Séance 4 : TP CIR2

JSON

Objectif :

Ajouter une interactivité évoluée avec l'utilisateur en JavaScript, comprendre la notation objet JavaScript JSON, utiliser l'objet Event du navigateur pour récupérer les événements liés au clavier, savoir développer des fonctions robustes capables de réagir lorsque des arguments manquent.

Exercice 1 : Formulaire de création de compte

Nous souhaitons construire un formulaire de création de compte en facilitant la tâche de l'utilisateur. Le formulaire comporte des suggestions de mot de passe en expliquant à l'utilisateur si son mot de passe correspond aux contraintes imposées.

Les contraintes imposées :

- Choisir au moins une lettre minuscule
- Saisir au moins une lettre majuscule
- Saisir au moins un chiffre
- Longueur de mot de passe entre 8 et 12 caractères
- Longueur de login entre 6 et 8 caractères
- 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 fourni (voir annexe)**. 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 :

   • Aligner tous les champs de formulaire verticalement en leur donnant une taille identique.

• A l'aide de la classe info, afficher à gauche de chaque label un fond comportant l'image « ampoule » fournie.

• 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.

• 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.

    Nous souhaitons donner l'impression à l'utilisateur que le formulaire est très court, de manière à ne pas le décourager et obtenir le plus haut taux de transformation de notre « tunnel de création de compte » et ne lui afficher que les champs utiles à chaque étape.

11. Une étape est associée à un ensemble de champs dans un fieldset. Au chargement de la page, seul le premier ensemble est affiché. Un bouton est généré dynamiquement lorsque tous les champs de cet ensemble sont valides, pour passer à l'ensemble suivant.
12. Lorsque ce bouton est cliqué, l'ensemble en cours est caché, l'ensemble suivant s'affiche.
13. Vérifier qu'à la fin de la validation du dernier ensemble de champs, l'ensemble des données de formulaire est bien envoyé au serveur.
14. Générer un fil d'Ariane dans l'entête de votre page permettant à l'utilisateur de connaître le nombre d'étapes de votre tunnel, l'étape en cours et de revenir aux étapes précédentes.

Exercice 2 : Génération dynamique de champs de formulaire

Nous commençons par expérimenter le format JSON, en proposant une fonctionnalité de complétion automatique des champs texte, utilisant un dictionnaire structuré en JSON.

1. Créer un objet JSON définissant quelques étudiants : Identifiant nom, prénom, classe, âge, moyenne.
2. Créer un formulaire muni d'un menu déroulant vide et d'un bouton submit renvoyant à la page php utilisée lors de l'exercice 1.
3. Utiliser l'objet JSON pour ajouter des options à votre menu déroulant. Chaque option présentera les caractéristiques des étudiants définies à la question 1. Le champ select enverra uniquement l'identifiant du l'étudiant lorsque le formulaire est soumis.
4. Faire du code précédent une fonction, prenant deux arguments : le nom du champ qui sera affiché à l'utilisateur, et le nom du champ qui sera envoyé au serveur.
5. Améliorez la robustesse de votre fonction : lorsque les arguments ne sont pas transmis, on enverra par défaut au serveur le champ 'id' de l'objet JSON, et c'est le champ 'label' qui sera affiché à l'utilisateur
6. Vérifier que votre travail continue de fonctionner lorsque le menu déroulant permet la sélection multiple.
7. Ajouter un champ d'entrée texte dans votre formulaire. Il vous permettra d'entrer le prénom & nom d'un étudiant. Placez un gestionnaire d'événement dans ce champ, permettant de capturer les caractères saisis au clavier. A chaque caractère entré, parcourez l'objet JSON pour rechercher les étudiants dont le nom ou le prénom correspondent à la chaine saisie dans le formulaire. Afficher la liste de ces étudiants sous le champ d'entrée texte, de manière que l'on puisse cliquer sur l'un des étudiants. Une fois cliqué, le prénom et le nom de l'étudiant sélectionné s'affichent dans le champ d'entrée texte.

Exercice 3 : Menu Déroulant « ServerLess »

Marre de toujours devoir adapter le code php pour styliser la page côté client ! Que les designers Web s'en chargent !

Dans cet exercice, le serveur se contente de préparer un objet JSON caractérisant les items du menu déroulant à produire. Libre au développeur Javascript d'employer une liste, des div ou une autre structure HTML pour afficher le menu côté client.

1. Créer un objet JSON définissant la structure d'un menu déroulant. On choisira une structure à deux niveaux de profondeur : le premier niveau contient les éléments de menu, le second les éléments du sous-menu correspondant. Chaque élément est défini par son label, un lien hypertexte indiquant où mène cet item, ainsi qu'une description courte.
2. Parcourir l'objet pour produire un menu déroulant affiché sous forme de bannière. Commencer par n'afficher que les éléments du premier niveau. Les éléments de la

bannière devront tous avoir la même taille. On pourra les définir comme flottants pour faciliter leur mise en place.

3. Lorsque la souris passe au-dessus d'un élément de menu, afficher le sous-menu en dessous, en respectant la même largeur que son père. Les éléments du sous-menu devront avoir une hauteur identique.
4. Les éléments de menu doivent être cliquables.
5. Les menus du premier niveau de l'objet JSON contiennent un champ booléen nommé 'selectioned'. Le menu pour lequel ce champ est vrai doit apparaître différemment lors de la production du menu.
6. En plus des animations précédentes, changer la couleur des éléments de menu survolés et afficher un popup contenant la description courte lorsque la souris passe au-dessus d'un menu et y reste plus de deux secondes. S'arranger pour que ce pop-up se place à une position permettant de continuer à lire le contenu des labels des autres items
7. L'objet JSON comporte une propriété numérique caractérisant l'ordre d'affichage des éléments du menu. Respecter cette information lors de la production du menu.
8. Pour les amateurs de récursivité, réfléchir à une manière de traiter des menus de profondeur quelconque.

Exercice 4 : Allons plus loin

Nous souhaitons mettre en œuvre une ergonomie riche pour réaliser l'affectation d'étudiants à des ateliers pendant certaines périodes.

Plutôt que d'utiliser des formulaires, nous souhaitons pouvoir glisser-déposer des étudiants dans des cases d'un tableau affichant les ateliers en ligne et les périodes en colonnes. Des règles supplémentaires doivent être respectées lorsque l'utilisateur relâche un étudiant dans un atelier.

• Des objets JSON caractérisent les offres d'atelier et les périodes, ainsi que les étudiants concernés. On fait l'hypothèse que les ateliers sont proposés pour toutes les périodes

• Une zone de sélection d'étudiants affiche une liste de boites

• Les boites peuvent être glissées dans un tableau atelier*période

• Un étudiant doit être affecté à un atelier/période

• Un étudiant ne peut faire deux fois le même atelier

• Un objet JSON est mis à jour à chaque affectation

• On peut glisser à partir de la zone de sélection ou du tableau

• Lorsqu'un étudiant est affecté pour toutes les périodes, son nom disparait de la zone de sélection

• On peut supprimer un étudiant à l'aide d'une croix placée au-dessus de la boite