



آموزشی کارپیدی

PHP

مؤلف: نیلوفر کاشفی

# آموزش کاربردی PHP

ساده و روان

تهیه و تألیف: نیلوفر کاشفی

آخرین بروزرسانی ۲۰ خرداد ۱۴۰۳

## فهرست مطالب

۱.....	مقدمه.....
۲.....	هدف این کتاب.....
۳.....	ساختار کتاب.....
۳.....	پیش‌نیازها.....
۳.....	نصب و راه‌اندازی PHP.....
۷.....	فصل اول - قواعد PHP.....
۸.....	حساسیت به سایز کاراکتر در PHP.....
۸.....	کامنت‌ها در PHP.....
۹.....	فصل دوم - متغیرها.....
۹.....	قوانین متغیرهای PHP.....
۱۱.....	محدوده دسترسی متغیرها.....
۱۳.....	فرامین چاپ.....
۱۶.....	فصل سوم - نوع داده‌ها.....
۱۷.....	تبدیل نوع داده‌ها.....
۱۸.....	ثوابت.....
۱۹.....	عملگرها.....
۲۳.....	تمرین بخش عملگرها.....
۲۴.....	فصل چهارم - دستورات کنترل برنامه.....

حلقه ها.....	۲۶
دستور break و continue.....	۲۷
تمارین بخش دستورات کنترل برنامه.....	۲۸
فصل پنجم - آرایه ها.....	۳۱
توابع در آرایه ها.....	۳۵
تمارین بخش آرایه.....	۳۷
فصل ششم - رشته ها.....	۳۹
توابع در رشته.....	۳۹
تمارین بخش رشته.....	۴۰
فصل هفتم - توابع.....	۴۲
تمارین بخش تابع.....	۴۴
فصل هشتم - متغیرهای سراسری.....	۵۱
فصل نهم - اشکال زدائی و مدیریت حالات استثنائی.....	۵۵
فصل دهم - دستور include و require.....	۵۷
دستور include_once و require_once.....	۵۸
فصل یازدهم - فرم ها.....	۵۹
اعتبارسنجی فرم ها.....	۶۳
فیلدهای اجباری در فرم ها.....	۶۶
تمارین بخش فرم.....	۷۰
فصل دوازدهم - فایل ها.....	۷۳
بارگذاری فایل.....	۷۸
فصل سیزدهم - کوکی ها.....	۸۲
فصل چهاردهم - نشست ها.....	۸۵

## فهرست مطالب ■ ۵

۸۷	..... پایگاه داده MySQL در PHP
۸۸	..... اتصال به MySQL
۹۸	..... فصل پانزدهم - نکات امنیتی مهم در برنامه‌نویسی PHP
۹۸	..... اعتبارسنجی و تصدیق داده‌های ورودی
۹۹	..... جلوگیری از حملات SQL Injection
۹۹	..... مدیریت صحیح نشست‌ها
۱۰۰	..... مدیریت صحیح خطاها
۱۰۰	..... استفاده از HTTPS
۱۰۱	..... محدود کردن دسترسی به فایل‌ها و پوشه‌ها
۱۰۱	..... استفاده از تابع‌های امن برای فایل‌ها
۱۰۱	..... به‌روزرسانی منظم نرم‌افزارها و کتابخانه‌ها

## مقدمه

زبان برنامه نویسی PHP یکی از محبوب ترین زبان های برنامه نویسی وب است که به دلیل سادگی، انعطاف پذیری و قابلیت های گسترده اش، توسط میلیون ها توسعه دهنده در سراسر جهان استفاده می شود. PHP در سال ۱۹۹۵ توسط Rasmus Lerdorf توسعه یافت. در ابتدا، PHP یک مجموعه ی کوچک از اسکریپت ها بود که برای ردیابی بازدیدکنندگان وبسایت شخصی Rasmus طراحی شده بود. با گذشت زمان و با مشارکت جامعه ی برنامه نویسان، PHP به یک زبان برنامه نویسی کامل و توانمند تبدیل شد که امروزه نسخه های جدیدتر و پیشرفته تری از آن در دسترس است.

PHP یک زبان برنامه نویسی سمت سرور است، به این معنا که صفحات PHP ابتدا توسط سرور پردازش می شوند و سپس خروجی به سمت مرورگر کاربر ارسال می گردد. به این زبان ویژگی HTML Embedded تخصیص می یابد به این معنا که با قرار گیری بین کدهای HTML، صفحات ایستای پیاده سازی شده با HTML را به صفحاتی پویا تبدیل می کنند. این پویاسازی سبب افزایش کارایی وبسایت، اعمال آسان تر تغییر در آن و امکان تعامل با کاربر می شود. وبسایت طراحی شده با PHP قابلیت آن را دارد که محتوای آن توسط افرادی که دانش برنامه نویسی ندارند به روزرسانی شود. لذا با طراحی یک وبسایت پویا با PHP و تحویل آن به کارفرما، برنامه نویس درگیر حداقل پشتیبانی خواهد بود.

### دلیل محبوبیت PHP

دلایل متعددی وجود دارد که PHP را به یکی از انتخاب های اصلی برنامه نویسان وب تبدیل کرده است:

- **یادگیری آسان PHP:** دارای قواعد ساده و قابل فهمی است که یادگیری آن را برای مبتدیان آسان می‌کند.
- **منبع باز و رایگان PHP:** یک نرم‌افزار منبع باز است و به صورت رایگان در دسترس است. این به معنای عدم هزینه‌ی اولیه برای شروع کار با PHP است.
- **انعطاف‌پذیری و سازگاری PHP:** بر روی اکثر سیستم‌عامل‌ها و سرورهای وب قابل اجراست و با بسیاری از پایگاه‌داده‌ها و سرویس‌های وب سازگار است.
- **پشتیبانی گسترده:** مستندات جامع، منابع آموزشی برخط<sup>۱</sup>، انجمن‌های پشتیبانی و کتابخانه‌های متنوعی برای PHP وجود دارد که به توسعه‌دهندگان در حل مشکلات و ارتقاء دانش‌شان کمک می‌کند.
- **کاربرد گسترده PHP:** در پروژه‌های کوچک و بزرگ، از وب‌سایت‌های شخصی تا سیستم‌های پیچیده‌ی مدیریت محتوا (CMS) مانند وردپرس، استفاده می‌شود.

## هدف این کتاب

هدف این کتاب، ارائه‌ی یک راهنمای جامع و عملی برای یادگیری زبان PHP به زبانی بسیار ساده است. این کتاب شامل مثال‌های عملی و تمرین‌های کاربردی است که به شما کمک می‌کند تا مفاهیم را به خوبی درک کرده و در پروژه‌های واقعی به کار بگیرید. در این کتاب سعی بر این است مطالب آموزشی به بیانی کاملاً روان به خوانندگان محترم ارائه گردد. ساختار و ترتیب قرارگیری مطالب از استاندارد روش‌های آموزش زبان‌های برنامه‌نویسی پیروی می‌کند و در انتهای هر فصل نیز تمرینی ارائه شده است. لازم به ذکر است که همانند روال فراگیری هر زبان برنامه‌نویسی، یادگیری زبان PHP مستلزم تمرین و تکرار است، لذا توصیه می‌شود که به منظور بهبود روند یادگیری، ضمن پیشروی گام به گام با کتاب به تمرین هر فصل پس از اتمام فصل بپردازید. امید است که کتاب حاضر مرجع مفیدی برای درخشیدن شما در عرصه طراحی وبسایت باشد.

---

<sup>1</sup> Online

## ساختار کتاب

این کتاب به چندین فصل تقسیم شده است که هر فصل به یکی از مباحث مهم PHP می‌پردازد. در هر فصل، ابتدا مفاهیم اساسی توضیح داده می‌شوند و سپس با مثال‌ها و تمرین‌های عملی، شما را در درک عمیق‌تر آن‌ها یاری می‌دهیم. در پایان هر فصل، تمرین‌هایی برای تمرین بیشتر و بهبود مهارت‌های شما قرار داده شده است.

## پیش‌نیازها

برای بهره‌گیری کامل از این کتاب، بهتر است با مبانی برنامه‌نویسی و اصول اولیه‌ی HTML آشنا باشید. اگر تازه وارد دنیای برنامه‌نویسی شده‌اید، نگران نباشید؛ ما سعی کرده‌ایم که تمام مفاهیم را به ساده‌ترین شکل ممکن توضیح دهیم تا همه بتوانند از آن بهره‌مند شوند.

## نصب و راه‌اندازی PHP

قبل از اینکه بتوانیم شروع به کدنویسی PHP کنیم، باید محیط توسعه‌ای را برای اجرای اسکریپت‌های PHP فراهم کنیم. اولین نکته این است که کدهای PHP را می‌توانید در هر محیطی حتی notepad++ پیاده‌سازی کنید. حال برای اجرای کدها به سروری نیاز داریم که از زبان برنامه‌نویسی PHP پشتیبانی کند و علاوه بر آن سامانه‌ای برای مدیریت پایگاه داده روی آن نصب باشد. برای این منظور لازم است بر روی سرور خود Apache (یا IIS)، PHP و MySQL (بسیاری از وب‌سایت‌ها از MySQL به عنوان پایگاه‌داده اصلی خود استفاده می‌کنند) را نصب نمایید.

همانطور که می‌دانید استفاده از سرور مستلزم هزینه است اما خبر خوب اینکه نیازی به پرداخت هزینه بابت سرور را ندارید زیرا برنامه‌های میزبانی وب با قابلیت پشتیبانی PHP و MySQL، نظیر XAMPP (Cross-platform Apache MySQL PHP Perl) وجود دارند که محیطی شبیه سرور را بر روی سیستم محلی ما شبیه‌سازی می‌کنند.

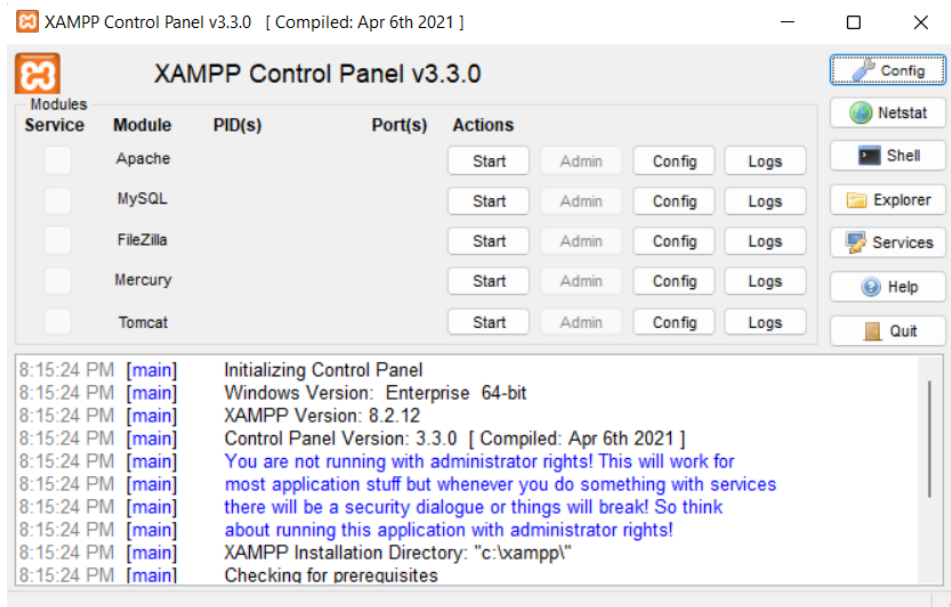


XAMPP برنامه ای چندسکوییست<sup>۱</sup> به این معنا که روی سیستم عامل های مختلف قابل پیاده سازی است. مزیت دیگر این آن است که نیاز به نصب هیچ برنامه اضافه تری ندارند و توسط آن به محیطی با پشتیبانی PHP و MySQL دست می یابیم. مراحل نصب و استفاده از XAMPP عبارتند از:

- **بارگیری:** نسخه مناسب XAMPP با سیستم خود را بارگیری کنید.
  - **نصب:** فایل بارگیری شده را به سادگی نصب کنید.
  - **اجرا:** پس از نصب XAMPP control panel را باز کنید.
  - **راه اندازی سرور و پایگاه داده:** برای شروع سرویس های Apache و MySQL، بر روی دکمه "Start" کلیک کنید. در صورتی که سرویس ها با موفقیت راه اندازی شوند، شما باید متن "Running" را در کنار Apache و MySQL ببینید.
  - **اجرای فایل برنامه:** برنامه های وب خود را که پسوند php دارند جهت اجرا در دایرکتوری httdocs که در مسیر نصب XAMPP قرار دارد، قرار دهید (بهتر است محتوای قبلی فولدر httdocs را حذف کنید). مرورگر وب خود باز کنید. در نوار آدرس، آدرس http://localhost (یا 127.0.0.1) را وارد کنید. در صفحه باز شده در مرورگر به طور پیش فرض صفحه ای که با نام index.php در فولدر httdocs اجرا می شود (بهتر است فایل برنامه خود را با نام index.php ذخیره نمایید).
  - **دسترسی به رابط پایگاه داده (phpMyAdmin):** برای این منظور، روی کلید "Admin" در XAMPP Control Panel در بخش MySQL کلیک کنید. در phpMyAdmin، می توانید پایگاه داده ها را ایجاد، مدیریت و سایر عملیات مربوط به پایگاه داده را انجام دهید.
- در شکل ۱ محیط برنامه XAMPP را مشاهده می کنید. از طریق کلید "start" راه اندازی سرویس ها و از طریق کلید "Admin" محیط آن ها در مرورگر باز خواهد شد.

---

<sup>1</sup> Cross-platform



شکل ١ - محیط برنامه XAMPP



# فصل اول – قواعد PHP

می توان گفت نقطه اصلی تمایز زبان های برنامه نویسی مختلف، قواعد آن ها می باشد. هر زبان برنامه نویسی قواعد مختص به خود را دارد که در گام اول یادگیری باید به آن پرداخته شود. یک اسکریپت PHP را می توان در هر جایی از سند قرار داد. یک اسکریپت PHP با `<?php` شروع می شود و با `>?` پایان می یابد:

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

پسوند فایل پیش فرض برای فایل های PHP، ".php" است. یک فایل PHP معمولاً حاوی تگ های HTML و بخشی کد برنامه نویسی PHP است. مثال:

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

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

## حساسیت به ساینز کاراکتر در PHP

در PHP، کلمات کلیدی (به عنوان مثال if, else, while, echo و غیره)، کلاس ها، توابع و توابع تعریف شده توسط کاربر به حروف بزرگ و کوچک حساس نیستند. در مثال زیر، هر سه عبارت echo (دستور چاپ در PHP) زیر برابر هستند:

```
<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
```

بر خلاف کلمات کلیدی، در PHP نام متغیرها به حروف بزرگ و کوچک حساس هستند. در مثال زیر فقط عبارت اول مقدار متغیر \$color را نمایش می دهد! دلیلش این است که \$color، \$COLOR و \$coLoR به عنوان سه متغیر مختلف در نظر گرفته می شوند:

```
<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLoR . "<br>";
?>
```

## کامنت ها در PHP

در PHP از // یا # برای ایجاد یک کامنت تک خطی یا /\* و \*/ برای ایجاد یک بلوک کامنت بزرگ استفاده می کنیم.

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

# This is also a single-line comment
/*
This is a multiple-lines comment block
that spans over multiple lines
*/
?>
```

## فصل دوم – متغیرها

متغیرها "ظروف" برای ذخیره اطلاعات هستند. در PHP، یک متغیر با علامت \$ شروع می‌شود و به دنبال آن نام متغیر می‌آید:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

**نکته:** هنگامی که یک مقدار متنی (رشته<sup>۱</sup>) را به یک متغیر اختصاص می‌دهیم، لازم است آن مقدار را داخل " " قرار دهیم، همانند "Hello world!".

**نکته ۲:** برخلاف سایر زبان‌های برنامه‌نویسی، PHP هیچ فرمانی برای اعلان متغیر ندارد. به این معنا که نیازی نیست نوع متغیر را در هنگام تعریف آن مشخص کنیم و با مقداری که آن، نوع متغیر بر اساس مقدار داده شده به طور خودکار مشخص می‌شود. به عنوان مثال با اجرا دستور `$txt = "Hello world!";` برای متغیر `$txt` نوع رشته در نظر گرفته می‌شود. با انواع متغیرها نظیر عدد صحیح<sup>۲</sup>، عدد اعشاری<sup>۳</sup>، رشته و ... در فصول بعد آشنا خواهید شد.

### قوانین متغیرهای PHP

نام گذاری متغیرها مستلزم رعایت قوانینی است که در صورت عدم رعایت آن‌ها برنامه به خطا می‌خورد. این قوانین عبارتند از:

---

<sup>۱</sup> string  
<sup>۲</sup> integer  
<sup>۳</sup> float

- یک متغیر با علامت \$ شروع می شود و به دنبال آن نام متغیر می آید
  - نام متغیر باید با یک حرف یا کاراکتر زیرخط شروع شود
  - نام متغیر نمی تواند با عدد شروع شود
  - نام متغیر فقط می تواند حاوی نویسه های عددی و زیرخط باشد (A-Z، ۰-۹، و \_)
  - نام متغیرها به حروف بزرگ و کوچک حساس هستند (age\$ و AGE\$ دو متغیر متفاوت هستند)
- خروجی:** دستور echo اغلب برای نمایش خروجی داده ها استفاده می شود. مثال زیر نحوه استفاده از این دستور و خروجی برنامه را نشان می دهد:

```
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

I love W3Schools.com!

مثال زیر همان خروجی مثال بالا را تولید می کند:

```
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
?>
```

I love W3Schools.com!

مثال زیر مجموع دو متغیر را به دست می دهد:

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

9

در این مثال، توجه کنید همانطور که گفته شد لزومی ندارد نوع داده را مشخص کنیم. PHP به طور خودکار یک نوع داده را بسته به مقدار آن متغیر به آن اختصاص می دهد. از آنجایی که در PHP انواع داده ها به معنای دقیق تنظیم نمی شوند، می توان اعمالی از قبیل اضافه کردن یک رشته به یک عدد صحیح را بدون ایجاد خطا انجام داد.

## محدوده دسترسی متغیرها

محدوده یک متغیر بخشی از اسکریپت است که در آن متغیر می تواند ارجاع یا استفاده شود. PHP دارای سه محدوده متغیر مختلف است: محلی<sup>۱</sup>، سراسری<sup>۲</sup>، ایستا<sup>۳</sup>

**محدوده سراسری و محلی:** متغیری که خارج از یک تابع اعلام شده است دارای یک محدوده سراسری است و فقط خارج از یک تابع قابل دسترسی است:

```
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
```

Variable x inside function is:

Variable x outside function is: 5

همانطور که در خروجی برنامه مشاهده می کنید متغیر x در داخل تابع قابل دسترسی نبود. یک متغیر اعلام شده در یک تابع دارای یک محدوده محلی است و فقط در آن تابع قابل دسترسی است. لازم به ذکر است که می توان متغیرهای محلی با نام یکسان در توابع مختلف داشت.

<sup>1</sup> local

<sup>2</sup> global

<sup>3</sup> static



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

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

Variable x inside function is: 5

Variable x outside function is:

**global:** کلمه کلیدی global برای دسترسی به یک متغیر سراسری از داخل یک تابع استفاده می شود. برای انجام این کار، از کلمه کلیدی global قبل از متغیرها (داخل تابع) استفاده کنید:

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

function myTest() {
    global $x, $y;
    $y = $x + $y;
}
myTest();
echo $y; // outputs 15
?>
```

خروجی برنامه بالا که حاصل جمع دو عدد ۵ و ۱۰ می باشد برابر ۱۵ است و با توجه به اینکه از کلمه کلیدی global استفاده شده است متغیر x و y با وجود اینکه خارج از تابع مقادیری شده اند، در داخل تابع نیز قابل دسترسی هستند و عملیات جمع بدون خطا روی آن ها صورت می گیرد.

ایستا: به طور معمول، زمانی که اجرای یک تابع کامل می شود، همه متغیرهای آن حذف می شوند. با این حال، گاهی اوقات می خواهیم یک متغیر محلی حذف نشود. برای این کار، زمانی که متغیر را برای اولین بار تعریف می کنیم، از کلمه کلیدی `static` استفاده می کنیم. توجه داشته باشید که متغیر همچنان محلی برای تابع است:

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

0  
1  
2

سپس، هر بار که تابع فراخوانی شود، آن متغیر همچنان اطلاعاتی را که از آخرین باری که تابع فراخوانی شده بود، خواهد داشت.

## فرامین چاپ

در PHP، دو راه اساسی برای دریافت خروجی وجود دارد: `echo` و `print`. هر دو برای خروجی داده ها به صفحه نمایش استفاده می شوند و تفاوت کمی دارند: `echo` مقدار بازگشتی ندارد در حالی که `print` مقدار بازگشتی ۱ دارد بنابراین می توان از آن در عبارات استفاده کرد. `echo` می تواند چندین پارامتر داشته باشد (اگرچه چنین استفاده ای نادر است) در حالی که `print` می تواند یک آرگومان داشته باشد. در نهایت می توان گفت که `echo` تا حدی سریعتر از `print` است.

**echo:** دستور `echo` را می توان با یا بدون پرانتز استفاده کرد: `echo` یا `echo()`. مثال زیر نحوه دریافت خروجی را با دستور `echo` نشان می دهد (توجه داشته باشید که متن می تواند دارای نشانه گذاری HTML باشد):

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

## PHP is Fun!

Hello world!  
I'm about to learn PHP!  
This string was made with multiple parameters.

مثال زیر نحوه خروجی متن و متغیرها با دستور echo را نشان می دهد:

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
```

## Learn PHP

Study PHP at W3Schools.com  
9

**Print:** دستور print را می توان با یا بدون پرانتز استفاده کرد: print یا print(). مثال زیر نحوه دریافت خروجی را با دستور print نشان میدهد (توجه داشته باشید که متن می تواند دارای نشانه گذاری HTML باشد):

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

## PHP is Fun!

Hello world!  
I'm about to learn PHP!

مثال صفحه بعد نحوه خروجی متن و متغیرها با دستور print را نشان می دهد:

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x + $y;
?>
```

# Learn PHP

Study PHP at [W3Schools.com](https://www.w3schools.com)

9

## فصل سوم - نوع داده ها

متغیرها می‌توانند داده‌های مختلف را ذخیره کنند و انواع داده‌های مختلف می‌توانند برای کاربردهای مختلف له کار برده شوند. برخی از انواع داده ها عبارتند از:

رشته: رشته، دنباله ای از کاراکترها است، مانند «سلام دنیا!». یک رشته می‌تواند هر متنی در داخل “” باشد. می‌توانید از “” یا “” استفاده کنید.

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

Hello world!  
Hello world!

**عدد صحیح:** یک نوع داده عدد صحیح یک عدد غیر اعشاری بین ۲,۱۴۷,۴۸۳,۶۴۸- و ۲,۱۴۷,۴۸۳,۶۴۷ است. اعداد صحیح قوانینی دارند که عبارتند از:

- یک عدد صحیح باید حداقل یک رقم داشته باشد
- یک عدد صحیح نباید دارای اعشار باشد
- یک عدد صحیح می‌تواند مثبت یا منفی باشد

اعداد صحیح را می‌توان به صورت اعشاری (مبنای ۱۰)، هگزا دسیمال (مبنای ۱۶)، هشتی (مبنای ۸)، یا باینری (پایه ۲) مشخص کرد. در مثال زیر \$x یک عدد صحیح است. تابع PHP var\_dump() نوع و مقدار داده را برمی‌گرداند:

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

int(5985)

اعداد ممیز شناور: شناور عددی است با اعشار یا عددی به صورت نمایی.

دودویی: یک دودویی یا Boolean دو حالت ممکن را نشان می دهد: TRUE یا FALSE

آرایه: یک آرایه چندین مقدار را در یک متغیر ذخیره می کند.

NULL: یک نوع داده خاص است که می تواند تنها یک مقدار داشته باشد: NULL

متغیری از نوع داده NULL متغیری است که هیچ مقداری به آن اختصاص داده نشده

است. اگر متغیری بدون مقدار ایجاد شود، به طور خودکار مقدار NULL به آن اختصاص

داده می شود. همچنین می توان متغیرها را با تنظیم مقدار NULL خالی کرد:

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

NULL

## تبدیل نوع داده ها

در مواقعی پیش می آید که شما نیاز دارید یک مقدار عددی را به نوع داده دیگری تبدیل

کنید که در PHP برای این منظور توابعی از پیش تعریف شده وجود دارند. به عنوان مثال،

تابع (int)، (integer) یا intval() اغلب برای تبدیل یک مقدار به یک عدد صحیح استفاده

می شود. برای سایر نوع داده ها نیز توابعی ارائه شده است.

```
<?php
// Cast float to int
$x = 23465.768;
$int_cast = (int)$x;
var_dump($int_cast);
echo "<br>";
// Cast string to int
$x = "23465.768";
$int_cast = (int)$x;
var_dump($int_cast);
?>
```

int(23465)  
int(23465)

علاوه بر این PHP دارای تابع is\_int() برای بررسی int بودن نوع متغیر است:

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

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

```
bool(true)
bool(false)
```

همچنین در PHP، تابع `is_numeric()` می تواند برای تعیین عددی بودن یک متغیر استفاده شود. اگر متغیر یک عدد یا یک رشته عددی باشد، تابع `true` است، در غیر این صورت `false`:

```
<?php
$x = 5985;
var_dump(is_numeric($x));
$x = "5985";
var_dump(is_numeric($x));
$x = "59.85" + 100;
var_dump(is_numeric($x));
$x = "Hello";
var_dump(is_numeric($x));
?>
```

```
bool(true)
bool(true)
bool(true)
bool(false)
```

## ثوابت

ثابت ها مانند متغیرها هستند، با این تفاوت که پس از تعریف، نمی توان آنها را تغییر داد. یک نام ثابت معتبر با یک حرف یا خط زیر شروع می شود (بدون علامت \$ قبل از نام ثابت). برخلاف متغیرها، ثابت ها بطور خودکار سراسری هستند.

```
<?php
define("GREETING", "Welcome to W3Schools.com!");

function myTest() {
    echo GREETING;
}
myTest();
?>
```

```
Welcome to W3Schools.com!
```

## عملگرها

عملگرها برای انجام عملیات روی متغیرها و مقادیر استفاده می شوند. PHP عملگرها را به گروه های زیر تقسیم می کند:

- ✓ عملگرهای حسابی (Arithmetic operators)
- ✓ اپراتورهای واگذاری (Assignment operators)
- ✓ عملگرهای مقایسه (Comparison operators)
- ✓ عملگرهای افزایش/کاهش (Increment/Decrement operators)
- ✓ عملگرهای منطقی (Logical operators)
- ✓ عملگرهای رشته ای (String operators)
- ✓ عملگرهای آرایه (Array operators)
- ✓ اپراتورهای انتساب مشروط (Conditional assignment operators)

**PHP Arithmetic Operators:** عملگرهای محاسباتی PHP با مقادیر عددی برای انجام عملیات حسابی رایج مانند جمع، تفریق، ضرب و غیره استفاده می شوند.

نام	عملگر
Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus	%
Exponentiation	**

**PHP Assignment Operators:** عملگرهای انتساب PHP با مقادیر عددی برای نوشتن یک مقدار به یک متغیر استفاده می شوند. عملگر اصلی انتساب در PHP، "=" است. این بدان معنی است که مقدار سمت چپ را در مقدار سمت راست قرار می دهد.



عملگر	معادل است با...
$x = y$	$x = y$
$x += y$	$x = x + y$
$x -= y$	$x = x - y$
$x *= y$	$x = x * y$
$x /= y$	$x = x / y$
$x \% = y$	$x = x \% y$

**PHP Comparison Operators:** عملگرهای مقایسه PHP برای مقایسه دو مقدار (عدد

یا رشته) استفاده می شوند:

عملگر	نام
$==$	Equal
$===$	Identical (same type)
$!=$	Not equal
$<>$	Not equal
$!==$	Not identical
$>$	Greater than
$<$	Less than
$>=$	Greater than or equal to
$<=$	Less than or equal to
$<=>$	Spaceship

عملگر  $<=>$  که در PHP 7 ارائه شده است به اینصورت عمل می کند که در صورت تساوی مقدار صفر را برمی گرداند، در غیر اینصورت چنانچه متغیر سمت چپ بزرگتر از متغیر سمت راست باشد مقدار ۱ را برمی گرداند و چنانچه متغیر سمت چپ کوچکتر از متغیر سمت راست باشد مقدار -۱ را برمی گرداند.

**PHP Increment / Decrement Operators:** عملگرهای افزایشی PHP برای افزایش مقدار یک متغیر استفاده می شوند و عملگرهای کاهش PHP برای کاهش مقدار یک متغیر استفاده می شوند.

نام	عملگر
Pre-increment	<code>++\$x</code>
Post-increment	<code>\$x++</code>
Pre-decrement	<code>--\$x</code>
Post-decrement	<code>\$x--</code>

**PHP Logical Operators:** عملگرهای منطقی PHP برای ترکیب عبارات شرطی استفاده می شوند.

نام	عملگر
And	<code>and</code>
Or	<code>or</code>
Xor	<code>xor</code>
And	<code>&amp;&amp;</code>
Or	<code>  </code>
Not	<code>!</code>

**PHP String Operators:** PHP دو عملگر دارد که مخصوص رشته‌ها طراحی شده‌اند.

مثال	نام	عملگر
<code>\$txt1 . \$txt2</code>	Concatenation	<code>.</code>
<code>\$txt1 .= \$txt2</code>	Concatenation assignment	<code>.=</code>

**PHP Array Operators:** عملگرهای آرایه PHP برای مقایسه آرایه ها استفاده می - شوند.

نام	عملگر
Union	+
Equality	==
Identity	===
Inequality	!=
Inequality	<>
Non-identity	!==

### PHP Conditional Assignment Operators: عملگرهای انتساب شرطی PHP برای

تعیین مقدار بسته به شرایط استفاده می شوند:

توضیح	نام	عملگر
$\$x = \text{expr1} ? \text{expr2} : \text{expr3}$ مقدار $x$ ، $\text{expr2}$ است اگر $\text{expr1} = \text{TRUE}$ مقدار $x$ ، $\text{expr3}$ است اگر $\text{expr1} = \text{FALSE}$	Ternary	?:
$\$x = \text{expr1} ?? \text{expr2}$ مقدار $x$ ، $\text{expr1}$ است اگر $\text{expr1}$ Null نباشد یا موجود باشد مقدار $x$ ، $\text{expr2}$ است اگر $\text{expr1}$ Null باشد یا موجود باشد	Null coalescing	??

## تمارین بخش عملگرها

۱. برنامه‌ای بنویسید که با در نظر گرفتن طول و عرض برای مستطیلی، محیط و مساحت آن را چاپ کند.

```
$x=5;
$y = 10;

echo "area: ",2*($x*$y),"<br>";
echo "perimeter: ",($x*$y);
```

۲. برنامه‌ای بنویسید که یک زمان مشخص بر اساس ساعت، دقیقه و ثانیه را در نظر بگیرد و مشخص کند چند ثانیه از شروع روز گذشته است.

```
$second=20;
$minute=50;
$hour=12;
echo ($minute *60)+($hour *3600)+$ second;
```

۳. برنامه‌ای بنویسید که تعدادی عدد را در نظر بگیرد و سپس قدرمطلق و ماکزیمم آن‌ها را با توابع ریاضی محاسبه کند.

```
$x=5;
$y =10;
$z=-9;
echo "abs is: ",abs($z);
echo "max is: ",max($x , $y , $z);
```

## فصل چهارم - دستورات کنترل برنامه

دستورات شرطی برای انجام اقدامات مختلف بر اساس شرایط مختلف استفاده می شود. اغلب هنگام نوشتن کد، می خواهید اقدامات مختلفی را برای شرایط مختلف انجام دهید. برای این کار می توانید از دستورات شرطی در کد خود استفاده کنید. در PHP دستورات شرطی زیر را داریم:

**دستور if:** اگر یک شرط درست باشد، کدی را اجرا می کند

**دستور if...else:** در صورت صحیح بودن یک شرط، کدی را اجرا می کند و اگر آن شرط نادرست باشد، کد دیگری را

**دستور if...elseif...else:** کدهای مختلف را برای بیش از دو شرط اجرا می کند

**عبارت switch:** یکی از بسیاری از بلوک های کد را برای اجرا انتخاب می کند

در ادامه به بررسی هر یک از ساختارهای کنترلی می پردازیم. مثال اول مربوط به دستور `if` است و با توجه به اینکه دستور داخل شرط `if` اجرا شده و است و "Have a good day" چاپ می شود می توان دریافت که شرط `("20" < $t)` برقرار است. مثال دوم نیز مربوط به شرط `if` است با این تفاوت که در صورت عدم برقراری شرط نیز تصمیمی اتخاذ گردیده به این معنی که اگر `("20" < $t)` باشد عبارت "Have a good day" و در غیراینصورت عبارت "Have a good night" (دستور داخل `else`) چاپ می شود. در حالت سوم در شرایطی که قصد بررسی بیش از دو شرط را داشته باشیم به کار می رود و به عبارتی تعداد حالات بیشتری را بررسی می کند. مشابه این کاربرد برای مثال آخر می باشد با این تفاوت که برای `switch-case` عبارت داخل شرط باید برابر با یک مقدار مشخص باشد.

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

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

Have a good day!

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

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

Have a good day!

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

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

The hour (of the server) is 10, and will give the following message:  
Have a good day!

```
<?php
$favcolor = "red";
```

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

Your favorite color is red!

## حلقه ها

اغلب هنگام نوشتن کد، می خواهید همان بلوک کد بارها و بارها به تعداد معینی اجرا شود. بنابراین، به جای اضافه کردن چندین خط کد تقریباً مساوی در یک برنامه، می توانیم از حلقه ها استفاده کنیم. حلقه ها برای اجرای دوباره همان بلوک کد استفاده می شوند، تا زمانی که یک شرط خاص درست باشد. در PHP انواع حلقه های زیر را داریم:

**while:** تا زمانی که شرط مشخص شده درست باشد حلقه ادامه دارد. در مثال زیر شرط توقف  $5 \leq x$  است، مقدار اولیه  $x$  برابر ۱ در نظر گرفته شده است و گام حلقه نیز ۱ است زیرا دستور  $x++$  در داخل بدنه حلقه **while** آورده شده است. نکته مهم این است که اگر گام در حلقه **while** ذکر نشود شرط توقف برقرار نمی شود و برنامه بی نهایت بار اجرا می شود. بنابراین ذکر گام در این حلقه بسیار ضروری است.

**do...while:** ابتدا حلقه اجرا می شود سپس شرط توقف آن چک می شود. بنابراین این حلقه حتماً حداقل یکبار اجرا خواهد شد حتی اگر شرط توقف هیچ گاه برقرار نباشد. حلقه **do while** پس از یکبار اجرا، تا زمانی که شرط مشخص شده درست باشد تکرار می کند. در مثال زیر تفاوت این حلقه با حلقه **while** قابل مشاهده است و با وجود اینکه مقدار  $x$  از ۵ بیشتر است یکبار حلقه اجرا می شود.

```
<?php
$x = 1;

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

The number is: 1  
The number is: 2  
The number is: 3  
The number is: 4  
The number is: 5

```
<?php
$x = 6;

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

The number is: 6

**for**: از طریق یک بلوک کد به تعداد مشخصی بارها تکرار می شود. برای مشخص کردن این تعداد مشخص سه کار لازم است انجام شود: نقطه شروع، اندازه گام در هر تکرار و نقطه پایان.

**Foreach**: از طریق یک بلوک کد برای هر عنصر در یک آرایه تکرار می شود. برخلاف حلقه for که لازم است تعداد گام ها مشخص باشد، در foreach در هر تکرار حلقه به ترتیب تمام عناصر آرایه خوانده می شود.

```
<?php
for ($x = 0; $x <= 5; $x++) {
    echo "The number is: $x <br>";
}
?>
```

The number is: 0  
The number is: 1  
The number is: 2  
The number is: 3  
The number is: 4  
The number is: 5

```
<?php
$colors = array("red", "green", "blue", "yellow");

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

red  
green  
blue  
yellow

## دستور break و continue

دستور break می تواند برای پرش از یک حلقه استفاده شود و ادامه آن را متوقف سازد. دستور continue یک تکرار (در حلقه) را در صورت وقوع یک شرط مشخص می شکند و مجددا اجرای حلقه را از تکرار بعدی ادامه می دهد.

```
<?php
$x = 0;
while($x < 6) {
    if ($x == 4) {
        $x++;
        continue;
    }
    echo "The number is: $x <br>";
    $x++;
}
```

The number is: 0  
The number is: 1  
The number is: 2  
The number is: 3  
The number is: 5



## تمارین بخش دستورات کنترل برنامه

۱. برنامه‌ای بنویسید که زوج و فرد بودن یک عدد را بررسی کند.

```
$x=5;

if($x%2==0)
echo "Even";
else
echo "Odd";
```

۲. برنامه‌ای بنویسید که تعدادی عدد را در نظر بگیرد و سپس ماکزیمم و میانگین آن‌ها را به صورت دستی (بدون استفاده از تابع) محاسبه کند.

```
$x1=5;
$x2=10;
$x3=9;

$max=$x1;
if($x2>$max)
    $max=$x2;
if($x3>$max)
    $max=$x3;
echo "max: $max";
echo "average: ",$(($x1+$x2+$x3)/3);
```

۳. برنامه‌ای بنویسید که سه ضلع یک مثلث را دریافت کند و مشخص کند مثلث متساوی الاضلاع، متساوی الساقین یا مختلف الاضلاع است.

```
$x=5;
$y = 10;
$z= 8;

If ($x==$y && $y==$z)
    echo "Equilateral triangle";
else if ($x==$y || $y==$z || $z==$x)
    echo "Isosceles triangle";
else echo "Scalene triangle";
```

۴. برنامه‌ای بنویسید که یک عدد صحیح را در نظر بگیرید. برای خورد کردن چنین مقدار پولی با سکه‌های یک تومانی، ۲ تومانی و ۱۰ تومانی به حداقل چه تعداد

سکه نیازمندیم. (مثلا اگر کاربر ۲۳ را وارد کرد باید در خروجی بنویسد دو سکه ۱۰ تومانی، یک سکه ۲ تومانی و یک سکه ۱ تومانی)

```
$x=5;

if($x>=10){
    $n=floor($x/10);
    $x%=10;
}
if($x>=2){
    $n+=floor($x/2);
    $x=$x%2;
}
echo "#coins: ", $n+$x;
```

برای تعیین تعداد دقیق سکه ها:

```
$ten=0; $two=0; $one=0;
$money=47;
if($money>=1)
{
    while($money>=10)
    {
        $money-=10;
        $ten++;
    }
    while($money>=2)
    {
        $money-=2;
        $two++;
    }
    if($money)
        $one++;
}
echo "#10: $ten, #2: $two, #1:$one";
```

۵. برنامه‌ای بنویسید که برای عدد فرضی  $n$ ، مجموع ۱ تا  $n$  را محاسبه کند.

```
$n=10;
$sum=0;
for($i=0;$i<=$n;$i++)
    $sum+=$i;
echo "sum is: $sum";
```

۶. برنامه‌ای بنویسید فاکتوریل  $n$  آن را محاسبه کند.

```
$n=5;
```

۳۰

```
$fac=1;
for($i=1;$i<=$n;$i++)
    $fac*=$i;
echo "fact is: $fac";
```

۷. برنامه‌ای بنویسید که یک عدد در نظر بگیرد و تعداد ارقامش، مجموع ارقامش، بزرگترین رقمش و معکوسش را چاپ کند.

```
$number=987;
$number_digits=0;
$sum_digits=0;
$inverse="";
$max=0;

while($number>=0){
    $last_digit=$number%10;
    $inverse = $ inverse . $last_digit;
    $sum_digits += $last_digit;
    if($last_digit > $max)
        $max=$last_digit;
    $number=floor($number/10);
    $number_digits++;
}
```

۸. برنامه بازی HOP را بنویسید که دو عدد  $m$  و  $n$  را از خروجی دریافت کند و در خروجی اعداد ۱ تا  $n$  چاپ می‌شود با این شرایط که به جای اعداد مضرب  $m$ ، کلمه HOP چاپ شود. برای مثال برای  $n=9$  و  $m=3$  باید خروجی زیر چاپ شود.

```
۱ ۲ HOP
۴ ۵ HOP
۷ ۸ HOP
$n=3;
$m=9;
for ($i=1;$i<=$m;$i++){
    if($i%$n==0)
        echo "HOP ";
    else
        echo $i," ";
}
```

## فصل پنجم – آرایه ها

یک آرایه چندین مقدار را در یک متغیر ذخیره می کند و به عبارتی می تواند حاوی لیستی از مقادیر عددی، رشته و یا هر دو باشند (آرایه های استاندارد در PHP از اعداد تشکیل می شوند). آرایه در واقع به صورت ظرفی عمل می کند، برای هر داده که در این ظرف قرار می گیرد یک کلید در نظر می گیریم تا در زمان نیاز به آن داده، با استفاده از کلید آن، دسترسی را مهیا سازیم. با استفاده از دستور `array()` در PHP میتوان آرایه ساخت. در PHP سه نوع آرایه وجود دارد:

آرایه های نمایه شده: آرایه هایی با اندیس عددی

آرایه های انجمنی: آرایه هایی با کلیدهای نامگذاری شده

آرایه های چند بعدی: آرایه هایی حاوی یک یا چند آرایه

آرایه های نمایه شده PHP: این نوع آرایه، آرایه هایی با اندیس عددی هستند. دو راه برای این آرایه ها وجود دارد:

۱. اندیس را می توان به طور خودکار اختصاص داد (اندیس همیشه از ۰ شروع می شود)، مانند زیر:

```
$cars = array("Volvo", "BMW", "Toyota");
```

۲. یا اندیس را می توان به صورت دستی اختصاص داد:

```
$cars[0] = "Volvo";  
$cars[1] = "BMW";  
$cars[2] = "Toyota";
```

مثال زیر یک آرایه نمایه شده به نام \$cars ایجاد می کند، سه عنصر را به آن اختصاص می دهد و سپس چاپ می کند:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

I like Volvo, BMW and Toyota.

همانطور که مشاهده می کنید برای دسترسی به یک عنصر آرایه کافیس اندیس آن را داخل [] قرار دهیم. حال گاهی لازم است به تک تک عناصر آرایه جهت انجام برخی عملیات دسترسی پیدا کنید، لذا از حلقه استفاده می شود. به عنوان مثال، برای حلقه زدن و چاپ تمام مقادیر یک آرایه نمایه شده، حلقه for به صورت زیر به کار گرفته می شود:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arlength = count($cars);

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

Volvo  
BMW  
Toyota

یکی از توابع تعریف شده برای آرایه ها تابع count است که طول عناصر آرایه را محاسبه می کند. در مثال ذکر شده تابع count(\$cars) تعداد عناصر آرایه \$cars را در متغیر \$arlength می ریزد. این مقدار برابر است با ۳. توجه داشته باشید که به طور کلی چاپ عناصر آرایه با دو دستور print\_r و var\_dump صورت می گیرد.

**آرایه های انجمنی PHP:** آرایه های انجمنی آرایه هایی هستند که از کلیدهای نامگذاری شده ای استفاده می کنند که شما به آنها اختصاص می دهید. دو راه برای ایجاد یک آرایه انجمنی وجود دارد:

۱. تعریف آرایه به طور یکجا، مانند زیر:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

۲. تعریف عناصر به صورت تک به تک:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

توسط کلیدهای در نظر گرفته شده برای عناصر می توان به عناصر آرایه های انجمنی دست پیدا کرد. به عنوان مثال کلیدهای نامگذاری شده را می توان به صورت زیر در یک اسکریپت استفاده کرد:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Peter is 35 years old.

حلقه در آرایه انجمنی: برای حلقه زدن و چاپ تمام مقادیر یک آرایه انجمنی، می توانید از یک حلقه foreach استفاده کنید:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

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

Key=Peter, Value=35  
Key=Ben, Value=37  
Key=Joe, Value=43

مثال ذکر شده کلیدها و مقادیر را به طور جداگانه چاپ می کند. در حلقه foreach ابتدا نام آرایه ذکر می گردد که در اینجا \$age می باشد و سپس با فرمت \$x => \$x\_value، کلیدها (\$x) و مقادیر آن ها (\$x\_value) از یکدیگر قابل تمیز دادن می شوند.

آرایه های چند بعدی در PHP: گاهی اوقات می خواهید مقادیر را با بیش از یک کلید ذخیره کنید. برای این منظور، آرایه های چند بعدی داریم. آرایه چند بعدی آرایه ای است که یک یا چند آرایه دارد که طول این آرایه ها می تواند با هم متفاوت باشد. PHP از آرایه های چند بعدی با عمق دو، سه، چهار، پنج یا بیشتر پشتیبانی می کند. با این حال، مدیریت

آرایه‌هایی با عمق بیش از سه سطح برای اکثر افراد دشوار است. بعد یک آرایه نشان دهنده تعداد اندیس‌هایی است که برای انتخاب یک عنصر نیاز دارید مثلاً برای یک آرایه دو بعدی برای انتخاب یک عنصر به دو اندیس نیاز دارید.

```
$cars = array (
    array("Volvo",22,18),
    array("BMW",15,13),
    array("Saab",5,2),
    array("Land Rover",17,15)
);
```

حالا آرایه دو بعدی \$cars شامل چهار آرایه است و دو اندیس دارد: ردیف و ستون. برای دسترسی به عناصر آرایه \$cars باید به دو اندیس (ردیف و ستون) اشاره کنیم:

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

```
Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.
```

حلقه در آرایه‌های چند بعدی: همچنین می‌توانیم یک حلقه for در حلقه for دیگری قرار دهیم تا عناصر آرایه \$cars را بدست آوریم (هنوز باید به دو اندیس اشاره کنیم):

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

## توابع در آرایه ها

در ادامه لیستی از توابع مهم در آرایه ها ارائه شده است.



اسم تابع	توضیح	مثال	خروجی
<b>Count()</b>	شمارش تعداد عناصر آرایه	<code>\$cars=array("Volvo","BMW","Toyota"); echo count(\$cars);</code>	3
<b>array_merge()</b>	ادغام دو آرایه در یک آرایه	<code>\$a1=array("red","green"); \$a2=array("blue","yellow"); print_r(array_merge(\$a1,\$a2));</code>	<pre>Array ( [0] =&gt; red [1] =&gt; green [2] =&gt; blue [3] =&gt; yellow )</pre>
<b>array_keys()</b>	ایجاد آرایه ای از کلیدها	<code>\$a=array("Volvo"=&gt;"XC90","BMW"=&gt;"X5","Toyota"=&gt;"Highlander"); print_r(array_keys(\$a));</code>	<pre>Array ( [0] =&gt; Volvo [1] =&gt; BMW [2] =&gt; Toyota )</pre>
<b>array_values()</b>	ایجاد آرایه ای از مقادیر	<code>\$a=array("Name"=&gt;"Peter","Age"=&gt;"41","Country"=&gt;"USA"); print_r(array_values(\$a));</code>	<pre>Array ( [0] =&gt; Peter [1] =&gt; 41 [2] =&gt; USA )</pre>
<b>array_combine()</b>	ایجاد یک آرایه با استفاده از کلیدها و مقادیر	<code>\$fname=array("Peter","Ben","Joe"); \$age=array("35","37","43"); \$c=array_combine(\$fname,\$age); print_r(\$c);</code>	<pre>Array ( [Peter] =&gt; 35 [Ben] =&gt; 37 [Joe] =&gt; 43 )</pre>
<b>Sort()</b>	مرتب سازی آرایه	<code>\$cars = array("Volvo", "BMW", "Toyota"); sort(\$cars);</code>	<pre>cars[0] = BMW cars[1] = Toyota cars[2] = Volvo</pre>
<b>in_array()</b>	بررسی وجود یک المان در آرایه (در array_search کلید المان را برمیگرداند).	<code>\$people = array("Peter", "Joe", "Glenn", "Cleveland"); if (in_array("Glenn", \$people)) {     echo "Match found"; } else {     echo "Match not found"; }</code>	Match found
<b>array_splice()</b>	حذف عناصر در آرایه	<code>\$a1=array("a"=&gt;"red","b"=&gt;"green","c"=&gt;"blue","d"=&gt;"yellow"); \$a2=array("a"=&gt;"purple","b"=&gt;"orange"); array_splice(\$a1,0,2,\$a2); print_r(\$a1);</code>	<pre>Array ( [0] =&gt; purple [1] =&gt; orange [c] =&gt; blue [d] =&gt; yellow )</pre>
<b>array_slice()</b>	برش آرایه	<code>\$a=array("red","green","blue","yellow","brown"); print_r(array_slice(\$a,2));</code>	<pre>Array ( [0] =&gt; blue [1] =&gt; yellow [2] =&gt; brown )</pre>
<b>array_shift()</b>	حذف عنصر اول و شیفت عناصر به چپ	<code>\$a=array("a"=&gt;"red","b"=&gt;"green","c"=&gt;"blue"); echo array_shift(\$a)."&lt;br&gt;"; print_r (\$a);</code>	<pre>red Array ( [b] =&gt; green [c] =&gt; blue )</pre>
<b>array_unshift()</b>	اضافه کردن عنصر به ابتدای آرایه	<code>\$a=array("a"=&gt;"red","b"=&gt;"green"); array_unshift(\$a,"blue"); print_r(\$a);</code>	<pre>Array ( [0] =&gt; blue [a] =&gt; red [b] =&gt; green )</pre>
<b>array_pop()</b>	حذف عنصر از انتهای آرایه	<code>\$a=array("red","green","blue"); array_pop(\$a); print_r(\$a);</code>	<pre>Array ( [0] =&gt; red [1] =&gt; green )</pre>
<b>array_push()</b>	اضافه کردن عنصر به انتهای آرایه	<code>\$a=array("red","green"); array_push(\$a,"blue","yellow"); print_r(\$a);</code>	<pre>Array ( [0] =&gt; red [1] =&gt; green [2] =&gt; blue [3] =&gt; yellow )</pre>

red blue brown	\$a=array("red","green","blue","yellow","brown"); \$random_keys=array_rand(\$a,3); echo \$a[\$random_keys[0]]." "; echo \$a[\$random_keys[1]]." "; echo \$a[\$random_keys[2]]." ";	ایجاد آرایه ای از کلیدهای رندوم	<b>array_rand()</b>
Array ( [0] => blue [1] => yellow )	\$a1=array("red","green"); \$a2=array("blue","yellow"); print_r(array_replace(\$a1,\$a2));	جایگزینی مقدار آرایه اول با آرایه دوم	<b>array_replace()</b>

## تمارین بخش آرایه

۱. برنامه ای بنویسید که اعدادی که بین ۲۰۰ تا ۲۵۰ قرار دارند و به ۴ بخش پذیر

هستند را داخل آرایه بریزید

```
$Numbers = array();
for ($i = 200; $i <= 250; $i++) {
    if ($i % 4 == 0) {
        $Numbers[] = $i;
    }
}
echo "<br> FIRST QUESTION: ";
for ($i = 0; $i < count($Numbers); $i++) {
    echo $Numbers[$i] . " - " ;
}
```

۲. برنامه ای بنویسید که از بین تعدادی عدد سه مینیمم اول را داخل یک آرایه بریزد

```
$Nums= array(33, 1, 8, 3, 18, 5, 15);
sort($Nums);
echo "<br> SECOND QUESTION: ";
for ($i = 0; $i < 3; $i++) {
    echo $Nums[$i] . " - " ;
}
```

۳. برنامه ای بنویسید که مقادیر تکراری یک آرایه را حذف کند.

```
$Numbers = array(1,3,1,3);
$Numbers= array_unique($Numbers);
echo "<br> THIRD QUESTION: ";
for ($i = 0; $i < count($Numbers); $i++) {
    echo $Numbers[$i] . " - " ;
}
```

۴. برنامه ای بنویسید که تعداد تکرار یک کاراکتر خاص در یک آرایه محاسبه کند.

```
$repCount=7;
$number=[2,9,7,6,8,7,7,0];
$totalCount=array_count_values($number);
echo '#repetition '.$repCount.' '. $totalCount[$repCount];
```

۵. برنامه ای بنویسید که در آرایه ای به طول ۱۰ عنصری را در اندیس شماره ۴ قرار دهد (عناصر از اندیس شماره ۴ یک خانه به جلو شیفت پیدا می کنند و طول آرایه ۱۱ می شود).

```
$Numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
$New=0;
array_splice($Numbers,4,0, $New);

echo "<br> FOURTH QUESTION: ";
for ($i = 0; $i < count($Numbers); $i++) {
    echo $Numbers[$i]. " - " ;
}
```

## فصل ششم – رشته ها

رشته ها در PHP مجموعه ای از کاراکترهای متوالی هستند که بین ‘، یا " قرار میگیرند.

```
$string = 'this is a PHP tutorial book';  
$string = "this is a PHP tutorial book";
```

در PHP ادغام دو رشته بسیار راحت است و توسط کاراکتر نقطه (.) صورت می گیرد. عملیات ادغام بین رشته و اعداد هم در PHP مقدور است.

```
$string1 = 'this is a PHP';  
$string2 = 'tutorial book';  
echo $string1 . $string2;
```

نکته جالب در PHP این است که تبدیل رشته به عدد به صورت خودکار صورت می گیرد. به عنوان مثال اگر رشته ای با عدد شروع شود، آن عدد قابلیت انجام عملیات ریاضی با سایر اعداد را دارد:

```
$string = '52PHP';  
$sum = 7 + $string;  
echo $sum; //59
```

### توابع در رشته

در ادامه برخی از توابع کاربردی در رشته ها ارائه شده است.

تابع	توضیحات
<a href="#"><u>strlen()</u></a>	طول یک رشته را برمی گرداند
<a href="#"><u>str_word_count()</u></a>	تعداد کلمات یک رشته را می شمارد
<a href="#"><u>strrev()</u></a>	یک رشته را معکوس می کند
<a href="#"><u>strpos()</u></a>	یک متن خاص را در یک رشته جستجو می کند. اگر مطابقت پیدا شود، تابع موقعیت کاراکتر اولین تطابق را برمی گرداند. اگر مطابقت پیدا نشد، FALSE برمی گردد
<a href="#"><u>str_replace()</u></a>	برخی از کاراکترها را با برخی از کاراکترهای دیگر در یک رشته جایگزین می کند
<a href="#"><u>strcmp()</u></a>	دو رشته را مقایسه می کند، در صورت تساوی عدد ۰ در غیراینصورت ۱ برمیگرداند
<a href="#"><u>str_split()</u></a>	کاراکترهای یک رشته را المان های آرایه تبدیل می کند
<a href="#"><u>implode()</u></a>	المان های یک آرایه را به هم متصل می کند و یک رشته می سازد. مقدار اول تابع می تواند " باشد تا جداکننده بین المان ها اسپیس باشد. علاوه بر اسپیس، سایر جداکننده ها نیز در این تابع قابل تعریف است
<a href="#"><u>explode()</u></a>	بر اساس جداکننده ای که برای این تابع تعریف می شود، عناصر رشته به المان های تابع نگاشت می شوند. به این معنا که هر کاراکتر بین جداکننده یک المان تابع می شود.

## تمارین بخش رشته

۱. برنامه ای بنویسید که اولین حرف رشته را بزرگ کند.

```
$str='thisIsaString';
$up=strtoupper($str[0]);
echo str_replace($str[0],$up,$str);
```

۲. برنامه ای بنویسید که سه حرف آخر یک رشته را چاپ کند.

```
$str='thisIsaString';
echo substr($str,strlen($str)-3);
```

۳. برنامه ای بنویسید که قسمت نام کاربری در یک ایمیل را چاپ کند. به عنوان

مثال برای ایمیل [youremail@gmail.com](mailto:youremail@gmail.com) مقدار youremail را چاپ کند.

```
$str='youremail@gmail.com';
$pos=strpos($str,'@');
echo substr($str,0,$pos)
```

۴. برنامه ای بنویسید که رشته ۰۹۲۲۳۷ را به ۰۹:۲۲:۳۷ تبدیل کند.

```
$i='094651';
echo substr($i,0,2).':';
echo substr($i,2,2).':';
echo substr($i,4,2);
```

۵. برنامه ای بنویسید که تمام صفرها را از رشته حذف کند.

```
$str='0740276940';
echo str_replace(0,"",$str);
```

## فصل هفتم - توابع

قدرت واقعی PHP از توابع آن ناشی می شود. PHP بیش از ۱۰۰۰ تابع داخلی دارد که می توان آنها را مستقیماً فراخوانی کرد. علاوه بر این توابع، این امکان وجود دارد که توابع شخصی سازی شده ایجاد کرد. یک تابع بلوکی از عبارات است که میتواند به طور مکرر در یک برنامه استفاده شود. یک تابع به طور خودکار هنگام بارگیری صفحه اجرا نمی شود بلکه با فراخوانی اجرا می شود.

```
<?php
function writeMsg() {
    echo "Hello world!";
}
writeMsg(); // call the function
?>
```

Hello world!

**PHP Function Arguments:** اطلاعات را می توان از طریق آرگومان ها به توابع منتقل کرد. یک آرگومان درست مانند یک متغیر است. آرگومان ها بعد از نام تابع در داخل پرانتز مشخص می شوند. می توانید هر تعداد آرگومان که می خواهید اضافه کنید، فقط آنها را با کاما جدا کنید. مثال زیر یک تابع با دو آرگومان \$fname و \$year دارد:

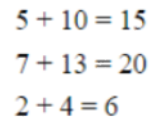
```
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}
familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

Hege Refsnes. Born in 1975  
Stale Refsnes. Born in 1978  
Kai Jim Refsnes. Born in 1983

## PHP Functions - Returning values: برای اینکه یک تابع مقداری را برگرداند، از

عبارت return استفاده کنید:

```
<?php
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}
echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```



5 + 10 = 15  
7 + 13 = 20  
2 + 4 = 6

## PHP Return Type Declarations: برای اعلام یک نوع برای تابع بازگشتی، هنگام

اعلان تابع، یک دونقطه ( : ) و نوع آن را درست قبل از براکت ( { ) بازکننده اضافه کنید. در مثال زیر نوع بازگشتی را برای تابع مشخص می کنیم:

```
<?php
function addNumbers(float $a, float $b) : float {
    return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```



6.4

## Passing Arguments by Reference: در PHP، آرگومان ها معمولا با مقدار ارسال

می شوند، به این معنی که یک کپی از مقدار در تابع استفاده می شود و متغیری که به تابع داده شده است قابل تغییر نیست. هنگامی که یک آرگومان تابع با مرجع ارسال می شود، تغییرات در آرگومان متغیری را که ارسال شده است نیز تغییر می دهد. برای تبدیل یک آرگومان تابع به مرجع، از عملگر & استفاده می شود:

```
function add_five(&$value) {
    $value += 5;
}
$num = 2;
add_five($num);
echo $num;
```



7



## تمارین بخش تابع

۱. تابعی برای ایجاد تصویر زیر (با استفاده از حلقه for):

```
*
**
***
****
*****
*****
*****
****
***
**
*
```

```
<?php
$stars = 1;
$limit = 5;
for ($i = 1; $i < $limit * 2; $i++) {
    for ($j = 1; $j <= $stars; $j++) {
        echo "*";
    }
    echo "<br>";

    if ($i >= $limit) {
        $stars -= 1;
    } else {
        $stars += 1;
    }
}
?>
```

۲. تابعی برای ایجاد کاراکتر A (با استفاده از حلقه while):

```
***
 *  *
 *  *
*****
 *  *
 *  *
 *  *
```

```
<?php
function createCharacterA(){
    echo "&nbsp;"."&nbsp;"."*****."<br>";
    line(2);
    echo "*****."<br>";
    line(3);
}
```

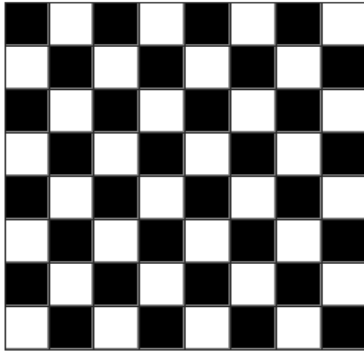
}

$$\left. \begin{array}{l} \{ \\ \{ \end{array} \right\}$$

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

}

۴. تابعی برای ایجاد صفحه شطرنج:



```
<?php
function generateChessboard() {
    $chessboard = '<table width="300px" height="300px" border="1px">';

    for ($row = 8; $row >= 1; $row--) {
        $chessboard .= '<tr>';
        for ($col = 1; $col <= 8; $col++) {
            $color = ($row + $col) % 2 ? 'black' : 'white';
            $chessboard .= '<td style="background-color: ' . $color . ';"></td>';
        }
        $chessboard .= '</tr>';
    }
    $chessboard .= '</table>';
    return $chessboard;
}
echo generateChessboard();
```

۵. تابعی بنویسید که چک کند آیا یک رشته متقارن است یا خیر؟ (رشته متقارن با معکوس خود برابر است)

```
<?php
function isSymmetrical($str) {
    $length = strlen($str);
    if ($length % 2 == 0) {
        $pointer1 = ($length / 2) - 1;
        $pointer2 = $length / 2;
        for ($i = 0; $i < (strlen($str) / 2); $i++) {
            if ($str[$pointer1] != $str[$pointer2])
                return false;
            $pointer1--;
            $pointer2++;
        }
    }
    else {
```

```

$pointer1 = (int)floor(($length / 2))-1;
$pointer2 = (int)ceil(($length / 2));
for ($i=0;$i<floor((strlen($str)/2));$i++){
    if($str[$pointer1]!=$str[$pointer2])
        return false;
    $pointer1--;
    $pointer2++;
}
return true;
}

```

```

$myString="ABBA";
$value=isSymmetrical($myString);

```

```

if($value)
echo "it is symetric ";
else
echo "it isn't symetric :(";

```

۶. تابعی برای پیاده سازی یک ماشین حساب بنویسید (با switch case).

```

<?php
case switch
function calculator($num1, $num2, $operator) {
    $result = 0;
    case switch
    switch ($operator) {
        case "add":
            $result = $num1 + $num2;
            break;
        case "sub":
            $result = $num1 - $num2;
            break;
        case "mul":
            $result = $num1 * $num2;
            break;
        case "div":
            if ($num2 == 0) {
                die("Cannot divide by zero");
            }
            $result = $num1 / $num2;
            break;
        default:
            die("Invalid operator");
    }
    return $result;
}
$num1 = 10;

```

```

$num2 = 11;
$operator = "div";
$result = calculator($num1, $num2, $operator);
echo "The result of $num1 $operator $num2 is $result";
?>

```

۷. تابعی بنویسید که چک کند یک آرایه زیرمجموعه آرایه دیگری است یا خیر.

```

<?php
$array1=[1,2,3,4,5,6];
$array2=[1,2,8];

if(isSubset($array1,$array2))
    echo "is subset";
else
    echo "isn't subset";

function isSubset($array1, $array2)
{
    $intersect = array_intersect($array1, $array2);
    if(empty($intersect))
        return false;
    else
        if ((count(array_diff($array2,$intersect)))==0) {
            return true;
        }
        return false;
}

```

۸. تابعی بنویسید که برای سه دانشجو آرایه انجمنی شامل نام، دروس ۱ و ۲ و ۳ تعریف کند. سپس مشخص کند کدام دو دانشجو با هم همکلاسی هستند (حداقل دو درس مشترک داشته باشند)

```

<?php
$student1 = array("name" => "Ali", "course1" => 'zaban', "course2" => 'tarikh', "course3" => 'riazi');
$student2 = array("name" => "Sara", "course1" => 'tarikh', "course2" => 'olum', "course3" => 'honar');
$student3 = array("name" => "Reza", "course1" => 'tarikh', "course2" => 'zaban', "course3" => 'olum');
//areClassmates($student1,$student2);
function areClassmates($array1, $array2) {
    $common_keys = array_intersect($array1, $array2);
    if (count($common_keys) >= 2) {
        return true;
    }
    return false;}
echo $student1["name"] . " and " . $student2["name"];
if (areClassmates($student1, $student2))
    echo " are classmates". "\n";
else

```

```

    echo " are not classmates"."\\n";
echo '</br>';
echo $student1["name"] . " and " . $student3["name"];
if (areClassmates($student1,$student3))
    echo " are classmates"."\\n";
else
    echo " are not classmates"."\\n";
echo '</br>';
echo $student3["name"] . " and " . $student2["name"];
if (areClassmates($student3,$student2))
    echo " are classmates"."\\n";
else
    echo " are not classmates"."\\n";

```

۹. آرایه ای ایجاد کنید شامل تعدادی اسم اسباب بازی و برای هر یک رنج سنی استفاده تعریف کنید. تابعی بنویسید که اسم اسباب بازی و سن را دریافت کند و بررسی کند ببیند آیا آن فرد مجاز به استفاده از اسباب بازی است یا خیر.

```

<?php
$toys = array(
    "teddy bear" => array("min_age" => 0, "max_age" => 10),
    "lego" => array("min_age" => 4, "max_age" => 12),
    "puzzle" => array("min_age" => 6, "max_age" => 15),
    "robot" => array("min_age" => 8, "max_age" => 18),
    "drone" => array("min_age" => 10, "max_age" => 20)
);
function canUseToy($toy_name, $age) {
    global $toys;
    if (isset($toys[$toy_name])) {
        $min_age = $toys[$toy_name]["min_age"];
        $max_age = $toys[$toy_name]["max_age"];
        if ($age >= $min_age && $age <= $max_age) {
            return "Yes, you can use the $toy_name.";
        } else {
            return "No, you cannot use the $toy_name.";
        }
    } else {
        return "I don't know this toy.";
    }
}
echo canUseToy("lego", 5) . "\\n"; // Yes, you can use the lego.
echo canUseToy("drone", 7) . "\\n"; // No, you cannot use the drone.
echo canUseToy("doll", 9) . "\\n"; // I don't know this toy.

```

۱۰. تابعی بنویسید که سه مقدار زیر را دریافت کرده و به عنوان استایل برای یک پاراگراف استفاده کند.

```
$color='blue';
$font_size='50';
$text_align='right';
```

```
<?php
function set_paragraph_style ($color, $font_size, $text_align) {
    $style= "<p style='color:$color; font-size:$font_size; text-align:$text_align'>";
    return $style;
}
echo set_paragraph_style ("blue", "50px", "left");
echo "این یک پاراگراف آزمایشی است";
echo "</p>";
```

## فصل هشتم – متغیرهای سراسری

در PHP برخی از متغیرهای از پیش تعریف شده وجود دارند که به آن‌ها "superglobals" گفته می‌شود. این متغیرها همیشه بدون توجه به محدوده ای که در آن قرار دارند، قابل دسترسی هستند. لذا می‌توان از هر تابع، کلاس یا فایلی بدون نیاز به انجام کار خاصی به آن‌ها دسترسی داشت. برخی از مهمترین متغیرهای superglobal عبارتند از:

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

\$GLOBAL: از این متغیر برای دسترسی به متغیرهای سراسری از هر نقطه برنامه استفاده می‌شود. PHP همه متغیرهای سراسری را در آرایه‌ای به نام \$GLOBALS[index] ذخیره می‌کند که index نام متغیر را نگه می‌دارد. مثال زیر نحوه استفاده از متغیر \$GLOBALS را نشان می‌دهد:

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

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}
addition();
echo $z;
?>
```

100



در مثال بالا، از آنجایی که  $z$  یک متغیر موجود در آرایه  $\$GLOBALS$  است، از خارج از تابع نیز قابل دسترسی است. در این مثال دو متغیر  $\$x$  و  $\$y$  در خارج از تابع `addition` تعریف شده اند که توسط  $\$GLOBAL$  در داخل این تابع قابل دسترسی هستند.

$\$_SERVER$ : تمامی داده هایی که سرور در یک پیام پاسخ HTTP به کاربر ارسال می کند داخل این آرایه ذخیره می شود. برخی از این اطلاعات عبارتند از:

$\$_SERVER['PHP\_SELF']$	نام فایل اسکریپت در حال اجرا را برمی گرداند.
$\$_SERVER['GATEWAY\_INTERFACE']$	نسخه CGI را که سرور از آن استفاده می کند، برمی گرداند.
$\$_SERVER['SERVER\_ADDR']$	آدرس IP سرور میزبان را برمی گرداند.
$\$_SERVER['SERVER\_NAME']$	نام سرور میزبان را برمی گرداند. مانند: <a href="http://www.w3schools.com">www.w3schools.com</a>
$\$_SERVER['SERVER\_PROTOCOL']$	نام و ورژن پروتکل اطلاعاتی مانند HTTP/1.1 را برمی گرداند.
$\$_SERVER['REQUEST\_METHOD']$	روش درخواست استفاده شده برای دسترسی به صفحه را برمی گرداند. مانند: POST
$\$_SERVER['REQUEST\_TIME']$	زمان شروع درخواست را برمی گرداند. مانند (timestamp)
$\$_SERVER['HTTP\_HOST']$	هدر میزبان از درخواست فعلی را برمی گرداند
$\$_SERVER['REMOTE\_ADDR']$	آدرس IP را از جایی که کاربر صفحه فعلی را مشاهده می کند، برمی گرداند.
$\$_SERVER['SCRIPT\_NAME']$	مسیر اسکریپت فعلی را برمی گرداند

`$_REQUEST`: متغیری است که برای جمع آوری داده ها پس از ارسال فرم HTML استفاده می شود. مثال زیر فرمی را با فیلد ورودی و دکمه ارسال نشان می دهد. هنگامی که کاربر داده ها را با کلیک بر روی "ارسال" ارسال می کند، داده های فرم به فایل مشخص شده در تگ `<form>` که توسط دستور زیر:

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

تعریف می شود، ارسال می شود. `$_SERVER['PHP_SELF']` نام فایل اسکریپت در حال اجرا را برمیگرداند و در این می خواهیم برای پردازش داده های فرم در اسکریپت فعلی صورت بگیرد. بنابراین با تغییر مقدار بازگردانده شده به `action` میتوان فایل دلخواهی را برای پردازش داده های فرم استفاده کرد. سپس، می توان از متغیر `$_REQUEST` برای جمع آوری مقدار فیلد ورودی استفاده کرد. لازم به ذکر است که از این متغیر برای هر دو حالت ارسال فرم با متد `post` و `get` قابل استفاده است:

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = htmlspecialchars($_REQUEST['fname']);
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
```

لازم به ذکر است که در HTML کاراکترهای از پیش تعریف شده ای مانند `<br>` وجود دارند که اگر کاربر امثال این عبارات را در فرم به عنوان داده ورودی وارد کند باعث ایجاد اختلال در اجرای برنامه می شود لذا جهت ایجاد امنیت از تابع `htmlspecialchars` استفاده می شود تا هر نوع کاراکتر را به عنوان رشته دریافت کند و تغییری در اجرای برنامه ایجاد نشود.

**\$\_POST**: زمانی که یک فرم داده های خود را با متد **post** به مقصد ارسال می کند، آن گاه این داده ها در اسکریپت مقصد با آرایه **\$\_POST** قابل دسترس خواهند بود. وقتی که فرم داده های خود را با این روش ارسال می کند، داده ها به صورت رمزنگاری شده ارسال می شوند و در **URL** مرورگر نمایش داده نمی شوند و این امنیت این متد نسبت به متد **get** می باشد.

**\$\_GET**: زمانی که یک فرم داده های خود را با متد **get** به مقصد ارسال می کند، آن گاه این داده ها در اسکریپت مقصد با آرایه **\$\_GET** قابل دسترس خواهند بود. وقتی که فرم داده های خود را با این روش ارسال می کند ، داده ها از طریق **URL** ارسال می شوند، لذا امنیت در این روش کم است اما سرعت آن نسبت به متد **post** بیشتر است.

## فصل نهم – اشکال زدائی و مدیریت حالات استثنائی

یک استثنا (Exception)، بیانگر خطا یا رفتار غیرمنتظره یک اسکریپت PHP می باشد که در طول برنامه امکان رخداد آن ها وجود دارد. اشکال زدائی عبارت است از رفع اشکالات و ایرادهای این چینی که در برنامه وجود دارد و معمولاً در زمان اجرا خود را نشان می دهد. با توجه به اینکه PHP خروجی محور است، شرایط مشاهده لحظه ای خطا و رفع آن محیاست. اما گاهی ممکن است خطاها از دید ما پنهان باشند و کاربر نهایی امکان بررسی منشا خطا را ندارد. در چنین شرایطی می توان با استفاده از امکاناتی که کامپایلر PHP می دهد بسیاری از خطاها را ریشه یابی کرد. یکی از مهمترین توابع در این زمینه تابع `error_reporting` است که مشخص می کند در هنگام رخداد خطا چه چیزهایی نمایش داده بشوند و چه چیزهایی نمایش داده نشوند. تکنیک دیگر استفاده از `try-catch` می باشد که قطعه کدهایی که مستعد خطا هستند داخل بلوک `try` قرار می دهیم و در بلوک `catch` قطعه کدی را قرار می دهیم که در صورت رخداد خطا در `try` اجرا شود.

```
try {
    code that can throw exceptions
} catch(Exception $e) {
    code that runs when an exception is caught
}
```

دستور `throw` به یک تابع یا متد تعریف شده توسط کاربر اجازه می دهد تا یک استثنا ایجاد کند. هنگامی که یک استثنا به بلوک `catch`، `throw` می شود، کد بعد از آن تا انتهای بلوک اجرا نمی شود و دستورات بلوک `catch` اجرا می شود. به عنوان مثال:

```

<?php
try {
    $dividend=5;
    $divisor=0;
    if($divisor == 0) {
        throw new Exception("Division by zero");
    }
    echo $dividend / $divisor;
}

} catch(Exception $e) {
    echo "Unable to divide.";
} finally {
    echo "Process complete.";
}
?>

```

Unable to divide. Process complete.

در صورت رخداد خطا، `new Exception` یک شی ایجاد می کند و توسط متد `throw` آن را به بلاک `catch` ارسال می کند (در این مثال شی جدید `$e` می باشد). بلاک `finally` یک بلاک اختیاری است که می توان به کد اضافه کرد، این بلاک در هر شرایطی اجرا می شود، چه استثنا رخ دهد چه ندهد.

## فصل دهم – دستور include و require

دستور include (یا require) تمام متن، کد و ... موجود در یک فایل را می گیرد و آن را در فایلی که از عبارت include استفاده کرده است کپی می کند. به این ترتیب می توان به جای نوشتن مجدد تمامی کد برنامه ای که در فایل دیگری موجود است، تنها با نوشتن دستور include (require) یک کپی از آن را در فایل جاری خود می توانیم داشته باشیم. گنجاندن فایل ها زمانی بسیار مفید است که می خواهید یک PHP، HTML یا متن را در چندین صفحه از یک وبسایت قرار دهید. گنجاندن فایل ها باعث صرفه جویی در کار می شود. این بدان معناست که شما می توانید یک فایل هدر، پاورقی یا منوی استاندارد برای تمام صفحات وب خود ایجاد کنید. سپس، زمانی که هدر باید به روز شود، فقط می توانید فایل شامل هدر را به روز کنید. دستور include و require، به جز در حالتی که خطا رخ دهد. دستور include در صورت رخداد خطا هشدار warning می دهد و به اجرای خود ادامه می دهد اما دستور require با ایجاد خطایی اجرای برنامه را متوقف می کند. لذا بسته به نوع کاربرد می توان از این دو دستور استفاده کرد.

```
include 'filename';
```

```
or
```

```
require 'filename';
```

## دستور `require_once` و `include_once`

این دو دستور مشابه دستورات `include` و `require` عمل می کنند با این تفاوت که پیش از کپی فایل بررسی می کنند اگر قبلاً کپی شده باشند مجدد آن ها را کپی نمی کنند. به عنوان مثال در برنامه نویسی های پیشرفته شما گاهی اوقات چند فایل را در هم ایمپورت می کنید. حالا تصور کنید شما در فایل `index.php`، فایل `x` و `y` را ایمپورت کرده اید. فایل `x` نیز خودش فایل `y` را درون خود ایمپورت کرده است. در این صورت شما دو بار فایل `y` را ایمپورت کرده اید. درست است؟ این کار اگر هم روال اجرای کدها را دچار مشکل نکند، باعث پردازش بیش از حد می شود.

## فصل یازدهم - فرم ها

تگ FORM در HTML برای تعامل با کاربر استفاده می شود. از طریق فرم ها می توان اطلاعاتی که کاربر وارد کرده است برای پردازش به صفحه دیگری (یا به همان صفحه) ارسال کرد. مواردی مثل جعبه های متن، دکمه های رادیویی، چک باکس ها، لیست های بازشونده و دکمه های ثبت را می توان در یک فرم قرار داد. ارسال اطلاعات فرم به دو صورت GET و POST انجام می شود و دسترسی به اطلاعات ارسال شده به ترتیب از دو متغیر سراسری \$\_GET و \$\_POST امکان پذیر است. هم GET و هم POST یک آرایه ایجاد می کنند (به عنوان مثال آرایه `key1 => value1, key2 => value2, key3 => value3, ...`) این آرایه جفت های کلید/مقدار را نگه می دارد، که key نام در نظر گرفته شده برای input فرم و value داده ورودی کاربر هستند. تفاوت این دو روش در این است که \$\_GET آرایه ای از متغیرها است که از طریق پارامترهای URL به اسکریپت فعلی ارسال می شود اما \$\_POST آرایه ای از متغیرها است که از طریق روش HTTP POST به اسکریپت فعلی ارسال می شود. ضمن ارائه مثال به بررسی این دو روش می پردازیم. مثال زیر یک فرم ساده HTML با دو فیلد ورودی و یک دکمه ارسال را نمایش می دهد:



```
<html>
<body>

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

</body>
</html>
```

هنگامی که کاربر فرم بالا را پر می کند و روی دکمه ارسال کلیک می کند، داده های فرم برای پردازش به فایل PHP به نام welcome.php ارسال می شود. برای نمایش داده های ارسالی، به قطعه کد زیر نیاز داریم:

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

این قطعه کد در اولین اجرا و پیش از وارد کردن ورودی دارای warning است زیرا متغیرهای `$_post["name"]` و `$_post["email"]` هنوز مقداری نگرفته اند (پس از پر کردن فرم و زدن گزینه ارسال مقدار در `$_post["name"]` و `$_post["email"]` قرار می گیرد) و امکان echo این متغیرها وجود ندارد. لذا بهتر است پیش از دستور echo یا هر دستور استفاده از پارامترهای ارسال شده به فرم، ابتدا مقداردهی شدن آن ها چک شود:

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

تابع `isset` چک می کند که `$_post["name"]` و `$_post["email"]` مقدار دهی شده اند یا خیر و در صورت مقداردهی شدن، مقدار آن ها را چاپ می کند. نکته حائز اهمیت دیگر این است که ممکن است فردی در در فرم از عبارات خاصی مثل تگ های HTML استفاده کند و می تواند اختلالاتی در اجرای کد داشته باشد. برای جلوگیری از این مشکل می توان از تابع `htmlspecialchars` استفاده کرد. با استفاده از این تابع، هر ورودی ارسال

شده توسط کاربر صرفاً به صورت یک رشته در نظر گرفته می شود و تغییری در کد ایجاد نمی کند. لذا داریم:

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

با فرض اینکه ورودی های فرم به صورت زیر باشد:

خروجی چاپ شده به صورت زیر است:

```
Welcome My Name
Your email address is: MyName@gmail.com
```

برای روش get هم همین موارد صدق می کند و و به ازای کد زیر خروجی مشابهی داریم:

```
<html>
<body>

<form action="get.php" method="GET">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
<br>
Welcome <?php if(!empty($_GET["name"])) echo htmlspecialchars($_GET["name"]); ?><br>
Your email address is: <?php if (!empty($_GET["email"])) echo htmlspecialchars($_GET["email"]); ?>

</body>
</html>
```

نکته تمایز روش get با روش post این است که در روش get آدرس مرورگر به این صورت تغییر می کند:

**127.0.0.1/get.php?name=My+Name&email=MyName%40gmail.com**

علامت سوال (?) به صورت یک جداکننده عمل می کند و پس از آن جفت مقادیر آرایه \$\_GET در آدرس URL قرار می گیرند و توسط & از هم جدا می شوند. در این مثال آرایه \$\_GET برابرست با:

```
array(2) { ["name"]=> string(7) "My Name" ["email"]=> string(16) "MyName@gmail.com" }
```

### چه زمانی از GET استفاده کنیم؟

اطلاعات ارسال شده از فرم با روش GET برای همه قابل مشاهده است (همه نام ها و مقادیر متغیرها در URL نمایش داده می شوند). GET همچنین محدودیت هایی در میزان ارسال اطلاعات دارد. محدودیت حدود ۲۰۰۰ کاراکتر است. با این حال، به دلیل اینکه متغیرها در URL نمایش داده می شوند، می توان از آن برای bookmark کرد. GET با توجه به سرعت بالاتری که نسبت به POST دارد می تواند برای ارسال داده های غیر حساس استفاده شود.

**توجه:** GET هرگز نباید برای ارسال رمز عبور یا سایر اطلاعات حساس استفاده شود!

### چه زمانی از POST استفاده کنیم؟

اطلاعات ارسال شده از فرم با روش POST برای دیگران غیرقابل مشاهده است. در این روش همه نام ها/مقادیر در بدنه درخواست HTTP تعبیه شده است و محدودیتی در میزان اطلاعات ارسالی ندارد. علاوه بر این POST روشی کارآمد برای آپلود فایل است. توسعه دهندگان POST را برای ارسال داده های فرم ترجیح می دهند.

## اعتبارسنجی فرم ها

همانطور که گفته شد تابع htmlspecialchars یکی از راه های جلوگیری از رفتار اختلال آمیز کاربران یا هکرهاست. کاربرد دیگر این تابع زمانی است که می خواهیم آدرس فایلی که قرار است اطلاعات فرم ارسال شود را در action یک تگ form قرار دهیم. یکی از فایل ها میتواند همان فایل جاری باشد که از طریق آرایه \$\_SERVER قابل دسترسی است. در این آرایه همانطور که در بخش متغیرهای سراسری گفته شد، اندیس PHP\_SELF در آرایه \$\_SERVER مربوط به اسم فایل جاری می باشد. لذا با استفاده از:

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

می توان ضمن جلوگیری از عملیات مخرب کاربران، نام فایل جاری را در action وارد می کند و اطلاعات فرم به فایل جاری ارسال می شود و در صورت رخداد خطا در همان صفحه قابل مشاهده است. در این حالت اگر کاربر در قسمت URL نام فایل را تغییر دهد قادر به ایجاد تغییر در روال برنامه نیست.

علاوه بر این برای امنیت بیشتر می توان با دو تابع trim و stripslashes می توان نسبت به حذف به ترتیب کاراکترهای غیرضروری (مثل اسپیس، تب، خط جدید) و (\) ها از ورودی کاربر پرداخت. با این حساب ترجیح بر این است که برای انجام این کارهای تکراری برای مجموعه ای از داده، تمام عملیات اعتبارسنجی را داخل یک تابع انجام داد و برای هر ورودی آن را فراخوانی کرد. در مثال ذکر شده برای این موضوع نام تابع test\_input می باشد:

```

<?php
// define variables and set to empty values
$name = $email = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = test_input($_POST["name"]);
    $email = test_input($_POST["email"]);
}

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

```

توجه داشته باشید که در ابتدای اسکریپت، بررسی می‌کنیم که آیا فرم با استفاده از روش POST ارسال شده است یا خیر. اگر `REQUEST_METHOD` برابر با `POST` باشد، فرم باید اعتبارسنجی شود. اگر ارسال نشده است، از تأیید اعتبار رد می‌شود و یک فرم خالی نمایش می‌دهد.

در بحث اعتبارسنجی فرم‌ها در مواقعی لازم است فرمت ورودی‌ها بررسی شود که آیا ورودی با فرمت مدنظر تطابق دارد یا خیر. به طور ویژه برای ایمیل و URL که همانطور که میدانیم فرمت خاصی دارند، به عنوان مثال `email@gmail.com` فرمت صحیحی برای یک ایمیل نیست و یا `this is a link` نمی‌تواند فرمت صحیحی برای یک URL باشد. روش‌های متفاوتی برای این منظور تعریف شده است که در ادامه به بررسی این روش‌ها می‌پردازیم:

۱. استفاده از تابع `filter_var`: تابع `filter_var()` یکی از توابع ارزیابی می‌باشد که بر اساس پارامترهای ورودی آن کاربردهای مختلفی می‌تواند داشته باشد. به عنوان مثال:

```

$email = "some@email.com";
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $emailErr = "Invalid email format";
}

```

پارامتر \$email ایمیلی است که به ارزیابی فرمت آن می خواهیم پردازیم و پارامتر FILTER\_VALIDATE\_EMAIL فیلتر تعریف شده برای بررسی فرمت \$email است. در این تابع سایر ارزیابی ها با پارامترهای دیگری مقدور است که برخی از آن ها در جدول زیر آمده است:

پارامتر دوم	
ارزیابی فرمت int	FILTER_VALIDATE_INT
ارزیابی فرمت float	FILTER_VALIDATE_FLOAT
ارزیابی فرمت IP	FILTER_VALIDATE_IP
حذف کاراکتر غیرمجاز از URL	FILTER_SANITIZE_URL

همانطور که میبینید FILTER\_VALIDATE فقط بررسی می کند اما FILTER\_SANITIZE به حذف کاراکترهای غیر مجاز می پردازد. به عنوان مثال:

```
<?php
// Variable to check
$email = "john.doe@example.com";

// Remove all illegal characters from email
$email = filter_var($email, FILTER_SANITIZE_EMAIL);
echo $email;

// Validate e-mail
if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo("$email is a valid email address");
} else {
    echo("$email is not a valid email address");
}
?>
```

filter\_var(\$email, FILTER\_SANITIZE\_EMAIL) ابتدا کاراکترهای غیرمجاز مثل کاراکترهای خاص (¥£ و ...) را از داخل \$email حذف می کند و سپس جهت ارزیابی فرمت از تابع filter\_var(\$email, FILTER\_VALIDATE\_EMAIL) استفاده می شود.

۲. استفاده از تابع `preg_match`: تفاوت این روش با روش قبل در این است که می توانیم فرمت ورودی را خودمان تعریف کنیم. به عنوان مثال:

```
<?php
// Variable to check
$email = "john.doe@example.com";

if (preg_match("~([a-zA-Z0-9!#$%&'*/+./=?^`{|}~])@([a-zA-Z0-9-]).([a-zA-Z0-9]{2,4})~", $email)) {
    echo 'This is a valid email.';
} else {
    echo 'This is an invalid email.';
}
?>
```

در این مثال `~([a-zA-Z0-9!#$%&'*/+./=?^`{|}~])@([a-zA-Z0-9-]).([a-zA-Z0-9]{2,4})~` کاراکترهای مجاز با فرمت تعریف شده برای ایمیل به عنوان پارامتر ورودی اول تابع در نظر گرفته شده است که فرمت `$email` را ارزیابی می کند. فرمت تعریف شده شامل بخش های زیر است:

۱. `~([a-zA-Z0-9!#$%&'*/+./=?^`{|}~])`
۲. `@`
۳. `([a-zA-Z0-9-])`
۴. `.`
۵. `([a-zA-Z0-9]{2,4})~`

که تداعی کننده بخش های یک ایمیل است.

## فیلدهای اجباری در فرم ها

در مثال های ذکر شده، تمام فیلدهای ورودی اختیاری بودند، به این معنا که اگر کاربر برای آن ها مقداری درج نمی کرد با خطایی مواجه نمیشد و داده های او ارسال می شد. در این بخش با اضافه کردن برخی فیلدهای اجباری کاربر را ملزم به وارد کردن مقدار برای فیلد مورد نظر می کنیم. برای این منظور می توان از متغیری جدید برای هر فیلد اجباری تعریف کنیم که خروجی مربوطه در صورت رخداد خطا (خطا در اینجا عدم مقدار دادن به فیلد) را

در خود نگه دارد. به عنوان مثال، در کد زیر متغیر `$nameErr` پیام خطای خالی بودن فیلد اجباری را در خود نگه می دارد. در مثال پیش رو با استفاده از دستورات شرطی `if else` مقداردهی شدن متغیر `$_POST` بررسی می گردد؛ به عبارتی اگر متغیر `$_POST` خالی باشد به این معناست که مقداردهی نشده است. در نتیجه زمانی که یک فیلد را اجباری در نظر بگیریم با استفاده از این شرط داریم:

خالی باشد `$_POST` اگر

آنگاه پیغام خطا را متغیر در نظر گرفته شده برای خطای آن فیلد بریز

در غیر این صورت

مقدار آن را دریافت کن

```
<?php
// define variables and set to empty values
$nameErr = "";
$name = "";

if (empty($_POST["name"])) {
    $nameErr = "Name is required";
} else {
    $name = ($_POST["name"]);
}

}
```

نمونه کاملی از یک فرم با فیلد اجباری و همراه با اعتبارسنجی در مثال زیر آورده شده است.

```
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
$nameErr = $emailErr = $genderErr = $websiteErr = "";
```



```

$name = $email = $gender = $comment = $website = "";

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

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

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

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

if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
} else {
    $gender = test_input($_POST["gender"]);
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

```

```

}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>
<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>

```

در این مثال ابتدا شروط بررسی خالی بودن/نبودن فیلد ذکر شده است، در بخش بعد تابع `input_test` تعریف شده است که به عنوان تابع ارزیابی (که در بخش گذشته توضیح داده شده بود) به کار می رود. سپس کدی شامل فرم ارائه گردیده است و در بخش انتهایی کد

۷۰

خروجی چاپ می گردد. لازم به ذکر است که همانطور که در شکل زیر مشاهده می کنید پیغام های خطا در این برنامه در کنار فیلد چاپ شده اند لذا دستورات حاوی چاپ مقادیر خطا نیز در بخش html فرم نوشته می شوند. به عنوان مثال قطعه ای از کد فرم مربوط به فیلد name برابر است با:

```
Name:<input type="text" name="name">
<span class="error">*<?php echo $nameErr;?></span>
```

که دستور <?php echo \$nameErr;?> مربوط به چاپ خطا در صورت رخداد می باشد.

**PHP Form Validation Example**

**\* required field**

Name:  **\* Name is required**

E-mail:  **\* Email is required**

Website:

Comment:

Gender: ☐ Female ☐ Male ☐ Other **\* Gender is required**

**Your Input:**

## تمارین بخش فرم

۱. برنامه ای بنویسید که در آن فرمی استایل پاراگراف را دریافت کند و آن را در پاراگراف اعمال کند. فرم شامل نام، سائیز، رنگ (قرمز، آبی، سبز و زرد) و توازن متن (چپ چین، وسط چین و راست چین) می باشد.

```
<html>
<?php
```

Y1

```
$name=$color=$align="";

if(isset($_POST['name'])) {
    $name = $_POST['name'];
} else {
    $name = 'My Name';
}

if(isset($_POST['size'])) {
    $size = $_POST['size'];
} else {
    $size=25;
}

if(isset($_POST['color'])) {
    $color = $_POST['color'];
} else {
    $color='black';
}

if(isset($_POST['align'])) {
    $align = $_POST['align'];
} else {
    $align = 'right';
}

echo '<p style="color: '.$color.';text-align:'.$align.';font-size: '.$size.'">';
echo 'Hello, I am '.$name.', I want change the style of this paragraph, I choose color '.$color.'';
echo '</p>';
?>

<form action="index.php" method="post">
    Name: <input type="text" name="name"><br>
    Font size: <input type="number" name="size"><br>
    Choose color:<br>
    Red<input type="radio" name="color" value="red">
    Blue<input type="radio" name="color" value="blue">
    Green<input type="radio" name="color" value="green">
    Yellow<input type="radio" name="color" value="yellow"><br>
    Text align:<br>
    Left<input type="radio" name="align" value="left">
    Right<input type="radio" name="align" value="right">
    Center<input type="radio" name="align" value="center"><br>
    <input type="submit" value="submit" name="submit">
</form>

</html>
```

۲. برنامه ای بنویسید که در فرم دو عدد از کاربر بگیرد که به عنوان سطر و ستون در نظر گرفته می شوند. سپس در آن اعداد از یک شروع به چاپ کنند و هر بار یک واحد یک واحد اضافه شوند.

```
<?php
```

```
echo "Enter n for nxn : <br>";
echo "<form method='POST'>
    Row:<input type='number' min='2'
        max='5' name='1d' value='1'/>
    Column:<input type='number' min='2'
        max='5' name='2d' value='1'/>
    <input type='submit' name='submit'
        value='Submit'/>
</form>";
```

```
// Submit user input data for 2D array
if (isset($_POST['submit'])) {
```

```
    // POST submitted data
    $dimention1 = $_POST["1d"];
```

```
    // POST submitted data
    $dimention2 = $_POST["2d"];
```

```
    echo "Entered 2d nxn: " . $dimention1
        . "x" . $dimention2 . " <br>";
    $d = [];
    $k = 0;
```

```
    for($row = 0; $row < $dimention1; $row++) {
        for ($col = 0; $col < $dimention2; $col++) {
            $d[$row][$col] = $k++;
        }
    }
```

```
    for ($row = 0; $row < $dimention1; $row++) {
        for ($col = 0; $col < $dimention2; $col++) {
            echo $d[$row][$col]. " ";
        }
        echo "<br>";
    }
}
?>
```

## فصل دوازدهم – فایل ها

مدیریت فایل بخش مهمی از هر برنامه وب است. شما اغلب نیاز به باز کردن و پردازش یک فایل برای کارهای مختلف دارید. در php چندین راه برای ایجاد، خواندن، آپلود و ویرایش فایل ها دارد. نکته قابل توجه این است که دستکاری و هرگونه اعمال تغییر در فایل ها بایستی با دقت صورت گیرد زیرا در صورت اشتباه امکان ایجاد خطاهای زیادی در برنامه وجود دارد که برخی از آن ها عبارتند از: ویرایش فایل اشتباه، پر کردن هارد دیسک با داده های بی ارزش و حذف محتوای یک فایل به صورت تصادفی. در ادامه این بخش به بررسی توابع موجود می پردازیم:

**تابع readfile():** این تابع یک فایل را می خواند و آن را در بافر خروجی می نویسد. فرض کنید ما یک فایل متنی به نام "filename.txt" داریم که در سرور ذخیره شده است که محتوای آن به شکل زیر است:

AJAX = Asynchronous JavaScript and XML  
 CSS = Cascading Style Sheets  
 HTML = Hyper Text Markup Language  
 PHP = PHP Hypertext Preprocessor  
 SQL = Structured Query Language  
 SVG = Scalable Vector Graphics  
 XML = EXtensible Markup Language

در ادامه مثال های این بخش از این فایل استفاده می کنیم. کد PHP برای خواندن فایل به صورت زیر است:

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

در این تابع تعداد بایت هایی که با موفقیت خوانده شده اند در انتهای خروجی چاپ می شود. خروجی مثال ذکر شده:

AJAX = Asynchronous JavaScript and XML CSS =  
Cascading Style Sheets HTML = Hyper Text Markup  
Language PHP = PHP Hypertext Preprocessor SQL =  
Structured Query Language SVG = Scalable Vector  
Graphics XML = EXtensible Markup Language 236

تابع `readfile` در صورتی می تواند کاربردی باشد که تنها کاری که می خواهید انجام دهید باز کردن یک فایل و خواندن محتویات آن باشد. یک روش بهتر برای باز کردن فایل ها با تابع `fopen()` است. این تابع گزینه های بیشتری نسبت به تابع `readfile` در اختیار شما قرار می دهد. پارامتر اول `fopen` حاوی نام فایلی است که باید باز شود و پارامتر دوم مشخص می کند که فایل در چه حالتی باید باز شود (در جدول زیر برخی از حالت هایی که فایل می تواند باز شود آورده شده است). اگر تابع `fopen` نتواند فایل مشخص شده را باز کند با استفاده از تابع `die` پیغام خطا چاپ می شود و اجرای اسکریپت فعلی متوقف می شود:

```
$myfile = fopen("filename.txt", "r") or die("Unable to open file!");
```

حالت فایل	عملکرد	توضیحات
<b>R</b>	فقط خواندن	نشانگر فایل از ابتدای فایل شروع می شود
<b>W</b>	فقط نوشتن	محتویات فایل را پاک می کند یا در صورت نبود فایل جدید ایجاد می کند. نشانگر فایل از ابتدای فایل شروع می شود
<b>A</b>	فقط نوشتن	داده های موجود در فایل حفظ می شود. نشانگر فایل از انتهای فایل شروع می شود. در صورت عدم وجود فایل، فایل جدیدی ایجاد می کند

<b>x</b>	یک فایل جدید فقط برای نوشتن ایجاد می کند	اگر فایل از قبل وجود داشته باشد FALSE را برمی‌گرداند و یک خطا را نشان می‌دهد
<b>r+</b>	خواندن/نوشتن	نشانگر فایل از ابتدای فایل شروع می‌شود
<b>w+</b>	خواندن/نوشتن	محتویات فایل را پاک می‌کند یا در صورت نبود فایل جدید ایجاد می‌کند. نشانگر فایل از ابتدای فایل شروع می‌شود
<b>a+</b>	خواندن/نوشتن	داده‌های موجود در فایل حفظ می‌شود. نشانگر فایل از انتهای فایل شروع می‌شود. در صورت عدم وجود فایل، فایل جدیدی ایجاد می‌کند
<b>x+</b>	یک فایل جدید برای خواندن/نوشتن ایجاد می‌کند	اگر فایل از قبل وجود داشته باشد FALSE را برمی‌گرداند و یک خطا را نشان می‌دهد

لازم به ذکر است که اگر در زمان باز کردن فایل با تابع `fopen`، فایل با نام ذکر شده موجود نباشد، فایل ایجاد خواهد شد. بنابراین از تابع `fopen` برای ایجاد فایل هم می‌توان استفاده کرد. نکته حائز اهمیت این است که زمان ایجاد فایل با `fopen`، پارامتر دوم باید یکی از مقادیر `w` یا `a` که به منظور نوشتن استفاده می‌شوند باشد.

تابع `fread()` این تابع می‌تواند یک فایلی که قبلاً باز شده بخواند. پارامتر اول `fread` حاوی نام فایلی است که باید از آن خوانده شود و پارامتر دوم حداکثر تعداد بایت‌های خواندنی را مشخص می‌کند.

```
fread($myfile, filesize("filename.txt"));
```

تابع `fclose()` برای بستن یک فایل باز استفاده می‌شود. باز بودن یک فایل و اجرای آن بدون اینکه نیازی به آن باشد باعث اشغال بی‌دلیل حافظه می‌شود. این تابع برای ورودی



تنها به نام فایل (یا متغیری که نام فایل را نگه می دارد) که می خواهیم ببندیم نیاز دارد. مثال زیر یک تابع را باز میکند، آن را می خواند و در نهایت می بندد:

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

**تابع fgetc():** تابع fgetc برای خواندن یک خط از یک فایل استفاده می شود. مثال زیر خط اول فایل "filename.txt" را خروجی می دهد:

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

AJAX = Asynchronous JavaScript and XML

پس از فراخوانی تابع fgetc، نشانگر فایل به خط بعدی منتقل می شود. **تابع feof():** تابع feof بررسی می کند که آیا به "انتهای فایل" (EOF) رسیده است یا خیر در نتیجه برای خواندن فایل هایی با طول نامعلوم کاربردی است. مثال زیر فایل "filename.txt" را خط به خط می خواند تا زمانی که به انتهای فایل برسد:

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

**تابع fgetc():** تابع fgetc برای خواندن یک کاراکتر از یک فایل استفاده می شود. مثال زیر کاراکتر به کاراکتر فایل "filename.txt" را می خواند تا زمانی که به انتهای فایل برسد:

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

**تابع fwrite()**: تابع `fwrite` برای نوشتن روی یک فایل استفاده می شود. پارامتر اول این تابع شامل نام فایلی است که می خواهیم روی آن بنویسیم و پارامتر دوم رشته ای است که باید نوشته شود. مثال زیر چند نام را در یک فایل جدید به نام `"newfile.txt"` می نویسد:

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "First\n";
fwrite($myfile, $txt);
$txt = "Secound\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

بعد از اینکه فایل `"newfile.txt"` را بستیم در صورتی که آن را مجدد باز کنیم و بخواهیم اطلاعاتی روی آن بنویسیم اطلاعات قبلی آن پاک می شود. برای رفع این مشکل از حالت باز کردن فایل `"a"` به جای `"w"` استفاده می کنیم. حالت `"a"` متن را به انتهای فایل اضافه می کند، در حالی که حالت `"w"` محتوای قدیمی فایل را پاک می کند. در مثال زیر فایل موجود خود `"newfile.txt"` را باز می کنیم و متنی به آن اضافه می کنیم و مشکلی برای داده های قبلی روی فایل به وجود نمی آید:

```
<?php
$myfile = fopen("newfile.txt", "a") or die("Unable to open file!");
$txt = "Donald Duck\n";
fwrite($myfile, $txt);
$txt = "Goofy Goof\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

## بارگذاری فایل

در PHP امکان آپلود فایل بسیار راحت است اما لازم است قبل از انجام این کار اجازه آپلود فایل را از طریق دستور زیر به برنامه داد. این دستور را لازم است در فایل “php.ini” (فایل تنظیمات پروژه) در قسمت file\_uploads قرار داد.

```
file_uploads = On
```

در ادامه لازم است در فرم HTML ورودی از نوع فایل تعریف شود:

```
<form action="upload.php" method="post" enctype="multipart/form-data">
  Select image to upload:
  <input type="file" name="fileToUpload" id="fileToUpload">
  <input type="submit" value="Upload Image" name="submit">
</form>
```

نکته قابل توجه این است که متد ارسالی فرم حتما باید post باشد. مورد دیگری که از الزامات فرم آپلود فایل می باشد این است که از enctype="multipart/form-data" به عنوان ویژگی فرم استفاده شود. این ویژگی مشخص می کند چه نوع داده ای توسط فرم ارسال می شود. در این فرم، ورودی type="file" باعث می شود که در کنار فیلد مربوط به آن دکمه ای قرار گیرد که با کلیک کردن آن کاربر می تواند فایل مورد نظر خود را از پنجره باز شده انتخاب و در نهایت آپلود کند. فرم ذکر شده اطلاعات را به فایل “upload.php” ارسال می کند که در این فایل داریم:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . ($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
?>
```

uploads را به عنوان فولدري در نظر مي گيريم که فايل هاي آپلود شده در آن قرار خواهند گرفت. لذا لازم است در پروژه خود اين فولدر را بسازيم. با توجه به کد ارائه شده داريم:

- **\$target\_dir**: مسيري که قرار است فايل در آن قرار گيرد را مشخص مي کند که ما در اينجا فولدر uploads را ايجاد و براي اين منظور در نظر گرفته ايم.

- **\$target\_file**: مسير فايلي که آپلود شده به همراه نام فايل را مشخص مي کند. متغير سراسري **\$\_FILES** یک آرايه انجمني است که اطلاعاتي از قبيل نام، اندازه، نوع و ... فايل آپلود شده را در خود نگه مي دارد. براي دسترسي به عناصر اين آرايه براي انديس اول نام فايل و براي انديس دوم کليدواژه هايي مثل **name**، **size**، **type** و ... قرار مي گيرد که به ترتيب نام، نوع و اندازه فايل را براي ما مشخص مي کند. بنابر اين با **\$\_FILES["fileToUpload"]["name"]** مي توانيم با نام فايل دسترسي داشته باشيم و با اتصال آن به **\$target\_dir** به مسير دسترسي به فايل دست يابيم.

- **\$uploadOk**: یک بولين است که در ابتدا مقدار ۱ مي گيرد و در طول برنامه جهت تعيين موفقيت/عدم موفقيت آپلود فايل به کار مي رود.

- **\$imageFileType**: تابع **pathinfo** آرايه اي را برمي گرداند که در آن مسير فايل، نام فايل و نوع فايلي که به عنوان ورودی گرفته است را برمي گرداند. در **pathinfo(\$target\_file,PATHINFO\_EXTENSION)** نوع فايل پارامتر اول يعني **\$target\_file** توسط پارامتر دوم يعني **PATHINFO\_EXTENSION** برگردانده مي شود.

در ادامه برخي از محدوديت هايي که براي آپلود فايل مي توان در برنامه لحاظ کرد ارائه شده است:

محدودیت	قطعه کد	توضیح قطعه کد
بررسی وجود داشتن/نداشتن فایل قبل از آپلود آن	<pre>if (file_exists(\$target_file)) {     echo "Sorry, file already exists.";     \$uploadOk = 0; }</pre>	تابع file-exists با توجه به مسیر \$target_file می تواند وجود یا عدم وجود فایل را بررسی کند.
اعمال محدودیت سائز برای فایل آپلودی	<pre>if (\$_FILES["fileToUpload"]["size"] &gt; 5000) {     echo "Sorry, your file is too large.";     \$uploadOk = 0; }</pre>	داخل آرایه سراسری \$_FILE در خانه ["fileToUpload"]["size"] سائز فایل آپلود شده قرار می گیرد
اعمال محدودیت نوع برای فایل آپلودی	<pre>if(\$imageFileType != "jpg" &amp;&amp; \$imageFileType != "png" &amp;&amp; \$imageFileType != "jpeg" &amp;&amp; \$imageFileType != "gif" ) {     echo "Sorry, only JPG, JPEG, PNG &amp; GIF files are allowed.";     \$uploadOk = 0; }</pre>	با توجه به عملگر && اگر فایل هیچ یک از انواع داده jpg, png, jpeg, png نباشد پیغام خطا دریافت خواهد کرد.

در ادامه کد کاملی جهت آپلود فرم ارائه شده است. در قسمت اول فرمی است که توسط آن امکان آپلود فایل فراهم می گردد و بخش دوم هم همانطور که پیش تر ذکر شد مقداردی پاراترهای مورد نیاز می باشد. بخش سوم به منظور اعمال محدودیت برای ورودی در کد لحاظ شده است؛ شرط اول جهت چک کردن وجود/عدم وجود فایل، شرط دوم جهت محدود کردن سائز و شرط سوم جهت محدود کردن نوع فایل به کار برده شده است. بخش آخر در ابتدا بررسی می کند که آیا فایل آپلود شده است یا خیر، در صورتی که فایل آپلود شده باشد (\$uploadOk = 1) آن گاه لازم است که فایل را در مسیر مدنظر قرار دهیم که این امر توسط تابع move\_uploaded\_file صورت می گیرد. پاراتر اول این تابع نام فایل روی سرور است و پاراتر دوم آن آدرسی است که در قصد داریم در آن فایل را قرار دهیم. پاراتر اول تابع توسط \$\_FILES["fileToUpload"]["tmp\_name"] به دست می آید که نام فایل موقت روی سرور است.

```

<form action="upload.php" method="post" enctype="multipart/form-data">
  Select image to upload:
  <input type="file" name="fileToUpload" id="fileToUpload">
  <input type="submit" value="Upload Image" name="submit">
</form>
-----
<?php
$target_dir = "uploads/";
$target_file = $target_dir . $_FILES["fileToUpload"]["name"];
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
-----
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}

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

// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
-----
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
    // if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
        echo "The file ". htmlspecialchars($_FILES["fileToUpload"]["name"]). " has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>

```

## فصل سیزدهم - کوکی ها<sup>۱</sup>

در هنگام کار با وبسایت ها در مواردی مثل خرید از سایت لازم است اطلاعاتی راجع به کاربر (در این مثال محصولات خریداری شده او به همراه تعداد آن ها و ...) ذخیره شود. در صورت عدم ذخیره اطلاعات زمان بیهوده برای وارد کردن مجدد صرف می شود. ذخیره سازی این قبیل اطلاعات توسط کوکی ها صورت می گیرد. کوکی اغلب برای شناسایی و احراز هویت کاربر استفاده می شود. کوکی یک فایل کوچک است که توسط دستوراتی که در کد یک وبسایت در نظر گرفته ایجاد می شود، سپس در کامپیوتر کاربر قرار می گیرد و هر بار که کاربر در مراجعه مجدد آن وبسایت را در مرورگر درخواست می کند، کوکی را نیز برای آن ارسال می کند. این امر باعث می شود که کاربر مجبور به حفظ اطلاعات احراز هویت مربوط به هر سایت نباشد. علاوه بر این کوکی به عنوان یک ردیاب آنلاین عمل می کند به این صورت که رفتار کاربر را بررسی می کند و بر اساس آن می تواند نیازها و علاقه-مندی های کاربر آگاهی پیدا کند. بسیاری از شرکت های تبلیغاتی با شناخت رفتار کاربر قادر هستند برای او تبلیغات هدفمند مطابق سلیقه و نیازش ارسال کنند. داده هایی که در کوکی ها ذخیره می شوند به صورت رشته های متنی هستند، بنابراین هر اطلاعاتی از کاربر که بتواند در غالب رشته نگهداری شود، قابل ذخیره سازی در کوکی می باشد. با PHP، هم می توان مقادیر کوکی را ایجاد و هم بازیابی کرد. یک کوکی با تابع `setcookie()` ایجاد می شود که پارامترهای مختلفی مثل نام، مقدار، تاریخ انقضا و ... می گیرد:

---

<sup>1</sup> cookie

```

<?php
$cookie_name = "user";
$cookie_value = "John Doe";
$secure = true;
$httponly = true;
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/", $secure, $httponly);
?>
<html>
<body>

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

</body>
</html>

```

مثال بالا یک کوکی به نام "user" با مقدار "John Doe" ایجاد می کند. کوکی پس از ۳۰ روز منقضی می شود (۸۶۴۰۰ \* ۳۰). "/" به این معنی است که کوکی در کل وب سایت موجود است (در غیر این صورت، می توان دایرکتوری مورد نظر خود را انتخاب کرد). متغیر \$secure بولین است و در صورتی که true باشد نشان دهنده این است که کوکی تنها در ارتباطات https و تحت یک ارتباط امن قابل ارسال به سایت است. متغیر \$httponly نیز یک متغیر بولین است که در صورت true بودن، لازم است که ارتباطات فقط تحت پروتکل HTTP باشد و در سایر ارتباطات که توسط جاوا اسکریپت یا سایر پروتکل ها برقرار می شوند تبادل نخواهد شد.

مقدار کوکی "user" با از استفاده از آرایه سراسری \$\_COOKIE بازیابی می شود. همچنین از تابع isset برای بررسی اینکه آیا کوکی تنظیم شده است یا خیر استفاده می کنیم. لازم به ذکر است که تابع setcookie باید قبل از <html> به کار برده شود. برای اصلاح یک کوکی کافی است مجدد با تابع setcookie مقداردهی جدید صورت گیرد و



برای حذف یک کوکی لازم است در تابع `setcookie` علاوه بر نام کوکی که تنها پارامتر اجباری این تابع است، تاریخ انقضایی برای گذشته به تابع داد. به عنوان مثال: `time() - 3600`

**امنیت کوکی ها:** کوکی ها قابلیت حذف به صورت دستی دارند که از نظر امنیتی کار درستی نیست. علاوه بر این کوکی یک فایل متنی است که قابل خواندن در کامپیوتری است که بر روی آن ذخیره شده است. توصیه می شود که اطلاعات حساس در کوکی ذخیره نشود زیرا کوکی ها قابل دسترسی است (حتی خارج از سیستم کاربر).

## فصل چهاردهم - نشست ها<sup>۱</sup>

وقتی کاربر با یک برنامه کار می کند، ابتدا آن را باز می کند، تغییراتی را انجام می دهد و سپس آن را می بندد. کامپیوتر کاربر می داند که او کیست، چه زمانی برنامه را شروع می کند و چه زمانی پایان می دهد، اما در فضای اینترنت وب سرور چنین اطلاعاتی ندارد. یک نشست راهی برای ذخیره اطلاعات در سمت سرور به منظور استفاده در چندین صفحه است. برخلاف کوکی ها، در زمان به کارگیری نشست ها، اطلاعات در کامپیوتر کاربر ذخیره نمی شود.

به طور پیش فرض، متغیرهای نشست ها تا زمانی که کاربر مرورگر را ببندد، باقی می مانند. برای ماندگاری دائم داده ها لازم است که داده ها در پایگاه داده ذخیره شوند. بنابراین؛ متغیرهای نشست اطلاعات مربوط به یک کاربر را در خود نگه می دارند و برای همه صفحات در یک برنامه کاربردی در دسترس هستند. یک نشست با تابع `session_start()` شروع می شود. متغیرهای نشست با آرایه سراسری `$_SESSION` ذخیره و همچنین قابل تنظیم می شوند.

لازم به ذکر است که تابع `session_start()` باید اولین چیز در برنامه باشد و قبل از تگ HTML قرار گیرد. زمانی که در یک صفحه از برنامه به متغیرهای یک نشست، مقداری تخصیص داده می شود، برای دسترسی به این مقادیر در سایر صفحات برنامه تنها کافیست در ابتدای کد آن ها نیز از تابع `session_start()` استفاده شود و نشست راه اندازی گردد. اکثر نشست ها یک کلید کاربر (PHPSESSID) در کامپیوتر کاربر تنظیم می کنند. سپس، هنگامی که یک نشست در صفحه دیگری باز می شود، کامپیوتر را برای کلید کاربر اسکن

---

<sup>1</sup> Session

می کند. اگر مطابقت وجود داشته باشد، به آن نشست دسترسی پیدا می کند، اگر نه، نشست جدیدی را شروع می کند.

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

برای اصلاح یک نشست کافی است متغیر مربوطه آن مجدداً مقداردهی شود. علاوه بر این برای حذف تمام متغیرهای یک نشست می توان از دو تابع `session_unset()` و `session_destroy()` استفاده کرد. تابع `session_unset()` برای حذف متغیرهای یک نشست به کار برده می شود، در حالی که تابع `session_destroy()` تمام نشست را به طور کامل حذف می کند.

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

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
```

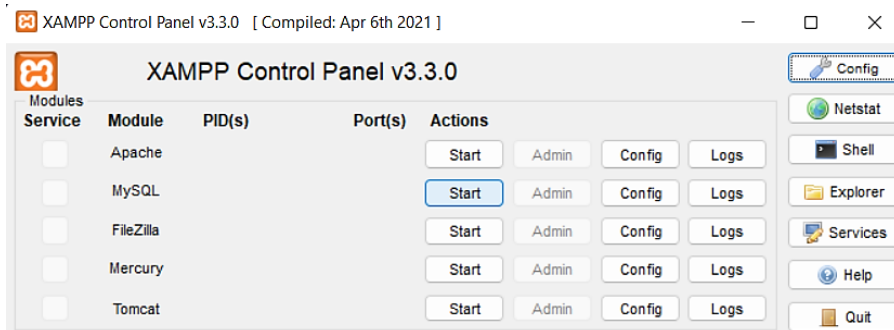
## پایگاه داده MySQL در PHP

بی شک هر وبسایتی نیاز به ذخیره داده های خود از قبیل اطلاعات کاربران، اطلاعات محصولات، فایل ها و ... در پایگاه داده دارد. این کار باعث می شود که سایت پویا شود به این صورت که هنگام طراحی سایت در بسیاری از بخش های آن نیازی نیست داده هایی مثل اخبار، اطلاعات محصولات، بلاگ ها، مقالات و ... به صورت دستی در فایل کد نوشته شوند، بلکه پس از اتصال سایت به پایگاه داده، داده ها از پایگاه داده خوانده می شوند و در سایت نمایش داده می شوند. زبان PHP قابلیت اتصال به پایگاه داده ها را دارد و MySQL محبوب ترین پایگاه داده ای است که در آن استفاده می شود. MySQL یک سیستم پایگاه داده است که مبتنی بر SQL استاندارد است، روی سرور اجرا می شود و می تواند برای برنامه های بزرگ و کوچک عملکرد خوب، سریع و امنی داشته باشد. داده ها در پایگاه داده MySQL، در جداول به صورت سطری و ستونی ذخیره می شوند و امکان طبقه بندی آن ها وجود دارد. با استفاده از دستورات SQL می توان:

- پایگاه داده ایجاد کرد
  - برنامه PHP را به پایگاه داده اتصال داد
  - در پایگاه داده جداول داده را ایجاد کرد (مثلا در یک سازمان جداول کارمندان، محصولات، مشتریان، سفارشات و ...)
  - در جدول داده درج کرد
  - داده های جدول را ویرایش/حذف کرد
  - داده های خاصی از جدول را انتخاب کرد
- به این دستورات SQL query گفته می شود که در این بخش به بررسی آن ها می پردازیم.

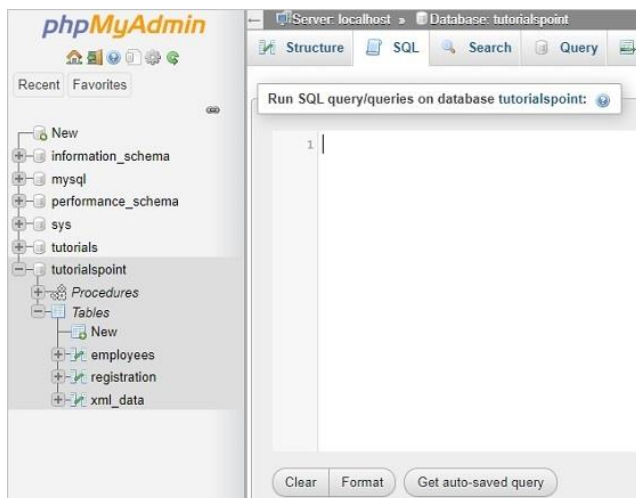
## اتصال به MySQL

سرور محلی XAMPP یک فضای گرافیکی محلی به نام phpmyadmin جهت برقراری ارتباط بین کدهای PHP و MySQL فراهم می‌کند و با دو مرحله می‌توان به آن دسترسی پیدا کرد. اول اینکه در پنل کنترل XAMPP گزینه MySQL فعال شود (شکل ۱-۱۴).



شکل ۱-۱۴: پنل کنترل XAMPP

دوم اینکه در مرورگر با استفاده از آدرس localhost/phpmyadmin به فضای گرافیکی پایگاه داده رفت. حال قادر هستیم برای کار با پایگاه داده جهت ایجاد جداول داده، درج مقدار در جداول، ویرایش یا حذف مقادیر و ... با استفاده از این ابزار گرافیکی استفاده کنیم. اما نکته حائز اهمیت این است که استفاده از ابزار گرافیکی استاندارد نیست و لازم است که دستورات SQL را داخل خود برنامه خود بنویسیم و از آن‌ها استفاده کنیم. با وجود اینکه در این ابزار گرافیکی امکان کد نویسی دستورات SQL نیز وجود دارد باز هم توصیه به استفاده از این دستورات داخل خود برنامه است. در شکل ۲-۱۴ بعد این بخش را مشاهده می‌کنید:



شکل ۲-۱۴: فضای گرافیکی پایگاه داده

قبل از اینکه بتوانیم به داده ها در پایگاه داده MySQL دسترسی داشته باشیم، باید بتوانیم به سرور متصل شویم. ایجاد این اتصال از طریق کد زیر و از طریق تابع `mysqli-connect` صورت می گیرد. این تابع نام سرور، نام کاربر و پسورد را به عنوان ورودی دریافت می کند. پیش از اینکه یک پروژه نهایی شود و روی سرور قرار بگیرد، بر روی سیستم توسعه دهنده و با سرور محلی XAMPP اجرا می شود و می توانیم نام آن را به عنوان مثال `localhost` در نظر بگیریم. بررسی شرط برقراری اتصال به دیتابیس امری حیاتی است، چنانچه اتصال برقرار نشود و قصد استفاده از پایگاه داده را داشته باشیم برنامه ما به خطا می خورد. پس لازم است از این اتصال اطمینان کسب کنیم.

```
<?php
$servername = "localhost";
$username = "root"; // XAMPP default user
$password = ""; // XAMPP default password

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

**بستن اتصال:** پس از پایان اسکریپت، اتصال به طور خودکار بسته می شود. در صورتی که بخواهیم زودتر اتصال را ببندیم (قبل از پایان اسکریپت و بسته شدن خودکار)، از دستور زیر استفاده می شود:

```
mysqli_close($conn);
```

**ایجاد پایگاه داده:** دستور CREATE DATABASE برای ایجاد پایگاه داده در MySQL

استفاده می شود. می خواهیم یک پایگاه داده با نام myDB ایجاد می کنیم:

```
// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}
```

mysqli\_query تابعی است که ازین پس توسط آن به سرور وصل می شویم و عملیات مورد نظر خود را در پایگاه داده انجام می دهیم. این تابع دو مقدار می گیرد، مقدار اول آن متغیر اتصال به سرور است که توسط تابع mysqli\_connect مقداردهی شده است (در مثال های ما متغیر \$conn می باشد) و مقدار دوم آن عملیاتی که می خواهیم در پایگاه داده انجام دهیم. این عملیات می تواند شامل ساخت جدول، درج ورودی در جدول، ویرایش

یا حذف داده ها، جستجوی داده ها و ... باشد. این عملیات که درواقع درخواستی از سمت کاربر می باشد، در SQL به عنوان query شناخته می شوند. بخش اصلی کار با پایگاه داده به کارگیری درست همین query هاست که در این بخش به بررسی انواع مختلف آن و نحوه استفاده از آن می پردازیم. یک query می تواند به شکل یک رشته ذخیره شود (در مثال ما در متغیر \$sql ذخیره می شود) و به عنوان پارامتر دوم تابع mysqli\_query ارسال شود. پس از ایجاد پایگاه داده، یکی از ملزومات این است که هنگام اتصال به سرور، نام پایگاه داده را نیز توسط تابع mysqli\_connect ارسال کنیم:

```
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
```

**ایجاد جدول:** یک پایگاه داده از یک یا چندین جدول ایجاد می شود. لذا لازم است هر جدول نام منحصر به فرد خود را داشته باشد. دستور CREATE TABLE برای ایجاد جدول در MySQL استفاده می شود. در قطعه کد زیر جدولی با نام MyGuests با چهار ستون ایجاد می کنیم و نام آن ها را "id"، "firstname"، "lastname" و "email" می گذاریم:

```
// sql to create table
$sql = "CREATE TABLE MyGuests (
  id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(30) NOT NULL,
  lastname VARCHAR(30) NOT NULL,
  email VARCHAR(50)
)";

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}
```



در این مثال "id"، "firstname"، "lastname" و "email" فیلدهای جدول ما هستند، زمانی که بخواهیم داده‌ای در این جدول درج کنیم به این معناست که برای هر داده به این چهار فیلد مقدار می‌دهیم. به عبارت دیگر هر داده در این جدول مربوط به یک فرد است که شامل یک شناسه، یک نام، یک نام خوانوادگی و یک ایمیل است. ما می‌توانیم تعیین کنیم که هر فیلد چه ویژگی‌هایی داشته باشد. مثلاً یک ویژگی این است فیلد نباید خالی بماند و هنگام درج داده باید حتماً مقدار بگیرد. برای این منظور از NOT NULL استفاده می‌شود و به این معنی است که این مقدار NULL نمی‌تواند باشد. و یا با استفاده از عبارت DEFAULT می‌توان یک مقدار پیش فرض برای فیلد تعریف کرد که در صورتیکه آن فیلد مقدار نگرفت، مقدار پیش فرض برای آن در نظر گرفته می‌شود. UNSIGNED برای نوع داده عددی استفاده می‌شود و داده‌های ذخیره شده را به اعداد مثبت و صفر محدود می‌کند. کاربرد AUTO\_INCREMENT این است که هر بار که یک رکورد جدید اضافه می‌شود به طور خودکار مقدار این فیلد افزایش پیدا می‌کند، بدیهی است که چنین فیلدی را نباید مقدار دهیم چون مقداردهی آن به صورت خودکار صورت می‌گیرد. هر جدول باید یک فیلد PRIMARY KEY داشته باشد که برای هر ردیف داده خود یک شناسه منحصر به فرد ایجاد کند، این فیلد اغلب با AUTO\_INCREMENT استفاده می‌شود.

**درج مقدار در جدول:** پس از ایجاد پایگاه داده و جداولی برای آن، می‌توان داخل جداول مقادیری درج کرد (همانطور که در بخش قبل ذکر شد درج یک مقدار در جدول یعنی مقداردهی به فیلدهای آن برای داده مورد نظر). نحوه درج یک ردیف داده به جدول به صورت زیر است:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

کافی است این عبارت را به صورت یک رشته درون یک متغیر مثل \$sql بریزیم و به عنوان query اجرا کنیم.

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');"
```

```
if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

حال اگر بخواهیم بیش از یک ردیف در یک جدول درج کنیم کافیتست مقادیر جدید را به متغیر \$sql الحاق کنیم و از تابع `mysqli_multi_query` استفاده نماییم:

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com');";
```

```
if (mysqli_multi_query($conn, $sql)) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

**انتخاب داده از جدول:** برای انتخاب داده ها از یک یا چند جدول در SQL از دستور **SELECT** استفاده می کنیم. نحوه استفاده از این دستور به صورت زیر است:

```
SELECT column_name(s) FROM table_name
```

به جای عبارت `column_name(s)` می توان نام فیلد مورد نظر را قرار دهیم. در صورتی که بخواهیم بیشتر از یک فیلد قرار دهیم می توانیم بین آن ها کاما بگذاریم و اگر هم بخواهیم تمام فیلدهای یک جدول را دریافت کنیم از \* استفاده می کنیم. مثال زیر ستون های شناسه، نام و نام خانوادگی را از جدول **MyGuests** انتخاب کرده و در صفحه نمایش می دهد.

```

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

```

ابتدا یک SQL query به صورت رشته داخل متغیر \$sql ایجاد می کنیم که ستون های شناسه، نام و نام خانوادگی را از جدول MyGuests انتخاب می کند. سپس یا استفاده از تابع mysqli\_query آن را اجرا می کنیم و مقدار خروجی را داخل متغیر \$result می ریزیم (در صورتیکه داده/داده هایی این با مشخصات خواسته شده پیدا شد در متغیر \$result ریخته می شود). در گام بعدی، تابع num\_rows() بررسی می کند که آیا بیش از صفر ردیف برگردانده شده است یا خیر (آیا داده ای برگردانده شده است یا خیر). اگر بیش از صفر ردیف برگردانده شود، تابع fetch\_assoc() همه نتایج را در یک آرایه انجمنی قرار می دهد و حلقه while() تا زمانی که داده ای برای نمایش مانده باشد اجرا می شود تا ستون های شناسه، نام و نام خانوادگی را چاپ کند.

یک ویژگی خیلی کاربردی استفاده از عبارت WHERE در دستور SELECT است. به کمک این عبارت می توانیم داده های دریافتی را فیلتر کنیم. به عنوان مثال اگر بخواهیم تنها داده هایی را دریافت کنیم که مقدار id آن ها برابر یک است داریم:

```

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE id=1";

```

قابلیت دیگری که می توان به دستور SELECT اضافه کرد، امکان مرتب سازی داده به صورت نزولی یا صعودی است که توسط عبارت ORDER BY صورت می گیرد. زمانی که از این عبارت استفاده می کنیم به طور پیش فرض داده ها را به صورت صعودی مرتب می کند. برای مرتب کردن رکوردها به ترتیب نزولی، از کلمه کلیدی DESC استفاده می

شود. نکته مهم این است که باید برای عبارت **ORDER BY** مشخص کنیم که مرتب سازی را بر اساس کدام ستون انجام دهد. مثال زیر ستون های شناسه، نام و نام خانوادگی را از جدول **MyGuests** انتخاب می کند و داده ها بر اساس ستون نام خانوادگی مرتب می شوند:

```
$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";
```

امکان محدود کردن تعداد داده های دریافت شده از دیگر قابلیت هایست که می توانیم در دستور **SELECT** استفاده کنیم. با استفاده از عبارت **LIMIT** مشخص می کنیم که دستور **SELECT** چه تعداد داده را برای ما برگرداند. **LIMIT** در جداول بزرگ بسیار مفید است. بازگرداندن تعداد زیادی داده که مورد نیاز نیستند می تواند بر عملکرد تأثیر بگذارد. فرض کنید می خواهیم ۳۰ داده ها از جدول **MyGuests** انتخاب کنیم. دستور **SQL** آن به شکل زیر خواهد بود:

```
$sql = "SELECT id, firstname, lastname FROM MyGuests LIMIT 30";
```

هنگامی که دستور بالا اجرا می شود، ۳۰ داده اول را برمی گرداند. حال اگر بخواهیم داده های ۱۶ - ۲۵ را انتخاب کنیم چه؟ راهکار **MySQL** برای این مورد استفاده از عبارت **OFFSET** است:

```
$sql = "SELECT * FROM Orders LIMIT 10 OFFSET 15";
```

**حذف داده از جدول:** دستور **DELETE** برای حذف داده ها از جدول استفاده می شود:

```
DELETE FROM table_name  
WHERE some_column = some_value
```

به این معنا که شرایط داده/دادهایی که قصد حذف آن ها را داریم در عبارت WHERE قرار می دهیم. اگر WHERE را حذف کنیم، تمام رکوردها حذف خواهند شد! ☹️  
مثالهای زیر داده با id=3 را در جدول MyGuests حذف می کند:

```
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}
```

به روزرسانی داده های جدول: دستور UPDATE برای به روز رسانی رکوردهای موجود در جدول استفاده می شود:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

عبارت WHERE مشخص می کند که کدام رکورد یا رکوردهایی باید به روز شوند. اگر WHERE را حذف کنیم، تمام رکوردها به روز می شوند! ☹️ نمونه های زیر رکورد را با id=2 در جدول MyGuests به روز می کند:

```
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}
```



## فصل پانزدهم – نکات امنیتی مهم در برنامه‌نویسی PHP

امنیت در برنامه‌نویسی PHP یکی از مهم‌ترین جنبه‌هایی است که باید مورد توجه قرار گیرد. به دلیل گستردگی کاربرد PHP در توسعه وب، برنامه‌های PHP ممکن است هدف حملات مختلف قرار گیرند. در این فصل، به بررسی نکات امنیتی مهم در برنامه‌نویسی PHP می‌پردازیم تا به شما کمک کنیم تا برنامه‌های امن‌تری بنویسید.

### اعتبارسنجی و تصدیق داده‌های ورودی

یکی از اولین مراحل در تأمین امنیت برنامه‌های PHP، اعتبارسنجی و تصدیق داده‌های ورودی است. هر داده‌ای که از کاربر دریافت می‌شود، باید به دقت بررسی و اعتبارسنجی شود.

#### • استفاده از فیلترها:

```
php
Copy code
$email = filter_input(INPUT_POST, 'email', FILTER_VALIDATE_EMAIL);
if ($email === false) {
    echo "Invalid email address.";
}
```

#### • اعتبارسنجی داده‌ها با استفاده از regex:

```
php
Copy code
if (preg_match("/^[a-zA-Z]*$/", $name)) {
    echo "Valid name.";
} else {
    echo "Invalid name.";
}
```

}

## جلوگیری از حملات SQL Injection

این حملات زمانی رخ می‌دهد که مهاجم بتواند کد SQL را به درخواست‌های پایگاه داده تزریق کند. برای جلوگیری از آن‌ها، همیشه از پرس‌وجوهای آماده استفاده کنید.

### • استفاده از PDO

```
php
Copy code
$stmt = $pdo->prepare("SELECT * FROM users WHERE email = :email");
$stmt->execute(['email' => $email]);
$user = $stmt->fetch();
```

### • استفاده از MySQLi

```
php
Copy code
$stmt = $mysqli->prepare("SELECT * FROM users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$result = $stmt->get_result();
```

## مدیریت صحیح نشست‌ها

نشست‌ها بخش مهمی از هر برنامه وب هستند و باید به درستی مدیریت شوند تا از حملات Hijacking و Fixation جلوگیری شود.

### • استفاده از شناسه‌های نشست تصادفی

```
php
Copy code
session_start();
session_regenerate_id(true);
```

### • تنظیمات امن برای کوکی‌های نشست:

```
php
Copy code
session_set_cookie_params([
    'lifetime' => 0,
    'path' => '/',
    'domain' => '',
    'secure' => true,
```



```
'httponly' => true,
'samesite' => 'Strict'
]);
```

## مدیریت صحیح خطاها

خطاهای نمایش داده شده به کاربر می‌توانند اطلاعات حساسی از ساختار داخلی برنامه شما را فاش کنند. برای مدیریت صحیح خطاها، از نمایش خطاها در محیط تولید خودداری کنید و از گزارش‌گیری خطاها استفاده کنید.

### • غیرفعال کردن نمایش خطاها:

```
php
Copy code
ini_set('display_errors', 0);
ini_set('display_startup_errors', 0);
error_reporting(0);
```

### • فعال‌سازی گزارش‌گیری خطاها:

```
php
Copy code
ini_set('log_errors', 1);
ini_set('error_log', '/path/to/error.log');
error_reporting(E_ALL);
```

## استفاده از HTTPS

برای اطمینان از امنیت انتقال داده‌ها بین مرورگر و سرور، همیشه از پروتکل HTTPS استفاده کنید. این پروتکل داده‌ها را رمزگذاری می‌کند و از شنود و حملات MITM (Man-In-The-Middle) جلوگیری می‌کند.

### • تنظیمات لازم برای استفاده از HTTPS

۱. دریافت یک گواهی SSL از یک مرجع معتبر. (CA)
۲. پیکربندی سرور وب برای استفاده از گواهی SSL.
۳. تغییر تنظیمات سایت برای اجباری کردن استفاده از HTTPS.

## محدود کردن دسترسی به فایل‌ها و پوشه‌ها

بسیاری از فایل‌ها و پوشه‌های سرور نباید قابل دسترسی از طریق وب باشند. با استفاده از فایل‌های htaccess یا پیکربندی سرور وب، دسترسی به این فایل‌ها را محدود کنید.

- مثال فایل htaccess برای محدود کردن دسترسی

```
apache
Copy code
<FilesMatch "\.(htaccess|htpasswd|ini|phar|zip|rar|tar)$">
    Require all denied
</FilesMatch>
```

## استفاده از تابع‌های امن برای فایل‌ها

هنگام کار با فایل‌ها در PHP، از تابع‌های امن استفاده کنید تا از حملات Directory Traversal و Remote File Inclusion جلوگیری کنید.

- استفاده از basename برای اطمینان از امن بودن نام فایل:

```
php
Copy code
$filename = basename($_FILES['uploaded_file']['name']);
```

- استفاده از مسیرهای مطلق:

```
php
Copy code
$filepath = '/var/www/uploads/' . $filename;
```

## به‌روزرسانی منظم نرم‌افزارها و کتابخانه‌ها

همیشه اطمینان حاصل کنید که سرور وب، PHP و تمام کتابخانه‌های مورد استفاده در برنامه شما به‌روز هستند. به‌روزرسانی منظم به شما کمک می‌کند تا از آسیب‌پذیری‌های شناخته‌شده جلوگیری کنید.



