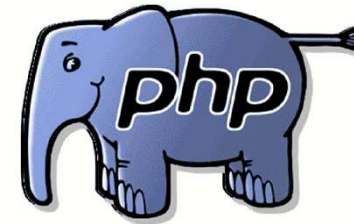


Développeur Web et Web Mobile

Le langage



Partie 1 : Les bases de PHP

PLAN DU COURS

Généralités

Présentation	6
Caractéristiques	7
Historique	8
Principe	9
Passage du html au PHP	10
Syntaxe générale	11

Les variables

Introduction	12
Syntaxe	13
Variable dynamique	14-15

PLAN DU COURS

suite

Les types de données

Introduction	16
les booléens	17
Les entiers	18
Les réels	19
les chaînes de caractères	20-23
les tableaux	24-26
les objets	27

Les opérateurs

Arithmétiques	28
Chaînes de caractères	29
D'assignation	30
De comparaison	31
Logiques	32
De test	33

PLAN DU COURS

suite

Les structures de traitement

Les alternatives	
si ... alors ... sinon	34
suivant ... décider entre	35
Les répétitives	
tantque ... faire	36
pour ... de ... a ... faire	37
pour ... chaque	38

Les fonctions

Syntaxe	39
Portée des variables	40-41
Passage de paramètres	42

PLAN DU COURS suite

La console

Directe (echo)
Formatée (printf)

45
46-47

Les règles de codage

Présentation du code
Les identifiants

Les constantes

45
46-47

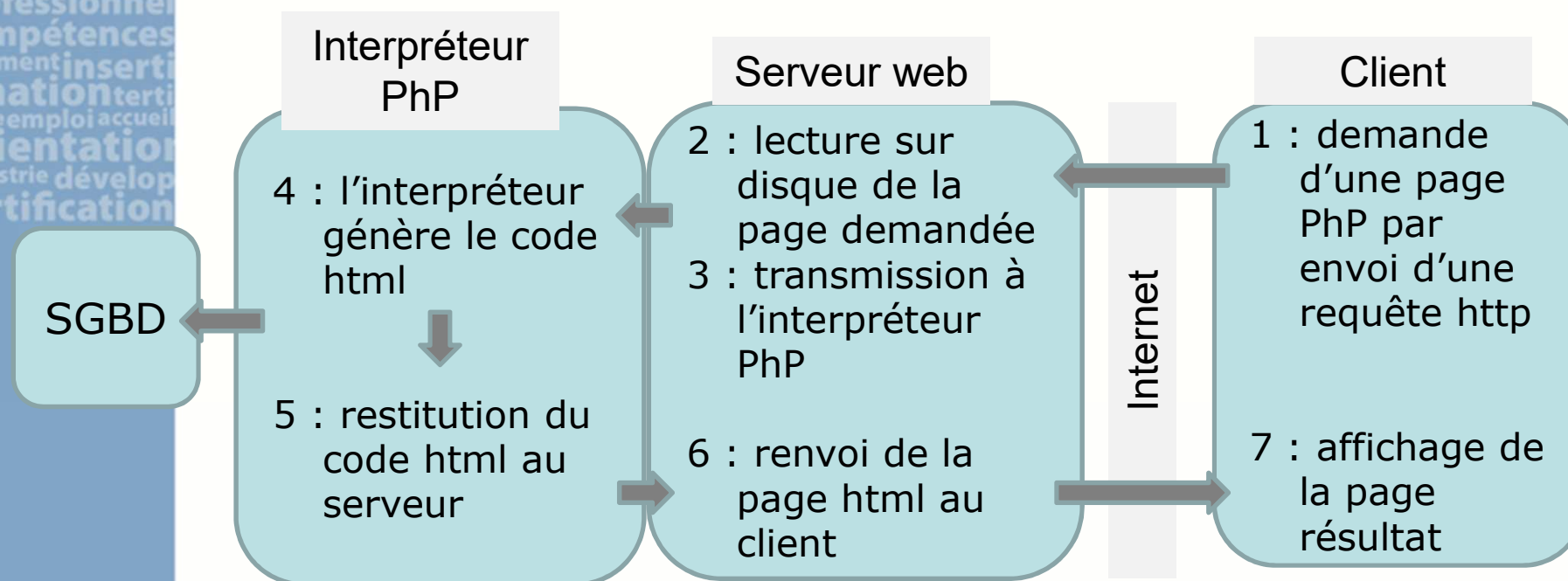
46-47

- Le « PHP: Hypertext Preprocessor », est un langage de script libre.
- Utilisé principalement pour créer des pages web dynamiques via un serveur http.
- Peut être utilisé en ligne de commande.
- C'est un langage interprété même si des solutions existent pour gagner en rapidité.
- Il a aujourd'hui les fonctionnalités complètes d'un langage orienté objet.

- Simplicité d'écriture de scripts
- Possibilité d'inclure le script PHP au sein d'une page HTML
- Simplicité d'interfaçage avec des bases de données (de nombreux SGBD sont supportés, mais le plus utilisé avec ce langage est MySQL, un SGBD gratuit sur les plates-formes Linux et Windows).
- Intégration au sein de nombreux serveurs web (Apache, Microsoft IIS, ...)
- Gratuité et la disponibilité du code source

- Bibliothèque écrite en PERL en 1994 par Rasmus Lerdorf pour son site web :
Il voulait conserver une trace des personnes qui venaient consulter son CV en ligne.
- Réécriture en C en 1995 et publié sous le nom de "Personal Home Page Tools".
- En 1997 deux étudiants Andi Gutmans et Zeev Suraski redéveloppent le cœur et la version 3 voit le jour sous le nom de Php: Hypertext Processor
- La version 4 suit avec un nouveau moteur appelé « Zend Engine » (**Z**eev **A**ndi)
- En 2002 c'est la version 5 avec Zend Engine 2 et un véritable modèle objet
- La dernière version est la 5.4.3 sortie en mai 2012
- Une version 6 est en cours

- Un script PHP est un simple fichier texte contenant des instructions incluses (encapsulées) dans un code HTML à l'aide de balises spéciales.



GENERALITES

passage du html au PHP

- Utilisation de balises spécifiques

<?php ?>

- Autres notations :

1. <? ?> (notation raccourcie)

2. <% %> (notation ASP)

Elles sont déconseillées car peuvent être désactivées dans la configuration du serveur

Interdit depuis PHP6

3. <script language="php"> </script>

Peu courante

- En ces 2 balises, autant de lignes php

GENERALITES syntaxe générale

<?php

echo "Coucou"; // commentaire 1 ligne

Séparation des
instructions

Commentaires

//
/* */

/* commentaire sur plusieurs lignes,
voilà la deuxième ligne*/

echo "Ceci est encore un test";

?>



implique



PHP différencie
MAJUSCULES et minuscules

LES VARIABLES

introduction

- Mémorise une information qui sera manipulée par le programme.
- PHP : langage faiblement typé
  pas de déclaration de type
- De nombreuses fonctions existent pour :
 connaître son type, savoir si elle est vide, ...
- Possibilité de modifier le type
  « transtypage »

Exemple :

```
$varEntier = 10;
```

```
$varChaine = (string)$varEntier;
```

- Une variable peut être : locale ou globale
  voir le chapitre sur les fonctions

23/11/2021

LES VARIABLES

syntaxe

Commence obligatoirement par
« \$ »

```
$maChaine = "Coucou";  
$monCompteur = 1;
```

IDENTIFICATEUR

- Préférer la notation
« **Camel Case** »
- Tout caractère
« **alphanumérique** »
- Pas de chiffre en
premier caractère
- Pas d'espace
- Doit être
« **auto informant** »



\$maChaine



\$MACHAINE

LES VARIABLES dynamiques

- Ce sont des variables dont le nom dépend d'autres variables.
- Ce nom est construit dynamiquement pendant exécution du programme.
- Utilisation du double « \$ » pour la syntaxe.

Exemple :

```
$mess = "variable dynamique";
$varDyn = "mess" ;
echo $$varDyn; //affiche "variable
                dynamique"
```

LES VARIABLES dynamiques (suite)

- Le recours aux variables dynamiques n'est généralement pas nécessaire (remplacées par les tableaux).
- Il est parfois utile d'avoir recours aux accolades.

Exemple :

```
$deuxieme = "contenu de $deuxieme";
$troisieme = "contenu de $troisieme";
$varDyn = "deux";
echo ${$varDyn}ieme;

//affiche "contenu de $deuxieme"

$varDyn = "trois";
echo ${$varDyn}ieme;

//affiche "contenu de $troisieme"
```

LES TYPES

Introduction

booléen (*boolean*)

nombre entier (*integer*)

nombre à virgule flottante (*float*)

chaîne de caractères (*string*)

types
scalaires

8 types de
données

types
composés

tableau (*array*)

objet (*object*)

types
spéciaux

ressource (*ressource*)

valeur nulle (*NULL*)

- type le plus simple
- 2 valeurs : **TRUE** ou **FALSE**

Exemple :

`$bool = TRUE;` // donne la valeur TRUE à \$bool

- Lors de la conversion de type :

Évalués à FALSE	Évalués à TRUE
Entier 0 ou réel 0.0	Entier ou réel différents de 0
La chaîne vide ""	Une chaîne non vide
La valeur NULL	

- Type
« **int** »

ou

- « **integer** »

de -2 147 483 648
à 2 147 483 647

- exprimés dans différentes bases :

entier

\$varEnt = 6837;

hexadécimale

\$varHex = 0x1AB5;

octale

\$varOct = 015265;

(\$varEnt, \$varHex et
\$varOct exprime la
même valeur)

- Type « **double** » ou « real » ou « float »

➤ dépendent de la plateforme
➤ généralement :
de $-1.8 \cdot 10^{-308}$ à $1.8 \cdot 10^{308}$

- Peuvent s'exprimer :

normale

\$vReal = 3.14159;

scientifique

\$vReal = 2.5e12;

Remarque : un calcul peut donner un résultat indéfini qui sera repéré par la constante « **NaN** » (Not a Number)

LES TYPES

Les chaînes de caractères

- Type « **string** »

Série de caractères
codés sur 1 octet

"cours de PhP"

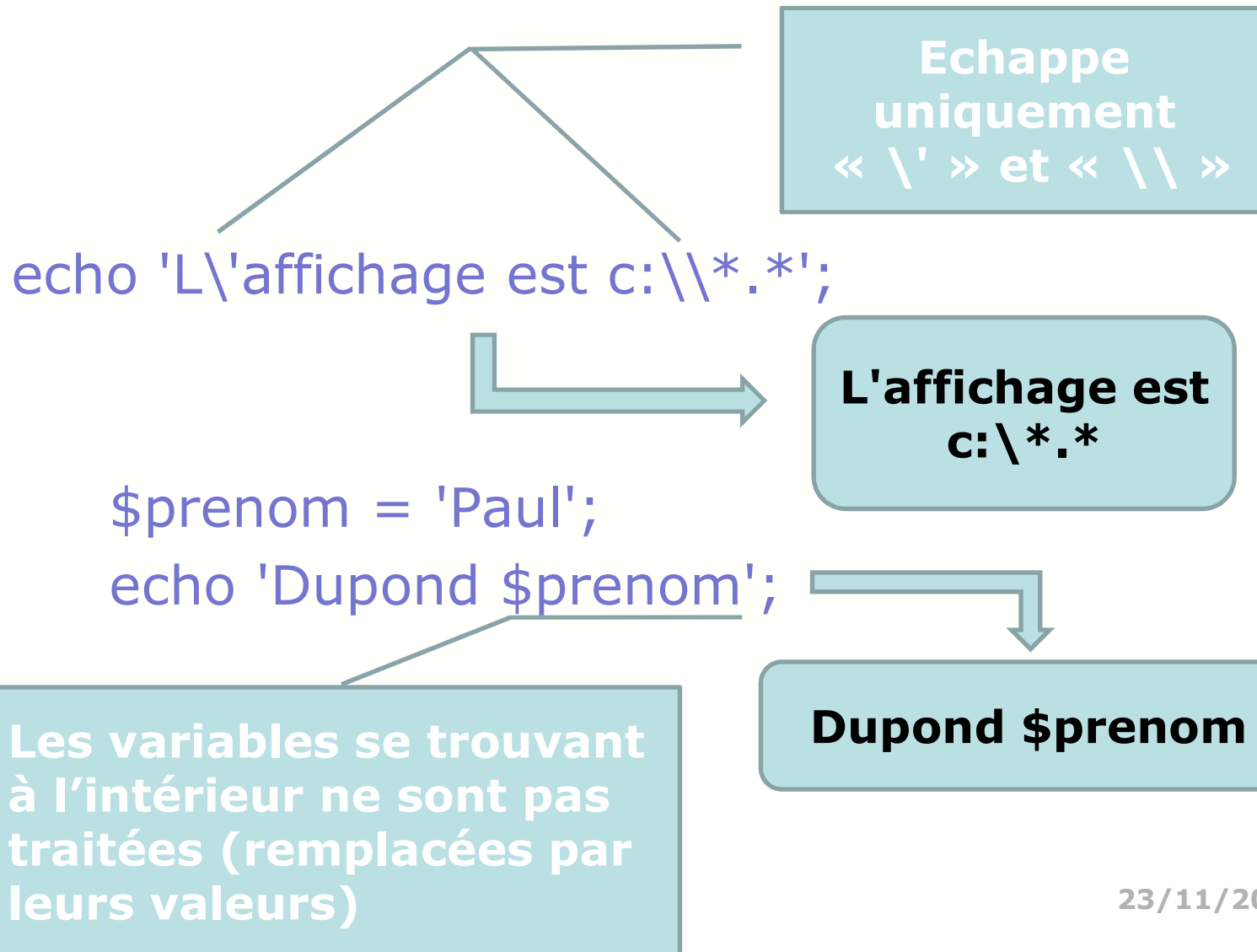
"cours de \"PhP\""

Echappement des
caractères spéciaux
en les préfixant par
« \ »

- Chaîne littérale : plusieurs syntaxes :
 - Entourée de guillemets simple « ' »
 - Entourée de guillemets double « " »
 - Syntaxe « heredoc »
 - Syntaxe « nowdoc »

LES TYPES

Les chaînes entre guillemets simples



Les chaînes entre guillemets doubles

- Echappe beaucoup de caractères spéciaux

<code>\n</code>	Fin de ligne
<code>\r</code>	Retour à la ligne
<code>\t</code>	Tabulation horizontale
<code>\\</code>	Antislash
<code>\\$</code>	Signe dollar
<code>\"</code>	Guillemet double

```
$prenom = "Paul";  
echo "Dupond $prenom";
```

Les variables se trouvant à l'intérieur sont traitées

Dupond Paul

LES TYPES

Les chaînes syntaxe heredoc

```
$prenom = <<<EOT
Paul
EOT;
```

Débute par « <<< »
suivi d'un **identifiant**
puis une **nouvelle ligne**

Se termine par l'identifiant

```
echo "Dupond $prenom";
```

Dupond Paul

- Se comporte comme une chaîne entourée de guillemets doubles
- La syntaxe « nowdoc » est identique mais se comporte comme pour les simples guillemets. L'identificateur est à mettre entre guillemets simples.

- Liste d'éléments pas obligatoirement du même type.

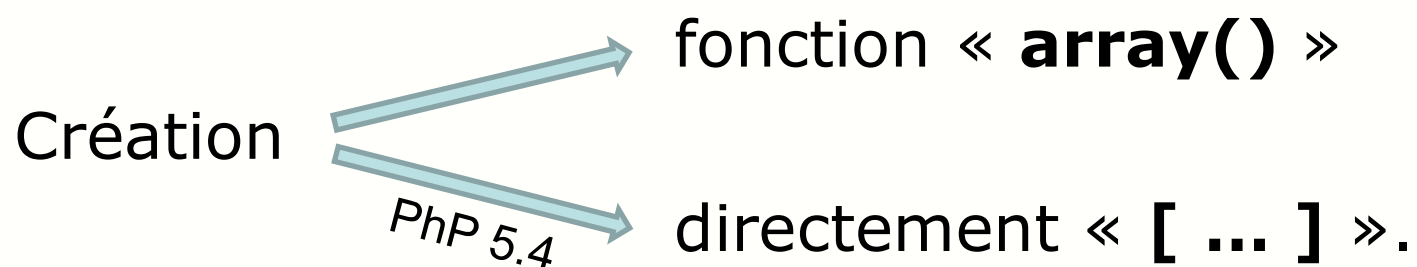


tableau indexé

suite de
« **valeurs** »
séparées par des
virgules

tableau associatif

suite de couples
« **clé** » => « **valeur** »
séparés par des virgules

- Il est possible de mélanger tableau indexé et associatif.

- En fait PHP ne distingue pas les tableaux indicés et associatifs.
- Si la clé est omise, elle est considérée comme numérique et s'incrémente de 1 à partir de la dernière valeur.
- Les clés peuvent être soumises à un transtypage vers le type entier.

"1"	->	1
"1.5"	->	1
1.5	->	1
true	->	1

Les tableaux: exemples

```
1 : $jourOuvre = array( "Lundi",
                        "Mardi",
                        "Mercredi",
                        "Jeudi",
                        "Vendredi");
```

```
echo $jourOuvre[2];
```

Mercredi



```
2 : $dateNaiss = array(17, "juin", 1998);
   $dateNaiss = [17, "juin", 1998];
   echo $dateNaiss[1];
```

juin



```
3 : $dateNaiss = array( "jour" => 17,
                        "mois" => "juin",
                        "annee" => 1998);
```

```
echo $dateNaiss["mois"];
```

juin

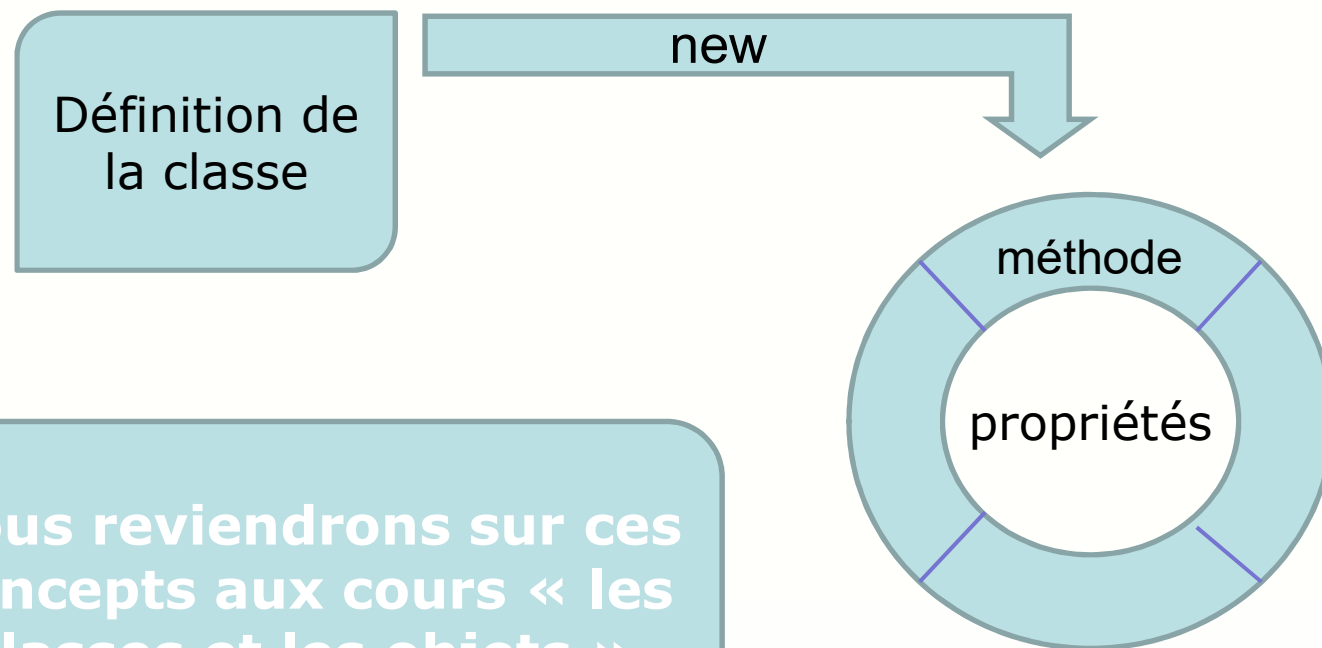


2 / 11 / 2021

LES TYPES

Les objets

- Un objet possède des valeurs par ses propriétés et des fonctions par ses méthodes.



Nous reviendrons sur ces concepts aux cours « les classes et les objets »

LES OPERATEURS arithmétiques

■ Opérateurs élémentaires

$-\$a$	Négation	Opposé de $\$a$
$\$a + \b	Addition	Somme de $\$a$ et $\$b$
$\$a - \b	Soustraction	Différence de $\$a$ et $\$b$
$\$a * \b	Multiplication	Produit de $\$a$ et $\$b$
$\$a / \b	Division	Quotient de $\$a$ et $\$b$
$\$a \% \b	modulo	Reste de la division de $\$a$ par $\$b$

■ Opérateurs d'in(dé)crémentement

$++\$a$	Pré-incrémentation	$\$a = \$a + 1$ puis retourne $\$a$
$\$a++$	Post-incrémentation	retourne $\$a$ puis $\$a = \$a + 1$
$--\$a$	Pré-décrémentation	$\$a = \$a - 1$ puis retourne $\$a$
$\$a--$	Post-décrémentation	retourne $\$a$ puis $\$a = \$a - 1$

LES OPERATEURS chaînes de caractères

- Opérateur de concaténation « . »

```
$debutChaine = "les opérateurs";  
$finChaine = " de chaines";  
$chaine = $debutChaine . $finChaine;  
echo $chaine;
```



les opérateurs de chaines

- Préférer la concaténation avec les chaines entre guillemets simples plutôt que des variables à l'intérieur des chaines entre guillemets doubles.

```
$finChaine = ' de chaines';  
$chaine = "les opérateurs $finChaine";  
$chaine = 'les opérateurs' . $finChaine;
```



LES OPERATEURS d'assignation

Assignment simple Utilisation du caractère

<< = >>

`$var = 56; // assigne 56 à la variable $var`

Assignment composée Opérateur suivi du signe d'assignation Fonctionne avec tout opérateur

`$var += 4; // $var prend la valeur 60`
`$chaine = "les opérateurs";`
`$chaine .= " de chaines";`
`echo $chaine;`

LES OPERATEURS de comparaison

■ Comparaison entre deux variables

<code>\$a == \$b</code>	égal	TRUE après transtypage
<code>\$a === \$b</code>	identique	TRUE si \$a même type que \$b
<code>\$a != \$b</code>	différent	TRUE après transtypage
<code>\$a <> \$b</code>	différent	TRUE après transtypage
<code>\$a !== \$b</code>	différent	TRUE si \$a différent de \$b ou si \$a pas même type que \$b
<code>\$a < \$b</code>	plus petit	TRUE si \$a strictement plus petit que \$b
<code>\$a <= \$b</code>	plus petit ou égal	TRUE si \$a plus petit ou égal à \$b
<code>\$a > \$b</code>	plus grand	TRUE si \$a strictement plus grand que \$b
<code>\$a >= \$b</code>	plus grand ou égal	TRUE si \$a plus grand ou égal à \$b

LES OPERATEURS logiques

■ Opérations entre deux variables

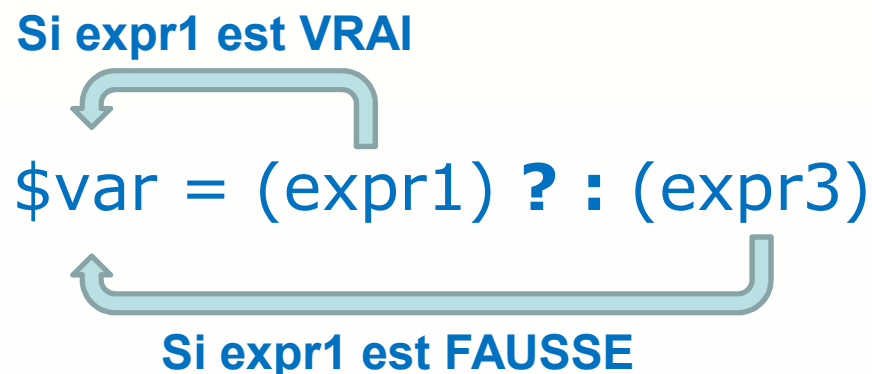
<code>\$a and \$b</code>	ET	TRUE si \$a ET \$b valent TRUE
<code>\$a or \$b</code>	OU	TRUE si \$a OU \$b valent TRUE
<code>\$a xor \$b</code>	OU exclusif	TRUE si \$a OU \$b valent TRUE mais pas les deux
<code>! \$a</code>	NON	TRUE si \$a n'est pas TRUE
<code>\$a && \$b</code>	ET	TRUE si \$a ET \$b valent TRUE
<code>\$a \$b</code>	OU	TRUE si \$a OU \$b valent TRUE

LES OPERATEURS de test (ternaire)

- Retourne la **valeur** d'une expression en fonction d'une condition.



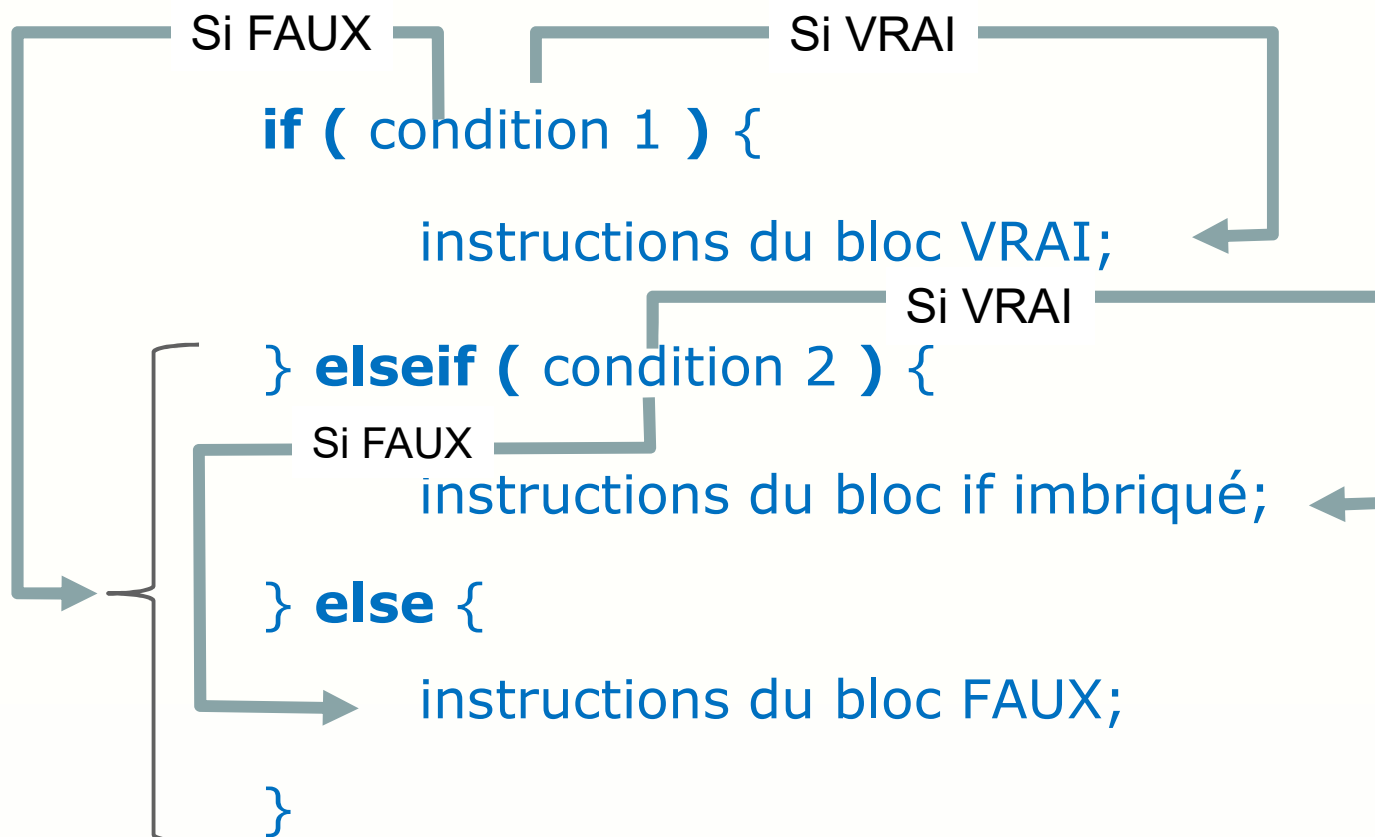
- Expr2 peut être omise :



LES STRUCTURES DE TRAITEMENT

Si ... Alors ... Sinon ...

- Mots clés : **if, else, elseif**



- Les conditions peuvent être complexes

LES STRUCTURES DE TRAITEMENT

Suivant ... Décider ... Entre ...

- Mot clé : **switch**

switch (variable) {

case valeur1 :

instructions cas valeur1;

break;

case valeur2 :

instructions cas valeur2;

break;

.....

case default :

instructions cas valeur1;

}

**Variable numérique
entière ou chaîne de
caractères**

LES STRUCTURES DE TRAITEMENT

Tant Que

- Mot clé : **while**
- La syntaxe générale :

```
while ( condition ) {  
    instructions condition VRAI;  
}
```

- Il existe un **do ... while** :

```
do {  
    instructions condition VRAI;  
} while ( condition )
```

LES STRUCTURES DE TRAITEMENT

Pour ... Faire

- Mots clés : **for**

exécutée 1 et 1 seule
fois avant toute itération
sert à initialiser la
variable de boucle

condition évaluée
à chaque itération
si VRAI réitération
si FAUX sortie

```
for( exprInit; condition; exprFin ) {  
    instructions;  
}
```

exécutée à la fin de chaque itération
sert à incrémenter la variable de boucle

LES STRUCTURES DE TRAITEMENT

Pour ... Chaque

- Mots clés : **foreach**

tableau (ou objet) sur lequel l'itération se fait

optionnel :
variable recevant le nom de la clé de l'élément du tableau
ou celui de la propriété de l'objet

```
foreach( $tableau as $cle => $valeur ) {  
    instructions;  
}
```

instructions utilisant la variable \$valeur

variable prenant la valeur de chaque élément du tableau ou celui de la propriété de l'objet

- Mots clés : **function**, **return**

```
function nomDeLaFonction( paramètres formels )
{
    instructions;
    return valeur;
}
```

Variable ou constante
représentant la
valeur de retour de la
fonction

liste de variables ou
constantes séparées
par des virgules

- déclaration possible de variables.

LES FONCTIONS

La portée des variables

**variables
globales**

```
<?php
$age = 0;
function calculAge($anneeNaiss)
{
    $annee = getDate()["year"];
    return $annee - $anneeNaiss;
}
$monAnnee = 1978;
$age = calculAge($monAnnee);
echo 'mon age est : ' . $age;
?>
```

**variables
locales**

LES FONCTIONS

La portée des variables : remarques

- Une fonction accède à une variable globale en la déclarant par le mot clé « **global** »

```
$age = 0;
function calculAge($anneeNaiss)
{
    global $age;
    .....
}
```
- Une variable locale à la durée de vie de la fonction (création à l'appel et destruction).
- Une variable locale à une fonction peut conserver sa valeur entre 2 appels si elle est déclarée par le mot clé « **static** ».

LES FONCTIONS

Le passage de paramètres

- PHP supporte le passage d'arguments
 - par « **valeur** » (comportement par défaut).
 - par « **référence** »

Utiliser le signe « **&** » devant l'argument dans la déclaration de la fonction
 - les **valeurs d'arguments par défaut**

Utiliser le signe « **=** » avec la valeur derrière l'argument dans la déclaration de la fonction (marche pour arg. par valeur et référence)

Si tous les arguments non pas de valeur par défaut, mettre ceux-ci en fin de liste.
 - les **listes variables** d'arguments

Utiliser les fonctions « `func_num_args()`, `func_get_arg()`, `func_get_args()` »

ECRITURE SUR LA CONSOLE

envoi direct

- La fonction « **echo** ».
- Ce n'est pas réellement une fonction.
(parenthèses non obligatoires)
- Admet une liste d'arguments variables de type « string ».

Exemple :

```
echo 'mon age est : ', $age;
```

Remarques :

- 1) Passage de 2 arguments
- 2) si \$age est de type numérique, il sera converti en type « string »
- 3) Il est aussi possible d'utiliser l'opérateur de concaténation

ECRITURE SUR LA CONSOLE

envoi formaté

- La fonction « **printf** »
- Permet une sortie formatée de chaque argument suivant une chaîne de formatage
- Admet un nombre variable d'arguments
- La syntaxe est :

`printf(chaîneFormat, arg1, arg2, ...)`

chaîneFormat est une chaîne de caractères dans laquelle sont présents des spécificateurs de type qui seront remplacés par les arguments

Exemple :

```
printf("mon age est de %d ans", $age);
```

ECRITURE SUR LA CONSOLE

envoi formaté (suite)

- Les principaux spécificateurs de types :

%s	Argument traité comme une chaîne
%c	Argument traité comme un caractère
%d	Argument traité comme un entier
%f	Argument traité comme un réel

- Possibilité de spécifier
 - Le signe du nombre : %+d
 - Le nombre de caractères : %5d
 - Un remplisseur (par défaut espace) : %05d
(ici 0, pour autre caractère le mettre entre ")
 - Un nombre de chiffres après virgule : %6.2f

LES REGLES DE CODAGE

présentation du code

```
$var = 56;
```

espace

```
If ((cond1) && (cond2)) {  
    instruction(s);  
} else {  
    instruction(s);  
}
```

indentation

pas d'espace

```
function maFonction($arg1, $arg2)  
{  
    code;  
}
```

Virgule + espace

LES REGLES DE CODAGE

les identifiants

Majuscule et « _ »

```
const MA_CONSTANTE = 45;
$_maVarGlobale = 56;
```

Minuscule et Camel

```
function maFonction($arg1, $arg2)
{
    $maVarLocale = 0;
}
```

**Majuscule
et Camel**

```
class MaClasse
{
    code;
}
```

LES REGLES DE CODAGE

les constantes

- PHP permet de définir des constantes par la fonction « **define()** ».

Exemple :

```
define("COURS", "PHP");  
define("VERSION", 5.3);
```

- L'avantage des constantes est qu'elles sont visibles dans tout le code.
- Nous verrons plus tard, que PHP défini tout un tas d'autres constantes prédéfinies.
(PHP_VERSION, __FILE__, TRUE, ...)

Utilisation :

```
echo "la version de php est ".VERSION;
```


LES REGLES DE CODAGE

les constantes (suite)

- Une constante n'admet que des valeurs de type scalaire (voir chapitre "Les types").
- Depuis la **version 5.3** de PHP, il est possible d'utiliser le mot clé « **const** »

Exemple :

```
const COURS = "PHP";
```

```
const VERSION = 5.3;
```

- Attention : il faut déclarer les constantes au plus haut niveau.
- Ne pas utiliser « const » dans les fonctions, les boucles ou les instructions si ... alors

Fin de la partie 1

