Функции

Функции, определяемые пользователем

Внутри функции можно использовать любой корректный PHP-код, в том числе другие функции и даже объявления классов. Имена функций следуют тем же правилам, что и другие метки в PHP.

Функции не обязаны быть определены до их использования, исключая тот случай, когда функции определяются условно. В случае, когда функция определяется в зависимости от какого-либо условия, обработка описания функции должна предшествовать ее вызову.

```php
<?php
    $makefoo = true;

    bar();

    function foo() {
        echo "Hello #2";
    }

    if ($makefoo) foo();

    function bar() {
        echo "Hello #1";
    }
?>
```

Вложенные функции

```php
<?php
    function foo() {
        function bar() {
            echo "Я не существую пока не будет вызвана foo()";
        }
    }

    foo();
    bar();
?>
```

PHP не поддерживает перегрузку функции, также отсутствует возможность переопределить или удалить объявленную ранее функцию.

Имена функций регистронезависимы, тем не менее, более предпочтительно вызывать функции так, как они были объявлены.

Можно вызывать функции PHP рекурсивно.

```php
<?php
function recursion($a) {
    if ($a <= 20) {
        echo "$a\n";
        recursion($a + 1);
    }
}

recursion(1);

// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
?>
```

Рекурсивный вызов методов/процедур с глубиной более 100-200 уровней рекурсии может вызвать переполнение стека и привести к аварийному завершению скрипта. В частности, бесконечная рекурсия будет считаться программной ошибкой.

Аргументы функции

Функция может принимать информацию в виде списка аргументов, который является списком разделенных запятыми выражений. Аргументы вычисляются слева направо.

PHP поддерживает передачу аргументов по значению (по умолчанию), передачу аргументов по ссылке, и значения по умолчанию.

```php
<?php
    $input = [
        0 => 2,
        1 => 3
    ];

    function callArray($input) {
        echo "$input[0] + $input[1] = ", $input[0] + $input[1];
    }

    callArray($input); // 2 + 3 = 5
?>
```

Передача аргументов по ссылке

По умолчанию аргументы в функцию передаются по значению (это означает, что если вы измените значение аргумента внутри функции, то вне ее значение все равно останется прежним). Если вы хотите разрешить функции модифицировать свои аргументы, вы должны передавать их по ссылке. Если вы хотите, что бы аргумент всегда передавался по ссылке, вы можете указать амперсанд (&) перед именем аргумента в описании функции.

Значения аргументов по умолчанию

```php
<?php
    function makeCoffee($type = "string") {
        return "Type: $type.<br/>";
    }

    echo makeCoffee();
    echo makeCoffee(null);
    echo makeCoffee("integer");

    /*
    Type: string.
    Type: .
    Type: integer.
    */
?>
```

Возврат значений

Значения возвращаются при помощи необязательного оператора возврата. Возвращаемые значения могут быть любого типа, в том числе это могут быть массивы и объекты. Возврат приводит к завершению выполнения функции и передаче управления обратно к той строке кода, в которой данная функция была вызвана.

Если конструкция return не указана, то функция вернет значение NULL.

```php
<?php
    function square($num) {
        return $num * $num;
    }

    echo square(4); // 16
?>
```

Функция не может возвращать несколько значений, но аналогичного результата можно добиться, возвращая массив.

Обращение к функциям через переменные

PHP поддерживает концепцию переменных функций. Это означает, что если к имени переменной присоединены круглые скобки, PHP ищет функцию с тем же именем, что и результат вычисления переменной, и пытается ее выполнить. Эту возможность можно использовать для реализации обратных вызовов, таблиц функций и множества других вещей.

```php
<?php
    $funcName = 'sayHello';
    $funcName('Orkhan');

    function sayHello($name) {
        echo "Hello, $name";
    }

    // Hello, Orkhan
?>
```

Анонимные функции

Анонимные функции, также известные как замыкания (closures), позволяют создавать функции, не имеющие определенных имен.

```php
<?php
    $greet = function($name) {
        print("Hello, $name");
    };

    $greet('PHP');

    // Hello, PHP
?>
```