

Rappels

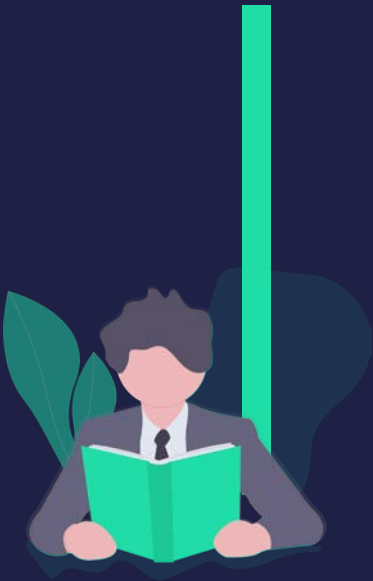
Michael
X  NATIS



Compétence demandée :
Appréhender les 5 concepts

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

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



1. Variables

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

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

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


Définition ou affectation ?

Définition : création de la variable
Affectation : attribution d'une valeur

Définition : création de la variable
Affectation : attribution d'une valeur



2. Instructions de base

Affectation

```
taille <- 34
```

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

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

```
yop <- Vrai
```

```
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;
```


①

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

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

②

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

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

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]



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

3. Blocs

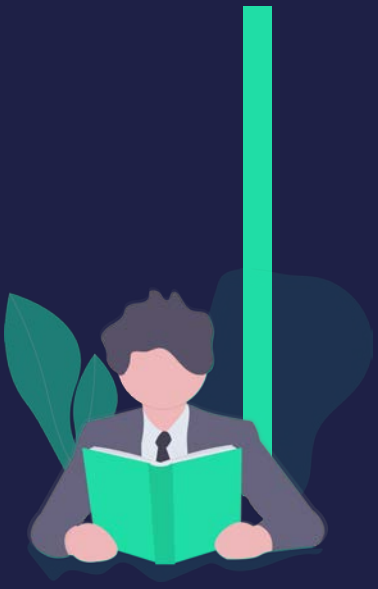
```
@DebutBloc  
resultat <- 'Grand'  
@FinBloc
```

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


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

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

Portée (scope)



4. Conditions

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

```
taille <- 34

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

```
let taille = 34;
let resultat = 'Petit';
if (taille >= 50)
{
  resultat = 'Grand';
}
```

prédicat

condition

```
let taille = 34;

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

Opérateurs binaires sur les prédicats

ET = « et en même temps ... »

OU = « ou soit ... »

Opérateurs unaires sur les prédicats

NON = « ne pas ... » ou
« contraire »


```
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';
}
```

```
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';
}
```

```
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';
}
```


5. Boucles

Les boucles

Les boucles permettent de **répéter un bloc d'instructions**

Il y a 3 types de boucles pour
répéter un bloc

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

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);
}
```


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]);  
}
```

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);
```

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;
}
```

