UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Angular Komponensek / Sablonok 2.

Dr. Bilicki Vilmos
Szoftverfejlesztés Tanszék

Összefoglaló

- Életciklus, horgonyok
- Komponens interakció
- Angular elemek
- Dinamikus komponensek
- Attribútum direktívák
- Struktúrális dirketívák
- Csövek

Életciklus, horgonyok

- A komponensek életciklusát az Angular menedzseli
  - Létrehozza, megjeleníti, létrehozza és megjeleníti a gyermekeit, megvizsgálja, ha az adatkötések értékei változnak, megsemmisíti mielőtt a DOM-ból is törölné
- Direktívák és komponensek esetén eseményekre iratkozhatunk fel ng+Esemény()
- Az Angular csak azokat hívja meg amelyek meg vannak valósítva

peek-a-boo.component.ts (excerpt)

```
export class PeekABoo implements OnInit {
  constructor(private logger: LoggerService) { }

  // implement OnInit's `ngOnInit` method
  ngOnInit() { this.logIt(`OnInit`); }

  logIt(msg: string) {
    this.logger.log(`#${nextId++} ${msg}`);
  }
}
```

Életciklus szekvencia

Esemény	Jelentése, időzítés
ngOnChanges()	Akkor hívódik meg amikor a kötött adatok bel lettek állítva/frissítve lettek. SimpleChangesobjektumot kap a mostani és az előző értékekkel. Az ngOnInit() előtt hívódik meg.
ngOnInit()	A dirketívák/komponensek adatkötéssel bíró paramétereinek beállítását végzi el egyszer miután a először kerülnek a képernyőre. Az első ngOnChanges() után hívódik meg.
ngDoCheck()	Azokat a változásokt veszi észre és kezeli amelyeket az Angular nem kezel. Minden változás detektáló ciklusban lefut az ngOnChanges() és ngOnInit() után
ngAfterContentInit()	Respond after Angular projects external content into the component's view / the view that a directive is in. Called once after the first ngDoCheck().
ngAfterContentChecked()	Egyszer hívódik meg az első ngDoCheck() után
ngAfterViewInit()	Az ngAfterContentInit() és minden ngDoCheck() után meghívódik miután megvizsgálták a kihelyzett directíva/kompones állapotát.
ngAfterViewChecked()	Az első ngAfterContentChecked() után hívódi meg egyszer miután az angular az adott és gyermek nézekteket megvizsgálta.
ngOnDestroy()	Mielőtt az adott komponenst/direktívát megsemmisítené-

Komponens interakció

- Adat átadás szülőtől gyermeknek input csatolással
- A bemeneti tulajdonságok változásának elkapása setter-rel
- A bemeneti tulajdonságok változásának elkapása ngOnchanges()-szel
- A szülő hallgat a gyermek eseményre
- A szülő együttműködik a gyermekkel lokális változó segítségével
- A szülő meghívja a @ViewChild metódust
- A szülő és a gyerek szolgáltatással kommunikál

Adat átadás szülőtől gyermeknek input csatolással

- @Input dekoráció

component-interaction/src/app/hero-child.component.ts

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

import { Hero } from './hero';

@Component({
  selector: 'app-hero-child',
  template: `
    <h3>{{hero.name}} says:</h3>
    <p>I, {{hero.name}}, am at your service, {{masterName}}
</p>
  `
})
export class HeroChildComponent {
  @Input() hero: Hero;
  @Input('master') masterName: string;
}
```

component-interaction/src/app/hero-parent.component.ts

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

import { HEROES } from './hero';

@Component({
  selector: 'app-hero-parent',
  template: `
    <h2>{{master}} controls {{heroes.length}} heroes</h2>
    <app-hero-child *ngFor="let hero of heroes"
      [hero]="hero"
      [master]="master">
    </app-hero-child>
  `
})
export class HeroParentComponent {
  heroes = HEROES;
  master = 'Master';
}
```

A bemeneti tulajdonságok változásának elkapása setter-rel

component-interaction/src/app/name-child.component.ts

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

@Component({
  selector: 'app-name-child',
  template: '<h3>"{{name}}"</h3>'
})
export class NameChildComponent {
  private _name = '';

  @Input()
  set name(name: string) {
    this._name = (name && name.trim()) || '<no name set>';
  }

  get name(): string { return this._name; }
}
```

component-interaction/src/app/name-parent.component.ts

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

@Component({
  selector: 'app-name-parent',
  template: `
  <h2>Master controls {{names.length}} names</h2>
  <app-name-child *ngFor="let name of names" [name]="name">
</app-name-child>
  `
})
export class NameParentComponent {
  // Displays 'Mr. IQ', '<no name set>', 'Bombasto'
  names = ['Mr. IQ', '   ', '  Bombasto  '];
}
```

A bemeneti tulajdonságok változásának elkapása ngOnchanges()-szel

component-interaction/src/app/version-child.component.ts

```ts
import { Component, Input, OnChanges, SimpleChange } from '@angular/core';

@Component({
  selector: 'app-version-child',
  template: `
    <h3>Version {{major}}.{{minor}}</h3>
    <h4>Change log:</h4>
    <ul>
      <li *ngFor="let change of changeLog">{{change}}</li>
    </ul>
  `
})
export class VersionChildComponent implements OnChanges {
  @Input() major: number;
  @Input() minor: number;
  changeLog: string[] = [];

  ngOnChanges(changes: {[propKey: string]: SimpleChange}) {
    let log: string[] = [];
    for (let propName in changes) {
      let changedProp = changes[propName];
      let to = JSON.stringify(changedProp.currentValue);
      if (changedProp.isFirstChange()) {
        log.push(`Initial value of ${propName} set to ${to}`);
      } else {
        let from = JSON.stringify(changedProp.previousValue);
        log.push(`${propName} changed from ${from} to ${to}`);
      }
    }
    this.changeLog.push(log.join(', '));
  }
}
```

component-interaction/src/app/version-parent.component.ts

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

@Component({
  selector: 'app-version-parent',
  template: `
    <h2>Source code version</h2>
    <button (click)="newMinor()">New minor version</button>
    <button (click)="newMajor()">New major version</button>
    <app-version-child [major]="major" [minor]="minor"></app-version-child>
  `
})
export class VersionParentComponent {
  major = 1;
  minor = 23;

  newMinor() {
    this.minor++;
  }

  newMajor() {
    this.major++;
    this.minor = 0;
  }
}
```

A szülő hallgat a gyermek eseményre

component-interaction/src/app/voter.component.ts

```ts
import { Component, EventEmitter, Input, Output } from '@angular/core';

@Component({
  selector: 'app-voter',
  template: `
    <h4>{{name}}</h4>
    <button (click)="vote(true)"  [disabled]="didVote">Agree</button>
    <button (click)="vote(false)" [disabled]="didVote">Disagree</button>
  `
})
export class VoterComponent {
  @Input()  name: string;
  @Output() voted = new EventEmitter<boolean>();
  didVote = false;

  vote(agreed: boolean) {
    this.voted.emit(agreed);
    this.didVote = true;
  }
}
```

component-interaction/src/app/votetaker.component.ts

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

@Component({
  selector: 'app-vote-taker',
  template: `
    <h2>Should mankind colonize the Universe?</h2>
    <h3>Agree: {{agreed}}, Disagree: {{disagreed}}</h3>
    <app-voter *ngFor="let voter of voters"
      [name]="voter"
      (voted)="onVoted($event)">
    </app-voter>
  `
})
export class VoteTakerComponent {
  agreed = 0;
  disagreed = 0;
  voters = ['Mr. IQ', 'Ms. Universe', 'Bombasto'];

  onVoted(agreed: boolean) {
    agreed ? this.agreed++ : this.disagreed++;
  }
}
```

A szülő együttműködik a gyermekkel lokális változó segítségével

- A szülő komponens nem tud hozzáférni a gyermek tulajdonságaihoz, metódusaihoz
- Helyi változó segítségével hozzáférhetővé tehető

component-interaction/src/app/countdown-parent.component.ts

```
1.  import { Component }                from '@angular/core';
2.  import { CountdownTimerComponent }  from './countdown-timer.component';
3.
4.  @Component({
5.    selector: 'app-countdown-parent-lv',
6.    template: `
7.    <h3>Countdown to Liftoff (via local variable)</h3>
8.    <button (click)="timer.start()">Start</button>
9.    <button (click)="timer.stop()">Stop</button>
0.    <div class="seconds">{{timer.seconds}}</div>
1.    <app-countdown-timer #timer></app-countdown-timer>
2.    `,
3.    styleUrls: ['../assets/demo.css']
4.  })
5.  export class CountdownLocalVarParentComponent { }
```

Példa

component-interaction/src/app/countdown-timer.component.ts

```
1.  import { Component, OnDestroy, OnInit } from '@angular/core';
2.
3.  @Component({
4.    selector: 'app-countdown-timer',
5.    template: '<p>{{message}}</p>'
6.  })
7.  export class CountdownTimerComponent implements OnInit, OnDestroy {
8.
9.    intervalId = 0;
0.    message = '';
1.    seconds = 11;
2.
3.    clearTimer() { clearInterval(this.intervalId); }
4.
5.    ngOnInit()    { this.start(); }
6.    ngOnDestroy() { this.clearTimer(); }
7.
8.    start() { this.countDown(); }
9.    stop()  {
0.      this.clearTimer();
1.      this.message = `Holding at T-${this.seconds} seconds`;
2.    }
3.
4.    private countDown() {
5.      this.clearTimer();
6.      this.intervalId = window.setInterval(() => {
7.        this.seconds -= 1;
8.        if (this.seconds === 0) {
9.          this.message = 'Blast off!';
0.        } else {
1.          if (this.seconds < 0) { this.seconds = 10; } // reset
2.          this.message = `T-${this.seconds} seconds and counting`;
3.        }
4.      }, 1000);
5.    }
6.  }
```

A szülő meghívja a @ViewChild() metódust

component-interaction/src/app/countdown-parent.component.ts

```
1.  import { AfterViewInit, ViewChild } from '@angular/core';
2.  import { Component }                from '@angular/core';
3.  import { CountdownTimerComponent }  from './countdown-timer.component';
4.
5.  @Component({
6.    selector: 'app-countdown-parent-vc',
7.    template: `
8.    <h3>Countdown to Liftoff (via ViewChild)</h3>
9.    <button (click)="start()">Start</button>
0.    <button (click)="stop()">Stop</button>
1.    <div class="seconds">{{ seconds() }}</div>
2.    <app-countdown-timer></app-countdown-timer>
3.    `,
4.    styleUrls: ['../assets/demo.css']
5.  })
6.  export class CountdownViewChildParentComponent implements AfterViewInit {
7.
8.    @ViewChild(CountdownTimerComponent)
9.    private timerComponent: CountdownTimerComponent;
0.
1.    seconds() { return 0; }
2.
3.    ngAfterViewInit() {
4.      // Redefine `seconds()` to get from the `CountdownTimerComponent.seconds` ...
5.      // but wait a tick first to avoid one-time devMode
6.      // unidirectional-data-flow-violation error
7.      setTimeout(() => this.seconds = () => this.timerComponent.seconds, 0);
8.    }
9.
0.    start() { this.timerComponent.start(); }
1.    stop() { this.timerComponent.stop(); }
2.  }
```

Szülő gyermek kommunikáció szolgáltatás segítségével

- A szülő és gyermek komponens egy szolgáltatást használ megosztva. Ezzel a családon belül lehet kommunikálni.
- A szolgáltatás szkópja a szülő és a gyermek komponens. Ezen kívül lévő komponenesek nem férnek hozzá.

component-interaction/src/app/mission.service.ts

```
1.  import { Injectable } from '@angular/core';
2.  import { Subject }    from 'rxjs';
3.
4.  @Injectable()
5.  export class MissionService {
6.
7.    // Observable string sources
8.    private missionAnnouncedSource = new Subject<string>();
9.    private missionConfirmedSource = new Subject<string>();
0.
1.    // Observable string streams
2.    missionAnnounced$ = this.missionAnnouncedSource.asObservable();
3.    missionConfirmed$ = this.missionConfirmedSource.asObservable();
4.
5.    // Service message commands
6.    announceMission(mission: string) {
7.      this.missionAnnouncedSource.next(mission);
8.    }
9.
0.    confirmMission(astronaut: string) {
1.      this.missionConfirmedSource.next(astronaut);
2.    }
3.  }
```

component-interaction/src/app/missioncontrol.component.ts

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

import { MissionService }     from './mission.service';

@Component({
  selector: 'app-mission-control',
  template: `
    <h2>Mission Control</h2>
    <button (click)="announce()">Announce mission</button>
    <app-astronaut *ngFor="let astronaut of astronauts"
      [astronaut]="astronaut">
    </app-astronaut>
    <h3>History</h3>
    <ul>
      <li *ngFor="let event of history">{{event}}</li>
    </ul>
  `,
  providers: [MissionService]
})
export class MissionControlComponent {
  astronauts = ['Lovell', 'Swigert', 'Haise'];
  history: string[] = [];
  missions = ['Fly to the moon!',
              'Fly to mars!',
              'Fly to Vegas!'];
  nextMission = 0;

  constructor(private missionService: MissionService) {
    missionService.missionConfirmed$.subscribe(
      astronaut => {
        this.history.push(`${astronaut} confirmed the mission`);
      });
  }

  announce() {
    let mission = this.missions[this.nextMission++];
    this.missionService.announceMission(mission);
    this.history.push(`Mission "${mission}" announced`);
    if (this.nextMission >= this.missions.length) { this.nextMission = 0; }
  }
}
```

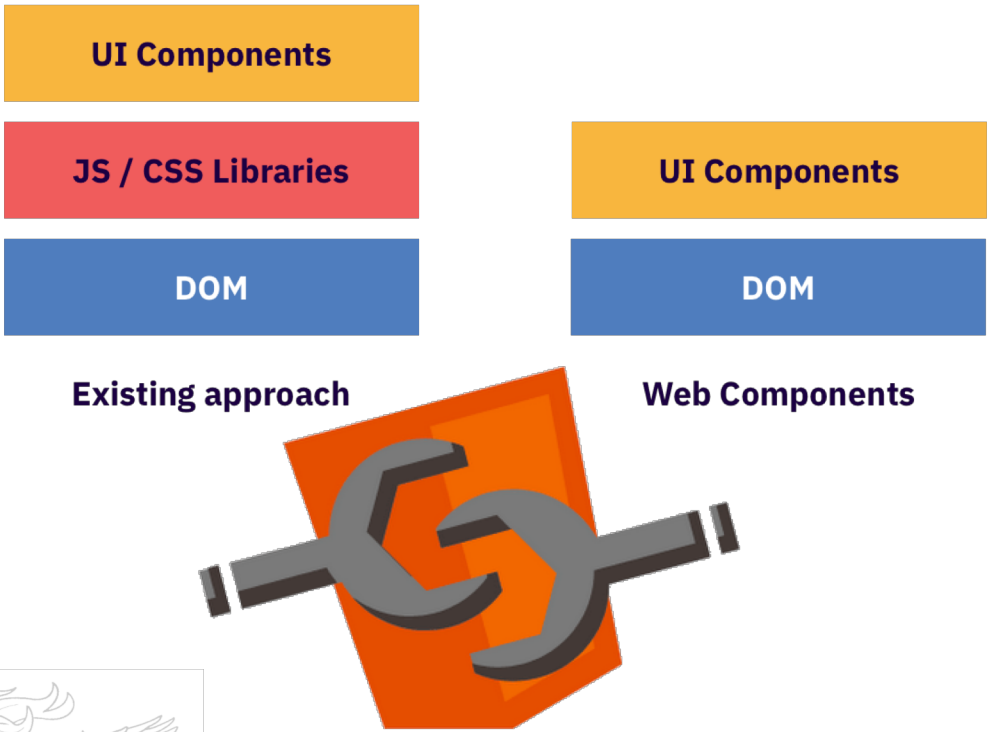
component-interaction/src/app/astronaut.component.ts

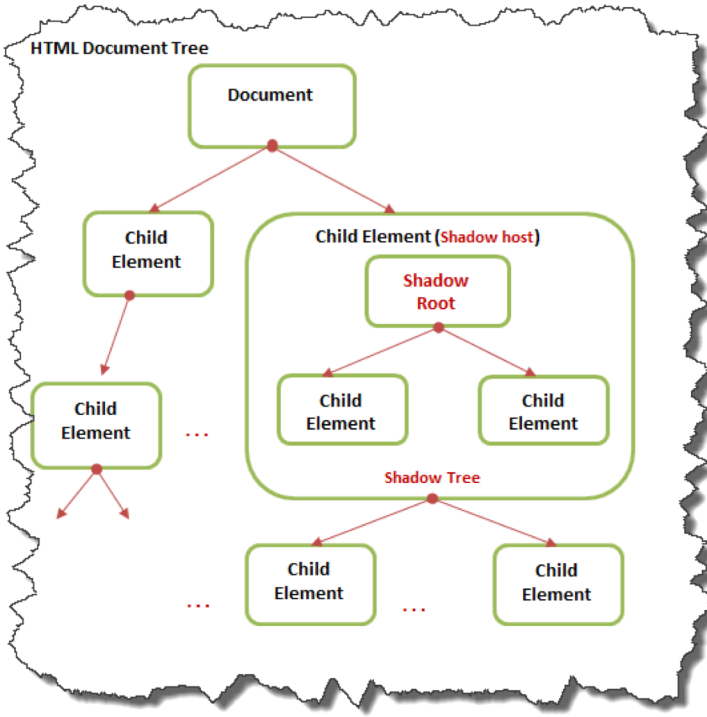
```ts
import { Component, Input, OnDestroy } from '@angular/core';

import { MissionService } from './mission.service';
import { Subscription }   from 'rxjs';

@Component({
  selector: 'app-astronaut',
  template: `
    <p>
      {{astronaut}}: <strong>{{mission}}</strong>
      <button
        (click)="confirm()"
        [disabled]="!announced || confirmed">
        Confirm
      </button>
    </p>
  `
})
export class AstronautComponent implements OnDestroy {
  @Input() astronaut: string;
  mission = '<no mission announced>';
  confirmed = false;
  announced = false;
  subscription: Subscription;

  constructor(private missionService: MissionService) {
    this.subscription = missionService.missionAnnounced$.subscribe(
      mission => {
        this.mission = mission;
        this.announced = true;
        this.confirmed = false;
    });
  }

  confirm() {
    this.confirmed = true;
    this.missionService.confirmMission(this.astronaut);
  }

  ngOnDestroy() {
    // prevent memory leak when component destroyed
    this.subscription.unsubscribe();
  }
}
```

Web komponensek

- HTML5 + AJAX + DOM
  - Saját elemek adott kereterendszerben
  - Egy egyszerű elemhez is kell az egész kererendszer
  - A keretrendszerek nem kompatibiliesk egymással
    - Anglular, React, Vue, ...
  - A böngészők különböző szinten támogatják
- Web komponens
  - A böngésző adja a saját környezetet

Web komponensek





WEB COMPONENTS

TEMPLATES

```
<template id="">
</template>
```

SHADOW DOM

```
div
  #document-fragment
    span
```

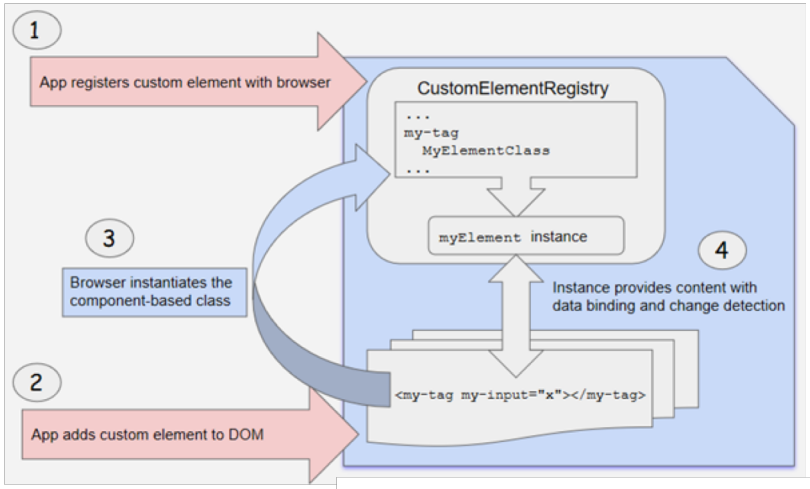
HTML IMPORTS

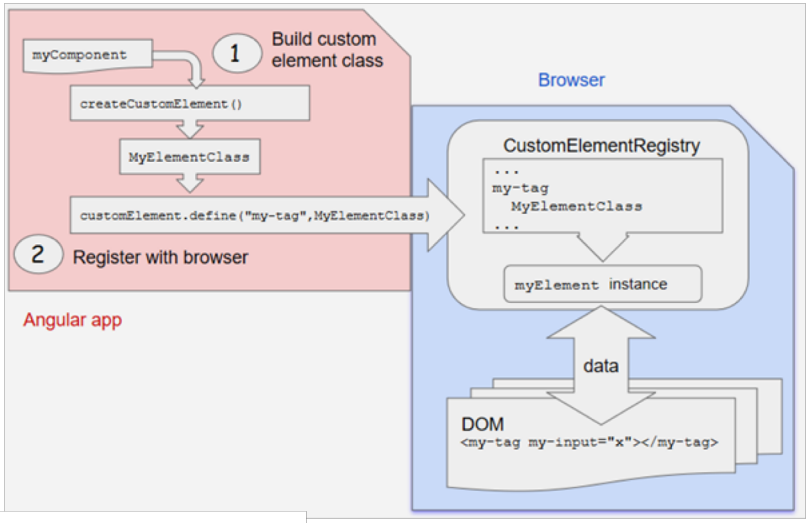
```
<link rel="import"
   href="part.html">
```

CUSTOM ELEMENTS

```
<my-elem>
</my-elem>
```

Angular elemek






```
import { Component, Injector } from '@angular/core';
import { createCustomElement } from '@angular/elements';
import { PopupService } from './popup.service';
import { PopupComponent } from './popup.component';

@Component({
  selector: 'app-root',
  template: `
    <input #input value="Message">
    <button (click)="popup.showAsComponent(input.value)">Show as component</button>
    <button (click)="popup.showAsElement(input.value)">Show as element</button>
  `,
})
export class AppComponent {
  constructor(injector: Injector, public popup: PopupService) {
    // Convert `PopupComponent` to a custom element.
    const PopupElement = createCustomElement(PopupComponent, {injector});
    // Register the custom element with the browser.
    customElements.define('popup-element', PopupElement);
  }
}
```

Dinamikus komponensek

- Futásidőben történő komponens betöltés

src/app/ad.directive.ts

```
import { Directive, ViewContainerRef } from '@angular/core';

@Directive({
  selector: '[ad-host]',
})
export class AdDirective {
  constructor(public viewContainerRef: ViewContainerRef) { }
}
```

src/app/ad-banner.component.ts (template)

```
template: `
            <div class="ad-banner-example">
              <h3>Advertisements</h3>
              <ng-template ad-host></ng-template>
            </div>
          `
```

src/app/ad-banner.component.ts (excerpt)

```
export class AdBannerComponent implements OnInit, OnDestroy {
  @Input() ads: AdItem[];
  currentAdIndex = -1;
  @ViewChild(AdDirective) adHost: AdDirective;
  interval: any;

  constructor(private componentFactoryResolver: ComponentFactoryResolver) { }

  ngOnInit() {
    this.loadComponent();
    this.getAds();
  }

  ngOnDestroy() {
    clearInterval(this.interval);
  }

  loadComponent() {
    this.currentAdIndex = (this.currentAdIndex + 1) % this.ads.length;
    let adItem = this.ads[this.currentAdIndex];

    let componentFactory = this.componentFactoryResolver.resolveComponentFactory(adItem.component);

    let viewContainerRef = this.adHost.viewContainerRef;
    viewContainerRef.clear();

    let componentRef = viewContainerRef.createComponent(componentFactory);
    (<AdComponent>componentRef.instance).data = adItem.data;
  }

  getAds() {
    this.interval = setInterval(() => {
      this.loadComponent();
    }, 3000);
  }
}
```

Attribútum direktívák

src/app/app.component.html (applied)

```
<p appHighlight>Highlight me!</p>
```

src/app/highlight.directive.ts

```
import { Directive, ElementRef } from '@angular/core';

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
    constructor(el: ElementRef) {
       el.nativeElement.style.backgroundColor = 'yellow';
    }
}
```

src/app/highlight.directive.ts

```
1.  import { Directive, ElementRef, HostListener } from '@angular/core';
2.
3.  @Directive({
4.    selector: '[appHighlight]'
5.  })
6.  export class HighlightDirective {
7.    constructor(private el: ElementRef) { }
8.
9.    @HostListener('mouseenter') onMouseEnter() {
10.     this.highlight('yellow');
11.   }
12.
13.   @HostListener('mouseleave') onMouseLeave() {
14.     this.highlight(null);
15.   }
16.
17.   private highlight(color: string) {
18.     this.el.nativeElement.style.backgroundColor = color;
19.   }
20. }
```

Struktúrális dirketívák

- A HTML szerkezetéért felelősek
- Adott host elemre és leszármazottjaira vonatkozik
- * - mikroszintaxis
- NgIf vs ngIf

src/app/app.component.html (ngif)

```
<div *ngIf="hero" class="name">{{hero.name}}</div>
```

src/app/app.component.html (built-in)

```
<div *ngIf="hero" class="name">{{hero.name}}</div>

<ul>
  <li *ngFor="let hero of heroes">{{hero.name}}</li>
</ul>

<div [ngSwitch]="hero?.emotion">
  <app-happy-hero    *ngSwitchCase="'happy'"    [hero]="hero"></app-happy-hero>
  <app-sad-hero      *ngSwitchCase="'sad'"      [hero]="hero"></app-sad-hero>
  <app-confused-hero *ngSwitchCase="'confused'" [hero]="hero"></app-confused-hero>
  <app-unknown-hero  *ngSwitchDefault           [hero]="hero"></app-unknown-hero>
</div>
```

NgIf

- Adott DOM részeket fizikailag töröl
  - Törli a DOM-ból
  - Lekapcsolja a DOM eseményekről
  - A komponenst lekapcsolja az Angular esenény detekcióról
  - A csomópont memóriája felszabadítható

src/app/app.component.html (ngif-true)

```
<p *ngIf="true">
  Expression is true and ngIf is true.
  This paragraph is in the DOM.
</p>
<p *ngIf="false">
  Expression is false and ngIf is false.
  This paragraph is not in the DOM.
</p>
```

```
<p _ngcontent-c0>
  Expression is true and ngIf is true.
  This paragraph is in the DOM.
</p>
<!--bindings={
  "ng-reflect-ng-if": "false"
}-->
```

Miért nem rejti el?

src/app/app.component.html (display-none)

```
<p [style.display]="'block'">
  Expression sets display to "block".
  This paragraph is visible.
</p>
<p [style.display]="'none'">
  Expression sets display to "none".
  This paragraph is hidden but still in the DOM.
</p>
```

```
<p _ngcontent-fwv-0 style="display: block;">
  Expression sets display to "block"" .
  This paragraph is visible.
</p>
<p _ngcontent-fwv-0 style="display: none;">
  "
    Expression sets display to "none" .
    This paragraph is hidden but still in the DOM.
  "
</p>
```

* prefix

- * szintaxis egyszerűsítés, szépítés

src/app/app.component.html (asterisk)

```
<div *ngIf="hero" class="name">{{hero.name}}</div>
```

src/app/app.component.html (ngif-template)

```
<ng-template [ngIf]="hero">
  <div class="name">{{hero.name}}</div>
</ng-template>
```

```
<!--bindings={
  "ng-reflect-ng-if": "[object Object]"
}-->
<div _ngcontent-c0>Mr. Nice</div>
```

ngTemplate

- Az <ng-template> egy Angular elem a HTML képernyőre helyezésére.
- Soha nem jelenik meg közvetlenül.
- Valójában az adott nézet megjelenítése előtt az <ng-template> és tartalma egy megjegyzésre cserélődik
- Struktúrális direktíva teszi működővé az ngTemplate elemet

src/app/app.component.html (template-tag)

```
<p>Hip!</p>
<ng-template>
  <p>Hip!</p>
</ng-template>
<p>Hooray!</p>
```

```
<p _ngcontent-c0>Hip!</p>
<!---->
<p _ngcontent-c0>Hooray!</p>
```

Hip!

Hooray!

Mikroszintaxis

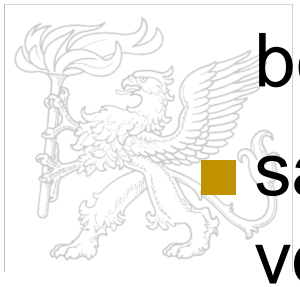
- Segítségével lehet egy diektívát kompakt, egyszerű módon konfigurálni
- ng-template attribútumokká alakítja az értelmező

src/app/app.component.html (inside-ngfor)

```
<div *ngFor="let hero of heroes; let i=index; let odd=odd; trackBy: trackById" [class.odd]="odd">
  ({{i}}) {{hero.name}}
</div>

<ng-template ngFor let-hero [ngForOf]="heroes" let-i="index" let-odd="odd" [ngForTrackBy]="trackById">
  <div [class.odd]="odd">({{i}}) {{hero.name}}</div>
</ng-template>
```

Sablon bemeneti változó

- Egy sablon példányon belül érhető el
- Nem ugyanaz mint a sablon referencia változó
  - #var vs. let var
  - a referencia változó a a csatolt elemre hivatkozik és bárhonnan elérhető a sablonon belül
  - sablon bemeneti változó egy példányra vonatkozik



<ng-container>

- Ha olyan befoglaló elemet szeretnénk ami stílust nem bontja meg

src/app/app.component.html (ngif

```
<p>
  I turned the corner
  <span *ngIf="hero">
    and saw {{hero.name}}. I waved
  </span>
  and continued on my way.
</p>
```

src/app/app.component.css (p-span)

```
p span { color: red; font-size: 70%; }
```

I turned the corner and saw Mr. Nice. I waved and continued on my way.

src/app/app.component.html (ngif-ngcontainer)

```
<p>
  I turned the corner
  <ng-container *ngIf="hero">
    and saw {{hero.name}}. I waved
  </ng-container>
  and continued on my way.
</p>
```

Csövek

src/app/hero-birthday1.component.ts

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

@Component({
  selector: 'app-hero-birthday',
  template: `<p>The hero's birthday is {{ birthday | date }}</p>`
})
export class HeroBirthdayComponent {
  birthday = new Date(1988, 3, 15); // April 15, 1988
}
```

src/app/hero-birthday2.component.ts (template)

```
template: `
  <p>The hero's birthday is {{ birthday | date:format }}</p>
  <button (click)="toggleFormat()">Toggle Format</button>
`
```

src/app/app.component.html

```
The chained hero's birthday is
{{  birthday | date:'fullDate' | uppercase}}
```

Saját cső


```
import { Pipe, PipeTransform } from '@angular/core';
/*
 * Raise the value exponentially
 * Takes an exponent argument that defaults to 1.
 * Usage:
 *   value | exponentialStrength:exponent
 * Example:
 *   {{ 2 | exponentialStrength:10 }}
 *   formats to: 1024
*/
@Pipe({name: 'exponentialStrength'})
export class ExponentialStrengthPipe implements PipeTransform {
  transform(value: number, exponent: string): number {
    let exp = parseFloat(exponent);
    return Math.pow(value, isNaN(exp) ? 1 : exp);
  }
}
```

Csövek és aváltozásdetektálás

- Csövek használata hatékony
- Típusai
  - Pure
  - Impure

```
@Pipe({
  name: 'myCustomPipe',
  pure: false/true        <----- here (default is `true`)
})
export class MyCustomPipe {}
```

Összefoglaló

- Felhasználói bevitel
- Életciklus, horgonyok
- Komponens interakció
- Angular elemek
- Dinamikus komponensek
- Attribútum direktívák
- Struktúrális dirketívák
- Csövek