Évaluation

Durée : 1h30 - Documents autorisés

**Calcul mental…**

Le paragraphe qui suit décrit le fonctionnement que l'on souhaite implémenter. Lisez attentivement ce dernier ; les parties que vous avez à réaliser sont exposées ensuite :

*Il s'agit d'écrire une application avec un serveur node qui écoute le port 8080 et un client envoyant des messages au serveur via un navigateur web (on suppose qu'il n'y a qu'**un seul utilisateur** de l'application).*
*Le scénario est le suivant :*

- *Le client demande la consultation de la page index.html ; celle-ci contient une question de la forme « additionne les nombres suivants 12 3 45 100 34 34 », une zone de saisie pour la réponse ainsi qu'un élément « timer » pour faire un compte à rebours de 30 secondes (pour l'instant, celui-ci affiche 30 et n'a pas encore démarré).*
- *Le compte à rebours démarre 29, 28, … 0 (décompte de 1 à chaque seconde). Le client a 30 secondes pour renvoyer une réponse au serveur. Attention, c'est le serveur qui chronomètre pas le client.*
- *Si le client répond correctement et en moins de 30 secondes, le serveur renvoie « bravo » au client, sinon il lui renvoie « perdu » (soit dans la même page, soit dans une nouvelle page – à vous de choisir).*

Les scripts côté client sont écrits en javascript et les scripts côté serveur en node.js (vous pouvez utiliser les modules node de votre choix).

**TRAVAIL DEMANDÉ**

**1 – Écriture du serveur : les routes**

Écrire un programme Node.js qui crée un serveur http qui écoute les requêtes sur le port 8080 (envoi du fichier **`index.html`** ou de « **`file not found`** ») et qui peut dialoguer par web sockets.

Dans un premier temps, le fichier index.html contient toujours la même question (*« additionne les nombres suivants 12 3 45 100 34 34 »*). Dans la version finale, les 6 nombres à additionner devront changer.

**2 – Écriture du serveur : le décompte**

Écrire le code côté serveur qui envoie toutes les secondes la nouvelle valeur du timer (et qui s'arrête à 0). Utiliser la méthode **`setInterval(f, ms)`** de l'objet **`global`**.

**3 – Génération de la question :**

Écrire une fonction qui génère aléatoirement 6 entiers compris entre 1 et 100 et qui renvoie un tableau contenant ces 6 valeurs.

Utiliser cette fonction pour générer la question envoyée à l'utilisateur.

**4 – Test avec Mocha** :

Ajouter un module personnel Utilitaire.js qui contient une méthode **`verifier(somme, tableau)`** qui renvoie **`true`** si la somme passée en paramètre est égale à la somme des valeurs de tableau.

Tester unitairement la méthode (la méthode de test doit être fournie avec l'application).

**5 – Contrôle de la réponse :**

Ajouter dans le fichier index.html le code pour renvoyer la réponse du client au serveur (soit par soumission d'un formulaire, soit par websockets).

Utiliser **`verifier(somme, tableau)`** pour répondre au client « **`gagné`** » ou « **`perdu`** ».

**6 – Ajout d'une image :**

Ajouter une image dans le fichier index.html.