

# ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΡΗΡ

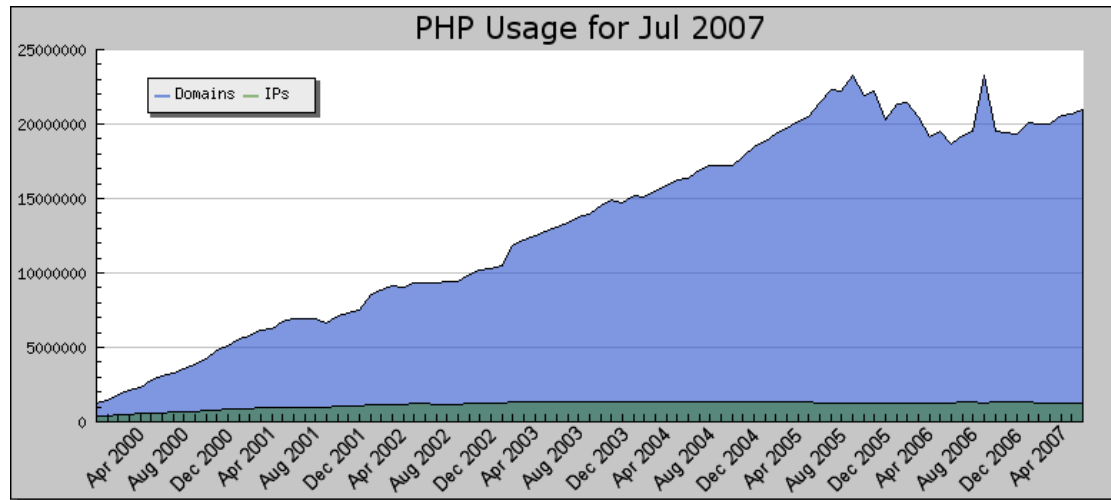
2010

Τεχνολογίες και Προγραμματισμός εφαρμογών στον Ιστό  
Οικονομικό Πανεπιστήμιο Αθηνών

# Εισαγωγή

2

- Η **PHP** ([www.php.net](http://www.php.net)) είναι μια **δημοφιλής, ανοικτού κώδικα, γλώσσα scripting** η οποία εκτελείται στην πλευρά του εξυπηρετητή (server-side)
- ▣ Σύμφωνα με έρευνα της Netcraft η PHP ίσως είναι η πιο δημοφιλής server-side τεχνολογία στο Web
- Όμως η παραπάνω έρευνα έγινε λαμβάνοντας υπόψη τις καταλήξεις των αρχείων, κριτήριο όχι πάντα αντιπροσωπευτικό



# Εισαγωγή

2

- Συμβατή με ένα ευρύ φάσμα λειτουργικών συστημάτων και εξυπηρετητών
- Σχετικά **εύκολο να μαθευτεί**, αλλά περιλαμβάνει **πολυάριθμες μεθόδους και πακέτα**
- Πιθανόν να μην είναι κατάλληλη για τον προγραμματισμό ενός **εξαιρετικά μεγάλου συστήματος** αλλά όπως και άλλα server-side scripting περιβάλλοντα (ASP, ColdFusion, κλπ.), συχνά χρησιμοποιείται και για αυτό το σκοπό

# Γιατί PHP?

4

- Εύκολη
  - ▣ Γνωστή από άλλες γλώσσες και όχι αυστηρή σύνταξη
- Ευέλικτη
  - ▣ Δεν επιβάλλει συγκεκριμένο στυλ προγραμματισμού
- Ισχυρή
  - ▣ Πολλές συναρτήσεις
  - ▣ Διαθέσιμος ελεύθερος κώδικας
  - ▣ Επεκτάσιμη
  - ▣ Συμβατή με πολλές τεχνολογίες (DBS, XML, κτλ)
- Δωρεάν
  - ▣ Συμβατή με όλες τις πλατφόρμες (καλύτερα όμως σε Linux)
- Ευρέως χρησιμοποιούμενη
  - ▣ Αρκετή βοήθεια από άλλους.
  - ▣ Έχει ζήτηση στην αγορά εργασίας

# Γιατί όχι PHP?

5

- Ανοικτού κώδικα
  - ▣ Ποιον επικαλούμαστε όταν όλα πάνε στραβά; Ποιος έχει τη νομική ευθύνη;
- Είναι τόσο απλή;
  - ▣ Αρκετοί προγραμματιστές δεν ακολουθούν τις καθιερωμένες πρακτικές προγραμματισμού γράφοντας "spaghetti code"
- Απουσία ποιότητας και προτυποποίησης
  - ▣ Πολλαπλά frameworks, εκδόσεις, ιδιορρυθμίες πλατφόρμας
- Όχι αρκετά καλή για μια επιχείρηση;
  - ▣ Συνήθως προτιμούνται πλατφόρμες Web που θεωρούνται ότι ταιριάζουν καλύτερα σε επιχειρήσεις όπως .NET, J2EE
  - ▣ Έλλειψη διασυνδέσεων(bridges) με μεγάλες πλατφόρμες όπως η ERP
  - ▣ Θα πρέπει να συνυπολογιστεί η ιδέα του "share nothing" και τα ευρήματα από συνέδρια που ασχολούνται με την επεκτασιμότητα της προτού κάποιος την επιλέξει
- Δεν επιβάλλει κάποιο προγραμματιστικό μοντέλο ή περιβάλλον
  - ▣ Έχουν προταθεί/υλοποιηθεί πολλά αλλά η κεντρική αρχή για components και frameworks υστερούν από τα αντίστοιχα της .NET και J2EE – (σταδιακά αλλάζει)

# Παραδείγματα εφαρμογών

6

- Απλή επεξεργασία φόρμας
  - ▣ Αποθήκευση σε βάση, φόρμα αποστολής email
- Σελίδες προσαρμοζόμενες στο χρήστη/browser
  - ▣ Αλλαγή όψης ή τεχνολογίας ανάλογα με χρήστη/browser
- Σελίδες συνεργασίας (collaboration)
  - ▣ Message boards, blogs κτλ
- Συστήματα Διαχείρισης Περιεχομένου (CMS)
  - ▣ Συντήρηση και ανανέωση του περιεχομένου ενός site χρησιμοποιώντας μόνο το browser
- E-commerce/Shopping carts
- Portals
- Και οτιδήποτε άλλο μπορείτε να φανταστείτε – παιχνίδια κτλ – εξάλλου είναι μια γλώσσα προγραμματισμού!

# Ιστορία της PHP

7

- Δημιουργήθηκε ως εργαλείο δημιουργίας προσωπικών σελίδων
  - ▣ Ιούνιος **1995** έκδοση 1.0
- Μέσα του **1996** (PHP/FI) – αρχίζει να μετατρέπεται σε script ενσωματωμένο στην HTML
- Ιούνιος **1998** – PHP **3.0**
  - ▣ Αυτή η έκδοση ξεκίνησε την πραγματική προσπάθεια υιοθέτησης της PHP
- Μάιος **2000** - PHP **4.0**
- Ιούλιος **2004** - PHP **5**
- PHP **6**: Η επόμενη έκδοση, ακόμη υπό ανάπτυξη

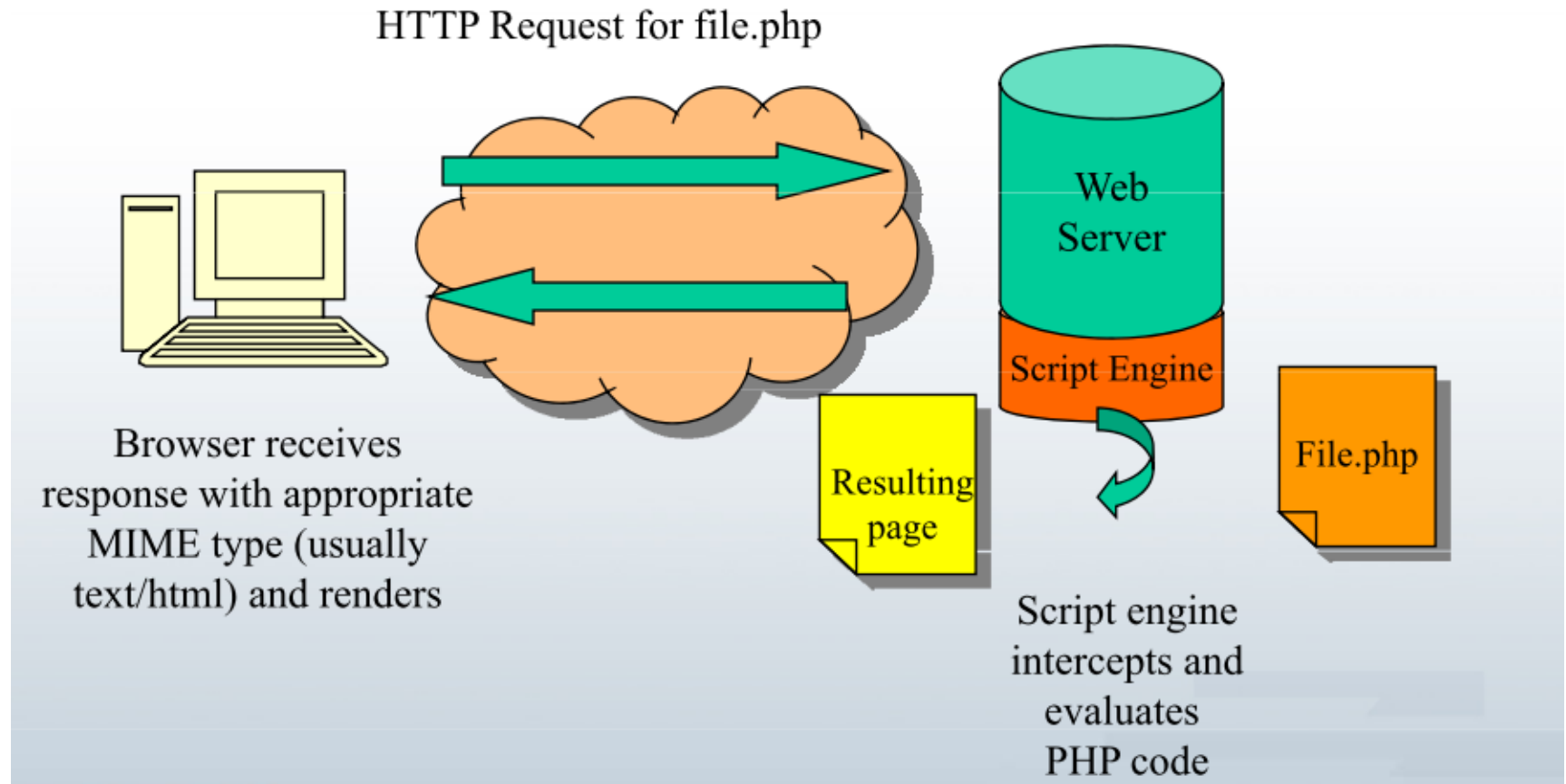
# Πώς λειτουργεί

8

- Στην απλούστερη περίπτωση, τα αρχεία PHP περιέχουν μίξη κώδικα PHP και (X)HTML/CSS/JavaScript
- Τα αρχεία PHP συλλέγονται και αξιολογούνται από μια scripting engine (συνήθως υλοποιημένη σαν ένα server module/ ISAPI), όταν ένα request γίνεται για αυτά, παράγοντας την ζητούμενη σελίδα
  - Η κατάληξη .php χρησιμοποιείται για να ενεργοποιήσει την script engine, αν και αυτό μπορεί να τροποποιηθεί με κατάλληλες ρυθμίσεις

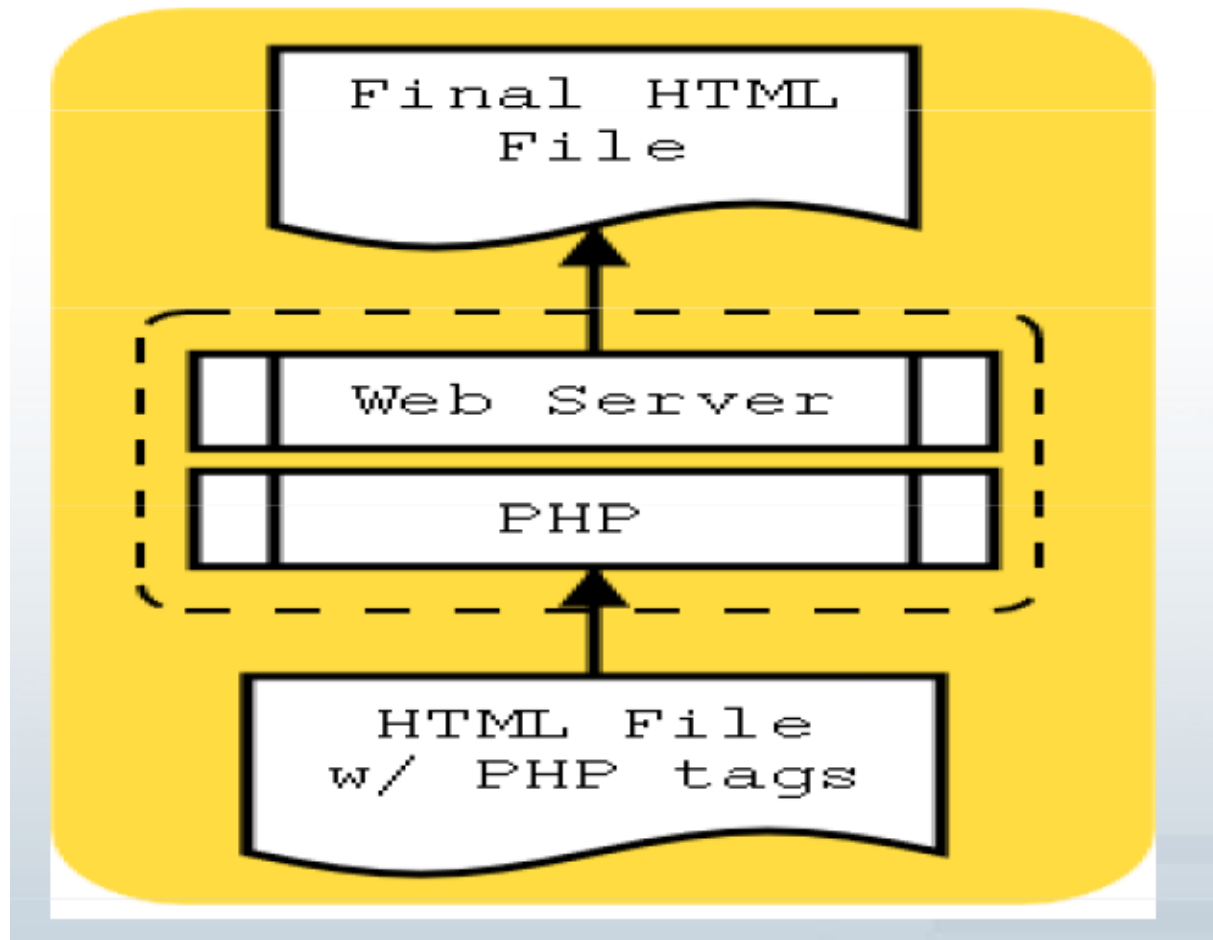


# Πώς λειτουργεί



# Πώς λειτουργεί (Abstract μορφή)

10



# Εγκατάσταση

11

- Διαθέσιμη σε εμπορική έκδοση ή open source
  - [www.zend.com](http://www.zend.com) (commercial)
  - [www.php.net](http://www.php.net) (open source)
- Επιλογή γλώσσας ανάλογα με λειτουργικό, μέθοδο υλοποίησης (CGI ή Module/ISAPI), έκδοση και τυχόν πρόσθετα που θα εγκατασταθούν στον εξυπηρετητή.
- Υπάρχει αναλυτικός οδηγός εγκατάστασης στο <http://php.net/manual/en/install.php> για κάθε περίπτωση

# Εργαλεία

12

- Για να προγραμματίσουμε χρειαζόμαστε ένα Web browser με πρόσβαση σε ένα περιβάλλον με εγκατεστημένη PHP και τις άλλες συσχετιζόμενες υπηρεσίες (π.χ. MySQL) και έναν επεξεργαστή κειμένου (editor)
- Μπορείτε να χρησιμοποιείτε όποιον editor θέλετε vi, emacs, notepad αλλά είναι καλύτερο να χρησιμοποιείτε κάποιο IDE
  - Στο εργαστήριο έχουμε εγκατεστημένα αρκετά εργαλεία για τις ανάγκες προγραμματισμού του μαθήματος

# Hello world

13

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8">
    <title>Hello World PHP Style</title>
</head>
<body>
    <?php
        echo 'Hello World from PHP!';
    ?>
</body>
</html>
```

# PHP Embedding Methods

14

- Συνήθως εισάγεται ακολουθώντας τα tags (**<?php ?>**)
  - ▣ Η συντόμευση **<? ?>** επιτρέπεται αλλά δεν συνίσταται όταν δεν ελέγχουμε απόλυτα το περιβάλλον στο οποίο θα τρέξει η εφαρμογή μας, διότι μπορεί να απαιτεί αλλαγές στο αρχείο **php.ini**
  - ▣ Θεωρείται γενικά ο προτιμότερος τρόπος
- Εναλλακτικά χρήση του **<script>** tag
  - ▣ `<script language="php">...</script>`
- Ακολουθώντας το στυλ ASP (**<% %>**)
  - ▣ Στόχος η γρήγορη εξοικείωση για τους προγραμματιστές ASP
  - ▣ Απαιτεί αλλαγή στο **php.ini**

# Direct echo

15

```
<input type="text" name="magic" value="<?= $foo ?>" />
```

- Μέσα στην HTML είναι σύνηθες να βρίσκονται PHP μεταβλητές οι οποίες εμφανίζονται άμεσα χρησιμοποιώντας το **<?= ?>**
- Εν προκειμένω, το **<?=** είναι συντόμευση του **<?php echo**
- Σε **MVC**-style αρχιτεκτονικές, χρησιμοποιείται συχνά στη συγγραφή των View scripts (πχ στο Wordpress)

# Ένταξη άλλων αρχείων

16

- Μια χρήσιμη δυνατότητα της PHP είναι η ένταξη κοινά χρησιμοποιούμενων scripts σε ένα αρχείο, πχ headers, footers, navigation, βιβλιοθήκες με συναρτήσεις ή classes κτλ
- Υπάρχουν πολλοί τρόποι για να εισάγουμε αρχεία στην PHP, συνηθέστερα χρησιμοποιούνται οι:
  - ▣ include
  - ▣ require
  - ▣ readfile()



# Ένταξη άλλων αρχείων

17

```
require 'global.php';  
include 'navbar.html';
```

- Συνηθέστερος τρόπος με την οδηγία **include**
  - ▣ Αν το αρχείο δεν βρεθεί η επεξεργασία της σελίδας θα συνεχιστεί αλλά θα δούμε μια προειδοποίηση
  - ▣ **Προσοχή:** Το περιεχόμενο του αρχείου δεν εισάγεται μόνο, αλλά και εκτελείται (αν είναι κώδικας PHP) ή εμφανίζεται
- Η οδηγία **require** είναι παρόμοια με την include, όμως αν το αρχείο δεν βρεθεί σταματάει η εκτέλεση (fatal error)
- **Δεν είναι συναρτήσεις** και άρα δεν χρειάζονται απαραίτητα παρενθέσεις

# Ένταξη άλλων αρχείων

18

```
readfile('footer.html');
```

- Η συνάρτηση **readfile()** διαβάζει και γράφει στην έξοδο τα περιεχόμενα ενός αρχείου
- Αν επιθυμούμε να διαβάσουμε μεν τα περιεχόμενα ενός αρχείου, αλλά να μην τυπωθούν στην έξοδο, μπορούμε να χρησιμοποιήσουμε την **file\_get\_contents()**
- Ουσιαστικά η εντολή **readfile('footer.html');** ισοδυναμεί με **echo file\_get\_contents('footer.html');**

# Καθορισμός των ρυθμίσεων της PHP

10

- Η PHP έχει πολλαπλές ρυθμίσεις για την ένταξη μεθόδων, συναρτήσεις που φορτώνονται, κτλ
  - ▣ Χρήση της **phpinfo()** για την εμφάνιση ρυθμίσεων της engine σε μια HTML σελίδα
  - ▣ Συνήθως αναζητείται το αρχείο php.ini για να γίνουν αλλαγές σε αυτό (αν ο χρήστης έχει τα κατάλληλα δικαιώματα)
    - Συνήθως αυτά ανήκουν στους διαχειριστές του εξυπηρετητή και όχι στον απλό χρήστη
  - ▣ Εκτελώντας **phpversion()** εμφανίζονται πληροφορίες για την έκδοση της php
    - Πχ: `echo 'Current PHP version: ' . phpversion();`

# Modules

20

- Η δυναμική της PHP είναι τα διάφορα modules τα οποία μπορούν να προστεθούν
- Όσα έχουν προστεθεί μπορούν να βρεθούν εκτελώντας σε ένα script την εντολή `get_loaded_extensions()`
  - ▣ Πχ: `print_r(get_loaded_extensions());`
- Οι συναρτήσεις ενός module μπορούν να εμφανιστούν εκτελώντας `get_extension_funcs(modname);`
  - ▣ Πχ: `print_r(get_extension_funcs('standard'));`

# Case Sensitivity

21

- Η PHP δεν είναι αυστηρά case-sensitive
  - ▣ Οι ενσωματωμένες **δομές** και **λέξεις-κλειδιά** της γλώσσας όπως while, class, if κ.ο.κ **δεν** είναι case-sensitive
  - ▣ Οι **classes** και οι **συναρτήσεις** επίσης **δεν** είναι case-sensitive
  - ▣ Οι **μεταβλητές** όμως είναι **case-sensitive**

# Statements και blocks

22

- Όπως και στις περισσότερες C-style γλώσσες:
  - ▣ Το τέλος των εντολών (statements) δηλώνεται με ';'
    - Παρατήρηση: Το τελευταίο statement δεν απαιτεί ';' πριν την ένδειξη τέλους του script (closing script delimiter)
  - ▣ Τα blocks οριοθετούνται με braces ({...})
  - ▣ Η PHP γενικά αγνοεί τα κενά (whitespace), αλλά όταν χρησιμοποιείται με συνδυασμό με άλλες γλώσσες (π.χ. HTML, JavaScript) για να γράφει στην έξοδο χρειάζεται προσοχή

# Ανάμειξη PHP & HTML

22

- Συχνά η PHP χρησιμοποιείται για να εκτυπωθεί κώδικας XHTML ή άλλος client-side κώδικας
  - ▣ `<?php print "<h1>Hey there!</h1>"; ?>`
- Εναλλακτικά:  
`<h1>`  
`<?php print "hey there!" ; ?>`  
`</h1>`
- Σε περίπτωση που επιλεγεί ο δεύτερος τρόπος τότε το στυλ της PERL για την έξοδο δεδομένων είναι πολύ χρήσιμο για την δημιουργία μεγαλύτερων block
- Δοκιμάστε τα παραπάνω παραδείγματα με κενά

# Εισαγωγή Σχολίων

2.4

- Τρεις τρόποι:
  - ▣ Στυλ UNIX
    - # I am a single line comment
  - ▣ Στυλ C
    - /\* I am a multi-line  
comment \*/
  - ▣ Στυλ C++
    - // I am a single line comment
- Τα σχόλια αφαιρούνται από την έξοδο από τον διερμηνευτή (όπως πρέπει)



# Ονόματα μεταβλητών

25

- Τα ονόματα των μεταβλητών ξεκινούν πάντα με '\$' και είναι **case-sensitive**
  - ▣ συνεπώς η \$Name είναι διαφορετική μεταβλητή από τις \$NAME \$NaMe
- Τα ονόματα μεταβλητών, πέρα από το αρχικό \$, μπορούν να αποτελούνται από γράμματα, αριθμούς και underscores (\_)
  - ▣ **Δεν πρέπει όμως να ξεκινούν με αριθμό**, πχ δεν επιτρέπεται η \$3\_stooges

# Τύποι δεδομένων

26

- Η PHP έχει 8 τύπους δεδομένων (data types)
  - ▣ 4 scalars
    - Integers
    - Floating-points
    - Strings
    - Booleans
  - ▣ 2 collections
    - Arrays
    - Object
  - ▣ 2 specials
    - Resource
    - Null

# Integers

27

- Decimal, octal, και hex τιμές επιτρέπονται
  - ▣ Π.χ. 754, -3, 0755 (octal), 0xFF (hex)
- Εύρος τιμών -2,147,483,648 έως 2,147,483,648
  - ▣ Μοιάζουν με τον τύπο long της C
  - ▣ Όταν υπερβείτε το εύρος τιμών τα δεδομένα πρέπει να μετατραπούν σε floating point τύπο μεταβλητής
- Μια τιμή μπορεί να ελεγχθεί αν είναι integer χρησιμοποιώντας τις **is\_int()** ή **is\_integer()**

# Floating point

28

- Απλό στυλ 3.14,-0.54 και επιστημονικό στυλ 0.314E10 ή 1.56E10
- Προσοχή στα προβλήματα ελέγχων με τις τιμές floating point
- Μια τιμή μπορεί να ελεγχθεί αν είναι floating point χρησιμοποιώντας τις **is\_float()** ή **is\_real()**

# Strings

20

- Ακολουθία χαρακτήρων οποιαδήποτε μήκους (δεν υπάρχει τύπος char)
- **Τα μονά και διπλά εισαγωγικά έχουν διαφορετική σημασία**
  - Αν βρεθεί μεταβλητή μέσα σε "" δίνει την τιμή της ενώ μέσα σε " είναι string
    - Προφανώς η εκτύπωση με " είναι γρηγορότερη καθώς δεν χρειάζεται να γίνει κάτι parse
    - Μην χρησιμοποιείτε διπλά εισαγωγικά χωρίς λόγο
  - Χαρακτήρες ελέγχου όπως \n,\"\\,\\t,\\ \\ κοκ λειτουργούν μόνο μέσα σε ""
- Χρησιμοποιώντας την **is\_string()** ελέγχουμε αν μια τιμή είναι string
- Υπάρχουν αμέτρητες συναρτήσεις χειρισμού string στην PHP

# Πίνακες (arrays)

30

- Πρώτο στοιχείο στην θέση μηδέν
- Κατασκευή με **array()**
  - ▣ `$stooges = array('Larry', 'Curly', 'Moe');`
  - ▣ `print $stooges[0];`
  - ▣ Απαρίθμηση των στοιχείων του πίνακα με την εντολή **foreach**  
`foreach ($stooges as $name) {  
 print "Hello $name <br />";  
}`
  - ▣ Πολλές διαθέσιμες συναρτήσεις για επεξεργασία πινάκων π.χ. `sort()`
  - ▣ Έλεγχος τύπου με **is\_array()**
- Η PHP υποστηρίζει και **associative arrays**!
  - ▣ `$foo['bar'] = 42;`
  - ▣ `array('foo' => true, 'bar' => 3.1415926);`

# Άλλοι τύποι δεδομένων

31

- Object
  - ▣ Θα το εξετάσουμε αργότερα
- Resource
  - ▣ Ειδικός τύπος δεδομένων που χρησιμοποιείται σαν file handle (στην πραγματικότητα είναι integer)
  - ▣ Συνήθως χρησιμοποιείται με συνδέσεις σε βάση
  - ▣ Έλεγχος τύπου με **is\_resource()**
- Null
  - ▣ Υποδηλώνει απουσία δεδομένων
  - ▣ Χρήσιμο για garbage collection
  - ▣ Έλεγχος τύπου με **is\_null()**

# Μεταβλητές

22

- Δηλώνονται αυτόματα κατά την πρώτη χρήση
- Μια μεταβλητή δεν περιορίζεται σε κάποιο τύπο δεδομένων, μπορεί να κρατήσει οποιονδήποτε τύπο
  - ▣ Είναι όμως πιο ασφαλές να χρησιμοποιείται πρώτα casting, πχ

```
$myName = (string) "Thomas";  
$favNum = (string) $favNum;
```
- Οι μεταβλητές πριν πάρουν τιμή είναι **null** (δεν υπάρχει εδώ διάκριση μεταξύ null και undefined)
- Το όνομα μιας μεταβλητής μπορεί να περιέχεται σε κάποιο string:
  - ▣

```
$foo = 'bar';  
$$foo = 'funky'; # τώρα έχουμε $bar = 'funky'
```



# References

22

- Οι μεταβλητές μπορούν να έχουν aliases
  - ▣ `$foo =& $bar` # τώρα το `$foo` είναι alias για το `$bar`
  - ▣ Προσοχή όμως γιατί με αυτό τον τρόπο αλλάζουν τιμές μεταβλητών με έμμεσο τρόπο
  - ▣ Μεγαλύτερη σύγχυση προκαλεί η προσπάθεια σβησίματος της τιμής (unset) μια και μπορεί να κρατηθεί από κάποιο alias
    - `$foo = "bar";`
    - `$foo2 =&$foo;`
    - `unset($foo);`
    - `print $foo2;` # Εμφανίζει bar
  - ▣ Τα aliases μπορούν να χρησιμοποιηθούν σαν παράμετροι σε συναρτήσεις για να αποφεύγεται η αντιγραφή μεγάλων σε όγκο δεδομένων
  - ▣ Δεν είναι pointers!

# Σταθερές

34

```
define('THIS_SCRIPT', 'home');  
define('THIS_SCRIPT', 'foo');  
echo THIS_SCRIPT // τυπώνει home
```

- Σταθερές μπορούν να οριστούν με το define construct
- Μια σύμβαση ονομασίας είναι να χρησιμοποιούνται ΚΕΦΑΛΑΙΑ για αυτές
- Μπορούν να κρατούν μόνο τιμές "μιας μεταβλητής" (scalar data), δηλαδή όχι λίστα, πίνακα κ.ο.κ
- Για την αποφυγή σύγχυσης με σταθερές όταν χρησιμοποιείται κώδικας άλλου προγραμματιστή μπορεί να χρησιμοποιηθεί η **get\_defined\_constants()**
- Υπάρχουν κάποιες "**μαγικές σταθερές**", οι οποίες είναι διαθέσιμες σε κάθε script (πχ `__LINE__`, `__FILE__`, `__FUNCTION__`, `__CLASS__`, `__METHOD__`) και συνήθως χρησιμοποιούνται για debugging

# Scope

25

- Μεταβλητές που ορίζονται σε μια συνάρτηση είναι τοπικές μεταβλητές
- Μεταβλητές που ορίζονται εκτός μιας συνάρτησης είναι global μεταβλητές
- Προφανώς οι παράμετροι μιας συνάρτησης έχουν τοπική εμβέλεια

# Scope

36

- Όταν αναφερόμαστε σε μια μεταβλητή \$foo εντός κάποιας συνάρτησης, θεωρείται ότι αναφερόμαστε στην τοπική (θυμηθείτε: αν δεν υπάρχει, δηλώνεται με τη χρήση), **είτε υπάρχει αντίστοιχη global είτε όχι**
- Για να δηλώσουμε ότι κάθε χρήση της \$foo θα αφορά την global μεταβλητή \$foo χρησιμοποιούμε το keyword **global**
  - global \$foo;

# Static Μεταβλητές

37

```
function foo() {  
    static $a = 1;  
    return $a++; // κάθε φορά αυξάνεται  
}
```

```
echo $a; // Κενό
```

- Μια μεταβλητή συνάρτησης που έχει μπροστά της το keyword **static** θα διατηρήσει τις τιμές της μεταξύ των κλήσεων της συνάρτησης

# Garbage collection

28

- Από τα παραπάνω είναι προφανές για ποιες μεταβλητές γίνεται Garbage Collection
- Η ανάθεση τιμής **null** ή καλύτερα η κλήση της **unset()** χρησιμοποιείται για την απελευθέρωση μνήμης
  - ▣ `$foo = null;`
  - ▣ `unset($foo);`

# Τελεστές

30

- Υποστηρίζονται όλοι οι τελεστές της Java με μόνη διαφορά τον τελεστή για συνένωση strings (string concatenation): Στην PHP χρησιμοποιείται η τελεία (.) αντί για το +
- Χρειάζεται προσοχή κατά το type conversion
- Μπορεί να χρησιμοποιηθεί casting (int), (float), (string), κτλ για την μετατροπή μιας μεταβλητής
  - ▣ `$a = "5"; $b = (int) $a;`

# Άλλοι τελεστές

40

- `===` και `!==` όπως στη JavaScript, για επιπρόσθετο έλεγχο ισότητας τύπων
- Υποστηρίζονται και **and**, **or**, **xor** αντί των λιγότερο κατανοητών `&&`, `||` (για το XOR δεν υπάρχει άλλος τελεστής)
- **@** για απόκρυψη σφάλματος
  - ▣ **Προσοχή:** Θεωρείται αρκετά αργό
- `` `` για εκτέλεση σε κέλυφος. Προσοχή!



# Δομές ελέγχου

41

- Υποστηρίζονται όλες οι δομές ελέγχου ροής της Java και η foreach για διάσχιση πινάκων
- Για τις περισσότερες δομές ελέγχου υποστηρίζεται και εναλλακτικός τρόπος γραφής. Συγκεκριμένα, για τις if, while, for, foreach, switch μπορούμε να αντικαταστήσουμε το opening brace ({} με : και το closing brace (}) με **endif; endwhile, endfor, endforeach, endswitch** αντίστοιχα

# Δομές ελέγχου: foreach

42

```
foreach ($array as $current)  
    statement or block;
```

- Επιτρέπει την απαρίθμηση στοιχείων σε ένα πίνακα
- Κατά την διάρκεια της επανάληψης είναι δυνατή η πρόσβαση και στον δείκτη και στις τιμές  

```
foreach ($array as $key => $value) { ... }
```
- Στις νεότερες εκδόσεις της PHP επιτρέπεται η μεταβλητή τιμής να είναι **reference** ώστε να τροποποιούμε τιμές χωρίς να απαιτείται σύνταξη τύπου `$array[$key]`  

```
foreach ($array as &$value) { $value *= 2; }
```

# Διακοπή εκτέλεσης

43

- return
  - ▣ Προκαλεί έξοδο από την συνάρτηση
- exit
  - ▣ Τερματίζει την τρέχουσα process/script
  - ▣ Είναι δυνατό το πέρασμα τιμής που υποδηλώνει status code ή μήνυμα λάθους
  - ▣ Είναι εναλλακτική της die( )
    - die("This script just died!");

# Κλήση Συναρτήσεων

44

```
$result = function_name([parameter, ...]);
```

- Γίνεται με τον ίδιο τρόπο είτε έχουν γραφτεί από το χρήστη είτε βρίσκονται στο σύστημα
- Δεν είναι αναγκαίο να παίρνουν παραμέτρους (ως συνήθως) ή να επιστρέφουν κάποια τιμή
- Παράδειγμα  

```
$len = strlen("UCSD"); // $len = 4
```

# Ορισμός Συναρτήσεων

45

```
function [&]function_name(parameter-list)
{
    statement(s) often with return(s)
}
```

- Ακολουθούν την παραπάνω γενική μορφή
- Παράδειγμα

```
function add2($x) {
    return $x+2;
}
```

```
$result = add2(5);
echo $result;
```

# Προαιρετικά ορίσματα

46

```
function add($x, $y=1) { return $x + $y; }  
echo add(5); // τυπώνει 6  
echo add(5, 10); // τυπώνει 15
```

- Όπως στη C++
- Προφανώς, ένα προαιρετικό γνώρισμα που προηγείται ενός υποχρεωτικού ουσιαστικά είναι επίσης υποχρεωτικό
- Σε αντίθεση με τη JavaScript, αν μια συνάρτηση κληθεί με λιγότερες παραμέτρους από αυτές που απαιτούνται θα εμφανιστεί προειδοποίηση

# Μεταβλητό πλήθος παραμέτρων

47

```
function foo() {  
    print 'I have ' . func_num_args();  
    print 'First argument 1 = ' . func_get_arg(0);  
    print_r(func_get_args());  
}
```

- Οι παράμετροι μιας συνάρτησης μπορούν να προσπελαστούν με τις συναρτήσεις **func\_num\_args()**, **func\_get\_arg()** και **func\_get\_args()**, ακόμα και μέσα στην ίδια τη συνάρτηση
- Η συνάρτηση **func\_get\_arg()** δέχεται έναν αριθμό που δηλώνει τη θέση της παραμέτρου που θέλουμε να επιστραφεί, ενώ η **func\_get\_args()** επιστρέφει έναν πίνακα με όλες τις παραμέτρους

# Επιστρεφόμενες Τιμές

48

- Το **return** μπορεί να χρησιμοποιηθεί για να επιστραφεί μια μόνο τιμή, πολλαπλές τιμές μπορούν να επιστραφούν μέσα σε ένα πίνακα
- Συνήθως η τιμή επιστρέφεται **με αντιγραφή**, αλλά για λόγους απόδοσης μπορούν οι τιμές να περαστούν με **reference**
  - Σε αυτή τη περίπτωση προστίθεται το **&** στον ορισμό της συνάρτησης με αποτέλεσμα οι επιστρεφόμενες τιμές να **μην** επιστρέφονται με αντιγραφή



# Πέρασμα αναφοράς (Reference)

40

- Συνήθως οι παράμετροι περνούν ως τιμές αλλά η εισαγωγή του `&` στο όνομα της παραμέτρου προκαλεί το πέρασμα με reference
- Αν έχουμε `function funref(&$x) {...}` και `funval($x) {...}` η συνάρτηση **funref** κάνει αλλαγές **απευθείας** στην `$x` ενώ στην **funval** αυτό δεν συμβαίνει
- **Προσοχή:** Κατά την κλήση μιας συνάρτησης, τίποτα δεν υποδεικνύει αν μια παράμετρος περνιέται με αναφορά ή όχι
  - ▣ Πχ στην κλήση **foo(\$bar);** η `$bar` περνιέται με αναφορά ή όχι;
    - Δεν μπορούμε να το συμπεράνουμε χωρίς τον ορισμό της συνάρτησης!

# Variable Functions

50

- Εδώ η PHP engine όταν συναντήσει μια έκφραση της μορφής `$x()`; αναζητά και εκτελεί τη συνάρτηση με όνομα ο,τι περιέχεται στη μεταβλητή `$x`
  - ▣ Αυτή η τεχνική μπορεί να χρησιμοποιηθεί για να υλοποιηθεί με κομψό τρόπο (ή με τρόπο που προκαλεί σύγχυση) λύσεις για function tables, callbacks κτλ
  - ▣ Παράδειγμα:  
`$x = 'foo';`  
`$x();` // καλεί την `foo()` εφόσον υπάρχει

# Anonymous Functions

51

```
$myfunction = create_function(args, code);
```

- Μπορεί να δημιουργηθούν χρησιμοποιώντας την συνάρτηση **create\_function**
  - ▣ Παράδειγμα:

```
$add2 = create_function('$x,$y', 'return $x+$y');  
echo $add2(5,10);
```
  - ▣ Ίδια μειονεκτήματα με τη χρήση του Function constructor στη JavaScript (αργή ταχύτητα, δύσκολο debugging)

52

# Αντικειμενοστραφής PHP

# Εισαγωγή

52

```
class CheckingAccount extends BankAccount {  
    private $balance;  
    // ...  
}
```

```
$my_account = new CheckingAccount();
```

- Στην PHP όπως και σε άλλες αντικειμενοστραφείς γλώσσες:
  - ▣ Ορίζουμε κλάσεις με τη λέξη-κλειδί **class**
  - ▣ Δημιουργούμε νέα instances με το keyword **new**
  - ▣ Κληρονομικότητα με τη λέξη-κλειδί **extends**
  - ▣ Έλεγχος πρόσβασης με τα keywords **public**, **protected**, **private**
  - ▣ **try...catch** που λειτουργεί όπως και στη Java
    - Διατίθενται ωστόσο πολύ λιγότερα built-in exceptions
  - ▣ abstract classes με τη λέξη-κλειδί **abstract**

# Properties & Methods

54

```
class BankAccount{  
    private $balance;  
    public function setBalance($balance){  
        $this->balance = $balance;  
    }  
    public function getBalance(){  
        return $this->balance;  
    }  
}
```

- Στην PHP, ο τελεστής πρόσβασης σε ιδιότητες και μεθόδους είναι ο `->` (σε αντίθεση με την τελεία `.` της Java και της JavaScript)
- **\$this**: Αντίστοιχο του **this** της Java

# Static ιδιότητες/μέθοδοι

55

```
class BankAccount {  
    private static $bankname;  
    public function getBankName(){  
        return self::$bankname;  
    }  
}
```

- Όπως και στη Java, χρησιμοποιούμε το keyword **static** για να ορίσουμε static μεταβλητές
- Ωστόσο, χρησιμοποιείται για να έχουμε πρόσβαση σε static ιδιότητες/μεθόδους, δεν μπορούμε να χρησιμοποιήσουμε τον τελεστή ->, αλλά χρησιμοποιείται ο ::
- Η λέξη-κλειδί **self** ισοδυναμεί με τη χρήση του ονόματος της κλάσης
  - ▣ Συνιστάται η χρήση της, για λόγους ευκολότερης συντήρησης

# Σταθερές

56

```
class BankAccount {  
    const MIN_BALANCE = 20;  
    public __construct($initial_balance){  
        if($initial_balance < self::MIN_BALANCE) {  
            throw new Exception('More money needed :(');  
        }  
    }  
}
```

- Χρησιμοποιείται η λέξη-κλειδί **const**
- Προσπέλαση σε αυτές γίνεται με τον ίδιο τρόπο με τις static ιδιότητες
- Προσέξτε ότι το όνομα τους δεν αρχίζει με \$
- Για μεθόδους, διατίθεται το keyword **final**, όπως και στη Java



# Magic methods

57

- Η PHP ορίζει ένα πλήθος μεθόδων που ξεκινούν με **\_\_ (δύο underscores)** οι οποίες όταν οριστούν σε μια class επιτελούν κάποια ειδική λειτουργία
- Ενέργειες για τις οποίες στη Java κάνουμε override μεθόδους της τάξης Object, στην PHP υλοποιούνται με "μαγικές" μεθόδους (πχ `__toString`, `__clone`)
- Στα επόμενα θα εξετάσουμε αρκετές από αυτές

# Constructors & destructors

58

```
function __construct(){  
    $this->balance = 50;  
}
```

- Constructors (κατασκευαστές) ορίζουμε με τη μαγική μέθοδο **\_\_construct**
  - ▣ **Προσοχή:** ξεκινάει με **2** underscores!
  - ▣ Μόνο **ένας** κατασκευαστής επιτρέπεται σε κάθε ορισμό αντικειμένου
  - ▣ Σε περίπτωση που επιθυμούμε πολυμορφισμό κατασκευαστών, μόνη λύση ο έλεγχος των παραμέτρων (όπως στη JavaScript)
- Destructors αντίστοιχα με την **\_\_destruct**

# Getters & Setters

50

- Τι γίνεται όταν επιχειρείται προσπάθεια ή τροποποίηση ανύπαρκτης ή κρυφής ιδιότητας;
- Μπορούμε να το ορίσουμε, αν δηλώσουμε τις "μαγικές" μεθόδους **\_\_get(\$property)** και **\_\_set(\$property, \$value)**
- Με κατάλληλους ελέγχους μπορούμε να ορίσουμε διαφορετική συμπεριφορά ανάλογα με το όνομα (και/ή την τιμή σε περίπτωση set)
- Συνήθως χρησιμοποιείται για να έχουμε ιδιότητες οι οποίες φαίνονται σαν public αλλά όταν διαβάζονται ή τροποποιούνται εκτελείται και επιπλέον κώδικας
- Μας επιτρέπουν να αποφύγουμε τη δήλωση περιττών getters και setters για προληπτικούς λόγους, όπως συχνά μας αναγκάζει η Java

# \_\_call() & \_\_callStatic()

60

```
public function __call($name, $arguments) {  
    echo "Calling object method '$name' " . implode(', ', $arguments);  
}  
  
public static function __callStatic($name, $arguments) {  
    echo "Calling static method '$name' " . implode(', ', $arguments);  
}
```

- Άλλο ένα παράδειγμα "μαγικών" μεθόδων
- Αν έχουν οριστεί, καλούνται **όταν επιχειρηθεί να κληθεί μη ορατή μέθοδος** (είτε λόγω πρόσβασης, είτε επειδή απλά δεν υπάρχει)
- Η **\_\_call()** αφορά κλήση σε instance, η **\_\_callStatic()** κλήση σε στατικό context
- Καλούνται με πρώτη παράμετρο το όνομα της μεθόδου και δεύτερη ένα array με τις παραμέτρους

# Serialization

61

- Η PHP διαθέτει τις μεθόδους **serialize()** και **unserialize()** για τη μετατροπή ενός αντικειμένου σε string ή την αποκωδικοποίηση μιας τέτοιας συμβολοσειράς, αντίστοιχα
- Συχνά χρησιμοποιούνται για αποθήκευση αντικειμένων σε βάση δεδομένων, αν και συνήθως κάτι τέτοιο δυσχεραίνει το χειρισμό των δεδομένων

# \_\_sleep() & \_\_wakeup()

62

- Όταν καλείται η **serialize()** για κάποιο αντικείμενο, καλείται πρώτα η μέθοδος **\_\_sleep()** αν έχει οριστεί σε αυτό
  - ▣ Μπορεί να χρησιμοποιηθεί για να πραγματοποιηθεί κάποιος "**καθαρισμός**" του αντικειμένου πριν τη σειριοποίηση
  - ▣ Πρέπει να επιστρέφει **ένα array με τα ονόματα των ιδιοτήτων** που πρέπει να σειριοποιηθούν
- Αντίστοιχα, η **\_\_wakeup()** καλείται όταν κληθεί η **unserialize()**
  - ▣ Μπορεί να χρησιμοποιηθεί για να ανακτήσει πράγματα που χάθηκαν κατά τη σειριοποίηση
    - Πχ κάποια σύνδεση με μια βάση δεδομένων

# Cloning

63

```
$copy_of_object = clone $object;
```

- Κλωνοποιούμε ένα αντικείμενο, χρησιμοποιώντας τη λέξη-κλειδί **clone**
- **Shallow copy** των ιδιοτήτων
- Αν έχει οριστεί μέθοδος **\_\_clone()** στο αντικείμενο, καλείται μετά τη δημιουργία του κλώνου για να αλλάξει τυχόν μεταβλητές που απαιτείται (πχ για deep copying)

# Callable classes

64

```
class CallableClass {  
    function __invoke($x) {  
        var_dump($x);  
    }  
}  
$obj = new CallableClass();  
$obj(5); // int(5)  
var_dump(is_callable($obj)); // bool(true)
```

- Από την PHP 5.3 υποστηρίζεται η μαγική μέθοδος **\_\_invoke**, η οποία καλείται όταν επιχειρηθεί να κληθεί ένα στιγμιότυπο ως συνάρτηση
- Ελέγχουμε αν ένα αντικείμενο είναι callable με τη συνάρτηση **is\_callable()**





# Built-in objects & functions

# Συναρτήσεις για String

66

- Σχεδόν 100 συναρτήσεις για χειρισμό συμβολοσειρών
- Συνηθέστερα χρησιμοποιούμενες:
  - ▣ **strlen()** μήκος string σε χαρακτήρες
  - ▣ **strtolower(), strtoupper, ucfirst(), ucwords()** μετατροπές case
  - ▣ **trim(), ltrim(), rtrim()** αφαίρεση κενών από την αρχή και/ή το τέλος
  - ▣ **explode()** διάσπαση string σε τμήματα και επιστροφή τους σε array
  - ▣ **substr()** επιστροφή τμήματος (substring) μιας συμβολοσειράς
  - ▣ **strpos(), stripos()** Σε ποια θέση υπάρχει (αν υπάρχει) μια υποσυμβολοσειρά;
  - ▣ **urlencode(), urldecode()** (από)κωδικοποίηση συμβολοσειράς βάσει του τρόπου που χρησιμοποιείται στα URI για GET parameters

# Συναρτήσεις για χειρισμό arrays

67

- Σχεδόν 80 συναρτήσεις για χειρισμό πινάκων
- Συνηθέστερες:
  - ▣ **sizeof** Πλήθος τιμών στον πίνακα
  - ▣ **implode** Ένωση τιμών του πίνακα σε string
    - αντίστροφη της **explode**
  - ▣ **in\_array** Έλεγχος παρουσίας τιμής στον πίνακα
  - ▣ **array\_keys** Επιστροφή των κλειδιών σε νέο πίνακα
  - ▣ **array\_merge** Συγχώνευση πινάκων
  - ▣ **array\_unique** Εξάλειψη διπλότυπων
  - ▣ **asort, arsort, ksort, krsort, usort, uasort, uksort**  
Ταξινόμηση πίνακα

# Έξοδος

68

- Είτε για εκτύπωση της διεπαφής και των δεδομένων, είτε για debugging
- **echo, print** Τυπώνουν μια συμβολοσειρά στην έξοδο
- Άλλοι τρόποι για την έξοδο δεδομένων:
  - ▣ **number\_format(), money\_format()**
  - ▣ **printf(), sprintf()** Εκτύπωση ή επιστροφή συμβολοσειράς με
  - ▣ Για debugging:
    - **print\_r()** για εκτύπωση πινάκων αναδρομικά
    - **var\_dump()** εκτύπωση τύπου και τιμής

# Μαθηματικές Συναρτήσεις

- Σχεδόν 50 μαθηματικές συναρτήσεις
- Επιλεγμένες συναρτήσεις:
  - ▣ **abs()** απόλυτη τιμή
  - ▣ **round(), ceil(), floor()** για στρογγυλοποιήσεις
  - ▣ **is\_finite(), is\_infinite(), is\_nan()** έλεγχος του τύπου του αριθμού
  - ▣ **rand(), srand()** δημιουργία ψευδοτυχαίων αριθμών



# Web development με PHP

# Superglobals

71

- Πίνακες που είναι διαθέσιμοι σε κάθε script
- Δεν χρειάζεται **global** keyword για να αναφερθούμε σε αυτές εντός συναρτήσεων
- Κυριότερες οι:
  - ▣ **\$\_GET** Δεδομένα που περάστηκαν μέσω GET
  - ▣ **\$\_POST** Δεδομένα που περάστηκαν μέσω POST
  - ▣ **\$\_COOKIE** Cookies που υπάρχουν
  - ▣ **\$\_REQUEST** = **\$\_POST** + **\$\_GET** + **\$\_COOKIE**
  - ▣ **\$\_SESSION** Μεταβλητές συνόδου
  - ▣ **\$\_SERVER** Στοιχεία για το τρέχον request και το περιβάλλον του εξυπηρετητή
  - ▣ **\$\_FILES** Αρχείο/α που ανέβασε ο χρήστης (αν υπάρχουν)

# Χειρισμός φορμών στην PHP

72

```
<form action="reply.php?t=1234" method="POST">
  <input type="text" name="title" />
  <textarea name="message"></textarea>
  <button type="submit">Post</button>
</form>
```

HTML

```
<?php
echo 'Δημοσιεύτηκε το μήνυμα σας με τίτλο "' .
$_POST['title'] . '" στη συζήτηση με ID #' . $_GET['id'];
?>
```

PHP

ή

```
<?php
echo 'Δημοσιεύτηκε το μήνυμα σας με τίτλο "' .
$_REQUEST['title'] . '" στη συζήτηση με ID #' . $_REQUEST['id'];
?>
```

PHP



# Registered globals

72

- Υπάρχει η δυνατότητα να δημιουργούνται αυτόματα μεταβλητές για κάθε form field
  - ▣ Πχ στο προηγούμενο παράδειγμα, θα δημιουργούνταν αυτόματα μεταβλητές **\$title** και **\$message**
- Σε πολλές εκδόσεις PHP το παραπάνω δεν είναι διαθέσιμο εξ ορισμού για λόγους ασφάλειας και προστασίας του κώδικα, όμως μπορεί να ενεργοποιηθεί στο αρχείο **php.ini**
  - ▣ Πρόβλημα: Πέρασμα (inject) δεδομένων κατευθείαν στην εφαρμογή με GET ή POST
- Έλεγχος
  - ▣ `ini_get('register_globals');`
  - ▣ Η χρήση της `ini_set()` ενεργοποιεί τα `register_globals` κατά την διάρκεια εκτέλεσης αλλά δεν έχει κανένα νόημα

# Importing Request Variables

74

```
// This will import GET and POST vars with an "rvar_" prefix
import_request_variables('gp', 'rvar_');
echo $rvar_title;
```

- Αν επιμένετε στην χρήση registered globals μπορείτε να χρησιμοποιήσετε την συνάρτηση **import\_request\_variables(*what* [,*prefix*])**
  - ▣ Στη θέση του πρώτου ορίσματος υπάρχει **g**, **p** και/ή **c** που υποδηλώνει ότι επιθυμούμε εισαγωγή τιμών με **GET**, **POST**, **COOKIE** αντίστοιχα
  - ▣ Η τιμή prefix είναι προαιρετική και θα προστεθεί στην αρχή σημαντικών μεταβλητών. Για λόγους ασφάλειας συνίσταται η χρήση της.

# Self Posting

75

```
if(form data not collected)
    // print form fields and allow user to submit
else if(invalid form data)
    // print form fields with errors and allow resubmit
else
    // perform form action (and often redirect)
```

- Μια προγραμματιστική επιλογή που συχνά χρησιμοποιείται (όχι μόνο στην PHP) είναι η ενσωμάτωση της φόρμας και της απάντησης στο ίδιο αρχείο.
- Η βασική ιδέα είναι η σελίδα να διαφοροποιείται ανάλογα αν συλλέγει δεδομένα από το χρήστη ή αν αποκρίνεται στην εισαγωγή δεδομένων από τον χρήστη. Φυσικά μπορεί να επιτελεί επίσης έλεγχο εγκυρότητας των δεδομένων

# Self Posting

76

```
<form action="reply.php?t=1234" method="POST">
  <!-- controls -->
  <button type="submit" name="submitted">Post</button>
</form>
```

HTML

```
if(isset($_POST['submitted'])) { ... }
```

PHP

- Ένας εύκολος τρόπος να προσδιοριστεί το action σε μια self posting φόρμα είναι η χρήση της τιμής **`$_SERVER['PHP_SELF']`** στο form action ή το **κενό**
- Πώς όμως ξέρουμε αν η φόρμα υποβλήθηκε;
  - ▣ Απόδοση ονόματος στο submit button και έλεγχος αν υπάρχει
  - ▣ Hidden input
    - `<input type="hidden" name="submitted" value="1">`

# Always validate

77

- ❑ **Ποτέ μην εμπιστεύεστε τα δεδομένα που προέρχονται από το χρήστη.**
  - ❑ Αναλογιστείτε μόνο πόσα μπορείτε να αλλάξετε ακόμη και μόνο με το Firebug
  - ❑ Και οι πιο καλοπροαίρετοι χρήστες κάνουν λάθη
    - Παράλειψη συμπλήρωσης κάποιου πεδίου
    - Λάθος μορφή στα δεδομένα
      - Σε πολλές περιπτώσεις, είναι σφάλμα όσον αφορά την ευχρηστία να απαιτείτε συγκεκριμένη μορφή στα δεδομένα
        - πχ τηλεφωνικοί αριθμοί χωρίς παύλες ή κενά. Απλά αφαιρέστε τα!

# Βασικοί Έλεγχοι Εγκυρότητας

78

- **isset()** Έλεγχος ότι μια μεταβλητή έχει τιμή
- **trim()** Αφαίρεση κενών
- **empty()** Ελέγχει αν μια τιμή απουσιάζει
- Type casting
  - ▣ `$age = (integer) $_POST['age'];`
- Έλεγχος εύρους και μορφής τιμών
  - ▣ If statements
  - ▣ Regular expressions ([preg\\_match](#))
  - ▣ Φίλτρα ([filter\\_var](#), από PHP 5.2)

# Headers & Environment

70

- Ένα request περιέχει διάφορες χρήσιμες πληροφορίες στο header όπως
  - ▣ User Agent, Accept, Accept-Language, κ.ο.κ
- Μπορεί επιπλέον να βρεθεί και το περιβάλλον από το οποίο έρχεται το request, που αρκετές φορές είναι χρήσιμο
  - ▣ Χρόνος πρόσβασης και διεύθυνση IP μπορεί να είναι χρήσιμα σε μερικές εφαρμογές

# Headers & Environment

- Τα πιο ενδιαφέροντα δεδομένα παρέχονται από τον superglobal πίνακα **\$\_SERVER[]**
  - ▣ Όλα τα HTTP headers θα αρχίζουν με "HTTP\_"
    - HTTP\_ACCEPT, HTTP\_REFERER, HTTP\_USER\_AGENT, κτλ
  - ▣ Τα υπόλοιπα θα είναι μεταβλητές
    - PHP\_SELF, REMOTE\_ADDR, REMOTE\_HOST, REQUEST\_METHOD
    - Περισσότερα: <http://gr.php.net/reserved.variables>



# Header Output

81

```
header('Content-Type: image/png');
```

- Οι HTTP headers μπορούν να ρυθμιστούν με χρήση της συνάρτησης **header()**
- Headers μπορούμε να ορίσουμε μόνο προτού σταλεί οποιαδήποτε έξοδος
  - ▣ Σε αντίθεση περίπτωση θα προκληθεί fatal error (*"Cannot modify header information -headers already sent"*)
  - ▣ Το ίδιο ισχύει φυσικά και για τα cookies, τα οποία εδώ ορίζουμε με την [setcookie\(\)](#)



# Παράδειγμα

Απλή φόρμα επικοινωνίας

# HTML

82

```
<form action="contact.php" method="post" id="contactform">
  <ol>
    <li><label>Όνομα: <input type="text" name="name"></label></li>
    <li><label>Email: <input type="email" name="email"></label></li>
    <li><label>Website: <input type="url" name="website"></label></li>
    <li><label>Το μήνυμά σας: <textarea name="message"></textarea></label></li>
    <li>
      <label>
        <input type="checkbox" name="subscribe" value="1">
        Θέλω να λάβω ένα αντίγραφο
      </label>
    </li>
  </ol>
  <button type="submit">Αποστολή</button>
</form>
```

- Θυμηθείτε τη φόρμα επικοινωνίας του 1<sup>ου</sup> εργαστηρίου...
- Σήμερα θα δούμε πώς θα μπορούσε να λειτουργεί

# Sending mail

84

```
<?php
```

```
$sender = $_POST['name'] . '<' . $_POST['email'] . '>';
```

```
$message = $_POST['message'] . "\r\n\r\nWebsite:" .  
$_POST['website'];
```

```
mail(  
    'auebweb@gmail.com' .  
        ($_POST['subscribe']? ', ' . $sender : ''),  
    'New mail from contact form' // Subject  
    $message,  
    'From:' . $sender; // Extra headers  
);  
?>
```

- Συνάρτηση mail() για αποστολή emails

# Βασικό Validation

85

- Το παραπάνω λειτουργεί, αλλά δεν αρκεί
- **Μη user-friendly**
  - ▣ Σε περίπτωση λάθους ο χρήστης δεν καταλαβαίνει τι ακριβώς πήγε στραβά
  - ▣ Σε περίπτωση που όλα πάνε καλά, ο χρήστης δεν ενημερώνεται
- **Μη ασφαλές:** Μπορεί εύκολα να χρησιμοποιηθεί για αποστολή spam μέσω του δικού μας server!

# Βασικό Validation

86

```
<?php
```

```
$errors = array();
```

```
// Υπάρχει όνομα;
```

```
if(!$ _POST['name']) {
```

```
    $errors[] = 'δεν συμπληρώσατε το όνομα σας';
```

```
}
```

```
// Υπάρχει email; Μοιάζει με (ένα) email;
```

```
if(!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
```

```
    $errors[] = 'Το email δεν φαίνεται σωστό';
```

```
}
```

```
// Υπάρχει μήνυμα;
```

```
if(!$ _POST['message']) {
```

```
    $errors[] = 'Δεν γράψατε μήνυμα';
```

```
}
```

# Putting it together

87

```
if(sizeof($errors) > 0) {  
    // Υπάρχουν σφάλματα, εμφάνιση αυτών  
    echo 'Συνέβησαν σφάλματα:<ul><li>' .  
        implode('</li><li>', $errors) . '</li></ul>';  
}  
else {  
    // Δεν υπάρχουν σφάλματα  
    // αποστολή email  
    echo 'Το email εστάλη επιτυχώς.';  
}  
  
?>
```




# Παράδειγμα

Έρευνα Δημοτικότητας του Μαθήματος



# Σύνοψη Έρευνας Δημοτικότητας

- Η βασική λειτουργία της εφαρμογής είναι να παρέχει στους χρήστες μια φόρμα στην οποία μπορούν να δηλώσουν αν τους αρέσει ή όχι το μάθημα.
- Η αρχιτεκτονική της εφαρμογής αποτελείται από δύο μέρη:
  - ▣ Μια HTML φόρμα
  - ▣ Κώδικά PHP που διαβάζει και γράφει ένα αρχείο και εμφανίζει HTML κώδικα με τα αποτελέσματα.
-  survey\_form.html, survey.php, survey\_result.php

# Αποθήκευση σε αρχείο κειμένου

- Χρήση `fopen()` για να ανοίξει το αρχείο για I/O
  - ▣ `fopen(filename, mode)`
    - Η συνάρτηση επιστρέφει ένα resource (filehandle)
    - filename: μονοπάτι αρχείου
    - modes: r,r+,w,w+,a,a+,x,x+
    - Πχ: `$fh = fopen('file.txt', "a+");`
      - Ανοίγει ή δημιουργεί ένα αρχείο για γράψιμο και διάβασμα
    - <http://php.net/function.fopen>

# Εγγραφή σε αρχείο κειμένου

01

- Χρήση της `fwrite()` για εγγραφή δεδομένων στο αρχείο
  - ▣ `fwrite(filehandle, datastring)`
  - ▣ `filehandle`: resource αρχείου
  - ▣ `datastring`: string που έχει τα δεδομένα για εγγραφή
  - ▣ Παράδειγμα: `fwrite($fh, "some data");`
  - ▣ Manual: <http://php.net/function.fwrite>
- Κλείσιμο αρχείου με `fclose($fh)`

# HTML Form Source

02

```
<!doctype html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
    charset=utf-8">
    <title>PHP Example</title>
  </head>
  <body>
    <h3>Do you like the class so far?</h3>
    <form name="survey" action="survey.php" method="POST"  >
      Yes: <input type="radio" name="vote" value="0" /><br />
      No: <input type="radio" name="vote" value="1" /><br />
      <input type="submit" value="Submit" />
    </form>
  </body>
</html>
```

# PHP Source

03

```
<?php
```

```
    $vote = $_REQUEST['vote'];
```

```
    //get content of textfile
```

```
    $filename = "survey_result.txt";
```

```
    $content = file($filename);
```

```
    //put content in array
```

```
    $array = explode("||", $content[0]);
```

```
    $yes = $array[0];
```

```
    $no = $array[1];
```

```
    if ($vote == 0) { $yes = $yes + 1; }
```

```
    if ($vote == 1) { $no = $no + 1; }
```

```
    //insert votes to txt file
```

```
    $insertvote = $yes."||".$no;
```

```
    $fp = fopen($filename,"w");
```

```
    fputs($fp,$insertvote);
```

```
    fclose($fp);
```

```
?>
```

Διάβασμα του αρχείου  
"survey\_result.txt"

String tokenization με διαχωριστή το  
"||" και αποθήκευση σε πίνακα

Αποθήκευση του περιεχομένου του  
αρχείου σε μεταβλητές και  
ενημέρωση τους ανάλογα με την  
είσοδο του χρήστη.

Αποθήκευση των νέων τιμών στο  
αρχείο διατηρώντας το μορφότυπο.

# PHP Source (Continued)

04

```
...  
<h2>Result:</h2>  
<table>  
<tr>  
<td>Yes:</td>  
<td>  
<?php echo(100*round($yes/($no+$yes),2)); ?>%  
</td>  
</tr>  
<tr>  
<td>No:</td>  
<td>  
<?php echo(100*round($no/($no+$yes),2)); ?>%  
</td>  
</tr>  
</table>
```

Απλή εμφάνιση των  
αποτελεσμάτων

95

## Σχεδίαση δυναμικών γραφικών

# GD

06

- Οι εκδόσεις 4.3 και έπειτα ενσωματώνουν την βιβλιοθήκη **GD** για την δυναμική δημιουργία γραφικών
  - ▣ αντίστοιχη με αυτή που επιτρέπει το canvas 2D API στη JavaScript
- Με αυτήν μπορούν να δημιουργηθούν και να επεξεργαστούν αρχεία εικόνων σε διάφορα format, όπως gif, png, jpg, bmp, και xpm χρησιμοποιώντας PHP
- Περιορισμένες δυνατότητες
- Συχνά κακή ποιότητα αποτελέσματος (πχ απουσία anti-aliasing για ορισμένες ενέργειες)



# ImageMagick

07

- Για εφαρμογές που απαιτούν σοβαρή επεξεργασία εικόνας, το GD δεν επαρκεί
- Σε αρκετές εγκαταστάσεις PHP διατίθεται και η βιβλιοθήκη ImageMagick
  - ▣ Είναι γραμμένη σε C++ αλλά διατίθεται και PHP API
  - ▣ Πολύ περισσότερες δυνατότητες
  - ▣ Πολύ καλύτερη ποιότητα αποτελέσματος
  - ▣ Δυσκολότερη εγκατάσταση και εκμάθηση

# Παράδειγμα χρήσης GD

08

```
// MIME type που υποδηλώνει (png) εικόνα
header('Content-type: image/png');
// Δημιουργία εικόνας
$img_handle = imagecreate(120, 30) or die('Cannot Create image');
// Allocation χρωμάτων που θα χρησιμοποιήσουμε
$back_color = imagecolorallocate($img_handle, 255, 255, 0);
$txt_color = imagecolorallocate($img_handle, 255, 0, 0);
// Σχεδίαση του string "Hello world!" στο σημείο (5,5)
imagestring($img_handle, 31, 5, 5, 'Hello world!', $txt_color);
// Τύπωμα της εικόνας στην έξοδο
imagepng($img_handle);
```

# Βασικά σχήματα

- Η βιβλιοθήκη υποστηρίζει την δημιουργία γραμμών, απλών σχημάτων κτλ. Για παράδειγμα:
  - ▣ `imageline($im, x1,y1,x2,y2,color)`
  - ▣ `imagerectangle ( $im, x1, y1, x2, y2, color )`
  - ▣ `imageellipse($im, cx,cy height, width, color)`
  - ▣ `imagepolygon($im, array of points, numpoints, color)`
  - ▣ `imagefilledpolygon($im,array of points, numpoints, color)`
  - ▣ `imagefill($im,x,y,color)`