307 - Réaliser des pages Web interactives

EMF – Fribourg / Freiburg Ecole des Métiers / Berufsfachschule Technique / Technik		Module du 15.05.23 Au 03.10.2023

Rapport personnel

Date de création : 15.05.2023

Version 2 du 06.09.2023

Nathan França

Table des matières

1 Introduction 7
2 Javascript 8
3 Exercices 8
3.1 Exercice 1 8
3.2 Exercice 2 9
3.2.1 Rendu final et Solution 10
3.2.2 Fonctions alert(), confirm() et prompt() 10
3.2.3 Bouton button et input 11
3.2.4 Récupérer un élément HTML depuis Js avec un id et récupérer texte dans un champ de formulaire 12
3.2.5 Convertir un texte en minuscule 12
3.3 Exercice 3 12
3.3.1 Objectif 12
3.3.2 Explications / Descriptions 12
3.3.3 Solutions 13
3.3.4 Captures 16
3.4 Exercice 4 17
3.4.1 Explications 17
3.4.2 Solutions 17
3.4.3 Captures 18
3.5 Exercice 5 18
3.5.1 Objectif 18
3.5.2 Explications 18
3.5.3 Solutions 18
3.5.4 Captures 19
3.6 Exercice 6 20
3.6.1 Objectif 20
3.6.2 Explications 20
3.6.3 Solutions 21
3.6.4 Captures 22
3.7 Exercice 7 23
3.7.1 Objectif 23
3.7.2 Explications 23

3.7.3 Solutions....23
3.7.4 Captures....25
3.8 Exercice 8....26
3.8.1 Objectif....26
3.8.2 Explications....26
3.8.3 Solutions....26
3.8.4 Captures....27
3.9 Exercice 9....27
3.9.1 Objectif....27
3.9.2 Explications....27
3.9.3 Solutions....27
3.9.4 Captures....28
3.10 Exercice 10....28
3.10.1 Objectif....28
3.10.2 Explications....28
3.10.3 Solutions....29
3.10.4 Captures....31
3.11 Exercice 11....31
3.12 Objectif....31
3.12.1 Explications....31
3.12.2 Solutions....32
3.12.3 Captures....34
3.13 Exercice 12....35
3.13.1 Objectif....35
3.13.2 Explications....36
3.13.3 Solutions....36
3.14 Exercice 13....37
3.14.1 Objectif....37
3.14.2 Explications....37
3.14.3 Solutions....38
3.14.4 Captures....39
3.15 Exercice 14....39
3.15.1 Objectif....39
3.15.2 Explications....40
3.16 Exercice 15....42
3.16.1 Objectifs....42
3.16.2 Explications....42

3.16.3 Solutions....42
3.16.4 Captures....42
3.17 Exercice 16....43
3.17.1 Objectif....43
3.17.2 Explications....43
3.17.3 Solutions....43
3.17.4 Captures....43
3.18 Exercice 17....44
3.18.1 Objectif....44
3.18.2 Explications....44
3.18.3 Requête GET....44
3.18.4 Requête POST....44
3.18.5 Requête PUT....44
3.18.6 Requête DELETE....44
3.18.7 Présentations....44
3.19 Exercice 18....44
3.19.1 Objectif....44
3.19.2 Explications....44
3.19.3 Solutions....45
3.19.4 Captures....46
3.20 Exercice 20....47
3.20.1 Objectif....47
3.20.2 Explications....47
3.20.3 Solutions....48
3.20.4 Captures....48
4 Projet....49
4.1 Introduction du projet....49
4.2 Analyse....49
4.2.1 Use-case....49
4.2.2 Maquette....50
4.3 Conception....53
4.3.1 Diagramme de navigation....53
4.4 Implémentation....54
4.4.1 Extraits de code....54
4.4.2 Problèmes rencontrés....56
4.5 Tests....56
4.5.1 Test des Webservices....56

4.5.2 Test du fonctionnement....56
4.6 Hébergement et fonctionnement....57
5 Conclusion....57
5.1 Analyse .... Erreur ! Signet non défini.
5.1.1.1 Diagramme .... Erreur ! Signet non défini.
5.1.1.2 Maquette .... Erreur ! Signet non défini.
5.2 Conception .... Erreur ! Signet non défini.
5.2.1 Diagramme de navigation .... Erreur ! Signet non défini.
5.3 Implémentation .... Erreur ! Signet non défini.
5.3.1 HTML / CSS .... Erreur ! Signet non défini.
5.3.2 Javascript général .... Erreur ! Signet non défini.
5.3.3 Javascript spécifiques .... Erreur ! Signet non défini.
5.3.4 Descente de code complète pour une action qui fait appel à un service Web Erreur ! Signet non défini.
6 Conclusion .... Erreur ! Signet non défini.

1 Introduction

Dans ce module nous allons revisiter les bases du module 101 ou nous avions vu le langage html et css. Nous avons également vu de manière très rapide et brève le Javascript, mais nous n'avons pas vraiment vu cela de manière complète. Dans ce module nous allons donc apprendre le langage Javascript ainsi que le mettre en lien avec du html et du css.

Voici les objectifs du module :

- Développer le caractère fonctionnel des pages Web interactives conformément aux données du problème.
- Développer une maquette pour la saisie et la présentation des données compte tenu des aspects ergonomiques.
- Choisir les éléments de formulaire appropriés pour la réalisation des données du problème, et garantir la validation des données entrées.
- Programmer l'application de manière modulaire et conformément aux directives de codification.
- Définir et en mettre œuvre des cas de tests appropriés pour des pages Web interactives et documenter dans le procès-verbal de tests.

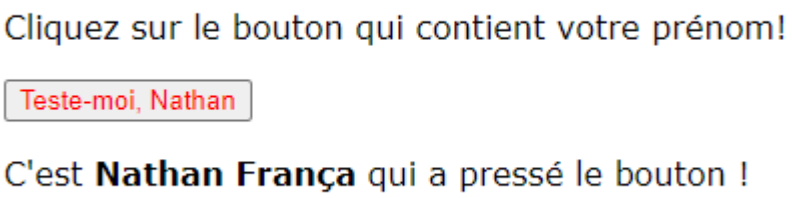
Voici encore un lien pour rappel, qui est très utile afin de comprendre l'utilisation du flex pour la partie css : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

2 Javascript

3 Exercices

3.1 Exercice 1

Dans l'exercice 1 nous avons tout d'abord dû effectuer quelques modifications simples dans le document fourni de base :



Nous avons changé la couleur du texte à l'intérieur du bouton en rouge, en faisant color : red ; dans l'id testez.

Cela était très simple et n'était clairement pas l'exercice en entier, ensuite nous avons examiné le document et essayé de comprendre le fichier Javascript.

Dans les fichiers qui ont été donné, l'écouteur se situe dans le fichier Javascript, la suite de l'exercice était de faire en sorte de mettre l'écouteur en dur dans le fichier html.

Voici les réponses aux questions qui sont fournies dans la donnée de l'exercice :

Comment ajouter un écouteur "click" via Javascript.

Ceci était déjà présent dans le fichier donné et on se réfère à un id d'une balise html :

```
function initCtrl() {
  // Ecouteur du bouton "Testez-moi..."
  document.getElementById("testez").addEventListener("click", testez);
}
```

Le premier « testez », fait référence à l'id qui était présente sur un bouton. On voit ensuite qu'il ajoute un eventlistener, qui va effectuer la méthode testez lorsque nous cliquons sur le bouton.

Comment ajouter directement un écouteur "click" sur un bouton.

Si nous voulons éviter de passer par l'écouteur depuis Javascript, à la place d'utiliser un id, nous pouvons directement procéder de cette manière :

```
<button onclick="testez()">Teste-moi, Nathan</button>
```

Expliquez comment exécuter du code Javascript lorsque la page HTML a fini d'être chargée.

Pour exécuter du code javascript lorsqu'une page HTML a fini d'être chargée, il faut utiliser le onload :

```
<body onload="initCtrl()">
```

Expliquez ce que signifie DOM.

Le Document Object Model est le modèle d'une page lorsque la page web est chargée. C'est une interface de programmation normalisée par W3C, qui permet aux scripts d'examiner ainsi que de modifier le contenu de notre navigateur web.

Où les scripts Javascript peuvent être chargés et pourquoi.

Voici la ligne de texte qui va permettre de référencer au fichier Javascript afin de pouvoir implémenter du Javascript :

```
<script type="text/javascript" src="js/indexCtrl.js" async></script>
```

Ensuite le onload que nous avons vu précédemment est implémenter dans la balise body, afin de s'assurer que tous les éléments html ont été implémenté d'abord avant d'exécuter le code Javascript.

3.2 Exercice 2

Dans le deuxième exercice nous nous sommes un peu plus attaqués à la partie javascript. Nous avons fait un exercice similaire à l'exercice 1, mais avec un login et des popups.

Voici les réponses aux questions :

À quoi sert l'attribut « placeholder » ?

Le placeholder sert à afficher un texte lorsque l'entrée est vide. Et quand nous entrons un nom ou un mot de passe, le texte s'enlève.

À quoi sert l'attribut « autofocus » ?

L'attribut autofocus est un booléen qui indique qu'un élément est supposé recevoir le focus lorsque la page se charge ou quand l'élément <dialog> dont il pourrait faire partie est affiché.

Y a-t-il déjà du JavaScript dans ce code ?

L'attribut onclick va permettre d'appeler une fonction javascript lorsque l'on appuiera sur le bouton. Donc oui il y a du Javascript dans ce code.

Quelle est la différence entre le bouton de l'exercice 1 et celui-ci ?

Le bouton original de l'exercice n'utilisait pas le onclick, mais passait par un id auquel on faisait ensuite référence dans le fichier Javascript. Alors que la il est en dur dans le fichier html.

Ensuite nous avons dû apprendre comment effectuer certaines des actions de base avec le langage Javascript afin de répondre aux questions suivantes :

- Comment définit-on une *« fonction »* (méthode) comme *« validerUtilisateur() »* ?

En faisaint :

```
function validerUtilisateur() {
}
```

- Comment définit-on une *« variable »* (par exemple pour *« username »* et *« password »*) ?

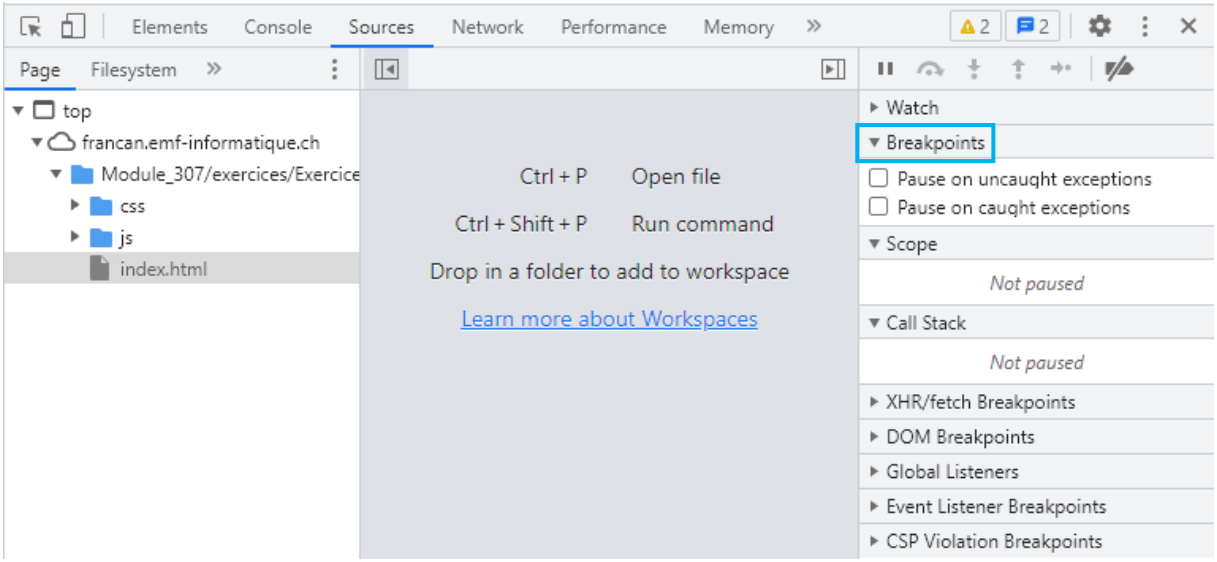
Il y a plusieurs manières comme var, let, const. Mais de manière générale pour une constante nous utilisons const et pour une variable qui pourrait changer de valeur nous utilisons let. Sauf si le code doit être compatible avec des versions d'avant 2015, dans ce cas nous utilisons toujours var.

- Les variables sont-elles *« typées »* (String par exemple) ?

Il n'est pas nécessaire de déclarer le type des variables.

- Où se trouve la console de débogage de Chrome ?

Nous pouvons trouver le debugger dans l'onglet srouces lorsque nous inspectons une page web :



3.2.1 Rendu final et Solution

Voici la méthode Javascript que j'ai implémenté à l'aide des instructions de la donnée :

```
function validerUtilisateur() {
let username = document.getElementById("username").value;
let password = document.getElementById("password").value;
  if(username == "admin" && password == "emf123"){
    window.alert("Validation OK.");
  }else{
    window.alert("Utilisateur ou mot de passe incorrect !!!");
  }
}
```

Ceci va créer deux variables avec le let ou nous stockons la valeur des balises username et password. Ensuite nous faisons un if else, ou nous vérifions si l'username et password correspondent à admin et emf123, si oui nous envoyons le premier message et sinon le deuxième.

3.2.2 Fonctions alert(), confirm() et prompt()

La fonction alert(), comme nous l'avons vu sert à envoyer un message à l'aide d'une fenêtre qui va s'afficher à l'utilisateur. La fonction window.alert() fait exactement la même chose.

```
    alert("Validation OK.");
```

La fonction confirm() va afficher une boite de dialogue avec un message, un bouton OK et un bouton Cancel. Elle retourne vrai si le bouton OK est appuyé et false si le Cancel est appuyé.

```
    confirm("Voulez-vous continuer?");
```

La fonction prompt() va afficher une boite de dialogue qui va inviter l'utilisateur à saisir des données.

```
    prompt("Veuillez écrire votre Prénom");
```

3.2.3 Bouton button et input

L'élément button peut avoir du contenu, contrairement au bouton input qui est un élément null. Donc on ne peut pas ajouter de texte au bouton input, mais nous pouvons nommer un bouton button.

3.2.4 Récupérer un élément HTML depuis Js avec un id et récupérer texte dans un champ de formulaire

Comme nous avons pu le voir avec les deux précédents exercices, nous pouvons récupérer un élément ainsi que sa valeur en nous servant de son id. Cela est très pratique pour placer un EventListener ou tout simplement récupérer sa valeur.

Nous pouvons procéder comme ceci :

```
          <input type="text" size="30" id="username"
            placeholder="un nom svp" autofocus />
```

Si-dessus nous avons ajouté un id « username » et maintenant nous allons procéder comme cela :

```
let username = document.getElementById("username").value;
```

Ici avec le document.getElementById, nous récupérons l'élément et le .value va obtenir sa valeur.

3.2.5 Convertir un texte en minuscule

Si nous voulons convertir un texte en minuscule, nous pouvons utiliser une des nombreuses fonctions String de Javascript :

```
  text.toLowerCase();
```

3.3 Exercice 3

3.3.1 Objectif

Cet exercice va nous permettre d'améliorer un peu la vue en décolorant les bordures du fieldset ainsi qu'en centrant et stylisant le bouton.

Ensuite nous allons faire la validation des informations de login par rapport à un script réalisé avec le langage PHP (qui va remplacer Javascript) sur un serveur Apache.

3.3.2 Explications / Descriptions

- Une explication sur les différents types de boutons programmables dans un formulaire HTML.

Dans cet exercice nous devions faire un programme qui permettait de vérifier un login grâce à un mot de passe. Il se servait du PHP afin de contrôler cela. La spécificité d'un élément input et que afin de le centrer, nous devons utiliesr « text-align » pui ensuite center.

Il existe plusieurs types de boutons avec HTML, voici un tableau qui resume cela :

Button	Un bouton simple.
Submit	Bouton qui envoie un formulaire vers un serveur lorsqu'il est cliqué.
Reset	Bouton qui réinitialise les valeurs d'un formulaire.
Image	Bouton via une image.
Checkbox	Bouton via checkbox.
Radio	Bouton via groupe de bouton.
File	Bouton qui permet de sélectionner un fichier.

- Par quoi commence et doit se terminer un code PHP ?

Par ces deux lignes :

```
<?PHP
?>
```

- Comment récupère-t-on une information dans le flux des informations GET ou POST ?

Nous pouvons les récuperer via l'attribut Name, afin de faire cela nous mettons un attribut Name a un input, puis nous allons utiliesr la fonction $_POST ou $_GET, ensuite nous allons pouvoir récuperer les informations de cet élément. (Exemple dans le point suivant.)

- Comment effectue-t-on une concaténation de chaines de caractères en PHP ?

Afin de concaténer deux chaines de caractères, il nous faut utiliser le point.

- Avec quelle fonction intrinsèque (dans PHP) convertit-on une chaine en minuscules ?

Avec cette fonction strtolower (qui veut plus ou moins dire String to Lower(case)):

```
$username = strtolower($_POST['username']);
```

- Quelle est la commande qui permet de renvoyer quelque chose vers le client ?

La commande Echo permet de renvoyer quelque chose vers l'utilisateur.

- Quel(s) autre(s) langage(s) trouve-t-on encore dans ce code PHP ?

Du Javascript dans les balises <script> afin d'envoyer la fenêtre d'alerte.

3.3.3 Solutions

Voici ce que j'ai implémenté pour centrer le bouton :

```
.user-form .button {
  text-align: center;
}

.user-form input[type=submit] {
  margin-top: 5px;
  color:white;
  background-color: grey;
}
```

Et voici le fichier html ainsi que le fichier php pour la méthode POST :

```
  <body>
    <form class="user-form"
      action="https://francan.emf-
informatique.ch/Module_307/exercices/Exercice_3/js/login.php"
      method="POST">
      <fieldset>
        <legend>Identification:</legend>
        <div class="field">
          <label for="username">Nom d'utilisateur:</label>
```

```
          <input type="text" size="30" id="username"
            placeholder="un nom svp" autofocus name="username"/>
        </div>
        <div class="field">
          <label for="username">Mot de passe:</label>
          <input type="password" size="30" id="password"
            placeholder="un mot de passe svp" name="password"/>
        </div>
        <div class="button">
          <input type="submit" value="Valider">
        </div>
      </fieldset>
    </form>
  </body>
```

php :

```
<?PHP
// test si on a reçu une donnée de formulaire nommée "username"
if (isset($_POST['username'])) {

  // récupération des données transmises dans des variables locales
    $username = strtolower($_POST['username']);
    $password = $_POST['password'];

  // affichage des infos reçues
  echo "username: ".$username."</br>";
  echo "password: ".$password."</br>";

  // test username et mot de passe
  if (($username == "admin") && ($password == "emf123")) {
    echo "<script>alert('Validation OK');</script>";
  } else {
    echo "<script>alert('Utilisateur ou mot de passe incorrect
!!!');</script>";
  }
}
?>
```

Et voici en mode GET :

```
    <form class="user-form"
    action="https://francan.emf-
informatique.ch/Module_307/exercices/Exercice_3/js/login-get.php"
    method="GET">
    <fieldset>
      <legend>Identification:</legend>
      <div class="field">
        <label for="username">Nom d'utilisateur:</label>
```

```
        <input type="text" size="30" id="username"
          placeholder="un nom svp" autofocus name="username"/>
      </div>
      <div class="field">
        <label for="username">Mot de passe:</label>
        <input type="password" size="30" id="password"
          placeholder="un mot de passe svp" name="password"/>
      </div>
      <div class="button">
        <input type="submit" value="Valider">
      </div>
    </fieldset>
  </form>
```

Php :

```
<?PHP
// test si on a reçu une donnée de formulaire nommée "username"
if (isset($_GET['username'])) {

  // récupération des données transmises dans des variables locales
    $username = strtolower($_GET['username']);
    $password = $_GET['password'];

  // affichage des infos reçues
  echo "username: ".$username."</br>";
  echo "password: ".$password."</br>";

  // test username et mot de passe
  if (($username == "admin") && ($password == "emf123")) {
    echo "<script>alert('Validation OK');</script>";
  } else {
    echo "<script>alert('Utilisateur ou mot de passe incorrect
!!!');</script>";
  }
}
?>
```

3.3.4 Captures

3.4 Exercice 4

Dans cet exercice nous essayons le site Internet JsFiddle avec un exercice ou nous appuyons sur un carré bleu créé à partir de html et css et lorsque nous appuyons, à l'aide de Javascript une alerte et censé nous dire que nous avons cliqué sur le carré bleu.

3.4.1 Explications

Voici les réponses aux questions de l'exercice 4 :

- Qu'est-ce que JSFiddle ? A quoi sert-il ?

JsFiddle est un site qui propose un outil afin de tester du code Javascript à partir d'un code html + css. Le tout est appelé un Fiddle et cela peut être très utile afin de tester un code Javascript.

- Quel est l'avantage d'utiliser cet outil ?

L'avantage c'est que nous ne devons pas lancer notre IDE et ne devons pas mettre les différents fichiers en place avant de pouvoir tester si notre code est fonctionnel. C'est donc très rapide et très efficace.

3.4.2 Solutions

Voici le code html, css et javascript pour l'exercice 4, qui n'était pas tellement difficile à faire :

Html :

```html
<html>

  <body onload="initCtrl()">
    <div id="carre">
    </div>
  </body>

</html>
```

Css :

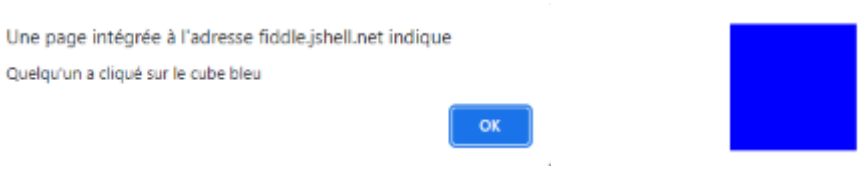
```css
#carre{
  background-color:blue;
  display:flex;
  justify-content:center;
  border: 1px solid black;
  width:100px;
  padding-top:50px;
  padding-bottom:50px;
}
```

Javascript :

```javascript
function initCtrl() {
  document.getElementById("carre").addEventListener("click", carrebleu);
}

function carrebleu() {
```

```
console.log("test");
window.alert("Quelqu'un a cliqué sur le cube bleu");
}
```

3.4.3 Captures



3.5 Exercice 5

3.5.1 Objectif

Dans l'exercice cinq nous allons effectuer toutes sortes commandes en javascript et beaucoup travailler avec les différents types de variables.

3.5.2 Explications

- Comment déclarez-vous une variable en Javascript ?

Nous pouvons comme vu dans le point 3.2 il existe plusieurs manières différentes, comme var, let et const (constante).

- Comment déclarez-vous une constante et quelle convention est utilisée ?

Nous déclarons une constante avec const a = 3 ;

Cependant, une fois qu'une valeur a été assigné à la constante, nous ne pouvons pas modifier sa valeur dans une autre ligne. Sinon cela donnera une erreur.

- Quelle est la différence entre les mots clés ***var*** et ***let*** ?

Le mot var était encore il y a quelques années l'unique façon de créer une variable en Javascript. Mais depuis 2015 nous pouvons également utiliser la variable let et aussi const pour les constantes.

- Quelle particularité y a-t-il à déclarer une variable sans les mots clés ***var*** et ***let*** ?

Et bien ceci reviendrait à utiliser une constante et comme dit avant nous ne pouvons pas réassigner une valeur plus tard et cela peut être utile dans certains cas.

- Combien y a-t-il de type primitif en Javascript et quels sont-il ?

Il y en a sept et ils sont très similaire à ceux de Java :

Tout d'abord le String qui est une chaine de caractère, ensuite Number qui est un Nombre, boolean, Null, undefined, Symbol et Bigint.

- Quelle instruction permet de connaître le type d'une variable ? Donnez un exemple.

L'instruction typeof permet cela et nous l'utilisons de cette manière :

```
console.log(typeof a);
```

3.5.3 Solutions

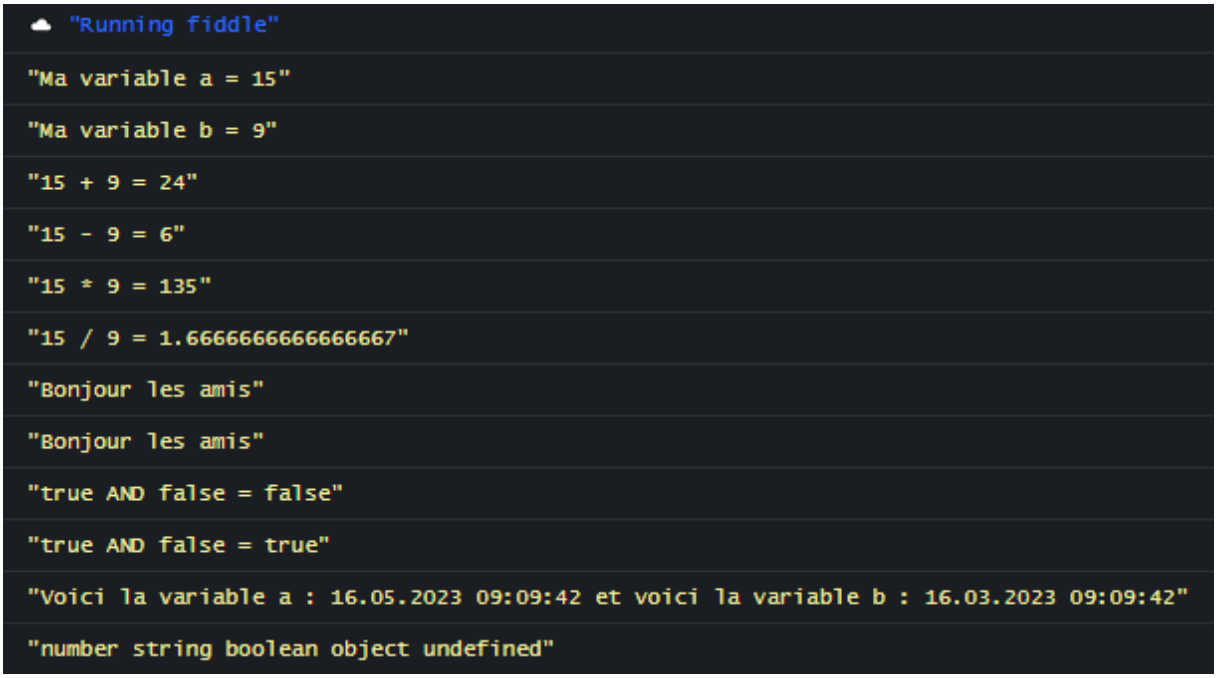
Voici les commandes que j'ai dû essayer :

```
console.clear();
let a = 15;
console.log("Ma variable a = " + a);
let b = 9;
console.log("Ma variable b = " + b);
console.log(a + " + " + b + ` = ${a + b}`);
```

```
console.log(a + " - " + b + ` = ${a - b}`);
console.log(a + " * " + b + ` = ${a * b}`);
console.log(a + " / " + b + ` = ${a / b}`);
a = "Bonjour";
b = "les amis";
console.log(a + " " + b);
console.log(`${a + " " + b}`);
a = true;
b = false;
console.log("true AND false = " + (a&&b));
console.log("true AND false = " + (a||b));
a = new Date();
b = new Date();
b.setDate(a.getDate() - 61);
console.log("Voici la variable a : " + a.toLocaleString() + " et voici la
variable b : " + b.toLocaleString());
a = Math.PI;
b = "bonjour";
let c = true;
let d = new Date();
let e;
console.log(typeof a + " " + typeof b + " " + typeof c + " " + typeof d + " "
+ typeof e);
```

3.5.4 Captures

Et voici une capture de l'output :



3.6 Exercice 6

3.6.1 Objectif

Nous reprenons l'exercice 1 afin de tester une nouvelle manière de faire une condition, c'est-à-dire le Switch que nous avions déjà vu dans le langage Java.

Dans notre application nous allons devoir afficher le jour de la semaine directement en remplacement du contenu d'une balise HTML qui existe déjà.

Nous avons également vu comment les tableaux fonctionnent et utilisé la commande getDay(), afin d'obtenir un nombre qui représente un jour de la semaine. (0 = Dimanche, 1 = Lundi etc).

3.6.2 Explications

- Comment récupérez-vous le numéro du jour actuel ?

Afin de pouvoir récupérer le jour actuel nous utilisons la commande getDay(), pour cela nous avons créer une new Date en faisant : let a = new Date(), et ensuite nous stockons a.getDay() dans une autre variable et cela nous retournera un nombre représentant un jour de la semaine.

- Comment pouvez-vous insérer du texte dans une balise html ?

Afin d'insérer du texte dans une balise html tel qu'un paragraphe par exemple, nous utilisons le innerHtml de cette manière :

```
  document.getElementById("info").innerHTML =
    "Nous sommes ";
```

- Comment pouvez-vous insérer du code html dans une balise html ? Par exemple "On est mardi"

De la même manière sauf que nous procédons comme cela :

```
document.getElementById("info").innerHTML =
"<b>On est mardi</b>";
```

- Expliquez comment créer un tableau, vide ou avec des valeurs.

Afin de créer un tableau nous utilisons les mêmes variables que pour les autres types.

Nous faisons comme ceci :

```
const LesJours = ["Salut", "Aurevoir"];
```

Et si il doit être vide il faut tout simplement mettre des parenthèses [] ; sans rien à l'intérieur.

- Expliquez comment ajouter et supprimer des valeurs, au début ou à fin, d'un tableau.

Nous faisons ainsi :

```
noms.push('Eric');         // ajoute Eric à la fin du tableau
noms.unshift('AA');      // ajoute AA au début du tableau

noms.pop();             // efface le dernier élément du tableau
noms.shift();            // efface le premier élément du tableau
```

- Expliquez comment récupérer une valeur, dans un tableau, à position donnée.

Nous pouvons tout simplement faire ainsi :

```
let nom = noms[2];
```

- Expliquez comment afficher l'ensemble des valeurs d'un tableau.

Nous pouvons utiliser une boucle comme nous le faisions en Java, par exemple une boucle for :

```
for (let i=0; i<noms.length; i++) {
  console.log(i + " : " + noms[i]);   // 0 : Alice …
}
```

3.6.3 Solutions

Voici les codes html et Javascript que j'ai effectué afin d'arriver à mon objectif :

```
<body >
  <div id="container">
    <p>
      Cliquez sur le bouton ci-dessous pour découvrir quel jour nous sommes.
    </p>
    <button onclick="afficherJourSemaine();">Testez-moi</button>
    <button onclick="afficherJourSemaineTableau();">Testez-moi
(Tableau)</button>
    <p id="info"> </p>
  </div>
</body>
```

Javascript :

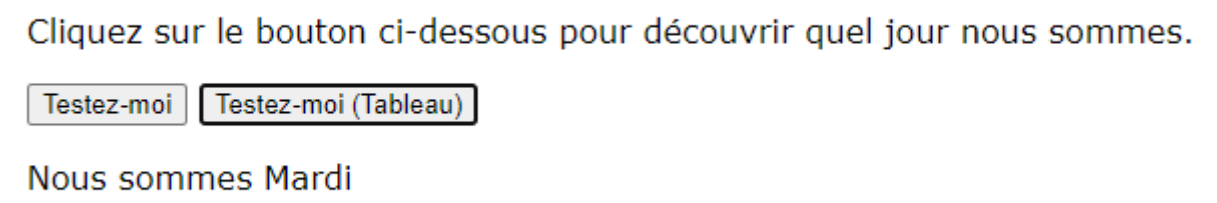
```
const date = new Date();
let Jour = date.getDay();
let print = "";
const LesJours = ["Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi",
"Vendredi", "Samedi"];

function afficherJourSemaine() {
  switch (Jour) {
    case 0:
      print = "Dimanche";
      break;
    case 1:
      print = "Lundi";
      break;
    case 2:
      print = "Mardi";
      break;
    case 3:
      print = "Mercredi";
      break;
```

```
    case 4:
      print = "Jeudi";
      break;
    case 5:
      print = "Vendredi";
      break;
    case 6:
      print = "Samedi";
      break;
  }
  document.getElementById("info").innerHTML =
    "Nous sommes " + print;
}

function afficherJourSemaineTableau(){
  print = LesJours[Jour];
  document.getElementById("info").innerHTML =
    "Nous sommes " + print;
}
```

Ici nous pouvons voir que la première fonction se sert de la commande getDay(), afin d'ensuite pouvoir faire des switch avec une liste de cas de retour. La deuxième fonction se sert également de la commande getDay(), mais compare son résultat au tableau des jours de la semaine.

3.6.4 Captures

Voici une capture du résultat lorsque nous lançons le programme :



3.7 Exercice 7

3.7.1 Objectif

Tout d'abord nous copions l'exercice 6 dans le dossier de l'exercice 7, ensuite nous voulons tester toutes les différentes boucles disponibles en JavaScript grâce à une application qui est similaire à celle de l'exercice 6 du point de vue de l'insertion d'informations dans du HTML via JavaScript.

3.7.2 Explications

- Comment utiliser la boucle for en JavaScript

La boucle for s'utilise comme en Java de cette manière :

```
for (let i = 0; i < 5; i++) {
  text += "i = " + i + "<br>";
}
```

- Comment utiliser la boucle while en JavaScript

La boucle while s'utilise de cette manière :

```
while (i < 5) {
  text += "i = " + i + "<br>";
  i++;
}
```

- Comment utiliser la boucle doWhile en JavaScript

La boucle do while s'utilise ainsi :

```
do{
  text += "i = " + i + "<br>";
  i++;
}while(i<5);
```

3.7.3 Solutions

J'ai utilisé la troisième option proposé dans la donnée, c'est-à-dire le fait de faire un onload dans le body afin de lancer la fonction initCtrl() des le chargement de la page html.

Ensuite en me servant des id, j'ai ajouté un EventListener pour faire en sorte de lancer les fonctions respectives selon le bouton appuyé.

Voici le code html et JavaScript :

```
<body onload="initCtrl()">
  <div id="container">
    <p>Cliquez sur les boutons ci-dessous pour tester
      les différentes sortes de boucles en JavaScript.</p>
    <button id="for">For</button>
    <button id="while">While</button>
    <button id="do">Do while</button>
    <p id="info"> </p>
  </div>
</body>
```

JavaScript :

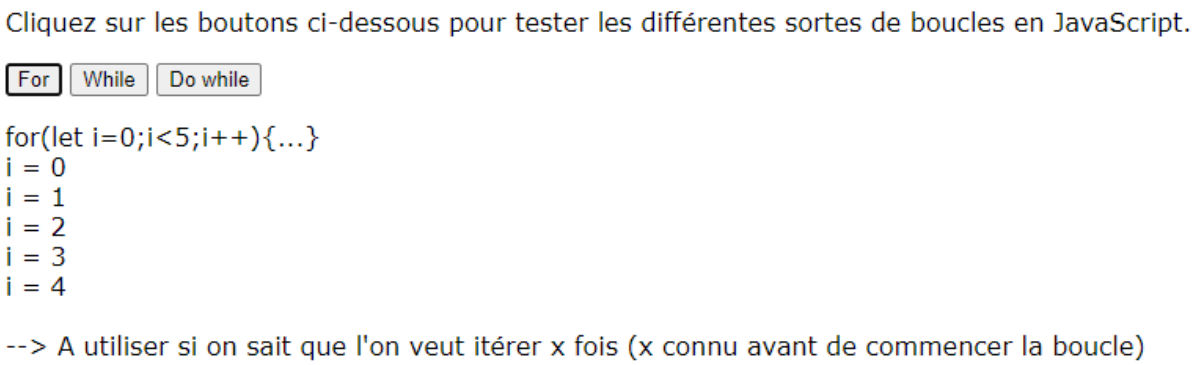
```
function initCtrl() {
  document.getElementById("for").addEventListener("click", testerFor);
```

```
  document.getElementById("while").addEventListener("click", testerWhile);
  document.getElementById("do").addEventListener("click", testerDoWhile);
}

function testerFor() {
  let text = "for(let i=0;i<5;i++){...}<br>";
  for (let i = 0; i < 5; i++) {
    text += "i = " + i + "<br>";
  }
  document.getElementById("info").innerHTML = text + "<br>" + "--> A utiliser
si on sait que l'on veut itérer x fois (x connu avant de commencer la
boucle)";
}

function testerWhile() {
  let text = "while(i<5){...}<br>";
  let i = 0;
  while (i < 5) {
    text += "i = " + i + "<br>";
    i++;
  }
  document.getElementById("info").innerHTML = text + "<br>" + "--> A utiliser
si on ne sait pas le nombre d'itérations au démarrage de la boucle";
}

function testerDoWhile() {
  let text = "do{...} while (i<5);<br>";
  let i = 0;
  do{
    text += "i = " + i + "<br>";
    i++;
  }while(i<5);
  document.getElementById("info").innerHTML = text + "<br>" + "--> A utiliser
si one ne sait pas le nombre d'itérations au démarrage de la boucle mais avec
un passage obligatoire";
}
```

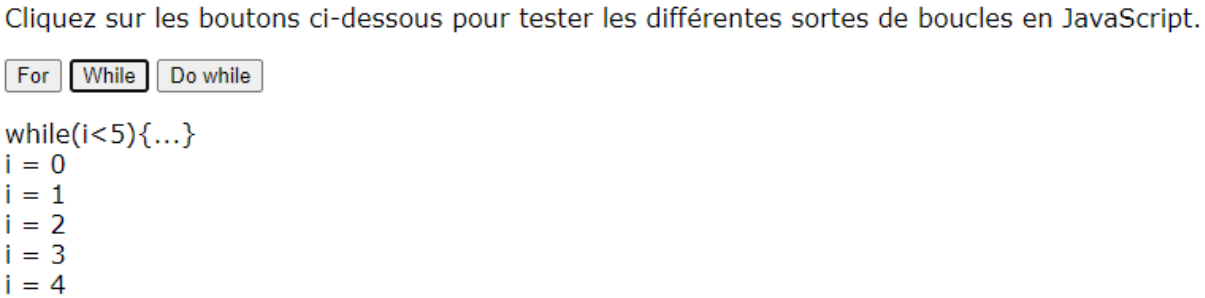
3.7.4 Captures

Voici les captures du résultat lorsque l'on appuie sur les trois boutons :

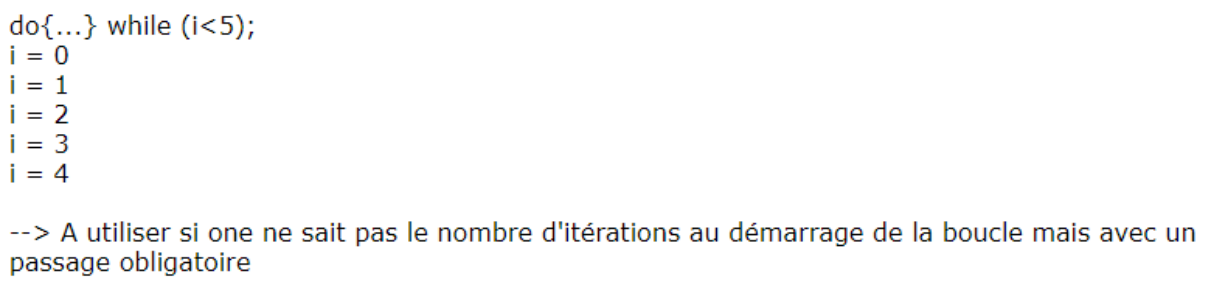
For :



While :



DoWhile :



3.8 Exercice 8

3.8.1 Objectif

En premier temps nous copions l'exercice 7 dans le dossier de l'exercice 8. Ensuite nous reconstitutions le code en testant une imbrication de 2 boucles sur les données d'un tableau de personnes défini par un objet de type JSON.

JSON est la structure de données la plus efficace pour JavaScript pour les raisons suivantes :

- Compréhensible pour les humains immédiatement
- Indépendante des langages informatiques
- Peut stocker plusieurs types de données
- Propose une structure en arborescence
- Très grande affinité entre un objet JavaScript et celui transcrite par le réseau

3.8.2 Explications

- Comment faire une boucle sur un tableau JSON et comment reprendre des informations de chaque itération de la boucle ?

Nous avons dû le faire dans cet exercice, en utilisant cette ligne :

```
    let personne = i + ". " + json.personnes[i].prenom + " " +
json.personnes[i].nom + " " + json.personnes[i].age;
```

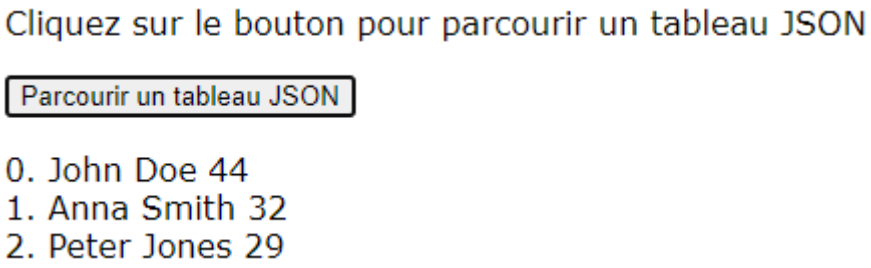
nous récupérons la case « i » du tableau json.personnes qui est une constante que nous avons créé auparavant. Ensuite nous pouvons utiliser ces informations comme nous le souhaitons.

3.8.3 Solutions

Voici le code Javascript que j'ai réalisé afin d'effectuer cet exercice :

```
const json = {
  personnes:[
    {prenom:"John",nom:"Doe",age:44},
    {prenom:"Anna",nom:"Smith",age:32},
    {prenom:"Peter",nom:"Jones",age:29}
  ]
};
function parcourirUnTableauJSON(){
  let text = "";
  for(let i=0;i<json.personnes.length;i++){
    let personne = i + ". " + json.personnes[i].prenom + " " +
json.personnes[i].nom + " " + json.personnes[i].age;
    text+= personne + "<br>";
  }
  document.getElementById("info").innerHTML = text;
}
```

3.8.4 Captures



3.9 Exercice 9

3.9.1 Objectif

Dans l'exercice 8 nous avons vu comment nous pouvons créer un objet possédant la notation JSON. Il existe cependant une 2ème manière de faire cela en créant une instance à partir de la fonction « Object ». Après nous pourrons définir des classes afin de pouvoir créer des objets, il en existe même 3 manières.

Nous verrons dans l'exercice les 3 manières de réaliser cela.

3.9.2 Explications

- Comment créer un objet avec la classe (selon la dernière manière décrite dans cet exemple)

Afin de pouvoir créer un objet avec la classe, nous utilisons la méthode de la classe mentionnée. Pour faire ceci, nous déclarons une classe qui dans sont constructeur demande les attributs nécessaires à cette classe. Puis nous pourrons définir les attributs comme en java avec le this.nom = valeur.

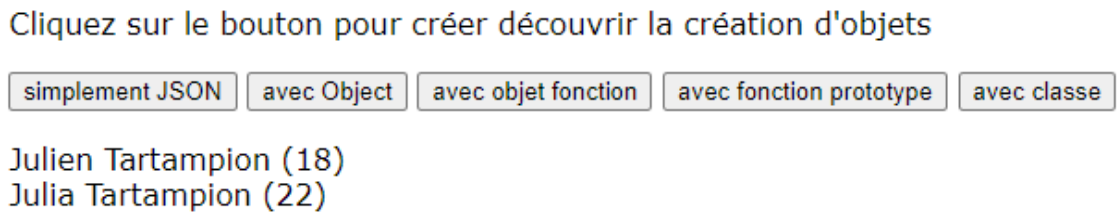
Nous pouvons également créer la méthode toString() par la suite, car elle va s'avérer utilise lorsque nous allons mettre un objet que nous avons créé sous forme d'un String quelconque. Cette méthode a pour but de transformer notre objet en String tout en prenant les attribut et en les mettant sous une forme précise.

3.9.3 Solutions

Voici un exemple :

```
class Eleve {
  constructor(prenom, nom, age) {
    this.prenom = prenom;
    this.nom = nom;
    this.age = age;
  }
  toString() {
    return this.prenom + " " + this.nom + " (" + this.age + ")";
  }
```

3.9.4 Captures
Voici un exemple d'utilisation :



3.10 Exercice 10
3.10.1 Objectif
Ici notre but est de construire une application qui nous permettra de gérer efficacement les informations d'une liste de personnes (prénom, nom, âge) grâce à des opérations « métier » comme « ajouter une personne » ou « supprimer une personne » que nous mettrons dans un fichier personne.js. Afin de gérer des personnes, il nous faut également disposer d'un bean « Personne » que nous allons stocker dans un fichier worker.js.

3.10.2 Explications
Voici un résumé de l'exercice avec les concepts clés :

Nous avons séparé les différents fichiers Javascripts afin que nous ayons un worker, un contrôleur et un bean. Cela nous a permis d'avoir une meilleure structure de fichier afin de ne pas nous emmêler les pinceaux. En premier temps nous avons défini le tableau de personnes dans le bean et avons écrit la méthode toString() :

```
function Personne(prenom, nom, age){
    this.prenom = prenom;
    this.nom = nom;
    this.age = age;
}

Personne.prototype.toString = function(){
    return this.nom + " " + this.prenom + " (" + this.age + ")";
}
```

Une fois cela fait nous avons enchainé avec le worker ou en premier temps nous avons créé le tableau avec les personnes déjà sensées être présentes. Avec le personnes.sort() cela va organiser de manière alphabétique les personnes dans le tableau.

Une fois cela fait nous créons les trois fonctions _trouverPersonne, ajouterPersonne et supprimerPersonne. Le premier utilise un for afin de savoir si la personne inscrite par l'utilisateur est déjà présente et renvoie -1 si elle n'est pas présente et un autre nombre s'il est déjà présent. Ensuite nous utilisons la fonction trouverPersonne afin de pouvoir en ajouter ou supprimer avec les autres méthodes.

La commande qui suit va permettre à ce que lorsque l'état du document change, que la méthode _afficherPersonne se relance afin d'afficher les nouvelles entrées ou de montrer qu'une entrée a été supprimé.

Et ensuite dans le contrôleur nous utilisons les méthodes que nous avons apprise jusqu'à maintenant.

3.10.3 Solutions

Voici les trois fichiers javascript :

Worker.js :

```
const personnes = [
    new Personne("John", "Doe", 44),
    new Personne("Anna", "Smith", 32),
    new Personne("Peter", "Jones", 29)
  ];

  personnes.sort();

  function _trouverPersonne(p){
    let idx = -1;
    for(i=0;i<personnes.length;i++){
        if(p.toString() == personnes[i].toString()){
            idx = i;
        }
    }
    return idx;
  }

  function ajouterPersonne(p){
    let idx = _trouverPersonne(p);
    if(idx == -1){
        personnes.push(p);
    }
    personnes.sort();
  }

  function supprimerPersonne(p){
    let idx = _trouverPersonne(p);
    if(idx != -1){
        personnes.splice(idx, 1);
    }
    personnes.sort();
  }
```

Personne.js :

```
function Personne(prenom, nom, age){
    this.prenom = prenom;
    this.nom = nom;
    this.age = age;
}

Personne.prototype.toString = function(){
    return this.nom + " " + this.prenom + " (" + this.age + ")";
}
```

indexCtrl :

```
document.onreadystatechange = function () {
  if (document.readyState === "complete") {
    _afficherPersonnes();
  }
}

function _afficherPersonnes() {
  let txt = "<ul>";
  for (let i = 0; i < personnes.length; i++) {
    txt += "<li>";
    txt += "<a href='#' onclick='selectionnerPersonne(" + i + ")'/>";
    txt += personnes[i].toString();
    txt += "</a>";
    txt += "</li>";
  }
  txt += "</ul>";
  document.getElementById("info").innerHTML = txt;
}

function _afficherInfosPersonnes(p) {
  document.getElementById("prenom").value = p.prenom;
  document.getElementById("nom").value = p.nom;
  document.getElementById("age").value = p.age;
}

function _lireInfosPersonne() {
  let p = new Personne(
    document.getElementById("prenom").value,
    document.getElementById("nom").value,
    document.getElementById("age").value
  )
  return p;
}

function selectionnerPersonne(i) {
  _afficherInfosPersonnes(personnes[i]);
}

function ajouter() {
  let p = _lireInfosPersonne();
  ajouterPersonne(p);
  _afficherPersonnes();
}

function supprimer() {
  let p = _lireInfosPersonne();
  supprimerPersonne(p);
```

```
  _afficherPersonnes();
}
```

3.10.4 Captures

- Doe John (44)
- Franca Nathan (22)
- Jones Peter (29)
- Smith Anna (32)

3.11 Exercice 11

3.12 Objectif

Le challenge de cet exercice est de réussir à recréer l'exercice 10 mais en utilisant des « classes » pour les fichiers Javascript que nous avions créés.

Ensuite, la seule grande difficulté est de savoir comment créer un objet contrôleur (ctrl) pour qu'il soit disponible pour la vue ? On vous donne ici la solution : l'objet « ctrl » doit être stocké dans l'objet « window » du navigateur car les méthodes de l'« indexCtrl » sont appelées depuis l' « index.html ». L'objet window doit être ajouté à la variable ou on ne met rien et la variable devient globale, surtout pas de let !

3.12.1 Explications

- Une explication sur comment créer une classe en Javascript.

Afin de créer une classe en Javascript, cela se rapproche de Java. Il faut un constructeur et nous pouvons y implémenter des méthodes mais ne devons pas utiliser function.

Ensuite pour toutes les variables ainsi que les méthodes qui sont présentes dans la même classe, nous devons utiliser le mot clé this.nom et si la méthode ou la variable se situe dans une autre classe il faut faire this.nomClase.nom.

- Comment instancier une classe pour qu'elle soit accessible dans la vue, le cas échéant, un HTML.

Il ne faut pas oublier de préciser dans quelle classe la méthode que nous voulons appeler se situe. Par exemple :

```
            <input type="button" value="Ajouter" onclick="ctrl.ajouter();">
            <input type="button" value="Supprimer"
onclick="ctrl.supprimer();">
```

Ici nous précisons avec le ctrl.ajouter et supprimer que ils se situent dans le contrôleur.

3.12.2 Solutions

Voici les Javascripts :

Worker.js :

```
class Worker{
constructor(){
  this.personnes = [
    new Personne("John", "Doe", 44),
    new Personne("Anna", "Smith", 32),
    new Personne("Peter", "Jones", 29)
  ];
  this.personnes.sort();
}

trouverPersonne(p){
    let idx = -1;
    for(let i=0;i<this.personnes.length;i++){
        if(p.toString() == this.personnes[i].toString()){
            idx = i;
        }
    }
    return idx;
  }

ajouterPersonne(p){
    let idx = this.trouverPersonne(p);
    if(idx == -1){
        this.personnes.push(p);
    }
    this.personnes.sort();
  }

supprimerPersonne(p){
    let idx = this.trouverPersonne(p);
    if(idx != -1){
        this.personnes.splice(idx, 1);
    }
    this.personnes.sort();
  }
}
```

Personne.js :

```
class Personne{
  constructor(prenom, nom, age){
    this.prenom = prenom;
    this.nom = nom;
    this.age = age;
  }
```

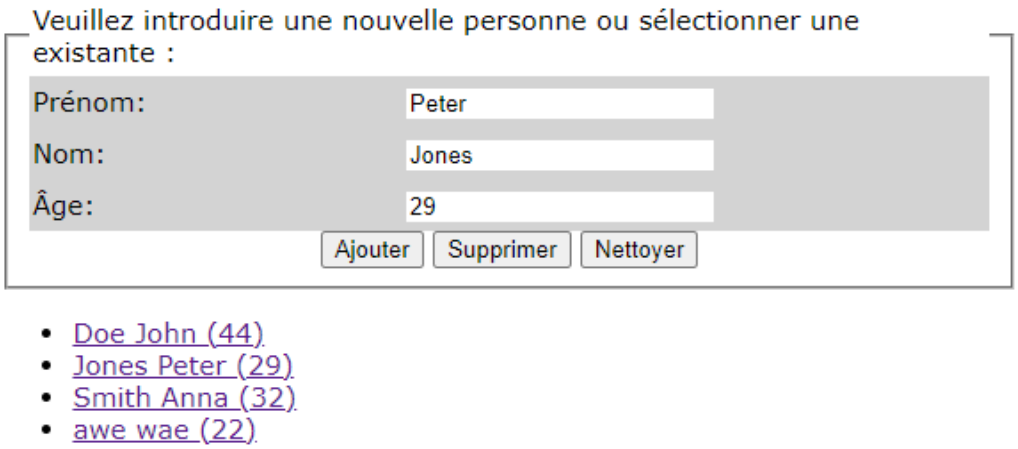
```
toString(){
    return this.nom + " " + this.prenom + " (" + this.age + ")";
}
}
```

IndexCtrl :

```
document.onreadystatechange = function () {
  if (document.readyState === "complete") {
    window.ctrl = new Ctrl();   // ou ctrl = new Ctrl();
    ctrl.afficherPersonnes();
  }
};

class Ctrl {
  constructor() {
      this.wrk = new Worker();
}

afficherPersonnes() {
  let txt = "<ul>";
  for (let i = 0; i < this.wrk.personnes.length; i++) {
    txt += "<li>";
    txt += "<a href='#' onclick='ctrl.selectionnerPersonne(" + i + ")'/>";
    txt += this.wrk.personnes[i].toString();
    txt += "</a>";
    txt += "</li>";
  }
  txt += "</ul>";
  document.getElementById("info").innerHTML = txt;
}

afficherInfosPersonnes(p) {
  document.getElementById("prenom").value = p.prenom;
  document.getElementById("nom").value = p.nom;
  document.getElementById("age").value = p.age;
}

lireInfosPersonne() {
  let p = new Personne(
    document.getElementById("prenom").value,
    document.getElementById("nom").value,
    document.getElementById("age").value
  )
  return p;
}

selectionnerPersonne(i) {
  this.afficherInfosPersonnes(this.wrk.personnes[i]);
}
```

```
ajouter() {
  let p = this.lireInfosPersonne();
  this.wrk.ajouterPersonne(p);
  this.afficherPersonnes();
}

supprimer() {
  let p = this.lireInfosPersonne();
  this.wrk.supprimerPersonne(p);
  this.afficherPersonnes();
}
}
```

3.12.3 Captures



3.13 Exercice 12

3.13.1 Objectif

Il nous faut transformer les fonctions qui suivront en dessous (val 1 à val 4) en additionneur en ajoutant un seul paramètre en l'additionnant à val :

Une fonction est premièrement définie et ensuite exécutée une à plusieurs fois. Voici trois manières d'écrire une fonction JS :

```
// Déclaration d'une fonction
function a() {
  let val = 1;
  console.log(val) ;
}
a();    // => exécution de la méthode et affichage de 1 dans la console
   // si on écrit a(); avant la déclaration de la fonction, cela fonctionne !


// Déclaration d'une expression fonction (fonction anonyme)
let b = function() {
  let val = 2;
  console.log(val) ;
} ;
b();    // => exécution de la méthode et affichage de 2 dans la console

// Déclaration d'une fonction flèche (arrow function) ; 3 cas de figure
let c = (a,b) => a + b ;
console.log(c(2,5)) ;  // = 7

let d = val => val*val ;  // avec un paramètre
console.log(d(5));    // = 25

let e = (a,b) => {    // avec plusieurs instructions
 let somme = a+b;
 return somme;
};
console.log(e(5,7));   // = 12
```

Il est, des fois, nécessaire de créer une fonction et l'exécuter directement : IIFE *« Immediately-Invoked Function Expression».* Voici deux manières d'écrire une fonction JS IIFE:

```
(function() {
  let val = 3;
  console.log(val) ;
})() ;    // => exécution et affichage de 3 dans la console


(() => {
  let val = 4;
  console.log(val) ;
})() ;  // => exécution et affichage de 4 dans la console via une fonction
flèche
```

3.13.2 Explications

- Quels sont les 3 types de déclarations de fonction possibles en Javascript. Donnez pour chacun un ou plusieurs exemples.

1.

```
function nom(){

}
```

2.

```
let nom = function(){

};
```

3.

```
(function(){

})();
```

3.13.3 Solutions

Nous avons tout simplement dû additionner un nombre à la variable 1 à 4 et ensuite faire un console.log.

3.14 Exercice 13

3.14.1 Objectif

Dans cet exercice nous récupérons un code donné dans la donnée et ensuite nous devons stocker le prénom ainsi que le nom de l'utilisateur qui aura été saisi dans les champs Prénom et Nom du formulaire dans un cookie. Lorsque l'utilisateur appuie sur le bouton enregistrer, cela déclenche l'enregistrement du cookie.

3.14.2 Explications

- Qu'est-ce qu'un cookie et à quoi sert-il ?

Les cookies se présentent comme de petits fichiers texte qui peuvent entreposer un nombre limité de données. Cette technique est très précieuse car nous pouvons conserver des contenus de variables mémoires réutilisables au fil de la navigation au travers de nombreuses pages de notre site.

- Comment créer un cookie et stocker une chaîne de caractère. Donner un exemple.

Afin de créer un cookie ainsi que stocker une chaine de caractère, nous pouvons procéder de la manière suivante :

```
  document.cookie = "cookie_exercice13=" + contenu + "; expires=" +
dateExpiration + "; path=/";
```

Stocker une chaine de caractère :

```
let personneJson = {
    prenomJson: prenom,
    nomJson: nom,
  };
  personneJson = personneJson.prenomJson + " " + personneJson.nomJson;
  setCookie(personneJson, 3);
```

- Comment récupérer une chaîne de caractère depuis un cookie. Donner un exemple.

Nous pouvons utiliser la méthode split, ce qui va récuperer la chaine de caractères.

```
    // Test si le cookie existe
if (document.cookie.length > 0) {
    // Récupérer le contenu du cookie et en extraire la partie qui se situe
après le "="; Vous pouvez utiliser la méthode "split"
    // liée aux chaînes de caractères.
    let fistTab = document.cookie.split("=")
    let temp = fistTab[1];
}
```

- Une explication du code à utiliser pour convertir un objet JSON en chaîne de caractères et pour convertir une chaîne de caractères en objet JSON.

Voici le code permettant de convertir un objet JSON en une chaine de caractères.

```
      //Création d'un tableau contenant toutes les parties de la chaîne de
caractères. Ex : "Salut toi" -> tab[salut][toi]
    let tab = personne.split(" ");
      //création d'un objetJson en prenant les cellules de notre tableau.
    let personneJson = {
      prenom: tab[0],
      nom: tab[1],
    };
```

Et dans l'autre sens :

```
let personneJson = {
    prenom: prenomPsn,
    nom: nomPsn,
  };
  // Convertir l'objet json "personneJson" en chaine de caractère.
Let personne = personneJson.nom + " " + personneJson.prenom;
```

Ou alors, nous pouvons utiliser les méthodes json.stringify ou json.parse.

3.14.3 Solutions

Voici le fichier main.js :

```
document.onreadystatechange = function () {
  // Code exécuté lorsque la page a terminé d'être chargée.
  if (document.readyState === "complete") {
    // Chargement de l'éventuelle cookie.
    getCookie();
    // Ajout d'un écouteur "click" sur le bouton "enregistrer".
    document.getElementById("enregistrer").addEventListener("click",
saveData);
  }
};

function saveData() {
  // Récupérer les valeurs contenues dans les champs "prenom" et "nom".
  //
  // Créer un objet json "personneJson" contenant le prénom et le "nom".
  //
  // Convertir l'objet json "personneJson" en chaine de caractère.
  //
  // Appeler la méthode "setCookie" en passant en paramètre l'objet
"personneJson" converti et en précisant le délai de 3 minutes.
  //
  let prenom = document.getElementById("prenom").value
  let nom = document.getElementById("nom").value
  let personneJson = {
    prenomJson: prenom,
    nomJson: nom,
  };
  personneJson = personneJson.prenomJson + " " + personneJson.nomJson;
  setCookie(personneJson, 3);
}

function setCookie(contenu, minutes) {
  // Générer une date d'expiration. Il doit s'agir de la date et l'heure
actuelle + le nombre de minute passé en paramètre.
  //
  let dateExpiration = new Date();
  dateExpiration.setTime(dateExpiration.getTime + minutes * 60 * 1000);
```

```
  // Créer le cookie.
  document.cookie = "cookie_exercice13=" + contenu + "; expires=" +
dateExpiration + "; path=/";
  console.log(contenu);
  console.log("Data saved in cookie");
}

function getCookie() {
  // Tester la présence du cookie. Pour cela, il faut tester si la taille du
cookie est supérieure à 0.
  if (document.cookie.length > 0) {
    // Récupérer le contenu du cookie et en extraire la partie qui se situe
après le "="; Vous pouvez utiliser la méthode "split"
    // liée aux chaînes de caractères.
    let temp = document.cookie.split("=");
    let contenuApresEgal = temp[1];
    //
    // Convertir la chaîne de caractères en objet Json.
    //
    let tab = contenuApresEgal.split(" ");
    let personneJson = {
      prenom: tab[0],
      nom: tab[1],
    };
    // Charger les deux champs avec la valeur du prénom et du du nom contenu
dans l'objet json.
    //
    let prenom = document.getElementById("prenom").value =
personneJson.prenom;
    let nom = document.getElementById("nom").value = personneJson.nom;
    console.log("Data retrieved from cookie");
  }
}
```

3.14.4 Captures

3.15 Exercice 14

3.15.1 Objectif

Dans cet exercice nous allons voir les bases du JQuery. Javascript possède 3 problèmes qu'il ne résout pas bien :

- On ne peut manipuler qu'un objet du DOM à la fois (donc il faut écrire un nombre de lignes incroyable pour modifier par exemple tout une série de <LI>...</LI>) ;
- Il est extrêmement verbeux (on doit écrire beaucoup pour obtenir peu) ;

- On doit parfois écrire plusieurs codes différents suivant les navigateurs.

3.15.2 Explications

- Expliquez brièvement ce qu'est jquery et à quoi il sert.

JQuery est une bibliothèque Javascript qui est très utilisée et appréciée dans le monde des programmeurs web. Elle peut résoudre la majorité des problèmes énoncés dans le point précédent.

Nous pouvons accéder aux objects du DOM en utilisant sa fonction $() ou JQuery(), cela nous permet également de gérer des évènements ainsi que de produire des effets graphiques ou encore d'implémenter AJAX.

- Comment reconnaît-on du code jquery ?

Avec le $() ou le JQuery().

- Quelles sont les 2 tâches principales de jquery ?

On peut accéder aux objets do DOM, gérer les évènements et produire des effets graphiques ainsi qu'implémenter AJAX.

- Expliquer comment trouver des éléments du DOM avec jquery selon un nom de tag, une classe CSS ou un id.

```
$("#info").html("<b>C'est bold</b>");
```

Ici par exemple nous implémentons du code html grâce à l'id « info ».

- Décrivez la fonction jquery qui est appelée lorsque la page a été chargée. Donnez un exemple.

L'évènement ready :

```
$(document).ready(function(){
    // Votre code jQuery doit se trouver ici
});
```

Nous permet de traverser et de manipuler une page HTML, mais attendons d'abord qu'elle soit chargée.

- Comment retrouvez un élément parent, le suivant, le précédent, un enfant ou les éléments frères avec jquery ?

Nous pouvons trouver un élément parent en procédant comme suit :

```
$("button").parent().css("border", "3px solid red");
```

Nous cherchons un bouton et disons que sont élément parent obtiendra les éléments css que l'on précise :

- Comment:
  - Modifier le css d'un élément de l'élément sélectionné ?

Il y'a plusieurs manières possibles, notamment avec le .css que nous venons de voir, mais si nous voulons en exécuter plusieurs :

```
$("li a").css({
  color: "red",
```

```
  fontWeight: "bold"
});
```

- Ajouter un élément avant ou après l'élément sélectionné ?

.next et .prev sont les commandes qui nous permettent d'effectuer cela.

- Supprimer un élément ?

Pour cela nous utilisons la commande .remove.

- Gérer les événements (click, hover, etc.)

```
$("a.menu").click(function(){
  $(this).next().toggle();
  return false;
});
```

```
$("li").hover(function(){
  $(this).animate({marginLeft: 38, marginRight: 0});
}, function(){
  $(this).animate({marginLeft: 18, marginRight: 18});
});
```

- Cacher ou afficher un élément ?

Nous utilisons dans ce cas le .hide ou le .show.

- Comment lire un fichier avec ajax ?

Nous pouvons par exemple lire un fichier xml en procédant de cette manière :

```
$.ajax({
    url: "file.xml",
    success: function( xml ) {
        $(xml).find("tab").each(function(){
            $("ul").append(
              "<li>" + $(this).text() + "</li>");
        });
    }
});
```

- Comment lire un fichier JSON avec la méthode $.getJSON()?

```
$.getJSON("file.json", function( obj ) {
    for ( var prop in obj ) {
        $("ul").append(
            "<li>" + prop + ": " + obj[prop] + "</li>");
    }
});
```

- Comment charger un fichier HTML dans un div existant ?

Nous pouvons faire cela en utilisant le .load :

```
$("div.load").load("file.html");
```

- Comment intégrer jquery dans vos pages ?

```
<script src='http://code.jquery.com/jquery.js'></script>
```

3.16 Exercice 15

3.16.1 Objectifs

En premier temps nous allons copier notre exercice 1 dans le nouveau dossier. Notre but va consister de créer une vue avec le code HTML donnée dans l'exercice. Ensuite nous devons gérer la vue avec l'un ou l'autre des 2 contrôleurs qui sont proposés.

3.16.2 Explications

- Comment récupérer la valeur d'un champ avec jquery.

Afin de faire cela nous utilisons la méthode .val(), il y aura un exemple donné plus tard.

- Comment une propriété css avec jquery.

Pour récuperer une propriété css avec Jquery, nous allons utiliser .css(NomPropriété).

- Comment utiliser une animation avec jquery.

Pour cela nous utilisons : .NomAnimation(paramètres)

3.16.3 Solutions

Voici la solution :

En premier temps nous récupérons une propriété puis la changeons en ajoutant un deuxième paramètre :

```
$("p").css("background-color", "yellow");
```

Voici comment nous récuperons la valeur d'un champ :

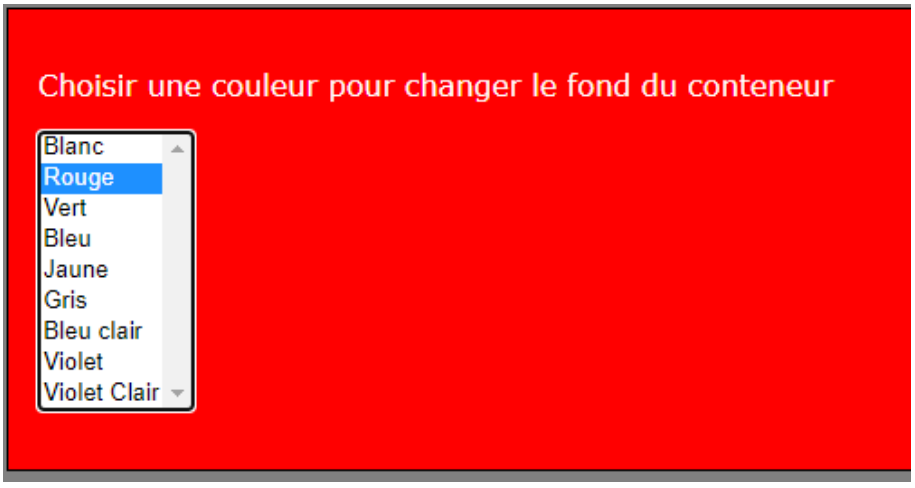
```
$("#couleurs").val();
```

Et finalement voici un exemple d'une animation avec JQuery :

```
$("#container").fadeOut("1") ;
```

3.16.4 Captures

Résultat de l'exercice 15 :



3.17 Exercice 16

3.17.1 Objectif

Dans cet exercice nous allons devoir être capable de traiter plusieurs textes en leur ajoutant des écouteurs. Pour faire cela nous avons utilisé la fonction : on, de Jquery. Par la suite nous devions faire disparaitre et apparaitre des textes, parfois avec un fondu ou alors en leur ajoutant des classes. C'était globalement un exercice très complet qui traitait les points capitaux du Jquery.

3.17.2 Explications

Afin de pouvoir sélectionner des éléments du DOM, il nous suffit d'utiliser les sélecteurs CSS dans $(). Ensuite afin d'ajouter ou supprimer une classe à un élément, il nous suffit d'utiliser la fonction Jquery addClass ou removeClass. Cela est très pratique afin de pouvoir changer du CSS à un moment précis.

Plus tard nous avons aussi dû être capable d'ajouter des éléments. Afin de faire cela nous avons utilisé la fonction append. La fonction attr quant à elle, permet de récupérer la valeur d'un attribut d'un élément, cela est pratique afin de par exemple connaitre une PK.

3.17.3 Solutions

Voici un exemple d'écouteur :

```
$("#btnShow").on("click", function () {
  afficher();
});
```

Voici un exemple de fondu :

```
$("#div_3").fadeOut();
```

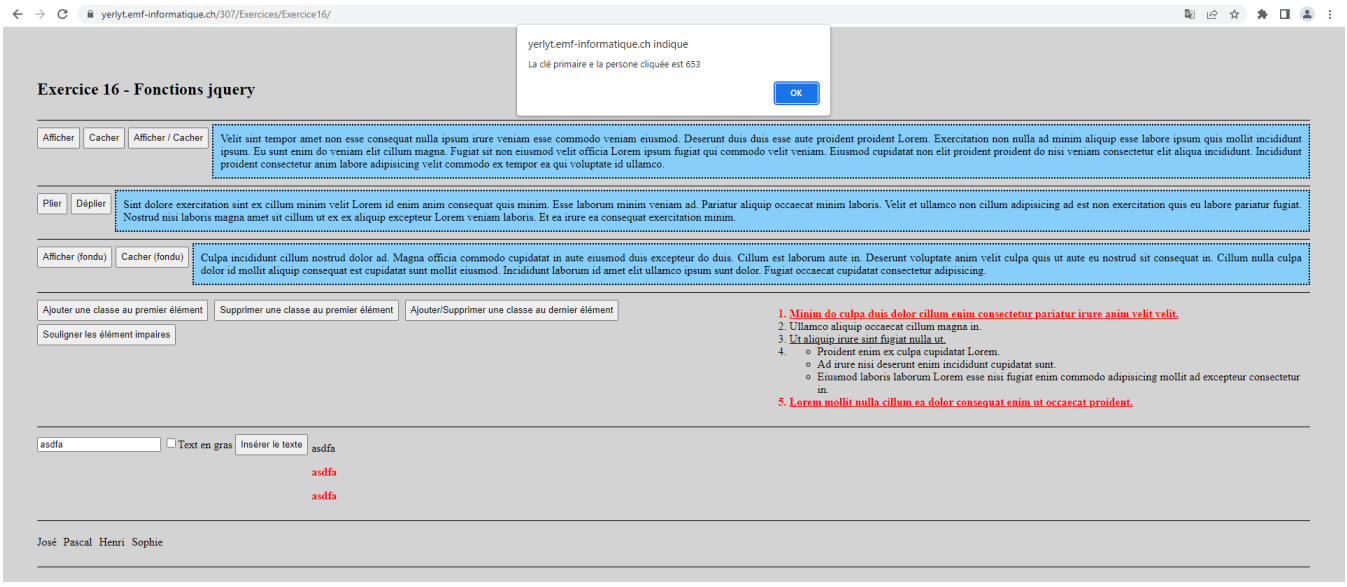
Voici un exemple d'ajout de classe :

```
        $("li").first().addClass("boldBlueText");
```

Voici un exemple de récupération d'attribut :

```
$(this).attr("pk")
```

3.17.4 Captures



3.18 Exercice 17

3.18.1 Objectif

Dans cet exercice nous avons dû voir certaines notions des API's et avons fait des présentations à ce sujet. (En gros découvrir les WebServices).

3.18.2 Explications

Il y a deux moyens d'utiliser les webservices, avec SOAP ou REST. Ce sont deux architectures qui permettent d'utiliser des API. Dans ce module, nous allons beaucoup être confronté aux APIs, principalement pour notre projet, alors c'est pour cela que cet exercice est particulièrement important. Afin de bien utiliser une API, il nous faut faire une requête :

Voici les différents types de requêtes :

3.18.3 Requête GET

Le GET nous permet de récupérer des données. C'est la requête que nous allons le plus souvent utiliser de toutes. Afin de l'effectuer, il nous faut plusieurs paramètres comme l'url de l'API ainsi que ses paramètres.

3.18.4 Requête POST

La requête POST nous permet d'insérer des données dans notre API. Nous l'utilisons pour stocker les identifiants par exemple. Une requête POST est composée de plusieurs paramètres, comme l'entête qui lui permet de définir le type de format de ses données, il y a également les autorisations qui permettent à l'utilisateur de passer en fonction de sa clé et ensuite il y a également le body qui contient les données qu'elles soient sous un format texte ou clé valeur.

3.18.5 Requête PUT

Cela nous permet de modifier des données dans une API, elle est par contre beaucoup moins utilisée que la requête POST. Afin de pouvoir l'effectuer nous devons donner un filtre et choisir ce que l'on va modifier puis donner le contenu que nous allons ajouter.

3.18.6 Requête DELETE

La requête delete comme son nom l'indique, permet d'effacer des données dans une API. Afin de pouvoir l'utiliser il nous faut donner un filtre sur ce que nous voulons supprimer.

3.18.7 Présentations

3.19 Exercice 18

3.19.1 Objectif

Nous sommes enfin prêts à faire de vraies requêtes entre client et serveur avec la maitrise de la structure MVC d'une application avec jQuery.

Deux web services PHP nous sont donnés, le premier de type POST qui retourne du XML et le second qui est de type GET qui retourne du JSON. En temps normal nous favorisons la deuxième technique car l'appel du service web ne modifie pas la base de données et que nous pouvons l'appeler aussi souvent que nous le voulons, nous devons utiliser GET et la réponse JSON est plus simple à traiter en JavaScript.

3.19.2 Explications

Nous allons faire un convertisseur de degrés Celsius en fahrenheit. Pour faire cela nous utilisons ajax et nous faisons soit une requête post ou alors une requête get. Il nous suffit de faire $.ajax(url : {objetJson}) afin d'effectuer cela. Il nous faut bien évidemment changer les paramètres de l'objet en fonction de la requête que nous désirons effectuer.

Grâce à ajax nous pouvons spécifier une requête en cas d'erreur ou de succès, pour ce faire il nous faut ajouter les paramètres success et error et spécifier quelles méthodes doivent être utilisées.

Nous pouvons tout de même avoir des erreur alors que notre programme est fonctionnel. Ceci est dû au CORS. C'est un mécanisme qui nous oblige a rajouter des entêtes http afin d'avoir accès aux ressources. Pour pouvoir l'outrepasser, il faut ajouter ceci dans le PHP :

```
header('Access-Control-Allow-Origin: *');
```

3.19.3 Solutions

Voici les fichiers Javascripts que j'ai fait :

indexCtrl :

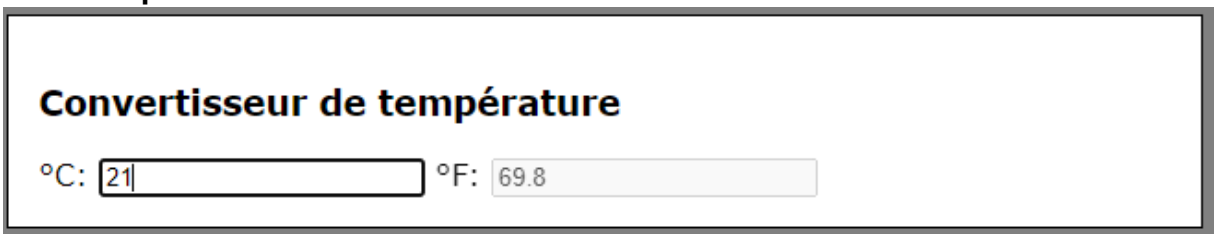
```
$().ready(function () {

  ctrl = new Ctrl();
  httpServ = new HttpServ();
});

class Ctrl {
  constructor() {
  }

  celsius2Fahrenheit() {
    // Lire les degrés du formulaire
    let degres = $("#celsius1").val();
    httpServ.celcius2Fahrenheit(degres,this.OKCelsius2Fahrenheit,
this.KOCelsius2Fahrenheit);
  }

  KOCelsius2Fahrenheit(xhr) {
    let erreur = xhr.status + ': ' + xhr.statusText
    alert('Erreur - ' + erreur);
  }

  OKCelsius2Fahrenheit(data) {
    let temp = $(data).find("temperature").text();
    let affiche = temp !== "NaN" ? temp : "";
    // Afficher les degrés dans le formulaire
    $("#fahrenheit1").val() = affiche;
  }
}
```

httpServ :

```
class HttpServ  {

  constructor(){
  }

  celcius2Fahrenheit(degres, successCallback, errorCallback) {
```

```
    let url = "http://francan.emf-
informatique.ch/MODULE_307/exercices/Exercice_18/js/convert_temp_p_xml.php";
    let param = "Temperature=" + degres + "&FromUnit=C&ToUnit=F";

    // envoi de la requête
    $.ajax(url, {
      type: "POST",
      contentType: "application/x-www-form-urlencoded; charset=UTF-8",
      data: param,
      success: successCallback,
      error: errorCallback
    });
  }
}
```

3.19.4 Captures



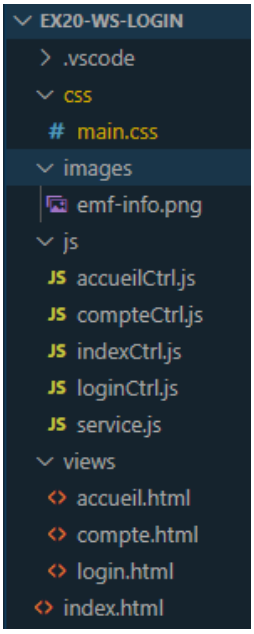
3.20 Exercice 20

3.20.1 Objectif

Cet exercice est basé sur l'exercice 3, nous allons découper l'application en vue en laquelle chaque vue a son propre contrôleur. Les vues seront chargées dynamiquement et l'URL ne va pas changer, à la méthode d'une application SPA (Single Page Application).

3.20.2 Explications

Dans cet exercice nous allons devoir être capable de créer une application de login grâce au PHP. Afin de faire cela nous avons dû modifier les erreurs du projet qui nous avait été donné pour pouvoir le transformer en application fonctionnelle. L'exercice devait êtreréalisé de cette manière :



Les pages doivent être loadées avec des méthodes javascript au bon moment.

- Que signifie SPA ? Expliquez le concept.

C'est une application web qui ne charge qu'un seule et unique document et qui va le mettre à jouer grâce à des API's javascript. Cela permet d'augmenter la fluidité du programme.

- Quelle méthode de Jquery pouvez-vous utiliser pour charger le code HTML d'une page, dans un élément HTML d'une autre page. Donnez un exemple.

Nous pouvons utiliser la méthode load afin d'effectuer cela. (Exemple au point suivant.)

- A quoir sert la méthode ajax $.ajaxSetup() ?

Cette méthode définit les valeurs par défaut pour les futures requêtes AJAX.

3.20.3 Solutions

Voici l'utilisation d'Ajax.

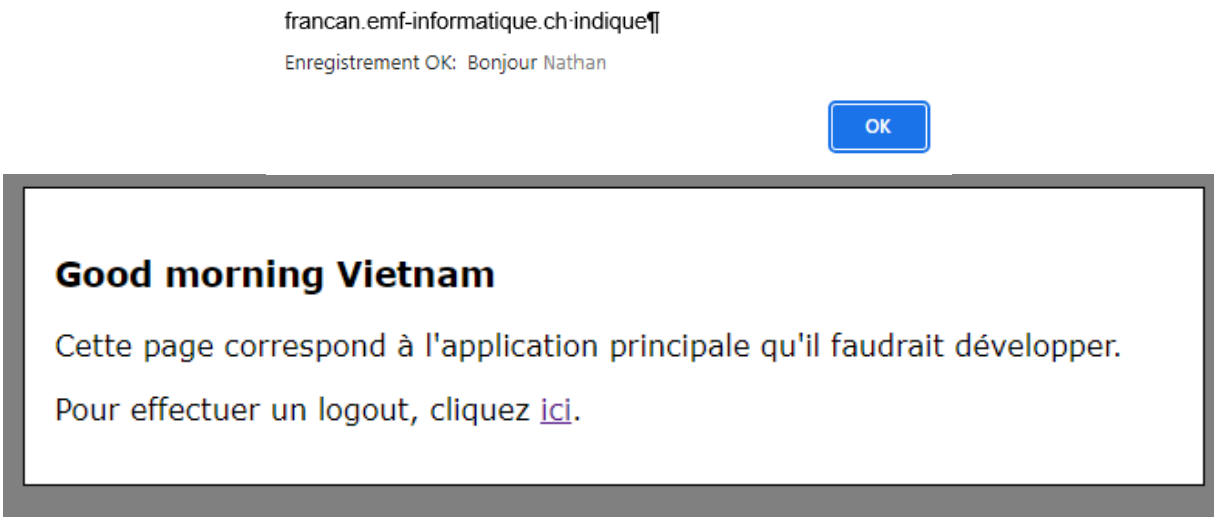
```
  login(identifiant, successCallback) {
  // Uploade votre propre fichier PHP et adaptez l'URL ci-dessous.
    let url = "https://yerlyt.emf-
informatique.ch/307/Exercices/Exercice20/php/login20.php";
    let param = "username=" + identifiant.username +
      "&password="+identifiant.password + "&domaine=" + identifiant.domaine +
      "&mail="+identifiant.mail+ "&langue="+ identifiant.langue;

    // envoi de la requête
    $.ajax(url, {
      type: "POST",
      contentType: "application/x-www-form-urlencoded; charset=UTF-8",
      data: param,
      success: successCallback
    });
  }
```

Et voici un exemple d'utilisation de load :

```
$("#view").load("views/" + vue + ".html")
```

3.20.4 Captures

Ci-dessous voici les réponses que nous recevons lorsque nous créons un utilisateur ou alors lorsque nous essayons de se connecter avec un autre utilisateur :



4 Projet

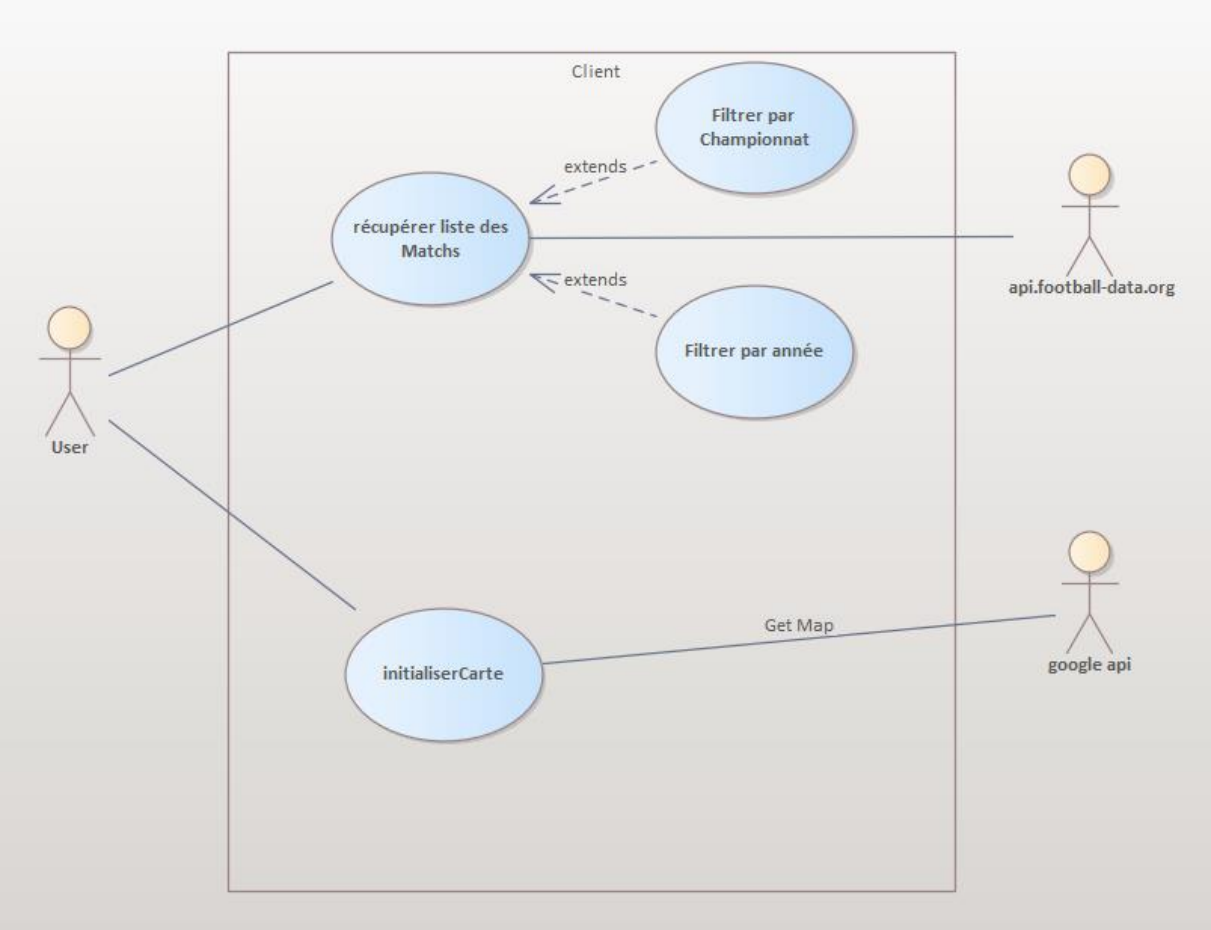
4.1 Introduction du projet

Étant un grand fan de Football, j'ai voulu m'inspirer de cet univers afin d'effectuer mon projet. Ensuite en effectuant quelques recherches sur des API's j'en ai trouvé quelques-unes qui me paraissaient très correspondantes, alors j'ai décidé d'en faire mon projet.

4.2 Analyse

4.2.1 Use-case

Voici le diagramme Use-Case qui est censé fournir une entrevue du projet et de comment nous allons procéder. Malheureusement avec mon retard je l'ai seulement fait à la fin.

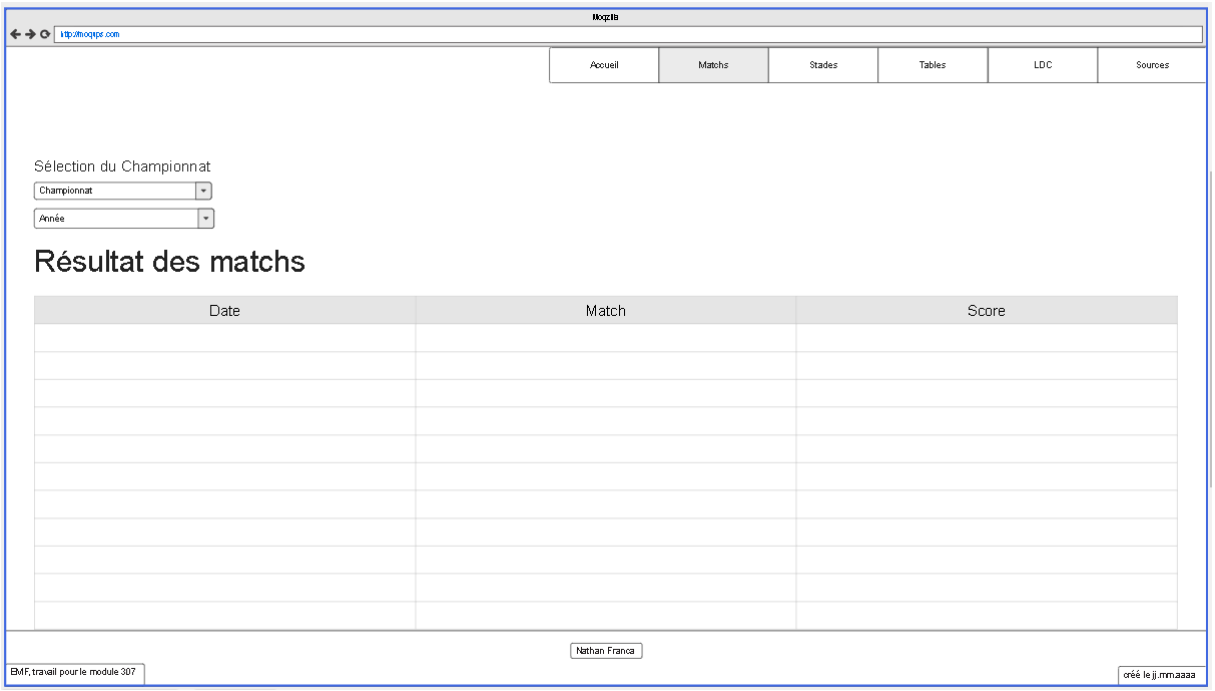


4.2.2 Maquette

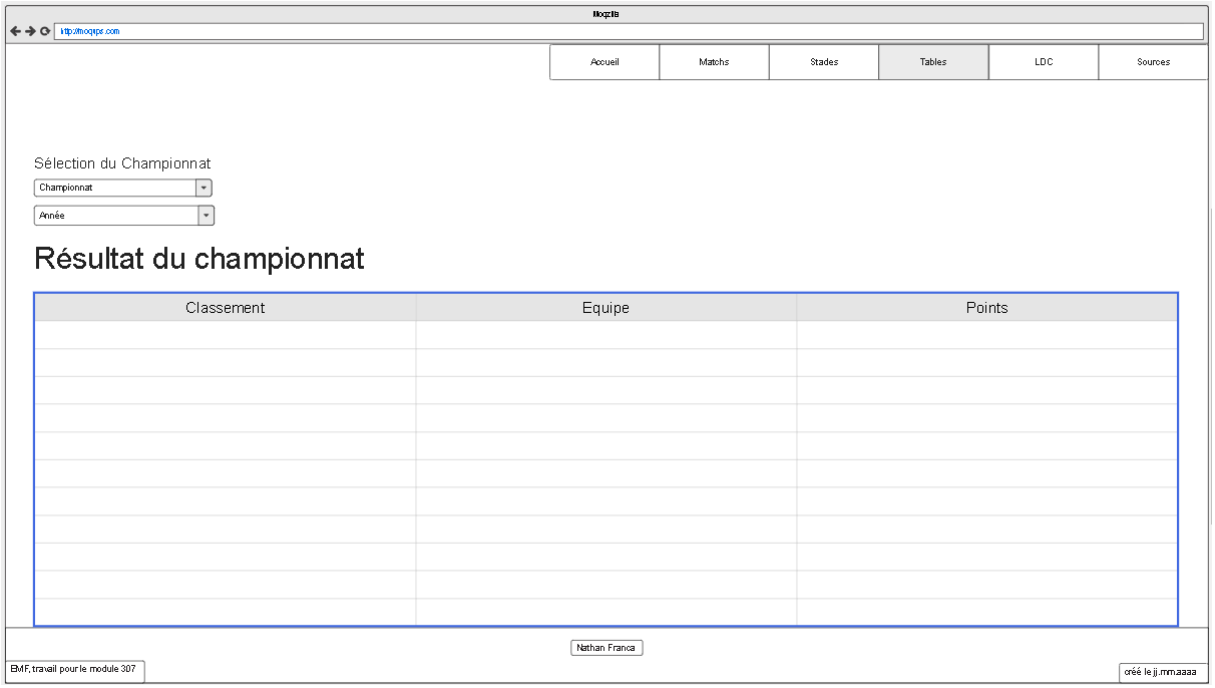
Voici les maquettes que j'ai réalisé pour les différentes pages du site Internet.

Accueil :

Matchs :



Tables :



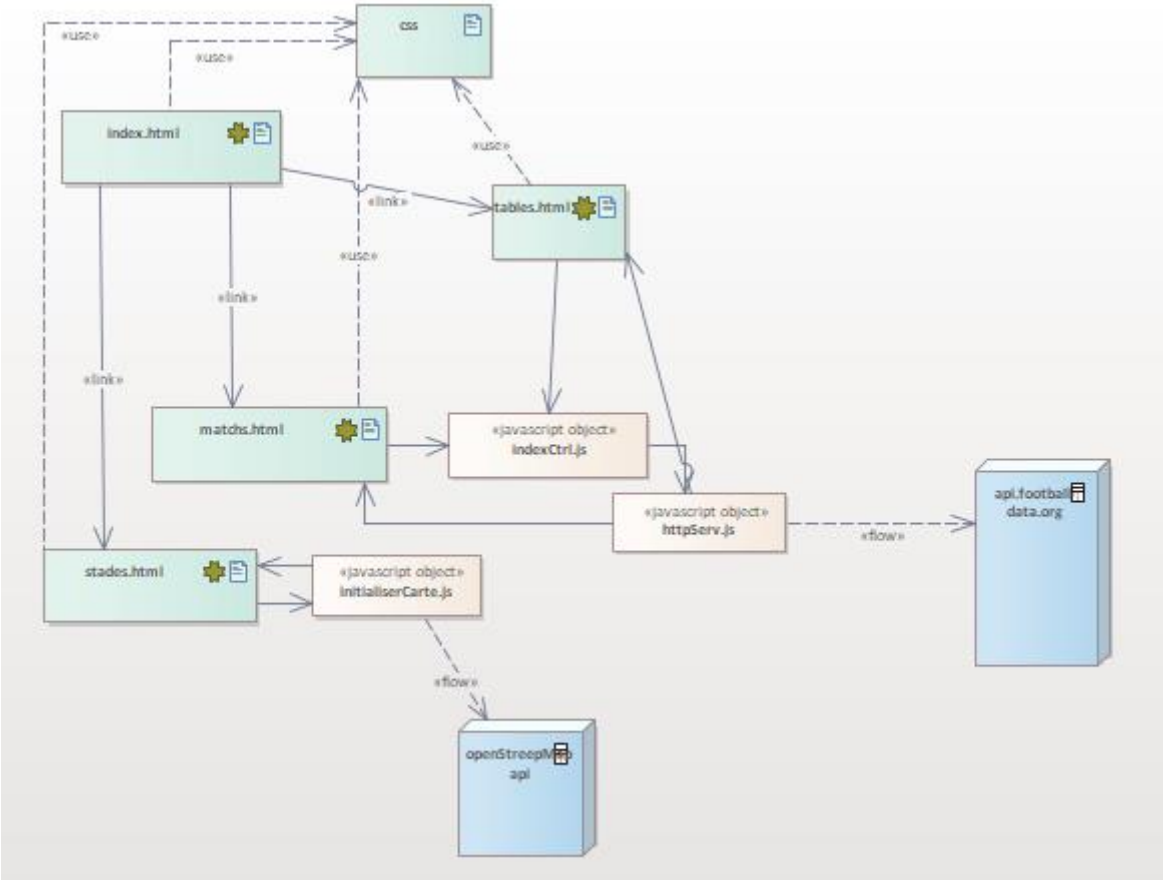
Stades :


Accueil
Matchs
Stades
Tables
LDC
Sources
Stades d'Angleterre
Nathan Franca
EMF, travail pour le module 307

4.3 Conception

4.3.1 Diagramme de navigation

Ensuite une fois mon application terminée avec toutes les pages html, css et Javascript présente, je peux effectuer un diagramme de navigation qui permet d'avoir une encore meilleure vue de notre site avec les liens et les appels entre les fichiers etc.



4.4 Implémentation

4.4.1 Extraits de code

Voici quelques extraits de code de mon site, logiquement je ne peux pas tout mettre ici, cependant sur Github on peut retrouver l'entièreté de mon site avec les extraits de code nécessaires (https://github.com/nffranca/Site_307).

Voici donc le fichier JavaScript initialiserCarte() :

```
/*
 * Contrôleur de la vue "stades.html".
 * @author França Nathan
 * @version 2.0 / 02.10.2023
 *
 */
function initialiserCarte() {
  const RedMarkerIcon = L.icon({
    iconUrl: "../images/redmarkericon.png",
    iconSize: [18, 30],
    iconAnchor: [9, 30],
    popupAnchor: [0, -20],
  });
  const GreenMarkerIcon = L.icon({
    iconUrl: "../images/greenmarkericon.png",
    iconSize: [18, 30],
    iconAnchor: [9, 30],
    popupAnchor: [0, -20],
  });
  const mapid = L.map("mapid").setView([53.4183726,-2.6701437], 9);

  L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
    attribution:
      '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors',
  }).addTo(mapid);

  L.marker([53.43089080173724, -2.9608757752981525], { icon: RedMarkerIcon })
    .addTo(mapid)
    .bindPopup("Voici le stade de Anfield, du club Liverpool");
  L.marker([53.4643424909605, -2.2917587888524773], { icon: GreenMarkerIcon })
    .addTo(mapid)
    .bindPopup("Voici le stade Old Trafford, du club Manchester United");
}
```

Voici encore un bout du fichier indexCtrl :

```
$().ready(function () {
  ctrl = new Ctrl();
  ctrl.matchs();
  ctrl.championnats();
});

class Ctrl {
  constructor() {
    this.httpServ = new HttpServ();
    this.httpServ.centraliserErreurHttp(this.afficherErreurHttp);
  }

  afficherErreurHttp(msg) {
    alert(msg);
  }

  matchs() {
    console.log("TEST");
    var elementSelectionneLeague = document.getElementById("idLeague");
    var competitionId = null;
    for (var i = 0; i < elementSelectionneLeague.options.length; i++) {
      if (elementSelectionneLeague.options[i].selected) {
        competitionId = elementSelectionneLeague.options[i].id;
        break;
      }
    }

    var elementSelectionneSaison = document.getElementById("idSeason");
    var optionSelectionneSaison = null;

    for (var i = 0; i < elementSelectionneSaison.options.length; i++) {
      if (elementSelectionneSaison.options[i].selected) {
        optionSelectionneSaison = elementSelectionneSaison.options[i].id;
        break;
      }
    }
    console.log("j'suis dans Match()");
    this.httpServ.getMatchs(
      competitionId,
      optionSelectionneSaison,
      this.populateTable
    );
  }
```

4.4.2 Problèmes rencontrés

Pendant mon projet j'ai rencontré quelques problèmes, mais jamais des gros problèmes qui ont causé un changement radical dans mon avancement de projet. Le plus gros problème rencontré était les remédiations que j'avais en deuxième année, ce qui m'a causé un retard pour plusieurs modules afin de me focaliser sur les remédiations.

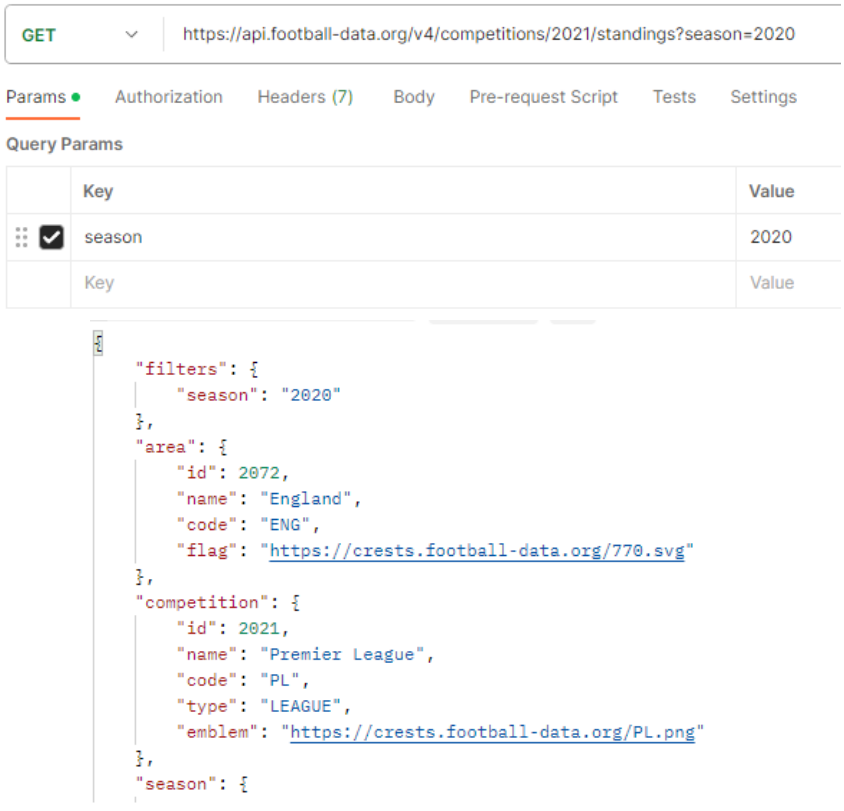
Sinon, cela n'était pas vraiment un problème, mais l'API que j'utilise n'accepte uniquement 10 requêtes par minutes, ce qui est parfois un peu pénible lorsque nous voulons tester si nos requêtes fonctionnent.

4.5 Tests

4.5.1 Test des Webservices

Afin de tester les Webservices il faudrait idéalement le faire avant de faire les maquettes, car si les Webservices n'avaient pas fonctionné, nous aurions dû changer tous les Webservices.

Heureusement mes Webservices étaient fonctionnels et voici donc le résultat en testant les API's avec Postman :



4.5.2 Test du fonctionnement

Voici les tests de fonctionnement de mon Projet.

Test	Résultat voulu	Résultat reçu
Affichage des pages	Ok	Ok
Affichage des astéroïdes	Ok	Ok
Affichage des images de Mars	Ok	Ok

4.6 Hébergement et fonctionnement

Mon Projet est hébergé sur le site Web affiché en dessus ainsi que sur mon Github indiqué plus haut :

https://francan.emf-informatique.ch/Module_307/exercices/Projet/index.html

Les API's que j'ai utilisé sont :

- api.football-data.org
- openStreetMap api

5 Conclusion

Ce module était un module très fascinant, complet et intéressant. J'ai énormément apprécié travailler autant sur les exercices que sur le projet. Cependant, nous avons commencé le projet très tard durant le bloc et manque de chance j'étais malade la semaine du lancement de notre projet, ce qui a fait que j'ai pris un retard considérable en plus des remédiations sur lesquels j'ai dû donner plus de tête que prévu.

Mais finalement je suis très content du résultat et j'ai su rattraper mon retard sans trop de stress.

Pour conclure c'est un module très intéressant et très important pour la suite de notre apprentissage et j'ai beaucoup pu apprendre grâce à cela.