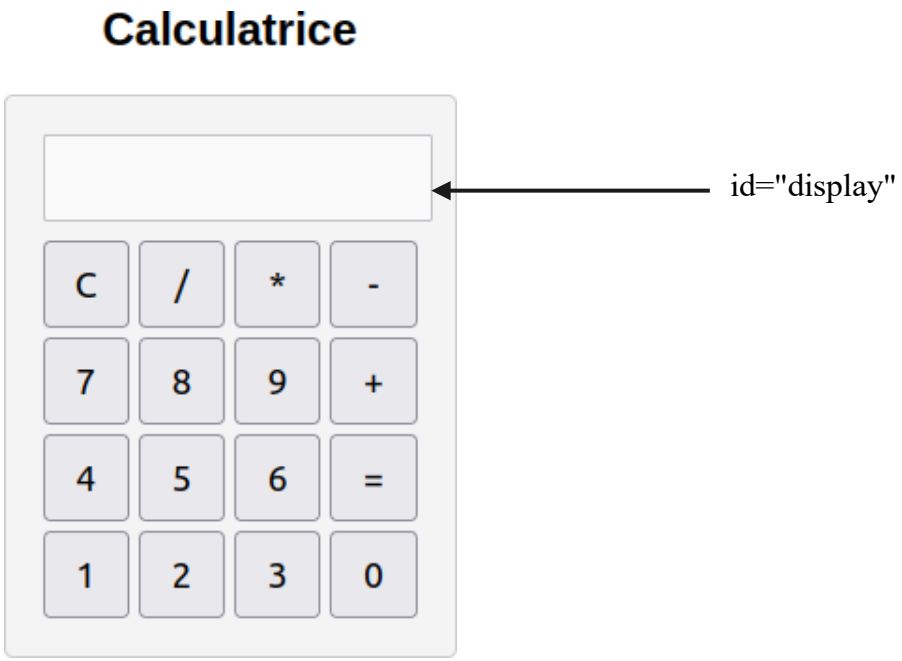
I. Objectif du TP

L'objectif de ce TP est de se familiariser avec les notions fondamentales de JavaScript, ainsi qu'avec son utilisation en complément de HTML et CSS. Ce TP comprend l'apprentissage des concepts de base du langage (les variables, les types de données, les structures de contrôle, les fonctions, etc.) . De plus, il aborde la manipulation du DOM (Document Object Model), permettant d'interagir dynamiquement avec la structure d'une page web.

Exercice 01 :

L'objectif de cet exercice est de réaliser une calculatrice simple permettant d'effectuer des opérations mathématiques de base (addition, soustraction, multiplication, division), comme le montre la figure suivante :



i. Éditez le fichier « calculatrice.html » et comprenez sa structure.

ii. Ajoutez le code JavaScript au fichier «« calculatrice.html » en respectant les directives et les étapes suivantes :

Étape 1 : Affichage des chiffres et opérateurs

1. Créez une fonction nommée `appendValue(value)`. Cette fonction doit :
   - Récupérer l'élément `<input>` du champ d'affichage grâce à `document.getElementById()`.

- Ajouter la valeur du bouton cliqué (chiffre ou opérateur) à la valeur actuelle du champ d'affichage (En utilisant l'attribut `value` : `document.getElementById("un_id").value = …`).

2. Modifiez le code HTML pour appeler appendValue(value) lorsque l'utilisateur clique sur un bouton (l'événement `onclick`) .

Étape 2 : Effacement de l'affichage

3. Créez une fonction nommée `clearDisplay()`. Elle doit :
   - Vider complètement le champ d'affichage en lui assignant une chaîne vide "".
4. Modifiez le code HTML pour appeler `clearDisplay()` lorsque l'utilisateur clique sur le bouton "C".

Étape 3 : Calcul du résultat

5. Créez une fonction nommée `calculateResult()`. Elle doit :
   - Utiliser la fonction `eval()` pour évaluer l'expression mathématique contenue dans le champ d'affichage (ex : `let r=eval("5+9") ; //r contiendra la valeur 14`).
   - Mettre à jour le champ d'affichage avec le résultat.
6. Modifiez le code HTML pour appeler `calculateResult()` lorsque l'utilisateur clique sur le bouton "=".
7. Modifiez `calculateResult()` pour éviter les erreurs : Affichez "Erreur" dans le champ d'affichage en cas de problème (Vous pouvez utiliser `try...catch`).

Question Supplémentaire : Utilisation du clavier,

Modifier le code JavaScript de la calculatrice pour permettre à l'utilisateur d'entrer des nombres et des opérations via le clavier en plus des boutons cliquables.

8. Ajoutez un écouteur d'événements (`addEventListener`) sur l'objet `document` pour détecter les touches pressées (`keydown`) :

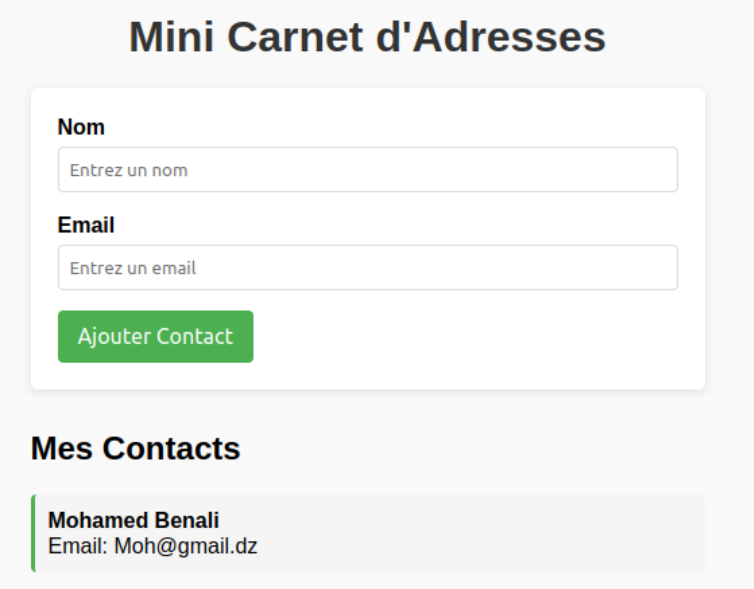
```
document.addEventListener("keydown", function(event) {
        // le code de la fonction ( suivez les directives suivantes)
});
```

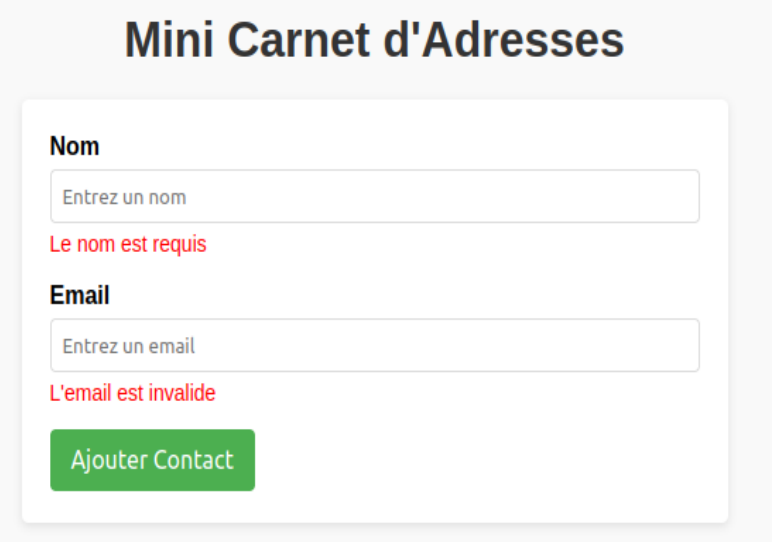
9. Déclarez une variable contenant les touches valides (0-9, +, -, *, /, .).
10. Si la touche pressée fait partie de ces touches valides, appelez la fonction `appendValue(event.key)`.

11. Si l'utilisateur appuie sur `Backspace`, supprimez le dernier caractère de l'affichage (Utilisez `document.getElementById("display").value.slice(0, -1)` pour la suppression).
12. Si l'utilisateur appuie sur `Escape`, effacez complètement l'affichage (`clearDisplay()`).

Exercice 02: Mini Carnet d'Adresses.

Le but de cet exercice est de créer un mini carnet d'adresses en JavaScript permettant d'ajouter des contacts avec leur nom et leur email, puis de les afficher dans une liste. Des messages d'erreur doivent être affichés si les valeurs saisies ne sont pas valides (voir les figures suivantes). Le fichier HTML (avec CSS) est déjà fourni (carnetAdresses.html); vous devez uniquement implémenter la partie JavaScript dans (carnetAdresses.js) .





i. Éditez le fichier « carnetAdresses.html » et comprenez sa structure.

ii. Ajoutez le code JavaScript au fichier «carnetAdresses.js » en respectant les directives et les étapes suivantes :

Étape 1 : Récupération des éléments du formulaire et variables globales

1. Récupérez tous les éléments HTML nécessaires pour manipuler le formulaire et la liste des contacts : *nom, email, nom-error, email-error, liste-contacts et contact-form*. En utilisant `document.getElementById()`.

   Exemple : `const nom = document.getElementById('nom');`

Étape 2 : Création de la fonction de validation du formulaire :

Créez une fonction nommée `validerFormulaire()` qui retourne un booléen (true si le formulaire est valide, false sinon), la validation se fait comme suit :

2. Vérifiez que le champ **nom** n'est pas vide : `nom.value.trim() !== ''`
   - Si le champ est vide, affichez le message d'erreur en modifiant la propriété display de l'élément d'erreur à 'block'.
   - Sinon, masquez le message d'erreur en modifiant la propriété `display` à `'none'`.

3. Vérifiez que l'email est dans un format valide en utilisant une expression régulière :
   - Déclarez une variable qui contient le pattern de l'email : ex :

     const emailPattern = /[a-zA-Z0-9._\-]+@[a-zA-Z0-9\-]+\.[a-zA-Z]{2,}/;
   - Vérifiez que l'email est dans un format valide en utilisant le pattern et la fonction `'test'` :

     `emailpattern.test(email.value.trim()) ;` (retourne true si la valeur saisie dans le champ email est conforme par rapport au pattern, et retourne false sinon.)
   - Si le champ email est valide, affichez le message d'erreur en modifiant la propriété display de l'élément d'erreur à 'block'.
   - Sinon, masquez le message d'erreur en modifiant la propriété `display` à `'none'`.

Étape 3 : Création de la fonction pour ajouter un contact.

Créez une fonction nommée `ajouterContact(nom, email)` qui prend en paramètre les valeurs nom et email saisies par l'utilisateur. Dans cette fonction :

4. Créez un nouvel élément div avec `document.createElement('div')` et ajoutez la classe `'contact-item'` à cet élément , ex : `contactElement.className = 'contact-item';`

5. Définissez le contenu HTML de l'élément pour afficher le nom et l'email :

   ```
   contactElement.innerHTML=`<strong>${nom}</strong><br> Email:${email}`;
   ```

6. Ajoutez l'élément au conteneur des contacts (*liste-contacts*) avec `appendChild()`.

Étape 4 : Gestionnaire d'événement pour traiter la soumission du formulaire

7. Ajoutez un écouteur d'événement `submit` au formulaire en utilisant `addEventListener()` avec une fonction ayant le squelette suivant :

```
form.addEventListener('submit', function(event) {
    // Empêcher le comportement par défaut du formulaire
    // Valider le formulaire
    // Ajouter le contact à la liste
    // Réinitialiser le formulaire
    }
});
```

- Empêcher le comportement par défaut du formulaire pour éviter que le formulaire soit envoyer en utilisant : `event.preventDefault();`
- Valider le formulaire à l'ai de la fonction `validerFormulaire()`.
- Ajouter le contact à la liste en appelant la fonction `ajouterContact(nom, email)`.
- Réinitialiser le formulaire à l'aide de : `form.reset();`

Question Supplémentaire : Ajouter un bouton pour supprimer un contacte.

Ajoutez un bouton supprimer qui vous permet à supprimer un contacte, comme le montre la figure suivante :