Component Interaction

Pass data from parent to child with input binding

https://angular.io/guide/component-interaction#component-interaction

//hero-child.component.ts

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

import { Hero } from './hero';

@Component({

selector: '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;

}

The second @Input aliases the child component property name masterName as 'master'.

The HeroParentComponent nests the child HeroChildComponent inside an \*ngFor repeater, binding its master string property to the child's master alias, and each iteration's hero instance to the child's hero property.

//hero-parent.component.ts

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

import { HEROES } from './hero';

@Component({

selector: 'hero-parent',

template: `

<h2>{{master}} controls {{heroes.length}} heroes</h2>

<hero-child \*ngFor="let hero of heroes"

[hero]="hero"

[master]="master">

</hero-child>

`

})

export class HeroParentComponent {

heroes = HEROES;

master = 'Master';

}

**Parent listens for child event**

Ex:

// voter.component.ts

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

@Component({

selector: 'my-voter',

template: `

<h4>{{name}}</h4>

<button (click)="vote(true)" [disabled]="voted">Agree</button>

<button (click)="vote(false)" [disabled]="voted">Disagree</button>

`

})

export class VoterComponent {

@Input() name: string;

@Output() onVoted = new EventEmitter<boolean>();

voted = false;

vote(agreed: boolean) {

this.onVoted.emit(agreed);

this.voted = true;

}

}

//votetaker.component.ts

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

@Component({

selector: 'vote-taker',

template: `

<h2>Should mankind colonize the Universe?</h2>

<h3>Agree: {{agreed}}, Disagree: {{disagreed}}</h3>

<my-voter \*ngFor="let voter of voters"

[name]="voter"

(onVoted)="onVoted($event)">

</my-voter>

`

})

export class VoteTakerComponent {

agreed = 0;

disagreed = 0;

voters = ['Mr. IQ', 'Ms. Universe', 'Bombasto'];

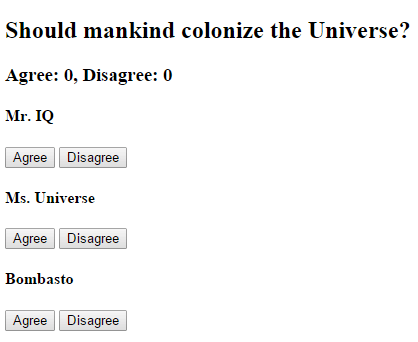
onVoted(agreed: boolean) {

agreed ? this.agreed++ : this.disagreed++;

}

}

Result:



Parent interacts with child via local variable

// countdown-timer.component.ts

import { Component, OnDestroy, OnInit } from '@angular/core';

@Component({

selector: 'countdown-timer',

template: '<p>{{message}}</p>'

})

export class CountdownTimerComponent implements OnInit, OnDestroy {

intervalId = 0;

message = '';

seconds = 11;

clearTimer() { clearInterval(this.intervalId); }

ngOnInit() { this.start(); }

ngOnDestroy() { this.clearTimer(); }

start() { this.countDown(); }

stop() {

this.clearTimer();

this.message = `Holding at T-${this.seconds} seconds`;

}

private countDown() {

this.clearTimer();

this.intervalId = window.setInterval(() => {

this.seconds -= 1;

if (this.seconds === 0) {

this.message = 'Blast off!';

} else {

if (this.seconds < 0) { this.seconds = 10; } // reset

this.message = `T-${this.seconds} seconds and counting`;

}

}, 1000);

}

}

// countdown-parent.component.ts

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

import { CountdownTimerComponent } from './countdown-timer.component';

@Component({

selector: 'countdown-parent-lv',

template: `

<h3>Countdown to Liftoff (via local variable)</h3>

<button (click)="timer.start()">Start</button>

<button (click)="timer.stop()">Stop</button>

<div class="seconds">{{timer.seconds}}</div>

<countdown-timer #timer></countdown-timer>

`,

styleUrls: ['demo.css']

})

export class CountdownLocalVarParentComponent { }

Result:

