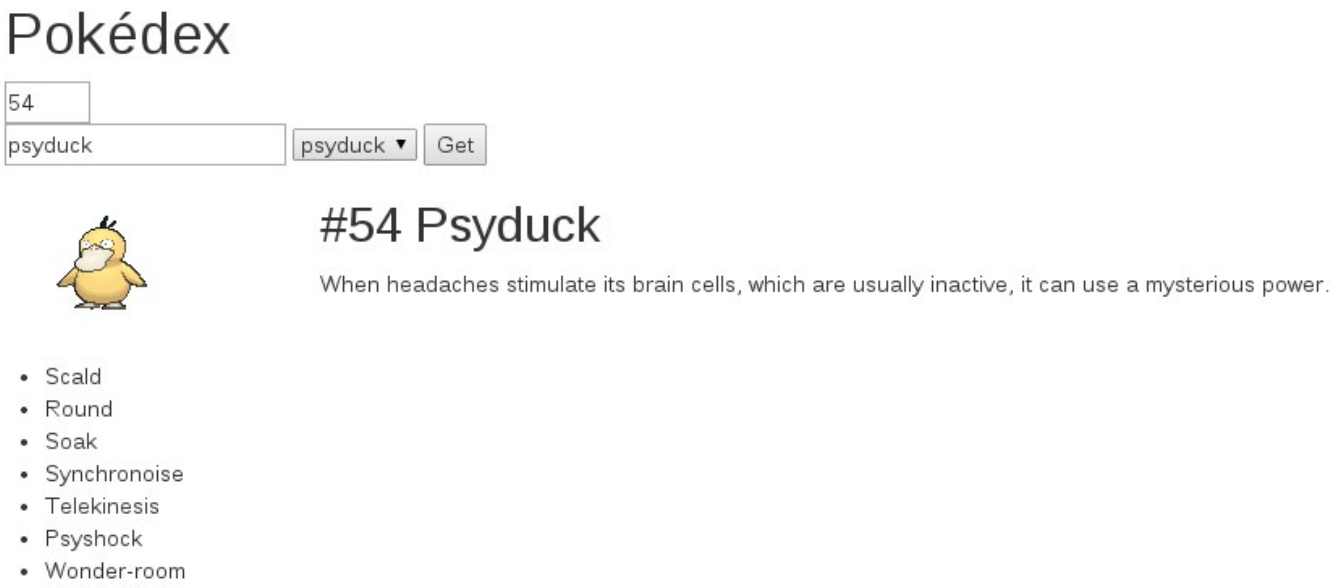
TP3 : AngularJS

Un jour je serai le meilleur codeur
Je développerai sans répit
Je ferai tout pour être vainqueur
Et gagner les défis
Je parcourrai le web entier
Traquant avec espoir
Les frameworks et leurs mystères
Le secret de leurs pouvoirs

[Refrain]
Angular !
Développez les tous
C'est notre TP
Ensemble pour la victoire
Angular !
Rien ne nous arrêtera
Notre binôme triomphera

Votre aventure dans le monde merveilleux des frameworks JavaScript commence avec AngularJS. AngularJS est un framework JavaScript développé par Google. Ce framework se base sur le pattern MVW (Model, View, Whatever). Il offre donc les éléments de bases pour lier le modèle à la vue mais laisse le choix au développeur pour implémenter l'intéraction entre ces deux éléments ainsi que le serveur.

Pour vous aider dans votre quête, le professeur Chen vous a laissé des instructions ainsi que les éléments de bases pour créer un Pokédex à cette adresse : https://github.com/gbecan/teaching-jxs-tp3 (l'utilisation d'une bicyclette est conseillée pour s'y rendre plus vite).



Recherche d'un pokémon via son numéro

La première étape pour développer notre pokédex consiste à proposer au dresseur de rechercher un pokémon via son numéro. Pour récupérer le numéro, nous allons utiliser la directive *ng-model* d'AngularJS (https://docs.angularjs.org/api/ng/directive/ngModel). Cette directive permet de lier la valeur d'une balise HTML à une variable du modèle.

Q1 : Définir une balise <*input*> liée à une variable *id* via la directive *ng-model*.

Pour tester cette directive, nous allons utiliser le méchanisme de data-binding qui permet d'afficher une variable du modèle via la syntaxe suivante : {{maVariable}}.

Q2 : Afficher sur la page la valeur de l'id renseigné dans la balise <input> précédente.

Recherche dans une liste

Malheureusement, seul le professeur Chen connaît précisément le numéro de tous les pokémons. Pour aider les jeunes dresseurs à utiliser le pokédex, nous allons offrir la liste des pokémons ainsi qu'un champ de recherche pour filtrer cette liste.

Pour réaliser cet élément plus complexe, nous allons définir un contrôleur qui va initialiser le modèle avec une liste de pokémons.

Q3 : Créer un contrôleur dans le fichier pokedex.js. Grâce à la directive *ng-controller,* associer le contrôleur à la balise contenant le système de recherche de pokémons.

Q4 : Ajouter une variable contenant une liste de pokémons (4 – 5 éléments suffiront) dans le modèle via le service $scope. Ce service représente le modèle lié au contrôleur précédemment défini. Par exemple, on peut accéder à l'id de la question 1 de la manière suivante : $scope.id

Pour information, chaque directive ng-controller crée une nouvelle instance du contrôleur et du modèle. Par conséquence, les valeurs contenues dans $scope sont différentes d'une instance à l'autre.

Q5 : Afficher la liste des pokémons dans une balise <select> en utilisant la directive *ng-repeat* (https://docs.angularjs.org/api/ng/directive/ngRepeat).

Q6 : Récupérer le choix du dresseur en liant la balise <select> au modèle avec *ng-model*.

Comme la liste des pokémons peut être très longue, nous allons proposer au dresseur de filtrer la liste.

Q7 : Ajouter un champ de texte et lier cet élément à la directive ng-repeat précédente afin d'en filtrer le contenu (https://docs.angularjs.org/api/ng/filter/filter).

Pour valider le choix du dresseur, nous allons ajouter un bouton « Go ! » dont le comportement sera défini dans le contrôleur.

Q8 : Ajouter une balise <button> à la page et lier ce bouton au contrôleur en utilisant la directive *ng-click*.

Q7 : Tester ce système de recherche en affichant le nom du pokémon recherché via le service $log (https://docs.angularjs.org/api/ng/service/$log).

Accès à une API

Le site http://pokeapi.co/ propose une API contenant de nombreuses informations sur les pokémons. En particulier, l'API offre la liste des pokémons (api/v1/pokedex/1) ainsi que des informations détaillées pour chacun d'entre eux (api/v1/pokemon/54 ou api/v1/pokemon/psyduck). Nous allons utiliser cette API comme source d'information pour notre pokédex.

Q8 : Récupérer la liste des pokémons en utilisant le service $http.

Pour récupérer les informations sur les pokémons, nous allons utiliser le service $resource au lieu du service $http (https://docs.angularjs.org/api/ngResource/service/$resource). Ce service implémente des requètes de bases pour interroger API REST (get, save, query, remove, delete). Le service $resource prend en paramètre une simple URL. Cette URL peut contenir de paramètres préfixés par le caractère « : ». Par exemple,

```
$resource("http://pokeapi.co/api/v1/type/:id/")
```

permet d'interroger facilement l'API à propos des types de pokémons en fonction de son identifiant.

L'accès à l'API est une partie importante de notre application et est suceptible d'être utlisée dans plusieurs contrôleurs. Nous allons donc créer un service pour encapsuler notre appel à $resource. Ce service pourra être utilisé de la même manière que $http ou $scope. Pour créer un service, nous utiliserons la factory proposée par angular (https://docs.angularjs.org/guide/services). Tout élément construit via la factory n'est instancié qu'une seule fois.

Q9 : Créer un service qui utilise $resource pour accéder aux informations d'un pokémon.

Q10 : Créer un nouveau contrôleur dédié à l'affichage des informations sur un pokémon. Utiliser le service précédemment créé pour récupérer les informations d'un pokémon. Combiner les différentes directives vues jusqu'ici pour afficher l'id, le nom et les attaques (moves) d'un pokémon.

Communication entre contrôleurs

À présent, nous avons deux parties à notre application. La première permet de rechercher un pokémon grâce à son numéro ou son nom. La deuxième récupère et affiche les informations d'un pokémon. Il ne reste plus qu'à relier ces deux parties pour finaliser notre pokédex. Pour faire communiquer nos deux contrôleurs, nous allons créer un nouveau service qui va contenir les informations à partager. Comme pour le service dédié à l'API, ce service ne sera instancié qu'une seule fois et permettra donc l'échange d'informations.

Q11 : Créer un service contenant (au minimum) le numéro et le nom du pokémon recherché. Modifier votre code pour utiliser ce service.

Les deux contrôleurs sont maintenant liés et utilisent les mêmes informations. Cependant, les informations du pokémon ne sont pas mises à jour si le dresseur change le numéro ou le nom du pokémon recherché. Pour détecter les changements de ces deux attributs, nous allons utiliser le service $scope.$watch.

Q12 : Utiliser le service $watch pour mettre à jour l'affichage du pokédex lors du changement du pokémon recherché.

Création d'une directive

AngularJS fait l'utilisation intensive de directives pour ajouter des fonctionnalités au code HTML d'une page. Il est même possible de créer sa propre directive (https://docs.angularjs.org/guide/directive). Pour en comprendre, le principe de base, nous allons créer une nouvelle directive « pokedex » et y insérer notre code. Cela facilitera la réutilisation de notre pokédex dans d'autres applications angular.

Q13 : Déplacer le code HTML du pokédex dans un nouveau fichier. Créer une nouvelle directive « pokedex » qui référence ce fichier. Utiliser la nouvelle directive dans le fichier index.html pour restaurer les fonctionnalités de notre application pokédex.

Q14 : Si vous avez le temps, compléter l'affichage du pokédex avec l'image du pokémon (sprite) ainsi que sa description.