Examen WebMobUI - IM 46

Le travail est à rendre par email à nicolas.chabloz@heig-vd.ch **avant 11h05** avec en pièce jointe un *zip* de tout votre projet, sauf le répertoire **node_modules**. (Vous êtes responsable de son contenu, et devez-vous assurer de la bonne réception de celui-ci en fin d'examen.)

- Tout document papier (et électronique en local ou sur le site du cours) autorisé.
- Utilisation de l'ordinateur restreinte aux applications suivantes: éditeur de code, browser restreint aux sites Web du cours et des liens qui s'y trouvent (mais aucun autre site, ni accès aux forums des sites).
- Pas d'autre communication (sauf les appels au WebService) autorisée pendant le travail.

Le non-respect de ces conditions donnera la note de 1 et l'impossibilité de se présenter à l'examen de remédiation !

Mise en place *(10 minutes)*

Vous devez développer une mini application de gestion de prêt sous la forme d'une PWA. Commencez par créez un fichier nommé *index_exa1.js* dans votre répertoire *src*, ainsi qu'un autre nommé *index_exa1.html* dans *dist*. Puis copiez le code HTML disponible ici https://chabloz.eu/e/index_exa1.html comme code HTML, ainsi que la police icomoon disponible ici https://chabloz.eu/e/fonts.zip (à dézipper dans votre dossier *dist*). N'oubliez pas aussi de changer la propriété *entry* de votre configuration webpack dans *webpack.config.js* pour y indiquer *index_exa1.js* à la place de *index.js* . Finalement, lancez les deux outils nécessaires à votre développement: *live-server* (**optionnel**) et *webpack* en mode *watch* (**indispensable**). Vérifiez que tout se passe bien avec un *Hello World*.

Pour des raisons pratiques de correction, **tout votre code javascript devra se trouver dans le fichier index_exa1.js**. Vous ne pouvez donc y importer que des modules déjà existants (ceux utilisés durant le cours) et ne devez pas créez de nouveaux modules. De plus, **votre CSS devra se trouver dans le fichier index_exa1.html lui même**, à la place du commentaire CSS déjà présent.

CSS pour desktop *(15 minutes)*

Mettez en place un code CSS n'agissant que sur les fenêtres de browser plus grande que *30rem* grâce à la *media-queries* suivante: **@media screen and (max-width: 30rem)**

Le code CSS devra : cacher la navigation de la PWA, cacher le bouton **+** possédant la classe **.loan-btn-add-person** ainsi qu'afficher les deux sections d'identifiants **#persons** et **#loans** dans une grille *(CSS Grid)*. Ceux ci seront positionnés dans une simple grille à deux colonnes (et d'une seule ligne). Cette grille sera appliquée à l'élément dom *#wrapper>main*. *#persons* se trouvera à gauche, et *#loans* à droite.

Gestion du menu sur smartphone (20 minutes)

Dans votre code JS, gérez les clics sur l'icone de menu afin de faire apparaître la navigation. Puis réalisez le routage nécessaire pour que la navigation soit opérationnelle tout en gardant actif les fonctionnalités de l'historique du browser (de manière identique à ce qui à été fait durant le TP). La section à afficher par défaut est la section *#loans*. **Important:** La totalité de votre PWA doit rester fonctionnelle même si l'utilisateur redimensionne la fenêtre durant son utilisation (remarque: cette partie est difficile).

Ajout des personnes *(20 minutes)*

Gérez l'ajout de personnes dans le *LocalStorage* (en utilisant la classe *JsonStorage*) en écoutant l'événement *submit* du formulaire **#person-form-add**. Une personne est simplement identifiée par son nom (une simple chaîne de caractère).

La liste des personnes doit être affichée dans l'élément **#persons-list** ainsi que comme *option* dans la liste déroulante **#loan-input-person**. Les personnes doivent être triées par ordre alphabétique (grâce à la méthode *localeCompare*). Vous trouverez une *template* **.tmpl-person** dans le HTML que vous **devez** utiliser pour chaque personne pour leur ajout à **#persons-list**. Pour ce qui est de la liste déroulante, les options doivent avoir comme valeur, l'identifiant (*key*) des personnes et comme texte, le nom des personnes.

Modification d'une personne *(15 minutes)*

Vous devez gérer le clic sur le bouton de modification **.person-btn-mod** d'une personne. Il devra déclencher un simple *prompt* afin que l'utilisateur puisse saisir la modification du nom de la personne. Le texte du *prompt* sera repris de l'attribut **data-prompt** du bouton. Cette modification sera sauvegardée dans le LocalStorage en y *écrasant* l'élément adéquat et donc provoquer un rafraîchissement du DOM des deux listes de personnes.

Ajout d'un prêt *(20 minutes)*

Gérez l'ajout d'une chose en prêt dans le *LocalStorage* en écoutant l'événement *submit* du formulaire **#loan-form-add**. Un prêt sera simplement identifié par deux choses: un libellé de la chose en prêt (un texte libre), et **l'identifiant** (*key*) de la personne qui l'a emprunté.

La liste des prêts doit être affichée dans l'élément **#loans-list**. Les prêts doivent être triées par ordre alphabétique du libellé. Vous trouverez une *template* **.tmpl-loan** dans le HTML que vous **devez** utiliser pour chaque prêt pour leur ajout à **#loans-list**. Pour vos tests, vérifiez que lorsque vous modifiez le nom d'une personne, son nom se trouvent aussi modifié dans les prêts à cette personne.

Suppression d'un prêt (5 minutes)

Vous devez gérer le clic sur le bouton de suppression d'un prêt **.loan-btn-del**. Il vous faudra simplement supprimer l'élément dans le *LocalStorage*. Cela devrait provoquer un rafraîchissement du DOM de la liste des prêts.

Suppression d'une personne (15 minutes)

De la même manière, vous devez gérer la suppression d'une personne (bouton **.person-btn-del**). Si l'utilisateur tente la suppression d'une personne ayant un prêt en cours, celle-ci n'est pas supprimée et une alerte (méthode JS *alert*) avec un message d'erreur est affiché. Le texte de l'alerte sera repris de l'attribut **data-alert** du bouton.