LECTURE: Static and Dynamic Webpage

Static webpages never change.

In static websites, the way to have a different result is to have multiple versions of the same page, like right1.php and wrong1.php, which show pretty much the same thing, and are different from each other just a little bit.

In dynamic webpages, things can change, like a score counter

Score: 2

what is the capital of Samar?
a._____
b._____

Examples of dynamic webpages

Yahoo Mail and Facebook are both dynamic webpages. When you log into Facebook, the webpage shows your facebook page. When somebody else logs into Facebook, the webpage shows a very different page. But it's showing the same file. We will do something like this later.

To create static webpages, all you need to use is HTML.

To create dynamic webpages, we need to use a programming language, like PHP.

here is a sample of PHP code. All it does is print the words Hello world! on screen:

```
<?php echo "Hello world!"; ?>
```

Go ahead and to try this. Create a new file (let's call it try.php - a nice, short, easy to type filename) in /mnt/sdcard/htdocs. Type the above code into scratch.php and open it in the browser by going to

http://localhost:8080/try.php

All PHP code must be placed between PHP delimiters, or what I call open and close php. They're like open and close parentheses, open and close quotes, and open and close brackets.

This is open php
<?php
and this is close php
?>

We will now enter the world of dynamic web pages and PHP.

Understanding Variables

A variable is a container for information. You can think of variables like very small files - you can read them and write to them, but they can contain only a single number or a string of letters, and they exist only in the computer's memory. Once you shut down the program they're in, they're gone.

In the following piece of code, we create a variable named score, and give it the value 1. Then we print out its value on screen. ('echo' is the command for print.)

Go ahead and type the following code into try.php and view it in a browser.

```
<?php
	$score=1;
	echo $score;
?>
```

The result is

1

All variables in PHP have a $ sign before their name. That's how PHP knows that they're variables.

In this piece of code, we add 1 to the value of score and save the result back into score.

```
<?php
	$score=1;
	echo $score;
	$score=$score+1;
	echo $score;
?>
```

The result is

12

That's not 12 - that's 1, and then 2. We forgot to add a newline between the two echos.

```
<?php
	$score=1;
	echo $score;
	echo "<br>";
	$score=$score+1;
	echo $score;
```

```
?>
```

Now the screen shows

```
1
2
```

Let's add 1 more to the score:

```
<?php
	$score=1;
	echo $score;
	echo "<br>";
	$score=$score+1;
	echo $score;
	echo "<br>";
	$score=$score+1;
	echo $score;
?>
```

The result is

```
1
2
3
```

There's a better way to see the value of a variable than 'echo'. Sometimes, variables contain nothing at all, and if you echo its value, you get nothing at all. To solve this problem, we can use something called var_dump, which is short for, well, variable dump.

To use var_dump, type this into try.php

```
<?php
	$score=1;
	var_dump($score);
?>
```

You get:

```
int(1)
```

This means that $score contains an integer (meaning it's a whole number - not a fraction or a decimal) with the value of 1.

On the other hand, if we do this:

```
<?php
	$score="Hello world!";
```

```
    var_dump($score);
?>
```

We get:

string(12) "Hello world!"

which means $score contains a string of 12 letters, which form the words Hello world!.

EXERCISE:

First, edit the file question1.php as shown below:

```
---------
File: question1.php
---------
Score: <?php echo $_REQUEST['score']; ?>
<br>
<br>What is the capital of Leyte?
<br><a href=wrong1.php>Catbalogan</a>
<br><a href=right1.php>Tacloban</a>
---------
```

Try entering this url into the browser: http://localhost:8080/question1.php?score=1
it will show "Score: 1"
Change the url to http://localhost:8080/question1.php?score=2
it will now show "Score: 2"

LECTURE:
Understanding Request Variables

score=2 is called a url parameter. It is a way for one webpage to give another webpage information. For example, the code below is how right1.php can tell question2.php that the user got the first question correctly, and that score is now 1.

```
<a href="http://localhost:8080/question2.php?score=1">Next</a>
```

When you click on that link, you go to question2.php. Also, question2.php receives information that says that score is now 1. This makes question2.php just a little more intelligent. It can now look like this:

```
----------------------
Score: 1

What is the capital of Samar?
a. _______
b. _______
```

These variables can be accessed like this:

<?php echo $_REQUEST['score']?>

and so are called Request Variables. Request Variables are automatically created when you add information after a url.

Many url parameters can be added after a url, like this:

http://localhost:8080/question2.php?score=1&firstname=johnny&lastname=bravo

This automatically creates the request variables:

$_REQUEST['score']
$_REQUEST['firstname']
$_REQUEST['lastname']

now, edit the rest of the files

File: right1.php

RIGHT!!!

next

File: wrong1.php

WRONG!!!

next

In this next file, we will use the value of $_REQUEST['score'] 3 times. To make it easier for us, we will copy its value to a variable with a much shorter name.

File: question2.php

<?php $s=$_REQUEST['score'];?>
Score: <?php echo $s;?>

What is the capital of Samar

<a href=right2.php?score=<?php echo $s?>>Catbalogan

<a href=wrong2.php?score=<?php echo $s?>>Tacloban

File: right2.php

RIGHT!!!

<a href=finalscore.php?score=<?php echo $_REQUEST['score']+1; ?>>Next
---
File: wrong.php

WRONG!!!

<a href=finalscore.php?score=<?php echo $_REQUEST['score']; ?>>Next
---
File: finalscore.php

Final Score: <?php echo $_REQUEST['score'];?>