Programmation Web Avancée Côté Client
TP n° 4

Licence Informatique 2ème année
Université de Nice-Sophia Antipolis

Introduction

Dans cette session de travaux pratiques, nous allons tester notre compréhension des méchanismes d'orientation objet du langage JavaScript.

1 Nombres complexes

1. Écrivez un constructeur qui crée des objets « nombre complexe ». Représentez le nombre complexe par sa partie réelle `re` et sa partie imaginaire `im`. Rappellons qu'un nombre complexe $z \in \mathbb{C}$ peut être écrit comme $x+iy$, où $x, y \in \mathbb{R}$ et i est l'unité imaginaire, telle que $i^2 = -1$: la partie réelle de z est alors x et sa partie imaginaire est y.
2. Dotez les nombres complexes des méthodes suivantes :
   - le module `abs()` ($\|a + ib\| = \sqrt{a^2 + b^2}$) — la méthode pour calculer la racine carrée en Javascript est `Math.sqrt()` ;
   - le conjugué `conj()` ($\overline{a + ib} = a - ib$) ;
   - le négatif `neg()` ($-(a + ib) = -a - ib$) ;
   - le réciproque `rec()` ($z^{-1} = \overline{z}/\|z\|^2$) ;
   - l'addition `plus()` ($(a + ib) + (c + id) = (a + c) + i(b + d)$) ;
   - la subtraction `minus()` ($z - z' = z + (-z')$) ;
   - la multiplication `times()` ($(a + ib)(c + id) = (ac - bd) + i(ad + bc)$) ;
   - la division `div()` ($z/z' = z \cdot (z'^{-1})$).
3. Créez une page HTML temporaire pour tester la correction des méthodes écrites.

2 Suites de nombres complexes

Nous allons maintenant utiliser les objets que nous venons de définir.

1. Créez une page HTML avec deux `<input type="text">` permettant à l'utilisateur de saisir la partie réelle et la partie imaginaire d'un nombre complexe c ;
2. Ajoutez un bouton « calculer suite A » qui calcule et affiche les éléments d'une suite définie par la récurrence

$$\begin{cases} z_0 &= 0, \\ z_{i+1} &= z_i^2 + c, \end{cases} \tag{1}$$

tels que $\|z_i\| < 2$, pour $i = 1, \ldots, 100$. Utilisez pour cela une liste `<ol>`.

3. Implémentez la méthode `toString()` pour visualiser correctement un nombre complexe en forme algebrique : $x + iy$; si $y = 0$, seule la partie réelle devra être affichée ; si $y < 0$, le nombre devra être affiché comme $x - i|y|$.
4. Ajoutez un autre bouton « calculer suite B » qui calcule et affiche les éléments d'une suite définie par la récurrence

$$\begin{cases} z_0 &= 0, \\ z_{i+1} &= \frac{z_i + c}{z_i \cdot c + i}, \end{cases} \tag{2}$$

tels que $\|z_i\| < 2$, pour $i = 1, \ldots, 100$. Pour ne pas dupliquer du code inutilement, pensez à définir un constructeur `Suite` qui crée un objet suite avec une propriété `value` contenenant la valeur actuelle, une méthode `calc()` qui s'occupe de l'affichage et une méthode `next()`, qui calcule la prochaine valeur de la suite. Il suffira donc de redéfinir cette dernière méthode pour obtenir une suite différente.

3 S'il vous reste du temps...

1. Inventez votre propre suite et ajoutez un autre bouton pour la calculer.
2. Faites en sorte qu'à chaque fois que l'utilisateur demande de calculer une suite, la suite précédemment affichée soit effacée avant de visualiser la suite demandée.
3. Il existe une représentation alternative des nombres complexes : la représentation polaire : $z = r(\cos\theta + i\sin\theta) = re^{i\theta}$. Il se trouve qu'en cette représentation, les opérations de réciproque, multiplication et division deviennent beaucoup plus simples à calculer :

$$\begin{aligned} (re^{i\theta})^{-1} &= \frac{1}{r}e^{-i\theta}, \\ re^{i\alpha} \cdot se^{i\beta} &= rse^{i(\alpha+\beta)}, \\ re^{i\alpha} / se^{i\beta} &= \frac{r}{s}e^{i(\alpha-\beta)}. \end{aligned}$$

Modifiez le code que vous avez écrit pour utiliser cette représentation. Si votre code est bien écrit, vous devriez pouvoir ne modifier que l'implémentation des nombres complexes, sans rien toucher au reste du code, qui les utilise.