TD Pweb Client riche. 19 janvier 2016
IUT PARIS – FI2.
J-M Ilié, C. Kurtz

TP1 javascript

Le but esr de lancer du code réactif écrit en javascript en réponse à des événements déclarés pour des balises, par exemple pour répondre à un clic sur le contenu d'une balise. En remarquant la nécessité de séparer le code javascript de la spécification HTML, on tentera de réduire le code javascript invoqué sur un événement, à un simple appel de fonction, de telle sorte que toute la complexité du code réactif soit encapsulée dans une fonction.
Les fonctions réactives, les variables et les fonctions annexes sont déclarées dans un ou plusieurs fichiers javascript..

Etape 1 : Rattacher un fichier de script à un fichier HTML

```
--- index.html
--- scriptJS
      ---- script.js
```

Dans votre répertoire de travail, créer un sous répertoire pour le TP
Considérer un fichier index.html et son contenu qui affiche un bloc `<div>`, contenant 2 paragraphes <p>. Créer un répertoire `scriptJS` dans lequel on placera les fichiers de scripts.
Rattacher un fichier de script javascript `script.js` à index.html avec les balises <script></script> à placer dans l'entête du fichier HTML. Indiquer comme valeur de l'attribut `src` de cette balise, le nom relatif du fichier `script.js`, Noter que quand on spécifie le fichier HTML, le répertoire du fichier html est vu comme le répertoire courant.

Etape 2 : Vérification du rattachement du fichier de script

Ecrire l'instruction suivante, un appel à la fonction `alert()` qui affiche un petit message d'alerte, à déclarer en paramètre de la fonction. Lancer `index.html` sur le navigateur pour voir apparaître la fenêtre, avant même l'affichage du corps de la page HTML. En déduire la façon dont le navigateur opère, concernant le fichier de script.

Etape 3 : Gestion HTML des événements

Réagir sur les clics de souris agissant sur le bloc pour lancer l'instruction `alert("cliqué").`

```
<div onclick="alert('cliqué');">
```

Le ";" est optionnel mais témoigne du fait que sur un clic, on lance tout un programme (écrit en javascript).
Noter l'alternance possible entre `'` et `"` pour éviter les confusions, de plus `\'` permet d'afficher le caractère `'`.

Afficher alternativement le message : c'est cliqué.

Etape 4 : Appel de fonctions réactives javascript,

Invoquer une fonction, par exemple `unClick()` destinée à réagir au clic précédent, et encapsulant le code (pour l'instant simple) de l'alerte, puis invoquer cette fonction à partir de

l'attribut onclick de la balise `<di>`. Déclarer et implémenter cette fonction dans le fichier javascript, :

```
function  unClick() (
}
```

Etape 5 : Accès et affichage d'objets mémoire HTML.
Utiliser la référence sur un objet.

- `this` sur l'objet courant au moment d'une réactionou
- une référence rétournée par une des fonctions prédéfinies définies dans l'objet `document`
- `window` référençant le premier objet mémoire du navigateur.

Etape 6 : Utiliser l'environnement de développement de Firefox.
Pour cela, avec le clic droit dans la fenêtre du navigateur, sélectionner `examiner élément`.

- l'onglet `console` affiche les messages du navigateur, log ou message d'erreurs.
- l'onglet `debogueur` permet d'afficher le fichier de script, de mettre des points d'arrêt dans les fonctions appelées, sur un point d'arrêt de visualiser les propriétés des objets.
- Sur un point d'arrêt, il est possible de voir le contenu des objets mémoire HTML présents dans le contexte d'exécution, soit en survolant le code, soit dans le cadre de droite qui apparaît dans la zone de debogage.

Etape 7 : Gestion hiérarchique sur un événement

Constater la diffusion des événements sur une hiérarchie de balises, ainsi que l'ordre de prise en compte de ces événements au niveau des balises.
Pour cela, réagir sur les clics de souris agissant sur le bloc `<div>` et sur le premier paragraphe `<p>` pour lancer une instruction `alert()` spécifique à chaque cas. quand on clique sur le contenu de la div et/ou le contenu du premier paragraphe.