Leçon 3-1 – Compléments sur JavaScript

s1 ----------

Dans cette leçon, nous allons introduire quelques notions JavaScript supplémentaires. Certaines notions vous seront très utiles pour développer le projet qui constitue la seconde partie du travail de cette semaine ou le projet final de ce MOOC.

s2 ----------

Nous allons introduire successivement quelques notions sur les tableaux imbriqués et les objets JavaScript (notions qui seront utiles pour le projet final de ce MOOC) ainsi que sur les formulaires (qui vous seront très utiles pour le projet de cette semaine). Nous finirons par les opérateurs usuels.

s3 ----------

Un mot sur les tableaux imbriqués.

Lorsque les éléments d'un tableau sont eux-mêmes des tableaux, on parle de tableaux imbriqués ou multidimensionnels. Les tableaux imbriqués sont généralement utilisés pour définir une collection d'éléments. En voici un exemple (que nous reprendrons dans un futur exercice).

Dans cet exemple, listeBalles est un tableau contenant quatre éléments ; chacun de ces éléments est lui-même un tableau contenant 2 éléments qui sont les caractéristiques de chaque balle, une chaîne de caractères (représentant ici le code couleur de la balle) et un entier (représentant ici le rayon de la balle).

Une autre façon de définir le tableau précédent est la suivante :

Comment accéder aux caractéristiques de chaque élément ? Dans les expressions ci-dessous, la paire de crochets la plus à gauche contient l'indice de chaque tableau à l'intérieur du tableau listeBalles ; la seconde paire de crochets contient l'indice d'un enregistrement à l'intérieur de ce dernier tableau.

Dans l'exemple précédent, tous les éléments du tableau listeBalles sont des tableaux qui contiennent le même nombre d'éléments, mais ce n'est pas une obligation. Voici un exemple d'un tableau imbriqué où les éléments n'ont pas la même dimension.

Dans cet exemple, le premier élément de `tab1` est le tableau `[100, 0, 10]` contenant 3 éléments ; le deuxième élément de `tab1` est la chaîne de caractères `"Marcel Proust"` ; le dernier élément de `tab1` est le tableau `[24, true]` qui contient 2 éléments.

Je vous propose de reproduire ces instructions JavaScript au sein d'une balise script et dans un fichier HTML et d'afficher (par une simple alerte) les variables couleurPremiereBalle et rayonTroisiemeBalle.

s4 ----------

Les objets sont en quelque sorte une autre façon de représenter une collection d'éléments possédant

les mêmes caractériques. Lors du projet final, certains d'entre vous préféreront peut-être utiliser des objets que des tableaux imbriqués.

En JavaScript, un objet est une entité indépendante, portant des propriétés et un type. Si, par exemple, on considère une balle : une balle possède des propriétés comme sa couleur, son rayon, etc. Une propriété peut être vue comme une variable associée à l'objet.
En JavaScript, on déclare un objet de la façon suivante.

On peut également accéder aux propriétés des objets JavaScript en utilisant des crochets.

Une façon élégante de définir un objet est la suivante.

Et bien entendu, rien ne nous empêche de définir une liste de balles comme un tableau d'objets.

Quelques remarques

- le nom des propriétés, comme les noms de variables, peut être n'importe quelle chaîne de caractères conforme ; si ce nom n'est pas valide (par exemple si ce nom contient un espace, un tiret ou commence par un chiffre) on ne pourra accéder à la valeur de la propriété que par la notation utilisant les crochets.
- attention, le nom des propriétés est sensible à la casse
- la valeur d'une propriété peut être une chaîne de caractères, un nombre, un tableau, voire un objet JavaScript

L'utilisation d'objets en JavaScript est très riche. Dans ce MOOC, j'ai choisi de ne pas aller au-delà de ces simples définitions.

Je vous propose de reproduire ces instructions JavaScript au sein d'une balise script et dans un fichier HTML.

s5 ----------

Quelques notions de base sur les formulaires. Vous avez certainement en tête ce qu'est un formulaire. On y trouve généralement des zones de texte à remplir, des listes déroulantes, des cases à cocher et un ou des boutons qui permettent de soumettre le contenu de ce formulaire.

Nous n'allons pas ici nous préoccuper du traitement des données du formulaire, notamment leur envoi vers un serveur et leur stockage dans une base de données (ceci fera l'objet d'un autre MOOC). Nous allons simplement nous intéresser à la façon de les récupérer et de les manipuler à travers des instructions JavaScript.

s6 ----------

Du point de vue HTML, voici comment se présente le fichier HTML du formulaire précédent. Pour des raisons de simplification du propos, j'ai enlevé certains champs dont le type était répété (le champ nom complet, les listes déroulantes Genre et Année de naissance, une des cases à cocher...) ainsi que la plupart des classes CSS. Je vous propose de le télécharger et de l'ouvrir avec un éditeur de texte.

Comment est structuré ce formulaire ?

La première balise importante est la balise form qui déclare la zone de formulaire. Cette balise form a deux attributs obligatoires : METHOD indique sous quelle forme les données seront envoyées. ACTION indique l'adresse d'envoi , par exemple une URL. Dans cet exemple, form possède en plus un attribut id.

À l'intérieur des balises ouvrante et fermante form, on trouve plusieurs autres balises.

On trouve deux blocs fieldset qui permettent de définir des parties de formulaires. L'affichage par défaut de ces blocs est d'être encadrés par une ligne. Une balise legend permet d'en définir le titre.

Dans le premier bloc, fieldset, on trouve trois fois le couple label input. Selon son type, la balise input représente un champ de saisie ou un bouton. Dans notre exemple, il y a des elements input de plusieurs types dont la signification est claire :

- email qui signifie que le formulaire s'attend à trouver une adresse mail
- password qui signifie que le formulaire s'attend à trouver un mot de passe (vous pourrez vérifier que les caractères seront remplacés par des points ou des étoiles à la saisie)
- text qui permet de saisir une ligne de texte
- checkbox, qui est une case à cocher
- hidden qui permet de stocker des vaariables non visibles pour l'utilisateur
- submit, enfin, qui définit un bouton

Horsmis l'attribut type, la balise input présente deux autres attributs obligatoires : les attributs name et value. Name permet au programme qui traitera le formulaire d'identifier un élément input ; value de connaître la valeur entrée par l'utilisateur ou la valeur par défaut.
Ici chaque élément input possède également un attribut id qui permet de le relier à une étiquette définie par l'élément label. Certains éléments input possèdent l'attribut aria-required qui permet de préciser que le formulaire ne pourra pas être envoyé tant que l'information n'a pas été fournie. L'attribut required, a été introduit par HTML5 pour les mêmes raisons, mais n'est pas encore pris en charge par tous les navigateurs.

Dans le second bloc, on trouve deux autres éléments de saisie :

- l'élément select
- l'élément textarea

La balise select permet de créer une liste déroulante d'éléments. Ces éléments sont définis par des balises OPTION à l'intérieur de l'élément select.
L'attribut name de la balise select permet au programme qui traitera le formulaire de l'identifier. L'attribut value de l'élément option permettra également de le repérer.

Enfin, la balise textarea permet de définir une zone de saisie sur plusieurs lignes. À nouveau, on retrouve les attributs name et value qui représente la valeur qui sera envoyée par défaut au script si le champ de saisie n'est pas modifié par une frappe de l'utilisateur.

Un dernier mot sur la balise button, qui est une façon alternative par rapport à input type=submit de définir un bouton. Ces deux boutons se comportent de la même façon. La différence réside dans le fait que l'on peut mettre du code HTML (des images, par exemple) à l'intérieur de l'élément button ; il faut aussi signaler de possibles soucis fonctionnels avec Internet Explorer.

Voilà pour quelques rudiments concernant la structure d'un formulaire. Avant de passer à la phase de

traitement JavaScript, je vous invite à faire quelques tests à partir du fichier HTML fourni.

s7 ----------

En général, une fois que le formulaire est rempli, on clique sur le bouton (l'élement input ou button qui possède l'attribut submit) et les données sont transmises à l'URL définie dans la balise form. Ce traitement on l'a dit ne fait pas l'objet de ce module.

Comment, récupérer les données grâce à des instructions JavaScript ? Ceci se fait simplement. Nous allons ajouter deux choses à la page HTML :

- des instructions JavaScript définissant une fonction afficherValeurs()
- un simple lien HTML qui déclenchera cette fonction.

Que fait la fonction afficherValeurs() ? Elle affiche la valeur saisie dans le champ email.

L'instruction se lit de la façon suivante : dans le document, dans le tableau (créé automatiquement) qui répertorie touts les formulaires présents (ici un seul formulaire est présent), cibler celui qui a pour nom registerForm ; parmi tous les éléments constituant ce formulaire, cibler celui qui pour nom email, et récupérer la valeur de son attribut value.
Maintenant que vous avez vu ce qu'était un objet JavaScript, cette instruction vous permet de comprendre plusieurs choses :

- l'élément JavaScript document est un objet
- cet objet possède plusieurs attributs dont l'attribut forms qui a pour valeur un tableau
- ce tableau a autant d'éléments ou d'attributs si vous voulez que de formulaires
- l'élément forms["registerForm"] est lui même un tableau contenant plusieurs éléments
- chacun de ces éléments est un objet
- celui qui nous intéresse, l'élément possédant le nom email est un objet possédant l'attribut value

Dans cette logique, donc l'instruction précédente pouvait être écrire de la façon suivante.

s8 ----------

Cherchons maintenant à afficher la valeur sélectionnée par l'utilisateur dans la liste déroulante.

La nouvelle instruction se lit de la façon suivante : dans le document, dans le tableau (créé automatiquement) qui répertorie touts les formulaires présents, cibler celui qui a pour nom registerForm ; parmi tous les éléments constituant ce formulaire, cibler celui qui a pour nom education-level, et récupérer la valeur de son attribut selectedIndex.

La dernière instruction se lit de la façon suivante : dans le document, dans le tableau (créé automatiquement) qui répertorie touts les formulaires présents, cibler celui qui a pour nom registerForm ; parmi tous les éléments constituant ce formulaire, cibler celui qui a pour nom education-level, parmi toutes les options constituant cette liste, récupérer la valeur de l'option correspondant à l'index selectedIndex.

S9 ----------

Pour finir, des précisions sur quelques opérateurs JavaScript. Le tableau ci-affiché se passe de commentaires, mais je vous engage à tester les opérateurs que vous ne connaissez pas.

S10----------

Voici donc pour les premières notions de JavaScript. Je vous propose à présent de suivre la prochaine leçon qui vous guidera pas à pas dans le développement du premier projet de jeu. A tout de suite.