

# Introduction to Server-Side Programming



Charles Liu

# Overview



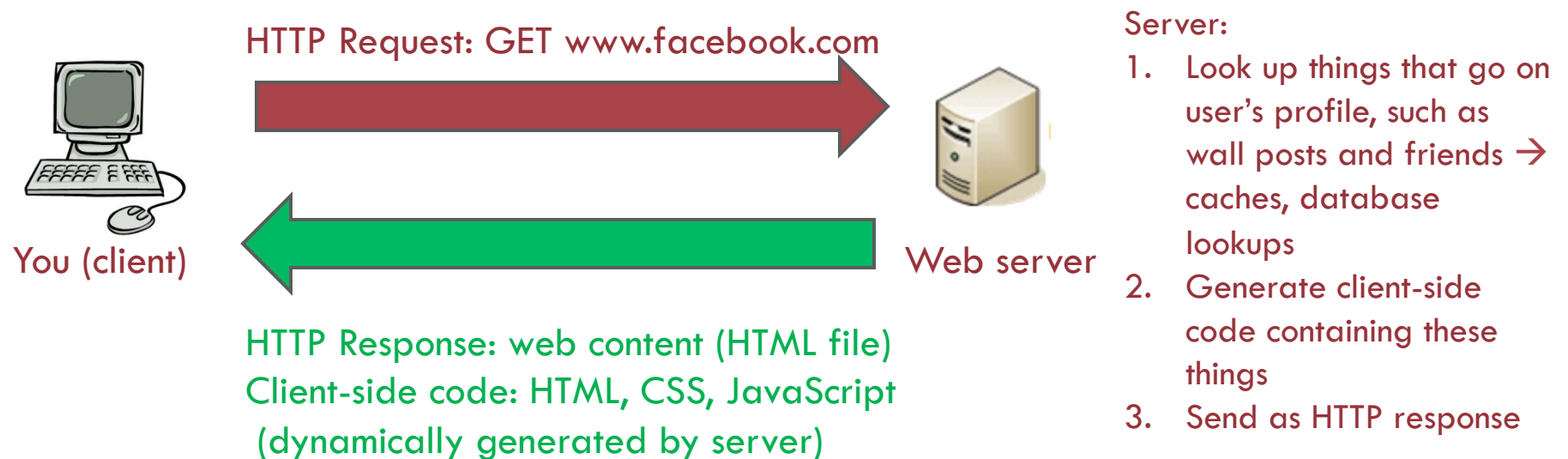
1. Basics of HTTP
2. PHP syntax
3. Server-side programming
4. Connecting to MySQL

# Request to a Static Site

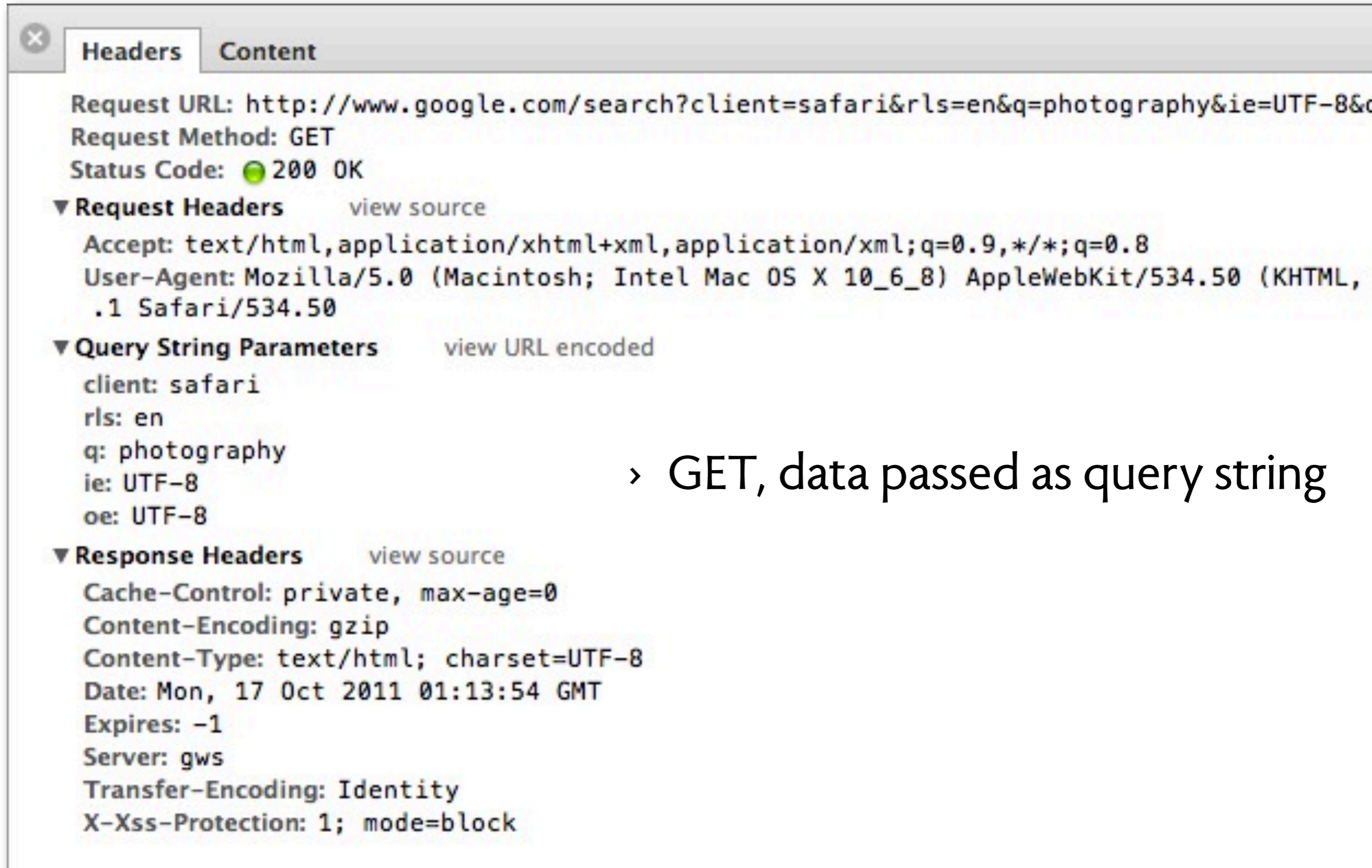


# Request to a Dynamic Site

- The server must respond dynamically if it needs to provide different client-side code depending on the situation
  - ▣ Date and time
  - ▣ Specifics of the user's request
  - ▣ Database contents – forms and authentication



# sample http interactions



The screenshot shows the 'Headers' tab of a web browser's developer tools. The request is a GET method to a Google search page. The status is 200 OK. The request headers include Accept, User-Agent, and Query String Parameters. The response headers include Cache-Control, Content-Encoding, Content-Type, Date, Expires, Server, Transfer-Encoding, and X-Xss-Protection.

```
Request URL: http://www.google.com/search?client=safari&rls=en&q=photography&ie=UTF-8&oe=UTF-8
Request Method: GET
Status Code: 200 OK

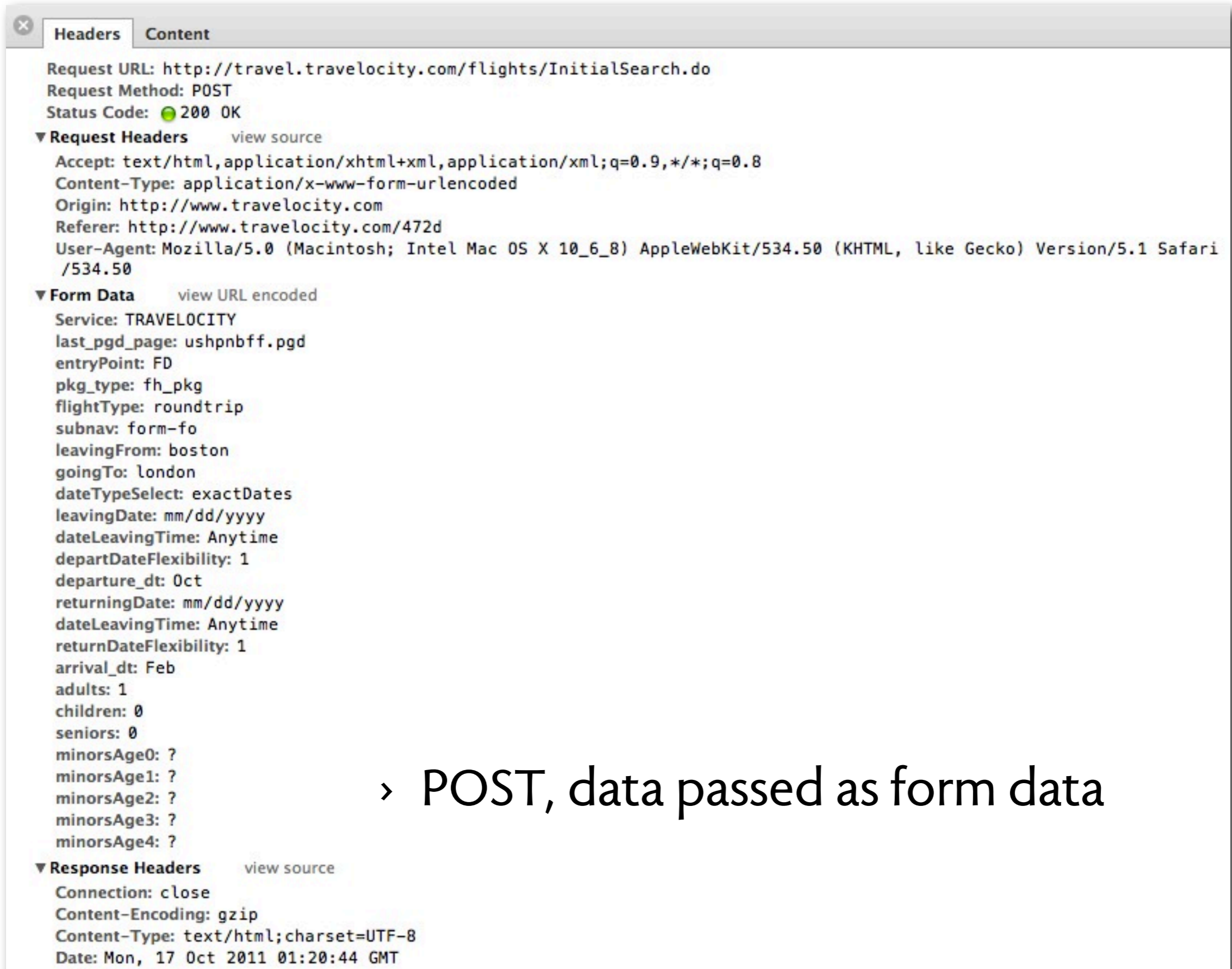
▼ Request Headers view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.50 (KHTML, like Gecko) Safari/534.50

▼ Query String Parameters view URL encoded
client: safari
rls: en
q: photography
ie: UTF-8
oe: UTF-8

▼ Response Headers view source
Cache-Control: private, max-age=0
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Mon, 17 Oct 2011 01:13:54 GMT
Expires: -1
Server: gws
Transfer-Encoding: Identity
X-Xss-Protection: 1; mode=block
```

› GET, data passed as query string

# sample http interactions



The screenshot shows a web browser's developer tools window with the 'Headers' tab selected. It displays the details of an HTTP POST request and its corresponding response.

**Request Headers**

- Request URL: `http://travel.travelocity.com/flights/InitialSearch.do`
- Request Method: `POST`
- Status Code: `200 OK`
- Accept: `text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`
- Content-Type: `application/x-www-form-urlencoded`
- Origin: `http://www.travelocity.com`
- Referer: `http://www.travelocity.com/472d`
- User-Agent: `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.50 (KHTML, like Gecko) Version/5.1 Safari/534.50`

**Form Data**

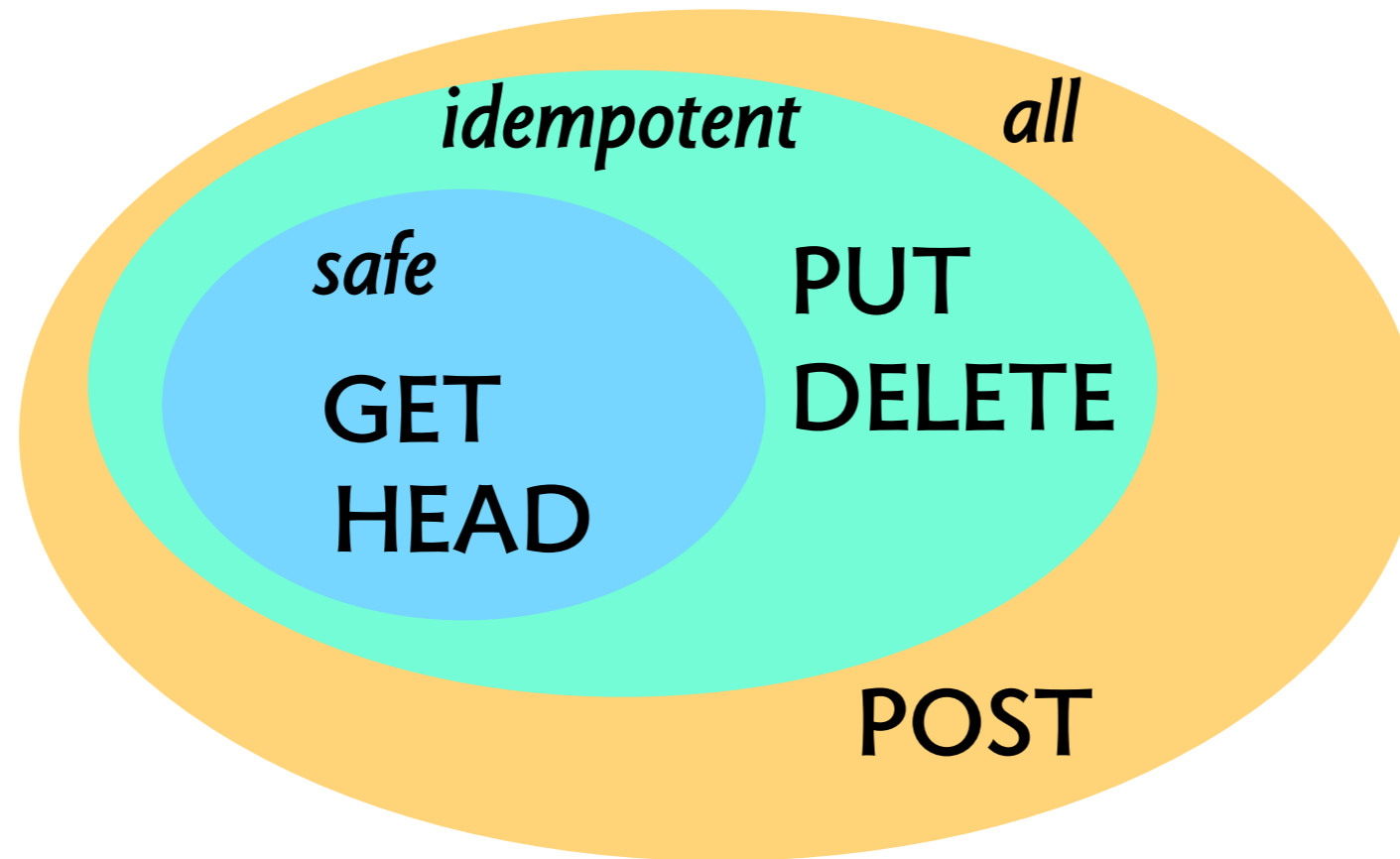
- Service: `TRAVEL0CITY`
- last\_pgd\_page: `ushpnbff.pgd`
- entryPoint: `FD`
- pkg\_type: `fh_pkg`
- flightType: `roundtrip`
- subnav: `form-fo`
- leavingFrom: `boston`
- goingTo: `london`
- dateTypeSelect: `exactDates`
- leavingDate: `mm/dd/yyyy`
- dateLeavingTime: `Anytime`
- departDateFlexibility: `1`
- departure\_dt: `Oct`
- returningDate: `mm/dd/yyyy`
- dateLeavingTime: `Anytime`
- returnDateFlexibility: `1`
- arrival\_dt: `Feb`
- adults: `1`
- children: `0`
- seniors: `0`
- minorsAge0: `?`
- minorsAge1: `?`
- minorsAge2: `?`
- minorsAge3: `?`
- minorsAge4: `?`

**Response Headers**

- Connection: `close`
- Content-Encoding: `gzip`
- Content-Type: `text/html; charset=UTF-8`
- Date: `Mon, 17 Oct 2011 01:20:44 GMT`

› POST, data passed as form data

# http methods



- › safe: no side effects
- › idempotent: doing twice same as once
- › PUT vs POST: PUT typically *names* a created object
- › PUT & DELETE used in APIs but not usually in browsers

# response status codes

## categories of codes

- › 1xx informational
- › 2xx success
- › 3xx redirect
- › 4xx client error
- › 5xx server error

## most common codes

- › 200 OK (request succeeded, resource is in message body)
- › 404 Not Found (resource doesn't exist)
- › 303 See Other (resource moved, see location header)
- › 500 Server Error (web app implementer messed up)



# Server-side options

---

- PHP – today
  - ▣ Easy to start, lower learning curve
  - ▣ Potentially messy as your site grows
- Javascript frameworks – node.js and Meteor.js
- Ruby on Rails
- Other options – Django, Flask, Sinatra...

# PHP

## Introduction and Basic Syntax



Charles Liu

# What is PHP?



- PHP = PHP: Hypertext Preprocessor
- Server-side scripting language that may be embedded into HTML
- Ultimate goal is to get PHP files to **generate** client-side code
  - ▣ must end up with HTML, CSS, JavaScript, other client-side code!

# Side-by-side

## PHP File:

```
<html>
<head>
<title> PHP Introduction </title>
</head>
<body>
This is HTML! <br />
<?php
    echo 'This is PHP! <br />';
?>
</body>
</html>
```

## Output: resulting HTML

```
<html>
<head>
<title> PHP Introduction </title>
</head>
<body>
This is HTML! <br />
This is PHP! <br />
</body>
</html>
```

# A closer look

```
<html>
<head>
    <title> PHP Introduction </title>
</head>
<body>
This is HTML! <br />
<?php
    echo 'This is PHP! <br />'; // prints to screen
    /*
    Here's a longer
    comment
    that spans multiple
    lines.
    */
?>
</body>
</html>
```

- PHP tags: <?php and ?>
- The echo command
- Single line comment ( // )
- Multiple line comment ( /\* and \*/)

# Viewing PHP files

---

- PHP files executed on the web server
- Save .php files in subdirectory of web server
  - ▣ /var/www/ on many Linux configurations
  - ▣ web\_scripts directory of your user directory on Athena
- Make call to web server via domain name (google.com), IP address (72.26.203.99), or localhost if on your own computer

# PHP

## Syntax: Variables, Operators, and Strings



Charles Liu

# Variables

- Store values for future reference, use variable name to refer to the value stored in it

```
$x = 42;          // store the value 42 in $x
echo $x;         // prints 42
echo $x+1;       // prints 43, value of $x is still 42
$x = 'hello!';   // type of $x can change
```

- PHP is a loosely-typed language
  - ▣ Do not need to declare the type of a variable
  - ▣ Type can change throughout the program



# Operators

---

- Arithmetic operators
  - ▣ `+`, `-`, `*`, `/`, `%` (modulus – remainder after division)
- Logical AND (`&&`), OR (`||`), NOT (`!`)
- Assignment operators (`=`)
- Shorthand for assignment operators:
  - ▣ `$x += $y` equivalent to `$x = $x + $y`
  - ▣ Also works with subtraction, multiplication, division, modulus, and string concatenation

# == versus ===

- Two “equality” operators
  - ▣ == tests for “equality” in value but not necessarily type
  - ▣ === tests for “identity” in value AND type
- == ignores the distinction between:
  - ▣ Integers, floating point numbers, and strings containing the same numerical value
  - ▣ Nonzero numbers and boolean TRUE
  - ▣ Zero and boolean FALSE
  - ▣ Empty string, the string ‘0’ and boolean FALSE
  - ▣ Any other non-empty string and boolean TRUE

# Strings

- Concatenation of strings – the . operator

```
$a = 'hello';  
$b = 'world';  
echo $a . ' ' . $b . '!'; // prints 'hello world!'
```

- String functions

- ▣ Length: strlen()
- ▣ Position of substring: strpos()
- ▣ More on string functions:

[http://www.w3schools.com/php/php\\_ref\\_string.asp](http://www.w3schools.com/php/php_ref_string.asp)

# PHP

## Syntax: Conditional and Looping Statements



Charles Liu

# Conditional Statements

---

```
if (condition / boolean expression) {
    statements
}
else if (another condition) {
    statements
}
// there may be more than one else if block
else {
    statements
}
```

```
$x = 5;
if ($x == 5) {
    echo 'The variable x has value 5!';
}
```

# Loops

```
$x = 2;
while ($x < 1000) {
    echo $x . "\n"; // \n is newline character
    $x = $x * $x;
}
```

```
do {
    echo $x . "\n";
    $x = $x * $x;
} while ($x < 1000); // note the semicolon
```

```
for ($i = 1; $i <= 10; $i++) {
    echo $i . ":" . ($i * $i) . "\n";
}
```

# PHP

## Syntax: Functions and Global Variables



Charles Liu

# Defining your own functions

```
function function_name ($arg1, $arg2) {  
    function code      function parameters  
    return $var // optional  
}
```

## Example: a simple multiply function

```
function multiply($x, $y) {  
    echo $x * $y;  
    echo "\n";  
}  
multiply(5, 1.2); → prints 6  
$a = 5;  
$b = 1.2;  
multiply($a, $b); → prints 6  
$a = array(1,2,3);  
multiply($a, $b); → error  
$a = "string"  
multiply($a, $b); → prints 0 (!?)
```



# Return values

- A function can return a value after it is done
  - ▣ Use this value in future computation, use like a variable, assign value to a variable

```
function multiply($x, $y) {  
    return $x * $y;  
}
```

`multiply(2,3);` → prints nothing! returns value, but we don't store anywhere

`echo multiply(2,3);` → prints 6

`$a = multiply(2,3);` → assigns the value 6 to the variable \$a

`$b = multiply(multiply(2,3), multiply(3,4));` → assigns the value 72 to the variable \$b

# Variable scope

- Variables declared within a function have *local scope*
  - ▣ Can only be accessed from within the function

```
<?php
function function1() {
    ... // some code
    $local_var = 5;           // this variable is LOCAL to
                              // function1()
    echo $local_var + 3;     // prints 8
}

... // some code
function1();
echo $local_var;           // does nothing, since $local_var is
                           // out of scope

?>
```

# Global variable scope

- Variables declared outside a function have global scope

- ▣ Use global keyword to gain access within functions

```
<?php
function function1() {
    echo $a;          // does nothing, $a is out of scope
    global $a;       // gain access to $a within function
    echo $a;         // prints 4
}

... // some code
$a = 4;             // $a is a global variable
function1();

?>
```

# PHP

## Syntax: Arrays



Charles Liu

# Arrays as a list of elements

- Use arrays to keep track of a list of elements using the same variable name, identifying each element by its *index*, starting with 0

```
$colors = array('red', 'blue', 'green', 'black', 'yellow');
```

- To add an element to the array:

```
$colors[] = 'purple';
```

- To remove an element from the array:

```
unset($colors[2]);
```

```
$colors = array_values($colors);
```

# Arrays as key-value mappings

- Use arrays to keep track of a set of unique *keys* and the *values* that they map to – called an *associative array*

```
$favorite_colors = array('Joe' => 'blue', 'Elena' =>
'green',
'Mark' => 'brown', 'Adrian' => 'black', 'Charles' =>
'red');
```

- **To add an element to the array:**

```
$favorite_colors['Bob'] = 'purple';
```

- **To remove an element from the array:**

```
unset($favorite_colors['Charles']);
```

- **Keys must be unique:**

```
$favorite_colors['Joe'] = 'purple' overwrites 'blue'
```

# The for-each loop

---

- The for-each loops allow for easy iteration over all elements of an array.

```
foreach ($colors as $color) {  
    echo $color; // simply prints each color  
}  
foreach ($colors as $number => color) {  
    echo "$number => $color"; // prints color with index  
    // to change an element:  
    // $colors[$number] = $new_color;  
}
```

# PHP

## HTTP Requests and Forms



Charles Liu



# Superglobals

---

- A few special associative arrays that can be accessed from anywhere in a PHP file
- The `$_SERVER` superglobal gives information about server and client
  - `$_SERVER['SERVER_ADDR']` → server IP
  - `$_SERVER['REMOTE_ADDR']` → client IP
  - `$_SERVER['HTTP_USER_AGENT']` → client OS and browser

# Passing information to the server

- Sometimes, we require additional values be passed from client to server
  - ▣ Login: username and password
  - ▣ Form information to be stored on server
- GET request: pass information via the URL
  - ▣ <http://www.yourdomain.com/yourpage.php?firstparam=firstvalue&secondparam=secondvalue>
  - ▣ Access values server-side using `$_GET` superglobal
    - `$_GET['firstparam'] => 'firstvalue'`
    - `$_GET['secondparam'] => 'secondvalue'`

# When to use \$\_GET vs. \$\_POST

---

- GET requests are sent via the URL, and can thus be cached, bookmarked, shared, etc
- GET requests are limited by the length of the URL
- POST requests are not exposed in the URL and should be used for sensitive data
- There is no limit to the amount of information passed via POST

# Dealing with forms

---

- Forms are generally used to collect data, whether the data needs to be stored on the server (registration) or checked against the server (login)
- 2 components to a form:
  - ▣ The HTML generating the form itself
  - ▣ The server-side script that the form data is sent to (via GET or POST), taking care of the processing involved
    - Server should respond appropriately, redirecting the user to the appropriate destination or generating the appropriate page

# Forms: client-side

```
<html>
  <head>
    <title> A Form Example </title>
  </head><body>
<form action="welcome.php" method="post">
Name: <br /> <input type="text" name="name" /><br />
Phone Number: <br /> <input type="text" name="phone" /><br /
>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

- form action – where to send the form data
- method – how to send the data (GET or POST)
- Name attributes become the keys used to access the corresponding fields in the `$_GET` or `$_POST` arrays

# Forms: server-side

```
<html>
<head><title>This is welcome.php</title></head>
<body>
The name that was submitted was: &nbsp;
<?php echo $_POST['name']; ?><br />
The phone number that was submitted was: &nbsp;
<?php echo $_POST['phone']; ?><br />
</body>
</html>
```

- A simple PHP file that displays what was entered into the form
  - Can do many other things server-side depending on the situation

# PHP

## Cookies and Sessions



Charles Liu

# Cookies and sessions

---

- HTTP is stateless – it does not keep track of the client between requests
- But sometimes we need to keep track of this information
  - Shopping cart
  - “Remember me” on login sites
- 2 solutions to this issue
  - Cookies – small file stored client-side
  - Sessions – relevant data stored on the server



# cookies in http

## cookie is

- › name-value pair
- › expiration, path & domain

## server sends

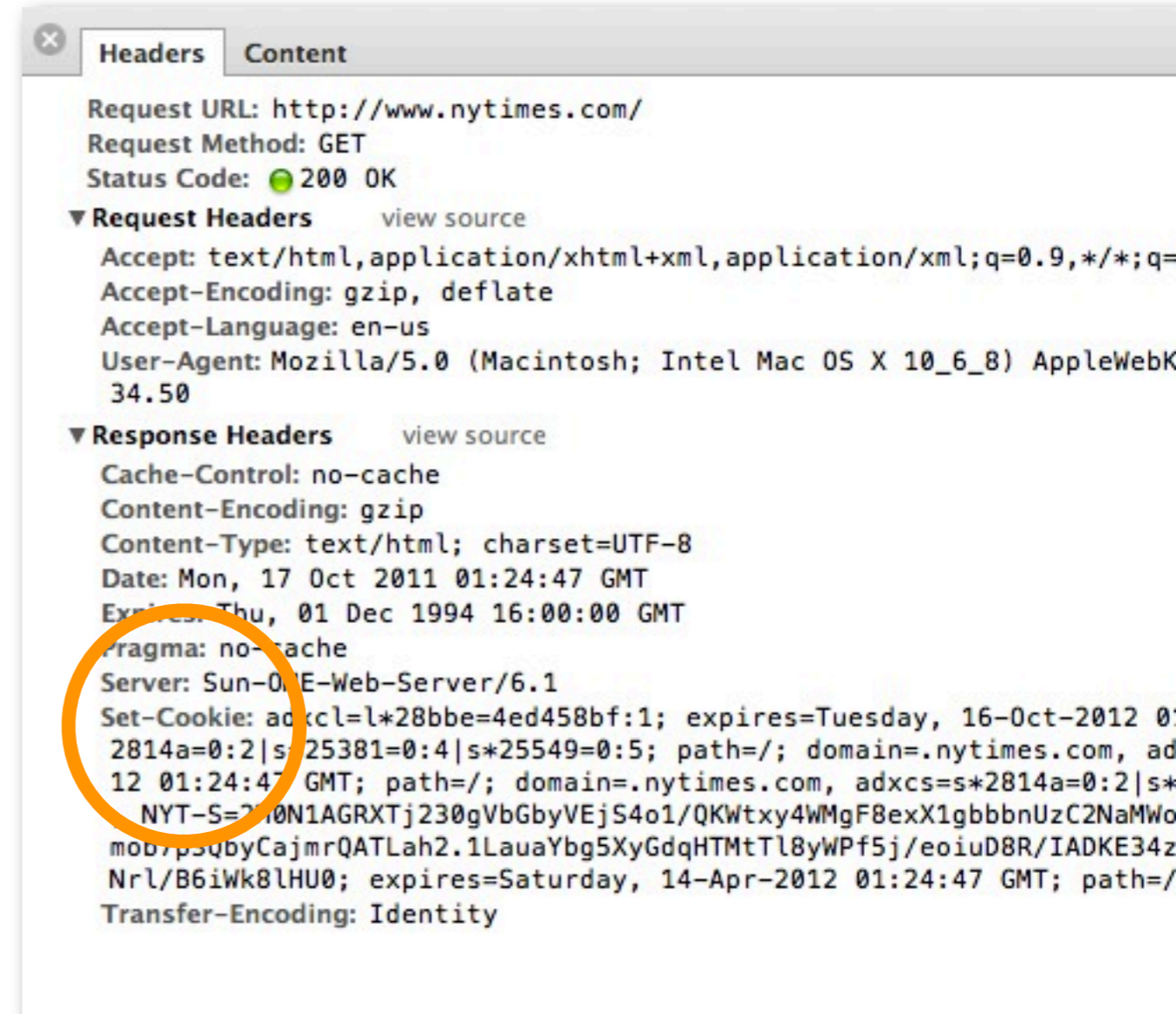
- › using set-cookie header

## browser sends back

- › all unexpired cookies
- › with matching path

## expiration

- › session cookies: on quit
- › persistent cookies: on expire



```
Headers Content
Request URL: http://www.nytimes.com/
Request Method: GET
Status Code: 200 OK
Request Headers
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.50.2
Response Headers
Cache-Control: no-cache
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Mon, 17 Oct 2011 01:24:47 GMT
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Pragma: no-cache
Server: Sun-OS-Web-Server/6.1
Set-Cookie: adxcl=l*28bbe=4ed458bf:1; expires=Tuesday, 16-Oct-2012 01:24:47 GMT; path=/; domain=.nytimes.com, adxcs=s*2814a=0:2|s*25381=0:4|s*25549=0:5; path=/; domain=.nytimes.com, NYT-S=710N1AGRXTj230gVbGbyVEjS4o1/QKWtxy4WMgF8exX1gbbbnUzC2NaMwo mob7psqbyCajmrQATLah2.1LauaYbg5XyGdqHTMtTl8yWPf5j/eoiuD8R/IADKE34zNrl/B6iWk8lHU0; expires=Saturday, 14-Apr-2012 01:24:47 GMT; path=/
Transfer-Encoding: Identity
```

## *a funny cookie tale*

nytimes.com used cookies to count #articles read, so viewers just deleted cookies...

# Cookies

---

- Cookies are stored on the user's browser, and are sent to the server on every relevant request
- The `$_COOKIE` superglobal makes a cookie a key-value pairing
  - ▣ Store user information as a value with a known key
  - ▣ Never assume a cookie has been set. Always check with `isset($_COOKIE[$cookie_name])` before trying to use the cookie's value

# The setcookie() function

- To set a cookie in PHP:

```
setcookie(name, value, expire, path, domain);
```

- Name and value correspond to `$_COOKIE[$name] = $value`
- Expiration – cookie will no longer be read after the expiration
  - Useful to use time in seconds relative to the present:
    - `time() + time in seconds until expiration`
- Path and domain refer to where on the site the cookie is valid
  - Usually `‘/’` for path and the top-level domain (`yoursitename.com`)
- To delete a cookie, set a new cookie with same arguments but expiration in the past

# Setting cookies

- Cookies are set via the HTTP header
  - ▣ Must be sent before the body – before any HTML, CSS, JS, etc.
- This code will not work:

```
if(isset($_COOKIE["6470"])) {  
    $value = $_COOKIE['6470'];  
    echo "Cookie is set to $value";  
}  
else {  
    $value = 0;  
}  
// after echo statement: will not work!  
setcookie("6470", $value+1, time()+60*60);?>
```

# Sessions

---

- Two main disadvantages of cookies
  - ▣ Limited in size by browser
  - ▣ Stored client-side → users / malicious people can change
- Sessions store user data on the server
  - ▣ Limited only by server space
  - ▣ Cannot be modified by users
- A potential downside to sessions is that they expire when the browser is closed
- Sessions are identified by a session id: often a small cookie! But the rest of the data is still stored on the server

# Using sessions

- Call `session_start()` at top of **every** page to start session
  - ▣ Sets a cookie on the client: must follow same rules as cookies (before any HTML, CSS, JS, echo or print statements)
- Access data using the `$_SESSION` superglobal

```
<?php
session_start();
if (isset($_SESSION["count"])) {
    $_SESSION["count"] += 1;
    echo "You\'ve visited here {$_SESSION['count']}
    times";
}
else {
    $_SESSION["count"] = 1;
    echo "You\'ve visited once";
}
?>
```

# Removing sessions

---

- Remove an individual element of the `$_SESSION` superglobal
  - ▣ `unset($_SESSION['key_name']);`
- Destroy the entire session, remove all data
  - ▣ Use the function `session_destroy()`
  - ▣ `$_SESSION` no longer valid
  - ▣ Will need to call `session_start()` to start a new session

# Recap: a comparison

	COOKIES	SESSIONS
Where is data stored?	Locally on client	Remotely on server
Expiration?	Variable – determined when cookie is set	Session is destroyed when the browser is closed
Size limit?	Depends on browser	Depends only on server (practically no size limit)
Accessing information?	<code>\$_COOKIE</code>	<code>\$_SESSION</code>
General use?	Remember small things about the user, such as login name. Remember things after re-opening browser	Remembering varying amount of data about the user in one browsing “session”. More sensitive info.



# PHP

## MySQL



Charles Liu

# Databases and MySQL



- Databases give us an easy way to issue “commands” to insert, select, organize, and remove data
- MySQL: open-source database, relatively easy to set up, easy to use with PHP
  - ▣ Other SQL databases, as well as non-SQL options such as MongoDB

# Connecting to MySQL

- MySQL database server can contain many databases, each of which can contain many tables

- Connecting to the server via PHP:

```
$db = mysql_connect(server, username, password);  
if (!$db) {  
    // terminate and give error message  
    die(mysql_error());  
}  
mysql_select_db(database_name, $db);
```

- `$db` is a database *resource type*. We use this variable to refer to the connection created

# Making SQL queries

- PHP function for making queries:

```
mysql_query(query_string, db_resource);
```

- Queries that return information, such as SELECT:  
returns a resource

```
$result = mysql_query(query_string, $db);
```

- ▣ In this case, this resource is stored in the variable `$result`
- Other queries, returns TRUE upon success.
- All queries return FALSE on failure. Best practice is to handle the error (e.g. `die(mysql_error())`)

# Retrieving information from a query

- Loop over the returned `$result` resource, row by row

```
$result = mysql_query(query, $db);  
while ($row = mysql_fetch_assoc($result)) {  
    $col1 = $row['column_1_name'];  
    $col2 = $row['column_2_name'];  
    // and so forth...  
}
```

# A shared database resource

---

- Don't repeat code - put database connection, select database code into the same file
- Reference the connection resource (`$db`) in other files (using `include($file_path)`)

# SQL queries

---

- ❑ `INSERT INTO table_name (col1, col2 ...) VALUES (val1, val2 ...)`
- ❑ `SELECT col1, col2 ... FROM table_name WHERE conditions`
- ❑ `CREATE TABLE table_name (column_name data_type(size), column_name data_type(size) ...)`

# The relational model

---

- Indicate *relations* between objects (rows) with an id
  - a pointer to a row in a different table
- The INNER JOIN



# what is a relational database?

a relation is a set of tuples

› tuple is ordered, set isn't

a relational database is

› a set of named relations

› with named columns

<i>users</i>				
<i>id</i>	<i>first</i>	<i>last</i>	<i>email</i>	<i>password</i>
1	Ann	Alert	aa@mit	aa
2	Chloe	Closure	cc@mit	blah
3	Ben	Bitdiddle	ben@mit	1010

<i>subjects</i>			
<i>id</i>	<i>by</i>	<i>name</i>	<i>category</i>
1	3	Lucid	2
2	2	Clover	1
3	3	Cosi	1

<i>reviews</i>				
<i>id</i>	<i>by</i>	<i>content</i>	<i>rating</i>	<i>about</i>
1	3	yummy!	5	2
2	2	neat	4	1

<i>categories</i>	
<i>id</i>	<i>name</i>
1	Food
2	Tech
3	Travel

# query operators

## relational algebra operators

- › select: filter rows by a predicate
- › project: filter by columns
- › product: combine two tables

## in SQL, all parts of select statement

```
-- show content and ratings of reviews about Clover  
select content, rating from subjects, reviews  
  where subjects.id = reviews.about and name = "Clover"
```

# deconstructing a query

<i>subjects</i>			
<i>id</i>	<i>by</i>	<i>name</i>	<i>category</i>
1	3	Lucid	2
2	2	Clover	1
3	3	Cosi	1

<i>reviews</i>				
<i>id</i>	<i>by</i>	<i>content</i>	<i>rating</i>	<i>about</i>
1	3	yummy!	5	2
2	2	neat	4	1

```
-- product operator (implicit in list of tables)  
select * from subjects, reviews
```

<i>id</i>	<i>by</i>	<i>name</i>	<i>category</i>	<i>id</i>	<i>by</i>	<i>content</i>	<i>rating</i>	<i>about</i>
1	3	Lucid	2	1	3	yummy!	5	2
1	3	Lucid	2	2	2	neat	4	1
2	2	Clover	1	1	3	yummy!	5	2
2	2	Clover	1	2	2	neat	4	1
3	3	Cosi	1	1	3	yummy!	5	2
3	3	Cosi	1	2	2	neat	4	1

examples from RazorSQL: available at <http://www.razorsql.com/>

# deconstructing a query

<i>subjects</i>			
<i>id</i>	<i>by</i>	<i>name</i>	<i>category</i>
1	3	Lucid	2
2	2	Clover	1
3	3	Cosi	1

<i>reviews</i>				
<i>id</i>	<i>by</i>	<i>content</i>	<i>rating</i>	<i>about</i>
1	3	yummy!	5	2
2	2	neat	4	1

-- selection operator (where)

```
select * from subjects, reviews
```

```
  where subjects.id = reviews.about and name = "Clover"
```

<i>id</i>	<i>by</i>	<i>name</i>	<i>category</i>	<i>id</i>	<i>by</i>	<i>content</i>	<i>rating</i>	<i>about</i>
1	3	Lucid	2	1	3	yummy!	5	2
1	3	Lucid	2	2	2	neat	4	1
2	2	Clover	1	1	3	yummy!	5	2
2	2	Clover	1	2	2	neat	4	1
3	3	Cosi	1	1	3	yummy!	5	2
3	3	Cosi	1	2	2	neat	4	1

<i>id</i>	<i>by</i>	<i>name</i>	<i>category</i>	<i>id</i>	<i>by</i>	<i>content</i>	<i>rating</i>	<i>about</i>
2	2	Clover	1	1	3	yummy!	5	2

# deconstructing a query

subjects			
id	by	name	category
1	3	Lucid	2
2	2	Clover	1
3	3	Cosi	1

reviews				
id	by	content	rating	about
1	3	yummy!	5	2
2	2	neat	4	1

```
-- projection operator (implicit in list of columns)
select content, rating from subjects, reviews
  where subjects.id = reviews.about and name = "Clover"
```

id	by	name	category	id	by	content	rating	about
1	3	Lucid	2	1	3	yummy!	5	2
1	3	Lucid	2	2	2	neat	4	1
2	2	Clover	1	1	3	yummy!	5	2
2	2	Clover	1	2	2	neat	4	1
3	3	Cosi	1	1	3	yummy!	5	2
3	3	Cosi	1	2	2	neat	4	1

id	by	name	category	id	by	content	rating	about
2	2	Clover	1	1	3	yummy!	5	2

content	rating
yummy!	5

# your turn

what does this query say?

```
select distinct name from subjects, reviews
  where rating = 5
```

```
-- lists all subject names: oops!
```

name
6005: Programming for Pleasure
6170: Spring 2012 Edition
Clover
Cosi
Lucid
Peets Coffee
Upton Tea

subjects

id	first	last	email	password
1	Chloe	Closure	cc@mit.edu	foo
2	Ben	Bitdiddle	bb@mit.edu	foo
3	Alice	Alert	aa@mit.edu	foo
4	Dee	Normalize	dee@mit.edu	password

catags

id	name
1	Food
2	Software
3	Education
4	Coffee Shops

reviews

id	by	content	rating	about
1	1	My favorite food truck. Delicious vegetar...	5	1
2	1	Diagramming software app. Cloud base...	2	2
3	2	I like this place too. And they have really...	4	1
4	2	Nice sandwiches and salads.	3	3
5	3	Great selection of teas, especially Indian...	5	4
6	3	Yeah, sure, the food is good. But what a...	1	1
7	1	This is a fun one! Almost no work, and a...	5	5
8	1	This class is not so good for my self este...	2	5
9	1	A nice place for a quick lunch.	3	3
10	4	I've never had so much fun in my life!	5	7
11	4	Pleasurable, but take 6170 for even mo...	4	5
12	1	Good to see the family product in use. C...	5	7

id	by	name	category	category_name
1	1	Clover	1	Food
2	1	Lucid	2	Software
3	2	Cosi	1	Food
4	3	Upton Tea	1	Food
5	1	6005: Programming for Pleasure	3	Education
6	1	Peets Coffee	4	Coffee
7	4	6170: Spring 2012 Edition	3	Education

# special operators

- › order by: sort the results by some column
- › sum, avg, count, max, min
- › group by: group rows before applying functions

```
-- show subjects and their average ratings
select name, avg(rating) from subjects, reviews
  where reviews.about = subjects.id group by subjects.id

-- show reviews ordered by rating
select name, content, rating from subjects, reviews
  where reviews.about = subjects.id order by rating
```

name	content	rating
Clover	Yeah, sure, the food is good. But what about the atmosphere? Especially in winter when it's snowing.	1
Lucid	Diagramming software app. Cloud based. Not at all bad, and likely to become better in the future. S...	2
6005: Programming for Pleasure	This class is not so good for my self esteem. Where are the closures?	2
Cosi	Nice sandwiches and salads.	3
Cosi	A nice place for a quick lunch.	3
Clover	I like this place too. And they have really cool strategy for taking orders, with people outside the truc...	4
6005: Programming for Pleasure	Pleasurable, but take 6170 for even more!	4
Clover	My favorite food truck. Delicious vegetarian dishes, and a relatively low ecoli count.	5
Upton Tea	Great selection of teas, especially Indian assams and darjeelings. Also lots of fancy teas (eg, their Oo...	5
6005: Programming for Pleasure	This is a fun one! Almost no work, and a laugh a minute! You'll have the time of your life. And just thi...	5
6170: Spring 2012 Edition	I've never had so much fun in my life!	5
6170: Spring 2012 Edition	Good to see the family product in use. Closures rule!	5

# PHP

## Conclusion



Charles Liu



# What we've talked about...

---

- Purpose of server-side programming
- Basic PHP syntax, arrays, functions
- Specifics to websites: cookies, sessions, HTTP requests and forms, MySQL
- Other server-side solutions:
  - ▣ ASP.NET
  - ▣ Python
- PHP's extensive documentation:  
<http://www.php.net/manual/en>

# PHP workshop and tomorrow

---

- Mostly to get you set up with a PHP server, write some simple code
- Tomorrow: more server-side frameworks
  - ▣ Node.js
  - ▣ Meteor.js
- 35-225, 11AM