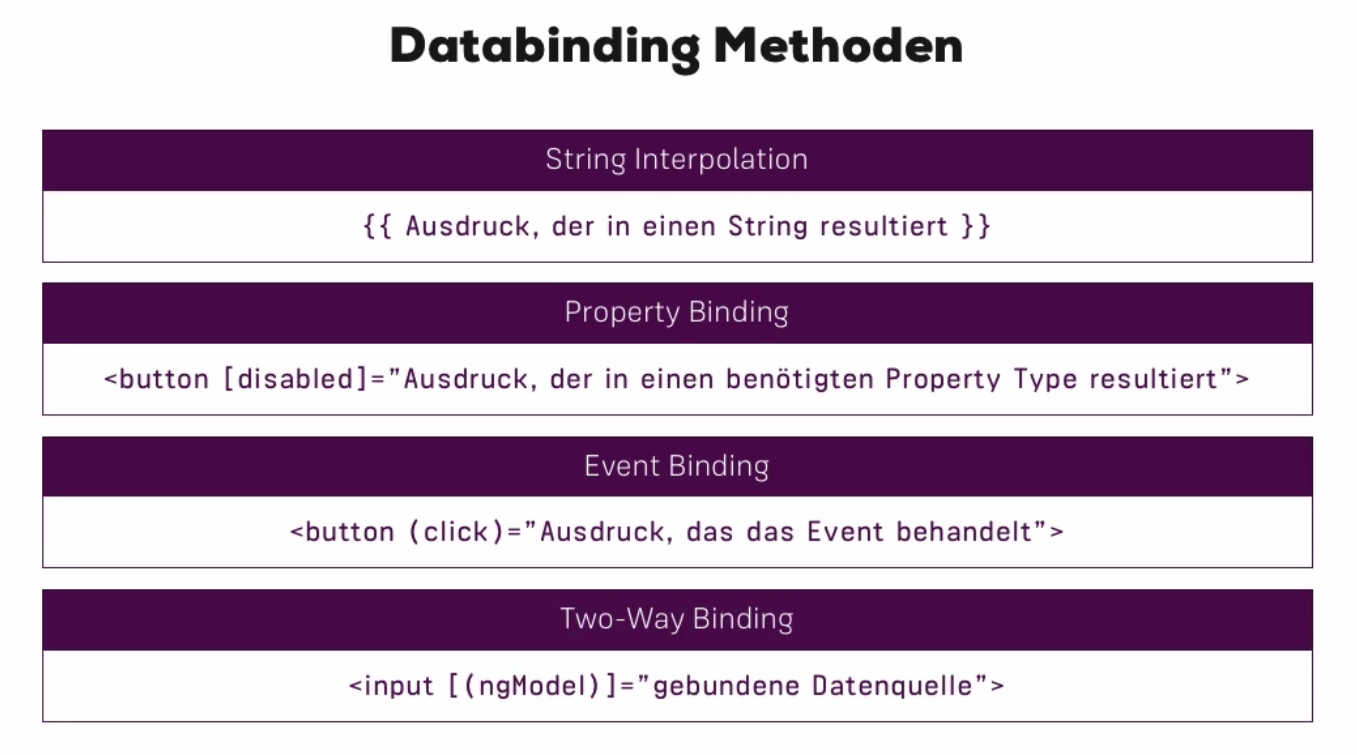
**app.modules.ts** muss unter @NgModule erfahren welche Components die SAP (Single Page Application) hat

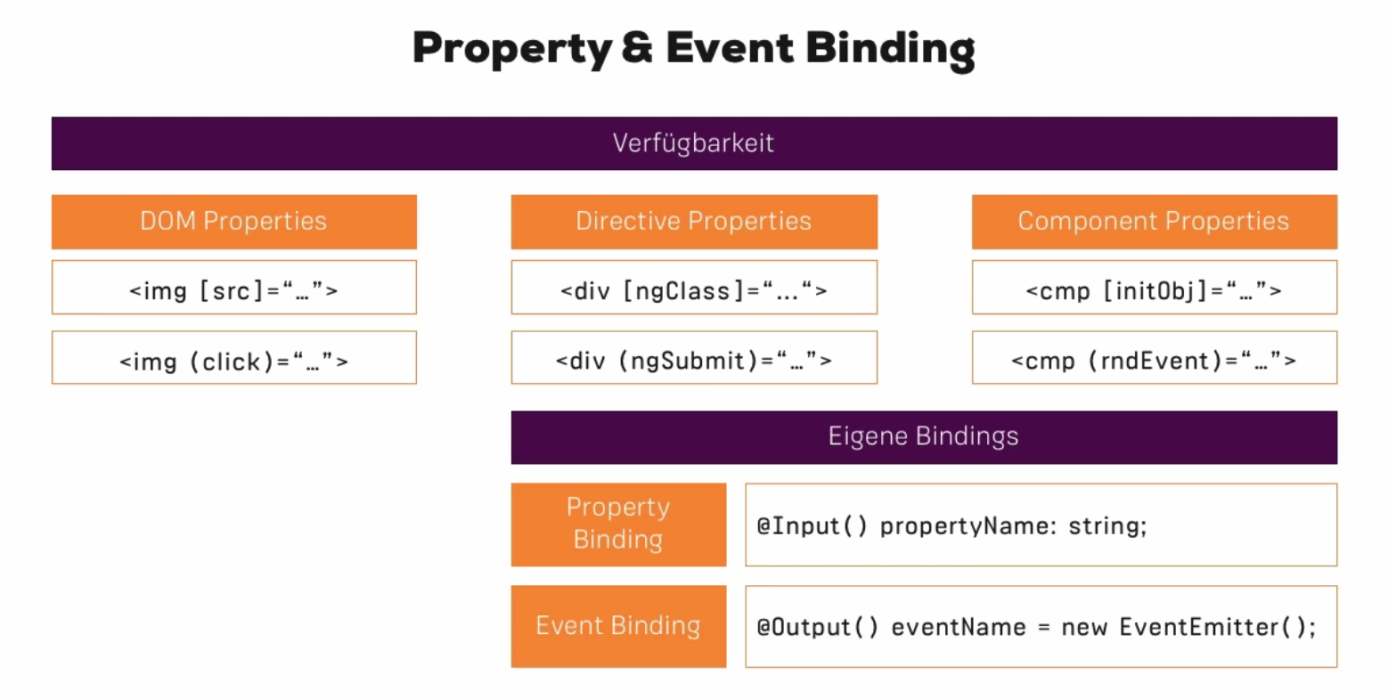
wichtig auch der Import dieser Components in dieser Datei

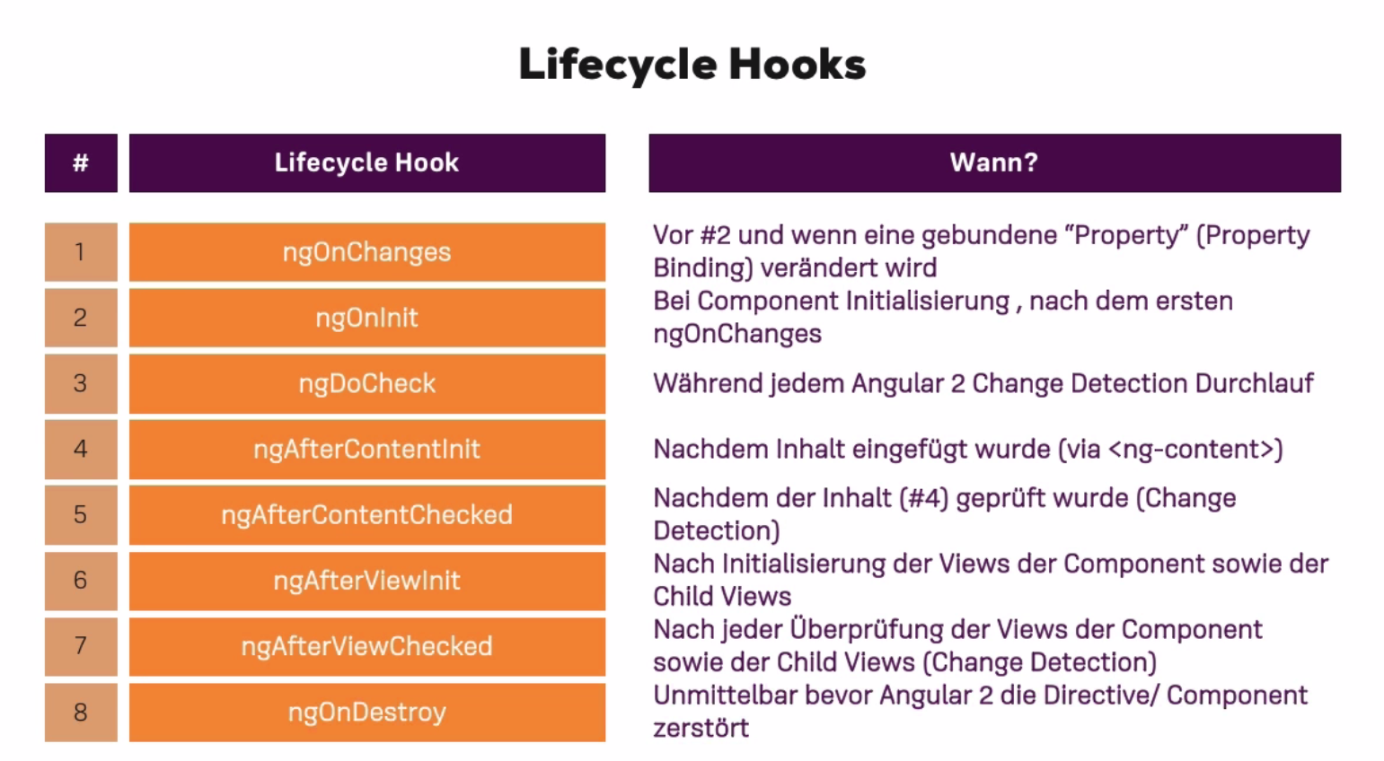
**Stylesheets** gelten immer nur in der Component wo sie definiert werden. Styles die allgemeingültig sein sollen, müssen innerhalb der style.css welche im src-Ordner liegt definiert werden.

Über spezielle Attribut-Selectors emuliert Angular für jedes HTML-Element ein sogenanntes Shadowdom.

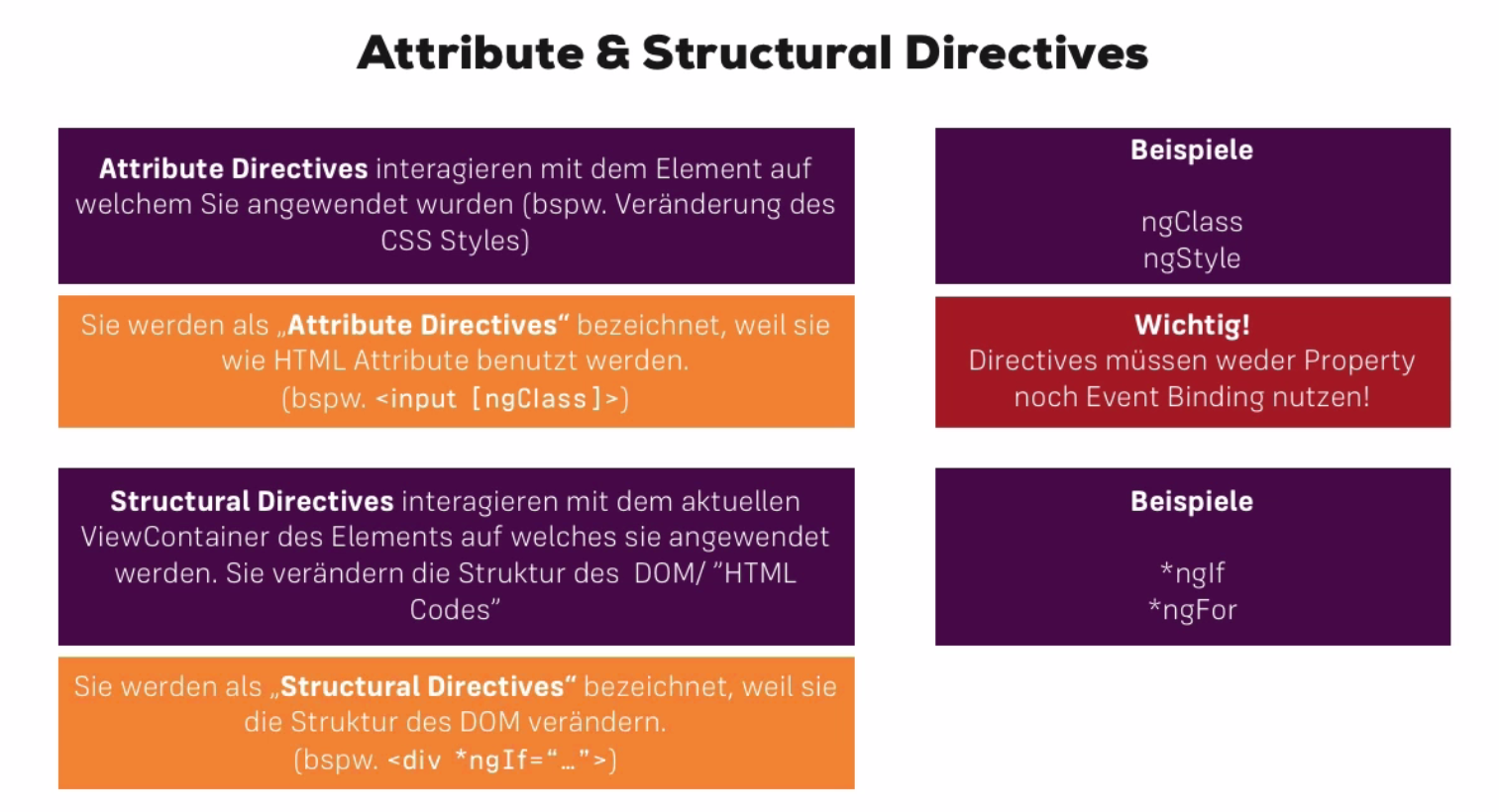
**<ng-content>** dient als Platzhalter für Inhalte.







Directives



Eigene Directive erstellen: **ng g d DirectivName**

Directives haben kein Template und kein Style.