Université de Strasbourg
Programmation Web 1
Licence 1 - UFR Mathématique - Informatique
Printemps 2019-20

Projet – Mise en œuvre d'un jeu de rendu de monnaie

Projet à réaliser individuellement

Objectifs du projet

- Mobiliser les connaissances acquises en cours et TP pour mettre en œuvre une application web basique
- Utiliser la documentation pour trouver les informations pertinentes pour répondre à un problème donné

Instructions

- Lisez le sujet en entier (plusieurs fois) pour bien identifier les différents points sur lesquels vous devez travailler
- Ne faites que ce qui est demandé dans le sujet. Le hors sujet ne rapporte pas de point, même si vous montrez que vous êtes capable d'envoyer une fusée sur Mars (ça n'est pas ce qui est évalué dans cette UE) : pour vous évaluer, on coche "fait" ou "pas fait" pour chaque élément demandé dans le sujet
- Certains développements vous demanderont potentiellement de chercher des informations dans la documentation : vous devez montrer que vous savez trouver les informations pertinentes pour résoudre un problème donné

Votre travail (fichiers `.html`, `.css` et `.js`) est à remettre au plus tard le 22 mai 2020 à 20h CET sur l'espace Moodle prévu à cet effet.

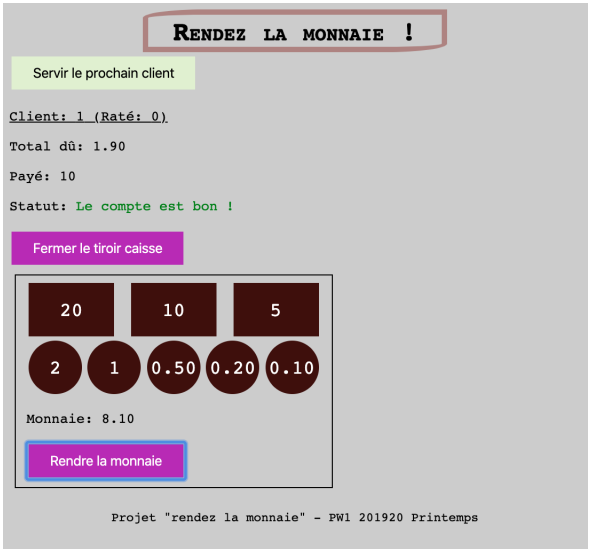

FIGURE 1 – Exemple d'aperçu du jeu "Rendez la monnaie !"

Rendre la monnaie ?

L'objectif de ce travail est de mettre en œuvre un petit jeu de rendu de monnaie : des clients effectuent des achats pour une certaine somme, vous donnent de l'argent et vous devez leur rendre la monnaie en accédant à votre caisse enregistreuse.

Ce travail comporte trois partie :

1. Mise en place d'une structure HTML ($\approx$ 6 points)
2. Mise en place d'un style CSS ($\approx$ 6 points)
3. Ajout des comportements de l'application avec Javascript ($\approx$ 8 points)

1 Structure HTML

Votre structure HTML doit contenir les éléments suivants :

1. Un titre
2. Un bouton pour passer au client suivant
3. Quelques informations :
   - ▷ Le numéro du client et le nombre de rendu de monnaie ratés
   - ▷ Le total à payer par le client
   - ▷ La somme remise par le client
   - ▷ Le statut du rendu (non traité / le compte est bon / le compte n'est pas bon)
4. Un bouton pour ouvrir le tiroir caisse
5. La caisse, qui contient :
   - ▷ de la monnaie (billets : 20, 10 et 5 Brouzoufs et pièces : 2, 1, 0.50, 0.20 et 0.10 Brouzoufs)
   - ▷ la monnaie que vous avez déjà récupérée
   - ▷ un bouton pour remettre sa monnaie au client
6. Un pied de page

2 Styles CSS

Les styles à appliquer à votre documents sont les suivants :

1. Le texte doit être écrit en `monospace`
2. Vous devez spécifier un style pour vos boutons :
   - ▷ appliquez une couleur de fond
   - ▷ appliquez un *padding* et une marge
   - ▷ appliquez une couleur de fond différente lorsque la souris passe sur le bouton
3. La caisse :
   - ▷ est non affichée par défaut
   - ▷ a une bordure
   - ▷ les éléments "billets" et "pièces" sont respectivement rectangulaires et ronds
4. Vous devez définir une *media query* pour passer en mode "mobile". Dans ce cas :
   - ▷ le titre de la page et le bloc "caisse" occupent en largeur 2/3 du *viewport*
   - ▷ la caisse est définie comme *flex* dont les items sont justifiés avec de l'espace autour.

3 Comportements Javascript

3.1 Accueil d'un client

Un client arrive à la caisse : il faut initialiser la somme dûe par le client et celui-ci va vous donner de l'argent sur lequel il faudra rendre la monnaie.

→ **Mettez en œuvre la fonction `function init()` contenant les opérations décrites ci-après.**

Total dû : on va générer aléatoirement la somme due. Pour cela, utilisez la fonction suivante :

```
function alea(min,max){
  min = Math.ceil(min);
  max = Math.floor(max);
  brouzoufs = Math.floor(Math.random() * (max - min)) + min;
  cents     = Math.floor(Math.random() * (10 - 0)) / 10.0;
  return brouzoufs+cents;
}
```

et modifiez l'élément HTML "`Total dû`" pour y faire apparaitre la somme générée.

Somme remise par le client : pour spécifier cette donnée, vous allez simplement utiliser la somme à régler et l'arrondir à la dizaine supérieure (par exemple, si le client doit payer 0.50 Brouzoufs, il donne 10 Brouzoufs) en utilisant l'expression `Math.ceil(total_du / 10) * 10`. Là aussi, il faut modifier l'élément HTML "`Payé`" pour y faire apparaitre la somme générée.

Reste ensuite à spécifier l'affichage des éléments suivants :

1. Dans le statut, on affiche "`Non traité`" (car le client attend sa monnaie). Le statut doit être affiché en noir.
2. La caisse doit être fermée : il faut modifier la propriété CSS `display` de l'élément HTML correspondant pour masquer ce dernier.
3. Le bouton pour ouvrir la caisse doit afficher "`Ouvrir le tiroir caisse`"
   Indication : pour accéder à la valeur d'un bouton défini par exemple par `<input type="button id="bouton" value="le texte de mon bouton">`, vous pouvez faire, en Javascript `document.getElementById("bouton").value = "du texte";` ou, en jQuery : `$("#bouton").attr("value", "du texte")`
4. La monnaie récupérée dans la caisse doit être mise à 0 (on n'a pas encore commencé à récupérer la monnaie)

Service en cours : tant que vous n'avez pas rendu la monnaie au client courant, vous ne pouvez pas appeler le client suivant :

1. Définissez comme variable globale (c'est-à-dire en dehors de tout bloc ou fonction) le booléen `var service_en_cours = true;`
2. Dans la fonction `init`, passez ce booléen à `true` et modifiez le bouton pour servir le prochain client comme suit : le bouton doit afficher "`Service en cours`" en gris et en italique.

3.2 Ouverture du tiroir caisse

Pour ouvrir la caisse, il faut ajouter, au bouton dédié dans la structure HTML, un écouteur d'événement `click`. Le comportement à exécuter lorsque cet événement est déclenché est le suivant : si la caisse n'est pas déjà ouverte, alors on change la valeur du bouton à "`Fermer le tiroir caisse`" et la propriété CSS `display` du bloc HTML correspondant à la caisse est spécifiée de telle sorte que la caisse est maintenant visible. Si la caisse est ouverte, recliquer sur le bouton fermera celle-ci : la valeur du bouton repasse à "`Ouvrir le tiroir caisse`" et la propriété CSS `display` du bloc HTML correspondant à la caisse est spécifié de telle sorte que celle-ci ne soit pas visible.

Indication : pour tester si la caisse enregistreuse est déjà ouverte, vous pouvez accéder à la valeur du bouton et la comparer (simplement en utilisant `==`) avec les valeurs attendues lorsque celle-ci est ouverte / fermée.

3.3 Faire de la monnaie

Quand le tiroir caisse est ouvert, il faut piocher dedans pour récupérer la monnaie nécessaire pour le client. Quand vous cliquez sur un billet ou une pièce, il faut récupérer la somme associée à l'élément cliqué et l'additionner à la somme déjà en main.

Indication : en supposant que vous avez défini les valeurs numériques à manipuler comme ceci : `<p>Monnaie: <span id="monnaie_collectee">0</span></p>`, la valeur numérique dans le *span* est accessible, en Javascript avec :

```
var val = document.getElementById("monnaie_collectee").textContent;
val = Number.parseFloat(val); /* pour une conversion en entier: Number.parseInt() */
```

ou en jQuery avec :

```
var val = $("#monnaie_collectee").text();
val = Number.parseFloat(val);
```

3.4 Rendre la monnaie récupérée

Quand vous (pensez que vous) avez la bonne somme à rendre, un clique sur le bouton "`Rendre la monnaie`" permet de rendre la monnaie au client. Lorsque l'on rend la monnaie, on affiche si celle-ci est correcte : on peut calculer la somme à rendre en faisant `payé - total dû` et comparer cette valeur avec la monnaie enregistrée :

- Si la différence est inférieure à la monnaie rendue, alors le statut affichera, en rouge, "`Rendu plus que nécessaire !`" et le nombre de rendus ratés sera incrémenté de 1
- Si la différence est supérieure à la monnaie rendue, alors le statut affichera, en rouge, "`Il manque de la monnaie !`" et le nombre de rendus ratés sera incrémenté de 1
- Sinon, le statut affiche en vert "`Le compte est bon`".

Dans tous les cas :

- le client courant a sa monnaie, le service en cours est terminé : passez à `false` le booléen `service_en_cours` et modifier le bouton pour servir le prochain client : celui-ci doit afficher "`Servir le prochain client`" en noir et sans italique.
- on incrémente le numéro de client (si on s'est trompé, tant pis ! on veut passer au client suivant)

3.5 Client suivant !

Pour appeler le client suivant, il faut cliquer sur le bouton affichant "`Servir le prochain client`". Lorsque l'on clique sur ce bouton, si le booléen `service_en_cours` est faux alors il faut appeler la fonction `init` (sinon on ne fait rien : il faut d'abord rendre sa monnaie au client courant).