Récupérer les valeurs d'un formulaire

Quand l'un de vos visiteurs entre les informations dans un formulaire, celle−ci sont récupérées sous forme de variables.
Le nom de ces variables dépend de la méthode d'envoi du formulaire.
Comme dans notre exemple suivant la méthode d'envoi est POST, il faut mettre comme nom $_POST['nom_du_champ'].
Pour les anciens qui exploitaient les variables de facon $nom_du_champ au lieu de $_POST['nom_du_champ'], je conseille de lire de toute urgence le tutoriel de flyingcow sur les variables globales à OFF et surtout d'arrêter de coder ainsi.
Cette variable contient ce qu'a entré le visiteur dans le champ, oops :). Allez, un exemple me paraît plus simple, ci−dessous le name="**nom**" devient **$_POST['nom']** et name="**prenom**" devient **$_POST['prenom']**, il ne reste plus qu'à faire un print() des variables et le tour est joué !
Pour simplifier le nom des variables, dans notre exemple, on fait $nom = $_POST['nom']
et $prenom = $_POST['prenom'] pour assigner la valeur de la variable $_POST['prenom'] a $prenom et idem pour $_POST['nom']
(attention un nom de variable ne doit pas contenir d'espace et ne doit pas commencer par un chiffre alors n'en mettez pas dans vos nom de champ).

Le code HTML du formulaire (ne copiez/collez pas ce code dans votre éditeur, retapez−le ou gare aux erreurs...)	**Donne comme résultat à l'écran**
`<html><body>` `<form method="post" action="verif.php">` `Nom : <input type="text" name="nom"` `size="12"> ` `Prénom : <input type="text"` `name="prenom" size="12">` `<input type="submit" value="OK">` `</form></body></html>`	Nom: Prénom:
Le code PHP de verif.php (ne copiez/collez pas ce code dans votre éditeur, retapez−le ou gare aux erreurs...)	**Donne comme résultat à l'écran après envoi "OK"**

`<?php`
`$prenom = $_POST['prenom'];`
`$nom = $_POST['nom'];`
`print("<center>Bonjour $prenom $nom</center>");`
`?>`
	Bonjour Thaal Rasha

Il va bien sûr maintenant falloir contrôler les informations que rentre le visiteur pour éviter au maximum les erreurs. La première fonction que nous utiliserons est empty(), qui permet de contrôler si un champs est vide. Ensuite nous allons contrôler que **$_POST['url']** commence bien par http:// à l'aide des deux fonctions strtolower() et substr().

Le code HTML du formulaire (ne copiez/collez pas ce code dans votre éditeur, retapez−le ou gare aux erreurs...)	**Donne comme résultat à l'écran**
`<html><body>` `<form method="post" action="verif.php">` `Titre : <input type="text" name="titre"` `size="12"> ` `URL : <input type="text" name="url"` `size="12" value="http://">` `<input type="submit" value="OK">` `</form></body></html>`	Titre: URL:
Le code PHP de verif.php (ne copiez/collez pas ce code dans votre éditeur, retapez−le ou gare aux erreurs...)	**Donne comme résultat à l'écran après envoi "OK"**

`<?php`
`$titre = $_POST['titre'];`
`$url = $_POST['url'];`
`if(empty($titre))`
`{`
`print("<center>Le '<b>Titre</b>' est`
`vide !</center>");`
	Erreur n°1 :
	Le '**Titre**' est vide !
	Erreur n°2 : L'URL doit commencer par **http://**
	Si pas d'erreur : phpdebutant : http://www.phpdebutant.org

```
exit();
}
// vérification du début de l'url
$verif_url = strtolower($url);
$verif_url = substr("$verif_url", 0, 7);
// on verifie les 7 premiers caractères
if ($verif_url!="http://")
{
print("L'URL doit commencer par
<b>http://</b>");
exit();
}
else
{
print("$titre :
<a href=\"$url\">$url</a>");
}
?>
```

Avec cet exemple nous commençons à attaquer les conditions, c'est un aspect primordial dans tous les langages. La première vérification porte sur le champ 'titre', la fonction empty() permet de contrôler si celui−ci est vide ou non. Ce qui nous donne :

- **if(empty($titre)){** print(" Le 'Titre' est vide ! "); exit(); **} :** Si la variable **$titre** est vide alors j'affiche le message : 'Le titre est vide' *(placé entre accolades)* et j'arrête l'exécution du reste du code avec la commande exit().
- Par contre si la variable n'est pas vide, l'exécution ne prend pas en compte ce qui se trouve entre accolades et continue.

La seconde vérification est plus fine puisqu'il s'agit de vérifier que les 7 premiers caractères qui ont été entrés par le visiteur sont bien **http://**. Pour commencer nous utilisons la fonction strtolower() qui permet de transformer tous les caractères en minuscules (ex. HTTP://www.MONsite.CoM devient http://www.monsite.com). Puis à l'aide de la fonction substr(), nous sélectionnons les 7 premiers caractères *(**0** est toujours le premier caractère d'une chaine − le second chiffre ' **7** ' étant le nombre de caractères à sélectionner)*, puis nous les comparons à ce que nous avons dans notre condition if :

- **if ($verif_url!="http://"){** print("L'URL doit commencer par http://"); exit(); **} :** Si les 7 premiers caractères sont différents (signe: **!=**) de http://, alors on exécute ce qui ce trouve entre accolades *(en l'occurrence on affiche un message d'erreur)*, puis nous arrêtons le reste du code avec la commande exit().
- Par contre si le résultat est correct, PHP ignore ce qui ce trouve entre accolades et exécute le reste du code.

Vous pourrez faire autant de tests que vous voudrez sur les champs, mais ne soyez pas trop draconien car les visiteurs n'aiment pas trop que l'on empiète sur leur liberté :). Les contrôles les plus fréquents s'effectuent sur les URL et email pour savoir si l'email comporte bien un "@" et un point.

Le code HTML du formulaire (ne copiez/collez pas ce code dans votre éditeur, retapez−le ou gare aux erreurs...)	**Donne comme résultat à l'écran**

`<html><body>`
`<form method="post" action="verif.php">`
`Votre email : <input type="text"`
`name="email" size="20">`
`<input type="submit" value="OK">`
`</form></body></html>`
	Votre email:

Le code PHP de verif.php (ne copiez/collez pas ce code dans votre éditeur, retapez−le ou gare aux erreurs...)	**Donne comme résultat à l'écran après envoi "OK"**

`<?php`
`$email = $_POST['email'];`
`$point = strpos($email,".");`
`$aroba = strpos($email,"@");`
	Erreur n°1 : Votre email doit comporter un **point** !
	Erreur n°2 : Votre email doit comporter un '**@**' !
	Si pas d'erreur : Votre email est : **email@email.com**

```php
if($point=='')
{
echo "Votre email doit comporter un
<b>point</b>";
}
elseif($aroba=='')
{
echo "Votre email doit comporter un
<b>'@'</b>";
}
else
{
echo "Votre email est: '<a
href=\"mailto:"."$email"."\"><b>$email</b></a>'";
}
?>
```

Comme son nom l'indique, la fonction strpos() retourne la position d'un caractère dans une chaîne si celui−ci existe, autrement strpos() retourne "rien". C'est ce que nous utilisons pour savoir si les point et @ sont bien présents dans l'email.

Exemple : Si strpos() retourne "10" cela veut dire que le premier caractère recherché est placé juste après les 10 premiers caractères donc en 11e position dans la chaîne, puisque vous devez toujours vous rappeler que php commence à compter à **0** et non pas **1**.