Commencé le	lundi 11 décembre 2023, 13:30
État	Terminé
Terminé le	lundi 11 décembre 2023, 16:00
Temps mis	2 heures 29 min
Note	**25,50** sur 40,00 (**63,75**%)

Description

LISEZ CES INSTRUCTIONS AVANT DE COMMENCER L'EXAMEN

L'examen est composé de plusieurs sections. Vous êtes libres de changer de section en tout temps et de changer les réponses à vos questions. Votre examen sera évalué seulement après avoir cliqué sur le bouton "Tout envoyer et terminer" et avoir confirmé la soumission.

En cas de ralentissement de votre ordinateur, fermez complètement Chrome et rouvrez l'instance.

Voici les règles pour l'évaluation:

- L'examen est noté sur un total de 40 points.
- La note partielle associée à chaque question est marquée à gauche de la question.
- Certaines questions requièrent plusieurs champs à remplir, tandis que d'autres questions sont à choix multiples.
- Un mauvais choix dans une question à réponses multiples impactera la note finale de la question.
- Une bonne réponse sans justification correcte sera pénalisée.
- L'espace disponible n'est pas nécessairement représentatif de la taille de la réponse attendue : **ne vous sentez pas obligés de remplir tout l'espace donné**.

Vous avez une seule tentative pour l'examen final! Ne soumettez pas votre tentative à moins d'être 100% sûr(e) d'avoir terminé l'examen !

En cas de doute sur le sens d'une question, faites une supposition raisonnable, énoncez-la clairement dans votre réponse et poursuivez. Une "question" vide est disponible sur la première page d'examen si vous avez besoin de plus de place ou pour des questions spécifiques.

Les téléphones et les ordinateurs portables ne sont pas permis. Sur votre poste de travail, vous pouvez utiliser seulement Moodle.

Les notes de cours peuvent être consultées dans Moodle, dans la section Ressources. Vous avez aussi droit à de la documentation papier (manuscrite ou imprimée).

Bon travail, bon temps des fêtes et bonnes vacances!

Question 1

Terminé

Non noté

Cette question est un espace dédié pour vos questions sur l'examen. Aucune réponse ne sera donnée pendant l'examen.

Priorisez de mettre vos commentaires et/ou hypothèses directement dans la question spécifique.

Ok.

Question 2

Terminé

Note de 6,00 sur 8,00

Vous travaillez sur un site Web d'achats de produits en ligne. Chaque produit a la structure suivante :

```
{ name: 'Steam Deck', price: 499.00, id: 3, quantity: 1 }
```

Votre système gère le panier de l'utilisateur à l'aide d'un *reducer* contenu dans l'objet *CartContext*. L'état du *reducer* est un objet avec 1 seul attribut *products* étant un tableau de produits: **{ products : [ ] }**

Par défaut, la quantité d'un produit ajouté au panier est de 1 et on peut ajouter plusieurs unités du même produit dans le panier à travers l'action "ADD" de votre reducer qui incrémente l'attribut *quantity*.

Voici un extrait de l'implémentation du component *Cart* qui représente la page de panier :

```
1  const Cart = () => {
2    const { state, dispatch } = useContext(CartContext);
3    const [total, setTotal] = useState(0);
4    const [warning, setWarning] = useState(false);
5
6    useEffect(() => {
7      const newPrice = (state.products.reduce(
8        (acc, x) => { return acc + (x.price * x.quantity) }, 0))
9        .toFixed(2);
10     setTotal(newPrice);
11     setWarning(newPrice > 100);
12     console.log('price > 100');
13   }, [total]);
14
15   return (<>
16     <h1>Votre panier</h1>
17     {state.products.map(product =>
18       <Product product={product} key={product.id} handleClick={() => {
19         dispatch({ type: ACTIONS.DELETE, payload: { id: product.id } })
20       }} />
21     )}
22     <p style={{ color: warning ? "red" : "white" }}>Prix total : {total}$</p>
23   </>
24   )
25 }
```

a) Le paramètre [total] a été rajouté dans l'appel à "*useEffect*" pour l'exécuter seulement lorsque le prix total des produits change.
Cependant, vous réalisez, grâce à la ligne 12, que le code est exécuté 2 fois à l'accès initial de la page, mais plus jamais, et le total affiché n'est jamais mis à jour, même si on supprime correctement un produit du panier.
Quelle est la cause de ce comportement et comment peut-on le régler? (3 points)

b) Voici l'implémentation du component parent *App* et du component *Header* affiché en haut de toutes les pages du site. En vous basant sur ce code, est-ce que l'ordre des lignes 3,4,5 et 6 de l'arborescence de l'application est important ? **Justifiez**. (3 points)

```
1  export default function App() {
2    return (
3      <BrowserRouter>
4        <CartProvider>
5          <Header></Header>
6          <Routes>
7            <Route exact path="/" element={<ProductPage />} />
8            <Route path="/cart" element={<Cart />} />
9          </Routes>
10       </CartProvider>
11     </BrowserRouter>
12   );
13 }
14 const Header = () => {
15   const { state } = useContext(CartContext);
16   return (
17     <nav id='navbar'>
18       <Link to="/">Produits</Link>
19       <Link to="/cart">Panier({state.products.length})</Link>
20     </nav>
21   )
22 };
```

c) Voici l'implémentation de l'action "REMOVE" de votre *reducer* qui permet de retirer une unité de produit de votre panier (basé sur son attribut **id**). Assumez que le *payload* utilisé contient bel et bien un attribut *id* de type *number*.
Quel est le défaut avec cette implémentation ? (2 points)

```
case ACTIONS.REMOVE:
  return {
   products: state.products.filter((x) => x.id !== action.payload.id),
  };
```

a) Le hook useEffect permet d'effectuer des actions après le rendu du la composante, il s'exécute, après le rendu initial de la composante et à chaque fois que la composante est modifiée sauf si on a des dépendances, un tableau de variables à surveiller. La fonction de rappelle sera exécutée seulement si les dépendances ont changées depuis le dernier rendu (dans ce cas la seule dépendance est total). De ce que j'ai compris le useEffect sera seulement éxecuter après le rendu initial et quand on modifie total, mais total n'est jamais modifié, seulement le nombre de produits le sont, on ne fait jamais appel à setTotal hors de son hook ce qui fait qu'il n'est jamais mis à jour, je suggère pour régler cela que l'on mette un tableau vide pour les dépendances du useEffect pour qu'il soit mis à jour à chaque changement de la composante.

b) Oui, l'ordre des lignes 3,4, 5 et 6 est très important, cela permet de créer des interactions entre plusieurs composantes dans l'arborescence (relations parent-enfant). Par exemple, si la valeur injectée par le parent change, l'enfant se recharge avec la nouvelle valeur. Le component Header doit être enfant de BrowserRouter puisqu'on y référence à travers des liens, seulement le sous-arbre dans Routes sera modifié, un component différent sera chargé en fonction du chemin (path).

c) Le défaut avec cette implémentation est qu'elle est écrite comme si son but était de supprimer un produit plutôt que retirer une unité de produit, je pense que ça devrait plus être écrit ainsi:
case ACTIONS.REMOVE:

return { name: state.name, price: state.price, id: state.id, quantity: state.quantity- action.payload };

NOTE: Le seul problème, c'est que je ne sais pas comment ca marcherait pour le cas ou il ne reste qu'une seule unité d'un produit et donc il devrait être supprimer (peut être faire un appel à ACTIONS.DELETE si le quantity est égale à 0?)

Commentaire :
a) Cause bien expliquée. **(2/3)**

Toutefois, un tableau vide dans la liste de dépendances ne suffit pas, car cela implique un render seulement à la création du component. Or, nous voulons que le total soit modifié lorsqu'un produit est supprimé. Donc, pour palier ce problème, il faut gérer la liste des dépendances du useEffect en remplaçant [total] par [state] ou [state.products] pour modifier le total à chaque fois produit change.

b) Oui, en effet, mais votre réponse manque 1 élément: Header doit être enfant à CartProvider puisqu'on utilise state. **(2/3)**

c) Effectivement, la différence entre ACTIONS.REMOVE et ACTIONS.DELETE est saisie. **(2/2)**

Puis, dans cet exemple, ceci va supprimer toutes les instances du produit au lieu de décrémenter quantity de 1 (retirer un seul produit).

Question 3

Terminé

Note de 2,50 sur 8,00

Vous travaillez sur un service Web qui permet de gérer votre librairie de jeux vidéo. Votre service permet d'accéder à des informations sur la route /games et manipuler les données du serveur.
Vous voulez mettre en place un middleware qui permet de limiter le nombre d'accès aux données en utilisant un entête personnalisé "**X-LIMIT**".

Voici un extrait de l'implémentation de votre service avec quelques données dans l'objet "games" :

```
1
2  const express = require('express');
3  const cors = require('cors');
4
5  const app = express();
6  app.use(express.urlencoded({ extended: true }));
7  app.use(cors());
8
9  const games = [
10     { id: 100, title: "Half Life 3", rating: 99 },
11     { id: 101, title: "Diablo 5", rating: 40 },
12     { id: 102, title: "FIFA 2024", rating: 10 }];
13
14 app.use('/games', (req, res) => {
15     res.set("Access-Control-Expose-Headers", "X-LIMIT");
16     const limit = req.get("X-LIMIT");
17     if (!limit || limit - 1 < 0)
18         return res.status(403).send();
19     const newLimit = limit - 1;
20     res.set("X-LIMIT", newLimit);
21 });
22
23 app.get('/games', (req, res) => {
24     res.send(games);
25 });
26
27 app.get('/games/:id', (req, res) => {
28     const game = games.find(x => x.id === req.params.id);
29     if (!game) {
30         return res.status(404).send({ error: "Ce livre n'existe pas" });
31     }
32     res.send(game);
33 });
34
35 app.listen(3000, () => {});
36
```

a) Suite à la mise en place du middleware de limite, les requêtes invalides ou ayant atteint la limite retournent correctement 403, mais les requêtes valides ne sont jamais répondues. Quelle est la cause du problème et comment peut-on le corriger ? (2 points)

b) Après avoir corrigé le problème plus haut, un collègue vous fait remarquer que vous avez un défaut qui permet de contourner la limite de requêtes même après avoir épuisé votre limite initiale. Quel est ce défaut et comment peut-on le corriger? (3 points)

c) Vous remarquez qu'une requête invalide à GET /games/:id (ex : GET /games/abcd) retourne correctement 404, mais une requête valide (ex: GET /games/101) retourne également 404, même si un objet avec cet id existe. Quelle est la cause de ce problème et quel serait le code de retour pour une requête valide après la correction du problème? (3 points)

a) Les requêtes valides ne sont jamais répondues parce qu'on ne fait jamais appel à next(), la fonction qui permet de passer au prochain middleware, donc on ne fait que set la nouvel limite et on ne passe pas au traitement des requêtes.

b) Le défaut qui permet de contourner la limite de requêtes est que la limite est set par par la requete (ligne 15)

c) La cause de ce problème est que l'on n'utilise pas await ce qui veut dire qu'on ne laisse pas le temps de trouver le game en question et voila pourquoi le if est valider a chaque fois, le code de retour pour une requête valide après la correction du problème sera 200 (par défaut).

NOTE: on utilise find, je pensais que c'était une fonction exclusive à MongoDB et ici on utilise express

Commentaire :
a) Cause bien identifiée, mais la solution n'est pas fournie: Il faut appeler next() à la fin du middleware *use("/games")*. **(1/2)**

b) Pas trop claire. **(0.5/3)**

Ici, le nombre restant avant d'atteindre 0 limites est dans un entête dans la requête. Un client peut donc modifier l'entête et se donner toujours une valeur positive.

Une solution possible est d'utiliser un entête configuré par le serveur pour identifier l'utilisateur.

c) *await* n'a rien à voir avec cet exemple, car nous ne gérons pas des manipulations asynchrones. *games* est une simple liste d'objets JavaScript. Ainsi, le *find* n'opère pas en lien avec MongoDB.

Le problème est qu'on effectue une comparaison "x.id === req.params.id" dans le find, qui sera toujours fausse, car x.id est number et req.params.id est un string.

Code de retour 200 est bien. **(1/3)**

Question 4

Correct

Note de 1,50 sur 1,50

Voici un URI relatif d'une requête gérée par Express : **/capitals?country=Canada**.

Comment peut-on récupérer la valeur *Canada* à partir de l'URI pour l'utiliser dans le code ?

- a. req.params.country
- b. req.body.country
- c. req.values.country
- d. req.country
- e. req.query.country ✔

Votre réponse est correcte.

La réponse correcte est :
req.query.country

Question 5

Incorrect

Note de 0,00 sur 1,50

Vous avez utilisé un serveur dynamique et un serveur statique différents dans vos TPs 3 et 4. Est-ce que ça aurait été possible d'avoir les mêmes fonctionnalités avec 1 seul logiciel ?

- a. Oui, mais il faut configurer l'option CORS du serveur pour accepter des origines différentes. ✖
- b. Oui, un logiciel peut être un serveur statique et un serveur dynamique en même temps.
- c. Non, un serveur statique ne peut pas exécuter de logique de gestion comme un serveur dynamique.
- d. Oui, mais seulement si le logiciel est basé sur les événements comme NodeJS.
- e. Non, il faut quand même 2 logiciels différents.

Votre réponse est incorrecte.

La réponse correcte est :
Oui, un logiciel peut être un serveur statique et un serveur dynamique en même temps.

Question 6

Correct

Note de 2,00 sur 2,00

Voici l'implémentation de la gestion de la route "/search/:keyword" de recherche par mot clé dans une application Express. La fonction "search" retourne l'objet trouvé en fonction de la valeur reçue en paramètre ou *undefined* si rien n'a été trouvé.

```
app.get("/search/:keyword", (req, res, next) => {
  const result = search(req.params.keyword);
  if (!result) {
    res.status(404).send();
  }
  res.set("Content-Type", "application/json").send(result);
});
```

Vous faites la requête suivante : **GET /search/python** au serveur. Sachant que "python" ne fait pas partie des données sur le serveur, qu'est-ce qui va se passer ?

- a. Le client reçoit une réponse HTTP avec le code 404 et la valeur "undefined" dans le corps.
- b. Le client va lancer une erreur.
- c. Le client reçoit une réponse HTTP avec le code 200 et la valeur "undefined" dans le corps.
- d. Le client reçoit une réponse HTTP avec le code 200 et un objet vide dans le corps.
- e. Le client reçoit une réponse HTTP avec le code 404 et aucun corps. ✔
- f. Le client reçoit une réponse HTTP avec le code 404 et une réponse HTTP avec le code 200.
- g. Le serveur va lancer une erreur. ✔

Votre réponse est correcte.

Les réponses correctes sont :
Le client reçoit une réponse HTTP avec le code 404 et aucun corps.,

Le serveur va lancer une erreur.

Question 7

Correct

Note de 1,50 sur 1,50

Vous travaillez sur un service Web destiné à la gestion des réservations de plages horaires pour une salle de sport. Votre API, suivant les bonnes pratiques de REST, est disponible sur la route dédiée **/api.**

Parmi les routes suivantes et leur fonctionnement, laquelle est sémantiquement incorrecte ?

a. GET /api/reservations/:id : récupère une réservation en fonction de son identifiant.

b. GET /api/reservations : récupère toutes les réservations dans le système.

c. PUT /api/reservations : ajoute une nouvelle réservation au système. ✔

d. DELETE /api/reservations/oldest : supprime la réservation la plus ancienne du système.

e. PATCH /api/reservations/:id : fait une demande de changement d'heure d'une réservation en fonction de son identifiant.

Votre réponse est correcte.

Les réponses correctes sont :
DELETE /api/reservations/oldest : supprime la réservation la plus ancienne du système.,

PUT /api/reservations : ajoute une nouvelle réservation au système.

Question 8

Correct

Note de 1,50 sur 1,50

Voici le code suivant qui récupère des données d'un serveur distant et affiche le résultat dans la console. Assumez que les 3 lignes sont exécutées ensemble d'un seul coup.

```
const res = await fetch("https://myServer.com/api/data/1");
const json = res.json();
console.log(json);
```

Assumez que le serveur est accessible et retourne des objets avec le format suivant :

```
{
  "id" : string,
  "_x" : number,
  "_y" : number
}
```

Assumez que la route fournie est correcte.

Quel serait l'affichage dans la console ?

a. Une erreur sera lancée

b. Promise {<pending>} ✔

c.
```
{
  id : "abcd",
  _x : 1,
  _y : 2
}
```

d. null

e. Promise {<fulfilled>}

f. Aucun affichage dans la console

Votre réponse est correcte.

La réponse correcte est :
Promise {<pending>}

Question 9

Correct

Note de 2,00 sur 2,00

Pourquoi est-ce que les méthodes du pilote MongoDB de NodeJS et l'API Fetch du navigateur sont asynchrones ?

a. Ceci est un choix fait par l'équipe de développement du pilote et des navigateurs.

b. La communication avec un serveur externe doit être asynchrone pour ne pas bloquer le fil d'exécution du navigateur ou de NodeJS. ✔

c. La communication avec un serveur ou une base de données doit être asynchrone puisqu'elle se fait par HTTP.

d. C'est faux : seulement les méthodes de l'API Fetch sont asynchrones.

e. NodeJS impose à ce que toutes ces méthodes soient asynchrones.

Votre réponse est correcte.

La réponse correcte est :
La communication avec un serveur externe doit être asynchrone pour ne pas bloquer le fil d'exécution du navigateur ou de NodeJS.

Question 10

Correct

Note de 1,50 sur 1,50

Vous avez une instance MongoDB configurée avec une redondance sur 2 machines secondaires.

Vous envoyez une requête pour insérer un nouveau document dans une de vos collections et recevez la confirmation de l'insertion par MongoDB.

Vous envoyez rapidement, par la suite, une requête pour récupérer ce même document de votre base de données.

Entre les 2 requêtes, l'instance principale tombe en panne et une des deux instances secondaires est élue à sa place. Est-ce que vous allez obtenir le document ?

a. Oui, la redondance de MongoDB garantit que les données sont disponibles, même si une instance tombe en panne.

b. Ça dépend si la répartition des données a été complétée.

c. Non puisque MongoDB n'est pas configurée pour de la répartition des données.

d. Ça dépend si la réplication de l'opération a été complétée. ✔

e. Oui, pourvu que la clé de partition utilisée ne soit pas une clé de hachage.

f. Non puisque l'instance primaire est en panne.

Votre réponse est correcte.

La réponse correcte est :
Ça dépend si la réplication de l'opération a été complétée.

Question 11

Correct

Note de 1,50 sur 1,50

Parmi les fonctionnalités suivantes, laquelle n'est pas une fonctionnalité ajoutée par la librairie Express ?

a. Express permet l'utilisation de témoins signés lors de la communication HTTP.

b. Express permet le regroupement de plusieurs gestionnaires de routes ayant le même préfixe commun.

c. Express permet la gestion d'une requête HTTP à travers une chaîne de middlewares.

d. Express permet la lecture et écriture de fichiers du système de fichiers de la machine qui héberge le serveur. ✔

e. Aucune de ces réponses.

Votre réponse est correcte.

La réponse correcte est :
Express permet la lecture et écriture de fichiers du système de fichiers de la machine qui héberge le serveur.

Question 12

Correct

Note de 1,50 sur 1,50

Que sera affiché dans l'élément *h1* après avoir cliqué trois fois sur le bouton *Click Me*?

```
const App = () => {
  const [counter, setCounter] = useState(0);

  return (
    <div>
      <h1>{counter}</h1>
      <button onClick={() => (counter = counter + 1)}>Click Me</button>
    </div>
  );
};
```

a. 1

b. 3

c. 2

d. 0 ✔

e. undefined

Votre réponse est correcte.

La réponse correcte est :
0

Question 13

Partiellement correct

Note de 0,50 sur 2,00

Voici un appel à une fonction asynchrone fictive "validateAsyncData". La fonction prend en paramètre un objet et retourne une promesse avec les valeurs CODES.SUCCESS ou CODES.FAIL en cas de traitement réussi ou rejète la promesse avec la valeur de CODES.ERROR en cas d'erreur. (voir l'objet CODES plus bas)

```
1  const CODES = {
2    SUCCESS: "true",
3    FAIL: "false",
4    ERROR: "error"
5  }
6
7  const handleResult = (res) => {
8    switch (res) {
9      case CODES.SUCCESS:
10       return console.log('Validation réussie');
11     case CODES.FAIL:
12       return console.log('Validation échouée');
13     case ERROR:
14       return console.log('Erreur dans le traitement');
15   }
16 }
17
18 validateAsyncData({ id: 123 }).then(handleResult).catch(e => console.log(e));
19 validateAsyncData({ id: 123 }).then(handleResult);
20 validateAsyncData({ id: 0 }).then(handleResult).catch(e => console.log(e));
```

L'objet **{ id: 123 }** devrait résulter en une validation réussie et l'objet **{ id: 0 }** devrait résulter en une promesse rejetée.
En vous basant sur ces informations et le code fourni, lesquels des énoncés plus bas sont vrais ?

- [x] a. L'appel à catch() à la ligne 19 est inutile puisque ce cas est déjà géré dans la fonction handleResult. ✖
- [ ] b. Le résultat de l'appel à la ligne 20 est 'Erreur dans le traitement'.
- [x] c. Le résultat des appels aux lignes 18 et 19 sera le même. ✔
- [ ] d. L'appel à la ligne 19 est invalide puisqu'il manque un appel à .catch().
- [ ] e. Les appels des lignes 18 à 20 vont lancer une erreur puisque la syntaxe fournie est invalide.
- [ ] f. Les lignes 13 et 14 sont inutiles puisqu'une promesse rejetée ne se rendra jamais là.

Votre réponse est partiellement correcte.

Vous en avez sélectionné correctement 1.
Les réponses correctes sont :
Les lignes 13 et 14 sont inutiles puisqu'une promesse rejetée ne se rendra jamais là.,

Le résultat des appels aux lignes 18 et 19 sera le même.

Question 14

Partiellement correct

Note de 1,50 sur 2,00

Parmi les affirmations suivantes la ou lesquelles sont fausse(s) ?

- [x] a. Dans une architecture client-serveur, le même logiciel peut jouer les 2 rôles en même temps. ✖
- [ ] b. Une architecture orientée services est un variant de l'architecture client-serveur.
- [x] c. Une architecture orientée événements nécessite l'utilisation d'un serveur. ✔
- [ ] d. Un système peut utiliser plusieurs types d'architectures en même temps.
- [x] e. Une architecture pair-à-pair est toujours moins sécuritaire qu'une architecture orientée services. ✔

Votre réponse est partiellement correcte.

Vous avez sélectionné trop d'options.
Les réponses correctes sont :
Une architecture orientée événements nécessite l'utilisation d'un serveur. ,

Une architecture pair-à-pair est toujours moins sécuritaire qu'une architecture orientée services.

Question 15

Partiellement correct

Note de 0,50 sur 2,00

Vous envoyez la requête suivante pour récupérer la ressource "abc123" à 14:30 GMT, le 11 décembre 2023:

GET /data/abc123 HTTP/1.1
HOST : https://monserveur.ca
Accept : text/html,application/xml;q=0.8,image/webp,*/*;q=0.9,application/pdf;q=0.7

Le serveur a cette ressource en format XML, JSON et PDF seulement et vous envoie la réponse suivante:

HTTP/1.1 200 OK
Cache-Control : public,max-age=500
Expires : Mon, 11 Dec 2023 14:35:00 GMT
Last-Modified : Thu, 09 Nov 2023 16:18:16 GMT
Vary : Accept

Parmi les énoncés suivants, lesquels sont vrais ?

- [ ] a. Le serveur ne pourra pas retourner une ressource valide puisque le client s'attend à un document HTML
- [ ] b. La valeur de l'entête "Last-Modified" peut être utilisée pour l'entête "If-Last-Modified" dans un GET conditionnel
- [x] c. La version de la ressource en cache ne peut pas être utilisée si demandée à 14:36 GMT ✖
- [ ] d. La ressource envoyée par le serveur sera en format JSON
- [ ] e. La valeur de max-age est ignorée à cause de la valeur "public" de l'entête Cache-Control
- [x] f. La ressource envoyée par le serveur est en format XML ✖
- [ ] g. La version de la ressource en cache peut être utilisée si demandée à 14:36 GMT

Votre réponse est partiellement correcte.

Vous en avez sélectionné correctement 0.
Les réponses correctes sont :
La version de la ressource en cache peut être utilisée si demandée à 14:36 GMT ,

La ressource envoyée par le serveur sera en format JSON

Commentaire :

Question 16

Partiellement correct

Note de 1,50 sur 2,00

Vous avez une base de données MongoDB avec des documents qui représentent des jeux vidéo dans un agrégateur de revues en ligne :
En voici un exemple :

```
{
    "title": "Baldur's Gate 3",
    "score": 96,
    "platform": [
        "PC",
        "Xbox",
        "PS5"
    ],
    "date": "2023-09-03"
}
```

Voici le code suivant qui récupère les jeux disponibles sur la plateforme **PS5** ayant un score supérieur à **75** et affiche certaines informations dans la console :

```
1 const res = await collection.find(
2   { platform: "PS5", score: { $gt: 75 } }).toArray();
3 const mongo_response = res.map((item) => ({ title: item.title }));
4 console.log(mongo_response);
```

Assumez que l'objet **collection** représente une collection existante sur votre base de données.
Sélectionnez les réponses qui affichent la même chose que le console.log de la ligne **4**.

a.

```
1 const mongo_response = await collection.find(
2   { platform: "PS5", score: { $gt: 75 } },
3   { projection: { _id: 0, title: 1 } }).toArray();
4 console.log(mongo_response);
```

✔

b.

```
1 const mongo_response = ((await collection.find({}).toArray())
2   .reduce((acc, item) => {
3     if (item.platform.includes('PS5') && item.score > 75)
4       acc.push({ title: item.title });
5       return acc;
6   }, []));
7 console.log(mongo_response);
```

✔

c.

```
1 const mongo_response = (await collection.find({},
2   { projection: { _id: 0, title: 1 } }).toArray())
3   .filter((item) => item.platform.includes("PS5") && item.score > 75);
4 console.log(mongo_response);
```

✖

d.

```
1 const mongo_response = await collection.find(
2   { platform: "PS5", score: { $gt: 75 } },
3   { projection: { _id: 0, title: 1, date: 0, score: 0, platform: 0 }
4 }).toArray();
  console.log(mongo_response);
```

Votre réponse est partiellement correcte.

Vous avez sélectionné trop d'options.
Les réponses correctes sont :

```
1 const mongo_response = await collection.find(
2   { platform: "PS5", score: { $gt: 75 } },
3   { projection: { _id: 0, title: 1 } }).toArray();
4 console.log(mongo_response);
```

,

```
1 const mongo_response = ((await collection.find({}).toArray())
2   .reduce((acc, item) => {
3     if (item.platform.includes('PS5') && item.score > 75)
4       acc.push({ title: item.title });
5       return acc;
6   }, []));
7 console.log(mongo_response);
```

Question 17

Incorrect

Note de 0,00 sur 1,50

Parmi les énoncés suivants, lequel définit mieux ce qu'est une application monopage (*Single Page Application*) ?

a. Une application monopage est restreinte à une seule page HTML et un seul URI. ✖

b. Une application monopage est une application qui n'a pas besoin d'interagir avec un serveur dynamique.

c. Une application monopage utilise des pages Web générées par un serveur.

d. Une application monopage ne permet pas de partager les informations entre plusieurs pages du même site.

e. Une application monopage est une application qui ne recharge pas son contexte JavaScript après son chargement initial.

Votre réponse est incorrecte.

La réponse correcte est :
Une application monopage est une application qui ne recharge pas son contexte JavaScript après son chargement initial.