SN1 - Les fonctions et les procédures en Javascript

1. Cas d'utilisation d'une fonction d'une fonction (ou d'une procédure)

- Pour diviser la complexité d'un problème :
  On écrit plusieurs fonctions qui effectuent chacune un traitement élémentaire. On pourra tester séparément ces fonctions pour les utiliser ensuite dans la résolution du problème.

- Quand un traitement est répétitif :
  On écrit une fonction (ou une procédure) qu'on appellera chaque fois que cela est nécessaire. On gagne ainsi en lisibilité et en compacité du code.

- Quand on veut réutiliser ultérieurement un traitement :
  Une fonction (ou une procédure) testée et mise au point peut être intégrée à une bibliothèque (collection des fonctions et/ou de procédures).
  Les exemples sont très nombreux en Javascript (Math…).

2. Différence entre fonction et procédure

- Une fonction effectue un calcul et renvoie un résultat.

- Une procédure effectue un traitement (affichage, modification de données…) et ne renvoie pas de valeur.

3. Déclaration

- Une fonction ou une procédure se déclare avec le mot clé `function`,

- Une fonction ou une procédure peut recevoir un ou plusieurs arguments,

- Une fonction retourne une ou plusieurs valeurs avec le mot clé **return**.

- Bonnes pratiques : Mettre un commentaire d'entête pour décrire ce que fait la fonction (documentation du code Javascript)

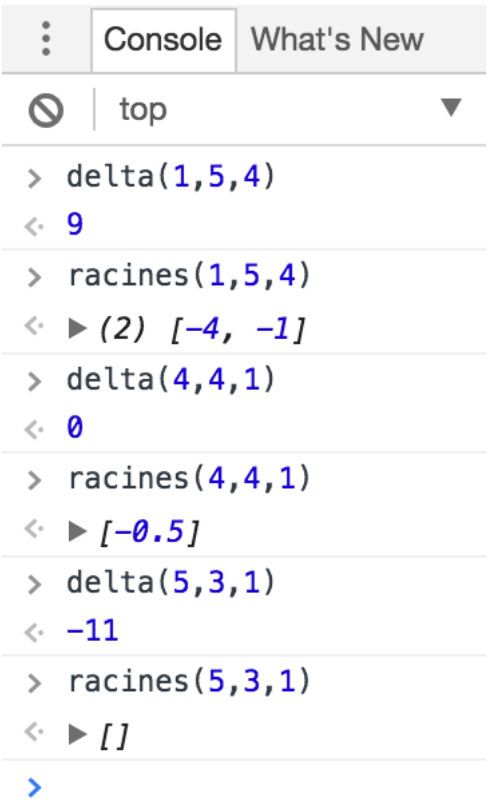
```
// Fonction calculant le discriminant d'une équation
// du second degré : ax2 + bx + c =0
function delta(a, b, c) {
    return (b**2 - 4*a*c)
}
```

4. Utilisation

- Une fonction s'utilise (appel) par son nom et la valeur des paramètres dans les parenthèses, comme ici dans une autre fonction :

```
// Fonction calculant les racines d'une équation
// du second degré : ax2 + bx + c =0
function racines(a, b, c) {
    var d = delta(a, b ,c);
    var solutions = [];
    if (d > 0) {
        solutions.push((-b-Math.sqrt(d))/(2*a));
        solutions.push((-b+Math.sqrt(d))/(2*a));
    }
    else if (d == 0) {
        solutions.push(-b/(2*a));
    }
    return solutions;
}
```

- Ou par la console Javascript :



5. Exercices

- Taper et tester le code Javascript des fonctions **delta(a,b,c)** et **racines(a,b,c)** dans un fichier **Eq2degre.html**

- Compléter le code avec ces fonctions ou procédures :

```
function saisie(parametre) {
    return (prompt("Entrer la valeur de " + parametre + " :"));
}

function afficheRacines(a, b, c) {
    var chaine = "L'équation " + a + "x2 + " + b + "x + " + c + " = 0";
    chaine += "\n" + "a pour solution(s) : ";
    chaine += racines(a, b ,c);
    alert(chaine);
}
```

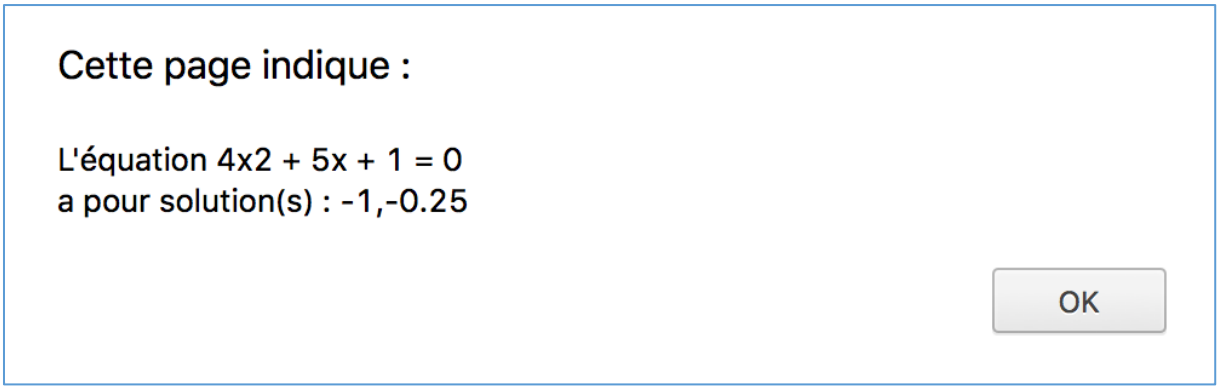
- **saisie(parametre)** est-elle une fonction ou une procédure ?

- **afficheRacines(a,b,c)** est-elle une fonction ou une procédure ?

- Créer les commentaires d'entête de ces fonctions ou procédures.

- Créer un programme principal à la suite permettant de saisir les 3 paramètres a, b, et c et d'afficher les racines de l'équation $ax^2 + bx + c = 0$

- Tester votre programme. Voici un exemple d'affichage du résultat :



6. **Exercices d'application**

- Créer et tester une fonction qui saisit une borne (nombre entier).
- Créer et tester une fonction qui renvoie une liste de nombres premiers inférieurs à une borne passée en paramètre.
- Créer et tester un programme principal qui saisit une borne et affiche la liste des nombres premiers inférieurs à cette borne.