Développement Web
TP n° 4 : Fiche de personnage

Dans ce TP vous allez développer une fiche de personnage éditable pour un jeu de type RPG.

Contexte

Dans les RPG, les personnages sont caractérisés par un ensemble de statistiques chiffrées. Celles-ci permettent de calculer les chances de réussite des actions du personnage en fonction des règles du jeu. Votre but est de créer une fiche qui permet de visualiser et de modifier ces statistiques.

Pour la visualisation de la fiche, une maquette vous est fournie (fichier `index.html`). Il vous faudra remplacer les différents *placeholders* par des données pertinentes. Les données d'univers de jeu et le modèle pour les personnages apparaissent respectivement dans les fichiers `univers.js` et `personnage.js`. Vous n'avez pas le droit de modifier ces fichiers. D'autres fichiers, principalement vides, sont déjà prévus pour le développement que vous devez réaliser.

Prise en main

Prenez le temps de naviguer dans le projet pour en appréhender la structure. Vous pouvez ignorer le fichier de style `main.css`. Il vous faut par contre comprendre la structure du fichier `index.html` et le modèle utilisé.

Un personnage est composé :
— de son univers de jeu,
— d'un nom,
— d'une classe,
— d'attributs dont la valeur est comprise entre 0 et 4 (inclus),
— de compétences dont la valeur est positive ou nulle.

Les personnages ont 9 points à répartir dans leurs différents attributs.

Chaque classe est caractérisée par :
— un nom,
— un attribut principal,
— des compétences fixes imposées,
— des compétences libres à choisir.

Chaque compétence est liée à un attribut qui représente sa valeur de base. Si une compétence est sélectionnée par un personnage, on ajoute un bonus de 3 à ce score pour obtenir la valeur finale. Les compétences fixes sont forcément sélectionnées par un personnage et ne peuvent être désélectionnées. Chaque personnage sélectionne librement un nombre de compétences supplémentaires en fonction de sa classe.

Un univers de jeu détermine quelles sont les classes et compétences disponibles. Deux univers sont prédéfinis (médiéval fantastique et science-fiction) et sont exportés dans la variable `UNIVERS` de `univers.js`. Ce fichier contient d'autres constantes et des fonctions utilitaires.

Travail à réaliser

Initialisation

Complétez la fonction `initialiser` dans `initialisation.js`. Cette fonction remplace les éléments du document par de nouveaux éléments qui correspondent aux attributs définis dans la constante `ATTRIBUTS`. Elle fait de même pour les compétences et classes en fonction de l'univers sélectionné (médiéval fantastique par défaut).

Affichage

Complétez la fonction `afficher` dans `affichage.js`. Cette fonction prend en paramètre une instance de `Personnage` et met à jour les valeurs des différents champs du document.

Écouteurs

Pour l'instant, il est possible de modifier les champs de la page Web. Cependant, cela n'a aucune incidence sur le modèle. Il faut ajouter des écouteurs pour synchroniser les données et l'affichage.

Complétez la fonction `ajouterEcouteurs` dans `ecouteurs.js`. Cette fonction ajoute les écouteurs aux différents champs du document. Aucun écouteur ne devra modifier directement le modèle. Il faut passer par l'objet `controleur` exporté par `controleur.js`. Complétez en parallèle la classe `Controleur` avec des méthodes adaptées aux traitements que vous souhaitez appliquer (en respectant une architecture MVC). En particulier, chaque modification du modèle par le contrôleur doit entraîner une mise à jour de l'affichage.

Le rôle des écouteurs est de :
— Synchroniser l'univers, le nom, la classe, les attributs et les compétences sélectionnées du personnage entre l'affichage et le modèle.
— Après un changement de classe, seules les compétences fixes doivent être sélectionnées.
— Après un changement d'attributs ou la sélection/désélection d'une compétence, les valeurs des compétences sont mises à jour.

Tag	`input[type="text"]` `input[type="number"]`	`select`	`input[type="checkbox"]`
Évènement	`"input"`	`"change"`	`"click"`

FIGURE 1 – Évènements produits par les champs.

On vous impose d'implémenter la délégation d'évènements pour au moins une des catégories (attributs ou compétences).

Persistance

Si on ferme le navigateur, toutes les données du modèle sont perdues. Modifiez le code pour que le modèle soit synchronisé avec une représentation stockée dans `localStorage`. À l'ouverture de la page, si un personnage est stocké dans `localStorage`, il est automatiquement chargé comme donnée initiale.

Dernières touches

Votre application est globalement fonctionnelle. On peut lui apporter des améliorations :
— Appliquez un style spécial pour l'attribut principal (fouillez un peu dans `main.css`).
— Assurez-vous qu'aucun attribut ne soit inférieur à 0 et ne dépasse 4.
— Empêchez l'utilisateur d'avoir plus de 9 points d'attributs.
— Empêchez l'utilisateur de désélectionner une compétence fixe.
— Empêchez l'utilisateur de sélectionner plus de compétences que sa classe ne l'autorise (s'il n'a pas encore de classe, interdisez complétement la sélection de compétence).
— Affichez des indicateurs sur le nombre de points d'attributs utilisés par rapport au maximum. Faites de même pour les compétences sélectionnées.