



جامعة طنطا
كلية التربية النوعية
قسم تكنولوجيا التعليم
المستوى الرابع (ساعات معتمدة)
برنامج إعداد معلم حاسب آلي

نظم برمجة التعليم القائم على لويب

إعداد

أ.م.د أمل حمادة

د / أميرة إبراهيم عبد الغني



مقدمة

الحمد لله والصلاة والسلام على المبعوث رحمة للعالمين سيد الخلق أجمعين سيدنا محمد صلى الله عليه وسلم. عزيزي الطالب عزيزتي الطالبة اننا نحمد الله تعالى أن أعاننا ووفقنا لاتمام هذا العمل الذي نرجو من الله أن يجعله في ميزان حسناتنا يوم اللقاء . فحرصاً منا على تقديم كل ما هو جديد ومفيد، بذلنا الجهد لنقدم اليكم شرحاً مبسطاً عن لغة PHP وكيفية التعامل معها، لغة PHP هي الأكثر طلباً اذا كنت تبحث عن وظيفة كمصمم ويب أو محرر لأوامر الويب. وفي كتابنا هذا نحاول النهوض بك عزيزي الطالب عزيزتي الطالبة، ولدينا أمل أن يكون تعلم هذه اللغة أسهل مما تعتقدون.

. أرجو أن يكون هذا الشرح المبسط فيه افادة بالنسبة لك بحيث يضعك على بداية الط ك به لن تصبح مبرمجاً محترفاً لابد أن تبذل الجهد لكي تصل.

الكتاب عبارة عن احدى عشر باباً ، كل باب مُقسَّم الى فصول وذلك بهدف التبسيط وسهولة الفهم .

نرجو أن تحققوا أقصى استفادة منه. ونحن ندعو الله دائماً أن يوفقكم ويهديكم إلى ما فيه الرشاد.

نتمنى لكم دوام التوفيق،

أ.م.د/ أمل حمادة.

د/ أميرة إبراهيم عبد الغني.

رسالة برنامج تكنولوجيا التعليم

يلتزم برنامج تكنولوجيا التعليم بتوفير بيئة تربوية تراعي الفروق الفردية لإعداد أخصائي تكنولوجيا التعليم متميز علمياً ومهنياً وفنياً مواكبا متطلبات سوق العمل التكنولوجي وقادر على الإسهام في تطوير مجال تكنولوجيا التعليم والمنافسة البحثية وخدمة المجتمع لتحقيق أهداف التنمية المستدامة.

وتنص أهداف برنامج تكنولوجيا التعليم على :

- ١- الإرتقاء بجودة أداء الكوادر البشرية من أخصائى تكنولوجيا التعليم للعمل فى المؤسسات التعليمية ومراكز التعليم الإلكتروني فى مجال تكنولوجيا التعليم.
- ٢- رفع كفاءة المنظومة التعليمية لزيادة القدرة التنافسية ومواكبة المستجدات ذات العلاقة بالتخصص، من خلال تحديث وتطوير البرامج التعليمية وأساليب وأدوات التعليم فى مؤسسات التعليم الجامعى وما قبل الجامعى.
- ٣- الريادة فى البحث العلمى والتميز والإبتكار فى مجالات تكنولوجيا التعليم.
- ٤- تفعيل الشراكات المجتمعية فى ضوء أهداف التنمية المستدامة من خلال رفع وعى الطلاب فى المشاركة فى أنشطة خدمة المجتمع، والتطوير التكنولوجى.
- ٥- وضع آلية للتحسين المستمر فى جميع عناصر العملية التعليمية والبحثية لتدويل برنامج تكنولوجيا التعليم.

قائمة المحتويات

الموضوع	رقم الصفحة
الباب الأول: مقدمة عن لغات البرمجة وقواعد البيانات.	٢
الباب الثاني: مفهوم البرمجة الكائنية ومصطلحاتها الأساسية.	٤٧
الباب الثالث: المتغيرات في لغة PHP.	٨١
الباب الرابع: المنطق الشرطي في Conditional Logic in PHP.	١٠٠
الباب الخامس: التتع نماذج HTML.	١٤٠
الباب السادس: الحلقات البرمجية.	١٩١

الباب الأول

مقدمة عن لغات البرمجة وقواعد البيانات

أهداف الباب الأول:

بعد الانتهاء من دراسة الوحدة يرجى ان تكون قادراً على :

١. يحدد مفهوم لغات البرمجة.
٢. تقارن بين أنواع لغات البرمجة.
٣. تحدد لغات البرمجة المستخدمة في نظم برمجة التعلم القائم على الويب
٤. تبين قواعد لغات البرمجة المستخدمة في التعلم القائم على الويب.
٥. تقارن بين لغة JAVA SCRIPT و PHP في برمجة التعلم القائم على الويب
٦. تحدد مفهوم قواعد البيانات .
٧. تحدد أنواع قواعد البيا
٨. تذكر أهمية قواعد البية
٩. تذكر فوائد قواعد البيانات .
١٠. تحدد قواعد البيانات المناسبة لبرمجة التعليم القائم على الويب
١١. تحدد مفهوم صفحات الويب
١٢. تذكر أنواع صفحات الويب .
١٣. تحدد أساسيات ومعايير تصميم وبناء صفحات الويب .
١٤. تشرح كيفية نشر الصفحات والمواقع على الانترنت .
١٥. تحدد أشهر برامج تحرير صفحات الويب شائعة الاستخدام.

مفهوم لغات البرمجة :

هي مجموعة من الأوامر والتعليمات التي تكتب بتسلسل معين وتستخدم للتواصل مع الحاسوب لتنفي مهام محددة . تعمل لغات البرمجة كوسيط بين الانسان والحاسوب ، حيث يتم التعليمات بلغات يمكن للحاسوب فهمها وتنفيذها بعد ترجمتها الى لغة الآلة .

تختلف لغات البرمجة في مستوى التجريد والدقة ، فهناك لغات برمجة منخفضة المستوى مثل لغة التجميع (Assembly) التي تتعامل مباشرة مع مكونات الحاسوب ، ولغات عالية المستوى مثل بايثون وجافا و C++ التي توفر سهولة اكبر في كتابة البرامج وتعتمد على بيئات تطوير متكاملة تساعد المطورين على كتابة التعمد كل أكثر كفاءة

أنواع لغات البرمجة :

هي لغات تستخدم لكتابة البرامج الحاسوبية والتطبيقات المختلفة، وهي الوسيلة التي تمكن المبرمجين من التواصل مع الحواسيب لتنفيذ الأوامر والتعليمات .تنقسم لغات البرمجة إلى عدة أنواع، منها:

تنقسم لغات البرمجة إلى عدة أنواع، بناءً على مستوى قربها من لغة الآلة أو اللغة البشرية وأغراض استخدامها، فيما يلي الأنواع الرئيسية:

١- اللغات منخفضة المستوى : (Low-Level Languages) :

هذه اللغات قريبة من لغة الآلة (الأصفار والواحدات) وتستخدم للتعامل مباشرة مع مكونات الحاسوب.

- لغة التجميع : (Assembly Language) تستخدم لكتابة البرامج التي تحتاج إلى أداء عالٍ وتحكم دقيق في العتاد .كل تعليمة في هذه اللغة تتوافق بشكل مباشر مع تعليمة في لغة الآلة.

٢- اللغات عالية المستوى (High-Level Languages):

تستخدم هذه اللغات لبناء التطبيقات المتقدمة وتتميز بسهولة كتابتها وفهمها لصياغتها القريبة من اللغة البشرية.

-Python: لغة متعددة الأغراض ومناسبة للتعلم الآلي، وتطوير الويب، وتحليل البيانات.

-Java: لغة موجهة للكائنات، تُستخدم لتطوير التطبيقات الكبيرة مثل تطبيقات المؤسسات، وتطبيقات الويب، والأجهزة المحمولة.

-JavaScript: تستخدم في برمجة الويب لإنشاء تطبيقات وتفاعل المستخدمين في المتصفحات.

-C: لغة تستخدم في تطبيقات سطح المكتب، وتطبيقات الألعاب باستخدام منذ .

٣- اللغات الموجهة للكائنات (Object-Oriented Languages):

تقوم على مبدأ "الكائنات" أو "objects" لتنظيم البرمجيات وتعزيز إعادة استخدام الكود.

Java، Python، C++، C# كلها لغات تدعم البرمجة الموجهة للكائنات، وهي مفيدة لتنظيم الأكواد البرمجية الكبيرة والمعقدة.

٤- لغات البرمجة الوظيفية (Functional Programming Languages):

تستخدم مفاهيم الرياضيات والوظائف، حيث يتم التركيز على التعبيرات والوظائف بدلاً من الأوامر والإجراءات.

-Lisp، Scala، Haskell هذه اللغات تُستخدم بشكل رئيسي في البحوث العلمية وبعض التطبيقات المتقدمة.

٥- لغات البرمجة الإجرائية : (Procedural Programming Languages)

تعتمد على الأوامر والإجراءات لتحقيق المهام، ويتم تقسيم البرنامج إلى إجراءات أو دوال.
C ، Pascal، BASIC: تُستخدم لتعليم البرمجة وإجراء بعض المهام البرمجية الأساسية.

٦- لغات البرمجة النصية : (Scripting Languages)

تستخدم لكتابة السكريبتات التي تعمل على تطبيقات أخرى أو لبرمجة الويب.
JavaScript ، PHP، Python، Ruby: تُستخدم لبرمجة المواقع والتطبيقات السريعة.

٧- لغات البرمجة الخاص -ت (Domain)

(Specific Languages) مخصصة لمجالات محددة، مثل قواعد البيانات أو تصميم المواقع.

SQL: -تستخدم لإدارة قواعد البيانات.

HTML- و CSS: تستخدم لتصميم صفحات الويب.

٨- لغات البرمجة المتوازية : (Parallel Programming Languages)

تُستخدم للبرمجة المتوازية حيث يتم تنفيذ عدة أوامر في نفس الوقت.
CUDA- ، OpenCL: تُستخدم في معالجة البيانات المتوازية على وحدات معالجة الرسومات.(GPUs)

كل نوع من لغات البرمجة له استخداماته الخاصة بناءً على متطلبات المشروع أو التطبيق والبيئة التشغيلية التي سيتم تنفيذها فيها.

ما هي لغات البرمجة المستخدمة في برمجة التعليم القائم على الويب ؟

لغات البرمجة المستخدمة في برمجة التعليم القائم على الويب تشمل مجموعة متنوعة من اللغات والتقنيات التي تستخدم لتطوير منصات وأنظمة التعلم الإلكتروني. بعض هذه اللغات تشمل:

١- HTML (Hypertext Markup Language) : تستخدم لإنشاء هيكل الصفحات الإلكترونية والمحتوى الأساسي.

٢- CSS (Cascading Style Sheets) : تستخدم لتنسيق صفحات الويب وتحديد الألوان والخطوط.

٣ - JavaScript : إضافة التفاعلية والديناميكية للمحتوى

صفحات الويب، مثل الاختبارات التفاعلية والعروض التعليمية المتحركة.

٤ - PHP (Hypertext Preprocessor) : تستخدم لتطوير تطبيقات

الويب الديناميكية وإدارة قواعد البيانات على الخادم. تستخدم بشكل شائع

في أنظمة إدارة المحتوى (Moodle).

٥ - Python : تُستخدم في تطوير تطبيقات الويب وخدمات الويب، ويمكن

استخدامها مع أطر عمل مثل Django و Flask لتطوير منصات التعلم الإلكتروني.

٦ - Ruby (on Rails) / تستخدم لتطوير تطبيقات ويب ديناميكية ويمكن

استخدامها لإنشاء أنظمة تعليمية تفاعلية.

٧ - Java : تستخدم لتطوير التطبيقات التي يمكن تشغيلها على الخوادم

والمصفحات، ولها استخدامات واسعة في الأنظمة التعليمية الكبرى.

٨- (Structured Query Language) : تستخدم لإدارة قواعد البيانات،

وهي ضرورية لتخزين واسترجاع البيانات المتعلقة بالطلاب والمحتوى التعليمي.

٩- Node.js :تستخدم لتنفيذ JavaScript على الخادم، مما يسمح بإنشاء تطبيقات ويب سريعة وكفوءة.

١٠- Frameworks : مثل React، Angular ، و Vue.js**تستخدم لبناء واجهات المستخدم التفاعلية وتجربة المستخدم المتقدمة في تطبيقات التعلم الإلكتروني.

يمكن استخدام مزيج من هذه اللغات والتقنيات لإنشاء تجربة تعلم إلكتروني متكاملة تشمل إدارة المحتوى، وتخصيص التعلم.

قواعد لغات البرمجة :

قواعد لغات البرمجة، أو "syntax" ، هي مجموعة من القواعد والتركيب التي تحدد كيفية كتابة التعليمات والأوامر بلغة برمجة معينة بحيث يمكن للمترجم أو المفسر (compiler or interpreter) فهمها وتنفيذها . هذه القواعد تشمل كيفية تعريف المتغيرات، وكتابة الشروط والحلقات، وإنشاء الدوال، واستخدام التعليقات، وغيرها من العناصر البرمجية.

أهم النقاط المتعلقة بقواعد لغات البرمجة:

١- تعريف المتغيرات : تحديد الأسماء المسموح بها للمتغيرات، ونوع البيانات التي يمكن تخزينها فيها.

٢- التركيب الشرطية : (كيفية كتابة الشروط) : مثل `if-else` التي تحدد مسار التنفيذ بناءً على القيم أو الحالات.

٣- الحلقات : استخدام الحلقات مثل : `for`, `while` .٣ التكرار تنفيذ جزء من الكود.

٤- الدوال : كيفية تعريف الدوال (functions) التي تحتوي على مجموعة من التعليمات التي يمكن استدعاؤها عند الحاجة.

٥- المشغلين : قواعد استخدام المشغلين) مثل (`/`, `*`, `-`, `+` لإجراء العمليات الحسابية أو المنطقية.

٦- التعليقات : كيفية إضافة تعليقات داخل الكود لتوضيح أجزاء معينة للمطورين الآخرين أو لنفسك في المستقبل.

٧- البنية الأساسية للكود : مثل كيفية تنظيم الكود باستخدام الأقواس أو المسافات البادئة (dents) ها.

كل لغة برمجة لـ ها الخاصة، وقد تكون متشابهة في

النواحي أو مختلفة تمامًا في نواحٍ أخرى .

ما هي لغة جافا سكريبت :

-جافاسكريبت(JavaScript) : هي لغة برمجة عالية المستوى تُستخدم أساسًا لتطوير مواقع الويب التفاعلية .تم تطويرها في الأصل لتعمل على جانب المتصفح (Client-side) لجعل صفحات الويب ديناميكية وتفاعلية، ولكنها أصبحت الآن تُستخدم أيضًا على جانب الخادم (Server-side) باستخدام بيئات مثل Node.js.

-جافاسكريبت تتيح للمطورين تنفيذ عمليات مثل التحقق من المدخلات، التعامل مع الأحداث، إنشاء الرسوم المتحركة، وتحديث محتوى الصفحة دون الحاجة إلى إعادة تحميلها .كما تُستخدم جافاسكريبت في العديد من

التطبيقات الحديثة بما في ذلك تطبيقات الويب التفاعلية، تطبيقات الهواتف الذكية، وحتى التطبيقات المكتبية.

قواعد لغة (JavaScript) تشمل مجموعة من التعليمات والقواعد الأساسية التي يجب على المبرمجين اتباعها لكتابة كود صحيح وقابل للتنفيذ .

أهم قواعد لغة JavaScript :

١ - تعريف المتغيرات :

- يمكن تعريف المتغيرات باستخدام الكلمات المحجوزة ``var``, ``let`` أو ``const`` :

```
````javascript
var x = ١٠; // يير القيمة لاحقًا
let y = ٢٠; // يمكن تغيير القيمة لاحقًا، ويكون محدود النطاق
 (block-scoped)
const z = ٣٠; // لا يمكن تغيير القيمة بعد التعيين، ويكون محدود
 النطاق (block-scoped)
````
```

٢ - التراكيب الشرطية :

- استخدام ``if``, ``else if`` و ``else`` للتحكم في تدفق البرنامج :

```
````javascript
if (x > y) {
 console.log("أكبر من: y")
} else if (x < y) {
 console.log("أصغر من: y")
}
```

```

 } else {
 console.log("x
 يساوي y");
 }
 ...

```

### ٣- الحلقات:

- كتابة الحلقات لتكرار تنفيذ جزء من الكود:

- حلقات: `for`

```javascript

```
for (let i = ٠; i < ٥; i++) {
```

```
    console.log(i);
```

```
}
```

...

- حلقات: `while`

```javascript

```
let i = ٠;
```

```
while (i < ٥) {
```

```
 console.log(i);
```

```
 i++;
```

```
}
```

...

### ٤- تعريف الدوال: (Functions)

- إنشاء دوال باستخدام الكلمة المحجوزة `function` أو باستخدام تعبير

دالي (function expression) أو الدوال السهمية: (arrow functions)

```

````javascript
function greet(name) {
  return "Hello, " + name;
}

const greet = function(name) {
  return "Hello, " + name;
};

const greet = (name) => "Hello, " + name;
...

```

٥- المشغلين : (tors)

- تشمل المشغلين الح (`+`, `-`, `*`, `/`) والمشغلين المن
 (`&`, `||`, `!`) والمشغلين
 المقارنة. (`==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, `<=`)

٦- التعليقات :

- التعليقات المفردة : تستخدم `//` لتعليق سطر واحد.

```

````javascript
// هذا تعليق
let x = ١٠; // تعريف متغير
...

```

- التعليقات المتعددة الأسطر : تستخدم `/\* ... \*/` لتعليق عدة أسطر.

```

````javascript
/*
هذا تعليق

```

متعدد الأسطر

*/
...

٧- القيم والأنواع: (Data Types)

JavaScript - تدعم أنواع البيانات مثل `Number`, `String`, `Boolean`, `Object`, `Function`, `Array` و. `Undefined`

٨- البنية الأساسية للكود:

- يجب إنهاء الأسطر بالفاصلة المنقوطة `;` اختياري في كثير من الحالات ولكن يفضل
التجنب الأخطاء.
- استخدام الأقواس `}` د كتل الكود في الدوال، الحلقات، والشد

٩- تنظيم الكود:

- المسافات البيضاء : مسافات فارغة أو أسطر جديدة لا تؤثر على تنفيذ الكود، لكن يُنصح باستخدام المسافات البادئة (indents) لتحسين قراءة الكود.

- الأقواس: الأقواس المربعة `[]` تستخدم للمصفوفات، والأقواس `{}` تستخدم للأجسام (Objects) ولتحديد كتل الكود.

١٠- إدارة الأخطاء: (Error Handling)

- استخدام `try`, `catch` لإدارة الأخطاء والتأكد من أن الكود يعمل بشكل صحيح حتى في حالة وجود خطأ.

```
````javascript
```

```
try {
```

```
// محاولة تشغيل هذا الكود
```

```
let result = riskyOperation();
```

```
 } catch (error) {
```

```
 // إدارة الخطأ هنا
```

### ١- القواعد الأساسية للتعليمات:

- التعليقات الأحادية :تستخدم `//` لتحديد تعليق من سطر واحد.
- التعليقات المتعددة الأسطر :تستخدم `/\* ... \*/` لتحديد تعليق من عدة أسطر.

### ٢- القواعد الأساسية لتعريف المتغيرات:

- تستخدم `var` أو `let` أو `const` لتعريف المتغيرات.
- `var` - يستخدم لتغييرات بطريقة تقليدية.
- `let` : يستخدم لتعريف يمكن تغييره فيما بعد داخل نطاق محد دالة أو كتلة.

`const` : يستخدم لتعريف متغير لا يمكن تغييره بعد تعيين قيمته.

### ٣- الفرق بين الأنواع:

- الأنواع الأولية , `null`, `boolean`, `number`, `string` :
- `symbol`, `undefined`, و `bigint`.
- الأنواع المرجعية , `array`, `object`, و: `function`.

### ٤-قواعد الشروط:

- يمكن استخدام جمل الشرط , `else if`, `if`, و `else` للتحكم في التدفق.

- يمكن استخدام جملة `switch` للحالات المتعددة.

### ٥-الحلقات:

, `while`, `for` - و `do...while` هي الحلقات الأساسية المستخدمة لتنفيذ الشيفرة بشكل متكرر.

## ٦-الدوال : (Functions)

- يمكن تعريف دالة باستخدام الكلمة الأساسية `function` أو باستخدام تعبير السهم. (`=>`)

- يمكن للدوال أن تحتوي على معلمات وتُرجع قيمة.

## ٧-القواعد الأساسية للنطاق : (Scope)

- النطاق المحدد بـ `var` هو نطاق دالة، بينما النطاق المحدد بـ `let` و `const` هو نطاق كتلة.

## ٨- تعامل مع الأحداث : (Event)

- جافا سكربت تدعم مع أحداث المستخدم مثل النقر بالماو الكتابة في حقل نصي باستخدام مستمعي الأحداث (`addEventListener`).

## ٩-التعامل مع الأخطاء :

- يمكنك استخدام `try...catch` للقبض على الأخطاء ومعالجتها في الشيفرة.

## ١٠-الاستخدام الآمن والمتوافق :

- تأكد من التحقق من التوافق بين المتصفحات، لأن بعض ميزات جافا سكربت قد لا تكون مدعومة على جميع المتصفحات.

## قواعد لغة PHP :

لغة البرمجة PHP (Hypertext Preprocessor) هي لغة برمجة تستخدم بشكل أساسي لتطوير تطبيقات الويب وتطبيقات الخادم. تعد من

لغات البرمجة النصية (Scripting Languages) التي تعمل على جانب

الخادم (Server-Side) إليك بعض القواعد الأساسية في لغة: PHP

١- بداية ونهاية الكود:

يتم كتابة كود PHP بين العلامات التالية:

```
```php
```

```
<?php
```

```
//الكود هنا
```

```
?>
```

```
```
```

٢- تعريف المتغيرات:

يتم تعريف المتغيرات با لامة الدولار '\$' ، ولا تحتاج إلى ت

مسبق بنوع البيانات . أمثلة:

```
```php
```

```
$name = "John"; //متغير نصي (string)
```

```
$age = ٣٠; //متغير عددي (integer)
```

```
$price = ١٩.٩٩; //متغير عددي عشري (float)
```

```
$is_student = true; //متغير منطقي (boolean)
```

```
```
```

٣- التعليقات:

يمكن كتابة التعليقات بطريقتين:

-تعليق سطر واحد:

```
```php
```

```
// هذا تعليق سطر واحد
```

...

-تعليق متعدد الأسطر:

```
```php
```

```
/*
```

هذا تعليق

متعدد الأسطر

```
*/
```

...

٤- الجمل الشرطية:

تستخدم 'if' ، 'else if' ، 'لتتحقق من الشروط.

```
```php
```

```
if ($age > ١٨) {
```

```
    echo "You are an adult.";
```

```
    } elseif ($age == ١٨) {
```

```
        echo "You are exactly ١٨ years old.";
```

```
    } else {
```

```
        echo "You are a minor.";
```

```
    }
```

...

٥-الحلقات التكرارية:

تدعم PHP عدة أنواع من الحلقات:

- **`for` loop:**

```
```php
```



```

for ($i = ٠; $i < ١٠; $i++) {
 echo $i;
}
...

```

– **while` loop:**

```

```php
$i = ٠;
while ($i < ١٠) {
    echo $i;
    $i++;
}
...

```

foreach` loop: مخصصة للمصفوفات:

```

```php
$colors = array("red", "green", "blue");
foreach ($colors as $color) {
 echo $color;
}
...

```

**٦- المصفوفات:**

تدعم PHP أنواعًا مختلفة من المصفوفات.

**مصفوفة مفهرسة (Indexed Array):**

```

```php

```

```
$fruits = array("Apple", "Banana", "Orange");
...

```

– مصفوفة ترابطية : (Associative Array)

```
```php
$ages = array("Peter" => ٣٥, "John" => ٢٠);
...

```

### ٧-الدوال:

تسمح PHP بإنشاء دوال لتعزيز إعادة استخدام الكود.

```
```php
function greet($name) {
    "Hello, " . $name;
}

echo greet("Alice");
...

```

٨- التعامل مع النصوص:

يمكن استخدام علامات الاقتباس المزدوجة `""` والمفردة `''` للتعبير عن

النصوص، مع وجود اختلافات في معالجة المتغيرات بداخلها:

– علامات الاقتباس المزدوجة : تسمح بتفسير المتغيرات:

```
```php
$name = "Alice";

echo "Hello, $name"; // سيظهر Hello, Alice
...

```

– علامات الاقتباس المفردة : لا تفسر المتغيرات:

```

`php
: Hello, $name سيطهر echo 'Hello, $name'; //
`

```

## ٩- التعامل مع الملفات:

توفر PHP دوالاً لفتح الملفات، وقراءة محتوياتها، وكتابتها، وإغلاقها:

```

`php
$file = fopen("example.txt", "r"); // $file = fopen للفتح
$content = fread($file, filesize("example.txt")); // $content = fread للقراءة
fclose($file); // fclose لإغلاق الملف
`

```

## ١٠- التعامل مع قواعد

تدعم PHP العديد من قواعد البيانات مثل MySQL يمكن استخدام مكتبة PDO أو `mysqli` للتعامل مع قواعد البيانات.

## قواعد البيانات Data base

هي نظام منظم لتخزين وإدارة البيانات بطريقة تسهل الوصول إليها واستخدامها وتحديثها. تُستخدم قواعد البيانات لتجميع البيانات في جداول مرتبطة تحتوي على معلومات منظمة، مثل الأسماء والعناوين والأرقام والمواعيد وغيرها من التفاصيل. يتم تنظيم البيانات في قواعد البيانات بشكل يسمح باسترجاعها بسرعة وبدقة باستخدام لغة برمجة مخصصة مثل لغة SQL.

تُستخدم قواعد البيانات في العديد من التطبيقات مثل أنظمة إدارة الموارد البشرية، وإدارة المخزون، والخدمات المصرفية، والتجارة الإلكترونية،

والشبكات الاجتماعية، وغيرها من المجالات التي تتطلب تخزين ومعالجة كميات كبيرة من البيانات بشكل فعال.

### أنواع قواعد البيانات :

هناك عدة أنواع من قواعد البيانات، وكل نوع يناسب احتياجات معينة ومتطلبات محددة، إليك بعض الأنواع الشائعة:

#### ١ - قواعد البيانات العلائقية : (Relational Databases)

تعتمد على تنظيم البيانات في جداول مترابطة، حيث يمكن ربط الجداول باستخدام المفاتيح الأساسية والخارجية. تُستخدم قواعد البيانات العلائقية بشكل واسع في الأعمال، مثل نظم إدارة قواعد البيانات (DBMS) مثل MySQL، PostgreSQL، و Oracle Database و SQL Server.

#### ٢ - قواعد البيانات غير العلائقية : (NoSQL Databases)

تُستخدم لتخزين البيانات غير المنظمة أو شبه المنظمة وهي مناسبة لتطبيقات البيانات الكبيرة (Big Data).

(Data) وتطبيقات الويب التي تحتاج إلى معالجة البيانات بسرعة. تشمل أنواع قواعد بيانات NoSQL قواعد البيانات المستندة إلى الوثائق (Document-based) مثل MongoDB، وقواعد البيانات الموزعة مثل Cassandra، وقواعد البيانات المستندة إلى الأعمدة (Column-based) مثل HBase.

#### ٣ - قواعد البيانات الهرمية : (Hierarchical Databases)

تستخدم بنية شجرية لتنظيم البيانات، حيث تكون كل سجلات البيانات مرتبطة بسجل رئيسي واحد. تُستخدم عادة في التطبيقات القديمة أو الأنظمة المالية.

#### ٤- قواعد البيانات الشبكية: (Network Databases)

تشبه قواعد البيانات الهرمية، ولكنها تسمح لكل سجل بأن يكون مرتبطاً بعدة سجلات أخرى، مما يوفر بنية شبكية أو غير خطية .

#### ٥ - قواعد البيانات الموزعة: (Distributed Databases)

تخزن البيانات في مواقع متعددة، قد تكون على شبكة محلية أو شبكة إنترنت، وتعمل كقاعدة بيانات واحدة. تُستخدم لتحسين الأداء وتوفير التوافر العالي للبيانات.

#### ٦ - قواعد البيانات الزمنية (Time-Series Databases)

مخصصة لتخزين وإدارة البيانات التي تعتمد على الزمن، مثل بيانات أجهزة الاستشعار أو التحليلات المالية. أمثلة على ذلك تشمل InfluxDB و TimescaleDB.

#### ٧- قواعد البيانات الكائنية: (Object-Oriented Databases)

تخزن البيانات في شكل كائنات كما هو الحال في البرمجة الكائنية (OOP) وهي مناسبة لتطبيقات تتطلب تخزين ومعالجة البيانات المعقدة.

#### ٨ - قواعد البيانات الرسومية: (Graph Databases)

تعتمد على بنية الرسم البياني لتخزين البيانات، حيث يتم تمثيل البيانات على شكل عقد وعلاقات. تُستخدم في التطبيقات التي تتطلب تحليل العلاقات، مثل الشبكات الاجتماعية، ومنصات التوصية، مثل Neo4j. كل نوع من قواعد البيانات له مميزاته وفوائده الخاصة، ويُستخدم بناءً على متطلبات التطبيق واحتياجات العمل.

## أهمية قواعد البيانات :

تتمثل أهمية قواعد البيانات في عدة جوانب، منها:

- ١-تنظيم البيانات وتخزينها بشكل فعال : تسمح قواعد البيانات بتخزين كميات كبيرة من البيانات بطريقة منظمة، مما يسهل الوصول إليها وإدارتها.
  - ٢- القدرة على البحث السريع واسترجاع البيانات : توفر قواعد البيانات أدوات قوية للبحث واسترجاع المعلومات بسرعة، مما يوفر الوقت والجهد.
  - ٣- تحقيق التكامل والاتساق : تساعد قواعد البيانات على ضمان تكامل البيانات واتساقها بين الأنظمة المختلفة، مما يقلل من الأخطاء والتكرار.
  - ٤- الحفاظ على سرية البيانات وأمانها : توفر قواعد البيانات أدوات للحفاظ على سرية البيانات مثل أنظمة التشفير والتحكم في الوصول
  - ٥- دعم القرارات : تدعم مؤسسات قواعد البيانات لتحليل البيانات واستخراج معلومات قيمة تدعم اتخاذ القرارات المستتيرة.
  - ٦- القدرة على مشاركة البيانات : تسمح قواعد البيانات بمشاركة البيانات بين المستخدمين المختلفين بشكل آمن وفعال.
  - ٧- توفير النسخ الاحتياطي والاستعادة :توفر قواعد البيانات وسائل لإنشاء نسخ احتياطية للبيانات واستعادتها في حالة حدوث خلل أو فقدان للبيانات .
- قواعد البيانات تعد أداة حيوية لتمكين الأنظمة المعلوماتية الحديثة من العمل بكفاءة وفعالية.

## فوائد قواعد البيانات متعددة وتشمل:

- ١- تحسين إدارة البيانات : تساعد قواعد البيانات في تخزين وتنظيم البيانات بشكل يسهل الوصول إليها وإدارتها بكفاءة.
- ٢- تقليل التكرار والازدواجية : تقلل قواعد البيانات من تكرار البيانات عن طريق تنظيمها في جداول وعلاقات محددة، مما يساهم في تحسين كفاءة النظام وتقليل مساحة التخزين المطلوبة.
- ٣- تحسين دقة البيانات : بفضل قواعد التحقق والتكامل، تضمن قواعد البيانات أن تكون البيانات المدخلة دقيقة ومتسقة.
- ٤- تحسين سرعة الوصول إلى البيانات : توفر قواعد البيانات آليات للبحث السريع واسترجاع البيانات، مما يوفر الوقت والجهد للمستخدمين.
- ٥- زيادة الأمان والحماية : قواعد البيانات مستويات أمان متقدمة التحكم في الوصول وا لحماية البيانات الحساسة من الوصول المصرح به.
- ٦- سهولة الصيانة والتحديث : تتيح قواعد البيانات تحديث البيانات وصيانتها بمرونة ودون الحاجة إلى تغييرات كبيرة في النظام.
- ٧- تحسين القدرة على اتخاذ القرارات : توفر قواعد البيانات أدوات للتحليل والتقارير التي تساعد المؤسسات على استخراج المعلومات وتحليلها لدعم اتخاذ قرارات أفضل.
- ٨- دعم العمل الجماعي : تمكن قواعد البيانات من مشاركة البيانات بين العديد من المستخدمين في نفس الوقت بطريقة آمنة ومنظمة.
- ٩- دعم النسخ الاحتياطي واستعادة البيانات : تسمح قواعد البيانات بإنشاء نسخ احتياطية دورية واستعادة البيانات في حالة حدوث خلل أو فقدان للبيانات.

هذه الفوائد تجعل قواعد البيانات أداة أساسية في إدارة المعلومات والبيانات في العديد من المجالات.

### أهمية قواعد البيانات في التعليم :

قواعد البيانات تلعب دورًا حيويًا في التعليم بعدة طرق، منها:

- ١- تخزين المعلومات : تتيح قواعد البيانات تخزين كميات كبيرة من المعلومات والبيانات التعليمية بطريقة منظمة وسهلة الوصول إليها.
- ٢- إدارة البيانات : تساعد في إدارة بيانات الطلاب والمعلمين، مثل الدرجات، والحضور، والأنشطة، مما يسهل متابعة التقدم الأكاديمي وتحليل الأداء.
- ٣- تحليل البيانات : توفر تحليل قوية تتيح للمدرسين والإداريين البيانات لاتخاذ قرارات لى المعلومات، مثل تحديد المجالات تحتاج لتحسين.

- ٤- تعزيز التعلم الشخصي : تمكن من إنشاء أنظمة تعليمية مخصصة تتكيف مع احتياجات كل طالب بناءً على سجلاته وأدائه.
- ٥- تحسين التواصل: تسهم في تحسين التواصل بين الطلاب والمعلمين عبر إدارة الأنشطة والموارد التعليمية بشكل أكثر كفاءة.

- ٦- تسهيل الوصول إلى المحتوى : تسهم في توفير الوصول السريع والموثوق إلى محتوى تعليمي متنوع وموارد تعليمية.

### ما هي قواعد البيانات المناسبة لبرمجة التعليم القائم على الويب؟

عند برمجة التعليم القائم على الويب، من المهم اختيار قاعدة بيانات تتناسب مع احتياجات النظام والتطبيقات. بعض قواعد البيانات الشائعة والمناسبة تشمل:



- ١- MySQL : قاعدة بيانات مفتوحة المصدر، مناسبة للتطبيقات الصغيرة والمتوسطة الحجم. توفر دعمًا جيدًا للمعاملات وتعتبر من الخيارات الشائعة في تطوير الويب.
- ٢- PostgreSQL : قاعدة بيانات مفتوحة المصدر، تدعم الكثير من المزايا المتقدمة مثل البيانات الجغرافية والمعاملات القوية، مما يجعلها خيارًا ممتازًا للتطبيقات الكبيرة والمعقدة.
- ٣- SQLite : قاعدة بيانات خفيفة الوزن ومبنية على ملفات، مناسبة للتطبيقات الصغيرة أو تلك التي تحتاج إلى قاعدة بيانات مدمجة.
- ٤- MongoDB : قاعدة بيانات NoSQL تدعم تخزين البيانات غير المنظمة أو شبه المنظّمة البيانات النصية الكبيرة. مناسبة للتطبيقات التي تحتاج إلى مرونة مل مع أنواع مختلفة من البيانات.
- ٥- Firebase Realtime Database : قاعدة بيانات في الوقت الفعلي، مخصصة للتطبيقات التي تحتاج إلى تحديثات مباشرة وتفاعلية، مثل التطبيقات التعليمية التعاونية.
- ٦- Microsoft SQL Server : قاعدة بيانات قوية وقابلة للتوسع، توفر دعمًا جيدًا للتعامل مع كميات كبيرة من البيانات والمعاملات المعقدة. اختيار القاعدة المناسبة يعتمد على حجم التطبيق، نوع البيانات، ومتطلبات الأداء .

إليك شرح تفصيلي لقواعد البيانات التي تم ذكرها:

## ١-MySQL:

- الوصف : قاعدة بيانات مفتوحة المصدر تعمل بنظام إدارة قواعد البيانات العلائقية، تُستخدم على نطاق واسع في تطوير الويب.

- المزايا:

- أداء عالٍ وقابلية التوسع.
- دعم كبير من قبل المجتمع والمطورين.
- توفر أدوات إدارة متقدمة مثل phpMyAdmin.

- العيوب:

- يمكن أن تكون محدودة في بعض المزايا المتقدمة مقارنةً بـ PostgreSQL.

## ٢- PostgreSQL :

- الوصف : قاعدة بيانات مفتوحة المصدر تُعرف بمرونتها وقدرتها على التعامل مع البيانات الـ تقدمية.
- المزايا:

- دعم للمزايا المتقدمة مثل الجداول المترابطة، المعاملات ACID ، والبيانات الجغرافية.

- القدرة على التعامل مع أنواع بيانات متعددة.
- دعم للغات البرمجة المدمجة.

- العيوب:

- قد تكون أكثر تعقيدًا في الإعداد والتكوين مقارنةً بـ MySQL.

## ٣- SQLite :

- الوصف : قاعدة بيانات خفيفة الوزن مبنية على ملفات، لا تحتاج إلى خادم قاعدة بيانات مخصص.
- المزايا:

- سهولة التثبيت والاستخدام.
- صغيرة الحجم وسريعة الأداء.

- مناسبة للتطبيقات الصغيرة والمتوسطة الحجم.

-العيوب:

- لا تدعم المعاملات المتقدمة مثل قواعد البيانات العلائقية الكبيرة.

- ليس مثاليًا للتطبيقات ذات البيانات الضخمة أو التفاعل العالي.

#### ٤- MongoDB :

-الوصف : قاعدة بيانات NoSQL تعتمد على تخزين البيانات بصيغة

JSON، مما يتيح مرونة كبيرة في تخزين البيانات غير المنظمة.

-المزايا:

- مرونة في التعامل مع أنواع مختلفة من البيانات.

- يدعم عمليات الكتابة السريعة.

- مناسبة للتطبيقات الـ إلى بيانات غير هيكلية أو بيانات

التغيير.

-العيوب:

- قد يكون من الصعب إدارة المعاملات المعقدة أو ضمان التكامل بين

البيانات.

- يمكن أن تستهلك موارد كبيرة في حالات الاستخدام المكثف.

#### ٥- Firebase Realtime Database :

- الوصف : قاعدة بيانات قائمة على السحابة توفر تحديثات فورية للبيانات

عبر الإنترنت.

- المزايا:

- تحديثات في الوقت الفعلي مما يجعلها مثالية للتطبيقات التعاونية

والتفاعلية.

- تتيح تخزين البيانات بصيغة JSON.

- سهولة التكامل مع خدمات Firebase الأخرى.

-العيوب:

- قد تكون التكلفة مرتفعة بناءً على حجم الاستخدام.

- قد تكون محدودة في بعض المزايا مقارنةً بقواعد بيانات أخرى.

## ٦-Microsoft SQL Server:

-الوصف: قاعدة بيانات تجارية تعتمد على نظام إدارة قواعد البيانات

العلائقية، تقدمها شركة Microsoft.

-المزايا:

- دعم متقدم للمعاملات والبيانات الكبيرة.

- أدوات إدارة وتقارير مثل L Server Management

Studio.

- تكامل جيد مع تطبيقات Microsoft الأخرى.

-العيوب:

- يمكن أن تكون تكلفة الترخيص مرتفعة.

- قد يكون الإعداد والإدارة أكثر تعقيدًا مقارنةً بقواعد البيانات المفتوحة

المصدر.

كل قاعدة بيانات لها استخداماتها الخاصة، والاختيار المناسب يعتمد على

متطلبات المشروع المحددة، مثل حجم البيانات، نوع البيانات، ومتطلبات

الأداء.

## كيفية استخدام قواعد البيانات في برمجة التعليم القائم على الويب

استخدام قواعد البيانات في برمجة التعليم القائم على الويب له أهمية كبيرة في إدارة المعلومات وتحسين تجربة التعلم. إليك بعض الطرق التي يمكنك استخدامها:

١- تخزين بيانات المستخدمين : يمكن تخزين معلومات الطلاب، المدرسين، والمواد التعليمية في قواعد البيانات. يشمل ذلك بيانات تسجيل الدخول، التقدم الأكاديمي، والمعلومات الشخصية.

٢- إدارة المحتوى التعليمي : يمكن تخزين المحتوى التعليمي مثل الدروس، الامتحانات، والموارد التعليمية في قاعدة البيانات بحيث يمكن الوصول إليها وتحديثها بسهولة.

٣- تتبع التفاعل : يمكن بيانات حول تفاعل الطلاب مع المحتوى مثل الأسئلة التي تم الإجابة عليها، الدرجات، والأوقات التي قضوها في الدراسة.

٤- إدارة التقييمات : يمكن إدارة الأسئلة والأجوبة والدرجات باستخدام قاعدة البيانات، مما يسهل على المعلمين تتبع أداء الطلاب وإصدار الشهادات.

٥- إمكانية التخصيص : يمكن تخصيص المحتوى التعليمي بناءً على بيانات من قاعدة البيانات، مثل تقديم توصيات بالدروس بناءً على أداء الطالب أو اهتماماته.

٦- التحليلات والتقارير : يمكن تحليل البيانات المخزنة في قواعد البيانات لإنشاء تقارير حول أداء الطلاب وفعالية البرامج التعليمية.

لتحقيق هذه الأهداف، تحتاج إلى اختيار قاعدة بيانات مناسبة (مثل PostgreSQL، MySQL، أو MongoDB) واستخدام لغات برمجة مثل

PHP، Python ، أو JavaScript للتفاعل مع قاعدة البيانات عبر تطبيقات الويب.

## مفهوم صفحات الويب :

صفحات الويب هي وثائق إلكترونية تعرض على شبكة الإنترنت وتكون مكتوبة بلغات ترميزية مثل HTML (لغة ترميز النص الشبكي) . يتم الوصول إلى هذه الصفحات من خلال متصفح ويب عن طريق إدخال عنوان URL (محدد موقع الموارد المحدد) أو عن طريق الربط .  
تحتوي صفحات الويب على مجموعة متنوعة من المحتوى ، بما في ذلك النصوص والصور الفيديو والروابط التشعبية والرسوم التفاعلية ويمكنها أيضا تنفيذ التعليق  
جانبية باستخدام سؤلهف لغات مثل SCRIPT لتوفير تفاعلات ديناميكية مع المستخدم  
تشكل مجموعة من صفحات الويب المترابطة موقع ويب كامل يمكن ان يكون لأغراض متعددة مثل التعليم ، الترفيه ، التجارة الإلكترونية وغيرها .

## أساسيات ومعايير تصميم وبناء صفحات الويب :

تصميم وبناء صفحات الويب يتطلب مراعاة مجموعة من الأساسيات والمعايير لضمان توفير تجربة مستخدم ممتازة وتحقيق أهداف الموقع . إليك أهم هذه الأساسيات والمعايير:

### ١ - التخطيط والتنسيق: (Layout and Structure)

- التخطيط الشبكي: (Grid Layout) استخدام تخطيط شبكي يساعد في تنظيم المحتوى بشكل منظم ومتناسق.

-التسلسل الهرمي للمحتوى (Content Hierarchy) ترتيب العناصر بطريقة تسهل على المستخدم فهم المعلومات، حيث تكون العناصر المهمة بارزة وواضحة.

-استخدام المساحات البيضاء (White Space) ترك مسافات فارغة بين العناصر لجعل المحتوى أكثر وضوحًا وتقليل الإجهاد البصري.

٢- التجاوب مع الأجهزة المختلفة : (Responsiveness)

-تصميم الصفحات لتكون متجاوبة (Responsive Design) بحيث تتكيف مع مختلف أحجام الشاشات) الحواسيب، الأجهزة اللوحية، الهواتف الذكية.

-استخدام وحدات قياس ثل النسب المئوية ( بدلاً من الوحدات لضمان مرونة التصميم.

٣- التوافق مع المعايير (Standards)

Compliance-

- الالتزام بمعايير اتحاد الشبكة العالمية (W٣C) لضمان تشغيل الموقع بشكل صحيح عبر جميع المتصفحات.

- استخدام لغات البرمجة المناسبة مثل HTML٥ و CSS٣ لتحقيق الهيكل الجيد والتنسيق المطلوب.

٤- تحسين الأداء والسرعة:(Performance Optimization) :

- تقليل حجم الصور والملفات الأخرى لتحسين سرعة تحميل الصفحة.

- استخدام تقنيات التخزين المؤقت (Caching) وتقليل طلبات HTTP.

- ضغط الملفات) مثل استخدام Gzip وتقليل الأكواد (Minification)

لتحسين أداء الموقع.

٥- التجربة البصرية(Visual Design) :

-اختيار ألوان متناسقة ومريحة للعين، مع مراعاة توجيه الانتباه إلى العناصر المهمة.

-استخدام خطوط قابلة للقراءة وأحجام خط مناسبة للنصوص.

-توظيف الصور والرسوم التوضيحية بشكل فعال لدعم المحتوى وليس تشتيت الانتباه.

٦- تحسين محركات البحث (SEO) :

-استخدام علامات HTML بشكل صحيح، مثل العناوين (H١)، H٢، ...)

ووصف الصور باستخدام خصائص الـ "alt"

-توفير محتوى عالي الجودة ومفيد يجذب الزوار ويبقيهم على الموقع لفترة أطول.

-تحسين الروابط الداخلي ية والتأكد من سرعة تحميل الموقع.

٧- سهولة الاستخدام (Usability) :

-تصميم واجهة مستخدم بسيطة وسهلة الفهم، مع تحديد واضح للأزرار وروابط التنقل.

-تقديم تجارب تفاعلية ومفهومة للمستخدم، مثل النماذج والقوائم المنسدلة.

-اختبار الموقع بشكل دوري مع المستخدمين الحقيقيين لضمان جودة الاستخدام.

٨- إمكانية الوصول (Accessibility) :

-تصميم الموقع ليكون متاحًا لجميع المستخدمين، بما في ذلك ذوي الاحتياجات الخاصة.

-استخدام تقنيات مثل WAI-ARIA لدعم التقنيات المساعدة كقراء الشاشة.

-التأكد من أن الألوان والتباين يلبيان معايير الوصول.

٩- الأمان (Security) :



- استخدام بروتوكول HTTPS لتأمين نقل البيانات بين المستخدم والخادم.
  - حماية الموقع من الهجمات الشائعة مثل الـ SQL Injection والـ XSS.
  - تحديث البرمجيات والمكونات بشكل دوري لسد الثغرات الأمنية.
- ١٠ - المحتوى (Content):

- تقديم محتوى قيم، ملائم، وجذاب للمستخدمين.
- تحديث المحتوى بشكل دوري للحفاظ على جاذبية الموقع وأهميته.
- مراعاة هذه الأساسيات والمعايير يساهم في بناء صفحات ويب ذات جودة عالية تحقق الأهداف المرجوة وتوفر تجربة استخدام مميزة.

## انواع صفحات الويب

أنواع صفحات الويب تت

- ١-الصفحات الثابتة (Static Web Pages) هي صفحات تحتوي على محتوى ثابت لا يتغير إلا بتدخل من مطور الويب . تُستخدم عادةً لعرض معلومات لا تتطلب التحديث بشكل منتظم، مثل صفحات المعلومات العامة للشركات.

- ٢- الصفحات الديناميكية (Dynamic Web Pages) هي صفحات تُنشأ بشكل ديناميكي باستخدام البرمجة الخلفية) مثل PHP أو ASP.NET) وقواعد البيانات .المحتوى يتغير بناءً على تفاعل المستخدم أو بناءً على البيانات الموجودة في قاعدة البيانات .

- ٣- صفحات الويب التفاعلية (Interactive Web Pages) هي صفحات تتيح للمستخدمين التفاعل مع المحتوى بطرق متعددة، مثل ملء النماذج، وإجراء عمليات البحث، والنقر على الأزرار، وغيرها.

- ٤- صفحات المدونات (Blog Pages) تُستخدم لنشر المقالات والمدونات، وتتيح عادة للمستخدمين التعليق والتفاعل مع المحتوى.
- ٥- صفحات التجارة الإلكترونية (E-commerce Pages) مخصصة لبيع المنتجات والخدمات عبر الإنترنت، مثل المتاجر الإلكترونية.
- ٦- صفحات الهبوط (Landing Pages) تُستخدم عادة لأغراض التسويق، وهي مصممة لجذب انتباه المستخدم وتشجيعه على اتخاذ إجراء معين، مثل التسجيل للحصول على خدمة أو منتج.
- ٧- صفحات الوسائط الاجتماعية (Social Media Pages) خاصة بشبكات التواصل الاجتماعي، مثل فيسبوك وتويتر، وتسمح للمستخدمين بنشر المحتوى والتقاء
- ٨- صفحات المحركات (Search Engine Pages) صفحا نتائج البحث للمستخدمين، مثل جوجل وياهو .
- كل نوع من هذه الأنواع يُستخدم لأغراض مختلفة ويخدم احتياجات محددة للمستخدمين والمواقع.

### كيفية نشر الصفحات والمواقع على الإنترنت :

- لنشر الصفحات والمواقع على الإنترنت، هناك عدة خطوات يجب اتباعها:
- ١- تصميم الموقع : يجب إنشاء وتصميم صفحات الموقع باستخدام لغات البرمجة مثل HTML، CSS ، وجافا سكريبت .يمكن استخدام أدوات وبرامج تصميم مواقع جاهزة مثل WordPress أو Wix لتسهيل العملية.
- ٢- شراء اسم النطاق : (Domain Name) اختيار وشراء اسم نطاق يعبر عن موقعك) مثل (www.example.com) من أحد مزودي خدمة النطاقات مثل GoDaddy أو Namecheap.

٣- اختيار مزود خدمة استضافة : (Web Hosting Provider)  
الاستضافة هي خدمة تقدمها شركات لإتاحة مساحة تخزين على خوادمها  
لوضع ملفات الموقع .يُمكنك اختيار مزود استضافة مثل Bluehost أو  
HostGator أو SiteGround

٤- رفع الملفات إلى الخادم : (Uploading Files to Server) بعد شراء  
النطاق والاستضافة، يجب رفع ملفات الموقع إلى الخادم باستخدام  
بروتوكول نقل الملفات (FTP) أو باستخدام أدوات توفرها لوحة التحكم  
الخاصة بمزود الاستضافة) مثل cPanel أو Plesk)

٥- تهيئة إعدادات الموقع :ضبط إعدادات الموقع مثل روابط الصفحات  
وإعدادات الأمان والشهادة ل (SSL) من خلال لوحة تحكم الاستد

٦- اختبار الموقع : بـ الموقع، يجب اختبار جميع وظائفه  
الإنترنت للتأكد من عمله بشكل صحيح عبر جميع المتصفحات والأجهزة.  
٧- تحسين الموقع لمحركات البحث : (SEO) لضمان ظهور موقعك في  
نتائج محركات البحث مثل جوجل، يجب تحسين محتوى الموقع وعناوين  
الصفحات والكلمات المفتاحية وفقًا لممارسات تحسين محركات البحث  
(SEO).

٨- الترويج للموقع : استخدم وسائل التواصل الاجتماعي، والتسويق عبر  
البريد الإلكتروني، وغيرها من الطرق لجذب الزوار إلى موقعك.

### **أشهر برامج تحرير صفحات الويب شائعة الاستخدام تشمل:**

١- Adobe Dreamweaver : يعد من أشهر وأقوى برامج تحرير الويب،  
يوفر واجهة مرئية وتصميمية، بالإضافة إلى دعم تحرير الأكواد بلغات  
مختلفة مثل HTML, CSS, و JavaScript

- ٢- Visual Studio Code : محرر كود مفتوح المصدر وشائع الاستخدام يدعم العديد من لغات البرمجة، بما في ذلك HTML و CSS و JavaScript، ويمكن تخصيصه عبر إضافات متعددة.
- ٣- Sublime Text : محرر نصوص خفيف الوزن وسريع يدعم لغات برمجة متعددة، ويتميز بواجهة بسيطة وسهولة في الاستخدام.
- ٤- Atom: محرر نصوص مفتوح المصدر من GitHub ، يوفر العديد من الميزات للمطورين مثل الإضافات والإعدادات القابلة للتخصيص.
- ٥-++Notepad: محرر نصوص مجاني يدعم لغات برمجة متعددة، ويتميز بالبساطة والخفة في الأداء.
- ٦- Brackets: محرر مفتوح المصدر من Adobe ، مخصصًا لتطوير الـ JavaScript. ويدعم لغات مثل CSS, ML, و JavaScript.
- ٧- WebStorm: محرر قوي من JetBrains يركز على JavaScript وتطوير الويب الشامل، ويدعم العديد من الأطر والمنصات. كل من هذه البرامج يتميز بمزايا معينة ويختار المطورون البرنامج الأنسب بناءً على احتياجاتهم ومستوى خبرتهم.

## أساسيات ومعايير تصميم وبناء صفحات الويب وما يتعلق بذلك من مفاهيم ومصطلحات

-أساسيات ومعايير تصميم وبناء صفحات الويب:

- ١- الهيكل الأساسي (Structure): يعتمد على استخدام لغة HTML (Hypertext Markup Language) لإنشاء الهيكل العظمي للصفحة،

حيث يتم تحديد العناصر الأساسية مثل العناوين، الفقرات، القوائم، الروابط، والجداول.

٢- التنسيق (Styling): يُستخدم CSS (Cascading Style Sheets) لتحديد شكل ومظهر الصفحة، بما في ذلك الألوان، الخطوط، الأحجام، والتخطيطات. (Layouts)

٣- الاستجابة (Responsiveness): يجب أن تكون صفحات الويب قادرة على التكيف مع مختلف أحجام الشاشات والأجهزة، مما يتطلب تصميمًا متجاوبًا يستخدم شبكات مرنة (Flexible Grids) وصورًا قابلة للتكيف.

٤- إمكانية الوصول (Accessibility): تعني إمكانية الوصول إلى المحتوى لجميع المستخدمين، بـ ذوي الاحتياجات الخاصة. يجب تـ نصوص بديلة للصور (Alt) ، وتنظيم الصفحة بحيث يمكن فيها باستخدام لوحة المفاتيح، واستخدام ألوان تتناسب مع الأشخاص الذين يعانون من مشاكل في الرؤية.

٥- الأداء (Performance): يجب تحسين صفحات الويب لتحميل أسرع، وذلك عن طريق تقليل حجم الصور، وضغط الملفات، وتقليل عدد طلبات الخادم. (Server Requests)

٦- الأمان (Security): يجب تأمين صفحات الويب باستخدام بروتوكول HTTPS وتحديثات الأمان المستمرة لمنع الاختراقات والتهديدات.

٧- التفاعل (Interactivity): استخدام JavaScript ولغات البرمجة الأخرى لإضافة تفاعلية إلى الموقع، مثل النماذج، القوائم المنسدلة، وتأثيرات التحريك.

٨- تحسين محركات البحث (SEO) يتضمن استخدام كلمات مفتاحية مناسبة، ووضع العناوين بشكل صحيح، وتضمين الميتاداتا (Metadata) لتعزيز ظهور الصفحة في نتائج محركات البحث.

### المفاهيم والمصطلحات ذات الصلة:

- HTML لغة ترميز النص التشعبي: هي اللغة المستخدمة لبناء الهيكل الأساسي للصفحة.

- CSS أوراق الأنماط المتتالية: هي اللغة المستخدمة لتنسيق وتصميم مظهر الصفحات.

- JavaScript: لغة برمجة لإضافة التفاعل والوظائف المتقدم

- UX/UI تجربة المستخدم: (يتعلق بتصميم

المستخدم مع الموقع، بينما UI يهتم بتصميم العناصر التفاعلية والمرئية.

- Framework الإطار: مجموعة من الأدوات والمكتبات الجاهزة لتسهيل

عملية تطوير الويب، مثل Bootstrap و Angular و React.

- CMS نظام إدارة المحتوى: (مثل WordPress) يُستخدم لإنشاء مواقع الويب وإدارة المحتوى بسهولة.

### أنواع صفحات ومواقع الويب:

#### ١- الصفحات الثابتة (Static Web Pages)

هي صفحات ويب ثابتة لا تتغير إلا عند تعديلها من قبل المطور. تستخدم عادة لعرض معلومات ثابتة، مثل المواقع التعريفية أو الصفحات الشخصية.

#### ٢- الصفحات الديناميكية (Dynamic Web Pages)

تعتمد على التفاعل مع المستخدم وتعرض محتوى يتغير بناءً على تفاعل المستخدم أو مدخلات معينة. تُستخدم في المواقع التي تحتوي على قواعد بيانات، مثل المنتديات أو المتاجر الإلكترونية.

### ٣- مواقع الويب التفاعلية: (Interactive Websites)

توفر تجربة تفاعلية مع المستخدمين من خلال تقنيات متقدمة مثل الألعاب الإلكترونية، والخرائط التفاعلية، وتطبيقات الويب.

### ٤- مواقع التجارة الإلكترونية: (E-commerce Websites)

متخصصة في البيع والشراء عبر الإنترنت، مثل أمازون وعلي إكسپرس.

### ٥- المدونات: (Blogs)

مواقع تقدم محتوى منت  
شكل مقالات أو منشورات، تُستخدم  
الأفكار والخبرات الشخص  
قالات المتخصصة.

### ٦- المواقع التعليمية: (Educational Websites)

توفر موارد تعليمية، مثل الكورسات الإلكترونية والمقالات العلمية، مثل

Khan Academy و Coursera.

### ٧- مواقع التواصل الاجتماعي: (Social Media Websites)

تُستخدم لتواصل الأفراد وتبادل المعلومات والمحتوى، مثل فيسبوك وتويتر وإنستجرام.

### ٨- مواقع إخبارية: (News Websites)

متخصصة في تقديم الأخبار والمقالات الصحفية، مثل BBC و CNN.

### ٩- واقع الحافظة: (Portfolio Websites)

تُستخدم لعرض أعمال المحترفين أو الفنانين، مثل مواقع المصورين أو المصممين.

## اللغات المستخدمة في تصميم وبناء صفحات ومواقع الويب:

(١) HTML : لغة توصيف النص الفائق: هي الأساس لبناء هيكل صفحات الويب.

(٢) CSS: تنسيق صفحات الطرز المتراصة: تُستخدم لتنسيق وتجميل صفحات الويب، مثل تحديد الألوان، والخطوط، والتخطيطات.

(٣) JavaScript: تُستخدم لإضافة التفاعلية والوظائف الديناميكية إلى مواقع الويب، مثل التحقق من صحة البيانات وإضافة الرسوم المتحركة.

(٤) PHP: لغة برمجة تستخدم لبناء تطبيقات الويب الديناميكية وتعمل على جانب الخادم. تُستخدم في الغالب مع قواعد البيانات، مثل MySQL.

(٥) Python: تُستخدم لتطبيقات الويب باستخدام أطر عمل Django و Flask.

(٦) Ruby: تُستخدم مع إطار عمل Ruby on Rails لبناء تطبيقات الويب.

(٧) SQL: لغة الاستعلام البنيوية: تُستخدم للتعامل مع قواعد البيانات وإدارة البيانات في مواقع الويب الديناميكية.

(٨) XML و JSON: تُستخدم لتبادل البيانات بين الخادم والمتصفح.

(٩) أطر العمل (Frameworks) مثل Angular ، و React، و Vue.js: تُستخدم لتطوير تطبيقات الويب الحديثة وتسهيل عملية البرمجة.

(١٠) Node.js:

بيئة تشغيل لـ JavaScript تستخدم لبناء تطبيقات ويب عالية الأداء على جانب الخادم.

هذه هي بعض الأنواع واللغات الرئيسية المستخدمة في تصميم وبناء مواقع وصفحات الويب.



## المكونات والمواصفات والمعايير القياسية لتصميم صفحات الويب وتقييمها:

تصميم صفحات الويب يتطلب الاهتمام بمجموعة من المكونات والمواصفات والمعايير القياسية لضمان أن تكون الصفحات فعّالة وجذابة وسهلة الاستخدام. إليك بعض المكونات والمعايير الأساسية:

### ١- مكونات تصميم صفحات الويب:

- الواجهة الرسومية (UI) تشمل الألوان، الخطوط، الأيقونات، والرسومات التي تعزز تجربة المستخدم

- البنية (Layout): تنظي ي والعناصر على الصفحة بطريقة م ومنظمة.

- التفاعل (Interaction): كيفية تفاعل المستخدمين مع العناصر المختلفة على الصفحة، مثل الأزرار والروابط.

- المحتوى: النصوص، الصور، والفيديوهات التي تقدم المعلومات الأساسية وتساهم في تحقيق أهداف الموقع.

### ٢- المواصفات:

- قابلية الاستخدام (Usability): يجب أن تكون الواجهة سهلة الاستخدام والتنقل.

- التجاوب (Responsiveness): يجب أن تكون الصفحة ملائمة لجميع الأجهزة ( الكمبيوتر، الهواتف الذكية، والأجهزة اللوحية).

- الأداء: يجب أن تكون الصفحات سريعة التحميل وتعمل بكفاءة عالية.

### ٣- المعايير القياسية:

- معايير W٣C: التزام بمعايير W٣C الخاصة بـ HTML و CSS لضمان التوافق مع مختلف المتصفحات.

- معايير الوصول (Accessibility): الالتزام بمعايير الوصول لضمان إمكانية الوصول إلى الصفحات لجميع المستخدمين، بما في ذلك ذوي الاحتياجات الخاصة.

- تحسين محركات البحث (SEO): تطبيق أفضل الممارسات لتحسين تصنيف الصفحات في نتائج محركات البحث.

### ٤ - تقييم تصميم صفح:

- اختبارات المستخدم (User T) تقييم تجربة المستخدم من اختبارات فعلية مع المستخدمين.

- تحليل الأداء (Performance Analysis): استخدام أدوات تحليل الأداء لمراقبة سرعة التحميل وكفاءة الصفحة.

- مراجعة التوافق: التحقق من التوافق عبر متصفحات وأجهزة متعددة لضمان تجربة موحدة.

هذه العناصر تساهم في تحسين فعالية صفحات الويب وضمان تحقيق تجربة مستخدم متميزة.

### ما هي أشهر برامج وأدوات بناء وتصميم المواقع والصفحات؟

من أشهر برامج وأدوات بناء وتصميم المواقع والصفحات:

١-WordPress: منصة إدارة محتوى شائعة تتيح لك إنشاء مواقع ويب بسهولة باستخدام قوالب وإضافات.

٢-Wix: أداة إنشاء مواقع تعتمد على السحب والإفلات، مما يجعل التصميم سهلاً حتى للمبتدئين.

٣-Squarespace: أداة تصميم مواقع توفر قوالب احترافية وأدوات مدمجة لإنشاء مواقع أنيقة.

٤-Weebly: منصة أخرى تعتمد على السحب والإفلات توفر أدوات لتصميم مواقع الويب والمتاجر الإلكترونية.

٥-Adobe Dreamweaver: برنامج تصميم وتطوير الويب يوفر بيئة متكاملة لتصميم الصفحات وتحرير الكود.

٦-Bootstrap : إطار عمل مفتوح المصدر لتصميم واجهات المستخدم بسرعة وفعالية.

٧-Webflow : أداة تتيح لك إنشاء تصميمات تفاعلية الحاجة لكتابة كود.

٨-Joomla : نظام إدارة محتوى يوفر مرونة وقوة في بناء المواقع وتخصيصها.

٩-Drupal : نظام إدارة محتوى قوي ومفتوح المصدر، مثالي للمواقع الكبيرة والمعقدة.

هذه الأدوات توفر مجموعة متنوعة من الخيارات لتلبية احتياجات مختلفة في تصميم وتطوير المواقع.

## كيفية نشر الموقع الذى قمت بتصميمه على شبكة الإنترنت:

لنشر موقعك على شبكة الإنترنت، اتبع الخطوات التالية:

١- اختيار مزود خدمة استضافة: تحتاج إلى اختيار شركة استضافة ويب

تقدم خدمات استضافة لموقعك .يمكنك البحث عن مزودين مثل

HostGator ، SiteGround،Bluehost أو.

٢- حجز اسم نطاق :اختر اسم نطاق لموقعك وقم بتسجيله .يمكن عادةً شراء

اسم النطاق من نفس مزود خدمة الاستضافة أو من مسجلي النطاقات مثل

GoDaddy.

٣- تحميل الملفات : قم بتحميل ملفات موقعك إلى الخادم الخاص

بالاستضافة باستخدام FT مثل FileZilla) أو من خلال

التحكم الخاصة بالاست

٤- إعداد قاعدة البيانات (إذا لزم الأمر): إذا كان موقعك يحتاج إلى قاعدة

بيانات، قم بإنشاء قاعدة بيانات من خلال لوحة التحكم الخاصة

بالاستضافة وقم بتهيئتها وفقًا لاحتياجات موقعك.

٥- تكوين الموقع :تأكد من أن جميع إعدادات الموقع صحيحة وتعمل بشكل

سليم بعد تحميل الملفات وإعداد قاعدة البيانات.

٦- اختبار الموقع :قم بزيارة اسم النطاق الخاص بموقعك للتحقق من أن كل

شيء يعمل كما هو متوقع.

٧- تحديث محتوى الموقع :قم بتحديث الموقع بمحتوى حديث بانتظام لضمان

استمرارية جاذبيته وفعاليته.

إذا كنت بحاجة إلى مساعدة إضافية، يمكن لمزود خدمة الاستضافة عادةً

تقديم دعم فني لمساعدتك في

عملية النشر.

## أمثلة لمواقع الويب التعليمية:

هناك العديد من مواقع الويب التعليمية التي تقدم موارد تعليمية متنوعة للمساعدة في التعلم، ومن أبرزها:

١-Coursera: يقدم دورات عبر الإنترنت من جامعات ومؤسسات عالمية، تغطي مجموعة واسعة من الموضوعات.

٢-Khan Academy: يوفر دروساً مجانية في العديد من الموضوعات، من الرياضيات والعلوم إلى الفن والاقتصاد.

٣-edX: يقدم دورات وشهادات عبر الإنترنت من جامعات عالمية مثل MIT وهارفارد.

٤-Udemy: يحتوي على الدورات التي يغطيها مدرسون مستقلون مجموعة متنوعة من الـ

٥-Duolingo: تطبيق وموقع لتعلم اللغات بشكل ممتع وبمبسط.

٦-TED-Ed: يقدم فيديوهات تعليمية قصيرة حول مجموعة متنوعة من المواضيع، مع أسئلة وأنشطة تعليمية.

٧-FutureLearn: يقدم دورات عبر الإنترنت تغطي مواضيع متعددة من جامعات عالمية.

٨-Codecademy: موقع متخصص في تعلم البرمجة وتطوير المواقع.

## الباب الثاني

مفهوم البر  
الكائنية ومصطلحاتها  
الأساسية

## أهداف الباب الثاني

بعد الانتهاء من دراسة هذا الباب ينبغي أن يكون الطالب قادراً على أن:

- ١- يتعرف على مفهوم البرمجة الكائنية.
- ٢- يعدد اللغات البرمجية التي تدعم مفهوم البرمجة الكائنية.
- ٣- يعدد مزايا البرمجة الكائنية.
- ٤- يعدد عيوب البرمجة الكائنية.
- ٥- يتعرف على مفهوم الكائن.
- ٦- يتعرف على مفهوم الصنف.
- ٧- يفرق بين الكائنات Objects و الكلاسات classes .
- ٨- يفرق بمثال بين
- ٩- ينشئ كلاس class.
- ١٠- يخزن الكلاس بداخل متغير.
- ١١- يعطي مثال على تخزين الكلاس بداخل متغير.
- ١٢- يعطي مثال على كيفية مشاهدة محتوى الكلاس الذي تم تخزينه بداخل المتغير.
- ١٣- ينشئ ملف PHP بمسمى test.php وحفظه داخل مجلد باسم OOP، بالسيرفر المحلي بجهازه.
- ١٤- يُعرف property إلى الكلاس.
- ١٥- يعطي مثال على تعريف الخصائص إلى الكلاس.
- ١٦- يدرك فيما تستخدم الدالة echo.
- ١٧- يدرك فيما تستخدم Public.
- ١٨- يفرق بين الدالة Var\_dump و الدالة echo.

١٩- يدرك قوة OOP تظهر عند استخدام كلاس class واحد لإنشاء أكثر من كائن Object .

٢٠- ينشئ وظيفتين واحدة تقوم بتغيير قيمة الخاصية

property\$ والأخرى تقوم بإرجاع القيمة لطباعتها.

٢١- ينشئ كائنين، ويقوم بطباعة خصائصهم.

ومن ثم تعديلها، وطباعتهم مرة أخرى لمشاهدة الفرق.

٢٢- يعرف مفهوم البرمجة المعتمدة على النموذج الأولي Prototype.

٢٣- يعرف المجال Namespace في لغة الجافا سكريبت .

٢٤- ينشئ مجال namespace في جافا سكريبت

٢٥- يُعد الك ساسية المبنية داخل لغة الجافا سكريبت

٢٦- ينشئ ثولتين instances من الصنف،

بالاسم person١ والثانية بالاسم person٢ .

٢٧- يستخدم الفعل getProperty لقراءة قيمة الخاصية.

٢٨- يدرك متى يُستدعى المُشيد constructor .

٢٩- يحدد فيما تستخدم الوراثة في الجافا سكريبت.

٣٠- يُنشئ كائنًا عامًا global وبالاسم MYAP .

٣١- يوضح بمثال الصيغة المستخدمة في إنشاء مجال

namespace وإضافة متغيرات ودوال.

٣٢- يُعرف الصنف Student كصنف ابن

للصنف Person باستخدام لغة الجافا سكريبت.

٣٣- يعرف التغليف Encapsulation.

٣٤- يعرف التجريد Abstraction.





## صل الأول

### مفهوم البرمجة الكائنية

## مفهوم البرمجة الكائنية Object Oriented Programming

إن البرمجة الكائنية (OOP) ما هي إلا نمط برمجي يُستخدم التجريد في إنشاء نماذج/نسخ لتجسيد العالم الحقيقي، وتستخدم البرمجة الكائنية في ذلك أساليب متعددة من هذا النمط، فهي تستخدم الوحدات module ، وتعددية الأشكال polymorphism والتغليف encapsulation ، وتصدر الإشارة إلى أن معظم لغات البرمجة تدعم مفهوم OOP أمثال اللغات البرمجية: جافا، بايثون، روبي، جافا سكريبت.

وهي أسلوب جديد من أساليب البرمجة حيث أصبحت وحدة بناء البرنامج هي الصنف class والذي يحوي على ( البيانات data والعمليات والدوال functions)، عادة ما تكون البرامج من هذا الأسلوب معقدة بشكل كبير حيث تم تقسيم البرنامج إلى مجموعة من المهام الرئيسية ومن ثم تقسم إلى مهام فرعية على حسب التعقيد للمهام الرئيسية لذلك فإن الـ الهيكلية تنتهج النهج ( من الأعلى إلى الأسفل ) Top Down .

### ولها عدة مسميات منها:

- OOP وهي اختصار لكلمة (Object Oriented Programming).
- برمجة كينونية.
- برمجة شيئية المنحى.
- برمجة موجهة.
- برمجة كائنية.
- برمجة كائنية التوجه.
- برمجة غرضية التوجه.
- برمجة كائنية المنحى.
- برمجة بالعناصر.
- برمجة موجهة نحو الكائنات (أو العناصر).
- البرمجة بالكائنات.

والبرمجة الكائنية بكل بساطة هي أسلوب برمجي يقوم بجمع العمليات المتقاربة في كلاسات أو مصنفات **Classes** ليساعد في اتمام عملية إنشاء كود برمجي قوي و مرن و سهل التطوير بالمستقبل، مما يسمح للأكواد ان تكون تحت مبدأ لا تكرر (Don't Repeat Yourself (DRY)، وظيفة هذا المبدأ في تطوير البرمجيات ان يحد من عملية تكرار كتابة نفس الأكواد أكثر من مرة لنفس البرنامج، وفائدة هذا المبدأ تتضح في البرمجيات ذات المعمارية متعددة المستويات **multi-tier architectures** ، ومن أهم فوائد مبدأ **DRY Programming** عندما ترغب بتعديل معلومة او وظيفة في البرنامج ، فغالبا انك ستقوم بتعديل واحد لتحديث طريقة عمل الكود.

أحد أكبر مخاوف وهموم المبرمجين هي صيانة الأكواد بالأخص في البرمجيات التي تعرض البيانات بأكثر من مكان و بأكثر من طريقة بحيث لو اردت التغيير في هذا او طريقة عرضها سيتحول الخو كابوس في ملاحقة الأخ  
لاحها وبالأخص إذا كنت تستخدم  
مكررة لعرض نفس البيانات، والبرمجة الكائنية **OOP** تخوف منه كثير من مبرمجي **PHP** وخاصة أنه تم تقديمها بصور النسخة ٥ من **PHP** فبعد ان تعود مطوري **PHP** على الطرق التقليدية **procedural code** في البرمجة ويشاهد لأول مرة البرمجة الكائنية ويرى بنية جديدة للبرمجة فيعتقد أنها صعبة التعليم وليست بسهولة البرمجة التقليدية **procedural code** ولكن بالواقع البرمجة الكائنية واضحة جدا وأسهل من البرمجة التقليدية.

## **مزايا البرمجة الكائنية (OOP) : Features of (OOP)**

١-التجريد **abstraction** وهو عملية تحديد الخصائص والعمليات التي تنتمي لصنف معين وهي نوعان: -

أ-تجريد البيانات: **Data abstraction** وهي عملية التعرف على الخصائص المرتبطة بكائن معين.

ب-تجريد العمليات: **Methods abstraction** وهو عملية تحديد العمليات والإجراءات دون ذكر شيء عن كيفية أدائها.

٢-التغليف ( الكبسلة Encapsulation : هي عملية تجميع كل الخصائص properties والطرق أو العمليات Methods في وحدة واحدة ( داخل غلاف واحد ) حيث لا يمكن الوصول إليها ( أي الخصائص والطرق ) إلا عن طريق الكائن.

٣-إخفاء البيانات Data Hiding : وهي ميزة ناتجة عن كبسلة البيانات وتعني إضافة مستوى حماية معين على البيانات حتى نمنع وصول الخطأ إليها.

٤-الوراثة Inheritance : وهي أن يرث صنف ما الخصائص والعمليات الموجودة في الصنف الآخر مما يساعد على إعادة الاستخدام للأصناف التي تم إنشاؤها من قبل المستخدم .

٥-تعدد الأشكال (الأوجه polymorphisms : تسمح ميزة تعدد الأشكال لنفس الدالة أن تتعرف بصورة مختلفة في أصناف مختلفة.

ويمكن عمل ذلك ب الوراثة مع تعدد الأشكال.

واللغات التي تدعم أسلوب البرمجة الشيئية كثيرة نذكر منها:

C++ و Java و C# و Vb.net و PHP.

## عيوب البرمجة الكائنية:-

- ١- صعوبة فصل البيانات على العمليات.
- ٢- إعادة إنشاء الحلول وعدة إعادة استخدامها.

ل الثاني

الكائن Object والصنف class

## فهم الكائن Object والصنف class

قبل التعمق بعالم البرمجة الكائنية يجب عليك فهم الفرق بين الكائن Object و الكلاس Class و طرق إنشائهم وطرق بناء الخصائص Properties والأفعال Methods وطرق استخدامها.

### الفرق بين الكائنات Objects و الكلاسات classes

**تعريف مبسط:** الصنف Class يشكل هيكلية البيانات و الأفعال و يستخدم هذه المعلومات لبناء الكائن، والصنف Class عبارة عن وحدة تحوي مجموعة من البيانات الخاصة والعامة، بالإضافة إلى تعريف الدوال والعمليات (حيث يطلق على الدوال المعرفة داخل الصنف بـ "أعضاء دالية" member Function).

**مثال للتوضيح: الصنف** مثل مخطط بناء المنزل وظيفته المذ يحدد شكل المنزل على ورق مع تحديد علاقة جميع أجزاء المنزل ببعض بكل عناية وبخطة واضحة مع أن المنزل لم يبنى بعد.

الكائن: مثل البيت الذي تم بناؤه بعناية على تعليمات المخطط والبيانات التي تم تخزينها بهذا الكائن، مثل حديد، خشب، أسلاك وغيرها التي تكون المنزل قبل بنائه حسب المخطط أنها مجموعة من الأغراض ولكن عند جمعها ببعض تكون المنزل.

تستطيع بناء أكثر من كائن من نفس الكلاس وبنفس الوقت لا تربطهم أي علاقة ببعض فلو اتبعنا المثال السابق، نستطيع بناء ٥٠ منزل بمخطط واحد جميعهم يتشابهون بالشكل ولكن يختلفون بالعوائل التي تسكن بها والديكورات الداخلية على حسب استخدام كل عائلة.

### إنشاء الكلاس Class

عملية إنشاء كلاس واضحة جدا بكل سهولة تستطيع استخدام كلمة class متبوعة باسم الكلاس وقوسين معقوفة curly brackets {}.

مثال:

```
<?php

class SimpleClass
{
 // Class properties and methods can be declared
 here
}

?>
```

عند إنشاء الكلاس تستطيع عمل نسخة منه وتخزينه في متغير باستخدام الكلمة `new`

مثال:

```
$object = new SimpleClass;
```

لمشاهدة محتوى الكلاس المخزن في المتغير سوف نستخدم الدالة `var_dump`.

مثال:

```
var_dump($object);
```

الآن قم بتجميع هذه الأمثلة بملف بي اتش بي بمسمى `test.php` وقم بحفظه داخل مجلد باسم OOP، بالسيرفر المحلي بجهازك لتجربة الكود ليصبح الكود بالشكل التالي:

```
<?php

class SimpleClass
{
```



```
// Class properties and methods can be declared
here
}

$object = new SimpleClass;

var_dump($object);

?>
```

ثم قم بفتح الملف عن طريق المتصفح بزيارة  
<http://localhost/oop/test.php> العنوان  
سيكون الناتج كالتالي:

```
Object(SimpleClass)#1 (0){}
```

باتباعك لهذه الخطوات البسيطة قمت بعمل أول سكريبت OOP Script خاص بك.

## تعريف خصائص الكلاس Class Properties

المتغيرات التي يتم تعريفها داخل الكلاس تسمى خصائص Properties وتستخدم لإضافة البيانات إلى الكلاس، إنشائها وتعريفها نفس إنشاء المتغيرات العادية في لغة PHP ولكن تختلف أنها مرتبطة بالكائن الذي تم إنشائه بواسطة الكلاس حيث أنك لن تستطيع التعامل معهم إلا عن طريق الكائن نفسه.

لتعريف property إلى الكلاس SimpleClass قم بإضافة هذا الكود داخل الكود السابق

```
public $property = "I'm a class property!";
```

ليصبح الكود كالتالي:

```
<?php

class SimpleClass
{
 public $property = "I'm a class property!";
}

$object = new SimpleClass;

var_dump($object);

?>
```

يتم استخدام الكلمة **public** ديد مستوى رؤية الخصائص والأفعا  
تسمية الخاصية **Property** بنفس قوانين تسمية المتغيرات المتبعة  
في PHP، ويتم تعيين قيمة له .

#### ملاحظة:

ليس بالضرورة إدخال قيم للخصائص ولكن يمكن تركها فارغة، لقراءة  
محتوى الخاصية **Property** و طباعة محتواها للمتصفح تستطيع استخدام  
الدالة **echo** كما يلي:

```
echo $object->property;
```

**لاحظ هنا** قمنا بتحديد اسم الكائن ثم استخدمنا السهم -> وبعدها ادخلنا اسم  
الخاصية **Property** لطباعتها، وهنا تحديد اسم الكائن مطلب ضروري كما  
قلنا سابقا انه يمكن عمل أكثر من كائن من نفس الكلاس، و إذا لم نحدد اسم  
الكائن الذي نرغب بدخول احد خصائصه او افعاله ستحصل على خطأ حيث  
ان PHP، لن تستطيع تحديد الكائن الذي نرغب بدخول محتواه والتعامل  
معه، واستخدام علامة السهم -> وهي من أسس البرمجة الكائنية OOP  
بحيث انها تسمح لك بالدخول على خصائص **Properties** وأفعال  
**Methods** أي كائن.

الآن قم بالتعديل على الكود داخل الملف test.php واستبدل الدالة echo-var\_dump لقراءة وطباعة الخاصية Property بدل طباعة جميع محتوى الكلاس.

ليصبح الكود كالتالي:

```
<?php

class SimpleClass
{
 public $property = "I'm a class property!";
}

$object = new SimpleClass;

$object->property;

?>
```

الآن قم بزيارة رابط الملف عن طريق المتصفح سيظهر لك التالي:

I'm a class property!

## تعريف أفعال الكلاس Class Methods

الأفعال Methods وهي الدوال التي يتم تعريفها داخل الكلاس. بمعنى آخر أي عمل أو نشاط، يستطيع الكائن تنفيذه يتم تعريفه داخل الكلاس كأفعال Methods .

## مثال

سنقوم بإنشاء وظيفتين واحدة تقوم بتغيير قيمة الخاصية \$property والأخرى تقوم بإرجاع القيمة لطباعتها.

قم بإضافة التالي للكود السابق:

```
public function setProperty($newValue)
{
 $this->property = $newValue;
}

public function getProperty()
{
 return $this->property . "
";
}
```

ليصبح الكود كالتالي:

```
<?php

class SimpleClass
{
 public $property = "I'm a class property!";

 public function setProperty($newValue)
 {
 $this->property = $newValue;
 }

 public function getProperty()
 {
 return $this->property . "
";
 }
}
```

```
$object = new SimpleClass;
```

```
echo $object->property;
```

```
?>
```

## ملاحظة

للتعامل مع الخصائص **Properties** و الأفعال **Methods** خارج الكلاس كنا نستخدم اسم الكائن ثم السهم ثم اسم الخاصية او الفعل ولكن داخل الكائن نستطيع التعامل مع أي من الأفعال او الخصائص باستخدام **\$this** ثم السهم ثم اسم الفعل او الخاصية ، والسبب يعود ان اسم الكائن غير ثابت كما تعلمنا اننا نستطيع إنشاء أكثر من كائن لنفس الكلاس، و لتفادي مشكلة تعدد الاستخدام تم وضع مرجع ثابت داخل الكلاس يعود باسم الكائن وهو **\$this** .

الآن سنقوم بقراءة قيمة باستخدام الفعل **getProperty** و للمتصفح، ثم نقوم بتغيير الخاصية **\$property** باستخدام **setProperty**، ثم نقوم بطباعة القيمة مرة أخرى للمتصفح.

قم بتعديل الكود السابق باستبدال ليصبح كالتالي

```
<?php
```

```
class SimpleClass
```

```
{
 public $property = "I'm a class property!";
```

```
 public function setProperty($newValue)
 {
 $this->property = $newValue;
 }
```

```
 public function getProperty()
 {
 return $this->property . "
";
 }
```

```
}
```

```
$object = new SimpleClass;
```

```
echo $object->getProperty(); // Get the property value
and print it
```

```
$object->setProperty("I'm a new class property
value!"); // Set a new value
```

```
echo $object->getProperty(); // Read it out again to
show the change
```

```
?>
```

قم بزيارة الملف من المتصفح لمشاهدة الناتج وهو كالتالي:

I'm a class property!

I'm a new class property value!

## **قوة OOP تظهر عند استخدام كلاس class واحد لإنشاء أكثر من كائن Object**

يتصور مفهوم البرمجة الكائنية OOP البرنامج كتشكيكة من الأشياء/الكائنات المتعاونة/المترابطة بدلاً من أن يتصوره كتشكيكة من الدوال (functions) أو كسرد من الأوامر. ففي مفهوم OOP ، كل كائن/شيء له القدرة على استقبال الرسائل، ومعالجة البيانات، وإرسال الرسائل إلى باقي الكائنات، ويمكن اعتبار أنه لكل كائن object كينونة خاصة به ودور/وظيفة مستقلة عن الكائن الآخر.

وتعزز البرمجة الكائنية القدرة على صيانة الشيفرة البرمجية والمرونة في التطوير، وأثبتت جدارتها على نطاق واسع في هندسة البرمجيات الكبيرة، ولأن البرمجة الكائنية تشدد على استخدام الوحدات module ، فإن الشيفرة/الكود المكتوب بمفهوم البرمجة الكائنية هو أبسط في التطوير وأسهل في الفهم مستقبلاً (عند التنقيح والتعديل)، وكما يعزز مفهوم البرمجة الكائنية التحليل المباشر للشفرة، وفهم الحالات الشائكة فهماً أفضل من باقي الأساليب البرمجية الأخرى.

### مثال:

سنقوم بتعديل بسيط على ملف test.php ، ونقوم بإنشاء كائنين، ونقوم بطباعة خصائصهم ومن ثم تعديلها، وطباعتهم مرة أخرى لمشاهدة الفرق.

قم بتغيير محتوى الملف      التالي:

```
<?php

class SimpleClass
{
 public $property = "I'm a class property!";

 public function setProperty($newValue)
 {
 $this->property = $newValue;
 }

 public function getProperty()
 {
 return $this->property . "
";
 }
}

// Create ٢ Objects From The Same Class
$object = new SimpleClass;
```

```

$object2 = new SimpleClass;

// Print The Value of $property Of Both Objects
echo $object->getProperty();
echo $object2->getProperty();

// Set New Values For Both Objects
$object->setProperty("I belong to the First instance!");
$object2->setProperty("I belong to the Second
instance!");

// Print The Value of $property Of Both Objects Again
echo $object->getProperty();
echo $object2->getProperty();

```

الآن توجه للمتصفح لمشاهدة الناتج الذي سيكون كالتالي:

I'm a class property!

I'm a new class property value!

I belong to the First instance!

I belong to the Second instance!

كما شاهدت في البرمجة الكائنية لا توجد علاقة بين الكائن الأول والثاني مع أن منشئهم كلاس واحد مما يجعل من السهل صيانة وتطوير الكود في المستقبل فكل ما كان الكلاس مصمم بذكاء ومرونة يسمح لك بإنشاء كائنات قوية مثل مخطط البيت كلما كان أفضل وأقوى ومصمم بذكاء كان المنزل الذي بني عليه أفضل وأقوى ويكون من السهل تطويره في المستقبل.



## البرمجة المعتمدة على النموذج الأولي Prototype

البرمجة المعتمدة على النموذج الأولي (Prototype-based programming) ما هي إلا نموذج من البرمجة الكائنية OOP ولكنها لا تستخدم الأصناف classes ، بل تقوم أولاً بإعداد سلوك أي صنف class وما ومن ثم تعيد استخدامه، ويُطلق البعض على هذا النموذج: البرمجة بلا أصناف classless ، أو البرمجة المَبْدئية المنحى prototype-oriented ، أو البرمجة المعتمدة على الأمثلة (instance-based).

يعود أصل اللغة المعتمدة على النموذج الأولي إلى لغة Self ، والتي طورها David Ungar و Randall Smith ، ولكن أسلوب البرمجة بدون أصناف class-less توسع ونال شهرة كبيرة في العقد الأخير، واعتمد من قبل العديد م البرمجية أشهرهم جافا سكريبت.

## الفصل الثالث

### البرمجة الكائنية باستخدام جافا سكريبت

## البرمجة الكائنية باستخدام جافا سكريبت

### المجال Namespace في جافا سكريبت

المجال هو أشبه بمستوعب/بحاوية (container) تسمح للمطور في تحزيم وظائف تحت اسم فريد، أو اسم تطبيق محدد، ففي جافا سكريبت المجال هو مجرد كائن object كأي كائن آخر يحتوي على طرق methods ، وخصائص properties ، وحتى كائنات objects.

**ملاحظة هامة:** من المهم جدًا الانتباه إلى أنه في جافا سكريبت، لا يوجد فرق بين الكائنات العادية والمجالات namespaces ، وهذا يختلف عن اللغات الكائنية الأخرى، الأمر الذي قد يُربك المبرمجين المبتدئين في جافا سكريبت.

إن إنشاء مجال namespace في جافا سكريبت بسيط للغاية، فمن خلال إنشاء كائن عام/مشتق global ، ستصبح جميع المتغيّرات variables والطرق methods ، والدوال functions خاصيات لهذا الكائن، ويُقلل استخدام المجالات namespaces أيضًا من احتمالية تضارب الأسماء في التطبيق، منذ أن كل كائن من كائنات التطبيق ما هي إلى خاصيات كائن شامل/عام معرّفة على مستوى التطبيق.

### مثال:

سُنشئ في الخطوة التالية كائنًا عامًا global وبالاسم MYAPP

```
// global namespace
var MYAPP = MYAPP || {};
```

يظهر في المثال السابق، كيف تم التأكد أولاً فيما إذا كان MYAPP معرفًا (سواء في نفس الملف أو في آخر)، ففي حال الإيجاب سُنستخدم الكائن العام MYAPP ، وفي حال عدم تعريفه مُسبقًا سُنشئ كائنًا خالٍ وبالاسم MYAPP والذي سيغلف encapsulate الطرق methods والدوال functions والمتغيرات variables والكائنات Objects . كما يمكن أيضًا إنشاء مجال فرعي: sub-namespaces

```
// sub namespace
MYAPP.event = {};
```

يوضح المثال التالي الصيغة المستخدمة في إنشاء مجال **namespace** وإضافة متغيرات ودوال:

```
// Create container called MYAPP.commonMethod
for common method and properties
MYAPP.commonMethod = {
 regExForName: "", // define regex for name
 validation
 regExForPhone: "", // define regex for phone no
 validation
 validateName: function(name){
 // Do something with name, you can access
 regExForName variable
 // using "this.regExForName"
 },

 validatePhoneNo: function(phoneNo){
 // do something with phone number
 }
}

// Object together with the method declarations
MYAPP.event = {
 addListener: function(el, type, fn) {
 // code stuff
 },
 removeListener: function(el, type, fn) {
 // code stuff
 },
 getEvent: function(e) {
 // code stuff
 }

 // Can add another method and properties
}
```

```
// Syntax for Using addListener method:
MYAPP.event.addListener("yourel", "type", callback);
```

## الكائنات الأساسية/القياسية المبنية داخل لغة جافا سكريبت

### Standard built-in objects

تتضمن لغة جافا سكريبت العديد من الكائنات في تركيبتها، على سبيل المثال، يوجد كائنات مثل `String`، `Array`، `Object`، `Math`، ويُظهر المثال التالي كيفية استخدام الكائن `Math` للحصول على رقم عشوائي باستخدام أحد طرق `method` هذا الكائن وهي الطريقة `Random()`.  
`console.log(Math.random());`

**ملاحظة:** يفترض المثال السابق وجميع الأمثلة التالية وجود دالة `function` بالاسم `console.log` معرفة تعريفاً عاماً (`globally`)، مع العلم أن هذه الدالة ليست جزء من اللغة نفسها، ولكنها دالة متوفرة في العديد من متصفحات الإنترنت لأغراض تشخيص الشفرة البرمجية `debugging`.

كل كائن في جافا سكريبت هو حالة/أمثلة `instance` من الكائن `Object` ويرث كافة خاصياته `properties` وطرقه `methods`.

### الكائنات الخاصة

#### الصنف (الفئة)

لغة جافا سكريبت لغة من النوع `prototype-based` ولا تحتوي على العبارة `class` كما هو حال باقي لغات البرمجة الكائنية، كما في روبي أو بايثون، ويُربك هذا الأمر المبرمجين المعتادين على اللغات التي تعتمد على هذه العبارة أو المفهوم، وتستخدم جافا سكريبت بدلاً من ذلك النوال `functions` لمحاكاة مفهوم الأصناف `classes`، وتعريف صنف هو بسهولة تعريف أي دالة:

```
var Person = function () {};
```

## الكائن أمثلة الصنف (class instance)

يتطلب إنشاء حالة/أمثلة instance جديدة من كائن object استخدام العبارة new object ، وتعيين النتيجة إلى متغير بغرض الوصول إليه فيما بعد.

مثال:

عرف في الشفرة السابقة صنف class بالاسم Person ، وفي الشفرة التالية، سينشئ حالتين/أمثولتين instances من هذا الصنف، الأولى بالاسم person١ والثانية بالاسم person٢.

```
var person١ = new Person();
var person٢ = new Person();
Please see Object.create() for a new, additional,
instantiation method that creates an uninitialized
nce.
```

## المُشيد The constructor

يُستدعى المُشيد constructor في لحظة الاستهلال instantiation اللحظة التي يُنشئ فيها الكائن)، والمُشيد ما هو إلا طريقة method من طرق الصنف class ، وفي جافا سكريبت تعمل الدالة على تشييد الكائن، ولذلك لا داعي إلى تعريف طريقة method من أجل عملية التشييد، وكل إجراء مصرّح في الصنف class يُنفذ في لحظة الاستهلال instantiation.

يُستخدم المُشيد في تعيين خاصيات properties الكائن، أو في استدعاء طرق methods معينة لتحضير الكائن للاستخدام، و يحدث ذلك باستخدام صيغة syntax مختلفة .

تُظهر الشفرة التالية كيف يُسجل log يُرسل رسالة نصية إلى طرفية المتصفح (console) مُشيد الصنف Person رسالة نصية حينما يُستهل .  
Instantiated

```
var Person = function () {
```

```
console.log('instance created');
};
```

```
var person١ = new Person();
var person٢ = new Person();
```

## الخاصية The property خاصية الكائن object

الخصائص properties ما هي إلا متغيرات محتواه في الصنف class ، وكل حالة/أمثلة من الكائن تمتلك هذه الخصائص، وتُعيّن الخصائص في دالة مُشيد الصنف بحيث تُنشئ مع كل حالة/أمثلة instance .

إن الكلمة المفتاحية **this** ، والتي تُشير إلى الكائن الحالي، تسمح للمطور بالعمل مع الخصائص من ضمن الصنف، والوصول (قراءةً وكتابةً) إلى الخاصية **property** خارج الصنف يكون من الصيغة **Instance.Property** كما هو الأمر في لغة ++. بلس بلس و **Java** والعديد من اللغات الأخرى، ومن داخل الصنف تُستخدم الصيغة **this.Property** للحصول على قيمة الخاصية أو لتعيين قيمتها.

في الشفرة التالية، عُرِّفت الخاصية **firstName** للصنف **Person** وفي لحظة الاستهلال: **instantiation**

```
var Person = function (firstName) {
 this.firstName = firstName;
 console.log('Person instantiated');
};
```

```
var person١ = new Person('Alice');
var person٢ = new Person('Bob');
```

```
// Show the firstName properties of the objects
console.log('person١ is ' + person١.firstName); // logs
"person١ is Alice"
console.log('person٢ is ' + person٢.firstName); // logs
"person٢ is Bob"
```

## الطرق The methods

الطرق methods ما هي إلا دوال (وتُعرّف كما تعرّف الدوال functions)، فيما عدا ذلك فهي تُشبه الخاصيات properties ، واستدعاء طريقة method مشابه إلى الوصول إلى خاصيّة ما، ولكن مع إضافة () في نهاية اسم الطريقة، وربما مع مُعطيات arguments ، ولتعريف طريقة، تُعيّن دالة إلى خاصيّة مُسمّات من خاصيّة الصنف prototype ، ويُمكن فيما بعد استدعاء الطريقة على الكائن بنفس الاسم الذي عُيّن للدالة.

### مثال:

في الشفرة التالية، عُرّفت ومن ثم أُستُخدمت الطريقة sayHello() للصنف Person .

```
var Person = function (firstName) {
 .firstName = firstName;

 Person.prototype.sayHello = function() {
 console.log("Hello, I'm " + this.firstName);
 };

 var person١ = new Person("Alice");
 var person٢ = new Person("Bob");

 // call the Person sayHello method.
 person١.sayHello(); // logs "Hello, I'm Alice"
 person٢.sayHello(); // logs "Hello, I'm Bob"
```

إن الطرق methods في جافا سكريبت ما هي إلا دالة كائن عادية مرتبطة مع كائن كخاصيّة property ، وهذا يعني أنه يُمكن استدعاء الطرق خارج السياق، كما في المثال التالي:

```
var Person = function (firstName) {
 this.firstName = firstName;
```



```

};

Person.prototype.sayHello = function() {
 console.log("Hello, I'm " + this.firstName);
};

var person١ = new Person("Alice");
var person٢ = new Person("Bob");
var helloFunction = person١.sayHello;

// logs "Hello, I'm Alice"
person١.sayHello();

// logs "Hello, I'm Bob"
person٢.sayHello();

// logs "Hello, I'm undefined" (or fails
// with a TypeError in strict mode)
helloFunction();

// logs true
console.log(helloFunction === person١.sayHello);

// logs true
console.log(helloFunction ===
Person.prototype.sayHello);

// logs "Hello, I'm Alice"
helloFunction.call(person١);

```

كما يظهر المثال السابق، جميع الحالات المستخدمة في استدعاء الدالة `sayHello` تُشير إلى نفس الدالة سواء الاستدعاء من خلال `person١` أو `Person.prototype` أو حتى في المتغير `helloFunction` وقيمة `this` خلال استدعاء الدالة يعتمد على الكيفية التي تُستدعى فيها، حيث تُشير الكلمة المفتاحية `this` إلى الكائن الحالي الذي تُستدعى عليه الطريقة `method`، بمعنى عندما تم استدعاء

الطريقة (`sayHello()`) على الكائن `person١` فإن `this` تشير إلى الكائن `person١` ، وعند استدعاء `sayHello` على الكائن `person٢` فإن `this` تشير إلى الكائن `person٢` ، ولكن إن تم الاستدعاء بطريقة مختلفة، فإن `this` ستُعيّن تعيينًا مختلفًا، فاستدعاء `this` من المتغيّر (كما في `helloFunction()`) سيُعيّن `this` إلى الكائن العام (`global`) والذي سيكون `window` في متصفح الإنترنت)، ومنذ أن هذا الكائن (على الأغلب) لا يملك الخاصيّة `firstName` ، ستكون النتيجة كما هو الحال في المثال السابق "Hello I'm undefined"، كما يمكن دائماً تعيين `this` صراحة باستخدام `Function#call` أو `Function#apply` وهو كما كان في نهاية المثال.

## الوراثة

تُستخدم الوراثة في جافا سكريبت في إنشاء صنف `class` كمثيل مخصص لصنف أو أكثر **تدعم كريبب وراثه وحيدة فقط gle** **inheritance**، ويُطلق على الصنف المخصص عادةً **ابن (child)** ، ويُطلق على الصنف الآخر عادةً **الأب (parent)** ، وفي جافا سكريبت يتم ذلك من خلال إسناد حالة/أمثولة من الصنف الأب إلى الصنف الابن، ومن ثم تخصيصه، وفي متصفحات الإنترنت الحديثة يمكن استخدام `Object.create` في تحقيق الوراثة `inheritance` أيضاً.

### ملاحظة:

لا تتفقد جافا سكريبت مُشيّد صنف، الابن `prototype.constructor` (راجع `Object.prototype`)، وعليه يجب التصريح عن ذلك يدوياً.

### مثال:

عرف في الشفرة التالية الصنف `Student` كصنف ابن للصنف `Person` ، ومن ثم أعد تعريف الطريقة (`sayHello()`) وأضيفت الطريقة (`sayGoodBye()`) علاوة على ذلك.

```
// Define the Person constructor
```

```

var Person = function(firstName) {
 this.firstName = firstName;
};

// Add a couple of methods to Person.prototype
Person.prototype.walk = function(){
 console.log("I am walking!");
};

Person.prototype.sayHello = function(){
 console.log("Hello, I'm " + this.firstName);
};

// Define the Student constructor
function Student(firstName, subject) {
 // all the parent constructor, g sure (using
 // call)
 // that "this" is set correctly during the call
 Person.call(this, firstName);

 // Initialize our Student-specific properties
 this.subject = subject;
}

// Create a Student.prototype object that inherits from
// Person.prototype.
// Note: A common error here is to use "new
// Person()" to create the
// Student.prototype. That's incorrect for several
// reasons, not least
// that we don't have anything to give Person for the
// "firstName"
// argument. The correct place to call Person is
// above, where we call
// it from Student.

```

```
Student.prototype =
Object.create(Person.prototype); // See note below
```

```
// Set the "constructor" property to refer to Student
Student.prototype.constructor = Student;
```

```
// Replace the "sayHello" method
Student.prototype.sayHello = function(){
 console.log("Hello, I'm " + this.firstName + ". I'm
studying "
 + this.subject + ".");
};
```

```
// Add a "sayGoodBye" method
Student.prototype.sayGoodBye = function(){
 console.log("Goodbye!");
};
```

```
// Example usage:
var student = new Student("Janet", "Applied
Physics");
student.sayHello(); // "Hello, I'm Janet. I'm
studying Applied Physics."
student.walk(); // "I am walking!"
student.sayGoodBye(); // "Goodbye!"
```

```
// Check that instanceof works correctly
console.log(student instanceof Person); // true
console.log(student instanceof Student); // true
```

فيما يخص السطر `(Student.prototype) = Object.create` في الإصدارات القديمة من جافا سكريبت `(Person.prototype)` ; والتي لا تدعم `Object.create` يمكن إما استخدام بعض الحيل في خداع المتصفحات —هذه الخدع معروفة إما بالاسم `polyfill` أو `shim` أو استخدام دالة تحقق نفس النتيجة كما في المثال التالي:

```
function createObject(proto) {
 function ctor() {}
 ctor.prototype = proto;
 return new ctor();
}
```

// Usage:

```
Student.prototype = createObject(Person.prototype);
```

التأكد من أن `this` تشير إلى الكائن المطلوب بغض النظر عن كيف للكائن أن يُستهل يمكن أن يكون صعبًا، ومع ذلك يوجد صياغة أبسط من شأنها أن تسهّل الأمر.

```
var Person = function(firstName) {
 if (this instanceof Person) {
 is.firstName = firstName;
 } else {
 return new Person(firstName);
 }
}
```

## التغليف Encapsulation

ليس بالضرورة أن يعلم الصنف `Student` كيف تمّ تنفيذ/تعريف الطريقة `walk()` للصنف `Person` لكي يستطيع استخدام تلك الطريقة، ولا يحتاج الصنف `Student` إلى تعريف تلك الطريقة صراحةً إلا إذا كان المطلوب التعديل عليها، ويُطلق على هذا الإجراء مفهوم التغليف `encapsulation`، والذي فيه يحزم كل صنف البيانات والطرق `methods` داخل وحدة/كينونة وحيدة.

إخفاء المعلومات سمة شائعة في باقي اللغات البرمجية وعادةً ما توجد كخاصيات/كطرق إما بالاسم `private` أو `protected`، وعلى الرغم من أنه يُمكن ممانلة/محاكاة ذات الأمر في جافا سكريبت، إلا أن هذا الأمر ليس مطلبًا من متطلبات البرمجة الكائنية.

## التجريد Abstraction

التجريد ما هو إلا ميكانيكية تسمح للمطور في تجسيد جانب من المشكلة التي يعمل عليها، إما من خلال الوراثة inheritance التخصيص specialization أو التركيب composition ، وتُحقق جافا سكريبت التخصيص من خلال الوراثة، والتركيب من خلال السماح لحالات/أمثولات الصنف لتكون قيمًا لخاصيات attributes الكائنات الأخرى.

الصنف Function في جافا سكريبت يرث من الصنف Object وهذا يوضّح التخصيص في هذا النموذج، والخاصية Function.prototype ما هي إلا حالة/أمثلة من الصنف Object وهذا يوضّح جزئية التركيب composition .

```
var foo = function () {};
```

```
s "foo is a Function: true"
console.log('foo is a Function: ' + (foo instanceof
Function));
```

```
// logs "foo.prototype is an Object: true"
console.log('foo.prototype is an Object: ' +
(foo.prototype instanceof Object));
```

## تعددية الأشكال Polymorphism

كما أن جميع الطرق methods والخاصيات properties معرّفة ضمن الخاصية prototype ، فيمكن لبقية الأصناف أن تُعرّف طرقًا methods بنفس الاسم، وستكون الطرق في نطاق الصنف الذي عُرفت به، إلا إذا كان الصنفان على علاقة من نوع أب وابن parent-child ، بمعنى آخر أحد الصنفان يرث من الآخر.



## الباب الثالث

# مقدمة في لغة PHP



## أهداف الباب الثالث

بعد الانتهاء من دراسة هذا الباب ينبغي أن يكون الطالب قادراً على أن:

- ١- يعرف لغة PHP.
- ٢- يحدد ما يحتاج إليه لتشغيل PHP بالنسبة لمستخدمي Windows.
- ٣- يحدد ما يحتاج إليه لتشغيل PHP بالنسبة لمستخدمي Apple.
- ٤- يحدد ما يحتاج إليه لتشغيل PHP بالنسبة لمستخدمي Linux.
- ٥- يعرف Wamp Server.
- ٦- يعدد مميزات برنامج Wamp Server.
- ٧- يشرح طريقة برنامج Wamp Server.
- ٨- يثبت برنامج Wamp Server على جهازه.
- ٩- يحفظ ملف PHP.
- ١٠- يحدد أي الصيغتين التاليتين صحيحة :

<http://localhost/index.php>

<c:/wamp/www/test١.php>

## عند الإقلاع إلى سكريبتات PHP.

- ١١- يميز الصيغة الصحيحة للإقلاع إلى السكريبتات.
- ١٢- يدرك الأخطاء التي من الممكن أن يقع بها عند الإقلاع إلى السكريبتات.

# صل الأول

## ما هي لغة PHP

## ما هي لغة PHP؟

لغة PHP هي لغة الأوامر الأكثر استخداماً على الويب. انها تستخدم في تحسين صفحات الويب. باستخدام لغة PHP، يمكنك أداء العديد من المهام مثل انشاء صفحات لادخال اسم المستخدم وكلمة المرور، فحص التفاصيل الواردة في استمارة، انشاء منتديات، معارض للصور، دراسات، وغير ذلك الكثير.

وتعرف لغة PHP بأنها لغة مدعمة للسيرفر. ذلك لأن هذه اللغة لا يتم تنفيذها على جهاز الحاسوب الخاص بك، ولكن على الحاسوب الذي طلبت الصفحة منه. ويتم عرض النتائج في المستعرض الخاص بك. عندما تتعلم لغة PHP فانك لا تحتاج أن تتعلم اللغات الأخرى التي ربما تكون سمعت عنها مثل ASP، Python، و لغة Perl.

وتعتمد لغة PHP، على لغة القبلية للنص الفائق. ، ترجع التسم الإصدار القديم من البرنامج الذي كان يطلق عليه أدوات الصفحة الرئيسية الشخصية (Personal Home Page Tools). ولذلك حصلت على المصطلح PHP.

## ما تحتاج اليه لتشغيل PHP؟

قبل الشروع في كتابة واختبار أوامر PHP، لابد أولاً من أن يتوافر لديك خادم Server. اما أن تحصل على بعض المساحة من شركة مستضافة التي تدعم لغة PHP، أو تقوم بجعل الحاسوب الخاص بك يعمل كسيرفر. ذلك لأن لغة PHP لا يتم تشغيلها على الجهاز الخاص بك وانما يتم تنفيذ الأوامر على السيرفر. والنتائج يتم ارسالها الى الجهاز العميل Client وهو جهاز الحاسوب الخاص بك.

هناك برنامج يدعى Wampserver ، هذا البرنامج يسمح لك بتنفيذ أوامر PHP على جهازك، انه مثبت كل ما تحتاج اليه اذا كنت تستخدم نظام التشغيل Windows. وهو اختصار لـ Windows Apache MySQL PHP، فوائده عديدة أهمها إعداد بيئة شبيهة ببيئة السيرفرات المستضافة للمواقع، لتجربة الاسكربتات قبل رفعها.

## مستخدمى Apple

بالنسبة لمستخدمى Apple، فان برنامج MAMP هو الأفضل. فهو يشبه برنامج Wampserver، انه يقوم بتهيئة كل ما تحتاج اليه ويمكنك تحميله من خلال الرابط التالي:

<https://www.mamp.info/en/>

كل ما تقوم به هو الحصول على أباتشي السيرفر وتشغيله، ولذلك فانه يمكنك من تنفيذ أوامر PHP دون الاتصال بالانترنت. انتبه الى المكان الذي يتم تخزين الملفات به ، وكذلك عنوان "Local host".

## مستخدمى Linux

هناك القليل من المواقع التي تساعد مستخدمى نظام التشغيل Linux للعمل بالأباتشي سيرفر ولغة PHP، لا يوجد وسيلة سهلة مثل ما هو الحال بالنسبة لمستخدمى نظامي التشغيل Windows, Mac. ستكتب ذلك فقط في جوجل Google

Lamp +Ubuntu

Or

Lamp +Debian

كما يوجد القليل من الفيديوهات على اليوتيوب، التي ربما تساعدك على تثبيت كل ما تحتاج اليه.

## مستخدمي Windows

بالنسبة لمستخدمي Windows ، يمكنهم تثبيت برنامج WAMP أولاً.

### تعريف السيرفر المحلي: Apache

السيرفر المحلي Apache ، عبارة عن برنامج تستطيع من خلاله تحويل جهاز الكمبيوتر لديك إلى سيرفر، و تركيب ما تشاء عليه من السكريبتات والصفحات التجريبية، بهدف فحصها و إصلاح ما بها من عيوب، قبل رفعها إلى الإنترنت. توجد العديد من البرامج التي تقوم بتحويل جهاز الكمبيوتر إلى سيرفر محلي (EasyPHP – XAMPP ...) من أشهرها برنامج الـ WAMP Server.

### ما هو MP Server

عبارة عن برنامج لعمل سيرفر شخصي على جهازك، و هو اختصار لـ Windows Apache MySQL PHP ، فوائده عديدة أهمها إعداد بيئة شبيهة ببيئة السيرفرات المستضيفة للمواقع، لتجربة الاسكريبتات قبل رفعها.

### مميزات برنامج WAMP Server

السماح بالانتقال عبر الإنترنت غير المتصل.

التعامل مع اصدارات متعددة من الـ Apache ، PHP ، MySQL.

إدارة الإعدادات الخاصة بالشبكة.

### • شرح تثبيت برنامج WAMP Server

لتثبيت برنامج الـ WAMP Server على جهازك، قم أولاً بتحميل آخر نسخة من البرنامج .

قم بتحميل النسخة المناسبة لنظام التشغيل لديك، إما ٣٢ Bits ، أو ٦٤ Bits .

بعد تنزيل البرنامج على جهازك، ابدأ عملية التنصيب بالضغط Double Click على أيقونة البرنامج.



شكل (٢-١) يوضح رمز البرنامج

عملية الإعداد لتنصيب السيرفر بدأت، اضغط Next للمتابعة. واتبع المراحل التالية:



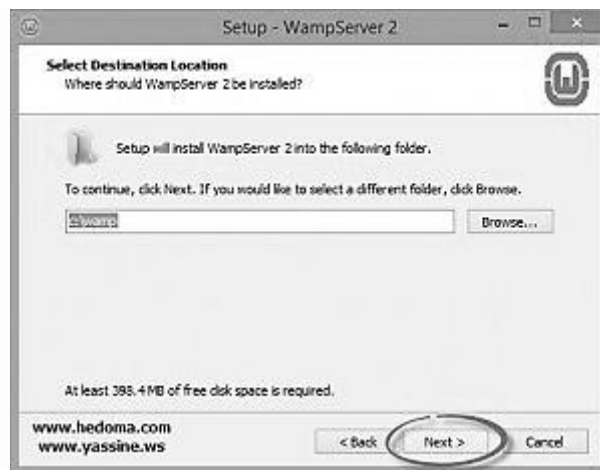
شكل (٢-٢) يوضح الخطوة رقم ٢ في معالج اعداد Wampserver.

أشر على الخيار I accept the agreement ، للموافقة على شروط استخدام البرنامج، ثم اضغط Next.



شكل (٢-٣) يوضح النافذة التي تتطلب الموافقة على شروط استخدام برنامج Wampserver.

قم بتحديد المسار الذي ت  
يب ملفات البرنامج عليه (يفضل تر  
هو)، ثم اضغط Next.



شكل (٢-٤) يوضح النافذة التي تتطلب المسار الذي تريد تنصيب البرنامج عليه .

اضغط على التالي Next



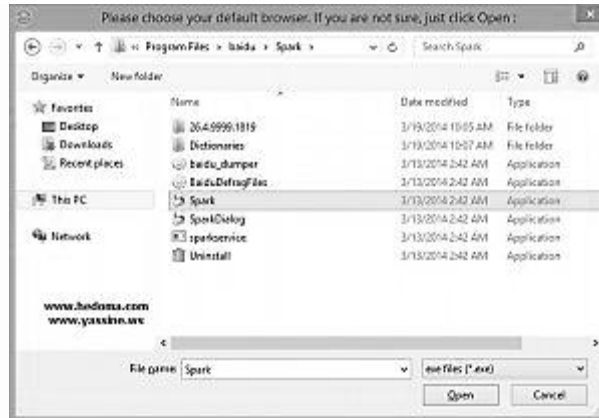
شكل (٥-٢) يوضح النافذة التي تسأل عن اختيار اظهار أيقونة البرنامج على سطح المكتب .



شكل (٦-٢) يوضح النافذة التي تدل على أن البرنامج جاهز للتثبيت عند الضغط على Install .

نختار المتصفح الذي نرغب في جعله افتراضيا للسيرفر شخصيا  
استخدم Spark .





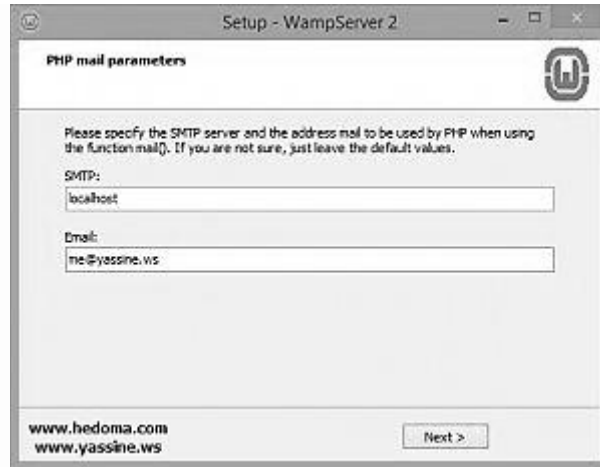
شكل (٧-٢) يوضح النافذة التي تسألك عن اسم المتصفح الذي ستجعله افتراضياً للسيرفر.

اضغط على access ن النافذة التي تظهر:



شكل (٨-٢) يوضح تحديد نوع الشبكة واختيار السماح بالوصول Allow .access

أدخل بريدك الإلكتروني في النافذة التي تظهر:



شكل (٩-٢) يوضح النافذة التي تطلب ادخال بريدك الالكتروني.



شكل (١٠-٢) يوضح اكتمال عملية التثبيت لبرنامج Wampserver.

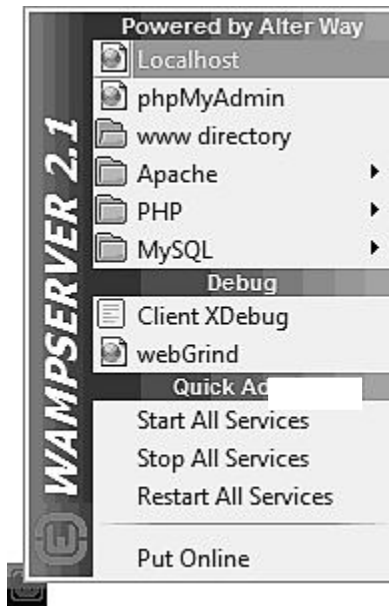
مباشرة بعد الضغط على Finish ، سوف تظهر أيقونة صغيرة بشريط المهام تحوي جميع إعدادات البرنامج.

، وفي الفصل التالي سترى وتؤكد من أن كل شيء قد تم تشغيله واعداده باستخدام Wampserver.

## الفصل الثاني

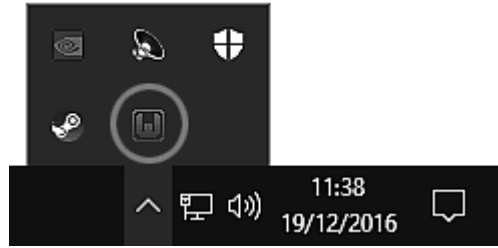
### اختبار برنامج Wampserver

بعد تحميل وتنصيب برنامج Wampserver. هذا سوف يمنحك سيرفر على جهازك الشخصي (لمستخدمي النوافذ). حيث يمكنك اختبار السكريبتات الخاصة بك، اذا نجحت عملية التنصيب سوف تجد أيقونة بجوار أيقونة الساعة في أقصى اليمين أسفل الشاشة.



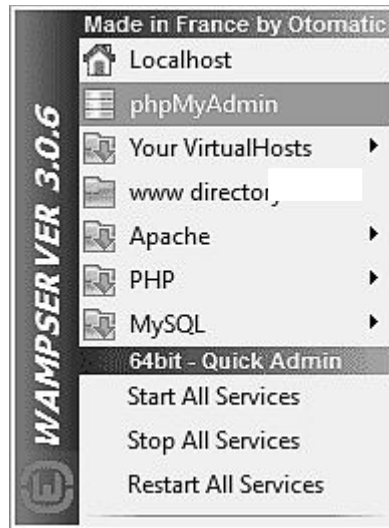
شكل (١١-٢) يوضح القائمة التي تظهر عند الضغط على أيقونة برنامج Wampserver

اضغط على هذه الأيقونة لترى القائمة الموضحة أعلى (في الإصدارات القديمة لبرنامج Wampserver). في ويندوز ١٠ (Windows ١٠)، اضغط ضغطاً مزدوجاً على الأيقونة Wampserver الموجودة على سطح المكتب بعد التنصيب. سترى حينئذ مربع أخضر في شريط المهام، حيث توجد الساعة. اذا كنت لا ترى المربع الأخضر، اضغط على السهم الأبيض لتراه.



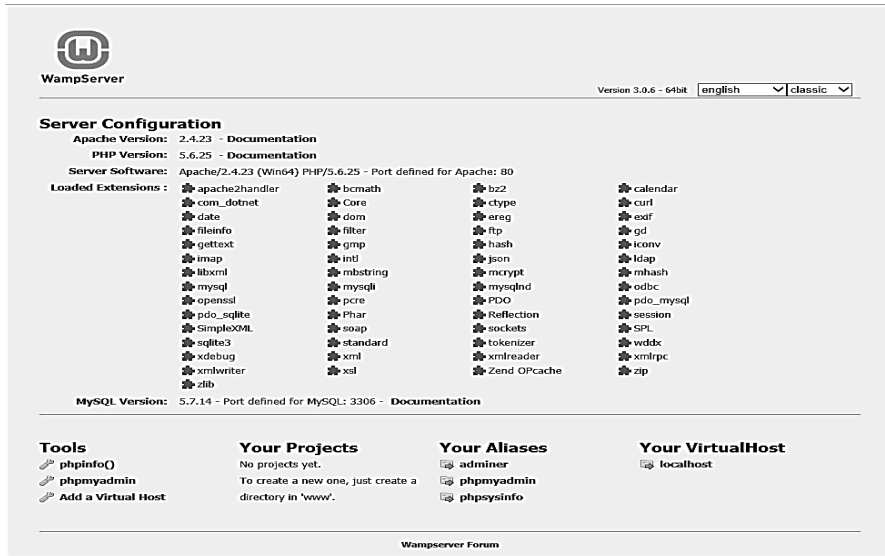
شكل (٢-١٢) يوضح المربع الأخضر لرمز البرنامج في شريط المهام.

والشكل التالي يوضح الإصدار رقم ٣,٠,٦ من برنامج Wampserver على Windows ١٠ ٦٤ bit:



شكل (٢-١٣) يوضح الإصدار رقم ٣,٠,٦ من برنامج Wampserver على Windows ١٠ ٦٤ bit

من هنا، يمكنك إيقاف السيرفر، أو الخروج منه، عرض ملفات المساعدة، ورؤية الصفحات المكونة. اضغط على Localhost ، وسوف تظهر هذه الصفحة ( تشير Localhost الى السيرفر الذي يعمل على جهازك، وهناك طريقة أخرى للإشارة الى السيرفر الخاص بك وتكون باستخدام عنوان IP التالي ١٢٧,٠,٠,١).



شكل (٢-١٤) يوضح النافذة التي تظهر عند الضغط على Localhost

اضغط على الرابط أسفل أدوات Tools والذي .Phpinfo()

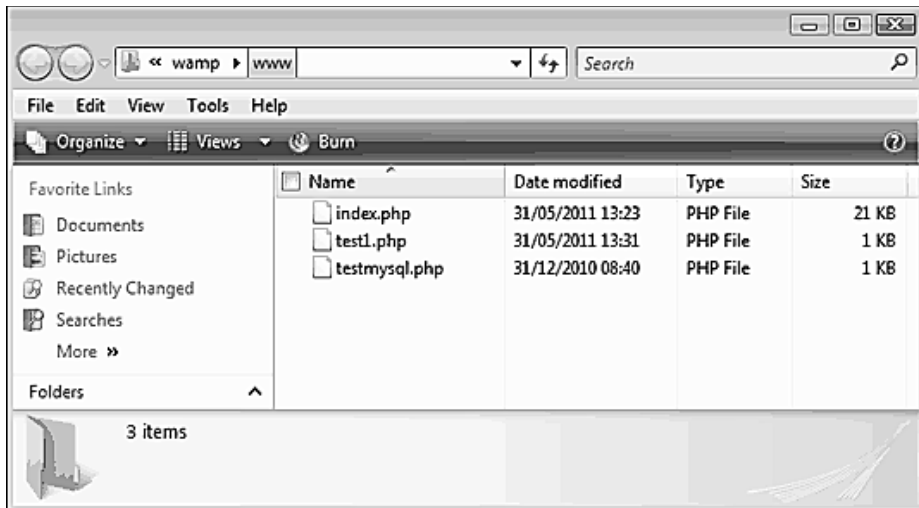
## حفظ ملفات PHP

عندما تقوم بإنشاء صفحة PHP جديدة، فأنك تحتاج إلى حفظها في المجلد WWW، ويمكنك رؤية أين هذا من خلال الضغط على البند الخاص بها كما هو موضح من القائمة التالية:



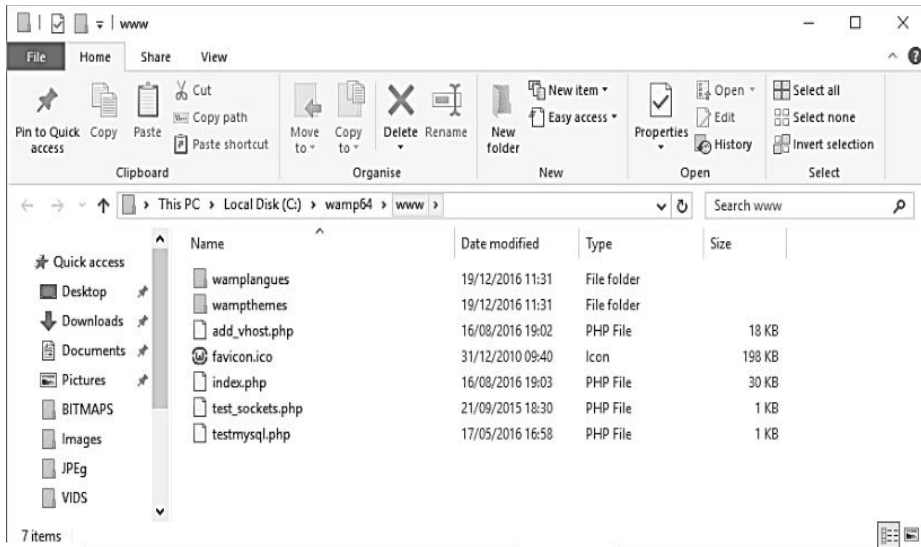
شكل (٢-١٥) يوضح اختيار المجلد WWW من القائمة.

عند الضغط على البند WWW d y، يمكنك رؤية نافذة مستعرض و النافذة التالية من Windows Vista (من المحتمل أن تجد ملفين فقط هما (index, testmysql).



شكل (٢-١٦) يوضح نافذة المستعرض التي تظهر عند الضغط على البند المجلد WWW.

وهذه هي المجلدات التي تظهر في Windows ١٠ (لاحظ أنه يوجد الكثير من الملفات في الإصدارات الحديثة من Wampserver).



شكل (١٧-٢) يوضح نافذة المستعرض التي تظهر عند الضغط على البند المجلد WWW في Windows ١٠.

هذا المجلد WWW لبرنامج Wampserver عادة ما يكون على الموقع التالي على القرص الصلب لديك:

**c:/wamp/www/**

وفي هذا الموقع للاصدارات الحديثة من برنامج Wampserver ونظم التشغيل:

**c:/wamp٦٤/www/**

### ملحوظة هامة

ضع في اعتبارك دائماً، عندما تضغط على File>Save As لحفظ السكريبتات الخاصة بلغة PHP: احفظهم بداخل المجلد WWW (لو أنك



تستخدم نظامي Mac أو Linux ، فانك ربما تحتاج الى حفظ ملفات PHP على المجلد htdocs بدلاً من WWW).

## الإقلاع الى سكريبتات PHP

افترض أنك أنشأت سكريبت باستخدام PHP يدعي test\php. للاقلاع الى هذا السكريبت، فانك تحتاج الى إضافة اسم السكريبت بعد السيرفر المحلي Localhost في المستعرض الخاص بك. ولذلك بدلاً من ذلك:

**<http://localhost/index.php>**

فانك ستكتب هذا:

**<http://localhost/test\php>**

انك لا تكتب اسم مجلد Wamp. فان ذلك يمثل خطأ، على سبيل المثال هكذا:

**<c:/wamp/www/test\php>**

وكذلك اذا كتبت هكذا:

**<http://localhost/www/test\php>**

السيرفر الخاص بك يعرف أين يكون مجلد WWW، لذلك فانك لا تحتاج الى كتابته، ولكن فقط عليك إضافة اسم السكريبت الى السيرفر المحلي Localhost. وبالمثل، لو أنك أنشأت مجلد تحت المجلد WWW، فانك ستكتب هذا:

**<http://localhost/folder name/script name.php>**

-



## الباب الرابع

# المتغيرات في لغة PHP

## أهداف الباب الرابع

في نهاية هذا الباب ينبغي أن يكون الطالب قادراً على أن:

- ١- يعرف المتغيرات.
- ٢- يذكر شروط تسمية المتغيرات في PHP.
- ٣- يعطي أمثلة على المتغيرات في PHP.
- ٤- يعدد الأخطاء التي من المحتمل أن يقع فيها عند كتابة الكود.
- ٥- يعرف كيفية كتابة اسم المتغير في لغة PHP.
- ٦- يخزن قيمة رقمية داخل متغير في لغة PHP.
- ٧- يكتب صيغة للجمع في PHP.
- ٨- يعطي مثال على      يم التي تحملها المتغيرات في PHP.
- ٩- يخزن قيمة نصية داخل متغير في PHP.
- ١٠- يعطي مثال على تخزين قيمة نصية داخل متغير في PHP.
- ١١- يميز الصياغة الخاطئة لتخزين قيمة نصية داخل متغير.
- ١٢- يعدد الأخطاء التي من الممكن أن يقع بها عند تخزين قيمة داخل متغير.
- ١٣- يدرك كيف يميز المستعرض نوع السكريبت.
- ١٤- يحدد فيما يستخدم الأمر Print.
- ١٥- يعرف بديل الأمر Print لعرض الأشياء على الصفحة.
- ١٦- يضيف تعليق في كود PHP.
- ١٧- يتعامل مع المتغيرات.
- ١٨- يكتب سكريبت ويقوم بتنقيذه.
- ١٩- يطبع نص على الصفحة بعد تخزين النص داخل متغير.
- ٢٠- يربط بين نص مباشر واسم متغير.

- ٢١ يربط بين متغيرين.
- ٢٢ يشرح كيفية الجمع في PHP.
- ٢٣ يعطي أمثلة على عملية الجمع في PHP.
- ٢٤ يشرح كيفية الطرح في PHP.
- ٢٥ يعطي أمثلة على عملية الطرح في PHP.
- ٢٦ يخلط بين عمليتي الجمع والطرح في مثال واحد.
- ٢٧ يدرك أهمية الأقواس في تغيير نتائج العمليات في يشرح كيفية الجمع في PHP.
- ٢٨ يشرح كيفية الضرب في PHP.
- ٢٩ يعطي أمثلة على عملية الضرب في PHP.
- ٣٠ يوضح بـ ية استخدام الأقواس في اجبار PHP النتائج.
- ٣١ يشرح كيفية إجراء عملية القسمة في PHP.
- ٣٢ يعطي أمثلة على عملية القسمة.
- ٣٣ يعلن عن عدد عشري.
- ٣٤ يدرك كيفية إجراء العمليات المختلفة على الأعداد العشرية.

## ل الأول

ما هو المتغير وكيفية تخزين القيم  
الرقمية داخل المتغيرات

المتغير هو حجز مساحة تخزينية. فأنت تضع الأشياء في أماكن أو مساحات مخصصة لتخزينها ( المتغيرات ) وهذا يمكنك من استخدامهم والتعامل معهم في البرنامج الخاص بك. والأشياء التي تريد أن تخزنها هي عبارة عن أرقام ونصوص.

إذا لم تتضح لديك فكرة المتغيرات، افترض أن لديك أشياء متعددة وتريد ترتيبها لوضعها في مكان ما، لذلك فإنك تحتاج الى مساعدة من شخصين ليصبحوا المساحة التخزينية التي تريدها. انهم سيمسكون بالأشياء لك، بينما تقوم أنت بترتيبها وتحديد ما تريده. هؤلاء الأشخاص يمثلون المتغيرات.

فتستطيع على سبيل المثال أن تعد كم معطفاً تمتلكه، وتعطيها للشخص الأول. يمكنك أن تعد الأحذية التي تمتلكها وتعطيها للشخص الآخر، ولكن السؤال هنا هو أي من هذان الشخصان يمسك المعطف وأيهم يمسك بالحذاء؟ إذا كانت الذاكرة لديك ضعيفة، فإن ما يساعدك على التذكر هو إعطاء أسماء لهؤلاء الأشخاص مثل:

mr coats  
mrs shoes

ولكن التسمية التي تعطيها للأشخاص ترجع اليك (المتغيرات). فإذا رغبت في تسميتها هكذا فلتفعل:

man coats  
woman shoes

أو

HimCoats  
HerShoes

ولكن بما أن الذاكرة ضعيفة، انه من الأفضل أن تعطيهم أسماء تساعدك على التذكر، ماذا يحمل كلاً منهما لك؟ ( ولكن هناك بعض الأشياء يرفض الأشخاص الحاق أسمائهم بها. فالمتغيرات كذلك ترفض أن توضع الشرطة التحتية في بداية الاسم ( \_ )، أو أن يبدأ الاسم برقم. ولكن معظم الحروف تكون مقبولة.

كل شخص الآن (المتغيرات) أصبح لديه اسماً. ولكن ليس من الجيد اعطاؤهم فقط أسماء. انهم سيقومون بأداء مهمة محددة، ولذلك لا بد أن نخبرهم بما سيفعلون. الأول سيمسك المعاطف، ولكن يمكننا تحديد كم عدد المعاطف التي سوف يمسك بها. فلو أن لديك عشرة من المعاطف ستعطيهم إياه، ولذلك يكون الاخبار بهذه الطريقة:

**mr\_coats = ١٠**

لذلك، فإن اسم المتغير يأتي أولاً، ثم علامة يساوي (=)، انك في هذه الحالة تخبر المتغير بما سيفعله، وهو أن يمسك الرقم ١٠، في هذه الحالة فان) علامة (=) لا تعني يساوي)، وانما يطلق عليها معامل التخصيص. ولا بد من استخدام علامة (=) لتخزين القيم داخل المتغيرات.

علاوة على ذلك فإن لغة PHP، تفتقد شيئين هنا، الأول، أن المتغيرات تحتاج الى علامة الدولار (\$) في بدايتها ، ولذلك فإن اسم المتغير الأول يكون هكذا:

**\$mr\_coats = ١٠**

لو أنك نسيت علامة الدولار (\$)، فإن الأشخاص سيرفضون العمل! ولكن الشيء الثاني المفقود هو حقاً دقيق ألا وهو الفاصلة المنقوطة (;). السطور في أكواد PHP، تحتاج الى كتابة الفاصلة المنقوطة (;) في نهايتها هكذا:

**\$mr\_coats = ١٠;**

إذا تلقيت رسالة خطأ عندما تحاول تنفيذ الكود، فإن أول ما عليك فعله هو فحص كم عدد الفاصلة المنقوطة التي تم نسيان كتابتها في نهاية السطر. انها عملية سهلة ولكنها قد تكون محبطة، ولكن اذا بحثنا عن الشيء الآخر وهو التأكد من وجود علامة الدولار (\$) الموجودة في بداية أسماء المتغيرات.

وعند الإعلان عن المتغير الآخر، يمكننا القيام بنفس الخطوات هكذا:

**\$mrs\_shoes = ٢٥;**

من السطر السابق يتضح لنا أن \$mrs\_shoes تمتلك القيمة ٢٥. إذا أردنا أن نجمع كم عدد العناصر التي يمتلكها الشخص كلها، يمكننا وضع متغير جديد (لاحظ علامة الدولار في بداية الاسم للمتغير الجديد) هكذا:



## \$total clothes

يمكننا بعد ذلك إضافة عدد المعاطف الى عدد الأحذية، فيكون الجمع في لغة PHP، بهذه الطريقة:

**\$total clothes = \$mr coats + \$mrs shoes;**

تذكر أن \$mr coat يحمل القيمة ١٠، وأن \$mrs shoes تحمل القيمة ٢٥. اذا استخدمت علامة الجمع (+)، فان PHP يعتقد أنك تريد اجراء عملية الجمع، ولذلك فانه يقوم بحساب الناتج من أجلك. ويتم تخزين الناتج في المتغير الجديد الذي أطلقنا عليه \$total clothes، ويمكنك أيضاً الجمع بهذه الطريقة:

**\$total clothes = ١٠ + ٢٥;**

لغة PHP، تعتبر أن علامة الجمع(+) تستخدم لاجراء عملية الجمع، ومن ثم يمكنك جمع أكثر من عنصرين كما يلي:

**\$total clothes = ١٠ + ٢٥ + ٧ + ٣٨ + ١٢٥٠;**

الفكرة واحدة وهي إضافة علامة (+) بين القيم التي تريد جمعها، ثم تخزين الناتج بداخل متغير ، ذلك المتغير الذي يكتب على يسار علامة يساوي (=).

في الفصل التالي سنري كيفية تخزين النصوص داخل المتغيرات.

## الفصل الثاني

**تخزين النصوص داخل المتغيرات في لغة PHP**

**Putting Text into Variables in PHP**

في الفصل السابق، رأينا كيف يمكن تخزين القيم الرقمية داخل المتغيرات. ولكن يمكنك أيضاً أن تقوم بتخزين النصوص داخل المتغيرات. افترض أنك تريد أن تعرف شيئاً ما عن المعاطف التي تمتلكها. هل هي معاطف شتوية؟ جاكيتات؟ معاطف صيفية؟ يمكنك أن تقرر ثم تقوم بتخزين القيمة النصية كما قمت سابقاً بتخزين القيمة الرقمية هكذا:

$\$coats_1 = "Winter Coats";$

أذكر مرة أخرى بأن اسم المتغير لا بد أن يبدأ بعلامة الدولار (\$) ثم تعطيه اسماً وصفيّاً يذكرك بالمهمة التي تريده أن يؤديها  $Coats_1$ . وعلامة يساوي (=) تتبع اسم المتغير. بعد علامة يساوي (=) مباشرة، اكتب القيمة النصية المراد تخزينها وهي  $Winter coats$ . ولكن لاحظ أن علامات التنصيص المزدوجة تحيط بالقيمة النصية. وإذا لم تحيط النص المباشر بهذه العلامات ("" ) علامات التنصيص المزدوجة، ستظهر لك رسائل أخطاء، وعلى الرغم من ذلك يمكنك استخدام علامات الاقتباس المفردة بدلاً من العلامات المزدوجة كما

$\$coats_1 = 'Winter Coats';$

ولكن لا يمكنك كتابتها هكذا:

$\$coats_1 = "Winter Coats";$

في السطر السابق، لقد بدأنا بعلامة تنصيص مفردة وانتهينا بعلامتين تنصيص مزدوجة، وهذا سيسبب خطأ عند تنفيذ البرنامج.

يمكنك تخزين قيم نصية أخرى بنفس الطريقة:

$\$coats_2 = "Jackets";$

$\$coats_3 = "Summer Coats";$

سيتم تخزين القيمة النصية داخل المتغير الموجود على يسار علامة يساوي(=).

لذلك فالمتغيرات كما قلنا سابقاً هي مساحات تخزينية. تستخدم هذه المساحات التخزينية للاحتفاظ بالأشياء مثل القيم الرقمية والنصية. ستستخدم المتغيرات

بكثرة، وسنرى في الفصل التالي بعض التدريبات على كيفية استخدام المتغيرات.

## الفصل الثالث

### بعض التدريب على استخدام المتغيرات لغة PHP

### Some Practice with PHP Variables

في الفصل السابق، تعلمنا أن المتغيرات ما هي الا مساحات تخزينية للاحتفاظ بالأشياء مثل القيم الرقمية والنصية. كأنك تخبر PHP أن يتذكر هذه القيم التي تريد أن تستخدمها فيما بعد لأداء مهام عليها. أما في هذا الفصل سنتناول بعض التدريبات على استخدام المتغيرات.

## التعامل مع المتغيرات في لغة PHP:

سيتم عرض النتائج في صفحة الويب. باستخدام محرر النصوص مثل برنامج Notepad، أو برنامج PHP، اكتب ما يلي (يمكنك نسخ ولصق الأكواد، اذا كنت تفضل ذلك، ولكن تعلم أولاً كيف تكتبها بنفسك، حتى تتعلم من الأخطاء!)

```
<html>
 <_____d>
<title>Variables - Some Practice</title>
</head>
<body>
```

```
<?php print("It Worked!"); ?>
```

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

عندما تنتهي من كتابة هذه السطور، احفظ الصفحة باسم Variables.php. ثم قم بتنفيذ السكريبت.

**تذكر:**

عندما تحفظ عملك، احفظه على المجلد WWW، كما سبق وأوضحنا. ولتنفيذ الصفحة، ابدأ بتشغيل المستعرض الخاص بك ثم اكتب هذا في شريط العنوان:

<http://localhost/variables.php>

لو أنك أنشأت مجلد بداخل المجلد WWW، فإن العنوان الذي يتم كتابته في المستعرض سيكون شبيهاً الى حد ما بما يلي:

<http://localhost/FolderName/variables.php>

إذا نجحت في كتابة الكود بطريقة صحيحة، ستري النص "it worked!" معروضاً في المستعرض، وفي هذه الحالة فإن السيرفر الخاص بك يعمل، إذا كنت تستخدم برنامج wampserver، ستري الأيقونة الخاصة به في أقصى اليمين من أسفل الشاشة. اضغط على الأيقونة ثم اختار Start All Services من القائمة.

سكريبت PHP عبارة عن سطر واحد فقط وهو:

<?php print("It Worked!"); ?>

أما بقية السكريبت فهو كود HTML عادي. لقد تم وضع كلمة PHP في الجزء BODY من ص HTML. يمكنك أيضاً كتابة السكريبت استخدام أي HTML. ولكن قبل أن يتعرف المستعرض على السكريبت، فإنه يحتاج الى بعض المساعدة. فعليك أولاً اخباره بنوع السكريبت. فالمستعرضات تتعرف على PHP بالبحث عن علامة الترقيم (التي يطلق عليها المعرفات أو الصياغة) التالية:

<?php ?>

لذلك فانك تحتاج الى قوس الزاوية اليسار (<) ثم علامة الاستفهام (?). بعد علامة الاستفهام، اكتب PHP (بحروف كبيرة أو صغيرة). بعد الانتهاء من كتابة السكريبت، اكتب علامة استفهام أخرى. وفي النهاية، فانك تحتاج الى قوس زاوية يمين (>). يمكنك وضع مسافات كما تريد بين فتح واغلاق الصياغة.

لعرض الأشياء على الصفحة، تم استخدام Print(). ما تريد من المستعرض أن يطبعه يتم وضعه بين القوسين. لو أنك تريد أن تطبع نص مباشر، فانك تحتاج الى علامات التنصيص (المفردة أو المزدوجة)، لطباعة ما يشتمل عليه المتغير يكفي فقط بكتابة اسم المتغير (مشتماً على علامة الدولار \$). وفي النهاية ينتهي سطر الكود كما هو معتاد بالفاصلة المنقوطة (;). وهناك طريق آخر لعرض الأشياء على الصفحة وهي استخدام بديل لـ Print() - echo().

دع ما هو مكتوب بـ HTML، ولكن غير فقط في PHP هكذا من:

```
<?php print("It Worked!"); ?>
```

الى:

```
<?php
```

```
print("It Worked!");
```

```
?>
```

انه ليس تغييراً جوهرياً! لكن اتساع الكود عن سطر واحد يجعل من السهل أن تفهم ما ستؤديه من مهام. الآن، انه من الواضح أنه يوجد سطر واحد فقط في الكود – Print. ولذلك أضف السطر التالي الى الكود:

```
?php
```

```
$test String = "It Worked!";
```

```
print("It Worked!");
```

```
?>
```

لقد استخدمنا متغير يدعى \$test String. وبعد علامة يساوي تم كتابة " It Worked". وتم انهاء السطر باستخدام الفاصلة المنقوطة (;). لا تقم بتنفيذ السكريبت بعد. غير السطر الذي يحتوي على Print الى هذا:

```
print($test String);
```

ثم قم بإضافة بعض التعليقات هكذا:

```
<?php
```

```
//-----TESTING VARIABLES-----
```

```
$test String = "It Worked!";
```

```
print($test String);
```



>?

التعليقات في PHP مفيدة لك. فهي تساعدك على تذكر ما هو متوقع من الكود أن يؤديه. ويمكن كتابة التعليق بإضافة الشرطتين المائلتين //. حيث أن هذا يخبر PHP أن يتجاهل بقية السطر. بعد الشرطتين المائلتين، يمكنك كتابة أي شيء تريده. وهناك طريقة أخرى لكتابة التعليق، تكون كما يلي:

<?php

/\* -----TESTING VARIABLES-----

Use this type of comment if you want to spill over to more than one line.

Notice how the comment begin and end.

\*/

\$test String = "It Worked!";

print(\$test String);

>?

أي طريقة تختارها، تأكد من أنك قد أضفت تعليقاً الى الكود: فانها بالفعل تساعدك. خاصة اذا كنت سترسل الكود الخاص بك الى شخص آخر.

لكن الآن يمكنك تنفيذ السكريبت المكتوب أعلاه، وتجربته.

كيف توصلت اليه؟ ستري نفس النص بالضبط قد تمت طباعته في الصفحة. كل ما حدث هو أنك قمت بتخزين قيمة نصية داخل متغير، ثم بعد ذلك تطلب من PHP أن يطبع المتغير. وهذه خطوة متقدمة عليك الآن أن تؤدي بعض التدريبات بنفسك!.

## تدريب

غير النص "It Worked" الى أي شيء آخر تريده. ثم قم بتنفيذ السكريبت مرة أخرى. حاول أن تكتب بعض الأرقام بين علامات التنصيص المزدوجة، بدلاً من النص.

## تدريب

غير علامتي التنصيص المزدوجة الى علامتي التنصيص المفردة. هل يؤدي ذلك الى حدوث أي تأثير؟ ضع علامة اقتباس مفردة في بداية النص، وعلامات تنصيص مزدوجة في النهاية. ماذا يحدث عندما تقوم بتنفيذ الكود؟

## تدريب

احذف علامة الدولار (\$) من اسم المتغير. ثم قم بتنفيذ الكود. ما هي رسالة الخطأ التي تظهر لك؟ قم بكتابة علامة الدولار (\$) مرة أخرى، ولكن احذف الفاصلة المنقوطة (;). قم بتنفيذ الكود مرة أخرى؟ ما هي رسالة الخطأ التي تحصل عليها هذه المرة؟ حاول أن تتذكر هذه الأخطاء فسوف تراهم كثيراً عندما تبدأ في العمل! لو أنك رأيتهم في المستقبل، فسيكون من الأفضل أن تكون قادراً على تصحيح أخطأك.

في الفصل التالي سنتناول تدريبات أكثر على استخدام المتغيرات.

ل الرابع

تدريبات أكثر عن المتغيرات في PHP

**More PHP Variable Practice**

في الفصل السابق، بدأنا التعامل مع المتغيرات. وكانت النتائج تظهر في صورة نصوص على الصفحة. وفي الفصل التالي سنتناول العديد من التدريبات للتعامل مع المتغيرات، وكيفية إجراء عمليات الجمع باستخدام PHP.

الآن يمكنك طباعة نص على الصفحة، دعنا نجرب بعض الأرقام. ابدأ بطباعة Basic Page مرة أخرى بـ PHP ، واحفظ العمل باسم 2.Variable.php:

```
<html>
<head>
<title>More on Variables</title>
</head>
<body>

 <?php

 print ("Basic Page");

 >

</body>
</html>
```

سنقوم الآن بأعداد متغير وطباعته على الصفحة. لذلك غير الكود الى ما يلي:

```
<?php

 $first_number = ١٠;
 print ($first_number);

?>
```

كل ما على الكود فعله أو القيام به هو طباعة محتويات المتغير الذي يدعى \$first number. تذكر: أنه لطباعة نص مباشر فانك تحتاج الى علامات التنصيص، لو أنك تريد طباعة المتغير في هذه الحالة لا تحتاج الى علامات

التنصيب، لترى لماذا، نفذ السكريبت الموجود أعلاه. ثم قم بتغيير سطر الطباعة Print الى ما يلي:

print ("first number");

وبعبارات أخرى، قم بإضافة علامات التنصيب المزدوجة حول اسم المتغير. هل هذا يسبب اختلافاً؟ ما النتائج التي تتوقع طباعتها؟ الآن قم بتغيير العلامات المزدوجة الى علامات تنصيب مفردة، وأعد تنفيذ السكريبت مرة أخرى. بعلامات الاقتباس المزدوجة، فإن الرقم ١٠ لا يزال مطبوعاً، ولكن مع استخدام علامات التنصيب المفردة سيتم طباعة اسم المتغير!

الفصل الخامس

الربط في PHP

**PHP Concatenation**

يمكنك ربط نص مباشر، بما هو موجود في أي متغير أياً كان. النقطة full stop، تستخدم لذلك. افترض أنك تريد أن تطبع ما يلي "My variable contains the value of ١٠". في PHP يمكنك أداء ذلك كما يلي:

<?php

```
$first_number = ١٠;
$direct text = 'My variable contains the value
of ';
print($direct text . $first_number);
```

?>

لذلك فإن لدينا اثنين من المتغيرات. المتغير الجديد يحتفظ بالنص المباشر. عندما نريد طباعة محتويات كلا المتغيرين، فإن النقطة تستخدم لفصل المتغيرين. قم بتدقيق كودك السابق، ولاحظ ماذا يحدث احذف النقطة ثم حاول أن تجرب الكود مرة أخرى. هل ظهرت لك أي أخطاء؟

يمكنك أيضاً أن تستخدم هذه الطريقة:

<?php

```
$first_number = ١٠;
print ('My variable contains the value of ' .
$first_number);
```

?>

في هذه المرة، فإن النص المباشر لم يتم تخزينه بداخل متغير، ولكن تم تضمينه فقط في جملة الطباعة Print. إذن يمكن أن تستخدم النقطة full stop للفصل بين النص المباشر واسم المتغير. ما فعلته هنا في هذه اللحظة يطلق عليه الربط. حاول تنفيذ الكود السابق لترى ما سيحدث.

السادس

الجمع في PHP

**Addition in PHP**



دعنا الآن نقوم ببعض عمليات الجمع. للجمع في PHP، تستخدم علامة الزائد(+): (إذا كان الكود المكتوب في الفصل السابق مفتوحاً، حاول تغيير النقطة full stop الى علامة الزائد(+). قم بتنفيذ الكود ولاحظ ما سيحدث).

لجمع محتويات المتغيرات، عليك فقط الفصل بين المتغيرات باستخدام علامة الزائد(+). حاول تجربة هذا السكريبت الجديد:

<?php

```
$first_number = ١٠;
$second_number = ٢٠;
$sum_total = $first_number +
$second_number;

$direct_text = 'The two variables added
together = ';

print ($direct_text . $sum_total);
```

?>

في السكريبت السابق، لقد أضفنا متغير ثاني، وخصصنا له القيمة:

```
$second_number = ٢٠;
```

أما المتغير الثالث الذي تم الإعلان عنه، والذي تم تسميته \$sum\_total على يمين علامة يساوي (=)، تم تخصيصه للاحتفاظ بناتج جمع محتويات المتغير الأول مع محتويات المتغير الثاني هكذا:

```
$sum_total = $first_number + $second_number;
```

وتعرف PHP ما هو مخزن بداخل المتغيرات، انها تعرف كلاً من \$first number، \$second number، لأننا خصصنا لهم قيم في السطور السابقة! وعندما يرى علامة الزائد (+) فانه يجمع القيمتين معاً. ويضع الناتج في المتغير الذي يقع على يسار علامة يساوي (=)، المتغير الذي تم تسميته \$sum\_total.

ولطباعة الناتج، تم استخدام الربط:

print (\$direct text . \$sum total);

هذا السكريبت معقد قليلاً عن ما قمت بأدائه قبل ذلك. لو أنك متحير قليلاً، تذكر فقط ماذا كنا نفعل: إضافة محتويات أحد المتغيرات الى محتويات المتغير الآخر. أهم سطر هو:

\$sum total = \$first\_number +  
\$second\_number;

عملية الجمع التي تقع على يمين علامة يساوي تحسب أولاً (\$first number + \$second number). ثم يتم تخزين ناتج الجمع الكلي في المتغير الذي يوجد على يسار علامة يساوي (=) (\$sum total=).

يمكنك أيضاً، جمع أكثر من رقمين. حاول تنفيذ التدريب التالي.

## تدريب

قم بإضافة متغير ثالث الى الكود السابق. وخصص القيمة ٣٠ لهذا المتغير. ضع اجمالي المجموع للثلاثة متغيرات في متغير يدعى \$sum total. استخدم الربط لاطهار وعرض النتائج. (أوبعبارات أخرى قم بجمع الأعداد الثلاثة ١٠، ٢٠، ٣٠!).

أنت لست مضطراً لاستخدام المتغيرات في أداء عملية الجمع. يمكنك أداء ما يلي:

print (١٠ + ٢٠ + ٣٠);

أو من الممكن كتابتها هكذا:

\$number = ١٠;

print (\$number + ٣٠);

لكن ما يجدر الإشارة اليه أنهما نفس الشيء- استخدم علامة الزائد (+) لاجراء عملية الجمع.

في الفصول التالية، ستتعلم كيفية اجراء عمليات الطرح والقسمة والضرب.

-

الفصل السابع  
الطرح في PHP

## Subtraction in PHP

لن أثقل عليك بأمور تتعلق بالرياضيات وتكون أكثر تعقيداً! ولكن ربما تحتاج الى معرفة كيفية استخدام المعاملات الأساسية. وفي هذا الفصل سنتناول عملية الطرح.

عندما قمت باجراء عملية الجمع، لقد اتبعت ما يلي:

<?php

```
$first_number = ١٠;
$second_number = ٢٠;
$sum_total = $first_number +
$second_number;

print ($sum_total);
```

?>

عملية الطرح هي عملية تقريباً تشبه ما سبق، ولكن مع استبدال علامة الجمع(+)، ببساطة بعلامة الناقص(-). غير السطر الذي يحتوى على \$sumtotal الى ما يلي، ثم قم بتنفيذ هذا الكود:

\$sum\_total = \$second\_number - \$first\_number;

السطر السابق هو نفسه السطر الذي استخدم أولاً. ماعدا أننا الآن، نستخدم علامة الناقص (-) بدلاً من علامة الجمع (وتعكس كلاً من المتغيرات). عندما تنفذ السكريبت فانك ستحصل بالطبع على الناتج ١٠. مرة أخرى فان PHP، تعرف ما بداخل المتغيرات التي تدعى \$second number و \$first number. وعندما تمر PHP على علامة الناقص (-) فانها تقوم باجراء عملية الطرح لك، وتضع الناتج في المتغير الموجود على يسار علامة يساوي (=). ثم نستخدم بعد ذلك جملة الطباعة Print لعرض ما بداخل المتغير.

مثل عملية الجمع، يمكنك طرح أكثر من رقم في المرة الواحدة، حاول تجربة ما يلي:

<?php

```
$first_number = ١٠;
$second_number = ٢٠;
$third_number = ١٠٠;

$sum_total = $third_number -
$second_number - $first_number;

print ($sum_total);
```

?>

الناتج الذي ينبغي أن تحصل عليه هو ٧٠. يمكنك أيضاً الخلط بين عمليتي الجمع والطرح. وفيما يلي مثال على ذلك:

<?php

```
$first_number = ١٠;
$second_number = ٢٠;
$third_number = ١٠٠;

$sum_total = $third_number -
$second_number + $first_number;

print ($sum_total);
```

?>

نفذ الكود السابق. ما هي النتيجة التي تحصل عليها؟ هل هي نفسها النتيجة التي كنت تتوقعها؟ لماذا تعتقد أنه سيطبع الرقم الذي طبعه؟ لو كنت تعتقد أنه سيطبع ناتج مختلف عن الناتج الذي حصلت عليه، فإن السبب في ذلك هو الترتيب الذي تم استخدامه لعملية الجمع. هل نقصد ١٠٠-٢٠، ثم نضيف عليهم القيمة ١٠؟ أم أننا كنا نقصد جمع الرقمين ١٠، ٢٠ ثم طرح الناتج من ١٠٠؟ العملية الأولى يكون ناتجها ٩٠، ولكن العملية الثانية يكون ناتجها ٧٠.

لكي توضح ما تقصد، يمكنك استخدام الأقواس عند إجراء العمليات الحسابية. وفيما يلي كيفية كتابة الإجمالي المختلف في الحالتين. حاول أن تجربهم في الكود. ولكن لاحظ أين تضع الأقواس:

للحصول على النتيجة الأولى:

$\$sum\ total = (\$third\ number - \$second\ number) + \$first\ number;$

للحصول على النتيجة الثانية:

$\$sum\ total = \$third\ number - (\$second\ number + \$first\ number);$

انه من الجيد دائماً أن تسد قواس عند إجراء العمليات، لكي توضح ما تريد من PHP أن يقوم بتنفيذه. وبهذه الطريقة فانك لن تحصل على نتيجة مختلفة عن التي تتوقعها!

هناك سبب آخر يدعوك الى استخدام الأقواس والذي يدعي أولوية المعامل. في PHP، فان بعض المعاملات (الرموز الرياضية) يتم حسابها قبل الأخرى. وهذا يعني أنك ستحصل على نتائج غير متوقعة تماماً! كما سيواجهنا في الفصل التالي الخاص بإجراء عملية الضرب.

ل الثامن

الضرب في PHP

**Multiplication in PHP**



للضرب في PHP (مثل غيرها من لغات البرمجة الأخرى) ، فإن الرمز \* هو الذي يستخدم. فإذا رأيت  $20 * 10$  ، فإن ذلك يعني حاصل ضرب 20 في 10. وفيما يلي الكود الذي عليك تنفيذه لتجربته:

<?php

```
$first_number = 10;
$second_number = 20;
$sum_total = $second_number *
$first_number;

print ($sum_total);
```

?>

في الكود السابق، فإننا فقط نضرب القيم المخزنة في المتغيرين أيًا كانت. ثم نخصص متغير للناتج على علامة يساوي (=). (يمكنك أن تخمن الناتج وذلك بدون تنفيذ الكود!)

كما هو الحال في عملية الجمع والطرح، يمكنك إجراء عملية الضرب لأكثر من رقمين كما يلي:

<?php

```
$first_number = 10;
$second_number = 20;
$third_number = 100;

$sum_total = $third_number *
$second_number * $first_number;

print ($sum_total);
```

?>

أو حتى يمكنك كتابتها هكذا:

```
$sum_total = $third_number * $second_number * 10;
```

ولكن حاول أن تجرب هذا الكود. هل من الممكن أن تخمن النتيجة قبل أن تجربته:

<?php

\$first\_number = ١٠;

\$second\_number = ٢;

\$third\_number = ٣;

\$sum\_total = \$third\_number +

\$second\_number \* \$first\_number;

print (\$sum\_total);

>

ما هي النتيجة التي توقعتها؟ لو أنك توقعت أن الناتج الذي ستحصل عليه هو ٥٠، فإنك تحتاج إلى معرفة أولويات المعاملات الحسابية! كما ذكرت من قبل، فإن بعض المعاملات (الرموز الرياضية) يتم حسابها قبل الأخرى في لغة PHP. فعمليتي الضرب والقسمة تكون أكثر أهمية من عمليتي الجمع والطرح. لذلك فإنه يتم حسابها أولاً. وفي العملية الحسابية السابقة، فإن PHP ترى الرمز \* ثم تقوم بإجراء عملية ضرب الرقمين أولاً، وعندما تحصل على النتيجة، فإنها تتوجه مباشرة إلى الرمز الآخر، وهو علامة الزائد (+). إنها تؤدي ذلك أولاً:

\$second\_number \* \$first\_number;

ثم تتجه إلى إجراء عملية الجمع. ولا تؤدي عملية الجمع في البداية:

\$third\_number + \$second\_number

وهذا يجعل استخدام الأقواس أكثر أهمية من ذي قبل! إنك تستخدمهم لإجبار PHP لتحسب بالطريقة التي تريدها أنت. وفيما يلي الشكلين المختلفين. حاول أن تجربهم:

الاستخدام الأول:

$\$sum\_total = \$third\_number + (\$second\_number * \$first\_number);$

الاستخدام الثاني:

$\$sum\_total = (\$third\_number + \$second\_number) * \$first\_number;$

في هذه الحالة، فانك تستخدم الأقواس لاجبار PHP لعرض نتيجة مختلفة. فما هو موجود بين الأقواس يتم تنفيذه أولاً في PHP، ثم تتوجه الى المعامل الآخر. في الاستخدام الأول، فانه تم استخدام الأقواس للتأكد من أن عملية الضرب هي التي ستتم أولاً. وعندما تحصل على ناتج الضرب، فانها تقوم باجراء عملية الجمع. ولكن في الاستخدام الثاني للأقواس، فانه قد تم استخدام الأقواس للتأكد من أن PHP ستقوم باجراء عملية الجمع أولاً، وعندما تحصل على النتيجة لعملية ، تقوم بعد ذلك باجراء عملية الضرب

في الفصل التالي سوف نلقي نظرة على عملية القسمة.

## الفصل التاسع

### الـ في PHP

### Division in PHP

لقسمة عدد ما على الآخر، فانه يمكنك استخدام الرمز (/) في PHP. عندما ترى ٢٠/١٠. فان ذلك يعني قسمة العدد ٢٠ على العدد ١٠، حاول أن تجرب ذلك بنفسك:

```
<?php
```

```
$first_number = ١٠;
$second_number = ٢٠;
$sum_total = $second_number / $first_number;

print ($sum_total);
```

```
?>
```

وللمرة الأخرى، لابد أن تركز حريصاً ومنتبهاً لأولويات المعاملات. حاول أن تجرب هذا الكود:

```
<?php
```

```
$first_number = ١٠;
$second_number = ٢٠;
$third_number = ١٠٠;

$sum_total = ($third_number - $second_number) /
$first_number;

print ($sum_total);
```

```
?>
```

لا تقوم PHP باجراء العمليات من اليسار الى اليمين! فالقسمة يتم اجراؤها قبل عملية الطرح. ولذلك فانه يتم حسابها أولاً:

```
$second_number / $first_number
```

وليس كما يلي:

```
$third_number - $second_number
```

ان استخدام الأقواس سوف يوضح العمليات بشكل أفضل. اليك الاستخدامين لتجرب ذلك بنفسك:

الاستخدام الأول:

```
$sum_total = $third_number - ($second_number /
$first_number);
```

الاستخدام الثاني:

```
$sum_total = ($third_number - $second_number) /
$first_number;
```

الاستخدام الأول سيؤدي الى الحصول على النتيجة ٩٨، ولكن الاستخدام الثاني تكون نتيجته هي ٨! لذلك تذكر أن: عمليتي القسمة والضرب يتم اجراؤها قبل عمليتي الجرح. استخدم الأقواس اذا أردت اجبار على الحساب بطريقة مختلفة والحصول على نتائج مختلفة.

في الفصل التالي سترى كيف تتعامل PHP مع الأرقام العشرية.

-

## الفصل العاشر

### الأعداد العشرية في PHP

## Floating Point Numbers in PHP

العدد العشري هو ذلك العدد الذي يحتوي على النقطة، مثل ٠,٥ و ١٠,٨. لا تحتاج الى أية صياغة خاصة للإعلان عن مثل هذا النوع من الأعداد. وفيما يلي مثال، حاول أن تؤديه:

<?php

```
$first number = ١,٢;
$second number = ٢,٥;
$sum total = $second number +
$first number;

print ($sum total);
```

?>

يمكنك اجراء عمليات الجمع، الطرح، القسمة والضرب بنفس الطريقة التي استخدمتها مع الأعداد. هناك تحذير يصاحب استخدام العشرية، وعلى الرغم من : أنك لا ينبغي أن تثق بهم، ولكن لو أنك بعد حقيقة، فان الحقائق تتطلب نتائج دقيقة!

## بعض التدريبات

لكي تستوعب ما سبق حول المتغيرات الرقمية، اليك بعض التدريبات (في جمل الطباعة، ينبغي ألا يكون هناك أية أرقام- فقط أسماء المتغيرات).

## تدريب

اكتب السكريبت اللازم لأداء عملية الجمع على الأرقام التالية ١٩٨,١٣٤,٧٦ استخدم جملة الطباعة Print للحصول على الناتج.



## تدريب

اكتب السكريبت اللازم لجمع الرقمين التاليين ١٥,٤٥ ثم اطرح الناتج من ١٠٠. استخدم جملة الطباعة Print للحصول على الناتج.

## تدريب

استخدم المتغيرات في حساب نتائج العمليات التالية:

$$\underline{(200 * 15) / 10}$$

استخدم جملة الطباعة للحصول على الناتج.

في الباب التالي سوف نلقي نظرة على المنطق الشرطي في PHP.



ب الخامس

المنطق الشرطي في PHP

**Conditional Logic in PHP**

## أهداف الباب الخامس:

بعد الانتهاء من دراسة هذا الباب ينبغي أن يكون الطالب قادراً على أن:

- ١- يحدد متى يستخدم المنطق الشرطي في Php.
- ٢- يسمح لبعض المستخدمين الوصول إلى الموقع.
- ٣- يكتب الصيغة المستخدمة للسماح لأحد الزوار بالدخول إلى الموقع.
- ٤- يكتب صيغة جملة If بدون اختبار.
- ٥- يميز موضع الأقواس الدائرية والمعقوفة ولا يخلط بينهما.
- ٦- يحدد الخطأ في الصيغة التالية:

```
($User_Name == "authentic"
```

```
//Code to Let user access the site here;
```

```
{
```

- ٧- يميز الأخطاء الناتجة عن كتابة الأقواس عند صياغة جملة If.
- ٨- يحدد أين يكتب الشرط في جملة If.
- ٩- يميز الصياغة الصحيحة لجملة If.
- ١٠- يستخدم جملة If لعرض صورة على الصفحة.
- ١١- يدرك ما هو الكود الذي يستخدم للتبديلين صورتين.
- ١٢- يدرك فيما تستخدم جملة Else.
- ١٣- يدرك متى تتجاهل Php سطر الكود المخصص لجملة If.
- ١٤- يضيف جزء Else If إلى جملة If-else.
- ١٥- يعطي مثال على استخدام جملة If-Else If-Else.

- ١٦- يدرك فيما تستخدم علامات يساوي المزدوجة (==).
- ١٧- يحدد معاملات المقارنة في PHP.
- ١٨- يدرك فيما يستخدم المعامل (!=).
- ١٩- يدرك فيما يستخدم المعامل Less than <.
- ٢٠- يدرك فيما يستخدم المعامل > أكبر من.
- ٢١- يدرك فيما يستخدم المعامل أصغر من أو يساوي <=.
- ٢٢- يدرك فيما يستخدم المعامل أكبر من أو يساوي >=.
- ٢٣- يعطي مثال على استخدام المعامل لا يساوي نفس القيمة في PHP.
- ٢٤- يعطي مثال على المعاملات أكبر من وأصغر من في PHP.
- ٢٥- يعطي مثال امالات أكبر من أو يساوي وأصغر يساوي في p
- ٢٦- يحسب الخصم وقدره ١٠٪ في حالة إذا أنفق الشخص أكثر من ١٠٠ ج.
- ٢٧- يدرك فيما تستخدم جمل التحويل Switch.
- ٢٨- يذكر مثال على استخدام جمل التحويل.
- ٢٩- يحدد المعاملات المنطقية في PHP.
- ٣٠- يدرك فيما يستخدم الرمز &&.
- ٣١- يعطي مثال على استخدام الرمز &&.
- ٣٢- يدرك فيما يستخدم المعامل Or.
- ٣٣- يعطي مثال على استخدام المعامل Or.
- ٣٤- يدرك فيما يستخدم المعاملين And، Or.
- ٣٥- يعطي مثال على استخدام المعاملين And، Or.
- ٣٦- يدرك فيما يستخدم المعامل XOR.
- ٣٧- يعطي مثال على استخدام المعامل XOR.

- ٣٨ يدرك فيما يستخدم معامل النفي !.
- ٣٩ يعطي مثال على استخدام المعامل !.
- ٤٠ يعطي مثال على استخدام القيم المنطقية في Php.
- ٤١ يوضح أولوية المعاملات في Php.
- ٤٢ يفرق بين المعاملات === و !==.
- ٤٣ يفرق بين المعاملات === و == ! بمثال.

# الفصل الأول

## ١ في PHP

### PHP If Statements

لقد رأيت في الباب السابق أن المتغيرات هي مساحات تخزينية للقيم النصية والرقمية. لكن السبب وراء تخزين هذه المعلومات هو أنه يمكنك استخدامها

في أداء مهام فيما بعد. لو أنك قمت بتخزين اسم المستخدم User name داخل متغير، على سبيل المثال، فانك تحتاج الى اختبار ما اذا كان هذا الاسم اسم مستخدم مقبول أو متاح. لمساعدتك على اجراء الاختبار لشيء ما فانك تحتاج الى المنطق الشرطي، سنلقي نظرة على المنطق الشرطي ونتناول هنا بعض التدريبات عليه.

## المنطق الشرطي

المنطق الشرطي في محمله يعني " ماذا يحدث لو..." عندما تضغط على زر مكتوب عليه " لا تضغط على هذا الزر – تحت أي ظرف!" فانك في هذه الحالة تستخدم المنطق الشرطي. فانك تتساءل، "حسناً، ما الذي يحدث اذا ضغطت على هذا الزر؟". انك تستخدم المنطق الشرطي في حياتك اليومية طوال الوقت:

" اذا قمت برفع صوت الموسيقى، هل سيكون الجيران مسرورون؟"

"اذا أنفقت كل مالى علي يد، هل هذا سيجعلني سعيداً؟"

"اذا قمت بدراسة هذا المقرر، هل هذا سيجحسن مكانتي بالمجتمع؟"

يستخدم المنطق الشرطي الكلمة "IF" كثيراً. وفي معظم الحالات فانك تستخدم المنطق الشرطي عندما تريد اختبار القيمة المخزنة داخل المتغير. ويمكنك بعدها أن تتخذ القرارات اعتماداً على القيمة المخزنة بداخل المتغير. على سبيل المثال، فكر في اسم المستخدم مرة أخرى. ينبغي أن يكون لديك متغير مثل هذا:

```
$User_Name = "My_Regular_Visitor";
```

النص " My\_Regular\_Visitor " سيتم تخزينه داخل المتغير الذي يدعى \$User\_Name. سوف تستخدم بعض المنطق الشرطي لتختبر هل يحتوى هذا المتغير على زوارك المنتظمين أم لا. فيكون السؤال بالصورة التالية:

" هل \$User\_Name اسم المستخدم معروف وموثق، اذن اسمح للمستخدم \$User\_Name للوصول الى الموقع ".

في PHP، فانك تستخدم الكلمة "IF" بالصورة التالية:



```
if ($User_Name == "authentic") {
 //Code to let user access the site here;
}
```

بدون أي اختبار، فإن جملة If تبدو هكذا:

```
if () {
}
```

كما ترى انها أكثر وضوحاً، هنا. لاختبار متغير أو تحقق شرط، فانك تبدأ بالكلمة "IF". ثم بعد ذلك تكتب زوج من الأقواس الدائرية round brackets. كما تحتاج أيضاً للعديد من الأقواس - الأقواس المعقوفة، تلك التي توجد على يمين الحرف "P" على لوحة المفاتيح ( لاسيما اللوحات الأمريكية). انك تحتاج س المعقوف اليسار { أولاً، ثم بع القوس المعقوف اليمين } في نهاية جملة IF. اذا تم عكسها، فإن PHP سترفض العمل. وسوف يسبب هذا لك خطأ:

```
if ($User_Name == "authentic") {
 //Code to Let user access the site here;
}
```

لاحظ أيضاً الصياغة التالية:

```
if ($User_Name == "authentic") {
 //Code to Let user access the site here;
}
```

في الصياغة الأولى تم استخدام الأقواس المعقوفة بالطريقة الخطأ (لأنه تم عكس الأقواس المعقوفة، ينبغي استخدام القوس اليسار أولاً ثم اليمين) ، بينما الصياغة الثانية فقد تم استخدام القوسين المجعدين اليسار. وكلاهما غير صحيح.

أما داخل الأقواس الدائرية، فانك تكتب الشرط الذي تريد اختباره. في المثال السابق، لقد قمت باختبار لترى ما اذا كان المتغير الذي يدعى **\$User\_Name** يحتفظ بالقيمة "authentic":

```
($User_Name == "authentic")
```

مرة أخرى، ستحصل على رسالة خطأ اذا لم تستخدم الأقواس الدائرية بشكل صحيح! لذلك فان الصياغة لجملة IF تكون هكذا:

```
if (Condition_or_Variable_to_test) {

 //your code here;

}
```

وفيما يلي، سوف نستخدم جملة IF لعرض صورة على الصفحة:

سوف نستخدم جملة الطباعة Print لكي تطبع نتائج كود HTML. على سبيل المثال، الكود التالي بلغة HTML يستخدم لعرض صورة:

```

```

انه فقط كود HTML عادي، ولكن يمكنك وضع الكود داخل جملة الطباعة كما يلي:

```
print("");
```

وعندما تقوم بتنفيذ الكود، فان الصورة ينبغي أن يتم عرضها. بالطبع، فانك تحتاج الى صورة تدعى mosque.jpg، وهي مخزنة في مجلد يدعى images بداخل المجلد WWW أولاً وقبل تنفيذ الكود. ثم قم بكتابة السكريبت التالي:

```
<?PHP
```

```
print("");
```

```
?>
```

احفظ هذا السكريبت على نفس المجلد باسم المجلد images (على الرغم من أنه ليس بداخل المجلد images). قم بتشغيل السيرفر، واجراء هذه المحاولة، وسترى الصورة mosque معروضه أمامك.

في الفصل التالي ، سنتناول بعض التدريبات كأمثلة على استخدام جملة If لتتضح أكثر.

## الفصل الثاني

### بعض التدريبات على استخدام جملة If في PHP

### Some Practice with PHP If Statements

يمكنك استخدام جملة IF لعرض صورتك، لقد تعلمت ذلك في الفصل السابق.  
إذا اختار المستخدم "mosque"، إذن سيتم عرض صورة mosque. إذا

اختار المستخدم "kitten"، سيتم عرض صورة أخرى ( الصورة المسماه kitten) اليك الكود:

```
<?PHP

$kitten_image = ١;
$mosque_image = ٠;

if ($kitten_image == ١) {

 print ("");

}

?>
```

قم بكتابة هذا الكود، ثم ق  
لم تستخدم HTML هنا!) باسم **testImages.php** .(لا ح

عندما تقوم بتنفيذ السكريبت، فإن الصورة التي تدعى kitten هي التي ينبغي عرضها. دعنا نلقي نظرة على الكود لنرى ما الذي يحدث.

في السطرين الأول والثاني تم اعداد المتغيرات:

```
$kitten_image = ١;
$mosque_image = ٠;
```

لقد تم تخصيص القيمة ١ للمتغير الذي يدعى **\$kitten\_image**. بينما القيمة صفر قد تم تخصيصها للمتغير الذي يدعى **\$mosque\_image**. ثم قمت بعد ذلك باستخدام جملة IF. ولقد تم استخدامها هنا بدون جملة الطباعة Print:

```
if ($kitten_image == ١) {

}
```

لاحظ أنه لا يوجد فاصلة منقوطة (;) في نهاية السطر الأول – انك لا تحتاج اليها. بعد الكلمة "If" ينبغي كتابة الأقواس الدائرية. ثم يأتي بعد ذلك اسم

المتغير : **\$kitten\_image**، الذي نريد أن نختبر القيمة المخزنة داخله. وبصفة خاصة، أريد أن أختبر امتلاك هذا المتغير للقيمة ١. ولذلك فإنك تحتاج الى علامتي يساوي المزدوجتين (==). وعلامات يساوي المزدوجتين هنا لا تعنى يساوي، وانما تعني "تعيين القيمة للمتغير".

لذلك فان ما نريد قوله هو :

" اذا كان المتغير الذي يدعى **\$kitten\_image** يمتلك القيمة ١ اذن قم بتنفيذ بعض الكود."

ولتكلمة السطر الأول في جملة IF لابد أن تكتب القوس الدائري الثاني، والقوس المعقوف اليسار. اذا نسيت أحدهما، فسوف تحصل على رسالة خطأ بالتأكيد! الكود الذي سيتم تنفيذه هو جملة Print، لذلك فان الصورة التي تدعى kitten هي التي سيتم عرضها وهذا هو ما تم تضمينه في جملة IF:

```
if ($kitten_image == ١) {

 print("");

}
```

ستحتاج الى الفاصلة المنقوطة (;) في نهاية جملة الطباعة Print.

ولكن لو كانت جملة IF عبارة عن سطر واحد فقط، يمكنك فقط أن تكتبها هكذا:

```
if ($kitten_image == ١) { print("<IMG SRC =
 images/kitten.jpg>"); }
```

وبعبارات أخرى، يتم كتابة كل شيء في نفس السطر. لا تلقي PHP اهتماماً بالمسافات الفارغة، ولذلك فان هذا الكود يكون مقبول تماماً. ليس جيداً في القراءة، ولكنه مقبول!

ولعرض الصورة التي تدعى mosque، واستخدامها في الكود، حاول أن تنفذ الكود التالي:

```
<?PHP
```

```

$kitten_image = ٠;
$mosque_image = ١;

if ($kitten_image == ١) {

 print("");

}

if ($mosque_image == ١) {

 print("");

}

?>

```

لاحظ أن المتغير الذي يدعى **\$kitten\_image** يمتلك الآن القيمة ٠ و أن المتغير الذي يدعى **\$mosque\_image** يمتلك القيمة ١. وأن جملة IF الجديدة هي نفسها الأولى. عندما تقوم بتنفيذ هذا السكريبت، سيتم عرض الصورة التي تدعى mosque بسبب هذا السطر:

```
if ($kitten_image == ١) {
```

والذي يعني، " لو أن المتغير الذي يدعى **\$kitten\_image** يمتلك القيمة ١...". فان PHP لا تكمل قراءة بقية جملة IF أي تتجاهله، لأن **\$kitten\_image** يمتلك القيمة ٠. ولذلك فانها تقفز لأسفل الى السطر الثاني لجملة IF، لتختبر ما يلي:

```
if ($mosque_image == ١) {
```

وحيث أن المتغير الذي يدعى **\$mosque\_image** هو الذي يمتلك القيمة ١ حقاً، لذلك فان الكود الموجود داخل جملة IF سيتم تنفيذه. هذا الكود سيعرض الصورة التي تدعى mosque في هذه المرة:

```
print("");
```

في الفصل التالي، سوف نتناول جمل **if ... Else** .



## الفصل الثالث

# جمل IF ... Else في PHP

## F ... Else Statem in PHP

بدلاً من استخدام اثنين من جمل IF، كما هو الحال في الفصل السابق، يمكنك استخدام جملة واحدة فقط وهي If...Else. كما يلي:

<?PHP

```
$kitten_image = ٠;
$mosque_image = ١;

if ($kitten_image == ١) {

 print("<IMG SRC
=images/kitten.jpg>");

}
else {

 print("<IMG SRC
=images/mosque.jpg>");

}
```

?>

قم بكتابة هذا السكريبت الجديد، واحفظه، ثم قم بتنفيذه. ستلاحظ أن الصورة التي تدعي mosque هي التي تم عرضها في المستعرض. هذه المرة، تم استخدام جملة If...Else. دعنا نرى كيف تعمل.

الصياغة لجملة If...Else هي كما يلي:

```
if (condition_to_test) {

}
else {

}
```

إذا تم التدقيق في هذه الصياغة عن قرب، ستري أنها في البداية جملة IF عادية، ولكنها متبوعة بالجزء الخاص ب Else بعدها. وفيما يلي الجزء الخاص ب Else :

```
else {
```

}

مرة أخرى، لاحظ الأقواس المعقوفة اليسار واليمين. ويكتب الكود الذي تريد أن تنفذه بين الأقواس المعقوفة. وفي الكود السابق، تم اعداد اثنين من المتغيرات:

```
$kitten_image = ٠;
$mosque_image = ١;
```

لقد تم تخصيص القيمة ٠ للمتغير الذي يدعى **\$kitten\_image** ، كما تخصيص القيمة ١ للمتغير الذي يدعى **\$mosque\_image**. السطر الأول من جملة **If** يختبر محتويات المتغير الذي يدعى **\$kitten\_image**. انه يختبر ليرى هل هذا المتغير يمتلك القيمة ١.

```
if ($kitten_image == ١) {
```

نتساءل الآن: " هل الصحيح أن المتغير الذي يدعى **\$kitten\_image** يمتلك القيمة ١؟ المتغير **\$kitten\_image** يمتلك القيمة ٠، ولذلك فإن **PHP** ترى أن هذا الشرط غير متحقق. لأن القيمة "ليست صحيحة **Not True** أو **False** يتم ارجاعها، وبالتالي تتجاهل **PHP** سطر الكود المخصص لجملة **IF** هذه. وبدلاً من ذلك، فانها تقوم بتنفيذ الكود الخاص بجزء **Else**. انها لا تحتاج أن تجري أي اختبارات – فجزء **Else** يعني " عندما تكون كل الخيارات مستنفذة وغير متحققة، نفذ الكود الموجود بين الأقواس المعقوفة بجزء **Else**". وبالنسبة للكود كان:

```
else {

 print("<IMG SRC
 =images/mosque.jpg>");

}
```

ولذلك فإن الصورة التي تدعي **Mosque** هي التي سيتم عرضها. غير المتغيرين من هذا:

```
$kitten_image = ٠;
$mosque_image = ١;
```

الى ما يلي:

```
$kitten_image = ١;
$mosque_image = ٠;
```

نفذ الكود مرة أخرى وراقب ماذا يحدث. سترى الصورة التي تدعى kitten ! ولكن هل يمكنك أن تستنتج لماذا؟

في الفصل التالي، سنتناول جمل If...Else.

## الفصل الرابع

### جمل If...Else If

### PHP if ... else if Statements

يمكنك أيضاً إضافة أجزاء "else if" لجمل If، التي تم استخدامها في الفصول السابقة. والصياغة تكون هكذا:

```
else if (another_condition_to_test) {

}
```

غير الكود السابق الى ما يلي، لترى كيف تعمل Else If:

<?PHP

```
$kitten_image = ١;
$mosque_image = ٠;

if ($kitten_image == ١) {

 print("<IMG SRC
=images/kitten.jpg>");

}
else if ($mosque_image == ١) {

 print("<IMG SRC
=images/mosque.jpg>");

}
else {

 print("No value of ١ detected");

}
```

?>

وهنا فقط تم الاختبار لترى أي المتغيرات يمتلك القيمة ١. لكن لاحظ سطور "Else If" ( وأنه يوجد مسافة بين كل من Else و If):

```
else if ($mosque_image == ١) {
```

```
print("<IMG SRC
=images/mosque.jpg>");

}
```

وهي تعني أنه إذا لم تكن جملة `if` متحققة، إذن جرب هذه. "ستحاول PHP أن تقيم الشرط الجديد. وإذا كان متحققاً ( فان المتغير الذي يدعي `$mosque_image` يمتلك القيمة ١ )، لذلك فان الكود الموجود بين الأقواس المعقوفة الجديدة سيتم تنفيذه. ولو كان غير متحقق (المتغير الذي يدعي `$mosque_image` لا يمتلك القيمة ١)، فان هذا السطر من الكود سيتم تجاهله، وتستمر PHP الى ما يليه.

ولاجراء أي تقييمات أخرى لا يزال لدينا جزء `Else` في النهاية. لاحظ أن كل الأجزاء (`if`, `else if`, and `else`) يتم توزيعها بدقة بين زوج الأقواس المعقوفة:

```
if ($kitten_image == ١) {

}

else if ($mosque_image == ١) {

}

else {

}
```

يمكنك إضافة أجزاء عديدة من `Else If` كما تحب، كل جزء للشرط الذي تود أن تختبره. ولكن غير المتغيرين من:

```
$kitten_image = ١;
$mosque_image = ٠;
```

إلى هذا:

```
$kitten_image = ٠;
$mosque_image = ٠;
```

ثم قم بتنفيذ الكود مرة أخرى. ما الذي تتوقع أن يحدث؟  
في الفصل التالي، سنتناول معاملات المقارنة.



## الفصل الخامس

### معام مقارنة في PHP

## PHP Comparison Operators

لقد تعرفت في الفصل السابق على كيفية اختبار القيمة المخزنة بداخل المتغير. لقد استخدمت If ، Else...If و Else. لقد استخدمت علامات يساوي المزدوجة (==) لنتحبر هل المتغير كان نفس الشيء مثل النص المباشر. علامات يساوي المزدوجة (==) تعرف بمعامل مقارنة. ويوجد العديد من المعاملات التي يمكنك استخدامها. وها هي القائمة بها. قم بالقاء نظرة، ثم بعد ذلك سوف أعرض لك بعض الأمثلة على كيفية استخدامهم.

#### جدول (٤-١) يوضح معاملات المقارنة في PHP

| Operand | Example                                    | Meaning                   |
|---------|--------------------------------------------|---------------------------|
| ==      | <code>\$variable1 == \$variable2</code>    | Has the same value as     |
| !=      | <code>\$variable1 != \$variable2</code>    | Is NOT the same value as  |
| <       | <code>\$variable1 &lt; \$variable2</code>  | Less Than                 |
| >       | <code>\$variable1 &gt; \$variable2</code>  | Greater Than              |
| <=      | <code>\$variable1 &lt;= \$variable2</code> | Less than or equals to    |
| >=      | <code>\$variable1 &gt;= \$variable2</code> | Greater than or equals to |

وفيما يلي بعض المعلومات عن تلك المعاملات الموضحة في الجدول أعلاه.

#### - علامة يساوي المزدوجة (==) وتعني (لهما نفس القيمة)

علامة يساوي المزدوجة يمكن أن تعني "يملك القيمة كذا" أو "يملك القيمة مثل". في المثال التالي، فإن المتغير الذي يدعى `$variable١`، سيتم مقارنته بالمتغير الذي يدعى `$variable٢`.

```
if ($variable١ == $variable٢) {
}
```

## - علامة لا يمتلك نفس القيمة (!=)

يمكنك اختبار أيضا الشرط أن أحد المتغيرين ليس نفس الآخر. وفي هذه الحالة فانك تحتاج الى اتحاد علامة التعجب (!) / مع علامة يساوي هكذا (!=). اذا كنت تختبر اسم المستخدم الحقيقي أو المسجل، على سبيل المثال، فانه يمكنك أن تكتبها هكذا:

```
if ($what_user_entered != $username) {

print("You're not a valid user of this site!");

}
```

الكود المكتوب أعلى يقول، " اذا كان اسم المستخدم الذي يتم إدخاله ليس بنفس القيمة المخزنة داخل المتغير الذي يدعى \$username اذن قم بطباعة الرسالة **You're not a valid user of this site**.

## - المعامل أقل من < Less Than

ويمكنك أن تستخدمها اذا أردت اختبار قيمة هل هي أقل من قيمة أخرى. ( ويستخدم قوس الزاوية اليسار لاجراء هذه العملية <).

## - المعامل أكبر من > (Greater Than)

ربما تريد أن تختبر أحد القيم هل هي أكبر من الأخرى. استخدم قوس الزاوية اليمين لهذه العملية (>).

## - المعامل أصغر من أو يساوي <= (Less than or equals to)

للقليل من الدقة أكثر، يمكنك أن تختبر هل احدى المتغيرات أقل من أو يساوي الآخر. استخدم قوس الزاوية اليسار متبوعاً بعلامة يساوي (<=).

## - المعامل أكبر من أو يساوي Greater than or equals to >=

إذا كنت تحتاج اختبار أحد المتغيرات هل هو أكبر من أو يساوي الآخر، استخدم قوس الزاوية اليمين متبوعاً بعلامة يساوي هكذا (>=).

في الفصول التالية ، سنتناول بعض الأمثلة على كيفية استخدام معاملات المقارنة. لقد استخدمت في هذا الفصل علامة يساوي المزدوجة، لذا سنبدأ الفصل التالي بعلامة " لا يملك نفس القيمة أو لا يساوي".

## الفصل السادس

استخدام المعامل لا يساوي نفس القيمة في PHP

**PHP Not Equal To**

في الفصل السابق، أدركت ماذا تعني معاملات المقارنة. أما في هذا الفصل، سنكتشف سوياً الدور الذي يقوم به المعامل != والذي يعني ليس نفس القيمة.

افتح محرر النصوص الخاص بك، ثم قم بإضافة هذا السكريبت:

<?PHP

```
$correct_username = 'logmein';
$what_visitor_typed = 'logMEin';

if ($what_visitor_typed != $correct_username)
{

 print("You're not a valid user of this
 site!");

}
```

?>

احفظ عملك ثم قم بتنفيذه. ينبغي أن تكون قادراً على أن تخمن ما الذي سيقوم بتنفيذه! لكن الشيء الجديد الذي تلاحظه هنا هو استخدام معامل المقارنة !=. بدلاً من استخدام علامة يساوي المزدوجة، فأنت تستخدم الآن علامة التعجب وعلامة يساوي مفردة. بينما بقية جملة If هي بالضبط نفس الصيغة التي استخدمتها من قبل.

لا بد أن تكون الأشياء التي يتم اجراء المقارنة عليها مختلفة، قيل أن ترجع PHP القيمة الحقيقية Value of true. في المتغير الثاني (\$what\_visitor\_typed)، الحروف "ME" كتبت بحروف كبيرة capital letters، ولكن في المتغير الأول كتبت بحروف صغيرة Lower case. لذلك فإن المتغيرين ليسا نفس الشيء. و لأننا استخدمنا المعامل لا يمتلك نفس القيمة Not equal to، فإن النص ستم طباعته. قم بتغيير السكريبت الى هذا:

```
$correct_username = 'logmein';
$what_visitor_typed = 'logmein';
```

```
if ($what_visitor_typed != $correct_username) {
 print("You're not a valid user of this site!");
}
else {
 print("Welcome back, friend!");
}
```

يمكنك أن تكتشف بنفسك ما الذي تغير في السكريبت. قبل أن تقوم بتنفيذه، ما الذي ستتم طباعته؟

في الفصل التالي، سنلق على كيفية استخدام كلاً من المعاملا من (<) والمعامل أكبر من (>).

## الفصل السابع

المعاملات أكبر من و أصغر من في PHP

**PHP Less Than, Greater Than**



الرموز (<) أقل من و أكبر من (>) استخدامهما بسيط ومتكرر. انها حقاً مفيدة في حلقات التكرار ( التي سوف نتناولها في الباب الخامس)، واختبار الأرقام بصفة عامة.

افترض أنك أردت اختبار اذا ما كان شخص ما قد أنفق أكثر من ١٠٠ جنيه على موقعك. ولو أنهم فعلوا، فسوف تمنحهم خصماً قدره ١٠٪. في هذه الحالة يمكنك استخدام المعاملات أكبر من و أصغر من. حاول أن تنفذ السكريبت التالي. افتح محرر النصوص، ثم اكتب السكريبت التالي. واحفظ عملك، ثم قم بتنفيذه على السيرفر الخاص بك.

```
<?PHP
```

```
$total_spent = ١١٠;
$discount_total = ١٠٠;

if ($total_spent > unt_total) {

 print("١٠ percent discount applies
 to this order!");

}
```

```
?>
```

باستخدام الرمز أكبر من (>)، نقول " اذا كانت النفقات الكلية كانت أكبر من الخصم الكلي اذن ستقوم بتنفيذ بعض الكود."

ويمكنك استخدام الرمز أقل من بنفس الطريقة. غير السكريبت الى هذا ( السطور السوداء العريضة هي الجديدة):

```
<?PHP
```

```
$total_spent = ٩٠;
$discount_total = ١٠٠;

if ($total_spent > $discount_total) {
```

```

 print("\00 percent discount applies to
 this order!");

 }
 else if($total_spent < $discount_total) {

 print("Sorry – No discount!");

 }

?>

```

في جزء Else الذي تمت اضافته أعلى، فانك تختبر لترى ما اذا كانت النفقات الكلية أقل من (<) ١٠٠ جنيه. ولو تحقق الشرط، فان رسالة جديدة سيتم عرضها. لاحظ أن المتغير **\$total\_spent** قد تم تقليله أو انقصه الى ٩٠.

في الفصل التالي، سنتناول المعاملات أكبر من أو يساوي و أقل من أو يساوي.

## الفصل الثامن

المعاملات أكبر من أو يساوي وأصغر من أو  
يساوي

**Less Than or Equal To, Greater  
Than or Equal To**

يمكنك استخدام نفس الكود الذي استخدمته في الفصل السابق، لتوضيح المعاملات " أقل من أو يساوي" و "أكبر من أو يساوي". قم بتغيير هذا السطر في الكود:

```
$total_spent = ٩٠;
```

الى هذا:

```
$total_spent = ١٠٠;
```

الآن قم بتنفيذ الكود مرة أخرى. هل تم طباعة أي شيء؟

السبب وراء عدم طباعة أي شيء، وعدم ظهور أي رسائل خطأ، لأنك لم تستخدم أي منطق شرطي يختبر المساواة للقيمة. لقد اختبرنا فقط اذا ما كان احدهما أصغر من (<)، أو أكبر من (>) الآخر. نحتاج الى أن نختبر اذا كانوا نفس الشيء(كما هم

وبدلاً من إضافة جزء Else If ، ليختبر اذا كان الاجماليين متساويين، يمكنك أن تستخدم المعاملات أصغر من أو يساوي <= أو >= . وفيما يلي كيف تم تغيير هذا السطر في الكود:

```
else if($total_spent < $discount_total) {
```

الى هذا:

```
else if($total_spent <= $discount_total) {
```

الشيء الوحيد الذي تغير هو الرمز أقل من أو يساوي الذي تم استخدامه بدلاً من استخدام علامة أصغر من فقط.

الآن قم بتنفيذ الكود مرة أخرى. لأننا الآن نقول " اذا كان اجمالي ما تم انفاقه أقل من أو يساوي اجمالي الخصم، اذن عليك بتنفيذ الكود." لذلك فان النص سيتم طباعته على الشاشة.

## تدريب

افترض أنك تريد تطبيق الخصم اذا ما كانت النفقات أكبر من أو تساوي ١٠٠. غير الكود الموجود أعلاه لكي تظهر الرسالة الصحيحة. استخدم الرمز >= لهذا التدريب.

المعاملات المنطقية تعد أقل في الاستخدام من معاملات المقارنة ولكنها تستحق أن نعطيها بعض الاهتمام. وقبل أن ننتقل اليهم، دعنا نلقي نظرة على تقنية أخرى للمنطق الشرطي يمكنك استخدامها وهي – جملة التحويل Switch Statement.

## الفصل التاسع

جمل الت أو التبديل في PHP

**PHP Switch Statements**

في بعض الأكواد التي تم استخدامها في الفصول السابقة، اعتماداً على القيمة المخزنة بداخل المتغير يتم تنفيذ الشرط. وقائمة طويلة من جمل If و Else...If تم استخدامها. اختيار أفضل، لو أن لديك متغير واحد فقط لاختباره، فانه لا بد من استخدام شيء ما يدعى جمل التحويل. لتري كيف تعمل جمل التحويل، قم بدراسة الكود التالي:

```
<?php

$picture = 'mosque';

switch ($picture) {

 case 'kitten':

 print('Kitten Picture');
 break;

 case 'mosque':

 print('Mosque Picture');
 break;

}

?>
```

في الكود السابق، وضعت النص المباشر "mosque" داخل المتغير الذي يدعى **\$picture**. انه هذا النص المباشر هو ما نريد أن نختبره. نريد أن نعرف ما بداخل المتغير، وبناءاً على ذلك يمكننا عرض الصورة الصحيحة. ولاختبار جملة مفردة بجملة التحويل، فان الصياغة التالية هي التي تستخدم:

```
switch ($variable_name) {

 case 'What_you_want_to_check_for':

 //code here
 break;
```

}

انها تبدو معقدة قليلاً، ولذلك سنقوم بكسرها أو ايقافها.

**switch (\$variable\_name) {**

لقد بدأت بالكلمة "switch" ثم زوج من الأقواس الدائرية. داخل الأقواس الدائرية، قمت بكتابة المتغير الذي تريد أن تختبره. وبعد الأقواس الدائرية، فانك تحتاج الى القوس المعقوف اليسار.

**case 'What\_you\_want\_to\_check\_for':**

الكلمة "case" تستخدم قبل أي قيمة تريد أن تختبرها. وفي الكود التالي، فان قائمة من القيم تتبع من قائمة منسدلة. هذه القيم كانت: mosque and kitten: كان التبديل بين الاثنين. هذه هي القيم التي نحتاجها بعد الكلمة "case". وبعد النص أو الذي تريد اختباره، تحتاج الى كتابة فوق بعض (:).

**//code here**

بعد النقطتين في سطر Case، اكتب الكود الذي تريد تنفيذه. لست بحاجة الى أن أذكرك، بأنك ستحصل على رسالة خطأ لو أنك نسيت أي فاصلة منقوطة في نهاية أي سطر من سطور الكود!

**break;**

تحتاج الى اخبار PHP، أن تتوقف "break out" عن جملة التحويل Switch statement. ولو أنك لم تفعل، فان PHP سوف تنتقل الى جملة Case التالية وتختبر ما بها. وللتخلص من عملية التحويل أو التبديل استخدم الكلمة "break".



في الفصل التالي، سنتناول المعاملات المنطقية.

## الفصل العاشر

### المعاملات المنطقية في PHP

### PHP Logical Operators

كما هو الحال في معاملات المقارنة التي تعاملت معها سابقاً، هناك شيء ما في PHP يطلق عليه المعاملات المنطقية. تحتاج الى استخدام هذه المعاملات لاختبار أكثر من شرط في نفس الوقت. على سبيل المثال، ربما تختبر هل اسم المستخدم وكلمة السر صحيحة في نفس جملة If. هذا هو الجدول الذي يضم هذه المعاملات.

جدول (٢-١) يوضح المعاملات المنطقية في PHP

| Operand           | Example                                         | Meaning                                   |
|-------------------|-------------------------------------------------|-------------------------------------------|
| <b>&amp;&amp;</b> | <code>\$variable1 &amp;&amp; \$variable2</code> | Are both values true?                     |
| <b>  </b>         | <code>\$variable1    \$variable2</code>         | Is at least one value true?               |
| <b>AND</b>        | <code>\$variable1 AND \$variable2</code>        | Are both values true?                     |
| <b>XOR</b>        | <code>\$variable1 XOR \$variable2</code>        | Is at least one value true, but NOT both? |
| <b>OR</b>         | <code>\$variable1 OR \$variable2</code>         | Is at least one value true?               |
| <b>!</b>          | <code>!\$variable1</code>                       | Is NOT something                          |

المعاملات الجديدة غريبة الى حد ما، اذا كنت تراها لأول مرة. ولكنها مفيدة جداً، على الرغم من ذلك، سأحاول أن أقرب معناها لك.

## - المعامل &&

هذا الرمز يعني And (و). استخدم هذا الرمز عندما تحتاج الى كل القيم تكون متحققة، كما هو الحال في المثال الخاص باختبار كلاً من اسم المستخدم وكلمة السر. على كل حال، أنت لا تريد أن تسمح للأشخاص بالدخول اذا كان لديهم اسم مستخدم صحيح ومعروف ولكن كلمة السر ليست صحيحة! واليك المثال التالي:

```
$username='user';
$password='password';

if ($username=='user' && $password
=='password') {

 print("Welcome back!");
```

```

}
else {

 print("Invalid Login Detected");

}

```

ان جملة `if` هنا تقوم بأداء ما اعتدت عليه سابقاً، ولكن لاحظ أنه في المثال السابق قد تم اختبار شرطين معاً في نفس الوقت:

```

$username == 'user' && $password
== 'password

```

وهذا السطر يخبرك، "إذا كان اسم المستخدم صحيحاً و كلمة السر كذلك، اذن اسمح للمستخدم بالدخول". وكلا الشرطين لابد من وضعهما بين الأقواس الدائرية الخاصة بجملة `IF`.

## || المعامل

الخطين المستقيمين هذا الرمز يعني `OR` (أو). استخدم هذا الرمز عندما تحتاج فقط الى أحد الشروط فقط يكون متحققاً. على سبيل المثال، افترض أنك تريد أن تطبق الخصم للأشخاص الذين أنفقوا أكثر من ١٠٠ جنيه أو الذين لديهم مفتاح خاص. وإلا فلن يحصلوا على أي خصم. ستكتب الكود كما يلي:

```

$total_spent = ١٠٠;
$special_key = 'SK١٢٣٤٥';

if ($total_spent == ١٠٠ || $special_key
== 'SK١٢٣٤٥') {

 print("Discount Granted!");

}
else {

```

```
print("No discount for you!");
```

```
}
```

في هذه المرة فانك تقوم باختبار شرطين ولكنك فقط تريد أن يكون أحدهما هو المتحقق. فإذا كان أحدهما متحقق، فإن الكود سيتم تنفيذه. وإذا كان كلا الشرطين غير متحقق، فإن PHP سوف تتحرك الى السطور التالية في الكود.

## المعاملين And و OR

هذه المعاملات تشبه تماماً المعاملان اللذان تحدثنا عنهما من قبل! فالمعامل And يشبه الرمز && و المعامل Or يشبه || . الاختلاف هنا واضح، ولكن كمبتديء، يمكنك استبدال السطر التالي:

```
$username == 'user' && $password == 'password'
```

بهذا السطر

```
$username == 'user' AND $password == 'password'
```

وهذا السطر

```
$total_spent == ١٠٠ || $special_key == 'SK١٢٣٤٥'
```

بهذا السطر

```
$total_spent == ١٠٠ OR $special_key == 'SK١٢٣٤٥'
```

دعنا نرجع الاختيار لك لتستخدم أيهما تريد. تعتبر And أسهل في القراءة من &&. وكذلك Or تعد أسهل بكثير في القراءة من ||.

ان الاختلاف، تلقائياً، يكون عند اجراء أولويات المعاملات. لقد تناولت هذا الموضوع عند الحديث عن المتغيرات، فيما قبل. المعاملات المنطقية لها ترتيب محدد، كما سيظهر قريباً في الجدول الكامل لذلك!

## - المعامل XOR

من المحتمل أن لا تستخدم هذا المعامل كثيراً. ولكنه يستخدم عندما تريد أن تختبر إذا كانت أحد القيم متحققة ولكن ليس الاثنين. لو كانت كل القيم متشابهة، فإن PHP ترى أن التعبير غير متحقق False. وإذا كان كلاهما مختلفين، إذن القيمة تكون True. افترض أنك تريد أن تختار الفائز من بين اثنين من المنافسين. فقط أحدهما سيكون الفائز. انك في هذه الحالة تحتاج الى المعامل XOR!

```
$contestant_one = true;
$contestant_two = true;

if ($contestant_one XOR $contestant_two)
{

 print("Only one wi ");

}
else {

 print("Both can't win!");

}
```

هل يمكنك أن تستنتج أي منهما الاثنين سيتم طباعته، قبل أن تقوم بتنفيذ السكريبت.

## - المعامل !

هذا المعامل يعني معاملة النفي. انك تستخدم هذا المعامل عندما تختبر شيء ما أنه ليس شيء ما آخر. ويمكنك أيضاً أن تستخدمه في عكس القيمة True أو القيمة False. على سبيل المثال، اذا كنت تريد تغيير قيمة المتغير الى True ، اذا كانت تم اعطاؤها القيمة False، والعكس. اليك الكود التالي:

```
$test_value = false;

if ($test_value == false) {
```

```
print(!$test_value);
```

```
}
```

الكود السابق سيطبع الرقم ١! (سوف ترى لماذا عندما نتعامل مع القيم المنطقية التي ستأتي في الفصل التالي.) فما نقوله هنا هو، " إذا كانت قيمة \$test\_value هي False اذن قم بتغييرها الى العكس." والعكس هنا هو القيمة True، ولذلك فانه سيحصل على هذه القيمة. ان هذا المنطق مرفوض قليلاً؟ انه يستخدم الاحتيال، ولكن قد يكون مفيداً في بعض الأحيان!

في الفصل التالي، سنتناول القيم المنطقية

الفصل الحادي عشر

القيم المنطقية في PHP

**PHP Booleans**



القيمة المنطقية هي التي تكون على إحدى حالتين. وهما اللذان تعرفان بالقيمتين True و False، في البرمجة. و تأخذ True القيمة ١، وتأخذ False القيمة ٠ صفر. ولكنك تعلن عنهم مثل باقي المتغيرات الأخرى:

```
$true_value = ١;
$false_value = ٠;
```

ويمكن استبدال القيمتين ١,٠ بالكلمات "true" و "false" (بدون استخدام علامات التقييم). ولكن ضع في اعتبارك، لو أنك فعلت ذلك. حاول أن تقوم بتنفيذ السكريبت التالي، وتري ما سيحدث:

```
<?php
```

```
$true_value = true;
$false_value = false;

print ("true_value true_value);
print (" false_value = " . $false_value);
```

```
?>
```

ما ينبغي أن تجده هو أن **true\_value** ستتطبع القيمة ١، ولكن **false\_value** لن يطبع أي شيء! الآن قم بتبديل True بالقيمة ١، و False بالقيمة ٠، في السكريبت السابق، ولاحظ ما سيتم طباعته.

القيم المنطقية أكثر شيوعاً في البرمجة، وغالباً ما ترى هذا النوع في الكود:

```
$true_value = true;

if ($true_value) {

 print("that's true");

}
```

هذه طريقة مختصرة للقول أنه "إذا كان \$true\_value يمتلك القيمة المنطقية ١ اذن ستكون الجملة متحققة True. وهذا كما هو الحال في:

```
if ($true_value == ١) {
 print("that's true");
}
```

المعامل NOT يستخدم أيضاً بكثرة مع هذا النوع من جمل IF:

```
$true_value = true;
if (!$true_value) {
 print("that's true");
}
else {
 print("that's not true");
}
```

سوف تقابل القيم المنطقية كثيراً، خلال حياتك البرمجية. انها تستحق بالفعل أن نتوقف عندها!

## الفصل الثاني عشر

أولوي      املات في PHP

## PHP Operators Precedence

وهذه هي قائمة بالمعاملات التي قابلتها فيما سبق، والترتيب الخاص بأسبقيتها. وهذا قد يسبب اختلاف، كما رأينا في المعاملات الرياضية. لا تقلق من ذلك كثيراً، إذا كنت مقتنع بأن المعاملات الرياضية والمنطقية قد تم استخدامها بطريقة صحيحة. وعلى أية حال، عليك الرجوع الى الجدول التالي لتحصل على الناتج المرغوب:

### جدول (١-٣) يوضح أولوية المعاملات في PHP

|                              |                           |
|------------------------------|---------------------------|
| <b>* / %</b>                 | <b>Highest Precedence</b> |
| <b>+ - .</b>                 |                           |
| <b>&lt; &lt;= &gt; &gt;=</b> |                           |
| <b>== !=</b>                 |                           |
| <b>&amp;&amp;</b>            |                           |
| <b>  </b>                    |                           |
| <b>And</b>                   |                           |
| <b>XOR</b>                   |                           |
| <b>OR</b>                    | <b>Lowest Precedence</b>  |

المعاملات الوحيدة التي لم تراها من قبل وقد وردت في القائمة السابقة هي المعامل **==** والمعامل **!=**.

#### - المعاملان **==** و **!=**

في الإصدارات الحديثة من PHP، تم إضافة اثنين من المعاملات: علامة يساوي الثلاثية (**===**) وعلامة تعجب، ثم علامتي يساوي (**==** و **!=**). وهذان المعاملان يستخدمان لاختبار إذا كان أحد القيم هو نفسه الآخر و نفس النوع. وكمثال على ذلك ما يلي:

```
$number = ٣;
$text = 'three';

if ($number === $text) {
 print("Same");
}
else {
 print("Not the same");
}
```

لذلك فان هذا الكود يسأل، "هل المتغيرات تتطابق تماماً؟" وحيث أن أحدهما نص والآخر رقم، فان الإجابة هي "لا No"، أو False. لن نستخدم هذه المعاملات كثيراً، وربما لاق!

في الباب التالي، سنلقي نظرة على نماذج HTML، وكيفية الحصول على البيانات منها. ويترتب على ذلك أداء مهام أخرى بجانب عملية الطباعة على الشاشة.

## السادس

# التعامل مع نماذج HTML

## أهداف الباب السادس

في نهاية هذا الباب ينبغي أن يكون الطالب قادراً على أن:

- ١- يعدد الكائنات التي يمكنه اضافتها للنموذج.
- ٢- يكتب الكود اللازم لعمل نموذج يشتمل على المربع النصي وزر التسجيل.
- ٣- يحفظ النموذج باسم BasicForm.php.
- ٤- يتعرف على الطرق Methods الأكثر شيوعاً والتي يمكن استخدامها وهي Get، Post.
- ٥- يدرك التأثير الذي تسببه الطريقة Get.
- ٦- يعطي مثال على استخدام الطريقة Get.
- ٧- يفرق بين الطريقة Get ، الطريقة Post.
- ٨- يعطي مثال على استخدام الطريقة Post.
- ٩- يتعرف على خاصية الحدث Action.
- ١٠- يدرك أنه Action ث
- ١١- يدرك كيف يتم ارسال السكريبت إلى نفس الصفحة وكان الصفحة ترسل إلى نفسها.
- ١٢- يكتب سطر الكود اللازم لتسجيل بيانات النموذج وارسالها إلى الصفحة التي يمكن ذكرها في خاصية الحدث.
- ١٣- يكتب الكود اللازم للحصول على النص الذي قام المستخدم بادخاله إلى المربع النصي.
- ١٤- يكتب الصيغة اللازمة لاستدعاء النص الذي قام المستخدم بكتابته.
- ١٥- يستخدم المنطق الشرطي لاختبار ما تم تخزينه بداخل متغير للتأكد من اسم المستخدم.
- ١٦- يكتب الكود اللازم لاختبار هل تم الضغط على زر التسجيل أم لا.
- ١٧- يكتب الكود اللازم لاختبار اسم المستخدم وفي حال كان صحيحاً يطبع عبارة Welcome back Friend وفي حال لم يكن يطبع You're not a member of this site.
- ١٨- يطبق الارسال على نفس الصفحة في كود اختبار اسم المستخدم.
- ١٩- يطبق الارسال على صفحة أخرى في كود اختبار اسم المستخدم.

- ٢٠- يكتب الصيغة التي تستخدم لإعادة إرسال التفاصيل إلى النموذج والاحتفاظ بالبيانات التي يقوم المستخدم بكتابتها.
- ٢١- يدرك أهمية الصيغة Value.



لو أن لديك خبرة قليلة في لغة HTML، فانك تعلم أن النماذج من الممكن استخدامها في التفاعل مع المستخدمين. الكائنات التي يمكن اضافتها للنموذج من الممكن أن تكون مربعات نصية، أزرار خيار، مربعات اختبار، القوائم المنسدلة، المساحات النصية، وأزرار التسجيل. نموذج HTML الأساسي بالمربع النصي وزر التسجيل يبدو هكذا:

```
<html>
<head>
<title>A BASIC HTML FORM</title>
</head>
<body>

 <FORM NAME ="form\1" METHOD ="
 " ACTION = "">

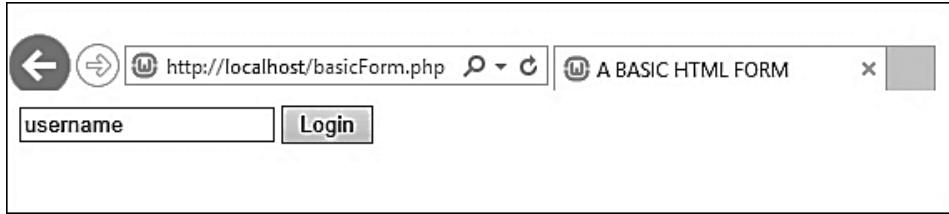
 <INPUT TYPE = " VALUE
 ="username">
 <INPUT TYPE = "Submit" Name =
 "Submit\1" VALUE = "Login">

 </FORM>

</body>
</html>
```

سوف نناقش الطريقة Method، الحدث Action وخصائص التسجيل Submit في النموذج الموضح أعلى، لأنهم الأكثر أهمية في الكود السابق.

لذلك قم بإنشاء النموذج السابق. واحفظه باسم **basicForm.php** (هذا الاسم سيكون مهم جداً!) ابدأ بتشغيل السيرفر، وتأكد من أن النموذج يتم تحميله بشكل جيد في المستعرض الخاص بك. ينبغي أن تكون قادراً على أن ترى المربع النصي وزر التسجيل. وفيما يلي الشكل الذي ينبغي أن يظهر عليه:



شكل (٥-١) يوضح المربع النصي وزر التسجيل على النموذج

إذا مرّ أي مستخدم بالموقع الخاص بك، وأراد أن يدخل، على سبيل المثال، فإنك تحتاج أن تحصل على التفاصيل من خلال مربعات نصية. وعندما تحصل على النصوص التي قام المستخدم بإدخالها، فإنك تختبر هذه البيانات وتقارنها بقائمة المستخدمين التي لديك ( وهذه القائمة عادة ما تكون مخزنة على قاعدة بيانات. لابد أن تكون لديك دراية كافية بخصائص لغة HTML، طرقها Method، و الحدث Action، و التسجيل. وسوف أوضح ذلك في الفصول القادمة التالية.



## الفصل الثاني

لغة TML صائص الطرق لنما

HTML

PHP and the Method Attribute of  
HTML Forms

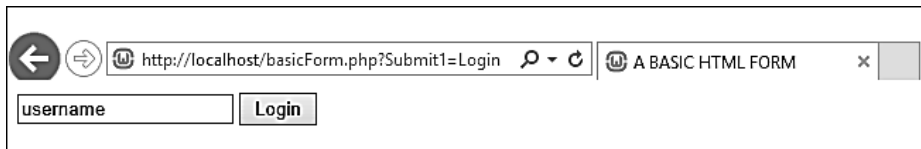
إذا ألقيت نظرة على السطر الأول من الكود الخاص بالنموذج، والذي تم تناوله في الفصل السابق، ستلاحظ أن خصائص الكيفية أو الطريقة **METHOD** هي:

```
<FORM NAME ="form\1" METHOD =" " ACTION = "">
```

ان خصائص **Method**، قد تم استخدامها لاختبار المستعرض كيف سيتم إرسال المعلومات الواردة في النموذج. والطرق **Methods** الأكثر شيوعاً التي يمكنك استخدامها هي **GET**، و **POST**. ولكن هنا الطريقة أو الكيفية **METHOD** فارغة. لذلك قم بتغييرها الى ما يلي:

```
<FORM NAME ="form\1" METHOD ="GET" ACTION
= "">
```

لترى ما التأثير الذي يسببه استخدامك للطريقة **GET**، احفظ الكود مرة أخرى ، ثم اضغط على زر التسجيل **Submit** في النموذج، وسوف ترى ما يلي:



شكل (٢-٥) يوضح تأثير **GET Method**.

الشيء الذي تلاحظه هنا هو ظهور ما يلي بعد **basicForm.php**:

**?Submit1=Login**

انه تتابع لاستخدام الكيفية **GET**. البيانات من النموذج تنتهي من شريط العنوان. ستري علامة استفهام، متبوعة ببيانات النموذج، في الشكل السابق، فان **Submit1** هو اسم الزر **NAME**، و **Login** هي القيمة **VALUE** للزر (النص المكتوب على الزر). وهذا ما سيتم ارجاعه باستخدام الطريقة أو

الكيفية GET. لقد استخدمت الطريقة أو الكيفية GET، عندما كانت البيانات التي تريد ارجاعها ليست معلومات مهمة تحتاج الى حماية.

يمكنك أيضاً استخدام POST كطريقة Method، بدلاً من GET، في الفصل التالي سنتناول الاختلاف.

## الفصل الثالث

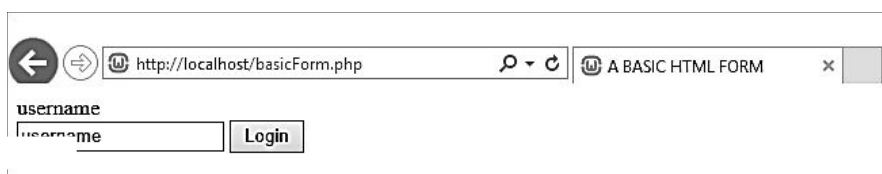
# لغة PHP      ية Post في نما HTML

## PHP and the Post Attribute of HTML Forms

في الفصل السابق، شاهدت ما الذي يحدث، في شريط العنوان الخاص بالمستعرض عندما استخدمت الكيفية GET لنموذج البيانات. البديل لاستخدام GET هو استخدام POST. قم بتغيير السطر الأول للنموذج الى ما يلي:

```
<FORM NAME = "form1" METHOD
= "POST" ACTION = "">
```

قم باغلاق المستعرض، ثم قم بفتحه واستعادته مرة أخرى. قم بتحميل الصفحة basicForm.php مرة أخرى، ثم اضغط على الزر. سيبدو شريط العنوان لديك كما يلي:



شكل (٣-٥) يوضح اختفاء Login=Submit؟ نتيجة استخدام POST.

ستلاحظ من الشكل السابق أن الجزء Login=Submit؟ الذي كان يظهر في شريط العنوان قد اختفى! وذلك لأنك قد استخدمت POST وهي تعني أن بيانات النموذج لن يتم إلحاقها بالعنوان الموجود في شريط العنوان والذي يراه الجميع. سوف تستخدم كلاً من الطريقتين ، ولكن اعتماداً على المشروع: إذا لم تكن البيانات سرية اذن يمكنك استخدام GET، والا يمكنك استخدام POST.

وثمة خاصية أخرى هامة للنموذج وهي الحدث Action، سوف نتناول الحدث Action في الفصل التالي.



## الفصل الرابع

# لغة PHP ية الحدث ction في نماذج HTML

## PHP and the Action Attribute of HTML Forms

خاصية الحدث هي خاصية مهمة. وهي تعني، " تحديد المكان الذي تريد من النموذج أن يرسل اليه؟". ولو أنك لم تستخدمه، فسوف يرسل النموذج البيانات الى أي مكان. يمكنك ارسال بيانات النموذج الى سكريبت PHP آخر، الى نفس PHP سكريبت، لعنوان بريد الكتروني، أو أي سكريبت لنموذج آخر.

في PHP، هناك أسلوب معروف وهو ارسال السكربت الى نفس الصفحة للنموذج المفتوح، وبعبارة أخرى، أنها ترسل الى نفسها. وسوف نستخدم هذه الأسلوب أولاً، ولكن يمكنك رؤية الأسلوبين فيما بعد في الحدث.

لذلك فانك تحتاج الى تغيير النموذج الذي قمت بانشائه في الفصل السابق، الذي تم تسميته basicForm.php. قم بتغيير سطر الحدث Action الى ما يلي:

```
<Form Name ="form\ " Method
="POST" ACTION = "basicForm.php">
```

لذلك فانك سوف ترسل بيانات النموذج الى نفس الصفحة. وكأن الصفحة ترسل الى نفسها. سنضع بعض من PHP على الصفحة للتعامل مع بيانات النموذج. ولكن الآن، احفظ الكود ثم اضغط على الزر Submit. انك لا ترى شيئاً ما قد اختلف، ولكنك لا ترى أيضاً أي رسائل خطأ!

في الحال عندما تقوم باعداد خاصية الحدث Action، يمكنك بعد ذلك الضغط على Submit، وهذا ما ستراه في الفصل التالي.

## الفصل الخامس

# لغة PHP و تسجيل Submit نماذج HTML

## PHP and the Submit Button of HTML Forms

في لغة HTML فان زر التسجيل Submit، يستخدم لتسجيل بيانات النموذج وارسالها الى السكريبت المذكور في خاصية الحدث ACTION كما يلي:

```
<Form Name ="form\ " Method ="POST" ACTION =
"basicForm.php">
```

ولذلك فان الصفحة التي يتم ذكرها في خاصية الحدث وهي basicForm.php. وتسجيل هذا السكريبت، فانك تحتاج فقط الى زر تسجيل HTML:

```
<INPUT TYPE = "Submit" Name = "Submit\ "
VALUE = "Login">
```

انك لا تحتاج الى أداء أية مهمة خاصة مع الزر Submit، كل إجراءات التسجيل تحدث تلقائياً. وهذا لأنه تم ضبط الحدث ACTION الخاص بزر SUBMIT، ولذلك فان بياناتك سيتم ارسالها الى المكان الذي تم تحديده. ولكن خاصية NAME للزر Submit تكون في متناول اليد. ويطلق على زر التسجيل Submit button هنا "Submit\ "، ولكن يمكنك تسميته بأي اسم تريده.

الآن بعد أن تعرفت على كل من الكيفية METHOD ، والحدث ACTION و التسجيل SUBMIT. يمكننا الانتقال الى كيفية معالجة البيانات التي قام المستخدم بادخالها، وكيف يمكن الحصول على القيم من خلال مربعات النصوص التي أضفتها وهذا ما سنتناوله في الفصل التالي.

## الفصل السادس

لغة PHP والمربعات النصية في نماذج HTML

# PHP and Text Boxes In HTML Forms

من الفصل السابق، فإن **basicForm.php** لها كيفية METHOD وحدث ACTION قد تم اعدادهم. سنستخدم ذلك لنعالج النصوص التي قام المستخدم بادخالها الى مربع نصي. خاصية METHOD تخبرك عن الكيفية التي يتم ارسال بيانات النموذج بها، وخاصية الحدث ACTION تخبرك بالمكان الذي يتم ارسالها اليه.

للحصول على النص الذي قام المستخدم بادخاله الى المربع النصي، فإن المربع النصي يحتاج أولاً الى خاصية الاسم NAME. ثم بعد ذلك يمكنك اخبار PHP باسم المربع النصي NAME الذي تريد أن تتعامل معه. المربع النصي الذي استخدمناه ليس له اسم بعد، ولذلك قم بتغيير كود HTML الى ما يلي:

```
<INPUT TYPE = "Text" VALUE ="username" NAME =
"username">
```

اسم NAME المربع الـ **username** . هذا هو الاسم سنستخدمه في سكريبت

ولكي تستعيد البيانات من عنصر من نموذج HTML، فانك تستخدم الصيغة الغريبة التالية:

```
$_POST['formElement_name'];
```

ويمكنك تخصيص هذا الى متغير:

```
$Your_Variable =
$_POST['formElement_name'];
```

وقبل أن أتناول الصياغة بالشرح بأكملها، قم بإضافة سكريبت PHP التالي الى كود HTML الذي سبق وأن تعاملنا معه. تأكد من إضافة الجزء المخصص للرأس HEAD والخاص بـ HTML (الجزء الذي ستضيفه للكود مكتوب بخط أسود عريض):

```
<html>
<head>
<title>A BASIC HTML FORM</title>
```

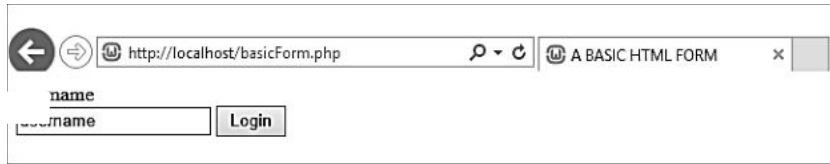
```
<?PHP
```

```
$username = $_POST['username'];
print ($username);
```

```
?>
```

```
</head>
```

احفظ الكود مرة أخرى، واضغط على زر تسجيل submit لكي تنفذ السكريبت. (لا تقلق اذا ظهرت لك رسالة خطأ عن وجود فهرس غير معرف "Undefined index". اضغط على الزر على اية حال.) وينبغي أن ترى ما يلي أعلى المربع النصي:



شكل (٤-٥) يوضح ظهور محتويات المربع النصي فوق المربع النصي.

قم بحذف النص "username" من المربع النصي، ثم قم بالضغط على الزر مرة أخرى. ستلاحظ أن النص الجديد الذي تكتبه ينبغي أن يظهر أعلى المربع النصي. المربع النصي نفسه لا يزال يحتوي على "username" بداخله. هذا لأن المربع النصي يتم استعادته مرة أخرى عندما يتم ارجاع البيانات الى المستعرض. ولكن خاصية القيمة Value للمربع النصي هي التي يتم عرضها.

### لذلك فكيف تعمل؟

الدالة `$_POST[ ]` هي دالة مبنية داخلياً، ويمكنك استخدامها للحصول على البيانات من النموذج. لو أن لديك الكيفية `METHOD = "GET"` في النموذج، فأنك لابد أن تستخدم هذا كبديل:

```
$username = $_GET['username'];
```

أي أنك تبدأ بعلامة الدولار ( \$ ) وعلامة الشرطة التحتية ( \_ ). ثم تأتي بعد ذلك الكيفية METHOD التي تريد أن تستخدمها، سواء كانت POST أو GET. وتحتاج بعدها إلى كتابة الأقواس المربعة، اكتب اسم العنصر NAME الذي تريده من عناصر نموذج HTML – وهو اسم المستخدم في هذا المثال **username**.

```
$_POST['username'];
```

وبالطبع، فإنك تحتاج إلى الفاصلة المنقوطة لانتهاء هذا السطر.

أيًا كانت القيمة VALUE لعنصر HTML فإنه سيتم إرجاعها. ويمكنك بعد ذلك تخصيصها إلى متغير:

```
$username = $_POST['username'];
```

ولذلك فإن PHP، سوف تبحث عن عنصر النموذج بالاسم NAME وهو **username**. ثم يحدد صيغة القيمة VALUE لعنصر النموذج ثم يقوم بإرجاع هذه القيمة لك لكي تستخدمها وتتعامل معها.

إلى هذه اللحظة، فإن كل ما قمت به هو استرجاع ما قام المستخدم بإدخاله وطباعته إلى الصفحة. ولكن يمكن استخدام القليل من المنطق الشرطي لاختبار ما هو مخزن بداخل المتغير. وكمثال على ذلك، قم بتغيير PHP إلى ما يلي:

```
$username = $_POST['username'];

if ($username == "letmein") {

 print ("Welcome back, friend!");

}
else {

 print ("You're not a member of this
 site");

}
```



الآن، أنت تختبر لتري هل المستخدم قام بإدخال النص "letmein". وإذا حدث ذلك فإن، اسم المستخدم صحيحاً، وإذا لم يكن كذلك، سيطلع رسالة أخرى.

قم بتنفيذه لتري ما الذي يحدث. وعندما تقوم بتحميل الصفحة، وقبل أن تضغط على الزر، ربما ترى النص "You're not a member of this site" تظهر أعلى المربع النصي. هذا لأنك لم تختبر إذا كان زر التسجيل Submit button على النموذج قد تم الضغط عليه أم لا.

في الفصل التالي، سوف تري كيفية اختبار ما إذا كان قد تم الضغط على زر التسجيل Submit button أم لا.

## الفصل السابع

أزرار التسجيل      Submit butt في P

**PHP Submit buttons**

في الفصل السابق، رأيت كيف يمكنك الحصول على النص من المربع النصي عند الضغط على زر التسجيل Submit button الموجود على النموذج. على الرغم من ذلك، فإنه عند بداية تحميل الصفحة فإن النص يتم عرضه.

والسبب في أن النص يتم عرضه في بداية تحميل الصفحة هو أن السكريبت يتم تنفيذه سواءاً تم الضغط على الزر أم لا. وهذه هي المشكلة التي تواجهها عندما يكون سكريبت PHP في نفس الصفحة مثل HTML. وسوف يتم التسجيل بها نفسها في خاصية الحدث ACTION attribute.

وللتغلب على ذلك، يمكنك أن تقوم بعمل اختبار بسيط باستخدام جملة IF. ما الذي تريده هو اختبار ما اذا كان زر التسجيل submit button قد تم الضغط عليه أم لا، استخدم السطر التالي:

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

انها تبدو الآن غامضة الـ ولكنها بالفعل تتكون من ثلاثة أجزاء

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

أن تعلم جيداً الصيغة العامة لجملة IF. ولكن بين الأقواس الدائرية، لدينا **isset( )**، انها دالة مبنية داخلياً والتي تختبر ما اذا كان المتغير قد تم الضغط عليه أم لا. وبين الأقواس، فإنك تكتب ما تريد أن تختبره **isset( )**. بالنسبة لنا هنا يتم كتابة **\$\_POST['Submit']**، وهذا يعني اذا قام المستخدم بعمل تنشيط refresh للصفحة، اذن لن تظهر أي قيمة لزر التسجيل Submit. فاذا ما ضغط المستخدم على زر التسجيل، فإن PHP ستعيد القيمة أتوماتيكياً. غير السكريبت الموجود في الفصل السابق الي ما يلي ثم حاول أن تنفذه:

```
if (isset($_POST['Submit'])) {

 $username = $_POST['username'];

 if ($username == "letmein") {
```

```
 print ("Welcome back, friend!");
 }
 else {
 print ("You're not a member of this
site");
 }
}
```

لاحظ أنه لا بد من كتابة الأقواس في مواضعها الصحيحة سواءاً الدائرية،  
والمربعة و المعقوفة. وإذا نسيت واحدة فستظهر لك رسالة خطأ!

في الفصل التالي، سترى كيف يمكنك التسجيل في نموذجك الي سكريبت  
PHP في صفحة أخرى



## الفصل الثامن

خاصية الد ACTION attribut

**PHP و HTML**

**The HTML ACTION attribute and  
PHP**

لست مضطراً الى تسجيل بيانات النموذج في نفس صفحة PHP، كما كنت تفعل قبل ذلك. يمكنك ارساله الى صفحة PHP مختلفة تماماً. ولتري كيف يمكنك القيام بذلك، حاول أن تنفذ ما يلي:

قم بانشاء الصفحة التالية، ثم قم بتسميتها **basicForm۲.php**. وهذا هو كود HTML. لاحظ خاصية الحدث ACTION attribute هنا:

```
<html>
<head>
<title>A BASIC HTML FORM</title>
</head>
<body>

<Form name ="form۱" Method
="POST" Action ="submitForm.php">

 <INPUT TYPE T" VALUE
 ="username" Name ="username">
 <INPUT TYPE = "Submit" Name =
 "Submit۱" VALUE = "Login">

</FORM>

</body>
</html>
```

الآن قم بانشاء الصفحة التالية، ثم قم بتسميتها **submitForm.php**:

```
<?PHP

$username = $_POST['username'];

if ($username == "letmein") {

 print ("Welcome back, friend!");
```

```

 }
 else {

 print ("You're not a member of this
 site");

 }

?>

```

في سكريبت PHP السابق، لاحظ كيف أنه لا يوجد أي دلائل على HTML. وأننا قد استغنيا عن الكود الذي يختبر ما اذا كان قد تم الضغط على زر التسجيل Submit button أم لا. ذلك لأنه لا يوجد سكريبت PHP متروك في الصفحة السابقة. يتم تنفيذ الكود فقط اذا ما تم الضغط على زر Submit.

ان ارسال بيانات النموذج الى صفحة سكريبت PHP مختلفة هي طريقة للحفاظ على كل من HTML و PHP منفصلين. ولكن هناك مشكلة ستواجهك، وربما تكون قد لاحظتها: يتم تنفيذ السكربت في صفحة جديدة. وهذا يعني أن النموذج لن يظهر!

سوف نحفظ بكود HTML و PHP سوياً. ولكن سيأتي عليك أوقات تحتاج فيها أن ترسل بيانات النموذج الى صفحة PHP مختلفة، كما ستري في الفصول التالية.





## الفصل التاسع

### استعادة البيانات في PHP

### PHP Data Retention

في الفصول السابقة، تعلمت كيف يمكنك انشاء نموذج HTML. وتعلمت كيفية الحصول على النصوص المسجلة في المربع النصي على النموذج، ولكن هناك مشكلة.

عند تسجيل النموذج **basicForm.php**، فان البيانات التي قام المستخدم بادخالها قد تم مسحها. وستترك فقط القيمة **VALUE** التي يتم اعدادها في HTML. بالنسبة لك، فان اسم المستخدم **username** تم الاحتفاظ به ظاهراً في المربع النصي عندما تضغط على الزر. ويمكنك الاحتفاظ بالبيانات التي قام المستخدم بادخالها بسهولة.

هذا السكريبت سوف يظهر كما يلي، قم بكتابة هذا السكريبت وتسميته **basicForm.php** ثم قم بتنفيذه على السيرفر.

```
<html>
<head>
 >A BASIC HTML FORM</

<?PHP
if (isset($_POST['Submit'])) {
 $username = $_POST['username'];

 if ($username == "letmein") {
 print ("Welcome back, friend!");
 }
 else {
 print ("You're not a member of this site");
 }
}
?>
</head>
<body>
<Form name = "form" Method = "POST" Action
= "basicForm.php">
<Input Type = "text" Value = "username" Name
= "username">
<Input Type = "Submit" Name = "Submit" Value =
```

```
"Login">
</FORM>
</body>
</html>
```

إذا نظرت الى خاصية القيمة VALUE للمربع النصي في HTML من السكريبت السابق، ستري أنه قد تم اعدادها الى **"username"**. لأن النموذج في هذه الحالة يرسل الى نفسه، وهذه القيمة سيتم الاحتفاظ بها وإعادة ظهورها في المربع النصي عندما يتم تسجيل الصفحة. ولو أنك تركت خاصية القيمة Value attributes فارغة، فان أي شيء سيقوم المستخدم بادخاله لن يظهر. وهذا يمكن أن يكون مجهولاً، لو أنك تسأل المستخدم أن يحاول مرة أخرى. والأفضل هو ارسال القيم التي قام المستخدم بادخالها.

ولكي تعيد ارسال التفاصيل الى النموذج ، والاحتفاظ بالبيانات التي يقوم المستخدم بكتابتها، يمكن أن تستخدم يلي:

```
VALUE="<?PHP print ame ; ?>"
```

وبعبارات أخرى، فان خاصية القيمة VALUE attribute هي الآن سطر من سطور كود PHP. سطر الكود التالي:

```
<?PHP
print $username ;
?>
```

انه سطر صعب القراءة، وذلك لأنه كله في سطر واحد.

تحتاج أيضاً الى تغيير كود PHP في جزء الرأس HEAD ليشتمل على جملة Else:

```
if (isset($_POST['Submit'])) {
 $username = $_POST['username'];
 if ($username == "letmein") {
```

```

 print ("Welcome back, friend!");
 }
 else {

 print ("You're not a member of this
 site");

 }

}
else {

 $username ="";

}

```

في جملة else في النهاية، عليك فقط ضبط قيمة المتغير الذي يدعى \$username عندما لا يتم الضغط على الزر، على سبيل المثال عندما يتم تنشيط الصفحة The page is refreshed .

على الرغم من ذلك، هناك بعض الاحتياطات الأمنية المرتبطة بمربعات النصوص ( وكذلك عناصر النموذج الأخرى). ولذلك فانك سوف ترى طريقة للأمن أكثر عند التعامل مع هذه المربعات النصية في الفصول التالية.

ولكن السطر الجديد من HTML للمربع النصي يتم كتابته هكذا:

```

<INPUT TYPE = 'TEXT' Name ='username'
VALUE="<?PHP print $username ; ?>">

```

وبعبارات أخرى، سنقوم بطباعة خاصية القيمة attribute VALUE في كود PHP.

الآن لقد تعلمت القليل عن الحصول على القيم من نماذج HTML، واليك بعض التدريبات على ذلك.

**تدريب**

قم بإضافة اثنين من المربعات النصية وزر تسجيل الى نموذج HTML، دع المستخدم أن يدخل الاسم الأول واللقب. عندما يتم الضغط على الزر، اطبع اسم الشخص بالكامل. لا تقلق بشأن ما هو الذي يظهر بداخل المربعات النصية بعد الضغط على الزر.

### **تدريب**

باستخدام نفس النموذج في التدريب السابق، اعرض الاسم الأول و اللقب في المربعات النصية، بدلاً من طباعتهم.

### **تدريب**

افترض أنه يوجد خمسة مستخدمين لموقعك. قم بإنشاء نموذج لتختبر ما اذا كان الزائر هو واحد من الخمسة مستخدمين أم لا. اعرض الرسالة المناسبة.

في الفصل التالي، سنلق على كيفية التعامل مع أزرار الخيا نموذج HTML.

## الباب السابع

### الحلقات البرمجية

## أهداف الباب السابع

بعد الانتهاء من دراسة هذا الباب ينبغي أن يكون الطالب قادراً على أن:

- ١- يتعرف على حلقات التكرار في Php.
- ٢- يدرك متى تستخدم حلقات التكرار.
- ٣- يكتب الكود اللازم لطباعة الأعداد من ١ إلى ١٠.
- ٤- يحدد الصيغة العامة لحلقات For.
- ٥- يميز بين معاملات For الثلاثة و أماكن كتابتها.
- ٦- يدرك ماذا تعني Start\_Value.
- ٧- يدرك ماذا تعني End\_Value.
- ٨- يدرك معنى Update expression.
- ٩- يدرك فيما يستخدم رمز الزائد المزدوج (++).
- ١٠- يدرك فيما يستخدم رمز الناقص المزدوج (--).
- ١١- يدرك أهمية الأقواس المعقوفة داخل حلقات For.
- ١٢- يكتب الـ م لطباعة جدول الضرب كمثال على التكرار باستخدام لغة HTML.
- ١٣- يكتب الكود اللازم لطباعة جدول الضرب كمثال على حلقات التكرار دون استخدام لغة HTML.
- ١٤- يفرق بين الصيغة العامة لحلقات For، وحلقات التكرار من نوع While.
- ١٥- يتعرف على الصيغة العامة لحلقات While.
- ١٦- يكتب الكود اللازم لزيادة قيمة المتغير الذي يدعى \$counter.
- ١٧- يدرك فيما تستخدم حلقات While ومتى يمكن استخدامها.
- ١٨- يدرك فيما تستخدم الشرطتين المائلتين //.
- ١٩- يكتب برنامج باستخدام حلقة While لطباعة جدول ضرب.
- ٢٠- يفرق بين حلقات While و حلقات Do.....While.
- ٢١- يدرك فيما تستخدم جملة التوقف Break.
- ٢٢- يعطي مثال على استخدام جملة التوقف Break.



## الفصل الأول

استخدم لقات في PHP

**PHP For Loops**

ماذا تعني الحلقات؟ الحلقة هي عبارة عن خطوات يتم تكرارها مرة تلو الأخرى. اذا طلبت منك أن تدبر اصبعك لمرات عديدة، فان هذا الأمر لا يعني مشكلة بالنسبة لك (الا إذا لم يكن لديك أصابع!) هكذا الأمر بالضبط في البرمجة. ماعدا أن الحلقات البرمجية ستدور باستمرار وتستمر في تنفيذ الخطوات الى أن تخبر البرنامج بأن يتوقف. تحتاج أيضاً الى أن تخبر البرنامج بأمرين آخرين – من أين تبدأ الحلقة، وماذا يفعل بعد انتهاء الحلقة الأولى ( والذي يطلق عليه تعبير التحديث).

يمكنك كتابة البرنامج بدون استخدام الحلقات التكرارية. لكن الأمر يكون أسهل باستخدامهم. فكر فيما يلي:

أنت تحتاج الى جمع الأعداد من ١ إلى ٤:  $1 + 2 + 3 + 4$ . يمكنك أن تؤديها هكذا:

```
$answer = ١ + ٢ + ٣ + ٤;
print $answer;
```

انها بسيطة جداً، كما رأيت. ولا تحتاج الى كود كبير، أيضاً. ولكن ماذا لو كنت تحتاج الى جمع ألف عدد؟ هل ستقوم بالفعل بكتابتهم جميعاً كما كتب الأرقام الأربعة سابقاً؟ انها عملية شاقة جداً. الحلقات التكرارية سوف تجعل الأمر أكثر سهولة. انك تستخدم الحلقات التكرارية عندما تحتاج الى تنفيذ نفس الكود مرات ومرات.

سنناقش بعض الحلقات البرمجية، ولكن حلقة For هي النوع الأكثر استخداماً، ولذلك سأناقشها معك أولاً.

## الحلقات التكرارية من النوع For Loops

فيما يلي سكريبت PHP يستخدم حلقات التكرار من النوع For Loop. قم بكتابة هذا السكريبت ثم حفظه، ثم قم بتنفيذه.

```
<?PHP
```

```

$counter = ٠;
$start = ١;

for($start; $start < ١١; $start++) {

 $counter = $counter + ١;
 print $counter . "
";

}

?>

```

ماذا حدث عند تنفيذ الكود؟ ينبغي أنك قد رأيت الأعداد من ١ إلى ١٠ مطبوعة في صفحة المستعرض.

الصيغة العامة لاستخدام حلقات التكرار من النوع For Loop تكون هكذا:

```

for (start value; end value; update
expression) {

}

```

الأمر الأول الذي عليك اتباعه هو أنك تحتاج الى كتابة اسم الحلقة التي تستخدمها، وفي هذه الحالة هي FOR. وبين الأقواس الدائرية، تقوم بكتابة الثلاثة معاملات وهي:

## القيمة الابتدائية Start Value

المعامل الأول هو اخبار PHP بالقيمة الأولى أو الابتدائية للحلقة. وبعبارة أخرى، بداية الحلقة ستكون من أي عدد؟ لقد استخمت في الكود السابق ما يلي:

```

$start = ١;

```

لقد خصصت القيمة ١ للمتغير الذي يدعى \$start مثل كل المتغيرات، يمكنك اختيار الاسم الذي تريده للمتغير. والاسم شائع الاستخدام للمتغير الذي يحمل القيمة الابتدائية هو |. يمكنك اعداد وضبط الشرط الأول قبل بداية الحلقة، كما فعلت في الكود السابق:

```
$start = ١;
```

```
for($start; $start < ١١; $start++) {
```

أو يمكنك تخصيص القيمة الابتدائية بداخل الحلقة في الكود كما يلي:

```
for($start = ١; start < ١١; start++) {
```

وفي كلتا الحالتين فإن النتيجة تكون نفسها – والعدد الأول لهذه الحلقة هو ١.

## القيمة النهائية End Value

الخطوة التالية، هي اخبار PHP متى ستنتهي الحلقة. والقيمة النهائية ربما تكون عدد، أو قيمة منطقية Boolean value ، أو قيمة حرفية، الخ. وفي المثال السابق أن تخبر PHP أن تستمر في تنفيذ وتكرار الحلقة حتى تصبح قيمة المتغير \$start أقل من ١١.

```
for($start; $start < ١١; $start++) {
```

عندما تصبح قيمة \$start هي ١١ أو أكثر، سنقوم PHP بانهاء الحلقة.

## تحديث التعبير Update Expression

الحلقات تحتاج الى طريقة للحصول على الرقم التالي في السلسلة. لو أن الحلقة لم تقدر على تحديث القيمة الابتدائية، فانها ستتوقف عند القيمة الابتدائية. فاذا لم تقم بتحديث القيمة الابتدائية، فإن الحلقة التي استخدمتها في

الكود السابق ستتوقف عند العدد ١. وبعبارة أخرى، فانك تحتاج الى اخبار الحلقة كيف يمكنها التكرار لمرات ومرات. ولقد استخدمت ذلك:

## **\$start++**

في كثير من لغات البرمجة (وكذلك لغة PHP) فان رمز الزائد المزدوج (++) يعني عملية الزيادة (أي زيادة القيمة بمقدار واحد). انها طريقة مختصرة فقط للتعبير عن ما يلي:

## **\$start = \$start + ١**

يمكنك الاستمرار في الحلقة ( بالانقاص ) وذلك باستخدام رمز الناقص المزدوجة (--)، ولكني لن أستخدمها الآن.

ولذلك فان الحلقة بأكملها تعني " أن تبدأ الحلقة بالقيمة ١، وتظل مستمرة طالما القيمة الابتدائية أقل من ١١. بزيادة القيمة الابتدائية بالقيمة ١ في كل مرة تدور فيها الحلقة.

كل مرة تدور فيها الحلقة، فان الكود المكتوب بين الأقواس المعكوفة { } يتم تنفيذه:

```
$counter = $counter + ١;
print
```

```
$counter . "
";
```

لاحظ أنك فقط تقوم بزيادة قيمة متغير العداد بمقدار ١ كل مرة تدور فيها الحلقة. بالضبط كما حدث مع القيمة الابتدائية للمتغير. لذلك فانه يمكنك أن تكتبها هكذا:

## **\$counter ++**

التأثير سيكون نفس الشيء. لأن هذا التعبير سيضبط قيمة متغير العداد \$counter الى ١١ خارج الحلقة ( انها في الوقت الحالي تساوي صفر (\$counter = ٠). قم باستخدام - \$counter بداخل الحلقة، أي استخدم ( علامة الناقص المزدوجة). هل يمكنك أن تخمن ما الذي سوف يحدث؟ هل سيطب PHP أي شيء؟ قم بحفظ الكود الذي كتبتة؟

وللتدريب أكثر على حلقات التكرار من النوع For Loop، ستتعلم كيف  
تكتب كوداً لجدول ضرب في الفصل التالي.

## الفصل الثاني

برنامج جد رب باستخدام HP

### A PHP Times Table Program

في الفصل السابق، رأيت ماذا تعني حلقات التكرار وكيفية استخدام الحلقات من نوع For ، في الفصل التالي، ستقوم بكتابة برنامج جدول ضرب لتوضيح كيفية عمل الحلقات.

قم بتسمية الكود التالي باسم **TimesTable.php**:

```
<html>

<head>

<title>A Times Table Programme</title>

<?PHP

 $times = ٢;

if (isset($_POST['Submit'])) {

 $start = $_POST['txtStart'];

 $end = $_POST['txtEnd'];

 $times = $_POST['txtTimes'];

 for($start; $start <= $end; $start++) {

 $answer = $start * $times;
```



```
 print $start . " multiplied by " . $times . " = "
 . $answer . "
";
```

```
 }
```

```
}
```

```
?>
```

```
</head>
```

```
<body>
```

```
<FORM NAME = frmOne Method = "POST" Action =
"timesTable.php">
```

```
Start Number: <INPUT TYPE = Text NAME = txtStart
SIZE = 10 value = "\ ">
```

```
End Number: <INPUT TYPE = Text NAME = txtEnd
SIZE = 10 value = "\ . ">
```

```
Multiply By: <INPUT TYPE = Text NAME = txtTimes
SIZE = 10 value = "<?PHP print $times; ?> ">
```

<P>

<INPUT TYPE = Submit Name = Submit\ VALUE =  
"Do Times Table">

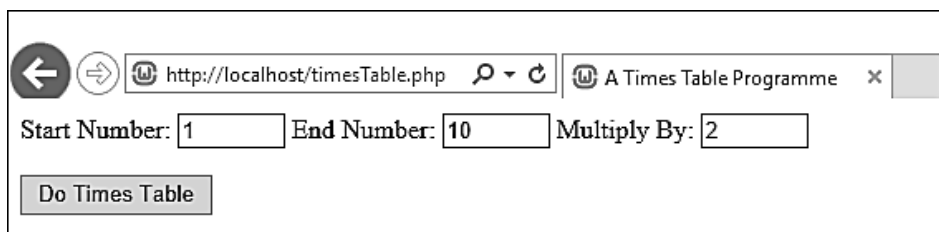
<P>

</FORM>

</body>

<html>

عندما تقوم بكتابة هذا الكود وحفظه وتحميله في المستعرض فإنه يبدو كما هو  
موضح في الشكل التالي:



The screenshot shows a web browser window with the address bar displaying 'http://localhost/timesTable.php'. The page title is 'A Times Table Programme'. The form contains three input fields: 'Start Number' with the value '1', 'End Number' with the value '10', and 'Multiply By' with the value '2'. Below these fields is a button labeled 'Do Times Table'.

شكل (٦-١) يوضح نموذج جدول الضرب للعدد ٢ في الأعداد من ١ إلى ١٠.

كل ما عليك فعله هو الحصول على القيم من المربعات النصية وإنشاء  
برنامج جدول ضرب. عند الضغط على الزر، ستكون المخرجات كما يلي:

← → http://localhost/timesTable.php A Times Table Programme ×

1 multiplied by 2 = 2  
 2 multiplied by 2 = 4  
 3 multiplied by 2 = 6  
 4 multiplied by 2 = 8  
 5 multiplied by 2 = 10  
 6 multiplied by 2 = 12  
 7 multiplied by 2 = 14  
 8 multiplied by 2 = 16  
 9 multiplied by 2 = 18  
 10 multiplied by 2 = 20

Start Number:  End Number:  Multiply By:

شكل (٦-٢) يوضح جدول ضرب ٢ والذي سيظهر عند الضغط على الزر.

وبعبارات أخرى، عند  
 على الزر سيطلع جدول الضرب  
 الصفحة. يمكنك عمل ج  
 ب أخرى مختلفة، بالاعتماد على الق  
 سيتم إدخالها في المربعات النصية. وسأتناول كود جدول الضرب في الفصل  
 التالي.

## الفصل الثالث

الكود الخاص بجدول الضرب في PHP

**Code for a PHP Times  
Table**

الكود الذي استخدمته في جدول الضرب في الفصل السابق يستخدم حلقة من النوع For Loop. بداية الحلقة سوف تأتي من المربع النصي الرقم الأول، ونهاية الحلقة ستأتي من المربع النصي الرقم النهائي End Number textbox. وهذا هو الكود بأكمله (بدون استخدام لغة HTML):

```
<?PHP
```

```
$times = ٢;
```

```
if (isset($_POST['Submit'])) {
```

```
 $start = $_POST['txtStart'];
```

```
 $end = $_POST['txtEnd'];
```

```
 $times = $_POST['txtTimes'];
```

```
 for($start; $start <= $end; $start++)
```

```
 {
```

```
 $answer = $start * $times;
```

```
 print $start . " multiplied by " .
```

```
 $times . " = " . $answer .
```

```
 "
";
```

```
 }
```

```
}
```

```
?>
```

## شرح الكود Code Explanation

نحتاج الى العدد التالي من المربعات النصية على النموذج، ولذلك فاننا نبدأ بما يلي:

```
$times = ٢;
```

```
if (isset($_POST['Submit'])) {
```

```
$start = $_POST['txtStart'];
$end = $_POST['txtEnd'];
$times = $_POST['txtTimes'];
```

```
}
```

السطر الأول يضع قيمة فقط للمتغير الذي يدعى \$times. وبسبب ذلك فإن المربع النصي "Multiply By" سيتخذ القيمة الافتراضية بمجرد تحميل الصفحة.

ثم تستخدم الدالة **isset( )** مرة أخرى، فقط لتختبر هل المستخدم قد ضغط على زر التسجيل Submit button. كما رأيت من قبل في الفصل السابق.

وللحصول على القيم من المربعات النصية، فإنه يمكنك أن تستخدم ما يلي:

```
$start = $_POST['txtStart'];
$end = $_POST['t '];
$times = $_POST['txtTimes'];
```

وهذا الكود أيضاً تعلمته في الفصل السابق. انك فقط تخصص القيم من المربعات النصية الى متغير جديد باستخدام `$_POST[]`. وبين الأقواس المربعة، قمت بكتابة الاسم NAME لمربعات HTML النصية. لذلك فإن هذا سيعطيك القيم التي قام المستخدم بادخالها الى النموذج. ثم تأتي بعد ذلك الحلقة For Loop كما تعلمتها من قبل.

```
for($start; $start <= $end; $start++) {
```

```
 $answer = $start * $times;
```

```
}
```

دعنا ننظر مرة أخرى الى السطر الأول:

```
for($start; $start <= $end; $start++) {
```

وفيه لديك قيمة ابتدائية للحلقة start value ، وقيمة نهائية end value ، وكذلك تعبير التحديث update expression. فالقيمة الابتدائية تأتي من

المتغير الذي يدعى `$start`. وهذا سيكون أي رقم يقوم المستخدم بادخاله في المربع النصي الأول. والقيمة الافتراضية له هي ١. انظر على المكتوب في القيمة النهائية:

### **`$start <= $end`**

القيمة النهائية عندما تكون القيمة المخزنة داخل المتغير الذي يدعى `$start` تكون أقل من أو يساوي القيمة المخزنة بداخل المتغير الذي يدعى `$end`. وهذا الكود صحيح ويعمل لأنك تقوم بزيادة قيمة المتغير `$start` في كل مرة تدور فيها الحلقة. المتغير الذي يدعى `$end` هو قيمة ثابتة، ويتم الحصول عليها من خلال المربع النصي على النموذج.

الجزء الأخير من الكود هو تعبير التحديث `update expression`. والذي يخبر PHP لزيادة قيمة `$start` (القيمة الابتدائية) في كل مرة تدور فيها الحلقة:

### **`$start++`**

علامة زائد المزدوجة (`++`) تعني إضافة العدد ١ الى الرقم المخزن في `$start`.

وهذا هو جوهر الحلقات من النوع `For loops`: كل ما عليك فعله هو تزويدها بالقيمة الابتدائية، القيمة النهائية، وكيف يمكن التحديث للقيمة في كل مرة تدور فيها الحلقة.

الكود الذي تقوم بكتابته داخل الحلقة، هو الكود الذي يتم تنفيذه في كل مرة تدور فيها الحلقة، وهو في الكود السابق:

### **`$answer = $start * $times;`**

تذكر أن، المتغير `$times` يحتفظ بجدول الضرب، جدول ضرب ٢ هو الافتراضي. وهذه القيمة يتم ضربها في أي قيمة يتم تخزينها داخل المتغير `$start`. وفي كل مرة تدور الحلقة، فإن `$start` سيحصل على قيمة مختلفة – الأول ١، ثم ٢، ثم ٣، الخ. ثم يتم تخزين الإجابة في المتغير الذي أطلقنا عليه اسم `$answer`. ولذلك فانه بالفعل يقوم بأداء ما يلي:

```
$answer = ١ * ٢;
$answer = ٢ * ٢;
$answer = ٣ * ٢;
etc
```

أخيراً، يمكنك عرض النتائج على الصفحة كما يلي:

```
print $start . " multiplied by " . $times
 . " = " . $answer . "
";
```

انها فقط عملية ربط أو دمج في جملة الطباعة. هل يمكنك معرفة المهام التي تؤديها هذه الأجزاء؟

هذا هو جدول الضرب الذي قمت باعداده بنفسك، اذا كان لديك أطفال، اعرض عليهم البرنامج الذي قمت بكتابته. سيسرهم ذلك كثيراً وسيخبرونك بكم أنك عبقرى. فالأطفال تحب ذلك.

بالطبع، ان البرنامج ليس ممتازاً، بالشكل الذي يدهش الأطفال عندما يكتشفون آلية عمله. خاصة اذا قام أحدهم بإدخال القيمة ١٠ كقيمة ابتدائية و القيمة ١ كقيمة نهائية. لماذا لا يطبع البرنامج أي شيء؟ ما الذي يمكنك القيام به للتغلب على هذا الخطأ؟ هل من الممكن أن يكون الحل هو استخدام جملة IF أخرى؟





## الفصل الرابع

### حلقات الت While من النوع

## PHP While Loops

يمكنك استخدام حلقات من النوع **while** بدلاً من استخدام حلقات من النوع **For**. البناء التركيبي لحلقة **while** يكون أبسط من حلقة **For**، لأنك فقط تقوم بتقييم شرط واحد فقط. والحلقة تدور وتدور الى أن يتحقق الشرط. عندما لا يتحقق الشرط، فان البرنامج يخرج من حلقة **while**. وهذه هي الصيغة لحلقة **while**:

```
while (condition) {

statement

}
```

وإليك هذا الكود، حاول تنفيذه، كل ما يؤديه الكود هو زيادة المتغير الذي يدعى **\$counter**:

```
$counter = ١ ;

while ($counter < ١١) {

print (" counter = " . $counter .
"
");
$counter++;

}
```

الشرط الذي قمت باختباره هو **\$counter < ١١**. كل مرة تدور فيها حلقة **while**، فان هذا الشرط يتم اختباره. طالما أن العداد أقل من ١١ فان الشرط يكون متحقق. وعندما يصبح **\$counter** أكبر من ١١ فان الشرط في هذه الحالة يكون غير متحقق. وستتوقف حلقة **while** عن الدوران عندما يكون الشرط غير متحقق.

عندما تستخدم حلقة **while**، ضع في اعتبارك أن تقوم بإنشاء حلقة غير منتهية. يمكنك استخدام هذا النوع من الحلقات عندما لا تكون لديك القيمة التي يصبح بها الشرط متحققاً. يمكنك عمل حلقة غير منتهية باستخدام حلقة **while** كما سبق. كل ما عليك فعله هو التعليق على السطر الذي يشتمل على زيادة قيمة متغير العداد **\$counter**

كما يلي:

```
$counter = ١;

while ($counter < ١١) {

 print (" counter = " . $counter .
 "
");
 //$counter++;

}
```

لاحظ أن الشرطتين المائلتين الأماميتين قبل **\$counter++**، فإن هذا السطر سيتم تجاهله. لأن الحلقة تدور وتدور طالما أن قيمة العداد أقل من ١١، لن تنتهي الحلقة أبداً – فالمتغير \$counter ستظل قيمته دائما ١.

هذه حلقة while والتي      دول ضرب ٢. حاول أن تقوم بتنفيذ السكريبت.

```
$start = ١;
$times = ٢;
$answer = ٠;

while ($start < ١١) {

 $answer = $start * $times;
 print ($start . " times " . $times . " =
 " . $answer . "
");
 $start++;

}
```

حلقة while تقوم بحساب جدول ضرب ٢. هل يمكنك أن تعرف كيف؟ تأكد من أنك بالفعل تفهم الكود السابق. وإذا لم تكن كذلك عليك العودة لقراءة هذا الفصل مرة أخرى.

في الفصل التالي، سأتناول نبذة مختصرة عن حلقات **Do ... While**.

## الفصل الخامس

حلقات Do ... While في PHP

PHP Do ... While loops

## PHP Do ... While loops

هذا النوع من الحلقات يكون شبيهاً الى حد ما بحلقة While، فيمعدا أن الشرط يأتي في النهاية:

**do**

**statement**

**while (condition)**

الاختلاف هو أن الجملة يتم تنفيذها على الأقل مرة واحدة. في حلقة while العادية، فإن الشرط يتم رؤيته قبل الجملة التي سيتم تنفيذها.

لا تقلق كثيراً بشأن حلقات do ... while. قم بالتركيز فقط على حلقات For و While. لكن يوجد نوع آخر من الحلقات التي ستتعامل معها وستكون في متناول يدك وهي For Each loop. وإليك أولاً، نبذة مختصرة عن جملة التوقف Break في الفصل القادم.

## الفصل السادس

### جملة التوقف break في PHP

### The PHP break statement

تحتاج في بعض الأوقات الى اجبار الحلقة على التوقف دون أن تكمل جميع الأوامر التي عليها تنفيذها، أو أنك تريد أن تتوقف الحلقة بسبب خطأ قام به المستخدم. وفي أي حال، يمكنك استخدام جملة **Break**. ولحسن الحظ فإنها لا تتطلب فقط سوى كتابة الكلمة **break**، الكود التالي ليس مفيداً الى حد ما ولكنه يحدد لك وظيفة جملة **break**:

```
$TeacherInterrupts = true;
$counter = ١;

while ($counter < ١١) {

 print(" counter = " + $counter +
 "
");

 if ($TeacherInterrupts == true) {

 break;

 }

 $counter++;

}
```

حاول أن تقوم بتنفيذ الكود السابق لترى ما الذي يحدث.

بهذا تكون قد تكونت لديك خبرة بسيطة عن التعامل مع الحلقات. نكتفي بها الآن.



## المراجع

أولاً: المراجع العربية:

الموسى، عبد الله عبد العزيز: ٢٠٠٨، استخدام الحاسب الآلي في التعليم.  
ط٤. الرياض.

ثانياً: المراجع الأجنبية:

**Tucker,A, Noonan,R(٢٠١٨): Programming languages:Principles and Paradigms.**

**R.Groff,Paul N.SQL:The complete Reference.**

**Peter, Coronel,C(٢٠٢٤): base Systems: Design,Implementation, and management..**

**Duckett,J (٢٠٢٤): HTML and CSS: Design and Build websites.**

ثالثاً: المواقع

[https://developer.mozilla.org/ar/docs/Web/JavaScript/Introduction to O  
\(Object-oriented programming\) البرمجة الكائنية](https://developer.mozilla.org/ar/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript#البرمجة_الكائنية_(Object-oriented_programming))

[http://refaat-mm.blogspot.com.eg/٢٠١١/٠٤/blog-  
post.html?showComment=١٣٠٤٥٢٧٢٠٨١١٩#c٥٠٠٢١٩٧٣٦٨٦٧١٩٠٩٠٥٤](http://refaat-mm.blogspot.com.eg/٢٠١١/٠٤/blog-post.html?showComment=١٣٠٤٥٢٧٢٠٨١١٩#c٥٠٠٢١٩٧٣٦٨٦٧١٩٠٩٠٥٤)

[/http://sho3a3.net/object-oriented-programming](http://sho3a3.net/object-oriented-programming)

<http://www.homeandlearn.co.uk/php/php\p\p.html>

/http://php.net

[/http://www.3alampro.com/series/php/php-basics/php-introduction](http://www.3alampro.com/series/php/php-basics/php-introduction)

[/http://www.swalif.net/softs/swalif%20/softs%20.%20](http://www.swalif.net/softs/swalif%20/softs%20.%20)

<https://ar.wikipedia.org/wiki/%D8%A7%D8%B1%D9%A0%D8%AC%D9%A9%D9%A3%D8%AV%D8%A6%D9%A7%D9%AA%D8%A9%D8%AV%D9%A4%D8%AA%D9%A7%D8%AC%D9%A7>

<https://eloquentjavascript.net>