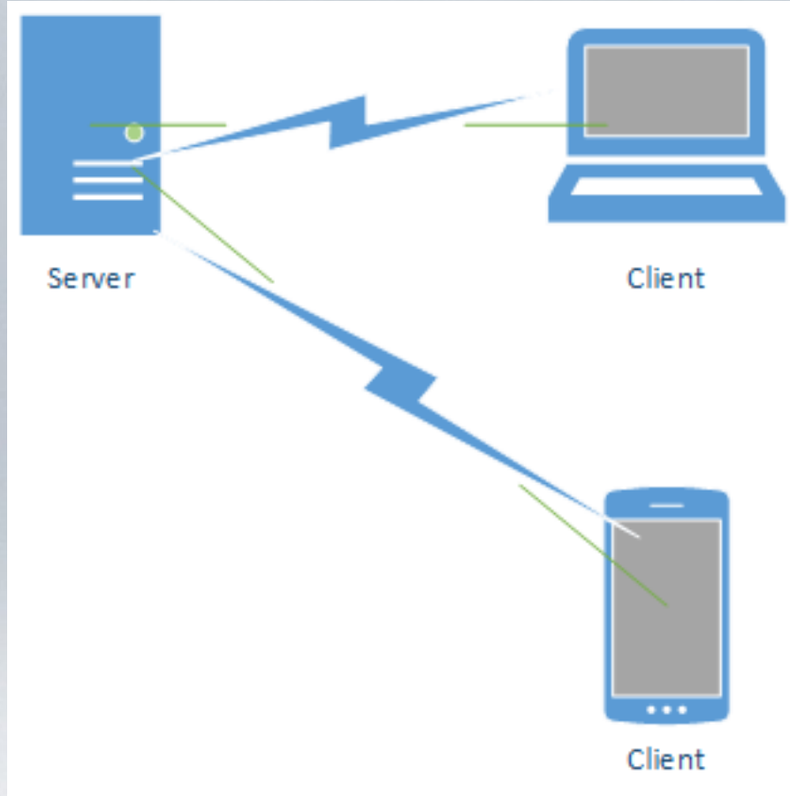
# IA FOR THE WEB

PHP
Fall 2014

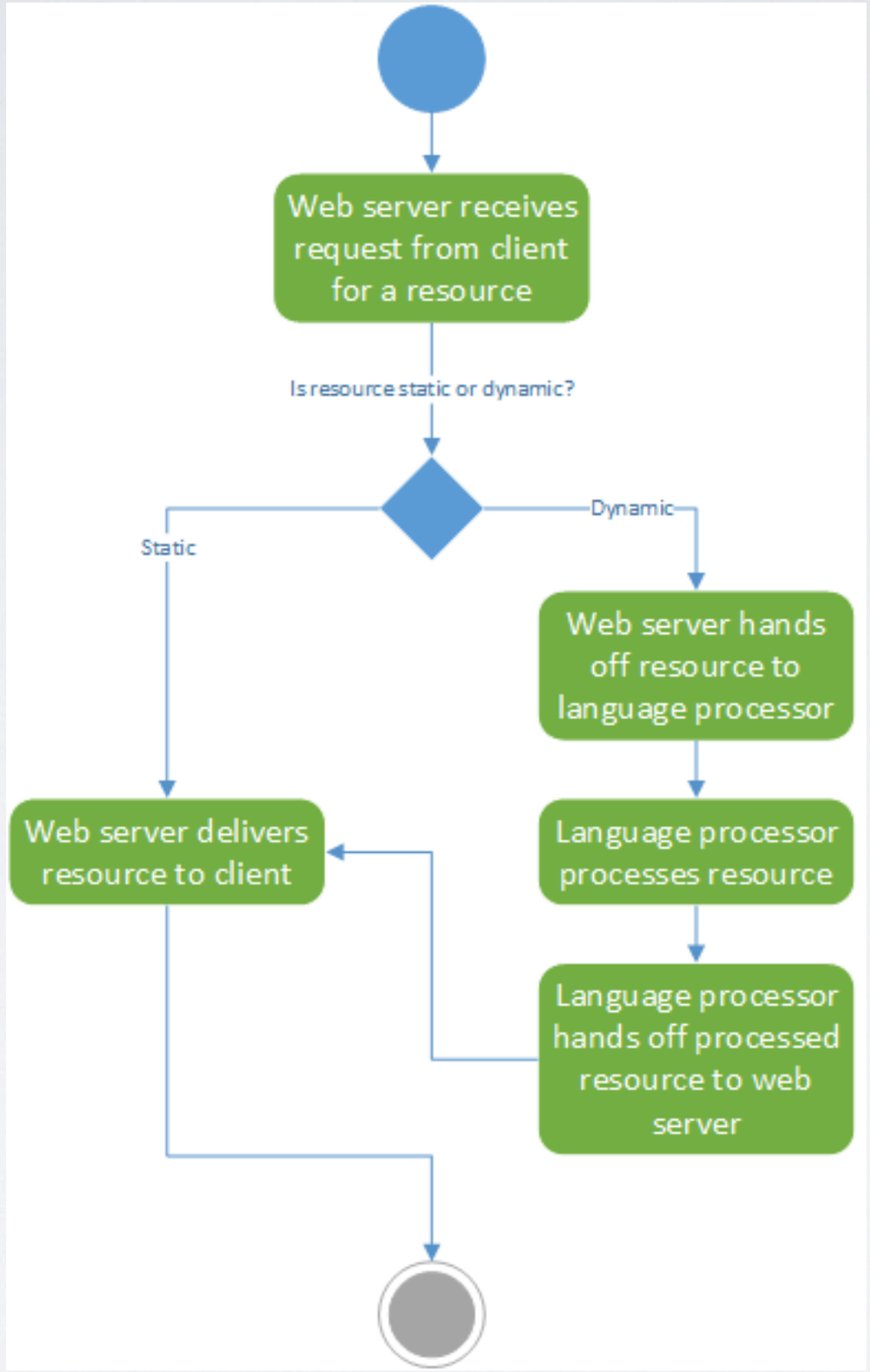
# PLAN

- This week we'll discuss PHP as a language
- Next week we'll explore using PHP in the server environment

# CLIENT-SERVER MODEL



# Web Server Activity Diagram




- PHP Hypertext Processor

# POWERS

WordPress

# YOUR FIRST PHP SCRIPT

```
<?php

print “Hello World!”;

?>
```

# YOUR SECOND PHP SCRIPT

```
<?php $name = "Grant"?>
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>Hello <?php print $name?>!</title>
</head>
<body>
<h1>Hello <?php print $name?>!</h1>
</body>
</html>
```

# DECLARING A PHP BLOCK

```
<?php

//Code here

?>
```

# STATEMENTS

- PHP is an imperative language
- Imperative languages use statements to direct the computer
- End with a semicolon (`;`)
- Semicolon not needed right before `?>`

# DIFFERENT STATEMENTS

- `print 4 + 3;`
- `$name = “Abraham Lincoln”;`
- `sort($classes);`

# COMMENTS

```
// Single-line comment (C++ style)
// Need to repeat it for more lines

/* Multi-line comment. (C style)
This will end
whenever you want it to. */

# Single-line comment (UNIX/Perl style)
# Need to repeat it for more lines
```

# VARIABLE NAMES

- Indicated by **`$`**
- Can start with letters or _
- Can include letters (ASCII or extended ASCII)
- Are case sensitive. **`$Counter`** and **`$counter`** are different
- Cannot be keywords or **`$this`**

# DECLARING VARIABLES

- Unlike some languages, there is no keyword to declare a variable
- `$age = “4”;`
- `$people = array(“Joe”, “Emily”, “Siobhan”, “Jamal”);`
- `$numPeople = count($people);`

# BASIC DATA TYPES

- Boolean
- Integer
- Floating point numbers
- Strings

# BOOLEAN

- `$socratesIsHuman = true;`
- Booleans are case insensitive
- False values: FALSE, false, 0, 0.0, “”, “0”, empty array, NULL
- True values: TRUE, true, any non-zero number (whether positive or negative)

# INTEGER

- `$numberOfStudents = 10;`
- `$numberOfStudents = -10;`
- `$numberOfStudents = 0b00001010; //binary`
- `$numberOfStudents = 0xA; //hexidecimal`

# FLOATING POINT NUMBERS

- `$volume = 1.3267000;`
- `$volume = 1.3267E-3; //E notation`

# STRINGS

```
$name = "Charlotte";

$name = 'Charlotte';

$name = <<<EOD
```

Sometimes you need to have a long string that is easy to read. This string will continue until the terminator is found.

```
EOD;
```

# SINGLE VS DOUBLE QUOTES

```
$age = 5;
$display = “I am $age years old.<br>”;
print $display;
```

```
$display = ‘I am $age years old.<br>’;
print $display;
```

Output:

```
I am 5 years old.
I am $age years old.
```

# OPERATORS

- Arithmetic
- Assignment
- Comparison
- Logical
- String

# ARITHMETIC

| Operator | Name | Example |
|---|---|---|
| `+` | Addition | `9 + 3;` |
| `-` | Subtraction | `4 - $height;` |
| `*` | Multiplication | `$height * $width;` |
| `/` | Division | `6 / 3;` |
| `%` | Modulus (Remainder) | `$rowNumber % 2` |

# ASSIGNMENT

| Operator | Example | Result |
|---|---|---|
| `=` | `$page = 4;` | `$page` is set to 4 |
| `+=` | `$page += 2;` | 2 is added to `$page` |
| `-=` | `$page -= 3;` | 3 is subtracted from `$page` |

# INCREMENTING

| Operator | Name | Example |
| --- | --- | --- |
| `++` | Increment | `$counter++;` |
| `--` | Decrement | `$triesLeft--;` |

# COMPARISON

| Operator | Name | Example |
|---|---|---|
| `==` | Equal | `4 == "4"` |
| `!=` | Not equal | `4 != 3` |
| `===` | Identical | `4 === 4`<br>`4 === "4"`<br>`4 === 4.0` |
| `!==` | Not identical | `3 !== 3` |
| `<` | Less than | `5 < 6` |
| `>` | Greater than | `5 > 6` |
| `<=` | Less than or equal to | `4 <= 4` |
| `>=` | Greater than or equal to | `4 >= 5` |

# LOGICAL

| Operator | Name | Example |
|---|---|---|
| `&&` | And | `true && true` |
| `\|\|` | Or | `true \|\| false`<br>`false \|\| true` |
| `xor` | Exclusive or | `true xor false` |
| `!` | Not | `!false` |

# STRING (CONCATENATION)

```
$greeting = “Hello” . “ World”;
print $greeting;
```

Output:

```
Hello World
```

# CONTROL STRUCTURES

# IF

```
if(/* expression */){
   / *Do something. */
}

if(true && 4 < 5){
  print “yep!”;
}
```

# IF-ELSE

```
if(/* expression */){
   /*Do something. */
}else{
   /*Do this. */
}

if(4 > 5){
  print “yep!”;
}else{
  print “nope!”;
}
```

# IF-ELSE IF-ELSE

```
if(/* expression */){
   /*Do something. */
}else if(/* expression 2 */){
   /*Do something else. */
}else{
   /*Do this. */
}

if(4 > 5){
  print “greater”;
}else if(4 == 4){
  print “equal”;
}else{
  print “lesser”;
}
```

# LOOPS

# WHILE

Counter

Expression
that must be true
to continue

```
$i = 10;
while($i >= 0){
   print $i . "<br>"
   $i--;
}
```

What to do
each time
the loop runs

What to do
with the counter
after each time the
loop runs

# FOR

Counter

Expression
that must be true
to continue

What to do
with the counter
after each time the
loop runs

```
for($i = 10; $i >= 0; $i--){
  print $i . “<br>”
}
```

What to do each time the loop runs

# FOREACH LOOP

```
foreach($cartItems as $item){
  print “<li>$item</li>”;
}
```

No condition expression needed!

# CALLING A FUNCTION

```
$name = strtoupper($name);
```

Variable to which the output of the function is assigned

Name of function being called

Input parameter(s) (Can be variables or literal values)

# BUILT-IN FUNCTIONS

- PHP has **many** built-in functions
- See the Function Reference for more information
- Be sure to read what a function does, what its parameters are, and what it returns

# USER-DEFINED FUNCTIONS

Function name

Parameter name(s)

```
function isEven($number){
    if($number % 2 == 0){
      return true;
    }else{
      return false;
    }
}
```

Return statement

# USER-DEFINED FUNCTIONS

```
function isEven($number){
  if($number % 2 == 0){
  return true;
  }else{
    return false;
  }
}

if(isEven(10)){
  print “10 is even!”;
}
```

# PARAMETERS

- Can take multiple parameters. Separate them with commas:
```
function compareTwoNumbers($number1, $number2)
```
- Can specify a default value if none is supplied:
```
function sortList($order = “ascending”)
```
- Can take zero parameters:
```
function sayHello(){
  print “Hello!”;
}
```

# RETURN STATEMENT

- Tells the function to return a value
- The value can be any type (integer, string, array, object, etc.)
- Using **`return`** by itself will return back to where the function was called without sending a value back

# WORKING WITH STRINGS

- Individual characters in the string have positions in the string, starting from 0
- PHP has many string functions

# GET STRING LENGTH

```
$name = ‘Ice King’;
print strlen($name);
//Outputs 8
```

# CHANGE CASE

```
$name = ‘Ice King’;
print strtoupper($name);
//Outputs ICE KING
print strtolower($name);
//Outputs ice king

$name = ‘finn’;
print ucfirst($name);
//Outputs Finn

$name = ‘princess bubblegum’;
print ucwords($name);
//Outputs Princess Bubblegum
```

# REPLACE TEXT

```
$name = ‘Bradley Manning’;
$name = str_replace(‘Bradley’, ‘Chelsea’, $name);
```

# COUNT WORDS

```
$title = ‘The Hitchhiker’s Guide to the Galaxy’;
print str_word_count($title);
//Outputs 6
```

# FIND POSITION OF SUBSTRING

```
/* strpos returns the numeric position of the start
of the substring or false if the substring does not
exist. */

$title = ‘The Hitchhiker’s Guide to the Galaxy’;
$pos = strpos($title, ‘The’);
if($pos === false){
   print “‘The’ was not found.”;
}else{
   print “‘The’ was found at position $pos”;
}
```

# ARRAYS

- A variable holds one item at a time
- An array holds multiple items

# DECLARING ARRAYS

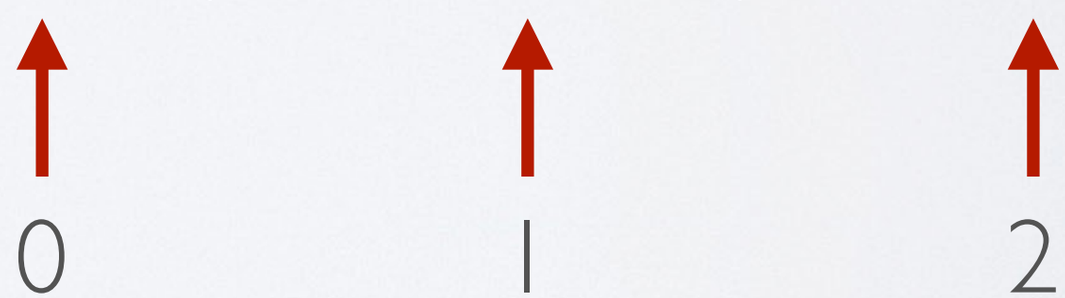
```
$names = array(“Joe”, “Eliza”,
“Naomi”);

/* Can also declare an empty array
to add to later */
$names = array();
```

# ARRAY POSITIONS

- Array items have positions called keys, starting from 0

```
$names = array("Joe", "Eliza", "Naomi");
```



# GETTING AN ITEM AT A SPECIFIC POSITION

```
$names = array(“Joe”, “Eliza”, “Naomi”);
print $names[2];
//Outputs Naomi
```

# CHANGING AN ITEM AT A SPECIFIC POSITION

```
$names = array(“Joe”, “Eliza”, “Naomi”);
$names[0] = “Eleanor”;
//$names is now Eleanor, Eliza, Naomi
```

# ADDING TO THE END

```
$names = array(“Joe”, “Eliza”, “Naomi”);
$names[] = ‘Siobhan’;
//$names is now Joe, Eliza, Naomi, Siobhan
```

# GETTING THE NUMBER OF ELEMENTS

```
$names = array(“Joe”, “Eliza”, “Naomi”);
print count($names);
//Outputs 3
```

# KEYS CAN BE STRINGS

```
$student = array(‘name’ => ’Duane’, ‘age’ => 30);

print $student[‘name’];
//Outputs Duane

$student[‘name’] = ‘Dwayne’;
$student[‘standing’] = ‘Freshman’;
```

# ARRAY ELEMENTS CAN BE HETEROGENOUS

```
$stuff = array(1, “Fish”, 4.5, true);
```

# ARRAYS CAN CONTAIN ARRAYS

```
$duane = array(‘name’ => ’Duane’, ‘age’ => 30);
$lamar = array(‘name’ => ’Lamar’, ‘age’ => 22);

$students = array(‘duane’=>$duane, $lamar=>$lamar);

//Or
$students = array(
   $duane = array(‘name’ => ’Duane’, ‘age’ => 30),
   $lamar = array(‘name’ => ’Lamar’, ‘age’ => 22)
);
```