ANGULAR 2

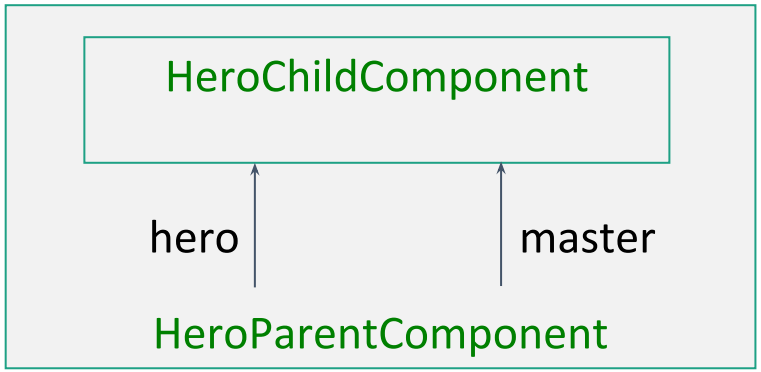
COMPONENTS COMMUNICATION WITH EVENTS

LUXOFT

LXFT LISTED NYSE

PASS DATA FROM PARENT TO CHILD WITH INPUT BINDING

```
@Component({
  selector: 'hero-parent',
  template: `
  <hero-child *ngFor="let hero of heroes"
    [hero]="hero" [master]="master">
  </hero-child>`
})
export class HeroParentComponent {
  heroes = HEROES;
  master: string = 'Master';
}
```




```
@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;
}
```

Master controls 3 heroes

Mr. IQ says:

I, Mr. IQ, am at your service, Master.

Magneta says:

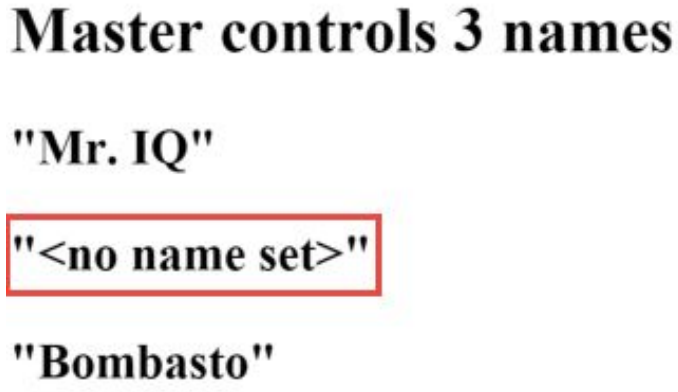
I, Magneta, am at your service, Master.

Bombasto says:

I, Bombasto, am at your service, Master.

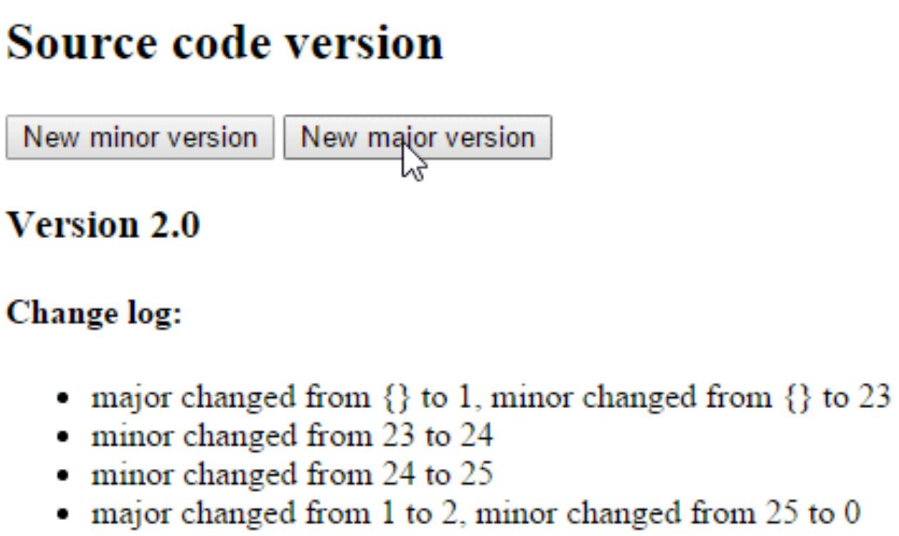
INTERCEPT INPUT PROPERTY CHANGES WITH A SETTER

```
@Component({
  selector: 'name-child',
  template: '<h3>"{{name}}"</h3>'
})
export class NameChildComponent {
  private _name = '';
  @Input()
  set name(name: string) {
    this._name = (name && name.trim()) || '<no name set>';
  }
  get name(): string { return this._name; }
}
```



INTERCEPT INPUT PROPERTY CHANGES WITH NGONCHANGES

```
@Component({
   selector: 'version-parent',
   template: `
   <h2>Source code version</h2>
   <button (click)="newMinor()">
      New minor version
   </button>
   <button (click)="newMajor()">
      New major version
   </button>
   <version-child [major]="major"  [minor]="minor">
   </version-child>` })
export class VersionParentComponent {
   major: number = 1;
   minor: number = 23;
   newMinor() { this.minor++; }
   newMajor() { this.major++;  this.minor = 0; }
}
```



INTERCEPT INPUT PROPERTY CHANGES WITH NGONCHANGES

```
@Component({
  selector: 'version-child',
  template: `
  <h3>Version {{major}}.{{minor}}</h3>
  <h4>Change log:</h4>
  <ul><li *ngFor="let change of changeLog">{{change}}</li></ul>`})
export class VersionChildComponent implements OnChanges {
  @Input() major: number;
  @Input() minor: number;
  changeLog: string[] = [];
  ngOnChanges(changes: {[propKey: string]: SimpleChange}) {
    let log: string[] = [];
    for (let propName in changes) {
      let changedProp = changes[propName];
      let from = JSON.stringify(changedProp.previousValue);
      let to =  JSON.stringify(changedProp.currentValue);
      log.push( `${propName} changed from ${from} to ${to}`);
    }
    this.changeLog.push(log.join(', '));
  }
}
```

Change log:

- major changed from {} to 1, minor changed from {} to 23
- minor changed from 23 to 24
- minor changed from 24 to 25
- major changed from 1 to 2, minor changed from 25 to 0

PARENT LISTENS FOR CHILD EVENT

```
@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;
   }
}
```

PARENT LISTENS FOR CHILD EVENT

```
@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++;
  }
}
```