

Pourquoi utiliser des fonctions

Fonctions natives sur les tableaux

Fonction natives sur les variables

Fonctions natives sur les dates

Création de fonction php personnalisées

Include / Require / Require once



Pourquoi utiliser des fonctions

Permet de réutiliser son code source

Gain en terme de temps de développement

Gain en terme de maintenance

Prend une ou plusieurs données en entrée



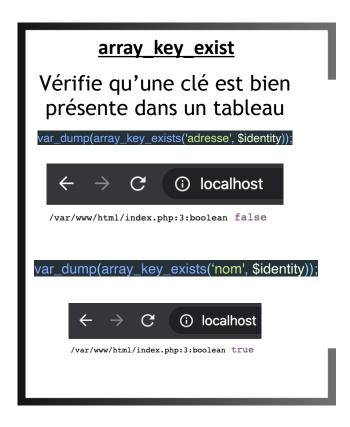
Peut retourner une donnée en sortie ou simplement exécuter une action

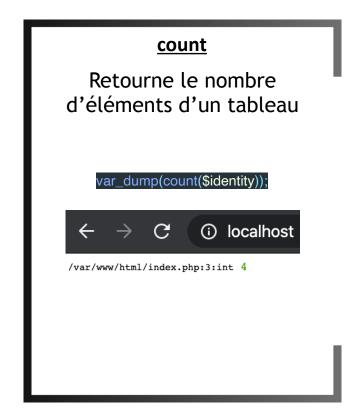
Effectue un traitement sur ses données

Une fonction se veut REUTILISABLE

Fonctions natives sur les tableaux

\$identity = ['nom' => 'Delorme', 'prenom' => 'Aurelien', 'age' => 28, 'dateNaissance' => date_create('30-03-1993')->format('d/m/Y')];







Fonctions natives sur les tableaux

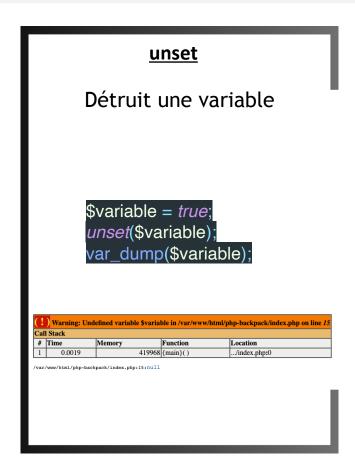
\$listeCourse = ['pomme', 'poire', 'chocolat', 'poire', 'pomme'];

array_search Recherche dans un tableau la clé associé à la première valeur var_dump(array_search('pomme', \$listeCourse)); ← → C ① localhost /var/www/html/index.php:3:int 0 var_dump(array_search('poire', \$listeCourse)); ← → C ① localhost /var/www/html/index.php:3:int 1

array_unique Retourne un tableau en supprimant les doublons var_dump(array_unique(\$listeCourse)); /var/www/html/php-backpack/index.php:5: array (size=3) 0 => string 'pomme' (length=5) 1 => string 'poire' (length=5) 2 => string 'chocolat' (length=8)

Fonctions sur les variables







Fonctions sur les variables



```
is string
Indique si une variable est de
             type string
<?php
  variable = 10;
  var_dump(is_string($variable));
                       (i) localhost
   /var/www/html/index.php:3:boolean false
       $variable = 'Aurélien';
       var_dump(is_string($variable));
                       (i) localhost
     /var/www/html/index.php:3:boolean true
```



Fonctions sur les dates

date(format)

Retourne la date du jour sous forme de chaine de caractère. Prend en paramètre le format de retour souhaité

```
<?php
$date = date('d/m/Y');
echo('Nous somme le '. $date)
2</pre>
```



Nous somme le 20/02/2020

date(format, timestamp)

Retourne la date sous forme de chaine de caractère. Prend en second paramètre le timestamp recherché

```
<?php
$date = date('d/m/Y', 733486810)
    echo('Nous somme le '. $date);
?>
```



Nous somme le 30/03/1993

date create

Permet de créer une date depuis le format suivant : yyyy-mm-dd

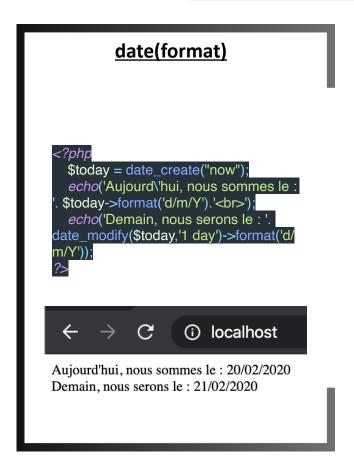
Equivalent à new DateTime()

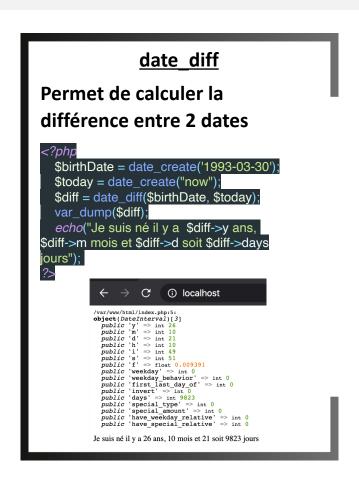
```
<?php
$date1 = date_create("2013-03-15");
$sameDate = new DateTime('2013-03-15');
var_dump($sameDate);
?>
```

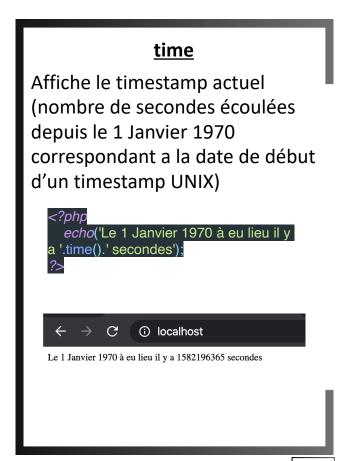
```
/var/www/html/index.php:4:
object(DateTime)[1]
public 'date' = string '2013-03-15 00:00:00.000000' (length=26)
public 'timezone type' => int 3
public 'timezone' => string 'UTC' (length=3)

/var/www/html/index.php:5:
object(DateTime)[2]
public 'date' => string '2013-03-15 00:00:00.000000' (length=26)
public 'timezone type' => int 3
public 'timezone type' => int 3
public 'timezone => string 'UTC' (length=3)
```

Fonctions sur les dates







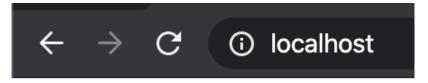
https://www.php.net/manual/fr/ref.datetime.php et pleins d'autres ...

Création de fonctions

- 1) utilisation du mot clé function nomDeLaFonction().
- Une fonction peut prendre 0 ou x paramètres function test (param1, param2, ...)
- 3) Si cette fonction doit retourner une variable, ne pas oublier le mot clé return à la fin de la fonction

Exemple 1. Une fonction qui ne retourne rien mais qui dit bonjour!

```
<?php
function sayHello($nom)
{
    echo 'Hello ' . $nom . ' !<br />';
}
sayHello('Aurélien');
sayHello('Everybody');
```



Hello Aurélien! Hello Everybody!

Création de fonctions

Exemple 2. Une fonction qui jouera le rôle d'une calculette. Elle prendra en entrée 3 paramètres et retournera un résultat!

```
function calculette($operator, $nombre1, $nombre2)
  switch ($operator) {
     case '+':
       return $nombre1 + $nombre2;
       break;
     case '-':
       return $nombre1 - $nombre2;
       break:
     case 1*1.
       return $nombre1 * $nombre2;
       break:
     case '/':
       return $nombre1 / $nombre2;
       break:
var_dump(calculette('+', 3, 3));
var_dump(calculette('*', 10, 12));
var_dump(calculette('/', 23, 4));
```



Include / Include once

Permet d'inclure un fichier PHP dans un autre. Très utilisé pour les raisons suivantes :

- Permet de développer un élément et de le réutiliser ailleurs
- Permet ainsi de gagner en temps de développement
- Permet de gagner en terme de maintenabilité.

Exemple : Mon site internet possède 20 pages différentes. Sur chacune de ses pages, j'ai :

- Un pied de page qui est systématiquement le même
 - Un menu qui sera systématiquement le même

- ...

Je vais donc pouvoir créer 2 fichiers (menu.php et footer.php). Ces derniers seront ajoutés sur toutes mes pages grâce à l'instruction Include!

Quand je souhaiterais modifier mon menu ou mon pied de page, j'aurais juste à modifier le fichier menu.php ou le fichier footer.php

12/17

Include

```
Menu
```

Contenu

Footer

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="utf-8" />
    <title>Un site pratique à utiliser</title>
</head>
<body>
<?php include("block/menu.php"); ?>
<div id="corps">
    <h1>Le contenu spécifique de ma page</h1>
    >
       C'est bien mieux de réutiliser ce que l'on a déjà fait avant !
    </div>
<?php include("block/footer.php"); ?>
</body>
</html>
```



Include

```
🖿 block > 🙌 menu.php >
```

Ici mon menu

- Mes restos préférés
- Les restos à découvrir
- Scanner un code !

Le contenu spécifique de ma page

C'est bien mieux de réutiliser ce que l'on a déjà fait avant !

Ici mon menu

- Lien
- Lien
- Lien



include / include_once

Ici mon menu

- Mes restos préférés
- Les restos à découvrir
- Scanner un code

Ici mon menu

- Mes restos préférés
- Les restos à découvrir
- Scanner un code!

Le contenu spécifique de ma page

C'est bien mieux de réutiliser ce que l'on a déjà fait avant !

Ici mon menu

- Mes restos préférés
- Les restos à découvrir
- Scanner un code!

Le contenu spécifique de ma page

C'est bien mieux de réutiliser ce que l'on a déjà fait avant !

```
<hody>
</php
   include("block/menu.php");
   include("block/menu.php")
?>
<div id="corps">
        <h1>Le contenu spécifique de ma page</h1>

        C'est bien mieux de réutiliser ce que l'on a déjà fait avant !

</div>

</ph>

</ph>

</ph>

<pre
```

require / require_once

Similaire à include. En revanche, si le fichier à inclure n'est pas trouvé, une erreur sera lancée. La ou include se contente d'un warning et affichera le reste de la page, require lancera une erreur et stoppera le script.

Avec require et un fichier inexistant

<?php require("block/unfichierinexistant.php"); ?>

(1) Warning: require(block/unfichierinexistant.php): failed to open stream: No such file or directory in /var/www/html/index.php on line 10						
Call Stack						
#	Time ?	demory	Function	Location	Location	
1	0.0044	359512	{main}()	/index.ph	/index.php:0	
Etal error: require(): Failed opening required 'block/unfichlerinexistant.php' (include_path='_Jusr/local/lib/php') in /var/www/html/index.php on line Call Stack						
#	Time	Memory	Function		Location	
1	0.0044		359512 {main}()		/index.php:0	

Comme include_once, require_once chargera le fichier seulement si ce dernier n'a pas été déjà chargé.

Echanges / Questions

