PHP Programming

By Jenish Borah, Faculty

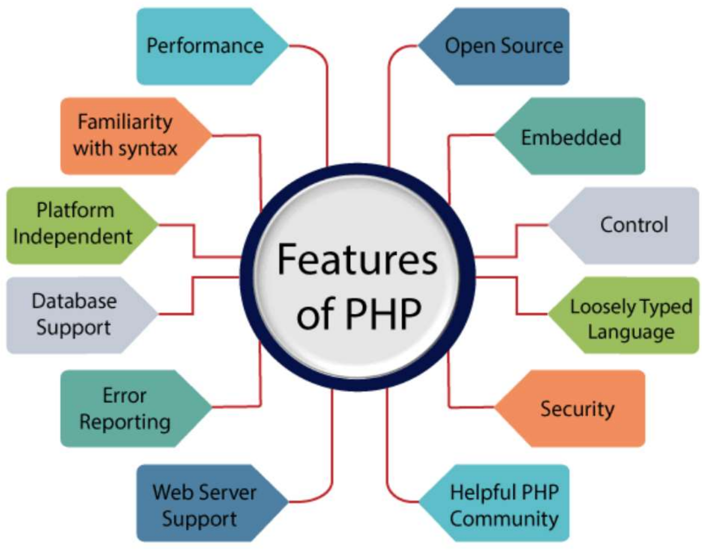
B.sc IT department, B borooah College

PHP introduction:

PHP is an open-source, server-side programming language that can be used to create websites, applications, customer relationship management systems and more. It is a widely-used general-purpose language that can be embedded into HTML.

- With PHP you can store data on the server as files or in databases like MySQL.
- With PHP, you can write code once and use it everywhere. Remember, you want DRY code (Don't Repeat Yourself).
- PHP lets you build dynamic webpages that respond to input from users.
- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download and use.
- PHP is easy to learn and runs efficiently on the server side.

PHP Features:



- Performance: PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.

- Open Source: PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

- Familiarity with syntax: PHP has easily understandable syntax. Programmers are comfortable coding with it.
- Embedded: PHP code can be easily embedded within HTML tags and script.
- Platform Independent: PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
- Database Support: PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.
- Error Reporting -PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.
- Loosely Typed Language: PHP allows us to use a variable without declaring its data type. It will be taken automatically at the time of execution based on the type of data it contains on its value.
- Web servers Support: PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.
- Security: PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.
- Control: Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

PHP inventions:

PHP stands for '**PHP: Hypertext Preprocessor'**, with the original PHP within this standing for 'Personal Home Page'. The acronym has changed as the language developed since its launch in 1994 to more accurately reflect its nature. Since its release, there have been 8 versions of PHP, as of 2024, with version 8.3 currently a popular choice among those using the language on their websites but on todays date version 8.2 is more stable.

The journey began in 1994 with the birth of PHP/FI (Personal Home Page/Forms Interpreter), created by **Rasmus Lerdorf**, who released its first version in **1995**. Since then, PHP has evolved significantly, with PHP 3 introducing advanced features like the Zend Engine.

PHP version:

PHP version history serves as a roadmap highlighting the progress and improvements made to the language over time. Each new release brings enhanced features, bug fixes, security patches, and performance optimizations.

Ass1: Write about the timeline of PHP versions including features/addition of Each version. (Up to version 8.3)

Ass2: State the scope of PHP in present days and in near future.

Important Tools and Software Requirements:

Development Tools: PHPStrom , Eclipse, Sublime Text, Cloud 9, NetBeans, Code Lobster etc.

Software Requirements:

- **PHP interpreter:** PHP can be installed as an Apache module or as a standalone Fast CGI process. The minimum PHP version required depends on the specific PHP application being used.
- **Web server:** PHP can be used with several web servers, including Apache, Nginx, and Microsoft IIS.

- Operating system: PHP can be installed on several operating systems, including Windows, Linux, and macOS.
- **Database:** Many PHP applications require a database, such as MySQL, PostgreSQL, or SQLite.

Basic Syntax:

A PHP script starts with <?php and ends with ?>. A PHP script can be placed anywhere in the document.

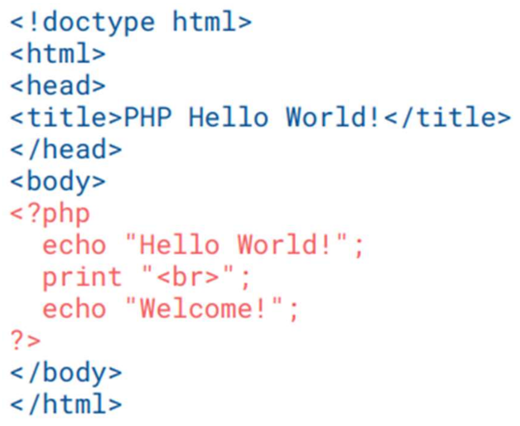
```
<?php

echo “Hello World!”;

print “<br>”;

echo “Welcome”;

?>
```



```
<!doctype html>
<html>
<head>
<title>PHP Hello World!</title>
</head>
<body>
<?php
  echo "Hello World!";
  print "<br>";
  echo "Welcome!";
?>
</body>
</html>
```

✓ How PHP looks on the server

PHP variables and constants:

Variables:

A variable is a place to store values. When you first create a variable, it does not have a value (it is null). You can set a value for a variable. Variables can hold different types of information, like words, numbers, and collections of data. The value of a variable can change over time. In PHP, you write a variable with **a dollar sign ($)** followed by the name of the variable

Example $name = “Jenish Borah” ; //String variable

$age = 23; // Numeric variable (You can store float value also)

$subjects = array(‘AI’, ‘PHP’, ‘CONM’); // Arrays

The $$var (double dollar) is a reference variable that stores the value of the $variable inside it.

Variable-naming rules:

When creating PHP variables, you must follow these four rules:

1. Variable names must start with a letter of the alphabet or the _ (underscore) character.

2. Variable names can contain only the characters a-z, A-Z, 0-9, and _ (underscore).

3. Variable names may not contain spaces. If a variable must comprise more than one word, it should be separated with the _ (underscore) character (e.g., $user_name).

4. Variable names are case-sensitive. The variable $High_Score is not the same as the variable $high_score.

Constants:

PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:

- **define ()** function
- Using **const** keyword.

, PHP constants should be defined in uppercase letters.

Using define () function

Syntax**:** define(name, value, case-insensitive)

1. name: It specifies the constant name.

2. value: It specifies the constant value.

3. case-insensitive: Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default

Example

```
<?php

define("NOTICE", "No one will be allowed to sit on internal exam if attendance is less than 75%");

echo NOTICE;

?>
```

Output: No one will be allowed to sit on internal exam if attendance is less than 75%

```
<?php

define("NOTICE", "No one will be allowed to sit on internal exam if attendance is less than 75%", true);

echo notice

?>
```

Output: No one will be allowed to sit on internal exam if attendance is less than 75%

Using Const Keyword

PHP introduced a keyword const to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function. The constant defined using const keyword are case-sensitive.

```
<?php

Const NOTICE = "No one will be allowed to sit on internal exam if attendance is less than 75%";

echo NOTICE;

?>
```

Output: No one will be allowed to sit on internal exam if attendance is less than 75%

Ass3: Write the differences between Variables and constants.

Types of Data in PHP:

Data Types: Data types defines the type of values a variable can store.

PHP supports the following data types:
· String
· Integer
· Float/Double
· Boolean
· Array
· Object
· NULL

Ass4: Explain the types of data in PHP.

Expressions:

Expressions are the very essential part of any programming language, that allows us to perform any calculations, make decisions, and manipulate data.

An expression is a combination of values, variables, operators, and function calls that can be evaluated to produce a value.

In PHP, an expression is a combination of values, variables, operators, and function calls that can be evaluated to produce a single value. These elements come together to form a logical operation, and the result of this operation is the output of the expression.

Components of an Expression:

1. **Values:** These are the raw data elements in programming. Examples include numbers (3, 10.2), strings (‘Hello World'), and booleans (true, false).

2. **Variables:** Variables are containers for holding values. They have names and can store different values at different points in program’s execution.

3. **Operators:** Operators are symbols or keywords that perform operations on values and variables. Examples include arithmetic operators (+, -, * /), comparison operators (==, !=, <, >) and logical operators (&&, ||).

4. **Function Calls:** Functions are reusable blocks of code that perform specific tasks. When you call a function, it can return a value. Function calls typically involve passing arguments enclosed in parentheses.

Examples of Expressions

1. **Arithmetic Expression:**

```
$result = 5 + 3;
```

Here, the expression 5+3 adds the values 5and 3, resulting in the value 8.

2. String Concatenation Expression:

```
$name      = 'John';
$greetings = 'Hello, ' . $name;
```

The expression "Hello, " . $name concatenates the string "Hello, " with the value store in the variable $name .

3. Function Call Expression:

```
$length = strlen('Hello');
```

The expression calls the `strlen` function to get the length of the string `'Hello'` , producing the value `5` .

4. Comparison Expression:

```
$age = 30;
$isAdult = $age >= 18;
```

The expression `$age >= 18` evaluates to `true` or `false` based on whether the value of `$age` is greater than or equal to `18`.

5. Ternary Operator Expression:

```
$number = 7;
$isEven = ($number % 2 == 0) ? true : false;
```

The expression uses the ternary operator to determine if. `$number` is even, producing `true` or `false` accordingly.

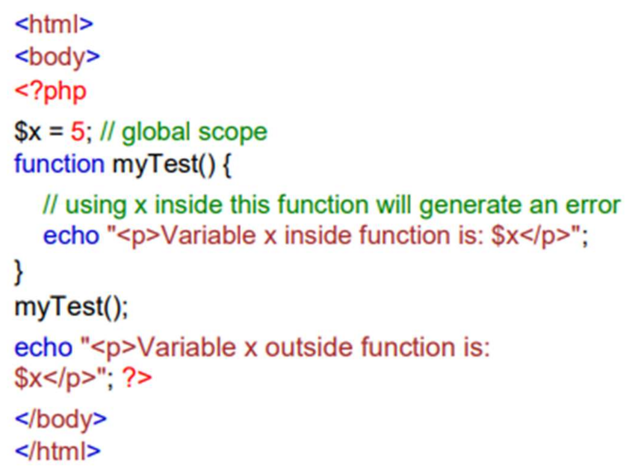
Scope of a Variable:

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:

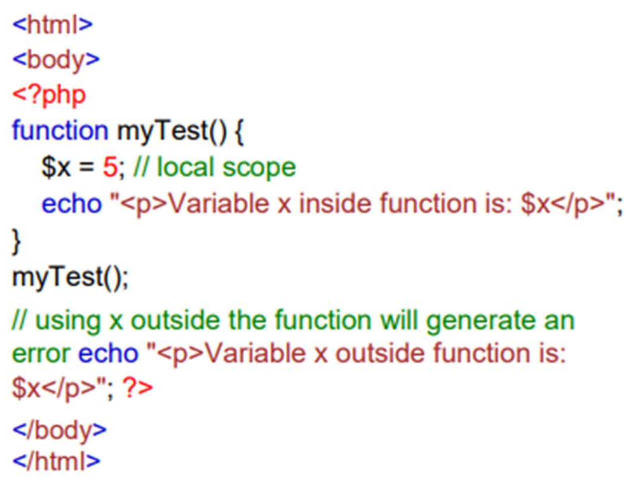


```
<html>
<body>
<?php
$x = 5; // global scope
function myTest() {
   // using x inside this function will generate an error
   echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is:
$x</p>"; ?>
</body>
</html>
```

OUTPUT: Variable x inside function is:

Variable x outside function is: 5

A variable declared within a function has a **LOCAL SCOPE** and can only be accessed within that function:

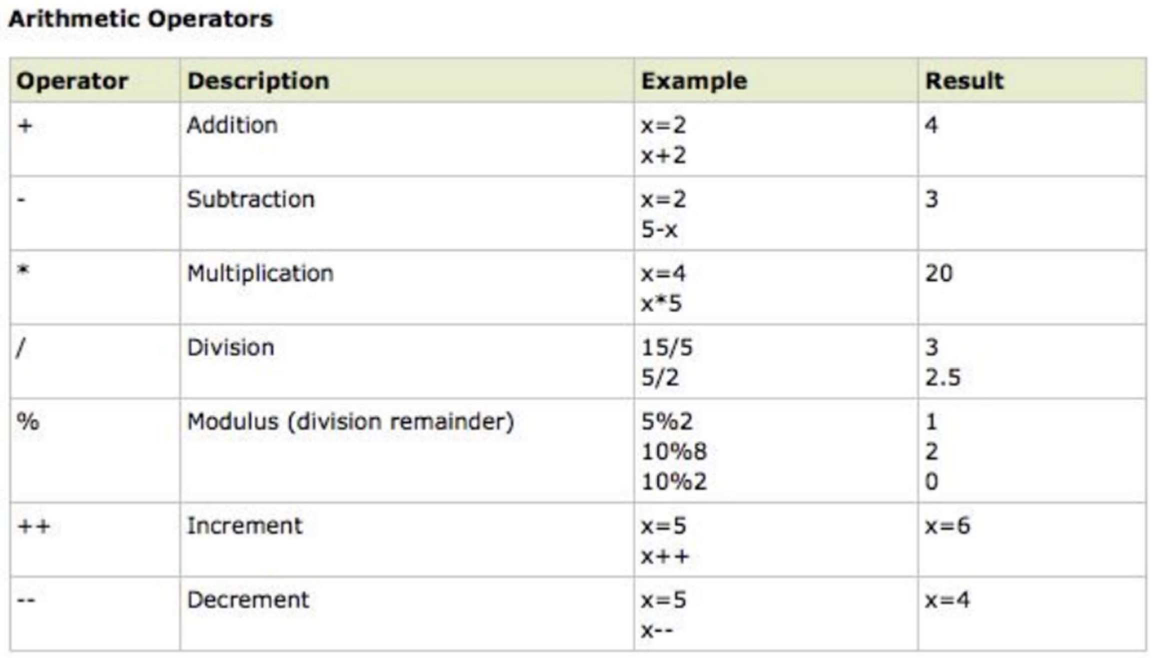


```
<html>
<body>
<?php
function myTest() {
   $x = 5; // local scope
   echo "<p>Variable x inside function is: $x</p>";
}
myTest();
// using x outside the function will generate an
error echo "<p>Variable x outside function is:
$x</p>"; ?>
</body>
</html>
```

output: Variable x inside function is: 5

Variable x outside function is:

<u>PHP operators</u>:



Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

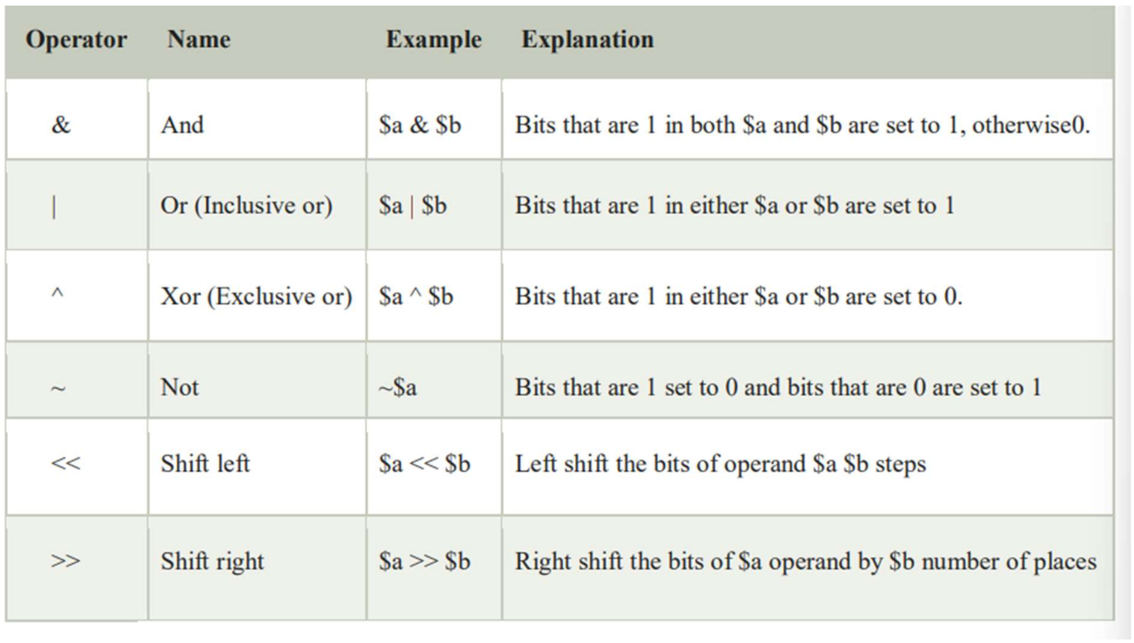
Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

Bitwise Operators



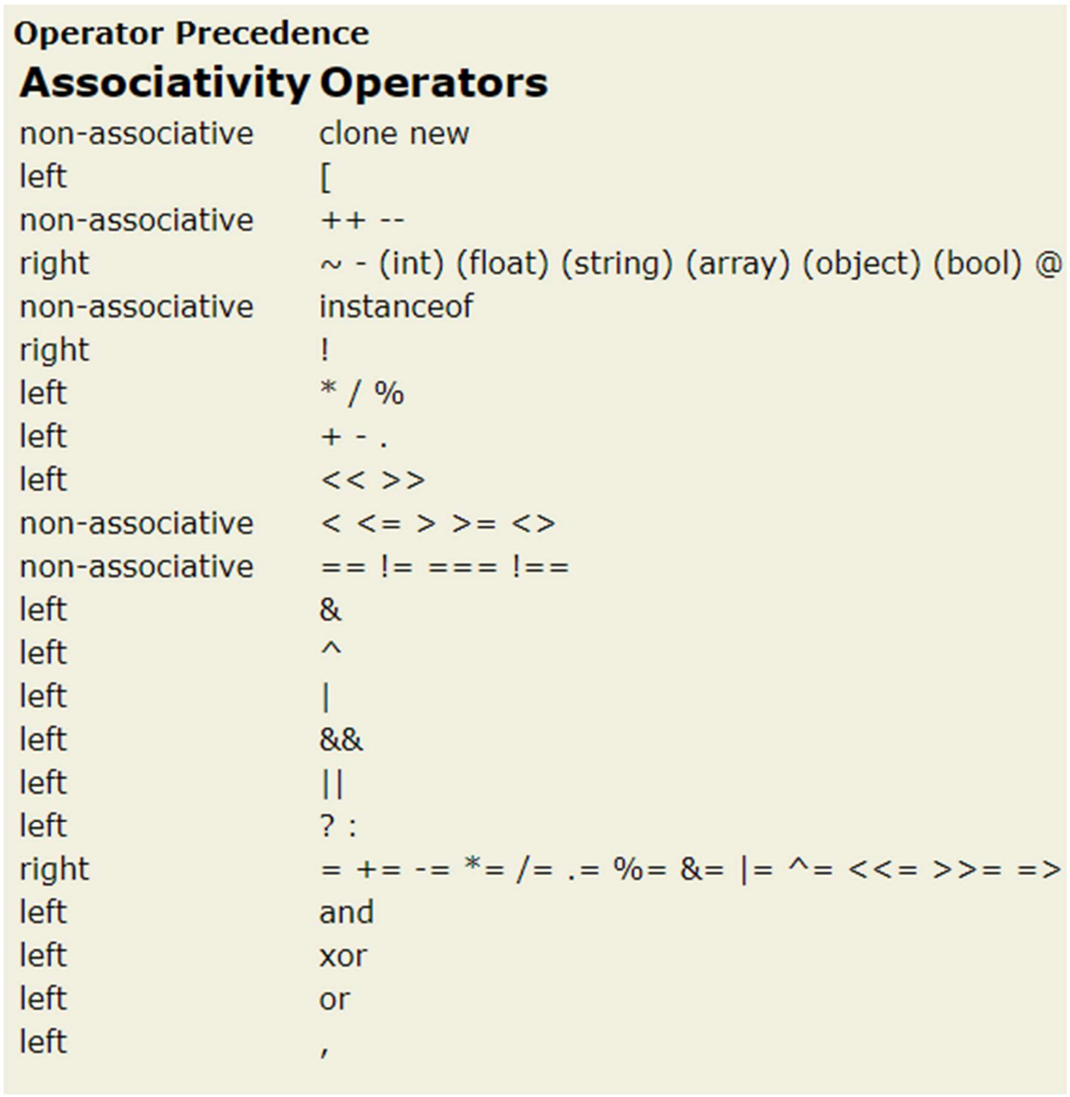
Operator	Name	Example	Explanation
&	And	$a & $b	Bits that are 1 in both $a and $b are set to 1, otherwise0.
|	Or (Inclusive or)	$a | $b	Bits that are 1 in either $a or $b are set to 1
^	Xor (Exclusive or)	$a ^ $b	Bits that are 1 in either $a or $b are set to 0.
~	Not	~$a	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	$a << $b	Left shift the bits of operand $a $b steps
>>	Shift right	$a >> $b	Right shift the bits of $a operand by $b number of places

Ass5: Write about other operators

PHP operator precedence and associativity:

The precedence of an operator specifies how "tightly" it binds two expressions together. For example, in the expression 1 + 5 * 3, the answer is 16 and not 18 because the multiplication ("*") operator has a higher precedence than the addition ("+") operator. Parentheses may be used to force precedence, if necessary. For instance: (1 + 5) * 3 evaluates to 18. If operator precedence is equal, left to right associativity is used.

The following table lists the precedence of operators with the highest-precedence operators listed at the top of the table. Operators on the same line have equal precedence, in which case their associativity decides which order to evaluate them in.



Operator Precedence

Associativity	Operators
non-associative	clone new
left	[
non-associative	++ --
right	~ - (int) (float) (string) (array) (object) (bool) @
non-associative	instanceof
right	!
left	* / %
left	+ - .
left	<< >>
non-associative	< <= > >= <>
non-associative	== != === !==
left	&
left	^
left	|
left	&&
left	||
left	? :
right	= += -= *= /= .= %= &= |= ^= <<= >>= =>
left	and
left	xor
left	or
left	,

Left associativity means that the expression is evaluated from left to right, right associativity means the opposite.

Thank You

Jenish Borah