

Fondamentaux du Web

HTML, CSS, JavaScript

Lab 5 — JavaScript en pratique

Variables, fonctions, conditions

Licence 1 DUT1-Informatique — 2025–2026

Informations pratiques

Où sommes-nous ? Au CM3, on a découvert le langage JavaScript dans la *console* (notre brouillon). Aujourd'hui, on quitte le brouillon : on écrit notre **premier vrai fichier de code**, `script.js`, branché à la page Jàng Afrig.

Avant de commencer, vous devez avoir :

- Votre repo `jàng-afrig` ouvert dans VS Code, avec Live Server qui fonctionne.
- Les pages HTML et le CSS des Labs précédents (jusqu'au tag `v0.4`).
- Les notions du CM3 en tête : `let/const`, types, conditions, boucles, fonctions, et les **trois règles d'or**.

Validation de fin de séance : votre fichier `script.js` contient toutes les fonctions d'entraînement, la console les exécute *sans erreur rouge*, et vous avez poussé un commit avec le tag `v0.5` ainsi qu'un rapport.

Objectifs du Lab 5

À la fin de cette séance, vous saurez :

1. Créer un fichier `script.js` et le brancher à `index.html`.
2. Afficher des messages dans la console avec `console.log`.
3. Déclarer des variables et écrire des fonctions qui renvoient un résultat.
4. Prendre des décisions avec `if/else` et le ternaire.
5. Parcourir un tableau avec `for...of`.
6. Coder cinq fonctions utiles : convertisseur FCFA, moyenne, majorité, FizzBuzz « sénégalais », palindrome.

Table des matières

Rappel express du CM3	2
1 Mise en place : créer et brancher <code>script.js</code>	2
1.1 Créer le fichier	2
1.2 Le brancher à la page	2
1.3 Vérifier que tout est branché	2

2 Parler à la console	3
3 Variables et types, pour de vrai	3
4 Ma première fonction	3
5 Prendre des décisions : les conditions	4
6 Répéter sans se fatiguer : for...of	4
7 Exercices d'application	5
Exercice 4 — FizzBuzz sénégalais	6
8 Aide-mémoire du Lab 5	7
9 Sauvegarder mon travail (routine Git)	7
10 Résumé et transition	8

Rappel express du CM3

Trois choses à garder sous la main :

- **Variables** : `const` par défaut, `let` si ça change, jamais `var`.
- **Comparaison** : toujours `===`, jamais `==`.
- **Fonction** : `function nom(parametres) { ... return ... }`.

1 Mise en place : créer et brancher `script.js`

1.1 Créer le fichier

Dans VS Code, à la racine de votre dossier `jàng-afrig` (au même endroit que `index.html` et `style.css`), créez un nouveau fichier nommé exactement `script.js`.

1.2 Le brancher à la page

Pour que la page exécute votre code, on ajoute la balise `<script>` dans le `<head>` de `index.html`, avec le mot `defer` (rappel du CM3 : « attends que la page soit prête »).

```
<head>
  <meta charset="UTF-8">
  <title>Jàng Afrig</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js" defer></script>
</head>
```

1.3 Vérifier que tout est branché

Dans `script.js`, écrivez une seule ligne :

```
console.log("script.js est bien branché !");
```

Vérification

Lancez la page avec Live Server, puis appuyez sur **F12** et ouvrez l'onglet **Console**. Vous devez voir le message « *script.js est bien branché!* ». Si rien n'apparaît, vérifiez l'orthographe du nom de fichier et la balise `<script>`. **Si ça ne marche pas, levez la main.**

La page ne « change » pas encore, c'est normal

Aujourd'hui, tout votre code s'affiche dans la **console**, pas sur la page. C'est voulu : apprendre à modifier la page visible viendra au **Lab 7** (le DOM). Pour l'instant, la console est votre terrain d'entraînement.

2 Parler à la console

On commence doux. Tapez ceci dans `script.js` (remplacez par votre prénom) :

```
const prenom = "Fatou";
console.log("Bonjour " + prenom);
console.log(`Bonjour ${prenom}, prêt(e) à coder ?`);
```

Les deux dernières lignes affichent un message personnalisé. La première version *colle* le texte avec `+` ; la seconde utilise une *template string* (les accents graves) du CM3 : plus lisible.

Vérification

La console doit afficher deux lignes commençant par « *Bonjour* ». Changez la valeur de `prenom`, sauvegardez : Live Server recharge, le message change.

3 Variables et types, pour de vrai

Rangeons quelques informations sur un joueur de Jàng Afrig :

```
const pseudo = "Aminata"; // texte (string)
let score = 0; // nombre (number), va changer -> let
const partieFinie = false; // booléen (boolean)

console.log(`Joueur : ${pseudo}`);
console.log(`Score de départ : ${score}`);

score = score + 5; // on réécrit l'ardoise
console.log(`Nouveau score : ${score}`); // 5
```

Vérification

La console doit montrer le score passer de 0 à 5. Essayez ensuite de réaffecter `pseudo` (`pseudo = "Moussa";`) : la console affichera une **erreur rouge**, car `pseudo` est un `const`. C'est exactement ce qu'on attend : c'est la preuve que l'étiquette est « gravée ».

4 Ma première fonction

Une fonction est une machine réutilisable (CM3). Écrivons-en une qui fabrique un message de bienvenue :

```
function direBonjour(nom) {  
  return `Bonjour ${nom}, bienvenue sur Jàng Afrig !`;  
}  
  
console.log(direBonjour("Cheikh"));  
console.log(direBonjour("Ndèye"));
```

On a écrit la machine *une fois*, et on l'appelle *autant de fois* qu'on veut, avec des noms différents.

Exercice éclair

Écrivez une fonction `direAuRevoir(nom)` qui renvoie « *À bientôt, [nom] !* », et appelez-la deux fois avec deux prénoms.

5 Prendre des décisions : les conditions

Une fonction peut **choisir** quoi renvoyer selon la situation. Reprenons l'appréciation d'une note (CM3) :

```
function appreciation(note) {  
  if (note >= 16) {  
    return "Très bien";  
  } else if (note >= 12) {  
    return "Bien";  
  } else if (note >= 10) {  
    return "Passable";  
  } else {  
    return "À retravailler";  
  }  
}  
  
console.log(appreciation(14)); // "Bien"  
console.log(appreciation(8));  // "À retravailler"
```

Quand le choix est binaire, le **ternaire** fait la même chose en une ligne :

```
function statutMajorite(age) {  
  return age >= 18 ? "majeur" : "mineur";  
}  
  
console.log(statutMajorite(20)); // "majeur"
```

6 Répéter sans se fatiguer : for...of

Pour traiter *chaque* élément d'une liste, on utilise `for...of` (CM3). Affichons une petite liste de pays :

```
const pays = ["Sénégal", "Mali", "Ghana", "Kenya"];  
  
for (const nom of pays) {  
  console.log(`Pays africain : ${nom}`);  
}  
  
console.log(`Il y a ${pays.length} pays dans la liste.`);
```

Notez `pays.length` : la *longueur* du tableau (ici 4). On s'en servira beaucoup.

Vérification

La console doit afficher quatre lignes « Pays africain : ... », puis « Il y a 4 pays dans la liste. ».

7 Exercices d'application

C'est le cœur de la séance. Ajoutez ces fonctions à `script.js` et testez-les avec `console.log`.

Exercice 1 — Convertisseur FCFA → EUR

Le taux est fixe : **1 EUR = 656 FCFA**. Écrivez `fcfaVersEuro(montant)` qui renvoie le montant en euros.

Indice : il faut **diviser** par 656. Testez avec 10000 FCFA.

```
function fcfaVersEuro(montant) {  
  return montant / 656;  
}  
  
console.log(fcfaVersEuro(10000)); // environ 15.24
```

Exercice 2 — Moyenne d'un étudiant

Écrivez `moyenne(notes)` qui reçoit un tableau de notes et renvoie leur moyenne.

Indice : additionnez toutes les notes dans une variable `somme` avec `for...of`, puis divisez par `notes.length`.

```
function moyenne(notes) {  
  let somme = 0;  
  for (const note of notes) {  
    somme = somme + note;  
  }  
  return somme / notes.length;  
}  
  
const notesAminata = [12, 15, 9, 14];  
console.log(`Moyenne : ${moyenne(notesAminata)}`); // 12.5
```

Exercice 3 — Majorité

Écrivez `estMajeur(age)` qui renvoie un **booléen** (`true` ou `false`).

Indice : la comparaison `age >= 18` est *déjà* un booléen ! On peut la renvoyer directement.

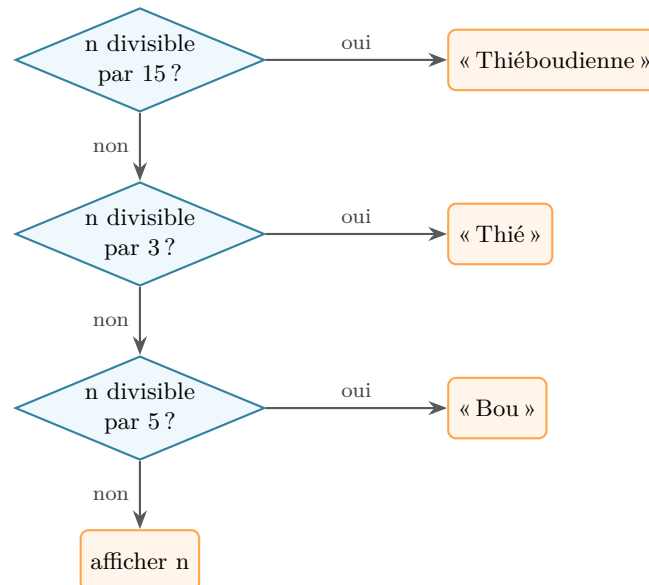
```
function estMajeur(age) {  
  return age >= 18;  
}  
  
console.log(estMajeur(20)); // true  
console.log(estMajeur(16)); // false
```

Exercice 4 — FizzBuzz « à la sénégalaise »

Un classique de la programmation, version Jàng Afrig. Pour chaque nombre de 1 à 30 :

- s'il est multiple de **3 et de 5** (donc de 15) → afficher « *Thiéboudienne* » ;
- sinon, s'il est multiple de **3** → afficher « *Thié* » ;
- sinon, s'il est multiple de **5** → afficher « *Bou* » ;
- sinon → afficher le nombre.

Le piège : l'ordre des tests compte. Si on testait « multiple de 3 » *avant* « multiple de 15 », le nombre 15 afficherait « *Thié* » au lieu de « *Thiéboudienne* ». On teste donc **toujours le cas le plus précis en premier** (le 15).



L'ordre des conditions : on teste d'abord le cas le plus précis (15).

```

function fizzBuzzSenegal() {
  for (let n = 1; n <= 30; n++) {
    if (n % 15 === 0) {
      console.log("Thiéboudienne");
    } else if (n % 3 === 0) {
      console.log("Thié");
    } else if (n % 5 === 0) {
      console.log("Bou");
    } else {
      console.log(n);
    }
  }
}

fizzBuzzSenegal();
  
```

Rappel : `n % 15 === 0` se lit « le reste de la division de `n` par 15 est nul », c'est-à-dire « `n` est un multiple de 15 ».

Exercice 5 — Palindrome

Un palindrome est un mot qui se lit pareil à l'endroit et à l'envers (« kayak », « radar », « ressasser »). Écrivez `estPalindrome(mot)` qui renvoie un booléen.

Indice : construisez le mot à l'envers en plaçant chaque lettre *devant* les précédentes, puis

comparez avec `===`.

```
function estPalindrome(mot) {
  let inverse = "";
  for (const lettre of mot) {
    inverse = lettre + inverse; // on empile à l'envers
  }
  return mot === inverse;
}

console.log(estPalindrome("kayak")); // true
console.log(estPalindrome("Dakar")); // false
```

8 Aide-mémoire du Lab 5

Besoin	Code
Brancher le JS	<code><script src="script.js" defer></script></code>
Afficher dans la console	<code>console.log(...)</code>
Variable fixe / variable	<code>const x = ... / let x = ...</code>
Message avec variable	<code>`Texte \${variable}`</code>
Fonction qui renvoie	<code>function f(p) { return ... }</code>
Décision	<code>if (...) {...} else {...} ou cond ?</code> <code>a : b</code>
Parcourir un tableau	<code>for (const e of tableau) {...}</code>
Taille d'un tableau	<code>tableau.length</code>
Reste d'une division	<code>a % b</code>

9 Sauvegarder mon travail (routine Git)

Comme à chaque Lab, on enregistre le travail dans Git et on pose le tag de la séance.

Routine Git de fin de Lab — tag v0.5

Dans le terminal de VS Code, à la racine du projet :

```
git status
git add .
git commit -m "Lab 5 : premières fonctions JavaScript"
git tag v0.5
git push
git push --tags
```

Vérifiez ensuite sur GitHub que le commit *et* le tag v0.5 apparaissent bien.

Rapport à produire

Avant la fin, déposez un rapport court (1 à 2 pages) dans le dossier **rapports/** de votre repo, nommé **rapport-lab-5-NOM-Prenom.pdf**. Il contient **quatre** sections :

1. **Ce que j'ai fait** : les fonctions codées (FCFA, moyenne, majorité, FizzBuzz, palindrome).
2. **Difficultés rencontrées** : copiez-collez les messages d'erreur qui vous ont bloqué.

3. **Comment je les ai résolues** : ce que vous avez cherché, à qui vous avez demandé.
4. **Lien vers mon repo** avec le tag : `.../jang-afrig/releases/tag/v0.5`.

L'erreur fait partie de l'apprentissage : soyez honnête, c'est le but du rapport.

10 Résumé et transition

Vous venez d'écrire votre **premier vrai fichier JavaScript** et de coder cinq fonctions qui marchent. Vous savez désormais déclarer des variables, écrire des fonctions, décider avec des conditions et répéter avec `for...of`.

Au **Lab 6**, on monte d'un cran : on apprend à manipuler des **collections** de données (des tableaux, des objets) avec des outils élégants — `forEach`, `map`, `filter`. On créera le fichier `data.js` contenant nos premiers pays africains : la matière première du futur quiz.

Première ampoule allumée. Au Lab 6, on remplit la maison de données.

Dr. El Hadji Bassirou TOURÉ — UCAD
Licence 1 DUT1-Informatique — Fondamentaux du Web — 2025-2026