الكافى

PHP

الجزء الاول

ابو حبيب الحسينى

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

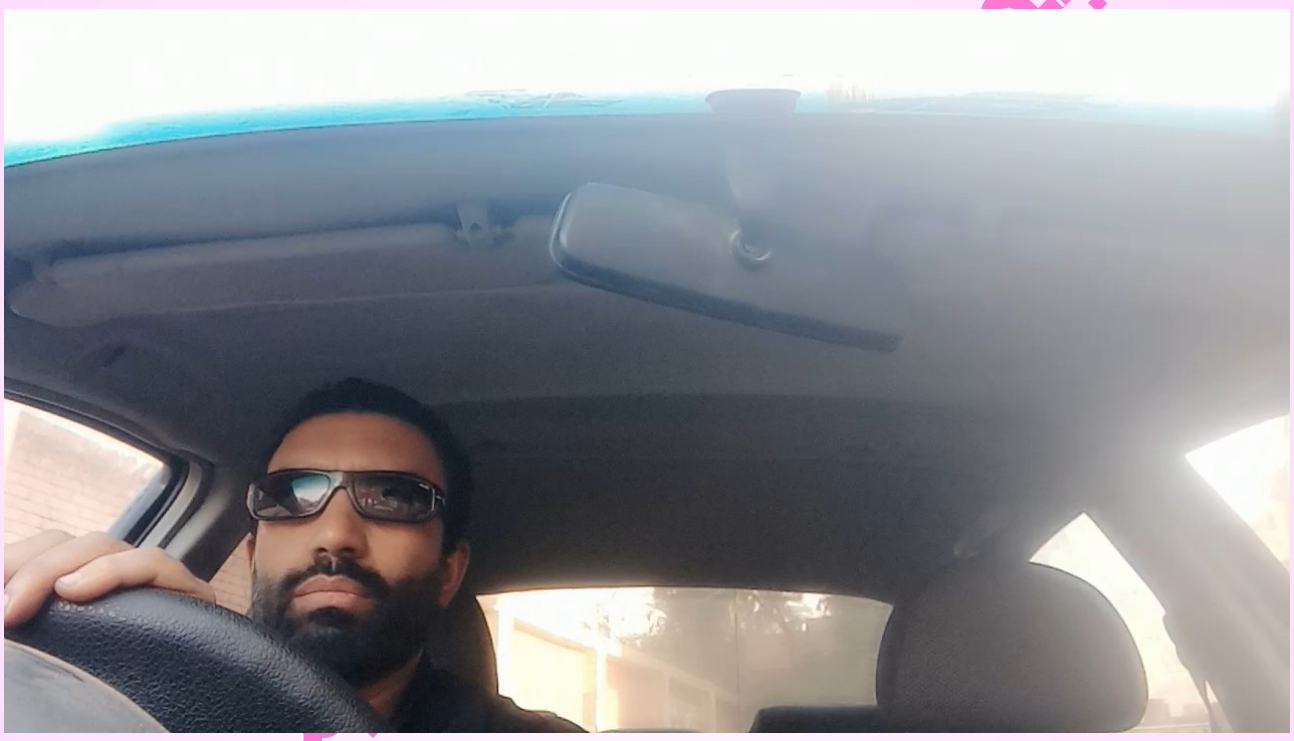
لاحظ هنا ان الجملة اصبحت غير مفهومة الى حد ما او غير مرتبة بشكل صحيح لان بعض الكلمات نقلت من مكانها بسبب وضع كلمات انجليزية وسط الجمل العربية فافى مثل هذة الحالات حاول ان تستنتج الجمله بنفسك وتفهما

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

المرجع العربى السريع

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

ابو حبيب الحسينى



الفهرس

مقدمة سريعة

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

و لن اتحدث كثير عن هذة اللغة لانها معروفة جيدا و يكفيك ان تعرف ان اكبر مواقع التواصل الاجتماعيه العملاقة فى العالم مصنوعةبـــ (بى اتش بى) حيث تقوم بى اتش بى بادارتها بكل سهولة ويسر وهى مواقع شديدة التعقيد مما يدل على قوة هذة اللغة الجبارة وسيطرتها الكاملة فى مجال الويب ويجب ان تعرف

ايضا ان منافسة بى اتش بى 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 بى اتش بى من سنين طويلة حيث مازالت بى اتش بى تسيطرة 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 عل سرعة بايثون فى التقدم والانشار ورغم ذلك فشلت بايثون ايضا فى نزع التاج من بى اتش بى و مازالت بى اتش بى تسيطر على 40 بالمئة من مواقع الانتر نت بالعالم ناهيك عن السيطرة الكامله على انظمة سي ام اس مما يدل على فشل المنافسات الشرسة المستمرة من سنين طويلة امام بى اتش بى بلا اى جدوى او تحقيق اى نتائج تذكر حتى الان ويوجد مميزات اخرى لهذة اللغة ستعرفها فى كتاب المرجع العربى الكامل لدوال الورد بريس سيشرح هذا المرجع مهام عشرات الالاف لدوال واجرائات الوردبريس انشاء وترجمة الذكاء الاصتناعى

مثال على الكود

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

<?php
echo "Abo Habib Alhosini  PHP script!";
?>

</body>
</html>
```


• يتم تنفيذ كود بى اتش بى على الخادم فقط

قبل ان تبدا

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 الاباتشى من الموقع الراسمى وسينصب لك كل احتياجات بى اتش بى دون ان تفعل شى وستجد ارشادات التثبيت على الموقع الراسمى للاباتشى

بناء الجمل الأساسي

يمكن وضع برنامج PHP في أي مكان في المستند النصي.

يبدأ الكود PHP بـ <?php وينتهي بـ ?>:

```
<?php
// PHP code goes here
?>
```

امتداد الملف الافتراضي لملفات PHP هو " _hosini.htm".

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

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

مثال

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

```
<body>

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

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

</body>
</html>
```

ملاحظة: تنتهي عبارات PHP بفاصلة منقوطة (;).

حساسية الحالة

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

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

مثال

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

```php
<?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 $coLOR و$COLOR, $color يتم التعامل مع مختلفة:

مثال

```php
<!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

التعليق في كود PHP هو سطر لا يتم تنفيذه كجزء من البرنامج. والغرض الوحيد 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 تذكرك التعليقات بما كنت تفكر فيه عندما كتبت الكود

تدعم لغة PHP عدة طرق للتعليق:

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، في PHP:

مثال

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


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

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

ملاحظة: على عكس لغات البرمجة الأخرى، ليس لدى PHP أي أمر للإعلان 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 (y و x مثل) يمكن أن يكون للمتغير اسم قصير السيارة، الحجم الإجمالي).

قواعد متغيرات PHP:

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

تذكر أن أسماء متغيرات PHP حساسة لحالة الأحرف!

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

المثال التالي سيخرج مجموع متغيرين:

مثال

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

ملحوظة: سوف تتعلم المزيد عن echo البيان وكيفية إخراج البيانات إلى 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 الصارمة، 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
<?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
<?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 الوظيفة.

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

مثال

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

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

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

HabeB();
```

```php
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
<?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.

اخراج وطباعة البيانات

هناك طريقتان أساسيتان للحصول على الإخراج، PHP مع echo و print.

في كل مثال تقريبًا. لذلك، يحتوي هذا print أو echo في هذا الكتاب نستخدم الفصل على مزيد من المعلومات حول هذين البيانين الناتجين.

اخراج وطباعة الاكواد

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

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

كود اخراج

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

عرض النص

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

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 الاخراج 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";
```

```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 الاكواد

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

PHP يدعم أنواع البيانات التالية:

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

نصوص

النصوص عبارة عن نصوص من الأحرف، مثل "Abo Habib Alhosinii !".

يمكن أن تكون النصوص أي نص داخل علامات الاقتباس. يمكنك استخدام علامات الاقتباس المفردة أو المزدوجة:

مثال

```php
<?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 موجبًا أو سالبًا

- 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 نوع البيانات وقيمتها:

مثال

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

الرقم العشرى

العدد العشرى (رقم النقطة العشرية) هو رقم بفاصلة عشرية أو رقم في شكل أسي.

بإرجاع PHP var_dump() هو عدد عشرى. تقوم الدالة $x في المثال التالي نوع البيانات وقيمتها:

مثال

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

منطقية

تمثل القيمة المنطقية حالتين محتملتين: TRUE أو FALSE.

```
$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.

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

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.

لنفترض أن لدينا كلاس تسمى **Hosese**. يمكن أن تحتوي السيارة على خصائص مثل الطراز واللون وما إلى ذلك. يمكننا تحديد متغيرات مثل **$color** و**$namme** وما إلى ذلك للاحتفاظ بقيم 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
<?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**.

المتغير من نوع البيانات **NULL** هو متغير ليس له قيمة مخصصة له.

نصيحة: إذا تم إنشاء متغير بدون قيمة، فسيتم تعيين قيمة **NULL** له تلقائيًا.

يمكن أيضًا إفراغ المتغيرات عن طريق تعيين القيمة إلى **NULL**:

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()

تقوم الدالة PHP strlen() بإرجاع طول النصوص.

مثال

إرجاع طول النصوص "Abo Habib Alhosinii !":

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

حساب الكلمات في النصوص - str_word_count()

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

مثال

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

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

عكس نصوص - strrev()

تقوم وظيفة PHP strrev() بعكس نصوص.

مثال

عكس الحروف لـــ"ابو حبيب الحسينى!":

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


البحث عن نص داخل نصوص - strpos ()

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

مثال

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

```php
<?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()

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

مثال

استبدال نص "حبيب" بكلمة "على":

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

أكمل مرجع النصوص

يحتوي مرجع نصوص PHP على وصف ومثال للاستخدام لكل وظيفة!

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.

لذلك، إذا قمت بتعيين قيمة عددية لمتغير، فسيكون نوع هذا المتغير عددًا صحيحًا تلقائيًا. وبعد ذلك، إذا قمت بتعيين نصوص إلى نفس المتغير، فسيتغير 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 كعشرى، لأن أحد المعاملات هو عشرى (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 10)، سداسي عشري (معتمد على 16 - مسبوق بـ 0x) أو ثماني (معتمد على 8 - مسبوق بـ 0)

تحتوي PHP على الثوابت المحددة مسبقًا التالية للأعداد الصحيحة:

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

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

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

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

الارقام العشرية

العدد العشرى هو رقم ذو علامة عشرية أو رقم في شكل أسي.

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

يمكن لنوع البيانات العشرية عادةً تخزين قيمة تصل إلى 1.7976931348623E+308 (يعتمد على النظام الأساسي)، وبدقة قصوى تصل إلى 14 رقمًا.

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

- PHP_FLOAT_MAX - أكبر رقم عشرى 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 - أصغر رقم موجب يمكن تمثيله، x بحيث يكون x + 1.0 != 1.0

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

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

مثال

تحقق مما إذا كان نوع المتغير رقم عشريا:

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

إنفينيتي

القيمة الرقمية الأكبر من PHP_FLOAT_MAX تعتبر لا نهائية.

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

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

مثال

تحقق مما إذا كانت القيمة الرقمية محدودة أم لا نهائية:

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

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

نان

NaN تعني ليس رقمًا.

NaN يستخدم للعمليات الرياضية المستحيلة.

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

- is_nan()

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

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));
?>
```

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

التحويل

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

مثال

إرسال عشرى 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

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

وظيفة pi()

ترجع الدالة pi() قيمة PI:

مثال

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

وظائف min() و max()

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

مثال

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

وظيفة abs()

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

مثال

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

وظيفة sqrt()

تقوم sqrt() الدالة بإرجاع الجذر التربيعي للرقم:

مثال

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

تقريب الإرقام ذو الفاصلة العشرية

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

مثال

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

أرقام عشوائية

تقوم rand() الدالة بإنشاء رقم عشوائي:

مثال

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

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

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

مثال

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

استكمال مرجع الرياضيات

يحتوي مرجع **PHP** على وصف ومثال للاستخدام لكل وظيفة الرياضي.

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

مثال

قم بإنشاء ثابت باسم غير حساس لحالة الأحرف :

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

المصفوفات الثابتة

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

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

function HabeB() {
  echo Hoseni;
}

HabeB();
?>
```

المشغلات

يتم استخدام العوامل لإجراء العمليات على المتغيرات والقيم.

تقسم PHP العوامل إلى المجموعات التالية:

- 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	مجموع $x و $y
-	الطرح عمليه	$x - $y	الفرق بين $x و$y
*	الضرب	$x * $y	منتج $x و $y
/	قسم	$x / $y	حاصل قسمة $x و $y
%	معامل	$x % $y	باقي $x مقسومًا على $y
**	الأس	$x ** $y	نتيجة رفع $x إلى قوة $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	عمليه الضرب

x /= y	x = x / y	قسم
x %= y	x = x % y	معامل

مشغلي المقارنة

تُستخدم عوامل المقارنة PHP (رقم أو نصوص) لمقارنة قيمتين:

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

يتم استخدام عوامل زيادة PHP لزيادة قيمة المتغير.

يتم استخدام عوامل إنقاص PHP لتقليل قيمة المتغير.

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

يتم استخدام عوامل PHP المنطقية لدمج العبارات الشرطية.

Operator	اسم	Example	نتيجة
and	و	$x and $y	صحيحين $y و$x صحيح إذا كان كل من
or	أو	$x or $y	صحيحًا $y أو $x صحيح إذا كان
xor	او	$x xor $y	حيحًا، ولكن ليس كليهما $y أو $x صحيح إذا كان
&&	و	$x && $y	صحيحين $y و$x صحيح 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	تجميع $Hosini1 و$Hosini2
.=	Concatenation assignment	$Hosini1 .= $Hosini2	اضافة $Hosini2 بـ $Hosini1

مشغلي مصفوفة

لمقارنة المصفوفات PHP يتم استخدام عوامل تشغيل المصفوفات.

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

---

مشغلي التعيين الشرطي

تُستخدم عوامل التعيين الشرطية في PHP لتعيين قيمة وفقًا للشروط:

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:

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

if كود

if ينفذ البيان بعض الاكواد إذا كان شرط واحد صحيحا.

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

مثال

أقل من 20 (HOUR): الإخراج "أتمنى لك يومًا سعيدًا!" إذا كان الوقت الحالي

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

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

- كود if...else

بعض الاكواد إذا كان الشرط صحيحًا ورمزًا آخر إذا كان **if...else** ينفذ البيان الشرط خاطئًا.

مثال

الإخراج "أتمنى لك يومًا سعيدًا!" إذا كان الوقت الحالي أقل من 20، و"أتمنى لك ليلة سعيدة!" خلاف ذلك:

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

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

- كود if...elseif...else

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

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
<?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 الحالة. يُستخدم **break** لمنع تشغيل الكود في الحالة التالية تلقائيًا. **default** يتم استخدام العبارة إذا لم يتم العثور على تطابق .

مثال

```php
<?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، بدلًا 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.

في PHP، لدينا أنواع الحلقات التالية:

- 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 الحلقة while 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
<?php
$x = 1;

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

شاهد المثال

- وقم بتعيين قيمة البداية على ،($x) س = 1; - قم بتهيئة عداد الحلقة$ 1
- أقل من أو يساوي $x 5 تابع الحلقة طالما أن - $x <= 5
- 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

- وقم بتعيين قيمة البداية على ،($x) س = 0; - قم بتهيئة عداد الحلقة$ 0
- أقل من أو يساوي $x 100 تابع الحلقة طالما أن - $x <= 100
- 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 هذة 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 **.($x = 1)** يقوم المثال أدناه أولاً بتعيين المتغير **$x** إلى **1** بـ **1.** ثم يتم التحقق **$x** بعض المخرجات، ثم تزيد المتغير **do while** حلقة وستستمر الحلقة في العمل طالما ،(أقل من أو يساوي **$x** ؟**5** هل) من الشرط **$x** أقل من أو يساوي **5**:

مثال

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

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

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

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

مثال

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

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

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

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

حلقة فور

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

حدود:

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

أمثلة

يعرض المثال أدناه الأرقام من 0 إلى 10:

مثال

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

شاهد المثال

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

مثال

```php
<?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 $0
- أقل من أو يساوي 100 $x تابع الحلقة طالما أن - $x <= 100;
- زيادة قيمة عداد الحلقة بمقدار 10 لكل تكرار - $x+=10

حلقة foreach

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

حلقة foreach

تعمل الحلقة foreach فقط على المصفوفات، وتُستخدم للتكرار خلال كل زوج 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

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

مثال

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

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

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

مثال

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

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

كلمة بريك

العبارة المستخدمة في فصل سابق من هذا الكتاب break لقد رأيت بالفعل switch البيان. 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

تكرارًا 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: الحلقاتbreak 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 مضمنة

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

الوظائف المحددة من قبل المستخدم

إلى جانب وظائف PHP الخاصة، من الممكن إنشاء وظائفك الخاصة.

- 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

يبدأ إعلان الدالة المعرفة من قبل المستخدم بالكلمة function:

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

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

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

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

مثال

```php
<?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 فى 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.

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

مثال

```php
<?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 يكون هذا في السطر الأول من ملف PHP.

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

مثال

```php
<?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 الدالة Hosini_Height() بدون وسيطات فإنها تأخذ القيمة الافتراضية كوسيطة:

مثال

```php
<?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
<?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 . أيضًا 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
<?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
<?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 البرامتراط 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() في لغة PHP، :

```
array();
```

هناك ثلاثة أنواع من المصفوفات في PHP، :

- 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

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

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

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 مفهرس

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

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، ألقِ 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)
);
```

الآن، تحتوي المصفوفة $habib_Array ثنائية الأبعاد على أربع مصفوفات، 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 أخرىfor يمكننا أيضًا وضع المصفوفة $habib_Array (لا يزال يتعين علينا الإشارة إلى المؤشرين):

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، تنازليًا أو تصاعديًا.

وظائف الفرز للمصفوفات -

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

- **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

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

مثال

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

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

مثال

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

بترتيب رقمي تنازلي $numbers يقوم المثال التالي بفرز عناصر المصفوفة:

مثال

```php
<?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، وفقًا للقيمة:

مثال

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

- فرز المصفوفة (ترتيب تصاعدي)، وفقًا للمفتاح ksort()

يقوم المثال التالي بفرز مصفوفة ترابطية بترتيب تصاعدي، وفقًا للمفتاح:

مثال

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

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

يقوم المثال التالي بفرز مصفوفة ترابطية بترتيب تنازلي، وفقًا للقيمة:

مثال

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

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

يقوم المثال التالي بفرز مصفوفة ترابطية بترتيب تنازلي، وفقًا للمفتاح:

مثال

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

متغيرات العامة - Superglobals

مما يعني ،**"superglobals"** هي **PHP** بعض المتغيرات المحددة مسبقًا في 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.

المتغيرات **PHP superglobal** هي:

- $_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

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

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

$GLOBALS: يوضح المثال أدناه كيفية استخدام المتغير الشامل الفائق

مثال

```php
<?php
$x = 75;
$y = 25;

function Hosini_addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
```

```
}

Hosini_addition();
echo $z;
?>
```

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

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

$_SERVER

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

$_SERVER: يوضح المثال أدناه كيفية استخدام بعض العناصر في

مثال

```php
<?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_Hosini_user_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```


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

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

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

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

المضيف الظاهري الذي تمت إضافته إلى الصفحات التي أنشأها الخادم	RE']
إرجاع المسار المستند إلى نظام الملفات إلى الكود الحالي	$_SERVER['PATH_TRANSLATED']
إرجاع مسار الكود الحالي	$_SERVER['SCRIPT_NAME']
إرجاع URI للصفحة الحالية	$_SERVER['SCRIPT_URI']

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

$_REQUEST

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

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

مثال

```
<html>
<body>

<form method="post" action="<?php echo $_SERVE
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 PHP $_POST أيضًا $_POST يُستخدم "post" = باستخدام الطريقة HTML إرسال نموذج على نطاق واسع لتمرير المتغيرات.

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

مثال

```
<html>
<body>

<form method="post" action="<?php echo $_SERVE
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 = $_POST['fname'];
  if (empty($Hosini_name)) {
    echo "Name is empty";
  } else {
    echo $Hosini_name;
  }
}
?>

</body>
</html>
```

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

$_GET[]

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

يمكن لـ $_GET أيضًا جمع البيانات المرسلة في عنوان URL.

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

```
<html>
<body>
<a href="test_get_hosini.htm?
subject=PHP&web=Abo Habib AlHosini.com">Test
$GET</a>
</body>
</html>
```

يتم إرسال المعلمات، "$GET " عندما ينقر المستخدم على الرابط "test_get_hosini.htm"، ويمكنك بعد ذلك الوصول إلى قيمها في "test_get_hosini.htm" باستخدام $_GET.

يوضح المثال أدناه الكود الموجود في "test_get_hosini.htm":

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 حرفًا واحدًا أو نمطًا أكثر تعقيدًا.

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 و0 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 preg_match() ما إذا كانت النصوص تحتوي على تطابقات للنمط .

مثال

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

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

---

preg_match_all() باستخدام

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

مثال

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

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

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

---

preg_replace() باستخدام

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

مثال

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

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

معدّلات التعبير العادي

يمكن للمعدلات 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	تمكين المطابقة الصحيحة للأنماط المشفرة UTF-8

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:

Metach aracter	وصف
|	ابحث عن تطابق لأي من الأنماط المفصولة بـ | كما في: قطة|كلب|سمكة
.	ابحث عن مثيل واحد فقط لأي شخصية
^	يجد تطابقًا كبداية نصوص كما في: ^Hello
$	يجد تطابقًا في نهاية النصوص كما في: Habib$
\d	ابحث عن رقم
\s	ابحث عن حرف مسافة بيضاء

\b	ابحث عن تطابق في بداية كلمة مثل هذه: \bWORD، أو في نهاية كلمة مثل هذه: WORD\b
\uxxxx	ابحث عن حرف Unicode المحدد بالرقم السداسي العشري xxxx

محددو الكمية

تحدد الكميات الكميات:

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

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: $HOsine = '/\?+/';

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 : متبوعًا 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 -

بسيطًا 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>
```


عندما يقوم المستخدم بملء النموذج أعلاه والنقر فوق زر الإرسال، يتم إرسال بيانات النموذج للمعالجة إلى ملف PHP يسمى **"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

يمكن أيضًا تحقيق نفس النتيجة باستخدام طريقة **HTTP GET:**

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>
```


ويبدو **"welcome_get_hosini.htm"** كما يلي:

```
<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.

فكر في الأمان عند معالجة نماذج PHP!

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

يقوم كل من GET وPOST بإنشاء مصفوفة (على سبيل المثال، 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 $_POST. و$_GET على أنهما POST وGET 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.

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

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

متى تستخدم GET؟

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

يمكن استخدام GET لإرسال بيانات غير حساسة.

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

فكر في الأمان عند معالجة نماذج **PHP**!

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، وزر 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	خياري. إذا كان موجودًا، فيجب أن يحتوي على عنوان URL صالح
Comment	خياري. حقل إدخال متعدد الأسطر (منطقة النص)
Gender	مطلوب. يجب اختيار واحد

أولاً سنلقي نظرة على كود HTML العادي للنموذج:

حقول النص

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

```
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

للنموذج كما يلي HTML يبدو كود:

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

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

ما هو المتغير $_SERVER["PHP_SELF"]؟

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

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.

ما هي وظيفة htmlspecialchars()؟

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

ملاحظة كبيرة حول Form Security

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

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

أحد أنواع الثغرات الأمنية في (XSS) تعد البرمجة النصية عبر المواقع المهاجمين من XSS الكمبيوتر والتي توجد عادةً في تطبيقات الويب. يمكّن 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 $_SERVE
R["PHP_SELF"];?>">
```

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

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

حتى الان جيدة جدا.

ومع ذلك، ضع في اعتبارك أن المستخدم يُدخل عنوان URL التالي في شريط 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 تنفيذ كود JavaScript (سيظهر للمستخدم مربع تنبيه). هذا مجرد مثال بسيط وغير ضار لكيفية استغلال متغير PHP_SELF.

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 htmlspecialch
ars($_SERVER["PHP_SELF"]);?>">
```

بتحويل الأحرف الخاصة إلى كيانات htmlspecialchars() تقوم الدالة HTML. الآن إذا حاول المستخدم استغلال المتغير PHP_SELF، فسيؤدي 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 PHP الخاصة بـ htmlspecialchars().

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 آمنًا ليتم عرضه على الصفحة أو داخل البريد الإلكتروني.

سنقوم أيضًا بأمرين آخرين عندما يرسل المستخدم النموذج:

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

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

سوف نقوم بتسمية الدالة test_input().

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;
}
?>
```

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

ومع ذلك، في 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 في الصفحة السابقة، نرى أن حقول "الاسم" و"البريد الإلكتروني" و"الجنس" مطلوبة. لا يمكن أن تكون هذه الحقول فارغة ويجب ملؤها في نموذج HTML.

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	خياري. إذا كان موجودًا، فيجب أن يحتوي على عنوان URL صالح
Comment	خياري. حقل إدخال متعدد الأسطر (منطقة النص)
Gender	مطلوب. يجب اختيار واحد

في الفصل السابق، كانت جميع حقول الإدخال اختيارية.

في الكود التالي أضفنا بعض المتغيرات الجديدة: **$Hosini_nameErr**، **$Hosini_emailErr**، **$Hosini_genderErr**، و **$Hosini_websiteErr**. ستحتوي متغيرات الخطأ هذه على رسائل خطأ يتحقق **$_POST** بيانًا لكل متغير **if else** للحقول المطلوبة. لقد أضفنا أيضًا وظيفة **empty()** (مع **PHP**). إذا كان المتغير **$_POST** هذا مما إذا كان المتغير فارغًا كانت فارغة، فسيتم تخزين رسالة خطأ في متغيرات الخطأ المختلفة، وإذا لم تكن فارغة، فسيتم إرسال بيانات إدخال المستخدم من خلال الوظيفة **test_input()** :

```
<?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  = "";
```

```php
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 htmlspecialc
hars($_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>
```


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 موقع الويب على عنوان صالح؟".

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

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";
}
```

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

التحقق من صحة البريد الإلكتروني -

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

في الكود أدناه، إذا لم يكن عنوان البريد الإلكتروني منسقًا 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";
}
```

- التحقق من صحة عنوان URL

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

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


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، 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 $Hosini_email و$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 ec
ho $Hosini_name;?>">

E-mail: <input type="text" name="email" value="<?php e
cho $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

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

وظيفة Date()

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

date(*format,timestamp*)

Parameter	وصف
format	مطلوب. يحدد تنسيق الطابع الزمني
timestamp	خياري. 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 (حرف صغير 'L') - يمثل يوم الأسبوع

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

يقوم المثال أدناه بتنسيق تاريخ اليوم بثلاث طرق مختلفة:

مثال

```php
<?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

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

مثال

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

احصل على الوقت

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

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

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

يقوم المثال أدناه بإخراج الوقت الحالي بالتنسيق المحدد:

مثال

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

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

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.

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

مثال

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

---

mktime() إنشاء وقت مع

طابعًازمنيًا. 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 الوظيفة **mktime()**:

مثال

```php
<?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 PHP strtotime() يتم استخدام وظيفة عدد الثواني منذ 1 يناير 1970 (Unix بواسطة الإنسان إلى طابع زمني بتوقيت جرينتش 00:00:00).

strtotime(time, now)

يقوم المثال أدناه بإنشاء التاريخ والوقت من الوظيفة strtotime():

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
<?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() ومع ذلك وضعتها هناك.

المزيد من أمثلة التاريخ

يوضح المثال أدناه تواريخ أيام السبت الستة التالية:

مثال

```php
<?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 مفيدًا جدًا عندما تريد تضمين نفس لغة أو النص في صفحات متعددة بموقع الويب.

تتضمن الأكواد

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

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

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

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " Abo Habib AlHosini.com</p>";
?>
```

لتضمين ملف التذييل في الصفحة، استخدم العبارة include:

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>
```

مثال 2

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

```
<?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 ** ** نحن نستخدم عنصر) هذا. وإليك كيفية القيام بذلك القائمة بسهولة باستخدام **CSS** لاحقًا):

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>
```

مثال 3

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

```
<?php
$color='black';
```

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

?>
```

بعد ذلك، إذا قمنا بتضمين ملف "vars_hosini.htm"، فيمكن استخدام 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>
```


---

require

يتم استخدام العبارة require أيضًا لتضمين ملف في كود PHP.

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

مثال

```
<html>
<body>

<h1>Welcome to Habib My home page!</h1>
<?php include 'noFileExists_hosini.htm';
echo "I have  noFileExists_ $Hosese.";
?>

</body>
</html>
```


إذا قمنا بنفس المثال باستخدام require عبارة echo العبارة، فلن يتم تنفيذ
أعادت العبارة خطأ فادحًا 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>
```

يُستخدم require عندما يكون الملف مطلوبًا بواسطة التطبيق.

يُستخدم include عندما لا يكون الملف مطلوبًا ويجب أن يستمر التطبيق عند عدم العثور على الملف.

تعد معالجة الملفات جزءًا مهمًا من أي تطبيق ويب. غالبًا ما تحتاج إلى فتح ملف ومعالجته لمهام مختلفة.

التعامل مع الملفات

العديد من الوظائف لإنشاء الملفات وقراءتها وتحميلها وتحريرها PHP لدى.

كن حذرا عند التعامل مع الملفات!

عندما تقوم بمعالجة الملفات يجب أن تكون حذراً للغاية.

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

وظيفة ملف القراءة .

تقوم الوظيفة readfile() بقراءة الملف وكتابته في المخزن المؤقت للإخراج.

لنفترض أن لدينا ملفًا نصيًا يسمى "Hosini_file.txt"، مخزّنًا على الخادم، ويبدو كالتالي:

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

مثال

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


readfile() مفيدة إذا كان كل ما تريد فعله هو فتح ملف وقراءة 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 خيارات 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 رسالة إذا كانت الدالة fopen() غير قادرة على فتح الملف المحدد:

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);
?>
```

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

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 موجودا. 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	إنشاء ملف جديد للكتابة فقط. إرجاع FALSE وخطأ إذا كان الملف موجودًا 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 موجودا. 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+	إنشاء ملف جديد للقراءة/الكتابة. إرجاع FALSE وخطأ إذا كان الملف موجودًا بالفعل

fread() - ملف قراءة

تقرأ الدالة fread() من ملف مفتوح.

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

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

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

---

fclose() - ملف إغلاق

يتم استخدام الوظيفة fclose() لإغلاق ملف مفتوح.

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

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

```
<?php
$Habib_file = fopen("Hosini_file.txt", "r");
// some code to be executed....
fclose($Habib_file);
?>
```

---

fgets () - قراءة سطر واحد

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

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

مثال

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


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

---

feof() - التحقق من نهاية الملف

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

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

يقرأ المثال أدناه ملف "Hosini_file.txt" سطرًا تلو الآخر، حتى يتم الوصول 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);
?>
```


---

fgetc() - قراءة حرف واحد

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

يقرأ المثال أدناه ملف "Hosini_file.txt" حرفًا تلو الآخر، حتى يتم الوصول 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);
?>
```

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

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

---

fopen () - إنشاء ملف

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

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

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

مثال

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

أذونات الملف

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

---

fwrite() - الكتابة إلى الملف

يتم استخدام الوظيفة fwrite() للكتابة إلى ملف.

fwrite() تحتوي المعلمة الأولى على اسم الملف المراد الكتابة إليه والمعلمة 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 "habib Doe" والثانية تحتوي على "Hosini Doe". وبعد أن انتهينا من الكتابة قمنا بإغلاق الملف باستخدام الدالة fclose().

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 ملفًا موجودًا 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);
?>
```

فسيختفي 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 "w" النص بنهاية الملف، بينما يقوم الوضع بتجاوز (ويمسح) المحتوى القديم 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، مع PHP.

ومع ذلك، مع السهولة يأتي الخطر، لذا كن حذرًا دائمًا عند السماح بتحميل الملفات!

التعديل على ملف التكوين الأساسي .ini.

أولاً، تأكد من تكوين PHP للسماح بتحميل الملفات.

ابحث عن file_uploads في ملف "php.ini"، التوجيه،الى ملف التكوين بى واضبطه على "تشغيل اتش بى":

```
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" enct
ype="multipart/form-data">
  Select image to upload:
  <input type="file" name="fileToUpload" id="fileToUpl
oad">
  <input type="submit" value="Upload
Image" name="submit">
</form>

</body>
</html>
```


أعلاه HTML بعض القواعد التي يجب اتباعها لنموذج:

- تأكد من أن النموذج يستخدم الطريقة = "post"
- يحتاج النموذج أيضًا إلى السمة التالية: enctype = "multipart/form-data". 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_EXTENSIO
N));
// 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;
  }
}
?>
```

شرح الكود PHP:

- يحدد الدليل الذي سيتم وضع الملف - $Hosini_dir = "uploads/" فيه
- يحدد $Hosini_file مسار الملف الذي سيتم تحميله
- $Hosini_uploadOk=1 لم يتم استخدامه بعد (سيتم استخدامه لاحقًا)
- يحتفظ $Hosini_imageFileType بامتداد الملف (بالأحرف 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 به ملف "upload_hosini.htm". سيتم حفظ الملفات التي تم تحميلها هناك.

تحقق مما إذا كان الملف موجودًا بالفعل

الآن يمكننا إضافة بعض القيود.

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

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

الحد من حجم الملف

أعلاه باسم HTML يُسمى حقل إدخال الملف في نموذج **"fileToUpload".**

الآن نريد التحقق من حجم الملف. إذا كان حجم الملف أكبر من **500** كيلو بايت، فسيتم عرض رسالة خطأ، ويتم تعيين **$Hosini_uploadOk** على **0**:

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

الحد من نوع الملف

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

يبدو الآن ملف "upload_hosini.htm" الكامل كما يلي:

```
<?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;
}
```

```php
// 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 الإرتباط؟

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 الإرتباط باستخدام

يتم إنشاء ملف تعريف الارتباط باستخدام الوظيفة setcookie().

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

مطلوب معلمة الاسم فقط . جميع المعلمات الأخرى اختيارية.

إنشاء/استرداد ملف تعريف الإرتباط

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

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

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>
```


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

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

تعديل قيمة ملف تعريف الإرتباط

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 الإرتباط

ذات تاريخ انتهاء setcookie() لحذف ملف تعريف الارتباط، استخدم الوظيفة الصلاحية في الماضي:

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 الوظيفة setcookie()، ثم احسب متغير المصفوفة $_COOKIE:

مثال

```php
<?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>
```

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 أنت أو ماذا تفعل، لأن عنوان HTTP لا يحتفظ بحالته.

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

تبدأ الجلسة بالوظيفة session_start().

يتم تعيين متغيرات الجلسة باستخدام المتغير العام PHP: $_SESSION.

الآن، لنقم بإنشاء صفحة جديدة تسمى "demo_session1_hosini.htm". في هذه الصفحة، نبدأ جلسة PHP جديدة ونقوم بتعيين بعض متغيرات الجلسة:

مثال

```php
<?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 في المستند . قبلsession_start()ملاحظة: يجب أن تكون الوظيفة أي علامات HTML.

احصل على القيم المتغيرة لجلسة

بعد ذلك، نقوم بإنشاء صفحة أخرى تسمى "demo_session2_hosini.htm". من هذه الصفحة، 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()).

لاحظ أيضًا أن جميع قيم متغيرات الجلسة مخزنة في المتغير $_SESSION العام:

مثال

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

```php
<?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>
```


هناك طريقة أخرى لإظهار كافة قيم متغيرات الجلسة لجلسة المستخدم وهي تشغيل الكود التالي:

مثال

```php
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
?>
```

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


كيف يعمل؟ كيف يعرف أنه أنا؟

تقوم معظم الجلسات بتعيين مفتاح مستخدم على كمبيوتر المستخدم والذي يبدو كالتالي: **765487cf34ert8dede5a562e4f3a7e12.** وبعد ذلك، عند فتح جلسة على صفحة أخرى، تقوم بفحص الكمبيوتر بحثًا 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
```

```php
// to change a session variable, just overwrite it
$_SESSION["Hosini_user"] = "arabic";
print_r($_SESSION);
?>

</body>
</html>
```

حذف جلسة

لإزالة كافة متغيرات الجلسة العامة وحذف الجلسة، استخدم session_destroy() و session_unset():

مثال

```php
<?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.

الفلترة

تُستخدم فلاتر PHP للتحقق من صحة المدخلات الخارجية وتصحيح ها.

يحتوي ملحق فلتر PHP على العديد من الوظائف اللازمة للتحقق من مدخلات المستخدم، وهو مصمم لجعل التحقق من صحة البيانات أسهل وأسرع.

يمكن استخدام الوظيفة filter_list() لسرد ما يقدمه فلتر PHP:

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()

تعمل الوظيفة filter_var() على التحقق من صحة البيانات وتصحيح ها.

تقوم الدالة filter_var() بتصفية متغير واحد باستخدام فلتر محدد. يستغرق قطعتين من البيانات:

- المتغير الذي تريد التحقق منه
- نوع الشيك المستخدم

تطهير نصوص

يستخدم المثال التالي الوظيفة filter_var() لإزالة كافة علامات HTML من نصوص:

مثال

```
<?php
$habebe = "<h1>Abo Habib Alhosinii !</h1>";
$newstr = filter_var($habebe,
FILTER_SANITIZE_STRING);
echo $newstr;
?>
```

التحقق من صحة عدد صحيح

الدالة filter_var() يستخدم المثال التالي للتحقق مما إذا كان المتغير $int عددًا صحيحًا، فسيكون مخرج الكود أدناه: "عدد $int عددًا صحيحًا. إذا كان $int 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;
```

```php
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");
}
?>
```

التحقق من صحة عنوان IP

يستخدم المثال التالي الوظيفة filter_var() للتحقق مما إذا كان المتغير $ip صالح IP هو عنوان:

مثال

```php
<?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");
}
?>
```

تصحيح والتحقق من صحة عنوان URL

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
<?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 وبين 1 و200، النوع INT:

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");
}
?>
```

التحقق من صحة عنوان IPv6

للتحقق مما إذا كان المتغير $ip يستخدم المثال التالي الوظيفة filter_var() 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_REQUIblack) === false) {
```

```
  echo("$url is a valid URL with a query string");
} else {
  echo("$url is not a valid URL with a query string");
}
?>
```

قم بإزالة الأحرف ذات قيمة ASCII> 127

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 كوسيطة 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

استخدم دالة مجهولة كرد اتصال array_map() لوظيفة PHP:

```
<?php
$hoshos = ["Hamza", "alee", "habib", "osama"];
$lengths = array_map( function($item)
{ return strlen($item); } , $hoshos);
print_r($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
<?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، يكود JSON إلى لتخزين البيانات وتبادلها.

هو تنسيق قائم على النص، فيمكن إرساله بسهولة من وإلى JSON نظرًا لأن تنسيق الخادم واستخدامه كتنسيق بيانات بواسطة أي لغة برمجة.

الطرق لجسون

بعض الوظائف المضمنة للتعامل مع PHP لدى JSON.

أولاً، سننظر في الوظيفتين التاليتين:

- json_encode()
- json_decode()

- json_encode()

يتم استخدام الدالة json_encode() لترميز قيمة بتنسيق JSON.

مثال

يوضح هذا المثال كيفية تشفير مصفوفة ترابطية في كائن JSON:

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

```
echo json_encode($Hosiny);
?>
```


مثال

يوضح هذا المثال كيفية تحويل مصفوفة مفهرسة في مصفوفة JSON:

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

echo json_encode($habib_Array);
?>
```

- json_decode()

يتم استخدام الدالة json_decode() لفك تشفير كائن JSON إلى كائن PHP أو مصفوفة اقترانية.

مثال

يقوم هذا المثال بفك تشفير بيانات JSON إلى كائن PHP:

```php
<?php
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';

var_dump(json_decode($jsonobj));
?>
```


تقوم الدالة **json_decode ()** بإرجاع كائن بشكل افتراضي. تحتوي الدالة **json_decode()** على معلمة ثانية، وعند ضبطها على القيمة **true**، يتم فك تشفير كائنات **JSON** إلى صفائف ترابطية.

مثال

يقوم هذا المثال بفك ترميز بيانات **JSON** إلى مصفوفة ترابطية **PHP:**

```php
<?php
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';

var_dump(json_decode($jsonobj, true));
?>
```

الوصول إلى القيم التي تم فك تشفيرها -

فيما يلي مثالان لكيفية الوصول إلى القيم التي تم فك تشفيرها من كائن ومن مصفوفة اقترانية:

مثال

يوضح هذا المثال كيفية الوصول إلى القيم من كائن PHP:

```
<?php
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';

$obj = json_decode($jsonobj);

echo $obj->Habib;
echo $obj->Mahmedd;
echo $obj->osman;
?>
```


مثال

يوضح هذا المثال كيفية الوصول إلى القيم من مصفوفة ترابطية PHP:

```
<?php
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';

$arr = json_decode($jsonobj, true);

echo $arr["Habib"];
echo $arr["Mahmedd"];
```

```
echo $arr["osman"];
?>
```

التكرار على جسون

يمكنك أيضًا تكرار القيم باستخدام حلقة foreach() :

مثال

يوضح هذا المثال كيفية التكرار عبر قيم كائن PHP:

```
<?php
$Hosini_json= '{"Habib":35,"Mahmedd":37,"osman":43}';

$obj = json_decode($jsonobj);

foreach($obj as $key => $value) {
  echo $key . " => " . $value . "<br>";
}
?>
```


مثال

يوضح هذا المثال كيفية التكرار عبر قيم مصفوفة الترابطية:

```php
<?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 الإستثناء؟

الاستثناء هو كائن يصف خطأ أو سلوكًا غير متوقع لبرنامج PHP النصي.

يتم طرح الاستثناءات بواسطة العديد من وظائف وكلاسات PHP.

يمكن للوظائف والكلاسات التي يحددها المستخدم أيضًا طرح استثناءات.

تعتبر الاستثناءات طريقة جيدة لإيقاف دالة عندما تصادف بيانات لا يمكنها استخدامها.

انشاء استثناء

لوظيفة أو طريقة محددة من قبل المستخدم بطرح استثناء throw يسمح البيان
.. عند طرح استثناء، لن يتم تنفيذ الاكواد التالية له

إذا لم يتم اكتشاف استثناء، فسيحدث خطأ فادح مع ظهور رسالة **"Uncaught Exception".**

لنحاول طرح استثناء دون جلبة :

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 الاستثناء 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 خطا:

```
<?php
function Hosini($hos1, $hos2) {
  if($hos2 == 0) {
    throw new Exception("Division by zero");
  }
  return $hos1 / $hos2;
}
```

```php
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 Exception واسم المتغير هو $e.

محاولة.....أخيرًا للكود

يمكن استخدام العبارة للجلب . سيتم دائمًا 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

```php
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 استثناءاو خطا ثم الإشارة إلى انتهاء العملية:

```php
<?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
<?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 الإستثناء

يحتوي كائن الاستثناء على معلومات حول الخطأ أو السلوك غير المتوقع الذي واجهته الوظيفة

```
new Exception(message, code, previous)
```

قيمه المعامل

Parameter	وصف
message	خياري. نصوص تصف سبب طرح الاستثناء
code	خياري. عدد صحيح يمكن استخدامه لتمييز هذا الاستثناء بسهولة عن الاستثناءات الأخرى من نفس النوع
previous	خياري. 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
getPrevious()	إذا تم تشغيل هذا الاستثناء بواسطة استثناء آخر، فستقوم هذه الطريقة بإرجاع الاستثناء السابق. إذا لم يكن الأمر كذلك، فسيتم عرضه فارغًا
getCode()	إرجاع كود الاستثناء
getFile()	إرجاع المسار الكامل للملف الذي تم طرح الاستثناء فيه
getLine()	إرجاع رقم السطر من الاكواد التي طرحت الاستثناء

مثال

معلومات الإخراج حول الاستثناء الذي تم طرحه:

```php
<?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";
}
?>
```

ابو حبيب الحسينى

التكملة فى الجزء الثانى باذن الله تعالى