

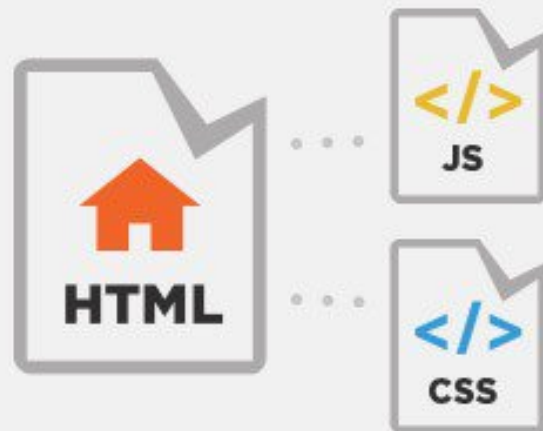
DAY1: Introduction to PHP

Getting to know the language

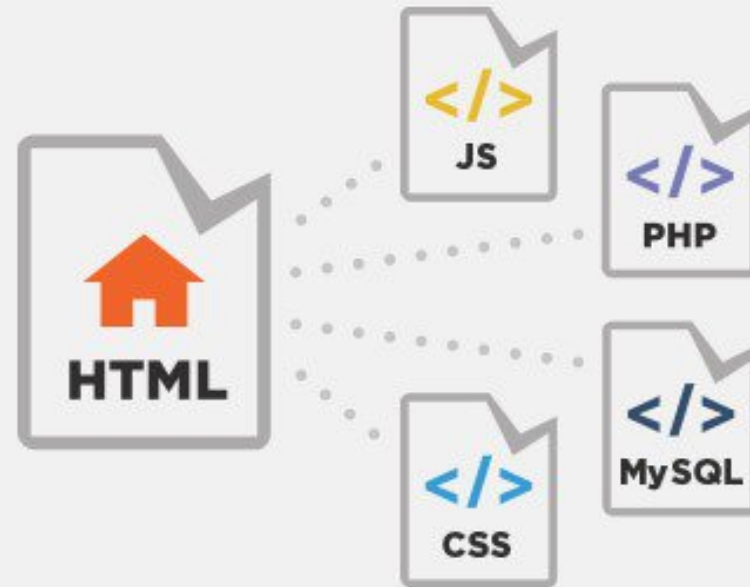


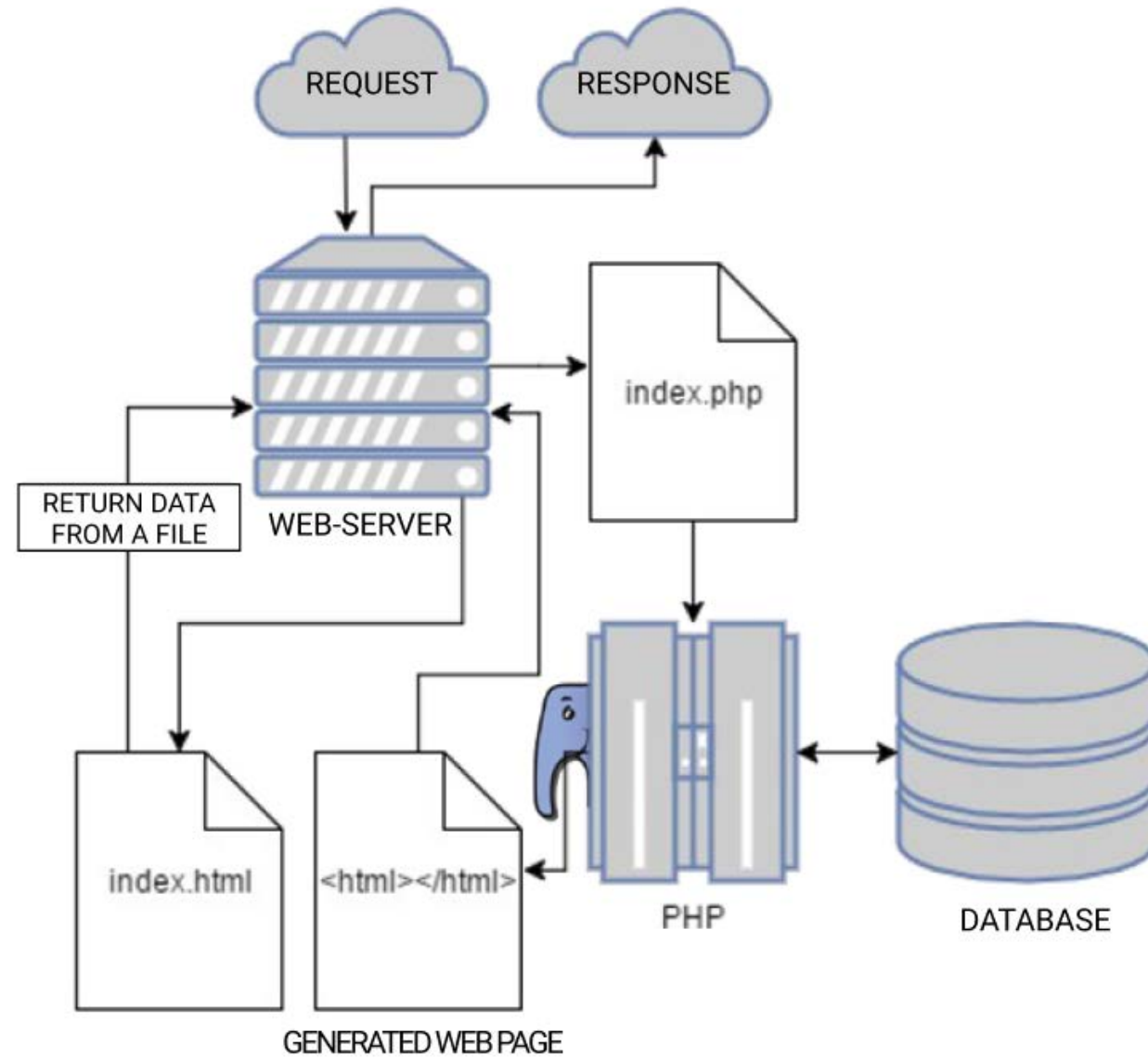
How does a dynamic
website operate?

Static Website



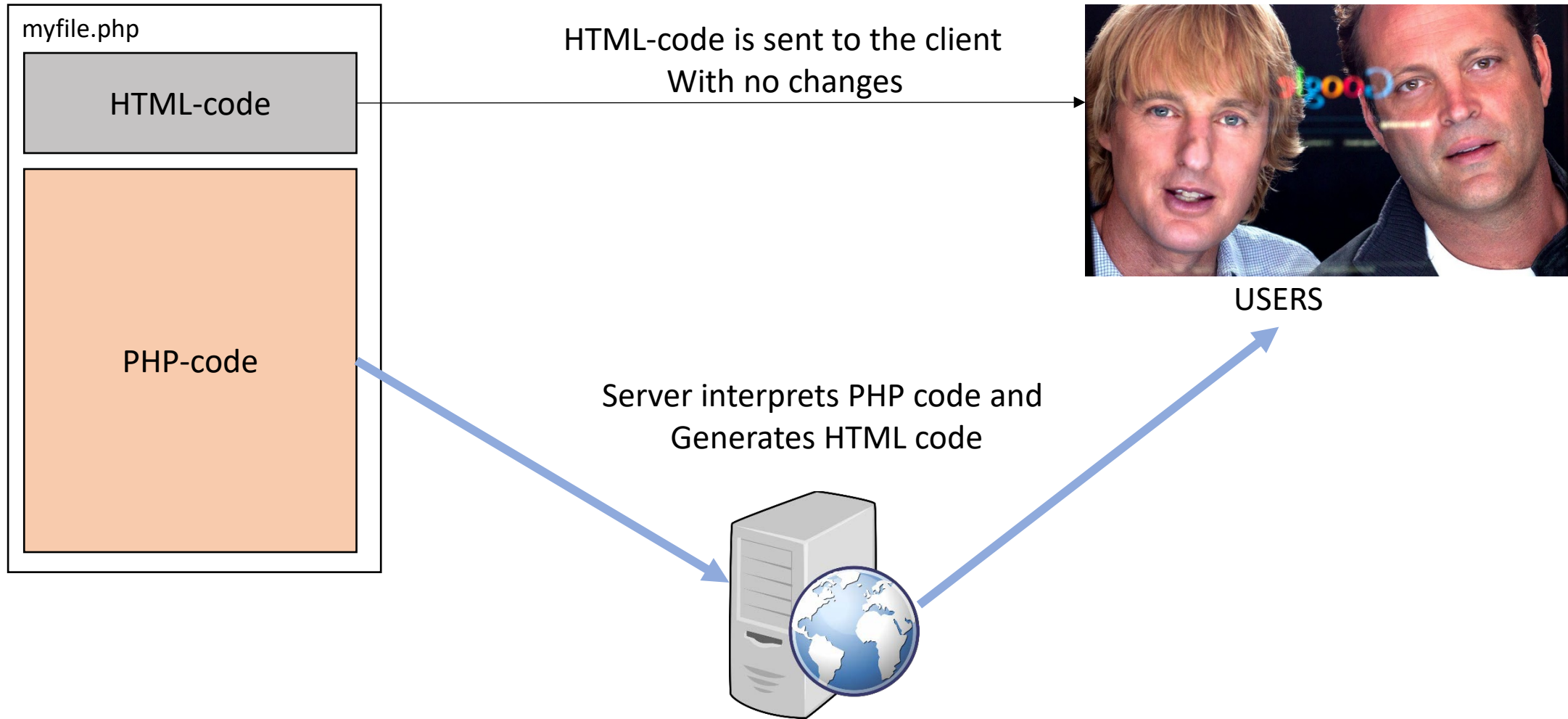
Dynamic Website







How does PHP work?



php

AJAX
Asynchronous Javascript And XML

 XAMPP

MySQL®


HTML
CSS **JS**



Apache

All these wonderful technologies...

Hello, World!

Your script:

```
1 <?php
2 echo "Hello, world!";
3 ?>
```


Variables in PHP

```
<?php  
$name = "Codefactory student";  
echo "Hello, $name!";  
?>
```



- \$variable
- \$myVariable
- \$_variable5



- \$1stvariable
- \$MYVARIABLE
- \$any%other/variablewithSymbols

Logical operators

Operand	Example	Meaning
&&	\$variable1 && \$variable2	Are both values true?
 	\$variable1 \$variable2	Is at least one value true?
AND	\$variable1 AND \$variable2	Are both values true?
XOR	\$variable1 XOR \$variable2	Is at least one value true, but NOT both?
OR	\$variable1 OR \$variable2	Is at least one value true?
!	! \$variable1	Is NOT something

Expressions

A combination of values, operators and functions as a result of which we get a new value.

Example

$$\text{\$y} = 3 * (\text{abs}(2 * \text{\$x}) + 4)$$

If-constructor: syntax

```
<?php  
    if (logical expression) operator;  
?>
```

The operator following the if-construction is carried out if the logical expression in the brackets is true

Example:

```
<?php
    $a=5;
    $b=3;
    if ($a > $b) echo "a value is larger than b value";
?>
```

A logical expression can consist of several parts

Example

```
<?php
```

```
$a=1;
```

```
$b=3;
```

```
if (isset ($a) and $a < $b and $a>0)
```

```
echo "a is smaller than b";
```

```
?>
```

If you need to perform several operations as a result of the condition being true/false, you have to place them in the curly brackets {...}

```
<?php
```

```
    if (logical expression) {
```

```
        operation1;
```

```
        operation2;
```

```
    }
```

```
?>
```

Example

```
<?php
$a=5;
$b=3;
if (isset($a, $b) and $a > $b)
{
    echo "a is larger than b";
    $b = $a;
}
?>
```


IF-ELSE

```
if (logical expression)  
    operation_1;  
else  
    operation_2;
```

Example

```
<?php
$a=5;
$b=3;
if (isset($a, $b) and $a > $b)
{
    echo "a is larger than b";
    $b = $a;
}
else echo "set a value for the variable";
?>
```

Construction elseif

```
if (logical_expression_1)
    operation_1;
elseif (logical_expression_2)
    operation_2;
else
    operation_3;
```

Example

```
<?php
$a=1;
$b="1";
if ($a > $b) {
    echo "a is larger than b";
} elseif ($a === $b) {
    echo "a equals b";
} else {
    echo "a is smaller than b";
}
?>
```

Alternative syntax:

if (logical expression):

 commands;

elseif (another logical expression):

 commands;

else:

 commands;

endif

Example

```
<?php
$a=5;
$b=2;
$c=3;
if ($a<$b):
    echo $b=$a;
elseif($b<$c):
    echo $b=$c;
else:
    echo $c=$a;
endif
?>
```

Operator ?

You pass an expression to operator ? Which it should compute, and you provide two alternative commands: one for the case when the expression is TRUE, and another for when it is FALSE.

Example

```
<?php  
echo $speed <= 5 ? "Needs fuel" : "Enough fuel";  
?>
```

switch-case

```
switch (expression) {  
    case value1: commands1; [break;]  
    case value2: commands2; [break;]  
    ...  
    case valueN: commandsN; [break;]  
[default: default_commands; [break]]  
}
```


How does switch-case work?

1. The value of the expression is calculated.
2. The switch construction is carried out in stages. When a case construct finds a value that matches the value of a switch statement, PHP executes corresponding instructions.
3. PHP continues to execute constructions until the end of the switch block until the break statement is encountered.
4. If none of the values in the set matches the value of the expression, then the default block is executed if it is specified.

Example 1

```
<?php
    $x=1;
    switch ($x) {
    case 0:
        echo "x=0<br>";
    case 1:
        echo "x=1<br>";
    case 2:
        echo "x=2<br>";
    }
?>
```

Example 2

```
<?php
    $x=0;
    switch ($x) {
    case 0:
        echo "x=0<br>"; break;
    case 1:
        echo "x=1<br>"; break;
    case 2:
        echo "x=2<br>"; break;
    }
?>
```

Example 3

```
<?php
$x="AUDI";
switch ($x) {
case "LADA":
    echo "This car is LADA"; break;
case "KIA":
    echo "This car is KIA"; break;
case "BWM":
    echo "This car is BWM"; break;
default: echo "No such car available";
}
?>
```

Loops in PHP

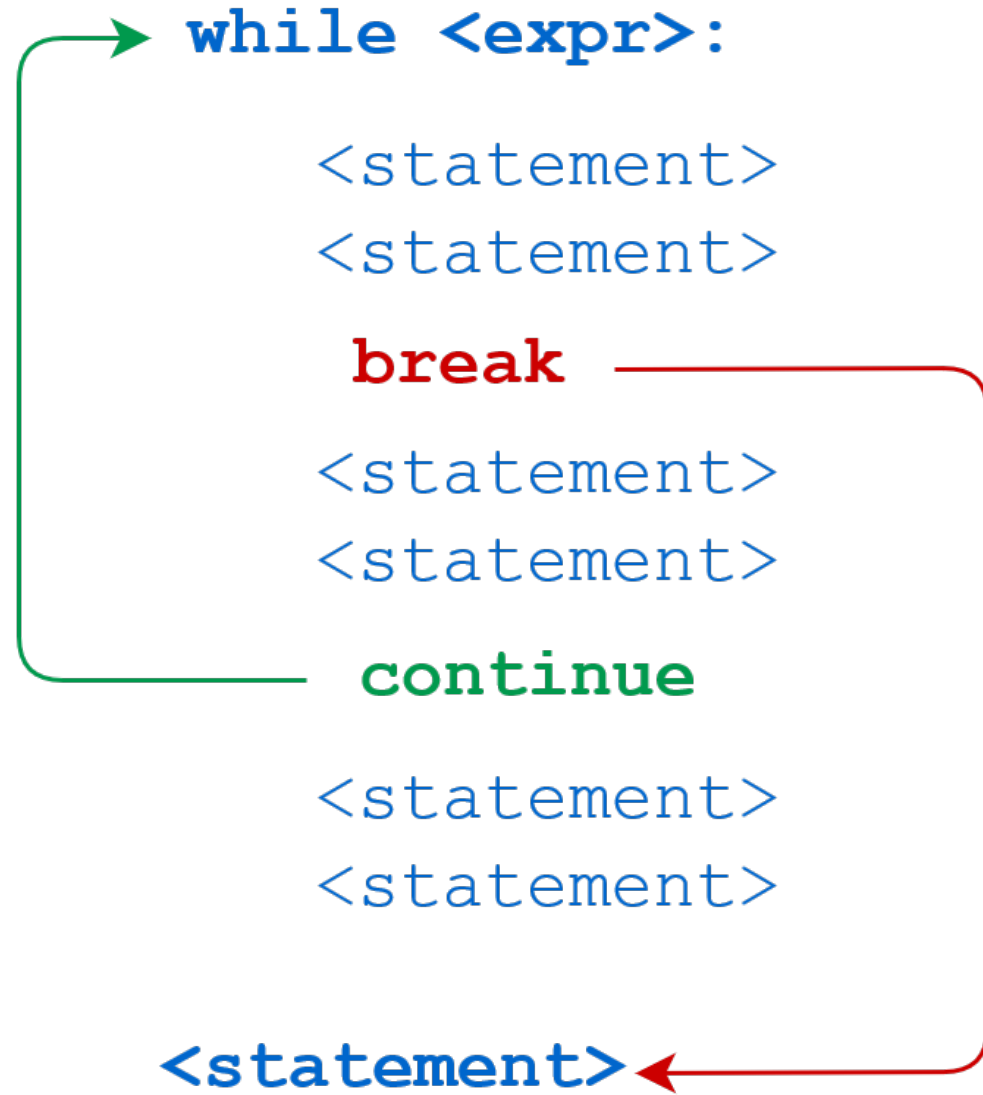
WHILE
DO-WHILE
FOR
FOREACH



You can use operators **BREAK** and **CONTINUE**

break breaks from the loop

continue stops the current iteration and goes to the beginning of the loop.



Break example

```
<?php
    $x=0;
    while ($x++<10) {
        if ($x==3) break; // when $x equals 3, the loop stops looping
        echo "Iteration $x<br>";
    }
?>
```


Continue example

```
<?php
    $x=0;
    while ($x++<5) {
        if ($x==3) continue; // The loop skips only the 3rd iteration
        echo "Iteration $x<br>";
    }
?>
```

While loop

```
<?php
    $x=0;
    while ($x++<10) echo $x;
?>
```

WHILE loop structure

Option 1:

```
while (logical_expression):  
    {  
instruction1;  
instruction2;  
    }
```

Option 2:

```
while (logical_expression):  
    instruction1;  
    ...  
endwhile;
```

Example

```
<?php
    $x=0;
    while ($x<10)
    {
        $x++;
        echo $x.'<br>';
    }
?>
```

DO-WHILE

The loop checks the condition after every iteration. The body of the loop is performed at least once.

```
do  
    {  
        Loop_body;  
    }  
while (logical_expression);
```

Example

```
<?php  
$x = 1;  
do {  
    echo $x;  
} while ($x++<10);  
?>
```

FOR-LOOP

Example1

```
<?php
    for ($x=0; $x<10; $x++) echo $x;
?>
```

Example 2

```
<?php
    for ($x=0; $x++<10;) echo $x;
?>
```

Example 3

```
<?php
    for ($x=0, $y=0; $x<10; $x++, $y++) echo $x;
?>
```

FOREACH

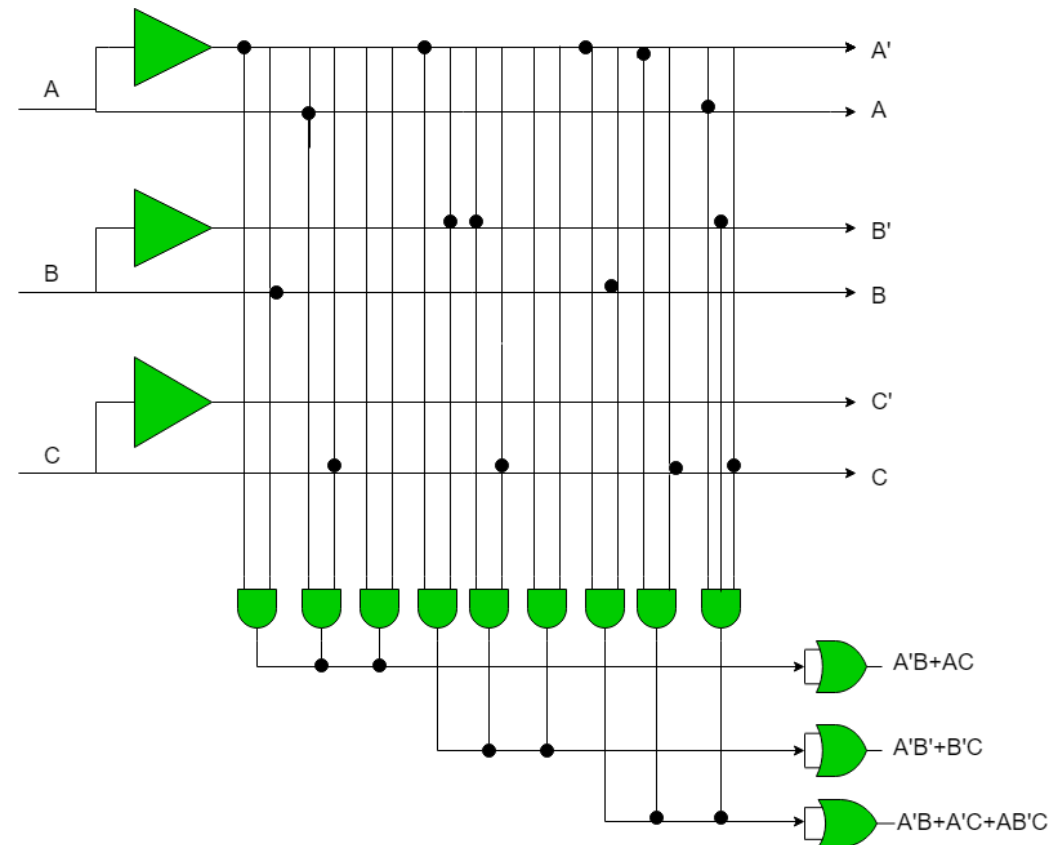
```
foreach (array_expression as $value)  
    statement
```

loops over the array given by array_expression. On each iteration, the value of the current element is assigned to \$value and the internal array pointer is advanced by one (so on the next iteration, you'll be looking at the next element).

```
foreach (array_expression as $key => $value)  
    statement
```


additionally assigns the current element's key to the \$key variable on each iteration.

ARRAYS IN PHP



NUMERIC ARRAYS

```
$movie[0] = 'Shaolin Monk';  
$movie[1] = 'Drunken Master';  
$movie[2] = 'American Ninja';  
$movie[3] = 'Once upon a time in China';  
$movie[4] = 'Replacement Killers';
```



Numeric numbers used as element
access keys

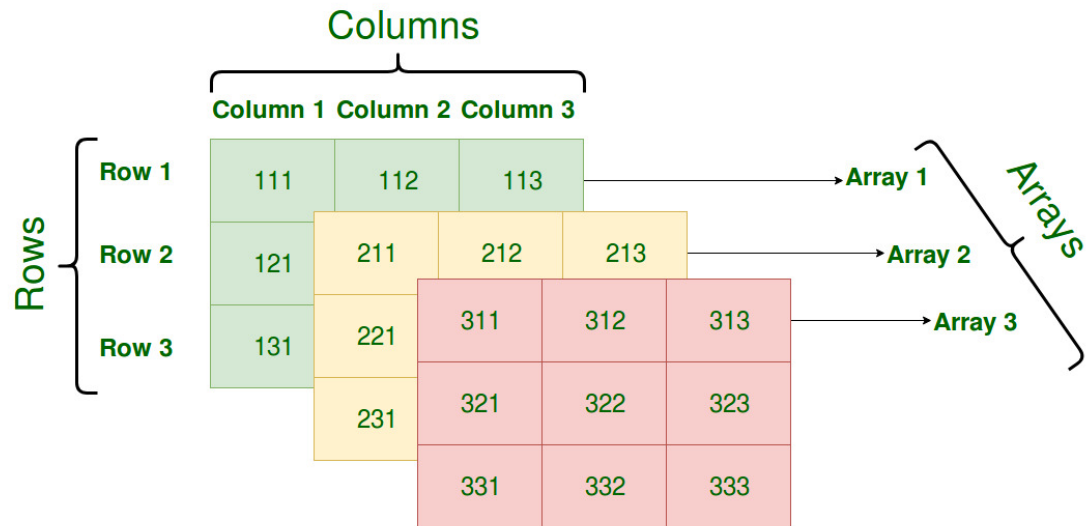
ASSOCIATIVE ARRAYS

```
$persons = array('Mary' => 'Female',  
                'John' => 'Male',  
                'Mirriam' => 'Female',  
                );
```



Descriptive captions used as array
element access key

MULTIDIMENSIONAL ARRAYS



Outer array defined as an associative array

```
$movies = array(  
    'comedy' => array('Pink Panther', 'John English'),  
    'action' => array('Die Hard', 'Expendables'),  
    'epic' => array('The Lord of the rings'),  
    'Romance' => array('Romeo and Juliet')  
);
```

Array values defined as numeric arrays

