CLOSE ENCOUNTERS with PHP

Level 3

Validation & Security

Email & Date Validation

Where Are We With Our List?

app.php

```php
<?php
if($_SERVER['REQUEST_METHOD'] === 'POST')) {
    $date = trim($_POST['date']);
    $email = trim($_POST['email']);
    $description = trim($_POST['desc']);

    if (!empty($date) && !empty($email) && !empty($description)) {
        echo "<p>Date: $date</p>";
        echo "<p>Email: $email</p>";
        echo '<p>' . htmlspecialchars($description) . '</p>';
    }
}
```

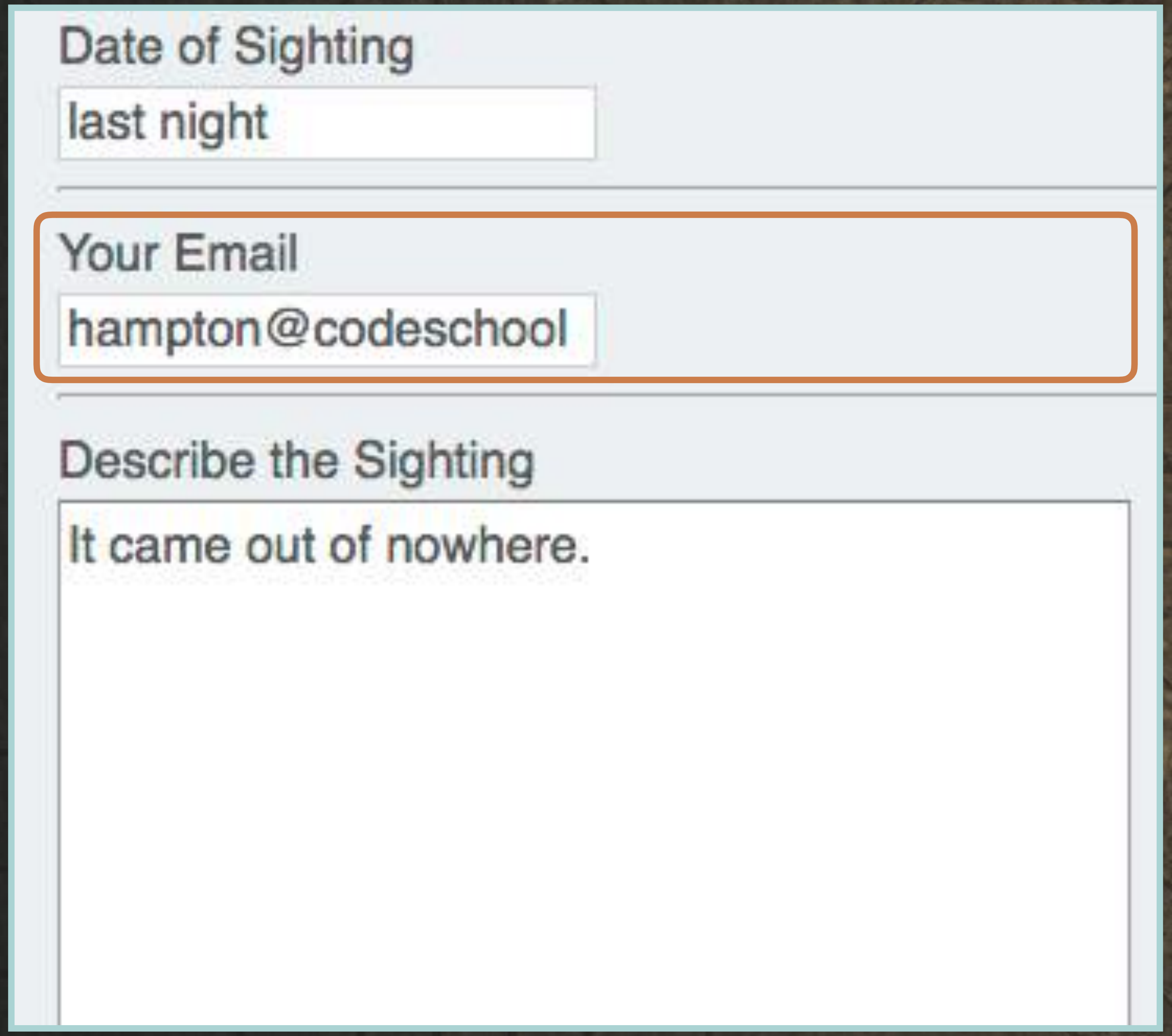
Validation to Do:

- $date exists ✓
- $email exists ✓
- $description exists ✓
- remove whitespace ✓
- sanitize output ✓
- validate email
- validate date

Validating the Email Address

Let's test to see if the email is valid before echoing the value.

Form Before Submit



Date of Sighting

last night

Your Email

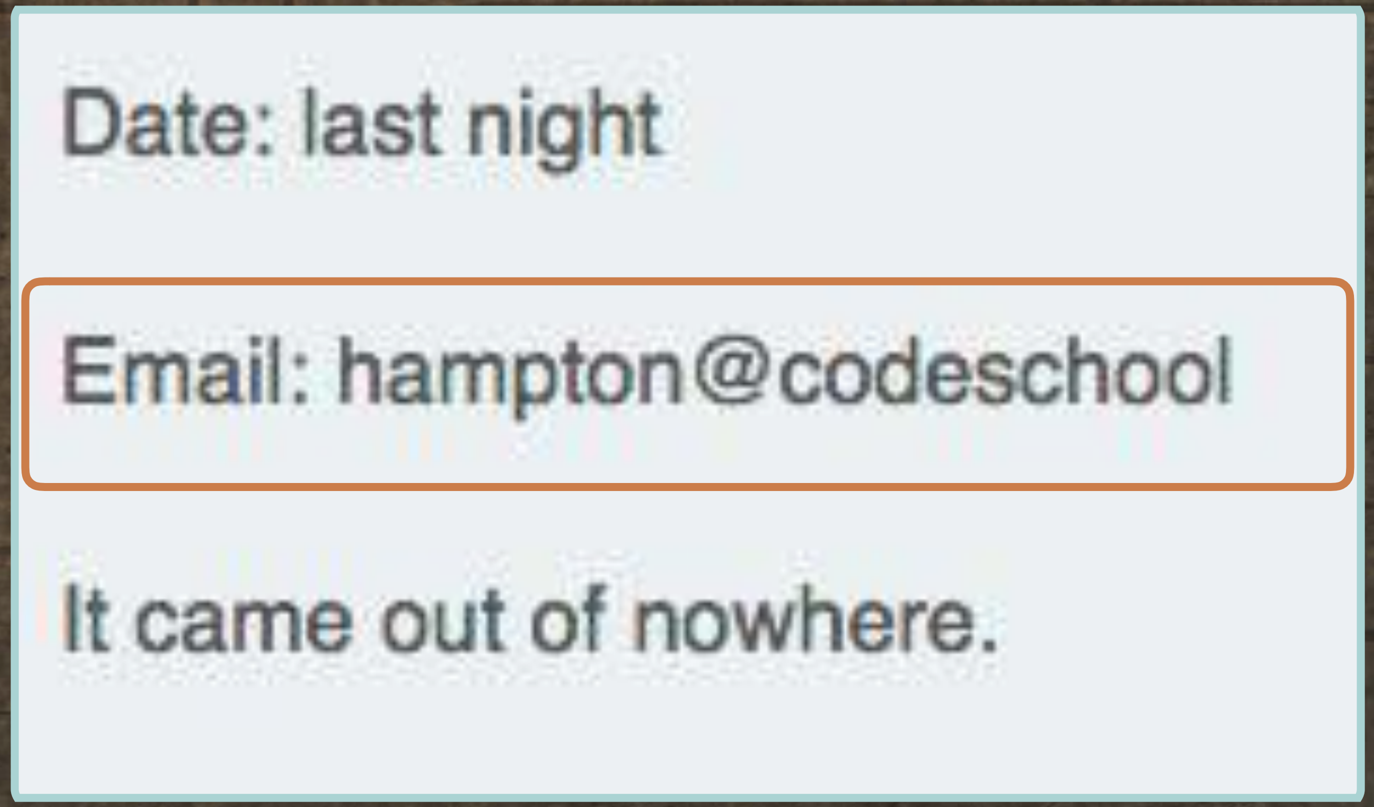
hampton@codeschool

Describe the Sighting

It came out of nowhere.

Using `hampton@codeschool` *as an example of an invalid email address*

Results After Submit



Date: last night

Email: hampton@codeschool

It came out of nowhere.

Submitting the form still echoes the invalid email!

We will need to test that our email complies with email address standards

CLOSE ENCOUNTERS with PHP

Validation of Email Address

Validation to Do:

- $date exists ✓
- $email exists ✓
- $description exists ✓
- remove whitespace ✓
- sanitize output ✓
- validate email ✓
- validate date

app.php

```php
<?php
if($_SERVER['REQUEST_METHOD'] === 'POST')) {
    $date = trim($_POST['date']);
    $email = trim($_POST['email']);
    $description = trim($_POST['desc']);

    if (!empty($date) && !empty($email) && !empty($description)) {
        echo "<p>Date: $date</p>";
        if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
            echo "<p>Email: $email</p>";
        }
        echo '<p>' . htmlspecialchars($description) . '</p>';
    }
}
```

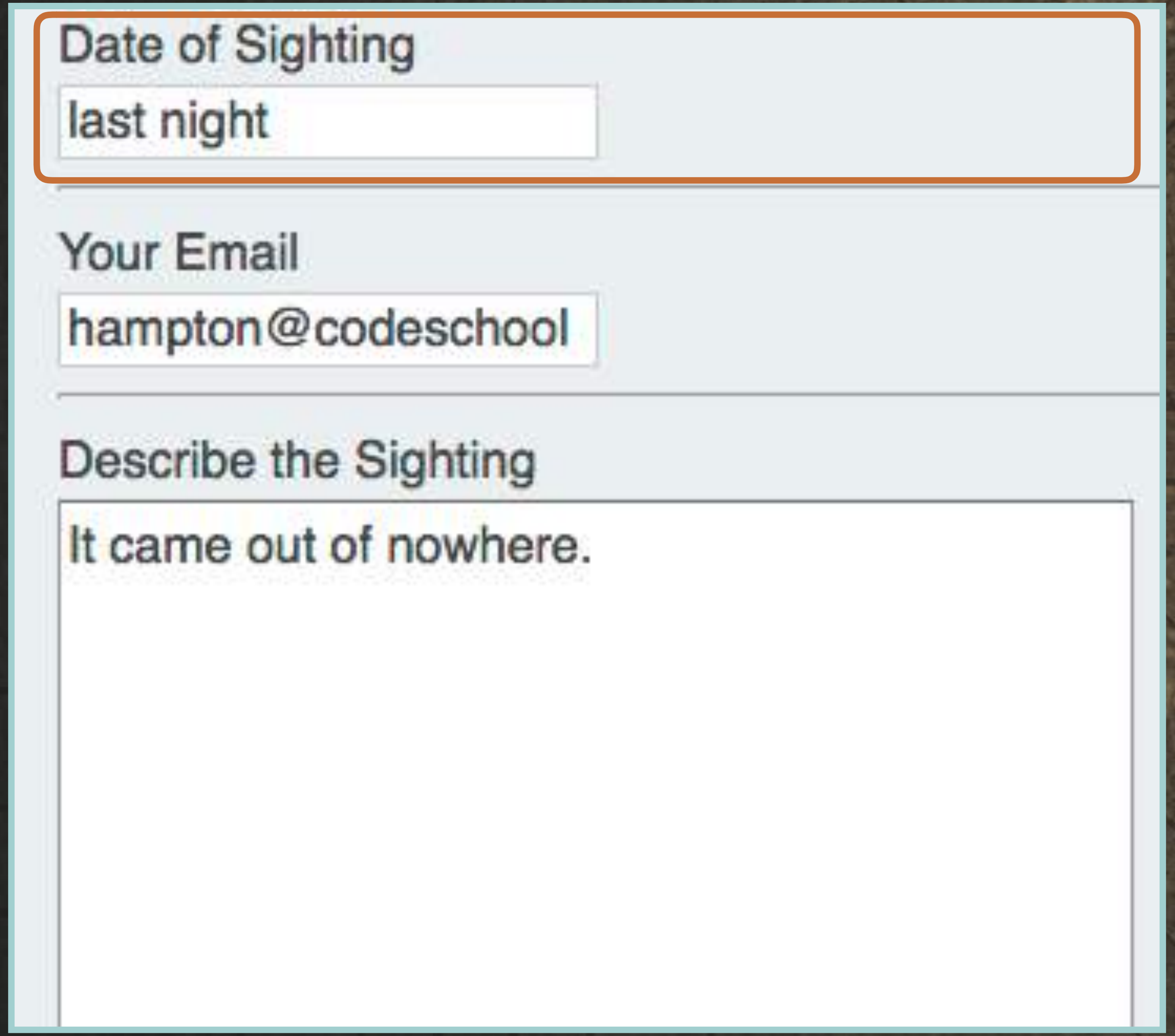
This is a PHP filter constant

`filter_var` *checks a variable against a filter and returns TRUE if it passes*

Validating the Date

Test to see if the date is valid, allow relative dates, and then format it.

Form Before Submit



Date of Sighting

last night

Your Email

hampton@codeschool

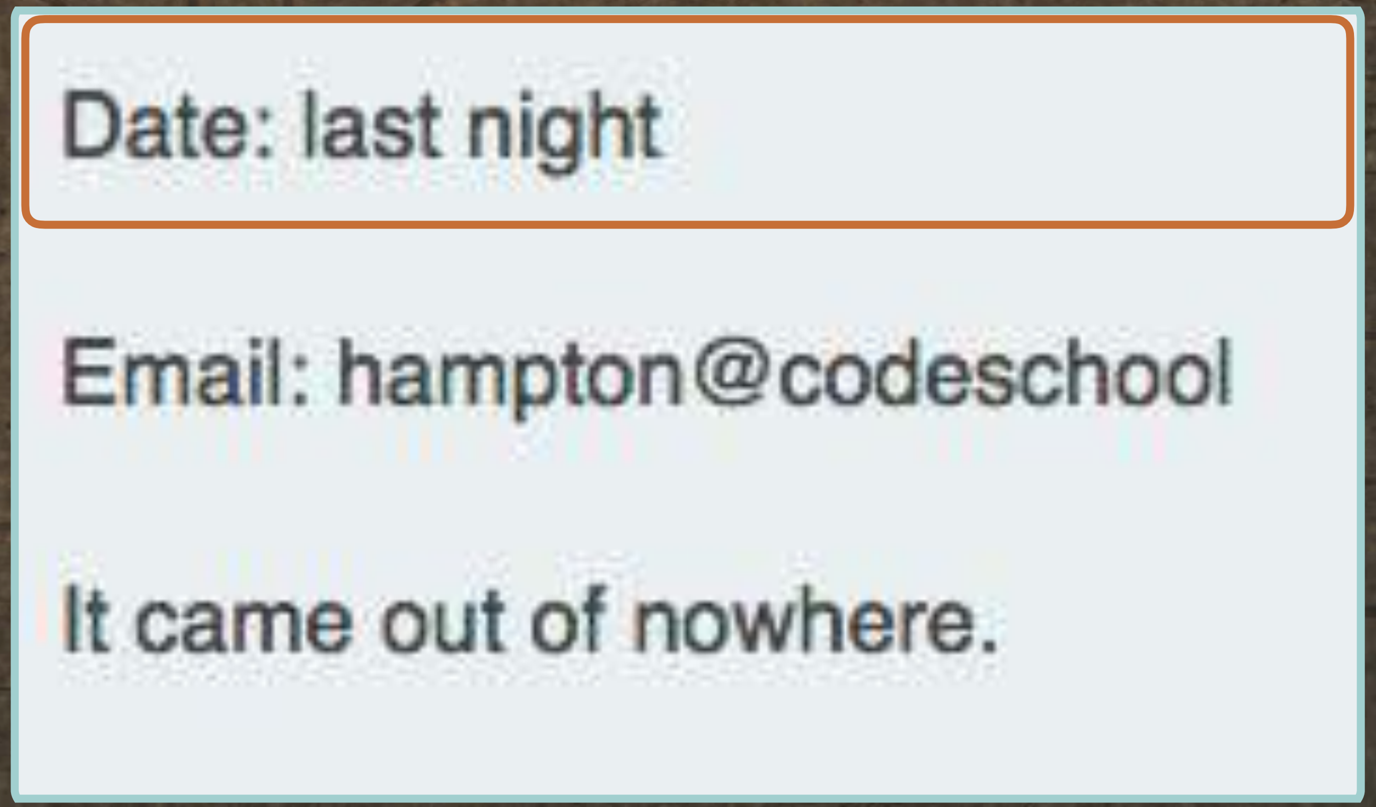
Describe the Sighting

It came out of nowhere.

Relative dates are fun, but this one is invalid.

We need to test that it is a valid date first!

Results After Submit



Date: last night

Email: hampton@codeschool

It came out of nowhere.

Dates need formatting

for UX consistency

CLOSE ENCOUNTERS with PHP

Validation of a Date

app.php

```
<?php
...
    if (!empty($date) && !empty($email) && !empty($description)) {

        if ($time = strtotime($date)) {
            echo "<p>Date: $date</p>";
        }

        if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
            echo "<p>Email: $email</p>";
        }
        echo '<p>' . htmlspecialchars($description) . '</p>';
    }
}
```

`strtotime` will convert most any date to a Unix timestamp

We are running `strtotime` and storing the timestamp in the `$time` variable

Relative Formats Using strtotime

The strtotime function accepts many date formats, including these relative formats.

```
strtotime('today');

strtotime('yesterday');
strtotime('tomorrow');

strtotime('last saturday of March 2010');

strtotime('30-June-2001');
strtotime('2001/7/30');
strtotime('June 30th 2001');
```

Will return midnight of the current day

Will also return midnight of the respective day

You can use more complex relative dates

Other than relative dates, you can use several different date formats

All of these will be converted to timestamps, which are measured in the number of seconds since Unix epoch (January 1 1970 00:00:00 GMT)

Converting the Timestamp Into a Date

app.php

```
<?php
...
    if (!empty($date) && !empty($email) && !empty($description)) {

        if ($time = strtotime($date)) {
            echo "<p>Date: date('F jS Y',$time)</p>";
        }

        if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
            echo "<p>Email: $email</p>";
        }
        echo '<p>' . htmlspecialchars($description) . '</p>';
    }
}
```

date *will make our timestamp human readable*

If the date is 1-1-2000, 'F jS Y' *will return* 'January 1st 2001'

Date Format Strings

The date function can take many different format strings as well as some PHP constants.

```
date('m/d/Y', $timestamp);
> 03/14/2015
```

→ *Both month and day have leading zeros*

```
date('F jS Y', $timestamp);
> March 14th 2015
```

→ *Full month with ordinal suffix*

```
date('l \t\h\e jS \o\f F', $timestamp);
> Saturday the 14th of March
```

→ *Using \ to escape characters*

```
date('W', $timestamp);
> 11
```

→ *This will return the week number of 2015*

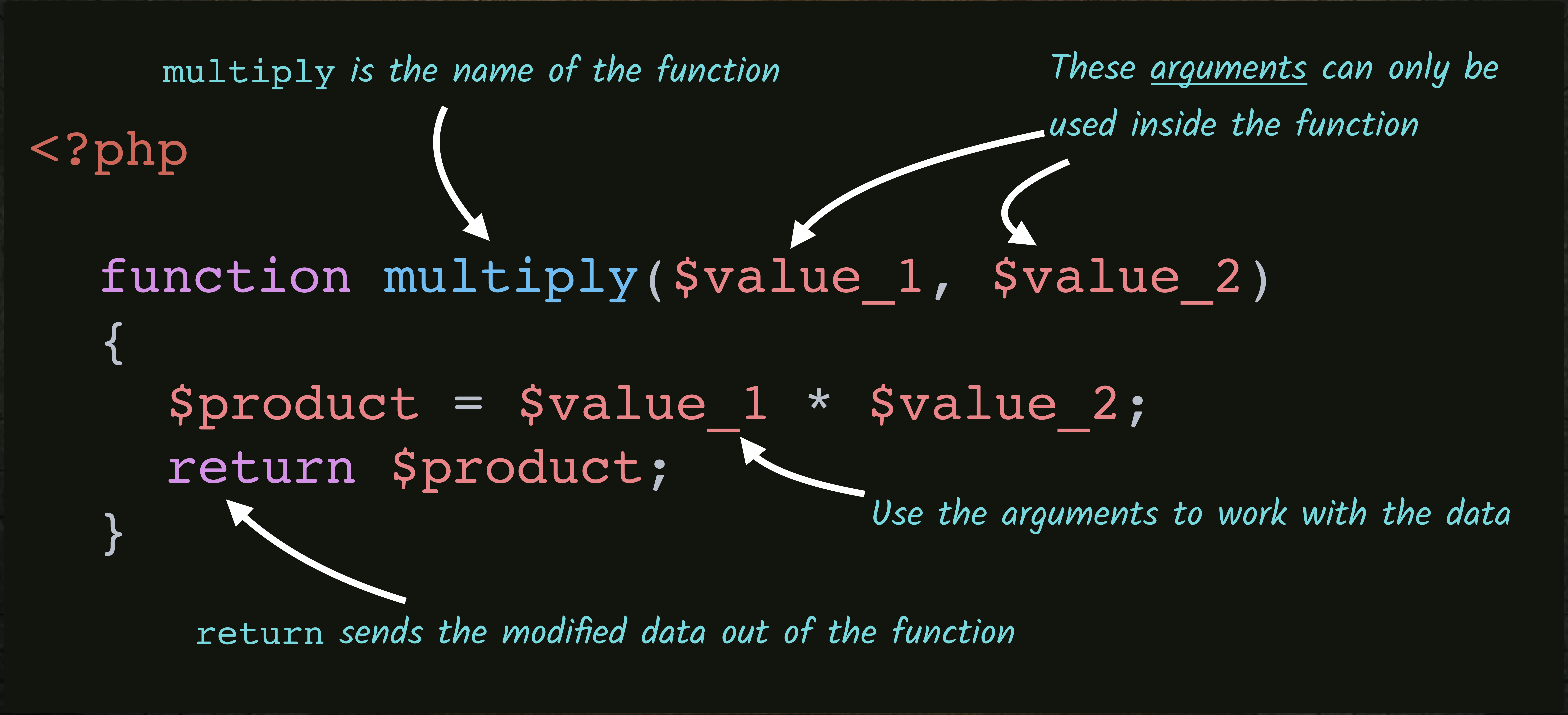
```
date(DATE_ATOM, $timestamp);
> 2015-03-14T00:00:00+00:00
```

→ *ATOM, which is the format for MySQL*

Even more formatting options can be found in the docs at go.codeschool.com/php-date

Custom Function for Validation

We can create a reusable block of custom code called a function.



```php
<?php

  function multiply($value_1, $value_2)
  {
    $product = $value_1 * $value_2;
    return $product;
  }
```

multiply is the name of the function
These arguments can only be used inside the function
Use the arguments to work with the data
return sends the modified data out of the function

Custom Function for Validation

We can create a reusable block of custom code called a function.

```php
<?php

    function multiply($value_1, $value_2)
    {
        $product = $value_1 * $value_2;
        return $product;
    }
```

`echo multiply(5, 7);` $\longrightarrow$ *35*

`echo multiply(42, 0);` $\longrightarrow$ *0*

`echo multiply(3, 14);` $\longrightarrow$ *42*

`echo multiply(12, 24);` $\longrightarrow$ *288*

No matter what combination of integers we feed into the function, we will always get the product of the two

Creating a Function for Validation

app.php

validate_date *is the name of our function*

```php
<?php
    function validate_date($date_string)
    {
        if ($time = strtotime($date_string)) {
            return date('F jS Y', $time);
        } else {
            return $date_string . ' does not look valid.';
        }
    }
```

Return a string error with the date_string *included*

```php
...

    if (!empty($date) && !empty($email) && !empty($description)) {

        if ($time = strtotime($date)) {
            echo "<p>Date: date('F jS Y',$date)</p>";
        }
        if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
```

Using Our New Function

app.php

```php
<?php
    function validate_date($date_string)
    {
        if ($time = strtotime($date_string)) {
            return date('F jS Y', $time);
        } else {
            return $date_string . ' does not look valid.';
        }
    }
```

This will output our error message, or the formatted date

```php
...

    if (!empty($date) && !empty($email) && !empty($description)) {

        echo validate_date($date);

        if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
```

Custom Validation Recap!

Let's walk through what we learned in this section.

- PHP's **filter_var** combined with built-in constants to validate our email address
- The **strtotime** function and converting relative/human-readable dates into Unix timestamps
- The date function and all its different types of formatting options
- Creating and using custom functions

CLOSE ENCOUNTERS with PHP