الفصل الأول

مراجعة حول اللغة HTML

1-1: مقدمة.

جاءت التسمية HTML من الاسم (**H**yper **T**ext **M**arkup **L**anguage) أي لغة تأشـــير النصوص الفائقة, حيث تتكون هذه اللغة من مجموعة من الأكواد التي تنسق في ملف نـــصي و يحفظ بأحد الامتدادين (.html) أو (.htm) ليتم استعراضه على أحد مستعرضات الويب (مثلاً Internet Explorer), و هناك العديد من محررات لغة الـــ HTML و الرائد بينها هو برنامج MS Front Page, تسمى أكواد هذه اللغة أوسمة و تتألف كل مجموعة من زوج مـــن الأوســـمة, وسم بداية شكله<tag>, و وسم نهاية</tag>, حيث يدل المحرف (/) علـــى أن هذا الوسم هو وسم إغلاق, هناك العديد من الأوسمة ليس لها وسم إغلاق سنتعرف عليها لاحقاً.

2-1: البنية العامة لملف الـــ HTML.

تتألف البنية العامة لملف الـــ HTML من الأجزاء المبينة في الشكل التالي:

```
<html>
<head>
<title>عنوان الصفحة</title>
</head>
<body>
جسم الصفحة, و كافة ما سيعرض على المستعرض
</body>
</html>
```

يحـــصر الـــزوج <html> و </html> ملـــف الـــ HTML كاملاً بما فيه من الترويسة و العنوان و جسم الملف, فلا بد أن نبدأ و ننتهي بهذا الزوج, أما الزوج <head> و </head> فيحدد ترويسة الملف و يحتوي مجموعة من الأوسمة أهمها <title> و </title> الذي يحدد عنوان الصفحة الذي سيعرض في شريط العنوان للمستعرض عند عرض الصفحة, أما الزوج <body> و </body> فيحتوي جسم الملف الذي سيترجم و يعرض في المستعرض.

3-1: أوسمة تنسيق النصوص.

1. وسم الفقرة ..... و الذي يحدد فقرة من النص تنسق كلها حسب ما ورد من خواص ضمن هذا الوسم, مثلاً:

```
<p align="center" dir="rtl">
```

```
جسم الفقرة
</p>
```

سيتم عرض هذه الفقرة و تكون محاذاتها في المنتصف, و جهة النص من اليمين إلى اليسار.

2. وســـم السطر الجديد <br>, و الذي ينشئ سطر جديد ضمن النص الذي ورد فيه و في مكان وروده, مثلاً:

```
You can take any of: <br> red <br> blue <br> green
```

سيعرض كما يلي:

```
You can take any of:
Red
Blue
green
```

3. وسم الخط <font…>… الذي يأخذ الشكل التالي:

```
font face="Tahoma" size="3" color="#ff0000">
النص الذي نريد أن يكون بهذا التنسيق
/font>
```

سيعرض النص بخط Tahoma و بالحجم 3 و سيكون لونه أحمر.

4. الوســـوم الثلاثة المعروفة … و … و <u>…</u>, و التي تجعل النص عريض و مائل و مسطر على الترتيب, مثلاً:

```
<b><i><u>I am Bilal</u></i></b>
```

سيعرض الجملة بالتنسيق:

I am Bilal

انتبه تغلق الوسوم بترتيب معاكس لترتيب فتحها.

5. الحجـــوم المعرفة للنصوص بالوسوم <h1>…</h1> و <h2>…</h2> و ذلك حتى <h7>…</h7>, و التي تجعل الخط بحجوم مختلفة (1 الصغير 2 أكبر فأكبر حتى 7 أكبر حجم خط), مثلاً:

```
<h1>Size1</h1><br>
<h2>Size2</h2><br>
<h3>Size3</h3><br>
<h4>Size4</h4><br>
<h5>Size5</h5><br>
<h6>Size6</h6><br>
<h7>Size6</h7><br>
```

سيكون لها الخرج التالي:

```
Size1
```

```
Size2
Size3
Size4
Size5
Size6
Size7
```

توجد أيضاً وسوم أخرى متعددة لتنسيق النص و لكن ما سبق أهمها.

1-4: وسوم الارتباط.

و التـــي تعتبر من الأساسيات التي لا غنى عنها في الموقع إذ تستخدم للتنقل بين الصفحات و للتنقل في الصفحة الواحدة عندما تكون طويلة,مثلاً:

```
<a href="page1.html" >Go To Page1</a>
```

ســيولد فـــي الـــصفحة الحالـــية الارتباط Go To Page و عند النقر عليه ننتقل إلى الصفحة Pgae1.html المحـــددة ضـــمن "....=href", و هـــذا رابط بين صفحتين, أما الرابط بين موقعين ضمن الصفحة فله الشكل:

```
<a href="#position">Go To Position</a>
.......
.......
.......
.......
<a name="position">Here Is The Position</a>
```

يـــولد الكـــود الـــسابق ارتباط Go To Position ضمن الصفحة, و عند النقر عليه ننتقل إلى العبارة Here Is The Position.

انتبه إلى أن خاصتي href و name لهما نفس القيمة ما عدا أن href تزيد بـــ # قبل القيمة.

إن الرابط الذي يقوم بإرسال بريد الكتروني له الشكل:

```
<a href="mailto:Name@WebSite.com">Send To Name</a>
```

يـــولد الكـــود الـــسابق عـــبارة Send To Name و التي بالنقر عليها يتم فتح برنامج البريد الافتراضـــي (مثلاً: Outlook Express), و في حقل Send to قد كتب البريد الالكتروني المحدد ضمن الارتباط (Name@WebSite.com).

1-5: وسم الصورة.

الشكل العام لوسم الصورة هو كالتالي:

```
<img    src="images/name.jpg"    width="50"    height="75"    border="2"
```

```
align="center">
```

الكـــود الـــسابق يقوم بإدراج الصورة التي اسمها name.jpg الموجودة في المجلد images المحددة بالعرض 50 و الارتفاع 75, و يجعل لها إطار بعرض 2, و تتوسط الجزء التي هي ضمنه.

يمكـــن جعـــل الصورة تشكل ارتباط ما, وذلك بإدراج وسمها ضمن وسمي الارتباط, كما في المثال التالي:

```
<a href="page1.html" >
<img src="images/name.jpg" width="50" height="75" border="2" align="center">
</a>
```

الآن أصـــبحت الـــصورة name.jpg تـــشكل ارتـــباطاً و بالنقـــر عليه ننتقل إلى الصفحة Page1.html.

1-6: وسوم القوائم(التعداد الرقمي و النقطي).

و تستخدم هذه الوسوم لجعل مجموعة من الأسماء أو البيانات منسقة ضمن قائمة, فمثلاً القوائم الرقمية لها الشكل التالي:

```
<OL type="1">
<Li>Number1
<Li>Number2
<Li>Number3
<Li>Number4
</OL>
```

الوسوم السابقة لها الخرج:

1. Number1
2. Number2
3. Number3
4. Number4

حيث حددت الخاصة type نمط الترقيم (قد يكون:.....,a,A,i,1).

أما القوائم النقطية فلها الشكل التالي:

```
<UL type="circle">
<Li>Number1
<Li>Number2
<Li>Number3
<Li>Number4
</UL>
```

الوسوم السابقة لها الخرج:

- Number1
- Number2
- Number3

```
        o   Number4
```

حيث حددت الخاصة type نمط التنقيط (قد يكون:.....,circle, square).

1-7: وسوم الجداول.

الشكل العام لوسم الجدول هو كما يلي:

```
<table...........>
    <tr.....>.............................................بداية السطر
        <td.....>هنا بيانات الخلية</td>
        <td.....>هنا بيانات الخلية </td>
    </tr>.............................................نهاية السطر


     <tr....>.............................................بداية السطر
        <td.....>هنا بيانات الخلية</td>
        <td.....>هنا بيانات الخلية </td>
    </tr>.............................................نهاية السطر
    ........................
    ........................
</table>
```

حيث:

1. الزوج <table>.....</table> يحددان بداية و نهاية الجدول.
2. الزوج <tr>.....</tr> يحددان بداية و نهاية سطر (صف أفقي)من الجدول.
3. الزوج <td>....</td> يحددان بداية و نهاية خلية ضمن الصف.

تتــضمن الأجزاء <table>,<tr>,<td> عادة مجموعة من محددات الجدول مثل عرض حد الجدول و الخلايا و نمط الخط ضمن الجدول و لونه ونمطه ولون تعبة الخلايا و العرض لكل خلية و اتجاه الخلية و عدة محددات أخرى تؤمنها محررات الــ HTML.

1-8: وسوم النماذج.

تــستخدم النماذج لإعطاء صفحة الويب صيغة التفاعلية مع الزائر و تمكنه من إجراء عمليات الاختيار و التحديد و تقديم المعلومات و إرسالها, يتألف النموذج من عناصر متعددة سنتعرف عليها الآن, يتألف وسم النموذج من الشكل العام التالي:

```
<Form action="Page.php" method="post  أو  get">
تدرج هنا عناصر النموذج و تحدد خصائصها ضمن وسومها
```

```
</Form>
```

إذاً:

1. الزوج <form>.....</form> يحدد بداية و نهاية النموذج و تدرج عناصر النموذج ضمن هذا الزوج.
2. يتضمن النموذج <form> الخاصتين المهمتين action و method, حيث تحدد الخاصة action عنوان الصفحة التي سنرسل لها بيانات عناصر النموذج, أما الخاصة method فتحدد طريقة إرسال البيانات و تأخذ إحدى القيمتين post و تعني إرسال البيانات إلى خارج المخدم و ترسل البيانات بشكل أزواج اسم المتحول و قيمته أما القيمة get فنستخدمها إذا كانت معالجة البيانات ستتم على نفس المخدم و لن يكون هناك عملية إرسال.

سنتعرف الآن و بشكل سريع على أهم عناصر النموذج, لدينا ثلاثة أجزاء و هي:

1- العنصر **Input**, شكله العام:

```
< خصائص متعددة للعنصر..... "نوع عنصر الإدخال"=Input type<
```

حيث يتوفر مجموعة من العناصر التي تندرج تحت الـ Input و يتم تحديدها بالخاصة Type, و هي:

- **العنصر Text:** و هو مربع نص بسط واحد يتم تحديد خصائصه ضمن وسمه, و من أهم هذه الخصائص لدينا (1): الخاصة name و التي يتم استخدامها في الصفحات التالية على أنها المتحول الذي يحمل قيمة مربع النص هذا, (2): الخاصة value و التي تحدد القيمة المبدئية (أي المحتويات) لمربع النص و التي يمكن تغييرها مع كتابة شيء جديد فيه, (3): الخاصة size و التي تحدد حجم مربع النص بالمحارف التي يتسع لها, بالإضافة لخواص أخرى متعلقة بنوع النص وما إلى ذلك, مثلاً:

```
<input type="text" name="user" size="20"  value="Here is your text">
```

- **العنصر Password:** لا يختلف عن الـ Text إلا أن محارفه تكتب بشكل نجوم, حيث أنه يستخدم لحقول كلمات المرور, مثلاً:

```
<input type="password" name="Text" size="20" value=" ">
```

- **العنصر Hidden:** و هو العنصر المخفي و يستخدم ليحمل قيم متحولات عندما لا نريد عرضها للمستخدم أثناء تنقلها بين الصفحات, حيث أنه لا يعرض على المستعرض شيء يدل عليه, مثلاً:

```
<input type="hidden" name="age"  value="form1">
```

- **العنصر Radio:** و يستخدم مع مجموعة عناصر مماثلة له لتمن المستخدم من انتقاء خـــيار من مجموعة خيارات تحددها هذه العناصر, إذ يطلب الإجابة عن سؤال و قد حددت أجوبته كتسميات لعناصر Radio, و يتم تفعيل عنصر Radio واحد من بين مجموعة عناصر عندما تحمل نفس خاصة الـــ name و يتم التعامل مع هذا العنصر عن طريق الخاصة value التي تحدد قيمته, يتمتع هذا العنصر بالخاصة Checked و التـــي تحدده فيما إذا كان محدد بشكل افتراضي للمرة الأولى, و عدم وجودها يعني أنه غير مفعّل, مثلاً:

```
Where are you??????<br>
<input type="radio" name="R1" value="V2">I am here<br>
<input type="radio" name="R1" value="V1" checked >I am there<br>
```

- **العنـــصر Checkbox:** و تم عن طريقه قبول أو رفض خيار واحد أمام الزائر أثناء المعالجة, يتميز بالخاصة name, و الخاصة value التي تحدد قيمته هل هو منتقى (ON) أو غير منتقى (OFF),و الخاصة Checked و التي تحدده فيما إذا كان محدد بشكل افتراضي للمرة الأولى, مثلاً:

```
<input type="checkbox" name="married" value="OFF" checked>
```

- **العنـــصر Submit:** و هو عبارة عن زر للنموذج الذي يستخدم للانتقال إلى الصفحة التالية و بنفس الوقت نقل متحولات النموذج إلى هذه الصفحة, مثلاً:

```
<input type="submit" value="أرسل الآن" name="send">
```

- **العنصر Reset:** و هو زر أيضاً لكن بالضغط عليه يتم إرجاع كافة عناصر النموذج الأخرى إلى قيمها الافتراضية (يسمى زر تصفير أحياناً), مثلاً:

```
<input type="reset" value="مسح البيانات" name="reset1">
```

2- العنـــصر **Select:** و هو قائمة الاختيار المنسدلة أو المتعددة الخيارات, الشكل العام لهذا العنصر هو:

```
<select size="1" name="food"............>
<option value="......">
<option value="......">
<option value="......">
.............
</select>
```

حيث:

- تحـــدد الخاصة size عدد الخيارات التي ستعرض في القائمة دون فتحها, فإذا كانت قيمتها (1) تصبح كصندوق النص و لكن يمكن فتحها لتعرض الخيارات الأخرى, أما إذا كانت قيمتها (2) فيظهر خيارين سوية دون فتحها, و هكذا....., أذكر هنا أنه يمن تحديـــد خيار و احد فقط من هذه القائمة إلا إذا وضعنا الخاصة Multiple التي تبين أنه يمكننا تحديد أكثر من خيار, و يتوجب على المستخدم عند تحديده عدة خيارات أن يضغط زر Ctrl طيلة فترة الاختيار.
- تحدد الخاصة name اسم القائمة و التي ستحمل القيم المختارة من القائمة.
- تحدد كل Option خيار ضمن القائمة يندرج فيها, و لكل option قيمة value تحدد الخيار المنتقى و يحمل لاسم القائمة على أنه نتيجة اختيار.

3- العنـــصر **TextArea**: و هـــو عنصر ناحية النص أو مساحة نصية, و يستخدم لتقديم إمكانية كتابة التعليقات و الملاحظات للزائر, الشكل العام له:

```
<textarea rows="10" name="notice" cols="30">
..................
</textarea>
```

حيث:

- تحدد الخاصة rows حجم المساحة النصية العمودي المعروض بالأسطر.
- تحدد الخاصة cols حجم المساحة النصية الأفقي المعروض بالأعمدة.
- الخاصـــة name تحدد اسم المساحة الذي سيكون بمثابة متحول قيمته هي محتويات المساحة, هناك خصائص أخرى لهذا العنصر تكميلية.

أذكر أن هذا الفصل لم يكن إلا تذكرة سريعة و عامة عن اللغة HTML التي كان لا بد منها للدخول إلى الـــ PHP, حيث أننا سنستخدم هذه الأساسيات لتنسيق الصفحات باستخدام PHP.

الفصل الثاني

مقدمة إلى PHP

1-1: ما هي PHP ؟

أتـــت التسمية PHP من العبارة **P**ersonal **H**ome **P**age, و هي لغة برمجة تستخدم غالباً لبـــناء مواقع الويب, و يمكن تشغيل برامج PHP على مخدم محلي أي على حاسب شخصي,

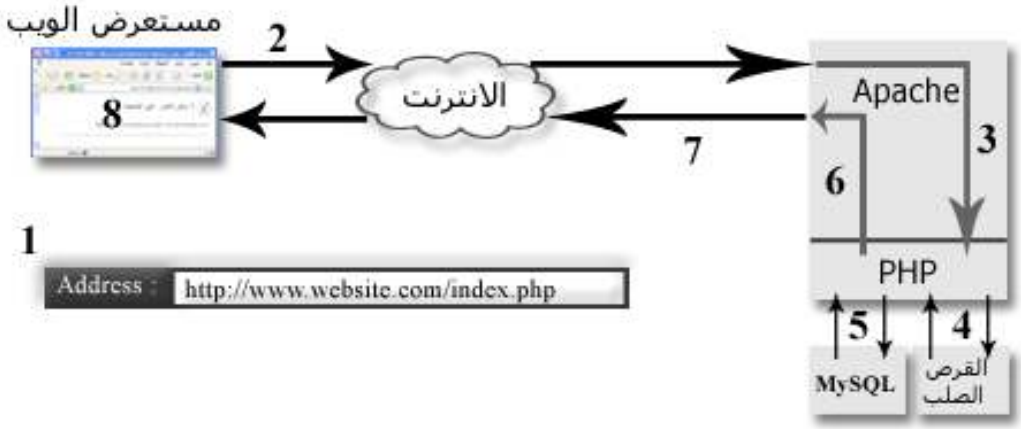
أو علـــى مخدم الويب, و يمكن الوصول لها و العمل عليها باستخدام مستعرضات الويب على جهاز المستخدم, و لا بد أن أنوه إلى أن لغة الـــ PHP لغة مجانية مفتوحة المصدر, و يمكننا الحصول على الكثير من أكوادها عبر شبكة الانترنت .

1-2: الاتصال مع صفحات PHP على المخدم.

يمكن حصر خطوات إجراء اتصال مع صفحة الـــ PHP مخزنة على مخدم ويب و عرضها على المستعرض بما يلي:

1. نقـــوم بكـــتابة عـــنوان صفحة الـــPHP في شريط العنوان (الموقع), مثلاً سنستدعي الصفحة: http://www.website.com/Index.php.
2. يقـــوم مـــستعرض الـــويب بإرســـال رســـالة عبر الانترنت إلى الكمبيوتر المسمى www.website.com طالباً منه الصفحة Index.php.
3. يـــستلم الـــ Apache (برنامج يعمل على الكمبيوتر www.website.com) الرسالة و يسأل مترجم الـــ PHP (PHP Interpreter: أيضاً برنامج يعمل على الكمبيوتر) عن الصفحة Index.php.
4. يقـــوم متـــرجم الـــــ PHP بقراءة الصفحة Index.php من ذاكرة المخدم (قرص التخزين).
5. يقـــوم مترجم الـــ PHP بتنفيذ الأوامر في الصفحة Index.php و عادة ما يتم تبادل بيانات مع برنامج قواعد البيانات (مثلاً: MySQL).
6. يقـــوم متـــرجم الـــ PHP بأخذ خرج برنامج الصفحة Index.php و يعيده إلى الـــ Apache كإجابة له.
7. يقـــوم الـــــ Apache بإرسال محتويات الصفحة التي حصل عليها من المترجم عبر الانترنت كإجابة لطلب مستعرض الويب.
8. يعـــرض مـــستعرض الويب الصفحة على الشاشة متبعاً تعليمات وسوم HTML في الصفحة.

الشكل التالي يوضح هذه الخطوات:



الشكل (1): مراحل الاتصال مع صفحات PHP على المخدم

1-3: كيف نبني صفحات الـــ PHP و نختبرها على مخدم محلي؟

لا بـــد لـــنا من استخدام محرر نصوص عادي (كالمفكرة مثلاً) و ذلك لكتابة و تنسيق أكواد PHP, و يجـــب حفـــظ صـــفحات الـــ PHP كملفات نصية بالامتداد **(PHP.)**, و تجمع مع بعضها البعض ضمن هيكلية محددة عندما تتبع لنفس الموقع.

و لكـــن مع انتشار هذه اللغة توفرت محررات أكواد PHP تقدم تسهيلات عدة في مجال هذه اللغـــة مـــثلما يقدم الـــ MS Front Page تسهيلات في تنسق أكواد الـــ HTML, من هذه البرامج (PHP Coder Pro) و (PHP Expert Editor).

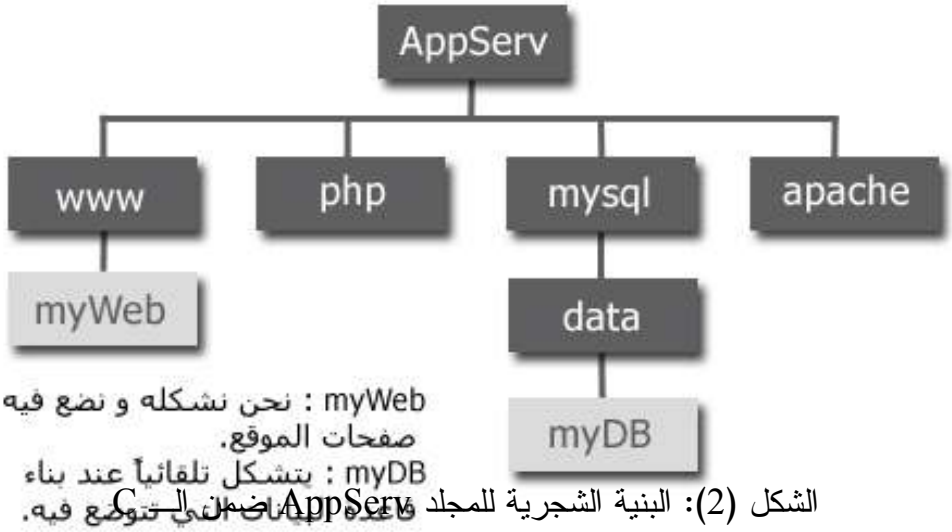
أمـــا لتشغيل الصفحات أثناء برمجتها (بغية التجريب) و أثناء إنهائها و تجريبها على حاسبنا الشخـــصي فإنـــنا نستخدم البرنامج **Apache Server** و الذي له عدة إصدارات, يعمل هذا البرنامج على جعل الحاسب و كأنه مخدم ويب رئيسي فيقدم خدمة ترجمة صفحات الـــ PHP و يقدم خدمة تنظيم و تشغيل قاعدة بيانات الـــ MySQL.

أثناء تنصيب هذا البرنامج علينا مراعاة النقاط التالية:

- نبقي اسم المخدم (Server name) كما هو, أي: localhost.
- نكتب اسم المستخدم: root.
- نبقي كلمة المرور فارغة.

إذ ســـيكون ذلـــك مهمـــاً أثناء تأسيس الاتصال مع قاعدة بيانات MySQL, و ستكون النقاط الثلاث السابقة بارومترات هذا الاتصال.

فـــبعد تنصيب الـــ Apache Server بشكل صحيح تصبح البنية الشجرية له كما في الشكل التالي:



الشكل (2): البنية الشجرية للمجلد AppServ ضمن الـــ C

و يكون مسار الصفحة Index.php هو http://localhost/myweb/index.php

1-4: كيف ندرج كود الـــ PHP ضمن الـــ HTML ؟

يقوم مترجم PHP بمعالجة أجزاء البرنامج التي تنحصر بين وسمي بداية و نهاية كود PHP, و مـــا هو خارج هذين الوسمين فيتم إخراجه دون تعديل, و بالتالي فإن هذين الوسمين يؤمنان سهولة في تضمين كود PHP ضمن كود HTML.

يـــبدأ كود PHP بالوسم (**<?** أو **<?php**) و ينتهي بالوسم (**?>**), و هناك طريقة أخرى و هي أن نبدأ بالوسم (**<Script Language="php">**) و ننهي بالوسم (**</Script>**).

مثال:

```
<html>
<head><title>First Page</title></head>
<body>
This is the first program by PHP.
<br>
<?
echo "Welcome to you friend...";
?>
</body>
</html>
```

بعد ترجمة هذا الكود البسيط بواسطة مترجم الـــ PHP في المخدم سيكون له الخرج التالي:

```
<html>
<head><title>First Page</title></head>
<body>
This is the first program by PHP.
<br>
Welcome to you friend...
</body>
</html>
```

هذا الخرج سيرسل إلى المستعرض و سيكون الخرج على المستعرض هو:

```
This is the first program by PHP.
Welcome to you friend...
```

لاحظ أننا استخدمنا التعليمة (echo) و هي تعليمة إخراج, تقابل الـــ (cout<<) في C++, و هناك تعليمة مكافئة لها في PHP و هي (print).

<u>ملاحظة هامة:</u>

قام مترجم الـــ PHP بترجمة الكود:

```
<?
echo "Welcome to you friend...";
```

```
?>
```

إلى الخرج (و هو كود HTML):

```
Welcome to you friend…
```

كمـــا لـــو أننا أدرجناه ضمن الوسمين (<body>….</body>) بدون التعليمة (echo), إذاً أعاد مترجم PHP لنا كود HTML و بالتالي يمكننا استخدام التعليمة (echo) لتشكيل وسوم HTML, كما في المثال التالي:

```
<?
echo "<html>";
echo "<head><title>Second Page</title><head>";
echo "<body>";
echo "<b><i>I study PHP…</i></b>";
echo "</body>";
echo "</html";
?>
```

و الذي سيكون خرجه على المستعرض:

I study PHP…

و ذلك بعد ترجمته إلى HTML.

<u>انتبه:</u>

- تنتهي كل تعليمة خرج (**echo**) بفاصلة منقوطة (**;**), و لا غنى عنها.
- علامات التنصيص المزدوجة ("……..") استخدمناها لتحديد أن ما نخرجه هو نص و ليس اسم متحول.

1-5: التعليقات و تنسيق الأكواد في PHP.

غالـــباً ما يستخدم المبرمجون أدوات التعليقات و الملاحظات التي يدرجونها ضمن أكوادهم إذ أنهـــا تـــسهل الرجوع لأجزاء الكود و تبين (مع ضخامة الكود) وظائف أجزائه و ملاحظات هامة عنه.

نـــستخدم الشرطتين (//) لإدراج سطر تعليقات أو ملاحظات واحد, و يتميز هذا السطر في أنـــه لا يترجم أبداً, و يتم تجاهله من قبل المترجم و كأنه لم يُكتب, و عندما نريد إدراج تعليق أو ملاحظـــة يمتد إلى عدة سطور فإننا نستخدم (**/***) لبدء التعليق ثم نكتب التعليق المطلوب بعد ذلك ننهي بـــ (**/***), مثال:

```
<?
//Here is your one line comment ……
echo "<html>";
echo "<head><title>Second Page</title><head>";
echo "<body>";
```

```
/*
you can make another comment
here If it is more than 1 line….
*/
echo "<b><i>I study PHP…</i></b>";
echo "</body>";
echo "</html";
?>
```

إن PHP كغيرها من لغات البرمجة حساسة لحالة الأحرف و ما يميزها أيضاً أنها لا تهتم إلى مواضـــع كـــتابة أكوادها عندما تكون صحيحة قواعدياً, إذ أن كتابة عدة عبارات echo على ســـطر واحـــد تكافـــئ كتابة كل واحدة على سطر منفرد, فالخرج سيكون وسوم HTML و عبارات نصية و أرقام ضمن هذه الوسوم, مثال:

الكود التالي مكتوب بطريقة أنيقة:

```
<?
echo "Hello";
echo "<br>";
echo "My Friend";
?>
```

و هو يكافئ الكود التالي الذي كُتب مبعثر:

```
<?

echo "Hello";                    echo "<br>";





echo "My                                    Friend";
?>
```

و لكلاهما الخرج التالي:

```
Hello
My Friend
```

لاحظ أن الفراغات المتعددة ما بين (My) و (Friend) قد أخرجت كفراغ واحد فقط, و هذا أمر مهم جداً.

1-6: الأنماط في PHP.

تدعم اللغة PHP ثمانية أنماط (Types), مبينة في الجدول التالي:

Boolean:	المنطقي (البولياني)	**Array:**	المصفوفات

:Integer	الصحيح (دون فواصل عشرية)	**:Object**	الأغراض (الصفوف)
:Float	العشري (بفواصل عشرية)	**Resource :**	المصادر (المراجع)
:String	السلاسل النصية	**:Null**	القيمة الفارغة

و عـــادة لا يحـــدد نمط المتحول من قبل المبرمج, بل يحدد أثناء التشغيل بواسطة الـــ PHP معـــتمداً على صيغة التعبير الذي ورد فيه هذا المتحول, و لذلك فإن المتحولات قد تسلك عدة طرق مختلفة في نفس المكان من كود PHP معتمدة على النمط التي تعبر عنه.

نستخدم الرمز ($) لتعريف متحول, إذ يوضع هذا الرمز قبل اسم المتحول و لا يفصل بينهما فراغ (مثلاً: $avg) و يستخدم في شكله هذا في أي تعبير صحيح ملائم له, مثال ذلك:

```
<?
$a;
$a=4;
echo $a;
?>
```

التعليمات الثلاث السابقة تقوم بالتالي:

- تعرف متحول اسمه (a), أي تحجز مكان له في الذاكرة و تسميه (a).
- تنسب له القيمة الصحيحة (4), فيكون هنا من (Integer).
- تخرجه على الشاشة بواسطة (echo).

لاحظ الفواصل المنقوطة في نهاية كل تعليمة.

من خلال هذه التعليمات البسيطة نرى أن الرمز ($) قد رافق اسم المتحول في كافة عملياته و بدونـــه فـــإن المـــتحول لا يعود له معنى و يصبح مجرد محرف عادي (a) و يخرج محرف أيضاً, و إذا أخرجنا المتحول ($a) بدون إعطاء قيمة أولية له فإن التعليمة لن تعيد شيء.

هـــناك شـــيء مهـــم ألا و هو أن الـــ PHP حساسة للأحرف إذ يختلف المتحول ($a) عن المتحول ($A).

1-7: بعض توابع التحقق من المتحولات.

هناك مجموعة من التوابع (الدوال) التي تستخدم للتحقق من التوابع من حيث تعريفها و قيمتها, سنعرض أهم هذه التوابع:

1. الدالة **isset**.

تستخدم هذه الدالة للتأكد من وجود متحول أم لا, أي هل تم تعريف هذا المتحول أم لا, و تعيد هذه الدالة القيمة المنطقية True عندما يكون هذا المتحول قد تم تعريفه, و تعيد False في حال لم يتم تعريفه, الشكل العام لهذه الدالة هو:

```
<?
$a=8;
If ( isset($a) )                    //result is: true
      echo "a is here...";
?>
```

2. الدالة unset.

تقـــوم هـــذه الدالـــة بإلغاء متحول معرف مسبقاً من الذاكرة, و بالتالي لا يمكن استخدامه بعد استدعاء هذه الدالة, المثال التالي يوضح هذه النقطة:

```
<?
$a=4;
echo $a;                //result is: 4
unset($a);              //delete a
echo $a;                //result is: no thing
?>
```

1-8: الثوابت في PHP.

تمكننا PHP من تعريف الثوابت و التعامل مع قيمها, و ذلك باستخدام التابع Define, و حالما يتم تعريف المتحول و إعطاءه قيمة فإنها لا تتغير, يمكننا تحديد ثوابت من الأنماط الأربعة: (Boolean, Integer, Float, String).

و يمكنـــنا الوصـــول إلـــى قـــيمة الـــثابت عن طريق اسمه دون استخدام الرمز ($) الخاص بالمـــتحولات, و إذا اســـتخدمت قـــيمة ثابت دون تعريف مسبق له فإن PHP تعتبر قيمة هذا الثابت هي اسمه, مثال:

```
<?
Define("a","alone");
echo "<b>".a."</b>";                //This well give: alone
echo c;                             //This well give: c
?>
```

لاحـــظ أن التابع Define قد مرر له وسيطين, الأول هو اسم الثابت, و الثاني هو قيمة هذا الثابت, و قد أحيط كل منهما بفواصل مزدوجة, و التي يمكن الاستغناء عنها.

الفصل الثالث

معالجة الأرقام و النصوص و الوقت و التاريخ

2-1: معالجة الأرقام في PHP.

سنعرض فيما يلي المجموعة الأهم من معاملات PHP, ثم سنقدم مجموعة من توابع التعامل مع الأرقام, نبدأ بالمعاملات:

1. المعاملات الرياضية (الحسابية) في PHP.

تمتلك الـــ PHP المعاملات الحسابية نفسها التي تمتلكها أي لغة برمجة أخرى, الجدول التالي يوضح هذه المعاملات:

الصيغة	الاسم	النتيجة
$a + $b	الجمع	مجموع المتحولين a و b
$a - $b	الطرح	طرح المتحول b من المتحول a
$a * $b	الضرب	حاصل ضرب المتحول a بالمتحول b
$a / $b	القسمة	حاصل قسمة المتحول a على المتحول b
$a % $b	باقي القسمة	باقي قسمة المتحول a على المتحول b

مثال:

```
<?
$a=13;
$b=6;
//Start Operation……
echo $a+$b;                    //Result is: 19
echo $a-$b;                    //Result is: 7
```

```
echo $a*$b;                    //Result is: 78
echo $a/$b;                    //Result is: 2.1666666666667
echo $a%$b;                    //Result is: 1
?>
```

انتبه:

تعطي عملية القسمة (/) نتيجة عشرية (Float), سواء كانت القيم المدخلة (a, b) صحيحة أو عشرية.

لا تملـــك العمليات الثلاث الضرب والقسمة و باقي القسمة (% ,/ ,*) أسبقية على نفسها فهي متساوية في أولوية التنفيذ, بل إنها تملك أسبقية على عمليتي الجمع و الطرح المتساويتين في الأولوية (- ,+), مثال:

```
<?
$a=4;
$b=6;
$c=8;
//Start Operation……
echo $a*$b/$c;                //Result is: 4*6/8=3
echo $b*$a/$c;                //Result is: 6*4/8=3
echo $a+$b*$c;                //Result is: 4+6*8=52, that is: 4+48
echo $c % $b -$a;             //Result is: 8%6-4=-2, that is: 2-4
?>
```

إلا أننا نستخدم الأقواس لنحد من أسبقية هذه العمليات, المثال التالي يوضح لنا ذلك:

```
<?
$a=4;
$b=6;
$c=8;
//Start Operation……
echo ($a+$b)*$c;              //Result is: (4+6)*8=80, that is: 10*8
echo $c % ($b -$a);           //Result is: 8%(6-4)=0, that is: 8 % 2
?>
```

ففي العملية الأولى: تم حساب التعبير ($a + $b) ثم تم احتساب جداء ناتجه مع المتحول ($c), و لولا الأقواس لتم حساب التعبير ($b * $c) ثم تم جمعه مع المتحول $a.

2. معاملات الزيادة بواحد و الإنقاص بواحد في PHP.

هذه المعاملات مألوفة لنا من لغة البرمجة (C++), الجدول التالي يوضح هذه العمليات:

الصيغة	الاسم	النتيجة
$a++	جمع لاحق	يعيد a ثم يقوم بزيادتها بواحد
++$b	جمع مسبق	يزيد b بواحد ثم يعيدها
$a--	طرح لاحق	يعيد a ثم ينقصها بواحد
--$b	طرح مسبق	ينقص b بواحد ثم يعيدها

أمثلة بسيطة:

```
<?
$a=16;
$b=7-;
$c=28 - $a--;                //Result is: c=12, a=15.................
$d=++$b - (++$a);            //Result is: d=-22, a=16, b=-6......
?>
```

3. المعاملات المنطقية في PHP.

و هـــي المعاملات التي يكون طرفاها متحولات منطقية (True, False) و تعيد قيمة منطقية تبعاً لنوع المعامل و الطرفين, الجدول التالي يوضح هذه المعاملات:

الصيغة	الاسم	النتيجة
$a and $b	وَ	True إذا كان كل من a و b له القيمة True
$a or $b	أو	True إذا كان a أو b يحمل القيمة True
$a xor $b	فرق تناظري	True إذا كان أحدهما True و الآخر False حصراً
! $a	النفي	True إذا كان a=False, و False إذا كان a=True
$a && $b	وَ	True إذا كان كل من a و b له القيمة True
$a || $b	أو	True إذا كان a أو b يحمل القيمة True

و نستخدمها غالباً عندما نود أن نتحقق من شرطين سويةً ضمن إحدى بنى التحكم.

أمثلة بسيطة:

```
<?
$a=true;
$b=false;
If ($a && $b)            //Result here is: False that is: (true and false)
   echo 2;               //No Output
If (!($b))               //Result here is: True  that is: not(false)
   echo "b";             //Output is: b
?>
```

4. معاملات الإسناد في PHP.

معامـــل الإسناد الأساسي هو العملية (=), و التي من الخطأ أن نقرأها (يساوي) إذ أنها تقوم بإسناد قيمة التعبير أو المتحول الموجود في الطرف الأيمن إلى المتحول الموجود في الطرف الأيسر, مثال:

```
<?
$a=11;
$b=4;
```

```
$c=2*$a + $b - 3;
//Result: a is equal to11,.......b is equal 4,.....c is equal to 23....
?>
```

إن قيمة تعبير الإسناد هي القيمة المسندة, فمثلاً قيمة التعبير ($a=9) هي (9) لذلك يمكننا أن نكتب:

```
<?
$a=($b=7)*11;
//Result: a is equal to 77 that is: 7*11,.......b is equal to 7.....
?>
```

بالإضافة لمعامل الإسناد الأساسي (=), هناك معامل إسناد موحد لكل المعاملات ذات الطرفين الحـــسابية و النصية التي تمكننا من استخدام متحول ما في التعبير و جعل قيمة هذا المتحول تساوي قيمة التعبير, أمثلة عن ذلك:

```
<?
$a=4;
$b=2;
$c="Hello ";
//Set Operators.............
$a+=7;                    //That is: $a=$a+7.................$a=11
$b-=$a;                   //That is: $b=$b-$a................$b=-9
$c.="Friend!";            //That is: $c=$c."Friend!".........$c="Hello Friend1"
?>
```

5. معاملات المقارنة في PHP.

تمـــتلك PHP معاملات المقارنة الأساسية, و التي تعيد إما True في حال الصحة أو False في حال الخطأ, الجدول التالي يوضح هذه المعاملات:

الصيغة	الاسم	النتيجة
$a == $b	يساوي	True إذا كان a يساوي b
$a===$b	يطابق	True إذا كان a يساوي b و من نفس النمط
$a!=$b	لا يساوي	True إذا كان a لا يساوي b
$a<>$b	لا يساوي	True إذا كان a لا يساوي b
$a!==$b	لا يطابق	True إذا كان a لا يساوي b أو من نمطين مختلفين
$a<$b	أصغر من	True إذا كان a أصغر من b
$a>$b	أكبر من	True إذا كان a أكبر من b
$a<=$b	أصغر أو يساوي	True إذا كان a أصغر أو يساوي b
$a>=$b	أكبر أو يساوي	True إذا كان a أكبر أو يساوي b

أمثلة بسيطة:

```
<?
$a=12;
$b=4.6;
```

```
$c=12.0;
If ($a==$b)
     echo "Equals";                          //No Output.....

If ($a>$b)
    echo "a is larger than b";               // Result is: a is larger than b....

If ($a===$c)
   echo "a and c from the same type";     //No Output.......
?>
```

الآن سنأتي على مجموعة من التوابع المستخدمة في معالجة الأرقام:

1. الدالة **bcadd** :

تقــوم هــذه الدالة بجمع عددين مهما كان نوعهما, و تتيح لنا تحديد عدد الأرقام بعد الفاصلة العــشرية, فــإذا لم نحددها تقوم هذه الدالة بإهمالها, الشكل العام لها:**bcadd(n1, n2, s)** , بحــيث يمكــن لكل من: n1 و n2 و s عبارة عن رقم أو متحول رقمي أو سلسلة مؤلفة من أرقام, أمثلة:

```
<?
echo bcadd(30.33, 25.18, 1);              //=55.5
echo bcadd("20.13", 12, 2);               //=32.33
$a=21;      $b=8;
echo bcadd($a, $b, 1);                    //=29
?>
```

2. الدالة **bccomp** :

تقوم هذه الدالة بالمقارنة بين عددين ممرين لها, بحيث أنها تعيد 0 إذا تساوى العددين, و تعيد 1 إذا كــان الأول اكبــر, و تعــيد -1 إذا كان الأول أصغر, و تتجاهل هذه الدالة الفواصل العشرية لكلا الرقمين إذا لم نحدد عدد الأرقام المقبولة بعد الفاصلة العشرية بواسطة بارومتر ثالث, أمثلة:

```
<?
echo bccomp(2, 3);                        //=-1
echo bccomp(2.3, 2.4);                    //=0
echo bccomp(2.8, 2.5, 1);                 //=1
?>
```

3. الدالة **bcsqrt** :

تعيد هذه الدالة الجذر التربيعي للرقم الممرر, و يمكننا تحديد عدد الأرقام بعد الفاصلة العشرية إذا كان الناتج عشري, و ذلك عبر بارومتر ثاني:

```
<?
```

```
echo bcsqrt(9);                    //=3
echo bcsqrt(17);                   //=4
echo bcsqrt(17,3);                 //=4.123
?>
```

4. الدالة **abs** :

تعيد القيمة المطلقة للرقم الممرر, أمثلة:

```
<?
echo abs(-12);                     //=12
echo abs(5);                       //=5
?>
```

5. الدالة **max** :

تعـــيد القيمة الأعظم بين مجموعة قيم ممرة إليها سواء كانت أرقام أو سلاسل نصية, و تعتمد في الحساب عند دخول السلاسل النصية على ترتيب المحارف في نظام ASCII, أمثلة:

```
<?
echo max(2, 3, 8);                 //=8
echo max("abc", 1, 33);            //33
echo max("abc", "xyz");            //xyz
?>
```

6. الدالة **min** :

نفس الدالة السابقة إلا أنها تعيد القيمة الصغرى.

7. الدالة **ceil** :

تعـــيد أول رقم صحيح (دون فواصل عشرية) يلي الرقم الممرر لها (أي أكبر منه) و ذلك إذا كان عشري, أما إذا كان صحيح فتعيده نفسه, أمثلة:

```
<?
echo ceil(32.13);                  //33
echo ceil(20);                     //20
echo ceil(-21.81);                 //-21
?>
```

8. الدالة **floor** :

تعيد أول رقم صحيح (دون فواصل عشرية) قبل الرقم الممرر لها (أي أصغر منه) و ذلك إذا كان عشري, أما إذا كان صحيح فتعيده نفسه, أمثلة:

```
<?
```

```
echo floor(32.13);                    //32
echo floor(20);                       //20
echo floor(-21.81);                   //-22
?>
```

9. الدالة **log** :

تعيد لوغاريتم العدد الممرر لها, مثال:

```
<?
echo log(25.5);                       //3.238676
?>
```

10. الدالة **sqrt** :

تعيد الجذر التربيعي للرقم المرسل إليها, مثال:

```
<?
echo sqrt(16)                         //4
echo sqrt(19)                         //4.358898
?>
```

2-2: معالجة النصوص.

الـــسلاسل النـــصية هـــي مجموعة قطع من النصوص, ذلك أنها تتألف من مجموعة محارف مفردة تجمع مع بعضها البعض, و يمكن للسلسلة أن تحوي أحرف أبجدية و أرقام و علامات ترقيم و فراغات و مسافات جدولة و أي محارف أخرى, أمثلة:

```
I would like 1 bowl of soup
Is it too hot? He asked
نحن الآن نعد لتصميم موقع ويب شامل
```

2-2-1: تعريف السلاسل النصية

تقدم لنا PHP عدة طرق لتعريف السلاسل النصية, من هذه الطرق أن نحيط السلسلة النصية بفواصل علوية مفردة ('), مثال:

```
<?
echo '1/2/2006';
echo ' "Is it too hot?"  He asked';
echo 'نحن الآن نعد لتصميم موقع ويب شامل';
?>
```

تعتبر السلسلة النصية كل ما هو داخل هاتين الفاصلتين.

ملاحظة هامة جداً:

إذا أردنا أن ندرج فاصلة علوية مفردة ضمن سلسلة نصية محددة بفواصل مفردة أيضاً بحيث تكـــون هـــذه الفاصلة أحد المحارف الأساسية للسلسلة (مثل: **I'll not go**) فلا بد لنا من أن نستخدم محرف الهروب (\ ) المتوضع على يسار زر الـــ Backspace في لوحة المفاتيح, بحـــيث نقوم بوضع هذا المحرف قبل الفاصلة الواردة في السلسلة لكي يتجاهلها المترجم و لا يعتبرها فاصلة إغلاق السلسلة.

و بـــنفس الطـــريقة إذا ورد المحرف (\ ) ضمن سلسلة نصية و كان حرف أساسي فيها أي ليست وظيفته محرف هروب (مثل: I said \Welcome\ to you....) فإننا نكرره مرتين, الأول هو محرف هروب للثاني.

لدينا محرفان يجب تجاهلهما بالمحرف (\ ) عند استخدام الفواصل المفردة هما:

- الفاصلة المفردة نفسها (').
- محرف الهروب نفسه (\ ), الذي يجب أن نميزه عن عامل القسمة (/).

يمكن للسلسلة المحددة بفاصلتين مفردتين أن تمتد على أكثر من سطر, مثال:

```
<?
echo ' To make your own web site:
You must study a good PHP course ';
?>
```

الطـــريقة الأخـــرى لتعـــريف السلاسل النصية هي استخدام الفواصل المزدوجة بحيث تحيط بالسلسلة فاصلتان في أولها و فاصلتنا في آخرها, مثال:

```
<?
echo "This is a true statement";
?>
```

في هذه الحالة هناك عدة محارف يجب تجاهلها و يكون لكل منها وظيفة محددة, الجدول التالي يبين هذه المحارف:

المحرف	شرحه
\n	يعطي سطر جديد
\r	يعود لبداية السطر
\t	يعطي مسافة جدولة 8 فراغات
\\	يعطي المحرف \
\$	يعطي المحرف $
\"	يعطي المحرف "

انتبه:

الفرق الأعظم بين الفواصل المفردة و الفواصل المزدوجة التي تحدد السلاسل النصية هو أنه إذا كـــان لديـــنا متحول, و ليكن $user, فإن إدراجه ضمن سلسلة محددة بفواصل مفردة لن يعطي قيمة هذا المتحول بل سيتعامل معه كأنه محارف عادية (أي: '$user'), أما إذا أدرجناه ضمن سلسلة محددة بفواصل مزدوجة فإن المترجم سيبدله بقيمته الحقيقية, المثال التالي يوضح ذلك:

```
<?
$user="Bilal";                    //$user='Bilal';   أو
echo '<b>Welcome to you $user</b>';   //Result: Welcome to you $user
echo "<b>Welcome to you $user</b>";  //Result: Welcome to you Bilal
?>
```

2-2-2: معامل ربط السلاسل النصية

لجمع سلسلتين أو أكثر في سلسلة واحدة (لإخراجها أو معالجتها) فإننا نستخدم المعامل (**.**), و هو عامل تتابع السلاسل مع بعضها, أمثلة:

```
<?
$str='You can '.'do that';        //Result: $str='You can do that'
echo "Also you can ".'do the same with this';
//Result: Also you can do the same with this
?>
```

2-2-3: بعض توابع PHP لمعالجة السلاسل

يلزمنا عادةً معالجة لبعض السلاسل المدخلة من مصدر خارجي فيما إذا كانت مقبولة بالصيغة أو المعنـــى, فمثلاً: قد نفحص حقل البريد الالكتروني (**E-mail**) الذي قام بإدخاله مستخدم ما فيما إذا كانت صيغته صيغة صحيحة لبريد الكتروني.

يتوفر في PHP مجموعة توابع لمعالجة مثل هذه الأمور, و منها:

1. **التابع: Trim( )**

يقوم هذا التابع بحذف الفراغات الزائدة من بداية و نهاية سلسلة ممرة إليه, و ذلك عندما تكون هذه الفراغات قد أدخلت بغير قصد, مثال:

```
<?
$str1='    You can type this        ';
$str2=trim($str1);
echo "*".$str2."*";              //Result well be: *You can type this*
?>
```

2. التابع: Strlen( )

يعيد هذا التابع عدد محارف سلسلة ممرة إليه, مثال:

```
<?
$str="I am going to Syria";
$count=Strlen($str);
echo "<b>Length of: (".$str.") Is: ".$count." Characters.</b>";
 //Result: Length of: (I am going to Syria) Is: 19 Characters.
?>
```

باستخدام التابعين السابقين يمكننا أن نتحقق من أن رمز المنطقة (ZIP) الذي أدخله مستخدم لا يزيد عن (5) منازل, و ذلك كما يلي:

```
<?
$zipcode=trim($zipcode);              //Removes white spaces
$zip_length=strlen($zipcode);

If ($zip_length != 5)
{
       echo "Enter ZIP  code that is 5 characters long!!";
}
?>
```

نستطيع دمج التابعين بصيغة واحدة في تعليمة (**If**), كما يلي:

```
If  ( strlen ( trim($zip_code) ) != 5) ..........
```

3. التابع: Strcasecmp(,)

نـــستخدم هـــذا التابع عندما نريد فحص تطابق سلسلتين ممررتين له, بغض النظر عن حالة الأحـــرف صـــغيرة كانت أم كبيرة, و ذلك بمساعدة عامل المقارنة (==), إذ يعيد هذا التابع القيمة (0) عند تطابق السلسلتين, مثال:

المقارنة التالية تتطلب أن يكون المستخدم قد سجل بريده الالكتروني بأحرف صغيرة حتى يتم قبوله:

```
<?
If ($email == 'mail@website.com')
   echo "Your e-mail was accepted";
?>
```

أمـــا باستخدام التابع ( )Strcasecmp فإن البريد الالكتروني يقبل مهما كانت حالة الأحرف عند تطابقها بالقيمة, و ذلك كما يلي:

```
<?
If (strcasecmp ($email ,  'mail@website.com') == 0)
   echo "Your e-mail was accepted";
?>
```

4. التابع: substr(, ,)

تعـــيد هذه الدالة سلسلة نصية جزئية مأخوذة من السلسلة الأصل, و تحدد السلسلة الجديدة من الأصـــلية بعـــدد محارف محدد يبدأ العد للحصول عليها من موقع محدد, المثال التالي يوضح ذلك:

```
<?
$str="I am going to my school";
echo "<b>".substr($str, 5, 11)."</b>";                //= going to my
?>
```

أي يمكن تصور الشكل العام لهذه الدالة كما يلي: **substr(string, start, length)** بحيث أن: string هـــي السلسلة الأصل, start موقع المحرف الذي سنبدأ منه العد, length الطول الذي سنعد به.

ملاحظة هامة:

يبدأ ترقيم السلاسل من الـــ 0 و بالتالي في المثال السابق كان للمحرف g الرقم 5 ثم تم عد 11 محرف حتى وصلنا إلى المحرف y.

2-3: معالجة الوقت و التاريخ.

العـــرض البـــسيط للـــتاريخ و الـــوقت هو إخبار المستخدم ما هو الوقت الآن, نستخدم الدالة Date() لعـــرض الوقت و التاريخ, و تعطينا PHP إمكانيات متعددة لعرض الوقت و التاريخ بصيغ متعددة, الأمثلة التالية تبين لنا ذلك:

```
<?
echo date('d/m/y');       //result is: 01/02/06
?>
```

حيث تم عبر الدالة Date() ترجمة محارف مفردة إلى قيم الوقت المحددة بها, فقام المحرف (d) بالتعبيـــر عن اليوم (day) من الشهر (أي 01 من 28), و عبر المحرف (m) عن الشهر من السنة أي الشهر الثاني من (12) شهر, أما المحرف (y) فقد عبر عن السنة (2006).

و بالتالـــي تعبـــر هـــذه المحارف عن صيغ تاريخ و وقت تفهمها PHP, و بنفس الوقت فإن المحـــرف (/) لـــيس محـــرف يدل على وقت أو تاريخ فإن PHP تتركه كما ليجعل تنسيق التاريخ هنا (01/02/06).

الجدول التالي يعرض مجموعة الأحرف الخاصة التي تفهمها PHP لعرض تنسيقات الوقت و التاريخ:

المحرف	شرحه
H	الساعة من 00 إلى 24
h	الساعة من 01 إلى 12
A	AM أو PM
a	am أو pm
G	الساعة من 0 إلى 23
g	الساعة من 0 إلى 11
i	الدقائق 00 إلى 59
s	الثواني من 00 إلى 61
d	رقم اليوم من الشهر , من 1 إلى 31
z	رقم اليوم من السنة من 0 إلى 365
w	رقم اليوم من الأسبوع و نبدأ من 0 يوم الأحد من 0 إلى 6
l	اسم اليوم بالإنكليزية
D	اسم اليوم المختصر
F	اسم الشهر بالإنكليزية
M	اسم الشهر مختصر
m	رقم الشهر من السنة من 01 إلى 12
n	رقم الشهر من السنة من 1 إلى 12
t	طول الشهر بالأيام
y	السنة بدون قرن 06
Y	السنة بقرن 2006
O	الفارق عن غرينتش شكله HHMM -/+
T	المنطقة الزمنية

أمثلة:

```
<?
echo date('H:i:s - a');       //result is: 18:12:35 – pm
echo date('h:i:s - a');       //result is: 06:12:35 - pm
echo "It is: ".date('l')." Has the order : ".date(z)." Of  365";
//result is:( It is: Tuesday Has the order : 30 Of 365)

?>
```

الفصل الرابع

بنى التحكم

3-1: المعنى الشامل لمتحولات الصحة True و False.

كل تعبير في PHP يمتلك قيمة منطقية True أو False, تكون هذه القيمة مهمة أحياناً عندما نستخدمها في الحسابات, إذ أن:

- معظم القيم العددية قيمتها True, ما عدا (0 و 0.0) لها القيم False.
- كل السلاسل النصية لها القيمة True, ما عدا سلسلتين هما: السلسلة الفارغة و السلسلة التي تحتوي المحرف (0: صفر).
- الثابت الخاص False هو متحول False, و ما عدا ذلك فهو True.
- يأخذ التعبير قيمة True أو False بعد حسابه و الحصول على قيمته, فإذا كانت True كانت قيمته True أيضاً, و العكس بالعكس.

مثال عن ذلك:

```
<?
If (4*8)                //4*8=32 that is true
{
echo "yes";             //Result is: yes
}
//.................................................................
If (2+5-7)              //2+5-7=0 that is false
{
echo "yes";             //no result here
}
?>
```

و كما قلنا سابقاً فإن قيمة تعبير الإسناد تساوي القيمة المسندة فيه, مثال:

```
<?
If ($a=8)               //value of $a=8 is 8 that is true
{
echo "yes";             //result is: yes
}
//.................................................................
If ($b=$a-$a)           //value of $b is 0 that is false
{
echo "yes";             // no result here
}
?>
```

3-2: اتخاذ القرارات في PHP.

باستخدام بنية If() يمكن تنفيذ مجموعة عبارات في برامجنا عند تحقق شروط محددة, و بذلك تحقق لبرامجنا إمكانية تنفيذ أفعال مختلفة تبعاً للظروف الحالية, فمثلاً يمكن التحقق من أن مستخدم ما قد أدخل بيانات صحيحة للسماح له بالإطلاع على معلومات مهمة أم لا.

سنناقش كل من البنيتين If و Switch.

الشكل الأول لبنية If() هو:

```
If (condition )
{
    Statements;
}
```

و كما هو معلوم يتم تنفيذ التعليمات الموجودة ضمن جسم If() إذا كان الشرط المحدد بعد If له قيمة True, أم إذا كانت قيمته False فإن البنية تتجاهل ما جاء بداخلها, مثال:

```
<?
$a=6;
If ($a==8)
{
echo "yes";              //result is: yes
}
?>
```

الشكل الثاني لبنية If() هو:

```
If (condition )
{
    Statements1;
}
else
{
    Statements2;
}
```

يتميــز عــن الشكل الأول في أنه إذا كانت قيمة الشرط False فتنفذ التعليمات التي تأتي بعد else, مثال:

```
<?
$a=6;
If ($a==8)
      echo "yes 8";
else
      echo "no";
?>
```

الشكل الثالث لبنية If() هو:

```
If (condition1 )
{
    Statements1;
}
elseif (condition2 )
{
    Statements2;
}
```

```
.................
else
{
    Statements4;
}
```

يتضمن هذا الشكل بنية تفرعية فيها عدة شروط, تتابع فيها اختبارات الشروط و مع تحقق كل شرط تنفذ مجموعة تعليمات تابعة له, مثال:

```
<?
$a=6;
If ($a==0)
      echo 0;
elseif ($a>0)
      echo 1;
else
      echo -1;
?>
```

البنية Switch

و لها الشكل العام التالي:

```
Switch (variable)
{
Case value1:
      Statement1;
      break;
Case value2:
      Statement2;
      break;
Case value2:
      Statement3;
      break;
Default:
      Statement4;
}
```

في هذه البنية نقوم بالسؤال عن المتحول variable ففي حال كانت قيمته تساوي value1 عند ذلــك يــتم تنفيذ التعليمة statement1 و يتم الخروج من التعليمة, أما إذا كانت قيمته تساوي value2 عــند ذلك يتم تنفيذ التعليمة statement2, و هكذا حتى النهاية فإذا لم تتطابق قيمة المــتحول variable مــع أي من القيم value يتم تنفيذ التعليمة statement4 التابعة للجزء الأخير الافتراضي default, مثال:

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

```
Case 1:
    echo "Saturday";
    break;
Case 2:
    echo "Sunday";
    break;
Case 3:
    echo "Monday";
    break;
Case 4:
    echo "Tuesday";
    break;
Case 5:
    echo "Wednesday";
    break;
Case 6:
    echo " Thursday";
    break;
Default:
    Echo "Friday";
}
?>
```

3-3: الحلقات التكرارية.

يلــزمنا فــي PHP أن نكرر عملية تنفيذ تعليمة أو مجموعة تعليمات مرات متعددة محددة أو غيــر محــددة, بحــيث إذا كانت غير محددة سنجعلها تتوقف بعد عدد مراحل محددة, فمثلاً: لعرض كافة الرسائل في صفحة البريد الالكتروني نقوم بإحضار عدد الرسائل, ثم نقوم بجعل حلقة تعد من 1 إلى عدد الرسائل المحدد, في كل مرة نقوم بجلب بيانات رسالة و نعرضها, ثم الرسالة التالية, ثم التالية و هكذا...

تقدم لغة PHP مجموعة بنى تحكم لتأمين هذه الوظيفة, و هي:

1. البنية التكرارية For.

الشكل العام لهذه البنية هو:

```
For (count=start_value;  condition of end;  changing count value)
{
Statements;
}
```

تأخذ هذه الحلقة ثلاث بارومترات أساسية و هي:

- يحدد البارومتر الأول صيغة تهيئة عداد الحلقة.

- يحدد البارومتر الثاني متى ستتوقف الحلقة.
- يحدد الثالث صيغة تغيير عداد الحلقة.

مثال:

```
For ($i=1; $i<=10; $i++)
{
echo "If i=".$i." then i^2=".$i*$i;        //i^2 means  i*i
echo "<br>";
}
```

تقــوم هذه الحلقة بالعد من 1 إلى 10, في كل مرة تعد فيها هذه الحلقة قيمة لـــ i تقوم بعرض العبارة التي تبين قيمة i و قيمة i*i أي سيكون الخرج كما يلي:

```
If i=1 then i^2=1
If i=2 then i^2=4
If i=3 then i^2=9
.............................
.............................
If i=10 then i^2=100
```

إذاً استفدنا من حلقة For لتكرار عملية مرات محددة قد حددناها مسبقاً.

مثال آخر:

```
For ($i=100; $i>=1; $i--)
{
  If ($i % 2 == 0)
  {
    echo "The number: ".$i." is even";
  }
  else
  {
   echo "The number: ".$i." is odd";
  }
 echo "<br>";
}
```

هــذه الحلقة تعد من 100 نزولاً إلى 1, في كل عدة للمتحول i تفحص هذا المتحول, فإذا كان زوجي تطبع العبارة الموضحة لذلك, أما إذا كان المتحول فردي فتطبع العبارة المناسبة لذلك, أي سيكون الخرج كما يلي:

```
The number: 100 is even
The number: 99 is odd
.................................
.................................
The number: 2 is even
The number: 1 is odd
```

ملاحظة:

نـــستطيع أن نحدد عدة قيم لبدء الحلقة و ذلك في الجزء الأول من الحلقة, و بالتالي نحدد في الجـــزء التالـــي صيغ إنهاء تابعة لقيم البدء التي تم تحديدها, و في الجزء الأخير نضع صيغ تغيير القيم التي بدأنا بها, المثال التالي يوضح هذا الأمر:

```
For ($i=1,$j=2  ;  $i<=13 ;  $i++, $j+=3)
{
 Statements.......
}
```

لاحظ أن الأجزاء الثلاثة للحلقة تفصل بينها فواصل منقوطة (;), أما إذا أردنا أن نحدد جزأين فرعيين في أحد الأجزاء الثلاثة الأساسية فإننا نفصل بينها بفواصل عادية (,).

2. البنية التكرارية While.

الشكل العام لهذه البنية هو:

```
While (condition)
{
Statements;
}
```

تقـــوم هـــذه البنية بتنفيذ التعليمات (statements) مرات عدة طالما أن الشرط (condition) محقـــق, إذ لا بد أن تحتوي الحلقة على تعليمة تغير من قيمة محدد توقف الحلقة أي الشرط, المثال التالي يوضح هذه البنية:

```
<?
$a=25;
$b=10;
While ($a>=1)
{
  echo (--$a)*$b;
  echo "<br>";
}
?>
```

تقوم هذه الحلقة بحساب جداءات قيمة المتحول b بقيم متغيرة للمتحول a, إذ تبدأ قيمة a بالـــ 39 ثم ننقصها ضمن جسم الحلقة و في كل مرة نقوم بحساب حداءه مع b قبل عملية الإنقاص, سيكون الخرج بالشكل التالي:

```
240
230
.........
.........
20
10
0
```

هــذه كانــت مقدمة عن بنى التحكم في PHP, ستتوضح فكرتها أكثر مع التقدم في PHP و توظيفها في إعداد الصفحة.

الفصل الخامس

معالجة المصفوفات

4-1: تعريف بأساسيات المصفوفة.

تتألف المصفوفة من عناصر, كل عنصر يمتلك مفتاح (فهرس) و قيمة, مثلاً المصفوفة التي تحتوي معلومات حول ألوان الخضروات, تكو أسماء الخضروات هي مفاتيح و الألوان تكون القيم المقصودة, كما في الشكل:

Key	Value
Corn	yellow
Beet	red
Carrot	orange
pepper	green
orange	orange

ملاحظة (1):

لا يمكـــن لمـــصفوفة أن تحتوي عنصرين في نفس المفتاح, فحسب المثال السابق لا يمكن أن يكـــون هـــناك لـــون آخـــر للذرة غير الأصفر أي توجد الذرة corn مرة واحدة كمفتاح في المصفوفة, أما قيم العناصر فيمكن لها أن تتكرر, حيث يمتلك الجزر و البرتقال اللون البرتقالي نفسه.

ملاحظة (2):

أي سلسلة نصية أو محرف أو عدد يمكن أن يكون مفتاح في المصفوفة (مثل: 'abc',8,-7), و لا يمكن للمصفوفات أن تكون مفاتيح في المصفوفة بل يمكن أن تكون قيماً ذات مفاتيح محددة فيها, و أيضاً تأخذ قيم المصفوفة السلاسل و الأعداد و القيم المنطقية (False, True).

2-4: إنشاء المصفوفة.

لإنـــشاء المصفوفة نقوم بربط قيمة ما إلى مفتاح خاص في المصفوفة, تُحدد مفاتيح المصفوفة بأقواس مربعة [ ], مثال يبين مصفوفة مفاتيحها هي سلاسل نصية:

```
<?
$veg['corn']='yellow';
$veg['beet']='red';
$veg['carrot']='orange';
?>
```

مثال آخر عن مصفوفة مفاتيحها أرقام:

```
<?
$dinner[0]='محشي';
$dinner[1]='رز و فاصولية';
$dinner[2]='بروستد';
?>
```

و يمكننا إنشاء مصفوفة الـــ dinner نفسها بطريقة أخرى و هي كالتالي:

```
<?
$dinner=array(0 => 'محشي', 1 => 'رز و فاصولية', 2 => 'بروستد');
?>
```

ملاحظات:

- قمنا باستخدام الصيغة: **$array_name=array()**.
- ضمن الأقواس حددنا أزواج المصفوفة (المفتاح و القيمة المربوطة معه).
- نستخدم العامل (=>) لربط المفتاح مع القيمة.
- يفصل بين الأزواج فواصل عادية.

عند إنشاء مصفوفة مفاتيحها أرقام متتالية يمكننا الاستغناء عن تحديد قيم المفاتيح أثناء تعريف المـــصفوفة باستخدام الصيغة **array()**, إذ يقوم مترجم الـــ PHP بربط القيم المحددة بشكل تلقائي مع مفاتيح هي أرقام تبدأ من القيمة (0), مثال:

عند تعريفنا المصفوفة التالية:

```
<?
$arr=array(5, 6, 10, -1, 0, 8);
?>
```

يمكننا استخدام الصيغ $arr[0], $arr[1], $arr[2],.

ملاحظة هامة:

يمكننا إنشاء مصفوفة ذات مفاتيح رقمية باستخدام الصيغة:

```
<?
$arr[]='Ahmad';
$arr[]='Samer';
$arr[]='Huda';
?>
```

لاحـــظ أنـــه لم نضع مفاتيح ضمن الأقواس المربعة [], إذ أن مترجم الـــ PHP يعطيها قيماً عددية بدءً من الصفر عندما ينشئ هذه المصفوفة.

إذ عرفنا مصفوفة ذات مفاتيح رقمية بالشكل:

```
<?
$arr=array(1, 6, -13, 0, 9);
?>
```

فيمكننا إضافة عنصر إلى المصفوفة $arr و ذلك باستخدام الصيغة:

```
<?
$arr[]=16;
?>
```

ستكون الـــ 16 المضافة ذات مفتاح قيمته 5 بشكل تلقائي.

3-4: معرفة عدد عناصر مصفوفة.

يمكنـــنا الـــتابع **count()** من معرفة عدد العناصر الموجودة في المصفوفة عندما نمرر اسم متحول المصفوفة إلى هذا التابع, مثال:

```
<?
$arr=array(1, 6, -13, 0, 9);
$c=count($arr);
echo $c;                                //This will give: 5
?>
```

4-4: التنقل بين عناصر المصفوفة.

مـــن الأشـــياء المهمـــة في معالجة المصفوفات هو الوصول إلى كل عنصر من عناصرها و معالجته بشكل منفرد, نستخدم التعليمة **Foreach( )** و لذلك لمعالجة كل عنصر من عناصر المصفوفة على حدى, المثال التالي يوضح هذا التابع:

```
<?
$arr=array('a' => 'A', 'b' => 'B', 'c' => 'C');
foreach ($arr as $key => $value)
{
echo "Upper case of: ".$key." Is : ".$value;          echo "<br>";
}
?>
```

مثال آخر:

```
<?
$arr=array(1, 6, -13, 0, 9);
foreach ($arr as $k)
{
If ($arr[$k] % 2 == 0)
    echo $arr[$k]." is even";
}
?>
```

ملاحظة هامة جداً:

عندما نريد التعديل في قيم عناصر المصفوفة يجب علينا استخدام الصيغة: $arr[$k] لتعديلها و ليس $value التي تقابل الصيغة السابقة.

يمكنـــنا اســـتخدام البنـــية التكرارية **For(...)** مع المصفوفة عندما تكون ذات مفاتيح عددية صحيحة و متتالية, مثال على ذلك:

```
<?
$arr=array(1, 6, -13, 0, 9);
For ($i=0, $num=count($arr) ; $i<$num ; $i++)
{
echo $arr[$i];                echo "<br>";
}
?>
```

5-4: البحث عن عنصر في مصفوفة وفق المفتاح.

تقدم لنا PHP دالة للتأكد من وجود قيمة مفتاح ضمن مفاتيح مصفوفة ما, الصيغة العامة لهذه الدالـــة هي: **array_key_exists(key_value, array_name)**, و تعيد هذه الدالة قيمة منطقية, و تكون True في حال وجود المفتاح الذي نبحث عنه ضمن مفاتيح المصفوفة, مثال:

```
<?
$arr=array('a' => 'A', 'b' => 'B', 'c' => 'C' , 'd' =>'D');
```

```
If (  array_key_exists('d', $arr)  )
{
echo $arr['d']." is in array elements";
}
?>
```

تقــوم هــذه الدالة بالتأكد من وجود المفتاح (d) ضمن مفاتيح المصفوفة, و الكود السابق يعيد المحرف (D) على أنه موجود ضمن عناصر المصفوفة و مفتاحه هو (d).

4-6: البحث عن عنصر في مصفوفة وفق قيمة العنصر.

تقدم PHP دالة لاختبار فيما إذا كان عنصر ما متواجد في المصفوفة, الشكل العام لهذه الدالة: **in_array(element_value, array_name)** و أيضاً تعيد القيمة True في حال تواجد العنصر في المصفوفة, مثال:

```
<?
$arr=array(1, 6, -13, 0, 9);
If (in_array(7, $arr))
     echo "7 is in array elements";
?>
```

4-7: البحث عن قيمة مفتاح في مصفوفة بدلالة قيمة العنصر.

لديــنا الدالة **array_search(...)** التي تعيد قيمة المفتاح عندما يمرر لها اسم المصفوفة و قيمة العنصر الذي نريد قيمة مفتاحه, مثال:

```
<?
$arr=array('ahmad' => 22,
              'samer'  => 20,
              'huda'    => 21);
$name=array_search(20, $arr);
If ($name)
{
echo $name."'s age is 20";                // result is: samer's age is 20
}
else
{
echo "no name of 20 age";
}
?>
```

ملاحظة هامة:

يــستخدم هــذا التابع مع المصفوفات التي لا تحمل مفاتيحها إحدى القيم المنطقية False, أي إحدى القيم: (0, 0.0, '0', ' ', false).

4-8: حذف عنصر من عناصر المصفوفة.

لحذف عنصر من المصفوفة نستخدم التابع unset(), مثال:

```
<?
$arr=array(1, 6, -13, 0, 9);
unset($arr[2]);                    //-13 will be removed
?>
```

و يجب أن نميزه عن جعل قيمة المتحول (0) أو سلسلة فارغة, إذ يؤثر حذف العنصر بـ unset() على عدد عناصر المصفوفة, إن جعل إحدى العناصر (0) أو سلسلة فارغة يبقي عدد عناصر المصفوفة كما هو.

4-9: تحويل عناصر المصفوفة إلى سلسلة نصية بتنسيق محدد.

عندما نريد طباعة كافة عناصر المصفوفة بنفس الوقت, الطريقة الأسرع هي استخدام التابع implode(), الذي يقوم بتشكيل سلسلة نصية مؤلفة من عناصر المصفوفة نفسها و لكن يفصل بين كل عنصرين سلسلة تمرر إلى التابع implode(), مثال ذلك:

```
<?
$arr=array('Ahmad', 'Samer', 'Huda');
$str=implode(', ' , $arr);
echo $str;                    //Result is: Ahmad, Samer, Huda
?>
```

يمكننا دمج عناصر المصفوفة بدون فواصل عندما نستخدم السلسلة الفارغة كوسيط بين العناصر المخرجة.

4-10: إنشاء مصفوفة من سلسلة نصية منسقة بشكل محدد.

عندما تتألف سلسلة نصية من مجموعة أجزاء يفصل بين كل جزأين سلسلة جزئية محددة و ثابتة, عند ذلك يُمكننا التابع explode() المعاكس بالوظيفة للتابع implode() من إنشاء مصفوفة عناصرها مؤلفة من هذه الأجزاء, مثال:

```
<?
$str="door, table, window, desk";
$arr=explode(', ', $str);
foreach($arr as $k => $v)
{
echo $arr[$k];        echo "<br>";
}
?>
```

4-11: توابع فرز عناصر المصفوفات.

تقدم لنا PHP مجموعة من توابع الفرز التي ترتب عناصر المصفوفة بشكل تنازلي و أخرى بشكل تصاعدي, و ذلك حسب كل من المفتاح و القيمة, و هذه التوابع:

1. التابع ()sort:

يستقبل هذا التابع اسم المصفوفة و يقوم بترتيب عناصرها حسب القيم تصاعدياً, مثال:

```
<?
$arr=array(3,0,8,6,-1);
Sort($arr);
foreach($arr as $k => $v)
{
echo $k." => ".$v;        echo ", ";
}
//Result is:  0 =>-1, 1 =>0, 2 =>3, 3 =>6, 4 =>8
?>
```

2. التابع ()asort:

عـــند استخدام التابع ()sort مع مصفوفة ذات مفاتيح غير رقمية أي سلاسل نصية, فإنه يقوم بترتيب عناصرها و ينسب بعد ذلك مفتاح رقمي لكل عنصر بدلاً من السلسلة النصية, و لكن لتجـــنب هذه المشكلة و ترك العناصر مرتبطة مع مفاتيحها التي تشكل سلاسل نصية نستخدم التابع ()asort, إذ ربما يكون لارتباط العنصر مع مفتاحه هنا دلالة هامة, مثال مع ()sort:

```
<?
$arr=array('b' => 'Bilal',
           'c' => 'Chon',
           'a' => 'Ahmad');
sort($arr);
foreach($arr as $k => $v)
{
echo $k." => ".$v;        echo "<br>";
}
?>
```

سيكون الخرج من الشكل:

```
0 => Ahmad
1 => Bilal
2 => Chon
```

أم إذا استخدمنا التابع ()asort بدلاً من التابع ()sort في المثال السابق فسيكون الخرج:

```
a => Ahmad
b => Bilal
c => Chon
```

3. التابع ksort():

يستخدم هذا التابع لترتيب عناصر المصفوفة وفق قيم المفاتيح, مثال:

التابع asort(), إذ ربما يكون لارتباط العنصر مع مفتاحه هنا دلالة هامة, مثال مع sort():

```
<?
$arr=array('m' => 'door',
           's' => 'desk',
           'a' => 'zoo');
ksort($arr);
foreach($arr as $k => $v)
{
echo $k." => ".$v;      echo "<br>";
}
?>
```

سيكون الخرج بالشكل التالي:

```
a => zoo
m => door
s => desk
```

<u>ملاحظة هامة جداً:</u>

تقوم التوابع: sort(), asort(), ksort() بترتيب المصفوفة بشكل تصاعدي و لكل منها نظير بنفس الأسلوب و لكن بشكل تنازلي و هي على الترتيب: rsort(), arsort(), krsort().

الفصل السادس

التوابع في PHP

5-1: التصريح عن التوابع و استدعائها.

لإنـــشاء تابـــع جديد نستخدم الكلمة المفتاحية Function متبوعة باسم التابع الذي نريد, وفي داخل الأقواس المجعدة يأتي جسم التابع , إذاً الشكل العام للتابع هو:

```
Function Function_name()
{
Function_body;
}
```

المثال التالي يعرف تابع لعرض عبارة ترحيب:

```
<?
Function show()
{
echo "welcome to you";
}
echo "If you have new user than say to him:";
show();
echo "<br>";
echo "he will be happy";
?>
```

رأينا أنه لتنفيذ جسم تابع نقوم باستدعائه عن طريق اسمه كما مر سابقاً, يمكننا في PHP أن نـــستدعي تابـــع تـــم تعريفه لاحقاً ضمن الكود, حيث يقوم مترجم PHP بالإطلاع على كافة محتويات الصفحة قبل البدء بترجمتها.

5-2: تمرير البارومترات للتابع.

عندما يؤدي التابع الوظيفة الثابتة نفسها لا داعي لتمرير بارومترات له, أما عندما يعتمد تنفيذ جسم التابع على متحولات تأخذ قيم مختلفة فيجب جعل هذه المتحولات بارومترات تمرر إلى التابع, المثال التالي يبين أنه إذا كان المستخدم هو Ahmad فيتم عرض رسالة ترحيب له, و إلا تعرض رسالة اعتذار عن عدم القبول.

```
<?
$str1="welcome to you friend ";
$str2="sorry, you must register";
$user="Ahmad";
If ($user == "Ahmad")
   user_check($str1);
else
   user_check($str2);
Function user_check($s)
{
echo $s;
echo "<br>";
}
?>
```

ملاحظة هامة جداً:

عـــند تحديـــد بارومتـــرات في رأس التابع أثناء كتابة جسمه يجب أن لا ننقص أي منها عند اســـتدعائه, إذ ســـيؤدي إنقـــاص واحدة أو أكثر من قيم البارومترات إلى حدوث خطأ يعلنه المترجم, المثال التالي يوضح هذه النقطة:

```
<?
Function add($a, $b)
{
$c=$a + $b;
echo "If a=$a and b=$b then a+b=$c";
echo "<br>";
}

echo "you have add operation";
echo "<br>";
add(6);
?>
```

ســـيؤدي تنفـــيذ هذا الكود إلى إعلان خطأ مرده إنقاص قيمة البارومتر b, لتجنب مثل هذه الأخطاء نقوم عند التصريح عن بارومترات في رأس التابع بإعطائها قيم أولية بحيث إذا أهمل أحـــدها أثناء الاستدعاء فتؤخذ قيمته المحددة في تعريف التابع, و البارومتر الذي يعطى قيمة في الاستدعاء تهمل قيمته الأولية, المثال السابق بعد تصحيحه:

```
<?
Function add($a=3 , $b=5)
{
$c=$a + $b;
echo "If a=$a and b=$b then a+b=$c";
echo "<br>";
}
echo "you have add operation";
echo "<br>";
add(6);
?>
```

ملاحظة (1):

لا يمكـــن تحديد متحول بدلا من القيمة في إعطاء قيم أولية للبارومترات في رأس التابع, أي الصيغة التالية خاطئة: Function add($a=6, $b=$d).

ملاحظة (2):

يمكن تحديد قيم أولية لبعض البارومترات في رأس تابع و ترك بعضها الآخر بدون تحديد.

ملاحظة (3):

تتبع دلالة القيم الممررة إلى التابع ترتيب ورود البارومترات في رأسه.

5-3: تأثير التابع على متحول ممرر إليه.

عـــندما تستدعي تابع و يكون قد استخدم متحول من البرنامج الرئيسي يقوم مترجم الـــ PHP بإحداث نسخة عن هذا المتحول و يغير قيمتها حسب عمليات التابع, إذاً يبقى المتحول الأصلي محافظاً على قيمته قبل استدعاء التابع, مثال:

```
<?
Function countdown($top)
{
While ($top>0)
{
echo "$top..";
$top--;
}
echo "end.";
}

$count=6;
Countdown($count);
echo "<br>";
echo "count after that is :$count ";
?>
```

سيكون الخرج بالشكل:

```
6..5..4..3..2..1..end
Count after that is :6
```

5-4: إعادة القيمة من التابع.

تحـــسب التوابع عادةً قيمة تسمى القيمة المعادة إذ يمكننا استخدام هذه القيمة في البرنامج فيما بعد, للحصول على هذه القيمة المحسوبة في التابع نقوم بإسناد التاع المستدعى إلى متحول ما, بالشكل: $var=Fun_name($x, $y,….).

إذ يقـــوم متـــرجم الـــ PHP بحساب القيمة التي سيعيدها التابع Fun_name عندما يمرر له المتحولات ($x, $y,….) ثم تسند هذه القيمة إلى المتحول ($var).

و لإعادة القيمة المحسوبة من التابع نستخدم الكلمة المفتاحية **(return)** مع القيمة التي ستعاد, و عند تنفيذ التابع حالما يصادف عبارة return يتوقف عن التنفيذ و ينهي نفسه, مثال ذلك:

```
<?
Function circle($r=10)
{
Define(p, 3.14);
$area=p*$r*$r;
return $area;
}

$a=circle(20);
echo "If r=$r then area=$a";
?>
```

ماذا جرى؟!!

- عرفنا تابعاً يمرر له بارومتر و نصف قطر الدائرة.
- عرفنا ضمن التابع ثابت هو p و قيمته 3.14.
- حسبنا المساحة area, ثم أعدناها بـــ return.
- خـــارج الـــتابع نسبنا استدعاء التابع إلى المتحول a, و بالتالي نسبنا القيمة المحسوبة بداخله area إلى المتحول a.
- عرضنا الناتج بشكل منسق.

ملاحظة:

عندما نسبنا استدعاء التابع إلى المتحول $a أصبح هذا ذو قيمة, و بالتالي يمكننا استخدامه في البـــرنامج الرئيـــسي كـــأي متحول عادي حددت قيمته بالشكل المألوف, إذاً نستخدمه في بنى التحكم و غيرها من عمليات PHP بما يلائم قيمتها.

5-5: إعادة مصفوفة قيم من التابع.

إن العـــبارة المفـــردة return تستطيع إعادة قيمة واحدة فقط من التابع, و لا يمكن استخدامها بالشكل: return $x, $y.

عندما نريد إعادة أكثر من قيمة من التابع نلجأ إلى إعادة مصفوفة قيم, بحيث نجعل القيمة التي نريد إرجاعها هي عناصر المصفوفة و بالتالي نستطيع معالجة المصفوفة بعد استدعاء التابع كأنها مصفوفة معرفة مسبقاً في البرنامج الرئيسي, المثال التالي يوضح ذلك:

```
<?
Function operations($a=6, $b=3)
{
$add=$a + $b;
$mul=$a * $b;
$sub=$a - $b;
$div=$a / $b;
$mod=$a % $b;
return array($add, $mul, $sub, $div, $mod);
}

$a=8;      $b=3;
$arr=operations($a, $b);
echo "If a=$a and b=$b  then:";
echo "<br>";
echo "a+b=$arr[0]";
echo "<br>";
echo "a*b=$arr[1]";
echo "<br>";
echo "a-+b=$arr[2]";
echo "<br>";
echo "a/b=$arr[3]";
echo "<br>";
echo "a%b=$arr[4]";
echo "<br>";
?>
```

ملاحظة هامة:

يمكنـــنا اســـتخدام أكثر من تعليمة return في نفس التابع, و لكن المترجم سيوقف تنفيذ التابع فـــور تنفـــيذ أول تعليمة return, نقوم بإدراج أكثر من تعليمة return في نفس التابع و لكن ضمن هيكلية تحكم تتطلب ذلك, المثال التالي يوضح أنه يتوج إعادة قيمة ما عند تحقق شرط أو إعادة قيمة أخرى عند عدم تحقق هذا الشرط:

```
<?
Function tax($cost=250)
{
$price=$cost + $cost*20/100;
If ($price > 325)
   return 325;
else
  return $price;
}

$a=225;     $b=280;
$tax_a=tax($a);
$tax_b=tax($b);

echo "If a=$a then tax of a is: $tax_a";
echo "<br>";
echo "If b=$b then tax of b is: $tax_b";
?>
```

ملاحظة:

قـــد يعيد تابع ما إحدى القيمتين True أو False و ذلك ببساطة نكتب مثلاً: return true; و هي قيمة معادة نستخدمها فيما بعد في إحدى بنى التحكم.

5-6: تغيير قيمة متحول عام ضمن تابع.

عندما يمرر متحول عام إلى تابع أو يستخدم هذا المتحول ضمن التابع فإن المترجم ينشئ عنه نـــسخة و بالتالي العمليات ضمن التابع لا تؤثر على القيمة الأصلية للمتحول العام و إنما على النـــسخة المنشأة, و لكن ماذا لو أردنا تغيير هذه القيمة حسب تعليمات التابع, عند ذلك نستخدم إحدى الصيغتين:

1. **$GLOBALS['a']** بدلاً من المتحول $a في كل عملية يرد فيها.
2. ندرج السطر **global $a;** في بداية جسم التابع و نتابع استخدام $a.

و ذلـــك لـــتحديد أن العمليات داخل التابع ستغير القيمة الأصلية للمتحول $a المتواجد خارج التابع و قد استخدم داخل التابع.

مثال على ذلك:

```
<?
Function power($b=3.5)
{
global $a;
$a=$b * $b;
return $a;
}
//أو نكتب التابع بالشكل التالي
Function power($b=3.5)
{
$GLOBALS['a']=$b * $b;
return $GLOBALS['a'];
}

$a=6;
echo "a after pow is: $a";
$c=power(8);
echo "a before pow is: $a";
?>
```

الفصل السابع

الأغراض في PHP

6-1: المفهوم الغرضي في البرمجة.

ضمن مفهوم البرمجة الغرضية, يمكن للغرض أن يكون أي بند أو مفهوم تقريباً, غرض مادي كالمستخدم مثلاً.

يكـــون البـــرنامج الغرضي (الكائني المنحى) مصمماً كمجموعة كائنات مستقلة لها سمات و عملـــيات, أم الـــسمات Attributes فهي خصائص أو متغيرات تتعلق بالغرض, و العمليات Operations هـــي الطرق أو الأعمال أو الدالات التي يستطيع الغرض أن ينفذها إما لتعديل نفسه أو لإحداث بعض التأثيرات الخارجية.

هـــناك خاصة هامة للبرمجة الغرضية ألا و هي التغليف, و التي تعني إخفاء البيانات, إذ أن الوصول إلى بيانات موجودة ضمن الغرض يتوفر فقط من خلال عمليات الغرض, التي تمثل واجهة الغرض.

6-2: إنشاء الفئات و تحديد سماتها و عملياتها.

لإنشاء فئة (صف) في PHP نستخدم الكلمة المفتاحية **Class**, إذ يكون الشكل العام للفئة:

```
class class_name
{
Variables;
Operation;
}
```

و لكي تكون الفئة مفيدة يجب أن تحتوي سمات و عمليات, و لإنشاء السمات داخل الفئة ننشئ متغيرات داخل الفئة باستعمال الكلمة var, و لإنشاء العمليات ننشئ دالات داخل الفئة كما كنا نعرفها عادةً, المثال التالي ينشئ فئة:

```
<?
class circle
{
   var $r;
   var $area;
   Function calc_area($r)
   {
    return 3.14*$r*$r;
    }
}
?>
```

6-3: العمليات المشيّدة في الفئات.

تمـــتلك معظـــم الفئات نوع خاص من العمليات (التوابع) يدعى المشيّد (البنّاء), تستدعى هذه العملية عند إنشاء الكائن, و عادة ما توكل وظيفة تهيئة السمات بقيم أولية له.

عندما ننشئ المشيّد ضمن الفئة يجب أن يُسمى بنفس اسمها, و كما ذكرنا تُستدعى هذه العملية تلقائياً فلا داعي لاستدعائها يدوياً, مثال البنّاء:

```
class class_build
{
Function class_build($a)
```

```
{
Echo "constructor called with parameter $a";
}
}
```

سيتوضح البنّاء أكثر في الأمثلة القادمة.

6-4: إنشاء تواجد الفئة بعد تعريفها.

بعد التصريح عن الفئة يلزمنا إنشاء غرض (كائن) من هذه الفئة و ذلك للعمل معه, نسمي هذه العملـــية إنشاء تواجد لفئة, نستخدم الكلمة الأساسية new لإنشاء الكائن من الفئة, مثال يوضح ذلك:

```
<?
Class class_name
{
Function class_name($a)
  {
   echo "I call for $a";
  }
}

$obj1=new class_name('you');
echo "<br>";
$obj2=new class_name('me');
?>
```

سيكون الخرج:

```
I call for you
I call for me
```

إذ أن المشيّد (الباني) يُستدعى كلما أنشأنا الكائن.

6-5: وصول العمليات إلى سمات الفئة.

يُمكننا المؤشر **$this** من الوصول إلى أي سمة من سمات الفئة و ذلك ضمن إحدى العمليات لضبطها أو الحصول على قيمتها, مثال:

```
<?
Class make
{
var $a;
Function assign($aa)
 {
  $this->a=$aa;
  echo $this->a;
```

```
 }
}

?>
```

عندما نكون داخل الفئة فإن $this تدل على أن السمة a من هذه الفئة, أم عندما نكون خارج الفئة فإننا نستخدم الصيغة التالية:

```
<?
Class make
{
var $a;
}
$m= new make();
$m->a=30;
echo $m->a;
?>
```

6-6: استدعاء عمليات الفئة خارجها.

يمكنـــنا استدعاء عمليات (دالات) الفئة خارجها بعد إنشاء تواجد لها بنفس الصيغة التي تصل بها السمات, مثال ذلك:

```
<?
Class class_fun
{
Function fun1()
 {
  echo "I am one<br>";
 }

Function fun2($s)
 {
  echo "I am $s<br>";
 }
}
$c=new class_fun();
$c->fun1();
$c->fun2('two');
?>
```

6-7: إعادة استعمال الشفرة.

تقدم PHP لنا تعليمتين بسيطتين لإعادة استعمال الشفرة المكتوبة ضمن ملفات مساعدة أخرى, التعلـــيمة الأولى هي **require()** و الثانية هي **include()** و هاتان التعليمتان تعملان بشكل مشابه, إلا أنه هناك فارق بسيط ألا و هو: يتم تقييم التعليمة include() عند تنفيذها فقط, أما التعلـــيمة require() فيـــتم تقييمها في أول مرة يتم تحليلها, بمعنى آخر إذا كان تنفيذ هاتين التعليمتـــين يـــستوجب إحضار ملف نصي من مخدّم الانترنت نريد الشفرة منه, فإن استدعاء require() سيجلب الملف نفذت التعليمة أم لم تنفذ, و ذلك عند توجب تحقق شرط لتنفيذها, أم عند استدعاء include() فلا يتم جلب الملف حتى يتحقق الشرط المربوطة معه, الأمثلة التالية تبين هاتين التعليمتين:

مثال (1):

الملف page1.php يحتوي الشفرة التالية:

```
<?
echo "Welcome to another code file, you can see me any time <br>";
?>
```

الملف page2.php يحتوي الشفرة التالية:

```
<?
echo "You see what in this file, next is from another file, see it<br>";
require("page1.php");
echo "Now you returned to me….";
?>
```

مثال (2):

الملف fun.php يحتوي الشفرة التالية:

```
<?
Function gcd($a=8, $b=36)
{
  While ($a != $b)
  {
   If ($a > $b)
       $a=$a-$b;
   else
       $b=$b-$a;
  }
return $a;
}
?>
```

الملف call.php يحتوي الشفرة التالية:

```
<?
$a=28;      $b=16;
include("fun.php");                //you shall see no thing here
$c=gcd($a, $b);
echo "The GCD of $a and $b is: $c";
?>
```

مثال (3):

الملف code1.php يحتوي الشفرة التالية:

```
<?
echo "Welcome to you friend Ahmad";
?>
```

الملف code2.php يحتوي الشفرة التالية:

```
<?
echo "Sorry, you must register";
?>
```

الملف user1.php يحتوي الشفرة التالية:

```
<?
$user="Ahmad";
If ($user="Ahmad")
   require("code1.php");
else
   require("code2.php");
?>
```

سيؤدي تنفيذ هذه الشفرة إلى إحضار الملفين code1.php و code2.php مهما كان الشرط, أم عندما يحتوي الملف user2.php الشفرة التالية:

```
<?
$user="Ahmad";
If ($user="Ahmad")
   include("code1.php");
else
   include("code2.php");
?>
```

فسيتم إحضار الملف الذي يتحقق الشرط المرفق بتنفيذه, أي لا داعي لإحضار الآخر.

الفصل الثامن

إدارة الكعكات و الجلسات

7-1: ما هي الكعكات والجلسات؟ و لماذا نستخدمها؟

لا يملك البروتوكول HTTP طريقة لملاحقة المستخدمين أو الحفاظ على المتحولات مع تنقل المستخدم بين صفحات الموقع, إلا أنه يمكننا باستخدام اللغة PHP التغلب على هذه الاستقلالية علـــى حالة الويب هذه, إذ يتوفر لدينا عدة خيارات لذلك, و الأكثر انتشاراً هو: ملفات تعريف الارتباط المدعوة (الكعكات: Cookies) و جلسات العمل (Sessions).

الكعكـــات: هي طريقة يستخدمها الملقم لتخزين معلومات عن كمبيوتر المستخدم و ذلك بشكل محـــدد سنـــشرحه لاحقـــاً, و بذلك يمكن للملقم تذكر هذا المستخدم على مر زيارته, مثلاً: إذا أعطـــيت الملقـــم اســـمك و أي علامـــة يمكن للملقم أن يتذكر اسمك فيما بعد عن طريق هذه العلامة.

الجلـــسات: تـــسمح لنا بتخزين البيانات على الملقم و ليس على مستعرض الويب, و يستخدم معرّف جلسة العمل لتحديد سجل مستخدم محدّد.

7-2: تعريف الكعكات برمجياً.

يتم إرسال الكعكات عن طريق التابع **setcookie(name, value)**, و ذلك كما في الشكل:

```
<?
setcookie('user', 'Ahmad');
?>
```

الآن:

- ندعو الـــ: user اسم الكعكة.
- ندعو الـــ: Ahmad قيمة الكعكة.

سيرســـل هـــذا السطر البسيط كعكة إلى المستعرض بالاسم user و القيمة Ahmad, أمثلة أخرى عن إرسال الكعكات:

```
<?
setcookie('ID', '123');
setcookie('mail', 'your_mail@mywebsite.com');
?>
```

ملاحظة:

- عند تسمية الكعكات لا نستخدم الفراغات أو علامات الترقيم ضمن الأسماء.
- يجب الانتباه إلى حالة الأحرف.

7-3: كيفية الوصول إلى الكعكة فيما بعد.

عندما تكون الكعكة قد عرفت بالشكل التالي:

```
<?
setcookie('user', 'Ahmad');
?>
```

فـــإن الوصـــول لقيمة الكعكة سيكون بشكل مشابه للوصول إلى قيمة خلية في المصفوفة عن طريق مفتاح هذه الخلية:

```
<?
echo "Welcome to you $_COOKIE['user']";
// Result is: Welcome to you Ahmad
?>
```

انتبه:

يجب كتابة $_COOKIE[ ] بأحرف كبيرة حصراً لكي تعمل.

7-4: تحديد عمر الكعكة.

نستخدم وسيط ثالث ضمن تعريف الكعكة لتحديد عمرها, سيكون هذا الوسيط هو زمن بالثواني للفترة التي تبقى بها الكعكة نشيطة, نستخدم هنا التابع **time()** و نضيف له عدد الثواني التي نـــريد لكي تعرف PHP لأي فترة ستبقى الكعكة تعمل, إذاً سيصبح شكل تعريف الكعكة من جديد كما يلي:

```
<?
setcookie('user', 'Ahmad', time()+3600);
//That means: your cookies will be alive 3600 sec, that is one hour.
?>
```

7-5: حذف الكعكة.

يلــزمنا عــند إجراء عملية تسجيل الخروج من قبل مستخدم ما أن نحذف الكعكة التي كانت تتعرف على هذا المستخدم في الموقع, لحذف كعكة نستخدم الصيغة:

```
<?
setcookie('user', '');
?>
```

و الذي يعني حذف القيمة Ahmad من الكعكة user و بالتالي إلغاء فعالية الكعكة.

سنقوم الآن بتطبيق مثال متكامل عن الكعكات:

(1): الصفحة **register.html** و هي كالتالي:

```
<html>
<head>
  <title></title>
</head>
<body>
<form method="post" action="register.php" >
Full name:  <input name="name" type="text" value=""><br>
User name: <input name="user" type="text" value=""><br>
password:  <input name="pass" type="password" value=""><br>
<input type="submit" value="Register">
</form>
</body>
</html>
```

و هــوي صفحة لتسجيل بيانات عن المستخدم: اسمه الكامل و اسم مستخدم و كلمة مرور, و التي سبق و أن ناقشنا كيف تصدر متحولات إلى PHP.

(2): الصفحة **register.php** و هي كالتالي:

```
<?
setcookie($user, $name, time()+3600);
echo "<center>Welcome to you: ".$_COOKIE[$user]."</center>";
?>
```

تقــوم هــذه الــصفحة باســتقبال المتحولات: $name, $user, $pass من الصفحة السابقة register.html, و يقــوم بإنــشاء كعكة تحمل اسم المستخدم نفسه, و لها قيمة تساوي الاسم الكامل لهذا المستخدم, ثم يرحب بهذا المستخدم عن طريق قيمة الكعكة نفسها.

نستدعي الصفحتين register.html و register.php ثم نغلق المستعرض قليلاً.

(3): الصفحة **login.html** و هي كالتالي:

```
<html>
<head>
<title></title>
</head>
<body>
<form action="login.php" method="post">
User name: <input name="user" type="text" value=""><br>
Password: <input name="user" type="password" value=""><br>
<input type="submit" value="Login">
</form>
</body>
</html>
```

تقــوم هــذه الصفحة بتسجيل الدخول للمستخدم باسم المستخدم و كلمة المرور فقط, و ترسل البيانات المدخلة إلى الصفحة **login.php** التالية:

(4): الصفحة **login.php** و هي كالتالي:

```
<?
echo "<center>Welcome to you: ".$_COOKIE[$user]." again.</center>";
?>
```

تقوم هذه الصفحة باستقبال المتحولين $user و $pass, و من ثم تحصل على قيمة الكعكة و ذلك عن طريق اسمها الذي هو اسم المستخدم نفسه.

و بعد قليل نستدعي الصفحتين login.html و login.php لنلاحظ أن الصفحة الثانية تعرفت على المستخدم باستخدام الكعكة.

7-6: البدء بجلسة عمل و إضافة متحولات جلسة.

من المهم في إدارة جلسات العمل session أنه يجب في كل صفحة نريد استخدام الجلسة فيها أن تــبدأ بالــتابع **session_start()** و الذي يطلب من PHP إما بدء جلسة عمل جديدة أو الوصول إلى جلسة عمل تعمل الآن, الكود البسيط التالي يبدأ جلسة عمل:

```
<?
session_start();
?>
```

الآن و بعــد بــدء جلسة عمل جديدة لا بد من إضافة متحولات جلسة لكي تكون هذه الجلسة مفيدة في موقعنا, هنالك شكلين لإضافة متحولات جلسة هما:

- الشكل (1): **session_register('variable')**.
- الشكل (2): **$_SESSION[variable']=value**.

المثال التالي يوضح ذلك:

```
<?
session_start();
$id=1982;
```

```
session_register('id');
$_SESSION['price']=0;
?>
```

في هذا المثال:

1. تم بدء جلسة عمل.
2. عرفنا متحول id و أعطيناه القيمة 1982.
3. جعلنا المتحول id متحول جلسة و ذلك بتسجيله بالشكل الأول.
4. جعلنا المتحول price متحول جلسة و ذلك بتسجيله بالشكل الثاني و جعلنا قيمته (0).

7-7: الوصول إلى متحولات الجلسة.

إذاً لا بـــد مـــن وضـــع التعليمة: session_start() في بداية الصفحة التالية التي ستستخدم متحولات هذه الجلسة المنشأة, و للوصول لمتحول جلسة نستخدم الصيغة التالية التي مرت في المثال السابق:

```
<?
session_start();
$_SESSION['price']=$_SESSION['price']+125;
?>
```

حيث أن المتحول $price قد سجل سابقاً في هذه الجلسة.

7-8: حذف متحول من الجلسة.

يلـــزمنا عـــادةً أن نحذف متحول جلسة لم يعد له لزوم, و لهذا الأمر نستخدم التابع **unset()** الـــذي يـــستخدم لحـــذف متحولات من PHP بشكل عام, ستكون صيغة حذف متحول جلسة بالشكل:

```
<?
session_start();
echo "Last Price is: ".$_SESSION['price']." SP.";
unset($_SESSION['price']);                    //This well remove price
?>
```

7-9: إنهاء جلسة عمل.

بعـــد معالجـــة صـــفحات متعددة في موقع ما قد يكون من المهم إنهاء جلسة العمل التي بدأها مستخدم في هذا الموقع, نستخدم التابع: **session_destroy()**, سيكون المثال التالي مع إنهاء الجلسة بالشكل:

```
<?
session_start();
echo "Last Price is: ".$_SESSION['price']." SP.";
unset($_SESSION['price']);                    //This will remove price
session_destroy();                            //This will destroy your session
```

```
?>
```

المثال المتكامل التالي يوضح مفهوم و وظيفة الجلسة بشكل واضح:

الصفحة (1): **login.html** و هي بالشكل:

```
<html>
<head>
<title></title>
</head>
<body>
<form method="post" action="login.php" >
User name: <input name="user" type="text" value=""><br>
First Price: <input name="price" type="text" value=""><br>
<input type="submit" value="calculate">
</form>
</body>
</html>
```

تشكل هذه الصفحة نموذجاً لتسجيل اسم المستخدم و السعر الذي يفرضه هو لأول مرة, و تقوم هذه الصفحة بإرسال متحولين يمثلان المدخلات $user و $price إلى الصفحة التالية.

الصفحة (2): **login.php** و هي بالشكل:

```
<?
session_start();
session_register('user');
session_register('price');
echo "<center>Welcome To You ".$_SESSION['user']."</center><br>";
echo "<center>Your First Price= ".$_SESSION['price']."</center><br>";
?>
```

تقوم هذه الصفحة بما يلي:

- بدء جلسة عمل جديدة (إذ لم يكن هناك جلسة عمل قد بدأت).
- تسجيل المتحولين $user و $price كمتحولين لهذه الجلسة.
- الترحيب بالمستخدم و إظهار السعر الذي أدخله, و ذلك باستخدام متحولات الجلسة.

الصفحة(3): **inc_price.php** و هي بالشكل:

```
<?
session_start();
$_SESSION['price']+=135;
echo "<center>Welcome To You ".$_SESSION['user']." again</center>";
echo "<br>";
```

```
echo "<center>Your new Price= ".$_SESSION['price']."</center><br>";
session_destroy();
?>
```

ماذا جرى في هذه الصفحة؟!!

1. وضـــعنا التعليمة session_start() و ذلك لتهيئة الجلسة التي قد بدأت سابقاً في هذه الصفحة.
2. وصلنا إلى متحول الجلسة price و أضفنا له القيمة 135.
3. قمنا بعرض اسم المستخدم مرة ثانية.
4. و عرضنا السعر الجديد بعد الإضافة.
5. عندما انتهينا من الجلسة قمنا بإنهائها بـــ destroy و بالتالي ستفقد كافة متحولاتها.

الفصل التاسع

معالجة قواعد البيانات MySQL باستخدام PHP

8-1: مقدمة إلى MySQL.

إن MySQL كغيرها من أنظمة قواعد البيانات تؤمن للمستخدم تعليمات لإنشاء قاعدة البيانات و بعد ذلك إنشاء جداول هذه القاعدة و ذلك بعد تحليل نظام الموقع بشكل صحيح و أمثلي بما يخص توزيع الحقول على الجداول و اختيار المفاتيح الرئيسية و المفاتيح الثانوية و ذلك لتأمين ربـــط الجـــداول بصيغة مفيدة معبرة عن علاقة حقيقية بين الحقول, كما و تؤمن MySQL تعلـــيمات هامـــة جداً من أجل إجراء عمليات على بيانات الجداول مثل: الإدخال و التعديل و الحـــذف, و أيضاً الحصول على البيانات من هذه الجداول بالصيغة المطلوبة و ذلك باستخدام تعليمات الاختيار المعروفة عبر أنظمة قواعد البيانات الأخرى.

تمكنـــنا MySQL أيضاً بعد إنشاء الجداول من تعديل بنية هذه الجداول و تعديل محتواها من بـــيانات, و بمـــا أننا نقوم ببناء المواقع باستخدام المخدّم AppServ فإن هذا المخدم يقدم لنا تطبيق جاهز يمكننا من بناء أي قاعدة بيانات بما فيها من جداول و عمليات على هذه الجداول.

8-2: أنماط البيانات في MySQL.

تـــدعم الـــــ MySQL مجموعة من أنماط البيانات التي يمكننا أن نحدد حقول الجداول منها أثناء بنائها, و أهم هذه الأنماط مبينة في الجدول التالي:

النمط	الحجم	الوصف
Char(L)	L بايت	يحتوي من 0 حتى 3 محرف
Varchar(L)	L+1 بايت	يحتوي من 0 حتى 255 محرف
Int(L)	L*4 بايت	رقم صحيح بـ L خانة
Float(L)	L*4 بايت	رقم عشري بـ L خانة
Double(L)	L*8 بايت	رقم عشري دقة مضاعفة بـ L خانة
Date	3 بايت	التنسيق YYYY-MM-DD
DateTime	8 بايت	التنسيق YYYY-MM-DD HH:MM:SS
Time	3 بايت	التنسيق HH:MM:SS

سنقوم الآن بإنشاء قاعدة بيانات نموذجية بواسطة الـــ:

phpMyAdmin Database Manager و هـــو عـــبارة عـــن تطبـــيق ويب نصل إليه بالعـــنوان: (http://localhost/phpmyadmin), ستظهر لنا الصفحة التي منها نبدأ بإنشاء قاعدة البيانات.

8-3: مثال عن قاعدة بيانات موقع نتائج امتحانات طلاب.

سنسمي قاعدة البيانات بـــ **examdb**, لبناء قاعدة البيانات تقوم بالخطوات التالي:

أولاً: تحليل نظام الامتحانات

يتميز كل طالب في المعهد برقم جامعي يختلف عن الأرقام الأخرى لزملائه, و بالتالي يمكننا اعتبار رقم الطالب الجامعي مفتاح أساسي في جدول الطالب, و بالإضافة لهذا المفتاح يحتوي جدول الطالب على الاسم الثلاثي للطالب و السنة و تاريخ الميلاد, وبالتالي ستكون بنية جدول الطالب كما يلي:

جدول الطالب **student**	
الحقل	نمطه
st_num (PK)	Int(5)
st_name	Varchar(30)
year	Int(1)
birth	Date

أم فـــي جدول المواد فسنقوم بإعطاء رقم مميز لكل مادة بحيث لا تشترك مادتان بنفس الرقم (أي أنـــه المفتاح الأساسي في هذا الجدول), بالإضافة له سيكون لدينا حقل اسم المادة فقط, و بالتالي سيكون لجدول المواد البنية التالية:

جدول المواد **course**	
الحقل	نمطه
co_num (PK)	Int(2)
co_name	Varchar(20)

الآن ســنقوم بإعــداد جدول ربط بين هذين الجدولين لحفظ درجات الطلاب في المواد بحيث يحتوي جدول الربط على حقل رقم الطالب و حقل رقم المادة و حقلي درجة الجزء العملي و درجـــة الجزء النظري لهذا الطالب في هذه المادة, و بالتالي سيتولد لدينا مفتاح أساسي مكون مـــن حقلي رقم الطالب و رقم المادة و بالتالي لن يكون لطالب ما درجتين في نفس المادة, إذاً بنية هذا الجدول هي:

جدول الدرجات **degree**	
الحقل	نمطه
st_num	Int(5)
co_num	Int(2)
t_degree	Int(2)
a_degree	Int(2)

ثانياً: بناء قاعدة البيانات و الجداول

نتبع الخطوات التالية لبناء قاعدة البيانات:

1. ندخل إلى البرنامج phpMyAdmin و ذلك عبر مستعرض الويب بالعنوان التالي: (http://localhost/phpmyadmin) لتظهـــر لنا صفحة فيها الخيار: ***تكـــوين قاعدة***

بيانات جديدة مرفق بحقل لكتابة اسم قاعدة البيانات و زر ***تكوين*** القاعدة, نقوم بكتابة اسم القاعدة (examdb) ثم نضغط زر (تكوين).

2. ننتقل إلى الصفحة التالية التي أنشأت بها قاعدة البيانات و قد ظهر اسمها على الجزء اليساري من الصفحة, و في جزئها اليميني لدينا الخيار: ***تكوين جدول جديد في قاعدة البيانات examdb***, مع توفير حقل لكتابة اسم الجدول و حقل آخر لتحديد عدد حقوله و زر ***تنفيذ*** لإنشاء الجدول, فلإنشاء جدول الطالب نقوم بكتابة student في حقل اسم الجدول و نكتب (4) في حقل عدد حقول الجدول ثم نضغط زر تنفيذ لننتقل إلى الصفحة التالية.
3. بعد ذلك تظهر لنا عناصر نموذج لتحديد حقول الجدول, سيكون عدد أسطر هذه العناصر بعدد حقول الجدول, و لكل حقل تخصص العناصر التالية:
   - عنصر الحقل: و نكتب فيه اسم الحقل.
   - عنصر النوع: و نحدد فيه نمط الحقل (Int, Varchar,...).
   - عنصر الطول (الحجم): و نحدد حجم الحقل حسبما يكون رقم أو نص.
   - عنصر الخالي: و نحدد فيما إذا كان هذا الحقل يقبل قيم فارغة أم لا.
   - عنصر الافتراضي: و نحدد فيه قيمة افتراضية للحقل.
   - عنصر الأساسي: و نحدد فيما إذا كان الحقل مفتاح أساسي.

نقوم بتعبئة هذه العناصر من أجل حقول الجدول student كما مرت بنيته سابقاً, ثم نضغط على الزر ***حفظ*** لننتقل إلى الصفحة التالية التي تبين بنية هذا الجدول, و لبناء جدول آخر نقوم بالضغط على اسم قاعدة البيانات في الجزء اليساري من الصفحة لنعود من جديد إلى صفحة ***تكوين جدول جديد في قاعدة البيانات examdb*** لننشئ الجدولين **course** و **degree** بنفس الطريقة.

سيُشكل المجلد examdb في الدليل التالي بالشكل: (c:\appserv\mysql\data\examdb), أي أن اسم مجلد قاعدة البيانات هو نفسه اسم قاعدة البيانات الذي حددناه أثناء بنائها, و توضع ضمن هذا المجلد ملفات قاعدة البيانات, بحيث يشكل لكل جدول ثلاثة ملفات.

ثالثاً: ربط قاعدة البيانات مع صفحة PHP و التحكم بها

سنقوم الآن ببناء مجموعة صفحات تؤدي مهام متعددة في مجال نشر درجات الطلاب على هذه الصفحات, و سنبدأ بالعملية الأولى:

1. عملية الإضافة.

تحتاج عملية الإضافة إلى صفحتين الأولى هي صفحة تسجيل بيان الطالب من رقم جامعي و اسم و تاريخ ميلاد و سنة و هي صفحة HTML, أما الصفحة الثانية فهي صفحة PHP تقوم باســتقبال مــتحولات الــصفحة الأولى و حشرها في قاعدة البيانات, فيما يلي سنرد كود كل صفحة مع الشرح.

الصفحة (1): **add_st.html** و هي:

```
<html>
<head>
  <title></title>
</head>
<body>
<form action="add_st.php" method="post">
student name: <input name="st_name" type="text" value=""><br>
univ number: <input name="st_num" type="text" value=""><br>
Year of univ: <select size="1" name="year">
      <option value="1">1</option>
      <option value="2">2</option>
      </select><br>
birth date: <input name="birth" type="text" value="">
<input type="submit" value="Add Student">
</form>
</body>
</html>
```

و تحــتوي على نموذج لإرسال المعلومات يحتوي الحقول: اسم الطالب st_name, و الرقم الجامعــي st_num, و السنة الدراسية year, و عام الميلاد birth, بالإضافة إلى زر إرسال هذه المعلومات إلى الصفحة add_st.php.

الصفحة (2): **add_st.php** و هي:

```
 1: <?
 2: @ $db=mysql_connect("localhost","root","");
 3: if (!db)
 4: {
 5: echo "Cannot Connect";
 6: exit;
 7: }
 8: mysql_select_db("examdb");
 9: $query="insert into student(st_num, st_name, year, birth)
```

```
10: values('".$st_num."', '".$st_name."', '".$year."', '".$birth."')";
11: $result=mysql_query($query);
12: if ($result)
13: echo "<center>تمت إضافة: ".mysql_affected_rows()." طالب</center>";
14: ?>
```

ماذا جرى في هذه الصفحة؟!!

فـــي السطر رقم (2) استدعينا التابع **mysql_connect()** الذي يتصل مع المخدم للوصول إلى قاعدة البيانات و نسبنا قيمته إلى المتحول db و هو متحول منطقي و سيحمل قيمة true فـــي حـــال نجاح الاتصال مع المخدم, أم إذا فشل الاتصال فسيحمل القيمة false, لاحظ أننا اســـتخدمنا الرمز (@) هو معامل منع الخطأ و الذي دوره منع استمرار الخطأ عند حدوثه و امتصاص هذا الخطأ.

في السطر رقم (3) قمنا باختبار الاتصال مع المخدم, و عند الفشل سنعرض الرسالة التي تبين أنه لا يمكن الاتصال الآن: (cannot connect) و نخرج من الصفحة أي لا نتم ترجمتها, و ذلك عبر الأسطر (4, 5, 6, 7).

في السطر رقم (4) استخدمنا التابع **mysql_select_db()** و مررنا له الوسيط: اسم قاعدة البيانات, و ذلك للاتصال بقاعدة بيانات الموقع (examdb).

فـــي السطر رقم (9) قمنا بإعداد استعلام إضافة (حشر أو إدخال) المتحولات التي جاءت من الصفحة add_st.html في الجدول student, و ذلك باستخدام الصيغة insert into.

فـــي الـــسطر رقم (11) أنشأنا المتحول result و الذي يحمل النتيجة المنطقية لنجاح أو فشل عملية الإضافة إذ نمرر له وسيط هو اسم الاستعلام الذي نريد السؤال عنه.

فـــي السطر رقم (12) اختبرنا صحة عملية الإضافة, و عند الإيجاب تم استدعاء التابع الذي يـــرجع لـــنا عدد الأسطر التي تمت إضافتها و هو: mysql_affected_rows() حيث قمنا بإعلانها.

2. عملية التعديل.

ســـنقوم الآن بتطبـــيق مـــثال حول تعديل بيان طالب بتغيير اسمه عند تمرير رقمه الجامعي كبارومتـــر, الصفحة الأولى هي صفحة **up_st.html** و التي سنحدد ضمنها رقم الطالب و اسمه الجديد, و الصفحة الثانية هي الصفحة **up_st.php** التي تستقبل المتحولين الدالين على رقم و اسم الطالب.

الصفحة (1): **up_st.html** و هي:

```
<html>
```

```
<head>
  <title></title>
</head>
<body>
<form action="up_st.php" method="post">
univ number: <input name="st_num" type="text" value=""><br>
New student name: <input name="st_name" type="text" value=""><br>
<input type="submit" value="Update Student">
</form>
</body>
</html>
```

الصفحة (2): **up_st.php** و هي:

```
  1: <?
  2: @ $db=mysql_connect("localhost", "root","");
  3: if (!db)
  4: {
  5: echo "Cannot Connect";
  6: exit;
  7: }
  8: mysql_select_db("examdb");
  9: $query="update student set st_name='".$st_name."'
 10: where st_num='".$st_num."'";
 11: $result=mysql_query($query);
 12: if ($result)
 13: echo "<center>تمت تعديل: ".mysql_affected_rows()." طالب</center>";
 14: ?>
```

3. عملية الحذف.

ســـنقوم ببـــناء تطبيق يقوم بحذف بيان طالب عند تمرير رقمه الجامعي, الصفحة الأولى هي صفحة **del_st.html** ستقوم بإرسال متحول يحمل الرقم الجامعي إلى الصفحة **del_st.php** و التي تجري عملية الحذف.

الصفحة (1): **del_st.html** و هي:

```
<html>
<head>
  <title></title>
</head>
```

```
<body>
<form action="del_st.php" method="post">
univ number: <input name="st_num" type="text" value=""><br>
<input type="submit" value="Del Student">
</form>
</body>
</html>
```

الصفحة (2): **del_st.php** و هي:

```
 1: <?
 2: @ $db=mysql_connect("localhost", "root","");
 3: if (!db)
 4: {
 5: echo "Cannot Connect";
 6: exit;
 7: }
 8: mysql_select_db("examdb");
 9: $query="delete from student  where st_num='".$st_num."'";
11: $result=mysql_query($query);
12: if ($result)
13: echo "<center>تمت حذف: ".mysql_affected_rows()." طالب</center>";
14: ?>
```

4. عملية استرجاع البيانات (الاختيار).

سنقوم الآن بتصميم صفحتين في الأولى رابط للصفحة الثانية التي تعطي أسماء الطلاب.

(1): **exam.html** و هي:

```
<html>
<head>
<title></title>
</head>
<body>
<center><a href="student.php">استعراض الطلاب</a></center><br>
</body>
</html>
```

الصفحة (2): **student.php** و هي:

```
 1: <?
```

```
 2: @ $db=mysql_connect("localhost", "root","");
 3: if (!db)
 4: {
 5: echo "Cannot Connect";
 6: exit;
 7: }
 8: mysql_select_db("examdb");
 9: $query="select * from student";
10: $result=mysql_query($query);
11: $num=mysql_num_rows($result);
12: if (($result) && ($num>0))
13: {
14:   echo "<p align=center dir=rtl>";
15:   for ($i=0; $i<$num; $i++)
16:   {
17:       $row=mysql_fetch_array($result);
18:       echo $row["st_num"];          echo " --- ";
19:       echo $row["st_name"];         echo " --- ";
20:       echo $row["year"];            echo " --- ";
21:       echo $row["birth"];           echo " --- ";
22:       echo "<br>";
23:   }
24:   echo "</p>";
25: }
26: ?>
```

ماذا جرى في هذه الصفحة!!

ما هو جديد في هذه الصفحة يبدأ بالسطر (11):

لدينا التابع **mysql_num_rows()** و الذي يعيد عدد السجلات التي أرجعها الاستعلام, ترى أننا نضع هذا العدد ضمن المتحول num.

في السطر (12): تم التحقق من صحة الاستعلام و أن هناك سجلات مرجعة.

في السطر (15): نبدأ حلقة For و التي تمر على كافة السجلات المعادة.

في السطر (17): نشكل مصفوفة اسمها row مفاتيحها هي أسماء الحقول و قيمها المقابلة هي قـــيم هذه الحقول, و التي نحصل عليها بالصيغة: $row=mysql_fetch_array($result), إذ مررنا لهذا التابع اسم الاستعلام.

في السطور (18 إلى 21): تم استخراج قيم المصفوفة عبر مفاتيحها.

ملاحظة هامة جداً:

توفر حلقة ***For*** عند كل عدة انتقال تلقائي للسجل التالي لتعرض قيمه.