# Handout Angular

## Angular Komponenten

Eine Komponente in Angular besteht aus 4 verschiedenen Dateien, welche alle ihren eigenen Zweck erfüllen:

- HTML (`app.component.html`)
- CSS (`app.component.css`)
- TypeScript (`app.component.ts`)
- TypeScript Konfiguration (app.component.spec.ts)

In der `app.component.html` wird definiert, wie eine Komponente beim Laden der Webseite als HTML Dokument dargestellt wird. Dieses HTML Template ist allerdings kein Standard HTML. Es hat einige extra Features wie das Benutzen von Platzhaltern, welche bei **HTML Extras** weiter erklärt werden.
Die `app.component.css` enthält alle Styles, die zu dem HTML der Komponente gehören. In unseren Aufgaben wird dies allerdings nicht weiter gebraucht.
Der gesamte Code der Komponente liegt in der `app.component.ts`. Sämtliche Attribute der dort deklarierten Klasse sind auch in dem HTML Template der Komponente benutzbar.

## HTML Extras

Angular bringt viele extra Features, welche man in den HTML Templates seiner Komponenten benutzen kann. Hier stellen wir eine kleine Liste von Extras genauer vor:

### Platzhalter für Klassenattribute

Am Ende der `app.component.ts` befindet sich eine Klassendefinition für die Komponente. Alle Attribute dieser Klasse können im HTML Template dieser Komponente verwendet werden. Z.B:

`app.component.ts`:

```
export class AppComponent {
      title: “Hello World”
}
```

`app.component.html`:

```
<h1>{{ title }}</h1>
```

Zuerst definieren wir ein Attribut der Komponentenklasse namens `title`. Dieses Attribut kann dann in dem HTML Template benutzt werden, indem wir es in doppelt geschweifte Klammern schreiben.

## Binden von Funktionen

Wenn wir nun Code ausführen wollen, statt nur statische Werte zu verwenden, müssen wir der Komponentenklasse eine Methode geben und diese dann an ein JavaScript Ereignis binden. So wird die Methode immer dann ausgeführt, wenn das Ereignis ausgelöst wird. Z.B:
`app.component.ts`:

```
export class AppComponent {
      saySomething() { console.log(“Hello World”); }
}
```

`app.component.html`:

```
<button (click)="saySomething()">Click Me!</button>
```

Wieder definieren wir zuerst etwas in unserer Komponentenklasse, in diesem Fall unsere Methode. Dann binden wir diese Methode an ein JavaScript Ereignis.
Das Ereignis selbst kann hierbei jedes beliebige JavaScript Ereignis sein plus ein paar spezielle Ereignisse, welche durch Angular gestellt werden.

## @if und @for

Einige Besonderheiten von Angular müssen erst in eine Komponente als Abhängigkeit importiert werden:
`app.component.ts`:

```
import { CommonModule } from '@angular/common';
…

@Component({
  imports: [CommonModule],
…
})
```

`@if` und `@for` gehören zu dieser Kategorie an Features, also müssen wir zuerst `CommonModule` als Abhängigkeit angeben.

`@if` ist selbsterklärend mit einem kleinen Beispiel:
`app.component.html`:

```
@if (item.done) {
    …
} @else {
    …
}
```

Bei `@for` gibt es ein paar Besonderheiten, welche es zu beachten gibt:
`app.component.html`:

```
@for(item of items; track item; let i = $index){
    …
}
```

`item of items`: Iteriert über einen array `items`. Funktioniert wie `foreach` in anderen Sprachen.
`track item`: Gibt an, was genau Angular bei für das neu rendern der Seite beachten muss. Etwas zu kompliziert für unseren Workshop, um genauer darauf einzugehen.
`let i = $index`: Optional. Gibt eine Variable `i` an , welche immer der Index des Elementes der momentanen Iteration ist.

## HTML Elemente als Template Variablen

Angular bietet die Möglichkeit HTML Elemente direkt als Variablen zu markieren und diese dann im HTML Template selbst direkt auch zu verwenden. So lässt sich ein Umweg über den Code der Komponente vermeiden.
`app.component.html`:

```
<input
    #newItem
    (keyup.enter)="addItem(newItem.value); newItem.value = ''"/>
```

In diesem Beispiel wird ein Input Element direkt mit `#newItem` als Template Variable gekennzeichnet. Daraufhin wird auf das `keyup.enter` Event 2 Funktionalitäten gebunden: Zuerst wird die Funktion `addItem()` aufgerufen und der Wert des Input-Elementes wird als Parameter übergeben. Danach wird der Wert des Inputs zurückgesetzt.

# Angular Workshop: To-Do-Liste

## Ziel

Bau eine To-Do-App in Angular. Man gibt eine Aufgabe ein, drückt Enter, sie landet in der Liste. Man kann sie als erledigt markieren und auch wieder löschen.

---

## Aufgaben

Zuerst sollte man sich die gesamte Struktur anschauen, um zu verstehen, wie alles aufgebaut ist und wie es funktioniert. Anschließend empfiehlt es sich, die HTML-Komponente (app.component.html), den Inhalt vollständig zu löschen. Statt eine neue Komponente zu erstellen, arbeitet man am besten mit der bereits von Angular bereitgestellten Komponente. Für die ganzen Aufgaben wird kein CSS benötigt, das fällt weg.

1. **Input-Feld für neue Aufgaben:**
    - Erstelle ein Input-Feld in der bereits vorhandenen Angular-Komponente.
    - Nutze das `(keyup.enter)` Event, damit die Aufgabe durch Drücken der Enter-Taste hinzugefügt wird.
    - Danach muss das Feld wieder leer sein und der Text mit der bestehenden Liste verbunden werden.
2. **Liste der Aufgaben anzeigen:**
    - Die Aufgaben müssen in einer Liste untereinander stehen.
    - Jede Aufgabe hat zwei Zustände: offen oder erledigt.
3. **Aufgaben als erledigt markieren:**
    - Füge eine Checkbox oder einen Button hinzu, mit dem man eine Aufgabe abhaken kann.
    - Wenn eine Aufgabe erledigt ist, muss das sichtbar sein (z. B. durchgestrichen oder andere Farbe).
4. **Aufgaben löschen:**
    - Neben jeder Aufgabe soll ein Button sein, mit dem man sie wieder löschen kann.
    - Sobald man draufdrückt, muss sie sofort verschwinden.