Commencé le	lundi 20 février 2023, 13:00
État	Terminé
Terminé le	lundi 20 février 2023, 14:21
Temps mis	1 heure 21 min
Note	**16,00** sur 25,00 (**64**%)

Description

LISEZ CES INSTRUCTIONS AVANT DE COMMENCER L'EXAMEN

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

Voici les règles pour l'évaluation:

- L'examen est noté sur un total de 25 points.
- La note partielle associée à chaque question est marquée à gauche de la question.
- Certaines questions requièrent un développement court, tandis que d'autres questions sont à choix multiples.
- Certaines questions demandent d'ajouter une justification à votre réponse. Une bonne réponse sans justification valide ne sera pas acceptée.
- 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 le contrôle périodique ! 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.

L'ensemble des notes de cours sont disponibles sur Moodle, dans la section Note de cours pour le contrôle pratique

Bon travail!

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.

Okay

Question **2**

Correct

Note de 1,00 sur 1,00

Quel(s) énoncé(s) parmi les suivants est(sont) vrai(s) pour l'élément HTML <form> ?

- a. Un formulaire HTML doit obligatoirement contenir au moins 1 élément <button> ou <input type=submit> pour être soumis.
- b. La balise <label> doit obligatoirement avoir un attribut **for** dont la valeur est la même que l'attribut **id** d'une balise <input> dans le même formulaire HTML.
- c. Certaines valeurs d'entré peuvent être vérifiées directement à travers les attributs HTML des bonnes balises. ✔
- d. Un formulaire HTML peut être soumis à un serveur sans écrire du code JavaScript supplémentaire. ✔
- e. La seule différence entre **method=GET** et **method=POST** dans une balise <form> est le besoin d'envoyer des données sensibles dans le formulaire.

Votre réponse est correcte.

Les réponses correctes sont :
Certaines valeurs d'entré peuvent être vérifiées directement à travers les attributs HTML des bonnes balises.,

Un formulaire HTML peut être soumis à un serveur sans écrire du code JavaScript supplémentaire.

Question **3**

Correct

Note de 2,00 sur 2,00

Indiquez le/les erreur(s) dans le code suivant :

```
1 <div>
2     <h1 id="title main-title">Quiz</h1>
3     <p id="question-1" class="question question-1">
4         <span class="bold">#1:</span>
5         Quel est le processus illustré dans le diagramme <italic>suivant?</italic>
6         <img src="assets/img/question_1.png"/>
7     </p>
8 <div>
```

- [ ] a. Un élément ne peut pas avoir un <id> et une <class> avec le même nom.
- [ ] b. L'élément <img> ne contient pas de balise fermante.
- [ ] c. Il manque des balises et autour du texte à la ligne 5.
- [x] d. L'élément parent ne contient pas de balise fermante. ✔
- [x] e. L'élément <italic> n'est pas une élément HTML valide. ✔
- [x] f. La syntaxe de l'attribut <id> de l'élément <h1> est invalide. ✔

Votre réponse est correcte.

Les réponses correctes sont : L'élément parent ne contient pas de balise fermante.,
La syntaxe de l'attribut <id> de l'élément <h1> est invalide.,

L'élément <italic> n'est pas une élément HTML valide.

Question **4**

Correct

Note de 1,00 sur 1,00

Un document HTML peut être lu et interprété au fur et à mesure pendant la lecture. Cependant, un document CSS doit être lu au complet avant de pouvoir être interprété. Pourquoi ?

a. CSS et HTML peuvent être interprétés durant la lecture. Seulement un fichier JS doit être lu au complet avant d'être exécuté.

b. Contrairement au CSS, un document HTML est défini comme un graphe.

c. Les règles CSS doivent être triées dans un ordre de priorité avant d'être appliquées. ✔

d. Les règles CSS sont appliquées sur des éléments du DOM, donc il faut que le DOM soit construit avant d'appliquer le CSS.

e. Le document CSS peut être interprété séquentiellement seulement s'il est chargé avec l'attribut **defer**.

Votre réponse est correcte.

La réponse correcte est :
Les règles CSS doivent être triées dans un ordre de priorité avant d'être appliquées.

Question **5**

Correct

Note de 1,00 sur 1,00

Soit le code CSS et HTML suivant:

```
<style>
  div > p {
    color: red;
  }

  body > div > p {
    color: green;
  }

  #container p {
    color: blue;
  }
</style>

<body>
  <div id="container">
    <p>Hello World!</p>
  </div>
</body>
```

Quelle sera la couleur du texte "Hello World!" ?

- a. La couleur dépend des paramètres du navigateur de l'utilisateur
- b. Vert
- c. Bleu ✔
- d. Rouge
- e. Nous n'avons pas assez d'information pour bien déterminer la couleur du texte

Votre réponse est correcte.

La réponse correcte est :
Bleu

Question **6**

Incorrect

Note de 0,00 sur 1,00

Soit le code HTML et CSS suivant:

```
<style>
  .container {
    display: flex;
    flex-direction: row;
    align-items: center;
  }

  .box {
    width: 50px;
    height: 50px;
    background-color: blue;
    margin: 10px;
  }
</style>

<div class="container">
  <div class="box"></div>
  <div class="box"></div>
  <div class="box"></div>
</div>
```

Quel sera l'alignement vertical des boîtes bleues par rapport à leur conteneur ?

- a. Les boîtes bleues ne seront pas positionnées par rapport au conteneur, elles seront positionnées de façon relative dans la page
- b. Les boîtes bleues ne seront pas positionnées par rapport au conteneur, elles seront positionnées de façon absolue dans la page ✖
- c. En haut du conteneur (Top)
- d. Au centre du conteneur (center)
- e. Au bas du conteneur (Bottom)

Votre réponse est incorrecte.

La réponse correcte est :
Au centre du conteneur (center)

Question **7**

Correct

Note de 2,00 sur 2,00

Soit le code HTML et CSS suivant:

```
<style>
  .container {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 50px 50px;
  }

  .box-1 {
    background-color: blue;
    grid-column: 1 / 3;
    grid-row: 1 / 3;
  }

  .box-2 {
    background-color: red;
    grid-column: 2 / 3;
    grid-row: 1 / 2;
  }

  .box-3 {
    background-color: green;
    grid-column: 2 / 3;
    grid-row: 2 / 3;
  }
</style>

<div class="container">
  <div class="box-1"></div>
  <div class="box-2"></div>
  <div class="box-3"></div>
</div>
```

Quelle sera la disposition des 3 boîtes de couleur à l'intérieur du conteneur (container) ?

a. La boîte bleue couvrira les première et deuxième colonnes des première et deuxième rangées, et les boîtes rouges et vertes seront empilées les unes sur les autres dans la troisième colonne des première et deuxième rangées

b. La boîte bleue couvrira les première et deuxième colonnes des première et deuxième rangées, et les cases rouge et verte seront côte à côte dans la deuxième colonne de la première rangée

c. La boîte bleue couvrira les première et deuxième colonnes des première et deuxième rangées, et les boîtes rouges et vertes seront empilées les unes sur les autres dans la deuxième colonne des première et deuxième rangées ✔

d. La boîte bleue couvrira toute la grille et les cases rouge et verte seront côte à côte dans la deuxième colonne de la première rangée

e. La boîte bleue couvrira les première et deuxième colonnes des première et deuxième rangées, et les cases rouge et verte seront côte à côte dans la troisième colonne de la première rangée

Votre réponse est correcte.

La réponse correcte est : La boîte bleue couvrira les première et deuxième colonnes des première et deuxième rangées, et les boîtes rouges et vertes seront empilées les unes sur les autres dans la deuxième colonne des première et deuxième rangées

Description

Voici un annexe de fonctions/méthodes JavaScript et des manipulations du DOM

Fonctions et méthodes JavaScript

Fonction/Méthode	Description
.length	Retourne la taille du tableau
.indexOf(valeur)	Retourne l'indice du tableau qui contient la valeur passée en paramètre. Sinon -1.
.push(valeur)	Ajoute un élément au tableau
.slice(begin, end)	Retourne un nouveau tableau à partir de la case à l'indice *begin* (obligatoire) jusqu'à l'indice *end* (optionnel).
.map(fonction)	Applique une fonction de transformation sur chaque élément du tableau et retourne un nouveau tableau
.filter(predicat)	Filtre les éléments du tableau en fonction du prédicat et retourne le résultat dans un nouveau tableau
.find(predicat)	Recherche le premier élément qui satisfait le prédicat. Sinon, retourne *undefined.*
.forEach(fonctionParc)	Applique la fonction *fonctionParc* sur chaque élément d'un tableau. Ne retourne rien
.sort(comparateur)	Retourne une nouveau tableau triée selon la fonction *comparateur*
Object.keys(obj)	Retourne toutes les clés de l'objet Javascript *obj*

Propriétés et méthodes de l'objet *document*

Fonction/Méthode	Description
getElementById()	Retourne l'élément possédant le ID passé en paramètre.
getElementsByTagName()	Retourne une collection contenant tous les éléments dont la balise correspond à celle passée en paramètre.
getElementsByClassName()	Retourne une collection contenant tous les éléments dont la classe correspond à celle passée en paramètre.
querySelector()	Retourne le premier élément correspondant au sélecteur CSS passé en paramètre.
querySelectorAll()	Retourne une collection contenant tous les éléments correspondant au sélecteur CSS passé en paramètre.
createElement()	Crée un élément correspondant à la balise passée en paramètre.
createTextNode()	Crée un nouveau noeud texte.

Propriétés et méthodes associées à un noeud N du DOM

Fonction/Méthode	Description
innerHTML	Contient tout le contenu HTML inclus dans un noeud.
childNodes	Collection contenant les noeuds fils, incluant les noeuds texte.
children	Collection contenant les noeuds fils, excluant les noeuds texte.
firstChild	Retourne le premier noeud fils
firstElementChild	Retourne le premier fils, en ne considérant pas les noeuds texte.
lastChild	Retourne le dernier noeud fils
lastElementChild	Retourne le dernier fils, en ne considérant pas les noeuds texte.
nextSibling	Retourne le noeud qui a le même parent que N et qui est situé immédiatement après.
nextElementSibling	Retourne le noeud qui a le même parent que N et qui est situé immédiatement après, en ne considérant pas les noeuds texte.
previousSibling	Retourne le noeud qui a le même parent que N et qui est situé immédiatement avant.

Fonction/Méthode	Description
previousElementSibling	Retourne le noeud qui a le même parent que N et qui est situé immédiatement avant, en ne considérant pas les noeuds texte.
parentNode	Retourne le noeud parent.
textContent	Retourne tout le contenu textuel d'un noeud (incluant ses descendants).
nodeType	Retourne le type d'un noeud. En particulier : 1 = noeud élément, 2 = noeud attribut, 3 = noeud texte.
style	Permet d'obtenir et de changer le style d'un élément. Exemple : monElement.style.color = blue.
className	Permet d'obtenir et de changer la classe d'un élément.
addEventListener(ev, gest, cap)	Associe un gestionnaire d'événement à un événement. Si le troisième argument est true le traitement se fait aussi au cours de la capture (pas seulement lors de la propagation vers le haut (bubbling).
appendChild()	Ajoute le noeud passé en paramètre après tous les noeuds fils de N .
insertBefore(nouv, ref)	Ajoute le noeud nouv juste avant le noeud ref , qui est un noeud fils de N .
remove()	Retire l'élément de l'arbre DOM

Question **8**

Terminé

Note de 2,50 sur 5,00

Voici le code d'une fonction JavaScript qui prend en paramètre un tableau et un prédicat (fonction qui retourne true/false).

```
1 function fun(arr, predicate) {
2   if (!arr || arr.length === 0) {
3     return undefined;
4   } else if (predicate(arr[0])) {
5     return arr[0];
6   } else {
7     return fun(arr.slice(1), predicate);
8   }
9 }
```

a) Expliquez ce que la fonction **fun** fait et quelle sera la valeur de retour si la fonction est appelée avec les paramètre suivantes : [1, 2, 3, 4], x => x%2 === 0 (1.5 points)

b) Quel est le problème potentiel avec la **ligne 7** du code ? Est-ce qu'il y a une manière d'éviter ce problème ? (1.5 points)

c) Que devez vous faire pour pouvoir utiliser cette fonction avec n'importe quel tableau (objet de type *Array*) sans avoir à le passer en paramètre ? Donnez les modifications à faire dans le code (2 points)

a) La fonction fun sert à retourner le premier élément de l'array qui satisfait le prédicat. Le retour de l'appel fun ([1, 2, 3, 4], x => x%2 === 0) sera donc 2.

b) Le problème est que la fonction fun est une fonction récursive. Le problème de ce genre de fonction est qu'elles peuvent facilement Overflow en cas de trop d'appels.

Commentaire :

a) 1,5/1,5

b) 1/1,5 Est-ce qu'il y a une manière d'éviter ce problème ?

c) 0/2

a) Si la fonction est appelée avec les paramètres [1, 2, 3, 4] et x => x % 2 === 0, elle parcourra le tableau et renverra la première valeur (le premier élément) qui est paire selon le prédicat. Dans ce cas, la valeur de retour sera 2 car c'est la première valeur paire dans le tableau.
b) Le problème potentiel avec la ligne 7 du code est la récursion sans fin si le prédicat n'est jamais satisfait pour aucun élément du tableau. Cela entraînerait un débordement de pile (stack overflow). Pour éviter ce problème, il serait préférable d'ajouter une condition pour gérer le cas où la récursion continue indéfiniment.
c) Pour pouvoir utiliser cette fonction avec n'importe quel tableau sans avoir à le passer en paramètre, vous pouvez ajouter la fonction "fun" en tant que méthode du prototype de l'objet Array:

```
Array.prototype.fun = function(predicate) {
  if (this.length === 0) {
    return undefined;
  }

  let index = 0;
  while (index < this.length) {
    if (predicate(this[index])) {
      return this[index];
    }
    index++;
  }

  return undefined;
};
```

Question **9**

Correct

Note de 1,00 sur 1,00

Voici un extrait de code JavaScript. La ligne 7 lance une exception "**TypeError: Object prototype may only be an Object or null: undefined**" Pourquoi ?

```
1 function Car(){
2     this.name = "Honda";
3     this.make = "Civic";
4     this.year = 2004;
5 }
6 const car = Car();
7 const honda = Object.create(car);
8 console.log(honda); // {}
```

a. La syntaxe de création d'objet dans la fonction Car() est invalide.

b. Object.create() ne fonctionne pas avec un objet qui n'est pas créé par une fonction constructeur

c. Object.create() ne fonctionne pas avec un objet qui est créé par une fonction usine

d. Il manque le retour "return this;" dans la fonction Car()

e. Il manque le mot clé "new" lors de l'appel de Car() à la ligne 6 ✔

Votre réponse est correcte.

La réponse correcte est :
Il manque le mot clé "new" lors de l'appel de Car() à la ligne 6

Question **10**

Incorrect

Note de 0,00 sur 1,00

Soit la fonction JavaScript suivante:

```
function divide(a, b) {
  if (b === 0) {
    throw new Error('Cannot divide by zero');
  }
  return a / b;
}

module.exports = divide;
```

Et soit le test Jest suivant:

```
const divide = require('./divide');

describe('divide', () => {
  it('divides two numbers', () => {
    expect(divide(4, 2)).toBe(2);
  });
});
```

Quelle sera la couverture du test sur la fonction par rapport aux instructions et aux branches?

- a. 100% de couverture pour les instructions et 25% de couverture pour les branches
- b. 100% de couverture pour les instructions et 75% de couverture pour les branches
- c. 33.33% de couverture pour les instructions et 25% de couverture pour les branches
- d. 0% de couverture pour les instructions et 25% de couverture pour les branches
- e. 66.66% de couverture pour les instructions et 25% de couverture pour les branches
- f. 100% de couverture pour les instructions et 50% de couverture pour les branches
- g. 33.33% de couverture pour les instructions et 50% de couverture pour les branches
- h. 100% de couverture pour les instructions et 100% de couverture pour les branches
- i. 0% de couverture pour les instructions et 100% de couverture pour les branches
- j. 66.66% de couverture pour les instructions et 50% de couverture pour les branches
- k. 66.66% de couverture pour les instructions et 75% de couverture pour les branches ✖

Votre réponse est incorrecte.

La réponse correcte est :
66.66% de couverture pour les instructions et 50% de couverture pour les branches

Question **11**

Partiellement correct

Note de 0,75 sur 1,00

Paul I. Technique a un différent avec son ami Éric T. Supérieur. Paul explique à son ami Éric qu'il est possible de réutiliser des tests d'intégrations pour faire des tests de régressions. Éric n'est pas d'accord.

Qui a raison?

- [x] a. Paul a partiellement raison puisque c'est possible de réutiliser les tests d'intégrations, mais seulement lorsque c'est des tests en boîte blanche ✖
- [ ] b. Éric a raison puisque c'est les tests unitaires qui devraient être utilisés pour les tests de régression
- [ ] c. Paul a partiellement raison puisque c'est possible de réutiliser les tests d'intégrations, mais seulement lorsque c'est des tests en boîte noire
- [ ] d. Éric a raison, réutiliser des tests existants n'est pas une bonne pratique et ça fausse les résultats
- [x] e. Paul a raison puisque les tests de régressions devraient réutiliser des tests existants ✔

Votre réponse est partiellement correcte.

Vous avez sélectionné trop d'options.
La réponse correcte est :
Paul a raison puisque les tests de régressions devraient réutiliser des tests existants

Question **12**

Terminé

Note de 1,00 sur 4,00

Le navigateur utilise une "boucle d'événements" pour gérer l'exécution du code JS. Cette boucle utilise un seul fil d'exécution et il est très important de ne pas le "bloquer" avec de longues fonctions puisque l'expérience utilisateur en serait grandement impactée.

Nous pouvons attacher un gestionnaire d'événement à un élément du DOM. Comment est-ce possible que le navigateur "écoute" continuellement pour un événement et exécute du code JS sans que l'écoute ne bloque le fil d'exécution principal ?

Expliquez le mécanisme qui permet au navigateur d'ajouter un gestionnaire d'événement à un élément du DOM sans bloquer le fil d'exécution principal. Décrivez le fonctionnement complet pour l'ajout du gestionnaire d'événement et le déclenchement de l'événement.
**Attention, ici, on cherche à comprendre le fonctionnement du mécanisme, pas le nom de la fonction à utiliser pour attacher un événement (addEventListener).

Le navigateur est capable d'écouter continuellement grâce aux appels aux WebAPIs exécutés sur des fils autres que le fil principal de JS. Cela se fait par la création de fonctions asynchrones. Un traitement long dans ces fils ne bloque pas le fil principal. À la fin de l'exécution, les méthodes asynchrones appellent le *callback* passé en paramètre avec le résultat de leur traitement. Ceci est une tâche, elle est exécutée à chaque tout de boucle et est toujours exécutées après le code synchrone (stack vide)

(Voir diapo 76 du cours sur DOM)

Commentaire :

Sujet	Point
Attacher l'event dans la stack	0/1
listening de l'event sur webAPI	0.5/1
Evenement déclanche task dans callback queue	0/1
Quand stack est vide, l'event loop envoie la task dans la stack	0.5/1
BONUS: Distinction de la task et rendering queue	0/0.5

Question **13**

Correct

Note de 1,00 sur 1,00

Soit le code HTML suivant:

```
<button id="myButton">Click me!</button>
```

Et le code JavaScript suivant:

```
function clicking(){
  console.log('Button clicked!');
}

window.onload = (event) => {
  const button = document.getElementById('myButton');
  button.addEventListener('click', clicking());
};
```

Que se passe-t'il lorsque le bouton est cliqué?

a. Le bouton disparaît de la page

b. Le texte "Button clicked!" apparaît dans la console

c. Une erreur est lancé parce que le gestionnaire d'événement n'est pas attaché correctement au bouton

d. Le texte "Button clicked!" apparaît sur la page

e. Rien ne se produit ✔

Votre réponse est correcte.

La réponse correcte est :
Rien ne se produit

Question **14**

Incorrect

Note de 0,00 sur 1,00

Soit le code HTML suivant:

```
<ul id="myList">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

Et soit le code JavaScript suivant:

```
window.onload = (event) => {
  const list = document.getElementById('myList');
  const secondItem = list.getElementsByTagName('li')[1];
  const newItem = document.createElement('li');
  newItem.innerHTML = 'Item 4';
  list.insertBefore(newItem, secondItem);
};
```

Qu'est-ce qui se passera au chargement de la page web?

a. Une erreur est générée car la méthode insertBefore ne peut pas être utilisée sur un élément

b. Il ne se passe rien ✖

c. Une erreur est généré puisque la méthode getElementsByTagName peut seulement être utilisée sur l'objet document.

d. Un nouvel élément <li> est ajouté à la fin de l'élément <ul>, contenant le texte "Item 4"

e. Un nouvel élément <li> est inséré avant le deuxième élément <li> dans l'élément <ul>, contenant le texte "Item 4"

Votre réponse est incorrecte.

La réponse correcte est :
Un nouvel élément <li> est inséré avant le deuxième élément <li> dans l'élément <ul>, contenant le texte "Item 4"

Question **15**

Terminé

Note de 2,75 sur 3,00

Jean Untel développe actuellement une application révolutionnaire sur un sujet top secret. Il a besoin de vos conseils d'experts pour le développement de son site web. Il souhaite sauvegarder des données utilisateurs directement dans le navigateur.

Il veut sauvegarder les données suivantes:

1. Le thème de couleur de préférence de l'utilisateur. Thème foncé ou clair (Dark and light theme) pour la personnalisation du site.
2. Le nom d'utilisateur pour faciliter les connexions futures.
3. Le mot de passe de l'utilisateur pour faciliter les connexions futures.

Jean se demande si c'est une bonne idée.

a) Expliquez le problème avec l'idée de Jean. (1 point)

Jean décide d'aller de l'avant et d'implémenter le storage local sur son site web malgré vos objections.

b) Devrait-il utiliser le localStorage ou le sessionStorage? Expliquez votre choix. (2 points)

a)

Pour le point 1 : On peut déjà noter qu'enregistrer les préférences de l'utilisateur est la déclarations la plus importante que peut faire faire par l'utilisateur. Il faudra donc prendre en compte que cela influera directement dans la déclaration du CSS. C'est-à-dire que le CSS sera impacter par ce choix. Il faut y faire attention.

Pour le point 2 et 3 : L'autre problème est un problème de sécurité. Le Storage directement sur le navigateur n'est pas sécurisé. Cela pose un problème d'autant que c'est un sujet top secret. De plus, sa taille de stockage sera limité à environ 5MB (dépend du navigateur).

b)

Il devrait utiliser **localStorage** parce qu'il veut que l'information persiste, c'est-à-dire qu'elle soit toujours là même même s'il ferme son navigateur. Il veut que quand il réouvre son navigateur les informations personnelles soit toujours enregistrée directement dans le navigateur.

Commentaire :

a) Le mot de passe est problématique.

b) Excellent.

Aller à…

Introduction ►