

# لغة ( PHP )

تتميز لغة PHP بالكثير من الخصائص التي جعلتها الخيار الأمثل لمبرمجي الويب في العالم :

## السهولة

تعتبر لغة PHP من أسهل لغات البرمجة تعلمًا، فهي تريحك من جميع تعقيدات إدارة الذاكرة وتعقيدات معالجة النصوص الموجودة في C من جهة، والكثير من الضعف الموجود في بنية وتصميم لغة البرمجة Perl من جهة أخرى.

تمتلك لغة PHP بنية وقواعد ثابتة وواضحة جدًا، معظم قواعد اللغة مأخوذة من كل من C و Java و Perl لصنع لغة برمجة عالية السهولة والسلاسة دون فقدان أي من القوة في اللغة، يفيدك ذلك إذا كنت تعلم أي شيء عن لغات البرمجة الأخرى مثل Visual Basic أو C أو Java حيث ستجد دائمًا بأنك تفهم مواد الدورة بسرعة، وستكتشف كيف تقوم PHP بتسهيل أصعب الأمور وإذلال العقبات التي تواجه المبرمج حتى يتفرغ تمامًا للإبداع فقط، كل ما تفكر به تستطيع تنفيذه بلغة PHP.

## السرعة

لغة PHP من اللغات المعروفة بسرعتها العالية في تنفيذ البرامج، وخاصة في الإصدار الرابعة من المترجم، حيث تمت كتابة مترجم PHP من الصفر ليعطي أداءًا في منتهى الروعة، كما أن لغة PHP مصممة أصلاً كنواة لمترجم، بحيث يمكن أن تضع هذه النواة في عدة قوالب أو أغلفة لتعمل مع التقنيات المختلفة، فيمكنك تشغيل مترجم PHP كبرنامج CGI مثلًا، ولكن الأفضل هو إمكانية تركيب مترجم PHP على مزود IIS في صورة وحدة إضافية تضاف إلى المزود عن طريق دوال ISAPI، وتوجد نسخة أخرى منه تركيب على مزود Apache أيضًا في صورة وحدة خارجية، وتوجد أيضًا نسخة مخصصة للدمج مع شفرة مزود Apache بحيث تصبح جزءًا من برنامج Apache نفسه، وهي الطريقة الأكثر استخدامًا الآن في مزودات الويب التي تعمل على أنظمة UNIX وهي الطريقة التي تعطي أفضل أداء لمترجم PHP، حيث يصبح المترجم جزءًا من المزود، وبالتالي فإنه سيكون محملاً في الذاكرة بانتظار صفحات PHP ليقوم بترجمتها وعرضها للزوار مباشرة دون التأخير الإضافي الذي تتطلبه برامج Perl/CGI مثلًا حيث يجب أن يتم تشغيل مترجم Perl مع كل زيارة للصفحة لترجمة الصفحة، ثم يتم إغلاق المترجم، ثم استدعاه مجددًا عند الزيارة الثانية وهكذا، وهذا يشكل فارقًا كبيرًا في المواقع ذات الضغط العالي بالذات، ويكون استخدام PHP حلاً أفضل بكثير.

## المزايا

يأتي مترجم PHP لوحده محملاً بعدد هائل من الدوال الجاهزة للاستخدام في جميع المجالات، من دوال المعالجة الرياضية والحسابية إلى دوال الوصول إلى قواعد البيانات ومزودات FTP، توفر لك دوال PHP مثلًا وصولًا إلى مزودات البيانات MySQL و PostgreSQL و MS SQL و Oracle وغيرها من مزودات قواعد البيانات، وهناك أيضًا مجموعة من الدوال لمعالجة ملفات XML، ودوال أخرى لإرسال واستقبال الملفات عن بعد باستخدام بروتوكول FTP، وهناك مجموعة من الدوال لمعالجة وإنتاج الصور ديناميكيًا وملفات Flash ديناميكيًا، ناهيك عن جميع الدوال الخاصة بمعالجة النصوص والمصفوفات.

## التوافقية

كما قلنا سابقًا، فعلى الرغم من أن هنالك الكثير من نسخ PHP التي يعمل كل منها في بيئة مختلفة، إلا أنها جميعًا تشترك في النواة الأصلية التي تقوم بالمعالجة الحقيقية لملفات PHP لذا فإن جميع مترجمات PHP تتصرف بنفس الطريقة فيما يتعلق بتنفيذ السكريبتات، فإذا كان السكريبت الذي عملته يعمل على

نظام Windows مع مزود IIS فيجب أن يعمل دون الحاجة لأية تغييرات عند نقله إلى مزود Apache، بالطبع تظل بعض الأمور البسيطة جدا التي يوفرها بعض المزودات دون غيرها، ولكن جميع البرامج التي كتبها منذ أن بدأت تعلمي للغة إلى الآن تعمل على جميع المزودات دون الحاجة لأي تغييرات، إضافة إلى ذلك فإن التغييرات التي حدثت باللغة الأساسية من الإصدار الثالثة إلى الرابعة قليلة جدا، وأغلب التغييرات كانت في البنية التحتية للمترجم.

## الحماية

يوفر PHP الكثير من المزايا المتقدمة، ولكنه يوفر لك الطرق المناسبة لوضع الحدود على هذه المزايا، فيمكنك التحكم بعدد الإتصالات المسموحة بقاعدة البيانات مثلا، أو الحجم الأقصى للملفات التي يمكن إرسالها عبر المتصفح، أو السماح باستخدام بعض الميزات أو إلغاء استخدامها، كل هذا يتم عن طريق ملف إعدادات PHP والذي يتحكم به مدير الموقع.

## قابلية التوسع

يمكنك توسعة مترجم PHP بسهولة وإضافة الميزات التي تريدها إليه بلغة C، وحيث أن الشفرة البرمجية للمترجم مفتوحة فإنك تستطيع تغيير ما تريده مباشرة لتحصل على النسخة التي تناسبك من المترجم، ويمكنك أيضا عمل الوحدات الإضافية التي تتركب على المترجم لزيادة ميزاته والوظائف المبيته فيه، وفي قد قام فريق تطوير مترجم PHP مسبقا بعمل هذه المهمة وتحويل كمية ضخمة من المكتبات المكتوبة بلغة C إلى مكتبات مخصصة لتضاف إلى المترجم، ومنها حصلنا على جميع الميزات التي تحدثنا عنها مثل الوصول إلى قواعد البيانات ومعالجة ملفات XML.

## تاريخ PHP

بدأت PHP كمكتبة من الدوال تضاف على لغة Perl لتسهل عمل برامج CGI بلغة Perl، وبعد أن تلقى Rasmus Lerdorf بعض الاقتراحات بتحويلها إلى مترجم بسيط، قام بعمل ذلك المترجم وطرحه على الإنترنت وسماه PHP أو Personal Home Pages أي الصفحات الشخصية، فقد كان عبارة عن نسخة مصغرة من Perl مع بعض الميزات الإضافية للويب، ثم أضاف إليه دعما لنماذج HTML وسماه PHP2/FI، فقام مجموعة من المبرمجين بالعمل على مترجم PHP وأضافوا إليه واجهة تطبيقات برمجية API لتسهيل عملية توسعته فأصبح لدينا PHP 3، بعد فترة من الزمن قامت شركة Zend للتقنيات بعمل مترجمها الخاص للغة والذي سمي zend أيضا، وقد ائصف هذا المترجم بالسرعة العالية وقدراته المحسنة، وجمع مع مكتبات PHP الأخرى لتكوين نواة المترجم PHP، مترجم PHP الآن مقسم على قسمان: المترجم zend ويتم تطويره على مزودات CVS الموجودة في موقع zend والقسم الثاني يسمى PHP وهو عبارة عن المكتبات والدوال الأساسية التي تأتي مع البرنامج، يقوم مترجم zend بقراءة الملفات ومعالجتها والتعامل مع المتغيرات وتنفيذ البرنامج وتوفير واجهة تطوير للتطبيقات API لتوسعة اللغة، أما PHP فتحتوي الآن على مكتبات مكتوبة بلغة C ومتوافقة مع واجهة التطبيقات التي يوفرها مترجم zend، وبالتالي يعمل القسمان معا لتكوين مترجم PHP، وعندما تزور [موقع PHP الرسمي](#) الآن وتحصل على مترجم PHP جاهزا أو تحصل على الشفرة البرمجية الخاصة بك، فإنك تحصل على كل من مترجم zend ومكتبات PHP معا.

تطور PHP تطورا مفاجئا في الفترة الأخيرة، وتشير إحصائيا Net Craft إلى أن مترجم PHP هو أكثر وحدات مزود Apache انتشارات على الإنترنت، كما أن مترجم PHP مركب على حوالي مليوني مزود ويب على الإنترنت.

## بنية ملفات PHP

ملفات PHP هي ملفات نصية بسيطة، تشبه في تركيبها ملفات ASP وملفات HTML بشكل عام، يتكون ملف PHP من قسمان، قسم HTML وقسم PHP، الملف بالصورة الطبيعية عبارة عن ملف HTML عادي، ولكنك تستطيع تحديد أجزاء معينة من الملف ليخرج فيها الملف من وضعية HTML إلى وضعية PHP، لإخراج الملف إلى وضعية PHP توجد عدة طرق :

1 - استخدام زوج الوسوم <?php و >? كالتالي :

```
<?php  
echo 'This is PHP output!';  
?>
```

2 - استخدام زوج المختصر <? و >? وهو يستخدم بنفس الطريقة السابقة ولكنه يكون بدون الكلمة php في وسم البداية، هذا النوع من الوسوم يحتاج إلى كمية أقل من الكتابة بالطبع، ولكنه يتعارض مع وسوم xml، لذا يقوم البعض بإغلاق ميزة الوسوم القصيرة حتى لا يحصل هذا التعارض ( يمكنك إغلاق هذه الميزة بسهولة عن طريق ملف إعدادات PHP ).

3 - استخدام زوج الوسوم ASP، وهو من اسمه زوج الوسوم المستخدم في ملفات ASP وهما <% و %>، ميزة وسوم ASP لا تكون فعالة بشكل قياسي ولكنك تستطيع تفعيلها عن طريق ملف إعدادات مترجم PHP.

4 - الطريقة الأخيرة هي استخدام زوج الوسوم التالي :

```
<script language="php" >  
    echo 'This is PHP output!';  
</script>
```

ولكن هذه الطريقة غير مستخدمة الآن، حيث أنها تصعب عملية التمييز بين شفرات PHP وباقي ملف HTML، وكذلك بالنسبة لبرامج كتابة ملفات HTML التي تعطي تلوينا للشفرة فأغلبها لا يتعرف على هذا النوع من الشفرة ويعتبره جزءاً من ملف HTML الاعتيادي.

أفضل الطرق السابقة للتحويل إلى وضعية PHP هو استخدام زوج الوسوم الأول بالطبع، حيث أنه الأكثر استخداماً، ولا يحتوي على أية تعارضات كما أنه يعمل على جميع مترجمات PHP مهما كانت إعداداتها، ولهذا السبب سنستخدمها في جميع الأمثلة التي ستجدها في هذه الدورة.

## كتابة ملفات PHP

ملفات PHP هي ملفات نصية بسيطة تماماً كما هي ملفات HTML، يمكنك كتابة سكريبت PHP بأي برنامج كتابة نصوص يتيح لك كتابة الملفات النصية البسيطة Plain Text مثل Notepad على النظام ويندوز، ولكن أغلبية مبرمجي PHP يستخدمون أدوات أخرى تسهل عليهم عملية البرمجة عن طريق تلوين الشفرات البرمجية، وتسهل عملية البحث عن الملفات واستبدال المقاطع من عدة ملفات في نفس الوقت، مثل HomeSite، على الرغم من أنك لن تحتاج إلى الكثير من هذه الميزات إلا أن استخدام Notepad في عمل ملفات PHP يعتبر أمراً صعباً جداً وخاصة في الملفات الضخمة حيث أن Notepad لا يتيح فتح الملفات الكبيرة، والمشكلة الأكبر هي أنها لا توفر ترقيماً للأسطر، فإذا ظهرت لك رسالة الخطأ تشير إلى وجود خطأ في السطر 53 فلن تستطيع معرفة السطر المطلوب في Notepad إلا إذا قمت بالعد يدوياً من السطر الأول وحتى 53 .. حسناً ماذا لو كان الخطأ في السطر 652، يمكنك البدء بكتابة سكريبتاتك بالبرنامج المتوفر الآن إلى أن تحصل على برنامج آخر، يمكنك بالطبع فتح ملفاتك بأي محرر نصوص، فإذا كتبتها باستخدام Notepad فهذا لا يعني بأنك ملزم باستخدام Notepad في جميع ملفاتك أو حتى في هذا الملف.

لعمل ملف PHP الآن قم بفتح محرر النصوص الذي اخترته وابدأ بكتابة الصفحة التي تريدها، ولا تنسى إحاطة شفرات PHP بالوسوم الخاصة بها، ثم احفظ الملف في أي مكان في دليل مزود الويب الخاص بك وأعطه الإمتداد المناسب php. أو php3. حسب إعدادات مزودك، ثم قم بزيارة الصفحة باستخدام المتصفح وستجد الصفحة وقد تمت ترجمتها وعرضها عليك.

تذكر بأنك يجب أن تزور الصفحة مرور بمزود الويب، ولا يمكنك عرض الصفحة عن طريق فتحها كملف خارجي، على سبيل المثال، إذا كان الدليل الجذري لصفحات مزودك هو : `C:\httpd\`

وقمت بعمل صفحة أسميتها test.php في ذلك الدليل، يجب أن تقوم الآن بتشغل مزود الويب وزيارة الصفحة على العنوان <http://localhost/test.php>، إذا قمت باستخدام الأمر Open من القائمة File في المتصفح لفتح الملف `C:\httpd\test.php` فلن ترى صفحة PHP مترجمة، وسترى شفرة PHP فقط.

## تدريب

قم بتنفيذ ملف PHP التالي :

```
This is the normal html page.<br>
<?php
    echo "This is inside PHP<br>";
    echo "Hello World!<br>";
?>
```

ما الذي تشاهده عند تنفيذ البرنامج السابق؟ من المفترض أن تشاهد الخرج التالي :

```
This is the normal html page.
This is inside PHP
Hello World!
```

ها قد انتهيت من كتابة برنامجك الأول بلغة PHP، لا تقلق إذا لم تفهم أي شيء فيه، سنتعلم الآن كيفية استخدام المتغيرات والعبارات بلغة PHP.

لنكتب سكريبتاً بسيطاً (فاتح شهية) :

```
>html dir = "rtl <"
التحية لدي أهل الإسلام هي
?>
Echo ("السلام عليكم ورحمة الله وبركاته")
<?
/>html <
```

قم بحفظ الملف باسم echo.php ستعرض علينا عبارته مكتوب فيها

التحية لدي أهل الإسلام هي السلام عليكم ورحمة الله وبركاته

شي بسيط أليس كذلك ؟

يتكون كود الphp من نصوص و كود و علامات ولغة html وقد لاتحتوي على نصوص html . لكي يعمل الكود يجب أن يكون إمتداد الملف php أو بأي إمتداد من إمتدادات الphp مثلأ php3 و phtml

--  
عندما تطلب صفحة في الإنترنت فإنك تجري اتصالاً مباشراً مع السيرفر هذه العملية تدعي request للسيرفر (يعني طلبية للسيرفر) يقوم السيرفر بتفسير طلبك والبحث عن الصفحة المطلوبة ويرسل اليك الصفحة المطلوبة كجزء مما يسمى response (استجابة) لمستعرض الانترنت لديك يقوم بعدها المتصفح لديك بأخذ الكود الذي ارجع إليه ويقوم بتجميعه (compile) لكي يصبح

صفحة صالحة للعرض هذه العملية التي حصلت تشبه نظرية العميل للخادم (client to server) بحيث أن المتصفح هو العميل والخادم هو السيرفر .  
الخادم يقوم بعملية تخزين وترجمة وتوزيع البيانات بينما يقوم العميل (مستعرض الانترنت لديك) بالعبور الى السيرفر واحضار البيانات

## بروتوكولات الانترنت :

لانريد هنا أن نذهب إلى التكلم عن تاريخ انترنت العتيق ، النقطة المهمة هي الشبكة المربوطة بنقاط nodes الانترنت صممت لكي تقوم بالحفاظ على المعلومات لكي يتم نقلها من مكان إلى آخر وهي تستخدم مجموعة من البروتوكولات مثل Tcp/Ip لكي يتم نقل البيانات عبر الشبكة .

### بروتوكول Tcp/Ip

من مميزات هذا البروتوكول أنه باستطاعته إعادته تمهيد طريقه للبيانات إذا تم خلل في نقطة أو مكان أثناء نقلها ويتم ذلك بسرعة شديدة. عندما يطلب المستخدم من المستعرض أن يجلب له صفحة من الانترنت فإن المستعرض يجلب هذه الأوامر باستخدام بروتوكول يدعي بروتوكول التحكم في نقل البيانات TCP هذا البروتوكول هو بروتوكول نقل للبيانات وهو يضمن أن البيانات قد تم إرسالها ووصولها بشكل صحيح .

قبل أن يتم إرسال البيانات عبر الشبكة يجب عنونها والبروتوكول الذي يقوم بعنونة البيانات يدعي HTTP يقوم هذا البروتوكول بوضع عنونة للبيانات لكي يعرف البروتوكول TCP أين سينقل البيانات (فهو لا يستطيع نقل البيانات إذا لم يكن لها هدف أو مكان) يستخدم البروتوكول HTTP عن طريق الويب في عملية نقل البيانات من كمبيوتر إلى آخر عندما ترى الصفحة متبوعة بـ http:// فانك تعلم مباشرة أن الانترنت يستخدم البروتوكول HTTP لإحضار هذه الصفحة يمكنك أن تأخذ صورة بأن الTCP عبارة عن ساعي بريد الذي يقوم بإيصال رسالة ، هذه الرسالة فيها طابع بريد وعنوان وهو مانسميه بالHTTP .

يتم تمرير الطلب من المستعرض إلى ملقم أو سيرفر الويب وهو ما يعرف بـ HTTP request ويقوم السيرفر برؤية مستودع البيانات لديه لكي يحصل على البيانات المطلوبة فإذا وجد الصفحة في المستودع قام بإرسالها على شكل حزم الى الجهة التي قامت بالطلب باستخدام بروتوكول TCP ويعنون هذه الحزم لمستعرض الانترنت لديك باستخدام بروتوكول http (نبيه دائماً الى أنه يرسلها على شكل حزم لكي تعرف السبب عند عدم ظهور صفحة ويب كاملة أن هناك حزمة لم ترسل بشكل جيد) ولكن إذا لم يجد السيرفر الصفحة المطلوبة فإنه يقوم بإرسال صفحة تحتوي على رسالة خطأ 404 وهذه الصفحة التي أرسلت من ملقم الويب الى المستعرض لديك تسمي بالHTTP response .

### بروتوكول ال HTTP

رغم ما أخذناه من معلومات كثيرة وقصص كثيرة تشبه قصص ألف ليلة أو حكايات الأطفال إلا أنه رغم ذلك يفوتنا الكثير من التفاصيل في هذا الموضوع لذلك دعنا نغوص قليلاً في التفاصيل عن بروتوكول HTTP بشكل خاص.

عندما تقوم بعملية طلب لصفحة من السيرفر هناك أمور إضافية ترسل مع عملية الطلب http request غير الURL وهي ترسل كجزء من http request .  
نفس الموضوع مع الhttp response هناك أمور أخرى تصل معه كجزء منه .

الكثير من هذه المعلومات تولد تلقائياً في رسالة الHTTP ولايقوم المستخدم بالتعامل معها مباشرة ، إذن لايجب أن تقلق نفسك بشأن هذه المعلومات إذا أنت لم تنشأها في الأصل ويجب أن تأخذ أيضاً في معلوماتك أن هذه المعلومات ترسل كجزء من الHTTP request والHTTP response لأن سكرت الPHP الذي نصنعه يمنحنا تحكماً إضافياً بهذه المعلومات .

كل رسائل الHTTP تأخذ تنسيقاً معيناً سواء كانت Request أو Response . نستطيع أن نقوم بتقسيم هذا التنسيق إلى ثلاثة أقسام :

- 1 - Request/response line
- 2 - Http header
- 3 - Http body

المحتوي من هذه الأشياء الثلاثة يعتمد على نوع الرسالة إذا كانت HTTP Request أو HTTP response لذلك سنتكلم عنهم بتعمق أكثر .

## Http Request

يجب أن يحتوي الrequest على الأقل الrequest line (سطر الطلب) والHOST .  
يرسل مستعرض الانترنت طلبية (HTTP request) إلى ملقم الويب تحتوي على التالي :

### The Request Line - 1

- السطر الأول من كل طلبية (http request) هي Request Line الذي يحتوي على ثلاثة أنواع من المعلومات :
- أ - أمر HTTP وهو مايعني بـ method .
- ب - المسار من السيرفر إلى المصادر المطلوبة (صفحات الانترنت) المطلوبة من قبل العميل (المستعرض)
- ج - إصدار الHTTP .

إذن كمثال على ال Request Line أنظر إلى السطر التالي :

GET /testpage.htm HTTP/1.1

الmethod يخبر السيرفر كيف يتعامل مع الطلب هناك ثلاثة أنواع شائعة من الmethod

## HTTP Header - 2

البيت الثاني من المعلومات هو الهيدر HTTP Header ،الذي يحتوي على تفاصيل أو وثائق عن العميل مثل نوع المتصفح (نتسكيب أو إكسبلور) الذي قام بطلب الصفحة والوقت والتاريخ والإعدادات العامة ال HTTP Header يحتوي على معلومات نستطيع تقسيمها الى ثلاث فئات وهي :  
أ - عامة GENERAL : تحتوي معلومات إما عن العميل أو السيرفر ولا تخصص إلى فرد أو مجموعة .  
ب - شخصية Entity : تحتوي على معلومات عن البيانات التي أرسلت بين المتصفح والسيرفر .  
ج - مطلوبة Request : تحتوي على بيانات عن إعدادات العميل والأنواع المختلفة المقبولة من البيانات .

وهذا مثال :

\* / \* :Accept

.Accept language: Arabic-KSA

.Connection: Keep -Alive

Host : <http://www.arabbuilder.com>

Referer: <http://www.arabbuilder.com/index.php?something=132>

(.....;User -Agent :Iexplor (win98

مثلما ترى ال HTTP Header عبارة عن إعداد يتكون من عدة سطور كل سطر يحتوي على قيم معينة .

هناك عدة سطور تشكل ال HTTP header وأكثرها إختياري , يقوم ال HTTP بالإخبار عن إنتهاء معلومات ال header بترك سطر فارغ (وهذا يكون في ال HTTP1.1) .

## The HTTP Body - 3

إذا تم استخدام الأمر POST في ال HTTP Request Line عندها يقوم ال HTTP بطلب المعلومات التي أرسلت في ال body الى السيرفر .

## Http Response

يرسل من السيرفر إلى المستعرض ويحتوي على ثلاثة أشياء :

the Response Line - 1

http header - 2

Http Body - 3

## The Response Line - 1

ال response line يحتوي فقط على نوعين من المعلومات :

1 - رقم إصدار ال HTTP .

2 - شفره أو كود ال http request التي تقوم بتحديد إذا كان ال request ناجحاً أم فاشل .

مثال :

HTTP/1.1 200 OK

في هذا المثال يقوم ال response line بإرجاع القيمة 200 متبوعة بالكلمة OK هذه تشكل وتشير إلى نجاح ال request ويكون ال response يحتوي على الصفحة المطلوبة والبيانات من السيرفر . ومثال آخر هو الشفرة 404 عندما تقوم بطلب صفحة ويفشل السيرفر في الحصول عليها .

## HTTP Header - 2

ال response header يعتبر مشابه ال request hader الذي ناقشناه في الأعلى . وتنقسم المعلومات التي فيه أيضا إلى ثلاثة أنواع :  
أ - عامة GENERAL : معلومات عن ال client أو السيرفر ولا تخصص إلى واحد منهما .  
ب - شخصية Entity : يحتوي على معلومات عن البيانات التي يتم إرسالها بين السيرفر والعميل .  
ج - الإجابة Response : يحتوي معلومات عن السيرفر الذي قام بإرسال الرد وكيفية تعامله ومعالجته للرد ( Response) .

كما قلنا سابقاً ، يتكون من عدة سطور ويتم وضع سطر فارغ للإعلام عن إنتهاء الهيدر .

مثال :

```
HTTP/1.1 200 OK -the status line
Date: Mon; 1st Nov 1999, 16:12:23 GMT -general header
Server : Apache/1.3.12 (Unix) (SUSE/Linux) PHP/4.0.2 -the response
Last-modified: Fri, 29 Oct 1999, 12:08:03 GMT -Entity Header
```

السطر الأول ناقشناه والسطر الثاني مفهوم من غير شرح ، السطر الثالث يقوم بتحديد البرنامج تبع السيرفر ونوعه ونظام التشغيل القائم عليه والسطر الأخير يقوم بتعريف آخر وقت تم فيه تعديل أو تجديد الصفحة .

ملاحظة : قد يحتوي الهيدر على أكثر من هذه المعلومات أو معلومات مختلفة وهذا يعتمد على نوع الشيء المطلوب من السيرفر .

### Http Body - 3

إذا تم معالجة الطلب بنجاح ، فإن الـ HTTP response Body يحتوي على كود الـ HTML ويقوم مستعرض الانترنت بتفسيرها وتحويلها إلى الصفحة النهائية التي تراها .

### أين سكربت الـ PHP من ذلك كله ؟

أصبح الآن لدينا مفهومية جيدة عن طريقة إرسال المستعرض طلب صفحة من السيرفر وكيفية استجابة السيرفر لهذا الطلب .

تكلمنا عن أن سكربت الـ php يتكون من ثلاثة أشياء : نص وكود الـ php وكود الـ html ، لانسطيع وصف الـ html بأنها لغة برمجة بشكل جيد ونستطيع أن نقول أن الـ php لغة سكربتات Scripting Language لأنها تضيف قدرات الـ html عليها مثل الجداول والفريمات بكود الـ html بداخل كود الـ php هناك لغات تسمى لغات سكربتات قد تكون متألماً معها مثل الجافا سكربت والفجول بيسك سكربت بإستثناء أن الفرق بينها وبين الـ php هو أن الـ php لغة تعتمد على جهة المزود أي السيرفر ويمكنك تخصيص المتصفح الذي يستعرضها .

تجعلنا الـ html نضمن سكربتات الـ php فيها ضمن قواعد لذلك لكي نستطيع تشغيلها ولكننا لاننسى أن إمتداد الملفات يظل كما هو الـ php أو الـ php3 بدون تغير فيه لكي يتم إرسال السكربت الى مكتبة الترجمة (scripting engine) التي تقوم بترجمة السكربت إلى الـ html (كانك تترجم من عربي لإنجليزي أو العكس )

### مفهوم الـ parsing و الـ Execution :

يمكن أن نقسم عملية الترجمة الذي يقوم بها سيرفر الـ php إلى قسمين أو عمليتين :  
العملية الأولى : هي أن السيرفر يقوم أولاً بفحص قواعد اللغة وهذا لا يضمن أن السكربت صحيح مائة بالمائة ولكنه تدقيق في الأوامر وقواعد اللغة وهذا مايسمونه بالـ Parsing  
العملية الثانية : هي تنفيذ السكربت بعدها وإخراجه على شكل كود الـ html وهذا مايسمى بالـ Execution .

بقي أن نقول أمراً معروفاً وهو أن السكربتات نوعين :

1 - وهو ماينفذ من جهة المزود

Server -Side scripting

2 - ماينفذ من جهة المستعرض (صفحة انترنت) .

### التعليقات

ما رأيك إذا كنت في شركة وكان معك أكثر من مبرمج وأردتم تصميم برنامج ، إذن قد تحتاجون لتنظيم العمل وتعديله لذا من اللازم أن تقوم بعمل توضيح لفائدة الكود الذي كتبته كي يسهل فهمه عليهم وإضافة تعديلات مناسبة ، إذن التعليقات تستخدم في الإفاده عن شرح الأكواد أو إضافة معلومات لاتستعمل إلا كتوضيح أو أي شيء آخر .

يمكنك عمل تعليق من سطر واحد كالتالي :

```
<?
هذا تعليق لافائدة له له اي معني//
?>
```

مثال آخر :

```
<?
هذه الداله تقوم بطباعه الكلمه تعليق//
Echo "تعليق";
?>
```

وأیضا يمكنك استخدام تعليق من أكثر من سطر كالتالي :

```
<?
تعليق يتكون من *
?>
```

اكثر من سطر بعلامة السلاش والنجمه

```
*/  
?>
```

## المتغيرات

### ماهي المتغيرات ؟

أبسط تعريف يمكن أن نقوله عن المتغير هو أنه مساحة من الذاكرة تستخدم لتخزين المعلومات ويتم التحكم فيها عن طريق المبرمج في الـ PHP ، المتغيرات تبدأ بعلامة الـ \$ ولكي تقوم بإدخال قيمة في المتغير فإنك تستخدم المعامل (=) إذن لكي تقوم بإنشاء متغير يحتوي على قيمة يمكنك القيام بذلك كالتالي :

```
$alfares = "How Are You Every Body?";  
$اسم_المتغير = قيمه ;
```

لاحظ أن السطر السابق يتكون من خمسة أشياء :

- 1 / المتغير وهو alfares
- 2 / وقبله علامة الـ \$ لكي يعرف مترجم الـ PHP أنه متغير
- 3 / المعامل (=)
- 4 / الفاصلة المنقوطة ( ; )
- 5 / القيمة وهي How Are You Every Body? وهي القيمة الموجودة في المتغير أو التي اقترحناها للمتغير أو التي وضعناها فيه ( لأن الذي اقترح القيمة هو أنت (مبرمج الـ php) )

### ملاحظات :

1- اسماء المتغيرات حساسة لحالة الأحرف إذا كانت كبيرة وصغيرة

```
<?  
$Ahmed = "salem";  
$ahmed = "slmoon";  
echo $ahmed;  
echo $Ahmed;  
?>
```

المتغيرين الذين بالأعلي مختلفين بسبب حالة الأحرف.

2 - يمكنك استخدام المعامل ( \_ )

```
$First_name
```

3 - يمكنك استخدام ألف حرف في تسميه المتغيرات (وفي الواقع هي غير محده) .

### علامات التنصيص

وهذه نقطة مهمة وهي لماذا وضعنا علامات التنصيص هذه ؟ فالإجابة تكون هي أن القيمة التي وضعناها حرفية أي تتكون من نصوص وهناك أنواع للمتغيرات وعلى ذلك سنفصل ونقول

### هناك أنواع للبيانات وهي :

1 - strings (حروف)

```
$Exa = "Just An Example";  
$Exa2 = "2.5";  
$Exa3 = "2";
```

2 - Integer (ارقام)

```
$Exam = 5;
```

3 - Double (ارقام ذات فواصل)

```
$num = 5.4
```

4 - array

ياتي تفصيلها فيما بعد

5 - objects



تفصيلها في دروس اخرى  
6 - Unknown .  
ياتي تفصيلها في درس اخر .

المتغيرات لا يتم تعريف نوعها من قبل المبرمج إنما مترجم الـ PHP يقوم بالتعرف عليها لكي يتم إتمام العمليات المختلفه عليها .

### البيانات الحرفيه /

في الـ PHP أي قيمة تكون بين علامتي تنصيص عادية أو علامة تنصيص مفردة يعتبرها الـ PHP قيمة حرفية أمثلة :

"هذا النص بين علامتي تنصيص عادية اومزدوجه"  
'هذا النص بين علامتي تنصيص مفردة او وحيدة'

يجب أن يبدأ النص وينتهي بنفس علامة التنصيص ، وإلا فلن يتعرف الـ PHP على القيمة الحرفية أو على النص .

```
<?
$d="غلط `
echo "خطا `
?>
```

لا يمكنك أيضاً أن تقوم بوضع علامة تنصيص من نفس النوع التي تستخدمه القيمة الحرفية في وسط العبارة الحرفية أو النص

```
<?
$variable = "هذا النص "خطا بسبب وجود علامة في النص من نفس النوع";
?>
```

وتصححه

```
<?
$variable = "هذا النص `صحيح";
?>
```

وأيضاً مثال آخر

```
<?
$r = "This is"BAD"; // خطأ
$t = "This is `good"; // صحيح
?>
```

أما إذا كنت مصراً على ذلك أو تحتاج إليها في عمليات ضرورية (كما سوف نرى فيما بعد حاجتنا إليها في صناعة النماذج) فيمكنك وضع معامل ( \ ) قبل علامة التنصيص . لكي تعمل معك بكل سهولة .

مثال :

```
<?
$u = "This Only An \" Example\" To Make You Understand Nothing";
?>
```

طيب ما رأيك لو أردنا أن نطبع المعامل ( \ ) بنفسه ؟  
الحل هو أن نتبعه بمثله ، وبالمثال يتضح المقال :

```
$file = "c:\windows\system.ini";
echo $file; // النتيجة c:windowssystem.ini

$file = "c:\\windows\\system.ini";
echo $file; // النتيجة c:\windows\system.ini
```

يمكنك الجمع بين أكثر قيم المتغيرات في متغير واحد عن طريقة الـ (.) .

```
<?
$first = "متدي";
$last = "المطور العربي";
$fullname = $first. $last
Echo $fullname ;
ولكننا نريد وضع فراغ بين الكلمتين //
```

```
$fullname= $first . ' ' . $last ;  
Echo $fullname ;  
?>
```

وأيضاً يمكننا أن نضيف إلى متغير قيمة متغير آخر :

```
<?  
$f="I Love M" ;  
$k= "y Country" ;  
//إضافه القيمة الى المتغير/  
$f = $f . $k;  
echo $f;  
?>
```

```
<?  
//تقريباً نفس العملية/  
$f="I Love M" ;  
$k= "y Country" ;  
$f.=$k;  
echo $f;  
?>
```

## الارقام

العدد الفردي والمزدوج  
الاختلاف المعروف لدي أنا حتى الآن هو أن الفرق بينهما هو الفاصلة العائمة (والله حتى اعطاءها هذا الاسم يجعل الواحد يشعر بالاحباط والخوف )  
لاحظ أننا لا نستخدم علامات التنصيص وذلك ليعرف الـ PHP أنها بيانات رقمية قد نستخدمها في عمليات حسابية معقدة  
ويمكننا تطبيق عمليات حسابية بسيطة عليها إذا كانت حرفية .

```
//هذا عدد فردي/  
$z=2  
//هذا عدد مزدوج/  
$h=4.5
```

## العمليات الحسابية

هي مثل الجمع والطرح والضرب والقسمة وهي مرتبة كالتالي :  
أولاً /الأقواس  
ثانياً /الضرب ثم القسمة .  
ثالثاً / الطرح ثم الجمع

```
<?  
Echo 5*2/5;  
Echo 5*(2/5) ;  
?>
```

مثال آخر :

```
<?  
Echo 5-6+9 ;  
?>
```

مثال لعملية حسابية نستخدم فيها متغير حرفي

```
<?  
$W="2L";  
$E= 2;  
$F = $W * $E;  
echo $W . ' ' . $E . ' ' . $F;  
?>
```

مثال لعملية أخرى لكنها لم تعمل عليك استنباط السبب بنفسك (هاه طل زين) :

```
<?
$W="L10";
$E= 2;
$F = $W * $E;
echo $W .''.$E .''.$F;
?>
```

يمكننا إضافة رقم واحد الى متغير بثلاث طرق متنوعة :  
مثال

```
$j++
```

أو

```
$j = $j+1
```

أو

```
$j += 1
```

ويمكننا على ذلك إضافة المتغير إلى نفسه كالتالي :

```
$j += $j
```

أو كالتالي :

```
$j = $j + $j
```

## متغيرات النظام

هناك متغيرات يستخدمها النظام يمكنك أن تستعملها ومنها

```
$HTTP_USER_AGENT
```

التي تظهر لديك نوع المستعرض الذي يستخدمه العميل

مثال :

```
<?
Echo $HTTP_USER_AGENT ;
?>
```

## الثوابت

يمكننا تعريف الثوابت بقول أنها قيم ثابتة لاتتغير ونعرفها عن طريق الدالة define  
الثوابت حساسة أيضا لحالة الأحرف

```
<?
Define ("author", "alfarees");
Echo "author is " . author ;
?>
```

هناك ثوابت يستخدمها النظام مثل

```
PHP_OS
```

التي تقوم بعرض نظام التشغيل الذي يستخدمه السيرفر

مثال :

```
<?
Echo PHP_OS;
?>
```

## معرفة وتحويل انواع البيانات

إذا أردت أن تعرف نوع متغير ما يمكنك استخدام الدالة gettype

مثال :

```
<?
$n=5;
$l="hi";
echo "The n Is " . gettype ($n) . "<br>";
echo "The l is " . gettype ($l);
?>
```

إذا أردت تحويل نوع متغير ما يمكنك ذلك باستخدام الدالة settype :

مثال :

```
<?
$n = 10 ;
echo "Before is " . gettype ($n) . "<br>";
settype ($n,"string");
echo "After That is go " . gettype ($n);
?>
```

### الدالة isset

لمعرفة إذا كان المتغير منشأ مسبقاً أم لم يتم انشاؤه وهي لا تتطلب غير اسم المتغير الذي تريد فحص وجوده وتقوم بإرجاع القيمة (1) إذا كان المتغير تم انشاؤه ولا ترجع أي قيمة إذا كان المتغير غير منشأ أو موجود .

مثال :

```
<?
$n = "n";
Echo isset ($n);
?>
```

### الدالة unset

تقوم بحذف المتغير إذا كان موجوداً وتحرير الذاكرة منه ( لذلك تأكد جيداً قبل استخدام هذه الدالة من اعطاء دعة الوداع للمتغير المسكين )

```
<?
$n = "n";
unset ($n);
Echo isset ($n);
?>
```

### الدالة empty

تقوم بإرجاع القيمة ( 1 ) إذا كان المتغير غير منشأ أو أن القيمة التي فيه صفر (0) أو نص فارغ ("" ) ولاتقوم بإرجاع أي شي اذا كان المتغير منشأ وفيه قيم غير المذكورة .

## داوال الوقت التاريخ

نستطيع إيجاد الوقت و التاريخ عن طريق دوال في الPHP من تلك الدوال الدالة

gmdate ()

مثال :

```
<?
Echo gmdate (m);
Echo gmdate (M);
?>
```

لاحظ أن هناك فرق في النتائج مع أننا نستخدم نفس الحرف لكن طريقه العرض تختلف عندما يكون الحرف كبيراً أو صغيراً .

تحتجز الphp بكثير من الدوال والكلمات المحجوزة التي تقوم بعمليات مختلفة مثل العمليات الحسابية المعقدة والقيام بإيجاد الوقت والتاريخ وإرسال الرسائل البريدية وإيقاف السكريبتات لعدة ثواني هذه الدوال ليس مطلوب منك أن تحفظها كما تحفظ اسمك إنما المطلوب منك أن تفهم ماهية عملها واستخدامها في الوقت الذي تراه مناسباً .

يمكنك أيضا عرض اليوم والشهر

مثال

```
<?
```

```
Echo gmdate ("M D");  
?>
```

لاحظ أننا استخدمنا علامات التنصيص لكي تنجح العملية عندما قمنا باستخدام أكثر من عامل في الدالة  
جرب استخدام الكود التالي :

```
<?  
Echo gmdate ("D, d M Y H:i:s")  
?>
```

## النماذج

النماذج في الويب أو صفحات الانترنت عبارة عن استمارات تقوم بتعبئتها ثم عند إرسالها ل خادم الويب (السيرفر) يتلقاها برنامج يقوم بإجراء العمليات عليها مثل JavaScript أو ASP أو php (في حالتنا) .

### فائدة النماذج

لنقل أنك مثلاً أردت شراء كتاب من الانترنت فإنك في الواقع تحتاج إلى تعبئة استمارة ببياناتك ورقم بطاقة الائتمان وغير ذلك من المعلومات ويتم ذلك عن طريق نموذج (فورم) .

في الواقع أنت تقوم باختيار الكتاب الذي تريد وتكتب اسمك ورقم هاتفك وصندوق بريدك (ربما) في فراغات أو عن طريق الإشارة الى مجموعة من الخيارات .  
يتم تخزين هذه القيم في المتغيرات التي يتم كتابتها في الخاصية name (نتكلم عنها في هذا الدرس) ويتم إرسالها عند ضغط زر - ارسال البيانات - (submit) الى (البرنامج) الصفحة التي سوف تقوم بمعالجة هذه البيانات (والتي يتم تحديدها في الخاصية ACTION) وإجراء العمليات عليها مثل تخزينها مثلاً في قاعدة البيانات أو إرسالها إلى البريد الإلكتروني وذلك عن طريق php.

### ماذا يعمل العميل في النماذج ؟

إنه باختصار يقوم بتعبئة مربعات نصوص (textBox) ويقوم بوضع علامة صح في مربعات الاختيار (check boxes) أو يقوم بالتصويت أحياناً لشئ معين فيختار زر اختيار (ازرار الراديو) .  
هذه الأشياء كلها يتم انشاءها بواسطة الhtml ودرسنا لهذا اليوم يناقش كيفية انشاءها وكيفية التعامل والحصول على البيانات منها ، بقي علينا كبدية أن نعرف أن هذه الأدوات تنشأ في الواقع بين وسمين من وسوم لغة الhtml وهي الوسمين

```
<form >  
</form>
```

### خصائص النماذج

يجمع النموذج جميع خصائص المضيف لكننا هنا سنتطرق الى اثنين منهما وهما ACTION و METHOD التي تستخدم بكثرة و مهمة لنا في دروسنا القادمة  
اما (ID;CLASS;NAME) فيلزمها تعمق في HTML خاصة عندما ندخل في ACCEPT-CHAR و ENCTYPE وستكون خارج نطاق موضوعنا حالياً وقد نصلها في دروس قادمة إن شاء الله .

### ACTION

وظيفة هذه الخاصية أن تخبر السيرفر مكان الصفحة التي يقوم بإرسال معلومات النموذج إليها أو عنوانها أياً كان نوعها ، وطبعاً في حالتنا ستكون الصفحة الثانية هي الصفحة التي تحتوي على سيركيت الphp .  
ليس مهماً أن تكون الصفحة php فقد تكون html ولكنها تحتوي على كود يختص بالتعامل مع برنامج تفاعلي لصفحات الويب مثل الجافا .  
ولانريد أن نخرج عن نطاق الموضوع فدعنا نعطي مثلاً على هذه الخاصية :

```
<FORM ACTION = "TEST.PHP">  
.....  
</FORM>
```

### METHOD

هذه الخاصية تقوم بإخبار النموذج طريقة ارسال المعلومات الى الصفحة الهدف وفي الواقع هناك طريقتين مشهورتين ومعروفتين لارسال المعلومات هما GET و POST .

```
<FORM ACTION = "test.php" METHOD = "GET">
```

أو

```
<FORM ACTIN = "test.php" METHOD = "POST">
```

ملاحظه /في الواقع يوجد اكثر من هذه الطريقتين لارسال المعلومات وهي (CONNECT:HEAD:OPTIONS:DELETE:TRACE) وغيرها ولكن لا نستخدم الا بشكل نادر .

دعنا الآن نفصل هاتين الطريقتين بشكل أوسع :

## GET

تقوم هذه الخاصية بإخبار مستعرض الانترنت لديك بأن يقوم بإضافة المعلومات التي تمت كتابتها في النموذج إلى متصفح الانترنت لديك وتكون طريقة كتابته كالتالي :

- 1- كتابه عنوان الصفحة المصدر .
- 2- اتباعها بعلامة استفهام .
- 3- كتابة العناوين والقيم .

<http://localhost/test.html?name=value>

قد تكون النقطتين الأخيرتين غير مفهومتين بشكل جيد بسبب أنك لم تتعامل مع النماذج من قبل .  
لكن الحقيقة أن النموذج يتكون من عناصر (مربع علامة , مربع نص , زر اختيار) ولكل من هذه العناصر عنوان خاص بها (name) ولكل منها قيمة خاصة بها (value) .  
وهي مشابهة للمتغيرات ويمكن أن يحتوي عنوان الصفحة على أكثر من عنوان (name) وأكثر من قيمة (value) ويقوم بالتعريف عنهما باستخدام المعامل (&).

مثال :

<http://localhost/test.html?animal=cat&age=30>

تسمى الإضافة التي تظهر بعد علامة الاستفهام (query String) نتيجة الاستعلام الحرفية.  
العنوان دائماً يكون باللغة الانجليزية (name) ونعامله كأنه اسم متغير من المفترض تعريفه في الصفحة الهدف (التي سنكتبها بالPHP).

قد تحتوي القيم على فراغات او معاملات مثل (%،#،\،-،+)  
يقوم المتصفح باستخدام لغة تشفير الصفحات URL ENCODING .  
أيضا يستخدم URL ENCODING مع الأحرف العربية أو اللغات الأخرى غير الإنجليزية في كتابة الحرف .

## URL Encoding

هناك بعض الأحرف لا يستطيع المتصفح إضافتها لعنوان الصفحة بصيغتها الحقيقية بل يستخدم لغة التشفير في التعريف عنها وهذه جداول بالرموز الذي يستخدم المتصفح كود بدلا من عرضها بصيغتها الحقيقية

الحرف	شفرته	الحرف	شفرته	الحرف	شفرته
Tab	%09	(	%28	;	%3B
Space	%20	)	%29	<	%3C
!	%21	+	%2B	>	%3E
"	%22	,	%2C	=	%3D
#	%23	.	%2E	?	%3F
@	%40	/	%2F	%	%25
\	%5C	:	%3A	&	%26

لا تقلق فليس عليك أن تحفظ كل هذه العلامات وتشفيراتها بل سيقوم المتصفح بالعملية كلها بدلا عنك .

## POST

في الواقع وظيفتها هي نفس وظيفة الget ولكنها لاترسل المعلومات في عنوان صفحة الانترنت بل تقوم وضعها في الbody التابع لـ http response .  
بالإضافة إلى أنه يستطيع ارسال البيانات بكمية أكبر من الGET .

## أيهما تستخدم GET أم POST ؟

قد يكون العيب في الخاصية GET عدم سرية المعلومات التي تقوم بكتابتها ومن الممكن أن تظهر للشخص الذي يجلس الى جوارك ... خاصة عندما تريد الحفاظ على سرية معلوماتك .  
أضف إلى ذلك أنها غير مفيدة في النصوص الكبيرة الحجم .  
ولكنها مفيدة في أشياء كثيرة فمثلاً محركات البحث يجب أن تستخدم هذه الخاصية لكي يستطيع المستخدم أن يستخدم عنوان البحث ويحفظ به لوقت آخر ولا يقوم من جديد بكتابة الكلمة التي يبحث عنها .

أيضا الPOST مفيدة في إخفاء المعلومات وإحتواء كميات كبيرة من البيانات ولكن لايمكن الاحتفاظ بعنوان الصفحة .... مع ذلك فإنها أيضا ليست جيدة في الحماية بحيث أن أي هاكر خبير يمكنه الحصول على المعلومات إذا لم يكن لها تشفير معين في

نقلها .. لكن إذا اردت فعلاً أن تجعلها محمية فيجب عليك استخدام اتصال محمي الى سيرفر محمي .... [او ماسمونه \( SCURE \)](#)  
(CONNECTION TO SCURE SERVER).

## أدوات التحكم في النماذج :

في الواقع أن أدوات التحكم عبارة عن مربعات النصوص العادية (التي يدخل فيها المستخدم اسمه وعنوانه ) وأزرار الراديو ( والتي يقوم المستخدم فيها باختيار شي معين (مثل الوجهة المفضلة لديه أو المشروب المفضل اليه ) ومربعات الاختيار (التي تتيح للمستخدم أن يختار مايشتهي ويحب من الخيارات المعروضة )  
وأيضاً القوائم التي تساعدك على اختيار أكثر من شي أو شي واحد .

في أغلب هذه الاشياء يتم استعمال الوسم

<INPUT>

وتلخيص تفصيله كالتالي :

```
<INPUT TYPE= type NAME= name VALUE= value other attribute>
```

### الشرح :

TYPE= type - 1

يحدد نوع الكائن إذا كان زر راديو أو مربع نص عادي أو مربعات الاختيار .

NAME= name - 2

تقوم فيها بإعطاء اسم للمتغير يتم حفظ القيمة فيه .

VALUE= value - 3

سيوضح وظيفته أكثر عندما ندرج عليه أمثله إذ أن عمله يختلف من أداة إلى أخرى .

## تطبيقات عملية

سنقوم في هذه التطبيقات بصنع برامج بسيطة تتكون من ملفين ، الملف الاول يحتوي على كود HTML يقوم بتكوين النموذج والملف الثاني يقوم بإستقبال النتائج وطباعتها .

### مربعات النصوص (TEXT Box) :

نقوم بعمل ذلك كالتالي :

1 - قم بتشغيل محرر النصوص لديك .

2 - اكتب الكود التالي :

```
<html dir = "rtl">
<FORM METHOD = "GET" ACTION = "textbox.php">
ماهي وجبتك المفضلة في الصباح ؟

<br>
<INPUT TYPE = "text" NAME = "food" value="جبنه ومربي">
<br>
<INPUT TYPE= submit VALUE="إرسال">
<INPUT TYPE= reset VALUE="مسح">
</form>
</html>
```

3 - قم بحفظ الملف كصفحة HTML . وسميته (textbox.html) .

4 - افتح محرر النصوص إذا كنت أغلقته .

5 - اكتب الكود التالي :

```
<?
Echo "وجبتك المفضله والى تموت في حبها هي" $food ;
?>
```

- 6 - قم بحفظ الملف كـ php . وقم بتسميته textbox.php .
- 7 - الآن قم بأخذ الملفين وضعهما في مجلد السيرفر لديك .
- 8-قم بتشغيل السيرفر واكتب في مستعرض الانترنت لديك <http://localhost/textbox.html>
- 9 - قم بكتابة وجبتك المفضلة واضغط زر إرسال .
- 10- ستظهر النتيجة .

لاحظ كيف ظهر العنوان :

<http://localhost/textbox.php?food=%CC%C8%E4%C9+%E6%E3%D1%C8%ED>

### الشرح

لقد قمنا في البداية بعمل صفحة تتكون من نص و مربع نص وزر يقوم بعملية إرسال البيانات  
قمنا بصناعة بداية النموذج بواسطة الوسم <FORM> وقمنا بتحديد المكان الذي سيتم ارسال البيانات إليه بواسطة  
ACTION="textbox.php"

وقمنا بصنع مربع النص بواسطة الوسم INPUT واخترنا ال

TYPE="text"

كما قمنا بوضع القيمة الافتراضية فيه بواسطة القيمة

Value="جبنه ومربي"

وقمنا بوضع الناتج الذي يضعه المستخدم في مربع النص في المتغير food .

(لاحظ ان تسميه المتغيرات حساسه لجاله الاحرف في PHP واننا لم نقم بوضع \$ في صفحه المتغير في كود الhtml).

وأیضا لقد قمنا بإضافة زر بواسطة

TYPE=SUBMIT

وقمنا بوضع كلمة على الزر وهي كلمة (إرسال)

VALUE = "إرسال"

أیضا قمنا بصنع زر آخر

Type =reset

وقمنا بجعل العبارة التي عليه (مسح)

Value="مسح"

هناك نوعين من الازرار هي SUBMIT وRESET

1- الsubmit يقوم بإرسال المعلومات .

2- الreset يقوم بمسح البيانات في جميع الأدوات في النموذج لإعادته إدخالها من جديد .

بعد ماقمنا بادخال البيانات وضغط زر الارسال قام النموذج بارسال البيانات إلى الصفحة المحددة في الخاصية ACTION وقامت  
الصفحة المحددة بإستقبال النتائج الموجودة في النموذج وهي نتيجة واحدة في مربع نصوص تم حفظ قيمته في المتغير food .  
وقامت بطباعتها بواسطة الدالة echo .

نظرا لاننا استخدمنا الاسلوب GET فقد تم اعطاءنا عنوان الصفحة بالاضافه الى (?) وايضا المعلومات المسجله في المتغيرات  
والتي تم استخدام الURL ENCODING فيها لانها تستخدم حروف عربه .

### مربعات النصوص الكبيره (text area) طلبات اكبر للطعام الشههي !

إذا كنت تريد أن تكتب رسالة متعددة الأسطر فإنك تحتاج إلى أداة تحكم تختلف تماماً عن مربع النص العادي وهي مربعات  
النصوص الكبيرة التي يمكنك فيها من إدخال نصوص كبيرة الحجم ومتعددة الاسطر .

تستخدم هذه الأداة وسم فتح ووسم إغلاق

<TEXTAREA>

</TEXTAREA>

ويمكنك تحديد حجمها بواسطة تحديد الصفوف بالخاصية rows والأعمدة بالخاصية cols .

### تمرين عملي

1- قم بفتح محرر النصوص لديك

2- قم بكتابة الكود التالي :

```
<html dir="rtl" >
```

```
<FORM ACTION = "TAREA.PHP" METHOD="POST">
```



ما هي وجبتك المفضلة ؟

```
<br>
<TEXTAREA NAME = "food" ROWS="10" COLS = "50" >
جينة
مربي
مكرونة
بيف برغر
سمبوسة
معصوب
مطبق
ماشادونا
ماخلص لو قعدت اكتب هاها
</TEXTAREA>
<br>
<INPUT TYPE = SUBMIT VALUE ="قم بإرسال الطلبات إلى الجرسون">
</FORM>
</html>
```

- 3 قم بحفظ الملف باسم TAREA.html .
- 4 الآن قم بفتح ملف جديد في محرر النصوص .
- 5 قم بكتابة الكود التالي :

```
<html dir="rtl">
وجبتك المفضلة هي :
<br>
<?
Echo $food;
?>
</html>
```

- 6 قم بحفظ الملف باسم tarea.php
- 7 قم بوضعهما في مجلد السيرفر لديك .
- 8 قم بتشغيل البرنامج .

<http://localhost/tarea.html>

- 9 قم بضغط الزر لارسال البيانات .
- 10 شاهد النتيجة.

### الشرح

لانضيف شيئاً على قولنا هنا سوى أننا نريدك أن تلاحظ كيف جهزنا القيمة الافتراضية بكتابة نصوص بين وسومات الـ textarea وأيضا أننا استخدمنا الأسلوب POST في ارسال البيانات مما جعلها لا تظهر في شريط العنوان .  
وأن الـ NAME تحدد اسم المتغير التي ستذهب إليه القيمة واسم المتغير في الكود لا يحتوى على \$ لأنه كود HTML وليس PHP .

### مربعات الاختيار (Check Box) اكثر من خيار في وقت واحد !

في الواقع قد نرى مربعات الاختيار في صفحات الويب عندما نريد الاشتراك في موقع معين لرؤية محتوياته أو عندما نريد تسجيل بريد إلكتروني أو حجز مساحة عند موقع .  
وفائدتها هي إتاحة فرصة للمستخدم لتحديد أنواع الأشياء التي يريد أن يشترك فيها مثلاً أو إتاحة فرصة له لقبول إتفاقية أو غير ذلك أو رفض الجميع أو قبول الجميع .

يمكننا صنع مربع العلامة بواسطة الوسم INPUT

```
<INPUT TYPE = "CHECKBOX" NAME = "swalif" value= "سوالف" checked>
```

نقوم بتحديد نوع الأداة بأنها مربع علامة في هذا الجزء

```
TYPE = "CHECKBOX"
```

نقوم بتحديد اسم المتغير في هذا الجزء

```
NAME = "swalif"
```

ونقوم بتحديد القيمة التي يتم وضعها في المتغير اذا قام المستخدم باختيار مربع العلامة في هذا الجزء :

```
value= "سوالف"
```

إذا لم تقم بوضع الخيار value فستكون القيمة الافتراضية هي on عند اختيار المستخدم مربع العلامة وستكون فراغ إذا لم يقم المستخدم باختيار المربع.

ونقوم بوضع القيمة الافتراضية بإضافة الكلمة checked فإذا تم وضع هذه الكلمة يكون مربع العلامة مختار تلقائياً أما إذا لم نكتبها فسيكون بدون علامة الاختيار .

Checked

### تطبيق عملي (1) :

1- قم بفتح المفكرة وقم بكتابة الكود التالي :

```
<html dir="rtl">
<FORM ACTION="CHECK.PHP" METHOD = "POST">
    مالذي تريد أن تفعله في الحياة ؟ (يمكنك اختيار أكثر من إجابة )
<br>
<INPUT TYPE="CHECKBOX" NAME = "WIFE" CHECKED>
الذي أريد أن أفعله في الحياة هو أنني أتزوج وأخلص وافتك من الزهق.
<br>
<input type= submit value = "إرسال">
</FORM>
</html>
```

2- قم بحفظ الملف باسم check.html .

3- قم بفتح ملف جديد في المفكرة وقم بكتابة التالي :

```
<?
Echo $WIFE ;
?>
```

4- قم بحفظ الملف باسم check.php .

5- قم بنقل الملفين الى مجلد السيرفر .

6- اكتب في المتصفح

<http://localhost/check.html>

7- النتيجة

### تطبيق عملي (2) :

1- افتح المفكرة واكتب الكود التالي وقم بحفظه في ملف جديد باسم check2.html

```
<html dir="rtl">
<FORM ACTION="CHECK2.PHP" METHOD = "POST">
    مالذي تريد أن تفعله في الحياة ؟ (يمكنك إختيار أكثر من إجابة )
<br>
<INPUT TYPE="CHECKBOX" NAME = "WIFE" value="زوجة" CHECKED>
الذي أريد أن أفعله في الحياة هو أنني أتزوج وأخلص وافتك من الزهق.
<br>
<INPUT TYPE="CHECKBOX" NAME = "jihad" value= "جهاد" >
أبغى أروح الجهاد واخمع رؤوس الكفرة والمشركين
<br>
<INPUT TYPE="CHECKBOX" NAME = "qran" value= "قران" CHECKED>
والله لو ألتحق بتحفيظ قرآن واحفظ القران كامل وأطبقه في عملي وحياتي حرتاح في حياتي كثير
<br>
<input type= submit value = "إرسال">
</FORM>
</html>
```

2- قم بفتح ملف جديد وقم بوضع الكود التالي فيه :

```
<html dir = "rtl">
<?
Echo $WIFE . " " . $jihad . " " . $qran ;
?>
</html>
```

3- قم بحفظه باسم check2.php

4- قم بتشغيل الملف .

5- النتيجة

### تطبيق عملي (3)

1- افتح محرر النصوص واكتب الكود التالي :

```
<html dir="rtl">
<FORM ACTION="CHECK3.PHP" METHOD = "POST">
مالذي تريد أن تفعله في الحياة ؟ (يمكنك إختيار أكثر من إجابة )
<br>
<INPUT TYPE="CHECKBOX" NAME = "alswalif[]" value= "زوجة" CHECKED>
الذي أريد أن أفعله في الحياة هو أنني أتزوج وأخلص وافتك من الزهق.
<br>
<INPUT TYPE="CHECKBOX" NAME = "alswalif[]" value= "جهاد" >
أبغى أروح الجهاد واطمع رؤوس الكفرة والمشركين
<br>
<INPUT TYPE="CHECKBOX" NAME = "alswalif[]" value= "قران" CHECKED>
والله لو ألتحق بتحفيظ قرآن واحفظ القران كامل وأطبقه في عملي وحياتي حرتاح في حياتي كثير
<br>
<input type= submit value = "إرسال">
</FORM>
</html>
```

2- قم بحفظه باسم check3.html وافتح محرر النصوص من جديد واكتب الكود التالي :

```
<html dir="rtl">
<?
Echo "$alswalif[0] <br>" ;
Echo "$alswalif[1] <br>" ;
Echo "$alswalif[2] <br>" ;
?>
</html>
```

3- قم بحفظه باسم check3.php وقم بنقلهما الى ملف السيرفر .

4- قم بتشغيل البرنامج

<http://localhost/check.html>

5- قم بضغط زر ارسال وانظر للنتيجه

### الشرح

في الواقع لقد قمنا بتطبيق ثلاث تمارين [التمرين الاول](#) أردنا لفت النظر إلى أننا قمنا بعدم استخدام value للمتغير وتم إعطاء القيمة on عند اختيار المستخدم مربع العلامة بالإضافة أن مربع العلامة كان مختاراً بسبب وضعنا الخاصية CHECKED ولكن التمرين غير عملي وغير جيد بدون وضع قيم VALUE عند وضعنا لأكثر من مربع اختيار لذلك فقد قمنا بإضافه قيم يتم وضعها في المتغيرات عند اختيار المستخدم لها كما في [التمرين الثاني](#) وأردنا لفت النظر في التمرين الى شى يسمى بالمصفوفات فإذا أردنا مثلاً أن نجعل اسم المتغير متشابهاً وإجراء عمليات تكون أسرع عليه نستخدم المصفوفات ولن نتطرق إلى المصفوفات حالياً ولكن أردنا لفت نظرك فقط وسنقوم بالتكلم عن المصفوفات بالتفصيل في الدروس القادمة باذن الله هي والتكرارات بعد التكلم عن العبارات الشرطية في الPHP .

### ازرار الراديو(RADIO BUTTONS) (اختر المشروب المفضل !)

ماهو اختيارك المفضل ؟ علماً بأنه لايمكنك اختيار أكثر من خيار واحد !!

في الواقع إن زر الراديو يتيح لك أن تختار شى واحد من بين عدة اختيارات ونراه كثيراً عند اتفاقيات البرامج حيث يعطيك فرصة إما بقبول الإتفاقية أو رفضها ويكون واحد من الاختيارين محدد (وهو خيار الرفض) .

يتم استخدام ازرار الراديو باستخدام العبارة <INPUT> كالتالى :

```
<INPUT TYPE = "radio" NAME = "name" value= "value" checked>
```

نقوم بتحديد نوع الكائن بأنه زر راديو في هذا الجزء :

```
TYPE = "radio"
```

نقوم بتحديد اسم المتغير في هذا الجزء :

NAME = "name"

نقوم بتحديد القيمة التي ستكون في المتغير هنا :

value= "value"

في الواقع مع ازار الراديو نقوم بجعل اسم المتغير name هو نفسه والقيم مختلفة value لكل سؤال . وإذا لم نضع قيمة فسيقوم PHP بوضع القيمة on للمتغير .

### تطبيق عملي :

1- قم بتشغيل محرر النصوص لديك واكتب الكود التالي و قم بحفظه في ملف اسمه radio.html .

```
<html dir="rtl">
<form action = radio.php method = "post">
ماهو مشروبك المفضل ؟
<br>
<br>
<INPUT TYPE = "radio" NAME = "mshroob" value= "شاي" checked>
شاي
<br>
<INPUT TYPE = "radio" NAME = "mshroob" value= "قهوة" >
قهوة
<br>
<INPUT TYPE = submit value= "إرسال" >

</form>
</html>
```

2- قم بفتح محرر النصوص واكتب الكود التالي و قم بحفظه باسم radio.php

```
<html dir = "rtl">
<?
echo " مشروبك المفضل هو " . " " . $mshroob;
?>
</html>
```

3 - قم باختيار المشروب المفضل واختر إرسال .

### الشرح :

في الواقع لقد قمنا بصنع أزرار راديو ولقد قمنا بوضع قيمة لكل زر تكون تابعة للعبارة التي بجوار الزر . ولقد قمنا بوضع عبارة checked لكي ترى كيف أن الأداة التي تحتوي على العبارة تكون محددة تلقائياً ولاحظ أن العبارة التي تكون بجانب الزر تكون موجودة أسفل كود الزر مثل :

```
<INPUT TYPE = "radio" NAME = "mshroob" value= "شاي" checked>
شاي
```

العبارة هي الملونة باللون الأحمر .  
وأيضا لاحظ أننا استخدمنا متغيراً واحداً فقط لجميع الاختيارات بحيث أن جميع الأزرار قيمتها تعود إلى هذا المتغير .

### القوائم (Lists Or drop down menus) اختر مواصفات زوجتك للمستقبل واسمها :

تستخدم القوائم في ال html بشكل مختلف قليلاً عن الأدوات السابقة إذ أننا نستخدم وسمين من وسوم لغة html وهما :  
<select> لنقوم بإنشاء القائمة و <OPTION> ونستخدم الخاصية MULTIPLE إذا كنا نريد إتاحة الفرص للمستخدم أن يختار أكثر من قيمة ونقوم بوضع القيمة التي يختارها المستخدم في متغير بواسطة الخاصية NAME أو في مصفوفة متغيرات (وسيتضح مفهوم المصفوفات لديك جيداً في درس المصفوفات بأذن الله .

### تطبيق عملي :

1- قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه في ملف باسم lists.html :

```
<html dir="rtl">
<form action = "lists.php" method = "post">
ماذا تريد ان يكون اسم زوجة المستقبل (لغير المتزوجين ) ؟
<br>
<select name = "wife" >
<option> هناء </option>
<option> جمانة </option>
<option> رزان </option>
```

```

<option>سحر</option>
<option>سارة</option>
<option>سمية</option>
<option>روان</option>
<option>دلال</option>
<option>اسم اخر</option>
</select>
<br>
ماذا تريد أن تكون مواصفاتها ؟
<br>
<select name="dis[]" multiple>
<option>جميلة</option>
<option>متدينة</option>
<option>شغراء</option>
<option>جعداء الشعر</option>
<option>سوداء</option>
<option>سمراء</option>
<option>بيضاء</option>
</select>
<br>
<INPUT TYPE=SUBMIT VALUE="إرسال">
</html>

```

2- قم بفتح ملف جديد واكتب فيه الكود التالي وقم بحفظه باسم lists.php :

```

<html dir="rtl">
<?
Echo "$wife . " " . " لقد أردت أن يكون اسم زوجتك ";
Echo "<br><br>";
Echo "ولقد أردت أن تكون مواصفاتها ";

Echo "<br><br>";
Echo "$dis[0] <br>";
Echo "$dis[1] <br>";
Echo "$dis[2] <br>";
Echo "$dis[3] <br>";
Echo "$dis[4] <br>";
Echo "$dis[5] <br>";
Echo "$dis[6] <br>";
?>
</html>

```

قم بتشغيل البرنامج

<http://localhost/lists.html>

واختر ماتريد ثم اضغط زر ارسال

### الشرح :

لقد قمنا بصناعة قائمة تسمح باختيار قيمة واحدة منها ثم تذهب هذه القيمة الى المتغير wife وصنعنا قائمة ثانياه تسمح باختيار أكثر من عنصر واحد وقمنا بوضع هذه القيم في مصفوفه متغيرات (سيتضح معني المصفوفات في دروس قادمه ان شاء الله ) .

### الاداة الخفيه (والمعلومات السريه!) (hidden control)

هناك بعض الأوقات تحتاج فيها إلى إرسال بعض المعلومات من صفحة ويب الى صفحة ويب اخرى عن طريق النماذج وفي نفس الوقت أنت لاتريد المستخدم أن يقوم برؤية هذه المعلومات .

في الواقع هناك أداة تساعدك على اخفاء هذه المعلومات على المستخدم يسمونها بحقل النموذج المخفي أو الأداة الخفية (hidden form field or hidden control) .

هذه الأداة تلعب دوراً مختلفاً ومتميزاً عن بقية الأدوات وهي اخفاء المعلومات التي تم ادخالها كما شرحنا في السابق وهي مفيدة جدا مع النماذج المصنوع بواسطة الـ PHP إذ أنها تسمح لنا أيضا بان تكون المعلومات المخفيه هي متغيرات . PHP

يتم صنع هذه الحقول المخفية كالتالي :

```
<INPUT TYPE=HIDDEN NAME =hidden1 VALUE="الرسالة السرية">
```

نقوم بوضع HIDDEN لكي يعرف المتصفح أن هذه المعلومات خفية (لا تظهر للمستخدم) ونضع اسما للمتغير الذي يقوم بالاحتفاظ بالمعلومات والذي يتخزن اسمه في الـNAME ونقوم بوضع المعلومات التي نريد اخفاءها في الـVALUE .

نستطيع الاستفادة أيضا منها عن طريق الـphp وذلك عن طريق كتابة كود الـHTML بواسطة الأمر (echo) في الـPHP كما في المثال التالي :

```
<?
$msg1= " هذه العبارة لن تظهر ";
Echo "<form>";
Echo "<input type=hidden name =secret value= '$msg1'>";
Echo "<input type=submit>";
Echo "</form>";
?>
```

هذا الكود الذي تراه عبارة عن كود HTML تم كتابته بالـPHP عن طريق الأمر (echo) ولقد استطعنا تخزين قيمة متغير الـphp (\$msg1) في متغير html (secret) .

### تطبيق عملي :

1 - افتح محرر النصوص واكتب الكود التالي واحفظه باسم hid.php :

```
<html dir="rtl">
<head></head>
<body>
<?
$car1= "لكزس";
$car2= "ماكسيما";
$car3="لاندكروزر";
Echo "<form method =get action='hid2.php'>";
Echo "ماهي السيارة التي تتمنى أن تشتريها أو تحظى بها ؟";
Echo "
<select name= 'favcar'>
<option>$car1</option>
<option>$car2</option>
<option>$car3</option>
</select><br><br>
<input type =hidden name = hid1 value='$car1'>
<input type =hidden name = hid2 value='$car2'>
<input type =hidden name = hid3 value='$car3'>
<input type = submit value='إرسال'>
</form>";
?>
</body>
</html>
```

3- افتح محرر النصوص واكتب الكود التالي واحفظه باسم hid2.php

```
<html dir="rtl">
<head></head>
<body>
<?
Echo "<br>لقد قمنا بعرض السيارات التالية عليك :";
Echo "$hid1<br>";
Echo "$hid2<br>";
Echo "$hid3<br>";
Echo "<br>ولقد قمت باختيار:"<br>";
Echo $favcar;
?>
</body>
</html>
```

3- قمت بنقل الملفين الى مجلد السيرفر ثم قم بتشغيل السكريبت :

<http://localhost/hid.php>

### الشرح :

لقد قمنا بعمل نموذج بسكريبت الـ php لاحظ أننا استخدمنا الـ() بدلاً من الـ() كما كنا نعمل في الـ html وذلك لاننا قلنا سابقاً أن القيم الحرفية ..... (راجع درس المتغيرات) ولقد قمنا بإدراج قيم متغيرات الـ php في كود الـ html مما يوفر علينا الكثير من إعادة الكتابة (في حال كان النص المستخدم طويلاً) .  
اقرأ المثال أكثر من مر وسيوضح لك المقال أكثر باذن الله .

### استخدام حقول كلمات السر (Password fields)

لكي تجعل المعلومات أكثر حماية من التعرض إلى السرقة أو غير ذلك يمكنك استخدام حقول كلمات السر الذي هو عبارة عن مربع نص بسيط يقوم بإظهار النص على شكل نجوم \*\*\*\* في حال كان الجهاز يستخدم على يد أكثر من شخص فان هذه الطريقة جيدة قليلاً في أن لا يري شخص معلومات الآخر السرية .  
في الواقع مع ذلك فإنك لاتكون قد اديت حماية إذا كان الـ اسلوب المستخدم في ارسال بيانات المستخدم هو الـ اسلوب الـ get إلا إذا كنت تستخدم تشفير البيانات ويكون أكثر جودة إذا استخدمت الـ اسلوب الـ post وايضا لن يكون محمياً من الهاكر إذا لم تكن تستخدم الـ SSL (Secure Socket Layer) لكي تقوم بتنشيط تشفير البيانات .

### تطبيق عملي

قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه باسم pass.php

```
<html dir="rtl">
<body>
<form method=post action="pass1.php">
اسم المستخدم
<br>
<input type="text" name ="user">
<br>
كلمة المرور
<input type="password" name ="pass">
<br>
<input type = submit value='إرسال'>
</form>
<body>
</html>
```

قم بفتح محرر النصوص لديك واكتب الكود التالي واحفظه باسم pass1.php

```
<?
Echo " اسم المستخدم هو : ";
Echo "<br>$user<br>";
Echo " وكلمه المرور هي : ";
Echo "<br><br>$pass"
<?
```

قم بنقل الملفين الى مجلد السيرفر لديك  
قم بتشغيل البرنامج ولاحظ النتيجة .

### ارسال البريد الالكتروني بواسطة الـ php :

البريد الإلكتروني هو الحياة التي تنبض بها السكريبتات فمثلاً هناك سكريبتات ارسال بريد الى صاحب الموقع تخبره بشي معين أو ملحوظة أو غير ذلك ويمكن استخدامها في أكثر من مجال .  
والدالة التي تستخدم في ذلك هي الدالة mail()

```
mail("$to", "$sub", "$msg", "From:$you");
```

وتقوم بوضع بريد الذي ستصله الرسالة في الخانة \$to وموضوع الرسالة في الخانة \$sub والرسالة في الخانة \$msg وبريدك أنت أو بريد المرسل في الخانة \$you .

```
<html dir=rtl>
<head>
  <title>برنامج إرسال بريد</title>
</head>
<body>
<form action="mail.php" method="post">
عنوان المرسل
<br>
<input type="text" name = "you">
<br>
عنوان المستقبل
<br>
<input type="text" name = "to">
<br>
موضوع الرسالة
<input type="text" name = "sub">
<br>
الرساله
<textarea rows=10 cols=20 name = "msg" >
</textarea>
<input type="submit" value = "إرسال البريد الالكتروني">
</form>
</body>
</html>
```

قم بإنشاء ملف آخر وقم بكتابة الكود التالي وقم بإعطائه الاسم mail.php .

```
<?
mail("$to", "$sub", "$msg", "From:$you");
?>
```

قم بوضع الملفين في مجلد السيرفر وقم بتشغيل البرنامج واملأ البيانات واضغط زر الإرسال وستري ان الرساله تم ارسالها بنجاح .

## برامج عمليه

برنامج او سكربت ارسال بطاقات بسيط يحتوي من ملفين الأول به البطاقات وعنوان البريد الإلكتروني والملف الثاني هو الذي يقوم بعملية الإرسال

الملف الأول هو chcard.php وكوده كالتالي :

```
<html dir="rtl">
<form action =card.php method = "post">
اختر البطاقة التي تريد ارسالها
<br>
<br>
<INPUT TYPE = "radio" NAME = "card" value= " http://www.khalaad.f2s.com/MADINA9_small.JPG" checked>
البطاقه الاولى
<br>
<br>

<br>
<br>
<INPUT TYPE = "radio" NAME = "card" value= " http://www.khalaad.f2s.com/Haram3.jpg" >
البطاقه الثانيه
<br>
<br>

<br>
```



```

اسمك
<br>
<input type="text" name = "myname">
<br>
بريدك الالكتروني
<br>
<input type="text" name = "you">
<br>
بريد صديقك
<br>
<input type="text" name = "to">
<br>
موضوع التهنينه
<input type="text" name = "sub">
<br>
الرساله
<br><br>
<textarea rows=10 cols=20 name = "msg" >
</textarea>
<br>
<INPUT TYPE = submit value= "إرسال البطاقة" >
</form>
</html>

```

الملف الثاني يقوم بعملية ارسال البطاقة وتقوم بكتابة الكود التالي واحفظه في ملف باسم card.php

```

<?
$message = " \n$msg".\n وهو يقول في نص رسالته لك " \n". " \n". " \n";
mail("$to", "$sub", "$message", "From:$you");
echo "<center>مبارك،،، لقد تم ارسال الرساله بنجاح</center>";
?>

```

**ملاحظة :**

الدالة \n تقوم فقط ببدء سطر جديد لاننا لانستطيع استخدام <br> في نص الرسالة

## الأوامر الشرطية

لقد أخذنا في الدروس السابقة فكرة عن المتغيرات وكيفية تعامل البيانات مع النماذج .. في هذا الدرس سنتعلم كيفية التحكم بالكود بمعنى تنفيذ سطر معين من الكود عند حصول شرط معين وعند عدم حصوله نتجاهل السطر ونتجه الى السطر الذي يليه .. هذا يمنحنا تحكما أكبر بالكود ويجعلنا نستخدم قرارات وتنفيذ أشياء ممتازة وبرامج رائعة بالـ PHP .

**دعنا نعطيكم فكرة من حياتنا اليومية ....**

تقوم في الصباح وتريد أن تحضر فطورك الذي يتكون من التالي :

عسل  
جبنه  
خبز  
شاي

سنقوم بالذهاب إلى التلاجه ثم نقوم بالبحث عن الأشياء التي يتكون منها فطورك ، فإذا لم تجد ما تريد تستعد للذهاب إلى المركز التجاري لشرايه حاجتك ، نذهب إلى المطبخ وتأكد مره أخرى وتبحث عن المؤونه التي يحتاجها البيت بشكل عام .

- 1- تبحث عن جبنه وإذا لم تجدها تنتقل إلى الخطوة 3 .
- 2- إذا وجدت جبنه فإنك تبحث عن العسل فإذا وجدته تنتقل إلى الخطوة 4 ، وإذا لم تجده تنتقل إلى الخطوة 5 .
- 3- تقوم بكتابتها في ورقة جانبية وتقوم بالبحث عن العسل .
- 4- تتجهز للذهاب إلى المركز التجاري .
- 5- تكتبه في ورقه جانبيه ثم تتجهز للذهاب إلى المركز التجاري .

هل لاحظت انك كنت تقوم بالبحث عن أشياء معينة فاذا وجدتها ( true ) قمت بالبحث عن التي تليها وإذا لم تجدها ( false ) تقوم بتسجيلها في قائمة المشتريات لديك .

## القيم المنطقية والدوال الشرطية

في الواقع لقد تكلمنا عن المتغيرات سابقاً وذكرنا بأن هناك متغيرات منطقية (قيمتها إما صحيح إم خطأ) ولم نعلم بشرحها ، وهذا الدرس سيتولى شرحها وإعطاء أمثلة على كيفية التعامل معها .

## العبارة IF

IF condition is true ( إذا كان الشرط صحيحاً )

```
{
  excute this code (قم بتنفيذ هذا الكود)
}
```

إن الدالة IF معروفة تقريباً في جميع لغات البرمجة ...حيث أنها تقوم بعملية التحقق من شيء معين وتنفيذ بعض الأشياء إذا كان الشرط صحيحاً (true) والقيام بتنفيذ أشياء أخرى إذا لم يكن صحيحاً .... سيقوم الـPHP بتنفيذ الكود التي بين { و } فقط إذا كان الشرط صحيحاً . أما إذا لم يكن صحيحاً فسيقوم بتجاوزه وتنفيذ الكود الذي يليه . ويمكنك أيضاً أن تقوم بجعلها بسطر واحد ولا تستخدم الأقواس بل تكتب الأمر مباشرة :

```
IF condition is true excute function;
```

لاحظ أنه لا بد من استخدام { و } إذا كان الكود يتكون من عدة أسطر أما إذا كان يتكون من سطر واحد فلا داعي لاستخدامها .

فالمثالين التاليين كليهما صحيحين  
مثال(1)

```
<?
$$=10
IF ($$=10) echo 11;
?>
```

مثال (2)

```
<?
$$=10
IF ($$=10){
  echo 11;
}
?>
```

لنتخيل مثلاً أن الجو ممطر وسنقوم بإعطاء المطر متغيراً ونسميه rain ونقوم بإعطاء المظلة اسم متغير آخر ونسميه umbrella وسنقوم بإفترض أن هناك أمر في الـphp يسمى go out حسناً الآن الكود الذي نريد أن نقوم بكتابته هو :

```
If $rain = true
{
  $umberrlla = true
}
go out();
```

فائدة هذا الكود هو أن تأمر الـPHP بحمل المظلة (\$umberrlla=true) معه إذا كان الجو ممطراً (\$rain=true) وإذا لم يكن ممطراً ولم يتحقق الشرط فإنه سيخرج إلى النزهة بدون أي مظلة .

طبعاً ليس هناك دالة تقوم بذلك إنما قمنا بذلك من أجل التوضيح للمستخدم هيكلية عمل الدالة بشكل عام .

## مقدمه الى القيم المنطقية (Boolean Values)

القيم المنطقية ترمز إلى الأشياء التي لا تتحمل أكثر من احتمالين وهما إما صح وإما خطأ ، وهي نوع جديد من القيم غير التي كنت نعرفها سابقاً (مثل الرقميه والنصيه) .

مثال

```
<?
$variable=true;
echo "$variable";
?>
```

لو قمت برؤيه النتيجة ستجد أنه يطبع الرقم واحد وهو قيمة المتغير إذا كان صحيحاً ، أما إذا كان خطأ أو غير صحيح فقيمه ستكون ( 0 ) .

## المعاملات المنطقية

لقد أخذنا المعاملات الرياضية فيما سبق بشيء من التفصيل (+،-،\*) والآن سنأخذ شيئاً جديداً من المعاملات وهي المعاملات المنطقية التي تساعدنا في صناعة الشروط والتقييدات على شيء معين وتعطينا تحكما أكبر في الكود .

### المعاملات : < و >

من المفترض أن تكون متآلفاً مع علامتي الأكبر من والأصغر من في الرياضيات التي تتعلمها في المدرسة مما يجعل فهم هذا الأمثلة بسيطاً .

```
<?
If (6>5)
{
echo "الرقم ستة أكبر من الرقم خمسة ";
}
Echo "end";
?>
```

سيقوم الـ PHP في مثالنا هذا بفحص الشرط (6>5) فإذا كان صحيحاً (true) سيقوم بطباعة السطر (الرقم ستة أكبر من الرقم خمسة) ثم يقوم بطباعة end ، وإذا لم يكن صحيحاً فسيقوم بتجاهل الكود وطباعة (end) فقط .

يمكننا أيضاً استعمالها في المقارنة بين متغير ورقم أو بين متغير وثابت (constant) أو العكس أو المقارنة بين متغيرين .

مثال (1)

```
<html dir="rtl">
<?
$LuckeyNumber = 5;
If ($LuckeyNumber<6)
{
echo ("رقم الحظ أصغر من الرقم ستة ");
}
?>
```

مثال (2)

```
<html dir="rtl">
<?
$f=5;
$r=10;
If ($f >$r)
{
echo " أكبر من المتغير $f$ المتغير ";
}
?>
```

## تطبيق عملي :

قم بتشغيل محرر النصوص واكتب الكود التالي واحفظه باسم thegame.php

```

<html dir = "rtl">
<body>
<form method =get action="game.php">
ماهو الرقم الذي أفكر به الآن والذي هو بين 1 و10 ؟
<input type="text" name="number">
<br>
<br>
<input type =submit>
</form>
</body>
</html>

```

قم بفتح محرر النصوص لديك من جديد واكتب الكود التالي واحفظه باسم game.php

```

<html dir="rtl">
<body>
<?
$num = rand (1,10);
if ($number>$num)
{
echo "لقد اخترت رقم أكبر من الذي أفكر فيه" ;
Echo " الرقم الذي أفكر فيه هو " ;
Echo $num;
Echo "<br>". "يوسفنا فعلاً أنك لم تنجح ، نتمنى أن نقول لك في المرات القادمة " .
}
if ($number<$num)
{
echo "لقد اخترت رقم أصغر من الذي أفكر فيه" ;
Echo "$numالرقم الذي كان في مخيلتي هو " ;
Echo "<br>". "يوسفنا فعلاً أنك لم تنجح ، نتمنى أن نقول لك في المرات القادمة " .
}
?>
لقد نجحت
</body>
</html>

```

## شرح التطبيق :

الدالة rand

تقوم هذه الدالة باختيار رقم عشوائي من بين رقمين يتم اعطاؤها إياها الرقم الأول (x) هو الأصغر والرقم الثاني هو الأكبر (y)

`Rand (x,y);`

يمكنك حفظ القيمة التي تقوم بإخراجها هذه الدالة في متغير مباشرة

مثال

`$Num = rand (5.57);`

وهذا يوضح ماقمنا به في الكود

`$num=rand(1,10);`

لقد قمنا باختيار قيمة عشوائية ثم قمنا بمقارنتها مع القيمة التي تم إدخالها من قبل المستخدم فإذا كانت القيمة التي أدخلها المستخدم أكبر من قيمة العدد العشوائي أخبرناه بأن الرقم الذي أدخله أكبر من الرقم الصحيح ... وهذا ماتجده جليا في الأسطر التالية :

```

if ($number>$num)
{
echo "لقد اخترت رقم أكبر من الذي أفكر فيه" ;
Echo " الرقم الذي أفكر فيه هو " ;
Echo $num;
Echo "<br>". "يوسفنا فعلاً أنك لم تنجح ، نتمنى أن نقول لك في المرات القادمة " .
}

```

فإذا لم ينطبق الشرط وكان الرقم الذي اختاره المستخدم أصغر من الرقم العشوائي فإنه يترك الشرط الأول ويتجه الى الشرط الثاني ويطبق الأوامر التي فيه والتي تقوم بإخباره بأن الرقم الذي قام باختياره أصغر من الرقم المطلوب ، وهذا ما تجده جلياً في الأسطر التالية :

```
if ($number < $num)
{
echo "لقد اخترت رقم أصغر من الذي أفكر فيه" ;
Echo "$num الرقم الذي كان في مخيلتي هو " ;
Echo "<br>". "يؤسفنا فعلاً أنك لم تنجح ، نتمنى أن نقول لك في المرات القادمة " ;
}
```

فإذا لم ينطبق الشرطين فإنه يتركهما ويكتب الكلمة (لقد نجحت ) بدون أي كلمات أخرى مثلما كنا نكتب الكلمة (يؤسفنا فعلاً أنك لم تنجح ، نتمنى أن نقول لك في المرات القادمة ) قبل كلمة (لقد نجحت ) ، أتمنى أنك قد فهمت جيداً ما أقول .... وتظهر هذه العبارة جلية في الأسطر التالية :

```
?>
لقد نجحت
</body>
</html>
```

على هذا نكون قد صنعنا لعبة كاملة تقوم بإخبار المستخدم عند نجاحه او خسارته .

### معاملات المساواة : == و ===

لقد قمنا باستخدام علامة المساواة الفردية سابقاً في تخزين قيمة في متغير وهانحن نأخذ نوعاً من علامات المساواة وهو علامة المساواة المزدوجة (==) وعلامة المساواة المضاعفة (===) .

لقد كنا نستخدم علامة المساواة الفردية او العادية في تخزين القيم في المتغيرات .

مثال :

```
<?
$m=12;
?>
```

ولكن العلامات التي نتكلم عنها الآن تستخدم في تحديد إذا ماكانت قيمة معينة تساوي قيمة أخرى .

مثال :

```
<?
$m="11";
$u=11;
If ($m==$u)
{
Echo "القيم متساوية " ;
}
?>
```

لاحظ أن \$m متغير حرفي وان \$u متغير رقمي . إذا كنا نريد ارجاع قيمة إلى متغير نستخدم علامة المساواة العادية (=) وإذا أردنا اختبار متغيرين أو قيمة معينة من أنها متساوية نقوم باختبار القيم بواسطة علامة المساواة المزدوجة (==) .

في الـ php4.01 تم إصدار علامة مساواة جديدة تقوم باختبار القيم ولا تعطي القيمة (true) إلا إذا كانت أنواع القيم متساوية وأنواع البيانات في المتغيرات ايضاً متساوية .

مثال (1) :

```
<?
$m="11";
$u=11;
If ($m==$u)
{
Echo "القيم متساوية";
}
?>
```

مثال (2) :

```
<?
$m="11";
$u=11;
If ($m=== $u)
{
Echo "القيم متساوية";
}
?>
```

## التوضيح

لاحظ أننا في المثال الأول استخدمنا علامة المساواة المزدوجة لاختبار القيم وكانت القيم متساوية في المتغيرين فتم طباعة أن القيم متساوية (مع أن نوع البيانات مختلف) ولكن في المثال الثاني عندما استخدمنا علامة المساواة المضاعفة لم يتم طباعة أي شيء وذلك لأن القيم متساوية ولكن نوع البيانات مختلف فالمتغير \$m حرفي بينما المتغير \$u رقمي .

### المعاملات : != و <>

إن عكس علامة المساواة هي علامة عدم المساواة (!=)

مثال :

```
<?
If (5!=99) echo "القيم غير متساوية";
?>
```

لاحظ أن 5 لا تساوي 99 لذلك فإن الشرط صحيح (true) لذلك قام بطباعة أن القيم غير متساوية .

إن الضد من علامة أكبر من وأصغر من هو علامة الـ (<>) وهو يقوم بإرجاع قيمة (true) إذا كانت القيمتين مختلفتين عن بعضهما أي أنه مثل علامة != تقريباً .

مثال :

```
<?
If (5<>99) echo "القيم غير متساوية";
?>
```

## تطبيق عملي على علامات المساواة وعدم المساواة

قم بفتح محرر النصوص لديك و اكتب الكود التالي :

```
<html>
<head></head>
<body>
<Form method =get ACTION= "quiz.php">
ما هو اسم الرجل الذي يسمى بالفاروق ؟
<br><br>
<input type ="radio" name = "man" value="عمر">
عمر بن الخطاب رضي الله عنه
<br>
<input type ="radio" name = "man" value="أبو بكر">
أبو بكر الصديق رضي الله عنه
```

```

<br>
<input type = "radio" name = "man" value="عثمان">
عثمان بن عفان رضي الله عنه
<br>
<input type = submit>
</form>
</body>
</html>

```

احفظها باسم quiz.html ...

قم بفتح محرر النصوص لديك واكتب الكود التالي :

```

<html dir="rtl">
<head></head>
<body>
<?
If ($q=="عمر") echo "الإجابة صحيحة";
If ($q!="عمر") echo "الإجابة خاطئة";
?>

```

قم بحفظه باسم quiz.php وضعهما في مجلد السيرفر

قم بتشغيل الملف quiz.html

## المعاملات المنطقية (AND, OR, NOT)

إن هذه المعاملات المنطقية تتيح لك بتنفيذ الكود بعد التحقق من مجموعة شروط وأيضا تنفيذ الكود إذا تحقق أكثر من شرط : (AND) أو تحقق شي معين من بين عدة أشياء : (OR) ويمكنك مثلا التحقق من عدم صحة شي لكي تقوم بتنفيذ شي آخر : (NOT)

فيمكنك مثلا أن تقول : إذا كان الجو ممطرا والعاصفة شديده فلن أخرج من البيت .  
ويمكنك أن تقول : إذا كان الجو هادئا أو لا يوجد أمطار فسأقوم بالخروج الى المنتزه .  
ويمكنك أيضا أن تقول : إذا لم يكن الجو ممطرا سأقوم بالخروج إلى نزهة .

ولكن عند استخدامك لهذه الدوال عليك مراعاة أن تقوم بجعل هذه الشروط بين قوسين .

### المعامل (AND) ونظيره (&&)

يمكننا استعمال المعامل (AND) والمعامل (&&) للتحقق من صحة عدة شروط لتنفيذ شي معين

مثال(1)

```

<?
$w=10;
$g=12;
IF ($w=10 and $g=12) echo ("لقد تحققت جميع الشروط");
?>

```

مثال (2)

```

<?
$w=10;
$g=12;
IF ($w=10 && $g=15) echo ("لقد تحققت جميع الشروط");
?>

```

في المثالين السابقين قمنا بعملية التحقق من أكثر من شرط باستخدام المعاملين (&& و and) فعندما تحققت جميع الشروط تم تنفيذ الأمر .... وعندما لم تكن جميع الشروط صحيحة تم تجاهل الأمر .  
لاحظ أننا قمنا بجعل الشروط بين قوسين ( ) لكي يعمل الكود بشكل صحيح :

```

($w=10 && $g=15)
($w=10 and $g=12)

```

### المعامل (OR) ونظيره (||)

المعامل OR يقوم بالتحقق من عدة شروط وإذا تحقق أي واحد منها فإنه يقوم بتنفيذ الكود ونظيره (||) الذي يقوم بنفس العملية .

مثال (1)

```
<?
$E=100;
$T=8;
IF ($E=14 OR $E=55 OR $E = 10 OR $T=8 ) echo ("لقد تحقق أحد هذه الشروط");
?>
```

مثال (2)

```
<?
$E=100;
$T=458;
IF ($E=14 || $E=55 || $E = 10 || $T=8 ) echo ("لقد تحقق أحد هذه الشروط");
?>
```

إذن عندما تحقق واحد من هذه الشروط تم طباعة السطر (لقد تحقق أحد هذه الشروط) .

ملحوظة قد لا تكون بتلك الأهمية لكن يجب أن تعرف أن الرموز && و || لها الأسبقية والأفضلية على استخدام AND و OR .

### المعامل NOT ونظيره (!)

في الواقع لا يمكنك استخدام NOT أبداً لأنها ليست أصلاً موجودة في لغة PHP لكن يمكنك استخدام المعامل (!) كبديل لها فهو يؤدي نفس وظيفتها وهي القيام بالتأكد من أن هناك قيمة غير صحيحة (FALSE) لكي يتم تنفيذ شيء معين .

```
<?
$F="الفارس";
IF !($F=="نعمان") echo ("أهلاً بك");
?>
```

في المثال السابق يقوم الـ PHP بالتأكد من أن المتغير \$F لا يحتوي على القيمة الحرفية (نعمان) ويتم ذلك باستخدام المعامل (!) وعندما يتم التأكد من ذلك يقوم بطباعة السطر (أهلاً بك)

ونشير إلى أننا عندما نقوم باختبار متغير بواسطة المعامل (!) فإن الـ PHP إذا وجد المتغير فارغاً أو لم يتم إنشاؤه يعطيه القيمة صفر وهي FALSE .

مثال

```
IF (!($R)) echo (10);
```

### استخدام المعاملات < و >=

من المعاملات المعروفة والمشهورة في الرياضيات هي علامتي أصغر من أو يساوي <= أو أكبر من أو يساوي >= وهي تستخدم بنفس وظيفتها بالـ php وهي معرفة إذا ما كانت قيمة أصغر أو أكبر من أو تساوي قيمة أخرى ، وهذه الأمثلة تعطيك مدخلاً أشمل لفهم هذه الدوال :

```
<?
$t = 15;
If ($t >= 10 ) echo ("ممتاز " . "<br>");
$t = 5;
If ($t <= 9 ) echo ("جيد جداً");
?>
```

## تجميع المعاملات

يمكننا في الشرط أن نتحقق من مجموعة من القيم باستخدام مجموعة من المعاملات ، ونقوم بتجميع هذه المجموعات داخل أقواس () مثلما كنا نستخدم سابقاً أكثر من معامل (+ ، - ، / ، \*) باستخدام الأقواس .

وسيبدو ذلك واضحاً وجلياً في مثالنا التالي :

```
<?
$a=10;
$y=5;
$t =29;
If (($a == 10) or ($a==54) and ($y !=25) and ($t >= 11)) echo "تحققت جميع الشروط";
?>
```

سيتم طباعة 18 لأنه قيمة تجميع التعبير السابق تكون صحيحة ولو قمنا بشرح المثال فسنقوم برؤية القسم الأول وهو :

```
($a == 10) or ($a==54)
```

وطبعاً المتغير يحمل القيمة 10 فسيكون هذا الجزء صحيحاً .



ثم نقوم برؤية الجزء :

(\$y !=25) and (\$t >= 11)

وطبعاً تم التحقق من جميع الشروط وتم طباعة الكلمة (تحققت جميع الشروط) .

## تعدد الشروط ( else و else if )

يمكننا استخدام أكثر من هيكلية للعبارة if فهناك مثلاً الهيكلية التالية :

```
If condition is true
{
Excute code
}
Else
{
Excute other code
}
```

وهي تقوم بالتحقق من الشرط فإذا وجدته صحيحاً قامت بتنفيذ الكود الأول وإذا لم تجده صحيحاً ستقوم بتنفيذ الكود الآخر .  
مثال

```
<?
$age=10;
If ($age>18)
{
echo "مرحبا بك في أكبر موقع تجاري إلكتروني";
}
else
{
echo "ممنوع دخول الأطفال الموقع لأنهم لايملكون المال";
}
?>
```

ويمكننا أيضاً استخدام الهيكلية التالية :

```
If condition is true
{
Excute code
}
Elseif
{
Excute other code
}
Else
{
Excute other code
}
```

وهي تقوم بتطبيق أكثر من شرط فإذا لم يكن أي شرط من الشروط صحيحاً سيتم تنفيذ الكود الذي يقع بعد كلمة else . مثال :

```
<?
$age=10;
If ($age<=18)
{
echo "مرحبا بك في أكبر موقع تجاري إلكتروني";
}
elseif ($y >= 44);
{
echo "مافي مشكلة برضه إذا كنت كبير";
}
else
{
echo "ممنوع البقية";
}
?>
```

## تعشيش العبارات الشرطية

يمكنك تعشيش العبارات الشرطية ، وتعني بتعشيش العبارات الشرطية هي أن تقوم بعملية تعشيش الشروط فمثلاً إذا كان شرط ما صحيحاً فإنه يجب أن يكون شرط آخر صحيحاً لكي يتم حصول شيء معين وغير ذلك .  
مثال :

```
<?
$h="ahmed";
$f=45;
If ($h == "ahmed" )
{
    If ( $f== 45)
    {
        echo "الاسم والرقم صحيحان";
    }
    else
    {
        echo (" الرقم غير صحيح");
    }
}
else {
echo " اسم تسجيل الدخول غير صحيح";
}
?>
```

هذا مجرد مثال بسيط جداً لتعشيش الدوال الشرطية حيث يقوم بإجراء اختبار على قيمة معينة ثم يقوم عند تجاوزه ذلك الاختبار بنجاح بإجراء اختبار ثاني فإذا تم تجاوز الاختبار الثاني يتم طباعة الاسم والرقم صحيحان وإذا لم يتم الاجتياز يتم طباعة عبارة الفشل في الاجتياز .

## تطبيق عملي

سنقوم في هذا التطبيق بصناعة مسابقة بسيطة نستخدم فيها ما تكلمنا عنه سابقاً  
1- قم بإنشاء ملف Msabqa.html .  
2- قم بكتابه الكود التالي فيه :

```
<html>
<body>

<form method="POST" action="msabqa.php" dir="rtl">
<br>من هو أول الخلفاء الراشدين
<p><br><input type="radio" value="abubaker" name="s">أبو بكر الصديق<br><input
type="radio" value="3mar" name="s">عمر
<br><input type="radio" value="3thman" checked name="s">عثمان<br><br><br>
</p>

<p><input type="submit" value=" ارسال " > <input type="reset "
value="حذف"></p>
</form>

</body><html>
```

قم بفتح ملف وقم بتسميته msabqa.php

```
<?
<html dir = "rtl">
If $s == "3mar"{
الإجابة صحيحة
}
else
{
echo "الإجابة خاطئة";
}
?>
```

## العبارة Switch

```
Switch (VARIABLE) {  
  CASE THING1 :  
    Excute code ;  
    break;  
  CASE THING2 :  
    Excute code ;  
    break;  
  Default;  
    Excute code ;  
}
```

تقوم العبارة بنفس عملية العبارة if ولكن بهيكلية أسهل ومحبية أكثر وتتيح لك اختبار قيمة متغير وإجراء أكثر من اختبار عليه .

**break;**

تقوم بالخروج من عبارته معينه مثل switch و if والذهاب الى الأوامر والعبارات التي بعدها .

**EXIT;**

تقوم بعملية الخروج من الكود نهائياً ولا تطبق أي أوامر بعدها ، وفي الأمثلة التوضيحية التالية سنجد أن break تخرج من العبارة فقط (Statement) بينما الـ exit; تقوم بالخروج من كامل الكود (code).

مثال :

```
<?  
$s=10;  
if ($s=10) {  
  echo "number=10";  
  exit;  
}  
elseif ($s<11) {  
  echo "number is less than 11"  
  {  
    echo "hello";  
  }  
?>
```

مثال :

```
<?  
$s=10;  
if ($s=10) {  
  echo "number=10";  
  break;  
}  
elseif ($s<11) {  
  echo "number is less than 11"  
  {  
    echo "Hello";  
  }  
?>
```

**Default;**

إذا لم تصلح جميع الحالات (Cases) في العبارة (Switch) فسيتم تنفيذ الأوامر التي تقع بعد هذه الكلمة وهي تؤدي نفس عمل else تقريباً في العبارة if .

مثال (1)

```
<?
$g= "ahmed";
Switch ($g) {
    Case "ahmed":
        Echo "مسموح";
        Break ;
    Case "khaled " :
        Echo "ممنوع";
        Break ;
    Case "salem" :
        Echo "ممنوع";
        Break ;
    Case "Mohmed " :
        Echo "مسموح";
        Break ;
Default ;
    Echo "لقد ادخلت اسم غير صالح";
}
?>
```

مثال (2)

```
Switch ($g) {
    Case $g>50:
        Echo "كبير";
        Break ;
    Case 40 :
        Echo "لاباس";
        Break ;
    Case ($g<15) :
        Echo "أطفال ممنوع";
        Break ;
    Case 30 :
        Echo "مسموح";
        Break ;
}
```

لاحظ أننا عند اختبارنا لنصوص نحتاج الى علامتي تنصيص مزدوجة وعند الارقام فاننا لانحتاج الي ذلك .

## تطبيق عملي

قم بفتح محرر النصوص لديك وكتب الكود التالي واحفظه باسم age.html

```
<html>
<form method=post action="age.php">
    كم عمرك ؟
<br>
<input type="text" name = "g">
<input type=submit value="ارسل">
</form>
</html>
```

قم بفتح محرر النصوص لديك وكتب الكود التالي واحفظه باسم age.php

```
<?
Switch ($g) {
```

```

    Case $g>50:
        Echo "كبير";
        Break ;
Case 40 :
        Echo "لاباس";
        Break ;
Case ($g<15) :
        Echo "أطفال ممنوع";
        Break ;
Case 30 :
        Echo "مسموح";
        Break ;
}
?>

```

## الشرح

تقوم العبارة **Switch** باختبار قيمة متغير ما ويمكنك إجراء أكثر من افتراض عليه ويجب عليك كتابة الكلمة **break** لكي تقوم بإيقاف تنفيذ العبارة **switch** فمثلاً لو قمت بكتابة الكود التالي :

```

<?
$g=40
Switch ($g) {
    Case $g<50:
        Echo "1";

    Case 40 :
        Echo "2";
}
?>

```

فإذا ادخل المستخدم الرقم 40 فسيتم طباعة الرقمين واحد واثنين كلاهما وذلك لأنك لم تقم بإيقاف العبارة فأكملت التحقق وطبقت جميع العمليات المطلوبة .

## التخلص من وسوم ال html

إذا قمت بوضع مربع نص وأردت من المستخدم كتابة شيء فيه فإنه يستطيع إدخال أي شيء ولن يفترض أنه كتب في مربع النص كالتالي :

I am <b>ahmed</b> ...

فسيقوم المتصفح بعرضها بعد معالجتها كالتالي :

I am ahmed ....

ولنقم بتطبيق عملي على ذلك  
قم بفتح محرر النصوص واكتب الكود التالي واحفظه باسم htmlch.html

```

<html dir="rtl">
<form method=post action="html.php">
أدخل اسمك الكريم
<br>
<input type="text" name = "fname">
<input type=submit value="رسل">
</form>
</html>

```

قم بفتح محرر النصوص واكتب الكود التالي واحفظه باسم html.php

```

<?
Echo "هذا هو الشكل الطبيعي للعبارة عند طباعتها";
Echo "<br>" . $fname;
?>

```

قم بوضع الملفات في مجلد السيرفر ثم قم بتشغيل الملف htmlch.html واكتب في مربع النص أي شيء وضعه بين وسوم html

مثال :

```
I am <b><i>alfareees</i></b></i>
```

ستجد أنه قد تم التعامل مع الوسوم كـ html وليس كنص عادي ولكي تعرضها كنص عادي فإنك تقوم باستخدام الدالة

```
htmlspecialchars();
```

حيث أنها ستقوم بمعاملة كود الـ html كنص عادي وطبيعي تماماً .  
إذاً نقوم بتعديل ملف الـ html.php ليصبح كالتالي :

```
<?
$name = htmlspecialchars($name);
Echo "هذا هو الشكل بعد استخدام الدالة "؛
Echo "<br>" . $name;
?>
```

## التكرارات والمصفوفات

لقد اخذنا في الدرس السابق شيئاً من أساسيات البرمجة وهو الدوال الشرطية وصناعة القرارات والآن نحن نتجه إلى شيء يجب جهاز الكمبيوتر عمله وهو التكرارات والمصفوفات .

في الواقع قد يكون لديك يومياً شيء تفعله بشكل مستمر مثل الإفطار في الصباح الباكر والنوم مساءً ، انك تستمر على هذا الروتين دائماً .... نحن نسمي هذا الشيء في لغة البرمجة التكرار .

هناك شيء آخر يسمى المصفوفات ... في الواقع قد يحتوي درج مكتبك الخاصة بالكتب على عدة أدرج الدرج الأول منها يحتوي على الكتب الإسلامية والدرج الثاني منها يحتوي على الكتب الرياضية والدرج الثالث يحتوي على كتب الرياضيات ... أو لنفرض أنك مدرس في إحدى المدارس ولديك جدول للحصص ففي الحصة الأولى لديك مثلاً تدريس مادة الرياضيات ... والحصة الثانية لديك تدريس مادة العلوم والثالثة لديك تدريس مادة الكيمياء ... إن حصصك مرتبة بشكل معين مع أنها كلها تسمى حصص إلا أن كل حصة تختلف عن الأخرى في المادة ! وهي مرتبة بشكل تصاعدي (الحصة الأولى ، الثانية ، الثالثة ....) .

نسمي هذه التقنية بالمصفوفات ... المصفوفات عبارة عن متغير اسمه ثابت ولها أكثر من قيمة وكل قيمة لها رقم معين ولكي تحصل على القيمة فانك تكتب المتغير ثم رقم القيمة التي فيه ، لا يشترط أن تكون هذه القيم متسلسلة فقد يكون هناك قيمتين ولكل قيمة رقم يختلف تماماً ويبعد كل البعد عن القيمة الثانية مثال رقم 1 و 258 كلاهما مختلف تماماً ويبعد كل البعد عن الآخر .

إن دمج ميزة التكرارات مع المصفوفات يساعدك على توفير عدد الأسطر للكود ويساعدك على صنع أشياء عجيبة في أقل عدد ممكن من الأسطر .

## التكرارات

التكرارات عبارة عن تكرار أمر معين بعدد معين من المرات ولقد اخذنا سابقاً الدوال الشرطية أو العبارات الشرطية بالأصح فوجدنا أن الكود الذي نكتبه في العبارات الشرطية لا تنفذ إلا عندما يكون الشرط صحيحاً أيضاً التكرارات فهي تختبر الشرط فإذا كانت قيمته صحيحة فإنها تقوم بعمل الكود المطلوب ثم تقوم بإعادة اختبار القيمة فإذا كان صحيحاً فإنها تقوم بإعادة تنفيذ الكود وهكذا ، أما عندما لا يكون الشرط صحيحاً فإنها تتوقف عن تنفيذ الكود ويتم اكمال البرنامج بشكل عادي ... هناك ثلاثة أنواع من التكرارات .  
إن أول دالة نقوم بأخذها في البداية هي الدالة **while**

### التكرار while

لقد قمنا بأخذ التكرار while لأنه بسيط جداً وصيغته هذا التكرار هي :

```
While (condition شرط )
{
code
}
```

مثال :

```
<?
$d = 10 ;
while ($d<15)
{
echo "$d <br>";
$d++;
}
```

<?>

سيقوم الـ PHP أولاً بإعطاء المتغير \$d القيمة 10 ثم يقوم بعد بدء التكرار while فإذا كان الشرط صحيحاً (وهو أن المتغير أصغر من الرقم 15) فإنه يقوم بتنفيذ الكود الذي بين الأقواس وعمل هذا الكود أن يقوم بطباعة المتغير ثم يقوم بإضافة واحد على القيمة الموجودة في المتغير \$d ثم بعد ذلك سيتم اختبار الشرط مرة ثانية فإذا كان صحيحاً فسيتم نفس العملية حتي يكون الشرط غير صحيح فيتوقف عندها التكرار ويتم إكمال الكود التي تقع بعد الأقواس .

إذا لم تقم بوضع حد للتكرار فلن يتوقف التكرار وقد يكون لانهائي ....  
مثال :

```
<?
$d = 10 ;
while ($d < 15)
{
echo "$d <br>";
}
?>
```

سيتم طباعة الرقم 10 ولن يتوقف التكرار لأن الشرط صحيح دائماً وليس هناك ما يوقفه بينما في الكود السابق استطعنا إيقاف الكود بسبب أننا كنا نضيف واحد على القيمة الموجودة في المتغير وكلما يتم إعادة اختبار الكود كل ما تتغير القيمة حتي يصبح الشرط غير صحيح بسبب أن \$d أكبر من 15 .

### التكرار do - while

هذا التكرار يعمل بنفس طريقه التكرار الأول إلا أنه يوجد بعض الاختلافات البسيطة وصيغته كالتالي :

```
do
code
while (شرط condition);
```

مثال :

```
<?
$f = 15 ;
do
{
echo "$f";
$f ++
}
while () ;
```

سيقوم التكرار بتنفيذ السطر الموجود بين القوسين أولاً ثم يقوم بتنفيذ باختبار الشرط فإذا كان الشرط صحيحاً قام بإعادة العملية الموجودة بين القوسين وهي إضافة واحد على المتغير \$f وهكذا حتي يكون الشرط غير صحيح فيتم التوقف .. لاحظ أننا في التكرار الأول قمنا باختبار الشرط قبل صناعة أي عمل بينما في التكرار الثاني قمنا بتنفيذ الكود أولاً ثم قمنا بإجراء الاختبار .

### التكرار FOR

يختلف هذا التكرار عن سابقه لكن وظيفته هي نفس وظيفتهما وهي تكرار الأوامر عند حصول شي معين

الصيغة :

```
For (اداء عمله على العداد set counter ; اختبار القيمة test value ; عداد counter)
{
شغرة code
}
```

مثال :

```
<?
For ($u = 18 ; $u > 10 ; $u--)
{
echo $u;
}
?>
```

يتكون هذا التكرار من ثلاثة أقسام .... القسم الأول نضع فيه متغير يحتوي على قيمة حيث سيبدأ التكرار العمل من عند هذه القيمة والقسم الثاني نكتب فيه الشرط الذي سيقوم التكرار بفحصه (والذي هو كالمعتاد اختبار لقيمة المتغير في القسم الأول) والقسم الثالث نضع فيه العمل الذي سيجري على المتغير عند كل تكرار ثم نقوم بكتابة كود التي سيقوم بتنفيذها التكرار بين القوسين .

كأننا نقول للـphp بشكل عامي أن يقوم في البداية بإعطاء المتغير \$u القيمة 18 وقبل ان يقوم بتنفيذ الكود عليه أن يقوم بتحليل الشرط فإذا كان الشرط صحيحاً فإنه يقوم بإنقاص واحد من المتغير \$u ويتم تنفيذ الكود حتي يصبح المتغير \$u قيمته 9 فيقوم الـPHP آنذاك بالخروج من التكرار والذهاب الي الكود الذي يلي القوسين .

## المصفوفات

لقد قمنا بتعريف المصفوفات سابقاً بشكل بسيط وحاد الوقت الآن لنعرفها ونعرف كيفية عملها . المصفوفات عبارة عن متغير وهذا المتغير يحتوي على أكثر من قيمة أو عنصر (element) وكل عنصر له فهرسة (Index) تبدأ هذه الفهرسة من الصفر إذا لم تقم بتحديد لها

مثال :

```
<?
$A[ ] = "alfareees";
$A[ ] = 13;
?>
```

في هذا المثال سيقوم الـPHP بإعطاء الفهرسة تلقائياً فسيقوم بوضع الرقم فتصبح المتغير فهرسته كالتالي :

```
$A[0] = "alfareees";
$A[1] = 13;
?>
```

إننا لم نقم بإدخال هذه الأرقام من تلقاء أنفسنا ولكن الـPHP قام بوضعها مع أنه يمكننا أن ندخلها بشكل عادي فمثلاً لو كتبنا :

```
<?
$A[0] = "alfareees";
$A[1] = 13;
?>
```

سيقوم الـPHP بأخذ الفهرسة المعتمدة ولن يضع أي فهرسة أخرى يمكننا أيضا أن نكتب أي فهرسة ولا نعتمد على الترتيب في الأرقام .

مثال :

```
<?
$A[10 ] = "alfareees";
$A[ 25 ] = 13;
?>
```

هل لاحظت أيضاً أننا لم نقم بتعريف نوع متغيرات المصفوفة وقام الـPHP بتعريفها تلقائياً بدلاً منا فمرة استخدمنا قيمة حرفية ومرة استخدمنا رقماً ورغم ذلك فلم يقم الـPHP بعمل أي اعتراض إضافة إلى ذلك فإن الـPHP يقوم بتحديد عدد عناصر المصفوفة تلقائياً فهو يعرف مثلاً من المثال السابق أن عدد عناصر المصفوفة الكلي هو عنصرين .  
يمنحنا الـPHP ميزة أخرى وهي عدم التقييد بالأرقام في الفهرسة فمثلاً يمكننا استخدام حروف عادية .

مثال :

```
<?
$A["a" ] = "alfareees";
$A["b" ] = 13;
?>
```

لاحظ أننا استخدمنا القيم الحرفية ولم يعترض الـPHP بتاتاً ويمكننا طباعة أي عنصر من عناصر المصفوفة بكل بساطة .

مثال :

```
<?
$r ["aa"] = "ahmed ali";
$r [1] = 13273;
$r [20] = 13273;
echo $r[aa];
echo $r[20];
echo $r["aa"];
?>
```

لا فرق بين أن نكتب النص الحرفي (aa) بين علامتي تنصيص عند الطباعة وعند كتابته بدون علامات تنصيص ... سيقوم الـPHP بمعرفة ذلك تلقائياً .

يمكننا تعريف المصفوفات أيضاً بطريقة أخرى

```
$variable = array (elements) ;
```

مثال :



```
<?
$t = array ("ahmed", "ali", "salem", "alfarsi");
echo $t [0];
?>
```

يقوم الـ PHP بإعطاء كل عنصر من عناصر المصفوفة رقم فهرسة فتصبح كالتالي :

العنصر Element	الفهرسة Index
Ahmed	0
Ali	1
Salem	2
alfarsi	3

إذن القيمة التي سيطبعها الـ PHP في النهاية هي ahmed ، لاحظ أن الـ PHP قام بإعطاء رقم الفهرسة وقام بالبدء من الصفر ولكن يمكننا جعل الـ PHP يبدأ الفهرسة من الرقم واحد كالتالي :

```
<?
$r = array (1=>"ahmed", "ali", "salem", "alfarsi");
?>
```

عند تعريفك لرقم الفهرسة للقيمة الأولى سيقوم الـ PHP بإعطاء أرقام فهرسة بشكل تسلسلي ، عندئذ ستصبح الفهرسة كالتالي :

العنصر Element	الفهرسة Index
ahmed	1
Ali	2
salem	3
alfarsi	4

هناك طريقة لتكون أيضا الفهرسة هي عبارة عن حروف :

```
<?
$r = array ("ss"=>"ahmed", "sf"=>"ali", "da"=>"salem", "bv"=>"alfarsi");
?>
```

عندئذ ستصبح الفهرسة كالتالي :

العنصر Element	الفهرسة Index
Ahmed	Ss
Ali	Sf
Salem	Da
Alfarsi	Bv

عندما نريد تغيير أي عنصر في المصفوفة فيمكننا عمل ذلك ببساطة .

مثال :

```
$r [ss] = "لمياء";
```

لاحظ أننا قمنا بتغيير القيمة من (ahmed) الى (لمياء) ... طريقة بسيطة أليس كذلك ( :

### قراءة المصفوفات واستخراج القيم

تكلّمنا سابقا عن التكرار For يمكننا استخراج عناصر مصفوفة وطباعتها في بساطة وتوفير وقت عن طريق التكرارات

لنفرض أن لديك هذه المصفوفة :

```
<?
$people = array ("ahmed", "ali", "salem", "alfarsi");
?>
```

واردت أن تطبع أسماء جميع الأشخاص المتواجدين فيها أولاً نحن نعرف أن المصفوفة إذا لم نقم بتعريف رقم فهرسة لها فإن الـ PHP يقوم ببداية فهرستها من الصفر وعلى ذلك فإن رقم العنصر الأول 0 ورقم العنصر الرابع 3 ... على ذلك يمكننا بكل بساطة كتابة الكود التالي الذي يقوم بطباعة المصفوفة كالتالي :

```
<?
```

```

$people = array ("ahmed", "ali", "saalem", "alfarsi");
echo "$people[0]. <br>";
echo "$people[1]. <br>";
echo "$people[2]. <br>";
echo "$people[3]. <br>";
?>

```

لنفرض أن لديك ثلاثين أو ثلاثة آلاف اسم في مصفوفة أُن تبدو هذه الطريقة متعبة قليلا !!!  
هناك طريقة أخرى وهي عن طريق التكرارات .

لنفرض أننا أردنا كتابة تكرار يقوم بطباعة الأرقام من واحد الى عشرة فإننا نستطيع كتابة التكرار بالشكل التالي :

```

<?
For ($i=1;$i<11;$i++)
{
Echo "$i <br>";
}
?>

```

والآن لنقل أننا نريد طباعة الأربعة عناصر في المصفوفة كل ما علينا هو إجراء عملية بسيطة على الكود لكي يتم ذلك :

```

<?
$people = array ("ahmed", "ali", "saalem", "alfarsi");

For ($i=0;$i<4;$i++)
{
Echo "$people[$i] <br>";
}
?>

```

لاحظ أننا بدأنا العداد بالقيمة صفر ثم اشتراطنا أن يكون أقل من 4 لأن آخر عنصر في المصفوفة رقم فهرسته 3 ثم قمنا بجعله يزداد بقيمة 1 لأننا نريد طباعة جميع عناصر المصفوفة وقمنا بوضع رقم العداد في خانة الفهرسة وعلى ذلك سيتم في كل تكرار طباع عنصر المصفوفة الذي فهرسته تساوي رقم العداد .

لقد تكلمنا سابقاً في درس النماذج عن إخراج القيم من قائمة على شكل مصفوفة .

مثال :

```

<form action = "array.php" method = post>
ما هو مشروبك المفضل ؟
<br>
<select name = "a[]" multiple>
<option>شاي</option>
<option>قهوة</option>
<option>كايتشينو</option>
<option>توت</option>
<option>برتقال</option>
</select>
<br>
<input type=submit value = "لذيذ" >
</form>

```

في ملف الـ array.php اكتب :

```

<html>
لقد قمت باختيار التالي :
<?
For ($i=0;$i<4;$i++)
{
Echo "$a[$i] <br>";
}
?>
</html>

```

لقد عرضنا في القائمة خمسة عناصر ... لاحظ أننا وضعنا في اسم المتغير للقائمة فوسين [ ] لكي يتعرف الـhtml علي أنه سيتم تخزين البيانات تلقائياً بعد ذلك قام الـPHP بفهرسة العناصر التي تم إرسالها من قبل العميل سواء كانت ثلاثة أو أربعة ولكنها بالطبع لن تزيد على خمسة .... على ذلك سيكون آخر رقم تنتهي به المصفوفة هو 4 .

أتوقع أنك الآن بدأت تحب المصفوفات .... يمكننا صناعة القائمة عن طريق المصفوفة أيضاً ....

مثال :

```
<form action = "list.php" method = post>
ما هو مشروبك المفضل ؟
<br>
<select name = "s" >
<?
$shrab =array("شاي","قهوة","كابتشينو","توت","برتقال");
For ($k=0;$k<4;$k++)
{
echo "<option>". $shrab[$k]. "</option>";
}
?>
</select>
</form>
```

عند اختيار المستخدم للقيمة سيتم وضعها في المتغير \$s يمكنك مراجعة درس النماذج لكي تفعل ذلك ، هذا المثال يقوم بصناعة مصفوفة للمشروبات ثم يقوم بإخراجها في قائمة مما يوفر علينا الوقت في كتابة الكود فلو كان لديك مثلاً حوالي مئة دولة فيمكنك مثلاً وضعها في مصفوفة وبعد ذلك بناء القائمة التي سوف تقوم ببناء القائمة التي ستحتوي على هذه الدول عن طريق المصفوفات والتكرارات .

قم بحفظ التغييرات في ملف إمتداده php وقم بكتابة الملف list.php اعتماداً على معلوماتك السابقة في درس النماذج .

## دوال المصفوفات

### الدالة key

لنفرض أن لدينا مصفوفة مكونة من عنصرين :

مثال :

```
$s= array ("محمد","على");
```

الآن لنصف إليها هذه السطور

```
<?
$s= array ("محمد","على");
$t=key ($s);
echo $t;
?>
```

يقوم الأمر key بإيجاد رقم الفهرسة (index) للعنصر النشط حالياً .... وهو الرقم صفر حيث أننا لم نضع فهرسة وهذه هي الفهرسة التي وضعها الـPHP تلقائياً عندما لم نضع فهرسة ... قد تحريك كلمة النشط لكن ستعرف أننا نستطيع التجول بين عناصر المصفوفة لاحقاً .

قد يكون رقم الفهرسة حروف أو كلمات

مثال :

```
<?
$s= array ("م"=>"م","ع"=>"ع");
$t=key ($s);
echo $t;
?>
```

### الدالة current()

تقوم الدالة current بإيجاد القيمة لعنصر المصفوفة الحالي (index value) .

مثال :

```
<?
$s= array ("ع"=>"على","م"=>"محمد");
$p=current ($s);
echo $p;
?>
```

في المثال السابق قمنا بإيجاد القيمة الحالية للعنصر النشط .... لاحظ أننا أوجدنا بالأمر key رقم الفهرسة بينما أوجدنا بالأمر current القيمة للعنصر المفهرس .

### كيف يمكننا تنشيط العناصر الاخرى للمصفوفة ؟!

يمكننا ذلك عن طريق الدالتين next() و prev اللتان تقومان بالتجول بين عناصر المصفوفة ..... لنفرض أن لدينا مصفوفة تتكون من ثلاثة عناصر  
مثال :

```
<?
$s= array ("ع"=>"على","م"=>"محمد","ا"=>"احمد");
echo key($s)."<br>";
echo current($s) . "<br>";
?>
```

لقد قمنا في هذا المثال بطباعة قيمة رقم الفهرسة للعنصر الحالي وقيمته ( اقصد برقم الفهرسة الحرف (ع) واقصد بالقيمة (على) .... لنقم الآن بالتجول بين عناصر المصفوفة ولنر نتيجة الطباعة .

مثال :

```
<?
$s= array ("ع"=>"على","م"=>"محمد","ا"=>"احمد");
next($s);
echo key($s)."<br>";
echo current($s) . "<br>";
?>
```

```
<?
$s= array ("ع"=>"على","م"=>"محمد","ا"=>"احمد");
next($s);
next($s);
echo key($s)."<br>";
echo current($s) . "<br>";
?>
```

لاحظ أننا كتبنا الدالة next() قبل أن نقوم بالانتقال لكي يتم تنشيط العنصر الثاني في أول مثال ولتنشيط العنصر الثالث في ثالث مثال (ولاحظ أننا كتبنا next() مرتين ) .

يمكننا الرجوع لتنشيط العنصر السابق بوضع الدالة prev() فمثلاً يمكننا تعديل المثال التالي :

```
<?
$s= array ("ع"=>"على","م"=>"محمد","ا"=>"احمد");
next($s);
next($s);
prev($s);
echo key($s)."<br>";
echo current($s) . "<br>";
?>
```

فسيقوم الـ PHP في هذه الحالة بطباعة العنصر الثاني وليس الثالث لأنه تم التراجع خطوه عن طريق prev()

ماذا سيحصل إذا قمنا بإضافة عنصر على مصفوفة غير محدودة الفهرسة ؟!  
لنفرض أن لدينا مصفوفة وأضفنا إليها عنصر غير محدد الفهرسة . مثل :

```
<?
$s= array (12=>"على",5=>"محمد",44=>"احمد");
$s[ ]="هشام";
Next($s);
Next($s);
Next($s);
Echo key ($s)."<br>";
Echo current( $s )."<br>";
?>
```

سيقوم PHP ببساطة بالبحث عن أكبر رقم فهرسة وبعد ذلك يبدأ بإعطاء الفهرسة تسلسلاً بعده فإذا كانت أرقام الفهرسة حروفاً بدأ من الصفر في إعطاء الرقم .. ولاحظ في هذا المثال بأنه قام بإعطاء العنصر الرقم 45 لأن أكبر عنصر في المصفوفة هو 44 وعلى ذلك قام بإعطاء الأرقام تسلسلاً بعد هذا الرقم .

## الدالة List و Each

لنفرض أنك قد قمت بصنع مصفوفة غير مفهرسة بالترتيب  
مثال :

```
<?
$s= array (12=>"على",5=>"محمد",44=>"احمد");
?>
```

على ذلك دعنا نخبرك بخبر سار وهو أنك تستطيع أن تجعل حياتك مع PHP أسهل مع حياتك مع نفسك !

While (list (Index,Element value) =each (array))

تستطيع بواسطة هذه الدالتين وعن طريق التكرار while استخراج جميع العناصر الموجودة في المصفوفة

```
While (list($e,$r) = each ($s))
```

```
{
echo "<br> $e<br> $r";
}
```

أولاً أنت تقوم بتسمية متغيرين واحد منهما لرقم الفهرسة (\$e) والثاني للعنصر (\$r) ويمكننا تسميتهما بأي اسم وفي حالة ما إذا أردنا عرض العنصر فقط أو معرفة العنصر فقط فيمكننا حذف (\$e) ولكننا لانحذف الفاصلة

```
While (list(,$r) = each ($s))
```

```
{
echo "<br> $e<br> $r";
}
```

لنعد الى المثال الذي فيه رقم الفهرسة والعنصر ... سيقوم التكرار بوضع رقم الفهرسة (الذي قد يكون نصياً) في المتغير \$e وسيضع قيمة العنصر الذي رقم الفهرسة له هو \$e في المتغير \$r ثم سيقوم بطباعة العناصر حتي ينتهي منها جميعها ...

**ملاحظة مهمة :** إذا لم تقم بتعريف فهرسة للمصفوفة (حروف أو أرقام أيا كان) فسيتم استخدام العناصر عندما يطلب التكرار الفهارس .

مثال :

```
<?
$e=array("fsda","terhfgfd","tewr");
While (list ($I,$V)=each($e))
{
echo "<br>$e[$I]";
}
?>
```

لاحظ مع H أننا طلبنا طباعة الفهرسة (index) إلا أنه تم أخذ العناصر (elements) بدلاً من الفهرسة

يمكننا بواسطة هذه الدالة صناعة أشياء مفيدة وكمثال لذلك لنفرض أن لدينا مصفوفة أرقام هواتف ونريد أن نخرج هذه المصفوفة على جدول html فنستطيع صناعة هذا الجدول عن طريق التكرار السابق بكل سهولة .

مثال :

```
<table align='center' dir = "rtl" border="1" width="100%" cellspacing="0" bordercolorlight="#000000"
bordercolordark="#000000" bordercolor="#000000">
```

```
<tr>
<td align='center'>الاسم</td>
<td align='center'>رقم التلفون</td>
</tr>
```

```
<?
$s = array (658=>"465873 , "سالم", <=456546, "عادل");
While (list($e,$r) = each ($s))
```

```
{
echo "<tr><td align='center'>". $r . "</td><td align='center'>". $e . "</td></tr>";
}
?>
```

```
</table>
```

أرايت كيف استخراجنا جميع أرقام التلفونات في جدول بواسطة تكرار بسيط ، يمكنك صناعة الأكثر واختصار الكثير من الوقت على ذلك إذا كانت المصفوفة تحتوي على المئات من الأرقام بواسطة هذا الكود بدلاً من أن تكتب الكود على شكل html وتكتب البيانات وتتعب نفسك .

يمكنك أيضاً معرفة عدد العناصر في مصفوفة معينة إذا كنت تريد معرفة عددها وذلك بالطريقة التالية :

```
<?
$s= array (12=>"على",5=>"محمد",44=>"احمد");
```

```

$S=0;
While (list($E,$r) = each ($s))
{
$S++;
}
ECHO " عدد عناصر المصفوفة " . $S++;
?>

```



هناك العديد من الدوال التي يوفرها لنا PHP لفرز المصفوفات . نحن سنأخذ نظرة عن الخمسة دوال الأكثر استخداماً :

### الدالة Sort()

هذه الدالة من أساسيات فرز المصفوفات وهي جداً أساسية وهي تقوم بأخذ محتويات المصفوفة ومن ثم تقوم بفرزها هجائياً اعتماداً على الأحرف الكبيرة أولاً ثم الصغيرة .. تتطلب هذه الدالة اسم المصفوفة التي سيتم عليها الفرز

```
Sort (ArrayName);
```

إذا قمنا بإنشاء مصفوفة بالشكل التالي :

```
$NaNo=array ("ali","saalem","hythem","Khaled","Ammar","Hesham");
```

فإذا أردنا فرزها عن طريق الدالة sort() فإننا نقوم باستخدامها كالتالي :

```

<?
$NaNo=array ("ali","saalem","hythem","Khaled","Ammar","Hesham");
sort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>

```

لاحظ أنه عند تنفيذك للمثال ستجد أن الـ PHP قام بالفرز اعتماداً على الأحرف الكبيرة أولاً ثم قام بالفرز بعدها اعتماداً على الأحرف الصغيرة .

### الدالة Asort()

هذه الدالة تعمل نفس عملية الدالة sort() ولكن هناك اختلاف بسيط فمثلاً لو كتبنا المصفوفة كالتالي :

```
$NaNo=array ( "ad"=>"ahmed", "kh"=> "khaled");
```

وأردنا فرزها وطباعة الفهارس والقيم كما في المثال التالي :

```

<?
$NaNo=array ( "ad"=>"ahmed", "kh"=> "khaled");
sort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>

```

قارن ناتج المثال السابق مع هذا المثال :

```

<?
$NaNo=array ( "ad"=>"ahmed", "kh"=> "khaled");
asort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}

```

```
?>
```

اعتقد انك قد عرفت الفرق ففي المثال الاول قامت الدالة sort باستبدال الحروف بأرقام في الفهرسة أما في المثال الثاني فقد تم وضع الحروف كما هي وتم فرزها كما تفعل الدالة sort في الفرز .  
باختصار لا يوجد فرق بين sort و asort إلا في أن الدالة sort تستبدل فهرسة الحروف بأرقام .

### الدالة Rsort() و arsort

تقوم بنفس عمل sort و asort ولكن بشكل عكسي جرب الأمثلة التالية :  
مثال :

```
<?
$NaNo=array ( "ad"=>"ahmed", "kh"=> "khaled");
rsort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
```

مثال :

```
<?
$NaNo=array ( "ad"=>"ahmed", "kh"=> "khaled");
arsort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
```

ستجد أن الدالة rsort تقوم بنفس عملية الدالة sort ولكن بشكل عكسي  
أيضاً الدالة arsort تقوم بنفس عملية asort ولكن بشكل عكسي .  
يمكنك استعمال كل هذه الدوال في الفرز مع الحروف العربية ( إذا كان السيرفر يدعم اللغة العربية )  
قم بتطبيق المثال التالي :

```
RSORT()
<?
$NaNo=array ( "ad"=>"سالم", "kh"=> "أحمد");
rsort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
<br>-----<br>
ARSORT()
<?
$NaNo=array ( "ad"=>"أحمد", "kh"=> "أحمد");
arsort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
<br>-----<br>
ASORT()
<?
$NaNo=array ( "ad"=>"هاشم", "kh"=> "جمال");
asort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
<br>-----<br>
SORT()
<?>
```

```

$NaNo=array ( "ad"=>"هاشم", "kh"=> "جمال");
sort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>

```

### الدالة ksort

تكلّمنا سابقاً عن طريقة فرز المصفوفات ولكن نريد أن نلفت نظرك أننا كنا نعتمد على العنصر في الفرز (element) ولكن هذه الدالة تقوم بالاعتماد على رقم الفهرسة في الفرز (index) مثال :

ort

```

<br>-----<br>
asort()
<?
$NaNo=array ( "ad"=>"هاشم", "kh"=> "جمال");
asort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>
<br>-----<br>
ksort()
<?
$NaNo=array ( "ad"=>"هاشم", "kh"=> "جمال");
ksort($NaNo);
While (list($e,$r) = each ($NaNo))
{
echo "<br> $e<br> $r";
}
?>

```

لقد اعتمد الphp على index ولم يعتمد على الelement في الفرز .

## دوال المصفوفات الإضافية

هناك الكثير من الدوال التي يمنحنا إياها الPHP للتعامل مع المصفوفات والتي لا يكفي الوقت لذكرها الآن سنقوم بشرح أهم الدالتين والمستخدمه بكثرة وهي `array_push()` و `array_pop()`

لنفرض أننا قمنا بإنشاء مصفوفة بالشكل التالي :

```

<?
$saher[ 5]="salem";
$saher[ 85]="khaled";
$saher[ 35]="mohmed";
$saher[ 19]="hajeer";
?>

```

وأردنا أن نضيف عنصر جديد لها فقمنا بالتالي :

```

<?
$saher[ 5]="salem";
$saher[ 85]="khaled";
$saher[ 35]="mohmed";
$saher[ 19]="hajeer";
$saher[ ]="Alfarees";
?>

```

انظر إلى العنصر الأخير الذي سيعطيه الPHP رقم الفهرسة (index) وسيكون رقم فهرسته هو 86 . نريد أن نلفت نظرك أننا نستطيع عمل إضافة لعنصر على المصفوفة بطريقة أخرى وهي عن طريق الدالة `array_push()` كالتالي :

```
array_push (ArrayName,اسم المصفوفه,Elemnt1, Elemnt2, Elemnt3,.....)
```

نضع في القسم الأول من الدالة اسم المصفوفة التي نريد إضافة العنصر لها ونضع في القسم الثاني عنصر واحد أو أكثر وهي التي سيتم إضافتها للمصفوفة .

مثال :

```
<?>
```



```
$saher[ 5]="salem";
$saher[ 85]="khaled";
$saher[ 35]="mohmed";
$saher[ 19]="hajeer";
array_push ($saher,Alfarees)
?>
```

مثال :

```
<?
$saher[ 5]="salem";
$saher[ 85]="khaled";
$saher[ 35]="mohmed";
$saher[ 19]="hajeer";
array_push ($saher,Alfarees,salem,sameer,thamer)
?>
```

ولو أردنا حذف مثلاً عنصر من المصفوفة فإننا نقوم بتعريف المصفوفة من جديد أو يمكننا استخدام الدالة `array_pop` التي تقوم بحذف آخر عنصر من المصفوفة والتي تتطلب فقط اسم المصفوفة

**Array\_pop(اسم المصفوفة ArrayName)**

مثال :

```
<?
$saher[ 5]="salem";
$saher[ 85]="khaled";
$saher[ 35]="mohmed";
$saher[ 19]="hajeer";
array_pop($saher)
?>
```

سيتم حذف العنصر `hajeer` من المصفوفة ولن يكون في المصفوفة غير ثلاث عناصر .

### Explode و Implode

تقوم هذه الدالتين بافتصاص قيمة معينة من مصفوفة أو نصوص وتقوم بإضافة قيمة معينة على مصفوفة أو نصوص .

### الدالة Implode

تقوم بإضافة قيمة على بين عناصر المصفوفة .

مثال :

```
<?
$stng =array ("ahmed", "salem", "ali", "alfarsi");
$r =implode ("H",$stng);
echo $r;
?>
```

### الدالة explode

تقوم بحذف قيمة من مصفوفة وذلك لايغني حذف عناصر من المصفوفة .

مثال :

```
<?
$stng =array ("ahmed", "salem", "ali", "alfarsi");
$r =implode ("-",$stng);
echo $r;
$r= explode ("-",$stng);
echo $r;
?>
```

### HTTP\_POST\_VARS و HTTP\_GET\_VARS

هذه ليست متغيرات بل مصفوفات ، نعم هذه مصفوفات ولكن في ماذا نستخدمها ولماذا ؟

في الواقع تحدثنا في الدرس السابق عن طريقة التعامل مع النماذج والحصول على البيانات من المستخدم وتكلمنا عن أسلوبيين لنقل البيانات وهما GET و POST  
عندما تصل البيانات محفوظة في متغيرات إلى صفحة الـ PHP فإنه يقوم بتعريفها تلقائياً ويمكنك طباعة المتغيرات وقيمتها مباشرة من غير تعريف .... ولكن هذه الميزة في الـ PHP يمكن إلغاؤها عن طريق الملف الـ PHP.INI وذلك بإغلاق ميزة register\_globals وذلك بوضع off بدلا من on

الوضع الافتراضي لها هو on ولكن تستطيع إغلاقها وقد تكون مستاجراً عند مزود خدمة ويب وسيط فيقوم بإغلاق هذه الميزة من باب الحماية ليس إلا .... لاتقلق يمكنك الحصول على البيانات فهي ما زالت موجودة ولكن يجب عليك أن تقوم باستخدام هذه المصفوفتين لكي تستخرج البيانات .

لنفرض أنك اشتركت عند مزود ويب وكان قد أغلق ميزة (register\_globals) حسناً لنفرض أنك قد صنعت نموذجاً يستخدم مربع نص ويحفظ قيمته في متغير اسمه Dorrah ثم بعد ذلك يقوم بإرسال هذه القيمة باستخدام الأسلوب GET إذا سيكون جزء من الكود في الصفحة الأولى والتي تحتوي على النموذج كالتالي

```
<form method =get action = "try.php">
ماهو اسم الطفل الذي استيقظ به العالم الاسلامي من غفلة قبل عده شهور !!
<br>
<input type=text name = "Dorrah">
<br>
```

في الملف الثاني(try.php) سنقوم بكتابة الجزء الذي سيقوم بطباعة القيمة كالتالي

```
<?
Echo HTTP_GET_VARS["Dorrah"];
?>
```

لاحظ أننا لم نستخدم \$ ولكن إذا أردنا الإحتفاظ بقيمة المتغير في متغير آخر فيمكننا ذلك بشكل عادي كالتالي :

```
<?
$Dorrah= HTTP_GET_VARS["Dorrah"];
?>
```

طريقه بسيطة .... أليس كذلك ولكن .... لنفترض أن مزود خدمة الويب لديك حريص جداً ولذلك فقد ألغى أيضاً ميزة استقبال هذه القيم في المصفوفات .... يمكنه ذلك في ملف الـ php.ini في اعدادات الـ track\_vars الذي يقوم بمنع السيرفر من استخدام هذه المصفوفات (هذه الميزة يمكن إلغاؤها في php4 ) .... على ذلك انصحك بإرسال رسال تذر وشكوي إلى مزود الخدمة لديك .. تعلن فيها أن الأمر أصبح لايتحمل .

### مصفوفه متعدده الابعاد

يمكنك صناعة مصفوفات بداخل مصفوفات على حسب ماتحتاجه في معلوماتك الرياضية فقد تحتاج مثلاً إلى إنشاء أشياء معقدة (ومقلقة نفسياً) نريد أن نخبرك على أية حال أنه يمكنك صناعة المصفوفات المتعددة الأبعاد ويمكنك استخدام حتي مائة مصفوفة متداخلة ولكن يجب أن تراعي حجم الذاكرة المستخدمة في السيرفر لديك (وعلى كل حال إن استطعت أن تقوم بالتركيز في صناعة عشر مصفوفات متداخلة بدون أي مشاكل أو مرض نفسي أو .... فأنت تستحق جائزة) .

يمكننا كتابة مصفوفة متداخلة كالتالي :

```
<?
$mon= array (1=>array ("sharkeh al-jafali",154786) ,2 => array ("saalem almazen",1257) );
while (list($personnum) =each ($mon))
{
echo ("<br>$personnum<br>");

while (list(,$phone)=each ($mon[$personnum]))
{
echo ("$phone");
}
}
?>
```

### الشرح

هذا المثال قد يكون غامضاً جداً لكن فكرته بسيطة أولاً افترض أنك تعلم عن list.each جيداً وتعرف صيغة التكرار الذي يستخدمها .

الآن لدينا مصفوفة تتكون من رقمين للفهرسة هذين الرقمين كل واحد منهما عنصره عبارة عن مصفوفة هذه المصفوفة تحتوي على عنصرين ( ولنتناسي أنهما يحتويان على أرقام فهرسة ) وهما اسم شخص ورقم هاتفه .

```
while (list($personnum) =each ($mon))
{
echo ("<br>$personnum");
```

قمنا بإخراج رقم الفهرسة الأساسي للمصفوفة والذي يعتبر هو الرقم التسلسلي للأشخاص أصحاب الهواتف ومن بعد ذلك يقوم بطباعة هذا الرقم التسلسلي ويبدأ من سطر جديد .

### في الخطوة الثانية :

```
while (list(,$phone)=each ($mon[$personnum]))
{
echo ("$phone");
}
```

نقوم بإخبار الـ PHP بطباعة العناصر الذي تحتويها المصفوفة التي تم طباعة رقم فهرستها ، ولاحظ (\$phone) أنها تشير إلى عناصر مصفوفة وليس فهرستها لأننا تجاهلنا فهرس المصفوفة الداخلية .  
لا تقلق الأمر سهل ولكنه يحتاج الى تدريب فقط ، وعليك أن تتدرب وصدقني أنني حاولت ان أبسط المثال من أجلك ... أتمني أن تكون قد فهمت .

### تطبيق عملي

افتح محرر النصوص لديك واكتب الكود التالي :

```
<?
Echo "<form method =post action = 'exam2.php' " ;
$boy=array ("أحمد" , "خالد" , "سعد" ,"حسن");
while (list(,$Name) = each ($boy))
{
echo " ما هي السنة الدراسية لـ $Name ";
Echo "<select name = 'school[]'>
<option>اول ثانوي</option>
<option>ثاني ثانوي</option>
<option>ثالث ثانوي</option>
</select>";
echo "<br><br>";
echo "<input type =hidden name =boy[] value ='$Name'>";
}
echo "<input type =submit ></form>";
?>
```

احفظ الكود باسم exam.php

افتح محرر النصوص واكتب الكود التالي واحفظه في ملف باسم exam2.php

```
<html dir = "rtl">
<?
While (list($I,$V)=each($school))
{
$friendschool[] = $school[$I].$boy[$I];
}
asort ($friendschool);
While (list ($I,$V)=each($friendschool))
{
echo "<br>$boy[$I]." ".$school[$I];
}
?>
```

قم بتشغيله بعد نقله لمجلد السيرفر

### الشرح

الذي قمنا به في المثال السابق هو أننا قمنا بإنشاء مصفوفة لعدة أشخاص (\$boy) ونريد أن نعرف مرحلتهم الدراسية في الثانوية فأنشأنا لكل طالب قائمة منسدلة بواسطة التكرار (list-each) بصناعة قوائم منسدلة وحقول مخفية يتم تخزين قيم الحقول (التي تحتوي على أسماء الأشخاص) في المصفوفة (\$boy) وسيتم تخزين نتائج كل القوائم في مصفوفة (\$school)

وبعد أن يختار المستخدم الإجابات التي تناسبه وإرسال البيانات سيتم استقبال المصفوفة التي فيها نتائج القوائم المنسدلة (\$school) واستقبال المصفوفة التي فيها أسماء الأشخاص (\$boy) ومن ثم يتم إنشاء مصفوفة جديدة باسم \$friendschool[] ويؤخذ منها معلومات المصفوفتين ويتم دمجها فيها ومن ثم يتم تكرار آخر طباعة عناصر المصفوفتين \$school و \$boy .

## تكرار foreach

هذا التكرار هو من الأشياء الجديدة في الـ php4 وهو يساعدك على معرفة عناصر مصفوفة معينة أو طباعة محتوياتها .

```
Foreach ($ArrayName As $ArrayItem)
{
    شفره code
}
```

مثال :

```
<?
$T= array (a=>"ahmed " , b => "basem", c=>"car")

Foreach ($T As $A => $r)
{
    echo $A ."-----". $r;
}
?>
```

## الدالة count

تقوم بحساب عدد العناصر الموجودة في المصفوفة

مثال :

```
<?
$c=array("a","b","c");
$v=count($c);
echo $v;
?>
```

## ترتيب الكود البرمجي

تعلمنا في الدروس السابقة أساسيات من أساسيات البرمجة وإعطينا مثال عن الروتين في الحياة اليومية وهو أن تقوم بعمل شيء أكثر من مرة في الحياة اليومية مثل شرب الشاي أو شرب القهوة وغير ذلك ، درسنا اليوم يتكلم عن ترتيب الكود ويتكلم تقريباً عن نفس فكره الروتين اليومي فأنت في حياتك تكرر بعض الأعمال بشكل روتيني وقد تكون مللت الروتين فأحضرت شيء يساعدك علي التخفيف من هذا الروتين ... فمثلاً عند استخدامك لبرنامج MS Word قد تكون مللت من تنسيق عدة نصوص بطريقة معينة فأنت عند ذلك تقوم بصناعة ماكرو يقوم بفعل العمل الذي كنت تفعله في عدة خطوات بخطوة واحد فقط !!

ولنقل أنك في حياتك اليومية وفي يوم إجازة وقررت أن تقوم بعمل تنظيف شامل (يا إلهي عليك غسيل أطباق الصحون وتنظيف الأثاث وتنظيف الأرضية وترتيب المكتبة وترتيب غرفة النوم و ... الخ ) عند ذلك فإنك تبحث عن طريقة عملية لكي يتم إنجاز هذه المهمة في أسرع وقت فتقوم بتقسيم هذه المهمة الكبيرة على عدة أقسام (التنظيف ، الترتيب ، الغسيل ، ..... ) ثم تقوم باستدعاء أطفالك و فلذات أكبادك وتقسم على كل واحد منهم مهمة بسيطة يستطيع القيام بها .. هذا التقسيم يسمى في عالم البرمجة بالـ function (دالة أو وظيفة )

## Function

الدالة هي جزء من كود البرنامج يتم تعريفه عن طريق المبرمج ليتم تنفيذ شيء معين بواسطتها ، تقوم الدالة بأخذ قيم وتسمي (arguments معطيات ) كمدخلات ، ، ثم تقوم بعمل بعض التعديلات على هذه المدخلات وتقوم بإخراج قيمة أخرى في أكثر الأحيان تقوم الدالة بأخذ القيم ووضعها في متغيرات أخرى تسمي بالـ (parameters) لكي يتم إجراء العمليات عليها داخل الدالة وهذه المتغيرات لا تعمل خارج الدالة أي أنها متغيرات خاصة بالدالة فقط ! ... في دروسنا السابقة قمنا باستخدام دوال عديدة مثل دوال فرز المصفوفات ودوال إيجاد نوع البيانات ، ، ، هذه المرة سنقوم ببناء دوالنا الخاصة بنا ، ، ومن صنعنا نقوم بإعطائها المعلومات والبيانات وهي تقوم بإجراء العمليات عليها ومن ثم إخراج الحلول ...

## تعريف واستدعاء الدوال

لكي تقوم بتعريف دالة فإنك تقوم بكتابة الكلمة function متبوعة باسم الدالة والبارامترات اللازمة والتي سيتم إجراء العمليات عليها بين قوسين ومن ثم تقوم بكتابة الكود اللازم وسط { و }

الصيغة :

```
Function functionname (parameters)
{
    function code
}
```

تقوم بكتابه اسم الدالة بدلاً من functionname ثم تقوم بتعريف المتحولات أو المتغيرات parameters ومن ثم تقوم بكتابه الكود الذي سوف يقوم بالمطلوب بين القوسين بدلاً من function code

دعنا الآن نقوم بكتابة دالة من إنشاءنا والتي تقوم بإجراء عملية الجمع على متغيرين وسنقوم بتسمية الدالة باسم sumnormal وهو اسم من تأليفنا ويبدل على وظيفة وهدف الدالة ويمكن أن تقوم بتسمية الدالة بأي اسم تريده ولست مجيراً بكتابه اسم معين

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return $a;
}
?>
```

نقوم في هذه الدالة بإجراء عملية إضافة 100 على المتغير أو القيمة التي يتم تمريرها .

### Return

يجب أن نضعها في نهاية كل دالة ، نستخدم هذه الكلمة لكي نقوم بإعلام الدالة ان وظيفتها انتهت وإيضاً نستخدمها إذا كان لدينا أكثر من قيمة ونريد أن نقوم بإخبار الـ PHP ماهي القيمة التي سيتم اعتمادها ففي مثالنا هذا أردنا إخبار الـ PHP بأن يقوم بأخذ المتغير \$a بأنه هو القيمة النهائية مع أنه لو لم نضع المتغير فسيتم اعتباره هو الناتج النهائي لأنه لا يوجد متغير آخر تم عليه أي عمليات

الذي اقصدنا أننا لو كتبنا الكود بالشكل التالي :

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
}
?>
```

فإنه لاضرر من ذلك لأنه لا يوجد لدينا إلا قيمة واحدة لن يتم اعتماد قيمة غيرها ولكن لو افترضنا أنه لدينا أكثر من قيمة كما في المثال التالي :

```
<?
Function sul($a,$b)
{
$a = $a + 100 ;
$b= $b*100;
return $a ;
}
?>
```

هنا يجب تحديد أي المتغيرين سيكون هو القيمة النهائية للدالة .

### شرح الدالة (sumnormal)

تقوم الدالة التي صنعناها بأخذ قيمتين ومن ثم فإنها تقوم بزيادة العدد الذي يتم تمريره 100

ولكي نقوم بإخراج نتيجة الدالة فإننا ببساطة نستطيع ذلك بإجراء أحد الأمرين echo أو print .  
مثال :

```
<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
}
echo sumnormal(500);
?>
```

لقد قمنا بتمرير رقم بدلاً من المتغير ويمكننا أيضاً تمرير متغير بدلاً من الرقم  
مثال :

```
<?
Function sumnormal($a)
{
```

```

$a = $a + 100 ;
return ;
}
$f=100;
echo sumnormal($f);
?>

```

لاحظ أننا استخدمنا متغير في الدالة (مما يثبت كلامنا في الأعلى أن للدالة متغيرات خاصة بها) وليس معني ذلك أننا لانستطيع استخدام متغيرات بنفس الاسم المذكور في الدالة فيمكننا مثلاً كتابة نفس اسم المتغير بدون حصول أي مشاكل كالتالي :

```

<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
}
$a=100;
echo sumnormal($a);
?>

```

يمكننا أيضا استدعاء دالة بشكل عادي إذا كانت هي تقوم بالطباعة  
مثال :

```

<?
Function sumnormal($a)
{
$a = $a + 100 ;
print $a;
return ;
}

$a=100;
sumnormal($a);
?>

```

## print

يقوم الأمر print بنفس عمل الدالة echo ولا يوجد بينهما اختلاف سوى أن الدالة echo قديمة وهي الأصل أما الدالة print فقد تم إنشاؤها في php4 ولا يوجد أي فرق بينهما اطلاقاً .  
مثال :

```

<?
Print "احمد";
?>

```

ويمكننا بها إخراج نتيجة دالة

```

<?
Function sumnormal($a)
{
$a = $a + 100 ;
return ;
}
$a=100;

print sumnormal($a);
?>

```

## اين يتم وضع الداله ؟

يمكنك وضع الدالة في أول الكود أو في آخرها أي أنه لا فرق بين :

```

<?
//لاحظ اننا قمنا بتعريف الداله اولاً ثم استدعاءها//
Function fares($d)
{

```

```
print "alfareees@hotmail.com";
}

fares($d) ;
?>
```

وبين :

```
<?
// لاحظ اننا قمنا باستدعاء الداله اولا ثم تعريفها
fares($d) ;

Function fares($d)
{
print "alfareees@hotmail.com";
}
?>
```

يمكنك أيضا عدم وضع متغيرات في الدالة كالتالي :

```
Html_header ()
{
Print "<html><head><title>alfareees</title></head>";
Return ;
}
```

هذه الدالة تقوم بكتابة الطور الأول من صفحة html لاحظ أننا لم نضع أي متغيرات او عوامل او متحولات (سمها كما شئت).

### تمرير القيم الى الدالة

هناك نوعين من تمرير القيم

- 1 - تمرير القيمة مباشرة الى الداله (passing by value) وذلك أن نضع القيمة مباشرة بدون إدراجها في متغيرات . مثال :

```
<?
Function alfars ($f)
{
$f=$f+$f;
return ;
}
echo alfars(100);
?>
```

لاحظ أننا قمنا بإدراج القيمة مباشرة للدالة من غير وضعها في متغيرات .

- 2 - تمرير القيمة عن طريق المرجع (passing by reference)

نقصد بهذا أننا نقوم بوضع القيمة في متغير أولاً ثم نضع هذا المتغير في الدالة لكي يتم اجراء العمليات عليه مثال :

```
<?
Function alfars ($f)
{
$f=$f+$f;
return ;
}
$r = 1000;
echo alfars($r);
?>
```

### اعداد قيمة افتراضيه للدالة

تستطيع أن تجعل الPHP4 يقوم بإدراج قيمة افتراضية عند عدم تمرير متغيرات إليه  
مثال :

```
<?
```

```
Function alfars ($f=40)
{
$f=$f+$f;
return ;
}
echo alfars();
?>
```

إذا لم يتم إعطاء قيمة للدالة فإنها ستفترض أن القيمة هي 40 مباشرة .  
 أما إذا تم تمرير قيمة أو متغير فإنه سيتم العمل بالقيمة التي تم تمريرها بدلاً من القيمة الافتراضية  
 مثال :

```
<?
Function alfars ($f=40)
{
$f=$f+$f;
return ;
}
echo alfars(100);
?>
```

### مدي المتغيرات (variable scope)

هناك متغيرات محلية (local) ومتغيرات عامة (global) ، نقصد بالمتغيرات المحلية التي تكون في داخل الدالة ونقصد بالعامة التي تكون في كود الـ PHP بشكل عام  
 مثال

```
<?
// هذا متغير عام
$r= "salem";
function ala($s)
{
// هذا متغير محلي
$s = "programer";
}
echo $r ;
ala($s);
echo $s;
?>
```

مثال :

```
<?
// هذا متغير عام
$r= "salem";
function ala($s)
{
// هذا متغير محلي
$s = "programer";
}
echo $r ;
$s=10;
echo $s;
?>
```

في المثال الأول استطعنا طباعة المتغير \$r ولم نستطع طباعة المتغير \$s لأنه محلي (لا يتم تنفيذه الا داخل الدالة ) وعندما نريد طباعته فإننا يجب أن نطبع ناتج الدالة لكي نحصل عليه (أي أننا لانستطيع طباعته بشكل مباشر)  
 مثال :

```
<?
// هذا متغير عام
$r = "salem";
function ala($s)
{
// هذا متغير محلي
$s = "programmer";
}
// استطعنا طباعته بشكل مباشر
echo $r ;
ala($s);
```



يجب استخدام الدالة لكي يتم طباعته //

```
echo ala($s);  
?>
```

لاحظ أننا حتى لو قمنا بعملية طباعة المتغير من نفس الدالة فالناتج يكون مختلف لأن لكل متغير عالمه الخاص به لكي نقوم بجعل المتغير الذي بداخل الدالة متغيراً عاماً فيمكننا ذلك بإحدى الطريقتين التاليتين :  
**الطريقة الأولى :**

```
<?  
function ala($y)  
{  
echo $y. "<br>";  
global $s;  
$s = "programmer";  
return ;  
}  
$f =10;  
ala($f);  
echo $s;  
?>
```

لاحظ أننا عندما استخدمنا global في داخل الدالة لكي يتم تعريف أن المتغير متغير عام وعندما قمنا باستخدام الدالة قامت بطباعة المتغير المراد طباعته ومن ثم بعد ذلك قامت بتعريف متغير جديد (\$s) وهذا المتغير متغير عام لأننا وضعنا قبله الكلمة global فاستطعنا طباعته بكل سهولة .

**الطريقة الثانية :** هي أن نستخدم المصفوفة \$GLOBALS التي تستخدم في PHP لتعريف المتغيرات العامة أيضاً  
مثال :

```
<?  
function ala($y)  
{  
echo $y. "<br>";  
$GLOBALS["s"] ;  
$s = "programmer";  
return ;  
}  
$f =10;  
ala($f);  
echo $s;  
?>
```

### **المتغيرات المستقرة (static variable)**

اقصد بالمتغيرات المستقرة هي التي تكون قيمتها ثابتة  
مثال :

```
<?  
Function addfares($y)  
{  
$y;  
$y=$y+1 ;  
return $y;  
}  
echo addfares($y);  
echo addfares($y);  
echo addfares($y);  
echo addfares($y);  
?>
```

```
<?  
Function addfares($y)  
{  
static $y;  
$y=$y+1 ;  
return $y;  
}  
echo addfares($y);
```

```
echo addfares($y);
echo addfares($y);
echo addfares($y);
?>
```

لاحظ عندما عرفنا المتغير بأنه static فانه يحتفظ بقيمته حتي لو انتهت الدالة .

## دوال متداخلة

يمكننا عمل تعشيش للدوال مثلما كنا نفعل مع بناء القرارات والتكرارات  
مثال :

```
<?
Function sum($sa)
{
    $sa=$sa-1;
function goadd ($r)
{
    $r = $r+$r;
return $r;
}
$sa= goadd ($sa);
return $sa;
}
echo sum (15);
?>
```

في مثالنا هذا لدينا الدالتين الأولى هي sum والدالة الثانية هي goadd

وظيفة الدالة الأولى هي أن تقوم بالإنقاص من العدد الذي يمرر إليها واحد ثم تقوم بتطبيق دالة داخلية فيها هي goadd تقوم بزيادة العدد على نفسه .. ومن ثم قمنا ببناء الدالة الأولى (لأنها هي الأساس التي يوجد به الدوال الداخلية ) وطباعة قيمتها .

## اشتمال الملفات (include files)

قد يكون لديك في برنامجك متغير متكرر في أكثر من صفحة أو رسالة خطأ معينة أو تريد إدراج نص كبير الحجم في صفحات متعددة  
هنا يمكنك اشتمال ملفات في داخل ملفات ال-PHP . هذه الملفات قد تحتوي على نصوص أو كود html أو كود PHP .

إن الصيغة التي تستخدمها لاشتمال الملفات هي :

```
Include (filename);
```

مثال :

قم بفتح ملف نصي واكتب فيه ماتشاء ثم احفظه باسم a.txt  
قم بإنشاء ملف php واكتب فيه ومن ثم احفظه باسم b.php

```
<?
Include ("a.txt");
?>
```

انقلهما الى مجلد السيرفر .. شغل ملف ال-b.php وانظر النتيجة .  
يمكنك أن تقوم بإنشاء ملف PHP وتحتفظ فيه بجميع الfunction المطلوبة لبرنامجك وعند إرادتك لاستخدام أي واحدة منها تقوم فقط باشتمال الملف ومن ثم استدعاءها .

## داله تلوين الكود

هل رايت مواقع تقوم بتلوين الكود بشكل مذهل مثل موقع zend ؟.... الأمر بسيط كل ما عليك أولاً  
قم بوضع الكود في ملف نصي وسمه باي اسم (مثلا file.txt ) وبعد ذلك قم باستخدام الدالة

Show\_source

مثال :

```
<?
show_source ("file.txt");
?>
```

# تتبع وتصيد ومنع الأخطاء (avoiding and handling errors)

إن مصطلح debug هو من المصطلحات الشائعة والشيقة في عالم البرمجة ، هذا المصطلح يشير إلى كيفية إصلاح أخطاء البرنامج وتوقعها قبل حدوثها ، هناك أنواع من الأخطاء تحدث بسبب المبرمج وهناك أنواع من الأخطاء تحصل بسبب المستخدم ، في العادة يجب أن يكون المبرمج متألماً مع مصطلح تتبع الأخطاء وإصلاحها .

قد يكون من أهداف تتبع الأخطاء الحماية بقدر أهميه البرنامج الجاري العمل عليه أو الموقع فكلما كان الموقع مهماً كان وجوب حمايته أكبر .

قد يكون من الأسباب التي تسبب تدميراً للمواقع هو أن صاحب الموقع يغطي كل صغيرة وكبيرة عن برنامج الذي يركبه في موقعه وقد يكون برنامج هذا غير محمي بسبب كاف أو يكون مسير بعدة ملفات فيقوم شخص بحذف ملف من الملفات الأساسية بسبب عدم دقة في التراخيص المعطاة مما يؤدي إلى دمار الموقع نهائياً .

وقد يكون صاحب الموقع مهملًا في الحد ذاته فلا يحتفظ بالمعلومات السرية لموقعه مما يسبب مشاكل أكبر من التدمير مثل احتلال الموقع بشكل كامل .

رسائل الخطأ في PHP لها طريقتها وتقنياتها الخاصة التي تسير عليها فهي ليست مثل الجافا وليست مثل cgi فال PHP لا تقوم بإرسال الخطأ إلى السيرفر بل تقوم بكتابة رسالة خطأ في مكان الخطأ .

قد يكون هناك أخطاء يصعب تتبعها أو معرفة مكانها في الأصل ، وقد يكون هذا بسبب أنك تستخدم PHP في صناعة موقع ديناميكي وتشرك معها الجافا سكرت وتضع علامات التعليق الخاصة التي تقوم بإخفاء الأخطاء في الجافا مما قد يجعلك تشعر بالحيرة وتجن أين مكان الخطأ

```
<!--  
رساله الخطأ  
-->
```

## أنواع الأخطاء

هناك أنواع من الأخطاء منها الإملائية (Syntax Error) ومنها المنطقية ومنها أخطاء تحدث في وقت التنفيذ

ومثال الأخطاء الإملائية :

```
<?  
Eco "1";  
// من المفترض أن تكت التالي :  
Echo "1";  
>
```

هذا سيعطيك رسالة خطأ Parse error

ومن الأخطاء الإملائية نسيان الفاصلة المنقوطة (semi-colon) في نهاية الدالة :

```
<?  
Echo "hello"  
// من المفترض أن تكت التالي :  
Echo "hello";  
>
```

هنا سوف يعطيك PHP رسالة خطأ لكن العجيب أنه لن يعطيك إياها بشكل صحيح فرسالة الخطأ تشير إلى أن السطر الرابع يحتوي على الخطأ بينما الخطأ هو في السطر الثاني .

وهناك خطأ آخر يحصل بسبب نسيان brace ( وهي الأقواس ) :

```
<? Php  
for ($loop = 0 ; $loop < 5 ; $loop ++ )  
{  
Echo "";  
>
```

إذا كنت قد نسيت إغلاق القوس فهذا من الأخطاء الشائعة ، والأخطاء الإملائية لا يمكن حصرها ، إنها أشبه بقواعد اللغة ، لكن أكثر الأخطاء الإملائية الشائعة في برامج PHP

1 - نسيان الأقواس . مثال :

```
<?  
for ($loop = 0 ; $loop < 5 ; $loop ++ )  
{
```

```
for ($loop1 = 0 ; $loop1 < 10 ; $loop1 ++ )
{
for ($loop = 0 ; $loop < 5 ; $loop ++ )
{
code ....
}
}
```

في المثال السابق ينقصنا قوس إغلاق التكرار الأخير ( }

2 - نسيان الفاصلة المنقوطة . مثال :

```
<?
Echo 10
<?
```

3 - خطأ إملاتي في اسم function . مثال :

```
<?
Htmlspecialchars($I);
?>
```

سيعطيك رسالة خطأ :

```
Fatal error : call to Undefined function : htmlspecialchars().
```

وتصحيحها أن تكون :

```
<?
Htmlspecialchars($I);
?>
```

4 - نسيان إغلاق النص . مثال :

```
<?
Echo "arabuilder;
?>
```

نسي الـ(") في نهاية الكلمة . وسيعطيك Parse error

### الأخطاء المنطقية (Logical Errors)

إن الأخطاء المنطقية هي الأكثر صعوبة في التتبع فقد تجد برنامجك يعمل بشكل صحيح وبكل سلامة ولكنه عند نقطة ما لا يتم تنفيذها كما تريد أنت ، لنضرب مثلاً على خطأ منطقي بسيط جداً ، لنفرض أنك قمت بعمل نموذج مكون من مربع نص وزر ، عند ضغطك لهذا الزر فأنت تريد أن يتم كتابة كلمة كبير إذا كان الرقم أكبر من 30 وكلمة صغير إذا كان الرقم أصغر من 30 لنقم بكتابة الكود للمثال الأول :

```
<?
echo " ادخل عمرك " ;
echo '<br>
<form method = "post" action = "age.php">
<input type= "text" name = "age">
<br>
<input type= submit value = " هل أنا كبير أم صغير ؟ " >
</form>' ;
?>
```

في ملف age.php اكتب الكود التالي :

```
<?
If ($age<30) echo "انت صغير";
If ($age>30) echo "انت كبير";
?>
```

سيعمل السيكريبت بشكل صحيح .. ولكن ربما تخطأ أنت في كتابة العلامات المنطقية ( التي باللون الأحمر ) فتأتي النتائج بشكا خاطئ .

ومن الأخطاء المنطقية الأخطاء التي تقع في وقت التشغيل (Run times error) والتي تكون قد تقوم بإيقاف برنامجك بشكل كامل  
مثال :

```
<?
$t=0;
$r=1;
$f=$r/$t;
?>
```

وعندها سينتج لك الرسالة التالية

Warning : Division by zero in (path) on line (line number )

هناك نوع آخر من الأخطاء المنطقية (unexpected) وهو لا يقوم بإيقاف البرنامج نهائياً بل يقوم بإخراج رسالة الخطأ في مكان الخطأ أو قد يقوم بتنفيذ البرنامج وإخراج البيانات بشكل غير صحيح أو قد لا يقوم بإخراج بيانات وهو المثال الأول الذي ذكرناه سابقاً (تقييم العمى) .

### أخطاء التكرارات

قد يكون لديك أيضاً تكرار فيه خطأ ولا يقوم بالتوقف نهائياً مثل هذا التكرار :

```
$c=1;
$t=true;
while ($t=true)
{
$c++;
}
```

لم نقم بعمل شيء يوقف التكرار مثل أن تضع شرط يختبر قيمة المتغير (\$c) ثم يقوم بإيقافه عند تعديه رقم معين وعلى ذلك فإن التكرار سيستمر بشكل غير متوقف ولن يعمل البرنامج .

### عدم ارجاع قيمه من function

مثال :

```
<?
Function ($d)
{
$d = $d+$d;
}
```

الخطا هنا اننا لم نستخدم return لكي ننهي الدالة أو قد تكون الدالة تحتوي على أكثر من قيمة وننسى أن نقوم بتحديد القيمة النهائية للدالة

### الخط في المعاملات الحسابية والمنطقية

مثال :

```
If ($y=10) echo 12 ;
```

والمفترض أن تكون :

```
If ($y= =10) echo 12 ;
```

### أفكار جيدة لتفادي الأخطاء

التعليقات

إن من الأفكار الجيدة للتقليل من الأماكن التي تبحث فيها عن الخطأ هو وضع تعليقات لوصف وظيفة دالة معينة . مثال :

```
<?
// هذه الكود يقوم بطباعة كلمة أحمد
Echo "أحمد" ;
?>
```

### الدوال

وأيضاً من الأفكار الجيدة أن تقوم بتقسيم وظائف البرنامج على دوال بحيث أن لكل دالة وظيفتها المعينة :

```
<?
/*
+-----+
|         هذه الدالة تقوم بقسمه العدد علي 2         |
+-----+
*/
function ($U)
{
```

```
$U=$U/2;
return $U ;
}
?>
```

## Regular Expressions

هذه التقنية تساعدك على تفادي الأخطاء في صفحتك عند حدوثه مثل أن يقوم مستخدم ما بكتابة بريد الكتروني غير صحيح (مثال : a@y@.k.d) هذا البريد غير صحيح ولأجل أن تقوم بمنع حصول أي خطأ مثل ذلك وتقييد العبارات التي يدخلها المستخدم فإنك تقوم باستخدام الـ RE (Regular Expressions) إنك بالأصح تجعل قواعد للكلمات التي يدخلها المستخدم فمثلاً تجعل المستخدم لا يدخل سوي أرقام أو حروف فقط أو شكل معين من الكلمات ، تقوم أولاً بإنشاء نمط للكلمة التي تريد المستخدم أن يقوم بادخالها .

## النمط (pattern)

ماهو النمط ؟ ما رأيك إذا كتب المستخدم جملة في مربع نص تحتوي على عدة كلمات وتريد أن تتأكد من وجود كلمة معينة وسط هذه الجملة ، على حسب ما اخذناه من معلومات على المصفوفات سابقاً نستطيع فعل ذلك كالتالي :

```
<?
$words="one, two, three, four, five,";
$ty =explode ("",$ty);
foreach ($ty as $w) {
    if ($w == "six") echo "found string `two`";
}
?>
```

لقد كان المتغير \$words يحتوي على جملة تتكون من عدة كلمات وعندما أردنا فحصه قمنا باستخلاصه في مصفوفة ثم بعد ذلك قمنا بفحص المصفوفة باستخدام التكرار foreach ، ومع ذلك الذي فعلناه فإن هذا الاستخدام غير عملي بتاتاً وهنا تبرز قوة Regular Expressions لاحظ الآن كيف نستخرجه بواسطة الـ Regular Expressions :

```
<?
$words="one, two, three, four, five,";
if (ereg("one",$words)) echo "لقد وجدت العدد `one`" ;
?>
```

في هذا المثال قمنا باستخدام الدالة (ereg) ووضعنا في خانتها الأولى النمط (pattern) الذي نريد أن نتأكد من وجوده (أوالكلمة المراد البحث عنها) ووضعنا في الخانة الثانية المتغير الذي سيتم البحث فيه عن الكلمة أو النمط . تقوم الدالة ereg بإعطاء القيمة true إذا تم العثور على الكلمة . في الواقع هناك استخدامات أكثر فعالية للأنماط . يمكننا مثلاً تخزين الكلمة إذا تم وجودها في مصفوفة خاصة كالتالي :

```
<?
$words="one, two, one, four, five,";
if (ereg("one",$words,$rok)) ;
echo $rok[0];
echo $rok[1];
?>
```

نقوم بوضع اسم المصفوفة التي نريد تخزين البيانات في الخانة الثالثة .. لاحظ مع أنه يوجد كلمتين في الجملة توافق النمط إلا أنه أعطانا كلمة واحدة فقط إذ أن وظيفته أن يتأكد من وجود النمط في الجملة فقط فإذا تأكد من وجودها مرة واحدة استكفى واعتبر الموضوع قد انتهى .

ماذا لو أردنا من التأكد من عدة كلمات ، عند ذلك فإننا نفعل التالي :

```
<?
$words="one, two, one, four, five,";
if (ereg("one",$words,$rok)) echo $rok[0];
if (ereg("two",$words,$rok)) echo $rok[0];
?>
```

واريد أن أبهك أن الـ ereg يقوم بإنشاء المصفوفة من جديد عند كل استعمال له فخذ حذرک من هذه النقط أيضاً فإن الـ ereg حساس لحالة الأحرف لاحظ هذا المثال :

```
<?
$words="one, two, vcx, four, five,";
if (ereg("One",$words,$rok)) echo $rok[0];
?>
```

لن يقوم بإخراج أي شيء فقط لأن حرف الـ O مختلف .

أيضا يمكنك البحث عن كلمة يسبقها فراغ مثلاً كالتالي :

```
<?
$words="one, two, vcxone, four, five,";
if (ereg("one",$words,$rok)) echo $rok[0];
?>
```

مثال آخر :

```
<?
$words="oned, two, vcxone, four, five,";
if (ereg("one",$words,$rok)) echo $rok[0];
?>
```

لاحظ في هذين المثالين أنه مع أن كلمة one غير موجودة بمفردها إنما موجودة كجزء من vcxone وoned ورغم ذلك فإن الدالة لم تأخذ اعتباراً لذلك بينما لو كتبنا كالتالي :

```
<?
$words="oned, two, vcxone, four, five,";
if (ereg(" one",$words,$rok)) echo $rok[0];
?>
```

فإنه سيبحث عن الكلمة مفصولة عن أي حرف ولن يجد كلمة كذلك فلن يقوم بكتابة أي شيء .

يمكننا أن نفحص قيمة موجودة في متغير كالتالي :

```
<?
$reu = "one";
$words="one, two, vcxone, four, five,";
if (ereg($reu,$words,$rok)) echo $rok[0];
?>
```

هل لاحظت أننا فحصنا قيمة المتغير \$reu بواسطة ereg مع \$word ولم يتطلب منا ذلك أي شيء إضافي غير اسم المتغير المراد البحث عن قيمته في الجملة .

يمكننا بالـ Regular Expression استعمال بعض الأحرف بشكل خاص التي لها استعمالها الخاص بواسطة الـ Regular Expressions

الأحرف الخاصة في الـ Regular Expression هي كالتالي :

. \* ? + [ ] ( ) { } ^ \$ | \

هذه الأحرف لها معناها الخاص في الـ Regular Expression

فقدیماً مثلاً كنا نقول أنه لا يمكننا أن نستخدم علامتي تنصيص متداخلة من نفس النوع كالتالي :

```
<?
$r="u\"";
?>
```

ولكي يتجاهل الـ PHP هذا المعني فإننا نقوم بوضع (\) قبل علامة التنصيص .  
أيضاً مع الـ ereg فإن للـ (.) قداستها ولكي يتم تجاهلها فإننا نستخدم الـ (\)

تقوم الـ (.) بأخذ مكان حرف أو فراغ فمثلاً لاحظ المثال التالي :

```
<?
$P="I love yamen";
if (ereg ("love....",$P,$R)) echo $R[0];
?>
```

هل لاحظت الناتج ؟؟

ولكي يتم تجاهل قداسة الـ (.) في الـ Regular Expressions نقوم بوضع (\) قبلها . مثال :

```
<?
$P="I love yamen";
if (ereg ("love\\.\\.\\.\\.",$P,$R)) echo $R[0];
?>
```

في هذا المثال لن يتم طباعة أي شيء لأنه لا يوجد أي كلمة تطابق (love....) لأن الـ(.) فقدت قداستها وبدأ التدقيق في الكلمة حرفاً حرفاً .

### صناعة فئة حروف [xyz]

أقصد بذلك أنني احدد نطاق معين من الكلمة من الممكن أن يكون في هذا النطاق أي حروف من الفئة التي أقوم بتحديدتها أو الحروف التي أقوم بتحديدتها .  
مثال :

```
<?
$y="how are you ? " ;
if (ereg("h[oe]" , $y)) echo "true";
?>
```

هنا قام الـ regular expression بالبحث عن أي كلمة تبدأ بالحرف h ومن ثم يتبعها أحد الحرفين o أو e مثال هذه الكلمات :  
Hey – He – Hew - Homer

ولكنها لا تطابق :

Hty – Hnt - Hlay

أتمني أن تكون فهمت ما أرمي إليه

يمكننا أيضاً أن نقوم بإخبار الـ regular expression بأن لا يقوم باختيار كلمات تحتوي على حروف معينة وذلك فقط بإضافة ^

```
<?
$y="how are you ? " ;
if (ereg("h[^oe]" , $y)) echo "true";
?>
```

نقوم هنا بإخبار الـ re أن يقوم بفحص الجملة فإذا وجد أي كلمة تبدأ بـ h ولا تحتوي على o أو e فإنه يقوم بإعطاء true وإذا لم يجد يقوم بإعطاء false وهذا الكلام يطابق الكلمات التالية :

Hay - Hana - Hkg

ولا يوافق هذه الكلمات :

Home – Hore - Here

يمكننا استعمال اختصارات لبعض الأمور فمثلاً إذا كنا نريد كلمة لا تحتوي على أي رقم كنا سنكتب كالتالي  
[^123456789]

يمكننا أن نستعمل اختصار لهذا الموضوع كالتالي :

[^0-9]

وحتى إذا أردنا أن يتأكد من وجود رقم من واحد الى تسعة فقط علينا مسح الـ  
[0-9]

وأيضاً الحروف الصغيرة من a الى z

[a-z]

وإذا نريد التأكد من عدم وجودها

[^a-z]

نفس القصة مع الحروف الكبيرة .

هناك اختصارات أخرى لهذا الموضوع كالتالي :

الاختصار	المطابق له	معناه ووظيفته
\d	[0-9]	أي رقم من 0 الى 9
\D	[^0-9]	ممنوع الأرقام من 0 الى 9
\w	[0-9A-Za-z_]	أي رقم من 0-9 أو حروف A-Z او احرف صغيره او _
\W	[^0-9A-Za-z_]	عكس السابق
\s	[\t\n\r]	يقبل مسافة أو سطر جديد أو علامة جدولة (tab)
\S	[^\t\n\r]	عكس السابق

### تحديد مكان الكلمة

يمكننا أن نقوم بتحديد مكان الكلمة ، اقصد بذلك أنه يمكنك تحديد مكان الكلمة إذا كانت في بداية أو نهاية النص ونستخدم لهذا الأمر العلامتين (^) لتحديد المكان لبداية الجملة و (\$) لنهاية الجمل .

مثال :

```
<?
```



```
$y="how are you ? " ;
if (ereg("^h",$y)) echo "true";
?>
```

هنا سيقوم الـ php بالبحث عن في الجملة فإذا وجد الجملة تبدأ بحرف h كانت قيمة الـ ereg تساوي true وإذا لم يجد كانت قيمة الـ ereg تساوي false

```
<?
$y="how gone?" ;
if (ereg("^g",$y)) echo "true";
?>
```

في هذا المثال ستكون قيمة الـ ereg خطأ لأن العبارة لا تبدأ بحرف g يمكننا فعل العكس بواسطة العلامة (\$) التي عملها عكس (^) فهي تفحص إذا كان الحرف المراد فحصه موجود في نهاية الجملة  
مثال :

```
<?
$y="how g" ;
if (ereg("g$",$y)) echo "true";
?>
```

يمكننا أيضاً اختيار إذا ما كان واحد من نمطين صحيحاً بواسطة العلامة ( | )

```
<?
$y="how g" ;
if (ereg("^y | g$",$y)) echo "true";
?>
```

في هذا المثال سيقوم الـ PHP بفحص الجملة فإذا وافقت أحد النمطين كانت قيمة الـ ereg عند ذلك true .

يمكننا أيضاً تحديد إذا ما كان حرف أو جملة متكررة بعدد من المرات أو مره واحدة باستخدام أحد هذه الثلاث رموز ( \* ، + ، ? ) تقوم علامه الضرب بالتحقق من أن الحرف الذي يسبقها مكرر مرة أو أكثر أو غير موجود بتاتاً  
مثال :

Bea\*t

وتوافق :

Bet  
Beat  
Beaat

تقوم علامة الجمع (+) بالتأكد من وجود عنصر مرة أو أكثر :

Bea+t

وتوافق :

Beat  
Beaat  
Beaaaaat

أما علامة الاستفهام فتقوم بالتأكد من وجود عنصر مرة واحدة أو عدم وجوده بتاتاً :

Bea?t

وتوافق :

Bet  
Beat

وتأكد دائماً أن هذه الثلاث علامات مسبوقة بحرف .

وعند إرادتك مثلاً التأكد من سبق حرفين أو ثلاث بشكل تحديدي يمكنك استخدام القوسين  
مثال :

(wo)?man

ويوافق :

man  
woman

يمكننا التأكد من تكرر حرف بشكل معين من المرات أو أكبر من عدد معين من المرات أو أصغر من عدد معين من المرات باستخدام القوسين {x,y} فمثلاً لو أردنا أن نتأكد من أن حرف (d) مكرر مرتين إلى أربع مرات :

d{2,4}

أما إذا أردنا أن نتأكد من أنه مكرر أكثر من مرتين إلى عدد غير محدود من المرات :

d{2,}

أما إذا أردناه أن يتكرر 4 مرات على الأكثر :

d{,4}

أو إذا أردناه أن يتكرر بعدد محدود من المرات :

أخيراً نريد أن نلفت النظر إلى الاختصار (\b) الذي معناه أي شيء ولكن ليس حرفاً (الحروف التي بين \w وبين \W تقريباً)

ملخص ما أخذناه من القواعد تجدونه في الجدول التالي :

القاعده	المعني
[abc]	أي حرف كان a او b او c
[^abc]	أي حرف غير a و b و c
[a-z]	كل الحروف من a الى z
\d\D	\d للارقام و \D لغير الارقام
\w\W	\w للحروف جميعها و\W لغير الحروف
\s\S	\s للفراغ (space) و\S لغير الفراغ (no space)
\b	الحروف التي بين \w و\W
.	أي حرف
(abc)	تقوم باعتبار abc كمجموعه ..
?	حرف او مجموعة حروف مكرره مره اوغير مكرره نهائيا
+	حرف او مجموعة حروف تتكرر مره او اكثر
*	حرف او مجموعة حروف تتكرر مره او اكثر او قد لاتتكرر نهائيا
{x,y}	تكرير بعدد معين من المرات ..
{,y}	تكرير بحد اقصي من المرات ..
{x,}	تكرير بحد ادني من المرات ...
{x}	تكرير بعدد معين من المرات
^	في بدايه النص
\$	في نهايه النص

### تعبير للناكد من ايميل

^[a-zA-Z0-9-]+(\.[A-Za-z0-9-]+)\*@[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)\*\$

شرح التعبير

الرمز	الشرح
^	يجب ان يبدأ النص
[A-Za-z0-9-]	أي حرف من a-z كبيرا كان او صغيرا او _ او ارقام
+	وقد يكون هذا الحرف متكررا اكثر من مره

بالإضافة الى انه قد يتبع النقطه وحروف وارقام	(\.[_A-Za-z0-9-]+)
وقد لا يتبعه او قد يتبعه ويتكرر اكثر من مره	*
وبعد ذلك يكون لديه حرف ال@	@
وايضا نفس القواعد في النهايه	[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*\$

مثال :

```
<?
Function mailcheck($mail,$t)
{
$T="^[_a-zA-Z0-9-]+(\.[_A-Za-z0-9-]+)*@[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)*$";
If (EREg($T,$mail))
{
$r="the mail is true";
echo $r;
}
else
{
$r="the mail is not true";
echo $r;
}
return ;
}
mailcheck("alfareees@hotmail.com", $t);
?>
```

eregi()

الفرق بين هذه الدالة والدالة ereg أنه غير حساسة لحالة الأحرف كبيرة أو صغيرة أي أنه يمكننا كتابة المثال السابق كالتالي :

```
<?
Function mailcheck($mail,$t)
{
$T="^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*$";
If (EREg($T,$mail))
{
$r="the mail is true";
echo $r;
}
else
{
$r="the mail is not true";
echo $r;
}
return ;
}
mailcheck("alfareees@hotmail.com", $t);
?>
```

ereg\_replace()

ماذا لو أردت تحرير عبارة ما من أحرف معينة وقد تكون متكررة في جملة أو غير ذلك لنفرض أن لدينا العبارة التالية :

```
Mohmed love his game .....
```

ونريد أن نتخلص من النقاط التي في نهاية العبارة أو لدينا مثلاً هذا المسار :

```
C:\windows\desktop
```

ونريد أن نستبدل العلامة (\) ب (/)

كل ذلك ممكن بواسطة الدالة ereg\_replace وقواعد ال regular expression التي أخذناها سابقاً البنية التي نستخدمها للدالة كالتالي :

```
Ereg_replace(reg,string,var);
```

نضع في مكان reg القاعدة لل regular expression ونضع مكان string الحرف الجديد ونضع بدلاً من ال var المتغير الذي نريد استخلاص الحروف منه .

مثال :

```
<?
$path = " C:\windows\desktop";
$tell= "Mohmed love his game ....." ;
$newpath= Ereg_replace("[\."],"/",$path);
$newtell= Ereg_replace("\.","",$tell);
echo $newpath;
echo "<br><br>";
echo $newtell;
?>
```



استخدام عبارة echo

هو من أقدم الأساليب وكان يستخدم مثلاً في فحص بعض متغيرات نموذج فمثلاً أنت لديك نموذج يقوم بإرسال معلومات إلى النموذج وقد تستخدم في اختبار الأخطاء المنطقية التي يستصعب متابعتها في الكود

مثال :

```
<?
Echo "this is : $name";
Echo "<br>";
Echo "this is : $Email";
// كود يقوم بمعالجة معلومات المتغيرين
// طباعة المتغيرين بعد اداء عملية المعالجة ورؤية النتائج
Echo "this is after : $name";
Echo "<br>";
Echo "this is after: $Email";
?>
```

فحص كود الhtml

قد تستخدم كود جافا سكريبت ويتم إخفاء الأخطاء وسط علامات التعليقات فعليك حينئذ فحص كود الhtml لرؤية إن كان هناك بعض الأخطاء المخفية أم لا .

تجاهل الأخطاء

لنفترض أنك تعلم أن الدالة التي صنعها بها أخطاء ولكنك تريد تجاهل هذه الأخطاء فكل ما عليك أن تقوم بوضع @ أمام الدالة لكي يتم تجاهل الخطأ عند حدوثه .

مثلاً نحن نعلم أن القسمة على الصفر من الأشياء الغير مقبولة في الPHP وأنت صنعت دالة تقوم بالقسمة على صفر ولن يتم تنفيذها لأنها بالأصل خطأ ولكنك تريد أن يقوم الPHP بتجاهلها فكل ما عليك أن تفعله هو وضع @ أمام الدالة .

مثال :

```
<?
function amail ($y)
{
$y=$y/0;
return $y;
}
```

```
$s= @mail(44);  
echo $s;  
>
```

## التعامل مع العميل

كما رأينا في الدروس السابقة ، فإن PHP يوفر رقم عظيم من المميزات عن html لبناء مواقع الويب ، من الأشياء الأساسية التي لم نتكلم عنها حتي الآن هي الموثوقية (أو الاستقرار) وهو بالمعني الصحيح والصريح :

القابلية على الاحتفاظ بالمعلومات بين صفحتين منفردتين أو مختلفتين في المستعرض ...

بدون أي إضافات ، HTTP لا يوفر أي ميكانيكيه للحفاظ على البيانات وجعلها مستقرة لمعالجة تتم بين صفحتين ، كل طلب لصفحة في الانترنت (request) ليس له أي علاقة بأي طلب آخر ... مثلاً عندما تتطلب موقع المطور العربي ومن ثم منندي المطور العربي فان كل الطلبين ليس لهما علاقة ببعضهما ...

بمصطلح آخر يمكننا أن نقول أن الـ HTTP فاقدة لحالتها (stateless) أي أنها لاتعرف أي أن أمر طلب الصفحة ينتهي عند انتهاء الطلب ، فهي عندما تقوم بنقل بيانات صفحة من السيرفر الى المستخدم فهي تعرف من هو المستخدم الذي يطلب البيانات وعلى أي نافذه سيتم نقل البيانات وعند انتهاء ذلك فان كل هذا الموضوع ينتهي وإذا عاد المستخدم فطلب صفحة أخرى فإنها لاتعرف إن كان هو نفس المستخدم أو لا !

إن القدرة على الحفاظ على وجود البيانات ليست وسيلة أو ميزة او قوة مقتصره على الـ PHP فقط .

فلقد رأيت كيف استطعنا ارسال معلومات من صفحة إلى صفحة بدون خسران أي معلومات وذلك عن طريق الـ html وبالرغم من ذلك فإن المستخدم عندما يقوم بإغلاق الصفحة عند استقبالها للبيانات فان ذلك يعني فقدانها للأبد ، عن طريق استخدام الـ PHP يمكننا اخبار السيرفر بأن يقوم بارجاع البيانات بطريقة تمكننا من الحفاظ عليها ، مثلما سنري في هذا الدرس ، هناك ثلاث طرق لعمل ذلك ....

التميز الحقيقي في قوة الفهم للـ PHP ، يتطلب منا مفهومية جيدة في كيفية استعمال الـ PHP في التفاعل مع المستخدم والمتصفح الذي يستخدمه لكي نتغلب على نقاط الضعف التي في الـ http .

هذا هو موضوعنا لهذا اليوم والذي سنتكلم فيه عن :

- 1 - الـ HTTP والـ html ومحدودية قدراتهم ، وكيف يستطيع الـ PHP التغلب على القصور فيهم .
- 2 - الاحتفاظ بالمعلومات التي نريد أن نستخدمها بين طلب لصفحتين مختلفتين .
- 3 - مكنكة الحفاظ على البيانات .
- 4 - الكعكات (cookies) وكيفية استخدامها .
- 5 - الـ PHP والـ native session - المكنكة الداخلية للحفاظ على وجودية البيانات .

هذا الدرس مفيد بشكل ظاهري لمن هو جديد على انشاء مواقع متفاعلة متوسطة - كبيرة الحجم بواسطة الـ PHP .. إنه يحتوي على الكثير من بعض الأمثلة التي تفيدك .

الهدف من هذا الدرس هو أن نتعرف على كيفية الحفاظ على معلومات المستخدم عبر متغير أو أكثر بين أكثر من صفحة ، مثل أن تجعل اسم المستخدم ظاهر في كل صفحة يقوم بالولوج إليها ... مما يؤكد استمرارية وجود البيانات .

لنفرض أن لدينا موقعاً على الانترنت هذا الموقع يهتم ببيع وتسويق مواد غذائية أو أن هذا الموقع يقدم مسابقات ثقافية ، في العادة عندما يقوم المستخدم بطلب شراء سلعة معينة أو عندما يختار الدخول في مسابقة من المسابقات الثقافية فإنه يقوم بدخول أكثر من صفحة بالتتابع

يختار السلعة في الصفحة الأولى وبعد ذلك يقوم برؤية معلومات السلعة في الصفحة الثانية والصفحة الثالثة يقوم فيها بتعبئة معلوماته للشراء أو غير ذلك إلى أن ينتهي من كافة المعلومات وبعد ذلك تنتج له في النهاية صفحة فيها معلوماته والسلعة التي قام باختيارها وفاتورة شراء !!

أو يقوم باختيار نوع المسابقة الثقافية في الصفحة الأولى وبعد ذلك يقوم بالحصول على عدة أسئلة مقسمة على عدة صفحات إلي أن ينتهي من المسابقة فتخرج له في النهاية مجموع الدرجات للأسئلة ومعلوماته وهل هو فائز أم خاسر!!

في الواقع هذا مايسمونه بالمحافظة على الجلسة (maintain session) وأقصد بذلك دخول المستخدم إلى صفحة وانتقاله من صفحة إلى صفحة مع المحافظة على معلوماته وغير ذلك من البيانات ، لكي نستطيع متابعته أولاً بأول .

في بروتوكول الhtml والhttp لانستطيع معرفة إذا ما كان الشخص عندما يطلب صفحة ما هو نفسه عندما يذهب إلى الصفحة الثانية إذ أن المستخدم عندما يطلب صفحة ما (request) من السيرفر فإن السيرفر يقوم بمعرفة من أي مكان بالعالم يتكلم هذا الشخص ويقوم بإرسال استجابته إليه باعطاءه الصفحة التي كان يطلبها (response) ولكن بعد ذلك فإن السيرفر لا يعرف إذا كان هذا الشخص هو نفسه الذي يقوم بطلب الصفحة الثانية أو الثالثة في السيرفر .

هنا تأتي ميزة الPHP وغيره من لغات برمجة الانترنت لصناعة ميكانيكية إبقاء تفاعل مستمر بين المستخدم والسيرفر عن طريق الsession و cookie ، ولكي لا نعقد الموضوع دعونا نتكلم عن ذلك عملياً فذلك أفضل لفهم الموضوع من الثرثرة التي لا فائدة منها .

## استخدام الحقول المخفية

سنقوم الآن بإنشاء ثلاث صفحات ، الصفحة الأولى تطلب من المستخدم ادخال اسمه ، والصفحة الثانية تقوم بالترحيب به واعطائه ثلاثة أسئلة ، والصفحة الثالثة تقوم باعطاءه النتيجة .

افتح محرر نصوص لديك واكتب الكود التالي :

```
<p dir="rtl" align="center">ادخل اسمك الكريم</p>
<form method="POST" action="quiz2.php">
<hr>
<input type="text" name="name" size="20"><br>
<input type="submit" value="إرسال" ></p>
</form>
```

احفظها باسم quiz.php

قم بفتح محرر النصوص واكتب الكود التالي :

```
<html dir="rtl">
<?
If (isset($name)) {
Echo "مرحباً بك يا $name" ;
Echo '
<br>
<form method="POST" action="quiz3.php" dir="rtl">
<input type="hidden" name="thename" value="' . $name . '">
<p>من هو أول الخلفاء الراشدين ؟
<p dir="rtl"><input type="radio" value="أبوبكر الصديق" name="khlifa">أبوبكر
الصديق .</p>
<p dir="rtl"><input type="radio" value="عمر بن الخطاب" checked name="khlifa">عمر
بن الخطاب</p>
<p dir="rtl">من هو الفاروق ؟</p>
<p dir="rtl"><input type="radio" name="faroq" value="عمر بن الخطاب">عمر بن الخطاب
</p>
<p dir="rtl"><input type="radio" name="faroq" value="سالم" checked>سالم بن
عامر</p>
<input type="submit" value="إرسال" dir="rtl">
</form>' ;
}
else
{
echo "غير مصرح لك بدخول هذه الصفحة" ;
}
?>
```

احفظها باسم quiz2.php

قم بفتح المفكرة واكتب الكود التالي :

```
<?
If ((isset($thename) && (isset($khlifa) && (isset($faroq))))
{
echo $thename ;
$range=0;
$co = 0;
if ($khlifa == "أبوبكر الصديق" ) {
$range=$range+10;

```

```

$co = $co +1;
}
if ($faroq == "عمر بن الخطاب")
{
    $range=$range+10;
    $co=$co+1;
}
if ( $range < 10)
{
    echo "ليس هناك أي إجابة صحيحة ";
}
else
{
    echo "<br>". "$co = عدد الاسئلة التي أجبت عليها";
    echo "<br>". "$range . " الدرجة التي حصلت عليها";
}
}
?>

```

قم بوضع الملفات الثلاثة السابقة في مجلد السيرفر ثم قم بتشغيلها

## الشرح

قمت في هذا المثال بمحاولة صنع مكنكة تواصل للبيانات ، بمعنى أنني أحاول أن أقوم بالاحتفاظ بالبيانات عبر الثلاث صفحات بشكل متواصل ، لاحظ أنني كنت اختبر في quiz2 و quiz3 باختبار المتغيرات قبل طباعة أي شيء فقد يقوم المستخدم مثلاً بالاحتفاظ بالصفحة التي وصل إليها في المفضلة ثم يقوم باكمال المسابقة في وقت آخر ولكني لا أريد ذلك بل أريد ان أجعل وقتها محدوداً (طبعاً هذا الكلام سيحصل إذا كانت المسابقة طويلة) لذلك فإنني في كل عند الانتقال من صفحة إلى صفحة أقوم باختبار إن كانت جميع هذه القيم موجودة ولاحظ أنني كنت اجتفظ دوماً بقيم المتغيرات في متغيرات جديدة في حقول مخفية وكلما كان عدد المعلومات أكبر في كل مره كان عدد الحقول المخفية أكثر ، إن لهذه الطريقة أيضا مشاكلها فقد يفتح المستخدم كود الhtml ويقوم بتفحص كيفية ملاحظته عبر المسابقة وقد يصنع هو الكود في وقت لاحق لكي يستطيع اكمال المسابقة بهذه الخدعة الماكرة ... لذلك يفضل أن لا تقوم بذلك وتقوم بجعل المسألة السابقة أكثر تعقيداً باستخدام الregular expression بمحاولة تلغيم البيانات بواسطة ومن ثم فك هذا التلغيم في الصفحات التي تصل إليها البيانات .

## ارسال بيانات بواسطة query strings

نستطيع ارسال بيانات بسيطة بواسطة الاستعلامات التي نقوم بإضافتها الى اسم الصفحة في الأعلى متبوعة بـ(?) علامة استفهام ثم اسم متغير وقيمته وإذا كان هناك أكثر من متغير يتم الربط بينهم بعلامة & وراجع درس النماذج لمزيد من المعلومات

قم بعمل صفحة وسمها ask.php وقم بكتابة الكود التالي فيها :

```

<?
If (isset($ask)) {
    If ($ask == login) {
        Echo "تم تسجيل الدخول إلى الصفحة ";
    }
}
if (!isset($ask)) {
    echo "<br>". "لم يتم تسجيل الدخول إلى الصفحة ";
    Echo "<a href=\$PHP_SELF?ask=login>اضغط هنا ليتم تسجيل دخولك</a><br>";
}
?>

```

قم بتجربة هذا المثال على موقع يدعم PHP على نظام تشغيل لينوكس إذا لم يعمل بشكل جيد على الوندوز

لاحظ أننا في أول الولوج الى الصفحة لم نستخدم أي استعلامات وعند الضغط على الرابط قام الرابط بارسال قيمه المتغير الذي يقوم الPHP باختبارها فإذا وجد انه قد تم ارسالها (بواسطة الرابط الذي تم الضغط عليه) قام بطباعة (تم تسجيل الدخول) وإذا لم يجدها قام بطباعة (لم يتم تسجيل الدخول) بالإضافة إلى طباعة الرابط الذي يحتوي على المتغير في طياته

## الكوكيز أو الكعكعات (cookies)

إذاً ماهي الكوكيز ، الكوكيز هي عبارة عن بعض المعلومات أو القطع الصغيرة من البيانات يتم الاحتفاظ بها في جهاز العميل لكي يتم الاحتفاظ بها عند الزيارات المختلفة للمستخدم (العميل) ، أنت لا تقوم بالاحتفاظ فيها بقيم ضخمة لكنك تستفيد منها في أشياء أخرى مثل :

1 - جعل لكل مستخدم الألوان الخاصة التي يري فيها صفحتك ( أي أن تجعل للمستخدم مثلاً إعدادات الألوان الخاصة لرؤية موقعك ) .

2 - جعل مفتاح للمستخدم لكي يستطيع به التحكم في بياناته الخاصة عند زيارته لموقعك في مرات اخرى.

الكوكيز مفيد للاستخدام في الأشياء البسيطة والغير خطيرة ، لكنه الآن يستخدم بشكل سيئ ، مثل استخدامه مثلاً في معرفة معلومات عن المستخدم بدون علم منه ، أو تخزين كميات كبيرة من البيانات فيه والتي من الأجدر أن يتم حفظها في ملف على السيرفر .  
ويكون استخدامه مفيداً عندما تضمن أن جميع زوار موقعك تسمح متصفحاتهم بالكوكيز (مثل طلبة المدارس أو شبكات انترانت )  
عندما يكون فقط لأشياء بسيطة لاضرر منها عند عدم السماح بالكوكيز بجهاز العميل .

### بدايتك مع الكوكيز

قبل أن نبدأ علينا معرفة بعض الأساسيات عن الكوكيز  
الكوكيز عبارة عن قطعة صغيرة من البيانات التي تستخدم لتخزين اسم متغير وقيمه مع معلومات حول الموقع التي أتت منه وتاريخ انتهاءها .

الكوكيز عباره عن تقنية للتخزين من جهة العميل (client-side storage) تتخزن في ملفات في جهاز العميل

يتم العبور إلى هذه الكوكيز ومسحها من المكان التي ارسلت منه .

عندما يطلب المستعرض صفحة من السيرفر وهذه الصفحة تقوم بتخزين كوكيز فإن السيرفر يقوم باخبار المستعرض بأنه سيقوم بوضع كوكيز للاستعمال لاحقاً .

عندما يتم طلب الصفحة في مرة أخرى يقوم المستعرض بإرسال البيانات التي تم إنشاؤها سابقاً عند طلب الصفحة .  
يتم انتهاء مده الكوكيز بإنهاء وقت صلاحيتها المحدد من قبل السيرفر ويتم مسحها فوراً عند اغلاق الصفحة إذا كان وقت صلاحيتها صفرأ من الثواني .

باختصار عندما يعطي السيرفر الكوكيز للمستعرض فإنه يقول لك هذا شي اذكرك به في وقت لاحق (قد يكون هذا الوقت من ضغط رابط آخر في الصفحة التي زرتها حتي بعد أسبوع أو أكثر) .

يقوم السيرفر بإرسال الكوكيز عبر الـ HTTP Headers الذي يتم إرساله قبل أي مخرج من مخرجات الـ HTML والمستعرض أيضا يقوم بإرسال الكوكيز عبر الـ HTTP Header بالإضافة إلى أن المستعرض يتعرف على من سيقوم بإرسال الكوكيز فلو كانت الكوكيز مثلاً مرسله من قبل الموقع [www.php.net](http://www.php.net) فإنه لن يقوم بإرسالها إلى موقع [www.phpbuilder.com](http://www.phpbuilder.com)

باستطاعتك عند إنشاء الكوكيز تحديد مسار يتم ارسال الكوكيز لكي يتم اقتصار عملية العبور إلى الكوكيز إلى أماكن معينة .  
قبل أن نقوم بوضع كود بسيط سنقوم الآن بتعريف كيفية تخزين الكوكيز وكيفية قراءتها :  
كون الـ PHP لغة حديثه لعمل سكريبتات ويب فإنها تأتي بدعم كامل للكوكيز بواسطة الدالة (`setcookie()`) باستثناء أنك عند استعمالها يجب استعمالها قبل طباعة أي مخرجات الـ HTML .

تاخذ الدالة (`setcookie()`) ثلاث معاملات ، الثلاثة الأولى هي الأهم والأمثل استخداماً وهي بالترتيب :

✓ قيمة حرفية يتم تخزينها كاسم للمتغير

✓ قيمة حرفية يتم تخزينها كقيمة لذلك المتغير

✓ Unix timestamp الذي يقوم بالإشارة إلى تاريخ إنتهاء الكوكيز

Unix timestamp عبارة عن رقم صحيح لا يحتوي على فواصل عشرية يقوم بحساب الثواني من منتصف ليلة 01/01/1970 .  
وإذا كنا نريد مثلاً أن نقوم بمسح الكوكيز بعد ساعة من تخزينه فإننا نقوم باستعمال الدالة (`time()`) التي تقوم بحساب الـ timestamp ثم نضيف عليه الوقت الذي نريده وفي حالتنا الساعه تساوي 3600 ثانية وعلى ذلك سنقوم بإضافة ناتج الدالة `time()` على 3600 لكي يتم مسح الكوكيز بعد ساعة واحدة !

الثلاث العوامل الأخرى التي يتم استخدامها أيضاً في الكوكيز ولكنها نادرة الاستخدام ولن نناقشها في موضوعنا هذا هي :  
n المسار الذي يتم إرسال الكوكيز إليه فلو تم فتح نفس الصفحة من نفس الموقع ولكن من مسار آخر ( مثلاً المسار كان `pag\url\one` وتم تغييره إلى `page\url\two` فإن المستعرض لن يقوم بإرسال البيانات إلي الصفحة لأنه تم تحديد المسار الذي سيتم إرسال الكوكيز اليه )  
n الدومين الذي سيتم إرسال البيانات إليه وهو مفيد في حالة ما إذا كان هناك أكثر من دومين تريد إرسال الكوكيز اليه  
n متغير من نوع integer يتم الإشارة إليه بـ `secure` يتم في حالة استخدام عمليات تشفير بالـ SSL

العبور إلى الكوكيز بسيط جداً فالمتغير الذي يتم إرساله يتم تخزينه ضمن المتغيرات العامة (global) وعندئذ فإنه لو كان لدينا كوكيز اسمه `ahmed` فإن قيمته توضع مباشرة في متغير اسمه `$ahmed` !!

يمكننا مسح الكوكيز بأكثر من طريقة ، بالطبع فإن المستخدم يستطيع مسح الكوكيز وتغيير محتوياتها بنفسه ولكن في حالة ما إذا أردنا أن نجعل السيرفر يقوم بمسحها فإننا نستخدم إحدى هاتين الطريقتين



إما أن نقوم بإخبار السيرفر بوقت قديم :

```
<?
Set cookie ("ahmed" , "0" , time()-999);
?>
```

وإما القيام بمسح الكوكيز بكتابة اسمه فقط :

```
<?
Setcookie ("ahmed");
?>
```

**مثال لتخزين وقراءة كوكيز**  
قم بفتح المفكرة واكتب الكود التالي :

```
<?
If ($thename) setcookie ("rname" , $thename , time()+3600);
Echo '<form method="post">
<input type = "text" name="thename">
<input type="submit" value="تسجيل">
</form>' ;
echo " قيمه المتغير الذي لديك " . " " . $thename . "<br><br>";
echo " = قيمة الكوكيز " . $rname ;
?>
```

## الشرح

### عند تشغيل الصفحة لأول مره

عند تشغيلك للصفحة سيتم اختبار ما إذا كان هناك متغير بالاسم \$thename فإذا تم الحصول عليه فسيتم وضع قيمته في كوكيز باسم (rname) (وطبعاً لن يتم الحصول عليه في أول مرة لأننا لم نقم بإرسال أي بيانات بعد) وبعد ذلك طباعة نموذج مربع نص واحد وزر لإرسال المعلومات .  
ويتم طباعة قيمة المتغير إذا كان هناك أي متغير تم إرساله باسم \$thename ويتم فحص قيمة الكوكيز \$rname وطباعتها وبالطبع لا يوجد حتي الآن أي كوكيز .

### المرحلة الثانية

الآن قم بكتابة أي شيء في مربع النص (اكتب اسمك مثلاً) ثم قم بضغط زر الإرسال سيتم إرسال البيانات الى نفس الصفحة ولكن هذه المرة سيتم تسجيل قيمة المتغير الذي يحمل البيانات في الكوكيز (rname) وبعد ذلك سيتم طباعة النموذج بشكل عادي وسيتم طباعة قيمة المتغير \$thename ولكن لن يتم طباعة قيمة المتغير \$rname لأننا فقط قمنا بتسجيله ولم يتم إرساله عند طلب الصفحة (لأننا نعرف أنه يتم إرسال الكوكيز عند طلب الصفحة وهذه المرة عندما طلبنا الصفحة لم يكن الكوكيز موجوداً بالأصل فلم يرسله السيرفر وقمنا نحن بتسجيله استعداداً للمرحلة القادمة) .

### المرحلة الثالثة

في هذه المرة سيكون الكوكيز موجوداً فسيتم إرساله على هيئة متغير ويتم إرساله ومن ثم طباعة النموذج وقيمه المتغير \$thename وقيمة الكوكيز الذي يوجد بجهازك !

## بدايتك الى الـ session

الـ session هي عبارة عن تقنية للترابط مع المستخدم وهي موجودة ضمن الـ PHP4 ولم تكن موجودة ضمن الإصدارات التي قبله بل كان يجب أن تقوم بتركيب مكتبة لكي تستطيع استخدام هذه التقنية ، يعتمد فهمنا للـ session على فهمنا للكوكيز وكيفية استعمالها ولقد تكلمنا عن الكوكيز بشكل جيد في الدرس السابق ، يستخدم الـ session لعمل ميكانيكية تواصل بين المستخدم والسيرفر ، فلقد قلنا أن الـ http لا يوفر لنا ميكانيكية لعمل تواصل ، فإذا طلب المستخدم صفحة من السيرفر فإن السيرفر يقوم بإعطائه ما أراد وينتهي عند ذلك فلا يعرف إن كان هو نفس المستخدم أو ليس هو ... لأجل ذلك تم انشاء تقنية الـ session لأجل عمل تقنية تواصل بين المستخدم والموقع ، فإستطاعتك مثلاً أن تقوم بتحديد عدد زيارات مستخدم معين لصفحتك ليوم واحد أو لأسبوع أو لمدة معينة من الوقت ... أو يمكنك عمل متجر إلكتروني بسيط يستطيع المستخدم شراء عدة أشياء دفعة واحدة من الموقع ويكون على تواصل بينه وبين الموقع عندما يقوم بإضافة مشتري إلي سلة التسوق أو حذف مشتريات .

قبل أن أتكلم عن كيفية استخدام الـ Session وإعطاء بعض الأمثلة البسيطة ، سأقوم بالتكلم عن كيفية إعداد الـ session مع الـ PHP .

### إعدادات الـ session في الـ PHP

لكي تستطيع التعامل مع الـ session بشكل جيد يجب عليك أن تتعرف على بعض الإعدادات التي في ملف الـ php.ini عندما تفتح الملف ستجد قسمًا خاصاً فيه بالـ session هناك حوالي 19 إعداد ولكن لن نتطرق إليها كلها بل سنتكلم عن الأساسية والمهمة منها فقط . كيداية لنا للتعرف على الـ session وكيفية عمله .

### إعداد طريقة التخزين

```
session.save_handler (files | mm | user)
```

ستجد هذه العبارة مكتوبة في الملف كالتالي بشكل افتراضي :

```
session.save_handler = files
```

وهذا الإعداد يقوم بتحديد طريقة التخزين للـ session وهناك ثلاث حالات للتخزين :

1 - التخزين في ملفات عادية على السيرفر :

```
session.save_handler = files
```

2 - التخزين على ذاكرة السيرفر :

```
session.save_handler = mm
```

3 - التخزين بطريقة أخرى معرفة ومعينة من قبل المستخدم مثل التخزين في قواعد البيانات وهذا ما سوف نقوم بالتفصيل عنه بعد الكلام عن قواعد البيانات :

```
session.save_handler = user
```

يجب أن تأخذ في اعتبارك عدد الملفات التي سيقوم الـ session بتخزينها عند استخدامك للأعداد الأول والإفتراضي خاصة عندما يكون عدد الزوار بالآلاف .

قد يكون استعمال الذاكرة أسرع ولكن المشكلة أنه من السهل مسح البيانات منها ببساطة .

الطريقة الثالثة قد تكون أكثر الطرق مرونة ، ولكنها معقدة وصعبة جداً ، وهي تعطيك مرونة لتخزين البيانات في أي وسائط مدعومة من قبل الـ PHP مثل قواعد بيانات mysql و oracle .

الذي افترضه الآن أنك قمت بوضع قيمة هذه الخاصية إلى files

## إعداد مكان التخزين

```
session.save_path (path/to/directory)
```

هذه الخاصية مفيدة إذا كنت قد ضبط الإعداد السابق إلى files تقوم هذه الخاصية بتحديد مكان التخزين على السيرفر ومن الأفضل أن تقوم بتحديد مكان التخزين بعيداً عن مجلد السيرفر لكي تمنع تصفح هذه الملفات .

## الإنشاء التلقائي للـ session

```
session.auto_start (0 | 1)
```

هذا الإعداد يقوم بتحديد إذا ما كان الـ session سيتم إنشاؤه تلقائياً عند كل زيارة للموقع أو لأي صفحة من صفحاته بدون إدراج كود الـ session في كل صفحة ... وعلى ذلك فإنك تقوم بوضع القيمة إلى (1) إذا أردت ذلك . وعلى افتراض أنك لا تحتاج إلى أن تجعل الـ PHP يقوم بعمل session لكل صفحة تلقائياً ومن غير طلب فستقوم بوضع قيمة هذا الإعداد إلى (0)

## SID

عندما يقوم الزائر بزيارة صفحتك فإن الـ session يستطيع تتبع هذا الزائر وعدد المرات التي قام فيه الزائر بالدخول لليوم الواحد ، يقوم الـ PHP بعمل SID (session identifier) أو رقم معرف تلقائي بشكل افتراضي عندما تقوم بطلب إنشاء session بالزائر ، وكل رقم معرف يختلف عن الآخر تماماً ، إن رقم المعرف الذي ينشئه الـ PHP شبيه للشكل التالي :

```
fc94ad8b1ee49ef79c713ee98ac1fcc4
```

هناك طريقتين يستطيع بها الـ PHP متابعة الـ SID للمستخدم :

- 1 - عن طريق المتابعة والتخزين بتسلسل في الكوكيز .
- 2 - عن طريق اتباع رقم المعرف بعنوان الصفحة في الانترنت .

سنأخذ أمثلة عن كلا الطريقتين :

## 1 - استخدام الكوكيز

بالطبع هذه هي أكثر الطرق شيوعاً للحصول على ترابط بين المستخدم والموقع وهي الأسهل ، ولكن يجب أن تضع في اعتبارك أن المستخدم قد يكون قد ألغى أو منع ميزة الكوكيز في المتصفح أو قد يكون متصفحه لا يدعم الكوكيز . خذ في اعتبارك أن بعض المتصفحات لا تسمح بأن يزيد حجم الكوكيز عن 5 كيلوبايت .

هناك بعض الإعدادات البسيطة في ملف php.ini التي يجب معرفة معلومات عنها قبل البدء باستخدام الـ session مع الكوكيز :

```
session.use_cookies (0 | 1)
```

هذه الخاصية تحدد ماذا كان يمكنك استخدام الكوكيز مع الـ session أو لا وعند وضع القيمة (0) فهذا يمنعك من استخدام الكوكيز مع الـ (session) وأما إذا كانت قيمته (1) فهذا يسمح باستخدام الكوكيز مع الـ session

```
session.name (Default: PHPSESSID)
```

هذا الإعداد يقوم بتحديد اسم الكوكيز الذي سيحتفظ برقم المعرف (SID) والإعداد الافتراضي هو PHPSESSID ولن اقوم بتغيير هذا الإعداد لكي تستطيع فهم المثال الذي سأطرحه بعد قليل

```
session.cookie_lifetime (Default: 0)
```

يقوم هذا الإعداد بتحديد المدة التي سيبقى فيها الكوكيز الذي يحتفظ بقيمة الـ(SID) والإعداد الافتراضي هو صفر ، أي أنه سيتم مسح الكوكيز تلقائياً بعد اغلاق المستخدم لنافذة المتصفح مباشرة

```
session.cookie_path (Default: /)
```

يقوم هذا الإعداد بتحديد مسار دومين يتم تخزين الكوكيز له .. لا تقم بتغيير قيمته ودعه كما هو

```
session.cookie_domain (Default: null)
```

يقوم هذا الإعداد بتعريف اسم دومين يتم تخزين الكوكيز لصالحه .. والقيمة الافتراضية هي null ، لا تقم بتغييرها

ضع في اعتبارك انه اذا كانت قيمة الاعداد (session.use\_cookies) تساوي واحد فان لا داعي لاستدعاء الدالة set\_cookie() لإعداد الكوكيز بل سيتم اعدادها تلقائياً بواسطة الـ PHP

## 2 - الإضافة أو الكتابة إلى عنوان الصفحة

إن إضافة عنوان الـ SID إلى عنوان الصفحة يعتبر من الأشياء البشعة جداً رغم أن طريقته سهلة ومفيدة في حالة ما إذا كان الكوكيز غير مدعوم في المتصفح بشكل جيد  
مثال :

```
<a href="configure.php?<?=SID?>">Go to the configuration page</a>
```

بهذه الطريقة نقوم بإضافة المتغير المرجعي SID الذي سيقوم بإعطاء رقم معرف للمستخدم .

## متابعة الـ session

لقد أخذنا حتي الآن معلومات تجعلنا ندخل عالم البرامج المسيره بالـ session بدون خوف ، سأبدأ الآن في طرح بعض الأمثلة البسيطة التي تثبت لديك بعض المفاهيم الأساسية في الـ session ... سأشرح في هذا المثال كيفية إنشاء الـ SID وتخزينه لاستعماله لاحقاً ، و خلاصة السيناريو للصفحة أننا نريد من المستخدم أن يفهم أنه يستطيع تخصيص لون الخلفية الذي يريد أن يشاهد به صفحات موقعنا ... سأقوم بتخزين قيمة ميدئية في المتغير الذي يقوم بتحديد لون الصفحة ، أنا افترض طبعاً أن المتصفح يدعم الكوكيز :

سكربت يقوم بإنشاء وتسجيل متغير session

```
<?
session_start();
session_register("zx");
session_register("co");
$zx=10;
$co++;
echo "<br>". "مرحبا بك في موقعنا أيها الزائر الكريم";

echo "عدد زيارتك لهذه الصفحة" . $co ;
echo "<br>";
echo '<a href="php2.php">الصفحة الثانية</a>';
?>
```

أقصد بالجلسة هي الـ (session) وإن كانت الترجمة غير صحيحة ولكن فقط نأخذه كمصطلح .  
متغير الجلسة هو الـ (session-variable) أو متغير الـ session أو سمه ما شئت .

## الشرح

يقوم هذا السكربت في البداية بإنشاء متغير اسمه (zx) ومتغير اسمه (co) وقمنا بإعطاء القيمة (10) للمتغير (zx) وقمنا بزيادة القيمة الموجودة (وهي الصفر) في (co) بواحد وكتبتنا مرحبا بك أيها الزائر الكريم في موقعنا ، ثم قلنا له إن عدد زيارتك لهذه الصفحة هي قيمة المتغير (co) ثم اعطيناه رابط للصفحة الثانية .  
في الواقع إن هذه المتغيرات وقيمها يتم الاحتفاظ بها في كوكيز له اسم خاص قمنا بتحديدده سابقاً من ملف PHP.ini ، وهذا الكوكيز يحتفظ بقيمة الـ SID للـ session .

نحن لا نقوم بإخبار الـ PHP أين سيحتفظ بقيمة المتغيرات لأننا بدأنا بكلمة الـ :

```
session_start();
```

وعلى هذا فإن الـ PHP سيفهم أنه سيقوم بتخزين القيمة في الكوكيز الخاص بالـ session .

قمنا بجعل المتغير co كعداد بسيط لعدد المرات التي سوف نقوم بها بزيارة الصفحة فعند عمل تحديث للصفحة سيتم زيادة العداد بمقدار واحد

```
$c++;
```

وطبعاً قبل زيادة العداد بقيمة واحد فإنه يتم حساب القيمة السابقة للمتغير عند إنشائه تلقائياً ... ومن ثم يتم الزيادة وبعد ذلك طباعة القيمة .

### كتابة رقم الـ SID

اكتب الآن الكود التالي واحفظه باسم php2.php

```
<?
session_start();
echo $PHPSESSID . "<br>";
echo $zx;
?>
```

في هذه الصفحة نقوم بطباعة قيمة الـ SID وذلك بطباعة قيمه المتغير \$PHPSESSID (الذي هو اسم الكوكيز الخاصة بالـ session).

بعد ذلك قمنا في النهاية بطباعة قيمة المتغير \$zx لكي ألفت نظرك بأن الكوكيز ما زال يحتفظ بها ولم يفقدها لأننا قد حددنا الإعداد في ملف php.ini الخاص بوقت الكوكيز الـ 3600 أي لمدته ساعة ثم بعد تلك الساعة سيتم مسح الكوكيز ولن يمكنك استرجاع قيمة أي متغير :

```
session.cookie_lifetime = 3600
```

واضف إلى معلوماتك أنه لا يمكنك قراءة القيم للكوكيز الخاصة بالـ session إلا عن طريق إضافه الأمر

```
session_start();
```

يجب أن تبدأ بهذا الأمر دائماً إذا أردت قراءة قيم المتغيرات التي يحتفظ بها الكوكيز الخاص بالـ session .

### مسح متغير من الـ session

كل ما عليك فعله هو استخدام هذه الدالة :

```
session_unregister(variable name);
```

تقوم بوضع اسم المتغير في مكان الـ (variable name)

مثال :

```
session_unregister("brn");
```

سيقوم هذا الأمر بمسح المتغير (brn) من الكوكيز الخاصة بالـ (session)

### قراءة قيم المتغيرات في الكوكيز الخاصة بالـ session

كل ما عليك فعله هو استخدام الدالة :

```
session_encode();
```

مثال :

```
<?
session_start();
session_register("bgcolor");
session_register("name");
session_register("email");
$bgcolor = "#8080ff";
$name = "alfareees almolthem";
$email = "php@php.com";
$e = session_encode();
print "The encoded string is: $e";
?>
```

بهذا السكريبت نكون قد أنهينا درسنا عن مقدمة بسيطة للـ session . هذه مجرد مقدمة ولكي نستطيع أن نتعمق بالـ session فيجب علينا أن نتعلم شيئاً عن قواعد البيانات .