noi/app-04.0/resources/extensions/noi-ask/main.js ---------- code start

/\*\*

 \* NoiAsk: Batch send messages to AI Chat.

 \*

 \* This file is a modified version of the GodMode.

 \* ref: https://github.com/smol-ai/GodMode/tree/main/src/providers

 \*/

class NoiAsk {

  static sync(message) {

    const inputElement = document.querySelector('textarea');

    if (inputElement) {

      const nativeTextareaSetter = Object.getOwnPropertyDescriptor(window.HTMLTextAreaElement.prototype, 'value').set;

      nativeTextareaSetter.call(inputElement, message);

      const inputEvent = new InputEvent('input', {

        bubbles: true,

        cancelable: true,

      });

      inputElement.dispatchEvent(inputEvent);

    }

  }

  static autoFocus() {

    const inputElement = document.querySelector('textarea');

    if (inputElement) {

      inputElement.focus();

    }

  }

  static simulateUserInput(element, text) {

    const inputEvent = new InputEvent('input', {

      bubbles: true,

      cancelable: true,

    });

    element.focus();

    element.value = text;

    element.dispatchEvent(inputEvent);

  }

  static autoClick(btn) {

    btn.focus();

    btn.disabled = false;

    btn.click();

  }

}

class OpenAIAsk extends NoiAsk {

  static name = 'ChatGPT';

  static url = 'https://chat.openai.com';

  static submit() {

    const btn = document.querySelector('button[data-testid="send-button"]');

    if (btn) this.autoClick(btn);

  }

}

class PoeAsk extends NoiAsk {

  static name = 'Poe';

  static url = 'https://poe.com';

  static submit() {

    const btn = document.querySelectorAll('button[class\*="ChatMessageSendButton\_sendButton"]')[0];

    if (btn) this.autoClick(btn);

  }

}

class ClaudeAsk extends NoiAsk {

  static name = 'Claude';

  static url = 'https://claude.ai';

  static sync(message) {

    const inputElement = document.querySelector('div.ProseMirror');

    if (inputElement) {

      inputElement.focus();

      inputElement.innerHTML = '';

      document.execCommand('insertText', false, message);

    }

  }

  static autoFocus() {

    this.sync('');

  }

  static submit() {

    // subsequent screens use this

    let btn = document.querySelector('button[aria-label\*="Send Message"]');

    if (!btn) { // new chats use this

      btn = document.querySelector('button:has(div svg)');

    }

    if (!btn) { // last ditch attempt

      btn = document.querySelector('button:has(svg)');

    }

    if (btn) this.autoClick(btn);

  }

}

class GeminiAsk extends NoiAsk {

  static name = 'Gemini';

  static url = 'https://gemini.google.com';

  static sync(message) {

    const inputElement = document.querySelector('.ql-editor.textarea');

    if (inputElement) {

      const inputEvent = new Event('input', { bubbles: true });

      inputElement.value = message;

      inputElement.dispatchEvent(inputEvent);

      // bard is weird

      inputElement.querySelector('p').textContent = message;

    }

  }

  static autoFocus() {

    const inputElement = document.querySelector('.ql-editor.textarea');

    if (inputElement) {

      inputElement.focus();

    }

  }

  static submit() {

    const btn = document.querySelector('button[aria-label\*="Send message"]') || document.querySelector('button.send-button');

    if (btn) {

      btn.setAttribute('aria-disabled', 'false'); // doesn't work alone

      btn.focus();

      btn.click();

    }

  }

}

class HuggingChatAsk extends NoiAsk {

  static name = 'HuggingChat';

  static url = 'https://huggingface.co/chat';

  static sync(message) {

    var inputElement = document.querySelector('textarea[placeholder\*="Ask anything"]');

    if (inputElement) {

      const inputEvent = new Event('input', { bubbles: true });

      inputElement.value = message;

      inputElement.dispatchEvent(inputEvent);

    }

  }

  static autoFocus() {

    var inputElement = document.querySelector('textarea[placeholder\*="Ask anything"]');

    if (inputElement) {

      inputElement.focus();

    }

  }

  static submit() {

    var btn = document.querySelector('form.relative > div > button[type="submit"]');

    if (btn) this.autoClick(btn);

  }

}

class PerplexityAsk extends NoiAsk {

  static name = 'Perplexity';

  static url = 'https://www.perplexity.ai';

  static submit() {

    const btns = Array.from(document.querySelectorAll('button.bg-super'));

    if (btns[0]) {

      const btnsWithSvgPath = btns.filter(button => button.querySelector('svg path'));

      const btn = btnsWithSvgPath[btnsWithSvgPath.length - 1];

      btn.click();

    }

  }

}

class CopilotAsk extends NoiAsk {

  static name = 'Copilot';

  static url = 'https://copilot.microsoft.com';

  static sync(message) {

    // SERP Shadow DOM

    const serpDOM = document.querySelector('.cib-serp-main');

    // Action Bar Shadow DOM

    const inputDOM = serpDOM.shadowRoot.querySelector('#cib-action-bar-main');

    // Text Input Shadow DOM

    const textInputDOM = inputDOM.shadowRoot.querySelector('cib-text-input');

    // This inner cib-text-input Shadow DOM is not always present

    const inputElement = textInputDOM ? textInputDOM.shadowRoot.querySelector('#searchbox') : inputDOM.shadowRoot.querySelector('#searchbox');

    if (inputElement) {

      this.simulateUserInput(inputElement, message);

    }

  }

  static autoFocus() {

    const serpDOM = document.querySelector('.cib-serp-main');

    const inputDOM = serpDOM.shadowRoot.querySelector('#cib-action-bar-main');

    const textInputDOM = inputDOM.shadowRoot.querySelector('cib-text-input');

    const inputElement = textInputDOM ? textInputDOM.shadowRoot.querySelector('#searchbox') : inputDOM.shadowRoot.querySelector('#searchbox');

    if (inputElement) {

      inputElement.focus();

    }

  }

  static submit() {

    try {

      // Access SERP Shadow DOM

      const serpDOM = document.querySelector('.cib-serp-main');

      // Action Bar Shadow DOM

      const actionDOM = serpDOM.shadowRoot.querySelector('#cib-action-bar-main');

      // Submit Button

      const submitButton = actionDOM.shadowRoot.querySelector('div.submit button');

      if (submitButton) {

        submitButton.click();

        submitButton.focus();

        setTimeout(() => {

          submitButton.click();

        }, 100)

      }

    } catch (e) {

      console.error('Copilot submit error', e);

    }

  }

}

class PiAsk extends NoiAsk {

  static name = 'Pi';

  static url = 'https://pi.ai/talk';

  static submit() {

    const inputElement = document.querySelector('textarea[placeholder="Talk with Pi"]');

    if (inputElement) {

      const event = new KeyboardEvent('keydown', {

        key: 'Enter',

        view: window,

        bubbles: true

      });

      inputElement.dispatchEvent(event);

    }

  }

}

class CozeAsk extends NoiAsk {

  static name = 'Coze';

  static url = 'https://www.coze.com/home';

  static submit() {

    const inputElement = document.querySelector('textarea');

    if (inputElement) {

      const nextElement = inputElement.nextElementSibling;

      if (nextElement) {

        const btn = nextElement.querySelector('button');

        if (btn) btn.click();

      }

    }

  }

}

class YouAsk extends NoiAsk {

  static name = 'YOU';

  static url = 'https://you.com';

  static submit() {

    const btn = document.querySelector('button[data-eventactionname="click\_send"]');

    if (btn) btn.click();

  }

}

class CozeCNAsk extends NoiAsk {

  static name = 'Coze';

  static url = 'https://www.coze.cn/home';

  static submit() {

    const inputElement = document.querySelector('textarea');

    if (inputElement) {

      const nextElement = inputElement.nextElementSibling;

      if (nextElement) {

        const btn = nextElement.querySelector('button');

        if (btn) btn.click();

      }

    }

  }

}

class ChatGMLAsk extends NoiAsk {

  static name = 'ChatGLM'; // 智谱清言

  static url = 'https://chatglm.cn';

  static submit() {

    const btn = document.querySelector('#search-input-box .enter img');

    if (btn) btn.click();

  }

}

class DoubaoAsk extends NoiAsk {

  static name = 'Doubao'; // 豆包

  static url = 'https://www.doubao.com';

  static submit() {

    const btn = document.querySelector('#flow-end-msg-send');

    if (btn) btn.click();

  }

}

class TongyiAsk extends NoiAsk {

  static name = 'TongYi'; // 通义千问

  static url = 'https://tongyi.aliyun.com/qianwen';

  static submit() {

    const inputElement = document.querySelector('textarea');

    if (inputElement) {

      const nextElement = inputElement.nextElementSibling;

      if (nextElement) {

        const btn = nextElement.querySelector('div[class^="chatBtn"]');

        if (btn) btn.click();

      }

    }

  }

}

class GroqAsk extends NoiAsk {

  static name = 'Groq';

  static url = 'https://groq.com';

  static submit() {

    const btn = document.querySelector('form button[type="submit"]');

    if (btn) btn.click();

  }

}

class SunoAsk extends NoiAsk {

  static name = 'Suno AI';

  static url = 'https://app.suno.ai';

  static submit() {

    const btn = Array.from(document.querySelectorAll('button')).find(i => i.innerText.includes('Create'));

    if (btn) btn.click();

  }

}

window.NoiAsk = {

  OpenAIAsk,

  PoeAsk,

  ClaudeAsk,

  GeminiAsk,

  HuggingChatAsk,

  PerplexityAsk,

  CopilotAsk,

  PiAsk,

  CozeAsk,

  YouAsk,

  CozeCNAsk,

  DoubaoAsk,

  ChatGMLAsk,

  TongyiAsk,

  GroqAsk,

  SunoAsk,

};

noi/app-04.0/resources/extensions/noi-ask/main.js ---------- code end

noi/app-04.0/resources/extensions/noi-ask/manifest.json ------- code start

{

  "manifest\_version": 3,

  "name": "@noi/ask",

  "version": "0.1.7",

  "homepage": "https://github.com/lencx/Noi/tree/main/extensions/noi-ask",

  "description": "The best assistant for batch asking and quick typing of prompts.",

  "content\_scripts": [

    {

      "matches": [

        "https://chat.openai.com/\*",

        "https://gemini.google.com/\*",

        "https://poe.com/\*",

        "https://claude.ai/\*",

        "https://huggingface.co/chat/\*",

        "https://www.perplexity.ai/\*",

        "https://copilot.microsoft.com/\*",

        "https://groq.com/\*",

        "https://pi.ai/talk/\*",

        "https://www.coze.com/home/\*",

        "https://you.com/\*",

        "https://www.coze.cn/home/\*",

        "https://chatglm.cn/\*",

        "https://www.doubao.com/\*",

        "https://tongyi.aliyun.com/qianwen/\*",

        "https://app.suno.ai/\*"

      ],

      "js": ["main.js"],

      "run\_at": "document\_end",

      "world": "MAIN"

    }

  ]

}

noi/app-04.0/resources/extensions/noi-ask/manifest.json ------- code end

noi/app-04.0/resources/extensions/noi-export-chatgpt/main.js ------------ code start

const icons = {

  check: `<svg width="18" height="18" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"><path fill="currentColor" d="m10.6 16.2l7.05-7.05l-1.4-1.4l-5.65 5.65l-2.85-2.85l-1.4 1.4zM5 21q-.825 0-1.412-.587T3 19V5q0-.825.588-1.412T5 3h14q.825 0 1.413.588T21 5v14q0 .825-.587 1.413T19 21zm0-2h14V5H5zM5 5v14z"/></svg>`,

  noCheck: `<svg width="18" height="18" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"><path fill="currentColor" d="M5 21q-.825 0-1.412-.587T3 19V5q0-.825.588-1.412T5 3h14q.825 0 1.413.588T21 5v14q0 .825-.587 1.413T19 21zm0-2h14V5H5z"/></svg>`,

}

const QUERY\_CHAT\_LIST = 'main [role="presentation"] div[data-testid]';

const QUERY\_CHECKBOX\_AREA = '.empty\\:hidden div.visible';

const QUERY\_ACTION\_BUTTON = '.empty\\:hidden div.visible button';

window.noiExport = function() {

  const allNodes = Array.from(document.querySelectorAll(QUERY\_CHAT\_LIST));

  if (!allNodes.length) {

    return {

      selected: [],

      all: [],

    };

  }

  return {

    selected: window.\_noiSelectedNodes,

    all: allNodes.map((el) => {

      const node = el.cloneNode(true);

      const msgNode = node.querySelector('[data-message-author-role]');

      if (msgNode) {

        msgNode.className = 'whitespace-pre-wrap break-words';

      }

      return node;

    }),

  };

}

// ------------------------------

window.addEventListener('load', () => {

  window.\_noiSelectedNodes = [];

  const debouncedHandleMainChanges = debounce(handleMainChanges, 250);

  const observer = new MutationObserver(mutations => {

    mutations.forEach(mutation => {

      if (!mutation.target.form) {

        debouncedHandleMainChanges();

      }

    });

  });

  const mainElement = document.querySelector('main');

  observer.observe(mainElement, { childList: true, subtree: true });

  debouncedHandleMainChanges();

  changeURL(() => {

    window.\_noiSelectedNodes = [];

  });

})

function debounce(func, wait) {

  let timeout;

  return function(...args) {

    clearTimeout(timeout);

    timeout = setTimeout(() => {

      func.apply(this, args);

    }, wait);

  };

}

function createButtonForNode(i, index, nodesMap) {

  i.setAttribute('noi-node-id', index);

  nodesMap.set(index, i);

  const btnArea = i.querySelector(QUERY\_CHECKBOX\_AREA);

  if (!btnArea) return;

  if (btnArea.querySelector('button.noi-checkbox')) return;

  const defaultBtn = i.querySelector(QUERY\_ACTION\_BUTTON);

  if (!defaultBtn) return;

  const btn = document.createElement('button');

  btn.classList.add(...defaultBtn.classList, 'noi-checkbox');

  btn.innerHTML = icons.noCheck;

  btn.setAttribute('data-checked', 'false');

  btn.onclick = () => {

    const isChecked = btn.getAttribute('data-checked') === 'true';

    btn.innerHTML = isChecked ? icons.noCheck : icons.check;

    btn.setAttribute('data-checked', `${!isChecked}`);

    i.classList.toggle('noi-selected', !isChecked);

    updateSelectedNodes(isChecked, i.cloneNode(true), nodesMap);

  }

  btnArea.insertBefore(btn, btnArea.firstChild);

}

function updateSelectedNodes(isChecked, node, nodesMap) {

  const nodeId = node.getAttribute('noi-node-id');

  const indexInAll = nodesMap.get(Number(nodeId));

  const indexInSelected = window.\_noiSelectedNodes.findIndex(n => n.getAttribute('noi-node-id') === nodeId);

  if (!isChecked) {

    if (indexInSelected === -1) {

      let inserted = false;

      for (let i = 0; i < window.\_noiSelectedNodes.length; i++) {

        const currentId = window.\_noiSelectedNodes[i].getAttribute('noi-node-id');

        if (nodesMap.get(Number(currentId)) > indexInAll) {

          window.\_noiSelectedNodes.splice(i, 0, node);

          inserted = true;

          break;

        }

      }

      if (!inserted) {

        const msgNode = node.querySelector('[data-message-author-role]');

        if (msgNode) {

          msgNode.className = 'whitespace-pre-wrap break-words';

        }

        window.\_noiSelectedNodes.push(node);

      }

    }

  } else {

    if (indexInSelected > -1) {

      window.\_noiSelectedNodes.splice(indexInSelected, 1);

    }

  }

  window.\_noiSelectedNodes.sort((a, b) => {

    const nodeIdA = a.getAttribute('noi-node-id');

    const nodeIdB = b.getAttribute('noi-node-id');

    return Number(nodeIdA) - Number(nodeIdB);

  });

  console.log(window.\_noiSelectedNodes);

}

function handleMainChanges() {

  const nodesMap = new Map();

  document.querySelectorAll(QUERY\_CHAT\_LIST)

    .forEach((node, index) => createButtonForNode(node, index, nodesMap));

}

(function (history) {

  function triggerEvent(eventName, state) {

    if (typeof history[eventName] === 'function') {

      history[eventName]({ state: state, url: window.location.href });

    }

  }

  function overrideHistoryMethod(methodName, eventName) {

    const originalMethod = history[methodName];

    history[methodName] = function (state, ...rest) {

      const result = originalMethod.apply(this, [state, ...rest]);

      triggerEvent(eventName, state);

      return result;

    };

  }

  overrideHistoryMethod('pushState', 'onpushstate');

  overrideHistoryMethod('replaceState', 'onreplacestate');

  window.addEventListener('popstate', () => {

    triggerEvent('onpopstate', null);

  });

})(window.history);

function changeURL(callback) {

  window.history.onpushstate = (event) => {

    console.log('pushState called:', event.url);

    callback && callback(event);

  };

  window.history.onreplacestate = (event) => {

    console.log('replaceState called:', event.url);

    callback && callback(event);

  };

  window.history.onpopstate = (event) => {

    console.log('popstate event triggered', window.location.href);

    callback && callback(event);

  };

}

noi/app-04.0/resources/extensions/noi-export-chatgpt/main.js ------------ code end

noi/app-04.0/resources/extensions/noi-export-chatgpt/manifest.json ------------ code start

{

  "manifest\_version": 3,

  "name": "@noi/export-chatgpt",

  "version": "0.1.0",

  "homepage": "https://github.com/lencx/Noi/tree/main/extensions/noi-export-chatgpt",

  "description": "ChatGPT chat history export, supports PDF, Image, and Markdown formats.",

  "content\_scripts": [

    {

      "matches": ["https://chat.openai.com/\*"],

      "js": ["main.js"],

      "css": ["style.css"],

      "run\_at": "document\_end",

      "world": "MAIN"

    }

  ]

}

noi/app-04.0/resources/extensions/noi-export-chatgpt/manifest.json ------------ code end

noi/app-04.0/resources/extensions/noi-reset/darg.js ------------ code start

// fix: sidebar not clickable

function removeAppRegion() {

  const allElements = document.querySelectorAll('\*');

  for (let element of allElements) {

    const style = window.getComputedStyle(element);

    if (style.webkitAppRegion === 'drag' || style.webkitAppRegion === 'no-drag') {

      element.style.webkitAppRegion = 'initial';

      // console.log('Removed -webkit-app-region from:', element);

    }

  }

}

setTimeout(removeAppRegion, 3000);

noi/app-04.0/resources/extensions/noi-reset/darg.js ------------ code end

noi/app-04.0/resources/extensions/noi-reset/manifest.json ------------ code start

{

  "manifest\_version": 3,

  "name": "@noi/reset",

  "version": "0.1.1",

  "homepage": "https://github.com/lencx/Noi/tree/main/extensions/noi-reset",

  "description": "Reset certain website styles to enhance compatibility with Noi.",

  "content\_scripts": [

    {

      "matches": [

        "https://vscode.dev/\*"

      ],

      "js": ["darg.js"],

      "run\_at": "document\_end",

      "world": "MAIN"

    },

    {

      "matches": [

        "<all\_urls>"

      ],

      "js": ["scrollbar.js"],

      "run\_at": "document\_end",

      "world": "MAIN"

    }

  ]

}

noi/app-04.0/resources/extensions/noi-reset/manifest.json ------------ code end

noi/app-04.0/resources/extensions/noi-reset/scrollbar.js ------------ code start

if (navigator.platform.toUpperCase().indexOf('MAC') < 0) {

  const style = document.createElement('style');

  style.innerHTML = `

    html {

      --noi-scrollbarBG: rgba(250, 250, 250, 0.1);

      --noi-thumbBG: #d8d8d8;

    }

    ::-webkit-scrollbar {

      width: 8px;

      height: 8px;

    }

    ::-webkit-scrollbar-corner {

      background-color: transparent;

    }

    ::-webkit-scrollbar-track {

      background: var(--noi-scrollbarBG);

      border-radius: 2px;

    }

    ::-webkit-scrollbar-thumb {

      border-radius: 2px;

      background-color: var(--noi-thumbBG);

    }

  `;

  document.head.appendChild(style);

}

noi/app-04.0/resources/extensions/noi-reset/scrollbar.js ------------ code end

noi/app-04.0/resources/extensions/extensions.json ------------ code start

{

  "name": "Noi Extensions",

  "homepage": "https://github.com/lencx/Noi/blob/main/extensions/noi.extensions.json",

  "extensions": [

    {

      "name": "@noi/ask",

      "description": "The best assistant for batch asking and quick typing of prompts.",

      "version": "0.1.7",

      "url": "https://github.com/lencx/Noi/tree/main/extensions/noi-ask",

      "dirname": "noi-ask",

      "disabled": false

    },

    {

      "name": "@noi/ask-custom",

      "description": "The best assistant for batch asking and quick typing of prompts.",

      "version": "0.1.0",

      "url": "https://github.com/lencx/Noi/tree/main/extensions/noi-ask-custom",

      "dirname": "noi-ask-custom",

      "disabled": false

    },

    {

      "name": "@noi/export-chatgpt",

      "description": "ChatGPT chat history export, supports PDF, Image, and Markdown formats.",

      "version": "0.1.0",

      "url": "https://github.com/lencx/Noi/tree/main/extensions/noi-export-chatgpt",

      "dirname": "noi-export-chatgpt",

      "disabled": false

    },

    {

      "name": "@noi/reset",

      "description": "Reset certain website styles to enhance compatibility with Noi.",

      "version": "0.1.1",

      "url": "https://github.com/lencx/Noi/tree/main/extensions/noi-reset",

      "dirname": "noi-reset",

      "disabled": false

    }

  ]

}

noi/app-04.0/resources/extensions/extensions.json ------------ code end

noi/app-04.0/resources/extensions/README.md ------------ code start

# Noi Extensions

Note that Noi does not support the full range of Chrome extensions APIs. See Supported Extensions APIs for more details on what is supported.

Learn more: [electronjs/doc](https://www.electronjs.org/docs/latest/api/extensions)

## Extensions

<!-- EXTENSIONS\_START -->

| Name | Version | Description |

| --- | --- | --- |

| [@noi/ask](https://github.com/lencx/Noi/tree/main/extensions/noi-ask) | 0.1.7 | The best assistant for batch asking and quick typing of prompts. |

| [@noi/ask-custom](https://github.com/lencx/Noi/tree/main/extensions/noi-ask-custom) | 0.1.0 | The best assistant for batch asking and quick typing of prompts. |

| [@noi/export-chatgpt](https://github.com/lencx/Noi/tree/main/extensions/noi-export-chatgpt) | 0.1.0 | ChatGPT chat history export, supports PDF, Image, and Markdown formats. |

| [@noi/reset](https://github.com/lencx/Noi/tree/main/extensions/noi-reset) | 0.1.1 | Reset certain website styles to enhance compatibility with Noi. |

<!-- EXTENSIONS\_END -->

noi/app-04.0/resources/extensions/README.md ------------ code end

noi/app-04.0/resources/libs/main.js ------------ code start

window.noi = {

  html2canvas: window.html2canvas,

  TurndownGfm: window.turndownPluginGfm,

  Turndown: window.TurndownService,

  parseMD(html) {

    const hljsREG = /^.\*(hljs).\*(language-[a-z0-9]+).\*$/i;

    const gfm = window.turndownPluginGfm.gfm;

    const turndownService = new window.TurndownService({

      headingStyle: 'atx',

      bulletListMarker: '-',

      strongDelimiter: '\*\*',

      emDelimiter: '\_',

      codeBlockStyle: 'fenced',

      fence: '```',

      linkStyle: 'inlined',

      hr: '---',

    })

      .addRule('code', {

        filter(node) {

          if (node.nodeName === 'CODE' && hljsREG.test(node.classList.value)) {

            return true;

          }

          return false;

        },

        replacement(content, node) {

          const classStr = node.getAttribute('class');

          if (hljsREG.test(classStr)) {

            const lang = classStr.match(/.\*language-(\w+)/)[1];

            if (lang) {

              return `\`\`\`${lang}\n${content}\n\`\`\``;

            }

            return `\`\`\`\n${content}\n\`\`\``;

          }

          return content;

        },

      })

      .addRule('ignore-text', {

        filter: (node) => {

          if (node.nodeName === 'DIV' && node.classList.contains('!invisible')) {

            return true;

          }

          return false;

        },

        replacement: () => '',

      })

      .addRule('ignore', {

        filter: ['button', 'img', 'input', 'textarea', 'svg'],

        replacement: () => '',

      })

      .addRule('ignoreCertainClasses', {

        filter(node) {

          return ['#noi-watermark'].some((selector) => node.matches(selector));

        },

        replacement: () => '',

      })

      .addRule('table', {

        filter: 'table',

        replacement(content) {

          return `\`\`\`${content}\n\`\`\``;

        },

      });

    turndownService.use(gfm);

    return turndownService.turndown(html);

  },

}

noi/app-04.0/resources/libs/main.js ------------ code end