QCM JAVASCRIPT

Q1 : Le **JavaScript** est principalement utilisé en tant que langage :

☐ client-side

☐ server-side

☐ client-side et server-side

Corrigés :

Le JavaScript est un langage conçu pour dynamiser les pages Web après leur téléchargement par les navigateurs, il est donc client-side vu qu'il s'exécute par le biais du navigateur. Cependant, il est possible de l'utiliser dans des applications server-side mais nous n'aborderons pas ce concept dans ce cours.

Q2 : Où dois-je placer de préférence mon **code JavaScript** dans une page web ?

☐ Dans la balise <head> qui est prévue à cet effet.

☐ Entre les balises <head> et <body>.

☐ Juste avant la fin de la balise <body>.

Corrigés :

Pour des raisons de rapidité d'exécution du code, il est préférable de placer votre code JavaScript avant la fin de la balise <body> afin que les vieux navigateurs ne chargent pas le JavaScript avant le contenu de la page Web.

Q3 : Quelle est la bonne syntaxe pour afficher un message à l'écran ?

☐ alert('Hello world!')

☐ alert('Hello world!');

☐ alert 'Hello world!';

☐ alert(Hello world!);

Corrigés :

Seule la deuxième réponse est correcte car la première ne contient pas de point-virgule (il

n'est pas obligatoire, certes, mais il s'agit d'une très mauvaise pratique) et la troisième ne contient pas les apostrophes qui doivent encadrer le texte à afficher.

Q4 : Je viens de réaliser une concaténation, est-elle correcte : **var text = "J'aime " - 'le JavaScript !';** ?

☐ Oui

☐ Non, il faut utiliser le signe + au lieu du - !

☐ Non, car on ne peut pas faire une concaténation dès la déclaration d'une variable.

☐ Non, il faut utiliser les apostrophes sur les deux chaînes de caractères.

Corrigés :

Pour faire une concaténation, il vous faudra toujours utiliser le signe + entre les deux chaînes de caractères à concaténer.

Q5 : Est-il possible de raccourcir la troisième ligne de ce code ?

```
var number1 = 60, number2 = 2;

number1 = number2 + 40;
```

☐ Oui, il suffit d'utiliser l'opérateur +=

☐ Non

Corrigés :

```
number1 = number2 + 40; // Là on n'ajoute rien du tout à "number1", on remplace juste sa valeur actuelle par celle de "number2" additionnée à la valeur 40.

// De ce fait, on ne peut donc pas écrire :

number1 += 40;
```

Q6 : Quel est le résultat de ce code ?

```
var number1 = "2", number2 = "3", resultat;

resultat = number1 + number2;
```

alert(resultat);

☐ -1

☐ 5

☐ 23

☐ Rien, le script rencontre une erreur

Corrigés :

N'oubliez pas : les chaînes de caractères se concatènent avec l'opérateur + même si elles contiennent des chiffres. Si vous voulez faire le calcul, pensez bien à utiliser la fonction parseInt() sur chaque chaîne !

Q7 : Dans quel ordre doit-on voir apparaître ces structures ?

☐ if > else > else if

☐ else if > if > else

☐ if > else if > else

☐ else > else if > if

Corrigés :

L'ordre de ces trois structures est très important et doit toujours être le suivant :

if (condition) {

} else if (condition) {

// Faire un truc

} else {

// Faire un autre truc

}

Q8 : Que va-t-il se passer si je clique sur le **bouton « OK »** dans la fenêtre de confirmation ?

if (!confirm('OK ?')) {

```
    alert("C'est OK !");
}
```

☐ Le message « C'est OK ! » va s'afficher.

☐ La page d'accueil de mon navigateur va s'afficher.

☐ Rien

Corrigés :

La fonction confirm() renvoie true lorsque l'on clique sur le bouton « OK », or j'ai mis l'opérateur NON au début de la condition, donc si vous cliquez sur « OK » confirm() renverra true qui sera ensuite inversé par l'opérateur NON pour obtenir false.

Donc, au final, il ne se passera rien !

Q9 : Quelle est la valeur d'output dans l'instruction suivante : **var output = count++;** ?

☐ count, incrémentée de 1

☐ count, sans incrémentation

☐ Juste l'incrémentation

Corrigés :

Si l'opérateur ++ se trouve après la variable, la valeur de la variable est retournée, et l'incrémentation se fait après. Ici, output contient donc la valeur de count, avant l'incrémentation.

Q10 : Quelle est l'utilité de **break** dans une boucle ?

☐ Il permet de faire une pause

☐ Il permet d'arrêter l'itération en cours et de passer à la suivante

☐ Il permet d'arrêter l'itération en cours et de quitter la boucle

Corrigés :

break sert à mettre fin à l'itération courante, et à sortir de la boucle.

Q11 : Quelle est la particularité d'une boucle **do while** ?

☐ Aucune, c'est la forme longue de la boucle while

☐ Les instructions sont exécutées au moins une fois

☐ La condition n'est exécutée qu'au début de chaque itération

Corrigés :

Les instructions contenues dans une boucle do while sont toujours exécutées au moins une fois. Après cette première exécution, la condition est testée pour savoir si la boucle peut continuer, et celle-ci se comporte alors comme une boucle while normale.

Q12 : Dans une boucle **for**, à quel moment le bloc d'incrémentation est-il exécuté ?

☐ Au début de chaque itération

☐ Pendant chaque itération

☐ À la fin de chaque itération

Corrigés :

Les trois blocs qui constituent la boucle for ne sont pas exécutés en même temps :

Initialisation : juste avant que la boucle ne démarre. C'est comme si les instructions d'initialisation avaient été écrites juste avant la boucle, un peu comme pour une boucle while.

Condition : avant chaque passage de boucle, exactement comme la condition d'une boucle while.

Incrémentation : après chaque passage de boucle. Cela veut dire que si vous faites un break dans une boucle for le passage dans la boucle lors du break ne sera pas compté.

Q13 : À quoi servent les fonctions ?

☐ À rien

☐ À se passer des boucles

☐ À n'écrire qu'une seule fois un même code pour ensuite l'appeler où on le souhaite

☐ À exécuter un code provenant d'un autre site Web

Corrigés :

Les fonctions servent effectivement à n'écrire qu'une seule fois le même code pour pouvoir le réexécuter quand on le souhaite et sans contraintes. Les fonctions sont d'autant plus avantageuses avec le système des arguments et des retours de valeurs.

Q14 : Quelle est la différence entre une variable globale et une variable locale ?

☐ La locale est accessible partout dans le code tandis que la globale est limitée à la fonction où elle est déclarée

☐ La globale est accessible partout dans le code tandis que la locale est limitée à la fonction où elle est déclarée

☐ Les globales et locales sont identiques tant qu'elles ne sont pas déclarées dans une boucle ou une condition

Corrigés :

Les variables globales sont toutes les variables déclarées en-dehors des fonctions, elles sont accessibles partout dans votre code, y compris dans les fonctions, tandis que les variables locales ne sont accessibles que dans la fonction où elles ont été déclarées, et nulle part ailleurs.

Q15 : Quelles sont les différences entre une fonction classique et une fonction anonyme ?

☐ Une seule différence : la fonction anonyme ne possède pas de nom.

☐ La fonction anonyme ne possède pas de nom et est, par conséquent, bien plus rapide à l'exécution.

☐ Les fonctions anonymes servent juste à isoler du code, contrairement aux fonctions classiques qui servent pour de nombreuses autres utilisations.

Corrigés :

Les fonctions anonymes n'ont, effectivement, que l'absence de nom pour seule différence avec les fonctions classiques.

Q16 : Si je veux accéder au **troisième item** d'un tableau, quel indice dois-je utiliser ?

☐ 4

☐ 3

☐ 2

Corrigés :

La numérotation des indices commence à 0. Donc, pour le troisième item, l'indice sera 2.

Q17 : Pour ajouter un item dans un objet littéral, il faut utiliser la méthode push()

☐ Vrai

☐ Faux

Corrigés :

C'est faux, la méthode push() n'est utilisable qu'avec les tableaux.

Q18 : Quelle est l'affirmation correcte ?

☐ Une propriété est une variable transmise à un objet

☐ Une méthode est une variable interne à un objet

☐ Une propriété est une fonction interne à un objet

☐ Une méthode est une fonction interne à un objet

Corrigés :

Une méthode est bien sûr une fonction interne à un objet, tandis qu'une propriété est une variable interne à un objet.

Q19 : Le code suivant fonctionne-t-il ?

```
function test() {
  if (true) { var a = "hello"; }
  alert(a);
```

}

☐ Non, erreur de syntaxe

☐Non, alert(a) n'affiche rien

☐ Oui, mais c'est vraiment moche

☐ Oui, et c'est une bonne idée

Corrigés :

Oui, ça fonctionne. La portée d'une variable déclarée avec var est locale à la fonctionne, et non au bloc if/else. Mais c'est vraiment une mauvaise pratique : préférez déclarer vos variables avec var dès le début de la fonction.