Angular Cheat Sheet

Concept & Content by Manfred Steyer

International JavaScript Conference

ANGULAR CLI

```
npm i -g @angular/cli

ng new my-project
cd my-project
ng serve -o

ng generate component MyComponent
ng generate service MyService
ng generate pipe MyPipe

ng build --prod
```

TEMPLATE AND DATA BINDING

```
<input [(ngModel)]="from" name="from">
                    <!-- two way binding -->
<input [(ngModel)]="to" name="to">

<div [hidden]="!to || !from">
                <!-- property binding (one way) -->
  <button (click)="search()">
                              <!-- event binding -->
    Search
  </button>
</div>

<table *ngIf="f.length > 0"> <!-- condition -->
  <tr *ngFor="let f of flights"> <!-- iteration -->
    <td>{{f.id}}</td> <!-- interpolation -->
    <td>{{f.from}}</td>
    <td>{{f.to }}</td>
    <td>{{f.date | date:'dd.MM.yyyy HH:mm'}}</td>
                                          <!-- pipe -->
    <td [class.active]="f === selectedFlight">
                                     <!-- condt. fmt. -->
      <a (click)="select(f)">Select</a>
    </td>
  </tr>
</table>
```

COMPONENT

```
@Component({
  selector: 'flight-search',
  templateUrl: './flight-search.component.html'
})
export class FlightSearchComponent {

  from: string;
  to: string;
  flights: Array<Flight> = [];
  selectedFlight: Flight;

  search(): void { [...] }

  select(f: Flight): void {
    this.selectedFlight = f;
  }
}
```

SERVICE

```
@Injectable({ providedIn: 'root' })
                              // service registration
export class FlightService {

  constructor(private http: HttpClient) { }
                             // constructor injection
  find(from: string, to: string): Observable<Flight[]> {
    let url = 'http://www.angular.at/api/flight';
    let params = { from, to };
    let headers = { accept: 'application/json'};
    return this.http.get<Flight[]>(
                              url, { params, headers });
  }
}
```

CONSUMING SERVICE

```
@Component({ [...] })
export class FlightSearchComponent {
  [...]
  constructor(private flightService: FlightService) { }
                                          // injection
  [...]
  search(): void {
    this
      .flightService
      .find(this.from,  this.to)
      .subscribe( // subscribing to observable
        flights => { this.flights = flights; },
        err => { console.error('err', err); }
      );
  }
}
```

CUSTOM PIPE

```
@Pipe({
  name: 'city',  // name to use in template
  pure: true
})
export class CityPipe implements PipeTransform {

  transform(value: string, fmt: string): string {
    [...]
    return formattedCity;
  }
}
```

INPUTS AND OUTPUTS

```
@Component({
  selector: 'flight-card',
  templateUrl: './flight-card.component.html'
})

export class FlightCardComponent {
  @Input() item: Flight;
  @Input() selected: boolean;
  @Output() selectedChange =
                                  new EventEmitter<boolean>();

  select() {
    this.selected = true;
    this.selectedChange.next(this.selected);
      // trigger event when needed
  }
  deselect() {
    this.selected = false;
    this.selectedChange.next(this.selected);
  }
}
```

CALLING COMPONENTS WITH INPUTS AND OUTPUTS

```
<flight-card
  [item]="f"
  [selected]="basket[f.id]"
  (selectedChange)="basket[f.id] = $event">
</flight-card>

<!-- if event is called like property + Change,
      e. g. selected and selectedChange, and
      event publishes new value for property,
      we can use syntax for two way binding -->

<flight-card
   [item]="f"
   [(selected)]="basket[f.id]">
</flight-card>
```

ROUTING CONFIGURATION

```
export const APP_ROUTES: Routes = [
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: 'home',
    component: HomeComponent},
    {path: 'flight-booking',
     loadChildren: () =>   // lazy loading
     import('./flight-booking/flight-booking.module')
               .then(m => m.FlightBookingModule)},
    {path: '**',
     redirectTo: 'home'}
];
```

IMPORTING THE ROUTERMODULE WITH THE CONFIGURATION

```
@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule,
    // FlightBookingModule,
    // don't import lazy modules!
    RouterModule.forRoot(APP_ROUTES)
                        // reference routing config
                 // use forChild for feature modules!
  ],
  declarations: [
     [...]
// don't forget to register your components
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MARKUP FOR ROUTING

```
<a routerLink="home"></a>
<a routerLink="flight-booking/flight-search">

[...]
<router-outlet></router-outlet>
```

iJS International JavaScript Conference

Angular Cheat Sheet

Concept & Content by Manfred Steyer

ROUTING CONFIG WITH PARAMETER

```
export const FLIGHT_BOOKING_ROUTES: Routes = [
  [...]
  {
    path: 'flight-edit/:id',
               // just for parameters in url segments
          // you don't need to register other url params
    component: FlightEditComponent
  }
];
```

LINK WITH ROUTING PARAMETER

```
<a [routerLink]=
         "['/flight-booking/flight-edit', item.id,
{showDetails: true}]">Edit</a>
```

READING ROUTING PARAMETER

```
@Component({
  selector: 'app-flight-edit',
  templateUrl: './flight-edit.component.html',})
export class FlightEditComponent implements OnInit {

  id: string;
  showDetails: string;

  constructor(private route: ActivatedRoute) { }
    // ActivatedRoute represents the current route
    // including parameter
  ngOnInit() {
    this.route.params.subscribe(p => {
      this.id = p['id'];
                        // from url segment as configured
      this.showDetails = p['showDetails'];
                                        // url parameter
    });
  }
}
```

LIFECYCLE HOOKS

```
@Component({
  selector: 'flight-card',
  templateUrl: './flight-card.component.html'
})
export class FlightCardComponent
  implements OnInit, OnChanges, OnDestroy {

  ngOnInit() {
    [...]
  }

ngOnChanges(changes: SimpleChanges): void {
  if (changes['item']) {
    // item changed
  }
}

ngOnDestroy(): void {
  [...]
}

[...]
}
```

LIFECYCLE HOOKS

```
@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    SharedModule,
    [...]],

  declarations: [
    FlightSearchComponent,
    FlightCardComponent,
    PassengerSearchComponent,
    FlightEditComponent],
  providers: [
    // Providers are global!
    // Traditional way for registering a service
    // Alternative to { providedIn: 'root' }
    { provide: FlightService, useClass: FlightService }
    // Alternative, when provide
    // and useClass points to same type:
    // FlightService
  ],
  exports: [
    // Exports can be used in other modules
    // which are importing this module
    FlightSearchComponent]
})
export class FlightBookingModule {
}
```

Manfred Steyer is a Trainer and Consultant with a focus on Angular. He is a Google Developer Expert (GDE) who writes for O'Reilly, Java Magazin, windows.developer, and Heise. He regularly speaks at conferences.

Web: *www.softwarearchitekt.at*

iJS International **JavaScript** Conference

Angular Cheat Sheet

Concept & Content by Manfred Steyer