UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Angular Komponensek,Sablonok

Dr. Bilicki Vilmos
Szoftverfejlesztés Tanszék

Összefoglaló

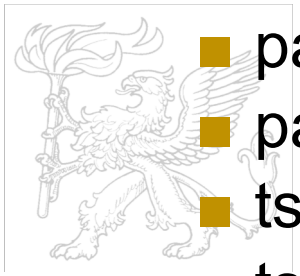
- Projekt Fájl Struktúra
- Adat megjelenítés
- Sablon szintaxis
- Felhasználói bevitel

Projekt Fájl Strukúra

- Workspace Configuration
- npm függőségek
- TypeScript konfiguráció
- Ahead-of-Time Compilation
- Building & Serving
- Testing
- Deployment
- Browser Support
- Dev Tool integration

Workspace

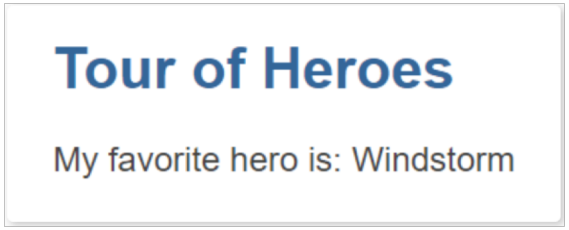
- Workspace vs. Projekt
  - Egy munkakönyvtrában több angular projekt lehet (cli ng new <project_name>)
- Munkakönyvtár szintű konfig
  - .editorconfig
  - .gitignore
  - angular.json (CLI konfig, eszökzökről, ...)
  - node_modules
  - package.json
  - package-lock.json
  - tsconfig.json
  - tslint.json
  - README.md



Project

- app/
- assets/
- environments/
- browserslist
- favicon.ico
- index.html
- main.ts
- polyfills.ts
- styles.sass
- test.ts
- tsconfig.app.json
- tsconfig.spec.json
- tslint.json

Adat megjelenítés



src/index.html (body)

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

src/app/app.component.ts

```
1.  import { Component } from '@angular/core';
2.
3.  @Component({
4.    selector: 'app-root',
5.    template: `
6.      <h1>{{title}}</h1>
7.      <h2>My favorite hero is: {{myHero}}</h2>
8.      `
9.  })
10. export class AppComponent {
11.   title = 'Tour of Heroes';
12.   myHero = 'Windstorm';
13. }
```

interpoláció

src/app/app.component.ts (template)

```
template: `
  <h1>{{title}}</h1>
  <h2>My favorite hero is: {{myHero}}</h2>
  `
```

TypeScript fordított aposztróf

Példa


```ts
import { NgModule } from '@angular/core';
import { BrowserModule }  from '@angular/platform-browser';

import { AppComponent } from './app.component';

@NgModule({
  imports: [
    BrowserModule
  ],
  declarations: [
    AppComponent
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

app.component.ts

```ts
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule);
```

app.module.ts

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

import { Hero } from './hero';

@Component({
  selector: 'app-root',
  template: `
  <h1>{{title}}</h1>
  <h2>My favorite hero is: {{myHero.name}}</h2>
  <p>Heroes:</p>
  <ul>
    <li *ngFor="let hero of heroes">
      {{ hero.name }}
      </li>
  </ul>
  <p *ngIf="heroes.length > 3">There are many heroes!</p>
`
})
export class AppComponent {
  title = 'Tour of Heroes';
  heroes = [
    new Hero(1, 'Windstorm'),
    new Hero(13, 'Bombasto'),
    new Hero(15, 'Magneta'),
    new Hero(20, 'Tornado')
  ];
  myHero = this.heroes[0];
}
```

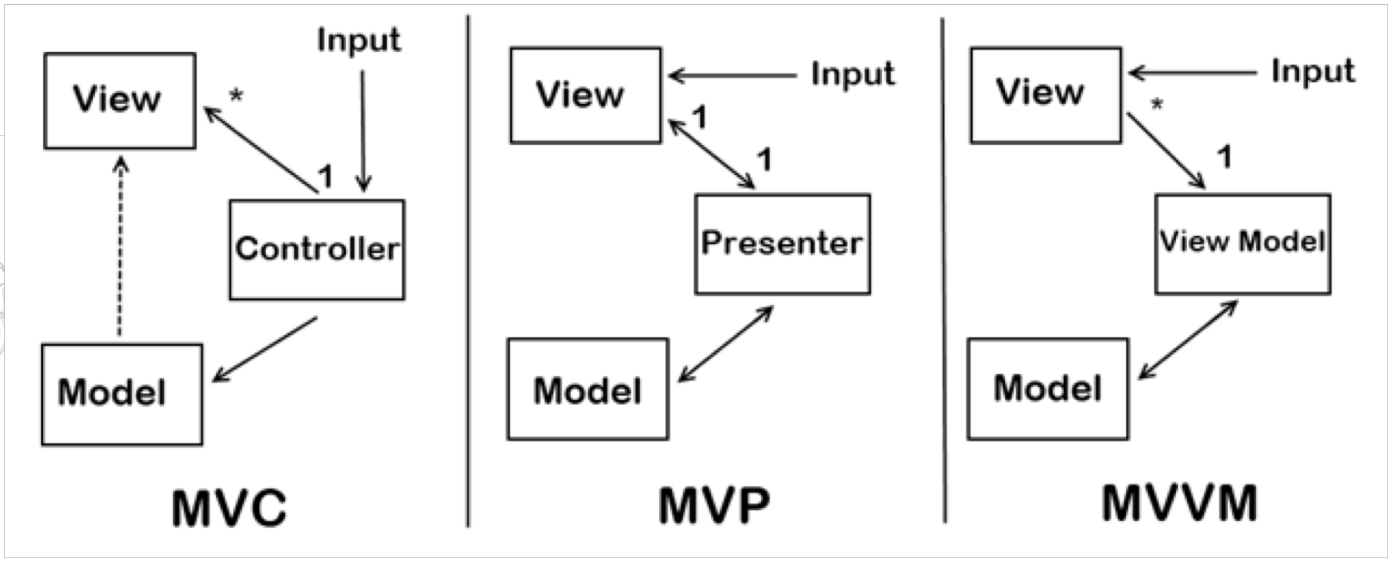
main.ts

```ts
export class Hero {
  constructor(
    public id: number,
    public name: string) { }
}
```

hero.ts

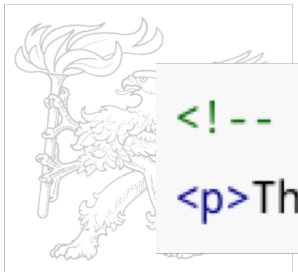
Sablon szintaxis

- A felhasználó élményt (mit lát, mit tehet) a komponens és a felhasználói felületet meghatározó sablon határozza meg
- MVC, MVVM
- Angular:
  - Controller:ViewModel
  - Sablon: View



HTML, Közbeszúrás

- Bármely HTML elem használható kivéve
  - <script>
- Közbeszúrás (Interpolation)
  - Interpolation {{...}}
    - Sablon kifejezés
  - Az Angular lecseréli a változót az értékre (minden eseményre)
    - Kiértékeli
    - Átalakítja karakterlánccá
    - Átadja a megfelelő elem, direktíva tulajdonságnak vagy HTML elemnek



```
<!-- "The sum of 1 + 1 is not 4" -->
<p>The sum of 1 + 1 is not {{1 + 1 + getVal()}}.</p>
```

Sablon kifejezés

- Egy értéket ad és a dupla kapcsos zárójelen belül van
- Tulajdonság kötés esetén a jobb oldalon jelenik meg [property]="expression”
- A legtöbb JS elem használható
- Kifejezés kontextus
  - Sablon változók + komponens példány + a dirketíva kontextusa
  - Nem fér hozzá a globális kontextushoz
  - Nem fér hozzá a window, document objektumokohoz
  - Nem tud logolni

Sablon kifejezés: ajánlások

- Angular „Egyirányú adatfolyam”
  - Ne változtasson mást csak azt amihez hozzá van kötve (egy érték kiolvasása ne változtasson más helyen lévő adatot)
- Cél: Idempotens kifejezés
- Gyors lefutás: minden változásdetektációs ciklusnál kiértékeli (sok aszinkron esemény: HTTP, egér, billentyű, időzítő, ...)
- Egyszerűség: tuladonság, metódus hívás de ne sok boolean kifejezés

Sablon deklaráció

- A deklaráció (statement) a csatolt elem (komponens, direktíva) eseményére válaszol (event)="statement”
- A kontextusa
  - a komponens példány
  - a sablon saját környezet
- Hasonló szabályok vonatkoznak rá mint a kifejezésre

src/app/app.component.html

```
<button (click)="onSave($event)">Save</button>
<button *ngFor="let hero of heroes" (click)="deleteHero(hero)">{{hero.name}}</button>
<form #heroForm (ngSubmit)="onSubmit(heroForm)"> ... </form>
```

Csatolás szintaxis

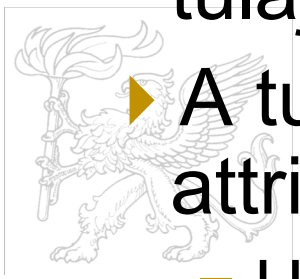
Adatirány	Szintaxis	Típus

Egyirányú az adatforrástól a nézet célig	`{{expression}}`
	`[target]="expression"`
	`bind-target="expression"`
		Közbeszúrás
		Tulajdonság
		Attribútum
		Osztály
		Stílus

Egyirányú a nézet céltól az adatforrásig	`(target)="statement"` `on-target="statement"`	Esemény
Kétirányú	`[(target)]="expression"` `bindon-target="expression"`	Kétirányú

HTML attribútum vs. DOM tulajdonság

- HTML vs. DOM
  - 1:1 összerendelés pl.: id
  - HTML attribútum és nincs DOM párja pl: colspan
  - DOM tulajdonság aminek nincs HTML megfelelője pl: textContent
  - Sok HTML attribútum tűnik csatoltnak de, nem
- Az attribútumok incializálják a DOM tulajdonságokat
- A tulajdonságok változhatnak az attribútumok nem
  - HTML attribútum → kezdeti érték
  - DOM tulajdonság → aktuális érték



Csatolás célok

Típus	Cél	Példa

Tulajdonság	Elem tulajdonság
	Komponens tulajdonság
	Direktíva tulajdonság
		src/app/app.component.html
		`<img [src]="heroImageUrl">`
		`<app-hero-detail [hero]="currentHero"></app-hero-detail>`
		`<div [ngClass]="{'special': isSpecial}"></div>`

Esemény	Elem esemény
	Komponens esemény
	Direktíva esemény
		src/app/app.component.html
		`<button (click)="onSave()">Save</button>`
		`<app-hero-detail (deleteRequest)="deleteHero()"></app-hero-detail>`
		`<div (myClick)="clicked=$event" clickable>click me</div>`

Kétirányú	Esemény és tulajdonság	src/app/app.component.html `<input [(ngModel)]="name">`
Attribútum	Attribútum (kivétel)	src/app/app.component.html `<button [attr.aria-label]="help">help</button>`
Osztály	Osztály tulajdonság	src/app/app.component.html `<div [class.special]="isSpecial">Special</div>`
Stílus	Stílus tulajdonság	src/app/app.component.html `<button [style.color]="isSpecial ? 'red' : 'green'">`

Tulajdonság csatolás

- Beállítja a nézet elem egy tulajdonságát
- Egyirányú adatfolyam
  - Komponens adat tulajdonságból az elem tulajdonságába
  - Nem használhatjuk arra, hogy kiolvassuk
  - Nem hívhatjuk meg a cél eleme metódusait sem
- @Input-tal jelöljük a másik oldalon azokat a tulajdonságokat amiket így szeretnénk kiajánlani

src/app/app.component.html

```
<img bind-src="heroImageUrl">
```

src/app/app.component.html

```
<img [src]="heroImageUrl">
```

src/app/app.component.html

```
<!-- ERROR: HeroDetailComponent.hero expects a
     Hero object, not the string "currentHero" -->
<app-hero-detail hero="currentHero"></app-hero-detail>
```

Közbeszúrás vagy Csatlás

- Karaterlánc esteében mindegy
- Nem karakterláncnál csatolást kell használni

src/app/app.component.html

```
<p><img src="{{heroImageUrl}}"> is the <i>interpolated</i> image.</p>
<p><img [src]="heroImageUrl"> is the <i>property bound</i> image.</p>

<p><span>"{{title}}" is the <i>interpolated</i> title.</span></p>
<p>"<span [innerHTML]="title"></span>" is the <i>property bound</i> title.</p>
```

Kód injektálás

- Az angular először „fertőtleníti” a karakterláncot és csak ezután szúrja be

src/app/app.component.ts

```
evilTitle = 'Template <script>alert("evil never sleeps")</script>Syntax';
```

src/app/app.component.html

```
<!--
  Angular generates warnings for these two lines as it sanitizes them
  WARNING: sanitizing HTML stripped some content (see http://g.co/ng/security#xss).
 -->
<p><span>"{{evilTitle}}" is the <i>interpolated</i> evil title.</span></p>
<p>"<span [innerHTML]="evilTitle"></span>" is the <i>property bound</i> evil title.</p>
```

"Template <script>alert("evil never sleeps")</script>Syntax" is the *interpolated* evil title.

"Template Syntax" is the *property bound* evil title.

Attribútum, osztály, stílus csatolás

- Attribútumok közvetlen beállítása
- Ezt csak akkor szabad/lehet használni ha nincs ilyen tulajdonság

```
<tr><td colspan="{{1 + 1}}">Three-Four</td></tr>
```

```
Template parse errors:
Can't bind to 'colspan' since it isn't a known native property
```

src/app/app.component.html

```
<table border=1>
  <!--  expression calculates colspan=2 -->
  <tr><td [attr.colspan]="1 + 1">One-Two</td></tr>

  <!-- ERROR: There is no `colspan` property to set!
    <tr><td colspan="{{1 + 1}}">Three-Four</td></tr>
  -->

  <tr><td>Five</td><td>Si
</table>
```

src/app/app.component.html

```
<!-- create and set an aria attribute for assistive technology -->
<button [attr.aria-label]="actionName">{{actionName}} with Aria</button>
```

Osztály csatolás

- CSS osztály neveket adhatunk hozzá, vagy vehetünk el
- class csatolás
- Javasolt az NgClass direktíva (több osztály egyszeri manipulásálásra)

src/app/app.component.html

```
<!-- reset/override all class names with a binding  -->
<div class="bad curly special"
     [class]="badCurly">Bad curly</div>
```

src/app/app.component.ts

```
currentClasses: {};
setCurrentClasses() {
  // CSS classes: added/removed per current state of component properties
  this.currentClasses =  {
    'saveable': this.canSave,
    'modified': !this.isUnchanged,
    'special':  this.isSpecial
  };
}
```

src/app/app.component.html

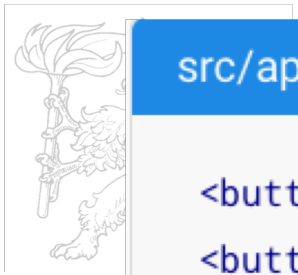
```
<div [ngClass]="currentClasses">This div is initially saveable, unchanged, and special</div>
```

Stílus csatolás

- [style.style-property]
- unit extension: .em, .%

src/app/app.component.html

```
<button [style.color]="isSpecial ? 'red': 'green'">Red</button>
<button [style.background-color]="canSave ? 'cyan': 'grey'" >Save</button>
```



src/app/app.component.html

```
<button [style.font-size.em]="isSpecial ? 3 : 1" >Big</button>
<button [style.font-size.%]="!isSpecial ? 150 : 50" >Small</button>
```

Esemény csatolás

- Elemtől → Kompnenshez
- Cél esemény (event)
- $event – az eseményhez tartozó adatokat adja át

src/app/app.component.html

```
<button (click)="onSave()">Save</button>
```

src/app/app.component.html

```
<input [value]="currentHero.name"
       (input)="currentHero.name=$event.target.value" >
```

Saját események

- Az egyes direktívák általában adnak ki eseményeket
- Jellemzően lefelé populálódik a komponens fán

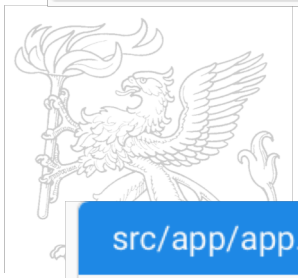
src/app/hero-detail.component.ts (template)

```
template: `
<div>
  <img src="{{heroImageUrl}}">
  <span [style.text-decoration]="lineThrough">
    {{prefix}} {{hero?.name}}
  </span>
  <button (click)="delete()">Delete</button>
</div>`
```

src/app/hero-detail.component.ts (deleteRequest)

```
// This component makes a request but it can't actually delete a hero.
deleteRequest = new EventEmitter<Hero>();

delete() {
  this.deleteRequest.emit(this.hero);
}
```



src/app/app.component.html (event-binding-to-component)

```
<app-hero-detail (deleteRequest)="deleteHero($event)" [hero]="currentHero"></app-hero-detail>
```

Kétirányú csatolás

- Szeretnénk kiírni és megváltoztatni is
- [(...)] banán a dobozban ☺
- HTML űrlap elemekhez (pl.: input) NgModel mert nem követik a x, xChange nevezéktant

src/app/app.component.html (two-way-1)

```
<app-sizer [(size)]="fontSizePx"></app-sizer>
<div [style.font-size.px]="fontSizePx">Resizable Text</div>
```

src/app/app.component.html (two-way-2)

```
<app-sizer [size]="fontSizePx" (sizeChange)="fontSizePx=$event"></app-sizer>
```

src/app/app.component.html

```
<input [value]="currentHero.name"
       (input)="currentHero.name=$event.target.value" >
```

src/app/app.component.html

```
<input [(ngModel)]="currentHero.name">
```

Beépített attribútum direktívák

- NgClass – CSS osztályok hozzáadása, eltávolítása
- NgStyle – HTML stílusok hozzáadása, eltávolítása
- NgModel – kétirányú adatkötés HTML űrlap elemhez
  - Csak azokra működik amelyek támogatva vannak (ControlValueAccessor)
    - CheckboxControlValueAccessor
    - DefaultValueAccessor
    - RadioControlValueAccessor
    - SelectControlValueAccessor
    - SelectMultipleControlValueAccessor
  - Saját komponens esetén be kell tartani a szinakxist

NgModel

src/app/app.module.ts (FormsModule import)

```
import { NgModule } from '@angular/core';
import { BrowserModule }  from '@angular/platform-browser';
import { FormsModule } from '@angular/forms'; // <--- JavaScript import from Angular

/* Other imports */

@NgModule({
  imports: [
    BrowserModule,
    FormsModule  // <--- import into the NgModule
  ],
  /* Other module metadata */
})
export class AppModule { }
```

src/app/app.component.html

```
<input
  [ngModel]="currentHero.name"
  (ngModelChange)="currentHero.name=$event">
```

src/app/app.component.html

```
<input [(ngModel)]="currentHero.name">
```

NgModel

src/app/sizer.component.ts

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

@Component({
  selector: 'app-sizer',
  template: `
  <div>
    <button (click)="dec()" title="smaller">-</button>
    <button (click)="inc()" title="bigger">+</button>
    <label [style.font-size.px]="size">FontSize: {{size}}px</label>
  </div>`
})
export class SizerComponent {
  @Input()  size: number | string;
  @Output() sizeChange = new EventEmitter<number>();

  dec() { this.resize(-1); }
  inc() { this.resize(+1); }

  resize(delta: number) {
    this.size = Math.min(40, Math.max(8, +this.size + delta));
    this.sizeChange.emit(this.size);
  }
}
```

src/app/app.component.html (two-way-1)

```html
<app-sizer [(size)]="fontSizePx"></app-sizer>
<div [style.font-size.px]="fontSizePx">Resizable Text</div>
```

Beépített struktúrális direktívák

- A HTML kinézetet befolyásolják → DOM struktúra változás segítségével
- Példa
  - NgIf
  - NgSwitch
  - NgForOf

src/app/app.component.html

```
<app-hero-detail *ngIf="isActive"></app-hero-detail>
```

src/app/app.component.html

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

src/app/app.component.html

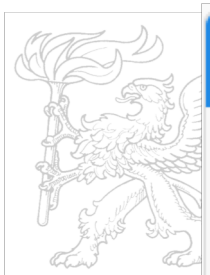
```
<div *ngFor="let hero of heroes; let i=index">{{i + 1}} - {{hero.name}}</div>
```

A * jelentősége

- Egy nyelvi elem
- Jelzi az angualrnak, hogy burkolja be egy sablonnal

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>
```

*ngFor trackBy

- Nagy listáknál egy elem változása is a teljes DOM újratöltését eredményezi
- A megváltozott elemet jelezhetjük az angularnak (milyen változást figyeljen)

src/app/app.component.ts

```
trackByHeroes(index: number, hero: Hero): number { return hero.id; }
```

src/app/app.component.html

```
<div *ngFor="let hero of heroes; trackBy: trackByHeroes">
  ({{hero.id}}) {{hero.name}}
</div>
```

Sablon referencia változók

- #var → egy sablonhoz tartozó DOM elemre mutató referencia
- a sablonon belül bárhonnan hivatkozhatunk rá
- más mint belül a let x

src/app/app.component.html

```
<input #phone placeholder="phone number">

<!-- lots of other elements -->

<!-- phone refers to the input element; pass its `value` to an event handler -->
<button (click)="callPhone(phone.value)">Call</button>
```

Bemenő és kimenő tulajdonságok

- @Input → beállítható tulajdonság
- @Output → megfigyelhető tulajdonság (EventEmitter)
- A saját publikus tulajdonságok annotálás nélkül is elérhetőek
- A komponenseket/direktívákat csak ezeken a csatornákon keresztül köthetjük össze
  - Ha más komponessel kötjük össze akkor az angular fordító hibát jelez

src/app/app.component.html

```
<app-hero-detail [hero]="currentHero" (deleteRequest)="deleteHero($event)">
</app-hero-detail>
```

```
Uncaught Error: Template parse errors:
Can't bind to 'hero' since it isn't a known property of 'app-hero-detail'
```

src/app/hero-detail.component.ts

```
@Input()  hero: Hero;
@Output() deleteRequest = new EventEmitter<Hero>();
```

src/app/hero-detail.component.ts

```
@Component({
  inputs: ['hero'],
  outputs: ['deleteRequest'],
})
```

Biztonságos hivatkozás

- biztonságos navigáció operátor (?.)
- null/undefined utak esetén nyújt védelmet
- laza betöltésnél fontos (még nemtöltődöt le az adat)

src/app/app.component.html

```
The current hero's name is {{currentHero?.name}}
```

src/app/app.component.html

```
The title is {{title}}
```

```
TypeError: Cannot read property 'name' of null in [null].
```

Felhasználói bevitel

- A felhasználói akciók mint kattintás, gombnyomás, szövegbevitel, … DOM eseményeket generálnak
- Angular esemény kötés (DOM eseményhez)

src/app/click-me.component.ts

```
<button (click)="onClickMe()">Click me!</button>
```

Kontextusa

src/app/click-me.component.ts

```
@Component({
  selector: 'app-click-me',
  template: `
    <button (click)="onClickMe()">Click me!</button>
    {{clickMessage}}`
})
export class ClickMeComponent {
  clickMessage = '';

  onClickMe() {
    this.clickMessage = 'You are my hero!';
  }
}
```

Felhasználó bevitel: $event

- A DOM események adatot is tartalmaznak
- DOM target a kontextus: event.target.value

src/app/keyup.components.ts (temp

```
template: `
  <input (keyup)="onKey($event)">
  <p>{{values}}</p>
`
```

src/app/keyup.components.ts (class v.1)

```
export class KeyUpComponent_v1 {
  values = '';

  onKey(event: any) { // without type info
    this.values += event.target.value + ' | ';
  }
}
```

src/app/keyup.components.ts (class v.1 - typed)

```
export class KeyUpComponent_v1 {
  values = '';

  onKey(event: KeyboardEvent) { // with type info
    this.values += (<HTMLInputElement>event.target).value + ' | ';
  }
}
```

Helyes használat

- $event közvetlen használata HTML függővé teszi a kódunkat
- Sablon referencia változó: #változó

src/app/loop-back.component.ts

```
@Component({
  selector: 'app-loop-back',
  template: `
    <input #box (keyup)="0">
    <p>{{box.value}}</p>
  `
})
export class LoopbackComponent { }
```

src/app/keyup.components.ts (v3)

```
@Component({
  selector: 'app-key-up3',
  template: `
    <input #box (keyup.enter)="onEnter(box.value)">
    <p>{{value}}</p>
  `
})
export class KeyUpComponent_v3 {
  value = '';
  onEnter(value: string) { this.value = value; }
}
```

src/app/keyup.components.ts (v4)

```
@Component({
  selector: 'app-key-up4',
  template: `
    <input #box
      (keyup.enter)="update(box.value)"
      (blur)="update(box.value)">

    <p>{{value}}</p>
  `
})
export class KeyUpComponent_v4 {
  value = '';
  update(value: string) { this.value = value; }
}
```

Összefoglaló

- Projekt Fájl Struktúra
- Adat megjelenítés
- Sablon szintaxis
- Felhasználói bevitel