Web Avancé - Séance 2
Javascript

Sommaire

L'objectif du TP est de s'entraîner à programmer en Objet avec Javascript.

Nous allons pour cela développer petit à petit un ensemble de classes qui nous permettront de construire facilement des formulaires (avec des champs de saisies précédés de labels).

1. La classe Screen. Premier contact

Tout au long de ce TP, nous allons travailler avec le fichier index.html suivant

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>librairie Window</title>
<script type="text/javascript" src="source.js"></script>
</head>
<body style="font-family:arial;color:black;font-size:12px;">
</body>
</html>
```

Au départ le fichier source.js vaut :

```
window.onload = function () {
  // le code ici sera exécuté quand la page sera complétement chargée

}
```

Exo 1

Le code correspondant au onload est le suivant :

```
screen=new application.Screen(); // L1
screen.addTitle("First Screen"); // L2
```

où L1 supprime tout le contenu de la balise body

et L2 ajoute simplement (pour l'instant) une balise div avec le texte correspondant au paramètre.

Votre mission : Créer/implémenter la classe application.Screen et sa méthode addTitle.

Quelques brides de code qui peuvent aider :

Pour L1 :

```
var body=document.getElementsByTagName("body").item(0);
for (var i=0;body.childNodes.length;i++)
      body.removeChild(body.childNodes.item(0));
```

Pour L2 :

```
      var title=document.createElement("div");
```

```
var titleText=document.createTextNode("Mon texte");
title.appendChild(titleText);
body.appendChild(title);
```

Exo 2

Ajouter à application.Screen une méthode setTitleStyle() qui fixe le style suivant à la balise DIV du titre uniquement en passant par Javascript :

"width: 100%; margin: 0px; padding-top: 10px; padding-bottom: 10px; background-color:#2B5278; top: 0px; left: 0px; text-align:center; font-size:18px; font-weight: bold; color = white;"

Dans cette nouvelle méthode, il faudra pouvoir référencer facilement la balise DIV… peut-être faut-il la conserver dès addTitle.

Quelques brides de code qui peuvent aider :

Pour width, l'équivalent javascript est :

```
balise.style.width="100%";
```

Exo 3

Ajouter la méthode addTextField qui rajoute un paragraphe comprenant :

- un label (une balise span)
- une zone de saisie de texte (une balise input de type text)

Les paramètres de cette méthode sont :

- name : le nom du champ (nom qui servira ensuite)
- parameters : qui est objet qui PEUT contenir une propriété label et qui PEUT contenir une propriété hint
    - s'il y a un label, il prendra place dans la balise span
    - s'il y a un hint, on le verra apparaître dans la zone de texte

Ex d'utilisation :

```
screen.addTextField ("nom", { label : "Nom :", hint : "Votre nom" });
screen.addTextField ("prenom", { label : "Prenom :", hint : "Votre prénom" });
```

Quelques brides de code qui peuvent aider :

Pour le hint, il faut utiliser l'attribut placeholder d'input

Exo 4

On commence à voir des document.createElement et createTextNode partout !

Créer un objet domHelp avec 2 fonctions

- addElement : 2 paramètres ou +. Le premier = la balise mère. Le deuxième = le nom de la balise à créer et à ajouter à la balise mère. Les autres paramètres = une succession de nomAttribut, ValeurAttribute si des attributs sont à fixer
- addText : 2 paramètres > la balise mère et le texte à ajouter

Ces 2 fonctions renvoient la balise créée.

Remplacer dans vos anciens document.createElement et createTextNode par les fonctions correspondantes et supprimer les appendChild

2. La classe Field au secours de Screen

Ce qu'on souhaite maintenant est la chose suivante :

```
screen.addTextField ("nom", { label : "Nom :", hint : "Votre nom" });
screen.fields.nom.label=”Name :”;
screen.fields.nom.hint=”Your name”;
```

Ceci implique 2 choses :

- chaque screen conserve les champs créés dans une propriété fields qui est un objet dont les propriétés sont les noms des champs (fixé précédemment)

- fields contient des objets avec un comportement particulier : affecter une valeur à label va modifier le noeud textuel de la balise span. affecter une valeur à hint va modifier la valeur de l'attribut placeholder de la balise input. On pourra aussi connaitre la valeur du label ou du hint grace aux 2 propriétés de chaque champ.

Exo 5

Créer la classe application.Field et utiliser à bon escient pour faire fonctionner la propriété fields de Screen.

Exo 6

Maintenant on veut la chose suivante :

screen.fields.nom.name=”nom2”;

screen.fields.nom // erreur

screen.fields.nom2 //ok

3. Palette de composants (héritage)

On veut utiliser l'héritage afin de créer facilement un ensemble de classes (dérivées de Field) qui permettrait ceci

```
screen.addTextField ("nom", { label : "Nom :", hint : "Votre nom" });
screen.addTextField ("prenom", { label : "Prénom :", hint : "Votre prénom" });
screen.addPasswordField ("mdp", { label : "Mot de passe :", hint : "Votre mot de passe"});
screen.addRadioField ("situation", { label : "Situation :", choices : ["marié(e)","célibataire"]});
screen.addCheckBoxField ("diplomes", { label : "Vos diplômes :", choices : ["Bac","Licence","Master","Doctorat"]});
screen.addButton ("valider", "Valider");
```

Exo 7

Créer les classes application.TextField et application.PasswordField qui héritent de application.Field. Cette classe a bien entendu changé (un peu). La création de la zone de texte n'y est plus faite car elle apparaît dans les classes dérivées.

Exo 8

Créer la classe application.RadioField qui héritent de application.Field.

Exo 9

Créer la classe application.CheckBoxField qui héritent de application.Field en un minimum de ligne (car il n'y a qu'un mot qui change … on aurait pu faire de même pour PasswordField). Il faudra modifier RadioField.

Et donc modifier TextField et PasswordField pour avoir le même principe.

Exo 10

Créer la classe application.Button qui n'hérite de rien :-)