

LOG4420

Conception de sites web dynamiques et transactionnels

Automne 2018

Contrôle périodique

Enseignant : Konstantinos Lambrou-Latreille

Cet examen comporte 3 questions

Durée : 2 heures

Pondération : 25% de la note finale

Directives :

Toute documentation permise.

Calculatrice non programmable permise.

Jeu de traduction Jeu de traduction Jeu de traduction

Mot:

Le mot CHIEN existe. Traduis-le.

☐ dog ☐ cat ☐ rabbit

(a) Mot inconnu

Mot:

Le mot CHIEN existe. Traduis-le.

☐ dog ☒ cat ☐ rabbit

Essaie encore...

(b) Mot trouvé + traductions

Mot:

Le mot CHIEN existe. Traduis-le.

☐ dog ☒ cat ☐ rabbit

Essaie encore...

(c) Choix d'une mauvaise réponse

Jeu de traduction

Mot:

Le mot CHIEN existe. Traduis-le.

☒ dog ☐ cat ☐ rabbit

Bravo!!!

(d) Choix d'une bonne réponse

Jeu de traduction

Mot:

Le mot CHAT existe. Traduis-le.

☐ bird ☒ cat ☐ lion

Bravo!!!

(e) Autre mot dans la BD.

FIGURE 1 – Affichages pour la question 1.

1 Javascript (10 points)

Nous voulons développer un jeu de traduction simple. La figure 1 représente un exemple d’affichage. Le programme à développer doit permettre à un joueur de choisir un mot en français et de sélectionner la traduction adéquate en anglais. Le mot en français est inséré dans un champ de texte, et les traductions possibles sont affichées dans des boutons radios. S’il choisit la bonne réponse, on affiche un message positif, sinon un message négatif. À tout moment, le joueur peut écrire un autre mot français et d’autres options de traductions deviendront disponibles.

Nous vous fournissons le code HTML. Vous ne pouvez pas le modifier.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Jeu de traduction</title>
  </head>
  <body>
    <h1>Jeu de traduction</h1>
    <div>
      <label for="mot">Mot: </label>
      <input type="text" name="mot" id="mot" value="chie">
    </div>

    <div>
      <p>Le mot <span></span> existe. Traduis-le.</p>

      <!-- <div id="possibleTranslations"> -->
      <!--   <input type="radio" name="traduction" id="dog" value="dog" checked> -->
      <!--   <label for="dog">Dog</label> -->
      <!--   <input type="radio" name="traduction" id="cat" value="cat"> -->
      <!--   <label for="cat">Cat</label> -->
      <!--   <input type="radio" name="traduction" id="rabbit" value="rabbit"> -->
      <!--   <label for="rabbit">Rabbit</label> -->
```

```

    <!-- </div> -->

    <p class="message bad">Essaie encore...</p>
    <p class="message good">Bravo!!!</p>
  </div>
</body>

<script src="node_modules/jquery/dist/jquery.min.js"></script>
<script src="js/main.js" type="module"></script>

</html>

```

Nous vous fournissons les fichiers Javascript incomplets suivants :

```

////////////////////////////////////
// Fichier main.js
////////////////////////////////////
import getTranslationsTheme from './translationsTheme.js'
import getTranslations from './model.js'

const $ = window.$

// À IMPLÉMENTER.
// Appelez ici 'initView' lorsque le DOM est chargé.

// À IMPLÉMENTER.
// État initial de la vue.
function initView () {
  // Cachez le <div> qui est 2ieme fils de <body>
  // Cachez <p class="message">

  // Gérez l'événement lorsqu'on tape sur une touche du clavier.
  // Associez la fonction de rappel 'enteredInputWordEvent' à
  // l'événement 'keyup' pour l'élément <input type="text" name="mot" id="mot">
}

/**
 * À IMPLÉMENTER.
 * Fonction de rappel lorsqu'on insère un caractères dans le 'input[type=text]'.
 *
 * @param {Event} event - Object événement
 */
function enteredInputWordEvent (event) {
  // Cachez le <div> qui est 2ieme fils de <body>
  // Cachez <p class="message">
  // Retirez du DOM la balise <div id=possibleTranslation>

  // Récupérez l'entrée de <input type="text" name="mot" id="mot"> et
  // appelez 'getTranslations' pour récupérer les traductions possibles
  // du mot dans le modèle.

  // Une fois la traduction récupérée, appelez 'getTranslationsTheme' pour
  // obtenir un objet DOM de la section de traductions possibles
  // Insérez cet objet au bon endroit dans le DOM.
  // Affichez le <div> qui est 2ieme fils de <body> (qui était caché)
  // Écrire le mot en entrée dans le <span> en majuscule.
}

/**
 * À IMPLÉMENTER.
 * Fonction de rappel lorsqu'on sélectionne une traduction du mot.
 * Si c'est le bon choix, on affiche le message de succès.
 * Sinon, le message d'erreur.
 * Attention à comment cette fonction est utilisée dans 'getTranslationsTheme'

```

```

*
* @param {Translation} translation - Object de type Translation
* @param choiceIndex - Index de la traduction sélectionnée
* @returns {Function<Event>} - Fonction de rappel pour l'événement
*/
// function selectedTranslationEvent ...

////////////////////////////////////
// model.js
////////////////////////////////////
import Translation from './translation.js'

const $ = window.$

const url = 'http://localhost:3000/translations'

/**
 * À IMPLÉMENTER.
 * Récupère les traductions possible d'un mot du serveur.
 *
 * La fonction de rappel en paramètre permet de renvoyer le résultat.
 * Si on reçoit un code 200 avec les traductions du serveur, alors
 * on appelle 'callback' avec les traductions comme premier paramètre.
 * Si on reçoit un code 404, alors on appelle 'callback' avec la
 * valeur 'null' comme premier paramètre.
 *
 * Un exemple d'objet retourné par le serveur serait pour le mot 'chien':
 * { word: 'chien', choices: ['dog', 'cat', 'rabbit'], correct: 0 }
 *
 * @param {String} word - Mot à traduire
 * @param callback - Fonction de rappel qui retourne les traductions
 * @returns undefined
 */
export default function getTranslations (word, callback) {
}

////////////////////////////////////
// translationTheme.js
////////////////////////////////////
const $ = window.$

/**
 * À IMPLÉMENTER.
 * Récupère le thème HTML des traductions possibles.
 *
 * Par exemple, pour le mot 'chien', il génère le HTML suivant:
 * <div id="possibleTranslations">
 *   <input type="radio" name="traduction" id="dog" value="dog" checked>
 *   <label for="dog">Dog</label>
 *   <input type="radio" name="traduction" id="cat" value="cat">
 *   <label for="cat">Cat</label>
 *   <input type="radio" name="traduction" id="rabbit" value="rabbit">
 *   <label for="rabbit">Rabbit</label>
 * </div>
 *
 * Associez l'événement 'change' sur chaque <input> pour qu'il puisse
 * exécuter la fonction 'selectTranslationEvent'.
 *
 * @param {Translation} translation - Objet de type Translation
 * @param {Event} selectTranslationEvent - Fonction de rappel qui s'exécute
 * lorsqu'on sélectionne un 'input'.
 * @returns - Le HTML selon le format de votre choix
 */

```

```

export default function getTranslationsTheme (translation, selectTranslationEvent) {
}

////////////////////////////////////
// translation.js
////////////////////////////////////
/**
 * @param {String} word - Mot à traduire
 * @param {Array<String>} choices - Choix possibles de traduction
 * @param {Number} correct - Indice de la bonne traduction à partir de 'choices'
 */
export default function Translation (word, choices, correct) {
  this.word = word
  this.choices = choices
  this.correct = correct
}

```

Supposez qu'il existe un service web qui renvoie la traduction d'un mot à partir de l'URL `http://localhost:3000/translations/:word`. Par exemple, une requête GET à l'URL `http://localhost:3000/translations/chien` renvoie le JSON suivant :

```
{ word: 'chien', choices: ['dog', 'cat', 'rabbit'], correct: 0 }
```

Vous devez compléter les fichiers Javascript fournis. Référez-vous aux commentaires dans le code. Voici les requis :

- (2 points) Implémentez la fonction `getTranslations`
- (2 points) Implémentez la fonction `getTranslationsTheme`
- (2 points) Implémentez la fonction `selectTranslationEvent`
- (2 points) Implémentez la fonction `enteredInputWordEvent`
- (1 point) Implémentez la fonction `initView`
- (1 point) Écrivez l'instruction qui appelle `initView` dès que le DOM est chargé.

2 HTTP (4 points)

- (1 point) Un serveur contient un fichier compressé sous deux formats : `gzip` et `br`. Un client veut récupérer ce fichier, donc il envoie une requête HTTP qui contient l'en-tête suivante :

```
Accept-Encoding: br;q=1.0,gzip;q=0.5
```

La réponse retournée par le serveur est le fichier en format `gzip`. Expliquez, en une courte phrase, ce qui s'est passé.

- (1 point) Il est 14h et vous demandez une page web à un serveur qui vous renvoie la réponse HTTP suivante (max-age est en secondes) :

```

HTTP/1.1 200 OK
Cache-Control: must-revalidate,max-age=600
Etag: "4sef1-fja3-h3110"

```

À 14h30, le serveur tombe en panne, mais vous lancez tout de même une requête conditionnelle. Recevez-vous le contenu en cache ou bien un code 404 ? Justifiez.

- c) (1 point) Dans quel contexte obtient-on un code HTTP 201 vs un code HTTP 200 ? Citez un cas concret.
 - d) (1 point) Expliquez la négociation de contenu contrôlée par le client.
- a) L'utilisateur préfère obtenir le format br, si possible, ensuite le format gzip. Le serveur a traité la demande, mais n'offre pas la compression br, donc il renvoie le fichier en format gzip.
- b) La ressource est expirée, donc il faut une requête conditionnelle pour savoir si on prend la ressource en cache. Avec 'must-revalidate', on doit attendre d'avoir la réponse du serveur avant d'utiliser la ressource en cache. Puisque le serveur est hors-ligne, un code 404 reçoit renvoyé.
- c) On reçoit un code 201 lorsqu'on a fait une requête POST qui a créé une nouvelle ressource sur le serveur.
- d) Le client fait une requête vers une page, et le serveur renvoie toutes les ressources possibles. C'est au client de choisir la bonne page selon les préférences de l'utilisateur.

3 HTML et CSS (6 points)

- a) (1 point) Quelle est la différence entre les balises sémantiques article et section ?

La balise article est utilisée pour regrouper du contenu indépendant qui peut être réutilisé dans une autre page. La section peut ne pas être réutilisée dans une autre page, mais elle doit contenir des titres pour être utilisée dans le sectionnement.

- b) (1 point) Pour une section d'un blog qui contient un ensemble de commentaires, quelle balise sémantique est la plus adaptée pour regrouper les meta-données d'un commentaire telles que la date de création, l'auteur, etc. ?

footer

- c) (1 point) Donnez la spécificité du sélecteur CSS suivant :

```
div#main > .message.important ~ p:first-child
```

0132

- d) (1 point) Soit les déclarations CSS suivantes spécifiées dans des origines différentes :

```
/* Déclaré dans les feuilles de style du navigateur */
.message { color: red; } /* 0 */
p { color: green; } /* 1 */

/* Déclaré dans les feuilles de style de la page web */
p !important { color: yellow; } /* 2 */
#message { color: blue; } /* 3 */
p !important { color: white; } /* 4 */

/*
 * Déclaré dans les feuilles de style spécifié par l'utilisateur
 * du navigateur (userContent.css)
 */
p !important { color: black; } /* 5 */
p.message { color: gray; } /* 6 */
```

Triez en ordre de priorité les déclarations précédentes qui seront appliquées sur le HTML suivant (utilisez les numérotations des déclarations) :

```
<p class="message">...</p>
```

5 4 2 6 0 1

- e) (2 points) Soit le code HTML suivant :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>CSS</title>
  </head>
  <body>
    <main>
      <article>
        <h1>Article</h1>
        <p>Voici un article sur Polytechnique de Montréal.</p>
        <p>Un premier paragraph...</p>
```



(a) Affichage initial



(b) Affichage voulu

FIGURE 2 – Affichages pour la question 3e)

```

    <p>Et puis un deuxième...</p>
    <div id="logo"></div>
  </article>
</main>
</body>
</html>

```

Écrivez le CSS qui permet de passer de 2a vers 2b.

```

main {
  display: flex;
}

article {
  position: relative;
  margin: 0 auto;
  border: solid black 1px;
  padding: 10px;
}

#logo {
  position: absolute;
  top: 0;
  right: 0;
}

```