Loops in PHP

To avoid repetition when writing code, it is often useful to loop through blocks of code. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task.

There are 4 main types of loops in PHP however **for**, **while** and **foreach** are the more common:

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

In order to demonstrate how each loop works we are going to use the following index array containing the names of week days, to echo a line stating "Today is ***weekday***" on a web page for each of the 5 days.

```
$weekDayArray  =  array("Mon", "Tue", "Wed", "Thur", "Fri", "Sat", "Sun");
```

for loop

A for loop in PHP is very similar to JavaScript. In this one a variable called **$i** is created and set equal to 0. The block of code will continue to be executed until **$i** becomes either equal to or greater than the number of elements in **$weekDayArray** which happens to be 5.

```
for( $i  =  0;  $i  <  count($weekDayArray);   $i += 1 ) {
        echo "<p>Today is " . $weekDayArray[ $i ] . "</p>";
}
```

foreach loop

A foreach loop works ***<u>exclusively for arrays</u>*** only. For every loop iteration, the value of the current array element is assigned to a new variable, in the case below this is **$value**, and the array pointer is moved by one, until it reaches the last array element. This is very handy when you have an array and want to adjust each element

```
foreach ( $weekDayArray  as  $value ) {
        echo "<p>Today is  " . $value . "</p>";
}
```

while loop

A similar premise to the for loop above however the while loop only contains a conditional statement within its parentheses. So long as this statement is true the loop will run. You can create the initialization before the block and the incrementing within the block.

While loops are often used when randomness becomes a factor. For instance keep doing something until a particular condition happens to arise!

```
$i = 0;                  // The initial variable that is incremented is created here

while( $i  <  count( $weekDayArray )) {
        echo "<p>Today is " . $weekDayArray[ $i ] . "</p>";
        $i += 1;         // It gets incremented during each loop here
}
```

do while loop

A do while loop is the same as a while loop with one main difference, the loop is guaranteed to run once even if the condition is not true. Only after the block of code in the do statement is executed the first time is the condition checked and thus the loop continues

```
$i  =  0;
do {
        // This code block is run once and then will continue if the while condition is true
        echo "<p>Today is " . $weekDayArray[ $i ] . "</p>";
        $i += 1;
}
while ( $i  <=  5 );
```

Switch Statements (*Another Type of Conditional*)

By this stage you have mastered the **if else** statements. These are conditional statements designed to run or ignore blocks of code depending on whether certain conditions are true or false.

A **switch** statement is another version of this. It is used when there are multiple blocks of code available to be selected on the same level but only one block of code must be executed. Basically it means **select one out of many blocks of code to be executed**.

Inside the switch parentheses we have **a variable that gets evaluated** or checked. The value of this is then compared to **each case** inside the block of code. If there is a match the block of code linked to that case is executed and the other cases are ignored. The **break;** text delineates the end of that block of code and prevents it from running into the next case. If there is no match then the code associated with the **default** statement is executed.

```
$nationality  =  "Canada";

switch ( $nationality ) {
        case "Canada":
                echo "You are Canadian eh!";
                break;
        case "Italy":
                echo "You are Italian. Ciao!";
                break;
        case "Japan":
                 echo "You are Japanese. Kon'nichiwa!";
                 break;
         default:
                echo "We don’t know your nationality.";
}
```