307 - RÉALISER DES PAGES WEB INTERACTIVES

Rapport personnel

Date de création : 15.05.2023  
Version 1 du 17.06.2023

Sascha Widmer

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

Table des matières

[1 Introduction 6](#_Toc137936800)

[1.1 Objectifs 6](#_Toc137936801)

[2 Exercices 6](#_Toc137936802)

[2.1 Exercice 1 (Base HTML5) 6](#_Toc137936803)

[2.1.1 Objectif de l’exercice 6](#_Toc137936804)

[2.1.2 Explications / Descriptions 6](#_Toc137936805)

[2.1.3 Extrait de code 7](#_Toc137936806)

[2.1.4 Capture 7](#_Toc137936807)

[2.2 Exercice 2 (Login) 8](#_Toc137936808)

[2.2.1 Objectif de l’exercice 8](#_Toc137936809)

[2.2.2 Explications / Descriptions 8](#_Toc137936810)

[2.2.3 Extrait de code 9](#_Toc137936811)

[2.2.4 Capture 10](#_Toc137936812)

[2.3 Exercice 3 (Login avec PHP) 10](#_Toc137936813)

[2.3.1 Objectif de l’exercice 10](#_Toc137936814)

[2.3.2 Explications / Descriptions 10](#_Toc137936815)

[2.3.3 Extrait de code 11](#_Toc137936816)

[2.3.4 Capture 12](#_Toc137936817)

[2.4 Exercice 4 (JSFiddle) 12](#_Toc137936818)

[2.4.1 Objectif de l’exercice 12](#_Toc137936819)

[2.4.2 Explications / Descriptions 12](#_Toc137936820)

[2.4.3 Extrait de code 12](#_Toc137936821)

[2.4.4 Capture 13](#_Toc137936822)

[2.5 Exercice 5 (Variable en Javascript) 13](#_Toc137936823)

[2.5.1 Objectif de l’exercice 13](#_Toc137936824)

[2.5.2 Explications / Descriptions 13](#_Toc137936825)

[2.5.3 Extrait de code 13](#_Toc137936826)

[2.5.4 Capture 15](#_Toc137936827)

[2.6 Exercice 6 (La structure de contrôle switch) 15](#_Toc137936828)

[2.6.1 Objectif de l’exercice 15](#_Toc137936829)

[2.6.2 Explications / Descriptions 15](#_Toc137936830)

[2.6.3 Extrait de code 15](#_Toc137936831)

[2.6.4 Capture 16](#_Toc137936832)

[2.7 Exercice 7 (Les boucles) 17](#_Toc137936833)

[2.7.1 Objectif de l’exercice 17](#_Toc137936834)

[2.7.2 Explications / Descriptions 17](#_Toc137936835)

[2.7.3 Extrait de code 17](#_Toc137936836)

[2.7.4 Capture 18](#_Toc137936837)

[2.8 Exercice 8 (Tableau d’objets JSON) 19](#_Toc137936838)

[2.8.1 Objectif de l’exercice 19](#_Toc137936839)

[2.8.2 Explications / Descriptions 19](#_Toc137936840)

[2.8.3 Extrait de code 19](#_Toc137936841)

[2.8.4 Capture 19](#_Toc137936842)

[2.9 Exercice 9 (Démo sur les objets en JavaScript) 20](#_Toc137936843)

[2.9.1 Objectif de l’exercice 20](#_Toc137936844)

[2.9.2 Explications / Descriptions 20](#_Toc137936845)

[2.9.3 Extrait de code 20](#_Toc137936846)

[2.9.4 Capture 22](#_Toc137936847)

[2.10 Exercice 10 (Programmation orientée "objets") 23](#_Toc137936848)

[2.10.1 Objectif de l’exercice 23](#_Toc137936849)

[2.10.2 Explications / Descriptions 23](#_Toc137936850)

[2.10.3 Extrait de code 23](#_Toc137936851)

[2.10.4 Capture 25](#_Toc137936852)

[2.11 Exercice 11 (Programmation orientée "classe" en Javascript) 25](#_Toc137936853)

[2.11.1 Objectif de l’exercice 25](#_Toc137936854)

[2.11.2 Explications / Descriptions 25](#_Toc137936855)

[2.11.3 Extrait de code 25](#_Toc137936856)

[2.11.4 Capture 27](#_Toc137936857)

[2.12 Exercice 12 (Fonctions et IIFE en JavaScript) 27](#_Toc137936858)

[2.12.1 Objectif de l’exercice 27](#_Toc137936859)

[2.13 Exercice 13 (Les cookies) 28](#_Toc137936860)

[2.13.1 Objectif de l’exercice 28](#_Toc137936861)

[2.13.2 Explications / Descriptions 28](#_Toc137936862)

[2.13.3 Extrait de code 28](#_Toc137936863)

[2.13.4 Capture 29](#_Toc137936864)

[2.14 Exercice 14 (Bases de jQuery) 30](#_Toc137936865)

[2.14.1 Objectif de l’exercice 30](#_Toc137936866)

[2.15 Exercice 15 (Première utilisation de jQuery) 30](#_Toc137936867)

[2.15.1 Objectif de l’exercice 30](#_Toc137936868)

[2.15.2 Explications / Descriptions 30](#_Toc137936869)

[2.15.3 Extrait de code 30](#_Toc137936870)

[2.15.4 Capture 31](#_Toc137936871)

[2.15.4.1 JS 31](#_Toc137936872)

[2.15.4.2 JQ 31](#_Toc137936873)

[2.16 Exercice 16 (Utilisation de jQuery) 32](#_Toc137936874)

[2.16.1 Objectif de l’exercice 32](#_Toc137936875)

[2.16.2 Extrait de code 32](#_Toc137936876)

[2.16.3 Capture 33](#_Toc137936877)

[2.17 Exercice 17 (WebServices - découverte) 34](#_Toc137936878)

[2.17.1 Objectif de l’exercice 34](#_Toc137936879)

[2.17.2 REST 34](#_Toc137936880)

[2.17.3 SOAP 34](#_Toc137936881)

[2.17.4 GET 35](#_Toc137936882)

[2.17.5 POST 35](#_Toc137936883)

[2.17.6 PUT 35](#_Toc137936884)

[2.17.7 DELETE 35](#_Toc137936885)

[2.18 Exercice 18 (Application HTML5 appelant un WS) 36](#_Toc137936886)

[2.18.1 Objectif de l’exercice 36](#_Toc137936887)

[2.18.2 Explications / Descriptions 36](#_Toc137936888)

[2.18.3 Extrait de code 36](#_Toc137936889)

[2.18.4 Capture 37](#_Toc137936890)

[2.19 Exercice 20 (Application de login sur un serveur PHP) 38](#_Toc137936891)

[2.19.1 Objectif de l’exercice 38](#_Toc137936892)

[2.19.2 Explications / Descriptions 38](#_Toc137936893)

[2.19.3 Extrait de code 38](#_Toc137936894)

[2.19.4 Capture 40](#_Toc137936895)

[3 Projet 41](#_Toc137936896)

[3.1 Analyse 41](#_Toc137936897)

[3.1.1 Maquettes 41](#_Toc137936898)

[3.1.1.1 Accueil 41](#_Toc137936899)

[3.1.1.2 Infos artiste 41](#_Toc137936900)

[3.1.1.3 Info perso 42](#_Toc137936901)

[3.1.2 Use case 42](#_Toc137936902)

[3.2 Conception 43](#_Toc137936903)

[3.2.1 Diagramme de navigation 43](#_Toc137936904)

[3.3 Implémentation 43](#_Toc137936905)

[3.4 Hébergement et fonctionnement 46](#_Toc137936906)

[3.4.1 Login 46](#_Toc137936907)

[3.4.2 Info sur les artistes 47](#_Toc137936908)

[3.4.3 Recommandations 49](#_Toc137936909)

[4 Conclusion 50](#_Toc137936910)

# Introduction

## Objectifs

* 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.

# Exercices

## Exercice 1 (Base HTML5)

### Objectif de l’exercice

Le but de ce premier exercice est d’installer notre outil de codage (Visual Studio Code) et d’y installer quelques plugins qui peuvent nous aider à coder plus rapidement tel que des formateurs. Et ensuite de faire un peu de code JS pour commencer à comprendre son fonctionnement.

### Explications / Descriptions

Dans cet exercice il y a 2 boutons que quand on les presse exécutent deux fonctions (méthode) qui écrivent quelque chose en dessous des boutons.

* Comment ajouter un écouteur "click" via Javascript.

Pour ajouter un écouteur sur un bouton il faut en premier sélectionner le bouton en utilisant "getElementById" et mettre son ID (dans l’exemple "testez"). Puis après il faut ajouter "addEventListener" et lui dire quand la méthode doit s’effectuer (dans ce cas la méthode doit s’effectuer lors d’un clique) et le nom de la méthode qui doit s’effectuer (dans l’exemple "testez").

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

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

Pour ajouter un écouteur directement sur un bouton il faut mettre l’attribut "onclick" sur notre bouton et lui donner en paramètre la méthode que l’on souhaite exécuter.

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

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

Pour exécuter du code JS lorsque la page HTML a fini d'être chargée il faut mettre l’attribut "onload" au body et donner la méthode que l’on souhaite exécuter en paramètre.

<body onload="initCtrl()">

* Expliquez ce que signifie DOM.

DOM signifie Document Object Model. Lorsque le navigateur se charge il créer un DOM de la page.

Voici l'arbre d'objets du DOM HTML :

Une image contenant texte, diagramme, Plan, Dessin technique

Description générée automatiquement

* Où le script Javascript peuvent être chargés et pourquoi.

Dans la balise body car cela permet de charger tous les éléments de notre page avant de pouvoir exécuter notre code js qui fera référence à certains éléments de notre page. Il faut aussi préciser notre script js dans la balise « head » à l’aide de la balise « script ».

### Extrait de code

function initCtrl() {

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

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

}

function testez() {

document.getElementById("info").innerHTML =

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

}

function testez2() {

document.getElementById("info").innerHTML =

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

}

### Capture

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

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

## Exercice 2 (Login)

### Objectif de l’exercice

Le but de cet exercice est de découvrir comment faire un formulaire en faisant une petite interface de login.

### Explications / Descriptions

Dans cet exercice lorsque l’on clique sur le bouton la fonction qui va vérifier si les informations de login correspondent à ce qu’il y a d’enregistré pour les comparer.

* À quoi sert l’attribut « placeholder » ?

Le placeholder fournit un petit texte indicatif sur les données que l’on doit rentrer dans un champ. Par exemple pour une date le placeholder pourrait être :

placeholder="DD/MM/YYYY"

* À quoi sert l’attribut « autofocus » ?

L’attribut autofocus permet lorsque la page se charge de mettre directement le focus sur l’élément par exemple de positionner automatiquement le curseur dans le champ pour que l’utilisateur puisse commencer à écrire directement.

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

Oui, il y a la fonction "validerUtilisateur()" et l’attribut "onclick".

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

Dans cet exercice nous testons que ce que l’utilisateur a mis correspond à nos données.

* Expliquer les fonctions alert(), confirm() et prompt().

alert() permet d’afficher une boite de dialogue avec le texte que l’on met en paramètre.

confirm() permet d’afficher une boite de dialogue avec le texte que l’on met en paramètre et la possibilité de mettre "ok" ou "annuler".

prompt() permet d’afficher une boite de dialogue avec le texte que l’on met en paramètre et la possibilité de mettre "ok" ou "annuler" et de saisir un texte.

* À quoi sert l’attribut « placeholder » dans la balise input ?

Elle sert à préciser ce que l’on veut que l’utilisateur y mette.

* À quoi sert l’attribut « autofocus » dans la balise input ?

Elle sert à directement la sélectionner lors du chargement de la page pour que l’utilisateur puisse directement commencer à y écrire.

* Une explication pour récupérer un élément HTML depuis Javascript grâce à son id.

Pour récupérer un élément HTML depuis Javascript grâce à son id on va utiliser la fonction document.getElementById qui nous permet de retrouver un élément HTML grâce à un ID qu’on lui a donné avec l’attribut id="".

* Comment écrire sur la console de votre navigateur ? Quelle est l'utilité ? Où se trouve la console de débogage de Chrome ?

En utilisant console.log(), cela nous permet de voir les erreurs et la ligne plus spécifiquement pour que l’on puisse débugger notre code.

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

Description générée automatiquement

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

function validerUtilisateur() {

...

}

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

let username = "admin";

### Extrait de code

Voici le code HTML pour le formulaire :

<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>

Voici le code pour le JS :

let username = "admin";

let password = "emf123";

function validerUtilisateur() {

if (document.getElementById("username").value == username) {

if (document.getElementById("password").value == password) {

window.alert("Validation OK.");

} else {

window.alert("Password not OK.");

}

} else {

window.alert("Username not OK.");

}

}

### Capture

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

Description générée automatiquement

Une image contenant texte, capture d’écran

Description générée automatiquement

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

Description générée automatiquement

## Exercice 3 (Login avec PHP)

### Objectif de l’exercice

Le but de l’exercice 3 est d’améliorer un peu l’exercice précèdent, notamment la vue en modifiant le bouton, et aussi de faire la validation du login avec un fichier PHP.

### Explications / Descriptions

Dans cet exercice lorsque le formulaire est rempli on exécute le fichier PHP qui va vérifier si les login correspondent avec ce qui est enregistré et va renvoyer l’utilisateur sur une autre page qui donne le nom d’utilisateur et le mot de passe si il est correct.

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

Voici une liste des différents types de boutons en HTML :

* 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 indiquée 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.
* Par quoi commence et doit se terminer un code PHP ?

Un code PHP doit se terminer par "?>" pour fermer la balise PHP.

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

La commande qui nous permet de convertir une chaine de caractères en minuscule est " strtolower " :

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

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

La commande qui nous permet de renvoyer quelque chose vers le client est "echo". Comme dans cet exemple :

// affichage des infos reçues

echo "username: ".$username."</br>";

echo "password: ".$password."</br>";

### Extrait de code

HTML :

<body>

<div id="container">

<form class="user-form" action="http://widmers.emf-informatique.ch/307/Exercices/Exercice\_3/login.php"

method="POST">

<fieldset>

<legend>Identification:</legend>

<div class="field">

<label for="username">Nom d'utilisateur:</label>

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

</div>

<div class="field">

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

<input name="password" type="password" size="30" id="password" placeholder="un mot de passe svp" />

</div>

<div class="button">

<input type="submit" value="Valider" />

</div>

</fieldset>

</form>

</div>

</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>";

}

}

?>

### Capture

Une image contenant logiciel, capture d’écran, Logiciel multimédia

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, Police, blanc, conception

Description générée automatiquement

## Exercice 4 (JSFiddle)

### Objectif de l’exercice

Dans cet exercice nous découvrons l’outil JSFiddle (<http://jsfiddle.net/>) qui nous permet de tester du code HTML, CSS, JS rapidement et efficacement car il n’y a pas besoin de créer tout la structure de fichier.

### Explications / Descriptions

Cet exercice se trouve sur JSFiddle, nous avons fait une fonction JS qui s’exécuter dès que l’on clique sur le carre bleu (un bouton) et qui va générer une pop-up

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

Fiddle est un site web qui nous permet de faire du code HTML, CSS, JS

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

L’avantage est que nous n’avons pas besoin de mettre toute la structure de fichier en place et que nous pouvons juste tester notre code.

### Extrait de code

HTML

<button onclick="messageAlert()"\>

CSS

button {

width: 100px;

height: 100px;

background-color: blue;

}

JS

function messageAlert() {

window.alert("Quelqu'un a cliqué sur le cube blue");

}

### Capture

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

## Exercice 5 (Variable en Javascript)

### Objectif de l’exercice

Dans l’exercice 5 nous devons faire plusieurs commande JS pour plusieurs points, ce qui nous permet de découvrir encore plus JS et nous montre des méthodes utilisable dans JS.

### Explications / Descriptions

Dans cet exercice nous avons créer des variable qui s’affichent et se modifie pour que l’on puissent tester plusieurs méthodes unique a JS.

### Extrait de code

Voici le code :

//clear la console

console.clear();

//créer a et b

let a;

let b;

//affiche a

console.log(a);

//donne la valeur 15 à a

a = 15;

//affiche la valeur de a

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

//donne la valeur 9 à b

b = 9;

//affiche la valeur de bs

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

//affiche le resultat de a + b

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

//affiche le resultat de a - b

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

//affiche le resultat de a \* b

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

//affiche le resultat de a / b

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

//donne la valeur "Bonjour" à a

a = "Bonjour";

//donne la valeur "les amis" à b

b = "les amis";

//affiche a et b en les concatenant avec "concat()"

console.log(a.concat(" ", b));

//affiche a et b en les concatenant en utilisant un littéral

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

//donne la valeur true à a

a = true;

//donne la valeur false à b

b = false;

//affiche le resultat de a && b (AND)

console.log(`${a} AND ${b} = ` + (a && b));

//affiche le resultat de a || b (OR)

console.log(`${a} OR ${b} = ` + (a || b));

//donne la date d'aujourd'hui à a

a = new Date(Date.now());

//donne la date d'aujourd'hui -61 jours à b

b = new Date();

b.setDate(a.getDate() - 61);

//affiche la date et l'heure de a et b

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

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

//affiche la date de a et b

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

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

//affiche l'heure de a et b

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

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

//donne la valeur PI à a et l'affiche avec son type

a = Math.PI;

console.log(a + " is a " +typeof a);

//donne la valeur "Bonjour" à b et l'affiche avec son type

b = "Bonjour";

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

//donne la valeur true à c et l'affiche avec son type

let c = true;

console.log(c + " is a " +typeof c);

//donne la date de aujourd'hui à d et l'affiche avec son type

let d = new Date(Date.now());

console.log(d + " is a " +typeof d);

//donne aucune valeur à e et l'affiche avec son type

let e;

console.log(e + " is a " +typeof e);

### Capture

Voici le résultat :

☁️ "Running fiddle"

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"

"a 16/05/2023 09:55:31"

"b 16/03/2023 09:55:31"

"a 16/05/2023"

"b 16/03/2023"

"a 09:55:31"

"b 09:55:31"

"number"

"string"

"boolean"

"object"

"undefined"

## Exercice 6 (La structure de contrôle switch)

### Objectif de l’exercice

Dans cet exercice nous créons une application qui nous donne la date du jour ce qui nous permet de découvrir comment fonctionne un switch.

### Explications / Descriptions

La fonction afficherJourSemaine utilise un switch qui pour chaque case attribut a la variable day un jour et ensuite affiche la variable day. La fonction afficherJourSemaine2 utilise la méthode getDay() pour afficher le jour, le résultat sera le même que pour la fonction afficherJourSemaine.

### Extrait de code

JS :

function afficherJourSemaine() {

switch (new Date().getDay()) {

case 0:

day = "Sunday";

break;

case 1:

day = "Monday";

break;

case 2:

day = "Tuesday";

break;

case 3:

day = "Wednesday";

break;

case 4:

day = "Thursday";

break;

case 5:

day = "Friday";

break;

case 6:

day = "Saturday";

}

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

}

function afficherJourSemaine2() {

const jours = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];

let x = new Date().getDay();

document.getElementById("info2").innerHTML = jours[x];

}

### Capture

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

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

## Exercice 7 (Les boucles)

### Objectif de l’exercice

Dans cet exercice nous allons découvrir comment fonctionnent les boucle, nous allons tester 3 boucles différentes (for, while, do while).

### Explications / Descriptions

Les fonction utilisent chacune des boucle qui s’itèrent et qui affiche la valeur de i pour montrer comment fonctionnent les différentes boucles.

### Extrait de code

JS :

function initCtrl() {

document.getElementById("for").addEventListener("click", testerFor);

document.getElementById("while").addEventListener("click", testerWhile);

document.getElementById("do").addEventListener("click", testerDoWhile);

}

function testerFor() {

let s = "";

console.log("aaa");

document.getElementById("info").innerHTML =

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

for (let i = 0; i < 5; i++) {

s += "i = " + i + "<br />";

}

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

}

function testerWhile() {

let i = 0;

let s = "";

document.getElementById("info").innerHTML = "while (i < 5) {...}" + "<br />";

while (i < 5) {

s += "i = " + i + "<br />";

i++;

}

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

}

function testerDoWhile() {

let i = 0;

let s = "";

document.getElementById("info").innerHTML =

"do {...} while (i < 5)" + "<br />";

do {

s += "i = " + i + "<br />";

i++;

} while (i < 5);

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

}

### Capture

Une image contenant texte, capture d’écran

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

## Exercice 8 (Tableau d’objets JSON)

### Objectif de l’exercice

Dans cet exercice nous allons voir comment fonctionne un tableau JSON et réutiliser une boucle vue dans l’exercice précèdent.

### Explications / Descriptions

Nous utilisons dans la fonction parcourirUnTableauJSON une boucle for qui parcourt tout un tableau JSON et qui affiche son contenu a chaque cellule.

### Extrait de code

JS :

let s = "";

const json = {

personnes: [

{ prenom: "John", nom: "Doe", age: 44 },

{ prenom: "Anna", nom: "Smith", age: 32 },

{ prenom: "Peter", nom: "Jones", age: 29 },

],

};

function parcourirUnTableauJSON() {

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

s +=i+". "

let personne = json.personnes[i];

for (let f in personne) {

s+= personne[f] +" ";

}

s+= "<br />"

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

}

};

### Capture

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

Description générée automatiquement

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

Description générée automatiquement

## Exercice 9 (Démo sur les objets en JavaScript)

### Objectif de l’exercice

Cet exercice a pour but de nous montrer comment créer un objet avec différentes méthode, le projet et les méthodes sont déjà fournis donc il suffit de regarder et comprendre le code.

### Explications / Descriptions

Toutes ces méthodes nous permettent de créer des objets mais de différentes façon (JSON, object, fonction, classPrototype, class).

### Extrait de code

JS :

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* JSON \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function creerDesObjetsJSON() {

console.log("---------------JSON-------------------------");

// créer un objet en JSON

let p1 = {

prenom: "Jules",

nom: "Tartampion",

age: 25,

};

console.log(p1);

// ajouter deux méthodes

p1.toString = function () {

return this.prenom + " " + this.nom;

};

p1["toStringAge"] = function () {

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

};

console.log(p1);

// let txt = p1.toString() + "<br>" + p1.toStringAge();

let txt = p1 + "<br>" + p1.toStringAge();

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

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Object \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function creerDesObjetsObject() {

console.log("-----------------Object----------------------");

// créer un objet avec la fonction Object

const p1 = new Object();

p1.prenom = "Juliette";

p1.nom = "Tartampion";

p1.age = 23;

console.log(p1);

// ajouter deux méthodes

p1.toString = function () {

return this.prenom + " " + this.nom;

};

p1["toStringAge"] = function () {

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

};

console.log(p1);

// var txt = p1.toString() + "<br>" + p1.toStringAge();

let txt = p1 + "<br>" + p1.toStringAge();

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

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Objet fonction \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function Personne(prenom, nom, age) {

this.prenom = prenom;

this.nom = nom;

this.age = age;

this.toString = function () {

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

};

}

function creerDesObjetsFonction() {

console.log("-----------------objet fonction------------------");

let p1 = new Personne("JuJu", "Tartampion", 40);

console.log(p1);

let p2 = new Personne("Julie", "Tartampion", 32);

console.log(p2);

let txt = p1 + "<br>" + p2;

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

}

// \*\*\*\*\*\*\*\*\*\*\*\*Function pseudo-classe => fonction-prototype \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function Professeur(prenom, nom, age) {

this.prenom = prenom;

this.nom = nom;

this.age = age;

}

Professeur.prototype.toString = function () {

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

};

function creerDesObjetsClassePrototype() {

console.log("-----------------prototype----------------------");

let p1 = new Professeur("De Ju", "Tartampion", 25);

console.log(p1);

let p2 = new Professeur("De Jiette", "Tartampion", 24);

console.log(p2);

let txt = p1 + "<br>" + p2;

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

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Class \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class Eleve {

constructor(prenom, nom, age) {

this.prenom = prenom;

this.nom = nom;

this.age = age;

}

toString() {

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

}

}

function creerDesObjetsClasse() {

console.log("-----------------Class----------------------");

let p1 = new Eleve("Julien", "Tartampion", 18);

console.log(p1);

let p2 = new Eleve("Julia", "Tartampion", 22);

console.log(p2);

let txt = p1 + "<br>" + p2;

}

### Capture

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

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

## Exercice 10 (Programmation orientée "objets")

### Objectif de l’exercice

Le but de cet exercice et de permette de gérer simplement les informations d’une liste de personnes, avec les méthodes pour ajouter, supprimer et nettoyer (reset).

### Explications / Descriptions

Dans cet exercice nous créons des personnes puis nous lisons les personnes, leur infos et finalement on les afficher.

### Extrait de code

JS :

Personne :

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 :

/\*

\* 1. DOM PRET : DEMARRAGE DE L'APPLICATION

\*/

document.onreadystatechange = function () {

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

\_afficherPersonnes();

}

};

/\*

\* 2. METHODES PRIVEES DE LECTURE/ECRITURE DANS LA VUE

\*/

// affiche la liste des données au bas de la vue (avec du HTML généré)

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;

}

// affiche les infos d'une personne dans le formulaire

function \_afficherInfosPersonne(p) {

document.getElementById("prenom").value = p.prenom;

document.getElementById("nom").value = p.nom;

document.getElementById("age").value = p.age;

}

// lit le contenu des masques de saisie pour en faire une personne

function \_lireInfosPersonne() {

let p = new Personne(

document.getElementById("prenom").value,

document.getElementById("nom").value,

document.getElementById("age").value

);

return p;

}

/\*

\* 3. METHODES PUBLIQUES NECESSAIRES A LA VUE

\*/

// appelée depuis la vue pour afficher les données de la personne sélectionnné

function selectionnerPersonne(i) {

\_afficherInfosPersonne(personnes[i]);

}

// appelée depuis la vue pour ajouter une personne

function ajouter() {

let p = \_lireInfosPersonne();

ajouterPersonne(p);

\_afficherPersonnes();

}

// appelée depuis la vue pour supprimer une personne

function supprimer() {

let p = \_lireInfosPersonne();

supprimerPersonne(p);

\_afficherPersonnes();

}

### Capture

Une image contenant texte, logiciel, Logiciel multimédia, Icône d’ordinateur

Description générée automatiquement

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

Description générée automatiquement

Une image contenant texte, logiciel, capture d’écran, Logiciel multimédia

Description générée automatiquement

## Exercice 11 (Programmation orientée "classe" en Javascript)

### Objectif de l’exercice

Cet exercice nous permet de découvrir les classes dans JS et nous montre comment elles fonctionnent et quelles sont les différence avec les classes de Java.

### Explications / Descriptions

Dans cet exercice nous allons écrire des nouvelle personne avec ce que l’utilisateur a mis et supprimer des données qui sont enregistré.

### Extrait de code

JS :

Personne :

class Personne {

constructor(prenom, nom, age) {

this.prenom = prenom;

this.nom = nom;

this.age = age;

}

toString() {

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

}

}

IndexCtrl :

/\*

 \* 1. DOM PRET : DEMARRAGE DE L'APPLICATION

 \*/

document.onreadystatechange = function () {

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

window.ctrl = new Ctrl();

}

};

class Ctrl {

constructor() {

this.wrk = new Worker();

this.afficherPersonnes();

}

/\*

\* 2. METHODES PRIVEES DE LECTURE/ECRITURE DANS LA VUE

\*/

// affiche la liste des données au bas de la vue (avec du HTML généré)

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;

}

// affiche les infos d'une personne dans le formulaire

afficherInfosPersonne(p) {

document.getElementById("prenom").value = p.prenom;

document.getElementById("nom").value = p.nom;

document.getElementById("age").value = p.age;

}

// lit le contenu des masques de saisie pour en faire une personne

lireInfosPersonne() {

let p = new Personne(

document.getElementById("prenom").value,

document.getElementById("nom").value,

document.getElementById("age").value

);

return p;

}

/\*

\* 3. METHODES PUBLIQUES NECESSAIRES A LA VUE

\*/

// appelée depuis la vue pour afficher les données de la personne sélectionnné

selectionnerPersonne(i) {

this.afficherInfosPersonne(this.wrk.personnes[i]);

}

// appelée depuis la vue pour ajouter une personne

ajouter() {

let p = this.lireInfosPersonne();

this.wrk.ajouterPersonne(p);

this.afficherPersonnes();

}

// appelée depuis la vue pour supprimer une personne

supprimer() {

let p = this.lireInfosPersonne();

this.wrk.supprimerPersonne(p);

this.afficherPersonnes();

}

}

### Capture

Une image contenant texte, logiciel, Logiciel multimédia, capture d’écran

Description générée automatiquement

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

Description générée automatiquement

## Exercice 12 (Fonctions et IIFE en JavaScript)

### Objectif de l’exercice

Dans cet exercice nous devons voir 4 méthode pour écrire une fonction, Voici les 4 différentes façon

// 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

(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 (Les cookies)

### Objectif de l’exercice

Le but de l’exercice 13 et de découvrir comment fonctionnent les cookies et de faire un formulaire que l’on va sauvegarder dans un cookie.

### Explications / Descriptions

Dans cet exercice nous récupérons les infos que l’utilisateur a entré et nous les mettons en cookies qui seront enregistre pendant quelques minutes.

### Extrait de code

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".

let nom = "";

nom = document.getElementById("nom").value;

let prenom = "";

prenom = document.getElementById("prenom").value;

// Créer un objet json "personneJson" contenant le prénom et le "nom".

let personneJson = {

prenom: prenom,

nom: nom,

};

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

let personneJsonString = JSON.stringify(personneJson);

// Appeler la méthode "setCookie" en passant en paramètre l'objet "personneJson" converti et en précisant le délai de 3 minutes.

setCookie(personneJsonString, 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 = addMinutes(new Date(), 3);

console.log(dateExpiration);

// Créer le cookie.

document.cookie =

"cookie\_exercice\_13=" +

contenu +

"; expires=" +

dateExpiration.toUTCString() +

"; path=/";

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

// Convertir la chaîne de caractères en objet Json.

let personne = JSON.parse(cookieS);

// Charger les deux champs avec la valeur du prénom et du du nom contenu dans l'objet json.

document.getElementById("prenom").value = personne.prenom;

document.getElementById("nom").value = personne.nom;

console.log("Data retrieved from cookie");

}

}

function addMinutes(date, minutes) {

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

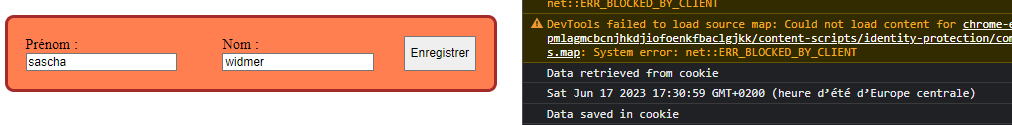
return date;

}

### Capture

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

Description générée automatiquement



## Exercice 14 (Bases de jQuery)

### Objectif de l’exercice

Le but de cet exercice et de découvrir les bases de jQuery grâce a un site web (<https://www.jcsinfo.ch/demo/jquery/>) qui est une sorte de présentation qui nous montre comment fonctionne jQuery et qui nous permet de tester certaines fonctions.

## Exercice 15 (Première utilisation de jQuery)

### Objectif de l’exercice

Le but de cet exercice est de nous montrer comment utiliser du jQuery, nous allons faire une vue qui va changer suivant ce que l’on sélectionne dans une liste, nous allons effectuer cet exercice en JS et en jQuery pour constater les différences.

### Explications / Descriptions

Dans cet exercice nous créons une petite interface avec une liste qui nous permet de changer la couleur suivant laquelle on sélectionne.

### Extrait de code

JS :  
IndexCtrlJS :

/\*

\* 1. DOM PRET : DEMARRAGE DE L'APPLICATION

\*/

document.onreadystatechange = function () {

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

ctrl = new Ctrl();

let couleur = document.getElementById("couleurs").value;

ctrl.changerCouleur(couleur);

}

};

class Ctrl {

constructor() {}

changerCouleur(couleur) {

document.getElementById("container").style.backgroundColor = couleur;

if (couleur == "red" || couleur == "blue" || couleur == "green") {

document.getElementById("titre").style.color = "white";

} else {

document.getElementById("titre").style.color = "black";

}

}

}

IndexCtrlJQ :

/\*

\* 1. DOM PRET : DEMARRAGE DE L'APPLICATION AVEC CLASSE CTRL

\*/

$(document).ready(function () {

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

changerCouleur(couleur);

});

function changerCouleur(couleur) {

$("#container").css({ "background-color": couleur });

if (couleur == "red" || couleur == "blue" || couleur == "green") {

$("#titre").css({ color: "white" });

} else {

$("#titre").css({ color: "black" });

}

}

### Capture

#### JS

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

Description générée automatiquement

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

Description générée automatiquement

#### JQ

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

Description générée automatiquement

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

Description générée automatiquement

## Exercice 16 (Utilisation de jQuery)

### Objectif de l’exercice

Le but de cet exercice et de nous familiariser avec jQuery nous devons faire différentes méthodes en jQuery ce qui nous montre les différentes possibilités avec jQuery,

### Extrait de code

JS :

$(document).ready(function () {

ctrl = new Ctrl();

$("#btnShow").on("click", function () {

ctrl.afficher();

});

$("#btnHide").on("click", function () {

ctrl.cacher();

});

$("#btnToggle").on("click", function () {

ctrl.afficherCacher();

});

$("#btnPlier").on("click", function () {

ctrl.plier();

});

$("#btnDeplier").on("click", function () {

ctrl.deplier();

});

$("#btnFonduShow").on("click", function () {

ctrl.fonduAfficher();

});

$("#btnFonduHide").on("click", function () {

ctrl.fonduCacher();

});

$("#btnSetClass").on("click", function () {

ctrl.setClass();

});

$("#btnRemoveClass").on("click", function () {

ctrl.removeClass();

});

$("#btnToggleClass").on("click", function () {

ctrl.toggleClass();

});

$("#btnPairesClass").on("click", function () {

ctrl.pairesClass();

});

$("#btnInsertText").on("click", function () {

ctrl.insererTexte();

});

});

class Ctrl {

constructor() { }

afficher() {

$("#div\_1").show();

}

cacher() {

$("#div\_1").hide();

}

afficherCacher() {

$("#div\_1").toggle();

}

plier() {

$("#div\_2").slideUp();

}

deplier() {

$("#div\_2").slideDown();

}

fonduAfficher() {

$("#div\_3").fadeIn();

}

fonduCacher() {

$("#div\_3").fadeOut();

}

setClass() {

$("div>ol>li:first-of-type").addClass("boldBlueText");

}

removeClass() {

$("div>ol>li:first-of-type").removeClass("boldBlueText");

}

toggleClass() {

$("div>ol>li:last-of-type").toggleClass("boldBlueText");

}

pairesClass() {

$("div>ol>li:nth-child(odd)").toggleClass("underlineClass");

}

insererTexte() {

$("#textes").append($("#textToInsert").value());

if ($("#textBold") == true) {

}

}

}

### Capture

Une image contenant texte, ligne, Police, nombre

Description générée automatiquement

Dans cet exercice nous découvrons différentes méthodes jQuery.

* Comment ajouter un écouteur via jQuery.
* Sélectionner des éléments du DOM via jQuery.
* Ajouter et supprimer des classes CSS à des éléments HTML via jQuery.
* Récupérer des valeurs d'un formulaire via jQuery.
* Ajouter des éléments au DOM via jQuery.
* Extraire la valeur d'un attribut HTML via jQuery.

## Exercice 17 (WebServices - découverte)

### Objectif de l’exercice

Le but de cet exercice est de s’informer sur un webServices et de le présenter ensuite au reste de la classe. Voici dans les points suivant des notions théorique sur tous les webServices.

### REST

Voici l’architecture de REST :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

RESTful :

* Chaque méthode doit être publique -> accessible par le client​
* Toutes les méthodes respectent l’architecture REST

### SOAP

Message SOAP :

* Messages XML qui encapsulent les données

Bindings SOAP :

* Comment les message SOAP doit être encode pour le transport

Fonctionnement

1. Le client envoie une requête SOAP au serveur (HTTP)
2. Le serveur reçoit la requête et extrait les données
3. Le serveur traite la requête selon l’opération
4. La réponse SOAP est encapsulée et renvoyée
5. Le client reçoit la réponse et extrait/traite les données

WSDL :

* Schema XML (vérifie la syntaxe)
* Spécifie ce qu’un message doit contenir

### GET

Sert à récupérer des informations (JSON ou XML)

Voici un exemple de lien pour récupérer des données :

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

Description générée automatiquement

### POST

Un POST permet d’ajouter, modifier, supprimer un objet dans une API, de définir un token pour avoir les droits.

### PUT

Un PUT permet d’insérer des données dans une API.

Voici un exemple d’utilisation :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

### DELETE

Le DELETE permet de supprimer des données. Pour faire un DELETE il faut mettre en paramètre l’url de l’objet que l’on souhaite supprimer.

Voici la commande en JS :

var settings = {

"url": "https://reqres.in/api/users/1",

"method": "DELETE",

"timeout": 0,

};

## Exercice 18 (Application HTML5 appelant un WS)

### Objectif de l’exercice

Cet exercice nous apprend comment fonctionne une requête ajax et comment il faut l’utiliser.

### Explications / Descriptions

Pour cet exercice nous utilisons des requêtes ajax en faisant des POST et des GET les méthodes vont transformer ce que l’utilisateur entre (en °C) en fahrenheit.

### Extrait de code

JS :

IndexCtrl :

// Attend la fin du chargement de la page

$().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 = "https://widmers.emf-informatique.ch/307/Exercices/Exercice\_18/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

});

}

}

PHP :

convert-temp\_g\_json :

<?php

// convert\_temp\_g\_json.php

// WebService Conversion de température GET et réponse JSON

//

header('Content-Type: application/json');

$temp = intval($\_GET['Temperature'], 10);

$from = $\_GET['FromUnit'];

$to = $\_GET['ToUnit'];

if($from === 'C'){

$tempf = $temp\*9/5+32;

} else if($from === 'F') {

$tempf = ($temp-32)\*5/9;

}

$result = array('temperature' => $tempf);

echo json\_encode($result); // fonction php pour transformer des données au format JSON

?>

convert-temp\_p\_xml :

<?PHP

// convert\_temp\_p\_xml.php

// WebService Conversion de température POST et réponse XML

// Naël Telfser

//

header('Content-Type: application/xml');

// header('access-control-origine-allow:\*')

$temp = intval($\_POST['Temperature'], 10);

$from = $\_POST['FromUnit'];

$to = $\_POST['ToUnit'];

if($from === 'C'){

$tempf = $temp\*9/5+32;

} else if($from === 'F') {

$tempf = ($temp-32)\*5/9;

}

$result = "<temperature>" . $tempf . "</temperature>";

echo $result;

?>

### Capture

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

Description générée automatiquement

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

Description générée automatiquement

## Exercice 20 (Application de login sur un serveur PHP)

### Objectif de l’exercice

Cet exercice a pour but de nous familiariser un peu plus avec ajax et de charger des vue. On va aussi utiliser cet exercice pour la base de notre projet.

### Explications / Descriptions

Cette application est faite single page application, donc il y a qu’une seule page HTML qui va chercher d’autre pages lorsqu’il est nécessaire. Nous allons utiliser cet exercice comme base pour notre projet final.

### Extrait de code

JS :

httpService :

centraliserErreurHttp(httpErrorCallbackFn) {

$.ajaxSetup({

error: function (xhr, exception) {

let msg;

if (xhr.status === 0) {

msg = "Pas d'accès à la ressource serveur demandée !";

} else if (xhr.status === 404) {

msg = "Page demandée non trouvée [404] !";

} else if (xhr.status === 500) {

msg = "Erreur interne sur le serveur [500] !";

} else if (exception === "parsererror") {

msg = "Erreur de parcours dans le JSON !";

} else if (exception === "timeout") {

msg = "Erreur de délai dépassé [Time out] !";

} else if (exception === "abort") {

msg = "Requête Ajax stoppée !";

} else {

msg = "Erreur inconnue : \n" + xhr.responseText;

}

httpErrorCallbackFn(msg);

},

});

}

login(identifiant, successCallback) {

// Uploade votre propre fichier PHP et adaptez l'URL ci-dessous.

let url = "https://widmers.emf-informatique.ch/307/Exercices/Exercice\_20/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

});

}

}

IndexCtrl :

$().ready(function () {

// service et indexCtrl sont des variables globales qui doivent être accessible depuis partout => pas de mot clé devant ou window.xxx

http = new HttpService();

indexCtrl = new IndexCtrl(); // ctrl principal

http.centraliserErreurHttp(indexCtrl.afficherErreurHttp);

});

class IndexCtrl {

constructor() {

this.vue = new VueService();

this.loadLogin();

}

afficherErreurHttp(msg) {

alert(msg);

}

// avec arrow function

loadLogin() {

this.vue.chargerVue("login", () => new LoginCtrl());

}

// avec function classique

loadAccueil(langue) {

this.vue.chargerVue("accueil", function() {

new AccueilCtrl(langue);

});

}

loadCompte() {

this.vue.chargerVue("compte", () => new CompteCtrl());

}

}

vueService :

class VueService {

constructor() { }

chargerVue(vue, callback) {

// charger la vue demandee

$("#view").load("views/" + vue + ".html", function () {

// si une fonction de callback est spécifiée, on l'appelle ici

if (typeof callback !== "undefined") {

callback();

}

});

}

}

### Capture

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

Description générée automatiquement

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

Description générée automatiquement

# Projet

## Analyse

### Maquettes

#### Accueil

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

Description générée automatiquement

#### Infos artiste

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

Description générée automatiquement

#### Info perso

Au final cette partie du site a été remplacé par une autre car je n’arrivais pas à avoir l’autorisation nécessaire.

Une image contenant texte, capture d’écran, Rectangle, Parallèle

Description générée automatiquement

### Use case

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

Description générée automatiquement

## Conception

### Diagramme de navigation

Une image contenant diagramme, ligne, Plan, texte

Description générée automatiquement

## Implémentation

Voici les requêtes utilisé pour ce projet :

class HttpService {

constructor() { }

/\*

\*\* $.ajaxSetup permet de définir une fois un élément sans le refaire par la suite. Ici cela se fait l'error

\*/

centraliserErreurHttp(httpErrorCallbackFn) {

$.ajaxSetup({

error: function (xhr, exception) {

let msg;

if (xhr.status === 0) {

msg = "Pas d'accès à la ressource serveur demandée !";

} else if (xhr.status === 404) {

msg = "Page demandée non trouvée [404] !";

} else if (xhr.status === 500) {

msg = "Erreur interne sur le serveur [500] !";

} else if (exception === "parsererror") {

msg = "Erreur de parcours dans le JSON !";

} else if (exception === "timeout") {

msg = "Erreur de délai dépassé [Time out] !";

} else if (exception === "abort") {

msg = "Requête Ajax stoppée !";

} else {

msg = "Erreur inconnue : \n" + xhr.responseText;

}

httpErrorCallbackFn(msg);

},

});

}

login(successCallBack) {

let clientID = document.getElementById("clientID").value;

let clientSecret = document.getElementById("clientSecret").value;

$.ajax({

url: "https://accounts.spotify.com/api/token",

type: "POST",

headers: {

"Content-Type": "application/x-www-form-urlencoded",

},

data: {

grant\_type: "client\_credentials",

client\_id: clientID,

client\_secret: clientSecret,

},

success: successCallBack,

error: function (error) {

// Handle the error response here

console.log(error);

},

});

}

getURI(successCallBack) {

$.ajax({

url: `https://api.spotify.com/v1/search?type=artist`,

type: "GET",

data: {

q: document.getElementById("artisteRecherche").value,

type: "artist",

},

headers: {

Authorization: "Bearer " + token,

},

// Gérer ici la réponse en cas de succès

success: successCallBack,

// Gérer ici la réponse en cas d'erreur

error: function (error) {

console.log(error);

},

});

}

getImgArtiste(artist, successCallBack) {

$.ajax({

url: "https://api.spotify.com/v1/artists/" + artist,

type: "GET",

headers: {

Authorization: "Bearer " + token,

},

// Gérer ici la réponse en cas de succès

success: successCallBack,

// Gérer ici la réponse en cas d'erreur

error: function (error) {

console.log(error);

},

});

}

getNomArtiste(artist, successCallBack) {

$.ajax({

url: "https://api.spotify.com/v1/artists/" + artist,

type: "GET",

headers: {

Authorization: "Bearer " + token,

},

// Gérer ici la réponse en cas de succès

success: successCallBack,

// Gérer ici la réponse en cas d'erreur

error: function (error) {

console.log(error);

},

});

}

getFollowerArtiste(artist, successCallBack) {

$.ajax({

url: "https://api.spotify.com/v1/artists/" + artist,

type: "GET",

headers: {

Authorization: "Bearer " + token,

},

// Gérer ici la réponse en cas de succès

success: successCallBack,

// Gérer ici la réponse en cas d'erreur

error: function (error) {

console.log(error);

},

});

}

getTracksArtiste(artist, successCallBack) {

$.ajax({

url: "https://api.spotify.com/v1/artists/" + artist + "/albums",

type: "GET",

headers: {

Authorization: "Bearer " + token,

},

data: {

include\_groups: "album,single",

},

// Gérer ici la réponse en cas de succès

success: successCallBack,

// Gérer ici la réponse en cas d'erreur

error: function (error) {

console.log(error);

},

});

}

getGenresArtiste(artist, successCallBack) {

$.ajax({

url: "https://api.spotify.com/v1/artists/" + artist,

type: "GET",

headers: {

Authorization: "Bearer " + token,

},

// Gérer ici la réponse en cas de succès

success: successCallBack,

// Gérer ici la réponse en cas d'erreur

error: function (error) {

console.log(error);

},

});

}

getRecommendation(seed\_artists, seed\_genres, seed\_tracks, successCallBack) {

$.ajax({

url: "https://api.spotify.com/v1/recommendations",

type: "GET",

headers: {

Authorization: "Bearer " + token,

},

data: {

limit: 5,

market: "CH",

seed\_artists: seed\_artists,

seed\_genres: seed\_genres,

seed\_tracks: seed\_tracks,

},

// Gérer ici la réponse en cas de succès

success: successCallBack,

// Gérer ici la réponse en cas d'erreur

error: function (error) {

console.log(error);

},

});

}

getTrackID(trackName, successCallBack) {

$.ajax({

url: "https://api.spotify.com/v1/search",

headers: {

"Authorization": "Bearer " + token

},

data: {

q: trackName,

type: "track",

limit: 1

},

// Gérer ici la réponse en cas de succès

success: successCallBack,

// Gérer ici la réponse en cas d'erreur

error: function (error) {

console.log("Erreur lors de la requête :", error);

}

});

}

}

## Hébergement et fonctionnement

### Login

Pour se login il faut se créer un compte Spotify développeur (<https://developer.spotify.com/>) puis créer une "app", ensuite aller dans les settings et ensuite entrer son client ID et Secret client dans le login.

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

Description générée automatiquement**



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

Description générée automatiquement

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

Description générée automatiquement

### Info sur les artistes

Pour obtenir des infos sur un artiste il suffit de mettre le nom de l’artiste dans le texte field. Parfois selon le nom de l’artiste il se peut que l’artiste désiré ne soit pas trouvé à cause de la ressemblance de nom ou car un autre artiste contient les même lettres dans son nom.

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

Une image contenant texte, capture d’écran, Visage humain, logiciel

Description générée automatiquement

Une image contenant texte, Appareils électroniques, capture d’écran, Site web

Description générée automatiquement

### Recommandations

Pour les recommandations il faut entrer en premier le nom d’un titre que l’on veut ensuite le nom d’un artiste et finalement choisir un genre de musique par ceux disponible dans la liste déroulante puis lorsque l’on appuie sur "search" 5 titre apparaissent qui sont généré par une IA de Spotify avec l’aide des informations donné.

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

Description générée automatiquement

Une image contenant texte, Appareils électroniques, capture d’écran, affichage

Description générée automatiquement

# Conclusion

J’ai beaucoup aimé ce module car j’aime faire du développement web et aussi car j’ai appris beaucoup de choses sur javascript que je ne connaissais pas très bien et je ne savais pas comment l’utiliser, mais grâce à ce module je sais maintenant comme l’utiliser. J’ai trouvé qu’il y avait un peu trop d’exercice ce qui nous a pris un peu de temps sur le projet, mais ils étaient intéressants quand même car nous apprenions quelque chose de nouveau à chaque exercice.