Loading JavaScript

How a design's scripts reach the page with ezscript_load and ezscriptfiles, the design.ini script lists, scripts required by one template, deferred bundles, configuration as JSON data instead of inline code, and jQuery…

Source file: `doc/eng-us/help/designers/theme-design/loading-javascript.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/designers/theme-design/loading-javascript.md

This page is for designers who add behavior to a design: a menu toggle, a slider, a cookie banner. It answers: which operator prints scripts, how to defer them, how a template asks for a script only when needed, and how to pass settings to a script without inline code.

Contents

1. The operators
2. In the pagelayout
3. Deferred scripts
4. A script for one kind of page
5. Script lists in design.ini
6. Passing settings to a script
7. Rules

The operators

From `extension/ezjscore/autoloads/ezjscpackertemplatefunctions.php`:

Operator	Does
`ezscript_load( files, type, language, charset, pack_level )`	prints <script> tags for the files given plus every file queued with `ezscript_require`
`ezscript_require( files )`	queues files for the next `ezscript_load` (prints at once if it already ran)
`ezscript( files, ... )`	prints tags for the files given only
`ezscriptfiles( files, pack_level, ignore_loaded )`	returns packed URLs, so you write the <script> tag yourself

Files are relative to `javascript/` and found through the design cascade. Default pack level is 2 (merged and whitespace removed). A name with `::` is a generator, not a file: `ezjsc::jquery` prints jQuery from the local copy configured in `ezjscore.ini [eZJSCore] LocalScripts[]` (jQuery 4.0.0 with jQuery Migrate 4.0.2 on this release).

In the pagelayout

At the end of the body, so the page paints before scripts run:

Template

```
    {ezscript_load( array( 'ezjsc::jquery', 'mytheme.js' ) )}
</body>
```

Scripts that must run before the first paint (for example a theme switch that sets a class on `<html>`) go in the head, as small separate files.

Deferred scripts

`ezscript_load` writes plain `<script src>` tags. For `defer`, ask for the URLs and write the tag:

Template

```
{foreach ezscriptfiles( array( 'gallery.js', 'consent.js' ), 2, true() ) as $script_url}
<script defer src="{$script_url}"></script>
{/foreach}
```

`true()` as third parameter (`ignore_loaded`) keeps this bundle independent of files already printed. The media design does exactly this for its non-critical scripts (`pagelayout/head/script.tpl`).

A script for one kind of page

Template

```
{* override/templates/full/gallery.tpl *}
{ezscript_require( array( 'gallery.js' ) )}
```

The pagelayout's `ezscript_load` prints it. Pages without galleries never download it.

Script lists in design.ini

`settings/design.ini [JavaScriptSettings]`:

Setting	Meant for
`FrontendJavaScriptList[]`	public designs
`BackendJavaScriptList[]`	every page of the administration designs (the kernel lists `tools/ezjsselection.js`, `ezadmin_menubar.js` ...); only scripts that run without jQuery, since admin4 and admin4l load jQuery only where a template requests it
`ClassicBackendJavaScriptList[]`	every page of the older administration designs (`admin`, `admin3`), after jQuery and `BackendJavaScriptList[]`; admin4 does not read it
`JavaScriptList[]`	older list for both (DEPRECATED in favor of the two above)

An admin4 template whose script needs jQuery or an Exponential UI module requests it itself, outside every cache-block: `{ezscript_require( array( 'ezjsc::jquery', 'myadmin.js' ) )}` (see admin4 and admin4l without jQuery).

As for stylesheets, a list only matters if your pagelayout reads it:

Template

```
{ezscript_load( array( 'mytheme.js', ezini( 'JavaScriptSettings', 'FrontendJavaScriptList', 'desi
gn.ini' ) ) )}
```

Passing settings to a script

Do not print values into an inline script. Use a JSON data block, which the browser does not execute and a Content Security Policy does not block, or `data-*` attributes:

Template

```
<script type="application/json" id="mytheme-config">{json_encode( hash(
    'searchUrl', '/content/search'|ezurl( 'no' ),
    'closeLabel', 'Close'|i18n( 'design/mytheme' ) ) )}</script>
```

JavaScript

```
// javascript/mytheme.js
const config = JSON.parse(document.getElementById('mytheme-config').textContent);
```

`json_encode` is an ezjscore operator. The media design passes its video consent texts this way.

Rules

- No inline `<script>` with code and no `onclick=""` attributes (No inline JavaScript).
- No `?v=` on script URLs; the packer's file names change with the content.
- Prefer small, page-specific files with `ezscript_require` to one large bundle on every page.
- Test with the browser console open: one error stops a whole packed bundle.

Related pages

- Loading stylesheets
- jQuery in designs
- The packer
- 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.