PHP If Statements

```php
if (TRUE){
  echo "TRUE is always true";
}

$condition1 = TRUE;
if ($condition1) {
  // This code block will execute
}

$condition2 = FALSE;
if ($condition2) {
  // This code block will not execute
}
```

PHP `if` statements evaluate a boolean value or expression and execute the provided code block if the expression evaluates to `TRUE`.

PHP Truthy and Falsy

```
if ("What's going on?"){   // evaluates to TRUE
  echo "Let us explain…";
}
// Prints: Let us explain…
```

PHP values within a condition will always be evaluated to `TRUE` or `FALSE`. Values that will evaluate to `TRUE` are known as *truthy* and values that evaluate to `FALSE` are known as *falsy*.

Falsy values include:

- `false`
- `0`
- empty strings
- `null`
- `undefined`
- `NaN`.

All other values are *truthy*.

PHP else statement

```
$condition = FALSE;
if ($condition) {
  // This code block will not execute
} else {
  // This code block will execute
}
```

A PHP else statement can follow an `if` block. If the condition of the `if` does not evaluate to `TRUE`, the code block following `else` will be executed.

PHP Boolean Values

```php
// booleans
$is_true = TRUE;
$is_false = FALSE;

echo gettype($is_true);
// Prints: boolean
echo gettype($is_false);
// Prints: boolean
```

PHP Boolean values are either `TRUE` or `FALSE`, which are the only members of the `boolean` type

PHP ternary operator

```php
// Without ternary
$isClicked = FALSE;
if ($isClicked) {
  $link_color = "purple";
} else {
  $link_color = "blue";
}
// $link_color = "blue";


// With ternary
$isClicked = FALSE;
$link_color = $isClicked ? "purple" : "blue";
//  $link_color = "blue";
```

In PHP, the ternary operator allows for a compact syntax in the case of binary (`if/else`) decisions. It evaluates a single condition and executes one expression and returns its value if the condition is met and the second expression otherwise.

The syntax for the ternary operator looks like the following:

```
condition ? expression1 : expression2
```

PHP comparison operators

```php
// Comparison operators
1 > 3; // FALSE
3 > 1; // TRUE
250 >= 250; // TRUE
1 === 1; // TRUE
1 === 2; // FALSE
1 === "1"; // FALSE
```

PHP *comparison operators* are used to compare two values and return `TRUE` or `FALSE` depending on the validity of the comparison. Comparison operators include:

- identical (`===`)
- not identical (`!==`)
- greater than (`>`)
- less than (`<`)
- greater than or equal (`>=`)
- less than or equal (`<=`)

PHP elseif statements

```php
$fav_fruit = "orange";

if ($fav_fruit === "banana"){
  echo "Enjoy the banana!";
} elseif ($fav_fruit === "apple"){
  echo "Enjoy the apple!";
} elseif ($fav_fruit === "orange"){
  echo "Enjoy the orange!";
} else {
  echo "Enjoy the fruit!";
}
// Prints: Enjoy the orange!
```

PHP `elseif` statements must be paired with an `if` statement, but many `elseif`s can be chained from a single `if`.

`elseif`s provide an additional condition to check (and corresponding code to execute) if the conditional statements of the `if` block and any preceding `elseif`s are not met.

PHP switch statement

```
switch ($letter_grade){
  case "A":
    echo "Terrific";
    break;
  case "B":
    echo "Good";
    break;
  case "C":
    echo "Fair";
    break;
  case "D":
    echo "Needs Improvement";
    break;
  case "F":
    echo "See me!";
    break;
  default:
    echo "Invalid grade";
}
```

PHP `switch` statements provide a clear syntax for a series of comparisons in which a value or expression is compared to many possible matches and code blocks are executed based on the matching `case`.

In PHP, once a matched `case` is encountered, the code blocks of all subsequent cases (regardless of match) will be executed until a `return`, `break`, or the end of the statement is reached. This is known as *fall through*.

PHP readline()

```
echo "\nWhat's your name?\n";
$name = readline(">> "); // receives user input
```

The PHP built-in `readline()` function takes a string with which to prompt the user. It waits for the user to enter text into the terminal and returns that value as a string.