Infini sujet Dragons list

Notez que webpack est en mode strict. Celui-ci vous apportera une meilleure granularité dans la remontée des erreurs.

Récupérez les données de l'exercice ci-dessous :

```
const dragons = {
    names: [
        { id: 1, name: "Apalala" },
        { id: 2, name: "Balaur" },
        { id: 3, name: "Bolla" }
    ],
    count: 3,
}

export default dragons;
```

1. Affichez les dragons dans une liste ul/li. Importez les données dans le fichier app.js et créez une fonction addDragons que vous appelerez dans ce fichier. La fonction addEventListener permettra de charger les données une fois le DOM construit :

```
document.addEventListener("DOMContentLoaded", (event) => {
    // appel de votre fonction addDragons avec ses paramètres
});
```

2. Affichez en premier leur nombre et modifiez la fonction addDragons.

- Babel

Babel va nous permettre de traduire du code ES6 ou ES2020 en ES5 compatible avec la plupart des navigateurs.

```
npm install --save-dev babel-loader @babel/core @babel/preset-env
```

Dans le fichier webpack.config.js vous devez définir le loader Babel. On utilise le presets @babel/preset-env.

```
const path = require('path');

module.exports = {
    // ...
    ,
   module: {
      rules: [
        {
          test: /\.m?js$/,
          exclude: /(node_modules|bower_components)/,
          use: {
            loader: "babel-loader",
            options: {
              presets: ["@babel/preset-env"],
            },
          },
        },
      ],
    }
}
```

- polyfill pour les nouvelles fonctionnalités des nouvelles versions de JS

Si vous utilisez les nouvelles features JS, vous devez installer des polyfill afin de les transcrire pour la plupart des navigateurs.

```
np     tall --save babel-polyfill
```

?

- onfigurez correctement le projet avec Webpack et Babel

4. Ajoutez les polyfill au projet

Ajoutez également dans votre fichier webpack.config.js, pour la clé **entry** la configuration suivante :

```
const path = require('path');

module.exports = {
    // ...
    entry: ['babel-plyfill', './src/app.js' ], // Point d'entrée

   // La suite de votre code ...
}
```

5. Dans le dossier src ajoutez les nouveaux dragons et importez ces dragons dans le fichier app.js

```
const dragons = {
    names: [
        { id: 1, name: "Apalala", element : 'fire' },
        { id: 2, name: "Balaur", element : 'water' },
        { id: 3, name: "Bolla" }
    ],
    count: 3,
}

export default dragons;
```

6. Affichez les dragons dans une liste ul/li et précisez à chaque fois s'il existe son élément (fire ou water). Utilisez la nouvelle syntaxe ES2020 suivante pour tester l'existence d'une propriété :

```
myObject?.attribut
```

- Que s'est-il passé dans le fichier bundle.js ? Il faut builder votre fichier et l'inspecter avec la console du navigateur.

7. Créez le fichier relationships.js suivant. Sous chaque dragon affichez ses relations avec les autres dragons :

```
const relationships =  [
    { id: 1, relations : [2, 3] },
    { id: 2, relations : [1] },
    { id: 3, relations : [2] }
]
```

8. Un jury a attribué des valeurs sur la force de chaque dragon. Affichez la moyenne de ces notes sous chaque dragon.

```
const forces =  [
    { id: 1, notes : [12, 13, 19, 11] },
    { id: 2, notes : [11, 15, 17, 9] },
    { id: 3, notes : [20, 11, 12, 7] }
]
```

9. (**) créez un bouton order pour ordonner l'affichage des dragons par ordre croissant ou décroissant de force.

Wireframe

```
---------------------------------------------

                  Dragons
[ order C/D ]
---------------------------------------------
   Apalala, element : fire
       force : ...
   Balaur , element : water
       force : ...
   Bolla
       force : ...
```

M vendredi 22 septembre 2023, 13:07