JavaScript and PHP

Prof. Hyuk-Yoon Kwon
https://sites.google.com/view/seoultech-bigdata

Most parts are based on slides used in
(http://cgi.csc.liv.ac.uk/~martin/teaching/comp519/)

JavaScript

Built-In Objects

Built-In Objects

- Some basic objects are built-in to JavaScript
  - String
  - Array
  - Date
  - Boolean
  - Math

Strings

- A String object is created every time you use a string literal
- Have many of the same methods as in Java
  - charAt, concat, indexOf, lastIndexOf, match, replace, search, slice, split, substr, substring, toLowerCase, toUpperCase

```
var carname = "Volvo XC60";  // Double quotes
var carname = 'Volvo XC60';  // Single quotes
```

Practice - String

- **Check the result of the following JavaScript**

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;

var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");

var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");

var str = "Apple, Banana, Kiwi";
var res = str.slice(7, 13);

var str = "Apple, Banana, Kiwi";
var res = str.slice(-12, -6);
```

```
str = "Please visit Microsoft!";
var n = str.replace("Microsoft", "W3Schools");
```

```
var text1 = "Hello";
var text2 = "World";
var text3 = text1.concat(" ", text2);
```

```
var txt = "a,b,c,d,e";   // String
txt.split(",");          // Split on commas
```

Arrays

- An Array object can be easily created by enumerating its items
- Properties
  - length
- Methods
  - concat, indexOf, join, lastIndexOf, pop, push, reverse, shift, slice, sort, splice, toString, unshift

```
var a = ['ali', 20, 14];
a.push(10);
a.sort();          // [10, 14, 20, "ali"]
```

Practice - Array

- **Check the result of the following JavaScript**

```javascript
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.pop();
```

```javascript
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var x = fruits.push("Kiwi");
```

```javascript
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits[fruits.length] = "Kiwi";
```

```javascript
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(0, 1);
```

```
var myGirls = ["Cecilie", "Lone"];
var myBoys = ["Emil", "Tobias", "Linus"];
var myChildren = myGirls.concat(myBoys);
```

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(3);
```

Dates

- The Date class makes working with dates easier
- Some methods
  - getYear, getMonth, getDay, getHours, getMinutes, getSeconds, getMilliseconds, getTime, parse()

```
<script>
var today = new Date();
var deadline = new Date(2018, 10, 20);
if (today < deadline) {
       days = (deadline-today) / (3600*24*1000);
       alert('You have' + days + ' days left');
}
</script>
```

Math

- The Math object encapsulates many commonly-used mathematical functions and constants
- Math functions
  - abs, acos, asin, atan, atan2, ceil, cos, exp, floor, log, max, min, pow, random, round, sin, sqrt, tan
- Math constants
  - E, LN2, LN10, LOG2E, LOG10E, PI, SQRT1_2, SQRT2

```
Math.sqrt(2);
Math.cos(Math.PI);
```

Document Object Model

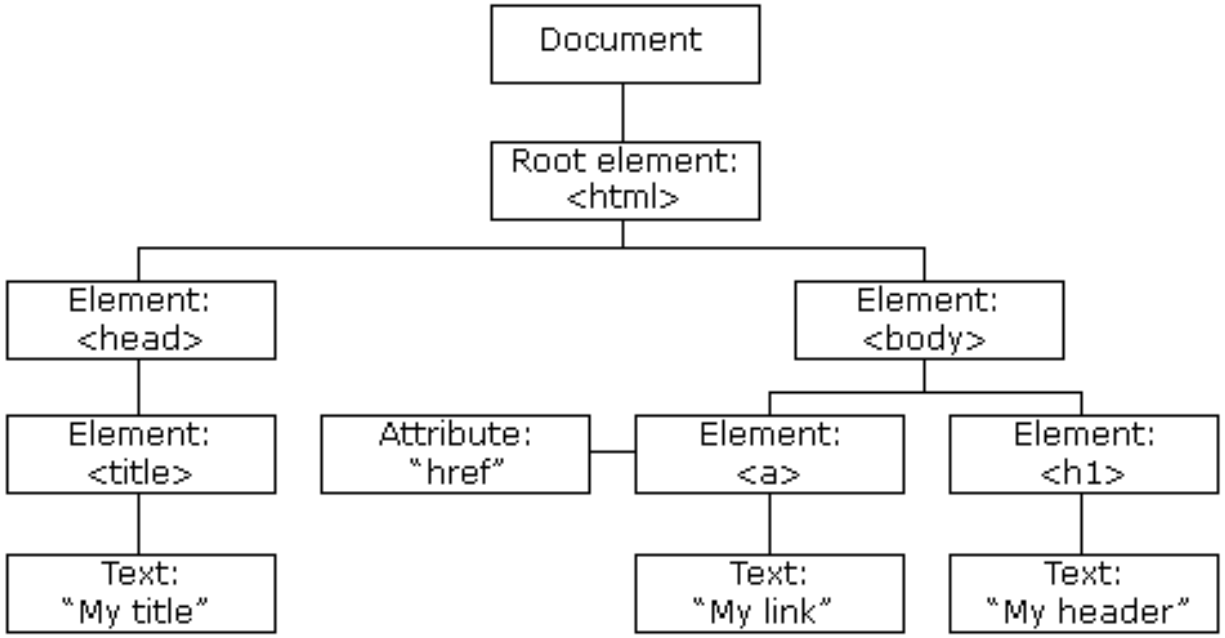
- The Document Object Model (DOM) defines a standard for accessing documents through an object model
- DOM is platform and language independent
- It allows programs and scripts to dynamically access and update the content, structure, and style of a document

The HTML DOM

- The HTML DOM defines:
  - The HTML elements as objects
  - The properties of all HTML elements
  - The methods to access all HTML elements
  - The events for all HTML elements

The HTML DOM Tree

- The HTML DOM represents HTML document as a tree of objects



Finding Elements

- **Find elements by using the element id**

```
<p id="intro">Hello World!</p>

<script>
var myElement = document.getElementById("intro");
alert("The text from the intro paragraph is " + myElement.innerHTML);
</script>
```

- **Find element by using tag name**
  - getElementsByTagName(“p”)

Find elements by using the class name

- getElementsByClassName(“intro”)

```
<p class="intro">The DOM is very useful.</p>
<p class="intro">This example demonstrates the <b>getElementsByClassName</b>
method.</p>

<script>
var x = document.getElementsByClassName("intro");
alert('The first paragraph (index 0) with class="intro": ' + x[0].innerHTML);
</script>
```

Updating Elements

- Change HTML content
  - element.innerHTML = *new HTML*
- Change an HTML attribute
  - element.*attribute* = *new value*

```
var x = document.getElementById('image');
x.src = 'sun.png'
x.title = 'The Sun!';
```

Updating Elements

Change HTML content

```
<p id="intro">Hello World!</p>
<p id="demo"></p>

<script>
var myElement = document.getElementById("intro");
document.getElementById("demo").innerHTML =
"The text from the intro paragraph is " + myElement.innerHTML;
</script>
```

Change the attribute value

```
<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>
```

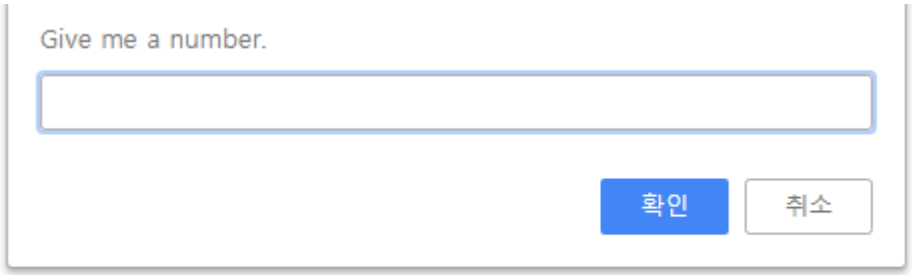
Practice – HTML DOM

- Write a JavaScript program to count and display the items of a dropdown list, in an alert window

```
<body><form>
Select your favorite Color :
<select id="mySelect">
<option>Red</option>
<option>Green</option>
<option>Blue</option>
<option>White</option>
</select>
<input type="button" onclick="getOptions()" value="Count and Output all
items">
</form></body>
```

HW Assignments #2 – HTML and JavaScript

1. Get two numbers using prompt, calculate the summation of two numbers, and alert the result

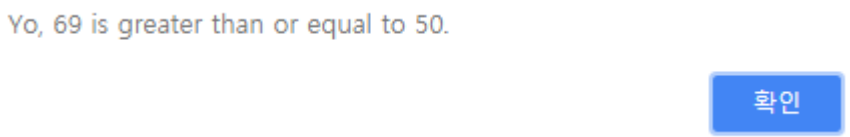


2. Get two numbers using prompt while satisfying the following requirement.

- Alert the message if the first number is less than second number
- After alert, increment the first number
- Repeat alert message while the first number is less than second number

3. Repeat the following steps for 5 times

- Generate a random number between 0 and 100
- Compare the generated random number with 50
- Alert the comparison result like this



- Hint: Use the following function for random value

```
function getRandom(max) {return (Math.floor(Math.random()*max))+1;}
```

4. Calculate the result while satisfying the following requirement.

- Give two values for operands for a operator
- Click a button for one operator (i.e., add, subtract, multiply, divide)
- Show the chosen operator in the second blank and show the result in the fourth blank

- Hints
  - We can call a specific function when clicking the button as follow

```
<input type="button" onClick="Add()" value="Add">
<input type="button" onClick="Subtract()" value="Subtract">
```

  - We can get the input value from a given form as follows

```
<form name="form01">
<input type="text" name="input01" value="" size="5">
<input type="text" name="input04" value="" size="1">
```

```
number1 = window.document.form01.input01.value;
```

  - Similarly, we can change the value in the form as follow

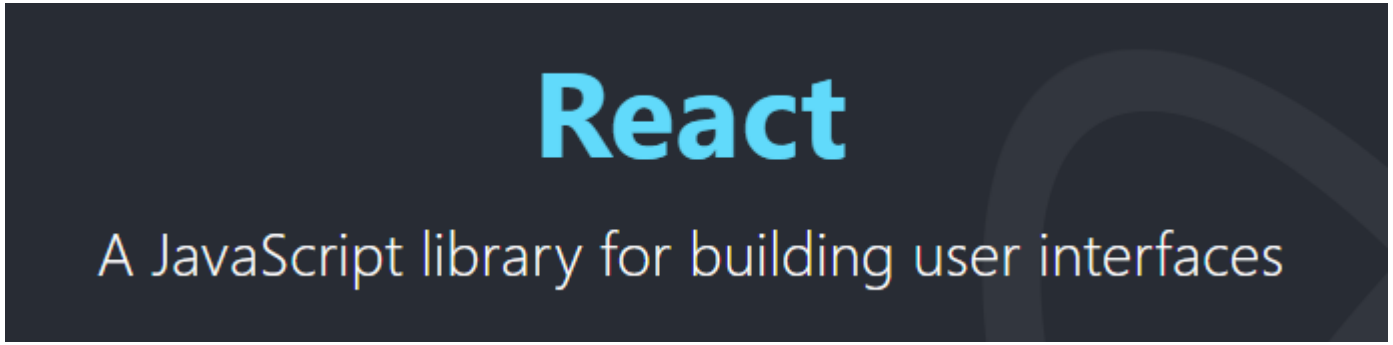
```
window.document.form01.input03.value = total;
window.document.form01.input04.value = "-";
```

Submissions

- Make one word file to include, for each problem, 1) captured final results, 2) result HTML and JavaScript codes, 3) code explanations
- Four HTML files

Recent Technologies - ReactJS

- **ReactJS (by Facebook)**




```html
<!DOCTYPE html>
<html lang="en">

<title>Test React</title>
<script src= "https://unpkg.com/react@16/umd/react.production.min.js"></script>
<script src= "https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>
<script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>

<body>
<div id="root"></div>

<script type="text/babel">
function tick() {
  const element = (<h1>{new Date().toLocaleTimeString()}</h1>);
  ReactDOM.render(element, document.getElementById('root'));
}

setInterval(tick, 1000);
</script>

</body>
</html>
```

Recent Technologies – Angular JS

Angular JS (by Google)

- AngularJS offers **functionality** to HTML applications

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div ng-app="">

<p>Input something in the input box:</p>
<p>Name : <input type="text" ng-model="name" placeholder="Enter name here"></p>
<h1>Hello {{name}}</h1>

</div>

</body>
</html>
```

PHP Basics

PHP Basics

- **Introduction to PHP**
  - a PHP file, PHP workings, running PHP.
- **Basic PHP syntax**
  - variables, operators, if...else...and switch, while, do while, and for.
- **Some useful PHP functions**
- **How to work with**
  - HTML forms, cookies, files, time and date.
- **How to create a basic checker for user-entered data**

Server-Side Dynamic Web Programming

- **CGI is one of the most common approaches to server-side programming**
  - Universal support: (almost) Every server supports CGI programming. A great deal of ready-to-use CGI code. Most APIs (Application Programming Interfaces) also allow CGI programming.
  - Choice of languages: CGI is extremely general, so that programs may be written in nearly any language. Perl is one of the most popular, but C, C++, Ruby, and Python are also used for CGI programming.
  - Drawbacks: A separate process is run every time the script is requested. A distinction is made between HTML pages and code.

```html
<!DOCTYPE html>
<html>
 <body>
  <form action="add.cgi" method="POST">
   Enter two numbers to add:<br />
   First Number: <input type="text" name="num1" /><br />
   Second Number: <input type="text" name="num2" /><br />
   <input type="submit" value="Add" />
  </form>
 </body>
</html>
```

```python
#!/usr/bin/env python2

import cgi
import cgitb
cgitb.enable()

input_data = cgi.FieldStorage()

print 'Content-Type:text/html' # HTML is following
print                          # Leave a blank line
print '<h1>Addition Results</h1>'
try:
  num1 = int(input_data["num1"].value)
  num2 = int(input_data["num2"].value)
except:
  print '<p>Sorry, we cannot turn your inputs into numbers (integers).</p>'
  return 1
print '<p>{0} + {1} = {2}</p>'.format(num1, num2, num1 + num2)
```

Other server-side alternatives try to avoid the drawbacks

- **Server-Side Includes (SSI)**
  - Code is embedded in HTML pages, and evaluated on the server while the pages are being served.
  - Add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program.
- **Active Server Pages (ASP and ASP.NET, Microsoft)**
  - The ASP engine is integrated into the web server so it does not require an additional process.
  - It allows programmers to mix code within HTML pages instead of writing separate programs.
- **Java Server Pages (JSP)**
  - Like ASP, another technology that allows developers to embed Java in web pages.

PHP (Hypertext Preprocessor)

- **Developed in 1995 by Rasmus Lerdorf (member of the Apache Group)**
  - originally designed as a tool for tracking visitors at Lerdorf's Web site
  - within 2 years, widely used in conjunction with the Apache server
  - developed into full-featured, scripting language for server-side programming
  - free, open-source
  - now fully integrated to work with mySQL databases

- **PHP is somewhat similar to JavaScript, only it's a server-side language**
  - PHP code is embedded in HTML using tags
  - when a page request arrives, the server recognizes PHP content via the file extension (`.php` or `.phtml`)
  - the server executes the PHP code, substitutes output into the HTML page
  - the resulting page is then downloaded to the client
  - user never sees the PHP code, only the output in the page

What do You Need?

- **WAMP supports PHP**
  - You don't need to do anything special!
  - You don't need to compile anything or install any extra tools!
  - Create some .php files in your web directory - and the server will parse them for you.

Basic PHP syntax

A PHP scripting block always starts with `<?php` and ends with `?>`.

A PHP scripting block can be placed (almost) anywhere in an HTML document.

```
<html>
<body>
   <?php echo '<p>While this is going to be parsed.</p>'; ?>
   <?php print('<p>Hello and welcome to <i>my</i> page!</p>');
   ?>

  <?php
   //This is a comment
  /*
   This is
   a comment
   block
  */
  ?>

</body>
</html>
```

view the output page

`print` and `echo` for output

`a semicolon (;)` at the end of each statement

// for a single-line comment

/* and */ for a large comment block.

The server executes the print and echo statements, substitutes output.

Variables and Data Types

All variables in PHP start with a $ sign symbol. A variable's type is determined by the context in which that variable is used (i.e. <u>there is no strong-typing in PHP</u>).

```
<html>
<body><p>
<?php
$foo = true; if ($foo) echo "1. It is TRUE! <br/>";

$txt = '1234'; echo "2. $txt <br/>";
$num = 1234; echo "3. $num <br/>";
$num = 1.234; echo "4. $num <br/>";

$beer = 'Heineken'; echo "5. $beer's taste is great <br/>";
$beer = 'Heineken'; echo "6. \$beer's taste is great <br/>";

$str = <<<EOD
7. Example of string
spanning multiple lines
using "heredoc" syntax.
EOD;
echo $str;
?>
</p></body>
</html>
```

view the output page

Four data types:

- boolean
  - true or false
- integer
- float
  - floating point numbers
- string
  - single quoted
  - double quoted

Arrays

An array in PHP is actually an ordered map. A map is a type that maps values to keys.

```
<?php
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

view the output page

`array()` = creates arrays

key = either an integer or a string.

value = any PHP type.

```
<?php
array(5 => 43, 32, 56, "b" => 12);
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

view the output page

if no key given (as in example), the PHP interpreter uses (maximum of the integer indices + 1).

if an existing key, its value will be overwritten.

```php
<?php
$arr = array(5 => 1, 12 => 2);
foreach ($arr as $key => $value) {
   echo $key, '=>', $value;
}
echo '<br>';
$arr[] = 56;    // the same as $arr[13] = 56;
$arr["x"] = 42; // adds a new element
foreach ($arr as $key => $value) {
   echo $key, '=>', $value;
}
?>
```

view the output page

Set values in Array

```
<?php
$arr = array(5 => 1, 12 => 2);
foreach ($arr as $key => $value) {
   echo $key, '=>', $value;
}
echo '<br>';
unset($arr[5]); // removes the element
foreach ($arr as $key => $value) {
   echo $key, '=>', $value;
}
unset($arr);    // deletes the whole array
?>
```

view the output page

`unset()` removes a key/value pair

```php
<?php
$arr = array(1 => 'one', 2 => 'two', 3 => 'three');
unset($arr[2]);
foreach ($arr as $key => $value) {
   echo $key, '=>', $value;
}
echo '<br>';
$b = array_values($arr);
foreach ($b as $key => $value) {
   echo $key, '=>', $value;
}
?>
```

view the output page

`array_values()`
makes reindexing effect
(indexing numerically)

Practice - Array

- Write a PHP script to get the largest key in the following array.
  - $ceu = array("Italy"=>"Rome", "Luxembourg"=>"Luxembourg", "Belgium"=> "Brussels", "Denmark"=>"Copenhagen", "Finland"=>"Helsinki", "France" => "Paris", "Slovakia"=>"Bratislava", "Slovenia"=>"Ljubljana", "Germany" => "Berlin", "Greece" => "Athens", "Ireland"=>"Dublin“);
- Write a PHP script to print “Second" and “Red” from the following array.
  - $color = array ("color" => array ("a" => "Red", "b" => "Green", "c" => "White"), "numbers" => array (1, 2, 3, 4, 5, 6), "holes" => array ("First", 5 => "Second", "Third"));
- Write a PHP script to count the total number of times a specific value appears in the following array.
  - $colors = array("c1"=>"Red", "c2"=>"Green", "c3"=>"Yellow", "c4"=>"Red");

Constants

A constant is an identifier (name) for a simple value.

```
<?php

// Valid constant names
define("FOO",    "something");
define("FOO2",    "something else");
define("FOO_BAR", "something more");

// Invalid constant names  (they shouldn't start
//      with a number!)

define("2FOO",    "something");

// This is valid, but should be avoided:
// PHP may one day provide a "magical" constant
// that will break your script

define("__FOO__", "something");

?>
```

You can access constants anywhere in your script without regard to scope.

Operators

- **Arithmetic Operators: +, -, *,/ , %, ++, --**
- **Assignment Operators: =, +=, -=, *=, /=, %=**

Example	Is the same as
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: ==, !=, >, <, >=, <=**
- **Logical Operators: &&, ||, !**
- **String Operators: . and .= (for string concatenation)**

```
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";
$a .= "World!";
```

Practice - Operators

1. Arithmetic operations on character variables : $d = 'A00'. Using this variable print the following numbers

   A01

   A02

   A03

   A04

   A05

2. Write a PHP program to calculate the mod of two given integers without using any inbuilt mod operator (%)

Conditionals: if else

Can execute a set of code depending on a condition

```
<?php
$d=date("D");
echo $d, "<br/>";
if ($d=="Fri")
     echo "Have a nice weekend! <br/>";
else
     echo "Have a nice day! <br/>";
?>
```

view the output page

if (condition)

code to be executed if condition is true;

else

code to be executed if condition is false;

date() is a built-in PHP function that can be called with many different parameters to return the date (and/or local time) in various formats

In this case we get a three letter string for the day of the week.

Conditionals: switch

Can select one of many sets of lines to execute

```
<?php
$x = rand(1,5);  // random integer
echo "x = $x <br/><br/>";
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
  break;
}
?>
```

view the output page

```
switch (expression)
{
case label1:
  code to be executed if
expression = label1;
  break;
case label2:
  code to be executed if
expression = label2;
  break;
default:
  code to be executed
  if expression is different
  from both label1 and label2;
  break;
}
```

Looping: while and do-while

Can loop depending on a condition

```
<?php
$i=1;
while($i <= 5)
{
  echo "The number is $i <br />";
  $i++;
}
?>
```

```
<?php
$i=0;
do
{
  $i++;
  echo "The number is $i <br />";
}
while($i <= 10);
?>
```

loops through a block of code if, and as long as, a specified condition is true

loops through a block of code once, and then repeats the loop as long as a special condition is true (so will always execute at least once)

Looping: for and foreach

Can loop depending on a "counter"

```
<?php
for ($i=1; $i<=5; $i++)
{
  echo "Hello World!<br/>";
}
?>
```

loops through a block of code a specified number of times

```
<?php
$a_array = array(1, 2, 3, 4);
foreach ($a_array as $value)
{
  echo "$value <br/>";
}
?>
```

view the output page

```
<?php
$a_array=array("a","b","c");
foreach ($a_array as $key => $value)
{
  echo $key . " = " . $value . "<br/>";
}
?>
```

view the output page

loops through a block of code for each element in an array

Practice - Loop

- Create a script that displays 1-2-3-4-5-6-7-8-9-10 on one line. There will be no hyphen(-) at starting and ending position

- Create a script to construct the following pattern, using a nested for loop.

```
*
* *
* * *
* * * *
* * * * *
* * * * *
* * * *
* * *
* *
*
```

User Defined Functions

Can define a function using syntax such as the following:

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
   echo "Example function.\n";
   return $retval;
}
?>
```

Can return a value of any type

```
<?php
function square($num)
{
   return $num * $num;
}
echo square(4);
?>
```

view the output page

```
<?php
function small_numbers()
{
   return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
echo $zero, $one, $two;
?>
```

view the output page

```
<?php
function takes_array($input)
{
   echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
 takes_array(array(1,2));
?>
```

view the output page

Practice - Functions

1. Write a function to check a number is prime or not
   - Note: A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself

2. Write a function to reverse a string
   - Hint: use substr($string, $start, $length);
     - Returns the portion of string specified by the start and length parameters.
     - For example, substr('abcdef', 1, 3) will return “bcd”