Dahoam4.0 Consulting

Saban Ünlü

Zwei Worte zu mir

Saban Ünlü

- Software Architekt und Programmierer
- Berater und Dozent seit 2000
- Autor
- Adobe ~~Influencer~~
- LinkedIn IoT & Google Expert
- Gründer von netTrek

netTrek

Agenda

- Gemeinsame Sichtung der Projekte
- Spezifizierung der Anforderungen
- Monorepo
- Lazy Loaded Modulue
- Lazy Loaded Components

Agenda

- Presentation Component
- Reaktive Programmierung mit RxJS
- Wiederverwendbarkeit
- Web component

Projektsetup

Monorepos

- Einfaches orchestrieren von Projekt-Gruppen.
- Leichte Wiederverwendbarkeit.
- Gemeinsame Bibliotheken.
- Kein Abhängigkeitspaketmanagement
- Ein Entwicklungsumgebung.
- Ein GIT

Monorepos - Nachteile

- Werden schnell unübersichtlich.
- Erschweren das Versionsmanagment.
- Kein Abhängigkeitspaketmanagement
- Ein GIT

Monorepos mit ng

- npm i -g @angular/cli
- ng new mono-workspace **--createApplication false**
- ng **generate application** mainApp
- ng **generate library** helper-lib

Monorepos mit nx

- npm install -g @nrwl/cli
- npx create-nx-workspace@latest
  - Workspace: dahomviernull
  - Angular Projekt
  - Projekt dahoam
  - SCSS

Monorepos mit nx

- cd dahomviernull
- nx g @nrwl/angular:application rathaus
- nx g @nrwl/angular:lib my-lib
- nx g @nrwl/angular:lib core **--publishable**
- https://github.com/netTrek/dahoam

Monorepos mit nx

- cd libs/my-lib/src/lib
- nx g m countdown
- cd countdown
- nx g c countdown --skip-tests --flat –export
- In index.ts der lib:
  export * from './lib/countdown/countdown.module';

Monorepos mit nx

- import von CountdownModul in AppModule von
  - Rathaus
  - Dahoam
- In der Vorlage (AppComponent.html) <app-countdown> nutzen

Presenter

Komponenten

Eltern-Kind-Kommunikation

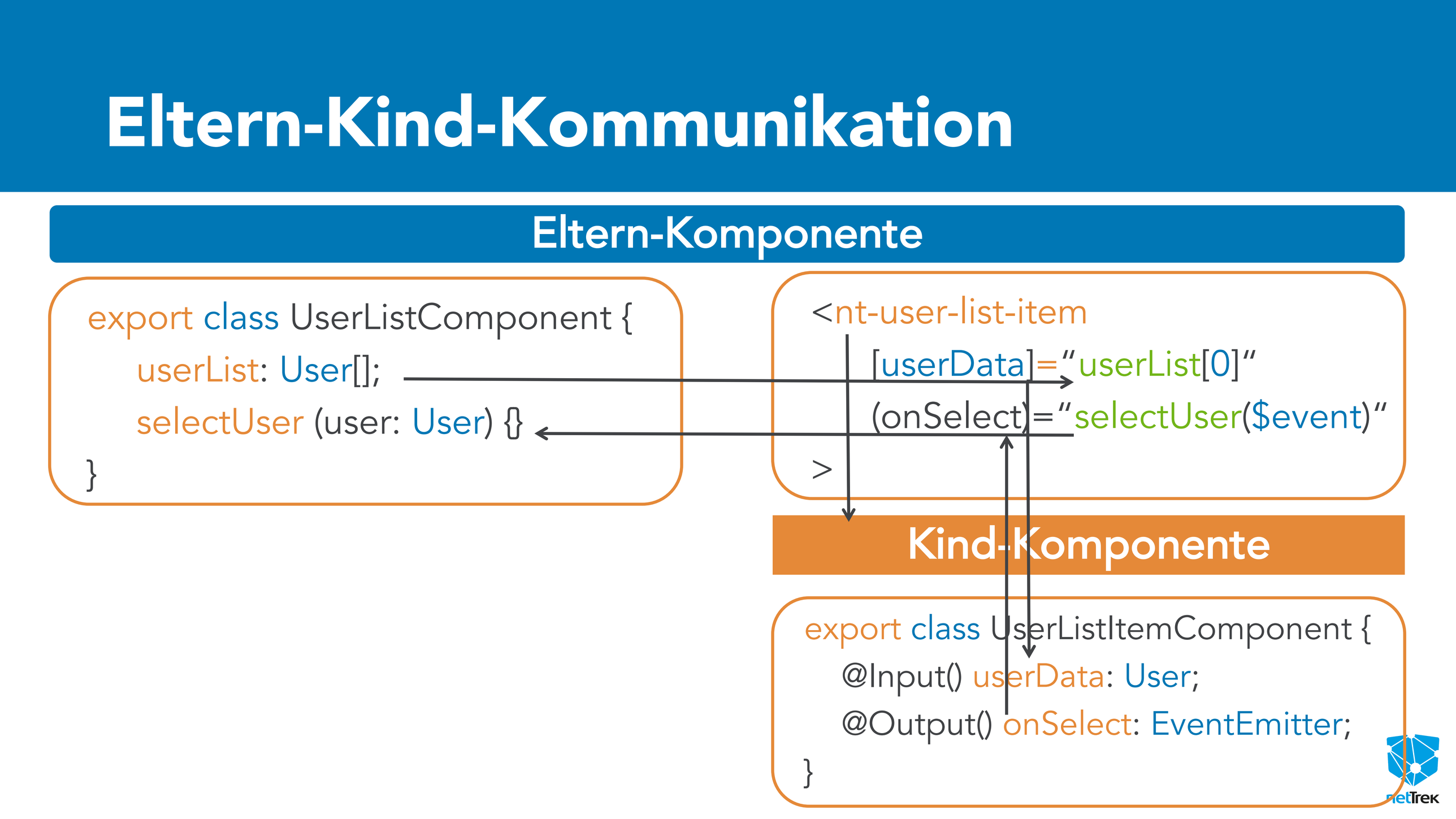
Eltern-Komponente

```
export class UserListComponent {
    userList: User[];
    selectUser (user: User) {}
}
```

```
<nt-user-list-item
    [userData]="userList[0]"
    (onSelect)="selectUser($event)"
>
```

Kind-Komponente

```
export class UserListItemComponent {
    @Input() userData: User;
    @Output() onSelect: EventEmitter;
}
```



Komponentenattribute

- Benutzerdefinierte Attribute lassen sich über den Eigenschaftsdekorator anlegen
  - @Input (OPT_ATTR_NAME) name: Type
- Auch für Setter nutzbar
- ngOnChanges : Hook informiert über neue Werte
  - SimpleChanges

Komponentenereignisse

- Benutzerdefinierte Ereignisse lassen sich über den Eigenschaftsdekorator anlegen
  - @Output (OPT_ATTR_NAME) name: EventEmitter<T>
- EventEmitter sendet Wert via emit
- Elter-Komponenten können sich an das Ereignis hängen
  - $event – Übertragener Ereigniswert

Komponenten-Lebenszyklus

- constructor
- ngOnChanges
- ngOnInit
- ngDoCheck
  - ngAfterContentInit
  - ngAfterContentChecked
  - ngAfterViewInit
  - ngAfterViewChecked
- ngOnDestroy

```
export class UserListComponent
<userList [data]=“userList">
```

UserList Vorlage

```
<userList>
<user></user>
    <user> </user>
</userList>
```

HostBindings- und Listener

- Mittels Eigenschaftsdekorator lassen sich auch Bindungen direkt in der Komponentenklasse definieren
  - @HostBinding (bind) NAME : boolean = true
  - @HostListener (EVT_NAME, [,$event']) HANDLER : Function = (evt)=>{}

Lazy

Lazy loaded Modules

- loadChildren ermöglich im CLI Kontext die einfache Umsetzung

- path : 'dash',
  loadChildren : import('./dash/dash.module')
  . then(m => m.DashModule)

  - Der Pfad zu dem Modul muss importiert und die Modul-Klasse als Promis zurückgegeben werden

- nx g m dash --routing --module app --route=dash

Lazy loaded Modules

- loadChildren ermöglich im CLI Kontext die einfache Umsetzung

- path : 'dash',
  loadChildren : import('./dash/dash.module')
  . then(m => m.DashModule)

  - Der Pfad zu dem Modul muss importiert und die Modul-Klasse als Promis zurückgegeben werden

- ng g m dash --routing --module app --route=dash

Lazy load component

- inject
  - ViewContainerRef, Injector, ComponentFactoryResolver
- Load component factory
  - import('./lazy-avatar/lazy-avatar.component')
    - .then(c => c.LazyAvatarComponent) => Promise

Create loaded component

- viewContainerRef
  - createComponent
    - Factory, Index, Injector
- componentFactoryResolver
  - resolveComponentFactory
    - ComponentClass

Use shared child

- Module
  - In lazy component erstellen ohne export
  - Benötigte Imports
  - Deklarationen
    - Component
    - Shared

rxjs

https://github.com/ReactiveX/rxjs

https://www.learnrxjs.io/

http://rxmarbles.com/

https://rxviz.com

rxjs - Observable

- Lieferant eines observierbaren Datenstroms
- Datenstrom, mit Operatoren manipulierbar und wo Observer (Beobachter) sich registrieren (Subscription)
- Cold (single cast) - Observable wartet auf Subscription
- Hot (multi cast) - Observable arbeitet bereits

rxjs - Observer

- Empfängt Werte, Fehler und Status vom Datenstrom
  - next
  - error
  - complete

rxjs - Subject

- Sowohl Observer als auch Observable (Hot)
  - Damit registrierbarer Datenstrom
  - Und Sender in einem

rxjs - Subscription

- Registrierung an Observable
  - next
  - error
  - complete
- unsubscribe (Deregistrierung)
- siehe: http://rxmarbles.com/

rxjs – Erstellung eins Observables

- new
- of
- range
- fromEvent
- ...

rxjs – Operationen am Datenstrom

- Pipe
  - map
  - filter
  - find
  - scan
  - ...

Web component

Angular Element hinzufügen

- ng add @angular/elements
  - Frameset
  - Angular => Webcomponent

createCustomElement

- AppModule
  - ngDoBootstrap
    - createCustomElement
      - Component, Injector
    - *customElements*
      - *Define*

DANKE

- https://bit.ly/2Jzt12i