

الكافى

PHP

الجزء الاول

ابو حبيب الحسينى

1 الكافى فى PHP الجزء الاول Abu Habib Al-Husini

ملحوظة مهمة جدا

ان وجود الكلمات الانجليزية فى وسط الجمل العربية ينقل بعض الكلمات من مكانها فتظهر الجمل بشكل غير صحيح ويصعب فهما وهذا عيب فى الترميز (يو تى اف)

مثال على هذا الكلام

النصي في أي مكان في المستند PHP يمكن وضع برنامج
:>? وينتهي بـ php?< بـ PHP يبدأ الكود

لاحظ هنا ان الجملة اصبحت غير مفهومة الى حد ما او غير مرتبة بشكل صحيح لان بعض الكلمات نقلت من مكانها بسبب وضع كلمات انجليزية وسط الجمل العربية فافى مثل هذه الحالات حاول ان تستنتج الجمله بنفسك وتفهما

حاولنا تقليل هذا العيب قدر المستطاع باستخدام بكتابة المصطلحات الانجليزية باللغة العربية مثل (سي اس اس) او (نود جى اس) وكذلك نقلنا اتجاة الصفحة من اليسارى الى اليمين لتفادى هذا العيب وللأسف لم تتم المعالجة بنسبة مئة بالمئه

المرجع العربي السريع

مرجع شامل لكل دوال وكلاسات واجرائات جملا انشاء وترجمة الذكاء الاصطناعي حيث يشرح مهام عشرات الاف من دوال واجرائات جملا باللغة العربية



ابو حبيب الحسيني



4 الكافي في PHP الجزء الاول Abu Habib Al-Husini

الفهرس

| | |
|---------------------------------|----|
| ملحوظة مهمة جدا..... | 2 |
| الفهرس..... | 4 |
| مقدمة سريعة..... | 9 |
| قبل ان تبدا..... | 14 |
| قم بإعداد جهاز الكمبيوتر..... | 15 |
| بناء الجمل الأساسي..... | 15 |
| حساسية الحالة..... | 16 |
| تعليقات..... | 18 |
| التعليقات في..... | 18 |
| متغيرات..... | 21 |
| إنشاء (الإعلان) عن متغيرات..... | 21 |
| متغيرات..... | 22 |
| متغيرات الإخراج..... | 22 |
| هي لغة مكتوبة بشكل واسع..... | 24 |
| نطاق متغيرات..... | 24 |
| النطاق العام والمحلي..... | 25 |
| متغير عام من داخل الدالة..... | 26 |
| الكلمة المحجوزة الثابتة..... | 28 |
| إخراج وطباعة البيانات..... | 29 |
| إخراج وطباعة الاكواد..... | 29 |
| كود اخراج..... | 30 |
| أمر الاخراج والعرض..... | 31 |
| أنواع البيانات..... | 32 |
| أنواع الاكواد..... | 32 |
| نصوص..... | 33 |
| عدد صحيح..... | 33 |
| الرقم العشري..... | 34 |
| منطقية..... | 35 |
| المصفوفة..... | 35 |
| كائن..... | 36 |
| القيمة الفارغة..... | 37 |

| | |
|--|----|
| الموارد..... | 38 |
| وظائف نصوص..... | 39 |
| strlen() - إرجاع طول النصوص..... | 39 |
| str_word_count() - حساب الكلمات في النصوص..... | 39 |
| strrev() - عكس نصوص..... | 40 |
| strpos () - البحث عن نص داخل نصوص..... | 40 |
| str_replace() - استبدال النص داخل نصوص..... | 41 |
| أكمل مرجع النصوص..... | 42 |
| الأرقام..... | 42 |
| الأعداد الصحيحة..... | 42 |
| الأرقام العشرية..... | 44 |
| إنفينيتي..... | 45 |
| نان..... | 46 |
| النصوص العددية..... | 47 |
| التحويل..... | 48 |
| الرياضيات..... | 49 |
| pi() ووظيفة..... | 49 |
| وظائف min() و max()..... | 49 |
| abs() ووظيفة..... | 50 |
| sqrt() ووظيفة..... | 50 |
| تقريب الأرقام ذو الفاصلة العشرية..... | 51 |
| أرقام عشوائية..... | 51 |
| استكمال مرجع الرياضيات..... | 52 |
| ثوابت..... | 53 |
| إنشاء ثابت..... | 53 |
| المصفوفات الثابتة..... | 55 |
| الثوابت عامة..... | 55 |
| المشغلات..... | 56 |
| مشغلي الحساب..... | 57 |
| مشغلي تعيين..... | 57 |
| مشغلي المقارنة..... | 58 |
| زيادة / إنقاص عوامل التشغيل..... | 59 |
| العوامل المنطقية..... | 60 |
| مشغلي نصوص..... | 60 |

| | |
|---|----|
| مشغلي مصفوفة..... | 61 |
| مشغلي التعيين الشرطي..... | 61 |
| الجمل الشرطية..... | 62 |
| كود if..... | 62 |
| - كود if...else..... | 63 |
| - كود if...elseif...else..... | 64 |
| كود التبديل..... | 65 |
| حلقات..... | 67 |
| العمل أثناء الحلقة..... | 68 |
| كيف تتم هذه الحلقة... .. | 70 |
| حلقة فور..... | 72 |
| حلقة foreach..... | 74 |
| حلقة foreach..... | 74 |
| كلمة بريك..... | 76 |
| متابعة..... | 77 |
| تابع الحلقات..... | 77 |
| وظائف مضمنة..... | 79 |
| الوظائف المحددة من قبل المستخدم..... | 79 |
| إنشاء وظيفة محددة من قبل المستخدم في..... | 79 |
| وسيطات دالة..... | 81 |
| التوسع في الكتابة..... | 82 |
| قيمة الوسيطة الافتراضية..... | 84 |
| وظائف - إرجاع القيم..... | 85 |
| تعريفات نوع الإرجاع..... | 85 |
| تمرير البرامترات حسب المرجع..... | 87 |
| ما هي المصفوفة؟..... | 88 |
| إنشاء مصفوفة في..... | 89 |
| count() احصل على طول المصفوفة - دالة..... | 90 |
| المصفوفات المفهرسة..... | 90 |
| حلقة على مصفوفة مفهرس..... | 91 |
| المصفوفات الترابطية..... | 92 |
| حلقة من خلال مصفوفة النقاىي..... | 93 |
| المصفوفات متعددة الأبعاد -..... | 94 |
| صفائف ثنائية الأبعاد..... | 95 |

| | |
|--|-----|
| وظائف الفرز للمصفوفات - | 97 |
| sort() - فرز المصفوفة بترتيب تصاعدي | 97 |
| rsort() - فرز المصفوفة بترتيب تنازلي | 98 |
| asort() - فرز المصفوفة (ترتيب تصاعدي)، حسب القيمة | 99 |
| ksort() - فرز المصفوفة (ترتيب تصاعدي)، وفقاً للمفتاح | 100 |
| arsort() - فرز المصفوفة (ترتيب تنازلي)، وفقاً للقيمة | 101 |
| krsort() فرز المصفوفة (ترتيب تنازلي)، وفقاً للمفتاح | 101 |
| Superglobals - متغيرات العامة | 102 |
| \$GLOBALS | 103 |
| \$_SERVER | 104 |
| \$_REQUEST | 108 |
| | 110 |
| \$_POST[] | 110 |
| | 111 |
| \$_GET[] | 112 |
| ما هو التعبير العادي؟ | 113 |
| وظائف التعبير العادي | 114 |
| preg_match() باستخدام | 114 |
| preg_match_all() باستخدام | 115 |
| preg_replace() باستخدام | 116 |
| معدّلات التعبير العادي | 116 |
| أنماط التعبير العادية | 117 |
| الأحرف الأولية | 117 |
| محددو الكمية | 118 |
| التجميع | 118 |
| بسيط HTML نموذج - | 119 |
| الحصول على مقابل البريد | 122 |
| GET متى تستخدم | 123 |
| متى تستخدم البريد؟ | 124 |
| التحقق من صحة نموذج | 124 |
| حقول النص | 125 |
| أزرار الراديو للاختيار | 126 |
| عنصر النموذج | 126 |
| Form Security ملاحظة كبيرة حول | 127 |
| كيف تتجنب عمليات الاستغلال والهندسة العكسية | 129 |

| | |
|--|-----|
| <code>\$_SERVER[_SELF]</code> | 129 |
| التحقق من صحة اكواد النموذج باستخدام | 130 |
| - الحقول المطلوبة..... | 132 |
| - عرض رسائل الخطأ..... | 136 |
| - التحقق من صحة الاسم..... | 138 |
| - التحقق من صحة البريد الإلكتروني..... | 138 |
| - URL التحقق من صحة عنوان..... | 139 |
| - URL التحقق من صحة الاسم والبريد الإلكتروني وعنوان..... | 140 |
| - احتفظ بالقيم في النموذج..... | 142 |
| التاريخ والوقت..... | 144 |
| Date() وظيفة..... | 144 |
| الحصول على الوقت..... | 145 |
| نصيحة..... | 146 |
| احصل على الوقت..... | 146 |
| احصل على منطقتك الزمنية..... | 147 |
| mktime() إنشاء وقت مع..... | 148 |
| strtotime() إنشاء تاريخ من نصوص باستخدام..... | 149 |
| المزيد من أمثلة التاريخ..... | 150 |
| تتضمن الاكواد..... | 152 |
| أمثلة..... | 153 |
| require..... | 156 |
| التعامل مع الملفات..... | 158 |
| وظيفة ملف القراءة..... | 159 |
| fopen() - ملف مفتوح..... | 160 |
| fread() - ملف قراءة..... | 161 |
| fclose() - ملف إغلاق..... | 162 |
| fgets () - قراءة سطر واحد..... | 162 |
| feof() - التحقق من نهاية الملف..... | 163 |
| fgetc() - قراءة حرف واحد..... | 164 |
| fopen () - إنشاء ملف..... | 165 |
| أذونات الملف..... | 166 |
| fwrite() - الكتابة إلى الملف..... | 166 |
| الكتابة فوق الملف..... | 167 |
| إضافة النص الى الملف..... | 168 |
| تحميل ملف..... | 169 |

| | |
|---|-----|
| ini.....التعديل على ملف التكوين الاساسى | 169 |
| HTML.....إنشاء نموذج | 170 |
| الكود لتحميل الملف | 171 |
| تحقق مما إذا كان الملف موجودًا بالفعل | 172 |
| الحد من حجم الملف | 173 |
| الحد من نوع الملف | 173 |
| استكمال تحميل الملف | 174 |
| ما هو ملف تعريف الارتباط؟ | 176 |
| إنشاء ملفات تعريف الارتباط باستخدام | 177 |
| إنشاء/استرداد ملف تعريف الارتباط | 177 |
| تعديل قيمة ملف تعريف الارتباط | 179 |
| حذف ملف تعريف الارتباط | 180 |
| تحقق مما إذا تم تمكين ملفات تعريف الارتباط | 181 |
| ما هي جلسة ؟ | 182 |
| ابدأ جلسة | 183 |
| احصل على القيم المتغيرة لجلسة | 184 |
| تعديل متغير الجلسة | 186 |
| حذف جلسة | 187 |
| الفلتر | 188 |
| لماذا استخدام الفلاتر؟ | 189 |
| دالة filter_var() | 190 |
| تطهير نصوص | 190 |
| التحقق من صحة عدد صحيح | 191 |
| IP.....التحقق من صحة عنوان | 192 |
| تصحيح والتحقق من صحة عنوان البريد الإلكتروني | 193 |
| URL.....تصحيح والتحقق من صحة عنوان | 194 |
| التحقق من صحة عدد صحيح ضمن النطاق | 195 |
| IPv6.....التحقق من صحة عنوان | 195 |
| يجب أن يحتوي على نصوص الاستعلام - URL.....التحقق من صحة عنوان | 196 |
| ASCII > 127.....قم بإزالة الأحرف ذات قيمة | 197 |
| وظائف رد الاتصال | 198 |
| عمليات الاسترجاعات في الوظائف المحددة من قبل المستخدم | 199 |
| ما هو جسون؟ | 201 |
| الطرق لجسون | 201 |

| | |
|---|-----|
| - json_encode()..... | 201 |
| - json_decode()..... | 202 |
| - الوصول إلى القيم التي تم فك تشفيرها - | 203 |
| التكرار على جسون..... | 205 |
| ما هو الاستثناء؟..... | 206 |
| انشاء استثناء..... | 206 |
| كود المحاولة وجلب الاستثناء او الخطا..... | 208 |
| محاولة..... أخيرًا للكود | 209 |
| كائن الاستثناء..... | 211 |

مقدمة سريعة

بي اتش بي هي لغة برمجة نصية للخادم، وأداة قوية لإنشاء صفحات ويب ديناميكية وتفاعلية.

ولن اتحدث كثير عن هذه اللغة لأنها معروفة جيدا و
يكفيك ان تعرف ان اكبر مواقع التواصل الاجتماعي
العملقة فى العالم مصنوعة بـ (بي اتش بي) حيث
تقوم بي اتش بي بادارتها بكل سهولة ويسر وهى مواقع
شديدة التعقيد مما يدل على قوة هذه اللغة الجبلة
وسيطرتها الكاملة فى مجال الويب ويجب ان تعرف

ايضا ان منافسة بي اتش بي فى مجال الويب امر
صعب ومعقد للغاية حتى التقنيات العالمية التى
تفوقت على كل اللغات فى الاداء والسرعة مثل نود
جى اس لم تستطع نزع التاج من بي اتش بي من سنين
طويلة حيث مزالت بي اتش بي تسيطر على كل
انظمة السي ام اس الناجحة فى العالم حتى الان بنسبة
مئة بالمئة حتى لغة بايثون التى وصفها البعض
بالحصان الاسود ومعنى الحصان الاسود فى الثقافة
الغربية يعتقدون ان الحصان الاسود اسرع من الحصان
الابيض مما يدل هذا الاسم على سرعة بايثون فى التقدم
والانشار ورغم ذلك فشلت بايثون ايضا فى نزع التاج من
بي اتش بي و مزالت بي اتش بي تسيطر على 40
بالمئة من مواقع الانترنت بالعالم ناهيك عن السيطرة
الكاملة على انظمة سي ام اس مما يدل على فشل
المنافسات الشرسة المستمرة من سنين طويلة امام بي
اتش بي بلا اى جدوى او تحقيق اى نتائج تذكر حتى
الان ويوجد مميزات اخرى لهذه اللغة ستعرفها فى كتاب
المرجع العربى الكامل لدوال الورد بريس سيشرح هذا
المرجع مهام عشرات الالاف لدوال واجرائات الورد بريس
انشاء وترجمة الذكاء الاصطناعى

مثال على الكود

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<?php
```

```
echo "Abo Habib Alhosini PHP script!";
```

```
?>
```

```
</body>
```

```
</html>
```

• يتم تنفيذ كود بي اتش بي على الخادم فقط

قبل ان تبدأ

قبل المتابعة، يجب أن يكون لديك فهم أساسي لما يلي:

- لغة البرمجة
- سي اس اس
- جافا سكريبت
-

التركيب والتثبيت

قم بإعداد جهاز الكمبيوتر

قم بتنصيب سيرفر الاباتشي من الموقع الرسمي وسينصب لك كل احتياجات بي اتش بي دون ان تفعل شي وستجد ارشادات التثبيت على الموقع الرسمي للاباتشي

بناء الجمل الأساسي

النصي في أي مكان في المستند PHP يمكن وضع برنامج

:> وينتهي بـ **php?** بـ PHP يبدأ الكود

```
<?php
```

```
// PHP code goes here
```

```
?>
```

"_hosini.htm" هو PHP امتداد الملف الافتراضي لملفات

وبعض أكواد البرمجة النصية HTML عادةً على علامات PHP يحتوي ملف PHP.

يستخدم وظيفة PHP بسيط، مع برنامج نصي PHP أدناه، لدينا مثال لملف لإخراج النص على صفحة ويب "echo" مدمجة PHP

مثال

```
<!DOCTYPE html>
```

```
<html>
```



```
<body>
```

```
<h1>Abo Habib Alhosini PHP page</h1>
```

```
<?php  
echo "Abo Habib Alhosinii !";  
?>
```

```
</body>  
</html>
```

(;) بفاصلة منقوطة PHP ملاحظة: تنتهي عبارات

حساسية الحالة

while و **if**، **else** على سبيل المثال) الكلمات المحجوزة، **PHP** في الكلاسات، والوظائف، والوظائف المعرفة من قبل، (وما إلى ذلك، **echo** و المستخدم ليست حساسة لحالة الأحرف.

في المثال أدناه، جميع عبارات الاخراج الثلاثة أدناه متساوية وصحيحة:

مثال

```
<!DOCTYPE html>  
<html>  
<body>
```

```
<?php
ECHO "Abo Habib Alhosinii !<br>";
echo "Abo Habib Alhosinii !<br>";
EcHo "Abo Habib Alhosinii !<br>";
?>
```

```
</body>
</html>
```

ملاحظة: ومع ذلك؛ جميع أسماء المتغيرات حساسة لحالة الأحرف

وذلك لأنه **\$color** انظر للمثال ادناه؛ فقط العبارة الأولى ستعرض قيمة المتغير على أنها ثلاثة متغيرات **\$color**, **\$COLOR** و **\$coLOR** يتم التعامل مع مختلفة:

مثال

```
<!DOCTYPE html>
<html>
<body>
```

```
<?php
$color = "black";
echo "Habib Hoseseis " . $color . "<br>";
echo "Habib My house is " . $COLOR . "<br>";
echo "Habib My boat is " . $coLOR . "<br>";
?>
```

```
</body>
```

```
</html>
```

تعليقات

التعليقات في

هو سطر لا يتم تنفيذه كجزء من البرنامج. والغرض الوحيد PHP التعليق في كود منه هو أن يقرأه شخص ينظر إلى الكود.

يمكن استخدام التعليقات من أجل:

- دع الآخرين يفهمون الاكواد.
- ذكّر نفسك بما فعلته - لقد مر معظم المبرمجين بالعودة إلى عملهم بعد عام أو عامين واضطروا إلى إعادة اكتشاف ما فعلوه. يمكن أن تذكر التعليقات بما كنت تفكر فيه عندما كتبت الكود

عدة طرق للتعليق PHP تدعم لغة:

مثال

للتعليقات ذات السطر الواحد:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<?php
// This is a single-line comment
```

```
# This is also a single-line comment
?>
```

```
</body>
</html>
```

مثال

للتعليقات متعددة الأسطر:

```
<!DOCTYPE html>
<html>
<body>
```

```
<?php
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
?>
```

```
</body>
</html>
```

مثال

استخدام التعليقات لترك أجزاء من الكود:

```
<!DOCTYPE html>
<html>
<body>
```

```
<?php
// You can also use comments to leave out parts of a
code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>
```

```
</body>
</html>
```

متغيرات

المتغيرات هي "حاويات" لتخزين المعلومات.

إنشاء (الإعلان) عن متغيرات

يبدأ المتغير بالعلامة \$، متبوعة باسم المتغير، PHP في

مثال

```
<?php
$Hosini = "Abo Habib Alhosinii!";
$x = 5;
$y = 10.5;
?>
```

Abo Habib سيحتفظ المتغير بالقيمة **\$Hosini**، بعد تنفيذ العبارات أعلاه بالقيمة **\$y** بالقيمة **5**، وسيحتفظ المتغير **\$x** وسيحتفظ المتغير **! Alhosinii**، بالقيمة **10.5**.

ملاحظة: عندما تقوم بتعيين قيمة نصية لمتغير، ضع علامات الاقتباس حول القيمة.

أي أمر للإعلان PHP ملاحظة: على عكس لغات البرمجة الأخرى، ليس لدى عن متغير. يتم إنشاؤه في اللحظة التي تقوم فيها بتعيين قيمة له لأول مرة.

فكر في المتغيرات كحاويات لتخزين البيانات.

متغيرات

أو اسم أكثر وصفًا (العمر، اسم (لا و x مثل) يمكن أن يكون للمتغير اسم قصير (السيارة، الحجم الإجمالي).

PHP: قواعد متغيرات

- يبدأ المتغير بالعلامة \$، متبوعة باسم المتغير
- يجب أن يبدأ اسم المتغير بحرف أو بشرطة سفلية
- لا يمكن أن يبدأ اسم المتغير برقم
- يمكن أن يحتوي اسم المتغير فقط على أحرف أبجدية رقمية وشروط (_ و 0-9، Az) سفلية
- \$Hosiny وهي \$Hosiny (أسماء المتغيرات حساسة لحالة الأحرف (متغيرين مختلفين

!حساسة لحالة الأحرف PHP تذكر أن أسماء متغيرات

متغيرات الإخراج

لإخراج البيانات إلى الشاشة **PHP echo** غالبًا ما يتم استخدام عبارة: سيوضح المثال التالي كيفية إخراج النص والمتغير

مثال

```
<?php
$Hosini = "Abo Habib AlHosini.com";
echo "I love $Hosini! 🇸🇩🇸🇩";
?>
```


المثال التالي سوف ينتج نفس الناتج كما في المثال أعلاه

مثال

```
<?php
$Hosini = "Abo Habib AlHosini 🇸🇩.com ";
echo "I love " . $Hosini . "!";
?>
```

المثال التالي سيخرج مجموع متغيرين

مثال

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

البيان وكيفية إخراج البيانات إلى **echo** ملحوظة: سوف تتعلم المزيد عن الشاشة في الفصل التالي.

هي لغة مكتوبة بشكل واسع

بنوع البيانات الذي PHP في المثال أعلاه، لاحظ أنه لم يكن علينا إخبار .ينتمي إليه المتغير

تلقائيًا بربط نوع البيانات بالمتغير، اعتمادًا على قيمته. نظرًا لعدم PHP تقوم تعيين أنواع البيانات بالمعنى الدقيق للكلمة، يمكنك القيام بأشياء مثل إضافة نصوص إلى عدد صحيح دون التسبب في خطأ.

تمت إضافة إعلانات النوع. للمتغير وهذا يعطي خيارًا لتحديد ، PHP 7 في نوع البيانات المتوقع عند الإعلان عن دالة، ومن خلال تمكين المتطلبات الصلابة، سيتم إلقاء "خطأ فادح" على عدم تطابق النوع.

وإعلانات نوع البيانات **-strict**

نطاق متغيرات

. يمكن الإعلان عن المتغيرات في أي مكان في الكود ، PHP في

.نطاق المتغير هو جزء من الكود حيث يمكن الرجوع/استخدام المتغير

:ثلاثة نطاقات متغيرة مختلفة PHP لدى

- محلي
- عام
- ثابت

النطاق العام والمحلي

المتغير المعلن خارج الدالة له نطاق عام ولا يمكن الوصول إليه إلا خارج الدالة:

مثال

متغير ذو نطاق عام:

```
<?php
```

```
$x = 5; // global scope
```

```
function HabeB() {
```

```
// using x inside this function will generate an error
```

```
echo "<p>Variable x inside function is: $x</p>";
```

```
}
```

```
HabeB();
```

```
echo "<p>Variable x outside function is: $x</p>";
```

```
?>
```

المتغير المعلن داخل دالة له نطاق محلي ولا يمكن الوصول إليه إلا من خلال تلك الوظيفة:

مثال

متغير مع النطاق المحلي:

```
<?php
function HabeB() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
HabeB();
```

```
// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

يمكن أن يكون لديك متغيرات محلية بنفس الاسم في وظائف مختلفة، لأنه يتم التعرف على المتغيرات المحلية فقط من خلال الوظيفة التي تم الإعلان عنها.

متغير عام من داخل الدالة

المحجوزة للوصول إلى متغير عام من داخل **global** يتم استخدام الكلمة **global**.

الكلمة المحجوزة قبل المتغيرات (داخل الدالة) **global** للقيام بذلك، استخدم

مثال

```
<?php
$x = 5;
$y = 10;
```

```
function HabeB() {  
    global $x, $y;  
    $y = $x + $y;  
}
```

```
HabeB();  
echo $y; // outputs 15  
?>
```

أيضًا بتخزين كافة المتغيرات العامة في مصفوفة PHP تقوم تسمى . يحمل اسم المتغير. يمكن الوصول إلى هذه المصفوفة أيضًا من داخل الوظائف ويمكن استخدامها لتحديث المتغيرات العامة مباشرةً.

`$GLOBALS[index]index`

يمكن إعادة كتابة المثال أعلاه على النحو التالي:

مثال

```
<?php  
$x = 5;  
$y = 10;
```

```
function HabeB() {  
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];  
}
```

```
HabeB();
```

```
echo $y; // outputs 15
```

```
?>
```

الكلمة المحجوزة الثابتة

عادةً، عند اكتمال/تنفيذ دالة، يتم حذف جميع متغيراتها. ومع ذلك، في بعض الأحيان نريد عدم حذف متغير محلي. نحن في حاجة إليها لمزيد من العمل الكلمة المحجوزة عند الإعلان عن المتغير لأول **static** للقيام بذلك، استخدم مرة:

مثال

```
<?php
function HabeB() {
    static $x = 0;
    echo $x;
    $x++;
}
```

```
HabeB();
HabeB();
HabeB();
?>
```

بعد ذلك، في كل مرة يتم فيها استدعاء الوظيفة، سيظل هذا المتغير يحتوي على المعلومات التي يحتوي عليها من آخر مرة تم فيها استدعاء الوظيفة. ملاحظة: لا يزال المتغير محليًا للدالة.

إخراج وطباعة البيانات

print و **echo**: هناك طريقتان أساسيتان للحصول على الإخراج، **PHP** مع **print** أو **echo**، يحتوي هذا الكتاب نستخدم الفصل على مزيد من المعلومات حول هذين البيانيين الناتجين.

إخراج وطباعة الكواد

أكثر أو أقل نفس الشيء. كلاهما يستخدم لإخراج البيانات إلى **print** وهم **echo** الشاشة.

لها قيمة إرجاع 1 لذا **print** ليس لها قيمة إرجاع بينما **echo**: الاختلافات صغيرة يمكن أن يأخذ معلمات متعددة (على **echo**. يمكن استخدامها في التعبيرات **echo**. يمكن أن يأخذ وسيطة واحدة **print** الرغم من ندرة هذا الاستخدام) بينما **print**. أسرع بشكل هامشي من

كود إخراج

echo() أو **echo**: يمكن استخدام العبارة مع أو بدون أقواس **echo**.

عرض النص

لاحظ أن النص (الأمر **echo** يوضح المثال التالي كيفية إخراج النص باستخدام HTML):

مثال

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Abo Habib Alhosinii !<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple
parameters.";
?>
```

عرض المتغيرات

echo : يوضح المثال التالي كيفية إخراج النص والمتغيرات باستخدام العبارة

مثال

```
<?php
$Hosini1 = "Learn PHP";
$Hosini2 = "Abo Habib AlHosini.com";
$x = 5;
$y = 4;

echo "<h2>" . $Hosini1 . "</h2>";
echo "Study PHP at " . $Hosini2 . "<br>";
```

```
echo $x + $y;  
?>
```

امر الإخراج والعرض

print() أو **print**: يمكن استخدام العبارة مع أو بدون أقواس **print**.

عرض النص

لاحظ أن النص (الأمر **print** يوضح المثال التالي كيفية إخراج النص باستخدام HTML): يمكن أن يحتوي على علامة

مثال

```
<?php  
print "<h2>PHP is Fun!</h2>";  
print "Abo Habib Alhosinii !<br>";  
print "I'm about to learn PHP!";  
?>
```

عرض المتغيرات

print: يوضح المثال التالي كيفية إخراج النص والمتغيرات باستخدام العبارة

مثال

```
<?php  
$Hosini1 = "Learn PHP";
```

```
$Hosini2 = "Abo Habib AlHosini.com";
```

```
$x = 5;
```

```
$y = 4;
```

```
print "<h2>" . $Hosini1 . "</h2>";
```

```
print "Study PHP at " . $Hosini2 . "<br>";
```

```
print $x + $y;
```

```
?>
```

أنواع البيانات

أنواع الإكواد

يمكن للمتغيرات تخزين بيانات من أنواع مختلفة، ويمكن لأنواع البيانات المختلفة القيام بأشياء مختلفة.

يدعم أنواع البيانات التالية PHP:

- نص
- عدد صحيح
- عشري (أرقام الفاصلة العشرية - تسمى أيضًا مزدوجة)
- منطقية
- مجموعة او مصفوفة
- خطأ

نصوص

"Abo Habib Alhosinii!". النصوص عبارة عن نصوص من الأحرف، مثل يمكن أن تكون النصوص أي نص داخل علامات الاقتباس. يمكنك استخدام علامات الاقتباس المفردة أو المزدوجة:

مثال

```
<?php
```

```
$x = "Abo Habib Alhosinii !";
```

```
$y = 'Abo Habib Alhosinii !';
```

```
echo $x;
```

```
echo "<br>";
```

```
echo $y;
```

```
?>
```

عدد صحيح

نوع البيانات الصحيح هو رقم غير عشري يقع بين -2,147,483,648 و 2,147,483,647.

قواعد الأعداد الصحيحة:

- يجب أن يحتوي العدد الصحيح على رقم واحد على الأقل
- يجب ألا يحتوي العدد الصحيح على نقطة عشرية
- يمكن أن يكون العدد الصحيح موجباً أو سالباً

- يمكن تحديد الأعداد الصحيحة في: نظام عشري (أساس 10)، سداسي عشري (أساس 16)، ثماني (أساس 8)، أو ثنائي (أساس 2)

بإرجاع `PHP var_dump()` هو عدد صحيح. تقوم الدالة `$x` في المثال التالي
بنوع البيانات وقيمتها:

مثال

```
<?php  
$x = 5985;  
var_dump($x);  
?>
```

الرقم العشري

العدد العشري (رقم النقطة العشرية) هو رقم بفاصلة عشرية أو رقم في شكل
أسّي.

بإرجاع `PHP var_dump()` هو عدد عشري. تقوم الدالة `$x` في المثال التالي
بنوع البيانات وقيمتها:

مثال

```
<?php  
$x = 10.365;  
var_dump($x);  
?>
```

منطقية

FALSE أو TRUE: تمثل القيمة المنطقية حالتين محتملتين

```
$x = true;
```

```
$y = false;
```

غالبًا ما تستخدم القيم المنطقية في الاختبارات الشرطية. سوف تتعلم المزيد عن الاختبار المشروط في فصل لاحق من هذا الكتاب.

المصفوفة

تقوم المصفوفة بتخزين قيم متعددة في متغير واحد.

PHP عبارة عن مصفوفة. تقوم الدالة `$habib_Array` في المثال التالي بإرجاع نوع البيانات وقيمتها `var_dump()`:

مثال

```
<?php
$habib_Array = array("Abo Habib *","Al Hosini *","Hamz
*");
var_dump($habib_Array);
?>
```

سوف تتعلم الكثير عن المصفوفات في الفصول اللاحقة من هذا الكتاب.

كائن

الكلاسات والكائنات هما الجانبان الرئيسيان للبرمجة الموجهة للكائنات.

الكلاس هي قالب للكائنات، والكائن هو مثيل للكلاس.

عندما يتم إنشاء الكائنات الفردية، فإنها ترث كافة الخصائص والسلوكيات من الكلاس، ولكن سيكون لكل كائن قيم مختلفة للخصائص.

يمكن أن تحتوي السيرة على **Hosese** لنفترض أن لدينا كلاس تسمى خصائص مثل الطراز واللون وما إلى ذلك. يمكننا تحديد متغيرات مثل **\$namme** و **\$color** هذه الخصائص

عندما يتم إنشاء الكائنات الفردية (فولفو، بي إم دبليو، تويوتا، وما إلى ذلك)، فإنها ترث جميع الخصائص والسلوكيات من الكلاس، ولكن سيكون لكل كائن قيم مختلفة للخصائص.

باستدعاء هذه **PHP** فسوف تقوم **__construct()** إذا قمت بإنشاء دالة الوظيفة تلقائيًا عندما تقوم بإنشاء كائن من كلاس.

مثال

```
<?php
class Hosese{
    public $color;
    public $namme;
    public function __construct($color, $namme) {
        $this->color = $color;
        $this->namme = $namme;
    }
}
```



```

}
public function message() {
    return "Habib Hoseseis a " . $this->color . " " . $this-
    >namme . "!";
}
}

```

```

$Hose3= new Hosese("red", "Habib *");
echo $Hosese3-> message();
echo "<br>";
$Hose3= new Hosese("black", "Hamz *");
echo $Hosese3-> message();
?>

```

القيمة الفارغة

NULL: هو نوع بيانات خاص يمكن أن يكون له قيمة واحدة فقط **Null**.
هو متغير ليس له قيمة مخصصة له **NULL** المتغير من نوع البيانات
له تلقائياً **NULL** نصيحة: إذا تم إنشاء متغير بدون قيمة، فسيتم تعيين قيمة
NULL: يمكن أيضاً إفراغ المتغيرات عن طريق تعيين القيمة إلى

مثال

```

<?php
$x = "Abo Habib Alhosinii !";

```

```
$x = null;  
var_dump($x);  
?>
```

الموارد

نوع المورد الخاص ليس نوع بيانات فعليًا. إنه تخزين مرجع للوظائف والموارد الخارجية لـ PHP.

أحد الأمثلة الشائعة لاستخدام نوع بيانات المورد هو استدعاء قاعدة البيانات. لن نتحدث عن نوع المصدر هنا، لأنه موضوع متقدم.

”النصوص عبارة عن نصوص من الأحرف، مثل ”ابو حبيب“.

وظائف نصوص

في هذا الفصل، سنلقي نظرة على بعض الوظائف الشائعة الاستخدام للتعامل مع النصوص.

إرجاع طول النصوص - strlen()

بإرجاع طول النصوص `strlen()` PHP تقوم الدالة

مثال

"Abo Habib Alhosinii !": إرجاع طول النصوص

```
<?php
echo strlen("Abo Habib Alhosinii !"); // outputs 12
?>
```

حساب الكلمات في - str_word_count() النصوص

بحساب عدد الكلمات في النصوص `str_word_count()` PHP تقوم وظيفة

مثال

"Abo Habib Alhosinii !": احسب عدد الكلمات في النصوص

```
<?php
echo str_word_count("Abo Habib Alhosinii !"); //
outputs 2
?>
```

عكس نصوص - strrev()

بعكس نصوص PHP **strrev()** تقوم وظيفة

مثال

:"!عكس الحروف لـ"ابو حبيب الحسيني

```
<?php
echo strrev("Abo Habib Alhosinii !"); //iinisoHlA bibaH
obA
?>
```

البحث عن نص داخل نصوص - strpos()

عن نص محدد ضمن نصوص. إذا تم العثور PHP **strpos()** تبحث وظيفة على تطابق، تقوم الدالة بإرجاع موضع الحرف للمطابقة الأولى. إذا لم يتم **FALSE** العثور على تطابق، فسيتم إرجاع

مثال

:"!Abo Habib Alhosinii" في النصوص "Habib" ابحث عن النص

```
<?php
echo strpos("Abo Habib Alhosinii !", "Habib");//
outputs 4
?>
```

نصيحة: موضع الحرف الأول في النصوص هو 0 (وليس 1)

استبدال النص داخل نصوص - str_replace()

باستبدال بعض الأحرف ببعض الأحرف PHP **str_replace()** تقوم وظيفة الأخرى في النصوص.

مثال

:"استبدال نص "حبيب" بكلمة "على"

```
<?php
echo str_replace("Habib", "Alee", "Abo Habib
Alhosinii !");// outputs Hello Abo Alee!
?>
```

أكمل مرجع النصوص

!على وصف ومثال للاستخدام لكل وظيفة PHP يحتوي مرجع نصوص

في هذا الفصل سوف ننظر بعمق في الأعداد الصحيحة، والأعداد العشرية، ونصوص الأرقام

الأرقام

هو أنه يوفر تحويلًا تلقائيًا لنوع البيانات PHP شيء واحد يجب ملاحظته حول ذلك، إذا قمت بتعيين قيمة عددية لمتغير، فسيكون نوع هذا المتغير عددًا صحيحًا تلقائيًا. وبعد ذلك، إذا قمت بتعيين نصوص إلى نفس المتغير، فسيتم تغيير النوع إلى نصوص. قد يؤدي هذا التحويل التلقائي في بعض الأحيان إلى كسر الاكواد

الأعداد الصحيحة

كلها أعداد صحيحة 2، 256، -256، 10358، -179567

العدد الصحيح هو رقم بدون أي جزء عشري

نوع البيانات الصحيح هو رقم غير عشري بين -2147483648 و

2147483647 في أنظمة 32 بت، وبين -9223372036854775808 و

9223372036854775807 في أنظمة 64 بت. سيتم تخزين القيمة الأكبر

(أو الأقل) من هذا على أنها عشرية، لأنها تتجاوز حد العدد الصحيح

ملحوظة: هناك شيء آخر مهم يجب معرفته وهو أنه حتى لو كانت $4 * 2.5$ هي 10، فسيتم تخزين النتيجة كعشري، لأن أحد المعاملات هو عشري (2.5).

فيما يلي بعض القواعد للأعداد الصحيحة:

- يجب أن يحتوي العدد الصحيح على رقم واحد على الأقل
- يجب ألا يحتوي العدد الصحيح على نقطة عشرية
- يمكن أن يكون العدد الصحيح موجبًا أو سالبًا
- يمكن تحديد الأعداد الصحيحة بثلاثة تنسيقات: عشري (مبني على أو ثماني (X معتمد على 16 - مسبوق بـ 0) 10)، سداسي عشري (معتمد على 8 - مسبوق بـ 0)

على الثوابت المحددة مسبقًا التالية للأعداد الصحيحة PHP تحتوي:

- PHP_INT_MAX - أكبر عدد صحيح مدعوم
- PHP_INT_MIN - أصغر عدد صحيح مدعوم
- PHP_INT_SIZE - حجم العدد الصحيح بالبايت

الوظائف التالية للتحقق مما إذا كان نوع المتغير عددًا صحيحًا PHP لدى:

- is_int()
- is_integer() - الاسم المختصر لـ is_int()
- is_long() - الاسم المختصر لـ is_int()

مثال

تحقق مما إذا كان نوع المتغير عددًا صحيحًا:

```
<?php
$x = 5985;
var_dump(is_int($x));
```

```
$x = 59.85;  
var_dump(is_int($x));  
?>
```

الأرقام العشرية

العدد العشري هو رقم ذو علامة عشرية أو رقم في شكل أسّي

كلها عوامات 2.0، 256.4، 10.358، 7.64E+5، 5.56E-5

يمكن لنوع البيانات العشرية عادةً تخزين قيمة تصل إلى

وبدقة قصوى، (يعتمد على النظام الأساسي) 1.7976931348623E+308، تصل إلى 14 رقمًا

(PHP 7.2 من) على الثوابت المحددة مسبقًا التالية للعوامات PHP تحتوي

- PHP_FLOAT_MAX - أكبر رقم عشري يمكن تمثيله
- PHP_FLOAT_MIN - أصغر رقم فاصلة عشرية موجب يمكن تمثيله
- PHP_FLOAT_MAX - أصغر رقم فاصلة عشرية سالب يمكن تمثيله
- PHP_FLOAT_DIG - عدد الأرقام العشرية التي يمكن تقريبها إلى عدد عشري والعودة دون فقدان الدقة
- PHP_FLOAT_EPSILON - x أصغر رقم موجب يمكن تمثيله - $x + 1.0 \neq 1.0$ بحيث يكون

:الوظائف التالية للتحقق مما إذا كان نوع المتغير رقم عشري PHP لدى

- is_float()
- is_double() - الاسم المختصر لـ is_float()

مثال

تحقق مما إذا كان نوع المتغير رقم عشريا

```
<?php
$x = 10.365;
var_dump(is_float($x));
?>
```

إنفينيتي

تعتبر لا نهائية **PHP_FLOAT_MAX** القيمة الرقمية الأكبر من

الوظائف التالية للتحقق مما إذا كانت القيمة الرقمية محدودة أم لا **PHP** لدى نهائية:

بإرجاع نوع البيانات وقيمتها **PHP** في **var_dump()** ومع ذلك، تقوم الدالة

مثال

تحقق مما إذا كانت القيمة الرقمية محدودة أم لا نهائية

```
<?php
$x = 1.9e411;
```

```
var_dump($x);
```

```
?>
```

نان

NaN تعني ليس رقمًا NaN.

للمعاملات الرياضية المستحيلة NaN يستخدم.

الوظائف التالية للتحقق مما إذا كانت القيمة ليست رقمًا PHP لدى:

- `is_nan()`

بإرجاع نوع البيانات وقيمتها PHP في `var_dump()` ومع ذلك، تقوم الدالة

مثال

NaN: ستؤدي العملية الحسابية غير الصالحة إلى إرجاع قيمة

```
<?php
```

```
$x = acos(8);
```

```
var_dump($x);
```

```
?>
```

النصوص العددية

لمعرفة ما إذا كان المتغير رقميًا `PHP is_numeric()` يمكن استخدام الدالة أم لا. ترجع الدالة صحيحًا إذا كان المتغير رقمًا أو نصوص رقمية، وترجع خطأ إذا كان المتغير رقمًا أو نصوص رقمية.

مثال

تحقق مما إذا كان المتغير رقميًا:

```
<?php  
$x = 5985;  
var_dump(is_numeric($x));
```

```
$x = "5985";  
var_dump(is_numeric($x));
```

```
$x = "59.85" + 100;  
var_dump(is_numeric($x));
```

```
$x = "Hello";  
var_dump(is_numeric($x));
```

```
?>
```

`is_numeric()` `FALSE` ستُرجع الدالة: ملاحظة: بدءًا من الإصدار **PHP 7.0** حيث أنها (مثل `0xf4c3b00c` مثل `0`) للنصوص الرقمية ذات الشكل الست عشري لم تعد تُعتبر نصوص رقمية.

التحويل

في بعض الأحيان تحتاج إلى إرسال قيمة رقمية إلى نوع بيانات آخر لتحويل قيمة إلى `intval()` أو (عدد صحيح) أو `(int)` غالبًا ما تُستخدم الدالة عدد صحيح.

مثال

إرسال عشري ونصوص إلى عدد صحيح

```
<?php
// Cast float to int
$x = 23465.768;
$int_cast = (int)$x;
echo $int_cast;
```

```
echo "<br>";
```

```
// Cast string to int
$x = "23465.768";
$int_cast = (int)$x;
```

```
echo $int_cast;  
?>
```

الرياضيات

على مجموعة من الوظائف الرياضية التي تسمح لك بأداء PHP تحتوي المهام الرياضية على الأرقام.

pi() وظيفة

PI قيمة pi() ترجع الدالة

مثال

```
<?php  
echo(pi()); // returns 3.1415926535898  
?>
```

min() و max() وظائف

للعثور على أدنى أو أعلى قيمة min() and max() يمكن استخدام الدالات في قائمة الوسائط:

مثال

```
<?php
echo(min(0, 150, 30, 20, -8, -200)); // returns -200
echo(max(0, 150, 30, 20, -8, -200)); // returns 150
?>
```

abs() وظيفة

:القيمة المطلقة (الإيجابية) للرقم **abs()** ترجع الدالة

مثال

```
<?php
echo(abs(-6.7)); // returns 6.7
?>
```

sqrt() وظيفة

:الدالة بإرجاع الجذر التربيعي للرقم **sqrt()** تقوم

مثال

```
<?php  
echo(sqrt(64)); // returns 8  
?>
```

تقريب الأرقام ذو الفاصلة العشرية

:بتقريب رقم الفاصلة العشرية إلى أقرب عدد صحيح `round()` تقوم الدالة

مثال

```
<?php  
echo(round(0.60)); // returns 1  
echo(round(0.49)); // returns 0  
?>
```

أرقام عشوائية

:الدالة بإنشاء رقم عشوائي `rand()` تقوم

مثال

```
<?php  
echo(rand());  
?>
```

للحصول على مزيد من التحكم في الرقم العشوائي، يمكنك إضافة الاختيارية لتحديد أقل عدد صحيح وأعلى عدد صحيح `min` و `max` معلمات سيتم إرجاعه.

على سبيل المثال، إذا كنت تريد عددًا صحيحًا عشوائيًا بين 10 و 100 على سبيل المثال، إذا كنت تريد عددًا صحيحًا عشوائيًا بين 10 و 100 فاستخدم `rand(10, 100)` (ضمنًا)، فاستخدم

مثال

```
<?php  
echo(rand(10, 100));  
?>
```

استكمال مرجع الرياضيات

الرياضي على وصف ومثال للاستخدام لكل وظيفة PHP يحتوي مرجع

الثوابت مثل المتغيرات إلا أنه بمجرد تعريفها لا يمكن تغييرها أو عدم تعريفها

ثوابت

. الثابت هو معرف (اسم) لقيمة بسيطة. لا يمكن تغيير القيمة أثناء الكود
يبدأ الاسم الثابت الصالح بحرف أو شرطة سفلية (لا توجد علامة \$ قبل الاسم
(الثابت).

ملاحظة: على عكس المتغيرات، تكون الثوابت عمومية تلقائيًا عبر الكود
بأكمله.

إنشاء ثابت

الدالة **define()** لإنشاء ثابت، استخدم

define(name, value, case-insensitive)

حدود:

- الاسم : يحدد اسم الثابت

- القيمة : تحدد قيمة الثابت
- غير حساس لحالة الأحرف : يحدد ما إذا كان الاسم الثابت يجب أن يكون حساسًا لحالة الأحرف. الافتراضي خطأ

مثال

: قم بإنشاء ثابت باسم حساس لحالة الأحرف

```
<?php  
define("Hoseni", "Welcome to Abo Habib  
AlHosini.com!");  
echo Hoseni;  
?>
```

مثال

: قم بإنشاء ثابت باسم غير حساس لحالة الأحرف

```
<?php  
define("Hoseni", "Welcome to Abo Habib  
AlHosini.com!", true);  
echo Hoseni;  
?>
```

المصفوفات الثابتة

`define()` يمكنك إنشاء ثابت مصفوفة باستخدام الوظيفة، PHP7 في

مثال

إنشاء ثابت مصفوفة:

```
<?php
define("Hoseses", [
    "Alfa Romeo",
    "Al Hosini *",
    "Hamz *"
]);
echo Hoseses[0];
?>
```

الثوابت عامة

تكون الثوابت عمومية تلقائيًا ويمكن استخدامها عبر الكود بأكمله

مثال

يستخدم هذا المثال ثابتًا داخل الدالة، حتى لو تم تعريفه خارج الدالة

```
<?php  
define("Hoseni", "Welcome to Abo Habib  
AlHosini.com!");
```

```
function HabeB() {  
    echo Hoseni;  
}
```

```
HabeB();  
>
```

المشغلات

يتم استخدام العوامل لإجراء العمليات على المتغيرات والقيم

:العوامل إلى المجموعات التالية PHP تقسم

- العمليات الحسابية
- مشغلي التعيين
- عوامل المقارنة
- عوامل الزيادة/الإنقاص
- العوامل المنطقية
- مشغلي النصوص
- مشغلي المصفوفة
- عوامل الاحالة المشروطة

مشغلي الحساب

مع القيم الرقمية لإجراء PHP يتم استخدام عوامل التشغيل الحسابية في العمليات الحسابية الشائعة، مثل الجمع والطرح والضرب وما إلى ذلك.

| Operator | اسم | Example | نتيجة |
|----------|----------------|--------------|---|
| + | إضافة | $\$x + \y | $\$x$ و $\$y$ مجموع |
| - | الطرح عملية | $\$x - \y | $\$x$ و $\$y$ الفرق بين |
| * | الضرب | $\$x * \y | $\$x$ و $\$y$ منتج |
| / | قسم | $\$x / \y | $\$x$ و $\$y$ حاصل قسمة |
| % | معامل | $\$x \% \y | $\$y$ مقسومًا على $\$x$ باقياً إلى قوة $\$x$ نتيجة رفع |
| ** | الأس | $\$x ** \y | $\$y$ 'th |

مشغلي تعيين

مع القيم الرقمية لكتابة قيمة إلى متغير PHP يتم استخدام عوامل تخصيص هو "=" وهذا يعني أنه يتم تعيين المعامل PHP عامل التعيين الأساسي في الأيسر على قيمة تعبير المهمة الموجود على اليمين.

| Assignment | Same as... | وصف |
|------------|-------------|--|
| $x = y$ | $x = y$ | يتم تعيين المعامل الأيسر على قيمة التعبير الموجود على اليمين |
| $x += y$ | $x = x + y$ | إضافة |
| $x -= y$ | $x = x - y$ | الطرح |
| $x *= y$ | $x = x * y$ | عملية الضرب |

| | | |
|------------|--------------|-------|
| $x /= y$ | $x = x / y$ | قسم |
| $x \% = y$ | $x = x \% y$ | معامل |

مشغلي المقارنة

لمقارنة قيمتين (رقم أو نصوص) PHP تُستخدم عوامل المقارنة

| Operator | Name | Example | Result |
|-----------------------|---------------|-------------------------------|---|
| <code>==</code> | Equal | <code>\$x == \$y</code> | Returns true if \$x is equal to \$y |
| <code>===</code> | Identical | <code>\$x === \$y</code> | Returns true if \$x is equal to \$y, and they are of the same type |
| <code>!=</code> | Not equal | <code>\$x != \$y</code> | Returns true if \$x is not equal to \$y |
| <code><></code> | Not equal | <code>\$x <> \$y</code> | Returns true if \$x is not equal to \$y |
| <code>!==</code> | Not identical | <code>\$x !== \$y</code> | Returns true if \$x is not equal to \$y, or they are not of the same type |
| <code>></code> | Greater than | <code>\$x > \$y</code> | Returns true if \$x is greater than \$y |
| <code><</code> | Less than | <code>\$x < \$y</code> | Returns true if \$x |

| | | | |
|-----|--------------------------|-------------|---|
| | | | is less than \$y |
| >= | Greater than or equal to | \$x >= \$y | Returns true if \$x is greater than or equal to \$y |
| <= | Less than or equal to | \$x <= \$y | Returns true if \$x is less than or equal to \$y |
| <=> | Spaceship | \$x <=> \$y | Returns an integer less than, equal to, or greater than zero, depending on if \$x is less than, equal to, or greater than \$y. Introduced in PHP 7. |

زيادة / إنقاص عوامل التشغيل

لزيادة قيمة المتغير PHP يتم استخدام عوامل زيادة

لتقليل قيمة المتغير PHP يتم استخدام عوامل إنقاص

| Operator | اسم | وصف |
|----------|-----------------|------------------------------------|
| ++\$x | الزيادة المسبقة | \$x بمقدار واحد، ثم يُرجع \$x يزيد |
| \$x++ | بعد الزيادة | بمقدار واحد \$x ثم يزيد، \$x يُرجع |

| | | |
|-------|-------------|-------------------------------------|
| --\$x | إنقاص مسبق | \$x بمقدار واحد، ثم إرجاع \$x إنقاص |
| \$x-- | بعد النقصان | بمقدار واحد \$x ثم تُنقص، \$x تُرجع |

العوامل المنطقية

المنطقية لدمج العبارات الشرطية PHP يتم استخدام عوامل

| Operator | اسم | Example | نتيجة |
|----------|-----|-------------|---|
| and | و | \$x and \$y | صحيحين \$x و \$y إذا كان كل من |
| or | أو | \$x or \$y | صحيحًا \$y أو \$x صحيح إذا كان |
| xor | او | \$x xor \$y | صحيحًا، ولكن ليس كليهما \$y أو \$x صحيح إذا كان |
| && | و | \$x && \$y | صحيحين \$x و \$y إذا كان كل من |
| | أو | \$x \$y | صحيحًا \$y أو \$x صحيح إذا كان |
| ! | لا | !\$x | غير صحيح \$x صحيح إذا كان |

مشغلي نصوص

عاملين مصممين خصيصًا للنصوص PHP لدى

| Operator | Name | Example | نتيجة |
|----------|-----------------------------|---------------------------|-----------------------------------|
| . | Concatenation | \$Hosini1 . \$Hosini2 | \$Hosini1 تجميع و \$Hosini2 |
| .= | Concatenation assignment | \$Hosini1 .= \$Hosini2 | بـ \$Hosini2 اضافة \$Hosini1 |

مشغلي مصفوفة

لمقارنة المصفوفات PHP يتم استخدام عوامل تشغيل المصفوفات

| Operator | Name | Example | نتيجة |
|----------|--------------|---------------|---|
| + | Union | $\$x + \y | $\$x$ و $\$y$ اتحاد |
| == | Equality | $\$x == \y | لهما $\$x$ و $\$y$ يُرجع صحيحًا إذا كان نفس أزواج المفتاح/القيمة |
| === | Identity | $\$x === \y | لهما $\$x$ و $\$y$ يُرجع صحيحًا إذا كان نفس أزواج المفاتيح/القيمة وبنفس الترتيب ومن نفس الأنواع |
| != | Inequality | $\$x != \y | لا يساوي $\$x$ يُرجع صحيحًا إذا كان $\$y$ |
| <> | Inequality | $\$x <> \y | لا يساوي $\$x$ يُرجع صحيحًا إذا كان $\$y$ |
| !== | Non-identity | $\$x !== \y | غير $\$x$ يُرجع صحيحًا إذا كان $\$y$ مطابق لـ |

مشغلي التعيين الشرطي

:لتعيين قيمة وفقًا للشروط PHP تُستخدم عوامل التعيين الشرطية في

تُستخدم العبارات الشرطية لتنفيذ إجراءات مختلفة بناءً على شروط مختلفة

الجمل الشرطية

في كثير من الأحيان، عند كتابة الاكواد، تريد تنفيذ إجراءات مختلفة لظروف مختلفة. يمكنك استخدام العبارات الشرطية في الاكواد للقيام بذلك.

لدينا العبارات الشرطية التالية في PHP في:

- البيان - ينفذ بعض الاكواد إذا كان أحد الشروط صحيحًا **if**
- البيان - ينفذ بعض الاكواد إذا كان الشرط صحيحًا ورمزًا آخر إذا **if...else** كان الشرط خاطئًا
- البيان - ينفذ اكوادا مختلفة لأكثر من شرطين **if...elseif...else**
- البيان - يختار واحدة من مجموعات الاكواد العديدة التي سيتم تنفيذها **switch**

if كود

بعض الاكواد إذا كان شرط واحد صحيحًا **if** ينفذ البيان.

```
if (condition) {
```

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

```
}
```

مثال

أقل من 20 (HOUR) الإخراج "أتمنى لك يوماً سعيداً!" إذا كان الوقت الحالي

```
<?php
```

```
$t = date("H");
```

```
if ($t < "20") {
```

```
    echo "Have a good day!";
```

```
}
```

```
?>
```

if...else كود -

بعض الاكواد إذا كان الشرط صحيحاً ورمزاً آخر إذا كان **if...else** ينفذ البيان الشرط خاطئاً.

مثال

الإخراج "أتمنى لك يوماً سعيداً!" إذا كان الوقت الحالي أقل من 20، و"أتمنى لك ليلة سعيدة!" خلاف ذلك

```
<?php
```

```
$t = date("H");
```

```
if ($t < "20") {
```

```
    echo "Have a good day!";
```

```
} else {
```

```
    echo "Have a good night!";
```

```
}
```

```
?>
```

if...elseif...else كود -

اكوادا مختلفة لأكثر من شرطين **if...elseif...else** ينفذ البيان

مثال

الإخراج "أتمنى لك صباح الخير!" إذا كان الوقت الحالي أقل من 10، و"أتمنى لك يومًا سعيدًا!" إذا كان الوقت الحالي أقل من 20. وإلا فسيتم إخراج "أتمنى لك ليلة سعيدة":

```
<?php  
$t = date("H");
```

```
if ($t < "10") {  
    echo "Have a good morning!";  
} elseif ($t < "20") {  
    echo "Have a good day!";  
} else {  
    echo "Have a good night!";  
}  
?>
```

لتنفيذ إجراءات مختلفة بناءً على شروط مختلفة **switch** يتم استخدام العبارة

كود التبديل

العبارة لتحديد إحدى مجموعات الاكواد العديدة التي سيتم **switch** استخدم تنفيذها .

(غالبًا ما يكون متغيرًا) n هذه هي الطريقة التي تعمل بها: أولاً لدينا تعبير واحد ويتم تقييمه مرة واحدة. ثم تتم مقارنة قيمة التعبير مع القيم ل حالة في البنية. إذا كان هناك تطابق، فسيتم تنفيذ كتلة الاكواد المرتبطة بهذه **default**. لمنع تشغيل الكود في الحالة التالية تلقائيًا **break** الحالة. يُستخدم . يتم استخدام العبارة إذا لم يتم العثور على تطابق

مثال

```
<?php
$Hosini_user = "black";

switch ($Hosini_user) {
  case "black":
    echo "Your favorite color is black!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "black":
    echo "Your favorite color is black!";
    break;
  default:
    echo "Your favorite color is neither black, blue, nor
black!";
}
?>
```

PHP ستتعلم في الفصول التالية كيفية تكرار الاكواد باستخدام الحلقات في

حلقات

في كثير من الأحيان، عندما اكواد برمجية، فأنت تريد تشغيل نفس كتلة الاكواد مرارًا وتكرارًا لعدد معين من المرات. لذا، بدلاً من إضافة عدة أسطر تعليمات برمجية متساوية تقريبًا في الكود، يمكننا استخدام الحلقات

تُستخدم الحلقات لتنفيذ نفس كتلة الاكواد مرارًا وتكرارًا، طالما كان شرط معين صحيحًا.

لدينا أنواع الحلقات التالية، PHP في

- **while** - حلقات من خلال كتلة من الاكواد طالما أن الشرط المحدد صحيح
- **do...while** - يتكرر عبر كتلة من الاكواد مرة واحدة، ثم يكرر الحلقة طالما كان الشرط المحدد صحيحًا
- **for** - حلقات من خلال كتلة من الاكواد لعدد محدد من المرات
- **foreach** - حلقات من خلال كتلة من الاكواد لكل عنصر في مصفوفة

سوف تشرح الفصول التالية وتعطي أمثلة لكل نوع من أنواع الحلقات

حلقات من خلال كتلة من الاكواد طالما أن الشرط المحدد **while** الحلقة صحيح.

العمل أثناء الحلقة

كتلة من الاكواد تنفذ طالما أن الشرط المحدد صحيح **while** تنفذ الحلقة

أمثلة

يعرض المثال أدناه الأرقام من 1 إلى 5

مثال

```
<?php
```

```
$x = 1;
```

```
while($x <= 5) {
```

```
    echo "The number is: $x <br>";
```

```
    $x++;
```

```
}
```

```
?>
```


شاهد المثال

- رقم بتعيين قيمة البداية على $(\$x)$ س $= 1$; - قم بتهيئة عداد الحلقة $\$x$.
- أقل من أو يساوي $5 \$x$ تابع الحلقة طالما أن $5 \leq \$x$.
- زيادة قيمة عداد الحلقة بمقدار 1 لكل تكرار - $\$x++$.

هذا المثال يصل إلى 100 في العشرات

مثال

```
<?php
```

```
$x = 0;
```

```
while($x <= 100) {
```

```
    echo "The number is: $x <br>";
```

```
    $x+=10;
```

```
}
```

```
?>
```

شاهد المثال

- رقم بتعيين قيمة البداية على $(\$x)$ س $= 0$; - قم بتهيئة عداد الحلقة $\$x$.
- أقل من أو يساوي $100 \$x$ تابع الحلقة طالما أن $100 \leq \$x$.
- زيادة قيمة عداد الحلقة بمقدار 10 لكل تكرار - $\$x+=10$.

تتكرر عبر كتلة من الاكواد مرة واحدة، ثم تكرر الحلقة - **do...while** الحلقة طالما كان الشرط المحدد صحيحًا.

كيف تتم هذه الحلقة...

دائمًا كتلة الاكواد مرة واحدة، ثم ستتحقق من الشرط، **do...while** ستنفذ الحلقة وتكرر الحلقة عندما يكون الشرط المحدد صحيحًا

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

أمثلة

بعد ذلك، ستكتب $(x = 1)$ إلى $x = 1$ يقوم المثال أدناه أولاً بتعيين المتغير بـ 1. ثم يتم التحقق x بعض المخرجات، ثم تزيد المتغير **do while** حلقة وستستمر الحلقة في العمل طالما، (أقل من أو يساوي 5؟ x هل) من الشرط: أقل من أو يساوي 5 x

مثال

```
<?php  
$x = 1;
```

```
do {  
    echo "The number is: $x <br>";  
    $x++;  
} while ($x <= 5);  
?>
```

الحلقة يتم اختبار الشرط بعد تنفيذ البيانات داخل **do...while** ملاحظة: في الحلقة ستنفذ بياناتها مرة واحدة على الأقل، **do...while** الحلقة. هذا يعني أن حتى لو كان الشرط خاطئاً. انظر المثال أدناه.

على 6، ثم يقوم بتشغيل الحلقة، ثم يتم التحقق **\$x** يضبط هذا المثال المتغير : من الشرط

مثال

```
<?php  
$x = 6;
```

```
do {  
    echo "The number is: $x <br>";  
    $x++;
```

```
} while ($x <= 5);  
?>
```

حلقات من خلال كتلة من الاكواد لعدد محدد من المرات -for الحلقة

حلقة فور

عندما تعرف مسبقاً عدد المرات التي يجب تشغيل **for** يتم استخدام الحلقة الكود فيها.

حدود:

- تهيئة قيمة عداد الحلقة: **init** عداد
- **TRUE** عداد الاختبار: يتم تقييمه لكل تكرار للحلقة. إذا تم تقييمه إلى
- تنتهي الحلقة، **FALSE** تستمر الحلقة. إذا تم تقييمه إلى
- عداد الزيادة: يزيد من قيمة عداد الحلقة

أمثلة

يعرض المثال أدناه الأرقام من 0 إلى 10

مثال

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

شاهد المثال

- رقم بتعيين قيمة البداية على (\$x) س = 0; - قم بتهيئة عداد الحلقة \$x
- أقل من أو يساوي 10 \$x تابع الحلقة طالما أن - \$x <= 10;
- زيادة قيمة عداد الحلقة بمقدار 1 لكل تكرار - \$x++

مثال

```
<?php
for ($x = 0; $x <= 100; $x+=10) {
    echo "The number is: $x <br>";
}
?>
```

شاهد المثال

- وقم بتعيين قيمة البداية على $(\$x)$ س $= 0$; - قم بتهيئة عداد الحلقة $\$$.
- أقل من أو يساوي $100 \$x$ تابع الحلقة طالما أن $100 \leq \$x$.
- زيادة قيمة عداد الحلقة بمقدار 10 لكل تكرار - $\$x += 10$.

foreach حلقة

حلقات من خلال كتلة من الاكواد لكل عنصر في مصفوفة - **foreach** الحلقة

foreach حلقة

فقط على المصفوفات، وتستخدم للتكرار خلال كل زوج **foreach** تعمل الحلقة من المفاتيح/القيمة في المصفوفة.

```
foreach ($array as $value) {
```

```
    code to be executed;
```

```
}
```

في كل تكرار للحلقة، يتم تعيين قيمة عنصر المصفوفة الحالي إلى قيمة $\$$ ويتم نقل مؤشر المصفوفة بمقدار واحد، حتى يصل إلى عنصر المصفوفة الأخير.

أمثلة

`$colors`): سوف يقوم المثال التالي بإخراج قيم المصفوفة المحددة

مثال

```
<?php
$colors = array("black", "black", "blue", "arabic");

foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

`$Hosiny`): سيُخرج المثال التالي كلاً من مفاتيح وقيم المصفوفة المحددة

مثال

```
<?php
$Hosiny
= array("Habib"=>"35", "Mahmedd"=>"37", "osman"=>"43");

foreach($Hosiny as $x => $val) {
    echo "$x = $val<br>";
}
?>
```

كلمة بريك

العبارة المستخدمة في فصل سابق من هذا الكتاب **break** لقد رأيت بالفعل البيان **switch**. تم استخدامه "للقفز" من

للقفز خارج الحلقة **break** يمكن أيضًا استخدام العبارة : مساوية لـ 4 x يقفز هذا المثال خارج الحلقة عندما تكون

مثال

```
<?php
for ($x = 0; $x < 10; $x++) {
    if ($x == 4) {
        break;
    }
    echo "The number is: $x <br>";
}
?>
```


متابعة

تكرّرًا واحدًا (في الحلقة)، في حالة حدوث شرط **continue** تتجاهل العبارة محدد، وتستمر مع التكرار التالي في الحلقة.

: يتخطى هذا المثال قيمة 4

مثال

```
<?php
for ($x = 0; $x < 10; $x++) {
    if ($x == 4) {
        continue;
    }
    echo "The number is: $x <br>";
}
?>
```

تابع الحلقات

continuewhile: الحلقات **break** يمكنك أيضًا استخدام

مثال كسر

```
<?php
$x = 0;

while($x < 10) {
```

```
if ($x == 4) {  
    break;  
}  
echo "The number is: $x <br>";  
$x++;  
}  
?>
```

متابعة المثال

```
<?php  
$x = 0;  
  
while($x < 10) {  
    if ($x == 4) {  
        $x++;  
        continue;  
    }  
    echo "The number is: $x <br>";  
    $x++;  
}  
?>
```

وظائف مضمونة

على أكثر من 1000 وظيفة مدمجة يمكن استدعاؤها مباشرة، من PHP تحتوي داخل الكود ، لتنفيذ مهمة محددة

الوظائف المحددة من قبل المستخدم

المضمنة، من الممكن إنشاء وظائف الخاصة PHP إلى جانب وظائف

- الدالة عبارة عن كتلة من البيانات التي يمكن استخدامها بشكل متكرر في البرنامج.
- لن يتم تنفيذ الوظيفة تلقائيًا عند تحميل الصفحة.
- سيتم تنفيذ الوظيفة عن طريق استدعاء الوظيفة.

إنشاء وظيفة محددة من قبل المستخدم في

function: يبدأ إعلان الدالة المعرفة من قبل المستخدم بالكلمة

```
function functionName() {
```

```
    code to be executed;
```

```
}
```

ملاحظة: يجب أن يبدأ اسم الوظيفة بحرف أو بشرطة سفلية. أسماء الوظائف ليست حساسة لحالة الأحرف.

نصيحة: أعطِ الدالة اسمًا يعكس ما تفعله الدالة

يشير قوس الفتح `"Hosinyy()"` في المثال أدناه، قمنا بإنشاء دالة باسم المتعرج `()` إلى بداية كود الوظيفة، ويشير قوس الإغلاق المتعرج `()` إلى الاستدعاء `"Abo Habib Alhosinii!"` نهاية الوظيفة. تقوم الوظيفة بإخراج `()` الدالة، ما عليك سوى كتابة اسمها متبوعًا بقوسين

مثال

```
<?php
function Hosinyy() {
    echo "Abo Habib Alhosinii!";
}
```

```
Hosinyy(); // call the function
```

```
?>
```

وسيطات دالة

يمكن تمرير المعلومات إلى الوظائف من خلال الوسائط. الحجة هي تماما مثل المتغير.

يتم تحديد الوسائط بعد اسم الوظيفة، داخل الأقواس. يمكنك إضافة أي عدد تريده من الوسائط، ما عليك سوى الفصل بينها بفاصلة.

عندما يتم (**\$fname**) يحتوي المثال التالي على دالة ذات وسيطة واحدة على سبيل المثال) نقوم أيضًا بتمرير اسم (**Habib_Name()** استدعاء الدالة ويتم استخدام الاسم داخل الدالة، والتي تنتج عدة أسماء أولى مختلفة، (**Jani**) ولكن اسم العائلة متساوٍ

مثال

```
<?php
function Habib_Name($fname) {
    echo "$fname Refsnes.<br>";
}
```

```
Habib_Name("Mahmoud");
Habib_Name("ahmed");
Habib_Name("Stale");
Habib_Name("Hamza");
Habib_Name("Mahmmed");
?>
```

(\$fname و\$year) يحتوي المثال التالي على دالة تحتوي على وسيطتين

مثال

```
<?php
function Habib_Name($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}
```

```
Habib_Name("ahmed", "1975");
Habib_Name("Stale", "3978");
Habib_Name("Hamza", "4983");
?>
```

التوسع في الكتابة

بنوع البيانات الذي PHP في المثال أعلاه، لاحظ أنه لم يكن علينا إخبار
ينتمي إليه المتغير.

تلقائيًا بربط نوع البيانات بالمتغير، اعتمادًا على قيمته. نظرًا لعدم PHP تقوم
تعيين أنواع البيانات بالمعنى الدقيق للكلمة، يمكنك القيام بأشياء مثل إضافة
نصوص إلى عدد صحيح دون التسبب في خطأ.

تمت إضافة إعلانات النوع. يمنحنا هذا خيارًا لتحديد نوع PHP 7، في
سيتم إلقاء ، **strict** البيانات المتوقع عند الإعلان عن دالة، وبإضافة الإعلان
"خطأ فادح" إذا كان نوع البيانات غير متطابق.

strict: في المثال التالي نحاول إرسال رقم ونصوص إلى الدالة دون استخدام

مثال

```
<?php
function Habib_Numbers(int $a, int $b) {
    return $a + $b;
}
echo Habib_Numbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to
int(5), and it will return 10
?>
```

يجب أن `declare(strict_types=1);` نحتاج إلى تعيين `strict` لتحديد PHP. يكون هذا في السطر الأول من ملف

في المثال التالي، نحاول إرسال رقم ونصوص إلى الدالة، لكننا هنا أضفنا `strict` التصريح :

مثال

```
<?php declare(strict_types=1); // strict requirement

function Habib_Numbers(int $a, int $b) {
    return $a + $b;
}
echo Habib_Numbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer, an
error will be thrown
?>
```

استخدام الأشياء بالطريقة المقصودة **strict** يفرض الإعلان

قيمة الوسيطة الافتراضية

يوضح المثال التالي كيفية استخدام المعلمة الافتراضية. إذا قمنا باستدعاء بدون وسيطات فإنها تأخذ القيمة الافتراضية `Hosini_Height()` الدالة كوسيطة:

مثال

```
<?php declare(strict_types=1); // strict requirement
function Hosini_Height(int $hoseni = 50) {
    echo "The height is : $hoseni <br>";
}
```

```
Hosini_Height(350);
Hosini_Height(); // will use the default value of 50
Hosini_Height(135);
Hosini_Height(80);
?>
```


وظائف - إرجاع القيم

العبارة **return** للسماح للدالة بإرجاع قيمة، استخدم

مثال

```
<?php declare(strict_types=1); // strict requirement
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}
```

```
echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

تعريفات نوع الإرجاع

كما هو الحال مع تعريف **return** أيضاً تعريفات النوع للبيان **PHP 7** يدعم النوع لوسيطات الوظيفة، من خلال تمكين المتطلبات الصارمة، فإنه سيؤدي إلى ظهور "خطأ فادح" في حالة عدم تطابق النوع.

للإعلان عن نوع لإرجاع الدالة، قم بإضافة نقطتين (:) والنوع مباشرة قبل {قوس الفتح المتعرج () عند الإعلان عن الدالة.

في المثال التالي نحدد نوع الإرجاع للدالة

مثال

```
<?php declare(strict_types=1); // strict requirement
function Habib_Numbers(float $a, float $b) : float {
    return $a + $b;
}
echo Habib_Numbers(1.2, 5.2);
?>
```

يمكنك تحديد نوع إرجاع مختلف عن أنواع الوسيطات، ولكن تأكد من أن الإرجاع هو النوع الصحيح:

مثال

```
<?php declare(strict_types=1); // strict requirement
function Habib_Numbers(float $a, float $b) : int {
    return (int)($a + $b);
}
echo Habib_Numbers(1.2, 5.2);
?>
```

تمرير البرامترات حسب المرجع

يتم تمرير الوسائط عادةً حسب القيمة، مما يعني أنه يتم استخدام PHP في نسخة من القيمة في الوظيفة ولا يمكن تغيير المتغير الذي تم تمريره إلى الوظيفة.

عندما يتم تمرير وسيطة دالة حسب المرجع، فإن التغييرات في الوسيطة تؤدي أيضًا إلى تغيير المتغير الذي تم تمريره. لتحويل وسيطة دالة إلى مرجع، & يتم استخدام عامل التشغيل:

مثال

:استخدم وسيطة تمريرية لتحديث متغير

```
<?php
function Habib_add(&$value) {
    $value += 5;
}
```

```
$num = 2;
Habib_add($num);
echo $num;
?>
```

تقوم المصفوفة بتخزين قيم متعددة في متغير واحد:

مثال

```
<?php
$habib_Array = array("Abo Habib *", "Al Hosini
*", "Hamz *");
echo "I like " . $habib_Array[0] . ", " . $habib_Array[1] . "
and " . $habib_Array[2] . ".";
?>
```

ما هي المصفوفة؟

المصفوفة عبارة عن متغير خاص يمكنه الاحتفاظ بأكثر من قيمة واحدة في المرة الواحدة.

إذا كانت لديك قائمة بالعناصر (قائمة بأسماء السيارات، على سبيل المثال)، فقد يبدو تخزين السيارات في متغيرات فردية كما يلي:

```
$habib_Array1 = "Abo Habib *";
```

```
$habib_Array2 = "Al Hosini *";
```

```
$habib_Array3 = "Hamz *";
```

ومع ذلك، ماذا لو كنت تريد التنقل بين السيارات والعثور على سيارة معينة؟ وماذا لو لم يكن لديك 3 سيارات بل 300؟

!الحل هو إنشاء مصفوفة

يمكن للمصفوفة أن تحتوي على العديد من القيم تحت اسم واحد، ويمكنك الوصول إلى القيم عن طريق الإشارة إلى رقم الفهرس.

إنشاء مصفوفة في

يتم استخدام الدالة لإنشاء مصفوفة `array()`، PHP في لغة

```
array();
```

هناك ثلاثة أنواع من المصفوفات، PHP في

- المصفوفات المفهرسة - المصفوفات ذات الفهرس الرقمي
- المصفوفات الترابطية - المصفوفات ذات المفاتيح النصية
- المصفوفات متعددة الأبعاد - المصفوفات التي تحتوي على مصفوفة واحدة أو أكثر

count() احصل على طول المصفوفة - دالة

لإرجاع طول (عدد العناصر) للمصفوفة **count()** تُستخدم الدالة

مثال

```
<?php
$habib_Array = array("Abo Habib *", "Al Hosini
*", "Hamz *");
echo count($habib_Array);
?>
```

المصفوفات المفهرسة

هناك طريقتان لإنشاء المصفوفات المفهرسة

:يمكن تعيين الفهرس تلقائيًا (يبدأ الفهرس دائمًا عند 0)، مثل هذا

```
$habib_Array = array("Abo Habib *", "Al Hosini *", "Hamz
*");
```

:أو يمكن تعيين الفهرس يدويًا

```
$habib_Array[0] = "Abo Habib *";
```

```
$habib_Array[1] = "Al Hosini *";
```

```
$habib_Array[2] = "Hamz *";
```

ويعين لها ثلاثة `$habib_Array`، ينشئ المثال التالي مصفوفة مفهرسة باسم عناصر، ثم يطبع نصًا يحتوي على قيم المصفوفة:

مثال

```
<?php
$habib_Array = array("Abo Habib *", "Al Hosini
*", "Hamz *");
echo "I like " . $habib_Array[0] . ", " . $habib_Array[1] . "
and " . $habib_Array[2] . ".";
?>
```

حلقة على مصفوفة مفهرسة

`for`، للتكرار وطباعة جميع قيم المصفوفة المفهرسة، يمكنك استخدام حلقة، مثل هذا:

مثال

```
<?php
$habib_Array = array("Abo Habib *", "Al Hosini
```

```
*, "Hamz *");
$Habib_length = count($habib_Array);

for($x = 0; $x < $Habib_length; $x++) {
    echo $habib_Array[$x];
    echo "<br>";
}
?>
```

المصفوفات الترابطية

المصفوفات الترابطية هي صفائف تستخدم المفاتيح النصية التي تقوم بتعيينها لها.

هناك طريقتان لإنشاء مصفوفة ترابطية:

```
$Hosiny = array("Habib"=>"35", "Mahmedd"=>"37",
"osman"=>"43");
```

أو:

```
$Hosiny['Habib'] = "35";
```

```
$Hosiny['Mahmoudd'] = "37";
```

```
$Hosiny['Joe'] = "43";
```


يمكن بعد ذلك استخدام المفاتيح النصية في برنامج نصي

مثال

```
<?php
$Hosiny
= array("Habib"=>"35", "Mahmedd"=>"37", "osman"=>"4
3");
echo "Habib is " . $Hosiny['Habib'] . " years old.";
?>
```

حلقة من خلال مصفوفة النقابي

للتكرار وطباعة جميع قيم المصفوفة الترابطية، يمكنك استخدام
مثل هذا **foreach** حلقة

مثال

```
<?php
$Hosiny
= array("Habib"=>"35", "Mahmedd"=>"37", "osman"=>"4
3");

foreach($Hosiny as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
```



في الصفحات السابقة، وصفنا المصفوفات التي هي عبارة عن قائمة واحدة من أزواج المفاتيح/القيمة.

ومع ذلك، في بعض الأحيان تريد تخزين القيم باستخدام أكثر من مفتاح واحد. لهذا، لدينا صفائف متعددة الأبعاد.

المصفوفات متعددة الأبعاد -

المصفوفة متعددة الأبعاد هي مصفوفة تحتوي على مصفوفة واحدة أو أكثر المصفوفات متعددة الأبعاد التي تكون بعمق مستويين أو PHP تدعم لغة ثلاثة أو أربعة أو خمسة أو أكثر. ومع ذلك، يصعب إدارة المصفوفات التي يزيد عمقها عن ثلاثة مستويات بالنسبة لمعظم الأشخاص.

يشير بُعد المصفوفة إلى عدد المؤشرات التي تحتاجها لتحديد عنصر

- بالنسبة للمصفوفة ثنائية الأبعاد، تحتاج إلى مؤشرين لتحديد عنصر

- بالنسبة لمصفوفة ثلاثية الأبعاد، تحتاج إلى ثلاثة مؤشرات لتحديد عنصر

صفائف ثنائية الأبعاد -

المصفوفة ثنائية الأبعاد هي مصفوفة من المصفوفات (المصفوفة ثلاثية الأبعاد هي مصفوفة من المصفوفات).

أولاً، ألق نظرة على الجدول التالي:

| Name | Stock | Sold |
|-------------|-------|------|
| Abo Habib * | 22 | 18 |
| Al Hosini * | 15 | 13 |
| omar | 5 | 2 |
| Hatem | 17 | 15 |

يمكننا تخزين البيانات من الجدول أعلاه في مصفوفة ثنائية الأبعاد، كما يلي:

```
$habib_Array = array (  
    array("Abo Habib *",22,18),  
    array("Al Hosini *",15,13),  
    array("omar",5,2),  
    array("Hatem",17,15)  
);
```

ثنائية الأبعاد على أربع مصفوفات، \$habib_Array الآن، تحتوي المصفوفة ولها مؤشران: الصف والعمود.

يجب أن نشير إلى المؤشرين \$habib_Array للوصول إلى عناصر المصفوفة (الصف والعمود):

مثال

```
<?php
echo $habib_Array[0][0].": In stock: ".$habib_Array[0]
[1].", sold: ".$habib_Array[0][2]."<br>";
echo $habib_Array[1][0].": In stock: ".$habib_Array[1]
[1].", sold: ".$habib_Array[1][2]."<br>";
echo $habib_Array[2][0].": In stock: ".$habib_Array[2]
[1].", sold: ".$habib_Array[2][2]."<br>";
echo $habib_Array[3][0].": In stock: ".$habib_Array[3]
[1].", sold: ".$habib_Array[3][2]."<br>";
?>
```

للحصول على عناصر **for** حلقة داخل حلقة أخرى **for** يمكننا أيضًا وضع (لا يزال يتعين علينا الإشارة إلى المؤشرين) \$habib_Array المصفوفة:

مثال

```
<?php
for ($row = 0; $row < 4; $row++) {
    echo "<p><b>Row number $row</b></p>";
    echo "<ul>";
    for ($col = 0; $col < 3; $col++) {
        echo "<li>".$habib_Array[$row][$col]."</li>";
    }
}
```

```
echo "</ul>";
```

```
}
```

```
?>
```

يمكن فرز العناصر الموجودة في المصفوفة بترتيب أبجدي أو رقمي، تنازلياً أو تصاعدياً.

وظائف الفرز للمصفوفات -

PHP: في هذا الفصل، سنتناول وظائف فرز المصفوفات التالية في

- **sort()** - ترتيب المصفوفات ترتيباً تصاعدياً
- **rsort()** - ترتيب المصفوفات ترتيباً تنازلياً
- **asort()** - فرز المصفوفات الترابطية بترتيب تصاعدي حسب القيمة
- **ksort()** - فرز المصفوفات الترابطية بترتيب تصاعدي حسب المفتاح
- **arsort()** - فرز المصفوفات الترابطية بترتيب تنازلي حسب القيمة
- **krsort()** - فرز المصفوفات الترابطية بترتيب تنازلي حسب المفتاح

sort() - فرز المصفوفة بترتيب تصاعدي

بترتيب أبجدي \$habib_Array يقوم المثال التالي بفرز عناصر المصفوفة تصاعدي:

مثال

```
<?php
$habib_Array = array("Abo Habib *", "Al Hosini
*", "Hamz *");
sort($habib_Array);
?>
```

بترتيب رقمي \$numbers يقوم المثال التالي بفرز عناصر المصفوفة
تصاعدي:

مثال

```
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
?>
```

rsort() - فرز المصفوفة بترتيب تنازلي

بترتيب أبجدي \$habib_Array يقوم المثال التالي بفرز عناصر المصفوفة
تنازلي:

مثال

```
<?php
$habib_Array = array("Abo Habib *", "Al Hosini
*", "Hamz *");
rsort($habib_Array);
?>
```

بترتيب رقمي تنازلي يقوم المثال التالي بفرز عناصر المصفوفة

مثال

```
<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);
?>
```

- فرز المصفوفة (ترتيب تصاعدي)، حسب القيمة asort()

يقوم المثال التالي بفرز مصفوفة ترابطية بترتيب تصاعدي، وفقاً للقيمة

مثال

```
<?php
$Hosiny
= array("Habib"=>"35", "Mahmedd"=>"37", "osman"=>"4
3");
asort($Hosiny);
?>
```

- فرز المصفوفة (ترتيب تصاعدي)، وفقاً للمفتاح
ksort()!

يقوم المثال التالي بفرز مصفوفة ترابطية بترتيب تصاعدي، وفقاً للمفتاح

مثال

```
<?php
$Hosiny
= array("Habib"=>"35", "Mahmedd"=>"37", "osman"=>"4
3");
ksort($Hosiny);
?>
```


- فرز المصفوفة (ترتيب تنازلي)، وفقاً للقيمة arsort()

يقوم المثال التالي بفرز مصفوفة ترابطية بترتيب تنازلي، وفقاً للقيمة

مثال

```
<?php
$Hosiny
= array("Habib"=>"35", "Mahmedd"=>"37", "osman"=>"4
3");
arsort($Hosiny);
?>
```

فرز المصفوفة (ترتيب تنازلي)، وفقاً للمفتاح krsort()

يقوم المثال التالي بفرز مصفوفة ترابطية بترتيب تنازلي، وفقاً للمفتاح

مثال

```
<?php
$Hosiny
= array("Habib"=>"35", "Mahmedd"=>"37", "osman"=>"4
3");
krsort($Hosiny);
?>
```

Superglobals - متغيرات العامة

مما يعني، "superglobals" هي PHP بعض المتغيرات المحددة مسبقاً في أنه يمكن الوصول إليها دائماً، بغض النظر عن النطاق - ويمكنك الوصول إليها من أي وظيفة أو كلاس أو ملف دون الحاجة إلى القيام بأي شيء خاص.

هي PHP superglobal المتغيرات:

- \$_SERVER
- \$_REQUEST
- \$_POST
- \$_GET
- \$_FILES
- \$_ENV
- \$_COOKIE
- \$_SESSION

وسوف تشرح الفصول التالية بعضًا من العولمة الفائقة، وسيتم شرح الباقي في فصول لاحقة.

المتغيرات العامة الفائقة هي متغيرات مضمنة متوفرة دائمًا في جميع النطاقات.

\$GLOBALS

يُستخدم للوصول إلى المتغيرات PHP هو متغير عام فائق \$GLOBALS النصي (أيضًا من داخل الوظائف أو PHP العامة من أي مكان في برنامج الأساليب).

بتخزين كافة المتغيرات العامة في مصفوفة تسمى PHP تقوم بحمل الفهرس اسم المتغير. \$GLOBALS[*index*].

\$GLOBALS: يوضح المثال أدناه كيفية استخدام المتغير الشامل الفائق

مثال

```
<?php
```

```
$x = 75;
```

```
$y = 25;
```

```
function Hosini_addition() {
```

```
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
```

```
}
```

```
Hosini_addition();
```

```
echo $z;
```

```
?>
```

\$GLOBALS متغير موجود داخل مصفوفة **z** في المثال أعلاه، نظرًا لأن الوصول إليه أيضًا من خارج الوظيفة!

المتغيرات العامة الفائقة هي متغيرات مضمنة متوفرة دائمًا في جميع النطاقات.

\$_SERVER

يحتوي على معلومات حول الرؤوس **PHP** هو متغير عام فائق **\$_SERVER** والمسارات ومواقع البرامج النصية

\$_SERVER: يوضح المثال أدناه كيفية استخدام بعض العناصر في

مثال

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_Hostini_user_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

\$_SERVER: يسرد الجدول التالي أهم العناصر التي يمكن إدخالها داخل

| Element/Code | وصف |
|---|---|
| <code>\$_SERVER['PHP_SELF']</code> | إرجاع اسم الملف للبرنامج النصي الذي يتم تنفيذه حاليًا |
| <code>\$_SERVER['GATEWAY_INTERFACE']</code> | إرجاع إصدار واجهة البوابة التي يستخدمها (CGI) العامة الخادم |
| <code>\$_SERVER['SERVER_ADDR']</code> | للخادم IP إرجاع عنوان المضيف |
| <code>\$_SERVER['SERVER_NAME']</code> | إرجاع اسم الخادم المضيف (مثل www.Abo Habib) |

AlHosini.com)

| | |
|---|---|
| <code>\$_SERVER['SERVER_SOFTWARE']</code> | إرجاع نصوص تعريف الخادم (مثل Apache/2.2.24) |
| <code>\$_SERVER['SERVER_PROTOCOL']</code> | إرجاع اسم ومراجعة بروتوكول (مثل HTTP/1.1) |
| <code>\$_SERVER['REQUEST_METHOD']</code> | إرجاع طريقة الطلب المستخدمة للوصول إلى الصفحة (مثل POST) |
| <code>\$_SERVER['REQUEST_TIME']</code> | إرجاع الطابع الزمني لبداية الطلب (مثل 1377687496) |
| <code>\$_SERVER['QUERY_STRING']</code> | إرجاع نصوص الاستعلام إذا تم الوصول إلى الصفحة عبر نصوص استعلام |
| <code>\$_SERVER['HTTP_ACCEPT']</code> | إرجاع رأس القبول من الطلب الحالي |
| <code>\$_SERVER['HTTP_ACCEPT_CHARSET']</code> | إرجاع رأس من Accept_Charset (مثل utf-8.ISO-8859-1) |
| <code>\$_SERVER['HTTP_HOST']</code> | إرجاع رأس المضيف من الطلب الحالي |
| <code>\$_SERVER['HTTP_REFERER']</code> | الكامل URL إرجاع عنوان للصفحة الحالية (غير موثوق به لأنه ليس كل وكلاء المستخدم يدعمونه) |
| <code>\$_SERVER['HTTPS']</code> | هل يتم الاستعلام عن الكود |

| | |
|--|--|
| | HTTP من خلال بروتوكول أمن |
| <code>\$_SERVER['REMOTE_ADDR']</code> | من المكان IP إرجاع عنوان الذي يفيد المستخدم الصفحة الحالية |
| <code>\$_SERVER['REMOTE_HOST']</code> | إرجاع اسم المضيف من المكان الذي يعرض فيه المستخدم الصفحة الحالية |
| <code>\$_SERVER['REMOTE_PORT']</code> | إرجاع المنفذ المستخدم على جهاز المستخدم للتواصل مع خادم الويب |
| <code>\$_SERVER['SCRIPT_FILENAME']</code> | إرجاع اسم المسار المطلق للبرنامج النصي الذي يتم تنفيذه حاليًا |
| <code>\$_SERVER['SERVER_ADMIN']</code> | إرجاع القيمة المعطاة لتوجيه في <code>SERVER_ADMIN</code> ملف تكوين خادم الويب (إذا كان الكود يعمل على مضيف ظاهري، فستكون القيمة المحددة لذلك المضيف (مثل) الظاهري) <code>someone@Abo Habib AlHosini.com</code> |
| <code>\$_SERVER['SERVER_PORT']</code> | إرجاع المنفذ الموجود على جهاز الخادم الذي يستخدمه خادم الويب للاتصال (مثل 80) |
| <code>\$_SERVER['SERVER_SIGNATURE']</code> | إرجاع إصدار الخادم واسم |

| | |
|---|---|
| <code>RE']</code> | المضيف الظاهري الذي تمت إضافته إلى الصفحات التي أنشأها الخادم |
| <code>\$_SERVER['PATH_TRANSLATED']</code> | إرجاع المسار المستند إلى نظام الملفات إلى الكود الحالي |
| <code>\$_SERVER['SCRIPT_NAME']</code> | إرجاع مسار الكود الحالي |
| <code>\$_SERVER['SCRIPT_URI']</code> | لصفحة الحالية URI إرجاع |

المتغيرات العامة الفائقة هي متغيرات مضمنة متوفرة دائمًا في جميع النطاقات.

\$_REQUEST

شامل يُستخدم لجمع البيانات بعد PHP هو متغير `$_REQUEST` PHP شامل إرسال نموذج HTML.

يوضح المثال أدناه نموذجًا يحتوي على حقل إدخال و زر إرسال. عندما يقوم المستخدم بإرسال البيانات بالنقر فوق "إرسال"، يتم إرسال بيانات النموذج في هذا المثال، نشير `<form>` إلى الملف المحدد في سمة الإجراء للعلامة إلى هذا الملف نفسه لمعالجة بيانات النموذج. إذا كنت ترغب في استخدام آخر لمعالجة بيانات النموذج، فاستبدله باسم الملف الذي PHP ملف لجمع قيمة `$_REQUEST` تختاره. بعد ذلك، يمكننا استخدام المتغير الشامل حقل الإدخال:

مثال

```
<html>
```

```
<body>
```

```
<form method="post" action="<?php echo $_SERVER  
R['PHP_SELF'];?>">
```

```
  Name: <input type="text" name="fname">
```

```
  <input type="submit">
```

```
</form>
```

```
<?php
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

```
  // collect value of input field
```

```
  $Hosini_name = $_REQUEST['fname'];
```

```
  if (empty($Hosini_name)) {
```

```
    echo "Name is empty";
```

```
  } else {
```

```
    echo $Hosini_name;
```

```
  }
```

```
}
```

```
?>
```

```
</body>
```

```
</html>
```

المتغيرات العامة الفائقة هي متغيرات مضمنة متوفرة دائمًا في جميع النطاقات.

\$_POST[]

شامل يُستخدم لجمع بيانات النموذج بعد PHP هو متغير `$_POST` PHP أيضاً `$_POST` يُستخدم. `"post"` = باستخدام الطريقة HTML إرسال نموذج على نطاق واسع لتمرير المتغيرات.

يوضح المثال أدناه نموذجًا يحتوي على حقل إدخال و زر إرسال. عندما يقوم المستخدم بإرسال البيانات بالنقر فوق "إرسال"، يتم إرسال بيانات النموذج في هذا المثال، نشير `<form>` إلى الملف المحدد في سمة الإجراء للعلامة إلى الملف نفسه لمعالجة بيانات النموذج. إذا كنت ترغب في استخدام ملف آخر لمعالجة بيانات النموذج، فاستبدله باسم الملف الذي تختاره. بعد PHP لجمع قيمة حقل الإدخال `$_POST` ذلك، يمكننا استخدام المتغير الشامل:

مثال

```
<html>
```

```
<body>
```

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
```

```
Name: <input type="text" name="fname">
```

```
<input type="submit">
</form>
```

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $Hosini_name = $_POST['fname'];
    if (empty($Hosini_name)) {
        echo "Name is empty";
    } else {
        echo $Hosini_name;
    }
}
?>
```

```
</body>
</html>
```

المتغيرات العامة الفائقة هي متغيرات مضمنة متوفرة دائماً في جميع النطاقات.

\$_GET[]

شامل يُستخدم لجمع بيانات النموذج بعد PHP هو متغير PHP \$_GET باستخدام الطريقة HTML إرسال نموذج "get".

URL أيضًا جمع البيانات المرسل في عنوان \$_GET يمكن لـ

تحتوي على رابط تشعبي يحتوي على HTML افترض أن لدينا صفحة
معلومات:

```
<html>
```

```
<body>
```

```
<a href="test_get_hosini.htm?
```

```
subject=PHP&web=Abo Habib AlHosini.com">Test
```

```
$GET</a>
```

```
</body>
```

```
</html>
```

يتم إرسال المعلومات، "\$GET" عندما ينقر المستخدم على الرابط
ويمكنك بعد ذلك، "test_get_hosini.htm" إلى "web" و "subject"
\$_GET باستخدام "test_get_hosini.htm" الوصول إلى قيمها في
"test_get_hosini.htm": يوضح المثال أدناه الكود الموجود في

مثال

```
<html>
```

```
<body>
```

```
<?php
```

```
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
```

?>

</body>

</html>

ما هو التعبير العادي؟

التعبير العادي عبارة عن نصوص من الأحرف التي تشكل نمط بحث. عندما تبحث عن بيانات في نص ما، يمكنك استخدام نمط البحث هذا لوصف ما تبحث عنه.

يمكن أن يكون التعبير العادي حرفاً واحداً أو نمطاً أكثر تعقيداً.

يمكن استخدام التعبيرات العادية لإجراء جميع أنواع عمليات البحث عن النص واستبدال النص.

التعبيرات العادية عبارة عن نصوص مكونة من محددات ونمط، PHP في ومعدلات اختيارية

```
$exp = "/Abo Habib AlHosini/i";
```

هو النمط الذي يتم **Abo Habib AlHosini** ، في المثال أعلاه، / هو المحدد وهو معدل يجعل البحث غير حساس لحالة الأحرف **i**، البحث عنه يمكن أن يكون المحدد أي حرف ليس حرفًا أو رقمًا أو شرطة مائلة عكسية أو مسافة. المحدد الأكثر شيوعًا هو الشرطة المائلة للأمام (/)، ولكن عندما يحتوي النمط على شرطة مائلة للأمام، فمن المناسب اختيار محددات أخرى مثل **#** أو **~**.

وظائف التعبير العادي

مجموعة متنوعة من الوظائف التي تسمح لك باستخدام التعبيرات **PHP** توفر والوظائف من أكثر الوظائف **preg_match()** العادية. تعد الوظائف **preg_match_all():preg_replace()** استخدامًا

| Function | وصف |
|-------------------------|--|
| preg_match() | يُرجع 1 إذا تم العثور على النمط في النصوص و 0 إذا لم يكن كذلك |
| preg_match_all() | إرجاع عدد المرات التي تم فيها العثور على النمط في النصوص، والذي قد يكون أيضًا 0 |
| preg_replace() | إرجاع نصوص جديدة حيث تم استبدال الأنماط المتطابقة بنصوص أخرى |

preg_match() باستخدام

ستخبرك الوظيفة ما إذا كانت النصوص تحتوي على تطابقات **preg_match()** للنمط.

مثال

"Abo" استخدم تعبيرًا عاديًا لإجراء بحث غير حساس لحالة الأحرف عن "Habib AlHosini" في نصوص:

```
<?php
$shabebe = "Visit Abo Habib AlHosini";
$Hosine = "/Abo Habib AlHosini/i";
echo preg_match($Hosine, $shabebe); // Outputs 1
?>
```

preg_match_all() باستخدام

ستخبرك الوظيفة بعدد التطابقات التي تم العثور عليها `preg_match_all()`. لنمط في نصوص

مثال

استخدم تعبيرًا عاديًا لإجراء حساب غير حساس لحالة الأحرف لعدد مرات ظهور "ain" في النصوص:

```
<?php
$shabebe = "The rain in SPAIN falls mainly on the
plains.";
$Hosine = "/ain/i";
```

```
echo preg_match_all($HOsine, $shabebe); // Outputs 4  
?>
```

preg_replace() باستخدام

باستبدال كافة مطابقات النمط في نصوص `preg_replace()` ستقوم الدالة بنصوص أخرى.

مثال

بـ `Microsoft` استخدم تعبيرًا عاديًا غير حساس لحالة الأحرف لاستبدال `Abo Habib AlHosini` في نصوص:

```
<?php  
$shabebe = "Visit Microsoft!";  
$HOsine = "/microsoft/i";  
echo preg_replace($HOsine, "Abo Habib AlHosini",  
$shabebe); // Outputs "Visit Abo Habib AlHosini!"  
?>
```

معدّلات التعبير العادي

يمكن للمعدّلات تغيير كيفية إجراء البحث.

| Modifier | وصف |
|----------|---|
| i | يقوم بإجراء بحث غير حساس لحالة الأحرف |
| m | إجراء بحث متعدد الأسطر (الأنماط التي تبحث عن بداية أو نهاية نصوص سوف تتطابق مع بداية أو نهاية كل سطر) |
| u | UTF-8 تمكين المطابقة الصحيحة للأنماط المشفرة |

أنماط التعبير العادية

تُستخدم الأقواس للعثور على مجموعة من الأحرف:

| Expression | وصف |
|------------|---|
| [abc] | ابحث عن حرف واحد من الخيارات الموجودة بين القوسين |
| [^abc] | ابحث عن أي حرف ليس بين قوسين |
| [0-9] | ابحث عن حرف واحد من النطاق من 0 إلى 9 |

الأحرف الأولية

الأحرف الأولية هي أحرف ذات معنى خاص:

| Metacharacter | وصف |
|---------------|---|
| | ابحث عن تطابق لأي من الأنماط المفصولة بـ كما في: قطة كلب سمكة |
| . | ابحث عن مثل واحد فقط لأي شخصية |
| ^ | ^Hello: يجد تطابقاً كبدائية نصوص كما في |
| \$ | Habib\$: يجد تطابقاً في نهاية النصوص كما في |
| \d | ابحث عن رقم |
| \s | ابحث عن حرف مسافة بيضاء |

| | |
|---------------------|---|
| <code>\b</code> | <code>\bWORD</code> ، ابحث عن تطابق في بداية كلمة مثل هذه <code>WORD\b</code> أو في نهاية كلمة مثل هذه |
| <code>\uxxxx</code> | المحدد بالرقم السداسي Unicode ابحث عن حرف <code>xxxx</code> العشري |

محددو الكمية

تحدد الكميات الكميات:

| Quantifier | وصف |
|---------------------|--|
| <code>n+</code> | واحد على الأقل <code>n</code> يطابق أي نصوص تحتوي على |
| <code>n*</code> | <code>n</code> يطابق أي نصوص تحتوي على صفر أو أكثر من تكرارات |
| <code>n?</code> | <code>n</code> يطابق أي نصوص تحتوي على صفر أو تكرار واحد لـ |
| <code>n{x}</code> | <code>Xn's</code> يطابق أي نصوص تحتوي على تسلسل |
| <code>n{x,y}</code> | <code>Y</code> إلى <code>X</code> يطابق أي نصوص تحتوي على تسلسل من |
| <code>n{x,r}</code> | على الأقل <code>Xn's</code> يطابق أي نصوص تحتوي على تسلسل |

ملاحظة: إذا كان تعبيرك يحتاج إلى البحث عن أحد الأحرف الخاصة، فيمكنك استخدام شرطة مائلة عكسية (`\`) للتجاهل منها. على سبيل المثال، للبحث عن علامة استفهام واحدة أو أكثر، يمكنك استخدام التعبير التالي `$H?O?S?ine` = `'\/?+/'`;

التجميع

يمكنك استخدام الأقواس (`()`) لتطبيق محددات الكمية على الأنماط بأكملها. ويمكن استخدامها أيضًا لتحديد أجزاء من النموذج لاستخدامها كمطابقة.

مثال

من خلال البحث "habib" استخدم التجميع للبحث عن كلمة bib : متبوعاً بحالتين من ha عن

```
<?php
$habebe = "Hamzas and habibs.";
$Hosine = "/ha(bib){2}/i";
echo preg_match($Hosine, $habebe); // Outputs 1
?>
```

لجمع بيانات \$_POST و \$_GET PHP superglobals يتم استخدام النموذج.

بسيط HTML نموذج -

بسيطاً يحتوي على حقلي إدخال وزر HTML يعرض المثال أدناه نموذج إرسال:

مثال

```
<html>
<body>
```

```
<form action="welcome_hosini.htm" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
```

```
</body>
</html>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

عندما يقوم المستخدم بملء النموذج أعلاه والنقر فوق زر الإرسال، يتم إرسال
بيانات النموذج للمعالجة إلى ملف
يسمى PHP "welcome_hosini.htm".
يتم إرسال بيانات النموذج باستخدام طريقة
HTTP POST.

لعرض البيانات المقدمة، يمكنك ببساطة تكرار جميع المتغيرات. يبدو
"welcome_hosini.htm" كما يلي:

```
<html>
<body>
```

```
Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
```

```
</body>
</html>
```

:يمكن أن يكون الإخراج شيء من هذا القبيل

Welcome habib

Your email address is habib@H.com

HTTP GET: يمكن أيضًا تحقيق نفس النتيجة باستخدام طريقة

مثال

```
<html>
```

```
<body>
```

```
<form action="welcome_get_hosini.htm" method="get"
```

```
>
```

```
Name: <input type="text" name="name"><br>
```

```
E-mail: <input type="text" name="email"><br>
```

```
<input type="submit">
```

```
</form>
```

```
</body>
```

```
</html>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

كما يلي "welcome_get_hosini.htm" ويبدو

```
<html>
```

```
<body>
```

```
Welcome <?php echo $_GET["name"]; ?><br>
```

```
Your email address is: <?php echo $_GET["email"]; ?>
```

```
</body>
```

```
</html>
```

الكود أعلاه بسيط للغاية. ومع ذلك، فإن الشيء الأكثر أهمية مفقود. تحتاج إلى التحقق من صحة بيانات النموذج لحماية الكود من الاكواد الضارة.

PHP! فكر في الأمان عند معالجة نماذج

لا تحتوي هذه الصفحة على أي تحقق من صحة النموذج، ولكنها توضح فقط كيف يمكنك إرسال بيانات النموذج واستردادها.

مع أخذ **PHP** ومع ذلك، فإن الصفحات التالية سوف تظهر كيفية معالجة نماذج الأمان بعين الاعتبار! يعد التحقق الصحيح من بيانات النموذج أمرًا مهمًا! لحماية النموذج من المتسللين ومرسلي البريد العشوائي

الحصول على مقابل البريد

،على سبيل المثال) بإنشاء مصفوفة **POST** و **GET** يقوم كل من `array(key1 => value1, key2 => value2, key3 => value3, ...)`. تحتوي هذه المصفوفة على أزواج المفاتيح/القيمة، حيث

المفاتيح هي أسماء عناصر التحكم في النموذج والقيم هي البيانات المدخلة من المستخدم.

إنها **\$_GET** و **\$_POST** على أنهما **GET** و **POST** يتم التعامل مع كل من كائنات عامة فائقة، مما يعني أنه يمكن الوصول إليها دائمًا، بغض النظر عن النطاق - ويمكنك الوصول إليها من أي وظيفة أو كلاس أو ملف دون الحاجة إلى القيام بأي شيء خاص.

عبارة عن مجموعة من المتغيرات التي تم تمريرها إلى الكود الحالي **\$_GET** عبر **URL** معالم.

عبارة عن مجموعة من المتغيرات التي تم تمريرها إلى الكود **\$_POST** **HTTP POST** الحالي عبر طريقة.

GET متى تستخدم

(تكون مرئية للجميع **GET** المعلومات المرسلة من نموذج باستخدام طريقة أيضًا **GET** لدى **URL** يتم عرض جميع أسماء المتغيرات وقيمها في عنوان حدود على كمية المعلومات التي سيتم إرسالها. الحد هو حوالي 2000 فمن الممكن **URL** حرف. ومع ذلك، نظرًا لأنه يتم عرض المتغيرات في عنوان وضع إشارة مرجعية على الصفحة. يمكن أن يكون هذا مفيدًا في بعض الحالات لإرسال بيانات غير حساسة **GET** يمكن استخدام

لإرسال كلمات المرور أو غيرها من **GET** ملحوظة: لا ينبغي أبدًا استخدام المعلومات الحساسة

متى تستخدم البريد؟

تكون غير مرئية **POST** المعلومات المرسلة من نموذج باستخدام أسلوب وليس (**HTTP** جميع الأسماء/القيم مضمنة في نص طلب) للآخرين لها حدود على كمية المعلومات التي سيتم إرسالها.

الوظائف المتقدمة مثل دعم الإدخال الثنائي **POST** علاوة على ذلك، يدعم متعدد الأجزاء أثناء تحميل الملفات إلى الخادم.

فمن غير الممكن وضع **URL** ومع ذلك، نظرًا لعدم عرض المتغيرات في عنوان إشارة مرجعية على الصفحة.

لإرسال بيانات النموذج **POST** يفضل المطورون.

بطريقة آمنة **PHP** بعد ذلك، دعونا نرى كيف يمكننا معالجة نماذج

للتحقق من صحة **PHP** يوضح هذا الفصل والفصول التالية كيفية استخدام بيانات النموذج.

التحقق من صحة نموذج

PHP! فكر في الأمان عند معالجة نماذج

مع أخذ الأمان في PHP ستوضح هذه الصفحات كيفية معالجة نماذج الاعتبار. يعد التحقق الصحيح من بيانات النموذج أمرًا مهمًا لحماية النموذج! من المتسللين ومرسلي البريد العشوائي

الذي سنعمل عليه في هذه الفصول على حقول إدخال HTML يحتوي نموذج متنوعة: حقول نصية مطلوبة واختيارية، وأزرار اختيار، و زر إرسال



قواعد التحقق من صحة النموذج أعلاه هي كما يلي:

| Field | قواعد التحقق من الصحة |
|---------|--|
| Name | مطلوب. + يجب أن يحتوي فقط على أحرف ومسافات بيضاء |
| E-mail | مطلوب. + يجب أن يحتوي على عنوان بريد إلكتروني صالح (مع @ و.) |
| Website | خيارية. إذا كان موجودًا، فيجب أن يحتوي على صالح URL عنوان |
| Comment | خيارية. حقل إدخال متعدد الأسطر (منطقة النص) |
| Gender | مطلوب. يجب اختيار واحد |

العادي للنموذج HTML أولاً سنلقي نظرة على كود

حقول النص

حقول الاسم والبريد الإلكتروني وموقع الويب هي عناصر إدخال نص، وحقل HTML التعليق عبارة عن منطقة نصية. يبدو كود

Name: `<input type="text" name="name">`

E-mail: `<input type="text" name="email">`

Website: `<input type="text" name="website">`

Comment: `<textarea name="comment" rows="5" cols="40`

`></textarea>`

أزرار الراديو للإختيار

كما يلي HTML حقول الجنس هي أزرار اختيار ويبدو كود

Gender:

`<input type="radio" name="gender" value="female">Fem`

ale

`<input type="radio" name="gender" value="male">Male`

`<input type="radio" name="gender" value="other">Othe`

r

عنصر النموذج

للمنموذج كما يلي HTML يبدو كود

```
<form method="post" action="<?php echo htmlspecialchars  
ars($_SERVER["PHP_SELF"]);?>">
```

"post" = عند إرسال النموذج، يتم إرسال بيانات النموذج باستخدام الطريقة

\$_SERVER["PHP_SELF"] ما هو المتغير

متغيرًا عامًا فائقًا يُرجع اسم ملف الكود \$_SERVER["PHP_SELF"] يعد الذي يتم تنفيذه حاليًا.

بيانات النموذج المرسل إلى \$_SERVER["PHP_SELF"] لذلك، يرسل الصفحة نفسها، بدلاً من الانتقال إلى صفحة مختلفة. بهذه الطريقة، سيحصل المستخدم على رسائل خطأ في نفس صفحة النموذج.

htmlspecialchars() ما هي وظيفة

بتحويل الأحرف الخاصة إلى كيانات htmlspecialchars() تقوم الدالة مثل < و > و HTML وهذا يعني أنه سيتم استبدال أحرف HTML. يمنع هذا المهاجمين من استغلال الاكواد عن طريق إدخال تعليمات برمجية في (هجمات البرمجة النصية عبر المواقع) Javascript أو HTML بتنسيق النماذج.

Form Security ملاحظة كبيرة حول

\$_SERVER["PHP_SELF"]! يمكن للمتسللين استخدام المتغير

في صفحتك، فيمكن للمستخدم إدخال شرط `PHP_SELF` إذا تم استخدام (XSS) مائلة (/) ثم تنفيذ بعض أوامر البرمجة النصية عبر المواقع

أحد أنواع الثغرات الأمنية في (XSS) تعد البرمجة النصية عبر المواقع المهاجمين من XSS الكمبيوتر والتي توجد عادةً في تطبيقات الويب. يمكن إدخال الكود من جانب العميل في صفحات الويب التي يشاهدها المستخدمون الآخرون.

افترض أن لدينا النموذج التالي في صفحة تسمى "test_form_hosini.htm":

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

العادي في شريط العناوين مثل URL الآن، إذا قام المستخدم بإدخال عنوان "http://www.H.com/test_form_hosini.htm"، فسيتم ترجمة الكود، وأعله إلى:

```
<form method="post" action="test_form_hosini.htm">
```

حتى الان جيدة جدا.

التالي في شريط URL ومع ذلك، ضع في اعتبارك أن المستخدم يدخل عنوان العناوين:

```
http://www.H.com/test_form_hosini.htm/%22%3E
```

```
%3Cscript%3Ealert('hacked')%3C/script%3E
```

في هذه الحالة، سيتم ترجمة الكود أعلاه إلى

```
<form method="post" action="test_form_hosini.htm/"><script>alert('hacked')</script>
```

يضيف هذا الكود علامة الكود وأمر التنبيه. وعندما يتم تحميل الصفحة، سيتم هذا مجرد **مثال** بسيط. (سيظهر للمستخدم مربع تنبيه) JavaScript تنفيذ كود PHP_SELF وغير ضار لكيفية استغلال متغير

انتبه إلى أنه يمكن إضافة أي كود جافا سكريبت داخل علامة <script>! يمكن للمتسلل إعادة توجيه المستخدم إلى ملف على خادم آخر، ويمكن أن يحتوي هذا الملف على تعليمات برمجية ضارة يمكنها تغيير المتغيرات العامة أو إرسال النموذج إلى عنوان آخر لحفظ بيانات المستخدم، على سبيل المثال

كيف تتجنب عمليات الاستغلال والهندسة العكسية

`$_SERVER[_SELF]`

باستخدام `$_SERVER["PHP_SELF"]` يمكن تجنب عمليات استغلال الدالة `htmlspecialchars()`.

يجب أن يبدو كود النموذج كما يلي:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

بتحويل الأحرف الخاصة إلى كيانات (`htmlspecialchars()`) تقوم الدالة فسيؤدي، `PHP_SELF` الآن إذا حاول المستخدم استغلال المتغير `HTML`.
ذلك إلى الإخراج التالي:

```
<form method="post" action="test_form_hosini.htm/
&quot;&gt;&lt;script&gt;alert('hacked')&lt;/script&gt;">
```

فشلت محاولة الاستغلال، ولم يحدث أي ضرر

التحقق من صحة اكواد النموذج باستخدام

أول شيء سنفعله هو تمرير كافة المتغيرات من خلال وظيفة `htmlspecialchars()` الخاصة بـ `PHP`.

؛ ثم إذا حاول المستخدم إرسال `htmlspecialchars()` عندما نستخدم الدالة ما يلي في حقل نصي:

```
<script>location.href('http://www.hacked.com')</script>
```

تجاهل، مثل هذا `HTML` لن يتم تنفيذ هذا، لأنه سيتم حفظه ككود -

```
<script>location.href('http://www.hacked.com')</script>
```

أصبح الكود الآن آمناً ليتم عرضه على الصفحة أو داخل البريد الإلكتروني.

سنقوم أيضاً بأمرين آخرين عندما يرسل المستخدم النموذج

1. قم بإزالة الأحرف غير الضرورية (مسافة إضافية، علامة التبويب، السطر.
 2. (`PHP Trim()` باستخدام وظيفة) الجديد) من بيانات إدخال المستخدم
 - (إزالة الخطوط المائلة العكسية (\)) من بيانات إدخال المستخدم
- `PHP stripslashes()` باستخدام وظيفة

الخطوة التالية هي إنشاء وظيفة تقوم بكل عمليات التحقق نيابةً عنا (وهي أكثر ملاءمة من كتابة نفس الكود مرارًا وتكرارًا).

`test_input()` سوف نقوم بتسمية الدالة

باستخدام وظيفة `$_POST` الآن، يمكننا التحقق من كل متغير `test_input()`، ويبدو الكود كما يلي:

مثال

```
<?php
// define variables and set to empty values
$Hosini_name = $Hosini_email = $Hosini_gender =
$Hosini_comment = $Hosini_website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $Hosini_name = test_input($_POST["name"]);
    $Hosini_email = test_input($_POST["email"]);
    $Hosini_website = test_input($_POST["website"]);
    $Hosini_comment = test_input($_POST["comment"]);
    $Hosini_gender = test_input($_POST["gender"]);
}

function test_input($Hosini_data) {
    $Hosini_data = trim($Hosini_data);
    $Hosini_data = stripslashes($Hosini_data);
    $Hosini_data = htmlspecialchars($Hosini_data);
    return $Hosini_data;
}
?>
```


الكافى فى PHP الجزء الاول Abu Habib Al-Husini

لاحظ أنه فى بداية الكود ، نتحقق مما إذا كان النموذج قد تم إرساله باستخدام `$_SERVER["REQUEST_METHOD"]`. إذا كان `REQUEST_METHOD` هو `POST` ، ويجب التحقق ، فقد تم إرسال النموذج - وإذا لم يتم إرساله ، قم بتخطي التحقق من الصحة واعرض نموذجًا من صحته. إذا لم يتم إرساله ، قم بتخطي التحقق من الصحة واعرض نموذجًا فارغًا.

ومع ذلك، فى المثال أعلاه، كافة حقول الإدخال اختيارية. يعمل الكود بشكل جيد حتى لو لم يُدخل المستخدم أي بيانات.

الخطوة التالية هي جعل حقول الإدخال مطلوبة وإنشاء رسائل خطأ إذا لزم الأمر.

يوضح هذا الفصل كيفية جعل حقول الإدخال مطلوبة وإنشاء رسائل خطأ إذا لزم الأمر.

الحقول المطلوبة -

من جدول قواعد التحقق من الصحة فى الصفحة السابقة، نرى أن حقول "الاسم" و"البريد الإلكتروني" و"الجنس" مطلوبة. لا يمكن أن تكون هذه الحقول HTML فارغة ويجب ملؤها فى نموذج.

| | |
|--------------|--|
| Field | قواعد التحقق من الصحة |
| Name | مطلوب. + يجب أن يحتوي فقط على أحرف ومسافات |

| | |
|----------------|--|
| | بيضاء |
| E-mail | مطلوب. + يجب أن يحتوي على عنوان بريد إلكتروني صالح (مع @ و.) |
| Website | خيارى. إذا كان موجودًا، فيجب أن يحتوي على عنوان صالح URL |
| Comment | خيارى. حقل إدخال متعدد الأسطر (منطقة النص) |
| Gender | مطلوب. يجب اختيار واحد |

في الفصل السابق، كانت جميع حقول الإدخال اختيارية.

\$Hosini_nameErr، في الكود التالي أضفنا بعض المتغيرات الجديدة

و \$Hosini_emailErr، \$Hosini_genderErr،

ستحتوي متغيرات الخطأ هذه على رسائل خطأ \$Hosini_websiteErr.

يتحقق \$_POST بيانًا لكل متغير **if else** للتحقق المطلوبة. لقد أضفنا أيضًا

إذا PHP) وظيفة (**empty()** مع) فارغًا \$_POST هذا مما إذا كان المتغير

كانت فارغة، فسيتم تخزين رسالة خطأ في متغيرات الخطأ المختلفة، وإذا لم

تكن فارغة، فسيتم إرسال بيانات إدخال المستخدم من خلال

: **test_input()** الوظيفة

```
<?php
```

```
// define variables and set to empty values
```

```
$Hosini_nameErr = $Hosini_emailErr = $Hosini_genderErr
```

```
= $Hosini_websiteErr = "";
```

```
$Hosini_name = $Hosini_email = $Hosini_gender =
```

```
$Hosini_comment = $Hosini_website = "";
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {  
    if (empty($_POST["name"])) {  
        $Hosini_nameErr = "Name is requiblack";  
    } else {  
        $Hosini_name = test_input($_POST["name"]);  
    }  
  
    if (empty($_POST["email"])) {  
        $Hosini_emailErr = "Email is requiblack";  
    } else {  
        $Hosini_email = test_input($_POST["email"]);  
    }  
  
    if (empty($_POST["website"])) {  
        $Hosini_website = "";  
    } else {
```

```
$Hosini_website = test_input($_POST["website"]);
```

```
}
```

```
if (empty($_POST["comment"])) {
```

```
$Hosini_comment = "";
```

```
} else {
```

```
$Hosini_comment = test_input($_POST["comment"]);
```

```
}
```

```
if (empty($_POST["gender"])) {
```

```
$Hosini_genderErr = "Gender is requiblack";
```

```
} else {
```

```
$Hosini_gender = test_input($_POST["gender"]);
```

```
}
```

```
}
```

```
?>
```

عرض رسائل الخطأ -

نضيف نصًا صغيرًا بعد كل حقل مطلوب، مما يؤدي HTML. ثم في نموذج إلى إنشاء رسالة الخطأ الصحيحة إذا لزم الأمر (أي إذا حاول المستخدم إرسال النموذج دون ملء الحقول المطلوبة):

مثال

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

```
Name: <input type="text" name="name">
```

```
<span class="error">* <?php echo $Hosini_nameErr;?></span>
```

```
<br><br>
```

E-mail:

```
<input type="text" name="email">
```

```
<span class="error">* <?php echo $Hosini_emailErr;?></span>
```

```
<br><br>
```

Website:

```
<input type="text" name="website">
```

```
<span class="error"><?php echo $Hosini_websiteErr;?></span>
```

```
<br><br>
```

```
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

```
<br><br>
```

Gender:

```
<input type="radio" name="gender" value="female">Female  
<input type="radio" name="gender" value="male">Male  
<input type="radio" name="gender" value="other">Other  
<span class="error">* <?php echo $Hosini_genderErr;?></span>  
<br><br>  
<input type="submit" name="submit" value="Submit">  
</form>
```

الكافي في PHP الجزء الأول Abu Habib Al-Husini

الخطوة التالية هي التحقق من صحة البيانات المدخلة، وهي "هل يحتوي حقل الاسم على أحرف ومسافات بيضاء فقط؟"، و"هل يحتوي حقل البريد الإلكتروني على صيغة عنوان بريد إلكتروني صالحة؟"، وإذا تم ملؤها، "هل يحتوي حقل URL؟ صالح؟" موقع الويب على عنوان

يوضح هذا الفصل كيفية التحقق من صحة الأسماء وعناوين البريد الإلكتروني وURL.

التحقق من صحة الاسم -

يعرض الكود أدناه طريقة بسيطة للتحقق مما إذا كان حقل الاسم يحتوي فقط على أحرف وشروط وفواصل عليا ومسافات بيضاء. إذا كانت قيمة حقل الاسم غير صالحة، قم بتخزين رسالة خطأ:

```
$Hosini_name = test_input($_POST["name"]);  
  
if (!preg_match("/^[a-zA-Z-' ]*$/",$Hosini_name)) {  
    $Hosini_nameErr = "Only letters and white space  
allowed";  
}
```

بالبحث عن نصوص عن النمط، وإرجاع `preg_match()` تقوم الدالة صحيح إذا كان النمط موجوداً، وخطأ في حالة وجوده.

التحقق من صحة البريد الإلكتروني -

الطريقة الأسهل والأكثر أماناً للتحقق من صحة عنوان البريد الإلكتروني هي PHP الخاصة بـ `filter_var()` استخدام وظيفة

في الكود أدناه، إذا لم يكن عنوان البريد الإلكتروني منسقاً بشكل جيد، فقم بتخزين رسالة خطأ:

```
$Hosini_email = test_input($_POST["email"]);  
if (!filter_var($Hosini_email, FILTER_VALIDATE_EMAIL)) {  
    $Hosini_emailErr = "Invalid email format";  
}
```

URL التحقق من صحة عنوان -

(صالحة URL يوضح الكود أدناه طريقة للتحقق مما إذا كانت بنية عنوان إذا كانت صيغة URL). يسمح هذا التعبير العادي أيضًا بشروط في عنوان غير صالحة، فقم بتخزين رسالة خطأ URL عنوان

```
$Hosini_website = test_input($_POST["website"]);  
if (!preg_match("/^b(?:(:https?|ftp):\\V|www\\.)(-a-z0-9+&@#\\V/?=~_!|,;)*[-a-z0-9+&@#\\V/?=~_!|,;]/i",  
$Hosini_website )) {  
    $Hosini_websiteErr = "Invalid URL";  
}
```

التحقق من صحة الاسم والبريد الإلكتروني وعنوان URL

الآن يبدو الكود كما يلي:

مثال

```
<?php
// define variables and set to empty values
$Hosini_nameErr = $Hosini_emailErr = $Hosini_genderErr
= $Hosini_websiteErr = "";
$Hosini_name = $Hosini_email = $Hosini_gender =
$Hosini_comment = $Hosini_website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $Hosini_nameErr = "Name is requiblack";
    } else {
        $Hosini_name = test_input($_POST["name"]);
        // check if name only contains letters and whitespace
        if (!preg_match("/^[a-zA-Z-' ]*$/", $Hosini_name)) {
            $Hosini_nameErr = "Only letters and white space
allowed";
        }
    }
}

if (empty($_POST["email"])) {
    $Hosini_emailErr = "Email is requiblack";
} else {
```



```

$Hosini_email = test_input($_POST["email"]);
// check if e-mail address is well-formed
if (!filter_var($Hosini_email,
FILTER_VALIDATE_EMAIL)) {
    $Hosini_emailErr = "Invalid email format";
}
}

if (empty($_POST["website"])) {
    $Hosini_website = "";
} else {
    $Hosini_website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular
expression also allows dashes in the URL)
    if (!preg_match("/\b(?:?:https?|ftp):\\V|www\\.)[-a-z0-9+&@#\\V%?=~_!:,;]*[-a-z0-9+&@#\\V%?=~_]/i",
$Hosini_website )) {
        $Hosini_websiteErr = "Invalid URL";
    }
}

if (empty($_POST["comment"])) {
    $Hosini_comment = "";
} else {
    $Hosini_comment =
test_input($_POST["comment"]);
}

```

```
if (empty($_POST["gender"])) {
    $Hosini_genderErr = "Gender is requiblack";
} else {
    $Hosini_gender = test_input($_POST["gender"]);
}
}
?>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

والخطوة التالية هي إظهار كيفية منع النموذج من إفراغ كافة حقول الإدخال عندما يقوم المستخدم بإرسال النموذج.

يوضح هذا الفصل كيفية الاحتفاظ بالقيم في حقول الإدخال عندما يضغط المستخدم على زر الإرسال.

احتفظ بالقيم في النموذج -

لإظهار القيم في حقول الإدخال بعد أن يضغط المستخدم على زر الإرسال، النصي داخل سمة القيمة لحقول الإدخال PHP نضيف القليل من برنامج التالية: الاسم والبريد الإلكتروني وموقع الويب. في حقل منطقة نص التعليق، يقوم الكود </textarea> و <textarea> نضع الكود بين علامتي و \$Hosini_email و \$Hosini_name الصغير بإخراج قيمة المتغيرات \$Hosini_website و \$Hosini_comment.

بعد ذلك، نحتاج أيضًا إلى إظهار زر الاختيار الذي تم تحديده. لهذا، يجب علينا التعامل مع السمة المحددة (وليس سمة القيمة لأزرار الاختيار):

```
Name: <input type="text" name="name" value="<?php echo $Hosini_name;?>">
```

```
E-mail: <input type="text" name="email" value="<?php echo $Hosini_email;?>">
```

```
Website: <input type="text" name="website" value="<?php echo $Hosini_website;?>">
```

```
Comment: <textarea name="comment" rows="5" cols="40"><?php echo $Hosini_comment;?></textarea>
```

Gender:

```
<input type="radio" name="gender"
```

```
<?php if (isset($Hosini_gender ) && $Hosini_gender == "female") echo "checked";?>
```

```
value="female">Female
```

```
<input type="radio" name="gender"
```

```
<?php if (isset($Hosini_gender ) && $Hosini_gender  
=="male") echo "checked";?>
```

```
value="male">Male
```

```
<input type="radio" name="gender"
```

```
<?php if (isset($Hosini_gender ) && $Hosini_gender  
=="other") echo "checked";?>
```

```
value="other">Other
```

التاريخ والوقت

لتنسيق التاريخ و/أو الوقت **PHP date()** يتم استخدام وظيفة

Date() وظيفة

بتنسيق الطابع الزمني لتاريخ ووقت أكثر قابلية **PHP date()** تقوم وظيفة للقراءة.

date(format,timestamp)

| Parameter | وصف |
|-----------|---|
| format | مطلوب. يحدد تنسيق الطابع الزمني |
| timestamp | خيارى. يحدد الطابع الزمني. الافتراضي هو التاريخ والوقت الحاليين |

الطابع الزمني عبارة عن نصوص من الأحرف تشير إلى التاريخ و/أو الوقت الذي وقع فيه حدث معين.

الحصول على الوقت

كيفية تنسيق التاريخ و الوقت `date()` تحدد معلمة التنسيق المطلوبة لوظيفة: فيما يلي بعض الأحرف الشائعة الاستخدام للتواريخ:

- د - يمثل اليوم من الشهر (01 إلى 31)
- م - يمثل الشهر (01 إلى 12)
- Y - يمثل السنة (بأربعة أرقام)
- l - يمثل يوم الأسبوع - ('L' حرف صغير)

يمكن أيضًا إدراج أحرف أخرى، مثل "/" أو "." أو "-" بين الأحرف لإضافة تنسيق إضافي.

يقوم المثال أدناه بتنسيق تاريخ اليوم بثلاث طرق مختلفة:

مثال

```
<?php  
echo "Today is " . date("Y/m/d") . "<br>";  
echo "Today is " . date("Y.m.d") . "<br>";  
echo "Today is " . date("Y-m-d") . "<br>";  
echo "Today is " . date("l");  
?>
```

نصيحة

: الوظيفة لتحديث سنة تلقائيًا على موقع الويب `date()` استخدم

مثال

```
&copy; 2010-<?php echo date("Y");?>
```

احصل على الوقت

:فيما يلي بعض الأحرف التي يتم استخدامها بشكل شائع في الأوقات

- H - 24 ساعة للساعة (00 إلى 23)
- h - 12 ساعة للساعة مع الأصفار البادئة (01 إلى 12)
- i - الدقائق ذات الأصفار البادئة (00 إلى 59)

- الثواني ذات الأصفار البادئة (00 إلى 59) - s
- صباحًا أو (Post meridiem و Ante meridiem أ - أحرف صغيرة (مساءً)

يقوم المثال أدناه بإخراج الوقت الحالي بالتنسيق المحدد

مثال

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

!ستعيد التاريخ/الوقت الحالي للخادم date() لاحظ أن وظيفة

احصل على منطقتك الزمنية

إذا كان الوقت الذي حصلت فيه على الكود غير صحيح، فمن المحتمل أن يكون ذلك بسبب وجود الخادم في بلد آخر أو تم إعداده لمنطقة زمنية مختلفة لذلك، إذا كنت تريد أن يكون الوقت صحيحًا وفقًا لموقع معين، فيمكنك ضبط المنطقة الزمنية التي تريد استخدامها.

ثم، "America/New_York" يضبط المثال أدناه المنطقة الزمنية على: يُخرج الوقت الحالي بالتنسيق المحدد

مثال

```
<?php
date_default_timezone_set("America/New_York");
echo "The time is " . date("h:i:sa");
?>
```

mktime() إنشاء وقت مع

طابعًا زمنيًا. إذا تم **date()** تحدد معلمة الطابع الزمني الاختيارية في الدالة حذفه، فسيتم استخدام التاريخ والوقت الحاليين (كما في الأمثلة أعلاه)

للتاريخ. يحتو **Unix** بإرجاع الطابع الزمني لنظام **PHP mktime()** تقوم الدالة يناير 1 **Unix** على عدد الثواني بين عصر **Unix** ي الطابع الزمني لنظام والوقت المحدد (**1970 00:00:00** بتوقيت جرينتش)

```
mktime(hour, minute, second, month, day, year)
```

من عدد من المعلمات في **date()** يقوم المثال أدناه بإنشاء تاريخ ووقت للدالة **mktime()** الوظيفة:

مثال

```
<?php
$d=mktime(11, 14, 54, 8, 12, 2014);
```



```
echo "Created date is " . date("Y-m-d h:i:sa", $d);  
?>
```

strtotime() إنشاء تاريخ من نصوص باستخدام

لتحويل نصوص تاريخ يمكن قراءتها **strtotime()** PHP يتم استخدام وظيفة عدد الثواني منذ 1 يناير 1970 Unix بواسطة الإنسان إلى طابع زمني (00:00:00 بتوقيت جرينتش).

```
strtotime(time, now)
```

strtotime() يقوم المثل أدناه بإنشاء التاريخ والوقت من الوظيفة

مثال

```
<?php  
$d=strtotime("10:30pm April 15 2014");  
echo "Created date is " . date("Y-m-d h:i:sa", $d);  
?>
```

ماهرة جدًا في تحويل النصوص إلى تاريخ، لذا يمكنك إدخال قيم مختلفة

مثال

```
<?php
$d=strtotime("tomorrow");
echo date("Y-m-d h:i:sa", $d) . "<br>";
```

```
$d=strtotime("next Saturday");
echo date("Y-m-d h:i:sa", $d) . "<br>";
```

```
$d=strtotime("+3 Months");
echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

فهو ليس **مثاليًا**، لذا تذكر التحقق من النصوص التي **strtotime()** يجمع ذلك وضعتها هناك

المزيد من أمثلة التاريخ

يوضح المثال أدناه تواريخ أيام السبت الستة التالية

مثال

```
<?php
$Hosini_startdate = strtotime("Saturday");
$Hosini_enddate = strtotime("+6 weeks",
$Hosini_startdate);
```

```
while ($Hosini_startdate < $Hosini_enddate) {  
    echo date("M d", $Hosini_startdate) . "<br>";  
    $Hosini_startdate = strtotime("+1 week",  
$Hosini_startdate);  
}  
>
```

يوضح المثال أدناه عدد الأيام حتى الرابع من يوليو:

مثال

```
<?php  
$d1=strtotime("July 04");  
$d2=ceil(($d1-time())/60/60/24);  
echo "There are " . $d2 . " days until 4th of July."  
>
```

كل النص/الكود/العلامات الموجودة في (**require** أو **include**) تأخذ العبارة الملف المحدد وتنسخها إلى الملف الذي يستخدم عبارة التضمين.

HTML أو PHP يعد تضمين الملفات مفيداً جداً عندما تريد تضمين نفس لغة أو النص في صفحات متعددة بموقع الويب.

تتضمن الكواد

آخر (قبل أن ينفذه PHP في ملف PHP من الممكن إدراج محتوى أحد ملفات الخادم)، مع عبارة التضمين أو الطلب

عبارات التضمين والطلب متطابقة، إلا في حالة الفشل

- **require** ويوقف (E_COMPILE_ERROR) سوف ينتج خطأ فادح الكود
- **include** وسيستمر الكود (E_WARNING) سينتج فقط تحذيرًا

لذا، إذا كنت تريد استمرار التنفيذ وإظهار المخرجات للمستخدمين، حتى إذا كان ملف التضمين مفقودًا، فاستخدم عبارة التضمين. بخلاف ذلك، في حالة المعقد، استخدم دائمًا عبارة PHP أو ترميز تطبيق، CMS، أو Framework، الطلب لتضمين ملف مفتاح لتدفق التنفيذ. سيساعد هذا على تجنب المساس بأمان التطبيق وسلامته، فقط في حالة فقدان ملف رئيسي واحد عن طريق الخطأ يؤدي تضمين الملفات إلى توفير الكثير من العمل. وهذا يعني أنه يمكنك إنشاء ملف رأس أو تذييل أو قائمة قياسي لجميع صفحات الويب. بعد ذلك، عندما يحتاج الرأس إلى التحديث، يمكنك فقط تحديث ملف تضمين الرأس

```
include 'filename';
```

or

```
require 'filename';
```

أمثلة

مثال 1

والذي "footer_hosini.htm" لنفترض أن لدينا ملف تذييل قياسي يسمى يبدو كما يلي:

```
<?php
```

```
echo "<p>Copyright &copy; 1999-" . date("Y") . " Abo
```

```
Habib AlHosini.com</p>";
```

```
?>
```

:العبارة **include** لتضمين ملف التذييل في الصفحة، استخدم

مثال

```
<html>
```

```
<body>
```

```
<h1>Welcome to Habib My home page!</h1>
```

```
<p>Some text.</p>
```

```
<p>Some more text.</p>
```

```
<?php include 'footer_hosini.htm';?>
```

```
</body>
```

```
</html>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

مثال 2

"menu_hosini.htm": افترض أن لدينا ملف قائمة قياسي يسمى

```
<?php
```

```
echo '<a href="/default_hosini.html">Home</a> -
```

```
<a href="/html/default_hosini.html">HTML Tutorial</a> -
```

```
<a href="/css/default_hosini.html">CSS Tutorial</a> -
```

```
<a href="/js/default_hosini.html">JavaScript
```

```
Tutorial</a> -
```

```
<a href="default_hosini.html">PHP Tutorial</a>';
```

```
?>
```

يجب أن تستخدم كافة الصفحات الموجودة في موقع الويب ملف القائمة بحيث يمكن تصميم **<div>** نحن نستخدم عنصر) هذا. وإليك كيفية القيام بذلك (لاحقاً CSS القائمة بسهولة باستخدام

مثال

```
<html>
```

```
<body>
```

```
<div class="menu">
```

```
<?php include 'menu_hosini.htm';?>
```

```
</div>
```

```
<h1>Welcome to Habib My home page!</h1>
```

```
<p>Some text.</p>
```

```
<p>Some more text.</p>
```

```
</body>
```

```
</html>
```

الكافي في PHP الجزء الأول Abu Habib Al-Husini

3 مثال

مع تحديد بعض "vars_hosini.htm" لنفترض أن لدينا ملفاً يسمى المتغيرات:

```
<?php
```

```
$color='black';
```

```
$Hosese='Al Hosini *';
```

```
?>
```

فيمكن استخدام "vars_hosini.htm" بعد ذلك، إذا قمنا بتضمين ملف المتغيرات في ملف الاستدعاء:

مثال

```
<html>
```

```
<body>
```

```
<h1>Welcome to Habib My home page!</h1>
```

```
<?php include 'vars_hosini.htm';
```

```
echo "I have a $color $Hosese.";
```

```
?>
```

```
</body>
```

```
</html>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

require

PHP أيضًا لتضمين ملف في كود **require** يتم استخدام العبارة

ومع ذلك، هناك فرق كبير بين التضمين والطلب؛ عندما يتم تضمين ملف العثور عليه، سيستمر الكود في PHP العبارة ولا يمكن لـ **include** مع التنفيذ:

مثال

```
<html>
```

```
<body>
```

```
<h1>Welcome to Habib My home page!</h1>
```

```
<?php include 'noFileExists_hosini.htm';
```

```
echo "I have noFileExists_ $Hosese.";
```

```
?>
```

```
</body>
```

```
</html>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

echo العبارة، فلن يتم تنفيذ عبارة **require** إذا قمنا بنفس المثال باستخدام :أعدت العبارة خطأ فادحاً **require** لأن تنفيذ الكود يتوقف بعد أن

مثال

```
<html>
```

```
<body>
```

```
<h1>Welcome to Habib My home page!</h1>
```

```
<?php require 'noFileExists_hosini.htm';
```

```
echo "I have noFileExists$Hosese.";
```

```
?>
```

```
</body>
```

```
</html>
```

عندما يكون الملف مطلوباً بواسطة التطبيق **require** يُستخدم

عندما لا يكون الملف مطلوباً ويجب أن يستمر التطبيق عند **include** يُستخدم عدم العثور على الملف

تعد معالجة الملفات جزءاً مهماً من أي تطبيق ويب. غالباً ما تحتاج إلى فتح ملف ومعالجته لمهام مختلفة

التعامل مع الملفات

العديد من الوظائف لإنشاء الملفات وقراءتها وتحميلها وتحريرها PHP لدى

!كن حذراً عند التعامل مع الملفات

عندما تقوم بمعالجة الملفات يجب أن تكون حذراً للغاية

يمكنك أن تسبب الكثير من الضرر إذا فعلت شيئاً خاطئاً. الأخطاء الشائعة هي: تحرير الملف الخاطئ، وملء القرص الصلب بالبيانات المهملة، وحذف محتوى الملف عن طريق الصدفة.

. وظيفة ملف القراءة

بقراءة الملف وكتابته في المخزن المؤقت للإخراج `readfile()` تقوم الوظيفة مخزنًا على الخادم، "Hosini_file.txt" لنفترض أن لدينا ملفًا نصيًا يسمى: ويبدو كالتالي:

لقراءة الملف وكتابته إلى المخزن المؤقت للإخراج هو كما يلي PHP كود:
(ترجع الدالة عدد البايتات المقروءة عند النجاح `readfile()`)

مثال

```
<?php  
echo readfile("Hosini_file.txt");  
?>
```

الكافي في PHP الجزء الأول Abu Habib Al-Husini

مفيدة إذا كان كل ما تريد فعله هو فتح ملف وقراءة `readfile()` تكون الوظيفة محتوياته.

ستعلمك الفصول التالية المزيد حول التعامل مع الملفات.

سنعلمك في هذا الفصل كيفية فتح ملف على الخادم وقراءته وإغلاقه.

fopen() - ملف مفتوح

الوظيفة. تمنحك هذه **fopen()** أفضل طريقة لفتح الملفات هي باستخدام الوظيفة **readfile()** الوظيفة خيرات أكثر من

:خلال الدروس "Hosini_file.txt" سنستخدم الملف النصي

على اسم الملف الذي سيتم فتحه، وتحدد **fopen()** تحتوي المعلمة الأولى المعلمة الثانية الوضع الذي يجب فتح الملف فيه. ينشئ المثال التالي أيضًا غير قادرة على فتح الملف المحدد **fopen()** رسالة إذا كانت الدالة

مثال

```
<?php
$Habib_file =
fopen("Hosini_file.txt", "r") or die("Unable to open
file!");
echo fread($Habib_file,filesize("Hosini_file.txt"));
fclose($Habib_file);
?>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

fread().fclose() نصيحة: سيتم شرح الوظائف أدناه

يمكن فتح الملف بأحد الأوضاع التالية:

Modes وصف

| | |
|----|---|
| r | فتح ملف للقراءة فقط. يبدأ مؤشر الملف في بداية الملف |
| w | فتح ملف للكتابة فقط. يقوم بمسح محتويات الملف أو إنشاء ملف جديد إذا لم يكن موجوداً. يبدأ مؤشر الملف في بداية الملف |
| a | فتح ملف للكتابة فقط. يتم الاحتفاظ بالبيانات الموجودة في الملف. يبدأ مؤشر الملف في نهاية الملف. يقوم بإنشاء ملف جديد إذا كان الملف غير موجود |
| x | وخطأ إذا كان FALSE إنشاء ملف جديد للكتابة فقط. إرجاع الملف موجوداً بالفعل |
| r+ | فتح ملف للقراءة/الكتابة. يبدأ مؤشر الملف في بداية الملف |
| w+ | فتح ملف للقراءة/الكتابة. يقوم بمسح محتويات الملف أو إنشاء ملف جديد إذا لم يكن موجوداً. يبدأ مؤشر الملف في بداية الملف |
| a+ | فتح ملف للقراءة/الكتابة. يتم الاحتفاظ بالبيانات الموجودة في الملف. يبدأ مؤشر الملف في نهاية الملف. يقوم بإنشاء ملف جديد إذا كان الملف غير موجود |
| x+ | وخطأ FALSE يقوم بإنشاء ملف جديد للقراءة/الكتابة. إرجاع إذا كان الملف موجوداً بالفعل |

fread() - ملف قراءة

من ملف مفتوح **fread()** تقراً الدالة

اسم الملف المراد القراءة منه وتحدد **fread()** تحتوي المعلمة الأولى على المعلمة الثانية الحد الأقصى لعدد البايتات المراد القراءة منها

حتى النهاية "Hosini_file.txt" التالي ملف PHP يقرأ كود

```
fread($Habib_file,filesize("Hosini_file.txt"));
```

fclose() - ملف إغلاق

لإغلاق ملف مفتوح **fclose()** يتم استخدام الوظيفة

من ممارسات البرمجة الجيدة إغلاق كافة الملفات بعد الانتهاء منها. لا تريد أن يستهلك الملف المفتوح الموجود على الخادم الموارد!

اسم الملف (أو المتغير الذي يحمل اسم الملف) الذي نريد **fclose()** يتطلب إغلاقه:

```
<?php
```

```
$Habib_file = fopen("Hosini_file.txt", "r");
```

```
// some code to be executed...
```

```
fclose($Habib_file);
```

```
?>
```

fgets () - قراءة سطر واحد

لقراءة سطر واحد من الملف **fgets()** يتم استخدام الوظيفة

"Hosini_file.txt": يوضح المثال أدناه السطر الأول من ملف

مثال

```
<?php
$Habib_file =
fopen("Hosini_file.txt", "r") or die("Unable to open
file!");
echo fgets($Habib_file);
fclose($Habib_file);
?>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

انتقل مؤشر الملف إلى السطر التالي، **fgets()** ملاحظة: بعد استدعاء الوظيفة

feof() - التحقق من نهاية الملف

(EOF) "مما إذا تم الوصول إلى "نهاية الملف" **feof()** تتحقق الوظيفة

الوظيفة مفيدة للتنقل عبر البيانات ذات الطول غير المعروف **feof()** تعد هذه

سطراً تلو الآخر، حتى يتم الوصول "Hosini_file.txt" يقرأ المثال أدناه ملف إلى نهاية الملف:

مثال

```
<?php
$Habib_file =
fopen("Hosini_file.txt", "r") or die("Unable to open
```

```
file!");  
// Output one line until end-of-file  
while(!feof($Habib_file)) {  
    echo fgets($Habib_file) . "<br>";  
}  
fclose($Habib_file);  
?>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

fgetc() - قراءة حرف واحد

لقراءة حرف واحد من ملف **fgetc()** يتم استخدام الوظيفة

حرفاً تلو الآخر، حتى يتم الوصول "Hosini_file.txt" يقرأ المثال أدناه ملف إلى نهاية الملف:

مثال

```
<?php  
$Habib_file =  
fopen("Hosini_file.txt", "r") or die("Unable to open  
file!");  
// Output one character until end-of-file  
while(!feof($Habib_file)) {  
    echo fgetc($Habib_file);  
}  
fclose($Habib_file);  
?>
```


الكافي في PHP الجزء الاول Abu Habib Al-Husini

يتحرك مؤشر الملف إلى الحرف التالي، **fgetc()** ملاحظة: بعد استدعاء الوظيفة

سنعلمك في هذا الفصل كيفية إنشاء ملف على الخادم والكتابة عليه.

() fopen - إنشاء ملف

أيضًا لإنشاء ملف. ربما يكون الأمر مربكًا بعض الشيء، **fopen()** تُستخدم الوظيفة يتم إنشاء الملف باستخدام نفس الوظيفة المستخدمة لفتح **PHP**، ولكن في الملفات.

على ملف غير موجود فسوف يقوم بإنشائه، بشرط أن **fopen()** إذا استخدمت **(a)** أو الاضافة **(w)** يكون الملف مفتوحًا للكتابة.

سيتم **"testfile_Hosini.txt"** يقوم المثال أدناه بإنشاء ملف جديد يسمى **PHP:** إنشاء الملف في نفس الدليل الذي يوجد به كود

مثال

```
$Habib_file = fopen("testfile_Hosini.txt", "w")
```

أذونات الملف

إذا كنت تواجه أخطاء عند محاولة تشغيل هذا الكود، فتأكد من أنك منحت ملف حق الوصول لكتابة المعلومات على محرك الأقراص الثابتة PHP.

fwrite() - الكتابة إلى الملف

للكتابة إلى ملف **fwrite()** يتم استخدام الوظيفة

اسم الملف المراد الكتابة إليه والمعلمة **fwrite()** تحتوي المعلمة الأولى على الثانية هي النصوص المراد كتابتها.

المثال أدناه يكتب بضعة أسماء في ملف جديد يسمى "newfile_Hosini.txt":

مثال

```
<?php
$Habib_file =
fopen("newfile_Hosini.txt", "w") or die("Unable to open
file!");
$Hosini = "habib Doe\n";
fwrite($Habib_file, $Hosini);
$Hosini = "Hosini Doe\n";
fwrite($Habib_file, $Hosini);
fclose($Habib_file);
?>
```

مرتين. في كل مرة `"newfile_Hosini.txt"` لاحظ أننا كتبنا إلى الملف التي تحتوي أولاً على `$Hosini` نكتب فيها إلى الملف، نرسل النصوص وبعد أن انتهينا من `"Hosini Doe"` والثانية تحتوي على `"habib Doe"` `fclose()` الكتابة قمنا بإغلاق الملف باستخدام الدالة.

فسيبدو كما يلي `"newfile_Hosini.txt"` إذا فتحنا ملف

habib Doe

Hosini Doe

الكتابة فوق الملف

على بعض البيانات، يمكننا `"newfile_Hosini.txt"` الآن بعد أن يحتوي إظهار ما يحدث عندما نفتح ملفاً موجوداً للكتابة. سيتم مسح جميع البيانات الموجودة ونبدأ بملف فارغ.

ونكتب `"newfile_Hosini.txt"` في المثال أدناه نفتح الملف الموجود لدينا بعض البيانات الجديدة فيه:

مثال

```
<?php
$Habib_file =
fopen("newfile_Hosini.txt", "w") or die("Unable to open
file!");
$Hosini = "Mickey Mouse\n";
fwrite($Habib_file, $Hosini);
$Hosini = "Minnie Mouse\n";
```

```
fwrite($Habib_file, $Hosini);  
fclose($Habib_file);  
?>
```

فسيفتفي كل من جون وجين، "newfile_Hosini.txt" إذا فتحنا الآن ملف
:ولن تكون هناك سوى البيانات التي كتبناها للتو

Mickey Mouse

Minnie Mouse

اضافة النص الى الملف

باضافة "a" يقوم الوضع "a" يمكنك اضافة البيانات بملف باستخدام الوضع
بتجاوز (ويمسح) المحتوى القديم "w" النص بنهاية الملف، بينما يقوم الوضع
للملف.

ونلحق "newfile_Hosini.txt" في المثال أدناه نفتح الملف الموجود لدينا
:به بعض النص

مثال

```
<?php  
$Habib_file =  
fopen("newfile_Hosini.txt", "a") or die("Unable to open  
file!");
```

```
$Hosini = "Donald Duck\n";  
fwrite($Habib_file, $Hosini);  
$Hosini = "Goofy Goof\n";  
fwrite($Habib_file, $Hosini);  
fclose($Habib_file);  
?>
```

تحميل ملف

من السهل تحميل الملفات إلى الخادم، PHP مع ومع ذلك، مع السهولة يأتي الخطر، لذا كن حذرًا دائمًا عند السماح بتحميل الملفات!

.ini. التعديل على ملف التكوين الأساسي

للسماح بتحميل الملفات PHP أولاً، تأكد من تكوين التوجيه، إلى ملف التكوين بـ `file_uploads` ابحث عن `"php.ini"` في ملف `:"اتش بي واضبطه على" تشغيل`

```
file_uploads = On
```

HTML إنشاء نموذج

يسمح للمستخدمين باختيار ملف الصورة HTML بعد ذلك، قم بإنشاء نموذج الذي يريدون تحميله:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<form action="upload_hosini.htm" method="post" enctype="multipart/form-data">
```

Select image to upload:

```
<input type="file" name="fileToUpload" id="fileToUpload">
```

```
<input type="submit" value="Upload Image" name="submit">
```

```
</form>
```

```
</body>
```

```
</html>
```

أعلاه HTML بعض القواعد التي يجب اتباعها لنموذج:

- **"post"** = تأكد من أن النموذج يستخدم الطريقة
- **enctype = "multipart/form-data"**: يحتاج النموذج أيضاً إلى السمة التالية وهو يحدد نوع المحتوى الذي سيتم استخدامه عند إرسال النموذج

بدون المتطلبات المذكورة أعلاه، لن يعمل تحميل الملف

أشياء أخرى يجب ملاحظتها

حقل الإدخال كعنصر **<input>** للعلامة **type = "file"** تعرض السمة •
تحكم لتحديد الملف، مع وجود زر "استعراض" بجوار عنصر التحكم في
الإدخال

"upload_hosini.htm"، يرسل النموذج أعلاه البيانات إلى ملف يسمى
والذي سنقوم بإنشائه بعد ذلك

الكود لتحميل الملف

على كود تحميل ملف "upload_hosini.htm" يحتوي الملف

```
<?php
```

```
$Hosini_dir = "uploads/";
```

```
$Hosini_file = $Hosini_dir .
```

```
basename($_FILES["fileToUpload"]["name"]);
```

```
$Hosini_uploadOk = 1;
```

```
$Hosini_imageFileType
```

```
= strtolower(pathinfo($Hosini_file,PATHINFO_EXTENSION));
```

```
// Check if image file is a actual image or fake image
```

```
if(isset($_POST["submit"])) {
```

```
    $Hosini_check =
```

```
    getimagesize($_FILES["fileToUpload"]["tmp_name"]);
```

```
    if($Hosini_check !== false) {
```

```
        echo "File is an image - " . $Hosini_check["mime"] . " .";
```

```
        $Hosini_uploadOk = 1;
```

```
    } else {
```

```
        echo "File is not an image.";
```

```
$Hosini_uploadOk = 0;
```

```
}
```

```
}
```

```
?>
```

PHP: شرح الكود

- `$Hosini_dir = "uploads/"` - يحدد الدليل الذي سيتم وضع الملف فيه
- مسار الملف الذي سيتم تحميله `$Hosini_file` يحدد
- لم يتم استخدامه بعد (سيتم استخدامه `$Hosini_uploadOk=1` لاحقاً)
- بامتداد الملف (بالأحرف `$Hosini_imageFileType` يحتفظ الصغيرة)
- بعد ذلك، تحقق مما إذا كان ملف الصورة هو صورة حقيقية أم صورة مزيفة

ملاحظة: ستحتاج إلى إنشاء دليل جديد يسمى "التحميلات" في الدليل الذي يوجد سيتم حفظ الملفات التي تم تحميلها هناك. "upload_hosini.htm" به ملف

تحقق مما إذا كان الملف موجوداً بالفعل

الآن يمكننا إضافة بعض القيود

أولاً، سوف نتحقق مما إذا كان الملف موجوداً بالفعل في مجلد "التحميلات". إذا على `$Hosini_uploadOk` حدث ذلك، فسيتم عرض رسالة خطأ، ويتم تعيين `0`:


```
// Check if file already exists
if (file_exists($Hosini_file)) {
    echo "Sorry, file already exists.";
    $Hosini_uploadOk = 0;
}
```

الحد من حجم الملف

"fileToUpload" أعلاه باسم HTML يُسمى حقل إدخال الملف في نموذج الآن نريد التحقق من حجم الملف. إذا كان حجم الملف أكبر من 500 كيلو بايت، على \$Hosini_uploadOk 0 فسيتم عرض رسالة خطأ، ويتم تعيين

```
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $Hosini_uploadOk = 0;
}
```

الحد من نوع الملف

PNG و JPEG و JPG يسمح الكود أدناه للمستخدمين فقط بتحميل ملفات جميع أنواع الملفات الأخرى تعطي رسالة خطأ قبل تعيين GIF. على \$Hosini_uploadOk 0

```
// Allow certain file formats
if($Hosini_imageFileType != "jpg" &&
```

```
$Hosini_imageFileType != "png" &&  
$Hosini_imageFileType != "jpeg"  
&& $Hosini_imageFileType != "gif" ) {  
    echo "Sorry, only JPG, JPEG, PNG & GIF files are  
allowed.";  
    $Hosini_uploadOk = 0;  
}
```

استكمال تحميل الملف

الكامل كما يلي "upload_hosini.htm" يبدو الآن ملف

```
<?php  
$Hosini_dir = "uploads/";  
$Hosini_file = $Hosini_dir .  
basename($_FILES["fileToUpload"]["name"]);  
$Hosini_uploadOk = 1;  
$Hosini_imageFileType  
= strtolower(pathinfo($Hosini_file,PATHINFO_EXTENSI  
ON));  
  
// Check if image file is a actual image or fake image  
if(isset($_POST["submit"])) {  
    $Hosini_check =  
getimagesize($_FILES["fileToUpload"]["tmp_name"]);  
    if($Hosini_check !== false) {  
        echo "File is an image - " . $Hosini_check["mime"] . " .";  
    }  
}
```

```
$Hosini_uploadOk = 1;
} else {
    echo "File is not an image.";
    $Hosini_uploadOk = 0;
}
}
```

```
// Check if file already exists
if (file_exists($Hosini_file)) {
    echo "Sorry, file already exists.";
    $Hosini_uploadOk = 0;
}
```

```
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $Hosini_uploadOk = 0;
}
```

```
// Allow certain file formats
if($Hosini_imageFileType != "jpg" &&
    $Hosini_imageFileType != "png" &&
    $Hosini_imageFileType != "jpeg"
    && $Hosini_imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are
    allowed.";
    $Hosini_uploadOk = 0;
}
```

```
// Check if $Hosini_uploadOk is set to 0 by an error
if ($Hosini_uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]
["tmp_name"], $Hosini_file)) {
        echo "The file ".
htmlspecialchars( basename( $_FILES["fileToUpload"]
["name"])). " has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```

ما هو ملف تعريف الارتباط؟

غالبًا ما يتم استخدام ملف تعريف الارتباط لتحديد هوية المستخدم. ملف تعريف الارتباط هو ملف صغير يقوم الخادم بتضمينه على جهاز الكمبيوتر الخاص بالمستخدم. في كل مرة يطلب فيها نفس الكمبيوتر صفحة بها متصفح، فإنه يمكنك إنشاء واسترجاع قيم PHP، سيرسل ملف تعريف الارتباط أيضًا. باستخدام ملفات تعريف الارتباط.

إنشاء ملفات تعريف الارتباط باستخدام

الوظيفة `setcookie()` يتم إنشاء ملف تعريف الارتباط باستخدام

```
setcookie(name, value, expire, path, domain, secure,  
httponly);
```

مطلوب معلمة الاسم فقط . جميع المعلمات الأخرى اختيارية

إنشاء/استرداد ملف تعريف الارتباط

يقوم المثال التالي بإنشاء ملف تعريف الارتباط المسمى "المستخدم" بالقيمة ستنتهي صلاحية ملف تعريف الارتباط بعد 30 يومًا (86400). "habib Doe". * (30). يعني "/" أن ملف تعريف الارتباط متاح في موقع الويب بأكمله (وإلا، حدد الدليل الذي تفضله).

باستخدام (نقوم بعد ذلك باسترداد قيمة ملف تعريف الارتباط "المستخدم الوظيفة لمعرفة ما إذا تم `isset()` نستخدم أيضًا `$_COOKIE` المتغير العام :تعيين ملف تعريف الارتباط

مثال

```
<?php
```

```
$Hosini_cookie_name = "Hosini_user";
```

```
$Hosini_cookie_value = "habib Doe";
```

```
setcookie($Hosini_cookie_name, $Hosini_cookie_value,  
time() + (86400 * 30), "/"); // 86400 = 1 day
```

```
?>
```

```
<html>
```

```
<body>
```

```
<?php
```

```
if(!isset($_COOKIE[$Hosini_cookie_name])) {  
    echo "Cookie named " . $Hosini_cookie_name . " is not  
set!";
```

```
} else {
```

```
    echo "Cookie " . $Hosini_cookie_name . " is set!<br>";
```

```
    echo "Value is: " . $_COOKIE[$Hosini_cookie_name];
```

```
}
```

```
?>
```

```
</body>
```

```
</html>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

<html> قبل العلامة **setcookie()** ملاحظة: يجب أن تظهر الوظيفة

ملاحظة: يتم تشفير قيمة ملف تعريف الارتباط تلقائياً عند إرسال ملف تعريف URL لمنع تشفير) الارتباط، ويتم فك تشفيرها تلقائياً عند استلامها (بدلاً من ذلك **setrawcookie()** استخدم

تعديل قيمة ملف تعريف الارتباط

لتعديل ملف تعريف الارتباط، ما عليك سوى تعيين (مرة أخرى) ملف تعريف الارتباط باستخدام الوظيفة **setcookie()**:

مثال

```
<?php
$Hosini_cookie_name = "Hosini_user";
$Hosini_cookie_value = "Alex Porter";
setcookie($Hosini_cookie_name, $Hosini_cookie_value,
time() + (86400 * 30), "/");
?>

<html>
<body>

<?php
if(!isset($_COOKIE[$Hosini_cookie_name])) {
    echo "Cookie named " . $Hosini_cookie_name . " is not
set!";
} else {
    echo "Cookie " . $Hosini_cookie_name . " is set!<br>";
    echo "Value is: " . $_COOKIE[$Hosini_cookie_name];
}
?>

</body>
</html>
```

حذف ملف تعريف الارتباط

ذات تاريخ انتهاء **setcookie()** لحذف ملف تعريف الارتباط، استخدم الوظيفة:
الصلاحيه فى الماضى:

مثال

```
<?php
// set the expiration date to one hour ago
setcookie("Hosini_user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'Hosini_user' is deleted.";
?>

</body>
</html>
```


تحقق مما إذا تم تمكين ملفات تعريف الارتباط

يقوم المثال التالي بإنشاء برنامج نصي صغير يتحقق من تمكين ملفات تعريف الارتباط. أولاً، حاول إنشاء ملف تعريف ارتباط اختباري باستخدام `$_COOKIE` ثم احسب متغير المصفوفة `setcookie()` الوظيفة

مثال

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

الجلسة هي وسيلة لتخزين المعلومات (في المتغيرات) لاستخدامها عبر صفحات متعددة.

وعلى عكس ملف تعريف الارتباط، لا يتم تخزين المعلومات على كمبيوتر المستخدم.

ما هي جلسة ؟

عندما تعمل مع تطبيق ما، فإنك تفتحه، وتقوم ببعض التغييرات، ثم تغلقه. هذا يشبه إلى حد كبير الجلسة. الكمبيوتر يعرف من أنت. فهو يعرف متى تبدأ التطبيق ومتى تنتهي. ولكن هناك مشكلة واحدة على الإنترنت: لا يعرف خادم الويب من لا يحتفظ بحالته **HTTP** أنت أو ماذا تفعل، لأن عنوان

تعمل متغيرات الجلسة على حل هذه المشكلة عن طريق تخزين معلومات المستخدم لاستخدامها عبر صفحات متعددة (مثل اسم المستخدم واللون المفضل وما إلى ذلك). بشكل افتراضي، تستمر متغيرات الجلسة حتى يقوم المستخدم بإغلاق المتصفح.

لذا؛ تحتوي متغيرات الجلسة على معلومات حول مستخدم واحد، وهي متاحة لجميع الصفحات في تطبيق واحد.

نصيحة: إذا كنت بحاجة إلى تخزين دائم، فقد ترغب في تخزين البيانات في [قاعدة بيانات](#).

ابدأ جلسة

`session_start()`. تبدأ الجلسة بالوظيفة

PHP: `$_SESSION`. يتم تعيين متغيرات الجلسة باستخدام المتغير العام

الآن، لنقم بإنشاء صفحة جديدة تسمى

PHP في هذه الصفحة، نبدأ جلسة `"demo_session1_hosini.htm"`.

جديدة ونقوم بتعيين بعض متغيرات الجلسة:

مثال

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["Hosini_user"] = "albadrashen";
$_SESSION["Habib_user"] = "arabic";
echo "Session variables are set.";
?>

</body>
</html>
```

أول شيء فى المستند . قبل `session_start()` ملاحظة: يجب أن تكون الوظيفة HTML أي علامات

احصل على القيم المتغيرة لجلسة

بعد ذلك، نقوم بإنشاء صفحة أخرى تسمى "demo_session2_hosini.htm". سنصل إلى معلومات الجلسة التي قمنا بتعيينها في الصفحة الأولى ("demo_session1_hosini.htm").

لاحظ أن متغيرات الجلسة لا يتم تمريرها بشكل فردي إلى كل صفحة جديدة، وبدلاً من ذلك يتم استرجاعها من الجلسة التي نفتحها في بداية كل صفحة (`session_start()`).

`$_SESSION` لاحظ أيضاً أن جميع قيم متغيرات الجلسة مخزنة في المتغير العام:

مثال

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

```
<?php
// Echo session variables that were set on previous page
echo " Hosini_user is
" . $_SESSION["Hosini_user"] . "<br>";
echo "Favorite animal is
" . $_SESSION["Habib_user"] . ".";
?>

</body>
</html>
```

الكافى فى PHP الجزء الاول Abu Habib Al-Husini

هناك طريقة أخرى لإظهار كافة قيم متغيرات الجلسة لجلسة المستخدم وهي تشغيل الكود التالي:

مثال

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
?>
```

```
</body>
</html>
```

الكافى فى PHP الجزء الاول Abu Habib Al-Husini

كيف يعمل؟ كيف يعرف أنه أنا؟

تقوم معظم الجلسات بتعيين مفتاح مستخدم على كمبيوتر المستخدم والذي يبدو وبعد ذلك، عند **cf34ert8dede5a562e4f3a7e12** كالتالي: **765487** فتح جلسة على صفحة أخرى، تقوم بفحص الكمبيوتر بحثاً عن مفتاح المستخدم. إذا كان هناك تطابق، فإنه يصل إلى تلك الجلسة، وإذا لم يكن الأمر كذلك، فإنه يبدأ جلسة جديدة.

تعديل متغير الجلسة

:لتغيير متغير الجلسة، ما عليك سوى الكتابة فوقه

مثال

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

```
<?php
```

```
// to change a session variable, just overwrite it
$_SESSION["Hosini_user"] = "arabic";
print_r($_SESSION);
?>
```

```
</body>
</html>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

حذف جلسة

لإزالة كافة متغيرات الجلسة العامة وحذف الجلسة، استخدم `session_unset()` و `session_destroy()`:

مثال

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

```
<?php
// remove all session variables
session_unset();
```

```
// destroy the session
```

```
session_destroy();
```

```
?>
```

```
</body>
```

```
</html>
```

التحقق من صحة البيانات = تحديد ما إذا كانت البيانات في النموذج الصحيح.

تصحيح البيانات = إزالة أي حرف غير قانوني من البيانات.

الفلتر

للتحقق من صحة المدخلات الخارجية وتصحيحها PHP تُستخدم فلاتر

على العديد من الوظائف اللازمة للتحقق من مدخلات PHP يحتوي ملحق فلتر المستخدم، وهو مصمم لجعل التحقق من صحة البيانات أسهل وأسرع

PHP لسرد ما يقدمه فلتر `filter_list()` يمكن استخدام الوظيفة

مثال

```
<table>
```

```
<tr>
```



```
<td>Filter Name</td>
<td>Filter ID</td>
</tr>
<?php
foreach (filter_list() as $id =>$filter) {
    echo '<tr><td>' . $filter . '</td><td>' .
filter_id($filter) . '</td></tr>';
}
?>
</table>
```

لماذا استخدام الفلاتر؟

تتلقى العديد من تطبيقات الويب مدخلات خارجية. يمكن أن تكون المدخلات/البيانات الخارجية:

- إدخال المستخدم من النموذج
- بسكويت
- بيانات خدمات الويب
- متغيرات الخادم
- نتائج استعلام قاعدة البيانات

! يجب عليك دائمًا التحقق من صحة البيانات الخارجية

يمكن أن تؤدي البيانات المقدمة غير الصالحة إلى مشاكل أمنية وتعطل صفحة الويب!

!يمكنك التأكد من حصول تطبيقك على الإدخال الصحيح، PHP باستخدام فلاتر

دالة filter_var()

على التحقق من صحة البيانات وتصحيحها `filter_var()` تعمل الوظيفة بتصفية متغير واحد باستخدام فلتر محدد. يستغرق `filter_var()` تقوم الدالة بقطعيتين من البيانات:

- المتغير الذي تريد التحقق منه
- نوع الشيك المستخدم

تطهير نصوص

من HTML لإزالة كافة علامات `filter_var()` يستخدم المثال التالي الوظيفة نصوص:

مثال

```
<?php
$shabebe = "<h1>Abo Habib Alhosinii !</h1>";
$newstr = filter_var($shabebe,
FILTER_SANITIZE_STRING);
echo $newstr;
?>
```

التحقق من صحة عدد صحيح

`$int` الدالة للتحقق مما إذا كان المتغير (`filter_var()`) يستخدم المثال التالي عددًا صحيحًا، فسيكون مخرج الكود أدناه: "عدد `$int` عددًا صحيحًا. إذا كان عددًا صحيحًا، فسيكون الإخراج: "عدد صحيح `$int` صحيح صالح". إذا لم يكن "غير صالح":

مثال

```
<?php
$int = 100;
```

```
if (!filter_var($int, FILTER_VALIDATE_INT) === false) {
    echo("Integer is valid");
} else {
    echo("Integer is not valid");
}
?>
```

على 0، فسوف تُرجع الدالة أعلاه "عدد `$int` في المثال أعلاه، إذا تم تعيين صحيح غير صالح". لحل هذه المشكلة استخدم الكود التالي:

مثال

```
<?php
$int = 0;
```

```
if (filter_var($int, FILTER_VALIDATE_INT) === 0 || !
filter_var($int, FILTER_VALIDATE_INT) === false) {
    echo("Integer is valid");
} else {
    echo("Integer is not valid");
}
?>
```

IP التحقق من صحة عنوان

\$ip للتحقق مما إذا كان المتغير (`filter_var()`) يستخدم المثال التالي الوظيفة: صالح IP هو عنوان

مثال

```
<?php
$ip = "127.0.0.1";

if (!filter_var($ip, FILTER_VALIDATE_IP) === false) {
    echo("$ip is a valid IP address");
} else {
    echo("$ip is not a valid IP address");
}
?>
```

تصحيح والتحقق من صحة عنوان البريد الإلكتروني

لإزالة جميع الأحرف غير الصحيحة (`filter_var()`) يستخدم المثال التالي الوظيفة أولاً، ثم التحقق مما إذا كان عنوان بريد إلكتروني `$Hosini_email` من المتغير صالحًا:

مثال

```
<?php
$Hosini_email = "habib@H.com";

// Remove all illegal characters from email
$Hosini_email = filter_var($Hosini_email,
FILTER_SANITIZE_EMAIL);

// Validate e-mail
if (!filter_var($Hosini_email, FILTER_VALIDATE_EMAIL)
=== false) {
    echo("$Hosini_email is a valid email address");
} else {
    echo("$Hosini_email is not a valid email address");
}
?>
```

URL تصحيح والتحقق من صحة عنوان

الوظيفة لإزالة كافة الأحرف غير الصحيحة (`filter_var()`) يستخدم المثال التالي
صالح URL هو عنوان \$url أولاً، ثم التحقق مما إذا كان URL من عنوان

مثال

```
<?php
```

```
$url = "https://www.Abo Habib AlHosini.com";
```

```
// Remove all illegal characters from a url
```

```
$url = filter_var($url, FILTER_SANITIZE_URL);
```

```
// Validate url
```

```
if (!filter_var($url, FILTER_VALIDATE_URL) === false) {
```

```
    echo("$url is a valid URL");
```

```
} else {
```

```
    echo("$url is not a valid URL");
```

```
}
```

```
?>
```

التحقق من صحة عدد صحيح ضمن النطاق

الدالة للتحقق مما إذا كان المتغير من `filter_var()` يستخدم المثال التالي
نوعين 1 و 200، `INT` النوع

مثال

```
<?php
$int = 122;
$min = 1;
$max = 200;

if (filter_var($int,
FILTER_VALIDATE_INT, array("options" => array("min_r
ange"=>$min, "max_range"=>$max))) === false) {
    echo("Variable value is not within the legal range");
} else {
    echo("Variable value is within the legal range");
}
?>
```

IPv6 التحقق من صحة عنوان

`$ip` للتحقق مما إذا كان المتغير `filter_var()` يستخدم المثال التالي الوظيفة
صالح `IPv6` هو عنوان

مثال

```
<?php
$ip = "2001:0db8:85a3:08d3:1319:8a2e:0370:7334";

if (!filter_var($ip, FILTER_VALIDATE_IP,
FILTER_FLAG_IPV6) === false) {
    echo("$ip is a valid IPv6 address");
} else {
    echo("$ip is not a valid IPv6 address");
}
?>
```

يجب أن يحتوي على - URL التحقق من صحة عنوان نصوص الاستعلام

\$url للتحقق مما إذا كان المتغير `filter_var()` يستخدم المثال التالي الوظيفة:
يحتوي على نصوص استعلام URL عبارة عن عنوان

مثال

```
<?php
$url = "https://www.Abo Habib AlHosini.com";

if (!filter_var($url, FILTER_VALIDATE_URL,
FILTER_FLAG_QUERY_REQUIRED) === false) {
```



```
echo("$url is a valid URL with a query string");
} else {
echo("$url is not a valid URL with a query string");
}
?>
```

ASCII > 127 قم بإزالة الأحرف ذات قيمة

الدالة لتطهير نصوص. سيؤدي ذلك إلى إزالة `filter_var()` يستخدم المثال التالي من النصوص، `ASCII > 127` وجميع الأحرف ذات قيمة `HTML`، كافة علامات

مثال

```
<?php
$habebe = "<h1>Abo Habib Alhosinii ÆØÅ!</h1>";

$newstr = filter_var($habebe, FILTER_SANITIZE_STRING,
FILTER_FLAG_STRIP_HIGH);
echo $newstr;
?>
```

وظائف رد الاتصال

وظيفة رد الاتصال (التي يشار إليها غالبًا باسم "رد الاتصال") هي دالة يتم تمريرها كوسيلة إلى وظيفة أخرى.

يمكن استخدام أي وظيفة موجودة كوظيفة رد اتصال. لاستخدام دالة كدالة رد اتصال، قم بتمرير نصوص تحتوي على اسم الدالة كوسيط لدالة أخرى.

مثال

لحساب طول كل نصوص `PHP array_map()` قم بتمرير رد اتصال إلى دالة في المصفوفة:

```
<?php
function Hosini_callback($item) {
    return strlen($item);
}

$hoshos = ["Hamza", "alee", "habib", "osama"];
$lengths = array_map("Hosini_callback", $hoshos);
print_r($lengths);
?>
```

تمرير وظائف مجهولة كوظائف رد اتصال PHP بدءًا من الإصدار 7، يمكن لـ

مثال

PHP: لوظيفة `array_map()` استخدم دالة مجهولة كرد اتصال

```
<?php
$shoshos = ["Hamza", "alee", "habib", "osama"];
$lengths = array_map( function($item)
{ return strlen($item); } , $shoshos);
print_r($lengths);
?>
```

عمليات الاسترجاعات في الوظائف المحددة من قبل المستخدم

يمكن للوظائف والأساليب المحددة من قبل المستخدم أيضًا أن تأخذ وظائف رد الاتصال كوسائط. لاستخدام دوال رد الاتصال داخل دالة أو طريقة معرفة من قبل المستخدم، قم باستدعائها عن طريق إضافة أقواس إلى المتغير وتمرير الوسائط: كما هو الحال مع الدوال العادية

مثال

قم بتشغيل رد اتصال من وظيفة معرفة من قبل المستخدم

```
<?php
function $hosine($habebe) {
    return $habebe . "! ";
}
```

```
function ask($habebe) {
    return $habebe . "? ";
}
```

```
function $hosine_printFormatted($habebe, $format) {
    // Calling the $format callback function
    echo $format($habebe);
}
```

```
// Pass "$hosine" and "ask" as callback functions to
$hosine_printFormatted()
$hosine_printFormatted("Abo Habib Alhosinii
", "$hosine");
$hosine_printFormatted("Abo Habib Alhosinii ", "ask");
?>
```

ما هو جسون؟

وهو عبارة عن صيغة JavaScript Object Notation، إلى JSON يكوّد لتخزين البيانات وتبادلها.

هو تنسيق قائم على النص، فيمكن إرساله بسهولة من وإلى JSON نظرًا لأن تنسيق الخادم واستخدامه كتتنسيق بيانات بواسطة أي لغة برمجة.

الطرق لجسون

JSON بعض الوظائف المضمنة للتعامل مع PHP لدى

أولاً، سننظر في الوظيفتين التاليتين

- `json_encode()`
- `json_decode()`

- `json_encode()`

JSON لترميز قيمة بتنسيق `json_encode()` يتم استخدام الدالة

مثال

JSON: يوضح هذا المثال كيفية تشفير مصفوفة ترابطية في كائن

```
<?php
```

```
$Hosiny
```

```
= array("Habib"=>35, "Mahmedd"=>37, "osman"=>43);
```

```
echo json_encode($Hosiny);  
?>
```

الكافى فى PHP الجزء الاول Abu Habib Al-Husini

مثال

JSON: يوضح هذا المثال كيفية تحويل مصفوفة مفهرسة في مصفوفة

```
<?php  
$habib_Array = array("Abo Habib *", "Al Hosini  
*", "Hamz *");
```

```
echo json_encode($habib_Array);  
?>
```

الكافى فى PHP الجزء الاول Abu Habib Al-Husini

json_decode() -

PHP إلى كائن **JSON** لفك تشفير كائن **json_decode()** يتم استخدام الدالة أو مصفوفة اقترانية.

مثال

PHP: إلى كائن **JSON** يقوم هذا المثال بفك تشفير بيانات

```
<?php
```

```
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';
```

```
var_dump(json_decode($jsonobj));
```

```
?>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

بإرجاع كائن بشكل افتراضي. تحتوي `() json_decode` تقوم الدالة `true` على معلمة ثانية، وعند ضبطها على القيمة `json_decode()` الدالة إلى صفائف ترابطية JSON يتم فك تشفير كائنات

مثال

PHP: إلى مصفوفة ترابطية JSON يقوم هذا المثال بفك ترميز بيانات

```
<?php
```

```
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';
```

```
var_dump(json_decode($jsonobj, true));
```

```
?>
```

الكافي في PHP الجزء الاول Abu Habib Al-Husini

الوصول إلى القيم التي تم فك تشفيرها -

فيما يلي مثالان لكيفية الوصول إلى القيم التي تم فك تشفيرها من كائن ومن مصفوفة اقترانية:

مثال

PHP: يوضح هذا المثال كيفية الوصول إلى القيم من كائن

```
<?php
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';

$obj = json_decode($jsonobj);

echo $obj->Habib;
echo $obj->Mahmedd;
echo $obj->osman;
?>
```

الكافي في PHP الجزء الأول Abu Habib Al-Husini

مثال

PHP: يوضح هذا المثال كيفية الوصول إلى القيم من مصفوفة ترابطية

```
<?php
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';

$arr = json_decode($jsonobj, true);

echo $arr["Habib"];
echo $arr["Mahmedd"];
```



```
echo $arr["osman"];  
?>
```

الكافى فى PHP الجزء الاول Abu Habib Al-Husini

التكرار على جسون

foreach() : يمكنك أيضًا تكرار القيم باستخدام حلقة

مثال

PHP: يوضح هذا المثال كيفية التكرار عبر قيم كائن

```
<?php  
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';  
$obj = json_decode($jsonobj);  
foreach($obj as $key => $value) {  
    echo $key . "=>" . $value . "<br>";  
}  
?>
```

الكافى فى PHP الجزء الاول Abu Habib Al-Husini

مثال

:يوضح هذا المثال كيفية التكرار عبر قيم مصفوفة الترابطية

```
<?php
```

```
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';
```

```
$arr = json_decode($jsonobj, true);
```

```
foreach($arr as $key => $value) {  
    echo $key . " => " . $value . "<br>";  
}
```

ما هو الاستثناء؟

النصي PHP الاستثناء هو كائن يصف خطأ أو سلوكاً غير متوقع لبرنامج PHP. يتم طرح الاستثناءات بواسطة العديد من وظائف وكلاسات. يمكن للوظائف والكلاسات التي يحددها المستخدم أيضاً طرح استثناءات. تعتبر الاستثناءات طريقة جيدة لإيقاف دالة عندما تصادف بيانات لا يمكنها استخدامها.

إنشاء استثناء

لوظيفة أو طريقة محددة من قبل المستخدم بطرح استثناء **throw** يسمح البيان .. عند طرح استثناء، لن يتم تنفيذ الاكواد التالية له

"Uncaught Exception" إذا لم يتم اكتشاف استثناء، فسيحدث خطأ فادح مع ظهور رسالة "Uncaught Exception".

: لنحاول طرح استثناء دون جلبه

مثال

```
<?php
function Hosini($hos1, $hos2) {
    if($hos2 == 0) {
        throw new Exception("Division by zero");
    }
    return $hos1 / $hos2;
}
```

```
echo Hosini(5, 0);
?>
```

:ستبدو النتيجة كالتالي

Fatal error: Uncaught Exception: Division by zero in C:\

webfolder\test_hosini.htm:4

Stack trace: #0 C:\webfolder\test_hosini.htm(9):

Hosini(5, 0) #1 {main} thrown in C:\webfolder\

test_hosini.htm on line 4

كود المحاولة وجلب الإستثناء او الخطا

العبارة للقبض **try...catch** لتجنب الخطأ من المثال أعلاه، يمكننا استخدام
على الاستثناءات ومتابعة العملية.

```
try {  
    code that can throw exceptions  
} catch(Exception $e) {  
    code that runs when an exception is caught  
}
```

مثال

إظهار رسالة عند طرح خطأ

```
<?php  
function Hosini($hos1, $hos2) {  
    if($hos2 == 0) {  
        throw new Exception("Division by zero");  
    }  
    return $hos1 / $hos2;  
}
```

```
try {  
    echo Hosini(5, 0);  
} catch(Exception $e) {  
    echo "Unable to Hosini."  
}  
?>
```

تشير كتلة الالتقاط إلى نوع الاستثناء الذي يجب اكتشافه واسم المتغير الذي يمكن استخدامه للوصول إلى الاستثناء. في المثال أعلاه، نوع الاستثناء هو **\$e** واسم المتغير هو **Exception**.

محاولة..... أخيراً للكود

يمكن استخدام العبارة للجلب . سيتم دائماً تشغيل **try...catch...finally** الكتلة بغض النظر عما إذا تم اكتشاف استثناء أم **finally** الاكواد الموجودة في الكتلة اختيارية **catch**، كانت موجودة **finally** لا. إذا

```
try {  
    code that can throw exceptions  
} catch(Exception $e) {  
    code that runs when an exception is caught
```

```
} finally {
```

code that always runs regardless of whether an

exception was caught

```
}
```

مثال

إظهار رسالة عند طرح استثناء او خطأ ثم الإشارة إلى انتهاء العملية

```
<?php
function Hosini($hos1, $hos2) {
    if($hos2 == 0) {
        throw new Exception("Division by zero");
    }
    return $hos1 / $hos2;
}
```

```
try {
    echo Hosini(5, 0);
} catch(Exception $e) {
    echo "Unable to Hosini. ";
} finally {
    echo "Process complete.";
}
?>
```

مثال

إخراج نصوص حتى لو لم يتم اكتشاف الاستثناء

```
<?php
function Hosini($hos1, $hos2) {
    if($hos2 == 0) {
        throw new Exception("Division by zero");
    }
    return $hos1 / $hos2;
}

try {
    echo Hosini(5, 0);
} finally {
    echo "Process complete.";
}
?>
```

كائن الاستثناء

يحتوي كائن الاستثناء على معلومات حول الخطأ أو السلوك غير المتوقع الذي واجهته الوظيفة

new Exception(message, code, previous)

قيمه المعامل

| Parameter | وصف |
|-----------|---|
| message | خيارى. نصوص تصف سبب طرح الاستثناء |
| code | خيارى. عدد صحيح يمكن استخدامه لتمييز هذا الاستثناء بسهولة عن الاستثناءات الأخرى من نفس النوع |
| previous | خيارى. إذا تم طرح هذا الاستثناء في كتلة التقاط لاستثناء آخر، فمن المستحسن تمرير هذا الاستثناء إلى هذه المعلمة |

طُرق

عند اكتشاف استثناء، يوضح الجدول التالي بعض الطرق التي يمكن استخدامها للحصول على معلومات حول الاستثناء:

| Method | وصف |
|---------------|--|
| getMessage() | إرجاع نصوص تصف سبب طرح الاستثناء إذا تم تشغيل هذا الاستثناء بواسطة استثناء آخر، فستقوم هذه الطريقة بإرجاع الاستثناء السابق. إذا لم يكن الأمر كذلك، فسيتم عرضه فارغاً |
| getPrevious() | إرجاع كود الاستثناء |
| getCode() | إرجاع المسار الكامل للملف الذي تم طرح الاستثناء فيه |
| getFile() | إرجاع رقم السطر من الاكواد التي طرحت الاستثناء |
| getLine() | |

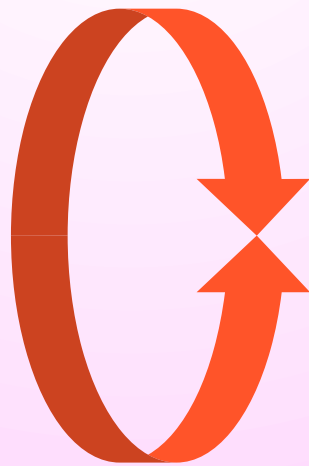
مثال

معلومات الإخراج حول الاستثناء الذي تم طرحه

```
<?php
function Hosini($hos1, $hos2) {
    if($hos2 == 0) {
        throw new Exception("Division by zero", 1);
    }
    return $hos1 / $hos2;
}

try {
    echo Hosini(5, 0);
} catch(Exception $ex) {
    $code = $ex->getCode();
    $message = $ex->getMessage();
    $file = $ex->getFile();
    $line = $ex->getLine();
    echo "Exception thrown in $file on line $line: [Code
    $code]
    $message";
}
?>
```

أبو حبيب الحسيني



214

الكافي في PHP الجزء الأول Abu Habib Al-Husini

التكملة في الجزء الثاني باذن الله تعالى

أبو حبيب الحسيني

215

الكافي في PHP الجزء الاول Abu Habib Al-Husini