CODEGYM

Raising the bar

Components Interaction

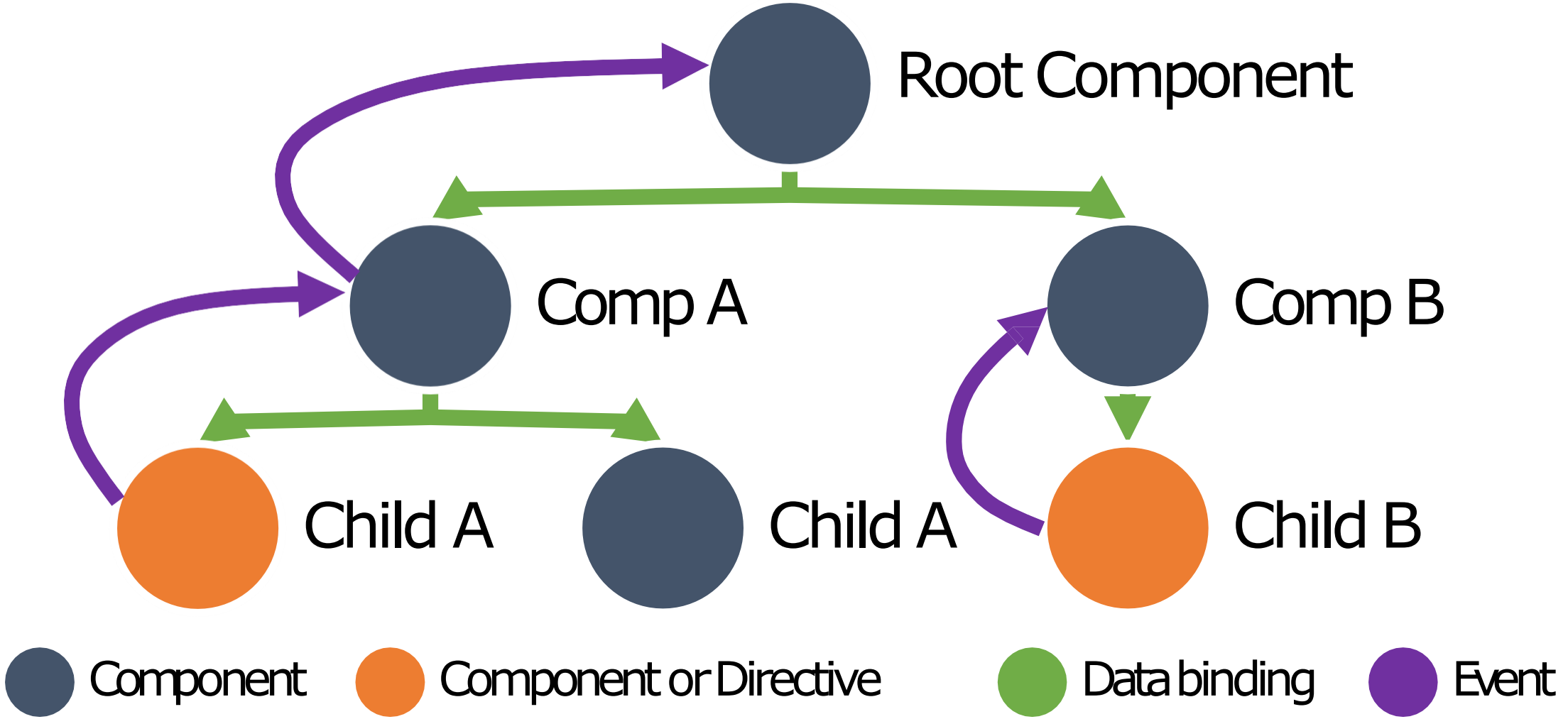
CODEGYM

Mục tiêu

- Truyền được dữ liệu vào component.
- Giao tiếp được giữa các component.

Tương tác giữa các component

Component interactions

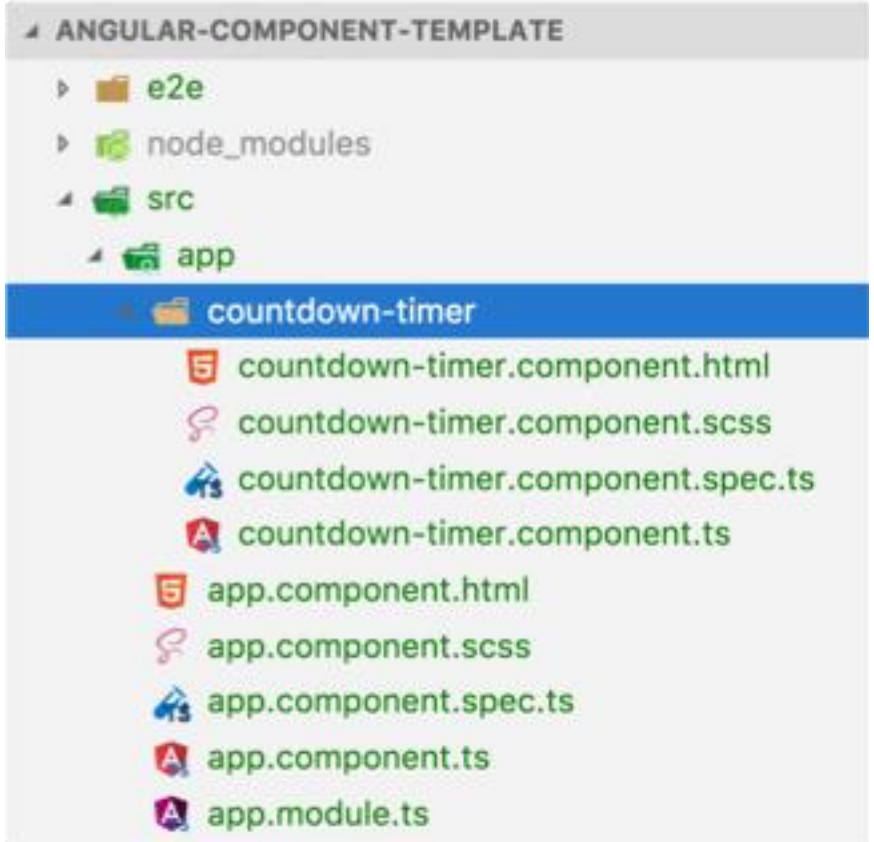


Truyền dữ liệu với @Input

- Component cha có thể truyền dữ liệu vào cho component con bằng cách sử dụng @Input.
- Chuẩn hóa dữ liệu đầu vào với get/set, ngOnChanges.
- Tạo alias cho @Input property.

Truyền dữ liệu với @Input

ng generate component countdown-timer



Truyền dữ liệu với @Input

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

@Component({
  selector: 'app-countdown-timer',
  templateUrl: './countdown-timer.component.html',
  styleUrls: ['./countdown-timer.component.scss']
})
export class CountdownTimerComponent implements OnInit, OnDestroy {

  @Input()
  seconds = 11;

}
```

Decorator @Input để đánh dấu property **seconds** sẽ là input mà component này cần nhận dữ liệu.
Property **seconds** có thể được thiết lập giá trị mặc định, nếu người dùng không truyền vào giá trị nào.

Truyền dữ liệu với @Input

- Giả sử chúng ta tạo component để hiện thị Countdown
- Component này cho phép người dùng truyền vào thời gian cần đếm ngược.
- Nếu người dùng không truyền vào thời gian cần đếm, chúng ta có thể thiết lập một giá trị mặc định.

Code thực thi của component

```
export class CountdownTimerComponent implements OnInit,
OnDestroy {
  private intervalId = 0;
  message = '';
  remainingTime:  number;

  @Input()
  seconds = 11;
  // ...
}
```

countdown-timer/countdown-timer.component.ts

Code thực thi của component

```
export class CountdownTimerComponent implements OnInit,
OnDestroy {
  // ...
  ngOnInit() {}
  ngOnDestroy() {}
  clearTimer() {}
  start() {}
  stop() {}
  reset() {}
  private countDown() {}
}
```

Angular component lifecycle, các hàm này sẽ được Angular tự động gọi khi khởi tạo(ngOnInit) và khi hủy(ngOnDestroy) component.

countdown-timer/countdown-timer.component.ts

Template của component

```
<h2>Countdown Timer</h2>
<div class="group-actions">
  <button (click)="start()">Start</button>
  <button (click)="stop()">Stop</button>
  <button (click)="reset()">Reset</button>
</div>
<div class="timer">
  <span class="timer-text">{{ remainingTime }}</span>
</div>
<p>{{ message }}</p>
```

countdown-timer/countdown-timer.component.html

Template của App component (parent)

```
<app-countdown-timer  [seconds]="6">
</app-countdown-timer>
```

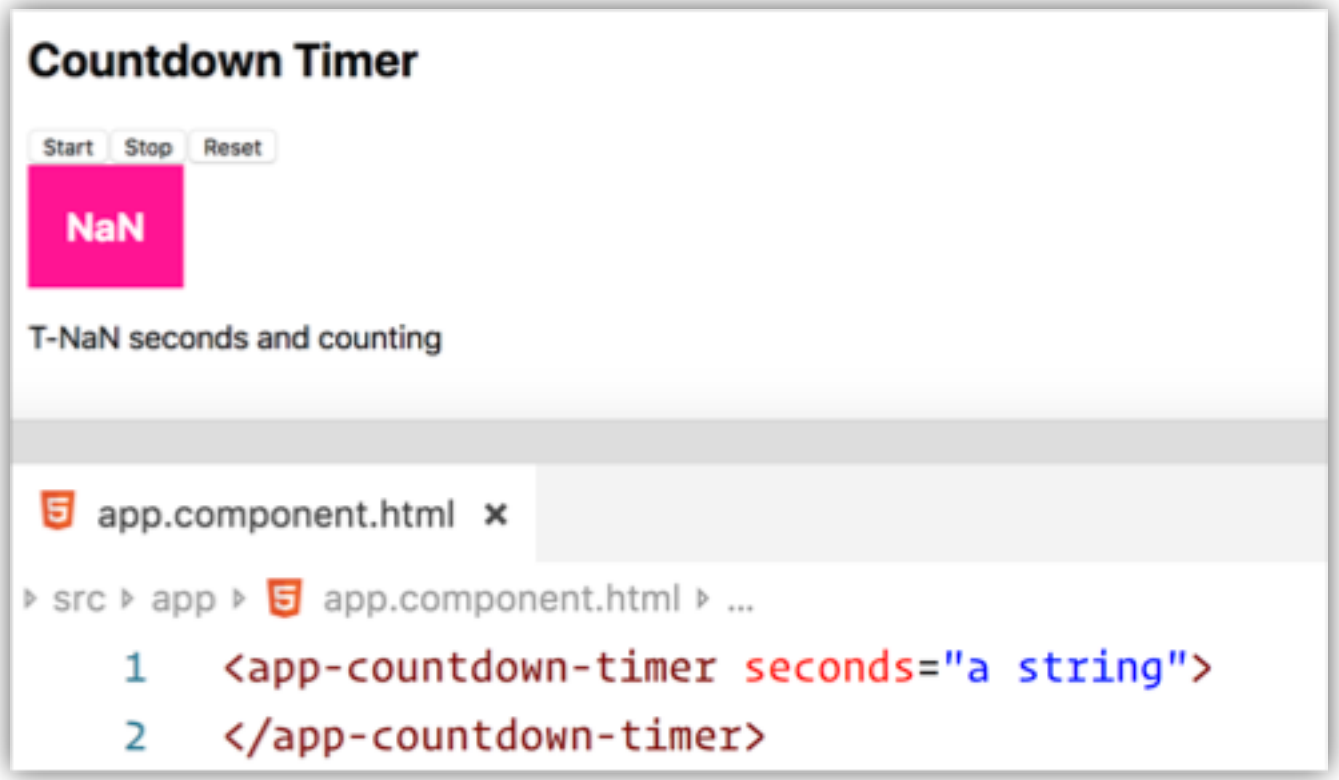
app.component.html

Kết quả


Countdown Timer
Start
Stop
Reset
6
Click start button to start the Countdown

Chuẩn hóa dữ liệu đầu vào

- Sẽ thế nào nếu người dùng truyền vào **seconds** là một kiểu dữ liệu khác **number**?



Chuẩn hóa dữ liệu đầu vào

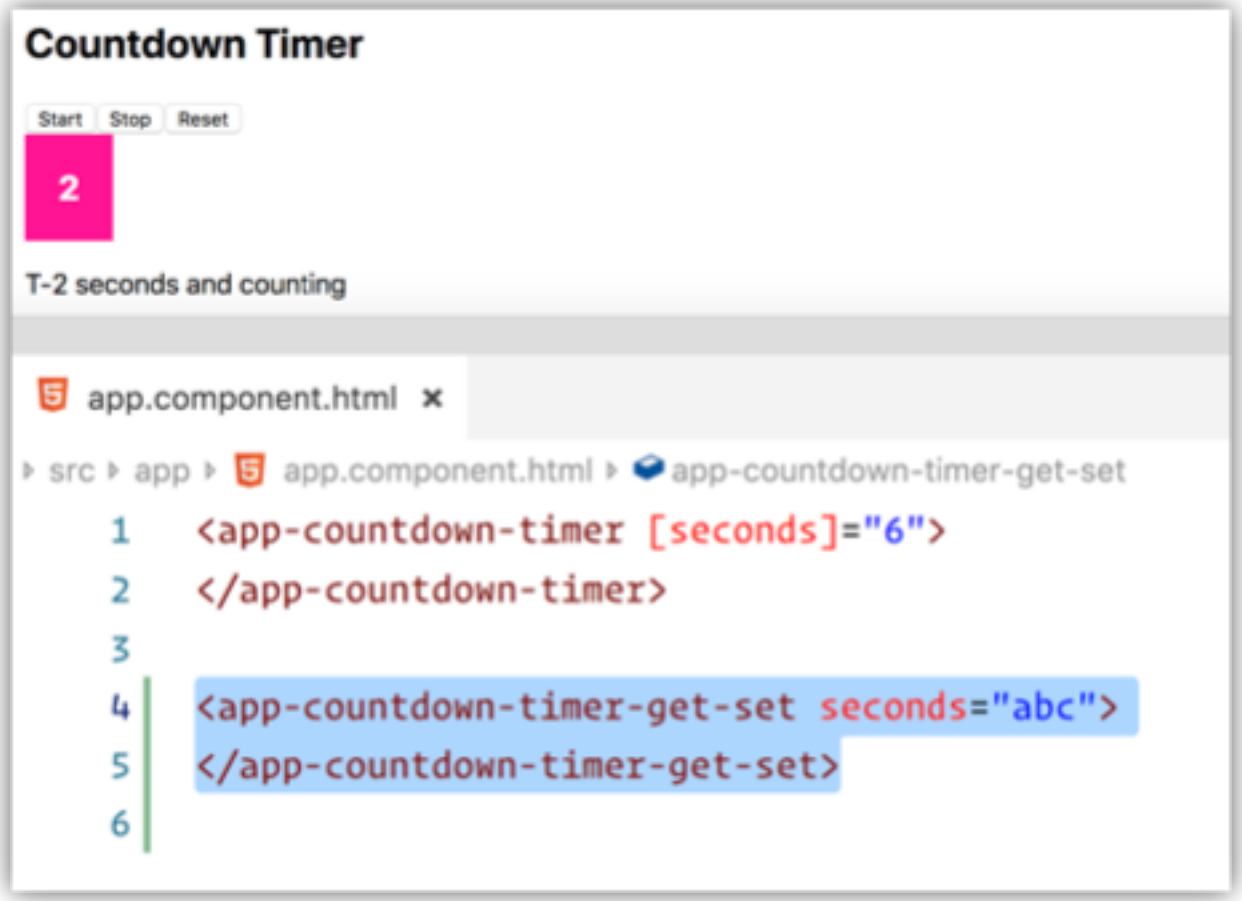
- Generate component mới:
  - `ng generate component countdown-timer-get-set`
- Copy các phần code của countdown-timer component sang component vừa tạo.
- Update logic để sử dụng get/set cho @Input.

Chuẩn hóa dữ liệu đầu vào

```
export class CountdownTimerGetSetComponent {
  private _seconds = 11;

  @Input()
  get seconds(): number {
    return this._seconds;
  }
  set seconds(v) {
    v = typeof v === 'undefined' ? 11 : v;
    const vFixed = Number(v);
    this._seconds = Number.isNaN(vFixed) ? 11 : vFixed;
  }
}
```

Chuẩn hóa dữ liệu đầu vào



Chuẩn hóa dữ liệu đầu vào

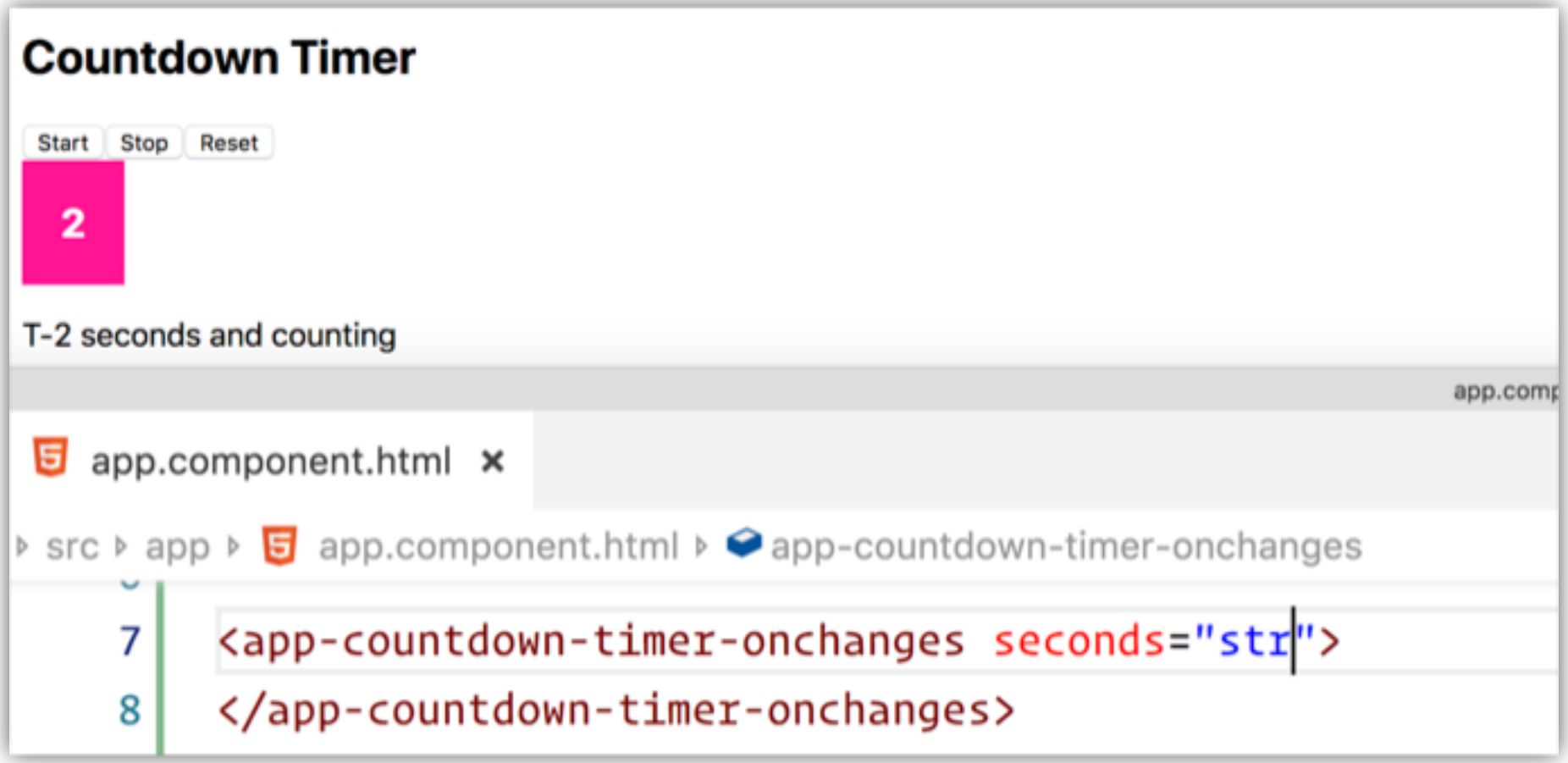
- Generate component mới:
  - `ng generate component countdown-timer-onchanges`
- Copy các phần code của countdown-timer component sang component vừa tạo.
- Update logic để sử dụng lifecycle hook (ngOnChanges) cho @Input.

Chuẩn hóa dữ liệu đầu vào

```
export class CountdownTimerOnchangesComponent implements OnChanges {
  @Input()
  seconds = 11;
  ngOnChanges(changes: SimpleChanges) {
    if ('seconds' in changes) {
      let v = changes.seconds.currentValue;
      v = typeof v === 'undefined' ? 11 : v;
      const vFixed = Number(v);
      this.seconds = Number.isNaN(vFixed) ? 11 : vFixed;
    }
  }
}
```

Angular component lifecycle, hàm này sẽ được Angular tự động gọi mỗi khi có một @Input property bị thay đổi.

Chuẩn hóa dữ liệu đầu vào



Tạo alias cho @Input

- Generate component mới:
  - `ng generate component countdown-timer-alias`
- Copy các phần code của countdown-timer-onchanges component sang component vừa tạo.
- Tạo alias cho @Input.

Tạo alias cho @Input

```
export class CountdownTimerAliasComponent {

  @Input('remaining-time')
  seconds = 11;

}
```

Tên gọi của property **seconds** mà component cha sẽ sử dụng để truyền dữ liệu cho component này

```
<app-countdown-timer-alias remaining-time="60">
</app-countdown-timer-alias>
```

Component Event với @Output

- Component con gửi dữ liệu cho component cha thông qua Event với @Output.
- Sử dụng alias cho @Outputproperty.

Component Event với @Output

- Generate component mới:
  - `ng generate component countdown-timer-event`
- Copy các phần code của countdown-timer-onchanges component sang component vừa tạo.
- Tạo Event cho component vừa tạo để thông báo khi nào quá trình đếm ngược kết thúc.
- Ở component cha thực hiện hiển thị thông báo cho người dùng biết quá trình đếm ngược đã kết thúc.

Component Event với @Output

```
export class CountdownTimerEventComponent {

  @Output()
  finish = new EventEmitter<boolean>();
  private countDown() {
    // other code
    if (this.remainingTime === 0) {
      this.finish.emit(true);
    }
  }
}
```

Component Event với @Output

app.component.html

```
<app-countdown-timer-event
  seconds="10"
  (finish)="finishCountdown()">
</app-countdown-timer-event>
<p>{{ countdownMsg }}</p>
```

app.component.ts

```
countdownMsg = '';
finishCountdown() {
  this.countdownMsg = 'Finished!';
}
```

Component Event với @Output

- Generate component mới:
  - `ng generate component countdown-timer-event-alias`
- Copy các phần code của countdown-timer-event component sang component vừa tạo.
- Tạo Event alias.

Component Event với @Output

```
export class CountdownTimerEventAliasComponent {

  @Output('timerEnd')
  finish = new EventEmitter<boolean>();
}
```

Component Event với @Output

app.component.html

```
<app-countdown-timer-event-alias
   seconds="10"
   (timerEnd)="endCountdown()">
</app-countdown-timer-event-alias>
<p>{{ countdownAliasMsg }}</p>
```

app.component.ts

```
countdownAliasMsg = '';
endCountdown() {
   this.countdownAliasMsg = 'Ended!';
}
```

Raising the bar