INTRO TO PHP CONT.

DR. ANDREW BESMER

- Operators
- Conditionals
- Logical

OPERATORS

OPERATORS

Operator	Purpose	Example
`+`	Addition	`$x + $y;`
`-`	Substraction	`$x - $y;`
`*`	Multiplication	`$x * $y;`
`/`	Division	`$x / $y;`
`%`	Modulus	`$x % $y;`
`=`	Assignment	`$x = $y;`
`.`	Concatentation	`$x . $y;`

- Modulus Example

COMBINED

Operator	Example	Becomes
`+=`	`$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;`

- Combined example

(INC)(DEC)REMENT

Operator	Purpose	Example
`++`	Increment	`$x++;`
`--`	Decrement	`$x--;`

- Increment or Decrement will increase or decrease the value of the variable by 1
- No assignment is needed since the operator operates directly on the variable
- Can increment or decrement either pre or post
  - Pre before statement and use in expression
  - Post after statement and use in expression

(INC)(DEC)REMENT

- Examples

CONDITIONALS

CODE BLOCK

Code Block: One or more statements surrounded by curly braces `{` and `}`

```php
<?php

{
    echo "hi";
    echo "csci 241";
}
```

- Use with control structures to indicate which statements to execute
- When you open a curly you must close it
- Curly braces should surrond complete statements and should not be interspersed

IF STATEMENTS

- If statements help control program logic

```
if(expression)
{
    /*
    * The code in these curly braces run only
    * if expression is true
    */
}
else
{
    /*
    * The code in these curly braces run only
    * if expression is false
    */
}
```

- Which code will run if the expression is `false`?

IF STATEMENTS

- Can conditionally condition!

```
if(expression) //Always checked
{
    //Some code
}
else if(expression2) //Checked only if prior is false
{
    //Some code
}

...

else if(exprssion7) //Checked only if all prior are false
{
    //Some code
}
else //Runs if all tests resulted in false
{
    //Some code
```

COMPARRISON OPERATORS

Operator	Purpose	Example
`==`	Equal	`$x == $y;`
`!=`	Not Equal	`$x != y;`
`<`	Less Than	`$x < $y`
`>`	More Than	`$x > $y`
`<=`	Less Than or Equal To	`$x <= $y`
`>=`	More Than or Equal To	`$x >= $y`

COMPARRISON OPERATORS

Operator	Purpose	Example
`===`	Exactly Equal	`$x === $y;`

- Difference between `==` and `===` is the addition of the type comparrison (or no type juggling)

```php
$x = 5;
$y = "5";
var_dump((bool)($x==$y)); //true
var_dump((bool)($x===$y)); //false
```

OPERATORS WARNINGS

- Careful of
  - = operator, why?
  - ` operator, why?

```
$x = 5;
$y = 6;
var_dump((bool)($x=$y)); // true

`rm -rf ~/somefolder`; // Deletes the folder!
```

LOGICAL

LOGICAL OPERATORS

Operator	Purpose	Example
`&&`	And	`$x && $y`
`\\`	Or	`$x || y`
`!`	Not	`!$x`

- Can be combined to form more complex logic

OPERATOR PRECEDENCE

Priority	Operators
1	`()`
2	`! ++ -- (type)`
3	`* / %`
4	`+ - .`
5	`< <= > >=`
6	`== != === !==`
7	`&&`
8	`||`

- Examples

EXISTANCE

- Check `$_GET` and `$_POST` data to see if a name or key value pair was transmitted using `isset()`
- `isset()` returns true or false… perfect for conditionals