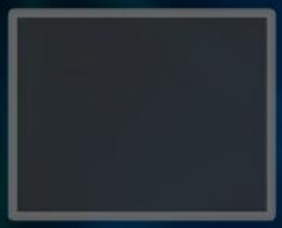
Révision



```
<?php /* des instructions */ ?>

instruction ;
{
    /* bloc d'instructions;
    bien parenthésé;
    */
}
```

Exécution du php :

-> c'est le serveur

/home/toto/public_html/partiel/blabla.php

localhost/~toto/partiel/blabla.php

Débogage du php:

```
//dans un fichier php
var_dump($ma_variable);
//sur la console
tail -f /var/log/apache2/error.log
```

Si var_dump("toto"); n'affiche rien, questions à se poser :

- Le fichier est-il bien parenthésé ?
- Chaque instruction se temine-t-elle bien par un point virgule?

Autres questions pertinantes :

- N'y a-t-il pas une faute de frappe dans les noms de variables/fichiers inclus ?
- est ce que le serveur apache a les droits de lecture/écriture sur les fichiers / repertoire que j'utilise / modifie ?

Tableaux :

```
//On suppose que $tab est un tableau primitif
```

Pour itérer sur tous les éléments d'un tableau:

```
foreach( $tab as $valeur)
{
    //$valeur prend successivement toutes les valeurs du tableau $tab
}
```

Pour connaître le nombre d'éléments d'un tableau :

```
count($tab);
```

Pour créer un nouveau tableau:

```
$ma_variable = array();
```

Pour ajouter un élément à la fin du tableau:

```
$ma_variable[] = "nouvel élément";
```

```
//On suppose que $tab2 est un tableau associatif (ie
qui contient des paires de clefs/valeurs
```

Pour itérer sur tous les éléments d'un tableau associatif:

```
foreach( $tab2 as $clef => $valeur)
{
    //$clef reçoit successivement toutes les clefs du tableau $tab
    //$valeur reçoit la valeur associé à la clef $clef
}
```

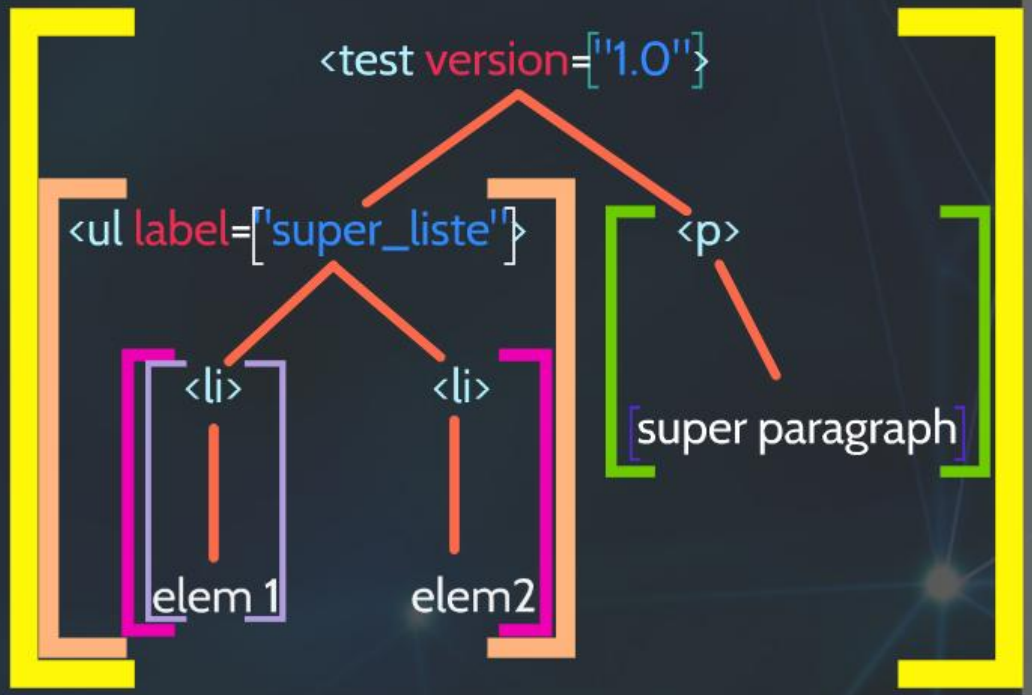
Pour ajouter un élément dans le tableau associatif:

```
$ma_variable["maSuperClef"] = "maSuperValeur";
```

Xml :

```xml
<?xml version='1.0' encoding='utf-8' ?>
<test version="1.0">
    <ul label="super_liste">
        <li> elem 1 </li>
        <li> elem 2</li>
    </ul>
    <p> super paragraph </p>
</test>
```

```php
$toto = simplexml_load_file("file.xml");

$mon_ul = $toto->ul;

$mon_paragraph = $toto->p;

$contenu_paragraph = (string)$toto->p;

$tab_li = $toto->ul->li;
        = $mon_ul->li;

$li0 = $tab_li[0];
     = $mon_ul->li[0];
     = $toto->ul->li[0];

$version_de_test = (string) $toto["version"];

$label_de_ul = (string) $mon_ul["label"];
             = (string) $toto->ul["label"];
```



itérer sur tous les li de ul:

```php
foreach($mon_ul->li as $li){ /* do some stuff*/}
foreach($toto->ul->li as $li){ /* do some stuff*/}
foreach($tab_li as $li){ /* do some stuff*/}
```

Formulaires :

```
<form method='post' action='auth.php?cleSansVal&test=true'>
    <input type="text" name="cleV1"/>
    <input type="password" name="pwd"/>
    <button type="submit">BlaBlaBla</button>
</form>
```

Dans auth.php les variables $_GET et $_POST sont automatiquement remplis avec les données du formulaire par le serveur apache

array_key_exists("cleSansVal", $_GET) vaut true
array_key_exists("test", $_GET) vaut true
array_key_exists("pwd", $_GET) vaut false
array_key_exists("cleV1", $_POST) vaut true
array_key_exists("pwd", $_POST) vaut true

$_GET["cleSansVal"] vaut ""
$_GET["test"] vaut "true"
$_POST["cleV1"] vaut "some_random_text"
$_POST["pwd"] vaut "p@sx2w*R$"

Ecrire en haut du fichier auth.php la ligne suivante est une bonne idée.

```
<?php var_dump($_GET); var_dump($_POST); ?>
```