**PHP Notes**

**Introduction**

* PHP (Hypertext Preprocessor) is a widely used scripting language primarily designed for web development. It is often embedded within HTML code to create dynamic and interactive web pages. PHP scripts are executed on the server side, meaning that they run on a web server before the resulting content is sent to the user's web browser. This allows PHP to generate dynamic content, interact with databases, handle forms, and perform various other server-side tasks.
* Why PHP?
  + PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
  + PHP is compatible with almost all servers used today (Apache, IIS, etc.)
  + PHP supports a wide range of databases
  + PHP is free. Download it from the official PHP resource: www.php.net
  + PHP is easy to learn and runs efficiently on the server side

**PHP Files**

* PHP files can contain text, HTML, CSS, JavaScript, and PHP code
* PHP code is executed on the server, and the result is returned to the browser as plain HTML
* PHP files have extension ".php"
* With PHP you are not limited to output HTML. You can output images or PDF files. You can also output any text, such as XHTML and XML.

**Setup**

* <https://www.youtube.com/watch?v=zZ6vybT1HQs&t=13337s>

**PHP Syntax**

* A PHP script is executed on the server, and the plain HTML result is sent back to the browser.
* A PHP script can be placed anywhere in the document.
* A PHP script starts with <?php and ends with ?>:
* A black text on a white background

  Description automatically generated
* PHP statements end with a semicolon (;).
* In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.
* In the example below, all three echo statements below are equal and legal:
* A screenshot of a computer code

  Description automatically generated
* However; all variable names are case-sensitive!
* Ex:
  + A computer code with black text

    Description automatically generated  
    Output:
  + A computer code with black text

    Description automatically generated
* In PHP, whitespace (spaces, tabs, and newlines) doesn't affect the functionality of the code. This means you can use whitespace to format your code in a readable and organized way without affecting how the code runs.
* However, an where whitespace can have an impact is with string outputs. Whitespace within strings is preserved when the string is output, so it will appear as intended.
* 

**Comments**

* Single Line
* A screenshot of a computer code

  Description automatically generated
* Multi-Line
* A screenshot of a computer code

  Description automatically generated
* Using comments to leave out parts of the code:
* A white background with text

  Description automatically generated

**Variables**

* In PHP, a variable starts with the $ sign, followed by the name of the variable:
* A white background with red and black text

  Description automatically generated
* After the execution of the statements above, the variable $txt will hold the value Hello world!, the variable $x will hold the value 5, and the variable $y will hold the value 10.5.
* Note: When you assign a text value to a variable, put quotes around the value.
* Note: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.
* A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume).
* Rules for PHP variables:
  + A variable starts with the $ sign, followed by the name of the variable
  + A variable name must start with a letter or the underscore character
  + A variable name cannot start with a number
  + A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
  + Variable names are case-sensitive ($age and $AGE are two different variables)
* We do not have to tell PHP which data type the variable is.
* PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.
* The PHP echo statement is often used to output data to the screen.
* The following example will show how to output text and a variable:
* Ex:
  + A close-up of a website

    Description automatically generated
  + Output: 
* Ex:
  + A math equations on a white background

    Description automatically generated
  + Output: 

**Variables Scope**

* PHP Variables Scope
* In PHP, variables can be declared anywhere in the script.
* The scope of a variable is the part of the script where the variable can be referenced/used.
* PHP has three different variable scopes:
  + local
  + global
  + static
* A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:
* Ex:
  + Variable with global scope:
  + A screenshot of a computer code

    Description automatically generated
* A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function:
* Ex:
  + A white background with text

    Description automatically generated
* You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
* The global keyword is used to access a global variable from within a function.
* To do this, use the global keyword before the variables (inside the function):
* Ex:
  + A white background with black text and symbols

    Description automatically generated
* PHP also stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.
* The example above can be rewritten like this:
  + A white background with black and orange text

    Description automatically generated
* Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.
* To do this, use the static keyword when you first declare the variable:
* Ex:
  + A white background with black text and numbers

    Description automatically generated
  + Output: 
  + Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.
  + Note: The variable is still local to the function.

**PHP echo and print Statements**

* With PHP, there are two basic ways to get output: echo and print.
* In this tutorial we use echo or print in almost every example. So, this chapter contains a little more info about those two output statements.
* echo and print are more or less the same. They are both used to output data to the screen.
* The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.
* The PHP echo Statement
  + The echo statement can be used with or without parentheses: echo or echo().
  + The following example shows how to output text with the echo command (notice that the text can contain HTML markup):
  + A white background with black text

    Description automatically generated
  + Output: A white background with black text

    Description automatically generated
  + The following example shows how to output text and variables with the echo statement:
  + A white background with black text

    Description automatically generated
  + Output: A black text on a white background

    Description automatically generated
* The PHP print Statement
  + The print statement can be used with or without parentheses: print or print().
  + Display Text
  + The following example shows how to output text with the print command (notice that the text can contain HTML markup):
  + A white background with red text

    Description automatically generated
  + The following example shows how to output text and variables with the print statement:
  + A white background with black text

    Description automatically generated

**PHP Data Types**

* Variables can store data of different types, and different data types can do different things.
* PHP supports the following data types:
  + String
  + Integer
  + Float (floating point numbers - also called double)
  + Boolean
  + Array
  + Object
  + NULL
* String
  + A string is a sequence of characters, like "Hello world!".
  + A string can be any text inside quotes. You can use single or double quotes:
  + A screen shot of a computer

    Description automatically generated
* Integer
  + An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
  + Rules for integers:
    - An integer must have at least one digit
    - An integer must not have a decimal point
    - An integer can be either positive or negative
    - Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation
  + In the following example $x is an integer. The PHP var\_dump() function returns the data type and value:
  + A close-up of a math equation

    Description automatically generated
  + Output: 
* Float
  + A float (floating point number) is a number with a decimal point or a number in exponential form.
  + In the following example $x is a float. The PHP var\_dump() function returns the data type and value:
  + A black text with red letters

    Description automatically generated with medium confidence
  + Output:
* Boolean
  + A Boolean represents two possible states: TRUE or FALSE.
  + A black and white text

    Description automatically generated
* Array
  + An array stores multiple values in one single variable.
  + In the following example $cars is an array. The PHP var\_dump() function returns the data type and value:
  + A text on a white background

    Description automatically generated
  + Output: A number of numbers and symbols

    Description automatically generated with medium confidence
* Object
  + Classes and objects are the two main aspects of object-oriented programming.
  + A class is a template for objects, and an object is an instance of a class.
  + When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.
  + Let's assume we have a class named Car. A Car can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties.
  + When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.
  + If you create a \_\_construct() function, PHP will automatically call this function when you create an object from a class.
  + A computer code with text

    Description automatically generated
  + Output: A close up of words

    Description automatically generated
* NULL
  + Null is a special data type which can have only one value: NULL.
  + A variable of data type NULL is a variable that has no value assigned to it.
  + Tip: If a variable is created without a value, it is automatically assigned a value of NULL.
  + Variables can also be emptied by setting the value to NULL:
  + A white background with black text

    Description automatically generated

**Strings**

* The PHP strlen() function returns the length of a string.
* A close up of a text

  Description automatically generated
* The PHP str\_word\_count() function counts the number of words in a string.
* A close up of a text

  Description automatically generated
* The PHP strrev() function reverses a string.
* A close up of text

  Description automatically generated
* The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position (0-index) of the first match. If no match is found, it will return FALSE.
* A black and red text

  Description automatically generated
* The PHP str\_replace() function replaces some characters with some other characters in a string.
* Replace the text "world" with "Dolly":
* A close up of a person's face

  Description automatically generated

**Numbers**

* One thing to notice about PHP is that it provides automatic data type conversion.
* So, if you assign an integer value to a variable, the type of that variable will automatically be an integer. Then, if you assign a string to the same variable, the type will change to a string.
* This automatic conversion can sometimes break your code.
* Integers
  + 2, 256, -256, 10358, -179567 are all integers.
  + An integer is a number without any decimal part.
  + An integer data type is a non-decimal number between -2147483648 and 2147483647 in 32 bit systems, and between -9223372036854775808 and 9223372036854775807 in 64 bit systems. A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer.
  + Note: Another important thing to know is that even if 4 \* 2.5 is 10, the result is stored as float, because one of the operands is a float (2.5).
  + Here are some rules for integers:
    - An integer must have at least one digit
    - An integer must NOT have a decimal point
    - An integer can be either positive or negative
    - Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)
  + PHP has the following predefined constants for integers:
    - PHP\_INT\_MAX - The largest integer supported
    - PHP\_INT\_MIN - The smallest integer supported
    - PHP\_INT\_SIZE - The size of an integer in bytes
  + PHP has the following functions to check if the type of a variable is integer:
    - is\_int()
    - is\_integer() - alias of is\_int()
    - is\_long() - alias of is\_int()
    - Ex:
    - A white background with black text

      Description automatically generated
    - Output: A close up of words

      Description automatically generated
* Floats
  + A float is a number with a decimal point or a number in exponential form.
  + 2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 are all floats.
  + The float data type can commonly store a value up to 1.7976931348623E+308 (platform dependent), and have a maximum precision of 14 digits.
  + PHP has the following predefined constants for floats (from PHP 7.2):
    - PHP\_FLOAT\_MAX - The largest representable floating point number
    - PHP\_FLOAT\_MIN - The smallest representable positive floating point number
    - PHP\_FLOAT\_DIG - The number of decimal digits that can be rounded into a float and back without precision loss
    - PHP\_FLOAT\_EPSILON - The smallest representable positive number x, so that x + 1.0 != 1.0
  + PHP has the following functions to check if the type of a variable is float:
    - is\_float()
    - is\_double() - alias of is\_float()
  + Ex:
    - Check if a variable is a float
    - A black text with red letters

      Description automatically generated
    - 
* Infinity
  + A numeric value that is larger than PHP\_FLOAT\_MAX is considered infinite.
  + PHP has the following functions to check if a numeric value is finite or infinite:
    - is\_finite()
    - is\_infinite()
  + However, the PHP var\_dump() function returns the data type and value:
  + A math equations and symbols

    Description automatically generated with medium confidence
  + Output: 
* NaN
  + NaN stands for Not a Number.
  + NaN is used for impossible mathematical operations.
  + PHP has the following functions to check if a value is not a number:
    - is\_nan()
  + However, the PHP var\_dump() function returns the data type and value:
  + Ex:
  + A math equations on a white background

    Description automatically generated
  + Output: 
* Numerical Strings
  + The PHP is\_numeric() function can be used to find whether a variable is numeric. The function returns true if the variable is a number or a numeric string, false otherwise.
  + Ex:
  + A screenshot of a computer code

    Description automatically generated
  + Output: A group of black text

    Description automatically generated
* Casting Strings and Floats to Integers
  + Sometimes you need to cast a numerical value into another data type.
  + The (int), (integer), or intval() function are often used to convert a value to an integer.
  + A screenshot of a computer code

    Description automatically generated
  + Output: A number on a white background

    Description automatically generated
  + Keep in mind that when casting from a floating-point number to an integer, the decimal portion will be truncated. The integer cast effectively rounds towards zero.
  + Keep in mind that when casting non-numeric strings, PHP will attempt to extract the leading numeric part of the string. If the string doesn't start with a numeric value, the result will be 0. Also, boolean true will be cast to 1 and boolean false will be cast to 0.

**Mutable vs Immutable Datatypes**

* PHP, objects are the primary data type that exhibits the behavior where modifications to one instance are reflected in another instance that refers to the same object. This behavior is due to object assignment by reference.
* Other data types like strings, integers, floats, booleans, and arrays are assigned by value. When you assign these types to another variable, you are creating a new copy of the value, and changes to one variable will not affect the other variable.
* Here's a quick recap:
  + Objects: Object instances are assigned by reference. If you assign an object to another variable, both variables will reference the same object instance, and changes made through one variable will affect the other variable.
  + Strings, Integers, Floats, Booleans: These primitive data types are assigned by value. When you assign one of these types to another variable, you create a copy of the value, and changes to one variable will not affect the other variable.
  + Arrays: Arrays in PHP are assigned by value. However, arrays are a bit special because they can contain a mixture of values (including objects). If an array contains object references, modifying the object through one variable will affect the array and, consequently, other variables that reference the same object.

**Math**

* A computer code with numbers and symbols

  Description automatically generated

**Constants**

* Constants are like variables except that once they are defined they cannot be changed or undefined.
* A valid constant name starts with a letter or underscore (no $ sign before the constant name).
* Note: Unlike variables, constants are automatically global across the entire script.
* To create a constant, use the define() function.
* define(name, value, case-insensitive)
  + name: Specifies the name of the constant
  + value: Specifies the value of the constant
  + case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false
* Ex:
  + Create a constant with a case-sensitive name:
  + A white background with black text

    Description automatically generated
* Ex:
  + Create a constant with a case-insensitive name:
  + A white background with black text

    Description automatically generated
* In PHP7, you can create an Array constant using the define() function.
* A screenshot of a computer code

  Description automatically generated
* Constants are automatically global and can be used across the entire script.
* Ex:
  + This example uses a constant inside a function, even if it is defined outside the function:
  + A white screen with black text

    Description automatically generated

**Operators**

* Operators are used to perform operations on variables and values.
* PHP divides the operators in the following groups:
  + Arithmetic operators
  + Assignment operators
  + Comparison operators
  + Increment/Decrement operators
  + Logical operators
  + String operators
  + Array operators
  + Conditional assignment operators
* Arithmetic Operators
  + A table with text and symbols

    Description automatically generated
* Assignment Operator
  + A screenshot of a math program

    Description automatically generated
* Comparison Operators
  + A table of math equations

    Description automatically generated
* Increment / Decrement Operators
  + A screenshot of a computer screen

    Description automatically generated
* Logical Operators
  + A screenshot of a computer

    Description automatically generated
* String Operators
  + A white background with black text

    Description automatically generated
* Array Operators
  + A screenshot of a math equation

    Description automatically generated
* Conditional Assignment Operators
  + A screenshot of a computer

    Description automatically generated

**Condition Statements**

* Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.
* In PHP we have the following conditional statements:
  + if statement - executes some code if one condition is true
  + if...else statement - executes some code if a condition is true and another code if that condition is false
  + if...elseif...else statement - executes different codes for more than two conditions
  + switch statement - selects one of many blocks of code to be executed
* The if Statement
  + A white background with black text

    Description automatically generated
* The if...else Statement
  + A computer code with text

    Description automatically generated with medium confidence
* The if...elseif...else Statement
  + A screenshot of a computer code

    Description automatically generated
* The switch Statement
  + A screenshot of a computer code

    Description automatically generated
  + A screenshot of a computer code

    Description automatically generated

**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
* while Loop
  + A white background with black text

    Description automatically generated
* do while Loop
  + The do...while loop - Loops through a block of code once, and then repeats the loop as long as the specified condition is true.
  + A white background with black text

    Description automatically generated
  + 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.
* for Loop
  + A number equation with numbers and symbols

    Description automatically generated with medium confidence
* foreach Loop
  + A white background with black text

    Description automatically generated
  + A white background with black text

    Description automatically generated
* break
  + A white background with black and red text

    Description automatically generated
* continue
  + A white background with black and red text

    Description automatically generated

**Functions**

* PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.
* Besides the built-in PHP functions, it is possible to create your own functions.
  + A function is a block of statements that can be used repeatedly in a program.
  + A function will not execute automatically when a page loads.
  + A function will be executed by a call to the function.
* A black text on a white background

  Description automatically generated
* Note: A function name must start with a letter or an underscore. Function names are NOT case-sensitive.
* Ex:
  + A screenshot of a computer code

    Description automatically generated
  + Output: A close up of black text

    Description automatically generated
* Loosely Typed
  + In the example above, notice that we did not have to tell PHP which data type the variable is.
  + PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.
  + In PHP 7, type declarations were added. This gives us an option to specify the expected data type when declaring a function, and by adding the strict declaration, it will throw a "Fatal Error" if the data type mismatches.
  + The strict declaration forces things to be used in the intended way.
  + In the following example we try to send both a number and a string to the function without using strict:
  + A white background with black text

    Description automatically generated
  + To specify strict we need to set declare(strict\_types=1);. This must be on the very first line of the PHP file.
  + In the following example we try to send both a number and a string to the function, but here we have added the strict declaration:
  + A screenshot of a computer code

    Description automatically generated
* Default Argument Value
  + The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:
  + A screenshot of a computer code

    Description automatically generated
* Returning Values
  + A computer code with black and red text

    Description automatically generated
  + PHP 7 also supports Type Declarations for the return statement. Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.
  + To declare a type for the function return, add a colon ( : ) and the type right before the opening curly ( { )bracket when declaring the function.
  + In the following example we specify the return type for the function:
  + A screenshot of a computer code

    Description automatically generated
* Passing Arguments by Reference
  + In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.
  + When a function argument is passed by reference, changes to the argument also change the variable that was passed in. To turn a function argument into a reference, the & operator is used:
  + A white background with black text

    Description automatically generated

**Arrays**

* An array stores multiple values in one single variable
* A close-up of a text

  Description automatically generated
* In PHP, the array() function is used to create an array:
* In PHP, there are three types of arrays:
  + Indexed arrays - Arrays with a numeric index
  + Associative arrays - Arrays with named keys
  + Multidimensional arrays - Arrays containing one or more arrays
* The count() function is used to return the length (the number of elements) of an array:
  + A close up of text

    Description automatically generated
* Indexed Arrays
  + There are two ways to create indexed arrays:
  + The index can be assigned automatically (index always starts at 0), like this:
  + 
  + or the index can be assigned manually:
  + A math equation with black text

    Description automatically generated
  + The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:
  + A close-up of a text

    Description automatically generated
  + To loop through and print all the values of an indexed array, you could use a for loop, like this:
  + A computer code with black text

    Description automatically generated
* Associative Arrays
  + Associative arrays are arrays that use named keys that you assign to them.
  + There are two ways to create an associative array:
  + 
  + Or:
  + A group of black text

    Description automatically generated
  + The named keys can then be used in a script:
  + A close up of a text

    Description automatically generated
  + To loop through and print all the values of an associative array, you could use a foreach loop, like this:
  + A white background with black text

    Description automatically generated
* Multidimensional Arrays
  + A multidimensional array is an array containing one or more arrays.
  + PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.
  + The dimension of an array indicates the number of indices you need to select an element. For a three-dimensional array you need three indices to select an element
  + First, take a look at the following table:
  + A screenshot of a computer

    Description automatically generated
  + We can store the data from the table above in a two-dimensional array, like this:
  + A white background with black text

    Description automatically generated
  + Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column.
  + To get access to the elements of the $cars array we must point to the two indices (row and column):
  + A screenshot of a computer code

    Description automatically generated
  + We can also put a for loop inside another for loop to get the elements of the $cars array (we still have to point to the two indices):
  + A screenshot of a computer code

    Description automatically generated
* Sorting Arrays
  + PHP has the following sort functions:
    - sort() - sort arrays in ascending order
    - rsort() - sort arrays in descending order
    - asort() - sort associative arrays in ascending order, according to the value
    - ksort() - sort associative arrays in ascending order, according to the key
    - arsort() - sort associative arrays in descending order, according to the value
    - krsort() - sort associative arrays in descending order, according to the key
  + The following example sorts the elements of the $cars array in ascending alphabetical order:
  + A close up of a text

    Description automatically generated
  + The following example sorts the elements of the $numbers array in ascending numerical order:
  + A close-up of a computer code

    Description automatically generated
  + The following example sorts the elements of the $cars array in descending alphabetical order:
  + A close up of a text

    Description automatically generated
  + The following example sorts the elements of the $numbers array in descending numerical order:
  + A close up of text

    Description automatically generated
  + The following example sorts an associative array in ascending order, according to the value:
  + A black and red text

    Description automatically generated
  + The following example sorts an associative array in ascending order, according to the key:
  + A close up of text

    Description automatically generated
  + The following example sorts an associative array in descending order, according to the value:
  + A black text on a white background

    Description automatically generated
  + The following example sorts an associative array in descending order, according to the key:
  + A black text on a white background

    Description automatically generated

**Global Variables – Superglobals**

* Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
* The PHP superglobal variables are:
  + $GLOBALS
  + $\_SERVER
  + $\_REQUEST
  + $\_POST
  + $\_GET
  + $\_FILES
  + $\_ENV
  + $\_COOKIE
  + $\_SESSION
* $GLOBALS
  + $GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).
  + PHP stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable.
  + The example below shows how to use the super global variable $GLOBALS:
  + A math equations on a white background

    Description automatically generated
  + In the example above, since z is a variable present within the $GLOBALS array, it is also accessible from outside the function!
* $\_SERVER
  + $\_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.
  + The example below shows how to use some of the elements in $\_SERVER:
  + A screenshot of a computer program

    Description automatically generated
* $\_REQUEST
  + PHP $\_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.
  + The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $\_REQUEST to collect the value of the input field:
  + A screen shot of a computer code

    Description automatically generated
* $\_POST
  + PHP $\_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $\_POST is also widely used to pass variables.
  + The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $\_POST to collect the value of the input field:
  + A screen shot of a computer code

    Description automatically generated
* $\_GET
  + PHP $\_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".
  + $\_GET can also collect data sent in the URL.
  + Assume we have an HTML page that contains a hyperlink with parameters:
  + A black text on a white background

    Description automatically generated
  + When a user clicks on the link "Test $GET", the parameters "subject" and "web" are sent to "test\_get.php", and you can then access their values in "test\_get.php" with $\_GET.
  + The example below shows the code in "test\_get.php":
  + A computer code with text

    Description automatically generated with medium confidence

**Form Handling**

* The PHP superglobals $\_GET and $\_POST are used to collect form-data.
* The example below displays a simple HTML form with two input fields and a submit button:
* A computer code with red and blue text

  Description automatically generated
* When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.
* To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:
* A black text on a white background

  Description automatically generated
* The same result could also be achieved using the HTTP GET method:
* A computer code with red and blue text

  Description automatically generated
* and "welcome\_get.php" looks like this:
* A black text on a white background

  Description automatically generated
* The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.
* GET vs POST
  + Both GET and POST create an array (e.g. array( key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.
  + Both GET and POST are treated as $\_GET and $\_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
  + $\_GET is an array of variables passed to the current script via the URL parameters.
  + $\_POST is an array of variables passed to the current script via the HTTP POST method.
* When to use GET?
  + Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
  + GET may be used for sending non-sensitive data.
  + Note: GET should NEVER be used for sending passwords or other sensitive information!
* When to use POST?
  + Information sent from a form with the POST method is invisible to others (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send.
  + Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
  + However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

**Form Validation**

* Suppose we have the following form within a .php file:
* A screen shot of a computer code

  Description automatically generated
* When the form is submitted, the form data is sent with method="post".
* The $\_SERVER["PHP\_SELF"] is a super global variable that returns the filename of the currently executing script. So, the $\_SERVER["PHP\_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.
* The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with &lt; and &gt;. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.
* The $\_SERVER["PHP\_SELF"] variable can be used by hackers!
* If PHP\_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.
* Assume we have the following form in a page named "test\_form.php":
* 
* Now, if a user enters the normal URL in the address bar like "http://www.example.com/test\_form.php", the above code will be translated to:
* 
* However, consider that a user enters the following URL in the address bar:
* <http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E>
* In this case, the above code will be translated to:
* 
* This code adds a script tag and an alert command. And when the page loads, the JavaScript code will be executed (the user will see an alert box). This is just a simple and harmless example how the PHP\_SELF variable can be exploited.
* Be aware of that any JavaScript code can be added inside the <script> tag! A hacker can redirect the user to a file on another server, and that file can hold malicious code that can alter the global variables or submit the form to another address to save the user data, for example.
* $\_SERVER["PHP\_SELF"] exploits can be avoided by using the htmlspecialchars() function.
* The form code should look like this:
* 
* The htmlspecialchars() function converts special characters to HTML entities. Now if the user tries to exploit the PHP\_SELF variable, it will result in the following output:
* A close up of a word

  Description automatically generated
* The exploit attempt fails, and no harm is done!
* Validate Form Data With PHP
  + The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.
  + When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:
  + <script>location.href('http://www.hacked.com')</script>
  + - this would not be executed, because it would be saved as HTML escaped code, like this:
  + &lt;script&gt;location.href('http://www.hacked.com')&lt;/script&gt;
  + The code is now safe to be displayed on a page or inside an e-mail.
  + We will also do two more things when the user submits the form:
    - Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
    - Remove backslashes (\) from the user input data (with the PHP stripslashes() function)
  + The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).
  + We will name the function test\_input().
  + Now, we can check each $\_POST variable with the test\_input() function, and the script looks like this:
  + A screenshot of a computer code

    Description automatically generated
  + Notice that at the start of the script, we check whether the form has been submitted using $\_SERVER["REQUEST\_METHOD"]. If the REQUEST\_METHOD is POST, then the form has been submitted - and it should be validated. If it has not been submitted, skip the validation and display a blank form.
  + However, in the example above, all input fields are optional. The script works fine even if the user does not enter any data.
  + The next step is to make input fields required and create error messages if needed

**Forms - Required Fields**

* In the previous chapter, all input fields were optional.
* In the following code we have added some new variables: $nameErr, $emailErr, $genderErr, and $websiteErr. These error variables will hold error messages for the required fields. We have also added an if else statement for each $\_POST variable. This checks if the $\_POST variable is empty (with the PHP empty() function). If it is empty, an error message is stored in the different error variables, and if it is not empty, it sends the user input data through the test\_input() function:
* A screenshot of a computer program

  Description automatically generated
* Then in the HTML form, we add a little script after each required field, which generates the correct error message if needed (that is if the user tries to submit the form without filling out the required fields):
* A screenshot of a computer program

  Description automatically generated

**Keep The Values in The Form**

* To show the values in the input fields after the user hits the submit button, we add a little PHP script inside the value attribute of the following input fields: name, email, and website. In the comment textarea field, we put the script between the <textarea> and </textarea> tags. The little script outputs the value of the $name, $email, $website, and $comment variables.
* Then, we also need to show which radio button that was checked. For this, we must manipulate the checked attribute (not the value attribute for radio buttons):
* A screenshot of a computer code

  Description automatically generated

**Include Files**

* The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.
* Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.
* The include and require statements are identical, except upon failure:
  + require will produce a fatal error (E\_COMPILE\_ERROR) and stop the script
  + include will only produce a warning (E\_WARNING) and the script will continue
* So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.
* Ex:
  + Assume we have a standard footer file called "footer.php", that looks like this:
  + A black and white text

    Description automatically generated
  + To include the footer file in a page, use the include statement:
  + A screenshot of a computer program

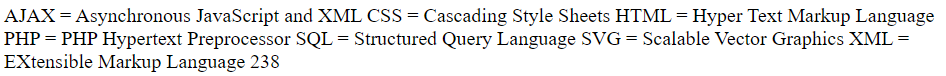
    Description automatically generated
* Ex:
  + A screenshot of a computer program

    Description automatically generated

**File Handling**

* PHP has several functions for creating, reading, uploading, and editing files.
* The readfile() function reads a file and writes it to the output buffer.
* Assume we have a text file called "webdictionary.txt", stored on the server, that looks like this:
* A white background with black text

  Description automatically generated
* The PHP code to read the file and write it to the output buffer is as follows (the readfile() function returns the number of bytes read on success):
* A black and red text

  Description automatically generated
* Output:
* 

**File Open/Read/Close**

* A better method to open files is with the fopen() function. This function gives you more options than the readfile() function.
* The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.
* The file may be opened in one of the following modes:
* A screenshot of a computer

  Description automatically generated
* The following example also generates a message if the fopen() function is unable to open the specified file:
* A computer code with red and black text

  Description automatically generated
* fread()
  + The fread() function reads from an open file.
  + The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
  + The following PHP code reads the "webdictionary.txt" file to the end:
  + 
* fclose()
  + The fclose() function is used to close an open file.
  + It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!
  + The fclose() requires the name of the file (or a variable that holds the filename) we want to close:
  + A computer code with text

    Description automatically generated
* fgets()
  + The fgets() function is used to read a single line from a file.
  + The example below outputs the first line of the "webdictionary.txt" file:
  + A close-up of a computer code

    Description automatically generated
  + Note: After a call to the fgets() function, the file pointer has moved to the next line.
* feof()
  + The feof() function checks if the "end-of-file" (EOF) has been reached.
  + The feof() function is useful for looping through data of unknown length.
  + The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:
  + A computer code with text

    Description automatically generated
* fgetc()
  + The fgetc() function is used to read a single character from a file.
  + The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached:
  + A computer screen shot of text

    Description automatically generated
  + Note: After a call to the fgetc() function, the file pointer moves to the next character.

**PHP File Create/Write**

* The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.
* If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).
* The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:
* 
* If you are having errors when trying to get this code to run, check that you have granted your PHP file access to write information to the hard drive.
* The fwrite() function is used to write to a file.
* The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.
* The example below writes a couple of names into a new file called "newfile.txt":
* A white background with black text

  Description automatically generated
* Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the fclose() function.
* If we open the "newfile.txt" file it would look like this:
* 
* Overwriting
  + Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.
  + In the example below we open our existing file "newfile.txt", and write some new data into it:
  + A white background with black text

    Description automatically generated
  + If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:
  + A group of black text

    Description automatically generated
* Append Text
  + You can append data to a file by using the "a" mode. The "a" mode appends text to the end of the file, while the "w" mode overrides (and erases) the old content of the file.
  + In the example below we open our existing file "newfile.txt", and append some text to it:
  + A computer code with text

    Description automatically generated
  + If we now open the "newfile.txt" file, we will see that Donald Duck and Goofy Goof is appended to the end of the file:
  + A group of black text

    Description automatically generated

**File Upload**

* First, ensure that PHP is configured to allow file uploads.
* In your "php.ini" file, search for the file\_uploads directive, and set it to On:
* 
* Next, create an HTML form that allow users to choose the image file they want to upload:
* A computer code with red and blue text

  Description automatically generated
* Some rules to follow for the HTML form above:
  + Make sure that the form uses method="post"
  + The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form
* Without the requirements above, the file upload will not work.
* Other things to notice:
  + The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control
* The form above sends data to a file called "upload.php", which we will create next.
* Create The Upload File PHP Script
  + The "upload.php" file contains the code for uploading a file:
  + A computer code with text

    Description automatically generated
  + PHP script explained:
    - $target\_dir = "uploads/" - specifies the directory where the file is going to be placed
    - $target\_file specifies the path of the file to be uploaded
    - $uploadOk=1 is not used yet (will be used later)
    - $imageFileType holds the file extension of the file (in lower case)
    - Next, check if the image file is an actual image or a fake image
  + Note: You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.
* Check if File Already Exists
  + Now we can add some restrictions.
  + First, we will check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:
  + A white background with black text

    Description automatically generated
* Limit File Size
  + The file input field in our HTML form above is named "fileToUpload".
  + Now, we want to check the size of the file. If the file is larger than 500KB, an error message is displayed, and $uploadOk is set to 0:
  + A white background with black text

    Description automatically generated
* Limit File Type
  + The code below only allows users to upload JPG, JPEG, PNG, and GIF files. All other file types gives an error message before setting $uploadOk to 0:
  + A black text on a white background

    Description automatically generated

**Cookies**

* A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.
* A cookie is created with the setcookie() function.
* 
* Only the name parameter is required. All other parameters are optional.
* The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 \* 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).
* We then retrieve the value of the cookie "user" (using the global variable $\_COOKIE). We also use the isset() function to find out if the cookie is set:
* A computer code with text

  Description automatically generated
* The setcookie() function must appear BEFORE the <html> tag.
* Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).
* Modify a Cookie Value
* To modify a cookie, just set (again) the cookie using the setcookie() function:
* A computer code with red and yellow text

  Description automatically generated
* To delete a cookie, use the setcookie() function with an expiration date in the past:
* A screenshot of a computer code

  Description automatically generated
* Check if Cookies are Enabled
* The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $\_COOKIE array variable:
* A screen shot of a computer code

  Description automatically generated

**Sessions**

* A session is a way to store information (in variables) to be used across multiple pages.
* Unlike a cookie, the information is not stored on the users computer.
* What is a PHP Session?
* When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
* Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.
* So; Session variables hold information about one single user, and are available to all pages in one application.
* If you need a permanent storage, you may want to store the data in a database.
* A session is started with the session\_start() function.
* Session variables are set with the PHP global variable: $\_SESSION.
* Now, let's create a new page called "demo\_session1.php". In this page, we start a new PHP session and set some session variables:
* A screen shot of a computer code

  Description automatically generated
* Note: The session\_start() function must be the very first thing in your document. Before any HTML tags.
* Next, we create another page called "demo\_session2.php". From this page, we will access the session information we set on the first page ("demo\_session1.php").
* Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session\_start()).
* Also notice that all session variable values are stored in the global $\_SESSION variable:
* A screen shot of a computer code

  Description automatically generated
* Another way to show all the session variable values for a user session is to run the following code:
* A screen shot of a computer code

  Description automatically generated
* How does it work? How does it know it's me?
* Most sessions set a user-key on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12. Then, when a session is opened on another page, it scans the computer for a user-key. If there is a match, it accesses that session, if not, it starts a new session.
* To change a session variable, just overwrite it:
* A screen shot of a computer code

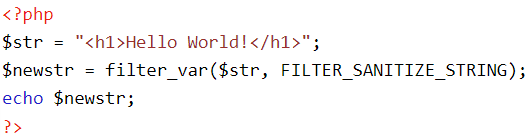
  Description automatically generated
* To remove all global session variables and destroy the session, use session\_unset() and session\_destroy():
* A screen shot of a computer code

  Description automatically generated

Filters

* PHP filters are used to validate and sanitize external input.
* The PHP filter extension has many of the functions needed for checking user input, and is designed to make data validation easier and quicker.
* Filter Extension
  + The filter\_list() function can be used to list what the PHP filter extension offers:
  + A computer code with black text

    Description automatically generatedA white sheet with black text

    Description automatically generated
* Many web applications receive external input. External input/data can be:
  + User input from a form
  + Cookies
  + Web services data
  + Server variables
  + Database query results
* You should always validate external data! Invalid submitted data can lead to security problems and break your webpage! By using PHP filters you can be sure your application gets the correct input!
* The filter\_var() function both validate and sanitize data.
* The filter\_var() function filters a single variable with a specified filter. It takes two pieces of data:
  + The variable you want to check
  + The type of check to use
* Sanitize a String
  + The following example uses the filter\_var() function to remove all HTML tags from a string:
  + 
* Validate an Integer
  + The following example uses the filter\_var() function to check if the variable $int is an integer. If $int is an integer, the output of the code below will be: "Integer is valid". If $int is not an integer, the output will be: "Integer is not valid":
  + A computer code with black text

    Description automatically generated
  + Tip: filter\_var() and Problem With 0
  + In the example above, if $int was set to 0, the function above will return "Integer is not valid". To solve this problem, use the code below:
  + A computer code with black text

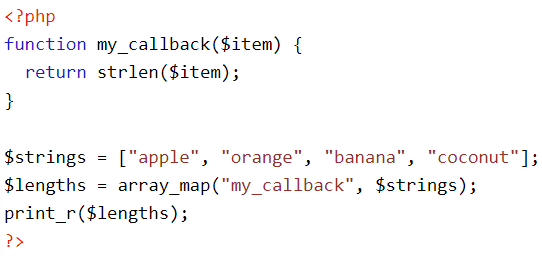
    Description automatically generated
* Validate an IP Address
  + The following example uses the filter\_var() function to check if the variable $ip is a valid IP address:
  + A screen shot of a computer code

    Description automatically generated
* Sanitize and Validate an Email Address
  + The following example uses the filter\_var() function to first remove all illegal characters from the $email variable, then check if it is a valid email address:
  + A computer code with text

    Description automatically generated with medium confidence
* Sanitize and Validate a URL
  + The following example uses the filter\_var() function to first remove all illegal characters from a URL, then check if $url is a valid URL:
  + A screenshot of a computer code

    Description automatically generated

**Callback Functions**

* A callback function (often referred to as just "callback") is a function which is passed as an argument into another function.
* Any existing function can be used as a callback function. To use a function as a callback function, pass a string containing the name of the function as the argument of another function:
  + Ex:
  + Pass a callback to PHP's array\_map() function to calculate the length of every string in an array:
  + 
  + Starting with version 7, PHP can pass anonymous functions as callback functions:
  + Use an anonymous function as a callback for PHP's array\_map() function:
  + A computer code with black text

    Description automatically generated
* User-defined functions and methods can also take callback functions as arguments. To use callback functions inside a user-defined function or method, call it by adding parentheses to the variable and pass arguments as with normal functions:
  + Ex:
  + A screenshot of a computer program

    Description automatically generated

**JSON**

* JSON stands for JavaScript Object Notation, and is a syntax for storing and exchanging data.
* Since the JSON format is a text-based format, it can easily be sent to and from a server, and used as a data format by any programming language.
* PHP has some built-in functions to handle JSON.
* json\_encode()
  + The json\_encode() function is used to encode a value to JSON format.
  + This example shows how to encode an associative array into a JSON object:
  + A black text with red and black text

    Description automatically generated with medium confidence
  + This example shows how to encode an indexed array into a JSON array:
  + A close up of text

    Description automatically generated
* json\_decode()
  + The json\_decode() function is used to decode a JSON object into a PHP object or an associative array.
  + A black text on a white background

    Description automatically generated
  + The json\_decode() function returns an object by default. The json\_decode() function has a second parameter, and when set to true, JSON objects are decoded into associative arrays.
  + This example decodes JSON data into a PHP associative array:
  + A black text on a white background

    Description automatically generated
* Accessing the Decoded Values
  + Here are two examples of how to access the decoded values from an object and from an associative array:
  + This example shows how to access the values from a PHP object:
  + A screenshot of a computer code

    Description automatically generated
  + This example shows how to access the values from a PHP associative array:
  + A screenshot of a computer code

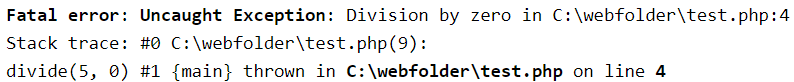
    Description automatically generated
* Looping Through the Values
  + You can also loop through the values with a foreach() loop:
  + This example shows how to loop through the values of a PHP object:
  + A computer code with black text

    Description automatically generated with medium confidence
  + This example shows how to loop through the values of a PHP associative array:
  + A computer code with black text

    Description automatically generated with medium confidence

**Exceptions**

* An exception is an object that describes an error or unexpected behaviour of a PHP script.
* Exceptions are thrown by many PHP functions and classes.
* User defined functions and classes can also throw exceptions.
* Exceptions are a good way to stop a function when it comes across data that it cannot use.
* Throwing an Exception
  + The throw statement allows a user defined function or method to throw an exception. When an exception is thrown, the code following it will not be executed.
  + If an exception is not caught, a fatal error will occur with an "Uncaught Exception" message.
  + Lets try to throw an exception without catching it:
  + A screenshot of a computer code

    Description automatically generated
  + The result will look something like this:
  + 
* The try...catch Statement
  + To avoid the error from the example above, we can use the try...catch statement to catch exceptions and continue the process.
  + A black text on a white background

    Description automatically generated
  + Show a message when an exception is thrown:
  + A screenshot of a computer code

    Description automatically generated
  + The catch block indicates what type of exception should be caught and the name of the variable which can be used to access the exception. In the example above, the type of exception is Exception and the variable name is $e.
* The try...catch...finally Statement
  + The try...catch...finally statement can be used to catch exceptions. Code in the finally block will always run regardless of whether an exception was caught. If finally is present, the catch block is optional.
  + A white background with black text

    Description automatically generated
  + Show a message when an exception is thrown and then indicate that the process has ended:
  + A screenshot of a computer code

    Description automatically generated
  + Output a string even if an exception was not caught:
  + A screenshot of a computer code

    Description automatically generated
* The Exception Object
  + The Exception Object contains information about the error or unexpected behaviour that the function encountered.
  + Syntax: 
  + A screenshot of a computer

    Description automatically generated
  + When catching an exception, the following table shows some of the methods that can be used to get information about the exception:
  + A screenshot of a computer

    Description automatically generated
  + Output information about an exception that was thrown:
  + A screenshot of a computer code

    Description automatically generated

**OOP**

* From PHP5, you can also write PHP code in an object-oriented style.
* Object-Oriented programming is faster and easier to execute.
* OOP stands for Object-Oriented Programming.
* Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.
* Object-oriented programming has several advantages over procedural programming:
  + OOP is faster and easier to execute
  + OOP provides a clear structure for the programs
  + OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
  + OOP makes it possible to create full reusable applications with less code and shorter development time
* Tip: The "Don't Repeat Yourself" (DRY) principle is about reducing the repetition of code. You should extract out the codes that are common for the application, and place them at a single place and reuse them instead of repeating it.
* So, a class is a template for objects, and an object is an instance of a class.
* When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.
* A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces.
* Below we declare a class named Fruit consisting of two properties ($name and $color) and two methods set\_name() and get\_name() for setting and getting the $name property:
* A screenshot of a computer code

  Description automatically generated
* In a class, variables are called properties and functions are called methods!
* Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.
* Objects of a class are created using the new keyword.
* In the example below, $apple and $banana are instances of the class Fruit:
* A screenshot of a computer code

  Description automatically generated
* In the example below, we add two more methods to class Fruit, for setting and getting the $color property:
* A screenshot of a computer code

  Description automatically generated
* The $this Keyword
  + The $this keyword refers to the current object, and is only available inside methods.
  + Look at the following example:
  + A computer code with black text

    Description automatically generated with medium confidence
  + So, where can we change the value of the $name property? There are two ways:
  + 1. Inside the class (by adding a set\_name() method and use $this):
  + A screenshot of a computer code

    Description automatically generated
  + 2. Outside the class (by directly changing the property value):
  + A computer code with black text

    Description automatically generated
* instanceof
  + You can use the instanceof keyword to check if an object belongs to a specific class:
  + A close up of a text

    Description automatically generated
* The \_\_construct Function
  + A constructor allows you to initialize an object's properties upon creation of the object.
  + If you create a \_\_construct() function, PHP will automatically call this function when you create an object from a class.
  + Notice that the construct function starts with two underscores (\_\_)!
  + We see in the example below, that using a constructor saves us from calling the set\_name() method which reduces the amount of code:
  + A computer code with many black text

    Description automatically generated with medium confidence
* The \_\_destruct Function
  + A destructor is called when the object is destructed or the script is stopped or exited.
  + If you create a \_\_destruct() function, PHP will automatically call this function at the end of the script.
  + Notice that the destruct function starts with two underscores (\_\_)!
  + The example below has a \_\_construct() function that is automatically called when you create an object from a class, and a \_\_destruct() function that is automatically called at the end of the script:
  + A computer code on a white background

    Description automatically generated
  + In PHP, a destructor is a special method within a class that is automatically called when an object of that class is no longer referenced or is explicitly destroyed using the unset() function. Destructors are used for cleanup tasks such as releasing resources or closing connections before an object is removed from memory.
  + A screen shot of a computer code

    Description automatically generated
  + In this example, the destructor method \_\_destruct() will be automatically called when the $obj object is unset. The destructor can also be implicitly called when an object goes out of scope (no longer referenced) and is garbage-collected by PHP's memory management system.
  + It's generally recommended to explicitly release resources when they are no longer needed, rather than relying solely on destructors. This ensures more predictable resource management in your application.
* Access Modifiers
  + Properties and methods can have access modifiers which control where they can be accessed.
  + There are three access modifiers:
    - public - the property or method can be accessed from everywhere. This is default
    - protected - the property or method can be accessed within the class and by classes derived from that class
    - private - the property or method can ONLY be accessed within the class
  + In the following example we have added three different access modifiers to three properties (name, color, and weight). Here, if you try to set the name property it will work fine (because the name property is public, and can be accessed from everywhere). However, if you try to set the color or weight property it will result in a fatal error (because the color and weight property are protected and private):
  + A computer code with text

    Description automatically generated
  + In the next example we have added access modifiers to two functions. Here, if you try to call the set\_color() or the set\_weight() function it will result in a fatal error (because the two functions are considered protected and private), even if all the properties are public:
  + A computer code with text

    Description automatically generated with medium confidence
* Inheritance
  + What is Inheritance?
  + Inheritance in OOP = When a class derives from another class.
  + The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.
  + An inherited class is defined by using the extends keyword.
  + Let's look at an example:
  + A computer code with text

    Description automatically generated with medium confidence
  + The Strawberry class is inherited from the Fruit class.
  + This means that the Strawberry class can use the public $name and $color properties as well as the public \_\_construct() and intro() methods from the Fruit class because of inheritance.
  + The Strawberry class also has its own method: message().
* Inheritance and the Protected Access Modifier
  + In the previous chapter we learned that protected properties or methods can be accessed within the class and by classes derived from that class. What does that mean?
  + Let's look at an example:
  + A computer code with red and green text

    Description automatically generated
* Overriding Inherited Methods
  + Inherited methods can be overridden by redefining the methods (use the same name) in the child class.
  + Look at the example below. The \_\_construct() and intro() methods in the child class (Strawberry) will override the \_\_construct() and intro() methods in the parent class (Fruit):
  + A computer screen shot of a program

    Description automatically generated
* The final Keyword
  + The final keyword can be used to prevent class inheritance or to prevent method overriding.
  + The following example shows how to prevent class inheritance:
  + A screen shot of a computer code

    Description automatically generated
  + The following example shows how to prevent method overriding:
* Class Constants
  + Constants cannot be changed once it is declared.
  + Class constants can be useful if you need to define some constant data within a class.
  + A class constant is declared inside a class with the const keyword.
  + Class constants are case-sensitive. However, it is recommended to name the constants in all uppercase letters.
  + We can access a constant from outside the class by using the class name followed by the scope resolution operator (::) followed by the constant name, like here:
  + A white background with black text

    Description automatically generated
  + Or, we can access a constant from inside the class by using the self keyword followed by the scope resolution operator (::) followed by the constant name, like here:
  + A white screen with black text

    Description automatically generated
* Abstract Classes
  + Abstract classes and methods are when the parent class has a named method, but need its child class(es) to fill out the tasks.
  + An abstract class is a class that contains at least one abstract method. An abstract method is a method that is declared, but not implemented in the code.
  + An abstract class or method is defined with the abstract keyword:
  + A computer code with black text

    Description automatically generated with medium confidence
  + When inheriting from an abstract class, the child class method must be defined with the same name, and the same or a less restricted access modifier. So, if the abstract method is defined as protected, the child class method must be defined as either protected or public, but not private. Also, the type and number of required arguments must be the same. However, the child classes may have optional arguments in addition.
  + So, when a child class is inherited from an abstract class, we have the following rules:
    - The child class method must be defined with the same name and it redeclares the parent abstract method
    - The child class method must be defined with the same or a less restricted access modifier
    - The number of required arguments must be the same. However, the child class may have optional arguments in addition
  + A screenshot of a computer code

    Description automatically generated
  + The Audi, Volvo, and Citroen classes are inherited from the Car class. This means that the Audi, Volvo, and Citroen classes can use the public $name property as well as the public \_\_construct() method from the Car class because of inheritance.
  + But, intro() is an abstract method that should be defined in all the child classes and they should return a string.
  + Let's look at another example where the abstract method has an argument, and the child class has two optional arguments that are not defined in the parent's abstract method:
  + A screen shot of a computer code

    Description automatically generated

**XML Parsers**

* The XML language is a way to structure data for sharing across websites.
* Several web technologies like RSS Feeds and Podcasts are written in XML.
* XML is easy to create. It looks a lot like HTML, except that you make up your own tags.
* To read and update, create and manipulate an XML document, you will need an XML parser.
* In PHP there are two major types of XML parsers:
  + Tree-Based Parsers
  + Event-Based Parsers
* Tree-Based Parsers
  + Tree-based parsers holds the entire document in Memory and transforms the XML document into a Tree structure. It analyzes the whole document, and provides access to the Tree elements (DOM).
  + This type of parser is a better option for smaller XML documents, but not for large XML document as it causes major performance issues.
  + Example of tree-based parsers:
    - SimpleXML
    - DOM
* Event-Based Parsers
  + Event-based parsers do not hold the entire document in Memory, instead, they read in one node at a time and allow you to interact with in real time. Once you move onto the next node, the old one is thrown away.
  + This type of parser is well suited for large XML documents. It parses faster and consumes less memory.
  + Example of event-based parsers:
    - XMLReader
    - XML Expat Parser
* SimpleXML is a PHP extension that allows us to easily manipulate and get XML data.
* The SimpleXML Parser
  + SimpleXML is a tree-based parser.
  + SimpleXML provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout.
  + SimpleXML turns an XML document into a data structure you can iterate through like a collection of arrays and objects.
  + Compared to DOM or the Expat parser, SimpleXML takes a fewer lines of code to read text data from an element.
  + PHP SimpleXML - Read From String
  + The PHP simplexml\_load\_string() function is used to read XML data from a string.
  + Assume we have a variable that contains XML data, like this:
  + A white background with black text

    Description automatically generated
  + The example below shows how to use the simplexml\_load\_string() function to read XML data from a string:
  + A computer code on a white background

    Description automatically generated
  + The output of the code above will be:
  + A close up of words

    Description automatically generated
  + Error Handling Tip: Use the libxml functionality to retrieve all XML errors when loading the document and then iterate over the errors. The following example tries to load a broken XML string:
  + A screenshot of a computer code

    Description automatically generated
  + The output of the code above will be:
  + A black text on a white background

    Description automatically generated
  + PHP SimpleXML - Read From File
  + The PHP simplexml\_load\_file() function is used to read XML data from a file.
  + Assume we have an XML file called "note.xml", that looks like this:
  + A white background with black text

    Description automatically generated
  + The example below shows how to use the simplexml\_load\_file() function to read XML data from a file:
  + A black text on a white background

    Description automatically generated
  + The output of the code above will be:
  + A close up of words

    Description automatically generated
  + PHP SimpleXML - Get Node Values
  + Get the node values from the "note.xml" file:
  + A computer code with text

    Description automatically generated
  + The output of the code above will be:
  + A black text on a white background

    Description automatically generated
  + Assume we have an XML file called "books.xml", that looks like this:
  + A screenshot of a computer code

    Description automatically generated
  + The following example gets the node value of the <title> element in the first and second <book> elements in the "books.xml" file:
  + A white background with black text

    Description automatically generated
  + The output of the code above will be:
  + A black and white text

    Description automatically generated
  + PHP SimpleXML - Get Node Values – Loop
  + The following example loops through all the <book> elements in the "books.xml" file, and gets the node values of the <title>, <author>, <year>, and <price> elements:
  + A computer code with text

    Description automatically generated
  + The output of the code above will be:
  + A close-up of a text

    Description automatically generated
  + The following example gets the attribute value of the "category" attribute of the first <book> element and the attribute value of the "lang" attribute of the <title> element in the second <book> element:
  + A computer code with text

    Description automatically generated with medium confidence
  + The output of the code above will be:
  + A black text on a white background

    Description automatically generated
* The XML Expat Parser
  + The Expat parser is an event-based parser.
  + Look at the following XML fraction:
  + 
  + An event-based parser reports the XML above as a series of three events:
    - Start element: from
    - Start CDATA section, value: Jani
    - Close element: from
  + The XML file "note.xml" will be used in the example below:
  + A white background with black text

    Description automatically generated
  + We want to initialize the XML Expat Parser in PHP, define some handlers for different XML events, and then parse the XML file.
  + A screen shot of a computer code

    Description automatically generated
  + A screenshot of a computer code

    Description automatically generated
  + Example explained:
    - Initialize the XML parser with the xml\_parser\_create() function
    - Create functions to use with the different event handlers
    - Add the xml\_set\_element\_handler() function to specify which function will be executed when the parser encounters the opening and closing tags
    - Add the xml\_set\_character\_data\_handler() function to specify which function will execute when the parser encounters character data
    - Parse the file "note.xml" with the xml\_parse() function
    - In case of an error, add xml\_error\_string() function to convert an XML error to a textual description
    - Call the xml\_parser\_free() function to release the memory allocated with the xml\_parser\_create() function
* PHP XML DOM Parser
  + The built-in DOM parser makes it possible to process XML documents in PHP.
  + The DOM parser is a tree-based parser.
  + Look at the following XML document fraction:
  + 
  + The DOM sees the XML above as a tree structure:
    - Level 1: XML Document
    - Level 2: Root element: <from>
    - Level 3: Text element: "Jani"
  + The XML file below ("note.xml") will be used in our example
  + A white background with black text

    Description automatically generated
  + We want to initialize the XML parser, load the xml, and output it:
  + A computer code with text

    Description automatically generated with medium confidence
  + The output of the code above will be:
  + 
  + If you select "View source" in the browser window, you will see the following HTML:
  + A white background with black text

    Description automatically generated
  + The example above creates a DOMDocument-Object and loads the XML from "note.xml" into it.
  + Then the saveXML() function puts the internal XML document into a string, so we can output it.
  + We want to initialize the XML parser, load the XML, and loop through all elements of the <note> element:
  + A computer code with black text

    Description automatically generated
  + The output of the code above will be:
  + A white background with black text

    Description automatically generated
  + In the example above you see that there are empty text nodes between each element.
  + When XML generates, it often contains white-spaces between the nodes. The XML DOM parser treats these as ordinary elements, and if you are not aware of them, they sometimes cause problems.

**MySQL**

* + MySQL is the most popular database system used with PHP.
  + PHP 5 and later can work with a MySQL database using:
    - MySQLi extension (the "i" stands for improved)
    - PDO (PHP Data Objects)
  + Both MySQLi and PDO have their advantages:
  + PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.
  + So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.
  + Both are object-oriented, but MySQLi also offers a procedural API.
  + Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.
  + Before we can access data in the MySQL database, we need to be able to connect to the server:
  + Example (MySQLi Object-Oriented)
    - A computer code with many colored text

      Description automatically generated with medium confidence
  + Example (MySQLi Procedural)
    - A computer code with text

      Description automatically generated
  + Example (PDO)
    - A computer code with text

      Description automatically generated
  + The connection will be closed automatically when the script ends. To close the connection before, use the following:
  + MySQLi Object-Oriented:
    - $conn->close();
  + MySQLi Procedural:
    - mysqli\_close($conn);
  + PDO:
    - $conn = null;

**Ajax**

* AJAX = Asynchronous JavaScript and XML.
* AJAX is a technique for creating fast and dynamic web pages.
* AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
* Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.
* Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.
* A diagram of a computer

  Description automatically generated
* AJAX is Based on Internet Standards
* AJAX is based on internet standards, and uses a combination of:
  + XMLHttpRequest object (to exchange data asynchronously with a server)
  + JavaScript/DOM (to display/interact with the information)
  + CSS (to style the data)
  + XML (often used as the format for transferring data)
* AJAX applications are browser- and platform-independent!
* AJAX PHP Example
  + The following example will demonstrate how a web page can communicate with a web server while a user type characters in an input field:
  + A screen shot of a computer

    Description automatically generated
  + In the example above, when a user types a character in the input field, a function called "showHint()" is executed.
  + The function is triggered by the onkeyup event.
  + Here is the HTML code:
  + A screen shot of a computer code

    Description automatically generated
  + Code explanation:
  + First, check if the input field is empty (str.length == 0). If it is, clear the content of the txtHint placeholder and exit the function.
  + However, if the input field is not empty, do the following:
    - Create an XMLHttpRequest object
    - Create the function to be executed when the server response is ready
    - Send the request off to a PHP file (gethint.php) on the server
    - Notice that q parameter is added to the url (gethint.php?q="+str)
    - And the str variable holds the content of the input field
* AJAX Database Example
  + The following example will demonstrate how a web page can fetch information from a database with AJAX:
  + A black and white text

    Description automatically generated
  + In the example above, when a user selects a person in the dropdown list above, a function called "showUser()" is executed.
  + The function is triggered by the onchange event.
  + Here is the HTML code:
  + A screenshot of a computer code

    Description automatically generated
  + Code explanation:
  + First, check if person is selected. If no person is selected (str == ""), clear the content of txtHint and exit the function. If a person is selected, do the following:
    - Create an XMLHttpRequest object
    - Create the function to be executed when the server response is ready
    - Send the request off to a file on the server
    - Notice that a parameter (q) is added to the URL (with the content of the dropdown list
  + The page on the server called by the JavaScript above is a PHP file called "getuser.php".
  + The source code in "getuser.php" runs a query against a MySQL database, and returns the result in an HTML table:
  + A screenshot of a computer program

    Description automatically generated
  + Explanation: When the query is sent from the JavaScript to the PHP file, the following happens:
  + PHP opens a connection to a MySQL server
  + The correct person is found
  + An HTML table is created, filled with data, and sent back to the "txtHint" placeholder