Evaluation Web avancé

Générateur de formulaire

Sujets

Vous travaillez dans une entreprise proposant à ses clients B2B des sites permettant de recueillir auprès de ses utilisateurs des besoins plus ou moins complexes. Ces besoins sont recueillis grâce à des formulaires. Le client vous indique la liste des besoins qu'il a besoin de recueillir et vous lui proposez une page web composée de champs de formulaires. Votre solution supporte différents types de champs.
Vous n'avez pas envie de devoir redévelopper chaque page à la main. Vous développez donc un générateur de formulaire. Ce générateur est un composant React prenant en entrée une propriété "blocks" décrivant votre formulaire et permettant d'afficher un formulaire complexe et d'en restituer les données.

Fonctionnalité clés

- Vote formulaire supporte les types de champ suivants :
  - texte
  - nombre
  - liste déroulante
  - case à cocher
  - radio
  - textarea
- Chaque champ doit avoir un identifiant unique et un libellé affiché
- Les champs de formulaire doivent être regroupés par bloc possédant un titre
- Les champs peuvent être initialisés avec un objet de valeur en entrée du composant
- Un champ ou un bloc (regroupement de champ) peuvent être visible ou non selon la valeur d'un autre champ
- Un champ peut être requis (une étoile rouge s'affiche dans ce cas à côté du libellé du champ)
- Un bouton en bas du formulaire permet de le soumettre
  - Si le formulaire n'est pas valide (un champ requis n'est pas rempli), un résumé des erreurs est affiché en haut du formulaire, et un message d'erreur "Ce champ est requis" sera affiché en dessous de chaque champ en erreur
  - A la soumission du formulaire, s'il est valide, les données du formulaire seront affichés dans la console sous la forme d'un objet, les clés seront l'identifiant du champ, la valeur sera la valeur du champ

Pour rappel, toutes ces fonctionnalités doivent être supportées par le fichier de configuration du formulaire.

Voici un exemple de fichier de configuration (vous n'êtes pas obligé de suivre cette structure) :

```
[
 {
   "id": "customer",
   "title": "Infos client",
   "inputs": [
     {
       "id": "firstname",
       "required": true,
       "label": "Prénom",
       "type": "text"
     },
     {
       "id": "age",
       "required": false,
       "label": "Age",
       "type": "number",
       "visibility": "firstname=John"
     }
   ]
 }
]
```

Dans cet exemple, le formulaire n'aura qu'un seul bloc ayant pour titre "Infos client", composé de deux champs :

- Prénom qui est un champ texte requis
- Age qui est un champ nombre non requis mais affiché seulement quand le champ Prénom vaut "John"

Dans cet exemple, si l'utilisateur clique sur le bouton en bas de page pour valider le formulaire, si l'utilisateur a rempli les deux champs. Il verra dans la console un objet :

```
{
 "firstname" : "John",
 "age": 30
}
```

Par contre, s'il n'avait pas rempli le champ Prénom, il verrait en haut de la page un bloc lui indiquant : Le champ "Prénom" est requis, et en dessous du champ Prénom un message "Ce champ est requis".

Votre projet doit être fait par groupe de 2 à 3 personnes.

Conseils et pistes

Vous devez respecter les fonctionnalités demandées mais l'implémentation technique est libre selon plusieurs conditions :

- En React
- Un seul composant permettant de prendre en entrée la configuration et les valeurs par défaut

En vrac :

- Travaillez en équipe
- Répartissez vous les tâches
- Attention au performance
- N'hésitez pas à ajouter des fonctionnalités supplémentaires comme des règles de validations autres que simplement le champ requis

Notations et points d'attentions

L'écriture de test d'interaction est un bonus.

Affichage de tous les types de champs avec libellé	6
Séparation des champs par bloc	2
Le composant peut être initialisé avec des valeurs	2
Gestion des erreurs	3
Gestion de la visibilité des champs	3
Affichage d'un log à la soumission	2
Affichage des erreurs à la soumission	2
Bonus (test, fonctionnalités supplémentaires, typescript, etc)	

Rendu

Le rendu devra se faire sous la forme d'une archive .zip avant le 29 mai 2023 à 23h59 par mail à l'adresse johnathan.meunier@enigma-school.com.
Si vous avez utilisé GIT pour versionner votre projet, vous pouvez me partager le repo Github : johnmeunier
N'oubliez pas de supprimer le dossier node_modules avant l'envoi du projet (vérifiez bien la validité de votre package.json par exemple en supprimant vos node_modules, les réinstaller, et vérifier que votre projet fonctionne toujours). L'objet du mail devra être sous la forme [3WEB] 2122 Prenom-NOM
Vous devez envoyer un mail même si vous m'avez partagé le repo Github en m'envoyant au moins le lien du repo.