Exercice 1 : Créer une application de liste de tâches

Objectifs :

- Manipuler le DOM avec JavaScript.
- Utiliser des événements en JavaScript.
- Gérer des données avec le stockage local (localStorage).

Étapes :

1. **Créer la structure HTML de base :**
    - Un champ de texte pour ajouter de nouvelles tâches.
    - Un bouton pour ajouter des tâches.
    - Une liste pour afficher les tâches ajoutées.
2. **Ajouter du style avec CSS :**
    - Stylez la liste pour qu'elle soit visuellement agréable.
    - Ajoutez des styles pour les tâches complétées et pour les boutons de suppression.
3. **Écrire la logique en JavaScript :**
    - Ajouter une nouvelle tâche à la liste.
    - Marquer une tâche comme complétée.
    - Supprimer une tâche de la liste.
    - Sauvegarder les tâches dans le localStorage pour persistance.

Fichiers à créer :

1. `index.html`
2. `style.css`
3. `script.js`

1. Structure HTML

Créez un fichier `index.html` avec la structure de base suivante :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Liste de tâches</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
```

```
    <div id="app">
        <h1>Ma Liste de Tâches</h1>
        <input type="text" id="new-task" placeholder="Nouvelle tâche">
        <button id="add-task">Ajouter</button>
        <ul id="task-list"></ul>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

2. Style CSS

Ajoutez un fichier `style.css` pour le style de base :

```
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
}

#app {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

input, button {
    padding: 10px;
    margin: 5px;
}

ul {
    list-style: none;
    padding: 0;
}

li {
    display: flex;
    justify-content: space-between;
    padding: 10px;
    background: #f9f9f9;
    margin: 5px 0;
    border-radius: 4px;
```

```
}

li.completed {
    text-decoration: line-through;
    color: #999;
}

body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
}

#app {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

input, button {
    padding: 10px;
    margin: 5px;
}

ul {
    list-style: none;
    padding: 0;
}

li {
    display: flex;
    justify-content: space-between;
    padding: 10px;
    background: #f9f9f9;
    margin: 5px 0;
    border-radius: 4px;
}

li.completed {
    text-decoration: line-through;
    color: #999;
}
```

3. Script JavaScript

Créez un fichier `script.js` pour la logique de votre application. Les étapes suivantes sont décrites pour vous guider :

1. **Récupérer les éléments du DOM.**
2. **Ajouter une nouvelle tâche à la liste.**
3. **Marquer une tâche comme complétée.**
4. **Supprimer une tâche.**
5. **Sauvegarder les tâches dans le localStorage.**
6. **Charger les tâches depuis le localStorage au démarrage.**

Voici un guide pour commencer, mais le code n'est pas fourni :

- Sélectionnez les éléments nécessaires du DOM (input, bouton, ul).
- Ajoutez un événement 'click' sur le bouton pour ajouter une nouvelle tâche.
- Ajoutez un événement 'click' sur chaque tâche pour la marquer comme complétée.
- Ajoutez un bouton de suppression pour chaque tâche, avec un événement 'click' pour supprimer la tâche.
- Sauvegardez les tâches dans le localStorage à chaque modification.
- Chargez les tâches depuis le localStorage lorsque la page se charge.

Challenge :

- Ajoutez une fonctionnalité pour éditer une tâche existante.
- Ajoutez une confirmation avant de supprimer une tâche.

Exercice 2 : Créer une application de calculatrice

Objectifs :

- Manipuler le DOM avec JavaScript.
- Gérer les événements de clic.
- Implémenter la logique de base pour les opérations arithmétiques.

Étapes :

1. **Créer la structure HTML de base :**
    - Un affichage pour montrer les entrées et les résultats.
    - Des boutons pour les chiffres, les opérations arithmétiques et les actions (C, =).
2. **Ajouter du style avec CSS :**
    - Stylez la calculatrice pour qu'elle soit visuellement agréable et facile à utiliser.
3. **Écrire la logique en JavaScript :**
    - Gérer les clics sur les boutons pour mettre à jour l'affichage.
    - Implémenter la logique pour effectuer les opérations arithmétiques.
    - Gérer les erreurs (comme la division par zéro).

Fichiers à créer :

1. `index.html`
2. `style.css`
3. `script.js`

1. Structure HTML

Créez un fichier `index.html` avec la structure de base suivante :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calculatrice</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="calculator">
```

```
        <div id="display">0</div>
        <div id="buttons">
            <button class="btn" data-value="7">7</button>
            <button class="btn" data-value="8">8</button>
            <button class="btn" data-value="9">9</button>
            <button class="btn operator" data-value="/">/</button>
            <button class="btn" data-value="4">4</button>
            <button class="btn" data-value="5">5</button>
            <button class="btn" data-value="6">6</button>
            <button class="btn operator" data-value="*">*</button>
            <button class="btn" data-value="1">1</button>
            <button class="btn" data-value="2">2</button>
            <button class="btn" data-value="3">3</button>
            <button class="btn operator" data-value="-">-</button>
            <button class="btn" data-value="0">0</button>
            <button class="btn" data-value=".">.</button>
            <button class="btn" data-value="C">C</button>
            <button class="btn operator" data-value="+">+</button>
            <button class="btn equals" data-value="=">=</button>
        </div>
    </div>
```

2. Style CSS

Ajoutez un fichier `style.css` pour le style de base :

```
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
}

#calculator {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    display: grid;
    grid-template-rows: 1fr 4fr;
}

#display {
    background: #333;
```

```
    color: white;
    font-size: 2em;
    text-align: right;
    padding: 20px;
    border-radius: 4px;
    margin-bottom: 10px;
}

#buttons {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    grid-gap: 10px;
}

.btn {
    background: #f9f9f9;
    border: none;
    padding: 20px;
    font-size: 1.5em;
    border-radius: 4px;
    cursor: pointer;
    transition: background 0.
```

3. Script JavaScript

Créez un fichier `script.js` pour la logique de votre application. Voici un guide pour commencer, mais le code n'est pas fourni :

1. **Récupérer les éléments du DOM.**
2. **Gérer les clics sur les boutons pour mettre à jour l'affichage.**
3. **Implémenter la logique pour les opérations arithmétiques.**
4. **Gérer les erreurs (comme la division par zéro).**

Voici un guide pour vous aider à structurer votre script :

- Sélectionnez les éléments nécessaires du DOM (affichage, boutons).
- Ajoutez des événements de clic sur les boutons pour mettre à jour l'affichage.
- Implémentez les opérations arithmétiques de base (+, -, *, /).
- Ajoutez des fonctionnalités pour les boutons 'C' (Clear) et '=' (Equal).

Challenge :

- Ajoutez une fonctionnalité pour gérer les nombres négatifs.
- Ajoutez un bouton pour la racine carrée.

Exercice 3 : Créer un jeu de devinettes de nombre

Objectifs :

- Manipuler le DOM avec JavaScript.
- Gérer les événements de clic.
- Utiliser des conditions pour la logique du jeu.

Étapes :

1. **Créer la structure HTML de base :**
    - Un champ de texte pour entrer une supposition.
    - Un bouton pour soumettre la supposition.
    - Un affichage pour montrer le résultat (trop haut, trop bas, correct).
    - Un affichage pour montrer le nombre d'essais restants.
    - Un bouton pour réinitialiser le jeu.
2. **Ajouter du style avec CSS :**
    - Stylez l'application pour qu'elle soit visuellement agréable et facile à utiliser.
3. **Écrire la logique en JavaScript :**
    - Générer un nombre aléatoire entre 1 et 100.
    - Gérer les soumissions pour vérifier si la supposition est correcte, trop haute ou trop basse.
    - Mettre à jour l'affichage en conséquence.
    - Gérer le nombre d'essais restants et afficher un message lorsque le joueur n'a plus d'essais.
    - Réinitialiser le jeu lorsqu'on clique sur le bouton de réinitialisation.

Fichiers à créer :

1. `index.html`
2. `style.css`
3. `script.js`

1. Structure HTML

Créez un fichier `index.html` avec la structure de base suivante :

```
<!DOCTYPE html>
<html lang="fr">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Jeu de Devinettes de Nombre</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="game">
        <h1>Devinez le Nombre</h1>
        <p>Entrez un nombre entre 1 et 100 :</p>
        <input type="number" id="guess-input" min="1" max="100">
        <button id="submit-guess">Soumettre</button>
        <p id="result-message"></p>
        <p id="attempts-remaining">Essais restants : 10</p>
        <button id="reset-game">Réinitialiser</button>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

2. Style CSS

Ajoutez un fichier `style.css` pour le style de base :

```
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
}

#game {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}

input, button {
    padding: 10px;
    margin: 5px;
}

#result-message {
    margin-top: 20px;
```

```
    font-size: 1.2em;
}

#attempts-remaining {
    margin-top: 10px;
    font-size: 1em;
    color: #555;
}
```

3. Script JavaScript

Créez un fichier `script.js` pour la logique de votre application. Voici un guide pour commencer, mais le code n'est pas fourni :

1. **Récupérer les éléments du DOM.**
2. **Générer un nombre aléatoire entre 1 et 100.**
3. **Gérer les soumissions pour vérifier la supposition.**
4. **Mettre à jour l'affichage en conséquence.**
5. **Gérer le nombre d'essais restants.**
6. **Réinitialiser le jeu lorsque le bouton de réinitialisation est cliqué.**

Voici un guide pour structurer votre script :

1. **Variables globales :**
    - Nombre aléatoire à deviner.
    - Nombre d'essais restants.
2. **Fonctions :**
    - `initializeGame()` : Réinitialise le jeu.
    - `checkGuess()` : Vérifie la supposition et met à jour l'affichage.
    - `updateMessage(message, isError)` : Met à jour le message affiché.
3. **Événements :**
    - Ajouter un événement de clic pour le bouton de soumission.
    - Ajouter un événement de clic pour le bouton de réinitialisation.

Code de départ pour `script.js` :

```
let randomNumber;
let attemptsRemaining;
const guessInput = document.getElementById('guess-input');
const submitGuessButton = document.getElementById('submit-guess');
const resultMessage = document.getElementById('result-message');
const attemptsRemainingDisplay = document.getElementById('attempts-remaining');
const resetGameButton = document.getElementById('reset-game');

function initializeGame() {
    randomNumber = Math.floor(Math.random() * 100) + 1;
    attemptsRemaining = 10;
```

```
    resultMessage.textContent = '';
    attemptsRemainingDisplay.textContent = `Essais restants : ${attemptsRemaining}`;
    guessInput.value = '';
}

function checkGuess() {
    const userGuess = Number(guessInput.value);
    if (userGuess < 1 || userGuess > 100 || isNaN(userGuess)) {
        updateMessage('Veuillez entrer un nombre entre 1 et 100.', true);
        return;
    }
    attemptsRemaining--;
    attemptsRemainingDisplay.textContent = `Essais restants : ${attemptsRemaining}`;
    if (userGuess === randomNumber) {
        updateMessage('Félicitations! Vous avez deviné le nombre!', false);
        submitGuessButton.disabled = true;
        return;
    } else if (userGuess < randomNumber) {
        updateMessage('Trop bas!', true);
    } else {
        updateMessage('Trop haut!', true);
    }
    if (attemptsRemaining === 0) {
        updateMessage(`Game over! Le nombre était ${randomNumber}.`, true);
        submitGuessButton.disabled = true;
    }
}

function updateMessage(message, isError) {
    resultMessage.textContent = message;
    resultMessage.style.color = isError ? 'red' : 'green';
}

submitGuessButton.addEventListener('click', checkGuess);
resetGameButton.addEventListener('click', initializeGame);

initializeGame();
```

Challenge :

- Ajoutez un champ pour permettre aux joueurs de choisir le nombre maximum (par exemple, entre 1 et 500).
- Ajoutez un compteur de score pour garder une trace des meilleurs scores (nombre minimum de tentatives pour deviner correctement).

Exercice4 : Créer un convertisseur de devises

Objectifs :

- Manipuler le DOM avec JavaScript.
- Gérer les événements de saisie et de sélection.
- Effectuer des calculs de conversion de devises.

Étapes :

1. **Créer la structure HTML de base :**
    - Un champ de saisie pour le montant à convertir.
    - Deux menus déroulants pour sélectionner les devises de départ et d'arrivée.
    - Un bouton pour effectuer la conversion.
    - Un affichage pour montrer le résultat de la conversion.
2. **Ajouter du style avec CSS :**
    - Stylez l'application pour qu'elle soit visuellement agréable et facile à utiliser.
3. **Écrire la logique en JavaScript :**
    - Récupérer les taux de change (vous pouvez utiliser des taux de change fictifs pour simplifier).
    - Gérer les événements de saisie et de sélection pour mettre à jour le calcul.
    - Effectuer les calculs de conversion et afficher le résultat.

Fichiers à créer :

1. `index.html`
2. `style.css`
3. `script.js`

1. Structure HTML

Créez un fichier `index.html` avec la structure de base suivante :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Convertisseur de Devises</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="converter">
        <h1>Convertisseur de Devises</h1>
```

```
        <input type="number" id="amount" placeholder="Montant">
        <select id="from-currency">
            <option value="USD">USD</option>
            <option value="EUR">EUR</option>
            <option value="GBP">GBP</option>
            <!-- Ajoutez d'autres devises si nécessaire -->
        </select>
        <select id="to-currency">
            <option value="USD">USD</option>
            <option value="EUR">EUR</option>
            <option value="GBP">GBP</option>
            <!-- Ajoutez d'autres devises si nécessaire -->
        </select>
        <button id="convert">Convertir</button>
        <p id="result"></p>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

2. Style CSS

Ajoutez un fichier `style.css` pour le style de base :

```
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
}

#converter {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}

input, select, button {
    padding: 10px;
    margin: 5px;
    font-size: 1em;
}
```

```
#result {
    margin-top: 20px;
    font-size: 1.2em;
}
```

3. Script JavaScript

Créez un fichier `script.js` pour la logique de votre application. Voici un guide pour commencer, mais le code n'est pas fourni :

1. **Définir des taux de change fictifs.**
2. **Récupérer les éléments du DOM.**
3. **Gérer les événements de clic et de changement de valeur.**
4. **Effectuer les calculs de conversion et afficher le résultat.**

Voici un guide pour structurer votre script :

1. **Taux de change fictifs :**
    - Définissez un objet pour les taux de change (par exemple, `const exchangeRates = { USD: 1, EUR: 0.85, GBP: 0.75 }`).
2. **Variables globales :**
    - Montant à convertir.
    - Devise de départ.
    - Devise d'arrivée.
3. **Fonctions :**
    - `convertCurrency()` : Effectue la conversion et met à jour l'affichage.
4. **Événements :**
    - Ajouter un événement de clic pour le bouton de conversion.
    - Ajouter des événements de changement de valeur pour les champs de saisie et les menus déroulants.

Code de départ pour `script.js` :

```
const exchangeRates = {
    USD: 1,
    EUR: 0.85,
    GBP: 0.75
    // Ajoutez d'autres devises et leurs taux de change ici
};

const amountInput = document.getElementById('amount');
const fromCurrency = document.getElementById('from-currency');
const toCurrency = document.getElementById('to-currency');
const convertButton = document.getElementById('convert');
const resultDisplay = document.getElementById('result');

function convertCurrency() {
```

```
    const amount = parseFloat(amountInput.value);
    const from = fromCurrency.value;
    const to = toCurrency.value;

    if (isNaN(amount) || amount <= 0) {
        resultDisplay.textContent = 'Veuillez entrer un montant valide.';
        return;
    }

    const convertedAmount = (amount / exchangeRates[from]) * exchangeRates[to];
    resultDisplay.textContent = `${amount} ${from} équivaut à ${convertedAmount.toFixed(2)}
${to}.`;
}

convertButton.addEventListener('click', convertCurrency);
```

Challenge :

- Ajoutez une fonctionnalité pour récupérer les taux de change en temps réel à partir d'une API publique.
- Ajoutez une animation lors de la conversion pour améliorer l'expérience utilisateur.