CPE Lyon - 3ETI - Année 2019/2020
Techniques et Langages d'Internet
Examen 1ère session - 18 décembre 2019

Tous documents papiers autorisés

Les 3 exercices sont **indépendants** ; le barème et la durée indiqués sont donnés **à titre indicatif**.

⚠ Lisez le sujet et les énoncés en entier

⚠ Toute réponse **non rédigée** ou **non justifiée** ne sera pas considérée !!!

Exercice 1. Questions de cours (20 min) / 4 pts

Répondez au questionnaire directement sur la feuille jointe. **N'oubliez pas de rendre le questionnaire avec votre copie** !

Exercice 2. Canvas / SVG (20 min) / 3 pts

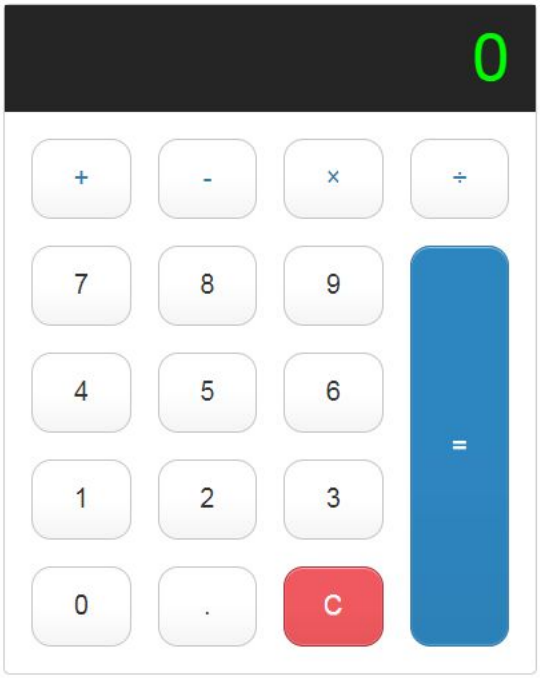
Question 1 : Quelles sont les différences entre SVG et HTML5 Canvas ? Écrivez un code en HTML5 Canvas qui affiche trois cercles concentriques (des ellipses qui partagent le même centre.) Répétez le même exercice en utilisant SVG.

Question 2 : Expliquez avec des exemples les différentes signatures et les objectifs des fonctions suivantes (HTML5 Canvas) :

1. `drawImage()`
2. `beginPath()`
3. `closePath()`

Exercice 3. HTML / CSS / JS (80 min) / 13 pts

Le but de l'exercice est de réaliser une calculatrice en HTML, CSS et JavaScript semblable à celle ci-dessous :



Cette calculatrice se compose de deux parties :

- une zone correspondant à l'écran de la calculatrice,
- une zone regroupant les différentes touches de la calculatrice.

Partie 1 : HTML

Question 1.1 : Quels éléments HTML utiliser pour l'écran et les touches de la calculatrice (on demande simplement le nom de la ou des balises) ?

Question 1.2 : Quels conteneurs HTML utiliser pour la calculatrice, l'écran et la zone des touches ?

Question 1.3 : Comment faire en sorte qu'on ne puisse pas taper du texte dans la zone "écran" ?

Question 1.4 : Ecrivez le code HTML décrivant la calculatrice (**Ne vous préoccupez pas pour l'instant du CSS ni du JavaScript, qui seront traités dans les questions suivantes**.

Partie 2 : Style / CSS

Les différents éléments de la calculatrice sont identifiés et stylisés par des classes CSS : `calculator`, `calculator-screen`, `calculator-keys`, `calculator-equal`...

Question 2.1 : Ecrivez le code CSS de la classe `calculator` : la calculatrice mesure 400 pixels de large et est matérialisée par une bordure pleine, de 1 pixel d'épaisseur, grise et aux coins légèrement arrondis. Elle doit être centrée horizontalement et verticalement sur la page.

Question 2.2 : Ecrivez le code CSS de la classe `calculator-screen` : l'écran occupe toute la largeur de la calculatrice, et mesure 80 pixels de haut. Il a un fond noir, et un remplissage de 10 pixels à gauche et 20 pixels à droite. Le texte affiché est écrit avec une police de taille 5 rem, en vert et aligné à droite.

Question 2.3 : Ecrivez le code CSS de la classe `calculator-keys` : les touches sont disposées à l'aide d'un Grid Layout, en quatre colonnes. Il y a 20 pixels d'espace autour des touches. Quand on survole une touche (sauf la touche `C` et la touche `=`), la couleur de fond devient gris foncé.

Question 2.4 : Ecrivez le code CSS de la classe `calculator-equal` : la touche `=` est de couleur bleue, et s'étire dans la dernière colonne sur les quatre dernières lignes. Pour le contraste, le symbole `=` lui-même est écrit en blanc. Quand on survole la touche, le fond passe en couleur `4e9ed4`.

Partie 3 : JavaScript

Rappel : une expression arithmétique est composée d'*opérandes* séparés par un *opérateur*. Par exemple, dans l'expression `1 + 2`, les opérandes sont `1` et `2`, et `+` est l'opérateur.

Question 3.1 : La calculatrice est modélisée par un *objet* JavaScript `calculator` possédant les attributs suivants :

- `displayValue` est une chaîne de caractères qui représente le nombre saisi par l'utilisateur, ou le résultat d'une opération ; il est initialisé à `0` ;
- `firstOperand` contient le premier opérande d'une expression ; il est initialisé à `null` ;
- `operator` contient l'opérateur d'une expression ; il est initialisé à `null` ;
- `waitingForSecondOperand` est un *drapeau* (c'est-à-dire une variable booléenne), qui vaut `true` si on attend qu'un deuxième opérande soit saisi (typiquement, après avoir cliqué sur un opérateur), ou `false` sinon.

Ecrivez le code JavaScript correspondant à l'objet `calculator`.

Question 3.2 : Ecrivez le code de la fonction `resetCalculator()` qui réinitialise la calculatrice.

Question 3.3 : Ecrivez le code de la fonction `updateDisplay()` qui actualise l'écran de la calculatrice avec le contenu de la variable `displayValue` à chaque nouvelle saisie.

Question 3.4 : Ecrivez le code de la fonction `inputDigit(digit)` qui modifie l'objet `calculator` lorsque l'utilisateur appuie sur une touche de type *chiffre* et en fonction de l'état de `waitingForSecondOperand`.

Question 3.5 : Ecrivez le code de la fonction `inputDecimalPoint()` qui gère l'appui sur la touche *point décimal*. Un opérande ne peut pas contenir plusieurs points, ni commencer par un point.

Question 3.6 : En vous servant du code suivant :

```
const performCalculation = {
    '/': (firstOperand, secondOperand) => firstOperand / secondOperand,
    '*': (firstOperand, secondOperand) => firstOperand * secondOperand,
    '+': (firstOperand, secondOperand) => firstOperand + secondOperand,
    '-': (firstOperand, secondOperand) => firstOperand - secondOperand,
    '=': (firstOperand, secondOperand) => secondOperand
};
```

écrivez le code de la fonction `handleOperator(operator)` qui gère l'appui sur une des touches de type *opérateur* ou `=`. Si on appuie sur deux opérateurs successivement, le dernier annule le premier. On doit pouvoir enchaîner plusieurs opérations sans appuyer sur la touche `=`. Par exemple, la séquence `1 + 2 + 3` est valide et affiche `6` si on appuie sur la touche `=`.

Question 3.7 : Enfin, écrivez le code qui associe le clic sur chacun des boutons à la fonction correspondante.