# wptexturize() in BricksBuilder & Polylang

## Kurze Antwort

**Teilweise.** Es kommt darauf an, welches Bricks-Element du verwendest:

| Element | wptexturize() aktiv? |
|---|---|
| **Rich Text Element** | ✅ Ja – läuft durch die WP-Content-Pipeline (inkl. `wpautop`, `wptexturize`) |
| **Basic Text Element** | ❌ Nein – gibt den Text "roh" aus, ohne die klassischen Content-Filter |
| **Heading, Text Link, etc.** | ❌ In der Regel nein, gleiche Ursache wie beim Basic Text |
| **WP-Query/Post-Content via `{content}`-Dynamic-Data** | ✅ Ja, wenn wirklich `the_content` genutzt wird |

Das wurde auch im offiziellen Bricks-Forum bestätigt: Nutzer beschwerten sich, dass Anführungszeichen im *Rich Text Element* automatisch "prettified" (→ typografische Anführungszeichen) werden, während das *Basic Text Element* unverändert bleibt. Die Bricks-Entwickler bestätigten: Das ist WP-Core-Verhalten (`wptexturize`), kein Bug in Bricks selbst.

Wer es abschalten will (nicht was du willst, aber zur Einordnung):

```
add_filter( 'run_wptexturize', '__return_false' );
```

## Und wie ist es mit mehreren Sprachen (Polylang)?

`wptexturize()` holt sich die passenden „intelligenten" Anführungszeichen (z. B. „…" im Deutschen vs. “…” im Englischen vs. «…» im Französischen) aus **übersetzten Strings** im WordPress-Core (`wp-includes/formatting.php`), die über die geladene **.mo/.po-Datei der aktuell aktiven Locale** kommen – nicht statisch für „Deutsch" hartcodiert.

Das heißt konkret:

1. **Wenn Polylang die Locale für die jeweils angezeigte Sprachversion korrekt setzt** (was es standardmäßig via `locale`-Filter frühzeitig im Request tut, bevor Content gerendert wird), dann bekommt `wptexturize()` beim Rendern der Seite auch die **richtige Locale** und liefert automatisch die passenden landestypischen Anführungszeichen – **aber nur für Elemente, die überhaupt durch `wptexturize()` laufen** (siehe Tabelle oben, also praktisch nur das Rich Text Element bzw. echten `the_content`-Output).
2. **Wichtige Einschränkung:** Die übersetzten Zeichen-Arrays innerhalb von `wptexturize()` werden **einmal pro Request statisch gecached** (`static` Variablen in der Funktion). Das ist in der Regel unproblematisch, weil eine Frontend-Anfrage ja nur eine Sprache/Locale betrifft – es sei denn du hast Setups, die mitten im Request die Sprache umschalten (z. B. bestimmte Multisite- oder Caching-Konstellationen), dann kann es zu falschen/veralteten Zeichen kommen.
3. **Basic Text / Heading Elemente bekommen NIE lokalisierte Anführungszeichen**, unabhängig von Polylang – weil `wptexturize()` dort gar nicht aufgerufen wird. Wenn du in diesen Feldern `"Text"` schreibst, bleibt es ein gerades Anführungszeichen, egal welche Sprache aktiv ist.

# Praktische Empfehlung

- **Willst du korrekt lokalisierte Anführungszeichen** → nutze für Textinhalte mit Anführungszeichen das **Rich Text Element** statt Basic Text/Heading, dann greift `wptexturize()` + Polylang-Locale zusammen automatisch richtig.
- **Willst du es für Basic Text/Heading auch erzwingen**, kannst du selbst einen Filter auf die Bricks-Render-Ausgabe legen, z. B.:

```
add_filter( 'bricks/text-basic/render_content', function( $content ) {
    return wptexturize( $content );
}, 10, 1 );
```

(Hook-Name kann je Bricks-Version leicht variieren – ggf. `bricks/heading/output` oder `bricks/text/content` prüfen, da Bricks nicht für jedes Element denselben Filternamen bereitstellt.)

- **Alternative:** Direkt schon in der ACF/Text-Eingabe die richtigen Unicode-Zeichen „ “ ” , ' ' verwenden, statt dich auf `wptexturize()` zu verlassen – dann ist es unabhängig vom Element und von Polylang-Locale-Timing zuverlässig korrekt, gerade bei sprachübergreifenden Multilingual-Setups.

**Fazit:** `wptexturize()` funktioniert in Bricks nur dort, wo Bricks tatsächlich den klassischen WordPress-Content-Filter-Stack nutzt (Rich Text Element). Mit Polylang bekommst du dort auch korrekt lokalisierte Anführungszeichen, sofern die Locale beim Rendern richtig gesetzt ist. Für alle anderen Bricks-Elemente (Basic Text, Heading etc.) greift die Funktion nicht – dort musst du selbst nachhelfen oder gleich die richtigen Zeichen eintippen.

Revision #1
Created 2026-07-13 17:42:14 UTC by art10m
Updated 2026-07-13 17:42:27 UTC by art10m