

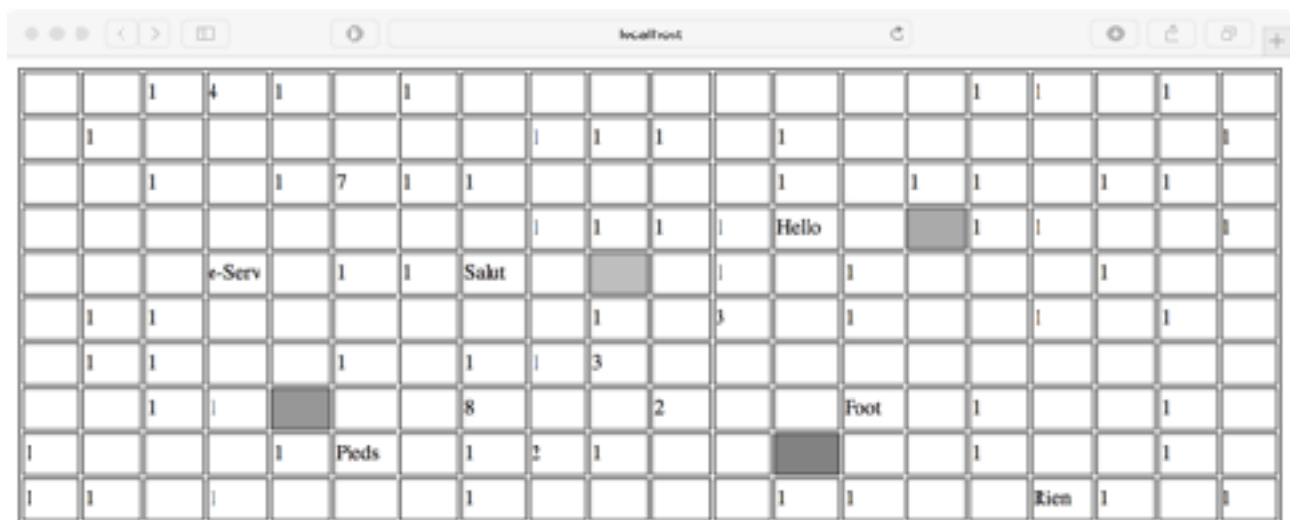
Contrôle de Web Avancé / Javascript

e-services FI - Lundi 24 Octobre 2016

Nom :

Prénom :

L'application qui servira à évaluer vos connaissances en Javascript est une simple page Web qui affiche un tableau comme suit :



		1	4	1		1								1	1		1	
	1							1	1	1		1						1
		1		1	7	1	1					1		1	1		1	1
								1	1	1		Hello		1			1	
			e-Serv		1	1	Sant					1		1			1	
	1	1							1		3		1			1		1
	1	1			1		1		1	3								
		1	1				8			2			Foot		1			1
1				1	Peds		1	2	1					1			1	
1	1						1					1	1				Lien	1

Mais si c'est une simple page Web ce n'est pas un simple tableau. La pseudo-classe Javascript (JS) qui permet de faire cela peut créer plusieurs tableaux : la largeur et la hauteur sont paramétrables, tout comme le style...

Une autre particularité du tableau est que les cases sont de 3 sortes :

- 1) une case avec un 1 dedans, qui, lorsque l'on clique dessus, s'incrémente de 1.
- 2) une case avec rien dedans, qui, lorsque l'on clique dessus, ouvre un prompt qui demande le nouveau contenu.
- 3) une case avec rien dedans, qui, lorsque l'on clique dessus, s'assombrit jusqu'à devenir noir.

Ce contrôle n'a pas pour objectif d'évaluer vos compétences en matière d'algorithmie mais de JS. Donc pour la plupart des questions, vous aurez une version en langage naturel d'un programme que vous devrez traduire en JS.

Les classes, dans ce contrôle, sont à créer à partir de fonctions et de prototypes (pas d'EcmaScript 6 donc, c'est-à-dire pas de mot-clé `class`).

La fonction `nombreAuHasard` (pour s'échauffer)

Elle prend un seul paramètre nommé `max`.

Contenu :

Si `max` est égale à 0, `nombreAuHasard` renvoie 0.

Sinon la fonction renvoie l'entier arrondi (fonction `Math.floor`) du décimal tiré au hasard (fonction `Math.random` >> entre 0 et 1) multiplié par `max`. Avec l'envoi, on y rajoute 1.

La classe `Tableau`

Le constructeur

Son constructeur prend un seul paramètre nommé `parametres`. Ce doit être un objet qui peut contenir les propriétés suivantes :

- `conteneurHTML`, qui sera l'élément HTML qui contiendra la table HTML correspondant au tableau
- `nbColonnes`, qui sera le nombre de colonnes du tableau
- `nbLignes`, qui sera le nombre de lignes du tableau
- `styleTable`, qui sera un objet dont chaque propriété sera appliquée telle quelle au style de l'élément `table` de notre tableau. Par exemple, si l'objet contient la propriété `background` avec la valeur `"#000000"`, cette valeur sera appliquée à la propriété `background` de la propriété `style` de l'élément `table`.
- `styleCellule`, la même chose que la propriété précédente mais ici sur chaque cellule du tableau (balises `td`).

Le constructeur ajoute/transfère chacune de ses propriétés dans le nouvel objet créé. Puis il appelle la méthode `construireLeTableau` (méthode définie, plus loin, dans la pseudo-classe)

La méthode construireLeTableau

La méthode commence par ajouter à l'objet une propriété `table` contenant un nouvel élément HTML de type `table`.

Ensuite elle ajoute une propriété `lignes` qui contient un tableau JS vide;

Une première boucle est faite avec comme itérateur `indexLigne` qui va de 0 à la valeur contenu dans la propriété `nbLignes` de l'objet.

Dans cette boucle, on ajoute dans la précédente propriété `lignes` un objet qui contient 2 propriétés : 1) une propriété `tr` qui vaut un nouvel élément HTML de type `tr`. 2) une propriété `cellules` qui vaut un tableau vide.

Ensuite, on ajoute à l'élément HTML référencé par la précédente propriété `table`, l'élément HTML référencé par la précédente propriété `tr`.

Sans sortir de la boucle, on crée une sous-boucle avec comme itérateur `indexColonne` qui va de 0 à la valeur contenu dans la propriété `nbColonnes` de l'objet.

Dans cette sous-boucle, on définit une variable `choixCellule` qui vaut un nombre au hasard entre 1 et 3.

On définit aussi une variable `cellule` qui n'a aucune valeur pour l'instant.

Si `choixCellule` vaut 1, `cellule` sera égale à une instance de `CelluleAvecEntier`.

Si `choixCellule` vaut 2, `cellule` sera égale à une instance de `CelluleAvecPrompt`.

Si `choixCellule` vaut 3, `cellule` sera égale à une instance de `CelluleColoree`.

Quelque soit la classe utilisée, ce sera toujours le même paramètre lors de l'instanciation: un objet avec 2 propriétés : 1) `style` qui vaut la propriété `styleCellule` de l'objet; 2) `baliseMere` qui vaut l'élément HTML référencé par la précédente propriété `tr`.

Chaque cellule créée est ajoutée au précédent tableau référencé par la précédente propriété `cellules` (voir propriété `lignes`).

À la sortie de la boucle principale, on ajoute l'élément HTML `table` (référéncé par la propriété `table`) à l'élément référencé par la propriété `conteneurHTML`.

Enfin, on applique chaque propriété de `styleTableau` à la propriété `style` de l'élément `table`.

La classe `Cellule`

La pseudo-classe `Cellule` est la classe qui servira de base aux 3 types de cases/cellules évoquées plus haut.

Constructeur

Le constructeur de `Cellule` prend, comme on l'a vu plus haut, un seul paramètre, un objet, qui a comme propriétés le `style` et la `baliseMere`. Ces deux propriétés sont transférées dans le nouvel objet créé par le constructeur.

Le constructeur définit ensuite un getter / setter pour la *pseudo-propriété* `texte` qui va aller chercher/modifier la propriété `data` d'un élément texte HTML référencé par la propriété `baliseTexte` de l'objet (voir plus bas).

Pour finir, le constructeur invoque la méthode `construireLaCellule`.

La méthode `construireLaCellule`

La méthode `construireLaCellule`, qui ne prend aucun paramètre, crée d'abord une balise HTML `td` qu'elle met dans la propriété `balise`. Elle ajoute cette balise à la balise référencée par la propriété `baliseMere` de l'objet. Ensuite une balise textuelle est créée, elle sera référencée par une propriété `baliseTexte` de l'objet. Cette balise est ajoutée à la balise référencée par la propriété `balise`.

Ensuite, comme pour le tableau, toutes les propriétés de la propriété `balise` de l'objet sont appliquées au `style` de la balise `td`.

Enfin, on ajoute l'objet aux auditeurs de `click` de la propriété/élément HTML `balise`.

D'ailleurs, il faut définir dans l'objet la propriété/méthode nécessaire pour que l'objet puisse réceptionner les clicks. Cette méthode, dans `Cellule`, ne fait rien.

La classe `CelluleAvecEntier`

Le comportement de cette classe a déjà été définie plus haut.

Elle hérite de `Cellule`. *Pour cette héritage, que va-t-il se passer pour le constructeur de `Cellule` et l'absence de paramètre (question délicate et non évoquée en cours :-)). Que faudrait-il ajouter et où ?*

Le constructeur de `CelluleAvecEntier` invoque celui de sa classe mère... avec les mêmes paramètres. Et ne fait rien d'autre.

La méthode `construireLaCellule` invoque la même méthode mais sur sa classe mère et met le texte '1' dans la cellule (en utilisant le setter).

La méthode de réponse au click est définie de manière à ajouter 1 à la valeur entière (après passage) du texte de la cellule.

La classe `CelluleAvecPrompt`

Note : la fonction `prompt` de JS prend comme 1er paramètre le titre du prompt/fenêtre de dialogue et comme 2ème paramètre la valeur de base qui sera dans la zone de saisie.

Le comportement de cette classe a déjà été définie plus haut.

Elle hérite de `Cellule`.

Le constructeur de `CelluleAvecPrompt` invoque celui de sa classe mère... avec les mêmes paramètres. Et ne fait rien d'autre.

La construction de la cellule est exactement la même que celle de sa classe mère... faut-il quand même redéfinir cette méthode ?

La méthode de réponse au click est définie de manière à demander la nouvelle valeur de la cellule avec, dans la zone de saisie, l'ancienne valeur.

La classe `CelluleColoree`

L'énoncé a un peu menti au début car il y a dans cette question un peu d'algorithmie. Pour changer la couleur de fond d'un élément, il faut passer par la propriété `background` de la propriété `style` (comme vu plus haut). Les valeurs possibles sont soit des noms de couleurs prédéfinies (ce qui ne nous arrange pas ici), soit le RGB via `#0000FF` (pour du bleu) en hexadécimal, soient enfin le RGB via `rgb(0, 0, 255)`. Cette dernière n'est pas une fonction javascript mais une fonction HTML. Mais c'est quand même cette fonction qui est la plus intéressante pour coder notre `CelluleColoree`.

Comment va être implémentée cette classe ?

Note : le constructeur, la méthode `construireLaCellule` et celle pour écouter le click doivent être redéfinies (comme pour `CelluleAvecEntier`). Il faudrait peut-être même une méthode `fixerLaCouleur` qui change la couleur de fond de l'élément HTML.