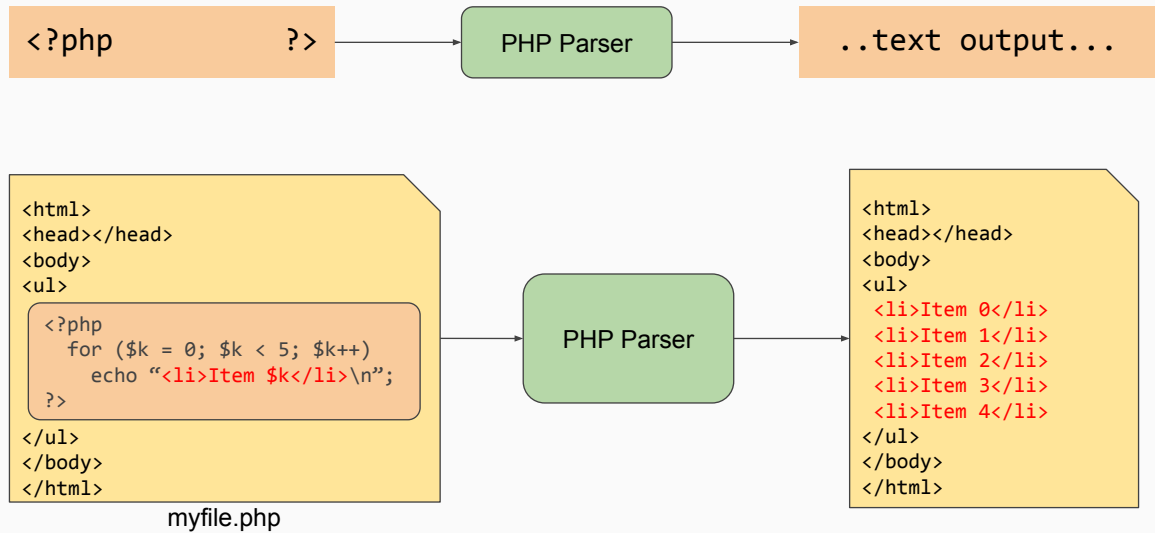


PHP

PHP 1.0: *Tag Replacement Parser*

PHP Parser: Tag Replacement

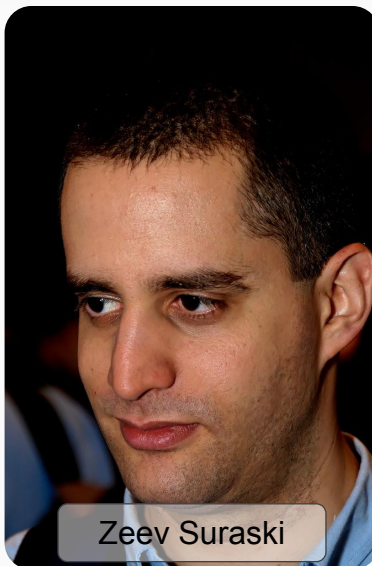


3

Key Developers of PHP



Rasmus Lerdorf



Zeev Suraski



Andi Gutmans

4

Release History

- 1994 PHP 1.0 (Rasmus Lerdorf: **P**ersonal **H**ome **P**age)
- 1995 PHP 2.0 (**P**HP: **H**yper**T**ext **P**reprocessor)
- 1998 PHP 3.0 (**Z**eev Suraski & **A**ndri Gutmans)
 - Support Databases: Oracle, SyBase, PostgreSQL, ODBC, ...
- 2000 PHP 4.0 (Zend Engine 1.0)
- 2004 PHP 5.0 (Zend Engine 2.0) → 2015 PHP 5.6
- *PHP 6.0 never released*
- 2015 PHP 7.0

5



6

JavaScript

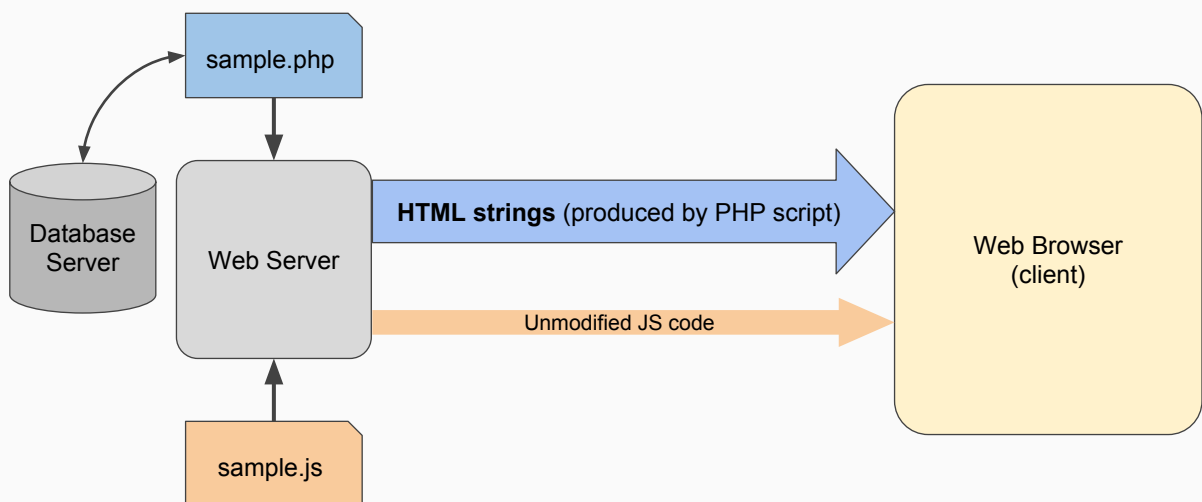
vs.

PHP

- Run on web client (browser)
 - **Client-Side** Scripting
 - APIs for manipulating DOM tree
 - Can generate dynamic and **interactive** content
 - New contents generated on the browser
 - No direct access to files on the server
 - SPAs (Single Page Applications)
- Run on web server, output HTML
 - **Server-Side** Scripting
 - Insert HTML output in place
 - Can generate **dynamic** page content
 - But may require a **round-trip** HTTP traffic
 - Direct access to files/**database** on the server
 - LAMP (Linux-Apache-MySQL-PHP)

7

JavaScript vs. PHP



8

Practical Use of PHP Scripts

- CRUD operation to database
- Collect user input (forms)
- Encrypt data

9

PHP Data Types and Variables

- [String](#)
- Integer
- Float
- Boolean
- [Array](#)

```
$name = "Lakers";  
var_dump($name);           // output string(6) "Lakers"  
  
$nine = 0b1001;  
var_dump($nine);           // output int(9)  
$fifty = 0x32;             // var_dump: int(50)  
$teen = 021;               // var_dump: int(17)  
$adult = 21;               // var_dump: int(21)  
  
// Keywords are case-inSenSitiVE  
$isPrime = true;  
$authenticated = TRUE;  
$checked = False;  
var_dump ($checked);       // output bool(false)  
  
// Equal vs. identical  
var_dump ("1" == true);    // output bool(true)  
var_dump ("1" === true);   // output bool(false)
```

Debugging functions: `var_dump()` or `print_r()`

10

String: conversion, concatenation, expansion

```
$val = 2 + "50";           // evaluate to int(52)
$val = "2" + "50";         // evaluate to int(52)

$val = "2" . "50";         // evaluate to string(3) "250"

$msg = "Hello World";
dump_var('Say $msg');      // string(8) "Say $msg"
dump_var("Say $msg");      // string(15) "Say Hello World"
```

11

Arrays

- PHP arrays are **associative** (key/value pair)
- Key type is either integer or string
- Allocated using `array()` or `[]`
- Functions
 - `count($arr)`: count the number of elements in the array
 - `unset($arr)`: delete the array
 - `array_push($arr, val1, val2, val3);`
 - `array_pop($arr)`
 - [Many more...](#)

12

Arrays (PHP Arrays are associative arrays)

```
$values = array(20, 5, 16);  
// OR $values = [20, 5, 16];  
  
for ($k = 0; $k < count($values); $k++)  
    echo $values[$k] . "\n";  
foreach ($values as $v)  
    echo $v . "\n";  
foreach ($values as $k => $v)  
    echo $values[$k] . "\n";
```

```
$currency = [  
    "USD" => "US Dollar",  
    "JPY" => "Japanese Yen",  
    "ARS" => "Argentina Peso"  
];  
foreach ($currency as $code => $desc)  
    echo "Currency code $code is $desc\n";
```

// The following two declarations
// are equivalent

```
array("Huron", "Superior", "Ontario")  
  
array(  
    0 => "Huron",  
    1 => "Superior",  
    2 => "Ontario"  
);
```

13

Array Sort Functions

```
$surfaceArea = ["Huron" => 23000, "Ontario" => 7340, "Michigan" => 22300,  
    "Erie" => 9910, "Superior" => 31700];  
  
// Sort by value  
asort($surfaceArea); // ["Ontario" => 7340,  
    // "Erie" => 9910,  
    // "Michigan" => 22300,  
    // "Huron" => 23000,  
    // "Superior" => 31700]  
  
// Sort by key  
ksort($surfaceArea); // ["Erie" => 9910,  
    // "Huron" => 23000,  
    // "Michigan" => 22300,  
    // "Ontario" => 7340,  
    // "Superior" => 31700]
```

14

Array functions: extract() and compact()

```
$book = [  
    "author" => "Knuth",  
    "title" => "The Art of Computer Programming"  
];  
extract($book); // creates two new variables: $author and $title  
  
echo $author;    // output "Knuth"  
echo $title;    // output "The Art of Computer Programming"
```

```
$lake = "Michigan";  
$area = 22300;  
$mich = compact('lake', 'area');  
// creates a new array [ "lake" => "Michigan", "area" => 22300]
```

15

Local vs. Global Variables

```
$total = 100;           // global var  
  
function doItLocal() {  
    var_dump ($total + 3); // output int(3), $total is local (undef)  
}  
  
function doItGlobal() {  
    var_dump ($GLOBALS['total'] + 5); // output int(105)  
  
    global $total;  
    var_dump ($total + 11);           // output int(111)  
}
```

16

Cloud9 Demo: <http://c9.io>

17

Cloud9 VM Setup

1. Create a new workspace using PHP & Apache template
2. Activate the server
3. Confirm PHP script runs correctly
4. Activate MySQL
5. Download Sample Database
6. Access Database from PHP script

18

PHP + MySQL Improved

19

PHP MySQL Database Connections

Assumptions:

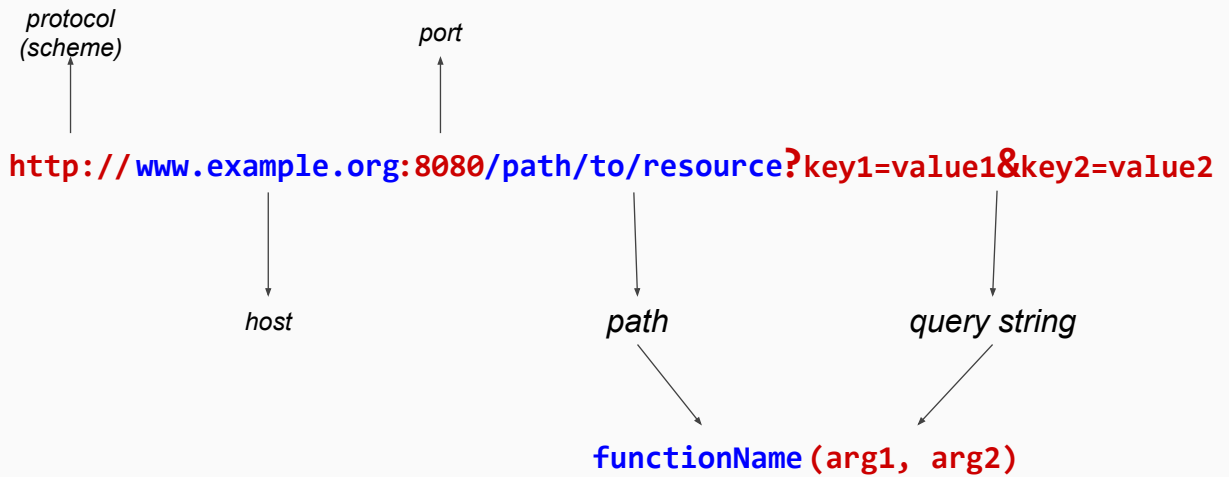
- Database: **personal**
- Table: **city** with two columns: **name** and **zipcode**

```
<?php
$db = new mysqli('localhost', 'youruser', 'yourpassword', 'personal');
if ($db->connect_error) {
    die("Can't connect to db" . $db->connect_error);
} else {
    // Or: $db->select_db("personal");

    $result = $db->query("SELECT name, zipcode FROM city");
    while ($row = $result->fetch_assoc()) {
        printf ("%5d %s\n", $row['zipcode'], $row['name']);
    }
}
?>
```

20

URL Structure



21

URL Encoding

```
showStockGraph ("AMZN", "Amazon Stock");
```

```
http://my.host.org/finance/showStockgraph.php?symbol=AMZN&title=Amazon%20Stock
```

Plain String	URL Encoded
menu: "PB&J"	?menu=PB%26J
filter: "A==500"	?filter=A%3D%3D500
startdate: "Jun 6", enddate: "Sep 22"	?startdate=Jun%206&enddate=Sep%2022
startdate: "Jun 6", enddate: "Sep 22"	?startdate=Jun+6&enddate=Sep+22

PHP functions:

```
$what = urlencode("PB&J");           // "PB%26J"  
$when = urldecode("Jun%206");        // "Jun 6"
```

22

explode() and urldecode()

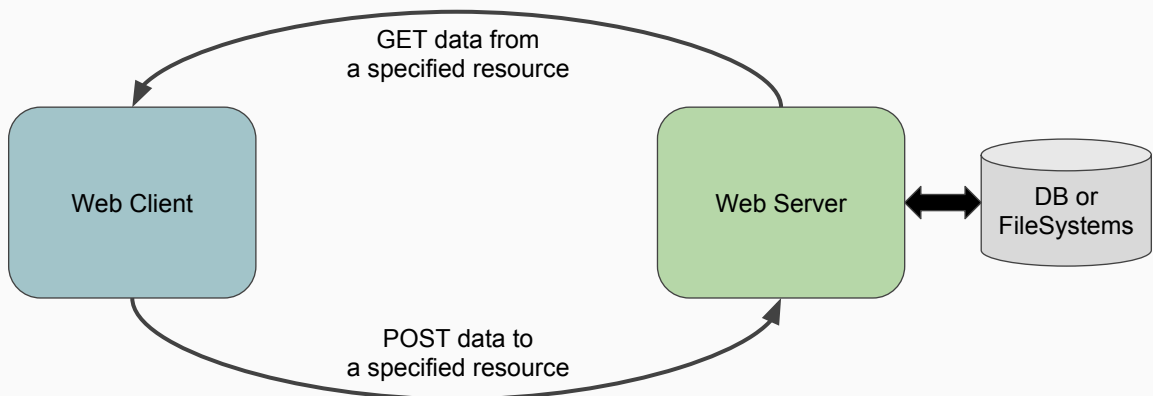
```
<?php
                                // "Stock Price in 2004"
    $samplequery = "stock=AMZN&title=Stock%20Price%20in%202004";

    foreach (explode('&', $samplequery) as $pair) {
        $param = explode('=', $pair);

        print_r ($param[0]);
        print_r (urldecode($param[1]));
    }
?>
```

23

HTTP Methods: GET and POST

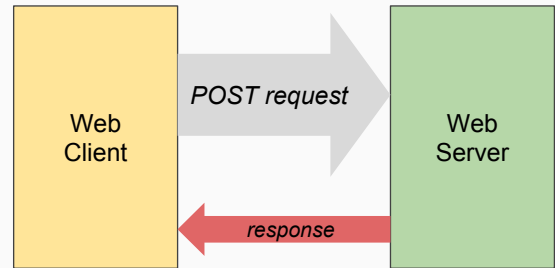
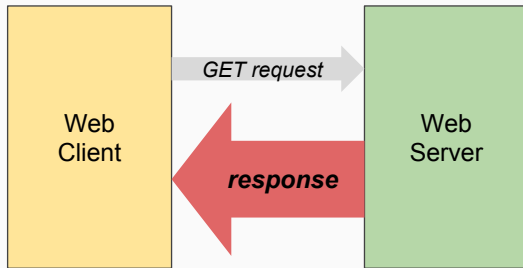


24

GET

vs.

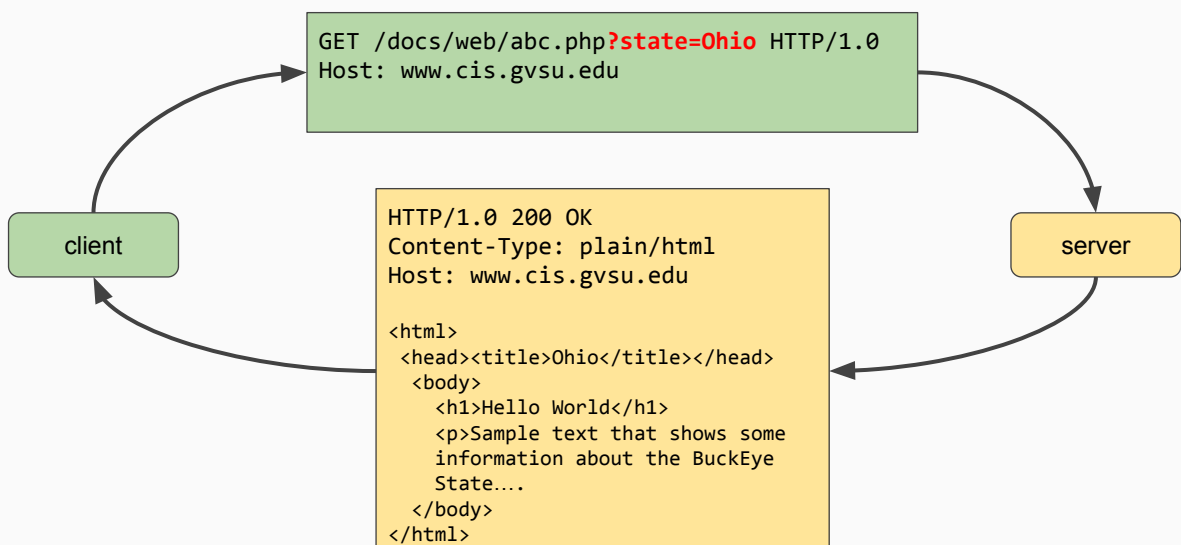
POST



25

HTTP Messages (GET)

Recall from HTTP lecture



26

HTTP Messages (POST)

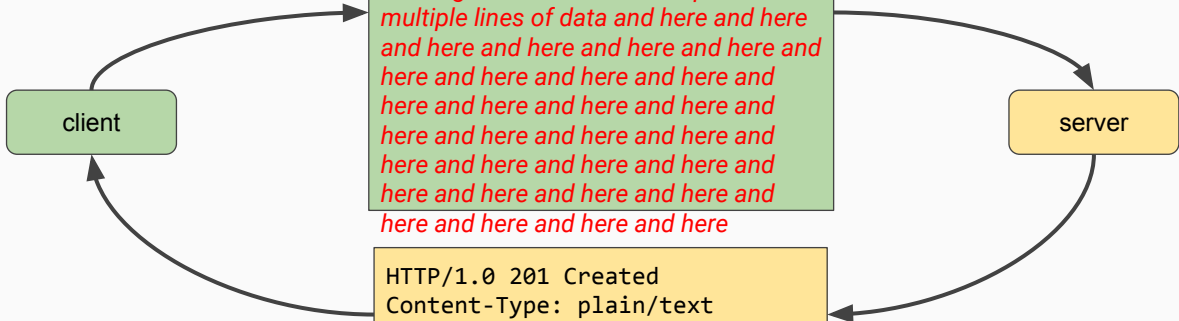
Content-Type: informs the server how to parse the payload

```
POST /docs/web/abc.php HTTP/1.0
Host: www.cis.gvsu.edu
Content-Type: _____
```

[illegible]

```
HTTP/1.0 201 Created
Content-Type: plain/text
Host: www.cis.gvsu.edu
```

Recall from HTTP lecture



27

GET

VS.

POST

- Details of request are encoded into the URL **query string**
- Should **not** be used for requests that alter the data on the server
- *Idempotent*: the response of a request **remains the same** regardless the number of times the URL is called
 - $X + 0 = X$ (add by zero in an idempotent operation)
- Details of the request (data payload) are encoded in the **message body**
- Should be used for requests that alter the data on the server
- Non-idempotent operations

28

PHP \$_SERVER (associative) array

```
GET /path/to/abc.php?state=ohio HTTP/1.0
Host: www.cis.gvsu.edu
```

\$_SERVER['SERVER_NAME']	www.cis.gvsu.edu
\$_SERVER['SERVER_PROTOCOL']	HTTP 1.0
\$_SERVER['SERVER_PORT']	80
\$_SERVER['REQUEST_METHOD']	GET
\$_SERVER['PATH_INFO']	/path/to
\$_SERVER['SCRIPT_NAME']	/path/to/abc.php
\$_SERVER['QUERY_STRING']	state=ohio

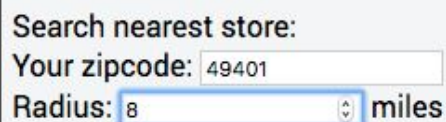
To see more entries, use `print_r ($_SERVER);` in your PHP script

29

HTML Forms

```
<form>
  Search nearest store:<br/>
  <label for="zip">Your zipcode:</label>
  <input type="text" name="zip"><br/>

  Radius: <input type="number" name="radius" /> miles<br/>
</form>
```



Search nearest store:
Your zipcode: 49401
Radius: 8 miles

30

```
<input type="____" />
```

<code><input type="button" /></code>	<code><input type="color" /></code>	<code><input type="date" /></code>
<code><input type="email" /></code>	<code><input type="file" /></code>	<code><input type="month" /></code>
<code><input type="number" /></code>	<code><input type="password" /></code>	<code><input type="radio" /></code>
<code><input type="range" /></code>	<code><input type="date" /></code>	<code><input type="time" /></code>
<code><input type="week" /></code>		

See them online ([JSFiddle](#))

31

HTTP GET

32

Form Actions (GET)

http://where.example.org/page.html

```
<!-- in page.html -->
<html>
<body>
  <h2>Search Restaurant?</h2>
  <form action="abc.php" method="GET">
    Zipcode: <input type="text" name="zip" />
    <input type="submit" value="Go"/>
  </form>
</body>
</html>
```

```
<!-- in abc.php -->
<?php
$uzip = $_GET['zip'];
$querystr = "SELECT * FROM restaurant WHERE zip = '$uzip'";
$db->query ($querystr);
// more code here
?>
```

Search Restaurant?

Zipcode:

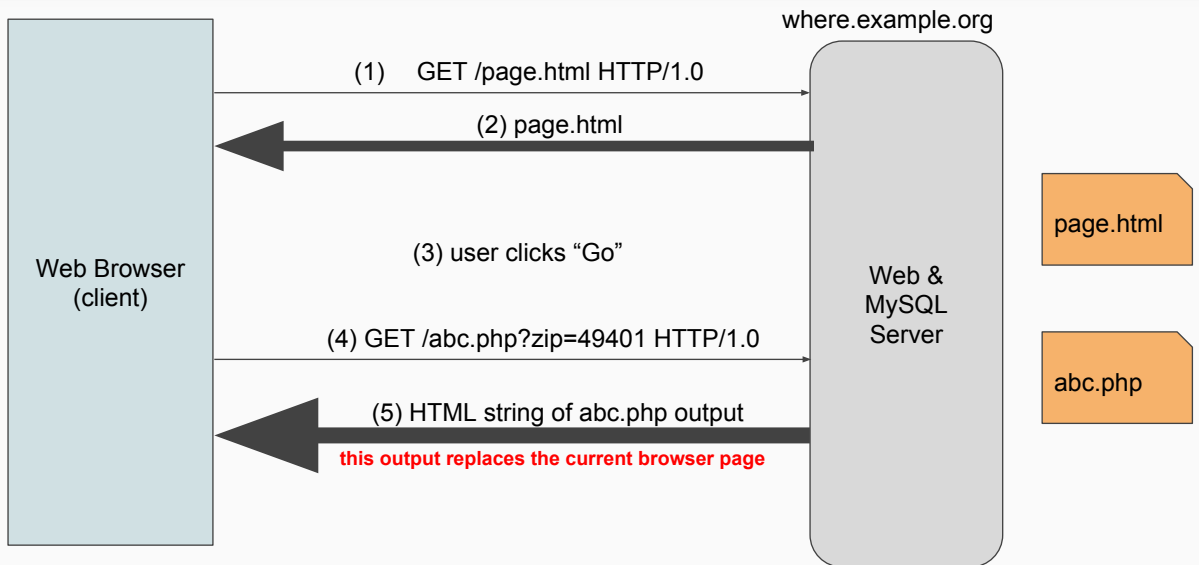
Go

33

Clould9 Demo: HTML Form

34

Request/Response Flow



35

HTTP POST

36

Form Actions (POST)

http://where.example.org/feedback.html

```
<!-- in page.html -->
<html>
<body>
  <h2>Your Dining Experience</h2>
  <form action="abc.php" method="POST">
    Date of Visit <input type="date" name="date" />
    Rate your visit
    <input type="number" name="rate" min="1" max="4" />
    <input type="submit" value="Submit" name="submit"/>
  </form>
</body>
</html>
```

```
<!-- in abc.php -->
<?php
$when = $_POST['date']; $rate = $_POST['rate'];
$querystr = "INSERT INTO review (date, rate) VALUES('$when', $rate)";
$db->query ($querystr);
// more code here
?>
```

Your Dining Experience

Visit Date:

Rate your visit:

Submit

37

File Upload and \$_FILES

```
<!-- in upload.html -->
<form enctype="multipart/form-data" action="upload.php" method="POST">
  File name: <input name="upload" type="file" /><br/>
  <input type="submit" name="submit" value="Upload"/>
</form>
```

```
<!-- in upload.php -->
<?php
$upInfo = $_FILES["upload"];
if ($upInfo["error"] == 0) {
  echo "Location on server " . $upInfo["tmp_name"];
  $destination = "./uploads/" . $upInfo["name"];
  if (is_uploaded_file($upInfo["tmp_name"])) {
    move_uploaded_file ($upInfo["tmp_name"], $destination);
  }
}
?>
```

38

<form enctype="____">

- **application/x-www-form-urlencoded**: spaces are converted to "+", other special characters are converted to ASCII HEX
- **multipart/form-data**: No character encoding. **Required for file upload!**
- **text/plain**: spaces are converted to "+", other special characters are not encoded

39

HTTP Post Request of File Upload

```
<form enctype="multipart/form-data"
  action="upload.php"
  method="POST">
File name:
<input name="ufile"
  type="file" /><br/>
<input type="text"
  name="compRatio" />
<input type="submit"
  name="submit"
  value="Upload"/>
</form>
```

Assume user input

- File name is **a.txt**
- Compression ration is **0.352**

HTTP POST request sent to your server

```
POST /path/to/your/upload.php HTTP/1.1
Content-Length: _____
Content-Type: multipart/form-data; boundary=CS371GVSU

--CS371GVSU
Content-Disposition: form-data; name="ufile"; filename="a.txt"
Content-Type: text/plain
```

Content of the uploaded file (a.txt) goes here and here and continue to the next line and over and more.

```
--CS371GVSU
Content-Disposition: form-data; name="compRatio"

0.352
--CS371GVSU--
```

40

Form Input Validation

- Validate individual input fields
 - `<input type="....." required />`
 - `<input type="....." pattern="regex" />` `<!-- only in HTML5 -->`
- OR validate the entire form
 - `<form action="...." onSubmit="return verifyForm()" method="POST">`
`<script>`
 `function verifyForm() {`
 `// validation logic here`
 `return false; // return true when all inputs are validated`
 `}`
`</script>`

41

/RegEx/: Regular Expressions

- An expression that represents a **set** of strings
 - Operands: character
 - Operators
 - * zero or more occurrence of string
 - + one or more occurrence of string
 - ? zero or one occurrence of string
 - [: a range of characters
 - . : any single character
 - \ escape character: assign a special meaning to the character after \

42

RegEx Examples

RegEx	Description	Matched Strings
Hi	'H' followed by 'i'	"Hi"
Hi*	'H' followed by zero or more 'i'	"H", "Hi", "Hii", "Hiiiiiiiiiii", ...
H*i	Zero or more 'H' followed by 'i'	"i", "Hi", "HHHHHHHi", ...
Hi+	'H' followed by one or more 'i'	"Hi", "Hii", "Hiiiiiiiiiii", ...
(Hi)+	One or more occurrence of "Hi"	"Hi", "HiHi", "HiHiHiHiHiHi", ...
H[io]	'H' followed by 'i' or 'o'	"Hi", "Ho"
H[io]+	'H' followed by one or more 'i' or 'o'	"Hi", "Ho", "Hioioiooo", "Hiii", "Ho"
H[a-d]	'H' followed 'a' or 'b' or 'c' or 'd'	"Ha", "Hb", "Hc", "Hd"
CS\d{3}	"CS" followed by exactly 3 digits	"CS111", "CS539", "CS371"

Complete References: [JavaScript Regular Expressions](#) and [Regular Expression Tester](#)

43

Accessing Forms in JavaScript functions

- `document.forms` is a global associative array that holds all the forms in a particular page
- Use `document.forms['form-name']` to access a specific form
- Use `document.forms['form-name']['input-name']` to access a specific input field

44

```
<form name="signup" onsubmit="return verifyme()">  
  <!-- other fields not shown -->  
  <input type="password" name="pwd" ... />  
  <input type="submit" ... />  
</form>
```

```
<script>  
function verifyme() {  
  var pass = document.forms['signup']['pwd'];  
  return pass.value.length >= 12;  
}  
</script>
```

PHP Filesystem Functions

- chgrp(), chown(), copy(), delete()
- file_get_contents(): reads a file into a (long) string
- file_put_contents(): writes a (long) string to a file
- f[get|put]csv(): reads/writes a a single row from|to a CSV file
- is_dir(), is_executable(), is_readable(), ...
- [More details](#)

HTTP Messages Delivered to Server

http://where.example.org/page.html

```
<!-- in page.html -->
<html>
<body>
  <h2>Search Restaurant?</h2>
  <form action="abc.php" method="GET">
    <input type="text" name="zip" />
    <input type="number" name="distance" />
    <input type="submit" value="Go"/>
  </form>
</body>
</html>
```

```
GET /abc.php?zip=49401&distance=10 HTTP/1.1
Host: where.example.org
```

```
<!-- in page.html -->
<html>
<body>
  <h2>Search Restaurant?</h2>
  <form action="abc.php" method="POST">
    <input type="text" name="zip" />
    <input type="number" name="distance" />
    <input type="submit" value="Go"/>
  </form>
</body>
</html>
```

```
POST /abc.php HTTP/1.1
Host: where.example.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

zip=49101&distance=10
```

Assumption: the user enters 49401 (for zip) and 10 miles (for distance)

47

MySQL in EOS

- Hostname: cis.gvsu.edu
- User: YourUserId
- Password: YourUserIdXXXX
 - XXXX: last four digits of your G-number
- Database Name: YourUserId

48


```
# Connect from terminal
```

```
eos22:> mysql -p -h cis.gvsu.edu  
Enter password: your-password-here
```

```
(mysql shell) > use dbName;      # your dbName is your userid  
(mysql shell) > select * from tableName;
```

```
<!-- connect from a PHP script -->  
<?php  
$db = new mysqli('cis.gvsu.edu', 'userid', 'password', 'dbName');  
print_r ($db);  
$queryStr = "select * from tableName";  
$db->query ($queryStr);  
// more code here ...  
?>
```

49

Basic MySQL Queries

Assumption: a table of (chemical) atom with three columns: symbol, name, (atomic) weight

```
INSERT INTO atom VALUES("C", "Carbon", 12);
```

```
INSERT INTO atom (symbol, name, weight) VALUES ("O", "Oxy", 16);
```

```
DELETE FROM atom WHERE weight > 30;
```

```
UPDATE atom SET name = "Oxygen" WHERE symbol = "O";
```

```
SELECT name WHERE weight >= 30 AND weight <= 47;
```

[Additional References](#)

50