XEBIA NATIS

Code

Laurence
XEBIA NATIS

XWIKI NATIS

Compétence demandée :
Appréhender les 5 concepts

XWIKI NATIS

1. Variables
2. Instructions de base
3. Blocs
4. Conditions
5. Boucles

XWNATIS

Processus habituel d'acquisition de compétences informatiques

1. Savoir écrire des algorithmes
2. Savoir traduire les algorithmes dans un langage
3. Savoir écrire et emboiter des fonctions
4. Savoir écrire des classes
5. Savoir concevoir une architecture

XWIKI NATIS

1. Variables

XWIKI NATIS

```
tab <- [23, 43, 32, 4, 3]
```

```
let tab;
tab = [23, 43, 32, 4, 3];
```

```
let tab = [23, 43, 32, 4, 3];
```

XNATIS

Définition ou affectation ?

XDNATIS

Définition : création de la variable

Affectation : attribution d'une valeur

XWIKI NATIS

2. Instructions de base

XDNATIS

Affectation

XEBIA NATIS

```
taille <- 34
```

```
toto <- [23, 34, 32, 3]
```

```
resultat <- 'Petit'
```

```
yop <- Vrai
```

XEBIANATIS

```
taille <- 34
```

```
let taille = 34;
```

```
toto <- [23, 34, 32, 3]
```

```
let toto = [23, 34, 32, 3];
```

```
resultat <- 'Petit'
```

```
let resultat = 'Petit';
```

```
yop <- Vrai
```

```
let yop = true;
```

XWIKI NATIS

```
pers <- {
    ‘age’ : 36,
    ‘prenom’ : ‘Michael’
}
```

```
let pers = {
    'age' : 36,
    'prenom' : 'Michael'
};
```

```
let pers = {
    age : 36,
    prenom : 'Michael'
};
```

XEBIANATIS

Structure de données	Actions possibles
Nombre	Addition Soustraction Division Multiplication
Chaîne de caractères	Concaténation
Boolean	Et Ou Non
Tableau	Adressage (position) Ajout Suppression
Dictionnaire	Adressage (clé) Ajout Suppression

```
3 + 4
10 - 23
20 / 10
5 * 4
```

```
'Hello' + ' ca va ?'
```

```
true && false
true || false
!true
```

```
tab[0]
tab.push(22)
tab.splice(0)
```

```
obj.age
obj.prenom = 'Michael'
delete obj.age
```

XEBIA NATIS

Structure de données	Actions possibles	Javascript
Nombre	Addition Soustraction Division Multiplication	+ - / *
Chaîne de caractères	Concaténation	+
Boolean	Et Ou Non	&& || !
Tableau	Adressage (position) Ajout Suppression	[position] push(element) splice(position)
Dictionnaire	Adressage (clé) Ajout Suppression	[clé] [clé] = element delete [clé]

XEBIA NATIS

Exercices de traduction.
Traduire les algorithmes en code JS :
1 & 2

XNATIS

Exercice 1 :

Echanger les valeurs de 2 variables R et Q

```
let R = 5
let Q = 34
```

Traduction

A = R (Création de la nouvelle variable à qui on affecte 5 (valeur de initiale de R)

R = Q (On affecte à R la valeur initiale de Q qui est 34)

Q = A (On affecte à Q la valeur initiale de R qui avait été transmise à la nouvelle variable A)

XEBIANATIS

Exercice 2 :

Echanger les valeurs de 3 variables R, Q et V. Dans V, mettre Q
Dans Q, mettre R et dans R, mettre V.

```
let R = 5
let Q = 34
let V = 45
```

Traduction

A = V (Création de la nouvelle variable A à qui on affecte 45 (valeur initiale de V)
V = Q (On affecte à V la valeur initiale de Q qui est 34)
Q = R (On affecte à Q la valeur initiale de R qui est 5)
R = A (On affecte à R la valeur initiale de V qui était 45 et qui avait été affecté à A)

XWIKI NATIS

3. Blocs

XWIKI NATIS

```
@DebutBloc
resultat <- 'Grand’
@FinBloc
```

```
{
    let resultat = 'Grand';
}
```

XEBIA NATIS

Les variables définies dans un bloc meurent à la fin du bloc

Portée (scope)

XWIKI NATIS

4. Conditions

XWIKI NATIS

```
taille <- 34

resultat <- 'Petit'
@Si taille >= 50
    @DebutBloc
    resultat <- 'Grand’
    @FinBloc
```

```
let taille = 34;

let resultat = 'Petit';
if (taille >= 50)
{
    resultat = 'Grand';
}
```

```
taille <- 34

resultat <- 'Petit'
@Si   @Non (taille < 50)
    @DebutBloc
    resultat <- 'Grand’
    @FinBloc
```

```
let taille = 34;

let resultat = 'Petit';
if (!(taille < 50))
{
    resultat = 'Grand';
}
```

XWNATIS

Opérateurs binaires sur les prédicats

ET = « et en même temps … »

OU = « ou soit … »

XWIKI NATIS

Opérateurs unaires sur les prédicats

NON = « ne pas … » ou
« contraire »

XNATIS

```
taille <- 34
forme <- 'Rectangle'

resultat <- 'Petit'
@Si taille >= 50 @Et forme = 'Rectangle’
    @DebutBloc
    resultat <- 'Grand’
    @FinBloc
```

```
let taille = 34;
let forme = 'Rectangle';

let resultat = 'Petit';
if (taille >= 50 && forme == 'Rectangle')
{
    resultat = 'Grand';
}
```

XEBIANATIS

```
taille <- 34
forme <- 'Rectangle'

resultat <- 'Petit'
@Si taille >= 50 @Ou forme = 'Rectangle’
    @DebutBloc
    resultat <- 'Grand’
    @FinBloc
```

```
let taille = 34;
let forme = 'Rectangle';

let resultat = 'Petit';
if (taille >= 50 || forme == 'Rectangle')
{
    resultat = 'Grand';
}
```

XWNATIS

```
taille <- 34
forme <- 'Rectangle'

@Si taille >= 50 @Ou forme = 'Rectangle’
    @DebutBloc
    resultat <- 'Grand’
    @FinBloc
@Sinon
    @DebutBloc
    resultat <- ’Petit’
    @FinBloc
```

```
let taille = 34;
let forme = 'Rectangle';

let resultat = 'Petit';
if (taille >= 50 || forme == 'Rectangle')
{
    resultat = 'Grand';
}
else
{
    resultat = 'Petit';
}
```

XNATIS

Table de vérité

Les tables de vérité présentent tous les résultats possibles d'une opération logique

a	b	a ET b
Faux	Faux	Faux
Faux	Vrai	Faux
Vrai	Faux	Faux
Vrai	Vrai	Vrai

a	b	a OU b
Faux	Faux	Faux
Faux	Vrai	Vrai
Vrai	Faux	Vrai
Vrai	Vrai	Vrai

XEBIA NATIS

Exercices de traduction.
Traduire les algorithmes en code JS :
3, 4 & 5

XDNATIS

Exercice 3 :

Vérifier si un utilisateur est majeur grâce à son âge. Mettre une variable « resultat » à Vrai si cela est le cas, sinon mettez la à Faux.

Instruction : Age <- 25

Traduction

```
let age = 25;

if (age >= 18)
{
   resultat = true;
}
else
{
   resultat = false;
}
```

Traduction

```
let age = 25;
let resultat = false;

If (age >= 18)
{
   resultat =  true;
}
```

XDNATIS

Exercice 4 :

Vérifier si un utilisateur est mineur grâce à son âge. Mettre une variable « resultat » à Vrai si cela est le cas, sinon mettez la à Faux.

Instruction : Age <- 12

Traduction

```
let age = 12;
let resultat = false;

if (age < 18)          (Ici l'âge est strictement inférieur à 18)
{
    resultat = true;
}
```

XEBNATIS

Exercice 5 :

Echanger/Inverser les 2 premiers éléments d'un tableau.

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction

```
let tab = [23, 4, 2, 543, 34, 12, 4];
let A = tab[0];
tab[0] = tab[1];
tab[1] = A;
```

On obtient : tab = [4, 23, 2, 543, 34, 12, 4]

XWIKI NATIS

5. Boucles

XDNATIS

Les boucles

Les boucles permettent de répéter un bloc d'instructions

XWIKI NATIS

Il y a 3 types de boucles pour répéter un bloc

1. @PourChaque
2. @Pour @De @A
3. @TantQue ou Boucle + @Stop

XNATIS

1. Il faut s'arrêter à la fin du tableau

```
tab <- [23, 43, 32, 4, 3]

@PourChaque element @Dans tab
    @DebutBloc
    Afficher element
    @FinBloc
```

```
let tab = [23, 43, 32, 4, 3];

for (const element of tab)
{
    console.log(element);
}
```

XNATIS

2. Il faut s'arrêter avec un nombre maximal

```
tab <- [23, 43, 32, 4, 3]

@Pour i @De 0 @A 4
    @DebutBloc
    Afficher tab[i]
    @FinBloc
```

```
let tab = [23, 43, 32, 4, 3];

for (let i = 0; i < 4; ++i)
{
    console.log(tab[i]);
}
```

XKNATIS

3. Il faut s'arrêter avec une condition

```
tab <- [23, 43, 32, 4, 3]

position <- 0
@TantQue tab[position] < 30
    @DebutBloc
    position <- position + 1
    @FinBloc
Afficher position
```

```
let tab = [23, 43, 32, 4, 3];

let position = 0;
while (tab[position] < 30)
{
    position = position + 1;
}
console.log(position);
```

XEBIA NATIS

3. Il faut s'arrêter avec une condition

```
tab <- [23, 43, 32, 4, 3]

position <- 0
@PourChaque element @Dans tab
    @DebutBloc
    @Si element >= 30
        @DebutBloc
        Afficher position
        @Stop
        @FinBloc
    position <- position + 1
    @FinBloc
```

```
let tab = [23, 43, 32, 4, 3];

let position = 0;
for (const element of tab)
{
    if (element >= 30)
    {
        console.log(position);
        break;
    }
    position = position + 1;
}
```

XEBNATIS

Exercices de traduction.
Traduire les algorithmes en code JS :
6, 7, 8, 9, 10, 11 & 12

XEBNATIS

Exercice 6 :

Compter le nombre d'éléments dans un tableau. Mettre le compte dans une variable "**resultat**".

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction

```
let tab = [23, 4, 2, 543, 34, 12, 4];
let resultat = 0;                          (On définit une variable "résultat" avec une valeur 0)

for (const element of tab)                 (Pour chaque élément du tableau parcouru)
{
    resultat = resultat++ (ou resultat + 1) (On ajoute 1 à la valeur du « resultat » donc on incrémente)
}
```

XEBNATIS

Exercice 7.1 :

Faire la somme des éléments d'un tableau.

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction

```
let tab = [23, 4, 2, 543, 34, 12, 4];
let resultat = 0;                          (On définit une variable "résultat" avec une valeur 0)

for (const element of tab)                 (Pour chaque élément du tableau parcouru)
{
    resultat = resultat + element;         (On ajoute chaque chiffre à « resultat » donc on fait une addition)
}
```

XNATIS

Exercice 7.2 :

Faire la somme des éléments d'un tableau.

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction

```
let tab = [23, 4, 2, 543, 34, 12, 4];
let compte = 0;                    (On définit une variable "résultat" avec une valeur 0)

for (const element of tab)         (Pour chaque élément du tableau parcouru)
{
    compte = compte + 1;           (On ajoute 1 à la valeur du "compte" donc incrémentation)
}

let resultat = 0;                  (On définit une variable "résultat" avec une valeur 0)
for (i = 0; i < compte; i++)       (Pour chaque élément parcouru dont la position est inférieure à la valeur de « compte »)
{
   resultat = resultat + tab[i] ; ("résultat" est l'addition de chaque élement sans dépasser le compte)
}
```

XDNATIS

Exercice 8 :

Trouver l'élément maximum d'un tableau. Mettre le nombre maximum dans une variable "**resultat**".

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction

```
let tab = [23, 4, 2, 543, 34, 12, 4];
let resultat = 0;

for (const element of tab)
    {
        if (element > resultat)
            {
                resultat = element;
            }
    }
```

(On définit une variable "résultat" avec une valeur 0)

(Pour chaque élément du tableau parcouru)

(On vérifie si l'élément parcouru est supérieur à « resultat »)

(Puis à chaque arrêt, on fait une comparaison avec l'élément précédent)

(L'élément maximum devient le « resultat »)

XDNATIS

Exercice 9 :

Trouver l'élément minimum d'un tableau. Mettre le nombre maximum dans une variable "**resultat**".

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction

```
let tab = [23, 4, 2, 543, 34, 12, 4];
let resultat = tab[0];
```

(On définit une variable "résultat" et on part de l'élément du tableau qui a l'indice 0 donc le 1er élément du tableau)

```
for (const element of tab)
```

(Pour chaque élément du tableau parcouru)

```
    {
        if (element < resultat)
                {
                    resultat = element;
                }
    }
```

(On vérifie si l'élément parcouru est inférieur à « resultat »)

(Puis à chaque arrêt, on fait une comparaison avec l'élément précédent)

(L'élément minimum devient le « resultat »)

XDNATIS

Exercice 10.1 :

Trouver le premier élément supérieur à 500. S'il n'y en a pas,
le résultat doit être 0.

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction (Méthode 1 : for… of)

```
let tab = [23, 4, 2, 543, 34, 12, 4];
let resultat = 0;                          (On définit une variable "résultat" avec une valeur 0)

for (const element of tab)                 (Pour chaque élément du tableau parcouru)
   {
            if (element > 500)             (On vérifie si l'élément parcouru est supérieur à 500)
              {
                 resultat = element;       (Si la condition est vérifiée, on l'attribue à « resultat »)
                 break;                    (On termine la boucle avec l'instruction « Break »)
              }
   }
```

XNATIS

Exercice 10.2 :

Trouver le premier élément supérieur à 500. S'il n'y en a pas, le resultat doit être 0.

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction (Méthode 2 : while)

```
let i = 0;
let indiceMax = tab.length -1;

while (tab[i] < 500 && i <= indiceMax) (Tant que l'élément à la position actuelle est strictement inférieure à 500 et que i ne dépasse pas la position maximale autorisée c'est que l'on a pas encore trouvé ce que l'on cherche, on peut avancer)
{
    i = i + 1;                    (On continue d'avancer)
    if (i > indiceMax)            (Si la position actuelle est strictement supérieure à la position maximale autorisée)
      {
        break;                    (On termine la boucle avec l'instruction « Break »)
      }
}
```

XNATIS

Exercice 10.3 :

Trouver le premier élément supérieur à 500. S'il n'y en a pas, le résultat doit être 0.

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction (Méthode 3 : For)

```
let resultat = 0;                    (On définit une variable "résultat" avec une valeur 0)
If (tab[i] >= 500)                   (Si l'on trouve ce que l'on cherche)
        {
            resultat = tab[i];       (On affecte la valeur trouvée à « resultat »)
        }
```

XDNATIS

Exercice 11 :

Copier le tableau "tab" dans un autre tableau intitulé "autre".

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction

Méthode 1

```
let autre = [];
for (const element of tab)
        {
            autre.push(element);
        }
```

(On définit une variable « autre" avec tableau vide)

(On ajoute « push » chaque élément un par un à ce nouveau tableau « autre »)

Méthode 2

```
let tab = [23, 4, 2, 543, 34, 12, 4];
let autre = tab
```

(On créé un nouveau tableau « autre » et on lui affecte le premier tableau « tab »)

XDNATIS

Exercice 12 :

Copier les premiers éléments d'un tableau dont la somme fait au moins 500 (dans un autre tableau intitulé « autre ».

Instruction : tab <- [23, 4, 2, 543, 34, 12, 4]

Traduction

```
let autre = [];                    (On définit une variable « autre" avec tableau vide)
let somme = 0;                     (On définit une variable « somme" avec une variable 0)
for (const element of tab)         (Pour chaque élément du tableau parcouru)
    {
        somme = somme + element; (La somme sera calculée avec chaque élément afin d'atteindre au moins 500)
        autre.push(element);           (Ces éléments seront copiés un par un dans le tableau vide)
        if (somme >= 500)              (Si la somme est supérieure à 500)
            {
                break;                 (On termine la boucle avec l'instruction « Break »)
            }
    }
```