

Signals

<https://angular.dev/guide/signals>

Signals

Kas yra signalai?

- Signalai yra **reaktyvūs primityvai**, leidžiantys **valdyti komponentų būseną** paprastesniu ir našesniu būdu nei **Observable**.
- Jie **automatiškai** atnaujina prilausomus komponentus be papildomų **subscribe()**.
- Naudojami kaip alternatyva Observable **UI būsenos valdymui**.

Signals vs Observables

	Signals	Observables
Reaktyvumas	Sinchroniniai, automatiniai, atnaujinimai	Reikalingi subscribe() ir unsubscribe()
Performance	Optimizuoti detaliams view management	Gali būti reikalo renderinti, jei neteisingai naudojami
Sudėtingumas	Paprasta, deklaratyvu	Reikia valdyti subscription, sudėtinga sintaksė
Data change	Reikšmė keičiama vietoje	Reikšmė transliuojama
Naudojimas su kitais įrankiais	Lengva su computed() ir effect()	Reikia pipe() , map() , tap()

Signals vs Observables

- **Signal** laiko synchroninę reikšmę ir atsinaujina automatiškai.
- **Observable** veikia asynchroniškai ir reikalauja **subscribe()**.
- **Observable** geriau tinka duomenų srautams, **Signal** – vienai reaktyviai būsenai.

Signals

signal()

- Pagrindinis reaktyvus komponentas, leidžiantis atnaujinti statines reikšmes.

```
export class SignalComponent {  
    count = signal(0);  
  
    increment() {  
        this.count.set(this.count() + 1);  
    }  
}
```

Signals computed()

- Perskaičiuoja reikšmes, kai kitų **signalų** reikšmės keičiasi.

```
export class ComputedComponent {  
    count = signal(0);  
  
    doubleCount = computed(() => this.count() * 2);  
  
    increment() {  
        this.count.set(this.count() + 1);  
    }  
}
```

Signals

effect()

- Vykdo kodą, kai signal reikšmė pasikeičia.
- Gali būti aprašomas tik "injectable kontekste" - konstruktoriuje.

```
export class EffectComponent {  
    count = signal(0);  
  
    constructor() {  
        effect(() => {  
            console.log('Count:', this.count());  
        });  
    }  
  
    increment() {  
        this.count.set(this.count() + 1);  
    }  
}
```

Signals

linkedSignal()

- Signal, kuris leidžia sekti kitų signalų reikšmes ir keisti source, nekeičiant kintamojo.

```
export class LinkedSignalComponent {  
    valueA = signal('Reikšmė is A');  
    valueB = signal('Reikšmė is B');  
  
    selectedSource = signal(this.valueA());  
  
    selectedValue = linkedSignal(() => this.selectedSource());  
  
    selectA() {  
        this.selectedSource.set(this.valueA());  
    }  
  
    selectB() {  
        this.selectedSource.set(this.valueB());  
    }  
}
```

Signals

Komponentų input signalai

- Nauja sintaksė, naudojant **Signal** pakeičia **@Input** dekoratoriu.
- **input()** - signal, vietoje **@Input**
- **input.required()** - vietoje **@Input({ required: true })**
- **output()** - vietoje **@Output()**

```
@Component({
  selector: 'app-inputs-child',
  imports: [],
  template: `<h1>inputs-child</h1>
    <h2>Child count {{ count() }}</h2>`,
})
export class InputsChildComponent implements OnInit {
  count = input.required<number>();
  rendered = output();

  ngOnInit() {
    this.rendered.emit();
  }
}
```

Signals

keitimas į/iš Observable - <https://angular.dev/ecosystem/rxjs-interop>

```
import { toSignal } from '@angular/core/rxjs-interop'
```

```
import { toObservable } from '@angular/core/rxjs-interop';
```

Signals

resource() - darbas su HTTP

- Developer preview - gali keistis.
- Sukuria signal-based struktūra, dirbtį su “resursais” (duomenimis, kurie kis)

```
export class ResourceComponent {
    #service = inject(HttpService);

    reloadSignal = signal(0);

    user = resource({
        loader: () => this.#service.fetchUsers(),
        request: () => this.reloadSignal(),
    });

    firstUser = computed(() => {
        if (this.user.status() === ResourceStatus.Loading) return 'Loading ...';
        if (this.user.status() === ResourceStatus.Error) return 'Error ...';
        return Object.values(this.user.value()?[0].name).join(' ');
    });

    reload() {
        this.reloadSignal.set(this.reloadSignal() + 1);
    }
}
```

Signals

Task #2

1. Pakeiskite **@Input** į **input**.
2. Pakeiskite visus **Observable** naudojimus į **signal** (išskyrus patį **HttpClient**);
3. **(Optional)**: naudokite **resource** ir **fetch** vietoje **HttpClient**.

Lifto Muzika.

