Commencé le	lundi 21 février 2022, 12:45
État	Terminé
Terminé le	lundi 21 février 2022, 14:45
Temps mis	1 heure 59 min
Note	**11,25** sur 25,00 (**45**%)

Description

LISEZ CES INSTRUCTIONS AVANT DE COMMENCER L'EXAMEN

Le contrôle pratique est composé de 5 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 possède **14** questions notées 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.
- 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 pratique ! 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**

Non répondue

Non noté

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

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

Question **2**

Incorrect

Note de 0,00 sur 1,00

Voici un extrait de la fonction *generateQuestion* qui permet de générer le contenu HTML d'une question sur une plateforme de quiz en ligne.

```
function generateQuestion() {
  const question = document.createElement("div");
  question.dataset.questionId = "id-1";
  question.dataset.category = "category-HTML";
  question.dataset.multipleChoice = "false";
  // reste de la configuration
  document.body.appendChild(question);
}
```

Suite à l'exécution de la fonction, quel sera le contenu de l'élément généré par la fonction ?

- a. ✖
- b. Aucun contenu. Le tiret dans "id-1" et "category-HTML" rend le code erroné : les tirets sont réservés pour les noms des attributs du dataset.
- c.
- d.
- e. Aucun contenu. La majuscule dans "category-HTML" rend le code erroné : pas de majuscules après un tiret dans un dataset.
- f.

Votre réponse est incorrecte.

La réponse correcte est :

Question **3**

Partiellement correct

Note de 0,83 sur 1,00

Quelles affirmations suivantes sont vraies par rapport à div et span ?

Veuillez choisir au moins une réponse. **Attention à votre sélection** : un mauvais choix aura un impact sur la note totale de la question.

Veuillez choisir au moins une réponse :

- [x] a. div et span ne sont pas des balises sémantiques ✔
- [ ] b. div est une balise en-ligne, alors que span est une balise bloc
- [ ] c. span est une balise sectionnante, contrairement à div
- [ ] d. div et span peuvent être utilisés interchangeablement
- [x] e. span est une balise en-ligne, alors que div est une balise bloc ✔
- [x] f. div est un conteneur sémantique qui regroupe des éléments communs, alors que span n'offre aucune sémantique aux éléments ✖

Votre réponse est partiellement correcte.

Vous avez sélectionné trop d'options.
Les réponses correctes sont :
div et span ne sont pas des balises sémantiques,

span est une balise en-ligne, alors que div est une balise bloc

Question **4**

Incorrect

Note de 0,00 sur 1,00

Voici le contenu de la balise <html> de votre fichier index.html

```
<head>
    <script src="script.js"></script>
</head>
<body>
    <p id="my-paragraph"></p>
</body>
```

Votre code JS tente de récupérer la balise <p> à travers la méthode document.getElementById("my-paragraph") de votre page web et modifier son contenu, mais vous recevez toujours une erreur comme l'élément paragraphe n'existe pas. Quelle est la raison la plus probable de ce problème ?

a. Le script est executé avant l'ajout du paragraphe dans le DOM.

b. La balise <p> est vide, donc elle n'est jamais ajoutée au DOM ✖

c. Ceci n'est pas une manière valide de charger un script, il faut passer par l'attribut href.

d. Le "-" dans l'attribut id de la balise <p> est invalide et rend l'élément non accessible.

Votre réponse est incorrecte.

La réponse correcte est :
Le script est executé avant l'ajout du paragraphe dans le DOM.

Question **5**

Terminer

Note de 1,00 sur 2,00

Est-ce que les deux sélecteurs CSS suivants pourraient s'appliquer au même élément :

div * p

div ~ p

Si oui, **expliquez** en donnant un **exemple** de code HTML. Si non, **expliquez pourquoi**.

```
<div>
<p>paragraph</p>
</div>
```

oui, car * est un sélectionneur universel et ~ **est un sélectionneur de voisins qui doit être au même niveau. Dans ce cas, les 2 sélectionnerons le même élément car il est le seul élément qui peut être sélectionner dans div et qui est au même niveau que div. Donc div * p le sélectionne et div ~ p aussi.**

Commentaire :

Question **6**

Partiellement correct

Note de 0,50 sur 1,00

Les bonnes pratiques de créations de pages avec HTML et CSS proposent de séparer les deux, donc aucune utilisation de l'attribut style dans le HTML.

Parmi les affirmations suivantes, laquelle ou lesquelles justifient cette pratique ?

- [x] a. Meilleure gestion de la spécificité des sélecteurs utilisés ✔
- [ ] b. Permet de changer la mise en forme de la page sans affecter le contenu
- [x] c. Évite la duplication de code CSS. ✔
- [ ] d. Permet d'avoir plusieurs mise en forme pour plusieurs tailles d'écran

Votre réponse est partiellement correcte.

Vous en avez sélectionné correctement 2.
Les réponses correctes sont :
Évite la duplication de code CSS.,

Permet de changer la mise en forme de la page sans affecter le contenu,

Meilleure gestion de la spécificité des sélecteurs utilisés,

Permet d'avoir plusieurs mise en forme pour plusieurs tailles d'écran

Description

Voici un annexe de fonctions/méthodes Javascript et 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.
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).

Fonction/Méthode	Description
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 **7**

Correct

Note de 3,00 sur 3,00

Implémentez la fonction *find* qui est similaire à la méthode *find* pour manipuler des tableaux Javascript. La fonction demandée retourne le premier élément qui satisfait un prédicat passé en paramètre, sinon renvoie *undefined*.

Gérez le cas où le tableau est vide ou *undefined*.

CONTRAINTES:
Vous ne pouvez pas utiliser la méthode *find* existante, car je vous demande de l'implémenter.
Respectez le principe d'immutabilité.

Des points seront donnés en fonction de l'élégance du code.
Vous avez autant d'essais que nécessaires pour faire fonctionner le code.

Par exemple:

Test	Résultat
`console.log(find([1,4,3], x => x > 2))`	4
`console.log(find([1,4,3], x => x > 5))`	undefined

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```
/**
 * Applique un prédicat sur chaque élément d'un tableau et renvoie le premier
 * éléments qui satisfait le prédicat. Sinon, renvoie undefined.
 * Cette fonction existe déjà, mais réimplémentez la.
 *
 * @example
 * find([1,4,3], x => x > 2) // renvoie 4
 * @example
 * find([1,4,3], x => x > 5) // renvoie undefined
 *
 * @param {Array<T>} arr
 * @param predicate - Fonction de rappel
 * @returns {T} premier élément trouvé ou undefined si rien n'est trouvé
 */
function find(arr, predicate)
{
    if (!arr)
        return undefined;
    for (let i of arr)
    {
        if (predicate(i))
            return i;
    }
```

	Test	Résultat attendu	Résultat obtenu	
✔	`console.log(find([1,4,3], x => x > 2))`	4	4	✔
✔	`console.log(find([[1, 2], [3, 4], [5, 6, 7]], x => x.length === 3))`	[5, 6, 7]	[5, 6, 7]	✔
✔	`console.log(find(undefined, v => v.length === 3))`	undefined	undefined	✔
✔	`console.log(find([]))`	undefined	undefined	✔
✔	`console.log(find([1,4,3], x => x > 5))`	undefined	undefined	✔

	Test	Résultat attendu	Résultat obtenu	
✔	`console.log(find([{id: 1, isResult:false},{id:2, isResult:true},{id:3, isResult:false}], x => x.isResult).id)`	2	2	✔

Tous les tests ont été réussis ! ✔

Solution de l'auteur de la question (Nodejs):

```
function find(arr, predicate) {
  if (!arr || arr.length === 0) {
      return undefined;
  } else if (predicate(arr[0])) {
      return arr[0];
  } else {
      return find(arr.slice(1), predicate);
  }
}
```

Correct

Note pour cet envoi : 3,00/3,00.

Question **8**

Terminer

Note de 1,00 sur 3,00

Voici 2 manipulations différentes sur le même tableau :

```
const arr = [-5, 6, 2, 0, -3, 4];
const option1 = arr.filter((x) => x > 0).map((x) => x * x);
const option2 = arr.reduce((x, y) => {
  if (y > 0) {
    x.push(y * y);
  }
  return x;
}, []);
```

Est-ce que les objets **option1** et **option2** ont le même contenu ? Si oui, quelle manipulation est plus efficace et **pourquoi?** Si non, quelle est la différence entre le résultat des 2 manipulations et **pourquoi?**

Oui, ils ont le même contenu qui est de [36, 4, 16] et la manipulation la plus efficace serait l'option1 car elle demande seulement d'un seule parametre qui est x et retourne automatiquement le tableau. option2 a besoin de 2 paramètres et doit envoyer la multiplication des y dans x et doit retourner x contrairement a loption1 qui utilise seulement x.

Oui elle retourne le meme contenu, c'est à dire un nouveau tableau avec que les valeurs superieurs à 0 maintenant au carré mais option1 utilise la méthode filter pour filtrer les éléments positifs, puis map pour mettre au carré chaque élément. C'est une approche plus déclarative et concise.
option2 utilise la méthode reduce pour effectuer les deux étapes. Dans la fonction de rappel de reduce, elle filtre les éléments positifs en poussant seulement ceux qui sont positifs dans le tableau x, puis elle met au carré chaque élément positif.
Option 2 est plus efficase car option 1 fait plusieurs copies de tableaux.

Commentaire :
Option 2 est plus efficace. Option 1 fait des copies de tableaux.

Question **9**

Incorrect

Note de 0,00 sur 1,00

Voici un extrait de code JavaScript. La sortie de la console à la ligne 10 est {} (Objet vide). Pourquoi ?

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

a. Il manque le mot clé "new" lors de l'appel de Car() à la ligne 8

b. Les attributs "name","make" et "year" sont sur le Prototype de l'objet "car"

c. Les attributs "name","make" et "year" sont sur le Prototype de l'objet "honda"

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

e. C'est faux, la ligne 10 affichera { name: 'Honda', make: 'Civic', year: 2004} dans la console

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

Votre réponse est incorrecte.

La réponse correcte est :
Les attributs "name","make" et "year" sont sur le Prototype de l'objet "honda"

Question **10**

Terminer

Note de 3,00 sur 3,00

Les objets suivants retournent des références vers les mêmes éléments du DOM.

a) **Quelles sont les différences** entre les 2 objets (byCN et byQS) ?

```
const byCN = document.getElementsByClassName("recette");
const byQS = document.querySelectorAll(".recette");
```

b) Les 2 objets sont déclarés comme "**const**". Est-ce qu'il aura une modification dans un ou les deux objets si j'ajoute un élément avec la classe "*recette*" à ma page web. Si oui, **lequel(s)** et **pourquoi** ? Si non, **pourquoi** ? **Justifiez** votre réponse.

Il n'y aura aucune une difference dans l'option 2 car querySelectorAll(".recette") indique de selectionner tous éléments correspondant au sélecteur CSS passé en paramètre et puisque le CSS ne sera pas modifier il n'y aura pas de modification.Mais pour l'option 1 il y aura une modification car elle prend tous les éléments dont la classe correspond à celle passée en paramètre et puisque un element a ete ajouter avec la classe recette alors celle-ci sera modifiee aussi.

a) byCN : document.getElementsByClassName("recette") renverra un objet HTMLCollection en tant que résultat qui est statique. byQS : document.querySelectorAll(".recette") renverra un objet NodeList en tant que résultat qui est dynamique.
b) Si vous ajoutez un élément avec la classe "recette" à votre page web, cela affectera uniquement l'objet byQS, pas byCN.

Commentaire :

Question **11**

Terminer

Note de 0,00 sur 3,00

Voici 3 manières différentes de rajouter un gestionnaire d'événement sur un bouton :

<!-- Dans index.html -->

```
<button id="btn">Cliquez moi! </button>
```

// Dans script.js

```
const button = document.getElementById("btn")
```

Option #1

```
button.onclick = function onClick() { console.log(this.innerHTML) }
```

Option #2

```
button.addEventListener('click', () => console.log(this.innerHTML) )
```

Option #3

```
function onClick() { console.log(this.innerHTML) }

button.addEventListener("click", onClick)
```

a) Les 3 manières ne donnent pas le même résultat. **Pourquoi** ?

b) Est-ce qu'il y a une autre différence entre l'utilisation d'une fonction fléchée (#2) et une fonction déclarée (#3) pour la gestion des événements?

Justifiez votre réponse.

a)Ils ne donnent pas le même résultat, car dans l'option 2, le gestionnaire est la fonction fléché et dans la 3 la fonction onclick qui est une fonction évènement est le gestionnaire et l'option 1 associe une fonction à un attribut de l'élément. Alors ils ne donneront pas tous le meme resultat a cause de leurs difference.

b)Oui, car le gestionnaire dans l'option 3 appel l'évènement onClick et non pas la fonction demandé comme gestionnaire contrairement a l'option 2 qui a un gestionnaire qui est la fonction fléchée

Commentaire :

a) Les trois méthodes de gestion des événements ne donnent pas le même résultat en raison de la manière dont elles gèrent le contexte ("this") et la réutilisation de la fonction.
Option 1: La fonction de gestion d'événement est définie en tant que fonction anonyme. Lorsque la fonction est exécutée, "this" se réfère au bouton lui-même. Par conséquent, "this.innerHTML" renverra le texte à l'intérieur du bouton ("Cliquez moi!").
Option 2: Elles héritent du contexte "this" de leur parent, qui est généralement le contexte global (window dans le navigateur). Par conséquent, "this.innerHTML" renverra "undefined" car il n'y a pas de "innerHTML" dans le contexte global.
Option 3: Lorsque l'événement est déclenché, "this" se réfère au bouton car il est défini comme le contexte de l'appel à "addEventListener". Par conséquent, "this.innerHTML" renverra le texte à l'intérieur du bouton.
b) la gestion des événements réside dans la gestion du contexte "this" comme mentionné précédemment. Cela a un impact sur la réutilisation de la fonction d'événement. Avec une fonction fléchée, vous ne pouvez pas accéder au contexte de l'élément qui a déclenché l'événement. En revanche, une fonction déclarée vous permet d'accéder au contexte de l'élément qui a déclenché l'événement, ce qui le rend plus flexible pour la réutilisation avec différents éléments.

Question **12**

Partiellement correct

Note de 0,17 sur 1,00

Votre entreprise décide d'adopter la syntaxe *ESM* pour votre prochain projet d'application web qui implique un site web ainsi qu'un serveur dynamique utilisant NodeJS.

Parmi les suivantes, quels sont les contraintes principales d'ESM ?

- a. ESM est supporté seulement par les navigateurs
- b. ESM n'offre pas la fonctionnalité de minification de code ✔
- c. Les modules ES doivent obligatoirement être servis par un serveur HTTP
- d. ESM est incompatible avec des outils de *bundling* tels que *Webpack* ✖
- e. ESM n'est pas compatible avec les navigateurs, contrairement à CJS

Votre réponse est partiellement correcte.

Vous en avez sélectionné correctement 1.
Les réponses correctes sont :
Les modules ES doivent obligatoirement être servis par un serveur HTTP,

ESM n'offre pas la fonctionnalité de minification de code

Question **13**

Incorrect

Note de 0,00 sur 1,00

Vous êtes aux prises avec le code source d'un site web de mauvaise qualité, parce que le choix initial de l'architecture était incorrect. Il s'ensuit que la maintenance de ce logiciel est difficile, parce qu'il est difficile de comprendre le code et de le modifier adéquatement.

Comment est-ce que les tests peuvent aider à corriger ce problème de qualité ? Sélectionnez toutes les réponses valides.

- a. Les tests ne peuvent pas corriger ce problème de qualité.
- b. Aucune de ces réponses n'est valide.
- c. Les tests peuvent aider à trouver des bogues dans l'architecture du logiciel. ✖
- d. Les tests peuvent aider à trouver des bogues dans les exigences du logiciel. ✖
- e. Les tests peuvent aider à trouver des bogues dans la documentation du logiciel.

Votre réponse est incorrecte.

La réponse correcte est :
Les tests ne peuvent pas corriger ce problème de qualité.

Question **14**

Terminer

Note de 1,75 sur 4,00

Voici le code de la fonction "**shrinkCoordinates**" qui permet de manipuler un tableau de coordonnées à l'aide d'un facteur de modification. Chaque coordonnée est un objet avec les attributs "x" et "y" : **{ x, y }**. Le facteur peut prendre n'importe quelle valeur réelle, mais la valeur absolue du facteur est toujours utilisée. Le facteur permet de réduire ou agrandir la valeur des positions "x" et "y" de chaque coordonnée du tableau. Dans le cas d'une entrée invalide pour le tableu de coordonnées, la fonction retourne un tableau vide.

```
function shrinkCoordinates(coordinates, factor) {
  const f = Math.abs(factor);
  const newCoordinates = [];
  if (!coordinates) return newCoordinates;
  for (const coord of coordinates) {
    const newCoord = {
      x: coord.x / f,
      y: coord.y / f,
    };
    newCoordinates.push(newCoord);
  }
  return newCoordinates;
}
```

a) Quels sont les tests en boîte noire qui permettent de tester adéquatement le comportement de la fonction ? Donnez les entrées et les sorties attendues pour chaque test. (3 points)

b) Cette fonction contient un défaut potentiel. Quel est ce défaut ? Proposez une manière de le gérer. (1 point)

a)test1: < {undefined, 1} , {newCoordinates} >

test2: < {{ **2, 4** }, -2} , { **{ 1, 2 }**} >

test2: < {{ **2, 4** }, 2} , { **{ 1, 2 }**} >

b) Le défaut de cette fonction est que les coordinates sont non iterables car ce sont des objets. La façon de gérer ce defaut potentiel serait de faire des tests unitaires sur le for loop et de faire que la fonction coordinates soit iterable

Commentaire :

a) coordiantes est un tableau d'objets. Mauvaise notation JS : [{ x: 1, y: 1}, {x: 2, y: 2 }]

Manque un test avec factor === 1

Manque un test avec factor entre 0 et 1

b) coordinates est un tableau, pas un seul objet ou une fonction

Aller à...

Introduction ▶