

Arrays, Functions and Graphics

Unit - II

2.1 Creating & Manipulating Array

- An array is a special variable, which can hold more than one value at a time.
- In PHP, *array()* function is used to create an array.
- Types of array:
 - ❑ Indexed arrays
 - ❑ Associative arrays
 - ❑ Multidimensional arrays
- *Count()* function is used to get the number of elements of an array.

2.1 Creating & Manipulating Array

- Indexed array:
 - ✓ These are the arrays with a numeric index.
 - ✓ To create an indexed array:
 - ✓ `$array_name = array("value1","value2","value3"..);`
 - or
 - ✓ `$array_name[0] = "value1";`
`$array_name[1] = "value2";`

2.1 Creating & Manipulating Array

- Indexed array:

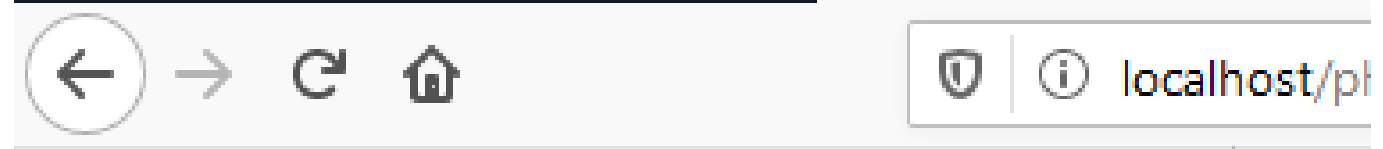
E.g.:

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    $arlength = count($cars);
    for($x = 0; $x < $arlength; $x++) {
        echo $cars[$x];
        echo "<br>";
    }
?>
```

2.1 Creating & Manipulating Array

- Indexed array:

```
<html>
<head><title>
    1D array
</title></head>
<body>
    <?php
        $arr=array('how','are','you','today');
        for($i=0;$i<4;$i++) {
            echo $arr[$i]."<br>";
        }
    ?>
</body>
</html>
```



how
are
you
today

2.1 Creating & Manipulating Array

- Associative array:

- ✓ These are the arrays with a named key as index.

- ✓ To create an associative array:

- ✓ `$array_name = array("key1"=>"value1", "key2"=>"value2", "key3"=>"value3"..);`

or

- ✓ `$array_name['key1'] = "value1";`
`$array_name['key2'] = "value2";`

2.1 Creating & Manipulating Array

- Associative array:

E.g.:

```
<?php
    $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
    foreach($age as $x => $x_value) {
        echo "Key=" . $x . ", Value=" . $x_value;
        echo "<br>";
    }
?>
```

2.1 Creating & Manipulating Array

- Associative array:

```
<html>
```

```
<head><title>
```

1D associative array

```
</title></head>
```

```
<body>
```

```
<?php
```

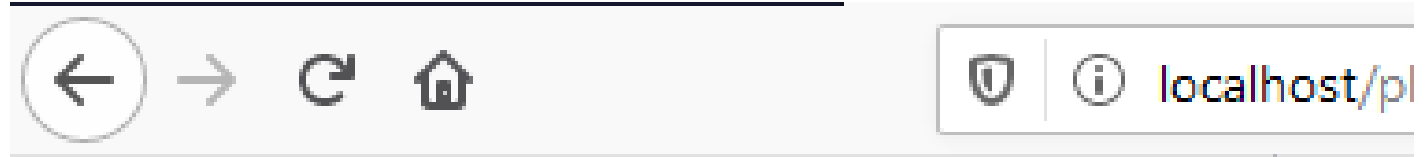
```
$arr=array('a'=>'how','b'=>'are','c'=>'you','d'=>'today');
```

```
foreach($arr as $x=>$x_v){
```

```
echo $x_v."<br>";}
```

```
?>
```

```
</body></html>
```



how
are
you
today

2.1 Creating & Manipulating Array

- Multidimensional array:
 - ✓ These are the arrays containing one or more arrays.
 - ✓ PHP supports multidimensional arrays that are two, three, four, five, or more levels deep.
 - ✓ However, arrays more than three levels deep are hard to manage.

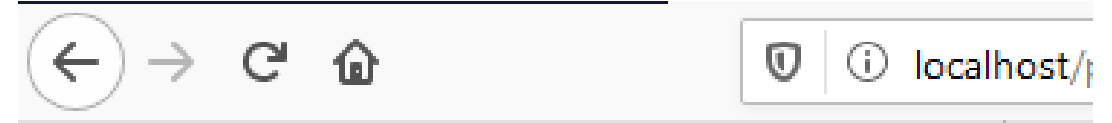
```
✓ $array_name = array(  
    array("val_col1","val_col2",val_col3),  
    array("val_col1","val_col2",val_col3),  
    array("val_col1","val_col2",val_col3),  
    array("val_col1","val_col2",val_col3),  
);
```

✓ *This will create a 2D array with 4 rows and 3 columns.*

2.1 Creating & Manipulating Array

- Multidimensional array:

```
<html>
<head><title>
    Multidimensional Indexed array
</title></head><body>
    <?php
        $arr=array(array(1,2,3),array(4,5,6),array(7,8,9),array(10,11,12));
        for($i=0;$i<4;$i++)
        {for($j=0;$j<3;$j++)
            {echo $arr[$i][$j]." ";
            }echo "<br>";}
    ?>
</body></html>
```



2.1 Creating & Manipulating Array

- Multidimensional array:

<html>

<head><title>

Multidimensional Associative array

</title></head><body>

<?php

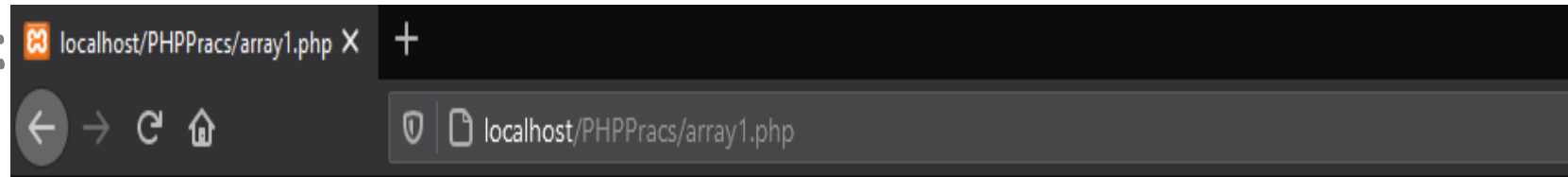
\$sports=array();

\$sports['cricket']=array('India'=>'Dhoni', 'South Africa'=>'Johnty');

\$sports['football']=array('Argentina'=>'Messi', 'Portugal'=>'Ronaldo');

print_r(\$sports);

?></body></html>



Array ([cricket] => Array ([India] => Dhoni [South Africa] => Johnty) [football] => Array ([Argentina] => Messi [Portuguese] => Ronaldo))

2.2 Extracting data from array

- `implode()`:
 - ✓ The `implode()` function returns a string from the elements of an array.

✓ *`implode(separator,array);`*

`separator`: Optional. Specifies what to put between the array elements.

`Array` : Required. The array to join to a string.

2.2 Extracting data from array

- implode():

```
<html>
```

```
<head><title>
```

```
    implode
```

```
</title></head>
```

```
<body>
```

```
<?php
```

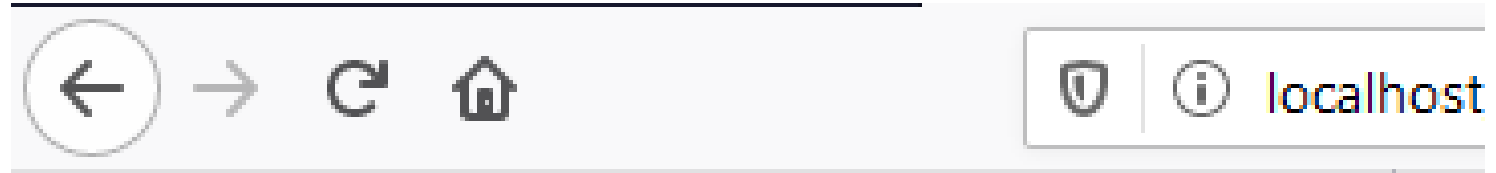
```
    $arr=array('how','are','you','today');
```

```
    echo implode(" ",$arr);
```

```
?>
```

```
</body>
```

```
</html>
```



how are you today

2.2 Extracting data from array

- Explode():
 - ✓ The explode() function breaks a string into an array.

✓ *explode(separator, string, limit);*

separator: Required. Specifies where to break the string.

String : Required. The string to split.

Limit : Optional. Specifies the number of elements to return.

Possible values:

Greater than 0 : returns an array with max of limit elements.

Less than 0 : returns an array except for the last -limit elements.

0 : Returns an array with one element.

2.2 Extracting data from array

- Explode():

```
<html><head><title>1D array</title></head>
```

```
<body><?php
```

```
    $arr='how,are,you,today';
```

```
    $x=explode(',',$arr,0);
```

```
    echo "Explode with 0<br>";
```

```
    foreach($x as $w)
```

```
    {echo $w."<br>";}
```

```
    $y=explode(',',$arr,3);
```

```
    echo "<br><br>Explode with +ve<br>";
```

```
    foreach($y as $w)
```

```
    {echo $w."<br>";}
```

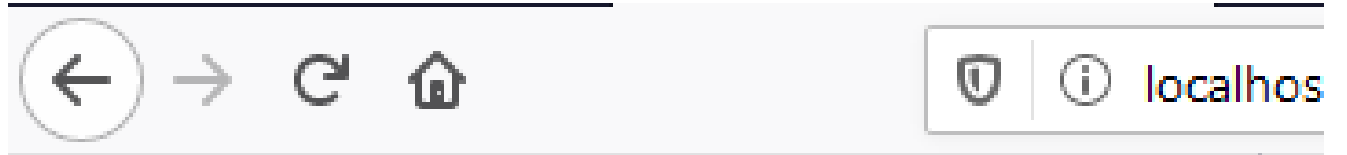
```
    $z=explode(',',$arr,-1);
```

```
    echo "<br><br>Explode with -ve<br>";
```

```
    foreach($z as $w)
```

```
    {echo $w."<br>";}?>
```

```
</body></html>
```



Explode with 0

how,are,you,today

Explode with +ve

how

are

you,today

Explode with -ve

how

are

you

2.2 Extracting data from array

- `Array_flip()`:
 - ✓ The `array_flip()` function flips/exchanges all keys with their associated values in an array.

✓ *`array_flip(array);`*

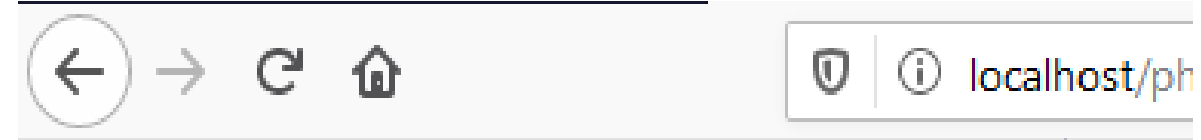
`array`: Required. Specifies an array of key/value pair to be flipped.

2.2 Extracting data from array

- Array_flip():

```
<html>
```

```
<body><?php
$a1=array("a"=>"red","b"=>"green",
"b"=>"blue","d"=>"yellow");
echo "<br> Before flip<br>";
foreach($a1 as $x=>$y)
{echo "$x => $y <br>";}
$result=array_flip($a1);
echo "<br><br><br>After flip<br>";
foreach($result as $x=>$y)
{echo "$x => $y <br>";}?>
</body></html>
```



Before flip

a => red

b => green

c => blue

d => yellow

After flip

red => a

green => b

blue => c

yellow => d

2.3 Traversing array

- `array_walk()`:
It runs each array element in a user-defined function. The array's keys and values are parameters in the user defined function.

Syntax:

`array_walk(array, myfunction, parameter...)`

where:

<i>array</i>	<i>: Required</i>
<i>myfunction</i>	<i>: Required. Name of the user-defined function.</i>
<i>Parameter,..</i>	<i>: Optional. Specifies a parameter to the user-defined function.</i>

2.3 Traversing array

- array_walk():

```
<html>
<body>
<?php
function myfunction($value,$key)
{
    echo "The key $key has the value $value<br>";
}
$a=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
array_walk($a,"myfunction");
?>
</body>
</html>
```

Output:

The key a has the value red
The key b has the value green
The key c has the value blue
The key d has the value yellow

2.3 Traversing array

- array_walk(): with extra parameter

```
<html>
<body>
<?php
function myfunction($value,$key,$p)
{
    echo "$key $p $value<br>";
}
$a=array("a"=>"red","b"=>"green","c"=>"blue");
array_walk($a,"myfunction","has the value");
?>
</body>
</html>
```

Output:

```
a has the value red
b has the value green
c has the value blue
```

2.3 Traversing array

- array_walk(): changing value of array

```
<html>
<body>
<?php
function myfunction(&$value,$key)
{
    if($key=="b")
        $value="yellow";
}
$a=array("a"=>"red","b"=>"green","c"=>"blue");
array_walk($a,"myfunction");
print_r($a);
?>
</body>
</html>
```

Output:

Array ([a] => red [b] => yellow [c] => blue)

2.3 Traversing array

- `array_map()`:
It sends each value of an array to a user-made function, and returns an array with new values, given by the user-made function.

Syntax:

`array_map(myfunction, array1, array2, array3, ...)`

where:

<code>myfunction</code>	: Required. Name of the user-defined function.
<code>array1</code>	: Required. Specifies an array.
<code>array2</code>	: Optional. Specifies an array.
<code>array3</code>	: Optional. Specifies an array.

2.3 Traversing array

- array_map():

```
<html>
<body>
<?php
function myfunction($num)
{
    return($num*$num);
}
$a=array(1,2,3,4,5);
print_r(array_map("myfunction",$a));
?>
</body>
</html>
```

Output:

Array ([0] => 1 [1] => 4 [2] => 9 [3] => 16 [4] => 25)

2.3 Traversing array

- `array_map()`: doesn't change the original array

```
<html>
```

```
<body>
```

```
<?php
```

```
function myfunction($v)
```

```
{ $v=strtoupper($v);
```

```
    return $v;
```

```
}
```

```
$a=array("Animal" => "horse", "Type" => "mammal");
```

```
print_r(array_map("myfunction",$a));
```

```
echo "<br>";
```

```
print_r($a);?>
```

```
</body>
```

```
</html>
```

Output

```
Array ( [Animal] => HORSE [Type] => MAMMAL )
```

```
Array ( [Animal] => horse [Type] => mammal )
```


2.3 Traversing array

- array_map(): using 2 arrays

```
<html>
<body>
<?php
function myfunction($v1,$v2)
{ if ($v1=== $v2)
  { return "same";
  }
return "different";
}
$a1=array("Horse","Dog","Cat");
$a2=array("Cow","Dog","cat");
print_r(array_map("myfunction",$a1,$a2));
?>
</body>
</html>
```

Output:

Array ([0] => different [1] => same [2] => different)

2.3 Traversing array

Parameter	Array_walk()	Array_map()
Changing Values	Changes value of input array	Doesn't change values of input array
Array keys access	Can access array keys	Cannot access array keys
Return value	Only returns true	Returns a new array
Iterating multiple arrays	Operates only on one array	Can operate on multiple arrays simultaneously
Passing arbitrary parameter	Can receive an extra arbitrary parameter	Cannot accept an extra arbitrary parameter

2.4 Functions and its types

- Sorting an array:
 - ✓ **sort()**: sort arrays in ascending order
 - ✓ **rsort()**: sort arrays in descending order
 - ✓ **asort()**: sort associative arrays in ascending order, according to the value
 - ✓ **ksort()**: sort associative arrays in ascending order, according to the key
 - ✓ **arsort()**: sort associative arrays in descending order, according to the value
 - ✓ **krsort()**: sort associative arrays in descending order, according to the key

2.4 Functions and its types

- PHP provides 2 types of functions:
 - ✓ Built-in Functions:
PHP provides huge collection of built-in library functions.
 - ✓ User Defined Functions:
PHP also allows us to create our own customized functions called user-defined functions

2.4 Functions and its types

- User Defined Functions:

A user defined function declaration starts with the word *function*:

Syntax:

```
function function_name()
{
    code to be executed;
}
```

Function names are not case sensitive.

2.4 Functions and its types

- User Defined Functions: with Arguments

- ✓ Arguments are specified after the function name, inside the parentheses.
- ✓ You can add any number of arguments as you want, just separate them with a comma.

```
<html>
<body>
<?php
function familyName($fname, $year) {
    echo "$fname $year <br>";
}
familyName("Hello","World!!");
?>
</body>
</html>
```

Output:

Hello World!!

2.4 Functions and its types

- User Defined Functions: Default Arguments

✓ Default arguments can be used inside function arguments. Default values are used if no value is provided for that argument at function calling.

```
<html>
```

```
<body>
```

```
<?php
```

```
function familyName($fname, $year="World!!") {  
    echo "$fname $year <br>";
```

```
}
```

```
familyName("Hello");
```

```
?>
```

```
</body>
```

```
</html>
```

Output:

Hello World!!

2.4 Functions and its types

- User Defined Functions: Pass by Reference

✓ In pass by reference, address of the value is passed as an argument using &.

```
<html>
<body>
<?php
$x=10;
function pbr(&$y) {
    $y++;
}
pbr($x);
echo "$x";
?>
</body>
</html>
```

Output
11

2.4 Functions and its types

- User Defined Functions: Returning values

✓ Return keyword is used to return a value from a function.

```
<html>
```

```
<body>
```

```
<?php
```

```
function familyName($fname, $year="World!") {  
    return $fname."<br>".$year;  
}
```

```
echo familyName("Hello");
```

```
?>
```

```
</body>
```

```
</html>
```

Output:
Hello
World!

2.4 Functions and its types

- User Defined Functions: Variable Function

- ✓ PHP supports the concept of variable functions.
- ✓ This means that if a variable name has parenthesis appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it.

```
<html>
<body>
<?php
function hell($t)
{
    echo "This is hell $t";
}
$x='hell';
$x(10);
?>
</body>
</html>
```

Output:

This is hell 10

2.4 Functions and its types

- User Defined Functions: Anonymous Functions
 - ✓ Sometimes you need to have a functionality that have the lowest probability of being executed,
 - ✓ but there is a chance that it will get executed, so you don't want to skip that code it in your program,
 - ✓ Also you don't want it available as a part of your code.
 - ✓ So instead of having a regular function, you can create anonymous functions.

2.4 Functions and its types

- User Defined Functions: Anonymous Functions

- ✓ Anonymous functions are similar to the regular functions as they contain the same type of code, they accept arguments, return values etc..
- ✓ The key difference is - *they have no names & there is semicolon (;) after the function definition.*

Syntax:

```
function($argument1, $argument2,...)  
{  
  //definition  
};
```

2.4 Functions and its types

✓ User Defined Functions: Anonymous Functions

Since there isn't any function name, anonymous functions cannot be referred anywhere but the following things can be done with it.

- ✓ *You can assign it to a variable and can call it using the variable name.*
- ✓ *You can store a bunch of different anonymous functions in a single array.*
- ✓ *You can pass this function to another function as a parameter (Callback).*
- ✓ *Return it from within an outer function so that it can access the outer function's variables (closure).*

2.4 Functions and its types

✓ User Defined Functions: Anonymous Functions

You can assign it to a variable and can call it using the variable name.

```
<html>
<body>
<?php
    $addition=function($arg1,$arg2)
    {
        return $arg1+$arg2;
    };
    echo "Sum=".$addition(10,20);
?>
</body>
</html>
```

Output:
Sum=30

2.4 Functions and its types

✓ User Defined Functions: Anonymous Functions

✓ *You can store a bunch of different anonymous functions in a single array.*

```
<html>
```

```
<body>
```

```
<?php
```

```
$myarray=array((function($f)
```

```
{ echo "Hello ".$f." This is 0th index.<br>";  
}),
```

```
(function()
```

```
{ echo "This is 1st index.<br>";  
}));
```

```
$call= $myarray[0];
```

```
$call("Zaid");
```

```
?>
```

```
</body>
```

```
</html>
```

Hello Zaid This is 0th index.

2.4 Functions and its types

✓ User Defined Functions: Anonymous Functions

You can pass this function to another function as a parameter (Callback).

A callback function is a function that you can pass to another function as an argument.

Once you access your callback function the receiving function can use it whenever it needs to.

2.4 Functions and its types

✓ User Defined Functions: Anonymous Functions

You can pass this function to another function as a parameter (Callback).

```
<html>
<body>
<?php
$num_array = array(1,2,3,4,5);
$new_array = array_map(function($num){
return $num*$num;
},$num_array);
echo "Original array <br>";
print_r($num_array);
echo "<br><br>New array array <br>";
print_r($new_array);
?>
</body></html>
```

Output:

Original array

Array ([0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5)

New array array

Array ([0] => 1 [1] => 4 [2] => 9 [3] => 16 [4] => 25)

2.4 Functions and its types

✓ User Defined Functions: Anonymous Functions

closure using Anonymous function:

A closure is an Anonymous function that can access variables imported from outside scope without using any global variables.

```
<html>
<body>
<?php
//define a regular variable
$user='Zaid';
//create a closure using anonymous function
$message=function() use ($user){
echo 'hello '. $user;
};
$message();
?>
</body>
</html>
```

Output:

hello Zaid

2.4 Functions and its types

✓ User Defined Functions: Anonymous Functions

closure using Anonymous function:

```
<?php
$params = 'Ronaldo!';
function sayHello()
{
    $params = 'Zidane!';
    $func = function() use ($params)
    {
        echo 'Hi, I am ' . $params;
    };
    $func();
}
sayHello();
?>
```

Hi, I am Zidane!

2.4 Functions and its types

- ✓ User Defined Functions: Anonymous Functions
to modify the value of outer scope element inside closure:

```
<?php
$params = 'Ronaldo!';
function sayHello()
{
    $params = 'Zidane!';
    $func = function() use (&$params)
    {
        $params = 'Messi!';
    };
    $func();
    echo 'I am ' . $params; // prints I am Dave!
}
sayHello();
?>
```

I am Messi!

2.5 Operations on String

✓ `str_word_count()`:

✓ It counts the no. of words in a string.

Syntax: `str_word_count(string, return, char);`

where:

string : Required. Specifies the string to check.

return : Optional. Specifies the return value of the function.

0 : Default. Returns the no. of words found.

1 : Returns an array with the words from the string.

2 : Returns an array where the key is the position of the word in the string, and value is the actual word.

char : Optional. Specifies special characters to be considered as word.

2.5 Operations on String

✓ `str_word_count()`:

```
<html>
<body>
<?php
print_r(str_word_count("Hello world!",0));
echo "<br>";
print_r(str_word_count("Hello world!",1));
echo "<br>";
print_r(str_word_count("Hello world!",2));
?>
</body>
</html>
```

2

Array ([0] => Hello [1] => world)

Array ([0] => Hello [6] => world)

2.5 Operations on String

✓ `str_word_count()`:

`<html>`

`<body>`

`<?php`

`print_r(str_word_count("Hello world & good morning!",1));`

`print "
";`

`print_r(str_word_count("Hello world & good morning!",1,"&"));`

`?>`

`</body>`

`</html>`

Array ([0] => Hello [1] => world [2] => good [3] => morning)

Array ([0] => Hello [1] => world [2] => & [3] => good [4] => morning)

2.5 Operations on String

✓strlen():

✓ It returns the length of a string.

Syntax:

strlen(string)

where:

string : Required. Specifies the string to check.

Output:
5

<html>

<body>

<?php

echo strlen("Hello");

?>

</body>

</html>

2.5 Operations on String

✓ `strrev()`:

✓ It reverses a string.

Syntax:

`strrev(string)`

where:

string : Required. It specifies the string to reverse.

Output:

olleH

`<html>`

`<body>`

`<?php`

`echo strrev("Hello");`

`?>`

`</body>`

`</html>`

2.5 Operations on String

✓ `strpos()`:

- ✓ It finds the position of the occurrence of a string inside another string. It is case-sensitive.

Syntax:

`strpos(string, find, start)`

where:

string : Required. It specifies the string to search.

Find : Required. Specifies the string to find.

Start : Optional. Specifies where to begin the search. If start is -ve number, it counts from the end of the string.

2.5 Operations on String

✓ strpos():

E.g.:

<html>

<body>

<?php

echo strpos("I love php, I love php too!","php");

?>

</body>

</html>

Output:
7

2.5 Operations on String

✓ strpos():

E.g.:

<html>

<body>

<?php

echo strpos("I love php, I love php too!","php",-9);

?>

</body>

</html>



19

2.5 Operations on String

✓ `str_replace()`:

- ✓ It replaces some characters with some other characters in a string.
- ✓ If the string to be searched is an array, it returns an array.
- ✓ If the string to be searched is an array, find and replace is performed with every array element.
- ✓ If find and replace are arrays, and replace has fewer elements than find, an empty string will be used as replace.

Syntax:

`str_replace(find, replace, string, count)`

where: find : Required. Specifies the string to find.

replace: Required. Specifies the value to replace the value in find.

string : Required. It specifies the string to search.

count : Optional. A variable that counts the no. of replacements.

2.5 Operations on String

✓ `str_replace()`:

```
<html>
<body>
<?php
echo "Normal find and replace<br>";
echo str_replace("world","Peter","Hello world!");
echo "<br><br> find and replace within array string<br>";
$arr = array("blue","red","green","yellow");
print_r(str_replace("red","pink",$arr,$i));
echo "Replacements: $i";
echo "<br><br>Find and replace with array find, array replace and array string<br>";
$find = array("Hello","world");
$replace = array("B");
$arr = array("Hello","world","!");
print_r(str_replace($find,$replace,$arr));?>
</body>
</html>
```

Output:

Normal find and replace
Hello Peter!

find and replace within array string
Array ([0] => blue [1] => pink [2] => green [3] => yellow) Replacements: 1

Find and replace with array find, array replace and array string
Array ([0] => B [1] => [2] => !)

2.5 Operations on String

- ✓ `strtoupper()`:
 - ✓ It converts a string to uppercase.

Syntax:

`strtoupper(string)`

where:

string: Required. Specifies the string to convert.

Output:

HELLO WORLD!

E.g.:

```
<html>
<body>
<?php
echo strtoupper("Hello WORLD!");
?>
</body>
</html>
```

2.5 Operations on String

✓ `strtolower()`:

✓ It converts a string to lowercase.

Syntax:

`strtolower(string)`

where:

string: Required. Specifies the string to convert.

Output:

hello world!

E.g.:

`<html>`

`<body>`

`<?php`

`echo strtolower("Hello WORLD!");`

`?>`

`</body>`

`</html>`

2.5 Operations on String

✓ `strcmp()`:

✓ It compares two string. It is case-sensitive.

Syntax:

`strcmp(string1, string2)`

where:

string1 : Required. Specifies the first string to compare.

string2: Required. Specifies the second string to compare.

Return values:

0 : if two strings are equal.

<0: if string1 is less than string2.

>0: if string1 is greater than string2.

2.5 Operations on String

✓ strcmp():

```
<html>
```

```
<body>
```

```
<?php
```

```
echo strcmp("Hello","Hello");
```

```
echo "<br>";
```

```
echo strcmp("Hello","hELLo");
```

```
?>
```

```
</body>
```

```
</html>
```

Output:

0

-32

2.5 Operations on String

✓ `ucwords()`:

✓ It converts the first character of each word in a string to uppercase.

Syntax:

`ucwords(string, delimiters)`

where:

string: Required. Specifies the string to convert.

delimiters: Optional. Specifies the word separator character.

2.5 Operations on String

✓ucwords():

```
<html>  
<body>  
<?php  
echo ucwords("hello|world", "|");?>  
</body>  
</html>
```

Output:
Hello|World

2.6 Basic graphic concepts

- PHP is not limited to creating just HTML output.
- It can also be used to create and manipulate image files in a variety of different image formats, including GIF, PNG, JPEG, WBMP, and XPM.
- Images appear in the form of logos, buttons, photographs, charts, advertisements and icons.
- Images can be created dynamically with PHP using GD (Graphics Draw) extension.

2.6 Basic graphic concepts

✓ Creating an Image:

- *imagecreate()* function creates a new palette based image.
- It returns an image identifier representing a blank image of specified size.

Syntax:

imagecreate(x_size, y_size);

where:

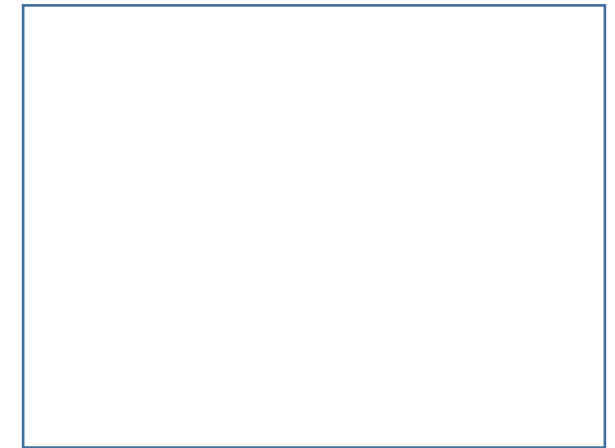
x_size & y_size parameters are in pixels.

Eg.

\$h=100;

\$w=200;

\$img=imagecreate(\$h,\$w);



2.6 Basic graphic concepts

✓ Creating an Image:

- To set color in an image *imagecolorallocate()* function can be used.
- Returns a color identifier representing the color composed of the given RGB components.
- *imagecolorallocate()* must be called to create each color that is to be used in the image represented by image.
- The first call to *imagecolorallocate()* fills the background color in palette-based images - images created using *imagecreate()*.

Syntax:

imagecolorallocate(image, red, green, blue);

where:

image: it is the image user is working on.

Red, Green, Blue: they are the color you want to fill the image with.
Each color can accept a value between 0-255

2.6 Basic graphic concepts

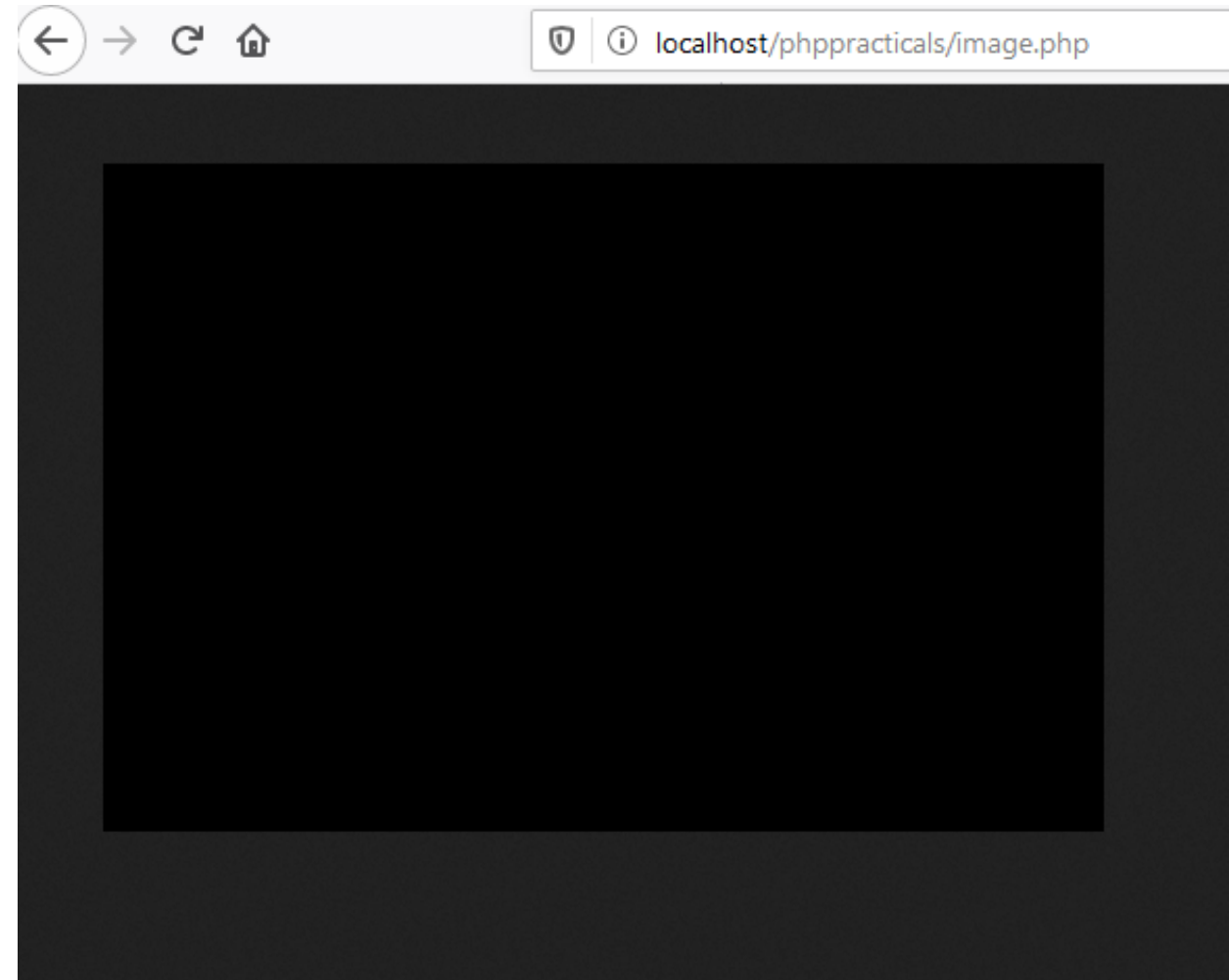
✓ Creating an Image: Image creation functions for various image formats

- `imagegif()`: output a GIF image to browser or file.
- `imagejpeg()`: output a JPEG image to browser or file.
- `imagewbmp()`: output a WBMP image to browser or file.
- `imagepng()`: output a PNG image to browser or file.

2.6 Basic graphic concepts

✓ Creating an Image:

```
<?php
header('Content-Type: image/jpeg');
//error_reporting(0);
$x=480;
$y=320;
$image1=imagecreate($x,$y);
imagejpeg($image1);
?>
```



****header:** a **Content-Type** header tells the client what the **content type** of the returned **content** actually is

2.6 Basic graphic concepts

✓ Images with text:

- It draw a string horizontally at the given coordinates.

Syntax:

imagestring(resource \$image, int \$font, int \$x, int \$y, string \$string, int \$color):bool;

where:

image: An image resource.

Font : Can be 1, 2, 3, 4, 5 for built-in font size in latin2 encoding.

X : x-coordinate of the upper-left corner.

Y : y-coordinate of the upper-left corner.

String: The string to be written.

Color : A color identifier created with `imagecolorallocate()`

2.6 Basic graphic concepts

✓ Images with text:

```
<?php
$x=480;
$y=320;
$image1=imagecreate($x,$y);
$bg_color=imagecolorallocate($image1,
    240,240,140);
$txt_color=imagecolorallocate($image1,0,0,0);
imagestring($image1,5,5,18,"Hello World!!",$txt_color);
imagejpeg($image1);
header('Content-Type: image/jpeg'); ?>
```



2.6 Basic graphic concepts

✓ Displaying images in HTML pages:

```
<html>  
  <head>  
    <title>  
      Displaying images in HTML pages  
    </title>  
  </head>  
  <body>  
      
  </body>  
</html>
```



2.6 Basic graphic concepts

✓ Scaling Images: There are two ways to change the size of an image.

- `imagecopyresized()`
- `imagecopyresampled()`

2.6 Basic graphic concepts

✓ Scaling Images:

- `imagecopyresized()`

This function copies a rectangular portion of one image to another image.

Syntax:

`imagecopyresized(dst, src, dst_x, dst_y, src_x, src_y, dst_w, dst_h, src_w, src_h)`

Where:

`dst / src`: Destination image / Source image.

`dst_x / src_x`: x coordinate of destination/source image.

`dst_y / src_y`: y coordinate of destination /source image.

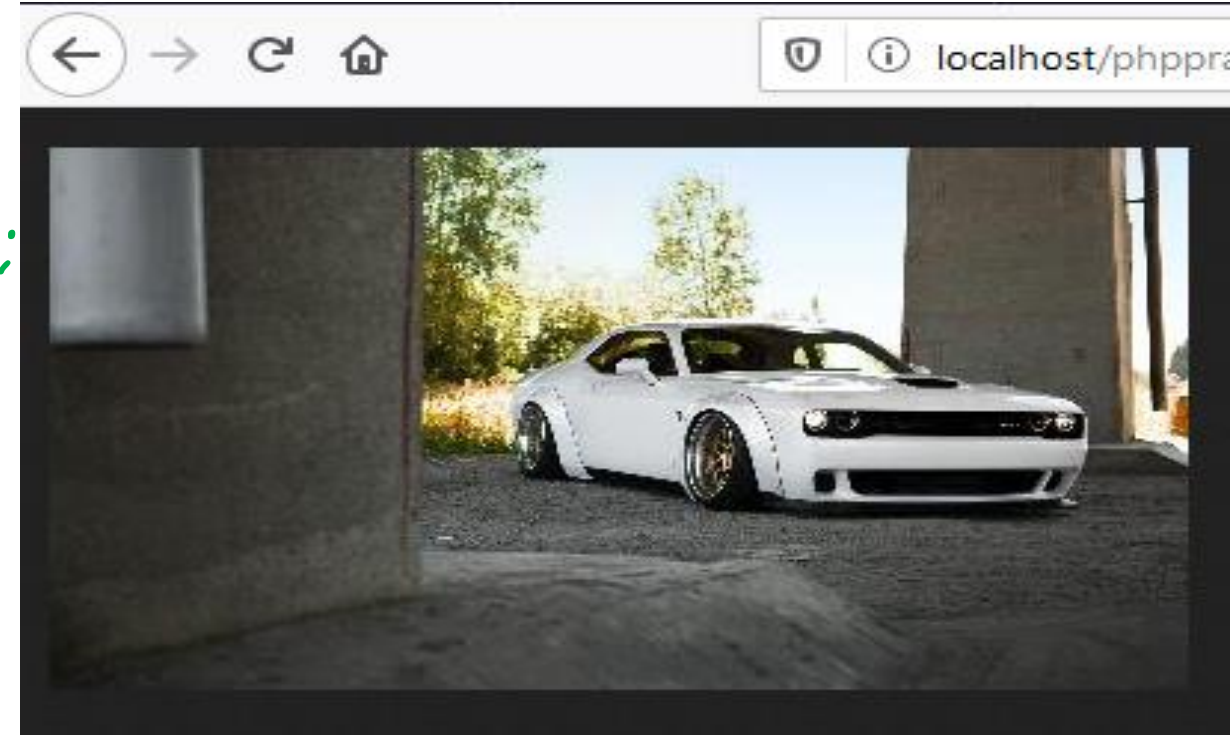
`dst_w/ src_w`: destination/source image width.

`dst_h/ src_h`: destination / source image height.

2.6 Basic graphic concepts

✓ Scaling Images: `imagecopyresized()`

```
<?php
$src=imagecreatefromjpeg('Dodge.jpg');
$width=ImageSx($src);
$height=ImageSy($src);
$x=$width/5;
$y=$height/5;
$dst=ImageCreateTrueColor($x,$y);
imagecopyresized($dst,$src,0,0,0,0,$x,$y,$width,$height);
header('Content-Type: image/jpeg');
imagejpeg($dst);
?>
```



2.6 Basic graphic concepts

✓ Scaling Images:

- `imagecopyresampled()`

This function copies a rectangular portion of one image to another image. It uses smoothing and pixel interpolation algorithm to yield better results.

Syntax:

`imagecopyresample(dst, src, dst_x, dst_y, src_x, src_y, dst_w, dst_h, src_w, src_h)`

Where:

`dst / src`: Destination image / Source image.

`dst_x / src_x`: x coordinate of destination/source image.

`dst_y / src_y`: y coordinate of destination /source image.

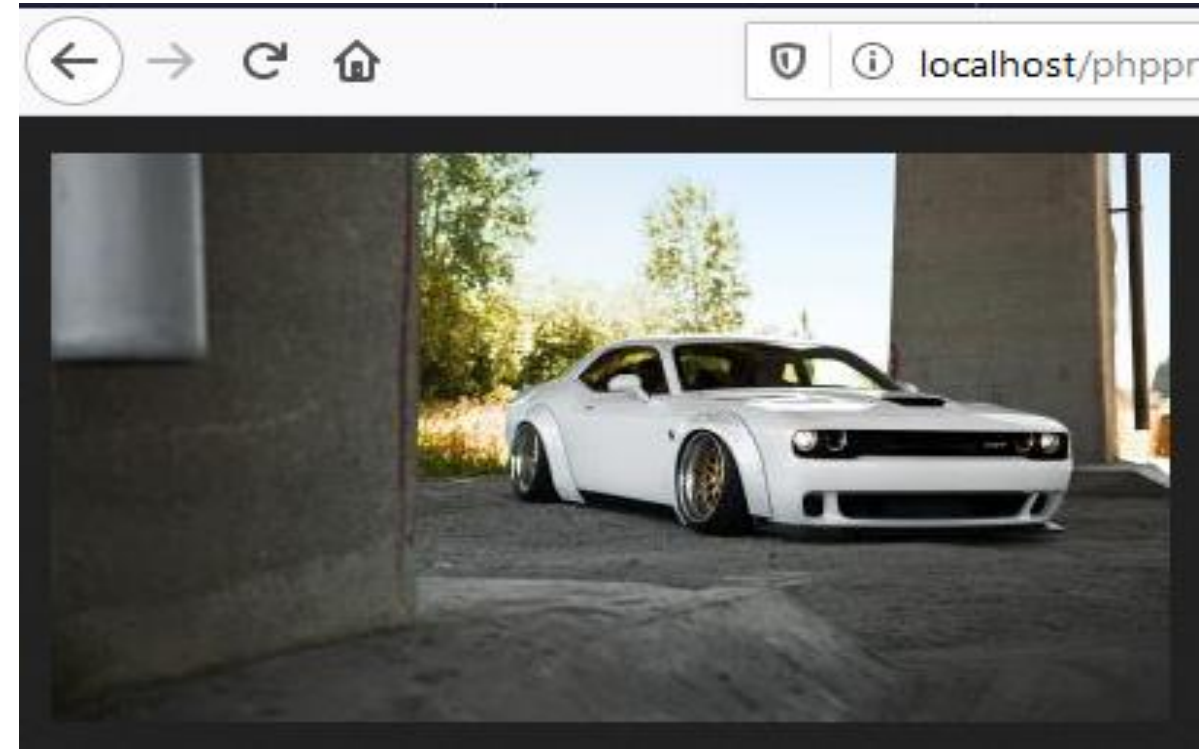
`dst_w/ src_w`: destination/source image width.

`dst_h/ src_h`: destination / source image height.

2.6 Basic graphic concepts

✓ Scaling Images: `imagecopyresampled()`

```
<?php
$src=imagecreatefromjpeg('Dodge.jpg');
$width=ImageSx($src);
$height=ImageSy($src);
$x=$width/5;
$y=$height/5;
$dst=ImageCreateTrueColor($x,$y);
imagecopyresampled($dst,$src,0,0,0,0,$x,$y,$width,$height);
header('Content-Type: image/jpg');
imagejpeg($dst);
?>
```



2.6 Basic graphic concepts

✓ Creation of PDF document:

- **FPDF** is a PHP class which allows to generate PDF files with PHP program without using the PDFlib library.
- F from FPDF stands for Free: you may use it for any kind of usage and modify it to suit your needs.

2.6 Basic graphic concepts

✓ Creation of PDF document: Advantages of FPDF

- Choice of measure unit, page format and margins.
- Page header and footer management.
- Automatic page break.
- Automatic line break and text adjustment.
- Image support (JPEG, PNG and GIF).
- Colors.
- Links.
- Page compression.

2.6 Basic graphic concepts

✓ Creation of PDF document: Advantages of FPDF

- **AddPage():**

- ❑ This function is used to add a page to newly created pdf object.
- ❑ The origin is at the upper-left corner and the current position is by default set at 1cm from the borders.
- ❑ Margins can be changed with SetMargins().

Syntax:

obj->AddPage();

2.6 Basic graphic concepts

✓ Creation of PDF document: Advantages of FPDF

- **SetFont():**

- ❑ It is mandatory to select a font before anything can be printed on the file.
- ❑ SetFont() function allows to choose different font, font style and font size.

Syntax:

obj->SetFont(font, font_style, size);

Where:

Font: It is font used for writing. E.g. Arial, Times etc

Font_style: It is font style applied. E.g. Bold-B, Italic-I, Underline-U

Size: It is the size of the font specified in points, not in millimeters.

2.6 Basic graphic concepts

✓ Creation of PDF document: Advantages of FPDF

- **Cell():**

- ❑ A cell is a rectangular area, possible framed, which contains a line of text.
- ❑ It is output at the current position.
- ❑ We have to specify its dimensions, its text(centered or aligned), if borders should be drawn and where the current position moves after it (to the right, below or to the beginning of the next line)

Syntax:

obj->Cell(width, height, text, border, ln, align, fill, link);

Where:

width: Cell width. If 0, the cell extends up to the right margin.

height: Cell height. Default value is 0.

text: String to be printed. Default value is empty string.

2.6 Basic graphic concepts

✓ Creation of PDF document: Advantages of FPDF

- **Cell():**

Syntax:

obj->Cell(width, height, text, border, ln, align, fill, link);

Where:

border: Indicates if borders must be drawn around the cell

values: 0-no border(default), 1-frame, L-left, T-Top, R-Right, B-Bottom

ln: Indicates where the current position should go after the call.

Values: 0-to the right, 1-to the beginning of the next line, 2-below

align: Allows to center or align the text

values: L-left align, R-right align, C-center

fill: Indicates if the cell background must be painted (True/False). False is default

link: URL or identifier returned by AddLink().

2.6 Basic graphic concepts

✓ Creation of PDF document:

- *file()*:

- ❑ Reads a file into an array.
- ❑ Each array element contains a line from the file, with newline character still attached.

file(filename, flag, context)

Filename: required. Specifies the path to the file to read

Flag: Optional.

FILE_USE_INCLUDE_PATH: search for the file in the include_path

FILE_IGNORE_NEW_LINES: skip the newline at the end of each array element.

FILE_SKIP_EMPTY_LINES: skip empty lines in the file.

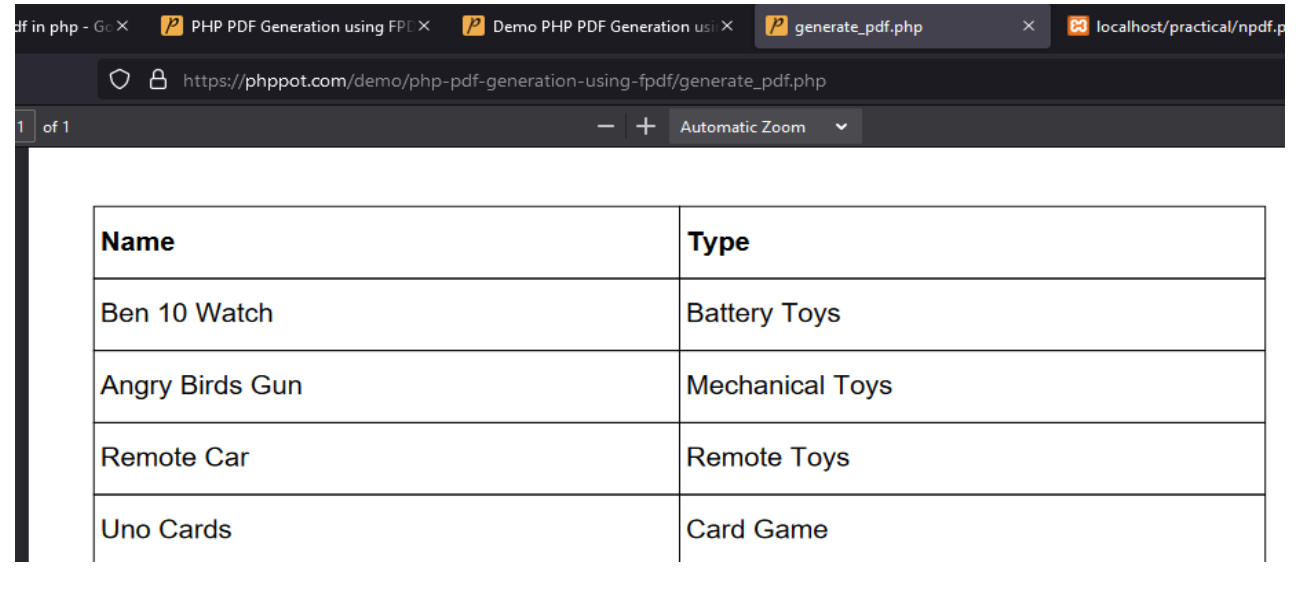
2.6 Basic graphic concepts

✓ To create a PDF file using FPDF

- Download FPDF package and keep it inside your folder.
- **Require** statement is used to copy all text/code/markup that exists in the specified & copies it into the file that uses the require statement.
- **Output** function sends the document to a given destination: browser, local file, string.

2.6 Basic graphic concepts

```
<?php
require('fpdf/fpdf.php');
$pdf = new FPDF();
$pdf->AddPage();
$row=file('toys.txt');
$pdf->SetFont('Arial','B',12);
foreach($row as $rowValue) {
    $data=explode(':', $rowValue);
    foreach($data as $columnValue)
        $pdf->Cell(90,12,$columnValue,1);
    $pdf->SetFont('Arial','',12);
    $pdf->Ln();
}$pdf->Output();
?>
```



Name	Type
Ben 10 Watch	Battery Toys
Angry Birds Gun	Mechanical Toys
Remote Car	Remote Toys
Uno Cards	Card Game

 toys - Notepad

File Edit Format View Help

```
Name;Type
Ben 10 Watch;Battery Toys
Angry Birds Gun;Mechanical Toys
Remote Car;Remote Toys
Uno Cards;Card Game
Keyboard;Musical Toys
Jigsaws;Board Game
```