Escaping output from PHP

How to escape values that PHP code writes into HTML, attributes, URLs, JSON, CSV, shell commands and logs - htmlspecialchars() with explicit flags for PHP 8.0, the wash operator and why templates are the better place...

Source file: doc/eng-us/help/php-developers/escaping-output-from-php.md Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/php-developers/escaping-output-from-php.md

This page is for PHP developers whose code writes values into output: HTML from a view or operator, JSON from an endpoint, CSV from an export, a shell command, a log line. The rule is simple: escape every value where it leaves your code, in the format of the place it goes. The template-side view of the same rule is Input validation and output escaping.

Contents

1. Prefer templates
2. HTML from PHP
3. Attributes, URLs and data for scripts
4. JSON
5. CSV, shell and logs
6. The context table
7. Mistakes that cause cross-site scripting

Prefer templates

HTML belongs in templates, where `|wash` escapes each value at the point it is printed:

```
<h2>{$item.title|wash}</h2>
<a href={concat( 'acmereport/item/', $item.id )|ezurl} title="{$item.title|wash}">{'Open'|i18n( '
extension/acmereport' )}</a>
```

A view passes values to the template (`$tpl->setVariable()`), not HTML. Then the escaping is visible in one place and a reviewer can check it.

HTML from PHP

When PHP must build markup (a template operator that returns HTML, an error message, an e-mail body), escape each value with explicit flags:

PHP

```php
function acmeEscape( $value )
{
    return htmlspecialchars( (string)$value, ENT_QUOTES | ENT_SUBSTITUTE | ENT_HTML401, 'UTF-8' )
;
}

$html = '<span class="acme-status">' . acmeEscape( $item->attribute( 'status' ) ) . '</span>';
```

Pass the flags every time. PHP 8.1 changed the default to `ENT_QUOTES | ENT_SUBSTITUTE | ENT_HTML401`; PHP 8.0's default leaves `'` unescaped, so a value inside a single-quoted attribute would break out on PHP 8.0. The kernel passes the flags explicitly in `wash` and 27 other files since 6.0.15 (PHP 8.0 support).

Attributes, URLs and data for scripts

- **Attributes**: quote every attribute and escape the value as above.
- **URLs**: build site URLs with `eZURI::transformURI()` (PHP) or `|ezurl` (templates); encode a part of a URL with `rawurlencode()`, and only allow `http` and `https` for addresses that came from input.
- **Data for JavaScript**: never print values into an inline `<script>`. Put them in a `data-*` attribute or a `<script type="application/json">` block that your external script reads. The admin's Content Security Policy refuses inline scripts and `style=""` (Content Security Policy for your code).

PHP

```php
$attr = acmeEscape( json_encode( array( 'id' => (int)$item->attribute( 'id' ) ) ) );
$html = '<div class="acme-chart" data-conf="' . $attr . '"></div>';
```

JSON

PHP

```php
header( 'Content-Type: application/json; charset=utf-8' );
echo json_encode( $data, JSON_HEX_TAG | JSON_HEX_AMP | JSON_HEX_APOS | JSON_HEX_QUOT );
```

The `JSON_HEX_*` flags make the output safe even if it ends up inside HTML. Set the `Content-Type`, so a browser never renders the answer as HTML. Through ezjscore or REST, return arrays and let the layer encode them.

CSV, shell and logs

Output	Escape
CSV opened in spreadsheets	`fputcsv()` for quoting; prefix values starting with =, +, - or @ so they are not run as formulas (the kernel's CSV export does)
shell commands	avoid them; otherwise `escapeshellarg()` for every argument
log lines	remove line breaks (`str_replace( array( "\r", "\n" ), ' ', $value )`), so one entry cannot fake another; never log secrets or personal data
e-mail headers	remove line breaks from every value

The context table

Output goes into	In PHP	In templates
HTML text	`htmlspecialchars( $v, ENT_QUOTES | ENT_SUBSTITUTE | ENT_HTML401, 'UTF-8' )`	`|wash`

Output goes into	In PHP	In templates
HTML attribute	the same, attribute quoted	`|wash`, attribute quoted
site URL	`eZURI::transformURI()`	`|ezurl`
part of a URL	`rawurlencode()`	`|rawurlencode`
JSON	`json_encode()` with `JSON_HEX_*`	`json_encode()|wash` in an attribute
SQL	`$db->escapeString()`, casts	(no SQL in templates)

Mistakes that cause cross-site scripting

- A number printed without escaping "because it is a number": a later change makes it text.
- `htmlspecialchars( $v )` without flags in code that must run on PHP 8.0.
- Escaping on input and storing escaped text: double escaping in one place, none in the next output that forgets.
- Building HTML in a view and passing it to a template that prints it raw.
- A value from the request echoed in an error message.
- A JSON value printed inside an inline script.

Related pages

- Security in PHP code
- Writing code that runs on PHP 8.0
- String template operators
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

Become an Exponential Partner

Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.

Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.