PHP Hypertext Preprocessor SL Review
LIJIA XU (1/1)

```php
<?php                    <script language="php">
echo "word!";            <|script>
?>
```
language constant

dynamic page
manage files data base
collecting form data
control user-access
encrypt data....

* each php statement must end with ;

// comment
/* multiline comments */

var types
- String
- Integer
- Float
- Boolean
- Array
- Object
- NULL
- Resource

$name = 'John'
- must start letter or _
- only letter number or _
- Case sensitive

define(name, value) case-insensitive
define a const

```php
$string1 = "Hello!";
$string2 = "good";
$string1 . $string2;
//Hello! good
```

Auto convert:
```php
$str = "10";
$int = 20;
$sum = $str + $int;
// 30
```

Variable variables => var to specify another var name
```php
$a = 'hello';
$hello = 'Hi!';
echo $$a;  // Hi!
```

```php
$a = 2; $b = $a++ //$a=3, $b=2
$a = 2; $b = ++$a; // Hi!
//$a=3, $b=3
```

x === y // x equal y and same type

and, or, xor, && , ||, !
```
$x xor $y //either $x or $y is true, but NOT both true
```
```
x | y => x * | y
x% = y => x=x%y
```

use global to access
```php
$name = 'Bob';
function getName(){
  global $name;
  echo $name;
}
// output 'Bob'
getName();  use one
```

POST => invisible to users, no limit
GET => visible to everyone, limit 2k chars
* NEVER use GET for sensitive Info

session_start() => must before <!DOCTYPE>
```php
<?php
session_start()
$_SESSION['color'] = "red";
$_SESSION['name'] = "John";
?>
```
Cookies => used to identify user
a small file that the server embeds on the user's computer

setcookie(name, value, expire, path, domain, secure, httponly)

the setcookie() function must before html
```php
<?php
$value = "John"
setcookie("user", $value, time()+(84400*30), '/')
if(!isset($_COOKIE['user'])){
  echo "value is: ". $_COOKIE['user'];
} ?>
```
Never store sensitive Into in cookie

```php
$myfile = fopen ("file.txt", "w")
```

r: read only
w: write only (erase or new)
a: write only, append
x: new for write only
rt: read/write
wt: read/write (erase or new)
at: read/write (new if no)
xt: create new for read/write

```php
<?php $myfile = fopen("name.txt", "w");
$txt = "John \n";
fwrite($myfile, $txt);
$txt = "David \n";
fwrite($myfile, $txt);
fclose($myfile); ?>
```

$names = array("A", "B", "C") /* can have diff types in array */
echo $names[1] // "B"

Associate Array: $names = array("A"=>1, "B"=>2)
echo $names["B"] // outputs 2

multi. D array

```php
$x = 10;
$y = 20;
if($x > $y){
  echo $x;
} else {
  echo $y;
}  // outputs 20
```

can add elseif
```php
$z = 1;
while ($z < 7){
  echo $z;
  $z++;
}
```
do{ //run >= once
  //code here
} while(condition)

for(init; test; incre){ }

foreach only works for arrays => include associate arrays
```php
foreach ($array as $value){ }
foreach($array as $key => $value){ }
```
continue => skipping over what remains of the current loop iterat

```php
for($i=0; $i<10; $i++){
  if($i%2==0){
    continue; //outputs
  }
  echo $i.' '; // 13579
}
```

include and required to php

```php
function fun_name(){ }
fun_name() // call function
return // sends a value
```

Predefined Variables
echo $_SERVER['SCRIPT_NAME']
* can be accessed anywhere in code

$_GET and $_POST => collect data was entered into a form

$_Get and $_Post is to collect data

```html
<form action="first.php" method="post">
<p>Name: <input type="text" name="name"/></p>
<p><input type="submit" name="submit" value="Submit"/></p>
</form>
```

```php
$day = 'Tue'
switch ($day){
  case 'Mon':
    echo "first day";
    break;
  case 'Tue':
  case "wed":
  case "thu":
    echo "workday";
    break;
  case "Fri":
    echo "Friday";
    break;
  default:
    echo "weekend";
}  // outputs "workday"
```

without break; will keep running!
```php
<?php
echo $_POST['name'];
?>
```

fclose($filename); close an open file and return a Boolean (success or fail close)

Appending to a file:
```php
<?php
if(!isset($_POST['text'])){
  $name = $_POST['text'];
  $handle = fopen('name.txt', 'a');
  fwrite($handle, $name."\n");
  fclose($handle);
} ?>
```

```php
$read = file('name.txt');
$count = count($read);
$i = 1;
foreach ($read as $line){
  echo $line;
  if($i < $count){
    echo ', ';
  } // not print ', ' last line
  $i++;
}
// $read is Array
```

```html
<form method="post">
Name: <input type="text" name="text"/>
<input type="submit" name="submit"/>
</form>
```

$stack = array('A','B')
Add array: array_push($stack, "C", "D");
print_r($stack);

OOP
$this // a pseudo-variable

```php
class Dog{
  public $legs = 4;
  public function display(){
    echo $this->legs;
  }
}
$d1 = new Dog();
$d1->display(); //4
$d2 = new Dog();
$d2->legs = 2; $d2->display() //2
```

__construct() // auto called when new object instantiated

__destruct() explicitly / unset($p)

```php
class Person{
  public $name;
  public $age;
  public function __construct($name, $age){
    $this->name = $name;
    $this->age = $age;
  } //func
} //class
$p = new Person("Bob", 40);
```

protected // with inheritance
Private // only internally in class
Interface // list of methods must implement

```php
class Animal{
  public $name;
  public function hi(){
    echo "Hi";
  }
}
class Dog extends Animal{ } // for the class
$d = new Dog();
$d->hi();
```

```php
<?php
interface AnimalInterface{
  public function makeSound();
}
class Dog implements AnimalInterface{
  public function makeSound(){
    echo "wang!";
  }
}
```

abstract functions can only appear in abstract abstract class
* A class inheriting from an abstract class must implement all the abstract methods

static properties or methods can be accessed without creating an object

```php
<?php
abstract class Fruit{
  private $color;
  abstract public function eat();
  public function setColor($c){
    $this->color = $c;
  }
}
class Apple extends Fruit{
  public function eat(){
    echo "eating";
  }
}
$obj = new Apple();
$obj->eat(); ?>
```

scope resolution operator ::
```php
<?php
class myClass{
  static $myP = 25;
}
echo myClass::$myP;
?>
```

final keyword defines methods can Not be overridden in child classes, also defines classes can Not be inherited