

**INF3054L – LIFWEB****Printemps 2023–2024**

Ce contrôle est un QCM corrigé **automatiquement**. Répondre au stylo bille foncé en **noircissant complètement** la case correspondante sur **la feuille de réponse fournie**.

Les questions **avec** le symbole ♣ peuvent présenter *zéro, une ou plusieurs bonnes réponses*. Ces questions sont notées sur 2 points avec -1 point par réponse fausse (0 minimum par question).

Les questions **sans** le symbole ♣ ont une *unique bonne réponse*. Ces questions sont notées sur 2 points avec 2 points si la réponse est correcte, 0 s'il n'y a pas de réponse et -0.5 si la réponse est fausse. Le barème des questions ouvertes est précisé sur chacune.

1 Bonnes pratiques JavaScript

On considère l'extrait de code ci-dessous.

```

1 function positiveOnlyDecorator(f) {
2     let res;
3     return function (arg) {
4         if (!isNaN(arg) && arg > 0) {
5             res = f(arg);
6             return res;
7         } else return undefined;
8     };
9 }
```

Les erreurs suivantes à propos de `positiveOnlyDecorator()` sont reportées :

Line 2 Please rename the variable `res`.

Line 3 The variable `arg` should be named `argument`.

Line 4 This `if` statement can be replaced by a ternary expression.

Line 4 Prefer `Number.isNaN` over `isNaN`

Line 7 Do not use useless `undefined`

Question 1 (2) Quelle est **la meilleure pratique** concernant la déclaration de variables ?

- [A] Utiliser `var` pour toutes les déclarations de variables pour garantir la compatibilité avec les anciennes versions de JavaScript.
- [B] Utiliser exclusivement `const` pour toutes les déclarations de variables, car cela améliore les performances du code.
- [C] La différence entre `var`, `let` et `const` est purement stylistique et n'affecte pas le comportement du code.
- [X] Utiliser `let` pour les variables dont la valeur peut changer et `const` pour les autres.

Question 2 ♣ (2) Quelles affirmations sont des bonnes pratiques de prévention des attaques XSS (*Cross-Site Scripting*) en JavaScript ?

- [X] Privilégier l'utilisation de `textContent` pour affecter du texte à un élément du DOM plutôt que d'utiliser `innerHTML`.
- [B] Utiliser `innerHTML` pour insérer du contenu HTML externe car il est automatiquement nettoyé par le navigateur.
- [X] Éviter d'insérer directement dans le DOM des contenus provenant de sources non fiables sans les avoir préalablement nettoyés.
- [X] Utiliser des bibliothèques ou des fonctions spécifiques de nettoyage des données pour traiter le contenu externe avant son insertion dans le DOM.

Question 3 (2) L'outil ESLint sert **principalement** à mettre en forme le code JavaScript.

- [A] Vrai.
- [X] Faux.



Question 4 (/2) Expliquer le message d'erreur *This if statement can be replaced by a ternary expression* ligne 4 de `positiveOnlyDecorator()`.

Question 5 (/6) Réécrire la fonction `positiveOnlyDecorator()` pour qu'il n'y ait plus aucun message d'erreur.

2 Programmation fonctionnelle

Question 6 ♣ (/2) Indiquer quels sont les styles de programmation utilisables en JavaScript.

- Événementiel.
- Fonctionnel.
- Objet.
- Impératif.

Question 7 (/2) Quelle est la description correcte d'une fermeture (*closure*) en JavaScript ?

- A Une fonction qui peut être passée comme argument à une autre fonction.
- C Un modèle de conception qui permet de créer des objets sans spécifier la classe exacte de l'objet qui sera créé.
- B Une fonction qui retient les variables de son périmètre lexical, même après que la fonction externe ait terminé son exécution.
- D Une fonction qui est renvoyée par une autre fonction, mais sans retenir aucune référence à son environnement lexical.

Question 8 ♣ (/2) Quel est un usage courant des fermetures (closures) en JavaScript ?

- A Pour augmenter la vitesse d'exécution du code en mémorisant les résultats des fonctions.
- B Pour empêcher le ramasse-miettes de JavaScript de libérer la mémoire utilisée par les fonctions inactives.
- C Pour maintenir un état dans des fonctions asynchrones ou des gestionnaires d'événements.
- D Pour créer des fonctions privées dans les modules ou objets, en exploitant la portée des fonctions.

Question 9 (/2) On utilise le *comma operator* (,) dans le code `const r = console.log((1, 2));`. Quel est le résultat de l'évaluation de cette expression ?

- A `r == undefined` et affiche 2.
- B `r == 2` et affiche 1.
- C `r == 2` et affiche 2.
- D `r == 1` et affiche 1.
- E `r == undefined` et affiche 1.
- F `r == 1` et affiche 2.

Question 10 (/2) Soit le tableau `tab = [1,2,3,4,5,6,7,8,9]`, donner le résultat de l'évaluation de `tab.some(n => n %3 === 0)&& tab.some(n => n % 5 === 0);`

- A `false`
- B *Cet appel produit une erreur à l'exécution.*

Question 11 (/2) Soit le tableau `tab = [1,2,3,4,5,6,7,8,9]`, donner le résultat de l'évaluation de `tab.filter(n => n % 2 === 0).map(n => n*n).reduce((acc,n2)=> acc+n2, 0);`

- A 20
- B *Cet appel produit une erreur à l'exécution.*
- C 42
- D 120

Question 12 (/4) Donner un nom raisonnable à la fonction suivante en s'appuyant sur ce qu'elle calcule : `let fred = (arr)=> arr.reduce((acc,x)=> acc || x, false);`



3 JavaScript asynchrone

On considère les extraits de code ci-dessous.

```
1 function delay(func, args, wait = 1000, error) {
2   return new Promise((resolve, reject) => {
3     setTimeout(() => {
4       if (error === undefined) {
5         resolve(func(...args));
6       } else {
7         reject(error);
8       }
9     }, wait);
10  });
11}
12 const addB = (x) => `${x}-B`;
```

```
1 /* The reduceRight() method of Array instances applies a function against
2    an accumulator and each value of the array (from right-to-left) to
3    reduce it to a single value.
4
5 const a = ["1", "2", "3", "4", "5"];
6 const right = a.reduceRight((prev, cur) => prev + cur);
7 console.log(right); // "54321" */
8 const ex = (x, y) => x.reduceRight((q, p) => q.then(p), Promise.resolve(y))
9 ;
10
11 const arr = [1, 2, 3].map((x) => (y) => y + x);
```

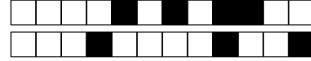
Question 13 ♣ (2) Expliquer simplement, sans périphrase, ce que fait la fonction `delay()`.

Question 14 ♣ (2) Quelles affirmations suivantes concernant l'utilisation de `async/await` sont correctes ?

- A Le mot-clé `async` devant une fonction indique que la fonction retournera une promesse.
- B Utiliser `await` dans une fonction non `async` est valide et ne génère pas d'erreur.
- C Les mots-clés `async` et `await` ne peuvent être utilisés qu'avec des fonctions génératrices.
- D Le mot-clé `await` permet d'attendre la résolution d'une promesse à l'intérieur d'une fonction `async`.

Question 15 ♣ (2) Quelles affirmations suivantes concernant les promesses sont correctes ?

- A Les promesses permettent de gérer des opérations asynchrones en fournissant une manière plus propre d'écrire du code asynchrone.
- B Il est recommandé d'utiliser des *callbacks* au lieu des promesses pour une meilleure gestion des erreurs asynchrones.
- C Une fois une promesse résolue ou rejetée, son état peut changer si une autre opération asynchrone la modifie.
- D La méthode `.then()` d'une promesse est utilisée pour spécifier quoi faire une fois la promesse résolue.



Question 16 ♣ (/2) Soit la fonction JavaScript `fa` définie ci-dessous:

```
1 function fa() {  
2   delay(addB, "A", 500)  
3     .then(console.log)  
4     .catch(console.error);  
5   delay(addB, "B", 1000)  
6     .then(console.log)  
7     .catch(console.error);  
8 }
```

On exécute `fa()`; au temps t_0 , indiquer ce qui s'affiche dans la console :

- A "BB" à $t_0 + 1500\text{ms}$ B "AB" à $t_0 + 1500\text{ms}$
 C "BB" à $t_0 + 1000\text{ms}$ D "AB" à $t_0 + 500\text{ms}$

Question 17 ♣ (/2) Soit la fonction JavaScript `fb` définie ci-dessous:

```
1 function fb() {  
2   delay(addB, "A", 500)  
3     .then(console.log)  
4     .then(() => delay(addB, "B", 1000))  
5     .then(console.log)  
6     .catch(console.error);  
7 }
```

On exécute `fb()`; au temps t_0 , indiquer ce qui s'affiche dans la console :

- A "BB" à $t_0 + 1000\text{ms}$ B "AB" à $t_0 + 500\text{ms}$
 C "BB" à $t_0 + 1500\text{ms}$ D "AB" à $t_0 + 1500\text{ms}$

Question 18 ♣ (/2) Soit la fonction JavaScript `fc` définie ci-dessous:

```
1 function fc() {  
2   delay(addB, "A", 500, new Error("..."))  
3     .then(console.log)  
4     .catch(console.error);  
5   delay(addB, "B", 1000)  
6     .then(console.log)  
7     .catch(console.error);  
8 }
```

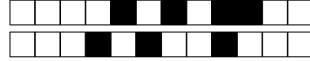
On exécute `fc()`; au temps t_0 , indiquer ce qui s'affiche dans la console :

- A "Une erreur" à $t_0 + 500\text{ms}$ F "BB" à $t_0 + 500\text{ms}$
 B "Une erreur" à $t_0 + 1000\text{ms}$ G "BB" à $t_0 + 1000\text{ms}$
 H "AB" à $t_0 + 1000\text{ms}$ I "Une erreur" à $t_0 + 1500\text{ms}$
 D "BB" à $t_0 + 1500\text{ms}$ J "AB" à $t_0 + 1500\text{ms}$
 E "AB" à $t_0 + 500\text{ms}$

Question 19 (/2) Donner le résultat de l'exécution de `ex(arr, 42).then(console.log);`.

- A 42 B 48
 C [Function (anonymous)] D *Cet appel produit une erreur à l'exécution.*

Question 20 (/6) Réécrire la fonction `ex` en remplaçant `reduce` par une boucle utilisant les promesses (sans `async/await`).



4 Node.js

Question 21 (/2) En Node.js, quel mécanisme permet de gérer des événements asynchrones, comme la fin de la lecture d'un fichier ?

- Utiliser l'objet `EventEmitter` du module `events` pour écouter et émettre des événements.
- B Appeler directement les méthodes du module `async` sans avoir besoin d'un système d'événements.
- C Créer une promesse qui résout automatiquement lorsqu'un fichier est entièrement lu, sans utiliser d'événements.
- D Utiliser une boucle `for` pour vérifier continuellement l'état du fichier jusqu'à sa complète lecture.

Question 22 ♣ (/2) Quel est le rôle du fichier `package.json` dans un projet Node.js ?

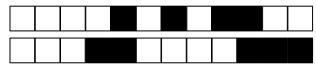
- Il peut spécifier des scripts pour automatiser des tâches courantes comme le démarrage du serveur ou les tests.
- Il sert de manifeste pour le projet, définissant les dépendances, scripts, et métadonnées du projet.
- C Il est utilisé pour stocker le code source JavaScript du projet, organisé en modules.
- Il configure l'environnement d'exécution de Node.js, comme la version de Node.js à utiliser.

Question 23 (/2) En Node.js, quand on utilise `fetch()`, on est soumis à la politique de sécurité CORS (*Cross-Origin Resource Sharing*).

- Faux.
- Vrai.

Question 24 (/2) Node.js est particulièrement adapté pour les applications nécessitant un traitement intensif du processeur.

- Faux.
- Vrai.



+172/6/7+



Feuille de réponses à compléter

← codez votre numéro d'étudiant à 8 chiffres
et écrivez votre nom et prénom ci-dessous.

Nom et prénom :

Question 1 : A B C D

Question 2 : A B C D

Question 3 : A B

Question 4 :

F I P J

La condition peut être posée autrement, car l'unique action qu'on fait à l'intérieur du if est de renvoyer ce résultat

Question 5 :

F I P J

```
function positiveOnly(decorateur(f)) {  
    const res;  
    return function(...arguments) {  
        if (!Number.isNaN(arguments[0]) && arguments[0] > 0) {  
            res = f(...arguments);  
        } else {  
            res = undefined;  
        }  
        return res;  
    };  
}
```

Question 6 : A B C D

Question 7 : A B C D

Question 8 : A B C D

Question 9 : A B C D E F



Question 10 : B

Question 11 : B C

Question 12 :



Selection - élément - présent



Question 13 :



Elle prend une fonction en paramètre qui retourne une promesse après un delay de 1 seconde

Question 14 : B C

Question 15 : C

Question 16 : A

Question 17 : A

Question 18 : B D E F H

Question 19 : A B D

Question 20 :



```
async ex () {
  try {
    const response = response.json()
      .then(console.log);
  }
  catch (err) {
    throw new Error ("Erreur");
  }
}
```

Question 21 : B C D

Question 22 : C

Question 23 :

Question 24 :