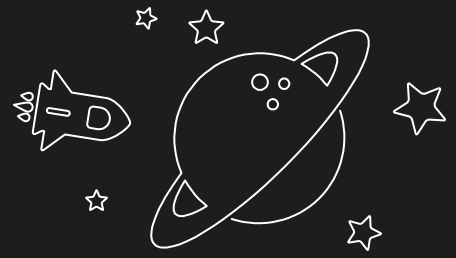
Les fondamentaux



Exemple de code

2

```php
<?php
/**
* Undocumented function
*
* @param string $message
* @return bool
*/
function display(string $message): bool
{
  // display message
  echo $message;
  return false;
}

display("Bonjour from PHP");
```

3

A RETENIR ABSOLUMENT

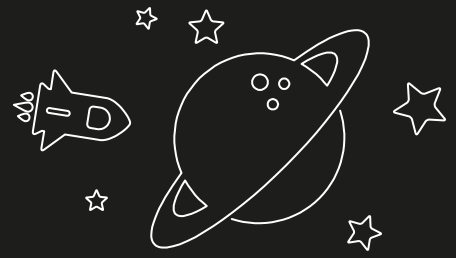
Une instruction PHP commence toujours par la balise “<?php” ou “<?=” et finit par “?>”

<?= “test” ?>

équivaut à <?php echo “test” ?>

Une instruction finit toujours par un point-virgule (;)

La déclaration ou l’utilisation d’une variable commence toujours par un dollar ($)



Les variables

5

Typage

Variable basique => type faible

Variable de fonction/Classe => typage fort

Types primitifs

string => “message”

int => 2 3

float => 2.3 7.5

bool => true false

null => null

Types complexes

Object => $date = new Date();

Array

Itératif => [1, 3, “test”]

Associatif => [

“type” => “voiture”,

“marque” => “Opel”

]

Les opérateurs

	+	-	*	/	.
Int / Float	Add	Sub	Mult	Div	
String					Concat
Boolean	Add	Sub	Mult	Div	
Array	Concat				

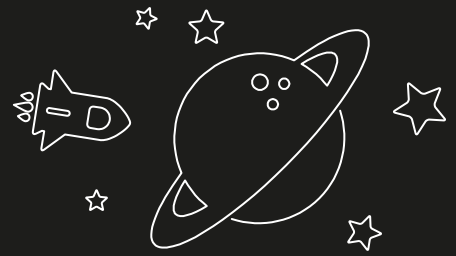
Exemples d'utilisation

```php
<?php

$a = 1;
$b = 2;
echo $a + $b; // affiche "3"

$a = 3;
$b = true;
echo $a + $b; // affiche "4"

$a = 3;
$b = " fourchettes";
echo $a . $b; // affiche "3 fourchettes"
```



BLOC DE STRUCTURE

9

Les conditions dans les balises PHP

if

```
If (cond) {
    //Something
} else if (cond2) {
    //Something
} else {
    //Something
}
```

Opérateurs

Supérieur	**>, >=**
Inférieur	**<, <=**
Égalité	**== ou ===**
Différent	**!= ou !==**

switch

```
switch($var) {
  case “v1”:
    //Something
    break;
  default:
    //Something
    break;
}
```

Les conditions dans une page HTML

```php
<?php
if ($time < 12) {
    echo "<h1>Bonjour</h1>";
} else {
    echo "<h1>Bonsoir</h1>";
}
?>
```

<==>

```php
<h1>
    <?php if ($time < 12) : ?>
        Bonjour
    <?php else : ?>
        Bonsoir
    <?php endif; ?>
</h1>
```

Les opérateurs ternaires

```php
<?php
echo "<h1>"
      . ( $time < 12
            ? "Bonjour" : "Bonsoir" ) .
      "</h1>";
?>
```

<==>

```php
<h1>
   <?=
      $time < 12 ? "Bonjour" : "Bonsoir"
      ?>
</h1>
```

Les boucles

Boucle for classique

```php
for ($i = 0; $i < 10; $i++) {
    echo "<p>Je suis la ligne $i</p>";
}
```

Boucle foreach

```php
// simple array
foreach ($users as $user) {
    echo "<h1>" . $user . "</h1>";
}
// associative array
foreach ($cart as $product => $quantity) {
    echo "<tr><td>" . $product . "</td><td>" .
$quantity . "</td></tr>";
}
```

Les boucles

Boucle do while

```php
$i = 0;
do {
    echo "<h1>" . $user[$i] . "</h1>";
} while (++$i < count($users));
```

Boucle while

```php
while ($i < count($users)) {
    echo "<h1>" . $user[$i++] . "</h1>";
}
```

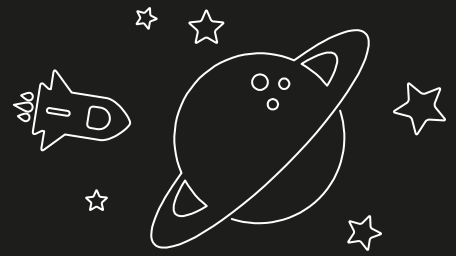
Les opérateurs de boucles

Break *X*

Stop les itérations de X boucles

Continue

Passe à l'itération suivante



Exercices
Manipulation du langage

16

Afficher le numéro de chaque question avant les différents listings

1) Afficher une liste de tâches contenus dans un *tableau simple*
2) Afficher une liste de tâches ainsi que son statut **completed *(bool)*** contenus dans un *tableau associatif*
3) Selon la valeur de la variable **isCompleted *(bool)***, afficher une liste de tâches ainsi que son statut **completed** contenus dans un tableau associatif uniquement si son statut vaut la variable **isCompleted**
4) Afficher une liste de tâches ainsi que son statut **completed** et son **auteur *(string)*** contenus dans un *tableau associatif*
5) Selon la valeur de la variable **isCompleted** et **filterOwner *(string)***, afficher une liste de tâches ainsi que son statut **completed** et son **auteur** contenus dans un tableau associatif uniquement si son statut vaut la variable **isCompleted** et son auteur commence par **filterOwner**
6) Si le *tableau associatif* est vide, afficher “Aucune tâche créée”
7) Si après filtres aucun élément est affiché, alors afficher “Aucune tâche correspondant aux critères”

Question 1

Faire exo1
Faire exo2
Faire exo3

Question 2

Faire exo1 completed
Faire exo2 not completed
Faire exo3 completed

Question 3 (filtre completed)

Faire exo1 completed
Faire exo3 completed

Question 3 (filtre not completed)

Faire exo2 not completed

Question 4

Faire exo1 completed User 1
Faire exo2 not completed User 2
Faire exo3 completed User 12

Question 5 (completed, User 1)

Faire exo1 completed User 1
Faire exo3 completed User 12

Question 5 (completed, User 12)

Faire exo3 completed User 12

Question 5 (not completed, User 1)

Question 6 (data)

Faire exo1 completed User 1
Faire exo2 not completed User 2
Faire exo3 completed User 12

Question 6 (no data)

Aucune tâche créée

Question 7 (data)

Faire exo1 completed User 1
Faire exo2 not completed User 2
Faire exo3 completed User 12

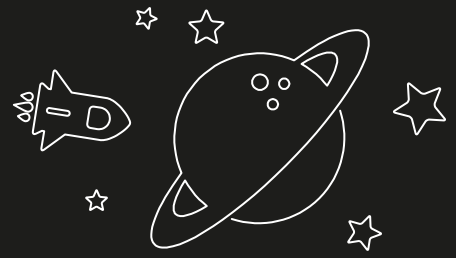
Question 7 (no data)

Aucune tâche créée

Question 7 (data, not completed, User 1)

Aucune tâche correspondant aux critères

Exemple attendu



Requête HTTP

19

20

Structure d'une URL HTTP

<protocol>**://**<hostname>**:**<port>**/**<path>**?**<query_params>**#**<anchor>

protocol: http ou https

hostname: représente l'adresse du site

port: si non visible 80 (http) ou 443 (https)

path: basiquement représente l'arborescence du site

query params: paramètre de la requête GET

ex: var1=val1**&**var2=val2

anchor: permet de scroller à l'élément ID

21

Variables globales

$_GET: représente les query params sous la forme d'un tableau associatif

Ex: **?**var1=val1**&**var2=val2

=> ['var1' => 'val2', 'var2' => 'val2']

$_POST: représente les post params (formulaire en méthode POST) sous la forme d'un tableau associatif

```
<form method="POST">
   <input name="email"/>
   <input name="password"/>
</form>
```

=>

```
[
   'email' => 't@test.com',
   'password' => '12345AZER'
]
```

22

Variables globales

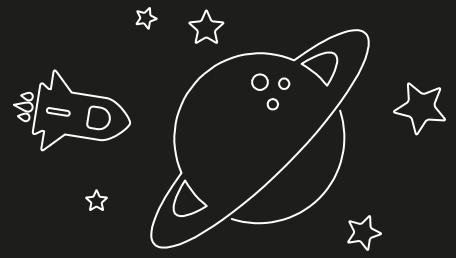
$_REQUEST: représente le mélange des query params et des post params sous la forme d'un tableau associatif. Les post params sont prioritaires

$_SERVER: contient les informations du serveur et de la requête sous forme de tableau associatif.

A retenir:

$_SERVER['REQUEST_METHOD**']** => donne la méthode HTTP utilisée

$_SERVER['REQUEST_URI**']** => donne le path



Découpage du code

23

24

Inclusion de fichiers

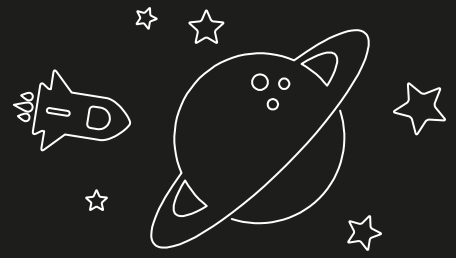
Include / **include_once** ./monfichier.php

Inclut le contenu du fichier cible dans le contexte actuelle (sorte de copié/collé).

Affiche un warning si le fichier n'est pas trouvé.

require / **require_once** ./monfichier.php

Même chose que les instructions *include* mais déclenche une erreur si le fichier n'existe pas.



Requête SQL (PDO)

25

26

Structure d’une URL PDO (DSN)

<protocol>**:dbname=**<dbname>**;host=**<hostname>**;port=**<port>

protocol: pgsql (PostgresSQL) mysql (MySql)

dbname: représente le nom de la database à se connecter

hostname: représente l’adresse de la database

port: si non visible 3306 (mysql) ou 5432 (PostgresSQL)

27

Utilisation de PDO

Créer une nouvelle connexion

```
$db = new PDO($dsn, $user, $password);
```

$dsn: chaîne représente une adresse DSN (slide précédente)

$user: nom d'utilisateur ayant accès à la database **<dbname>**

$password: mot de passe de l'utilisateur précédent

28

Utilisation de PDO

Requête en utilisant la méthode QUERY

/!\ Risque d'injection SQL à utiliser si les valeurs ne viennent pas d'une entrée utilisateur

```
$db = new PDO($dsn, $user, $password);

$stmt = $db->query('SELECT * FROM task', PDO::FETCH_ASSOC);

$rows = $stmt->fetchAll();
```

$stmt: représente un statement, curseur vers les résultats de la requête

29

Utilisation de PDO

Requête en utilisant la méthode PREPARE

/!\ Vérifie les params de requête pour éviter les injections SQL

```
$db = new PDO($dsn, $user, $password);

$stmt = $db->prepare('SELECT * FROM task WHERE id = :id');

$stmt->execute([ 'id' => $idTask ]);

$rows = $stmt->fetchAll();
```

30

Utilisation de PDO

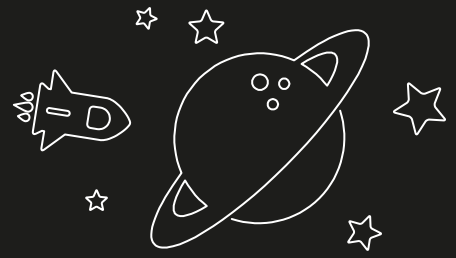
Fonctions disponibles pour les statements

->fetchAll(): retourne l'ensemble des résultats sous forme de tableau itératif

->fetch(): retourne la prochaine valeur des résultats disponibles (utilisable plusieurs fois)

->fetchColumn($indexColumn**):** retourne la colonne à l'index $indexColumn du premier résultat (pratique pour des COUNT)

->rowCount(): retourne le nombre total de ligne du résultat d'une requête



Les sessions

31

32

Les sessions PHP

Les sessions PHP permettent d'identifier les **visiteurs** d'un site. Elles ne permettent pas d'identifier directement un **utilisateur**.

Fonctionnement :

$_SESSION: variable globale qui permet de stocker des informations sur le visiteur

PHPSESSID: cookie permettant d'identifier une session donc un visiteur

session_start: fonction permettant de démarrer ou reprendre une session, doit être lancée dès le premier script PHP lancé par la requête.

session_destroy: fonction permettant de détruire une session

Authentification (workflow)

/login

- Démarrer une session : **session_start()**
- Récupérer l'utilisateur dans la DB par son mail
- Vérifier son mot de passe crypté: **password_verify**
- Ajouter l'utilisateur dans la session : **$_SESSION['user'] = $user**

/logout

- Détruire la session: **session_destroy()**