Semestre 1 – Piscine PHP

Rush 1

Bienvenue dans ce premier Rush de piscine ! ☺

Le temps de ce Rush nous allons mettre de côté les notions de PHP que vous avez pu apprendre jusqu'à aujourd'hui.

Ce rush va vous permettre d'aborder le HTML5, CSS3 et Javascript, langages informatiques que vous devrez maîtriser pour mener à bien la plupart de vos projets de cette année.

Les Rushs étant beaucoup plus libres que les exercices, n'hésitez pas à ajouter des fonctionnalités. Ces fonctionnalités pourront être récompensées par des points bonus. Mais n'oubliez pas qu'avant de penser aux bonus, votre projet devra implémenter les fonctionnalités de base !

Sommaire

Etape 0

Vous êtes libres de travailler avec les façons et méthodes que vous souhaitez, mais chacun des membres d'un groupe devra avoir le dossier « /home/piscine/rendu/Piscine_PHP_Rush_1 » identique aux autres membres de son groupe lors de la date de ramassage indiquée sur l'intranet.

Consignes de vos correcteurs pour toutes les soutenances [~40 min]

- Carte étudiant : Première formalité, il faut vérifier la carte de l'étudiant, c'est une habitude à avoir, comme pour le fichier/constante auteur en examen (non applicable en piscine).
- Présentation : L'étudiant doit être acteur de sa soutenance. Vous ne devez que l'écouter ! Il est impératif d'écrire des commentaires dans tous les blocs de notation, pour expliquer ce qui a bien (ou mal) été réalisé, avant de passer à l'étudiant suivant.
    - L'étudiant débute la soutenance en se présentant puis en expliquant l'organisation qu'il a mis en place pour réaliser le projet (rétro-planning, découpage, difficultés rencontrées, tâches non terminées, etc) avec un support à l'appui (git, trello, gantt, uml, doc, pdf, etc.) **[5 min]**
    - Ensuite, l'étudiant présente toutes les fonctionnalités qui auront été implémentés, en suivant scrupuleusement le fil conducteur du sujet, étape par étape. Vous ne noterez donc pas de points pour les fonctionnalités implémentées qui n'auront pas été présentées, d'où l'importance de bien écouter l'étudiant car il ne sera pas possible de revenir sur une notation si l'étudiant a oublié quelque-chose **[15 min]**
    - Puis vous effectuerez une revue de code et poserez des questions à l'étudiant ou proposerez un recode **[10 min]**
- Retours de soutenance, commentaires et norme dont voici les références : **[10 min]**
    - JavaScript : http://jshint.com/
    - CSS3 : https://jigsaw.w3.org/css-validator/
    - HTML5 : https://validator.w3.org/
    - PHP : https://github.com/squizlabs/PHP_CodeSniffer

Toute erreur de norme entraine systématiquement un malus équivalent à 1/4 de la note par langage. C'est-à-dire, si « 1 warning » est détecté en JavaScript et « 42 notices et 1 fatal error » sont détectés en PHP, vous perdez la moitié de vos points. Intéressez-vous à la norme avant de débuter.

Etape 1

Rendu : Piscine_PHP_Rush_1/step_1/

Restrictions :

- Aucun attribut de style dans le HTML, si besoin, il faut un fichier CSS à part.
- Aucun code JavaScript dans le HTML, si besoin, il faut un fichier JS à part.
- **Tout doit être valide W3C !**

Réalisez une page HTML nommée "my_form.php" qui sera **valide W3C**. Elle comportera un formulaire de type POST où l'action se fera sur cette même page.

Les champs du formulaire seront :

- "sexe" de type radio, avec comme valeurs "Homme" ou "Femme",
- "civilite" de type select, avec comme valeurs "M.", "Mme." ou "Mlle.",
- "nom" de type text, avec au moins 2 caractères alphabétiques,
- "prenom" de type text, avec au moins 2 caractères alphabétiques,
- "email" de type email,
- "telephone" de type tel,
- "website" de type url,
- "date de naissance" de type date, au format dd/mm/yyyy,
- "hobbies" de type checkbox, avec comme valeurs possibles : "Jeux video", "Cinema", "Lecture", "Sport" et "Informatique",
- "token" de type hidden, avec pour valeur "my first website",
- "validation" de type submit.

Chaque champ aura son label associé.

Etape 2

Rendu : Piscine_PHP_Rush_1/step_2/

Restrictions :

- Aucun attribut de style dans le HTML, si besoin, il faut un fichier CSS à part.
- Aucun code JavaScript dans le HTML, si besoin, il faut un fichier JS à part.
- **Tout doit être valide W3C !**

Reprenez les pages créées dans la première étape et intégrez une feuille de style nommée « my_style.css » qui sera **valide W3C**.

Vous devrez faire en sorte que :

- La taille de la police (hors titres) ait une taille de 12 pixels,
- Un champ sur 3 doit avoir une bordure de 2 pixels en pointillés de couleur bleue,
- Tous les labels seront alignés avec une longueur de 150 pixels,
- Chaque champ aura son label à sa gauche et sera cliquable (pour mettre le focus),
- Quand le curseur passe au-dessus d'un bouton ou d'un champ, une animation se produit,
- Le bouton de validation aura une ombre verticale noire et des bordures arrondies de 5 pixels.

Etape 3

Rendu : Piscine_PHP_Rush_1/step_3/

Restrictions :

- Aucun attribut de style dans le HTML, si besoin, il faut un fichier CSS à part.
- Aucun code JavaScript dans le HTML, si besoin, il faut un fichier JS à part.
- **Tout doit être valide W3C !**

Reprenez les pages créées dans les étapes précédentes et intégrez un script Javascript nommé « my_script.js ». Ce-dernier comportera une fonction de validation de votre formulaire. Elle sera appelée à chaque fois que l'on cliquera sur le bouton de validation, et autorisera ou non l'envoi du formulaire.

De plus, ce script devra :

- Colorer le fond des champs non remplis en rouge et afficher les erreurs de façon précise.

Bonus 1

Rendu : Piscine_PHP_Rush_1/bonus/

Restrictions :

- Aucun attribut de style dans le HTML, si besoin, il faut un fichier CSS à part.
- Aucun code JavaScript dans le HTML, si besoin, il faut un fichier JS à part.
- **Tout doit être valide W3C !**

Réalisez une page HTML5 nommée "my_navigator.php" qui sera **valide W3C**. Cette page devra être accessible une fois l'utilisateur connecté. Vous devez donc implémenter un formulaire de connexion vérifiant le login/mail et le mot de passe de l'utilisateur voulant se connecter.

Vérifiez sans recharger la page (#ajax) que tous les champs sont remplis correctement et que l'utilisateur qui essaye de se connecter existe bien. Autrement, signalez une erreur à l'aide de JavaScript / CSS, toujours sans actualiser la page.

Vous devez faire en sorte qu'il existe au moins 3 utilisateurs qui puissent se connecter durant votre soutenance.

Bonus 2

Rendu : Piscine_PHP_Rush_1/bonus/

Restrictions :

- Aucun attribut de style dans le HTML, si besoin, il faut un fichier CSS à part.
- Aucun code JavaScript dans le HTML, si besoin, il faut un fichier JS à part.
- **Tout doit être valide W3C !**

Reprenez les pages créées dans les deux étapes précédentes. Vous devrez faire en sorte qu'il soit possible de créer un compte utilisateur lorsque l'on n'est pas connecté. Ce formulaire d'inscription devra implémenter les champs login/mail, password/confirmation.

Si le formulaire est soumis alors qu'un champ n'est pas ou mal rempli, vous devez le signaler à l'utilisateur à l'aide de JavaScript et CSS sans recharger la page.

Le login/mail de l'utilisateur connecté devra apparaitre dans la page accompagné d'un lien de déconnexion. Il faut aussi un lien pour « supprimer le compte ».

Bonus 3

Rendu : Piscine_PHP_Rush_1/bonus/

Restrictions :

- Aucun attribut de style dans le HTML, si besoin, il faut un fichier CSS à part.
- Aucun code JavaScript dans le HTML, si besoin, il faut un fichier JS à part.
- **Tout doit être valide W3C !**

Idées de Bonus :

- Possibilité d'enregistrement des données,
  - Vérification des doublons
- Possibilité d'édition des données,
- Pré-remplissage des différents champs,
- Possibilité d'uploader un avatar dans un champ "avatar" de type file,
- Raccourcis/code clavier pour effectuer des tests unitaires (debug),
- Une interface utilisateur pratique et bien pensée,
- Possibilité de récupérer son mot de passe par mail/question secrète en cas d'oubli,
- Possibilité de modifier son mot de passe une fois connecté,
- Possibilité d'uploader un avatar et de l'afficher une fois connecté,
- Possibilité d'envoyer des messages aux autres utilisateurs,
- Sécurisation des fichiers pour chaque utilisateur,
  - Possibilité de dossiers partagés entre plusieurs utilisateurs
    - Possibilité de gestion des droits par utilisateur/groupe
- Une interface utilisateur pratique et bien pensée,
- Un historique des actions effectuées,
- Full AJAX,
- …