
php

# PHP: Langage de script pour le Web

- Qu’est-ce que PHP ?
  - Langage de script interprété par le serveur,
  - Orienté objet.
  - Produit du code HTML.
  - Syntaxe proche du C/C++,
  - Créé en 1994-1995 par Rasmus Lerdorf
  - Langage multi plate-forme (UNIX / Windows…) et Open Source
  - Versions actuelles: - PHP5
    - PHP7

# PHP

- Compris entre les balises `<?php` et `?>`
- Les instructions terminent par ;
- L'instruction ***`echo`*** permet de décrire une chaîne de caractère contenant du code HTML. Ce code sera inséré dans la page HTML envoyée au client.
- On ajoute $ devant le nom de la variable
- le type d'une variable :

  - Est implicite (pas besoin de déclarer une variable).
  - Dépend de la valeur qu'on lui affecte (peut varier dans le temps).
- PHP permet de définir des constantes a l'aide de la fonction **`define`**.

- La fonction ***isset*** détermine si une variable est définie et est différente de **NULL** tandis que la fonction ***unset*** permet de supprimer la variable.

**Exemple:**

```php
<?php
$welcome_text = "Welcome to PHPPOT";
unset($welcome_text);
if(isset($welcome_text)) {
echo "<strong>My welcome text is: </strong><i>" . $welcome_text . "</i>";
} else {
echo "<strong>My welcome text is: </strong> empty";
}
?>
```

- La fonction ***empty()*** détermine si une variable est considérée comme vide. Une variable est considérée comme vide si elle n'existe pas, ou si sa valeur équivaut à **FALSE** ou **0** ou "".

```php
<?php
$a=0;
echo " isset : ". isset($a).'</br>'; // affiche:  isset : 1
echo "empty: ". empty($n).'</br>'; // affiche: empty: 1
?>
```

- Avec le quotte double " les variables contenues dans cette chaîne sont interprétées.

```php
<?php
$nom= "PHP";
  echo "Hello, $nom"."</br>";      // Hello, PHP
  echo 'Hello, $nom';         // Hello, $nom
?>
```

- ***var_dump (parm1,parm2, …):*** affiche les informations structurées d'une variable(ou plusieurs), y compris son type et sa valeur.
- ***print_r (parm1, parm2):*** affiche des informations lisibles pour une variable. Lorsque ce ***parm2*** vaut **TRUE**, ***print_r ()*** retournera l'information plutôt que de l'afficher.

```php
$b = 4.1;
$c = "hello";
var_dump($b,$c);//float(4.1) string(5) "hello"
echo "<br>";

print_r($b);//4.1
echo "<br>";

$x=print_r($c,true); // $x contient l'affichage de print_r
print_r($x);//hello
```

# LES TABLEAUX

1. Pour créer un tableau en PHP, on utilise généralement la fonction ***array*.**

```php
<?php
$tab1=$arrayName = array(12,"fraise",2.5);
echo "$tab1[2] </br>"; // 2.5
```

2. Vous pouvez aussi créer manuellement le tableau case par case

```php
$tab3[0]=12;
$tab3[1]="fraise";
$tab3[2]=2.5;
echo "$tab3[0] </br>"; // 12
```

3. Vous pouvez laisser PHP sélectionner le numéro de la case automatiquement en laissant les crochets vides

```php
$tab2[]=12;
$tab2[]="fraise";
$tab2[]=2.5;
echo "$tab2[1]"; // fraise
?>
```

- Lorsqu'un tableau est créé, on peut le parcourir avec la boucle **for**. On peut utiliser un autre type de boucle plus adapté aux tableaux, **foreach**. Avec **foreach**, lors de chaque passage, elle va mettre la valeur de cette ligne dans une variable temporaire.

```php
$tab= array(12,"fraise",2.5);
foreach($tab as $v)
{
  echo "Val: $v<br>";
}
```

# Tableaux associatifs

- Les tableaux associatifs fonctionnent sur le même principe, sauf qu'au lieu de numéroter les cases, on va les étiqueter en leur donnant à chacune un nom différent.
- Pour en créer, on utilise aussi la fonction **array**, mais on va mettre « *l'étiquette* » devant chaque information, sous la forme d'une paire *key => value*

```php
// On crée notre array $coordonnees
$coordonnees = array (
    'prenom' => 'Amine',
    'nom' => 'Nasri',
    'adresse' => '3 Rue du Paradis',
    'ville' => 'Setif');
// ajouter un element!
$coordonnees["Tel"]=057000001;
var_dump($coordonnees);
```

```
array(5) {
  ["prenom"]=>
  string(5) "Amine"
  ["nom"]=>
  string(5) "Nasri"
  ["adresse"]=>
  string(16) "3 Rue du Paradis"
  ["ville"]=>
  string(5) "Setif"
  ["Tel"]=>
  int(12320769)
}
```

- Les tableaux associatifs peuvent contenir des clés de type integer et string en même temps

```php
$array = array(
    1    => "a",
    "2"  => "b",
    "ville" => "setif",
    "age" => 25,
    3 => "c",
);

var_dump($array);
```

```
array(5) {
  [1]=>
  string(1) "a"
  [2]=>
  string(1) "b"
  ["ville"]=>
  string(5) "setif"
  ["age"]=>
  int(25)
  [3]=>
  string(1) "c"
}
```

- Il est possible de spécifier la clé seulement pour quelques éléments et ne pas la fournir pour d'autres

```php
$array = array(
            "a",
            "b",
        "Secret" => "c",
            "d",
          7 => "e",
            "f"
);
var_dump($array);
```

```
array(6) {
  [0]=>
  string(1) "a"
  [1]=>
  string(1) "b"
  ["Secret"]=>
  string(1) "c"
  [2]=>
  string(1) "d"
  [7]=>
  string(1) "e"
  [8]=>
  string(1) "f"
}
```

- Pour modifier une valeur en particulier, il convient d'assigner une valeur en spécifiant sa clé. Pour effacer une paire clé/valeur, on peut appeler la fonction **`unset()`** sur la clé désirée

```php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56;    // Identique à $arr[13] = 56;
                // à cet endroit du script

$arr["x"] = 42; // Ceci ajoute un nouvel élément au
                // tableau avec la clé "x"
var_dump($arr); // array(4) { [5]=> int(1) [12]=> int(2) [13]=> int(56) ["x"]=> int(42) }
echo "<br>";
unset($arr[5]); // Ceci efface l'élément du tableau
var_dump($arr); // array(3) { [12]=> int(2) [13]=> int(56) ["x"]=> int(42) }

unset($arr);    // Ceci efface complètement le tableau
```

- On peut parcourir un tableau associatif avec la boucle **foreach** comme ceci :

```php
// On crée notre array $coordonnees
$coordonnees = array (
    'prenom' => 'Amine',
    'nom' => 'Nasri',
    'adresse' => '3 Rue du Paradis',
    'ville' => 'Setif');
// ajouter un element!
$coordonnees["Tel"]=057000001;

foreach($coordonnees as $element)
{
    echo $element . '<br />';
}
// or
foreach($coordonnees as $cle => $element)
{
    echo '- '.$cle . ' est: ' . $element . '<br />';
    /*
    - prenom est: Amine
    - nom est: Nasri
    - adresse est: 3 Rue du Paradis
    - ville est: Setif
    - Tel est: 12320769
    */
}
```


AJAX
Asynchronous Javascript And XML

# AJAX

- AJAX est un ensemble de techniques préexistantes qui dépend essentiellement de XMLHttpRequest, un objet coté client utilisable en JavaScript
- XMLHttpRequest est un objet JavaScript qui permet d'obtenir des données au format XML, JSON, mais aussi HTML, ou encore texte simple à l'aide de requêtes HTTP.
- Ajax permet de modifier partiellement la page affichée par le navigateur pour la mettre à jour sans avoir à recharger la page entière. Par exemple le contenu d'un champ de formulaire peut être changé, sans avoir à recharger la page avec le titre, les images, le menu, etc.
- Le terme asynchrone, signifie que l'exécution de JavaScript continue sans attendre la réponse du serveur qui sera traitée quand elle arrivera. Tandis qu'en mode synchrone, le navigateur serait gelé en attendant la réponse du serveur.

- AJAX peut par exemple vous suggérer des valeurs comme le fait *google*

# FONCTIONNEMENT

1. Instancier un objet XMLHttpRequest;  XHR = **new** XMLHttpRequest();

2. Une fois que nous avons un objet de type XMLHttpRequest, nous allons pouvoir envoyer une requête au serveur. L'objet XHR dispose de deux méthodes:

   - open: établit une connexion.
   - send: envoie une requête au serveur.

- open s'utilise de cette façon : open(*Method*, *Url*, *aAsync*)

**a.** **Method** indique à l'objet XHR la méthode qui devra être utilisée pour la requête. Laissez le nom de la méthode en majuscules comme spécifié par la norme HTTP

**b.** **Url** pour indiquer l'adresse du programme chargé de traiter notre requête, ça peut être une page dynamique (PHP, ASP) ou une page statique (TXT, HTML...)

- ✓ Si la méthode **GET** est utilisée dans **sMethod**, on peut indiquer la liste des paramètres que nous souhaitons passer à celui-ci.
- ✓ Avec **POST**, il faut spécifier les variables dans l'argument de la méthode send. Il faut aussi changer le type **MIME** de la requête avec la méthode setRequestHeader , sinon le serveur ignorera la requête

```
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
```

**c.** **aAsync** définit si le mode de transfert est asynchrone ou synchrone.

- send: envoie une requête au serveur.

```
/* instancier un objet XHR*/
var xhr = getXMLHttpRequest();
/*ou */
var xhr = new XMLHttpRequest();
/* avec GET*/
xhr.open("GET", "page.php", true);
xhr.send();
/* GET avec des params*/
xhr.open("GET", " page.php?fname=Henry&lname=Ford", true);
xhr.send();
/* avec POST*/
xhr.open("POST", "page.php", true);
xhr.send();
/* POST avec des params*/
xhr.open("POST", "page.php", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.send("fname=Henry&lname=Ford");
```

Quand on envoie une requête HTTP via XMLHttpRequest, celle-ci passe par plusieurs états différents. l'état est donné par la propriété readyState de XMLHttpRequest:

**0** : L'objet XHR a été créé, mais pas encore initialisé (la méthode open n'a pas encore été appelée)

**1** : Connexion établie avec le serveur

**2** : Requête reçue

**3** : réponse en cours

**4** : Toutes les données sont réceptionnées

Pour détecter les changements d'état nous utilisons la propriété onreadystatechange qui définit une fonction à exécuter lorsque le readyState change

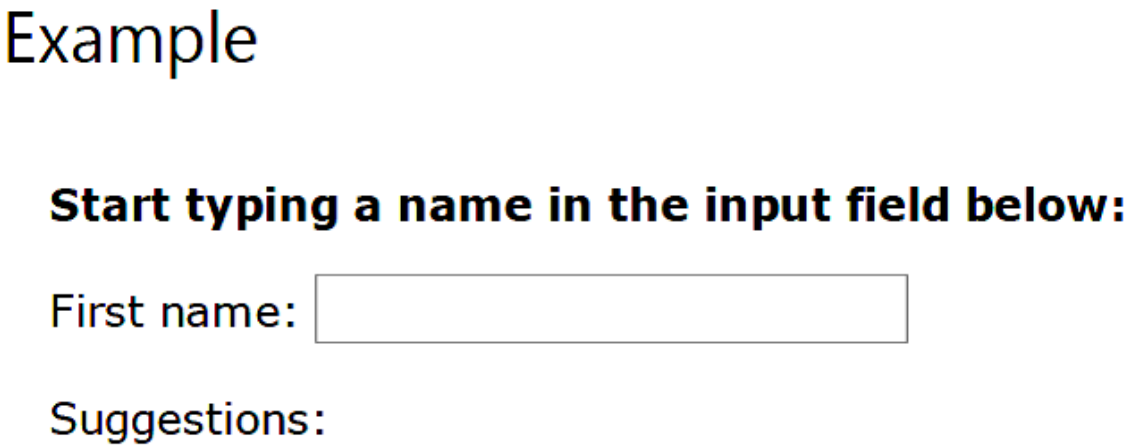
L'objet XMLHttpRequest permet d'interagir avec le serveur, grâce à ses méthodes et ses propriétés:

- Propriétés:

1. **readyState**: C'est cette propriété qu'on va tester dans le onreadystatechange. Elle représente l'état de l'objet et peut prendre des valeurs entre 0 et 4
2. **status:** Le code de la réponse du serveur (200:requête réussie, 404: page non trouvée, …
3. **responseText**: La réponse retournée par le serveur, au format texte.
4. **responseXML**: La réponse retournée par le serveur, au format **XML**.

```
<div>
  <h2 id="demo">Let AJAX change this text</h2>
</div>
<button type="button" onclick="loadDoc()">Change Content</button>
<script>
function loadDoc() {
  var xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
     document.getElementById("demo").innerHTML = this.responseText;}
  };
  xhr.open("GET", "reponse.txt", true);
  xhr.send();
}
</script>
```

- L'exemple suivant montre comment une page Web peut communiquer avec un serveur Web lorsqu'un utilisateur tape des caractères dans un champ de saisie:




```
<p>
    <b>Start typing a name in the input field below:</b>
</p>
<form>
    First name: <input type="text" onkeyup="showHint(this.value)">
</form>
    <p>Suggestions: <span id="txtHint"></span></p>
```

```
<script>
function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("txtHint").innerHTML = this.responseText;
            }
        };
        xmlhttp.open("GET", "gethint.php?q=" + str, true);
        xmlhttp.send();
    }
}
</script>
```

```php
<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
$a[] = "Diana";
$a[] = "Eva";
$a[] = "Fiona";
$a[] = "Gunda";
$a[] = "Hege";
$a[] = "Inga";
$a[] = "Johanna";
$a[] = "Kitty";
$a[] = "Linda";
$a[] = "Nina";
$a[] = "Ophelia";
$a[] = "Petunia";
$a[] = "Amanda";
$a[] = "Raquel";
$a[] = "Cindy";
$a[] = "Doris";
$a[] = "Eve";
$a[] = "Evita";
$a[] = "Sunniva";
$a[] = "Tove";
$a[] = "Unni";
$a[] = "Violet";
$a[] = "Liza";
```

```php
// get the q parameter from URL
$q = $_REQUEST["q"];

$hint = "";

// lookup all hints from array if $q is different from ""
if ($q !== "") {
    $q = strtolower($q);
    $len=strlen($q);
    foreach($a as $name) {
        //The stristr() function searches for the first occurrence
        // of a string inside another string, and return the rest of the string
        //echo stristr("Hello world!","wor"); result: world!
        //substr: Retourne le segment de $string défini par $start et $length.
        //$returnValue = substr('ajax', 1, 2); result: ja
        if (stristr($q, substr($name, 0, $len))) {
            if ($hint === "") {
                $hint = $name;
            } else {
                $hint .= ", $name";
            }
        }
    }
}

// Output "no suggestion" if no hint was found or output correct values
echo $hint === "" ? "no suggestion" : $hint;
```