8 层功力的 remnote 制卡 并多多一键制卡更新，支持列表项制卡，增加一键移除

作者：zxhd86
原文链接：https://ld246.com/article/1677506141805
来源网站：链滴

需要了解是什么参见并夕夕版 文档树制卡 v0.5 支持直接获取路径和分屏，演示见下：

列表项制卡的符号参考了remnote，也就是》》和>>。要是有什么容易打出来也不大可能有其他用、而且好看的符号也可以提，当然更欢迎自行添加。

另外，由于接口的限制，暂时没办法移除使用快捷键添加的卡片——因为获取不到ALL卡包。

喜欢的话留个言，也让我知道除我以外还有人用。

```
// 一键移除卡片 v0.1
// 请求函数
// 使用匿名函数，防止污染空间
(() => {

    function request(url, data = null, method = "POST") {
        return new Promise((resolve, reject) => {
            if (method.toUpperCase() == "POST") {
                fetch(url, {
                    method: "POST",
                    headers: { "Content-Type": "application/json" },
                    body: JSON.stringify(data),
                })
                    .then(
                        (data) => resolve(data.json()),
                        (error) => {
                            reject(error);
                        }
                    )
                    .catch((err) => {
                        console.error("请求失败:", err);
                    });
            }
        });
```

```
}

// 弹出提示信息
async function showMessage(msg) {
    await request("/api/notification/pushMsg", { msg, timeout: 3000 });
}


function getCurrentPage() {
    //获取当前屏幕
    const currentScreen = document.querySelector(".layout__wnd--active");
    //获取当前页面
    const currentPage = currentScreen.querySelector(
        ".fn__flex-1.protyle:not(.fn__none)"
    );
    return currentPage;
}

//传入 node节点列表 arr和数据抽取函数 fliter。
// fliter 传入一个节点，返回一个字典
// 字典格式：
// {status,data}
// status 为 ok，则代表为抽取成功，将data压入返回列表anwer，并最后返回。
function iterArr(arr, fliter) {

    if (!arr && arr.length <= 0) {
        return [];
    }

    length = arr.length;
    anwer = [];
    for (let node of arr) {
        data = fliter(node);
        if (data["status"] === "ok" && data["data"]) {
            anwer.push(data["data"]);
        }
    }
    return anwer;
}

function fliterBlockCard(node) {
    return {
        status: "ok",
        data: node.getAttribute("data-node-id")
    }
}

async function removeCards() {
    //获取全部deck
    let res = await request("/api/riff/getRiffDecks", {});

    //获取deck.id
    let customDeckIdArr = [];
    Array.from(res.data).forEach((item) => {
```

```
            customDeckIdArr.push(item.id);
        })

        if (customDeckIdArr.length === 0) {
            showMessage(`未能获取到卡包`);
            return;
        }

        // 获取需要被制成卡片的块的ID
        const currentPage = getCurrentPage()
        //已制卡块
        const cardedBlockList = currentPage.querySelectorAll(".protyle-wysiwyg [data-node-id][
ustom-riff-decks]")


        let arr = [];
        var cardedBlockIdList = iterArr(cardedBlockList, fliterBlockCard)

        arr = arr.concat(cardedBlockIdList)


        if (arr.length === 0) {
            showMessage(`未能获取到页面的卡片`);
            return;
        }

        // 去重
        arr = [... new Set(arr)];

        for (deckIndex in customDeckIdArr) {

            // 把标记所在块全部制成卡片
            let body = {
                deckID: customDeckIdArr[deckIndex],
                blockIDs: arr,
            };
            let res = await request("/api/riff/removeRiffCards", body);
            if (res.code === 0) {
                showMessage(`${res.data.name}卡包的总卡片数：${res.data.size}`);
            }
        }

    }

    const barMode = document.getElementById("barMode");
    barMode.insertAdjacentHTML(
        "beforebegin",
        '<div id="removeCards" class="toolbar__item b3-tooltips b3-tooltips__se" aria-label="
除卡片" ></div>'
    );
    const removeCardsBtn = document.getElementById("removeCards");
    removeCardsBtn.style.width = "auto";
    const removeCardsBtnIcon = `<svg xmlns="http://www.w3.org/2000/svg" height="48" vie
```

```
Box="0 96 960 960" width="48"><path d="M600 826v-60h145v60H600Zm0-368v-60h280v6
H600Zm0 184v-60h235v60H600ZM125 381H80v-60h170v-45h135v45h170v60h-45v415q0 24
18 42t-42 18H185q-24 0-42-18t-18-42V381Zm60 0v415h265V381H185Zm0 0v415-415Z"/><
svg>`;
    removeCardsBtn.innerHTML = removeCardsBtnIcon;
    removeCardsBtn.addEventListener(
        "click",
        (e) => {
            removeCards();
            e.stopPropagation();
            e.preventDefault();
        },
        true
    );

})();

// 一键制卡 v0.6
// 请求函数
// 使用匿名函数，防止污染空间
(() => {
    function request(url, data = null, method = "POST") {
        return new Promise((resolve, reject) => {
            if (method.toUpperCase() == "POST") {
                fetch(url, {
                    method: "POST",
                    headers: { "Content-Type": "application/json" },
                    body: JSON.stringify(data),
                })
                    .then(
                        (data) => resolve(data.json()),
                        (error) => {
                            reject(error);
                        }
                    )
                    .catch((err) => {
                        console.error("请求失败:", err);
                    });
            }
        });
    }

    // 弹出提示信息
    async function showMessage(msg) {
        await request("/api/notification/pushMsg", { msg, timeout: 3000 });
    }

    // 获取当前文档id
    function getFileID() {
        //获取当前页面
        const currentPage = getCurrentPage();
        //获取当前页面id
        const currentPageID = currentPage.querySelector(
            "span.protyle-breadcrumb__item--active"
```

```
    ).getAttribute("data-node-id");

    return currentPageID;
}

function getCurrentPage() {
    //获取当前屏幕
    const currentScreen = document.querySelector(".layout__wnd--active");
    //获取当前页面
    const currentPage = currentScreen.querySelector(
        ".fn__flex-1.protyle:not(.fn__none)"
    );
    return currentPage;
}

//传入 node节点列表 arr和数据抽取函数 fliter。
// fliter 传入一个节点，返回一个字典
// 字典格式：
// {status,data}
// status 为 ok，则代表为抽取成功，将data压入返回列表anwer，并最后返回。
function iterArr(arr, fliter) {

    if (!arr && arr.length <= 0) {
        return [];
    }

    length = arr.length;
    anwer = [];
    for (let node of arr) {
        data = fliter(node);
        if (data["status"] === "ok" && data["data"]) {
            anwer.push(data["data"]);
        }
    }
    return anwer;
}

// 传入列表下的段落，返回被标记段落的父节点：列表项的id
function fliterListCard(node) {
    var re = /(>>|(》》))\s?\u200b?$/;
    if (re.test(node.innerText)) {
        return {
            status: "ok",
            data: node.parentElement.getAttribute("data-node-id")
        }
    }
    return { status: "no", data: null };
}

function fliterMarkCard(node) {
    return {
        status: "ok",
        data: node.parentElement.parentElement.getAttribute("data-node-id")
    }
```

```
}

function fliterSuperBlockCard(node) {
    return {
        status: "ok",
        data: node.getAttribute("data-node-id")
    }
}

// 获取当前文档路径
async function getHpath() {
    currentPageID = getFileID()
    body = { "id": currentPageID }
    let res = await request("/api/filetree/getHPathByID", body);

    //检测返回值
    if (res.code === 0) {
        let currentPagePath = res.data;
        return currentPagePath;
    }
    return null;
}

// 添加多个卡片
async function addCards() {
    //获取全部deck
    let res = await request("/api/riff/getRiffDecks", {});

    //获取当前打开文档的路径
    focusItemPath = await getHpath()
    if (focusItemPath === null) {
        showMessage(`未能当前文档读取路径`);
        return;
    }

    //获取deck.id, 其deck.name 包含在 focusItemPath 中
    let customDeckIdArr = [];
    Array.from(res.data).forEach((item) => {
        let name = item.name;
        if (focusItemPath.search(name) != -1) {
            customDeckIdArr.push(item.id);
        }
    })

    if (customDeckIdArr.length === 0) {
        showMessage(`未能获取到：${focusItemPath} 的卡包`);
        return;
    }

    // 获取需要被制成卡片的块的ID
    const currentPage = getCurrentPage()
    //标记块
    const markList = currentPage.querySelectorAll("span[data-type='mark']")
    //超级块
```

```
        const superBlockList = currentPage.querySelectorAll("div[class='sb']")
        //列表块
        const listList = currentPage.querySelectorAll("div[class='li']:has(div[class='li'])  > div[class
= 'p']")

        let arr = [];
        var markCardList = iterArr(markList, fliterMarkCard)
        var superBlockCardList = iterArr(superBlockList, fliterSuperBlockCard)
        var listCardList = iterArr(listList, fliterListCard)
        arr = arr.concat(markCardList, superBlockCardList, listCardList)


        if (arr.length === 0) {
            showMessage(`未能获取到：${focusItemPath} 页面的卡片`);
            return;
        }

        // 去重
        arr = [... new Set(arr)];

        for (deckIndex in customDeckIdArr) {

            // 把标记所在块全部制成卡片
            let body = {
                deckID: customDeckIdArr[deckIndex],
                blockIDs: arr,
            };
            let res = await request("/api/riff/addRiffCards", body);
            if (res.code === 0) {
                showMessage(`${res.data.name}卡包的总卡片数：${res.data.size}`);
            }
        }

    }

    // 添加一个按钮
    const barMode = document.getElementById("barMode");
    barMode.insertAdjacentHTML(
        "beforebegin",
        '<div id="addCards" class="toolbar__item b3-tooltips b3-tooltips__se" aria-label="制作
片" ></div>'
    );
    const addCardsBtn = document.getElementById("addCards");
    addCardsBtn.style.width = "auto";
    const addCardsBtnIcon = `<svg t="1674130669751" class="icon" viewBox="0 0 1024 1024"
version="1.1" xmlns="http://www.w3.org/2000/svg" p-id="4526" width="200" height="200"
<path d="M928 160H96a32 32 0 0 0-32 32v672a32 32 0 0 0 32 32h832a32 32 0 0 0 32-32V1
2a32 32 0 0 0-32-32z m-32 672H128V224h768v608z" p-id="4527" fill="#9aa0a6"></path><
ath d="M230.592 448.096H544a32 32 0 1 0 0-64H230.592a32 32 0 0 0 0 64zM230.592 640.09
H544a32 32 0 1 0 0-64H230.592a32 32 0 1 0 0 64zM768 704a32 32 0 0 0 32-32V350.016a32
2 0 1 0-64 0V672a32 32 0 0 0 32 32z" p-id="4528" fill="#9aa0a6"></path></svg>`;
    addCardsBtn.innerHTML = addCardsBtnIcon;
    addCardsBtn.addEventListener(
        "click",
```

```
        (e) => {
            addCards();
            e.stopPropagation();
            e.preventDefault();
        },
        true
    );

})();
```