PHP Conditional Statements

Like most programming languages, PHP also allows you to write code that perform different actions based on the results of a logical or comparative test conditions at run time. This means, you can create test conditions in the form of expressions that evaluates to either true or false and based on these results you can perform certain actions.

There are several statements in PHP that you can use to make decisions:

- The **if** statement
- The **if...else** statement
- The **if...elseif. .. else** statement
- The **switch .. case** statement

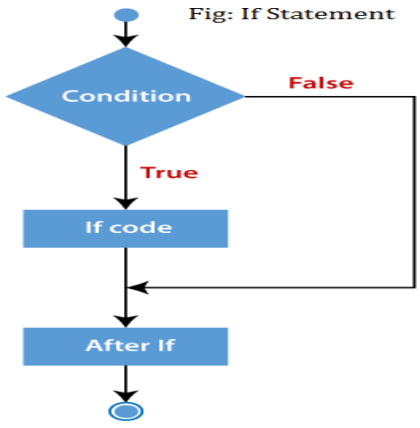
1. PHP If Statement

- PHP if statement allows conditional execution of code. It is executed if condition is true.
- If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.

Syntax

```
if(condition)
{
//code to be executed
 }
```

Flowchart



Example

```
<?php
$num=12;
if($num<100)
{
echo "$num is less than 100";
}
?>
```

Output:

```
12 is less than 100
```

PHP If-else Statement

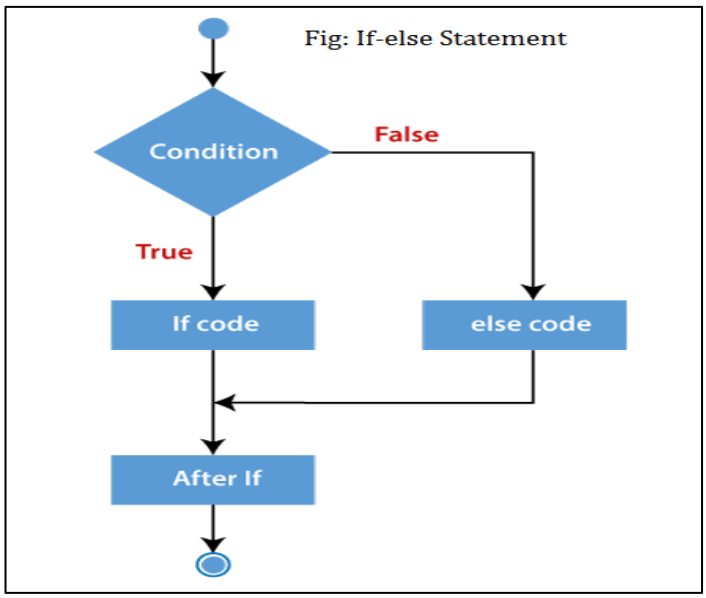
PHP if-else statement is executed whether condition is true or false.

If-else statement is slightly different from if statement. It executes one block of code if the specified condition is **true** and another block of code if the condition is **false**.

Syntax

```
if(condition)
{
//code to be executed if true
}
Else
{
//code to be executed if false
}
```

Flowchart



Example

```
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}
Else
{
echo "$num is odd number";
}
?>
```

Output:

```
12 is even number
```

PHP If-else-if Statement

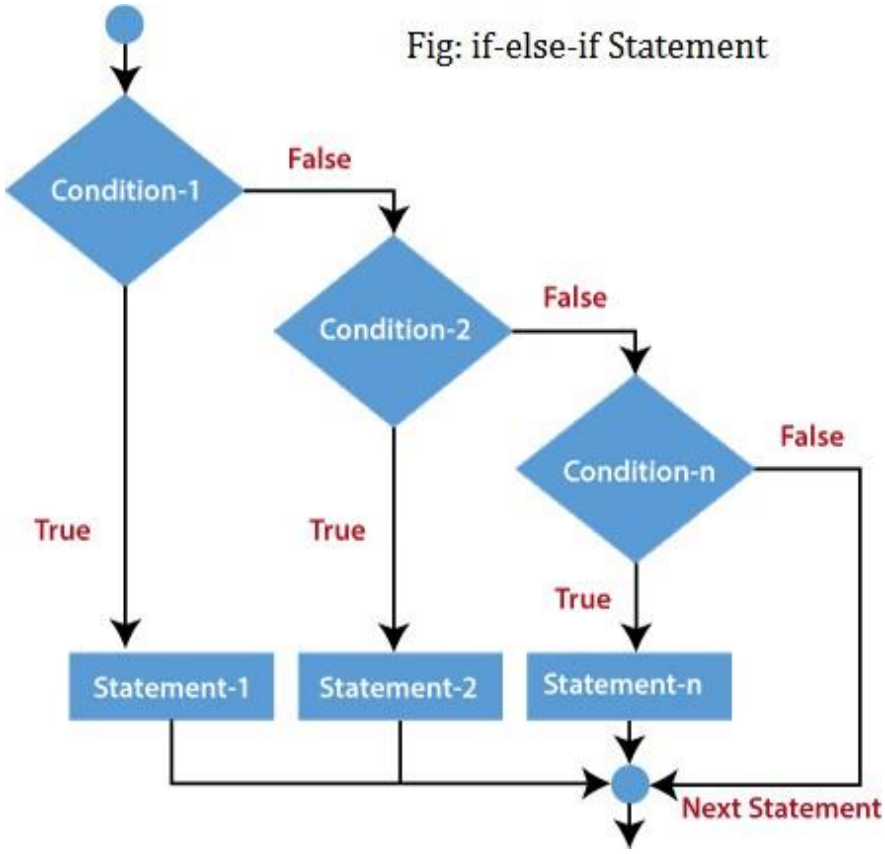
The PHP if-else-if is a special statement used to combine multiple if?.else statements. So, we can check multiple conditions using this statement.

Syntax

```
if (condition1)
{
//code to be executed if condition1 is true
}
elseif (condition2)
{
//code to be executed if condition2 is true
}
elseif (condition3)
```

```
{
//code to be executed if condition3 is true
}
Else
{
//code to be executed if all given conditions are false
}
```

Flowchart



Example

```
<?php
$marks=69;
if ($marks<33)
```

```
 {
    echo "fail";
}
else if ($marks>=34 && $marks<50)
 {
    echo "D grade";
}
else if ($marks>=50 && $marks<65)
 {
   echo "C grade";
}
else if ($marks>=65 && $marks<80)
 {
    echo "B grade";
}
else if ($marks>=80 && $marks<90)
 {
    echo "A grade";
}
else if ($marks>=90 && $marks<100)
 {
    echo "A+ grade";
}
else
```

```
    {
       echo "Invalid input";
    }
?>
```

Output:

B Grade

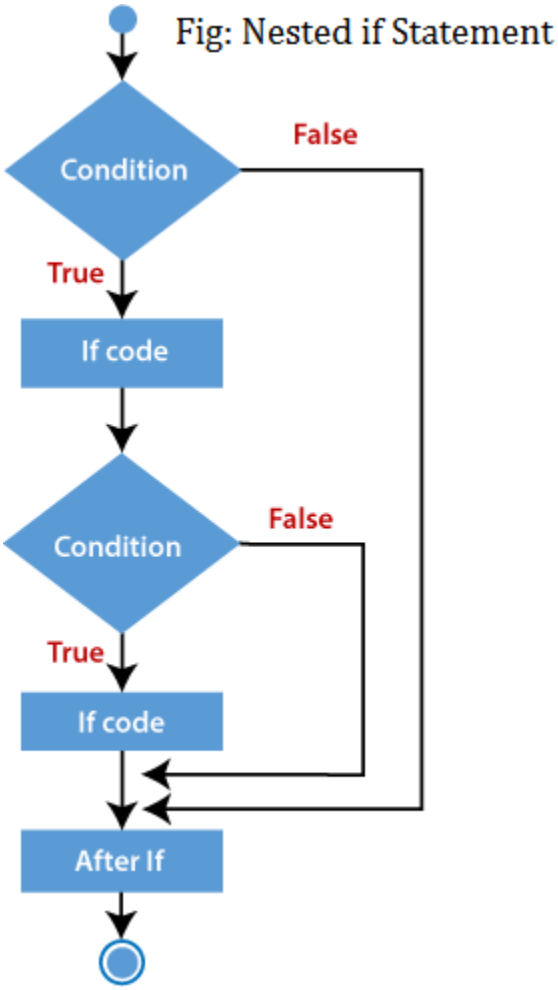
PHP nested if Statement

The nested if statement contains the if block inside another if block. The inner if statement executes only when specified condition in outer if statement is **true**.

Syntax

```
if (condition)
{
//code to be executed if condition is true
if (condition)
{
//code to be executed if condition is true
}
}
```

Flowchart



Example

```
<? php
        $age = 23;
       $nationality = "Indian";
        //applying conditions on nationality and age
        if ($nationality == "Indian")
        {
```

```
        if ($age >= 18)
        {
            echo "Eligible to give vote";
        }
        else
        {
            echo "Not eligible to give vote";
        }
    }
?>
```

Output:

```
Eligible to give vote
```

❖ PHP Switch

PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.

Syntax

```
switch(expression)
{
case value1:
//code to be executed
break;
case value2:
//code to be executed
break;
.....
default:
code to be executed if all cases are not matched;
}
```

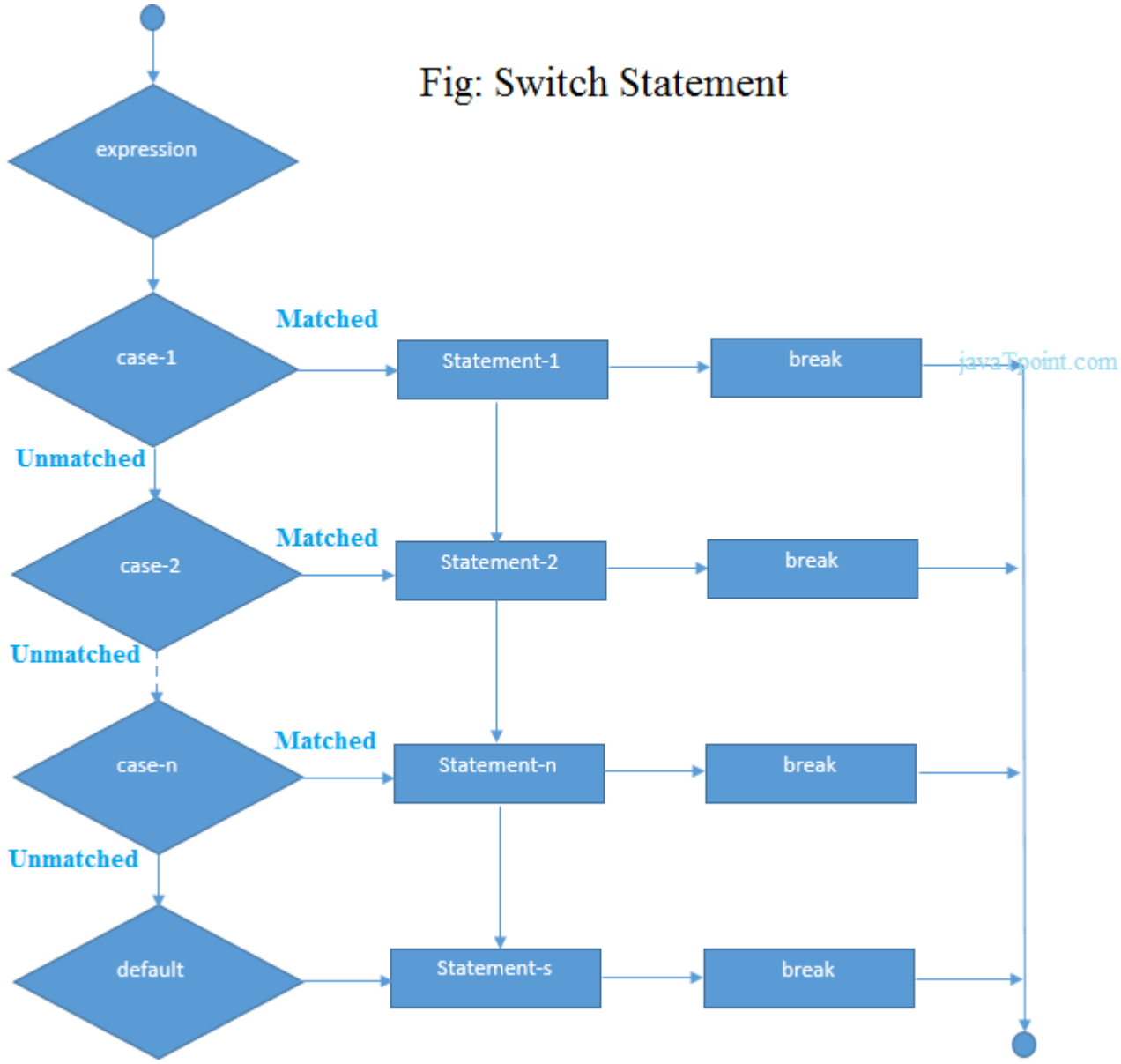
Important points to be noticed about switch case:

1. The **default** is an optional statement. Even it is not important, that default must always be the last statement.
2. There can be only one **default** in a switch statement. More than one

default may lead to a **Fatal** error.

3. Each case can have a **break** statement, which is used to terminate the sequence of statement.
4. The **break** statement is optional to use in switch. If break is not used, all the statements will execute after finding matched case value.
5. PHP allows you to use number, character, string, as well as functions in switch expression.
6. Nesting of switch statements is allowed, but it makes the program more complex and less readable.
7. You can use semicolon (;) instead of colon (:). It will not generate any error.

PHP Switch Flowchart



PHP Switch Example

```
<?php
$num=20;
switch($num){
case 10:
```

```
echo("number is equals to 10");
break;
case 20:
echo("number is equal to 20");
break;
case 30:
echo("number is equal to 30");
break;
default:
echo("number is not equal to 10, 20 or 30");
}
?>
```

Output:

```
number is equal to 20
```

PHP switch statement with String

PHP allows to pass string in switch expression. Let's see the below example of course duration by passing string in switch case statement.

```
<?php
$ch = "B.Tech";
switch ($ch)
{
   case "BCA":
```

```
        echo "BCA is 3 years course";
        break;
        case "Bsc":
        echo "Bsc is 3 years course";
        break;
        case "B.Tech":
        echo "B.Tech is 4 years course";
        break;
        case "B.Arch":
        echo "B.Arch is 5 years course";
        break;
        default:
        echo "Wrong Choice";
        break;
    }
?>
```

Output:

```
B.Tech is 4 years course
```

PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

1) The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

while (*condition is true*)

```
{
  code to be executed;
}
```

Examples

The example below displays the numbers from 1 to 5:

```
<?php
$x = 1;
```

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

Example Explained

- $x = 1; - Initialize the loop counter ($x), and set the start value to 1
- $x <= 5 - Continue the loop as long as $x is less than or equal to 5
- $x++; - Increase the loop counter value by 1 for each iteration

This example counts to 100 by tens:

Example

```
<?php
$x = 0;

while($x <= 100) {
  echo "The number is: $x <br>";
  $x+=10;
}
?>
```

Example Explained

- $x = 0; - Initialize the loop counter ($x), and set the start value to 0
- $x <= 100 - Continue the loop as long as $x is less than or equal to 100
- $x+=10; - Increase the loop counter value by 10 for each iteration

2) The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
  code to be executed;
}

while (condition is true);
```

Examples

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php
$x = 1;

do {
        echo "The number is: $x <br>";
        $x++;
    }

        while ($x <= 5);
?>
```

Note: In a do...while loop the condition is tested AFTER executing the statements within the loop. This means that the do...while loop will execute its statements at least once, even if the condition is false. See example below.

This example sets the $x variable to 6, then it runs the loop, **and then the condition is checked**:

Example

```
<? php
$x = 6;
```

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

3) The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

for (*init counter; test counter; increment counter*)

```
{
  code to be executed for each iteration;
}
```

Parameters:

- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

Examples

The example below displays the numbers from 0 to 10:

Example

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

```
{
  echo "The number is: $x <br>";
}
?>
```

Example Explained

- $x = 0; - Initialize the loop counter ($x), and set the start value to 0
- $x <= 10; - Continue the loop as long as $x is less than or equal to 10
- $x++ - Increase the loop counter value by 1 for each iteration

This example counts to 100 by tens:

Example

```
<?php
for ($x = 0; $x <= 100; $x+=10) {
  echo "The number is: $x <br>";
}
?>
```

Example Explained

- $x = 0; - Initialize the loop counter ($x), and set the start value to 0
- $x <= 100; - Continue the loop as long as $x is less than or equal to 100
- $x+=10 - Increase the loop counter value by 10 for each iteration

4) The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

foreach ($*array* as $*value*)

```
{
  code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

Examples

The following example will output the values of the given array ($colors):

Example

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

{
  echo "$value <br>";
}
?>
```

The following example will output both the keys and the values of the given array ($age):

Example

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

{
  echo "$x = $val<br>";
}
?>
```

You will learn more about arrays in the PHP Arrays chapter.

*********END********