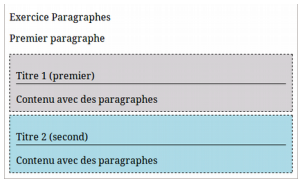
TP Frontend
EBM 2017/2018 - S1

Objectif : Découvrir les bonnes pratiques de mise en forme à l'aide de (X)HTML et CSS. Apprendre à utiliser Firefox et son plugin Firebug. Interactions avec les éléments du DOM à l'aide de javascript.

Plan de la séance :

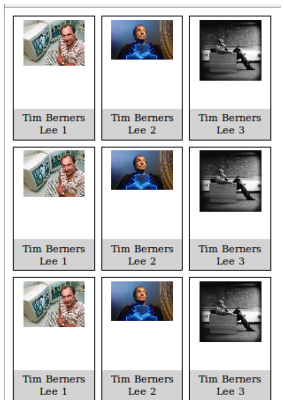
1. Discussions autour des IDE pour le WEB, chaque élève doit avoir la maîtrise de son IDE préféré
2. Installation & Présentation de l'outil Firebug :
    - Onglet HTML pour visualiser la stucture d'un document
    - Modification dynamique du CSS depuis Firebug
    - Onglet Firebug/Console Web pour générer des traces d'éxécution en javascript
3. Rappel des bons sites de documentation : W3schools, Alsacréations, A list apart
4. Chaque élève doit avoir parcouru les refcards HTML & CSS (leur impression est conseillée)
5. Pour chaque exercice du TP, un fichier HTML de base est fourni, il faut le compléter en suivant les indications de l'énoncé.
6. Pour chaque exercice du TP, une fois terminé, vous validerez votre travail à l'aide du validateur du W3C et appellerez l'enseignant pour débriefer. Vous sauvegarderez vos exercices pour pouvoir les retrouver plus tard.

Exercice 1 : Paragraphes



Styliser des paragraphes en leur associant identifiants et classes. Changer leur style par des sélecteurs. Faire apparaître leurs boites. Supprimer les retours à la ligne de leur contenu. Déplacer le code CSS dans un fichier de style séparé.

Exercice 2 : Galerie d'images



Créer une gallerie d'images à partir d'un ensemble de balises images, sans rien changer à la structure du fichier XHTML fourni. On produira des cadres de 200x250 pixels.
2ème version : les cadres d'images « s'allument » lorsque la souris les survole, le curseur de la souris change de forme comme s'il s'agissait d'un lien.

Exercice 3 : display_ACorriger

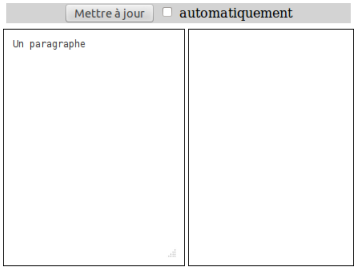
- Corriger le code pour que les deux boutons fonctionnent
    - Mettre en oeuvre 3 solutions
- Afficher et cacher tour à tour les boutons pour qu'il n'y en ait toujours qu'un seul visible à la fois.
- Supprimer un bouton et modifier le code pour que le même bouton puisse servir à cacher/afficher. Utiliser le mot-clé « this ».

Exercice 4 : Librairie utils.js

1. Réaliser une fonction trace
    - Utilise console.log si le navigateur n'est pas Internet Explorer
    - Cf. http://www.quirksmode.org/js/support.html
2. Réaliser une fonction debug(s)
    - Cette fonction ne fait plus rien après un nombre d'appels donné
    - Elle affiche la valeur du compteur si on ne lui fournit pas d'argument
3. Réaliser les fonctions show, hide, html, val
    - Leurs arguments peuvent être des identifiants ou des références : utiliser typeof
4. Transformer votre code en une librairie de fonctions

Exercice 5 : TryIt Editor

Nous souhaitons implémenter le formulaire proposé sur le site de W3Schools permettant de tester la syntaxe des langages HTML et CSS.



1. Un champ textarea permet d'éditer du code HTML, Un div placé à côté montre le rendu correspondant lorsque l'utilisateur clique sur un bouton. On pensera à utiliser des variables globales pour enregistrer des références vers les cadres d'édition et d'affichage.
2. Une case à cocher est placée à côté du bouton. Elle est associée au label 'automatiquement'.
3. Si la case est cochée, nous souhaitons mettre en place un mécanisme permettant une mise à jour à chaque modification du champ textarea. Nous utiliserons l'événement onkeyup. Pourquoi l'événement onchange n'est pas conseillé pour détecter les changements ? Comment connaître l'état de la case à cocher ?
4. 2ème version : Afficher des infos-bulles lorsque l'utilisateur déplace sa souris au dessus du bouton ou de la case à cocher
5. 3ème version : Nous souhaitons que la mise à jour automatique se déroule à intervalles réguliers, en utilisant la méthode window.setTimeout. Quels sont les arguments de cette méthode ? Quelle est leur nature ?
    - On informera l'utilisateur que quelquechose se passe en affichant une image de chargement (que l'on peut récupérer sur le site ajaxload.com), en s'arrangeant pour qu'elle soit visible suffisamment longtemps.
    - On désactivera la mise à jour automatique en utilisant la touche « ESC ».

Exercice 6 : Cas d'étude signin : Formulaire de création de compte

Nous souhaitons construire un formulaire de création de compte en facilitant la tâche de l'utilisateur. Nous lui suggérerons un mot de passe et lui indiquerons si ses identifiants correspondent bien aux contraintes qui lui sont imposées : choisir au moins une lettre minuscule, une lettre majuscule et un chiffre pour son mot de passe, entre 6 et 8 lettres pour son login. Tous les champs texte devront être remplis.

1. Nous utiliserons le formulaire fourni. Définir le paramètre action du formulaire avec comme cible le fichier php de l'exercice 1. Ajouter deux boutons radio pour choisir le sexe, un menu déroulant pour choisir la promo. Permettre la sélection de plusieurs centres d'intérêt et plusieurs compétences à la fois. Donner des noms aux champs, effectuer une première soumission et vérifier que toutes les données sont correctement récupérées côté serveur.

2. Définir les labels de chacun des champs, et donner des légendes pour chaque ensemble proposé, à l'aide des balises fieldset/legend. Associer un identifiant css à chaque ensemble pour pouvoir le contrôler. Associer la classe « info » à chacun des labels.
3. Des images sont fournies pour cet exercice, au format svg et png. Quelle est la différence entre ces formats ? Peuvent-il être affichés tous les deux à l'aide d'une balise img ou comme image de fond ?
4. Établir une feuille de style pour cette page :
1. Aligner tous les champs de formulaire verticalement en leur donnant une taille identique.
2. A l'aide de la classe info, afficher à gauche de chaque label un fond comportant l'image « ampoule » fournie.
1. Définir une classe valide et une classe invalide qui sera utilisée pour chaque champ pour lui donner une image de fond. Un champ valide comporte à sa droite un smiley souriant, un champ invalide un smiley mécontent.
2. La bordure d'un champ d'entrée ayant le focus doit être coloriée en rouge.
5. Lorsqu'un champ est cliqué, on souhaite que son contenu soit sélectionné pour faciliter l'édition.
6. Créer une fonction javascript qui sera appelée lorsque l'utilisateur passera la souris au dessus d'un champ label, pour afficher un popup d'explication sur le champ à remplir. Cette fonction prendra en paramètre une chaîne de caractères précisant le nom du champ survolé.

Nous souhaitons offrir à l'utilisateur la possibilité de générer son propre mot de passe automatiquement, et d'afficher ou cacher le champ de mot de passe pour pouvoir visualiser son contenu en toutes lettres.

7. Définir des gestionnaires d'événements pour les boutons du formulaire permettant d'offrir ces fonctionnalités. Le mot de passe généré devra être aléatoire, mais respecter les contraintes indiquées dans l'introduction de l'exercice.

Nous souhaitons informer l'utilisateur de l'état de son formulaire, pour lui préciser s'il a oublié de remplir des champs, et si les champs qu'il a remplis respectent les critères précédents.

8. Développer une fonction prenant en paramètre une référence sur le champ à tester et renvoyant un booléen caractérisant le respect des contraintes. On se servira d'expressions régulières pour l'implémentation de cette fonction. La fonction modifiera également la classe du champ considéré pour lui donner la valeur « valide » ou « invalide ».
9. Appeler la fonction de validation d'un champ à chaque modification de son contenu.
10. Rédiger une fonction validant tous les champs textes d'un coup, en utilisant la méthode getElementsByTagName, et l'appeler lorsque le navigateur a terminé de charger la page