

TD JavaScript n°1 : Variables, saisie, conditions, boucles

Niveau : 1SIO, première séance de JavaScript

Environnement : Node.js, exécution en console

Consignes générales :

- Chaque exercice est écrit dans son propre fichier (ex01.js, ex02.js, ...).
- Un programme s'exécute avec : `node ex01.js`.
- Testez systématiquement avec plusieurs valeurs.
- Quand un message d'erreur apparaît, lisez-le : il indique le fichier, la ligne et la nature du problème.
- Les parties **Défi** sont facultatives.

Préparation de l'environnement

Vérifier Node.js

Dans un terminal :

```
node --version
npm --version
```

Les deux commandes doivent afficher un numéro de version.

Créer le dossier du TD

```
mkdir td-js1
cd td-js1
npm init -y
```

La commande `npm init -y` crée un fichier `package.json` qui décrit le projet.

Installer la bibliothèque de saisie

En Node.js, la fonction `prompt()` des navigateurs n'existe pas. On utilise la bibliothèque `prompt-sync` :

```
npm install prompt-sync
```

Un dossier `node_modules` apparaît : il contient la bibliothèque.

Utiliser la bibliothèque

Tout programme qui demande une saisie doit commencer par ces deux lignes :

```
const prompt = require("prompt-sync")();
```

Test à réaliser dans un fichier `test.js` :

```
const prompt = require("prompt-sync")();

const nom = prompt("Votre prénom ? ");
console.log("Bonjour " + nom);
```

Lancez-le avec `node test.js`. Si votre prénom s'affiche, l'environnement est prêt.

Rappel important : `prompt()` renvoie toujours **une chaîne de caractères**. Pour obtenir un nombre, on convertit avec `Number(texte)`.

Partie A : Variables, constantes, affichage

Exercice 1 : Premiers affichages

Énoncé

1. Dans `ex01.js`, affichez votre prénom avec `console.log()`.
2. Déclarez trois variables : `prenom` (chaîne), `age` (nombre), `ville` (chaîne).
3. Affichez chacune sur une ligne séparée.
4. Affichez une phrase unique construite avec des accents graves et `${}` :

```
Je m'appelle Léa, j'ai 17 ans et j'habite à Lyon.
```

Résultat attendu : quatre affichages dans le terminal, dont la phrase complète avec vos propres valeurs.

Exercice 2 : Déclarer, affecter, modifier

Énoncé

1. Déclarez une constante `prixBaguette` valant 1.2.
2. Déclarez une variable `nbBaguettes` valant 3.
3. Affichez : `Total : 3.6 €` (le total est calculé, pas écrit à la main).
4. Affectez la valeur 5 à `nbBaguettes` et affichez de nouveau le total.
5. Essayez d'affecter 1.5 à `prixBaguette`.

Questions

1. Quel message d'erreur obtenez-vous à l'étape 5 ?
2. Quelle différence faites-vous entre *déclarer* une variable et lui *affecter* une valeur ?
3. Dans quel cas choisir `const` plutôt que `let` ?

Exercice 3 : La TVA

Énoncé

Un article coûte 49,90 € HT. Le taux de TVA est de 20 %.

1. Stockez le prix HT et le taux dans deux constantes.
2. Calculez le montant de la TVA, puis le prix TTC.
3. Affichez les trois lignes suivantes (avec vos valeurs calculées) :

```
Prix HT   : 49.9 €  
TVA       : 9.98 €  
Prix TTC  : 59.88 €
```

Défi : affichez les montants arrondis à 2 décimales avec `.toFixed(2)`. Testez avec un prix HT de 19,99 € : que constatez-vous sans `.toFixed(2)` ?

Exercice 4 : Prédire avant d'exécuter

Énoncé

Pour chaque bloc, **écrivez d'abord sur papier** ce que vous pensez voir s'afficher. Exécutez ensuite le code et comparez.

Bloc 1 :

```
let a = 10;  
let b = a;  
a = 20;  
console.log(a, b);
```

Bloc 2 :

```
console.log("5" + 3);  
console.log("5" - 3);  
console.log("5" * "2");
```

Bloc 3 :

```
console.log(7 / 2);  
console.log(7 % 2);  
console.log(10 % 5);
```

Bloc 4 :

```
console.log(5 == "5");  
console.log(5 === "5");
```

```
console.log(5 !== 6);
```

Questions

1. Pour chaque résultat différent de votre prédiction, expliquez pourquoi.
2. À quoi sert l'opérateur % ? Donnez un cas d'usage (par exemple savoir si un nombre est pair).
3. Quelle différence entre == et === ?

Partie B : Saisie et conditions

Exercice 5 : Calculateur d'IMC

Rappel : le fichier doit commencer par `const prompt = require("prompt-sync")()`; `prompt("question")` renvoie une chaîne ; `Number(texte)` la convertit en nombre.

Énoncé

1. Demandez le poids en kg, puis la taille en mètres (exemple : 1.75).
2. Convertissez les deux saisies en nombres.
3. Calculez l'IMC : `poids / (taille * taille)`.
4. Affichez : Votre IMC est de 22.9 (arrondi à 1 décimale avec `.toFixed(1)`).
5. Affichez ensuite l'interprétation :
 - moins de 18,5 : Insuffisance pondérale
 - de 18,5 à moins de 25 : Corpulence normale
 - de 25 à moins de 30 : Surpoids
 - 30 et plus : Obésité

Tests à réaliser : 70 kg / 1,75 m ; 50 kg / 1,80 m ; 95 kg / 1,70 m.

Question : que se passe-t-il si vous oubliez la conversion avec `Number()` ? Le programme fonctionne-t-il quand même ? Pourquoi ?

Exercice 6 : Majeur ou mineur ?

Énoncé

1. Demandez l'âge de l'utilisateur et convertissez-le en nombre.
2. Affichez Majeur s'il a 18 ans ou plus, Mineur sinon.
3. Ajoutez la catégorie Senior à partir de 65 ans (utilisez `else if`).

Tests à réaliser : 17, 18, 40, 65, 0.

Défi : si l'utilisateur saisit un texte non numérique (par exemple abc), `Number()` renvoie NaN. Affichez Saisie invalide dans ce cas (indice : `isNaN(valeur)`).

Exercice 7 : Le tarif du cinéma

Énoncé

Un cinéma applique les tarifs suivants :

Situation	Tarif
Moins de 12 ans	5 €
De 12 à 25 ans inclus	7 €
Plus de 25 ans	10 €
Le mercredi, quel que soit l'âge	6 €

1. Demandez l'âge puis le jour de la semaine (en minuscules : lundi, mardi, ...).
2. Affichez le tarif applicable : Tarif : 7 €.
3. Le tarif du mercredi est prioritaire sur les tarifs par âge.

Tests à réaliser : (10, lundi), (20, lundi), (40, lundi), (10, mercredi), (40, mercredi).

Défi : le tarif « moins de 12 ans » est gratuit s'il est accompagné d'un adulte (question supplémentaire, réponse oui ou non). Utilisez &&.

Exercice 8 : Pierre, feuille, ciseaux

Fourni : ce code tire un choix aléatoire pour l'ordinateur.

```
const choix = ["pierre", "feuille", "ciseaux"];
const ordi = choix[Math.floor(Math.random() * 3)];
```

Énoncé

1. Demandez le choix du joueur avec `prompt()` (pierre, feuille ou ciseaux).
2. Affichez les deux choix : Vous : pierre / Ordinateur : ciseaux.
3. Affichez le résultat : Gagné, Perdu ou Égalité.

Règles : pierre bat ciseaux, ciseaux bat feuille, feuille bat pierre.

Contrainte : ne pas écrire 9 cas séparés. Traitez d'abord l'égalité, puis les trois cas où le joueur gagne ; tous les autres cas sont des défaites.

Défi : si la saisie n'est pas l'un des trois mots, affichez `Choix invalide`.

Partie C : Itérations

Exercice 9 : Compter

Énoncé

1. Avec une boucle `for`, affichez les entiers de 1 à 10, un par ligne.
2. Affichez ensuite les nombres **pairs** de 2 à 20.
3. Demandez un nombre *n* à l'utilisateur et affichez sa table de multiplication de 1 à 10 :

```
7 x 1 = 7
7 x 2 = 14
...
7 x 10 = 70
```

Défi : affichez la somme des entiers de 1 à 100 (résultat attendu : 5050).

Exercice 10 : Somme et moyenne

Énoncé

1. Déclarez une variable `total` initialisée à 0.
2. Avec une boucle, demandez 5 notes (`Note 1`, `Note 2`, ...), convertissez-les en nombres et ajoutez-les à `total`.
3. Affichez la moyenne avec 2 décimales.
4. Affichez `Admis` si la moyenne est supérieure ou égale à 10, `Refusé` sinon.

Test : notes 12, 8, 15, 10, 9 → moyenne 10.80, `Admis`.

Défi : affichez aussi la meilleure note (utilisez une variable `max` mise à jour dans la boucle).

Exercice 11 : Le nombre mystère

Fourni :

```
const secret = Math.floor(Math.random() * 100) + 1;
```

Énoncé

1. Déclarez un compteur `essais` à 0.
2. Tant que le joueur n'a pas trouvé `secret`, demandez une proposition, incrémentez `essais` et affichez `Trop grand` ou `Trop petit`.
3. À la fin, affichez : `Bravo ! Trouvé en 7 essais`.

Conseil : pour tester, affichez temporairement `secret` dans la console.

Défi : limitez à 7 essais maximum ; au-delà, affichez `Perdu, le nombre était ....`

Partie D : Découverte des fonctions

Exercice 12 : Ma première fonction (guidé)

Énoncé

Voici une fonction qui calcule un prix TTC (TVA à 20 %) :

```
function calculerTTC(prixHT) {  
  return prixHT * 1.2;  
}
```

1. Recopiez-la et appelez-la avec 49.9, 100 et 15.5 : `console.log(calculerTTC(49.9));`
2. Identifiez dans le code : le nom de la fonction, le paramètre, la valeur renvoyée.
3. Modifiez-la pour qu'elle accepte un second paramètre `taux` (par exemple 0.055 pour les produits alimentaires).

Question : quel est l'intérêt d'une fonction par rapport à recopier le calcul à chaque fois ?

Exercice 13 : À vous d'écrire

Énoncé

1. Écrivez une fonction `estMajeur(age)` qui renvoie `true` si `age >= 18`, `false` sinon.
 2. Utilisez-la dans un `if` pour afficher `Accès autorisé` ou `Accès refusé`.
 3. Écrivez une fonction `estPair(n)` qui renvoie `true` si `n` est pair.
 4. Utilisez `estPair` dans une boucle pour afficher uniquement les nombres pairs de 1 à 20.
-

Partie E (bonus) : Lire et écrire des fichiers

Jusqu'ici, vos programmes « oublient » tout à la fin de leur exécution. Avec le module `fs` (inclus dans Node.js, rien à installer), un programme peut écrire et relire des fichiers.

```
const fs = require("fs");  
  
fs.writeFileSync("test.txt", "Bonjour\n"); // crée ou ÉCRASE le fichier  
fs.appendFileSync("test.txt", "Ligne ajoutée\n"); // ajoute à la fin  
const contenu = fs.readFileSync("test.txt", "utf8");  
console.log(contenu);  
console.log(fs.existsSync("test.txt")); // true ou false
```

Attention

- `writeFileSync` écrase le contenu existant sans prévenir.
 - Les chemins sont relatifs au dossier depuis lequel vous lancez `node`, pas à celui du script.
 - Lire un fichier qui n'existe pas provoque une erreur : testez d'abord avec `fs.existsSync()`.
-

Exercice 14 : Meilleur score du nombre mystère

Énoncé Reprenez votre programme du nombre mystère (exercice 11) et ajoutez la mémorisation du meilleur score dans un fichier `score.txt`.

1. Au démarrage, si `score.txt` existe, lisez-le et affichez : `Meilleur score actuel : X essais`. Sinon, n'affichez rien.
2. Pensez à convertir le contenu lu avec `Number()`.
3. À la fin de la partie, comparez le nombre d'essais au meilleur score :

- si le fichier n'existe pas, ou si le score de la partie est meilleur (plus petit), écrivez-le dans `score.txt` et affichez `Nouveau record !` ;
 - sinon, ne modifiez pas le fichier.
4. Lancez le programme plusieurs fois de suite et vérifiez que le record est bien conservé entre deux exécutions.

Résultat attendu (deuxième lancement) :

```
Meilleur score actuel : 7 essais
Devinez le nombre entre 1 et 100 : ...
```

Défi : supprimez `score.txt` à la main et relancez : le programme ne doit pas planter.

Exercice 15 : Analyse d'un fichier de log

Préparation Créez un fichier `serveur.log` avec le contenu suivant (recopiez-le tel quel) :

```
2026-10-08 08:01:12 INFO Démarrage du serveur
2026-10-08 08:03:45 INFO Connexion de alice
2026-10-08 08:15:02 ERROR Échec de connexion pour bob
2026-10-08 08:15:30 WARNING Disque à 85%
2026-10-08 08:17:11 ERROR Échec de connexion pour bob
2026-10-08 08:20:00 INFO Connexion de carole
2026-10-08 09:02:33 ERROR Timeout base de données
2026-10-08 09:05:10 INFO Sauvegarde terminée
```

Énoncé Écrivez un programme `ex15.js` qui :

1. lit le fichier `serveur.log` ;
2. découpe son contenu en lignes avec `contenu.split("\n")` ;
3. parcourt les lignes avec une boucle `for` ;
4. compte les lignes contenant `INFO`, celles contenant `WARNING` et celles contenant `ERROR` (utilisez `ligne.includes("ERROR")`) ;
5. affiche :

```
INFO : 4
WARNING : 1
ERROR : 3
```

1. affiche ensuite uniquement les lignes d'erreur, précédées de leur numéro de ligne.

Questions

1. Que se passe-t-il si une ligne vide termine le fichier ? Comment l'ignorer ?
2. En quoi ce type de programme est-il utile à un administrateur système ?

Défi : écrivez les lignes `ERROR` dans un nouveau fichier `erreurs.log` avec `fs.appendFileSync` dans la boucle.

Exercice 16 : Sauvegarder des données en JSON

Énoncé Écrivez un programme `ex16.js` qui gère une liste de contacts :

1. Au démarrage, si `contacts.json` existe, chargez-le avec `JSON.parse(fs.readFileSync("contacts.json", "utf8"))`. Sinon, partez d'un tableau vide `[]`.
2. Affichez le nombre de contacts déjà enregistrés.
3. Demandez un nom et un téléphone à l'utilisateur, puis ajoutez le contact au tableau avec `contacts.push({ nom: nom, tel: tel })`.
4. Répétez tant que l'utilisateur ne saisit pas un nom vide.
5. À la fin, sauvegardez le tableau avec `fs.writeFileSync("contacts.json", JSON.stringify(contacts, null, 2))`.
6. Affichez la liste complète des contacts.

Vérification

1. Lancez le programme, ajoutez deux contacts, quittez.
2. Relancez-le : les deux contacts doivent être présents.
3. Ouvrez `contacts.json` dans un éditeur de texte et observez le format.

Question : quelles sont les limites de cette solution par rapport à une base de données (pensez à deux utilisateurs en même temps, à la recherche, à la taille) ?

Aide-mémoire

Besoin	Syntaxe
Lancer un programme	<code>node fichier.js</code>
Importer la saisie	<code>const prompt = require("prompt-sync")();</code>
Afficher	<code>console.log(valeur)</code>
Saisir	<code>prompt("question")</code> (renvoie du texte)
Convertir en nombre	<code>Number(texte)</code>
Arrondir à n décimales	<code>nombre.toFixed(n)</code>
Partie entière	<code>Math.floor(x)</code>
Nombre aléatoire entre 0 et 1	<code>Math.random()</code>
Condition	<code>if (...) { } else if (...) { } else { }</code>
ET / OU / NON	<code>&& / / !</code>
Boucle comptée	<code>for (let i = 1; i <= 10; i++) { }</code>
Boucle conditionnelle	<code>while (condition) { }</code>
Fonction	<code>function nom(param) { return valeur; }</code>

From:

<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:

<http://slamwiki2.kobject.net/sio/bloc1/td1?rev=1791499636>

Last update: **2026/10/09 00:47**

