

TP 3.2 – Angular

Introduction

L'exercice de programmation que l'on vous propose a pour objectif de vous permettre de consolider les premiers concepts présentés dans les premiers TD d'introduction à Angular.

Prérequis

- Environnement de travail opérationnel
 - Savoir créer un projet Angular avec des composants.
 - Avoir réalisé les TD jusqu'à 2.3
 - Avoir compris les modes d'échanges entre Parent et Enfant d'élément du DOM
- Voir : <https://angular.io/guide/inputs-outputs>

Objectif

On souhaite concevoir une application web qui affiche une table de multiplication (1 à 10), selon une valeur soumise par l'utilisateur.

Une table de multiplication affiche dans les lignes et colonnes le résultat de la multiplication des petits nombres entiers naturels. Le terme usité du Moyen Âge au XVIe siècle était « livret ».

— Table de multiplication https://fr.wikipedia.org/wiki/Table_de_multiplication

Le système de numération décimale de position permet d'effectuer la multiplication de deux nombres quelconques à l'aide de la seule connaissance des produits des nombres de 0 à 9 entre eux. C'est à l'école primaire que s'effectue l'apprentissage des tables qui récapitulent tous ces produits. La tradition a longtemps exigé la connaissance des tables de multiplication portant jusqu'à 12 ou 13 au lieu de 9. (source : wikipedia)

Durée moyenne prévue : 6h à 8h (rapport compris)

Informations

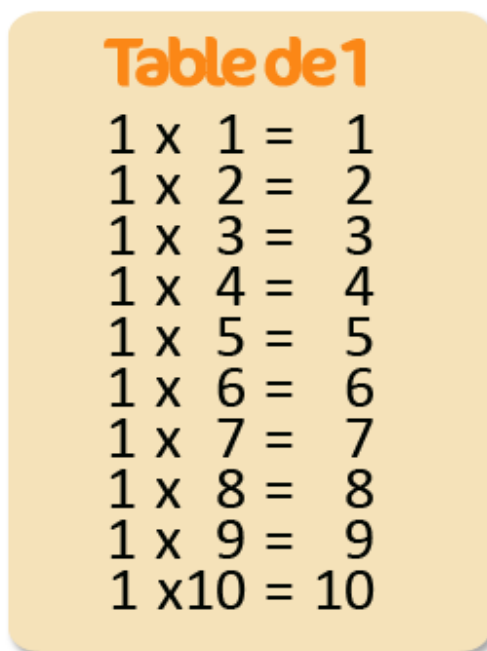
Partie 1

On se contentera d'une application **mono-page**. La page principale fera usage du composant **TableMultiplication** (à concevoir) dont le rôle est de présenter une table de multiplication.

Ce composant devra être paramétré : il exploite une valeur qui lui est transmise par son **parent**: un nombre entier qui correspond à la table de multiplication à afficher.

La page principale de l'application devra permettre à l'utilisateur de saisir une valeur pour l'affichage de la table de multiplication correspond. Voici une maquette :

Entrez un nombre : 1



1	x	1	=	1
1	x	2	=	2
1	x	3	=	3
1	x	4	=	4
1	x	5	=	5
1	x	6	=	6
1	x	7	=	7
1	x	8	=	8
1	x	9	=	9
1	x	10	=	10

TP 3.2 – Angular

Partie 2

Dans une seconde partie, la page principale présentera, en dessous du résultat précédent, les tables de multiplication (celle que l'on apprend en primaire). Voici une maquette :

Les tables de multiplication

Table de 1	Table de 2	Table de 3	Table de 4	Table de 5
1 x 1 = 1	2 x 1 = 2	3 x 1 = 3	4 x 1 = 4	5 x 1 = 5
1 x 2 = 2	2 x 2 = 4	3 x 2 = 6	4 x 2 = 8	5 x 2 = 10
1 x 3 = 3	2 x 3 = 6	3 x 3 = 9	4 x 3 = 12	5 x 3 = 15
1 x 4 = 4	2 x 4 = 8	3 x 4 = 12	4 x 4 = 16	5 x 4 = 20
1 x 5 = 5	2 x 5 = 10	3 x 5 = 15	4 x 5 = 20	5 x 5 = 25
1 x 6 = 6	2 x 6 = 12	3 x 6 = 18	4 x 6 = 24	5 x 6 = 30
1 x 7 = 7	2 x 7 = 14	3 x 7 = 21	4 x 7 = 28	5 x 7 = 35
1 x 8 = 8	2 x 8 = 16	3 x 8 = 24	4 x 8 = 32	5 x 8 = 40
1 x 9 = 9	2 x 9 = 18	3 x 9 = 27	4 x 9 = 36	5 x 9 = 45
1 x 10 = 10	2 x 10 = 20	3 x 10 = 30	4 x 10 = 40	5 x 10 = 50
Table de 6	Table de 7	Table de 8	Table de 9	Table de 10
6 x 1 = 6	7 x 1 = 7	8 x 1 = 8	9 x 1 = 9	10 x 1 = 10
6 x 2 = 12	7 x 2 = 14	8 x 2 = 16	9 x 2 = 18	10 x 2 = 20
6 x 3 = 18	7 x 3 = 21	8 x 3 = 24	9 x 3 = 27	10 x 3 = 30
6 x 4 = 24	7 x 4 = 28	8 x 4 = 32	9 x 4 = 36	10 x 4 = 40
6 x 5 = 30	7 x 5 = 35	8 x 5 = 40	9 x 5 = 45	10 x 5 = 50
6 x 6 = 36	7 x 6 = 42	8 x 6 = 48	9 x 6 = 54	10 x 6 = 60
6 x 7 = 42	7 x 7 = 49	8 x 7 = 56	9 x 7 = 63	10 x 7 = 70
6 x 8 = 48	7 x 8 = 56	8 x 8 = 64	9 x 8 = 72	10 x 8 = 80
6 x 9 = 54	7 x 9 = 63	8 x 9 = 72	9 x 9 = 81	10 x 9 = 90
6 x 10 = 60	7 x 10 = 70	8 x 10 = 80	9 x 10 = 90	10 x 10 = 100

On attend la création d'un nouveau composant, que vous nommerez **TablesMultiplication** (à concevoir). Ce composant sera également paramétré : Il recevra un nombre entier (valeur transmise par son parent) : En effet l'utilisateur aura la possibilité de saisir le nombre de tables à afficher (10 par défaut)

Note : le travail demandé peut être réalisé sans création de nouveaux composants, mais ce n'est pas l'objectif de ce TP.

Ce qui est attendu : Deadline : **Mardi 16 novembre** - travail individuel de préférence

- 1] Concevoir l'application Web à l'aide du framework Angular, en respectant les noms des composants attendus.
- 2] Concevoir un **rapport détaillé** sous la forme d'un **README** et diagramme de classe UML (voir plus loin)
- 3] Pousser votre application sur un dépôt distant (GitLab, GitHub...)
- 4] Communiquer par mail une version .pdf de votre rapport à vos professeurs (rappel, votre rapport contiendra un lien vers votre dépôt)

IMPORTANT : Dans votre README, la teneur des explications sera l'objet d'évaluation. Ce rapport devra intégrer un diagramme de classes UML des principales classes .ts de votre application. Pour cela vous utiliserez la solution *Plantuml*.

Pour une intégration dans VSCode, voir le blog d'Andrea :

<https://blog.anoff.io/2018-07-31-diagrams-with-plantuml/>

Comme expliqué dans cet article, dans Preferences/Settings pensez à choisir comment rendre le serveur en ligne de PlantUML :

```
// PlantUMLServer: Render diagrams by server which is specified with
// "plantuml.server". It's much faster, but requires a server.
// Local is the default configuration.
```

```
"plantuml.render": "PlantUMLServer",
```

```
// Plantuml server to generate UML diagrams on-the-fly.
```

```
"plantuml.server": "https://www.plantuml.com/plantuml",
```

Vous pouvez également, ou conjointement, utiliser plantuml en ligne :
<http://www.plantuml.com/plantuml/uml/> (génération d'une image à intégrer)

Syntaxe : <https://plantuml.com/fr/class-diagram>