ac_signature.js

Documentation technique v1.0

Zone de signature tactile / souris — Export base64 + PNG

artisan-code.fr · Jean-Noël Vidallet · 2026

1. Présentation

ac_signature.js est une classe JavaScript vanilla autonome permettant d'intégrer une zone de signature tactile et souris dans n'importe quelle page web. Elle produit une image PNG encodée en base64, utilisable pour le stockage en base de données ou pour la sauvegarde sous forme de fichier.

La classe est propriétaire (artisan-code.fr) et fait partie de la Classothèque.

1.1 Caractéristiques principales

- Zéro dépendance externe — JavaScript vanilla pur
- Canvas HTML5 responsive — suit la largeur du conteneur
- Compatible souris et tactile (tablette, smartphone)
- Trait lisse grâce aux options round cap/join
- Export base64 PNG pour stockage en BDD
- Téléchargement PNG direct depuis le navigateur
- Miniature de la signature enregistrée
- Mode lecture seule — affichage sans édition
- Convention Classothèque : getValeur() / setValeur()
- Deux événements JavaScript custom

1.2 Philosophie

ac_signature.js ne fait pas de CRUD. Elle produit une base64 et émet un événement — c'est la page appelante qui décide quoi en faire : stocker en BDD, sauvegarder en fichier PNG sur le serveur, ou les deux. La classe ne connaît ni le contexte métier, ni la base de données.

> *Cette séparation des responsabilités est la philosophie centrale de toute la Classothèque.*

2. Intégration

2.1 Inclusion

```
<script src="ac_signature.js"></script>
```

2.2 Usage minimal

```
<input type="hidden" id="ma-signature">

<script>
  var widget = new ac_signature('ma-signature');
</script>
```

> *L'input d'origine est automatiquement masqué. Le widget est inséré juste après dans le DOM.*

2.3 Usage complet avec options

```
var widget = new ac_signature('ma-signature', {
    hauteur       : '200px',
    couleur_trait : '#111111',
    epaisseur     : 2,
    lecture_seule : false,
    nom_fichier   : 'signature-devis',
});
```

3. Options de configuration

Propriété	Type	Défaut	Description
`hauteur`	string	`'160px'`	Hauteur du canvas de signature. La largeur est toujours 100% du conteneur parent.
`couleur_trait`	string	`'#111111'`	Couleur du trait de signature. Toute valeur CSS valide (hex, rgb, nom).
`epaisseur`	number	`2`	Épaisseur du trait en pixels. Valeurs recommandées : 1.5 à 3.
`lecture_seule`	bool	`false`	Si true, le canvas est en lecture seule : aucun dessin possible, boutons Effacer et Valider masqués. Utile pour l'affichage d'une signature existante.
`nom_fichier`	string	`'signature'`	Nom du fichier PNG généré lors du téléchargement (sans extension).

4. Méthodes publiques

Méthode	Retour	Description
`getValeur()`	`string`	Retourne la base64 PNG de la dernière signature validée, ou une chaîne vide si aucune signature n'a été validée.
`getValue()`	`string`	Alias de getValeur() — convention Classothèque.
`setValeur(b64)`	`void`	Charge une base64 existante (issue de la BDD par exemple), l'affiche dans le canvas et dans la miniature. Passer une chaîne vide ou null est équivalent à vider().
`setValue(v)`	`void`	Alias de setValeur() — convention Classothèque.
`vider()`	`void`	Efface le canvas, réinitialise la valeur et masque la miniature. Émet l'événement ac_signature:change avec vide:true.
`estVide()`	`bool`	Retourne true si aucun trait n'a été dessiné sur le canvas depuis la dernière réinitialisation. Ne tient pas compte de la valeur validée.
`telecharger()`	`void`	Déclenche le téléchargement du PNG dans le navigateur. Utilise le canvas courant si un trait est présent, sinon la dernière base64 validée.

> *getValeur() retourne la base64 uniquement après un clic sur Valider — pas à chaque trait. Ceci évite de surcharger la BDD avec des données partielles.*

5. Événements JavaScript

Les événements sont dispatchés sur l'élément input d'origine avec bubbles:true.

Événement	detail	Description
`ac_signature:valider`	`{ base64, vide }`	Émis lors du clic sur le bouton Valider. base64 contient le PNG encodé. vide vaut toujours false à ce moment.
`ac_signature:change`	`{ base64, vide }`	Émis lors de la validation (vide:false) et lors de l'effacement (vide:true, base64:'').

Exemple d'écoute :

```
document.getElementById('ma-signature').addEventListener(
    'ac_signature:valider',
    function(e) {
        console.log('base64 :', e.detail.base64.substring(0, 40));
        console.log('vide    :', e.detail.vide);
    }
);
```

6. Flux d'intégration — Signature d'un devis

Voici le flux complet pour intégrer ac_signature.js dans un contexte de signature de devis, avec stockage en BDD et sauvegarde du fichier PNG sur le serveur.

6.1 Page PHP — affichage du devis

```
<input type="hidden" id="signature" value="<?=
htmlspecialchars($devis['signature'] ?? '') ?>">

<script>
var widget = new ac_signature('signature', {
    hauteur     : '180px',
    nom_fichier : 'devis-' + <?= $devis['id'] ?>,
    // lecture_seule si déjà signé
    lecture_seule : <?= $devis['signature'] ? 'true' : 'false' ?>,
});

// Si signature existante en BDD, la charger
<?php if ($devis['signature']): ?>
    widget.setValeur(<?= json_encode($devis['signature']) ?>);
<?php endif; ?>

// Écouter la validation
document.getElementById('signature').addEventListener(
    'ac_signature:valider', function(e) {
        var fd = new FormData();
        fd.append('devis_id',     <?= $devis['id'] ?>);
        fd.append('signature_b64', e.detail.base64);
        fetch('sauver_signature.php', { method: 'POST', body: fd })
            .then(r => r.json())
            .then(rep => { if (rep.ok) alert('Signature enregistrée.'); });
    }
);
</script>
```

6.2 Fichier PHP — sauver_signature.php

```
<?php
header('Content-Type: application/json');

$devisId = (int)($_POST['devis_id'] ?? 0);
$b64     = $_POST['signature_b64'] ?? '';

if (!$devisId || !$b64) {
    echo json_encode(['ok' => false, 'msg' => 'Données manquantes']); exit;
}

// 1. Stocker la base64 en BDD
$pdo->prepare('UPDATE devis SET signature = ?, date_signature = NOW()
               WHERE id = ?')->execute([$b64, $devisId]);

// 2. Sauvegarder le fichier PNG sur le serveur
$data = base64_decode(preg_replace('#^#', '', $b64));
file_put_contents('signatures/devis_' . $devisId . '.png', $data);

echo json_encode(['ok' => true]);
```

> *La classe ne fait aucun appel réseau. C'est toujours la page appelante qui gère la persistance.*

7. Interface utilisateur

7.1 Éléments affichés

- Canvas de dessin avec ligne de base et texte guide "Signez ici" (disparaît au premier trait)
- Bouton Effacer — grisé si canvas vide
- Bouton ↓ PNG — télécharge le PNG courant ou la dernière base64 validée
- Bouton Valider — grisé si canvas vide, déclenche la sauvegarde
- Miniature — affichée après validation ou après setValeur(), masquée après vider()

7.2 Mode lecture seule

En mode lecture_seule:true, les boutons Effacer et Valider sont masqués. Seul le bouton ↓ PNG reste visible. Le canvas n'accepte aucune saisie. Ce mode est utilisé pour afficher une signature déjà enregistrée.

7.3 Comportement tactile

Sur tablette et smartphone, touch-action:none est appliqué sur le canvas pour empêcher le défilement de la page pendant la signature. Les événements touchstart, touchmove et touchend sont tous gérés avec preventDefault().

8. Bonnes pratiques

- Utiliser un input type="hidden" — la valeur base64 n'a pas vocation à être affichée ni saisie manuellement.
- Vérifier estVide() avant de soumettre un formulaire pour s'assurer que le client a bien signé.
- En mode lecture seule, toujours appeler setValeur() après l'instanciation pour charger la signature existante.
- Stocker la base64 en BDD dans un champ MEDIUMTEXT ou LONGTEXT — une base64 PNG fait typiquement entre 5 000 et 50 000 caractères selon la complexité de la signature.
- Sauvegarder aussi le PNG sur le serveur — la base64 permet de reconstruire l'image, mais un fichier PNG est plus pratique pour l'impression ou l'envoi par email.
- Préférer un chemin de stockage structuré : signatures/2026/03/devis_42.png.

— Fin de la documentation —