

Coding & Webdev

- [Mastodon Comments in Bookstack](#)

Mastodon Comments in Bookstack

There are several people online who have written JS scripts to turn a Mastodon thread into a comment thread on a website. The ones I came across did not directly work with Bookstack however. This is a quick guide for doing the same with Bookstack, although there are surely better ways, this was very quick to setup.

“ Want to see what it looks like first? Just scroll to the bottom, it's on this page!

Based on this Repo: <https://github.com/dpecos/mastodon-comments>

In Bookstack go to **Settings > Customization**

Add the following into the **Custom HTML Head Content** section:

Custom HTML Head Content

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/dompurify/2.4.1/purify.min.js"
integrity="sha512-
uH0KtSfJWScGmyyFr202+efpDx2nhwHU2v7MVeptzZoiC7bdF6Ny/CmZhN2AwIK1oCFiVQ5DA/L9FSzyPNu6Q=="
crossorigin="anonymous" referrerpolicy="no-referrer"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">

<!-- Convert Mastodon Toot link into comment form: https://github.com/dpecos/mastodon-
comments -->
<!-- This is needed b/c Bookstack doesn't allow custom tags like "mastodon-comments"
directly in the page -->
<!-- Recommended link format: <a class="mastodon-link"
href="https://hostux.social/@JaggedJax/TOOT_ID" target="_blank" rel="noopener">Leave a
Comment</a> -->
<script>
  const styles = `
:root {
```

```
--font-color: #5d686f;
--font-size: 1.0rem;

--block-border-width: 1px;
--block-border-radius: 3px;
--block-border-color: #ededf0;
--block-background-color: #f7f8f8;

--comment-indent: 40px;
}

mastodon-comments {
  font-size: var(--font-size);
}

p {
  margin: 0 0 1rem 0;
}

#mastodon-stats {
  text-align: center !important;
  font-size: calc(var(--font-size) * 1.5);
}

#mastodon-title {
  font-size: calc(var(--font-size) * 1.5);
  font-weight: bold !important;
}

#mastodon-comments-list {
  margin: 0 auto;
  padding: 0;
}

#mastodon-comments-list ul {
  padding-left: var(--comment-indent);
}

#mastodon-comments-list li {
```

```
list-style: none;
}

.mastodon-comment {
  background-color: var(--block-background-color);
  border-radius: var(--block-border-radius);
  border: var(--block-border-width) var(--block-border-color) solid;
  padding: 15px;
  margin-bottom: 1.5rem;
  display: flex;
  flex-direction: column;
  color: var(--font-color);
}

.mastodon-comment p {
  margin-bottom: 0px;
}

.mastodon-comment .author {
  padding-top: 0;
  display: flex;
}

.mastodon-comment .author a {
  text-decoration: none;
}

.mastodon-comment .author .avatar img {
  margin-right: 1rem;
  min-width: 60px;
  border-radius: 5px;
}

.mastodon-comment .details {
  margin-left: 35px;
}

.mastodon-comment .author .details {
  display: flex;
```

```
flex-direction: column;
line-height: 1.2em;
}

.mastodon-comment .author .details .name {
  font-weight: bold;
}

.mastodon-comment .author .details .user {
  color: #5d686f;
  font-size: medium;
}

.mastodon-comment .author .date {
  margin-left: auto;
  font-size: small;
}

.mastodon-comment .content {
  margin: 15px 0;
  line-height: 1.5em;
}

.mastodon-comment .author .details a,
.mastodon-comment .content p {
  margin-bottom: 10px;
}

.mastodon-comment .attachments {
  margin: 0px 10px;
}

.mastodon-comment .attachments > * {
  margin: 0px 10px;
}

.mastodon-comment .attachments img {
  max-width: 100%;
}
```

```
.mastodon-comment .status > div, #mastodon-stats > div {
  display: inline-block;
  margin-right: 15px;
}

.mastodon-comment .status a, #mastodon-stats a {
  color: #5d686f;
  text-decoration: none;
}

.mastodon-comment .status .replies.active a, #mastodon-stats .replies.active a {
  color: #003eaa;
}

.mastodon-comment .status .reblogs.active a, #mastodon-stats .reblogs.active a {
  color: #8c8dff;
}

.mastodon-comment .status .favourites.active a, #mastodon-stats .favourites.active a {
  color: #ca8f04;
}
`;

class MastodonComments extends HTMLElement {
  constructor() {
    super();

    this.host = this.getAttribute("host");
    this.user = this.getAttribute("user");
    this.tootId = this.getAttribute("tootId");

    this.commentsLoaded = false;

    const styleElem = document.createElement("style");
    styleElem.innerHTML = styles;
    document.head.appendChild(styleElem);
  }
}
```

```

connectedCallback() {
  this.innerHTML = `
    <div id="mastodon-stats"></div>
    <div id="mastodon-title">Comments</div>

    <noscript>
      <div id="error">
        Please enable JavaScript to view the comments powered by the Fediverse.
      </div>
    </noscript>

    <p>You can use your Fediverse (i.e. Mastodon, among many others) account to reply to
this <a class="link"
      href="https://${this.host}/${this.user}/${this.tootId}" rel="ugc">post</a>.
    </p>
    <ul id="mastodon-comments-list"></ul>
  `;

  const comments = document.getElementById("mastodon-comments-list");
  const rootStyle = this.getAttribute("style");
  if (rootStyle) {
    comments.setAttribute("style", rootStyle);
  }
  this.respondToVisibility(comments, this.loadComments.bind(this));
}

escapeHtml(unsafe) {
  return (unsafe || "")
    .replace(/&/g, "&amp;")
    .replace(/</g, "&lt;")
    .replace(/>/g, "&gt;")
    .replace(/"/g, "&quot;")
    .replace(/'/g, "&#039;");
}

toot_active(toot, what) {
  var count = toot[what + "_count"];
  return count > 0 ? "active" : "";
}

```

```

toot_count(toot, what) {
    var count = toot[what + "_count"];
    return count > 0 ? count : "";
}

toot_stats(toot) {
    return `
        <div class="replies ${this.toot_active(toot, "replies")}">
            <a href="${
                toot.url
            }" rel="ugc nofollow"><i class="fa fa-reply fa-fw"></i>${this.toot_count(
                toot,
                "replies",
            )}</a>
        </div>
        <div class="reblogs ${this.toot_active(toot, "reblogs")}">
            <a href="${
                toot.url
            }/reblogs" rel="nofollow"><i class="fa fa-retweet fa-fw"></i>${this.toot_count(
                toot,
                "reblogs",
            )}</a>
        </div>
        <div class="favourites ${this.toot_active(toot, "favourites")}">
            <a href="${
                toot.url
            }/favourites" rel="nofollow"><i class="fa fa-star fa-fw"></i>${this.toot_count(
                toot,
                "favourites",
            )}</a>
        </div>
    `;
}

user_account(account) {
    var result = `@${account.acct}`;
    if (account.acct.indexOf("@") === -1) {
        var domain = new URL(account.url);
    }
}

```

```

    result += `@${domain.hostname}`;
  }
  return result;
}

render_toots(toots, in_reply_to) {
  var tootsToRender = toots
    .filter((toot) => toot.in_reply_to_id === in_reply_to)
    .sort((a, b) => a.created_at.localeCompare(b.created_at));
  tootsToRender.forEach((toot) => this.render_toot(toots, toot));
}

render_toot(toots, toot) {
  toot.account.display_name = this.escapeHtml(toot.account.display_name);
  toot.account.emojis.forEach((emoji) => {
    toot.account.display_name = toot.account.display_name.replace(
      `:${emoji.shortcode}:`,
      `![Emoji ${
        emoji.shortcode
      }](${this.escapeHtml(emoji.static_url)})

```

```

    
  </div>
  <div class="details">
    <a class="name" href="${toot.account.url}" rel="nofollow">${
      toot.account.display_name
    }</a>
    <a class="user" href="${
      toot.account.url
    }" rel="nofollow">${this.user_account(toot.account)}</a>
  </div>
  <a class="date" href="${
    toot.url
  }" rel="nofollow">
    <time datetime="${toot.created_at}">
      ${formatDate(toot.created_at)}${toot.edited_at ? "*" : ""}
    </time>
  </a>
</div>
<div class="content">${toot.content}</div>
<div class="attachments">
  ${toot.media_attachments
    .map((attachment) => {
      if (attachment.type === "image") {
        return `<a href="${attachment.url}" rel="ugc nofollow"></a>`;
      } else if (attachment.type === "video") {
        return `<video controls preload="none"><source src="${attachment.url}"
type="${attachment.mime_type}"></video>`;
      } else if (attachment.type === "gifv") {
        return `<video autoplay loop muted playsinline><source
src="${attachment.url}" type="${attachment.mime_type}"></video>`;
      } else if (attachment.type === "audio") {
        return `<audio controls><source src="${attachment.url}"
type="${attachment.mime_type}"></audio>`;
      } else {

```

```

        return `

```

```

document.getElementById("mastodon-comments-list").innerHTML =
  "Loading comments from the Fediverse...";

let _this = this;

fetch("https://" + this.host + "/api/v1/statuses/" + this.tootId)
  .then((response) => response.json())
  .then((toot) => {
    document.getElementById("mastodon-stats").innerHTML =
      this.toot_stats(toot);
  });

fetch(
  "https://" + this.host + "/api/v1/statuses/" + this.tootId + "/context",
)
  .then((response) => response.json())
  .then((data) => {
    if (
      data["descendants"] &&
      Array.isArray(data["descendants"]) &&
      data["descendants"].length > 0
    ) {
      document.getElementById("mastodon-comments-list").innerHTML = "";
      _this.render_toots(data["descendants"], _this.tootId, 0);
    } else {
      document.getElementById("mastodon-comments-list").innerHTML =
        "<p>No comments found</p>";
    }

    _this.commentsLoaded = true;
  });
}

respondToVisibility(element, callback) {
  var options = {
    root: null,
  };
};

```

```
var observer = new IntersectionObserver((entries, observer) => {
  entries.forEach((entry) => {
    if (entry.intersectionRatio > 0) {
      callback();
    }
  });
}, options);

observer.observe(element);
}
}
```

```
function convertMastodonLink() {
  return new Promise((resolve, reject) => {
    // Find the anchor tag with class "mastodon-link"
    const mastodonLink = document.querySelector('a.mastodon-link');

    if (!mastodonLink) {
      reject('No anchor tag with class "mastodon-link" found');
    }

    // Get the href attribute
    const href = mastodonLink.href;

    if (!href) {
      reject('Mastodon link has no href attribute');
    }

    try {
      // Parse the URL to extract host and tootId
      const url = new URL(href);
      const host = url.hostname;

      // Extract tootId from pathname (assuming format like /@user/123456789)
      const pathParts = url.pathname.split('/');
      const tootId = pathParts[pathParts.length - 1];

      // Extract username (assuming format like /@username/tootId)
```

```

let user = '';
const userIndex = pathParts.findIndex(part => part.startsWith('@'));
if (userIndex !== -1) {
    user = pathParts[userIndex].substring(1); // Remove the @ symbol
}

// Validate that we have the required data
if (!host || !user || !tootId || isNaN(tootId)) {
    reject('Could not parse host, user, or tootId from URL:', href);
}

// Create the new mastodon-comments element
const mastodonComments = document.createElement('mastodon-comments');
mastodonComments.setAttribute('host', host);
mastodonComments.setAttribute('user', user);
mastodonComments.setAttribute('tootId', tootId);
mastodonComments.setAttribute('style', 'width: 840px');

// Replace the original anchor tag with the new element
mastodonLink.parentNode.replaceChild(mastodonComments, mastodonLink);

console.log(`Successfully converted mastodon link to mastodon-comments
element`);
console.log(`Host: ${host}, User: ${user}, TootId: ${tootId}`);

resolve();

} catch (error) {
    console.error('Error parsing Mastodon URL:', error);
    reject(error);
}
});
}

function loadMastodonScript() {
    return new Promise((resolve, reject) => {
        customElements.define("mastodon-comments", MastodonComments);
    });
}

```

```

// Execute conversion first, then load MastodonComments
document.addEventListener('DOMContentLoaded', async () => {
  try {
    await convertMastodonLink();
    console.log('Mastodon conversion complete, loading MastodonComments');
    await loadMastodonScript();
    console.log('MastodonComments loaded successfully');
  } catch (error) {
    console.warn('Unable to load MastodonComments:', error);
  }
});
</script>

```

Now on the page you want to add comments, add the following **<a>** tag where you want the comments to load. Fill in the link to your post on your home server:

```

<a class="mastodon-link" href="https://hostux.social/@JaggedJax/T00T_ID" target="_blank"
rel="noopener">Leave a Comment</a>

```

Note: You need to click the **<>** icon in the Bookstack WYSIWYG editor to paste in this link there. The WYSIWYG does not let you set a custom class that we're using to convert this link. Users without Javascript will just see the link to you post.

How It Works:

The mastodon-comments script is looking for a custom HTML tag which Bookstack doesn't allow. This script instead looks for an **<a>** tag with the class mastodon-link, and converts that into the necessary mastodon-comments tag. This gives the advantage of working well without JS enabled as well.

If no valid **<a>** tag is found then the mastodon-comments script is not triggered.

Improvements:

- The JS here could be cleaned up a lot. Much of the CSS is unused, and the use of a Promise is overkill here. That's leftover from my initial attempts and is not necessary.
- Font Awesome is being used to show icons. It would be nice to remove that
- The mastodon-comments script this is taken from uses DOMPurify to help prevent XSS. That's a great thing, but we could just embed the code locally to avoid the callout.
- If you make your own improvements please comment and let me know so I can make them too!

[Leave a Comment](#)