503073

WEB PROGRAMMING & APPLICATIONS

CHAPTER 5: PHP - MYSQL

LESSON 07 – ADVANCED PHP

Instructor: Mai Van Manh

OUTLINE

1. HTML Form Handling
2. File Handling
3. Cookies
4. Session
5. Authentication
6. Error Handling

OUTLINE

1. **HTML Form Handling**
2. File Handling
3. Cookies
4. Session
5. Authentication
6. Error Handling

1. HTML Form Handling

- HTML Forms
- GET vs POST Method
- When to use GET and POST
- URL Encoding
- Handing POST and GET data in PHP

HTML Forms

- HTML forms are used to pass data to a server via serveral input elements.
- All input elements are placed in between **<form>** and **</form>**.

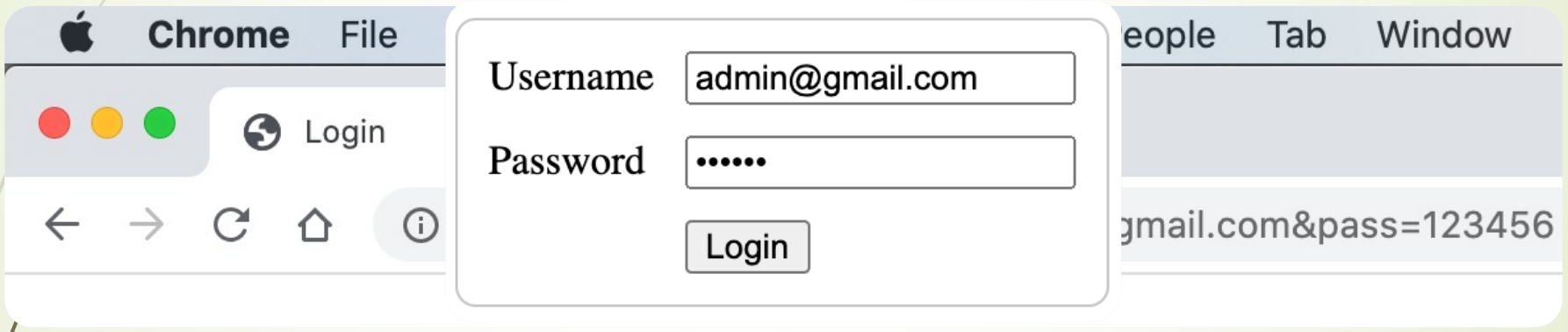
```
1 <form action="
2     Email: <in
3     Password:
4     <input typ
5 </form>
```

GET Method

- Information is **visible** to everyone.
- All variable names and values are **displayed** in the URL.
- GET Request **can be bookmarked**.
- Only **ASCII** characters are supported.
- GET has a limitation of **2KB payload**.

http://localhost/login.php?
email=admin%40gmail.com&pass=123456

GET Method


Chrome File
eople Tab Window
Login
Username admin@gmail.com
Password ••••••
Login
gmail.com&pass=123456

POST Method

- Information is **invisible** to everyone.
- All variable names and values are attached to **HTTP Body**.
- POST request **can not be bookmarked**.
- No restriction on data type, **binary data** is also supported.
- Support **unlimited** of data size.
- More **secure** than GET.

POST vs GET

- Use GET request for non-sensitive data.
- Use POST request for: sensitive data like username, password, bank transactions, file uploading. (Other security mechanisms should also be applied).
- Use GET request when we allow user to bookmark the page:

https://timkiem.vnexpress.net/?q=corona

URL Encoding

- Before the browser sends the information, it encodes it using a scheme called URL encoding.
- In this scheme, name/value pairs are joined with **equal signs** and different pairs are separated by the ampersand.

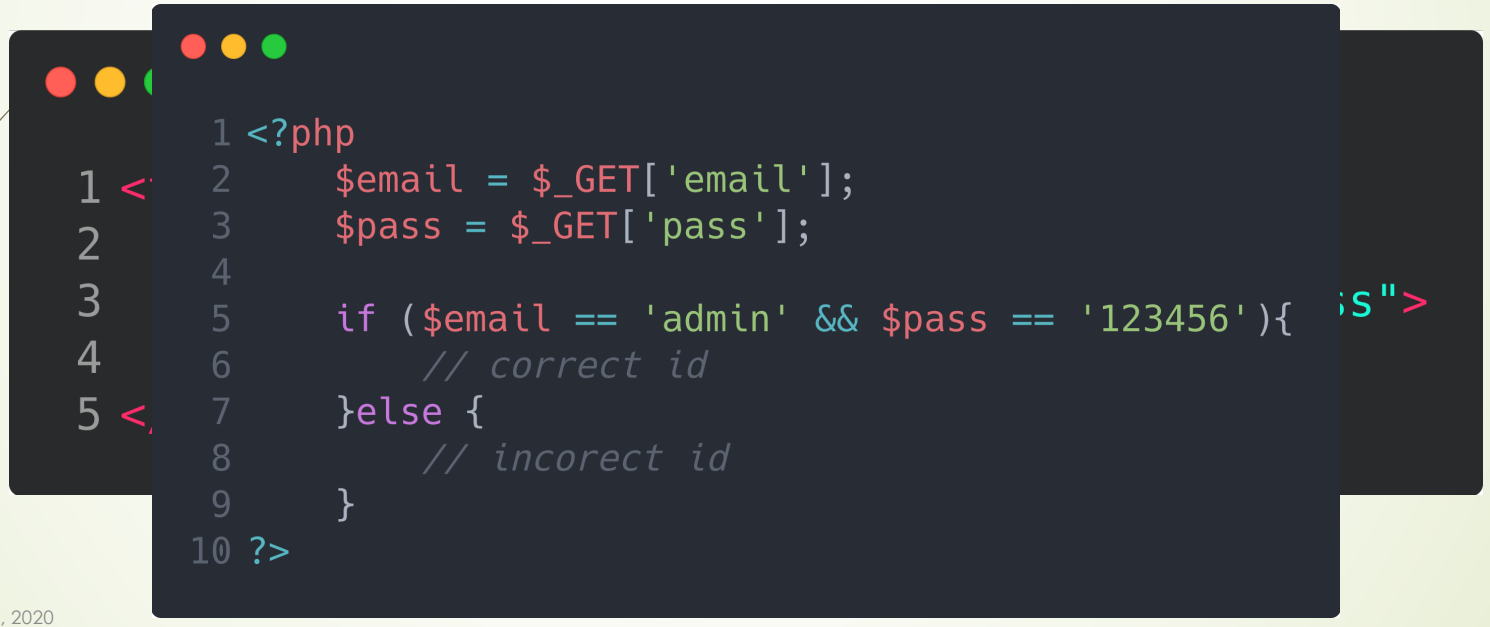
http://localhost/login.php?**email=abc@gmail.com&pass=123456**

- Spaces are removed and replaced with the + character and any other nonalphanumeric characters are replaced with a hexadecimal values.

Tuyển sinh → tuy%E1%BB%83n+sinh

Handling GET Request

- The PHP provides **$_GET** associative array to access all the sent information using GET method.



```php
<?php
    $email = $_GET['email'];
    $pass = $_GET['pass'];

    if ($email == 'admin' && $pass == '123456'){
        // correct id
    }else {
        // incorect id
    }
?>
```

Handling POST Request

- The PHP provides **$_POST** associative array to access all the sent information using POST method.

```php
<?php
    $email = $_POST['email'];
    $pass = $_POST['pass'];

    if ($email == 'admin' && $pass == '123456') {
        // correct id
    }else {
        // incorect id
    }
?>
```

PHP Form Validation

- We better check if data is available before accessing them in $_GET and $_POST.
- Isset()
  - returns false if the parameter is null
  - returns true if the parameter is not null, but the parameter may be empty.
- empty():
  - returns true if the parameter is null or zero-length string.
  - returns false if the parameter has value or has length greater than zero.

PHP Form Validation

- $a = NULL;
- $b = ‘’;
- $c = ‘Hello’;

	isset()	empty()
$a	false	true
$b	true	true
$c	true	false

PHP Form Validation


```php
<?php
    if (isset($_POST['email']) && isset($_POST['pass'])) {

        $email = $_POST['email'];
        $pass = $_POST['pass'];

        if (empty($email) || empty($pass)) {
            echo "Vui lòng nhập email & password";
        }
        else if ($email == 'admin' && $pass == '123456') {
            echo "Đăng nhập thành công";
        }else {
            echo "Sai email hoặc password";
        }
    }else {
        echo "Vui lòng sử dụng login form";
    }
?>
```

OUTLINE

1. HTML Form Handling
2. **File Handling**
3. Cookies
4. Session
5. Authentication
6. Error Handling

2. File Handling

- File handling is an important part of any web application.
- When you are manipulating files you must be very careful.
- Common errors are: editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.
- File handling in PHP is very similar to file handling in C language.

2. File Handling

- File reading
- File writing

Open a file

- The PHP **fopen()** is used to open a file for reading/writing.
- The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

```php
<?php
    $myFile = fopen('data.txt','rt')
        or die('Can not open data.txt');
?>

```

Open a file

- The file may be opened in one of the following modes:

Mode	Description
r	Open a file for read only
w	Write only. Opens and truncates the file; or creates a new file if it doesn't exist. Place file pointer at the beginning of the file.
a	Write only. Opens and writes to the end of the file or creates a new file if it doesn't exist.
r+	Read/Write. Starts at the beginning of the file.
w+	Read/Write. Opens and truncates the file; or creates a new file if it doesn't exist. Place file pointer at the beginning of the file.
a+	Read/Write. Preserves file content by writing to the end of the file.

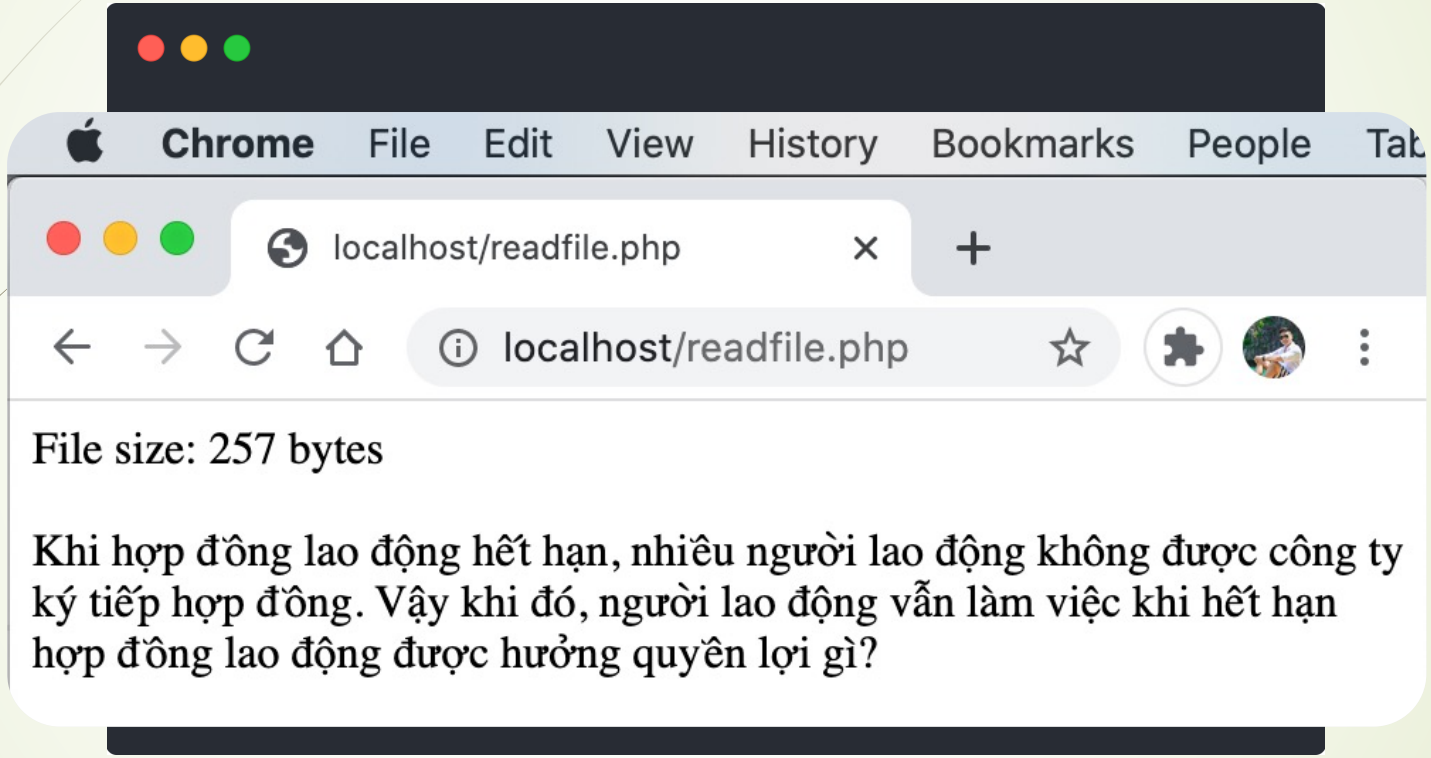
Read a file

- Here are the required steps to read a file with PHP.
  1. Open a file using fopen() function.
  2. Get the file's length using filesize() function.
  3. Read the file's content using fread() function.
  4. Close the file with fclose() function.

File reading functions

- **fread()**: read blocks of bytes
- **fgets()**: read a line
- **fgetc()**: read a character
- **file_get_contents()**: read all content of a file without opening.

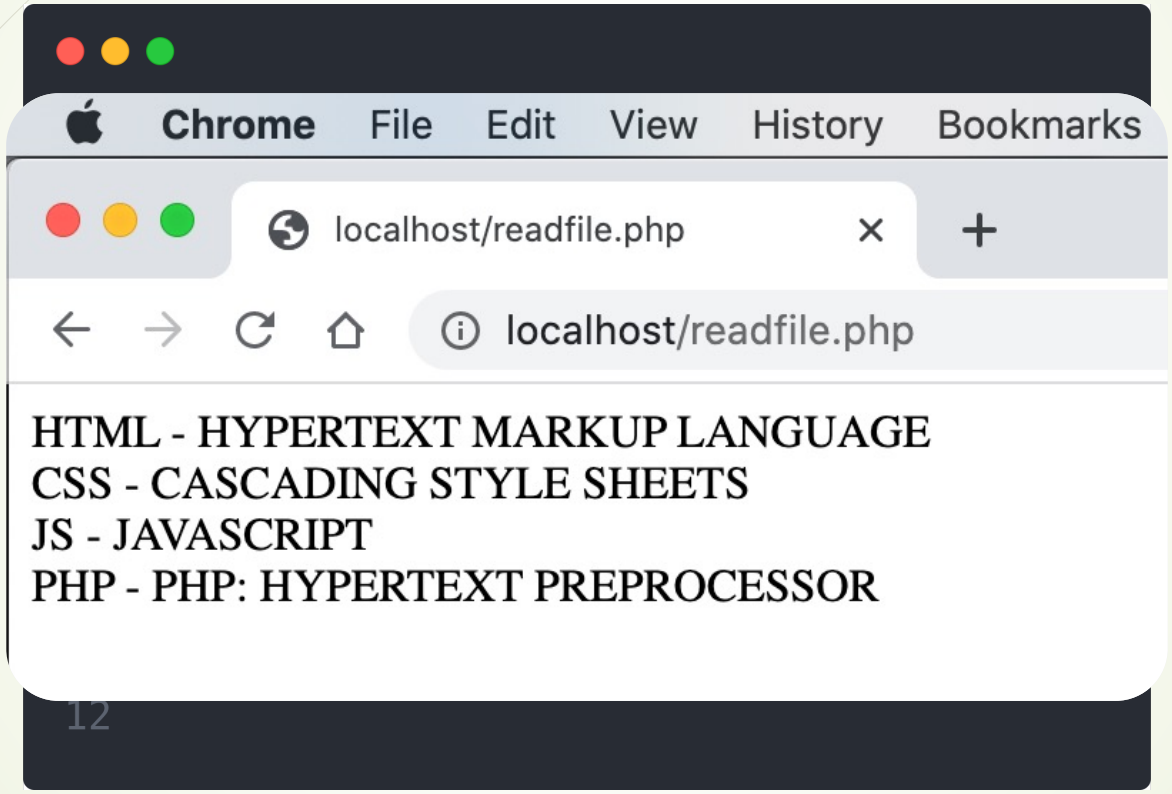
Read a file


Chrome File Edit View History Bookmarks People Tab
localhost/readfile.php
localhost/readfile.php
File size: 257 bytes
Khi hợp đồng lao động hết hạn, nhiều người lao động không được công ty ký tiếp hợp đồng. Vậy khi đó, người lao động vẫn làm việc khi hết hạn hợp đồng lao động được hưởng quyền lợi gì?

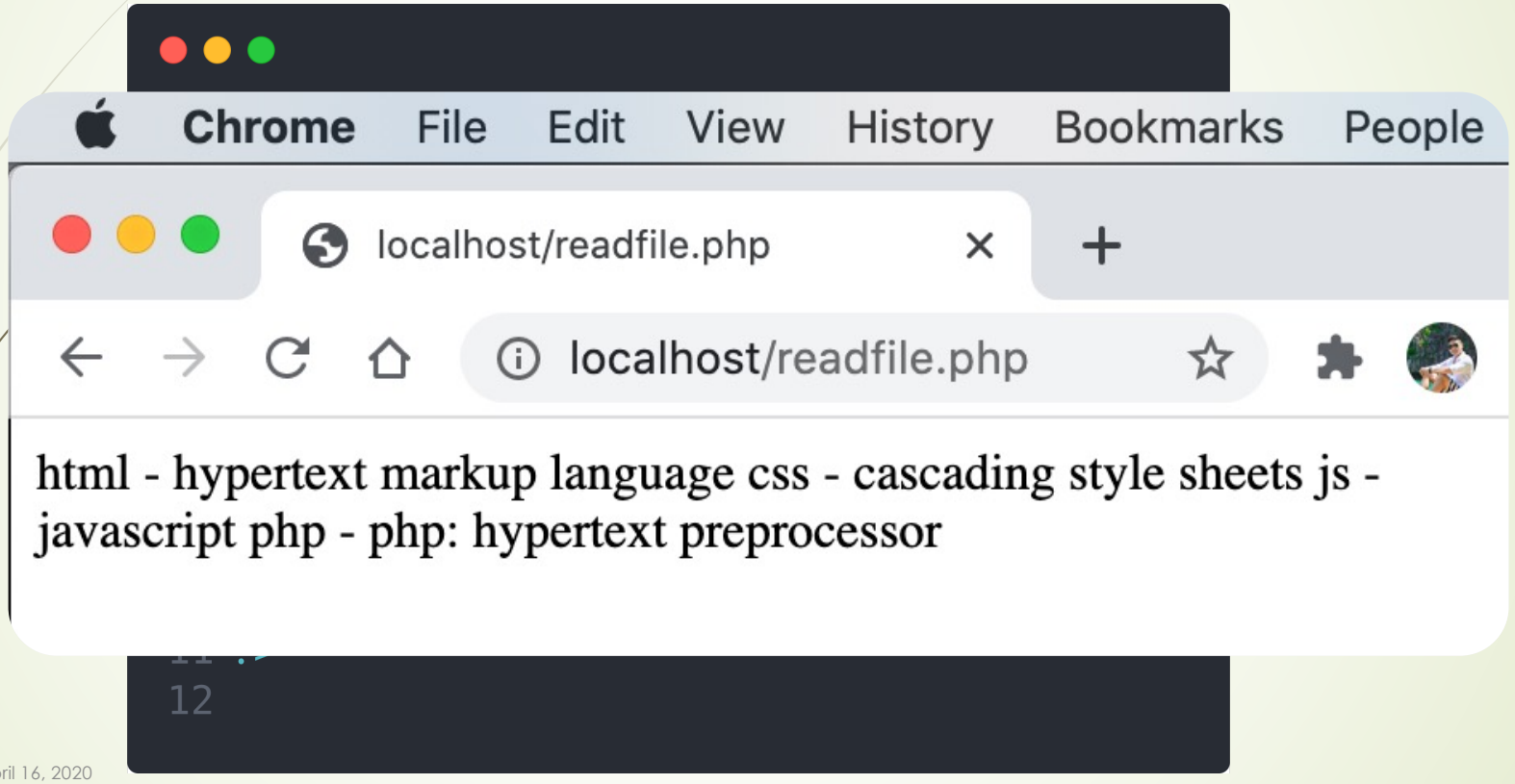
Read a file line by line?


```
1 HTML - Hypertext Markup Language
2 CSS - Cascading Style Sheets
3 JS - JavaScript
4 PHP - PHP: Hypertext Preprocessor
```

Read a file line by line



Read a file character by character



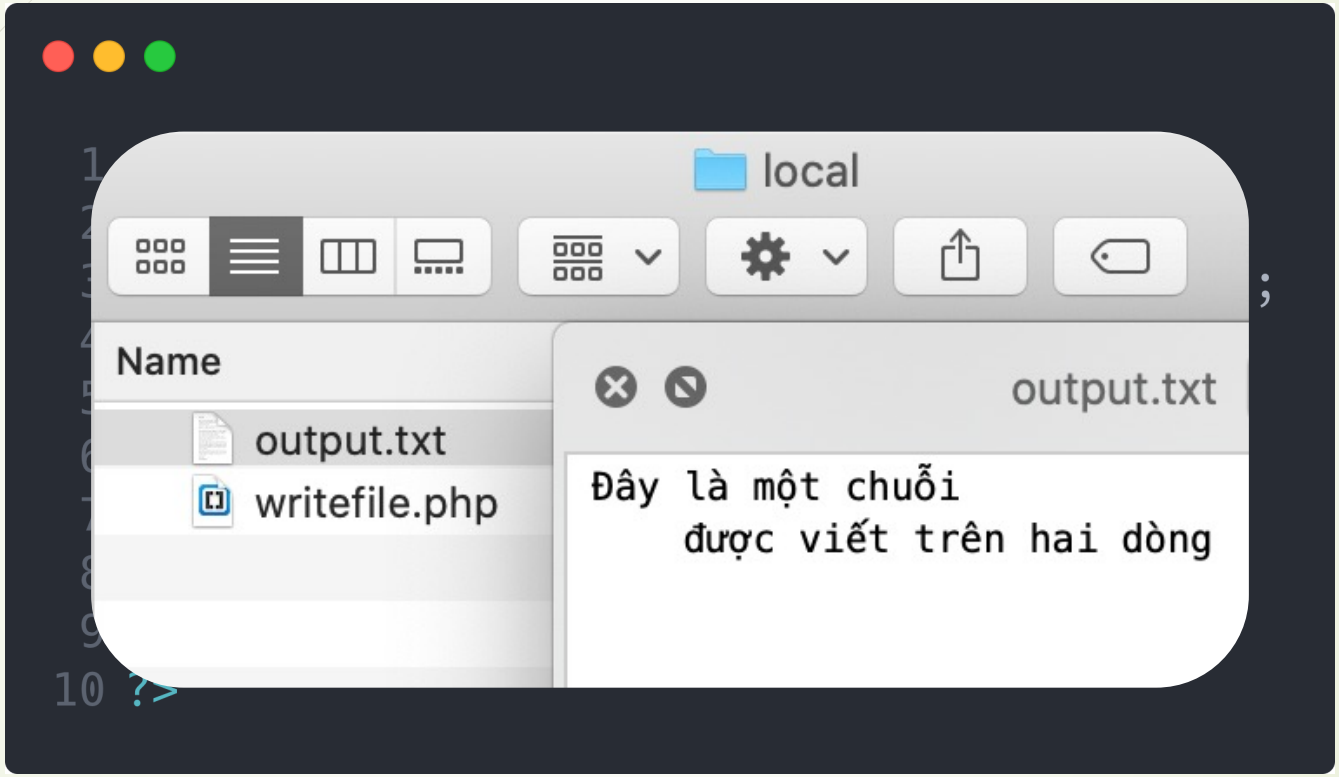
File writing

- Here are the required steps to write a file with PHP:
  1. Open a file using fopen() function.
  2. Check if the file has successfully opened.
  3. Write data to file using fwrite() function.
  4. Close the file with fclose() function.

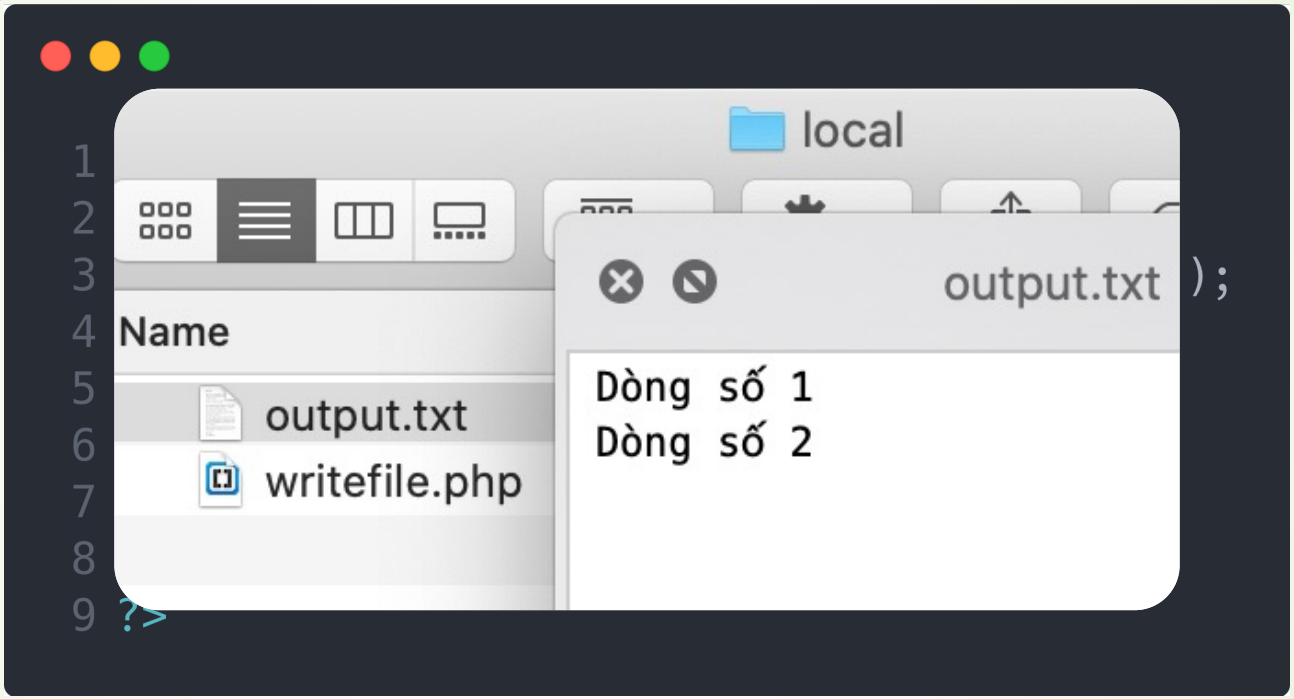
File writing functions

- **fwrite()**: write blocks of bytes
- **fgets()**: write a line
- **fgetc()**: write a character
- **file_put_contents()**: write all content of a file without opening.

Write a file


local
Name
output.txt
writefile.php
output.txt
Đây là một chuỗi
được viết trên hai dòng
10 ?>

Write a file line by line



OUTLINE

1. HTML Form Handling
2. File Handling
3. **Cookies**
4. Session
5. Authentication
6. Error Handling

3. Cookies

- What are cookies
- Create a cookie
- Modify a cookie value
- Delete a cookie

3. Cookies

- A cookie is a small file that the server embeds on the user's computer.
- Cookies are created when your browser visits a website that uses cookies to keep track of your movements within the site, help you resume where you left off, remember your registered login….
- Each time the same computer requests a page with a browser, it will send the cookie too.

3. Cookies

- Each cookie is effectively a small lookup table containing pairs of (key, data) values - for example (firstname, John) (lastname, Smith).

Key	Value
email	abc@gmail.com
pass	123456
key	23sadbas98sakjd924lkas

- By default the cookie is destroyed when the current browser window is closed.
- The time of expiry of a cookie can be set when the cookie is created.

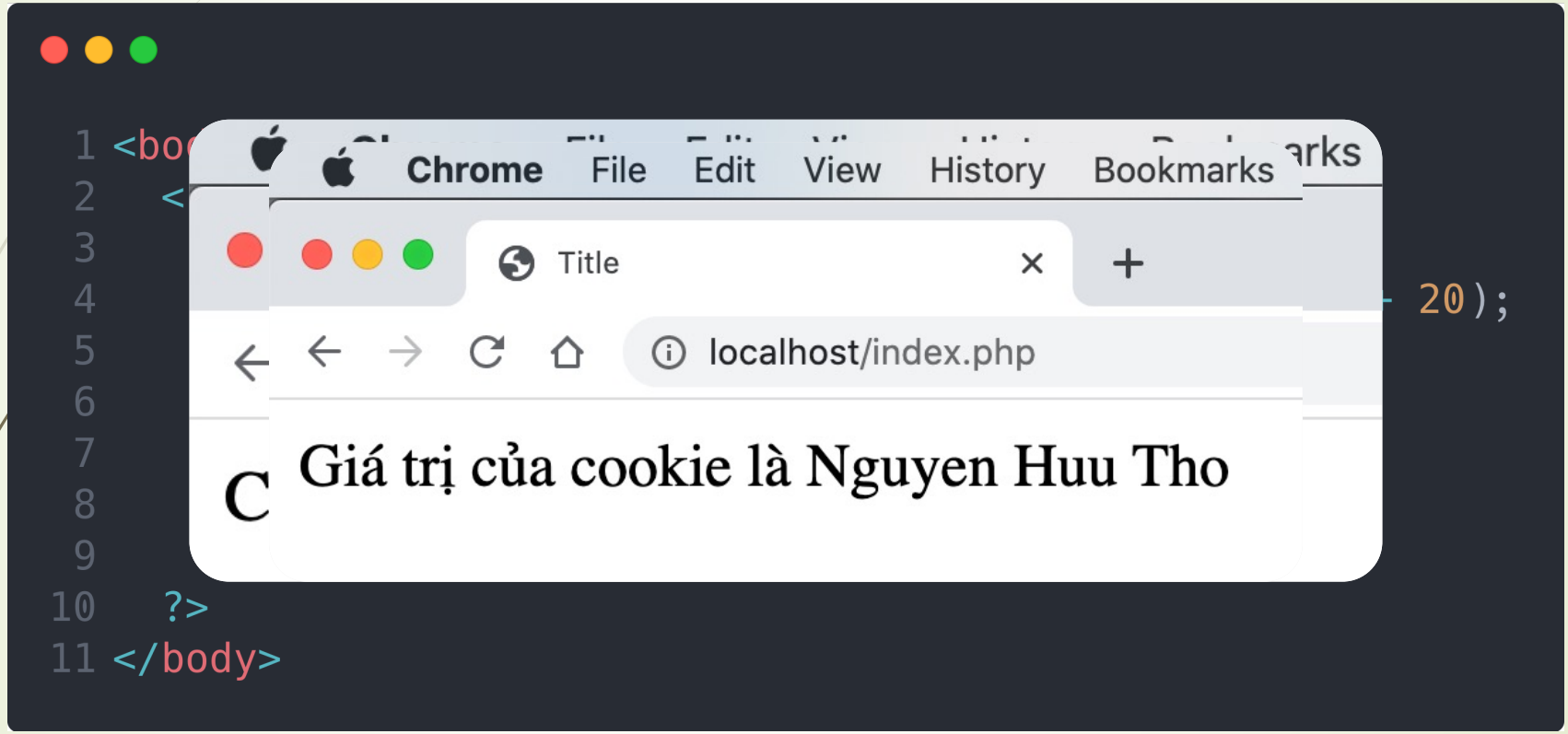
Create Cookies With PHP

- A cookie is created with the **setcookie**() function.

setcookie(*name, value, expire, path, domain, secure, httponly*);

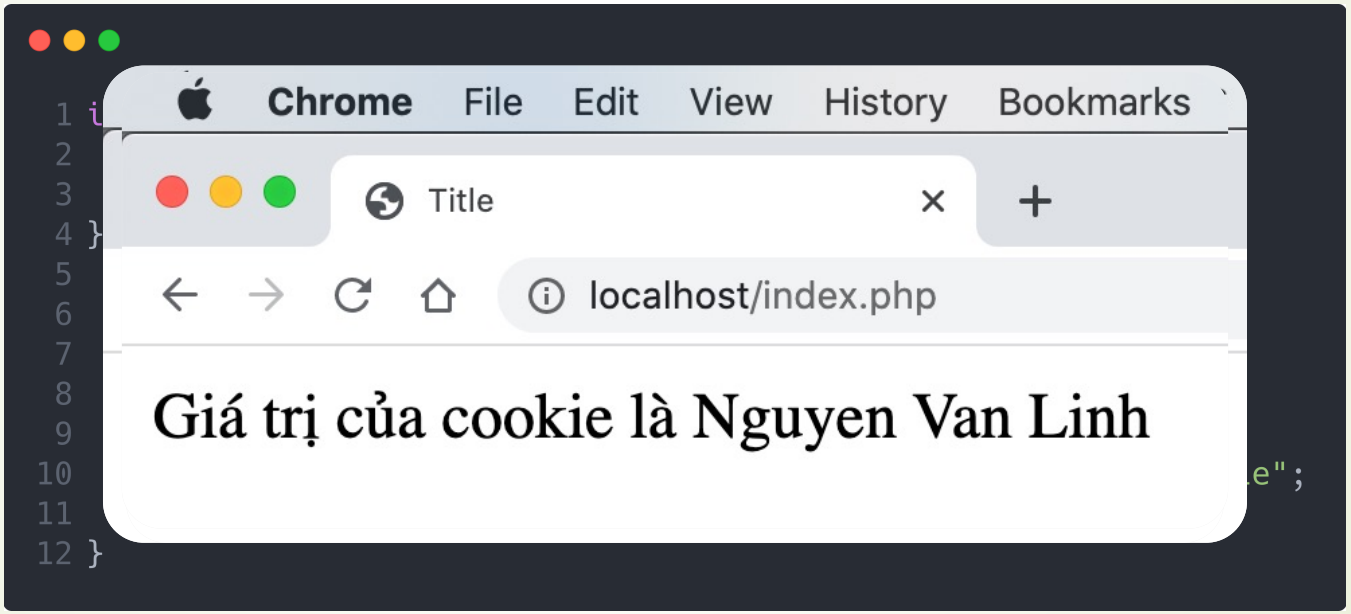
- Only the name parameter is required. All other parameters are optional.
- The value of the cookie is automatically URL encoded when sending the cookie, and automatically decoded when received.

Create Cookies With PHP



Modify a Cookie Value

- To modify a cookie, just set (again) the cookie using the **setcookie**() function.



Delete a Cookie

- To delete a cookie, use the **setcookie**() function with an expiration date in the past:

```
<body>
Array
(
    [Phpstorm-97d7f6fd] => f6693070-19ba-4f82-99b9-48c40d02ad8e
    [Phpstorm-97d7f6fe] => 0d66fcb0-7a54-48c5-8a38-54c28728d2b7
    [_ga] => GA1.1.151088546.1601028007
    [PHPSESSID] => 32jgok3m90nir2r4s4n0kauaai
)
</body>
```

OUTLINE

1. HTML Form Handling
2. File Handling
3. Cookies
4. **Session**
5. Authentication
6. Error Handling

4. Sessions

- What is session
- Start a session
- Get/Set PHP session values
- How PHP session works
- Destroy a PHP session

4. Sessions

- Unlike a cookie, the information is not stored on the users computer. Instead, it will be stored on web server.
- A session is a way to store information (in variables) to be used across multiple pages.
- Session variables hold information about one single user, and are available to all pages in one application.

How PHP Session Works

- Most sessions set a session-id on the user's computer that looks something like this: 765487cf34ert8dede5a562e4f3a7e12.
- When a client sends a request, the web server checks for session-id in cookies.
- If there is a match, it accesses that session from file (or database/cache).
- If not, it starts a new session and returns session-id as a cookie.
- By default in Apache, session will be expired after 24 mins of inactivity.

Start a PHP Session

- A session is started with the **session_start**() function.
- The **session_start**() function must be the very first thing in your document, before any HTML tags.
- Session variables are set with the PHP global variable: **$_SESSION**.

Get/Set PHP Session Variable Values


```php
<?php
    session_start();
?>
<html>
<body>
    <?php
        if (isset($_SESSION['user']) &&
            isset($_SESSION['email'])) {

            $user = $_SESSION['user'];
            $email = $_SESSION['email'];

            echo "User: $user, email: $email";
        }
        else echo 'Session chưa có thông tin';
    ?>
</body>
</html>
```

Destroy a PHP Session

- To remove all global session variables and destroy the session, use **session_unset**() and **session_destroy**():

```php
<?php
    // viết câu này trước tất cả các echo và
    // câu lệnh HTML
    session_start();
?>
<!DOCTYPE html>
<body>
    <?php
        // xóa tất cả giá trị có trong session
        session_unset();

        // hủy toàn bộ session
        session_destroy();
    ?>
</body>
</html>
```

OUTLINE

1. HTML Form Handling
2. File Handling
3. Cookies
4. Session
5. **Authentication**
6. Error Handling

5. Authentication

1. HTTP Authentication.
2. Authentication using session.

5.1 HTTP Authentication

- HTTP supports the use of several authentication mechanisms to control access to pages and other resources.
- These mechanisms are all based around the use of the 401 status (Unauthorized) code and the WWW-Authenticate, Authorization response headers.
- The most widely used HTTP authentication mechanisms are:
  - Basic: The client sends the username and password as unencrypted base64 encoded text.
  - Digest: The client sends a hashed form of the password to the server (MD5).

BASIC HTTP Authentication

- A method for an HTTP user agent to provide a username and password when making a request.
- A request contains a header field in the form of Authorization: Basic <credentials>, where credentials is the **base64** encoding of id and password joined by a single colon :
- Examples:

Base64(admin:123456)

BASIC HTTP Authentication

- When the server wants the user agent to authenticate itself towards the server, the server must respond appropriately to unauthenticated requests (401).

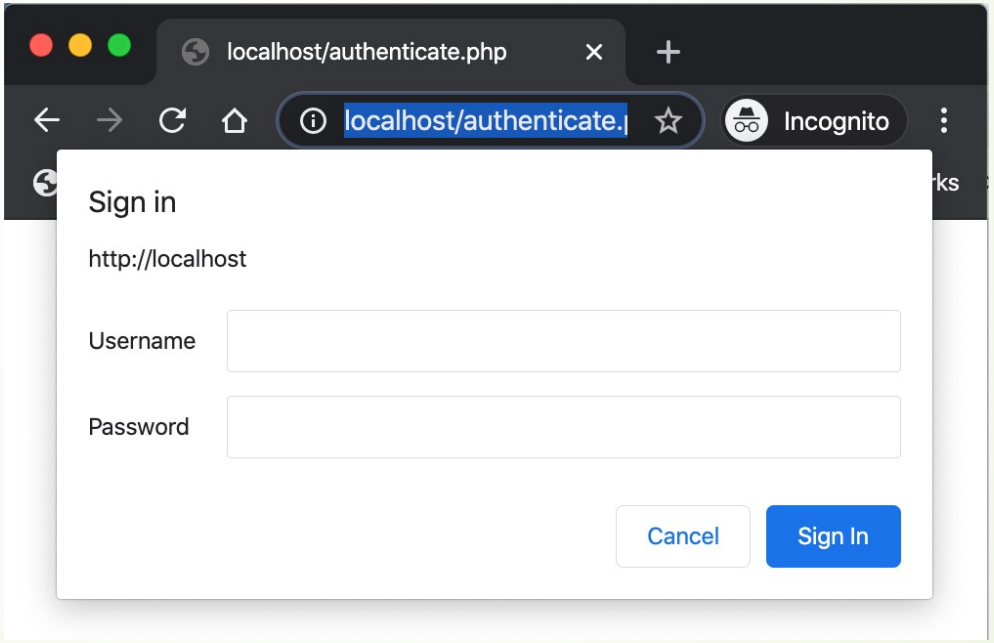
WWW-Authenticate: Basic realm= "Media"

- When the user agent wants to send authentication credentials to the server, it may use the Authorization field.

Authorization: Basic YWRtaW46MTIzNDU2

YWRtaW46MTIzNDU2 = Base64(admin:123456)

BASIC HTTP Authentication



BASIC HTTP Authentication


```php
<?php
  if (!isset($_SERVER['PHP_AUTH_USER'])) {
      header('WWW-Authenticate: Basic realm="My Realm"');
      header('HTTP/1.0 401 Unauthorized');
      echo 'Text to send if user hits Cancel button';
      exit;
  } else {
      echo "<p>Xin chào {$_SERVER['PHP_AUTH_USER']}.</p>";
      echo "<p>Password là: {$_SERVER['PHP_AUTH_PW']}.</p>";
  }
?>
```

5.2 Authentication using Session

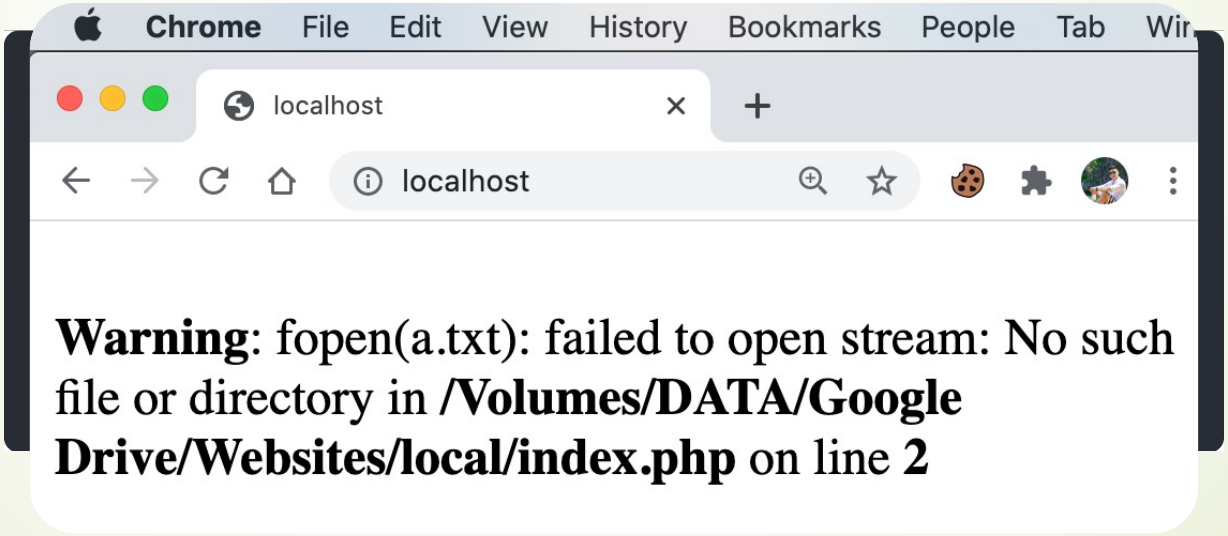
- Demo: PHP login page using session.

OUTLINE

1. HTML Form Handling
2. File Handling
3. Cookies
4. Session
5. Authentication
6. **Error Handling**

Error Handling

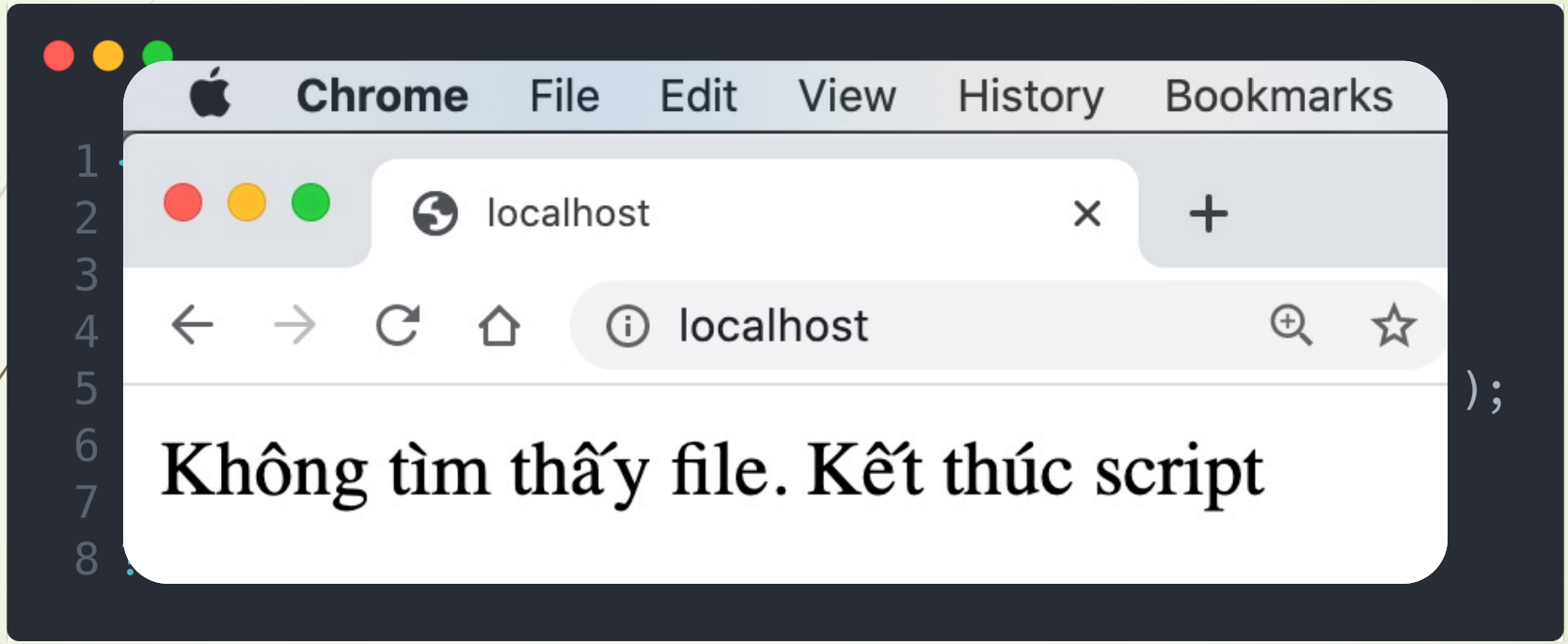
- When an error occurs: a message with filename, line number and a message describing the error is sent to the browser.
- If our code lacks error checking code, our program may look very unprofessional and we may be open to security risks.



Error Handling

- There different error handling methods:
  - die() function
  - Custom errors and error triggers
  - Error reporting

The die() function



Custom Error Handler

- We simply create a special function that can be called when an error occurs in PHP.

function error_function(error_level,error_message, error_file,error_line,error_context).

- Then use *set_error_handler()* to change the default error handler.

Error Report levels

Value	Constant	Description
1	E_ERROR	A fatal run-time error. Execution of the script is stopped
2	E_WARNING	A non-fatal run-time error. Execution of the script is not stopped
8	E_NOTICE	A run-time notice. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	A fatal user-generated error. This is like an E_ERROR, except it is generated by the PHP script using the function trigger_error()
512	E_USER_WARNING	A non-fatal user-generated warning. This is like an E_WARNING, except it is generated by the PHP script using the function trigger_error()
1024	E_USER_NOTICE	A user-generated notice. This is like an E_NOTICE, except it is generated by the PHP script using the function trigger_error()
2048	E_STRICT	Not strictly an error.
8191	E_ALL	All errors and warnings (E_STRICT became a part of E_ALL in PHP 5.4)

Custom Error Handler


```php
<?php
    function myErrorHandler($errno, $errstr) {
        echo "<b>Error:</b> [$errno] $errstr<br>";
        die();
    }
?>
```

Custom Error Handler

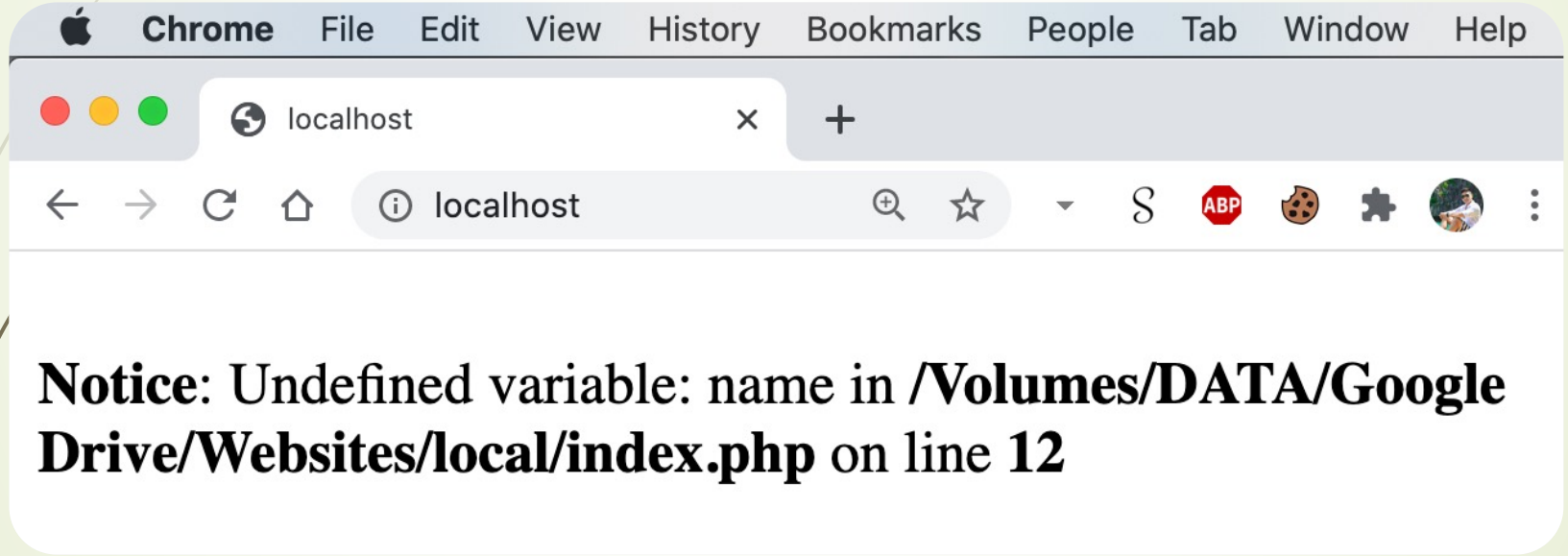

```php
<?php
    // định nghĩa hàm xử lý lỗi
    function myErrorHandler($errno, $errstr) {
        echo "<b>Error:</b> [$errno] $errstr<br>";
        die();
    }

    // đưa hàm vào sử dụng
    set_error_handler('myErrorHandler');

    // cố tình in một biến chưa khai báo
    echo $name;
?>
```

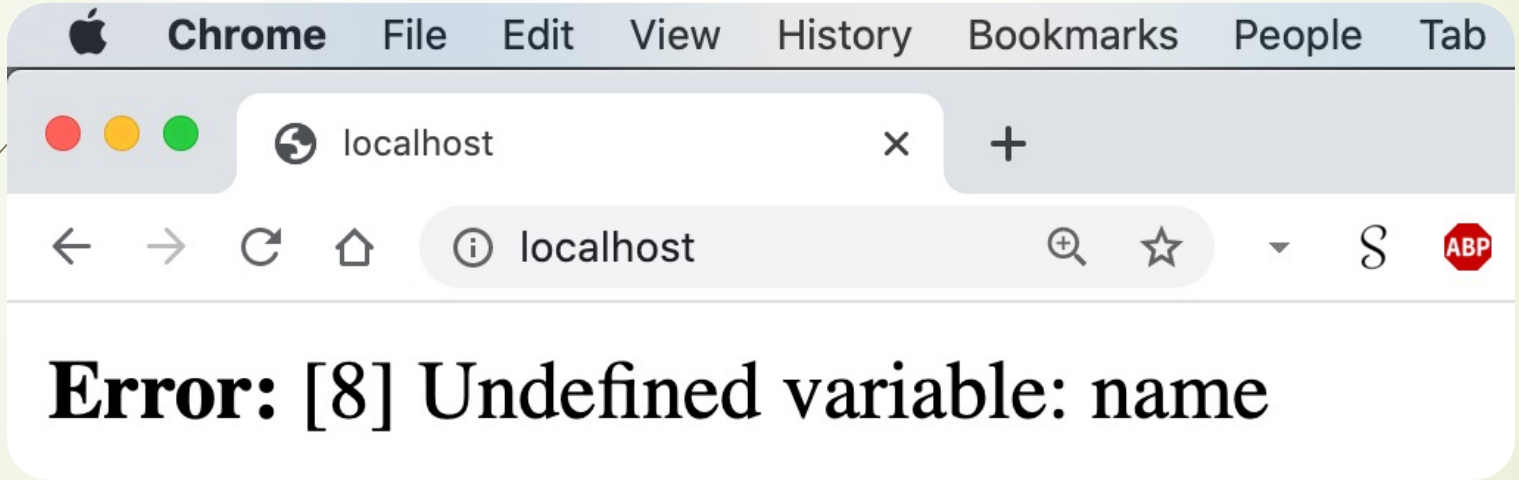
Custom Error Handler

- **Without** error handler

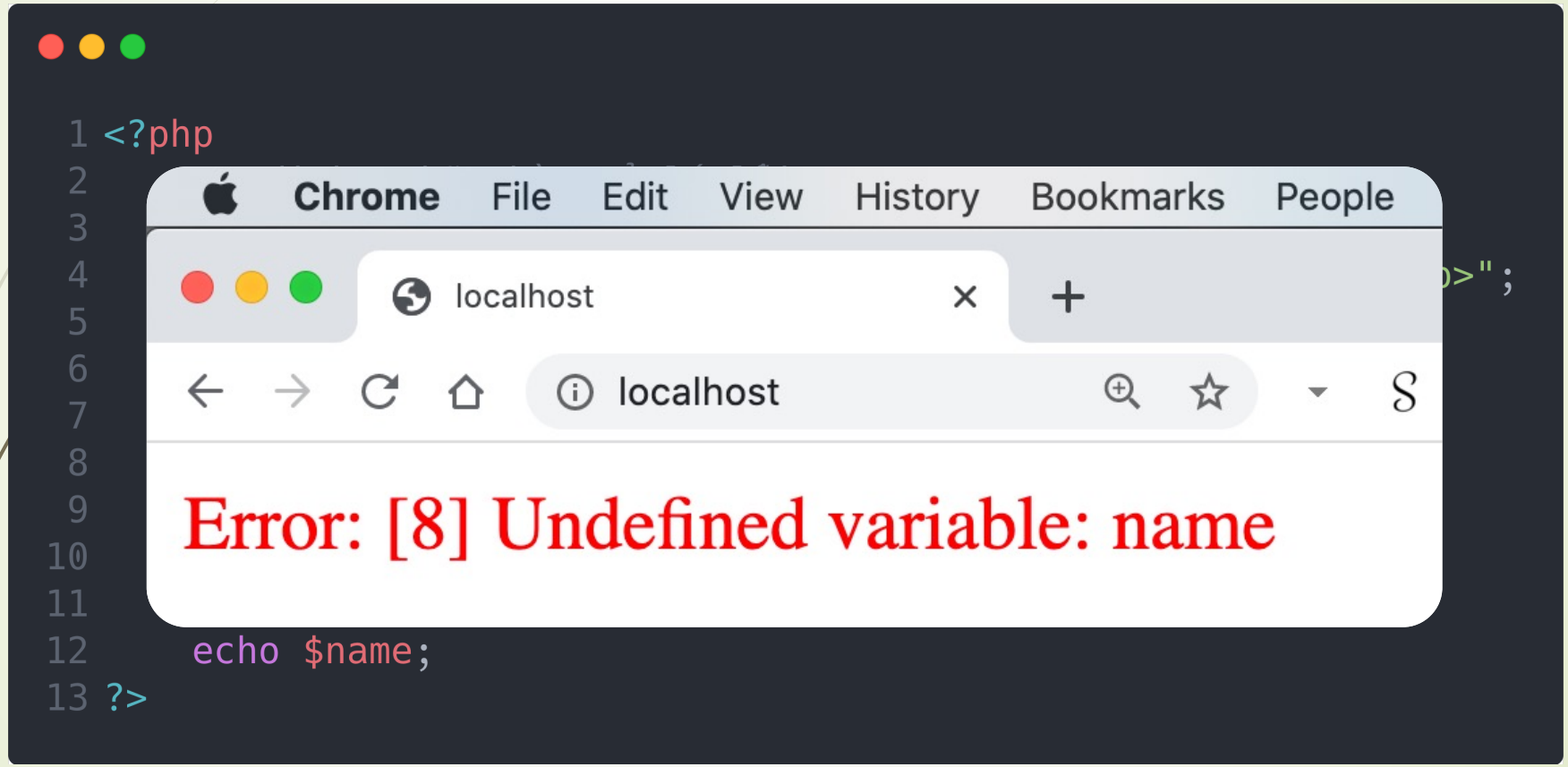


Custom Error Handler

- **With** error handler



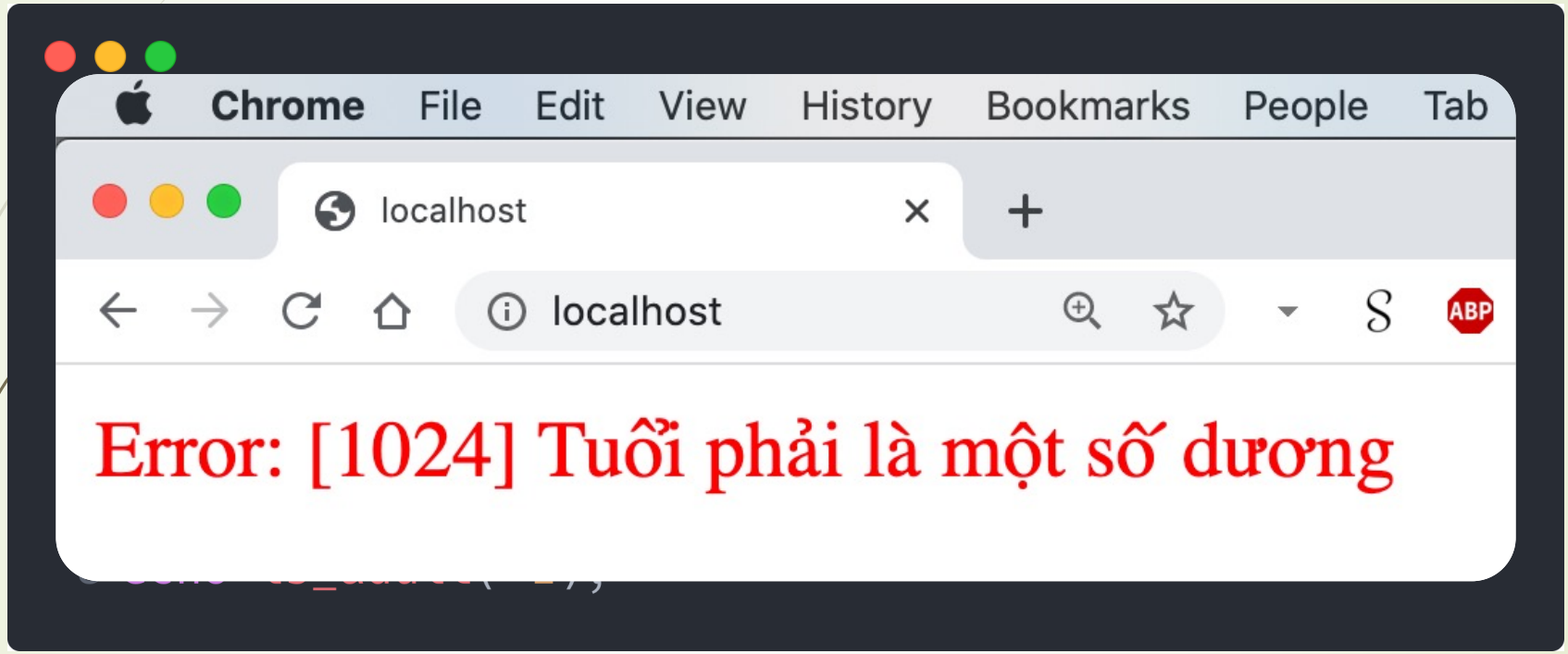
Custom Error Handler



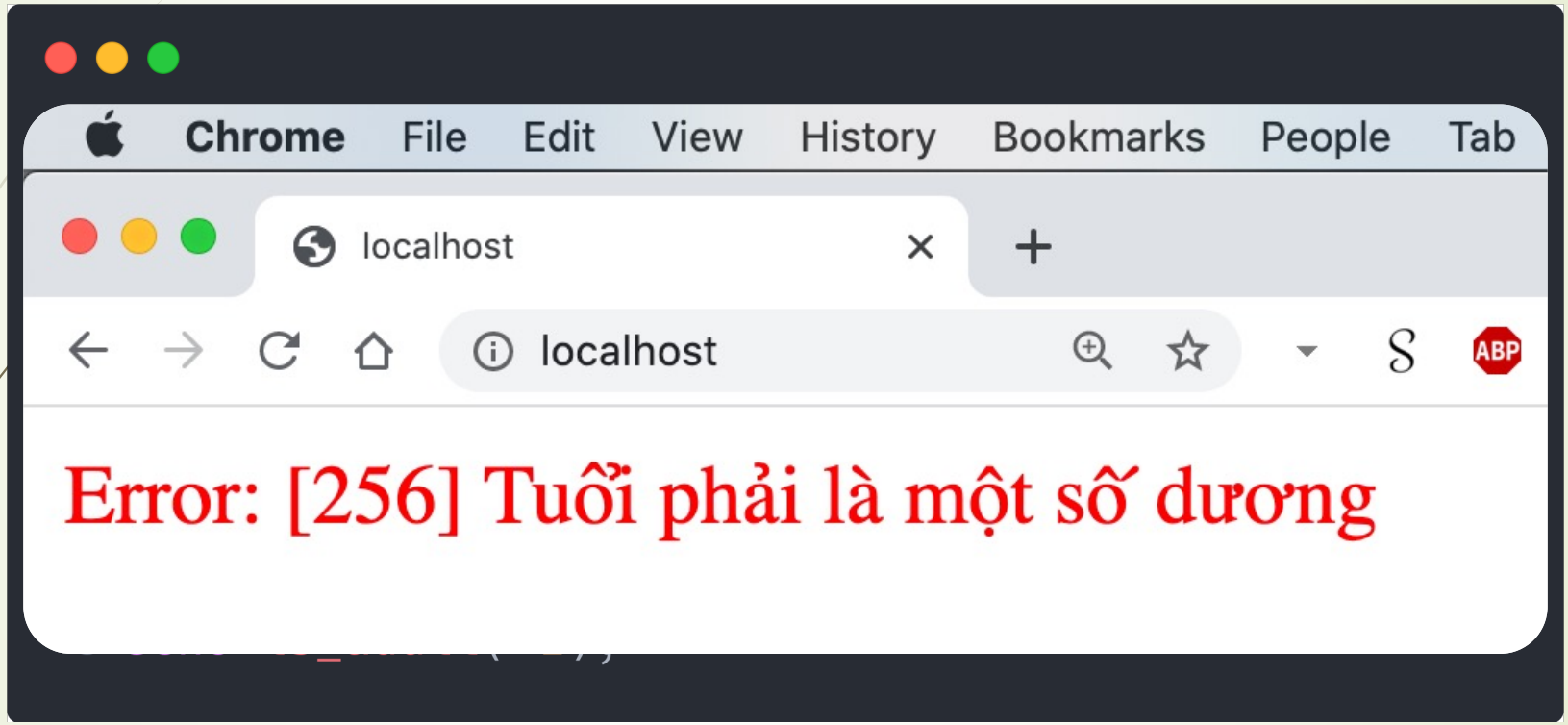
Trigger an Error

- In a script where users can input data it is useful to trigger errors when an illegal input occurs.
- In PHP, this is done by the trigger_error() function.

Trigger an Error



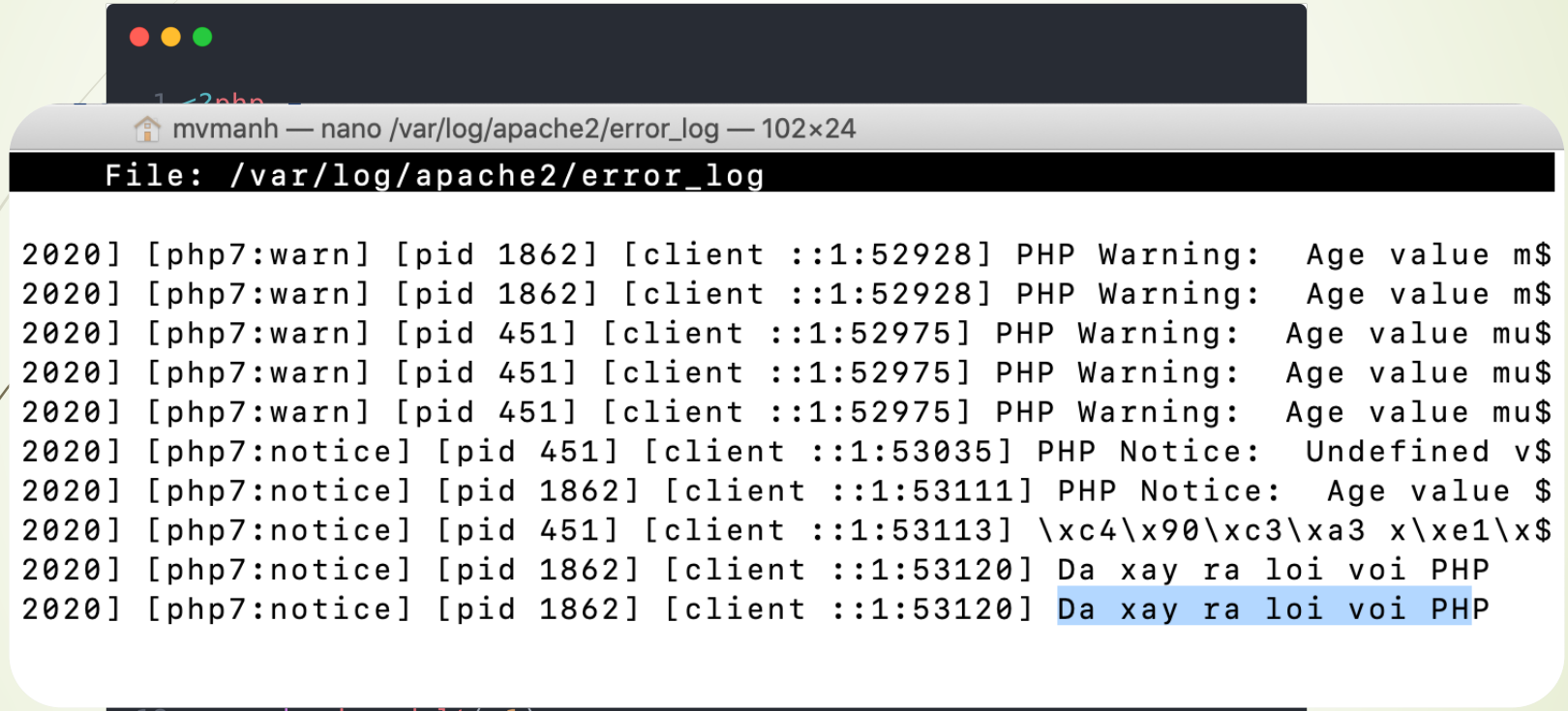
Trigger an Error



Error Logging

- By default, PHP sends an error log to the server's logging system or a file.
- By using the error_log() function you can send error logs to a specified file or a remote destination.

Error Logging



End

- Q&A