

```
1 Ignorer et passer au contenu
2 Historique de chat
3
4 Vous avez dit :
5 je voudrais créer un plugin pour gérer un treeview
6 VisualNEO Web Expert a dit :
7 Excellente idée ☺ Créer un plugin TreeView pour VisualNEO Web est tout à fait
  possible.
8 On peut s'inspirer de la structure du plugin neoTable et des indications de la
  documentation officielle sur la création de plugins.
9
10 Étapes pour créer un plugin TreeView
11 1. Utiliser le Plugin Generator
12 VisualNEO Web inclut un outil appelé Plugin Generator (menu Tools → Plugin Generator).
13 Cet outil crée le "squelette" de ton plugin : un fichier .json qui définit les
  commandes, et un fichier .js (JavaScript) qui implémente la logique.
14
15 2. Choisir une bibliothèque JavaScript TreeView
16 Comme VisualNEO Web est basé sur HTML/JS, tu peux intégrer une librairie existante.
  Quelques options populaires :
17
18 Bootstrap Treeview : simple, basé sur Bootstrap.
19
20 jsTree : très complet, avec événements, drag&drop, checkboxes, etc.
21
22 VanillaJS TreeView : si tu veux garder le plugin léger.
23
24 3. Définir les commandes NeoScript
25 Ton plugin doit exposer des commandes NeoScript, comme pour neoTable :
26
27 Exemple (pseudo-code pour un futur neoTree plugin) :
28
29 neoTreeCreate [treeId] [targetObject] [jsonData]
30 neoTreeAddNode [treeId] [parentNodeId] [nodeId] [nodeText]
31 neoTreeRemoveNode [treeId] [nodeId]
32 neoTreeExpand [treeId] [nodeId]
33 neoTreeCollapse [treeId] [nodeId]
34 Chaque commande sera définie dans ton fichier JSON, avec description et paramètres.
35
36 4. Implémenter en JavaScript
37 Ton fichier .js contiendra les fonctions appelées par VisualNEO Web.
38 Exemple minimal pour créer un TreeView dans un container VisualNEO :
39
40 // Créer un TreeView
41 NeoPlugins.neoTreeCreate = function (treeId, targetId, jsonData) {
42   let container = document.getElementById(targetId);
43   if (!container) return;
44   container.innerHTML = "";
45
46   let data = JSON.parse(jsonData);
47   let ul = document.createElement("ul");
48
49   function buildTree(nodes, parent) {
50     nodes.forEach(node => {
51       let li = document.createElement("li");
52       li.textContent = node.text;
53       li.id = node.id;
54       parent.appendChild(li);
55       if (node.children) {
56         let childUl = document.createElement("ul");
57         li.appendChild(childUl);
58         buildTree(node.children, childUl);
59       }
60     });
61   }
62
63   buildTree(data, ul);
64   container.appendChild(ul);
65 };
66 5. Créer un tutoriel de test
67 Comme avec neoTable, tu peux créer une app de démonstration où :
68
69 Tu appelles neoTreeCreate sur un Container.
```

```

70
71 Tu ajoutes des nœuds via NeoScript.
72
73 Tu relies des événements (clic sur un nœud → affiche une alerte ou met une variable).
74
75 Ressources utiles
76 Exemple complet plugin (neoTable) : montre comment déclarer des commandes et gérer
  les événements.
77
78 Documentation Plugins (section "Plugin Generator" et "Plugins Command Reference").
79
80 ☞ Veux-tu que je prépare un modèle complet de plugin neoTree (avec .json + .js) que
  tu pourras directement importer dans VisualNEO Web, comme pour neoTable ?
81
82 Vous avez dit :
83 oui
84 VisualNEO Web Expert a dit :
85
86 Réflexion durant 51s
87 Super ! Voici un modèle prêt-à-adapter pour un plugin neoTree (TreeView) pour
  VisualNEO Web : il comprend la définition JSON des commandes, le code JavaScript, un
  peu de CSS, ainsi qu'un mini-projet de test (NeoScript) pour le valider.
88
89 Je m'appuie sur la logique utilisée dans le tutoriel du plugin neoTable (comment
  structurer les commandes et manipuler des variables/événements) et sur la section
  Plugin Generator de l'aide officielle, afin que tout reste 100 % conforme à VisualNEO
  Web.
90
91 1) Arborescence du plugin
92 VisualNEO Web/
93 └─ Plugins/
94   └─ neoTree/
95       └─ neoTree.json      ← déclaration des commandes (Plugin Generator)
96       └─ neoTree.js        ← implémentation
97       └─ neoTree.css       ← styles (facultatif mais recommandé)
98 Astuce : crée d'abord un squelette avec Tools → Plugin Generator (nom: neoTree) puis
  remplace le contenu des 3 fichiers par ceux ci-dessous.
99
100 2) Fichier neoTree.json
101 Ce format correspond à ce que génère le Plugin Generator : nom, version, fichiers
  inclus et commandes visibles dans l'Action Editor.
102 (Si ton squelette ajoute d'autres champs, garde-les ; remplace seulement la section
  commands.)
103
104 {
105   "name": "neoTree",
106   "id": "neoTree",
107   "version": "1.0.0",
108   "author": "Ton Nom",
109   "description": "TreeView (arborescences) pour VisualNEO Web.",
110   "files": ["neoTree.js", "neoTree.css"],
111   "commands": [
112     {
113       "name": "neoTreeCreate",
114       "params": "[treeId]|STRING|ID logique du tree|[targetObject]|STRING|ID d'un
        objet conteneur (Container, Paragraph, etc.)|[jsonData]|STRING|Données JSON
        (tableau de nœuds)",
115       "help": "Crée/rend un TreeView dans l'objet cible. jsonData = tableau JSON de
        nœuds {id, text, children:[]}.",
116       "category": "neoTree"
117     },
118     {
119       "name": "neoTreeDestroy",
120       "params": "[treeId]|STRING|ID du tree",
121       "help": "Supprime le TreeView et libère ses ressources.",
122       "category": "neoTree"
123     },
124     {
125       "name": "neoTreeAddNode",
126       "params": "[treeId]|STRING|ID du tree|[parentNodeId]|STRING|ID du parent (ou
        vide pour racine)|[nodeId]|STRING|ID du nœud|[nodeText]|STRING|Texte du nœud",
127       "help": "Ajoute un nœud sous le parent spécifié.",
128       "category": "neoTree"

```

```

129     },
130     {
131         "name": "neoTreeRemoveNode",
132         "params": "[treeId]|STRING|ID du tree|[nodeId]|STRING|ID du nœud",
133         "help": "Supprime un nœud (et ses enfants).",
134         "category": "neoTree"
135     },
136     {
137         "name": "neoTreeExpand",
138         "params": "[treeId]|STRING|ID du tree|[nodeId]|STRING|ID du nœud",
139         "help": "Déploie un nœud.",
140         "category": "neoTree"
141     },
142     {
143         "name": "neoTreeCollapse",
144         "params": "[treeId]|STRING|ID du tree|[nodeId]|STRING|ID du nœud",
145         "help": "Replie un nœud.",
146         "category": "neoTree"
147     },
148     {
149         "name": "neoTreeSelect",
150         "params": "[treeId]|STRING|ID du tree|[nodeId]|STRING|ID du nœud",
151         "help": "Sélectionne un nœud (et déclenche l'événement interne).",
152         "category": "neoTree"
153     },
154     {
155         "name": "neoTreeGetSelected",
156         "params": "[treeId]|STRING|ID du tree|[returnVar]|STRING|Variable VisualNEO de sortie",
157         "help": "Retourne l'ID du nœud sélectionné dans [returnVar].",
158         "category": "neoTree"
159     },
160     {
161         "name": "neoTreeLoadJSON",
162         "params": "[treeId]|STRING|ID du tree|[jsonData]|STRING|Données JSON (tableau de nœuds)",
163         "help": "Remplace l'arbre courant par les données JSON.",
164         "category": "neoTree"
165     },
166     {
167         "name": "neoTreeToJSON",
168         "params": "[treeId]|STRING|ID du tree|[returnVar]|STRING|Variable VisualNEO de sortie",
169         "help": "Exporte l'arbre au format JSON dans [returnVar].",
170         "category": "neoTree"
171     },
172     {
173         "name": "neoTreeOn",
174         "params": "[treeId]|STRING|ID du tree|[eventName]|STRING|evenement (click)|[actionCode]|STRING|Actions VisualNEO à exécuter",
175         "help": "Associe des actions VisualNEO à un événement (ex : click). Les variables [neoTreeEvent], [neoTreeNodeId], [neoTreeNodeText] sont remplies avant l'exécution.",
176         "category": "neoTree"
177     }
178 ]
179 }
180 3) Fichier neoTree.css (simple, modulable)
181 /* Container du tree */
182 .neoTree { font-family: system-ui, Arial, sans-serif; font-size: 14px; line-height: 1.3; }
183 .neoTree ul { list-style: none; padding-left: 1rem; margin: 0.2rem 0; }
184 .neoTree li { position: relative; margin: 0.1rem 0; cursor: default; }
185
186 .neoTree__label { display: inline-flex; align-items: center; gap: .4rem; cursor: pointer; user-select: none; }
187 .neoTree__toggle { width: .9rem; height: .9rem; display: inline-block; transform: rotate(-90deg); transition: transform .15s ease; }
188 .neoTree__toggle::before { content: "□"; font-size: .8rem; color: #666; }
189 .neoTree__toggle.is-open { transform: rotate(0deg); }
190
191 .neoTree__text { padding: .08rem .25rem; border-radius: .2rem; }
192 .neoTree__text.is-selected { background: #ed792f; color: #fff; }

```

```

193
194 .neoTree__children { display: none; margin-left: .8rem; }
195 .neoTree__children.is-open { display: block; }
196 4) Fichier neoTree.js
197 Implémentation « vanilla » (pas de dépendances).
198 Le pont avec VisualNEO : si l'API JS de VNW est dispo, on utilise Neo.setVar et
Neo.executeActions (noms courants). Sinon on dégrade gentiment (les variables seront
posées en window[...]). Adapte ces 2 helpers si besoin selon la doc Accessing
Variables from JavaScript / Calling actions from JavaScript.
199
200 /* neoTree.js - v1.0.0 */
201
202 (function () {
203     const REGISTRY = {}; // treeId -> instance
204
205     // ==== Helpers VisualNEO bridge (adapte les noms si besoin) ====
206     function setVar(name, value) {
207         if (window.Neo && typeof Neo.setVar === "function") Neo.setVar(name,
String(value));
208         else window[name] = String(value);
209     }
210     function executeActions(code) {
211         if (!code) return;
212         if (window.Neo && typeof Neo.executeActions === "function")
Neo.executeActions(code);
213         else if (window.Neo && typeof Neo.callAction === "function") Neo.callAction(code);
214         // sinon: pas d'exécution (mode test sans bridge)
215     }
216
217     // ==== DOM utils ====
218     function el(tag, cls) { const e = document.createElement(tag); if (cls) e.className
= cls; return e; }
219     function byId(id) { return document.getElementById(id); }
220
221     // ==== Core Tree ====
222     class NeoTree {
223         constructor(treeId, targetId, data) {
224             this.id = treeId;
225             this.targetId = targetId;
226             this.actions = { click: "" }; // events -> actionCode
227             this.selectedId = "";
228             this.nodesById = new Map(); // id -> node model
229             this.root = Array.isArray(data) ? data : [];
230             this.container = null;
231         }
232
233         mount() {
234             const target = byId(this.targetId);
235             if (!target) return;
236             // clean + host
237             target.innerHTML = "";
238             const host = el("div", "neoTree");
239             host.id = "neoTree__" + this.id;
240             target.appendChild(host);
241             this.container = host;
242             // render
243             this.nodesById.clear();
244             const ul = el("ul");
245             this._buildTree(this.root, ul);
246             host.appendChild(ul);
247         }
248
249         destroy() {
250             const target = byId(this.targetId);
251             if (target) target.innerHTML = "";
252             this.nodesById.clear();
253             this.container = null;
254         }
255
256         _buildTree(nodes, parentUL) {
257             nodes.forEach(node => {
258                 // normalize
259                 if (!node || !node.id) return;

```

```

260     node.text = node.text || node.id;
261     node.children = Array.isArray(node.children) ? node.children : [];
262     this.nodesById.set(node.id, node);
263
264     const li = el("li");
265     li.dataset.nodeId = node.id;
266
267     const label = el("span", "neoTree__label");
268     // toggle (only if has children)
269     let toggle = null;
270     if (node.children.length) {
271         toggle = el("span", "neoTree__toggle");
272         label.appendChild(toggle);
273     } else {
274         // spacer to align
275         const spacer = el("span");
276         spacer.style.display = "inline-block";
277         spacer.style.width = ".9rem";
278         label.appendChild(spacer);
279     }
280
281     const text = el("span", "neoTree__text");
282     text.textContent = node.text;
283     label.appendChild(text);
284     li.appendChild(label);
285
286     // children block
287     let childrenUL = null;
288     if (node.children.length) {
289         childrenUL = el("ul", "neoTree__children");
290         li.appendChild(childrenUL);
291         this._buildTree(node.children, childrenUL);
292     }
293
294     // interactions
295     label.addEventListener("click", (ev) => {
296         ev.stopPropagation();
297         this._select(node.id, text);
298         // click event to VNW
299         setVar("[neoTreeEvent]", "click");
300         setVar("[neoTreeNodeId]", node.id);
301         setVar("[neoTreeNodeText]", node.text);
302         executeActions(this.actions.click);
303     });
304
305     if (toggle && childrenUL) {
306         label.addEventListener("dblclick", (ev) => {
307             ev.stopPropagation();
308             // toggle open/close
309             const isOpen = childrenUL.classList.toggle("is-open");
310             toggle.classList.toggle("is-open", isOpen);
311         });
312     }
313
314     parentUL.appendChild(li);
315 });
316 }
317
318 _select(nodeId, textEl) {
319     // clear previous selection
320     if (this.container) {
321         const prev = this.container.querySelector(".neoTree__text.is-selected");
322         if (prev) prev.classList.remove("is-selected");
323     }
324     this.selectedId = nodeId;
325     if (textEl) textEl.classList.add("is-selected");
326 }
327
328 addNode(parentId, nodeId, nodeText) {
329     const newNode = { id: nodeId, text: nodeText || nodeId, children: [] };
330     if (!parentId) {
331         this.root.push(newNode);
332     } else {

```

```

333         const parent = this.nodesById.get(parentId);
334         if (!parent) return;
335         parent.children = parent.children || [];
336         parent.children.push(newNode);
337     }
338     this.mount(); // simple re-render
339 }
340
341 removeNode(nodeId, nodes = this.root) {
342     for (let i = 0; i < nodes.length; i++) {
343         if (nodes[i].id === nodeId) {
344             nodes.splice(i, 1);
345             this.mount();
346             return true;
347         }
348         if (nodes[i].children && nodes[i].children.length) {
349             if (this.removeNode(nodeId, nodes[i].children)) return true;
350         }
351     }
352     return false;
353 }
354
355 expand(nodeId) {
356     if (!this.container) return;
357     const li = this.container.querySelector(`li[data-node-id="${nodeId}"]`);
358     if (!li) return;
359     const children = li.querySelector(".neoTree__children");
360     const toggle = li.querySelector(".neoTree__toggle");
361     if (children) children.classList.add("is-open");
362     if (toggle) toggle.classList.add("is-open");
363 }
364
365 collapse(nodeId) {
366     if (!this.container) return;
367     const li = this.container.querySelector(`li[data-node-id="${nodeId}"]`);
368     if (!li) return;
369     const children = li.querySelector(".neoTree__children");
370     const toggle = li.querySelector(".neoTree__toggle");
371     if (children) children.classList.remove("is-open");
372     if (toggle) toggle.classList.remove("is-open");
373 }
374
375 select(nodeId) {
376     if (!this.container) return;
377     const li = this.container.querySelector(`li[data-node-id="${nodeId}"]`);
378     if (!li) return;
379     const textEl = li.querySelector(".neoTree__text");
380     if (textEl) this._select(nodeId, textEl);
381     // event to VNW (like a user click)
382     setVar("[neoTreeEvent]", "click");
383     setVar("[neoTreeNodeId]", nodeId);
384     setVar("[neoTreeNodeText]", (this.nodesById.get(nodeId) || {}).text || nodeId);
385     executeActions(this.actions.click);
386 }
387
388 toJSON() { return JSON.stringify(this.root); }
389
390 loadJSON(data) {
391     this.root = Array.isArray(data) ? data : [];
392     this.mount();
393 }
394
395 on(eventName, actionCode) {
396     this.actions[eventName] = actionCode || "";
397 }
398 }
399
400 // ==== Commands exposed to VisualNEO ====
401 window.NeoPlugins = window.NeoPlugins || {};
402
403 window.NeoPlugins.neoTreeCreate = function (treeId, targetObject, jsonData) {
404     try {
405         const data = JSON.parse(jsonData || "[]");

```

```

406         if (REGISTRY[treeId]) REGISTRY[treeId].destroy();
407         const tree = new NeoTree(treeId, targetObject, data);
408         REGISTRY[treeId] = tree;
409         tree.mount();
410     } catch (e) { console.error("neoTreeCreate JSON error:", e); }
411 };
412
413 window.NeoPlugins.neoTreeDestroy = function (treeId) {
414     if (REGISTRY[treeId]) { REGISTRY[treeId].destroy(); delete REGISTRY[treeId]; }
415 };
416
417 window.NeoPlugins.neoTreeAddNode = function (treeId, parentNodeId, nodeId,
418     nodeText) {
419     const tree = REGISTRY[treeId]; if (!tree) return;
420     tree.addNode(parentNodeId || "", nodeId, nodeText);
421 };
422
423 window.NeoPlugins.neoTreeRemoveNode = function (treeId, nodeId) {
424     const tree = REGISTRY[treeId]; if (!tree) return;
425     tree.removeNode(nodeId);
426 };
427
428 window.NeoPlugins.neoTreeExpand = function (treeId, nodeId) {
429     const tree = REGISTRY[treeId]; if (!tree) return;
430     tree.expand(nodeId);
431 };
432
433 window.NeoPlugins.neoTreeCollapse = function (treeId, nodeId) {
434     const tree = REGISTRY[treeId]; if (!tree) return;
435     tree.collapse(nodeId);
436 };
437
438 window.NeoPlugins.neoTreeSelect = function (treeId, nodeId) {
439     const tree = REGISTRY[treeId]; if (!tree) return;
440     tree.select(nodeId);
441 };
442
443 window.NeoPlugins.neoTreeGetSelected = function (treeId, returnVar) {
444     const tree = REGISTRY[treeId]; if (!tree) return;
445     setVar(returnVar, tree.selectedId || "");
446 };
447
448 window.NeoPlugins.neoTreeLoadJSON = function (treeId, jsonData) {
449     const tree = REGISTRY[treeId]; if (!tree) return;
450     try { tree.loadJSON(JSON.parse(jsonData || "[]")); }
451     catch (e) { console.error("neoTreeLoadJSON JSON error:", e); }
452 };
453
454 window.NeoPlugins.neoTreeToJSON = function (treeId, returnVar) {
455     const tree = REGISTRY[treeId]; if (!tree) return;
456     setVar(returnVar, tree.toJSON());
457 };
458
459 window.NeoPlugins.neoTreeOn = function (treeId, eventName, actionCode) {
460     const tree = REGISTRY[treeId]; if (!tree) return;
461     tree.on(eventName, actionCode);
462 };
463
464 5) Exemple d'utilisation (NeoScript) dans une page de test
465 Crée un Container (id : TreeHost) et un Paragraph (id : Log) pour voir les événements.
466 Copie-colle ces commandes (elles sont valides pour VisualNEO : SetVar, SetObjectHTML,
467     MessageBox, etc. sont documentées).
468 Remarque : on utilise des quotes simples dans le JSON pour faciliter la saisie ;
469     VisualNEO accepte très bien cette chaîne.
470
471 ; --- Données JSON ---
472 SetVar [treeData] "[{'id':'root','text':'Racine','children':[{'id':'n1','text':'Noeud
473     1'},{'id':'n2','text':'Noeud 2','children':[{'id':'n2a','text':'Feuille
474     2A'},{'id':'n2b','text':'Feuille 2B'}]}]]]"
475
476 ; --- Création du tree ---
477 neoTreeCreate "myTree" "TreeHost" [treeData]

```

```
474
475 ; --- Brancher un événement click (log + alerte) ---
476 neoTreeOn "myTree" "click" "SetObjectHTML 'Log' 'Clique: [neoTreeNodeId] -
[neoTreeNodeText]'; AlertBox 'Tree' 'Clique sur [neoTreeNodeId]' '"
477
478 ; --- Démo: ajout et interaction ---
479 neoTreeAddNode "myTree" "root" "n3" "Noeud 3"
480 neoTreeExpand "myTree" "root"
481 neoTreeSelect "myTree" "n3"
482
483 ; --- Récupérer la sélection ---
484 neoTreeGetSelected "myTree" "[currentNode]"
485 AlertBox "Sélection" "Nœud sélectionné: [currentNode]" ""
486 6) Points clés & bonnes pratiques
487 Événements : neoTreeOn remplit automatiquement les variables
488
489 [neoTreeEvent] (ex : click)
490
491 [neoTreeNodeId]
492
493 [neoTreeNodeText]
494 puis exécute le code d'actions que tu fournis (même approche que les tutoriels de
plugins : actions déclenchées côté VisualNEO).
495
496 Bridge VNW : si jamais Neo.setVar / Neo.executeActions de l'API interne avaient un
nom différent dans ta version, adapte les 2 helpers en haut de neoTree.js (voir
chapitres Accessing Variables from JavaScript et Calling actions from JavaScript de
l'aide).
497
498 Performances : pour de grands arbres, remplace mount() complet par un rendu
incrémental (DOM patching) ; ce modèle ré-affiche simplement (simple et fiable pour
commencer).
499
500 Style : modifie neoTree.css pour un look Bootstrap-like si tu utilises Bootstrap dans
ton app.
501
502 7) Installation
503 Tools → Plugin Generator → crée neoTree.
504
505 Remplace le contenu de neoTree.json, neoTree.js, neoTree.css par ceux ci-dessus.
506
507 Redémarre VisualNEO Web si besoin. Les commandes apparaissent sous la catégorie
neoTree dans l'Action Editor.
508
509
```