



Angular 4 fra Scratch til advanced med Hello World eksempler

download eksempler

Angular 4

Udvikling af Angular apps

© Copyright ZoomTek 2017 - Søren Anderson. All rights reserved.

Dette materiale må ikke kopieres, reproduceres, indscannes, affotograferes, overføres eller reduceres til noget elektronisk medie eller til maskinlæsbar form eller papir. Ej heller må materialet afskrives på papir e.l. Alle forbehold gælder materialet i sin helhed eller dele heraf. Dette forbehold gælder også for cd-rommer, papirudskrifter m.m. som måtte følge med dette materiale. Brud på ovenstående må kun ske med dateret skriftlig tilladelse fra Søren Anderson, ZoomTek. Selv om materialet bliver løbende tjekket for fejl, tager ZoomTek/Søren Anderson intet ansvar for fejl i materialet. Dette gælder både for egentlige fejl, men også for udeladelse af information. ZoomTek/Søren Anderson er ikke ansvarlig for handlinger udført på server, klienter m.m. ud fra anvisninger i dette materiale. Endvidere henvises til forretningsbetingelser på www.zoomtek.dk/salglevbetingelser.html. Kurset afholdes på kundens softwarelicenser. Dette værk - dele af det eller i sin helhed - må kun benyttes af personer, der har deltaget på tilsvarende ZoomTek kursus.

I dette kursusmateriale refereres til forskellige firmaer og softwareprodukter. Især Microsoft, IBM, Word, Excel, PowerPoint, Access, Project, XML, Sun, Google, SharePoint, SharePoint Designer, Infopath, IBM Lotus Notes, IBM Lotus Domino, Inotes, SameTime, LotusScript m.fl. Disse firmaer og produkter er alle registrerede varemærker. I kursusmaterialet kan forekomme tekstmateriale fra diverse eksterne kilder. Kildeangivelse og rettigheder vil i så fald være angivet.

Kursusmaterialet er trykt d. 24. august 2017

Dansk og Engelsk sprogbrug

It-verdenen er fuld af engelske udtryk. Derfor har undertegnet for mange år siden opgivet at entals- og flertalsbøje danske og engelske ord korrekt - især når de forekommer i samme sætning. Derfor kan der f.eks. sagtens forekomme engelske ord med dansk 'er' bøjning i flertal. Undskyld, Klaus Rifbjerg !!

Skærmdumps er ofte klippet

Af pladshensyn er der ofte klippet i grafikken, så menuer synes kortere, og dialogbokse virker mindre, end de i virkeligheden er.

Ris og Ros

Vi er altid glade for at høre, hvad du synes om denne bog. Har du ris eller ros - eller måske forslag til forbedringer eller nye bøger - hører vi gerne fra dig på info@zoomtek.dk eller tlf. 4250 5040.

Med venlig hilsen

Søren Anderson

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Forord

Angular Kursus

Kurset som hører til dette materiale er sat til 4 dage. Der anbefales 4 dage, hvis kursisterne intet ved om Angular. Ellers kan kurset køres igennem på kortere tid. Kursusmaterialet er lavet til både Angular 2 og 4. Det er angivet hvis kode eller andet kun kan benyttes i den ene version.

Velkommen til kursus hos ZoomTek.dk

Velkommen til kurset 'Udvikling af Angular apps' hos ZoomTek. ZoomTek udbyder kurser i hele Danmark inden for især:

SharePoint Kurser (27 kurser)	Webudvikling (47 kurser)	XML (20 kurser)
Web Services (3 kurser)	Programmering (6 kurser)	CMS (5 kurser)
Microsoft Office (29 kurser)	IBM Notes (23 kurser)	Open Office (8 kurser)
Grafiske Kurser (10 kurser)	Powershell (3 kurser)	Drupal (15 kurser)

- ➡ SharePoint, SharePoint Designer, InfoPath
- ➡ .Net programmering, Powershell
- ➡ CMS: Drupal, Magento, WordPress, Joomla
- ➡ IBM Domino (Lotus Notes)
- ➡ XML, XSLT, Web Services, REST
- ➡ Webudvikling: HTML5, Mobile, Php, JavaScript, Angular
- ➡ Microsoft Office, OpenOffice, LibreOffice
- ➡ Grafiske Kurser, GIMP

[Se alle kurser her](#)

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagere på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udriftning af 1 eksemplar udelukkende til eget brug er tilladt.



Kontakt ZoomTek - Gratis Hotline

Efter endt kursus er du altid velkommen til at kontakte os enten med kommentarer eller spørgsmål – sidstnævnte inden for rimelighedens grænser. Læs mere om denne service her: [Læs mere om Gratis Hotline](#). Denne service er inkluderet i kurset. ZoomTek kan kontaktes således:

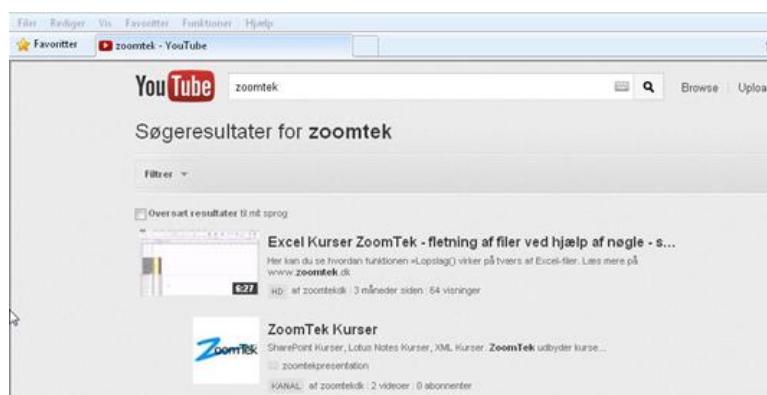
- ▶ E-mail: info@zoomtek.dk
- ▶ Tlf. 4250 5040
- ▶ Web: www.zoomtek.dk
- ▶ C.V.: <http://zoomtek.dk/download/cvdansk.pdf>

Youtube, Blog, OnLine-hjælp – En service for dig !

På et ZoomTek kursus kan vi altid gå ud over pensum. Fører kurset ud over kursusmaterialet, har vi mulighed for at:

YouTube

Lave en video på youtube, som du får et link til. Her kan vi demonstrere et bestemt emne, og hvordan det udføres på computeren.



Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Blog.zoomtek.dk

På **blog.zoomtek.dk** --> 'Kursus' (til højre) vil der efter kurset være links, ressourcer og evt. debat relevant for netop dit kursus. Der oprettes altså et blog indlæg pr. kursus. Du kan som kursist kommentere på dette.

[Se bloggen her](#)

Aalborg Universitet InfoPath SharePoint

Oprettet af [Søren Anderson](#) mand, december 05, 2011 09:18:14

Kursus

InfoPath under forms (genbrugelige form-komponenter)

Opret ny infopath-form. Benyt typen skabelondel - efter du har valgt Filer|Ny. Indsæt kontrolelementer, regler, grafik, m.m. • Gem filen. Den bliver gemt i XTP format.

Når du vil benytte skabelondelen i en anden form går du nederst i opgaveruden til højre under kontrolelementer, og vælger tilføj skabelondel. Herfra kan den efterfølgende trækkes ind på skærmen.

Language-pakker

Her kan man læse om Language Packs. Disse skal installeres før brug, og slæs til fra Central Administration: <http://technet.microsoft.com/en-us/library/cc62108.aspx>

Her kan du downloade Language Packs: <http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=2411>

Usage Reports: <http://mvsharepointark.blogspot.com/2010/06/usage-reports-sharepoint-2010.html>

[http://technet.microsoft.com/en-us/library/ee748626.aspx](#)

Kommentarer(0)

[Share](#)

Tidligere indlæg

Composite Application link

tir, oktober 13, 2011

10:29:58

fra Lotus Notes til Outlook

- kursusmateriale

ons, oktober 12, 2011

10:23:44

LibreOffice

tir, oktober 11, 2011

16:13:25

XML og MySQL kurser i

samarbejde med IBC

... -- -- --

16:09:28

YouTube video om

ZoomTek

tir, oktober 11, 2011

16:02:01

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagere på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Hjælp på din skærm

Vi kan hjælpe dig på din skærm direkte. Ring til os efter kursus. På under 1 minut kan vi overtage din skærm, og direkte vise dig løsningen på din egen skærm, mens vi har dig i telefonen. Det kræver blot, at du har en internforbindelse.

Referencer



[Se Referenceliste her](#)

Kurser for 1 person uden ekstraprism

Som nok det eneste kursuscenter i Danmark tilbyder ZoomTek kurser for 1 person uden ekstraprism.

[Læs mere om 1:1 kurser hos ZoomTek eller via Skype](#)

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

ZoomTekSky

Efter kursus får har du tidsubegrænset adgang til ZoomTekSky, som vi kalder vores Cloud. Du får et link udleveret på kurset. Fra dette link kan du tilgå filer, kursusmaterialer, kursusbeviser m.m.

Hvis den Gratis Hotline giver behov for flere filer, ressourcer, vil disse blive lagt ud her.

Afholdsesgaranti inden 30 dage

Alle kurser afholdes inden for 30 dage. Er der kun 1 tilmeldt afholdes kurset enten på vores adresse ved Juelsminde - eller som et OnLine Kursus.

Tilmeld 2-3 personer og få et lukket firmakursus, hvor I er medbestemmende på pensum og niveau. På lukkede kurser kan vi direkte inddrage jeres cases i kurset.

**ZoomTek garanterer afholdelse
inden 30 dage - ellers er kurset gratis**

også selv om du kun er 1 person

[Læs mere om 30 dages afholdsesgaranti \(herunder betingelser\) her](#)

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagere på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Gratis Hotline efter kurset

Efter kursus har du Gratis Hotline. Du kan altid ringe og skrive til os. Denne service er tidsubegrænset.

Hotlinen er sådan set ubegrænset - men inden for rimelighedens grænser. Du kan ringe/skrive, og vi kan overtage din skærm.

Gratis hotline efter kursus

pr. telefon, email, skærmovertagelse,
cloud-ressourcer

Din underviser sidder ikke altid på telefonen. Derfor kan der godt gå et par dage, før du får svar.

[Læs mere om Gratis Hotline her](#)

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagere på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Indholdsfortegnelse

Kapitel 1 Hvad er Angular?	14
Definition af Angular.....	14
Historik	14
Forskelle på AngularJS og Angular 2/4.....	16
Performance.....	17
Web Components	17
Mobile Devices	18
Brugervenlighed	Fejl! Bogmærke er ikke defineret.
Opsummering	19
Kapitel 2 Opsætning af Angular	20
Download af Node.JS.....	20
Installation og Setup af Node.js	21
CLI – Command Line Interface	24
Installér CLI med denne simple kommando	25
Kapitel 3 Udviklingsmiljø og Editorer	27
Notepad++	29
Visual Studio Code (favoritten)	30
Microsoft Visual Studio	32
Kapitel 4 CLI – Simpel brug	33
Oprettelse og kørsel af projekt.....	34
Oprettelse af filer i dit projekt	38
Kapitel 5 Filer i en applikation	41
tsconfig.json.....	45
typings.json	46
Package.json	46
Index.html.....	47
Må man sætte egen kode ind i Index.html?.....	48
app/main.ts og app.module.ts.....	49
Features.....	51
Overblik over alle filer	53
BootStrap er en del af Angular	54
Konklusion.....	55

Kapitel 1 Hvad er Angular?

Kapitel 6 CSS - Style Sheets	63
src/styles.css	63
Kapitel 7 Angular Arkitektur	56
Komponenter.....	58
Et simpelt komponent	58
1-way binding - interpolation (metode 1)	61
2-way binding med ([ngModel])	62
Kapitel 8 Semantic Versioning (SemVer)	63
Hvad er Semantic Versioning?	65
Time-based Releases	66
Deprecation (Udløb af API)	66
Experimental	66
Public API Surface.....	66
Kapitel 9 TypeScript	68
Hvorfor benytte TypeScript?	71
Datatyper	72
Klasser	75
Nedarvning af klasser	78
TypeScript kode for dummies	79
Interfaces vs. klasser	81
Fat Arrow Funktioner =>	82
Function Expression.....	82
tsconfig.json.....	86
TypeScript Watch-mode	86
Kapitel 10 Nyheder i Angular 4	88
Kapitel 11 Services og Dependency Injection	89
Hvordan ser servicen ud?.....	91
Kapitel 12 HTTP	92
app.component.ts.....	93
Servicen med JSON-kald.....	95
Kapitel 13 Routing	96
Simpel eksempel med Routing.....	97
Opsætning af Router i module.ts	98
Opsætning af Router i app.component.ts	100
Child/nested Router.....	101
Route parameter - dynamisk sti	103

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Kapitel 1 Hvad er Angular?

forChild()-metode 104

Kapitel 14 Navigation Guards 107

Sådan laves en simpel Guard - CanActivate 108
 Opbygning af Guards - Hello World eksempel i detaljer 110
 Implementering af Guards som Service 112
 Deactivating Routes 114

Kapitel 15 Pipes 116

Brug af indbyggede Pipes i Angular 117
 Betingelser i en interpolation 117
 Custom Pipes 117

Kapitel 16 Indbyggede Directives 120

Attribute Directives 121
 Attribute Directive - ngClass 123
 Structural Directives 124
 *Nglf-else 125
 ngNonBindable 127

Kapitel 17 Komponent interaktion 128

Simpel Interaktion mellem parent/child komponent 128
 Overflytning af data - Property Binding 130
 Overflytning af data - Event Binding 132
 Samlet, simpelt eksempel 134

Kapitel 18 Content Projection 136**Kapitel 19 Directives 139**

Opbygning af egne Directives 139
 Decorators 139

Kapitel 20 Hostlisteners og HostBinding 140

HostListener 140
 @HostListener på globale objekter 141
 Info om eventen 141
 @HostBinding 142
 ElementRef 144

Kapitel 21 @ViewChild interaktion 146

Ikke bare til funktionskald 148
 @VieBwChildren 149
 @ContentChild og @ContentChildren 149

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Kapitel 22 Binding	150
Simpel interpolation (metode 1)	150
Betingelse i interpolation.....	150
ngModel og ngForm (metode 2).....	151
[ngModel] (metode 3)	153
([ngModel]) - klassisk 2-way binding (metode 4)	154
Samlet eksempel.....	155
Kapitel 23 Events	156
\$event	157
host.....	159
Kapitel 24 Template Syntaks	160
Property Binding	162
Kapitel 25 Animations	163
Indsæt script-reference i index.html	163
Import.....	164
Kapitel 26 Template-driven forms (FormsModule)	165
Template-driven forms	166
Validering af felter - Valid, Dirty, Touched	169
Validering med Template-driven forms.....	169
Validering pr. Felt	170
Formatering af felter ud fra validering.....	171
Template Model Binding	173
Kapitel 27 Model-driven forms (Reactive)	178
Den Simple Form	179
FormGroup og FormControl	180
Definition af felter med FormControl	181
Opbygning af FormBuilder i constructoren	181
Validering af Controls	182
Import Validators	184
Test af validering	185
Benyt observables på felter	187
Gruppering af felter med <fieldset>.....	187
Den færdige form	188
Kapitel 28 LifeCycle Hooks	191
Import af interfaces.....	192
Kapitel 29 Debugging - forskellige redskaber	194

Kapitel 1 Hvad er Angular?

\$0 - ændre instansen direkte	195
Debugging af TypeScript direkte	197
JSON-pipe	198
Angular Augury	199

Kapitel 30 Unit Testing 201

Unit Test med Angular	201
Jasmine	202
Karma	202
karma.conf.js.....	203
TestBed Konfiguration - src/test.ts	205
ng generate --spec.....	206
Følgende sker når en test køres:	208
Opbygning af en simpel test.....	210
Pending specs.....	211
Virker testmiljøet?	212
Debugging.....	213
Test af komponent	215
Test af funktioner	218
Test af komponent med child-komponent.....	218
Asynkrone tests	221
Unit Testing i Visual Studio	222

Kapitel 31 Index.html 223

Package.json og index.html	224
----------------------------------	-----

Kapitel 32 Polyfills 225

Kapitel 33 Reactive Programmering 226

Overblik	226
Streams	227
Simpelt eksempel med ren html/javascript	228
Hvad er Observables?	231
RxJS i Angular - Hello World.....	232
Subscribe i detaljer	235
Observables og fejlhåndtering.....	236
Unsubscribe()	236
Lidt baggrund: Call Stacks og Event Tables	237
Design Pattern.....	238
Alternativer til Event Loop	238
RxJS tilbyder en robust infrastruktur	240
Observables versus Promises	240
Hvad er Promises? Opsummering	241
Hvad er Observables? Opsummering	241

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Kapitel 34 Stateful og Stateless komponenter	242
Hvad er stateful?.....	242
Hvad er stateless?	243
Kapitel 35 Oprettelse af egne moduler	245
Opret et nyt projekt	245
Eksport af komponentet.....	247
Publisering af modul til github (offentligt modul).....	248
Publisering af modul til nodejs (privat modul).....	251
Kapitel 36 Brug af 3.parts moduler	252
Public API Surface.....	254
Kapitel 37 Build og Publish af projekter	255
Tilpasning af main.ts og environment.ts	255
Webpack Bundle Analyzer	258
Appendix A Kursusoversigt	262
Appendix B Single Page vs MultiPage	263
Appendix C Opsætning af Visual Studio og Node.JS	265
Appendix D NPM Kommandoer	266
samlet liste over npm kommandoer	267
Appendix E Google Developer Tools - DevTools	268
Menuer	268
Elements.....	269
Console.....	270
Appendix F Patterns / Antipatterns	272
Angular og Design Patterns	272
Appendix G Eksterne Programmer	274
Appendix H Node.js	275
Appendix I Opsætning af Angular miljø Uden CLI	276
Download og installér TypeScript	276
Installering af TypeScript	277
Installation af Angular	279
Appendix J Visual Studio og Node.JS	284
Appendix K RxJS operatorer - komplet liste	285

Kapitel 1 Hvad er Angular?

- Definition af Angular
- Angular Historik
- Forskelle på AngularJS 1x og Angular
- Hvad opnår man med Angular 4?



4

Definition af Angular

Angular er et Javascript Framework, som muliggør udvikling af reactive Single Page Applications (SPAs).

Angular tilføjer funktionalitet til HTML. Nogle kalder Angular for HTML++ - eller hvad HTML havde været, hvis det var lavet til webapplikationer.

Navnet Angular kommer fra **Angle** (vinkel) - og refererer til "vinkel-parenteserne", som bruges i html - f.eks. <div>

Historik

- | | |
|-----------------|---|
| September 2014: | Angular 2 først nævnt på ng-Europe konference (alpha versioner) |
| 30. april 2015: | Alpha -> Developer Preview |
| December 2015: | Developer Preview -> Beta |
| Maj 2016: | RC (Release Candidate) |
| September 2016: | Angular 2 Gold |
| Marts 2017: | Angular 4.x |

Fra nu af taler man ikke om Angular 2 eller Angular 4 - men blot Angular.

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Kapitel 1 Hvad er Angular?

Angular 4 kørte i Gold Release i september 2016. Angular 4 er væsentlig forskellig fra AngularJS 1x. Både i brug, måden det kodes på og i de faciliteter, som Angular 4 tilbyder.

Angivelse af versioner i denne bog

I resten af denne bog vil **Angular** betyde den nyeste version. Hvis der undtagelsesvis skal refereres til AngularJS 1.x benævnes det **AngularJS**. I de få tilfælde, hvor der er forskel på Angular 2 og 4, vil versionsnummeret blive brugt.

Har man ikke arbejdet med AngularJS før, bør man glemme alt om denne version. Ellers vil det blot forvirre. Denne bog kræver ingen forkundskaber til Angular.

Angular 3 findes ikke. Man er gået direkte fra Angular 2 til Angular 4 på grund af visse delprogrammers konflikt med versionsnumre.

Bemærk navnene AngularJS og Angular 4. Der er ikke noget, som hedder AngularJS 2 eller AngularJS 4. Ej heller er der noget, der hedder Angular 1.

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Forskelle på AngularJS og Angular 2/4

AngularJS er et framework – Angular 2/4 er en platform

Forskelle på AngularJS og Angular

Scope / Controllers vs. Component / Directives	Native kald af fx. webSocket Mobile Oriented
Brugen af Directives er kraftigt simplificeret... med @directive - decoratoren	AJS 1 er let at sætte op (simpel reference). AJS 2 er afhængig af flere biblioteker og kræver i praksis en nodejs eller iis-server. AJS1 kan installeres fladt på en vilkårlig webserver.
Dependency Injection er udvidet... har flere muligheder .	Bootstrap foregår via kode i AJS2. I AJS brugte man ng-app
TypeScript (bygger på Ecma6)	Structural Directives ændret: ng-repeat er ude. NgFor er inde...
Forms & Validation (FormBuilder og Control Group)	Filters (ajs1) er afløst af Pipes. Syntaksen er den samme
Mere funktionalitet er flyttet ud af Core - derfor hurtigere performance	

Figur 1

Ovenstående har kun relevans, hvis man har udviklet rigtig meget i AngularJS.

Angular er bygget op omkring TypeScript. Nedenfor ses et eksempel på Angular kode. Rent teknisk er koden udført i TypeScript, så Angular 2++ er mere frameworket, hvor TypeScript er programmeringssproget.

```

@Directive({selector: '[hover]',
  properties: ['text: hover']
}

,
hostListeners: {
  hostListeners: {
    'onmouseenter': 'onMouseEnter()',
    'onmouseleave': 'onMouseLeave()'
  }
}
)
  
```

Figur 2

Angular 4 kan med fordel kodes direkte i Visual Studio. Men der kan være mange argumenter for at vælge en anden editor. I denne kursusbog er benyttet notepad++ og i få tilfælde Visual Studio Code (ikke det samme som Microsoft Visual Studio). Men mere om valg af editorer senere.

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Performance

Da AngularJS blev skabt, var det ikke oprindeligt beregnet til udviklere. Det var et redskab målrettet til designere, så de hurtigt kunne opbygge relativ komplekse HTML-formularer.

Over tid blev flere features inkluderet til at bygge flere og mere komplekse applikationer. Man har altså måtte foretage trinvise ændringer i designet for at holde AngularJS i luften, efterhånden som behovene i moderne webapplikationer har ændret sig.

Men der er begrænsninger. En række af disse grænser er omkring performance primært på grund af den eksisterende binding og template opbygning i webapplikationer/browsere.

ECMAScript 7 er introduceret. Moderne browsere støtter allerede mange af disse funktioner. Browsere støtter nu moduler, klasser, interfaces, lambdas, generatorer m.m. Dette vil ændre brugen af JavaScript.

[6th Edition - ECMAScript 2015](#)

[7th Edition - ECMAScript 2016](#)

[8th Edition - ECMAScript 2017](#)

Versioner og kompatibilitet ændrer sig løbende. TypeScript og dens versioner er afhængig af ECMAScript.

Web Components

Web Components er introduceret. Udtrykket Web Components dækker over 4 W3C specifikationer:

- ➡ **Custom Elements** - Giver udvidelse af HTML gennem brugerdefinerede tags.
- ➡ **HTML Import** - Giver import af forskellige ressourcer (HTML, CSS, JS, etc.).
- ➡ **Template Element** - Muliggør såkaldt inaktiv HTML i et dokument.
- ➡ **Shadow DOM** - Aktiverer indkapsling af DOM og CSS.

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Kapitel 1 Hvad er Angular?

Ved at kombinere disse fire områder kan udviklere skabe brugerdefinerede elementer, som er fuldt indkapslet (Shadow DOM).

Komponenterne kan håndtere egne metadata og kan let eksporteres til anden kode (HTML import).

Nogle af disse specifikationer er allerede introduceret i f.eks. Chrome.

Den måde AngularJS benytter databinding er dog ikke kompatibel med disse nye specifikationer. Det er en af de væsentligste grund til, at man har måtte omskrive hele databinding-funktionaliteten i Angular.

Mobile Devices

Angular er i højere grad rettet mod mobile devices. Selv om det egentlig ikke var intentionen fra starten. Men opbygningen med komponenter giver bedre ydeevne på mobile devices.

Kompleksitet i udvikling

AngularJS er ikke nemt at udvikle i (sammenlignet med Angular). I starten virker alt nemt. Men skal man sætte mere avancerede ting i værk, kommer man hurtigt til kort, med mindre man er en særdeles rutineret JavaScript-udvikler.

Angular 4 er anderledes. Her kan man lave rigtig meget i ren HTML støttet af directives (attributter) m.m., og alt sammen akkompagneret af TypeScript.

TypeScript er også JavaScript, men et subscript. TypeScript tillader en masse ting (klasser, constructors, safe datatypes), som ikke er mulige i standard JavaScript - med mindre man programmerer helt ud af en tråd.

Angular 4 sammen med TypeScript tilbyder en enkelthed, der er uovertruffen.

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udriftning af 1 eksemplar udelukkende til eget brug er tilladt.



Opsummering

- ➡ Angular er mere simpel end AngularJS. Angular er simpelt hen nemmere at forstå.
- ➡ Angular kører altid via en webserver. Det forbedrer performance, da dataload beregnes serverside.
- ➡ Angular understøtter nyeste browsere.
- ➡ Den virker på tværs af platforme og frameworks.
- ➡ Angular er skabt til mobile devices.
- ➡ Selv om der loades mange data, virker alle events og UI Response.
- ➡ Fungerer sammen med Dart, TypeScript og andre sprog, der kompilerer til JavaScript.
- ➡ Dependency Injection er blevet nemt.
- ➡ Alt er baseret på komponenter.

Angular 4 - Udvikling af Angular apps

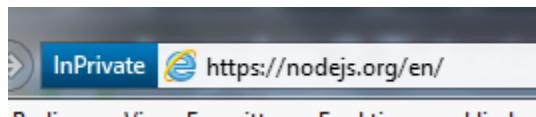
©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Kapitel 2 Opsætning af Angular

Angular bygger på et udviklingsmiljø bestående af Node.js og i anden række CLI. CLI sørger automatisk for installation af andre ting – herunder TypeScript.

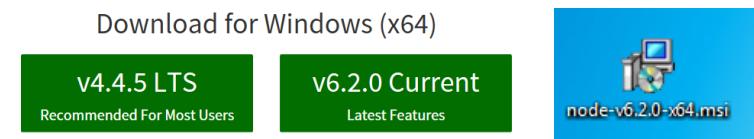
Det er muligt at køre Angular kun med Node.js og uden CLI, men så skal man selv downloade og installere bl.a. TypeScript, som CLI ellers sørger for.

Download af Node.JS



Figur 3

Node.js downloades fra <https://nodejs.org/en/>. Download er lige ud ad landevejen.



Figur 4

'Current' version downloades. Og MSI-filen installeres.

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Opgradering af eksisterende Node.js installation

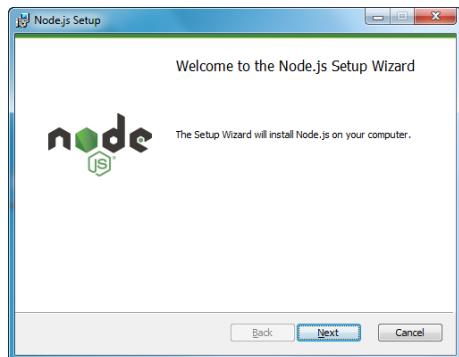
Har man allerede Node.js installeret, kan man opgradere til nyeste version således:

- ```
1 npm install @angular/common@latest @angular/compiler@latest
 @angular/compiler-cli@latest @angular/core@latest @angular/forms@latest
 @angular/http@latest @angular/platform-browser@latest
 @angular/platform-browser-dynamic@latest @angular/platform-
 server@latest @angular/router@latest @angular/animations@latest
 typescript@latest --save
```

Figur 5

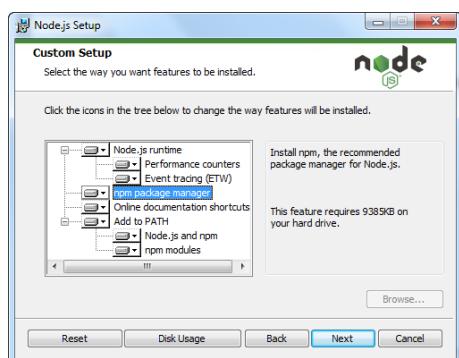
## Installation og Setup af Node.js

Opsætning af Node.js er nemt. Følg wizarden.



Figur 6

Tilvælg Node.js and npm samt npm modules.

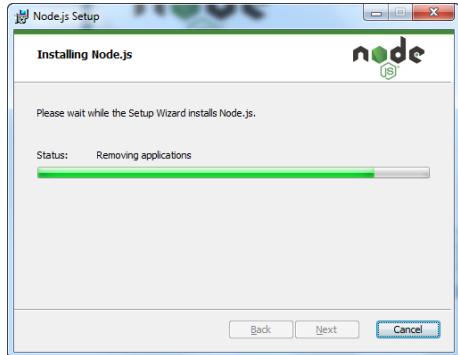


Figur 7

## Angular 4 - Udvikling af Angular apps

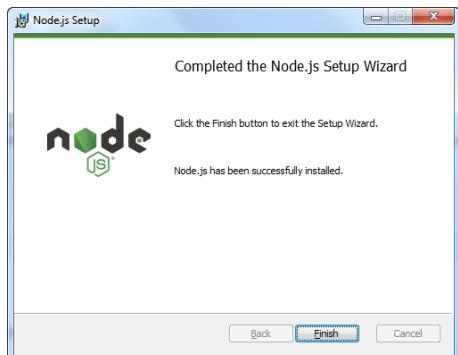
©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltager på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 2 Opsætning af Angular



Figur 8

Efter nogle minutter er Node.js installeret. Undertegnet har aldrig oplevet komplikationer med installationen. Node.js kører på både mac og windows. NPM installeres sammen med Node.js.



Figur 9

Med en henholdsvis **node -v** og en **npm-v** kommando kan man tjekke versionen af npm. Den bør være mindst som på nedenstående billede.

Node.js kører i en dos-prompt. Højreklik på title-bjælken for at ændre farverne i dos-prompten. Gør skriftypen større og forøg bredde og højde på konsollen.

Konsollen (TypeScript kompileren) kaster mange fejlmeddelelser af sig, når man arbejder med Angular. Derfor er det ærgerligt at arbejde med en lille, sort konsol med røde fejlbeskeder.

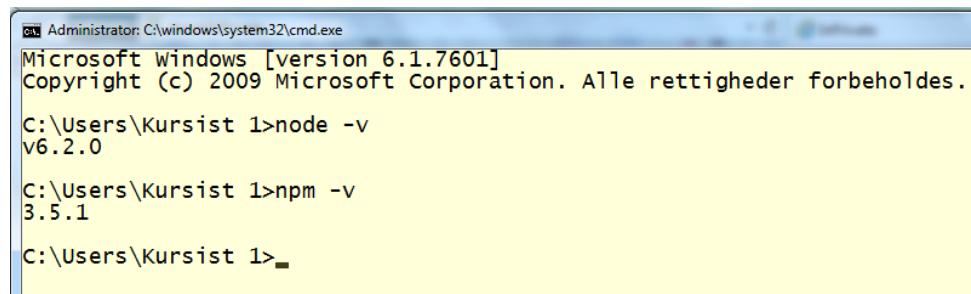
Små detaljer, der gør arbejdet med Node.js nemmere.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 2 Opsætning af Angular



```
C:\Administrator C:\windows\system32\cmd.exe
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Alle rettigheder forbeholdes.

C:\Users\Kursist 1>node -v
v6.2.0

C:\Users\Kursist 1>npm -v
3.5.1

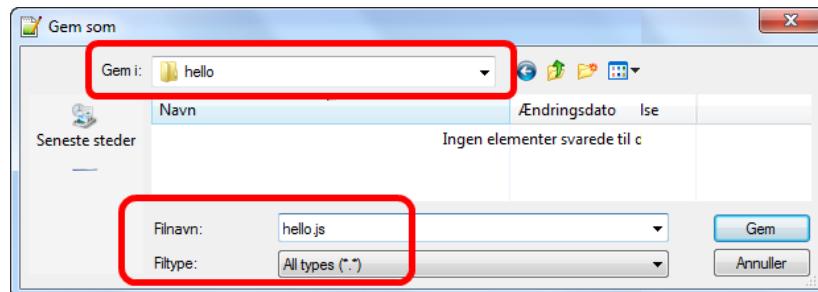
C:\Users\Kursist 1>
```

Figur 10 Tjek versionen af Node.js og npm med **-v**

```
1 console.log('Node is installed!');
2 // gem filen i selvvalgt mappe...
3 // evt. undermappe under cmd-standardmappe
4
```

Figur 11 Test om din installation er korrekt

Lav en JavaScript-fil. Gem den som hello.js i et bibliotek under CMD-roden (her hello).



Figur 12

```
C:\Users\Kursist 1\hello>node hello.js
Node is installed!
```

```
C:\Users\Kursist 1\hello>
```

Figur 13

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## CLI – Command Line Interface

CLI er et add-on til Angular, som man kan vælge at benytte. **Man kan sagtens arbejde med Angular uden CLI** – men CLI giver en række genveje. Derfor anbefales det at benytte CLI.

For at benytte CLI skal man have en opdateret Node.js. Kører man f.eks. en 6.2.0. version, vil CLI brokke sig.

```
[WARN engine] angular/cli@1.2.0: wanted: {"node":">>= 6.9.0", "npm": ">= 3.0.0"} (current: {"node": "6.2.0", "np
```

Figur 14

Man tjekker versionen således med **node -v**.

```
C:\Users\Kursist 1>node -v
v6.2.0
```

Figur 15

På nettet kan man finde mange kommandoer til opgradering. Det nemmeste er at gå direkte på <https://nodejs.org/en/> og downloade og køre seneste version.

Man behøver end ikke at genstarte sin computer.

```
C:\Users\Kursist 1>node -v
v6.2.0

C:\Users\Kursist 1>node -v
v6.11.0
```

Figur 16

**CLI** er god til store projekter. Selv gigantiske projekter med 100 vis af filer kan CLI nemt styre. Med CLI kan man fokusere på koden og komponenterne.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Installér CLI med denne simple kommando

```
npm install -g @angular/cli
```

```
>npm install -g @angular/cli
```

Figur 17

### White Screen of Death

Får man en hvid skærm, når man kører et projekt med **ng serve**, bør man prøve at opdatere CLI med nedenstående kommando:

```
npm install --save-dev @angular/cli@latest
```

White Screen of Death kan opstå, når man f.eks. benytter Internet Explorer 11 som standardbrowser. Copy/Paste url'en til Google Crome eller Edge – så burde White Screen of Death forsvinde.

Internet Explorer 11 og andre "gamle browsere" kan selvfølgelig benyttes, men så skal der indsættes Polyfills (se kapitlet herom) i index.html-filen.

CLI giver en række muligheder med simple kommandoer. Få hjælp til CLI online med denne kommando:

```
ng help
```

Figur 18

```
--poll (Number) Enable and define the file watching poll time period (mill
 aliases: -poll <value>
--app (String) Specifies app name to use.
 aliases: -a <value>, -app <value>

ng version <options...>
 Outputs Angular CLI version.
 aliases: v, --version, -v
 --verbose (Boolean) (Default: false) Adds more details to output logging.
 aliases: --verbose

ng xi18n <options...>
 Extracts i18n messages from source code.
 --i18n-format (String) (Default: xlf) Output format for the generated file
 aliases: -f <value>, -xmb (--i18n-format=xmb), -xlf (--i18n-format=xlf),
 --output-path (Path) (Default: null) Path where output will be placed.
 aliases: -op <value>, --outputPath <value>
 --verbose (Boolean) (Default: false) Adds more details to output logging.
 aliases: --verbose
 --progress (Boolean) (Default: true) Log progress to the console while runn
 aliases: --progress
 --app (String) Specifies app name to use.
 aliases: -a <value>, -app <value>
 --locale (String) Specifies the source language of the application.
 aliases: -l <value>, --locale <value>
 --out-file (String) Name of the file to output.
 aliases: -of <value>, --outFile <value>
```

Figur 19

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

### Hjælp i udklipsholderen

Skriv I stedet **ng help | clip**. Hjælpen overføres i stedet til udklipsholderen – og kan pastes ind i notepad e.l.

Man kan læse dokumentation og downloade CLI direkte på github på dette link:

<https://github.com/angular/angular-cli>

```
C:\Users\Kursist_6\ajs\introduktion\NytPakke>ng serve --open
Your global Angular CLI version (1.2.3) is greater than your local
version (1.2.2). The local Angular CLI version is used.
```

CLI (lige som alt andet i Node.js) kan installeres både lokalt og globalt – også på samme tid. Her ses en besked, hvor der netop er installeret CLI både lokalt og globalt. Den lokale version benyttes altid.

Brug npm uninstall til at afinstallere.

Bruges flaget -g installeres/afinstalleres der globalt – ellers lokalt.

Med kommandoen **ng version** kan man teste versionen af CLI

```
C:\Users\Kursist_6\ajs>ng version
```

```
Angular CLI: 1.2.3
Node: 6.10.0
OS: win32 x64
```

Figur 20

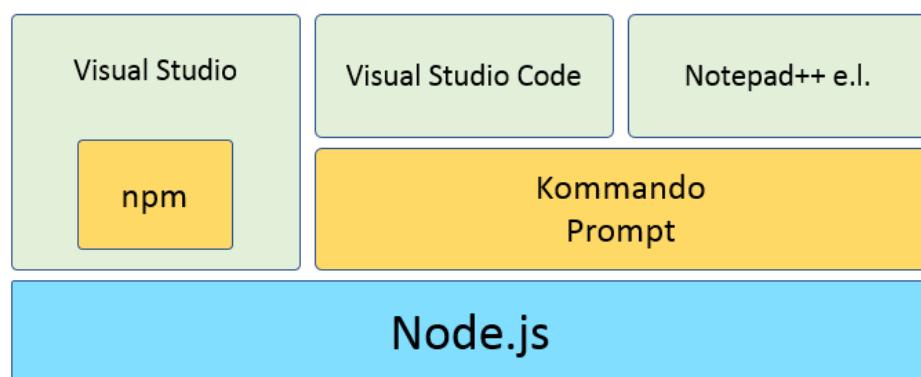
### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 3 Udviklingsmiljø og Editorer

Man kan køre Angular på flere måder. Man har behov for en web server. En sådan web server vil ofte være Node.js. Node.js kan betjenes via en konsol.



Figur 21

Først skal Node.js installeres. Node.js er en web server, der betjenes via npm kommandosproget. **Npm står for Node Package Manager**, hvilket fortæller ret godt, hvad Node.js og npm i virkeligheden tilbyder.

Nemlig en nem både for udviklere at dele, downloade og vedligeholde JavaScript-baseret kode (og andet). Npm installeres sammen med Node.js.

Man kan vælge mellem forskellige udviklingsværktøjer. Har man selv en editor, f.eks. Notepad++, er det helt fint. Al kode skrives i editoren. Og alle npm-kommandoer skrives i konsollen. Konsollen er blot en dos-prompt, som alt andet lige startes med **Start | cmd**.

Valg af udviklingsmiljø handler i sidste ende om smag og behag, licenskroner og vaner. Der kan være nogle redskaber, som kun findes i en given editor. F.eks. har NotePad++ ikke IntelliSense til TypeScript - til gengæld virker den lettere at arbejde med end Microsoft Visual Studio. Koden der kommer ud i den anden ende er identisk.

### Angular 4 - Udvikling af Angular apps

## Kapitel 3 Udviklingsmiljø og Editorer

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

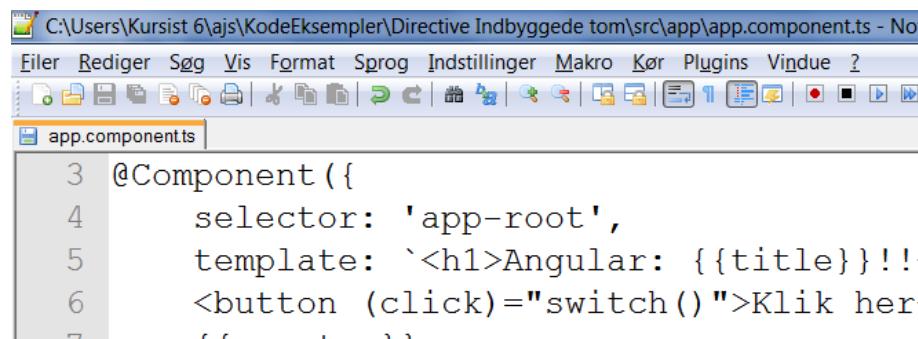


## Notepad++

I dette materiale er Notepad++ primært benyttet. Den er nem at arbejde med og kræver ikke specielle forkundskaber.

Nodepad++ er en god editor, når man starter på et nyt udviklingssprog. Man får lidt hjælp, men ikke nær så meget som i de øvrige editorer.

Men Notepad++ er gratis, starter lynhurtigt og faktisk mange indstillinger til f.eks. IntelliSense på JavaScript og sågar C#.net.



The screenshot shows the Notepad++ interface with the file 'app.component.ts' open. The code displays an Angular component definition:

```
3 @Component({
4 selector: 'app-root',
5 template: `<h1>Angular: {{title}}!!</h1>
6 <button (click)="switch()">Klik her
7 </button>
8 }`
```

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Visual Studio Code (forfatters favorit)

**Visual Studio Code** er ikke det samme som Microsoft Visual Studio. Dette program er fantastisk at arbejde med. Det har automatisk IntelliSense til TypeScript - altså får man hjælp til at skrive kode i Angular.

Den har alt det gode fra Microsoft Visual Studio, men er lettere at arbejde med - og editoren starter og reagerer hurtigere.

Man kan indstille Visual Studio Code, så den f.eks. har mørk baggrund og lys tekst. Programmet kan downloades gratis - er licensfrit - og har ikke noget med Visual Studio at gøre.

The screenshot shows a code editor window for a file named 'child.component.ts'. The code is as follows:

```

1 import { Component, Host } from '@angular/core';
2
3 @Host
4 Type of the Host metadata. Host decorator and metadata.
5 @HostBinding
6 @HostDecorator
7 <ng-content select="[art=b]"></ng-content>
8 <ng-content select="[art=c]"></ng-content>
9
10 }
11 export class ChildComponent {
12
13 }

```

A tooltip is displayed over the word 'Host' in line 2, providing the documentation: 'Type of the Host metadata. Host decorator and metadata.' Below the tooltip, there are three more options: 'HostBinding', 'HostDecorator', and 'HostListener'.

Figur 22

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Alternative Editorer

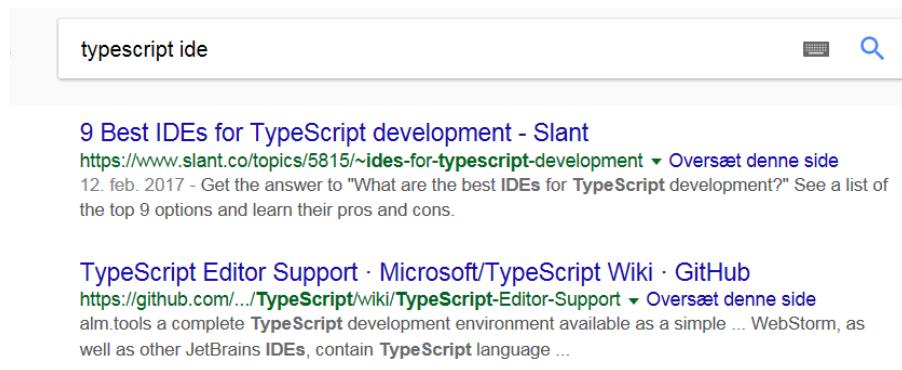
Endvidere kan man overveje:

- ➡ **WebStorm**
- ➡ **Atom**
- ➡ **Sublime Text**

Alle 3 har IntelliSense til TypeScript, men begge koster licens.  
Overvej også:

- ➡ NetBeans
- ➡ Vim
- ➡ Emacs
- ➡ AIM
- ➡ Eclipse

På Google finder man dokumentation, download m.m. ved at søge på 'TypeScript IDE'.



Figur 23

Dette kursus viser brugen af Nodepad++ eller Visual Code Editor.

### Angular 4 - Udvikling af Angular apps

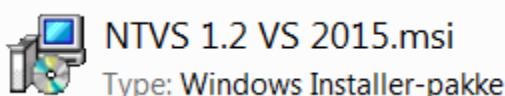
©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Microsoft Visual Studio

Benytter man i forvejen Visual Studio, kan programmet med fordel benyttes. Her er en række templates til ASP.net og en masse andre ting. Visual Studio er den mest omfattende editor, men den er også svære at lære - og koster ofte licens. Om den altid er bedst til Angular kan diskuteres.

Microsoft Visual Studio har indbygget editor og npm-kommandoer.

Man kan dog downloade Node Tools for Visual Studio, som giver ekstra muligheder.



Figur 24

```

Administrator: C:\windows\system32\cmd.exe
30-06-2016 10:36 24.777.849 FormsRunningDemo.zip
22-10-2016 19:18 <DIR> HelloWorld
06-06-2016 22:23 <DIR> http
08-06-2016 23:02 <DIR> input
16-01-2017 09:37 <DIR> Introduktion
13-01-2017 21:29 <DIR> introduktion2
13-01-2017 21:31 1.056.310 introduktion2.zip
16-01-2017 10:06 <DIR> introduktion3
06-06-2016 08:19 <DIR> Zhero
22-10-2016 20:00 <DIR> ZPlaneter
22-10-2016 20:02 <DIR> ZServiceDI
08-06-2016 21:33 <DIR> zTypeScript
 4 fil(er) 25.835.810 byte
 29 mappe(r) 209.105.342.464 byte ledig
C:\Users\Kursist 1\AJS>

```

Figur 25

Node.js kan downloades og installeres relativt simpelt (se appendix). Man tilgår Node.js direkte gennem CMD. Node.js benytter sig af npm-kommandoer til at stoppe, starte, installere og så videre. Selve koden laves i en simpel editor som f.eks. Notepad++.

Alternativet er at køre Node.js sammen med Visual Studio. I stedet for at skrive npm-kommandoer direkte, kan disse vælges/eksekveres fra menuer/ikoner/højreklik. Visual Studio tilbyder også en række templates, som forenkler arbejdet med Angular.

Man har mulighed for at installere Node.js og Visual Studio og køre disse samlet. Dermed kan man udvikle Angular løsninger direkte i Visual Studio.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 4 CLI – Simpel brug

CLI står for **Command Line Interface**

CLI hjælper med at facilitere Angular udvikling. Har man styr på CLI, kan man fokusere på ren kodning. Uden et **tooling** redskab som CLI, skal man manuelt lave en masse operationer.

CLI hjælper med:

- ➡ Oprettelse af projekter
- ➡ Oprettelse af komponenter, directives m.m.
- ➡ Automatisere kode
  
- ➡ Eliminering af trivielle fejl
- ➡ Kørsel af en server med LiveReload support, så applikationer kan ses direkte under udvikling
  
- ➡ Kørsel af end-to-end test (E2E)
- ➡ Flytte applikationen fra Udvikling til Produktion

Man kan finde den officielle CLI-dokumentation på dette link:

<https://github.com/angular/angular-cli/wiki>

## Oprettelse og kørsel af projekt

### `ng new`

Benyt `ng new mit_projekt_navn` til at oprette et nyt projekt<sup>1</sup>. Det tager nogle minutter at oprette et projekt. Følgende sker, når man benytter `ng new`:

- ➡ En ny mappe (directory) oprettes
- ➡ Alle filer nødvendige for ens Angular applikation oprettes
- ➡ Npm dependencies installeres
- ➡ Typescript opsættes
- ➡ Protractor end-to-end test framework sættes automatisk op
- ➡ Karma Unit Test Runner sættes automatisk op
- ➡ Såkaldte Environment filer (se senere) sættes op med fair standardværdier

Skriv `ng --help` (2 bindestreger) for at få hjælp.

Ønskes specifik hjælp til `ng new` skrives: (dobbelt bindestreg foran help).

`ng new --help` eller `ng new --help |clip` (sidstnævnte kopierer til udklipsholder)

Her fremgår det, at `ng new minapp -si true` opretter projekt, som ikke installeres.

**ng står for Angular.** Når ng udtales på engelsk lyder det lidt som Angular.

---

<sup>1</sup> En installation downloader mindst 150MB data i form af moduler og andre filer. Selv det mindste projekt med en lille SPA-side skal altså slæbe rundt på 150MB data. Det er fordi, der køres i udvikler mode. Når projektet er færdigt kan det **buildes**. Så bliver siden reduceret til nærmest ingenting (afhængig af features). Når projektet er buildet kan det afvikles på en driftserver.

### Angular 4 - Udvikling af Angular apps



**Om parametre/argumenter**

Parametre til ng kommandoer kan enten skrives helt ud eller forkortet. F.eks. kan man skrive både:

```
ng new MitFedeProjekt --skip-install true
```

og

```
ng new MitFedeProjekt --si
```

I 'litteraturen' vises parametre oftest med dobbelt bindestreg ' - - '. Dog kan man sagtens benytte enkelt bindestreg – uanset om man skriver parameternavnet forkortet eller helt ud.

Omvendt forholder det sig til ng -help. Her skal der benyttes dobbelt bindestreg.

Efter nogle minutter vises denne besked:

```
Installing packages for tooling via npm.
Installed packages for tooling via npm.
Successfully initialized git.
Project 'A4test' successfully created.
```

Figur 26

**ng server** og **ng init** fungerer næsten ens. Ng server opretter en ny mappe og opretter et projekt i denne mappe.

Ng init opretter et projekt i den mappe, man står i. Med ng init skal man altså manuelt oprette mappen i forvejen.

Ng new fungerer næsten som npm install. Npm install er en 'ren' npm kommando, som benyttes, hvis ikke CLI er installeret. CLI er jo en overbygning på Node.js.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## ng serve

**ng serve** starter app'en. Som default startes på <http://localhost:4200>. Skriver man blot **ng server** uden parametre, skal man selv åbne browseren.



ng serve kan indeholde parametre f.eks:

ng serve --host 1.2.3.4 --port 33000 --open

ved at bruge **--open** åbnes der automatisk et browservindue. Ønsker man at starte med visse parametre, kan dette konfigureres i package.json:

```

5 "license": "MIT",
6 "scripts": {
7 "ng": "ng",
8 "start": "ng serve --open",
9 "build": "ng build",
10 "test": "ng test",
11 "lint": "ng lint".

```

Figur 27

Herefter kan man blot skrive:

**npm start**

Figur 28

Npm start er en 'ren' npm kommando. CLI er jo en overbygning, som i sidste ende eksekverer npm kommandoer på Node.js.

## Hvad sker der præcist?

→ Angular CLI henter sin opsætning fra .angular-cli.json

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



- ➡ Angular CLI kører Webpack for at *builde og bundle* al JavaScript og CSS kode
- ➡ Angular CLI starter Webpack Dev Server, så websiden kan vises på localhost:4200.

**CLI benytter sig af LiveReload.** Det er en teknik, så alle filændringer automatisk vises på websiden uden refresh. Man siger også, at **TypeScript er i watchmode**.

```
chunk { } polyfills.bundle.js
chunk { } main.bundle.js, ma
chunk { } styles.bundle.js, s
chunk { } vendor.bundle.js, v
chunk { } inline.bundle.js, i
webpack: Compiled successfully.
```

Figur 29

Bemærk, at konsollen 'hænger'. LiveReload er aktiv. Sessionen lukkes med CTRL-C.

```
^CAFbryd batchjob (J/N)? j_
```

Figur 30

Skal man lave andre operationer, mens konsollen 'hænger', kan en ny cmd-prompt startes op parallelt.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Oprettelse af filer i dit projekt

Alle projekter består af de samme byggesten. Med **ng generate** kan man oprette:

|           |        |           |
|-----------|--------|-----------|
| component | enum   | service   |
| class     | module | interface |
| directive | pipe   | guard     |

I praksis gøres det således:

- ng generate class Test-class
- ng generate component Test-component
- ng generate directive Test-directive
- ng generate enum Test-enum
- ng generate module Test-module
- ng generate pipe Test-pipe
- ng generate service Test-service
- ng generate interface Test-interface
- ng generate guard Test-guard

Kommandoen skal udføres i projektmappen: (find evt. hjælp med **ng generate --help**)

```
C:\Users\Kursist 6\ajs\introduktion\NyPakke>ng generate component ChildComponent
installing component
 create src\app\child-component\child-component.component.css
 create src\app\child-component\child-component.component.html
 create src\app\child-component\child-component.component.spec.ts
 create src\app\child-component\child-component.component.ts
 update src\app\app.module.ts
```

Figur 31

Komponentet **ChildComponent** er herefter oprettet. Senere i materialet arbejdes der med disse filer i detaljer. Disse filer er de legoklodser, som alle Angular projekter består af. **Uden CLI og ng generate skulle filerne oprettes manuelt** (alternativ kan andre pakker/boilerplates benyttes).

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Man kan forkorte kommandoen. Ovennævnte kommando kan også skrives således:

`ng g c ChildComponent`

man bør faktisk ikke kalde komponentet `ChildComponent` men blot `Component`. Ellers risikerer man at komponent-klassen hedder `ChildComponentComponent`. Man kan manuelt rette fejlen efterfølgende.

Alle filer oprettes efter konventionerne udstukket af Angular-teamet. F.eks vil ovenstående kommando oprette `ChildComponent` uanset casen i kommandoen. Uanset om man skriver `childcomponent` eller `CHILDCOMPONENT`.

Når et komponent oprettes, vil der under `src/app` oprettes en mappe med navnet `child-component` (bindestreg og lowercase sker automatisk).

|                                                                                                                       |                        |
|-----------------------------------------------------------------------------------------------------------------------|------------------------|
|  child-component.component.css     | CSS Document           |
|  child-component.component.html    | Chrome HTML Document   |
|  child-component.component.spec.ts | TypeScript Source File |
|  child-component.component.ts      | TypeScript Source File |

Figur 32

De viste filer oprettes. Selve komponentet ligger i den nederste `child-component.component.ts` fil.

Udover oprettelse af disse filer tilføjes komponentet i `src/app/app.module.ts`.

```
1 import { AppComponent } from './app.component';
2 import { ChildComponentComponent } from
3 './child-component/child-component.component';
4
5 @NgModule({
6 declarations: [
7 AppComponent,
8 ChildComponentComponent
9],
10 })
11
```

Figur 33

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 4 CLI – Simpel brug

Mere om Module-filen i et andet kapitel. Men kort sagt, når man benytter **ng generate component**, kan man fokusere på at udvikle komponentet og projektet.

#### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 5 Filer i en applikation

En **applikation**<sup>2</sup> består af en række filer. I praksis vil man have en skabelon, som på den ene eller anden måde kopieres rundt ved opstart af en ny applikation.

Applikationer indeholder en række filer, som fra gang til gang vil være ens. Disse filer vil man ikke oprette fra scratch, men låne fra en eksisterende applikation, fra en skabelon, fra github eller med en fiks CLI/npm kommando.

Angular applikationer er opbygget af features, hvoraf den vigtigste type er komponenter (Components). En given applikation kan bestå af mange komponenter. Et komponent er en TypeScript klasse.<sup>3</sup>

En applikation er sådan set bare en mappe. Den indeholder dels de brugte komponenter (som man udvikler i NotePad++ e.l.). Derudover ligger der faste filer, som benyttes til at styre modulet (TypeScript, dependencies, indlæsning m.m.).

Alle applikationer består grundlæggende af følgende filer. Bemærk, at denne liste kan afvige afhængig af Node.js-installation, programmering i de enkelte komponenter, opsætning af TypeScript m.m.

---

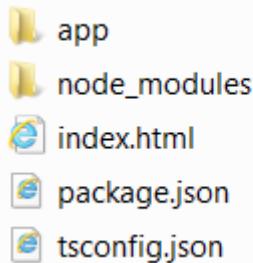
<sup>2</sup> Begreberne applikation og projekt benyttes ofte i flæng - og er egentlig også det samme. Applikation er set med brugerens øjne. Altså hvordan ser websiden ud, når den er færdig. Projekt er set med udviklerens øjne og handler blandet andet om, hvilke filer, der indgår. I dette materiale benyttes de 2 begreber i flæng og som synonymer.

<sup>3</sup> En feature er lidt et uofficielt begreb. Man kan ikke kalde dem komponenter - for et komponent er blot én type. Nogle kalder disse features for klasser. Problemet med dette er blot, at de ikke alle er klasser - f.eks. interfaces.

### Angular 4 - Udvikling af Angular apps

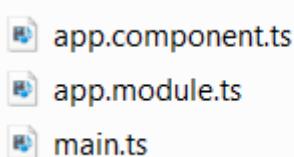
## Kapitel 5 Filer i en applikation

En typisk applikation er opbygget således:



Figur 34

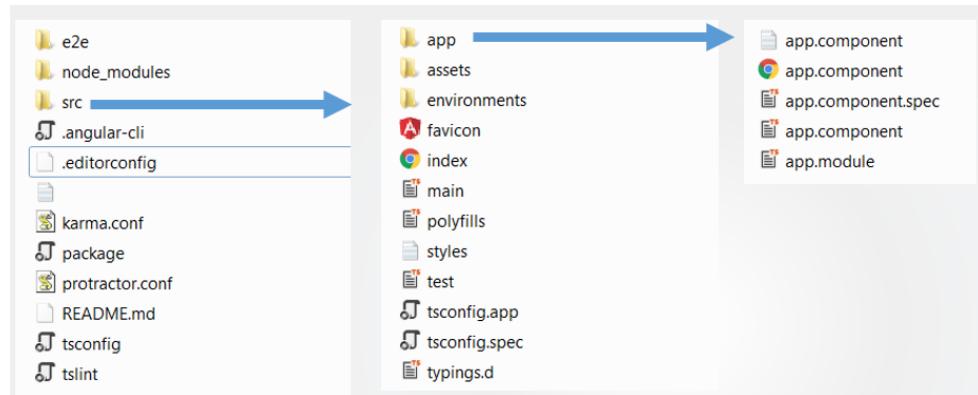
**app-mappen** indeholder alle komponenter, som man skal kode. App-mappen kan typisk se således ud:



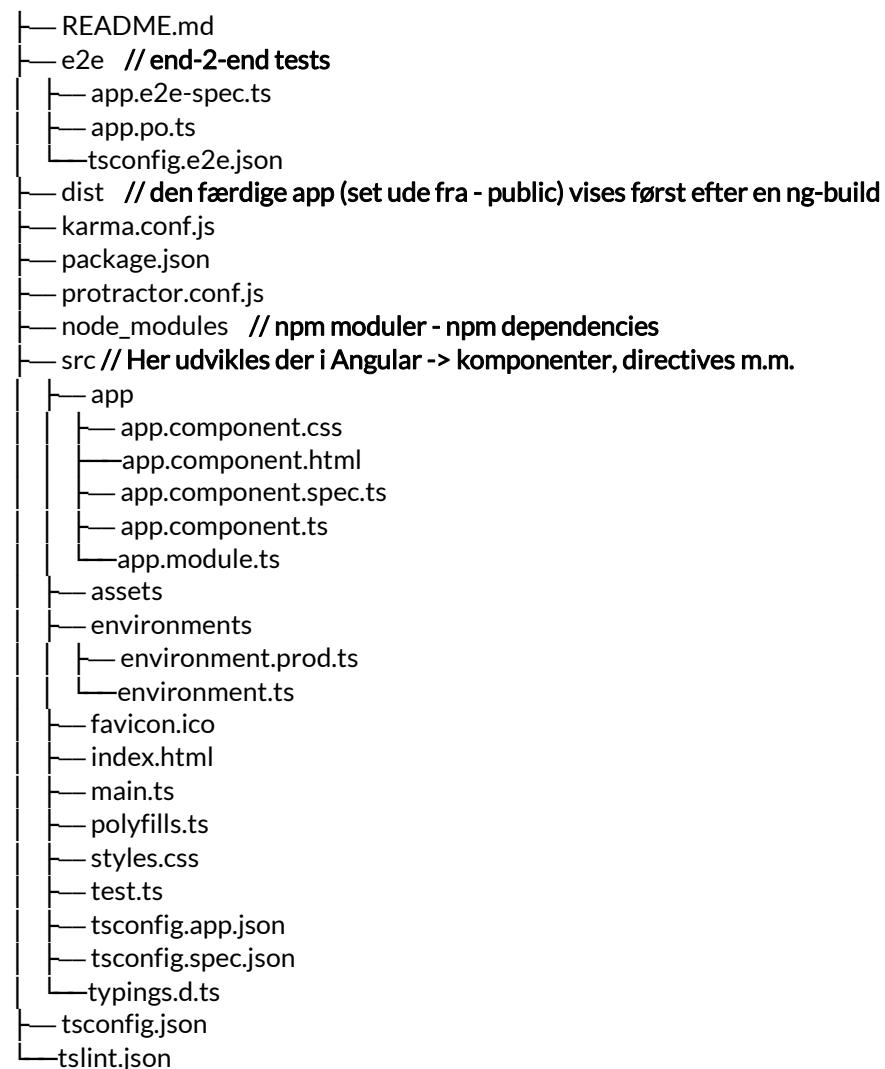
Figur 35

## Filer med CLI

Når man installerer med CLI (ng new) oprettes følgende filer (vises både grafisk – og i 'copy/paste' format:



Figur 36



### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 5 Filer i en applikation

Figur 37

Her er et komponent (app.component.ts). Der er ret stramme konventioner om navngivning af filer.

Hvilke filer der præcist skabes, når man opretter et projekt kan variere. F.eks. er der forskel på Angular 2 og Angular 4. Der er også forskel på Angular med og uden CLI.

### Navnekonventioner

Et modul kan som sagt bestå af rigtig mange komponenter og andre typer filer, som ikke er listet her. Derfor er navnekonventioner vigtige. Alle komponenter hedder:

*navn\_på\_komponent.component.ts*.

**ts** er TypeScript. Komponenter er altså udviklet i TypeScript. Nedenfor ses de samme filer igen med en lille forklaring.

#### Filer (root | app)

- **tsconfig.json** (compiler-info)
- **package.json** (dependencies)
- **typings.json** (typescript definition files)
- **systemjs.config.js** (configure SystemJS)
- **index.html**
- **main.ts**
- **angular-2-hello-world.component.ts**
- **angular-2-hello-world-app.component.ts**

Figur 38

Alle filer, der hedder noget med *spec* benyttes til fejlsøgning og såkaldt mocking. Her er det ikke bare hensigtsmæssigt, at filen indeholder *spec*. Det er direkte nødvendigt, fordi systemet til fejlfinding kigger efter *spec*-filer.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## tsconfig.json

```
1 {
2 "compilerOptions": {
3 "target": "es5",
4 "module": "commonjs",
5 "moduleResolution": "node",
6 "sourceMap": true,
7 "emitDecoratorMetadata": true,
8 "experimentalDecorators": true,
9 "removeComments": false,
10 "noImplicitAny": true,
11 "suppressImplicitAnyIndexErrors": true,
12 "typeRoots": [
13 "./node_modules/@types/"
14]
15 },
16 "compileOnSave": true,
17 "exclude": [
18 "node_modules/*",
19 "**/*-aot.ts"
20]
}
```

Figur 39

tsconfig.json er TypeScript Compiler Konfigurationsfilen.

Filen fortæller compileren, hvordan den skal kompilere TypeScript til JavaScript.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## typings.json

Ofte findes også en typings.json fil (ikke altid). Den benyttes til ekstern funktionalitet, som ikke umiddelbart kan fortolkes af TypeScript-fortolkeren.

```
{
 "globalDependencies": {
 "core-js": "registry:dt/core-js#0.0.0+20160612141332",
 "node": "registry:dt/node#6.0.0+20160621241320",
 "jasmine": "registry:dt/jasmine#2.2.0+20160621624255"
 }
}
```

Figur 40

Her tilføjes Jasmine (fejlretning), core-js (ES6) samt Node, som bruges til at referere til objekter i Node.js.

## Package.json

Når man opretter et modul skal det installeres med npm install kommandoen. Package.json sørger for at downloade og installere alle dependencies (afhængigheder).

### package.json

- dependencies
- start-up scripts:
- Linje 9: TypeScript Compiler (w-flag betyder watch)
- Linje 5: Start task starter liteweight http-server med 'npm run lite'
- Fuld dokumentation: <https://docs.npmjs.com/files/package.json>

```

4 "scripts": {
5 "start": "tsc && concurrently \"npm run tsc:w\" \"npm run lite\" ",
6 "lite": "lite-server",
7 "postinstall": "typings install",
8 "tsc": "tsc",
9 "tsc:w": "tsc -w",
10 "typings": "typings"

```

Figur 41

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Index.html

Index.html er den side, der reelt vises på skærmen. Det er en almindelig html5-side. I headeren læses en række js-filer ind til forskellige formål.

Nederst i <body>-taggen findes følgende kode:

```
</head>
<body>
 <app-root></app-root>
</body>
```

Figur 42

Index.html kan variere fra package til package. Hovedkompositionen er den samme. Ofte - men ikke altid - hedder rodkomponentet my-app<sup>4</sup>. Det skal man naturligvis være opmærksom på, inden man kaster diverse app.component.ts-filer ind udefra.

Det er her rodkomponentet læses ind.

### index.html

- systemjs booter alle moduler - linje 8
- Polyfills sikrer bagud kompatibilitet med tidlige IE-versioner (helt præcist hvis browseren er afhængig af "ES2015 promises and dynamic module loading")
- RxJS = Reactive Extensions

```
1 <script src="node_modules/core-js/client/shim.min.js"></script>
2 <script src="node_modules/zone.js/dist/zone.js"></script>
3 <script src="node_modules/reflect-metadata/Reflect.js"></script>
4 <script src="node_modules/systemjs/dist/system.src.js"></script>
5 <!-- 2. Configure SystemJS -->
6 <script src="systemjs.config.js"></script>
7 <script src="node_modules/angular2/bundles/angular2-polyfills.js"></script>
8 <script src="node_modules/rxjs/bundles/Rx.js"></script>
9 <script src="node_modules/angular2/bundles/angular2.dev.js"></script>
10 <script src="node_modules/angular2/bundles/router.dev.js"></script>
```

Figur 43

Index.html kan indeholde en række polyfills. Det er JS-biblioteker, som gør at ældre browserversioner kan benytte Angular. Hvilke versioner skal man teste/tjekke fra gang til gang i dokumentationen. Det afhænger af den downloadede package.

<sup>4</sup> Default for elementnavne kan sættes i cli-config filen.

### Angular 4 - Udvikling af Angular apps

At polyfills dækker 100% af al funktionalitet i Angular er også lidt naivt. Der henvises her til dokumentation, github, angular.io, stackoverflow m.m., da det rigtige svar kan variere fra case til case.

## Index.html med CLI er lidt anderledes

Bemærk: Benyttes CLI til Angular (ng serve) vil denne ikke indeholde Polyfills:

```

2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>NyPakke</title>
6 <base href="/">
7
8 <meta name="viewport" content="width=device-width,
9 initial-scale=1">
10 <link rel="icon" type="image/x-icon" href="favicon.ico">
11 </head>
12 <body>
13 <app-root></app-root>
14 </body>
15 </html>
```

Figur 44

## Må man sætte egen kode ind i Index.html?

Både ja og nej. Index.html er en almindelig html-side. Så det korrekte svar er, at man sådan set godt kan sætte egen html ind i både <head>-delen og <body>-delen.

Fordelen er jo indlysende. Alle komponenter indlæses via index.html og rod-komponentet. Derfor er det en nem og komfortabel måde at lave styling af sine komponenter på.

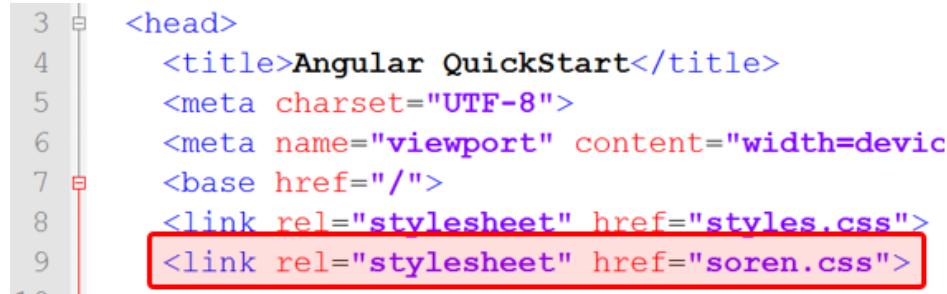
Angular har egne TypeScript-baserede metoder til f.eks. at lave CSS - og disse bør benyttes jfr. dokumentationen. På den måde er man sikker på, at alle komponenter fungerer korrekt. Komponenter er autonome enheder, som skal kunne fjernes og sættes ind andre steder. Derfor er det teoretisk set bedst, hvis alt omkring komponentet er kodet her.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Men undertegnet bryder selv reglerne en smule:



```
3 <head>
4 <title>Angular QuickStart</title>
5 <meta charset="UTF-8">
6 <meta name="viewport" content="width=device-width, initial-scale=1">
7 <base href="/">
8 <link rel="stylesheet" href="styles.css">
9 <link rel="stylesheet" href="soren.css">
```

Figur 45

Argumentet er, at jeg af undervisningsmæssige årsager, gerne vil holde min kode så ren som muligt - og ikke snavse komponenter til med CSS alle vegne.<sup>5</sup>

## app/main.ts og app.module.ts

I app-mappen<sup>6</sup> findes yderligere 2 filer, som benyttes til at loade de øvrige komponenter ind.

Et modul kan opbygges på forskellig vis. mail.ts og module.ts kan derfor være kombineret i én fil eller hedde noget andet. Det afhænger af hele opsættet.

```
1 import { enableProdMode } from '@angular/core';
2 import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3 import { AppModule } from './app/app.module';
4 import { environment } from './environments/environment';
5
6 if (environment.production) {
7 enableProdMode();
8 }
9 platformBrowserDynamic().bootstrapModule(AppModule);
```

Figur 46

<sup>5</sup> Benyttes CLI, vil der automatisk være oprettet en src/styles.css til globale stylesheets. Derfor vil man med CLI sjældent have behov for at pege på sin egen CSS-fil. At CLI benytter src/styles.css kan ses/redigeres i .angular-cli.json filen.

<sup>6</sup> Benyttes CLI vil main.ts filen ikke ligge i src/app-mappen, men derimod i src-mappen. Denne fil kan være placeret forskelligt alt efter opsæt - og alt efter om man arbejder med Angular 2 eller 4.

## Angular 4 - Udvikling af Angular apps

## Kapitel 5 Filer i en applikation

Her ses et eksempel på main.ts filen. Den primære opgave er at pege på **Module.ts** filen (linje 3 og 9). I parentes skal det nævnes, at der kan sættes parametre efter, om der køres i udvikler- eller drift-/produktionsmiljø.

```

1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppComponent } from './app.component';
5
6 @NgModule({
7 imports: [BrowserModule],
8 declarations: [AppComponent],
9 bootstrap: [AppComponent]
10 })
11 export class AppModule { }

```

Figur 47

### main.ts

- Linje 1 - import af *Bootstrap Decorator*  
denne kan afvige afhængig af klient (browser, mobile) -  
se evt. <http://www.developer.com/services/writing-my-first-angularjs-2-app.html> (søg på 'bootstrap decorator function')
- Linje 2 - import af *App Component funktion*
- Linje 3 - *Bootstrap*/Loading af *Root Component*

```

1 import { bootstrap } from '@angular/platform-browser-dynamic';
2 import { AppComponent } from './app.component';
3 bootstrap(AppComponent);

```

Figur 48

Module.ts bliver altså loaded ind af main.ts. Normalt regnes module.ts for at være den "øverste" fil. Det er her ens projekt bliver orkestreret - det er her de forskellige features og moduler angives.

Module.ts loader forskellige moduler ind (NgModule og BrowserModule). Under decoratoren @NgModule indlæses AppComponent.

AppComponent er vores grundlæggende komponent - vores rodkomponent. Det er det yderste komponent i hvilke alle andre komponenter (hvis der er flere) er indlejret).

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Features

---

Features er en fælles betegnelse for:

- ▶ class
- ▶ component (den **vigtigste**)
- ▶ directive
- ▶ enum
- ▶ module
- ▶ pipe
- ▶ service
- ▶ guards
- ▶ interface

Om features kan man sige:

- ▶ Alle projekter er opbygget af features
- ▶ Features er filer skrevet i TypeScript (.ts extension) **7**
- ▶ Alle projekter har mindst 1 component.

---

**7** Det er muligt at udvikle Angular uden TypeScript. Man kan bruge Dart eller ren JavaScript. Men Angular-teamet anbefaler på det kraftigste TypeScript. Forfatter vurderer, at over 98% af al Angular udvikling er med TypeScript.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

I filen src/styles.css kan man indsætte global css til hele ens projekt. Man skal ikke linke til filen fra dine komponenter.

```

1 @import url('https://fonts.googleapis.com/c
2 body {
3 padding:0px;
4 font-family:Lato;
5 color:darkblue;
6 height:200%;
7 overflow-y: auto;
8 overflow-x: hidden;
9 }
10
11 button, input[type=button] {
12 background-color:#fafafa;
13 border:none;
14 padding:10px;
15 font-size:16px;
16
17 hr {background-color:yellow;}
18
19

```

Styles kan sættes til 3 scopes.

**Emulated** (standard) betyder at styles angivet under @Component kun hører til det aktuelle komponent. Styles angivet i HTML-templetten hører kun til komponentet.

**Native** betyder, at kun styles angivet under @Component virker i det aktuelle komponent. Styles angivet i HTML-templetten ignoreres.

**None** betyder, at styles i @Component føres ud i HTML-templetten. Derfor er sådanne styles synlige for alle andre komponenter.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Overblik over alle filer

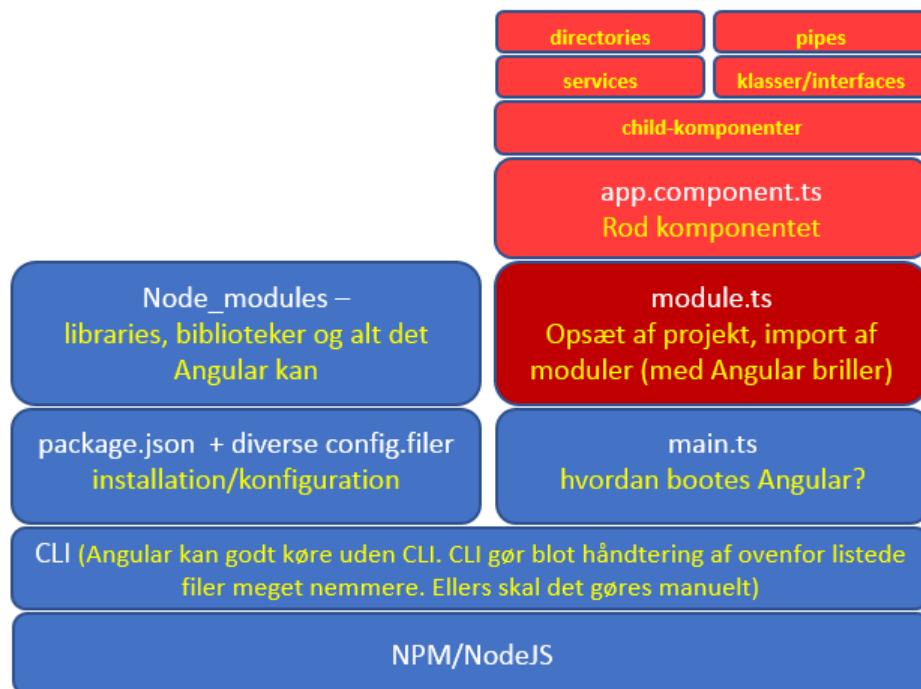
Her ses et overblik over hele platformen, lidt groft reduceret til få kasser.

### De røde/brune kasser

De røde/brune kasser er selve Angular. Her udvikles der i TypeScript. De blå kasser er en blanding af servere, konfigurationsfiler, javascript-filer (node\_modules), css filer og meget andet.

### De blå kasser

De blå kasser er groft sagt hele infrastrukturen omkring Angular, hvor de blå kasser er stedet, hvor der udvikles i editoren. Denne bog handler primært om de blå kasser.



Figur 49

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## BootStrap er en del af Angular

Med en simpel npm-kommando kan man downloade og installere bootstrap.

```
npm install --save bootstrap
```

Herefter redigeres `.angular-cli.json` filen.



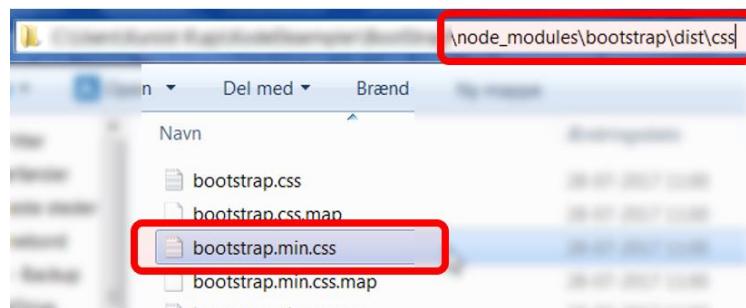
Figur 50

Fra `styles` tilføjes en sti til Bootstrap:

```
"prefix": "app",
"styles": [
 "../node_modules/bootstrap/dist/css/bootstrap.min.css",
 "styles.css"
],
```

Figur 51

Stien peger på Bootstrap css.filens.



Figur 52

Alternativt kan man selvfølgelig undlade at downloade Bootstrap og ganske enkelt linke til Bootstrap cdn-filen fra `<head>`-elementet i `index.html`. Sidstnævnte er blot ikke den officielle Angular metode.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Konklusion

---

De nævnte filer er framework opsættet for et modul. Disse filer vil normalt være relativt stationære. Selvfølgelig kan der komme ændringer - især i forbindelse med opgradering af Node.js serveren eller ændringer i dependencies.

Men i praksis sker det meste af udviklingen i de forskellige features (components, directives, services m.m.). De resterende filer er groft sagt opsættet omkring komponenterne/klasserne.



app.component.ts

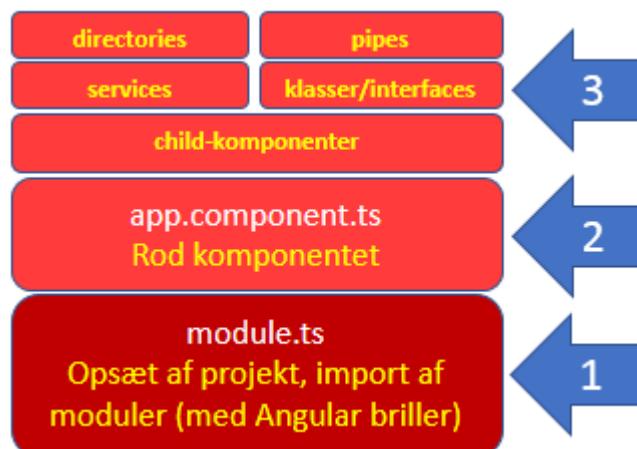
Figur 53

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 6 Angular Arkitektur

For at forstå arkitekturen i Angular skal man forstå komponenter.



Figur 54

Alle røde kasser her symboliserer en TypeScript-fil. Der kan kun være en module.ts (pil 1) fil. Men de øvrige filer kan der være mange af.

Alle projekter har et rod-komponent (pil 2). Et komponent, hvor det hele starter. Fra det komponent loades alle andre komponenter ind.

Rodkomponentet hedder app.component.ts<sup>8</sup>

---

<sup>8</sup> Teknisk set kan app.component.ts godt navngives anderledes. Men for at holde den røde tråd og følge best practice og alle konventioner antages det, at rodkomponentet altid hedder app.component.ts.

## Angular 4 - Udvikling af Angular apps



Figur 55

Rodkomponentet vil her være den blå firkant. Inden i rodkomponentet findes flere andre komponenter (rød, grøn, gule firkanter). Disse kaldes ofte for **child-komponenter**.

Hver komponent er en fil. Index.html griber fat i rod-komponentet - og kun rod-elementet. Det er op til rodelementet at loade de øvrige child-komponenter.

- ➡ Der er altid kun **1 rod-komponent**
- ➡ Der kan være **ubegrænset antal child-komponenter** både i dybden og bredden. Der kan sagtens være en child til en child til en child.
- ➡ **Komponenter kan genbruges.** Det øverste grønne komponent kan vises midt i det gule komponent.
- ➡ **Routing giver mulighed for at udskifte komponenter** med andre komponenter (kommer senere). Indholdet i den gule firkant kan dynamisk udskiftes

**Vigtig pointe:** Der er tale om kun 1 html-side (index.html). Den reloades på intet tidspunkt. De forskellige komponenter kan vises inden i hinanden - udskiftes - fjernes - vises flere gange og så videre. Men siden reloades ikke. Derfor hedder det en **Single Page Application SPA**.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Komponenter

Angular består af en række features eller building blocks. Disse er:

- class
- **component (den vigtigste)**
- directive
- enum
- module
- pipe
- service
- interface
- guards

Der er tale om selvstændige filer, som allerede nævnt. Et projekt kan f.eks. bestå af 4 klasser, 2 services, 1 directive og 1 pipe.

Komponenter er de vigtigste. App.component.ts er det vigtigste komponent. Det er rodkomponentet, som loades ind i index.html.

Et komponent er dybest set en TypeScript klasse, så Angular kan lave instanser/objekter ud fra denne klasse.

## Et simpelt komponent

Når man oprette et ny projekt med CLI-kommandoen `ng new mit-projekt-navn` får følgende rod-komponent  
(src/app/app.component.ts)

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

@Component({
 selector: 'app-root',
 template: `<h1>Angular Komponent</h1>`
})

export class AppComponent { }
```

Figur 56

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Den røde markering viser klassen. Hele filen er en TypeScript-klasse. Klassen indeholder ikke noget kode - men skal angives. I klassen laver man alt logikken tilhørende komponentet. Klassen programmeres i TypeScript.

Den blå markering viser Meta-data til klassen. Metadata er her 2 parametre. Nemlig **Selector**, der grundlæggende viser det html-element, der skal benyttes for at vise komponentet (<app-root>). **Templaten** er den html, som vil blive vist i browseren.



Figur 57

Index.html har en <app-root>. Når projektet startes med **ng new**, hentes app.component.ts ind. Dens har en selector, som matcher (app-root). Dermed vises templetten i app.component.ts i index.html

Med template menes html-kode - det som brugeren kan se i browseren. Templetten angives i komponentet (hvis **template** er benyttet) eller i en separat html-fil, hvis **templateUrl** er benyttet.

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

Figur 58

Den grønne markering (import) angiver, at component-modulet skal importeres fra '@angular/core'. Denne linje skal altid ligge øverst i alle komponenter. Denne import tillader brug af **@Component-dekoratoren**.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

```

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 title = 'app';
}

```

Figur 59

**Component** (rød) importeres - og angives herefter som en såkaldt **dekorator** (grøn). **@Component**-linjen (grøn) kaldes en **Decorator Function** eller **Component Artifact**.

Alt inden for **@Component** (blå) kaldes **Meta Object**. Man siger at **Decorator Funktionen tager et Meta Object**.

**Meta Object'er består af felter** (her **selector** og **template**), men kan også indeholde **styles**, **styleUrls**, **host** og flere andre.

**AddComponent (gul)** er en klasse som indeholder **logik** - herunder services, routring m.m. Klassen kan i mange tilfælde være tom.

**Export** skal angives. Ellers kan komponentet ikke importeres andre steder fra.

## 1-way binding - interpolation (metode 1)

1-way binding fungerer helt automatisk i Angular. 1-way binding betyder, at properties (variable) i klassen kan vises direkte i templatene.

```
@Component({
 selector: 'my-app',
 template: `<h1>Angular 2: Hello World</h1>
 <p>Hej {{navn}} </p>
 `

})
export class AppComponent {
 navn:string="ZoomTek";
}
```

Figur 60

1-way binding laves med interpolation - dobbelt tuborg-parenteser. Interpolation kan indeholde properties (som vist her), class properties, beregninger m.m.

## 2-way binding med ([ngModel])

2-way binding laves med (**[ngModel]**) (ikke at forveksle med ngModule, som er noget ganske andet).

```
<input [(ngModel)]="navn">
```

Figur 61

Her er et input-felt. Når der skrives i input-feltet, vil navn-proprietien ændres. Når navn-proprietien ændres, vil værdien øjeblikkelig blive ændret i feltet.

Der findes reelt 5 former for binding, hvoraf de 3 benytter ngModel, [ngModel] samt [(ngModel)] (sidstnævnte vist ovenfor). Derudover er der simpel interpolation (vist ovenfor) og endelig binding direkte på form-objektet med ngForm.

Alle 5 metoder er nærmere beskrevet i kapitlet om Template-Driven forms.



# Kapitel 7 CSS - Style Sheets

## src/styles.css

I filen **src/styles.css** kan man indsætte **global css til hele ens projekt**.  
Man skal ikke linke til filen fra dine komponenter.

Som med al anden CSS kan man lokalt overskrive de globale definitioner med **styles** og **styleUrls** i **@Component**.



```
1 @import url('https://fonts.googleapis.com/...
2 body {
3 padding:0px;
4 font-family:Lato;
5 color:darkblue;
6 height:200%;
7 overflow-y: auto;
8 overflow-x: hidden;
9 }
10 button, input[type=button] {
11 background-color:#fafafa;
12 border:none;
13 padding:10px;
14 font-size:16px;
15 }
16 hr {background-color:yellow;}
```

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 7 CSS - Style Sheets

Styles kan sættes til 3 scopes. Scope angives med **encapsulation** i forbindelse med @Component-metadata.

```
@Component ({
 selector: 'app-child3',
 templateUrl: './child3.component.html',
 encapsulation: ViewEncapsulation.None,
 styleUrls: ['./child3.component.css']
})
```

**Emulated** (standard) betyder at styles angivet under @Component kun hører til det aktuelle komponent. Styles angivet i HTML-templaten hører kun til komponentet.

**Native** betyder, at kun styles angivet under @Component virker i det aktuelle komponent. Styles angivet i HTML-templaten ignoreres.

**None** betyder, at styles i @Component føres ud i HTML-templaten. Derfor er sådanne styles synlige for alle andre komponenter.



## Kapitel 8 Semantic Versioning (SemVer)

Angular Teamet har indført en detaljeret plan for versionering af Angular. Herunder release af nye udgivelser. Således vil der i fremtiden komme en Angular 5, Angular 6 og så videre. Derfor bør man blot kalde hele platformen for Angular - uden nummer.

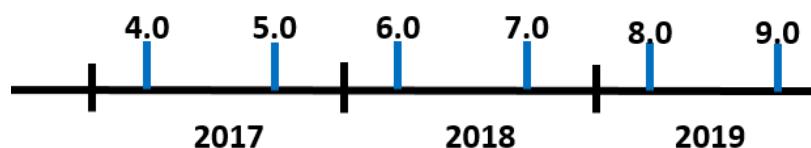
For at sikre en udvikling af Angular, uden at man som udvikler skal bekymre sig for om ens kode og viden bliver forældet, indfører man såkaldt Semantic Versioning. Dermed er nye releases forudsigelige, indbyrdes kompatible og alt kan nemmere dokumenteres.

Efter Angular 2.0.0 foregår fremtidig udvikling på følgende måde:

- ➡ Semantisk Versionering benyttes for angivelse af Angular versioner
- ➡ Nye releases kommer ud med en fast tidsinterval, så planlægning er nemmere
- ➡ Angular Teamet har indført en 'deprecation policy' - altså hvornår dele af API'et udgår
- ➡ Der skelnes meget firkantet mellem 'stable'og 'experimental' API'er

### Hvad er Semantic Versioning?

SemVer betyder at versionnumre giver mening - at der er tænkt over dem. Patch releases ændrer ikke funktionalitet, Minor releases vil kun indeholde tilføjelser og større ændringer er forbeholdt Major Releases.



Figur 62

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Time-based Releases

---

Der er lavet tidsskema for Major og Minor Releases. Patches udgives efter behov.

Generelt vil der komme en patch release hver uge. Og for hvert halve år forventes 3 Minor Updates og 1 Major Update. Alle Minor og Major Release vil først komme ud i en Beta udgave og en Release Candidate - før den endelige Release.

Med dette system, kan udviklere se de seneste nyheder, men stadig have et stabilt produktionsmiljø.

## Deprecation (Udløb af API)

---

- ➡ Når en deprecation annonceres, vil der også blive anviset, hvordan ens kode kan opdateres.
- ➡ API'et kører gennem deprecation perioden - og der vil mindst være 6 måneder til at foretage en opdatering.
- ➡ Alt med dependencies bliver først ændret ved Major Releases.

## Experimental

---

Visse dele af API'et er angivet som Experimental. Disse API'er kan godt køre i produktion, men der kan komme ændringer og de kan blive deprecated.

## Public API Surface

---

På dette link kan man finde de officielle packages i Angular API'et. Også kaldes Public API Surface.

[https://github.com/angular/angular/blob/master/docs/PUBLIC\\_API.md](https://github.com/angular/angular/blob/master/docs/PUBLIC_API.md)

@angular/animations  
 @angular/core  
 @angular/common  
 @angular/platform-browser  
 @angular/platform-browser-dynamic  
 @angular/platform-server  
 @angular/platform-webworker

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 8 Semantic Versioning (SemVer)

```
@angular/platform-webworker-dynamic
@angular/upgrade
@angular/router
@angular/forms
@angular/http
```

På <http://angular.io> under API (se menu), kan man se hele API'et.  
Herunder kan man filtrere på:



Figur 63

Component	Directive	HostBinding
HostListener	Input	Output
Pipe	AfterContentChecked	AfterContentInit
AfterViewChecked	AfterViewInit	DoCheck
OnChanges	OnDestroy	OnInit
CUSTOM_ELEMENTS_SCI	ModuleWithProviders	NgModule
ViewEncapsulation	Version	Injector
ReflectiveInjector	Provider	TypeProvider

Figur 64

Dokumentationen herinde er ikke altid fyldestgørende - og man kan være henvist til andre kilder.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 9 TypeScript

- ➡ Hvad er TypeScript helt præcist?
- ➡ Opsætning af TypeScript (med og uden Visual Studio)
- ➡ Syntaks
- ➡ Datatyper
- ➡ Klasser
- ➡ Constructors
- ➡ Decorators
- ➡ Arrays

## Et par npm kommandoer

**npm install -g typescript@2.2.0**

Opgraderer/installerer version 2.2.0, som er nødvendig for at køre CLI. Strengt taget er version 2.1.0 nok til Angular 4.<sup>9</sup>

**tsc -v**

Test versionen som er installeret på computeren.

## Hjem står bag TypeScript?

TypeScript er udviklet af Google og Microsoft. Danskeren Anders Hejlsberg har været den ledende person for udvikling af TypeScript. Det er også Anders Hejlsberg, der har udviklet C#, Delphi og Turbo Pascal.

TypeScript er open source. TypeScript benyttes i Angular, men Angular har "lånt" TypeScript. TypeScript er ikke lavet til Angular som sådan.

---

<sup>9</sup> TypeScript installeres automatisk sammen med CLI. Men derfor er det godt at kunne opdatere og installere via npm-kommandoer, hvis noget går galt.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Hvad er TypeScript helt præcist?

### TypeScript

- Programmeringssprog - compiles ned i plain JS. Browzere kan kun forstå JS
- Compileren hedder *tsc*

Figur 65

TypeScript er det sprog, som man bruger til udvikling af Angular. Sproget er et superset af JavaScript, og kender man JavaScript, vil man hurtigt lære TypeScript.

TypeScript er et såkaldt **strict syntactical superset** af JavaScript (ECMAScript 2015). Det vil sige en udvidelse af JavaScript. Alt JavaScript kode kan kopieres direkte ind i TypeScript kode. TypeScript kom frem i 2012.

TypeScript er både nemt, og man behøver ikke kunne alle detaljer for at lave Angular kode. TypeScript er ret nemt at læse. Her er et eksempel:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4 selector: 'my-app',
5 template: `<h1>Angular 2: Hello World</h1>
6 <input [(ngModel)]="navn">
7 <p>Hej {{navn}} </p>
8 `
9 })
10 export class AppComponent {
11 navn:string="Søren Anderson";
12 }
```

Figur 66

Alt i Angular er TypeScript.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# TypeScript skal compiles til JavaScript

TypeScript kan dog ikke vises direkte i browseren. Angular har en kompiler, som kører i runtime.

Filen ovenfor kompileres til en JavaScript-fil - uden at man som udvikler skal gøre noget for det. JavaScript-filen bliver så vist i browseren. Man kan faktisk ændre i JavaScript-filen. [10](#)

Kompileringen sker on-the-fly, og browseren opdateres med det samme (uden F5). Nedenfor ses den samme fil, blot som den ser ud i JavaScript.

```
1 "use strict";
2 var _decorate = (this && this._decorate) || function (decorators, target, key, desc) {
3 var c = arguments.length, r = c < 3 ? target : desc === null ? desc = Object.getOwnPropertyDescriptor(target,
4 key) : desc, d;
5 if (typeof Reflect === "object" && typeof Reflect.decorate === "function") r = Reflect.decorate(decorators,
6 target, key, desc);
7 else for (var i = decorators.length - 1; i >= 0; i--) if (d = decorators[i]) r = (c < 3 ? d(c) : c > 3 ? d(
8 target, key, r) : d(target, key)) || r;
9 return c > 3 ? r : Object.defineProperty(target, key, r);
10 }
11 var _metadata = (this && this._metadata) || function (k, v) {
12 if (typeof Reflect === "object" && typeof Reflect.metadata === "function") return Reflect.metadata(k, v);
13 }
14 var core_1 = require('@angular/core');
15 var AppComponent = (function () {
16 function AppComponent() {
17 this.navn = "Ole Christian Anderson";
18 }
19 AppComponent = _decorate([
20 core_1.Component({
21 selector: 'my-app',
22 template: '<h1>Angular 2: Hello World</h1>\n<t<input [(ngModel)]="navn"/></t><p>He: {{navn}}</p>\n</t>\n</t>',
23 }),
24 _metadata('design:paramtypes', [])
25);
26 })(AppComponent);
27
```

Figur 67

Der er skruet lidt ned for størrelsen - for filen er størrelsesmæssigt markant større - og koden er meget mere kompleks. Et sandt mareridt at kode selv direkte i JavaScript.

Der er absolut ingen grund til at kode direkte i JavaScript. All dokumentation til Angular er lavet i TypeScript (og under 1% i Dart).

**10** Benyttes CLI kan man ikke se JS-filerne. Det er helt ude på teoriens overdrev at skulle redigere direkte i JS-filerne. Uden CLI og ved ældre boilerplates fra Github ville både JS-filerne og såkaldte MAP-filer være synlige parallelt med ts-filerne. F.eks. ville der både være en app.component.ts og en app.component.js fil. JS-filerne er der stadig, men er skjult i nogle underbiblioteker. Kun JS-filer kan vies i browseren. Alt dette er konfigureret automatisk - og der er ingen grund til at lede efter disse JS-filer.

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Men i teorien kan man slå TypeScript-kompileren fra - og kode direkte i JavaScript - men i praksis vil man dog aldrig, aldrig gøre dette.

## Hvorfor benytte TypeScript?

---

TypeScript indeholder en række ekstra features, som enten slet ikke findes i JavaScript eller som kræver meget programmering.<sup>11</sup> Dette kursus kommer ikke rundt i alle hjørner i TypeScript.

- ➡ Type annotations and compile-time type checking
- ➡ Type inference
- ➡ Type erasure
- ➡ Interfaces
- ➡ Enumerated type
- ➡ Mixin
- ➡ Generic
- ➡ Namespaces
- ➡ Tuple
- ➡ Await

Her er listet lidt løst og fast, som gælder for TypeScript.

---

<sup>11</sup> Det ligger ud over denne bog, at gå i detaljer med disse features. Se evt. Wikipedia på <https://en.wikipedia.org/wiki/TypeScript>

## Kapitel 9 TypeScript

- Fejltjek ved kompilering
- TS understøtter OOP – encapsulation, inheritance, Abstraction, Polymorphism
- TS retter uhensigtsmæssigheder i JavaScript
- JavaScript kan skrives native inden i TS
- TS skal downloades og installeres med npm
- TS kan startes med npm start eller ng serve (CLI)
- Optional Static Typing
- Komponenter: Sprog (syntaks), Compiler (har brug for Nodejs)
- Mere struktureret end JavaScript
- TS er open source – designet af en dansker
- Alternativer til TS: Ren Javascript eller Dart
- TS kører ikke direkte i browseren
- TS skal kompileres til JavaScript
- Understøtter klasser og Interfaces**

Figur 68

## Datatyper

TypeScript tilbyder **static typing**. Man kan ignorere static typing og i stedet bruge **dynamic typing**, som det kendes fra almindelig JavaScript.

- **Boolean:** var synlig: boolean = false;
- **Number:** var alder: number = 34;
- **String:** var navn: string = "Bent";
- **Array:** var liste:number[] = [1, 2, 3];
- **Enum**
- **Any:** var unknownType: any = 8;  
Any gør TS 'optionally statically typed'
- **Void:**

Figur 69

TypeScript er type safe. Det er nemt at definere datatyper i TypeScript:

```
let erBesvaret: boolean = false;
```

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Figur 70

Alle datatyper har en defaultværdi. Denne kan ændre ved at skrive = *et\_eller\_andet*, der selvfølgelig passer til datatypen.

### Type Checking foretages ved compiling

Iet er frivilligt at skrive. Varianter af ovenstående kode kunne være:

```
erBesvaret:boolean;
erBesvaret:boolean=false;
```

Figur 71

## Eksempler på andre datatyper

```
7 tal:number=20;
8 navn:string='Søren';
9 adresse:string='';
10
11 tal: number[] = [1, 2, 3];
12 talGeneric: Array<number> = [1, 2, 3];
13
14 enum By {Kolding, Vejle, Horsens};
15 let b: By = By.Vejle;
16
17 kinkyVariabel: any = 2;
```

Figur 72

Her ses eksempler på **string**, **array**, **generics**, **enum** og **any**.

En af de mest hyppige fejl i Angular er fejl med datatyper. Disse kan man ofte slippe væk fra, ved at man definerer en variable som Any - selv om det ikke altid er den anbefalede løsning. Afhængig af situationen vil det oftes være bedre at finde den rigtige datatype.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Brug af *this*

```
export class AppComponent {
 text: string;
 value:string;
 onChange(value:string) {
 this.text=value;
 }
}
```

Figur 73

Her ses en klasse i TypeScript. **Text** er defineret som en string. I metoden **onChange()** refereres til **text**, men bemærk hvordan man skal referere til variablen med **this**.

Laver man interpolation i templatene skal *this* ikke skrives med. I stedet skriver man {{text}}



## Klasser

I Angular er et komponent (eller et directive, pipe osv.) en klasse. Ser man på en simpel klasse består den typisk af 3 dele.

- ➡ En import del,
- ➡ En decorator-del
- ➡ Selve klassen

Disse 3 dele, er alle dele af klassen - og er alle skrevet i TypeScript. **Import** sørger for, at de rette elementer er til rådighed. **Decorator**-delen angiver metadata, mens **selve klassen** indeholder metoder og andre members - altså selve indmaden i klassen.

### TypeScript og klasser

- Kan (skal ikke) indeholde en constructor
- member variable
- properties
- metoder
- Access Modifiers (`private, public(default)`) for variable og funktioner (members)

Figur 74

Klasser benyttes massigt i Angular. Angular er komponent-baseret. Al kode er bygget op i komponenter. Et komponent er det samme som en klasse.

Så skæret ind til benet, består et Angular modul af 1 til mange klasser, der interagerer med hinanden. Hver klasse bygges op i en simpel fil - og filerne i modulet taler sammen.

Klasser og komponenter kan godt være komplikerede. Men i et Hello World eksempel består de af få linjer kode. Man kan godt lave flere klasser i samme fil. Men igen - som udgangspunkt er hver fil en klasse/et komponent.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Her er et eksempel på en klasse:

```

1 class Kunde {
2 kundenavn: string;
3 constructor(_navn: string) {
4 this.kundenavn = _navn;
5 }
6 velkommen() {
7 return "Hej, " + this.kundenavn;
8 }
9 }
10 kunde=new Kunde("Niles Crane");

```

Figur 75

Klassen indeholder 3 members. En member variabel (linje 2)**12**, en konstruktur (linje 3-5) og en metode (linje 6-8).

En klasse behøves ikke have members. Ofte vil man i Angular sammenhæng se en klasse uden members. Et komponent skal nemlig **altid have defineret en klasse**.

## En Klasse skal eksporteres

```

1 export class KundeComponent { }

```

Figur 76

Her ses en klasse, som den kunne tage sig ud i Angular. Bemærk *Export* som gør den synlig for andre komponenter (svarer nærmest til *public* i f.eks. C#.net).

Endvidere har klassen fået navnet KundeComponent. En klasse bør i Angular hedde noget med Component.

Der findes andre typer klasser end komponenter - f.eks. Service, Directive osv. Typen bør også afspejle sig i klassenavnet.

Klasser (eller interfaces) benyttes ofte til at 'hardcode' en række datasammenhæng. Til at definere et Schema - for at bruge et XML-udtryk.

**12** En member variabel kaldes også en class property

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



```
1 export class Medarbejder {
2 nr: number;
3 navn: string;
4 adresse: string;
5 postnummer: string;
6 by: string;
7 }
```

Figur 77

Ovenstående er det eneste indhold i en fil. *Export*gør klassen synlig for omverden. Filen hedder givetvis *medarbejder.ts* (små bogstaver).

Klassen kan benyttes fra andre filer/klasser/komponenter ved at skrive:

```
import { Medarbejder } from './medarbejder';
```

Skriv IKKE .ts efter filnavn

Figur 78

Bemærk Case. Man må **ikke** skrive *./medarbejder.ts*. TS er underforstået.

Bemærk: Medarbejderklassen hedder *Medarbejder* og ikke *MedarbejderComponent*. Det skyldes, at man kun definerer klassen. Man definerer ikke et komponent.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Nedarvning af klasser

### TypeScript og Inheritance (nedarvning)

- Benyt *Extends*
- super() kalder Base-class constructor

```
class Cat extends Animal {
 constructor(name: string, domestic: boolean)
 {
 super(name);
 this.domestic = true;
 }
}
```

Figur 79

Klasser kan nedarves med *extends*

## Parametre i funktioner

### Optional parametre i funktioner

- Benyt ? efter variabelnavn

```
function Demo(arg1: number, arg2? :number) {
```

Figur 80

## TypeScript - ressourcer

- Læs om Moduler
- Hvordan laves TS description file (tsc) ud fra ts-fil
- Declare keyword
- TSD TypeScript Definition Manager - til fx. at køre JQuery...
- Debugging af TS
- Function Overloading

<http://www.talkingdotnet.com/typescript-interview-questions/>

Figur 81

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## TypeScript kode for dummies

Her ses en simpel klasse.

```
1 export class Kunde {
2 navn:string;
3 adresse:string;
4 postnummer:string;
5 by:string;
6 }
```

Figur 82

Koden ligger i en selvstændig fil. Den importeres ingenting, og der er ingen decoratorer. Klassen eksporteres for at kunne bruges. Idéen med klassen er blot, at man kan bruge en *Kunde* i andre komponenter.

Klassen kan oprettes manuelt eller med CLI:

```
ng generate class kunde
```

Figur 83

Linje 1 + 6 oprettes automatisk. Linje 2-5 skal man selv tilføje. Fra component.ts importeres klassen<sup>13</sup>:

```
1 import { Component, OnInit } from '@angular/core';
2 import { Kunde } from './kunde';
3
4 @Component({
5 selector: 'app-root',
6 template: '
7 <h1>Angular: {{title}}!</h1>
8 <app-kunde/>
9 ',
10 })
11 export class AppComponent implements OnInit {
12 title = 'Welcome to Angular';
13 kunde: Kunde;
14
15 constructor() {
16 this.kunde = new Kunde('John Doe', 'John Doe', '1234 5678 9012 3456', 'København');
17 }
18
19 ngOnInit() {
20 console.log(this.kunde);
21 }
22 }
```

Figur 84

<sup>13</sup> Klassen kan også indsættes direkte i komponentet i stedet for at blive importeret. Best practice er at indsætte klassen lige under de øvrige import-linjer.

## Instance af klassen

```

13 export class AppComponent implements OnInit {
14 title = 'app';
15 hilsen:string;
16 kunde:Kunde;
17 constructor() {
18 this.kunde=new Kunde();
19 }
20 ngOnInit() {
21 this.hilsen="Med venlig hilsen";
22 this.kunde.navn="Palle Gulvballe";
23 }

```

Figur 85

I den røde firkant oprettes objektet *kunde*.

I den blå firkant defineres kunde som en **instance** af Kunde

I den grønne firkant sættes får navn-proprietien på instansen en værdi.

Her benyttes både **constructor** og **OnInit**. Constructoren udføres først - herefter OnInit. I dette eksempel kan man indsætte koden vilkårligt. Al koden kan sagtens være i constructoren. Det hedder sig (best practice), at constructoren skal være så light som muligt.

```

template: `
 <h1>Angular: {{title}}!!</h1>
 {{hilsen}}

 {{kunde.navn}},
 {{kunde.postnummer}}
 {{kunde.by}}
`
```

**Angular: app!!**


  
Med venlig hilsen

Palle Gulvballe, 4000 Roskilde

Figur 86

Postnummer og by vises fordi klassen er ændret til at indeholde default-værdier.

```

export class Kunde {
 navn:string;
 adresse:string;
 postnummer:string="4000";
 by:string="Roskilde";
}

```

Figur 87

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Interfaces vs. klasser

Man kan også benytte Interfaces. Disse minder meget om klasser. Mange Angular udviklere er af den mening, at man skal benytte klasser frem for interfaces. Klasser kan indeholde metoder, interfaces kan også indeholder metoder, men disse skal implementeres i den klasse, hvor de importeres.

Her ses samme klasse som ovenfor - blot med en simpel metode tilføjet:

```
export class Kunde {
 navn:string;
 adresse:string;
 postnummer:string="4000";
 by:string="Roskilde";
 helloWorld() {
 alert(this.navn);
 }
}
```

Figur 88

Klassen bliver allerede importeret til komponentet, som blev benyttet før. Dette komponent ændres en smule til:

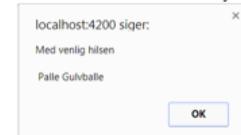
```
ngOnInit() {
 this.hilsen="Med venlig hilsen";
 this.kunde.navn="Palle Gulvballe";
 this.kunde.helloWorld();
}
```

Figur 89

Man kan nemt parse værdier igennem med metoden:

```
this.kunde.helloWorld(this.hilsen);

 helloWorld(tekst:string) {
 alert(tekst+"\n\n "+this.navn);
 }
```



Figur 90

## Angular 4 - Udvikling af Angular apps

## Fat Arrow Funktioner =>

Her ses en almindelig funktion:

```
simpelFunktion () {
};
```

Her ses en anonymous funktion. Den er anonym, fordi den ikke har et navn.

```
setTimeout(function() {
 console.log("setTimeout er kaldt")
,1000);
```

I EcmaScript 6 og TypeScript kan koden skrives således:

```
setTimeout(() => {
 console.log("setTimeout er kaldt")
,1000);
```

Er der kun 1 statement i funktionen, kan koden yderligere forkostes således:

```
setTimeout(() => console.log("setTimeout er kaldt"),1000);
```

## Function Expression

Her er et eksempel på en såkaldt Function Expression.

```
plus=function(a,b) {
 return (a+b)
}
ngOnInit() {
 console.log(this.plus(5,4));
}
```

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Med **Fat Arrow** kan dette omskrives til:

```
plus=(a,b) => {
 return (a+b)
}
```

og Tuborg parenteser kan fjernes. Skrives koden på 1 linje, skal *return* også fjernes.

```
plus=(a,b) => a+b;
```

Laver man funktioner i funktioner, og fra den inderste funktion refererer med *this* til den yderste funktion, vil der returneres enten en fejl eller en undefined.

```
obj={
 navn:"Søren",
 funkex:function () {
 setTimeout(function() {
 console.log(this.navn);
 },1000)
 }
}
ngOnInit() {
 this.obj.funkex();
```

6

```
Angular is running in the development mode.
undefined
```

>

Med Fat Arrow syntaks opstår dette problem ikke.

Nu er resultatet:

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 9 TypeScript

Angular is running in the development mode.

Søren

når koden ændres til:

```
obj={
navn:"Søren",
funkex:function() {
 setTimeout(() => console.log(this.navn),1000)
}
}
```

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Fat Arrow funktioner (eller bare Arrow Funktioner) benyttes massivt i Angular.

```
let subscription = this.data.subscribe(
 value => this.values.push(value),
 error => this.anyErrors = error,
 () => this.finished = true
) ;
```

Figur 91

Fat arrow benyttes til at kalde og definere anonyme funktioner (funktioner uden navn).

```
value => this.values.push(value),
(function(error) {this.anyErrors = error}),
error => this.anyErrors = error,
() => this.finished = true
```

Figur 92

## Editor Support

Editorer anbefalet i denne bog supporterer alle TypeScript. Visual Code Editor er et oplagt valg.

- ➡ Visual Studio
- ➡ Visual Code Editor
- ➡ Sublime

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## tsconfig.json

tsconfig angiver parametre for TypeScript installation

```

1 {
2 "compilerOptions": {
3 "target": "es5",
4 "module": "commonjs"
5 "moduleResolution": "node",
6 "sourceMap": true,
7 "emitDecoratorMetadata": true,
8 "experimentalDecorators": true,
9 "removeComments": false,
10 "noImplicitAny": false
11 }
12 }
```

Figur 93

Parametrene kan være anderledes på ens egen installation. Hele tiden sker der jo opdateringer af delprogrammer.

## TypeScript Watch-mode

### TypeScript Compiler Watch Mode

- tsc -w
- Er automatisk 'tændt'
- Sikrer: (1) browser opdateres dynamisk uden refresh, (2) der dannes/opdateres automatisk js/map-filer ud fra ts-filer.

```

chunk { } polyfills.bundle.js, polyfills.bundle.js.map (polyfill) 5.37 kB { }
chunk { } main.bundle.js, main.bundle.js.map (main) 11.8 kB { }
chunk { } styles.bundle.js, styles.bundle.js.map (styles) 2.4 kB { }
chunk { } vendor.bundle.js, vendor.bundle.js.map (vendor) 0 byt
chunk { } inline.bundle.js, inline.bundle.js.map (inline) 0 byt
webpack: Compiled successfully.
```

Figur 94

Når man kører en applikation med **ng serve** (eller npm start), vil projektet startes op i watch-mode. Ændringer i filer vil blive direkte afspejlet på applikationen i browseren, uden at denne skal opdateres.

Det kan ses ved, at **konsollen hænger**. Alt fungerer. Man lukker for sessionen/watchmode ved at vælge CTRL-C og trykke J for Ja.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



```
chunk { } vendor.bundle.js,
chunk { } inline.bundle.js,
webpack: Compiled successfully.
^CAfbryd batchjob (J/N)?
```

Figur 95

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagere på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 10 Nyheder i Angular 4

- 1) Animation er taget ud af @angular/core
- 2) Forbedre nglf - nu med if/else - variable
- 3) Angular Universal - run Angular on a server
- 4) TypeScript version opdateret (2.1 og 2.2 kompatibel)
- 5) Source Maps -> bliver genereret, når der opstår fejl i kode.
- 6) Flat ES-Modules (Flat ESM / FESM)
- 7) Closure Compability

Disse emner vil blive løbende behandlet i de følgende kapitler.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



# Kapitel 11 Services og Dependency Injection

**Dependency Injection (DI)** er et design pattern, som overfører objekter som såkaldte dependencies (afhængigheder) i forskellige komponenter på tværs af en applikation.

DI opretter en ny instans af klassen sammen med de nødvendige dependencies.

Nedenfor er et simpelt eksempel på DI:

```
1 import { Component, OnInit } from '@angular/core';
2 import { MinService } from './service.component';
3 @Component({
4 selector: 'my-app',
5 template: `
6 <h1>Angular 2: Services</h1>
7 <p>{{this.hilsen}}</p>
8 `,
9 providers: [MinService]
10 })
11 export class AppComponent {
12 hilsen:string;
13 constructor(private ms:MinService) {
14 }
15 }
16 ngOnInit() {
17 this.hilsen = this.ms.helloWorld();
18 }
19 }
20 }
```

Figur 96

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 11 Services og Dependency Injection

Et helt almindeligt component:

```
2 import { MinService } from './service.component';
```

Figur 97

I linje 2 importeres **MinService** fra filen service.component.ts.  
Bemærk navnekonventionen.

```
9 providers: [MinService]
```

Figur 98

I linje 9 angives **MinService** som provider for mit component. I Constructoren i linje 13 oprettes en instans af services med objektet *ms*. Der oprettes også en member-variabel med navnet *hilsen* som string.

```
11 export class AppComponent {
12 hilsen:string;
13 constructor(private ms:MinService) {
14 ...
15 }
```

Figur 99

I linje 16 kaldes funktionen **ngOnInit**. Herfra kaldes metoden **HelloWorld()**, der (ikke overraskende) returnerer en hilsen.

```
16 ngOnInit() {
17 this.hilsen = this.ms.helloWorld();
18 }
19 }
```

Figur 100

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Hvordan ser servicen ud?

Servicen er her meget simpel.

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable()
4 export class MinService {
5 helloWorld() {
6 return "Hello World. Eksempel med DI og Service";
7 }
8 }
```

Figur 101

En service er et komponent. Forskellen ligger i importen af modulet **Injectable**, samt brug af decoratoren **@Injectable()**.

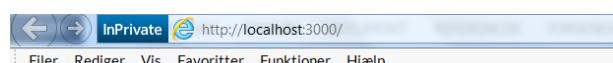
```
export class MinService {
 helloWorld() {
 return "Hello World. Eksempel med DI og Service";
 }
}
```

Figur 102

Klassen indeholder metoden **HelloWorld()**, som returnerer en simpel tekst.

Services virker på tværs af alle komponenter. En service kan indeholde x antal metoder, som kan hente data eller udføre handlinger.

Resultat er følgende:



## Angular 2: Services

Hello World. Eksempel med DI og Service

Figur 103

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 12 HTTP

Med Angulars indbyggede HTTP klient kan man kommunikere med Remote Servers.

De fleste browsere supporterer **XMLHttpRequest (XHR)** API'et samt **JSONP** API'et. Med Angular får man en forsimpler tilgang til disse API'er.

Man benytter **HTTP** for at lave HTTP-kald.

```
import { Http, Response } from '@angular/http';
```

Figur 104

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## app.component.ts

---

Først importeres OnInit, Film og FilmService. Servicen læser fra JSON-filen og Film er selve klassen.

```
1 import { Component, OnInit } from '@angular/core';
2 import { Film } from './film';
3 import { FilmService } from './film.service';
```

Figur 105

@Component skaber viewet. Dybest set blot en masse HTML med interpolation til dataene i applikationen. Ved hjælp af deklarationer som \*ngFor listes indholdet fra JSON-filen (servicen).

```
<h1>Film</h1>
<div class="row grid-view">
 <div class="col-md-4" *ngFor="let film of films">
 <div class="film">

 <p>{{film.navn}}</p>
 <p>{{film.spilletid}}</p>
 </div>
 </div>
</div>
```

Figur 106

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 12 HTTP

Fra Klasse-definitionen oprettes filmService som en instance af FilmService (service (separat fil), hvor JSON importeres).

Der subscribes på hentFilm() i filmService. Der lyttes på hentFilm().  
Der er tale om et asynkront kald.

```
export class AppComponent implements OnInit {
 films: Film[] = [];
 constructor(
 private filmService: FilmService
) {}
 ngOnInit() {
 this.filmService.hentFilm().subscribe(
 data => {
 this.films = data.films;
 }
)
 }
}
```

Figur 107



## Servicen med JSON-kald

---

```
@Injectable()
export class FilmService {

 constructor(private http:Http) { }

 hentFilm() {
 return this.http.get('/app/json/film.json')
 .map((res:Response) => res.json());
 }
}
```

Figur 108

Der foretages kald til JSON-filen med metoden get(). Hvis man ændrer JSON-filen, vil get() automatisk opdatere.

HTTP benytter sig af Observables. Observables benyttes til at abonnere på data - oftest, men ikke altid - kommende udefra (her en JSON-fil).

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 13 Routing

Routing er et kerneområde inden for Angular og en central del af ethvert projekt. Single Page Applications bygger netop på kun 1 side - index.html. Siden vil for brugeren virke som flere sider, da indholdet kan skiftes ud og ændres dynamisk.

Denne udskiftning betyder i praksis at forskellige komponenter udskiftes, fjernes, indsættes dynamisk. Denne proces kaldes **Routing**.<sup>14</sup>

---

<sup>14</sup> Routing er en del af CLI-opsættet og ligger under '@angular/router' - og kan derfor særligt installeres med npm install --save @angular/router. Normalt er sådan en installation ikke nødvendig.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Simpel eksempel med Routing

### Angular Router

[Danmark](#) [Sverige](#)

Danmark Component

Figur 109

I det her meget simple eksempel benyttes 3 komponenter. App.component.ts (den blå tekst og de 2 links) kalder henholdsvis danmark.component.ts og sverige.component.ts afhængig af hvilket link, der trykkes på.

### Angular Router

[Danmark](#) [Sverige](#)

Sverige Component

Figur 110

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Opsætning af Router i module.ts

I module.ts skal Routeren importeres:

```

module.ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { RouterModule, Routes} from '@angular/router';
import { AppComponent } from './app.component';
import { DanmarkComponent } from './danmark.component';

```

Figur 111

Herefter defineres Routing - også i module.ts.

```

import { DanmarkComponent } from './danmark.component';
import { PageNotFoundComponent } from './page-not-found.component';

const appRoutes: Routes = [
 {path: '', redirectTo: "/danmark", pathMatch: 'full'},
 {path: 'danmark', component: DanmarkComponent },
 {path: 'sverige', component: SverigeComponent },
 {path: '**', component: PageNotFoundComponent }
];
@NgModule({
 imports: [
 BrowserModule,

```

Figur 112

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



appRoutes arrayet består her af 4 elementer.

Nummer 1 element er **default elementet**. Når siden loades ind vil der ske en **redirectTo** "/danmark".

Nummer 2 og 3 element er henholdsvis 'danmark' og 'sverige'.

Nummer 4 element (\*\*\*) er en **page not found henvisning**. Alle sider (url's), der ikke findes vil blive redirigeret til PageNotFoundComponent.

```
@NgModule({
 imports: [
 BrowserModule,
 FormsModule,
 RouterModule.forRoot(appRoutes)
],
 declarations: [
 AppComponent,
 DanmarkComponent
]
})
```

Figur 113

Fra @NgModule dekoratoren angives RouterModule som import. Metoden **.forRoot(appRoutes)** anvendes.

appRoutes blev defineret ovenfor.

appRoutes er et array af routes, hver med sin egen konfiguration.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Opsætning af Router i app.component.ts

```
template: <h1>Hello</h1>
app.component.ts <nav>
 Danmark
 Sverige
</nav>
<router-outlet></router-outlet>
```

Figur 114

I app.component.ts indeholder templetten instruktioner til opsætning af Routing.

I den røde blok defineres linkene. Her benyttes `<a>`-elementet, men man kan i principippet benytte andre html-elementer - f.eks. `<img>`.

routerlink-attributten fortæller, hvordan der skal routes, hvis brugeren trykker på linket. routerLinkActive muliggør, at et link kan være aktivt, hvilket ikke er det samme, som at linket er default.

I den blå blok benyttes `<router-outlet>`-elementet. Det er et 'flettefelt' for, hvor child komponenterne skal routes ind.



## Child/nested Router

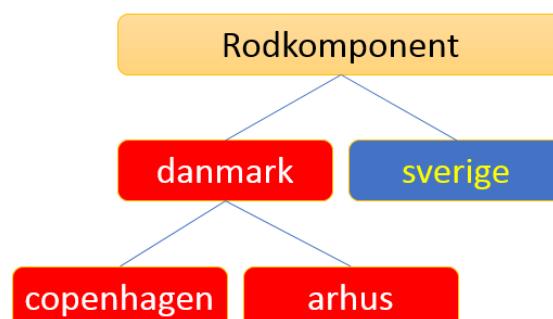
Som løsningen ser ud nu findes flere URL's til child komponenter:

① [localhost:4200/danmark](http://localhost:4200/danmark)

② [localhost:4200/sverige](http://localhost:4200/sverige)

Figur 115

Under /danmark ønskes yderligere 2 child-elementer implementeret med Routing. Nemlig copenhagen og arhus.



Figur 116

Vælger man 'danmark' loades danmark-komponentet ind. I dette komponent er der igen links til 'copenhagen' og 'arhus', som igen kan loades ind med routing.



Figur 117

Her er rod-komponentet, danmark-komponentet og copenhagen komponentet vist.

- (1) Rod-komponentet
- (2) Navigationen på rod-komponentet (mellem danmark og sverige)
- (3) danmark-komponentet
- (4) Navigationen på danmark-komponentet (mellem copenhagen og arhus)

### Angular 4 - Udvikling af Angular apps

## Kapitel 13 Routing

## (5) copenhagen-komponentet

For at lave nestet routing skal child-komponenterne (copenhagen og arhus) naturligvis eksistere. Disse laves nemmest med:

**ng generate component copenhagen** og **ng generate component arhus**

CLI bør redigere module.ts, så komponenterne importeres samt indsættes under declarations.

```
import { CopenhagenComponent }
from './copenhagen/copenhagen.component';
import { ArhusComponent }
from './arhus/arhus.component';

declarations: [
 AppComponent,
 DanmarkComponent,
 SverigeComponent,
 PageNotFoundComponent,
 CopenhagenComponent,
 ArhusComponent
```

Figur 118

Module.ts skal have tilføjet de nye stier:

```
const appRoutes: Routes = [
 {path: '', redirectTo: "/danmark", pathMatch: 'full'},
 {path: 'danmark', component: DanmarkComponent, children: [
 { path: '', redirectTo: 'copenhagen', pathMatch: 'full' },
 { path: 'copenhagen', component: CopenhagenComponent },
 { path: 'arhus', component: ArhusComponent }
]},
 {path: 'sverige', component: SverigeComponent },
 {path: '**', component: PageNotFoundComponent }
];
```

husk komma her

Figur 119

Denne kode sikrer, at routing mellem komponenterne fungerer nemt.

På danmark-komponentet, der skal indeholde links til copenhagen- og arhus-komponenterne indsættes følgende kode:

```
<hr>
Copenhagen
Arhus
<router-outlet></router-outlet>
</div>
```

Figur 120

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Linket til copenhagen-komponentet bliver f.eks.:

① [localhost:4200/danmark/copenhagen](http://localhost:4200/danmark/copenhagen)

Figur 121

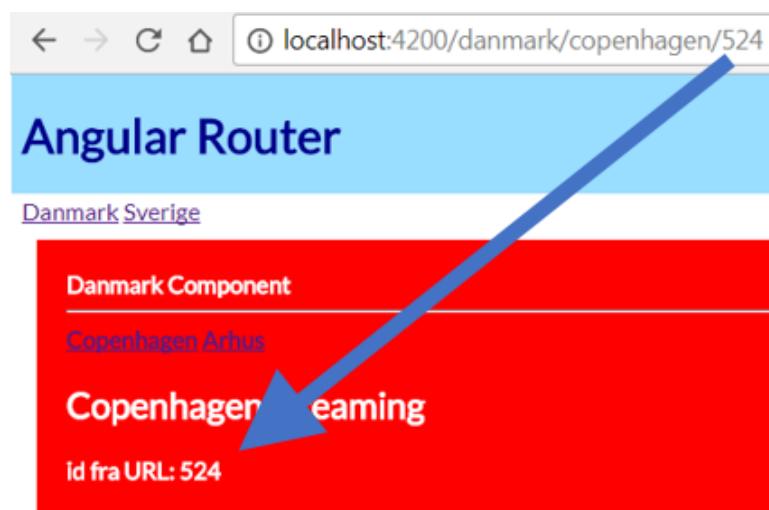
## Route parameter - dynamisk sti

En sti kan være dynamisk.

① [localhost:4200/deltager/sorenanderson](http://localhost:4200/deltager/sorenanderson)

Figur 122

Når man kalder et child-element, kan man parse url'en ind i child-elementet. Således kan man tage 'sorenanderson' og indsætte i child-elementet - eller omvendt. Forvirret? Se her:



Figur 123

copenhagen-komponentet er et nested child-komponent. Det ønskes, at man kan parse en URL afsted til elementet - og elementet kan modtage værdien. Her sendes 524 i URL'en. Skrives 631 i URL'en skal denne naturligvis kunne vises på skærmen (eller hvad man nu vil bruge tallet til).

Tallet (kan også være en tekst) kan f.eks. være et kundenummer, varenummer eller lignende.

Løsningen ser således ud:

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 13 Routing

```

 { path: 'copenhagen', component: CopenhagenComponent },
 { path: 'copenhagen/:id', component: CopenhagenComponent },
 { path: 'articles', component: ArticlesComponent }
]

```

Figur 124

I path'en i module.ts tilføjes endnu en child-path med /:id

: (kolonnet) er vigtigt. Det er det, som gør linket dynamisk. At det kan indeholde alle tal.

Fra **copenhagen** komponentet tilføjes følgende:

### forChild()-metode

Et alternativ til **forRoot()**-metoden er **forChild()**-metoden. De fungerer stort set identisk.

```

1 import { Component, OnInit } from '@angular/core';
2 import { ActivatedRoute } from '@angular/router';
3
4 @Component({
5 selector: 'app-copenhagen',
6 template: `<h2>Copenhagen Dreaming</h2>
7 <div *ngIf='idOprettet'>id fra URL: {{id}}</div>
8 `
9 })
10 export class CopenhagenComponent implements OnInit {
11 idOprettet=false;
12 id: any;
13 forbindelse: any;
14 constructor(private acr: ActivatedRoute) { }
15
16 ngOnInit() {
17 this.forbindelse = this.acr.params.subscribe(params => this.id = params['id']);
18 this.idOprettet=!isNaN(this.id);
19 }
20 ngOnDestroy() {
21 this.forbindelse.unsubscribe();
22 }
23 }

```

Figur 125

```

ngOnInit() {
 this.forbindelse = this.acr.params.subscribe(params =>
 this.id = params['id']);
 this.idOprettet=!isNaN(this.id);
}

```

Figur 126 - Forstørret op

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Øverst importeres ActivatedRoute, som muliggør, at klassen kan spørge på URL'en. ActivatedRoute bliver injected i komponentet med Dependency Injection i Constructoren.

Nederst i ngOnInit() hentes id-værdien ind med en Observable. Der lævest til sidst et check på om parameteren overhovedet er tastet (isNaN()).

Til sidst skrives tallet (f.eks. 524 fra eksemplet) ud i templatlen (blå firkant).

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagere på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

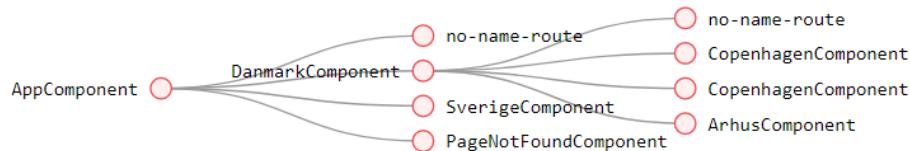
Bemærk, at ønsker man at benytte @router-funktionalitet i Angular, skal der muligvis tilføjes et <base>-element i index.html umiddelbart under <html>-elementet. I nyere CLI-implementeringer er dette element dog allerede indsat.

```

2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>Skabelon2</title>
6 <base href="/">
```

Figur 127

Til sidst ser strukturen i Routing således ud:



Denne graf kan man faktisk få Angular til at generere selv. I senere kapitel vil det blive vist.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



# Kapitel 14 Navigation Guards

Der findes 5 former for Guards:

<b>CanActivate</b>	tester om brugeren kan tilgå en router
<b>CanActivateChild</b>	tester om brugeren kan tilgå en routers children
<b>CanDeactivate</b>	tester om en bruger kan forlade en router
<b>Resolve</b>	udfører data retrieval før router aktiveres
<b>CanLoad</b>	tester om brugeren kan route til et modul, som er Lazy Loaded

Navigation Guards (eller Router Guards eller blot Guards) benyttes i forbindelse med Routing. Guards er (normalt) en selvstændig klasse og kan laves med en **ng generate** kommando.

Guards muliggør at give tilladelse til eller fjerne tilladelse fra dele af routingen. Andre guards kan forhindre, at en bruger forlader en side, uden at have gemt eller submitted indhold.

En given Route kan have ingen, én eller mange Guards.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Sådan laves en simpel Guard - CanActivate

Tidligere eksempel med Routing bruges. Eksemplet hvor der blev nавигeret mellem Danmark og Sverige.

### ng generate guard vagt

med **ng generate** oprettes en guard ved navn **vagt** som en selvstændig klasse.

Der oprettes en vagt.guard.ts (og en spec til testing).

**vagt.guard.spec.ts**

**vagt.guard.ts**

For at lave en Hello World Guard skal der manipuleres med Guard-klassen samt module.ts

**app.module.ts**

**navn.guard.ts**

### I module.ts laves 3 tilføjelser:

```
import { VagtGuard } from './vagt.guard';
```

(1) vagt.guard.ts importeres.

```
{
 path: 'sverige',
 canActivate: [VagtGuard],
 component: SverigeComponent
},
```

(2) På Routingen tilføjes Guard-klassen - her under canActivate

```
providers: [VagtGuard],
```

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udriftning af 1 eksemplar udelukkende til eget brug er tilladt.



(3) Under 'Providers' tilføjes VagtGuard.

## Ændringer i *navn.guard.ts*

ng generate tilføjede **vagt.guard.ts**. Heri laves følgende simple ændringer i canActivate-metoden i klassen:

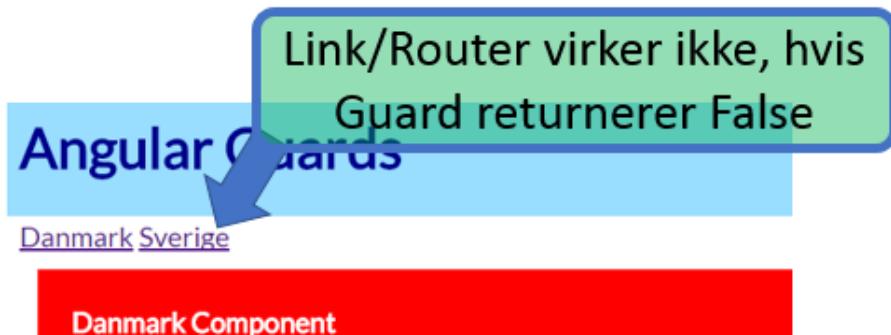
```
canActivate () {
 return true;
}
```

I **canActivate** metoden fjernes alt om Observables og Promises. I stedet bibeholdes en simple **return true**. Så længe der returneres en true, vil routingen til 'sverige' fungere.

I det øjeblik koden ændres til false, vil routingen ikke virke.

```
canActivate () {
 return false;
}
```

Man kan gøre returneringen afhængig af brugerens rettigheder.



## Opbygning af Guards - Hello World eksempel i detaljer

Guards bliver implementeret som en service. Derfor vil man typisk lave en `@Injectable`-klasse.

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot
 #angular/router'
import { Observable } from 'rxjs/Observable'

@Injectable()
export class VagtGuard implements CanActivate {
 canActivate(
 next: ActivatedRouteSnapshot,
 state: ActivatedRouteSnapshot
) {
 return true;
 }
}
```

Guards returnerer enten **True eller False**. Enten som direkte **bolisk værdi**, som vist i Hello World eksemplet ovenfor. Alternativt som en **Observable** eller **Promise**, som senere bliver til en true/false. Observables/Promises er aktuelle, når man ønsker at tilgå asynkrone data.

### vagt.guard.ts

```
import { CanActivate } from '@angular/router';
```

**CanActivate**-interfacet importeres. Interfacet implementeres i klassen og **canActivate**-metoden defineres.

```
export class VagtGuard implements CanActivate {
 canActivate() {
 return true;
 }
}
```

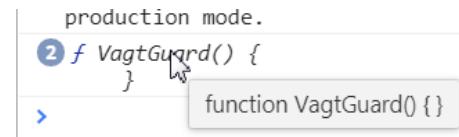
## Console.log

Med en simpel console.log(VagtGuard) kan man se funktionen blive kaldt.

```
canActivate() {
 console.log(VagtGuard);
 return true;
}
```

Danmark Sverige

Sverige Component



En given Guard kan tilføjes 1 eller flere Routes i module.ts. En Guard er derfor først koblet op mod en bestemt Route, når dette er angivet i module.ts. Guarden er en ren service, som alene returnerer true/false.

## Implementering af Guards som Service

En Service implementeres via Constructoren. Den tidlige VagtGuard bibeholdes fuld indtagt. Nu laves der en ny Guard, som rent implementeres som en Service. Denne Guard angiver om brugeren har adgang eller ej. Guarden oprettes således:

**ng g g adgang\_**

Ovenstående er i virkeligheden en forkortelse for

**ng generate guard adgang\_**

Herefter laves en Service (meget simpel) - for at lave et simpelt hello world eksempel).

**ng g s Adgang\_**

For eksemplets skyld er denne Service simpel. Den indeholder én funktion:

**adgang.service.ts**

```
import { Injectable } from '@angular/core';

@Injectable()
export class AdgangService {
 testLogin():boolean {
 return true;
 }
}
```

I Guard'en implementeres servicen i constructoren, og der spørges på værdien af testLogin()-funktionen.

**adgang.guard.ts**

```
canActivate():boolean {
 if (this.adgangService.testLogin()) {
 return true;
 } else { return false};
}
```

En Guard er en klasse. Men en Guard er ikke et komponent. Derfor kan en Guard ikke indeholde en @Component-dekorator, og Component kan ikke importeres (giver fejl).

AdgangService skal importeres i service.ts - ellers kan den ikke refereres i constructoren.

Den skal også importeres i module.ts og her angives under providers (sammen med Guard's).

## Deactivating Routes

Med **Deactivating** kan man bestemme, at en bruger ikke må forlade en side - klassisk hvis brugeren ikke har gemt sine indtastninger.

```
module.ts
{
 path: 'sverige',
 canActivate: [VagtGuard, Adgan
 canDeactivate: [ForladGuard],
 component: SverigeComponent
},
```

I module.ts tilføjes canDeactivate-guarden. Husk at indsætte den under providers også.

I sverige.component.ts indsættes et tekstfelt:

```
sverige.component.ts
<div class='sverige'>Sverige Component

<input type="text" (change)="isDirty = true" />
Ændret? : {{isDirty}}
</div> `,
```

Ved ændringer i feltet vil propertien **isDirty** ændres til true. Bemærk at der ikke direkte kaldes nogen metode. Alligevel indeholder sverige.component.ts en metode, som bruges til tjek af ændringer.

```
export class SverigeComponent {
 isDirty: boolean = false;
 canDeactivate(): boolean {
 return !this.isDirty;
 }
}
```

Herefter laves en Guard (ForladGuard). Her er flere interessante ting. For det første importeres **CanDeactivate** ( modsat forrige eksempel, hvor CanActivate blev importeret).

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udriftning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 14 Navigation Guards

For det andet importeres det komponent, der skal laves tjek på (her SverigeComponent). Derfor ser import-linjerne således ud:

```
import { Injectable } from '@angular/core';
import { CanDeactivate } from '@angular/router';
import { SverigeComponent } from './sverige.component';
```

CanDeactivate implementeres med SverigeComponent.

```
export class ForladGuard implements CanDeactivate<SverigeComponent>
```

Med en simpel funktion spørges der på den boliske version af canDeactivate-funktionen i `sverige.component.ts`, hvilket igen svarer til `!isDirty` (det bolisk modsatte af `isDirty`).

```
if (component.canDeactivate()) {
 return true;
} else {
 confirm('Vil du gemme dine ændringer?');
}
```

Resultatet er nu, at der kommer en confirm-boks på skærmen, hvis der har været ændringer i `sverige.component.ts`.



Der mangler naturligvis en submit-knap, som gemmer data.

# Kapitel 15 Pipes

## Hvad er en Pipe?

I AngularJS blev Pipes kaldt for filtre. Pipes omformer data til noget andet. Det kendes også fra PowerShell m.m.

```
template: `<p>Personen er født {{ cpr | date }}</p>`
```

Her vises cpr (fødselsdag). I stedet for at liste cpr direkte som {{cpr}} laves en pipe med **| date**. Hermed formateres og vises cpr som en dato. Man har langt mere detaljeret mulighed for at definere en dato, f.eks. **12. november 2016**.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Brug af indbyggede Pipes i Angular

Der er mange indbyggede pipes i Angular:

- ➡ DatePipe
- ➡ UpperCasePipe
- ➡ LowerCasePipe
- ➡ PercentPipe
- ➡ CurrencyPipe

Flere af disse kan tage parametre. F.eks. kan en dato vises som:

```
{ { cpr | date:"dd. MM yyyy" } }
```

Figur 128

Pipes kan **chaines** (kædes).

```
{ { cpr | date | uppercase} }
```

Figur 129

## Eksempler på andre indbyggede pipes

se alle pipes og detaljer her: <http://voidcanvas.com/angular-2-pipes-filters/>

## Betingelser i en interpolation

Directives (\*ngIf) kan benyttes til betingelser. Men de kan også laves direkte i en interpolation:

```
{{finished?'færdig':'ikke færdig'}}
```

Man kan nemt lave en betingelse i en interpolation. Finished er en bolisk værdi. Er den *true* vil interpolationen vise **færdig**. Er den boliske værdi *false*, vil interpolationen vise **ikke færdig**

## Custom Pipes

Man kan lave egne pipes. En Pipe er en selvstændig fil på linje med et komponent eller et directive.

Benytter man CLI, kan man med fordel benytte følgende kommando til at oprette en pipe:

**Ng generate pipe MinPipe**

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Her er et eksempel:

```

1 import { Component } from '@angular/core';
2 /*Custom Pipes: valutaberegner.
3 Skriver tal/7 hvis parameter er "EU"*/
4 @Component({
5 selector: 'my-app',
6 template: `
7 <h1>Custom Pipes</h1>
8 <p>DKK/EURO: {{tal | valutaBeregner: "EU"}}</p>
9 `
10 })
11 export class AppComponent {
12 tal:number=35;
13 }

```

Figur 130

Her benyttes pipen **valutaBeregner**. Hvis parameteren beregnes skrives **tal** omregnet i Euro (divideret med 7). Hvis alt andet er valgt som parameter skrives **tal** som det er.



## Custom Pipes

DKK/EURO: 5

Figur 131

Pipen er en selvstændig fil, som naturligvis skal importeres via Module:

```

1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppComponent } from './app.component';
5 import { ValutaPipe } from './custom.pipe';
6 @NgModule({
7 imports: [BrowserModule],
8 declarations: [AppComponent, ValutaPipe],
9 bootstrap: [AppComponent]
10 })
11 export class AppModule { }

```

Figur 132

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Herefter kan man lave selve pipen:

```
1 import { Pipe, PipeTransform } from '@angular/core';
2 /* Eksempel på Custom Pipe*/
3 @Pipe({name: 'valutaBeregner'})
4 export class ValutaPipe implements PipeTransform {
5 transform(value: number, valuta: string): number {
6 return (valuta=="EU"? value/7: value);
7 }
8 }
```

Figur 133

Først importeres modulerne Pipe og PipeTransform. PipeTransform er et interface og skal implementeres i klasse (linje 4)

Pipen hedder **valutaBeregner**. Erklæres i linje 3 med dekoratoren **@Pipe**.

**transform-metoden** køres (skal hedde transform - ligger som en del af PipeTransform). 2 værdier hentes ind. Value:number, som er beløbet - og valuta:string, som er valutaen (f.eks. "EU").

I linje 6 laves en simpel beregning: Hvis valuta er lig "EU" returneres beløb divederet med 7 - ellers returneres beløbet som det er.

# Kapitel 16 Indbyggede Directives

## Directives

---

Directives er instruktioner i DOM. Der findes 3 former for directives:

- ➡ **Komponenter** - directives med en template
- ➡ **Attribute Directives** - Ændrer *behaviour* af komponent, uden direkte at ændre template
- ➡ **Structural Directives** - directiver, der ændrer *behaviour* på et komponent, ved at ændre måden templaten renders/beregnes på.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Attribute Directives

Et Attribute Directive minder på mange måder om et Component.  
Det angives med dekoratoren `@directive`.

Et `@directive` er som et `@component` uden template (html-del). Den kan benyttes til at tilføje funktionalitet til andre komponenter - uden selv at vise noget på skærmen. Bruges ofte - men ikke alene - til `@hostListener` og `@hostBinding`.

Et Component udvider et Directory. Et Directory er et subset af et Component.

Et Attribute Directive ændrer ved et DOM element. Det fjerner det f.eks. ikke.

Følgende Indbyggede Attribute Directives findes:

- ➡ `ngClass`
- ➡ `ngStyle`

## Attribute Directive - ngStyle

```
template: `<h1>Attribute Directives</h1>
<p [ngStyle]="{
 'color': 'red', 'font-weight': 'bold',
 'border': '3px solid blue', 'padding': '15px'
}">ngStyle er et Attribute Directive</p>
```

Figur 134

I eksemplet ændres elementets style direkte med [ngStyle].



**ngStyle er et Attribute Directive**

Figur 135

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Attribute Directive - ngClass

```
template: `<h1>Attribute Directives</h1>
 <p ngClass="boks">her bruges ngClass</p>
``,
styles : [`.boks {color:red; border:2px blue solid;}`]
```

Figur 136

Her ændres klassen på elementet direkte. Bemærk, at man også kan tildele et array:

```
<p [ngClass]="['boks', 'cirkel']">NgClass</p>
```

Figur 137

Her er det vigtigt at have firkantede parenteser omkring [ngClass], da det er et array.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Structural Directives

Structural Directives ændrer den måde html-kode vises. Der findes følgende Structural Directives:

- ➡ \*if
- ➡ \*for
- ➡ \*switch

Hvorfor er der en \* foran if, for og switch? Fordi disse directives returnerer et array. Alternativet til \* ville være at sætte f.eks. *for* i firkantede parenteser.

\* indikerer, at der er tale om et Structural Directive, som ændrer DOM.

### \*ngIf

```
<h2>Structural Directives</h2>
<button (click)="toggleVis()">toggle komponent</button>
<p *ngIf="vises">
 Her er et komponent, som vises/skjules med *ngIf
</p>
```

Figur 138

Når man trykker på knappen kaldes funktionen *toggleVis()*. <p>-elementet nedenfor vises kun, hvis variablen *vises* er true.

Funktionen ændrer *vises* fra true til false og tilbage igen.

```
export class AppComponent {
 vises:boolean=true;
 toggleVis() { this.vises=!this.vises; }
}
```

Figur 139

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## \*NgIf-else

---

```
1 <p *ngIf="valgt"; else ikkeValgt">Du har valgt rigtigt</p>
2 <ng-template #ikkeValgt>
3 <p>Intet var valgt</p>
4 </ng-template>
```

Figur 140

Med \*NgIf-else kan man lave et klassisk if-else statement.

#ikkeValgt kaldes en **Local Reference**. **ng-template** er et directive indbygget i Angular, som kan benyttes til at markere områder i DOM - nærmest som <span> gör det i almindelig HTML.

Man kan også benytte **then**.

```
<div *ngIf="synlig;then lige else ulige">Demo af ngIf</div>
<ng-template #ulige>ulige antal</ng-template>
<ng-template #lige>lige antal</ng-template>
```

## \*ngFor

I forbindelse med arrays, objekter og andet kan man benytte \*ngFor.

```
export class AppComponent {
 lande = [
 { land: 'Norge', by: 'Oslo' },
 { land: 'England', by: 'London' },
 { land: 'Irland', by: 'Dublin' },
 { land: 'Holland', by: 'Amsterdam' },
 { land: 'Frankrig', by: 'Paris' },
 { land: 'Spanien', by: 'Madrid' }
];
}
```

Figur 141

Her er opbygget et simpelt array. Dette array kan nemt udskrives på en liste med \*ngFor.

```
<li *ngFor="let la of lande">
 {{la.land}}

```

Figur 142

Resultatet er:

- Norge
- England
- Irland
- Holland
- Frankrig
- Spanien

Figur 143

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Lokale variable sammen med \*ngFor

\*ngFor understøtter også følgende variable:

- ➡ index
- ➡ first
- ➡ last
- ➡ even
- ➡ odd

`*ngFor="let la of lande, let i=index, let ulige=odd">`  
Figur 144

variablerne vises i templatet med simpel interpolation.

```
{ {i+1} } { {la.land} }
```

Figur 145

husk: index starter med 0. Derfor *i+1*

- 1 Norge
- 2 England
- 3 Irland
- 4 Holland
- 5 Frankrig
- 6 Spanien

Figur 146

## ngNonBindable

Med ngNonBindable kan man fortæller compileren, at visse dele af koden ikke skal kompileres, men blot parses direkte til browseren.

```
Mit navn er {{ navn }}
```

Denne linje vil skrive **Mit navn er {{ navn }}** på skærmen.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 17 Komponent interaktion

Interaktion mellem komponenter kan ske på flere måder:

- Simpel Interaktion mellem parent/child
- Property Binding med @Input
- Overflytning af data med Event Binding
- Komponentinteraktion med @ViewChild

## Simpel Interaktion mellem parent/child komponent

Subkomponenter kan refereres på følgende måde:

### module.ts:

```

1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { AppComponent } from './app.component';
4 import { ChildComponent } from './child.component';
5
6 @NgModule({
7 imports: [BrowserModule],
8 declarations: [AppComponent, ChildComponent],
9 bootstrap: [AppComponent]
10 })
11 export class AppModule { }

```

Figur 147

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## component.ts (parent)

```
1 import { Component } from '@angular/core';
2 @Component({
3 selector: 'my-app',
4 template: `
5 <h1>Angular 2: Input</h1>
6 | <counter></counter>
7 `
8 })
9 export class AppComponent { }
```

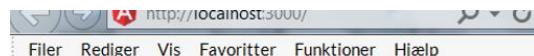
Figur 148

## child.ts (child)

```
1 import { Component } from '@angular/core';
2 @Component({
3 selector: 'counter',
4 template: `
5 <h2>Child Komponent Hello World</h2>
6 `
7 })
8 export class ChildComponent { }
```

Figur 149

## Resultat



## Angular : Input

Child Komponent Hello World

Figur 150

## Angular 4 - Udvikling af Angular apps

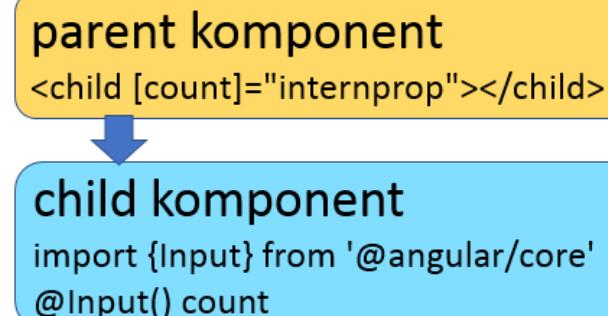
©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Overflytning af data - Property Binding

Der er 2 måder at flytte data ind i et subkomponent. Enten ved hjælp af **Property Binding** eller **Event Binding**.

### Property Binding

Property Binding kan binde værdier i parent og child sammen med @Input-dekoratoren.



Figur 151

```

1 import { Component } from '@angular/core';
2 @Component({
3 selector: 'my-app',
4 template: `
5 <h1>Angular 2: Input binding</h1>
6 <counter [count]=\"initialCount\"></counter>
7 count kaldes en 'binding' eller en 'custom property' med {initialCount}
8 `
9)
10)
11 export class AppComponent {
12 initialCount:number=15;
13 }

```

Figur 152

I **app.component** (parent) findes variablen **initialCount**, som er sat til 15. Denne værdi ønsker man at sende til **child.component**.

I linje 6 vises templetten (**<counter>**) for child komponentet. Her er indsat en 'binding' eller 'custom property' **[count]**, som sættes lig med **'initialCount'**.

Hermed kan værdien af **initialCount** sendes til child.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



```
9 export class ChildComponent {
10 @Input() count:number=0;
11 increment() {
12 this.count++;
13 }
14 decrement() {
15 this.count--;
16 }
17 }
```

Figur 153

For at importere værdien til child-elementet, skal count-variablen dekoreres med @Input(). At værdien sættes til 0 træder kun i kraft i fald, der ikke kan importeres nogen værdi.

## Lille Trick

---

Ønsker man at adskille navnene på bindingen med variablen i child.ts, kan dette gøres således:

```
app.component.ts
<counter [start]="initialCount"></counter>
```

Figur 154

```
child.component.ts
@Input('start') count:number=0;
```

Figur 155

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Overflytning af data - Event Binding

Der er 2 måder at flytte data ind i et subkomponent. Enten ved hjælp af **Property Binding** eller **Event Binding**. Her beskrives Event Binding.

### Event Binding

En event binding beskrives i en template, direkte i et html-tag.

```
<button (click)="plus()">Plus</button>
```

Figur 156

Denne kode vil kalde plus()-funktionen, når der klikkes på knappen. Den svarer reelt til onClick i klassisk JavaScript. Man kan benytte alle DOM events.

```
plus() {
 this.tal++;
}
```

Figur 157

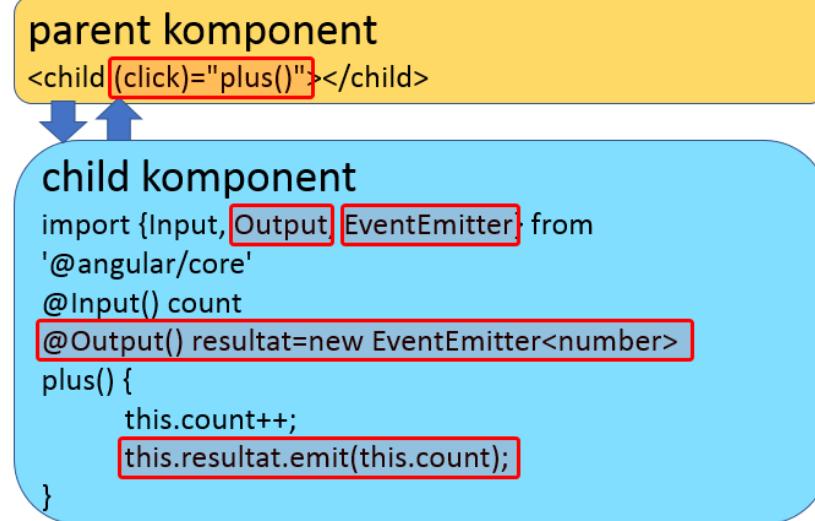
Funktionen plus kan findes i egen klasse eller i en klasse i et subkomponent.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Med **@Output** kan man sende information tilbage ud af et subkomponent. Dette gøres med en EventEmitter.



Figur 158

Man kan altså kalde en funktion i childkomponentet. Returværdien sendes tilbage til parentkomponentet med **@Output()**-dekoratoren samt EventEmitter.

#### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Samlet, simpelt eksempel

Her er en eksempel med grafisk angivelse af sammenhænge. Til venstre ses AppComponent (parent) og til højre child.component (ChildComponent).

```

import { Component } from '@angular/core';
@Component({
 selector: 'app-root',
 template: `
 <h1>Angular: @Input @Output</h1>
 <app-child1
 [fraParent]="parenttekst"
 (sendEvent) = "printBesked($event)"
 ></app-child1>
 {{returFraChild}}
 `,
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 returFraChild:string;
 parenttekst = 'Besked oppefra Parent';
 printBesked(besked) {
 this.returFraChild = besked;
 }
}

1 import { Component, OnInit, Input, Output, EventEmitter
2 @Component({
3 selector: 'app-child1',
4 template: `
5 <h2 (click)="kald()">Child1-komponentet</h2>
6 Fra Parent: {{fraParent}}
7 `
8 })
9 export class Child1Component implements OnInit {
10 @Input() fraParent: string;
11 @Output() sendEvent = new EventEmitter<string>();
12 cTekst:string="Variabel fra Child";
13 constructor() { }
14 kald() {
15 this.sendEvent.emit(this.cTekst);
16 }
17 ngOnInit() {
18 }

```

Figur 159

- @Input/@Output modtager/sender info fra/til Child
- @Input/@Output angives altid på Child-elementet - aldrig Parent-elementet
- @Input - sende info fra AppComponent til ChildComponent (husk Import Input i Child-app) - og @Input angives i Child-element.
- [fraParent]="parenttekst" -> Denne kode indsættes i child-tagget i AppComponent
- "parenttekst" er variabel i AppComponent
- -> parenttekst = 'Besked oppefra Parent';

## Angular: @Input @Output

### Child1-komponentet

Fra Parent: Besked oppefra Parent

Figur 160

#### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 17 Komponent interaktion

- [fraParent] er det objekt, der importeres i Child vha @Input  
    -> @Input() fraParent : string; (I ChildComponent)
- ➡ @output - sende info fra ChildComponent til ParentComponent (husk Import Output, EventEmitter i ChildComponent)
  - ➡ EventEmitter benyttes
  - ➡ @Output() sendEvent = new EventEmitter<string>();  
    -> Der oprettes et sendEvent-objekt
  - ➡ Med sendEvent objektet kaldes emit-metode med ønsket tekst som parameter:  
kald() {  
    this.sendEvent.emit(this.cTekst);  
}

### Angular 4 - Udvikling af Angular apps

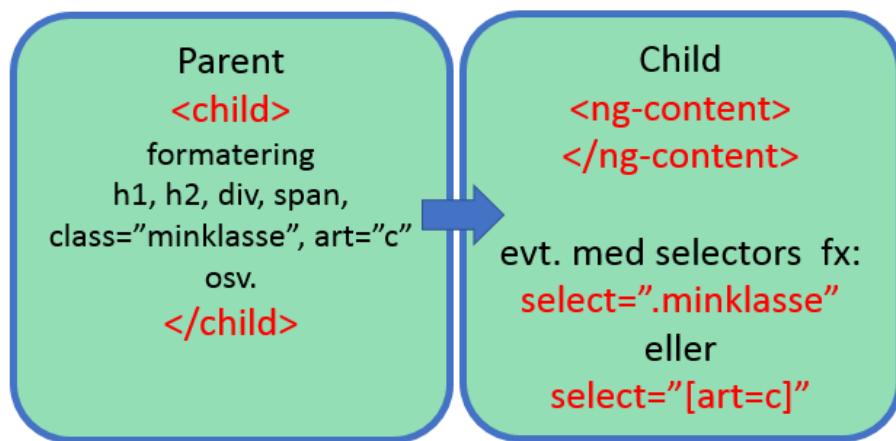
©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagere på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 18 Content Projection

Content Projection eller blot Projection kaldes også for **Transclusion**. I denne bog benyttes udelukkende begrebet **Projection**.

Et child-element har sit eget udseende - sin egen template. Herunder hører også CSS m.m.

Med projection kan man lave et child-element, men lade det være op til parentelementet, hvordan templetten skal se ud. På den måde kan man lave et fikst child-element, som kan implementeres under mange forskellige parents - og hver gang have et forskellige udseende. Det kaldes **Projection**.



Normalt er child-elementet en 100% selvstændig enhed.

```

<child>
 Normalt kommer denne tekst ikke
 med i child.. det er kun child-elementet
 der suverænt vises
</child>

```

Child-elementet er et "flettefelt" for child-selectoren i child-komponentet. Teksten i midten vises ikke på skærmen.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Det kan nemt ændres med <ng-content> i child-komponentet:

```
selector: 'child',
template: `
 <h2>child component...</h2>
 <ng-content></ng-content>

`)
```

med <ng-content> vises indholdet. <ng-content> fjernes helt - og erstattes af alt mellem <child> </child> i parent komponentet.

## Projection: Parent Component

### child component...

Normalt kommer denne tekst ikke med i child.. det er kun child-elementet der suverænt vises

ZoomTek Kurse

I child-komponentet kan man være mere specifik.

```
<ng-content select=".we"></ng-content>
<ng-content select="h3"></ng-content>
<ng-content select="[art=c]"></ng-content>
```

Her er indsat flere <ng-content>-elementer<sup>15</sup>.

ng-content kan fange alle elementer fra parent-komponentet, som opfylder visse krav. På den måde kan man i parent-komponentet formitere og style alt indholdet frit.

Rækkefølgen bestemmes suverænt på child-elementet.

---

<sup>15</sup> <ng-content> uden select vil fange alt indhold fra parent-komponentet. Man kan ikke sætte flere <ng-content>-elementer ind efter hinanden. Den første vil så vise alt - og den næste ingenting. I stedet kan man bruge Select, som udvælger f.eks. alle <h3>-elementer - eller alt med en given class-attribut osv.

## Kapitel 18 Content Projection

Projection muliggør et lille smart komponent, som kan indsættes på forskellige parents uanset parent-komponentets formatering - og child-elementet vil se ud som om, det var lavet specielt til dette parent-komponent.

I parent-komponentet kan man f.eks. indsætte følgende formatering:

```
<child>
 <h3>{navn}</h3>
 <h4 class='em'>{email}</h4>
 <h4 class='we'>{web}</h4>
 <h4 style='background-color:yellow' art='c'>{web}</h4>
</child>

export class AppComponent {
 navn:string="ZoomTek.dk";
 email:string="info@zoomtek.dk";
 web:string="www.zoomtek.dk";
}
```

På et andet parent-komponent vil html-templaten se anderledes ud.  
Og dette vil afspejles i child-komponentet. Resultatet vil være:

**Placeringen af elementerne bestemmes på child-komponentet  
Udseende/formatering af elementerne bestemmes på parent-komponentet.**

## Projection: Parent Com

child component...

[info@zoomtek.dk](mailto:info@zoomtek.dk)

[www.zoomtek.dk](http://www.zoomtek.dk)

**ZoomTek.dk**

[www.zoomtek.dk](http://www.zoomtek.dk)

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udriftning af 1 eksemplar udelukkende til eget brug er tilladt.



# Kapitel 19 Directives

## Opbygning af egne Directives

---

Benytter man CLI, kan man med fordel benytte følgende kommando til at oprette en pipe:

**Ng generate pipe MinPipe**

Her gennemgås også, hvordan man opretter Directives **manuelt**. Det er vigtig viden at have, når man skal fejlrette, flytte rundt på filer m.m.

## Decorators

---

### Class Decorators

---

- ➡ @Component
- ➡ @Directive
- ➡ @Pipe
- ➡ @Injectable

### Class Field Decorators

---

Der findes en række såkaldte Class Field Decorators:

- ➡ @Input()
- ➡ @Output()
- ➡ @HostBinding()
- ➡ @HostListener()
- ➡ @ContentChild()
- ➡ @ContentChildren()
- ➡ @ViewChild()
- ➡ @ViewChildren()

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 20 Hostlisteners og HostBinding

HostListeners og HostBinding er 2 vigtige dekoratører. De benyttes ofte i separate directives - og kobles på et komponent efterfølgende.

## HostListener

---

En HostListener kan lytte på en event. Her er et simpelt eksempel

```
import { Directive, ElementRef, HostListener }
from '@angular/core';

@Directive({
 selector: '[fokus]'
})
export class FokusDirective {
 @HostListener('click') beskedVedKlik() {
 console.log("klikket på element");
 }
}
```

Figur 161

Alle elementer med **fokus** attributten vil få vist en besked i console.loggen, når man klikker på elementet.

**HostListener kan med fordel bruges på Attribute Directives.**

Med **@HostListener** Dekoratoren kan man sætte en klassisk Listener på et hvilket som helst html-element.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

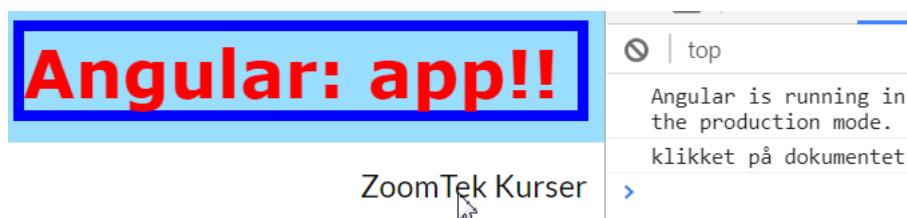


## @HostListener på globale objekter

Med denne syntaks kan man angive et globalt objekt - som f.eks. `document` eller `window`.

```
@HostListener('document:click') klikDokument(){
 console.log("klikket på dokumentet");
}
```

Når man klikker på dokumentet (alt i body-elementet), skrives der en besked i loggen.



## Info om eventen

HostListener har en optional parameter.

```
@HostListener('document:click',['$event']) klikDokument(x) {
 console.log("klikket på dokumentet",x);
}
```

Det giver en masse information i loggen.

```
klikket på dokumentet
MouseEvent {isTrusted: true
 altKey: false
 bubbles: true
 button: 0
 buttons: 0
 cancelBubble: false
 cancelable: true
 clientX: 263
 clientY: 97
 composed: true
 ctrlKey: false
 currentTarget: null
 defaultPrevented: false
 detail: 1
 eventPhase: 0
 fromElement: null
 isTrusted: true}
```

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## @HostBinding

- ➡ En HostListener kan binde en event til et element
- ➡ En HostBinding kan binde en style til et element

HostBinding benyttes ofte i @directives.

```

1 import { Component } from '@angular/core';
2
3 @Component({
4 moduleId: module.id,
5 selector: 'my-app',
6 template: `
7 <h1>Angular 2 Attribute Directive</h1>
8 <p myHighlight>Tryk her</p>
9 `})
10 export class AppComponent {
11 color: string;
12 }
```

Figur 162

Ovenfor ses app.component.ts

@direktive er lavet i en separat, der adresseres fra @NgModule

```

1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { HighlightDirective } from './highlight.directive';
4 import { AppComponent } from './app.component';
5
6 @NgModule({
7 imports: [BrowserModule],
8 declarations: [AppComponent,HighlightDirective],
9 bootstrap: [AppComponent]
10 })
11 export class AppModule { }
```

Figur 163

Ovenfor ses module.ts

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 20 Hostlisteners og HostBinding

```
1 import { Directive, HostListener, HostBinding }
2 from '@angular/core';
3
4 @Directive({ selector: '[myHighlight]' })
5 export class HighlightDirective {
6
7 private color='yellow';
8 @HostListener('click')
9 click() {
10 this.color='lightgreen';
11 }
12 @HostBinding('style.backgroundColor')
13 get bentebent() {
14 return this.color;
15 }
16 }
```

Figur 164

Grundlæggende er @Directive-filen bygget op som et Component. Her er tilføjet 2 dekoratorer: **@HostListener** og **HostBinding**.

**@HostListener** lytter som bekendt til en event.

**@HostBinding** kobler et DOM-element til en funktion eller variabel. I eksemplet virker **@HostListener** og **@HostBinding** sammen.

**@HostListener** 'lytter' efter et klik fra brugeren. **@HostBinding** forbinder baggrundsfarven (og ikke alt mulig andet) til TypeScript-variablen Color.

## Angular 2 Attribute Directive

Tryk her

Figur 165

Baggrundsfarven ændres til *lightgreen* ved klik.

## ElementRef

```
<h1 fokus>Angular: {{title}} !!</h1>
```

Her ses et **attribute directive** som hedder **fokus**. Når fokus er sat på et element, skal elementet have rød skrift og en gul kant omkring.  
Det gøres således:

Først oprettes et directive med CLI

```
ng g directive fokus
```

Det nye directive bliver oprettet, og bliver automatisk aktiveret via **module.ts**.

```
import { AppComponent } from './app.component';
import { FokusDirective } from './fokus.directive';

@NgModule({
 declarations: [
 AppComponent,
 FokusDirective
```

Selve directive-klassen tilføjes små ændringer.

```
import { Directive, ElementRef } from '@angular/core';

@Directive({
 selector: '[fokus]'
})
export class FokusDirective {
 constructor(element: ElementRef) {
 let tag=element.nativeElement.style;
 tag.color="red";
 tag.border="5px solid blue";
 }
}
```

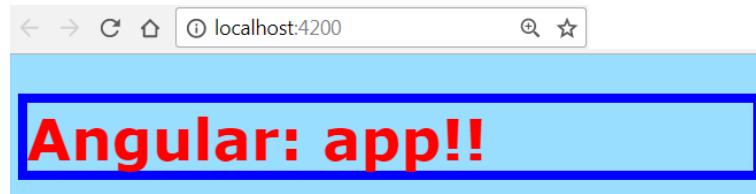
**ElementRef** importeres (blå), så det kan indsættes i constructoren som DI (grå). **element.nativeElement** (grøn) repræsenterer html-elementet og **.style** elements css.

Herefter er det en nem sag, at ændre skriftfarve og border (lilla).

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.





ZoomTek Kurser

ElementRef giver adgang til det underliggende DOM element.  
 ElementRef er derfor ofte et alternativ til @HostBinding.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 21 @ViewChild interaktion

Når en parent-klasse skal læse variable i en subklasse, findes der en række metoder. Skal man eksekvere metoder i en subklasse, skal der mere til. Her kan man benytte **ViewChild**.

Nedenfor benyttes app.component.ts og child.component.ts. **Målet er fra app.component.ts at kalde en metode i child.component.ts.**

Man indsætter child-klassen på vanlig vis gennem **app.module.ts**.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChildComponent } from './child.component';
import { AppComponent } from './app.component';

@NgModule({
 imports: [BrowserModule],
 declarations: [AppComponent, ChildComponent],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

Figur 166

I app.module.ts loades både AppComponent og ChildComponent ind.

Fra templet i app.component.ts aktiveres kald()-metoden med click. Kald()-metoden ligger i child.component.ts.

```
template: `<child (click)="kald()"></child>`
```

Det er muligt fordi:

**ViewChild** importeres i app.component.ts

```
import { Component, ViewChild } from '@angular/core';
```

Figur 168

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 21 @ViewChild interaktion

- AfterViewInit (interface) importeres i app.component.ts og implementeres i klassen.

```
import { AfterViewInit } from '@angular/core';
```

```
| export class AppComponent implements AfterViewInit
```

Figur 169

- @ViewChild-dekorator erklæres:

```
export class AppComponent implements AfterViewInit {
 | @ViewChild(ChildComponent) child: ChildComponent
}
```

Figur 170

Denne linje er opdrenningspunktet. @ViewChild kalder ChildComponent i parentesen. Endelig oprettes et objekt (child) ud fra ChildComponent.

Objektet child kan herefter benyttes til kald direkte i ChildComponent.

```
kald() {
 | this.child.hentFraChild();
}
```

Figur 171 her kaldes funktionen hentFraChild(), som ligger i child-komponentet.

Der skal ikke gøres noget i child-komponentet. Intet skal importeres - og intet skal erklæres.

```
import { Component } from '@angular/core';
@Component({
 selector: 'child',
 template: `
 <h2>Child Komponent</h2>
 <p>Klik på overskriften. Dette er en demo af @ViewChild</p>
 `
})
export class ChildComponent {
 constructor() {}
 hentFraChild() {
 alert("hentFraChild funktion er kaldet vha. @ViewChild...");
```

Figur 172

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Ikke bare til funktionskald

Man har adgang til alt i child-elementet. Herunder funktioner, variable, objekter, observables m.m. Her erklæres variablen **firma** i child.component.ts.

```
export class ChildComponent {
 firma:string="ZoomTek.dk";
 constructor() {}
 hentFraChild() {
 alert("hentFraChild fra ChildComponent");
 }
}
```

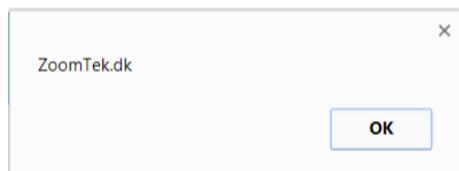
Figur 173

Med **this.child.firma** er der direkte adgang til variablen fra app.component.ts.

```
kald() {
 this.child.hentFraChild();
 alert(this.child.firma);
}
```

Figur 174

Og en alert-boks vises.



Figur 175

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## @VieBwChildren

---

@ViewChildren gør egentlig det samme som ViewChild - blot med et array af children. Man skal importere **ViewChildren** og **QueryList**.

```
import { Component, ViewChildren, QueryList, AfterViewInit } from
'@angular/core';
```

## @ContentChild og @ContentChildren

---

Disse fungerer reelt som ViewChild og ViewChildren. @ViewChild er implementeret ved angivelse af @viewchild i parent-komponentet.

@contentChild virker på de child-komponeneter som **projekteres** ind i komponentet.

## Kapitel 22 Binding

Binding vil sige, at man kan synkronisere data mellem templatens og klassen. Altså populært sagt HTML-koden (templatens), som brugeren kan se i form af felter - og variable og objekter i klassen, som hører til ethvert komponent.

Man kan lave Binding på hele 4 måder.

### Simpel interpolation (metode 1)

Denne teknik er gennemgået tidligere. Her er et simpelt eksempel:

```
@Component({
 selector: 'app-root',
 template: `
 <h1>Angular: {{title}}!!</h1>
 <p> {{navn}}</p>
 `,
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 title:string = 'Hello World';
 navn:string='Søren Anderson';
}
```

Figur 176

### Betingelse i interpolation

```
{{finished?'færdig':'ikke færdig'}}
```

Man kan nemt lave en betingelse i en interpolation. Finished er en bolisk værdi. Er den *true* vil interpolationen vise **færdig**. Er den boliske værdi *false*, vil interpolationen vise **ikke færdig**

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## ngModel og ngForm (metode 2)

Metode 2, 3 og 4 kræver, at **FormsModule** bliver importeret i **module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
 declarations: [AppComponent],
 imports: [BrowserModule, FormsModule],
 providers: [],
 bootstrap: [AppComponent]
})
```

Figur 177

Hvis html-koden er en form - det vil sige omkrænset af et `<form>`-tag, kan man lave et formobjekt.

```
<form novalidate #f="ngForm">
 Html-kode her.....
```

Figur 178

Her oprettes en variabel **f**, som repræsenterer formen. Alle input-elementer med ngModel angivet vil blive underlagt dette formobjekt.

```
Navn: <input
 type="text"
 name="navn"
 ngModel>
```

Figur 179

Form-objektet er ikke det samme som klassen. Data/variable fra formobjektet kan kun bruges i formen og kan ikke trækkes ned i klassen.

Man kan bruge formobjektet frit i selve formen - men ikke i klassen.  
Så der er ikke tale om 2-way binding.

Angular 4 - Udvikling af Angular apps

Man kan interpolere indholdet af formobjektet således:

```
{} f.value | json {}
```

Figur 180

f er naturligvis form-objektet. Json er piping af objektet til JSON-format.

Navn

```
{ "navn": "Aage Brodtgaard" }
```

Figur 181



## [ngModel] (metode 3)

Denne metode kræver **FormsModule** (indsættes via Module.ts), som metode 2.

Denne metode kræver **ikke nødvendigvis <form>-tags** i html-koden ( modsat metode 2).

```
Email:<input
| | | | type="email"
| | | | name="email"
| | | [ngModel]="bruger.konto.email">
```

Figur 182

Her er defineret et email-felt med [ngModel]. Med denne metode kan jeg henvise til værdier i klassen/interfacet. Den svarer reelt til

```
{{bruger.konto.email}}
```

```
8 export class AppComponent {
9 bruger:Bruger = {
10 navn: '',
11 konto: {
12 email: 'soren@zoomtek.dk',
13 confirm: ''
14 }
15 };
16}
```

Figur 183

Variablen **bruger.konto.email** får her default værdien 'soren@zoomtek.dk'. Denne vises i email-feltet i templetten, når siden loades op.

## ([ngModel]) - klassisk 2-way binding (metode 4)

Denne metode kræver **FormsModule** (indsættes via Module.ts), som metode 2.

Denne metode kræver ikke nødvendigvis <form>-tags i html-koden ( modsat metode 2).

```
Bekræft: <input
 type="email"
 name="confirm"
 [(ngModel)]="bruger.konto.confirm">
```

Figur 184

I syntaks svarer den til metode 3. Der er blot [()] rundt om ngModel.

Denne metode giver ægte 2 way binding. Det vil sige, at templet og klassen synkronisere værdierne i realtid.

Følgende 2 måder at benytte ngModel er ens. Den nederste er klart den korteste.

[ngModel]="email" (ngModelChange)="email = \$event"

og

[(ngModel)]="email"

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Samlet eksempel

```

1 <form novalidate #f="ngForm">
2
3 Navn: <input
4 type="text"
5 name="navn"
6 ngModel>
7 Email: <input
8 type="email"
9 name="email"
10 [ngModel]="bruger.konto.email">
11 Bekræft: <input
12 type="email"
13 name="confirm"
14 [(ngModel)]="bruger.konto.confirm">
15 <button type="submit">Godkend !!</button>

```

Figur 185

Her ses de 3 felter, hvor henholdsvis ngModel, [ngModel] og [(ngModel)] benyttes.

```

{{ f.value | json }}
{{bruger.navn}}
{{bruger.konto.email}}
{{bruger.konto.confirm}}

```

Figur 186

Interpolation kan laves direkte i html-koden. Den første {{f.value | json }} udskriver hele formobjektet i json-format.

Formobjektet vedligeholdes for alle felter, der har én af de 3 ngModel opsætninger.

Formobjektet ændres i realtid, mens der tastes. Formobjektet kan IKKE overføres til klassen.

Navn:  Email:

```

1) Form-object (JSON)
 "navn": "Sven",
 "email": "soren@zoomtek.dk",
 "confirm": ""
}

```

Figur 187

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 23 Events

Events benyttes således i Angular:

```
<button (click)="onGivBesked()">Klik her</button>
```

Figur 188

Her er en knap (button). Når man trykker på knappen (click) kaldes metoden 'onGivBesked()'. Metoden kan hedde hvad som helst.

```

1 import { Component } from '@angular/core';
2
3 @Component({
4 selector: 'app-root',
5 template: `<h1>Angular: Events</h1>
6 <button (click)="onGivBesked()">Klik her</button> {{tekst}} `
7 })
8 export class AppComponent {
9 tekst:string='';
10 tal:number=0;
11 onGivBesked() {
12 this.tal++;
13 this.tekst="Du har trykket på knappen "+this.tal+" gange";
14 }
15 }
```

Figur 189

Metoden findes i klassen (linje 11-14). Tal plusses med 1 og tekst propertien bliver sat. Værdien af tekst bliver interpoleret til html-templaten. Resultatet er således:

## Angular: Events

[Klik her](#) Du har trykket på knappen 5 gange

Figur 190

I stedet for at kalde en metode kan man lægge logik (kode) direkte ind efter `(click)='logik/kode'`. Det anbefales dog at holde logik/kode til klassen - og ikke templatene.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## \$event

---

```
<input type='text' (input)='opdaterFelt($event)'>
```

Figur 191

Her indsættes et simpelt felt. (input) er en event, som reagerer på hvert tastetryk. Når man trykker på en tast kaldes metoden 'opdaterFelt(\$event)'.

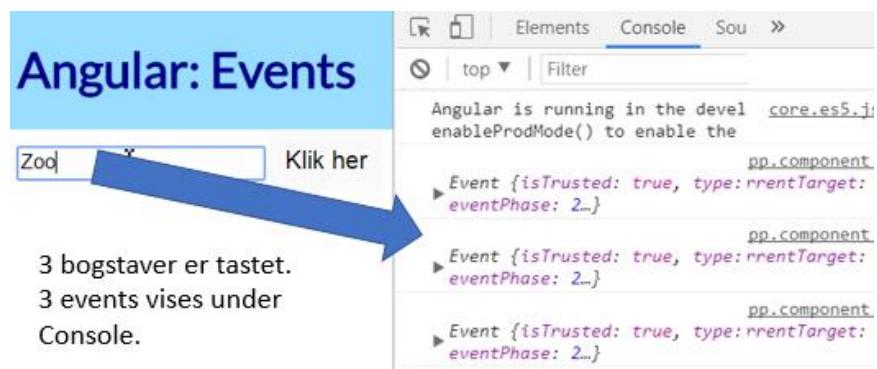
Det er muligt at bruge **\$event**. Dette objekt indeholder en lang række informationer om det element eller det event, som der arbejdes med. På et felt er det muligt at finde værdien af feltet - på (click) på en knap, er det muligt at finde cursorens position på skærmen og så videre.

```
opdaterFelt(event:any) {
 console.log(event);
}
```

Figur 192

Selve metoden modtager \$event (lokal variabel event). Den sendes til console.loggen. Når applikationen køres kan man laves en Inspection/undersøgelse<sup>16</sup>. Vælg Konsol.

Hver gang man taster kan man se en Event:



Figur 193

Udfolder man en af de 3 events kan man igen udfolde Target og igen Value, kan man se værdien 'Zoo'.

<sup>16</sup> I Google Chrome højreklikkes på skærmen - hvorefter der vælges Undersøg eller Inspect.

## Kapitel 23 Events

```

 title: ""
 translate: true
 type: "text"
 useMap: ""
 validationMessage: ""
 ▶ validity: ValidityState
 value: "Zoo" Zoo
 valueAsDate: null
 valueAsNumber: NaN
 ▶ webkitEntries: Array(0)
 webkitdirectory: false
 width: 0
 willValidate: true
 ▶ __zone_symbol__inputfalse: Arra

```

Figur 194

Værdien af feltet kan returneres således:

```

opdaterFelt(event:Event) {
 this.tastet=(<HTMLInputElement>event.target).value;
}

```

Figur 195

Der laves en typecast med `<HTMLInputElement>`. Sender man `$event` afsted på (click) får andre parametre:

```

MouseEvent {isTrusted: true,
▶ screenX: 221, screenY: 166, clientX:
 220, clientY: 100...}

```

Figur 196

Derfor kan man spørge på:

```

onGivBesked(event) {
 console.log(event);
 alert(event.screenX);
 this.tal++;
 this.tekst="Du har trykket
}

```

Figur 197

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## host

---

```
3 @Component({
4 selector: 'app-root',
5 template: '<h1>Angular Events</h1>
6 <input type="text" (input)="updateFelt($event)
7 <button (click)="onGivBesked($event)">HJER
8 <div style="position: absolute; top: 0; left: 0;">
9 host: { "(mousemove)": "onGivBesked($event)" }
10 })
```

Figur 198

Her angives host som metadata-parameter. Når musen flyttes inden for body-elementet vil onGivBesked(\$event) blive kaldt.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagere på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 24 Template Syntaks

## Interpolation

---

Med interpolation kan man vise data i html-templetten. Her er listet en række eksempler.

### Simpel interpolation

---

Interpolation laves med dobbelt-tuborg: `{{navn}}`.

Figur 199

### Interpolation som html-attribut

---

Interpolation kan bruges i html-tags: `<a href="{{link}}>Kurser</a>`

Figur 200

### Interpolation med beregning

---

Interpolation kan lave beregninger: `2 + 2 = {{2+2}}`

Figur 201

Man kan ikke benytte `++`, `--`, `+=` og tilsvarende.

### Interpolation med metodekald

---

Interpolation kan kalde funktioner: `2+2+4 = {{2+2+fire()}}`

Figur 202

### Interpolation med if-statement

---

Interpolation kan bruge if: `1==1?'ja':'nej' = {{1==1?'ja':'nej'}}`

Figur 203

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## (1) Interpolation

Interpolation laves med dobbelt-tuborg: Anderson.

Interpolation kan bruges i html-tags: [Kurser](#)

Interpolation kan lave beregninger:  $2 + 2 = 4$

Interpolation kan kalde funktioner:  $2+2+4 = 8$

(du kan ikke bruge `++`, `--`, `+=` og tilsvarende)

Interpolation kan bruge if: `1==1?'ja':'nej' = ja`

Figur 204

## Property Binding

Property Binding kan benyttes til flere ting.

```
<input [hidden]="skjult"
 [style.color]="farve"
 [value]="navn"
 type="text"
 />
```

Figur 205

Her ses et input-element.

**[hidden]** er sat til "skjult". Skjult er en member variabel. Har den værdien true er input-elementet skjult (false og det er synligt).

**[style.color]** giver blå font-color i feltet.

**[value]** sætter værdien af feltet til member variablen *text*.

Ser man i klassen er følgende værdier sat:

```
export class AppComponent {
 navn:string="Anderson";
 farve:string="blue";
 skjult:boolean=false;
```

Feltet vil derfor være synligt, have blå skrift og indeholde en default værdi på 'Anderson'.

### (2) Property Binding

properties er alt i firkantede parenteser (hidden, style.color, value)

disse bliver bindet til højre for =... som igen er en variabel i klassen.

Fx har skjult værdien true/false, farve værdien blue og value værdien Anderson.

Figur 206

Værdierne kan ændres dynamisk. Enten ved metoder, som kaldes fra events eller fra andre komponenter med f.eks. @input.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



# Kapitel 25 Animations

## 1) Animation er taget ud af @angular/core

Med Animations kan man lave bevægelse på skærmen. Få ting til at flyve væk og dukke op.

Animations har eksisteret siden AngularJS, men har undergået en stor forandring. I det følgende laves en kort, noget overfladisk Hello World gennemgang af Animation APIlet.

## Indsæt script-reference i index.html

Først skal man indsætte følgende script-reference list før </body> i index.html:

```
<script src="/assets/vendors/web-animations.min.js"></script>
```

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Import

Følgende packages skal importeres fra '@angular/animations'. **17**

```
import { trigger, style, transition, animate, group }
 from '@angular/animations';
```

Figur 207

Importen skal ske i app.component.ts. Ligeledes skal man tilføje modulet BrowserAnimationsModule i module.ts:

```
import { BrowserAnimationsModule }
 from '@angular/platform-browser/animations'
```

Figur 208

**17** Forskel på Angular 2 og 4: I Angular skal man importere fra '@angular/core'. I Angular 4 skal man importere fra '@angular/animations'. Animationer er blevet udskilt fra 'core' i Angular 4.

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



# Kapitel 26 Template-driven forms (FormsModule)

Der findes flere måder at lave forms (formularer) på i Angular:

- ➡ **Template-driven forms (FormsModule)**  
Forms som er opbygget udelukkende i templetten - altså ved hjælp af HTML5-elementer. Det er en nem måde at udvikle simple forms.
- ➡ **Model-driven (ReactiveFormsModule)**  
Sådanne forms bruger også HTML5 elementer, men tilføjer et yderligere lag af funktionalitet, da formen skal defineres igennem ens klasse. Dermed kan elementerne i formen tilgås med Typescript.

Der er tale om forskellige indgangsvinkler. Det kan virke overvældende at overskue alle måder/klasser. De fleste udviklere vil i praksis benytte blot én af klasserne til alle forms-løsninger.

For at benytte forms skal man vælge mellem at importere **FormsModule** eller **ReactiveFormsModule**. I praksis gøres dette i module.ts-filen:

I praksis vil man ikke kombinere FormsModule med ReactiveFormsModule. Ovenfor ses opsætning af FormsModule i module.ts.

Forms arbejder direkte sammen med html-controls. Typisk <input>-elementer. Man kan benytte både html4 og html5 - forskellen ligger i, om id-attributten bruges eller ej. Angular er som sådan ligeglads.

```
<input type="text" name="kundenavn" id="knavn">
<input type="text" name="kundenavn">
```

Figur 209

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Template-driven forms

Template-driven forms tager direkte udgangspunkt i HTML5 input elementer.

```

1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-
3 import { FormsModule } from '@angular/forms';
4 import { AppComponent } from './app.component';
5
6 @NgModule({
7 imports: [BrowserModule, FormsModule],
8 declarations: [AppComponent],
9 bootstrap: [AppComponent]
10 })
11 export class AppModule { }

```

Figur 210

Her er et eksempel.

Navn:	<input [(ngmodel)]="kunde.navn" name="navn" type="text"/>  	Navn:: Peter Madsen
Adresse:	<input [(ngmodel)]="kunde.adresse" name="adresse" type="text"/>  	Adresse: Ydre Skovvej 14
Postnummer:	<input [(ngmodel)]="kunde.postnummer" name="postnummer" type="text"/>  	Postnummer: 3000
By:	<input [(ngmodel)]="kunde.by" name="by" type="text"/>  	By: Helsingør

Figur 211

Dataene i felterne er leveret af klassen (kan ikke ses på kodeeksemplet) igennem [(ngModel)].

```
<input type="text" name="name" ngModel>
```

NgForm er en del af FormsModule. Skrives ngForm på skærmen som {{f.value | json}} vil denne være tom, med mindre man angiver ngModel eksplisit for de enkelte felter. NgForm skaber automatisk en FormGroup. Derudover skal name-attributten angives som nøgle/opslag til feltet.

Ønsker man at håndtere formdata, skal man tilføje nogle få attributter i Angular.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltageere på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 26 Template-driven forms (FormsModule)

```

1 <form novalidate #f="ngForm" (ngSubmit)="sendData(f)">
2 Navn:
3 <input type="text" name="navn"
4 [(ngModel)]="kunde.navn">

5 Adresse:
6 <input type="text" name="adresse"
7 [(ngModel)]="kunde.adresse">

8 Postnummer:
9 <input type="text" name="postnummer"
10 [(ngModel)]="kunde.postnummer">

11 By:
12 <input type="text" name="by"

```

Figur 212

For at benytte ngForm, skal man normalt ikke gøre noget. Benyttes beta-versioner af Angular 2 kan en import være nødvendig (læs grå boks).

~~import { NgForm } from '@angular/forms';~~

Figur 213

Det er ikke nødvendigt at importere ngForm i Angular 4. I Angular 2 skal den ifølge dokumentationen heller ikke importeres. NgForm importeres automatisk som en del af **FormsModule**. I de helt første udgaver af Angular 2, har forfatter dog oplevet, at en import var nødvendig.

Når FormsModule importeres, oprettes der automatisk en instance af ngForm. Endvidere oprettes automatisk en **FormGroup**. Denne skal heller ikke importeres særskilt, når FormsModule benyttes.

Endelig skal der være en metode, som kan foretage sig noget, når formen submittes.

```

sendData(form:NgForm) {
 alert('formen er nu submitted');
}

```

Figur 214

Metoden skal ligge i app.component.ts i klassen

### Angular 4 - Udvikling af Angular apps

## Kapitel 26 Template-driven forms (FormsModule)

```
9 export class AppComponent {
10 kunde:Kunde = {
11 navn: 'Peter Madsen',
12 adresse: 'Ydre Skovvej 14',
13 postnummer: '3000',
14 by: 'Helsingør'
15 konto: {
16 brugernavn: 'oren@zoomtek.dk',
17 password:
18 }
19 };
20 sendData(form:NgForm) {
21 alert('formen er nu submitted');
22 }
23 }
```

Figur 215

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Validering af felter - Valid, Dirty, Touched

```
<form novalidate #f="ngForm" (ngSubmit)="sendData(f)">
 <button type="submit" [disabled]="f.invalid">Godkend !!</button>
</form>
```

Figur 216

Man tilføjer nemt valideringstjek med `[disabled]="f.invalid"`.

Med denne teknik benytter man `type`-attributten på html5 input-elementet direkte.

```
[disabled]!="f.valid||!f.dirty"
```

Figur 217

Dirty betyder, at et felt har været redigeret. `f.dirty` forhindrer submit, hvis blot et felt ikke er ændret.

Et email-felt med `type="email"` har forfatter ikke kunnet få til at virke med `f.valid`. Selv om der tastes **ulla** eller andet, der tydeligvis ikke er en email-adresse godkendes formen.

## Validering med Template-driven forms

Validering på template-driven forms er alene lavet med standard html5.

Her er et par eksempler med Regular Expressions.

**Password:**

```
<input type="text" name="password" pattern=".{4,5}"
ngModel>

```

Figur 218

Password feltet må indeholde 4 eller 5 karakterer - og det skal udfyldes.

**Postnummer:**

```
<input type="number" name="postnummer"
pattern="^\d{4}$"
ngModel>

```

Figur 219

Her **skal** postnummer feltet indeholde 4 tal.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Feltet kan valideres med Regular Expressions. Bemærk at RegEx-teknikker som Branch Reset Groups og Look-around ikke virker i HTML5. Kontakt ZoomTek. Deltag eventuelt i vores Regular Expression kursus.

## Validering pr. Felt

### Postnummer:

```
<input type="number" name="postnummer"
pattern="^\d{4}$"
ngModel required>
```

Figur 220

Her er et postnummer felt. **Typen** er number. Det betyder, at brugeren ikke kan indtaste tekst, men kun tal. Feltet er en del af **ngForm**, da **ngModel** er angivet for feltet. Feltet har en **name**-attribut (postnummer), som handler.

Der er indsat et **pattern** med et Regular Expression udtryk, som kun lovliggør **4 cifrede postnumre**. Udtrykket kunne være udvidet til ikke at tillade tomme værdier. I stedet er attributten **required** indsat.

```
Valid? {{f.form.controls.postnummer?.valid}}
Dirty? {{f.form.controls.postnummer?.dirty}}
Touched? {{f.form.controls.postnummer?.touched}}
```

Figur 221

Indsættes denne kode **18** kan man spørge på om postnummeret er

- |                |                                                                             |
|----------------|-----------------------------------------------------------------------------|
| <b>Valid</b>   | Overholder type, pattern og required (samt evt. andre attributter).         |
| <b>Dirty</b>   | Har der været tastet i feltet (også selv om indtastningen er slettet igen). |
| <b>Touched</b> | Har feltet været markeret. Aktiveres først, når feltet fjernes.             |

---

**18** ? (spørgsmålstege) i koden kaldes Elvis operatoren. Helt præcist betyder den: Kald kun propertien til højre for ? (f.eks. valid), hvis propertien til venstre for ? ikke er *null*.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udriftning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 26 Template-driven forms (FormsModule)

Postnummer:

Valid? true

Dirty? true

Touched? true

Figur 222

Feltet er **Validt**. Feltet er **Dirty** og har ændret sin oprindelige (tomme) værdi. Feltet er **Touched**, fordi markøren har været placeret i feltet.

```
<div *ngIf="f.form.controls.postnummer?.valid" >
 Postnummer er korrekt
</div>
```

Figur 223

Indsættes et `*ngIf` directive kan man skrive en besked på skærmen, hvis postnummeret er korrekt. Skrives `invalid` i stedet for valid, vil man spørge modsat. `*ngIf` opdateres, mens man skriver i feltet.

Postnummer:

Postnummer er korrekt

Figur 224

## Formatering af felter ud fra validering

Postnummer:

Figur 225

Feltet får nu automatisk en rød baggrundsfarve, når postnummeret tastes forkert (sker dynamisk mens der tastes).

Feltet wrappes ind i et div-element (andre elementer kan også lavet tricket).

```
div-element uden om felt {
 <div [ngClass]="{'red': f.form.controls.postnummer?.invalid}">
 <input type="number" name="postnummer"
 pattern="^\d{4}$"
 ngModel required>
 </div>
```

## Angular 4 - Udvikling af Angular apps

## Kapitel 26 Template-driven forms (FormsModule)

Figur 226

I div elementet indsættes en [ngClass] med denne værdi:

```
[ngClass]="{'red': f.form.controls.postnummer?.invalid}"
```

Figur 227



## Template Model Binding

---

Controls i forms kan bindes 1-way eller 2 way.

**1-way binding** opnås således:

```
<input type="text" name="navn" [ngModel]="knavn">
```

Figur 228

knavn refererer til en klasse-variabel, som kan være defineret således:

```
knavn:string="Niels Hansen";
```

Figur 229

Resultatet er således en default-værdi i feltet.

2-way binding fås med "banana-box" syntaksen:

```
<input type="text" name="navn" [(ngModel)]="knavn">
```

Figur 230

Fjern formGroup, formGroupName, formControl og formControlName fra formen. Disse hører til ReactiveFormsModule

## NgModelGroup

I tilfælde af mange felter, der skal vises på en form, hvor felterne næsten hedder det samme, kan det være et mareridt at holde styr på felter. Hvis data er bygget op som komplekse objekter, som angivet nedenfor, kan man hurtigt løbe ud i store problemer med lange referencenavne i ens kode:

```
{
 "kunde": {
 "nr": "001",
 "navn": "Dansk Kamel Import",
 "adresse": "Grusvejen 14",
 "by": "Padborg"
 },
 "kontaktperson": {
 "navn": "Finn Hansen",
 "tlf": "67450013",
 "email": "fh@dki.dk",
 },
 osv
}
```

Figur 231

For at undgå navnesammenfald ender man hurtigt ud i html-kode altså:

```
<input type="text" name="kontaktperson_navn" id="kontaktperson_navn">
```

Figur 232

NgModelGroup gør det hele lidt nemmere:

Ved at benytte <fieldset> og ngModelGroup bliver koden nemmere at opsætte. Nedenfor er ngModelGroup benyttet. Felter (input-elementer) i samme ngModelGroup kan behandles samlet.

Navn:	<input type="text"/>
Adresse:	<input type="text"/>
Postnummer:	<input type="text"/>
By:	<input type="text"/>

Figur 233

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 26 Template-driven forms (FormsModule)

```
<fieldset>
 Navn: <input type="text" name="navn" ngModel required/>
 Adresse: <input type="text" name="adresse" ngModel required/>
 Postnummer: <input type="text" name="postnummer" ngModel required/>
 By: <input type="text" name="by" ngModel required/>
</fieldset>
{{f.value | json}}
```

Figur 234

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 26 Template-driven forms (FormsModule)

Nederst udskrives formobjektet (f.value) i json-format:

```
{
 "navn": "",
 "adresse": "",
 "postnummer": "",
 "by": ""
}
```

Figur 235

```
Navn: <input type="text" name="navn">
<fieldset ngModelGroup="adresseinfo">
 Adresse: <input type="text" name="adresse">
 Postnummer: <input type="text" name="postnummer">
 By: <input type="text" name="by">
</fieldset>
```

Figur 236

Fieldset-elementet flyttes så det wrapper adresse, postnummer og by - men ikke navn. Herefter tilføjes attributten  
**ngModelGroup="adresseinfo"**

Herefter er formobjektet ændret til:

```
{
 "navn": "",
 "adresseinfo": {
 "adresse": "",
 "postnummer": "",
 "by": ""
 }
}
```

Figur 237

Herefter kan man direkte benytte ngModelGroup til validering.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 26 Template-driven forms (FormsModule)



```
<div [ngClass]="'red': f.form.controls.gruppe?.invalid">
<fieldset ngModelGroup="gruppe">
Postnummer:
<input type="number" name="postnummer"
pattern="^\d{4}$"
ngModel required>
Husnummer:
<input type="number" name="husnummer"
pattern="^\d{1,2}$"
ngModel required>
</fieldset>
</div>
```

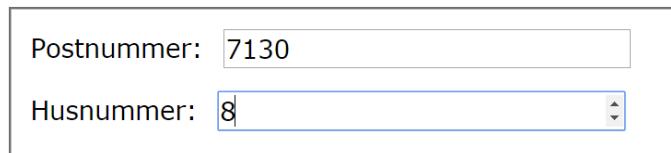
Figur 238

Postnummer og husnummer er wrappet ind i `<fieldset>`-elementet med `ngModelGroup`-attributten **gruppe**.

`<Fieldset>`-elementet er wrappet ind i et `<div>`-element. Det er det samme element, som vist tidligere. Forskellen er blot, at der nu refereres til **gruppe** i stedet for navnet på et konkret felt.

Boksen (div) bliver rød, hvis **postnummeret** ikke er 4 tal **eller** **husnummeret** ikke er på 1 eller 2 tal.

Korrekt postnummer og husnummer:



Postnummer:	7130
Husnummer:	8

Figur 239

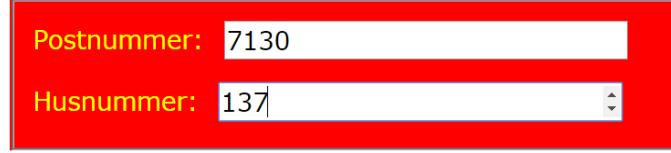
Forkert postnummer og korrekt husnummer:



Postnummer:	71305
Husnummer:	8

Figur 240

Korrekt postnummer og forkert husnummer



Postnummer:	7130
Husnummer:	137

Figur 241

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 27 Model-driven forms (Reactive)

Der findes flere måder at lave forms (formularer) på i Angular:

→ **Template-driven forms (FormsModule)**

Forms som er opbygget udelukkende i templanen - altså ved hjælp af HTML5-elementer. Det er en nem måde at udvikle simple forms.

→ **Model-driven (ReactiveFormsModule)**

Sådanne forms bruger også HTML5 elementer, men tilføjer et yderligere lag af funktionalitet, da formen skal defineres igennem ens klasse. Dermed kan elementerne i formen tilgås med Typescript.

I dette kapitel er **fokus** på den sidste type - **Model-driven** formularer bygget op omkring **ReactiveFormsModule**.

Reactive forms kaldes også for model-driven forms. Med Reactive forms undgår man brug af directives og attributter som ngModel, required m.fl.

I stedet for at lade Angular understøtte ens form, benyttes de underliggende API'er direkte til udvikling af formen.

Man laver altså formen direkte i klassen. Man laver en instance af formen, som man kan programmere direkte. Det giver en dybere og mere direkte tilgang til form, felter m.m. Samtidig gør det unit testing nemmere. Alt kode, data, logik m.m. er samlet i en klasse i stedet for på diverse html-templates.

Fordelen ved denne metoder er, at man kan tilgå data programmeringsteknisk. Det giver en række andre muligheder, end dem HTML5 tilfældigvis giver.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Den Simple Form

```
<pre>
 Fornavn: <input type="text">
 Efternavn: <input type="text">
 Adresse: <input type="text">
 Postnr.: <input type="text">
 By: <input type="text">
 Kommentar: <input type="text">
 <button type="submit">Godkend</button>
</pre>
```

Figur 242

Her er en form med nogle simple felter.

Fornavn:  
Efternavn:  
Adresse:  
Postnr.:  
By:  
Kommentar:


Godkend

Øverst importeres en række interfaces og moduler, som sidenhen bruges til at binde html-input-elementerne sammen med klassen.

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, FormControl, Validators} from
'@angular/forms';
```

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## FormGroup og FormControl

Html-koden omkrænses af form-elementer:

```
<form [formGroup]="navneForm">
<h1>Angular - Model-driven Reacti
<pre>
 Fornavn: <input type="text">
 Efternavn: <input type="text">
 Adresse: <input type="text">
 Postnr.: <input type="text">
 By: <input type="text">
 Kommentar: <input type="text">
 <button type="submit">Godkend</button>
</pre>
</form>
```

Figur 243

FormGroup<sup>19</sup> navngiver formen - eller giver den en handle om man vil. Herefter giver man alle input-elementer et **formControlName**.

```
Fornavn: <input type="text" formControlName="fornavn">
Efternavn: <input type="text" formControlName="efternavn">
Gade: <input type="text" formControlName="gade">
Postnr.: <input type="text" formControlName="postnummer">
By: <input type="text" formControlName="by">
<button type="submit">Godkend</button>
```

Figur 244

I form-elementet blev [formGroup]="navneForm" defineret.  
**navneForm** er en property i klassen og denne skal defineres:

```
export class AppComponent
 navneForm: FormGroup;
```

Herefter er klassen bundet sammen med den fysiske html-form.

<sup>19</sup> For at benytte FormGroup skal denne være importeret.

## Definition af felter med FormControl

Herefter defineres de enkelte felter i klassen.

```
fornavn = new FormControl("", Validators.required);
efternavn = new FormControl("", Validators.required);
gade = new FormControl("", Validators.required);
```

Der laves en ny instans med **new FormControl**. Bemærk parameterne. Med **Validators.required** kan der tilføjes simpel validering til feltet.

Alle disse **instanser af FormControl** overføres til deres respektive objekter (fornavn, efternavn, gade m.m.). Indtil nu er disse blot løse objekter uden forbindelse til html-formen.

Med FormBuilder kan man samle det hele.

## Opbygning af FormBuilder i constructoren

```
constructor(private formBuilder:FormBuilder) {}
```

FormBuilder skal importeres

Med DI oprettes formBuilder objektet ud fra FormBuilder klassen. Sagt kort får man et formBuilder-objekt, som kan benyttes til at samle alle felter.

**FormBuilderen har en Group()-funktion**, som kan gruppere felter. Typisk vil den bruges enten direkte i constructoren eller i ngOnInit som eksekveres umiddelbart efter constructoren.

```
ngOnInit() {
 this.navneForm=this.formBuilder.group({
 "fornavn": this.fornavn,
 "efternavn": this.efternavn,
 "gade": this.gade,
 "postnummer": this.postnummer,
 "by": this.by
 }) ;
}
```

## Angular 4 - Udvikling af Angular apps

Group()-funktionen indeholder et array. "fornavn" får værdien af this.fornavn (som svarer til new FormControl defineret ovenfor). Alle felter samles dermed i den samme formBuilder.group().

Alle disse felter lægges i this.navneForm, som er navnet på vores formGroup. Første gang skal man lige se koden et par gange, men logikken er til at få øje på.

## Validering af Controls

Man kan validere Reactive Forms på flere måder. Når man validerer har man mulighed for at benytte Observables. Man kan lytte på en stream (strøm).

Ved klassisk JavaScript, vil man ofte lave en imperativ funktion. Altså en funktion, som kaldes eksplisit og udfører et eller andet. Man kan forestille sig i forbindelse med Submit(), at man kalder en funktion og tester værdierne af et eller flere felter.

Her illustreret med "snydekode".

```
if felt1 = "" eller felt2 = "" eller felt3= "" then fejl/ikke submit.
```

Ved at bruge Observables kan man dels lytte/subscribe og validere formen undervejs. Dette kan oven i købet gøres på alle felter samtidig. Felterne er jo netop **FormControl**s samlet i **Groups**.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Validering af et enkelt felt på den "hårde måde"

The screenshot shows a simple form field labeled 'Fornavn:' followed by an empty input field. Below the input field, the text 'Tast fornavn!' is displayed in red, indicating a validation error.

Figur 245

Så snart man taster i fornavn-feltet forsvinder teksten nedenfor. Det sker fordi valideringen/fejlhåndteringen i virkeligheden er en observable.

```

 Tast fornavn!

```

Bemærk, at der ikke defineres en observable. Det er indbygget i Angular.

Figur 246

## Import Validators

```
import { Validators } from '@angular/forms';
```

Figur 247

Man kan importere **Validators**. I forbindelse med FormControl kan man effektivt validere felter.

```
new FormControl("", Validators.required);
```

Figur 248

Validators angives i forbindelse med **new FormControl**. Her er en required felt. **20**

## Regular Expressions

Man kan benytte Regular Expressions. Det vil se således ud:

```
postnummer = new FormControl(
 "", [
 Validators.required,
 Validators.pattern("[0-9]{4}")
]
);
```

2. parameteren i FormControl kan indeholde et array af Validators. Her er tilføjet 2 Validators.

**Den første** (blå) angiver, at feltet er required.

**Den næste (lilla)** (regular expressions) angiver, at feltet skal indeholde 4 tal - f.eks. 2620.

Strengt taget er *required* ikke nødvendigt, for regular expression udtrykket vil heller ikke acceptere en tom værdi. Men eksemplet illustrerer, at man kan tilføje flere validators som array.

---

**20** Først når feltet grupperes under FormBuilder, peger valideringen på et konkret felt i html-templetten.

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Test af validering

Med **FormControl** og **Validators** fåes et redskab til validering af de enkelte felter.

Valideringstest kan laves enten når man submitter en form - eller løbende med en Observable.

### Validering, når der submittes

```
<form [formGroup]="navneForm" (ngSubmit)="onSubmit(navneForm)">
 (ngSubmit)="onSubmit(navneForm)">
```

I html-templetten tilføjes en (ngSubmit)-event. Herefter kaldes funktionen onSubmit med navneForm (altså selve formen) som argument.

```
onSubmit({value, valid}: {value:Person, valid:boolean}) {
 alert(valid);
 alert(value.fornavn);
}
```

onSubmit()-funktionen modtager **navneForm**. I metoden gribes den som 2 argumenter.

**Det første argument** er alle felterne. Felterne defineres i forhold til **Person**, som er et interface, der er defineret i en separat fil.

```
export interface Person {
 fornavn:string;
 efternavn:string;
 gade:string;
 postnummer:string;
 by:string;
}
```

**Det andet argument** er en bolisk værdi, som ganske simpelt returnerer om formen er valid eller ej. Vel at mærke alle felter. Hvis alle felter (FormControls med tilhørende Validators) er valide vil argument være *true* ellers *false*.

## Kapitel 27 Model-driven forms (Reactive)

Den gule boks viser hvordan man kan hente dels feltværdierne (f.eks. value.fornavn), dels om formen som sådan er valid (valid).

```
onSubmit({value, valid}): void {
 alert(valid);
 alert(value.fornavn);
}
```

## Validering, løbende med observable

FormGroup som repræsenteres af navneForm har en property ved navn valueChanges, som er en observable.

```
this.navneForm.valueChanges
 .filter(data=>this.navneForm.valid)
 .subscribe(data=>this.tekst=(JSON.stringify(data)));
```

Denne kode er indsatt i `ngOnInit()`.

`.valueChanges (gul)` giver mulighed for at lytte på ændringer i formen. Herefter benyttes value chaining. Først laves et filter og herefter en subscribe.

`.filter (blå)` spørger direkte på om formen er valid. Der sendes altså først output videre til `.subscribe (orange)`, når data er valide. Først når brugeren har tastet et postnummer på 4 tal, vil `.subscribe` blive udført.

`.subscribe (orange)` laver en simpel `stringify21` og returnerer værdierne til tekst-proprietten, som igen vises i html-templetten med `{{tekst}}`.

## Model-driven Reactive Forms!!

Fornavn:	Søren
Efternavn:	Anderson
Gade:	Vognporten 14
Postnr.:	2620
By:	Albertslund

Godkend

```
{"fornavn":"Søren","efternavn":"Anderson","gade":"Vognporten
14","postnummer":"2620","by":"Albertslund"}
```

ZoomTek Kurser

<sup>21</sup> Stringify (som navnet lidt antyder), laver json om til en string.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Benyt observables på felter

```
ngOnInit() {
 this.kommentarControl.valueChanges.subscribe(value => {
 alert("Observable");
 });
}
```

Figur 249

```
<input type="text" [FormControl]="kommentarControl">
```

Figur 250

Når man taster i kommentarfeltet vil alert-boksen blive kaldt.

## Gruppering af felter med <fieldset>

Felter kan grupperes med html-elementet <fieldset>. Et fieldset har et **formGroupName** som kan refereres fra **FormBuilder**.

```
<fieldset formGroupName="adresse">
 Gade: <input type="text" formControlName="gade">
 Postnr.: <input type="text" formControlName="postnr">
 By: <input type="text" formControlName="by">
</fieldset>
```

Figur 251

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Den færdige form

```

1 import { Component, OnInit } from '@angular/core';
2 import { FormBuilder, FormGroup, FormControl, Validators }
 from '@angular/forms';
3 import { Person } from './person';

@Component({
 selector: 'app-root',
 template: `

```

(1) FormBuilder, FormGroup, FormControl og Validators importeres

(2) Person-interfacet indlæses.

```

3 <form [formGroup]="navneForm" (ngSubmit)="onSubmit(navneForm)">
 <h1>Model-driven Reactive Forms!!</h1>
 <pre>
4 Fornavn: <input type="text" formControlName="fornavn">
 Efternavn: <input type="text" formControlName="efternavn">
 <fieldset formGroupName="Adresse">
 Gade: <input type="text" formControlName="gade">
 Postnr.: <input type="text"
 formControlName="postnummer">
 By: <input type="text" formControlName="by">
 </fieldset>
 <button type="submit">Godkend</button>
 </pre>
5
 Tast fornavn!

</form>
6 {{tekst}}

```

`)`

(3) HTML-templaten erklæres som FormGroup med navnet navneForm.  
Endvidere (blå) angives en funktion, når der submittes.

(4) Felterne navngives med formControlName.

(5) \*ngIf benyttes til at udskrive en besked (Tast fornavn!), hvis feltet er tomt.

(6) *tekst* vises på skærmen. Tekst er lavet ud fra en observable - stringify.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



```

7 export class AppComponent implements OnInit {
8 tekst:string;
9 navneForm: FormGroup;
10 fornavn = new FormControl("", Validators.required);
11 efternavn = new FormControl("", Validators.required);
12 gade = new FormControl("", Validators.required);
13 postnummer = new FormControl(
14 "", [
15 Validators.required,
16 Validators.pattern("[0-9]{4}")
17]
18);
19 by = new FormControl("", Validators.required);

20 constructor(private formBuilder:FormBuilder) {}

```

(7) klassen defineres

(8) objektet **navneForm** defineres som en FormGroup. Samtidig oprettes FormControl objekter, som instans af FormControl. Disse får tilført valideringsregler.

(9) Postnummeret skal have 4 tal. Defineret med Regular Expressions

(10) Constructoren laver DI og leverer objektet formBuilder

```

ngOnInit() {
 this.navneForm=this.formBuilder.group({
 "fornavn": this.fornavn,
 "efternavn": this.efternavn,
 "gade": this.gade,
 "postnummer": this.postnummer,
 "by": this.by
 });
 this.navneForm.valueChanges
 .filter(data=>this.navneForm.valid)
 .subscribe(data=>this.tekst=(JSON.stringify(data)));
}

onSubmit({value, valid}:{value:Person, valid:boolean}) {
 alert(valid);
 alert(value.fornavn);
}

```

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 27 Model-driven forms (Reactive)

(11) Inden for **ngOnInit** defineres **formBuilder.group**, som består af alle felterne. Group metoden er gaffatapen, der får det hele til at hænge sammen.

(12) **valueChanges** laver et filter og en observable med value chaining. Først filtreres på valide data med **.filter**. Sidenhen laves der en **.subscribe**, som lytter til ændringer i realtime på formen.

(13) **onSubmit()** funktionen kaldes når formen submittes. (defineret i form-elementet (3)).

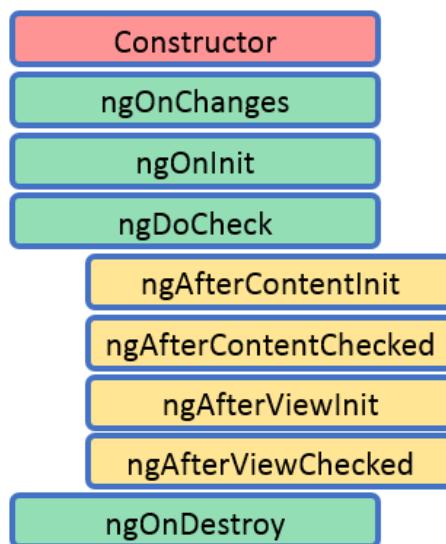


# Kapitel 28 LifeCycle Hooks

Et komponent har en livscyklus - forskellige faser fra komponentet kaldes ind på skærmen til det bliver fjernet igen.

Disse faser kan man 'hooke' ind i - og kalde sin egen kode. Man kan lave metoder med bestemte navne, som automatisk kaldes, når disse hooks aktiveres - når en given fase er nået.

Følgende hooks findes - og de udføres i denne rækkefølge:



Følgende LifeCycle Hooks kan benyttes i Angular. Constructoren er teknisk set ikke en LifeCycle-hook, men medtages ofte, da den reelt agerer som en sådan.

De grønne kasser er faser til selve komponentet. De gule kasser er faser, der reagerer på eventuelle child-komponenter.

**ngOnChanges** kaldes hver gang, der er en ændring på input-elementer på skærmen - f.eks. når brugeren skriver i et almindelig tekstfelt.

**ngOnInit** svarer nærmest til onLoad i JavaScript. Kaldes når komponentet initialiseres. Kaldes kun 1 gang.

## Angular 4 - Udvikling af Angular apps

**ngDoCheck** kaldes, når **change detectoren** aktiveres. Change detector bruges blandt andet ofte i unit testing. Dermed kan man som udvikler selv styre opdatering af template, variable og properties.<sup>22</sup>

**ngOnDestroy** er den sidste metode, der kaldes før komponentet lukker - inklusive hvis brugeren lukker browseren. Metoden bruges primært til unsubscribe eller afmelding af HostListeners og andet som eller bliver hængende i hukommelsen.

**ngAfterContentInit** kører kører efter Content Projection (se afsnit herom).

**ngAfterContentChecked** kører efter change detection har kørt på child-komponentet.

**ngAfterViewInit** kører efter child-komponentet har fået opbygget sit view.

## Import af interfaces

---

For at benytte disse skal man importere interfaces:

```
import { Component, OnInit, OnChanges } from '@angular/core';
Figur 252
```

De forskellige interfaces hedder det samme som hook'en - blot uden ng. Når et interface er importeres skal det implementeres. Der skal altså laves en funktion, der hedder ngOnInit() og ngOnChanges() for at matche ovenstående import.

- ➡ OnInit benyttes sammen med ngOnInit
- ➡ OnChanges benyttes sammen med de øvrige

Metoder kan i demoversion se således ud:

```
ngOnInit() {
 this.firma="Nyt firma";
 alert("Init...");
}
```

Figur 253

---

<sup>22</sup> ngDoCheck skal ikke bruges sammen med ngOnChanges.

Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



```
ngOnDestroy() {
 alert("2 Destroy..");
}
```

Figur 254

Funktionerne placeres i en klasse. De vil herefter automatisk blive kaldt.

## Angular 4 - Udvikling af Angular apps

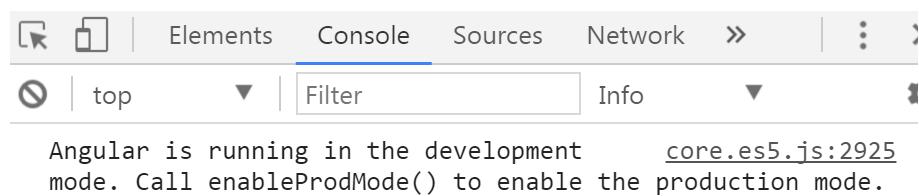
©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 29 Debugging - forskellige redskaber

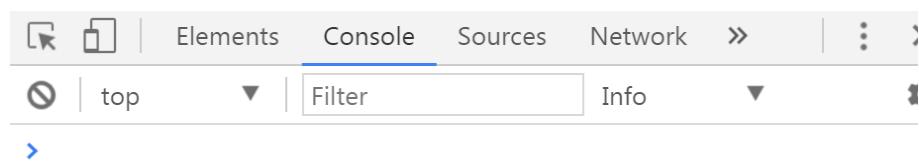
Når man opretter og kører et projekt, vil dette default køre i Developer Mode.

```
.. \users\kristoffer\aj\kodeeksempler\debugging>ng serve --open
* NG Live Development Server is listening on localhost:4200,
11% building modules 9/10 modules 1 active ..\odeEksempler\de
```

I Developer Tools ser det således ud:



Efter at have sat projektet i *production mode* vil konsolen ikke vise nogen besked.

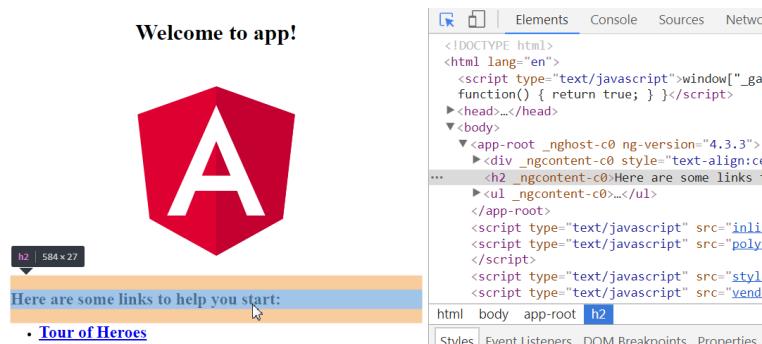


Man sætter et projekt i *production mode* ved at ændre production-parameteren i src/environments/environment.ts til true.

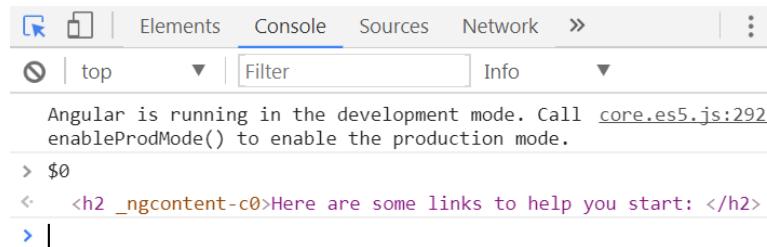
For at debugge, skal projektet køre i Developer Mode.

## \$0 - ændre instancen direkte

I Developer Tools kan man fra Elements pege på et givent element på skærmen.



Ved straks at gå til Konsolen og skrive \$0 - får man en handle på dette element.



Dette giver en reference til elementet. Med ng.probe(\$0) kan man studere alle parametre omkring elementet.

```
> ng.probe($0)
<- DebugElement {_debugContext: DebugContext_, nativeNode: h2,
 ▼ parent: DebugElement, listeners: Array(0), properties: Object
 ...} ⓘ
 ► attributes: Object
 ►childNodes: Array(1)
 children: (...)
 ► classes: Object
 componentInstance: (...)
 context: (...)
 injector: (...)
 ► listeners: Array(0)
 name: "h2"
 ► nativeElement: h2
 ► nativeNode: h2
```

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 29 Debugging - forskellige redskaber

```

▶ style: CSSStyleDeclaration
 tabIndex: -1
 tagName: "H2"
 textContent: "Here are some links to help you start: "
 title: ".nativeNode.textContent"
 translate: true
 ...
 ▶ HTML Document Element

```

Her ses hvordan man kan ændre teksten på et element på skærmen. Koden man skal skrive, aflures nemt ved at lade musen glide hen over propertien.

› `ng.probe($0).nativeNode.textContent="Ændring af tekst";`

Ændringen vises på skærmen. **23**



### Ændring af tekst

- [Tour of Heroes](#)
- [CLI Documentation](#)

---

**23** Ikke alle ændringer vises. I visse tilfælde skal man køre en `detectChanges()`-metode. Søg evt. efter `detectChanges` og [Pluralsight.com](#) for yderligere dokumentation.

#### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Debugging af TypeScript direkte

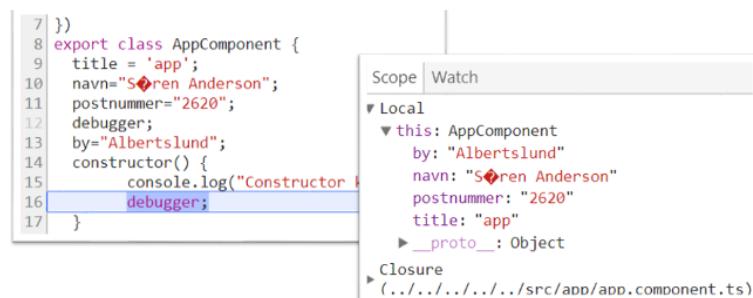
Man kan debugge TypeScript direkte. Rediger tsconfig.app.json<sup>24</sup> med følgende parameter:

```
tsconfig.app.json
 "module": "es2015",
 "sourceMap": true,
 "types": []
}
```

Ved at skrive `debugger;` i sin kode kan man lave et breakpoint.

```
8 export class AppComponent {
9 title = 'app';
10 navn="Søren Anderson";
11 postnummer="2620";
12 debugger;
13 by="Albertslund";
14 constructor() {
15 | console.log("Constructor kaldt");
16 | debugger;
17 }
18 }
```

Herefter kan man debugge direkte i TypeScript i Developer Tools



<sup>24</sup> I nyere CLI-versioner er det ikke nødvendigt at indsætte denne parameter. Her kan man lave debugging direkte i TypeScript automatisk.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## JSON-pipe

Man kan bruge JSON-pipen til at liste alle objekter og variable i ens app. Det gøres simpelt hen ved at skrive `{{this|json}}` i templat'en.

```

3 @Component({
4 selector: 'app-root',
5 template: `<h1>Debugging</h1>
6 | | {{this | json}}
7 |`
```

Det vil her give resultatet

A screenshot of a browser window showing the URL `localhost:4200`. The page content is a single line of JSON output: `{ "title": "app", "navn": "Søren Anderson", "postnummer": "2620", "by": "Albertslund" }`.

## Debugging

```
{ "title": "app", "navn": "Søren Anderson",
 "postnummer": "2620", "by": "Albertslund" }
```

### Angular 4 - Udvikling af Angular apps

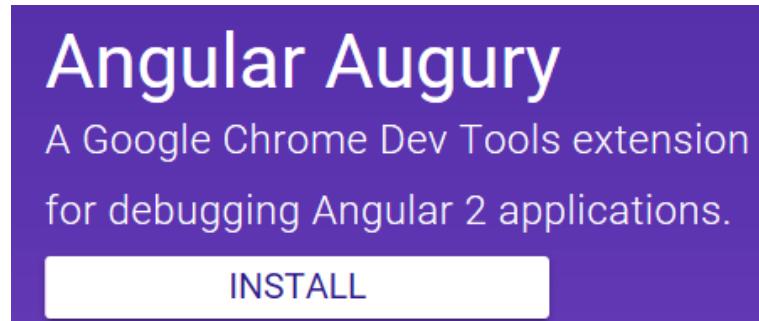
©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



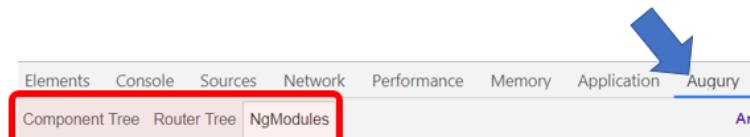
## Angular Augury

Angular Augury er det bedste debugging værktøj til Angular. Det kan downloades på denne adresse:

<https://augury.angular.io/>



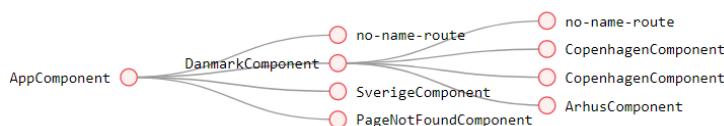
Efter download kan man benytte Augury i Google Developer Tools. Når man åbner DevTools, er der kommet et nyt Augury faneblad helt yderst til højre.



Vælger man Augury fanebladet fremkommer 3 underfaneblade. Her kan man se Component Tree, som kort er vist nedenfor.



Under Router Tree kan man grafisk se Routing. Man kan klikke på de enkelte dele og få yderligere detaljer.



### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 29 Debugging - forskellige redskaber

Under NgModules kan man se alle modules og deres providere i hele projektet.

Imports	Exports	Providers	Declarations	ProvidersInDeclarations
BrowserModule FormsModule RouterModule	None	Location UrlSerializer Router ChildrenOutletContexts ActivatedRoute NgModuleFactoryLoa... RouterPreloader NoPreloading PreloadAllModules LocationStrategy PreloadingStrategy NgProbeToken	AppComponent DanmarkComponent SverigeComponent PageNotFoundComponent CopenhagenComponent AarhusComponent	None
<hr/>				
Imports	Exports	Providers	Declarations	ProvidersInDeclarations
None	CommonModule ApplicationModule	Sanitizer DomSanitizer ErrorHandler InjectionToken: Event... InjectionToken: Event...	None	None

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



# Kapitel 30 Unit Testing

## Unit Test med Angular

---

Der finde flere muligheder for Unit Testing i Angular:

### Protractor

---

Benyttes til såkaldte e2e (end to end) test. Sådanne test simulerer applikationen som brugeren vil se den. 2 processer køres. Den ene kører applikationen almindeligt. Den anden kører testen som simulerer bruger interaktion.

### Angular Testing Utilities

---

Skaber et testmiljø for Angular under test

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Jasmine

---

Bruges til grundlæggende tests. Test køres igennem en test-runner, som eksekveres i browseren.

Selve "sproget", som testen skrives i er Jasmine.

## Karma

---

**Karma Test Runner** bruges til at teste en applikation midt i et testforløb. Karma vil blive gennemgået nedenfor. Benyttes CLI vil Karma Test Runner automatisk blive konfigureret.

Karma kaldes et **Automation Tool**. Karma bestemmer bl.a. den browser, der skal vise resultater (kan også køres i konsol eller program).

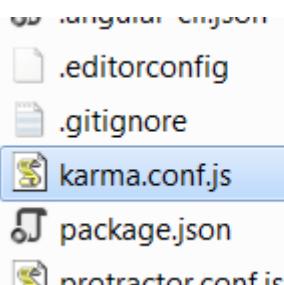
## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## karma.conf.js

Karma styres efter filen **karma.conf.js**. Denne fil opsættes automatisk, når man benytter CLI.



karma.conf.js genereres afhængig af installation. Sammen med CLI oprettes den automatisk. Uden CLI skal karma oftest installeres særskilt og karma.conf.js filen kan opsættes med kommandoen **karma init**

## Framework

```
frameworks: ['jasmine', '@angular/cli'],
plugins: [
```

Her vises det (de) framework(s), som ønskes benyttet. Disse skal være installeret gennem npm - alternativt som et karma-plugin.

## Plugins

```
plugins: [
 require('karma-jasmine'),
 require('karma-chrome-launcher'),
 require('karma-jasmine-html-reporter'),
 require('karma-coverage-istanbul-reporter'),
 require('@angular/cli/plugins/karma')
```

Diverse plugins som er påkrævet. Bl.a. karma-crome-launcher.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Files og Preprocessors

---

**Files** benyttes ikke af CLI, men det kan være filer, som skal loades ind sammen med Karma

**Preprocessors** benyttes ikke af CLI, men kan være operationer (oftest JS), der skal udføres umiddelbart før Karma testen.

## Webpack

---

Benyttes Webpack kan der være parametre om denne. CLI benytter Webpack, men har ikke behov for speciel konfiguration. Derfor findes der ikke særskilte Webpack parametre, når CLI benyttes.

## CoverageReporters

---

Disse benyttes til at konfigurere uddata fra det rapporteringsværktøj, som benyttes.

Under plugins ses det, at **istanbul-reporter** benyttes.

```
require('karma-jasmine-html-reporter'),
require('karma-coverage-istanbul-reporter'),
require('@angular/cli/plugins/karma')
```

Derfor er denne konfigureret under CoverageReporters

```
coverageIstanbulReporter: {
 reports: ['html', 'lcovonly'],
 fixWebpackSourcePaths: true
},
```

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## TestBed Konfiguration - src/test.ts

src/test.ts indeholder konfiguration af TestBed, som benyttes i selve testen. I selve filen er øverst skrevet følgende:

```
// This file is required by karma.conf.js and loads recursively all the
// .spec and framework files.
```

Filen kan hedde noget forskelligt og være placeret udenfor eller indeni src-mappen afhængig af installation (læs: med/uden CLI).

I denne fil findes en masse parametre og imports.

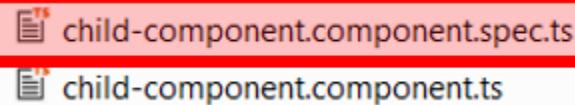
```
3 import 'zone.js/dist/long-stack-trace-zone';
4 import 'zone.js/dist/proxy.js';
5 import 'zone.js/dist/sync-test';
6 import 'zone.js/dist/jasmine-patch';
7 import 'zone.js/dist/async-test';
8 import 'zone.js/dist/fake-async-test';
9 import { get TestBed } from '@angular/core/testing';
10 import {
11 BrowserDynamicTestingModule,
12 platformBrowserDynamicTesting
13 } from '@angular/platform-browser-dynamic/testing';
14
15 // Unfortunately there's no typing for the `__karma__` vari
16 declare const __karma__: any;
17 declare const require: any;
18
19 // Prevent Karma from running prematurely.
20 karma.loaded = function () {};
```

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## ng generate --spec

Ved brug at **ng generate** kan man tilføje parameteren --spec. Dermed tilføjes en spec-fil, som benyttes til test.



Figur 255

## ng test

Med kommandoen **ng test<sup>25</sup>** kan man teste alle spec-filer i samtlige features på samme tid.

```
10% building modules 3/4 modules 1 active ... \introduktion\NyPakke\src\polyfills.ts
22 07 2017 10:28:01.226:INFO [karma]: Karma v1.7.0 server started at http://0.0.0.0:9876/
22 07 2017 10:28:01.229:INFO [launcher]: Launching browser Chrome with unlimited concurrency
10% building modules 4/4 modules 0 active22 07 2017 10:28:01.358:INFO [launcher]: Starting browser Chrome
22 07 2017 10:28:45.098:WARN [karma]: No captured browser, open http://localhost:9876/
22 07 2017 10:28:47.403:INFO [chrome 59.0.3071 (Windows 7 0.0.0)]: Connected on socket -M8VFTg7PJKGNNJHAAAA
Chrome 59.0.3071 (Windows 7 0.0.0): Executed 4 of 4 SUCCESS (1.136 secs / 1.11 secs)
```

Figur 256

Til sidst vises unit-testen i browseren.



Figur 257

<sup>25</sup> Med CLI hedder kommandoen **ng test**. Hvad den præcis hedder, og hvordan den opfører sig, kan man se/indstille i package.json. **npm start** skrives, hvis man ønsker at benytte npm.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 30 Unit Testing

```
1 // Karma configuration file, see link for more information
2 // https://karma-runner.github.io/0.13/config/configuration-api.html
3
4 module.exports = function (config) {
5 config.set({
6 basePath: '',
7 frameworks: ['jasmine', '@angular/cli'],
8 plugins: [
9 require('karma-jasmine'),
10 require('karma-chrome-launcher'),
11 require('karma-jasmine-html-reporter'),
12 require('karma-coverage-istanbul-reporter'),
13 require('@angular/cli/plugins/karma')
14],
15 },
16 client: {
17 clearContext: false // leave Jasmine Spec Runner output
18 },
19 coverageIstanbulReporter: {
20 reports: ['html', 'lcovonly'],
21 fixWebpackSourcePaths: true
22 }
23 }
```

Figur 258 - karma.conf.js er filen, som konfigurerer karma.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Følgende sker når en test køres:

- Angular CLI loader .angular-cli.json (ligger i rodren).

```

10 "assets": [
11 "assets",
12 "favicon.ico"
13],
14 "index": "index.html",
15 "main": "main.ts",
16 "polyfills": "polyfills.ts",
17 "test": "test.ts",
18 "tsconfig": "tsconfig.app.json",

```

Figur 259: .angular-cli.json

- Angular kører Karma Test Runner ud fra specifikationerne i .angular-cli.json. Den kører default karma.conf.js, som ligger i rodren af projektet.
- Karma.conf.js beskriver bl.a., hvilken browser, der skal benyttes. Karma åbner i denne browser.

```

1 // Karma configuration file, see link for more information
2 // https://karma-runner.github.io/0.13/config/configuration
3
4 module.exports = function (config) {
5 config.set({
6 basePath: '',
7 frameworks: ['jasmine', '@angular/cli'],
8 plugins: [
9 require('karma-jasmine'),
10 require('karma-chrome-launcher'),
11 require('karma-jasmine-html-reporter'),
12 require('karma-coverage-istanbul-reporter'),
13 require('@angular/cli/plugins/karma')
14],
15 client: {
16 clearContext: false // leave Jasmine Spec Runner output

```

Figur 260: Karma.conf.ts

- Browseren kører src/test.ts (automatisk oprettet med ng new) efter ordre fra Karma. Den benytter Default Jasmine frameworket, men kan også benytte andre frameworks. Dette står også i Karma konfigurationsfilen. Src.test.ts kører alle spec-filer. Den søger simpelt hen på navnet.

```

1 // This file is required by karma.conf.js and loads recursive
2 .spec and framework files
3 import 'zone.js/dist/long-stack-trace-zone';
4 import 'zone.js/dist/proxy.js';
5 import 'zone.js/dist/sync-test';
6 import 'zone.js/dist/jasmine-patch';
7 import 'zone.js/dist/async-test';
8 import 'zone.js/dist/fake-async-test';
9 import { getTestBed } from '@angular/core/testing';
10 import {
11 BrowserDynamicTestingModule,
12 platformBrowserDynamicTesting
13 } from '@angular/platform-browser-dynamic/testing';

```

Figur 261: src/test.ts

- Karma returnerer testresultatet i konsollen.
- Herefter vil Karma løbende overvåge (watch) alle .ts filer for ændringer. Disse vil i realtime blive vist i browseren og konsollen.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



```
10 "assets": [
11 "assets",
12 "favicon.ico"
13],
14 "index": "index.html",
15 "main": "main.ts",
16 "polyfills": "polyfills.ts",
17 "test": "test.ts",
18 "tsconfig": "tsconfig.app.json",
```

Figur 262

## Angular 4 - Udvikling af Angular apps

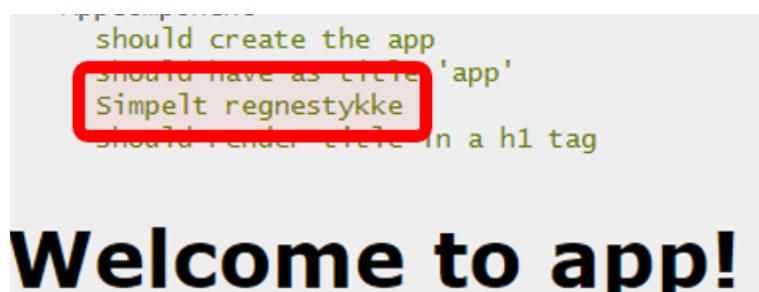
©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Opbygning af en simpel test

Her er et eksempel på verdens mest simple test.

```
it('Simpelt regnestykke', () => {
 expect(2+2).toEqual(4);
});
```

Hvis denne skrives ind i app.component.spec.ts-filen og der herefter køres en **ng test** er resultatet følgende:

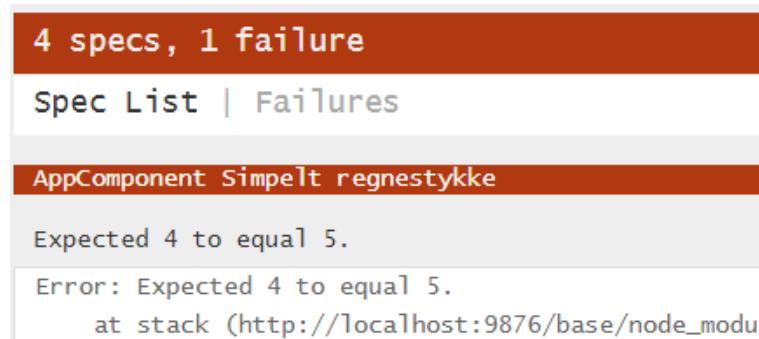


## Welcome to app!

Ændres koden til

```
it('Simpelt regnestykke', () => {
 expect(2+2).toEqual(5);
});
```

Så vises der en fejl.



**expect** benyttes til at opbygge et bolisk udtryk. Er dette sandt vil testen blive godkendt. Er det boliske udtryk falsk vil testen fejle.<sup>26</sup> Så simpelt er det i virkeligheden. Al anden unit testing er groft sagt yderligere udvidelse af denne simple kode.

## Pending specs

Ved at indsætte et simpelt x foran it()-metoden kan man midlertidigt sætte en test ud af funktion.

```
xit('Simpelt regnestykke', () => {
 expect(2+2).toEqual(5);
});
```

4 specs, 0 failures, 1 pending spec

```
AppComponent
 should create the app
 should have no title
 Simpelt regnestykke
 Should render 'Hello' in a h1 tag
```

<sup>26</sup> chai er et bibliotek som gør test-syntaksen mere raffineret. Den er som standard ikke koblet op mod CLI. Chai muliggør kode a la `chai.expect(2+2).to.be.a('number')`. Ønsker man chai, skal dette installeres og indsættes via karma.config.js. Læs evt. mere på [rangle.io - using chai](https://rangle.io - using chai)

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

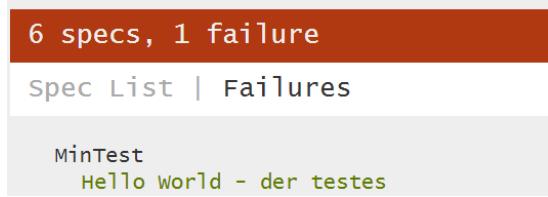
## Virker testmiljøet?

Man kan lave en hurtig testfil - og gemme den og gemme den i src/app. Testfilen kan se ud som nedenfor:

```
1 describe('MinTest', () => {
2 it('Hello World - der testes', () => expect(true).toBe(true));
3 }
4);
```

Figur 263

### Tast ng test



Figur 264

En testfil som denne skal blot hedde noget med spec og en extension på .ts. Fx kan filen hedden **testVirker.spec.ts**

Ng test kører på alle spec-filer - også selv om der ikke er koblet et komponent/feature op mod denne.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Debugging

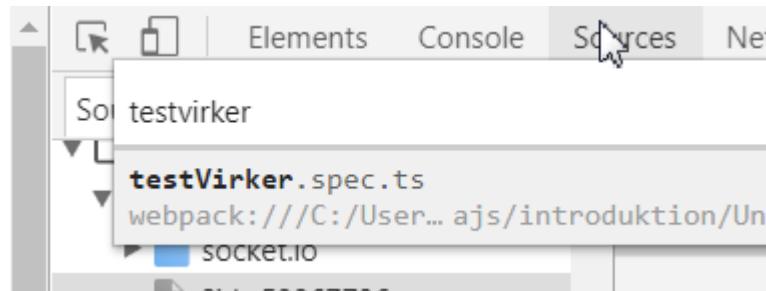
Kør ng test



Figur 265

Øverst til højre trykkes på DEBUG-knappen. Et ny vindue åbner. Gå tilbage til det oprindelige vindue (hvor der blev trykket på debug-knappen).

Vælg 'Developer Tools' (CTRL-Shift-I)



Figur 266

Med Ctrl-P kan man søge på sin spec-fil. Her søgeres efter `testVirker.spec.ts`.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 30 Unit Testing

The screenshot shows the Chrome DevTools Sources tab. On the left, there's a tree view of files: 'top' (expanded), 'localhost:9876' (expanded), 'socket.io' (under localhost), and a file named '?id=53367736' (selected). On the right, the code editor shows a file named 'testVirker.spec.ts'. Line 2 contains a breakpoint, indicated by a blue arrow pointing to the left margin. The code is as follows:

```

1 describe('MinTest', () => {
2 it('Hello World - der testes', () =>
3 expect(true).toEqual(true)
4);
5 });

```

Figur 267

Når filen fremkommer kan man sætte et breakpoint (her linje 2) ved at dobbeltklikke yderst til venstre, hvor cursorpilen peger.

Når browseren refreshes vil karma stoppe ved dette punkt.

The screenshot shows a terminal window with the following log output:

```

[0] Chrome 56.0.2924 (Windows 7 0.0.0): Executed 6 of 6 (1 FAILED) (0.362 sec)
[0] 6:33:56 PM - File change detected. Starting incremental compilation...
[0] 6:34:00 PM - Compilation complete. Watching for file changes.
[1] 15 02 2017 18:34:00.312:INFO [watcher]: Changed file "C:/Users/Kursist 1/
[1] Chrome 56.0.2924 (Windows 7 0.0.0) AppComponent should have expected <h1>
[1] Expected 'Test Tour of Heroes' to match /angular/i, '<h1>' should say
[1] Chrome 56.0.2924 (Windows 7 0.0.0): Executed 3 of 6 (1 FAILED) (0 secs / 0
[1] Chrome 56.0.2924 (Windows 7 0.0.0) AppComponent should have expected <h1>
[1] Expected 'Test Tour of Heroes' to match /angular/i, '<h1>' should say
[1] Chrome 56.0.2924 (Windows 7 0.0.0): Executed 6 of 6 (1 FAILED) (0.319 sec)
[0] 6:34:18 PM - File change detected. Starting incremental compilation...
[0] src/app/1st.spec.ts(3,2): error TS1200: Line terminator not permitted before
[0] 6:34:21 PM - Compilation complete. Watching for file changes.
[1] 15 02 2017 18:34:21.808:INFO [watcher]: Changed file "C:/Users/Kursist 1/
[1] Chrome 56.0.2924 (Windows 7 0.0.0) AppComponent should have expected <h1>
[1] Expected 'Test Tour of Heroes' to match /angular/i, '<h1>' should say
[1] Chrome 56.0.2924 (Windows 7 0.0.0): Executed 3 of 6 (1 FAILED) (0 secs / 0
[1] Chrome 56.0.2924 (Windows 7 0.0.0) AppComponent should have expected <h1>
[1] Expected 'Test Tour of Heroes' to match /angular/i, '<h1>' should say
[1] Chrome 56.0.2924 (Windows 7 0.0.0): Executed 6 of 6 (1 FAILED) (0.338 sec)

```

Figur 268

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Test af komponent

- En Unit Test fil skal hedde *et\_eller\_andet.component.spec.ts*, hvor *et\_eller\_andet* oftest er navnet på komponentet. Så spec-filen kan f.eks. hedde:

**app.component.spec.ts**

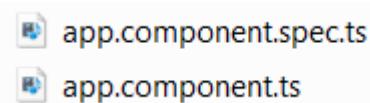
Den simpleste unit test kræver 2 filer. Et component (typisk *app.component.ts*) samt en test-fil (typisk *app.component.spec.ts*).

*app.component* kan se således ud:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4 selector: 'my-app',
5 template: '<h1>Hello World: {{title}}</h1>'
6 })
7 export class AppComponent {
8 title = 'Angular Testing';
9 }
```

Figur 269

I det følgende testes på værdien af 'title'. Derfor laves en *app.component.spec.ts* fil - som lægges i den samme mappe.



Figur 270

Følgende kode indsættes:<sup>27</sup>

```
1 import { ComponentFixture, TestBed } from '@angular/core/testing';
2 import { By } from '@angular/platform-browser';
3 import { DebugElement } from '@angular/core';
4 import { AppComponent } from './app.component';
```

Figur 271

<sup>27</sup> Koden ses i alle projekter, der er lavet med CLI. Som standard er async automatisk implementeret.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 30 Unit Testing

Hele koden ligger i modulet '@angular/core/testing'.

```
import { TestBed, async } from '@angular/core/testing';
```

- ➡ **TestBed importeres** (linje 1) - TestBed laver testmodul med parametre, som defineres af **.configureTestingModule**. Med andre ord kloner man det rigtige modul, så man har noget at lege og teste med.

```
5 describe('AppComponent (inline template)', () => {
6 // Al anden kode pastes ind her
7 // inkl. beforeEach, ConfigureTestingModule
8 // og it/expect
9 // -> Forklaring kommer senere
10});
```

Figur 272

Lige efter linje 5 pastes denne kode ind:

```
7 let comp: AppComponent;
8 let fixture: ComponentFixture<AppComponent>;
9 let de: DebugElement;
10 let el: HTMLElement;
```

Figur 273

- ➡ **TestBed.configureTestingModule** sætter parametre på testmodul - laver reelt det samme som **module.ts** - blot på testmodul.

Herefter denne kode:

```
12 beforeEach(() => {
13 TestBed.configureTestingModule({
14 declarations: [AppComponent], // Testkomponent
15 });
16 fixture = TestBed.createComponent(AppComponent);
17 // fixture er instans af AppComponent
18 comp = fixture.componentInstance; // AppComponent t
19 // query for the title <h1> by CSS element selector
20 // comp refererer til applikations properties. comp
21 // lig med title i AppComponent-klassen.
22 // el (nativeelement) refererer til html-element -
23 // h1 hentes senere ud med el.textContent
24 de = fixture.debugElement.query(By.css('h1'));
25 el = de.nativeElement;
26});
```

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Figur 274

- ➡ **fixture** er en instans af AppComponent.
- ➡ **comp** svarer direkte til AppComponent. f.eks. kan man refererer til *comp.title* direkte, hvilket vil svare til title defineret som variabel i AppComponent-klassen. (a la title:string)
- ➡ med **Query(By.css(...))** kan man i bedste JQuery stil spørge direkte på h1-elementet i templatet
- ➡ **el** svarer til **nativeElement** for h1.

Til sidst denne kode:

```
25 it('Bor altid være sand - er comp.title=el.textContent', () => {
26 fixture.detectChanges();
27 expect(el.textContent).toContain(comp.title);
28 });
29 it('Aendrer property (title) og sammenligner med "Bimse"', () => {
30 comp.title = 'Bimse';
31 fixture.detectChanges();
32 expect(el.textContent).toContain('Bimse');
33 });
```

Figur 275: Change Detection i linje 26 og 31

Her laves selve testen. Det er de 2 blokke, som starter med it(). Begge indeholder .detectChanges(), som reagerer på ændringer.

I den første (linje 25-28) sammenlignes (expect) tekstdindholdet af h1-elementet (el.textContent) med title applicationspropertien (comp.title). Dette vil altid være sandt.

I den anden (linje 29-32) ændres først på applikations propertien title til 'Bimse'. Herefter spørges der om (expect) tekstdindholdet af h1 (el.textContent) indeholder (.toContain) 'Bimse'. Det må man sige ja til. Testen er positiv.

Med CLI er Unit Testing automatisk inkluderet, når man opretter et nyt projekt med **ng new**.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Test af funktioner

```

@Component({
 selector: 'app-root',
 template: `
 <h1>Welcome to {{title}}!</h1>
 `,
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 title = 'app';
 nyOverskrift(tekst: string) {
 this.title = tekst;
 }
 sletOverskrift() {
 this.title = "";
 }
}

```

Her er 2 simple funktioner. De kaldes ikke fra templetten, men alligevel kan man lave en test.

En test kan laves således:

```

it('Metoden skal sætte en tekst i overskrift', () => {
 const fixture = TestBed.createComponent(AppComponent);
 const app = fixture.debugElement.componentInstance;
 app.nyOverskrift('ZoomTek');
 expect(app.title).toBe('ZoomTek');
});

```

Metoden `nyOverskrift()` kaldes med en fiktiv værdi, som der efterfølgende testes på.

## Test af komponent med child-komponent

Et child-component oprettes let med `ng generate component child`

Default vil selectoren hedde 'app-component'

### Angular 4 - Udvikling af Angular apps

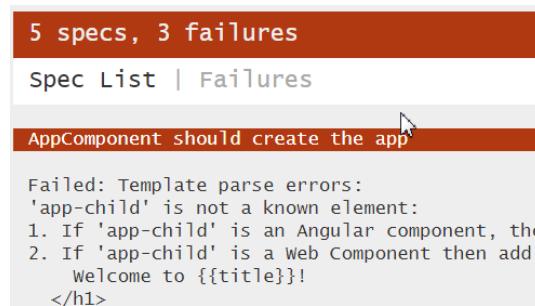
©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



```
3 @Component ({
4 selector: 'app-child',
5 templateUrl: './child.component.html',
6 styleUrls: ['./child.component.css']}
```

Figur 276

Herefter kan man nemt køre applikationen med **ng serve --open**. Men kører man en **ng test** får man en fejl.



Figur 277

Her er den fulde fejl:

```
2 Failed: Template parse errors:
3 'app-child' is not a known element:
4 1. If 'app-child' is an Angular component, then verify that it is part of this
 module.
5 2. If 'app-child' is a Web Component then add
 'CUSTOM_ELEMENTS_SCHEMA' to the '@NgModule.schemas' of this
 component to suppress this message. ("
6 Welcome to {{title}}!
7 </h1>
8 [ERROR ->]<app-child></app-child>
9 </div>
10 <h2>Here are some links to help you start: </h2>
11 ") : ng:///DynamicTestModule/AppComponent.html@5:1
```

Figur 278

## Angular 4 - Udvikling af Angular apps

## Kapitel 30 Unit Testing

Fejlen opstår fordi ChildComponent ikke er importeret i TestModulet:

I app.component.spec.ts indsættes ChildComponent:

```
1 import { TestBed, async } from '@angular/core/testing';
2
3 import { AppComponent } from './app.component';
4 import { ChildComponent } from './child/child.component';
5
6 describe('AppComponent', () => {
7 beforeEach(async(() => {
8 TestBed.configureTestingModule({
9 declarations: [
10 AppComponent,
11 ChildComponent
12],
13 }).compileComponents();
14 }));
15
16 it('should create the app',
17 inject([ElementRef, ComponentFixture], (ref: ElementRef, fixture: ComponentFixture) => {
18 expect(fixture.nativeElement.querySelector('h1')).toHaveText('Welcome to Angular!');
 }));
19 });
20
```

Figur 279



## Asynkrone tests

En Asynkron test er nødvendig, så snart der refereres til eksterne filer.

```
1 import { Component } from '@angular/core';
2
3 @Component({
4 selector: 'app-root',
5 templateUrl: './app.component.html',
6 styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9 title = 'app';
10 }
```

Figur 280: templateUrl og styleUrls er asynkrone kald

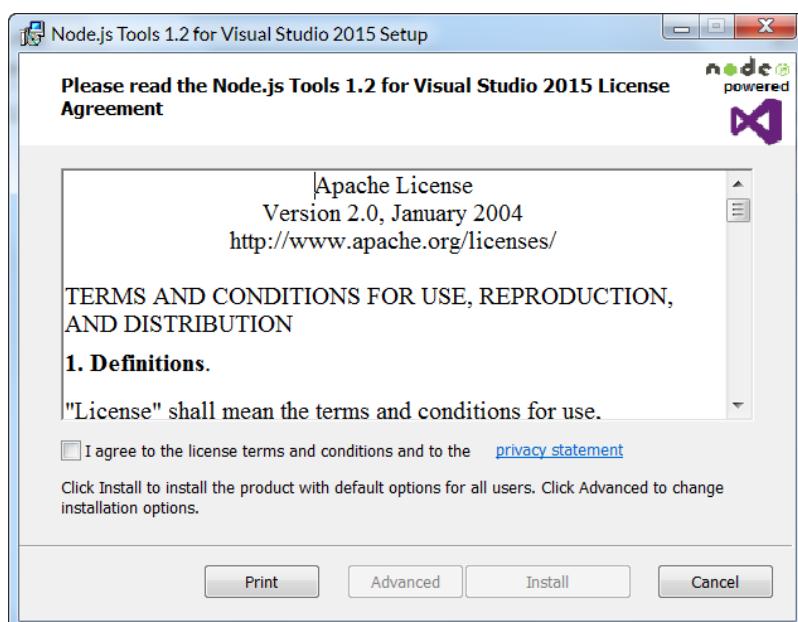
Asynkrone kald er mulige ved brug af **async()**.

```
6 describe('AppComponent', () => {
7 beforeEach(async(() => {
8 TestBed.configureTestingModule({
9 declarations: [
10 AppComponent, ChildComponent
11],
12 }).compileComponents();
13 }));
14 });
15
16 it('should create the app', async(() => {
17 const fixture = TestBed.createComponent(AppComponent);
18 const app = fixture.debugElement.componentInstance;
19 expect(app).toBeTruthy();
20 }));
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
237
238
239
239
240
241
242
243
244
245
245
246
247
247
248
248
249
249
250
250
251
251
252
252
253
253
254
254
255
255
256
256
257
257
258
258
259
259
260
260
261
261
262
262
263
263
264
264
265
265
266
266
267
267
268
268
269
269
270
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
13
```

## Unit Testing i Visual Studio

Til Visual Studio findes tillægsprogrammet **NodeJS Tools for Visual Studio**. Når det er installeret giver det en række ekstra muligheder i Visual Studio.

Node.js Tools for Visual Studio understøtter udførelse af unit tests. Man får mulighed for at styre, køre, debugge og filtrere unit tests uden en CMD. Brug mocha eller udvidet Visual Studio, så det kan arbejde med det ønskede unit testing framework.



Figur 283

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



# Kapitel 31 Index.html

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
5 <meta charset="utf-8">
6 <title>Skabelon2</title>
7 <base href="/">
8 ...
9 <meta name="viewport" content="width=device-width, initial-scale=1">
10 <link rel="icon" type="image/x-icon" href="favicon.ico">
11 </head>
12 <body>
13 <app-root></app-root>
14 </body>
15 </html>
```

Figur 284

Laves man Vis Kilde på websiden, ser den noget anderledes ud:

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
5 <meta charset="utf-8">
6 <title>Skabelon2</title>
7 <base href="/">
8 ...
9 <meta name="viewport" content="width=device-width, initial-scale=1">
10 <link rel="icon" type="image/x-icon" href="favicon.ico">
11 </head>
12 <body>
13 <app-root></app-root>
14 <script type="text/javascript" src="inline.bundle.js"></script><script>
15 type="text/javascript" src="polyfills.bundle.js"></script><script>
16 type="text/javascript" src="styles.bundle.js"></script><script>
17 type="text/javascript" src="vendor.bundle.js"></script><script>
18 type="text/javascript" src="main.bundle.js"></script></body>
19 </html>
```

Figur 285

Nederst i linje 14 indlæses en række js-filer, som ikke er angivet i index.html. Disse bliver injected automatisk af CLI, når man laver en **ng serve**. Ved opstart bliver de korrekte js-filer automatisk bundlet/samlet.

Inkluderet i disse filer, ligger information om at starte main.ts, som igen peger på module.ts.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Package.json og index.html

Kigger man på package.json, vil man se en lang række dependencies. Disse dependencies vil blive bundled (bundtet) og injected nederst i index.html ved runtime.

Bruger man ikke CLI, men i stedet diverse boilerplates fra Github, vil der typisk blive importeret en række js-filer direkte i headeren på index.html - altid. CLI er lidt smartere - for den injector disse link i sidste øjeblik - og samler kun de js-filer, der er nødvendige.

```
"dependencies": {
 "@angular/animations": "^4.0.0",
 "@angular/common": "^4.0.0",
 "@angular/compiler": "^4.0.0",
 "@angular/core": "^4.0.0",
 "@angular/forms": "^4.0.0",
 "@angular/http": "^4.0.0",
 "@angular/platform-browser": "^4.0.0",
 "@angular/platform-browser-dynamic": "^4.0.0",
 "@angular/router": "^4.0.0",
 "core-js": "^2.4.1",
 "martinhansenxx": "^1.0.0",
 "rxjs": "^5.4.1",
 "zone.js": "^0.8.14"
},
}
```

Figur 286

Øverst står der 'dependencies'. Længere nede i package.json filen finde **devDependencies**.

```
"devDependencies": {
 "@angular/cli": "1.2.3",
 "@angular/compiler-cli": "^4.0.0",
 "@angular/language-service": "^4.0.0",
 "@types/jasmine": "~2.5.53",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "~3.0.1".
}
```

Figur 287

Når projektet i sidste ende buildes med **ng build** vil disse dependencies ikke medgå.

### Angular 4 - Udvikling af Angular apps

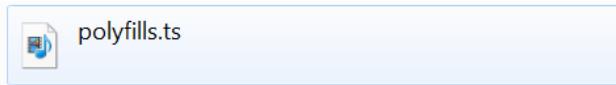
©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



# Kapitel 32 Polyfills

Polifills er JavaScript-filer, som benyttes til at lave cross-browser løsninger. Ikke alle browsere supporterer alt i Angular. Dertil har man lavet Polyfills.

Benytter man CLI, findes der i src-mappen en fil ved navn polyfills.ts



Figur 288

Denne fil er fuld af links til JS-filer. Fjern kommenteringen (de 2 skråstregen foran **Import**).

Filen læses før kørsel af en app.

```
18 * BROWSER POLYFILLS
19 */
20
21 /** IE9, IE10 and IE11 requires all of the
22 import 'core-js/es6/symbol';
23 import 'core-js/es6/object';
24 import 'core-js/es6/function';
25 import 'core-js/es6/parse-int';
26 import 'core-js/es6/parse-float';
27 import 'core-js/es6/number';
28 import 'core-js/es6/math';
29 import 'core-js/es6/string';
```

Figur 289

Her ses polyfills krævet, hvis Internet Explorer 9, 10 eller 11 skal benyttes. Bemærk at polyfills og indeholdet i denne fil løbende opdateres ændres.

Angular 4 - Udvikling af Angular apps

# Kapitel 33 Reactive Programmering

- ➡ Overblik
- ➡ Et Simpelt eksempel
- ➡ Hvad er Observables
- ➡ Lidt baggrund: Call Stacks og Event Tables
- ➡ Alternativer inden for Asynkrone data
- ➡ Hvad er RxJS
- ➡ Brug af Operatorer
- ➡ Hvad er Promises

Observables og Promises giver os mulighed for at arbejde med applikationer og data, der er asynkrone af natur.

## Overblik

---

Dette kapitel giver et overblik over Reactive Programmering, hvor de berømte Observables og sekundært Promises indgår.

APIlet i reactive programmering er enormt. Angular benytter reactive programmering igennem RxJS.

RxJS kan benyttes til JavaScript, TypeScript, php, .net og meget andet. Angular benytter RxJS gennem TypeScript.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Streams

Et centralt begreb i reactive programmering er streams.

En Stream er en sekvens af værdier over tid.

Eksempler på streams kunne være:

- ➡ Et tal der stiger med 1 for hver sekund -> 1,2,3,4,5..... osv.
- ➡ En sekvens af karakterer tastet i et felt
- ➡ En sekvens af koordinater (x,y) for museklik
- ➡ Tastetryk på skærmen (event.keyCode)
- ➡ Data fra en json-fil
- ➡ Musik, der hentes ned fra Spotify
- ➡ osv.

Reactive programmering er et koncept - en tænkemåde - men også konkret kodeteknik.

Reactive programmering handler om at kunne lave al programmering ved hjælp af *streams* og tilhørende operationer.

Alternativet til Reactive programmering (lidt sort/hvidt) kaldes **imperativ programmering**. Det er hvad man forstår ved klassisk programmering. Imperativ kodeopbygning består typisk af funktioner, som kaldes og som returnerer værdier.

Med reactive programmering kan man *lytte* på en strøm, og den lytten vil før eller siden kaste data af sig, som der så reageres på. Man lytter altså **passivt**, hvor imperative funktionskald er **aktive**.

**Problemet ved Imperativ programmering** er at bestemme, hvornår data ændres og en funktion skal kaldes igen. Hvornår skal data opdateres - og hvilke andre data skal opdateres, når de første data er ændret.

**Reactive programmering** laver sammenhænge i data. Ændringer af data - uanset hvor de kommer fra - bliver automatisk bestemt, og de rette funktioner kan kaldes - som igen ændrer data - og derfor automatisk ændrer andre data (læs: streams).

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

ZoomTek leverer kurser til A.P.Møller, Novo, Danske Bank, Vestas, Forsvaret, Grundfos,

Det er denne automatik/lytten, som er forskellig fra imperativ programmering. Data i moderne webapplikationer kan ændres mange steder. Af brugeren, af serveren, af eksterne filer/programmer osv. Reactive programmering reagerer automatisk på sådanne ændringer.

## Simpelt eksempel med ren html/javascript

For at benytte RxJS skal man importere 1 eller flere js-filer. Her importeres rx.all.js. Den indeholder al rx.js. [28](#)

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/rxjs/4.1.0/rx.all.js"></scr
ipt>
```

Herefter oprettes et 'lytter'-objekt ud fra Rx.Observable.

```
let lytter = Rx.Observable
 .interval(100)
 lytter.subscribe(
 data => console.log("Abonnement nr: " + data)
);
```

Efterfølgende sættes et interval til 100 millisekunder (10 i sekundet).

.interval(100) er en *Observable Chain*.

.interval kaldes også en operator. Dem findes der et utal af. De kædes sammen med et punktum.

```
Rx.Observable
 .operator1
 .operator2
 .operator3
```

Operatorer styrer, filtererer, sammenlægger og på anden måde manipulerer med de indkomne data.

---

[28](#) Man kan importere brudstykker af denne fil. At importere alt kan selvfølgelig give et performanceproblem, men det er der ikke fokus på her - kun funktionalitet.



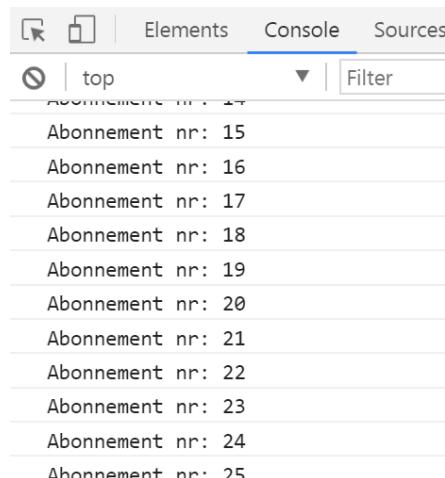
En operator og Observable Chains minder meget om CMDlets, der bliver pipet i PowerShell. Blot hvis du kender PowerShell. En kold observable er som en norsk fodboldkamp. Intet sker

Denne observable (lytter) kaldes en **kold observable**. Kører man koden på skærmen er det åbenlyst - for der sker ikke noget som helst overhovedet. Omtrent som i en norsk fodboldkamp.

Man gør en observable varm med subscribe.

```
lytter.subscribe(
 | data => console.log("Abonnement nr: " + data)
);
```

Her skrives data til loggen. I praksis ser det således ud:



A screenshot of a browser's developer tools showing the 'Console' tab selected. The output area displays a series of log messages: 'Abonnement nr: 15', 'Abonnement nr: 16', 'Abonnement nr: 17', 'Abonnement nr: 18', 'Abonnement nr: 19', 'Abonnement nr: 20', 'Abonnement nr: 21', 'Abonnement nr: 22', 'Abonnement nr: 23', 'Abonnement nr: 24', and 'Abonnement nr: 25'. Each message is preceded by a timestamp and followed by a timestamp.

Man kan tilføje flere operationer på. F.eks. med .map

```
.interval(100)
.map(()=>Date.now())
.map(x=>Math.floor(x/1000));
lytter.subscribe(
 | data => console.log("Abonnement
);
```

Date.now() returnerer antal millisekunder siden 1. januar 1970.  
Math.floor(x/1000) dividerer alle disse milliarder af millisekunder med 1000.

Resultatet er nu:

#### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

10 Abonnement nr: 1503398574  
 10 Abonnement nr: 1503398575  
 10 Abonnement nr: 1503398576  
 10 Abonnement nr: 1503398577  
 10 Abonnement nr: 1503398578  
 10 Abonnement nr: 1503398579  
 10 Abonnement nr: 1503398580

Den øverste (1503398574) er logget 10 gange. Hvorfor det?

Fordi Math.floor dividerer med 1000. Altså returnerer Math.floor en ny værdi hvert sekund. Efter som 'lytter' kører med et interval på 100 millisekunder, kommer der altså 10 pr. sekund.

Lidt teoretisk - men interessant. Ændres intervallet til 90 (i stedet for 100), vil der komme 11,1 svar i sekundet. Decimalen hober sig op til 12 hvert 9. sekund.

12 Abonnement nr: 1503398956  
 11 Abonnement nr: 1503398957  
 11 Abonnement nr: 1503398958  
 11 Abonnement nr: 1503398959  
 11 Abonnement nr: 1503398960  
 11 Abonnement nr: 1503398961  
 11 Abonnement nr: 1503398962  
 11 Abonnement nr: 1503398963  
 11 Abonnement nr: 1503398964  
 12 Abonnement nr: 1503398965  
 11 Abonnement nr: 1503398966

Der findes (i skrivende stund) **451 RxJS operatorer**.

Disse kan ses her med dokumentation:

<http://reactivex.io/documentation/operators.html>. (eller søg på Google). Det er ud over dette kursus at komme ind på alle disse.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Hvad er Observables?

Angular Docs (Angular's officielle dokumentation) beskriver Observables således:

*You can think of an observable as an array whose items arrive asynchronously over time. Observables help you manage asynchronous data, such as data coming from a backend service. Observables are used within Angular itself, including Angular's event system and its http client service. To use observables, Angular uses a third-party library called Reactive Extensions (RxJS). Observables are a proposed feature for ES 2016, the next version of JavaScript.*

Observables benyttes massigt i Angular. Først opsættes et lidt kunstigt eksempel, hvor der opsættes og subscribes på en observable i samme fil. Observables benyttes til http, DOM events, timers og meget andet. Så snart data er asynkrone, kan man benytte Observables.

Promises er en anden måde (alternativ til Observables) at lytte til asynkrone data. I Angular benyttes primært Observables - og knap så meget Promises.

Dybest set har Observables ikke opfundet sammen med Angular, men er i stedet en del af ES7.

En Observable åbner en kanal, over hvilken mange data over tid kan udveksles. Med en Observable får man en række metoder til at håndtere data. Metoder, der minder om Array-metoder.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## RxJS i Angular - Hello World

---

RxJS og Observables benyttes meget i Angular - og er derfor vigtige at kende til. Nogle gange skal man selv definere en Observable - andre gange ligger den implicit i koden.

Du finder bl.a. Observables her:

- ➡ HTTP - når data trækkes ind fra en JSON-fil
- ➡ EventEmitter
- ➡ @HostListener
- ➡ Forms (som gennemgået i særskilt kapitel)

Her er et simpelt eksempel:

### Angular: Observables RxJS

#### Returværdier

```
Allan Andersen, Albertslund
Allan Andersen, Albertslund
Bente Bredahl, Brøndby
Carla Christensen, Charlottenlund
Allan Andersen, Albertslund
Allan Andersen, Albertslund
Bente Bredahl, Brøndby
Carla Christensen, Charlottenlund
Bente Bredahl, Brøndby
Carla Christensen, Charlottenlund
Bente Bredahl. Brøndby
```

Figur 290

Først importeres Observable fra `rxjs/Observable`.<sup>29</sup>

```
import {Observable} from 'rxjs/Observable';
```

Figur 291

Når man trykker på knappen kaldes `start()`-funktionen, hvor en observable startes:

---

<sup>29</sup> Rxjs består af flere biblioteker. Ovenfor importeres /Observable. Man kan vælge at importere enkelte js-filer, bestemte biblioteker eller hele rsjs.

Sidstnævnte er bestemt det nemmeste, men selvfølgelig ikke det mest økonomiske i performance.

#### Angular 4 - Udvikling af Angular apps



```
start() {
 this.data = new Observable((observer) => {
 setTimeout(() => { observer.next("Allan Andersen, Albertslund"); },2000);
 setTimeout(() => { observer.next("Bente Bredahl, Brøndby"); },4000);
 setTimeout(() => { observer.next("Carla Christensen, Charlottenlund"); },4000);
 setTimeout(() => { observer.error(new Error('fejl'))};),8000);
 setTimeout(() => { observer.complete(); },10000);
 })
}
```

Figur 292

En Observable 'holder øje' med en datastrøm. Med en JavaScript SetTimeOut genereres handlinger med et tidsinterval.

Observer.next sender beskeder til **data (observer)**. Denne sendes så til alle subscribers af denne observable. Den holdes der her øje med 2 observers. Den første fremkommer efter 2 sekunder, de to næste 2 sekunder senere.

setTimeOut regner i millisekunder. 2000 betyder således 2 sekunder. I eksemplet er der 2 observers. Den første er sat til 2000, den næste til 4000. Starttidspunktet for disse observers er absolut i forhold til starttidspunktet. Den første starter efter 2 sekunder - den næste efter yderligere 2 sekunder - og ikke 4 sekunder efter den første observer.

Efter 10 sekunder kaldes **complete**, der afbryder ens observable.

Herefter laves den konkrete **subscribe**.

```
let subscription = this.data.subscribe(
 value => this.values.push(value),
 error => this.anyErrors = error,
 () => this.finished = true
);
```

Figur 293

Her lyttes på 3 ting:

**(1) Værdier.** values.push lytter på værdierne (som her er Allan...., Bente... og Carla...). Altså de værdier, som hentes ind.

**(2) Fejl.** error lytter til eventuelle fejl.

**(3) Afslutning.** () lytter på om subscribe er afsluttet. Her vil afslutning ske efter 20 sekunder. Sætter man ikke en afslutning, vil subscribe køre i hele browsersessionens levetid.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Subscribe i detaljer

---

Subscribe lytter/abonnerer på data. Ændres disse data, mens der lyttes vil subscribe reagere.

Subscribe har 3 parametre:

- ➡ 1 til at håndtere data
- ➡ 1 til at håndtere fejl
- ➡ 1 til at håndtere afslutning

Det kan også skrives således:

```
Subscribe(this.håndterData(),this.håndterFejl(),
this.håndterAfslutning())
```

I dette tilfælde vil der blive kaldt 3 metoder (håndterData(), håndterFejl() og håndterAfslutning())

## Observables og fejlhåndtering

```
start() {
 this.data = new Observable((observer) => {
 setTimeout(() => { observer.next("Allan Andersen, Albertslund"); },2000);
 setTimeout(() => { observer.next("Bente Bredahl, Brøndby"); },4000);
 setTimeout(() => { observer.next("Carla Christensen, Charlottenlund"); },4000);
 setTimeout(() => { observer.error(new Error('fejl'))},8000);
 setTimeout(() => { observer.complete(); },10000);
 })
}
```

Figur 294

Man kan benytte .error metoden til håndtering af eventuelle fejl. Her rejses blot en fejl. Det angives altså, at man efter 3 sekunder definerer en fejl.

```
let subscription = this.data.subscribe(
 value => this.values.push(value),
 error => this.anyErrors = error,
 () => this.finished = true
);
```

Figur 295

Under subscribe kan angives at this.anyErrors = error. Den rejste fejl kan herefter udskrives i templetten:

<h2>Fejl</h2> {anyErrors}	<b>Fejl</b> Error: fejl
------------------------------	----------------------------

Figur 296

## Unsubscribe()

Det er nemt at Unsubscribe(). Altså slukke for en subscribe().

```
setTimeout(() => {
 subscription.unsubscribe();
 this.interrupt="Subscribe er afbrudt";
}, 12500);
```

Figur 297

I eksemplet bliver subscription (som der subscribes på) kaldt med en unsubscribe()-metode efter 12,5 sekunder.

Alle observers, som måtte blive kaldt efter 12,5 sekunder, vil så ikke blive eksekveret, da selve subscription objektet jo er unsubscribed().

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Ovenstående testes bedst ved at ændre de 12,5 sekunder op og ned.

## Lidt baggrund: Call Stacks og Event Tables

---

JavaScript (som TypeScript bygger på) er som udgangspunkt synkront. Alle programlinjer udføres én efter én. Alle instruktioner fra programmet lægges i en Call Stack og udføres i den rækkefølge de kom ind.

Af og til benyttes *Asynkrone Callback-funktioner*. Altså en funktion hvor én af disse situationer opstår/ønskes:

- Callback-funktionen skal kaldes forsinket - altså ude i fremtiden. Enten fordi man venter på noget helt tredje - eller simpelt han fordi man ønsker funktionen bliver kaldt forsinket.
- Callback-funktionen kaldes med det samme, men får ikke svar tilbage med øjeblikkeligt. Måske kaldes data på en server - og disse har en svartid.

Sådanne funktioner placeres i en **Event Table**. Altså en liste over funktioner, der skal kaldes - eller hvor der ikke er kommet svar. Når disse funktioner på et tidspunkt kan udføres, føres de tilbage til **Call Stacken** (denne proces kaldes **Event Loop**).

Uden Event Table og Event Loop ville browseren blive blokeret før eller siden.

Event table kan sammenlignes med en ToDo-liste. Synkrone handlinger kunne være:

- Fotokopiere nogle papirer (synkron)
- Rydde op på skrivebordet (synkron)
- Tilmelde sig julefrokost (synkron)
- Ringe til en kunde og få bekræftet en ordre (asynkron)
- Underskrive papirer vedr. kontrakt (synkron)
- Lave et Angular program til en kunde (asynkron)
- Privat - sms'e hjem at man har overarbejde i dag (synkron)

I det her eksempel er de fleste handlinger synkrone. At få bekræftet en ordre hos en kunde er i den gang asynkron - da kunden måske slet ikke er på kontoret i dag - eller først skal spørge chefen, som i øvrigt er på ferie.

At lave et Angular program er her Asynkront, da der altid skal testes og måske googles lidt.

Disse 2 handlinger bliver måske udsat. Man kan måske først ringe til kunden i morgen. Og Angular-programmet må vente, fordi der er for meget andet 'halløj' på kontoret den dag.

## Design Pattern

Observables er et Design Pattern, hvor et objekt (kaldet Subject), vedligeholder en liste af observers. Hver af disse observers får automatisk besked om ændringer i deres tilstand (f.eks. kører, fejl, afsluttet).

## Alternativer til Event Loop

### Callback-funktioner

Man kan benytte callback-funktioner. Disse er asynkrone. Callback-funktioner er funktioner i funktioner. Callback-funktioner giver ofte en masse funktioner i funktioner. Koden bliver hurtig uoverskuelig og svær/umulig at fejlsøge på.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



```
1 hentNoget(function(result){
2 hentEndnuMere(function(resulter){
3 hentMereRekursivtForHvertResultat(resulter, function(endnuMere){
4 | Puha_EndeligKanViLaveNoget(endnuMere);
5 });
6 });
7 });
```

Figur 298

## XMLHttpRequest

---

Denne metode benyttes i forbindelse med AJAX - og været fremme i mange år. Den er gennemtestet og fungerer. Metoden er dog meget simpel og er tit bundet snævert op mod et givent API.

## Fetch API

---

Fetch API muliggør også asynkrone kald ved hjælp af Promises.

## Promises

---

Promises kan benyttes i Angular. De har en del ulemper og et par fordele. Først og fremmest er det ikke muligt at afbryde en Promise .

På nettet diskutes det hæftigt om en Promise kan afbrydes. Det kan muligvis gøres med 3.parts produkter/libraries - men som sådan er det umuligt.

Promises er også bedre til at håndtere store mængder data, arrays og den slags.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## RxJS tilbyder en robust infrastruktur

Da Angular i sin natur består af mange komponenter, services m.m., er det essentielt at kunne udveksle data.

RxJS importeres i et komponent eller en service. Dermed kan man gøre brug af Observables. RxJS forener Callbacks, XMLHttpRequest, Promises, DOM Events, Web Workers og Web Socket i et framework.

RxJS er udviklet af bl.a. Microsoft og Netflix.

## RxJS skal installeres først

```
git clone https://github.com/Reactive-Extensions/rxjs.git
```

Figur 299

Man kan clone RxJS fra **Github**. Herefter installeres RxJS med:

```
npm install rx $ npm install -g rx
```

Figur 300

På nedenstående link kan man læse den officielle github dokumentation.

<https://github.com/Reactive-Extensions/RxJS>

## Observables versus Promises

Observables og Promises er lidt det samme og alligevel ikke.

En Observable har både en **Subscribe()** og en **Unsubscribe()**. Det har en Promise ikke.

Observables har en **retry-metode** til at genkalde en observer. Det har en Promise ikke.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Hvad er Promises? Opsummering

---

- ➡ Returnerer en enkelt værdi
- ➡ Man kan tilføje `.toPromise()` til en observable og på den måde returnere en Promise i stedet for.
- ➡ Klassisk AJAX benytter Promises
  
- ➡ Man kan ikke cancel en Promise
- ➡ Promises har kun **success** og **error** handling / Reject - Resolve
- ➡ Kører funktioner asynkront, og bruger returværdien eller fejlen - men kun 1 gang.
  
- ➡ Ikke Lazy
- ➡ Kan ikke køres igen (retry). En Promise skal have adgang til den oprindelige funktion for at lave en retry.

## Hvad er Observables? Opsummering

---

- ➡ Observable virker med mange værdier over tid (stream)
- ➡ Observables kan afbrydes (cancel)
- ➡ Understøtter operatorer som map, filter, reduce m.fl.
- ➡ Del af EcmaScript2016
- ➡ Observables benytter sig af RxJS (Reactive Extensions)
- ➡ Observables er et array, hvis elementer ankommer asynkront over tid
- ➡ Finally Handling
- ➡ Lazy
- ➡ Kan oprettes manuelt eller fra f.eks. events
- ➡ Observables er funktioner, som kan subscribes nu eller senere
- ➡ `get`, `post`, `put` og `delete` returnerer Observables

Angular benytter Rx.js Observables i stedet for promises sammen med http.

Se liste over alle RxJS operatorer i Appendix.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

# Kapitel 34 Stateful og Stateless komponenter

Komponenter kan være **stateful** og/eller **stateless**. Sådanne komponenter kan også kaldes smart/dumb eller container/presentational eller impure/pure funktioner.

Disse begreber findes også i andre sammenhænge/frameworks end Angular.

## Hvad er stateful?

Er noget stateful, er det en programstump/komponent, som gemmer information om komponentets tilstand. Komponentet har evnen til at ændre dette. Man kan sige, at et stateful komponent er levende, har viden om hvad der tidligere er sket, hvad der sker lige nu og fremtidige ændringer af en tilstand/state.

Kendetegnende for en stateful funktion er:

- ➡ Kører state-ændringer gennem stateless-funktioner
- ➡ Styrer data
- ➡ Kan indeholde default værdier
- ➡ Har viden om state - hvor er man henne/nået til?
- ➡ Kan kommunikere med DI

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Hvad er stateless?

---

Når noget er stateless, forholder komponentet sig kun til sin interne tilstand/state, men ændrer ikke varigt på det. Dette tillader, at man kan referere til komponentet udefra. Med det samme input udefra vil et stateless element altid give samme returværdi/output. Data passerer nærmest blot igennem. Et stateless komponent opbevarer altså ikke information på nogen måde - ej heller om historik og fremtidige tilstænde.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 34 Stateful og Stateless komponenter

Kendetegnende for en stateless funktion er:

- Den kan let testes med testdata (mockdata)
- Den kan genbruges x antal gange, og vil give samme resultat med samme input
- Input-parametre er fast defineret
- Output-parametre er fast defineret

#### Angular 4 - Udvikling af Angular apps



## Kapitel 35 Oprettelse af egne moduler

Indtil nu er en række standardmoduler blevet gennemgået - herunder Router, Response, Http og flere.

I det følgende vises, hvordan man kan lave sit eget library og sætte det op som et modul. Fordelen er, at man kan lave komponenter, services, directives og meget mere, som kan benyttes på tværs af projekter.

Indtil nu har komponenter m.m. kun virket inden for et projekt.

### Opret et nyt projekt

---

Med CLI oprettes et nyt projekt:

```
ng new FrameForEgetModul --prefix zoom
```

Figur 301

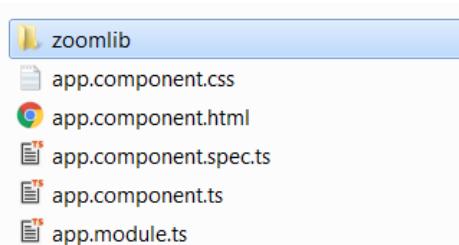
--prefix betyder at alle komponenters selector vil starte med zoom-

Når projektet er installeret, går man ind i projektet med kommando-prompten. Herfra laves et nyt modul med **ng generate**.

```
ng g module zoomlib
```

Figur 302

Et helt normalt projekt er oprettet. Eneste forskel er, at projektet også indeholder et modul:



Figur 303

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 35 Oprettelse af egne moduler

ZoomTek leverer kurser til A.P.Møller, Novo, Danske Bank, Vestas, Forsvaret, Grundfos,

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



zoomlib-mappen indeholder en module.ts fil:



Figur 304

Gå ind i src/app/zoomlib mappen fra prompten og opret et ny komponent med ng g c eksempel.

FrameForEgetModul\src\app\zoomlib>ng g c eksempel\_

Herfra kan man oprette child-komponenter, services, directives, css-filer og meget andet.

For at demonstrere princippet, laves her en Hello World. Der ændres blot i templet på  
src/app/zoomlib/eksempel/eksempel.component.ts

```
3 @Component({
4 selector: 'zoom-eksempel',
5 template: `<h1>Eksempel komponent fra zoomlib</h1>`,
6 styleUrls: ['./eksempel.component.css']
7 })
8 export class EksempelComponent implements OnInit {
```

Figur 305

Der er lavet simple, synlige ændringer i form af en ændring af templet. Yderligere ændringer og features har ikke direkte noget med emnet at gøre.

module.ts i modulet, som i dette eksempel hedder eksempel.module.ts skal **omdøbes til index.module.ts** før modulet publiseres.

## Eksport af komponentet

Lidt uvant fra tidligere pensum, skal komponentet **eksporterdes**. Dette gøres i module.ts-filen.(src/app/zoomlib/zoomlib.module.ts). Vær omhyggelig med at udfylde og placere filer rigtigt. Det mest bøvlede ved at lave et projekt er stier og filnavne.

Men i module.ts filen tilføjes **exports**. Navnet EksempelComponent er naturligvis klassenavnet fra eksempel.component.ts..

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 35 Oprettelse af egne moduler

```

5 @NgModule({
6 imports: [
7 CommonModule
8],
9 declarations: [EksempelComponent],
10 exports: [EksempelComponent]
11 })
12 export class ZoomlibModule { }

```

Figur 306

## Publisering af modul til github (offentligt modul)

Følgende er lavet indtil nu:

- Projektet **FrameForEgetModul** er oprettet.
- modulet **zoomlib** er oprettet i dette projekt med **ng g module zoomlib --prefix zoom**
- komponentet **eksempel** er lavet og små visuelle ændringer er indsat i templatene.
- **module.ts-filen for modulet er tilpasset** (der er 2 module.ts filer). Der er ændret i module.ts filen, som ligger i zoomlib-mappen). exports er tilføjet.

## Package.json fil kopieres

Herefter kopieres package.json filen fra projektet ind i zoomlib biblioteket.

```

2 "name": "zoomlib",
3 "version": "1.0.16",
4 "description": "Angular library",
5 "scripts": {
6 "start": "ng serve",
7 "build": "ng build",
8 "test": "ng test",
9 "lint": "ng lint",
10 "e2e": "ng e2e"
11 },
12 "dependencies": {
13 "@angular/animations": "^4.0.0",
14 "@angular/common": "^4.0.0",

```

Figur 307

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Kapitel 35 Oprettelse af egne moduler

Sæt et passende navn, ændr eventuelt versionen. Og ændr **private** til **false**. Sidstnævnte betyder reelt, at modulet bliver *public*.

Fra zoomlib-biblioteket i prompen køres en **npm init**.

```
Press ^C at any time to quit.
name: (zoomlib) soren
version: (1.0.16)
description: Sørens demo af moduler
entry point: (index.js)
git repository:
keywords:
+-----+
```

Figur 308

Værdierne lægges i package.json.

Afslut med **npm publish**

```
\FrameForEgetModul\src\app\zoomlib>npm publish
```

Figur 309

Har man ikke en konto på npmjs.com, vil man blive bedt om at oprettet en sådan, inklusive at bekræfte ens email m.m. Før man har gjort det, vil npm publish give fejl.<sup>30</sup>

Når projektet er publiseret, kan man efterfølgende importere hvorsomhelst fra. Man kan gå til et andet projekt og skrive:

```
npm install --save sorenb
```

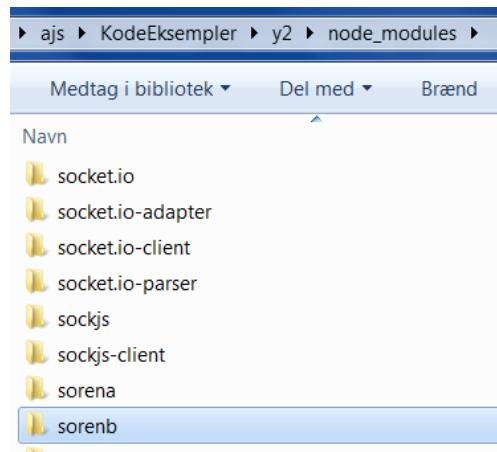
Figur 310

**sorenb** svarer til name-parameteren i package.json. Når man publiserer, er der 2 krav. Dels skal man have en package.json fil i projektet - og denne skal have et name-property, som her er sorenb.

Hvert projekt, der publiseres, skal have et unikt navn på npmjs.com (her sorenb). For at publisere skal man være oprettet som bruger. Dette kan gøres online på npmjs.com eller med kommandoen npm adduser. Bemærk - man kan godt have flere brugernavne pr. email-adresse.

<sup>30</sup> med npm adduser kan man oprette en ny bruger på github. med npm init kan man redigere name-paramteren i package.json

Herefter vil modulet være installeret under node\_modules



Figur 311

I module.ts skal komponenter, directives og andet importeres som vanligt:

```

1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { AppComponent } from './app.component';
4 import { EksempelComponent } from 'sorenna/eksempel/eksempel.component';
5
6 @NgModule({
7 declarations: [
8 AppComponent,
9 EksempelComponent
10],

```

Figur 312

Til sidst tilføjes html-element, der peger på child-komponentet - eller den kode, som i øvrigt er nødvendig. Men koden adskiller sig ikke fra vanlig Angular.

```

3 @Component({
4 selector: 'app-root',
5 template: `
6 <h1>{{ title }}</h1>
7 <app-eksempel></app-eksempel>
8 `,

```

Figur 313

Og resultatet er:

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Angular: app!!

eksempel works!

ZoomTek Kurser

Figur 314

## Publisering af modul til Node.js (privat modul)

```
src\app\zoomlib>npm init --scope=zoomlib
```

Figur 315

skriv følgende kode. Herefter skrives npm publish. Følgende fejl vil komme frem:

```
You need a paid account to perform this action.
```

Figur 316

Herefter kan man til hvert en tid installere modulet i et givent projekt:

```
npm install --save zoomlib
```

Figur 317

npm install --save zoomlib køres fra roden af et givet projekt.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Kapitel 36 Brug af 3.parts moduler

<https://nolanlawson.com/2016/08/15/the-cost-of-small-modules/>

Der findes 1000-vis af moduler og projekter, man kan installere. Et modul bør være selvbærende og ikke afhængig af dependencies. Men sidstnævnte kan være et problem.

<https://github.com/valor-software/ng2-plans>

På denne side kan man downloade nogle spændende komponenter.  
Søg evt. på

### Welcome ng2 modules coordination repository

Figur 318

Klikker man på ng2-charts får man mulighed for at downloade og installere komponenter og biblioteker til at lave grafer og diagrammer med - midt på sine Angular sider.

Oftest følger en vejledning med.

```
npm install ng2-charts --save
```

```
npm install chart.js --save
```

Figur 319

Et modul kan oprettes med denne CLI-kommando:

Ng generate module MitModul

```
>ng generate module MitModul
```

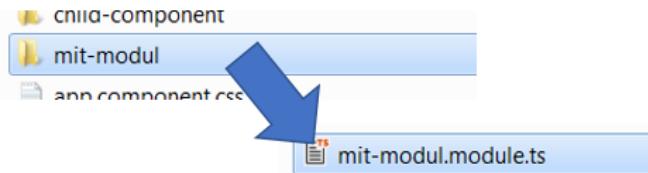
Figur 320

Der oprettes et directory `src/app/mit-modul` med filen `mit-modul.module.ts` indeni.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.





Figur 321

```
installing module
create src\app\mit-modul\mit-modul.module.ts
WARNING Module is generated but not provided, it must be provided to be used
```

Figur 322

Et modul skal manuelt importeres i module.ts filen.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Public API Surface

Angular Teamet har angivet en samling af packages som **Public API Surface**.

Disse kan findes på dette link:

[https://github.com/angular/angular/blob/master/docs/PUBLIC\\_API.md](https://github.com/angular/angular/blob/master/docs/PUBLIC_API.md)

Nedenfor er de listet. Listen ændres hele tiden og er ikke nødvendigvis 100% opdateret her.

- ➡ @angular/animations
- ➡ @angular/core
- ➡ @angular/common
- ➡ @angular/platform-browser
- ➡ @angular/platform-browser-dynamic
- ➡ @angular/platform-server
- ➡ @angular/platform-webworker
- ➡ @angular/platform-webworker-dynamic
- ➡ @angular/upgrade
- ➡ @angular/router
- ➡ @angular/forms
- ➡ @angular/http @angular/
- ➡ @angular/

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



# Kapitel 37 Build og Publish af projekter

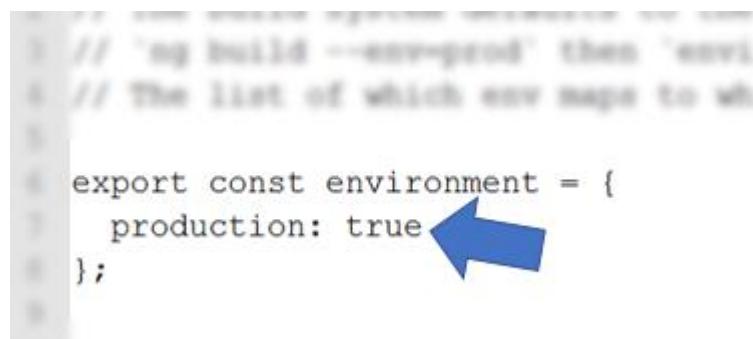
Når et projekt på et tidspunkt er færdigt, skal det i drift. Her gennemgås processen for overgang fra *developer mode* til *production mode*.

## Tilpasning af main.ts og environment.ts

I main.ts tilføjes

```
1 import { enableProdMode } from '@angular/core';
2 import { platformBrowserDynamic } from '@angular/platform-
3 import { AppModule } from './app/app.module';
4 import { environment } from './environments/environment';
5
6 if (environment.production) {
7 enableProdMode();
8 }
9
10 platformBrowserDynamic().bootstrapModule(AppModule);
```

Afhængig af CLI-version vil denne kode allerede være tilføjet.  
Værdien af **environment.production** kan sættes i  
**src/environments/environment.ts**



```
// 'ng build --prod' then 'www'
// The list of which now maps to wh
export const environment = {
 production: true
};
```

Man behøver dybest set blot at køre følgende npm kommando:

**ng build**

Denne kommando opretter *dist*mappen (distribution)

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

**ng build** kan have mange forskellige parametre. Man kan liste disse og overføre listen med forklaring til udklipsholderen med følgende kommando:

```
ng build --help | clip
```

Herefter kan man nemt paste indholdet ind i notepad eller lignende.



Hele projektet er nu samlet i 5 filer (+index.html + favicon). Disse 5 filer refereres der til i index.html.

Navn	Størrelse
index.html	1 KB
favicon.ico	6 KB
inline.bundle.js	6 KB
main.bundle.js	10 KB
polyfills.bundle.js	198 KB
styles.bundle.js	13 KB
vendor.bundle.js	2.127 KB

Disse filer bærer hele projektet og kan overføres til webserveren.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.





ZoomTek Kurser

Her er filerne overført til ZoomTek.dk, der hostes på one.com. Overførslen er sket manuelt med ftp. One.com's webserver er Apache.<sup>31</sup>

**ng build --prod --stats-json**

Med denne kommando laver man også et build. Men løsningen bliver compressed.

---

<sup>31</sup> Det er ud over denne bog, at beskrive opsætning af webserver. Af webservere kan nævnes Microsoft IIS, Apache, IBM Domino m.fl.

## Kapitel 37 Build og Publish af projekter

Med disse filer kan man deploye en Angular app på en given webserver.

--prod parameteren får CLI til at lave projektet om til *production mode*.

--stat-json opretter en stats.json fil i dist-mappen. Denne fil indeholder et væld af informationer. Filen er ret stor. På en simpel HelloWorld-app fylder den her i testen 18.000 linjer.

CLI benytter Webpack til at lave bundling af filer. Alså den proces, som samler filerne til en produktionsserver.

## Webpack Bundle Analyzer

```
npm install --save-dev webpack-bundle-analyzer
10 "title": "ng title",
11 "bundle-report": "webpack-bundle-analyzer dist/stats.json",
12 "e2e": "ng e2e"
```

```
npm run bundle-report
```

Resultatet er følgende:



Hver farve repræsenterer en bundle. Dem er der 3 af. Vendor-delen som indeholder al node-modules kode. Polifills-delen og selve applikationen.

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagere på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

Lader man musen køre hen over de enkelte dele, vil man se yderligere detaljer.



Her kan ses størrelsen af de enkelte dele i applikationen. Man kan se dele, som ikke benyttes - og som kan fjernes.

# Appendix

## Appendix A : Kursusoversigt

# Appendix A Kursusoversigt

Her finder man alle vore kurser

<http://zoomtek.dk/kurser>

The screenshot shows the ZoomTek website's course overview page. At the top, there's a header with the ZoomTek logo, a search bar, and navigation links for 'Forside', 'Kurser' (selected), 'Konsulent', 'Referencer', 'Kampagner', 'Produkter', 'Artikler', 'Nyheder', 'Om Os', 'Ressourcer', and 'Kontakt'. Below the header, there's a sidebar with information about the selected course ('Mere info om dette kursus') and links to 'Praktisk information', 'Gratis hotline efter kurset', 'Afholdelsesgaranti 30 dage', 'For Kurset: Tilpasning og kompetenceafklaring direkte med instruktøren', 'Under Kurset: Levende instruktør, der kan gå ud over pensum', 'Efter Kurset: Gratis hotline, Cloud-Share og ekstra kursusmateriale', 'FlexDage', 'Klippekort - rabat på kurser', 'Download brochure om ZoomTek', and 'Ordbog for ikke-nørder'. The main content area is titled 'Kursusoversigt - 150 kurser' and features several categories of courses arranged in a grid:

- SharePoint Kurser (27 kurser)**: SharePoint Designer, Visual Studio .NET, Administration, Brugerkurser, Workflows, Webpart udvikling, InfoPath, Expression Web
- Webudvikling (47 kurser)**: HTML5, CSS3, Responsive, Web Design RWD, AngularJS, jQuery, php, LESS, SASS, JavaScript, Dojo
- XML (20 kurser)**: XML Schema, XSLT 1.0+2.0, ODBC, OLEDB, Java-/Javaintegration
- Web Services (3 kurser)**: XML Web Services, REST, SOAP\*, WS-Security, WSDL, UDDI
- Programmering (6 kurser)**: C#/.NET, Java, PHP, LotusScript, JavaScript
- CMS (5 kurser)**: Joomla, Wordpress, Drupal, Site Core, Magento, Webshop
- Microsoft Office (29 kurser)**: Outlook, Word, Excel, PowerPoint, Access, OneNote, Project, Office 365, Lync/Skype for Business, Publisher, OneDrive, Visio, InfoPath
- IBM Notes (23 kurser)**: Domino Designer, Administration, Helpdesk, Webudvikling, Brugerkurser, Connections, XPages
- Open Office (8 kurser)**: Base, Calc, Impress, Writer, LibreOffice
- Grafiske Kurser (10 kurser)**: GIMP, Acrobat, GoLive, Illustrator, InDesign, PageMaker, Photoshop, Dreamweaver, Flash
- Powershell (3 kurser)**: PowerShell Grundkursus, PowerShell til SharePoint, Desired State Configuration
- Drupal (15 kurser)**: Kurser i udvikling

At the bottom of the page, there's a footer section with logos for 'Business Partner IBM' and 'Microsoft Partner Network', and a large red 'Flex Dage' logo. The footer also contains the text 'Kurser on-site i hele Danmark'.

Figur 323

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Appendix B Single Page vs MultiPage

### Single-Page vs MultiPage Design

- Hvad er One-Page Design/Single-Page Design (SPA)?
- Kan SPA og MPA kombineres?
- Fordele ved SPA (=ulemper ved MPA)
- Ulemper ved SPA (=fordele ved MPA)

Figur 324

### Hvad er Single-Page Design (SPA)

- Forsøger at give mindst mulig information til brugeren, som denne skal reagere på.
- SPA-sider har ikke yderligere sider
- Eksempler: <https://onepagelove.com/gallery/most-loved/page/2>
- Info hentes ind f.eks. ved:
  - Modal Views
  - Accepterede ekstrasider - f.eks. sprogversioner i subdomæner f.eks. fr.zoomtek.dk (zoomtek på fransk - not)
  - AJAX-import af forskellig slags (dynamisk routing m.m.)
  - "Impressum"-websites - handelsbetingelser m.m.

Figur 325

### Hvorfor SPA (måske) er fremtiden

- *The one where we finally free websites from the outdated conventions of print design and fully utilize the digital platform they're built on. Where we kick archaic elements like pages to the curb and instead create unique, satisfying, web-native experiences.*  
(citat: www.dtelepathy.com)

Figur 326

- Brugeroplevelse er nemmere, for der er kun 1 side
- Ofte mere responsive end MPA.... da scrolling er primære navigationsmiddel

Figur 327

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Appendix B : Single Page vs MultiPage

## Hvad afgør valg mellem SPA/MPA

- Mængden af information. Mere taler for MPA
- Hierarki. Er navigationen omfattende? Mere taler for MPA
- Skal al information være tilgængelig for alle. Ja.. taler for MPA

Figur 328

## Fordele ved SPA

- Simpelt design - kører (normalt) bedre på mobil
- Nemmere at fortælle en historie, med kort indledning
- Giver ofte højere Conversion Rates...
- Lettere Iterations

Figur 329

## Ulemper ved SPA

- Kun til basic hjemmesider. Bedst til newsletter, contacts
- Dårlig til lange tekster, mange produkter, produktvarianter
- Markant dårligere SEO-performance
- SPA er ikke Scalable på samme både. Svært at indsætte 65 nye produkter med produktbeskrivelser.
- Sværere at analysere - f.eks. med Google Analytics og lignende

Figur 330

## Angular 2 - og SPA/MPA

- Normalt forbinderes Angular 2 med SPA udvikling. SPA er som sådan et komponent med subkomponenter.
- MEN et klassisk MPA website som ZoomTek.dk kan sagtens (og med fordel) være bygget på Angular 2 - og indeholde komponenter
- Mikser man SPA/Angular 2 med MPA taler man om **Hybrid Applikationer**.

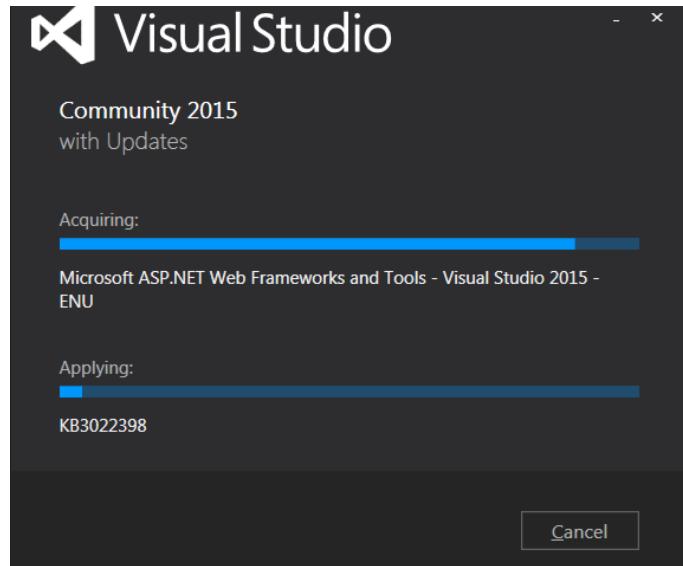
Figur 331

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Appendix C Opsætning af Visual Studio og Node.JS



Figur 332

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Appendix D NPM Kommandoer

### npm init

Opretter en ny package.json fil. Bruges i en tom mappe til at oprette et nyt modul. Efterfølgende spørges om de enkelte parametre i filen.

```
C:\Users\Kursist 1\AJ5\slet>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.
See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package
and save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (slet) demo_af_npm_init
version: (1.0.0) 2.3.4
description: npm init bruges til at oprette en ny package.json
entry point: (index.js) _
```

Figur 333

```
1 {
2 "name": "ulla",
3 "version": "2.1.2",
4 "description": "dette er en test",
5 "main": "a",
6 "scripts": {
7 "test": "b"
```

Figur 334

kommandoen **npm init --yes** gør det samme, men udfylder med testdata. Man slipper for at indsætte data i command prompten, men kan blot rette filen til efterfølgende.

## NPM Kommandoer II

<b>npm start</b>	runs the compiler and a server at the same time, both in "watch mode"
<b>npm run tsc</b>	runs the TypeScript compiler once
<b>npm run tsc:w</b>	runs the TypeScript compiler in watch mode; the process keeps running, awaiting changes to TypeScript files and recompiling when it sees them
<b>npm run lite</b>	runs the lite-server, a light-weight, static file server with excellent support for Angular apps that use routing
<b>npm run typings</b>	runs the typings tool separately
<b>npm run postinstall</b>	called by npm automatically after it successfully completes package installation. This script installs the TypeScript definition files defined in typings.json

Figur 335

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udriftning af 1 eksemplar udelukkende til eget brug er tilladt.



- npm -v → version af npm
- node -v → version af nodejs
- npm install -g typescript → installerer typescript globalt (skal downloades først)
- git clone [link] → downloader package fra github (installerer ikke - kun download)
- npm install → installerer package i det bibliotek, hvor du er placeret. Kigger i package.json for at downloade dependencies (nødvendige filer).
- npm -g install --save-dev gulp-install (installerer gulp globalt)

Figur 336

## samlet liste over npm kommandoer

---

access, adduser, bin, bugs, c, cache, completion, config, ddp, dedupe, deprecate, dist-tag, docs, edit, explore, get, help, help-search, i, init, install, install-test, it, link, list, ln, login, logout, ls, outdated, owner, pack, ping, prefix, prune, publish, rb, rebuild, repo, restart, root, run, run-script, s, se, search, set, shrinkwrap, star, stars, start, stop, t, tag, team, test, tst, un, uninstall, unpublish, unstar, up, update, v, version, view, whoami

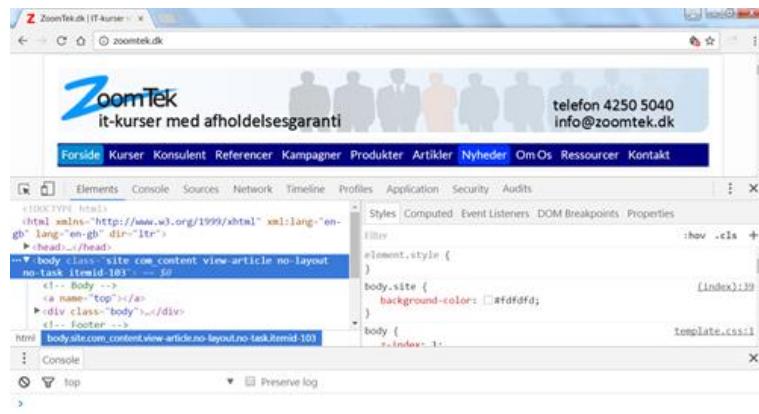
### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Appendix E : Google Developer Tools - DevTools

## Appendix E Google Developer Tools - DevTools

Google Developer Tools - herefter DevTools startes i Google Chrome med F12:



Figur 337

## Menuer



Figur 338

Øverst findes en menu, som bliver gennemgået nedenfor.

## Device Toolbar

Få vist dit website fra forskellige devices. Tryk på ikonet:



Figur 339

Herefter fremkommer en liste over forskellige telefoner.  
Herigennem kan man se sitet, som det vil se ud på de pågældende  
telefoner. Der er mulighed for at zoome m.m.

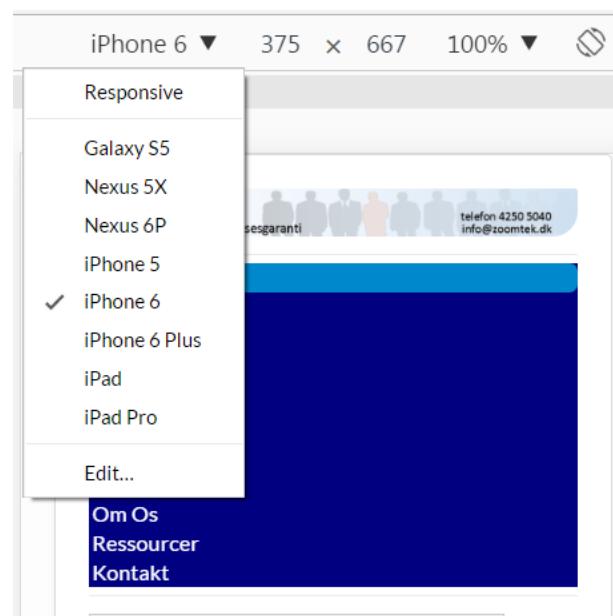
## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## DevTools

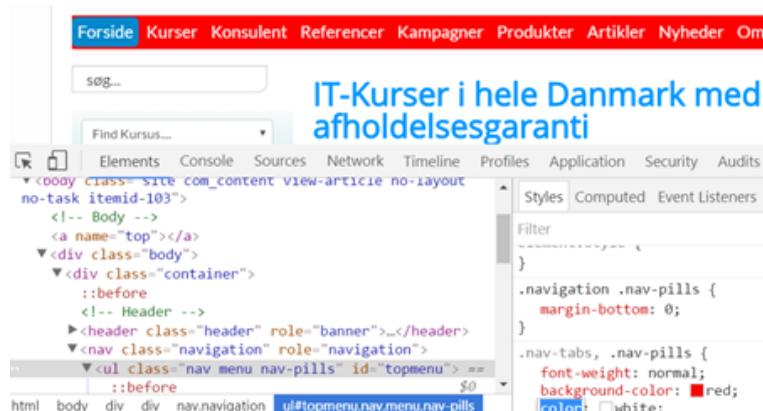
### Appendix E : Google Developer Tools -



Figur 340

## Elements

Kør igennem dit site og peg på html-elementer. Manipuler med disse on the fly



Figur 341

Her er top-baren (class nav-pills) ændret til rød. Selvfølgelig kun mens testen kører.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Appendix E : Google Developer Tools - DevTools

## Console

The screenshot shows the DevTools interface with the 'Console' tab selected. At the top, there are icons for back, forward, and refresh, followed by tabs for 'Elements', 'Console' (which is underlined in blue), 'Sources', 'Network', 'Timeline', and 'P'. Below the tabs, there is a search bar with a magnifying glass icon and a dropdown menu labeled 'top'. A checkbox labeled 'Preserve log' is checked. The main area displays a log entry:

```
> console.log("Hej Konsol");
Hej Konsol
↳ undefined
> |
```

Figur 342

Man kan benytte simpel JavaScript i loggen:

The screenshot shows the DevTools interface with the 'Console' tab selected. The log entries are:

```
> var i=20;
↳ undefined
> console.log(i)
20
```

Figur 343

Man kan også lægge en `console.log()` kommando ind i JavaScript/TypeScript/Angular. Som udgangspunkt skrives gentagne, ens beskeder som én entry med en angivelse af antal ved siden af.

The screenshot shows the DevTools interface with the 'Console' tab selected. The log entry is:

```
> for (var i=0;i<10;i++) {console.log("ZoomTek")}
10 ZoomTek
```

Figur 344

Vælg 'show timestamps' i DevTools Indstillinger for at slå dette fra.

Vælg



Figur 345

For at beholde loggen mellem sessioner. Højreklik og vælg Save As for at gemme historikken.

Vælg `clear()`, `console.clear()` eller højreklik og vælg 'clear' for at tømme konsollen.

The screenshot shows the DevTools interface with the 'Console' tab selected. The log entry is:

```
> |
```

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Figur 346

Øverst kan man vælge 'kontekst' - her 'top'. Har man iframes e.l. på skærmen, kan man vælge disse iframes - med tilhørende variable m.m.

Af andre muligheder kan nævnes:

- ➡ Filtrering
- ➡ Indstillinger i Konsollen

## Console

- Hide network messages
- Log XMLHttpRequests
- Preserve log upon navigation
- Show timestamps
- Enable custom formatters

Figur 347

## Appendix F Patterns / Antipatterns

Anti-patterns er specielle metoder og udviklingsstandarder, som anses for at være dårlig programmeringsskik. Anti-patterns er noget negativt, som man skal undgå.

Anti-patterns er det modsatte af Design Patterns, som dækker over anerkendte metoder til løsning af givne problemer. Anti-patterns er det modsatte, og noget man søger at undgå.

Inden for klassisk objektorienteret programmering OOP, er det ofte et mål at splitte kode ud i mindre enheder. Derfor laver man flere klasser, hver med relaterede funktioner/members.

Laver man en God Class, der indeholder alt, vil man miste fleksibilitet og opnå større kompleksitet, ringere fejlsøgningsmuligheder, dårligere hukommelsesstyring og meget andet uønsket.

```
class GodKlasse {
 member PerformInitialization() {}
 member ReadFromFile() {}
 member PerformCalculation() {}
 member ValidateInput() {}
 member WriteToFile() {}
 member DisplayToScreen() {}
}
```

Figur 348

I stedet bør man opdele en sådan klasse i flere klasser med hver deres members.

## Angular og Design Patterns

Angular er i sig selv et design pattern. Konkurrenter til Angular er f.eks. AngularJS og React. Begge er Javascript-biblioteker. Koden laves i JavaScript, og man har ikke noget valg.

Angular er et *Framework*. Der findes flere implementeringer til Angular:

- ➡ TypeScript (så afgjort den mest benyttede)
- ➡ Dart
- ➡ Ren JavaScript (EcmaScript)

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagere på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Appendix G Eksterne Programmer

Der findes en række eksterne redskaber/produkter til Angular:

- ➡ Plunkr
- ➡ Github
- ➡ e2e
- ➡ Jasmine
- ➡ Karma
- ➡ Protractor
- ➡ Webpack
- ➡ Grunt
- ➡ Gulp
- ➡ Rollup,
- ➡ Browserify,
- ➡ Webpack,
- ➡ Closure
- ➡ Yargs

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Appendix H Node.js

Warning kan have flere indikatorer:

```
npm WARN optional Skipping failed optional dependency /@angular/cli/chokidar/fsevents:
npm WARN notsup Not compatible with your operating system or architecture: fsevents@1.1.2
```

Figur 349

### Optional

Fejlen er ikke væsentlig. Alt er relativ, men ofte er der tale om et begrænset problem.

### Notsup

Not Supported. Ovenfor kan ses af fsevents@1.1.2 ikke er supporteret af OS. Her anbefales det at google fsevents - og tage stilling til problemet herigennem.

#### Tips & Tricks

##### Kopier med PowerShell

Normalt benyttes CMD til Node.js/NPM. CMD benyttes også til Powershell. Derfor kan man frit benytte PowerShell-kommandoer sammen med NPM.

F.eks. kan man ikke kopiere fejlmeddelelser og andet i CMD-vinduet. Dette løses nemt med | clip.

```
ng serve --help | clip
```

ng serve --help skriver hjælp på skærmen. | clip kopierer udskriften til udklipsholderen.

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Appendix I    Opsætning af Angular miljø Uden CLI

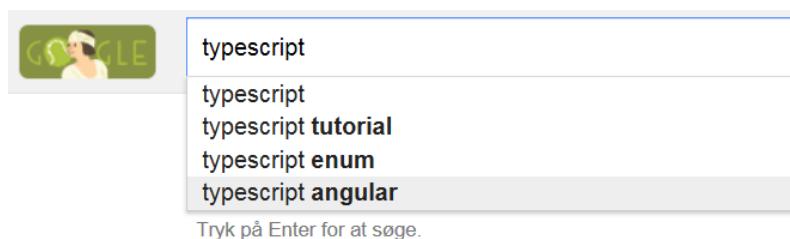
### Bemærk om Angular 2 vs. Angular 4

Instruktioner i dette kapitel er lavet med udgangspunkt i Angular 2. Fremgangsmåden er den samme i Angular 4. Der kan være små forskelle i programmer og kommandoer.

I Angular 4 (og ofte også i Angular 2) vil man opsætte Angular **med** CLI.

### Download og installér TypeScript

TypeScript er det sprog, man koder Angular i. TypeScript lever i runtime automatisk kompileret til JavaScript, som er det eneste browsere forstår.



Figur 350

Vil man teste en allerede installeret Typescript version, gøres dette med

**tsc -v**

[TypeScript - JavaScript that scales.](https://www.typescriptlang.org/)

<https://www.typescriptlang.org/> ▾ Oversæt denne side

TypeScript starts from the same syntax and semantics that millions of JavaScript do today. Use existing JavaScript code, incorporate popular ...

[Playground](#)

let greeter = new Greeter("world");  
let button = document.

[Docs](#)

Quick start. Let's get started by building a simple web ...

[Flere resultater fra typescriptlang.org »](#)

Figur 351

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

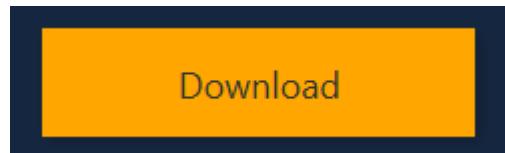


Uden CLI

## Appendix I : Opsætning af Angular miljø



Figur 352



Figur 353

## Get TypeScript



Figur 354

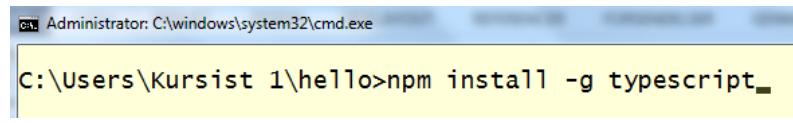
## Installering af TypeScript

TypeScript installeres nemt med **npm install -g typescript** fra Node.js konsollen.

### INSTALL

```
npm install -g typescript
```

Figur 355



Figur 356

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Appendix I : Opsætning af Angular miljø Uden CLI

```
C:\Users\Kursist 1\hello>npm install -g typescript
(node:5444) fs: re-evaluating native module sources is not supported. If you are using the graceful-fs module, please loadRequestedDeps = after
```

Figur 357

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Installation af Angular

For at installere Angular skal man bruge en package.json fil. Denne json-fil fortæller, hvordan angular skal installeres og hvilke filer, der skal hentes.

Når man har package.json filen, kan man køre en 'npm install' fra node.js. Den nemmeste (ikke nødvendigvis rigtige) package at downloade er:

<https://www.github.com/angular/quickstart>

### Køreplan for installation af Angular

På Google søger man på

**package.json angular2**

Figur 358

Herefter vil der fremkomme forskellige *boilerplates* og *packages*, som man kan installere.

GitHub - angular/quickstart: Angular 2 QuickStart - source from the  
<https://github.com/angular/quickstart> Proxy Fremhævet  
Install the npm packages described in the package.json and verify that it works:  
npm install npm start. Doesn't work in Bash for Windows which does not support ...

Figur 359

Her er en lovende quickstart. Dels ved jeg, at **quickstart** er en brugbar package - da det nemlig er en samling HelloWorld eksempler. Dels kører den via GitHub, hvilket gør installation nem. Tryk på linket.

Her er en anden package (noget ældre version).

`>git clone https://github.com/mschwarzmueller/angular-2-beta-boilerplate.git`

Fra den grønne knap **Clone or Download** fremkommer der et link, som man kopierer til udklipsholderen. Tryk evt. på knappen ved siden af felt.

### Angular 4 - Udvikling af Angular apps

## Appendix I : Opsætning af Angular miljø Uden CLI

I cmd-prompten skriver man **git clone /link**, hvor link er det link, man har i udklipsholderen (pastes ind med **højreklik | sæt ind**.

```
C:\Users\Kursist 1\AJS>git clone https://github.com/angular/quickstart.git
```

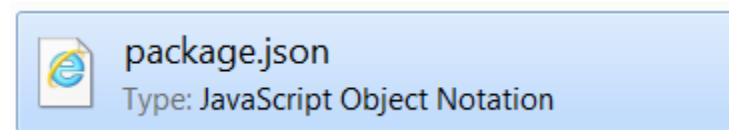
Figur 360

I principippet kan man blot skrive ovenstående direkte i en CMD-prompt uden at gå via github's hjemmeside.

```
Cloning into 'quickstart'...
remote: Counting objects: 1320, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 1320 (delta 0), reused 0 (delta 0), pack-reused 1317
Receiving objects: 100% (1320/1320), 1.10 MiB | 491.00 KiB/s, done.
Resolving deltas: 100% (727/727), done.
Checking connectivity... done.
```

Figur 361

Et bibliotek ved navn Quickstart er oprettet (kan omdøbes). Herfra ligger en lang række filer. Der skal stadig køres en installation. Installationen læser i filen package.json:



Figur 362

Denne fil indeholder alle de dependencies, moduler og programstumper, der skal installeres for at få Angular til at køre.

På nuværende tidspunkt behøver man ikke kigge i filen - og man skal under ingen omstændigheder rette i den.

Kør i stedet installationen med **npm install**.

```
C:\Users\Kursist 1\AJS\quickstart>npm install
```

Figur 363

Det tager nogle minutter. Undervejs skriver den en masse ting. Der kan komme nogle warnings. Kun hvis den skriver fejl (**ERR**), bør man reagere.

```
(node:9552) fs: re-evaluating native module sources is not supported.
ToadDep:symbol-observable ■
```

Figur 364

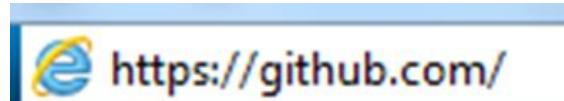
## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materiale må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materiale må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

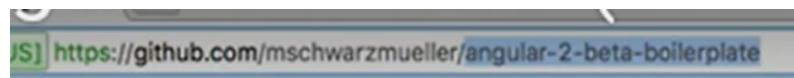


Uden CLI

## Appendix I : Opsætning af Angular miljø

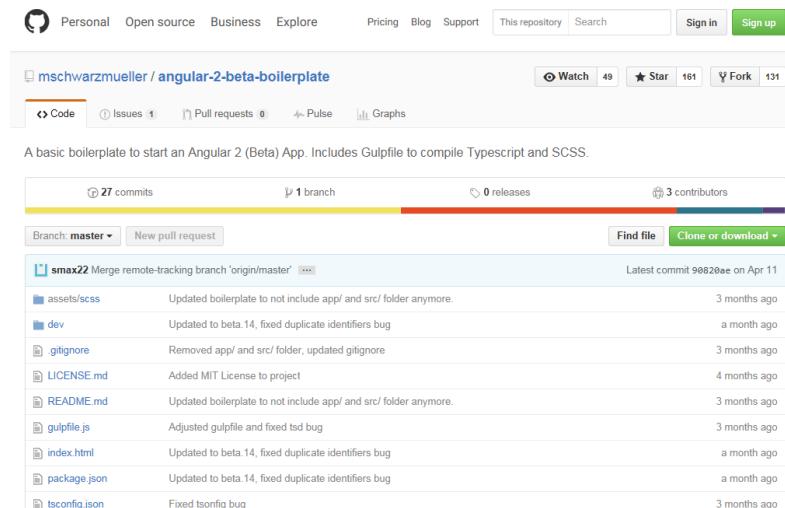


Figur 365



Figur 366

<https://github.com/mschwarzmueller/angular-2-beta-boilerplate>



Commit	Description	Date
assets/scss	Updated boilerplate to not include app/ and src/ folder anymore.	3 months ago
dev	Updated to beta 14, fixed duplicate identifiers bug	a month ago
gitignore	Removed app/ and src/ folder, updated gitignore	3 months ago
LICENSE.md	Added MIT License to project	4 months ago
README.md	Updated boilerplate to not include app/ and src/ folder anymore.	3 months ago
gulpfile.js	Adjusted gulpfile and fixed tsd bug	3 months ago
index.html	Updated to beta 14, fixed duplicate identifiers bug	a month ago
package.json	Updated to beta 14, fixed duplicate identifiers bug	a month ago
tsconfig.json	Fixed tsconfig bug	3 months ago

Figur 367

```
Cloning into 'angular-2-beta-boilerplate'...
remote: Counting objects: 133, done.
Receiving objects: 78% (104/133)
Receiving objects: 100% (133/133), 25.78 KiB | 0 bytes/s, done.
Resolving deltas: 100% (55/55), done.
Checking connectivity... done.
```

Figur 368

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialeet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialeet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Appendix I : Opsætning af Angular miljø Uden CLI

```
C:\Users\Kursist 1\hello>dir
Disken i drev C er Min Harddisk
Diskens serienummer er EEA5-AF3A

Indhold af C:\Users\Kursist 1\hello

24-05-2016 21:47 <DIR> .
24-05-2016 21:47 <DIR> ..
24-05-2016 21:48 <DIR> angular-2-beta-boilerplate
24-05-2016 21:26 116 hello.js
 1 fil(er) 116 byte
 3 mappe(r) 257.554.829.312 byte ledig

C:\Users\Kursist 1\hello>
```

Figur 369

```
>cd angular-2-beta-boilerplate.
```

Figur 370

```
lo\angular-2-beta-boilerplate>npm install.
```

Figur 371

```
C:\Users\Kursist 1\hello\angular-2-beta-boilerplate>npm install
(node:308) fs: re-evaluating native module sources is not supported. If you are using the graceful-fs module, please update to v7.0. Please update 'npm ls graceful-fs' to find it in the tree.
npm WARN deprecated graceful-fs@3.0.8: graceful-fs v3.0.0 and before will fail on node releases >= v7.0. Please update 'npm ls graceful-fs' to find it in the tree.
npm WARN deprecated lodash@1.0.2: lodash@<3.0.0 is no longer maintained. Upgrade to lodash@^4.0.0.
npm WARN deprecated graceful-fs@1.2.3: graceful-fs v3.0.0 and before will fail on node releases >= v7.0. Please update 'npm ls graceful-fs' to find it in the tree.
preinstall
```

Figur 372

```
npm WARN optional Skipping failed optional dependency /chokidar/fsevents:
npm WARN notsup Not compatible with your operating system or architecture: fsevents@1.0.12
npm WARN angular2-boilerplate@1.0.0 No repository field.
```

Figur 373

```
..\node_modules\chokidar\lib\os\win32\fs\watcher.js:100
 const {existsSync, writeSync} = require('fs');
 ^

Error: EPERM: operation not permitted, write 'C:\Users\Kursist 1\hello\angular-2-beta-boilerplate\node_modules\chokidar\lib\os\win32\fs\watcher.js'
 at Object.writeFileSync (fs.js:447:3)
 at Object.writeFileSync (fs.js:447:3)
```

Figur 374



Figur 375

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialelet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialelet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



Uden CLI

## Appendix I : Opsætning af Angular miljø

```
>gulp
```

(gulp skal først installeres)

Figur 376

```
te>npm install --save-dev gulp-install
```

Figur 377

```
>gulp
```

(igen)

```
>npm -g install gulp
```

Figur 378

Gulp skal installeres GLOBALT med -g parameteren.

Dette vindue vil 'hænge' evigt. Det kører serveren. Start et nyt npm vindue -> og gå til angular-2-beta-boilerplate mappen. Kør 'npm start'

Herefter startes en browser med Angular



## Angular 2 Boilerplate

Hello World!

Figur 379

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Appendix J Visual Studio og Node.JS

- ➡ Up and Running med Visual Studio og Node.JS
- ➡ Hello World - hul igennem
- ➡ Brugerflade, Templates, Værd at vide

Her er køreplanen for opsætning af Angular i Microsoft Visual Studio 2.

- ➡ Installér Node.js - hvis det ikke allerede er gjort.
- ➡ Installér Microsoft Visual Studio **2015 Update 3**
- ➡ Opsæt External Web Tools
- ➡ Installér TypeScript til Visual Studio 2015

Nu er Microsoft Visual Studio klar til brug

- ➡ Download en boilerplate - med fordel QuickStart-files
- ➡ Lav et ASP.net projekt i Visual Studio
- ➡ Kopier filer til de rette mapper
- ➡ Lav en Restore af package (installering)
- ➡ Kør dit modul.

For at sikre et fuldt opdateret vejledning peger linket nedenfor på den fuldstændige dokumentation:

<https://angular.io/docs/ts/latest/cookbook/visual-studio-2015.html>

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## Appendix K RxJS operatorer - komplet liste

Aggregate, All, Amb, ambArray, ambWith, and\_, And, Any, apply, as\_blocking, asObservable, AssertEqual, asyncAction, asyncFunc, Average, averageDouble, averageFloat, averageInteger, averageLong,

blocking, blockingFirst, blockingForEach, blockingIterable, blockingLast, blockingLatest, blockingMostRecent, blockingNext, blockingSingle, blockingSubscribe, Buffer, bufferWithCount, bufferWithTime, bufferWithTimeOrCount, byLine,

cache, cacheWithInitialCapacity, case, Cast, Catch, catchError, catchException, collect, collect (RxScala version of Filter), collectInto, CombineLatest, combineLatestDelayError, combineLatestWith, Concat, concat\_all, concatAll, concatArray, concatArrayDelayError, concatArrayEager, concatDelayError, concatEager, concatMap, concatMapDelayError, concatMapEager, concatMapEagerDelayError, concatMapIterable, concatMapObserver, concatMapTo, concatWith, Connect, connect\_forever, cons, Contains, controlled, Count, countLong, Create, cycle,

Debounce, decode, DefaultIfEmpty, Defer, deferFuture, Delay, delaySubscription, delayWithSelector, Dematerialize, Distinct, distinctKey, distinctUntilChanged, distinctUntilKeyChanged, Do, doAction, doAfterTerminate, doOnComplete, doOnCompleted, doOnDispose, doOnEach, doOnError, doOnLifecycle, doOnNext, doOnRequest, doOnSubscribe, doOnTerminate, doOnUnsubscribe, doseq, doWhile, drop, dropRight, dropUntil, dropWhile,

ElementAt, ElementAtOrDefault, Empty, emptyObservable, empty?, encode, ensures, error, every, exclusive, exists, expand,

failWith, Filter, filterNot, Finally, finallyAction, finallyDo, find, findIndex, First, firstElement, FirstOrDefault, firstOrElse, FlatMap, flatMapFirst, flatMapIterable, flatMapIterableWith, flatMapLatest, flatMapObserver, flatMapWith, flatMapWithMaxConcurrent, flat\_map\_with\_index, flatten, flattenDelayError, foldl, foldLeft, for, forall, ForEach, forEachFuture, forEachWhile, forIn, forkJoin, From, fromAction, fromArray, FromAsyncPattern, fromCallable, fromCallback, FromEvent, FromEventPattern, fromFunc0, fromFuture, fromIterable, fromIterator, from\_list, fromNodeCallback, fromPromise, fromPublisher, fromRunnable,

Generate, generateWithAbsoluteTime, generateWithRelativeTime, generator, GetEnumerator, getIterator, GroupBy, GroupByUntil, GroupJoin,

head, headOption, headOrElse,

### Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.

## Appendix K : RxJS operatorer - komplet liste

if, ifThen, IgnoreElements, indexOf, interleave, interpose, Interval,  
 intervalRange, into, isEmpty, items,  
 Join, join (string), jortSort, jortSortUntil, Just,  
 keep, keep-indexed,  
 Last, lastElement, lastOption, LastOrDefault, lastOrElse, Latest, latest  
 (Rx.rb version of Switch), length, let, letBind, lift, limit, LongCount,  
 ManySelect, Map, map (RxClojure version of Zip), MapCat, mapCat  
 (RxClojure version of Zip), map-indexed, mapTo, mapWithIndex,  
 Materialize, Max, MaxBy, Merge, mergeAll, mergeArray,  
 mergeArrayDelayError, merge\_concurrent, mergeDelayError,  
 mergeObservable, mergeWith, Min, MinBy, MostRecent, Multicast,  
 multicastWithSelector,  
 nest, Never, Next, Next (BlockingObservable version), none, nonEmpty,  
 nth,  
 ObserveOn, ObserveOnDispatcher, observeSingleOn, of, of\_array,  
 ofArrayChanges, of\_enumerable, of\_enumerator, ofObjectChanges,  
 OfType, ofWithScheduler, onBackpressureBlock, onBackpressureBuffer,  
 onBackpressureDrop, OnErrorResumeNext, onErrorReturn,  
 onErrorReturnItem, onExceptionResumeNext, onTerminateDetach,  
 orElse,  
 pairs, pairwise, partition, partition-all, pausable, pausableBuffered, pluck,  
 product, Publish, PublishLast, publish\_synchronized, publishValue,  
 raise\_error, Range, Reduce, reduceWith, reductions, RefCount, Repeat,  
 repeat\_ininitely, repeatUntil, repeatWhen, Replay, rescue\_error, rest,  
 Retry, retry\_ininitely, retryUntil, retryWhen, Return, returnElement,  
 returnValue, runAsync,

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.



## liste

## Appendix K : RxJS operatorer - komplet

safeSubscribe, Sample, Scan, scanWith, scope, Select (alternate name of Map), select (alternate name of Filter), selectConcat, selectConcatObserver, SelectMany, selectManyObserver, select\_switch, selectSwitch, selectSwitchFirst, selectWithMaxConcurrent, select\_with\_index, seq, SequenceEqual, sequence\_eql?, SequenceEqualWith, Serialize, share, shareReplay, shareValue, Single, singleElement, SingleOrDefault, singleOption, singleOrElse, size, Skip, SkipLast, skipLastWithTime, SkipUntil, skipUntilWithTime, SkipWhile, skipWhileWithIndex, skip\_with\_time, slice, sliding, slidingBuffer, some, sort, sorted, sort-by, sorted-list-by, split, split-with, Start, startAsync, startFuture, StartWith, startWithArray, stringConcat, stopAndWait, subscribe, subscribeActual, SubscribeOn, SubscribeOnDispatcher, subscribeOnCompleted, subscribeOnError, subscribeOnNext, subscribeWith, Sum, sumDouble, sumFloat, sumInteger, sumLong, Switch, switchCase, switchIfEmpty, switchLatest, switchMap, switchMapDelayError, switchOnNext, switchOnNextDelayError, Synchronize,

Take, take\_with\_time, takeFirst, TakeLast, takeLastBuffer, takeLastBufferWithTime, takeLastWithTime, takeRight (see also: TakeLast), TakeUntil, takeUntilWithTime, TakeWhile, takeWhileWithIndex, tail, tap, tapOnCompleted, tapOnError, tapOnNext, Then, thenDo, Throttle, throttleFirst, throttleLast, throttleWithSelector, throttleWithTimeout, Throw, throwError, throwError, throwException, TimeInterval, Timeout, timeoutWithSelector, Timer, Timestamp, To, to\_a, ToArray, ToAsync, toBlocking, toBuffer, to\_dict, ToDictionary, ToEnumerable, ToEvent, ToEventPattern, ToFlowable, ToFuture, to\_h, toIndexedSeq, tolterable, tolterator, ToList, ToLookup, toMap, toMultiMap, ToObservable, toSet, toSortedList, toStream, ToTask, toTraversable, toVector, tumbling, tumblingBuffer,

unsafeCreate, unsubscribeOn, Using,

When, Where, while, whileDo, Window, windowWithCount, windowWithTime, windowWithTimeOrCount, windowed, withFilter, withLatestFrom,

Zip, zipArray, ziplterable, zipWith, zipWithIndex

## Angular 4 - Udvikling af Angular apps

©ZoomTek.dk. Dette materiale på ikke kopieres eller efterlignes helt eller delvist eller overføres til andet medie eller format. Materialet må kun benyttes af deltagerne på Angular kursus hos ZoomTek.dk. Materialet må ikke uploades, gemmes eller sendes via elektronisk medie. Udprintning af 1 eksemplar udelukkende til eget brug er tilladt.