Formation Angular 2.0

ANGULARJS

by Google

Planning

Jour 1: Mise en place de l'environnement de développement / découverte de TypeScript

Jour 2: Découverte d'Angular 2.0 (components, directives, services...)

Jour 3: Fonctionnalités avancées (routing, service http ...) & récap

Setup

Nodejs

SublimeText

Package Control

TypeScript plugin

TypeScript

Extension du langage Javascript

N’importe quel code Javascript est valide en TypeScript

Le code TypeScript est transformé en Javascript (transpilation)

npm install -g typescript

Transpilation

tsc --target ES5 --experimentalDecorators example.ts && node example.js

Target ES5: permet par example l’utilisation des getters/setters

experimentalDecorators: active les décorateurs

Pourquoi utiliser TypeScript ?

Avantages du langage objet avec typage:

- le code est rendu plus lisible (classes, interfaces)
- faciliter le découpage du code
- gestion de la visibilité des variables
- moins d'erreurs lors du développement

https://github.com/MarcDeletang/angular-2/blob/master/examples/TSvsJS.ts

Modules et namespaces

La communication entre fichiers se fait via le mot clé import

Le mot clé namespace permet de définir différents domaines afin d'éviter les conflits

https://github.com/MarcDeletang/angular-2/blob/master/examples/import.ts

https://github.com/MarcDeletang/angular-2/blob/master/examples/namespace.ts

Class

Effectue une tâche spécifique

Peut hériter d’une et une seule autre classe ou classe abstraite

Peut implémenter autant d’interfaces que nécessaire

Permet l’encapsulation des variables (private, protected, public)

https://github.com/MarcDeletang/angular-2/blob/master/examples/visibility.ts

Abstract class

Il s’agit bien d’une classe (possibilité de définir des propriétés et des méthodes)

Cependant il n’est pas possible de créer un objet à partir de cette classe (instanciation)

Il est possible de définir des fonctions abstraites qui devront être implémentés par la classe fille

https://github.com/MarcDeletang/angular-2/blob/master/examples/abstract.ts

Interface

Contract que se doit de respecter une classe

Peut posséder des propriétés ou des méthodes

Permet de manipuler plus facilement différents classes au comportement similaire

https://github.com/MarcDeletang/angular-2/blob/master/examples/interface.ts

Getters / Setters

Permettent le contrôle de l'accès aux variables privés

La notation est similaire à une variable mais il s’agit bien d’un appel de méthode

https://github.com/MarcDeletang/angular-2/blob/master/examples/getterSetter.ts

Enum / Static

Enum: Il s’agit d’une structure facilitant la représentation de valeurs numériques

Static: Un membre statique est appelé directement via le nom de classe

https://github.com/MarcDeletang/angular-2/blob/master/examples/static.ts

https://github.com/MarcDeletang/angular-2/blob/master/examples/EAnimals.ts

Readonly

Une propriétée readonly ne peut être modifiée qu'au moment de sa création et/ou dans le constructeur

https://github.com/MarcDeletang/angular-2/blob/master/examples/readonly.ts

Decorators

Permet de modifier le fonctionnement d’une classe, méthode, getter/setter, propriétée, ou paramètre

https://github.com/MarcDeletang/angular-2/blob/master/examples/decorator.ts

Mise en place de l’environnement

Angular CLI:

Permet de créer et déployer rapidement une application angular2

https://github.com/angular/angular-cli

npm install -g angular-cli

Création d’un nouveau projet:

ng new jour1

cd jour1

ng serve --port 1337

Exercice Jour 1

Le but est de créer différentes classes simulant le fonctionnement d’un restaurant ainsi que l’achat des stocks à un magasin.

Exercice - Classes récap

Classes:
Store
Product: IProduct
Restaurant: Bankable
Order: IOrder

Interfaces:
IProduct, IOrder

Classe abstraite:
Bankable

Exercice - The Product

Créer la classe Product qui implémente l'interface IProduct:

Elle contient un identifiant unique, un nom et un prix

https://github.com/MarcDeletang/angular-2/blob/master/base/IProduct.ts

Exercice - The Bankable

Implémenter la classe abstraite bankable

Elle va gérer l’aspect “monétaire” du restaurant

https://github.com/MarcDeletang/angular-2/blob/master/base/Bankable.ts

Exercice - The Store

Permet de stocker les différents produits. Il possède un stockage illimité et de l'argent à l'infini !

https://github.com/MarcDeletang/angular-2/blob/master/base/store.txt

Exercice - The Order

Il s’agit de la commande envoyée au restaurant, elle implémente l’interface IOrder

https://github.com/MarcDeletang/angular-2/blob/master/base/IOrder.ts

Exercice - The Restaurant

Cette classe hérite de bankable

Le constructeur prend un store en paramètre

Une commande peut être passée via passOrder(order: IOrder)

Exercice -Aller plus loins avec les decorators

Créer un décorateur pour la class bankable

function authority(target: any)

Création de décorateurs pour les méthodes addMoney et spendMoney

function authorityMethodIn(target: Bankable, key: any, descriptor: any)

function authorityMethodOut(target: Bankable, key: any, descriptor: any)