307 - Réaliser des pages Web interactives

Rapport Personnel

Esther Pham

Date de création : 15.05.2023

Version 1 du 15.06.2023

|  |  |  |
| --- | --- | --- |
|  |  | Du 15.10.2021 au 13.06.2023 |

Table des matières

[1 Introduction 6](#_Toc137760656)

[1.1 Objectif 6](#_Toc137760657)

[2 Exercice 1 7](#_Toc137760658)

[2.1 EventListener 7](#_Toc137760659)

[2.2 Documentation 8](#_Toc137760660)

[2.2.1 Exécution du javascript 8](#_Toc137760661)

[2.2.2 DOM 8](#_Toc137760662)

[3 Exercice 2 9](#_Toc137760663)

[3.1 Code HTML 9](#_Toc137760664)

[3.2 Code CSS 9](#_Toc137760665)

[3.3 Code javascript 10](#_Toc137760666)

[3.4 Documentation 11](#_Toc137760667)

[3.4.1 Questions HTML 11](#_Toc137760668)

[3.4.2 Questions javascript 11](#_Toc137760669)

[3.4.3 Questions générales 11](#_Toc137760670)

[4 Exercice 3 13](#_Toc137760671)

[4.1 Code php 13](#_Toc137760672)

[4.2 Questions 13](#_Toc137760673)

[5 Exercice 4 15](#_Toc137760674)

[5.1 Vue de JSFiddle 15](#_Toc137760675)

[5.2 Questions 15](#_Toc137760676)

[6 Exercice 5 16](#_Toc137760677)

[6.1 Code JavaScript 16](#_Toc137760678)

[6.2 Questions 17](#_Toc137760679)

[7 Exercice 6 19](#_Toc137760680)

[7.1 Code JavaScript 19](#_Toc137760681)

[7.2 Questions 20](#_Toc137760682)

[8 Exercice 7 21](#_Toc137760683)

[8.1 Boucle For 21](#_Toc137760684)

[8.2 Boucle While 21](#_Toc137760685)

[8.3 Boucle DoWhile 21](#_Toc137760686)

[9 Exercice 8 22](#_Toc137760687)

[9.1 Code JavaScript (boucle sur des éléments json) 22](#_Toc137760688)

[10 Exercice 9 23](#_Toc137760689)

[10.1 Code JavaScript 23](#_Toc137760690)

[11 Exercice 10 24](#_Toc137760691)

[11.1 Code JavaScript 24](#_Toc137760692)

[11.1.1 Index.js 24](#_Toc137760693)

[11.1.2 Workers.js 25](#_Toc137760694)

[11.1.3 Personne.js 26](#_Toc137760695)

[12 Exercice 11 27](#_Toc137760696)

[12.1 But 27](#_Toc137760697)

[12.2 Class Ctrl (index.js) 27](#_Toc137760698)

[12.3 Class personne(personne.js) 28](#_Toc137760699)

[12.4 Class Worker(worker.js) 29](#_Toc137760700)

[13 Exercice 12 31](#_Toc137760701)

[13.1 But 31](#_Toc137760702)

[13.2 Déclaration d’une fonction normale 31](#_Toc137760703)

[13.3 Déclaration d’une fonction anonyme 31](#_Toc137760704)

[13.4 Déclaration d’une fonction flèche 31](#_Toc137760705)

[14 Exercice 13 32](#_Toc137760706)

[14.1 Questions 32](#_Toc137760707)

[15 Exercice 14 33](#_Toc137760708)

[15.1 JQuery 33](#_Toc137760709)

[16 Exercice 15 (JQuery) 34](#_Toc137760710)

[16.1 Récupérer la valeur d'un champ 34](#_Toc137760711)

[16.2 Propriété css avec jquery 34](#_Toc137760712)

[16.3 Animation avec jquery 34](#_Toc137760713)

[17 Exercice 16 35](#_Toc137760714)

[17.1 Ajouter un écouteur 35](#_Toc137760715)

[17.2 Sélectionner des éléments du DOM 35](#_Toc137760716)

[17.3 Ajouter et supprimer des classes CSS 35](#_Toc137760717)

[17.4 Récupérer des valeurs d'un formulaire 35](#_Toc137760718)

[17.5 Ajouter des éléments au DOM 35](#_Toc137760719)

[17.6 Extraire la valeur d'un attribut HTML 35](#_Toc137760720)

[18 Exercice 18 36](#_Toc137760721)

[18.1 Appel GET / POST en Ajax 36](#_Toc137760722)

[18.1.1 GET 36](#_Toc137760723)

[18.1.2 POST 36](#_Toc137760724)

[18.1.3 Donner des paramètres 36](#_Toc137760725)

[18.1.4 Méthode de retour en cas de succès 36](#_Toc137760726)

[18.1.5 Méthode de retour en cas d'erreur 36](#_Toc137760727)

[18.2 Appel GET / POST avec Postman 36](#_Toc137760728)

[18.2.1 GET 36](#_Toc137760729)

[18.2.2 POST 37](#_Toc137760730)

[19 Exercice 20 38](#_Toc137760731)

[19.1 SPA 38](#_Toc137760732)

[19.2 Charger page HTML dans une autre page HTML 38](#_Toc137760733)

[19.3 ajax $.ajaxSetup() 38](#_Toc137760734)

[20 WebServices (ex 17) 40](#_Toc137760735)

[20.1 REST 40](#_Toc137760736)

[20.2 SOAP 40](#_Toc137760737)

[20.2.1 Fonctionnement 40](#_Toc137760738)

[20.2.2 WSDL 41](#_Toc137760739)

[20.3 Commandes 41](#_Toc137760740)

[20.3.1 GET 41](#_Toc137760741)

[20.3.1.1 Codes http 42](#_Toc137760742)

[20.3.2 POST 42](#_Toc137760743)

[20.3.3 PUT 42](#_Toc137760744)

[20.3.4 DELETE 43](#_Toc137760745)

[20.3.5 Paramètres 43](#_Toc137760746)

[20.3.6 Autorisations 44](#_Toc137760747)

[21 Projet 45](#_Toc137760748)

[21.1 API 45](#_Toc137760749)

[21.1.1 Tests postman 45](#_Toc137760750)

[21.2 Analyse 49](#_Toc137760751)

[21.2.1 Diagramme use cases 49](#_Toc137760752)

[21.2.2 Maquette 49](#_Toc137760753)

[21.3 Conception 51](#_Toc137760754)

[21.4 Test 52](#_Toc137760755)

[22 Conclusion 53](#_Toc137760756)

# Introduction

## Objectif

* Rappel sur les bases des techniques du Web côté client (HTML, CSS et autres)
* Tester un environnement de travail efficace pour le développement web
* Comprendre les bases du développement web côté client (intelligence locale) en étudiant un langage de script et une ou l’autre bibliothèque adaptée
* Comprendre les bases de la consommation de services web existants et tester en pratique par des requêtes HTTP
* Comprendre les bases d’un découpage à couches d’une application web côté client et tester en pratique
* Mettre en forme des données JSON ou XML provenant d’un serveur.
* Comprendre la problématique de l’adaptation des applications Web interactives à des écrans de diverses natures et mettre en pratique
* Créer et améliorer un projet global « accompagné », basé sur les techniques vues pendant le module

# Exercice 1

Pour utiliser javascript dans un fichier html il faut le mettre dans la balise head du fichier :

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <link rel="icon" href="#" />

  <link rel="stylesheet" href="css/main.css" />

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

  <title>Ex01 — Exercice de base</title>

</head>

## EventListener

Voici 2 méthodes permettant d’ajouter des écouteurs, via javascript, sur un bouton. L’action du click permettra d’afficher des éléments en-dessous.

1ère méthode, ajout d’un écouteur « click » via javascript :

Fichier html :

<!-- corps de la page avec enregistrement d'écouteur dans le JS-->

<body onload="initCtrl()">

    <div id="container">

      <p>Cliquez sur le bouton qui contient votre prénom (js) !</p>

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

      <p id="info">&nbsp;</p>

    </div>

  </body>

Fichier javascript :

function initCtrl() {

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

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

}

function testez() {

  document.getElementById("info").innerHTML = "C'est <b>MOI</b> qui a pressé le bouton !";

}

2ème méthode, ajout direct d’un écouteur « click » sur un bouton :

Fichier html :

<!-- corps de la page avec enregistrement d'écouteur dans le HTML de la page-->

<!-- Ajoutez l'écouteur testez() directement sur le bouton-->

<!-- Jouez avec les commentaires pour executer un <body> après l'autre-->

<body>

  <div id="container">

  <p>Cliquez sur le bouton qui contient votre prénom (html) !</p>

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

  <p id="info">&nbsp;</p>

  </div>

</body>

Fichier javascript :

function testez() {

  document.getElementById("info").innerHTML = "C'est <b>MOI</b> qui a pressé le bouton !";

}

## Documentation

### Exécution du javascript

Pour utiliser le code javascript coder, il faut avoir des interactions avec la page, dans notre cas ce sera un click. Cependant il existe d’autres manière d’interagir avec la page web.

### DOM

Le DOM ou Document Object Model est un modèle créé par le navigateur lorsque l’on charge la page Web :

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

Description générée automatiquement

Ce modèle permet à javascript de :

* Modifier tous les éléments HTML de la page
* Modifier tous les attributs HTML de la page
* Modifier tous les styles CSS de la page
* Supprimer des éléments et des attributs HTML existants
* Ajouter de nouveaux éléments et attributs HTML
* Réagir à tous les événements HTML existants dans la page
* Créer de nouveaux événements HTML dans la page

# Exercice 2

## Code HTML

<!DOCTYPE html>

<html>

<!--

   But :    valider le login det le password de connexion

   Auteur : esther pham

   Date :   15.05.2023 / V1.0

   -->

<!-- entête de la page -->

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <link rel="icon" href="#" />

  <link rel="stylesheet" href="css/main.css" />

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

  <title>Ex02 — Exercice avancé</title>

</head>

<body>

  <form class="user-form" id="container">

    <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>

</body>

</html>

## Code CSS

/\*

  But :

  Auteur : esther pham

  Date :   15.05.2023 / V1.0

\*/

body {

  background-color: grey;

}

#container {

  background-color: white;

  width: 90%;

  padding: 1em;

  border: 1px solid black;

  margin: auto;

  font-family: Verdana, Arial, serif;

}

.user-form .field{

  padding: 0.5em;

  background-color: lightgrey;

}

.user-form .field label {

  display: inline-block;

  width: 10em;

}

.user-form .field input {

  display: inline-block;

}

#valider{

  width: 7em;

  border-radius: 10pt;

  background-color: greenyellow;

}

## Code javascript

/\*

  But :  valider les informations misent dans les champs d'informations (nom utilisateur et mot de passe)

  Auteur : Esther Pham

  Date :   15.05.2023 / V1.0

\*/

function validerUtilisateur() {

  if (document.getElementById("username").value == "admin" && document.getElementById("password").value == "emf123") {

    window.alert("Validation OK.");

  } else {

    window.alert("Utilisateur ou mot de passe incorrect !!!");

  }

}

## Documentation

### Questions HTML

* À quoi sert l’attribut « placeholder » ?

Il permet d’indiquer ce que l’on attend dans un champ :

* À quoi sert l’attribut « autofocus » ?

Cela permet de mettre directement le curseur pour écrire à un endroit

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

Oui

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

Ce n’est pas une balise bouton mais un attribut indiquant que c’est un bouton

### Questions javascript

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

function validerUtilisateur() {}

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

En mettant un id à la balise concernée :

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

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

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

Oui, mais ce ne sont pas les mêmes qu’en JAVA

* Où se trouve la console de débogage de Chrome ?

Lorsque l’on inspecte ou que l’on appuie sur F12 un onglet s’affiche dans la page et dans cet onglet, il y a également l’onglet console.

### Questions générales

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

Window.[alert()/confirm()/prompt()] permettent d’afficher différentes types de pop-up :

* + alert() : ouvre une pop-up pour signaler une erreur ou une validation.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* + confirm() : ouvre une pop-up de confirmation
  + prompt() : ouvre une pop-up d’input permettant à l’utilisateur de saisir du texte
* Quelle est la différence entre bouton "button" et un bouton "input" ?

Un bouton « button » est une balise bouton et un bouton « input » est un type.

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

Il suffit d’utiliser la méthode :

document.getElementById(« ID »)

* Comment récupérer le texte contenu dans un champ de formulaire ?

Cette méthode permet de récupérer la valeur se trouvant dans le champ correspondant à l’ID.

document.getElementById(« ID »).value

* Comment écrire sur la console de votre navigateur ? Quelle est l'utilité ?

Grâce à la balise input

# Exercice 3

## Code 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>";

}

}

?>

## Questions

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

Un bouton pour un formulaire, permet d’indiquer à quoi va servir le bouton par rapport au form :

1. Submit : soumet des information au formulaire.
2. Reset : recharge le formulaire à ses données initiales.

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

<?PHP?>

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

// récupération des données transmises dans des variables locales

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

  $password = $\_POST['password'];

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

$concat = .$usernam. .$password.

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

Strtolower()

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

echo "<script>alert('Validation OK');</script>";

echo "<script>alert('Utilisateur ou mot de passe incorrect !!!');</script>";

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

Du html :

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

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

# Exercice 4

## Vue de JSFiddle

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

## Questions

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

JSFiddle est une IDE en ligne qui permet de tester du html, css et javascript sans avoir besoin de créer des fichiers à chaque fois.

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

Il est pratique pour tester les différents éléments que l’on veut utiliser pour une page web, sans avoir besoin de créer des fichiers à chaque fois. Il est plus rapide grâce à celui-ci de tester et de changer, comme un brouillon.

# Exercice 5

## Code JavaScript

// Effacer le contenu de la console avec « console.clear() » ;

console.clear();

// Créer une variable nommée « a » ;

let a;

// Afficher le contenu de « a » avec console.log;

console.log(a);

// Stocker la valeur 15 dans cette variable ;

a = 15;

// Afficher le contenu de cette variable dans la console sous la forme « Ma variable a = ? »

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

// Créer une variable nommée « b » et lui assigner directement la valeur 9 ;

let b = 9;

// Afficher le contenu de cette variable dans la console sous la forme « Ma variable b = ? »

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

// Faire l’addition de ces 2 variables en affichant directement le résultat dans la console sous cette forme : « 15 + 9 = ? » ; (essayer d’utiliser un littéral avec `…${…}…`)

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

// Compléter en faisant de même pour une soustraction, une multiplication et une division des deux variables ;

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

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

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

// Stocker « Bonjour » dans la variable a ;

a = "bonjour";

// Stocker « les amis » dans la variable b ;

b = "les amis";

// Afficher « bonjour les amis » dans la console en concaténant les variables ;

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

// Faites la même chose en utilisant un littéral avec `…${…}…`;

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

// Stocker « true » dans la variable a ;

a = true;

// Stocker « false » dans la variable b ;

b = false;

// Effectuer une opération AND entre les 2 variables et afficher le résultat sous cette forme « true AND false = ? » ;

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

// Effectuer une opération OR entre les 2 variables et afficher le résultat sous cette forme « true OR false = ? » ;

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

// Stocker la date du jour dans la variable a avec new Date();

a = new Date();

// Calculer une nouvelle date dans la variable b qui est 61 jours avant la date courante (utilisation getDate, setDate)

b= new Date();

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

// Afficher les dates contenues dans les variables a et b en vous aidant de « toLocaleString(), toLocaleDateString() et toLocaleTimeString()» . Afficher la date et l'heure, la date uniquement et l'heure uniquement.

console.log(`a = ${a.toLocaleDateString("ch-fr")}`);

console.log(`b = ${b.toLocaleDateString("ch-fr")}`);

// Le mot réservé « typeof » permet de connaitre le type utilisé momentanément pour une variable. Stocker la valeur de Math.PI dans a, « bonjour » dans b, créer et assigner true dans c,

//créer, assigner la date courante dans d et déclarez la variable e sans rien lui affectez, puis afficher le type pour les 5 variables :

a = Math.PI;

b = "bonjour";

let c = true;

let d = new Date();

let e;

console.log(`le type de la variable a est : ${typeof a}`);

console.log(`le type de la variable b est : ${typeof b}`);

console.log(`le type de la variable c est : ${typeof c}`);

console.log(`le type de la variable d est : ${typeof d}`);

console.log(`le type de la variable e est : ${typeof e}`);

## Questions

* Comment déclarez-vous une variable en Javascript ?

let e;

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

const e = 10 ;

Une constante est utile a déclaré pour :

1. Un tableau
2. Un objet
3. Une fonction

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

Les variables avec le mot clé let ne peuvent pas redéclaré mais les variables avec le mot clé var le peuvent

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

Cela crée une variable globale étant visible dans toute l’application.

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

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

Description générée automatiquement

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

console.log(`le type de la variable a est : ${typeof a}`);

# Exercice 6

## Code JavaScript

/\*

But : indique le jour de la date acuelle

Auteur : Esther Pham

Date : 16.05.2023 / V1.0

\*/

function afficherJourSemaine() {

let jour;

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

case 0:

jour = "dimanche";

break;

case 1:

jour = "lundi";

break;

case 2:

jour = "mardi";

break;

case 3:

jour = "mercredi";

break;

case 4:

jour = "jeudi";

break;

case 5:

jour = "vendredi";

break;

case 6:

jour = "samedi";

break;

default:

jour = "BABA";

break;

}

document.getElementById("info").innerHTML = "On est " + jour;

}

function afficherJourSemaineTab() {

const jours = ["dimanche", "lundi", "mardi", "mercredi", "jeudi", "vencredi", "samedi"];

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

document.getElementById("infoTab").innerHTML = "On est " + jours[jour] + "(tableau)";

}

## Questions

* Comment récupérez-vous le numéro du jour actuel ?

new Date().getDay()

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

document.getElementById("info").innerHTML = "On est " + jour; 🡪 info correspon à l’id du paragraphe et .innerHTML permet d’y rajouter du texte

* Comment pouvez-vous insérer du code html dans une balise html ? Par exemple "<b>On est mardi</b>"

document.getElementById("info").innerHTML += `<b> i = ${nbrs[i]}</b>`;

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

Sans valeur :

const jours = [];

Avec valeur :

const jours = ["dimanche", "lundi", "mardi", "mercredi", "jeudi", "vencredi", "samedi"];

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

Ajouter au début :

unshift()

Ajouter à la fin :

push()

Supprimer au début :

shift()

Supprimer à la fin :

pop()

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

arr[index\_position]

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

Il faut faire une boucle qui parcours le tableau, une boucle for est préférable.

# Exercice 7

## Boucle For

function testerFor() {

const nbrs = [0, 1, 2, 3, 4];

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

document.getElementById("info").innerHTML += "for (let i=0; i<5; i++){...}";

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

document.getElementById("info").innerHTML += `<br/> i = ${nbrs[i]}`;

}

document.getElementById("info").innerHTML += "<br/> --> À utiliser si on sait que l'ont veut itérer x fois (x connu avant de commencer la boucle)";

}

## Boucle While

function testerWhile() {

let i = 0;

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

document.getElementById("info").innerHTML += "for (i<5){...}";

while (i < 5) {

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

i++;

}

document.getElementById("info").innerHTML += "<br/> --> À utiliser si on ne sait pas le nombre d'itérations au démarrage de la boucle";

}

## Boucle DoWhile

function testerDoWhile() {

i = 0;

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

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

do {

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

i++

} while (i < 5);

document.getElementById("info").innerHTML += "<br/> --> À utiliser si on ne sait pas le nombre d'itérations au démarrage de la boucle mais avec un passage obligatoire";

}

# Exercice 8

## Code JavaScript (boucle sur des éléments json)

/\*

But : indique le jour de la date acuelle

Auteur : Esther Pham

Date : 16.05.2023 / V1.0

\*/

function parcourirUnTableauJSON() {

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];

document.getElementById("info").innerHTML += "<br/>" + i + " : "

for (let f in personne) {

document.getElementById("info").innerHTML += personne[f] + " ";

}

}

}

# Exercice 9

## Code JavaScript

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;

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

}

# Exercice 10

## Code JavaScript

### Index.js

Dans cette première partie de code on vérifie si le document est bien chargé pour pouvoir effectuer les personnes.

/\*

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

\*/

document.onreadystatechange = function () {

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

\_afficherPersonnes();

}

};

Cette méthode permet d’afficher les personnes déjà présentent dans la base de données json.

/\*

\* 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>";

personnes.forEach(personne => {

txt += '<li onclick="selectionnerPersonne(' + \_trouverPersonne(personne) + ')" >';

txt += personne.toString();

txt += "</li>";

});

txt += "</ul>";

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

}

Cette méthode permet d’afficher les différentes infos des personnes.

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

function \_afficherInfosPersonne(p) {

let prenom = p.prenom;

let nom = p.nom;

let age = p.age;

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

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

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

}

Cette méthode permet d’ajouter des personnes.

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

function \_lireInfosPersonne() {

let p = new Personne();

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

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

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

return p;

}

Affiche les infos de la personne sélectionnée, la particularité de cette méthode est quelle est appelé depuis le vue.

/\*

\* 3. METHODES PUBLIQUES NECESSAIRES A LA VUE

\*/

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

function selectionnerPersonne(i) {

return \_afficherInfosPersonne(personnes[i]);

}

Cette méthode, également appelé depuis la vue permet d’ajouter une personne.

// appelée depuis la vue pour ajouter une personne

function ajouter() {

let p = \_lireInfosPersonne();

ajouterPersonne(p);

\_afficherPersonnes();

}

Cette méthode permet de supprimer une personne, elle sera appelée par la vue.

// appelée depuis la vue pour supprimer une personne

function supprimer() {

let p = \_lireInfosPersonne();

supprimerPersonne(p);

\_afficherPersonnes();

}

### Workers.js

// définition du modèle de données, créé au chargement du js dans le index.html

const personnes = [

new Personne("John", "Doe", 44),

new Personne("Anna", "Smith", 32),

new Personne("Peter", "Jones", 29)

];

// premier tri de la liste de personnes

personnes.sort();

// fonction privée pour retrouver l'index d'une personne dans le tableau, -1 autrement

// il faut comparer avec toString()

function \_trouverPersonne(p) {

let idx = -1;

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

if (p.toString() == personnes[i].toString()) {

idx = i;

break;

}

}

return idx;

}

// ajouter une personne dans la liste des personnes si pas trouvée

function ajouterPersonne(p) {

let idx = \_trouverPersonne(p);

if (idx == -1) {

if (p.prenom.length > 0 && p.nom.length > 0 && p.age > 0) {

personnes.push(p);

}

}

}

// supprimer une personne dans la liste des personnes si trouvée

function supprimerPersonne(p) {

let idx = \_trouverPersonne(p);

if (idx > -1) {

personnes.splice(idx, 1);

}

}

### Personne.js

function Personne(prenom, nom, age) {

this.prenom = prenom ;

this.nom = nom ;

this.age = age ;

}

Personne.prototype.toString = function() {

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

};

# Exercice 11

## But

Dans cette exercice, nous avons transformer les fonctions de l’exercice 10 en classe.

## Class Ctrl (index.js)

/\*

\* 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();

}

};

/\*

\* Première ligne de la classe Ctrl

\*/

class Ctrl {

constructor() {

this.wrk = new Worker(); 🡪 permet d’instancier la classe 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>";

this.wrk.personnes.forEach(personne => {

txt += '<li onclick="ctrl.selectionnerPersonne(' + this.wrk.\_trouverPersonne(personne) + ')" >';

txt += personne.toString();

txt += "</li>";

});

txt += "</ul>";

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

}

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

\_afficherInfosPersonne(p) {

let prenom = p.prenom;

let nom = p.nom;

let age = p.age;

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

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

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

}

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

\_lireInfosPersonne() {

let p = new Personne();

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

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

p.age = 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) {

return 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();

}

}

## Class personne(personne.js)

class Personne {

constructor(prenom, nom, age) {

this.prenom = prenom;

this.nom = nom;

this.age = age;

}

toString() {

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

}

}

## Class Worker(worker.js)

class Worker {

personnes = null;

constructor() {

this.personnes = [

new Personne("John", "Doe", 44),

new Personne("Anna", "Smith", 32),

new Personne("Peter", "Jones", 29)

];

// premier tri de la liste de personnes

this.personnes.sort();

}

// fonction privée pour retrouver l'index d'une personne dans le tableau, -1 autrement

// il faut comparer avec toString()

\_trouverPersonne(p) {

let idx = -1;

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

if (p.toString() == this.personnes[i].toString()) {

idx = i;

}

}

return idx;

}

// ajouter une personne dans la liste des personnes si pas trouvée

ajouterPersonne(p) {

let idx = this.\_trouverPersonne(p);

if (idx == -1) {

if (p.prenom.length > 0 && p.nom.length > 0 && p.age > 0) {

this.personnes.push(p);

}

}

}

// supprimer une personne dans la liste des personnes si trouvée

supprimerPersonne(p) {

let idx = this.\_trouverPersonne(p);

if (idx > -1) {

this.personnes.splice(idx, 1);

}

}

}

# Exercice 12

## But

Il existe 3 façons de déclaré une fonction dans JavaScript, nous allons les présentées dans cette exercice.

## Déclaration d’une fonction normale

function a() {

let val = 1;

console.log(val) ;

}

a();

## Déclaration d’une fonction anonyme

let b = function() {

let val = 2;

console.log(val) ;

} ;

b();

## Déclaration d’une fonction flèche

let e = (a,b) => { // avec plusieurs instructions

let somme = a+b;

return somme;

};

console.log(e(5,7));

# Exercice 13

## Questions

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

Les cookies permettent de stocker un nombre limité de données pouvant être réutilisé plus tard.

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

document.cookie = "cookie\_exercice\_13=" + contenu + "; expires=" + date.toUTCString() + "; path=/";

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

let contenu = document.cookie.split(`=`)[1];

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

Convertir un objet JSON en chaîne de caractère :

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

let personneJson = {

"nom": nom,

"prenom": prenom

};

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

let personne = JSON.stringify(personneJson);

# Exercice 14

## JQuery

Créé en 2005 par John Resig et libéré pour la 1ère fois en janvier 2006. JQuery est une librairie open source simplifiant l’interaction entre du HTML et du JavaScript. Cette librairie a été créé pour simplifier la manipulation du DOM et pour qu’il y a moins de problème dans les différents navigateurs.

JQuery est bien documenté et est utilisé par une grande communité de développeurs. Elle a également une série intéressante de plug-ins, est de petite taille(~25kb) et est compatible avec de nombreux navigateurs.

# Exercice 15 (JQuery)

## Récupérer la valeur d'un champ

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

## Propriété css avec jquery

$("#container").slideToggle().css("background-color", couleur);

## Animation avec jquery

$("#container").fadeOut(1000, function () {

$("#container").slideToggle()

})

# Exercice 16

## Ajouter un écouteur

$("#btnShow").click(function OnClick() {

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

})

## Sélectionner des éléments du DOM

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

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

})

## Ajouter et supprimer des classes CSS

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

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

})

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

$("li:first").removeClass("boldBlueText");

})

## Récupérer des valeurs d'un formulaire

$(document).ready(function () {})

## Ajouter des éléments au DOM

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

## Extraire la valeur d'un attribut HTML

$("#textToInsert").val()

# Exercice 18

## Appel GET / POST en Ajax

### GET

$.ajax({

type: "GET",

url: "https://botw-compendium.herokuapp.com/api/v2/category/creatures",

success: successCallback

});

### POST

$.ajax(url, {

type: "POST",

contentType: "application/x-www-form-urlencoded; charset=UTF-8",

data: param,

success: successCallback,

error: errorCallback

});

### Donner des paramètres

$.ajax(url, {

type: "POST",

contentType: "application/x-www-form-urlencoded; charset=UTF-8",

data: param,

success: successCallback,

error: errorCallback

});

### Méthode de retour en cas de succès

OKCelsius2Fahrenheit(data) {

let temp = $(data).find("temperature").text();

let affiche = temp !== "NaN" ? temp : "";

// Afficher les degrés dans le formulaire

$("#fahrenheit1").val(affiche);

}

### Méthode de retour en cas d'erreur

KOCelsius2Fahrenheit(xhr) {

let erreur = xhr.status + ': ' + xhr.statusText

alert('Erreur - ' + erreur);

}

## Appel GET / POST avec Postman

### GET

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

Description générée automatiquement

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

Description générée automatiquement

### POST

Une image contenant capture d’écran, texte, ligne

Description générée automatiquement

Une image contenant texte, logiciel, Page web, Police

Description générée automatiquement

# Exercice 20

## SPA

Une SPA ou Single Page Application est une application à page unique, ce qui veut dire qu’elle ne va pas charger de nouvelle page HTML lorsque l’utilisateur va interagir avec l’application.

## Charger page HTML dans une autre page HTML

vueService.js :

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();

}

});

}

}

indexCtrl.js :

class IndexCtrl {

constructor() {

this.vue = new VueService();

this.loadLogin();

}

afficherErreurHttp(msg) {

alert(msg);

}

// avec arrow function

loadLogin() {

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

}

## ajax $.ajaxSetup()

/\*

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

\*/

$.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);

},

});

}

# WebServices (ex 17)

## REST

L’architecture de base de REST est :

* Client
* Server
* Uniform Resource Identifiers (URIs)
* HTTP Methods
  + GET
  + POST
  + PUT
  + PATCH
  + DELETE
* Representations
  + JSON
  + XML
  + HTML
* Statelessness
* Response Code

ATTENTION : les méthodes doivent être publiques, donc accessibles par le client et elles doivent également respecter l’architecture REST.

## SOAP

SOAP ou Simple Object Access Protocol utilise xml et http, les messages SAOP sont des messages xml qui encapsulent les données à envoyer entre les applications. Binding SOAP indique comment le message doit être encodé pour le transport.

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

Description générée automatiquement

### Fonctionnement

1. Le client envoie une requête SOAP (l'enveloppe) au serveur en utilisant le protocole HTTP
2. Le serveur reçoit la requête SOAP et extrait les informations de l'enveloppe
3. Le serveur traite la requête en fonction de l'opération spécifiée.
4. La réponse SOAP est encapsulée dans une enveloppe et renvoyée au client.
5. Le client reçoit la réponse SOAP, extrait les informations de l'enveloppe et traite les données reçues.

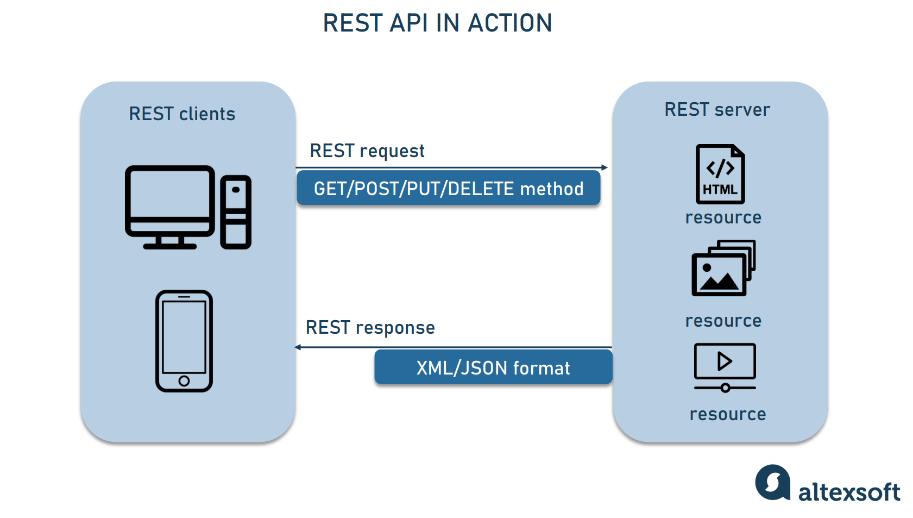
### WSDL

Web Services Description Language est le schéma xml auquel doit correspondre les messages transportés.

## Commandes

### GET

Une requête GET récupère des informations dans une DB et retourne celle-ci au format JSON ou XML.



#### Codes http

Voici quelques codes http qui permettent d’indiquer des informations par rapport à la requête (réussites, erreurs client/serveur, …) :

* 1xx : informations
* 2xx : des succès
* 3xx : redirections
* 4xx : erreurs client
* 5xx: erreurs serveur

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

Description générée automatiquement

### POST

Une requête POST permet d’ajouter un objet dans une API, les différents éléments se trouvant dans la requête sont :

* Destination
* Format
* Objets
* Pas de params
* Permet de définir un token pour avoir les droits
* Content-type permet de définir le type du contenu
* Dans le body de postman, sélectionner moyen d’insérer données (raw, form-data, …)

### PUT

Une requête PUT permet de modifier des entrées, mais également d’en créer une nouvelle.

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

Description générée automatiquement

### DELETE

Une requête DELETE supprime des données du DB. Voici deux manières de faire cette requête, bien sûr il en existe d’autre :

* Cmd :

curl --location --request DELETE 'https://reqres.in/api/users/1’

* JS :

var settings = {

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

"method": "DELETE",

"timeout": 0,

};

### Paramètres

Il est possible de rajouter des paramètres avec le « ? »

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

Description générée automatiquement

### Autorisations

Il est également possible de rajouter des autorisations dans l’onglet «Authorization», dans postman.

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

Description générée automatiquement

# Projet

Dans ce projet, le but est de créer des pages web avec comme base le modèle MVC. Cela nous permet d’avoir une bonne organisation pour le codage des pages web.

L’objectif de ce projet est d’apprendre à utiliser une API est de coder avec javascript, pour que nos pages web aillent chercher des informations sur un service web.

Pour le fonctionnement, les différentes informations présentes sur les pages seront récupérées à partir d’une API.

## API

Ce lien permet de voir la base de données de l’API :

<https://botw-compendium.herokuapp.com/api/v2>

L’API a été trouvé sur GitHub et ce lien permet d’accéder à la documentation :

<https://gadhagod.github.io/Hyrule-Compendium-API/#/>

Lien pour accéder à l’API sur github :

<https://github.com/gadhagod/Hyrule-Compendium-API>

### Tests postman

Voici quelques exemples d’utilisation de l’API avec des GET :

* Cette requête permet de récupérer l’entrée dont le nom correspond à « white-maned lynel » (remplacer les espaces entre les noms par un « \_ »)

GET <https://botw-compendium.herokuapp.com/api/v2/entry/white-maned_lynel>

Résultat :

{

"data": {

"category": "monsters",

"common\_locations": [

"Hyrule Field",

"Hebra Mountains"

],

"description": "These fearsome monsters have lived in Hyrule since ancient times. Their ability to breathe fire makes White-Maned Lynels among the toughest of the species; each one of their attacks is an invitation to the grave. There are so few eyewitness accounts of this breed because a White-Maned Lynel is not one to let even simple passersby escape with their lives.",

"drops": [

"lynel horn",

"lynel hoof",

"lynel guts"

],

"id": 123,

"image": "https://botw-compendium.herokuapp.com/api/v2/entry/white-maned\_lynel/image",

"name": "white-maned lynel"

}

}

* Cette requête permet de récupérer toutes les entrées se trouvant dans la catégorie « monsters »

GET <https://botw-compendium.herokuapp.com/api/v2/category/monsters>

Résultat :

{

"data": [

{

"category": "monsters",

"common\_locations": null,

"description": "Silver Lynels are not to be trifled with. They have been influenced by Ganon's fiendish magic, so they are the strongest among the Lynel species, surpassing even the strength of those with white manes. The term \"silver\" denotes not only their color but also their rarity. The purple stripes help them to stand out even more.",

"drops": [

"lynel horn",

"lynel hoof",

"lynel guts",

"topaz",

"ruby",

"sapphire",

"diamond",

"star fragment"

],

"id": 124,

"image": "https://botw-compendium.herokuapp.com/api/v2/entry/silver\_lynel/image",

"name": "silver lynel"

},

{

"category": "monsters",

"common\_locations": [

"Hyrule Field",

"Tabantha Frontier"

],

"description": "These spell-casting monsters can be found all over Hyrule. They use their thunderstorm rods to hurl balls of electricity or to summon monsters surging with electricity and have been known to cause thunderstorms in the area. The weather will normalize once the Wizzrobe is defeated.",

"drops": [],

"id": 102,

"image": "https://botw-compendium.herokuapp.com/api/v2/entry/thunder\_wizzrobe/image",

"name": "thunder wizzrobe"

},

{

"category": "monsters",

"common\_locations": [

"Gerudo Highlands",

"Hebra Mountains"

],

"description": "This low-level gel monster is engulfed in freezing-cold air. Its strength varies depending on its size. It tends to explode if attacked from close range, so the use of spears, arrows, and other ranged weapons is advised.",

"drops": [

"white chuchu jelly"

],

"id": 86,

"image": "https://botw-compendium.herokuapp.com/api/v2/entry/ice\_chuchu/image",

"name": "ice chuchu"

},

...

* Cette requête permet de récupérer toutes les entrées de l’API

GET <https://botw-compendium.herokuapp.com/api/v2/all>

Résultat :

{

"data": {

"creatures": {

"food": [

{

"category": "creatures",

"common\_locations": [

"Hyrule Field",

"Faron Grasslands"

],

"cooking\_effect": "defense up",

"description": "This beetle's hard body resembles armor. When the shell is cooked with monster parts, the resulting elixir boosts your defense.",

"hearts\_recovered": 0,

"id": 76,

"image": "https://botw-compendium.herokuapp.com/api/v2/entry/rugged\_rhino\_beetle/image",

"name": "rugged rhino beetle"

},

{

"category": "creatures",

"common\_locations": [

"Necluda Sea",

"Lanayru Sea"

],

"cooking\_effect": "defense up",

"description": "This porgy's body is covered in armor-hard scales. The compounds in its scales, when cooked into a dish, fortify your bones and temporarily boost your defense.",

"hearts\_recovered": 1,

"id": 60,

"image": "https://botw-compendium.herokuapp.com/api/v2/entry/armored\_porgy/image",

"name": "armored porgy"

},

{

"category": "creatures",

"common\_locations": [

"Lanayru Great Spring",

"East Necluda"

],

"cooking\_effect": "defense up",

"description": "Calcium deposits in the scales of this ancient fish make them as hard as armor. Cooking it into a dish will fortify your bones, temporarily increasing your defense.",

"hearts\_recovered": 1,

"id": 57,

"image": "https://botw-compendium.herokuapp.com/api/v2/entry/armored\_carp/image",

"name": "armored carp"

},

...

* Cette requête permet de récupérer l’image correspondant à l’entrée mise en paramètre

GET <https://botw-compendium.herokuapp.com/api/v2/entry/white-maned_lynel/image>

Une image contenant dessin humoristique, capture d’écran, Personnage de fiction, Jeu PC

Description générée automatiquement

## Analyse

### Diagramme use cases

ATTENTION : la map n’a pas été faite, car il y avait assez de requête sur l’API cela a été convenu avec le professeur.



### Maquette

Première page (permet de choisir la catégorie, monsters / creatures / treasure / equipment / materials) :

Une image contenant capture d’écran, diagramme, conception

Description générée automatiquement

1 page par catégorie :

Une image contenant capture d’écran, conception, électroménager

Description générée automatiquement

Page MAP pour trouver magasins qui vendent le jeu :

ATTENTION : la map n’a pas été faite, car il y avait assez de requête sur l’API cela a été convenu avec le professeur.

Une image contenant capture d’écran, ligne, croquis, blanc

Description générée automatiquement

## Conception



* Tous les rectangles en vert correspondent aux contrôleurs, qui permettent d’effectuer le code JavaScript pour que l’application soit interactive, mais aussi pour récupérer les informations de l’API.
* Les rectangles bleu correspondent aux pages HTML, ce sera les différents contenant de la page HTML.
* Les cubes orange correspondent aux API dans lesquelles l’on va récupérer les données pour notre page.

## Test

|  |  |
| --- | --- |
| Test | Résultat |
| Chargement du JavaScript | OK : toutes les fonctions interactives de l’application sont fonctionnelles |
| Chargement du CSS | OK : l’apparence de l’application correspond au fichier CSS |
| Liens avec les librairies externes | OK : on peut charger des infos dan l’HTML avec JQuery |

# Conclusion

Dans ce module, nous avons appris à utiliser JavaScript pour avoir une application interactive. Nous avons également appris à récupérer des données sur des API grâce au Webservice. J’ai bien aimé avoir un projet à faire et décidé sur quoi et comment l’application pouvait être interactive. Je trouvais cependant difficile de finir à temps les exercice, de les documenter et de finir le projet en si peu de temps, car certains exercices m’ont pris plus de temps à faire et je trouvais cela dommage.