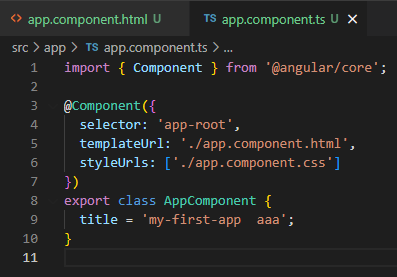
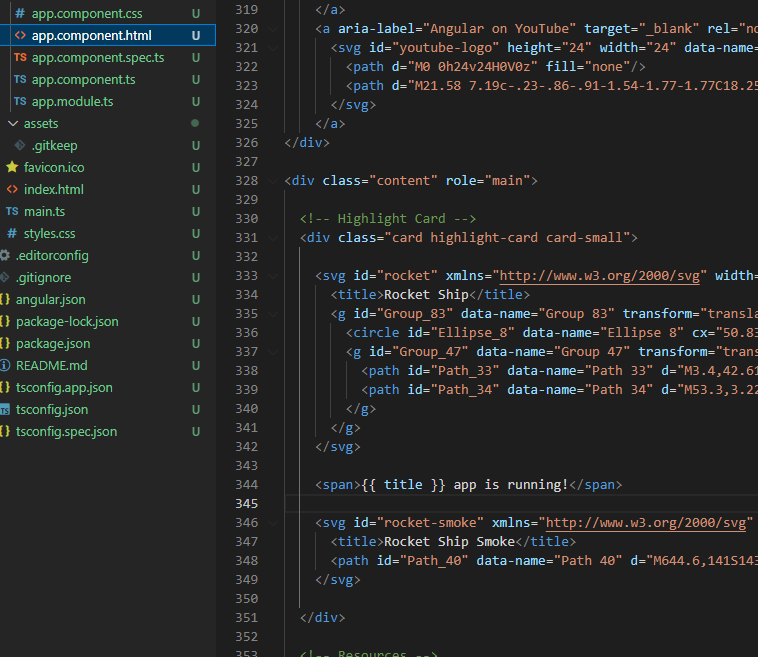


ng serve

<http://localhost:4200/>

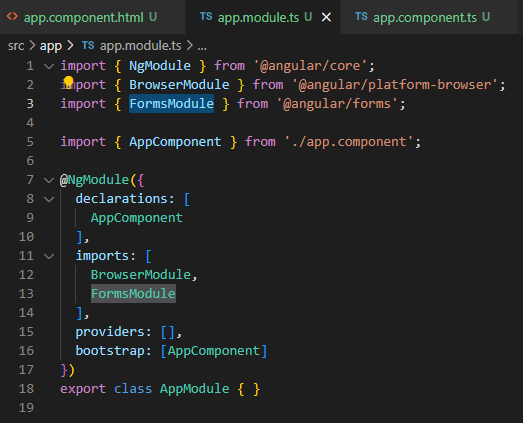


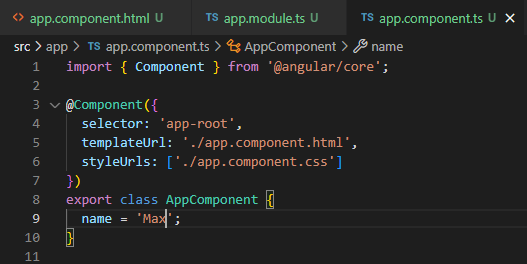


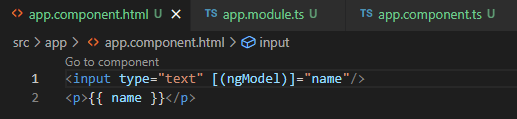
Itt van a title és a componens ahol lehet változatatni magát a title-t.

Statikus és dinamikus dolgokat ötvözi.

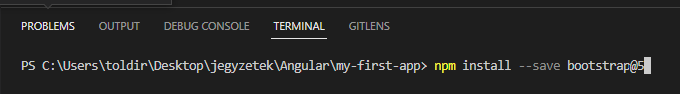
**Data binding, input html teggel:**

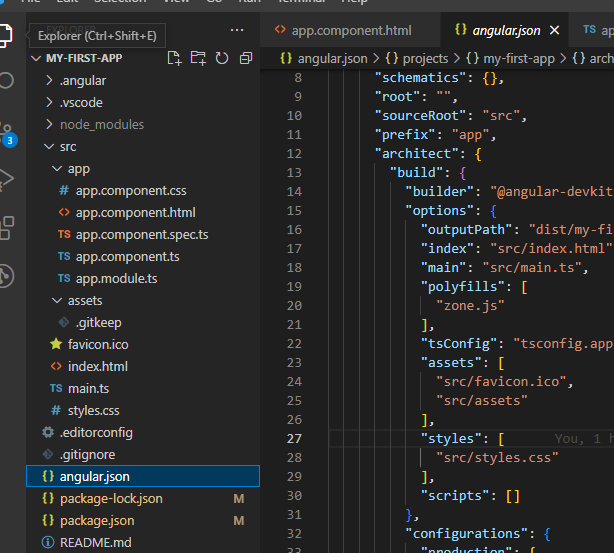


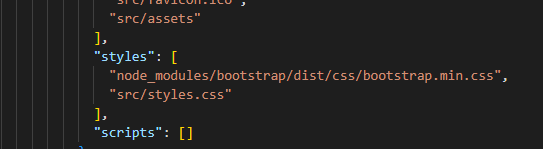




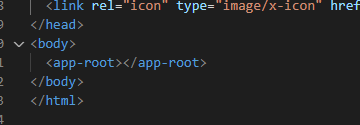
**Bootstrap telepítése a projektbe:**







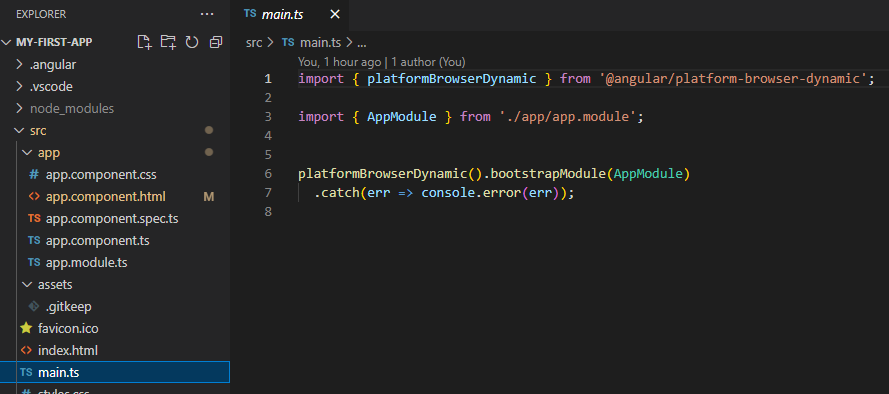
index.html file az ami a szervernek ki van szolgáltatva



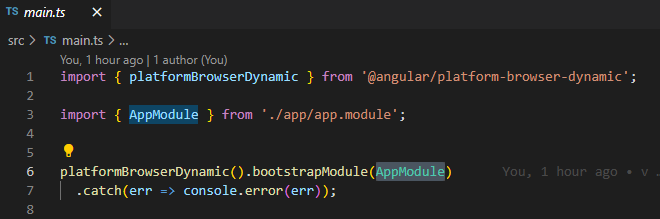
Ez a rész ami az app.component.html-t benyeli.



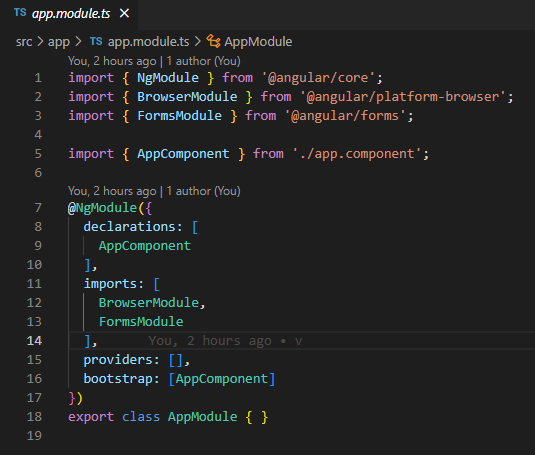
Ez fogja az approot contentet felcserélni az aktuális app.component.html-lel.



Ez az első script kód ami feldolgozásra kerül.



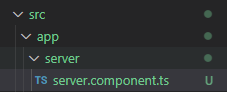
Ez köti össze az app.module.ts-el a main.ts-t.



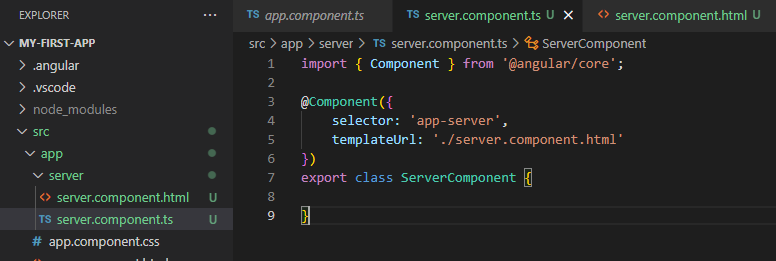
AppComponent-tel záródik itt a kör.

**Components:**

Minden komponensnek saját mappát érdemes csinálni, app folderben.

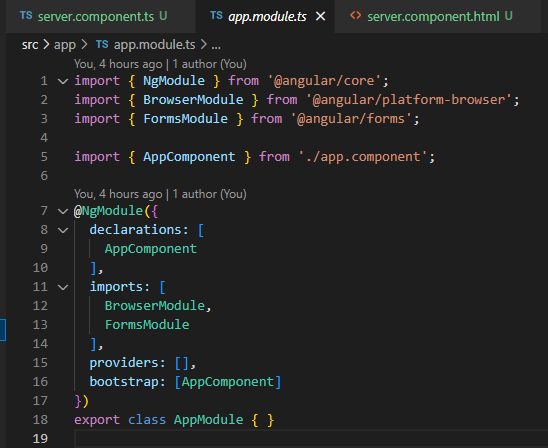


Érdemes ezt a naming convention-t használni.

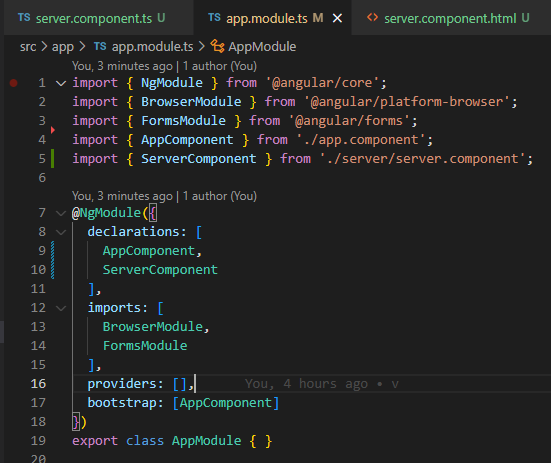


Import kell hogy a component attributumot felismerje és ott meg kell adni bizonyos tulajdonságokat, a selectort amivel hivatkozunk rá és a template-t ahol használjuk.

Hogy használni tudjuk még az app.module-on változtatnunk kell.



Eredeti állapot. Ez gyüjti össze a modulokat, bundle of functionalities of your app.



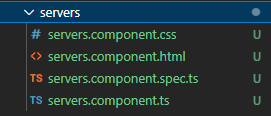
Ennyi itt.



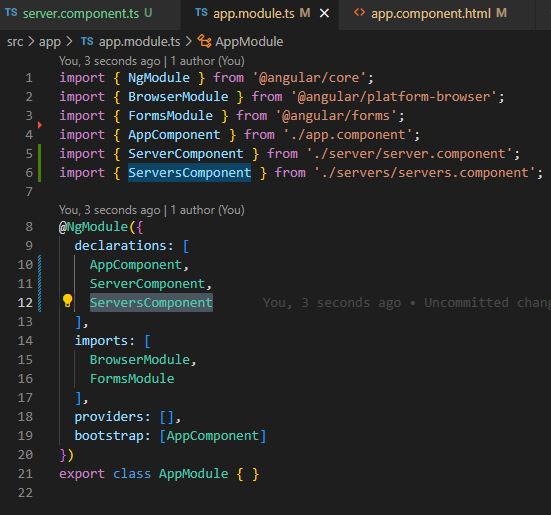
így lehet a komponenst meghívni az oldalra, megjeleníteni

**Creating Components with the CLI & Nesting Components:**

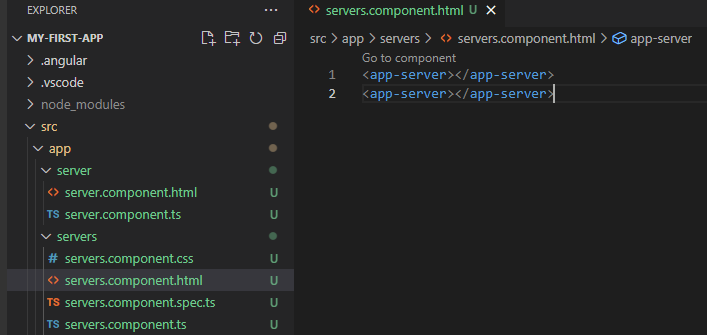
Terminalba: ng generate component név v. ng g c név

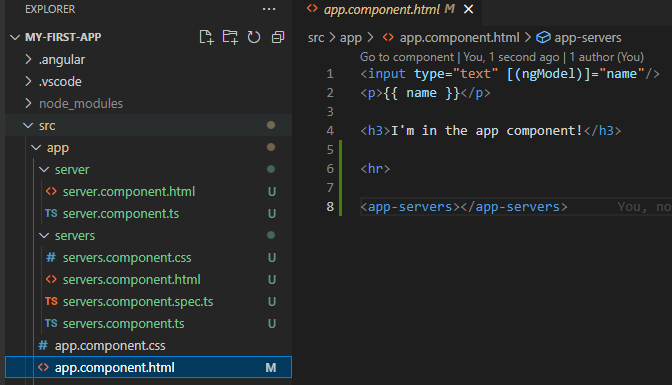


Meg is csinálta a componenseket és a mappát is.

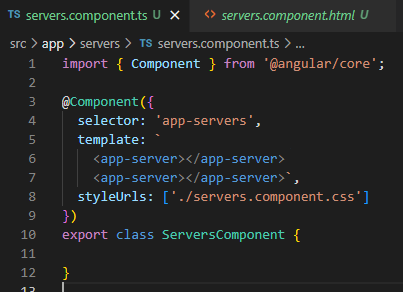


Ide is belerakja a CLI.

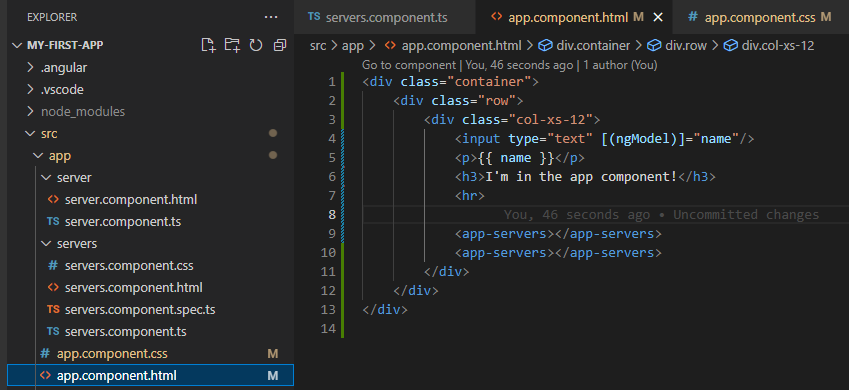




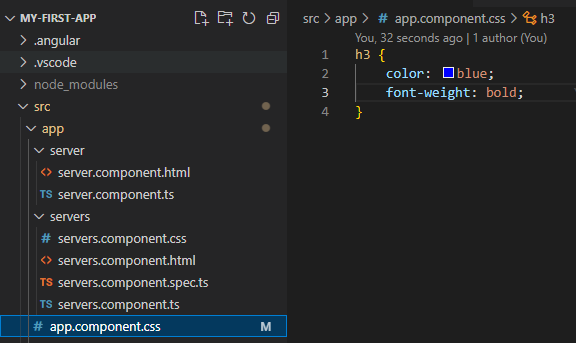
így lesz egyik componens a másikba meghívva és az a app.component-ben.



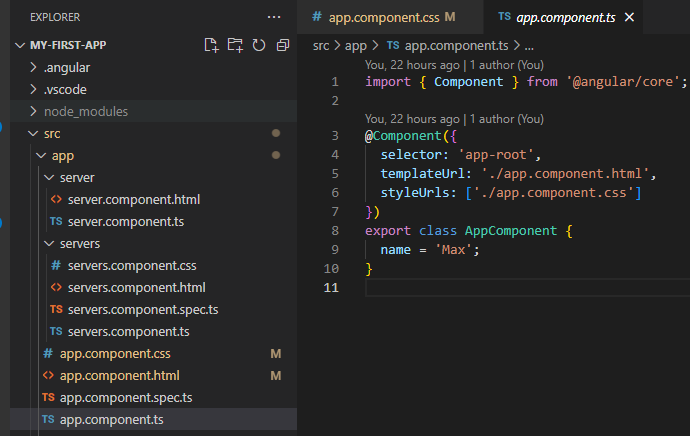
így is lehet használni…template-nek minimum lennie kell a komponensben vagy templateUrl-el kell megcsinálni



így lehet bootstrappel formázni

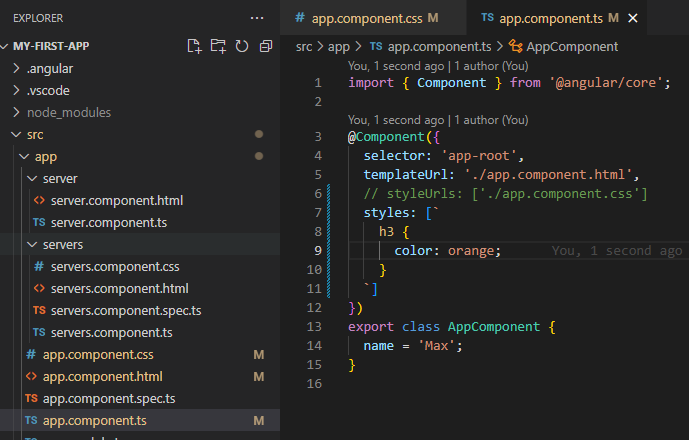


css-sel így lehet



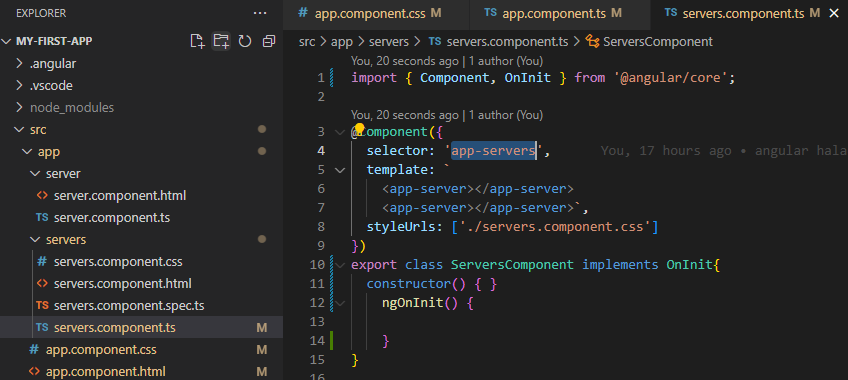
Itt lehet a html-t összekötni a css file-al, itt kapta meg a változó értékét is default-ként.

A styleUrls egy tömb, szóval több css file-t is be tud fogadni.



így is lehet közvetlenül formázni, styleUrl és style nem kombinálható, de mind2-nek array-nek kell lennie.

Component selector – fontos, hogy egyedinek kell lennie, hogy ne keveredjenek.

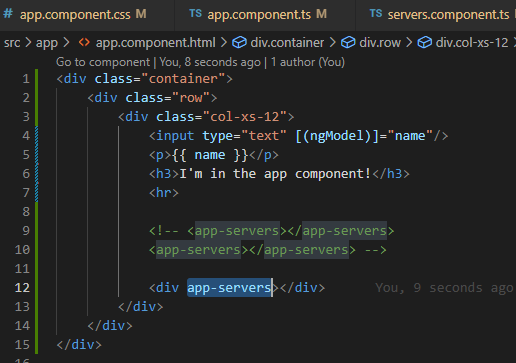


app-servers a komponent selector, de raktunk most bele construktort meg ezt az ngOnInit-et

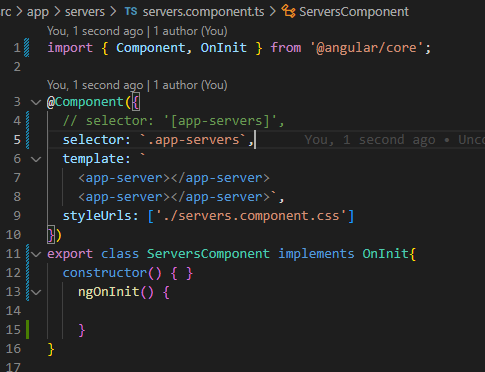
app-servers -> css selectorként funkcionál kb



így már attributum-ként fog funkcionálni és a sima <app-servers></app-servers> teg-ek már nem fognak menni



így már html teg attributum-ként tud a componens funkcionálni



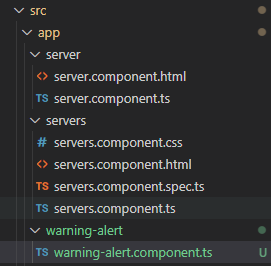
így pedig osztályként tudjuk majd használni

Fontos hogy altgr + 7 et használunk nem sima aposztrófot

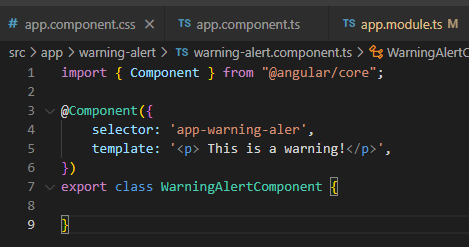


Komponensek esetén a szokás a saját element készítése és <app-servers></app-servers> ily módon történő használata.

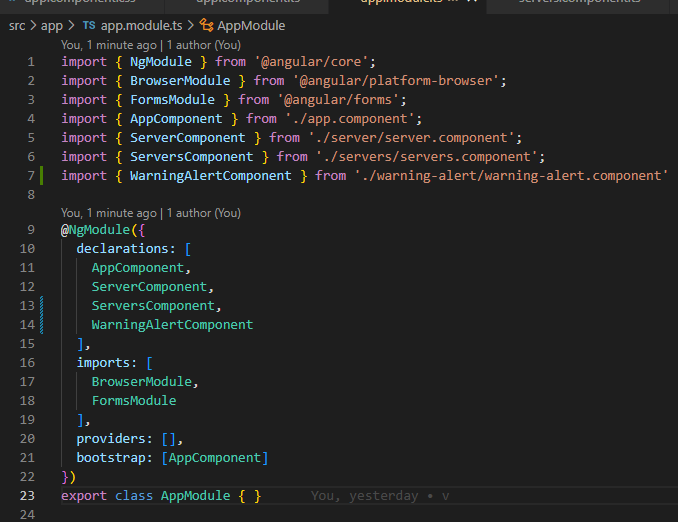
Új componens felvétele:



mappa, fájl szerkezet megcsinálva



komponens megcsinálva



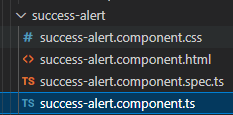
komponens a gyűjtőbe beolvasva

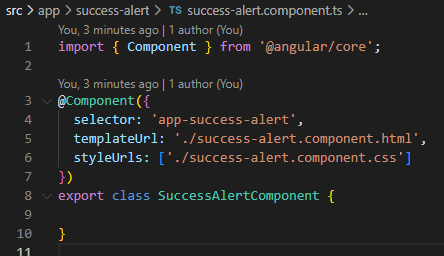


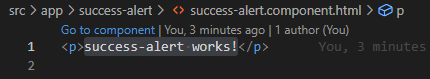
komponens használva

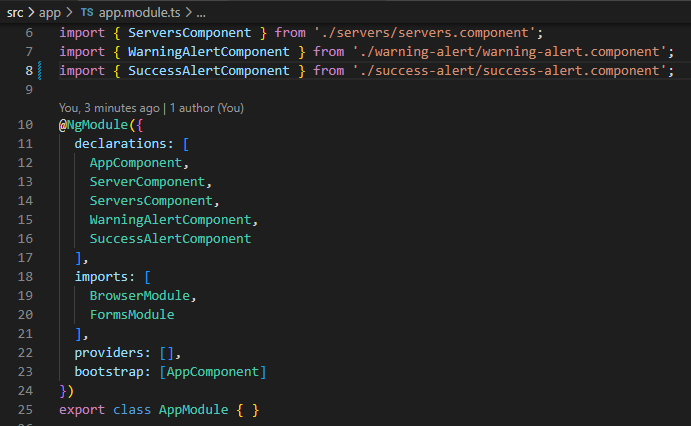
VAGY:









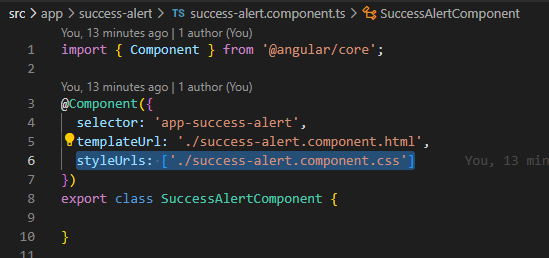


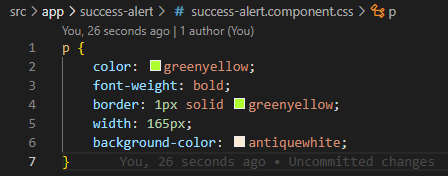


Stílus hozzáadása:

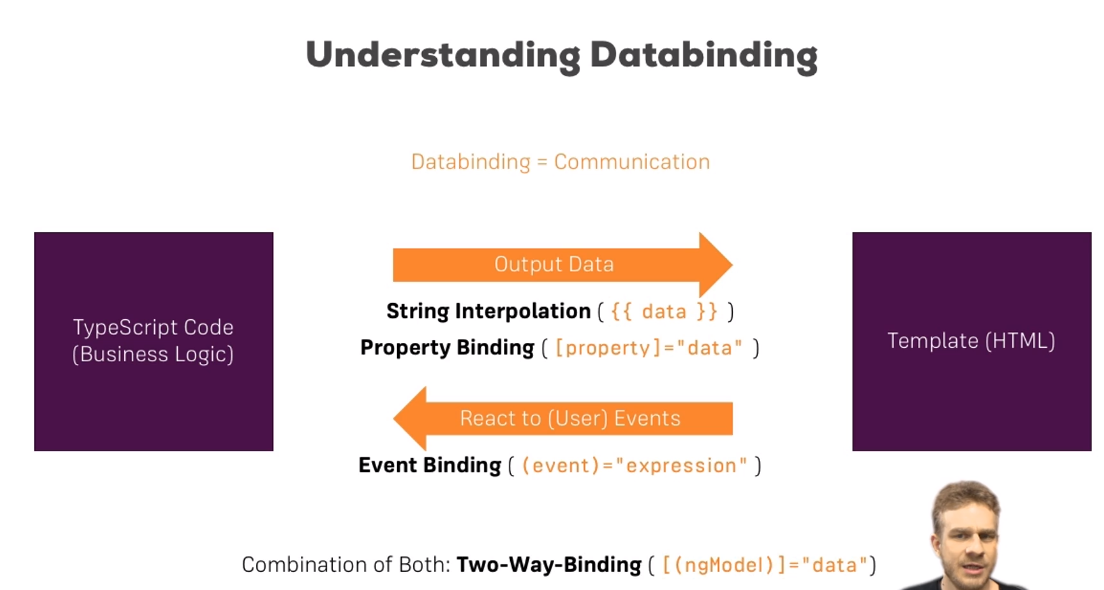


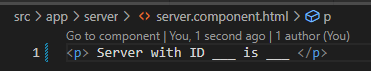
VAGY:

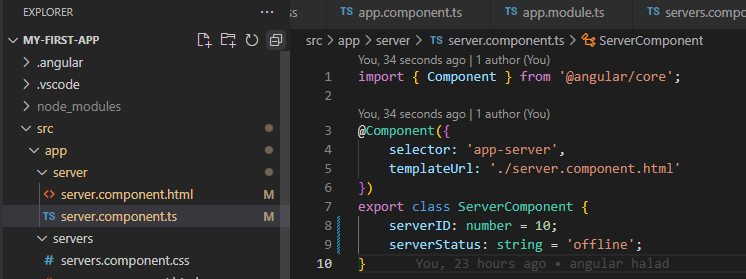




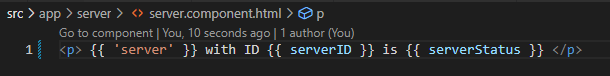
**DataBinding:**







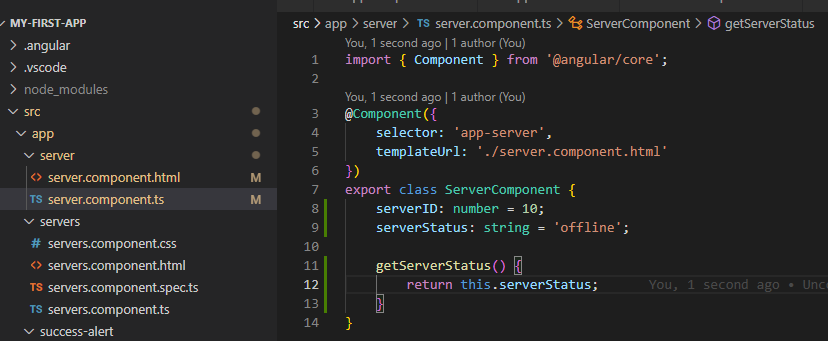
Most hardcode-olva vannak default értékek.





Annyi, hogy több soros kódot nem tudunk a {{ }} közé rakni.

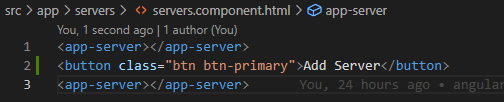
Ez string interpolation.



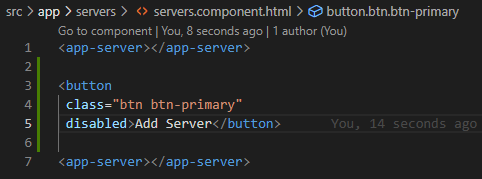
metódus return értékkel



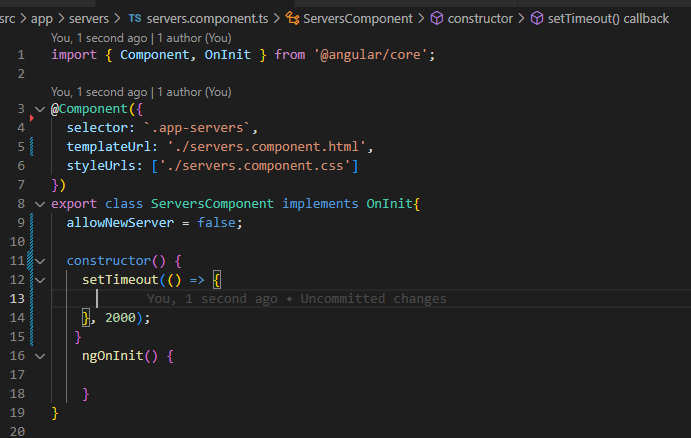
**Property binding:**



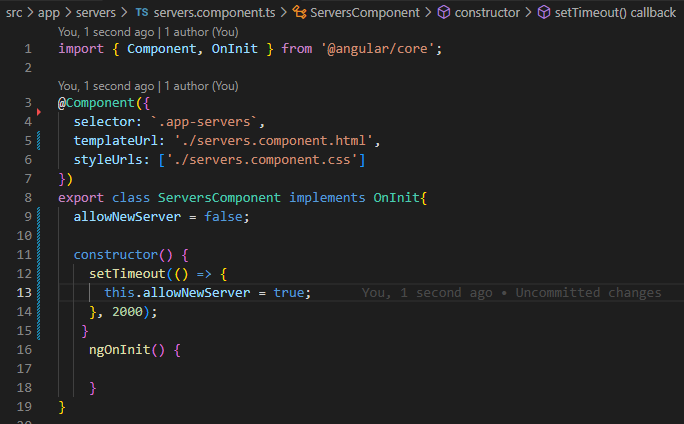




így hardcodeolva van a gomb disable attributum



arrow metódus, ()-be megy az argumentum {}-be megy a metódus body



metódus kész, már csak bindingolni kell



így lehet bindingolni

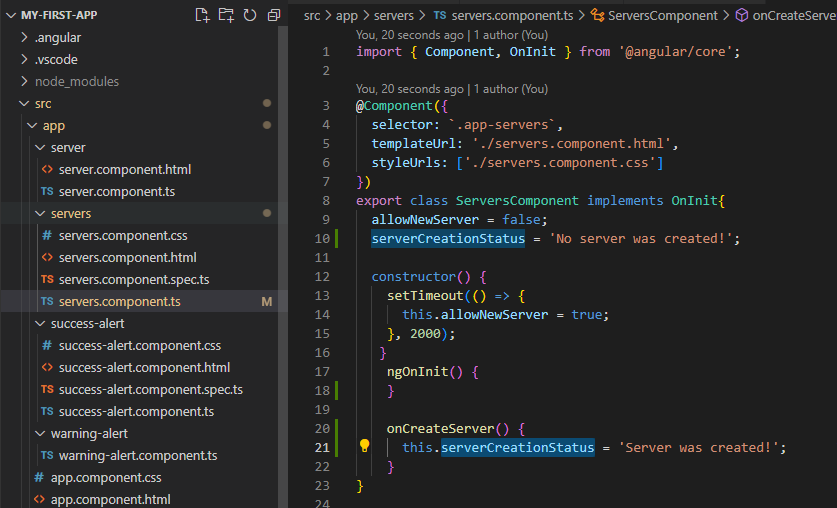
Betöltés után disabled a button 2 sec-ig, aztán meg már enabled lesz.



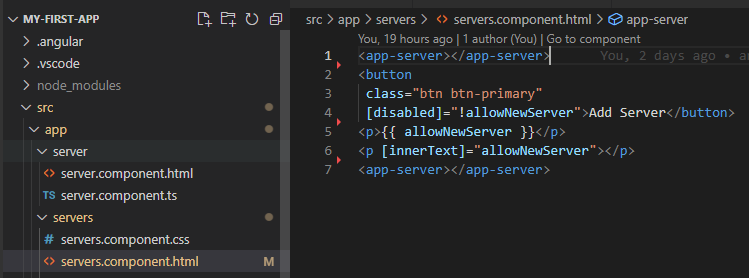
így is lehet data bindingolni

Ha valamit outputolni akarunk akkor string interpolációt használjunk, ha property-t váltani akkor property binding-ot.

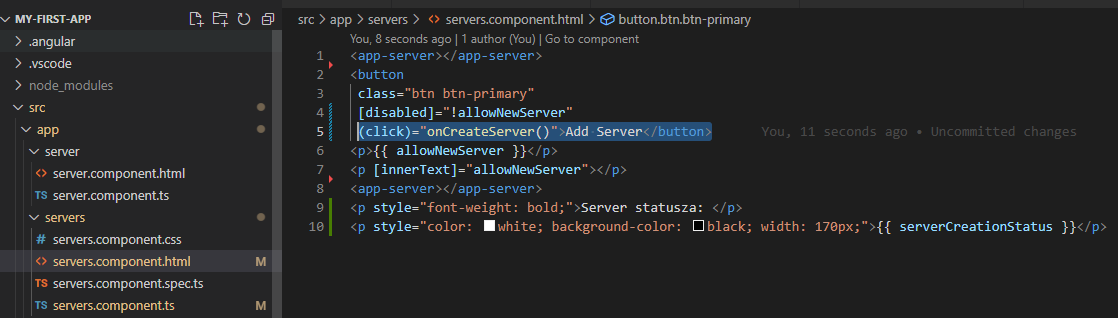
**Event Binding:**



Felvettünk egy property-t, majd egy metódussal az értékét megváltoztattuk.



Ilyen az eredeti állapot.



Gomb nyomására a serverCreationStatus értéke változik.

ow do you know to which Properties or Events of HTML Elements you may bind? You can basically bind to all Properties and Events - a good idea is to console.log()  the element you're interested in to see which properties and events it offers.

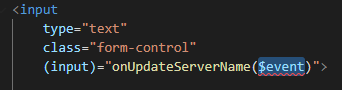
**Important**: For events, you don't bind to onclick but only to click (=> (click)).

The MDN (Mozilla Developer Network) offers nice lists of all properties and events of the element you're interested in. Googling for YOUR\_ELEMENT properties  or YOUR\_ELEMENT events  should yield nice results.

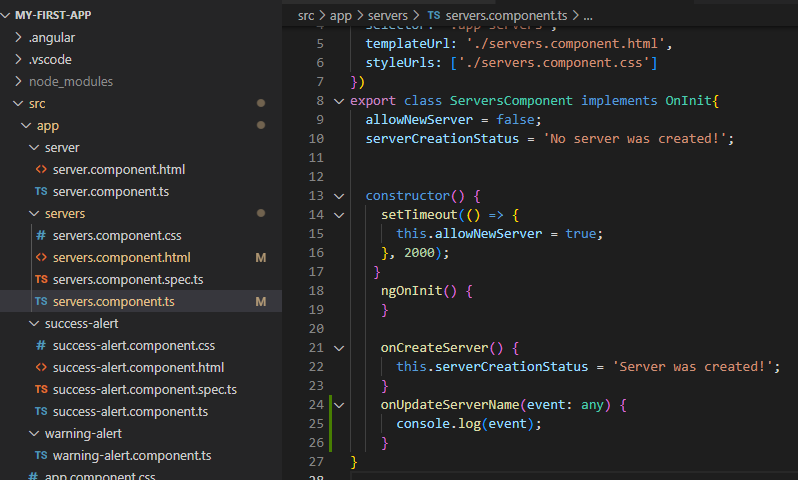
**Passing and using data with event binding:**

****

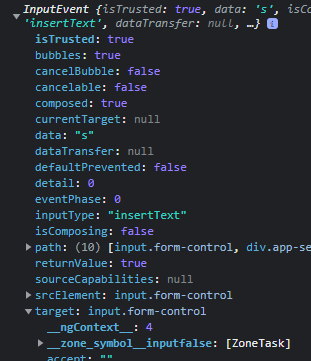
Html részét megcsináltuk

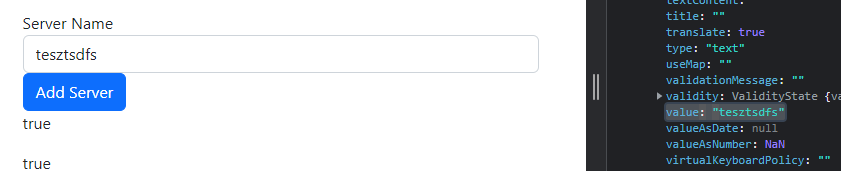


$event az event adata lesz, ez egy reserved változó név

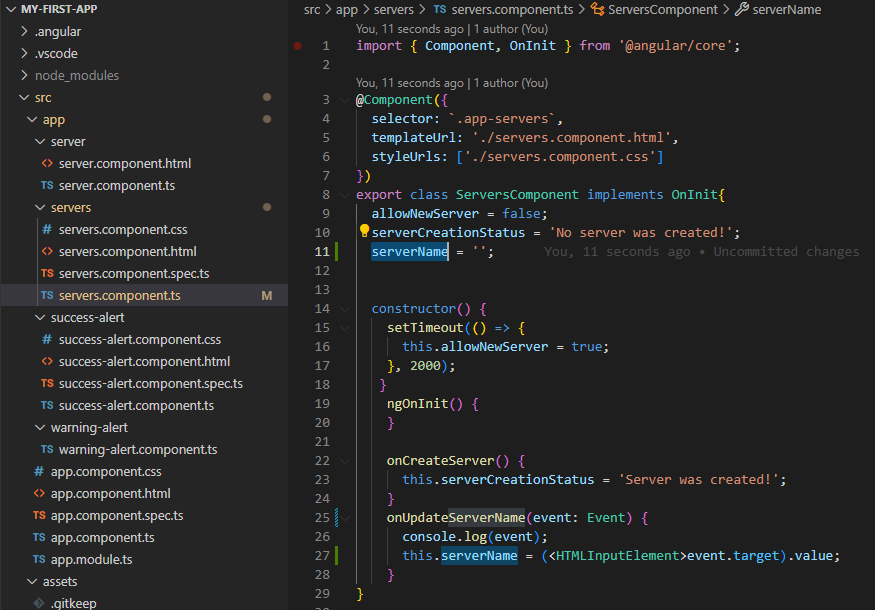


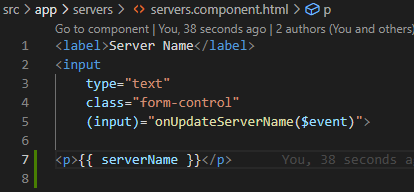
itt a metódus





Target lenyitni és legalul value: lesz az input érték. Van hogy a … -ra is rá kell nyomni.





így lehet használni ezt a $ jeles adatkötést

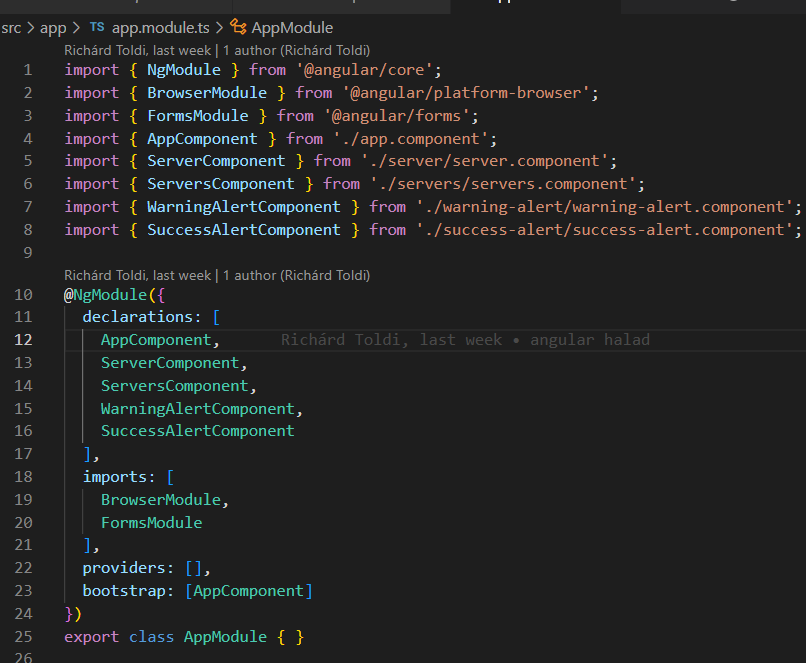
**Two-way binding:**

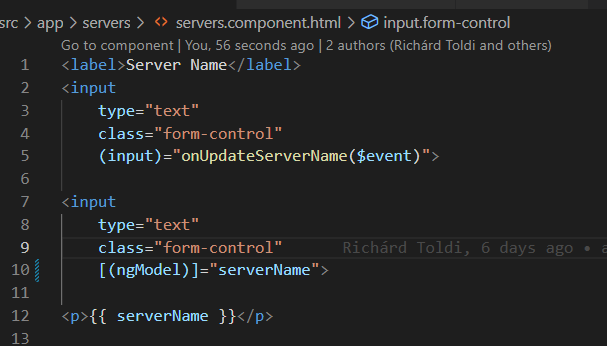
****

Important: For Two-Way-Binding (covered in the next lecture) to work, you need to enable the ngModel  directive. This is done by adding the FormsModule  to the imports[]  array in the AppModule.

You then also need to add the import from @angular/forms  in the app.module.ts file:

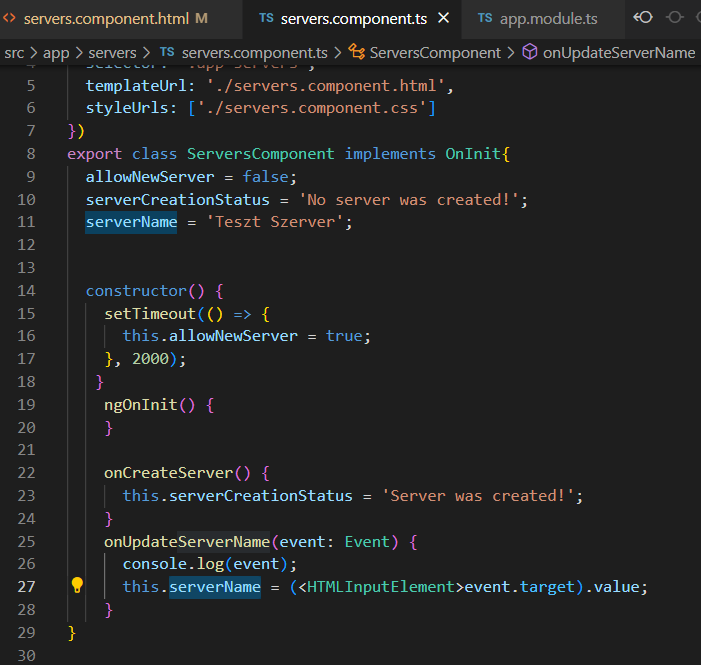
import { FormsModule } from '@angular/forms';

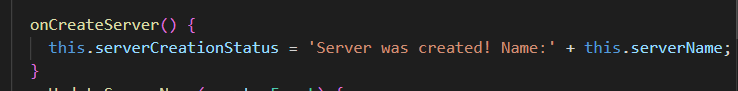




Ezzel a kötéssel, ha egy inputban megadok egy értéket ami egy p-teg közt megjeleni, az jelenik meg az ngModel-es inputban is. Szóval 1 irányú az adatkötés, ha gombnyomásra váltom a teg-ben akkor változik az inputban is, de ha az inputban változtatom akkor a teg-ben is változik.

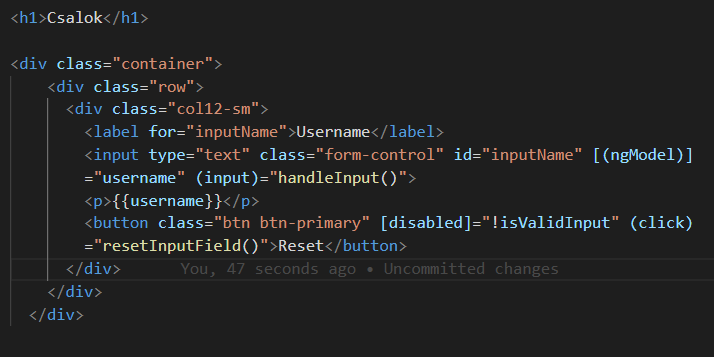
Eredeti állapot:

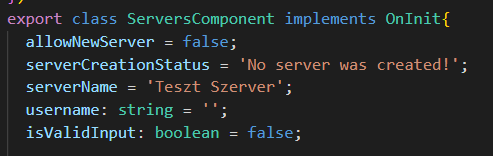




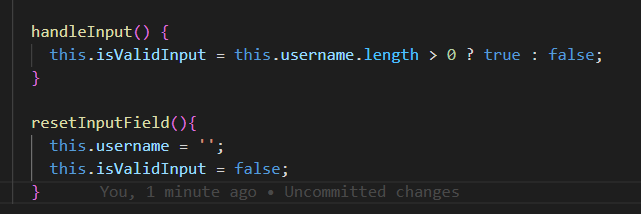
Input mező egy p teges kiíratással 2 irányú adatkötéssel.

Gomb disabled amíg az input-ba nincs valami írva, ha van akkor enabled és reseteli az input field értékét és megint disabled lesz.

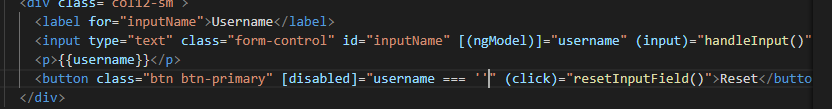




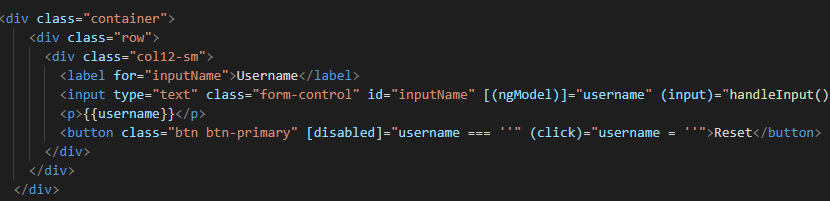
Adatok hozzá, utolsó 2 property az ami ezzel kapcsolatos.



Metódusok a működéshez.



így is lehet a disabled manipulálni, fontos hogy ez még space esetén, kattintható lesz!

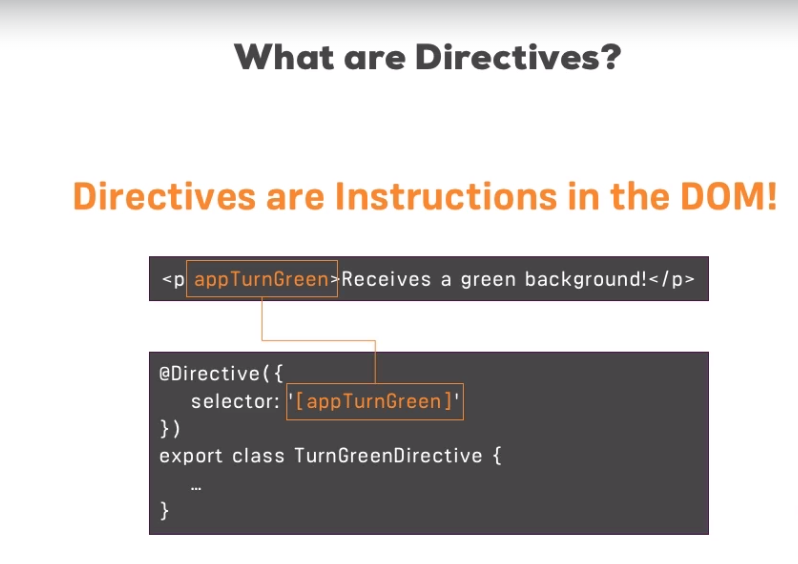


így is meg lehet oldani egyszerűen, nem feltétlen kellenek extra metódusok és változók

**Directives:**

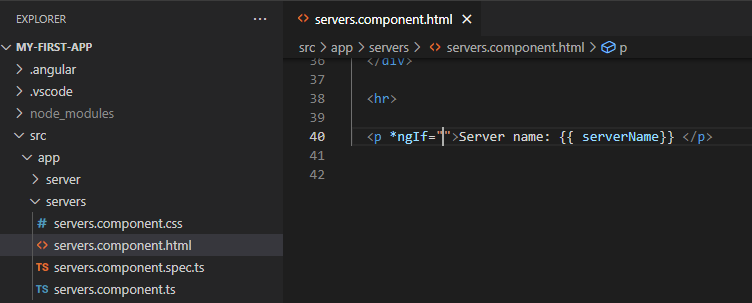
**\*ngIf - sturctural directives.**

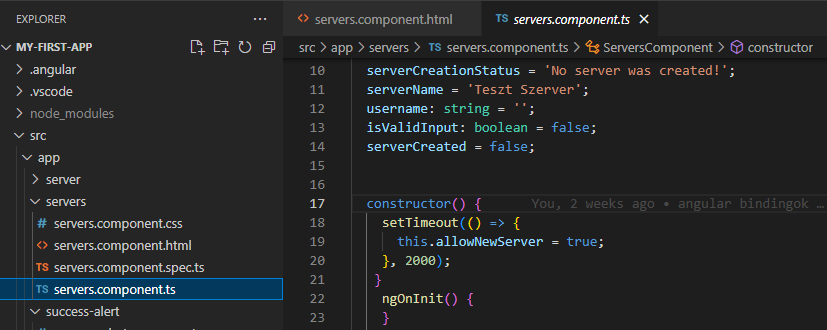
Direktívák utasítások a DOM-ban.



Csinálhatunk saját direktívákat vagy használhatjuk a build-in direktívákat is.

A direktívákat attribute selector-ként szoktuk használni.

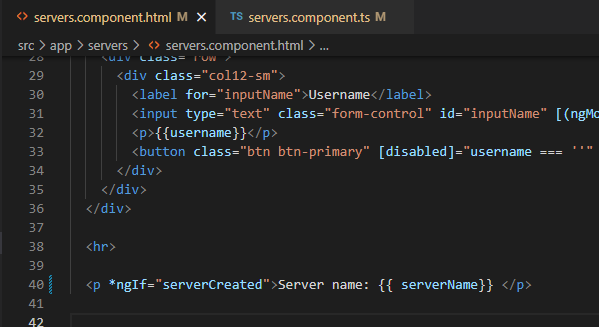


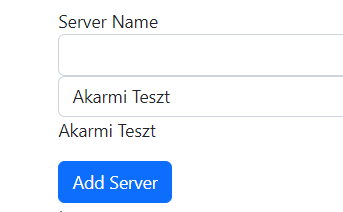


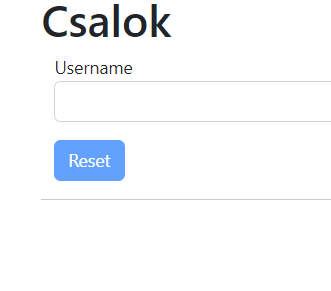
megcsináltuk a serverCreated változót



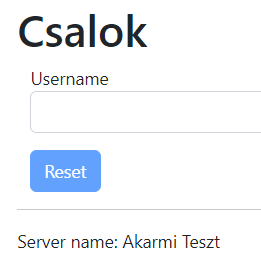
a metódusnál true-ra állítjuk a változót



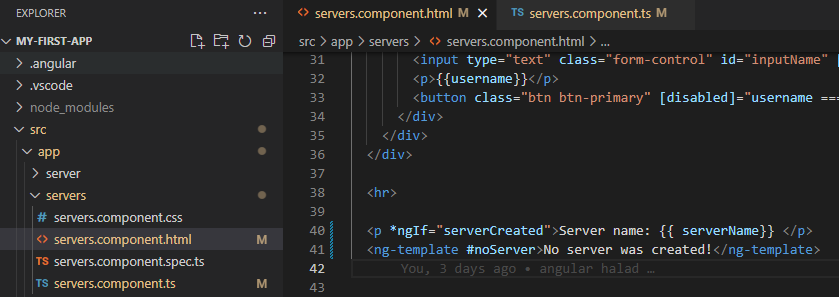


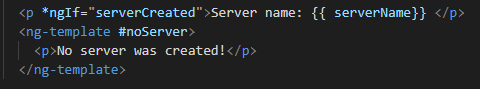


Még nem látszik a server neve, de a gombra kattintás után:

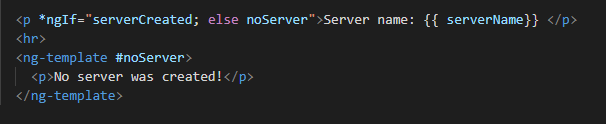


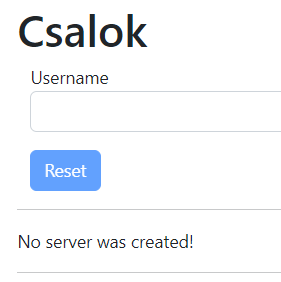
Most mivel a változó true lett így már mindig látszik a szerver neve. Fontos, hogy amikor nem látszik a név akkor nem elrejtve van a szöveg, hanem nincs ott!





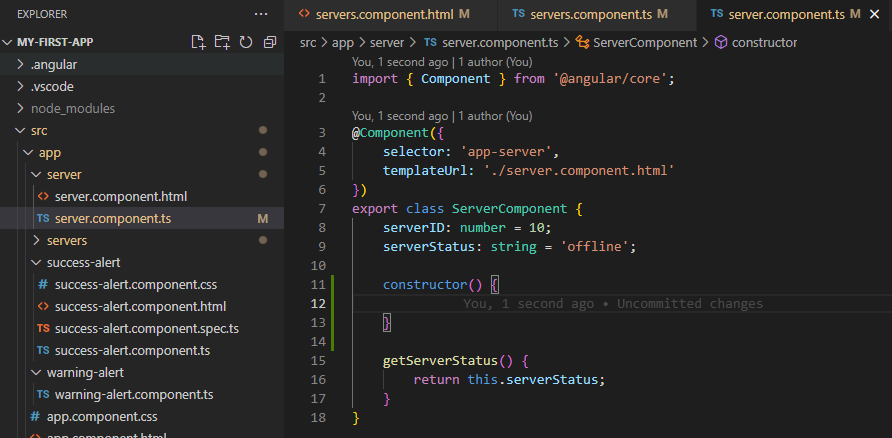
Feltétel hozzárakása:



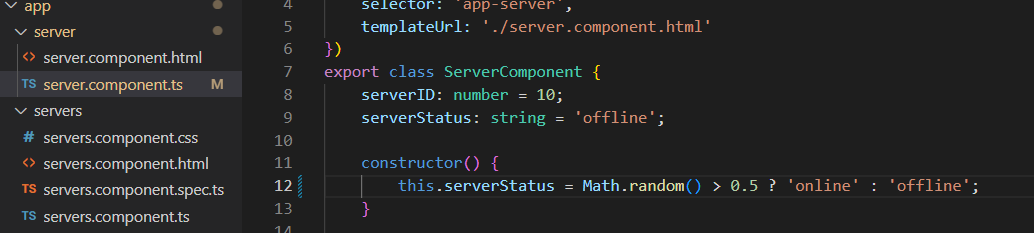


Frissítés után azonnal ez látszik, ha meg beírunk valamit vagy csak a gombot megnyomjuk akkor már a szerver nevét mutatja.

**NgStyle-al dinamikus element stilizálás: attributum direktíva**



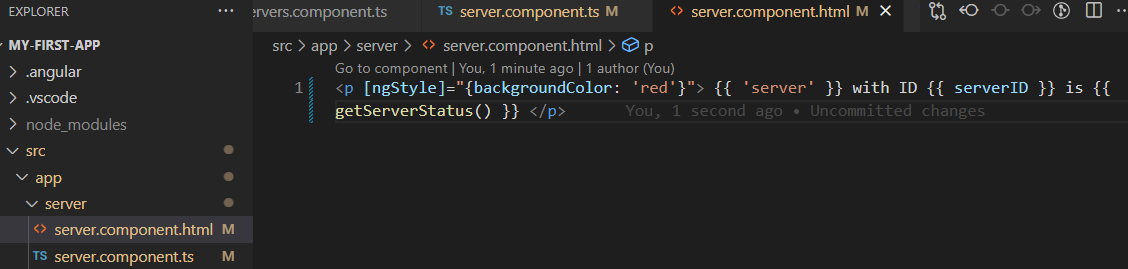
Unlike structural directives, attribute directives dont add or remove elements. They only change the element they were placed on.

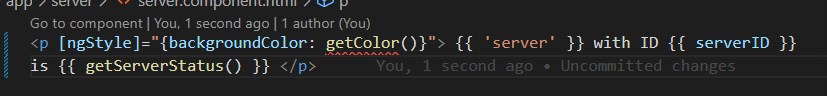


0 és 1 között random számot kisorsol, float-ot a math.random és trenary operatorral állítjuk az értéket

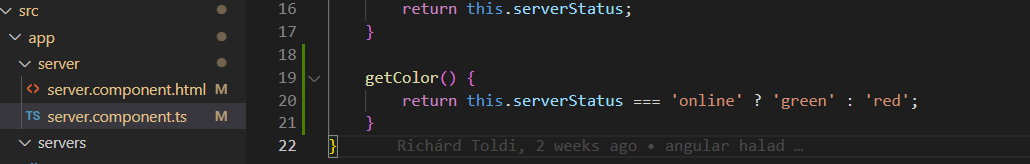


ezzel a szögletes zárójellel végzünk property bindingot





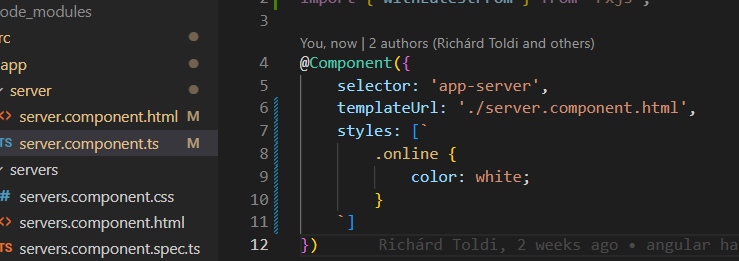
így lehet nem fix értéket magadni neki, hogy egy metódussal adjuk meg



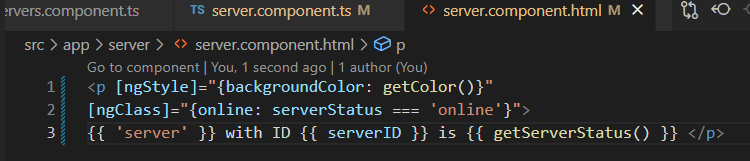
trenary expressionnel így lehet megadni, hogy ha online zöld legyen, ha off akkor red a háttere

**ngClass:**

lehetővé teszi css osztályok hozzáadását vagy eltávolítását az adott html elemnél



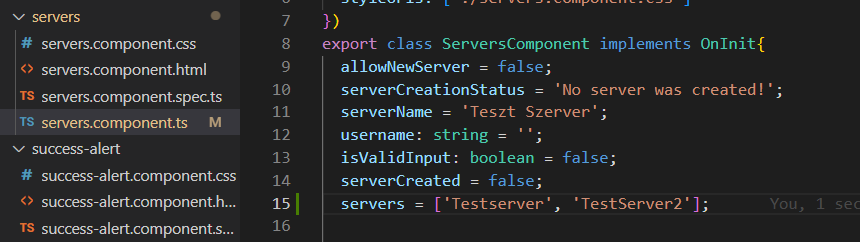
altgr + 7-el kell!

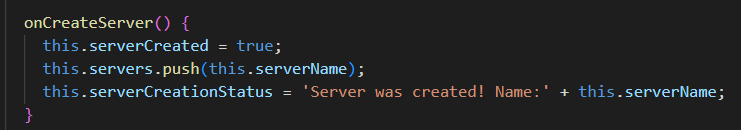


Inline lett megoldva, metódusba is ki lehet ezt szervezni, ha a szerver online akkor a betűk fehérek lesznek.

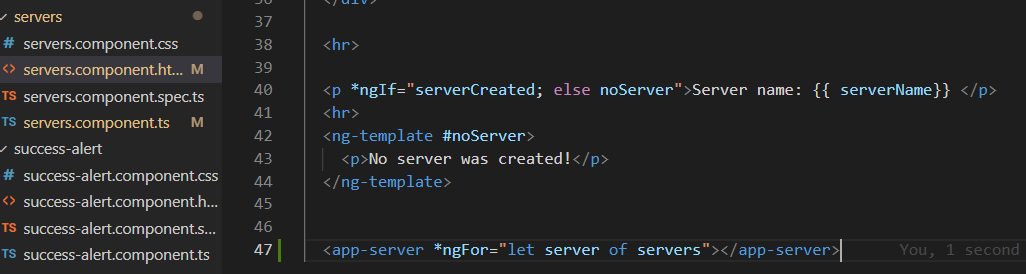
**ngFor:**

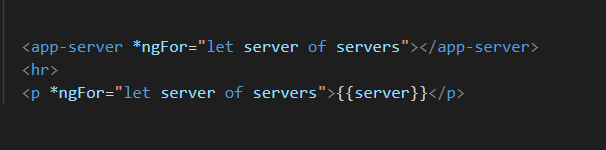
outputing list with ngFor

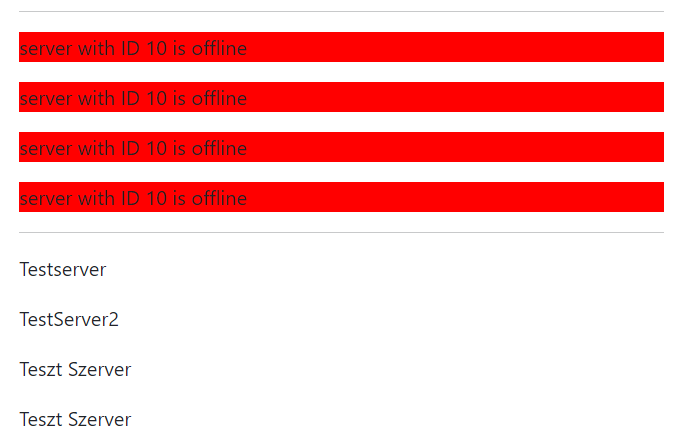


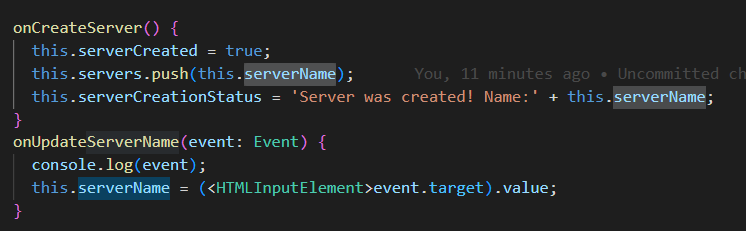


metódust a servers.push-al kiegészítettük



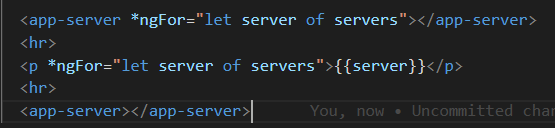






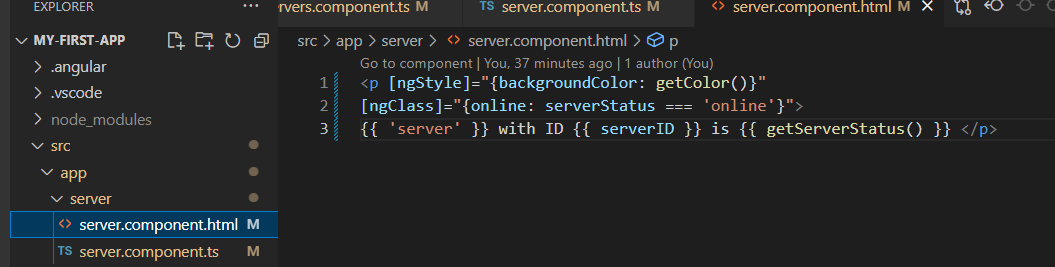
this.servers.push(this.serverName);

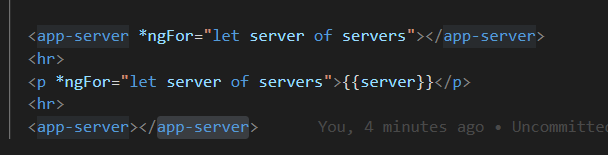
Hozzáadja az inputba beírt szerver nevét, kibővíti a listát és azt is kilistázza az ngfor.





Saját komponens:





itt van felhasználva a komponens amit csináltunk a fenti példa mintájára

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

username = "";

seeSecret = false;

count = 0;

loggerArr = [];

toggleSecret(){

this.count++;

this.logger();

this.seeSecret = !this.seeSecret;

}

logger(){

let time = new Date().toISOString();

this.loggerArr.push({"time": time, "count": this.count})

}

}

.textcolor{

color: white;

}

<div class="row">

<div class="col-xs-12">

<button (click)="toggleSecret()">Display Details</button>

<p \*ngIf="seeSecret">Secret Password = tuna</p>

<hr>

<label>Logger</label>

<p

\*ngFor="let log of loggerArr"

[ngStyle]="{backgroundColor: log.count > 4 ? 'blue' : 'white'}"

[ngClass]="{textcolor: log.count > 4}"

>{{log.count}}. {{log.time}}

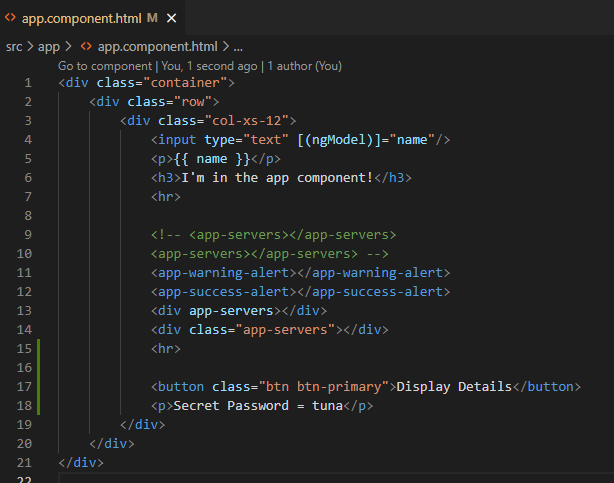
</p>

</div>

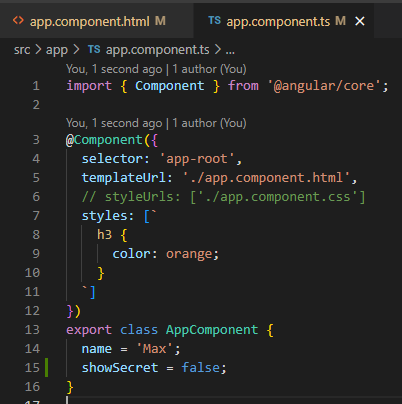
</div>

Assignment…

**Assignment:**



felvettük a szükséges változót:





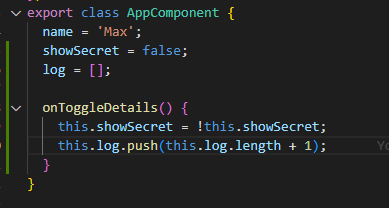
így lehet gyorsan toggle-t létrehozni, negálással, ha true akkor false lesz ha false akkor true lesz



kész is a gombos funkció

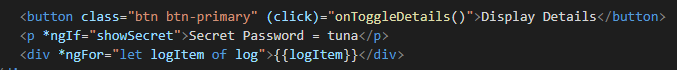


átalakítottuk hogy metódussal menjen

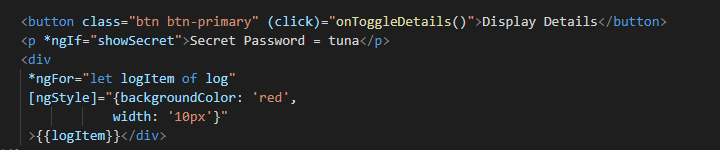


ide a metódusba átkerült a showSecret negálás

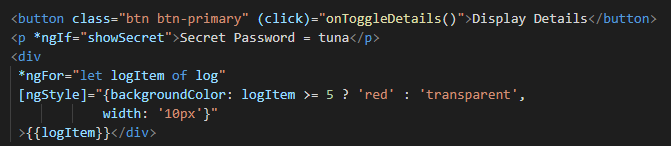
és a log tömb bővítése, egy számot ad mindig hozzá a lista hosszát + 1-t



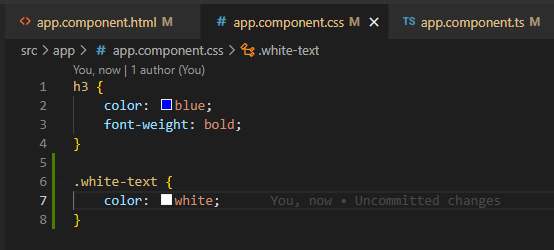
így ki lesz íratva a lista minden eleme



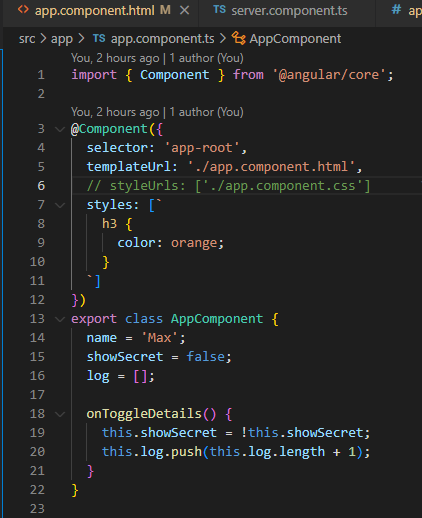
így már custom lesz a style



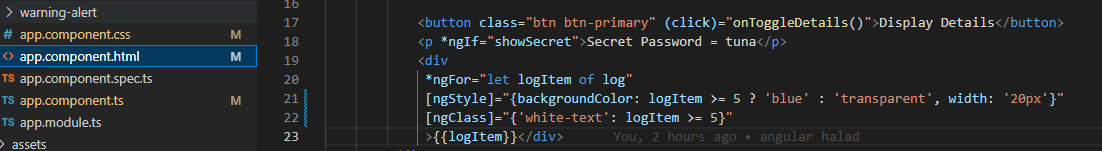
így 5 alatt a számnak nincsen háttere, felette pedig piros lesz



csináltunk egy osztályt hogy fehér legyen a betű színe

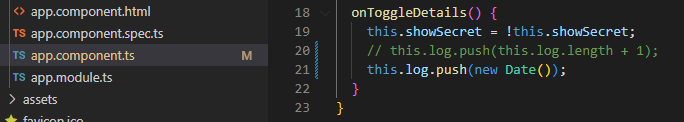


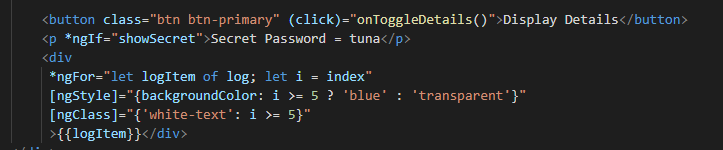
ki van kommentelve a css kapcsolat ezért nem olvasta be a css -t



így már jó, megkapja a class-t ha a tömb hossza nagyobb vagy egyenlő mint 5

**Index when using ngFor:**

****

****

így lehet az ngFor-nál az index szerint dolgozni, kinyelni az adott elem indexét és használni