Rapport du Mini-Projet Calculatrice en JavaScript

Formation : Technicien Spécialisé en DI - 2ème Année
Module : Programmation Client-Serveur (JavaScript)
Année : 2025-2026

Table des matières

Introduction

Ce projet consiste en la réalisation d'une calculatrice simple (non scientifique) en utilisant les technologies web HTML, CSS et JavaScript. L'application permet d'effectuer des opérations mathématiques de base et conserve un historique des trois dernières opérations grâce à l'API Web localStorage.

L'interface utilisateur a été conçue pour être intuitive et moderne, avec un design sombre inspiré des calculatrices iOS. La calculatrice est entièrement responsive et peut être utilisée aussi bien au clavier qu'à la souris.

Structure du projet

Le projet est organisé de manière claire et structurée, séparant les responsabilités entre les différents fichiers :

Fichier	Rôle	Technologies
`index.html`	Structure de la page	HTML5
`styles.css`	Mise en forme et design	CSS3
`script.js`	Logique de calcul et historique	JavaScript ES6

Cette séparation respecte le principe de séparation des préoccupations (Separation of Concerns), facilitant la maintenance et l'évolution du code.

Description des fichiers

index.html

Le fichier HTML définit la structure de l'application avec deux sections principales :

La première section est la **calculatrice** elle-même, composée d'une zone d'affichage (expression en cours et résultat) et d'une grille de boutons. Les boutons sont organisés en cinq lignes comprenant les fonctions spéciales (AC, parenthèses, pourcentage), les chiffres de 0 à 9, les opérateurs arithmétiques, le point décimal, le backspace et le bouton égal.

La seconde section est l'**historique** qui affiche les trois dernières opérations effectuées. Cette zone est mise à jour dynamiquement par JavaScript.

styles.css

Le fichier CSS implémente un design moderne avec les caractéristiques suivantes :

Le thème sombre utilise un dégradé de fond allant du bleu nuit (#1a1a2e) au bleu marine (#16213e). Les boutons sont circulaires avec des coins arrondis à 50%, créant un aspect visuel similaire aux calculatrices modernes. Les couleurs distinguent les

différents types de boutons : gris foncé (#333333) pour les chiffres, bleu (#007AFF) pour les opérateurs et fonctions spéciales.

Le design est responsive grâce à l'utilisation de media queries qui adaptent la taille des éléments pour les écrans plus petits.

script.js

Le fichier JavaScript contient toute la logique de l'application, organisée en plusieurs sections clairement commentées. Chaque fonction est documentée avec des commentaires JSDoc expliquant son rôle, ses paramètres et sa valeur de retour.

Explication des fonctions JavaScript

Fonctions d'affichage

Fonction	Description
`updateDisplay()`	Met à jour l'affichage de l'expression et du résultat dans l'interface. Elle est appelée après chaque modification de l'expression pour synchroniser l'affichage avec l'état interne.
`displayHistory()`	Récupère l'historique depuis localStorage et génère le HTML correspondant pour l'afficher dans la section historique. Si l'historique est vide, un message informatif est affiché.

Fonctions de gestion de l'expression

Fonction	Description
`appendToExpression(value)`	Ajoute un caractère à l'expression en cours. Cette fonction gère plusieurs cas spéciaux : les parenthèses (délégation à `handleParentheses()`), le pourcentage, la prévention de plusieurs opérateurs consécutifs, et la validation des points décimaux.
`handleParentheses()`	Gère l'ajout intelligent de parenthèses. Elle alterne automatiquement entre parenthèse ouvrante et fermante selon le contexte, en utilisant un compteur pour suivre les parenthèses non fermées.
`backspace()`	Supprime le dernier caractère de l'expression. Elle met également à jour le compteur de parenthèses si le caractère supprimé est une parenthèse.
`clearAll()`	Réinitialise complètement la calculatrice en effaçant l'expression et en remettant le compteur de parenthèses à zéro.

Fonctions de calcul

Fonction	Description
`calculate()`	Fonction principale de calcul. Elle convertit les symboles visuels (×, ÷) en opérateurs JavaScript (* , /), ferme les parenthèses non fermées, évalue l'expression et sauvegarde le résultat dans l'historique.
`evaluateExpression(expression)`	Évalue une expression mathématique de manière sécurisée. Elle vérifie d'abord que l'expression ne contient que des caractères autorisés, puis utilise `Function()` pour l'évaluation (plus sécurisé que `eval()`).
`formatResult(result)`	Formate le résultat pour un affichage propre. Elle limite le nombre de décimales, supprime les zéros inutiles et utilise la notation scientifique pour les très grands nombres.

Fonctions utilitaires

Fonction	Description
`isOperator(char)`	Vérifie si un caractère est un opérateur mathématique (+, -, ×, ÷, *, /). Retourne un booléen.
`getLastNumber()`	Récupère le dernier nombre de l'expression en cours. Utilisée pour vérifier si un point décimal peut être ajouté.

Fonctions de gestion de l'historique

Fonction	Description
`getHistory()`	Récupère l'historique depuis localStorage. Retourne un tableau vide si aucun historique n'existe ou si les données sont corrompues.
`saveToHistory(expression, result)`	Sauvegarde une nouvelle opération dans l'historique. L'opération est ajoutée au début du tableau et seules les trois dernières sont conservées.
`clearHistory()`	Efface complètement l'historique en supprimant la clé correspondante dans localStorage.

Gestion de l'historique avec localStorage

L'API Web localStorage permet de stocker des données de manière persistante dans le navigateur. Contrairement aux cookies, les données localStorage n'ont pas de date d'expiration et ne sont pas envoyées au serveur à chaque requête.

Fonctionnement

L'historique est stocké sous forme de chaîne JSON avec la clé `calculatrice_historique`. Chaque entrée de l'historique contient trois propriétés : l'expression calculée, le résultat obtenu et un horodatage.

```
// Structure d'une entrée d'historique
{
    expression: "9+6",
    result: "15",
    timestamp: "2026-01-14T10:30:00.000Z"
}
```

Méthodes utilisées

La méthode `localStorage.getItem(key)` récupère une valeur stockée. La méthode `localStorage.setItem(key, value)` sauvegarde une valeur. La méthode `localStorage.removeItem(key)` supprime une entrée. Les fonctions `JSON.parse()` et `JSON.stringify()` permettent de convertir entre objets JavaScript et chaînes JSON.

Limitation à 3 opérations

Conformément aux exigences du projet, seules les trois dernières opérations sont conservées. Lors de chaque nouvelle sauvegarde, si le tableau dépasse trois éléments, les plus anciens sont supprimés avec la méthode `slice(0, 3)`.

Fonctionnalités implémentées

Fonctionnalités de base

L'application permet d'effectuer les quatre opérations arithmétiques fondamentales : addition, soustraction, multiplication et division. Le calcul du pourcentage est également disponible, convertissant automatiquement la valeur en fraction décimale.

Fonctionnalités avancées

La gestion des parenthèses permet d'effectuer des calculs complexes avec priorité des opérations. Le système d'alternance automatique facilite la saisie en ajoutant la parenthèse appropriée selon le contexte.

Support clavier

La calculatrice peut être entièrement contrôlée au clavier. Les touches numériques, les opérateurs (+, -, *, /), la touche Entrée pour calculer, Backspace pour effacer et Escape pour réinitialiser sont tous supportés.

Gestion des erreurs

Les erreurs de calcul (division par zéro, expressions invalides) sont interceptées et affichent un message “Erreur” à l’utilisateur sans faire planter l’application.

Conclusion

Ce mini-projet de calculatrice démontre l’utilisation pratique des technologies web fondamentales. Le HTML structure le contenu, le CSS apporte un design moderne et responsive, et le JavaScript gère toute la logique métier et l’interaction utilisateur.

L’utilisation de localStorage pour l’historique illustre comment les applications web peuvent conserver des données localement sans nécessiter de serveur backend. Le code est organisé de manière modulaire avec des fonctions bien documentées, facilitant sa compréhension et sa maintenance.

Les bonnes pratiques de développement ont été respectées : séparation des préoccupations, commentaires explicatifs, gestion des erreurs et validation des entrées utilisateur.