Conditional logic

Introduction to conditional logic

In the previous lesson, you learned that a variable is used to store information. The reason you are storing this information is that you can do something with them. For example, if you have registered a user name in a variable, check if it is a valid user name.
For that, you will use something called "*conditional logic*".

The conditional logic consists of asking "What is happening **IF** ...".
When you see a big button with an inscription "Do not press this button! You will use conditional logic. Wondering "What's happening **IF** I push the button?"

Most of the time, you use conditional logic to test what's inside a variable.

Using IF

Let's take as an example a list of movies.
We're trying to find out if the current movie is "Pirates of the Caribbean":

```
if ($ movie == "pirate caraibes") {
        // Code if it's the good movie
}
```

If it's not the good movie, we can tell him to run another code:

```
if ($ movie == "caraibes pirate") {
                // Code if it's the right movie
                }
else {
   // Code if it's not the right movie
}
```

What we say here is "If the previous condition is not true, then we do that".

It is also possible to test several values:

```
if ($ movie == "pirate des caraibes") {
   // Code if it's the pirate movie
}
else if ($ movie == "star wars") {
                // Code if it's the movie star wars
}
else {
   // Code if we do not know the movie
}
```

What we say here is "IF the previous condition is not true, then try this one, and otherwise we do that".

Comparison Operators

So far, you have used the equal double sign **==** to test whether the variable was the same as text.
The equal double sign **==** is a comparison operator.

<u>There are several comparison operators. Here is the list:</u>

== - Contains the same value as
```
        if ($ var1 == $ var2) {
   }
```

=== - Same as AND AND
```
   if ($ var1 === $ var2) {
   }
```

! = - N ' it's not the same value as
```
   if ($ var1! = $ var2) {
   }
```

! == - Is different or not of the same type.
```
   if ($ var1! == $ var2) {
   }
```

< - Smaller than
```
   if ($ var1 <$ var2) {
   }
```

> - Larger than
```
   if ($ var1> $ var2) {
   }
```

<= - Smaller or equal
```
   if ($ var1 <= $ var2) {
   }
```

> = - Greater or equal
```
        if ($ var1> = $ var2) {
        }
```

Using Switch

If you have only one variable to test, there is a better alternative to IF: Switch.

Example with our movie list:

```
<? Php
   $ movies = 'hacker';

   switch ($ movies) {
      case 'pirate':
         echo 'The pirate film of the Caribbean';
         break;

      case 'starwars':
         echo 'The movie star wars';
         break;

       default:
               echo 'We do not know the movie';
               break;
   }
?>
```

- **"Switch (...) {...}"** is the general block
- **"case value: ..."** are the blocks of code that test the value of the condition
- **"default:"**is the code block executed when no other value matches the condition.

You must tell PHP "*break*" to get out of the switch statement. If you do not do it, PHP will just go to the next case and check the rest.
Use the word 'break' to exit the Switch statement.

Logical operators Logical

operators are used when you want to test multiple conditions at once. For example, you can check if a username AND password are correct in the same IF statement.

<u>There are several logical operators. Here is the list:</u>

&& - AND
Use AND or && if you want both values to be true

```
   if ($ user_name && $ password) {
   }
```

|| - OR
Use OR or || if you want at least one of the two values to be true

```
   if ($ user_name || $ password) {
   }
```

XOR
Use XOR if you want at least one of the two values to be true but not both

```
   if ($ user_name XOR $ password) {
   }
```

!
! is the inverse of a result

```
   if (! $ user_name_valid) {
   }
```

Booleans

We had already spoken about this during the first class.
A boolean is a variable that can only be in two states: True (1) or False (0).

You can test the value of a boolean like this:

```
<? Php
   if ($ bool == 1) {
      echo "True!";
   }
?>
```

Here, we will test if our boolean is True (value: 1).
However, there is an easier way to test if the value of a Boolean is True:

```
<? php
   if ($ bool) {
      echo "True!";
   }
?>
```

The two syntaxes mean the same thing.