PHP Conditional Statements

PHP 5 if...else...elseif & switch 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

In PHP we have the following conditional statements:

- **if statement** - executes some code only if a specified condition is true
- **if...else statement** - executes some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - specifies a new condition to test, if the first condition is false
- **switch statement** - selects one of many blocks of code to be executed

PHP - The if Statement

The if statement is used to execute some code **only if a specified condition is true**.

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

```
<?php
        $d = date("D");
        if($d == "Fri"){
          echo "Have a nice weekend!";
        }
?>
```

The if...else Statement

You can enhance the decision making process by providing an alternative choice through adding an *else* statement to the *if* statement. The *if...else* statement allows you to execute one block of code if the specified condition is evaluates to true and another block of code if it is evaluates to false. It can be written, like this:

```
if(condition){
   // Code to be executed if condition is true
} else{
   // Code to be executed if condition is false
}
```

```
<?php
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
} else{
    echo "Have a nice day!";
}
?>
```

The if...elseif...else Statement

The *if...elseif...else* a special statement that is used to combine multiple *if...else* statements.

```
if(condition){
  // Code to be executed if condition is true
} elseif(condition){
  // Code to be executed if condition is true
} else{
  // Code to be executed if condition is false
}
```

```
<?php
    $d = date("D");
    if($d == "Fri"){
        echo "Have a nice weekend!";
        } elseif($d == "Sun"){
        echo "Have a nice Sunday!";
        } else{
        echo "Have a nice day!";
    }
?>
```

PHP If...Else Vs Switch...Case

The switch-case statement is an alternative to the if-elseif-else statement, which does almost the same thing. The switch-case statement tests a variable against a series of values until it finds a match, and then executes the block of code corresponding to that match.

```
<?php
$today = date("D");
switch($today){
  case "Mon":
    echo "Today is Monday. Clean your house.";
    break;
  case "Tue":
    echo "Today is Tuesday. Buy some food.";
    break;
```

```
    case "Wed":
        echo "Today is Wednesday. Visit a doctor.";
        break;
    case "Thu":
        echo "Today is Thursday. Repair your car.";
        break;
    case "Fri":
        echo "Today is Friday. Party tonight.";
        break;
    case "Sat":
        echo "Today is Saturday. Its movie time.";
        break;
    case "Sun":
        echo "Today is Sunday. Do some rest.";
        break;
    default:
        echo "No information available for that day.";
        break;
}
?>
```

Different Types of Loops in PHP

Loops are used to execute the same block of code again and again, until a certain condition is met. The basic idea behind a loop is to automate the repetitive tasks within a program to save the time and effort. PHP supports four different types of loops.

- **while** – loops through a block of code until the condition is evaluate to true.
- **do…while** – the block of code executed once and then condition is evaluated. If the condition is true the statement is repeated as long as the specified condition is true.
- **for** – loops through a block of code until the counter reaches a specified number.
- **foreach** – loops through a block of code for each element in an array.

PHP while Loop

The `while` statement will loops through a block of code until the condition in the `while` statement evaluate to true.

```
while(condition){
    // Code to be executed
}
```

```
<?php
$i = 1;
while($i <= 3){
    $i++;
    echo "The number is " . $i . "<br>";
}
?>
```

PHP do...while Loop

The `do-while` loop is a variant of while loop, which evaluates the condition at the end of each loop iteration. With a `do-while` loop the block of code executed once, and then the condition is evaluated, if the condition is true, the statement is repeated as long as the specified condition evaluated to is true.

```
<?php
$i = 1;
do{
    $i++;
    echo "The number is " . $i. "<br>";
}
while($i <= 3);
?>
```

PHP for Loop

The `for` loop repeats a block of code until a certain condition is met. It is typically used to execute a block of code for certain number of times.

```
for(initialization; condition; increment){
  // Code to be executed
}
```

The parameters of `for` loop have following meaning:

- `initialization` — it is used to initialize the counter variables, and evaluated once unconditionally before the first execution of the body of the loop.
- `condition` — in the beginning of each iteration, condition is evaluated. If it evaluates to `true`, the loop continues and the nested statements are executed. If it evaluates to `false`, the execution of the loop ends.
- `increment` — it updates the loop counter with a new value. It is evaluate at the end of each iteration.

```
<?php
for($i=1; $i<=3; $i++){
    echo "The number is " . $i . "<br>";
}
?>
```

```
<?php
for ($i=1; ; $i++){
   if($i > 5){
      break;
   }
   echo "The number is " . $i . "<br />";
}
?>
```

ফর লুপ ব্যবহার করে তারার পিরামিড

```
<center>
<?php
function pyramid($hm){
   for($i=1; $i<=$hm; $i++){
      for($j=1; $j<=$i; $j++){
         echo '*';
      }
      echo '<br/>';
   }
}
pyramid(5);
?>
</center>
```

ব্যাখ্যা: $hm প্যারামিটারে যে সংখ্যা দিবেন সেই সংখ্যক তারকা শেষ সারিতে দেখাবে। ১ম লুপে (অর্থ্যাৎ $i এর লুপে) $i এখানে ১ থেকে শুরু হয়েছে এবং প্রতিবার লুপিং এর সময় ১ করে বাড়বে কারন ৩য় এক্সপ্রেশন $i++ আছে। আর লুপটি চলবে $hm বার কারন ২য় এক্সপ্রেশনে $i <= $hm দেয়া আছে। এবার ২য় লুপে ($j এর লুপটিতে) $j এর মান ১ থেকে শুরু হবে এবং $j<=$i এটার জায়গায় প্রতিবার লুপের সময় $i এর মান ওখানে ১ করে বাড়বে। সুতরাং এভাবে ১ম বার একটি * echo হবে এরপর একটা br ট্যাগ এরপরের বার ২য় ফর লুপে $i এর মান ২ হয়ে যাবে এবং দুইবার * echo হবে, ৩য় বার আবার ২য় লুপে ($j এর লুপে) $i এর মান ৩ হবে এবং ৩টি * echo হবে এবং একটি br ট্যাগ। এভাবে pyramid() এর ভিতর প্যারামিটার যে সংখ্যা দিবেন সেই সংখ্যক বার লুপ চলবে এবং * echo হতে থাকবে।

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;
}
```

```
<?php

$colors = array("red", "green", "blue", "yellow");

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

Or

```
<?php
$superhero = array(
    "name" => "Peter Parker",
    "email" => "peterparker@mail.com",
    "age" => 18
);

// Loop through superhero array
foreach($superhero as $key => $value){
    echo $key . " : " . $value . "<br>";
}
?>
```