307 - Réaliser des pages Web interactives

|  |
| --- |
| Rapport Personnel |

Date de création : 15.05.2023  
Version 1 du 06.06.2023

Meyer Yaël

|  |  |
| --- | --- |
|  | Module du 15.05.2023 au 13.06.2023 |

|  |
| --- |
| Table des matières |
|  |

1 Introduction 4

2 Exercice 1 4

3 Exercice 2 5

4 Exercice 3 7

5 Exercice 4 8

6 Exercice 5 8

7 Exercice 6 11

8 Exercice 7 12

9 Exercice 8 13

10 Exercice 9 13

11 Exercice 10 14

12 Exercice 11 14

13 Exercice 12 15

14 Exercice 13 16

15 Exercice 14 17

16 Exercice 15 18

17 Exercice 17 18

18 Projet 20

18.1 Analyse 20

18.2 Conception 24

18.3 Implémentation 24

18.4 Tests 26

18.5 Hébergements et fonctionnement 26

19 Conclusion 27

# Introduction

Ce module 307 reprend quelques peu les bases du module 101 qui traitait de la création d’un site web par HTML/CSS en y rajoutant l’aspect de JavaScript. Voici les principaux objectifs de ce 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.

Nous avons créé un site web personnel qui regroupe tous les exercices vus en module, voici le lien : <https://meyery.emf-informatique.ch>

# Exercice 1

Cet exercice prend en considération pour la première fois JavaScript. Ce dernier a une extension de fichier .js et se compose comme tout autre fichier web (html, css) et comme son nom l’indique il ressemble énormément au langage Java.

Si nous avons besoin d’initialiser certains éléments du JavaScript au lancement de la page alors que l’ensemble de la page doit être chargé normalement, il est possible d’ajouter un appel de la méthode « initCtrl() » dans le body, comme ci-dessous, ou de déplacer le script qui normalement se trouve dans le header à la fin du body :

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

…

<body onload="initCtrl()">

Pour ajouter un écouteur click via JavaScript, il faut créer un élément sur lequel on veut interagir avec dans le fichier html, ici un bouton, lui donner un identifiant comme ceci :

<button id="testez">Teste-moi, James</button>

Ensuite il faut se rendre dans le fichier JS, créer une méthode initCtrl() qui sera appelé dans le html comme vu ci-dessus, et mettre dans cette méthode le code suivant en remplaçant « testez » par l’identifiant choisi précédemment et le nom de la méthode qui sera appelé.

function initCtrl() {

  // Ecouteur du bouton "Testez-moi..."

  document.getElementById("testez").addEventListener("click", testez);

}

function testez() {

   document.getElementById("info").innerHTML =

      "C'est <b>Yaël Meyer</b> qui a pressé le bouton !";

}

Il est aussi possible, mais moins élégant, d’appeler la méthode « testez() » comme ci-dessus, directement dans l’html, c’est-à-dire créer un écouteur dans l’html, comme ci :

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

DOM pour Document Object Model représente et interagit avec des documents HTML et XML qui permet donc d’afficher à l’écran ce qui peut être affiché depuis de l’HTML ou de l’XML. Il est aussi utilisé par JavaScript pour modifier le contenu, la structure ou le style de la page.

# Exercice 2

Ce deuxième exercice ira un peu plus en détail avec JavaScript en implémentant un système de login à un compte. Cette variante se fait en local, ce qui n’est pas le cas dans la vrai vie pour des raisons de sécurité notamment, mais cet exercice permet de comprendre mieux le mécanisme de JS.

Voici le code permettant de créer un formulaire en HTML :

    <form class="user-form">

      <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 />

        </div>

        <div class="field">

          <label for="username">Mot de passe:</label>

          <input type="password" size="30" id="password"

placeholder="un mot de passe svp" />

        </div>

        <input type="button" value="Valider" id="valider" onclick="validerUtilisateur();">

      </fieldset>

    </form>

L’attribut « placeholder » permet d’insérer du texte dans un champ, c’est-à-dire insérer « un mot de passe svp » dans le champ mot de passe comme ceci :



placeholder="un mot de passe svp"

L’attribut autofocus permet de sélectionner automatiquement le champ lorsque la page se charge pour que l’utilisateur puisse interagir avec sans devoir le sélectionner manuellement.

Le code donné contient du JavaScript, il y en a au « onclick » de la balise « input ».

Le bouton « valider » de cet exercice est différent à celui de l’exercice 1 parce que c’est un input comme l’user et mdp qui a l’aspect d’un bouton.

Pour définir une fonction/Méthode en JS, il faut écrire « function <nomDeLaFonction> ([Paramètre.s]) { }

Pour définir une variable en JS, il y a trois possibilités, soit var, let ou const.

Var et let se ressemble énormément à la différence que let est fonctionnel uniquement dans un bloc, comme une boucle par exemple, tandis que var est fonctionnel dans toute la méthode. Const est pour une constante comme en Java.

Les variables ne sont pas typées, c’est-à-dire qu’elles sont dynamiques et prennent tous les types de valeurs.

La console de débogage sur Chrome se trouve en actionnant le raccourci F12 et en se rendant dans l’onglet console.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

La fonction console.log() permet d’afficher une valeur dans la console ce qui peut être pratique pour déboguer.

Il existe plusieurs fonctions de popups pouvant être appelé en JS avec windows.<popup> :

* alert() : popup d’information
* confirm() : popup de confirmation
* prompt() : popup de renseignement texte

Pour récupérer un élément HTML depuis JavaScript grâce à son ID et retirer sa valeur :

document.getElementById("username").value;

Pour convertir un texte en minuscule, comme en Java, il suffit d’utiliser la méthode .toLowerCase().

Il existe plusieurs manières de charger un script, soit en dernière ligne du header pour éviter des ralentissements lors de chargement de grosses bibliothèques soit avec l’attribut « async » qui indique au navigateur qu’aucune modification du DOM ne sera effectuée et donc le script se chargera sur un thread séparé, ce qui améliorera les performances.

# Exercice 3

Il existe plusieurs types de boutons programmables en HTML lesquels sont :

* type="button" : ce type crée un simple bouton cliquable sans fonctionnalité particulière.
* type="submit" : ce type crée un bouton qui envoie un formulaire vers un serveur lorsqu'il est cliqué. Le visiteur sera conduit à la page indiqué dans l'attribut action.
* type="reset" : ce type crée un bouton qui réinitialise les valeurs d'un formulaire à leurs valeurs par défaut.
* type="image" : ce type crée un bouton qui est représenté par une image spécifiée dans l'attribut src.
* type="checkbox" : ce type crée un bouton qui peut être coché ou décoché.
* type="radio" : ce type crée un bouton qui peut être sélectionné dans un groupe de boutons liés.
* type="file" : ce type crée un bouton qui permet à l'utilisateur de sélectionner un fichier sur son ordinateur.

Un code PHP doit toujours commencer par le mot clé <?php et finir par ?>

Pour récupérer une information du fichier HTML depuis PHP, il faut utiliser le mot clé « GET ».

Pour concaténer des chaînes de caractères en PHP, il faut utiliser le . comme ceci :

 echo "username: ".$username."</br>";

Pour convertir une chaîne de caractère en minuscule en PHP, il faut utiliser la méthode strtolower().

Pour renvoyer une information au client depuis PHP, il est possible d’utiliser « echo » ou « print »

Dans le code PHP suivant, on trouve aussi du langage HTML.

<?php

/\*

  But : php

  Auteur : Yaël Meyer

  Date :   15.05.2023 / V1.0

\*/

// 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>";

  }

}

?>

# Exercice 4

L’outil [JSFiddle](https://jsfiddle.net/) permet de créer des mini-projets directement sur Internet avec du HTML, CSS et JavaScript en n’étant pas obligé de renseigner les balises mères (<html>,…) et permet d’enregistrer ses projets pour être rapide. Cet outil permet notamment de ne pas avoir à lancer son IDE et mettre en place ses fichiers.

# Exercice 5

Ce cinquième exercice approfondi les connaissances en JavaScript avec une série de questions, les voicis :

Pour effacer le contenu de la console :

Console.clear() ;

Console was cleared

Pour créer une variable « a » :

let a;

Pour afficher le contenu de « a » :

console.log(a);

Pour stocker la valeur 15 dans « a » :

a = 15;

Pour afficher le contenu de « a » sous la forme « Ma variable a = ? » :

console.log("Ma variable a = " + a) ;

Ma variable a = 15

Pour créer une variable b et lui assigner la valeur 9 :

let b = 9 ;

Pour afficher le contenu de « b » sous la forme « Ma variable b = ? » :

console.log("Ma variable b = " + b);

Ma variable b = 9

Pour additionner « a » et « b » en affichant le résultat sous la forme « 15 + 9 = ? » :

console.log("15 + 9 = " + (a + b));

15 + 9 = 24

Avec un littéral :

console.log(`15 + 9 = ${a + b}`);

15 + 9 = 2

Pour une soustraction :

console.log(`15 - 9 = ${a - b}`);

15 - 9 = 6

Pour une multiplication :

console.log(`15 \* 9 = ${a \* b}`);

15 \* 9 = 135

Pour une division :

console.log(`15 / 9 = ${a / b}`);

15 / 9 = 1.6666666666666667

Pour stocker « bonjour » dans « a » :

a = "bonjour";

Pour stocker « les amis » dans « b » :

b = "les amis";

Pour afficher « bonjour les amis » :

console.log(a + " " + b);

bonjour les amis

Avec un littéral :

console.log(`${a} ${b}`);

bonjour les amis

Pour stocker « true » dans « a » :

a = true;

Pour stocker « false » dans « b » :

b = false;

Pour effectuer AND entre « a » et « b » et afficher le résultat sous la forme « true AND false = ? » :

console.log("true AND false = " + a && b);

false

Pour effectuer OR entre « a » et « b » et afficher le résultat sous la forme « true OR false = ? » :

console.log("true OR false = " + a || b);

true OR false = true

Pour stocker la date du jour dans « a » :

a = new Date();

Pour calculer une nouvelle date dans « b » qui est 61 jours avant :

b = new Date(a);

b.setDate(a.getDate() - 61) ;

Pour afficher les dates dans « a » et « b » :

console.log("Date et heure : " + a.toLocaleString());

Date et heure : 16/05/2023 08:24:11

console.log("Date et heure : " + b.toLocaleString());

Date et heure : 16/03/2023 08:24:11

console.log("Date : " + a.toLocaleDateString());

Date : 16/05/2023

console.log("Date : " + b.toLocaleDateString());

Date : 16/03/2023

console.log("Heure : " + b.toLocaleTimeString());

Heure : 08:24:11

onsole.log("Heure : " + a.toLocaleTimeString());

Heure : 08:24:11

Pour connaître le type de plusieurs variables de types différents :

console.log(typeOf(a) + " " + typeOf(b) + " " + typeOf(c) + " " + typeOf(d) + " " + typeOf(e));

number string boolean date null

En JS, il y plusieurs manières de déclarer une variable, mais il faut privilégier « let » et pour déclarer une constante, il faut écrire « const » et écrire le nom de la constante en majuscule comme convention.

« var » est comme « let » mais il laisse trop de possibilité à JS, c’est-à-dire qu’il est possible de renseigner 2x la même variable.

Si l’on déclare une variable sans mot clé, elle sera d’office de type « var ».

Il y a 7 type primitifs en JS :

String

Number

Boolean

Null

Undefined

Symbol

Bigint

L’instruction « typeOf » permet de connaître le type d’une variable.

console.log(typeOf(a) + " " + typeOf(b) + " " + typeOf(c) + " " + typeOf(d) + " " + typeOf(e));

number string boolean date null

# Exercice 6

Cet exercice reprend les bases de l’exercice 1, sauf que cette fois lorsqu’on appuie sur le bouton, c’est le jour de la semaine actuel qui s’affiche.

Pour récupérer le numéro du jour actuel, il faut créer une variable date avec la date actuelle et créer une autre variable numero pour récupérer le numero du jour :

  let date = new Date();

  let numero = date.getDay();

Pour insérer du texte dans une balise HTML, il faut créer une balise « p » et entrer le texte entre les balises :

<p>

Cliquez sur le bouton ci-dessous pour découvrir quel jour nous sommes.

</p>

Pour insérer du code dans une balise HTML, il faut utiliser la manière littérale pour pouvoir reprendre la valeur d’une variable comme ceci :

`<p>${tab[numero - 1]}</p>`;

Pour créer un tableau vide il suffit de faire :

const tab1 = [];

Pour créer un tableau avec des valeurs :

const tab1 = [];

Pour ajouter une valeur au début :

noms.unshift('AA'); // ajoute AA au début du tablea

Pour ajouter des valeurs à la fin :

noms.push('Eric'); // ajoute Eric à la fin du tableau

Pour supprimer une valeur au début :

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

Pour supprimer une valeur à la fin :

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

Pour récupérer une valeur dans un tableau à une position donnée :

tab[numero - 1]

Pour afficher l’ensemble des valeurs d’un tableau :

for (let i=0; i<noms.length; i++) {

  console.log(i + " : " + noms[i]); // 0 : Alice …

}

# Exercice 7

Cet exercice traite les différentes boucles en JS :

For :

let html = "<p>for (let i=0; i<5; i++){}</p>";

  for (let i = 0; i < 5; i++) {

    html += `<p>i = ${i}</p>`;

  }

  html +=

    "<p>--> A utiliser si on sait que l'on veut étirer x fois (x connu avant de commencer la boucle)</p>";

  document.getElementById("info").innerHTML = html;

While :

let html = "<p>while (i<5){}</p>";

  let i = 0;

  while (i<5){

    html += `<p>i = ${i}</p>`;

    i++;

  }

  html +=

  "<p>--> A utiliser si on ne sait pas le nombre d'itérations au dméarrage de la boucle</p>";

  document.getElementById("info").innerHTML = html;

Do While :

let html = "<p>do {} while (i<5);</p>";

  let i = 0;

  do {

    html += `<p>i = ${i}</p>`;

    i++;

  } while (i<5);

  html +=

    "<p>--> A utiliser si on ne sait pas le nombre d'itérations au dméarrage de la boucle mais avec un passage obligatoire</p>";

  document.getElementById("info").innerHTML = html;

# Exercice 8

Cet exercice 8 implémente les aspects de tableau JSON pour JavaScript Object Notation. JSON fonctionne avec un mécanisme de clé-valeur. Voilà mon code JS :

function parcourirUnTableauJSON() {

  let txt = "";

  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];

    txt += `<p>${i}. ${personne.prenom} ${personne.nom} ${personne.age}</p>`;

  }

  document.getElementById("info").innerHTML = txt;

}

# Exercice 9

L’exercice 9 parle des objets en JavaScript, il y a plusieurs manières de déclarer des objets mais la plus importante est la suivante :

class Personne {

constructor(prenom, nom, age) {

this.prenom = prenom;

this.nom = nom;

this.age = age;

}

toString() {

return this.nom + " " + this.prenom + " (" + this.age + ")";

}

}

Comme on peut le voir, la création ressemble beaucoup à ce qu’on connaît en Java, on créé une classe et on lui donne un constructeur. Ensuite, les attributs seront déclarés avec le mot clé « this ». Il est aussi recommandé de créer un « toString » dans la classe pour formater l’affichage.

# Exercice 10

Cet exercice 10 est l’un des plus importants de ce module, il reprend tous les aspects vus précédemment en rajoutant le côté MVC (Modèle Vue Contrôleur).

.onreadystatechange est un écouteur de l’évènement readystatechange. C’est-à-dire qu’une fois que tout le document HTML est créé, que les scripts sont chargés, la méthode sera appelée.

# Exercice 11

Pour créer une classe en JavaScript, c’est globalement comme en Java à la différence que les appels dans la classe ainsi que la déclaration d’attributs dans le constructeur se feront avec le mot clé « this ». Voici un exemple :

class Personne {

constructor(prenom, nom, age) {

this.prenom = prenom;

this.nom = nom;

this.age = age;

}

toString() {

return this.nom + " " + this.prenom + " (" + this.age + ")";

}

}

Pour faire en sorte qu’une classe soit accessible dans la vue, il suffit de mettre le mot clé « window » devant l’attribut comme ceci :

/\*

\* DOM PRET : DEMARRAGE DE L'APPLICATION dans le fichier indexCtrl.js

\*/

document.onreadystatechange = function () {

if (document.readyState === "complete") {

window.ctrl = new Ctrl(); // ou ctrl = new Ctrl();

ctrl.afficherPersonnes();

}

};

# Exercice 12

Il y a trois types de déclaration de fonctions en JS, les voici :

// 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

Foncton IIFE (Immediately-Invoked Function) :

(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

# Exercice 13

Un cookie sert à stocker des informations JSON en cache sur le disque dur du client pour ensuite pouvoir réutiliser les informations qui y sont stockés.

Pour stocker des informations dans un cookie, il faudra premièrement créer un objet JSON et le passer en string comme ceci, par exemple :

let objetJSON = {

    prenom: “Yaël”

    nom: “Meyer”

  };

  // Convertir l'objet json "personneJson" en chaine de caractère.

  //

  let jsonString = JSON.stringify(objetJSON);

Ensuite on peut créer le cookie :

let maintenant = new Date();

  maintenant.setMinutes(maintenant.getMinutes() + minutes);

  // Créer le cookie.

  document.cookie = "cookie\_exercice\_13=" + jsonString + "; expires=" + maintenant + "; path=/";

  console.log("Data saved in cookie");

Pour récupérer le cookie, il faudra tester la présence du cookie et ensuite extraire le contenu du cookie avec split pour et denouveau le convertir en JSON :

// 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 content = document.cookie.split("=")[1];

    // Convertir la chaîne de caractères en objet Json.

    let jsonContent = JSON.parse(content);

# Exercice 14

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

Jquery est une librairie open source qui simplifie l’interaction entre du HTML et du JavaScript. Ce dernier est très bien documenté et a une grande communauté de développeurs.

* Comment reconnaît-on du code jquery ?

On reconnaît du code jquery grâce au signe $ qui est l’objet jquery.

* Quelles sont les 2 tâches principales de jquery ?

Sélection d’éléments du DOM et leur manipulation.

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

Par nom de balise : $('tag')

Par classe CSS : $('.classe')

Par ID : $('#id')

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

La fonction jQuery qui est appelée lorsque la page a été chargée est $(document).ready(). Elle permet d'exécuter du code JavaScript une fois que le DOM est prêt. Voici un exemple :

$(document).ready(function() {

// Code à exécuter une fois que le DOM est prêt

// ...

});

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

Parent : parent()

Suivant : next()

Précédent : prev()

Enfant : children()

Frères : siblings()

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

$('element').css('propriete', 'valeur');

* + Ajouter un élément avant ou après l'élément sélectionné ?

$('element').before('<p>Nouvel élément avant</p>');

$('element').after('<p>Nouvel élément après</p>');

* + Supprimer un élément ?

$('element').remove();

* + Gérer les événements (click, hover, etc.)

$('element').on('click', function() {

// Code à exécuter lors du clic sur l'élément

// ...

});

* + Cacher ou afficher un élément ?

$('element').hide();

$('element').show();

* Comment lire un fichier avec ajax ?

$.ajax({

url: 'fichier.txt',

success: function(data) {

// Code à exécuter une fois que le fichier est chargé avec succès

// Utilisez la variable 'data' pour accéder au contenu du fichier

}

});

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

$.getJSON('fichier.json', function(data) {

// Code à exécuter une fois que le fichier JSON est chargé avec succès

// Utilisez la variable 'data' pour accéder aux données JSON

});

* Comment charger un fichier HTML dans un div existant ?

$('div').load('fichier.html');

* Comment intégrer jquery dans vos pages ?

Avec la balise « script » :

<head>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

</head>

# Exercice 15

Pour récupérer la valeur d’un champ en JQ :

let couleur = $("#couleurs").val();

#couleurs étant l’id de l’attribut

Pour modifier une propriété css :

$("div").css("backgroundColor", couleur);

Pour utiliser une animation en css :

 $("div").fadeOut(1000);

  $("div").toggle().slideToggle(1000);

# Exercice 17

Pour cet exercice, nous avons dû présenter les webservices, voici ce qui a été appris :

REST pour Representational State Transfer est un ensemble de webservices qui se basent sur http pour permettre l’échange de données. Les types de requêtes en REST sont :

* GET -> récupérer
* POST -> créer
* PUT -> mettre à jour
* DELETE -> supprimer

Les webservices SOAP pour Simple Object Access Protocol sont une autre architecture de webservices cette fois basé sur du XML pour encapsulés les informations.

* Pour réaliser un GET en REST dans Postman :
  + Sélectionnez la méthode HTTP "GET".
  + Entrer l'URL de l'API ou de l'endpoint
  + Cliquer sur le bouton "Send" pour envoyer la requête GET.
  + Le résultat de la requête sera affiché dans la réponse de Postman.
* Pour réaliser un POST en REST dans Postman :
  + Sélectionnez la méthode HTTP "POST".
  + Entrer l'URL de l'API ou de l'endpoint
  + Dans l'onglet "Body", sélectionner le format de données (par exemple, JSON) et saisir les données à envoyer.
  + Cliquer sur le bouton "Send" pour envoyer la requête POST.
  + La réponse de la requête sera affichée dans Postman.
* Pour réaliser un PUT en REST dans Postman
  + Sélectionner la méthode HTTP "PUT".
  + Entrez l'URL de l'API ou de l'endpoint
  + Dans l'onglet "Body", sélectionner le format de données (par exemple, JSON) et saisir les données à envoyer pour mettre à jour la ressource.
  + Cliquer sur le bouton "Send" pour envoyer la requête PUT.
  + La réponse de la requête sera affichée dans Postman.
* Pour réaliser un DELETE en REST dans Postman :
  + Sélectionner la méthode HTTP "DELETE".
  + Entrez l'URL de l'API.
  + Cliquer sur le bouton "Send" pour envoyer la requête DELETE.
  + La réponse de la requête sera affichée dans Postman.

# Projet

## Analyse

Pour le projet de ce module, j’aimerais réaliser un site orienté fitness. Mon site sera principalement divisé en trois parties :

* Exercices musculations
* Calories brûlées par activité
* Carte avec les fitness du canton

J’irai chercher mes informations sur les APIs suivantes :

https://exercises-by-api-ninjas.p.rapidapi.com/v1/exercises

https://calories-burned-by-api-ninjas.p.rapidapi.com/v1/caloriesburned

C’est-à-dire qu’il ressemblera à ça :

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Une image contenant texte, capture d’écran, affichage, diagramme

Description générée automatiquement

Une image contenant texte, capture d’écran, affichage, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, affichage, diagramme

Description générée automatiquement

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Voici l’use case :

Une image contenant diagramme, capture d’écran, ligne, cercle

Description générée automatiquement

## Conception

Ci-dessous se trouve le diagramme de navigation :

Une image contenant diagramme, texte, ligne, capture d’écran

Description générée automatiquement

## Implémentation

Premièrement, j’ai créé un menu sur chaque page, pour ouvrir le menu il suffit d’appuyer dessus et cliquer à côté pour le refermer, voici sont code :

afficherMenu() {

    $(".menu").click(function() {

      $(this).next().toggle();

    });

    $("#enteteAccueil ul li").click(function() {

      $(this).parent().hide();

    });

    $(document).click(function(event) {

      const target = event.target;

      const menu = $("#enteteAccueil ul");

      if (!menu.is(target) && !$(".menu").is(target)) {

        menu.hide();

      }

    });

  }

Pour rechercher les exercices en fonction des filtres, je vais chercher le contenu des filtres et j’effectue une recherche en fonction et je créé une cellule avec son nom d’affiché :

rechercheFiltreListe() {

    let type = $("#filtreType").val();

    let muscle = $("#filtreMuscle").val();

    let difficulty = $("#filtreDifficulty").val();

    var settings = {

      "url": "https://exercises-by-api-ninjas.p.rapidapi.com/v1/exercises?type=" + type + "&muscle=" + muscle + "&difficulty=" + difficulty,

      "method": "GET",

      "timeout": 0,

      "headers": {

        "X-RapidAPI-Key": "7d63c19110mshc301819ad761fb4p172cb6jsn0f8b6be8bb2c",

        "X-RapidAPI-Host": "exercises-by-api-ninjas.p.rapidapi.com"

      },

    };

    $.ajax(settings).done(function (response) {

      $("#resultats").empty();

      // Parcourir les données de la réponse de l'API

      $.each(response, function (index, exercice) {

        // Créer une cellule pour chaque exercice

        var cellule = $('<div class="cellule">');

        cellule.data('exercice-info', exercice);

        cellule.text(exercice.name);

        // Ajouter la cellule au conteneur

        $("#resultats").append(cellule);

      });

    });

  }

Globalement le même principe pour filtrer en fonction du nom et pour le filtre de calories.

## Tests

A picture containing text, screenshot, number, font

Description automatically generated

## Hébergements et fonctionnement

J’ai hébergé mon site sur le domaine de l’école à l’adresse :

<https://meyery.emf-informatique.ch/307Projet/307Projet.html>

J’ai transféré les informations via Filezilla et tout fonctionne correctement depuis n’importe quel navigateur.

# Conclusion

Pour conclure ce RP 307, je dirais que je n’avais jamais abordé JavaScript parce que je comprenais pas trop ce que c’était et je trouvais dur. Finalement, je me suis rendu compte que ce dernier ressemble énormément à Java et donc j’ai pu me débrouiller. Je suis content d’avoir appris ce sujet parce que je sais que je vais l’utiliser et que c’est une partie importante du métier d’informaticien, surtout dans le développement.