Développement web S1 (R112) – TP #4

Liens utiles : CM #1 · TP #2 · TP #3

Avant de commencer, créez un répertoire nommé `r112-tp4` contenant des répertoires `exo1`, `exo2`, etc. Chacun de ces répertoires contient votre travail sur l'exercice correspondant. Le fichier HTML de chaque exercice doit s'appeler `index.html`. **Vos exercices ne seront pas corrigés si vous ne respectez pas ces instructions.**

À la fin de la séance, vous devez compresser (en format ZIP ou RAR) le répertoire `r112-tp4` avec tous vos exercices et le soumettre sur AMeTICE.

Seuls les exercices marqués avec une étoile (*) seront évalués. La date limite pour rendre vos exercices est visible sur AMeTICE, et aucun travail ne sera accepté après. Le corrigé sera accessible après l'évaluation des exercices.

Prévenir et corriger les erreurs

En programmation les erreurs (appelés *bugs*) arrivent tout le temps, donc il faut savoir les éviter, les trouver et les corriger. Il existe deux types d'erreurs : les erreurs de syntaxe et les erreurs logiques

Les **erreurs de syntaxe** sont les fautes d'orthographes, quand vous écrivez mal les instructions de JavaScript, vos variables, ou que vous oubliez de fermer les parenthèses.
Un bon éditeur de texte est capable de détecter ces erreurs. Indentez bien votre code pour éviter ce type d'erreurs.

```
for(let i = 0;i < 10; i++) {if(i<=7) {
message += `${mots[i]}, `} else
if (i == 8){message += `{mots[i]} et `
}else
{message += `{mots[i]}.`}
```

```
for (let i = 0; i < 10; i++) {
  if (i <= 7) {
    message += `${mots[i]}, `
  } else if (i == 8) {
    message += `{mots[i]} et `
  } else {
    message += `{mots[i]}.`
  }
// il manque une accolade ici
```

Les **erreurs logiques** sont les erreurs dans votre algorithme. Le code marche, mais il ne fait pas ce que vous voulez. Pour les trouver, vous devez *déboguer* votre programme.

Pour suivre l'exécution de votre code pas à pas, vous pouvez utiliser l'instruction `debugger` et les outils de développement de votre navigateur web. Ouvrez l'outil de débogage de votre navigateur web, ouvrez la page web, et le code s'arrêtera à l'instruction `debugger`. Vous pourrez ensuite contrôler l'exécution de la suite de votre programme.

```
let prenom = "Frank"
let nom = "Zappa"
debugger
console.log(`Connaissez-vous ${prenom} ${nom} ?`)
```

Exercices

Exercice 1

Dans cet exercice vous allez manipuler un tableau qui contient des objets.

Vous devez afficher les comptes les plus suivis d'Instagram. Pour chaque compte vous devez afficher un message du type «Le compte de Cristiano Ronaldo (@cristiano) a 484 millions d'abonnés.»

Utilisez ce tableau :

```
let comptes = [
  {"nom": "Instagram", "compte": "@instagram", "abonnes": 552},
  {"nom": "Cristiano Ronaldo", "compte": "@cristiano", "abonnes": 484},
  {"nom": "Kylie Jenner", "compte": "@kyliejenner", "abonnes": 370},
  {"nom": "Lionel Messi", "compte": "@leomessi", "abonnes": 363},
  {"nom": "Selena Gomez", "compte": "@selenagomez", "abonnes": 348},
  {"nom": "Dwayne Johnson", "compte": "@therock", "abonnes": 339},
  {"nom": "Ariana Grande", "compte": "@arianagrande", "abonnes": 332},
  {"nom": "Kim Kardashian", "compte": "@kimkardashian", "abonnes": 330},
  {"nom": "Beyoncé", "compte": "@beyonce", "abonnes": 276},
  {"nom": "Khloé Kardashian", "compte": "@khloekardashian", "abonnes": 273}
]
```

Exercice 2

Faites un programme qui demande à l'utilisateur le nom des habitants des Bouches-du-Rhône jusqu'à avoir la réponse correcte (Bucco-Rhodaniens).

Le programme pose la question à l'utilisateur, qui doit donner une réponse. Il y a alors deux possibilités :

- La réponse est correcte : le programme finit
- La réponse n'est pas correcte : on recommence

Vous aurez besoin d'une boucle de répétition.

Exercice 3

Copiez le code de l'exercice 2 et ajoutez les améliorations suivantes :

- Si l'utilisateur ne donne pas la bonne réponse, le message commence par «Réponse incorrecte»
- Quand l'utilisateur donne la bonne réponse, une fenêtre modale lui félicite et lui dit combien de réponses il a donné.

Exercice 4* noté

Faites un programme qui demande à l'utilisateur de deviner un nombre entier secret entre 1 et 100.

Le nombre secret est le 41. Après chaque réponse, le programme dit si le nombre est plus grand ou plus petit et demande une nouvelle réponse. Le programme s'arrête quand l'utilisateur donne la réponse correcte.

Exercice 5

Faites les améliorations suivantes sur l'exercice 4 :

- Quand l'utilisateur donne la bonne réponse, une fenêtre modale lui félicite et lui dit combien de réponses il a donné.
- Si l'utilisateur donne un nombre qui n'est pas compris entre 1 et 100, on lui rappelle que le nombre secret est entre 1 et 100.
- Si l'utilisateur écrit un nombre qui n'est pas entier, on lui rappelle que le nombre secret est un entier.

Pour savoir si un nombre est entier, utilisez la fonction Number.isInteger() :

```
let nombre = 3.14
console.log(Number.isInteger(nombre)) // affiche false
console.log(Number.isInteger(3)) // affiche true
```

L'exercice suivant est beaucoup plus compliqué que les précedents. Faites-le seulement si vous en avez le temps et l'envie.

Exercice 6

Faites un programme pour jouer au pendu.

L'utilisateur a 10 coups pour trouver le mot secret. Vous ne pouvez pas montrer le dessin avec la potence (pour l'instant), mais vous pouvez afficher le nombre de coups restants.

Ce programme est plus facile à programmer si vous enregistrez le mot secret dans un tableau, avec une lettre dans chaque position.