C.M. / T.D. Algorithmique avec JavaScript

Sorbonne Université
GEOMAT – Groupes 2 et 3
par Maxime Forriez

Télécharger l'AlgoBox

- https://www.xm1math.net/algobox/index.html
- https://www.xm1math.net/algobox/doc.html

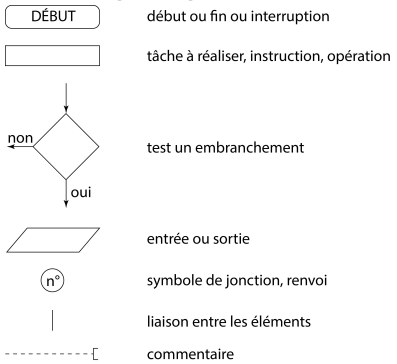
Qu'est-ce qu'un algorithme ?

- Le mot « algorithme » vient du nom du mathématicien persan, Muhammad ibn Musa al-Khuwarizmi dit Al Khuwarizmi (IXe siècle), latinisé en Algoritmi.
- Un algorithme est une **recette**. Il s'agit d'une **suite d'instructions** qui, lorsqu'elles sont exécutées correctement, aboutissent au résultat attendu. C'est un énoncé en **langage clair**, bien défini et ordonné.
- Un algorithme décrit une **méthode de résolution** de problèmes courants, le plus souvent par le calcul.

Qu'est-ce qu'un algorithme ?

- Apprendre l'algorithmique (ou algorithmie), c'est apprendre à programmer dans les **règles de l'art**.
- L'algorithmique doit permettre le développement de **mécanisme de pensée par l'apprentissage** permettant d'optimiser les traitements que le développeur doit programmer.
  - Vitesse de calcul
  - Occupation de la mémoire
  - Quantité de lignes de programmation
  - *etc.*

Organigramme



Le langage de description d'algorithmes (L.D.A.)

- Un **pseudo-code** fait référence à une **syntaxe ressemblant à un code**.
  - Il indique le fonctionnement d'une syntaxe de code.
  - Il illustre la conception d'un élément d'architecture de code.
  - Il ne fait aucune référence à un quelconque langage de programmation.
- Le pseudo-code permet de décrire avec **plus ou moins** de détails l'algorithme.
  - Il permet de passer certains éléments complexes temporairement pour se concentrer sur la vision d'ensemble du code.
- **Attention !** Le pseudo-code n'a **aucune convention**, mais il existe un certain consensus.
  - https://users.csc.calpoly.edu/~jalbey/SWE/pdl_std.html

L'utilité du pseudo-code

1. Mesurer la difficulté de la mise en œuvre d'un algorithme
2. Mesurer la difficulté de développer une démarche structurée dans la construction d'un algorithme

N.B. 1. Il n'est pas nécessaire de multiplier les éléments de pseudo-code. Il faut se limiter à l'essentiel :
- Lire
- Afficher
- Saisir
- Enregistrer
- *etc.*

N.B. 2. Le pseudo-code peut être représenté sous la forme d'un organigramme.

7

Les éléments universels (ou presque) dans tout programme

- Les variables
- Les opérateurs
- Les instructions
- Les conditions
- Les boucles
- Les boucles conditionnelles
- Les fonctions / Les procédures
- Les objets

8

JavaScript

- Le JavaScript est un langage orienté objet à **prototypes** (1996).

  Attention ! Le JavaScript est un langage multiparadigme :
  - paradigme de script
  - paradigme impératif
  - paradigme fonctionnel

9

Intégration d'un script dans une page HTML

L'insertion d'un script s'effectue en fin de page juste avant la balise </body>.

Exceptionnellement, il est possible d'insérer des scripts dans l'entête. Cela reste vivement déconseillé. En effet, la balise <script> est bloquante, c'est-à-dire que, tant que le script n'est pas totalement exécuté, la page s'arrête de charger.

- Méthode 1 :
  - <script> *code en JavaScript* </script>
- Méthode 2 :
  - <script src="*nom du fichier*.js"></script>

10

Intégration d'un script dans une page HTML

```
<!DOCTYPE html>
<html lang="fr">
   <head>
        <meta charset="UTF-8">
        <title>Tests JavaScript</title>
   </head>
   <body>
        <h1>Tests JavaScript</h1>
        <script src="test.js"></script>
   </body>
</html>
```

11

Les instructions

- Une instruction est une séquence identifiée par un début et une fin
  - DÉBUT
    - INSTRUCTIONS…
  - FIN
- En général, les instructions constituent un **bloc**.

12

La syntaxe en JavaScript

- // Commentaire sur une ligne
- /* Commentaire sur plusieurs lignes */
- { Instructions }
- Toute ligne se termine par un ;

N.B. Il est possible de s'en passer. Néanmoins, en PHP, il est obligatoire. Il est par conséquent vivement conseillé de s'habituer à mettre le point-virgule final.

13

Les variables

- Une variable représente quelque chose.
- Une variable a un nom (https://dev.to/tontz/comment-nommer-ses-variables-fonctions-et-classes-clean-code-37p6).
- Une variable doit être affectée par un opérateur.
  - L'affectation s'opère de droite à gauche. La variable à gauche subit l'affection, la variable de droite permet l'affectation.
    - X = 1
    - X = X +1
    - Que vaut X ?
- Une variable peut avoir un type :
  - Nombre entier (int)
  - Nombre réel (double, float, *etc.*)
  - Booléen (0 ou 1 ; OUI ou NON ; VRAI ou FAUX)
  - Chaîne de caractères (texte)
    Attention ! La chaîne de caractères est une variable ayant des spécificités propres qu'il faut approfondir pour n'importe quel langage de programmation.
  - *etc.*
- Une variable peut être constante.
- Une variable est soit globale (tout le programme), soit locale (spécifique à une instruction).

14

Les variables en pseudo-code

Déclaration d'une variable

VAR

Nom de la variable : Type de variable

Affectation d'une variable

Nom de la variable ← Valeur de la variable

15

Les variables en JavaScript

- Toute variable en JavaScript est déclarée par le modificateur :
  - **let** (ou var dans les anciennes versions)

    let a;

    let b;

    N.B. On peut déclarer plusieurs variables en même temps en les séparant par une virgule.

    let a, b;
- Toute constante est déclarée par le modificateur
  - **const**

    const A;

N.B. Il est possible de **déclarer** une variable sans l'affecter.

16

Les variables en JavaScript

- L'opérateur d'affectation des variables est **=**.

  let c = *valeur*;

  let c = 1;

  let c = "Chaîne de caractères";

  let c = true;

  let c = null;

  let c = undefined;
- Le typage des variables est automatique en JavaScript. La fonction typeof(*variable*) permet de connaître le type de la variable.
- Le nom d'une variable doit être **explicite**.
  - Conventions : https://www.gekkode.com/developpement/6-regles-pour-savoir-comment-nommer-les-variables-en-javascript/

17

Les variables en JavaScript

- Pour tester votre code, vous aurez besoin d'afficher les variables.
- Deux méthodes :
  - console.log(*variable*);
  - alert(*variable*);
- Une méthode pour saisir des données
  - let saisie = prompt("Quel est ton nom ?");

18

Les opérateurs

- En général, les langages ont :
  - les opérateurs mathématiques classiques
    - +, -, *, /, modulo
  - les opérateurs de comparaison
    - ==, != , <, >, <=, >=
  - les opérateurs logiques
    - OU, ET, NON
  - *etc.*

19

Les opérateurs en JavaScript

- Les opérateurs d'affectation
- Les opérateurs arithmétiques
- Les opérateurs de comparaison
- Les opérateurs de chaîne de caractères
- Les opérateurs logiques

20

Les opérateurs d'affectation en JavaScript

=	opérateur d'affectation
+=	ajout d'une valeur à une variable affectée
-=	soustraction d'une valeur à une variable affectée
*=	multiplication d'une valeur par une variable affectée
/=	division d'une valeur par une variable affectée
%=	modulo d'une valeur et d'une variable affectée

21

Les opérateurs arithmétiques en JavaScript

+	addition
-	soustraction
*	multiplication
/	division
%	modulo
++	incrément de 1 post-incrémentation : N++ (N=N, puis N+1) pré-incrémentation : ++N (N+1, puis N=N+1)
--	décrément de 1 post-décrémentation pré-décrémentation
(…)	parenthèse

22

Les opérateurs de comparaison en JavaScript

==	égalité
!=	inégalité
===	identité
!==	non identité
>	strictement supérieur
<	strictement inférieur
=>	supérieur ou égal à
=<	inférieur ou égal à

23

Les opérateurs de chaîne de caractères en JavaScript

+ concaténation

"texte 1" + "texte 2"

== test d'égalité (renvoie un booléen)

!= test de différence (renvoie un booléen)

<, >, =>, =< test de comparaison opéré sur la première lettre de chaque mot testé

24

Les opérateurs logiques en JavaScript

&&	ET	(AND)
||	OU	(OR)
!	NON	(NOT)

25

Les conditions

- Une condition est une instruction décrivant une alternative pour une structure.
  - SI...
  - ALORS…
  - SINON…
- **N.B.** Il peut exister autant de « SINON » que l'on veut.

26

Les conditions en JavaScript

```
if(test)
{
        instructions si le test est vrai
}
else if(test 2)
{
        instructions si le test est faux, mais le test 2 est vrai
}
else
{
        instructions si le test est faux
}
```

27

Les conditions en JavaScript

```
switch(test d'une variable)        ← nombre ou texte
{
        case valeur 1:             ← valeur testé par rapport à la variable
                instructions;
                break;             ← sortie du bloc switch
        case valeur 2:
                instructions;
                break;
        […]
        default:
                instructions;
                break;
}
```

28

Les conditions en JavaScript

N.B. Il est possible d'imbriquer des conditions.

29

Les boucles

- La boucle est une instruction permettant la répétition d'une structure.
  - RÉPÉTER nombre de fois
    - Instructions…

30

Les boucles

for(*initialisation* ; *condition d'arrêt* ; *incrémentation*)
{
instructions;
}

Exemple : for(let i = 0 ; i < 5 ; i++)

31

Les boucles conditionnelles

- La boucle conditionnelle est une instruction permettant de répéter une action si une condition est toujours remplie.
  - TANT QUE condition
    - FAIRE instructions…

32

Les boucles conditionnelles en JavaScript

```
let i = 0;
while(test)
{
        instructions;
        i++;
}
```

Cette boucle correspond au pseudo-code « tant que ». Le test définit la condition d'arrêt de la boucle.

33

Les boucles conditionnelles en JavaScript

```
let i = 0;
do
{
        instructions
        i++;
}
while(test);
```

Cette boucle exécute au moins une fois les instructions du bloc, même si la condition n'existe pas.

34

Les boucles en JavaScript

N.B. 1. **break** permet d'interrompre la boucle.

N.B. 2. **continue** permet de relancer une itération de la boucle en ignorant le reste des instructions à accomplir.

35

Les tableaux

- Les tableaux permettent d'ordonner et de structurer les variables en suivant des lignes et des colonnes.
- Les tableaux peuvent accueillir n'importe quelle variable, y compris d'autres tableaux.
- **N.B.** Les tableaux correspondent à la structure la plus employée dans un programme, mais il existe d'autres schémas permettant l'organisation des données : les listes, les piles, les files, les arbres, *etc*.

36

Les tableaux en JavaScript

Pour déclarer un tableau vide :

```
let table = [];
```

ou

```
let table = new Array();
```

ou

```
let table = new Array(nombre d'éléments);
```

Pour déclarer un tableau avec des éléments :

```
let table = [élément 0, élément 1, élément 2, …]
```

ou

```
let table = new Array(élément 0, élément 1, élément 2, …);
```


Les tableaux en JavaScript

Pour accéder à un élément du tableau :

```
table[0];
```

ou

```
table.at(0);
```

Pour modifier la valeur d'un élément du tableau :

```
table[0] = valeur;
```


Les tableaux en JavaScript

Pour parcourir un tableau :

```
for(let i = 0 ; i < table.length ; i++)
{
    console.log(table[i]);
}
```

N.B. length est une propriété d'instance donnant la longueur du tableau.

Les boucles de parcours en JavaScript

- Il existe deux boucles de parcours
  - for… in… → Parcours des numéros ordonnant les éléments
  - for… of…→ Parcours des éléments

Les fonctions

- Une fonction accomplit une tâche et renvoie une réponse.
  - FONCTION nom de la fonction, paramètres d'entrée de la fonction
    - Instructions…
- Exemple. Élever un nombre au carré
  - FONCTION mettreAuCarre, nombre d'entrée x
    - X = X^2

Les procédures

- Une procédure accomplit une tâche, mais, à la différence d'une fonction, elle ne renvoie aucune réponse.
  - FONCTION direBonjour, nom de l'utilisateur
    - Afficher "Bonjour + nom de l'utilisateur !"

Les fonctions ou procédures

- Pas de panique ! JavaScript et PHP ne distinguent pas les deux.
- Il existe de nombreuses fonctions natives :
  - fonctions mathématiques
  - fonctions de conversion des variables
  - fonctions sur les chaînes de caractères
  - fonctions sur les tableaux
  - *etc*.

Les fonctions en JavaScript

- Une **fonction** affecte un bloc d'instructions qui ne fait qu'un traitement.
  - Elle a un identificateur.
  - Elle peut avoir des paramètres.

```
function nomDeLaFonction(paramètre 1, paramètre 2, …)
{
    instructions utilisant les paramètres en variables locales
}
```


Les fonctions en JavaScript

```
function f(x)
{
    return x*x;
}
let carreDeDeux = f(2);
```

← *appel de la fonction f avec l'**argument** 2.*

Les fonctions en JavaScript

- Les fonctions sont **en principe** définies **au début** du programme.

N.B. 1. Lorsque JavaScript rencontre une fonction, il ne l'exécute pas directement ; il enregistre le bloc d'instructions sous le nom de la fonction.

N.B. 2. Une fonction peut utiliser les variables d'une portée globale.

N.B. 3. Il est possible de faire une expression de fonction (ou un fermeture), c'est-à-dire affecter une fonction à une variable.

```
let fonctionCarree = function (x) {return x*x;};
fonctionCarree(2);
```

N.B. 4. function (x) {return x*x;}; est appelé **fonction anonyme**.

Les fonctions fléchées en JavaScript

- La méthode la plus moderne d'écrire une fonction en JavaScript est d'utiliser une fonction fléchée.

```
let fonctionCarree = (x) => {return x*x;};
```


return en JavaScript

return est la commande qui permet de renvoyer une valeur une fonction que la fonction à terminer son exécution.

N.B. Si aucun return n'est placé en fin de fonction, la valeur de retour est par défaut undefined.

Les fonctions auto-invoquées en JavaScript

- Les fonctions auto-invoquées s'appellent d'elles-mêmes ou s'exécutent d'elles-mêmes. Elles sont souvent compilées avec les fonctions anonymes.

```
(function (x)
{
        x = prompt("Entrez un nombre : ");
        alert("Le carré de votre nombre est : " + x*x);
}
)();
```


L'objet Function en JavaScript

Toute fonction en JavaScript est un **objet** Function. Il est possible de créer une fonction en utilisant la notation objet.

```
let f = new Function("x", "return x*x;");
```


Les objets

- Un objet est un moyen de structurer les variables (ou données).
- Un objet contient deux groupes d'éléments :
  - des propriétés, des attributs (les variables de l'objet)
  - des méthodes (les fonctions de l'objet)
- Sauf exceptions, pour utiliser un objet, il faut le créer (en paramétrant ses propriétés) et accéder à ses méthodes.
- PHP et JavaScript sont des langages orientés objets. L'objectif de l'algorithmique est de comprendre les instructions élémentaires afin de pouvoir concevoir des objets.

Les objets en JavaScript

- La notion d'objet en JavaScript est très complexe, et parfois ambiguë.

```
function Image(source, titre, largeur, hauteur)
{
        this.source = source;
        this.titre = titre;
        this.largeur = largeur;
        this.hauteur = hauteur;
}
let photo = new Image("photo.jpg", "Ma photo", 80, 120);
```

N.B. this est utilisé pour renvoyer à Image.

Les objets en JavaScript

```
class Image
{
        constructor(source, titre, largeur, hauteur)
        {
                this.source = source;
                this.titre = titre;
                this.largeur = largeur;
                this.hauteur = hauteur;
        }
}
let photo = new Image("photo.jpg", "Ma photo", 80, 120);
```

N.B. Il s'agit d'une version plus moderne et plus proche des langages orientés objets à classes, mais ce n'est pas une classe. Le JavaScript reste un langage orienté objet à prototypes.

Les objets en JavaScript

```
class Image
{
        constructor(source, titre, largeur, hauteur)
        {
                this.source = source;
                this.titre = titre;
                this.largeur = largeur;
                this.hauteur = hauteur;
        }
}
let photo = new Image("photo.jpg", "Ma photo", 80, 120);
```

N.B. Il s'agit d'une version plus moderne et plus proche des langages orientés objets à classes, mais ce n'est pas une classe. Le JavaScript reste un langage orienté objet à prototypes.

Les objets en JavaScript

```
class Image
{
        constructor(source, titre, largeur, hauteur)
        {
                this.setSource(source);
                this.setTitre(titre);
                this.setLargeur(largeur);
                this.setHauteur(hauteur);
        }
        getSource()
        {
                return this.source:
        }
        setSource(source)
        {
                this.source = source;
        }
```


Les objets en JavaScript

```
getTitre()
{
        return this.titre:
}
setTitre(titre)
{
         this.titre = titre;
}
getLargeur()
{
        return this.largeur:
}
setLargeur(largeur)
{
         this.largeur = largueur;
}
```


Les objets en JavaScript

```
        getHauteur()
        {
                return this.hauteur:
        }
      setHauteur(hauteur)
        {
                 this.hauteur = hauteur;
        }
}
let photo = new Image("photo.jpg", "Ma photo", 80, 120);
```

N.B. On appelle les fonctions get… et set… : les *getter* et les *setter*. Ce sont les méthodes qui permettent de gérer les attributs de l'objet Image.

```
photo.getSource();                  photo.setSource("nouvelle-photo.jpg");
photo.getTitre();                   photo.setTitre("Ma nouvelle photo");
photo.getLargeur();                 photo.setLargeur(100);
photo.getHauteur();                 photo.setHauteur(160);
```


Les objets en JavaScript

Tout est objet en JavaScript. Une fois cette idée est bien comprise, il est possible d'approfondir les objets Function et Array que nous avons déjà utilisé.

Parmi les objets à connaître, il y a :

- Math
- RegExp
- Date
- String
- *etc.*

À vous de découvrir toutes les possibilité du JavaScript !