GUÍA PASO A PASO

Theme app extensions de Shopify

La extensión completa de ejemplo, con un app block, un app embed, sus assets y sus traducciones; cómo validarla y publicarla con Shopify CLI; cómo dejar los script tags sin cortar el servicio, y la checklist para limpiar el código que una app dejó en el tema.

PARA	Quien construye apps de Shopify que agregan contenido o scripts a la tienda
PROBADO CON	Shopify CLI 3.88.1 · shopify theme check · Node 22.22.3 · sin conectar a ninguna tienda
ACTUALIZADA	19 de noviembre de 2026
DOCUMENTACIÓN	shopify.dev/docs/apps/build/online-store/theme-app-extensions
ARTÍCULO	ecomlabs.dev/g/extension
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Qué vas a construir
Paso 01	El app block
Paso 02	El app embed
Paso 03	Traducciones
Paso 04	Validar, publicar y dejar los script tags
Al terminar	Checklist de limpieza del tema

ANTES DE EMPEZAR

Qué vas a construir

Una extensión con dos bloques: **stock-badge**, un app block que avisa en la ficha de producto cuando quedan pocas unidades, y **help-bubble**, un app embed con un enlace de ayuda flotante y su propio JavaScript. Necesitas una app creada con Shopify CLI 3.0 o superior, una tienda de desarrollo y un tema Online Store 2.0 para probar el app block. Desde la carpeta de la app:

```
shopify app generate extension --template theme_app_extension --name stock-badge
```

Borra el bloque de estrellas de ejemplo (necesita un metafield `demo.avg_rating`). `shopify.extension.toml` queda así:

```
name = "stock-badge"
type = "theme"
```

PASO 01

El app block

`blocks/stock-badge.liquid`: `"target": "content"` lo hace app block, `autofill` toma el producto de la sección y la condición sobre `inventory_management` evita leer unidades vendidas cuando no se rastrea el inventario.

```
{%- liquid
  assign badge_product = block.settings.product
  assign variant = badge_product.selected_or_first_available_variant
  assign quantity = variant.inventory_quantity
-%}
{%- if variant.inventory_management == 'shopify' and quantity > 0 and quantity <= block.settings.threshold -%}
  <p class="stock-badge" style="--stock-badge-color: {{ block.settings.color }}">
    {{ 'stock_badge.low_stock' | t: quantity: quantity }}
  </p>
{%- endif -%}

{% schema %}
{
  "name": "t:blocks.stock_badge.name",
  "target": "content",
  "stylesheet": "stock-badge.css",
  "enabled_on": { "templates": ["product"] },
  "settings": [
    { "type": "product", "id": "product", "label": "t:blocks.stock_badge.product", "autofill": true },
    { "type": "range", "id": "threshold", "label": "t:blocks.stock_badge.threshold", "min": 1, "max": 20,
"step": 1, "default": 5 },
    { "type": "color", "id": "color", "label": "t:blocks.stock_badge.color", "default": "#b42318" }
```

```
  ]
}
{% endschema %}
```

`assets/stock-badge.css`:

```
.stock-badge { margin: 0.5rem 0; color: var(--stock-badge-color); font-weight: 600; }
```

PASO 02

El app embed

`blocks/help-bubble.liquid`: `"target": "body"` lo hace app embed y `enabled_on` carga el script solo en producto y colección. Sin enlace configurado no pinta nada.

```
{%- if block.settings.link != blank -%}
  <aside class="help-bubble" data-help-bubble hidden>
    <a href="{{ block.settings.link }}">{{ block.settings.label | escape }}</a>
    <button type="button" data-help-close aria-label="{{ 'help_bubble.close' | t }}">&times;</button>
  </aside>
{%- endif -%}

{% schema %}
{
  "name": "t:blocks.help_bubble.name",
  "target": "body",
  "javascript": "help-bubble.js",
  "stylesheet": "help-bubble.css",
  "enabled_on": { "templates": ["product", "collection"] },
  "settings": [
    { "type": "url", "id": "link", "label": "t:blocks.help_bubble.link" },
    { "type": "text", "id": "label", "label": "t:blocks.help_bubble.label", "default": "Help" }
  ]
}
{% endschema %}
```

`assets/help-bubble.js` espera al DOM, muestra la burbuja y recuerda en la sesión que el cliente la cerró, aunque el navegador bloquee `sessionStorage`:

```
(() => {
  const KEY = "help-bubble-closed";
  const start = () => {
    const box = document.querySelector("[data-help-bubble]");
    if (!box) return;
    try {
      if (sessionStorage.getItem(KEY)) return;
    } catch {}
    box.hidden = false;
```

```
    box.querySelector("[data-help-close]").addEventListener("click", () => {
      box.hidden = true;
      try {
        sessionStorage.setItem(KEY, "1");
      } catch {}
    });
  };
  if (document.readyState === "loading") document.addEventListener("DOMContentLoaded", start);
  else start();
})();
```

`assets/help-bubble.css`:

```
.help-bubble { position: fixed; right: 1rem; bottom: 1rem; z-index: 10; display: flex; gap: 0.5rem; align-items: center; padding: 0.5rem 0.75rem; border-radius: 999px; background: #fff; box-shadow: 0 2px 8px rgb(0 0 0 / 0.15); }
.help-bubble[hidden] { display: none; }
.help-bubble button { min-width: 44px; min-height: 44px; border: 0; background: none; font-size: 1.25rem; cursor: pointer; }
```

PASO 03

Traducciones

Los textos de la tienda van en `locales/en.default.json` y `locales/es.json`; las etiquetas del editor, en `locales/en.default.schema.json` y `locales/es.schema.json`, con las mismas claves. Los dos en español:

```
{
  "stock_badge": { "low_stock": "Quedan {{ quantity }} en existencia" },
  "help_bubble": { "close": "Cerrar" }
}

{
  "blocks": {
    "stock_badge": { "name": "Aviso de existencias", "product": "Producto", "threshold": "Mostrar con estas o menos", "color": "Color del texto" },
    "help_bubble": { "name": "Burbuja de ayuda", "link": "Página de ayuda", "label": "Texto" }
  }
}
```

Sin comentarios ni comas finales: las extensiones no aceptan JSON con comentarios.

PASO 04

Validar, publicar y dejar los script tags

1. `shopify theme check --path extensions/stock-badge` debe decir «no offenses found». El `target` revísalo a mano: un valor inventado no lo marca.

2. `shopify app dev` abre la vista previa (solo en Chrome); `shopify app deploy` crea y publica una versión. Límites: 10 MB, 30 bloques, 100 KB de Liquid.
3. Manda al comerciante al editor con el embed activado: `https://<tienda>.myshopify.com/admin/themes/current/editor?context=apps&template=product&activateAppId={client_id}/help-bubble`.
4. Confirma que está activo con `app.extensions()` o en `config/settings_data.json`, y solo entonces borra los script tags. Desde el 1 de octubre de 2026 no se pueden crear; el 1 de marzo de 2027 dejan de cargarse.

```
query AppScriptTags {
  scriptTags(first: 50) {
    nodes { id src displayScope createdAt }
  }
}

mutation DeleteScriptTag($id: ID!) {
  scriptTagDelete(id: $id) {
    deletedScriptTagId
    userErrors { field message }
  }
}
```

AL TERMINAR

Checklist de limpieza del tema

Para el código que la versión vieja de la app dejó en el tema. Trabaja en una copia.

- [ ] Duplicaste el tema publicado y bajaste la copia con `shopify theme pull`.
- [ ] Buscaste la app y el dominio de sus scripts: `grep -rn "app-badge\|tu-app.com" layout sections snippets templates`.
- [ ] Quitaste cada `<script>` pegado en `layout/theme.liquid`, sin dejar líneas en blanco de más.
- [ ] Quitaste cada snippet de la app junto con su `{% render %}`; nunca uno sin el otro.
- [ ] `shopify theme check` pasa en la copia limpia.
- [ ] El código fuente de la página ya no menciona el dominio de la app y el script carga una sola vez.
- [ ] La consulta `scriptTags` devuelve una lista vacía y el embed está activo en el tema que vas a publicar.
- [ ] La pantalla de bienvenida de la app trae estas instrucciones y el enlace directo.