Javascript pour les solutions de l'équation $x^2 + 3 = 3$

En Javascript , une ***variable*** est créée au moment où on la déclare, par « var », et son type (booléen, nombre, texte ou tableau) est déterminé par sa valeur initiale. Par exemple, avec « var x="coucou" », Javascript saura que x est une chaîne de caractères, parce que "coucou" en est une.
L'affectation d'une variable se note « = ». Par exemple en écrivant « x=2 », on veut que x soit égal à 2, on demande à Javascript de faire en sorte que ce soit le cas, et comme Javascript est serviable, Javascript fait en sorte que ce soit le cas, ce après quoi x est effectivement égal à 2. En pseudolangage on note $2 \to x$ (ou éventuellement $x \leftarrow 2$).
Pour modifier une variable, on l'affecte à nouveau, par sa nouvelle valeur. Pour ajouter par exemple 1 à x, ce qui en pseudocode se note par $x + 1 \to x$, on peut écrire x=x+1 (ce qui est faux, x n'est pas égal à x+1, dans ce cas on doit comprendre que le nouveau x est égal à l'ancien x plus un...) ou x+=1.
Par exemple, pour ajouter $\frac{1}{3}$ à x, on écrit indifféremment x=x+1/3 ou x+=1/3.
Aussi il revient au même d'écrire x+=-2 ou x-=2 puisque dans les deux cas, on soustrait 2 à x.
Si c'est 1 qu'on ajoute, on peut abréger par x++ (et si c'est -1 qu'on ajoute à x, on peut écrire x--).
Ainsi, pour ajouter 1 à n (on dit « incrémenter n »), on peut écrire l'une quelconque des instructions suivantes :
n=n+1
n+=1
n++
++n

Variables booléennes

Valeurs

Une variable booléenne ne prend que deux valeurs possibles, « true » et « false ».
Par exemple, « 2==2 » vaut *true*,

« 4 !=4 » prend la valeur *false*,
« 3<3 » prend la valeur *false*,
et « 3<=3 » prend la valeur *true*.

Négation

La négation est notée « ! ». Ainsi !true vaut false et !false vaut true.
Les variables booléennes !(x==y) et x !=y sont les mêmes (les deux disent que $x \neq y$).

Conjonction

La conjonction « et » est notée &&.
On a true&&true=true, true&&false=false, false&&true=false et false&&false=false.

Disjonction

La disjonction « ou alors » est notée ||.
On a true||true=true, true||false=true, false||true=true et false||false=false.

Relations entre variables

L'égalité se note ==, l'inégalité large <= et l'inégalité stricte <.
On a aussi != (différent de), >= et >.
Que donnera Math.PI==3.14 ? true ou false ?

Nombres

Opérations

Les quatre opérations se notent +, -, * et /. La division est celle des réels.
Pour représenter un nombre négatif, le mettre entre parenthèses.
Ainsi (-2.5)+(-4) donne -6.5
(-2.5)-(-4) donne 1.5
(-2.5)*(-4) donne 10
et (-2.5)/(-4) donne 0.625.

La division par zéro produit « NaN » (abréviation de « not a number ») en guise de message d'erreur (plutôt que dire qu'on ne peut pas diviser par zéro, Javascript préfère dire qu'on peut mais que le quotient n'est pas un nombre). Ainsi « 0/0 » donne « NaN ». La division de 1 par 0 donne un résultat qui sera expliqué en cours de première.
Le reste dans la division euclidienne se note %. Ainsi le reste de 2010 par 4 s'obtient par 2010 % 4 et vaut 2 : 2010 n'est pas bissextile. Il n'est pas nécessaire que le diviseur soit entier. Ainsi 10 % Math.PI donne

0.5752220392306207

ce qui signifie qu'un angle de 10 radians a la même mesure qu'un angle d'environ 0,575222 radians. Pour élever x à la puissance n, on écrit Math.pow(x,n). Ainsi Math.pow(5,3) et 5*5*5 sont deux manières différentes de trouver que $5^3 = 125$.
Pour calculer la racine carrée, voir la section « Mathématiques ».

Infini

N'importe quel nombre trop grand pour être traité par Javascript est considéré comme infini (positif), et noté « Infinity ». Ainsi Infinity+10 donne Infinity puisque en rajoutant 10, la somme est encore trop grande. Par contre, ce qui est moins évident, c'est que Infinity-10 donne aussi Infinity. La notation $+\infty$ correspond assez bien calculatoirement parlant à ce concept mais $+\infty$ n'est pas considéré comme un nombre.
De même, -Infinity désigne un nombre trop petit pour être traité par Javascript (il correspond à $-\infty$).

Fonctions mathématiques

Arrondis

Math.round(x) donne l'arrondi de x à l'entier le plus proche.
Par exemple, si on calcule
Math.sqrt(2)*Math.sqrt(2),
on trouve 2.0000000000000004 alors que $\sqrt{2} \times \sqrt{2} = \left(\sqrt{2}\right)^2 = 2$ par définition. Cette erreur d'approximation n'est pas due à *JavaScript* mais au navigateur internet qui l'interprète. Or on a parfois besoin d'être sûr qu'un nombre est entier (par exemple dans

"for(n=0 ;n<=10 ;n++)" où le nombre n sert à compter des boucles), et pour ce faire on complète l'exemple précédent en le suivant :
Math.round(Math.sqrt(2)*Math.sqrt(2)) qui, lui, donnera bien 2.

Math.floor(x) donne la troncature (l'arrondi entier par défaut) de x.
Math.ceil(x) donne l'arrondi par excès de x (en anglais, plancher se dit floor et plafond se dit ceiling).
La fonction toFixed(2) arrondit à deux décimales. Par exemple, Math.PI.toFixed(2) donne 3.14

Puissances et racines

Math.pow(x,n) calcule x^n. Donc l'inverse de x se calcule par Math.pow(x,-1) (en anglais, puissance se dit power).
Math.sqrt(x) donne la racine carrée de x (square root en anglais). Si on essaye Math.sqrt(-4), on obtient NaN qui rappelle qu'un réel négatif n'a pas de racine carrée.
Math.abs(x) donne la valeur absolue de x (le nombre obtenu en forçant le signe de x à être positif). Ainsi Math.abs(-6) donne 6. La valeur absolue de x se note $|x|$ et on a $|x| = \sqrt{x^2}$.

trigonométrie

Math.cos(x) donne le cosinus de x, *en radians*
Math.sin(x) donne le sinus de x, *en radians*
Math.tan(x) donne la tangente de x, *en radians*

Minimum et maximum

Le plus grand des deux nombres x et y est donné par Math.max(x,y)
et le plus petit des deux par Math.min(x,y).