Framework coté client : Angular

Atelier 2

Les étapes pour la gestion des tâches avec angular :

1. Création d'un component :

Crée un nouveau composant pour la gestion des tâches.

Commande :

```
ng generate component task
```

2. Afficher les données : data binding

- Dans le typeScript de task : définir les propriétés de task

Utilise l'interpolation pour afficher les données d'une tâche.

Dans le template HTML :

```
<h2>{{ task.title }}</h2>
<p>{{ task.description }}</p>
<p>Créée le : {{ task.dateCreation }}</p>
<p>Status : {{ task.status }}</p>
```

3. Event binding :

Ajoute un bouton pour changer le statut de la tâche lorsqu'il est cliqué.

HTML :

```
<button (click)="changeStatus(task)">Changer le
statut</button>
```

Dans le TypeScript :

```
changeStatus(task: Task): void {
  task.status = task.status === 'completed' ?
'pending' : 'completed';
}
```

4. Attribute binding :

Utilise le binding d'attribut pour gérer les classes CSS dynamiquement.

HTML :

```
<h2 [attr.title]="task.title">{{ task.title }}</h2>
```

5. Model : Task

Crée un modèle pour représenter une tâche.

TypeScript :

```
export class Task {
  id: number;
  title: string;
  description: string;
  dateCreation: Date;
  dateUpdate: Date;
  status: string;
}
```

Exemple d'utilisation :

```
task: Task = {
  id: 1,
  title: 'Nouvelle Tâche',
  description: 'Description de la tâche',
  dateCreation: new Date(),
  dateUpdate: new Date(),
  status: 'pending'
};
```

6. Condition d'affichage d'une tâche :

Affiche une tâche uniquement si elle est en statut "pending".

HTML :

```
<div *ngIf="task.status === 'pending'">
  <h2>{{ task.title }}</h2>
  <p>{{ task.description }}</p>
</div>
```

7. Afficher liste de tâches :

Utilise la directive `*ngFor` pour afficher une liste de tâches.

HTML :

```
<div *ngFor="let task of tasks">
  <h2>{{ task.title }}</h2>
  <p>{{ task.description }}</p>
</div>
```

Dans le TypeScript :

```
tasks: Task[] = [
  { id: 1, title: 'Tâche 1', description:
'Description 1', dateCreation: new Date(),
dateUpdate: new Date(), status: 'pending' },
  { id: 2, title: 'Tâche 2', description:
'Description 2', dateCreation: new Date(),
dateUpdate: new Date(), status: 'completed' },
];
```

8. Ajouter de style dynamique :

Change dynamiquement le style des tâches selon leur statut.

HTML :

```
<div [ngStyle]="{ 'background-color': task.status === 
'completed' ? 'green' : 'red' }">
  <h2>{{ task.title }}</h2>
  <p>{{ task.description }}</p>
</div>
```

9. Ajouter de class CSS :

Ajoute des classes CSS dynamiques en fonction du statut de la tâche.

HTML :

```
<div [ngClass]="{ 'completed-task': task.status === 
'completed', 'pending-task': task.status === 
'pending' }">
  <h2>{{ task.title }}</h2>
  <p>{{ task.description }}</p>
</div>
```

CSS :

```
.completed-task {
  color: white;
  background-color: green;
}
.pending-task {
  color: black;
  background-color: yellow;
}
```