Neue Seite

```
<style>
#bs-tree-menu {
    position: fixed;
    left: 0;
    top: 70px;
    bottom: 0;
    width: 320px;
    overflow-y: auto;
    background: #fff;
    border-right: 1px solid #ddd;
    z-index: 9999;
    padding: 12px;
    font-size: 14px;
}

#bs-tree-menu ul {
    list-style: none;
    margin: 0;
    padding-left: 18px;
}

#bs-tree-menu > ul {
    padding-left: 0;
}

#bs-tree-menu li {
    margin: 2px 0;
}

#bs-tree-menu .tree-row {
    display: flex;
    align-items: center;
    gap: 5px;
    min-height: 28px;
}
```

```
#bs-tree-menu button {
    width: 22px;
    height: 22px;
    padding: 0;
    border: 0;
    background: none;
    cursor: pointer;
    font-size: 15px;
}

#bs-tree-menu a {
    color: #333;
    text-decoration: none;
}

#bs-tree-menu a:hover {
    text-decoration: underline;
}

#bs-tree-menu .children {
    display: none;
}

#bs-tree-menu .open > .children {
    display: block;
}

#bs-tree-menu .page {
    padding-left: 27px;
}

#bs-tree-menu .error {
    color: #b00020;
    padding: 10px;
}
</style>

<div id="bs-tree-menu">
    <strong>CONTENT</strong>
```

```
    <div id="bs-tree-status">Loading…</div>
    <ul id="bs-tree"></ul>
</div>

<script>
(async function () {
    const root = document.getElementById('bs-tree');
    const status = document.getElementById('bs-tree-status');

    const api = async (url) => {
        const response = await fetch(url, {
            credentials: 'same-origin',
            headers: {
                'Accept': 'application/json'
            }
        });

        if (!response.ok) {
            throw new Error(url + ' → HTTP ' + response.status);
        }

        return response.json();
    };

    const all = async (endpoint) => {
        let page = 1;
        let result = [];

        while (true) {
            const data = await api(
                '/api/' + endpoint + '?count=500&page=' + page
            );

            const rows = data.data || [];
            result = result.concat(rows);

            if (rows.length < 500) {
                break;
            }
```

```
        page++;
    }

    return result;
};

const esc = (value) => {
    const div = document.createElement('div');
    div.textContent = value || '';
    return div.innerHTML;
};

const node = (name, url, children, expandable) => {
    const li = document.createElement('li');

    const row = document.createElement('div');
    row.className = 'tree-row';

    if (expandable) {
        const button = document.createElement('button');
        button.textContent = '+';

        button.addEventListener('click', () => {
            li.classList.toggle('open');
            button.textContent = li.classList.contains('open') ? '−' : '+';
        });

        row.appendChild(button);
    } else {
        const spacer = document.createElement('span');
        spacer.style.width = '22px';
        row.appendChild(spacer);
    }

    const link = document.createElement('a');
    link.href = url;
    link.innerHTML = esc(name);

    row.appendChild(link);
    li.appendChild(row);
```

```
    if (children) {
        children.className = 'children';
        li.appendChild(children);
    }

    return li;
};

const list = () => {
    const ul = document.createElement('ul');
    return ul;
};

try {
    status.remove();

    const shelves = await all('shelves');
    const books = await all('books');
    const chapters = await all('chapters');
    const pages = await all('pages');

    const booksByShelf = {};
    const chaptersByBook = {};
    const pagesByBook = {};

    books.forEach(book => {
        const shelfId = book.shelf_id;

        if (!booksByShelf[shelfId]) {
            booksByShelf[shelfId] = [];
        }

        booksByShelf[shelfId].push(book);
    });

    chapters.forEach(chapter => {
        if (!chaptersByBook[chapter.book_id]) {
            chaptersByBook[chapter.book_id] = [];
        }
```

```
    chaptersByBook[chapter.book_id].push(chapter);
});

pages.forEach(page => {
    if (!pagesByBook[page.book_id]) {
        pagesByBook[page.book_id] = [];
    }

    pagesByBook[page.book_id].push(page);
});

shelves.forEach(shelf => {
    const shelfChildren = list();

    const shelfBooks = booksByShelf[shelf.id] || [];

    shelfBooks.sort((a, b) =>
        (a.priority || 0) - (b.priority || 0)
    );

    shelfBooks.forEach(book => {
        const bookChildren = list();

        const bookChapters =
            chaptersByBook[book.id] || [];

        bookChapters.sort((a, b) =>
            (a.priority || 0) - (b.priority || 0)
        );

        bookChapters.forEach(chapter => {
            const chapterChildren = list();

            const chapterPages =
                pages.filter(page =>
                    page.chapter_id === chapter.id
                );

            chapterPages.sort((a, b) =>
```

```
            (a.priority || 0) - (b.priority || 0)
        );

        chapterPages.forEach(page => {
            chapterChildren.appendChild(
                node(
                    page.name,
                    '/books/' +
                    book.slug +
                    '/chapter/' +
                    chapter.slug +
                    '/page/' +
                    page.slug,
                    null,
                    false
                )
            );
        });

        bookChildren.appendChild(
            node(
                chapter.name,
                '/books/' +
                book.slug +
                '/chapter/' +
                chapter.slug,
                chapterChildren,
                chapterPages.length > 0
            )
        );
    });

    const bookPages =
        pagesByBook[book.id] || [];

    bookPages
        .filter(page => !page.chapter_id)
        .sort((a, b) =>
            (a.priority || 0) - (b.priority || 0)
        )
```

```
                .forEach(page => {
                    bookChildren.appendChild(
                        node(
                            page.name,
                            '/books/' +
                            book.slug +
                            '/page/' +
                            page.slug,
                            null,
                            false
                        )
                    );
                });

            const expandable =
                bookChildren.children.length > 0;

            shelfChildren.appendChild(
                node(
                    book.name,
                    '/books/' + book.slug,
                    bookChildren,
                    expandable
                )
            );
        });

        root.appendChild(
            node(
                shelf.name,
                '/shelves/' + shelf.slug,
                shelfChildren,
                shelfBooks.length > 0
            )
        );
    });

} catch (error) {
    console.error(error);
```

```
        const errorBox = document.createElement('div');
        errorBox.className = 'error';
        errorBox.textContent =
            'BookStack structure could not be loaded: ' +
            error.message;

        root.appendChild(errorBox);
    }
})();
</script>
```

CONTENT

Loading...

Revision #2
Created 2026-09-23 23:16:22 UTC by Admin
Updated 2026-09-23 23:20:25 UTC by Admin