les variables

Les variables sont une manière de faire référence à un objet.

Ce sont des outils qui permettent de stocker une valeur.

On va donc **déclarer** une variable et lui **affecter** une valeur.

A quoi ça sert ?

cela permet de généraliser le programme et de nommer les entrées, sorties ou éléments intermédiaires. C’est ensuite ces noms qui seront utilisés dans les algorithmes, peut importe les valeurs qui leur sont affectées.

Les types de données

- logique : boolean
- chaîne de caractères
- numérique

On choisira toujours un nom de variable clair qui représente ce que l’on souhaite nommer et manipuler

Explication technique: déclaration et affectation

Une variable doit être définies. On utilise alors l'un de ces 2 mots clés

- let -> variable
- const -> variable immuable

const toto;

let titi;

On affecte une valeur à une variable avec le caractère =

toto = 3;

Si on affecte pas de valeur à une variable, celle-ci a la valeur **undefined**

Explication technique: les types de valeurs

JS est un langage à typage dynamique, l'interpréteur de JS attribue un type à la variable au moment de l'exécution et ce en fonction du type de sa valeur.

Les types primitifs :

Boolean: const vrai = true; const faux = false

Number: const deux = 2, const deuxEtDemi = 2.5

String: Texte entouré de simple, double ou back quotes

- const chaineValide = “oui”
- const chaineValide = ‘oui’
- const chaineValide = `oui`
- const chaineValide = ‘L\’ouïe’

Explication technique: les opérations sur les variables

Il est possible d'effectuer des opération sur les variables:

opérateur +, -, /, *, %(modulo)...

quels opérateur pour quels types ?

remarque : le typage étant dynamique, tout type peut être évalué comme un booléen.

Pratique

Définition:

Le pseudo-code est une représentation textuelle d'un algorithme ne dépendant d'aucun langage de programmation.

- Un partie déclaration et une partie instruction
- l’affectation est représentée par le caractères ←

Fiche exercice variable

les fonctions

Regroupement logique d’instructions auquel on donne un nom

A quoi ça sert ?

La fonction permet :

- de découper son code
- de donner du sens et de la lisibilité à son code
- de réutiliser des parties de code sans les dupliquer

Explication technique

une fonction a plusieurs caractéristiques:

- Possède un nom
- Peut prendre un ou plusieurs paramètres
- Peut retourner une valeur (pour en retourner plusieurs, il faut les retourner dans un objet ou un tableau)

Elle doit d’abord être déclarée.

Elle ne sera ensuite exécutée que lorsqu’on y fera référence : on dit alors qu’on appelle la fonction.

Les fonctions sont traitées comme toutes les autres variables en JS (first-class functions), on peut donc les stocker mais aussi les retourner !

Explication technique: le scope

Le scope représente la visibilité de la variable dans le programme.

Une variable est scopée et n’est pas accessible en dehors du scope dans lequel elle a été déclarée

Exemple

Différentes façon de les **déclarer**: (principalement pour des questions de scope):

```
function maFonction(argument1, argument2) {
        // Corps de la fonction
        console.log(“arguments:”, argument1, argument2)
}

const maFonction = function (argument1, argument2) {
        // Corps de la fonction
        console.log(“arguments:”, argument1, argument2)
}

const maFonction = (argument1, argument2) => {
        // Corps de la fonction, ECMAScript 2015
        console.log(“arguments:”, argument1, argument2)
}
```

Appeler une fonction

```
maFonction(“oui”, “non”)
        “arguments: oui non”

maFonction(“oui”)
        “arguments: oui undefined”

maFonction()
        “arguments: undefined undefined”
```

Pratique

Mario kart