***Le PHP :***

Le PHP dans une page HTML est appelé via une balise :

<?php

?>

Pour afficher quelque chose grâce à PHP, on utilisera le mot clef ***echo .***

<?php

echo "Bienvenue dans le monde du PHP";

?>

Le console.log qu’on utilise sur JS pour vérifier une variable est différente sur PHP.

En effet, il faut utiliser ***var-dump(contenu à tester) ;***

Contrairement à JS, il n’y a pas de console et le résultat apparaîtra directement sur la page.

Dès lors qu’il y a du PHP dans une page, alors cette page devra avoir l’extension .php et non .html .

Pour concaténer, en JS on va utiliser +.

***En PHP on va utiliser un point.***

echo "bonjour "." monsieur";

On peut afficher avec des guillemets et des apostrophes.

Toutefois, les apostrophes vont rendre un texte littéral alors que les guillemets vont interpréter les variables ou autres qui sont à l’intérieur :

$var1='Hello DWWM';

echo '$var1';

echo '<br>';

echo "$var1";

// Ici, le premier $var1 va être affiché comme écrit alors que le deuxième entre guillemets va appliquer

// la variable et lire Hello DWWM.

On peut mettre un nombre entier (integer) en variable. Pour un float (nombre à virgule), il faut séparer les chiffres par un point :

$x = 147;

$y = 147.8;

Pour les entiers, on peut aller de -2 000 000 000 à + 2 000 000 000. Pour les float, on peut aller bien plus loin.

Les variables :

Pour une variable, il suffit simplement de mettre le signe $ :

$x=10;

Les variables booléenes (true or false) :

Le booléen que l’on écrit dans JS marche de la même manière dans PHP.

// Ici, on vient dire que 5 est supérieur à 2.

$x = (5>2);

// On vient tester si 5 est supérieur à 2 alors affiche vrai, sinon faux. Comme 5 est

// effectivement supérieur à 2 alors il va afficher true.

if($x){

echo "true";

}

else {

echo "false";

}

Les incrémentations :

Comme sur JS, les incrémentations seront différentes selon l’ordre dans lequel on met les signes.

$x=10;

// Ici, serveur va afficher 10 et ensuite lui ajouter 1.

echo $x++ . "<br>";

$t=10;

// Ici, il va d'abord ajouter 1 et ensuite afficher le résultat, ce qui donnera 11.

echo ++$t;

Pour les tests en if/else ou autre (les for, …), il n’y a aucune différence avec javascript.

Les fonctions :

Elles s’écrivent comme dans JS :

function bienvenue(){

echo 'bienvenue chez nous!';

}

bienvenue();

function bienvenue($x){

echo 'bienvenue '.$x.' !'.'<br>';

}

bienvenue('Marc');

bienvenue('Marie');

Les variables externes dans les fonctions :

Ici, contrairement à JS, il y a une commande pour utiliser une variable extérieure à la fonction :

$e=["marc","sophie","marie","paul"];

function tableau($x){

// Ici, le global va préciser que la variable e extérieure à la fonction va être utilisée.

// Si on ne le précise pas, il ne connaitra pas la variable e.

global $e;

echo 'bienvenue '.$e[$x].'<br>';

}

tableau(0);

tableau(1);

tableau(2);

tableau(3);

Il est donc possible d’utiliser un nom de variable propre à une fonction qui n’existera que dans cette fonction. Bien évidemment, il faut toujours avoir des noms différents pour éviter les erreurs.

Une variable appelée en global, si elle est modifiée dans une fonction, restera modifiée après être passée dans la fonction. Il faut prendre ça en considération pour éviter les erreurs.

Les modifications de variables :

Quand on appelle une variable, on ne fait qu’afficher son résultat ou s’en servir pour arriver à un résultat. La variable en elle-même ne change pas. Il est possible de changer l’adresse mémoire d’une variable et donc de modifier son résultat  avec le signe & :

function foo(&$var) {

$var++;

}

$a=5;

foo ($a);

// $a vaut 6 maintenant

Les paramètres par défaut :

On peut rentrer des variables qui ont un paramètre par défaut dans une fonction. Cependant,

Les paramètres par défaut doivent toujours être les derniers de la liste.

// Ici, l'âge par défaut est donc 11 ans.

function agg($nom,$age=11){

echo "nom: ".$nom." , âge =".$age. " ans. <br>";

}

agg("marie");

agg("max");

agg("solange");

// Ici, on veut modifier ce qui va être affiché après âge = donc on vient le préciser.

agg("pierre",47);

Les tableaux :

Il existe deux types de tableaux : les tableaux indéxés et les tableaux associatifs.

Les tableaux indexés sont pareil que sur JS. On va appeler tel élément grâce à [x].

Ici, au lieu de .length on utilisera le mot count.

Les tableaux associatifs sont différents. On va associer une valeur à un mot, afin que l’on puisse appeler la valeur en mettant le mot entre crochets.

// Créer un tableau:

$x= array();

// ou

$x=[];

// Pour le créer.

// LES TABLEAUX INDEXES:

// Ils se créent comme dans JS, il faut ajouter les valeurs dans un ordre précis et appeler un

// numéro précis pour avoir la valeur correspondante. Comme partout, la première valeur du tableau est 0.

// Pour ajouter une valeur à un tableau à la suite:

$x[count($x)]="toto";

// Ici, le count remplace le .length et vient directement compter le tableau et ajouter à la suite.

// Pour ajouter à une place en particulier ou remplacer une valeur:

$x[2]="Jean";

// Ici, on cible la donnée que l'on veut remplacer.

// LES TABLEAUX ASSOCIATIFS:

// Ici, on va associer une valeur à un mot afin de ressortir la valeur en appellant le mot:

// La flèche => va définir que telle mot renvoie à telle valeur.

$z=["nom" =>"Maureau", "prenom" => "Maurice", "age" => 38];

// Si on veut sortir le prénom:

echo $z["prenom"];

Les tableaux associatifs renvoient à une personne en particulier. Il est possible de stocker plusieurs tableaux associatifs dans un tableau indexé par exemple.

Les implode, explode et str\_split :

Il esiste des variables pour agir sur les tableaux :

// IMPLODE:

// Implode permet de créer un string à partir d'un tableau et les séparer par le premier paramètre

// que l'on rentre:

$x=["marc","sophie","jean","louise","hector"];

// var\_dump est l'équivalent php du console.log.

var\_dump(implode(",",$x));

echo "<br>";

// EXPLODE:

// Explode permet de créer un array à partir d'une string. Le premier paramètre va permettre de dire quel caractère va permettre

// de séparer la string:

$str="un,deux,trois,quatre";

// Ici, il va chercher les virgules dans la string et saparer ce qu'il y a entre les virgules.

var\_dump(explode(",",$str));

echo "<br>";

// STR\_SPLIT:

// str\_split va prendre une string et la diviser tout les x éléments, définis en second paramètre pour la transformer en array:

$str2="abcdefghijklmnopqrstuvwxyz";

// ici, l'alphabet sera divisé toutes les 5 lettres et chaque portion équivaudra à une valeur du tableau.

var\_dump(str\_split($str2,5));

Parcourir un tableau associatif :

$x=["nom"=>"marc","prenom"=>"pierre","age"=>21];

// ici, on vient dire que pour chaque valeur du tableau $x

// as $k(nom de valeur au pif qui fait référence au mot key) => $v(nom de valeur au pif qui fait référence au mot value)

// on renvoie le nom de la clé et la valeur qu'elle représente.

foreach ($x as $k=>$v){

echo"key=".$k." valeur =".$v."<br>";

}

On utilisera le mot clef foreach pour aller chercher chaque key et la value à laquelle elle rapporte.

Les constantes :

Ce sont des valeurs que l’on utilisera dans tout le programme et que l’on ne pourra jamais modifier.

Ce sont les seules variables qui ne prennent pas de $. Par convention, on les écrit en majuscules pour les différencier.

// Pour definir une constante:

define("SOCIETE","Microsoft");

// ici, à chaque fois qu'on appellera SOCIETE alors Microsoft ressortira.Il faut impérativement respecter les majuscules.

// Il est possible de créer une variable qui n'est pas case sensitive:

define("SOCIETE","Microsoft",true);

// avec le true, même si on écrit societe sans majuscules alors ça marchera.

Afficher un tableau lui-même contenu dans un tableau :

// Ici, on a un tableau. Dans ce tableau est stocké un autre tableau qui lui-même contient un autre tableau.

$x=array(1,2,3,array(10,20,30,array(100,200,300)));

// Pour afficher les données des tableaux internes:

// On va créer une fonction qui va lire les tableaux et les afficher: On met un paramètre pour ensuite l'utiliser avec un

// tableau en paramètre.

function returnArray($tab){

// Pour chaque élément ($v) contenu dans le tableau ($tab):

foreach($tab as $v){

// Si l'élément v est un tableau:

if(is\_array($v)){

// Ici, la fonction s'appelle elle-même pour dire: "affiche les éléments contenus dans le tableau". On met $v en

// paramètre pour en afficher les éléments si c'est un tableau.

// Comme on est rentré dans le tableau interne (grâce au if),alors tout ce qui est contenu dans le tableau

// va s'afficher.

returnArray($v);

}

// Si l'élément v n'est pas un tableau, alors on va l'afficher.

else{

echo $v."<br>";

}

}

}

// Ici, on va appeler la fonction et l'utiliser sur le tableau $x. Les éléments du tableau qui sont eux-même des tableaux

// vont s'afficher à la suite puisque la fonction est une boucle qui va traiter le cas où les valeurs sont un tableau.

returnArray($x);

La fonction sans les commentaires pour la réutiliser :

function returnArray($tab){

if(is\_array($v)){

returnArray($v);

}

else{

echo $v."<br>";

}

}

}

returnArray($x);

Tester des tableaux associatifs stockés dans un tableau indexé :

function returnArray($tab)

{

// Ici, pour chaque tableau construit de manière associative (donc construit de manière $k=>$v):

foreach ($tab as $k => $v) {

// Si $v est un tableau:

if (is\_array($v)) {

// On utilise la fonction avec $v en paramètre pour afficher ce qu'il y a dans ce tableau:

returnArray($v);

} else {

// Montre de quelle manière afficher les éléments qui ne sont pas des tableaux:

echo $k . " " . $v;

}

}

}

returnArray($x);

Sans les commentaires :

function returnArray($tab)

{

foreach ($tab as $k => $v) {

if (is\_array($v)) {

returnArray($v);

} else {

echo $k . " " . $v;

}

}

}

returnArray($x);

Le return :

Comme dans JS, on utilise return pour retourner un résultat après un test de fonction. Le return permet de sortir de la fonction. Il est donc inutile d’écrire quelque chose dans une fonction après un return, puisque s’il est appliqué alors on sort de la fonction et ce qui vient ensuite ne sera jamais lu par l’ordinateur.

// Ici, on vient dire que si le premier paramètre est plus grand que le deuxième alors on retourne le premier,

// sinon le deuxième. Si le premier est plus grand alors le return fait directement sortir de la fonction.

function maxab($a,$b){

if($a>$b){

return $a;

}

else{

return $b;

}

}

$c= maxab(2,7);

echo $c;

// Ici, le viewport va donc afficher 7.

Les formulaires :

***La différence entre post et get :***

Le get va transférer les infos via l’URL (donc les mdp également aussi, il faut faire attention). Alors que le post va envoyer les informations via un header caché.

Pour récupérer les infos d’un formulaire :

<form action="test\_get.php" method="get">

<label >Nom</label> <input type="text" name="nom"> <br>

<label >Prenom</label> <input type="text" name="prenom"> <br>

<label >Age</label> <input type="text" name="age"> <br>

<input type="submit" value="envoyer"><br>

</form>

***Si on veut reprendre les infos en get :***

// Le $\_GET est une variable super globale. Elle est toujours la même et générée via PHP. Il faut donc l'écrire

// telle qu'elle.

$nom=$\_GET['nom'];

$prenom=$\_GET['prenom'];

$age=$\_GET['age'];

echo "votre nom est: <b><u>".$nom."</u></b> <br>";

echo "votre prenom est: <b><u>".$prenom."</u></b> <br>";

echo "votre âge est: <b><u>".$age."</u></b> <br>";

?>

***Si on veut reprendre les infos en post :***

<!-- On met évidemment le method="post" en HTML. Le action = « » fait référence au fichier PHP qui va créer les variables. -->

<form action="bienvenue.php" method="post">

<label >Nom</label> <input type="text" name="nom"> <br>

<label >Prenom</label> <input type="text" name="prenom"> <br>

<label >Age</label> <input type="text" name="age"> <br>

<input type="submit" value="envoyer"><br>

</form>

***$\_POST est également une super globale.***

$nom=$\_POST['nom'];

$prenom=$\_POST['prenom'];

$age=$\_POST['age'];

echo "votre nom est: <b><u>".$nom."</u></b> <br>";

echo "votre prenom est: <b><u>".$prenom."</u></b> <br>";

echo "votre âge est: <b><u>".$age."</u></b> <br>";

Un autre exemple avec un text-area ;

<form action="bienvenue.php" method="get">

<textarea name="text" id="" cols="30" rows="10"></textarea><br>

<input type="submit" value="envoyer"><br>

</form>

Le fichier de sortie « bienvenue.php » va afficher le texte entré dans le text area. On le verra également dans l’URL parce qu’on utilise la méthode GET.

$text=$\_GET["text"];

echo $text;

Compter la longueur d’une string :

function longueurChaine($a){

// strlen permet de compter la longueur d'une string.

echo strlen( $a);

}

$alphabet="abcdefghijklmnopqrstuvwxyz";

longueurChaine($alphabet);

Remplacer un caractère ou une chaîne de caractères dans une string :

Il est possible de remplacer le contenu d’une string grâce à la fonction str\_replace :

$alphabet="abcdefghijklmnopqrstuvwxyz";

function remplaceCaractere($a){

// 1/ Le premier paramètre va définir ce que l'on veut remplacer.

// 2/ Le second ce par quoi on veut le remplacer.

// 3/ Enfin le dernier va servir à définir dans quelle string ou array on veut le remplacer.

echo str\_replace("e","\_",$a);

}

remplaceCaractere($alphabet);

Les strings en majuscule ou minuscule :

function listVal($tab)

{

foreach ($tab as $k => $v) {

// strtoupper va afficher le paramètre en majuscules. strtolower les affichera en minuscules.

echo strtoupper($k) ." ". strtolower($v) . "(" . strlen($v) . ") <br>";

}

}

listVal($tableau);

Obtenir l’adresse ip :

Pour obtenir une adresse IP, il faut utiliser la super globale $\_SERVER :

echo $\_SERVER['REMOTE\_ADDR'] ;