Syntax-Highlighter-Libraries für JS: Überblick & Empfehlung □□

Für die direkte CDN-Einbindung in eine Website (ohne Build-Tools wie Webpack/Vite) kommen praktisch drei relevante Kandidaten in Frage. Hier eine Einordnung – speziell mit Blick auf deinen Workflow mit **Bricks Builder** und **Divi**.

Die wichtigsten Libraries

Library	Prinzip	CDN-tauglich?	Besonderheit
Prism.js	Client-seitig, statische Grammatiken	□ Sehr gut	Riesiges Theme-Angebot, Plugin-System *(Zeilennummern, Copy-Button, Diff-Highlighting)*
highlight.js	Client-seitig, automatische Spracherkennung	□ Sehr gut	Erkennt Sprache oft automatisch, wenn du `class="language-xxx"` vergisst
Shiki	Nutzt die *TextMate*-Grammatiken von VS Code	⚠ Bedingt	Optisch das genaueste Highlighting, aber eigentlich für *Build-Time* /SSR gedacht – für reines CDN-Einbinden overkill
Rainbow.js	Client-seitig, minimalistisch	□	Kaum noch weiterentwickelt – eher meiden
Torchlight	API-basiert *(Server rendert HTML)*	⚠	Kostenpflichtig, kein "echtes" CDN-Snippet

□□ Empfehlung: Prism.js

Für deinen Anwendungsfall – **Code-Snippets per Custom-HTML-Element in Bricks oder Divi einbinden** – ist **Prism.js** die beste Wahl:

- *Extrem leichtgewichtig*: Basis-Bibliothek nur wenige KB, jede Sprache kostet nur **0,3–0,5 KB** zusätzlich.
- Riesige Auswahl an **fertigen Themes** *(z. B. Tomorrow Night, Okaidia, Solarized)* – passt gut zu deinem CSS/SCSS-Faible, da sich Themes leicht per eigenem CSS überschreiben lassen.
- **Kein Build-Prozess** nötig – reines `<script>`- und `<link>`-Tag reicht, ideal für PHP-Einbindung in WordPress ohne tiefes Verständnis der internen Abläufe.
- Sehr gute Doku, riesige Verbreitung, aktiv gepflegt.

Minimal-Setup per CDN

```
<!-- Theme -->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/prism/1.29.0/themes/prism-tomorrow.min.css">

<!-- Core + gewünschte Sprachen -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/prism/1.29.0/components/prism-core.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/prism/1.29.0/plugins/autoloader/prism-autoloader.min.js"></script>
```

Mit dem **Autoloader-Plugin** musst du dir um einzelne Sprachdateien keine Gedanken machen – Prism lädt die passende Grammatik automatisch nach, sobald es ein `<code class="language-php">` (oder `-js`, `-css` etc.) findet.

```
<pre><code class="language-php">
function hello() {
    echo "Hallo Welt";
}
</code></pre>
```

In Bricks Builder einbinden

- **Head-Skript** *(Einstellungen → Custom Code → Head)*: die `<link>`- und `<script>`-Tags dort platzieren.
- Für die eigentlichen Code-Blöcke im Content ein **Code-Element** oder ein **Custom-HTML-Element** nutzen und die `<pre><code>`-Struktur einfügen.

- Da Prism nach dem Laden automatisch beim `DOMContentLoaded` scannt, brauchst du meist **keine eigene JS-Logik** – perfekt für deine Ausgangslage bei JS/PHP.

Alternative: highlight.js

Wenn dir **automatische Spracherkennung ohne `language-xxx`-Klasse** wichtiger ist als Theme-Vielfalt, ist highlight.js die zweitbeste Option:

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/highlight.js/11.9.0/styles/github-dark.min.css">
<script
src="https://cdnjs.cloudflare.com/ajax/libs/highlight.js/11.9.0/highlight.min.js"></script>
<script>hljs.highlightAll();</script>
```

Nachteil gegenüber Prism: etwas *größere Grundlast* und weniger flexibles Plugin-Ökosystem *(z. B. für Zeilennummern braucht man Zusatz-Snippets)*.

Warum nicht Shiki?

Shiki liefert zwar das **optisch genaueste** Ergebnis *(gleiche Engine wie VS Code)*, ist aber primär für **Server-Side-Rendering oder Build-Time-Highlighting** konzipiert. Für ein einfaches CDN-Skript auf einer WordPress-Seite ist der Overhead *(WASM-basierte Oniguruma-Engine, größere Bundle-Größe)* unnötig komplex – das würde dir ohne tiefere JS-Kenntnisse mehr Kopfschmerzen als Nutzen bringen.

Kurz gesagt: □□ *Prism.js* ist für dein Setup der Sweet Spot aus **einfacher CDN-Einbindung**, **geringer Ladezeit** und **hoher visueller Anpassbarkeit** über CSS – ganz ohne dass du dich mit WordPress-internen PHP-Prozessen auseinandersetzen musst.

Revision #1
Created 2026-07-22 08:11:11 UTC by art10m
Updated 2026-07-22 08:11:22 UTC by art10m