Exercice : Calculateur d'IMC Avancé avec Historique Local

Créez une application web complète qui calcule l'Indice de Masse Corporelle (IMC) et maintient un historique des mesures, le tout **sans aucun *backend***.

La Tâche

L'application doit présenter un formulaire de saisie pour le **Poids (en kg)** et la **Taille (en cm)**. Après le calcul, elle doit afficher immédiatement le résultat de l'IMC avec son interprétation (catégorie) et ajouter cette mesure à une liste d'historiques qui persiste après le rafraîchissement du navigateur.

Formule Essentielle

Le cœur de la logique JavaScript réside dans l'application de la formule de l'IMC :

$$\text{IMC} = \frac{Poids(kg)}{Taille(m^2)}$$

Crucial : Vous devez d'abord convertir la taille saisie en centimètres en **mètres** pour effectuer le calcul correctement.

Exigences Techniques

Pour réussir cet exercice, vous devez maîtriser les concepts suivants :

1. **Manipulation du DOM et Gestion des Événements (JavaScript) :**
    - Gérez l'événement de soumission du formulaire (`'submit'`) et utilisez **`event.preventDefault()`** pour bloquer le rechargement de la page.
    - Créez dynamiquement de nouveaux éléments HTML (`document.createElement()`) pour afficher chaque nouvelle entrée dans la liste de l'historique.
    - Implémentez la logique de **catégorisation** de l'IMC (Minceur, Normale, Surpoids, Obésité, etc.) et mettez à jour la classe CSS du résultat en fonction.
2. **Persistance des Données (JavaScript) :**
    - Utilisez l'API **`localStorage`** pour sauvegarder un tableau (Array) de toutes les mesures historiques. C'est votre substitut de base de données.
    - Vous devez utiliser **`JSON.stringify()`** avant de stocker dans `localStorage` et **`JSON.parse()`** lors de la récupération des données au chargement de l'application.
3. **Gestion de l'Historique Dynamique (JavaScript) :**
    - Chaque entrée d'historique affichée doit inclure un bouton ou une icône de suppression.

    - Utilisez la **délégation d'événements** pour intercepter les clics sur ces boutons de suppression, retirer l'élément correspondant du tableau dans JavaScript, mettre à jour `localStorage`, et rafraîchir le DOM.
4. **Conception et Esthétique (CSS) :**
    - Structurez la mise en page principale de l'application à l'aide de **Flexbox** ou de **CSS Grid** pour garantir un design **réactif (responsive)**.
    - Créez des **classes CSS distinctes** (ex: `.result-normal`, `.result-obese`) pour appliquer des styles et couleurs spécifiques au résultat immédiat.
    - Utilisez les **transitions CSS** pour introduire des changements fluides (par exemple, lors du changement de couleur du résultat) afin d'améliorer l'expérience utilisateur.
5. **Robustesse et Validation (HTML & JavaScript) :**
    - Appliquez les attributs HTML de validation de base (`required`, `min`, `max`) sur les champs de saisie.
    - Ajoutez une couche de **validation JavaScript** pour vérifier la plausibilité des données (pas vide, valeurs dans une plage raisonnable) et affichez un message d'erreur clair si la validation échoue.