

TEST DE CONNAISSANCE

correction



1. Les variables sont comme des boîtes, parce que...
 - a) On peut les ouvrir
 - b) On peut mettre des choses dedans
 - c) On peut stocker des choses dedans

Explication

On peut les ouvrir, on peut mettre des choses dedans, mais ce qui nous intéresse vraiment, c'est de pouvoir stocker des données. Comme tu le ferais si tu voulais stocker ta collection de cartes Pokémon dans une boîte pour plus tard.

2. Qu'est-ce que le JavaScript ?

A.1 Le JavaScript est un langage de programmation populaire utilisé pour créer des fonctionnalités interactives sur les sites web.

A.2 Le JavaScript est un logiciel utilisé pour coder des applications mobiles.

A.3 Le JavaScript est un framework utilisé pour le développement de jeux vidéo.

A.4 [Correct] Le JavaScript est un langage de programmation utilisé pour ajouter de l'interactivité et du dynamisme aux sites web.

3. Les outils pour coder le JavaScript :

A.1 Les outils pour coder en JavaScript incluent Visual Studio Code, Sublime Text et Atom.

A.2 Les outils pour coder en JavaScript incluent Photoshop, Illustrator et Sketch.

A.3 Les outils pour coder en JavaScript incluent MySQL, MongoDB et PostgreSQL.

A.4 [Correct] Les outils pour coder en JavaScript incluent Visual Studio Code, Sublime Text et Atom.

4. Où s'écrit ce JavaScript ?

A.1 Le JavaScript s'écrit uniquement dans les fichiers HTML.

A.2 Le JavaScript s'écrit uniquement dans les fichiers CSS.

A.3 Le JavaScript s'écrit dans des fichiers séparés avec l'extension ".js" ou directement dans les balises <script> dans les fichiers HTML.

A.4 [Correct] Le JavaScript s'écrit dans des fichiers séparés avec l'extension ".js" ou directement dans les balises <script> dans les fichiers HTML.

5. Les commentaires :

A.1 Les commentaires en JavaScript sont ignorés par le navigateur et ne servent qu'à des fins de documentation.

A.2 Les commentaires en JavaScript sont utilisés pour exécuter des parties spécifiques du code.

A.3 Les commentaires en JavaScript sont utilisés pour masquer des parties du code que l'on ne souhaite pas exécuter.

A.4 [Correct] Les commentaires en JavaScript sont ignorés par le navigateur et ne servent qu'à des fins de documentation.

6. La syntaxe :

A.1 La syntaxe JavaScript est basée sur des instructions écrites uniquement en majuscules.

A.2 La syntaxe JavaScript suit des règles strictes et utilise des points-virgules à la fin de chaque instruction.

A.3 La syntaxe JavaScript ne permet pas l'utilisation de parenthèses.

A.4 [Correct] La syntaxe JavaScript suit des règles strictes et utilise des points-virgules à la fin de chaque instruction.

7. Les variables :

- A.1 Les variables en JavaScript ne peuvent pas être modifiées une fois qu'elles ont été déclarées.
- A.2 Les variables en JavaScript sont déclarées avec le mot-clé "const".
- A.3 Les variables en JavaScript permettent de stocker et de manipuler des données.
- A.4 [Correct] Les variables en JavaScript permettent de stocker et de manipuler des données.

8. La concaténation :

- A.1 La concaténation en JavaScript est utilisée pour effectuer des calculs mathématiques.
- A.2 La concaténation en JavaScript est utilisée pour fusionner des chaînes de caractères.
- A.3 La concaténation en JavaScript est utilisée pour ajouter des éléments à un tableau.
- A.4 [Correct] La concaténation en JavaScript est utilisée pour fusionner des chaînes de caractères.

9. Les types de données :

- A.1 En JavaScript, il n'existe qu'un seul type de données : le nombre.
- A.2 Les types de données en JavaScript comprennent les nombres, les chaînes de caractères, les booléens et les tableaux, entre autres.
- A.3 Les types de données en JavaScript sont définis uniquement par le développeur.
- A.4 [Correct] Les types de données en JavaScript comprennent les nombres, les chaînes de caractères, les booléens et les tableaux, entre autres.

10. Les opérations/opérations d'affectation :

- A.1 Les opérations d'affectation en JavaScript sont utilisées pour comparer des valeurs.
- A.2 Les opérations d'affectation en JavaScript sont utilisées pour créer des boucles.
- A.3 Les opérations d'affectation en JavaScript sont utilisées pour attribuer une valeur à une variable.
- A.4 [Correct] Les opérations d'affectation en JavaScript sont utilisées pour attribuer une valeur à une variable.

11. Les structures de contrôle :

- A.1 Les structures de contrôle en JavaScript permettent d'effectuer des opérations

mathématiques complexes.

A.2 Les structures de contrôle en JavaScript permettent de gérer la logique et le flux d'exécution du code, notamment avec les boucles et les conditions.

A.3 Les structures de contrôle en JavaScript permettent de créer des animations et des transitions visuelles.

A.4 [Correct] Les structures de contrôle en JavaScript permettent de gérer la logique et le flux d'exécution du code, notamment avec les boucles et les conditions.

12. Les fonctions :

A.1 Les fonctions en JavaScript sont utilisées uniquement pour afficher des messages à l'utilisateur.

A.2 Les fonctions en JavaScript permettent d'organiser le code en regroupant des instructions spécifiques.

A.3 Les fonctions en JavaScript ne peuvent pas prendre de paramètres.

A.4 [Correct] Les fonctions en JavaScript permettent d'organiser le code en regroupant des instructions spécifiques.

13. La portée des variables :

A.1 En JavaScript, toutes les variables sont globales et peuvent être utilisées n'importe où dans le code.

A.2 La portée des variables en JavaScript détermine leur visibilité et leur accessibilité dans différentes parties du code.

A.3 La portée des variables en JavaScript est définie par les mots-clés "public" et "private".

A.4 [Correct] La portée des variables en JavaScript détermine leur visibilité et leur accessibilité dans différentes parties du code.

14. Mise en application avec un projet :

A.1 La mise en application avec un projet en JavaScript consiste uniquement à créer des animations visuelles.

A.2 La mise en application avec un projet en JavaScript implique la création d'une application web interactive ou d'un jeu.

A.3 La mise en application avec un projet en JavaScript est limitée aux calculs mathématiques complexes.

A.4 [Correct] La mise en application avec un projet en JavaScript implique la création d'une application web interactive ou d'un jeu.

15. Les événements :

- A.1 Les événements en JavaScript sont utilisés exclusivement pour manipuler des images.
- A.2 Les événements en JavaScript permettent de détecter les actions de l'utilisateur, tels que les clics de souris ou les pressions de touches.
- A.3 Les événements en JavaScript sont utilisés pour gérer les connexions à des bases de données.
- A.4 [Correct] Les événements en JavaScript permettent de détecter les actions de l'utilisateur, tels que les clics de souris ou les pressions de touches.

16. Les tableaux :

- A.1 Les tableaux en JavaScript sont utilisés uniquement pour stocker des nombres.
- A.2 Les tableaux en JavaScript permettent de regrouper plusieurs valeurs dans une seule variable.
- A.3 Les tableaux en JavaScript ne peuvent pas être modifiés une fois qu'ils ont été créés.
- A.4 [Correct] Les tableaux en JavaScript permettent de regrouper plusieurs valeurs dans une seule variable.

17. Les objets :

- A.1 Les objets en JavaScript sont utilisés uniquement pour styliser les éléments HTML.
- A.2 Les objets en JavaScript permettent de représenter des entités avec leurs propriétés et leurs méthodes.
- A.3 Les objets en JavaScript ne peuvent pas contenir d'autres objets.
- A.4 [Correct] Les objets en JavaScript permettent de représenter des entités avec leurs propriétés et leurs méthodes.

18. La manipulation du DOM :

- A.1 La manipulation du DOM en JavaScript concerne uniquement la création de styles CSS.
- A.2 La manipulation du DOM en JavaScript permet de modifier la structure et le contenu d'une page web en utilisant des méthodes telles que getElementById ou querySelector.
- A.3 La manipulation du DOM en JavaScript est réservée aux développeurs backend.
- A.4 [Correct] La manipulation du DOM en JavaScript permet de modifier la structure

et le contenu d'une page web en utilisant des méthodes telles que getElementById ou querySelector.

19. Les modules :

- A.1 Les modules en JavaScript sont utilisés exclusivement pour les opérations mathématiques.
- A.2 Les modules en JavaScript permettent d'organiser le code en regroupant des fonctionnalités connexes.
- A.3 Les modules en JavaScript sont obsolètes et ne doivent plus être utilisés.
- A.4 [Correct] Les modules en JavaScript permettent d'organiser le code en regroupant des fonctionnalités connexes.

20. La gestion des erreurs :

- A.1 La gestion des erreurs en JavaScript n'est pas nécessaire car le langage est exempt de bugs.
- A.2 La gestion des erreurs en JavaScript implique l'utilisation de la déclaration try-catch pour détecter et gérer les erreurs.
- A.3 La gestion des erreurs en JavaScript est réservée aux développeurs expérimentés.
- A.4 [Correct] La gestion des erreurs en JavaScript implique l'utilisation de la déclaration try-catch pour détecter et gérer les erreurs.

21. Les frameworks populaires :

- A.1 Les frameworks populaires en JavaScript incluent React, Angular et Vue.js.
- A.2 Les frameworks populaires en JavaScript sont utilisés exclusivement pour le développement de jeux vidéo.
- A.3 Les frameworks populaires en JavaScript sont obsolètes et ne sont plus utilisés.
- A.4 [Correct] Les frameworks populaires en JavaScript incluent React, Angular et Vue.js.

22. Quelle est l'utilité d'une variable ?

- a) On peut manipuler des données avec**
- b) On peut supprimer des données**
- c) On peut stocker des données**

Explication

Le but principal est de stocker et manipuler des données

23.

24. Quelle variable permet de stocker puis modifier des données ?

- a) let**
- b) const**

Explication : C'est let

25.

26. Quelle variable permet de stocker mais n'autorise pas à modifier des données ?

- a) let**
- b) const**

Explication

Cette fois c'est const 😊

Et var dans tout ça ?

Effectivement, il est aussi possible de déclarer des variables avec var au lieu de let.

Mais var est maintenant dépréciée et ne devrait plus être utilisée.

C'est quoi déprécié ?

var a les mêmes caractéristiques que let, mais il y a une différence importante :

- var est globale alors que let est locale (on verra ça plus tard)**
- var peut être déclarée plusieurs fois, alors que let ne peut être déclarée qu'une seule fois (ce qu'on a vu avant)**
- var peut être utilisé avant d'être déclaré, alors que let ne peut pas (on le verra plus tard)**

27.dd

28.dd

29. Comme déclarer une variable de type number qu'on peut modifier ?

- a) `const age = 12`
- b) `var age = '12'`
- c) `var age = 12`
- d) `age = 12`
- e) `let age = 12`

Explication

la syntaxe avec `var` est une bonne réponse, malgré que ce ne soit pas recommandé. La vraie bonne réponse, c'est la dernière, avec `let`.

30. Quel est le type de donnée utilisé pour déclarer à la fois des nombres entiers et à virgule en JavaScript ?

- a) `boolean`
- b) `number`
- c) `string`
- d) `undefined`

Explication

En JavaScript, le type `'number'` est utilisé pour déclarer à la fois des nombres entiers et à virgule.

31. Comment convertit-on une chaîne en nombre en JavaScript ?

- a) `Number()`
- b) `.toNumber()`
- c) `number()`
- d) `+` devant une variable
- e) `parseInt()`

Explication

`.toNumber()` n'existe pas et `number` s'écrit avec une majuscule.

32. Quelle est la valeur spéciale en JavaScript qui signifie 'Pas un Nombre' ?

- a) `null`

b) undefined

c) NaN

d) false

Explication

En JavaScript, NaN est une valeur spéciale qui signifie 'Not a Number'.

33.dddd

34. Que renvoie la méthode `Number.isNaN()` si la valeur convertie en Number n'est pas un nombre ?

a) true

b) false

c) NaN

d) undefined

Explication

La méthode `Number.isNaN` renvoie 'true' si la valeur convertie en Number n'est pas un nombre. Car le nombre sera NaN !

35. La condition est true ou false :

```
if (0) {  
    // si la condition est vraie  
} else {  
    // si la condition est fausse  
}
```

a) true

b) false

Explication

Elle est false, car 0 est considéré comme falsy !

Ici, ce qui arrive, c'est la logique de falsy et de truthy qui rentre en jeu, et c'est un concept très important en programmation.

Tu peux retrouver la liste de toutes les valeurs considérées comme falsy par JavaScript et truthy ici :

- [falsy](#)
- [truthy](#)

En résumé, toutes les valeurs qui ne sont pas considérées comme falsy sont considérées comme truthy. Voici la liste des valeurs falsy :

- false (le booléen)
- 0 (le nombre)
- NaN (le nombre)
- "" (la chaîne vide)
- "" (la chaîne vide)
- null (la valeur null)... on la verra plus tard
- undefined (la valeur undefined)... on la verra plus tard

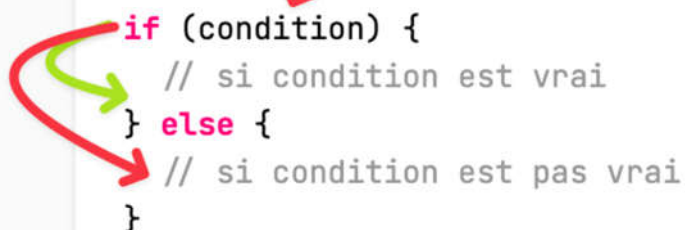
Donc, pour vérifier qu'une valeur n'est pas égale à 0, on peut faire :

JS

```
if (value) {  
    // si la condition est vraie  
}
```

Ici, si value est égale à 0, JavaScript va la transformer en false et donc ne pas rentrer dans notre boucle.

Transforme la condition en boolean



```
if (condition) {  
    // si condition est vrai  
} else {  
    // si condition est pas vrai  
}
```

On peut le tester nous-même :

```
console.log("=== FALSY ===")

console.log("0", Boolean(0))
console.log("String vide", Boolean(""))
console.log("Null", Boolean(null))
console.log("NaN", Boolean(NaN))

console.log("=== Truthy ===")

console.log("1", Boolean(1))
console.log("String PAS vide", Boolean("a"))
console.log("Eva", Boolean({ } ))
```

- === FALSY ===
- 0 false
- String vide false
- Null false
- NaN false
- === Truthy ===
- 1 true
- String PAS vide true
- Eva true

36.Vrai ou faux...

JS

```
const variable = 4;
```

```
if (variable > 3 && variable < 5) {
```

```
// ...
```

```
}
```

a) vrai

b) faux

Explication

Ici c'est vrai, la variable est plus grande que 3 et plus petite que 5.

37. Vrai ou faux...

JS

```
const variable = 4;
```

```
if (variable && variable > 3) {
```

```
  // ...
```

```
}
```

a) vrai

b) faux

Explication

Ici c'est vrai, la variable est 'truthy' et la variable est plus grande que 3.

38. Vrai ou faux...

JS

```
const variable = 4;
```

```
if (variable && variable < 3) {
```

```
  // ...
```

```
}
```

a) vrai

b) faux

Explication

Ici c'est faux, la variable est 'truthy' mais la variable n'est PAS plus petite que 3.

39. Vrai ou faux...

JS

```
const variable = 1;
```

```
if (variable > 3 || variable < 5) {
```

```
  // ...
```

```
}
```

a) vrai

b) faux

Explication

Vrai, la variable est plus petite que 5.

40. Vrai ou faux...

JS

```
const variable = 4;
```

```
if (variable === 3 || variable < 3 || variable > 1) {
```

```
  // ...
```

```
}
```

a) vrai

b) faux

Explication

Vrai encore ! Comme tu le vois, on peut mixer les opérateurs logiques.

Explication du dernier quiz

Ici, il y a 3 conditions, mais JavaScript va toujours check par "batch" de 2. Les deux premières égales :

variable === 3 : faux variable < 3 : faux

Donc faux || faux = faux

Ensuite il va check le dernier :

`variable > 1 : vrai`

Et ici il va faire : `faux || vrai = vrai`

Et c'est pour ça que la réponse finale sera vrai.

NON (!)

Ce signe, c'est le plus drôle ! C'est ce que j'appelle "le contraire" ! Tu peux mélanger ce NON avec les autres opérateurs logiques.

- `=== => !==`
- `== => !=`

Mais tu peux aussi le mélanger avec des variables, pour qu'au lieu de tester le "truthy" de la variable, tu testes le "falsy" !

JS

```
const name = "John";
```

```
if (!name) {  
    // si name est falsy  
}
```

Ce que fait le `!name` c'est qu'il... transforme name en boolean !

- `name => true`
- `!name => false`

JavaScript est un peu magique, dans d'autres langages, ils te demanderaient de toi-même explicitement dire que tu veux transformer name en boolean. Mais, en JavaScript, il le fait pour toi !

Si on mélange un peu ce qu'on a appris ça donne :

41. Vrai ou faux...

JS

const variable = 4;

if (!variable && variable > 3) {

// ...

}

a) vrai

b) faux

Explication

Faux, la variable est 'truthy' et vu qu'on a un '!' devant, il va la transformer en 'falsy' et donc la condition sera fausse. Pas besoin de check la deuxième condition car un && demande que TOUT soit vrai.

42. Vrai ou faux...

JS

const variable = 4;

if (!variable || variable > 3) {

// ...

}

a) vrai

b) faux

Explication

Vrai, la première condition est false mais la deuxième est vraie. On sait que || n'attend qu'un seul vrai pour être vrai.

43. Vrai ou faux...

JS

```
const variable = 104;
```

```
if (variable || (variable > 1 && variable !== 104)) {
```

```
// ...
```

```
}
```

a) vrai

b) faux

Explication

Vrai, la première condition est vraie, on peut déjà s'arrêter. Tout le reste est inutile.

Ici, le **||** nous fait aller étape par étape, vu que la première condition est vraie, on peut déjà s'arrêter.

Mais comment ça se passe si la condition est fautive, par exemple :

JS

```
const variable = 104;
```

```
if (!variable || (variable > 1 && variable !== 104)) {
```

```
// ...
```

```
}
```

Ici, on va aller voir la deuxième condition, qui est vraie... MAIS... cette condition est suivie d'un **&&**, ce qui nous oblige à vérifier que la 3ème condition est aussi vraie... sinon la deuxième ne l'est plus 😊

Quand tu utilises **&&** JavaScript est obligé de tester les deux côtés du **&&**, ce qui va donner :

```
false || true && false => false || false => false
```

J'espère que c'est pas trop flou.

44. Quel est le résultat ?

TS

```
console.log(4 == "4" || 4 === "4");
```

a) true

b) false

Explication

True, car le premier double égal va transformer la string 4 en number, et donc retourner vrai. Le reste devient donc inutile.

45. Quel signe privilégier pour faire une condition ?

a) ==

b) ===

Explication

Le triple égal est toujours à privilégier !

46. Comment traduire cette condition :

JS

```
if (name.length > 5 || name.length > 2 && name[0] === "B")
```

a) Si name possède plus de 5 caractères et qu'il possède plus de 2 caractères et qu'il commence par B

b) Si name possède plus de 5 caractères ou qu'il possède plus de 2 caractères et qu'il commence par B

c) Si name possède plus de 5 caractères et qu'il possède plus de 2 caractères ou qu'il commence par B

Explication

La deuxième phrase utilise les bons termes entre 'ou' et 'et'.

47. Comment traduire cette condition :

JS

if (name && name === "Melvyn" || name === "Baptiste" && name.length > 3 && name.length < 10)

- a) Si name est défini et qu'il est égal à Melvyn ou Baptiste ou qu'il possède entre 3 et 10 caractères.
- b) Si name est défini et qu'il est égal à Melvyn et à Baptiste ou qu'il possède entre 3 et 10 caractères.
- c) **Si** name est défini et qu'il est égal soit à Melvyn ou Baptiste et qu'il possède entre 3 et 10 caractères.
- d) Si name est défini et qu'il est égal soit à Melvyn ou Baptiste ou qu'il possède entre 3 et 10 caractères.

Explication

La 3ème phrase est correcte !

48. Comment transformer une string en minuscule ?

- a) .toUpperCase()
- b) .ToUpperCase()
- c) .toUpper()
- d) **.toLowerCase()**
- e) .ToLowerCase()
- f) .ToLower

Explication

C'est .toLowerCase() !

49. Quels runtimes existent ?

- a) **v8**
- b) JavaScriptCore
- c) chrome
- d) node

Explication

V8 est le runtime de Chrome, mais Chrome n'est pas un runtime.

V8 est un moteur d'exécution JavaScript open source développé par Google. Il est utilisé dans plusieurs environnements d'exécution, notamment dans le navigateur Chrome et dans le runtime Node.js. V8 est réputé pour sa rapidité et sa performance, et il joue un rôle essentiel dans l'exécution du code JavaScript dans ces environnements.

50. Comment vérifier qu'un nombre (n) est égal à NaN ?

- a) `Number.ISNaN(n)`
- b) `Number.IsNaN(n)`
- c) `Number.isNaN(n)`
- d) `NaN === n`
- e) `NaN == n`
- f) `n.isNaN()`

Explication

La méthode `Number.isNaN(n)` est une fonction statique fournie par JavaScript qui permet de vérifier si une valeur est de type NaN (Not-a-Number).

NaN est une valeur spéciale en JavaScript qui représente l'absence d'une valeur numérique valide. Il est retourné lorsqu'une opération mathématique échoue ou lorsque la valeur d'un nombre ne peut pas être représentée en tant que nombre valide.

La méthode `Number.isNaN(n)` renvoie `true` si la valeur passée en argument est de type NaN, sinon elle renvoie `false`. Elle est préférée à l'ancienne fonction `isNaN(n)` car elle ne convertit pas la valeur en un nombre avant de vérifier si elle est NaN. Cela signifie que `Number.isNaN(n)` renverra `false` pour toute valeur autre que NaN, y compris les types non-numériques tels que les chaînes de caractères, les objets, les tableaux, etc.

Donc, pour vérifier si un nombre n est égal à NaN, vous pouvez utiliser Number.isNaN(n) et si cela renvoie true, cela signifie que n est NaN.

51. Combien de paramètres prend cette fonction ?

JS

```
const showWarningMessage = (title, message) => {  
  console.log(title)  
  console.log(message)  
}
```

- a) 0**
- b) 1**
- c) 2**
- d) 3**

Explication

title et message sont les paramètres de la fonction, il y en a donc 2

52. Ce code est-il valide ?

JS

```
const showWarningMessage = (title) => { console.log(title) }
```

- a) Oui**
- b) Non**

Explication

Oui, les accolades sont présentes même si elles ne font pas de retours à la ligne.

53. Soit le script suivant

```
const APP = () => {  
  const currentValue = 0;  
  const newValue = 0;  
  const willRender = currentValue === newValue;  
  if (willRender) {  
    return (  
      <div className='p-4 bg-green-300'>  
        The APP Will Render  
      </div>  
    );  
  } else {  
    return (  
      <div className='p-4 bg-red-300'>  
        The APP Will-Not Render  
      </div>  
    );  
  }  
};  
export default APP;
```

a) The APP Will Render

b) The APP Will- Not Render

la variable `willRender` est définie en comparant `currentValue` et `newValue` à l'aide de l'opérateur de comparaison `===`. Si `currentValue` est égal à `newValue`, alors `willRender` sera évalué à `true`.

Dans le cas où `willRender` est évalué à `true`, le composant retourne un élément `<div>` avec la classe CSS `'p-4 bg-green-300'` et le texte "The APP Will Render". Cela signifie que si `currentValue` et `newValue` sont égaux, le composant affichera le message "The APP Will Render" avec un fond vert.

54. Soit le script suivant

```
const APP = () => {
  const currentValue = 0;
  const newValue = 0;
  const willRender = currentValue === newValue;
  const renderDiv = willRender
    ? '<div class="p-4 bg-green-300">The APP Will Render</div>'
    : '<div class="p-4 bg-red-300">The APP Will Not Render</div>';
  return renderDiv;
};
export default APP;
```

c) The APP Will Render

d) The APP Will- Not Render

55. Soit le script suivant

```
const a = 'Myra';
const b = 'Myra';

if (a === b) {
  console.log('TRUE');
} else {
  console.log('FALSE');
}
```

a) TRUE

b) FALSE

56. Soit le script suivant

```
const a = { username: 'Myra' };
const b = { username: 'Myra' };
if (a.username === b.username) {
```

```

console.log('TRUE');
} else {
  console.log('FALSE');
}

```

a) TRUE

b) FALSE

57. Soit le script suivant

```

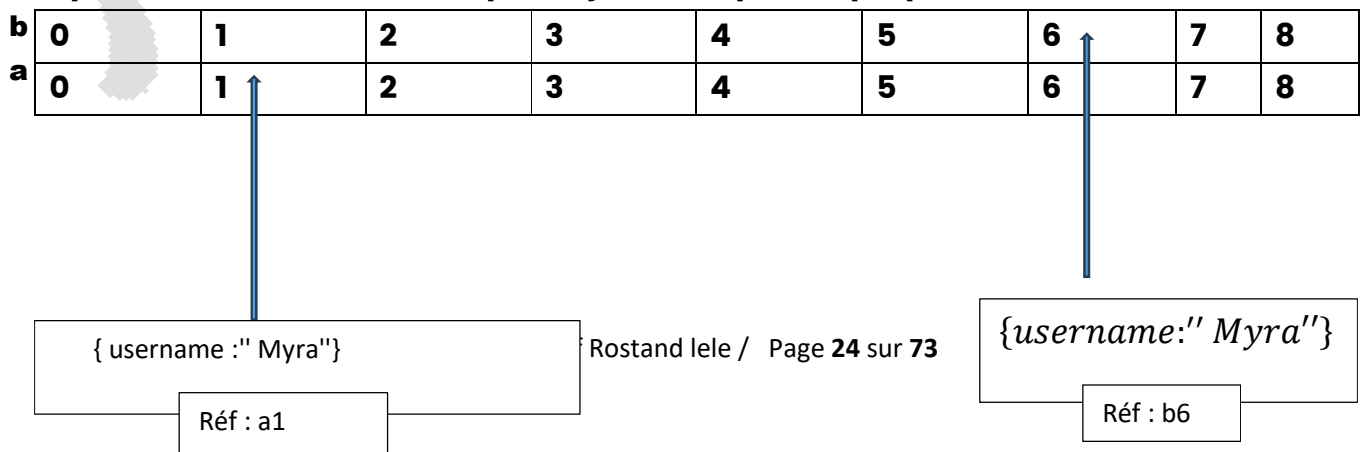
const a = { username : 'Myra' };
const b = { username : 'Myra' };
if (a === b) {
  console.log('TRUE');
} else {
  console.log('FALSE');
}

```

a) TRUE

b) FALSE

Explication : False, car chaque objet crée par sa propre référence



A1===b6 =false

Chemin a rue 6

Chemin b rue 1

58. Soit le script suivant

```
const a = ['Myra'];
```

```
const b = ['Myra'];
```

```
if (a === b) {
```

```
  console.log('TRUE');
```

```
} else {
```

```
  console.log('FALSE');
```

```
}
```

a) TRUE

b) FALSE

59. Soit le script suivant

```
const a = "10";
```

```
const b = 10;
```

```
if (a == b) {
```

```
  console.log('TRUE');
```

```
} else {
```

```
  console.log('FALSE');
```

```
}
```

a) TRUE

b) FALSE

60. Soit le script suivant

```
const valeur1 = { "current": 0 };  
const valeur2 = valeur1;  
const valeur3 = valeur2;  
valeur3.current = 1;  
console.log({ valeur1, valeur2, valeur3 });
```

- a) Valeur 1=0 ; valeur 2=0 ; valeur 3=1
- b) Valeur1=1 ; valeur 2=0 ; valeur3=1
- c) Valeur 1=102 ; valeur 2=1 ; valeur3=1

61. Soit le script suivant

```
let valeur1 = 0;  
let valeur2 = valeur1;  
let valeur3 = valeur2;  
valeur3 = 102;  
console.log({ valeur1, valeur2, valeur3 });
```

- a) Valeur 1=0 ; valeur 2=102 ; valeur 3=0 ;
- b) **Valeur1=0** ; valeur 2=0 ; valeur3=102 ;
- c) Valeur 1=102 ; valeur 2=0 ; valeur3=0 ;

62. À quoi sert le mot-clé return ?

- a) Arrête la fonction**
- b) Retourne une fonction
- c) Retourne une variable
- d) Retourne 'rien' (undefined)
- e) Throw une erreur

Explication

return vient stopper la fonction en plus de retourner une valeur, une fonction ou même

une variable

63.Ce code implémente-t-il le pattern de early return ?

JS

```
const verifyUser = (user) => {  
  if (user.name === "John") {  
    if (user.age > 18) {  
      if (user.email) {  
        return true;  
      }  
    }  
  }  
  return false;  
}
```

- a) Non
- b) Oui

Explication

Non, il n'implémente pas le pattern d'early return car il y a des if imbriqués.

Imagine-toi cette fonction :



```
1 function calculateTotalPrice(quantity, price, isMember, promoCode) {
2   if (quantity > 0) {
3     if (price > 0) {
4       let totalPrice = quantity * price;
5       if (isMember) {
6         totalPrice *= 0.9; // Remise de 10% pour les membres
7         if (promoCode === "SOLDES") {
8           totalPrice *= 0.5; // Remise supplémentaire de 50% avec le code promo
9           return totalPrice;
10        }
11        return totalPrice;
12      } else {
13        return totalPrice;
14      }
15    } else {
16      throw new Error("Le prix unitaire doit être strictement positif");
17    }
18  } else {
19    throw new Error("La quantité doit être strictement positive");
20  }
21  throw new Error("Une erreur est survenue.");
22 }
```

Il y a peu de chances que nous ayons la même réponse ! C'est un peu compliqué à comprendre, il faut lire le code en entier pour comprendre ce qui se passe.

De plus, on évite toujours les if imbriqués.

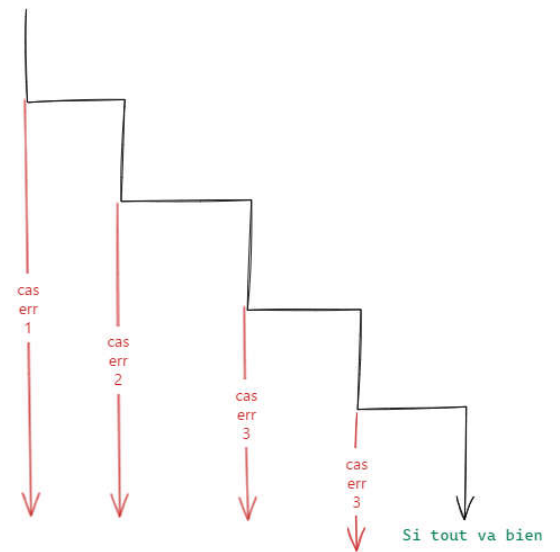
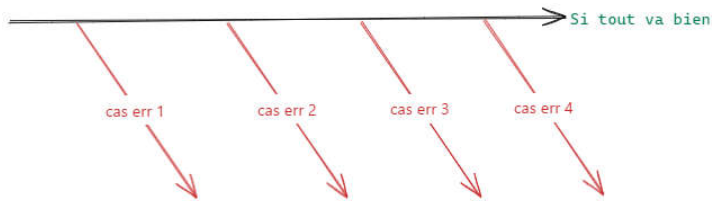
C'est à ce moment que le early return est très pratique, le concept étant de gérer tous les cas d'erreurs en premier, puis de gérer le cas "normal" à la fin.

Essaie de "refactor" la fonction afin d'utiliser le pattern de early return.

Early Return

VS

No Early Return



```
function calculateTotalPrice(quantity, price, isMember, promoCode) {  
  if (quantity <= 0) {  
    throw new Error("La quantité doit être strictement positive");  
  }  
  
  if (price <= 0) {  
    throw new Error("Le prix unitaire doit être strictement positif");  
  }  
  
  let totalPrice = quantity * price;  
  
  if (!isMember) {  
    return totalPrice;  
  }  
  
  totalPrice *= 0.9;  
  
  if (promoCode === "SOLDES") {  
    return totalPrice * 0.5;  
  }  
}
```

```
return totalPrice;  
}
```

64. Quelle valeur sera affichée dans la console ?

JS

```
const demo = (v) => {  
  if (v > 4) {  
    return "Hello";  
    if (v > 5){  
      return "World";  
    }  
  }  
}  
console.log(demo(10));
```

a) Hello

b) World

c) Hello World

d) Rien

Explication

Uniquement Hello sera affiché car après on return et on sort de la fonction.

Je suis méchant avec mes quiz, je sais, le but est de te pousser à la réflexion et de te faire comprendre les subtilités de return et d'early return !

J'espère que tu maîtrises mieux le sujet maintenant.

65. C'est possible ?

JS

```
const names = ["Rostand", "Eva", "Myra"];  
names = ["Jean"]
```

a) Oui

b) Non

Explication

Non, on modifie la référence ici, ce qui va provoquer une erreur à cause du `const` !

66. Soit le script suivant

```
let valeur1 = 0;  
let valeur2 = valeur1;  
let valeur3 = valeur2;  
valeur3 = 102;  
console.log({ valeur1, valeur2, valeur3 });
```

a) Valeur 1=0 ; valeur 2=102 ; valeur 3=0 ;

b) **Valeur1=0** ; valeur 2=0 ; valeur3=102 ;

c) Valeur 1=102 ; valeur 2=0 ; valeur3=0 ;

67.Ce tableau est-il valide ?

JS

```
const names = ["Myra", 1, false];
```

a) **Oui**

b) Non

Explication

Oui ! Les types de tableaux sont dynamiques ce qui permet de mettre des valeurs de types différents.

Méthodes

Les objets possèdent, comme les numbers et les strings, des méthodes.

On a déjà vu `.length` qui permet de récupérer la taille d'un tableau.

Mais on aura aussi :

- `.push(value)` qui permet d'ajouter un élément à la fin d'un tableau
- `.pop(value)` qui permet de supprimer le dernier élément d'un tableau
- `.shift(value)` qui permet de supprimer le premier élément d'un tableau
- `.unshift(value)` qui permet d'ajouter un élément au début d'un tableau

Et plein d'autres que l'on verra plus tard, avec notamment les méthodes de tableaux les plus utilisées `.map`, `.filter` et `.reduce`.

Destructuring

Un concept important des objets et souvent utilisé, c'est le destructuring. En JavaScript, il est facile de récupérer les valeurs d'un tableau dans des variables.

JS

```
const fruits = ["🍏", "🍌", "🍇"];
```

```
const [apple, banana, grape] = fruits;
```

```
// apple = "🍏"
```

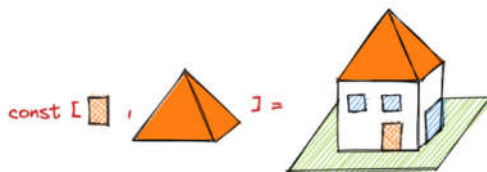
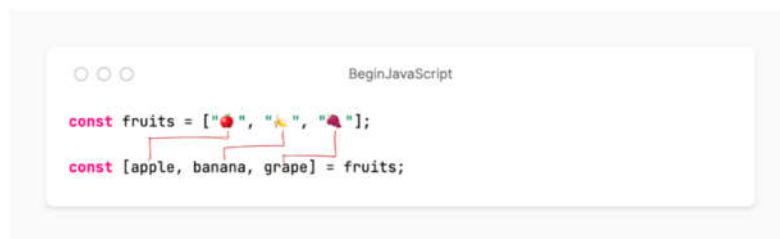
```
// banana = "🍌"
```

```
// grape = "🍇"
```

La syntaxe de structure est la même que la syntaxe de déclaration de tableau, mais on utilise des variables à la place des valeurs.

Chaque valeur va être assignée à la variable correspondante en fonction de leur index.

Tableau



Il y a un autre opérateur très intéressant, c'est le ... qui permet de récupérer le reste des valeurs dans une variable.

Par exemple, si on voulait récupérer uniquement la grape et stocker le reste dans une variable rest :



```
1  const fruits = ["🍎", "🍌", "🍇"];
2
3  const [apple, ...rest] = fruits;
4  // rest = ["🍌", "🍇"]
5  // apple = "🍎"
6
```

Tu vois comme c'est puissant ! On peut faire des choses très intéressantes avec ça. Ce qui est important à noter par rapport à nos références, c'est qu'ici rest est un nouveau tableau !

Pour dupliquer un tableau dans une nouvelle "référence", on peut justement utiliser cette syntaxe :



```
1  const fruits = ["🍎", "🍌", "🍇"];
2
3  const fruitsCopy = [...fruits];
4
5  console.log(fruits === fruitsCopy); // false
6
```

Nomenclature

Petit point très important pour moi. Comment nomme-t-on un tableau ?

On utilise le pluriel !



```
1  const fruits = ["🍎", "🍌", "🍇"];
2  const users = ["Melvyn", "Sarah", "John"];
3  const vegetables = ["🥕", "🥦", "🥒"];
4  const products = ["📱", "🍔", "👟"];
5  const animals = ["🐶", "🐱", "🐭"];
6  const countries = ["Canada", "Espagne", "Italie"];
7  const cities = ["Bafoussam", "Bafou", "Bafang"];
8
```

Voici comment il ne faut pas nommer un tableau :

// X on sait déjà avec `fruits` au pluriel que c'est un tableau

const fruitsList = ["🍎", "🍌", "🍇"];

// X array est redondant

const usersArray = ["Melvyn", "Sarah", "John"];

// X pas de mot français et c'est inutile

const vegetablesTableau = ["🥕", "🥦", "🥒"];

68. brands est-il égal à rest ?

a) Oui

b) Non

Explication

Non, la référence est différente. C'est comme si on avait dupliqué la maison.

69. Comment nommer un tableau qui représente la liste de tous mes produits ?

a) product

b) productList

- c) productList
- d) products
- e) productsArray
- f) productsTableau

Explication

`products` est le seul nom que j'accepte ici.

Dans l'exercice principal de ce module, nous avons utilisé l'impératif dans les parties 1 et 2, puis le déclaratif par la suite.

C'est un concept et des mots importants à comprendre pour la suite :

70. Code impératif et/ou déclaratif ?

```
const numbers = [1, 2, 3, 4, 5];  
const doubledNumbers = [];  
  
for (let i = 0; i < numbers.length; i++) {  
  doubledNumbers.push(numbers[i] * 2);  
}
```

- a) Impératif
- b) Déclaratif

Impératif : On utilise for et on définit ce que l'on veut faire.

71. Code impératif et/ou déclaratif ?

```
const button = document.getElementById('myButton');  
const paragraph = document.createElement('p');  
paragraph.textContent = 'New paragraph';
```

- a) Impératif
- b) Déclaratif

Impératif : On utilise document et on définit ce que l'on veut faire.

72.Code impératif et/ou déclaratif ?

```
const numbers = [1, 2, 3, 4, 5];  
const doubledNumbers = numbers.map(num => num * 2);
```

a) Impératif

b) Déclaratif

Déclaratif : On utilise map et on définit ce que l'on veut obtenir.

73.Code impératif et/ou déclaratif ?

```
function Greeting({ isLoggedIn }) {  
  return isLoggedIn ? <p>Welcome back!</p> : <p>Please log in.</p>;  
}
```

```
ReactDOM.render(  
  <Greeting isLoggedIn={true} />,  
  document.getElementById('root')  
);
```

a) Impératif

b) Déclaratif

Déclaratif : On définit ce que l'on veut obtenir de manière simple.

74.Code impératif et/ou déclaratif ?

```
function findEvenNumbers(numbers) {  
  const evenNumbers = [];  
  
  for (let i = 0; i < numbers.length; i++) {  
    const num = numbers[i];  
  
    if (num % 2 === 0) {  
      evenNumbers.push(num);  
    }  
  }  
}
```

```
}
```

```
    return evenNumbers;
```

```
}
```

a) Impératif

b) Déclaratif

Impératif : On définit exactement ce que l'on veut faire.

75.Code impératif et/ou déclaratif ?

```
const data = [  
  { name: 'John', age: 30, occupation: 'Engineer' },  
  { name: 'Alice', age: 25, occupation: 'Designer' },  
  { name: 'Bob', age: 35, occupation: 'Developer' },  
  { name: 'Sarah', age: 28, occupation: 'Manager' }  
];
```

```
const filteredData = data  
  .filter(person => person.age >= 30)  
  .map(person => ({  
    name: person.name,  
    occupation: person.occupation  
  }));
```

```
console.log(filteredData);
```

a) Impératif

b) Déclaratif

Déclaratif : On définit ce que l'on veut obtenir de manière simple.

Comme tu peux le voir, c'est un concept très simple à comprendre une fois qu'on l'a assimilé. Pour écrire du code déclaratif, il va falloir comprendre les fonctions `.map`, `.filter`, `.reduce`, `.sort` etc... qui sont l'une des manières de faire du code déclaratif.

Dans le prochain module, nous verrons comment faire du code déclaratif avec ces fonctions.

Pourquoi on dit que React est déclaratif ?

React est un framework déclaratif et si on dit ça, ce n'est pas pour rien.

Imagine que tu veuilles faire un "bouton" qui, lors du clic, affiche une alerte.

Tu vas devoir faire ceci en JavaScript :

1. Ajouter le bouton dans le HTML

HTML

```
<button id="myButton">Cliquez-moi</button>
```

2. Ajouter l'écouteur d'événement dans le JavaScript

JS

```
document.getElementById('myButton').addEventListener('click', function() {  
    alert('Alerte affichée !');  
});
```

Comme tu peux le voir ici, on déclare ce que l'on veut faire :

Je récupère le bouton `myButton` et j'ajoute l'écouteur d'événement `click` qui affiche une alerte.

En React, pour faire la même chose, on pourrait faire ça :

JSX

```
<button onClick={() => {
```

```
    alert('Alerte affichée !');  
  }}>
```

```
    Cliquez-moi  
</button>
```

Ici, on déclare ce que l'on veut de manière simple avec une très grande abstraction.

Je crée un bouton qui, lors du clic, affiche une alerte.

76.

77. Quel ordre est déclaratif ?

- a) Va nettoyer ta chambre maintenant !
- b) Fais tes devoirs après le dîner.
- c) J'aimerais que ta chambre soit propre.
- d) Les devoirs doivent être faits avant de regarder la télévision.

Explication

C est la seule bonne réponse car c'est la seule qui demande un résultat, qui exprime ce qu'elle veut à la fin. Toutes les autres réponses spécifient plutôt le chemin, ce qu'il faut faire.

78. Quels ordres sont impératifs ?

- a) Obtiens la somme des éléments du tableau
- b) Liste tous les éléments du tableau un par un.
- c) Augmente la valeur de chaque élément du tableau de un.
- d) Chaque élément du tableau doit être une unité plus grand que son état actuel.

Explication

La première est la dernière spécifie le résultat souhaité, alors que les deux du milieu expriment les actions à faire.

79. Les boucles 'for' et 'while' sont des exemples typiques de la programmation déclarative.

a) True

b) False

Explication

Ces boucles sont caractéristiques de la programmation impérative, car elles définissent explicitement comment itérer sur des données.

80. La programmation fonctionnelle, souvent considérée comme impérative, se concentre sur le '**comment**' faire plutôt que le '**quoi**' faire.

a) True

b) False

Explication

La phrase est fausse parce que la programmation fonctionnelle est justement déclarative (comme pour React) et elle se concentre sur le QUOI.

Utilisez des noms de variables claires et prononçables

C'est-à-dire des noms que tu peux lire en anglais dans ta tête, qui sont logiques.

JS

// X

```
const yyyymmddstr = moment().format("YYYY/MM/DD");
```

JS

// ✓

```
const currentDate = moment().format("YYYY/MM/DD");
```

JS

// X *Pas d'acronyme*

```
const dbny = moment().format("YYYY/MM/DD");
```

JS

// ✓


```
const dateBeforeNewYear = moment().format("YYYY/MM/DD");
```

JS

```
// X Pas de raccourci
```

```
const getUserInfo = () => {};
```

JS

```
// ✓
```

```
const getUserInformation = () => {};
```

Utilise un vocabulaire cohérent

Utilise le même vocabulaire pour le même type de variable. Dans une application, il y a toujours des mots qu'on réutilise souvent, reste cohérent !

JS

```
// X
```

```
getUserInfo();
```

```
getClientData();
```

```
getCustomerRecord();
```

JS

```
// ✓
```

```
getUser(); // Ou getUserInformation
```

Aussi, comme tu le vois ici, n'utilise pas des suffixes inutiles.

Pas de magic value

Les magic value sont les variables qui "tombent du ciel" dont on ne comprend pas la provenance. C'est-à-dire des valeurs qui sont utilisées dans le code mais dont on ne sait pas d'où elles viennent.

JS

//X

```
setTimeout(event, 86400000);
```

JS

//✓

```
const MILLISECONDS_IN_A_DAY = 1000 * 60 * 60 * 24;
```

```
setTimeout(event, MILLISECONDS_IN_A_DAY);
```

En général, on utilise des constantes pour les magic value afin de :

- Pouvoir décrire ce que c'est avec leur nom
- Pouvoir les réutiliser plusieurs fois
- Pouvoir les changer facilement

Pas d'index magic

Les tableaux sont un piège car on a rapidement envie d'utiliser les index pour accéder aux valeurs. Mais les index sont des magic value, on ne sait pas ce qu'ils représentent.

Alors qu'on peut utiliser le destructuring pour récupérer les valeurs.

JS

//X

```
const EXTRACT_EMAIL_REGEX = /^[^@]+@([^\.]+)\.(.+)$/;
```

```
const email = "test@gmail.com";
```

```
const result = email.match(EXTRACT_EMAIL_REGEX);
```

```
analytics.track(result[2]);
```

JS

//✓

```
const EXTRACT_EMAIL_REGEX = /^([^\@]+)\@([^\.\.]+\.)\.(.+)$/;
```

```
const email = "test@gmail.com";
```

```
const [, , domain] = email.match(EXTRACT_EMAIL_REGEX);
```

```
analytics.track(domain);
```

Éviter les noms en 1 lettre

On aime bien lors des boucles utiliser des noms en une seule lettre de variable pour que ce soit plus court. Mais cet argument n'est plus valide depuis qu'il y a l'autocomplétion.

JS

//✗

```
const users = ["John", "Paul", "George", "Ringo"];
users.forEach((u) => {
  something(u);
  otherThing(u);
  // ...
  doSomething(u);
});
```

JS

//✓

```
const users = ["John", "Paul", "George", "Ringo"];
users.forEach((user) => {
  something(user);
  otherThing(user);
  // ...
  doSomething(user);
});
```

```
});
```

Le seul moment où c'est autorisé, c'est pour les "one-liners" où de toute façon tu retrouves facilement le nom de la variable.

JS

```
const users = ["John", "Paul", "George", "Ringo"];
```

```
users.forEach((u) => doSomething(u));
```

N'ajoute pas des choses inutiles

Quand on nomme des variables ou des clés d'objet, il faut toujours faire simple.

JS

```
//✗
```

```
const user = {  
  userName: "John",  
  userLastName: "Doe",  
  userAge: 30,  
};
```

JS

```
//✓
```

```
const user = {  
  name: "John",  
  lastName: "Doe",  
  age: 30,  
};
```

De la même manière pour les informations qui seront dans une scope de fonction ou autre.

Conclusion

Si tu suis les principes ci-dessus, non seulement tu seras plus performant mais tes collègues se diront que tu es une vraie star !

81. Quel nom de variable privilégié pour la variable x dans ce contexte :

JS

```
const adminUsers = ["John", "Paul", "George", "Ringo"];
```

```
adminUsers.forEach((x) => {  
  something(x);  
  otherThing(x);  
  // ...  
  doSomething(x);  
});
```

a) adminUser

b) user

c) aU

d) u

Explication

Dans la boucle, le fait que ce soit un admin et pas un user n'a pas beaucoup de valeur. On peut sans autre utiliser juste user sauf si la variable existe déjà dans notre contexte.

82. Comment bien nommer cette variable : productInfo ?

a) productsInfo

b) productInformation

c) productInformations

Explication

productInformations est le meilleur nom, car on évite les noms 'raccourcis' et qu'on utilise le pluriel pour faire comprendre qu'il y a plusieurs informations.

83. Comment rendre ce code plus propre ?

JS

```
const users = [{  
  name: "John",  
  age: 30,  
}, ...]
```

```
for (const user of users) {  
  doSomething(user[0], user[1])  
}
```

- a) Utiliser une boucle `.map`
- b) Il faut déstructurer `user` pour nommer les différentes variables.
- c) Il faut appeler `user`userNameAndAge``

Explication

La boucle `for of` est recommandée, pas de souci là-dessus. Par contre, il serait intéressant de déstructurer `user` dans celle-ci pour ne pas récupérer les éléments avec leur index.

Dans la boucle `for...of` donnée, chaque itération extrait un élément `user` du tableau `users` et passe cet élément dans la fonction `doSomething` en utilisant `user[0]` et `user[1]`. Cela suppose que chaque élément `user` est un tableau avec exactement deux éléments.

Cependant, d'après la structure de l'objet `users` donnée dans le code, chaque élément `user` est un objet avec les propriétés `name` et `age`. Par conséquent, pour accéder aux propriétés `name` et `age`, il est nécessaire de déstructurer l'objet `user` lors de l'itération de la boucle.

Le code corrigé ressemblerait à ceci :

```
for (const { name, age } of users) {  
  doSomething(name, age);  
}
```

Dans cette version corrigée, la boucle `for...of` utilise la déstructuration pour extraire les propriétés `name` et `age` de chaque objet `user` du tableau `users`. Ainsi, les variables `name` et `age` sont utilisées directement dans l'appel à `doSomething`, ce qui permet de simplifier l'accès aux propriétés de chaque utilisateur.

Donc, la réponse correcte est b) Il faut déstructurer `user` pour nommer les différentes variables.

84. Quel nom serait le plus approprié pour cette variable :

JS

```
const dbny = getDateBeforeNewYear();
```

- a) `date`
- b) `dateBefore`
- c) `dateBeforeNew`
- d) `dateBeforeNewY`
- e) `dateBeforeNewYear`

Explication

Mieux vaut être explicite et long que vouloir faire court.

85. Quelle syntaxe respecte la nomenclature React ?

- a) `const [state, setState]`
- b) `const [isActive, setIsActive]`
- c) `const [hasEat, setHasEat]`

Explication

La réponse deux n'a pas de majuscule pour le ``isActive``, ce qui la rend invalide.

86. Qu'est-ce que ce code va produire ?

TS

```
let [state, setState] = useState("test")
```

a) state = "new value"

b) La variable state sera maintenant égale à 'test'

c) React va render notre composant

d) React ne va pas render notre composant

e) L'application va crasher

Explication

La variable va changer, mais l'interface ne changera pas, elle restera strictement identique.

87. Quelles phrases sont vraies au sujet des renders ?

a) Un render n'est que l'appel de la fonction du composant.

b) Tous les renders update le DOM.

c) Un render n'update pas forcément le DOM.

d) Un render se produit uniquement lors du démarrage de l'application.

e) Un render se produit quand un state est modifié et lors du mount.

Explication

Un render n'est que l'appel de la fonction du composant, il n'update pas forcément le DOM et se produit quand un state est modifié ou lors du mount.

Les phrases vraies au sujet des renders sont :

c) Un render n'update pas forcément le DOM.

e) Un render se produit quand un state est modifié et lors du mount.

Explication :

a) Un render n'est que l'appel de la fonction du composant : Cette phrase est incorrecte. Un render n'est pas seulement l'appel de la fonction du composant, mais il implique également la mise à jour du DOM ou d'autres éléments de rendu.

- b) Tous les renders update le DOM : Cette phrase est incorrecte. Tous les renders ne mettent pas à jour automatiquement le DOM. La bibliothèque ou le framework utilisé peut implémenter des mécanismes de rendu conditionnels ou de virtual DOM pour optimiser les performances et éviter les mises à jour inutiles du DOM.
- c) Un render n'update pas forcément le DOM : Cette phrase est vraie. Un render peut se produire sans nécessiter une mise à jour du DOM. Cela peut se produire lorsque certains éléments de l'état (state) ou des propriétés (props) d'un composant changent, mais que ces changements n'ont pas d'impact sur le rendu final.
- d) Un render se produit uniquement lors du démarrage de l'application : Cette phrase est incorrecte. Un render peut se produire à plusieurs moments, y compris lors du démarrage initial de l'application, mais aussi lorsqu'un composant est monté (mounted) dans l'arborescence du DOM ou lorsqu'un état (state) est modifié.
- e) Un render se produit quand un state est modifié et lors du mount : Cette phrase est vraie. Un render est déclenché lorsque l'état (state) d'un composant est modifié, ce qui entraîne une mise à jour du rendu. De plus, un render se produit également lors du montage (mount) initial d'un composant dans le DOM.

88. Qu'est-ce que JavaScript côté serveur ?

- a) Une version spéciale de JavaScript utilisée exclusivement pour le développement web.
- b) L'utilisation de JavaScript pour exécuter du code sur le serveur plutôt que sur le navigateur.
- c) Une bibliothèque JavaScript pour créer des interfaces utilisateur côté serveur.

[Réponse correcte : b]

89. Comment installe-t-on nodeJS avec volta ?

- a) En utilisant la commande "npm install volta" dans le terminal.
- b) En téléchargeant et en installant le package volta à partir du site officiel.

c) En utilisant le gestionnaire de packages yarn pour installer volta.

[Réponse correcte : b]

90. Comment installe-t-on nodeJS sur Windows ?

a) En téléchargeant et en installant l'exécutable nodeJS à partir du site officiel.

b) En utilisant la commande "npm install node" dans le terminal.

c) En ajoutant le chemin d'installation de nodeJS à la variable d'environnement PATH.

[Réponse correcte : a]

91. Qu'est-ce que les Streams en JavaScript côté serveur ?

a) Des flux de données utilisés pour lire et écrire des fichiers.

b) Des fonctions pour effectuer des opérations mathématiques complexes côté serveur.

c) Des méthodes pour manipuler des tableaux de données en JavaScript.

[Réponse correcte : a]

92. Comment lit-on et écrit-on des fichiers en JavaScript côté serveur ?

a) En utilisant les méthodes "readFile()" et "writeFile()" du module fs.

b) En utilisant les fonctions "read()" et "write()" du module fileSystem.

c) En ajoutant des attributs spécifiques aux éléments HTML pour les rendre déplaçables.

[Réponse correcte : a]

93. Qu'est-ce que le serveur (express) en JavaScript côté serveur ?

a) Une bibliothèque JavaScript pour créer des interfaces utilisateur côté serveur.

b) Un environnement de développement pour écrire et tester du code JavaScript côté serveur.

c) Un framework JavaScript minimaliste et flexible pour créer des applications web côté serveur.

[Réponse correcte : c]

94. Qu'est-ce que les routes en JavaScript côté serveur ?

a) Des méthodes pour définir des chemins d'accès et gérer les requêtes HTTP.

b) Des fonctions pour générer des nombres aléatoires dans une plage donnée.

c) Des techniques pour valider le comportement attendu d'un programme ou d'une fonction.

[Réponse correcte : a]

95. Qu'est-ce que la base de données MongoDB en JavaScript côté serveur ?

- a) Une bibliothèque pour manipuler des chaînes de caractères en JavaScript.
- b) Un outil de débogage intégré dans les navigateurs web.
- c) Un système de gestion de base de données NoSQL utilisé avec JavaScript côté serveur.

[Réponse correcte : c]

96. Qu'est-ce que le CRUD en JavaScript côté serveur ?

- a) Une méthode pour vérifier et valider la syntaxe d'une adresse email.
- b) Un acronyme qui représente les opérations de base sur les données : Create, Read, Update, Delete.
- c) Une technique pour compresser des fichiers en JavaScript.

[Réponse correcte : b]

97. Qu'est-ce que JavaScript côté serveur ?

- a) Une version spéciale de JavaScript utilisée exclusivement pour le développement web.
- b) L'utilisation de JavaScript pour exécuter du code sur le serveur plutôt que sur le navigateur.
- c) Une bibliothèque JavaScript pour créer des interfaces utilisateur côté serveur.

[Réponse correcte : b]

98. Comment installe-t-on nodeJS avec volta ?

- a) En utilisant la commande "npm install volta" dans le terminal.
- b) En téléchargeant et en installant le package volta à partir du site officiel.
- c) En utilisant le gestionnaire de packages yarn pour installer volta.

[Réponse correcte : b]

99. Comment installe-t-on nodeJS sur Windows ?

- a) En téléchargeant et en installant l'exécutable nodeJS à partir du site officiel.

- b) En utilisant la commande "npm install node" dans le terminal.
- c) En ajoutant le chemin d'installation de nodeJS à la variable d'environnement PATH.

[Réponse correcte : a]

100. Qu'est-Désolé, je me suis trompé dans la numérotation des questions. Voici la suite des questions :

101. Qu'est-ce que les Streams en JavaScript côté serveur ?

- a) Des flux de données utilisés pour lire et écrire des fichiers.
- b) Des fonctions pour effectuer des opérations mathématiques complexes côté serveur.
- c) Des méthodes pour manipuler des tableaux de données en JavaScript.

[Réponse correcte : a]

102. Comment lit-on et écrit-on des fichiers en JavaScript côté serveur ?

- a) En utilisant les méthodes "readFile()" et "writeFile()" du module fs.
- b) En utilisant les fonctions "read()" et "write()" du module fileSystem.
- c) En ajoutant des attributs spécifiques aux éléments HTML pour les rendre déplaçables.

[Réponse correcte : a]

103. Qu'est-ce que le serveur (express) en JavaScript côté serveur ?

- a) Une bibliothèque JavaScript pour créer des interfaces utilisateur côté serveur.
- b) Un environnement de développement pour écrire et tester du code JavaScript côté serveur.
- c) Un framework JavaScript minimaliste et flexible pour créer des applications web côté serveur.

[Réponse correcte : c]

104. Qu'est-ce que les routes en JavaScript côté serveur ?

- a) Des méthodes pour définir des chemins d'accès et gérer les requêtes HTTP.
- b) Des fonctions pour générer des nombres aléatoires dans une plage donnée.
- c) Des techniques pour valider le comportement attendu d'un programme ou d'une fonction.

[Réponse correcte : a]

- 105. Qu'est-ce que la base de données MongoDB en JavaScript côté serveur ?**
- a) Une bibliothèque pour manipuler des chaînes de caractères en JavaScript.**
 - b) Un outil de débogage intégré dans les navigateurs web.**
 - c) Un système de gestion de base de données NoSQL utilisé avec JavaScript côté serveur.**
- [Réponse correcte : c]**
- 106. Qu'est-ce que le CRUD en JavaScript côté serveur ?**
- a) Une méthode pour vérifier et valider la syntaxe d'une adresse email.**
 - b) Un acronyme qui représente les opérations de base sur les données : Create, Read, Update, Delete.**
 - c) Une technique pour compresser des fichiers en JavaScript.**
- [Réponse correcte : b]**
- 107. Qu'est-ce que le middleware en JavaScript côté serveur ?**
- a) Un terme utilisé pour décrire un développeur JavaScript travaillant exclusivement sur le côté serveur.**
 - b) Une fonction intermédiaire qui peut être utilisée pour effectuer des opérations supplémentaires lors du traitement des requêtes et des réponses.**
 - c) Une bibliothèque JavaScript pour faciliter la gestion des dépendances côté serveur.**
- [Réponse correcte : b]**
- 108. Qu'est-ce que l'authentification et l'autorisation en JavaScript côté serveur ?**
- a) Des concepts utilisés pour garantir la sécurité des applications web en vérifiant l'identité des utilisateurs et en contrôlant leur accès aux ressources.**
 - b) Des fonctions JavaScript spéciales pour interagir avec les bases de données côté serveur.**
 - c) Des techniques pour optimiser les performances et la vitesse d'exécution des applications web en JavaScript côté serveur.**
- [Réponse correcte : a]**

109. Qu'est-ce qu'un canvas en JavaScript ?

- a) Un élément HTML utilisé pour afficher des images et des graphiques générés par programmation.**
- b) Une fonction JavaScript utilisée pour dessiner des cercles et des rectangles.**
- c) Une méthode pour manipuler des tableaux de données en JavaScript.**

[Réponse correcte : a]

110. Qu'est-ce que le Drag & Drop en JavaScript ?

- a) Une méthode pour déplacer des éléments HTML en utilisant la souris.**
- b) Une fonction JavaScript pour trier des éléments dans un tableau.**
- c) Une technique pour compresser des fichiers en JavaScript.**

[Réponse correcte : a]

111. Quel est l'objectif du bloc "try/catch" en JavaScript ?

- a) Exécuter un bloc de code spécifique plusieurs fois.**
- b) Capturer et gérer les erreurs potentielles dans un bloc de code.**
- c) Créer des variables locales dans un contexte spécifique.**

[Réponse correcte : b]

112. Qu'est-ce que le "strict mode" en JavaScript ?

- a) Un mode de fonctionnement qui permet d'exécuter du code asynchrone.**
- b) Une directive pour activer un ensemble de règles strictes lors de l'écriture de code JavaScript.**
- c) Une méthode pour convertir des chaînes de caractères en nombres entiers.**

[Réponse correcte : b]

113. Qu'est-ce que le module import/export en JavaScript ?

- a) Une méthode pour importer et exporter des fichiers CSV en JavaScript.**
- b) Une fonction JavaScript pour fusionner deux tableaux en un seul.**
- c) Un mécanisme pour importer et exporter des fonctions et des objets entre différents fichiers JavaScript.**

[Réponse correcte : c]

114. Quel est l'objectif du TP "Todolist" en JavaScript ?

- a) Créer une application pour gérer une liste de tâches à faire.**
- b) Manipuler des chaînes de caractères en utilisant des expressions**

régulières.

c) Générer des nombres aléatoires en JavaScript.

[Réponse correcte : a]

115. Qu'est-ce que les tests en JavaScript ?

a) Des fonctions spéciales pour générer des nombres aléatoires.

b) Des méthodes pour valider la syntaxe d'une adresse email.

c) Des techniques pour vérifier et valider le comportement attendu d'un programme ou d'une fonction.

[Réponse correcte : c]

116. Qu'est-ce que l'ECMAScript en JavaScript ?

a) Une bibliothèque populaire de fonctions utilitaires pour JavaScript.

b) Une version standardisée de JavaScript avec des mises à jour régulières.

c) Un outil de débogage intégré dans les navigateurs web.

[Réponse correcte : b]

117. Qu'est-ce que le TypeScript en JavaScript ?

a) Une bibliothèque pour manipuler des dates et des heures en JavaScript.

b) Une extension de JavaScript qui ajoute un typage statique optionnel.

c) Une méthode pour créer des animations en JavaScript.

[Réponse correcte : b]

118. Quelle est la fonction principale d'un canvas en JavaScript ?

a) Gérer les erreurs et les exceptions lors de l'exécution de code JavaScript.

b) Afficher des éléments HTML en utilisant des classes CSS spécifiques.

c) Dessiner des graphiques, des animations et des images générés par programmation.

[Réponse correcte : c]

119. Comment utilise-t-on le Drag & Drop en JavaScript ?

a) En utilisant les fonctions "drag()" et "drop()" pour déplacer des éléments HTML.

b) En ajoutant des attributs spécifiques aux éléments HTML pour les rendre déplaçables.

c) En utilisant les méthodes "dragStart()" et "dragEnd()" pour gérer les

événements de glisser-déposer.

[Réponse correcte : c]

120. Quelle est la syntaxe pour utiliser le bloc "try/catch" en JavaScript ?

- a) try { ... } except { ... }
- b) try { ... } catch { ... }
- c) try { ... } catch (error) { ... }

[Réponse correcte : c]

121. Quels avantages apporte le "strict mode" en JavaScript ?

- a) Une meilleure performance de l'exécution du code JavaScript.
- b) 13. Quels avantages apporte le "strict mode" en JavaScript ?
- a) Une meilleure performance de l'exécution du code JavaScript.
- b) Une vérification plus stricte des erreurs de syntaxe et des pratiques déconseillées.
- c) Une simplification de la syntaxe JavaScript pour faciliter la lecture du code.

[Réponse correcte : b]

122. Comment utilise-t-on les modules import/export en JavaScript ?

- a) En utilisant les mots-clés "import" et "export" suivis du nom du module.
- b) En utilisant les fonctions "import()" et "export()" pour charger et exporter des modules.
- c) En ajoutant des attributs spécifiques aux éléments HTML pour les rendre déplaçables.

[Réponse correcte : a]

123. Quelle est la principale fonction du TP "Todolist" en JavaScript ?

- a) Apprendre à manipuler des tableaux en JavaScript.
- b) Créer une application pour gérer une liste de tâches à faire.
- c) Utiliser des fonctions pour générer des nombres aléatoires en JavaScript.

[Réponse correcte : b]

124. Qu'est-ce qu'un test en JavaScript ?

- a) Une méthode pour vérifier si une variable est de type string.
- b) Une technique pour valider le comportement attendu d'un programme ou d'une fonction.

c) Une fonction pour générer des nombres aléatoires dans une plage donnée.
[Réponse correcte : b]

125. Qu'est-ce que l'ECMAScript en relation avec JavaScript ?

- a) Une version spécifique de JavaScript utilisée pour les applications mobiles.
- b) Une norme de langage qui définit les spécifications et les fonctionnalités de JavaScript.
- c) Une bibliothèque JavaScript populaire pour créer des interfaces utilisateur.

[Réponse correcte : b]

126. Qu'est-ce que le TypeScript en relation avec JavaScript ?

- a) Une extension de JavaScript qui ajoute un typage statique optionnel.
- b) Une méthode pour effectuer des opérations mathématiques complexes en JavaScript.
- c) Une bibliothèque pour manipuler des chaînes de caractères en JavaScript.

[Réponse correcte : a]

127. Quelle est la fonction principale d'un canvas en JavaScript ?

- a) Afficher des éléments HTML en utilisant des classes CSS spécifiques.
- b) Dessiner des graphiques, des animations et des images générés par programmation.
- c) Gérer les erreurs et les exceptions lors de l'exécution de code JavaScript.

[Réponse correcte : b]

128. Comment utilise-t-on le Drag & Drop en JavaScript ?

- a) En utilisant les fonctions "drag()" et "drop()" pour déplacer des éléments HTML.
- b) En ajoutant des attributs spécifiques aux éléments HTML pour les rendre déplaçables.
- c) En utilisant les méthodes "dragStart()" et "dragEnd()" pour gérer les événements de glisser-déposer.

[Réponse correcte : c]

129. Qu'est-ce que la programmation orientée objet (POO) en JavaScript ?

- a) Un style de programmation qui utilise principalement des fonctions.

b) Une approche de programmation qui se concentre sur les objets et leurs interactions.

c) Une méthode pour optimiser les performances d'un code JavaScript.

[Réponse correcte : b]

130. Quelles sont les différentes manières de créer des objets en JavaScript ?

a) Les fonctions constructeurs, les factory functions et les prototypes.

b) Les fonctions, les objets littéraux et les tableaux.

c) Les classes, les méthodes et les propriétés.

[Réponse correcte : a]

131. Qu'est-ce qu'une méthode dans un objet JavaScript ?

a) Une fonction qui est appelée lorsqu'un événement se produit.

b) Une fonction qui est attachée à un objet et peut être appelée pour effectuer une action spécifique.

c) Une instruction permettant de définir une variable dans un objet.

[Réponse correcte : b]

132. Qu'est-ce qu'une méthode "notices" dans un objet JavaScript ?

a) Une méthode qui affiche des notifications à l'utilisateur.

b) Une méthode qui gère les erreurs et les exceptions dans un objet.

c) Il n'y a pas de notion de méthode "notices" dans JavaScript.

[Réponse correcte : c]

133. Quelle est la fonctionnalité principale d'une fonction constructeur en JavaScript ?

a) Créer des instances d'objets avec des propriétés et des méthodes spécifiques.

b) Exécuter du code de manière asynchrone.

c) Manipuler des chaînes de caractères sous forme d'objets.

[Réponse correcte : a]

134. Qu'est-ce qu'une factory function en JavaScript ?

a) Une fonction qui crée un objet à partir d'un autre objet existant.

b) Une fonction qui retourne un nouvel objet à chaque appel.

c) Une fonction qui gère les erreurs et les exceptions dans un objet.

[Réponse correcte : b]

135. Qu'est-ce qu'un prototype en JavaScript ?

- a) Un modèle utilisé pour créer de nouveaux objets avec des propriétés et des méthodes héritées.**
- b) Un outil de débogage pour inspecter les objets JavaScript.**
- c) Une fonction qui permet de cloner un objet existant.**

[Réponse correcte : a]

136. Qu'est-ce qu'une classe en JavaScript ?

- a) Un groupe d'objets partageant des caractéristiques communes.**
- b) Une fonction qui crée des instances d'objets avec des propriétés et des méthodes spécifiques.**
- c) Une méthode pour encapsuler du code et le réutiliser dans plusieurs objets.**

[Réponse correcte : b]

137. Quelle est la fonctionnalité principale de l'héritage en JavaScript ?

- a) Permettre à un objet d'accéder aux propriétés et aux méthodes d'un autre objet.**
- b) Créer une copie exacte d'un objet existant.**
- c) Ajouter de nouvelles propriétés à un objet existant.**

[Réponse correcte : a]

138. Quel est l'objectif du projet "Yoga App" en JavaScript ?

- a) Créer une application pour gérer des séances de yoga en utilisant des objets.**
- b) Manipuler des tableaux en JavaScript pour stocker des données de séances de yoga.**
- c) Valider la syntaxe d'une adresse email à l'aide d'une expression régulière.**

[Réponse correcte : a]

139. Comment crée-t-on un objet en utilisant une fonction constructeur en JavaScript ?

- a) En utilisant le mot-clé "new" suivi du nom de la fonction constructeur.**
- b) En déclarant un objet littéral avec des propriétés et des méthodes.**
- c) En utilisant la méthode "create()" de l'objet JavaScript.**

[Réponse correcte : a]

- 140. Comment ajoute-t-on une méthode à un objet en JavaScript ?**
- a) En utilisant le mot-clé "method" suivi du nom de la méthode et de sa définition.
 - b) En utilisant la syntaxe "objet.nomMethode = fonction() {...}".
 - c) En utilisant la méthode "addMethod()" de l'objet JavaScript.
- [Réponse correcte : b]**
- 141. Comment crée-t-on un nouvel objet à partir d'un autre objet en JavaScript ?**
- a) En utilisant la méthode "clone()" de l'objet source.
 - b) En utilisant la syntaxe "Object.create(objetSource)".
 - c) En utilisant la méthode "newObject()" de l'objet JavaScript.
- [Réponse correcte : b]**
- 142. Quelle est la différence entre une fonction constructeur et une factory function en JavaScript ?**
- a) Une fonction constructeur crée des objets avec le mot-clé "new", tandis qu'une factory function crée des objets en les retournant directement.
 - b) Une fonction constructeur crée des objets avec le mot-clé "create", tandis qu'une factory function crée des objets avec le mot-clé "new".
 - c) Il n'y a pas de différence entre une fonction constructeur et une factory function en JavaScript.
- [Réponse correcte : a]**
- 143. Comment crée-t-on une classe en JavaScript ?**
- a) En utilisant le mot-clé "class" suivi du nom de la classe et de sa définition.
 - b) En utilisant la syntaxe "function maClasse() {...}".
 - c) En utilisant la méthode "createClass()" de l'objet JavaScript.
- [Réponse correcte : a]**
- 144. Comment réalise-t-on l'héritage entre deux classes en JavaScript ?**
- a) En utilisant le mot-clé "inherit" suivi du nom de la classe parente.
 - b) En utilisant la syntaxe "ChildClass extends ParentClass".
 - c) Il n'est pas possible de réaliser l'héritage entre deux classes en JavaScript.
- [Réponse correcte : b]**

145. Qu'est-ce que la méthode "super" en JavaScript ?

- a) Une méthode pour appeler une fonction parente à partir d'une classe enfant.**
- b) Une méthode pour vérifier si un objet est une instance d'une classe spécifique.**
- c) Une méthode pour accéder à des propriétés et des méthodes privées d'un objet.**

[Réponse correcte : a]

146. Comment crée-t-on une instance d'une classe en JavaScript ?

- a) En utilisant la méthode "createInstance()" de l'objet JavaScript.**
- b) En utilisant le mot-clé "new" suivi du nom de la classe.**
- c) En utilisant la syntaxe "class.createInstance()" avec le nom de la classe.**

[Réponse correcte : b]

147. Qu'est-ce que le polymorphisme en JavaScript ?

- a) Une technique pour masquer les détails d'implémentation d'une classe.**
- b) Une méthode pour gérer les erreurs et les exceptions dans une classe.**
- c) Une capacité d'un objet à prendre différentes formes à travers l'héritage.**

[Réponse correcte : c]

Qu'est-ce que l'encapsulation en JavaScript ?

- a) Une technique pour regrouper des objets similaires dans une même classe.**
- b) Une méthode pour limiter l'accès aux propriétés et aux méthodes d'un objet.**
- c) Une capacité d'un objet à hériter des propriétés et des méthodes d'un autre objet.**

[Réponse correcte : b]

148. Qu'est-ce qu'une API en JavaScript ?

- a) Un outil pour créer des interfaces utilisateur graphiques.**
- b) Un ensemble de fonctions et de protocoles permettant aux applications de communiquer entre elles.**

c) Une méthode pour effectuer des opérations mathématiques complexes.

[Réponse correcte : b]

149. Quel objet JavaScript est utilisé pour effectuer des requêtes HTTP asynchrones ?

a) XMLHttpRequest.

b) JSON.parse.

c) Math.random.

[Réponse correcte : a]

150. Quelle est la méthode moderne pour effectuer des requêtes HTTP en JavaScript ?

a) XMLHttpRequest.

b) Fetch.

c) AJAX.

[Réponse correcte : b]

151. Quel est l'objectif du projet "Générateur de blagues" en JavaScript ?

a) Afficher des blagues aléatoires à l'écran.

b) Générer des mots de passe aléatoires.

c) Valider la syntaxe d'une adresse email.

[Réponse correcte : a]

152. Que sont les "headers" dans le contexte d'une requête HTTP ?

a) Les instructions conditionnelles pour exécuter certaines actions.

b) Les informations supplémentaires envoyées avec une requête ou une réponse HTTP.

c) Les propriétés et méthodes d'un objet JavaScript.

[Réponse correcte : b]

153. Quelles sont les méthodes HTTP utilisées pour envoyer des données au serveur ?

a) GET, PUT, DELETE.

b) POST, DELETE, PATCH.

c) POST, PUT, DELETE.

[Réponse correcte : c]

154. Qu'est-ce que l'asynchronisme en JavaScript ?

- a) Une méthode pour exécuter du code de manière synchrone, étape par étape.**
- b) Une technique pour manipuler des chaînes de caractères de manière asynchrone.**
- c) La capacité d'exécuter du code de manière non bloquante et de continuer l'exécution en parallèle.**

[Réponse correcte : c]

155. Quel est l'objectif du projet "User API" en JavaScript ?

- a) Gérer les utilisateurs d'une application en utilisant une API externe.**
- b) Manipuler les propriétés d'un objet utilisateur en utilisant des méthodes prédéfinies.**
- c) Valider la syntaxe d'un nom d'utilisateur et d'un mot de passe.**

[Réponse correcte : a]

156. Qu'est-ce que le JSON en JavaScript ?

- a) Une méthode pour vérifier la validité d'une chaîne de caractères en utilisant une expression régulière.**
- b) Un format de données utilisé pour stocker et échanger des informations structurées.**
- c) Un objet JavaScript utilisé pour effectuer des opérations mathématiques.**

[Réponse correcte : b]

157. Qu'est-ce que le "Local Storage" et le "Session Storage" en JavaScript ?

- a) Des méthodes pour stocker des données dans le navigateur de l'utilisateur.**
- b) Des fonctions pour manipuler des tableaux en JavaScript.**
- c) Des outils pour gérer les connexions réseau en JavaScript.**

[Réponse correcte : a]

158. Qu'est-ce que les "cookies" en JavaScript ?

- a) Des petits fichiers texte stockés sur l'ordinateur de l'utilisateur pour stocker des informations spécifiques.**
- b) Des méthodes pour vérifier la validité d'une adresse email en utilisant une expression régulière.**
- c) Des outils pour effectuer des opérations mathématiques complexes en**

JavaScript.

[Réponse correcte : a]

159. Quel est l'objectif du projet "Meal API" en JavaScript ?

- a) Générer des repas aléatoires à partir d'une API externe.**
- b) Valider la syntaxe d'une adresse IP à l'aide d'une expression régulière.**
- c) Créer des animations graphiques en JavaScript.**

[Réponse correcte : a]

160. Quel est le terme utilisé pour décrire les méthodes de communication entre un client JavaScript et une API externe ?

- a) API interne.**
- b) API locale.**
- c) API externe.**

[Réponse correcte : c]

161. Quelle méthode JavaScript est utilisée pour convertir un objet JavaScript en une chaîne JSON ?

- a) JSON.stringify().**
- b) JSON.parse().**
- c) JSON.stringify().**

162. Comment récupère-t-on les données d'une API en utilisant Fetch en JavaScript ?

- a) fetch(url, options).**
- b) fetch(url).**
- c) fetch(options).**

[Réponse correcte : a]

163. Quelle méthode HTTP est généralement utilisée pour obtenir des données à partir d'une API ?

- a) GET.**
- b) POST.**
- c) PUT.**

[Réponse correcte : a]

164. Quel est le résultat attendu d'une requête DELETE vers une API ?

- a) Supprimer toutes les données de l'API.**

- b) Mettre à jour les données de l'API.
- c) Supprimer une ressource spécifique de l'API.

[Réponse correcte : c]

165. Que signifie JSONP en JavaScript ?

- a) JavaScript Object Notation Protocol.
- b) JavaScript Object Notation Padding.
- c) JavaScript Object Notation Parsing.

[Réponse correcte : b]

166. Qu'est-ce que CORS en JavaScript ?

- a) Un protocole de sécurité utilisé pour restreindre les requêtes HTTP.
- b) Une bibliothèque JavaScript pour créer des animations graphiques.
- c) Une méthode pour gérer les cookies en JavaScript.

[Réponse correcte : a]

167. Comment utilise-t-on l'API de géolocalisation en JavaScript ?

- a) navigator.geolocation.getCurrentPosition().
- b) navigator.location.getCoordinates().
- c) navigator.position.getCurrent().

[Réponse correcte : a]

168. Quels sont les types de données de base en JavaScript ?

- a) Number, String, Boolean.
- b) Integer, Float, Character.
- c) Text, Date, Array.

[Réponse correcte : a]

169. Qu'est-ce qu'un tableau en JavaScript ?

- a) Une structure de données permettant de stocker des valeurs de types différents.
- b) Un ensemble d'instructions conditionnelles.
- c) Une fonction prédéfinie pour effectuer des opérations mathématiques.

[Réponse correcte : a]

170. Qu'est-ce qu'un objet en JavaScript ?

- a) Une variable qui ne peut pas être modifiée une fois définie.**
- b) Un conteneur pour stocker des données de différentes propriétés.**
- c) Une fonction prédéfinie pour manipuler des chaînes de caractères.**

[Réponse correcte : b]

171. Quelles sont les structures de contrôle en JavaScript ?

- a) Les boucles, les conditions et les fonctions.**
- b) Les tableaux, les objets et les chaînes de caractères.**
- c) Les opérations mathématiques et logiques.**

[Réponse correcte : a]

172. Quel est l'objectif du projet "texte typing" en JavaScript ?

- a) Générer des mots de passe aléatoires.**
- b) Créer un effet de saisie de texte à l'écran.**
- c) Vérifier la validité d'un formulaire.**

[Réponse correcte : b]

173. Quelles sont les méthodes couramment utilisées pour les nombres en JavaScript ?

- a) toUpperCase(), toLowerCase(), trim().**
- b) push(), pop(), slice().**
- c) parseInt(), parseFloat(), toFixed().**

[Réponse correcte : c]

174. Quelles sont les méthodes couramment utilisées pour les objets en JavaScript ?

- a) split(), join(), slice().**
- b) hasOwnProperty(), keys(), values().**
- c) map(), filter(), reduce().**

[Réponse correcte : b]

175. Qu'est-ce que les "dot notation" en JavaScript ?

- a) Une méthode pour concaténer des chaînes de caractères.**
- b) Une notation pour accéder aux propriétés et méthodes d'un objet.**
- c) Une règle de syntaxe pour les boucles conditionnelles.**

[Réponse correcte : b]

176. Quel est l'objectif du projet "Générateur de MDP" en JavaScript ?

- a) Vérifier la validité d'un formulaire.**
- b) Générer des mots de passe aléatoires.**
- c) Manipuler les propriétés d'un objet.**

[Réponse correcte : b]

177. Qu'est-ce que le "destructuring" en JavaScript ?

- a) Une méthode pour diviser une chaîne de caractères en un tableau.**
- b) Une technique pour extraire des valeurs d'un tableau ou d'un objet en utilisant une syntaxe concise.**
- c) Une méthode pour valider un format de chaîne de caractères à l'aide d'une expression régulière.**

[Réponse correcte : b]

178. Qu'est-ce que les "datasets" en JavaScript ?

- a) Des ensembles de données utilisés pour effectuer des opérations mathématiques complexes.**
- b) Des attributs spéciaux d'éléments HTML permettant de stocker des données personnalisées.**
- c) Des objets pré-définis contenant des méthodes pour manipuler des chaînes de caractères.**

[Réponse correcte : b]

179. Qu'est-ce que les "REGEX" en JavaScript ?

- a) Des expressions régulières utilisées pour valider des formats de chaînes de caractères.**
- b) Des méthodes pour convertir des nombres en chaînes de caractères.**
- c) Des opérations mathématiques pour manipuler des nombres.**

[Réponse correcte : a]

180. Quel est l'objectif du projet "Form checker" en JavaScript ?

- a) Valider les formulaires en vérifiant que tous les champs sont remplis.**
- b) Générer des formulaires HTML à partir de données d'un objet.**
- c) Analyser les performances d'un site web en collectant des données utilisateur.**

[Réponse correcte : a]

181. Quels sont les principaux types de données en JavaScript ?

- a) Number, String, Array.
- b) Integer, Float, Object.
- c) Function, Boolean, Date.

[Réponse correcte : a]

182. Comment déclare-t-on un tableau vide en JavaScript ?

- a) `var myArray = [];`
- b) `var myArray = {};`
- c) `var myArray = "";`

[Réponse correcte : a]

183. Comment ajoute-t-on un élément à la fin d'un tableau en JavaScript ?

- a) `myArray.push(element);`
- b) `myArray.pop(element);`
- c) `myArray.unshift(element);`

[Réponse correcte : a]

184. Comment accède-t-on à la valeur d'une propriété d'un objet en JavaScript ?

- a) `object.value;`
- b) `object.getProperty();`
- c) `object.property;`

[Réponse correcte : c]

185. Quelle structure de contrôle permet d'exécuter une instruction uniquement si une condition est vraie en JavaScript ?

- a) for loop.
- b) if statement.
- c) switch statement.

[Réponse correcte : b]

186. Quelle méthode JavaScript peut être utilisée pour convertir une chaîne de caractères en nombre entier ?

- a) `parseFloat().`
- b) `toInt().`

c) `parseInt()`.

[Réponse correcte : c]

187. Quelle méthode JavaScript vérifie si un objet possède une propriété spécifique ?

a) `object.hasProperty()`.

b) `object.includesProperty()`.

c) `object.hasOwnProperty()`.

[Réponse correcte : c]

188. Qu'est-ce que le DOM en JavaScript ?

a) Une méthode pour effectuer des opérations mathématiques.

b) Une représentation hiérarchique des éléments d'une page web.

c) Une technologie obsolète.

[Réponse correcte : b]

189. Quels sont les sélecteurs en JavaScript ?

a) Des outils pour créer des boucles.

b) Des méthodes pour cibler et sélectionner des éléments HTML.

c) Des fonctions réservées aux développeurs backend.

[Réponse correcte : b]

190. Que fait la méthode `AddEventListener` en JavaScript ?

a) Ajoute un style CSS à un élément HTML.

b) Détecte et réagit à des événements spécifiques sur un élément HTML.

c) Effectue des opérations mathématiques.

[Réponse correcte : b]

191. Quels types d'événements sont disponibles en JavaScript ?

a) Uniquement les clics de souris.

b) Les clics de souris, les pressions de touches, les survols d'éléments, etc.

c) Les développeurs peuvent définir leurs propres types d'événements.

[Réponse correcte : b]

192. Que font les méthodes autour des événements en JavaScript ?

a) Effectuer des opérations mathématiques.

b) Manipuler les propriétés des éléments HTML lors de la détection d'un événement.

c) Réserver des ressources système pour les événements.

[Réponse correcte : b]

193. Quelle est l'utilisation de la méthode forEach en JavaScript ?

a) Créer des boucles.

b) Itérer sur les éléments d'un tableau et exécuter une fonction spécifique sur chaque élément.

c) Effectuer des opérations mathématiques.

[Réponse correcte : b]

194. Qu'est-ce que la manipulation du DOM en JavaScript permet de faire ?

a) Modifier les styles CSS des éléments HTML.

b) Créer, modifier ou supprimer des éléments HTML et leurs propriétés.

c) Résoudre des équations mathématiques complexes.

[Réponse correcte : b]

195. Que fait la méthode setProperties en JavaScript ?

a) Effectuer des opérations mathématiques.

b) Définir ou modifier les propriétés d'un objet.

c) Créer des boucles.

[Réponse correcte : b]

196. Qu'est-ce que le BOM (Browser Object Model) en JavaScript ?

a) Une représentation hiérarchique des éléments d'un navigateur web.

b) Un outil pour manipuler et accéder aux fonctionnalités du navigateur.

c) Une technologie obsolète.

[Réponse correcte : b]

197. Qu'est-ce qu'un générateur de bulles en JavaScript ?

a) Une animation graphique complexe.

b) Une génération aléatoire de bulles visuelles à l'écran.

c) Un calcul mathématique visuel.

[Réponse correcte : b]

198. Quel est le rôle du DOM en JavaScript ?

a) Manipuler les bases de données.

- b) Manipuler et accéder aux éléments HTML d'une page web.
- c) Manipuler les fichiers du système d'exploitation.

[Réponse correcte : b]

199. Quels sont les sélecteurs couramment utilisés en JavaScript ?

- a) getElementById et getElementsByClassName.
- b) addEventListener et removeEventListener.
- c) setTimeout et setInterval.

[Réponse correcte : a]

200. Comment ajoute-t-on un événement à un élément HTML en JavaScript ?

- a) En utilisant la méthode addEventListener.
- b) En utilisant la méthode setProperty.
- c) En utilisant la méthode getElementById.

[Réponse correcte : a]

201. Quels sont les types d'événements couramment utilisés en JavaScript ?

- a) Click, mouseover, keypress.
- b) Loop, condition, function.
- c) Class, object, inheritance.

[Réponse correcte : a]

202. Comment peut-on manipuler les propriétés des éléments HTML lors de la détection d'un événement en JavaScript ?

- a) En utilisant les méthodes autour des événements comme event.preventDefault().
- b) En utilisant les méthodes autour des événements comme event.target et event.stopPropagation().
- c) En utilisant les méthodes autour des événements comme event.addEventListener().

[Réponse correcte : b]

203. Quelle est l'utilisation de la méthode forEach en JavaScript ?

- a) Itérer sur les propriétés d'un objet.
- b) Itérer sur les éléments d'un tableau et exécuter une fonction spécifique sur chaque élément.

c) Itérer sur les caractères d'une chaîne de caractères.

[Réponse correcte : b]

204. Comment peut-on modifier les styles CSS des éléments HTML en JavaScript ?

a) En utilisant la méthode `setStyle()`.

b) En utilisant la méthode `addClass()`.

c) En utilisant les propriétés `style` et `className` des éléments.

[Réponse correcte : c]

205. Qu'est-ce que le BOM (Browser Object Model) en JavaScript ?

a) Une représentation hiérarchique des éléments d'un navigateur web.

b) Un ensemble d'objets qui représentent différentes fonctionnalités du navigateur.

c) Une technique pour cacher des éléments sur une page web.

[Réponse correcte : b]

206. Qu'est-ce qu'un générateur de bulles en JavaScript ?

a) Une fonction qui génère des nombres aléatoires.

b) Un effet visuel qui crée des bulles animées à l'écran.

c) Un algorithme de tri pour organiser des éléments dans un tableau.

[Réponse correcte : b]

207. Comment peut-on supprimer un élément du DOM en JavaScript ?

a) En utilisant la méthode `removeElement()`.

b) En utilisant la méthode `deleteElement()`.

c) En utilisant les méthodes `removeChild()` ou `remove()`.

[Réponse correcte : c]

La différence entre une bibliothèque JavaScript et un framework JavaScript réside dans le niveau de contrôle et de structure qu'ils offrent aux développeurs pour construire des applications.

Une bibliothèque JavaScript est un ensemble de fonctions et de méthodes prédéfinies qui facilitent certaines tâches de développement. Elle fournit des

fonctionnalités spécifiques que les développeurs peuvent utiliser selon leurs besoins. Les bibliothèques JavaScript sont généralement plus légères et flexibles, permettant aux développeurs de choisir les fonctionnalités qu'ils souhaitent utiliser et de les intégrer à leur propre code.

Un framework JavaScript, en revanche, est un ensemble complet d'outils, de bibliothèques et de conventions qui fournissent une structure et une architecture pour le développement d'applications. Il offre un cadre de travail prédéfini avec des règles et des conventions strictes à suivre. Les frameworks JavaScript sont plus complets et fournissent une solution clé en main pour le développement d'applications, ce qui peut accélérer le processus de développement mais peut également limiter la flexibilité.

En résumé, une bibliothèque JavaScript est un ensemble d'outils pour effectuer des tâches spécifiques, tandis qu'un framework JavaScript est une structure complète qui offre une solution globale pour le développement d'applications. Le choix entre une bibliothèque et un framework dépend des besoins du projet et de la préférence du développeur en termes de flexibilité et de contrôle.