```
1
2 <h1>
3         Techno Web
4 </h1>
5
```

1
2
3
4
5

RÉMI POURTIER

accenture

POLYTECH NICE-SOPHIA

RECHARGE +

LA FABRIQUE LOGICIEL

3 Évaluation régulière GIT

- ➔ Tous les membres du groupe doivent contribuer au projet
- ➔ Nous regarderons chaque semaine les contributions de chaque projet
- ➔ La régularité de votre travail comptera dans l'évaluation “Évaluations techniques”

4 Démarrage du projet

➔ Approche projet !

1 starter de code Frontend : https://classroom.github.com/a/RI2sB5fM

1 point de départ : le sujet et sa documentation technique :
https://lms.univ-cotedazur.fr/2022/course/view.php?id=10101&section=11#tabs-tree-start

Basés sur un exemple de code :
https://github.com/NablaT/ps6-correction-td1-td2-v2

2 TDs :
- TD 1 frontend
- TD 2 backend

5 Démarrage du projet

➔ La démarche pour développer

Objectif : Développer la fonctionnalité A

1 Sujet

2 Documentation Technique

Comprendre les concepts

6 Démarrage du projet

➔ La démarche pour développer

Objectif : Développer la fonctionnalité A

1 Sujet

2 Documentation Technique

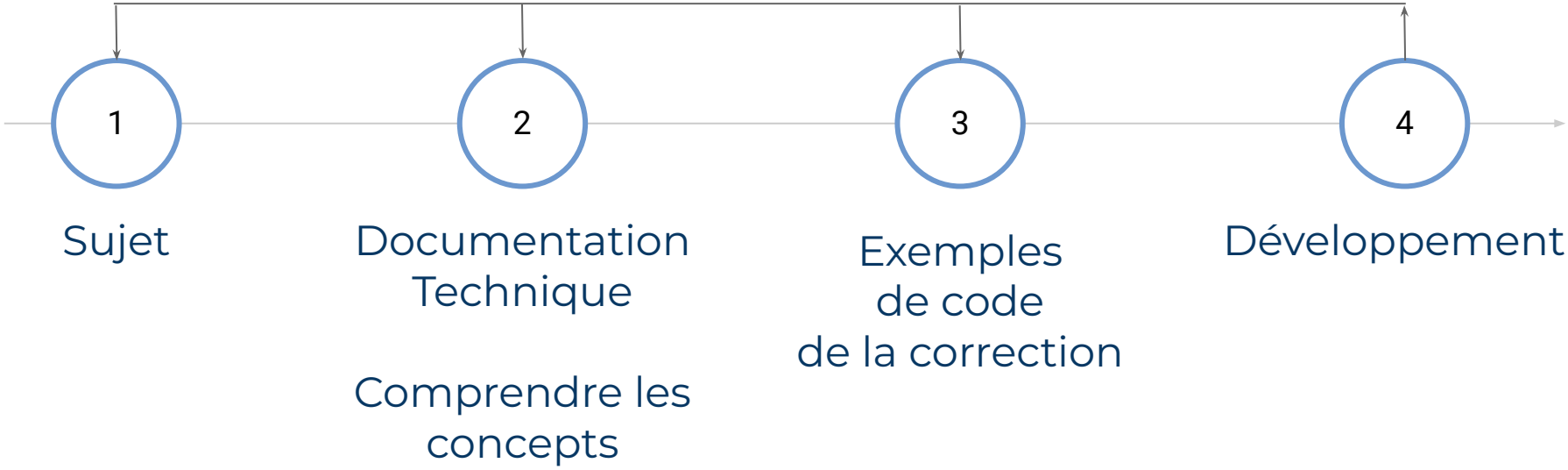
Comprendre les concepts

Découverte LMS

7 Démarrage du projet

➔ La démarche pour développer

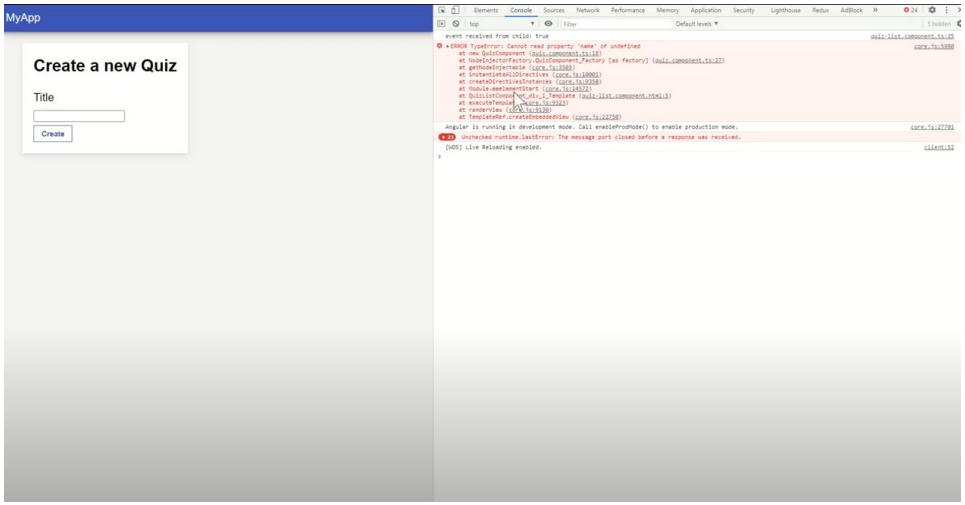
Objectif : Développer la fonctionnalité A

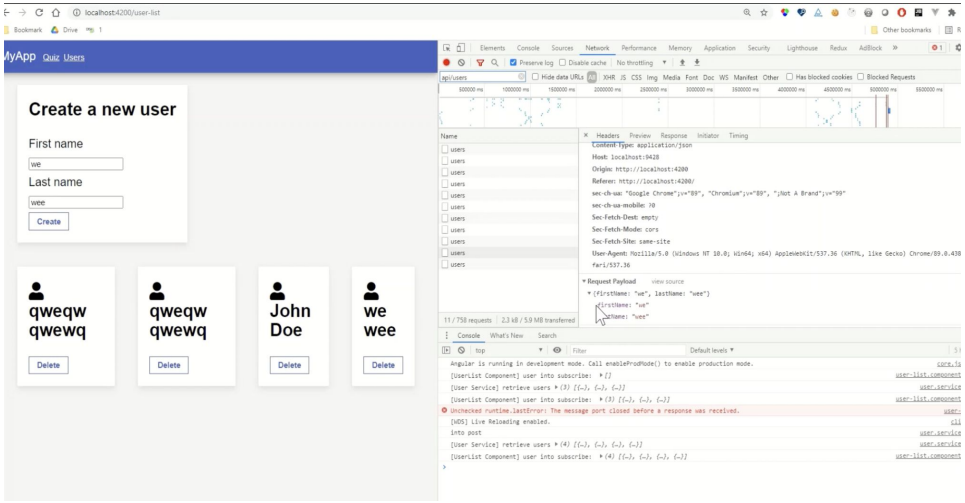


8 Démarrage du projet

[Présentation du starter]

[Implémentation de la gestion des utilisateurs]





9 Démarrage du projet

➔ Approche projet !

Les clés pour réussir :

Avancez **tous** chaque semaine sur le projet

Bonne **répartition** du travail

Profitez de l'équipe encadrante

10 Vous avez les bases

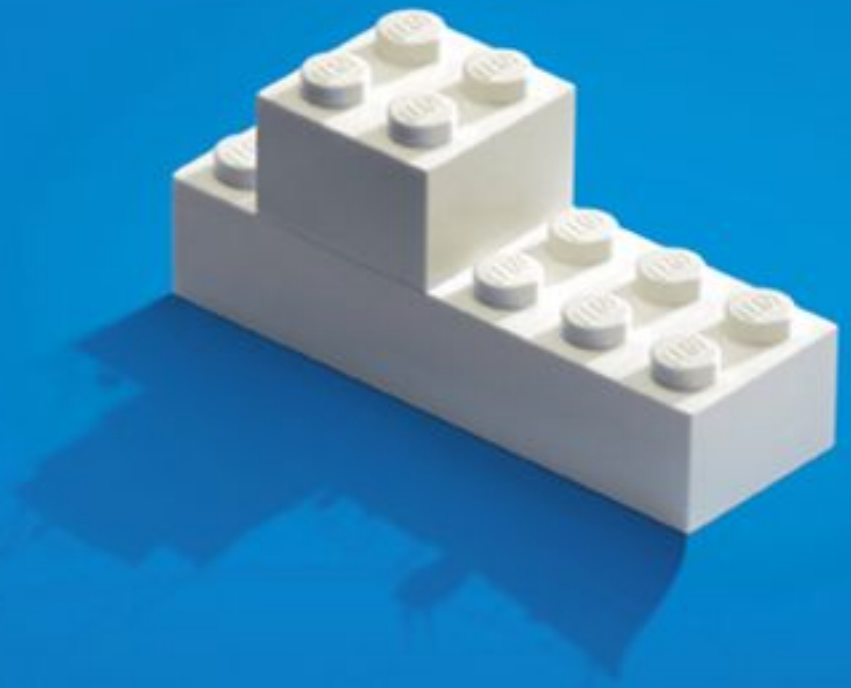
11 Frameworks


REQUIRE·JS
npm
TS
BABEL
JS
A

13 ANGULAR

14

WEB COMPONENTS



15 WEB COMPONENTS

Approche modulaire

À quoi ressemble une page web ?

Juste une grosse entité

16 WEB COMPONENTS

Approche modulaire

Qui peut être complexe...

17 WEB COMPONENTS

Approche modulaire

Conséquences ?

Duplication de code, difficile à modifier et à maintenir

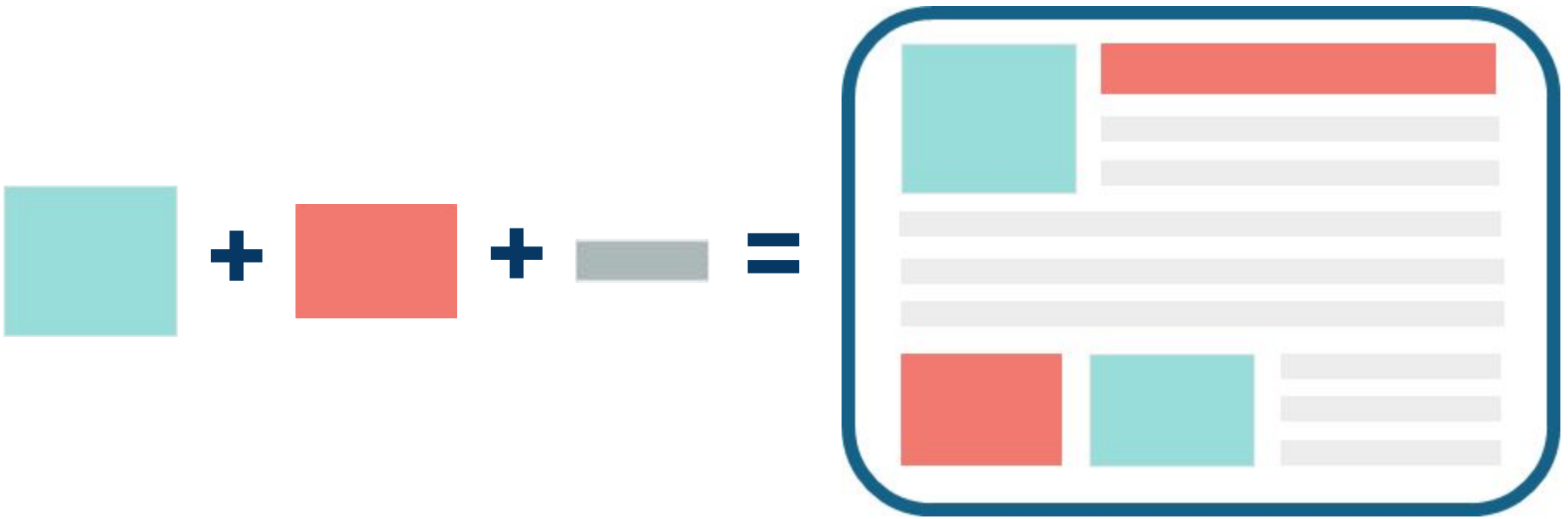
18 WEB COMPONENTS

Approche modulaire

Comment découper cette page ?

19 WEB COMPONENTS

Approche modulaire



20 WEB COMPONENTS

Approche modulaire

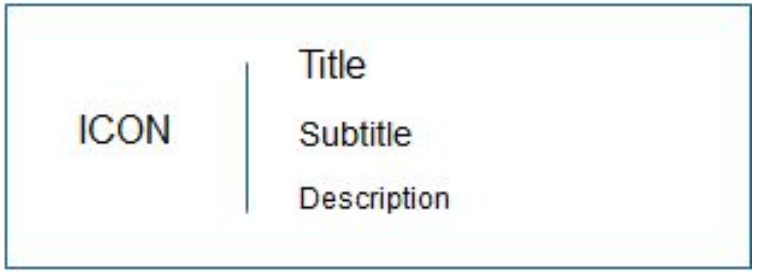
Et en vrai,
on fait comment?

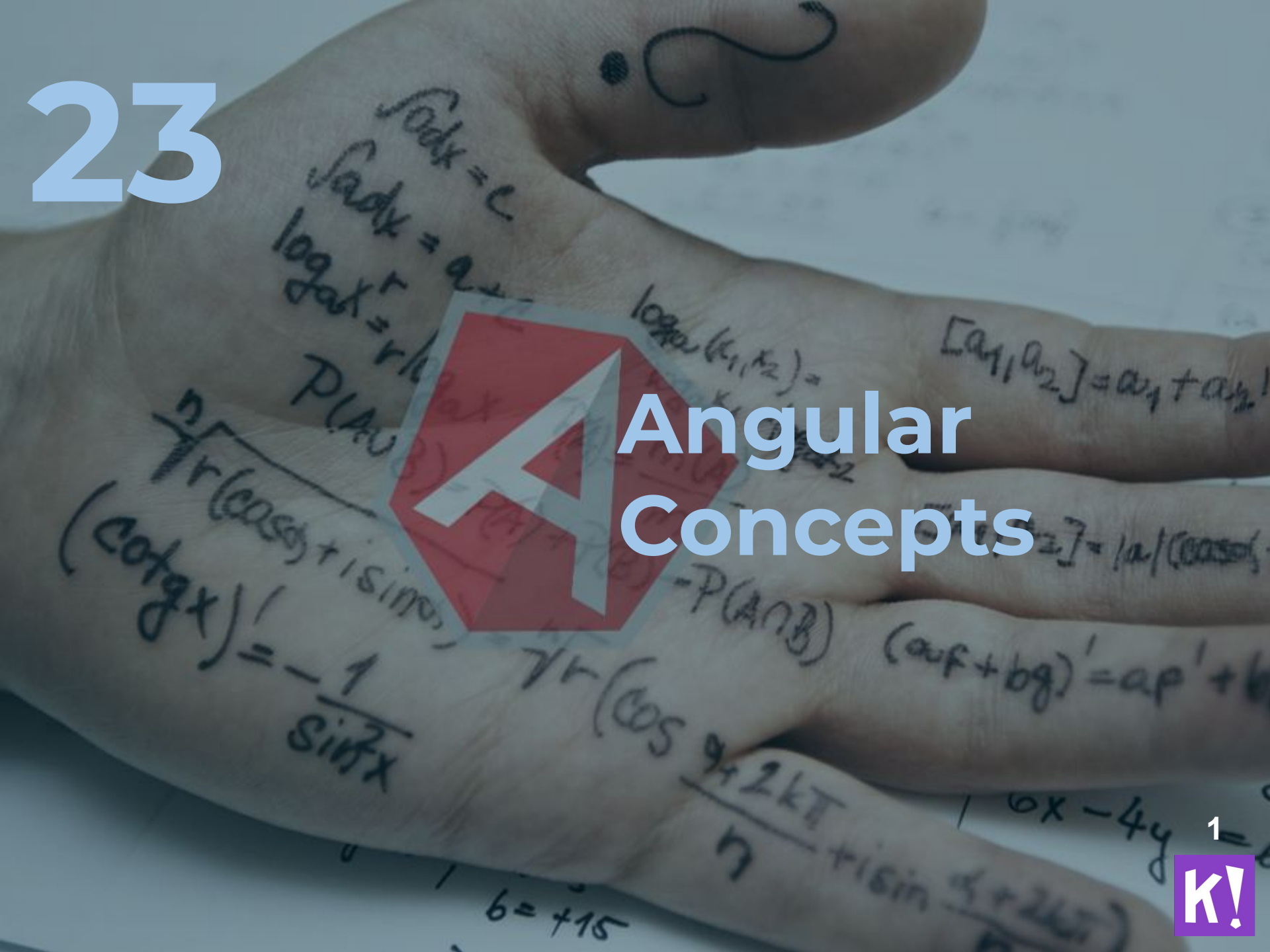
21 WEB COMPONENTS

Approche modulaire

22 WEB COMPONENTS

Approche modulaire





23

Angular Concepts

24 ANGULAR CONCEPTS

COMPONENT

Template

Component Class

Component Test

Component style

25 ANGULAR CONCEPTS

COMPONENT

```
import { Component, OnInit } from "@angular/core";

@Component({
  selector: "app-home",
  templateUrl: "./home.component.html",
  styleUrls: ["./home.component.css"]
})
export class HomeComponent implements OnInit {

  public title: string;
  public description: string;

  constructor() {
    this.title = "My app";
    this.description = "My first app";
  }

  ngOnInit() { }

}
```

```
<h2>{{title}}</h2>
<div class="description">
  {{description}}
</div>
```

```
:host {
  background-color: blue;
}

h2 {
  color: white;
  font-weight: bold;
}

.description {
  color: lightgray;
}
```

26 ANGULAR CONCEPTS

COMPONENT
Utilisation

```
<header>
    <h1>MyApp</h1>
</header>
<body>
    <app-home></app-home>
    <button>Next page</button>
</body>
```

27 ANGULAR CONCEPTS

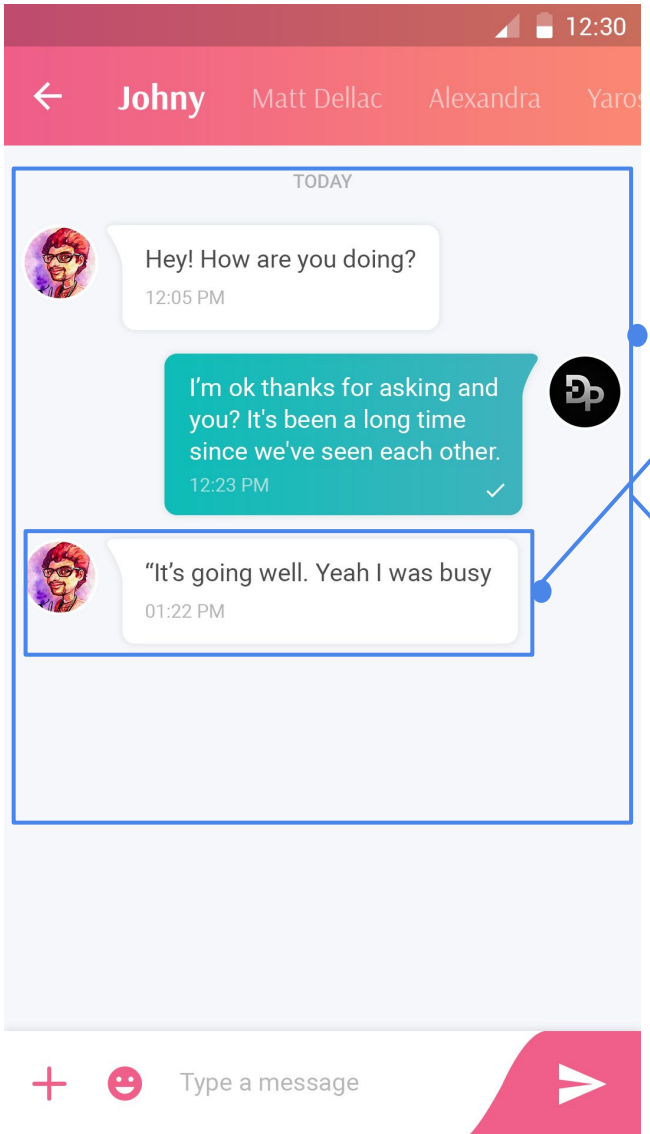
COMPONENT

```ts
import { Component, OnInit } from "@angular/core";

@Component({
  selector: "app-home",
  templateUrl: "./home.component.html",
  styleUrls: ["./home.component.css"]
})
export class HomeComponent implements OnInit {

  public title: string;
  public description: string;

  constructor() {
    this.title = "My app";
    this.description = "My first app";
  }

  ngOnInit() { }

}
```

```html
<h2>{{title}}</h2>
<div class="description">
  {{description}}
</div>
```

```css
:host {
  background-color: blue;
}

h2 {
  color: white;
  font-weight: bold;
}

.description {
  color: lightgray;
}
```

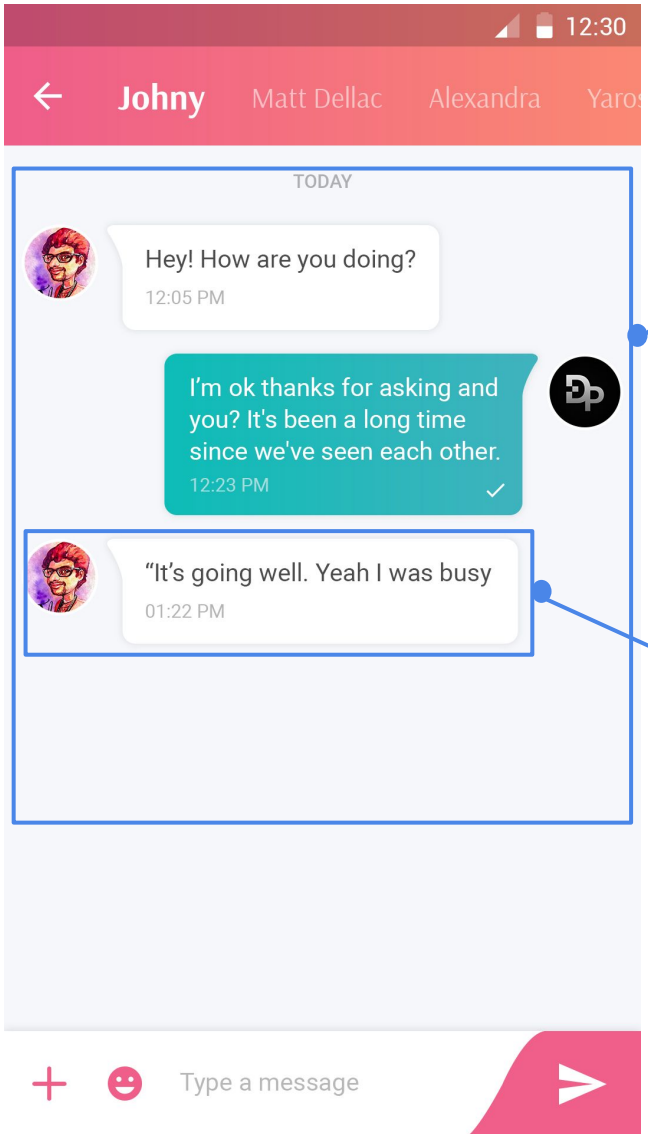
28 ANGULAR CONCEPTS



MessageComponent

MessageListComponent

29 ANGULAR CONCEPTS



Input & Output

MessageComponent

@Output

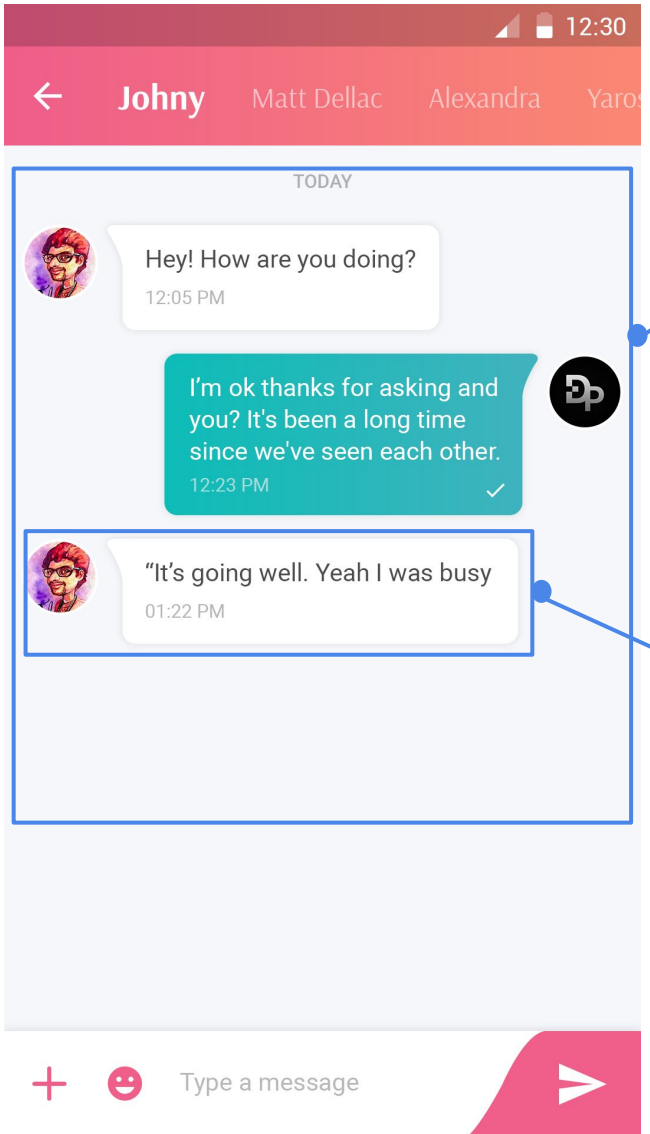
Event
"Message lu"

@Input

Message
Sender
Date

MessageListComponent

30 ANGULAR CONCEPTS



Input

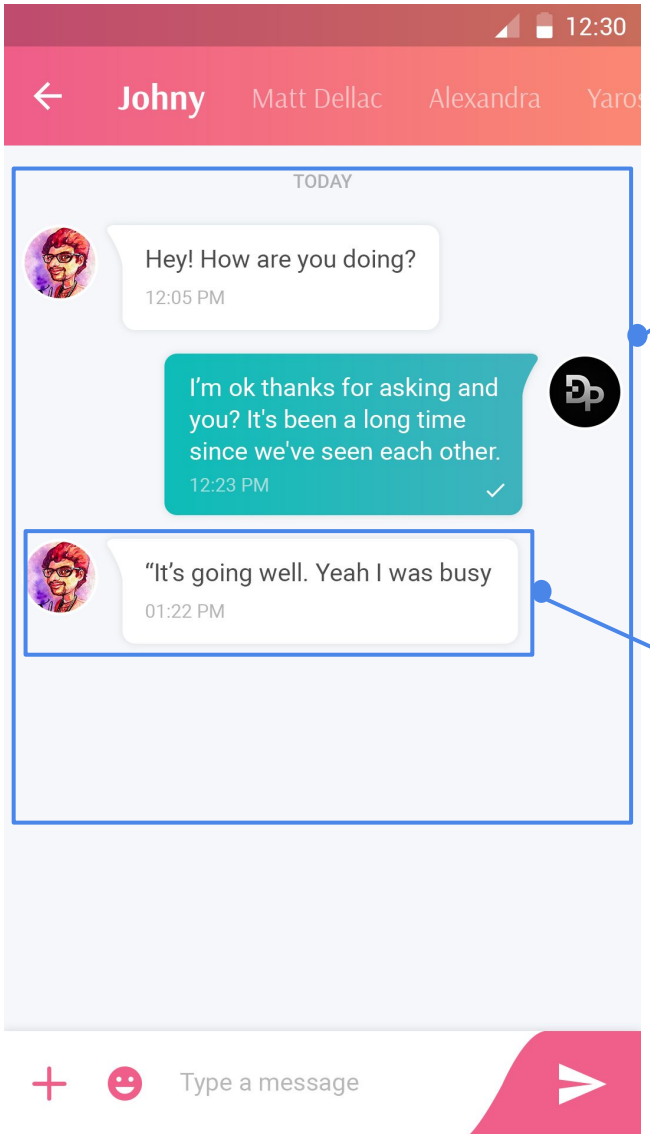
MessageComponent

@Input

Message
Sender
Date

MessageListComponent

31 ANGULAR CONCEPTS



Services

MessageComponent

@Output

Event
"Message lu"

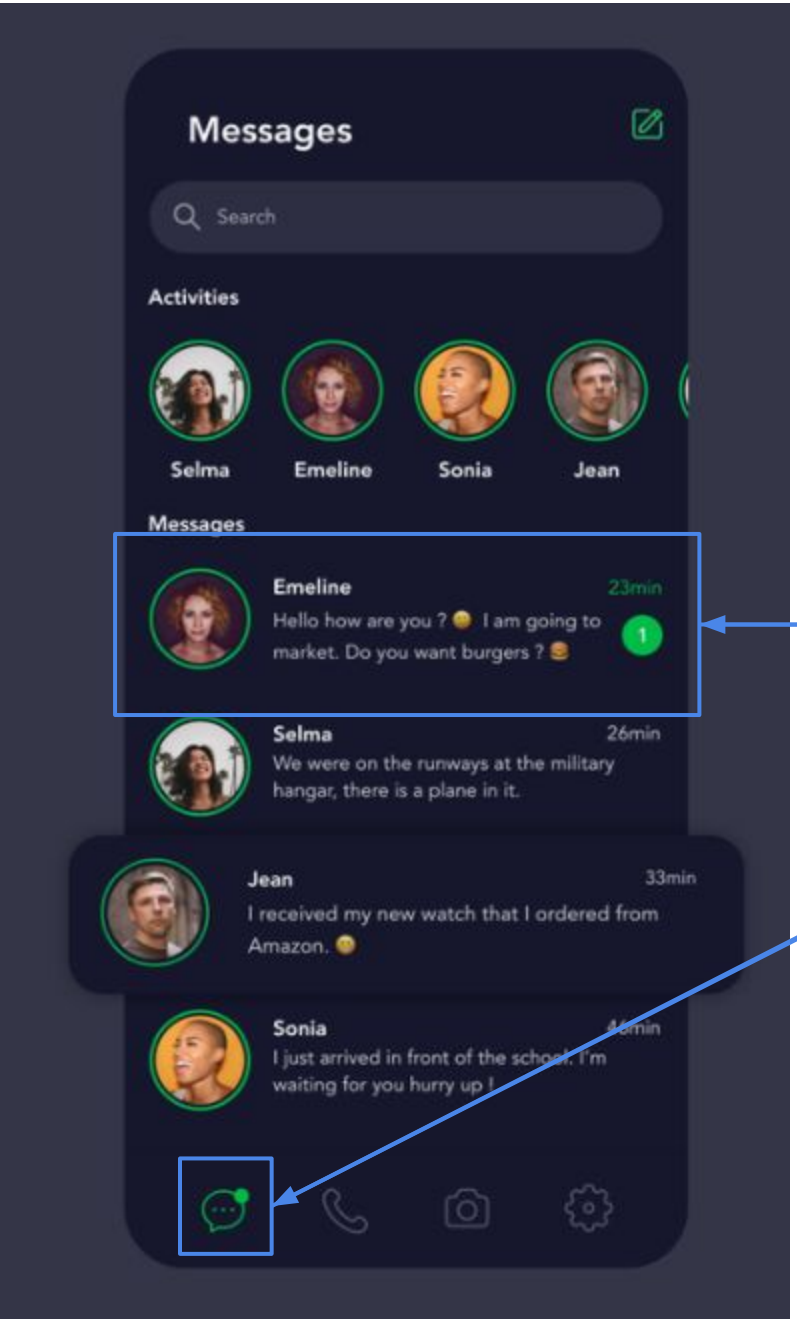
@Input

Message
Sender
Date

MessageListComponent

getMessages()

MessageService



Services

MessageService

SERVICE

_ Contient un ensemble de fonctions partagées par plusieurs composants

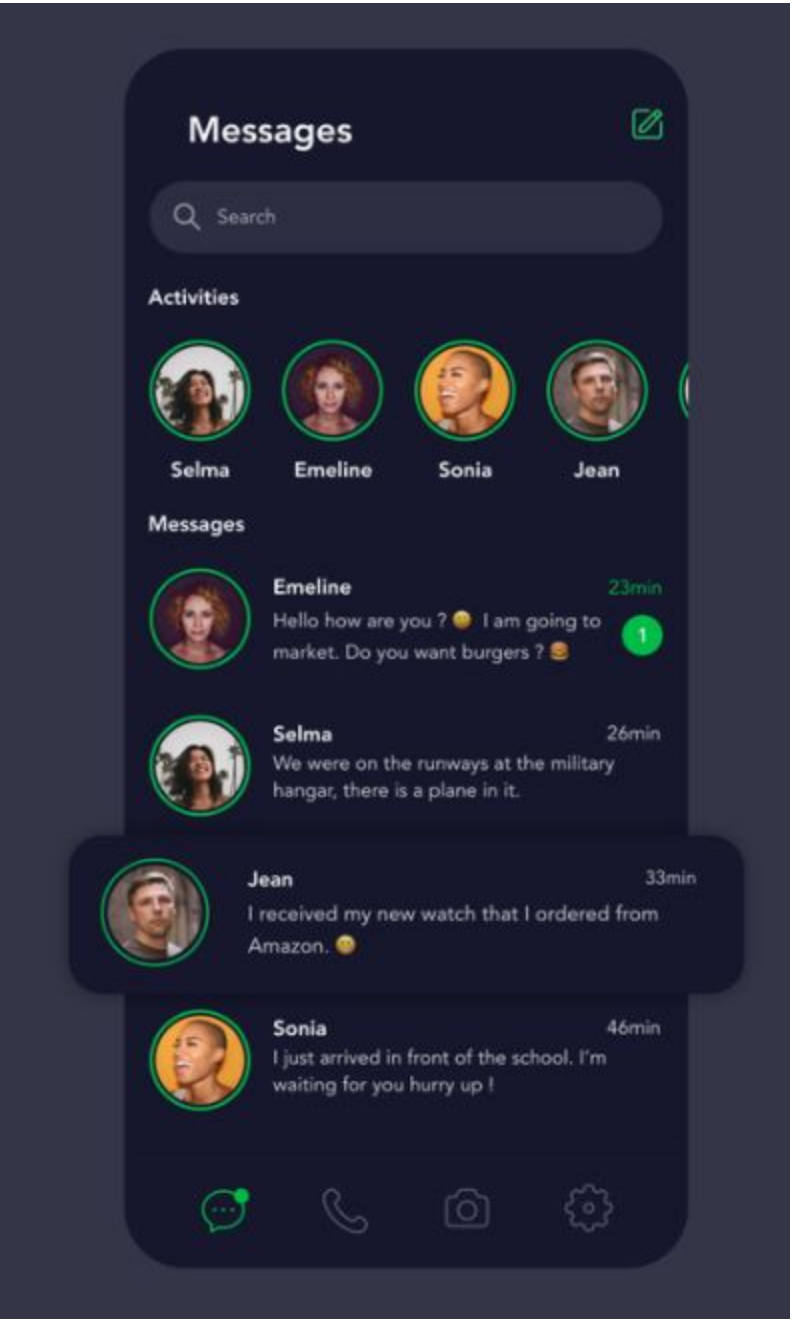
_ Les fonctionnalités du service doivent être uniques

_ Tous les composants peuvent utiliser un service


```typescript
@Injectable()
export class LoginService {

  public verifyCredential(user) { }
  public updatePassword(user, password) { }

}
```

Pour communiquer
On utilise les
INPUTS/OUTPUTS

Pour communiquer
On utilise les
SERVICES



Component Layer

Message

Message

Message

Messages

Icon Icon

Menu

Update the data

Get data

Service Layer

MessageService

Messages: [
{…}, {…}, {…}
]

UserService

currentUser: {…}

36 ANGULAR CONCEPTS

DIRECTIVE & PIPE

_ Directives:
- Ajoute un comportement à un élément du DOM

_ Pipe:
- Reçoit en entrée des données
- Les transforme
- Les retourne

20/02/18

Mardi 20 Février 2018

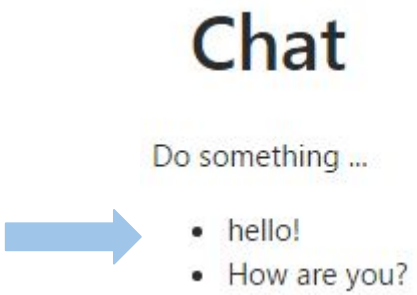
37 ANGULAR CONCEPTS

DIRECTIVES NATIVES

_ ngIf, ngFor, ngSwitch, ngClass, ngModel ...

```ts
public title: string;
public messageList: string[];

constructor() {
  this.title = "Chat";
  this.messageList = ["hello!", "how are you?"];
}
```

```html
<div *ngIf="title === 'Chat'">
  Do something ...
</div>

<div *ngFor="let message of messageList">
  {{message}}
</div>
```



38

Handle async

39 JS : callbacks

```
function foo(finalCallback) {
  request.get(url1, function(err1, res1) {
    if (err1) { return finalCallback(err1); }
    request.post(url2, function(err2, res2) {
      if (err2) { return finalCallback(err2); }
      request.put(url3, function(err3, res3) {
        if (err3) { return finalCallback(err3); }
        request.del(url4, function(err4, res4) {
          // let's stop here
          if (err4) { return finalCallback(err4); }
          finalCallback(null, "whew all done");
        })
      })
    })
  })
}

// use that function somewhere
foo(function(err, message) {
  if (err) {
    return console.log("error!", err);
  }
  console.log("success!", message);
});
```

40 JS : promesses

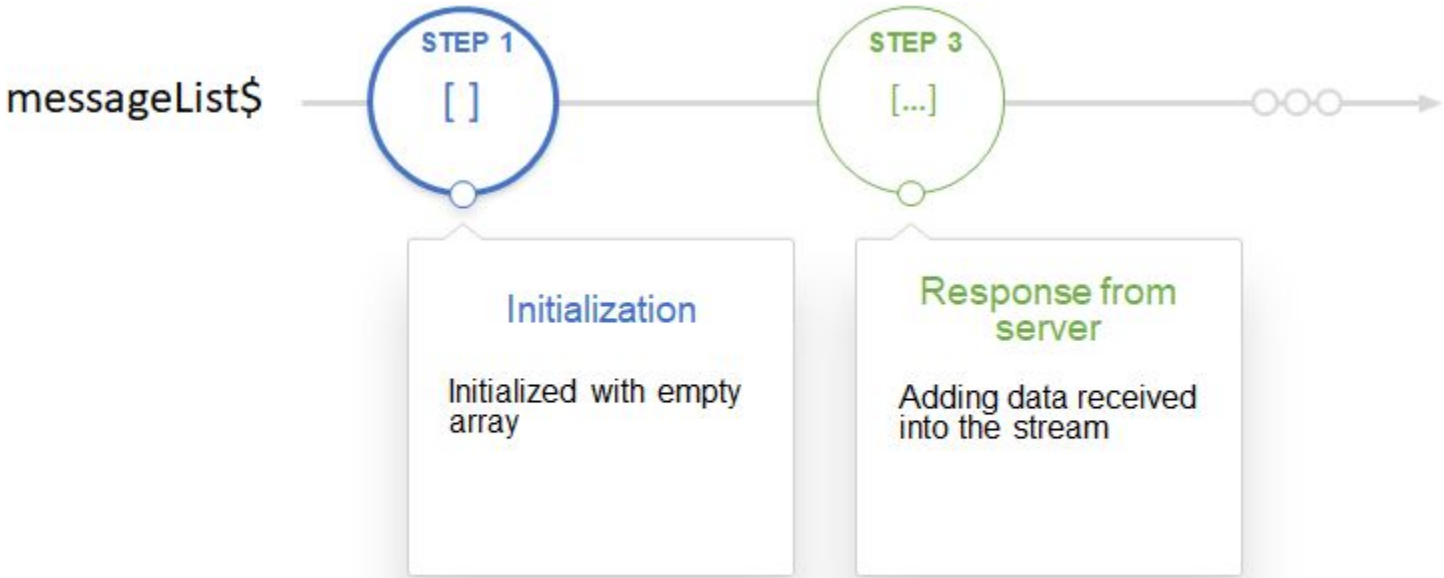
```
function foo() {
  return request.getAsync(url1)
  .then(function(res1) {
    return request.postAsync(url2);
  }).then(function(res2) {
    return request.putAsync(url3);
  }).then(function(res3) {
     return request.delAsync(url4);
  }).then(function(res4) {
     return "whew all done";
  });
}

// use that function somewhere
foo().then(function(message) {
  console.log("success!", message);
}).catch(function(err) {
  console.log("error!", err);
});
```

41 JS : async/await

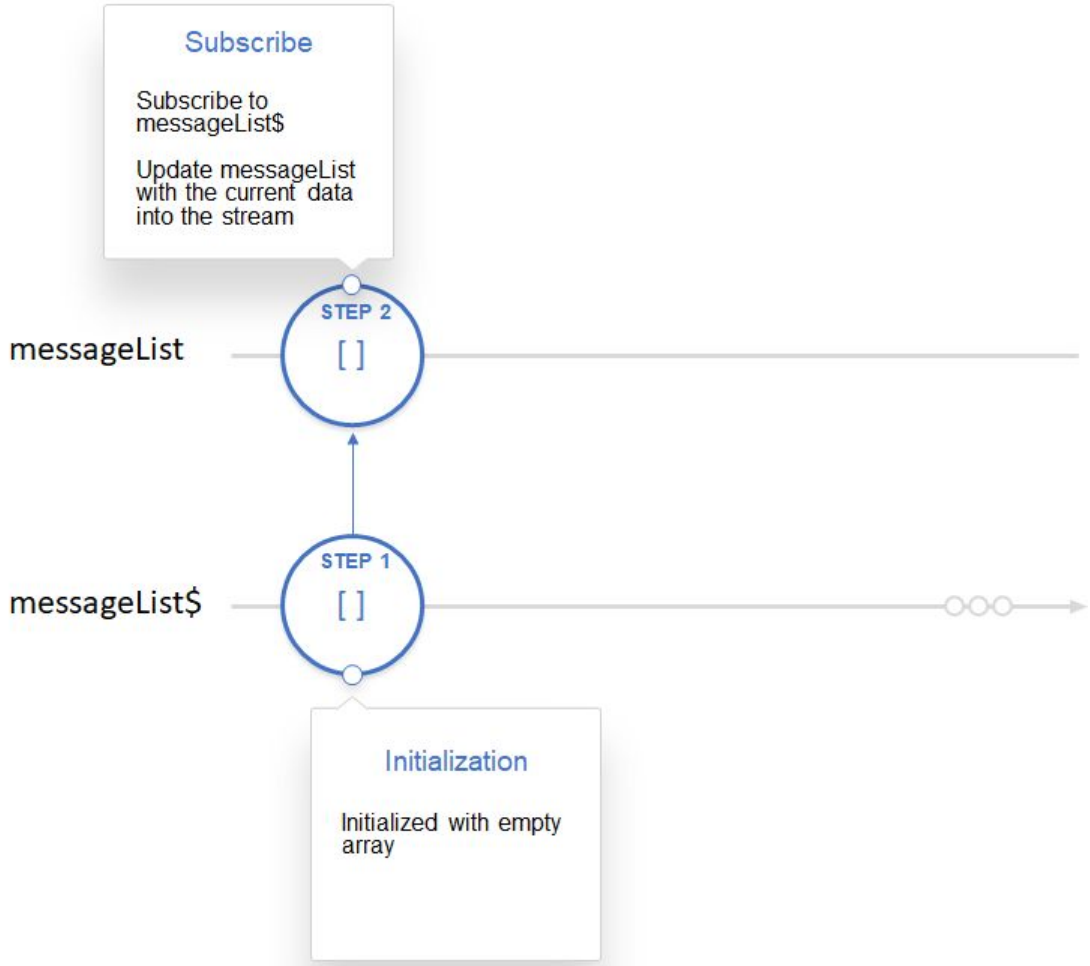
```
async function foo() {
  var res1 = await request.getAsync(url1);
  var res2 = await request.getAsync(url2);
  var res3 = await request.getAsync(url3);
  var res4 = await request.getAsync(url4);
  return "whew all done";
}

// use that function somewhere
foo().then(function(message) {
  console.log("success!", message);
}).catch(function(err) {
  console.log("error!", err);
});
```

42 Observables

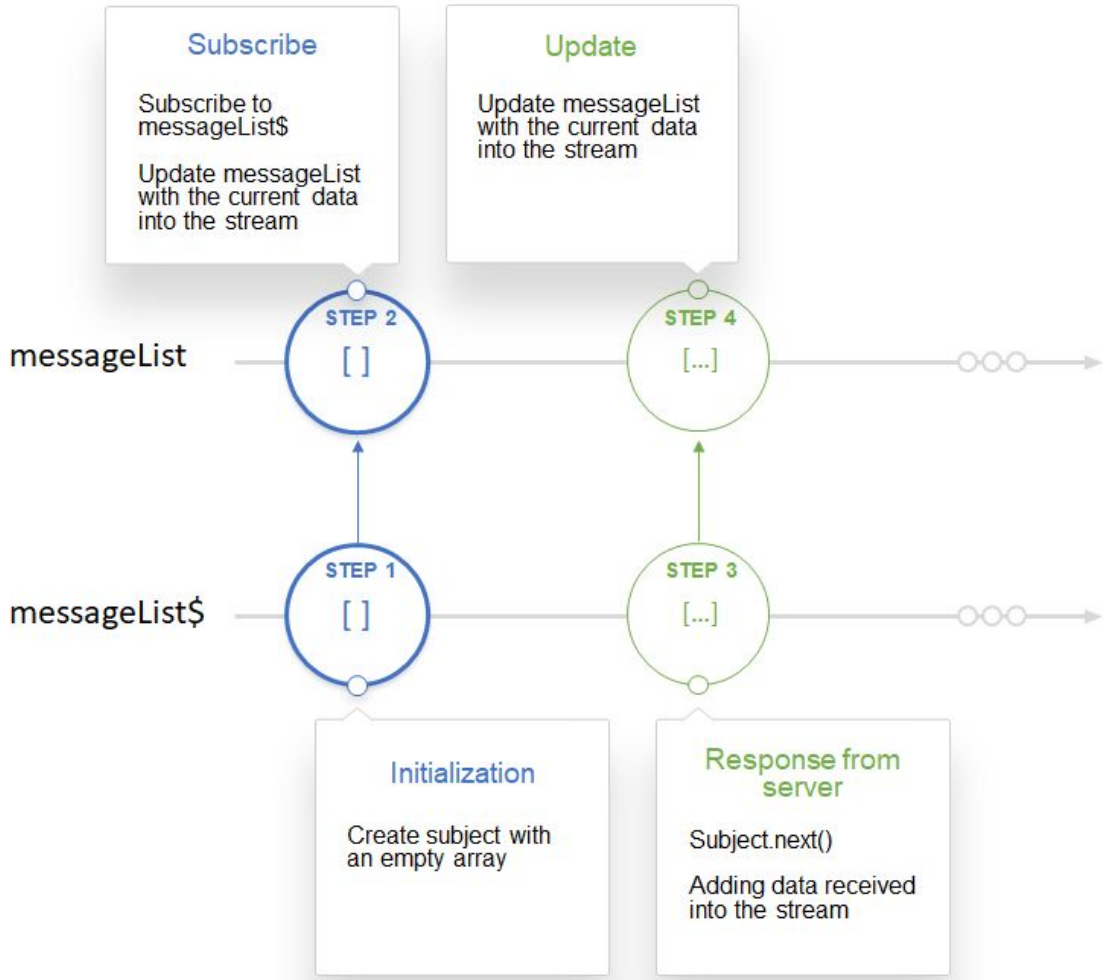
43 Observables



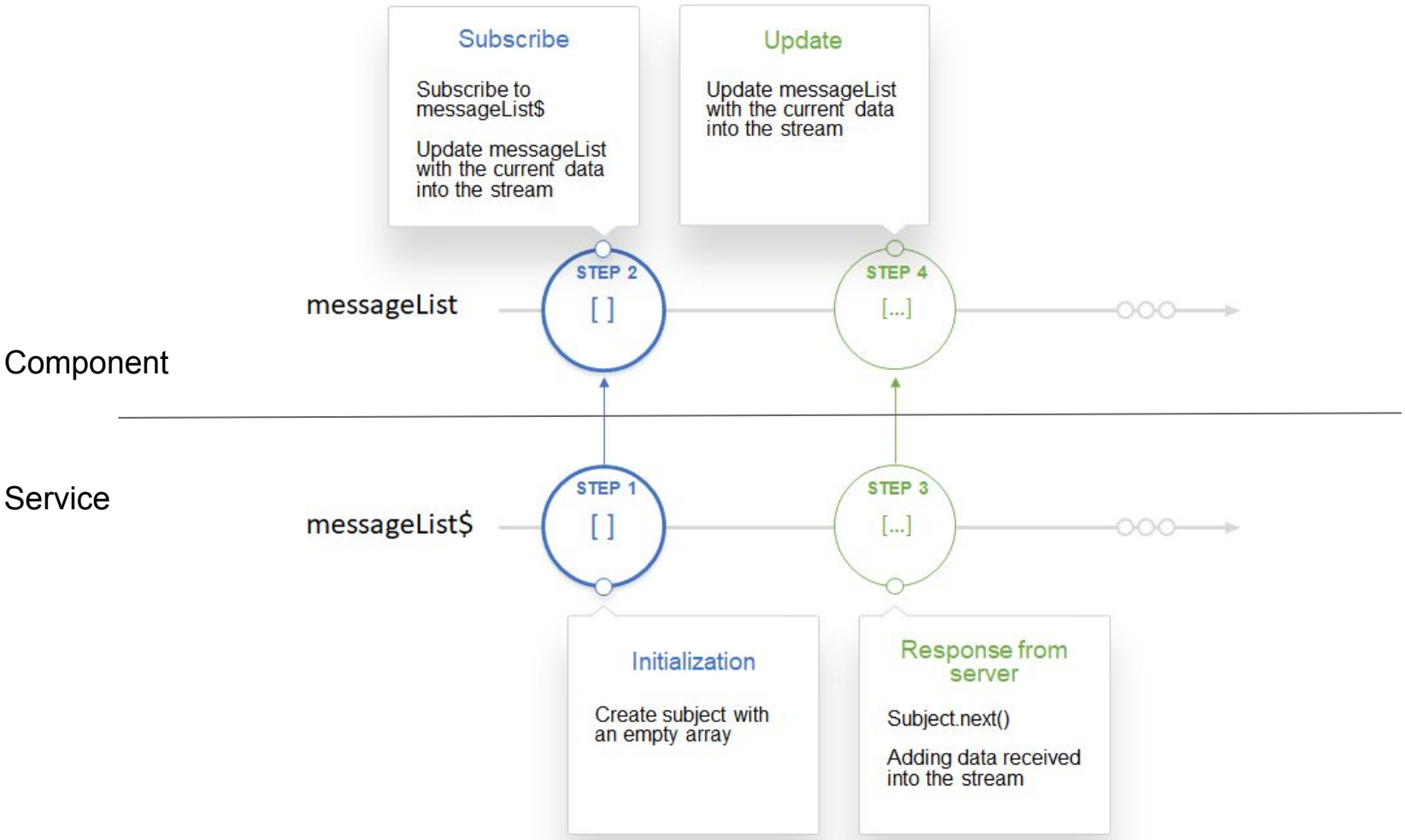
44 Observables



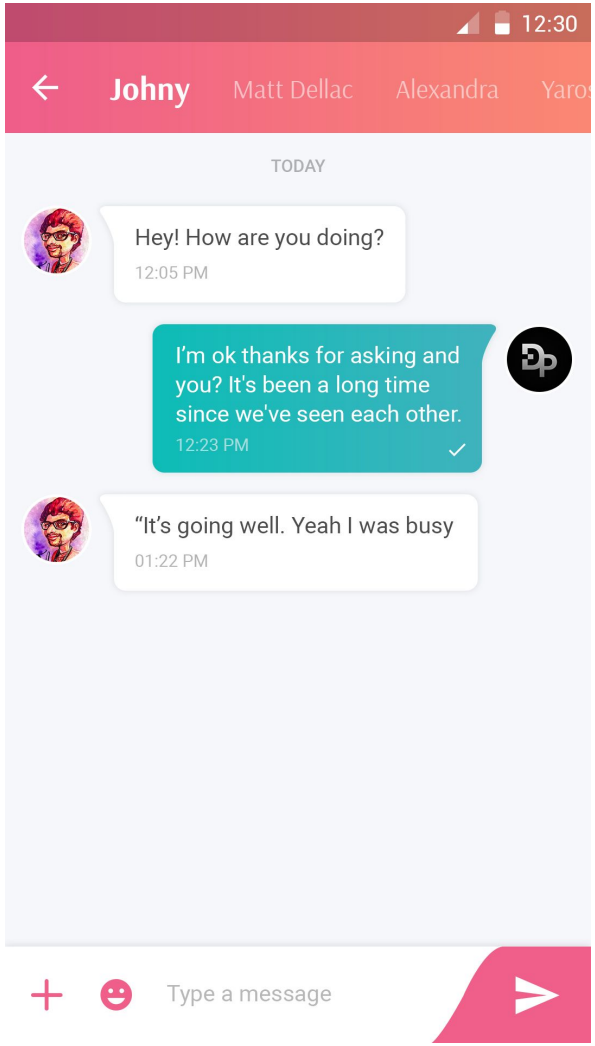
45 Observables



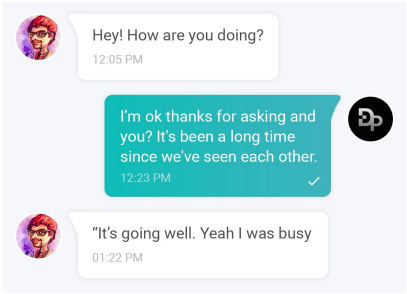
46 Observables



47 Observables



Messages Component



subscribes

In Service Messages$

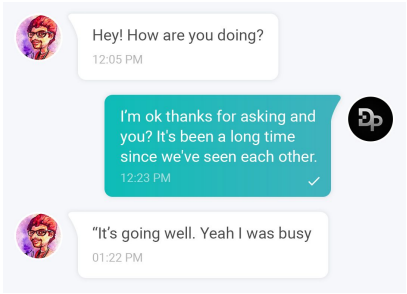
1

[ ]

2

[message1, message2, message3]

Messages Component



subscribes

In Service Messages$

1

[ ]

2

[message1, message2, message3]

3

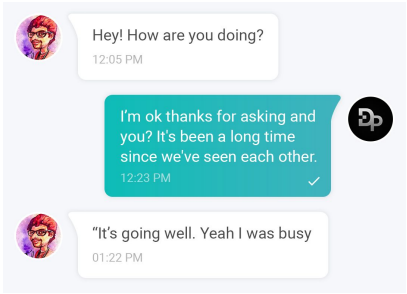
[message1, message2, message3, **message4**]

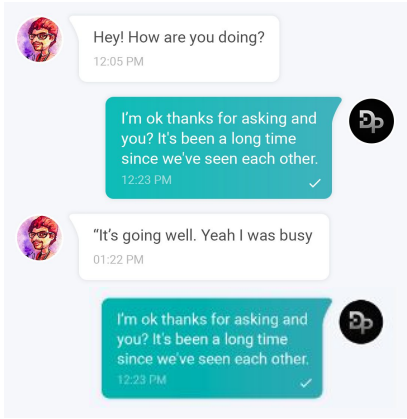
updates

Type a message

MessageForm Component

Messages Component





subscribes

In Service Messages$

1

[ ]

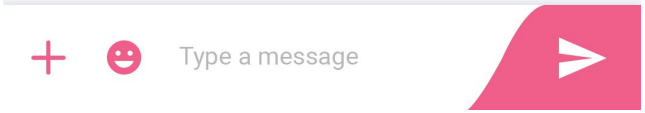
2

[message1, message2, message3]

3

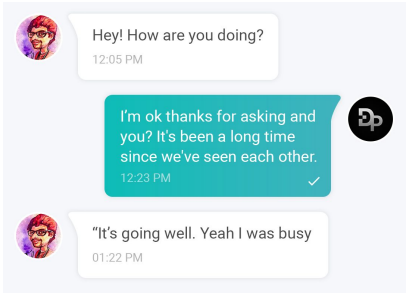
[message1, message2, message3, message4]

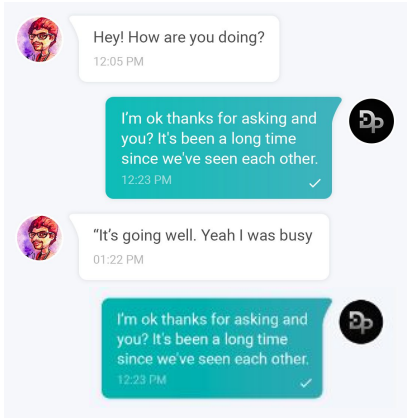
updates



MessageForm Component

Messages Component





subscribes

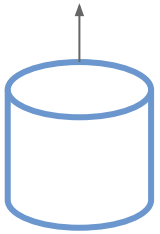
In Service Messages$

1 → [ ]

2 → [message1, message2, message3]

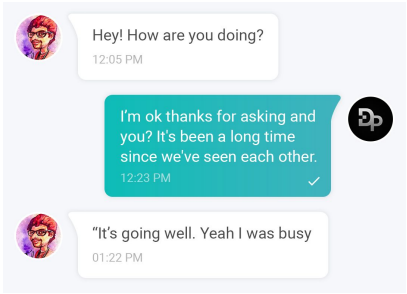
3 → [message1, message2, message3, message4]

4 → [message1, message2, message3, message4, **message5**]



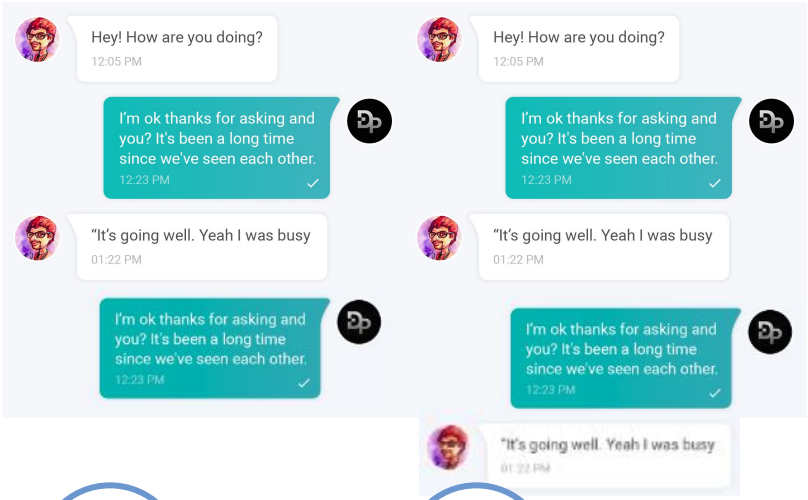
New message from server

Messages Component



subscribes

In Service Messages$



1 — [ ]

2 — [message1, message2, message3]

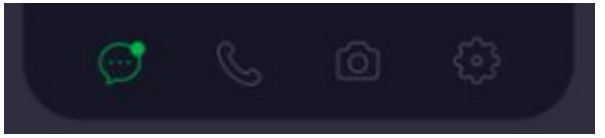
3 — [message1, message2, message3, message4]

4 — [message1, message2, message3, message4, message5]

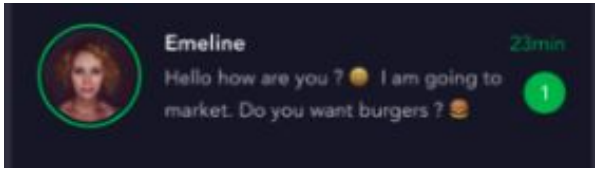
New message from server

subscribes

Menu Component



Conversation Component



53 Observables

Service

```
@Injectable()
export class MessageService {

  public messageList$: BehaviorSubject<MessageModel[]>;

  constructor() {
    this.messageList$.next( value: []);
  }

  updateMessageList(messageList: MessageModel[]) {
    this.messageList$.next(messageList);
  }

}
```

1

3

Component

```
export class MessageListComponent implements OnInit {

  public messageList: MessageModel[];

  constructor(private messageService: MessageService) {
      this.messageService.messageList$.subscribe(
        next: (messageList) => this.messageList = messageList
      );
  }

  ngOnInit() { }
}
```

2


QUESTIONS?