

1

2 <h1>

3 Techno Web

4 </h1>

5

2 Évaluation régulière GIT

- À partir du 22 Mars, début du développement de votre projet
- 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 "Evaluations techniques" (Cf slide Evaluation Anne-Marie cours 1)

3 Vous n'avez pas les bases



4

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello World !</title>
</head>
<body>

  <span class="yay">Hello</span>
  <span class="test">World</span>
  <span class="coucou">!</span>

</body>
</html>
```

+

```
.yay {
  color: red;
}
.test {
  color: green;
  text-decoration: underline;
}
.coucou {
  font-size: 32px;
}
```

=

Hello World !

Tous les tags html

<https://www.w3schools.com/tags/default.asp>

Toutes les propriétés CSS

<https://www.w3schools.com/cssref/default.asp>

5 Javascript ?

- Faire bouger, apparaître ou disparaître des éléments de la page (un titre, un menu, un paragraphe, une image...).
- Mettre à jour des éléments de la page sans recharger la page (changer le texte, recalculer un nombre, etc).
- Demander au serveur un nouveau bout de page et l'insérer dans la page en cours, sans la recharger.
- Attendre que l'utilisateur fasse quelque chose (cliquer, taper au clavier, bouger la souris...) et réagir (faire une des opérations ci-dessus suite à cette action).

6

→ Faire bouger, apparaître ou disparaître des éléments de la page (un titre, un menu, un paragraphe, une image...).

```
// Add element to the DOM
var $h1 = document.createElement('h1');
$h1.innerHTML = 'Hello World !';
document.querySelector('body').appendChild($h1);

// Remove it from the DOM
document.querySelector('h1').remove();

// Hide
document.querySelector('h1').style.display = 'none';
// or
document.querySelector('h1').style.visibility = 'hidden';

// Show
document.querySelector('h1').style.display = 'block';
// or
document.querySelector('h1').style.visibility = 'visible';

// Make it move
document.querySelector('h1').style.left = '20px';
```

7

→ Mettre à jour des éléments de la page sans recharger la page (changer le texte, recalculer un nombre, etc).

```
document.querySelector('div').innerHTML =
    '<h1>' +
    '    Hello World ' +
    '    <span>' +
    '        (bis)' +
    '    </span> ' +
    '    !' +
    '</h1>';

var $h1 = document.createElement('h1');
$h1.innerHTML = 'Hello World ';
var $span = document.createElement('span');
$span.innerHTML = '(bis)';
$h1.appendChild($span);
$h1.innerHTML += ' !';
document.querySelector('div').appendChild($h1);

document.querySelector('.resultat').innerHTML =
    parseInt(document.querySelector('input[type="number"].numberOne').value) +
    parseInt(document.querySelector('input[type="number"].numberTwo').value)
```


8

→ Demander au serveur un nouveau bout de page et l'insérer dans la page en cours, sans la recharger.

```
function httpGet(URL, callback) {  
    var xmlHttp = new XMLHttpRequest();  
    xmlHttp.onreadystatechange = function () {  
        if (xmlHttp.readyState === 4 && xmlHttp.status === 200) {  
            callback(xmlHttp.responseText);  
        }  
    };  
    xmlHttp.open('GET', URL, true);  
    xmlHttp.send(null);  
}  
  
httpGet('http://une-url-qui-renvoie-du-html.com', function (resultatHTML) {  
    // WARNING : Security...  
    document.querySelector('body').appendChild(resultatHTML);  
});
```


9

→ Attendre que l'utilisateur fasse quelque chose (cliquer, taper au clavier, bouger la souris...) et réagir (faire une des opérations ci-dessus suite à cette action).

```
var counter = 0;
document.querySelector('body').addEventListener('click', function (event) {
    document.querySelector('body').innerHTML = ++counter;
});

document.querySelector('body').addEventListener('mousemove', function (event) {
    var $point = document.createElement('div');
    $point.style.background = 'red';
    $point.style.width = '10px';
    $point.style.height = '10px';
    $point.style.position = 'fixed';
    $point.style.left = event.clientX + 'px';
    $point.style.top = event.clientY + 'px';
    document.querySelector('body').appendChild($point);
    setTimeout(function () {
        $point.remove();
    }, 5000);
});
```

10 Lien utiles

La bible :

<https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript>

Un youtubeur vraiment compétent :

<https://www.grafikart.fr/formations/debuter-javascript>

La doc de Mozilla :

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps

11 Frameworks



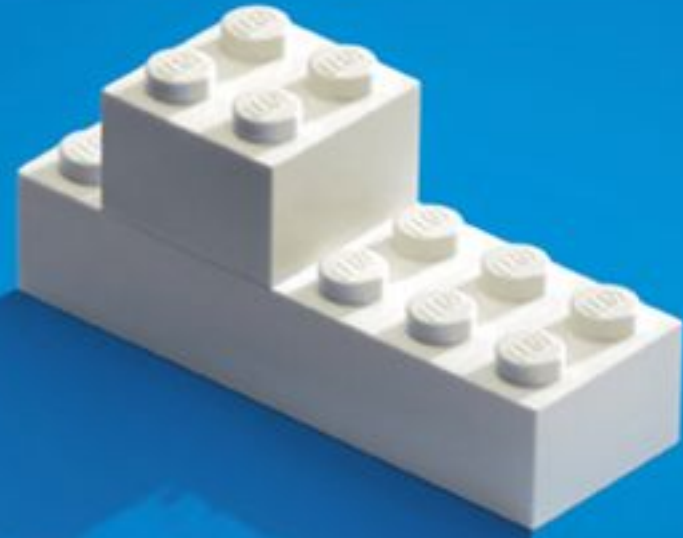


13 ANGULAR



14

WEB COMPONENTS

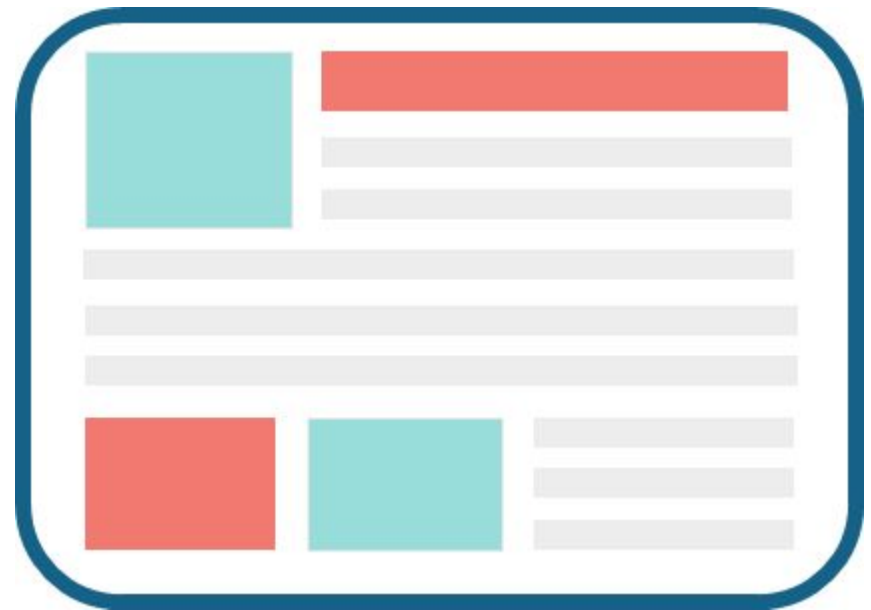


15 WEB COMPONENTS

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 ?

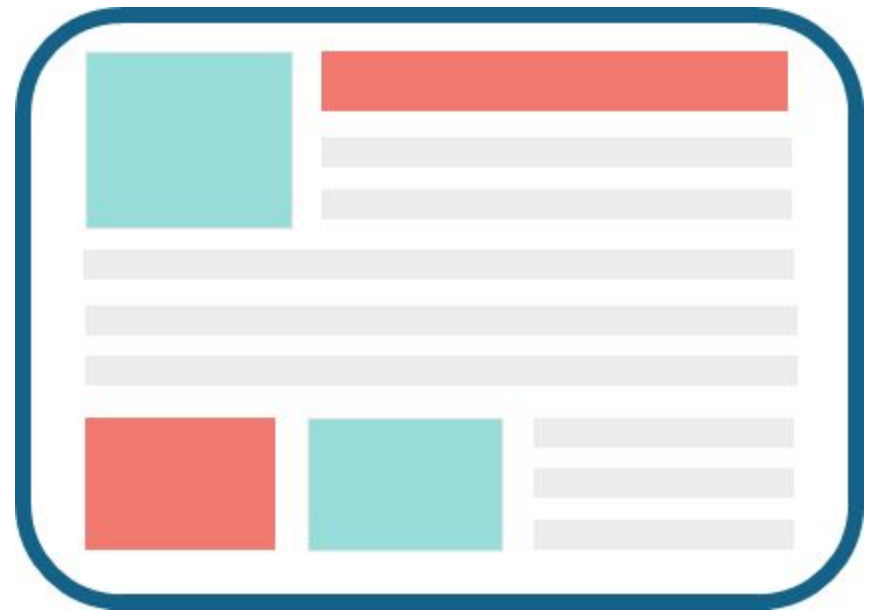
Duplication de code,
difficile à modifier et à
maintenir



18 WEB COMPONENTS

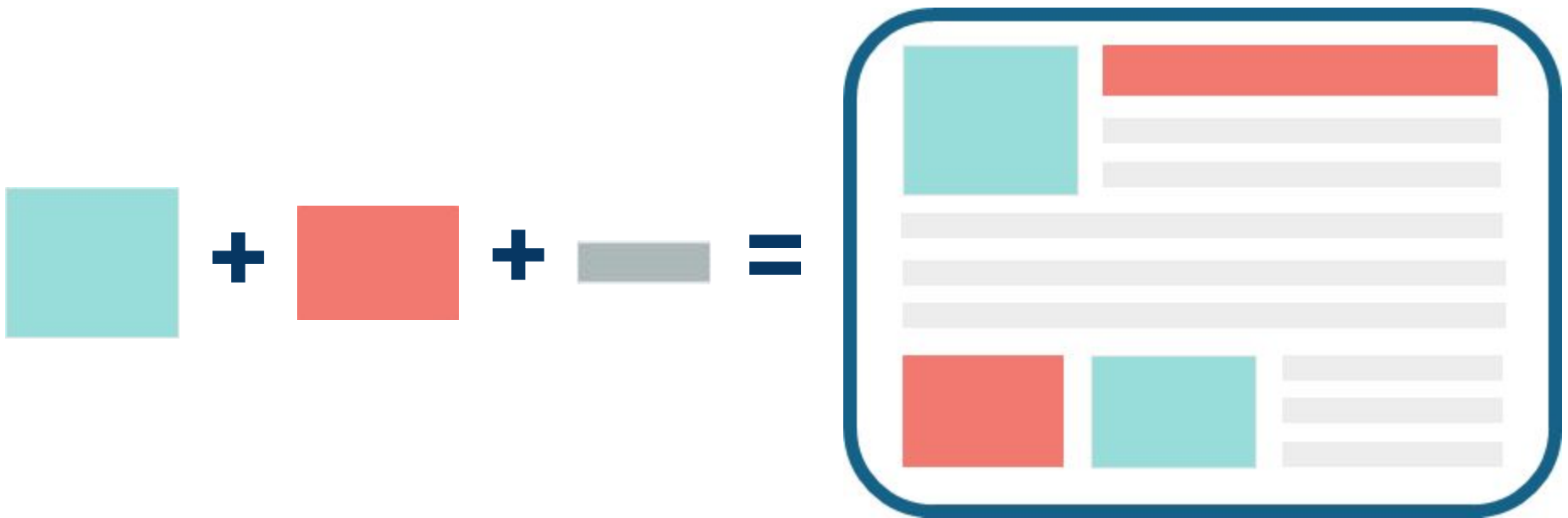
Approche modulaire

**Comment découper
cette page ?**



19 WEB COMPONENTS

Approche modulaire



20 WEB COMPONENTS

Approche modulaire

Et en vrai,
on fait comment?



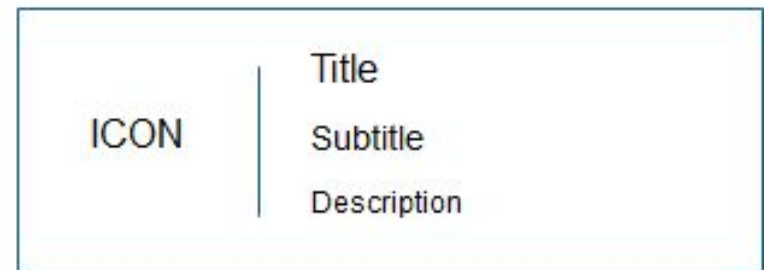
21 WEB COMPONENTS

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

```
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() { }  
}
```

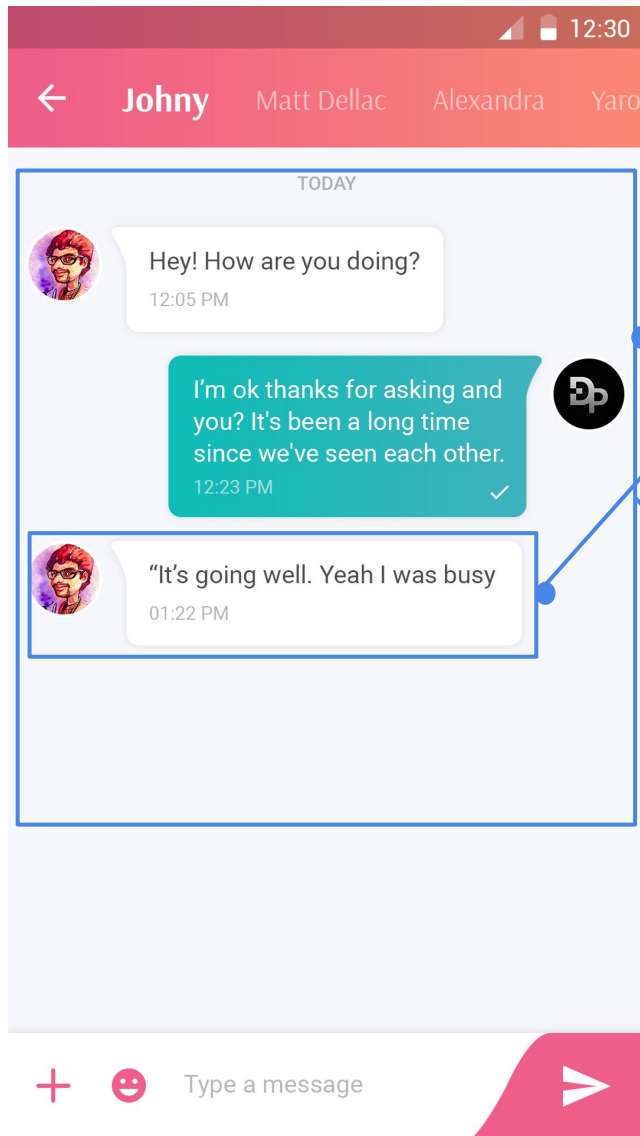
```
<h2>{{title}}</h2>  
<div class="description">  
  {{description}}  
</div>
```

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

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

```
.description {  
  color: lightgray;  
}
```

28 ANGULAR CONCEPTS



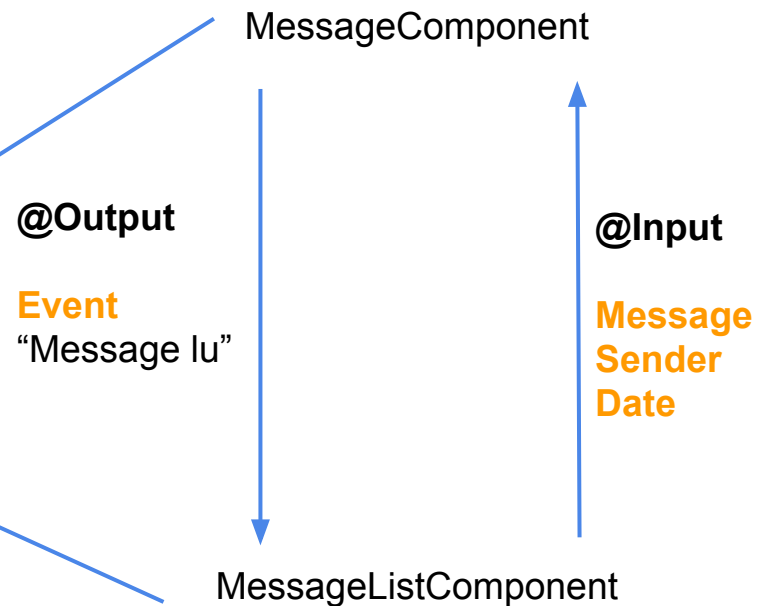
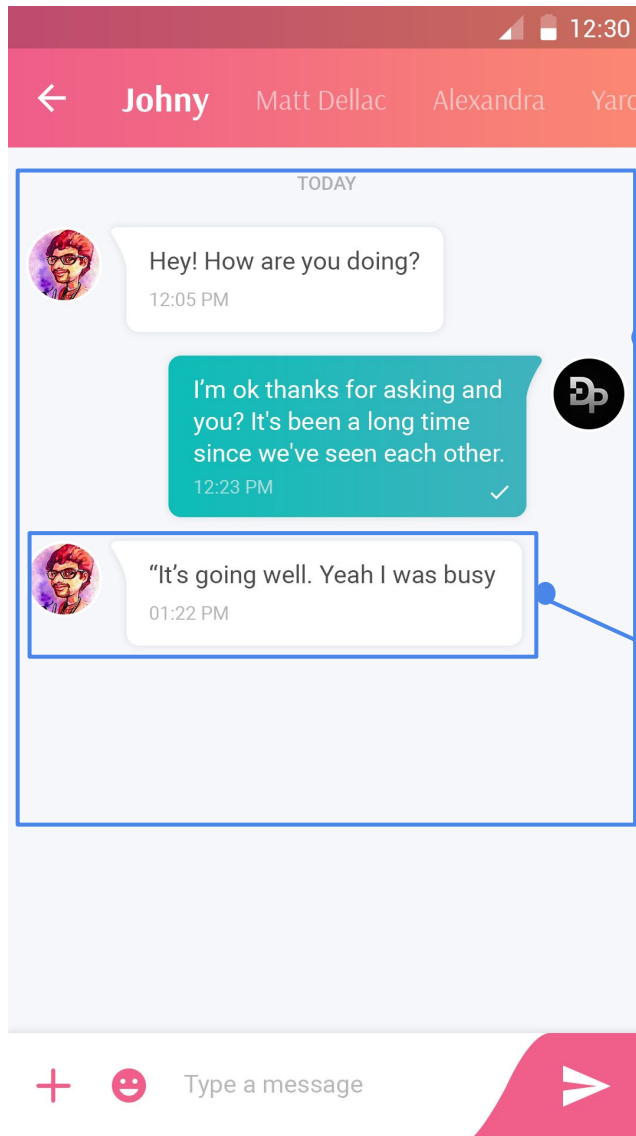
MessageComponent

MessageListComponent

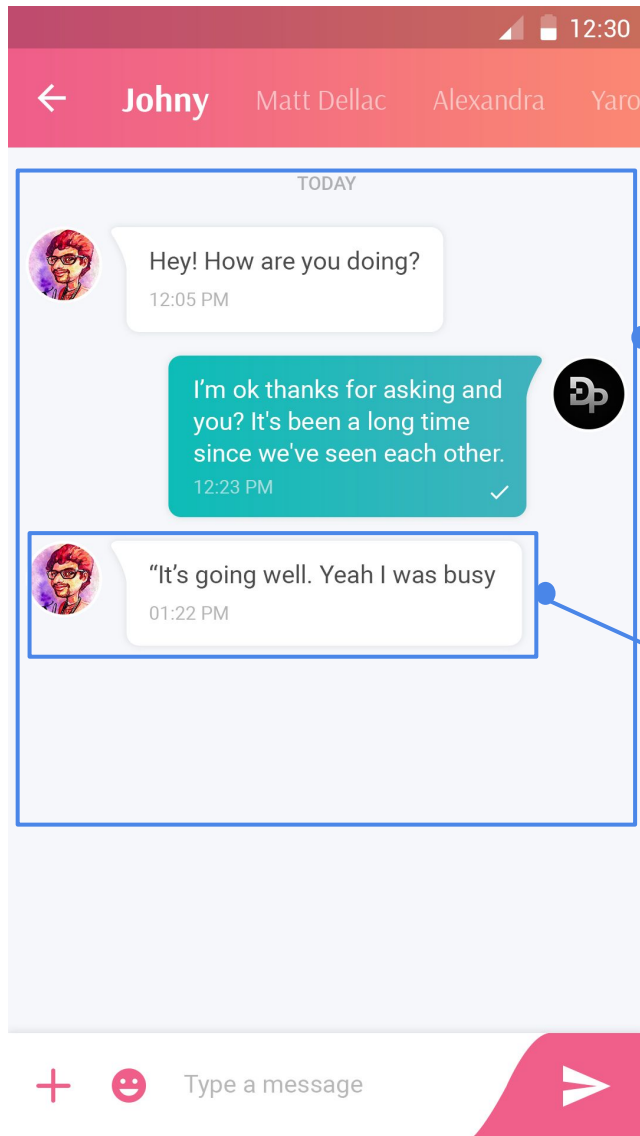
29

ANGULAR CONCEPTS

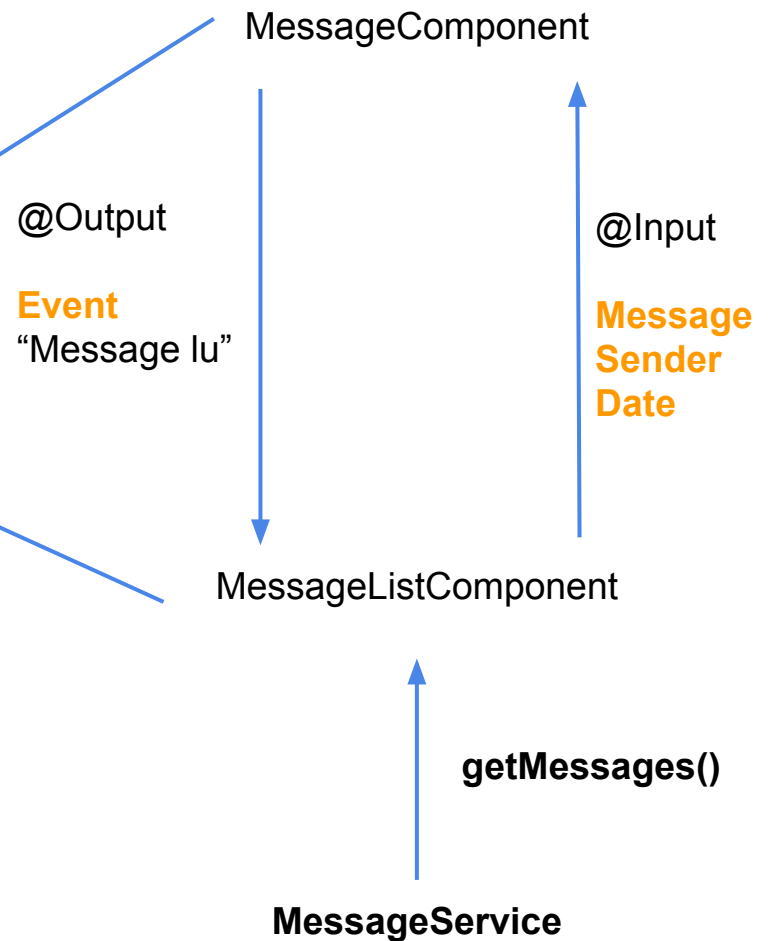
Input & Output

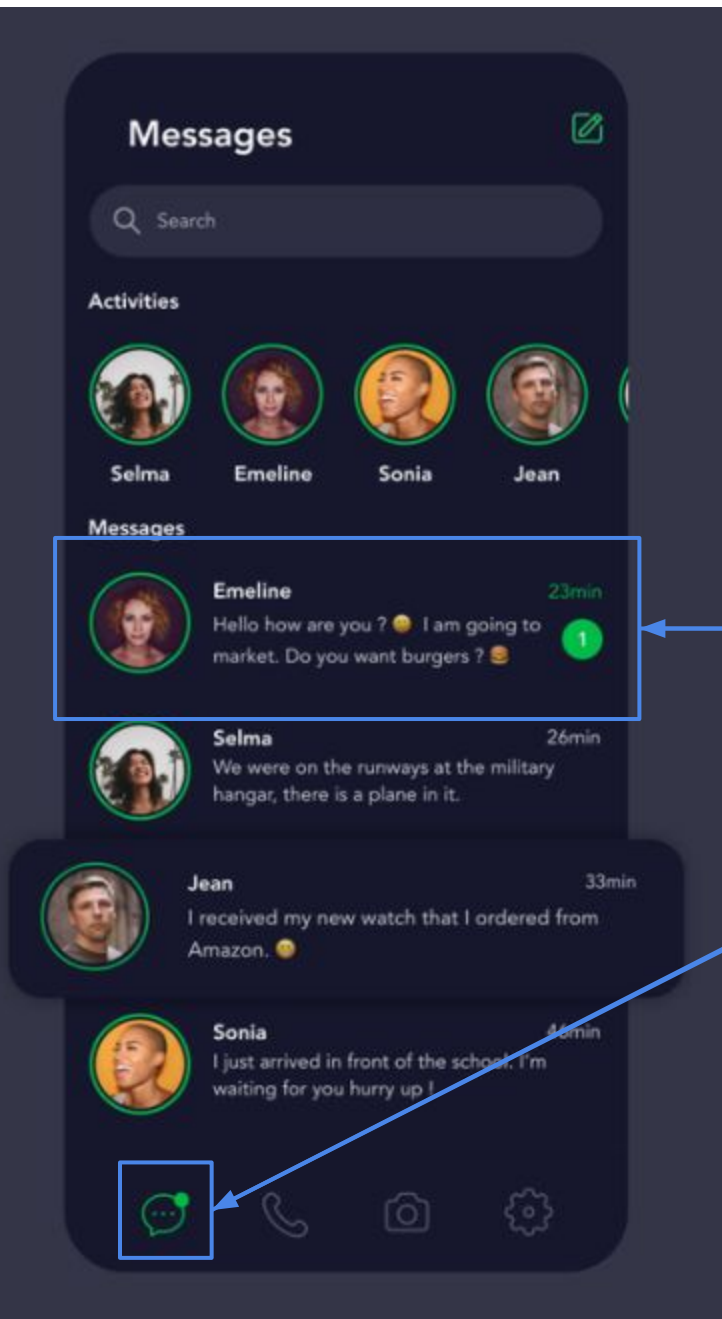


30 ANGULAR CONCEPTS



Services





Services

MessageService

32 ANGULAR CONCEPTS

SERVICE

- _ Contient un ensemble de fonctions partagées par plusieurs composants
- _ Les fonctionnalités du service doivent être uniques
- _ Tous les composants peuvent utiliser un service



```
@Injectable()
export class LoginService {

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

}
```


33 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

34 ANGULAR CONCEPTS

DIRECTIVES NATIVES

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



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

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



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

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

Chat

Do something ...

- hello!
- How are you?

35

Handle async

36 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);
});
```

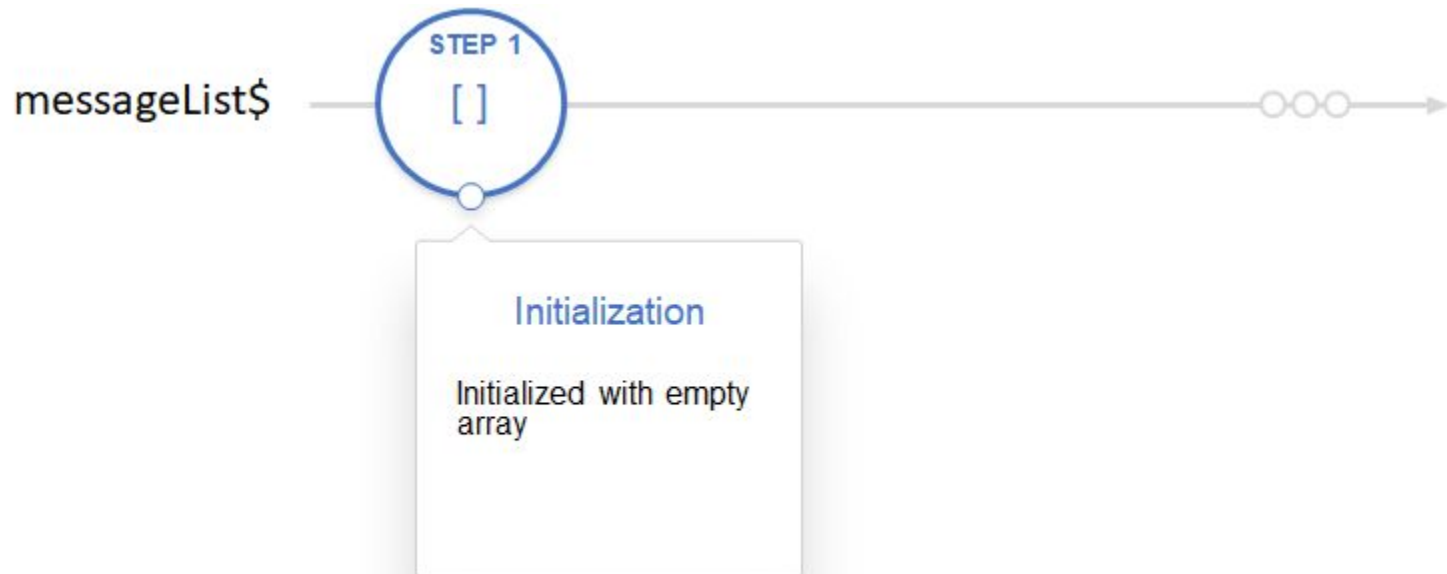
37 JS : promises

```
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);  
});
```

38 JS : async/await

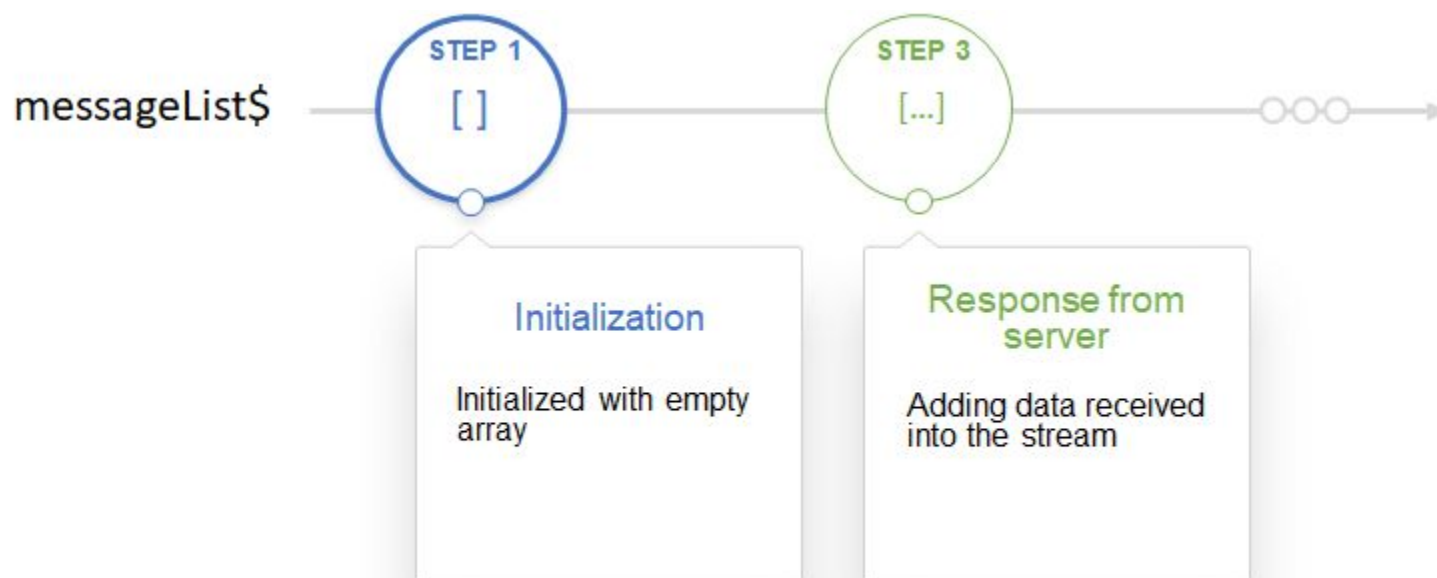
```
async function foo() {  
  var res1 = await request.getAsync(url1);  
  var res2 = await request.getAsync(url2);  
  var res3 = await request.getAsync(url3);  
  var res4 = await request.getAsync(url4);  
  return "whew all done";  
}  
  
// use that function somewhere  
foo().then(function(message) {  
  console.log("success!", message);  
}).catch(function(err) {  
  console.log("error!", err);  
});
```

39 Observables



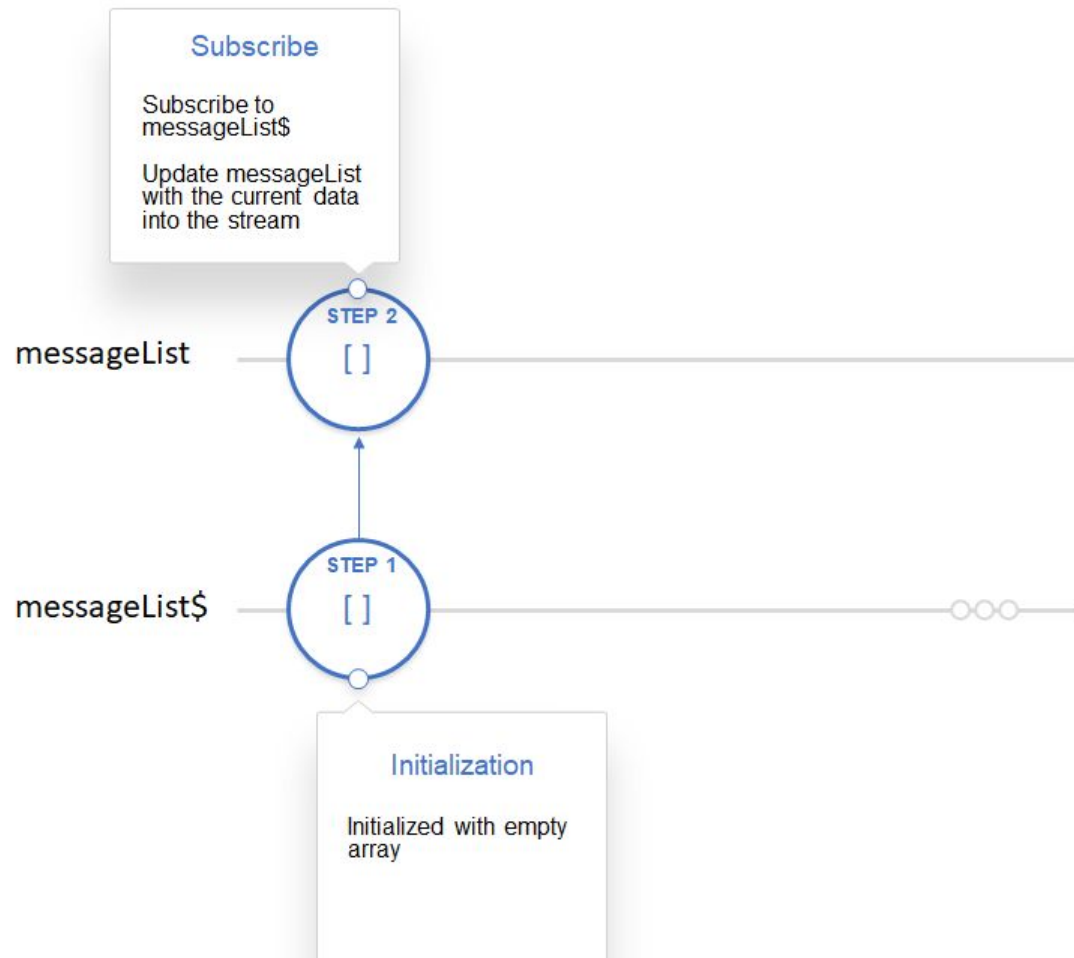
4

Observables

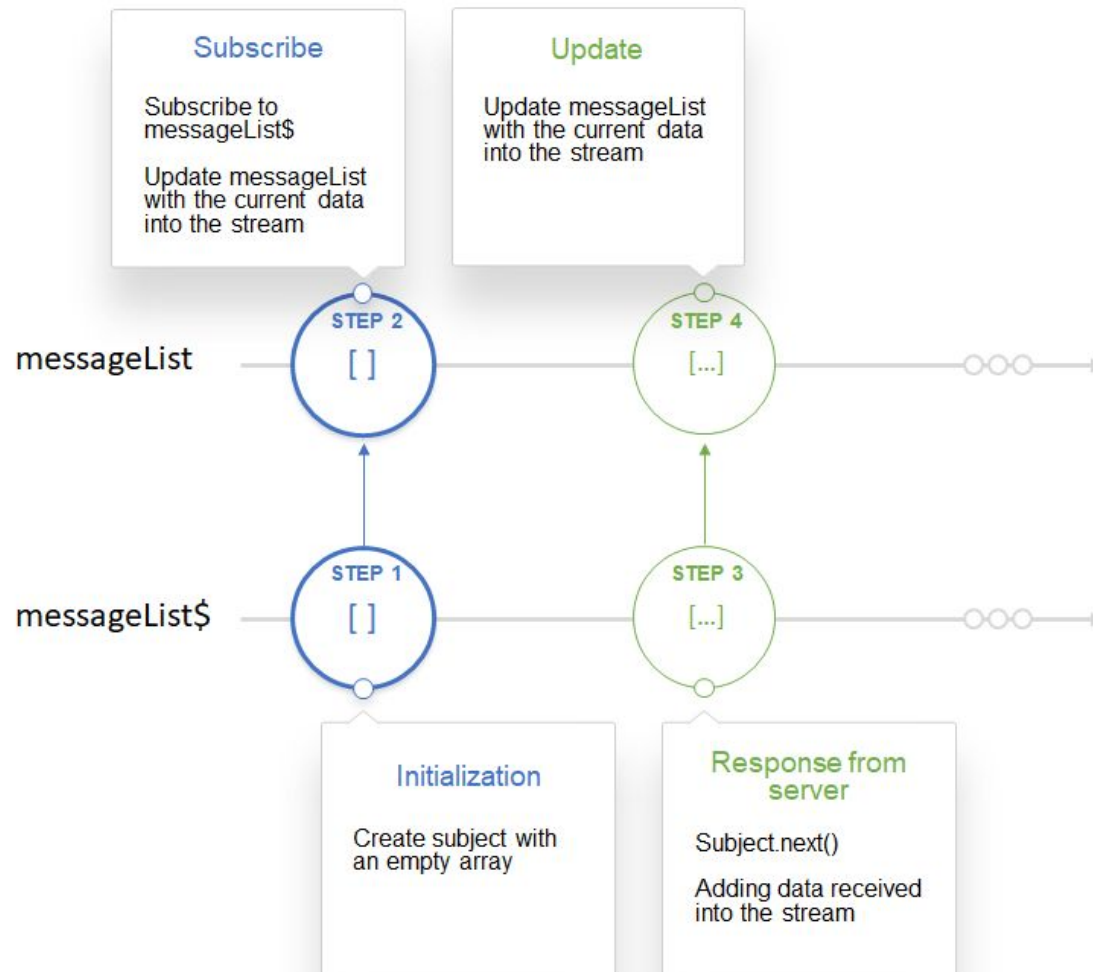


41

42 Observables

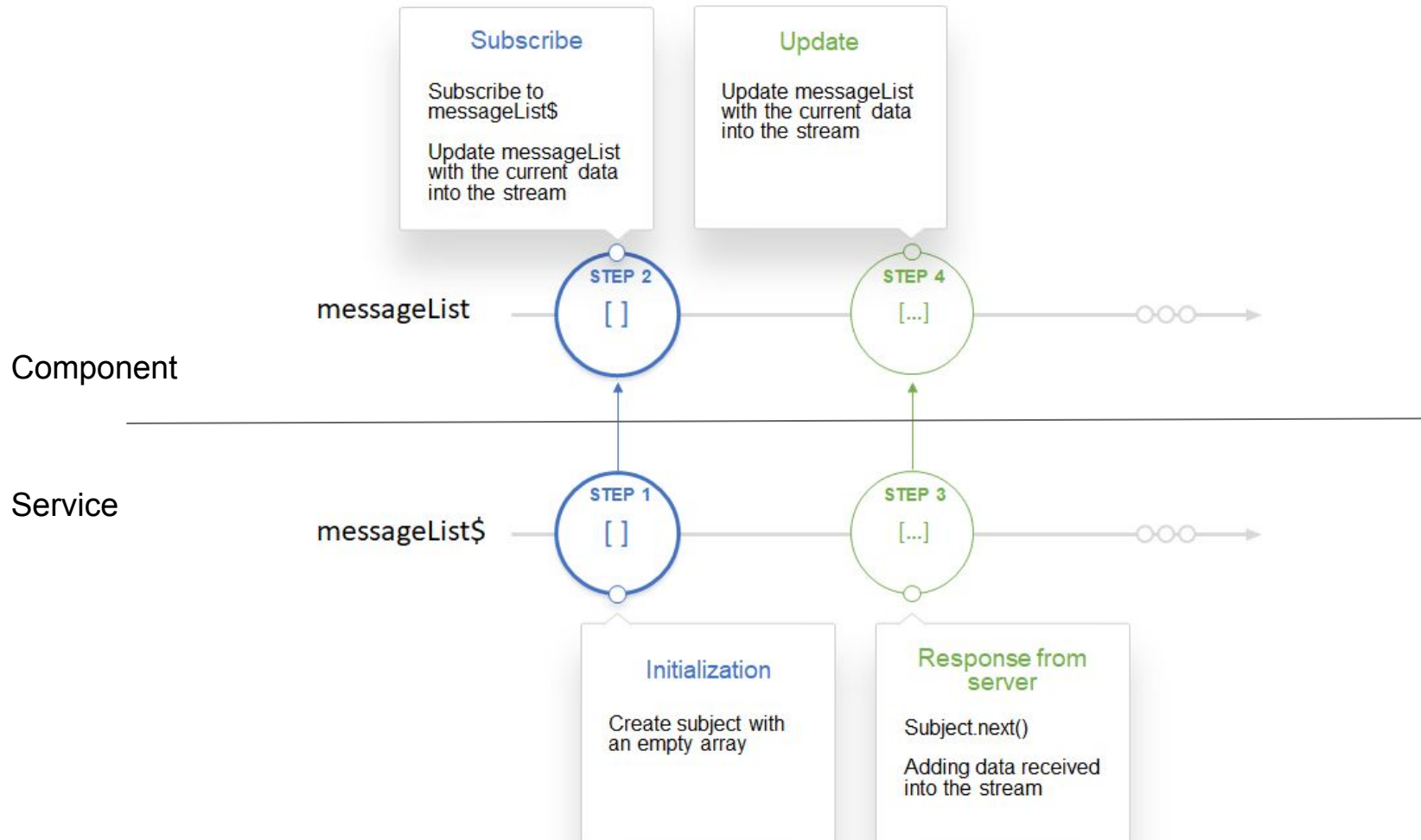


43 Observables



4

Observables



45 Observables

Service

```
@Injectable()
export class MessageService {

  public messageList$: BehaviorSubject<MessageModel[]>;

  constructor() {
    this.messageList$.next( value: []); 1
  }

  updateMessageList(messageList: MessageModel[]) {
    this.messageList$.next(messageList); 3
  }
}
```

Component

```
export class MessageListComponent implements OnInit {

  public messageList: MessageModel[];

  constructor(private messageService: MessageService) { 2
    this.messageService.messageList$.subscribe(
      next: (messageList) => this.messageList = messageList
    );
  }

  ngOnInit() { }
}
```

46



QUESTIONS?

2

