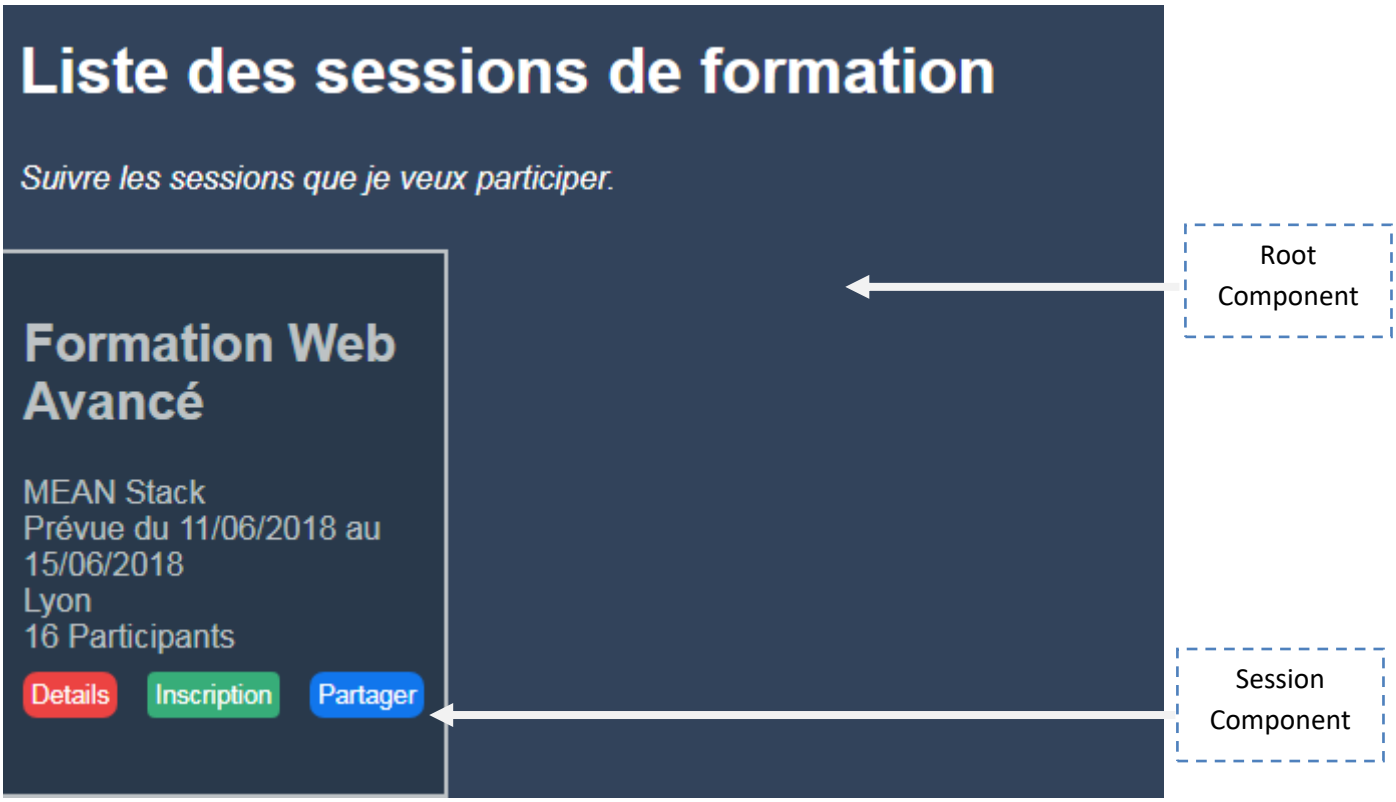
TP1: Framework Angular (Components ET Data Binding)

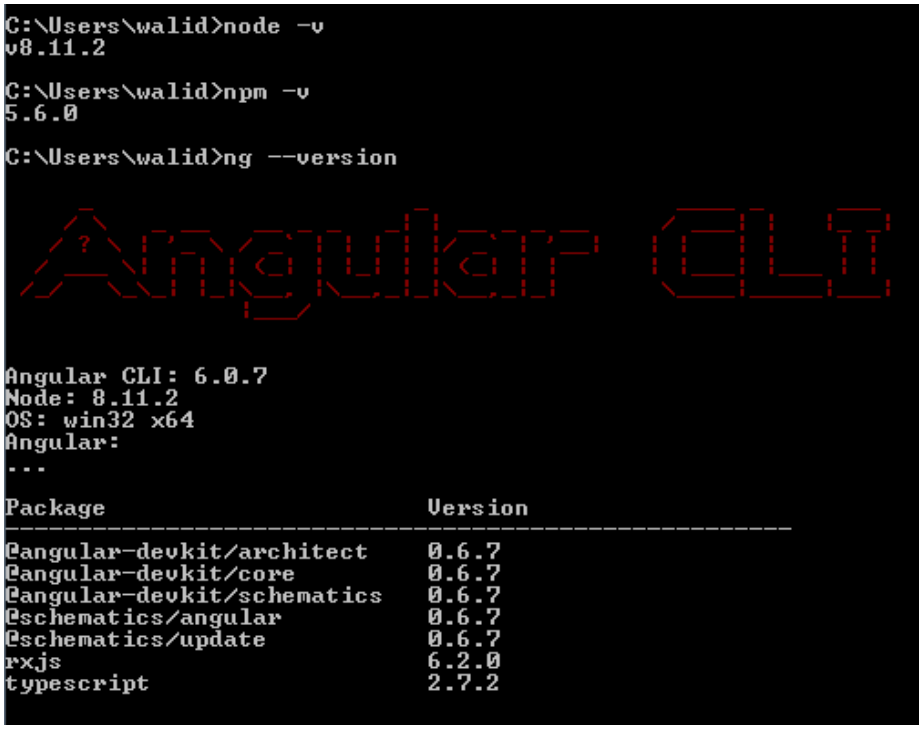
On vous demande de développer une application Angular permettant de gérer les sessions des formations en ligne en développement web et mobile (sous forme de *webinar*). Cette application est illustrée par la figure suivante :



Etapes d'implémentations

Partie 1 : Angular-CLI

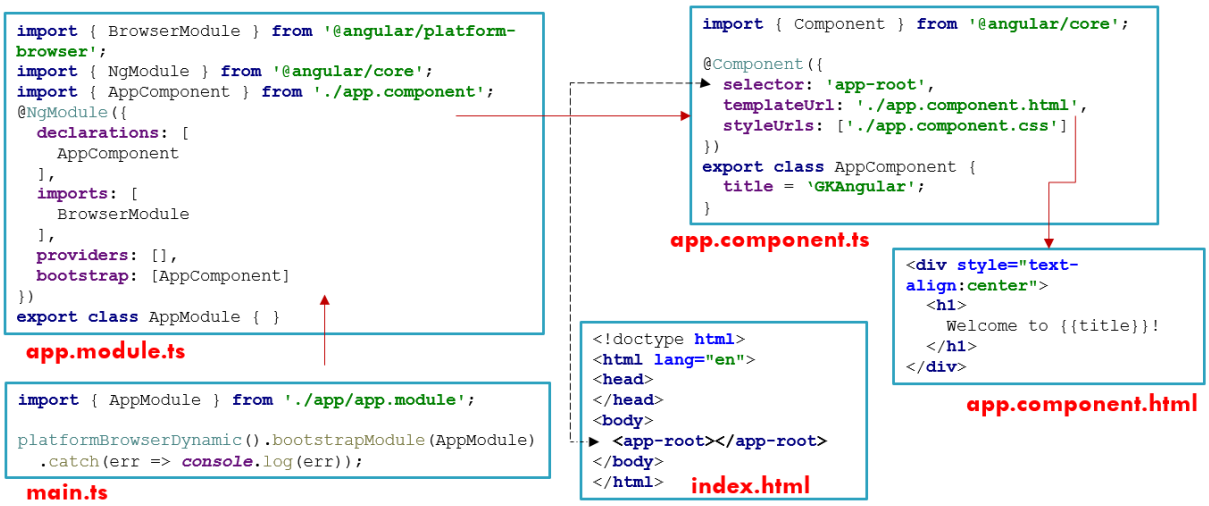
1. Procéder par la mise en place de l'environnement Angular :
   a. Installer NodeJS pour qu'on puisse utiliser le gestionnaire des modules npm : https://nodejs.org/en/download/
   b. Installer Angular-CLI en utilisant la commande npm :
      https://angular.io/guide/quickstart
      npm install -g @angular/cli



2. En utilisant Angular-CLI, créer une nouvelle application appelée « training-app »
   $ ng new training-app
   $ cd training-app
   $ ng serve
3. Parcourir l'arborescence de l'application afin de connaitre la structure d'une application Angular ainsi que ses différents éléments (Composants, Templates, Modules, fichiers de configuration, etc.)
4. Déposer votre application dans un dépôt Github

Partie 2 : Components

1. A l'aide d'un schéma, donner la mécanique de lancement de l'application (Amorçage de l'application ou bien le Bootstrap)



2. Modifier le style du composant AppComponent par le code CSS suivant :

```
styles: [`
          section {
            width: 100%;
            background-color: #32435b;
          }
          h1{
              Color : #ffffff;
          }`,
          `.description{
            Font-style: italic ;
            Color : #ffffff;
            }
`]
```

3. Modifier le Template du composant AppCompnent par le code HTML suivant :

```
<section>
  <header>
    <h1>Liste des sessions de formation </h1>
    <p class="description">Suivre les sessions que je veux participer.</p>
  </header>
</section>
```

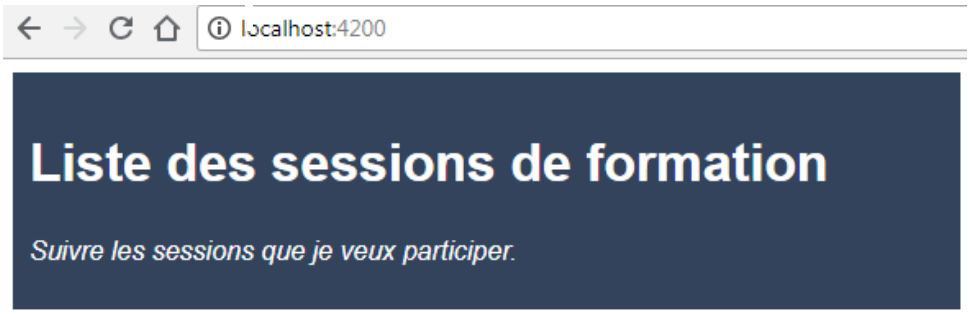
4. Saisir le code CSS suivant dans le fichier app.component.css

```
:host {
  display: flex;
  font-family: Arial, Helvetica, sans-serif;
```

```
}
section {
  width: 100%;
  background-color: #32435b;
}
section > header {
  color: #ffffff;
  padding: 10px;
}
section > header > h1 {
  font-size: 2em;
}
section > header .description {
  font-style: italic;
}
```

5. Dans le fichier *app.component .ts*, modifier la propriété styleUrls du composant AppComponent de la manière suivante :

```
@Component({
  styleUrls: ['./app.component.css']
})
```



6. Ajouter un nouveau composant « sesseion-item » avec les fichiers template/css suivants :
   a. Template HTML

```
<h2>Formation Programmation Web</h2>
<div>MEAN Stack</div>
<div>Prévue du 11/06/2018 au 15/06/2018</div>
<div>Lyon</div>
<div class="tools">
  <a class="delete">
    Details
  </a>
  <a class="inscrire">
    Inscription
  </a>
  <a class="partager">
    Partager
  </a>
</div>
```

   b. Style CSS

```
:host {
  display: flex;
  flex-direction: column;
  width: 200px;
  height: 280px;
  border: 2px solid;
  background-color: #29394b;
  padding: 10px;
  color: #bdc2c5;
}
h2 {
  font-size: 1.6em;
}
:host.session-web {
  border-color: #53ace4;
}
:host.session-web > h2 {
  color: #53ace4;
}
:host.session-mobile {
  border-color: #45bf94;
}
:host.session-mobile > h2 {
  color: #45bf94;
}
.tools {
  margin-top: 8px;
  display: flex;
  flex-wrap: nowrap;
  justify-content: space-between;
}
.partager {
  display: block;
  background-color: #1176ec;
  padding: 4px;
  font-size: .8em;
  border-radius: 7px;
  color: #ffffff;
  cursor: pointer;
}
.delete {
  display: block;
  background-color: #ec4342;
  padding: 4px;
  font-size: .8em;
  border-radius: 7px;
  color: #ffffff;
  cursor: pointer;
}
.inscrire {
  display: block;
  background-color: #37ad79;
  padding: 4px;
  font-size: .8em;
  border-radius: 4px;
  color: #ffffff;
  cursor: pointer;
}
```

7. Tester le composant « session-item » en utilisant un appel imbriqué dans le composant AppComponent :

```
<section>
  <header>
    <h1>Liste des sessions de formation </h1>
    <p class="description">Suivre les sessions que je veux participer.</p>
  </header>
  <app-session-item></app-session-item>
  <app-session-item></app-session-item>
</section>
```

Partie 3 : Data Binding

1. Cette question s'intéresse au Data Binding (**technique d'interpolation**).
   a. Ajouter la variable name dans le composant « session-item »

```
export class SessionItemComponent implements OnInit {
name = 'Formation Web';
```

   b. Dans le template HTML (DOM), faites appel à cette variable en utilisant la technique d'interpolation.

```
<h2>{{name}}</h2>
<div>MEAN Stack</div>
<div>Prévue du 11/06/2018 au 15/06/2018</div>
<div>Lyon</div>
```

2. Cette question s'intéresse au Data Binding (**technique Property Binding**).
   a. Le Property Binding sur **un élément du DOM** :
      i. Document HTML (session-item.component.html)

```
<h2 [textContent]="name"></h2>
<h2 textContent="{{name}}"></h2>
<div align="{{alignement}}">MEAN Stack</div>
<div [align]="alignement">MEAN Stack</div
```

      ii. Composant SessionItemComponent (session-item.component.ts)

```
export class SessionItemComponent implements OnInit {
//name = 'Formation Web';
alignement = 'right';
couleur = 'red';
```

b. Le Property Binding sur **un attribut directive**

i. Document HTML (session-item.component.html)

```
<div [align]="alignement" [ngStyle]="{color:couleur}">MEAN Stack</div>
```

Commentaires :

Le Data Binding va remplacer la variable couleur dans le template par sa valeur 'red' définie dans le Component.

c. Le Property Binding sur **une propriété d'un component** (Exemple 1)

i. Composant AppComponent (app.component.ts)

```
8  export class AppComponent {
9    session_name = 'Formation Web';
```

ii. Document HTML (app.component.html)

```
1  <section>
2    <header>
3      <h1>Liste des sessions de formation </h1>
4      <p class="description">Suivre les sessions que je veux participer.</p>
5    </header>
6    <app-session-item [name]="session_name"></app-session-item>
7  </section>
```

iii. Composant SessionItemComponent (session-item.component.ts)

```
8   export class SessionItemComponent implements OnInit {
9   //name = 'Formation Web';
10  alignement = 'right';
11  couleur = 'red';
12  @Input() name: string;
```

iv. Document HTML (session-item.component.html)

```
1   <h2>{{name}}</h2>
```

Commentaires :

Grâce au décorateur @Input, le template de SessionItemComponent affiche la propriété *name* sans problème. Il faut importer la classe Input.

d. Le Property Binding sur **une propriété d'un component** (Exemple 2)

i. Composant AppComponent (app.component.ts)

```
export class AppComponent {
  //session_name = 'Formation Web';
  nbpart: number = 0;
   firstSession = {
    id: 1,
    name: 'Formation Web',
    track: 'MEAN Stack',
    date: 'Prévue du 11/06/2018 au 15/06/2018',
    local: 'Lyon',
    participants: 0
  };
```

ii. Document HTML (app.component.html)

```
<section>
  <header>
    <h1>Liste des sessions de formation </h1>
    <p class="description">Suivre les sessions que je veux participer.</p>
</header><app-session-item [session]="firstSession"></app-session-item>
</section>
```

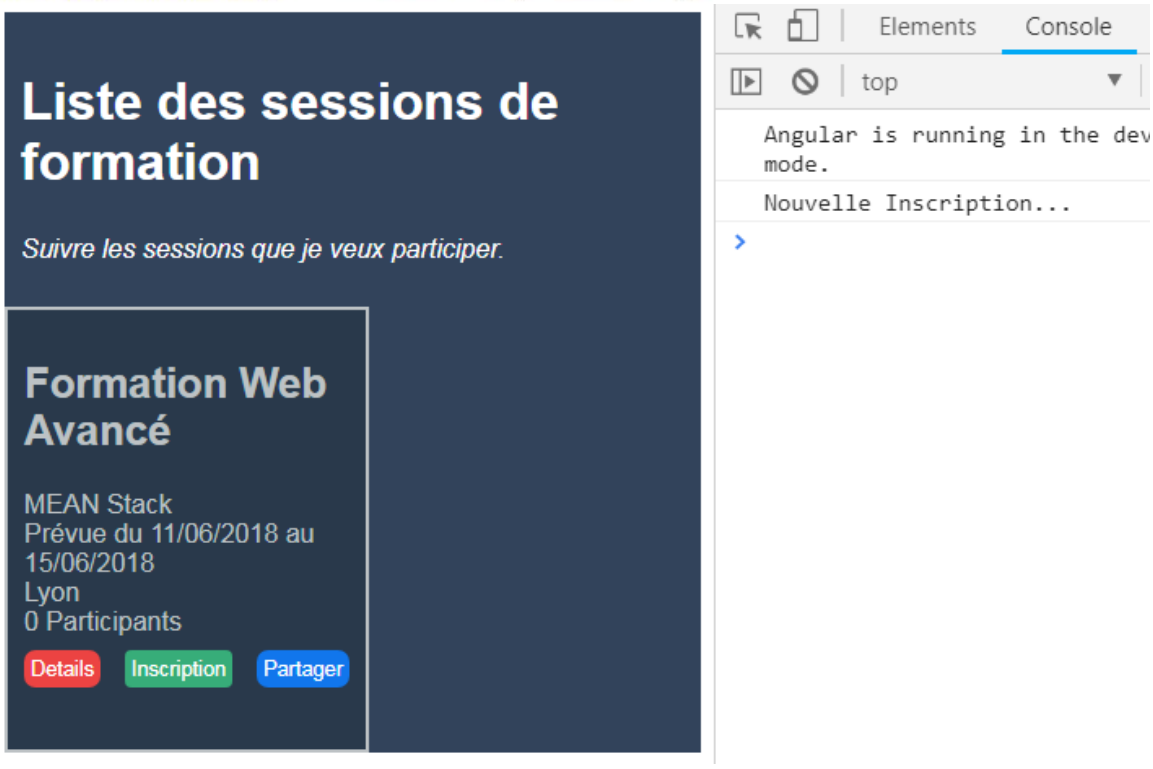
iii. Composant SessionItemComponent (session-item.component.ts)

```
export class SessionItemComponent implements OnInit {
//name = 'Formation Web';
alignement = 'right';
couleur = 'red';
@Input() session: any;
```

iv. Document HTML (session-item.component.html)

```
<h2>{{session.name}}</h2>
<div>{{session.track}}</div>
<div>{{session.date}}</div>
<div>{{session.local}}</div>
<div>{{session.participants}} Participants</div>
```

3. Cette question s'intéresse au Data Binding (**technique Event Binding**). Le clic sur le lien *Inscription* permet d'appeler la méthode inscrire() qui modifiera la propriété *name* par la chaine « Formation Web Avancé » en affichant le message « Nouvelle Inscription sur le console ».



i. Composant SessionItemComponent (session-item.component.ts)

```
18  inscrire() {
19    console.log('Nouvelle Inscription...');
20    this.session.name = 'Formation Web Avancé';
21      }
```

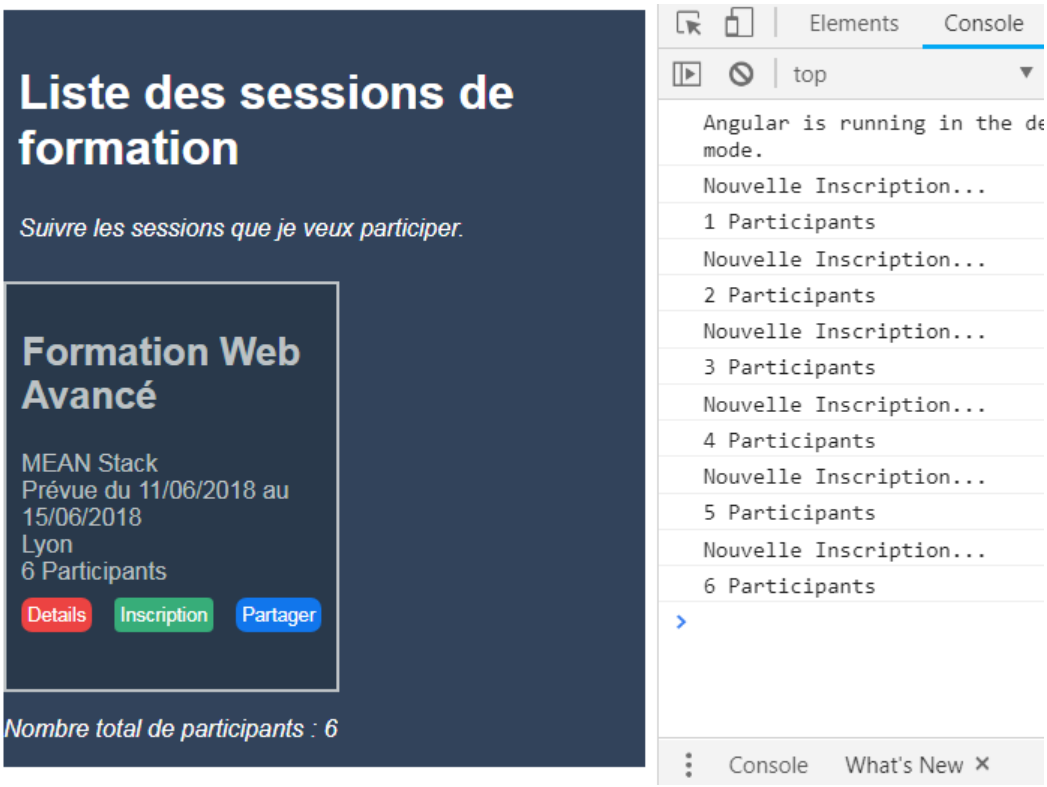
ii. Document HTML (session-item.component.html)

```
11    <a class="inscrire" (click)="inscrire()" >
12      Inscription
13    </a>
```

Commentaires :

Avec ce mécanisme, vous pouvez être averti des évènements utilisateurs tels que le click, la frappe sur le clavier, le touch, etc.
Par exemple, le clic sur le lien *Inscription* permet d'appeler la méthode inscrire() qui modifiera la propriété *name*. On aura donc une interaction dans le DOM qui modifiera le modèle.

4. Cette question s'intéresse au Data Binding (**technique Two-way Binding**). Le clic sur le lien Inscription permettra d'incrémenter le nombre des participants.



i. Composant SessionItemComponent (session-item.component.ts)

```typescript
import { Component, OnInit, Input, Output, EventEmitter} from '@angular/core';

@Component({
  selector: 'app-session-item',
  templateUrl: './session-item.component.html',
  styleUrls: ['./session-item.component.css']
})
export class SessionItemComponent implements OnInit {

alignement = 'right';
couleur = 'red';
@Input() session: any;
@Output() participantsChange = new EventEmitter<any>();
  constructor() { }

  ngOnInit() {
  }
inscrire() {
    console.log('Nouvelle Inscription...');
    this.session.name = 'Formation Web Avancé';
    this.session.participants = +this.session.participants + 1;
    console.log(this.session.participants + ' Participants');
    this.participantsChange.emit({
        value: this.session.participants
    });
  }
}
```

ii. Document HTML (app.component.html)

```html
<section>
  <header>
    <h1>Liste des sessions de formation </h1>
    <p class="description">Suivre les sessions que je veux participer.</p>
  </header>
  <app-session-item [session]="firstSession"  (participantsChange)="nbrParticipantsChange($event)"></app-session-item>
 <p class="description">Nombre total de participants : {{nbpart}}</p>
</section>

```

iii. AppComponent (app.component.ts)

```typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
  nbpart: number = 0;
   firstSession = {
    id: 1,
    name: 'Formation Web',
    track: 'MEAN Stack',
    date: 'Prévue du 11/06/2018 au 15/06/2018',
    local: 'Lyon',
    participants: 0
  };
  nbrParticipantsChange(event)
  {
   this.nbpart =  event.value;
  }
}
```

Commentaires : On déduit que le code du composant SessionItemComponent aura en @Input nommé session (avec 0 Participants) et un @output nommé *participantsChange*. Cela constitue donc la règle du Data Binding 2-way : dans la notation [session], session est le modèle à modifier et *participantsChange* est l'évènement lancé lors d'une modification de la valeur de l'attribut participants (*session.participants*). Cet évènement sera donc intercepté par le composant parent AppComponent afin de récupérer cette nouvelle valeur.