# Javascript Angular Schulung Fortgeschritten

**1. JiT / AoT**

|  |  |
| --- | --- |
| JIT | AOT |
| * Technik zum Übersetzen von Progammen in Maschinencode zur Laufzeit | * Technik zum Übersetzen von Progammen in Maschinencode vor der Ausführung |
|  | + StartUp Time wird schneller |
|  |  |

**2. Angular CLI**

ng new

--dry-run -> Zeigt was erstellt werden würde

--prefix -> Definiert einen Prefix für erstellte Komponenten

--style -> legt die art der style dateien fest (css, scss)

--routing true -> erstellt die datei app-routing.module.ts

ng serve

--aot -> kompiliert den code mit Ahead of Time

ng build

--prod -> minimalisiert den context

-> überprüft auf unnötige Dateien und schmeißt sie weg (Tree shaking)

-> Debugging wird ausgehebelt

**Dateistruktur**

.editorconfig -> Konfigurationen für den Editor

.gitignore -> Bestimmt welche Dateien nicht von Git beobachtet werden sollen

angular.json -> Definiert das Projekt

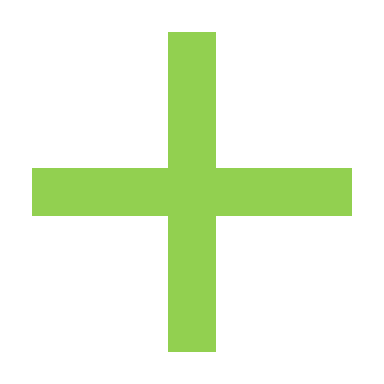
package.json -> Spezifiziert die npm package.json behandlung



In Angular 6 müssen Services nichtmehr als provider in ***„app.module.ts“*** angegeben werden. Funktioniert nun über ***„@injectable( provider in 'root' )“***

**3. Progressiv Web App**

Webanwendung die herunterladbar und installierbar sind.

Startup-Zeit deutlich schneller

**4. Debugging**

Debuging tools Angular:

* [Angular Augury](https://augury.angular.io/)

*Augury hilft Angular 2.0-Entwicklern, die Anwendung durch Komponentenbäume und visuelle Debugging-Tools zu visualisieren. Entwickler erhalten sofort Einblick in ihre Anwendungsstruktur, Änderungserkennung und Leistungsmerkmale.*

debugger; -> Generiert einen breakpoint an der stellle im code wo er plaziert worden ist

**5. RxJs**

[RxJS](https://rxjs-dev.firebaseapp.com/) ist eine Bibliothek für die reaktive Programmierung mit Observables, um das Erstellen asynchroner order Callback- basierter Codes zu vereinfachen

Observables -> dienen dem asynchronen nachladen von Daten

-> Könen subscribed werden müssen aber auch wieder unsubscribed werden

Zum unsubscriben sollte ein neuer lifecyclehook implementiert werden ***„onDestroy“*.** Liste mit allen Subscriptions erstellen, nachdem die komponente nicht mehr aktiv ist, sollte durch die Liste der Subscription iteriert werden und alle Subsciptions aufgehoben werden

Cold Observable -> Fängt wieder vom neuen an wenn eine neue Subscription erfolgt

Hot Observable -> Observer hängt sich an das arbeitende Observable dran

**Subscriben eines Observables in der Template Schicht**

Syntax :

{{$appService.myVal | async}}

**6. Change Detection**

markForCheck() -> Aktualisiert alles bis hoch zum root Verzeichnis

detectChanges() -> Aktualisiert aktuelle Eltern Komponente und ihre Kinder Komponenten

**7. Routing**

Der Angular *Router* ermöglicht die Navigation von einer Ansicht zur nächsten, wenn Benutzer Anwendungsaufgaben ausführen.

<router-outlet></router-outlet> -> Platzhalter für Komponenten die mit Routing angesprochen werden

routerlink - Directive -> Definiert die Route – SEO friendly

routerLinkActive - Directive -> Damit können CSS Klassen vergeben werden für die Aktive Route

Activated Route - Service -> Gibt Informationen über die aktuelle Route

**8. Guards**

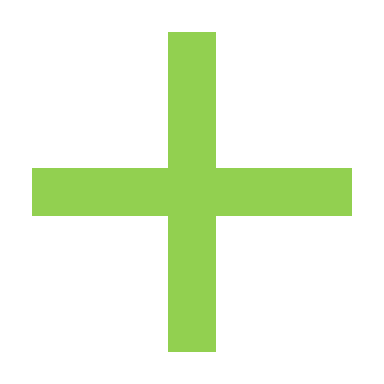
Bestimmt Routenwechsel mit bestimmten Konditionen. Die Guards weden innerhalb der Routen definitionen definiert

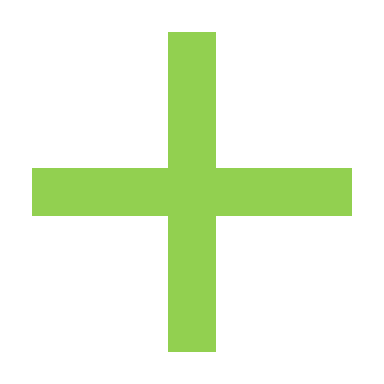
* Resolve Guard + Service
* CanActive Guard + Service

**9. Formulare**

1.Vorlagen – getrieben -> Vorlage gibt Formularmodel und Validatoren vor

2. Reaktiv (Daten - getrieben)

Schneller

Dynamisch

dirtry -> es hat sich aktiv was verändert in einem Input Feld

touched -> das feld wurde fokosiert

Bsp:

const heroForm = new [FormGroup](https://angular.io/api/forms/FormGroup)({

'name': new [FormControl](https://angular.io/api/forms/FormControl)(),

'alterEgo': new [FormControl](https://angular.io/api/forms/FormControl)(),

'power': new [FormControl](https://angular.io/api/forms/FormControl)()

});