Functions

This week:

- Concept of a function
- Built-in PHP functions

Code demonstrations on GitHub.

PHP Functions: concept of a function

Before going any further, It would be wise to review what are functions in computer science.

Functions are series of statements (instructions), so blocks of code, that are defined in an application and that can be executed on demand. Executing a function is called ***calling*** the function. Usually, functions have the following properties:

- They have their own execution context, so any variable declared inside them exists only inside them
- They are declared through a ***signature***, an instruction that define all the structure of the function, but not its contents.
- They can have *intrants* (inputs); values ***passed*** to the function when it is called (the values are called ***arguments***) that will be made available inside the function's execution context by being assigned to local variables (the variables are called ***parameters***) .
- They can have an output: a value that is ***returned*** when the function terminates.
- They can have a identifier (name) or be anonymous.

PHP Functions: concept of a function

To help understand the concept, you can imagine a function as being an industrial machine: you give it some input(s), it does something with it, then outputs a result.

In PHP, once defined, functions defined with an identifier (name) in the global space (outside of a class) can be referenced to (and thus be called/executed) subsequently in the application via their name and the call pseudo-operator. However contrary to some other languages, because they are global references, PHP function identifiers are unique, and so we cannot define functions with the same name, even if they have a different signature. *Function overloading doesn't work in PHP the same way it does in other OOP languages*. That means that you cannot have variations of a function with the same name but different parameters.

In object-oriented programming, when defined as ***members of an class***, functions are called **methods**. Methods differ from functions in that they are defined in a class, and thus exist in that class's context. To reference them, one must go through the class (static methods) or class instance (non-static).

PHP built-in functions

PHP built-in functions

PHP has an enormous amount of utility functions that are defined in PHP itself, aka that are built-in.

You can use them as you need in your own development work.

We will now see some of the more usual and most often used built-in PHP functions. The list is separated in sections based on what they are used to affect.

When writing this function list, for clarity and simplicity, although some of them predate the existence of parameter typing, I have written the functions with parameter types. I have tried to make the parameter names as self-explanatory as possible. They might not match their name in PHP, but that has no effect on the usage.

Finally, some functions have optional parameters; parameters that can receive an argument, but do not need to. I have written those between square brackets for clarity.

Format: <function name>(<required parameters> [, <optional parameters>]) : <return type>

Example: isset(mixed $variable, [mixed ...$moreVariables]) : bool

*Note: the « ... » used above means that the function is **variadic** and can take any number of such arguments.*

PHP built-in functions: Variable handling functions

See PHP variable handling functions

- isset(mixed $aVariable, [mixed ...$otherVariables]) : bool The isset() function returns true if (all) the variable(s) passed are defined and not null.
- unset(mixed $aVariable, [mixed ...$otherVariables]) : void The unset() function destroys the specified variables.
- empty(mixed $aVariable) : bool The empty() function returns true if the passed variable is considered empty. *A variable is considered empty if it does not exist or if its value equals false.*
- var_dump(mixed $expression, [mixed ...$expressions]) : void The var_dump() functions sends information about the expression(s) passed *to the output*, including type and value.
- var_export(mixed $expression, [bool $return = false]) : ?string The var_export() function behaves like the var_dump() one for a single expression, but the generated string is valid PHP code than can be executed, and if $return is set to true, it will return the result as a string instead of dumping it to the output.

PHP built-in functions: Variable handling functions (continued)

- gettype(mixed $value) : string The gettype() function returns the type of the passed $value as a string. See gettype() for the specific possible return values.
- settype(mixed &$variable, string $type) : bool The settype() function attempts to set the type of the passed variable. See settype() for the specific acceptable values for $type. Returns true on success, false on failure.
- is_a(mixed $object_or_classname, string $classname, [bool $allow_string = false]) : bool The is_a() function returns true if the value of $object_or_classname is of a specific class ($classname) or has this specific class ($classname) as one of its parents. If $allow_string is set to false (which it is by default), string class names as values for $object_or_classname are not allowed.

PHP built-in functions: String functions

See PHP string functions

- strlen(string $string) : int Returns the length of the passed $string.
- strpos(string $haystack, string $needle, [int $offset = 0]): int|false Returns the position of the first occurrence of the $needle sub-string inside the $haystack string, beginning from the start of the string, or false if it is not found. If $offset is set, starts the search from the specified offset position in the $haystack string.
- strtolower(string $string): string Returns a copy of the $string string with all the ASCII characters converted to lowercase. ***Non-ASCII characters are not converted***.
- strtoupper(string $string): string Returns a copy of the $string string with all the ASCII characters converted to uppercase. ***Non-ASCII characters are not converted***.
- str_contains(string $haystack, string $needle): bool Returns true if the $needle sub-string is contained inside the $haystack string, and false otherwise. The check is case-sensitive.

NOTE: most PHP string functions work only with single-byte ASCII characters. To handle multi-byte character strings, you must use the *mb_** versions of the functions. See multibyte string functions for details.

PHP built-in functions: String functions (continued)

- str_replace(array|string $search, array|string $replace, string|array $subject, [int &$count = null]) : string|array Returns a string or an array with all occurrences of $search in $subject replaced with the given $replace value. See str_replace() for the specific behavior when using array arguments.
- sprintf(string $format, [mixed ...$values]) : string Returns a string produced according to the formatting string $format. See sprintf() for details on formatting options.
- ltrim(string $string, [string $characters = " \n\r\t\v\x00"]) : string Strips whitespace (or other $characters) from the beginning of a $string. Returns the modified string.
- rtrim(string $string, [string $characters = " \n\r\t\v\x00"]) : string Strips whitespace (or other $characters) from the end of a $string. Returns the modified string.
- trim(string $string, [string $characters = " \n\r\t\v\x00"]) : string Strips whitespace (or other $characters) from both ends of a $string. Returns the modified string.
- is_numeric(mixed $value) : bool Returns true if $value is a number or a numeric string, false otherwise.

PHP built-in functions: Math functions

See PHP math functions

- min(mixed $comparable, [mixed ...$comparables]) : mixed Finds the lowest value among the passed comparable argument(s).
- min(array $array) : mixed Finds the lowest value in the passed array.
- max(mixed $comparable, [mixed ...$comparables]) : mixed Finds the greatest value among the passed comparable arguments.
- max(array $array) : mixed Finds the greatest value in the passed array.
- abs(int|float $number) : int|float Returns the absolute value of a $number.
- round(int|float $number, [int $precision = 0, int $mode = PHP_ROUND_HALF_UP]) : float Returns the rounded value of $number to specified $precision (number of digits after the decimal point); $precision can also be negative or zero (default).

PHP built-in functions: Math functions

- ceil(int|float $number) : float Returns the next highest integer value (as a float) by rounding ***up*** $number if necessary.
- floor(int|float $number) : float Returns the next lowest integer value (as a float) by rounding ***down*** $number if necessary.
- pow(mixed $number, mixed $exponent) : int|float|object Returns $number raised to the power of $exponent.
- rand([int $min = 0, int $max = 2147483647]) : int Generate a ***non-cryptographically secure*** random integer between $min and $max inclusively.
- random_int(int $min, int $max) : int Generate a ***cryptographically secure*** random integer between $min and $max inclusively.

PHP built-in functions: Array functions

See PHP array functions

- count(Countable|array $value, [int $mode = COUNT_NORMAL]) : int Counts all elements in an array or in a Countable object.
- array_key_exists(string|int $key, array $array) : bool Returns true if the given $key is set in the $array.
- array_search(mixed $needle, array $haystack, [bool $strict = false]) : int|string|false Searches the $haystack array for a given $needle value and returns the first corresponding key if successful, false otherwise.
- is_array(mixed $variable) : bool Returns true if the $variable is an array, false otherwise.
- implode(string $separator, array $array) : string Returns a string containing a string representation of all the $array elements in the same order, with the $separator string between each element.
- explode(string $separator, string $string, [int $limit = PHP_INT_MAX]) : array Returns an array of strings, each of which is a sub-string of $string formed by splitting it on boundaries formed by the $separator.

PHP built-in functions: Array *cursor* functions

- current(array|object $array) : mixed Returns the current element in an $array. Every array has an internal pointer to its "current" element, which is initialized to the first element inserted into the array.
- reset(array|object &$array) : mixed Rewinds the $array's internal pointer to the first element and returns the value of the first array element or false if the $array is empty.
- each(array|object &$array) : array|false Returns the current key and value pair from an $array and advances the internal pointer. If the internal pointer points past the end of the array's contents, each() returns false. ***Removed in PHP 8.0+*** .
- next(array|object &$array) : mixed Returns the next value in the $array and advances the internal pointer by one. If the internal pointer points past the end of the array's contents, each() returns false.
- prev(array|object &$array) : mixed Behaves just like next(), except it rewinds the internal pointer one place instead of advancing it.
- end(array|object &$array) : mixed Advances the $array's internal pointer to the last element, and returns its value.

PHP built-in functions: Array *sorting* functions

- sort(array &$array, [int $flags = SORT_REGULAR]) : bool Sorts the $array in place by values in ascending order. Always returns true. See sort() for information on flags and specific behavior.
- rsort(array &$array, [int $flags = SORT_REGULAR]) : bool same as sort(), but in descending order.
- asort(array &$array, [int $flags = SORT_REGULAR]) : bool Sort an $array in ascending order while maintaining key associations.
- arsort(array &$array, [int $flags = SORT_REGULAR]) : bool Same as asort() but in descending order.
- ksort(array &$array, [int $flags = SORT_REGULAR]) : bool Sorts the $array in place by keys in ascending order.
- krsort(array &$array, [int $flags = SORT_REGULAR]) : bool Same as asort() but in descending order.
- usort(array &$array, callable $callback) : bool Sorts the $array by values using a user-supplied $callback function to determine the order. The callback function must return an int less than, equal to, or greater than zero if the **first argument** is considered to be respectively less than, equal to, or greater than the **second**:

  callback(mixed **$a**, mixed **$b**) : int

Next week

- User-defined functions
- Handling dates and time in PHP