# TypeScript

day

Sujet
Dans un premier temps nous vous proposons le développement d'un jeu du pendu en version simplifié. On vous donne une structure de fichiers et dossiers et du code pour commencer celui-ci.

Le jeu consistera à deviner un mot partiellement caché en un minimum de 7 coups.

Créez une constante enum pour la gestion du Status dans le jeu ; définissez par exemple 3 constantes : Progress, Loser et Winner.

Dans cet exercice nous utiliserons Node pour rendre interactif le jeu en console.

En fin de document vous avez un exemple détaillé de fonctionnement de ce jeu.

On propose, si vous avez le temps, un exercice facultatif plus complet et donc plus complexe à réaliser à la fin de ce document.

**Prérequis techniques : Node et npm doivent être installés sur vos machines.**

## Mise en place de l'exercice

Vous devez avoir le module TypeScript d'installé sur vos machines, si ce n'est pas déjà fait taper la ligne suivante :

```
npm install -g typescript
```

Récupérez les sources pour commencer le jeu et tapez dans la console (nous installons les définitions de types Node pour TypeScript) :

```
npm install
```

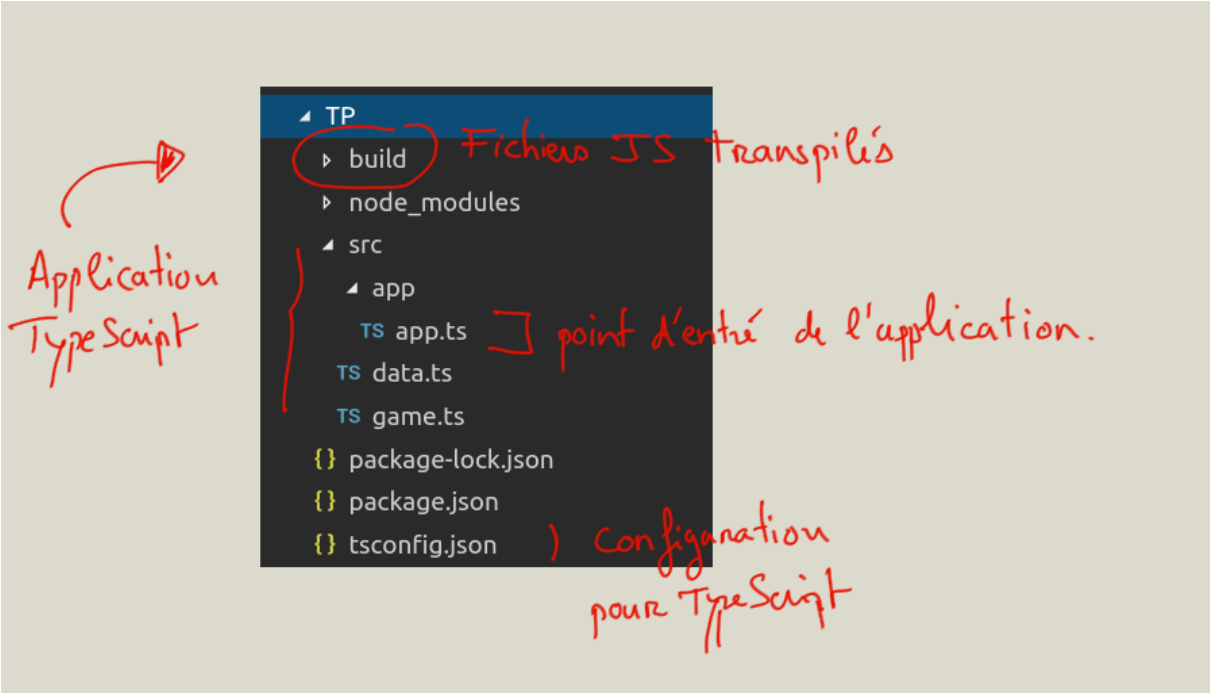
Une commande permet également d'installer cette dépendance, dans le cas où vous auriez un fichier package.json :

```
npm install --save @types/node
```

Pour compiler (transpiler) le code vous devez utiliser la commande suivante :

```
tsc -w
```

**Présentation de la structure des dossiers et fichiers**




Dans le fichier app.ts dans les sources nous vous avons écrit la partie flux à vous de mettre toute la logique dans le jeu.

**Comment interagir avec la console**

Voyez le code dans le fichier app.ts du dossier source pour commencer le jeu.

### Les données du jeu et définition de classe

La classe Word permet de définir un type que nous utiliserons dans la constante MockWords, le typage ici indique que cette constante est un tableau de Word :

```
export class Word {
    word: string;
    hide: string;
}

export const MockWords: Word[] = [
    { word: "cornedrue", hide: "#o######e" },
    { word: "cognards", hide: "c######s" },
    { word: "fourchelang", hide: "########a#g" },
    { word: "gringotts", hide: "#######tts" },
    { word: "hyppogriffes", hide: "####o######s" },
];
```

Vous importerez les données dans le fichier app.ts de la manière suivante, notez **qu'il faut exporter les données avant de les importer** sans quoi vous ne pourrez pas les exploiter dans le fichier app.ts, elles sont scopées dans le module data.ts :

```
import { MockWords } from "../data";
```

### La classe Game

Vous ferez au minimum une classe pour implémenter la logique du jeu. Tout doit être correctement typé.

Instance de la classe Game

```
let game = new Game(MockWords); // on passe les mots à l'objet Game pour initialiser le jeu
```

La méthode run de la classe Game est la méthode « centrale » qui permettra de mettre une grande partie de la logique du jeu.

### Variable enum Status

La variable Status est particulière, en effet nous avons fait le choix d'utiliser un enum en TypeScript. Ce dernier permet de définir une collection de constantes, nous l'utiliserons pour définir l'état du jeu :

```
// Enum
enum Status {
    Winner,
    Loser,
    Progress,
}

let something: Status; // définition d'une variable de type Status
something = Status.Winner; // on assigne une valeur

// Pour la comparaison utiliser l'alias "as" sinon erreur de type lors de la transpilation
// vers le code JS
Status.Progress as Status == something;
Status.Winner as Status == something;
Status.Loser as Status == something;
```

### Le fichier app.ts

C'est le boostrap de votre application, vous testerez le jeu en console à partir de ce fichier.

### Exemple d'interaction dans la console

Lancer le jeu en console

```
➜  pendu_simple node build/app/app.js
Voici un jeu de pendu vous devez deviner le mot caché en 7 coups au plus, vous pouvez
        uniquement proposer un mot, certaines lettres du mot à trouver sont affichées. Bonne chance ! mot : ########a#g
>
```

On peut alors proposer des mots pour deviner le mot caché :

```
➜  pendu_simple node build/app/app.js
Voici un jeu de pendu vous devez deviner le mot caché en 7 coups au plus, vous pouvez
        uniquement proposer un mot, certaines lettres du mot à trouver sont affichées. Bonne chance ! mot : ########a#g
> h
Bien essayé, mais votre mot : h n'est pas le message caché...
                Recommencez, ########a#g
                 nombre de coup(s) restant : 6
> fourche
Bien essayé, mais votre mot : fourche n'est pas le message caché...
                Recommencez, ########a#g
                 nombre de coup(s) restant : 5
>
```

Une fois le mot deviner le jeu s'arrête :

```
> fourchelang
Bravo vous avez trouvé le mot fourchelang en 4
➜  pendu_simple
```

**Partie facultative**

Vous pouvez améliorer le jeu en proposant les options suivantes, dans ce cas créer un autre dossier pendu_optionals :

- Donnez la possibilité de choisir des lettres ou des mots.
- Enregistrez dans une classe Player ou User le pseudo et le score d'un joueur et donner la possibilité de rejouer.
- Mémorisez les lettres déjà jouer et si l'utilisateur propose la même lettre ne pas décompter les coups avant arrêt du jeu.
- Créez une méthode permettant de cacher le mot automatiquement (dans ce cas retirer l'attribut hide dans la classe Word.