PHP I

How does a Web Server Work

- Up until now we have been using a webserver to perform two simple tasks
  - Convert the URL to the local file and retrieve it
  - Control access to the website
- Both of these require no processing of the pages themselves
  - Commonly known as serving static pages
- You don't even need a webserver to do this
  - HTML can be viewed just fine in a browser

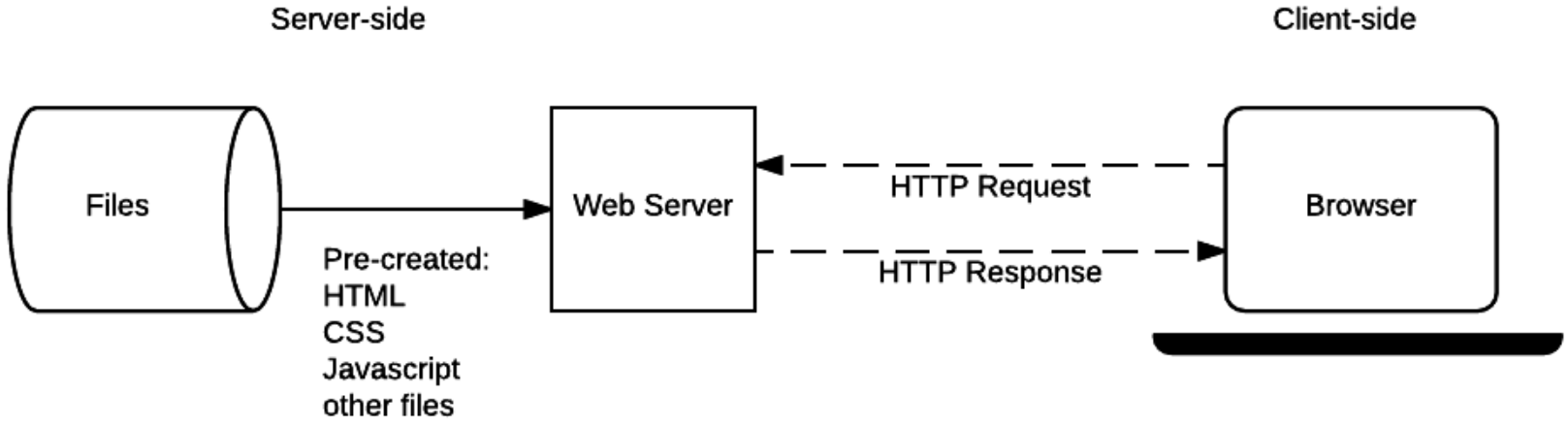

Image from https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction

Dynamic Webserver

- Web servers are capable of much more, including
  - Executing code prior to returing a request
  - Compressing files before sending them
  - Running SSL for HTTPs and everything that goes with it
  - Logging

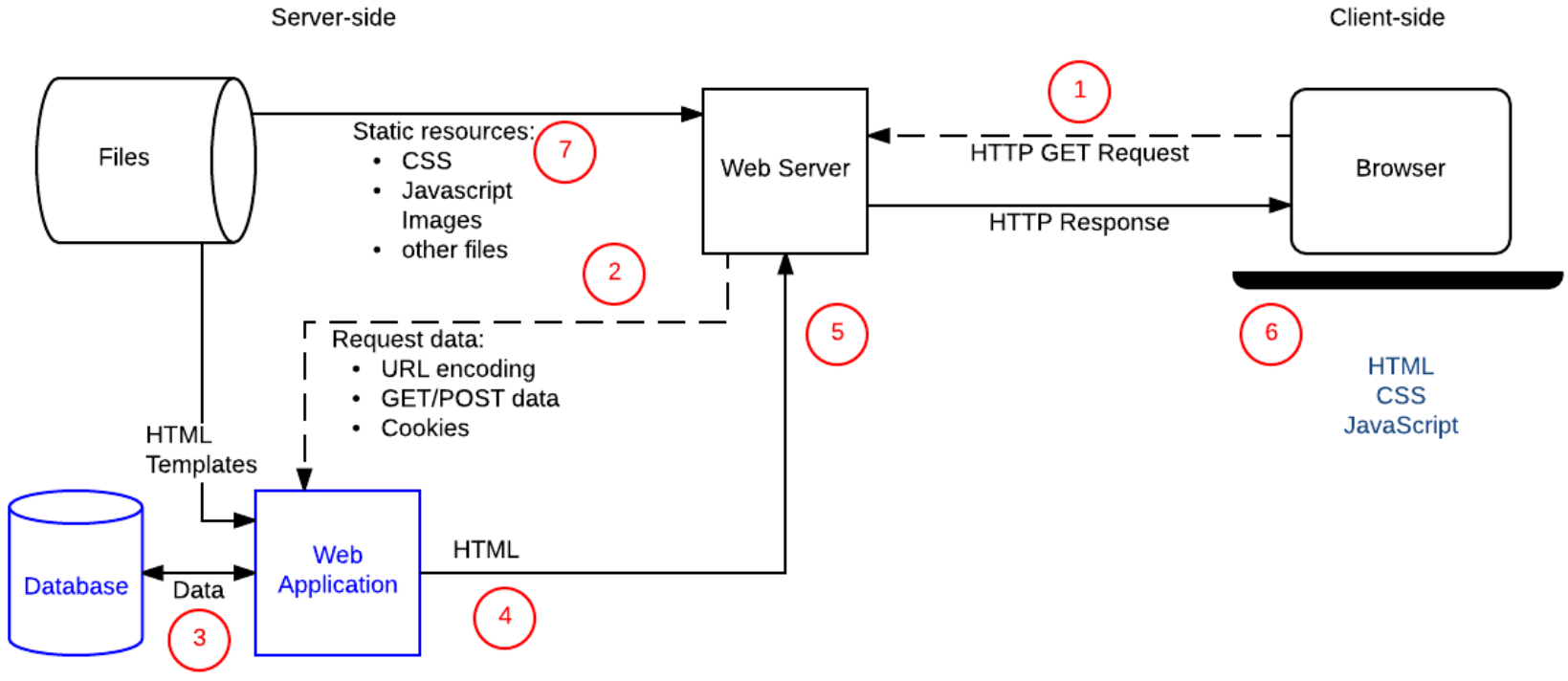

Image from https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction

CGI vs Embedded

- Prior to PHP, most web applications used an API known as the Common Gateway Interface (CGI)
  - Perls support of CGI was one of the big reasons for its popularity
  - CGI scripts are forked from the webserver
    - Can be safer (runs in a separate process)
    - Can be much more dangerous (is running as any script file would)
- PHP, and many other languages that have come after it, run inside the webserver, in what is known as embedded mode
  - Is generally safer for beginning programmers, as the webserver offers many protections
  - Has access to the server API

PHP History

- Invented in 1994 by Rasmus Lerdorf
- Stands for PHP HyperText Processor
  - Orginally stood for Personal Home Page
- Primarily used on server-side -Can be used in both command-line and GUI scripting as well

PHP Versions

- PHP 4 reached end of life in 2008, any website using it is very insecure!!
- PHP 5 came out in 2004, PHP 5.6 came out in 2014
  - GL has PHP 5.4 in the SWE environment
- PHP 6 never came out, delayed over and over and over
- PHP 7 released in 2015, 7.2 comes out the end of this month
  - Supported in most webhost now

Major Users of PHP

- Facebook
- Wikipedia (and all sites running MediaWiki)
- Wordpress and Drupal
- Baidu
- Many, many more
  - up to 80% of all sites by some estimates

Embedding PHP in HTML

- PHP is often found interspersed within an HTML document
  - PHP can be written without any HTML
- The webserver pre-processes all files ending with `*.php`
  - Any code between the PHP tags is run, and output is placed directly into the HTML page
  - The page is sent to the client with no PHP in it, only the results of executing the PHP

PHP Tags

- To indicate a section of PHP code, a PHP tag is needed
- There are many different styles but you should only use `<?php`
  - This is the only one garunteed to always work!!
- Others you may see
  - `<?`
  - `<%`

Running at https://www.csee.umbc.edu/~bwilk1/433/php_examples/simple.php

```
<!DOCTYPE html>
<?php $color = "#057f14" ?>
<html>
    <head>
    <style>
        #example1{background-color:
            <?php echo $color; ?>}
    </style>
    </head>
    <body>
        <p id="example1">My background is
        <?php echo $color; ?></p>
    </body>
</html>
```

Comments in PHP

Comments in PHP can either be

- C/C++ Style
  - `//` for single line
  - `/* */` for multiline
- Scripting Language Style
  - `#` for single line

In [1]:
```
//This is a comment, nothing will happen
/*
 This is also a comment
 echo "HELLO"
 won't print
*/
 # This is a comment too
echo "HI"
```

Out[1]:
```
HI
```

Variables

- All variables in PHP **must** start with `$` (dollar sign)
  - Dollar signs can be stacked to use the value of one variable as the name of another (variable variables)
- The first character after the dollar sign cannot be a number
- The variables scope is
  - global if not declared in a function
  - the function if declared in one

In [2]:
```
$a = "var";
$$a = "value";
$$$a = "whaaat";
echo $a;
echo $$a , $var;
echo $$$a, $$var, $value;
```

Out[2]:
```
var
```

Out[2]:
```
value
```

Out[2]:
```
value
```

Out[2]:
```
whaaat
```

Out[2]:
```
whaaat
```

Out[2]:
```
whaaat
```

Data Types

- PHP has 8 data types
- Primitives/Scalars
  - boolean
  - integer
  - float
  - string
- Other types
  - Arrays
  - Objects
  - Resources (files, etc.)
  - NULL

In [3]:
```
2001
```

Out[3]:
```
2001
```

In [4]:
```
-309
```

Out[4]:
```
-309
```

In [5]:
```
+33
```

Out[5]:
```
33
```

In [6]:
```
0755
```

Out[6]: 493

In [7]:
```
+010
```

Out[7]: 8

In [8]:
```
0xFF
```

Out[8]: 255

In [9]:
```
0xcc
```

Out[9]: 204

In [10]:
```
3.14
```

Out[10]:
```
3.14
```

In [11]:
```
0.016
```

Out[11]:
```
0.016
```

In [12]:
```
0.314E1
```

Out[12]:
```
3.14
```

In [13]:
```
16.0e-3
```

Out[13]:
```
0.016
```

In [14]:
```
'This is a string'
```

Out[14]:
```
"This is a string"
```

In [15]:
```
"This is also a string"
```

Out[15]:
```
"This is also a string"
```

In [16]:
```
false
```

Out[16]:
```
false
```

In [17]:
```
true
```

Out[17]:
```
true
```

In [18]:
```
0?'T':'F'
```

Out[18]: "F"

In [19]:
```
0.0?'T':'F'
```

Out[19]: "F"

In [20]:
```
516?'T':'F'
```

Out[20]: "T"

In [21]:
```
"0"?'T':'F'
```

Out[21]:
```
"F"
```

In [25]:
```
"0.0"?'T':'F';
```

Out[25]:
```
"T"
```

In [23]:
```
""?'T':'F'
```

Out[23]:
```
"F"
```

In [26]:
```
(array())?'T':'F'
```

Out[26]:
```
"F"
```

In [27]:
```
NULL?'T':'F'
```

Out[27]:
```
"F"
```

Casting

- PHP is weakly typed and will coerce data types when possible
- To force casting to another datatype, use

```
(datatype) $var;
```

In [28]:
```
"5" * 2
```

Out[28]:
```
10
```

In [29]:
```
'3.14' + 5
```

Out[29]:
```
8.14
```

In [30]:
```
(int) "5"
```

Out[30]: 5

In [33]:
```
(int) "4+16"
```

Out[33]: 4

In [34]:
```
(int) "3.14"
```

Out[34]: 3

In [35]:
```
(int) "19.99"
```

Out[35]: 19

In [36]:
```
(float)"3.14"
```

Out[36]:
```
3.14
```

In [37]:
```
(float) "3.14 is pi"
```

Out[37]:
```
3.14
```

In [38]:
```
(string) 1
```

Out[38]:
```
"1"
```

In [39]:
```
(string) true
```

Out[39]:
```
"1"
```

In [40]:
```
(string) false
```

Out[40]:
```
""
```

```
In [41]: (bool) 0
```

```
Out[41]: false
```

```
In [42]: (bool)3.14
```

```
Out[42]: true
```

```
In [43]: (bool)1
```

```
Out[43]: true
```

```
In [44]: (bool)""
```

```
Out[44]: false
```

```
In [45]: (bool)"NO"
```

```
Out[45]: true
```

Constants

- Constants are created by calling the `define` function, passing in a constant name and value
  - Constant names don't start with `$`

```
define("NAME",value);
```

- There are numerous predefined constants that are useful for introspection
  - `__LINE__` gives the current line of the file
  - `__FILE__` gives the file currently being executed
  - `__FUNCTION__` provides the function name
  - `__CLASS__` provides the class name

In [46]:
```
define("PI",3.14);
define("TU", "TUESDAY");
echo "Sometimes " . PI . " day is on a " . TU;
echo "I am running from " . __LINE__;
```

Out[46]:
```
Sometimes 3.14 day is on a TUESDAY
```

Out[46]:
```
I am running from 4
```

Selection Structures

- PHP has 3 selection structures
  - If/Then/Else
  - Switch
  - Ternary Operator
- These behave mostly like other langugaes, but can be mixed with HTML to produce efficent custom HTML pages

In [47]:
```
$today = "Tuesday";
if($today == "Monday"){
    echo "Happy Monday!";
}
else if($today == "Tuesday"){
    echo "One Day Down!";
}
else{
    echo "Start Counting Down Towards the Weekend!";
}
```

Out[47]:
```
One Day Down!
```

In [50]:
```
$today = "Tuesday";
if($today == "Monday"){
    echo "Happy Monday!";
}
elseif($today == "Tuesday"){
    echo "One Day Down!";
}
else{
    echo "Start Counting Down Towards the Weekend!";
}
```

Out[50]:
```
One Day Down!
```

In [51]:
```
$today = "Tuesday";
$mon = "Monday";

switch($today){
    case $mon:
        echo "Happy Monday!";
        break;
    case "Tuesday":
        echo "One Day Down!";
        break;
    default:
        echo "Start Counting Down Towards the Weekend!";
}
```

Out[51]:
```
One Day Down!
```

Running at https://www.csee.umbc.edu/~bwilk1/433/php_examples/if.php

```
<?php
if(date('s') % 2 == 0){
?>
<p>You opened this page during an <strong>even</strong> second.
<?php
}
else{ ?>
<p>You opened this page during an <strong>odd</strong> second.
<?php } ?>
```

Selection Structures and HTML Practice

- Write a simple PHP page that intermixes PHP and HTML
  - Get a random number using the PHP function `rand()`
  - If the number is even, display a blue `h1` tag with the words "It's Even!"
  - If the number is odd, display a green `h2` tag with the words "It's Odd!"

Quoting

- In PHP, strings can be declared using
  - single quotes
  - double quotes
  - HERE docs
- Variables are interpolated in both double quoted string and here docs
  - They are not interpolated in a single quoted string
  - Can use `${}` notation like in bash

In [52]:
```
$lyric = "Teapot";
echo 'I\'m a little $lyric';
```

Out[52]:
```
I'm a little $lyric
```

In [53]:
```
$lyric = "Teapot";
echo "I'm a little $lyric";
```

Out[53]:
```
I'm a little Teapot
```

HERE Docs

- HERE Docs are used to make multi-line strings more conviently
- A HERE doc starts with three less-than symbols (<<<) followed by an identifier
- It ends when the indentifier is repeated, and is the only thing on the line besides as semicolon

In [54]:
```
$var = "this";
$a_long_string = <<<HERE
This string is like a quoted
string, it will interpolate all
variables in it, like $var.
It only ends when HERE is on a
line of it's own
HERE;
echo $a_long_string;
```

Out[54]:
```
This string is like a quoted
string, it will interpolate all
variables in it, like this.
It only ends when HERE is on a
line of it's own
```

String Operators

- The concatenation operator in PHP is the dot (.)
  - This can be combined with the assignment operator (.=)
- Strings can be compared using the standard operators
  - == Compares equality and coerces if needed
  - === Compares equality, but never coerces
  - <,<=,>,>= compare for lexicographic order, coerce if necessary

In [55]:
```
$alphabet = "a" . 'b' . 'c' . 'd';
echo $alphabet;
$alphabet .= 'e';
echo $alphabet;
```

Out[55]: abcd

Out[55]: abcde

In [56]:
```
"3" == 3;
```

Out[56]:
```
true
```

In [57]:
```
"3" === 3;
```

Out[57]:
```
false
```

In [65]:
```
"3" <= 4;
```

Out[65]:
```
true
```

String Practice

- Using concatenation, conversion, and interpolation, use the following variables to print the string "Today is the 20th day of class"

In [69]:
```
$num1 = "15";
$num2 = 5;
$rest = "day of class";
$num = $num1 + $num2;
echo "Today is the ${num}th ".  $rest;
```

Out[69]:
```
Today is the 20th day of class
```

String Functions

- PHP provides many robust, built-in, string functions
- Some common ones are:
  - `explode` - Similar to string_split in other languages
  - `htmlentities` - Encodes anything that needs to be encoded for HTML (e.g. <, >, etc.)
  - `html_entity_decode` - Converts HTML encoded characters back into their regular string values
  - `join` - Joins an array together (can also use `implode`)
  - `parse_str` - Turns a query string into an array
  - `strip_tags` - Removes all HTML and PHP tags from a string
  - `trim` - Removes whitespace from both ends of the string
- A full list can be found at http://php.net/manual/en/ref.strings.php

In [70]:
```
echo print_r(explode(',', "John,Paul,George,Ringo"),true);
```

Out[70]:
```
Array
(
    [0] => John
    [1] => Paul
    [2] => George
    [3] => Ringo
)
```

In [71]:
```
$html = "This is an example of an HTML tag 3 < 4 <html>. It must be closed like th
is: </html>";
echo htmlentities($html);
```

Out[71]:
```
This is an example of an HTML tag 3 &lt; 4 &lt;html&gt;. It must be closed lik
e this: &lt;/html&gt;
```

In [72]:
```
$html = "Some characters, like the amperstand, need to be escaped in HTML, like &a
mp;";
echo html_entity_decode($html);
```

Out[72]:
```
Some characters, like the amperstand, need to be escaped in HTML, like &
```

In [73]:
```
parse_str("key1=value1&key2=value2",$result);
echo print_r($result,true);
```

Out[73]:
```
Array
(
    [key1] => value1
    [key2] => value2
)
```

In [74]:

```
$wiki = <<<WIKI
<div class="thumbinner" style="width:302px;"><a href="/wiki/File:Manchester_Mark2.
jpg" class="image"><img alt="" src="//upload.wikimedia.org/wikipedia/en/thumb/d/d
8/Manchester_Mark2.jpg/300px-Manchester_Mark2.jpg" width="300" height="238" class
="thumbimage" srcset="//upload.wikimedia.org/wikipedia/en/d/d8/Manchester_Mark2.jp
g 1.5x" data-file-width="355" data-file-height="282" /></a>
<div class="thumbcaption">
<div class="magnify"><a href="/wiki/File:Manchester_Mark2.jpg" class="internal" ti
tle="Enlarge"></a></div>
The Manchester Mark 1 was one of the world's first stored-program computers.</div>
</div>
</div>
<p>The <b>Manchester Mark 1</b> was one of the earliest <a href="/wiki/Stored-prog
ram_computer" title="Stored-program computer">stored-program computers</a>, develo
ped at the <a href="/wiki/Victoria_University_of_Manchester" title="Victoria Unive
rsity of Manchester">Victoria University of Manchester</a> from the <a href="/wik
i/Manchester_Small-Scale_Experimental_Machine" title="Manchester Small-Scale Exper
imental Machine">Small-Scale Experimental Machine</a> (SSEM) or "Baby" (operationa
l in June 1948). It was also called the <b>Manchester Automatic Digital Machine</b
>, or <b>MADM</b>.<sup id="cite_ref-FOOTNOTELavington199820_1-0" class="referenc
e"><a href="#cite_note-FOOTNOTELavington199820-1">[1]</a></sup> Work began in Augu
st 1948, and the first version was operational by April 1949; a program written to
 search for <a href="/wiki/Mersenne_primes" class="mw-redirect" title="Mersenne pr
imes">Mersenne primes</a> ran error-free for nine hours on the night of 16/17 June
 1949.</p>
<p>The machine's successful operation was widely reported in the British press, wh
ich used the phrase "electronic brain" in describing it to their readers. That des
cription provoked a reaction from the head of the University of Manchester's Depar
tment of Neurosurgery, the start of a long-running debate as to whether an electro
nic computer could ever be truly creative.</p>
WIKI;
echo strip_tags($wiki);
```

Out[74]:

In [75]:
```
$evil_user = "I am going to try an run some code <?php echo 'Hi There';?>";
echo strip_tags($evil_user);
```

Out[75]:
```
I am going to try an run some code
```

In [76]:
```
$str_with_spaces = "             There are so many spaces on either side of me
                ";
echo trim($str_with_spaces);
```

Out[76]:
```
There are so many spaces on either side of me
```

PHP On GL

- The PHP mod is already part of the Apache server running on GL,so no set up is needed
- Pages must end with .php and be word-readable
  - `chmod a+r`
- Can only use the `<?php` version of the tag

Getting Help with PHP

- The PHP documentation is very well done and has information on every built-in PHP function
  - http://php.net/manual/en/
- The comments below are usually quite helpful