Javascript (2) : Javascript dans la page web

Vous répondrez aux questions présentes dans le sujet dans le fichier `README.txt` présent à la racine du dossier contenant les fichiers pour le TP.

Exercice 1 : Premiers branchements javascript sur une page web

Récupérez le mini site exemple présent sur UPdago `MiniSite.zip`. Dans cet exercice, vous allez regarder le fichier `index.html` situé à la racine du site et le fichier `interaction.ts` présent dans le répertoire `Script/TypeScript`. Ce fichier a été transpilé JavaScript et le résultat de cette transpilation est le fichier `Script/JavaScript/interaction.js`

a. Regardez le code du fichier HTML, et notez dans le fichier README.txt où se fait le lien avec le fichier javascript (ligne du fichier HTML) ;

b. Pour mémoire, les fonctions javascript sont appelées en réaction à des événements ou à des actions de l'utilisateur. Elles sont ici invoquées dans des attributs des balises.
Repérez dans le code HTML, les appels de fonctions javascript et les événements auquels elles sont associées. Indiquez les lignes du fichier HTML concerné ainsi que les événements en question.
Déclenchez ces événements sur la page web ouverte dans votre navigateur et observez leurs effets.

c. Examinez le code typescript fourni et, en vous inspirant des exemples, complétez les fichiers HTML et javascript pour obtenir les comportements suivants :

1. lorsqu'on clique sur l'élément <**input /**> une fenêtre surgissante apparaît (`window.alert()`) affichant des informations sur le navigateur utilisé (`navigator.userAgent`) et la résolution de l'écran (`window.screen.width` et `window.screen.height`).
2. lorsque la page a fini de se charger (attribut `onload` de la balise <**body**>), l'heure est affichée dans le champ de texte mais à la place du texte "et des brouettes" c'est le nombre de minutes (méthode `getMinutes()` des objets Date) qui apparait, de sorte que le nombre de minutes soit affiché sur 2 chiffres, par exemple 19h01, 10h09, 13h34...
3. lorsqu'on clique sur le lien d'envoi de mail, un entier est tiré aléatoirement entre 0 et 100 et l'adresse de destination (`href`) change et devient
`sylvie.alayrangues@univ-poitiers.fr?subject=valeur` où valeur est l'entier aléatoirement obtenu. Ainsi, si on a tiré la valeur 5, le contenu de href devient
`sylvie.alayrangues@univ-poitiers.fr?subject=5`
Pour obtenir un nombre aléatoire entre 0 compris et 1 exclu, on utilise la méthode `Math.random()`.
Pour obtenir la partie entière d'un nombre k, on appelle `Math.floor(k)`.
4. lorsqu'on réalise un double-clic sur la première image (attribut `ondblclick`), ses attributs `src` et `alt` sont échangés avec ceux de l'autre image.
Le tableau document.images contient toutes les images présentes dans une page, la case document.images[0] contient la première image présente dans la page, la case document.images[1] la deuxième, etc. Attention, on ne peut pas affecter une nouvelle valeur à une case de ce tableau mais on peut modifier les propriétés des objets présents dans chaque case. Ainsi `document.images[0].alt="nouveau texte alternatif";` *remplacera le texte alternatif de la première image présente dans la page par* `nouveau texte alternatif`.

Exercice 2 : Calculette

Le but de l'exercice est d'utiliser des fonctions TypeScript / JavaScript pour transformer la page `calculatrice.xhtml` en calculette sommaire. Comme dans tous les exercices, on vous demande de répondre précisément aux questions posées et pas de créer une calculatrice élaborée en copiant-collant du code présent en ligne ou généré par une IA.

a. Questions préliminaires pour découvrir les "outils" dont vous aurez besoin dans la suite de l'exercice. Elles sont à réaliser, sauf mention expresse du contraire, dans la console du navigateur.

Dans la console, on écrit directement du JavaScript. La principale différence avec TypeScript est qu'il ne faut pas préciser les types des variables que l'on manipule.

Vous allez ici travailler dans la *console web* de Firefox. Pour mémoire, elle est accessible :

- soit via le sous-menu *Outils de développement Web* du menu *Outils du navigateur* (si la barre de menu de Firefox est affichée),
- soit dans le sous-menu *Outils de développement web* de l'élément *Outils supplémentaires* accessible après avoir cliqué sur l'icone composée de 3 barres horizontales en haut à droite de la fenêtre,
- via une combinaison de touches qui dépend de la machine utilisée (elle est mentionnée dans les menus précédents).

La console web vous permet de taper du code et de l'évaluer facilement en utilisant le raccourci CTRL+ENTRÉE ou COMMAND+ENTRÉE selon votre système d'exploitation. Le résultat de l'évaluation est affiché dans une autre fenêtre, qui se trouve par défaut à droite de la console.

Si vous voulez évaluer seulement une partie des instructions présentes dans la console, il vous faut sélectionner le code en question avant de lancer l'évaluation avec le raccourci.

Pour mémoire, le code javascript s'exécute dans l'environnement fourni par votre navigateur, et notamment dans l'environnement fourni par la page web sur laquelle est ouverte la console web.

Ainsi, si vous gardez bien la page `calculatrice.html` ouverte dans Firefox et la console ouverte à côté, cette dernière va vous permettre d'interagir avec la page et de la modifier dynamiquement. Notez bien que la modification est temporaire : dès que vous rechargez la page, les modifications réalisées dans la console disparaissent. Si vous ne comprenez pas bien pourquoi, demandez à vos encadrants de TP.

- *Utilisation de la méthode* `eval` :

  Testez une à une les instructions javascript suivantes et observez leur effet :

  - `eval("3+5");`
  - `eval("3/0");`
  - `eval("7*4");`

  Déduisez-en une des utilisations de la méthode `eval`.

- *Utilisation de la méthode* `document.getElementById(param)` *qui prend en paramètre l'identifiant d'un élément contenu dans la page HTML et renvoie l'objet javascript correspondant.*

  Créez une variable qui contienne l'objet javascript correspondant à l'élément de la page d'identifiant *res* (il s'agit de l'unique balise <**input** /> du document).

  Avec une instruction javascript, faites afficher le texte de votre choix dans cette zone de saisie [1]. Vous utiliserez pour cela l'attribut *value* de l'élément input correspondant.

  Changez maintenant le style du bord de l'élément via l'attribut *border* de l'attribut *style* de l'objet pour le rendre rouge en pointillés :

  ```
  res.style.border = "dashed red";
  ```

- *Utilisation de la méthode* `document.getElementsByTagName(param)` *qui prend en paramètre le nom d'une balise HTML, par exemple, "button" et renvoie une liste d'objets javascript contenant tous les éléments de la page correspondant à cette balise.*

  Mémorisez dans une variable la liste de tous les boutons de la page.

  Parcourez cette liste avec une boucle `for` pour que les textes de tous les boutons soient écrits en bleu (attribut *color* de l'attribut *style* de chaque objet).

1. notez au passage qu'un script peut tout à fait écrire dans une zone de saisie "readonly"

Recopiez le code dans la procédure `updateColorButton()` présente dans le fichier `Script/TypeSCript/t` et modifiez-le pour qu'il explicite le type de la variable recevant la collection d'éléments "button". Le type en question est `HTMLCollectionOf<HTMLButtonElement>`.

Transpilez le fichier `test.ts`. Vérifiez que le fichier `Script/JavaScript/test.js` contient bien la procédure que vous venez d'écrire.

Rechargez la page `calculatrice.xhtml` dans le navigateur.

Appelez la procédure dans la console : `updateColorButton();`.

Notez bien que cela ne fonctionne que parce que le fichier `test.js` est inclus dans le fichier `calculatrice.xhtml`.

- *Utilisation du mot clé* `this` *qui permet de manipuler l'objet javascript correspondant à l'élément HTML sur lequel a été appelée la méthode.*

  Complétez dans le fichier `test.ts` la procédure `updateFontSize(o : HTMLElement) : void` pour qu'elle fixe à "16pt" la taille de la police de caractères de l'objet, *o*, qui lui est passé en paramètre `o.style.fontSize="16pt";`.

  Modifiez le code HTML pour qu'elle soit exécutée vec le paramètre `this` lorsqu'on clique sur le bouton "=".

b. Revenons maintenant à la calculette. En vous aidant des éléments vus dans les questions précédentes, complétez le fichier `calculatrice.ts` pour qu'il fournisse :

- une procédure qui efface le contenu (autrement dit la valeur) de la zone de saisie d'identifiant *"res"*. Branchez-la sur la page pour qu'elle soit exécutée lors d'un clic sur le bouton "C". Comment pouvez-vous tester si votre procédure fonctionne dans la mesure où le champ de saisie est "readonly" ?
- une procédure qui évalue le contenu de la zone de saisie d'identifiant *"res"* et qui remplace le contenu de cette zone de saisie par la valeur évaluée. Branchez-la sur la page pour qu'elle soit exécutée lors d'un clic sur le bouton "=" et testez-la.
- une procédure qui, lorsque l'on clique sur un bouton, ajoute le texte qui apparaît sur le bouton (attribut *textContent*) dans la zone de saisie d'identifiant *"res"*, à la suite du texte qui y était déjà. Par exemple, si la zone de saisie contenait "bonjour " et que le bouton sur lequel est branché cette fonction porte le texte "monde", la zone de saisie contiendrait après clic sur le bouton le texte "bonjour monde". Branchez cette méthode sur la page pour qu'elle soit exécutée lors d'un clic sur n'importe quel bouton contenant un chiffre ou un opérateur ;

Maintenant si vous cliquez par exemple sur "1" puis "2" puis "+" puis "4", la chaîne "12 + 4" doit apparaître dans la zone de saisie, et si vous appuyez sur "=", elle doit être remplacée par le résultat "16".

c. Question subsidiaire (seulement pour ceux qui se sentent à l'aise). Chaque objet javascript représentant un bouton possède un attribut *onclick*. Si on veut faire exécuter à un bouton la fonction `mafonction()` il suffit de donner à son attribut *onclick* la valeur `mafonction`. Reprenez le fichier HTML initial (sans les attributs *onclick*) et rajoutez dans votre fichier typescript une fonction permettant de brancher les fonctions précédentes sur les boutons appropriés. Cette nouvelle fonction sera appelée lors du chargement de la page, autrement dit dans l'attribut *onload* de l'élément <**body**>.

Voyez-vous l'intérêt de procéder ainsi plutôt que de tout écrire dans le document HTML ?

d. Variation et complément : comme vous l'avez constaté, un élément HTML <**button**> peut avoir un contenu (ce qui n'est pas le cas d'un bouton créé par la balise <**input** />) ; lorsque le contenu de cet élément est simple (du texte, par exemple), on peut accéder au contenu comme vous l'avez précédemment fait en utilisant l'attribut *textContent* mais on peut également utiliser la hiérarchie du DOM. En effet, le "contenu" d'un tel bouton est, dans l'arbre du DOM, contenu par le premier enfant du bouton, accessible via l'attribut *firstChild* du bouton. On peut alors récupérer le contenu de ce premier enfant par l'attribut *nodeValue* ; par conséquent, le texte qui s'affiche sur le bouton "courant" est désigné par `this.firstChild.nodeValue`. Testez cette nouvelle manière d'accéder au contenu d'un bouton.

Exercice 3 : Réalisation d'une page dynamique (inspiré du CC 2015)

Pour cet exercice, plusieurs fichiers vous sont fournis sur UPdago : un squelette de page HTML, `exercice2.html`, que vous allez compléter en suivant les indications qui suivent, un fichier de style, `exercice2.css`, un fichier de code TypeScript, `tableau.ts` qui contient quelques fonctions utiles qui seront présentées un peu plus loin et le résultat de sa transpilation en JavaScript.

a. Partie HTML

Complétez le fichier HTML fourni pour que la page commence par un titre de niveau 1 ("Damier"). L'élément suivant dans la page est un tableau dont le corps est composé de 2 lignes, chacune contenant 3 cases. La première et la deuxième case de chaque ligne contiennent du texte ou un label au choix, suivi d'un champ de saisie de texte (input), comme représenté sur la figure 1 [2]. La dernière case de chaque ligne contient un bouton qui permettra ultérieurement de lancer des scripts. Enfin la page se termine par une ligne horizontale. Les champs de saisie de texte possèdent des valeurs par défaut que vous pouvez voir également sur la figure 1 [3]. Votre fichier doit bien évidemment passer l'épreuve du validateur.

Pour les deux questions suivantes, vous aurez besoin des fonctions définies dans `tableau.js` :

- `buildTable(l,c)` : crée un tableau de l lignes et c colonnes dans la page HTML ;
- `getNumberOfLines()` : renvoie le nombre de lignes du tableau ;
- `getNumberOfColumns()` : renvoie le nombre de colonnes du tableau ;
- `setElementValue(line,col,val)` : écrit le contenu du paramètre *val* dans la case située sur la ligne *line* et la colonne *col*. Les lignes et les colonnes sont numérotées à partir de 1 ;
- `getElementValue(line, col)` : renvoie la valeur contenue dans la case située sur la ligne *line* et la colonne *col* ;
- `getElement(line, col)` : renvoie l'objet correspondant à la case située sur la ligne *line* et la colonne *col* ;
- `fillTable()` : remplit chaque case du tableau avec la chaîne (i, j) où i et j sont respectivement les numéros de ligne et de colonne de la case.

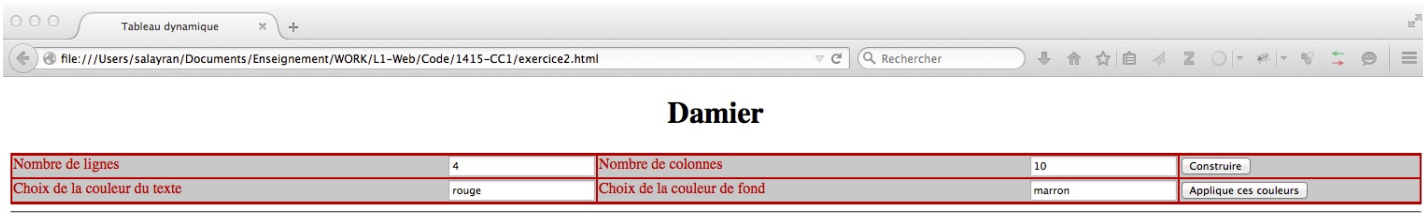

FIGURE 1 – Capture d'écran de la page à reproduire.

b. Javascript, partie I

- Ajoutez ce qu'il faut dans votre fichier HTML pour pouvoir y utiliser les fonctions définies dans `tableau.js`.
- Dans cette partie, les instructions javascript seront écrites et testées dans la *console web* ouverte sur la page `exercice2.html`. Vous sauvegarderez votre travail dans le fichier `log_consoleweb.js` (les instructions et les résultats de leur évaluation).

  En utilisant les fonctions fournies, écrivez une suite d'instructions pour :

2. Le style, notamment les couleurs et la position des éléments dans les cases du tableau, est géré par le fichier CSS fourni, ce n'est pas à vous de le faire. Et si l'apparence de votre page n'est pas exactement celle apparaissant sur l'image, ce n'est pas nécessairement un problème.
3. La figure 1 est aussi accessible en version couleur sur UPdago.

- créer dans la page un tableau de 4 lignes et 5 colonnes avec sur chaque case du tableau la chaîne (i, j) où i et j sont respectivement les numéros de ligne et colonne de la case ;
- affecter à une variable v le contenu de la case située sur la $3^{ème}$ ligne et $2^{ème}$ colonne et vérifier que la variable contient bien la valeur attendue en affichant son contenu dans la console ;
- écrire la chaîne de caractères "Hi !" dans la première case du tableau (en haut à gauche) ;
- donner à la même case la couleur de fond (propriété *backgroundColor* de l'attribut *style*) rouge ("red").

c. Javascript, partie II

Dans cette partie, vous allez écrire des fonctions dans un fichier `exercice2.ts`[4] que vous transpilerez dans le fichier javascript `exercice2.js`.

- Modifiez votre document HTML pour y inclure le fichier `exercice2.js` ;
- Écrivez une procédure *initTable* : elle vérifie d'abord que les valeurs présentes dans les deux premiers champs de texte de votre page html sont bien strictement positives. Si c'est le cas, elle construit un tableau dans la page html avec le nombre de lignes et de colonnes demandées et le remplit de sorte que chaque case porte la chaîne de caractères "(i,j)", avec i et j ses numéros de ligne et de colonne. Si l'une au moins des valeurs dans les champs de saisie est négative ou nulle, un message d'erreur est affiché via une alerte. Branchez cette fonction sur le bouton *Construire*.
- Écrivez une fonction *translateColor* qui prend en paramètre une chaîne de caractères représentant le nom d'une couleur en français et renvoie la chaîne correspondant à la couleur en anglais. Elle est capable de traiter les couleurs suivantes : "rouge" / "red", "bleu" / "blue", "marron" / "brown", "vert" / "green", "jaune" / "yellow", "blanc" / "white". Pour toute autre chaîne de caractères, la fonction renvoie "black".
- Écrivez enfin une procédure *colorTable* qui récupère les chaînes de caractères présentes dans les deux autres champs de saisie de votre page HTML et colorie le texte écrit dans toutes les cases du tableau avec la couleur indiquée dans le premier (propriété *color* de l'attribut *style* d'une case) et le fond de toutes les cases avec la couleur indiquée dans le second. Le coloriage correspondra à ce qui est attendu si les noms de couleurs sont des noms français parmi ceux précédemment cités. Ce sera noir sinon. Branchez cette fonction sur le bouton *Applique ces couleurs*

4. On demande une fonction principale par question mais rien ne vous empêche d'écrire et d'utiliser des fonctions auxiliaires...bien au contraire !