ac_sidebar

Classe JS autonome — Barre de navigation configurable

Jean-Noël Vidallet · artisan-code.fr · v1.2.0 · Août 2026

---

1. Présentation

ac_sidebar est une classe JavaScript autonome qui génère une barre de navigation configurable, avec menus et sous-menus, filtrage par rôle, plusieurs positions (dont un mode libre déplaçable avec magnétisme), et un thème capable de s'intégrer automatiquement au design d'une page existante.

Caractéristiques principales :

- Classe JS unique — CSS et jeu d'icônes SVG entièrement embarqués
- Zéro dépendance externe
- 2 niveaux de menu (menu → sous-menu), un menu pouvant contenir des options directes et/ou un sous-menu
- Filtrage par rôle utilisateur (1 à 5) avec masquage automatique des menus devenus vides
- 5 positions : gauche, droite, haut, bas, libre (déplaçable à la souris/au doigt)
- Mode libre avec magnétisme — se rapproche d'un bord d'écran, elle s'y ancre et redevient une sidebar classique
- Repli en rail d'icônes ou en burger seul (masquable), déploiement au survol de la souris
- En-tête logo + titre, bouton « épingler » pour forcer le déploiement permanent
- 3 thèmes : clair, sombre, système (déduit automatiquement des couleurs de la page hôte)
- Couleurs personnalisées, badges compteurs colorés par entrée
- Multi-instances — plusieurs sidebars indépendantes sur une même page
- Mémorisation optionnelle de l'état entre deux visites (localStorage)

---

2. Installation et fichiers

Fichier	Emplacement
ac_sidebar.js	/assets/classes/ac_sidebar.js

Inclusion dans la page (avant `</body>`) :

```
<script src="/assets/classes/ac_sidebar.js"></script>
```

Dans le HTML, placer un conteneur :

```
<div id="ma_sidebar"></div>
```

---

3. Utilisation minimale

```
const sb = new AcSidebar('#ma_sidebar', {
    roleUtilisateur: 2,
    entete: { titre: 'Mon application' },
    menus: [
        {
            libelle: 'Accueil', icone: 'maison',
            chemin: '/accueil.php', droitMinimum: 5,
        },
        {
            libelle: 'Administration', icone: 'bouclier',
            enfants: [
                {
                    libelle: 'Utilisateurs',
                    icone: 'groupe',
                    chemin: '/utilisateurs.php',
                    droitMinimum: 2,
                },
            ],
        },
    ],
});
```

Chargement depuis un fichier JSON externe, plutôt qu'un objet JS :

```
const sb = await AcSidebar.depuisJson('#ma_sidebar',
'/config/sidebar.json');
```

---

4. Options de configuration

Option	Type	Défaut	Description
police	objet	null	{ principale, secours: [...] } — police de la sidebar, avec liste de secours
theme	string	'clair'	'clair' | 'sombre' | 'systeme' (déduit de la page hôte)
couleurs	objet	null	Surcharge 7 couleurs (voir section 7) — prime sur le thème, y compris sur le burger
largeur	number | 'auto'	'auto'	Largeur déployée en px, ou calculée automatiquement sur le contenu (voir ci-dessous)
largeurRail	number	64	Largeur du rail replié en px (si masquable=false)
roleUtilisateur	number	5	Rôle courant, 1 à 5 — plus petit = plus de pouvoirs
position	string	'gauche'	'gauche' | 'droite' | 'haut' | 'bas' | 'libre'

masqueParDefaut	bool	false	La sidebar démarre repliée ou déployée
masquable	bool	false	false = rail d'icônes replié ; true = burger seul, rien de visible
deploiementAuSurvol	bool	true	Déploiement automatique au survol de la souris (désactivé si masquable=true)
pointRupture	number	768	Largeur de fenêtre (px) sous laquelle la sidebar se comporte comme masquable=true, quelle que soit la config — 0 pour désactiver
iconeBurger	string	'burger'	Identifiant du jeu d'icônes intégré
iconePosition	string	'gauche'	'gauche' | 'droite' — réglage global pour toute la sidebar
seuilMagnetisme	number	30	Distance en pixels déclenchant l'ancrage en mode libre
memoriserPosition	bool	false	Mémorise l'état (déployée/repliée, position en mode libre) entre deux visites
cle	string	null	Clé de stockage stable — nécessaire si plusieurs instances sur une page
entete	objet	null	{ logo, titre } — obligatoire en pratique dès que masquable=true, pointRupture actif, ou position='libre'
menus	array	[]	Arborescence des menus — voir section 5

Largeur automatique (`largeur: 'auto'`, défaut) : au chargement, la classe déploie temporairement tous les sous-menus, mesure la largeur naturelle du contenu le plus large — libellé, icône, décalage d'indentation compris —, puis restaure l'état d'origine sans flash visuel. Une valeur numérique explicite prime toujours sur ce calcul.

Les badges (voir section 5) comptent dans ce calcul : un badge large sur un libellé court peut élargir la sidebar plus que prévu. À retirer si ce n'est pas souhaité — la classe ne les exclut pas du calcul, ce serait recréer le problème inverse (texte et badge qui se chevauchent une fois déployée).

---

5. Structure d'un nœud de menu

La structure est récursive : un nœud est soit un conteneur (il a des `enfants`), soit une feuille cliquable (elle a un `chemin`) — jamais les deux à la fois. Un sous-menu ne peut pas lui-même contenir un autre sous-menu ; il peut en revanche contenir autant d'options qu'on veut.

Propriété	Type	Requis	Description
libelle	string	oui	Texte affiché
icone	string	non	Identifiant du jeu d'icônes intégré (voir section 6)

chemin	string	si feuille	URL de destination — présence = ce nœud est une feuille cliquable
droitMinimum	number	si feuille	Rôle maximal (1 à 5) autorisé à voir cette option
enfants	array	si conteneur	Nœuds enfants — présence = ce nœud est un conteneur (menu ou sous-menu)
badge	objet	non	{ valeur, couleur } — badge compteur coloré affiché à droite du libellé

⚠️ Le droit minimum ne se pose que sur les feuilles. Un menu ou sous-menu sans aucune feuille visible pour le rôle courant se masque automatiquement — inutile de dupliquer `droitMinimum` sur les conteneurs.

Le badge se masque automatiquement en mode rail (`masquable: false`, replié) — comme le libellé, il n'a pas la place de s'afficher à côté d'une simple icône. Il redevient visible dès que la sidebar est déployée. Il se masque aussi sous `pointRupture` (voir section 9) — l'espace y est compté au plus juste, le badge n'a pas plus de place qu'en mode rail.

---

6. Jeu d'icônes intégré

Le champ `icone` attend un identifiant, pas du SVG brut — la classe embarque son propre jeu d'icônes sobres, style outline. Liste complète accessible via `AcSidebar.listeIcones()` :

maison, grille, liste, burger, chevronBas, chevronDroite, fleche, utilisateur, groupe, badgeUser, bouclier, cadenas, cle, engrenage, oeil, deconnexion, document, dossier, livre, base, boite, telecharger, televerser, corbeille, crayon, filtre, recherche, graphique, camembert, tendance, calendrier, horloge, courrier, cloche, telephone, carte, localisation, batiment, image, lien, etoile, coeur, drapeau, info, alerte, coche, croix, plus, epingle.

Un identifiant inconnu déclenche un avertissement en console (liste des icônes disponibles) plutôt qu'un échec silencieux.

---

7. Thèmes et couleurs

Thème clair / sombre

Deux palettes toutes faites, sélectionnées via `theme: 'clair'` ou `theme: 'sombre'`.

Thème système

`theme: 'systeme'` ne fixe aucune couleur — la classe observe les couleurs réellement appliquées par la page hôte (fond et texte de `<body>`, couleur des liens) et en déduit une palette cohérente automatiquement. Zéro convention à documenter côté intégrateur. Si le fond de la page est transparent ou si aucun signal exploitable n'est trouvé, repli silencieux sur le thème clair.

Couleurs personnalisées

L'option `couleurs` accepte un objet avec jusqu'à 7 clés, chacune prioritaire sur le thème choisi :

fond, texte, texteDoux, accent, accentFond, survol, bordure.

```
couleurs: {
    accent: '#ff9f43',
    fond: '#20293a',
},
```

---

8. Positions et mode libre

Les 4 positions classiques (`gauche`, `droite`, `haut`, `bas`) ancrent la sidebar à un bord de l'écran. La position `haut`/`bas` reste une limite connue pour les hiérarchies profondes — l'espace horizontal restreint rend les sous-menus (affichés en popup) confortables seulement avec peu d'entrées et des libellés courts.

La position `libre` rend la sidebar déplaçable par glisser-déposer (poignée = zone logo+titre de l'en-tête). Elle flotte au-dessus du contenu (ombre portée, coins arrondis) tant qu'elle n'est pas proche d'un bord. Dès qu'elle passe à moins de `seuilMagnetisme` pixels d'un bord d'écran, elle s'y ancre et redevient une sidebar classique à part entière — bouton épingler et burger inclus.

Hors ancrage, une sidebar en position `libre` reste normalement toujours visible (le burger externe n'a alors aucune utilité) — sauf sous `pointRupture` (voir section 9) : elle s'y masque comme n'importe quelle autre position, et le burger redevient alors le seul moyen de la rouvrir.

---

9. Repli responsive (pointRupture)

Sous `pointRupture` pixels de largeur de fenêtre (768 par défaut), la sidebar se comporte comme `masquable=true` — repli complet, burger flottant pour la redéployer — quelle que soit la valeur configurée de `masquable` par ailleurs. Le survol est désactivé dans ce mode (pas de survol pertinent au tactile), et un bouton de repli apparaît dans l'en-tête pour le premier repli manuel.

Dans ce même mode, les positions `haut`/`bas` sont inhibées — une barre horizontale trop fine et des sous-menus en popup deviennent inutilisables au doigt — et la sidebar se replie visuellement sur `gauche`, sans jamais modifier la position configurée : celle-ci reprend la main dès la sortie du mode compact. Le bouton épingler est également désactivé — la sidebar n'est jamais fixée sur un écran où, une fois déployée, elle prendrait tout l'espace.

`pointRupture: 0` désactive complètement ce comportement. Le seuil se réévalue à chaque redimensionnement de fenêtre, pas seulement au chargement.

---

10. API publique

Méthode	Description
basculer()	Déploie ou replie la sidebar. Sans effet si épinglée ou si le survol pilote l'affichage.
deployer()	Force le déploiement.
replier()	Force le repli. Mêmes conditions que basculer().
basculerEpingle()	Active/désactive l'épinglage (déploiement permanent forcé).
etat()	Retourne une copie de l'état courant (déployée, épinglée, mode, bord, x, y).
oublierPosition()	Efface l'état mémorisé de cette instance dans localStorage.
destroy()	Retire la sidebar du DOM et débranche tous ses écouteurs.
AcSidebar.depuisJson(cible, url)	Fabrique asynchrone — charge la configuration depuis un fichier JSON.
AcSidebar.listeIcones()	Retourne la liste des identifiants d'icônes disponibles.

---

11. Poussée du contenu de la page (v1.1.0)

Quand la sidebar est épinglée (jamais en mode `libre`), elle pose automatiquement sur `<body>` une classe `ac-sidebar-fixee--{bord}` et une variable `--ac-sidebar-decalage` (la largeur réelle déployée pour `gauche`/`droite`, la hauteur pour `haut`/`bas`). C'est tout ce qu'elle fait : elle n'écrit aucune règle de mise en page elle-même — c'est à la page de décider comment utiliser ce point d'accroche.

```
body.ac-sidebar-fixee--gauche {
    margin-left: var(--ac-sidebar-decalage);
    transition: margin-left .2s ease;
}
```

Pourquoi ce n'est pas la classe qui pousse `body` directement : `body` n'est pas toujours la bonne cible (wrapper applicatif, grille CSS...), et un décalage automatique casserait les éléments `position:fixed`/`sticky` propres à la page (bandeau cookie, header collant...) ou les sections en `width:100vw`, sans que rien ne le signale. La classe ne peut pas deviner la bonne réponse à ta place — elle expose la donnée, la page choisit la technique.

Une seule instance à la fois détient ce droit sur toute la page — gérer plusieurs poussées simultanées serait ingérable. La première sidebar qui s'épingle l'acquiert et le garde jusqu'à ce qu'elle se désépingle ou soit détruite ; toute autre instance épinglée entre-temps reste en simple recouvrement, même si elle est bien épinglée. Au désépinglage, le droit n'est pas transféré automatiquement à une autre instance déjà épinglée — il faut qu'elle se désépingle puis se réépingle pour l'acquérir.

⚠️ Si la page n'a pas de règle CSS consommant `--ac-sidebar-decalage`, la classe avertit une seule fois en console (avec la règle exacte à ajouter) plutôt que d'échouer silencieusement.

---

12. Multi-instances

Plusieurs `ac_sidebar` peuvent coexister sur une même page, totalement indépendantes. Dans ce cas, préciser une `cle` distincte pour chaque instance — sinon la clé de stockage retombe sur l'id du conteneur, ou un identifiant auto-incrémenté instable d'un chargement de page à l'autre (à éviter si `memoriserPosition` est actif).

---

13. Sécurité

⚠️ Le JSON ne pilote que l'affichage. Chaque page listée dans un `chemin` doit continuer à vérifier elle-même le droit d'accès côté serveur — la sidebar ne remplace jamais ce contrôle, elle l'accompagne visuellement.

---

14. Bonnes pratiques

- Toujours vérifier côté serveur le droit d'accès à chaque page listée dans un `chemin` — le filtrage de la sidebar n'est qu'un affichage.
- Préciser `cle` explicitement dès qu'il y a plus d'une instance sur une page, ou que `memoriserPosition` est actif.
- Poser `droitMinimum` uniquement sur les feuilles — jamais sur les conteneurs, le masquage automatique s'en charge.
- Fournir un `entete` (au moins un titre) dès que `masquable=true`, `pointRupture` actif, ou `position='libre'` — sans lui, le bouton de repli interne ou la poignée de préhension n'ont nulle part où exister.
- Pour un thème qui s'intègre à un site existant, essayer `theme: 'systeme'` avant de personnaliser manuellement les couleurs.
- Retirer les badges des libellés si la largeur automatique doit rester compacte — ils comptent dans le calcul.
- Écrire la règle CSS `body.ac-sidebar-fixee--{bord}` dès que l'épinglage est utilisé — sans elle, un avertissement console le rappelle, mais rien ne pousse la page.