PHP

March 7, 2022

Les langages de programmations

- PHP est un langage de programmation
- comme d'autres : C, C++, C#, Java, Perl, Python, Basic, ActionScript, etc.
- enfin pas tout à fait comme tous les autres
- Langage interprété (script)
- il y en a d'autres : script-shell, batch-file, perl, python
- fait pour le développement WEB
- il en existe d'autres : perl, python, VBscript, Ruby
- exécuté coté serveur

Les langages de programmations

- PHP est un langage de programmation
- comme d'autres : C, C++, C#, Java, Perl, Python, Basic, ActionScript, etc.
- enfin pas tout à fait comme tous les autres
- Langage interprété (script)
- il y en a d'autres : script-shell, batch-file, perl, python
- fait pour le développement WEB
- il en existe d'autres : perl, python, VBscript, Ruby
- exécuté coté serveur

Les langages de programmations

- PHP est un langage de programmation
- comme d'autres : C, C++, C#, Java, Perl, Python, Basic, ActionScript, etc.
- enfin pas tout à fait comme tous les autres
- Langage interprété (script)
- il y en a d'autres : script-shell, batch-file, perl, python
- fait pour le développement WEB
- il en existe d'autres : perl, python, VBscript, Ruby
- exécuté coté serveur

Les langages de programmations

- PHP est un langage de programmation
- comme d'autres : C, C++, C#, Java, Perl, Python, Basic, ActionScript, etc.
- enfin pas tout à fait comme tous les autres
- Langage interprété (script)
- il y en a d'autres : script-shell, batch-file, perl, python
- fait pour le développement WEB
- il en existe d'autres : perl, python, VBscript, Ruby
- exécuté coté serveur

Les langages de programmations

- PHP est un langage de programmation
- comme d'autres : C, C++, C#, Java, Perl, Python, Basic, ActionScript, etc.
- enfin pas tout à fait comme tous les autres
- Langage interprété (script)
- il y en a d'autres : script-shell, batch-file, perl, python
- fait pour le développement WEB
- il en existe d'autres : perl, python, VBscript, Ruby
- exécuté coté serveur

Les langages de programmations

- PHP est un langage de programmation
- comme d'autres : C, C++, C#, Java, Perl, Python, Basic, ActionScript, etc.
- enfin pas tout à fait comme tous les autres
- Langage interprété (script)
- il y en a d'autres : script-shell, batch-file, perl, python
- fait pour le développement WEB
- il en existe d'autres : perl, python, VBscript, Ruby
- exécuté coté serveur

Les langages de programmations

- PHP est un langage de programmation
- comme d'autres : C, C++, C#, Java, Perl, Python, Basic, ActionScript, etc.
- enfin pas tout à fait comme tous les autres
- Langage interprété (script)
- il y en a d'autres : script-shell, batch-file, perl, python
- fait pour le développement WEB
- il en existe d'autres : perl, python, VBscript, Ruby
- exécuté coté serveur

Les langages de programmations

- PHP est un langage de programmation
- comme d'autres : C, C++, C#, Java, Perl, Python, Basic, ActionScript, etc.
- enfin pas tout à fait comme tous les autres
- Langage interprété (script)
- il y en a d'autres : script-shell, batch-file, perl, python
- fait pour le développement WEB
- il en existe d'autres : perl, python, VBscript, Ruby
- exécuté coté serveur

Avantages

- Ne nécessite pas de compilation
  - facilité de déploiement des applications
  - portabilité (à condition d'avoir un interpréteur pour chaque plate-forme)
- adapté au développement web : structures et fonctions appropriées
- typage faible : rapidité de développement / souplesse

Avantages

- Ne nécessite pas de compilation
  - facilité de déploiement des applications
  - portabilité (à condition d'avoir un interpréteur pour chaque plate-forme)
- adapté au développement web : structures et fonctions appropriées
- typage faible : rapidité de développement / souplesse

Avantages

- Ne nécessite pas de compilation
  - facilité de déploiement des applications
  - portabilité (à condition d'avoir un interpréteur pour chaque plate-forme)
- adapté au développement web : structures et fonctions appropriées
- typage faible : rapidité de développement / souplesse

Avantages

- Ne nécessite pas de compilation
  - facilité de déploiement des applications
  - portabilité (à condition d'avoir un interpréteur pour chaque plate-forme)
- adapté au développement web : structures et fonctions appropriées
- typage faible : rapidité de développement / souplesse

Avantages

- Ne nécessite pas de compilation
  - facilité de déploiement des applications
  - portabilité (à condition d'avoir un interpréteur pour chaque plate-forme)
- adapté au développement web : structures et fonctions appropriées
- typage faible : rapidité de développement / souplesse

Inconvénients

- Ne nécessite pas de compilation :
  - moins rapide en exécution
  - pas de vérification syntaxique
  - débugage à l'exécution
  - problème de protection des sources
  - il faut installer un interpréteur
- Typage faible : pas de vérification de cohérence des types de données

Inconvénients

- Ne nécessite pas de compilation :
  - moins rapide en exécution
  - pas de vérification syntaxique
  - débugage à l'exécution
  - problème de protection des sources
  - il faut installer un interpréteur
- Typage faible : pas de vérification de cohérence des types de données

Inconvénients

- Ne nécessite pas de compilation :
  - moins rapide en exécution
  - pas de vérification syntaxique
  - débugage à l'exécution
  - problème de protection des sources
  - il faut installer un interpréteur
- Typage faible : pas de vérification de cohérence des types de données

Inconvénients

- Ne nécessite pas de compilation :
  - moins rapide en exécution
  - pas de vérification syntaxique
  - débugage à l'exécution
  - problème de protection des sources
  - il faut installer un interpréteur
- Typage faible : pas de vérification de cohérence des types de données

Inconvénients

- Ne nécessite pas de compilation :
  - moins rapide en exécution
  - pas de vérification syntaxique
  - débugage à l'exécution
  - problème de protection des sources
  - il faut installer un interpréteur
- Typage faible : pas de vérification de cohérence des types de données

Inconvénients

- Ne nécessite pas de compilation :
  - moins rapide en exécution
  - pas de vérification syntaxique
  - débugage à l'exécution
  - problème de protection des sources
  - il faut installer un interpréteur
- Typage faible : pas de vérification de cohérence des types de données

Inconvénients

- Ne nécessite pas de compilation :
  - moins rapide en exécution
  - pas de vérification syntaxique
  - débugage à l'exécution
  - problème de protection des sources
  - il faut installer un interpréteur
- Typage faible : pas de vérification de cohérence des types de données

Environnement de développement

- Serveur
  - Serveur web
  - Interpréteur PHP
  - Accès au serveur (répertoire web)
- en local : Wamp/Lamp/Xamp
  - idem serveur mais en local
  - via IDE eclipse :
    - vérification syntaxique
    - débuguage

Environnement de développement

- Serveur
  - Serveur web
  - Interpréteur PHP
  - Accès au serveur (répertoire web)
- en local : Wamp/Lamp/Xamp
  - idem serveur mais en local
  - via IDE eclipse :
    - vérification syntaxique
    - débuguage

Environnement de développement

- Serveur
  - Serveur web
  - Interpréteur PHP
  - Accès au serveur (répertoire web)
- en local : Wamp/Lamp/Xamp
  - idem serveur mais en local
  - via IDE eclipse :
    - vérification syntaxique
    - débuguage

Environnement de développement

- Serveur
  - Serveur web
  - Interpréteur PHP
  - Accès au serveur (répertoire web)
- en local : Wamp/Lamp/Xamp
  - idem serveur mais en local
  - via IDE eclipse :
    - vérification syntaxique
    - débuguage

Environnement de développement

- Serveur
  - Serveur web
  - Interpréteur PHP
  - Accès au serveur (répertoire web)
- en local : Wamp/Lamp/Xamp
  - idem serveur mais en local
  - via IDE eclipse :
    - ⋆ vérification syntaxique
    - ⋆ débuguage

Environnement de développement

- Serveur
  - Serveur web
  - Interpréteur PHP
  - Accès au serveur (répertoire web)
- en local : Wamp/Lamp/Xamp
  - idem serveur mais en local
  - via IDE eclipse :
    - ★ vérification syntaxique
    - ★ débuguage

Environnement de développement

- Serveur
  - Serveur web
  - Interpréteur PHP
  - Accès au serveur (répertoire web)
- en local : Wamp/Lamp/Xamp
  - idem serveur mais en local
  - via IDE eclipse :
    - ★ vérification syntaxique
    - ★ débuguage

Environnement de développement

- Serveur
  - Serveur web
  - Interpréteur PHP
  - Accès au serveur (répertoire web)
- en local : Wamp/Lamp/Xamp
  - idem serveur mais en local
  - via IDE eclipse :
    - ★ vérification syntaxique
    - ★ débuguage

Environnement de développement

- Serveur
  - Serveur web
  - Interpréteur PHP
  - Accès au serveur (répertoire web)
- en local : Wamp/Lamp/Xamp
  - idem serveur mais en local
  - via IDE eclipse :
    - ★ vérification syntaxique
    - ★ débuguage

Les atouts de PHP

- Open Source
- Communauté d'utilisateurs très importante (aide/doc)
- Interopérabilité avec la plupart des technologies liées à internet
  - services internet : FTP, LDAP, MAIL
  - base de données : MySQL MS SQL, PostgreSQL, Oracle, informix
  - web : XML, Web services
  - interface avec les systèmes de paiement sécurisé

Les atouts de PHP

- Open Source
- Communauté d'utilisateurs très importante (aide/doc)
- Interopérabilité avec la plupart des technologies liées à internet
  - services internet : FTP, LDAP, MAIL
  - base de données : MySQL MS SQL, PostgreSQL, Oracle, informix
  - web : XML, Web services
  - interface avec les systèmes de paiement sécurisé

Les atouts de PHP

- Open Source
- Communauté d'utilisateurs très importante (aide/doc)
- Interopérabilité avec la plupart des technologies liées à internet
  - services internet : FTP, LDAP, MAIL
  - base de données : MySQL MS SQL, PostgreSQL, Oracle, informix
  - web : XML, Web services
  - interface avec les systèmes de paiement sécurisé

Les atouts de PHP

- Open Source
- Communauté d'utilisateurs très importante (aide/doc)
- Interopérabilité avec la plupart des technologies liées à internet
  - services internet : FTP, LDAP, MAIL
  - base de données : MySQL MS SQL, PostgreSQL, Oracle, informix
  - web : XML, Web services
  - interface avec les systèmes de paiement sécurisé

Les atouts de PHP

- Open Source
- Communauté d'utilisateurs très importante (aide/doc)
- Interopérabilité avec la plupart des technologies liées à internet
  - services internet : FTP, LDAP, MAIL
  - base de données : MySQL MS SQL, PostgreSQL, Oracle, informix
  - web : XML, Web services
  - interface avec les systèmes de paiement sécurisé

Les atouts de PHP

- Open Source
- Communauté d'utilisateurs très importante (aide/doc)
- Interopérabilité avec la plupart des technologies liées à internet
  - services internet : FTP, LDAP, MAIL
  - base de données : MySQL MS SQL, PostgreSQL, Oracle, informix
  - web : XML, Web services
  - interface avec les systèmes de paiement sécurisé

Les atouts de PHP

- Open Source
- Communauté d'utilisateurs très importante (aide/doc)
- Interopérabilité avec la plupart des technologies liées à internet
  - services internet : FTP, LDAP, MAIL
  - base de données : MySQL MS SQL, PostgreSQL, Oracle, informix
  - web : XML, Web services
  - interface avec les systèmes de paiement sécurisé

Rappels de survie sur http

Fonctionnement d'un serveur WEB

Le Web

- un protocole : http
- un format : html (heu! et d'autres html4, html5, xhtml,...)

Fonctionnement

- Le client envoie une requête au serveur:
  - http://ufrsciencestech.u-bourgogne.fr ⇒ requête http (via protocole http - port 80)
  - ftp://... ⇒ requête ftp (via protocole ftp - port 21)
  - demande l'adresse IP du serveur via DNS

Fonctionnement d'un serveur WEB

Le Web

- un protocole : http
- un format : html (heu! et d'autres html4, html5, xhtml,...)

Fonctionnement

- Le client envoie une requête au serveur:
  - http://ufrsciencestech.u-bourgogne.fr ⇒ requête http (via protocole http - port 80)
  - ftp://... ⇒ requête ftp (via protocole ftp - port 21)
  - demande l'adresse IP du serveur via DNS

Rappels de survie sur http

- Le Serveur renvoie :
  - via http
  - la page par défaut (index.html) du site localisé sur le serveur
    si l'extension est .html ou .htm le fichier est retourné tel quel.
  - ou le résultat de l'exécution du programme par défaut (index.php) du site
- **Pour que le code php soit interprété (coté serveur) le fichier doit avoir l'extension .php**
- Le client interprète le contenu de la réponse,
  (contenu fourni au format html)

Exemple

- index.html

```
<html>
  <body>
  <p>Coucou tout le monde</p>
  </body>
</html>
```

- index.html

```
<html>
  <body>
    <p>
        <?php print('Coucou tout le monde'); ?>
    </p>
  </body>
</html>
```

Exemple

- index.html

```
<html>
  <body>
  <p>Coucou tout le monde</p>
  </body>
</html>
```

- index.html

```
<html>
  <body>
    <p>
        <?php print('Coucou tout le monde'); ?>
    </p>
  </body>
</html>
```

Exemple

- index.php

```
<html>
  <body>
    <p>
        <?php print('Coucou tout le monde'); ?>
    </p>
  </body>
</html>
```

PHP : B-A BA

Structure d'un programme

= ensemble d'instructions entre "<?php" et "?>".

```
<?php
print("instruction 1");
?>
```

"<?php" indique à l'interpréteur que les lignes qui suivent doivent être traitées comme du code php

Structure d'un programme

Le résultat est interprété par défaut par le client comme un code html

```php
<?php
print("instruction 1");
print("instruction 2");
?>
```

```php
<?php
print("instruction 1\n");
print("instruction 2");
?>
```

```php
<?php
print("instruction 1<br/>");
print("instruction 2");
?>
```

Structure d'un programme

Le résultat est interprété par défaut par le client comme un code html

```
<?php
print("instruction 1");
print("instruction 2");
?>
```

```
<?php
print("instruction 1\n");
print("instruction 2");
?>
```

```
<?php
print("instruction 1<br/>");
print("instruction 2");
?>
```

Structure d'un programme

Le résultat est interprété par défaut par le client comme un code html

```
<?php
print("instruction 1");
print("instruction 2");
?>
```

```
<?php
print("instruction 1\n");
print("instruction 2");
?>
```

```
<?php
print("instruction 1<br/>");
print("instruction 2");
?>
```

Structure d'un programme

```
<html>
<head><title>Exemple html/php</title></head>
<body>
<p>HTML dans le fichier</p>
<p>
        <?php
        print('<hr/>html généré par php<hr/>');
        ?>
</p>
<p>de nouveau du html dans le texte</p>
        <?php
        print('<h1> Ce text est généré');
        print(' via php</h1>');
        ?>
</p>
</body>
</html>
```

Les variables

- nom commençant par "$" : $toto, $machin
- doit commencer par une lettre ou (_)
- sensible à la casse
- la déclaration se fait lors de l'affectation
- pas de déclaration de type, le type est déterminé automatiquement lors de l'affectation (typage dynamique).

Les variables

Notation alternative : ${'nom_de_la_variable'}.
Variables dynamiques

```
$X=0 ;
${'X'}=10 ; //assigne la valeur 10 à la variable X
print($('X'}); // affiche 10
```

Exemple

```
$perso1 = 'luc';
$perso2 = 'pierre';
$perso3 = 'jason';
for($i = 1; $i <= 3; $i++)
    echo ${'perso' . $i};
```

Mais les tableaux c'est mieux!!!

Les variables

Notation alternative : ${'nom_de_la_variable'}.
Variables dynamiques

```
$X=0 ;
${'X'}=10 ; //assigne la valeur 10 à la variable X
print($('X'}); // affiche 10
```

Exemple

```
$perso1 = 'luc';
$perso2 = 'pierre';
$perso3 = 'jason';
for($i = 1; $i <= 3; $i++)
    echo ${'perso' . $i};
```

Mais les tableaux c'est mieux!!!

Les variables

Assignation par référence &nom_variable

```php
<?php
$foo = 'Pierre'; //Assigne la valeur 'Pierre' à $foo
$bar = &$foo;              // Référence $foo avec $bar.
$bar = "Mon nom est $bar";  // Modifie $bar...
echo $foo;                 // $foo est aussi modifiée
echo $bar;
?>
```

Les variables : portée des variables :

Une variable définie dans le script d'une page

- = variable globale
- Portée =
  - la totalité du script
  - les scripts de la même page
  - les scripts "inclus" dans la page

Les variables globales

- sont accessibles par une variable superglobale : le tableau $GLOBALS
- $GLOBALS['toto'] accède à la variable globale $toto.

Les variables : portée des variables :

Une variable définie dans le script d'une page

- = variable globale
- Portée =
  - la totalité du script
  - les scripts de la même page
  - les scripts "inclus" dans la page

Les variables globales

- sont accessibles par une variable superglobale : le tableau $GLOBALS
- $GLOBALS['toto'] accède à la variable globale $toto.

Les variables : portée des variables :

Une variable définie dans une fonction

- a une portée locale (limitée à la fonction)
- on peut accéder aux variables globales en :
  - déclarant dans la fonction : global $toto;
  - via la variable superglobale $GLOBALS

ATTENTION à l'abus des variables globales

- = effet de bords
- débogage très difficile

Exemples ▸ résultats ▸ fichier source

Les variables : portée des variables :

Une variable définie dans une fonction

- a une portée locale (limitée à la fonction)
- on peut accéder aux variables globales en :
  - déclarant dans la fonction : global $toto;
  - via la variable superglobale $GLOBALS

ATTENTION à l'abus des variables globales

- = effet de bords
- débogage très difficile

Exemples ▸ résultats ▸ fichier source

les variables : variables prédéfinies

Les variables prédéfinies = var superglobale toujours disponibles,

- $GLOBALS : Référence toutes les variables disponibles dans un contexte global
- $_SERVER : Variables de serveur et d'exécution
- $_GET : Variables HTTP GET
- $_POST : Variables HTTP POST
- $_FILES : Variable de téléchargement de fichier via HTTP
- $_REQUEST : Variables de requête HTTP
- $_SESSION : Variables de session
- $_ENV : Variables d'environnement
- $_COOKIE : Cookies HTTP
- $php_errormsg : Le dernier message d'erreur
- $HTTP_RAW_POST_DATA : Données POST brutes
- $http_response_header : En-têtes de réponse HTTP
- $argc : Le nombre d'arguments passés au script
- $argv : Tableau d'arguments passés au script

Les variables

Etat d'une variable

Affectée (set)

- : $toto= 3. ;
- test via la fct : isset($toto) (vraie si affectée)

Vide

- si la valeur 0 ou chaîne vide ("")
- test via la fonction empty($toto)

Les variables

Etat d'une variable

Affectée (set)

- : $toto= 3. ;
- test via la fct : isset($toto) (vraie si affectée)

Vide

- si la valeur 0 ou chaîne vide ("")
- test via la fonction empty($toto)

Les variables

Etat d'une variable (suite)

Non affectée

- sans valeur : $toto= NULL ou pas encore reçue de valeur ou effacée (fonction unset());
- test via la fonction is_null($toto)

Remarques

Les fonctions isset() ou empty() peuvent être utilisées pour tester si le champ d'un formulaire a été rempli.
Mais attention une variable peut être affectée à vide.

Les variables

Etat d'une variable (suite)

Non affectée

- sans valeur : $toto= NULL ou pas encore reçue de valeur ou effacée (fonction unset());
- test via la fonction is_null($toto)

Remarques

Les fonctions isset() ou empty() peuvent être utilisées pour tester si le champ d'un formulaire a été rempli.
Mais attention une variable peut être affectée à vide.

Les constantes

Définition:

- define("MA_CSTE_COUCOU","coucou");
- Par convention le nom est en majuscule (mais pas obligatoire)

Utilisation :

```
<?php
define("NOM_SITE","Cool Site");
print(NOM_SITE);
define("VERSION_SITE",3.1);
$a = VERSION_SITE + 1;
?>
```

On les utilise souvent pour les connexions à la BD!
Ok, mais le sécurité alors ?

Les constantes

Remarque sur les constantes:

- Il existe des constantes définies par défaut : exemple PHP_VERSION.
- pour obtenir la liste des constantes définies par défaut : fonction get_defined_constants())

```
<pre>
<?php
print_r(get_defined_constants());
?>
</pre>
```

print_r() : affiche les informations à propos d'une variable

Les types

Types scalaires

- boolean
- integer
- float (≈double)
- string

Types composés

- array
- object

types spéciaux

- resource : (avec 1 s en anglais) référence vers une ressource externe, retournée pas des fonctions (exemples: BD, ftp, pdf, image,...)
- NULL

pseudo-types

- mixed ; indique qu'un paramètre peut accepter plusieurs types (ex: la fonction gettype() accepte tous les types)
- number : soit un entier soit un nombre décimal

Conversion de type

```php
$s=(string) 1; // $s contient maintenant la chaîne "1"
$n=(int) $s; // $n contient 1
$b=(bool) $n; // $b contient TRUE
$n=(int) 1.8; // $n contient 1
$b=(bool) -3; // $b contient TRUE
$b=(bool) 0; // $b contient FALSE
$f=(float) " 1.45 "; // $f contient 1.45
```

Type des variables

test du type

- is_numeric() : Attention la chaîne '42' est "numérique"
- is_string : mais '42' est aussi une chaîne Exemples ▸ démo
- is_array()
- is_double(), is_float(),is_int()
- is_object()
- is_resource()
- is_scalar() (integer, float, string, ou boolean (mais pas array, object , ressource)

Type des variables

test du type

- is_numeric() : Attention la chaîne '42' est "numérique"
- is_string : mais '42' est aussi une chaîne Exemples ▸ démo
- is_array()
- is_double(), is_float(),is_int()
- is_object()
- is_resource()
- is_scalar() (integer, float, string, ou boolean (mais pas array, object , ressource)

Les types numériques

integer, float

Opérateurs

- + L'addition
- - La soustraction
- * La multiplication
- / La division
- % Le modulo
- ** La puissance

Les chaînes de caractères

- Ensemble de caractères délimité par (') ou (")
- si (') la chaîne est telle quelle est écrite
- si (") si la chaîne contient des variables elles sont remplacées par leur valeur.

Les chaînes de caractères

Exemple

```
$toto='Lucien';
print('boujour $toto'); // affiche : bonjour $toto
print("boujour $toto"); // affiche : bonjour Lucien
print("boujour \$toto"); // affiche : bonjour $toto
```

Remarque

- les chaînes avec (") nécessitent un traitement supplémentaire.
- Les utiliser que s'il y a des variables dont on souhaite récupérer les valeurs.

Les chaînes de caractères

Exemple

```php
$toto='Lucien';
print('boujour $toto'); // affiche : bonjour $toto
print("boujour $toto"); // affiche : bonjour Lucien
print("boujour \$toto"); // affiche : bonjour $toto
```

Remarque

- les chaînes avec (") nécessitent un traitement supplémentaire.
- Les utiliser que s'il y a des variables dont on souhaite récupérer les valeurs.

Affichage des chaînes et variables

Usage

- echo('bonjour');

Affichage des guillemets

Alternance des (') et (") :

- echo 'bonjour "toto"';
- echo "bonjour 'toto'";

Ou utilisation du caractère d'échappement (\)

- echo 'bonjour \'toto\' ';
- echo "bonjour \"toto\"";

Opérateurs et fonctions sur les chaînes

concaténation (.)

- print("cou" . 'cou'); // affiche : "coucou"
- (.=) : $str1 .= 'coucou' equivaut à $str1 = $str1 . 'coucou'
- pour les fonctions de traitement des chaînes voir la doc en ligne.

Quelques fontions

Nettoyage de chaînes :

ltrim()	Supprime les espaces (ou d'autres caractères) de début de chaîne
rtrim()	Supprime les espaces (ou d'autres caractères) de fin de chaîne
trim()	Supprime les espaces (ou d'autres caractères) en début et fin de chaîne
strip_tags()	Supprime les balises HTML et PHP d'une chaîne
stripslashes()	Supprime les antislashs d'une chaîne

autres traitements :

explode()	retourne un tableau de chaînes, chacune d'elle étant une sous-chaîne
implode()	Rassemble les éléments d'un tableau en une chaîne
htmlspecialchars()	Convertit les caractères spéciaux en entités HTML
parse_str()	Analyse une chaîne de caractères comme s'il s'agissait des paramètres passés via l'URL et créer les variables correspondantes
soundex (string $str)	La clé soundex possède la propriété qui fait que deux mots prononcés similairement auront la même clé soundex.

Les structures de contrôle

if else

```php
<?php
$i = 4;
if ($i > 5)
{
print("i est plus grand que 5");
}
else
{
print("i est plus petit que 5");
}
?>
```

Les structures de contrôle

Les opérateurs de comparaison

- $a == $b : vrai si $a est égal à $b après le transtypage
- $a === $b : vrai si $a est égal à $b et si ces deux variables sont de même type
- $a != $b : vrai si $a est différent de $b
- $a !== $b : vrai si $a est différent de $b (en type ou en valeur)
- $a > $b : vrai si $a est supérieur à $b
- $a < $b vrai si $a est inférieur à $b
- $a >= $b vrai si $a est supérieur ou égal à $b
- $a <= $b vrai si $a est inférieur ou égal à $b

Les structures de contrôle

Les opérateurs logiques

- $a and $b : vrai si $a ET $b sont vraies
- $a && $b : vrai si $a ET $b sont vraies (identique à and)
- $a or $b : vrai si $a OU $b sont vraies
- $a || $b : vrai si $a OU $b sont vraies (identique à or)
- $a xor $b : vrai si $a OU $b sont vraies, mais pas les deux
- !$a : vrai si $a est fausse

Les structures de contrôle

if .. elseif

```
<?php
if ((($couleur=="rouge") || ($couleur=="jaune") ||($couleur=="
    bleue")))
print("primaire");
elseif ($couleur == "noire")
print("noire");
elseif ($couleur == "blanche")
print("blanche");
else
print("mélange");
?>
```

Les structures de contrôle

switch

```
switch ($couleur)
{
case "rouge":
print("R");
break;
case "bleue":
print("B");
break;
case "jaune":
print("J");
break;
default:
print("?");
break;
}
```

Un 'case' peut contenir plusieurs instructions

Les structures de contrôle

while

```php
$i = 0;
while ($i < 5)
{
print("$i<br/>");
$i++;
}
print("<br/>fin de la boucle");
```

Les structures de contrôle

while

```php
$i = 0;
while ($i < 5)
{
print("$i<br/>");
$i++;
}
print("<br/>fin de la
    boucle");
```

do while

```php
$i = 6;
do
{
print($i);
}
while ($i < 5);
```

Exécuté une fois

Les structures de contrôle

for

```php
for ($i = 0; $i <= 10; $i++)
{
print("$i <br/>");
}
```

Possibilité d'utiliser "break" et "continue"

Les fonctions

Déclaration

```
function nom_fonction($param1="val_par_defaut",$parma2="
    val_par_defaut")
{
//bloc_de_code;
}
```

Exemple

```
function facto($n)
{ if ($n == 1) return 1;
  else return $n * facto($n-1); }
```

Les fonctions

Déclaration

```
function nom_fonction($param1="val_par_defaut",$parma2="
    val_par_defaut")
{
//bloc_de_code;
}
```

Exemple

```
function facto($n)
{ if ($n == 1) return 1;
  else return $n * facto($n-1); }
```

les fonctions

Remarques

- pas de déclaration du type de la valeur retournée,
- déterminé automatiquement à partir de la valeur retournée par "return"

Les fonctions

Les variables : local, global (cf portée des variables), static

les variables "static"

- définies que dans une fonction
- elles ont une portée locale
- mais elles ne perdent pas leur valeur à l'appel de la fonction
- déclarées avec le mot clé "static"

Les fonctions : Exemple variable static

Variable locale

```php
<?php
function test()
{
    $a = 0;
    echo $a;
    $a++;
}
?>
```

Affiche toujours 0

Variable static

```php
<?php
function test()
{
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

'a' est incrémenté à chaque appel

Les fonctions : Exemple variable static

Variable locale

```
<?php
function test()
{
    $a = 0;
    echo $a;
    $a++;
}
?>
```

Affiche toujours 0

Variable static

```
<?php
function test()
{
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

'a' est incrémenté à chaque appel

Inclusion de fichier

- include() :
  inclus le fichier mais se contente d'afficher un avertissement et poursuit l'exécution du script s'il y a échec
- include_once() :
  le fichier n'est inclus que s'il ne l'a pas été auparavant.
- require() :
  cette fonction arrête le script si elle n'est pas parvenue à inclure le fichier passé en argument.
- require_once() :
  le fichier n'est inclus que s'il ne l'a pas été auparavant.

Inclusion de fichier

- include() :
  inclus le fichier mais se contente d'afficher un avertissement et poursuit l'exécution du script s'il y a échec
- include_once() :
  le fichier n'est inclus que s'il ne l'a pas été auparavant.
- require() :
  cette fonction arrête le script si elle n'est pas parvenue à inclure le fichier passé en argument.
- require_once() :
  le fichier n'est inclus que s'il ne l'a pas été auparavant.

Inclusion

Remarques

Test des erreurs

```php
if (include('script.php') == false) {
echo 'Erreur lors de l'inclusion de script.php';
}
```

Extension des fichiers inclus

- par convention .inc.php ou (.php)
- jamais .inc
- surtout pour les fichiers contenant les login/passwd

Inclusion

Attention

```php
// Ne fonctionne pas : file.txt n'a pas été traité par
    www.example.com comme du PHP
include 'http://www.example.com/file.txt?foo=1&bar=2';

// Ne fonctionne pas : le script cherche un fichier nommé
// 'file.php?foo=1&bar=2' sur le système local
include 'file.php?foo=1&bar=2';

// Réussi
include 'http://www.example.com/file.php?foo=1&bar=2';

$foo = 1;
$bar = 2;
include 'file.txt';  // Ok.
include 'file.php';  // Ok.
```