Learn to Build Awesome[r] Apps with Angular

DAY THREE!

Strong grasp on how to **launch** and **support** an Angular application

Agenda

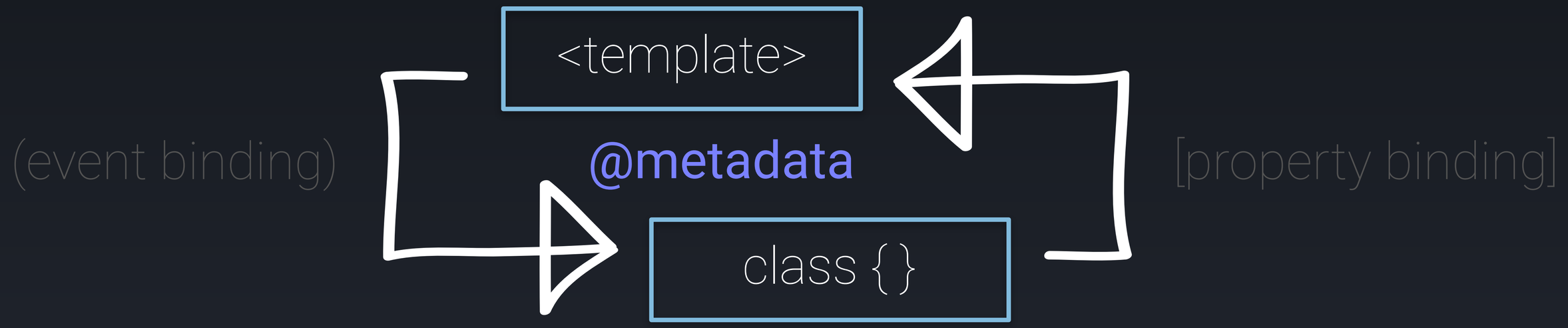
- Review Challenge
- Application Complexity
- Immutable Operations
- Reactive Forms
- Event Communication
- Route Parameters
- BONUS! Angular and Firebase

Agenda

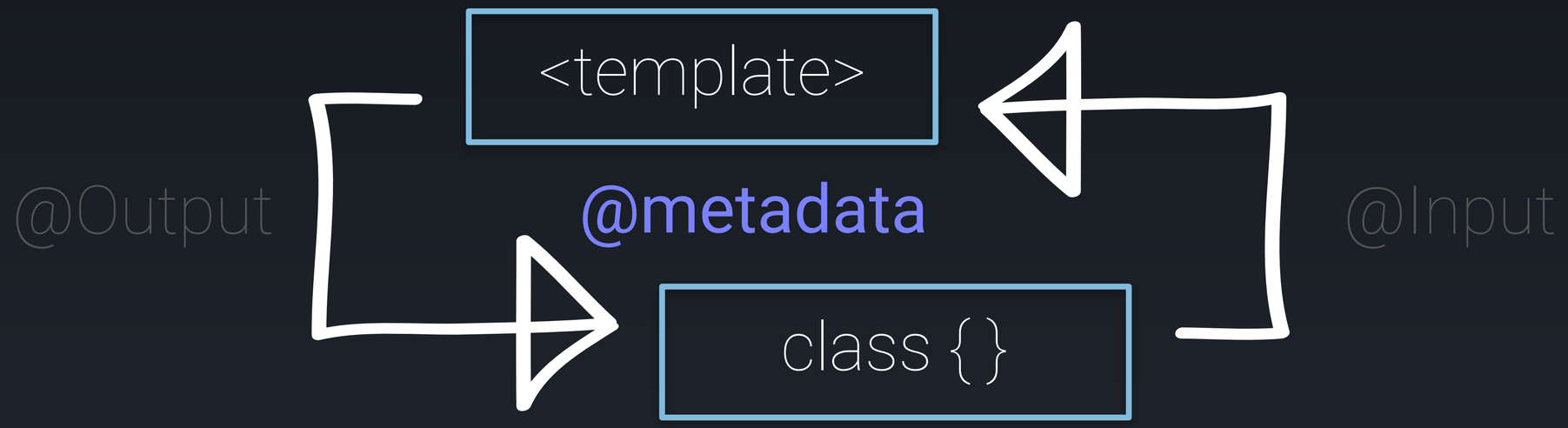
- **Make It Work:** Testing and Debugging
- **Make It Right:** Static Analysis
- **Make It Fast:** Angular Performance
- **Make It Live:** Deploying

REVIEW Time!

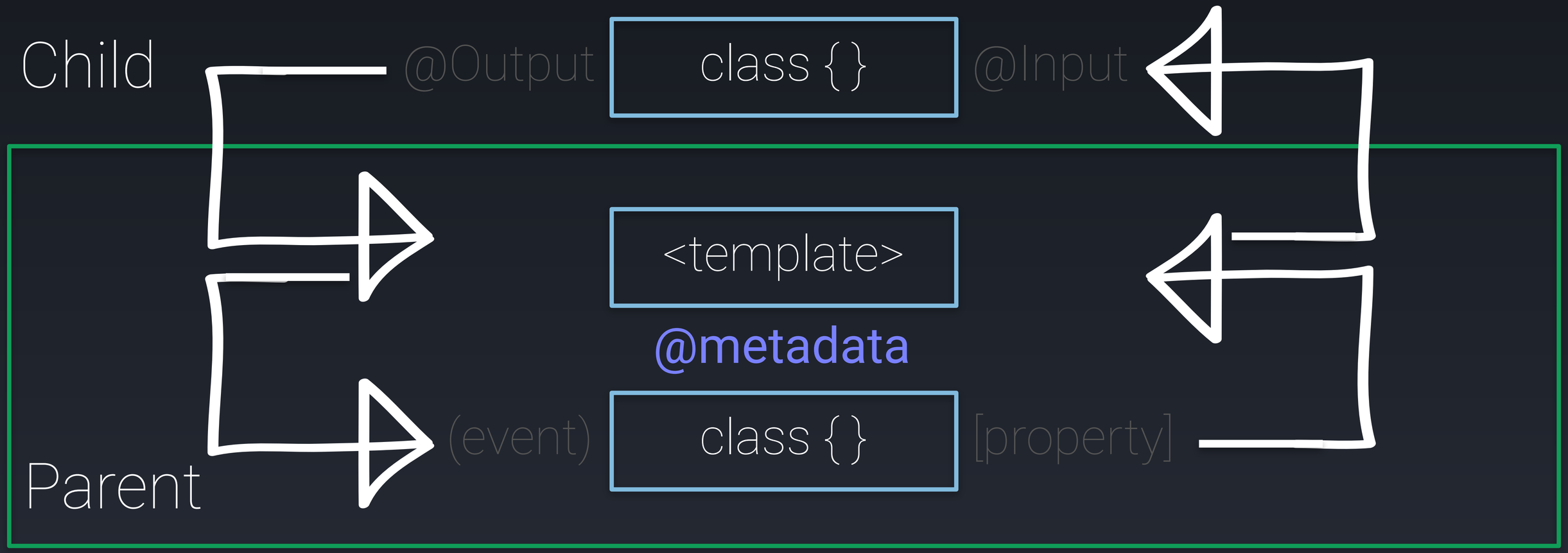
What kind of binding goes on each side?



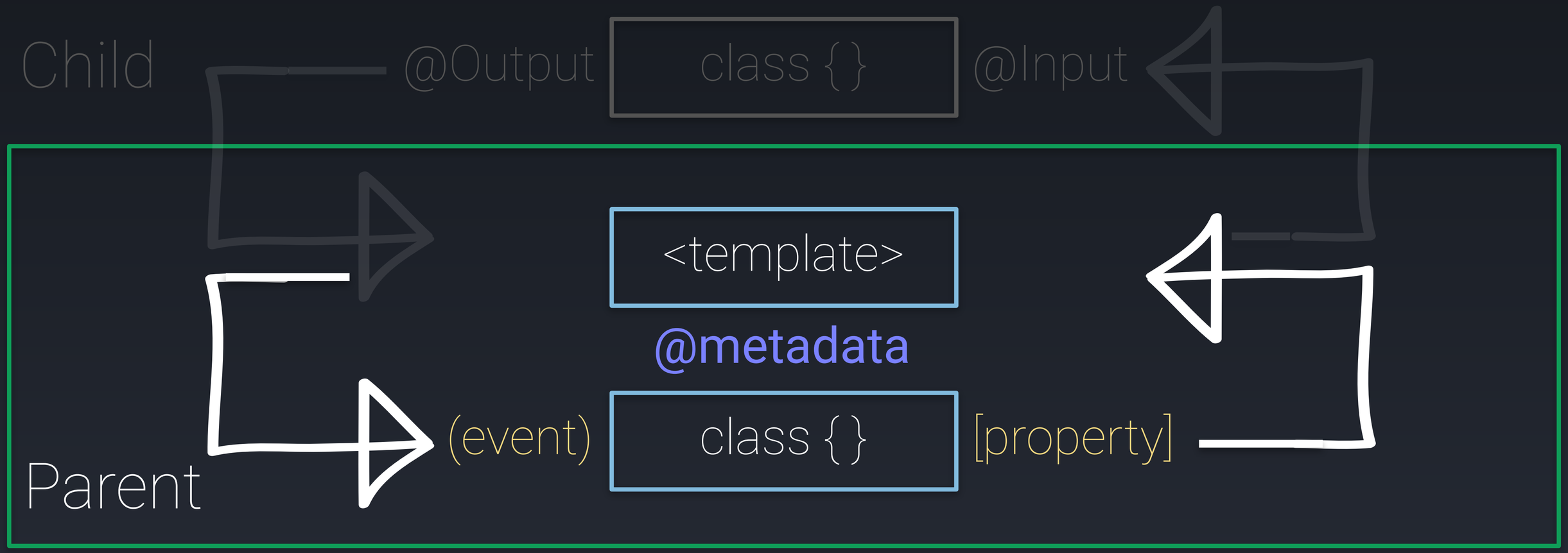
How do we define custom bindings?



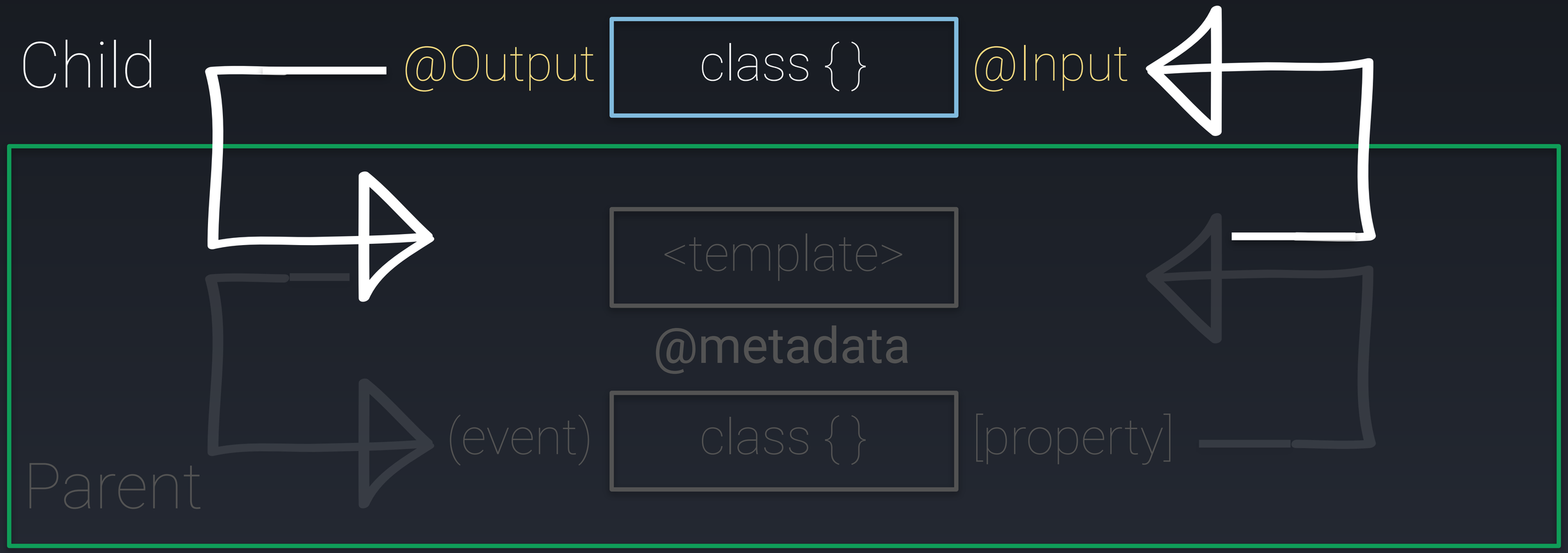
What are the missing bindings?



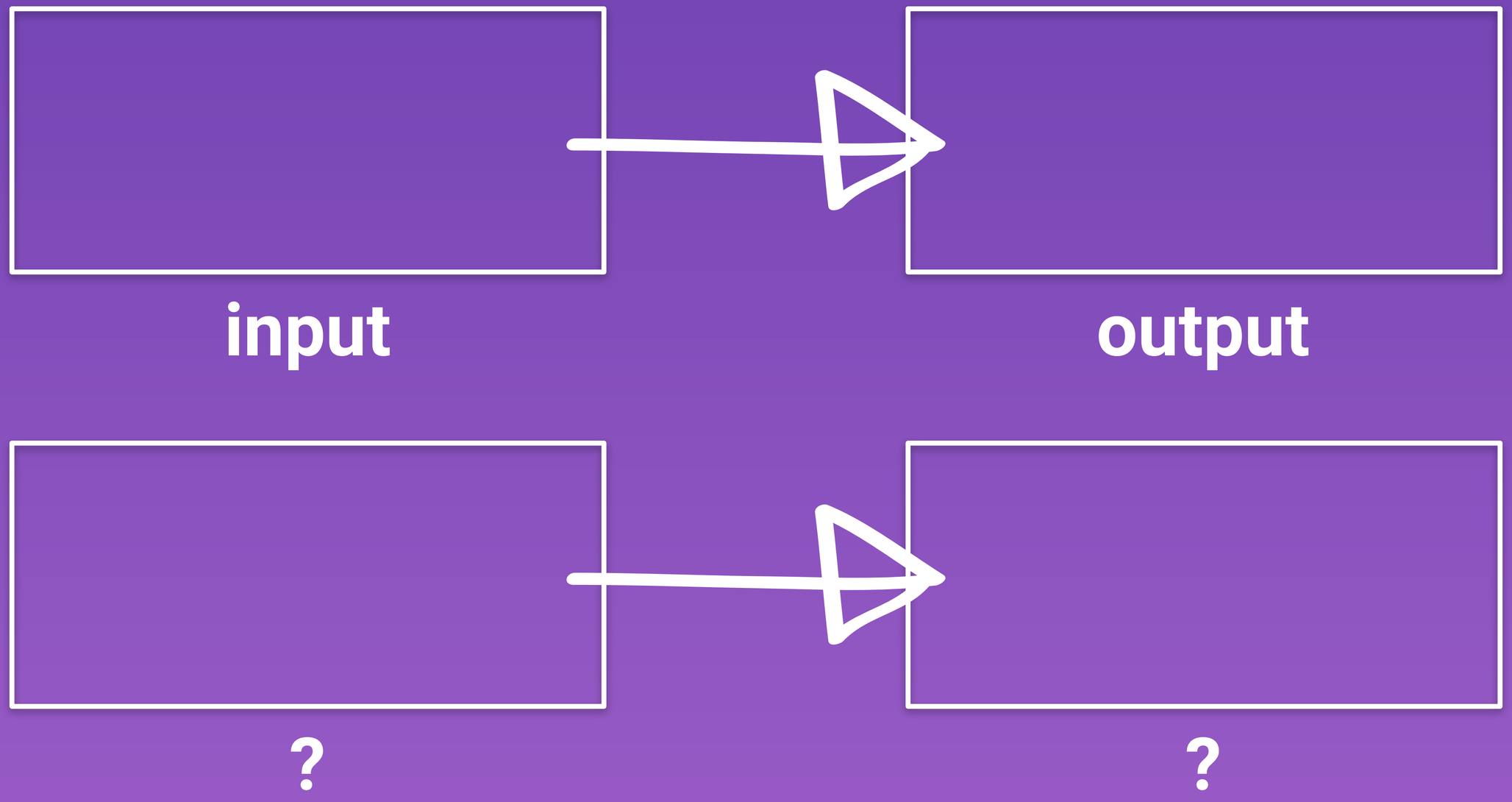
Parent and Child



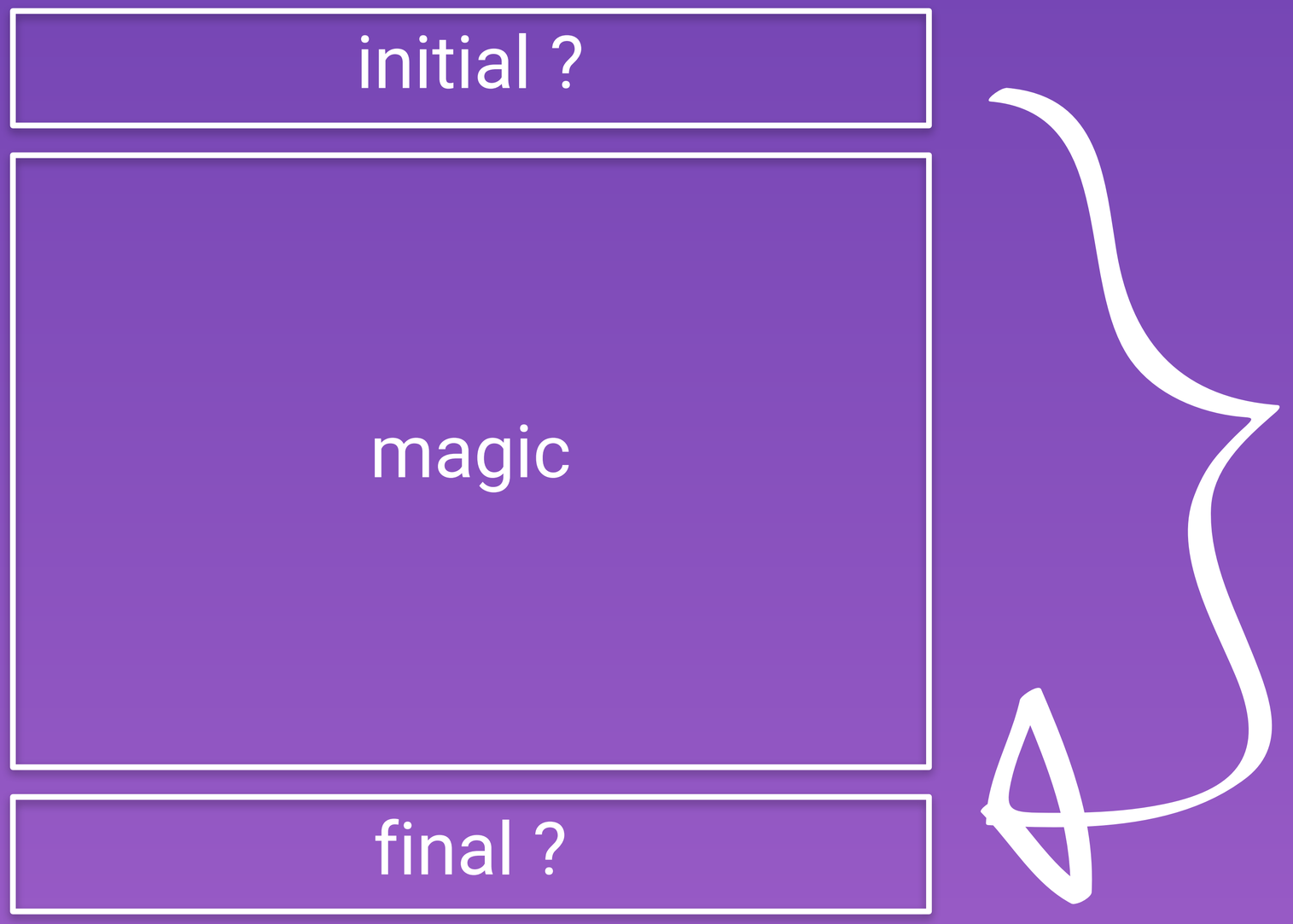
Parent and **Child**



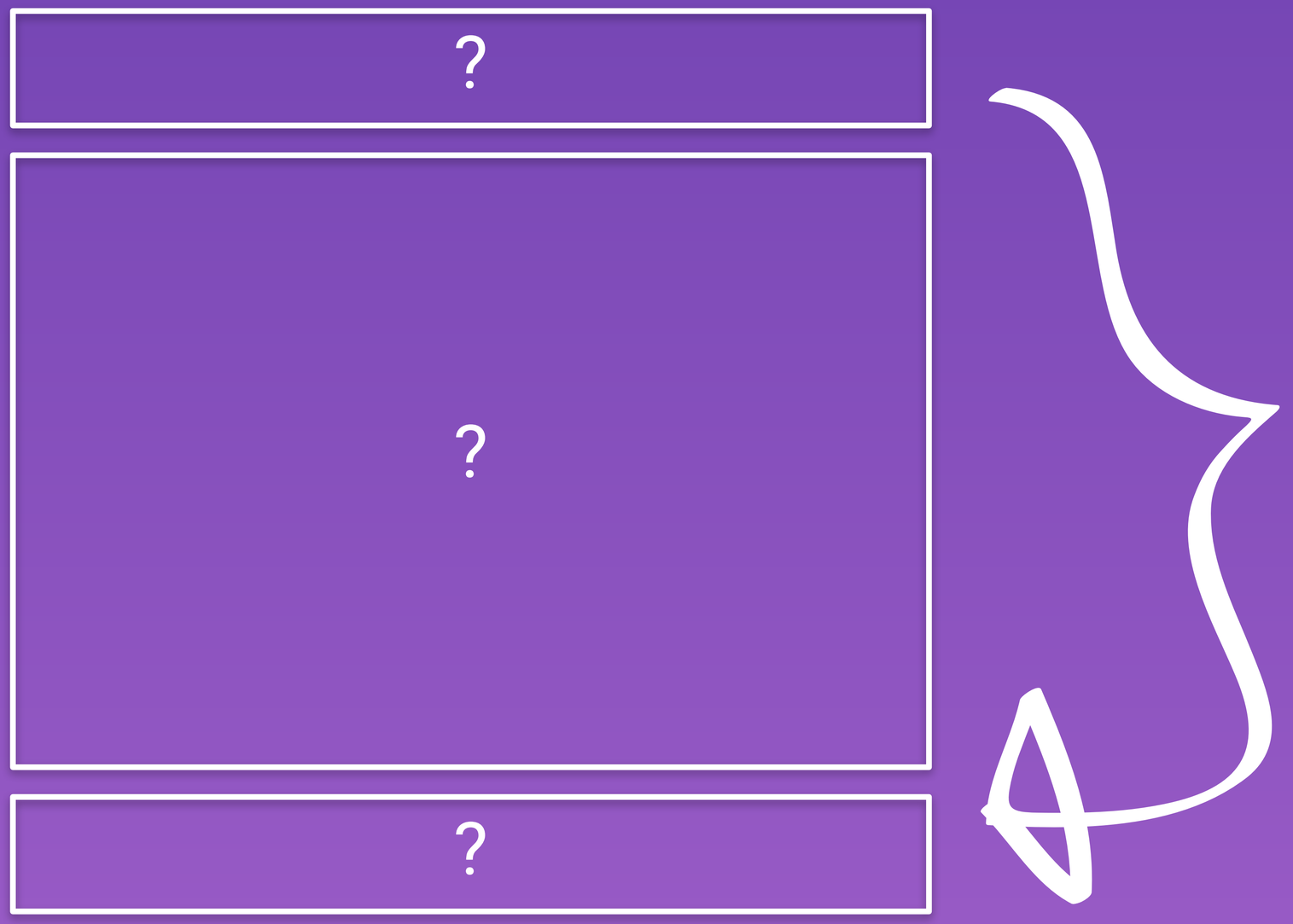
How does observables change this diagram?



What is missing and how does the diagram work?



How does this look in real life?



Where are the pieces of the observable sequence?

```
ngOnInit() {
  const search$ = Observable.fromEvent(this.getNativeElement(this.itemsSearch), 'keyup')
    .debounceTime(200)
    .distinctUntilChanged()
    .map((event: any) => event.target.value)
    .switchMap(term => this.itemsService.search(term))
    .subscribe(items => this.onResults.emit(items));
}
```

Challenges

- Create a **status** component and place it in the **home** component
- Create a **currentStatus input**
- Create a **logout output**
- Implement the necessary UI elements to facilitate communication between the **home** component and the **status** component

Application **Complexity**

The biggest problem in the development and maintenance of large-scale software systems is complexity — large systems are hard to understand.

Out of the Tarpit - Ben Mosely Peter Marks

We believe that the major contributor to this complexity in many systems is the handling of state and the burden that this adds when trying to analyse and reason about the system. Other closely related contributors are code volume, and explicit concern with the flow of control through the system.

Out of the Tarpit - Ben Mosely Peter Marks

Complexity and **purgatory**

```
class ItemsComponent {
  total: number = 0;
  currentCategory: string = 'cool';
  inbound(item) {
    const newTotal: number;
    switch(this.currentCategory) {
      case 'fun':
        // calculate total based on fun factor
        break;
      case 'cool':
        // calculate total based on cool factor
        break;
      case 'dangerous':
        // calculate total based on dangerous factor
        break;
      default:
        // do nothing at all
    }
    return newTotal;
  }
}
```

```
class ItemsComponent {
  total: number = 0;
  currentCategory: string = 'cool';
  inbound(item) {
    const newTotal: number;
    switch(this.currentCategory) {
      case 'fun':
        // calculate total based on fun factor
        break;
      case 'cool':
        // calculate total based on cool factor
        break;
      case 'dangerous':
        // calculate total based on dangerous factor
        break;
      default:
        // do nothing at all
    }
    return newTotal;
  }
}
```

```
const itemsComponents = new ItemsComponent();
const myItem = {name:'My Item'};
itemsComponents.inbound(myItem); // Some result

itemsComponents.currentCategory = 'fun'; // Changing state
itemsComponents.inbound(myItem); // Same parameter but different result

itemsComponents.currentCategory = 'cool'; // Changing state
itemsComponents.inbound(myItem); // Same parameter but different result

itemsComponents.currentCategory = 'dangerous'; // Changing state
itemsComponents.inbound(myItem); // Same parameter but different result
```

```
class ItemsComponent {
  total: number = 0;
  currentCategory: string = 'cool';
  currentAgeGroup: string = 'child';
  inbound(item) {
    const newTotal: number;
    switch(this.currentCategory) {
      //...
      case 'dangerous':
        if(this.currentAgeGroup !== 'child') {
          // calculate total based on dangerous factor
          this.currentCategory = 'dangerous';
        } else {
          // calculate total based on alternate dangerous factor
        }
        break;
      default:
      // do nothing at all
    }
    return newTotal;
  }
}
```

State **management**

```
class Inventory {
  ledger = { total: 1200 };
}
class ItemsComponent {
  ledger: any;
  constructor(private inventory:Inventory) {
    this.ledger = inventory.ledger;
  }
  add(x) { this.ledger.total += x; }
}
class WidgetsComponent {
  ledger: any;
  constructor(private inventory:Inventory) {
    this.ledger = inventory.ledger;
  }
  add(x) { this.ledger.total += x; }
}
```

```
class Inventory {
  ledger = { total: 1200 };
}
class ItemsComponent {
  ledger: any;
  constructor(private inventory:Inventory) {
    this.ledger = inventory.ledger;
  }
  add(x) { this.ledger.total += x; }
}
class WidgetsComponent {
  ledger: any;
  constructor(private inventory:Inventory) {
    this.ledger = inventory.ledger;
  }
  add(x) { this.ledger.total += x; }
}
```

Controlling flow

```
function doWork() {
  return $http.post('url')
    .then(function(response){
      if(response.data.success)
        return response.data;
      else
        return $q.reject('some error occurred');
    })
}
doWork().then(console.log, console.error);
```

```
var retriesCount = 0;
function doWork() {
  return $http.post('url')
    .then(function(response){
      if(response.data.success)
        return response.data;
      else
        return $q.reject('some error occurred');
    })
    .then(null, function(reason){
      if(retriesCount++ < 3)
        return doWork();
      else
        return $q.reject(reason);
    });
}
doWork().then(console.log, console.error);
```

Code **volume**

Immutable Operations

Immutable Operations

- Object.freeze
- Immutable Add
- Immutable Update
- Immutable Delete

```
case CREATE_WIDGET:
  state.push(action.payload);
  return state;
```

Mutable!

```
case UPDATE_WIDGET:
  state.forEach((widget, index) => {
    if (widget[comparator] === action.payload[comparator]) {
      state.splice(index, 1, action.payload);
    }
  });
  return state;
```

Mutable!

```
case DELETE_WIDGET:
  state.forEach((widget, index) => {
    if (widget[comparator] === action.payload[comparator]) {
      state.splice(index, 1);
    }
  });
  return state;
```

Mutable!

```
case CREATE_WIDGET:
  Object.freeze(state);
  state.push(action.payload);
  return state;
```

The Object.freeze() method freezes an object: that is, prevents new properties from being added to it; prevents existing properties from being removed; and prevents existing properties, or their enumerability, configurability, or writability, from being changed. **In essence the object is made effectively immutable. The method returns the object being frozen.**

Object.freeze

```
case CREATE_ITEM:
  return [...state, action.payload];

case CREATE_ITEM:
  return state.concat(action.payload);
```

The concat() method is used to merge two or more arrays. This method does not change the existing arrays, but instead returns a new array. - MDN

Immutable!

```
case UPDATE_ITEM:
  return state.map(item => {
    return item[comparator] === action.payload[comparator]
      ? Object.assign({}, item, action.payload) : item;
  });
```

The map() method creates a new array with the results of calling a provided function on every element in this array.

The Object.assign() method is used to copy the values of all enumerable own properties from one or more source objects to a target object. It will return the target object.

Immutable!

```
case DELETE_ITEM:
  return state.filter(item => {
    return item[comparator] !== action.payload[comparator];
  });
```

The filter() method creates a new array with all elements that pass the test implemented by the provided function.

Immutable!

Reactive **Forms**

Pre Challenge

- Create an **Newsletter** component
  `ng g c newsletter -m app.module.ts`
- Add the **Newsletter** component to the **HomeComponent** template
- Go to **http://bit.ly/newsletter-snippets** and copy the snippets to the component

Reactive Forms

- **FormControl** tracks the value and validation status of an individual form control
- **FormGroup** tracks the value and validity state of a group of FormControl instances.
- **formGroup** directive binds an existing FormGroup to a DOM element.
- **formControlName** directive syncs a FormControl in an existing FormGroup to a form control element by name.
- **FormBuilder** is essentially syntactic sugar that shortens the new FormGroup(), new FormControl(), and new FormArray() boilerplate that can build up in larger forms.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
```

ReactiveFormsModule

```
form = new FormGroup({
  first: new FormControl('Nancy', Validators.minLength(2)),
  last: new FormControl('Drew'),
});
```

FormGroup

```
<form [formGroup]="form" (ngSubmit)="onSubmit()">
  <input formControlName="first" placeholder="First name">
  <input formControlName="last" placeholder="Last name">

  <button type="submit">Submit</button>
</form>
```

FormGroup Directive

```
export class SimpleFormGroup {
  form = new FormGroup({
    first: new FormControl('Nancy', Validators.minLength(2)),
    last: new FormControl('Drew'),
  });

  get first(): any { return this.form.get('first'); }

  onSubmit(): void {
    console.log(this.form.value);  // {first: 'Nancy', last: 'Drew'}
  }

  setValue() { this.form.setValue({first: 'Carson', last: 'Drew'}); }
}
```

Simple’ish Version

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-newsletter',
  templateUrl: './newsletter.component.html',
  styleUrls: ['./newsletter.component.css']
})
export class NewsletterComponent implements OnInit {
  subscriber: FormGroup;

  constructor(private fb: FormBuilder) { }
}
```

FormBuilder

```
subscriber: FormGroup;

constructor(private fb: FormBuilder) { }

ngOnInit() {
  this.subscriber = this.fb.group({
    name: ['', Validators.required],
    email: ['', Validators.required]
  });
}
```

FormBuilder

```
<form novalidate [formGroup]="subscriber" (submit)="subscribe(subscriber)">
  <md-card-content>
    <md-input-container class="full-width">
      <input type="text"  mdInput placeholder="Name" formControlName="name">
    </md-input-container>
    <md-input-container class="full-width">
      <input type="text" mdInput placeholder="Email" formControlName="email">
    </md-input-container>
  </md-card-content>
  <md-card-actions>
    <button type="submit" md-button>Subscribe!</button>
    <button type="button" md-button (click)="reset()">Cancel</button>
  </md-card-actions>
</form>
```

```
<form novalidate [formGroup]="subscriber" (submit)="subscribe(subscriber)">
  <md-card-content>
    <md-input-container class="full-width">
      <input type="text"  mdInput placeholder="Name" formControlName="name">
    </md-input-container>
    <md-input-container class="full-width">
      <input type="text" mdInput placeholder="Email" formControlName="email">
    </md-input-container>
  </md-card-content>
  <md-card-actions>
    <button type="submit" md-button>Subscribe!</button>
    <button type="button" md-button (click)="reset()">Cancel</button>
  </md-card-actions>
</form>
```

formGroup Directive

```
<form novalidate [formGroup]="subscriber" (submit)="subscribe(subscriber)">
  <md-card-content>
    <md-input-container class="full-width">
      <input type="text"  mdInput placeholder="Name" formControlName="name">
    </md-input-container>
    <md-input-container class="full-width">
      <input type="text" mdInput placeholder="Email" formControlName="email">
    </md-input-container>
  </md-card-content>
  <md-card-actions>
    <button type="submit" md-button>Subscribe!</button>
    <button type="button" md-button (click)="reset()">Cancel</button>
  </md-card-actions>
</form>
```

formControlName Directive

```
subscribe({ value, valid }: { value: Subscriber, valid: boolean }) {
  console.log(value, valid);
  this.reset();
}

reset() {
  this.subscriber.reset({
    name: '',
    email: ''
  });
}
```

Demonstration

Challenges

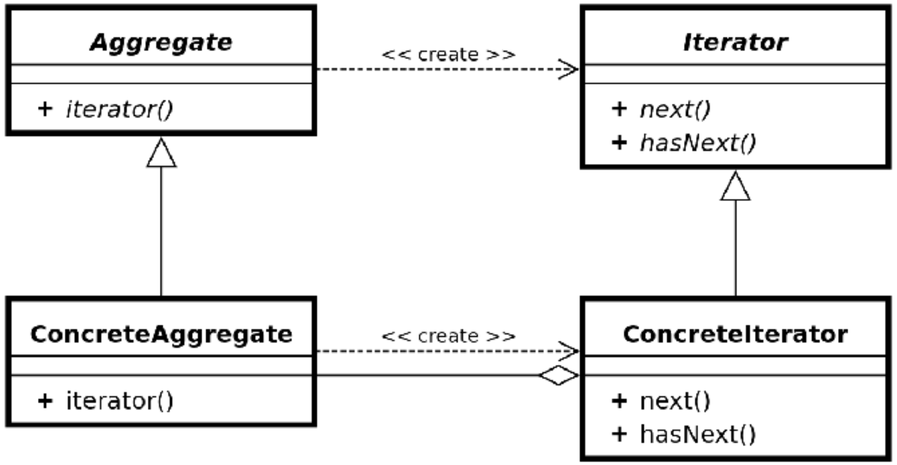
- Create a **subscriber FormGroup** using **FormBuilder**
- Connect **subscriber** to the template form using the **formGroup** and **formControlName** directives
- Submit **subscriber** and log its **value** to the console and if it is **valid**
- Reset the **subscriber**

Event Communication

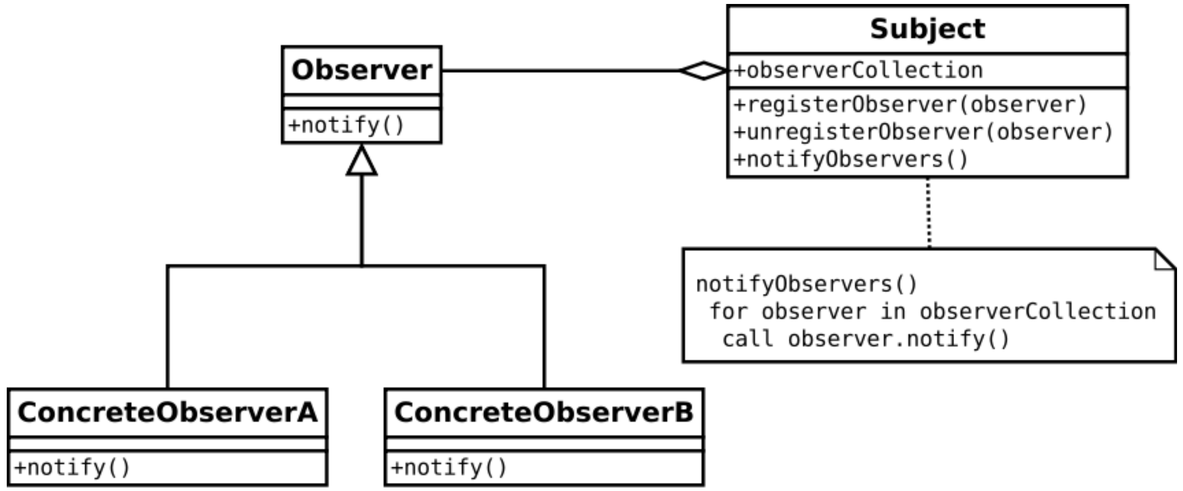
Communicate
state over **time**

http://reactivex.io/rxjs/

Iterator Pattern



Observer Pattern



State

Communication

```
export class NotificationsService {
  private subject = new Subject();
  notifications$ = this.subject.asObservable();

  emit(notification) {
    this.subject.next(notification);
  }
}
```

```
export class AppComponent implements OnInit {

  constructor(private snackbar: MdSnackBar,
              private ns: NotificationsService) {}

  ngOnInit() {
    this.ns.notifications$
      .subscribe(notification => this.showNotification(notification));
  }

  showNotification(notification) {
    this.snackbar.open(notification, 'OK', {
      duration: 3000
    });
  }
}
```

Route **Params**

```
const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'items', component: ItemsComponent},
  {path: 'item/:id', component: ItemComponent},
  {path: 'widgets', component: WidgetsComponent},
  {path: '**', redirectTo: '', pathMatch: 'full'}
];
```

```
export class ItemComponent implements OnInit {
  item: Item;

  constructor(
    private itemsService: ItemsService,
    private route: ActivatedRoute
  ) {}

  ngOnInit() {
    this.route.paramMap
      .switchMap((params: ParamMap) => this.itemsService.load(+params.get('id')))
      .subscribe(item => this.item = item);
  }
}
```

Make It Right

Static Analysis

```
ng lint
```

Linting

1. lukas@Lukass-MacBook-Pro: ~/Projects/angular-rest-app (zsh)

```
➜  ~ projects
➜  Projects angular-rest-app
➜  angular-rest-app git:(master) ✗ clear
➜  angular-rest-app git:(master) ✗ ng lint
Warning: The 'no-use-before-declare' rule requires type information.

ERROR: src/app/app.component.ts[26, 5]: Forbidden 'var' keyword, use 'let' or 'const' instead
ERROR: src/app/app.component.ts[26, 9]: Identifier 'foo' is never reassigned; use 'const' instead of 'var'.
ERROR: src/app/app.component.ts[25, 3]: Implement lifecycle hook interface OnInit for method ngOnInit in class AppCo
mponent (https://angular.io/styleguide#style-09-01)

Lint errors found in the listed files.
➜  angular-rest-app git:(master) ✗
```

```
ng test -cc
```

Code Coverage

Code coverage report for All fil

localhost:63342/angular-rest-app/coverage/index.html?_ijt=b3a3u8u33g687gne0hntsm01bm

All files

72.28% Statements 266/368 **61.43%** Branches 43/70 **40%** Functions 46/115 **71.57%** Lines 219/306

File		Statements		Branches		Functions		Lines	
src		100%	29/29	100%	0/0	100%	1/1	100%	29/29
src/app		100%	28/28	80%	8/10	100%	8/8	100%	23/23
src/app/home		50%	11/22	75%	6/8	12.5%	1/8	50%	9/18
src/app/items		40.91%	18/44	60%	6/10	13.33%	2/15	40.54%	15/37
src/app/items/item		100%	13/13	75%	6/8	60%	3/5	100%	11/11
src/app/items/item-detail		91.67%	11/12	40%	2/5	100%	3/3	100%	9/9
src/app/items/items-list		73.68%	14/19	0%	0/2	33.33%	2/6	75%	12/16
src/app/items/items-search		86.96%	20/23	66.67%	2/3	57.14%	4/7	85%	17/20
src/app/newsletter		72.22%	13/18	75%	6/8	60%	3/5	78.57%	11/14
src/app/shared		66.22%	49/74	66.67%	4/6	20.69%	6/29	60.32%	38/63
src/app/user		100%	17/17	100%	0/0	75%	3/4	100%	11/11
src/app/user/profile		100%	6/6	100%	0/0	100%	3/3	100%	4/4
src/app/widgets		41.03%	16/39	40%	2/5	13.33%	2/15	42.42%	14/33
src/app/widgets/widget-detail		75%	9/12	20%	1/5	66.67%	2/3	77.78%	7/9
src/app/widgets/widgets-list		100%	9/9	100%	0/0	100%	2/2	100%	7/7
src/testing		100%	3/3	100%	0/0	100%	1/1	100%	2/2

Code coverage generated by istanbul at Fri Sep 22 2017 07:23:33 GMT-0500 (CDT)

```
npm install -g source-map-explorer
ng build
source-map-explorer ./dist/vendoer.*
```

Source Map Explorer

```
ng build -prod -sm
source-map-explorer ./dist/vendoer.*
```

Source Map Explorer

./dist/vendor.f2181c96361a4

file:///var/folders/nd/cjsjdyy938976mh3y91sx41m0000gn/T/117822-29078-1nvffsj.slf6.html

/ • 1.25 MB • 100.0%
Users • 1.25 MB • 100.0%
lukas • 1.25 MB • 100.0%
Projects • 1.25 MB • 100.0%
angular-rest-app • 1.25 MB • 100.0%
node_modules • 1.03 MB • 82.4%
@angular • 884.72 KB • 70.7%

material • 392.13 KB • 31.3%
@angular • 392.13 KB • 31.3%
material.es5.js • 392.13 KB • 31.3%

core • 119.24 KB • 9.5%
@angular • 119.24 KB • 9.5%
core.es5.js • 119.24 KB • 9.5%

router • 71.34 KB • 5.7%
@angular • 71.34 KB • 5.7%
router.es5.js • 71.34 KB • 5.7%

animations • 66.95 KB • 5.3%
@angular • 66.95 KB • 5.3%
animations • 62.17 KB •
browser.es5.js • 62.17 KB • 5.0%

forms • 66.6 KB • 5.3%
@angular • 66.6 KB • 5.3%
forms.es5.js • 66.6 KB • 5.3%

cdk • 64.25 KB • 5.1%
@angular • 64.25 KB • 5.1%
cdk • 64.25 KB • 5.1%
overlay.es5.js • 25.39 KB • 2.0%
a11y.es5.js • 12.29 KB • 1.0%
table.es5.js • 9.06 KB • 0.7%
portal.es5.js • 5.83 KB • 0.5%
scroll • 4.11 KB • 0.3%

platform-browser • 49.52 KB • 4.0%
@angular • 49.52 KB • 4.0%
platform-browser.es5.js • 41.92 KB • 3.3%
platform-browser • 7.6 KB • 0.6%

common • 35.09 KB • 2.8%
@angular • 35.09 KB • 2.8%
common.es5.js • 35.09 KB • 2.8%

http • 19.61 KB • 1.6%
@angular • 19.61 KB •
http.es5.js • 19.61 KB • 1.6%

rxjs • 55.67 KB • 4.4%
operator • 20.66 KB • 1.7%
observable • 14.83 KB • 1.2%
util • 3.62 KB • 0.3%
Subscr • 3.24 KB • 0.3%
sched • 2.55 KB • 0.2%
Subje • 2.4 KB • 0.2%
Sub • 1.8 KB •

core-js • 53.3 KB • 4.3%
modules • 51.44 KB • 4.1%
es6.symbol.js
es6 • 1. KB • 0.1

zone.js • 37.91 KB • 3.0%
dist • 37.91 KB • 3.0%
zone.js • 37.91 KB • 3.0%

src • 220.13 KB • 17.6%
$_gendir • 220.13 KB • 17.6%
node_modules • 220.13 KB • 17.6%
@angular • 220.13 KB • 17.6%
material • 220.13 KB • 17.6%

Make It Fast

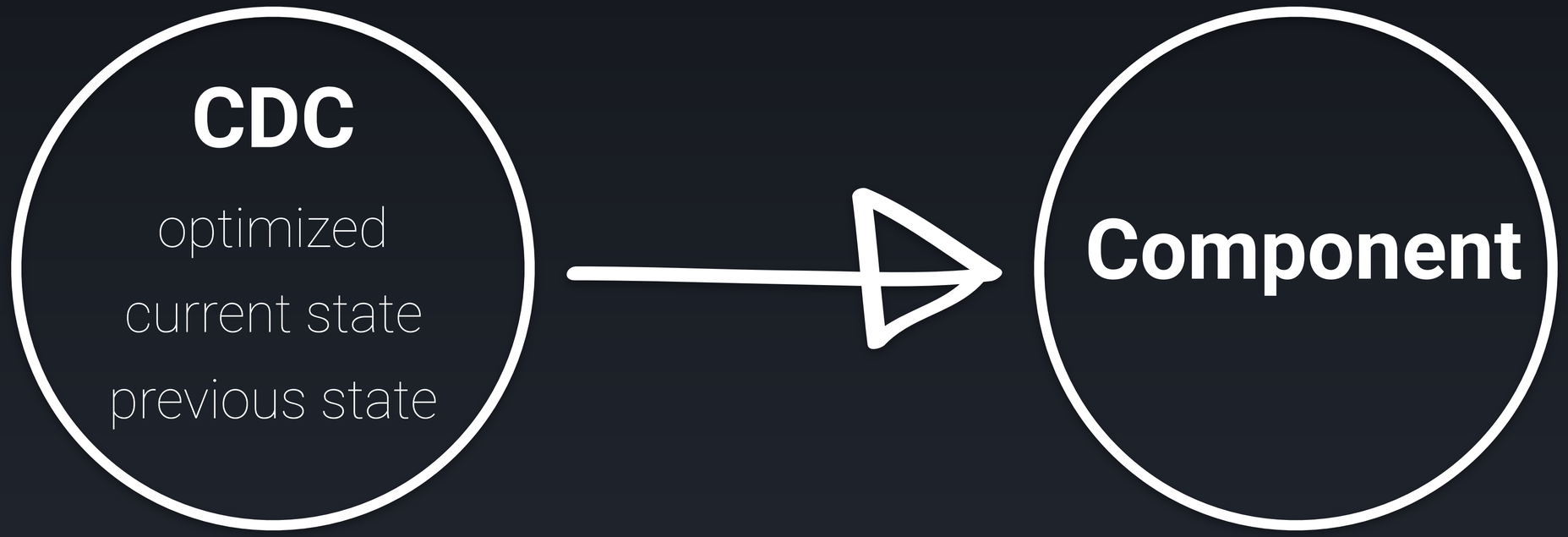
Performance

Performance

- Change Detection
- Lazy Loaded Modules

Change Detection

Zone.js

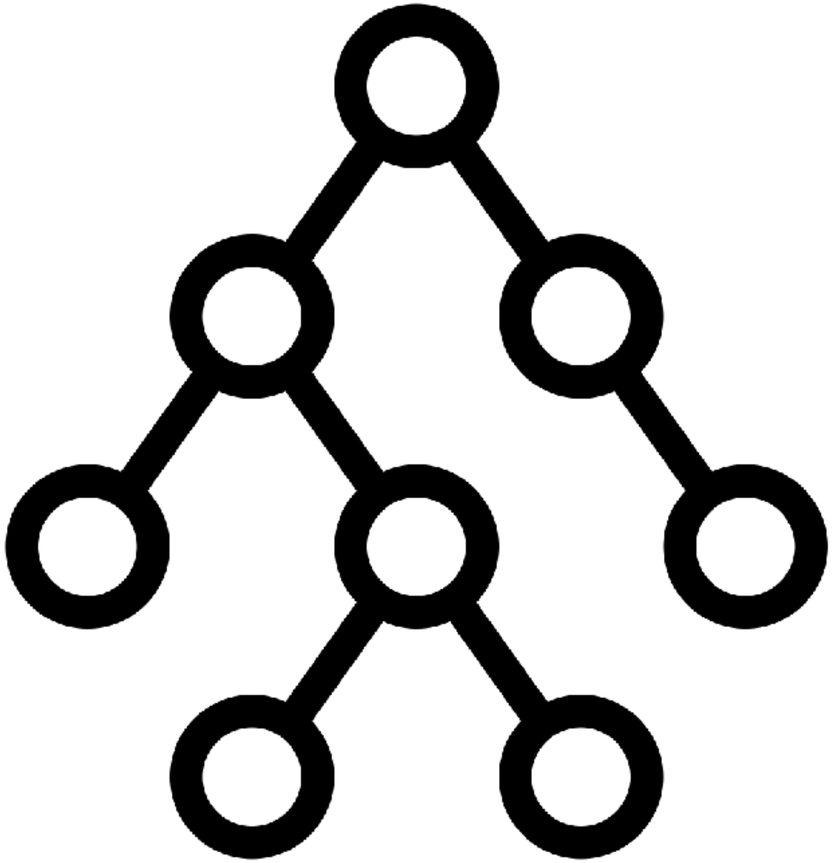


Change Detection Classes

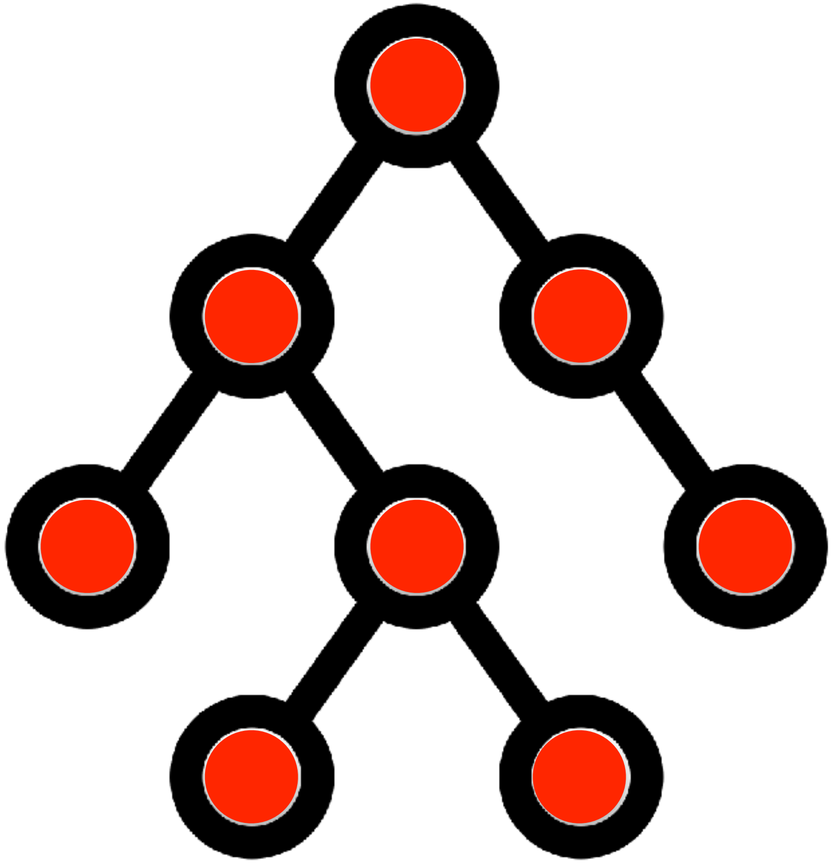
3 -10x Faster

Change Detection Strategy

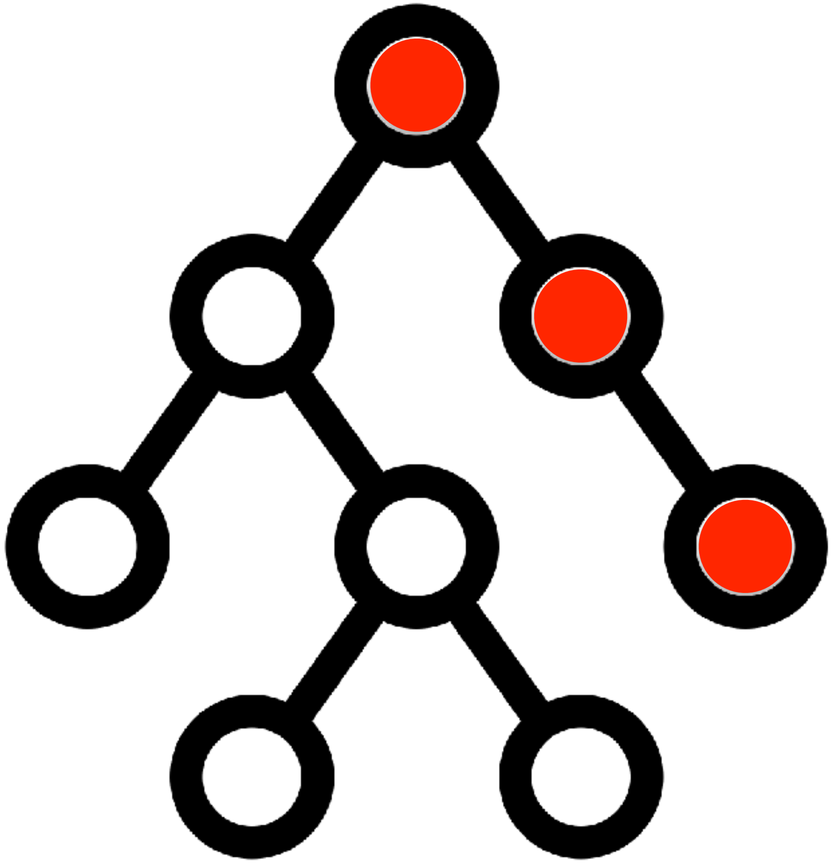
- We can control how Angular will respond when it detects changes within the application
- By default, Angular will always detect changes and respond on all nodes
- We can set the change detection strategy to **onPush** meaning that Angular will only check for changes once
- We are essentially turning off change detection for a component branch in our application which has serious performance implications



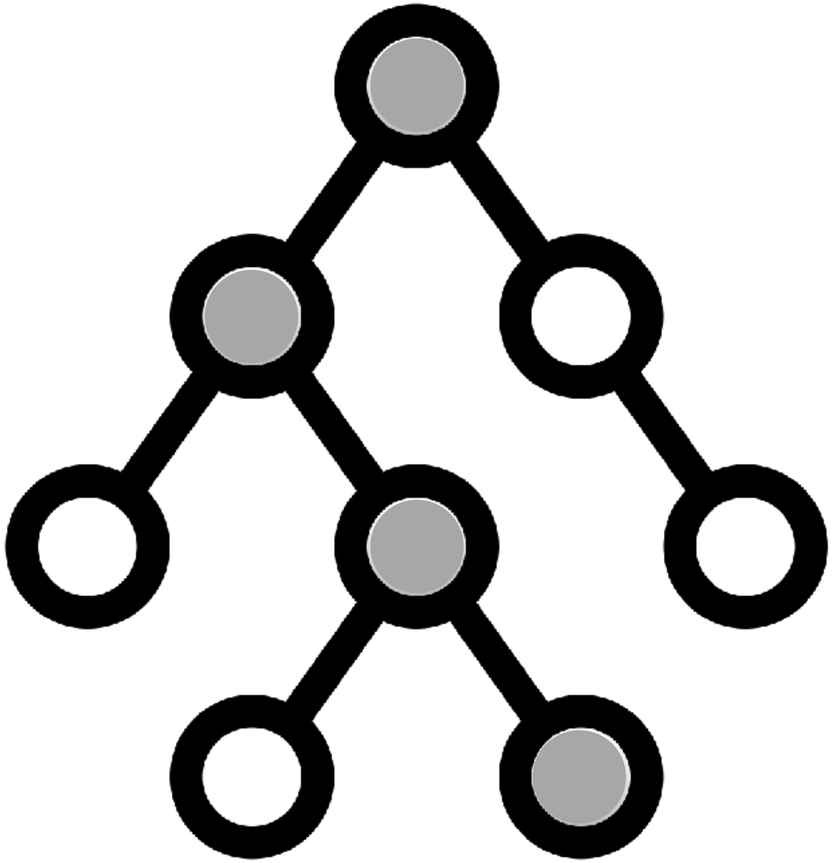
Detecting Change



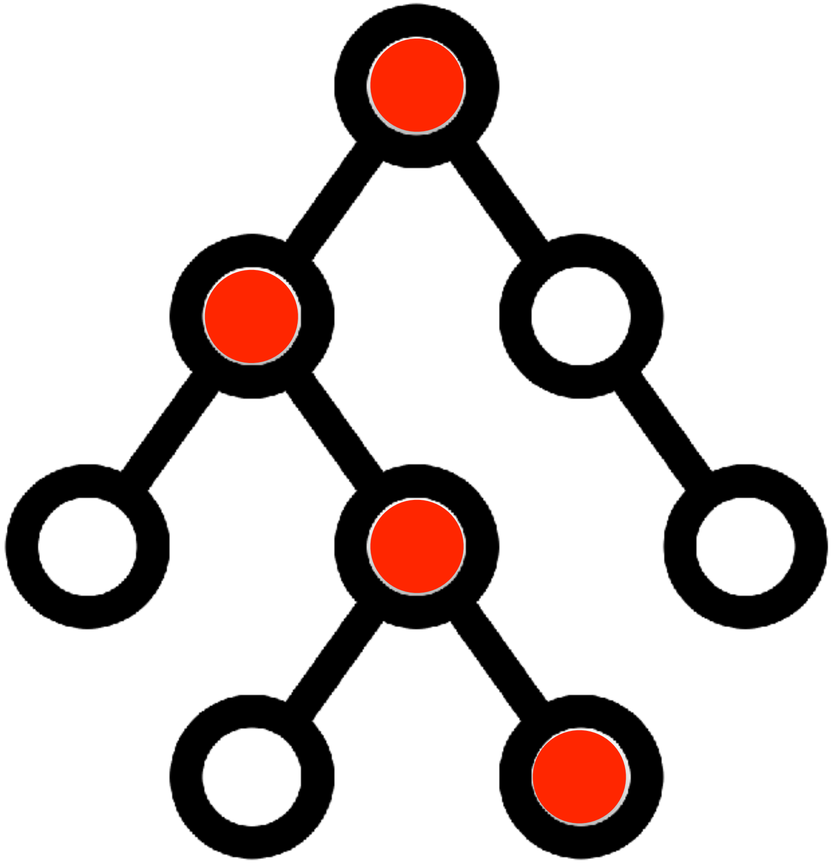
Default Change Detection



OnPush Change Detection



Observables



Observables

```
@Component({
  selector: 'app-widgets',
  templateUrl: './widgets.component.html',
  styleUrls: ['./widgets.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class WidgetsComponent implements OnInit { }
```

ChangeDetectionStrategy.onPush

```
Observable.fromEvent(document, 'click')
  .map(event => 100)
  .startWith(5)
  .subscribe(coolness => {
    this.coolness = coolness;
  });
```

Will Not Update

```
constructor(private cd: ChangeDetectorRef) {}

ngOnInit() {
  Observable.fromEvent(document, 'click')
    .map(event => 100)
    .startWith(5)
    .subscribe(coolness => {
      this.coolness = coolness;
      this.cd.detectChanges();
    });
}
```

Will Update

Lazy Loaded Modules

```
const routes: Routes = [
  { path: '', component: ProfileComponent }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class UserRoutingModule { }
```

Child Module

```
const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'items', component: ItemsComponent},
  {path: 'widgets', component: WidgetsComponent},
  {path: 'profile', loadChildren: './user/user.module#UserModule'},
  {path: '**', redirectTo: '', pathMatch: 'full'}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
  providers: []
})
export class AppRoutingModule {
}
```

App Module

```
const routes: Routes = [
  {path: '', component: HomeComponent},
  {path: 'items', component: ItemsComponent},
  {path: 'widgets', component: WidgetsComponent},
  {path: 'profile', loadChildren: './user/user.module#UserModule'},
  {path: '**', redirectTo: '', pathMatch: 'full'}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
  providers: []
})
export class AppRoutingModule {
}
```

App Module

localhost:4200/profile

Angular REST App

- Home
- Items
- Widgets
- Profile

Anders Hejlsberg

Software Engineer

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Audits | Augury | Redux | Adblock Plus

View: | Group by frame | Preserve log | Disable cache | Offline | No throttling

Filter | Regex | Hide data URLs | All | XHR | JS | CSS | Img | Media | Font | Doc | WS | Manifest | Other

Name	Status	Type	Initiator	Size	Time	Waterfall	100.00 ms
user.module.chunk.js	200 OK	script	inline.bundle.js:109 Script	8.1 KB 7.8 KB	10 ms 9 ms		
lake.8dc4651556e45fdd4370.jpg	304 Not Modified	jpeg	vendor.bundle.js:31063 Script	212 B 303 KB	9 ms 5 ms		
lake.jpg /assets/img	304 Not Modified	jpeg	vendor.bundle.js:117880 Script	212 B 303 KB	8 ms 4 ms		

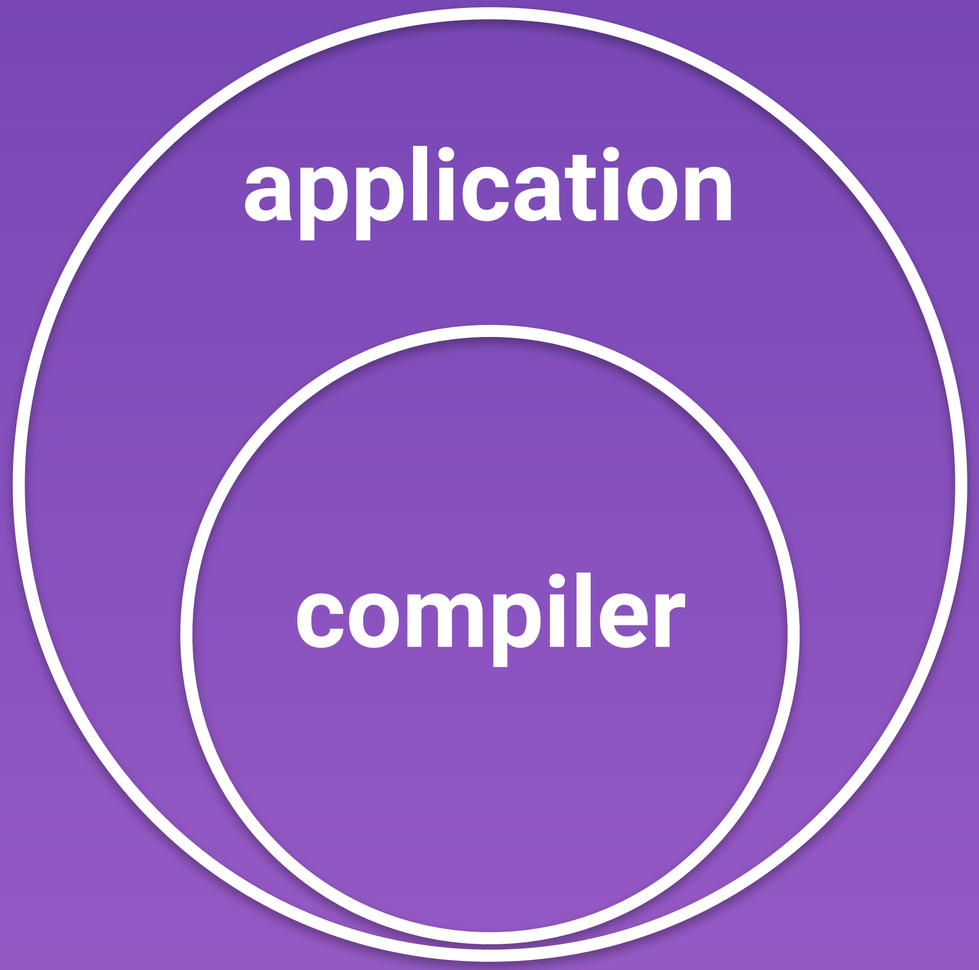
Make It Live

Deploying Applications

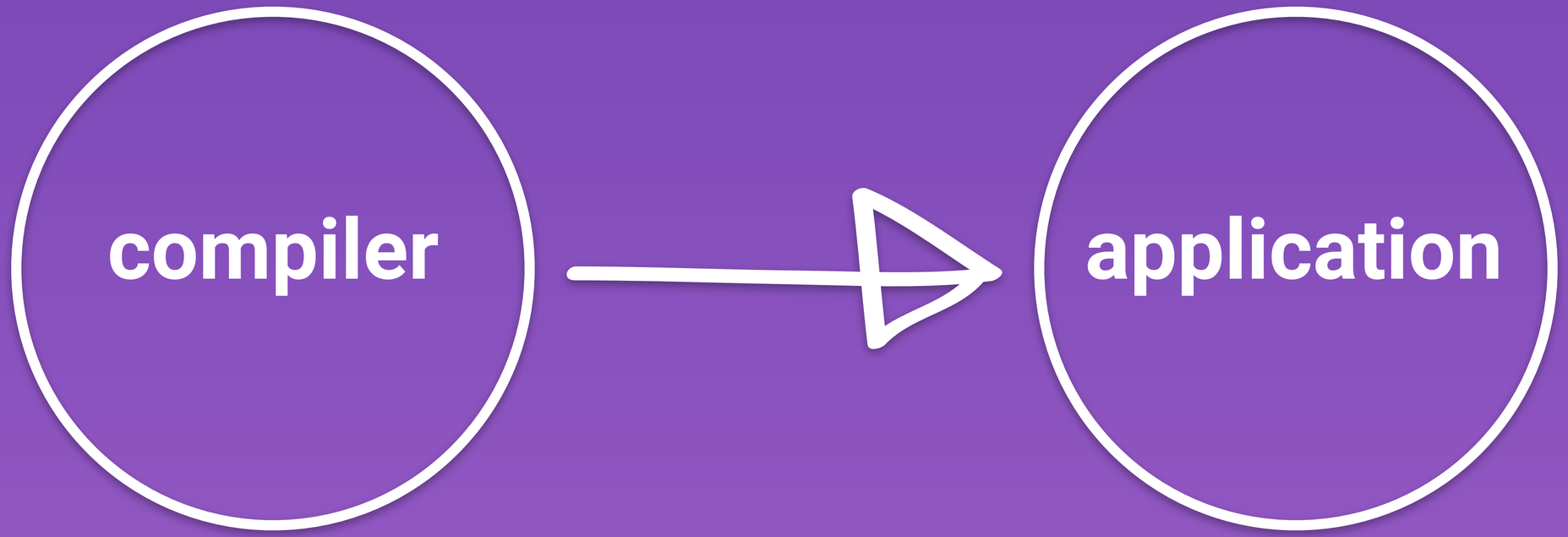
```
# these are equivalent
ng build --target=production --environment=prod
ng build --prod --env=prod
ng build --prod

# and so are these
ng build --target=development --environment=dev
ng build --dev --e=dev
ng build --dev
ng build
```

	dev	prod
aot	FALSE	TRUE
environment	dev	prod
output-hashing	media	all
sourcemaps	TRUE	FALSE
extract-css	FALSE	TRUE
named-chunks	TRUE	FALSE

AOT Compilation



JIT Compilation



AOT Compilation

smaller payload

fewer async requests

faster rendering

Benefits of AOT

Make It Realtime

Angular and Firebase

The **Realtime** Observable Stream

Start with a

realtime database

You called?

```
export const environment = {
  production: false,
  firebase: {
    apiKey: ‘MARCGRABANSKIISABEAST!',
    authDomain: ‘awesome-app.firebaseapp.com',
    databaseURL: 'https://awesome-app.firebaseio.com',
    projectId: 'awesome-app',
    storageBucket: '',
    messagingSenderId: '846368973507'
  }
};
```

```
imports: [
  AngularFireModule.initializeApp(environment.firebase),
  AngularFireDatabaseModule,
  BrowserAnimationsModule,
  BrowserModule,
  FormsModule,
  HttpModule,
  ReactiveFormsModule,
  AppRoutingModule,
  AppMaterialModule
]
```

```
imports: [
  AngularFireModule.initializeApp(environment.firebase),
  AngularFireDatabaseModule,
  BrowserAnimationsModule,
  BrowserModule,
  FormsModule,
  HttpModule,
  ReactiveFormsModule,
  AppRoutingModule,
  AppMaterialModule
]
```

Consume the realtime stream

```
notification$: FirebaseObjectObservable<Notification>;

constructor(db: AngularFireDatabase) {
  this.notification$ = db.object('/notification');
  this.notification$
    .skip(1)
    .subscribe(notification => this.subject.next(notification));
}

emit(notification: Notification) {
  this.notification$.set(notification);
}
```

```
notification$: FirebaseObjectObservable<Notification>;

constructor(db: AngularFireDatabase) {
  this.notification$ = db.object('/notification');
  this.notification$
    .skip(1)
    .subscribe(notification => this.subject.next(notification));
}

emit(notification: Notification) {
  this.notification$.set(notification);
}
```

Update the realtime stream

```
notification$: FirebaseObjectObservable<Notification>;

constructor(db: AngularFireDatabase) {
  this.notification$ = db.object('/notification');
  this.notification$
    .skip(1)
    .subscribe(notification => this.subject.next(notification));
}

emit(notification: Notification) {
  this.notification$.set(notification);
}
```

Make It Pretty

Angular Animations

https://www.yearofmoo.com/

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
```

BrowserAnimationsModule

```
import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';
import { Item } from '../../shared';
import {
  animate, group, query, stagger, style, transition, trigger
} from '@angular/animations';

@Component({
  selector: 'app-items-list',
  templateUrl: './items-list.component.html',
  styleUrls: ['./items-list.component.css'],
  animations: [ ]
})
export class ItemsListComponent implements OnInit {
}
```

@angular/animations

```
animations: [
  trigger('listAnimation', [
    transition(':enter, :leave, * => pending', []),
    transition('* => *', [
      // animate both the newly entered & removed items on the page at the same time
      group([
        query(':enter', [
          style({ opacity: 0, height: '0px' }),
          stagger('50ms', [
            animate('500ms cubic-bezier(.35,0,.25,1)', style('*'))
          ])
        ], { optional: true }),
        query(':leave', [
          stagger('50ms', [
            animate('500ms cubic-bezier(.35,0,.25,1)',
              style({ opacity: 0, height: '0px', borderTop: 0, borderBottom: 0 }))
          ])
        ], { optional: true })
      ]),
    ]),
  ]),
]
```

```
animations: [
  trigger('listAnimation', [
    transition(':enter, :leave, * => pending', []),
    transition('* => *', [
      // animate both the newly entered & removed items on the page at the same time
      group([
        query(':enter', [
          style({ opacity: 0, height: '0px' }),
          stagger('50ms', [
            animate('500ms cubic-bezier(.35,0,.25,1)', style('*'))
          ])
        ], { optional: true }),
        query(':leave', [
          stagger('50ms', [
            animate('500ms cubic-bezier(.35,0,.25,1)',
              style({ opacity: 0, height: '0px', borderTop: 0, borderBottom: 0 }))
          ])
        ], { optional: true })
      ]),
    ]),
  ]),
]
```

```
<md-list
  [@listAnimation]="prepareListState()"
  [@.disabled]="animationsDisabled">
  <md-list-item *ngFor="“let item of items; trackBy: trackItem">
    <h3 md-line>{{item.name}}</h3>
    <p md-line>
      {{item.description}}
    </p>
  </md-list-item>
</md-list>
```

```
<md-list
  [@listAnimation]="prepareListState()"
  [@.disabled]="animationsDisabled">
  <md-list-item *ngFor="“let item of items; trackBy: trackItem">
    <h3 md-line>{{item.name}}</h3>
    <p md-line>
      {{item.description}}
    </p>
  </md-list-item>
</md-list>
```

```
export class ItemsListComponent implements OnInit {
  animationsDisabled = true;

  trackItem(index, item) {
    return item.id;
  }

  ngOnInit() {
    setTimeout(() => {
      this.animationsDisabled = false;
    }, 500)
  }

  prepareListState() {
    return this.items ? this.items.length : 'pending';
  }
}
```

I ♥ YOU!

@simpulton

Thanks!

WAT.