Programming IV: SCS2108

Handout 2: PHP

Prasad Wimalaratne, PhD, SMIEEE
spw@ucsc.cmb.ac.lk

Overview

- Introduction to Web Application Development, 3DWeb, Client Server Application development
- HTML
- PHP
- PHP-MYSQL

Terms & Definitions

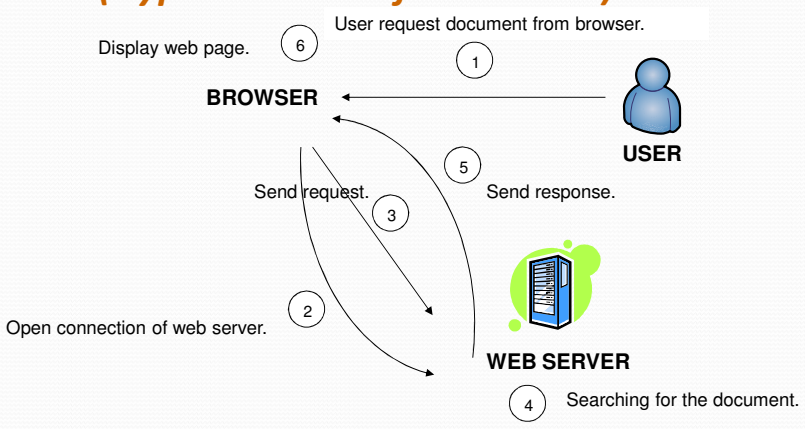
Client

A **client** is the *requesting program* in a client/server relationship, **e.g**, the user of a *Web browser* is effectively making **client** requests for pages from servers all over the Web.

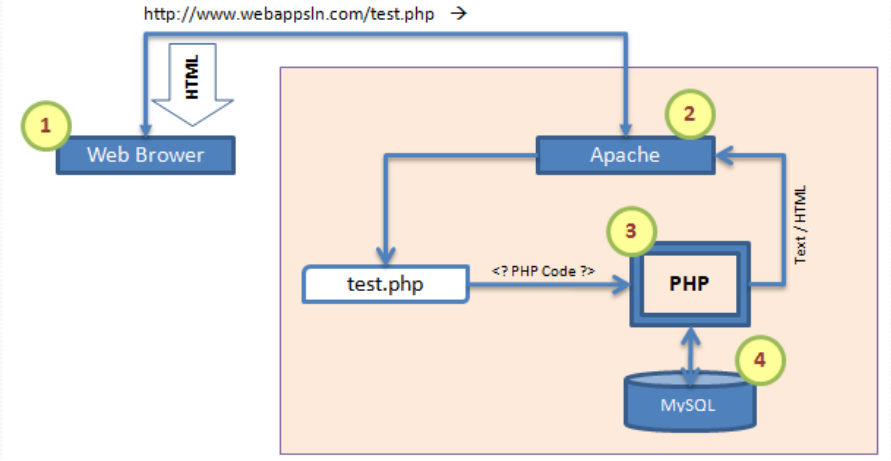
Server

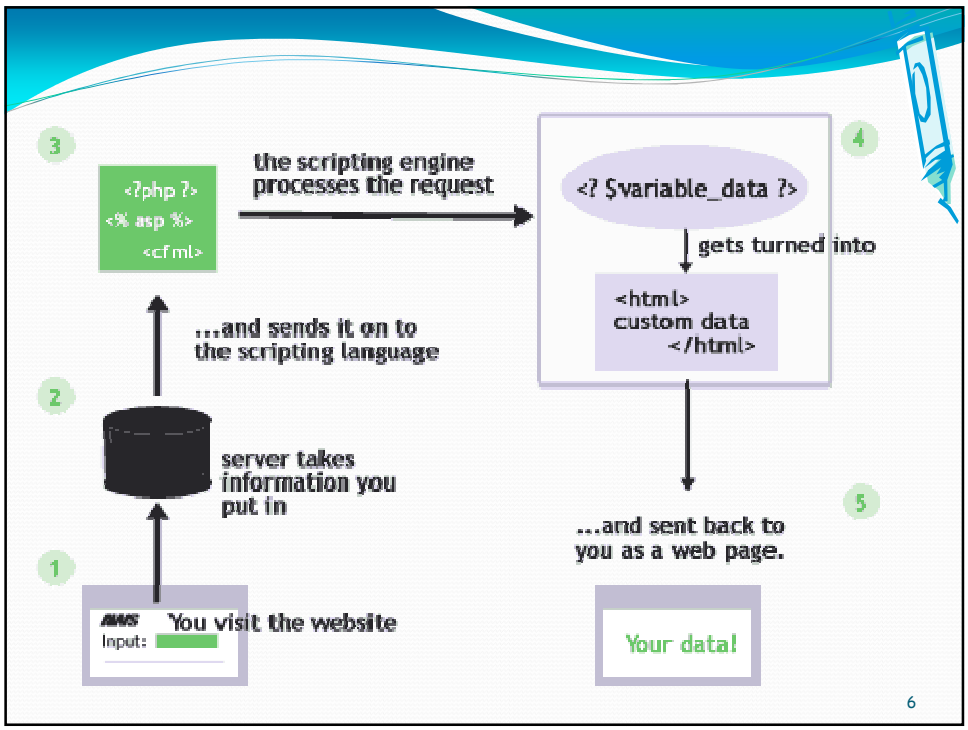
In general, a **server** is a computer program that provides services to other computer programs in the same or other computers.

HTTP (HyperText Transfer Protocol):



PHP/MYSQL


http://www.webappsln.com/test.php
HTML
1
Web Brower
2
Apache
3
test.php
<? PHP Code ?>
PHP
Text / HTML
4
MySQL


3
<?php ?>
<% asp %>
<cfml>
the scripting engine processes the request
4
<? $variable_data ?>
gets turned into
<html>
custom data
</html>
...and sends it on to the scripting language
2
server takes information you put in
1
Input:
You visit the website
5
...and sent back to you as a web page.
Your data!

Three-tiered Web Site: LAMP

Client
User-agent: Chrome

example request

```
GET / HTTP
Host: www.ucsc.cmb.ac.lk
User-Agent: Chrome
...
```

Server
Apache HTTP Server

response

Database
MySQL

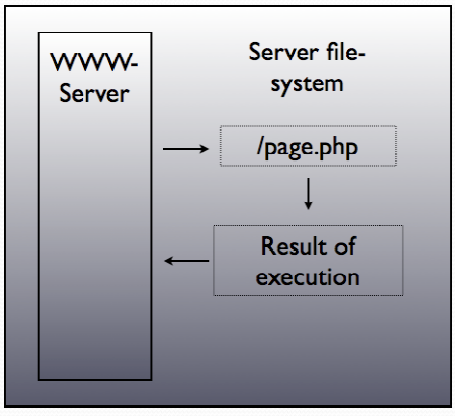
Response

Request →

www.gov.lk

Response ←

"<html><head><title>...</html>"



PHP Introduction

Programming the Web

Client-Side Code

- What is client-side code?
  - Software that is downloaded from Web server to browser and then executes on the browser client
- Why client-side code?
  - Better scalability: less work done on server
  - Better performance/user experience(UX)
  - Create UI constructs not inherent in HTML
    - Drop-down and pull-out menus
    - Tabbed dialogs
  - Visual effects, e.g. animation
  - Data validation

Programming the Web

Server-Side Code

- What is server-side code?
  - Software that runs on the server, not the client
  - Receives input from
    - URL parameters
    - HTML form data
    - Cookies
  - Can access server-side databases, e-mail servers, files, Legacy Applications, etc.
  - Dynamically builds a custom HTML response for a client

Programming the Web

Server-Side Code

- Why server-side code?
  - Accessibility
    - You can reach the Internet from any browser, any device, any time, anywhere
  - Manageability
    - Does not require distribution of application code
    - Easy to change code
  - Security
    - Source code is not exposed
    - Once user is authenticated, can only allow certain actions
  - Scalability
    - Web-based 3-tier architecture can scale out
    - **Three-tier** architecture is a client–server architecture in which the user interface (presentation), functional process logic ("business rules"), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms. E.g CodeIgnitor

Web Application Development

- Server-side
  - CGI
    - C, Perl
  - Java EE
  - ASP.NET
    - VB, C#
  - PHP
  - Ruby
  - Python
  - Web services
  - NodeJS
- Client-side
  - HTML, CSS
  - JavaScript
  - Applet
  - Flash
  - AnjularJS

PHP Introduction

- PHP is a recursive acronym for "PHP: Hypertext Pre-processor" -- It is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

- PHP code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was.

PHP Introduction

- > PHP is a server-side scripting language
- > PHP scripts are executed on the server
- > PHP supports many databases (MySQL, MongoDB, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- > PHP is open source software
- > PHP is free to download and use

PHP Introduction

- > 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 from the official PHP resource: www.php.net
- > PHP is easy to learn and runs efficiently on the server side
- **PHP on Google Cloud Platform**
  - https://cloud.google.com/php/

What is PHP

- Interpreted language, scripts are parsed at run-time rather than compiled beforehand
- Executed on the server-side
- Source-code can be made not visible by client
  - 'View Source' in browsers does not display the PHP code
- Various built-in functions allow for fast development

What does PHP code look like?

- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag

```
<?php
   …
?>
```

PHP Introduction

- Some info on MySQL which will be covered in the handout 3..

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms

PHP Introduction

- Instead of lots of commands to output HTML (as seen in C or Perl), PHP pages contain HTML with embedded code that does "something“

- The PHP code is enclosed in special start and end processing instructions **<?php** and **?>** that allow you to jump into and out of "PHP mode."

PHP Getting Started

- On windows, you can download and install WAMP. With one installation and you get an Apache webserver, database server and php.

- http://www.wampserver.com

- http://phptester.net/
- http://phpfiddle.org/

PHP Hello World

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <?php echo '<p>Hello World</p>'; ?>
 </body>
</html>
```

- Above is the PHP source code.

PHP Hello World

- It renders as HTML that looks like this:

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <p>Hello World</p>
 </body>
</html>
```


Displaying Script Results

- To return to the client the results of any processing that occurs within a PHP code block, you must use an `echo()` statement or the `print()` statement

- The **`echo()`** and **`print()`** statements create new text on a Web page that is returned as a response to a client

Displaying Script Results

- The `echo()` and `print()` statements are language constructs of the PHP programming language
- A **programming language construct** refers to a built-in feature of a programming language
- The `echo()` and `print()` statements are virtually identical except:
  - The `print()` statement returns a value of 1 if it is successful
  - It returns a value of 0 if it is not successful
- **echo** has no return value while **print** has a return value of 1 so it can be used in expressions. **echo** can take multiple parameters (although such usage is rare) while **print** can take one argument. **echo** is marginally faster than **print**.

Echo

- The PHP command 'echo' is used to output the parameters passed to it
  - The typical usage for this is to send data to the client's web-browser
- Syntax
  - void echo (string arg1 [, string argn...])
  - In practice, arguments are not passed in parentheses since echo is a language construct rather than an actual function

Echo example

```
<?php
$foo = 25;            // Numerical variable
$bar = "Hello";       // String variable

echo $bar;            // Outputs Hello
echo $foo,$bar;       // Outputs 25Hello
echo "5x5=",$foo;     // Outputs 5x5=25
echo "5x5=$foo";      // Outputs 5x5=25
echo '5x5=$foo';      // Outputs 5x5=$foo
?>
```

- **Notice** how echo '5x5=$foo' outputs $foo rather than replacing it with 25
- Strings in single quotes (' ') are not interpreted or evaluated by PHP
- This is true for both variables and character escape-sequences (such as "\n" or "\\")

Escaping the Character

- If the string has a set of double quotation marks that must remain visible, use the \ [backslash] before the quotation marks to ignore and display them.

```
<?php
$heading="\"UCSC, 35 Reid Av Colombo
7\"";
Print $heading;
?>
```

```
"UCSC, 35 Reid Av Colombo 7"
```

Comments in PHP

- Standard C, C++, and shell comment symbols

```
// C++ and Java-style comment

# Shell-style comments

/* C-style comments
     These can span multiple lines */
```


PHP Comments

- In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

PHP Variables

```
<?php
$txt="Hello World!";
$x=16;
?>
```

- In PHP, a variable does not need to be declared before assigning a value to it.
- In the example above, you see that you do not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value.

PHP Variables

- A variable name must start with a letter or an underscore "_" -- not a number
- A variable name can only contain alpha-numeric characters, underscores (a-z, A-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string) or with capitalization ($myString)

https://www.tutorialspoint.com/php/php_variable_types.htm

Variables in PHP

- PHP variables must begin with a "$" sign
- Case-sensitive ($Foo != $foo != $fOo)
- Global and locally-scoped variables
  - Global variables can be used anywhere
  - Local variables restricted to a function or class
- Certain variable names reserved by PHP
  - Form variables ($_POST, $_GET)
  - Server variables ($_SERVER) ($_SERVER is an array containing information such as headers, paths, and script locations.)
  - Etc.

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

Variable usage

```
<?php
$foo = 25;            // Numerical variable
$bar = "Hello";       // String variable

$foo = ($foo * 7);    // Multiplies foo by 7
$bar = ($bar * 7);    // Invalid expression
?>
```

superglobals

- Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- An associative array containing references to all variables which are currently defined in the global scope of the script. The variable names are the keys of the array.


```
<?php
function test() {
    $foo = "local variable";

    echo '$foo in global scope: ' . $GLOBALS["foo"] . "\n";
    echo '$foo in current scope: ' . $foo . "\n";
}

$foo = "Example content";
test();
?>
```

```
$foo in global scope: Example content
$foo in current scope: local variable
```

```
<html>
<head></head>
<body>
Regular HTML here!
<br />
<?php
// assign variable values
$institute = 'UCSC';
$address1 = 35;
$address2 = 'Reid Avenue Colombo 7';
// print output
echo "Address  <b>$institute </b>,
  <b>$address1</b> <b>$address2</b>.";
?>
</body>
</html>
```


Defining Constants

- A constant contains information that does not change during the course of program execution
- Constant names do not begin with a dollar sign
- Constant names use all uppercase letters
- Use the define() function to create a constant

```
define("CONSTANT_NAME", value);
define("VOTING_AGE",18);
define("VOTING_AGE",18,TRUE);
```

- The value you pass to the define() function can be a text string, number, or Boolean value
  - To set a constant, use the define() function - it takes three parameters: The first parameter defines the name of the constant, the second parameter defines the value of the constant, and the optional third parameter specifies whether the constant name should be case-insensitive. Default is false.

case sensitivity

```
<?php
define("GREETING","Hello world!",TRUE);
echo constant("greeting");
?>
```

- case sensitive (both user defined and PHP defined)
  - variables
  - constants
  - array keys
  - class properties
  - class constants
- Case insensitive (both user defined and PHP defined)
  - functions
  - class constructors
  - class methods
  - keywords and constructs (if, else, null, foreach, echo etc.)

PHP Concatenation

- > The concatenation operator (.) is used to put two string values together.
- > To concatenate two string variables together, use the concatenation operator:

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

PHP Concatenation

- The output of the code on the last slide will be:

```
Hello World! What a nice day!
```

- If we look at the code you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

PHP Operators

- Operators are used to operate on values. There are four classifications of operators:

- > Arithmetic
- > Assignment
- > Comparison
- > Logical

PHP Operators

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

PHP Operators

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

PHP Operators

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

Date Display

```
2016/4/1
```

```
$datedisplay=date("yyyy/m/d");
Print $datedisplay;
# If the date is April 1st, 2016
# It would display as 2016/4/1
```

```
Wednesday, April 1, 2016
```

```
$datedisplay= date("l, F n, Y");
Print $datedisplay;
# If the date is April 1st, 2016
# Wednesday, April 1, 2016
```

http://www.allabtcomputing.com/2012/05/date-and-time-functions-in-php.html

Month, Day & Date Format Symbols

M	Jan
F	January
m	01
n	1

Day of Month	d	01
Day of Month	J	1
Day of Week	l	Monday
Day of Week	D	Mon

PHP Conditional Statements

- > Very often when you write code, you want to perform different actions for different decisions.
- > You can use conditional statements in your code to do this.
- > In PHP we have the following conditional statements...

PHP Conditional Statements

- > **if** statement - use this statement to execute some code only if a specified condition is true
- > **if...else** statement - use this statement to execute some code if a condition is true and another code if the condition is false
- > **if...elseif....else** statement - use this statement to select one of several blocks of code to be executed
- > **switch** statement - use this statement to select one of many blocks of code to be executed

PHP Conditional Statements

- The following example will output "Have a nice weekend!" if the current day is Friday:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

PHP Conditional Statements

- Use the **if....else** statement to execute some code if a condition is true and another code if a condition is false.

```
<html>
<body>

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

</body>
</html>
```


PHP Conditional Statements

- If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces { }

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>

</body>
</html>
```

PHP Conditional Statements

- The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>

</body>
</html>
```


PHP Conditional Statements

- Use the switch statement to select one of many blocks of code to be executed.

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

PHP Conditional Statements

- For switches, first we have a single expression n (most often a variable), that is evaluated once.

- The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed.

- Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

PHP Conditional Statements

```
<html>
<body>

<?php
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";
}
?>

</body>
</html>
```

PHP Arrays

- > An array variable is a storage area holding a number or text. The problem is, a variable will hold only one value.
- > An array is a special variable, which can store multiple values in one single variable.
- Use the **`count()`** function to find the total number of elements in an array

```
echo " number of elements = ", count($students)
```


PHP Arrays

- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1="Saab";
$cars2="Volvo";
$cars3="BMW";
```

PHP Arrays

- In PHP, there are three kind of arrays:
- > **Numeric array** - An array with a numeric index
- > **Associative array** - An array where each ID key is associated with a value
- > **Multidimensional array** - An array containing one or more arrays

PHP Numeric Arrays

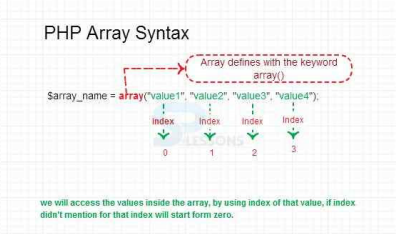
- > A numeric array stores each array element with a numeric index.
- > There are two methods to create a numeric array.

PHP Numeric Arrays

- In the following example the index is automatically assigned (the index starts at 0):

```
$cars=array("Saab","Volvo","BMW","Toyota");
```

- In the following example we assign the index manually:




```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

http://www.splessons.com/lesson/php-arrays/

PHP Numeric Arrays

- In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

- The code above will output:

```
Saab and Volvo are Swedish cars.
```


PHP Associative Arrays

- > With an associative array, each ID key is associated with a value.
- > When storing data about specific named values, a numerical array is not always the best way to do it.
- > With associative arrays we can use the values as keys and assign values to them.

PHP Associative Arrays

- In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

- This example is the same as the one above, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```


PHP Associative Arrays

The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

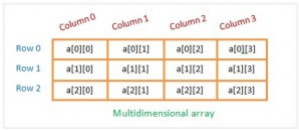
The code above will output:

```
Peter is 32 years old.
```

PHP Multidimensional Arrays

- *In a multidimensional array, each element in the main array can also be an array.*

- And each element in the sub-array can be an array, and so on.



http://programmingsphere.com/multidimensional-array-in-php/

PHP Multidimensional Arrays

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
  (
  "Griffin"=>array
  (
  "Peter",
  "Lois",
  "Megan"
  ),
  "Quagmire"=>array
  (
  "Glenn"
  ),
  "Brown"=>array
  (
  "Cleveland",
  "Loretta",
  "Junior"
  )
  );
```

PHP Multidimensional Arrays

The array above would look like this if written to the output:

```
Array
(
[Griffin] => Array
   (
   [0] => Peter
   [1] => Lois
   [2] => Megan
   )
[Quagmire] => Array
   (
   [0] => Glenn
   )
[Brown] => Array
   (
   [0] => Cleveland
   [1] => Loretta
   [2] => Junior
   )
)
```


PHP Multidimensional Arrays

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

The code above will output:

```
Is Megan a part of the Griffin family?
```

PHP Loops

- > Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

- > In PHP, we have the following looping statements:

PHP Loops

- > **while** - loops through a block of code while a specified condition is true
- > **do...while** - loops through a block of code once, and then repeats the loop as long as a specified condition is true
- > **for** - loops through a block of code a specified number of times
- > **foreach** - loops through a block of code for each element in an array

PHP Loops - While

- The while loop executes a block of code while a condition is true. The example below defines a loop that starts with
- i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

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

</body>
</html>
```


PHP Loops - While

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

PHP Loops – Do ... While

- The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true.

- The next example defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than, or equal to 5:

PHP Loops – Do ... While

```
<html>
<body>

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

</body>
</html>
```

PHP Loops – Do ... While

Output:

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```


PHP Loops - For

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init; condition; increment)
  {
  code to be executed;
  }
```

PHP Loops - For

- Parameters:
- > **init**: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- > **condition**: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- > **increment**: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)

PHP Loops - For

- The example below defines a loop that starts with i=1. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<html>
<body>

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

</body>
</html>
```

PHP Loops - For

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```


PHP Loops - Foreach

```
foreach ($array as $value)
  {
  code to be executed;
  }
```

- For every loop iteration, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop iteration, you'll be looking at the next array value.

PHP Loops - Foreach

- The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>

<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>

</body>
</html>
```


PHP Loops - Foreach

Output:

```
one
two
three
```

PHP Functions

- > We will now explore how to create your own functions.
- > To keep the script from being executed when the page loads, you can put it into a function.
- > A function will be executed by a call to the function.
- > You may call a function from anywhere within a page.

PHP Functions

- A function will be executed by a call to the function.

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

- > Give the function a name that reflects what the function does
- > The function name can start with a letter or underscore (not a number)

PHP Functions

- A simple function that writes a name when it is called:

```
<html>
<body>

<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>

</body>
</html>
```


PHP Functions - Parameters

- Adding parameters...
- > To add more functionality to a function, we can add parameters. A parameter is just like a variable.
- > Parameters are specified after the function name, inside the parentheses.

Functions

- Functions MUST be defined before then can be called
- Function headers are of the format

```
function functionName($arg_1, $arg_2, …, $arg_n)
```

  - Note that no return type is specified
- Unlike variables, function names are not case sensitive (foo(…) == Foo(…) == FoO(…))

Functions example

```
<?php
    // This is a function
    function foo($arg_1, $arg_2)
    {
      $arg_2 = $arg_1 * $arg_2;
      return $arg_2;
    }

    $result_1 = foo(12, 3);                    // Store
    the function
    echo $result_1;                  // Outputs 36
    echo foo(12, 3);                 // Outputs 36
?>
```

PHP Functions - Parameters

The following example will write different first names, but equal last name:

```
<html>
<body>

<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```


PHP Functions - Parameters

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

PHP Functions - Parameters

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>

</body>
</html>
```

This example adds different punctuation.

PHP Functions - Parameters

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?
```

Include Files

Include “test.php”;

This inserts files; the code in files will be inserted into current code.

Include (“footer.php”);

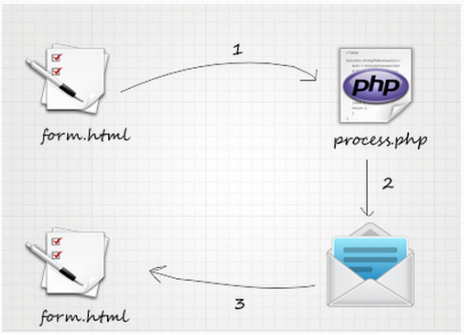
The file footer.php might look like:

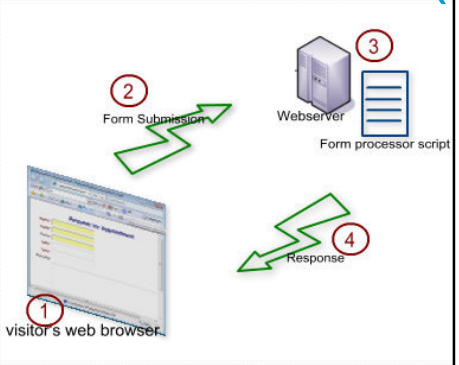
```
<hr SIZE=11 NOSHADE WIDTH="100%">
<i>Copyright © 2015-2016 hello world</i></font><br>
<i>ALL RIGHTS RESERVED</i></font><br>
<i>URL: http://www.helloworld.com</i></font><br>
```


What are forms?

- <form> is just another kind of HTML tag
- HTML forms are used to create (rather primitive) GUIs on Web pages
  1. Usually the purpose is to ask the user for information
  2. The information is then sent back to the server
- A form is an area that can contain form elements
  - The syntax is: <form *parameters*> *...form elements...* </form>
  - Form elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc
    - Other kinds of HTML tags can be mixed in with the form elements
  - A form usually contains a Submit button to send the information in he form elements to the server
  - The form’s *parameters* tell JavaScript how to send the information to the server (there are two different ways it could be sent)
  - Forms can be used for other things, such as a GUI for simple programs

forms





http://www.w3schools.com/php/php_form_complete.asp

Forms and JavaScript

- The JavaScript language can be used to make pages that "do something"
  - You *can* use JavaScript to write complete programs, but…
  - Usually you just use snippets of JavaScript here and there throughout your Web page
  - JavaScript code snippets can be attached to various form elements
    - For example, you might want to check that a zipcode field contains a 5-digit integer before you send that information to the server
- Microsoft sometimes calls JavaScript "active scripting"
- HTML forms can be used without JavaScript, and JavaScript can be used without HTML forms, but they work well together

The <form> tag

- The <form *arguments*> … </form> tag encloses form elements (and probably other HTML as well)
- The arguments to form tell what to do with the user input
  - action="*url*" (required)
    - Specifies where to send the data when the Submit button is clicked
  - method="get" (default)
    - Form data is sent as a URL with ?form_data info appended to the end
    - Can be used *only* if data is all ASCII and not more than 100 characters
  - method="post"
    - Form data is sent in the body of the URL request
    - Cannot be bookmarked by most browsers
  - target="*target*"
    - Tells where to open the page sent as a result of the request
    - *target*= _blank means open in a new window
    - *target*= _top means use the same window

The <input> tag

- Most, but not all, form elements use the input tag, with a type="…" argument to tell which kind of element it is
  - type can be text, checkbox, radio, password, hidden, submit, reset, button, file, or image
- Other common input tag arguments include:
  - name: the name of the element
  - value: the "value" of the element; used in different ways for different values of type
  - readonly: the value cannot be changed
  - disabled: the user can't do anything with this element
  - Other arguments are defined for the input tag but have meaning only for certain values of type

Text input

A text field:

```
<input type="text" name="textfield" value="with an initial value">
```

A text field: with an initial value

A multi-line text field

```
<textarea name="textarea" cols="24" rows="2">Hello</textarea>
```

A multi-line text field: Hello

A password field:

```
<input type="password" name="textfield3" value="secret">
```

A password field: ••••••

- Note that two of these use the input tag, but one uses textarea

Buttons

- A submit button:
  `<input type="submit" name="Submit" value="Submit">`
- A reset button:
  `<input type="reset" name="Submit2" value="Reset">`
- A plain button:
  `<input type="button" name="Submit3" value="Push Me">`

A submit button: Submit

A reset button: Reset

A plain button: Push Me

- submit: send data
- reset: restore all form elements to their initial state
- button: take some action as specified by JavaScript

- Note that the type is input, not "button"

Checkboxes

- A checkbox:

```
<input type="checkbox" name="checkbox"
       value="checkbox" checked>
```

A checkbox: ☑

- type: "checkbox"
- name: used to reference this form element from JavaScript
- value: value to be returned when element is checked
- Note that there is *no text* associated with the checkbox—you have to supply text in the surrounding HTML

Radio buttons

```
Radio buttons:<br>
  <input type="radio" name="radiobutton" value="myValue1">
  male<br>
  <input type="radio" name="radiobutton" value="myValue2" checked>
  female
```

Radio buttons:
male
female

- If two or more radio buttons have the same name, the user can only select one of them at a time
  - This is how you make a radio button "group"
- If you ask for the value of that name, you will get the value specified for the selected radio button
- As with checkboxes, radio buttons do not contain any text

Drop-down menu or list

- A menu or list:

```
<select name="select">
    <option value="red">red</option>
    <option value="green">green</option>
    <option value="BLUE">blue</option>
</select>
```

  - Additional arguments:
    - size: the number of items visible in the list (default is "1")
    - multiple: if set to "true", any number of items may be selected (default is "false")

Hidden fields

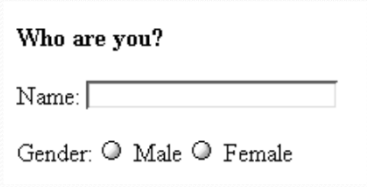
- <input type="hidden" name="hiddenField" value="nyah">
  - right there, don't you see it?

A hidden field: <-- right there, don't you see it?

- What good is this?
  - All input fields are sent back to the server, including hidden fields
  - This is a way to include information that the user doesn't need to see (or that you don't want her to see)
  - The value of a hidden field can be set (by JavaScript) before the form is submitted

A complete example

```
<html>
<head>
<title>Get Identity</title>
....
</head>
<body>
<p><b>Who are you?</b></p>
<form method="post" action="">
  <p>Name:
    <input type="text" name="textfield">
  </p>
  <p>Gender:
    <input type="radio" name="gender" value="m">Male
    <input type="radio" name="gender" value="f">Female</p>
  </form>
</body>
</html>
```



Forms

- Forms have always been one of quickest and easiest ways to add interactivity to your Web site.
- A form allows you to ask customers if they like your products, casual visitors for comments on your site
- And PHP can simplify the task of processing the data generated from a Web-based form substantially, as this first example demonstrates. Let's call this one form.htm.
-

```
•<html>
<head></head>
<body>
<form action="message.php" method="post">
Enter your message: <input type="text"
name="msg" size="30">
<input type="submit" value="Send">
</form>
</body>
</html>
```

Forms ctd…

- The "action" attribute of the <form> tag specifies the name of the server-side script (message.php in this case) that will process the information entered into the form.
- The "method" attribute specifies *how* the information will be passed.

```
<html>
<head></head>
<body>
<form action="message.php" method="post">
Enter your message: <input type="text" name="msg"
size="30">
<input type="submit" value="Send">
</form>
</body>
</html>
```


Forms ctd…

- message.php script reads the data submitted by the user and "does something with it".
-

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

<?php
// retrieve form data
$input = $_POST['msg'];
// use it
echo "You said: <i>$input</i>";
?>

</body>
</html>
```

Accessing Data Submitted

- Access submitted data in the relevant array for the submission type, using the input name as a key.

```
<form
 action="path/to/submit/page.php"
      method="get">
<input type="text" name="email">
</form>
```

```
$email = $_GET['email'];
```

Server Side Script

PHP Forms - $_GET Function

- > The built-in $_GET function is used to collect values from a form sent with method="get".
- > Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send (max. 100 characters).

PHP Forms - $_GET Function

```
<form action="welcome.php" method="get">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

```
http://www.w3schools.com/welcome.php?fname=Peter&age=37
```

Notice how the URL carries the information after the file name.

PHP Forms - $_GET Function

- The "welcome.php" file can now use the $_GET function to collect form data (the names of the form fields will automatically be the keys in the $_GET array)

```
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
```

PHP Forms - $_GET Function

- > When using method="get" in HTML forms, all variable names and values are displayed in the URL.
- > This method should not be used when sending passwords or other sensitive information!
- > However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- > The get method is not suitable for large variable values; the value cannot exceed 100 chars.

PHP Forms - $_POST Function

- > The built-in $_POST function is used to collect values from a form sent with method="post".
- > Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
- > Note: However, there is an 8 Mb max size for the POST method, by default (can be changed by setting the post_max_size in the php.ini file).

PHP Forms - $_POST Function

```
<form action="action.php" method="post">
 <p>Your name: <input type="text" name="name" /></p>
 <p>Your age: <input type="text" name="age" /></p>
 <p><input type="submit" /></p>
</form>
```

And here is what the code of action.php might look like:

```
Hi <?php echo htmlspecialchars($_POST['name']); ?>.
You are <?php echo (int)$_POST['age']; ?> years old.
```


PHP Forms - $_POST Function

- When to use **method="post"**?
- > Information sent from a form with the **POST** method is invisible to others and has no limits on the amount of information to send.
- > However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

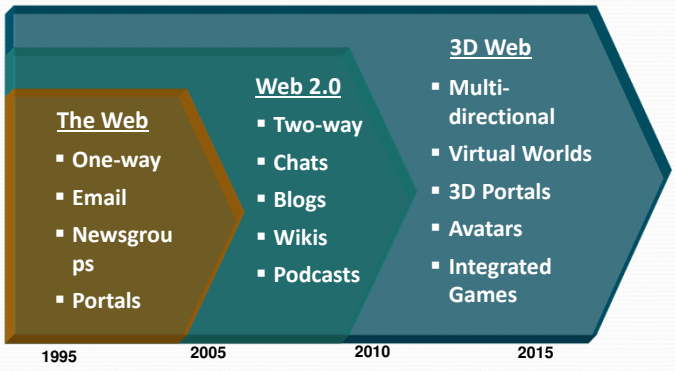
Form Data Submitted?

- We also need to check before accessing data to see if the data is submitted, use isset() function.

```
if (isset($_POST['username'])) {
  // perform validation
}
```


The Evolution Web Applications

enhancedWeb2.0+3D=WEB3D (?)



Web3D?

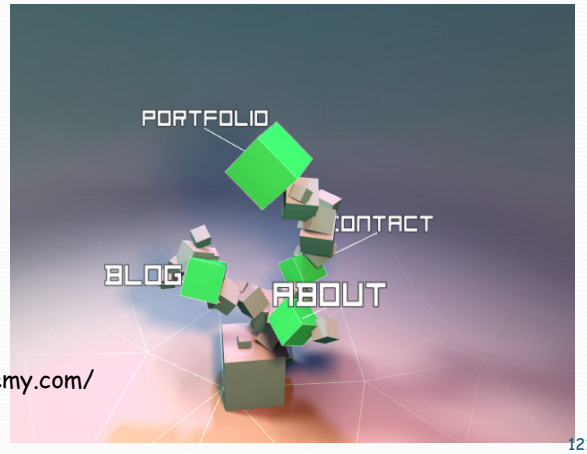
- Generally 3D Web is used for the description of applications using 3D objects in the static or dynamic HTML documents.
- "***Web3D*** *was initially the idea to fully display and navigate Web sites using 3D. By extension, the term now refers to all interactive 3D content which are embedded into web pages html, and that we can see through a web browser.*"
- "***Web 3D*** *refers to interactive 3D technology that one can use through a web browser. However, users are usually required to install a called plugin. Web 3D also can refer to technology that allows to browser the Web in 3D"*

Web3D?

- The term Web3d describes any programming or descriptive language that can be used to deliver interactive 3D objects and worlds across the internet.
- This includes open languages such as Virtual Reality Modeling Language (VRML), X3D, Java3D and - also any proprietary languages that have been developed for the same purpose come under the umbrella of Web3d.

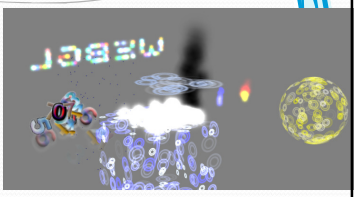
http://aleksandarrodic.com/

- See Interactive Menus



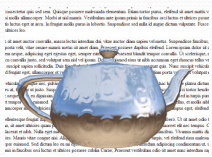
http://www.webglacademy.com/

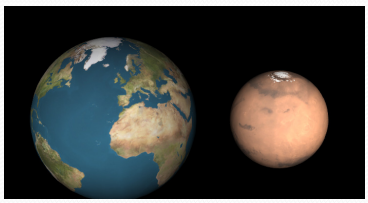
WebGL Demos



http://www.khronos.org/webgl/wiki/Demo_Repository

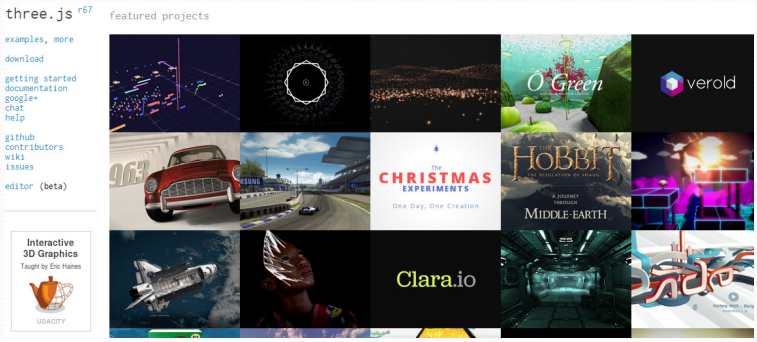
- http://goo.gl/aL1n7h



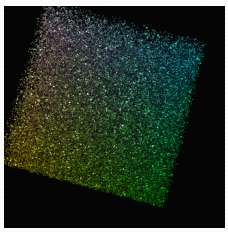


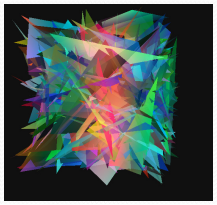
http://threejs.org/

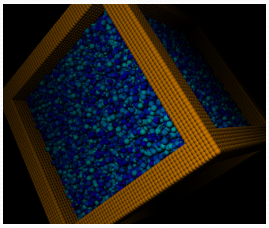
- http://goo.gl/we8pYN
- See Samples of http://threejs.org/

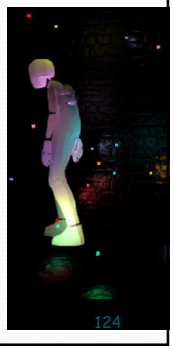


http://threejs.org/



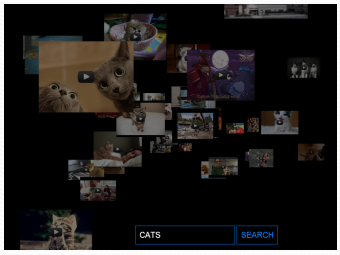






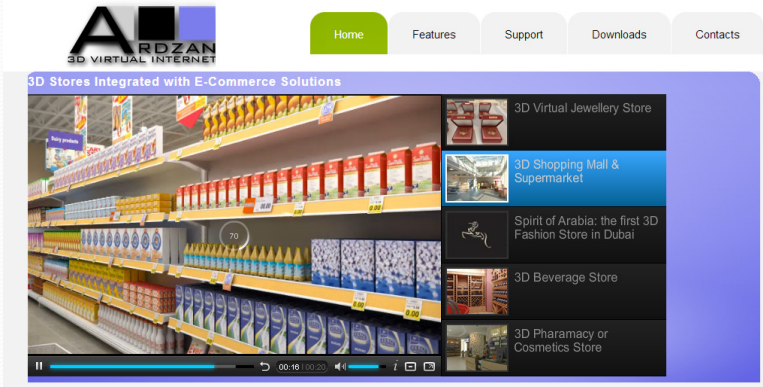
CSS3D Examples

- http://threejs.org/examples/#css3d_youtube (try searching!)
- http://threejs.org/examples/#css3d_sprites
- http://threejs.org/examples/#css3d_panorama



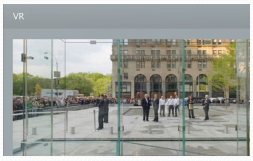
Web3D Applications: eCommerce example?

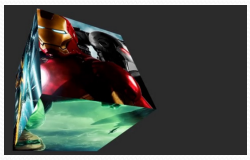
- http://www.ardzan.com/

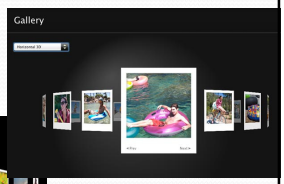


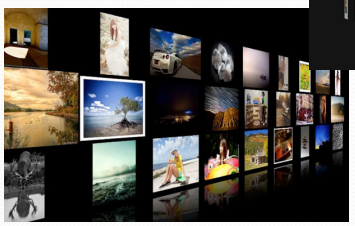
Examples of CSS 3D transforms

- http://joecritchley.com/demos/time-machi
- http://goo.gl/pQyVQU

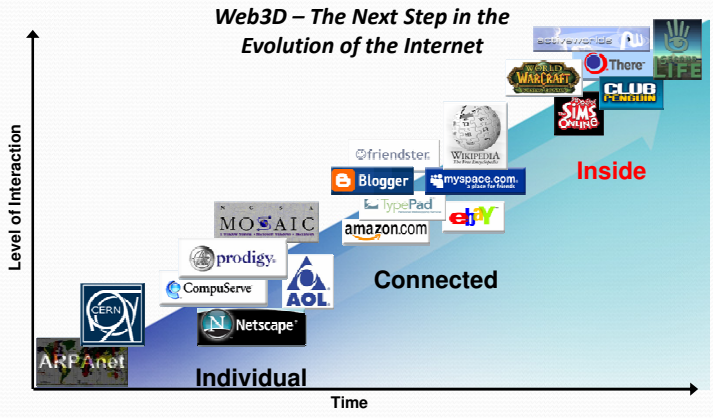








Growing importance of virtual worlds



3D Social Media : Second Life

Web 3.D: a virtual world created by its users

(and not created by "few smart builders" of the actual web 2.0 websi

1. User's connection

Web3D client

Integrated Application

2. Creation of his own avatar

3. Access to a virtual interactive environment

4. The user meet other avatars of friends or customers, discuss via voice or text chat, visit places, partecipate in business meetings, purchase products and use online services. Every user build its structures and expand the world.

e.g: http://secondlife.com/

Greater leverage across an experts' organization

Turning into a Enterprise 2.0 means

Increased Efficiency
+
Improved Access
+
Enhanced Learning and Collaboration
+
Decreased Cost

Realtime collaboration on tasks

Faster response to company and client issues

Enables connection of globally dispersed teams

Tutorial

- Complete Introductory and forms section of W3Schools tutorial
- http://www.w3schools.com/php/

- Further reading
- http://www.tutorialspoint.com/php/
- Php.net