

307 - Réaliser des pages web interactives

Rapport personnel

Date de création 15.05.2023

Version 1 du 15.05.2023

Yerly Tom

EMF – Fribourg / Freiburg

Ecole des Métiers / Berufsfachschule

Technique / Technik

Module du 15.05.2023

Au 13.06.2023

Table des matières

1. Introduction	7
2. Exercices	7
2.1 Exercice 1	7
2.1.1 Objectif de l'exercice.....	7
2.1.2 Explication / Descriptions	7
2.1.3 Extrait de code	7
2.1.4 Capture	8
2.2 Exercice 2	8
2.2.1 Objectif de l'exercice.....	8
2.2.2 Explication / Descriptions	8
2.2.3 Extrait de code	9
2.2.4 Capture	9
2.3 Exercice 3	10
2.3.1 Objectif de l'exercice.....	10
2.3.2 Explication / Descriptions	10
2.3.3 Extrait de code	10
2.3.4 Capture	11
2.4 Exercice 4	12
2.4.1 Objectif de l'exercice.....	12
2.4.2 Explication / Descriptions	12
2.4.3 Extrait de code	12
2.4.4 Capture	12
2.5 Exercice 5	12
2.5.1 Objectif de l'exercice.....	12
2.5.2 Explication / Descriptions	13
2.5.3 Extrait de code	13
2.5.4 Capture	14
2.6 Exercice 6	15
2.6.1 Objectif de l'exercice.....	15
2.6.2 Explication / Descriptions	15
2.6.3 Extrait de code	16
2.6.4 Capture	17
2.7 Exercice 7	17
2.7.1 Objectif de l'exercice.....	17
2.7.2 Explication / Descriptions	17

2.7.3	Extrait de code	18
2.7.4	Capture	18
2.8	Exercice 8.....	19
2.8.1	Objectif de l'exercice.....	19
2.8.2	Explication / Descriptions	19
2.8.3	Extrait de code.....	19
2.8.4	Capture	20
2.9	Exercice 9.....	20
2.9.1	Objectif de l'exercice.....	20
2.9.2	Explication / Descriptions	20
2.9.3	Extrait de code.....	20
2.9.4	Capture	21
2.10	Exercice 10.....	21
2.10.1	Objectif de l'exercice.....	21
2.10.2	Explication / Descriptions	21
2.10.3	Extrait de code.....	21
2.10.4	Capture	21
2.11	Exercice 11.....	22
2.11.1	Objectif de l'exercice.....	22
2.11.2	Explication / Descriptions	22
2.11.3	Extrait de code.....	22
2.11.4	Capture	23
2.12	Exercice 12.....	23
2.12.1	Objectif de l'exercice.....	23
2.12.2	Explication / Descriptions	23
2.12.3	Extrait de code.....	23
2.12.4	Capture	24
2.13	Exercice 13.....	24
2.13.1	Objectif de l'exercice.....	24
2.13.2	Explication / Descriptions	24
2.13.3	Extrait de code.....	24
2.13.4	Capture	25
2.14	Exercice 14.....	25
2.14.1	Objectif de l'exercice.....	25
2.14.2	Explication / Descriptions	26
2.14.3	Extrait de code.....	27
2.14.4	Capture	27
2.15	Exercice 15.....	27
2.15.1	Objectif de l'exercice.....	27

2.15.2	Explication / Descriptions	27
2.15.3	Extrait de code	27
2.15.4	Capture	27
2.16	Exercice 16.....	28
2.16.1	Objectif de l'exercice.....	28
2.16.2	Explication / Descriptions	28
2.16.3	Extrait de code.....	28
2.16.4	Capture	29
2.17	Exercice 17.....	29
2.17.1	Objectif de l'exercice.....	29
2.17.2	Explication / Descriptions	29
2.17.3	Requête GET	29
2.17.4	Requête POST	29
2.17.5	Requête PUT	29
2.17.6	Requête DELETE	30
2.18	Exercice 18.....	30
2.18.1	Objectif de l'exercice.....	30
2.18.2	Explication / Descriptions	30
2.18.3	Extrait de code.....	30
2.18.4	Capture	30
2.19	Exercice 20.....	31
2.19.1	Objectif de l'exercice.....	31
2.19.2	Explication / Descriptions	31
2.19.3	Extrait de code.....	32
2.19.4	Capture	32
3.	Introduction du projet	32
4.	Analyse	33
4.1	Use case.....	33
4.2	Maquette.....	33
5.	Conception	36
5.1	Diagramme de navigation	36
6.	Implémentation.....	36
6.1	Extraits de code	36
6.2	Problèmes rencontrés	37
7.	Tests.....	37
7.1	Test des Webservices	37
7.2	Test du fonctionnement	38
8.	Hébergement et fonctionnement.....	38

9. Conclusion.....	38
---------------------------	-----------

1. Introduction

Dans ce module, nous allons faire de la programmation web, notamment du javascript pour faire du frontend. Ce module est particulièrement important nous allons utiliser des APIs pour la première fois. C'est pour cela qu'il faut bien apprendre les bases.

2. Exercices

Voici les exercices du module :

2.1 Exercice 1

2.1.1 Objectif de l'exercice

Comprendre les bases du Javascript et comprendre le fonctionnement des écouteurs.

2.1.2 Explication / Descriptions

Dans cet exercice, nous devons adapter un code HTML pour que l'on fasse un écouteur dans le HTML et non via Javascript.

Il existe deux moyens de faire des écouteurs. Premièrement, nous pouvons faire un écouteur dans je fichiers javascript qui va exécuter une fonction en récupérant l'ID d'un objet. Ou alors on peut directement via le HTML faire un écouteur en utilisant onclick qui appellera une fonction.

Pour charger du code javascript après qu'une page web soit chargée, il faut utiliser l'évènement onload sur le body de notre document.

Le DOM est une norme W3C qui est utilisée pour définir les standards des éléments, des propriétés, des méthodes et des événements. Il est séparé en trois partie : Core DOM, XML DOM, et HTML DOM.

Le code javascript est chargé via le onload dans le body. Il est nécessaire qu'il soit dans le body car cela permet que tous les éléments soient chargés avant que le code javascript associé soit exécuté. Cependant, il est tout de même nécessaire de faire une référence au fichier dans le head avec la balise « script ».

2.1.3 Extrait de code

Voici un exemple d'écouteur via JS. Pour ce faire, nous l'avons créé dans la fonction initCtrl qui est utilisé lorsque la page a fini d'être chargée grâce au onload :

```
function initCtrl() {  
    // Ecouteur du bouton "Testez-moi..."  
    document.getElementById("testez").addEventListener("click", testez);  
}  
  
function testez() {  
    document.getElementById("info").innerHTML =  
        "C'est <b>James Bond</b> qui a pressé le bouton !";  
}
```

Voici le second exemple avec l'écouteur sur le bouton :

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

Et la fonction « testez » est la même qu'avant.

Voici un exemple d'utilisation du onload :

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

Voici un exemple de balise script :

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

2.1.4 Capture

Voici ce qui arrive lorsque l'on appuie sur le bouton :

Cliquez sur le bouton qui contient votre prénom (html) !

Teste-moi, James

C'est **James Bond** qui a pressé le bouton !

2.2 Exercice 2

2.2.1 Objectif de l'exercice

Voir les bases du javascript en utilisant des variables et des tests.

2.2.2 Explication / Descriptions

Dans cet exercice, nous devons pouvoir récupérer les informations de login et les vérifier en affichant un pop-up qui disait validation réussie ou login ou mot de passe faux. Le code HTML nous était déjà donné, nous devons juste faire le CSS et le javascript.

Les fonctions alert(), confirm() et prompt() sont des fonctions qui permettent de faire apparaître des fenêtres d'information ou de confirmations.

L'attribut « placeholder » permet de mettre un texte de fond, c'est pratique pour donner un message visible seulement quand l'utilisateur n'a pas rempli l'objet. L'attribut « autofocus » permet de dire que l'élément qui l'utilise devrait recevoir le focus au début du chargement de la page.

Dans un input, on peut faire plusieurs choses comme des boutons, des labels, des titres seulement un input ne peut contenir que du texte tandis qu'avec une balise button, on a des possibilités limitées mais on peut faire des requêtes vers des formulaires.

Pour récupérer un élément en fonction de son id, il faut utiliser la fonction getElementById. Puis si l'on veut récupérer le texte, il faut utiliser la fonction value sur l'élément. Un exemple sera donné plus tard dans le point 2.2.3

Pour écrire sur la console du navigateur, il suffit d'ajouter à un moment par exemple si l'on veut vérifier la valeur d'une variable console.log(nomDeLaVariable). En inspectant on peut voir dans la console. De plus si l'on a une erreur dans notre code, la console web va debugger et nous dira sur quelle ligne notre code est faux.

Pour définir une fonction, il suffit de dire fonction nomDeLaFonction() et les dans les accolades on peut mettre notre code. Pour une variable, il suffit de dire let nomDeLaVariable = valeur. Comme nous utilisons let pour toutes les variables, on peut dire qu'elles ne sont pas typées.

On peut charger un javascript lorsque l'on l'appelle avec un écouteur. Que se soit un onload ou un onclick. Le code donné pour faire le html ne contenait pas de javascript.

Pour convertir un texte en minuscule, il faut utiliser tolowercase(). Un exemple sera donné dans le point suivant.

2.2.3 Extrait de code

Voici un exemple ou nous allons récupérer la valeur d'un élément avec un id spécifique :

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

Pour créer une fonction voici un exemple :

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

Pour créer une variable voici un exemple :

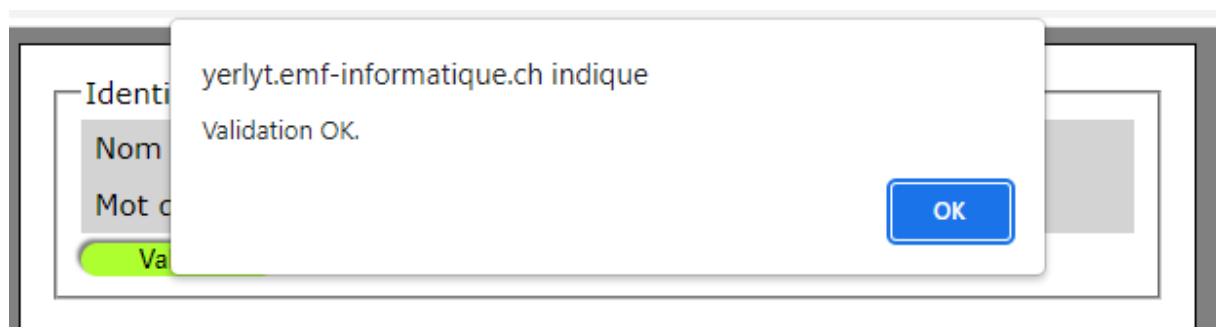
```
let i = 0 ;
```

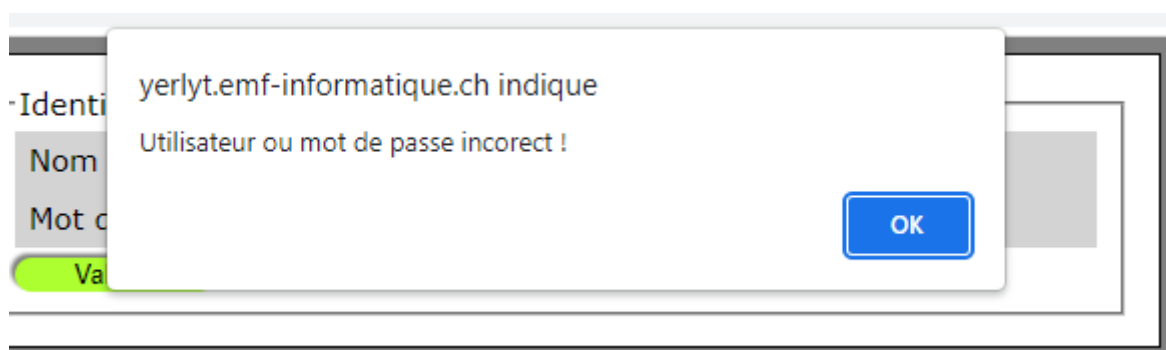
Voici un extrait du code javascript que nous devons faire :

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

2.2.4 Capture

Voici un screenshot de l'application lorsque l'on réussi et lorsque l'on fait faux :





2.3 Exercice 3

2.3.1 Objectif de l'exercice

Comprendre les bases du PHP avec le GET et le POST.

2.3.2 Explication / Descriptions

Dans cet exercice, nous devons faire un programme qui vérifiait un login avec un mot de passe. Il utilisait le PHP pour pouvoir le contrôler. La spécificité d'un élément input c'est que pour le centrer, il faut utiliser « text-align ».

Il existe plusieurs types de boutons avec HTML voici un tableau résumé :

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.

Un code PHP doit commencer par un « <? PHP » et doit se finir par un « ?> ». On peut récupérer les informations dans le flux GET ou POST via l'attribut Name. Pour ce faire, on peut mettre un attribut Name à un input et en utilisant la fonction \$_POST ou \$_GET, on va pouvoir récupérer les informations de cet élément. Un Exemple sera donné dans le point suivant.

Pour concaténer deux chaînes de caractères avec PHP, il faut utiliser le point et pour la transformer en minuscule, on utilise la fonction : « strtolower ». Un Exemple sera donné dans le point suivant.

La commande « Echo » du PHP permet de retourner la valeur de quelque chose au client.

On peut retrouver dans notre code PHP du code Javascript. Car il est utilisé pour faire un popup et spécifier si les identifiants sont bons.

2.3.3 Extrait de code

Voici le code PHP pour le bouton qui utilisait le POST. On peut voir que l'on va récupérer les valeurs avec la fonction \$_POST et que l'on la transforme en minuscule. Les chaînes de caractères sont concaténées avec des points.

```
<?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 == "enf123")) {
        echo "<script>alert('Validation OK');</script>";
    } else {
        echo "<script>alert('Utilisateur ou mot de passe incorrect
!!!');</script>";
    }
}
?>
```

Le script PHP avec un GET est le même, il n'y a que les fonctions POST qui sont changées par des GET. On privilégie le POST car il n'affiche pas les informations dans l'URL et n'est pas limité par un certain nombre de caractères.

2.3.4 Capture

En regardant l'onglet Network on peut voir notre PHP ainsi que les informations de notre script :

The screenshot shows a web browser with the address bar displaying `yerlyt.emf-informatique.ch/307/Exercices/Exercice3/login.php`. Below the address bar, the page content shows the output of the PHP script: `username: admin` and `password: enf123`.

Below the browser window, the Network tab of the developer tools is open. It shows a list of network requests, with the first request, `login.php`, selected. The 'Payload' tab is active, displaying the 'Form Data' of the request:

Form Data
username: admin
password: enf123

2.4 Exercice 4

2.4.1 Objectif de l'exercice

Comprendre l'utilité de JSFiddle et comprendre comment utiliser cet outil.

2.4.2 Explication / Descriptions

Dans cet exercice, nous allons utiliser JSFiddle qui est une plateforme qui permet de modifier et d'exécuter des extraits de code. C'est un outil qui est particulièrement utile car il nous permet de tester de simples choses sans devoir faire tout le programme, il nous permet aussi de faire des tests rapides. Dans l'exercice que nous devons faire, nous devons créer un bouton qui lorsque l'on appuie dessus un popup nous avertit et nous dit que le bouton a été appuyé. Avec JSFiddle, nous n'avons besoin que de faire le bouton en html, le script avec Javascript et le css du bouton si nécessaire.

2.4.3 Extrait de code

Voici le code HTML :

```
<input type="button" onclick="Appuiez()"/>
```

Voici le code CSS :

```
input{  
  background-color: blue;  
  width: 100px;  
  height: 100px;  
  border: none;  
}
```

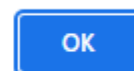
Voici le code Javascript :

```
function Appuiez(){  
  alert("Quelqu'un a appuyé sur le bouton")  
}
```

2.4.4 Capture

Voici l'action qui se passe lorsque l'on appuie sur le bouton :

Une page intégrée à l'adresse fiddle.jshell.net indique
Quelqu'un a appuyé sur le bouton



2.5 Exercice 5

2.5.1 Objectif de l'exercice

Dans cet exercice, nous allons voir l'utilité et les fonctionnalités des variables en Javascript.

2.5.2 Explication / Descriptions

Dans cet exercice, nous devons suivre des étapes qui nous faisaient faire des exercices sur les variables. Par exemple concaténer deux variables, additionner deux variables faire des tests avec les AND et OR, faire des calculs de date ainsi que vérifier le type d'une variable.

Pour déclarer une variable, on utilise le mot-clé « let » en mettant « let » un espace et le nom de la variable, on peut créer une variable. Puis avec un égal on peut lui assigner une valeur.

Pour faire une constante c'est la même chose qu'une variable simple seulement que l'on doit remplacer le « let » par un « const ». Il n'y a pas de convention de nommage spéciale mais c'est une bonne pratique de le mettre en majuscule. Attention c'est une constante donc la valeur ne peut pas être changée.

Le mot-clé « var » peut aussi être utilisé pour créer une variable mais les variables peuvent avoir le même nom. Donc ce n'est pas le plus pratique de créer une variable avec var il faut mieux créer une variable avec un nom unique avec le let.

On peut aussi créer une variable sans mot-clé mais ce sera une variable globale. Et ce sera utilisé plus tard dans le module.

Il existe 4 types de variables de bases. String pour les chaînes de caractères, number pour les nombres (décimaux ou pas), boolean pour les vrai ou faux, object pour les autres types.

Pour connaître le type d'une variable on peut utiliser la fonction typeof. Un exemple est donné dans le point suivant.

2.5.3 Extrait de code

Voici le code de l'exercice :

```
//nettoyage de la console
console.clear();
//création de variable plus affichage
let a;
console.log(a);
a = 15;
console.log("Ma variable a = " + a);
let b = 9;
console.log("Ma variable b = " + b);
//calcul avec les variables
console.log(a + " + " + b + ` = ${a + b}`);
console.log(a + " - " + b + ` = ${a - b}`);
console.log(a + " * " + b + ` = ${a * b}`);
console.log(a + " / " + b + ` = ${a / b}`);
//Concatenation de strings
a = "bonjour";
b = " les amis";
console.log(a + b);
console.log(`${a} ${b}`);
//Tests de variables avec AND et OR
a = true;
b = false;
```

```
console.log(a + " AND " + b + " = " + (a && b));
console.log(a + " OR " + b + " = " + (a || b));
//Affichage et calcul de dates
a = new Date();
b = new Date(a);
b.setDate(a.getDate() - 61);
console.log(a.toLocaleString());
console.log(b.toLocaleString());
console.log(a.toLocaleDateString());
console.log(b.toLocaleDateString());
console.log(a.toLocaleTimeString());
console.log(b.toLocaleTimeString());
//Types de variables
a = Math.PI;
b = "Bonjour";
let c = true;
let d = new Date();
let e;
console.log(typeof a);
console.log(typeof b);
console.log(typeof c);
console.log(typeof d);
console.log(typeof e);
```

2.5.4 Capture

Voici une capture de l'exercice :

```
undefined
"Ma variable a = 15"
"Ma variable b = 9"
"15 + 9 = 24"
"15 - 9 = 6"
"15 * 9 = 135"
"15 / 9 = 1.6666666666666667"
"bonjour les amis"
"bonjour les amis"
"true AND false = false"
"true OR false = true"
"16/05/2023 09:17:09"
```

Et voici les résultats :

```
undefined
```

```
"Ma variable a = 15"
"Ma variable b = 9"
"15 + 9 = 24"
"15 - 9 = 6"
"15 * 9 = 135"
"15 / 9 = 1.6666666666666667"
"bonjour les amis"
"bonjour les amis"
"true AND false = false"
"true OR false = true"
"16/05/2023 09:27:19"
"16/03/2023 09:27:19"
"16/05/2023"
"16/03/2023"
"09:27:19"
"09:27:19"
"number"
"string"
"boolean"
"object"
"undefined"
```

2.6 Exercice 6

2.6.1 Objectif de l'exercice

Comprendre le fonctionnement et l'utilisation des switches ainsi que l'utilité des tableaux.

2.6.2 Explication / Descriptions

Dans cet exercice, nous devons pouvoir définir quel jour de la semaine nous étions lorsque l'on appuyait sur un bouton. Nous devons mettre en place un moyen pour que cela fonctionne une fois en utilisant un tableau et une fois un switch.

Pour récupérer le numéro du jour, on peut utiliser la fonction `getDate()`. Elle nous sera pratique lors de cet exercice. Ensuite pour récupérer du texte dans une balise HTML, on utilise `document.getElementById("ID").innerHTML` et la valeur de ce que l'on veut retourner. Pour retourner une balise, il suffit de rajouter les balises au début et à la fin de notre valeur de retour.

Pour créer un tableau, il suffit de créer une variable avec comme valeur des crochets. Cela nous créera un tableau vide pour le remplir on peut soit utiliser des fonctions ou alors directement mettre les valeurs dans les crochets avec des virgules comme séparateurs.

Voici un tableau récapitulatif des fonctions et Tips

<code>console.log(tab[0])</code>	Affiche la cellule 0 de notre tableau tab.
<code>tab.length</code>	Donne la taille du tableau tab
<code>tab.push(1)</code>	Ajoute 1 à la fin du tableau
<code>tab.unshift(1)</code>	Ajoute 1 au début du tableau
<code>tab.pop()</code>	Efface le premier élément

tab.shift()	Efface le dernier élément
-------------	---------------------------

Pour afficher l'ensemble des valeurs d'un tableau le moyen le plus facile est d'utiliser une boucle foreach. Un exemple sera donné dans le point suivant.

2.6.3 Extrait de code

Voici le code javascript de l'application :

```
function afficherJourSemaine() {
  //création des variables
  var a = new Date();
  a = a.getDay();
  let b;
  //Tests via le switch
  switch (a) {
    case 1:
      b = "Lundi";
      break;
    case 2:
      b = "Mardi";
      break;
    case 3:
      b = "Mercredi";
      break;
    case 4:
      b = "Jeudi";
      break;
    case 5:
      b = "Vendredi";
      break;
    case 6:
      b = "Samedi";
      break;
    case 0:
      b = "Dimanche";
      break;
  }
  //retour du jour
  document.getElementById("info").innerHTML = `On est ${b}`;
}

function afficherJourSemaineTableau() {
  //création des variables
  var a = new Date();
  a = a.getDay();
  var b;
  const TAB = ["Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"];
}
```



```
//Recuperation du jour
b = TAB[a]

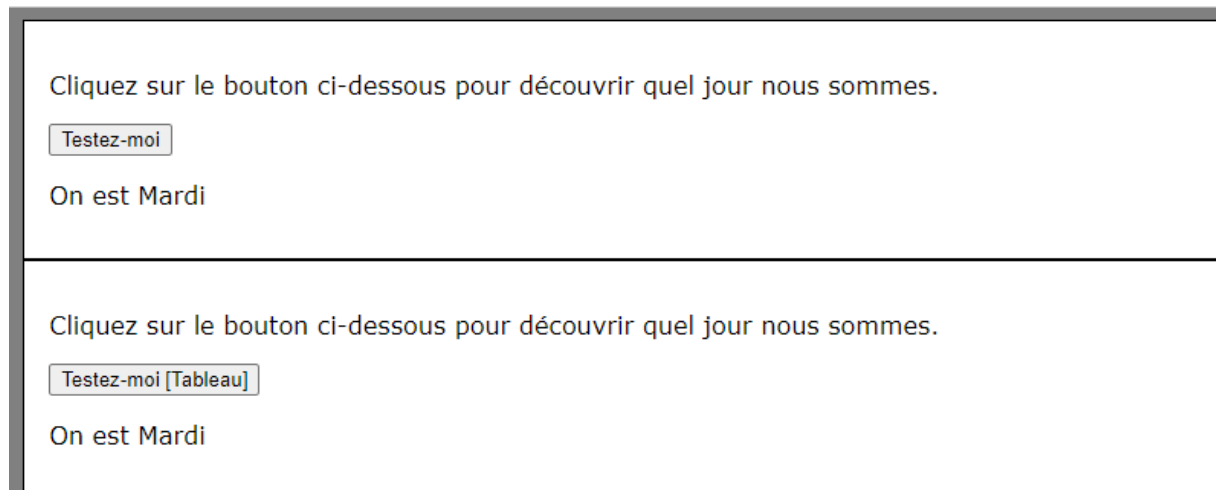
//retour du jour
document.getElementById("infoTab").innerHTML = `On est ${b} [Tableau]`;
}
```

Et voici un exemple de boucle foreach qui affiche toutes les cellules d'un tableau.:

```
tab.forEach(function(ce1) {
  console.log(ce1);
});
```

2.6.4 Capture

Voici le résultat de l'exercice :



2.7 Exercice 7

2.7.1 Objectif de l'exercice

Dans cet exercice, nous avons vu les bases des boucles javascript.

2.7.2 Explication / Descriptions

Dans cet exercice, nous devons afficher un compteur qui s'incrémente jusqu'à 10 tout en l'affichant. Pour ce faire, nous avons dû utiliser des boucles et pour mieux se familiariser avec elles, nous avons utilisé une fois la boucle for, while, et do-while.

La boucle for est utilisable lorsque l'on sait combien de fois on veut faire celle-ci. Elle est très utilisée pour des compteurs. Pour en créer une, il suffit de mettre « for (let i = 0 ; condition ; i++){...} ».

La boucle while est utilisé lorsque l'on ne sait pas combien de fois on veut faire la boucle, on l'écrit comme ceci : « while (condition){...} ».

Et la boucle do-while est utilisé lorsque l'on ne sait pas combien de fois on veut faire la boucle mais que l'on doit y passer au moins une fois. Pour de faire, on l'écrit : « do{...} while(condition) ; »

2.7.3 Extrait de code

Voici le code javascript de l'exercice :

```
function initCtrl() {
    //création des écouteurs pour chaque bouton
    document.getElementById("for").addEventListener("click", testerFor);
    document.getElementById("while").addEventListener("click", testerWhile);
    document.getElementById("do").addEventListener("click", testerDoWhile);
}
function testerFor() {
    //création de la variable retour
    let retour = "for (let i = 0; i < 10; i++) {...}<br>";
    //boucle for
    for (let i = 0; i < 10; i++) {
        retour += "i = " + i + "<br>";
    }
    document.getElementById("info").innerHTML = retour + "--> A utiliser si l'on sait que l'on veut itérer x fois (x connu avant de commencer la boucle)";
}
function testerWhile() {
    //création du compteur et de la variable retour
    let i = 0;
    let retour = "while (i < 10) {...}<br>";
    //boucle while
    while (i < 10) {
        retour += "i = " + i + "<br>";
        i++;
    }
    document.getElementById("info").innerHTML = retour + "--> A utiliser si on ne sait pas le nombre d'itérations au démarrage de la boucle";
}
function testerDoWhile() {
    //création du compteur et de la variable retour
    let i = 0;
    let retour = "{...} while (i < 10)<br>";
    //boucle do-while
    do {
        retour += "i = " + i + "<br>";
        i++;
    } while (i < 10);
    document.getElementById("info").innerHTML = retour + "--> A utiliser si on ne sait pas le nombre d'itérations au démarrage de la boucle mais avec un passage obligatoire";
}
```

2.7.4 Capture

Voici le résultat de l'exercice :

Cliquez sur les boutons ci-dessous pour tester les différentes sortes de boucles en JavaScript.

```
while (i < 10) {...}
```

```
i = 0
```

```
i = 1
```

```
i = 2
```

```
i = 3
```

```
i = 4
```

```
i = 5
```

```
i = 6
```

```
i = 7
```

```
i = 8
```

```
i = 9
```

--> A utiliser si on ne sait pas le nombre d'itérations au démarrage de la boucle

2.8 Exercice 8

2.8.1 Objectif de l'exercice

Dans cet exercice, nous devons comprendre l'utilité du JSON avec javascript.

2.8.2 Explication / Descriptions

Dans cet exercice, nous avons pour but d'afficher des personnes avec leur nom, prénom et âge. Pour ce faire, nous avons utilisé d'un tableau JSON de personnes. Et grâce à une boucle récupérer les valeurs.

Pour faire une boucle sur un tableau json on peut utiliser le for in et pour récupérer les valeurs de ce tableau, on peut utiliser le champ field (f). Un exemple est donné ci-dessous.

2.8.3 Extrait de code

Voici le code javascript de l'exercice :

```
function parcourirUnTableauJSON(){
  let retour = "";
  const json = {
    personnes: [
      {prenom: "John", nom: "Doe", age: 44},
      {prenom: "Anna", nom: "Smith", age: 32},
      {prenom: "Peter", nom: "Jones", age: 29}
    ]
  };

  for(let i = 0; i < json.personnes.length; i++){
    let personne = json.personnes[i];
    for (let f in personne){
      retour += personne[f] + " ";
    }
    retour += "<br>"
  }
}
```

```
}  
  
document.getElementById("info").innerHTML = retour;  
  
}
```

2.8.4 Capture

Voici ce qui arrive lorsque <l'on appuie sur le bouton :

Cliquez sur le bouton pour parcourir un tableau JSON

Parcourir un tableau JSON

John Doe 44
Anna Smith 32
Peter Jones 29

2.9 Exercice 9

2.9.1 Objectif de l'exercice

Comprendre créer des objets en javascript.

2.9.2 Explication / Descriptions

Pour créer un objet, nous utilisons la méthode de la classe. Pour ce faire, nous devons déclarer une classe. Puis dans le constructeur demander les attributs nécessaires pour cette classe. Puis on peut définir les attributs comme en java avec le `this.nom = valeur`.

Puis on peut créer la méthode `toString()`. Cette méthode va être utilisé lorsque l'on va mettre un objet que l'on a créé sous forme de String. Elle a pour but de transformer notre objet en String en prenant les attributs et les mettant sous une forme précise.

2.9.3 Extrait de code

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 + ")";  
  }  
}
```

2.9.4 Capture

Voici un exemple d'utilisation :

Cliquez sur le bouton pour créer découvrir la création d'objets

simplement JSON

avec Object

avec objet fonction

avec fonction prototype

avec classe

Julien Tartampion (18)

Julia Tartampion (22)

2.10 Exercice 10

2.10.1 Objectif de l'exercice

Programmer orienter objets.

2.10.2 Explication / Descriptions

Dans cet exercice, nous avons dû faire un programme qui permettait de créer, d'afficher et de supprimer des personnes. Pour ce faire, nous avons créé une classe personne qui contenait un nom, un prénom et un âge ainsi que la fonction toString qui permettait d'afficher une personne sous forme de String.

Ensuite, dans le worker, nous avons créé notre tableau de personne puis des méthodes permettant d'ajouter et de supprimer des personnes. Grâce aux méthodes que nous avons vu précédemment sur les tableaux.

Puis pour finir dans notre indexCtrl, nous avons créé les fonctions de notre contrôleur. Ces fonctions ne sont pas très importantes car elles sont basiques, elles utilisent des outils simples comme des boucles et des fonctions comme getElementById etc... Le plus important, est la fonction document.onreadystatechange. Cette méthode va faire que lorsque notre DOM a fini d'être chargé, va lancer une fonction. Il s'agit d'un onload amélioré car il est plus rapide et n'attend pas que toute la page ait fini de charger.

Nous avons aussi vu que pour faire des fonctions privées, la convention de nommage nécessite un tiret du bas avant le nom.

2.10.3 Extrait de code

Voici notre fonction document.onreadystatechange, elle va utiliser la fonction privée afficherPersonnes :

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

2.10.4 Capture

Voici le résultat de l'exercice :

Veuillez introduire une nouvelle personne ou sélectionner une existante :

Prénom:	<input type="text" value="prénom"/>
Nom:	<input type="text" value="nom"/>
Âge:	<input type="text" value="âge"/>

- [Doe John \(44\)](#)
- [Jones Peter \(29\)](#)
- [Smith Anna \(32\)](#)

2.11 Exercice 11

2.11.1 Objectif de l'exercice

Programmer orienter classe en javascript.

2.11.2 Explication / Descriptions

Dans cet exercice, nous avons vu comment utiliser des classes comme dans nos projets java.

Nous avons dû refaire l'exercice 10 mais avec des classes personnes, worker, controller. Pour commencer, il faut déclarer une classe avec un constructeur, puis tout le reste est la même que s'il n'y en avait pas. Mais il y a quelques changements lors de l'utilisation des méthodes et des variables. Pour utiliser une méthode qui vient de la classe, il faut utiliser le mot-clé this. Si l'on veut utiliser une méthode d'une autre classe par exemple la méthode checkpersonne dans notre worker depuis notre contrôleur, il faudra utiliser this.wrk.checkpersonne.

L'utilisation du onreadystatechange change avec des classes.

De plus dans cet exercice, nous avons utilisé 3x= pour pouvoir faire un test. Ce trois égal nous donne une information supplémentaire en testant le type en plus de la valeur des variables.

2.11.3 Extrait de code

Voici un exemple de classe avec une méthode et un constructeur :

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

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

Et voici l'utilisation du `onreadystatechange` lorsque l'on utilise des classes :

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

2.11.4 Capture

Voici le résultat de l'exercice :

Veuillez introduire une nouvelle personne ou sélectionner une existante :

Prénom:	<input type="text" value="John"/>
Nom:	<input type="text" value="Doe"/>
Âge:	<input type="text" value="44"/>

- [Doe John \(44\)](#)
- [Smith Anna \(32\)](#)
- [Jones Peter \(29\)](#)

2.12 Exercice 12

2.12.1 Objectif de l'exercice

Créer et comprendre les fonctions et les fonction IIFE.

2.12.2 Explication / Descriptions

Dans cet exercice, nous avons vu toutes les façons de faire des fonctions. Premièrement la méthode que l'on connaît tous avec le mot-clé `function`. Deuxièmement avec une expression fonction (fonction anonyme). Et pour finir l'arrow function en utilisant la flèche.

Ce sont les trois types de fonctions différentes. De plus on peut créer des fonctions IIFE (immediately-invoked Function expression) qui sont des fonctions utilisées directement après avoir été créées. Des exemples sont donnés dans le point suivant.

2.12.3 Extrait de code

Voici les fonctions créées dans l'exercice :

```
function a(add) {  
  let val = 1;  
  console.log(val + add) ;  
}  
a(3);
```

```
let b = function(add) {  
  let val = 2;  
  console.log(val + add) ;  
} ;  
b(2);  
(function(add) {  
  let val = 3;  
  console.log(val + add) ;  
})(1) ;  
((add) => {  
  let val = 4;  
  console.log(val + add) ;  
})(2) ;
```

2.12.4 Capture

Il n'y a pas de capture possible comme résultat.

2.13 Exercice 13

2.13.1 Objectif de l'exercice

Comprendre les bases des cookies.

2.13.2 Explication / Descriptions

Dans cet exercice, nous avons vu les bases sur les cookies. Les cookies sont des petits fichiers textes qui sont situés sur le poste de l'utilisateur et pouvant stocker des données. Pour créer un cookie et stocker une chaîne de caractères, il faut utiliser `document.cookie` puis mettre un `=` et lui donner un nom puis la valeur un exemple sera donné au point ci-dessous.

On peut récupérer une chaîne de caractères d'un cookie via la méthode `split`. Le cookie contient une valeur que l'on va spliter car on ne souhaite pas récupérer le nom du cookie.

2.13.3 Extrait de code

Pour créer un cookie et lui assigner une valeur on va faire comme ceci :

```
document.cookie = "cookie_exercice_13=" + contenu
```

Puis pour récupérer une chaîne de caractères depuis un cookie, il faut faire ceci :

```
// 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];  
}
```

Et voici un exemple de code pour convertir une chaîne de caractères en objet json et vice-versa :


```
//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],
};
```

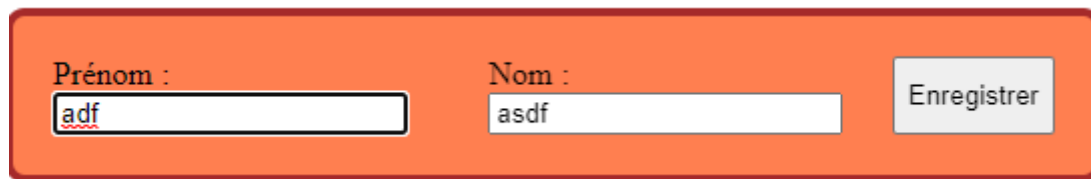
Et dans l'autre sens :

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

Où alors, on peut utiliser les méthodes `json.stringify` ou `json.parse`

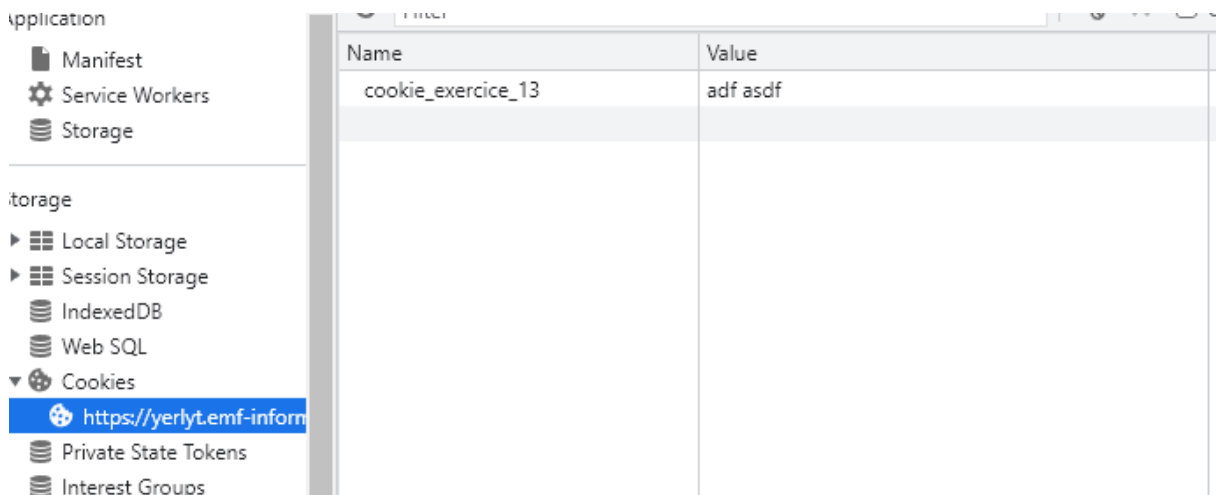
2.13.4 Capture

Voici ce que l'on obtient lorsque l'on recharge la page :



The screenshot shows a web form with an orange background. It contains two input fields: "Prénom :" with the value "adf" and "Nom :" with the value "asdf". To the right of the "Nom" field is a button labeled "Enregistrer".

Et on peut contrôler avec l'inspecteur :



2.14 Exercice 14

2.14.1 Objectif de l'exercice

Comprendre les bases de JQuery.

2.14.2 Explication / Descriptions

JQuery est une librairie javascript open source qui facilite l'interaction entre le javascript et le HTML ainsi que le CSS.

Pour reconnaître du code JQuery, il suffit juste de voir si le code contient un \$ puis des parenthèses.

Le JQuery a pour but d'accéder aux éléments du DOM ainsi que de les modifier. Pour retrouver certains éléments du DOM avec JQuery, il suffit d'utiliser le \$ et les parenthèses et donner une référence que se soit un ID, une classe, ou un type de balise par exemple :

```
$("div")
```

Ici nous récupérerons un set de N nœuds de DIV. Mais avec ".eleve", on récupérerait tous les éléments avec une classe eleve. Et le # pour les ID.

Il existe aussi quelques moyens de faire des enchaînements avec la méthode .parent() pour récupérer l'élément parent, .next() pour l'élément suivant, .prev() pour l'élément précédent, .children() pour les éléments enfants et .siblings() pour les éléments frères.

Voici un exemple :

```
$("button").parent()
```

Pour modifier le CSS d'un élément, il suffit d'utiliser .css(). Voici un exemple où nous modifions la couleur des liens en rouge :

```
$("a").css({  
    color: "red"  
});
```

Pour ajouter un élément avant ou après un autre, il suffit d'utiliser after ou before. Dans l'exemple ci-dessous, nous ajoutons du texte après les balises <a> :

```
$("a") .after("<p>Salut</p>");
```

Pour supprimer un élément, il faut utiliser remove(). Par exemple ici nous supprimons tous les paragraphes :

```
$("p").remove();
```

On peut aussi utiliser les fonctions submit(), click() ou hover() pour gérer les événements. Des exemples sont donnés ici : <https://www.jcsinfo.ch/demo/jquery/>

Pour cacher ou afficher un élément, il suffit d'utiliser la méthode hide() ou show(). Voici un exemple où nous allons gentiment cacher notre élément puis gentiment le réafficher.

```
$("div").hide("slow", function(){  
    $(this).show("slow");  
});
```

Pour lire un fichier avec ajax, il suffit d'utiliser la méthode ajax avec comme paramètre url. Puis on peut suivre les informations données ici sur cette méthode : <http://api.jquery.com/jQuery.ajax/>

Cependant l'utilisation du JSON a été simplifiée si l'on veut lire un fichier JSON, on peut utiliser la méthode getJSON(). Il ne nous reste qu'à donner le chemin d'accès puis la fonction que l'on veut pour récupérer des informations. Voici un exemple sans fonction :

```
$.getJSON("file.json", function( obj ) {...});
```

Pour charger un fichier HTML dans un div existant, il nous suffit d'utiliser la méthode load() et donner le nom du fichier. Voici un exemple :

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

Pour ajouter JQuery dans nos pages, il suffit d'ajouter cette balise dans le head de notre index.html :

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

2.14.3 Extrait de code

Les extraits de codes sont déjà donnés plus tôt pour faciliter la lecture.

2.14.4 Capture

Il n'y a pas de capture à faire pour cet exercice.

2.15 Exercice 15

2.15.1 Objectif de l'exercice

Utiliser JQuery pour la première fois.

2.15.2 Explication / Descriptions

Dans cet exercice, nous avons dû faire une application qui changeait de couleur lorsqu'en fonction d'une liste. Pour ce faire, nous avons utilisé JQuery et javascript. Nous avons pu voir que JQuery est beaucoup plus facile à utiliser que javascript. Pour récupérer une propriété avec JQuery, pour récupérer la propriété css, il suffit d'utiliser .css(laPropriété).

Pour récupérer la valeur d'un champ avec JQuery, nous devons utiliser, .val(). Un exemple sera donné plus tard.

Et pour utiliser une animation avec JQuery, il suffit d'utiliser .NomAnimation(paramètres).

2.15.3 Extrait de code

Voici comment récupérer une propriété et la changer en ajoutant un deuxième paramètre :

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

Voici comment récupérer la valeur d'un champ :

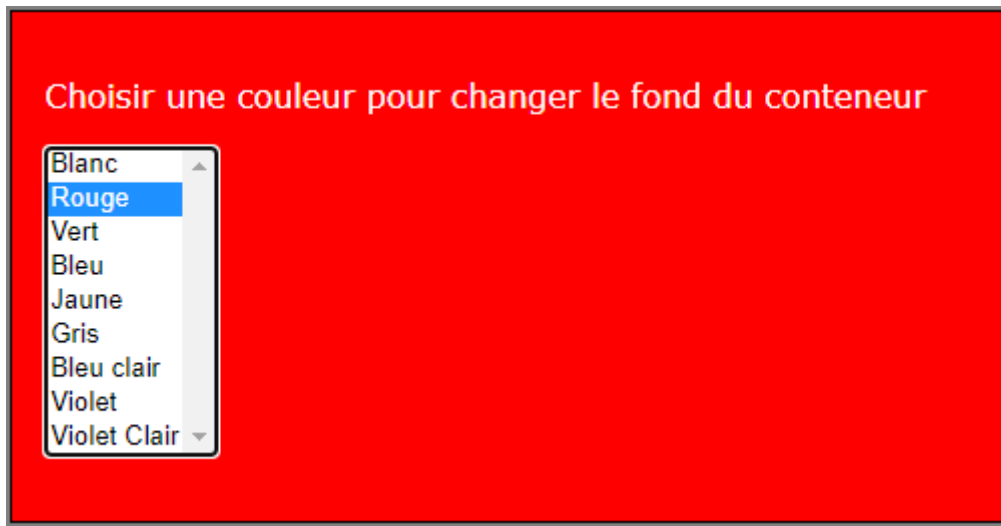
```
$("#couleurs").val();
```

Et voici un exemple d'animation avec JQuery :

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

2.15.4 Capture

Voici le résultat de l'exercice :



2.16 Exercice 16

2.16.1 Objectif de l'exercice

Apprendre à bien utiliser JQuery.

2.16.2 Explication / Descriptions

Dans cet exercice, nous devons être capable de traiter plusieurs textes en leur ajoutant des écouteurs. Pour les faire, nous avons utilisé la fonction `on` de JQuery. Après avoir créé les écouteurs, nous devons faire disparaître et apparaître des textes, quelquefois avec un fondu, parfois leur ajouter une classe... C'était un exercice complet qui traitait les points capitaux du JQuery.

Pour pouvoir sélectionner des éléments du DOM, il suffit d'utiliser les sélecteurs CSS dans `$()`. Pour ajouter ou supprimer une classe à un élément, il suffit d'utiliser la fonction JQuery `addClass` ou `removeClass`. C'est très pratique pour pouvoir changer du CSS à un moment précis.

Pour la suite, nous devons aussi être capable d'ajouter des éléments. Pour ce faire, nous avons utilisé la fonction `append`. La fonction `attr` permet de récupérer la valeur d'un attribut d'un élément c'est pratique pour connaître par exemple une PK.

2.16.3 Extrait de code

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 :

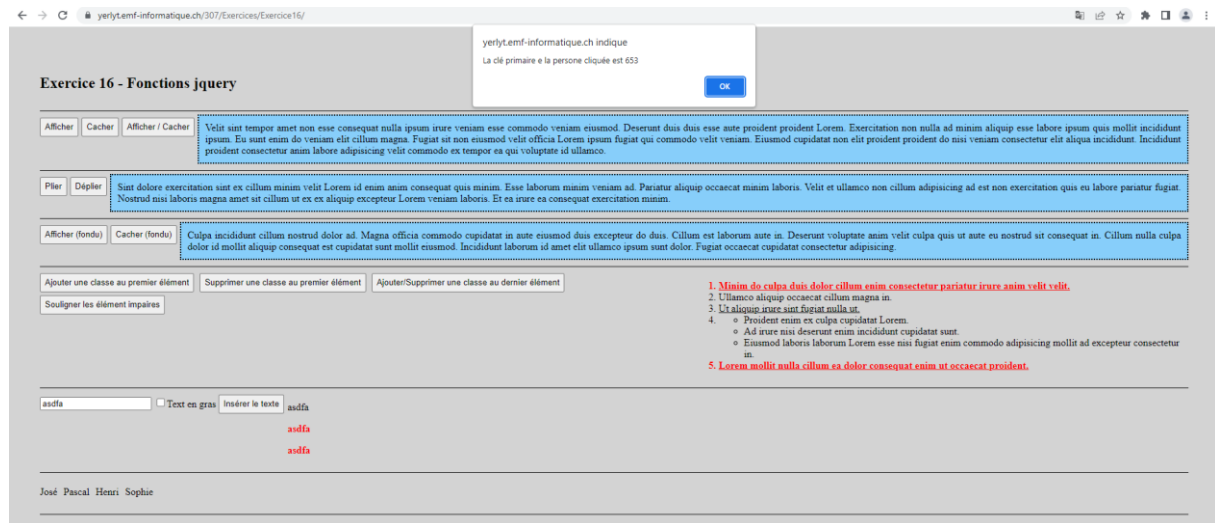
```
$("li").first().addClass("boldBlueText");
```

Voici un exemple de récupération d'attribut :

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

2.16.4 Capture

Voici la capture de l'exercice :



2.17 Exercice 17

2.17.1 Objectif de l'exercice

Découvrir les WebServices.

2.17.2 Explication / Descriptions

Il existe deux moyens d'utiliser des webservices avec SOAP ou REST. Ces deux architectures permettent d'utiliser des API. Dans ce module, nous allons utiliser des APIs pour notre projet c'est pour cela qu'elles sont très importantes à voir. Pour pouvoir utiliser une API, nous devons faire une requête.

2.17.3 Requête GET

Le GET est la requête qui permet de récupérer des données. Cette requête est la plus utilisée de toutes. Pour faire une requête GET, il faut donner plusieurs paramètres comme l'URL de l'API et les paramètres.

2.17.4 Requête POST

La requête POST permet d'insérer des données dans une API. Elle est très utilisée pour stocker des Identifiants. Une requête POST est composée de plusieurs paramètres. Comme l'en-tête qui permet de définir le type de format de nos données. Les autorisations qui permettent de laisser l'utilisateur passer en fonction de sa clé et le body qui contient les données qu'elles soient sous un format texte ou clé-valeur.

2.17.5 Requête PUT

La requête put permet de modifier des données dans une API, elle est moins utilisée que la requête post. Pour pouvoir faire une requête put, il faut donner un filtre pour pouvoir choisir ce que l'on va modifier et donner le contenu que l'on veut ajouter.

2.17.6 Requête DELETE

La requête delete permet de supprimer des données dans une API. Pour pouvoir l'utiliser, il faut faire un filtre sur ce que l'on veut supprimer.

2.18 Exercice 18

2.18.1 Objectif de l'exercice

Mettre en ligne un service PHP et effectuer des appel AJAX en JS.

2.18.2 Explication / Descriptions

Dans cet exercice, nous allons faire un convertisseur de degrés Celsius en fahrenheit. Pour ce faire, nous allons utiliser ajax et faire soit une requête post soit une requête get. Il suffit de faire \$.ajax(url : {objetJson}). Bien sure, il faut changer les paramètres de l'objet en fonction de notre requête.

Avec ajax, on peut spécifier une requête en cas d'erreur ou de succès pour ce faire, il suffit d'ajouter les paramètres succes et error et de spécifier quelles méthodes doivent être utilisée.

On peut tout de même avoir des erreurs alors que notre programme est fonctionnel. Cela est dû au CORS. Le CORS est un mécanisme qui oblige a rajouter des entête http pou pouvoir accéder aux ressources pour pouvoir l'outrepasser, il faut ajouter ceci dans le PHP :

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

2.18.3 Extrait de code

Voici comment faire une requête post avec Ajax et des paramètres spécifiques :

```
celcius2Fahrenheit(degrees, successCallback, errorCallback) {  
    let url = "https://yerlyt.emf-  
informatique.ch/307/Exercices/Exercice18/php/convert-temp_p_XML.PHP";  
    let param = "Temperature=" + degrees + "&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  
    });  
}
```

2.18.4 Capture

Voici le résultat de l'exercice lorsque l'on met des degrés que l'on veut transformer :

Convertisseur de température

°C: °F:

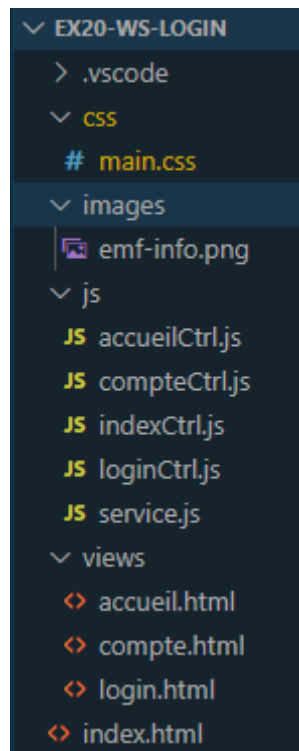
2.19 Exercice 20

2.19.1 Objectif de l'exercice

Expliquer et implémenter une application selon MVC, utiliser des requêtes ajax entre un client et un serveur et charger des vues dynamiquement dans une application.

2.19.2 Explication / Descriptions

Dans cet exercice, nous devons être capable de créer une application de login avec PHP. Pour ce faire, nous avons du modifier les erreur du projet qui nous était donné pour pouvoir le transformer en application fonctionnelle. L'exercice devait être réalisé de cette manière :



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

Une application SPA est une application web qui ne charge qu'un seul document puis le met à jour grâce à des API javascript. Ce qui permet d'augmenter la fluidité du programme.

On peut utiliser la méthode load pour pouvoir charger du code html d'une page dans un élément. Un exemple sera donné au point suivant. La méthode ajaxSetup() définit les valeurs par défaut pour les futures requêtes AJAX.

2.19.3 Extrait de code

Un exemple d'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")
```

2.19.4 Capture

Voici les réponses que l'on reçoit lorsque l'on crée un utilisateur ou lorsque l'on essaye de se connecter avec un autre :

yerlyt.emf-informatique.ch indique

Enregistrement OK: Bonjour tom

OK

Good morning Vietnam

Cette page correspond à l'application principale qu'il faudrait développer.

Pour effectuer un logout, cliquez [ici](#).

3. Introduction du projet

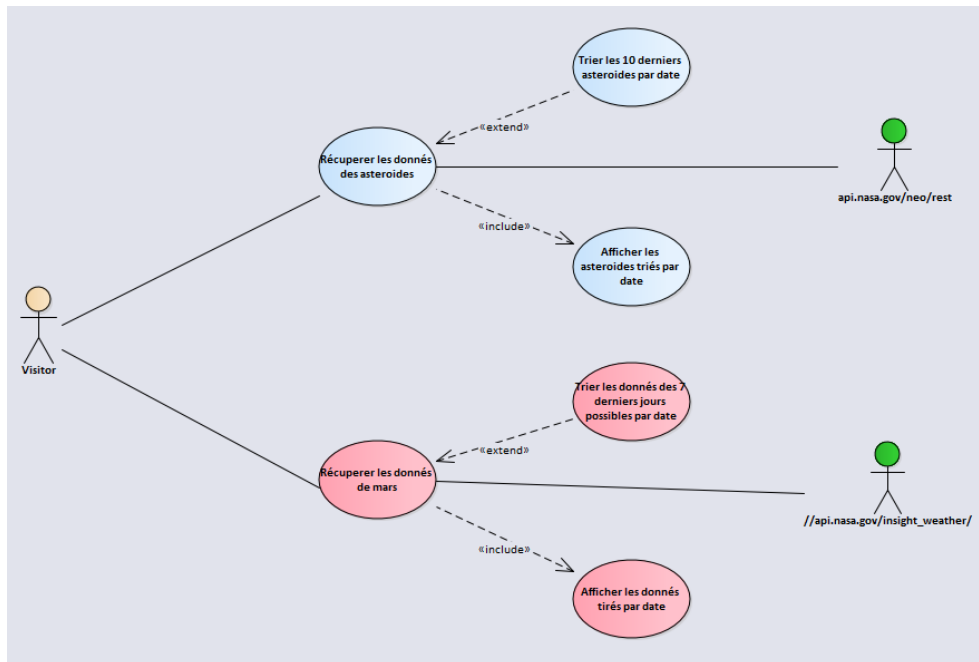
Dans ce projet, j'ai voulu montrer l'espace. J'ai donc décider d'aller voir chez la Nasa s'ils avaient des Api sur l'espace disponibles. Et ils proposent des APIs sur l'espace en donnant

des données en temps réel. Durant ce projet j'ai rencontré une ou deux difficultés, attention il se peut que parfois la page sur les astéroïdes mette du temps à charger.

4. Analyse

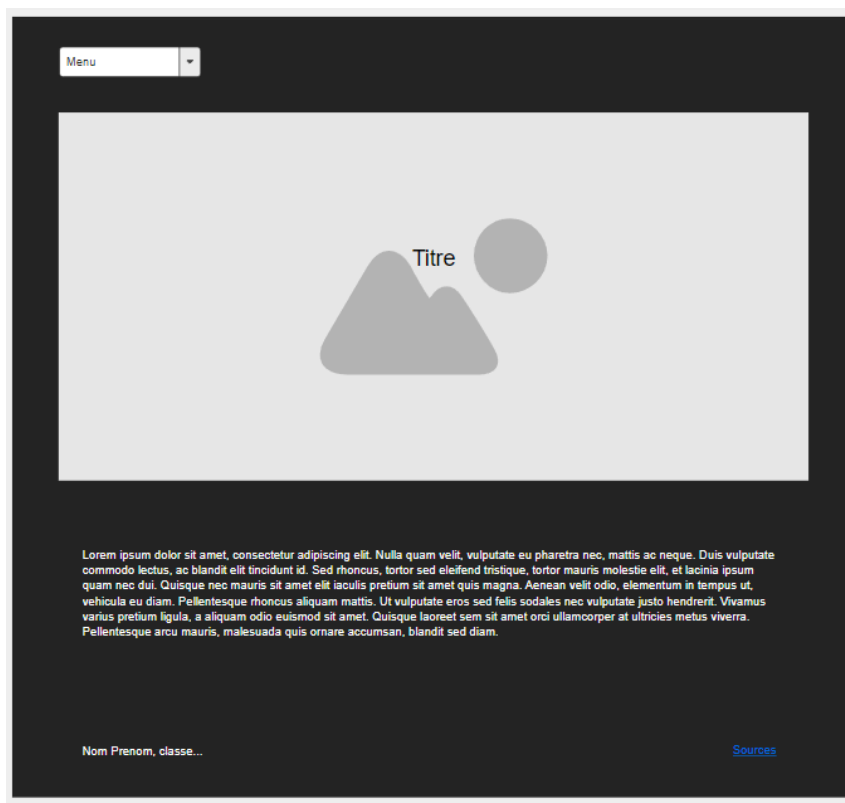
4.1 Use case

Pour commencer l'analyse de mon projet, j'ai commencé par faire les Use case qui me permettront de définir le fonctionnement de mon site web. Voici donc les use cases :



4.2 Maquette

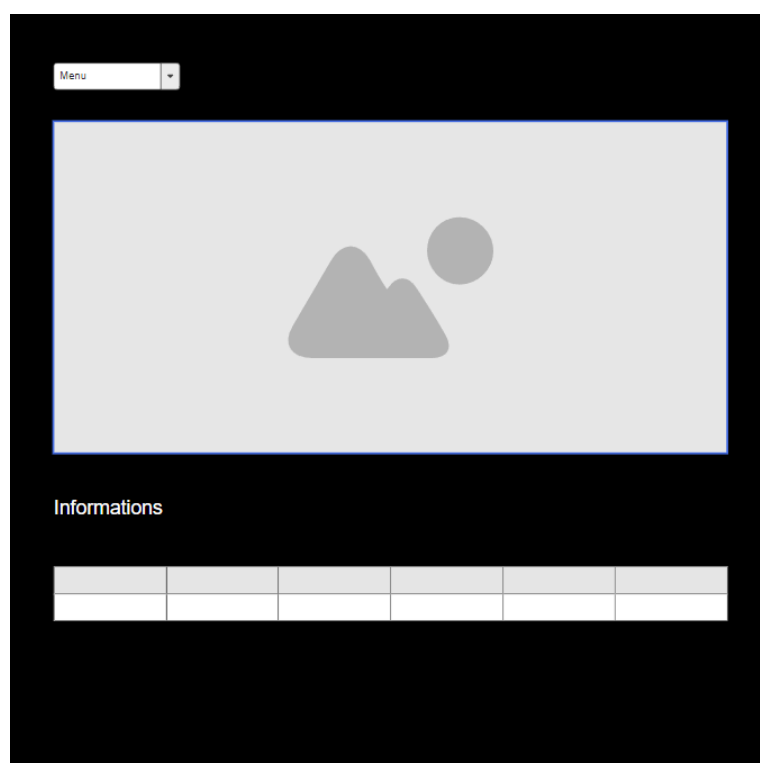
Pour continuer et me donner une idée d'ensemble très simple, j'ai fait des maquettes pour mon site. Voici donc la maquette de la page d'accueil :



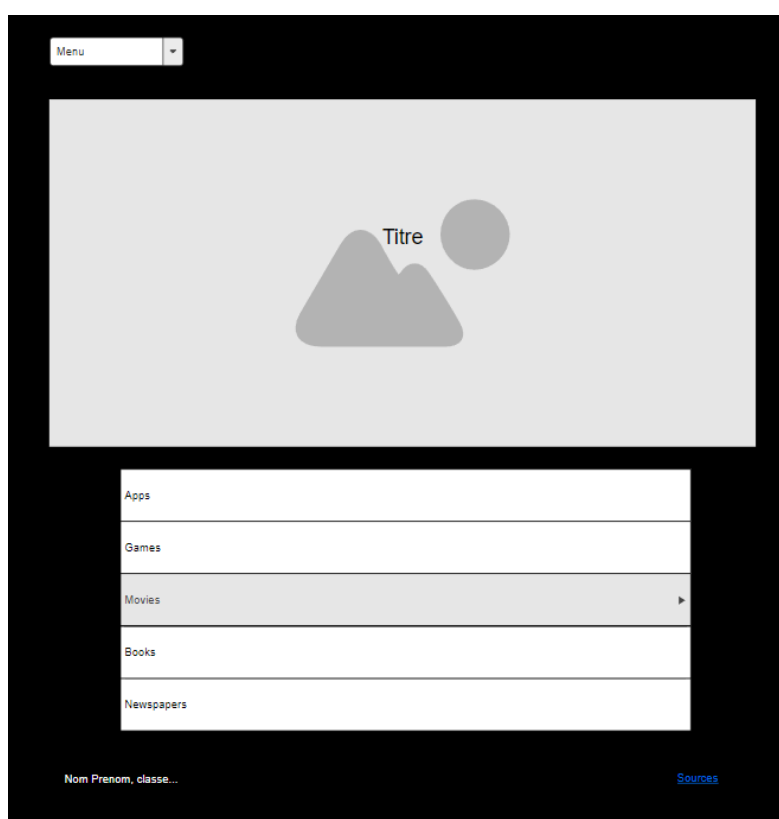
La page qui permettra de donner les informations sur les derniers astéroïdes répertoriés :



La maquette de la page qui donnera la météo de mars :



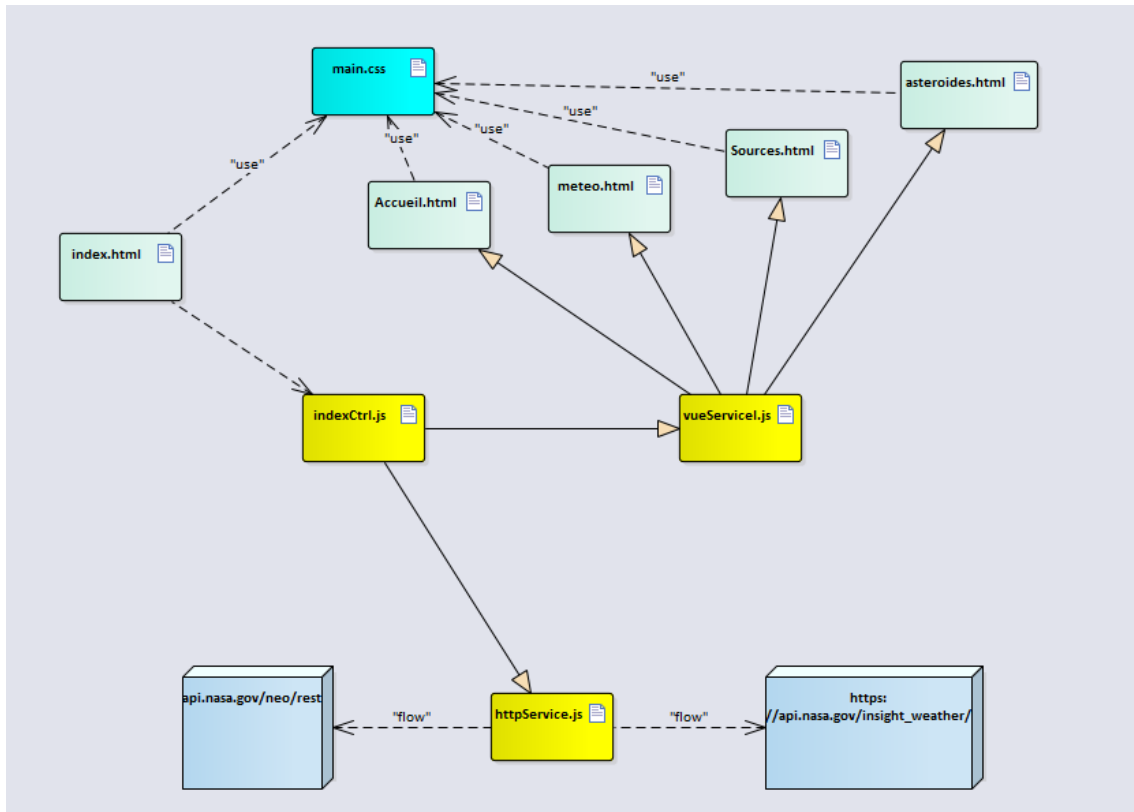
Et voici la page de sources :



5. Conception

5.1 Diagramme de navigation

Pour faire la conception de ce projet, rien de mieux qu'un diagramme de navigation de mon site web. Le voici :



6. Implémentation

6.1 Extraits de code

Il n'y a pas tout le code du projet. Mais on peut le trouver sur mon gitHub (https://github.com/YerlyT/EMF_Module307_Projet/).

Voici un extrait de code spécifique pour la carte du monde. La variable mapid doit être globale pour que la carte s'affiche :

```

initialiserCarte(id) {

    //crée la carte au bon endroit
    mapid = L.map(id[0]).setView([35.2465334, -116.7666609], 5);

    //ajoute les informations en bas a droite
    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);

  //crée les marqueurs
  let marker = L.marker([25.9951, -97.1542]).addTo(mapid);
  marker.bindPopup("SpaceX South Texas Launch Site", { offset: [0, -30]
}).openPopup();

  mapid.setView([35.2465334, -116.7666609], 6);

  marker = L.marker([35.2465334, -116.7666609]).addTo(mapid);
  marker.bindPopup("Base de la Nasa", { offset: [0, -30] }).openPopup();
}

```

6.2 Problèmes rencontrés

Durant la réalisation, l'API que j'utilisais pour faire la météo de Mars à malheureusement été désactivé dû aux mauvaises conditions de mars. La documentation a été changé et m'a prévenu de ce changement :

THIS SERVICE HAS SIGNIFICANT MISSING DATA DUE TO INSIGHT NEEDING TO MANAGE POWER USE:Please check out the [seasonal weather report plot](#) for an illustration of missing data and read [this article](#) about how dust and distance from the sun affect Insight's power situation.

C'est donc pour cela que j'ai changé la partie sur la météo de mars.

7. Tests

7.1 Test des Webservices

Pour continuer, voici la partie la plus importante de l'analyse, les tests des WS. Il aurait été mieux de les faire avant de faire mes maquettes car si les WS ne fonctionnent pas, il aurait fallu changer de Webservice.

Voici le résultat en testant les APIs avec Postman :

```

{
  "links": {
    "next": "http://api.nasa.gov/neo/rest/v1/neo/browse?page=1&size=20&api_key=THWLKdcvg1kZcPqb8BbDND0jR7cnYNAKYVjFdFh",
    "self": "http://api.nasa.gov/neo/rest/v1/neo/browse?page=0&size=20&api_key=THWLKdcvg1kZcPqb8BbDND0jR7cnYNAKYVjFdFh"
  },
  "page": {
    "size": 20,
    "total_elements": 32663,
    "total_pages": 1634,
    "number": 0
  },
  "near_earth_objects": [
    {
      "links": {
        "self": "http://api.nasa.gov/neo/rest/v1/neo/2000433?api_key=THWLKdcvg1kZcPqb8BbDND0jR7cnYNAKYVjFdFh"
      },
      "id": "2000433",
      "neo_reference_id": "2000433",
    }
  ]
}

```

On peut voir que l'on reçoit un bon résultat en avec les données que l'on souhaite. En testant le deuxième, on reçoit aussi des données.

7.2 Test du fonctionnement

Voici les tests du fonctionnement de mon site :

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

8. Hébergement et fonctionnement

Le projet est hébergé sur ce site web et le repos GitHub a déjà été donné plus haut:

<https://yerlyt.emf-informatique.ch/307/Exercices/Projet307/>

Et les APIs utilisées sont :

- api.nasa.gov/neo/rest
- api.nasa.gov/mars-photos/api/v1/rovers/curiosity

9. Conclusion

Ce module était un module très intéressant. C'est l'un des rare module qui touche à la programmation web. De plus le frontend rend le module encore plus intéressant car il nous montre comment fonctionnent les site web qui utilisent des APIs ou des bases de données.

De plus ce module comprenait beaucoup d'exercice pratique même presque un peu trop. Ce qui nous a pris plus temps avant de commencer le projet.

Pour conclure, ce module était très bien appris et très captivant sans oublier que l'on avait du temps en classe pour faire notre rapport personnel.