



PatchNotes : Écrire pour exister

Dans ce TechTalk, nous allons explorer un outil de communication souvent négligé, mais pourtant essentiel dans le développement logiciel : les PatchNotes. Nous verrons à qui ils s'adressent, pourquoi ils sont indispensables, quand et comment les rédiger, et comment les structurer de manière professionnelle.

 par Stephan M

À Qui S'adresse un PatchNote ?



Développeurs

Internes, contributeurs externes et partenaires techniques ont besoin de comprendre les évolutions du code pour une meilleure collaboration.



QA / Testeurs

Pour identifier précisément ce qui doit être testé, garantissant ainsi la qualité des nouvelles versions.



Support Client

Afin d'anticiper les demandes des utilisateurs et de répondre efficacement aux questions ou aux bugs signalés.



Product Owners / Managers

Pour suivre l'évolution du produit, valider les fonctionnalités et aligner les équipes sur la roadmap.



Utilisateurs Finaux

Une version simplifiée des PatchNotes les informe des nouveautés et améliorations, augmentant leur satisfaction.

Pourquoi Rédiger des PatchNotes ?



Transparence et Alignement

Les PatchNotes assurent une transparence sur l'évolution du produit, évitant régressions et incompréhensions au sein de l'équipe.



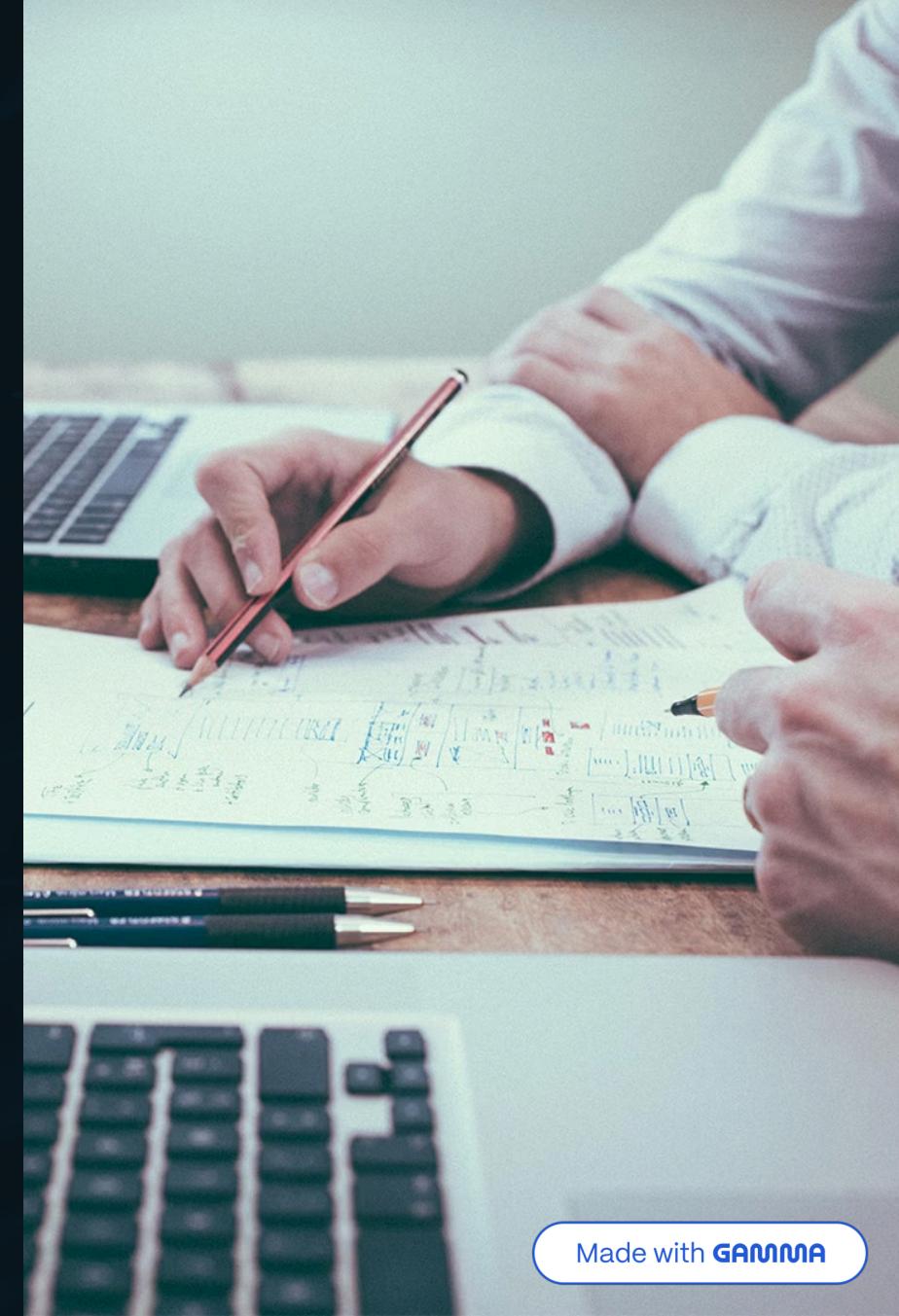
Historique Précis

Ils documentent chaque changement de version, facilitant les migrations et servant de référence précieuse pour le débogage.



Valorisation des Mises à Jour

Les PatchNotes prouvent la valeur ajoutée des mises à jour aux utilisateurs et clients, renforçant leur confiance.



Structurer Efficacement un PatchNote

Versioning Sémantique

Adoptez le versioning sémantique (ex: 1.2.3) pour indiquer clairement la nature des changements (Majeur.Mineur.Patch).

Catégorisation des Changements

- **[Added]** : Nouvelles fonctionnalités
- **[Changed]** : Changements de comportement
- **[Fixed]** : Bugs corrigés
- **[Removed]** : Suppression de fonction
- **[Security]** : Correctifs de sécurité
- **[Deprecated]** : Fonctionnalité à retirer prochainement

Exemple de Structure

Version 2.1.0 – 2025-06-25

[Added] :

- Ajout d'une API /users/stats

[Changed] :

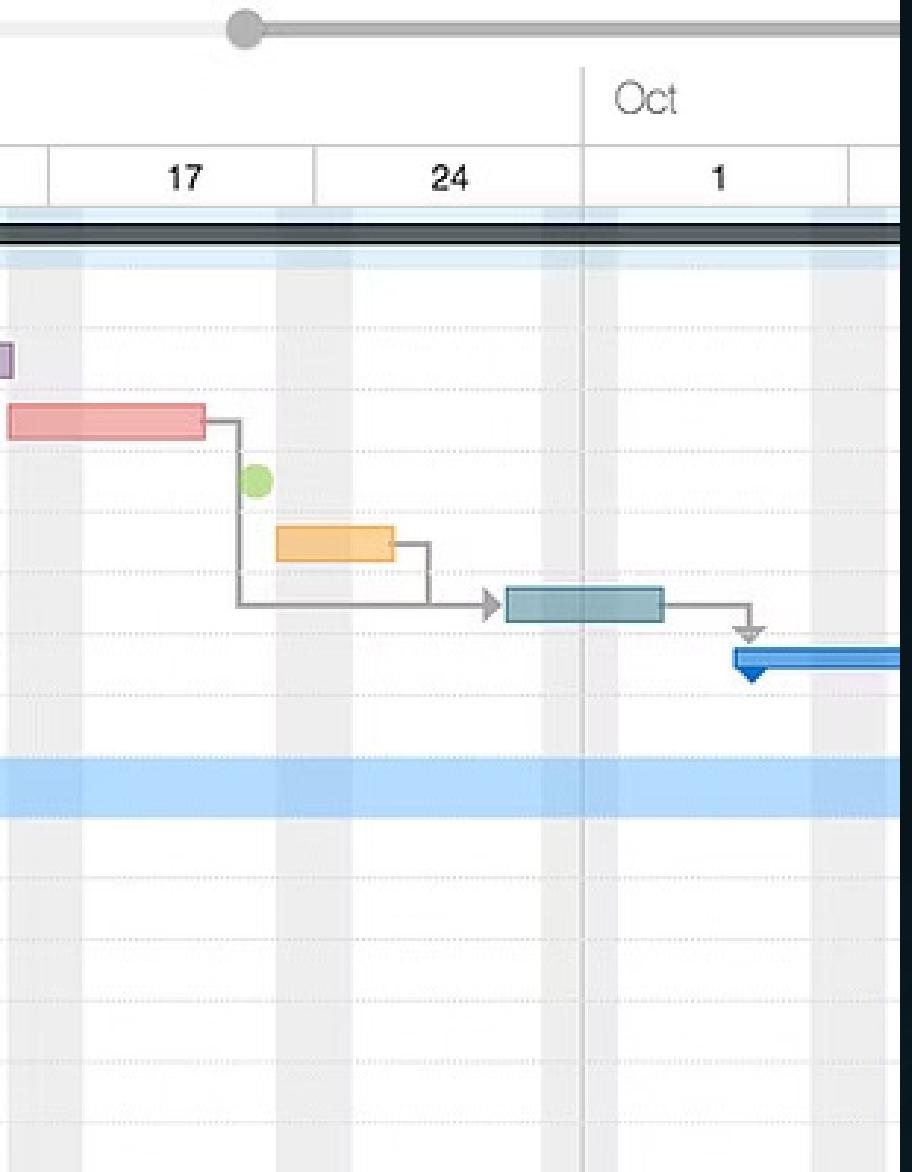
- Changement du champ email en champ obligatoire

[Fixed] :

- Résolution d'un bug d'affichage sur mobile

[Security] :

- Mise à jour de la lib JWT (faille critique corrigée)



Quand Rédiger un PatchNote ?

1

Mise en Production (MEP)

Chaque déploiement en production, qu'il s'agisse d'une nouvelle version majeure ou mineure, requiert des PatchNotes détaillés.

2

Hotfix Important

Un correctif d'urgence pour un bug critique doit être accompagné d'un PatchNote pour informer rapidement les parties prenantes.

3

Ajout/Modif/Retrait

Toute modification significative de fonctionnalité ou de comportement du produit doit être documentée.

4

Modification de Dépendance/API

Les changements dans les dépendances critiques ou les APIs impactent souvent les développeurs et nécessitent une documentation précise.

Règle simple : si quelque chose change dans l'expérience utilisateur ou dans le comportement du produit → un PatchNote est attendu.

Outils et Plateformes de Gestion des PatchNotes



Keep a Changelog & CHANGELOG.md

Un format recommandé pour un fichier `CHANGELOG.md`, placé à la racine de votre dépôt, pour une traçabilité simple et efficace.



GitHub Releases

Utilisez les fonctionnalités de GitHub pour associer le texte des PatchNotes à un tag de version, facilitant l'accès et la publication.



Génération Automatique

Des outils comme `semantic-release` ou `standard-version` peuvent générer des changelogs directement depuis les messages de commit, assurant cohérence et gain de temps.



Intégration Documentaire

Intégrez vos PatchNotes dans des plateformes comme Notion, Confluence ou vos documentations publiques pour une diffusion large et centralisée.

5.2.0 Changelog

Breaking Changes	Bugs	Released
0 Changes	31 Bug Fixes	May 06, 2021

NPM Packages
 [@clr/ui](#)
[@clr/angular](#)
[@clr/icons](#)

Sketch template
 [Download Light](#)
[Theme from Github](#)
[Download Dark](#)
[Theme from Github](#)

Highlights

- Added standalone pagination component to clarity core (#5859)
- Added a 'neutral' button style to cds-buttons (#5859)
- Added signature, IoT, thin client, timeline, and update icons (#5860)
- Added 'xs' size for cds-icon (#5872)
- Added ng-add capability to @cds/angular (#5890)
- Added cds-i18n documentation to Clarity Core storybook (#5901)

Bug Fixes

- Fixed several issues with Core accordion CSS, including alignment (#5902)
- Fixed issue with content overflow in cds-accordion-header (#5903)
- Lightened the border, text, and icon colors of disabled inputs (#5903)
- Updated file input so that the clear-all control is removed (#5903)
- Allowed focus with the tab key for focusable content inside accordions (#5903)
- Disabled the cds-toggle animation until after its initial load (#5903)
- Changed cds-text=truncate so it would not cut off glyphs (#5903)
- Updated Angular to 11.2.5 (#5748)
- Fixed dark theme configuration in @clr/angular dev app (#5810)
- Updated clr-buttons so that cds-icons could rotate within them (#5810)
- Prevent combobox from firing double open event with cds-open (#5810)
- Corrected the alignment of the selected count checkbox (#5810)
- Keep invalid password input from changing its width unexpectedly (#5810)
- Fixed clr-stepper so that it would run validation after a user interaction (#5810)
- Fixed animation of accordion header toggles (#5810)
- Updated Clarity UI HSL values to remove deprecated warning (#5810)
- Fixed sizing of Clarity Icons inside Clarity UI's compact buttons (#5810)
- Added CSS variables to nav styles to improve theming (#5810)
- Fixed the alignment of the collapse/expand icon in nav (#5810)
- Added missing CSS properties to enable theming of clr-select (#5810)
- Updated Clarity UI dark theme to use rem-sizing variables (#5810)
- Improved @cds/angular (#5810)
- Made with **GAMMA** ([View on GitHub](#))
- Fixed unit tests after ng-add was updated (#5878)

Bonnes Pratiques de Rédaction

1 Clarté et Concision

Rédigez des phrases claires, courtes et faciles à comprendre, évitant le jargon technique excessif.

2 Pertinence de l'Information

Ne noyez pas l'information essentielle dans des détails techniques superflus. Concentrez-vous sur ce qui est pertinent pour le lecteur.

3 Séparation des Changements

Si nécessaire, séparez les changements fonctionnels des changements techniques pour mieux cibler les différents publics.

4 Régularité et Cohérence

Maintenez une publication régulière des PatchNotes. Un changelog inactif peut donner une impression d'abandon du projet.

5 Adaptation au Public

Adaptez le ton, le niveau de détail et la complexité de l'information en fonction du public ciblé (développeurs, utilisateurs finaux, etc.).

FREE PROFESSIONAL DOCUMENT TEMPLATE

Fully editable and customizable, this template offers unparalleled flexibility to meet all your professional needs.



Available in **MS Word**

Conclusion : La Valeur des PatchNotes

Les PatchNotes sont bien plus qu'un simple fichier texte ; ce sont des outils de communication essentiels qui transcendent la simple information technique. En tant que boussoles de l'évolution logicielle, ils **gardent votre équipe alignée, informent vos utilisateurs des nouveautés, et valorisent le travail acharné de votre équipe.**

Bien structurés et rédigés avec soin, ils reflètent le **professionnalisme de votre projet**, facilitent la collaboration entre les différentes parties prenantes et apportent une **clarté indispensable** à l'ensemble de votre processus de livraison. Intégrez-les pleinement dans votre cycle de développement pour maximiser leur impact.

[En savoir plus sur Keep a Changelog](#)