Les algorithmes

Les algorithmes c'est les fondations d'un programme, il s'agit d'une suite d'instructions logique que votre ordinateur doit suivre scrupuleusement afin d'obtenir une solution.

C'est un peu comme une recette de cuisine, nous allons réaliser des instructions dans l'ordre pour obtenir un bon plat.

Pour parvenir à la bonne solutions, il existe ce que l'on nomme des structures de contrôles :

- **Les conditions** : C'est la possibilité d'emprunter telle solution ou une autre en fonction de certaines condition. Ne pas afficher un article si l'utilisateur à moins de 18 ans.
- **Les boucles** : C'est la possibilité de répéter une solutions autant de fois qu'on le désire : en cuisine, pétrir la pates une centaines de fois.

Les conditions

En javascript, il est possible de créer une structure de contrôle `if / else` permettant de réaliser des instructions en fonction de certaines condition. Pour cela, on utilise le mot clef `if` suivie (entre parenthèse) d'une expression dite « booléenne » (soit vrai, soit faut). Il faut ensuite spécifier un block de code à réaliser lorsque la condition s'avère être remplie :

```
const age = 12

// Création d'une condition
if (age >= 18) {
  // Ici, tout ce qui se trouve dans ce block est ignoré
  // parce-que la condition n'est pas remplie
  console.log('Je suis majeur')
}

if (age > 10 && age < 17) {
  // Ici, tout ce qui se trouve dans le block est
  // exécuté ! Car la condition est remplie !
  console.log('je suis adolescent')
}
```

Souvent le `if` est associé au `else` (sinon) :

```
const age = 12

// Création d'une condition
if (age >= 18) {
  // Ici, tout ce qui se trouve dans ce block est ignoré
  // parce-que la condition n'est pas remplie
  console.log('Je suis majeur')
} else {
  // ici la condition est remplie donc le block de code
  // est exécuté
  console.log('Je suis mineur')
}
```

Nous avons la possibilité de réaliser un `else if` (sinon si) :

```
const age = 24

// Création d'une condition
if (age >= 50) {
  // Ici, tout ce qui se trouve dans ce block est ignoré
  // parce-que la condition n'est pas remplie
  console.log('Je suis senior')
} else if (age >= 18) {
  // ici c'est le bon block de code car la condition est remplie
  console.log('Je suis majeur')
} else {
  // ici le sinon n'est pas remplie, c'est toujours le premier qui gagne.
  console.log('Je suis mineur')
}
```

Les « switch »

C'est une autre forme de condition, mois évidente, plus rare en javascript, qui permet de réaliser du code, des instructions, en fonction d'une valeur d'une variable :

```
const classe = '6eme'

// Je demande à lancer des instructions en fonction de ce que
// contient la variable class
switch (classe) {
  // Dans le cas ou la variable contient 6eme :
  case '6eme':
    console.log('Je suis en 6eme, ro le débutant ...')
    // Le break permet de sortir du switch, si nous l'omettons
    // ça continue et du coup le cas par défaut sera aussi lancé !
    break

  case '5eme':
    console.log('Je suis en 5eme, ça ce complique')
    break

  case '4eme':
    console.log('Je suis en 4eme, ça devient sérieux')
    break

  default:
    console.log('me voici en 3eme, bientôt le lycée')
}
```

La condition ternaire

La condition ternaire, est une condition classique mais condensé sur une seule ligne et possédant une seule instruction dans les différents cas :

```
// Voici un code permettant de détecter si une personne
// est majeur en utilisant un boolean « isMayor »
let isMayor = false
const age = 24

if (age >= 18) {
  isMayor = true
} else {
  isMayor = false
}

console.log(isMayor) // true
```

```
// Voici le même code, mais utilisant une constante
// avec une ternaire
const age = 24
const isMayor = age >= 18 ? true : false

console.log(isMayor)
```

Anatomie d'une ternaire :

```
age >= 18 ? true : false
     |        |        |
     |        |        |
  condition   |        |
       Valeur si vrai  |
                Valeur si faux
```

Le `?` se traduis par `alors` et le `:` par `sinon`

Cette solution est concises mais hélas peut être illisible, à utiliser avec parcimonie.

Les boucles

Le principe de boucles c'est de pouvoir répéter une opération. Il éxiste 3 familles de boucles :

- Les boucles conditionnel (`while`)
- Les boucles de parcours (`for`)
- Les boucles fonctionnelle (inspiré des math, `map`, `filter`, `reduce`, `forEach` ...)

Ici, nous couvrirons que les 2 premières boucles, pour les boucles fonctionnelle il faut connaître les fonctions.

Les boucles conditionnel

Ce sont des instructions qui le lance autant de fois qu'une condition est vrai :

```
let compteur = 0

// On utilise un boucle conditionnelle « while » (tant que) :
while (compteur < 10) {
  console.log(`Le compteur est inférieur à 10 car égale à : ${compteur}`)
  compteur += 1
}
```

ATTENTION :

```
// Boucle infinie ... Elles sont très dangereuse !
while (true) {
  console.log('Je tourne en boucle ....')
}
```

> Il faut être vigilent aux boucles infinie lorsque l'on utilise `while`, cependant elles sont utilisé dans beaucoup de domaine ... Jeux Video

```
while (true) {
  console.log('Je tourne en boucle ....')
  // Nous pouvons manuellement quitter une boucle en utilisant break!
  break
}
```

Il existe aussi la possibilité de « sauter » une itération de boucle en utilisant e mot clef `continue` :

```
let compteur = 0

while (compteur < 50) {
  compteur += 1

  if (compteur % 2 === 0) {
    continue
  }

  console.log(`Compteur est impair : ${compteur}`)
}
```

```
let compteur = 0

while (compteur < 50) {
  compteur += 1

  if (compteur % 2 === 0) {
    continue
  } else {
    console.log(`Compteur est impair : ${compteur}`)
  }
}
```

Les boucles de parcours

Les boucles de parcours on la responsabilité de boucler sur tout les éléments d'une liste (array). Différemment du while, les boucles de parcours n'ont pas de conditions, elle se lance sur chaque élément d'une liste.

Pour faire une boucle de parcours on utilise 2 formes :

- `for` ... `in` : Boucle sur tout les positions (index) d'une liste (array)
- `for` ... `of` : Boucle sur toutes les valeurs d'une liste (array)

Exemple

```
const notes = [12, 8, 9, 18, 17]

// Boucle sur les positions du tableaux
for (let index in notes) {
  // Ici, index prendra à chaque tour de boucle
  // la position de l'élément du tableaux
  console.log(`La note n°${index} est ${notes[index]}`)
}

// Boucle sur les valeurs du tableaux
for (let note of notes) {
  // Ici, note prendra à chaque tour de boucle
  // la valeur de l'élément du tableaux
  console.log(`Note ${note}`)
}
```