

# Découvrir PHP

Michael  
X  NATIS



**Compétence demandée :**  
**Appréhender les 6 concepts**

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



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

















# 1. Variables

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

```
$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

```
let taille = 34;
```

```
$taille = 34;
```

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

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

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

```
$resultat = 'Petit';
```

```
let yop = true;
```

```
$yop = true;
```

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

```
$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)	[position]
	Ajout	push(element)
	Suppression	splice(position)
Dictionnaire	Adressage (clé)	[clé]
	Ajout	[clé] = element
	Suppression	delete [clé]



Structure de données	JS
Nombre	$3 + 4$ $10 - 23$ $20 / 10$ $5 * 4$
Chaîne de caractères	'Hello' + ' ca va ?'
Boolean	true && false true    false !true
Tableau	tab[0] tab.push(22)
Dictionnaire	obj.age obj.prenom = 'Michael'

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

'Hello' . ' ca va ?'

true && false  
true || false  
!true

\$tab[0]  
\$tab[] = 22

\$obj['age']  
\$obj['prenom'] = 'Michael'







## 3. Blocs

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

```
{  
  $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



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

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

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

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

```
$taille = 34;
```

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

```
$taille = 34;
```

```
$resultat = 'Petit';  
if (!($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 »

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

```
$taille = 34;  
$forme = 'Rectangle';  
  
$resultat = 'Petit';  
if ($taille >= 50 && $forme == 'Rectangle')  
{  
  $resultat = 'Grand';  
}
```

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

```
$taille = 34;  
$forme = 'Rectangle';  
  
$resultat = 'Petit';  
if ($taille >= 50 || $forme == 'Rectangle')  
{  
  $resultat = 'Grand';  
}
```

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

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

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

$resultat = 'Petit';
if ($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

```
let tab = [23, 43, 32, 4, 3];  
  
for (const element of tab)  
{  
    console.log(element);  
}
```

```
$tab = [23, 43, 32, 4, 3];  
  
foreach ($tab as $element)  
{  
    var_dump($element);  
}
```

## 2. Il faut s'arrêter avec un nombre maximal

```
let tab = [23, 43, 32, 4, 3];  
  
for (let i = 0; i < 4; ++i)  
{  
  console.log(tab[i]);  
}
```

```
$tab = [23, 43, 32, 4, 3];  
  
for ($i = 0; $i < 4; ++$i)  
{  
  var_dump($tab[$i]);  
}
```

### 3. Il faut s'arrêter avec une condition

```
let tab = [23, 43, 32, 4, 3];  
  
let position = 0;  
while (tab[position] < 30)  
{  
    position = position + 1;  
}  
console.log(position);
```

```
$tab = [23, 43, 32, 4, 3];  
  
$position = 0;  
while ($tab[$position] < 30)  
{  
    $position = $position + 1;  
}  
var_dump($position);
```

### 3. Il faut s'arrêter avec une condition

```
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;
}
```

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

$position = 0;
foreach ($tab as $element)
{
    if ($element >= 30)
    {
        var_dump($position);
        break;
    }
    $position = $position + 1;
}
```





## 6. Fonctions

```
function additionner(a, b)
{
    return a + b;
}
```

```
function additionner(int $a, int $b): int
{
    return $a + $b;
}
```

<https://www.php.net/manual/fr/language.types.declarations.php>