Εισαγωγή

Τι είναι η PHP; Η PHP είναι μια γλώσσα προγραμματισμού που σχεδιάστηκε για τη δημιουργία *δυναμικών σελίδων* στο δυαδίκτυο και είναι επισήμως γνωστή ως: HyperText Preprocessor. Είναι μια *server-side scripting γλώσσα* (εκτελείτε στον διακομιστή) που γράφεται συνήθως πλαισιωμένη από HTML, για μορφοποίηση των αποτελεσμάτων. Αντίθετα από μια συνηθισμένη HTML σελίδα η σελίδα PHP δεν στέλνεται άμεσα σε έναν πελάτη (client), αντί αυτού πρώτα αναλύεται και μετά αποστέλλεται το παραγόμενο αποτέλεσμα. Τα στοιχεία HTML στον πηγαίο κώδικα μένουν ως έχουν, αλλά ο PHP κώδικας ερμηνεύεται και εκτελείται. Ο κώδικας PHP μπορεί να θέσει ερωτήματα σε βάσεις δεδομένων, να δημιουργήσει εικόνες, να διαβάσει και να γράψει αρχεία, να συνδεθεί με απομακρυσμένους υπολογιστές , κ.ο.κ. Σε γενικές γραμμές οι δυνατότητες που μας δίνει είναι απεριόριστες.

Σύνταξη

Η σύνταξη της PHP: Ένα κομμάτι PHP πάντα ξεκινάει με το **<?php** και τελειώνει με **?>** και μπορεί να τοποθετηθεί οπουδήποτε στη σελίδα. Υπάρχουν 2 βασικές δηλώσεις στη PHP, η **echo** και η **print**. Στο παράδειγμα που ακολουθεί χρησιμοποιούμε τη δήλωση echo για να εξάγουμε το κείμενο "Hello World".

```
<html>
<body>
	<?php
		echo "Hello World";
	?>
</body>
</html>
```

Σημείωση: Το αρχείο θα πρέπει να έχει κατάληξη **.php**, εάν η κατάληξη του αρχείου είναι .html, ο PHP κώδικας δε θα εκτελεστεί.

Σχόλια στην PHP: χρησιμοποιούμε το // για να κάνουμε σχόλια μιας γραμμής και το /* και */ για να κάνουμε μεγαλύτερα κομμάτια σχολίων.

```
<html>
<body>
	<?php
		//This is a comment

		/*
		This is
		a comment block
		*/
	?>
</body>
</html>
```

Μεταβλητές στην PHP: Οι μεταβλητές χρησιμοποιούνται για την αποθήκευση τιμών, όπως κείμενο, αριθμούς ή πίνακες. Όταν καθορίζονται οι μεταβλητές, μπορούν να χρησιμοποιηθούν ξανά και ξανά μέσα στο script μας. Όλες οι μεταβλητές στην PHP ξεκινάνε με το σύμβολο **$**. Παραλείποντας το σύμβολο $, ο κώδικας δε θα τρέξει σωστά.

```
$var name = value;
```

Παράδειγμα:

```
<?php
$txt="Hello World!";
$x=16;
?>
```

- ✓ Το όνομα μιας μεταβλητής θα πρέπει να ξεκινάει με ένα γράμμα ή με την κάτω παύλα "_"
- ✓ Το όνομα μιας μεταβλητής θα πρέπει να περιέχει μόνο αλφαριθμητικούς χαρακτήρες και κάτω παύλες (a-z, A-Z, 0-9, and _)
- ✓ Το όνομα μιας μεταβλητής δε θα πρέπει να περιέχει κενά. Εάν το όνομα μιας μεταβλητής είναι περισσότερες από μία λέξεις, αυτές θα πρέπει να χωρίζονται με κάτω παύλες ($my_string), ή με κεφαλαία γράμματα ($myString)

String μεταβλητές στην PHP: Οι μεταβλητές τύπου string χρησιμοποιούνται για τιμές που περιέχουν χαρακτήρες. Ένα string μπορεί να χρησιμοποιηθεί απευθείας μέσα σε μία συνάρτηση ή μπορεί να αποθηκευτεί μέσα σε μία μεταβλητή. Στο παράδειγμα που ακολουθεί, το PHP script καταχωρεί το κείμενο "Hello World" σε μία μεταβλητή τύπου string που ονομάζεται $txt:

```
<?php
$txt="Hello World";
echo $txt;
?>
```

Η έξοδος του κώδικα είναι:

```
Hello World
```

Ο τελεστής συνένωσης: Υπάρχει μόνο ένα τελεστής συνένωσης string στην PHP και είναι ο (.) που χρησιμοποιείται για να τοποθετήσει δύο τιμές string μαζί:

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

Η έξοδος του κώδικα είναι:

```
Hello World! What a nice day!
```

Η συνάρτηση strlen(): χρησιμοποιείται για να επιστρέψει το μήκος ενός string:

```
<?php
echo strlen("Hello world!");
?>
```

Η έξοδος του κώδικα είναι:

```
12
```

Η συνάρτηση strpos(): χρησιμοποιείται για να αναζητήσει χαρακτήρα/κείμενο σε ένα string. Αν βρεθεί κάτι που ταιριάζει με την αναζήτηση, η συνάρτηση θα επιστρέψει τη θέση του χαρακτήρα του πρώτου ταιριάσματος. Εάν κανένα ταίριασμα δε βρεθεί, η συνάρτηση θα επιστρέψει ΨΕΥΔΕΣ (FALSE). Στο παράδειγμα που ακολουθεί αναζητούμε τη λέξη "world" μέσα στο string μας:

```
<?php
echo strpos("Hello world!","world");
?>
```

Η έξοδος του κώδικα είναι:

```
6
```

Σημείωση: Ο λόγος που η θέση του string "world" στο παράδειγμα είναι 6 (και όχι 7) είναι επειδή η πρώτη θέση χαρακτήρα του string είναι το 0, και όχι το 1.

Τελεστές

Αριθμητικοί Τελεστές

Τελεστής	Περιγραφή	Παράδειγμα	Αποτέλεσμα
+	Πρόσθεση	x=2; x+2	4
-	Αφαίρεση	x=2; 5-x	3
*	Πολλαπλασιασμός	x=4; x*5	20
/	Διαίρεση	15/5; 5/2	3; 2.5
%	Modulus (Υπόλοιπο διαίρεσης)	5%2; 10%8; 10%2	1; 2; 0
++	Αύξηση	x=5; x++	x=6
--	Μείωση	x=5; x--	x=4

Ανάθεση Τελεστών

Τελεστής	Παράδειγμα	Αντίστοιχα
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

Τελεστές Σύγκρισης

Τελεστής	Περιγραφή	Παράδειγμα
==	ισούται με	5==8 επιστρέφει ψευδές
===	ισούται ακριβώς με (τιμή και τύπος)	5!=8 επιστρέφει αληθές
!=	δεν ισούται με	5<>8 επιστρέφει αληθές

>	είναι μεγαλύτερο με	5>8 επιστρέφει ψευδές
<	είναι μικρότερο με	5<8 επιστρέφει αληθές
>=	είναι μεγαλύτερο ή ίσο με	5>=8 επιστρέφει ψευδές
<=	είναι μικρότερο ή ίσο με	5<=8 επιστρέφει αληθές

Λογικοί Τελεστές

Τελεστής	Περιγραφή	Παράδειγμα
&&	και	x=6; y=3; (x < 10 && y > 1) αληθές
||	ή	x=6; y=3; (x==5 || y==5) ψευδές
!	διάφορο	x=6; y=3; !(x==y) αληθές

Δηλώσεις If...else

Πολύ συχνά όταν γράφουμε κώδικα, θέλουμε να δημιουργήσουμε διαφορετικές ενέργειες για διαφορετικές αποφάσεις. Έτσι, μπορούμε να χρησιμοποιούμε δηλώσεις καταστάσεων στον κώδικά μας για να το κάνει αυτό. Στην PHP έχουμε τις ακόλουθες δηλώσεις καταστάσεων:

- ✓ *if*
- ✓ *if...else*
- ✓ *if...else if....else*
- ✓ *switch*

Δήλωση If: Χρησιμοποιούμε τη δήλωση if για την εκτέλεση κάποιου κώδικα εάν η κατάσταση που εξειδικεύεται είναι αληθής:

```
if (condition) code to be executed if condition is true;
```

Παράδειγμα: Στο ακόλουθο παράδειγμα θα εμφανιστεί ως έξοδος το "Have a nice weekend!" εάν η τρέχουσα μέρα είναι η Παρασκευή:

```
<html>
<body>
        <?php
                $d=date("D");
                if ($d=="Fri") echo "Have a nice weekend!";
        ?>
</body>
</html>
```

Δήλωση If...else: Χρησιμοποιούμε τη δήλωση if....else για την εκτέλεση κάποιου κώδικα εάν η κατάσταση είναι αληθής και κάποιου άλλου κώδικα εάν η κατάσταση δεν είναι αληθής:

```
if (condition)
 code to be executed if condition is true;
else
 code to be executed if condition is false;
```

Παράδειγμα 1: Στο ακόλουθο παράδειγμα θα εμφανιστεί ως έξοδος το "Have a nice weekend!" εάν η τρέχουσα μέρα είναι η Παρασκευή, διαφορετικά θα εμφανιστεί ως έξοδος το "Have a nice day!":

```
<html>
<body>
        <?php
                $d=date("D");
                if ($d=="Fri")
                 echo "Have a nice weekend!";
                else
                 echo "Have a nice day!";
        ?>
</body>
</html>
```

Παράδειγμα 2: Εάν περισσότερες από μία γραμμές κώδικα θα πρέπει να εκτελεστούν αν μία συνθήκη είναι αληθής/ψευδής, οι γραμμές θα πρέπει να εσωκλείονται μέσα σε κατάλληλες αγκύλες:

```
<html>
```

```
<body>
        <?php
                $d=date("D");
                if ($d=="Fri") {
                   echo "Hello!<br />";
                   echo "Have a nice weekend!";
                   echo "See you on Monday!";
                 }
        ?>
</body>
</html>
```

Δήλωση If...else if...else: Χρησιμοποιούμε τη δήλωση if....else if....else για την επιλογή ενός από τα πολλά κομμάτια κώδικα προς εκτέλεση:

```
if (condition)
 code to be executed if condition is true;
else if (condition)
 code to be executed if condition is true;
else
 code to be executed if condition is false;
```

Παράδειγμα: Στο ακόλουθο παράδειγμα θα εμφανιστεί ως έξοδος το "Have a nice weekend!" εάν η τρέχουσα μέρα είναι η Παρασκευή, και "Have a nice Sunday!" εάν η τρέχουσα μέρα είναι η Κυριακή, διαφορετικά θα εμφανιστεί ως έξοδος το "Have a nice day!":

```
<html>
<body>
        <?php
                $d=date("D");
                if ($d=="Fri")
                  echo "Have a nice weekend!";
                elseif ($d=="Sun")
                  echo "Have a nice Sunday!";
                else
                  echo "Have a nice day!";
        ?>
</body>
</html>
</html>
```

Δήλωση Switch

Δήλωση switch: Χρησιμοποιούμε τη δήλωση switch για την επιλογή ενός από τα πολλά κομμάτια κώδικα προς εκτέλεση. Αρχικά έχουμε μία απλή έκφραση *n* (συνήθως είναι μία μεταβλητή), η οποία είναι προς αξιολόγηση. Η τιμή της έκφρασης συγκρίνεται με τις τιμές για κάθε περίπτωση της δομής. Εάν υπάρξει κάποιο ταίριασμα, το κομμάτι κώδικα που αντιπροσωπεύει αυτή την περίπτωση εκτελείται. Χρησιμοποιούμε τη **break** για να εμποδίσουμε τον κώδικα να τρέξει την επόμενη περίπτωση αυτόματα. Οι προκαθορισμένες δηλώσεις χρησιμοποιούνται εάν δεν υπάρξει κανένα ταίριασμα.

```
switch (n)
{
case label1:
 code to be executed if n=label1;
 break;
case label2:
 code to be executed if n=label2;
 break;
default:
 code to be executed if n is different from both label1 and label2;
}
```

Παράδειγμα:

```
<html>
<body>
        <?php
                switch ($x) {
                        case 1:
                          echo "Number 1";
                          break;
```

```
                    case 2:
                      echo "Number 2";
                      break;
                    case 3:
                      echo "Number 3";
                      break;
                    default:
                      echo "No number between 1 and 3";
                }
        ?>
</body>
</html>
```

Πίνακες

Τι είναι ένας πίνακας; Μία μεταβλητή είναι ένας αποθηκευτικός χώρος που κρατάει έναν αριθμό ή ένα κείμενο. Το πρόβλημα είναι ότι μία μεταβλητή μπορεί να κρατήσει μία μόνο τιμή. Ένας πίνακας (array) είναι μία ειδική μεταβλητή, η οποία μπορεί να αποθηκεύσει πολλαπλές τιμές σε μία μόνο μεταβλητή. Εάν έχουμε μία λίστα από αντικείμενα, για παράδειγμα μία λίστα από ονόματα αυτοκινήτων, το να αποθηκεύαμε τα αυτοκίνητα σε απλές μεταβλητές θα ήταν κάπως έτσι:

```
$cars1="Saab";
$cars2="Volvo";
$cars3="BMW";
```

Παρόλα αυτά, τι θα γινόταν εάν θα θέλαμε να κάνουμε μία εκλογή μεταξύ των αυτοκινήτων και να βρίσκαμε κάποιο συγκεκριμένο; Και σκεφτείτε τι θα γινόταν εάν δεν είχαμε 3 αυτοκίνητα, αλλά 300! Η καλύτερη λύση εδώ είναι η χρήση πίνακα! Ένα πίνακας μπορεί να κρατήσει όλες τις τιμές των μεταβλητών κάτω από το ίδιο όνομα, και μπορούμε να έχουμε πρόσβαση στις τιμές κάνοντας αναφορά στο όνομα του πίνακα. Κάθε στοιχείο στον πίνακα έχει ένα δικό του μοναδικό ID ώστε να είναι εύκολα προσβάσιμο.

Στην PHP υπάρχουν 3 είδη πινάκων:

- ✓ Αριθμητικοί πίνακες
- ✓ Πίνακες συσχετίσεων
- ✓ Πολυδιάστατοι πίνακες

Αριθμητικοί πίνακες: Ένας αριθμητικός πίνακας αποθηκεύει κάθε στοιχείο του πίνακα μαζί με έναν αριθμό ευρετηρίου. Υπάρχουν 2 μέθοδοι για δημιουργία ενός αριθμητικού πίνακα:

1. Στο ακόλουθο παράδειγμα το ευρετήριο καθορίζεται αυτόματα (το ευρετήριο ξεκινάει από το 0):

```
$cars=array("Saab"
```

2. Στο ακόλουθο παράδειγμα εμείς καθορίζουμε το ευρετήριο "χειροκίνητα":

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

Παράδειγμα: Στο ακόλουθο παράδειγμα έχουμε πρόσβαση στις τιμές της μεταβλητής κάνοντας αναφορά στο όνομα του πίνακα και του ευρετηρίου:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

Η έξοδος του κώδικα είναι:

```
Saab and Volvo are Swedish cars.
```

Πίνακες συσχετίσεων: Σε ένα πίνακα συσχετίσεων, κάθε κλειδί ID συνδέεται με μία τιμή. Όταν θέλουμε να αποθηκεύονται δεδομένα για συγκεκριμένες ονομαζόμενες τιμές, ένας αριθμητικός πίνακας δεν είναι πάντα η καλύτερη λύση. Με τους πίνακες συσχετίσεων μπορούμε να χρησιμοποιούμε τις τιμές σαν κλειδιά και να συνδέουμε τις τιμές με αυτά.

Παράδειγμα 1: Στο παράδειγμα που ακολουθεί χρησιμοποιούμε έναν πίνακα για να συνδέσουμε ηλικίες με διαφορετικούς ανθρώπους:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

Παράδειγμα 2: Στο παράδειγμα που ακολουθεί, είναι όπως και στο προηγούμενο, αλλά δείχνει έναν διαφορετικό τρόπο δημιουργίας του πίνακα:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

Τα κλειδιά ID μπορούν να χρησιμοποιηθούν σαν script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

Η έξοδος του κώδικα είναι:

```
Peter is 32 years old.
```

Πολυδιάστατοι πίνακες: Σε έναν πολυδιάστατο πίνακα, κάθε στοιχείο στον κύριο πίνακα μπορεί επίσης να είναι ένας πίνακας, και κάθε στοιχείο στον υποπίνακα μπορεί να είναι ένας πίνακας, κ.ο.κ.

Παράδειγμα 1: Σε αυτό το παράδειγμα δημιουργούμε έναν πολυδιάστατο πίνακα με αυτόματη αντιστοίχιση ID κλειδιών:

```
$families = array
 (
 "Griffin"=>array
 (
 "Peter",
 "Lois",
 "Megan"
 ),
 "Quagmire"=>array
 (
 "Glenn"
 ),
 "Brown"=>array
 (
 "Cleveland",
 "Loretta",
 "Junior"
 )
 );
```

Η έξοδος του κώδικα είναι:

```
Array
(
[Griffin] => Array
  (
  [0] => Peter
  [1] => Lois
  [2] => Megan
  )
[Quagmire] => Array
  (
  [0] => Glenn
  )
[Brown] => Array
  (
  [0] => Cleveland
  [1] => Loretta
  [2] => Junior
  )
)
```

Παράδειγμα 2: Ας προσπαθήσουμε να δείξουμε μόνο μία τιμή από τον προηγούμενο πίνακα:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

Η έξοδος του κώδικα είναι:

```
Is Megan a part of the Griffin family?
```

Βρόγχοι επανάληψης

Συχνά, όταν γράφουμε κώδικα, θέλουμε το ίδιο κομμάτι κώδικα να τρέχει ξανά και ξανά σε μία γραμμή. Αντί να προσθέτουμε πάρα πολλές και σχεδόν ίδιες γραμμές κώδικα σε ένα script, μπορούμε να χρησιμοποιήσουμε βρόγχους επανάληψης ώστε να εκτελείται η εργασία μας με αυτό τον τρόπο. Στο JavaScript, υπάρχουν 2 είδη βρόγχων επανάληψης:

- ✓ *while*
- ✓ *do…while*
- ✓ *for*
- ✓ *foreach*

Ο βρόγχος επανάληψης While: Επαναλαμβάνει ένα κομμάτι κώδικα όσο η συνθήκη που εμπεριέχει είναι αληθής.

```
while (condition)
 {
```

```
 code to be executed
 }
```

Παράδειγμα:

```
<html>
<body>

<?php
$i=1;
while($i<=5)
 {
 echo "The number is " . $i . "<br />";
 $i++;
 }
?>

</body>
</html>
```

Η έξοδος του κώδικα είναι:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

Ο βρόγχος επανάληψης do...while: Είναι μία παραλλαγή της επανάληψης while, η οποία εκτελεί ένα κομμάτι κώδικα ΜΟΝΟ ΜΙΑ ΦΟΡΑ, και έπειτα το επαναλαμβάνει όσο η συνθήκη που εμπεριέχει είναι αληθής.

```
do
 {
 code to be executed
 }
while (condition);
```

Παράδειγμα:

```
<html>
<body>
        <?php
                $i=1;
                do {
                        $i++;
                        echo "The number is " . $i . "<br />";
                 } while ($i<=5);
        ?>
</body>
</html>
```

Η έξοδος του κώδικα είναι:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

Ο βρόγχος επανάληψης For: Χρησιμοποιείται όταν γνωρίζουμε εκ των προτέρων πόσες φορές θα τρέξει το script.

```
for (init; condition; increment)
 {
 code to be executed;
 }
```

Παράμετροι:

- ✓ init: συνήθως χρησιμοποιείται για να θέσει έναν μετρητή
- ✓ condition: αξιολογείται σε κάθε επανάληψη του βρόγχου
- ✓ increment: συνήθως χρησιμοποιείται για να αυξήσει το μετρητή

Σημείωση: Κάθε μία από τις παραπάνω παραμέτρους μπορεί να είναι κενή ή να έχει πολλαπλές εκφράσεις που να χωρίζονται απλά με κόμμα.

Παράδειγμα:

```
<html>
<body>
        <?php
                for ($i=1; $i<=5; $i++) {
                        echo "The number is " . $i . "<br />";
                 }
        ?>
</body>
</html>
```

Η έξοδος του κώδικα είναι:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

Ο βρόγχος επανάληψης foreach: Χρησιμοποιείται ως βρόγχος επανάληψης μέσα σε πίνακες. Για κάθε επανάληψη του βρόγχου, η τιμή του τρέχοντος στοιχείου του πίνακα αντιστοιχίζεται στο $value – και ο δείκτης του πίνακα μετακινείται κατά μία θέση- έτσι στην επόμενη επανάληψη, ελέγχουμε την επόμενη τιμή του πίνακα.

```
foreach ($array as $value)
 {
 code to be executed;
 }
```

Παράδειγμα: Το ακόλουθο παράδειγμα δείχνει έναν βρόγχο επανάληψης που θα τυπώνει τις τιμές του δοθέντα πίνακα:

```
<html>
<body>
        <?php
                $x=array("one","two","three");
                foreach ($x as $value) {
                        echo $value . "<br />";
                 }
        ?>
</body>
</html>
```

Η έξοδος του κώδικα είναι:

```
one
two
three
```

Συναρτήσεις

Για να συγκρατήσουμε το script ώστε να μην εκτελεστεί όταν φορτώνει η σελίδα, μπορούμε να το τοποθετήσουμε αυτά μέσα σε μία συνάρτηση. Με συνάρτηση θα εκτελεστεί μόνο όταν την καλέσουμε. Μπορούμε να καλέσουμε μία συνάρτηση σε οποιοδήποτε σημείο μέσα στη σελίδα.

```
function functionname(var1,var2,...,varX)
{
code to be executed;
}
```

Μεταβλητές ή τιμές που περνάνε μέσα στη συνάρτηση

Αρχή και τέλος της συνάρτησης

Παράδειγμα:

```
<html>
<body>
        <?php
                function writeName() {
                        echo "Kai Jim Refsnes";
                }
                echo "My name is ";
                writeName();
        ?>
</body>
</html>
```

Η έξοδος του κώδικα είναι:

```
My name is Kai Jim Refsnes
```

Πρόσθεση παραμέτρων: Για να δώσουμε περισσότερη λειτουργικότητα σε μία συνάρτηση, μπορούμε να προσθέσουμε παραμέτρους. Μία παράμετρος είναι σαν μία μεταβλητή. Οι παράμετροι καθορίζονται μετά το όνομα της συνάρτησης μέσα σε παρενθέσεις.

Παράδειγμα 1:

```
<html>
<body>
        <?php
                function writeName($fname) {
                        echo $fname . " Refsnes.<br />";
                }
                echo "My name is ";
                writeName("Kai Jim");
                echo "My sister's name is ";
                writeName("Hege");
                echo "My brother's name is ";
                writeName("Stale");
        ?>
</body>
</html>
```

Η έξοδος του κώδικα είναι:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

Παράδειγμα 2:

```
<html>
<body>
	<?php
		function writeName($fname,$punctuation) {
			echo $fname . " Refsnes" . $punctuation . "<br />";
		}
		echo "My name is ";
		writeName("Kai Jim",".");
		echo "My sister's name is ";
		writeName("Hege","!");
		echo "My brother's name is ";
		writeName("Ståle","?");
	?>
</body>
</html>
```

Η έξοδος του κώδικα είναι:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Stale Refsnes?
```

Επιστροφή τιμών: Για να επιτρέψουμε μία συνάρτηση να επιστρέψει μία τιμή, χρησιμοποιούμε τη δήλωση **return**.

Παράδειγμα:

```
<html>
<body>
	<?php
		function add($x,$y) {
			$total=$x+$y;
			return $total;
		}
		echo "1 + 16 = " . add(1,16);
	?>
</body>
</html>
```

Η έξοδος του κώδικα είναι:

```
1 + 16 = 17
```

Φόρμες & Δεδομένα Χρήστη

Οι PHP μεταβλητές $_GET και $_POST χρησιμοποιούνται για ανάκτηση πληροφορίας από φόρμες, όπως δεδομένων που εισάγει ο χρήστης.

Χειρισμός Φόρμας: Το πιο σημαντικό πράγμα που θα πρέπει να προσέξουμε όταν ασχολούμαστε με HTML φόρμες και PHP, είναι ότι κάθε στοιχείο της φόρμας σε μία HTML σελίδα αυτόματα θα είναι διαθέσιμο στα PHP scripts μας.

Παράδειγμα: Το ακόλουθο παράδειγμα περιέχει μία φόρμα HTML με δύο πεδία κειμένου και ένα κουμπί υποβολής:

```
<html>
<body>
	<form action="welcome.php" method="post">
	Name: <input type="text" name="fname" />
	Age: <input type="text" name="age" />
	<input type="submit" />
	</form>
</body>
</html>
```

Όταν ο χρήστης συμπληρώσει τη φόρμα και κάνει κλικ στο κουμπί υποβολής, τα δεδομένα της φόρμας στέλνονται στο PHP αρχείο που ονομάζεται "welcome.php":

```
<html>
<body>
Welcome <?php echo $_POST["fname"]; ?>!<br />
You are <?php echo $ POST["age"]; ?> years old.
</body>
</html>
```

Η έξοδος του κώδικα είναι:

```
Welcome John!
You are 28 years old.
```

Έλεγχος Φόρμας: Τα δεδομένα του χρήστη θα πρέπει να ελέγχονται στον browser όπου είναι δυνατόν (από client scripts). Ο έλεγχος στον browser είναι πιο γρήγορος και μειώνει το φόρτο στον server. Ο έλεγχος στην πλευρά του server θα πρέπει να εξεταστεί στην περίπτωση που τα δεδομένα του χρήστη θα πρέπει να εισαχθούν σε μία βάση δεδομένων.

Στην PHP, η μεταβλητή **$_GET** χρησιμοποιείται για συλλογή δεδομένων από μία φόρμα μέσω της μεθόδου "**get**".

Η απεσταλμένη πληροφορία από μία φόρμα με τη μέθοδο GET είναι ορατή στον καθένα και έχει περιορισμούς ως προς την ποσότητα της απεσταλμένης πληροφορίας.

Παράδειγμα:

```
<form action="welcome.php" method="get">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

Όταν ο χρήστης κάνει κλικ στο κουμπί "Submit", το URL που αποστέλλεται στο server θα μοιάζει κάπως έτσι:

```
http://www.blablabla.com/welcome.php?fname=Peter&age=37
```

Το αρχείο "welcome.php" μπορεί τώρα να χρησιμοποιείσαι τη μεταβλητή $_GET για να συλλέξει τα δεδομένα της φόρμας:

```
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
```

Πότε χρησιμοποιείται η method="get"; Όταν χρησιμποιείται η method="get" σε HTML φόρμες, όλα τα ονόματα των μεταβλητών και οι τιμές εμφανίζονται στο URL. Η μέθοδος αυτή δε θα πρέπει να χρησιμοποιείται όταν αποστέλλονται κωδικοί ή άλλες ευαίσθητες πληροφορίες!

Στην PHP, η μεταβλητή **$_POST** χρησιμοποιείται για συλλογή δεδομένων από μία φόρμα μέσω της μεθόδου "**post**".

Η απεσταλμένη πληροφορία από μία φόρμα με τη μέθοδο POST δεν είναι ορατή στον καθένα και δεν έχει περιορισμούς ως προς την ποσότητα της απεσταλμένης πληροφορίας.

Παράδειγμα:

```
<form action="welcome.php" method="post">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

Όταν ο χρήστης κάνει κλικ στο κουμπί "Submit", το URL που αποστέλλεται στο server θα μοιάζει κάπως έτσι:

```
http://www.blablabla.com/welcome.php
```

Το αρχείο "welcome.php" μπορεί τώρα να χρησιμοποιείσαι τη μεταβλητή $_POST για να συλλέξει τα δεδομένα της φόρμας:

```
Welcome <?php echo $ POST["fname"]; ?>!<br />
You are <?php echo $_POST["age"]; ?> years old.
```

Στην PHP, η μεταβλητή **$_REQUEST** περιέχει το περιεχόμενο των $_GET, $_POST, και $_COOKIE, και μπορεί να χρησιμοποιηθεί για συλλογή δεδομένων παράλληλα από όλες τις άλλες μεθόδους.

Παράδειγμα:

```
Welcome <?php echo $_REQUEST["fname"]; ?>!<br />
You are <?php echo $_REQUEST["age"]; ?> years old.
```