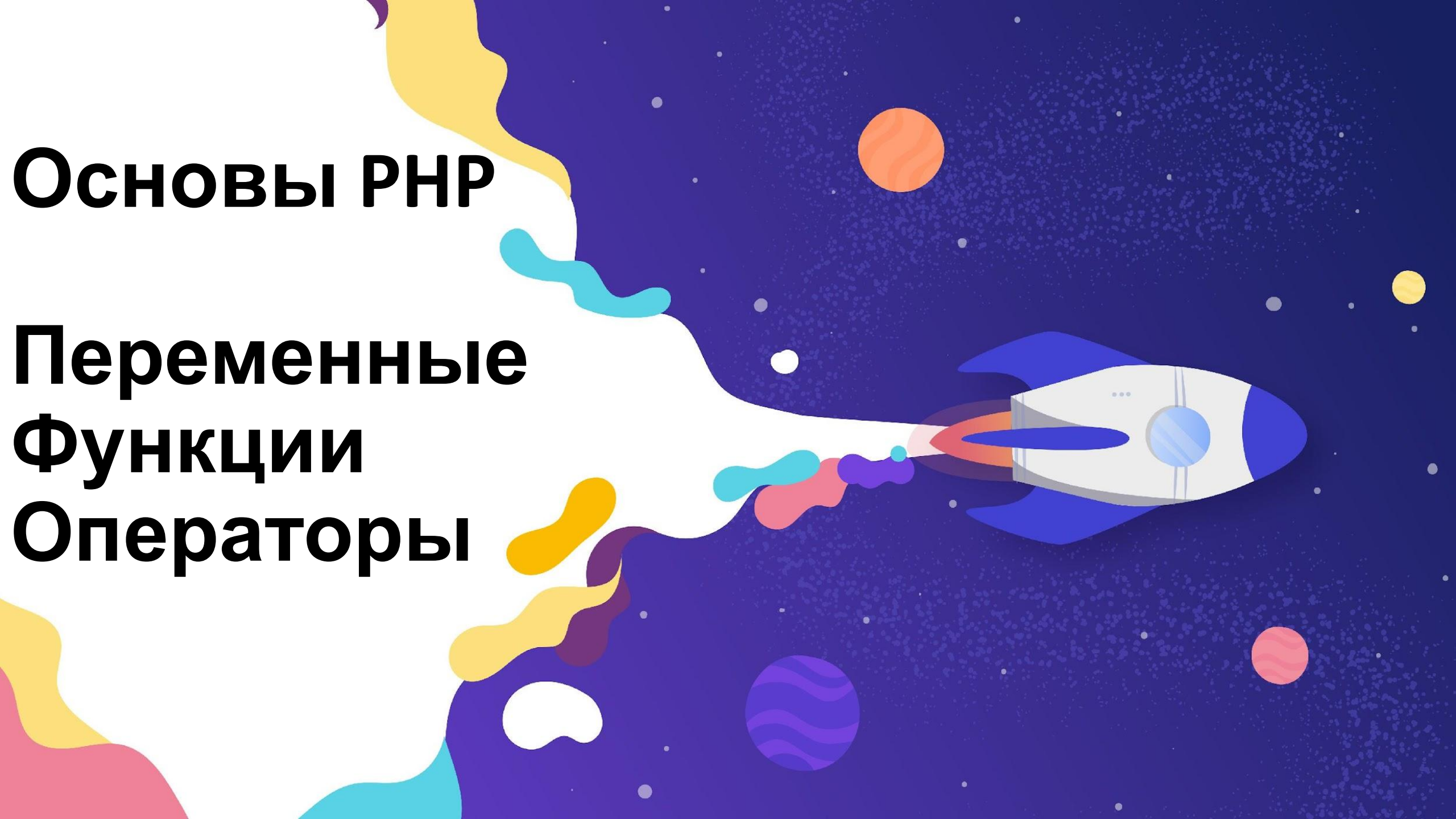
Основы PHP

Переменные
Функции
Операторы



Типы данных.

PHP является языком с динамической типизацией. Это значит, что тип данных переменной выводится во время выполнения, и в отличии от ряда других языков программирования в PHP не надо указывать перед переменной тип данных.
В PHP есть десять базовых типов данных:

- Скалярные
  - bool (логический тип)
  - int (целые числа)
  - float (дробные числа)
  - string (строки)
- Нескалярные
  - array (массивы)
  - object (объекты)
  - callable (функции)
  - resource (ресурсы)
- Специальные
  - null (отсутствие значения)
  - mixed (любой тип)

Типы данных. Integer

Представляет целое число со знаком.

```
<?php
    $number = -100;
    echo $number;
?>
```

Здесь переменная $number представляет целочисленный тип, так как ей присваивается целочисленное значение.

Кроме десятичных целых чисел PHP обладает возможностью использовать также двоичные, восьмеричные и шестнадцатеричные числа. Шаблоны чисел для других систем:

- шестнадцатеричные : 0[xX][0-9a-fA-F]
- восьмеричные : 0[0-7]
- двоичные : 0b[01]

Типы данных. Float

Размер числа с плавающей точкой зависит от платформы. Максимально возможное значение, как правило, составляет $1.8*10^{308}$ с точностью около 14 десятичных цифр.

```
<?php
    $a1 = 1.5;
    $a2 = 1.3e4; // 1.3 * 10^4
    $a3 = 6E-8; // 0.00000006
?>
```

Типы данных. Bool

Переменные логического типа могут принимать два значения: true и false
или иначе говоря истина и ложь.
Чаще всего логические значения используются в условных конструкциях:

```
<?php
    $foo = true;
    $bar = false;
?>
```

Типы данных. String

Для работы с текстом можно применять строки. Строки бывают двух типов: в двойных кавычках и одинарных. От типа кавычек зависит обработка строк интерпретатором. Так, переменные в двойных кавычках заменяются значениями, а переменные в одинарных кавычках остаются неизменными.

```
<?php
	$a=10;
	$b=5;
	$result = "$a+$b <br>";
	echo $result;
	$result = '$a+$b';
	echo $result;
?>
```

В этом случае мы получим следующий вывод:

10+5

$a+$b

Типы данных. String

Кроме обычных символов, строка может содержать специальные символы, которые могут быть неправильно интерпретированы. Например, нам надо добавить в строку кавычку:

```
$text = "Модель "Apple II"";
```

Данная запись будет ошибочна. Чтобы исправить ошибку, мы можем сочетать различные типы кавычек ('Модель "Apple II"' или "Модель 'Apple III'") или использовать слеш, чтобы ввести кавычку в строку:

```
$text = "Модель \"Apple II\"";
```

Типы данных. Null

Значение null указывает, переменная не имеет значения. Использование данного значения полезно в тех случаях, когда необходимо указать, что переменная не имеет значения.

Null выделяют, как отдельный тип, т.к. начиная с версии PHP 7.4 для свойств класса, а также для аргументов функций и их возвращаемых значений можно использовать конструкцию ?<type>. Например,

```
<?php
    public function myAwesomeFunction (string $a, ?array $b  = null): ?int {...}
?>
```

Если же просто определить переменную без ее инициализации, и затем попробовать ее использовать, то интерпретатор выдаст сообщение, что переменная не установлена.

Типы данных. Null

Использование значения null поможет избежать данной ситуации. Кроме того, можно проверить наличие значения и в зависимости от результатов проверки производить те или иные действия:

```
<?php
	$a = null;
	echo "a = $a";
?>
```

Константа null не чувствительна к регистру, поэтому можно написать и так:

```
	$a = NULL;
```

Но лучше так не делать, и придерживаться стандартов кодирования.

Типы данных. Mixed

Mixed тип создан для того, чтобы принимать любые значения.
Доступные варианты:

object|resource|array|string|float|int|bool|null

Поскольку PHP - язык с динамической типизацией, то мы можем присваивать одной и той же переменной значения разных типов:

```
<?php
	$id = 123;
	echo "<p>id = $id</p>";
	$id = "jhveruuyeru";
	echo "<p>id = $id</p>";
?>
```

Типы данных. Array

Массив в PHP - это структура данных, которая устанавливает соответствие между значением и ключом.

Этот тип оптимизирован в нескольких направлениях, поэтому его можно использовать его как собственно массив, список (вектор), хеш-таблицу, словарь, коллекцию, стек, очередь, а также в качестве других менее популярных структур данных.

Так как значением массива может быть другой массив PHP, можно также создавать деревья и многомерные массивы.

Типы данных. Array

Массивы создаются с помощью конструкции array(), либо [].
Пример:

```
$array = array(0, 1, 2);
$arrayShort = [0, 1, 2, 3];
```

Массивы бывают индексируемые и ассоциативные.
Как уже упоминалось, массив подразумевает под собой соответствие ключ-значение.
Разница в типах массива заключается в типе ключа.

Если ключ не задан, как в примере выше, то он по-умолчанию является положительным целочисленным автоинкрементным значением.

Типы данных. Array

Ассоциативные массивы отличаются от индексируемых тем, что в них ключи имеют конкретное значение. Типом ключей в массивах могут выступать либо целочисленные, либо строковые значения, при этом в качестве значения может выступать любой тип данных.

```
$array = [
    ‘data’ => [
        ‘a’ => 1,
        ‘b’ => 1.1
    ],
    3 => ‘Some string’,
    ‘isTrue’ => true,
    true => false, // вызовет ошибку
];
```

Типы данных. Object

Объект, как экземпляр любого класса имеет базовый тип object, а также тип класса, из которого данный объект создан.

```
<?php
    class Foo {...}

    $foo = new Foo();
    var_dump(is_object($foo)); // true
    var_dump($a instanceof Foo); //true
?>
```

Типы данных. Callable

Callable — это специальный псевдотип данных в PHP, означающий «нечто, что может быть вызвано как функция».
Как будет видно ниже, значения этого псевдотипа могут быть самых разных реальных типов, но всегда есть нечто, что их объединяет — это способность быть использованными в качестве функции.
Анонимные функции можно присваивать переменным и затем вызывать с помощью этих переменных.
Анонимная функция может быть передана в качестве аргумента в другую функцию или быть возвращена функцией с помощью оператора return, что вместе с семейством функций вроде array_map или array_reduce позволяет реализовать подход функционального программирования.

Типы данных. Callable

В PHP существует специальный системный класс Closure. При создании новой анонимной функции, объект этого класса создается неявным образом.

Стоит заметить, что callable-строка может содержать в себе конструкцию вида 'ClassName::method' — это не возбраняется, такие строки тоже будут callable. Есть важная особенность — скобки списка аргументов в таком случае не пишутся

```php
class Foo {
    1 usage
    public static function bar(): int {
        return 42;
    }
}

$x = 'Foo::bar';

var_dump(is_callable($x)); //true
var_dump( value: 42 === call_user_func($x)); //true
```

Типы данных. Callable

Вторая особенность такой callable строки в том, что невозможно вызвать ее напрямую, с помощью $x(): будет вызвана ошибка вида
«Fatal error: Call to undefined function Foo::bar()»

Для того, чтобы совершать подобные вызовы, существует специальная функция call_user_func(), которая умеет вызывать значения псевдотипа callable, даже если это невозможно с помощью обычного синтаксиса.

```php
class Foo
{
  public static function bar(): int
  {
    return 42;
  }

  public function fooBar(): int
  {
    return 43;
  }

  public function barFoo(int $a): int
  {
    return $a;
  }
}

$bar = new Foo();
$x = 'Foo::bar';
echo call_user_func($x); //42

echo call_user_func(
  static function () {
    return 44;
  }
); //44

call_user_func([$bar, 'fooBar']); //43

call_user_func_array([$bar, 'barFoo'], [40]); //40
```

Типы данных. Resource

Ресурс является специальным типом данных к которому невозможно применить стандартные процедуры обработки. Ресурс как правило содержит служебную информацию, такую как ссылка на внешний источник (например ссылка на соединение с базами данных) или служебную информацию, необходимую для соединения с базой данных. Для создания переменной типа **ресурс** применяются специальные функции.

Для использования ресурсов применяются специализированные функции, такие как mysql_connect() и т.д.

Правила именования переменных

Требования языка PHP

Названия переменных должны не только передавать смысл, но и соответствовать синтаксическим правилам.

- имя переменной должно начинаться со знака $
- следующим символом за $ может быть либо латинская буква, либо знак нижнего подчеркивания
- название может содержать только буквы латинского алфавита, знаки нижнего подчеркивания, цифры.

Правила именования переменных

Общепринятые правила именования

$foo — пример простого имени переменной, но существуют более сложные имена. Довольно часто они составные, то есть включают в себя несколько слов. Например, «имя пользователя». В разных языках применяются разные стили кодирования, и имя переменной будет отличаться.

В именовании переменных можно выделить три основных подхода, которые иногда комбинируют друг с другом. Все эти подходы проявляют себя, когда имя переменной состоит из нескольких слов

Правила именования переменных

- snake_case — для разделения используется подчеркивание. Например: my_super_var. Такой подход обычно применяют для именования полей в базе данных.
- CamelCase — каждое слово в переменной пишется с заглавной буквы. Например: MySuperVar. Данный подход применяется для именования классов, трейтов, интерфейсов.
- lowerCamelCase — каждое слово в переменной пишется с заглавной буквы, кроме первого. Например: mySuperVar.

Правила именования переменных

В PHP используется CamelCase и его вариация lowerCamelCase, при котором первая буква первого слова — строчная. Именно lowerCamelCase применяется для переменных.

Это значит, что имена соединяются друг с другом, при этом все имена кроме первого становятся с заглавной буквы:

```
$userName
```

С тремя и более словами это выглядит так:

```
$myAwesomeVariable
```

Другое общепринятое правило гласит: не используйте транслит для имён, только английский язык.

Приведение типов

Это преобразование значения одного типа в значение другого типа.

Есть два вида приведения типов:

- явное
- неявное

Неявное приведение типа выполняется интерпретатором автоматически, без непосредственного участия программиста. Например, значение будет автоматически преобразовано, если оператор ожидает, числовые операнды:

```
$sum = "3" + 4;  // Строка "3" будет неявно преобразована в число 3

$sum = "3" + "4";  // Строка "4" также будет неявно преобразована в число 4
```

Результаты примеров выше будет составлять 7 (оба), т.к. произойдет неявное преобразование числовых строк в числа.

Приведение типов

Чтобы выполнить явное приведение, нужно указать в круглых скобках имя требуемого типа непосредственно перед приводимым значением или переменной:

```
<?php
    $num = (int)"50";
    echo gettype($num); //integer
?>
```

Приведение типов

Запись имени типа в круглых скобках называется **оператором приведения типа**. Допускаются следующие операторы приведения типов:

- (int), (integer) — приведение к integer.
- (bool), (boolean) — приведение к boolean.
- (float), (double), (real) — приведение к float.
- (string) — приведение к string.
- (array) — приведение к array.
- (object) — приведение к object.
- (unset) — приведение к NULL.

Приведение типов. Приведение к int

При приведении bool в integer, FALSE преобразуется в 0 (нуль), а TRUE — в 1 (единицу).

При приведении float в integer, дробная часть будет округлена в сторону нуля. Значение NULL преобразуется в 0:

```
echo (int) false;   // 0

echo (int) true;    // 1

echo (int) 12.3;    // 12

echo (int) NULL;  // 0
```

Приведение типов. Приведение к int

Строки преобразуются по следующим правилам:

- Если первый символ строки является цифрой, знаком + или -, то интерпретатор переходит к анализу второго символа, если второй символ строки является цифрой, то интерпретатор переходит к анализу третьего символа и так до тех пор, пока не будет обнаружен символ отличный от цифры, после этого интерпретатор возвращает полученное целое число.

- Если строка пустая или первый символ строки не является цифрой, знаком +или -, она преобразуется в 0.

Для других типов поведение преобразования в integer не определено. Поэтому не нужно полагаться на любое наблюдаемое поведение, так как оно может измениться без предупреждения.

Приведение типов. Приведение к bool

Следующие значения в результате преобразования дают значение FALSE:

- само значение FALSE
- 0 и 0.0 (нуль)
- пустая строка, и строка "0"
- массив без элементов
- NULL
- Объекты SimpleXML, созданные из пустых тегов

Все остальные значения при преобразовании дают в результате значение TRUE.

В любом контексте, когда интерпретатор PHP ожидает получить булево значение, ложные значения интерпретируются как FALSE, а истинные значения — как TRUE.

Приведение типов. Приведение к string

Булево значение TRUE преобразуется в строку "1", а значение FALSE преобразуется в "" (пустую строку).

Целое число или число с плавающей точкой преобразуется в строку, состоящую из цифр числа.

Массивы всегда преобразуются в строку "Array".

NULL всегда преобразуется в пустую строку.

Resource всегда преобразуется в строку вида "Resource id #1".

Для преобразования объекта в строку, объект должен иметь метод __toString(). Если объект не имеет метод __toString(), то в результате преобразования будет вызвана фатальная ошибка.

Операторы. Арифметические операторы

+$x		Преобразование $x в int или float
-$x	отрицание	Отрицание $x.
$x + $y	сложение	Сумма $x и $y.
$x - $y	вычитание	Разница $x и $y.
$x * $y	умножение	Произведение $x и $y.
$x / $y	деление	Частное от $x и $y.
$x % $y	значение по модулю	Остаток от деления $x на $y.

Арифметические операторы выполняют вычисление между двумя или более числовыми значениями.

Один из упомянутых выше операторов — это оператор по модулю.

```php
<?php
    echo (7 % 3); // output 1
    echo (7 % -3); // outputs 1
    echo (-7 % 3); // outputs -1
    echo (-7 % -3); // outputs -1
?>
```

Операторы. Операторы присваивания

$x += $y	$z = $x + $y	сложение
$x -= $y	$z = $x - $y	вычитание
$x *= $y	$z = $x * $y	умножение
$x /= $y	$z = $x / $y	деление
$x %= $ y	$z = $x % $y	модуль
$x **= $y	$z = $x ** $y	возведение в степень

Операторы присваивания используются для присвоения значения переменной.

В языке есть возможность присвоить значение по ссылке, используя операторы присваивания.

В этом случае фактическое значение не присваивается, а адрес значения присваивается переменной, где ее значения хранятся в памяти.

Операторы. Операторы сравнения

$x == $y	равно	истина, если значения x и y совпадают после преобразования типов
$x === $y	строгое равенство	истина, если и типы и значения x и y совпадают
$x != $y	не равно	истина, если значения x и y не совпадают после преобразования типов
$x <> $y	не равно	истина, если значения x и y не совпадают после преобразования типов

$x !== $y	Строго не равно	истина, если и типы и значения x и y не совпадают
$x < $y	Меньше	истина, если *$x* меньше, чем *$y*.
$x > $y	Больше	истина, если *$x* больше, чем *$y*.
$x <= $y	Меньше или равно	истина, если *$x* меньше или равно *$y*.
$x >= $y	Больше или равно	истина, если *$x* больше или равно *$y*.
$x <=> $y	Космический корабль	Целое число меньше, равно или больше нуля, когда *$x* меньше, равно или больше *$y*, Соответственно.

Операторы сравнения выполняют сравнение между левым и правым значениями.

В предыдущих версиях PHP, если строка сравнивается с числовым значением или числовой строкой, результат оказывается неожиданным, поскольку строка сначала преобразуется в число.

Операторы. Операторы инкремента\декремента

Операторы инкремента и декремента соответственно увеличивают или уменьшают значение на единицу. Они работают только с числами и строками. Операторы инкремента/декремента не работают ни с булевыми переменными, ни с массивами.

```php
<?php
    $x = 5;
    echo ++$x; // outputs 6
    echo $x++; // outputs 6
    echo $x; // outputs 7
?>
```

```php
<?php
    $x = 5;
    echo - -$x; // outputs 4
    echo $x- -; // outputs 4
    echo $x; // outputs 3
?>
```

Операторы. Логические операторы

$x and $y	And	истина, если оба *$x* и *$y* равны true.
$x or $y	Or	истина, если либо *$x* or *$y* равны true.
$x xor $y	Xor	истина, если либо *$x* or *$y* равны true, но не оба.
!$x	Not	истина, если *$x* не равны true.
$x && $y	And	истина, если оба *$x* и *$y* равны true.
$ x || $ y	Or	истина, если либо *$x* or *$y* равны true.

Логические операторы выводят либо истину, либо ложь.

Логические операторы можно использовать в виде цепочек и производить операции более, чем над двумя операндами.

При подобном подходе важным является приоритетность операторов. Наибольший приоритет дает использование круглых скобок

Операторы. Строковые операторы

В PHP есть два строковых оператора.

- Операторы конкатенации (.)
- Операторы присваивания конкатенации (.=)

Оператор конкатенации соединяет левую строку с правой строкой. Оператор конкатенационного присваивания соединяет правый аргумент с аргументом слева.

```
<?php
	$x = "Hey ";
	$y = $x . "PHP!"; // now $y contains "Hey PHP!"
	$x = "Hey ";
	$y .= "PHP!"; // now $y contains "Hey PHP!"
?>
```

Операторы. Оператор условного присваивания

В PHP есть три оператора условного присваивания:

- && логический оператор И
- || логический оператор ИЛИ
- ?: *тернарный оператор*.

Тернарный оператор - это краткая форма оператора if..else. Он возвращает значение в зависимости от условия.

```
<?php
    $x = $a ? $a : $b;
    $x = $a ?: $b; // идентично
?>
```

Операторы. Оператор Null Coalescing

Аналогичен тернарному, но этот оператор проверяет наличие NULL.

Оператор применим, когда, например, необходимо проверить значение в массиве и взять значение по-умолчанию, если значение в массиве пустое или не существует

```
<?php
    $x = $var1 ?? $var2 : $var3;
    $y = $arr1[‘someKey’] ?? $defaultVarValue;
?>
```

Операторы. Условные операторы

Для операторов условий в PHP операнды всегда приводятся к типу boolean. В случае, если это значение *true*, то считается, что условие выполнено, а в случае, если *false* – то условие не выполнено. В зависимости от того, выполнено ли условие, будут совершены или не совершены какие-либо действия.

Условный оператор представляет из себя конструкция, состоящую из обязательного оператора if и комбинации необязательных операторов else/elseif.

В общем случае конструкция выглядит следующим образом

Операторы. Условные операторы

```php
<?php

if ($condition) {
    // код, который нужно выполнить, если условие выполнено;
} else {
    // код, который нужно выполнить, если условие не выполнено;
}

// Пример №1:
$x = 20;

if ($x > 10 && $x < 20) {
    echo 'Условие выполнено';
} else {
    echo 'Условие не выполнено';
}
```

Операторы. Условные операторы

```php
//Пример №2:
$x = 20;

if ($x > 10 && $x < 20) {
   echo '$x больше 10 и меньше 20';
} elseif ($x === 20) {
   echo '$x равен 20';
} else {
   echo '$x за рамками условия';
}
```

Примечание: количество конструкций elseif не ограничено, но стоит помнить о том, что каждое ветвление - затраты ресурсов

Операторы. Условный оператор Switch

Помимо конструкции if-else есть ещё один оператор условия switch.

В фигурных скобках перечисляются операторы *case*, после которых указывается значение, с которым сравнивается значение операнда *switch*.
Сравнение происходит нестрогое, как при использовании оператора ==.
Если условие выполнилось, то выполняется код, указанный после двоеточия.

Если же ни одно из условий не выполнилось, то выполняется код из секции *default*, которой в общем-то может и не быть, и тогда ничего выполняться не будет.

Стоит отметить, что внутри каждой секции *case*, в конце прописывается оператор *break*. Это делается для того, чтобы после выполнения кода в случае выполнения условия не продолжилась проверка условий.

Операторы. If vs Switch vs Match

```php
<?php

$x = 1;

switch ($x) {
    case 1:
        echo 'Число равно 1';
        break;
    case 2:
        echo 'Число равно 2';
        break;
    default:
        echo 'Число не равно ни 1, ни 2';
        break;
}
```

```php
<?php

$x = 1;

if ($x == 1) {
    echo 'Число равно 1';
} elseif ($x == 2) {
    echo 'Число равно 2';
} else {
    echo 'Число не равно ни 1, ни 2';
}
```

```php
<?php

$x = 1;

echo match ($x) {
    1 => 'Число равно 1',
    2 => 'Число равно 2',
    default => 'Число не равно ни 1, ни 2',
};
```

Операторы. If vs Switch vs Match

Начиная с версии 8.0 в PHP была добавлена поддержка другой, похожей конструкции - match.
Она позволяет оптимизировать конструкцию switch. Конструкция match также принимает некоторое выражение и сравнивает его с набором значений.

В отличие от switch, конструкция match возвращает некоторый результат. Поэтому после каждого сравнимого значения ставится оператор =>, после которого идет возвращаемый результат.

Другое важное отличие конструкции switch от match состоит в том, что switch сравнивает только значение, но не учитывает тип выражения. Тогда как match также учитывает тип сравниваемого выражения.

Операторы. Циклы

В PHP существует четыре варианта операторов циклов.

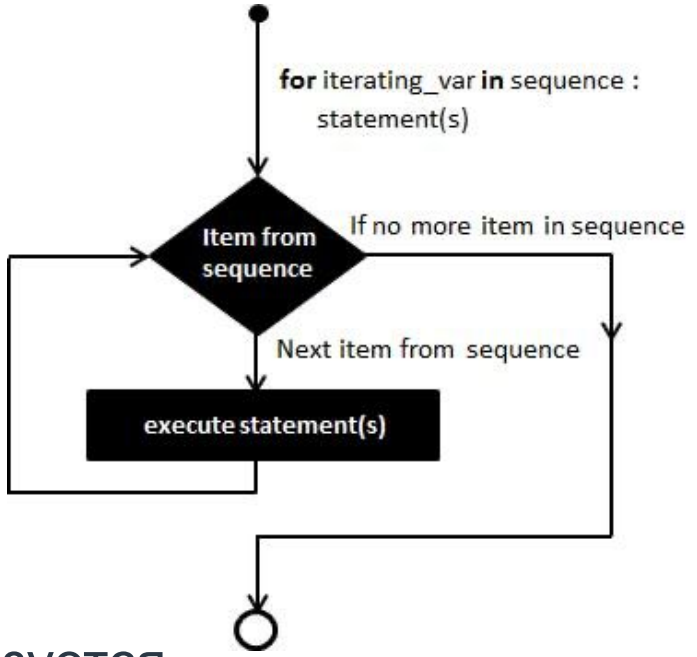
- for — перебирает через цикл блок кода определенное количество раз.
- while – перебирает через цикл блок кода, если и до тех пор, пока указанное условие является истинным.
- do … while — перебирает через цикл блок кода, а затем повторяет цикл, пока выполняется специальное условие.
- foreach – перебирает через цикл каждый элемент в массиве.

Операторы. Циклы. Оператор for

Оператор for используется, когда вы знаете, сколько раз необходимо выполнить оператор или блок операторов.
Синтаксис:

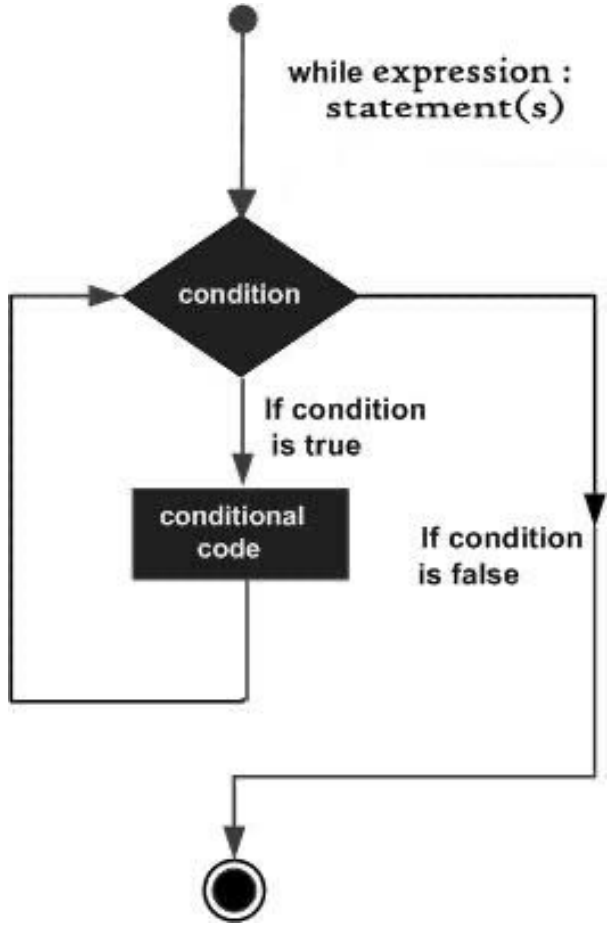
```php
<?php

for ($i = 0; $i < 20; $i++) {
    echo $i;
}
```



Для установки начального значения счетчика используется инициализатор. Для этой цели может быть объявлена переменная, ее традиционно называют $i.

Операторы. Циклы. Оператор while



Оператор while выполняет блок кода, если и до тех пор, пока условное выражение истинно. Если условное выражение истинно, тогда будет выполняться блок кода. После выполнения кода условное выражение снова будет оценено, и цикл будет продолжаться до тех пор, пока условное выражение не окажется ложным.

Синтаксис:

```php
<?php

$i = 0;
while ($i++ < 20) {
    echo $i;
}
```

Операторы. Циклы. Оператор do…while

Оператор do … while выполняет блок кода хотя бы один раз — он повторяет цикл, пока условие выполняется.

Синтаксис:

```php
<?php

$i = 0;
do {
    echo $i;
} while ($i++ < 20);
```

Операторы. Циклы. Оператор foreach

Оператор foreach используется для преобразования массивов через цикл. При каждой итерации текущему элементу массива присваивается значение $, а указатель массива перемещается на одну позицию, и при следующем проходе будет обрабатываться следующий элемент.

Синтаксис:

```php
<?php

$heightList = [
    'Petrov' => 1.74,
    'Ivanov' => 1.82,
    'Zorin' => 1.65,
];
foreach ($heightList as $key => $value) {
    echo 'Рост студента ' . $key . ' составляет ' . $value . ' м';
}
```

Операторы. Break и continue

Ключевое слово break используется для преждевременного прекращения выполнения цикла. Оператор break находится внутри блока операторов. С его помощью можно остановить цикл, когда это необходимо.

```php
<?php

while (true) { //бесконечный цикл
    $currentTime = getCurrentTime();
    if ($currentTime === 'обед') {
        continue;
    }
    if ($currentTime === 'пора домой') {
        break;
    }
}
```

Ключевое слово continue используется для остановки текущей итерации цикла, но оно не завершает цикл. Как и оператор break, оператор continue находится внутри блока операторов, содержащих код, который выполняется циклом. Если при проходе кода, встречается оператор continue , остальная часть кода цикла пропускается, и начинается следующий проход цикла.

Функции. Основные понятия

Функция — это именованный фрагмент кода.

В обычную переменную можно записать число, строку или массив, а затем получить его обратно, обратившись к значению по имени переменной. Функции устроены похожим образом, только вместо строки или числа в ней хранится блок кода, который вызывается при использовании этой «переменной».

Функции помогают повторно использовать код, который нужен во многих местах программы.

Функции нужны, чтобы не переписывать один и тот же код много раз.

Функции. Основные понятия

Функции бывают встроенные и пользовательские.

Встроенные функции за вас уже написали создатели языка, и вы можете просто брать их и использовать. В PHP существуют тысячи готовых функций на все случаи жизни — например, `sort()` для сортировки массивов, `print()` для вывода строк на экран или функции для работы с базами данных.

Пользовательские функции программисты пишут сами — например, чтобы проверить данные пассажира по номеру авиабилета или определить, високосный ли сейчас год. Эти функции, как правило, используются только внутри одного проекта, но бывают исключения — и такие функции выносят в библиотеки.

Функции. Аргументы и область видимости

Функция — это по сути программа в программе. Это значит, что внутри неё не будут доступны переменные, которые определялись за её пределами. Чтобы передать внутрь функции информацию извне, нужно использовать *аргументы функции*.

Аргументы функции — это переменные, которые передаются в функцию для обработки. Аргументов может быть несколько.

Пример. Мы хотим показывать на сайте, является ли выбранный пользователем год високосным. Напишем функцию, в которую будем передавать год. В результате работы функции мы должны получить `true`, если год високосный, и `false` — если нет.

Функции. Аргументы и область видимости

В такой функции необходим лишь один аргумент - номер выбранного года `$year`.

```php
<?php

function isLeapYear(int $year): bool
{
    return $year % 4 === 0;
}
```

Важно отметить, что хорошим стилем является указание типов аргументов, а также - возвращаемого значения. Если функция не возвращает ничего, стоит в качестве типа возвращаемого значения указать void.

Функция «не видит» переменные, которые создаются за её границами. Поэтому переменные в функцию нужно передавать явно — то есть, через аргументы.

Верно и обратное — переменные, определенные внутри функции, не будут доступны извне. Такие переменные называются локальными, потому что они локальны по отношению к функции.

Функции. Аргументы по-умолчанию

Иногда аргументы, передаваемые в функцию, часто имеют одно и то же значение. Например, чаще всего при использовании функции определения високосного года, пользователю было бы интересно узнать информацию о текущем годе. Поэтому, в качестве аргумента чаще всего передавался бы текущий год. В таких случаях стоит использовать аргументы по-умолчанию.

```php
<?php

function isLeapYear(int $year = 2023): bool
{
    return $year % 4 === 0;
}

echo isLeapYear();
```