ΣΙΒΙΤΑΝΙΔΕΙΟΣ ΔΗΜΟΣΙΑ ΣΧΟΛΗ ΤΕΧΝΩΝ ΚΑΙ ΕΠΑΓΓΕΛΜΑΤΩΝ

ΣΗΜΕΙΩΣΕΙΣ ΓΙΑ ΤΟ ΚΑΛΟΚΑΙΡΙΝΟ ΣΧΟΛΕΙΟ

«ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΗΛΕΚΤΡΟΝΙΚΟΥ ΕΜΠΟΡΙΟΥ ΓΙΑ ΤΗΝ ΠΡΟΩΘΗΣΗ ΚΑΙ ΠΩΛΗΣΗ ΤΟΠΙΚΩΝ ΠΑΡΑΔΟΣΙΑΚΩΝ ΠΡΟΪΟΝΤΩΝ»

Ι Ο Υ Ν Ι Ο Σ 2 0 0 8

www.kostasdelimaris.gr
Software & Web Developer

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Οι σημειώσεις αυτές συντάχθηκαν από τον Καθηγητή Κωνσταντίνο Δελημάρη και υπεύθυνο προγράμματος του εν λόγω θέματος του καλοκαιρινού σχολείου. Ελπίζουμε ότι θα σας βοηθήσουν στο να λάβετε τις βασικές γνώσεις που χρειάζονται για την ανάπτυξη δυναμικών ιστοσελίδων και εφαρμογών στο Διαδίκτυο.

Τμήματα των σημειώσεων αντλήθηκαν από τον ιστότοπο php.net αλλά και από τον παγκόσμιο ιστό.

Τέλος ο συγγραφέας θα ήθελε να ευχαριστήσει τη Δρ Μαρία Χαλκίδη, η οποία έδωσε την άδειά της ώστε να προστεθούν τμήματα των διαλέξεών της στο Πανεπιστήμιο Πειραιώς σε αυτό το μικρό σύγγραμα.

<u>Οι υπεύθυνοι καθηγητές:</u>

Δελημάρης Κων/νος

…….. Αναστασία

…….. Ιωάννης

Η ΓΛΩΣΣΑ HTML

Η γλώσσα HTML είναι η γλώσσα που χρησιμοποιείται για την κατασκευή των σελίδων του παγκόσμιου ιστού. Υπάρχουν τρεις διαδοχικές εκδόσεις (1.0, 2.0, 3.0) της HTML που έχουν βασιστεί στα πρότυπα που είχε εκδώσει το W3C (World-Wide Web Consortium). Παράλληλα εταιρείες όπως η Netscape και η Microsoft προέκτειναν με δικές τους αποκλειστικά εντολές το σύνολο των εντολών που υπήρχε, με αποτέλεσμα τη δημιουργία πολλών επιπλέον εντολών που δίνουν στο χρήστη μεγαλύτερη δυνατότητα παρέμβασης στο όλο οπτικό αποτέλεσμα των εμφανιζόμενων
σελίδων.
Έτσι τελικά προτάθηκε η HTML 3.2, η οποία ενσωματώνει πλήθος από τα χαρακτηριστικά που είχαν χρησιμοποιήσει αποκλειστικά και μόνο κάποιοι κατασκευαστές πελατών παγκόσμιου ιστού και προτείνει άλλα νέα. Η HTML 3.2 είναι η τελευταία έκδοση της πιο δημοφιλούς γλώσσας του Web και είναι αυτή η οποία περιγράφεται στη συνέχεια με μεγαλύτερη λεπτομέρεια. Σε επόμενο κεφάλαιο περιγράφεται η Dynamic HTML (D-HTML) που δίνει ακόμη πιο πολλά δυναμικά χαρακτηριστικά για σελίδες παγκόσμιου ιστού.

Προδιαγραφές της HTML V3.2

Η HyperText Markup Language (HTML 3.2) έγινε πρότυπο στις αρχές του 1996. Το νέο αυτό πρότυπο που εκδόθηκε από το W3C (World Wide Web Consortium) έχει γίνει αποδεκτό από τις μεγαλύτερες εταιρείες που δραστηριοποιούνται στο χώρο του Διαδικτύου και του παγκόσμιου ιστού όπως οι IBM, Microsoft, Netscape Communications Corporation, Novell, SoftQuad, Spyglass, Sun Microsystems και άλλες. Η HTML 3.2 περιλαμβάνει εξελιγμένα χαρακτηριστικά όπως πίνακες (tables), applets και ροή του κειμένου γύρω από εικόνες, ενώ είναι πλήρως συμβατή με το πρότυπο της HTML 2.0.

Η δομή των HTML σελίδων

Οι HTML σελίδες αρχίζουν με τη δήλωση <!DOCTYPE> και ακολουθούνται από τις HTML ετικέτες HEAD και BODY ως εξής:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">

<HTML>

<HEAD>

<TITLE>Ο τίτλος της παρούσας σελίδας</TITLE>

.... άλλα στοιχεία που τοποθετούνται στο στοιχείο HEAD ....

</HEAD>

<BODY>

... κύριος κορμός της σελίδας ...

</BODY>

</HTML>

Στην πράξη οι ετικέτες αρχής και τέλους των HTML, HEAD και BODY μπορούν να παραλειφθούν διότι υπονοούνται από τη δήλωση του HTML 3.2 DTD. Κάθε κείμενο που είναι γραμμένο σε HTML 3.2 πρέπει να αρχίζει με τη δήλωση <!DOCTYPE>, η οποία είναι απαραίτητη για το διαχωρισμό του παρόντος κειμένου από κάποιο άλλο παλιότερης έκδοσης. Κάθε κείμενο HTML 3.2 πρέπει να περιέχει την ετικέτα τίτλου. Άρα το μικρότερο HTML 3.2 κείμενο μοιάζει με το ακόλουθο:

<TITLE>Οποιοδήποτε κείμενο</TITLE>

Το στοιχείο HEAD

Περιλαμβάνει την κεφαλή u964 του κειμένου, αλλά είναι δυνατή η παράλειψη των ετικετών αρχής και τέλους για το HEAD.

Το στοιχείο HEAD περιλαμβάνει τις εξής ετικέτες:

TITLE Ορίζει τον τίτλο της σελίδας

ISINDEX Για απλές περιπτώσεις αναζήτησης (βλέπε PROMPT)

BASE Ορίζει τη βάση αρχής για καθορισμό των σχετικών URL

SCRIPT Καθορισμός της scripting γλώσσας

STYLE Καθορισμός Style Sheets

META Καθορισμός ζεύγους πληροφοριών όνομα και τιμή

LINK Ορισμός σχέσεων με άλλα κείμενα

Τα TITLE, SCRIPT and STYLE απαιτούν ετικέτες και αρχής και τέλους. Στα υπόλοιπα, επειδή δεν περιέχουν άλλες ετικέτες ανάμεσά τους, η χρήση ετικετών τέλους απαγορεύεται.

Το στοιχείο TITLE

Κάθε κείμενο σε HTML 3.2 πρέπει να έχει **ακριβώς μία** ετικέτα TITLE στην επικεφαλίδα του HEAD. Παρέχει ένα βοηθητικό τίτλο ο οποίος παρίσταται στον

τίτλο του παραθύρου του απεικονιζόμενου κειμένου στον πελάτη WWW. Οι χαρακτήρες που μπορούν να βρίσκονται στην TITLE είναι οι συμβολικοί χαρακτήρες και οι χαρακτήρες διαφυγής (& και <). Δεν επιτρέπεται η χρήση άλλων ετικετών μέσα στο TITLE.

Παράδειγμα:

<TITLE>Αυτή είναι η προσωπική μου WWW σελίδα</TITLE>

Τα στοιχεία STYLE και SCRIPT

Χρησιμοποιούνται στον καθορισμό style sheets και client-side scripts (όπως θα αναφερθεί στο κεφάλαιο της Dynamic HTML). Οι φυλλομετρητές παγκόσμιου ιστού θα πρέπει να μην απεικονίζουν τα περιεχόμενα των παραπάνω στοιχείων.

Το στοιχείο ISINDEX

Το στοιχείο ISINDEX ορίζει στον φυλλομετρητή ότι θα πρέπει να απεικονίσει στο χρήστη μια γραμμή εισαγωγής δεδομένων για να εισάγει μια συμβολοσειρά προς αναζήτηση. Ο χρήστης μπορεί να εισάγει οποιουδήποτε αριθμό χαρακτήρων. Το χαρακτηριστικό PROMPT μπορεί να χρησιμοποιηθεί για πεδίο εισαγωγής δεδομένων.

Παράδειγμα:

<ISINDEX PROMPT = "Αναζήτηση Λέξης">

Η εννοιολογική σημασία του ISINDEX είναι πλήρως καθορισμένη όταν το URL βάσης είναι ένα πλήρες HTTP URL της μορφής ”http://www.upatras.gr/x.html”. Όταν

ο χρήστης πατά το πλήκτρο enter (return) του πληκτρολογίου, η συμβολοσειρά που στέλνεται στον εξυπηρετητή περιέχει το URL βάσης του παρόντος κειμένου μαζί με την escaped- μορφή της συμβολοσειράς αναζήτησης. Για παράδειγμα αν η συμβολοσειρά που δόθηκε είναι η “Query String” και το URL βάσης είναι το:

http://www.domain.gr

τότε η αίτηση προς τον Web Server που προκύπτει είναι η:

“http://www.domain.gr/?Query+String”

Θα πρέπει να σημειωθεί ότι κάθε κενός χαρακτήρας (Space Character) αντικαθίσταται _____με το χαρακτήρα “+” και ότι εφαρμόζονται οι γνωστοί URL κανόνες

κωδικοποίησης

[1]. Στην πράξη οι χαρακτήρες που μπορούν να εισαχθούν ως συμβολοσειρά αναζήτησης θα πρέπει να είναι σύμφωνα με την κωδικοποίηση Latin-1, διότι δεν υπάρχει μέχρι στιγμής υπάρχον μηχανισμός καθορισμού εναλλακτικού συνόλου χαρακτήρων για την αναζήτηση.

Το στοιχείο BASE

Το στοιχείο BASE δίνει τη δυνατότητα καθορισμού της βάσης των σχετικών URLs. Για παράδειγμα:

<BASE HREF="http://www.domain.gr/dir1/ ">

...

<IMG SRC="images/mypic.jpg">

Η εικόνα αναφέρεται στο εξής πλήρες URL:

http://www.domain.gr/dir1/images/mypic.jpg

Το στοιχείο META

Το στοιχείο META μπορεί να χρησιμοποιηθεί για τον καθορισμό ζευγών ονόματος/τιμής που περιγράφουν τα χαρακτηριστικά του υπερκειμένου όπως: όνομα του συγγραφέα, ημερομηνία λήξης, ένα σύνολο από λέξεις κλειδιά, κλπ. Το χαρακτηριστικό NAME καθορίζει το όνομα της οντότητας, ενώ η παράμετρος CONTENT καθορίζει την τιμή.

Για παράδειγμα:

<META NAME = "Author" CONTENT="Panagiotis B. Kappos">

<META NAME = "Keywords" CONTENT = "University of Patras, CEID Faculty, Computer Science">

Η παράμετρος HTTP-EQUIV μπορεί να χρησιμοποιηθεί στη θέση της παραμέτρου NAME και έχει ιδιαίτερη σημασία για τα κείμενα τα οποία μεταφέρονται μέσω του πρωτοκόλλου HTTP. Οι εξυπηρετητές HTTP μπορούν να χρησιμοποιήσουν την τιμή της παραμέτρου HTTP-EQUIV για τη δημιουργία ενός συμβατού με το RFC 822 header στην HTTP απάντηση[2].

Για παράδειγμα:

<META HTTP-EQUIV="Expires" CONTENT="Wed, 09 Oct 1996 20:20:20 GMT+2">

θα επιστρέψει τον εξής HTTP header:

Expires: Wed, 09 Oct 1996 20:20:20 GMT+2

Μια μεγάλη επίσης χρησιμότητα της παραμέτρου HTTP-EQUIV είναι ο καθορισμός του πότε ένα Caching σύστημα (Browser ή Proxy Server) θα ζητήσει ξανά ένα cached κείμενο. Αν στο παραπάνω παράδειγμα θέταμε μια ημερομηνία του παρελθόντος π.χ. του έτους 1973, τότε η σελίδα του κειμένου δε θα γινόταν cache από κανένα σύστημα.

Το στοιχείο LINK

Το στοιχείο LINK παρέχει έναν ανεξάρτητο τρόπο καθορισμού σχέσεων της παρούσας σελίδας με άλλα κείμενα ή ηλεκτρονικούς πόρους πληροφόρησης. Παρόλο που το LINK είναι μέρος της HTML από την πρώτη έκδοσή της, ελάχιστοι φυλλομετρητές την εκμεταλλεύονται (οι πιο πολλοί απλά την αγνοούν).

Τα στοιχεία LINK μπορούν να χρησιμοποιηθούν με δύο τρόπους:

1) Για επιλεκτική πλοήγηση μέσω μενού όταν το στοιχείο LINK βρίσκεται στην κορυφή του κειμένου.

2) Για να ελέγχουν το πως ένα πλήθος από HTML αρχεία απεικονίζονται ως κείμενα προς εκτύπωση.

 HREF

Καθορίζει το URL που δείχνει τον δεικτοδοτημένο πόρο.

 REL

Η "απ' ευθείας σχέση".

 REV

Η "αντίστροφη σχέση". Έτσι αν υπάρχει ένα link από το κείμενο Α προς το κείμενο Β με REV= relation εκφράζει την ίδια σχέση ενός link από το Β προς το Α με REL=relation. Υπάρχει άλλο ένα είδος της σχέσης REV, όταν REV=made. Αυτή χρησιμοποιείται μερικές φορές για να καθορίσει τον κατασκευαστή της σελίδας, είτε με ένα mailto URL είτε με ένα link στην προσωπική σελίδα του κατασκευαστή.

 TITLE

Μια σύντομη επεξήγηση στον δεικτοδοτημένο πόρο.

Μερικές από τις πιο συχνές σχέσεις που συναντάμε στις σελίδες είναι οι:

 REL=Contents

Το link αναφέρεται σε ένα κείμενο που πρόκειται για του πίνακα περιεχομένων.

 REL=Index

Το link αναφέρεται σε ένα κείμενο που παρέχει ένα δείκτη στο παρόν κείμενο.

 REL=Glossary

Το link αναφέρεται σε ένα κείμενο που παρέχει ένα γλωσσάριο όρων του παρόντος κειμένου.

 REL=Copyright

Το link αναφέρεται σε πληροφορίες πνευματικής ιδιοκτησίας του παρόντος κειμένου.

 REL=Next

Το link αναφέρεται στο επόμενο κείμενο μιας σειράς κειμένων.

 REL=Previous

Το link αναφέρεται στο προηγούμενο κείμενο μιας σειράς κειμένων.

 REL=Help

Το link αναφέρεται σε ένα κείμενο βοηθείας.

 REL=Bookmark

Τα Bookmarks χρησιμοποιούνται για να παρέχουν απευθείας δείκτες σε κείμενα με μεγάλο εύρος πληροφόρησης.

Για παράδειγμα:

<LINK REL="Contents" HREF=toc.html>

<LINK REL="Previous" HREF=doc10.html>

<LINK REL="Next" HREF=doc12.html>

Το στοιχείο BODY

Μεταξύ της ετικέτας αρχής και τέλους BODY περιέχονται πλήθος από HTML στοιχεία όπως επικεφαλίδες (Headings), το στοιχείο ADDRESS, στοιχεία για ορισμό μπλοκ, στοιχεία καθορισμού κειμένου:

Οι παράμετροι για το στοιχείο BODY είναι οι:

 bgcolor

Καθορίζει το χρώμα του φόντου της σελίδας.

 text

Καθορίζει το χρώμα των χαρακτήρων του κειμένου.

 link

Καθορίζει το χρώμα των χαρακτήρων των link που δεν έχει επισκεφτεί ακόμη ο χρήστης.

 vlink

Καθορίζει το χρώμα των χαρακτήρων των link που έχει επισκεφτεί ο χρήστης.

 alink

Καθορίζει το χρώμα των χαρακτήρων των link τη στιγμή που ο χρήστης κάνει click πάνω στο link.

 background

Καθορίζει _____το URL μια εικόνας (είδους .gif ή .jpg) που θα χρησιμοποιηθεί επαναληπτικά (tile) για το σχηματισμό του φόντου της σελίδας

Τα χρώματα θα πρέπει να δίνονται με την RGB μορφή τους σε δεκαεξαδική αναπαράσταση (π.χ. LINK="#AABBCC") ή ως το όνομα του χρώματος με βάση το παρακάτω πίνακα:

Όνομα Δεκαεξαδική αναπαράσταση

Black #000000

Green #008000

Silver #C0C0C0

Lime #00FF00

Gray #808080

Olive #808000

White #FFFFFF

Yellow #FFFF00

Maroon #800000

Navy #000080

Red #FF0000

Blue #0000FF

Purple #800080

Teal #008080

Fuchsia #FF00FF

Aqua #00FFFF

Η δεκαεξαδική RGB μορφή των χρωμάτων

Για παράδειγμα:

<BODY BGCOLOR=WHITE TEXT=BLACK LINK=RED VLINK=TEAL ALINK=GREY>

Block and Text level elements

Τα περισσότερα στοιχεία που τοποθετούνται στο κυρίως σώμα ενός HTML κειμένου μπορούν να ενταχθούν σε δύο κατηγορίες:

Τα **στοιχεία ορισμού περιοχής** (block level elements), δημιουργούν τερματισμούς παραγράφων H1 μέχρι H6, παραγράφους (P), οριζόντιες γραμμές (HR) και τα **στοιχεία ορισμού κειμένου** (text level elements), που μορφοποιούν τα στοιχεία κειμένου όπως: τα B, I και FONT δίνουν έμφαση στο κείμενο, A (συνδέσεις υπερκειμένου), IMG and APPLET (προσαρτημένα αντικείμενα) and BR (Διακοπές Γραμμών). Να σημειωθεί ότι τα περισσότερα στοιχεία ορισμού περιοχής γενικά εμπεριέχουν τα στοιχεία ορισμού κειμένου ή και άλλα στοιχεία ορισμού περιοχής, ενώ τα στοιχεία ορισμού κειμένου μπορούν να εμπεριέχουν μόνο άλλα στοιχεία ορισμού κειμένου.

Υπάρχουν έξι επίπεδα από επικεφαλίδες, από το H1 (το πιο σημαντικό) μέχρι το H6 (το λιγότερο σημαντικό). Είναι απαραίτητη η ύπαρξη της ετικέτας αρχής και τέλους. Οι πιο σημαντικές επικεφαλίδες αναπαρίστανται με μεγαλύτερη οικογένεια χαρακτήρων (font) από τις λιγότερο σημαντικές. Η προαιρετική παράμετρος ALIGN χρησιμοποιείται για να οριστεί ο τύπος ευθυγράμμισης (align) του κειμένου που βρίσκεται ανάμεσα στην επικεφαλίδα αρχής και τέλους.

Για παράδειγμα:

<H1 ALIGN=CENTER> ... κεντραρισμένο κείμενο ... </H1>

Η εξ' ορισμού ευθυγράμμιση είναι προς τα αριστερά, αλλά μπορεί να αλλάξει χρησιμοποιώντας τα στοιχεία DIV ή CENTER.

Το στοιχείο ADDRESS

Το στοιχείο ADDRESS θα πρέπει να έχει ορισμένη την αρχή και το τέλος του και παρέχει πληροφορίες όπως την ταυτότητα του κατασκευαστή του κειμένου, τη διεύθυνση και τηλέφωνό του κατασκευαστή.

Για παράδειγμα:

<ADDRESS>
***Πανεπιστήμιο Πατρών
***
***Τμήμα Η/Υ και Πληροφορικής
***
***Πανεπιστημιούπολη - Ρίο
***
Τηλ. +30 61 997585
</ADDRESS>

Στοιχεία περιοχής

Τα κύρια u963 στοιχεία περιοχής (Block elements) σε ένα υπερκείμενο είναι:

P Παράγραφοι

UL Μη διατεταγμένες λίστες

OL Διατεταγμένες λίστες

DL Λίστες ορισμού

PRE εξ' αρχής μορφοποιημένο κείμενο

DIV Χωρισμοί κειμένου

CENTER Ευθυγράμμιση κειμένου

BLOCKQUOTE quoted παράγραφος

FORM Φόρμες

ISINDEX HTML φόρμες απλού τύπου

HR οριζόντιες γραμμές

TABLE Πίνακες

Τα κύρια στοιχεία περιοχής

Παράγραφοι

Το στοιχείο <P> ορίζει παραγράφους στο κείμενο. Απαιτεί την ετικέτα αρχής, ενώ η ετικέτα τέλους δεν είναι πάντα απαραίτητη, διότι τις περισσότερες φορές το τέλος μπορεί εύκολα να προσδιοριστεί από το λεκτικό αναλυτή.

Για παράδειγμα:

<P>Αυτή είναι η πρώτη παράγραφος.

<P> Αυτή είναι η δεύτερη παράγραφος.

Χρησιμοποιώντας την παράμετρο ALIGN ορίζουμε τον τύπο οριζόντιας ευθυγράμμισης του κειμένου μέσα στα όρια μιας παραγράφου. Σε περίπτωση που δεν καθοριστεί ο τύπος ευθυγράμμισης τότε υπονοείται προς τα αριστερά.

 align=left

Η παράγραφος είναι οριζόντια ευθυγραμμισμένη προς τα αριστερά

 align=center

Η παράγραφος είναι κεντραρισμένη στο παρόν παράθυρο

 align=right

Η παράγραφος είναι οριζόντια ευθυγραμμισμένη προς τα δεξιά

Για παράδειγμα:

<p align=center>Αυτή η παράγραφος είναι κεντραρισμένη

Λίστες

Οι λίστες μπορούν να εμπεριέχουν στοιχεία ορισμού περιοχής, κειμένου ή άλλες ένθετες λίστες

Μη διατεταγμένες λίστες

Οι μη διατεταγμένες λίστες είναι της μορφής:

<UL>

<LI> ... Πρώτο πεδίο

<LI> ... *Δεύτερο πεδίο*

...

</UL>

Οι ετικέτες αρχής και τέλους είναι απαραίτητες. Τα στοιχεία LI χρησιμοποιούνται για να διαχωρίσουν μεταξύ τους τα πεδία της λίστας. Το στοιχείο τέλους LI μπορεί να παραληφθεί. Τα στοιχεία LI μπορούν να περιέχουν ένθετες λίστες. Με χρήση της παραμέτρου COMPACT απαιτούμε από τον φυλλομετρητή να απεικονίσει τη λίστα σε πιο περιεκτική μορφή.

Η παράμετρος TYPE χρησιμοποιείται για να αλλάξει το είδος της εικόνας που χρησιμοποιείται μέσα στα UL και LI. Συγκεκριμένα:

- όταν <li type=disc>
- όταν <li type=square>
- όταν <li type=circle>

Διατεταγμένες λίστες

Οι διατεταγμένες λίστες είναι της μορφής:

<OL>

<LI> ... Πρώτο πεδίο

<LI> ... *Δεύτερο πεδίο*

...

</OL>

Η μεταβλητή START χρησιμοποιείται για να ορίσει την αρχή της ακολουθίας αριθμών (εξ' ορισμού ισούται με 1). Μπορεί να τεθεί ακόμη και αργότερα με τη μεταβλητή VALUE στα LI στοιχεία. Οι δύο παραπάνω μεταβλητές θα πρέπει να

έχουν ακέραιες τιμές. Η μεταβλητή TYPE επιτρέπει τον ορισμό των παρακάτω επιλογών:

Μεταβλητή TYPE Όνομα Είδος επιλογών

1 Αριθμοί 1, 2, 3, ...

A Πεζοί χαρακτήρες a, b, c, ...

A Κεφαλαίοι χαρακτήρες A, B, C, ...

I Πεζοί Ρωμαϊκοί χαρακτήρες i, ii, iii, ...

I Κεφαλαίοι Ρωμαϊκοί χαρακτήρες I, II, III, ...

Η μεταβλητή TYPE

Λίστες ορισμού

Οι λίστες ορισμού είναι της μορφής:

<DL>

<DT> Όνομα όρου

<DD> Ορισμός όρου

...

</DL>

Τα στοιχεία DT περιέχουν στοιχεία ορισμού κειμένου, ενώ τα στοιχεία DD μπορούν να περιέχουν επιπλέον και στοιχεία ορισμού περιοχής εκτός από επικεφαλίδες και το στοιχείο ADDRESS.

Για παράδειγμα:

<DL>

<DT>Ορισμός 1<DD>Αυτός είναι ο ορισμός του πρώτου όρου

<DT>Ορισμός 2<DD>Αυτός είναι ο ορισμός του δεύτερου όρου

</DL>

θα έπρεπε να εμφανιστεί ως εξής:

Ορισμός 1

Αυτός είναι ο ορισμός του πρώτου όρου

Ορισμός 2

Αυτός είναι ο ορισμός του δεύτερου όρου

Με χρήση της παραμέτρου COMPACT απαιτούμε από τον φυλλομετρητή την απεικόνιση σε πιο περιεκτική μορφή.

Στοιχεία DIR και MENU

Τα παραπάνω στοιχεία παρόλο που ήταν μέρος της HTML από την πρώτη έκδοση, οι φυλλομετρητές δεν διαχωρίζουν τα στοιχεία DIR και MENU, αλλά τα απεικονίζουν με μορφή αντίστοιχη με αυτή που χρησιμοποιούν για τα στοιχεία UL. Κανονικά θα έπρεπε να απεικονίζονταν τα στοιχεία DIR ως λίστες καταλόγου με πολλαπλές στήλες και τα στοιχεία MENU ως λίστες καταλόγου μια στήλης.

Εξ' αρχής μορφοποιημένο κείμενο

Το στοιχείο PRE χρησιμοποιείται για να συμπεριλάβει εξ' αρχής μορφοποιημένο κείμενο. Οι πελάτες χρήστη απεικονίζουν το επιλεγόμενο κείμενο με χαρακτήρες σταθερού πλάτους λαμβάνοντας υπόψη την ύπαρξη των κενών χαρακτήρων.

XMP, LISTING και PLAINTEXT

Οι παραπάνω ετικέτες δεν χρησιμοποιούνται πια αλλά οι φυλλομετρητές θα πρέπει να τις υποστηρίζουν για λόγους συμβατότητας προς τα πίσω. Οι δημιουργοί σελίδων θα πρέπει να αποφεύγουν τη χρήση των παραπάνω ετικετών.

DIV και CENTER

Το στοιχείο DIV χρησιμοποιείται για το διαχωρισμό των κειμένων HTML σε περιοχές. Με την παράμετρο ALIGN ορίζεται ο επιθυμητός τύπος ευθυγράμμισης για όλα τα στοιχεία που βρίσκονται ανάμεσα στα όρια αρχής και τέλους του στοιχείου DIV. Δεκτές τιμές είναι οι: LEFT, CENTER ή RIGHT οι οποίες ορίζονται με ίδιο τρόπο όπως στο στοιχείο παραγράφου. Επειδή το στοιχείο DIV ορίζει μια περιοχή (block) κάθε ανοιχτή παράγραφος θα τερματιστεί.

Το στοιχείο CENTER (το οποίο ορίστηκε αυθαίρετα από την Netscape πριν την HTML 3 και διατηρείται και στην HTML 3.2 λόγο της μεγάλης διάδοσης που έχει ο WWW browser της παραπάνω εταιρείας) είναι ταυτόσημο με το DIV με παράμετρο ALIGN=CENTER.

Οι ετικέτες αρχής και τέλους είναι απαραίτητες και στο DIV και στο CENTER.

BLOCKQUOTE

Χρησιμοποιείται για να συμπεριλάβει κομμάτια λόγου, αποφθέγματα ή αναφορές σε άλλες εργασίες. Οι ετικέτες αρχής και τέλους είναι υποχρεωτικές.

FORM

Το στοιχείο φόρμα είναι από τα πιο σημαντικά αλλά συνάμα και δυσκολότερα σημεία κατασκευής δυναμικών σελίδων, που παρέχει χαρακτηριστικά που δεν θα υπήρχαν χρησιμοποιώντας μόνο ετικέτες της HTML. Ένα κείμενο μπορεί να περιέχει πολλαπλές φόρμες. Σε κάθε φόρμα θα πρέπει να ορίζεται με σαφή τρόπο η αρχή και το τέλος της, χρησιμοποιώντας τις ετικέτες αρχής και τέλους. Η πιο απλή φόρμα είναι αυτή που περιέχει το στοιχείο ISINDEX.
Μια φόρμα μπορεί να περιέχει απλά ή πολλαπλά πεδία κειμένου (text fields), radio buttons, checkboxes και μενού.

Πιο συγκεκριμένα:

 action

Ορίζει το URL το οποίο θα χρησιμοποιηθεί από τον εξυπηρετητή για να επεξεργαστεί τα δεδομένα που συμπληρώθηκαν στη φόρμα. Το URL μπορεί να είναι ένα εκτελέσιμο αρχείο που θα χρησιμοποιηθεί από τον εξυπηρετητή (π.χ. “http://www.domain.gr/cgi-bin/doit”) ή ένα Mail-to URL (π.χ. “mailto:takis@domain.gr”)

 method

Όταν η παράμετρος action ορίζει μια HTTP κλήση, η παράμετρος method ορίζει τη μέθοδο αποστολής των περιεχομένων της φόρμας στον εξυπηρετητή. Μπορεί να είναι GET ή POST. Εξ’ ορισμού είναι GET.

 enctype

Καθορίζει τον τύπο κωδικοποίησης των περιεχομένων της φόρμας. Εξ’ ορισμού είναι application/x-www-form-urlencoded.

Οριζόντιος Κανόνας

Οι οριζόντιοι κανόνες (Horizontal Rule) χρησιμοποιούνται για να δείξουν μια εννοιολογική αλλαγή στο κείμενο. Οπτικά είναι μια οριζόντια γραμμή.
Οι παράμετροι του στοιχείου HR είναι:

 align

Ορίζει την ευθυγράμμιση του κανόνα ως προς τα παρόντα όρια του παραθύρου απεικόνισης. Τιμές είναι οι: left, center ή right. Εξ' ορισμού είναι η center.

 noshade

Ο κανόνας δεν θα έχει τρισδιάστατη όψη και θα είναι χωρίς σκιά

 size

Ορίζει το πάχος του κανόνα σε pixel.

 width

Ορίζει το μήκος του κανόνα σε pixel (π.χ. width=100) ή σε ποσοστό μήκους μεταξύ του παρόντος αριστερού και δεξιού άκρου του παραθύρου (π.χ. width="30%"). Εξ' ορισμού είναι width=100%.

Πίνακες

Οι πίνακες είναι της ακόλουθης γενικής μορφής:

<TABLE BORDER=8 CELLSPACING=5 CELLPADDING=2 WIDTH="90%">
<CAPTION> ... Τίτλος Πίνακα ... </CAPTION>
<TR><TD> πρώτο κελί</TD> <TD> δεύτερο κελί</TD></TR>
<TR> ...
...
</TABLE>

Οι παράμετροι των πινάκων είναι προαιρετικές. Εξ΄ ορισμού, οι πίνακες απεικονίζονται χωρίς οπτική απεικόνιση των ορίων τους και αυτόματα προσαρμόζονται ώστε να περιλαμβάνουν τα περιεχόμενα κάθε κελιού. Υπάρχει η δυνατότητα καθορισμού του απόλυτου πλάτους των πινάκων. Οι παράμετροι BORDER, CELLSPACING και CELLPADDING παρέχουν επιπλέον ευκολίες οπτικής απεικόνισης. Οι τίτλοι των πινάκων στο πάνω ή κάτω μέρος καθορίζονται ανάλογα με την παράμετρο ALIGN.

Κάθε γραμμή περιέχεται ανάμεσα στο στοιχείο TR, παρόλο που η ετικέτα τέλους μπορεί να παραληφθεί. Τα κελιά ορίζονται με το στοιχείο TD για δεδομένα και με το TH για τίτλους (header). Όπως και με το TR, τα παραπάνω στοιχεία τέλους μπορούν να παραληφθούν. Τα TH και TD παίρνουν ως παράμετρο τα: ALIGN και VALIGN για ευθυγράμμιση του περιεχομένου του κελιού, ROWSPAN και COLSPAN για κελιά που καλύπτουν περισσότερες από μία σειρά ή στήλη. Ένα κελί μπορεί να περιέχει πλήθος από στοιχεία περιοχής ή κειμένου, καθώς και φόρμες ή άλλους ένθετους πίνακες.

Το στοιχείο TABLE πρέπει πάντα να έχει ετικέτες αρχής και τέλους. Στη συνέχεια παρατίθενται οι παράμετροι που υποστηρίζουν οι πίνακες:

 ALIGN

Παίρνει παραμέτρους μία από τις: (LEFT, CENTER ή RIGHT). Καθορίζει την οριζόντια τοποθέτηση του πίνακα σχετικά με το παρόν αριστερό και δεξί σύνορο. Εξ' ορισμού ισχύει το LEFT, αλλά αυτό μπορεί να αλλαχθεί από κάποιο από τα στοιχεία DIV ή CENTER, αν αυτά συμπεριλαμβάνονται μέσα στον πίνακα..

 WIDTH

Αν παραληφθεί αυτό το όρισμα τότε το πλάτος του πίνακα καθορίζεται αυτόματα από τον φυλλομετρητή. Το όρισμα WIDTH θέτει είτε το απόλυτο πλάτος σε pixel (WIDTH=452) είτε το επί τοις εκατό κενό διάστημα μεταξύ του αριστερού και του δεξιού ορίου (WIDTH=60%).

 BORDER

Το όρισμα αυτό καθορίζει το πλάτος του εξωτερικού συνόρου που θα βρίσκεται τριγύρω από τον πίνακα για έναν ορισμένο αριθμό από pixel (π.χ. BORDER=3). Όταν η παραπάνω τιμή είναι μηδενική δεν εμφανίζεται καθόλου περίγραμμα στον πίνακα.

 CELLSPACING

Καθορίζει το μέγεθος της κενής περιοχής που θα υπάρχει γύρω από κάθε κελί σε σχέση με τα γειτονικά κελιά. Το μέγεθος έχει ως μονάδες τον αριθμό των pixel (π.χ. CELLSPACING=5). Η ίδια τιμή καθορίζει το κενό διάστημα μεταξύ του ορίου του πίνακα με τα όρια των εξωτερικών κελιών.

 CELLPADDING

Καθορίζει το μέγεθος σε pixel μεταξύ του ορίου καθενός κελιού με τα περιεχόμενά του.

Κάθε pixel αναφέρεται στα pixel οθόνης και θα πρέπει να πολλαπλασιαστούν με κατάλληλο συντελεστή όταν αυτό απεικονίζεται σε τερματικά υψηλής ευκρίνειας όπως είναι οι εκτυπωτές σελίδας. Για παράδειγμα, όταν η οθόνη του χρήστη έχει πλάτος 75 pixel/ίντσα και απεικονίζεται σε έναν εκτυπωτή σελίδας 600στιγμών/ίντσα, τότε οι τιμές των pixel δοσμένες σε μονάδες HTML θα πρέπει να πολλαπλασιαστούν με τον παράγοντα 8.

Το στοιχείο CAPTION έχει μόνο μια παράμετρο ALIGN, η οποία καθορίζει την τοποθέτηση του τίτλου του πίνακα και μπορεί να δεχθεί ως τιμές τις ALIGN=TOP

ή ALIGN=BOTTOM. Εξ΄ ορισμού, η τοποθέτηση του τίτλου του πίνακα είναι στο πάνω μέρος.

Τα στοιχεία TR και TD

Το στοιχείο TR απαιτεί μια ετικέτα αρχής, αλλά όχι απαραίτητα την ετικέτα τέλους και εγκλείει τα κελιά του πίνακα. Έχει τις εξής παραμέτρους:

 ALIGN

Καθορίζει την εξ' ορισμού οριζόντια ευθυγράμμιση των περιεχομένων των κελιών. Αποδεκτές τιμές είναι οι: LEFT, CENTER ή RIGHT.

 VALIGN

Καθορίζει την εξ' ορισμού κάθετη ευθυγράμμιση των περιεχομένων των κελιών. Αποδεκτές τιμές είναι οι: TOP, MIDDLE ή BOTTOM για τοποθέτηση των περιεχομένων του κελιού στη κορυφή, τη μέση ή τη βάση του κελιού αντίστοιχα.

Υπάρχουν δύο στοιχεία για τον καθορισμό της μορφής των κελιών. Το TH χρησιμοποιείται για την επικεφαλίδα των κελιών, ενώ το TD για τα δεδομένα (περιεχόμενα) των κελιών. Οι επικεφαλίδες αρχής TH και TD είναι απαραίτητες, αλλά οι επικεφαλίδες τέλους μπορούν να παραληφθούν. Τα κελιά μπορούν να έχουν τα ακόλουθα χαρακτηριστικά:

 NOWRAP

Η παρουσία αυτής της παραμέτρου απενεργοποιεί την αυτόματη τύλιξη κειμένου μέσα στο χώρο ενός κελιού (π.χ. <TD NOWRAP>). Αυτό είναι ισοδύναμο με το

 ROWSPAN

Έχει ως τιμή έναν ακέραιο αριθμό που ορίζει το πόσες γραμμές θα καταληφθούν από αυτό το κελί. Εξ' ορισμού ισούται με 1.

 COLSPAN

Έχει ως τιμή έναν ακέραιο αριθμό που ορίζει το πόσες στήλες θα καταληφθούν από αυτό το κελί. Εξ' ορισμού ισούται με 1.

 ALIGN

Καθορίζει την εξ' ορισμού οριζόντια ευθυγράμμιση των περιεχομένων του κελιού και μπορεί να επανακαθορίσει την παράμετρο ALIGN μιας σειράς του πίνακα. Δυνατές τιμές: LEFT, CENTER και RIGHT. Σε παράλειψη της παραμέτρου ALIGN, εξ΄ ορισμού είναι αριστερή ευθυγράμμιση για το <TD> και κεντρική ευθυγράμμιση για το <TH>, αλλά μπορούν να αλλαχθούν και

από την παράμετρο ALIGN του στοιχείου TR.

 VALIGN

Καθορίζει την εξ' ορισμού κάθετη ευθυγράμμιση των περιεχομένων του κελιού και μπορεί να επανακαθορίσει την παράμετρο VALIGN μιας σειράς του πίνακα. Δυνατές τιμές: TOP, MIDDLE και BOTTOM. Σε παράλειψη της παραμέτρου VALIGN, εξ΄ ορισμού είναι κεντρική ευθυγράμμιση, αλλά μπορεί να αλλαχθεί με την παράμετρο VALIGN του στοιχείου TR.

 WIDTH

Καθορίζει το απαιτούμενο πλάτος των περιεχομένων ενός κελιού σε pixel εξαιρώντας τις σκιές του κελιού.

 HEIGHT

Καθορίζει το απαιτούμενο ύψος των περιεχομένων ενός κελιού σε pixel εξαιρώντας τις σκιές του κελιού.

Όλοι οι νέοι φυλλομετρητές υποστηρίζουν την παράμετρο BGCOLOR στα TH και TD. Καθορίζει το χρώμα του φόντου του κελιού και έχει την ίδια σύνταξη με αυτή του στοιχείου BODY που περιγράφτηκε πιο πριν.

Οι αλγόριθμοι που χρησιμοποιούνται για την αυτόματη αλλαγή του μεγέθους των πινάκων θα πρέπει να λαμβάνουν υπόψη τους το μέγιστο και ελάχιστο πλάτος που απαιτείται για το κελί.

Το ελάχιστο και μέγιστο πλάτος των ένθετων πινάκων συμβάλλουν στο μικρότερο και μεγαλύτερο πλάτος του κελιού στο οποίο συμμετέχουν. Μόλις οι απαιτήσεις σε πλάτος γίνουν γνωστές για την κορυφή ενός πίνακα, τα πλάτη των στηλών μπορούν να οριστούν. Αυτό επιτρέπει στα πλάτη των ένθετων στηλών να ορίζονται εύκολα.

Στοιχεία ορισμού τύπων κειμένου

Τα στοιχεία ορισμού κειμένου τα οποία ορίζουν τύπους χαρακτήρων γενικά μπορούν να έχουν u940 άλλα ένθετα στοιχεία, αλλά όχι και στοιχεία ορισμού περιοχής.

Στοιχεία τύπου γραμματοσειράς

Τα στοιχεία τύπου γραμματοσειράς (Font Style Elements) απαιτούν την ετικέτα αρχής και την ετικέτα τέλους. Για παράδειγμα:

<B>Αυτό είναι κάποιο κείμενο σε bold μορφή</B>.

Τα στοιχεία ορισμού κειμένου πρέπει να είναι ένθετα με σωστή μορφή. Το ακόλουθο

είναι λανθασμένο:

Αυτό είναι κάποιο <B> bold και <I></B> italic κείμενο </I>.

Οι φυλλομετρητές θα πρέπει να καταβάλλουν κάθε προσπάθεια για την απεικόνιση των ένθετων στοιχείων κειμένου: Για παράδειγμα:

Αυτό είναι κάποιο<B>bold και <I>italic κείμενο</I></B>.

Τα διαθέσιμα στοιχεία είναι:

TT Κείμενο σε teletype ή monospaced μορφή

I Πλαγιαστός τύπος κειμένου

B Τύπος κειμένου bold

U Τύπος υπογραμμισμένου κειμένου

STRIKE Τύπος κειμένου strike-through

BIG Κείμενο σε κεφαλαία

SMALL Κείμενο σε πεζά

SUB Κείμενο σε μορφή δείκτη

SUP Κείμενο σε μορφή εκθέτη

Στοιχεία κειμένου

Στοιχεία φορμών

Τα στοιχεία INPUT, SELECT και TEXTAREA επιτρέπονται μόνο μέσα στο στοιχείο FORM. Το στοιχείο INPUT παίρνει μια πληθώρα από παραμέτρους περιλαμβάνοντας

πεδία κειμένου, πεδία κωδικών, πεδία επιλογής, checkboxes, κουμπιά radio, submit and reset κουμπιά, κρυμμένα πεδία, file upload, και κουμπιά εικόνας. Τα στοιχεία SELECT χρησιμοποιούνται για μονής ή πολλαπλής επιλογής μενού. Τα στοιχεία TEXTAREA χρησιμοποιούνται για πεδία κειμένου πολλαπλών γραμμών.

Το στοιχείο INPUT

Στα στοιχεία INPUT απαγορεύεται να υπάρχει ετικέτα τέλους.

 type

Καθορίζουν τον τύπο του πεδίου εισόδου:

 type=text(εξ' ορισμού)

Πρόκειται για κείμενο μιας γραμμής του οποίου το ορατό μήκος καθορίζεται με την παράμετρο size, π.χ. size=40 για κείμενο μήκος 40 χαρακτήρων. Οι χρήστες

μπορούν να εισάγουν περισσότερους χαρακτήρες από το παραπάνω όριο. Σε περίπτωση που θα πρέπει να καθοριστεί ένας μέγιστος αριθμός χαρακτήρων, τότε υπάρχει η παράμετρος maxlength. Η παράμετρος name χρησιμοποιείται για να δώσει κάποιο όνομα στο παρόν πεδίο και η παράμετρος value δίνει μια αρχική τιμή στην γραμματοσειρά που θα φαίνεται στο πεδίο μόλις φορτωθεί η φόρμα.

<input type=text size=40 name=user value="Τάσος Καλός">

 type=password

Είναι όμοιο με τον τύπο text, με τη διαφορά ότι απεικονίζει το χαρακτήρα "*" αντί του χαρακτήρα που πληκτρολογείται, κρύβοντας τον κωδικό που εισάγεται. Μπορούν να χρησιμοποιηθούν και οι παράμετροι size και maxlength ελέγχοντας το ορατό και το μέγιστο μέγεθος του εισαγόμενου κωδικού.

<input type=password size=12 name=passwd>

 type=checkbox

Ορίζει απλές δυαδικές τιμές ή για τιμές. Υποστηρίζονται πολλαπλές τιμές και να επιτευχθεί με τη χρήση πολλαπλών στοιχείων checkbox με το ίδιο όνομα αλλά με διαφορετική τιμή στην παράμετρο value. Κάθε ενεργοποιημένο checkbox δημιουργεί ένα ξεχωριστό ζεύγος ονόματος/τιμής στα δεδομένα που στέλνονται, ακόμα κι αν αυτό καταλήγει στη δημιουργία διπλών ίδιων ονομάτων. Η παράμετρος checked αρχικοποιεί το checkbox στην ενεργοποιημένη του κατάσταση.

<input type=checkbox checked name=uscitizen value=yes>

 type=radio

Ο τύπος αυτός μπορεί να δεχθεί μια τιμή μεταξύ μιας πληθώρας εναλλακτικών τιμών. Κάθε κουμπί radio που συμμετέχει στην ίδια ομάδα θα πρέπει να έχει το ίδιο όνομα (παράμετρος name) και απαιτεί οπωσδήποτε μια τιμή. Μόνο το ενεργοποιημένο κουμπί radio δημιουργεί ένα ζεύγος ονόματος/τιμής στα δεδομένα που στέλνονται. Σε μια ομάδα από κουμπιά radio, ένα από αυτά θα πρέπει να οριστεί ως ενεργοποιημένο με χρήση της παραμέτρου checked.

<input type=radio name=age value="0-12">

<input type=radio name=age value="13-17">

<input type=radio name=age value="18-25">

<input type=radio name=age value="26-35" checked>

<input type=radio name=age value="36-">

 type=submit

Δημιουργεί ένα κουμπί, το οποίο μπορούν να χρησιμοποιήσουν οι χρήστες για την αποστολή των δεδομένων της φόρμας στον εξυπηρετητή παγκόσμιου ιστού. Η ετικέτα του κουμπιού ονοματίζεται με την παράμετρο value. Αν δοθεί και όνομα στο πεδίο submit, τότε στέλνεται και το ζεύγος ονόματος/τιμής του πεδίου submit στον εξυπηρετητή. Μπορούν να χρησιμοποιηθούν πολλαπλά κουμπιά submit σε μια φόρμα. Για κουμπιά με εικόνες δείτε υπάρχει ο τύπος image.
<input type=submit value="Let's Go to Patras Carnival Festival">

Η γλώσσα PHP

Η PHP, της οποίας τα αρχικά αντιπροσωπεύουν το "PHP: Hypertext Preprocessor" είναι μια ευρέως χρησιμοποιούμενη, ανοιχτού κώδικα, γενικού σκοπού scripting γλώσσα προγραμματισμού, η οποία είναι ειδικά κατάλληλη για ανάπτυξη εφαρμογών για το Web και μπορεί να ενσωματωθεί στην HTML.

Απλή απάντηση, αλλά τι σημαίνει; Ένα παράδειγμα:

Παράδειγμα 1-1. Ένα εισαγωγικό παράδειγμα

```
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
        echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

Παρατηρήστε πως αυτό είναι διαφορετικό από ένα script γραμμένο σε άλλες γλώσσες προγραμματισμού όπως η Perl ή η C : Αντί να γράφετε ένα πρόγραμμα με πολλές εντολές για να εξάγετε HTML, γράφετε ένα HTML script με κάποιο ενσωματωμένο κώδικα για να κάνει κάτι (σε αυτή την περίπτωση, να εμφανίζει κάποιο κείμενο). Ο κώδικας PHP είναι εσώκλειστος σε ειδικά tags (ετικέτες) αρχής και τέλους που σας επιτρέπουν να μεταφέρεστε μέσα και έξω από το "PHP mode" (PHP τρόπο λειτουργίας).

Αυτό που διαχωρίζει την PHP από κάτι σαν client-side Javascript είναι ότι ο κώδικας εκτελείται στον server (εξηπηρετητή). Αν είχατε ένα script σαν το παραπάνω στον server σας, ο client θα έπαιρνε τα αποτελέσματα της εκτέλεσης αυτού του script, χωρίς να υπάρχει κανένας τρόπος να καταλάβει τι κώδικας υπάρχει από κάτω.

Μπορείτε ακόμη να ρυθμίσετε τον web server σας να χειρίζεται όλα τα HTML αρχεία σας με την PHP, και τότε πραγματικά δεν υπάρχει τρόπος ο χρήστης να καταλάβει τι έχετε κάτω από το μανίκι σας.

Τα καλύτερο πράγμα στην PHP είναι ότι είναι εξαιρετικά απλή για ένα νεοφερμένο αλλά προσφέρει πολλές προηγμένα χαρακτηριστικά για ένα επαγγελματία προγραμματιστή. Μην τρομάζετε όταν διαβάζετε την μακροσκελή λίστα με τα χαρακτηριστικά της PHP. Μπορείτε να εξοικειωθείτε μέσα σε πολύ λίγο χρόνο και να αρχίσετε να γράφετε απλά script σε λίγες ώρες.

Αν και η ανάποτυξη της PHP εστιάζεται σε server-side scripting, μπορείτε να κάνετε πολύ περισσότερα με αυτή. Διαβάστε παρακάτω και δείτε περισσότερα στην παράγραφο Τι μπορεί να κάνει η PHP;.

Τι μπορεί να κάνει η PHP;

Οτιδήποτε. Η PHP επικεντρώνεται κυρίως στο server-side scripting, έτσι μπορείτε να κάνετε οτιδήποτε ένα άλλο CGI πρόγραμμα μπορεί να κάνει, όπως να μαζέψει δεδομένα, να παράγει δυναμικό περιεχόμενο σελίδων, ή να στείλει και να πάρει cookies. Αλλά η PHP μπορεί να κάνει πολύ περισσότερα.

Υπάρχουν τρεις κύριοι τομείς που χρησιμοποιείται ένα PHP script.

- Server-side scripting. Αυτό είναι το πιο παραδοσιακό και το κύριο πεδίο για την PHP. Χρειάζεστε τρία πράγματα για να δουλέψει αυτό. Τον PHP μεταγλωττιστή (parser) (CGI ή server module), ένα webserver (εξηπηρετητή σελίδων) και ένα web browser ("φυλλομετρητή"). Πρέπει να τρέξετε τον webserver, με μια συνδεδεμένη εγκατάσταση της PHP. Μπορείτε να προσπελάσετε τα αποτελέσματα του PHP προγράμματος με ένα web browser, βλέποντας την σελίδα PHP μέσα από τον server. Για περισσότερες πληροφορίες, δείτε την παράγραφο οδηγίες εγκατάστασης.
- Command line scripting. Μπορείτε να φτιάξετε ένα PHP script για να το τρέχετε χωρίς server ή browser. Χρειάζεστε μόνο τον PHP μεταγλωττιστή για

να την χρησιμοποιήσετε με αυτό τον τρόπο. Αυτός ο τύπος είναι ιδανικός για script που εκτελούνται συχνά με τη χρήση της cron (σε *nix ή Linux) ή με τον Task Scheduler (στα Windows). Αυτά τα script μπορούν επίσης να χρησιμοποιηθούν για απλές εργασίες επεξεργασίες κειμένου. Δείτε την ενότητα σχετικά με την Command line χρήση της PHP για περισσότερες πληροφορίες.

- Εγγραφή client-side GUI εφαρμογών (Γραφικά περιβάλλοντα χρηστών). Η PHP ίσως να μην είναι η πιο καλή γλώσσα για να γράψει κανείς παραθυριακές εφαρμογές, αλλά αν ξέρετε PHP πολύ καλά και θέλετε να χρησιμοποιήσετε κάποια προχωρημένα χαρακτηριστικά της PHP στις client-side εφαρμογές σας, μπορείτε επίσης να χρησιμοποιήσετε το PHP-GTK για αυτού του είδους τα προγράμματα. Έχετε επίσης τη δυνατότητα να γράφετε cross-platform εφαρμογές με αυτό τον τρόπο. Το PHP-GTK είναι μια επέκταση της PHP και δεν συμπεριλαμβάνεται στην κύρια διανομή. Αν ενδιαφέρεστε για το PHP-GTK, επισκεφτείτε την δική του ιστοσελίδα.

Η PHP μπορεί να χρησιμοποιηθεί σε όλα τα κύρια λειτουργικά συστήματα, συμπεριλαμβανομένου του Linux, πολλών εκδοχών του Unix (HP-UX, Solaris και OpenBSD), Microsoft Windows, Mac OS X, RISC OS και πιθανώς σε άλλα. Η PHP υποστηρίζει επίσης τους Apache, Microsoft Internet Information Server, Personal Web Server, Netscape και iPlanet servers, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd, και πολλούς άλλους webserver. Για την πλειοψηφία των server η PHP έχει ένα module, για τους υπόλοιπους η PHP μπορεί να λειτουργήσει ως ένας CGI επεξεργαστής.

Έτσι με την PHP έχετε την ελευθερία επιλογής ενός λειτουργικού συστήματος και ενός web server. Επιπλέον, έχετε επίσης την ελευθερία να χρησιμοποιήσετε συναρτησιακό (procedural) ή αντικειμενοστρεφή (object oriented) προγραμματισμό ή μια ανάμειξη τους. Αν και η παρούσα έκδοση δεν υποστηρίζει όλα τα πρότυπα χαρακτηριστικά, μεγάλες βιβλιοθήκες κώδικα και μεγάλες εφαρμογές (συμπεριλαμβανομένης και της βιβλιοθήκης PEAR) είναι γραμμένες μόνο με αντικειμενοστρεφή κώδικα.

Με την PHP δεν είστε περιορισμένοι να εξάγετε HTML. Οι δυνατότητες της PHP συμπεριλαμβάνουν την εξαγωγή εικόνων, αρχείων PDF, ακόμη και ταινίες Flash

(χρησιμοποιώντας τα libswf και Ming) παράγονται αμέσως. Μπορείτε επίσης να εξάγετε εύκολα οποιοδήποτε κείμενο όπως XHTML και οποιοδήποτε άλλο XML αρχείο. Η PHP μπορεί να δημιουργεί αυτόματα αυτά τα αρχεία και να τα αποθηκεύει στο σύστημα αρχείων, αντί να τα εκτυπώνει, αποτελώντας έτσι μια server-side cache για το δυναμικό σας περιεχόμενο.

Ένα από τα πιο δυνατά και σημαντικά χαρακτηριστικά της PHP είναι η υποστήριξη που έχει για ένα μεγάλο σύνολο βάσεων δεδομένων. Η συγγραφή μιας σελίδας που υποστηρίζει βάσεις δεδομένων είναι εξαιρετικά απλή. Οι εξής βάσεις δεδομένων υποστηρίζονται μέχρι στιγμήςσ:

Adabas D	Ingres	Oracle (OCI7 and OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

Έχουμε επίσης μια αφαιρετική επέκταση DBX βάσεων δεδομένων (DBX database abstraction extension) που σας επιτρέπει διάφανα να χρησιμοποιείτε οποιαδήποτε βάση δεδομένων υποστηρίζεται από αυτή την επέκταση. Επιπλέον η PHP υποστηρίζει το ODBC, το Open Database Connection standard (Ανοιχτό πρότυπο Σύνδεσης Βάσεων δεδομένων) έτσι μπορείτε να συνδεθείτε σε οποιαδήποτε βάση δεδομένων που υποστηρίζει αυτό το παγκόσμιο πρότυπο.

Η PHP έχει επίσης υποστήριξη για επικοινωνία με άλλες υπηρεσίες χρησιμοποιώντας πρωτόκολλα όπως LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (στα Windows) και αμέτρητα άλλα. Μπορείτε επίσης να ανοίξετε raw network sockets και να αλληλεπιδράσετε με οποιοδήποτε άλλο πρωτόκολλο. Η PHP έχει ακόμη υποστήριξη για την περίπλοκη ανταλλαγή δεδομένων WDDX μεταξύ σχεδόν όλων των Web programming γλωσσών. Μιλώντας για δια-επικοινωνία, η PHP υποστηρίζει instantiation αντικειμένων Java και τα χρησιμοποιεί διάφανα σαν αντικείμενα PHP.

Μπορείτε επίσης να χρησιμοποιήσετε την CORBA επέκταση μας για να προσπελάσετε remote (απομακρυσμένα) αντικείμενα.

Η PHP έχει εξαιρετικά χρήσιμα χαρακτηριστικά επεξεργασίας κειμένων, από την POSIX επέξταση ή τις Perl regular expressions μέχρι XML parsing αρχείων. Για τη μεταγλώττιση και την πρόσβαση αρχείων XML, υποστηρίζουμε τα πρότυπα SAX και DOM. Μπορείτε να χρησιμοποιήσετε την XSLT επέκταση μας για να μετατρέπετε τα XML αρχεία σε άλλες μορφές.

Καθώς χρησιμοποιείτε την PHP στον τομέα του ecommerce, θα βρείτε τις Cybercash payment, CyberMUT, VeriSign Payflow Pro και CCVS συναρτήσεις χρήσιμες για τα online προγράμματα πληρωμής σας.

Τελευταίο αλλά σημαντικό, έχουμε πολλές άλλες ενδιαφέρουσες επεκτάσεις, τις mnoGoSearch search engine συναρτήσεις, πολλά εργαλεία συμπίεσης (gzip, bz2), μετατροπές ημερολογίου, μεταφράσεις...

Όπως βλέπετε αυτή η σελίδα δεν είναι αρκετή για να απαριθμήσει όλα τα χαρακτηριστικά και πλεονεκτήματα της PHP. Διαβάστε τις παρακάτω παραγράφους σχετικά με την εγκατάσταση της PHP και δείτε το μέρος με την παραπομπή συναρτήσεων για επεξήγηση των επεκτάσεων που αναφέρονται εδώ.

Ένα απλό tutorial

Εδώ θέλουμε να σας δείξουμε τα πολύ βασικά της PHP σε ένα μικρό και απλό μάθημα. Αυτό το κείμενο δείχνει μόνο δυναμική δημιουργία ιστοσελίδων με την PHP αλλά η PHP είναι ικανή να κάνει και πολλά άλλα πράγματα από την δημιουργία ιστοσελίδων. Δείτε την ενότητα με τίτλο Τι μπορεί να κάνει η PHP για περισσότερες πληροφορίες.

Οι PHP-ικανές σελίδες αντιμετωπίζονται σαν συνηθισμένες HTML σελίδες και μπορείτε να τις δημιουργήσετε και να τις μορφοποιήσετε με τον ίδιο τρόπο που συνήθως δημιουργείτε κανονικές HTML σελίδες.

Τι χρειάζομαι;

Σε αυτό το tutorial υποθέτουμε πως ο server σας έχει ενεργή την υποστήριξη για PHP και πως όλα τα αρχεία που έχουν κατάληξη `.php` τα χειρίζεται η PHP. Στους περισσότερους server αυτή είναι η προκαθορισμένη επέκταση για τα PHP αρχεία, αλλά ρωτήστε τον διαχειριστή του server για να είστε σίγουροι. Αν ο server σας υποστηρίζει PHP, τότε δεν χρειάζετε να κάνετε τίποτα. Απλά δημιουργείτε τα `.php` αρχεία και τα τοποθετείτε στην web directory και ο server θα τα μεταγλωττίσει μαγικά για σας. Δεν χρειάζεται να κάνετε compile τίποτα ούτε να εγκαταστήσετε επιπλέον εργαλεία. Σκεφτείτε αυτές τις PHP-ικανές σελίδες ως απλά HTML αρχεία με μια μεγάλη οικογένεια καινούριων μαγικών tags που σας επιτρέπουν να κάνετε όλων των ειδών τα πράγματα.

Η πρώτη σας PHP-ικανή σελίδα

Δημιουργήστε ένα αρχείο με όνομα `hello.php` κάτω από την webserver root directory με το ακόλουθο περιεχόμενο:

Παράδειγμα 2-1. Our first PHP script: `hello.php`

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <?php echo "Hello World<p>"; ?>
 </body>
</html>
```

Η έξοδος του script θα είναι:

```
<html>
```

```
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 Hello World<p>
 </body>
</html>
```

Προσέξτε πως αυτό δεν είναι σαν ένα CGI script. Το αρχείο δεν χρειάζεται να είναι εκτελέσιμο με οποιοδήποτε τρόπο. Σκεφτείτε το σαν ένα κανονικό HTML αρχείο που τυγχαίνει να έχει ένα σετ από ειδικά tags που είναι διαθέσιμα σε σας που κάνουν πολλά και ενδιαφέροντα πράγματα.

Αυτό το πρόγραμμα είναι εξαιρετικά απλό και πραγματικά δεν χρειάζεστε την PHP για να δημιουργήσετε μια σελίδα σαν και αυτή. Το μόνο που κάνει είναι να εμφανίζει: `Hello World` χρησιμοποιώντας την συνάρτηση **echo()** της PHP.

Αν δοκιμάσατε αυτό το παράδειγμα και δεν είχε καμία έξοδο ή σας προέτρεψε σε ένα download, ή βλέπετε όλο το αρχείο ως κείμενο, οι πιθανότητες είναι πως ο server που χρησιμοποιείτε δεν έχει ενεργοποιημένη την PHP. Ζητήστε από τον διαχειριστή να την ενεργοποιήσει χρησιμοποιώντας το κεφάλαιο Εγκατάσταση αυτού του manual. Αν θέλετε να αναπτύσσετε PHP scripts τοπικά, δείτε την ενότητα downloads. Μπορείτε να αναπτύσσετε τοπικά σε οποιοδήποτε λειτουργικό σύστημα αλλά σιγουρευτείτε ότι εγκαταστήσατε και ένα κατάλληλο web server.

Το νόημα του παραδείγματος είναι να δείξει την ειδική μορφή του PHP tag (της PHP ετικέτας). Σε αυτό το παράδειγμα χρησιμοποιήσαμε το `<?php` για να δείξουμε την αρχή ενός PHP tag. Έπειτα βάλαμε την PHP πρόταση και αφήσαμε PHP λειτουργία προθέτοντας το tag κλεισίματος `?>`. Μπορείτε να πηδάτε μέσα και έξω από την PHP λειτουργία σε ένα HTML αρχείο όσο θέλετε.

Μια σημείωση στους text editors: Υπάρχουν πολλοί text editors (κειμενογράφοι) και Integrated Development Environments (IDEs) (Ολοκληρωμένα περιβάλλοντα ανάπτυξης) που μπορείτε να χρησιμοποιήσετε για να δημιουργήσετε, να μορφοποιήσετε και να χειριστείτε αρχεία PHP. Μια μικρή λίστα αυτών των εργαλείων διατηρείται στο PHP Editor's List. Αν θέλετε να προτείνετε ένα συντάκτη,

παρακαλούμε επισκεφτείτε την παραπάνω σελίδα και ζητήστε από τον διαχειριστή της σελίδες να προσθέσει τον συντάκτη στην λίστα.

Μια σημείωση για τους Word Processors: Οι word processors (επεξεργαστές κειμένου) όπως τα Openoffice.org, StarOffice Writer, Microsoft Word και Abiword δεν είναι καλή επιλογή για την μορφοποίηση αρχείων PHP.

Αν επιθυμείτε να χρησιμοποιήσετε ένα από αυτούς για αυτό το δοκιμαστικό script σιγουρευτείτε πως θα αποθηκεύσετε το αρχείο ως ΑΠΛΟ ΚΕΙΜΕΝΟ αλλιώς η PHP δεν θα είναι ικανή να εκτελέσει το script.

Μια σημείωση για το Windows Notepad: Αν γράφετε τα PHP scripts χρησιμοποιώντας το Windows Notepad, θα πρέπει να σιγουρευτείτε πως τα αρχεία σας αποθηκεύονται με την επέκταση .php. (Το Notepad προσθέτει μια .txt επέκταση στα αρχεία αυτόματα εκτός και αν προβείτε σε ένα από τα ακόλουθα κείμενα για να το αποφύγετε.)

Όταν αποθηκεύτετε το αρχείο και σας ζητείται να ορίσετε ένα όνομα για το αρχείο, βάλτε το όνομα του αρχείο σε εισαγωγικά (δηλ. "hello.php").

Εναλλακτικά, μπορείτε να κάνετε κλικ στο 'Text Documetns' drop-down menu στο παράθυρο αποθήκευσης αρχείου και να αλλάξετε την επιλογή σε "All files". Μπορείτε τότε να εισάγετε το όνομα του αρχείου χωρίς εισαγωγικά.

Κάτι χρήσιμο

Ας κάνουμε κάτι λίγο πιο χρήσιμο τώρα. Θα ελέγξετε τι είδους browser χρησιμοποιεί το άτομο που βλέπει τη σελίδα. Για να το κάνουμε αυτό, ελέγχουμε το user agent string που στέλνει ο browser σαν μέρος του HTTP request (αιτήματος). Αυτή η πληροφορία αποθηκεύεται σε μια μεταβλητή. Οι μεταβλητές πάντα αρχίζουν με ένα σύμβολο δολλαρίου στην PHP. Η μεταβλητή που μας ενδιαφέρει τώρα είναι `$_SERVER["HTTP_USER_AGENT"]`.

Σημείωση για τις PHP Autoglobals: Η $_SERVER είναι μια ειδική δεσμευμένη μεταβλητή της PHP η οποία περιέχει όλες τις πληροφορίες του web server. Είναι γνωστή ως μια Autoglobal (ή Superglobal). Δείτε την σχετική σελίδα του manual για τις Autoglobals για περισσότερες πληροφορίες. Αυτές οοι ειδικές μεταβλητές εισηγήθηκαν στην PHP 4.1.0. Πριν από χρησιμοποιούσαμε αντί αυτού τον παλαιότερο array `$HTTP_*_VARS`, όπως η `$HTTP_SERVER_VARS`. Αν και ξεπερασμένες, αυτές οι παλαιότερες μεταβλητές ακόμη υπάρχουν. (Δείτε επίσης την σημείωση στον παλιό κώδικα.)

Για να εμφανίσουμε αυτή την μεταβλητή, μπορούμε απλά να κάνουμε:

Παράδειγμα 2-2. Εκτυπώνοντας μια μεταβλητή (Στοιχείο array)

```
<?php echo $_SERVER["HTTP_USER_AGENT"]; ?>
```

Ένα παράδειγμα εξόδου αυτού του script θα μπορούσε να είναι:

```
Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
```

Υπάρχουν πολλά είδη τύπων μεταβλητών στην PHP. Στο παραπάνω παράδειγμα εκτυπώσαμε ένα στοιχείο ενός Array (πίνακα). Τα arrays είναι πολύ χρήσιμοι τύποι μεταβλητών.

Η `$_SERVER` είναι ακόμη μια μεταβλητή που γίνεται αυτόματα διαθέσιμη σε σας από την PHP. Μπορείτε να δείτε μια λίστα στην ενότητα Δεσμευμένες μεταβλητές του manual ή μπορείτε να πάρετε μια πλήρη λίστα αυτών, δημιουργώντας ένα αρχείο που είναι κάπως έτσι:

Παράδειγμα 2-3. Εμφάνιση όλων των προκαθορισμένων μεταβλητών με την phpinfo()

```
<?php phpinfo(); ?>
```

Αν φορτώσετε αυτό το αρχείο στον browser σας θα δείτε μια σελίδα γεμάτη με πληροφορίες για την PHP μαζί με μια λίστα όλων των μεταβλητών που είναι διαθέσιμες σε σας.

Μπορείτε να τοποθετήσετε πολλαπλές προτάσεις PHP σε ένα tag της PHP και να δημιουργήσετε μικρά μπλοκ κώδικα που κάνουν περισσότερα από μια απλή echo. Για παράδειγμα, αν θέλαμε να ελέγξουμε για την χρήση του Internet Explorer θα μπορούσαμε να κάνουμε κάτι σαν και αυτό:

Παράδειγμα 2-4. Παράδειγμα: Δομές ελέγχου (control structures) και Συναρτήσεις

```
<?php
if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) {
        echo "You are using Internet Explorer<br />";
}
?>
```

Ένα παράδειγμα εξόδου αυτού του script μπορεί να είναι:

```
You are using Internet Explorer<br />
```

Εδώ εισάγουμε μερικές καινούριες έννοιες. Έχουμε μια πρόταση if. Αν είστε γνώριμοι με την βασική σύνταξη που χρησιμοποιείται από την γλώσσα προγραμματισμού C, τότε αυτό θα σας φανεί λογικό. Αν δεν ξέρετε αρκετή C ή κάποια άλλη γλώσσα προγραμματισμού που χρησιμοποιείται η παραπάνω σύνταξη, ίσως χρειαστείτε να διαλέξετε ένα εισαγωγικό βιβλίο PHP και να διαβάσετε τα πρώτα κεφάλαια, ή να διαβάσετε το κομμάτι Αναφορά στην Γλώσσα αυτού του manual. Μπορείτε να βρείτε μια λίστα βιβλίων για PHP στο http://www.php.net/books.php.

Η δεύτερη έννοια που εισάγεται είναι η κλήση της συνάρτησης **strstr()**. Η **strstr()** είναι μια συνάρτηση ενσωματωμένη μέσα στην PHP που ψάχνει ένα string για να βρεί μέσα του ένα άλλο string. Σε αυτή την περίπτωση ψάχνουμε για το `"MSIE"` μέσα στο `$_SERVER["HTTP_USER_AGENT"]`. Αν το string βρεθεί, η συνάρτηση επιστρέφει **TRUE** και αν δεν βρεθεί, επιστρέφει **FALSE**. Αν επιστρέψει **TRUE**, η πρόταση if γίνεται και αυτή **TRUE** και ο κώδικας μέσα στα {άγκιστρα} εκτελείται. Αλλιώς, δεν εκτελείται. Δημιουργήστε και εσείς παρόμοια παραδείγματα με if , else και άλλες συναρτήσεις όπως τις **strtoupper()** και **strlen()**. Κάθε σχετιζόμενη σελίδα του manual περιέχει παραδείγματα επίσης.

Μπορούμε να προχωρήσουμε ένα βήμα παραπέρα και να δείξουμε πως μπορούμε να μπαίνουμε και να βγαίνουμε από την PHP-λειτουργία (PHP-mode) ακόμη και στη μέση ενός PHP μπλοκ:

Παράδειγμα 2-5. Ανακατεύοντας και HTML και PHP λειτουργίες

```
<?php
if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) {
?>
<h3>strstr must have returned true</h3>
<center><b>You are using Internet Explorer</b></center>
<?php
} else {
?>
<h3>strstr must have returned false</h3>
<center><b>You are not using Internet Explorer</b></center>
<?php
}
?>
```

Ένα παράδειγμα εξόδου αυτού του script μπορεί να είναι:

```
<h3>strstr must have returned true</h3>
<center><b>You are using Internet Explorer</b></center>
```

Αντί να χρησιμοποιούμε μια PHP echo πρόταση για να εξάγουμε κάτι, βγήκαμε από την PHP λειτουργία PHP λειτουργία και στείλαμε απλή HTML. Το σημαντικό και δυνατό στοιχείο που πρέπει να προσέξουμε εδώ είναι ότι η λογική ροή του script παραμένει ανέπαφη. Μόνο ένα από τα παραπάνω HTML μπλοκ θα σταλεί στον θεατή, ανάλογα με το αν η **strstr()** επέστρεψε **`TRUE`** ή **`FALSE`**. Με άλλα λόγια, αν το string `MSIE` έχει βρεθεί ή όχι.

Χειρίζοντας φόρμες (Form)

Ένα από τα πιο ισχυρά χαρακτηριστικά της PHP είναι ο τρόπος που χειρίζεται τις HTML φόρμες (forms). Η βασική ιδέα που είναι σημαντική να καταλάβετε είναι πως οποιοδήποτε στοιχείο της φόρμας θα γίνει διαθέσιμο στο PHP script σας. Διαβάστε την ενότητα του manual για τις Μεταβλητές από έξω από την PHP για περισσότερες πληροφορίες στο πως να χρησιμοποιείτε φόρμες με την PHP. Ένα παράδειγμα μιας HTML φόρμας:

Παράδειγμα 2-6. Μια απλή HTML φόρμα

```
<form action="action.php" method="POST">
 Your name: <input type="text" name="name" />
 Your age: <input type="text" name="age" />
 <input type="submit">
</form>
```

Δεν υπάρχει τίποτα ειδικό σχετικά με αυτή τη φόρμα. Είναι μια απλή HTML φόρμα χωρίς ειδικά tags οποιουδήποτε είδους. Όταν ο χρήστης γεμίσει αυτή τη φόρμα και πατήσει το κουμπί submit (υποβολή), η σελίδα `action.php` καλείται. Σε αυτό το αρχείο, θα έχετε κάτι σαν και αυτό:

Παράδειγμα 2-7. Εκτυπώνοντας δεδομένα από μια φόρμα

```
Hi <?php echo $_POST["name"]; ?>.
You are <?php echo $_POST["age"]; ?> years old.
```

Ένα παράδειγμα εξόδου αυτού του script μπορεί να είναι:

```
Hi Joe.
You are 22 years old.
```

Πρέπει να είναι εμφανές το τι κάνει. Δεν υπάρχει τίποτα άλλο εκτός από αυτό. Οι μεταβλητές `$_POST["name"]` και `$_POST["age"]` αυτόματα ορίζονται για σας από την PHP. Νωρίτερα χρησιμοποιήσαμε την `$_SERVER` autoglobal, τώρα παραπάνω μόλις χρησιμοποιήσαμε την $_POST autoglobal που περιέχει όλα τα δεδομένα POST. Προσέξτε πως η *method* (μέθοδος) στην φόρμα μας είναι η POST. Αν χρησιμοποιούσαμε την *GET* method, τότε οι πληροφορίες της φόρμας μας θα ζούσαν

αντίστοιχα μέσα στην $_GET autoglobal. Μπορείτε επίσης να χρησιμοποιήσετε την $_REQUEST autoglobal αν δεν νοιάζεστε για την πηγή των δεδομένων σας. Περιέχει μια ανάμιξη από GET, POST, COOKIE και FILE δεδομένων. Δείτε επίσης την συνάρτηση **import_request_variables()**.

Χρησιμοποιώντας παλιό κώδικα με νέες εκδόσεις της PHP

Τώρα που η PHP έχει μεγαλώσει και είναι μια δημοφιλής scripting γλώσσα, υπάρχουν περισσότεροι πόροι εκεί έξω οι οποίοι έχουν καταλόγους από κώδικα που μπορείτε να επαναχρησιμοποιήσετε στα δικά σας scripts. Στο μεγαλύτερο μέρος, οι PHP developers (προγραμματιστές) της γλώσσας PHP προσπάθησαν να είναι προς-τα-πίσω συμβατοί (backwards compatible), ούτως ώστε ένα script γραμμένο για μια πιο παλιά έκδοση θα τρέχει (ιδανικά) χωρίς αλλαγές σε μια νεότερη έκδοση της PHP. Πρακτικά όμως, κάποιες αλλαγές θα χρειαστούν να γίνουν.

Δύο από τις πιο σημαντικές αλλαγές που επιρεάζουν παλαιότερο κώδικα είναι:

- Το παλιό array `$HTTP_*_VARS` έχει ξεπεραστεί (το οποίο χρειάζεται να αναγραφεί ως global όταν χρησιμοποιείται σε μια function ή method). Οι εξής autoglobal arrays εισήχθησαν στην PHP 4.1.0. Είναι οι: `$_GET`, `$_POST`, `$_COOKIE`, `$_SERVER`, `$_ENV`, `$_REQUEST`, and `$_SESSION`. Τα παλαιότερα `$HTTP_*_VARS` arrays, όπως το $HTTP_POST_VARS, υπάρχουν ακόμη από την PHP 3.
- Οι εξωτερικές μεταβλητές δεν γίνονται register (καταχωρούνται) ως global εξ ορισμού. Με άλλα λόγια, από την PHP 4.2.0 το PHP directive register_globals είναι εξ ορισμού *off* στο in `php.ini`. Η μέθοδος που προτιμείται για προσπέλαση αυτών των τιμών είναι μέσα από τα autoglobal arrays που αναφέρθηκαν νωρίτερα. Παλαιότερα scripts, βιβλία και tutorials (μαθήματα) μπορεί να στηρίζονται σε αυτή την ρύθμιση να είναι ενεργοποιημένη (on). Αν είναι ενεργοποιημένη, για παράδειγμα, κάποιος θα μπορούσε να χρησιμοποιήσει το `$id` από το URL

`http://www.example.com/foo.php?id=42`. Είτε on είτε off, η `$_GET['id']` είναι διαθέσιμη.

Βασικοί Κανόνες Σύνταξης

Βγαίνοντας από την HTML

Όταν η PHP μεταγλωτίζει (parses) ένα αρχείο, απλά κάνει ένα πέρασμα στο κείμενο του αρχείου μέχρι να συναντήσει ένα από τα ειδικά tags που της λένε να αρχίσει να μεταφράζει το κείμενο ως κώδικα PHP. Ο parser (μεταγλωττιστής) τότε εκτελεί ολόκληρο τον κώδικα που βρίσκει, μέχρι να συναντήσει το επόμενο PHP tag κλεισίματος, το οποίο λέει στον parser να αρχίσει να κάνει ξανά, απλά ένα πέρασμα στο κείμενο. Αυτός είναι ο μηχανισμός που σας επιτρέπει να προσθέτετε PHP κώδικα μέσα σε HTML: ο,τιδήποτε βρίσκεται έξω από τα tags της PHP μένει τελείως μόνο, ενώ οτιδήποτε μέσα μεταγλωττίζεται ως κώδικας.

Υπάρχουν τέσσερα σύνολα από tags που μπορούν να χρησιμοποιηθούν για να δηλώσουμε τα κομμάτια που έχουν κώδικα σε PHP. Από αυτά, μόνο δύο (<?php. . .?> and <script language="php">. . .</script>) είναι πάντα διαθέσιμα. Τα άλλα μπορούν να ενεργοποιηθούν και να απενεργοποιηθούν από το `php.ini` αρχείο ρυθμίσεων. Ενώ τα short-form tags και τα tags που μοιάζουν με αυτά της ASP μπορεί να είναι βολικά, δεν είναι τόσο portable όσο οι μακρύτερες εκδόσεις. Επίσης, αν σκοπεύετε να προσθέσετε PHP κώδικα σε XML ή XHTML, θα χρειαστεί να χρησιμοποιήσετε την <?php. . .?> φόρμα για να προσαρμοστεί στην XML.

Τα tags που υποστηρίζονται από την PHP είναι:

Παράδειγμα 5-1. Τρόποι για να βγείτε (escape) από την HTML

```
1.  <?php echo("if you want to serve XHTML or XML documents, do
like this\n"); ?>

2.  <? echo ("this is the simplest, an SGML processing
instruction\n"); ?>
    <?= expression ?> This is a shortcut for "<? echo expression
?>"

3.  <script language="php">
        echo ("some editors (like FrontPage) don't
              like processing instructions");
    </script>

4.  <% echo ("You may optionally use ASP-style tags"); %>
    <%= $variable; # This is a shortcut for "<% echo . . ." %>
```

Ο πρώτος τρόπος, <?php. . .?>, είναι και ο προτιμότερος, καθώς επιτρέπει τη χρήση της PHP σε κώδικα συμβατό με την XML όπως η XHTML.

Ο δεύτερος τρόπος δεν είναι πάντα διαθέσιμος. Τα σύντομα tags είναι διαθέσιμα μόνο όταν έχουν ενεργοποιηθεί. Αυτό μπορεί να γίνει μέσω της συνάρτησης **short_tags()** (μόνο στην PHP 3), ενεργοποιώντας την επιλογή ρύθμισης short_open_tag στο αρχείο ρυθμίσεων της PHP, ή κάνοντας compile την PHP με την επιλογή --enable-short-tags στο **configure**. Ακόμη και αν είναι ενεργοποιημένο ως προεπιλογή στο php.ini-dist, η χρήση των short tags δεν προτιμάται.

Ο τέταρτος τρόπος είναι διαθέσιμος μόνο αν τα ASP-style tags έχουν ενεργοποιηθεί χρησιμοποιώντας την asp_tags επιλογή ρυθμίσεων.

Σημείωση: Υποστήριξη για τα ASP-style tags προστέθηκε στην έκδοση 3.0.4.

Σημείωση: Η χρήση των short tags θα πρέπει να αποφεύγεται κατά την ανάπτυξη εφαρμογών ή βιβλιοθηκων (libraries) που προορίζονται για διανομή (redistribution), ή εφαρμογή σε PHP servers που δεν τους χειρίζεστε οι ιδιοι, επειδη τα short tags μπορεί να μην υποστηρίζονται από τον τελικό server. Για μεταφέρσιμο (portable), κώδικα που θα προορίζεται για χρήση και από άλλους, βεβαιωθείτε ότι δεν κάνετε χρήση των short tags.

Το tag κλεισίματος για το block θα συμπεριλάβει το αμέσως επόμενο trailing newline αν υπάρχει. Επίσης, το tag κλεισίματος αυτόματα υποδηλώνει και ένα ερωτηματικό. Δεν χρειάζεται να έχετε ερωτηματικό για να τερματίσετε την τελευταία γραμμή ενός PHP block.

Η PHP σας επιτρέπει να χρησιμοποιήσετε δομές σαν αυτή:

Παράδειγμα 5-2. Προχωρημένος τρόπος για να κάνετε escape

```
<?php
if ($expression) {
    ?>
    <strong>This is true.</strong>
    <?php
} else {
    ?>
    <strong>This is false.</strong>
    <?php
}
?>
```

Αυτό λειτουργεί όπως περιμέναμε, επειδή όταν η PHP φτάνει στα ?> tags κλεισίματος, απλά αρχίζει να εμφανίζει ο,τιδήποτε βρει μέχρι να συνατήσει ένα άλλο tag ανοίγματος. Το παράδειγμα που δόθηκε εδώ έχει επινοηθεί, φυσικά, με σκοπό να εμφανίσουμε μεγάλα blocks κειμένου, αφού το να ξεφεύγουμε από τη μεταγλώττιση της PHP είναι γενικά πιο αποτελεσματικό από το να στέλνουμε ολόκληρο το κείμενο μέσω της συνάρτησης **echo()** ή της **print()** ή κάτι τέτοιο.

Διαχωρισμός εντολών

Οι εντολές διαχωρίζονται με τον ίδιο τρόπο όπως και στην C ή την Perl - τερματίζουμε κάθε εντολή με ένα ερωτηματικό.

Το tag κλεισίματος (?>) επίσης υποδηλώνει το τέλος μιας έκφρασης-δηλωσης, συνεπώς τα ακόλουθα είναι ισοδύναμα:

```
<?php
    echo "This is a test";
?>

<?php echo "This is a test" ?>
```

Σχόλια

Η PHP υποστηρίζει σχόλια σαν της 'C', 'C++' και του Unix shell. Για παράδειγμα:

```
<?php
    echo "This is a test"; // This is a one-line c++ style comment
    /* This is a multi line comment
       yet another line of comment */
    echo "This is yet another test";
    echo "One Final Test"; # This is shell-style style comment
?>
```

Το σχόλιο "μια γραμμής" σχολιάζει μόνο μέχρι το τέλος μιας γραμμής ή του τρέχοντος block στον κώδικα της PHP, όποιο είναι πρώτο.

```
<h1>This is an <?php # echo "simple";?> example.</h1>
<p>The header above will say 'This is an example'.
```

Πρέπει να προσέχετε να μην εμφωλεύετε σχόλια σαν αυτά της 'C', κάτι που είναι πιθανό να συμβεί όταν σχολιάζετε μεγάλα blocks.

```
<?php
 /*
    echo "This is a test"; /* This comment will cause a problem */
 */
?>
```

Το σχόλιο μιας γραμμής στην πραγματικότητα σχολιάζει μέχρι το τέλος μιας γραμμής ή του τρέχοντος block στον κώδικα της PHP, όποιο είναι πρώτο. Αυτό σημαίνει πως ο HTML κώδικας που ακολουθεί το // ?> Θα εκτυπωθεί. Το tag ?> ξεφεύγει από την κατάσταση της PHP και επιστρέφει στην HTML, και το // δεν μπορεί να το επηρεάσει.

Τύποι

Εισαγωγή

Η PHP υποστηρίζει οχτώ πρωταρχικούς τύπους.

Για βαθμωτούς τύπους:

- **boolean** (δυαδικός)
- integer (ακέραιοι)
- **float** (αριθμοί κινητής υποδιαστολής, γνωστοί και ως '**double**')
- **string**

Δύο σύνθετους τύπους:

- array (πίνακες)
- object (αντικείμενα)

Και τέλος δυο ειδικούς τύπους:

- **resource**
- **NULL**

Αυτό το εγχειρίδιο επίσης σας εισάγει μερικούς pseudo-types (ψευδοτύπους) για λόγους αναγνωσιμότητας:

- mixed
- number
- callback

Ίσως βρείτε και κάποιες αναφορές στο τύπο "double". Θεωρείστε το double όμοιο με τον τύπο float, και τα δυο ονόματα υπάρχουν μόνο για ιστορικούς λόγους.

Ο τύπος μιας μεταβλητής δεν καθορίζεται συνήθως από τον προγραμματιστή, αλλά μάλλον καθορίζεται κατά τη διαρκεια εκτέλεσης από την PHP ανάλογα με το περιεχόμενο με το οποίο χρησιμοποιείται αυτή η μεταβλητή.

Σημείωση: Αν θέλετε να ελένξετε τον τύπο και την τιμή μιας συγκεκριμένης έκφρασης, χρησιμοποιείστε την **var_dump()**.

Σημείωση: Αν θέλετε απλά μια αναπαράσταση του τύπου, που να μπορεί εύκολα να διαβαστεί για το debugging, χρησιμοποιείστε την **gettype()**. Για τον έλεγχο ενός συγκεκριμένο τύπου, *μην* χρησιμοποιείτε την **gettype()**, αλλά τις `is_type`συναρτήσεις. Μερικά παραδείγματα:

```
<?php
$bool = TRUE;   // a boolean
$str  = "foo";  // a string
$int  = 12;     // an integer

echo gettype($bool); // prints out "boolean"
echo gettype($str);  // prints out "string"
```

```
// If this is an integer, increment it by four
if (is_int($int)) {
    $int += 4;
}

// If $bool is a string, print it out
// (does not print out anything)
if (is_string($bool)) {
    echo "String: $bool";
}
?>
```

Αν θέλετε να αναγκάσετε μια μεταβλητή να μετατραπεί σε έναν συγκεκριμένο τύπο, μπορείτε είτε να χρησιμοποιήσετε την cast στην μεταβλητή ή την **settype()** συνάρτηση πάνω της.

Σημειώστε ότι η μεταβλητή μπορεί να υπολογιστεί με διαφορετικές τιμές σε συγκεκριμένες περιπτώσεις, ανάλογα με το τι τύπου είναι κάθε φορά. Για περισσότερες πληροφορίες, δείτε το κομμάτι σχετικά με το Type Juggling. Επίσης, ίσως σας ενδιαφέρει να δείτε τους πίνακες σύγκρισης τύπων, καθώς δείχνουν παραδείγματα από συγκρίσεις σχετικές με διάφορους τύπους.

Booleans

Αυτός είναι ο ευκολότερος τύπος. Ένας **boolean** εκφράζει μια αληθινή τιμή. Μπορεί να είναι είτε **TRUE** είτε **FALSE**.

Σημείωση: Ο τύπος boolean εισήχθη στην PHP 4.

Σύνταξη

Για να καθορίσετε ένα boolean λεκτικό (literal), χρησιμοποιείστε είτε τη λέξη κλειδί **TRUE** ή την **FALSE**. Και οι δυο είναι case-insensitive.

```
<?php
$foo = True; // assign the value TRUE to $foo
?>
```

Συνήθως χρησιμοποιείτε κάποιο είδος τελεστή ο οποίος επιστρέφει μια **boolean** τιμή, και μετά την περνάει πάνω σε μια δομή ελέγχου.

```
<?php
// == is an operator which test
// equality and returns a boolean
if ($action == "show_version") {
    echo "The version is 1.23";
```

```
}

// this is not necessary...
if ($show_separators == TRUE) {
    echo "<hr>\n";
}

// ...because you can simply type
if ($show_separators) {
    echo "<hr>\n";
}
?>
```

Μετατρέποντας σε boolean

Για να μετατρέψετε μια τιμή σε τύπο **boolean**, χρησιμοποιείστε είτε την `(bool)` ή την `(boolean)` για cast (μετατροπή). Πάντως, στις περισσότερες περιπτώσεις δεν χρειάζεται να χρησιμοποιείτε την cast, αφού μια τιμή θα μετατραπεί αυτόματα αν ο τελεστής, η συνάρτηση ή μια δομή ελέγχου απαιτεί μια παράμετρο τύπου **boolean**.

Δείτε επίσης το Type Juggling.

Όταν μετατρέπονται σε **boolean**, οι ακόλουθες τιμές είναι **FALSE**

- Η ίδια η boolean γίνεται **FALSE**
- ο integer 0 (μηδέν)
- ο float 0.0 (μηδέν)
- το κενό string, και το string "0"
- ένας array με κανένα στοιχείο
- ένα object με κανένα μέλος
- ο ειδικός τύπος NULL (συμπεριλαμβανομένου των unset variables)

Οποιαδήποτε άλλη τιμή θεωρείται **TRUE** (συμπεριλαμβάνεται οποιοδήποτε resource).

Προειδοποίηση
Το `-1` θεωρείται **TRUE**, όπως και οποιοδήποτε άλλο μη-μηδενικός (είτε θετικός είτε αρνητικός) αριθμός!

```
<?php
echo gettype((bool) "");        //
bool(false)
echo gettype((bool) 1);         //
bool(true)
echo gettype((bool) -2);        //
bool(true)
echo gettype((bool) "foo");     //
bool(true)
echo gettype((bool) 2.3e5);     //
bool(true)
echo gettype((bool) array(12)); //
bool(true)
echo gettype((bool) array());   //
bool(false)
?>
```

Integers (Ακέραιοι)

Ένας **integer** είναι ένας αριθμός του συνόλου Z = {..., -2, -1, 0, 1, 2, ...}.

Δείτε επίσης: Ακέραιοι αυθαίρετου μεγέθους / GMP, Αριθμού κινητής υποδιαστολής, και Αυθαίρετη ακρίβεια / BCMath

Σύνταξη

Οι ακέραιοι μπορούν να καθοριστούν στο δεκαδικό (με βάση το 10), δεκαεξαδικό (με βάση το 16) ή οκταδικό (με βάση το 8) σύστημα, και προαιρετικά μπορεί να προστεθεί το πρόσημο (- ή +).

Αν χρησιμοποιείτε το οκταδικό σύστημα, πρέπει να γράφετε πριν το αριθμό το `0` (μηδέν), και αν χρησιμοποιείτε το δεκαεξαδικό πρέπει να γράφετε πριν τον αριθμό το `0x`.

Παράδειγμα 6-1. Integer literals (Λεκτικά ακεραίων)

```
<?php
$a = 1234; # decimal number
$a = -123; # a negative number
$a = 0123; # octal number (equivalent to 83 decimal)
$a = 0x1A; # hexadecimal number (equivalent to 26 decimal)
?>
```

Επισήμως, η πιθανή δομή για τα λεκτικά ακεραίων είναι:

```
<?php
decimal     : [1-9][0-9]*
            | 0

hexadecimal : 0[xX][0-9a-fA-F]+

octal       : 0[0-7]+

integer     : [+-]?decimal
            | [+-]?hexadecimal
            | [+-]?octal
?>
```

Το μέγεθος του ακεραίου εξαρτάται από την πλατφόρμα, ενώ μια μέγιστη τιμή περίπου 2 δισεκατομμυρίων είναι η συνηθισμένη τιμή (δηλαδή 32 bits προσημασμένα). Η PHP δεν υποστηρίζει μη προσημασμένους ακεραίους.

Υπερχείλιση ακεραίων (Integer overflow)

Αν προσδιορίσετε έναν αριθμός εκτός των ορίων του τύπου **integer**, θα διερμηνευτεί ως **float** (κινητής υποδιαστολής). Επίσης, αν κάνετε μια πράξη το αποτέλεσμα της οποίας δίνει αριθμό πέρα από τα όρια του τύπου **integer**, θα επιστραφεί αριθμός τύπου **float** (κινητής υποδιαστολής).

```
<?php
$large_number =  2147483647;
var_dump($large_number);
// output: int(2147483647)

$large_number =  2147483648;
var_dump($large_number);
// output: float(2147483648)

// this goes also for hexadecimal specified integers:
var_dump( 0x80000000 );
// output: float(2147483648)

$million = 1000000;
$large_number =  50000 * $million;
var_dump($large_number);
// output: float(50000000000)
?>
```

Προειδοποίηση
Δυστυχώς, υπήρχε ένα bug στην PHP με αποτέλεσμα αυτό να μην λειτουργεί παντα σωστά όταν υπάρχουν αρνητικοί αριθμοί. Για παράδειγμα: όταν κάνετε `-50000 * $million`, το αποτέλεσμα θα είναι `-429496728`. Σε περίπτωση βέβαια που και οι δυο τελεστές είναι θετικοί δεν υπάρχει πρόβλημα. Αυτό λύθηκε στην PHP 4.1.0.

Δεν υπάρχει τελεστής διαίρεσης στην PHP. Το `1/2` μετατρέπεται στον αριθμό **float** `0.5`. Μπορείτε να μετατρέψετε την τιμή σε ακέραιο να το στρογγυλοποιήσετε προς τα κάτω, ή να χρησιμοποιήσετε τη συνάρτηση **round()** .

```
<?php
var_dump(25/7);         // float(3.5714285714286)
var_dump((int) (25/7)); // int(3)
var_dump(round(25/7));  // float(4)
?>
```

Μετατρέποντας σε ακέραιο

Για να μετατρέψετε ρητά μια τιμή σε **integer**, χρησιμοποιήστε είτε το `(int)` είτε το `(integer)` για τη μετατροπή (cast). Πάντως, στις περισσότερες περιπτώσεις δεν χρειάζεται να χρησιμοποιήσετε την cast, αφού η τιμή θα μετατραπεί αυτόματα αν ένας τελεστής, μια συνάρτηση ή μια δομή ελέγχου απαιτεί μια παράμετρο τύπου

integer . Μπορείτε επίσης να μετατρέψετε μια τιμή σε ακέραιο μέσα στη συνάρτηση **intval()**.

Δείτε επίσης type-juggling.

Από booleans

FALSE θα προκύψει `0` (μηδέν), και από **TRUE** θα προκύψει `1` (ένα).

Από αριθμούς κινητής υποδιαστολής

Όταν γίνεται μετατροπή από αριθμού κινητής υποδιαστολής σε ακέραιο, ο αριθμός θα στρογγυλευτεί *προς το μηδέν*.

Αν ο αριθμός κινητής υποδιαστολής είναι εκτός των ορίων του ακεραίου (συνήθως `+/- 2.15e+9 = 2^31`), το αποτέλεσμα είναι απροσδιόριστο, αφού ο αριθμός κινητής υποδιαστολής δεν έχει αρκετή ακρίβεια για να δώσει ένα ακριβές ακέραιο αποτέλεσμα. Καμιά προειδοποίηση, ούτε κάποια άλλη υπενθύμιση γίνεται σε τέτοιες περιπτώσεις!

Προειδοποίηση

Ποτέ μην αλλάζετε τον τύπο μιας παράστασης σε **integer**, αφού αυτό μπορεί μερικές φορές να οδηγήσει σε μη αναμενόμενα αποτελέσματα.
 	
`<?php`
`echo (int) ( (0.1+0.7) * 10 ); // echoes 7!`
`?>`
 	
Για περισσότερες πληροφορίες δείτε σχετικά με τιςπροειδοποιήσεις για την ακρίβεια αριθμών κινητής υποδιαστολής.

Από strings

Δείτε Μετατροπή του String σε αριθμό

Από άλλους τύπους

Προσοχή
Ο τρόπος με τον οποίο γίνεται η μετατροπή σε ακέραιο δεν είναι καθορισμένος για άλλους τύπους. Προς το παρόν, η μετατροπή γίνεται θεωρώντας ότι η τιμή πρώτα μετατρέπεται σε boolean. Πάντως, *μην* βασίζεστε σε αυτόν τον τρόπο, αφού μπορεί να αλλάξει χωρίς προειδοποίηση.

Αριθμοί κινητής υποδιαστολής

Αριθμοί κινητής υποδιαστολής (συμβολίζονται και ως "floats", "doubles" or "real numbers") μπορούν να προσδιοριστούν χρησιμοποιώντας οποιονδήποτε από τους ακόλουθους τρόπους σύνταξης:

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

Formally:

```
LNUM          [0-9]+
DNUM          ([0-9]*[\.]{LNUM}) | ({LNUM}[\.][0-9]*)
EXPONENT_DNUM ( ({LNUM} | {DNUM}) [eE][+-]? {LNUM})
```

Το μέγετος ενός αριθμού κινητής υποδιαστολής δεν εξαρτάται από την πλατφόρμα, αν και ένας μέγιστος αριθμός περίπου ίσος με 1.8e308 με ακρίβεια σχεδόν 14 δεκαδικών ψηφίων είναι μια συνηθισμένη τιμή (αυτό είναι 64 bit IEEE format).

Ακρίβεια αριθμών κινητής υποδιαστολής
Συνηθίζεται σε απλές δεκαδικές παραστάσεις όπως `0.1` ή `0.7` να μην είναι εφικτή η μετατροπή τους σε εσωτερική δυαδική αντιστοίχιση χωρίς ένα μικρό χάσιμο σε ακρίβεια. Αυτό μπορεί να οδηγήσει σε διφορούμενα αποτελέσματα: για παράδειγμα, η `floor((0.1+0.7)*10)` συνήθως επιστρέφει `7` σε αντίθεση με το αναμενόμενο `8` αφού το αποτέλεσμα της εσωτερικής αναπαράστασης είναι κάτι σαν `7.9999999999...`. Αυτό σχετίζεται με το γεγονός ότι είναι αδύνατο να εκφράσουμε ακριβώς μερικές παραστάσεις σε δεκαδική μορφή με έναν πεπερασμένο αριθμό ψηφίων. Για παράδειγμα, το `1/3` στη δεκαδική μορφή γίνεται `0.3333333. . ..` Συνεπώς μην εμπιστεύεστε ποτέ αποτελέσματα αριθμών κινητής υποδιαστολής μέχρι το τελευταίο ψηφίο και ποτέ μη συγκρίνεται για ισότητα αριθμούς κινητής υποδιαστολής. Αν θέλετε πραγματικά μεγαλύτερη ακρίβεια, θα πρέπει να χρησιμοποιείτε τις συναρτήσεις αυθαίρετης μαθηματικής ακρίβειας ή τη συνάρτηση gmp .

Μετατρέποντας σε αριθμό κινητής υποδιαστολής

Για περισσότερες πληροφορίες σχετικά με το πότε και πώς τα strings μετατρέπονται σε αριθμούς κινητής υποδιαστολής, δείτε το τμήμα με τον τίτλο Μετατροπή των strings σε αριθμούς. Για τιμές άλλων τύπων, η μετατροπή είναι η ίδια με αυτή που θα είχε η τιμή αν μετατρεπόταν πρώτα σε ακέραιο και μετά σε αριθμό κινητής υποδιαστολής. Δείτε το τμήμα Μετατρέποντας σε ακέραιο για περισσότερες πληροφορίες.

Strings

Ένα **string** είναι μια σειρά χαρακτήρων. Στην PHP, ένας χαρακτήρας είανι το ίδιο με ένα byte, δηλαδή, υπάρχουν ακριβώς 256 διαφορετικοί πιθανοί χαρακτήρες. Αυτό επίσης σημαίνει ότι η PHP δεν υποστηρίζει Unicode. Δείτε την **utf8_encode()** και την **utf8_decode()** σχετικά με υποστήριξη Unicode.

Σημείωση: Δεν υπάρχει πρόβλημα για ένα string να γίνει πολύ μεγάλο. Δεν υπάρχει πρακτικά κάποιο όριο για το μέγεθος των strings που να επιβάλλει η PHP, συνεπώς δεν υπάρχει λόγος να ανησυχείτε για μεγάλα strings.

Σύνταξη

Ένα λεκτικό string μπορεί να προσδιοριστεί με τρεις διαφορετικούς τρόπους.

- με απλό εισαγωγικό
- με διπλό εισαγωγικό (quote)
- Σύνταξη heredoc

Μονό εισαγωγικό (Single quoted)

Ο ευκολότερος τρόπος για να ορίσετε ένα απλό string είναι να το βάλετε μέσα σε μονά εισαγωγικά (δηλαδή στον χαρακτήρα `'`).

Για να ορίσετε ένα απλό εισαγωγικό, θα χρειαστεί να προσθέσετε ένα backslash (`\`), όπως και σε πολλές άλλες γλώσσες. Αν το backslash πρέπει να εμφανιστεί πριν από ένα απλό εισαγωγικό ή στο τέλος του string, θα χρειαστεί να το διπλασιάσετε. Σημειώστε ότι αν προσπαθήσετε να αποφύγετε (escape) οποιονδήποτε άλλο χαρακτήρα, το backslash θα τυπωθεί! Συνήθως δεν υπάρχει ανάγκη να θέλουμε να αποφύγουμε (escape) την εμφάνιση του ίδιου του backslash.

Σημείωση: Στην PHP 3, όταν συμβαίνει αυτό θα εμφανιστεί μια προειδοποίηση στο επίπεδο `E_NOTICE` .

Σημείωση: Σε αντίθεση με τις δύο άλλες συντάξεις, οι μεταβλητές και οι ακολουθίες από escape για ειδικούς χαρακτήρες *δεν* θα επεκταθεί όταν εμφανίζονται σε strings που ορίζονται από απλά εισαγωγικά.

```
<?php
echo 'this is a simple string';

echo 'You can also have embedded newlines in
strings this way as it is
okay to do';

// Outputs: Arnold once said: "I'll be back"
echo 'Arnold once said: "I\'ll be back"';

// Outputs: You deleted C:\*.*?
```

```
echo 'You deleted C:\\*.*?';

// Outputs: You deleted C:\*.*?
echo 'You deleted C:\*.*?';

// Outputs: This will not expand: \n a newline
echo 'This will not expand: \n a newline';

// Outputs: Variables do not $expand $either
echo 'Variables do not $expand $either';
?>
```

Διπλά εισαγωγικά

Αν το string περικλείεται σε διπλά εισαγωγικά ("), η PHP καταλαβαίνει περισσότερες ακολουθίες από escape (escape sequences) για ειδικούς χαρακτήρες:

Πίνακας 6-1. Χαρακτήρες που εξαιρούνται (Escaped characters)

sequence	meaning
`\n`	linefeed (LF or 0x0A (10) in ASCII)
`\r`	carriage return (CR or 0x0D (13) in ASCII)
`\t`	horizontal tab (HT or 0x09 (9) in ASCII)
`\\`	backslash
`\$`	dollar sign
`\"`	double-quote
`\[0-7]{1,3}`	η ακολουθία των χαρακτήρων που ταιριάζουν στην κανονική έκφραση είναι ένας χαρακτήρας στο οχταδικό σύστημα
`\x[0-9A-Fa-f]{1,2}`	η ακολουθία των χαρακτήρων που ταιριάζουν στην κανονική έκφραση είναι ένας χαρακτήρας στο δεκαεξαδικό σύστημα

Επαναλαμβάνουμε ότι αν προσπαθήσετε να αποφύγετε (escape) οποιοδήποτε άλλο χαρακτήρα, το backslash θα τυπωθεί!

Αλλά το πιο σημαντικό χαρακτηριστικό των strings που ορίζονται σε διπλά εισαγωγικά είναι το γεγονός ότι τα ονόματα των μεταβλητών θα επεκταθούν. Δείτε το string parsing για λεπτομέρειες.

Heredoc

Ένας άλλος τρόπος για να ορίσουμε strings είναι χρησιμοποιώντας τη σύνταξη heredoc ("<<<"). Θα πρέπει να προσθέσουμε έναν identifier μετά τα <<<, στη συνέχεια το string, και μετά τον ίδιο identifier για να κλείσουμε την αναφορά.

Ο identifier κλεισίματος *πρέπει* να αρχίζει στην πρώτη στήλη της γραμμής. Επίσης, ο identifier που χρησιμοποιείται πρέπει να ακολουθεί τους ίδιους κανόνες ονοματολογίας όπως και οποιοδήποτε άλλο label στην PHP: πρέπει να περιέχει μόνο αλφαριθμητικούς χαρακτήρες και underscores, και πρέπει να αρχίζει με ένα μη αριθμητικό χαρακτήρα ή underscore.

Προειδοποίηση
Είναι πολύ σημαντικό να σημειώσουμε ότι η γραμμή με τον identifier κλεισίματος δεν περιέχει άλλους χαρακτήρες, εκτός *ίσως* από ένα ελληνικό ερωτηματικό (`;`). Αυτό σημαίνει ιδιαίτερα ότι ο identifier *μπορεί να μην βρίσκεται σε εσοχή (quote)*, και μπορεί να μην υπάρχουν spaces ή tabs μετά ή πριν το ελληνικό ερωτηματικό. Είναι επίσης σημαντικό να συνειδητοποιήσετε ότι ο πρώτος χαρακτήρας πριν τον identifier κλεισίματος πρέπει να είναι μια καινούρια γραμμή (newline) όπως αυτή ορίζεται από το λειτουργικό σας σύστημα. Αυτό είναι το `\r` στο Macintosh για παράδειγμα. Αν αυτός ο κανόνας δεν τηρείται και ο identifier κλεισίματος δεν είναι "ξεκάθαρος" τότε δεν θεωρείται ότι είναι identifier κλεισίματος και η PHP θα συνεχίσει να ψάχνει για έναν τέτοιο. Αν σ'αυτή την περίπτωση ένας κατάλληλος identifier κλεισίματος δεν βρεθεί τότε ένα parse error θα εμφανιστεί με τον αριθμό της γραμμής που τελειώνει το script.

Το Heredoc text συμπεριφέρεται ακριβώς όπως το double-quoted string, χωρίς όμως τα double-quotes. Αυτό σημαίνει πως δεν χρειάζεται να προσπαθείτε να κάνετε escape quotes στα έγγραφα σας εδώ, αλλά μπορείτε να χρησιμοποιείτε τους κώδικες για escape που αναφέρθηκαν παραπάνω. Οι μεταβλητές επεκτείνονται, αλλά η ίδια προσοχή πρέπει να δίνεται όταν εκφράζουμε σύνθετες μεταβλητές μέσα σε ένα τέτοιο έγγραφο όπως και με τα strings.

Παράδειγμα 6-2. Παράδειγμα Heredoc string quoting

```
<?php
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;

/* More complex example, with variables. */
class foo
{
    var $foo;
    var $bar;

    function foo()
    {
        $this->foo = 'Foo';
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
}

$foo = new foo();
$name = 'MyName';
```

```
echo <<<EOT
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital 'A': \x41
EOT;
?>
```

Σημείωση: Η υποστήριξη Heredoc προστέθηκε στην PHP 4.

Μεταγλώττιση Μεταβλητών

Όταν ένα string ορίζεται σε double quotes ή με heredoc, οι μεταβλητές μεταγλωττίζονται μέσα σ'αυτό.

Υπάρχουν δυο τρόποι σύνταξης, ο απλός και ο σύνθετος . Ο απλός τρόπος σύνταξης είναι ο πιο κοινός και βολικός, παρέχει έναν τρόπο για μεταγλώττιση μεταβλητής, μιας τιμής ενός **array**, ή μια ιδιότητας αντικειμένου.

Ο σύνθετος τρόπος σύνταξης εισήχθη στην PHP 4, και μπορεί να αναγνωριστεί από τα curly braces (άγκιστρα) που περιβάλουν την έκφραση.

Απλή σύνταξη

Αν το σύμβολο του δολαρίου (`$`) εμφανιστεί, ο parser (μεταγλωττιστής) θα πάρει όσο πιο πολλά tokens μπορεί για να σχηματίσει ένα έγκυρο όνομα μεταβλητής. Βάλτε το όνομα της μεταβλητής σε curly braces αν θέλετε να καθορίσετε ρητά το τέλος του ονόματος.

```
<?php
$beer = 'Heineken';
echo "$beer's taste is great"; // works, "'" is an invalid character
for varnames
echo "He drank some $beers";   // won't work, 's' is a valid
character for varnames
echo "He drank some ${beer}s"; // works
echo "He drank some {$beer}s"; // works
?>
```

Ομοιώς, μπορείτε να έχετε ένα **array** index ή μία ιδιότητα αντικειμένου για μεταγλώττιση. Στα ευρετήρια πινάκων, η αγκύλη κλεισίματος (`]`) ορίζει το τέλος του ευρετηρίου. Για τις ιδιότητες αντικειμένων οι ίδιοι κανόνες εφαρμόζονται όπως και στις απλές μεταβλητές, ενώ με τις ιδιότητες αντικειμένων δεν υπάρχει τέτοιο trick όπως αυτό με τις μεταβλητές.

```
<?php
// These examples are specific to using arrays inside of strings.
```

```
// When outside of a string, always quote your array string keys
// and do not use {braces} when outside of strings either.

// Let's show all errors
error_reporting(E_ALL);

$fruits = array('strawberry' => 'red', 'banana' => 'yellow');

// Works but note that this works differently outside string-quotes
echo "A banana is $fruits[banana].";

// Works
echo "A banana is {$fruits['banana']}.";

// Works but PHP looks for a constant named banana first
// as described below.
echo "A banana is {$fruits[banana]}.";

// Won't work, use braces.  This results in a parse error.
echo "A banana is $fruits['banana'].";

// Works
echo "A banana is " . $fruits['banana'] . ".";

// Works
echo "This square is $square->width meters broad.";

// Won't work. For a solution, see the complex syntax.
echo "This square is $square->width00 centimeters broad.";
?>
```

Για κάτι πιο πολύπλοκο, θα πρέπει να χρησιμοποιείτε τη σύνθετη σύνταξη.

Σύνθετη (curly) σύνταξη

Η σύνταξη αυτή δεν καλείται σύνθετη επειδή είναι η ίδια σύνθετη, αλλά επειδή μπορείτε να συμπεριλάβετε σύνθετες εκφράσεις με αυτόν τον τρόπο.

Στην πραγματικότητα, μπορείτε να συμπεριλάβετε οποιαδήποτε τιμή υπάρχει στο namespace των strings με αυτή τη σύνταξη. Απλά γράφετε την έκφραση με τον ίδιο τρόπο που θα τη γράφατε έξω από το string, και στη συνέχεια να την συμπεριλάβετε στα { και }. Αφού δεν μπορείτε να κάνετε escape στο '{', αυτή η σύνταξη θα αναγνωρίζεται μόνο όταν το $ ακολουθεί αμέσως το {. (Χρησιμοποιείστε το "{\$" ή το "\{$" για να πάρετε το λεκτικό "{$"). Μερικά παραδείγματα για να το κάνουν πιο ξεκάθαρο:

```
<?php
// Let's show all errors
error_reporting(E_ALL);

$great = 'fantastic';

// Won't work, outputs: This is { fantastic}
```

```
echo "This is { $great}";

// Works, outputs: This is fantastic
echo "This is {$great}";
echo "This is ${great}";

// Works
echo "This square is {$square->width}00 centimeters broad.";

// Works
echo "This works: {$arr[4][3]}";

// This is wrong for the same reason as $foo[bar] is wrong
// outside a string.  In otherwords, it will still work but
// because PHP first looks for a constant named foo, it will
// throw an error of level E_NOTICE (undefined constant).
echo "This is wrong: {$arr[foo][3]}";

// Works.  When using multi-dimensional arrays, always use
// braces around arrays when inside of strings
echo "This works: {$arr['foo'][3]}";

// Works.
echo "This works: " . $arr['foo'][3];

echo "You can even write {$obj->values[3]->name}";

echo "This is the value of the var named $name: {${$name}}";
?>
```

String που προσπελαύνονται από χαρακτήρες

Οι χαρακτήρες μέσα σε strings μπορούν να προσπελαστούν ορίζοντας το zero-based offset του επιθυμητού χαρακτήρα μετά το string σε curly braces.

Σημείωση: Για προς τα πίσω συμβατότητα, μπορείτε ακόμη να χρησιμοποιείτε array-braces για τον ίδιο σκοπό. Πάντως, αυτή η σύνταξη δεν συνίσταται στην PHP 4.

Παράδειγμα 6-3. Μερικά παραδείγματα από strings

```
<?php
// Get the first character of a string
$str = 'This is a test.';
$first = $str{0};

// Get the third character of a string
$third = $str{2};

// Get the last character of a string.
$str = 'This is still a test.';
$last = $str{strlen($str)-1};
?>
```

Χρήσιμες συναρτήσεις και τελεστές

Τα strings μπορούν να συννενωθούν χρησιμοποιώντας τον τελεστή '.' (τελεία). Σημειώστε ότι ο '+' (συν) τελεστής δε θα δουλέψει σ'αυτή την περίπτωση. Παρακαλώ δείτε το Τελεστές για Strings για περισσότερες πληροφορίες.

Υπάρχουν πολλές χρήσιμες συναρτήσεις για αλλαγές στα strings.

Δείτε το τμήμα για συναρτήσεις των strings για γενικές συναρτήσεις, τις συναρτήσεις κανονικών εκφράσεων για προχωρημένη αναζήτηση&αντικατάσταση (σε δυο εκδόσεις: Perl και POSIX extended).

Υπάρχουν επίσης συναρτήσεις για URL-strings, και συναρτήσεις για κρυπτογράφηση/αποκρυπτογράφηση strings (mcrypt και mhash).

Τέλος, αν ακόμη δεν βρήκατε αυτό που ψάχνατε, δείτε επίσης τις συναρτήσεις για τύπους χαρακτήρων.

Μετατρέποντας σε string

Μπορείτε να μετατρέψετε μια τιμή σε string χρησιμοποιώντας την `(string)` cast (μεατροπή), ή τη συνάρτηση **strval()**. Η μετρατοπή σε String γίνεται αυτόματα για σας στην εμβέλεια της έκφρασης όταν απαιτείται ένα string. Αυτό συμβαίνει όταν χρησιμοποιείτε τις **echo()** ή **print()** συναρτήσεις, ή όταν συγκρίνετε μια μεταβλητή τιμή με ένα string. Διαβάστε στο manual τα τμήματα σχετικά με Τύπους και Type Juggling για να καταλάβετε καλύτερα. Δείτε επίσης **settype()**.

Μια **boolean** **`TRUE`** τιμή μετατρέπεται στο string `"1"`, ενώ η **`FALSE`** τιμή αναπαρίσταται από το `""` (κενό string). Μ'αυτόν τον τρόπο μπορείτε να μετατρέψετε τις τιμές από boolean σε string και αντιστρόφως.

Ένας **integer** ή ένας αριθμός κινητής υποδιαστολής (**float**) όταν μετατρέπεται string αναπαρίσταται από τον αριθμό με τα ψηφία του (συμπεριλαμβάνεται το μέρος του εκθέτη για τους αριθμούς κινητής υποδιαστολής).

Οι Arrays μετατρέπονται πάντα στο string `"Array"`, έτσι δεν μπορείτε να εμφανίσετε το περιεχόμενο ενός **array** με την **echo()** ή την **print()** για να δείτε τι υπάρχει μέσα σ'αυτούς. Για να δείτε ένα στοιχείο, θα κάνετε κάτι όπως `echo $arr['foo']`. Δείτε παρακάτω για tips σχετικά με την εμφάνιση ολόκληρου του περιεχομένου.

Τα Objects μετατρέπονται πάντα στο string `"Object"`. Αν θέλετε να εκτυπώσετε τη τιμή ενός μέλους της μεταβλητής ενός **object** για λόγους debugging, διαβάστε τις παρακάτω παραγράφους. Αν θέλετε να βρείτε το όνομα της κλάσης της οποίας ένα object είναι στιγμιότυπο (instance), χρησιμοποιείστε την **get_class()**.

Τα Resources πάντα μετατρέπονται σε strings με τη δομή `"Resource id #1"` όπου το `1` είναι ο μοναδικός αριθμός του **resource** που ανατίθεται από την PHP κατά τη

διάρκεια της εκτέλεσης. Αν θέλετε να πάρετε τον τύπο του resource, χρησιμοποιείστε την **get_resource_type()**.

Το **NULL** μετατρέπεται πάντα σε κενό string.

Όπως μπορείτε να δείτε παραπάνω, η εκτύπωση των arrays, των objects ή των resources δεν σας παρέχει χρήσιμες πληροφορίες σχετικά τις ίδιες τις τιμές. Δείτε τις συναρτήσεις **print_r()** και **var_dump()** για καλύτερους τρόπους εμφάνισης των τιμών για το debugging.

Μπορείτε επίσης να μετατρέψετε τις τιμές της PHP σε strings και να τις αποθηκεύσετε μόνιμα. Αυτή η μέθοδος ονομάζεται serialization, και μπορεί να γίνει με τη συνάρτηση **serialize()**. Μπορείτε επίσης να κάνετε serialize τις PHP values σε XML δομές, αν έχετε προσθέσει υποστήριξη για WDDX κατά τη διάρκεια του setup της PHP.

Μετατροπή των Strings σε αριθμούς

Όταν ένα string υπολογίζεται σαν αριθμητική τιμή, η τιμή που προκύπτει και ο τύπος της ορίζονται ως ακολούθως.

Το string θα υπολογιστεί ως **float** αν περιέχει οποιοδήποτε από τους χαρακτήρες '.', 'e', or 'E'. Διαφορετικά, θα υπολογιστεί ως ακέραιος.

Η τιμή δίνεται από το αρχικό μέρος του string. Αν το string αρχίζει με ένα έγκυρο αριθμητικό δεδομένο, αυτή θα είναι και η τιμή που θα χρησιμοποιηθεί. Διαφορετικά, η τιμή θα είναι 0 (μηδέν). Τα έγκυρα αριθμητικά δεδομένα είναι ένα προαιρετικό σύμβολο, ακολουθούμενο από ένα ή περισσότερα ψηφία (προαιρετικά περιλαμβάνουν ένα δεκαδικό σημείο), ακολουθούμενο από ένα προαιρετικό εκθετικό. Το εκθετικό είναι το 'e' ή το 'E' ακολουθούμενο από ένα ή περισσότερα ψηφία.

```
<?php
$foo = 1 + "10.5";                // $foo is float (11.5)
$foo = 1 + "-1.3e3";              // $foo is float (-1299)
$foo = 1 + "bob-1.3e3";           // $foo is integer (1)
$foo = 1 + "bob3";                // $foo is integer (1)
$foo = 1 + "10 Small Pigs";       // $foo is integer (11)
$foo = 4 + "10.2 Little Piggies"; // $foo is float (14.2)
$foo = "10.0 pigs " + 1;          // $foo is float (11)
$foo = "10.0 pigs " + 1.0;        // $foo is float (11)
?>
```

Για περισσότερες πληροφορίες σχετικά με αυτή τη μετατροπή, δείτε το manual του Unix στη σελίδα για strtod(3).

Αν θέλετε να ελένξετε κάποιο από τα παραδείγματα σ'αυτό το τμήμα, μπορείτε να κάνετε cut και paste στα παραδείγματα και να εισάγετε την ακόλουθη γραμμή για να δείτε μόνοι σας τι συμβαίνει:

```
<?php
echo "\$foo==$foo; type is " . gettype ($foo) . "<br />\n";
?>
```

Μην περιμένετε να πάρετε τον κώδικα από έναν χαρακτήρα απλά μετατρέποντας τον σε ακέραιο (όπως θα κάνατε στη C για παράδειγμα). Χρησιμοποιήστε τις συναρτήσεις **ord()** και **chr()** για μετατροπές ανάμεσα σε charcodes και characters.

Arrays

Ένας array στην PHP είναι στην πραγματικότητα ένας ταξινομημένος χάρτης (map). Ένας χάρτης είναι ένας τύπος που αντιστοιχεί τις *τιμές* σε *κλειδιά*. Αυτός ο τύπος έχει βελτιστοποιηθεί με πολλούς τρόπους, έτσι ώστε να μπορείτε να τον χρησιμοποιήσετε σαν πραγματικό array, ή ως λίστα (vector), hashtable (το οποίο είναι μια υλοποίηση ενός map), ευρετήριο, συλλογή, στοίβα, ουρά και πιθανώς και άλλα. Επειδή μπορείτε να έχετε και άλλον PHP-array ως τιμή, μπορείτε επίσης σχετικά εύκολα να προσομοιώσετε δέντρα (trees).

Η εξήγηση τέτοιων δομών είναι πέρα από τους σκοπούς αυτού του manual, αλλά θα βρείτε τουλάχιστο ένα παράδειγμα για κάθε μια από αυτές τις δομές. ΓΙα περισσότερες πληροφορίες σας παραπέμπουμε σε εξωτερική βιβλιογραφία σχετικά με αυτό το ευρύ θέμα.

Σύνταξη

Ορίζοντας array()

Ένας **array** μπορεί να δημιουργηθεί από τη γλωσσική δομή (language-construct) **array()**. Απαιτείται ένας ορισμένος αριθμός από `key => ζεύγη τιμών` που χωρίζονται με κόμματα.

```
array( [key =>] value
     , ...
     )
// key may be an integer or string
// value may be any value

<?php
$arr = array("foo" => "bar", 12 => true);

echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

Ένα `κλειδί (key)` είναι είτε **integer** είτε **string**. Αν ένα key είναι η standard αναπαράσταση ενός **integer**, τότε θα ερμηνευθεί ως τέτοια (π.χ. το `"8"` θα ερμηνευθεί ως `8`, ενώ το `"08"` θα ερμηνευθεί ως `"08"`). Δεν υπάρχουν διαφορετικοί indexed και

associative τύποι από arrays στην PHP, υπάρχει μόνο ένας τύπος array, που μπορεί να περιέχει τόσο ακέραια όσο και string ευρετήρια.

Μια τιμή μπορεί να είναι οποιουδήποτε PHP τύπου.

```
<?php
$arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));

echo $arr["somearray"][6];    // 5
echo $arr["somearray"][13];   // 9
echo $arr["somearray"]["a"];  // 42
?>
```

Αν παραλείψετε ένα κλειδί, το μέγιστο του ακέραιου-ευρετηρίου λαμβάνεται, και το νέο κλειδί θα είναι αυτό το μέγιστο + 1. Αν καθορίσετε ένα κλειδί που του έχει ήδη ανατεθεί μια τιμή, αυτή η τιμή θα επικαλυφθεί από τη νέα (overwritten).

```
<?php
// This array is the same as ...
array(5 => 43, 32, 56, "b" => 12);

// ...this array
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

Προειδοποίηση
Όσον αφορά την PHP 4.3.0, η συμπεριφορά παραγωγής του index που περιγράφηκε παραπάνω έχει αλλάξει. Τώρα, αν προσθέσετε σε έναν πίνακα του οποίου το τρέχον μέγιστο κλειδί είναι αρνητικό, τότε το επόμενο κλειδί που θα δημιουργηθεί θα είναι μηδέν (`0`). Παλιότερα, το καινούριο index θα είχε οριστεί στο μεγαλύτερο υπάρχον κλειδί + 1, όπως συμβαίνει και με τις θετικές τιμές.

Χρησιμοποιώντας το **`TRUE`** ως κλειδί θα υπολογιστεί ο ακέραιος `1` ως κλειδί.
Χρησιμοποιώντας το **`FALSE`** ως κλειδί θα υπολογιστεί ο ακέραιος `0` ως κλειδί.
Χρησιμοποιώντας το `NULL` ως κλειδί θα έχουμε ως αποτέλεσμα ένα κενό string.
Χρησιμοποιώντας ένα κενό string ως κλειδί θα δημιουργηθεί (ή επανεγγραφεί) ένα κλειδί με ένα κενό string και η τιμή του δε θα είναι η ίδια με αυτή που θα είχαμε αν χρησιμοποιούσαμε κενές παρενθέσεις.

Δεν μπορείτε να χρησιμοποιήσετε arrays ή objects ως κλειδιά. Αν το κάνετε θα εμφανιστεί η προειδοποίηση: `Illegal offset type`.

Δημιουργώντας/Αλλάζοντας με την square-bracket σύνταξη

Μπορείτε επίσης να αλλάξετε έναν υπάρχον array, ορίζοντας ρητά τιμές σ'αυτόν.

Αυτό γίνεται αναθέτοντας τιμές στον array ενώ συγχρόνως καθορίζεται το κλειδί στις παρενθέσεις. Μπορείτε βέβαια να παραλήψετε το κλειδί και να προσθέσετε ένα κενό ζευγάρι από παρενθέσεις ("[]") στο όνομα της μεταβλητής στην περίπτωση αυτή.

```
$arr[key] = value;
$arr[] = value;
// key is either string or nonnegative integer
// value can be anything
```

Αν η $arr δεν υπάρχει ακόμη, θα δημιουργηθεί. Αυτός είναι ένας εναλλακτικός τρόπος για να καθορίσετε έναν array. Για να αλλάξετε μια συγκεκριμένη τιμή, απλά αναθέστε μια νέα τιμή σε ένα στοιχείο που ορίζεται με το κλειδί του. Αν θέλετε να αφαιρέσετε ένα ζευγάρι κλειδιού/τιμής, θα χρειαστεί να κάνετε **unset()** σ'αυτό.

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

$arr[] = 56;    // This is the same as $arr[13] = 56;
                // at this point of the script

$arr["x"] = 42; // This adds a new element to
                // the array with key "x"

unset($arr[5]); // This removes the element from the array

unset($arr);    // This deletes the whole array
?>
```

Σημείωση: Όπως έχει προαναφερθεί, αν γράψετε τις παρανεθέσεις χωρίς να ορίσετε κάποιο κλειδί, τότε ο μέγιστος από τους υπάρχοντες ακεραίους λαμβάνεται, και το καινούριο κλειδί θα είναι αυτή η μέγιστη τιμή + 1 . Αν δεν υπάρχει κάποιος ακέραιος ακόμη, το κλειδί θα είναι 0 (μηδέν). Αν ορίσετε ένα κλειδί που έχει ήδη μια τιμή αυτή η τιμή θα γραφτεί από πάνω (overwritten).

Προειδοποίηση
Όσον αφορά την PHP 4.3.0, η συμπεριφορά παραγωγής του index που περιγράφηκε παραπάνω έχει αλλάξει. Τώρα, αν προσθέσετε σε έναν πίνακα του οποίου το τρέχον μέγιστο κλειδί είναι αρνητικό, τότε το επόμενο κλειδί που θα δημιουργηθεί θα είναι μηδέν (0). Παλιότερα, το καινούριο index θα είχε οριστεί στο μεγαλύτερο υπάρχον κλειδί + 1, όπως συμβαίνει και με τις θετικές τιμές.

Σημειώστε ότι το μέγιστο ακέραιο κλειδί που χρησιμοποιείται *δεν χρειάζεται απαραιτήτως να υπάρχει στον πίνακα*. Απλά πρέπει να υπήρχε στον πίνακα κάποια στιγμή από την τελευταία φορά που έγινε στον πίνακα αναταξινόμηση (re-index). Το ακόλουθο παράδειγμα δείχνει τα παραπάνω:

```
<?php
// Create a simple array.
$array = array(1, 2, 3, 4, 5);
print_r($array);
```

```
// Now delete every item, but leave the array itself intact:
foreach ($array as $i => $value) {
    unset($array[$i]);
}
print_r($array);

// Append an item (note that the new key is 5, instead of 0 as you
// might expect).
$array[] = 6;
print_r($array);

// Re-index:
$array = array_values($array);
$array[] = 7;
print_r($array);
?>
```

Το παραπάνω παράδειγμα θα παρήγαγε το ακόλουθο αποτέλεσμα:

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 4
    [4] => 5
)
Array
(
)
Array
(
    [5] => 6
)
Array
(
    [0] => 6
    [1] => 7
)
```

Χρήσιμες συναρτήσεις

Υπάρχουν αρκετές χρήσιμες συναρτήσεις για να δουλέψετε με arrays, δείτε το τμήμα Συναρτήσεις για arrays .

Σημείωση: Η συνάρτηση **unset()** σας επιτρέπει να κάνετε unset στα κλειδιά ενός array. Πρέπει να λάβετε υπόψη ότι ο array ΔΕΝ θα ξαναταξινομηθεί. Μόνο αν χρησιμοποιήσετε το "usual integer indices" (αρχίζοντας από το μηδέν, και αυξάνοντας κατά ένα), θα μπορέσετε να πετύχετε την αναταξινόμηση χρησιμοποιώντας την **array_values()**.

```
<?php
$a = array(1 => 'one', 2 => 'two', 3 => 'three');
unset($a[2]);
```

```
/* will produce an array that would have been defined as
   $a = array(1 => 'one', 3 => 'three');
   and NOT
   $a = array(1 => 'one', 2 =>'three');
*/

$b = array_values($a);
// Now b is array(1 => 'one', 2 =>'three')
?>
```

Η δομή ελέγχου foreach υπάρχει ειδικά για τους arrays. Παρέχει έναν εύκολο τρόπο για να προσπελαύνετε έναν array.

Τι κάνουν και τι δεν κάνουν οι Arrays

Γιατί είναι η `$foo[bar]` λάθος?

Πρέπει πάντα να χρησιμοποιείτε εισαγωγικά έξω από ένα ευρετήριο ενός associative array. Για παράδειγμα, χρησιμοποιήστε την $foo['bar'] και όχι την $foo[bar]. Αλλά γιατί είναι η $foo[bar] λανθασμένη; Ίσως έχετε δει τον ακόλουθο τρόπο σύνταξης σε παλιότερα scripts:

```
<?php
$foo[bar] = 'enemy';
echo $foo[bar];
// etc
?>
```

Αυτό είναι λάθος, αλλά δουλεύει. Τότε, γιατί είναι λάθος? Ο λόγος είναι ότι αυτός ο κώδικας έχει μια απροσδιόριστη σταθερά (την bar) παρά ένα string ('bar' - προσέξτε τα εισαγωγικά), και η PHP ίσως μελλοντικά ορίσει σταθερές οι οποίες, δυστυχώς για τον κώδικα σας, έχουν το ίδιο όνομα. Δουλεύει, επειδή η PHP αυτόματα μετατρέπει ένα *bare string* (είναι ένα string χωρίς εισαγωγικά που δεν αντιστοιχεί σε κάποιο γνωστό σύμβολο) σε string που περιέχει το bare string. Για παράδειγμα, αν δεν υπάρχει κάποιο ορισμένη σταθερά με το όνομα **`bar`**, τότε η PHP θα αντικαταστήσει το string `'bar'` και θα χρησιμοποιεί αυτό.

Σημείωση: Αυτό δε σημαίνει να βάζετε *πάνταalways* εισαγωγικά στο κλειδί. Δεν θέλετε να βάζετε εισαγωγικά στα κλειδιά που είναι σταθερές ή μεταβλητές, καθώς αυτό θα εμποδίζει την PHP από το να τα μεταγλωττίσει.

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', true);
ini_set('html_errors', false);
// Simple array:
$array = array(1, 2);
$count = count($array);
for ($i = 0; $i < $count; $i++) {
    echo "\nChecking $i: \n";
    echo "Bad: " . $array['$i'] . "\n";
```

```
    echo "Good: " . $array[$i] . "\n";
    echo "Bad: {$array['$i']}\n";
    echo "Good: {$array[$i]}\n";
}
?>
```

Σημείωση: Το αποτέλεσμα από το παραπάνω θα είναι:

```
Checking 0:
Notice: Undefined index:  $i in /path/to/script.html on line 9
Bad:
Good: 1
Notice: Undefined index:  $i in /path/to/script.html on line 11
Bad:
Good: 1

Checking 1:
Notice: Undefined index:  $i in /path/to/script.html on line 9
Bad:
Good: 2
Notice: Undefined index:  $i in /path/to/script.html on line 11
Bad:
Good: 2
```

Περισσότερα παραδείγματα για να δείξουν το παραπάνω:

```
<?php
// Let's show all errors
error_reporting(E_ALL);

$arr = array('fruit' => 'apple', 'veggie' => 'carrot');

// Correct
print $arr['fruit'];  // apple
print $arr['veggie']; // carrot

// Incorrect.  This works but also throws a PHP error of
// level E_NOTICE because of an undefined constant named fruit
//
// Notice: Use of undefined constant fruit - assumed 'fruit' in...
print $arr[fruit];    // apple

// Let's define a constant to demonstrate what's going on.  We
// will assign value 'veggie' to a constant named fruit.
define('fruit', 'veggie');

// Notice the difference now
print $arr['fruit'];  // apple
print $arr[fruit];    // carrot

// The following is okay as it's inside a string.  Constants are not
// looked for within strings so no E_NOTICE error here
print "Hello $arr[fruit]";      // Hello apple

// With one exception, braces surrounding arrays within strings
// allows constants to be looked for
print "Hello {$arr[fruit]}";    // Hello carrot
```

```
print "Hello {$arr['fruit']}";  // Hello apple

// This will not work, results in a parse error such as:
// Parse error: parse error, expecting T_STRING' or T_VARIABLE' or
T_NUM_STRING'
// This of course applies to using autoglobals in strings as well
print "Hello $arr['fruit']";
print "Hello $ GET['foo']";

// Concatenation is another option
print "Hello " . $arr['fruit']; // Hello apple
?>
```

Όταν ενεργοποιήσετε το **error_reporting()** να εμφανίζει **E_NOTICE** για level errors (όπως το να τεθεί στη σταθερά **E_ALL**) τότε θα δείτε αυτά τα λάθη. Η προκαθορισμένη ρύθμιση είναι, το error_reporting να μην έχει ενεργοποιημένη τη ρύθμιση για εμφάνιση τους.

Όπως έχει δηλωθεί στο τμήμα σύνταξης , πρέπει να υπάρχει μια έκφραση μεταξύ στις αγκύλες ('[' και ']'). Αυτό σημαίνει ότι μπορείτε να γράψετε πράγματα όπως αυτό:

```
<?php
echo $arr[somefunc($bar)];
?>
```

Αυτό είναι ένα παράδειγμα χρήσης μιας συνάρτησης που επιστρέφει κάποια τιμή ως δείκτη πίνακα (array index). Η PHP επίσης γνωρίζει τις σταθερές, όπως θα έχετε δει `E_*` και προηγουμένως.

```
<?php
$error_descriptions[E_ERROR]   = "A fatal error has occured";
$error_descriptions[E_WARNING] = "PHP issued a warning";
$error_descriptions[E_NOTICE]  = "This is just an informal notice";
?>
```

Σημειώστε ότι η `E_ERROR` είναι επίσης ένας έγκυρος identifier, όπως η `bar` στο πρώτο παράδειγμα. Αλλά το τελευταίο παράδειγμα είναι στην πραγματικότητα το ίδιο με το να γράφαμε:

```
<?php
$error_descriptions[1] = "A fatal error has occured";
$error_descriptions[2] = "PHP issued a warning";
$error_descriptions[8] = "This is just an informal notice";
?>
```

επειδή η `E_ERROR` ισούται με `1`, κτλ.

Όπως έχουμε ήδη εξηγήσει σε προηγούμενα παραδείγματα, η `$foo[bar]` δουλεύει ακόμη αλλά είναι λάθος. Δουλεύει, επειδή `bar` εξαιτίας της σύνταξης της αναμένεται να είναι μια σταθερή έκφραση. Πάντως, σ'αυτή την περίπτωση καμία σταθερά με το όνομα `bar` δεν υπάρχει. Η PHP τώρα υποθέτει ότι εννοούσατε την `bar` κυριολεκτικά, όπως το string `"bar"`, αλλά έχετε ξεχάσει να γράψετε τα εισαγωγικά.

Τότε γιατί είναι κακή;

Σε κάποια στιγμή στο μέλλον, η ομάδα της PHP ίσως θελήσει να προσθέσει και κάποια άλλη σταθερά ή keyword, ή μπορεί εσείς να θέλετε να εισάγετε κάποια σταθερά στην εφαρμογή σας, και τότε να αντιμετωπίσετε πρόβλημα. Για παράδειγμα, δεν μπορείτε ήδη να χρησιμοποιήσετε τις λέξεις `empty` και `default` μ'αυτόν τον τρόπο, αφού είναι ιδιαίτερες δεσμευμένες λέξεις.

Σημείωση: Για επανάληψη, μέσα σε ένα **string** με διπλα εισαγωγικά, είναι έγκυρο να μην περιλαμβάνετε array indexes με εισαγωγικά, συνεπώς το `"$foo[bar]"` είναι έγκυρο. Δείτε τα παραπάνω παραδείγματα για λεπτομέρειες σχετικά με το γιατί όπως επίσης το τμήμα σχετικά με το πέρασμα μεταβλητών σε strings.

Μετατρέποντας σε array

Για οποιοδήποτε από τους τύπους: **integer**, **float**, **string**, **boolean** και **resource**, αν μετατρέψετε μια τιμή σε **array**, παίρνετε έναν array με ένα στοιχείο (με index 0), το οποίο είναι η βαθμωτή τιμή με την οποία αρχίσατε.

Αν μετατρέψετε ένα **object** σε έναν array, παίρνετε τις ιδιότητες (μεταβλητές μελών) αυτού του αντικειμένου ως στοιχεία του array. Τα κλειδιά είναι τα ονόματα των μεταβλητών μελών.

Αν μετατρέψετε μια **NULL** τιμή σε array, παίρνετε έναν κενό array.

Παραδείγματα

Ο τύπος array στην PHP είναι πολύ ευπροσάρμοστος, έτσι εδώ θα δείτε μερικά παραδείγματα που θα σας δείξουν τις πολλές δυνατότητες των arrays.

```
<?php
// this
$a = array( 'color' => 'red',
            'taste' => 'sweet',
            'shape' => 'round',
            'name'  => 'apple',
                       4        // key will be 0
          );

// is completely equivalent with
$a['color'] = 'red';
$a['taste'] = 'sweet';
$a['shape'] = 'round';
$a['name']  = 'apple';
$a[]        = 4;        // key will be 0

$b[] = 'a';
$b[] = 'b';
$b[] = 'c';
// will result in the array array(0 => 'a' , 1 => 'b' , 2 => 'c'),
// or simply array('a', 'b', 'c')
```

```
?>
```

Παράδειγμα 6-4. Χρησιμοποιώντας την array()

```
<?php
// Array as (property-)map
$map = array( 'version'    => 4,
              'OS'         => 'Linux',
              'lang'       => 'english',
              'short_tags' => true
            );

// strictly numerical keys
$array = array( 7,
                8,
                0,
                156,
                -10
              );
// this is the same as array(0 => 7, 1 => 8,
...)

$switching = array(         10, // key = 0
                     5    =>  6,
                     3    =>  7,
                     'a'  =>  4,
                              11, // key = 6
(maximum of integer-indices was 5)
                     '8'  =>  2, // key = 8
(integer!)
                     '02' => 77, // key = '02'
                     0    => 12  // the value 10
will be overwritten by 12
                   );

// empty array
$empty = array();
?>
```

Παράδειγμα 6-5. Collection (Συλλογή)

```
<?php
$colors = array('red', 'blue', 'green', 'yellow');

foreach ($colors as $color) {
    echo "Do you like $color?\n";
}

/* output:
Do you like red?
Do you like blue?
Do you like green?
Do you like yellow?
*/
?>
```

Σημειώστε ότι προς το παρόν δεν είναι δυνατό να αλλάξουμε τις τιμές του array άμεσα σε ένα τέτοιο loop. Δείτε το ακόλουθο:

Παράδειγμα 6-6. Collection

```
<?php
foreach ($colors as $key => $color) {
    // won't work:
    //$color = strtoupper($color);

    // works:
    $colors[$key] = strtoupper($color);
}
print_r($colors);

/* output:
Array
(
    [0] => RED
    [1] => BLUE
    [2] => GREEN
    [3] => YELLOW
)
*/
?>
```

Αυτό το παράδειγμα δημιουργεί έναν one-based array.

Παράδειγμα 6-7. One-based index

```
<?php
$firstquarter  = array(1 => 'January', 'February', 'March');
print_r($firstquarter);

/* output:
Array
(
    [1] => 'January'
    [2] => 'February'
    [3] => 'March'
)
*/
```

Παράδειγμα 6-8. Γεμίζοντας έναν array

```
// fill an array with all items from a directory
$handle = opendir('.');
while (false !== ($file = readdir($handle))) {
    $files[] = $file;
}
closedir($handle);
?>
```

Οι arrays είναι διατεταγμένοι. Μπορείτε επίσης να αλλάξετε τη σειρά χρησιμοποιώντας διάφορες συναρτήσεις ταξινόμησης. Δείτε το τμήμα συναρτήσεις για arrays για περισσότερες πληροφορίες. Μπορείτε να μετρήσετε τον αριθμό των στοιχείων σε ένα array χρησιμοποιώντας τη συνάρτηση **count()** .

Παράδειγμα 6-9. Array ταξινόμησης

```
<?php
sort($files);
print_r($files);
?>
```

Επειδή η τιμή ενός array μπορεί να είναι οτιδήποτε, μπορεί επίσης να είναι ένας άλλος array. Μ'αυτόν τον τρόπο μπορείτε να κάνετε αναδρομικούς και πολυδιάστατους arrays.

Παράδειγμα 6-10. Αναδρομικοι και πολυδιάστατοι arrays

```
<?php
$fruits = array ( "fruits"  => array ( "a" => "orange",
                                       "b" => "banana",
                                       "c" => "apple"
                                     ),
                  "numbers" => array ( 1,
                                       2,
                                       3,
                                       4,
                                       5,
                                       6,
                                     ),
                  "holes"   => array (      "first",
                                       5 => "second",
                                            "third"
                                     )
                );

// Some examples to address values in the array above
echo $fruits["holes"][5];    // prints "second"
echo $fruits["fruits"]["a"]; // prints "orange"
unset($fruits["holes"][0]);  // remove "first"

// Create a new multi-dimensional array
$juices["apple"]["green"] = "good";
?>
```

Πρέπει να προσέχετε επειδή οι αναθέσεις των array πάντα εμπλέκουν αντιγραφή τιμών. Θα χρειαστεί να χρησιμοποιήσετε τον τελεστή αναφοράς για να αντιγράψετε έναν array με αναφορά.

```
<?php
$arr1 = array(2, 3);
$arr2 = $arr1;
$arr2[] = 4; // $arr2 is changed,
             // $arr1 is still array(2, 3)

$arr3 = &$arr1;
$arr3[] = 4; // now $arr1 and $arr3 are the same
?>
```

Objects

Αρχικοποίηση αντικειμένου

Για να αρχικοποιήσετε ένα αντικείμενο, χρησιμοποιείτε τη δήλωση `new` για να δημιουργήσετε ένα στιγμιότυπο του αντικειμένου σε μια μεταβλητή.

```
<?php
class foo
{
    function do_foo()
    {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();
?>
```

Για μια πλήρη ανάλυση, παρακαλώ διαβάστε το τμήμα σχετικά με Classes και Objects.

Μετατρέποντας σε object

Αν ένα object μετατρέπεται σε άλλο object, αυτό σημαίνει ότι δεν αλλάζει. Αν μια τιμή οποιοδήποτε άλλου τύπου μετατρέπεται σε object, τότε ένα καινούριο στιγμιότυπο του `stdClass` built στην class δημιουργείται. Αν η τιμή ήταν null, το νέο στιγμιότυπο θα είναι κενό. Για οποιαδήποτε άλλη τιμή, μια μεταβλητή μέλους ονομαζόμενη `scalar` θα περιέχει την τιμή.

```
<?php
$obj = (object) 'ciao';
echo $obj->scalar;  // outputs 'ciao'
?>
```

Resource

Μια resource είναι μια ειδική μεταβλητή, που κρατάει μια αναφορά σε μια εξωτερική resource. Οι resources δημιουργούνται και χρησιμοποιούνται από ειδικές συναρτήσεις. Δείτε το appendix για μια λίστα όλων αυτών των συναρτήσεων και των αντίστοιχων resource τύπων.

Σημείωση: Ο τύπος resource εισήχθη στην PHP 4

Μετατρέποντας σε resource

Επειδή οι resource τύποι έχουν ειδικούς handlers για ανοιγμένα αρχεία, database connections, image canvas areas και παρόμοια, δεν μπορείτε να μετατρέψετε οποιαδήποτε τιμή σε resource.

Ελευθερώνοντας resources

Εξαιτίας του reference-counting συστήματος που εισήχθη με την Zend-engine της PHP4, γίνεται αυτόματη ανίχνευση πότε σταματάει να γίνεται αναφορά σε ένα resource (όπως και στη Java). Σ'αυτή την περίπτωση, όλες οι resources που χρησιμοποιούνταν γι'αυτή τη resource ελευθερώνονται από τον garbage collector. Γι'αυτό το λόγο, είναι σπάνια αναγκαίο να ελευθερώσετε τη μνήμη manually χρησιμοποιώντας κάποια συνάρτηση όπως τη free_result.

Σημείωση: Τα Persistent database links είναι ιδιαίτερα, *δεν* καταστρέφονται από τον garbage collector. Δείτε επίσης το τμήμα σχετικά με σταθερές (persistent) συνδέσεις.

NULL

Η ειδική τιμή **NULL** σημαίνει ότι μια μεταβλητή δεν έχει τιμή. Η **NULL** είναι η μόνη πιθανή τιμή του τύπου **NULL**.

Σημείωση: Ο τύπος null εισήχθη στην PHP 4

Μια μεταβλητή θεωρείται ότι είναι **NULL** αν

- έχει ανατεθεί στη σταθερά **NULL**.
- δεν της έχει τεθεί ακόμη καμία τιμή.
- της έχει γίνει **unset()**.

Σύνταξη

Υπάρχει μόνο μια τιμή τύπου **NULL**, και αυτή είναι το case-insensitive keyword **NULL**.

```
<?php
$var = NULL;

?>
```

Δείτε επίσης **is_null()** και **unset()**.

Ψευδο-τύποι που χρησιμοποιήθηκαν σ'αυτό το documentation

mixed

`mixed` δείχνει ότι μια παράμετρος μπορεί να δεχθεί πολλαπλούς (αλλά όχι απαραίτητα όλους) τύπους.

Η **gettype()** για παράδειγμα θα δεχθεί όλους τους τύπους της PHP, ενώ η **str_replace()** θα δεχθεί μόνο strings και arrays.

---

number

Η `number` δείχνει ότι μια παράμετρος μπορεί να είναι είτε **integer** είτε **float**.

---

callback

Μειρκές συναρτήσεις όπως η **call_user_function()** ή η **usort()** δέχονται callback συναρτήσεις οριζόμενες από το χρήστη ως παράμετροι. Οι callback συναρτήσεις δεν μπορούν να είναι απλές συναρτήσεις αλλά επίσης object methods που περιέχουν static class methods.

Μια συνάρτηση σε PHP καλείται απλά με το όνομα της ως string. Μπορείτε να περάσετε οποιαδήποτε builtin ή οριζόμενη από το χρήστη συνάρτηση χρησιμοποιώντας exception από **array()**, **echo()**, **empty()**, **eval()**, **exit()**, **isset()**, **list()**, **print()** και **unset()**.

Μια μέθοδος ενός αντικειμένου που έχει κάποιο στιγμιότυπο περνιέται ως array που περιέχει ένα object ως στοιχείο με index 0 και ένα όνομα μεθόδου ως στοιχείο με index 1.

Οι Static class methods μπορούν επίσης να περαστούν χωρίς να δημιουργήσουμε στιγμιότυπο ενός object αυτής της class περνώντας το όνομα της κλάσης αντί για ένα object όπως το element με index 0.

Παράδειγμα 6-11. Παραδείγματα για Callback συναρτήσεις

```
<?php

// simple callback example
function my_callback_function() {
    echo 'hello world!';
}
call_user_func('my_callback_function');

// method callback examples
```

```
class MyClass {
    function myCallbackMethod() {
        echo 'Hello World!';
    }
}

// static class method call without instantiating an object
call user func(array('MyClass', 'myCallbackMethod'));

// object method call
$obj = new MyClass();
call_user_func(array(&$obj, 'myCallbackMethod'));
?>
```

Type Juggling

Η PHP δεν απαιτεί (ή υποστηρίζει) σαφή δήλωση τύπων κατά τη δήλωση μεταβλητών. Ένας τύπος μεταβλητής καθορίζεται από το περιεχόμενο με το οποίο αυτή η μεταβλητή θα χρησιμοποιηθεί. Δηλαδή, αν ορίσετε μια τιμή string σε μια μεταβλητή *$var*, η *$var* γίνεται string. Αν στη συνέχεια αναθέσετε μια integer τιμή στη *$var*, τότε γίνεται integer.

Ένα παράδειγμα της αυτόματης μετατροπής τύπου στην PHP είναι ο τελεστής πρόσθεσης '+'. Αν οποιοσδήποτε από τα τελούμενα είναι float, τότε όλα υπολογίζονται ως floats, και το αποτέλεσμα θα είναι float. Διαφορετικά, τα τελούμενα θα ερμηνεύονται ως integers, και το αποτέλεσμα θα είναι επίσης integer. Σημειώστε ότι αυτό ΔΕΝ αλλάζει τους τύπους των ίδιων των τελούμενων. Η μόνη αλλαγή είναι στο πώς υπολογίζονται τα τελούμενα.

```
<?php
$foo = "0";  // $foo is string (ASCII 48)
$foo += 2;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs";     // $foo is integer (15)
?>
```

Αν τα τελευταία δυο παραδείγματα σας φάνηκαν περίεργα, δείτε το Μετατροπή του String σε number.

Αν επιθυμείτε να αναγκάσετε μια μεταβλητή να υπολογιστεί σαν να ήταν ενός συγκεκριμένου τύπου, δείτε το τμήμα σχετικά με Type casting. Αν επιθυμείτε να αλλάξετε τον τύπο μιας μεταβλητής, δείτε το **settype()**.

Αν θέλετε να ελέγξετε οποιοδήποτε από αυτά τα παραδείγματα σ'αυτό το τμήμα, μπορείτε να χρησιμοποιήσετε τη συνάρτηση **var_dump()**.

Σημείωση: Η συμπεριφορά μιας αυτόματης μετατροπής σε array είναι προς το παρόν απροσδιόριστη.

```
<?php
$a = "1";      // $a is a string
$a[0] = "f";  // What about string offsets? What happens?
?>
```

Αφού η PHP (για ιστορικούς λόγους) υποστηρίζει indexing σε strings μέσω offsets χρησιμοποιώντας την ίδια σύνταξη όπως και στο indexing των arrays, το παράδειγμα παραπάνω οδηγεί σε ένα πρόβλημα: θα πρέπει το $a να γίνει array με το πρώτο στοιχείο του να είναι το "f", ή θα πρέπει το "f" να γίνει ο πρώτος χαρακτήρας του string $a?

Οι τρέχουσες εκδόσεις της PHP μεταγλωττίζουν τη δεύτερη ανάθεση ως string offset πιστοποίηση, συνεπώς η $a γίνεται "f", το αποτέλεσμα όμως αυτής της αυτόματης μετατροπής θα πρέπει να θεωρηθεί απροσδιόριστο. Η PHP 4 εισήγαγε τη νέα σύνταξη χρημοποιώντας curly bracket για να αποκτήσει πρόσβαση σε χαρακτήρες ενός string, συνεπώς χρησιμοποιήστε αυτή τη σύνταξη αντί αυτής που παρουσιάστηκε παραπάνω:

```
<?php
$a    = "abc"; // $a is a string
$a{1} = "f";   // $a is now "afc"
?>
```

Δείτε το τμήμα με τίτλο Πρόσβαση των String με χαρακτήρες για περισσότερες πληροφορίες.

Type Casting

Η μετατροπή τύπων (ή Type casting) στην PHP δουλεύει σχεδόν όπως και στη C: το όνομα του επιθυμητού τύπου γράφετε σε παρενθέσεις πριν από τη μεταβλητή στην οποία θα γίνει το cast.

```
<?php
$foo = 10;   // $foo is an integer
$bar = (boolean) $foo;   // $bar is a boolean
?>
```

Τα επιτρεπόμενα casts είναι τα:

- (int), (integer) - cast σε integer
- (bool), (boolean) - cast σε boolean
- (float), (double), (real) - cast σε float
- (string) - cast σε string
- (array) - cast σε array
- (object) - cast σε object

Σημειώστε ότι τα tabs και τα spaces επιτρέπονται μέσα στις παρενθέσεις, συνεπώς τα ακόλουθα είναι εξίσου λειτουργικά:

```
<?php
$foo = (int) $bar;
$foo = ( int ) $bar;
?>
```

Σημείωση: Αντί να κάνουμε cast μιας μεταβλητής σε string, μπορείτε επίσης να κλείσετε τη μεταβλητή σε διπλά εισαγωγικά.

```
<?php
$foo = 10;            // $foo is an integer
$str = "$foo";        // $str is a string
$fst = (string) $foo; // $fst is also a string

// This prints out that "they are the same"
if ($fst === $str) {
    echo "they are the same";
}
?>
```

Ίσως δεν είναι ακριβώς προφανές τι θα συμβεί όταν γίνεται casting μεταξύ συγκεκριμένων τύπων. Για περισσότερες πληροφορίες, δείτε αυτά τα τμήματα:

- Μετατρέποντας σε boolean
- Μετατρέποντας σε integer
- Μετατρέποντας σε float
- Μετατρέποντας σε string
- Μετατρέποντας σε array
- Μετατρέποντας σε object
- Μετατρέποντας σε resource
- Πίνακες σύγκρισης τύπων

Μεταβλητές

Βασικά

Οι μεταβλητές στην PHP αναπαρίστανται από το σύμβολο του δολαρίου ακολουθούμενο από το όνομα της μεταβλητής. Το όνομα της μεταβλητής είναι case-sensitive.

Τα ονόματα των μεταβλητών ακολουθούν τους ίδιους κανόνες όπως και οι labels στην PHP. Ένα έγκυρο όνομα μεταβλητής αρχίζει με ένα γράμμα ή underscore, ακολουθούμενο από οποιονδήποτε αριθμό από γράμματα, αριθμούς, ή underscores. Ως κανονική έκφραση θα γραφόταν ως εξής: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'

Σημείωση: Για τους σκοπούς μας εδώ, ένα γράμμα είναι από a-z, A-Z, και οι ASCII χαρακτήρες από το 127 ως το 255 (0x7f-0xff).

```
<?php
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";      // outputs "Bob, Joe"

$4site = 'not yet';     // invalid; starts with a number
$_4site = 'not yet';    // valid; starts with an underscore
$tδyte = 'mansikka';    // valid; 'δ' is (Extended) ASCII 228.
?>
```

Στην PHP 3, οι μεταβλητές ανατίθενται πάντα με τιμή. Δηλαδή, όταν αναθέτετε μια έκφραση σε μια μεταβλητή, ολόκληρη η τιμή της αρχική έκφρασης αντιγράφεται στη μεταβλητή προορισμού. Αυτό σημαίνει, για παράδειγμα, ότι αφού αναθέσετε την τιμή μιας μεταβλητής σε μια άλλη, αλλάζοντας μια από αυτές τις μεταβλητές δε θα επηρεαστεί η άλλη. Για περισσότερες πληροφορίες γι'αυτού του είδους την ανάθεση, δείτε το κεφάλαιο σχετικά με Εκφράσεις.

Η PHP 4 προσφέρει έναν άλλο τρόπο για να αναθέσετε τιμές σε μεταβλητές: ανάθεση με αναφορά. Αυτό σημαίνει ότι η νέα μεταβλητή απλά αναφέρεται (με άλλα λόγια, "γίνεται alias για" ή "δείχνει σε") στην αρχική μεταβλητή. Αλλαγές στη νέα μεταβλητή επηρεάζουν την αρχική, και αντιστρόφως. Αυτό σημαίνει επίσης ότι δε γίνεται αντιγραφή. Συνεπώς, η ανάθεση συμβαίνει πιο γρήγορα. Πάντως, οποιοσδήποτε τρόπος επιτάχυνσης θα γίνει εμφανής μόνο σε tight loops ή όταν γίνεται ανάθεση μεγάλων arrays ή objects.

Για να αναθέσουμε με αναφορά, απλά βάζουμε μπροστά ένα ampersand (&) στην αρχή της μεταβλητής στην οποία γίνεται η ανάθεση (η αρχική μεταβλητή). Για παράδειγμα, το ακόλουθο κομμάτι κώδικα δίνει το αποτέλεσμα 'My name is Bob' δύο φορές:

```
<?php
$foo = 'Bob';              // Assign the value 'Bob' to $foo
$bar = &$foo;              // Reference $foo via $bar.
$bar = "My name is $bar";  // Alter $bar...
echo $bar;
echo $foo;                 // $foo is altered too.
?>
```

Ένα σημαντικό πράγμα που πρέπει να σημειώσουμε είναι ότι μόνο οι μεταβλητές με όνομα μπορούν να ανατεθούν με αναφορά.

```
<?php
$foo = 25;
$bar = &$foo;      // This is a valid assignment.
$bar = &(24 * 7);  // Invalid; references an unnamed expression.

function test()
{
   return 25;
}

$bar = &test();    // Invalid.
?>
```

Προκαθορισμένες μεταβλητές

Η PHP παρέχει έναν μεγάλο αριθμό από προκαθορισμένες μεταβλητές σε οποιοδήποτε script τρέχει. Αρκετές από αυτές τις μεταβλητές πάντως, δεν μπορούν να τεκμηριωθούν εντελώς αφού εξαρτώνται από τον server στον οποίο τρέχουν, την έκδοση και το setup του server, καθώς και από άλλους παράγοντες. Μερικές από αυτές τις μεταβλητές δε θα είναι διαθέσιμες όταν η PHP τρέχει σε command line. Για μια λίστα αυτών των μεταβλητών, παρακαλώ δείτε το τμήμα Δεσμευμένες προκαθορισμένες μεταβλητές.

Προειδοποίηση
Στην PHP 4.2.0 και μετά, η προκαθορισμένη τιμή για την ντιρεκτίβα της PHP register_globals είναι *off*. Αυτή είναι μια σημαντική αλλαγή για την PHP. Έχοντας τις register_globals *off* επηρεάζεται το σύνολο των προκαθορισμένων μεταβλητών που είναι διαθέσιμες σε global εμβέλεια. Για παράδειγμα για να πάρετε το `DOCUMENT_ROOT` θα χρησιμοποιήσετε την `$_SERVER['DOCUMENT_ROOT']` αντί για την `$DOCUMENT_ROOT`, ή `$_GET['id']` από το URL `http://www.example.com/test.php?id=3` αντί για την `$id`, ή την `$_ENV['HOME']` αντί για την `$HOME`. Για πληροφορίες σχετικές μ'αυτή την αλλαγή, διαβάστε το configuration entry για register_globals, το κεφάλαιο για ασφάλεια Χρησιμοποιώντας Register Globals , καθώς επίσης και την PHP 4.1.0 και 4.2.0 Release Announcements. Είναι προτιμότερο να χρησιμοποιείτε τις διαθέσιμες PHP Reserved Predefined Μεταβλητές, όπως την superglobal arrays.

Από την έκδοση 4.1.0 και μετά, η PHP παρέχει ένα επιπρόσθετο σύνολο από προκαθορισμένους arrays που περιέχουν μεταβλητές από τον web server (αν είναι δυνατό), το environment (περιβάλλον), και αυτά που εισάγει ο χρήστης. Αυτοί οι νέοι arrays είναι μάλλον ιδιαίτεροι από την άποψη ότι είναι αυτόματα global--π.χ., αυτόματα διαθέσιμοι για κάθε εμβέλεια. Γι'αυτό το σκοπό, είναι συχνά γνωστοί και ως 'autoglobals' ή 'superglobals'. (Δεν υπάρχει μηχανισμός στην PHP για superglobals που μπορεί να ορίσει ο χρήστης.) Οι superglobals παρατίθενται παρακάτω. Πάντως, για μια λίστα των περιεχομένων τους και περεταίρω συζήτηση πάνω στις προκαθορισμένες μεταβλητές της PHP και στη φύση τους, παρακαλώ δείτε το τμήμα Δεσμευμένες προκαθορισμένες μεταβλητές. Επίσης, θα παρατηρήσετε πώς οι παλιότερες προκαθορισμένες μεταβλητές (`$HTTP_*_VARS`) υπάρχουν ακόμη. Από την PHP 5.0.0, τα μεγάλα προκαθορισμένα σταθερά array μπορούν να απενεργοποιηθούν με το register_long_arrays directive.

Μεταβλητές μεταβλητών: Οι superglobals δεν μπορούν να χρησιμοποιηθούν ως μεταβλητές μεταβλητών.

Αν μια συγκεκριμένη μεταβλητή στην variables_order δεν έχει οριστεί, οι κατάλληλοι προκαθορισμένοι arrays της PHP μένουν κενοί.

PHP Superglobals

$GLOBALS

Περιέχουν μια αναφορά σε κάθε μεταβλητή που είναι διαθέσιμη μέσα στην global εμβέλεια του script. Τα κλειδιά αυτού του array είναι τα ονόματα των global μεταβλητών. Η `$GLOBALS` υπάρχει από την PHP 3.

$_SERVER

Είναι οι μεταβλητές που ορίζονται από τον web server ή διαφορετικά είναι άμεσα συνδεδεμένες με το περιβάλλον εκτέλεσης του τρέχοντος script. Είναι ανάλογες με τον παλιό `$HTTP_SERVER_VARS` array (ο οποίος είναι ακόμη διαθέσιμος, αλλά δε συνιστάται).

$_GET

Είναι οι μεταβλητές που παρέχονται στο script μέσω του HTTP GET. Είναι ανάλογες με τον παλιό `$HTTP_GET_VARS` array (ο οποίος είναι ακόμη διαθέσιμος, αλλά δε συνιστάται).

$_POST

Είναι οι μεταβλητές που παρέχονται στο script μέσω του HTTP POST. Είναι ανάλογες με τον παλιό `$HTTP_POST_VARS` array (ο οποίος είναι ακόμη διαθέσιμος, αλλά δε συνιστάται).

$_COOKIE

Είναι οι μεταβλητές που παρέχονται στο script μέσω της HTTP cookies. Είναι ανάλογες με τον παλιό `$HTTP_COOKIE_VARS` array (ο οποίος είναι ακόμη διαθέσιμος, αλλά δε συνιστάται).

$_FILES

Είναι οι μεταβλητές που παρέχονται στο script μέσω του HTTP post file uploads. Είναι ανάλογες με τον `$HTTP_POST_FILES` array (ο οποίος είναι ακόμη διαθέσιμος, αλλά δε συνιστάται). Δείτε το POST method uploads για περισσότερες πληροφορίες.

$_ENV

Είναι οι μεταβλητές που παρέχονται στο script μέσω του environment. Είναι ανάλογες με τον παλιό `$HTTP_ENV_VARS` array (ο οποίος είναι ακόμη διαθέσιμος, αλλά δε συνιστάται).

$_REQUEST

Είναι οι μεταβλητές που παρέχονται στο script μέσω του μηχανισμού εισαγωγής δεδομένων από το χρήστηξ, και συνεπώς δεν είναι αξιόπιστες. Η

παρουσία και σειρά των μεταβλητών που περιέχονται στον array καθορίζεται σύμφωνα με την variables_order ντιρεκτίβα για configuration. Αυτός ο πίνακας δεν είναι ανάλογος με κάποιον άλλον σε προηγούμενες εκδόσεις την PHP προν την 4.1.0. Δείτε επίσης την **import_request_variables()**.

Προσοχή
Από την PHP 4.3.0, η πληροφορία για την FILE από την `$_FILES` δεν υπάρχει πια στην `$_REQUEST`.

Σημείωση: Όταν τρέχουμε σε command line , αυτό *δεν* θα συμπεριλάβει την `argv` και την `argc` εισόδους. Αυτές είναι παρούσες στον `$_SERVER` πίνακα.

$_SESSION

Είναι οι μεταβλητές που είναι προς το παρόν εγγεγραμμένες σε ένα session ενός script. Είναι ανάλογες με τον παλιό `$HTTP_SESSION_VARS` array (ο οποίος είναι ακόμη διαθέσιμος, αλλά δε συνιστάται). Δείτε το τμήμα Χρησιμοποιώντας συναρτήσεις για sessions για περισσότερες πληροφορίες.

Εμβέλεια μεταβλητών

Η εμβέλεια των μεταβλητών καθορίζεται από το περιεχόμενο μέσα στο οποίο ορίζεται. Για την πλειοψηφία των μεταβλητών της PHP υπάρχει μόνο ενός είδους εμβέλεια. Αυτή η εμβέλεια περιέχει τόσο τα αρχεία που περιέχονται όσο και αυτά που απαιτούνται. Για παράδειγμα:

```
<?php
$a = 1;
include "b.inc";
?>
```

Εδώ η μεταβλητή `$a` θα είναι διαθέσιμη μέσα στο εμπεριεχόμενο `b.inc` script. Πάντως, μέσα σε συναρτήσεις που ορίζονται από το χρήστη εισάγεται εμβέλεια τοπικής συνάρτησης (local function scope). Οποιαδήποτε μεταβλητή χρησιμοιποιείται μέσα σε μια συνάρτηση είναι εκ των προτέρων περιορισμένη σε εμβέλεια τοπικής συνάρτησης. Για παράδειγμα:

```
<?php
$a = 1; /* global scope */

function Test()
{
    echo $a; /* reference to local scope variable */
}

Test();
?>
```

Αυτό το script δε θα δώσει κάποιο αποτέλεσμα επειδή η echo δήλωση αναφέρεται σε μια τοπική έκδοση της `$a` μεταβλητής, και δεν της έχει ανατεθεί μια τιμή μέσα σ'αυτή την εμβέλεια. Ίσως παρατηρείσετε ότι είναι λίγο διαφορετική από τη C γλώσσα από την άποψη ότι οι global μεταβλητές στη C είναι άμεσα διαθέσιμες σε συναρτήσεις εκτός και αν επικαλύπτονται αυτόματα από κάποια τοπική αναφορά. Αυτό μπορεί να προκαλέσει μερικά προβλήματα αν κάποιοι, ακούσια, αλλάξουν μια global μεταβλητή. Στην PHP οι global μεταβλητές πρέπει να οριστούν ως global μέσα στη συνάρτηση αν πρόκειται να χρησιμοποιηθούν σ'αυτή τη συνάρτηση. Ένα παράδειγμα :

The global keyword

Πρώτα, ένα παράδειγμα από τη χρήση της `global`:

Παράδειγμα 7-1. Χρησιμοποιώντας global

```
<?php
$a = 1;
$b = 2;

function Sum()
{
    global $a, $b;

    $b = $a + $b;
}

Sum();
echo $b;
?>
```

Το παραπάνω script θα δώσει "3". Δηλώνοντας την `$a` και την `$b` ως global μέσα σε μια συνάρτηση, όλες οι αναφορές σε οποιαδήποτε από τις μεταβλητές θα αναφέρονται στην global έκδοση. Δεν υπάρχει περιορισμός στον αριθμό των global μεταβλητών που μπορεί να χειριστεί μια συνάρτηση.

Ένας δεύτερος τρόπος για να προσπελάσουμε μεταβλητές από global εμβέλεια είναι να χρησιμοποιήσουμε τον ειδικά ορισμένο από την PHP `$GLOBALS` array. Το προηγούμενο παράδειγμα μπορεί να ξαναγραφτεί ως:

Παράδειγμα 7-2. Χρησιμοποιώντας την `$GLOBALS` αντί για την global

```
<?php
$a = 1;
$b = 2;

function Sum()
{
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}
```

```
Sum();
echo $b;
?>
```

Ο πίνακας `$GLOBALS` είναι ένας associative array με το όνομα της global μεταβλητής να είναι το κλειδί και τα περιεχόμενα αυτής της μεταβλητής να είναι η τιμή του στοιχείου του array. Σημειώστε πώς η `$GLOBALS` υπάρχει σε κάθε εμβέλεια, κάτι το οποίο συμβαίνει επειδή η $GLOBALS είναι μια superglobal. Δείτε ένα παράδειγμα που δείχνει τη δύναμη των superglobals:

Παράδειγμα 7-3. Παράδειγμα που δείχνει τις superglobals και την εμβέλεια

```
<?php
function test_global()
{
    // Most predefined variables aren't "super" and require
    // 'global' to be available to the functions local scope.
    global $HTTP_POST_VARS;

    print $HTTP_POST_VARS['name'];

    // Superglobals are available in any scope and do
    // not require 'global'.  Superglobals are available
    // as of PHP 4.1.0
    print $_POST['name'];
}
?>
```

Χρησιμοποιώντας στατικές μεταβλητές

Ένα επιπλέον σημαντικό χαρακτηριστικό της εμβέλειας μεταβλητών είναι η *static* μεταβλητή. Μια στατική μεταβλητή υπάρχει μόνο σε εμβέλεια τοπικής συνάρτησης, αλλά δεν χάνει την τιμή της όταν η εκτέλεση του προγράμματος αφήνει αυτή την εμβέλεια. Θεωρείστε το ακόλουθο παράδειγμα:

Παράδειγμα 7-4. Παράδειγμα που δείχνει την ανάγκη για στατικές μεταβλητές

```
<?php
function Test ()
{
    $a = 0;
    echo $a;
    $a++;
}
?>
```

Αυτή η συνάρτηση είναι σχεδόν άχρηστη αφού κάθε φορά που καλείται θέτει την `$a` σε `0` και τυπώνει "0". Η `$a++` η οποία αυξάνει τη μεταβλητή δεν εξυπηρετεί κάποιο σκοπό αφού μόλις η συνάρτηση τελειώσει η μεταβλητή `$a` εξαφανίζεται. Για να

φτιάξουμε μια χρήσιμη συνάρτηση μέτρησης η οποία δεν θα χάνει τον τρέχοντα υπολογισμό, η μεταβλητή `$a` δηλώνεται ως στατική:

Παράδειγμα 7-5. Παράδειγμα στατικών μεταβλητών

```
<?php
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

Τώρα, κάθε φορά που θα καλείται η συνάρτηση Test() θα τυπώνει την τιμή της `$a` και θα την αυξάνει.

Οι στατικές μεταβλητές επίσης παρέχουν έναν τρόπο για να χειριστούμε αναδρομικές συναρτήσεις. Μια αναδρομική συνάρτηση είναι αυτή που καλεί τον εαυτό της. Πρέπει να δίνεται προσοχή όταν γράφουμε μια αναδρομική συνάρτηση επειδή είναι πιθανό να την κάνουμε να επαναλαμβάνεται ατέρμονα. Πρέπει να βεβαιωθείτε ότι έχετε έναν επαρκή τρόπο για να τερματίσετε την αναδρομή. Η ακόλουθη απλή συνάρτηση αναδρομικά μετράει ως το 10, χρησιμοποιώντας την στατική μεταβλητή `$count` για να ξέρει πότε θα σταματήσει:

Παράδειγμα 7-6. Στατικές μεταβλητές με αναδρομικές συναρτήσεις

```
<?php
function Test()
{
    static $count = 0;

    $count++;
    echo $count;
    if ($count < 10) {
        Test ();
    }
    $count--;
}
?>
```

Αναφορές με στατικές και global μεταβλητές

Η Zend Engine 1, που οδηγεί την `PHP4`, υλοποιεί τον στατικό και global modifier για τις μεταβλητές όσον αφορά τις αναφορές. Για παράδειγμα μια πραγματική global μεταβλητή που εισάγετε μέσα σε μια εμβέλεια συνάρτησης με τη δήλωση `global` στην πραγματικότητα δημιουργεί μια αναφορά στην global μεταβλητή. Αυτό μπορεί να οδηγήσει σε μη αναμενόμενη συμπεριφορά που φαίνεται στο ακόλουθο παράδειγμα:

```
<?php
```

```
function test_global_ref() {
    global $obj;
    $obj = &new stdclass;
}

function test_global_noref() {
    global $obj;
    $obj = new stdclass;
}

test_global_ref();
var_dump($obj);
test_global_noref();
var_dump($obj);
?>
```

Εκτελώντας το παράδειγμα θα έχουμε το ακόλουθο αποτέλεσμα:

```
NULL
object(stdClass)(0) {
}
```

Μια παρόμοια συμπεριφορά εφαρμόζεται στη `στατική` δήλωση. Οι αναφορές δεν αποθηκεύονται στατικά:

```
<?php
function &get_instance_ref() {
    static $obj;

    echo "Static object: ";
    var_dump($obj);
    if (!isset($obj)) {
        // Assign a reference to the static variable
        $obj = &new stdclass;
    }
    $obj->property++;
    return $obj;
}

function &get_instance_noref() {
    static $obj;

    echo "Static object: ";
    var_dump($obj);
    if (!isset($obj)) {
        // Assign the object to the static variable
        $obj = new stdclass;
    }
    $obj->property++;
    return $obj;
}

$obj1 = get_instance_ref();
$still_obj1 = get_instance_ref();
echo "\n";
$obj2 = get_instance_noref();
$still_obj2 = get_instance_noref();
```

```
?>
```

Εκτελώντας το παράδειγμα θα έχουμε το ακόλουθο αποτέλεσμα:

```
Static object: NULL
Static object: NULL

Static object: NULL
Static object: object(stdClass)(1) {
  ["property"]=>
  int(1)
}
```

Αυτό το παράδειμα δείχνει πως όταν αναθέτουμε μια αναφορά σε μια στατική μεταβλητή, δεν *μένει στη μνήμη* όταν καλείτε τη συνάρτηση `&get_instance_ref()` για δεύτερη φορά.

Μεταβλητές μεταβλητών

Μερικές φορές είναι βολικό να μπορούμε να έχουμε μεταβλητά ονόματα μεταβλητών. Αυτό σημαίνει πως, ένα όνομα μεταβλητής μπορεί να οριστεί και να χρησιμοποιηθεί δυναμικά. Μια κανονική μεταβλητή ορίζεται με μια δήλωση όπως:

```
<?php
$a = "hello";
?>
```

Μια μεταβλητή μεταβλητής παίρνει την τιμή μιας μεταβλητής και της συμπεριφέρεται ως όνομα μεταβλητής. Στο παραπάνω παράδειγμα, η *hello*, μπορεί να χρησιμοποιηθεί ως το όνομα της μεταβλητής χρησιμοποιώντας το σύμβολο του δολαρίου δυο φορές. π.χ.

```
<?php
$$a = "world";
?>
```

Σ'αυτό το σημείο δυο μεταβλητές έχουν οριστεί και αποθηκευτεί στο δέντρο συμβόλων (symbol tree) της PHP : η `$a` με περιεχόμενο "hello" και η `$hello` με περιεχόμενο "world". Συνεπώς, αυτή η δήλωση:

```
<?php
echo "$a ${$a}";
?>
```

παράγει ακριβώς το ίδιο αποτέλεσμα όπως η:

```
<?php
echo "$a $hello";
?>
```

π.χ. και οι δυο παράγουν: `hello world`.

Προκειμένου να χρησιμοποιήσετε μεταβλητές μεταβλητών με arrays, πρέπει να λύσετε το πρόβλημα της ασάφειας. Δηλαδή, αν γράφετε `$$a[1]` τότε ο parser χρειάζεται να ξέρει αν θέλετε να χρησιμοποιήσετε την `$a[1]` ως μεταβλητή, ή αν θέλετε την `$$a` ως μεταβλητή και συνεπώς το [1] index από αυτή τη μεταβλητή. Η σύνταξη για να λύσουμε αυτή την ασάφεια είναι: `${$a[1]}` για την πρώτη περίπτωση και `${$a}[1]` για τη δεύτερη.

Προειδοποίηση
Παρακαλώ σημειώστε ότι οι μεταβλητές μεταβλητών δεν μπορούν να χρησιμοποιηθούν με τους Superglobal arrays της PHP. Αυτό σημαίνει ότι δεν μπορείτε να κάνετε πράγματα όπως `${$_GET}`. Αν ψάχνετε έναν τρόπο για να χειριστείτε τη διαθεσιμότητα των superglobals και της παλιάς `HTTP_*_VARS`, ίσως πρέπει να χρησιμοποιήσετε την αναφορά μεταξύ αυτών.

Μεταβλητές έξω από την PHP

Φόρμες της HTML (GET και POST)

Όταν μια φόρμα εισάγεται σε ένα PHP script, η πληροφορία από τη φόρμα γίνεται αυτόματα διαθέσιμη στο script. Υπάρχουν πολλοί τρόποι για να προσπελάσετε την πληροφορία, για παράδειγμα:

Παράδειγμα 7-7. Μια απλή φόρμα σε HTML

```
<form action="foo.php" method="POST">
    Name:  <input type="text" name="username"><br>
    Email: <input type="text" name="email"><br>
    <input type="submit" name="submit" value="Submit me!">
</form>
```

Ανάλογα με το ιδιαίτερο setup και τις προσωπικές προτιμήσεις, υπάρχουν πολλοί τρόποι για να προσπελάσετε τα δεδομένα από τις HTML φόρμες σας. Μερικά παραδείγματα είναι:

Παράδειγμα 7-8. Προσπελαύνοντας δεδομένα από μια απλή POST HTML φόρμα

```
<?php
// Available since PHP 4.1.0

   print $_POST['username'];
   print $_REQUEST['username'];
```

```
   import_request_variables('p', 'p_');
   print $p_username;

// Available since PHP 3.  As of PHP 5.0.0, these long predefined
// variables can be disabled with the register_long_arrays
directive.

   print $HTTP_POST_VARS['username'];

// Available if the PHP directive register_globals = on.  As of
// PHP 4.2.0 the default value of register_globals = off.
// Using/relying on this method is not preferred.

   print $username;
?>
```

Η χρήση μιας GET φόρμας είναι παρόμοια εκτός από το ότι πρέπει να χρησιμοποιήσετε την κατάλληλη προκαθορισμένη μεταβλητή GET. Η GET επίσης χρησιμοποιείται στο QUERY_STRING (η πληροφορία μετά το '?' σε ένα URL). Συνεπώς, για παράδειγμα η `http://www.example.com/test.php?id=3` περιέχει GET δεδομένα τα οποία είναι προσπελάσιμα με την `$_GET['id']`. Δείτε επίσης την $_REQUEST και την **import_request_variables()**.

Σημείωση: Οι superglobal arrays, όπως ο `$_POST` και ο `$_GET`, έγιναν διαθέσιμοι στην PHP 4.1.0

Όπως δείξαμε, πριν την PHP 4.2.0 η προκαθορισμένη τιμή για τη register_globals ήταν *on*. Και στην PHP 3 ήταν πάντα on. Η κοινότητα της PHP ενθαρρύνει όλους να μην βασίζονται σ'αυτή την ντιρεκτίβα καθώς προτιμάται να υποθέτουμε ότι είναι *off* και να γράφουμε κώδικα σύμφωνα με αυτή την υπόθεση.

Σημείωση: Η magic_quotes_gpc configuration ντιρεκτίβα επηρεάζει τις τιμές της Get, της Post και της Cookie. Αν γίνει on, η τιμή (It's "PHP!") θα γίνει αυτόματα (It\'s \"PHP!\"). Χρειάζεται να γίνει escape για εισαγωγή εισαγωγικών. Δείτε επίσης τις **addslashes()**, **stripslashes()** και magic_quotes_sybase.

Η PHP επίσης καταλαβαίνει arrays σχετικά με το περιεχόμενο από τις μεταβλητές φορμών (δείτε το σχετικό faq). Ίσως, για παράδειγμα ομαδοποιήσετε σχετικές μεταβλητές μαζί, ή χρησιμοποιήσετε αυτό το χαρακτηριστικό για να πάρετε τιμές από ένα multiple select input. Για παράδειγμα, ας στείλουμε μια φόρμα στον εαυτό της και μετά την υποβολή εμφανιστούν τα δεδομένα:

Παράδειγμα 7-9. Περισσότερες σύνθετες μεταβλητές φορμών

```
<?php
if (isset($_POST['action']) && $_POST['action'] == 'submitted') {
    print '<pre>';
    print_r($_POST);
    print '<a href="'. $_SERVER['PHP_SELF'] .'">Please try
again</a>';

    print '</pre>';
```

```
} else {
?>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
    Name:  <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Beer: <br>
    <select multiple name="beer[]">
        <option value="warthog">Warthog</option>
        <option value="guinness">Guinness</option>
        <option value="stuttgarter">Stuttgarter
Schwabenbrδu</option>
    </select><br>
    <input type="hidden" name="action" value="submitted">
    <input type="submit" name="submit" value="submit me!">
</form>
<?php
}
?>
```

Στην PHP 3, η χρήση της μεταβλητής array της φόρμας είναι περιορισμένη σε μονοδιάστατους arrays. Στην PHP 4, δεν εφαρμόζονται τέτοιοι περιορισμοί.

IMAGE SUBMIT ονόματα μεταβλητής

Όταν εισάγετε μια φόρμα, είναι δυνατό να χρησιμοποιήσετε μια εικόνα αντί για το καθιερωμένο κουμπί υποβολής με ένα tag σαν και αυτό:

```
<input type="image" src="image.gif" name="sub">
```

Όταν ο χρήστης κάνει κλίκ κάπου στην εικόνα, η ακόλουθη φόρμα θα μεταφερθεί στον server με δυο επιπρόσθετες μεταβλητές, την sub_x και την sub_y. Αυτές περιέχουν τις συντεταγμένες του σημείου που έκανε κλικ ο χρήστης μέσα στην εικόνα. Ο έμπειρος χρήστης ίσως παρατηρήσει ότι τα πραγματικά ονόματα μεταβλητών που στάλθηκαν από τον browser περιέχουν μια period παρά ένα underscore, αλλά η PHP μετατρέπει αυτή την period σε ένα underscore αυτόματα.

HTTP Cookies

Η PHP με διαφάνεια υποστηρίζει τα HTTP cookies όπως έχουν οριστεί στο Netscape's Spec. Τα cookies είναι ένας μηχανισμός αποθήκευσης δεδομένων σε έναν απομακρυσμένο browser για τον εντοπισμό και αναγνώριση χρηστών που ξαναεπισκέπτονται στο site. Μπορείτε να ορίσετε cookies χρησιμοποιώντας τη συνάρτηση **setcookie()** . Τα cookies είναι μέρος του HTTP header, συνεπώς η συνάρτηση SetCookie πρέπει να καλείται πριν το αποτέλεσμα σταλεί στον browser. Αυτό έχει τους ίδιους περιορισμούς όπως και στη συνάρτηση **header()**. Τα δεδομένα των cookies είναι εν συνεχεία διαθέσιμα στους κατάλληλους cookie data arrays, όπως ο `$_COOKIE`, ο `$HTTP_COOKIE_VARS` καθως επίσης και ο `$_REQUEST`. Δείτε την **setcookie()** στη σελίδα του manual για περισσότερες πληροφορίες και παραδείγματα.

Αν επιθυμείτε να αναθέσετε πολλαπλές τιμές σε μια μόνο μεταβλητή cookie, μπορείτε να κάνετε την ανάθεση αυτή όπως και σε έναν array. Για παράδειγμα:

```
<?php
  setcookie("MyCookie[foo]", "Testing 1", time()+3600);
  setcookie("MyCookie[bar]", "Testing 2", time()+3600);
?>
```

Αυτό θα δημιουργήσει δυο διαφορετικά cookies παρόλου που το MyCookie θα είναι τώρα ένα απλό array στο script σας. Αν θέλετε να ορίσετε ακόμη ένα cookie με πολλαπλές τιμές, χρησιμοποιείστε την **serialize()** ή την **explode()** πρώτα στην τιμή.

Σημειώστε ότι ένα cookie θα αντικαταστήσει ένα προηγούμενο cookie με το ίδιο όνομα στον browser σας εκτός και αν το path ή το domain είναι διαφορετικό. Συνεπώς, για μια εφαρμογή ενός shopping cart ίσως χρειαστεί να κρατήσετε έναν μετρητή και να τον περάσετε. π.χ.

Παράδειγμα 7-10. Ένα παράδειγμα με την setcookie()

```
<?php
if (isset($_COOKIE['count'])) {
    $count = $_COOKIE['count'] + 1;
} else {
    $count = 1;
}
setcookie("count", $count, time()+3600);
setcookie("Cart[$count]", $item, time()+3600);
?>
```

Τελείες σε εσωτερικά ονόματα μεταβλητών

Τυπικά, η PHP δεν αλλάζει τα ονόματα των μεταβλητών όταν αυτά περνιούνται σε ένα script. Πάντως, πρέπει να σημειωθεί ότι η τελεία dot (period, full stop) δεν είναι ένας έγκυρος χαρακτήρας στην PHP για ονόματα μεταβλητών. Γι'αυτό το λόγο, δείτε αυτό:

```
<?php
$varname.ext;  /* invalid variable name */
?>
```

Τώρα, όταν ο parser βλέπει μια μεταβλητή με το όνομα `$varname`, ακολουθούμενη από τον τελεστή συννένωσης string, ακολουθούμενο από ένα barestring (π.χ. ένα string χωρίς εισαγωγικά το οποίο δεν ταιριάζει με κανένα γνωστό κλειδί ή κάποια δεσμευμένη λέξη) 'ext'. Προφανώς, αυτό δεν έχει το αναμενόμενο αποτέλεσμα.

Γι'αυτό το λόγο, είναι σημαντικό να σημειώσουμε ότι η PHP θα αντικαταστήσει αυτόματα όλες τις τελείες σε εσωτερικά ονόματα μεταβλητών, με underscores.

Ορίζοντας τύπους μεταβλητών

Επειδή η PHP καθορίζει τους τύπους των μεταβλητών και τους μετατρέπει (γενικά) όπως χρειάζεται, δεν είναι πάντα προφανές τι τύπου είναι μια δεδομένη μεταβλητή οποιαδήποτε στιγμή. Η PHP περιέχει διάφορες συναρτήσεις οι οποίες βρίσκουν τι τύπο έχει κάθε μεταβλητή, όπως οι: **gettype()**, **is_array()**, **is_float()**, **is_int()**, **is_object()**, και η **is_string()**. Δείτε επίσης το κεφάλαιο σχετικά με τους Τύπους.

Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ)

Τι είναι ένα ΣΔΒΔ?

 Συλλογή από συσχετιζόμενα δεδομένα

 Σύνολο προγραμμάτων για προσπέλαση των δεδομένων

ΣΔΒΔ περιλαμβάνει περιλαμβάνει

 Δομές για την αποθήκευση της πληροφορίας

 Παροχή μηχανισμών για το χειρισμό της πληροφορίας

 Στόχος του ΣΔΒΔ είναι να παρέχει ένα περιβάλλον το οποίο είναι *βολικό* και *αποδοτικό* στη χρήση.

Εφαρμογές βάσεων δεδομένων:

 Banking: all transactions

 Airlines: reservations, schedules

 Universities: registration, grades

 Sales: customers, products, purchases

 Manufacturing: production, inventory, orders, supply chain

 Human resources: employee records, salaries, tax deductions

Σκοπός των Συστημάτων ΒΔ

 Στα πρώτα χρόνια, οι εφαρμογές των βάσεων δεδομένων χτίζονταν πάνω από συστήματα αρχείων

 Μειονεκτήματα της χρήσης συστημάτων αρχείων για να αποθηκεύσουμε δεδομένα:

 Πλεονασμός (redundancy) δεδομένων και ασυνέπεια (inconsistency)

 Πολλαπλές μορφές αρχείων (file formats), εμφάνιση της πληροφορίας σε διαφορετικά αρχεία

 Δυσκολία στην προσπέλαση δεδομένων

 Απαίτηση για ανάπτυξη ενός νέου προγράμματος για να εκτελέσουμε κάθε νέα εργασία

 Απομόνωση δεδομένων

 δεδομένα κατανεμημένα σε πολλαπλά αρχεία ή/και σε διαφορετικές μορφές

 Προβλήματα ακεραιότητας

 Περιορισμοί ακεραιότητας (π.χ. account balance > 0) γίνονται μέρος του κώδικα

 Δύσκολο να προσθέσουμε νέους περιορισμούς ή να αλλάξουμε υπάρχοντες περιορισμούς

Μειονεκτήματα της χρήσης των συστημάτων αρχείων

 Ατομικότητα ενημερώσεων

 Αποτυχίες μπορεί να αφήσουν τη βάση δεδομένων σε ασυνεπή κατάσταση όταν εκτελείται μέρος των ενημερώσεων

 π.χ. Μεταφορά χρημάτων από έναν λογαριασμό σε έναν άλλο θα πρέπει να εκτελείται πλήρως ή να μην εκτελείται καθόλου

 Ταυτόχρονη προσπέλαση από πολλαπλούς χρήστες

 Μη ελεγχόμενη ταυτόχρονη προσπέλαση μπορεί να οδηγήσει σε ασυνέπειες

 π.χ. Δύο χρήστες διαβάζουν το υπόλοιπο ενός λογαριασμού και το ενημερώνουν την ίδια στιγμή.

 Προβλήματα ασφάλειας

 Τα συστήματα βάσεων δεδομένων προσφέρουν λύσεις σε όλα τα παραπάνω προβλήματα

Επίπεδα αφαίρεσης

 Φυσικό επίπεδο: περιγράφει πως αποθηκεύεται τα δεδομένα (περιγραφή δομές δεδομένων)

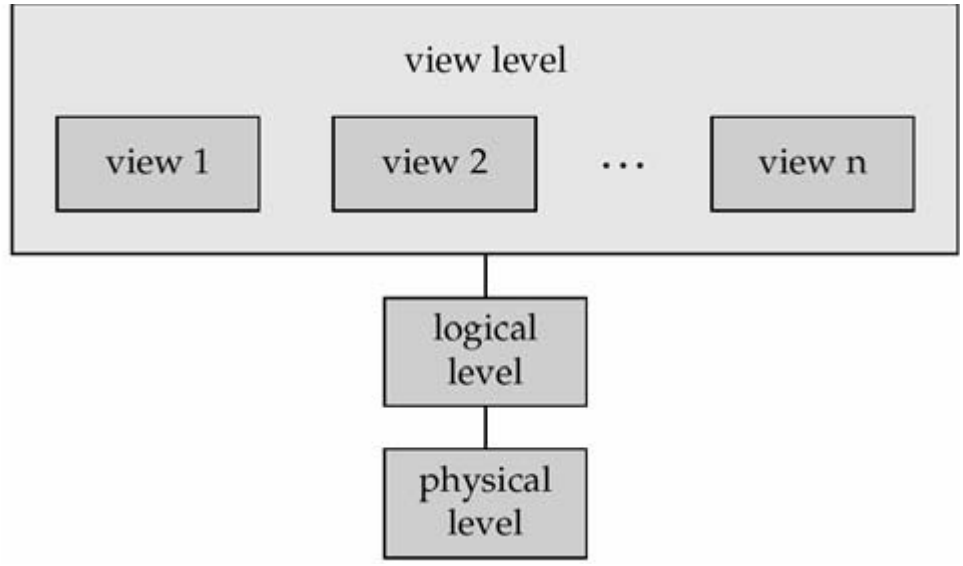
 Λογικό επίπεδο: περιγράφει δεδομένα που αποθηκεύονται σε μία βάση δεδομένων και τις σχέσεις ανάμεσα στα δεδομένα.

```
type customer = record
        name : string;
        street : string;
        city : integer;
end;
```

 Επίπεδο όψης (view level): περιγράφει μέρος της βάσης δεδομένων. Οι όψεις μπορούν να κρύβουν πληροφορία για λόγους ασφάλειας (π.χ. salary).

Όψη των δεδομένων

Σχέση τριών επιπέδων αφαιρετικότητας



Στιγμιότυπα και Σχήματα

 Σχήμα ≡ τύπος μεταβλητής, Στιγμιότυπο ≡ τιμή μεταβλητής
 Σχήμα – η λογική δομή της ΒΔ
 π.χ η βάση δεδομένων αποτελείται από πληροφορία για ένα σύνολο πελατών και λογαριασμών και τις σχέσεις μεταξύ τους
 Φυσικό σχήμα: σχεδιασμός βάσεις δεδομένων στο φυσικό επίπεδο
 Λογικό σχήμα : σχεδιασμός βάση δεδομένων σε λογικό επίπεδο
 Στιγμιότυπο – το πραγματικό περιεχόμενο της βάσης δεδομένων σε μία συγκεκριμένη στιγμή στο χρόνο
 Ανάλογο με την τιμή μίας μεταβλητής
 Φυσική ανεξαρτησία των δεδομένων – η ικανότητα να τροποποιούμε το φυσικό σχήμα χωρίς να αλλάζουμε το λογικό σχήμα
 Οι εφαρμογές εξαρτώνται από το λογικό σχήμα
 Γενικά, οι διεπαφές ανάμεσα στα διαφορετικά επίπεδα και τα συστατικά μέρη θα πρέπει να καθορίζονται ώστε αλλαγές σε κάποια τμήματα να μην επηρεάζουν σημαντικά τα υπόλοιπα.

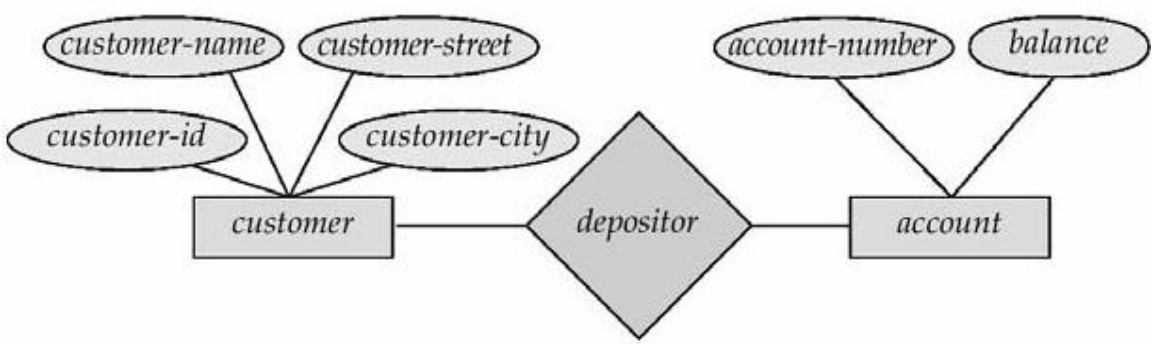
Μοντέλα Δεδομένων - Data Models

 Μια συλλογή εργαλείων για περιγραφή
 δεδομένων
 συσχετίσεις μεταξύ των δεδομένων
 σημασιολογία δεδομένων (data semantics)
 περιορισμούς δεδομένων
 Μοντέλο Οντοτήτων- Συσχετίσεων
 Σχεσιακό μοντέλο
 Άλλα μοντέλα:
 Αντικειμενοστραφές μοντέλο
 Μοντέλο δεδομένων σχεσιακών αντικειμένων

□□ **Πριν το σχεσιακό:** Ιεραρχικό μοντέλο

Μοντέλο Οντοτήτων - Συσχετίσεων

Παράδειγμα του σχήματος στο μοντέλο οντοτήτων-συσχετίσεων



E-R μοντέλο πραγματικού κόσμου

□ Οντότητες (αντικείμενα)

□ π.χ. customers, accounts, bank branch

□ Συσχετίσεις ανάμεσα σε οντότητες

□ Π.χ.. Account A-101 διατηρείται από τον πελάτη Johnson

□ Το σύνολο συσχέτισης *καταθέτης (depositor)* συσχετίζει πελάτες με λογαριασμούς

 Ευρέως χρησιμοποιούμενα για το σχεδιασμό βάσεων δεδομένων

 Ο σχεδιασμός βάσεων δεδομένων στο μοντέλο E-R συνήθως μετασχηματίζεται σε σχεδιασμό σχεσιακού μοντέλου το οποίο χρησιμοποιείται για αποθήκευση και επεξεργασία

SQL

 SQL: ευρέως χρησιμοποιούμενη μη-διαδικαστική γλώσσα

Παραδείγματα

 Find the name of the customer with customer-id 192-83-7465

select *customer.customer-name*
from *customer*
where *customer.customer-id* = '192-83-7465'

 Find the balances of all accounts held by the customer with customer-id 192-83-7465

select *account.balance*
from *depositor, account*
where *depositor.account-number = account.account-number*
and *depositor.customer-id* = '192-83-7465'

 Τα προγράμματα εφαρμογών γενικά προσπελαύνουν τις βάσεις δεδομένων μέσω

 Επεκτάσεις της γλώσσας που επιτρέπουν ενσωματωμένη SQL

 Διεπαφές προγραμμάτων εφαρμογών (π.χ. ODBC/JDBC) που επιτρέπουν να

στέλνονται ερωτήσεις SQL στη ΒΔ.

Χρήστες ΒΔ

 Οι χρήστες διαφοροποιούνται από τον τρόπο που

αναμένεται να αλληλεπιδράσουν με το σύστημα

 Προγραμματιστές Προγραμματιστές εφαρμογών εφαρμογών – αλληλεπιδρούν με το σύστημα
μέσω DML calls

 Προχωρημένοι Προχωρημένοι users users – διατυπώνουν αιτήσεις σε κάποια γλώσσα
επερωτήσεων

 Εξειδικευμένοι Εξειδικευμένοι – γράφουν εξειδικευμένες εφαρμογές ΒΔ που δεν ταιριάζουν σε ένα παραδοσιακό πλαίσιο επεξεργασίας δεδομένων

 Naïve users – επικαλούνται μία από τις εφαρμογές προγραμμάτων
που έχουν γραφτεί προηγουμένως

 π.χ. Προσπέλαση ΒΔ στο web, bank tellers, clerical staff

Διαχειριστής ΒΔ

 Συντονίζει όλες τις δραστηριότητες του συστήματος
βάσης δεδομένων

 Έχει καλή αντίληψη των πληροφοριακών πόρων του
οργανισμού και των αναγκών.

 Καθήκοντα:

 Ορισμός σχήματος (Schema definition)

 Ορισμός δομής αποθήκευσης και μεθόδων προσπέλασης

 Τροποποίηση σχήματος και φυσικής οργάνωσης

 Παροχή πιστοποίησης προσβασης των χρηστών στη ΒΔ

 Ορισμός περιορισμών ακεραιότητας

 Ρουτίνες συντήρησης

 Backup, διασφάλιση ύπαρξης ελεύθερου χώρου, διασφάλιση
αποδοτικότητας συστήματος

Διαχείριση Συναλλαγών

 Συναλλαγή (*transaction):* σύνολο από λειτουργίες που εκτελούν μία απλή λογική λειτουργία(logical function) σε μία εφαρμογή ΒΔ

 Το συστατικό στοιχείο της διαχείρισης συναλλαγών (Transaction- management component) επιβεβαιώνει ότι η ΒΔ παραμένει σε συνεπή (σωστή) κατάσταση παρά τις αποτυχίες του συστήματος (e.g., προβλήματα ισχύς, λειτουργικού συστήματος) και αποτυχίες συναλλαγών.

 Ο διαχειριστής ταυτόχρονου ελέγχου ελέγχει την αλληλεπίδραση ανάμεσα σε ταυτόχρονες συναλλαγές για να επιβεβαιώσει τη συνέπεια της βάσης δεδομένων

Διαχείριση αποθήκευσης

 Δ**ιαχειριστής αποθήκευσης** είναι μία λειτουργική μονάδα προγράμματος που παρέχει τη διεπαφή ανάμεσα στα χαμηλού επιπέδου δεδομένα που αποθηκεύονται σε μία ΒΔ , στα προγράμματα εφαρμογών και στα ερωτήματα που υποβάλλονται στο σύστημα

 Δ**ιαχειριστής αποθήκευσης** χειρίζεται

 Αρχεία δεδομένων, λεξικό δεδομένων (μετά-δεδομένα δομής ΒΔ), ευρετήρια

 Ο **διαχειριστής αποθήκευσης** είναι υπεύθυνος για τις ακόλουθες εργασίες:

 Αλληλεπίδραση με το διαχειριστή αρχείων

 Αποδοτική αποθήκευση, ανάκτηση και ενημέρωση δεδομένων

ΠΕΡΙΣΣΟΤΕΡΑ ΓΙΑ ΤΗ ΓΛΩΣΣΑ SQL

SQL (Structured Query Language) είναι η τυποποιημένη "standard" γλώσσα στις σχεσιακές Βάσεις. Η πρώτη χρήση ήταν στο πρότυπο σύστημα της IBM, που ονομάστηκε SYSTEM-R, το οποίο ανεπτύχθη στα ερευνητικά εργαστήρια της εταιρείας (San Jose, California) στα μέσα της δεκαετίας το 1970. Η SQL έχει υποστεί πολλές τροποποιήσεις.

- **Υπάρχουν 4 βασικές εντολές:**

– select

– insert

– update

– delete

- **Το αποτέλεσμα μιας εντολής / πράξης σε Σχέσεις είναι (πάντα) μια νέα Σχέση**

ΠΑΡΑΔΕΙΓΜΑ:

Θεωρήστε την παρακάτω ερωταπόκριση: "Βρείτε τα ονόματα των Υπαλλήλων που έχουν μισθό μεγαλύτερο των 600,000"

Στην SQL:

```
select e.Name
from EMPLOYEE e
where (e.Salary > 600000)
```

– e είναι μια μεταβλητή πλειάδος που ορίζεται να παίρνει τιμές από την Σχέση EMPLOYEE (στην from πρόταση)

– e.Name, ως περιορισμένη μεταβλητή πλειάδος, προσδιορίζει την τιμή του e στο γνώρισμα Name, και αποτελεί το target list (προσδιορίζει στην select πρόταση τις προβολές των στηλών)

– (e.Salary > 30000) είναι η ικανοποίηση συνθήκης (qualification) (προσδιορίζει στην where πρόταση όλες τις επιλογές και συνενώσεις)

Στην Insert αντίστοιχα ένα παράδειγμα έχει ως εξής:

insert into DEPARTMENT**values** (6, “inventory”, 9879, “30.5.45”)

Στην Update έχουμε:

update EMPLOYEE
set Salary = Salary * 1.14
where DNumber **in (select** DNumber
from DEPARTMENT
where DName = “admin”)

Στην Delete έχουμε

delete from EMPLOYEE
where DNumber **in (select** DNumber
from DEPARTMENT
where DName = “admin”)