INTRO TO PHP & MYSQL

TJ Nicolaides
Original Curriculum: Izzy Johnston

PART 1: WHAT IS PHP?

Intro to PHP & MySQL

DOWNLOADS

- Practice Files:
- XAMPP: https://www.apachefriends.org/index.html
- Text Editors:
  - PCs: Notepad++, jEdit, Aptana, Brackets
  - Macs: TextWrangler, jEdit, Aptana, Atom, Sublime Text
  - Browser-based: Cloud9, CodeAnywhere

COURSE ITINERARY

- June 14, AM: **What are PHP and MySQL?**
- June 14, PM: **Getting started with MySQL**
- June 21, AM: **Database manipulation**
- June 21, PM: **Advanced PHP / PHP in "the real world"**

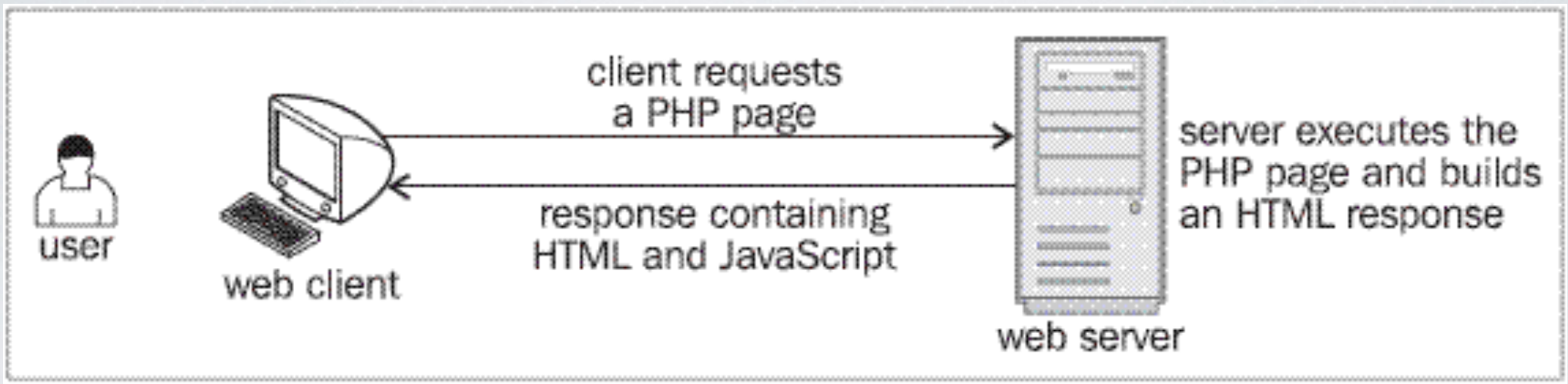
PHP - DEFINED FOR NERDS

- **PHP**: originally stood for "**P**ersonal **H**ome **P**age"
- Now stands for "**H**ypertext **P**rocessor"
- Server-side language
- Supports database languages like MySQL
- Usually interpreted, instead of compiled (like Java or Go)

```
<div class="wrapper">
    <div class="header">
        <h1>Results Page</h1>
    </div>
    <div class="content">
        <?php

            $f = $_GET['degrees'];
            $f = htmlspecialchars($f, ENT_QUOTE

            echo "You entered " . $f . " degree

            $c= ($f-32)*5/9;
            $c = round($c, 2);

            echo "That is " . $c. " Celsius.";
        ?>

    </div>

    <div class="footer">
        <img src="assets/cc_by.png" alt="Creat
on, in conjunction with Girl Develop It! </div>
  </div>
body>
>
```

PHP - IN ENGLISH, PLEASE

- Dynamic content
- Database integration
- Do more with HTML
- Why code thousands of pages when you could make just one?



VISUALIZE

Client-side vs. Server-side languages

CLIENT-SIDE VS. SERVER-SIDE

- Actions that a web browser can interpret
- Ability to see the source code (Dev Tools)
- Can see / execute without being live on the web

CLIENT-SIDE VS. **SERVER-SIDE**

- Actions too complicated for most web browsers
- Source code rendered as HTML + JS
- Need a public (or local) server

OKAY LET'S WRITE SOME PHP

VARIABLES

Format:

$name

Valid naming:

$myvariable, $myVar,
$my_variable, $myVar2

Invalid naming:

$3amigos, $_heh

- Variables are symbolic names containing data in your script
- They can hold numbers (integers, floats), words (strings), or true/false values (booleans) and more
- Names start with letters, not numbers or characters like underscores

ASSIGNING VARIABLES

- Assign new value:
  - **$a = 1;**
  - **$b = “hello”;**
- Reassign value from another variable:
  - **$d = $c;**

MATH OPERATIONS

- **$c = $a + $b;**
- **$d = $c - ($a + $b);**
- **$e = $f * $g;**
- **$h = $i / $j;**

STRING OPERATIONS

- Assign values:
  - **$a = “Hello”;**
  - **$b = “my name is:”;**
- Concatenate:
  - **$c = $a . $b;**
- (Don’t forget the spaces)
  - **$a = “Hello “;**

SCRIPT FLOW: IF/ELSE

```
if ($some_parameter) {

    // do one thing

} else {

    // do another thing

}
```

- Checks inside the parentheses to determine which code block to execute

- Is $some_parameter true or not?

SCRIPT FLOW: IF/ELSE

```
$a = 2; $b = 3;

if ($a > $b) {

    // ‘A’ is greater than ‘B’!

} else {

    // ‘B’ is greater than or
equal to ‘A’.

}
```

- Use comparisons inside the parentheses to create a true / false value
- Which code block will execute here?

SCRIPT FLOW: WHILE LOOPS

```
$a = 0;

while($a < 10) {

    echo $a;

    $a = $a + 1;

}
```

- When once just isn't good enough
- Execute a code block over and over until we tell it to continue on
- If condition to continue is never reached: INFINITE LOOP

MATHEMATICAL OPERATORS

Symbol	Meaning	Example
+	*Addition*	*$x + 2 = 4*
-	*Subtraction*	*$x - 2= 0*
*	*Multiplication*	*$x * 2 = 4*
/	*Division*	*$x / 2 = 1*
%	*Modulus (Remainder)*	*5 % $x = 1*
++	*Increment*	*$x ++ = 3*
--	*Decrement*	*$x -- = 1*

*For all examples, $x=2

COMPARISON OPERATORS

Symbol	Meaning	Example
==	*Equals?*	*$x == $y FALSE*
!=	*Does not equal?*	*$x != $y TRUE*
>	*Is greater than?*	*$x > $y FALSE*
<	*Is less than?*	*$x < $y TRUE*
&&	*AND*	*if ($x<5 && $y >5)*
||	*OR*	*if ($x<5 || $y >5)*
!	*NOT*	*if !($x==$y)*

*For all examples, $x=2 and $y=3

FUNCTIONS: DON'T REPEAT YOURSELF

```
function doSomething() {

    echo "Do something!";

}

echo "Don't just stand
there. ";

doSomething();
```

- Self-contained block of code you can trigger whenever
- Take complex or repetitive operations out of the script flow to simplify your code

FUNCTIONS

```
function multiply($a, $b) {
    $c = $a * $b;
    return $c;
}
$product = multiply(3, 5);
echo $product;
```

- Functions can do an action or they can just return a value
- Functions are just another type of variable in your script - follow the same naming practices
- Pass arguments in the parentheses

LET'S DEVELOP IT: HTML FORM

form.html

```
<form action="result.php"
method="get">

<label>Enter Degrees in
Fahrenheit</label>

<input name="degrees" type="text"
/>

<input  type="submit" value="Get
Degrees in Celsius”/>

</form>
```

result.php

```
<?php

 $f = $_GET['degrees'];

 $f = htmlspecialchars($f,
ENT_QUOTES, 'UTF-8');

?>
```

LET'S DEVELOP IT: PRINT DATA

```
<?php

 $f = $_GET['degrees'];

 $f = htmlspecialchars($f, ENT_QUOTES, 'UTF-8');

 echo "You entered " . $f . " degrees Fahrenheit.";

?>
```

LET'S DEVELOP IT: MANIPULATE DATA

```
<?php

  $f = $_GET['degrees'];

  $f = htmlspecialchars($f, ENT_QUOTES, 'UTF-8');

  echo "You entered " . $f . " degrees Fahrenheit.”;

  $c= ($f-32)*5/9;

  echo "That is " . $c. " Celsius.";

?>
```

LET'S DEVELOP IT: USING A FUNCTION

```
<?php

  $f = $_GET['degrees'];

  $f = htmlspecialchars($f, ENT_QUOTES, 'UTF-8');

  echo "You entered " . $f . " degrees Fahrenheit.”;

  $c= ($f-32)*5/9;

  $c = round( $c, 2);

  echo "That is " . $c. " Celsius.";

?>
```

FORMS AND PHP

- Use action = "filename.php" to tell the HTML form where to send info and method = "get" to show data in URL
  - **<form action="result.php" method="get">**
- Give form elements name attributes
  - **<input type="text" name="degrees">**
- Call the name by the method in the file from action, assign it to a variable
  - **$variable = $_GET['degrees'];**

STARTER FILES

Download from Github as a ZIP

Refer to the completed example, also at Github

DEVELOP THIS NEXT

- Using your temperature converter files as a starting point:
    - Make a form to convert inches to centimeters. Round to the nearest 100th (2 digits after the decimal)
    - Make a form that asks for your first name and last name. Concatenate the two with a friendly greeting.
    - NOTE: duplicate your **/temperature-converter** directory so you don't overwrite your previous work, for a total of three