ANGULAR FRONTEND FEJLESZTÉS

6. forduló

A kategória támogatója: TCS - TATA Consultancy Services

RENDELKEZÉSRE ÁLLÓ IDŐ:

10:00

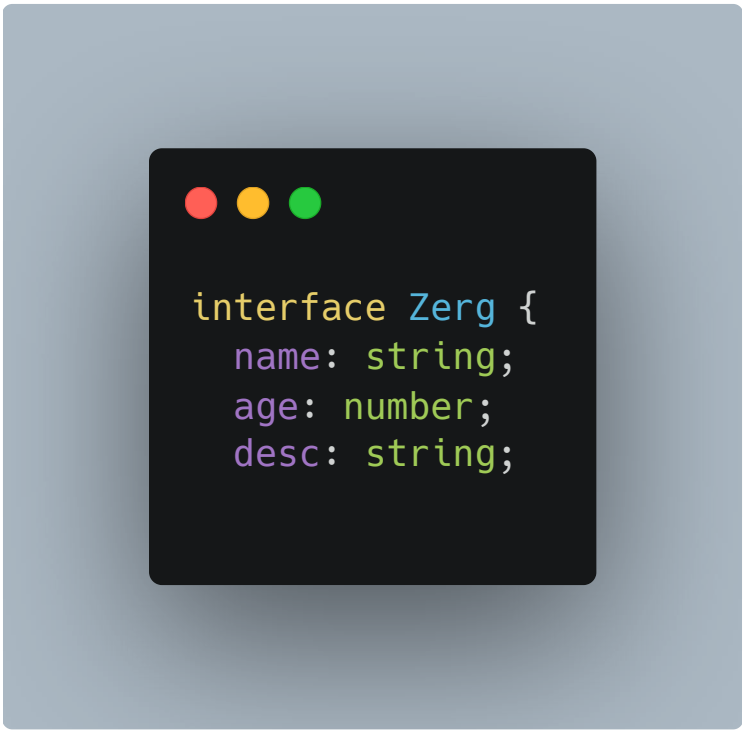
Ismertető a feladathoz

Felhasznált idő: 00:57/10:00

Elért pontszám: 0/20

1. feladat 0/8 pont

Adott az alábbi TypeScript interface:



Az alábbi objektumok közül melyik van helyesen típusozva? (Tehát nem kapunk rá compilation errort.)

Válaszok

- [x] *const zerg1: Partial<Zerg> = { name: "zerg1" };*
- [x] *const zerg2: Omit<Zerg, "desc" | "age"> = { name: "zerg2" };*
- [ ] *const zerg3: Omit<Zerg, "desc" & "age"> = { name: "zerg3" };*
- [ ] *const zerg4: Pick<Zerg, "name" | "age"> = { name: "zerg4" };*
- [x] *const zerg5: Zerg | { name: string } = { name: "zerg5" };*
- [ ] *const zerg6: Zerg & { name: string } = { name: "zerg6" };*
- [ ] *const zerg7: Extract<keyof Zerg, keyof { name: string }> = { name: "zerg7" };*

Magyarázat

1. a Partial-ban minden property opcionális, tehát elég a name.
2. Az Omit-al egy új típust kapunk, ami megegyezik a Zerg-el, kivéve a desc és age property-ket, tehát elég a name.
3. A "desc & age" never-re értekelődik ki, tehát Omit<Zerg, never> lesz ami megegyezik a Zerg-el, nem elég a name.
4. A Pick-el olyan típust készítünk, amin a Zerg-en lévő name és age property-k lesznek, tehát nem elég a name.
5. A típusunk Zerg VAGY { name: string }, tehát elég a name
6. A típusunk Zerg ÉS { name: string }, ami megegyezik a Zerg-el, tehát nem elég a name

7. Az Extract teljesen máshogy működik, itt a kapott típus a "name" lesz, azaz csak és kizárólag a "name" sztringgel működne.

További információk: https://www.typescriptlang.org/docs/handbook/utility-types.html

2. feladat 0/12 pont

Adott az alábbi komponens template (részlet):

```
<div *ngIf="isButtonVisible">
  <button #myButton mat-button>Press me!</button>
</div>
```

És a hozzá tartozó komponens ts file:

```
@Component({
  selector: 'app-starcraft,
  templateUrl: './starcraft.component.html',
  styleUrls: ['./starcraft.component.scss']
})
export class StarcraftComponent {
  public isButtonVisible = false;
  @ViewChild('myButton', { static: true }) public myButton: ElementRef;
  public showButton(): void {
    this.isButtonVisible = true;
    this.myButton.nativeElement.focus();
  }
}
```

A **showButton** függvény meghívásakor azt szeretnénk, hogy legyen látható a gomb, és kapjon fókuszt, azonban a kódban több hiba is van. Az alábbiak közül mik azok, amik feltétlenül szükségesek, hogy megfelelően működjön?

Válaszok

- [x] A **ViewChild** 2. paraméterében **static: false** kell
- [x] A **ViewChild** 2. paraméterében read: **ElementRef** kell
- [ ] Az **ElementRef** generikus, így kellene használni: **ElementRef<HTMLButtonElement>**
- [ ] A **this.myButton-**on nincs **nativeElement,** közvetlenül **focus()-**t kellene hívni
- [x] A **this.isButtonVisible = true;** sor után **changeDetectorRef.detectChanges()**-t kell hívni
- [ ] A **this.isButtonVisible = true**; sor után **changeDetectorRef.markForCheck()-t** kell hívni
- [ ] A ViewChild első paramétere nem egyezhet meg az adattag nevével, **public myButton** helyett valami más nevet kell adni

Magyarázat

1. Static: false kell, mert az *ngIf miatt az elem csak később kerül be a DOM-ba
2. read: ElementRef kell, mert defaultban a MatButton komponenst Read-elné
3. Az ElementRef valóban generikus, de opcionális a generikus paramétere, nem feltétlenül szükséges
4. Van rajta nativeElement. Bővebben: https://angular.io/api/core/ElementRef
5. Kell a detectChanges, mert az isButtonVisible true-ra állítása után nem fog azonnal megjelenni az elem. Hiába használunk Default ChangeDetection-t, csak a függvény lefutása után futna le magától a ChangeDetector.
6. markForCheck nem elég, mert azonnal futtatnunk kell a ChangeDetectort. Default ChangeDetection strategy-vel nem is lenne értelme meghívnunk.
7. Nincs ilyen megkötés.