Kit de documentation

Documentation — Étendre le Kit — FR

1 Objet

Le Kit couvre un ensemble fini de présentations et de types de document. Un projet rencontre tôt ou tard un besoin qui n'y figure pas: une mise en forme absente, un type de bloc particulier, une règle propre à son domaine.

Deux voies existent. Modifier les fichiers du Kit répond au besoin immédiatement, et impose à la première mise à jour de choisir entre l'adaptation et la nouvelle version. Ajouter ses propres fichiers à côté demande un peu plus de travail au départ et laisse les deux coexister.

Ce document décrit la seconde voie: où placer vos fichiers, comment les nommer, comment un générateur les combine avec ceux du Kit, et ce qui casse quand le Kit évolue.

2 Ce qui appartient au Kit et ce qui vous appartient

La frontière tient à un préfixe. Tout fichier dont le nom commence par "Kit -" ou "kit_" vient du Kit et sera remplacé à la prochaine mise à jour. Vos fichiers portent l'un de deux autres préfixes, selon leur portée.

- **Kit-X —, pour vos extensions du Kit:** un *stylesheet* supplémentaire, une palette de couleurs, un outil à vous. Ces fichiers valent pour tous vos projets et vous suivent de l'un à l'autre sans être renommés. Le X est pour extension.
- **[Préfixe] —, pour ce qui ne vaut que pour un projet:** ses documents, son *Registry*, ses *modules de données*, ses artefacts exécutables. La liste complète est à *Structure commune* §2.

Note: *Vos fichiers Kit-X n'existent qu'en un seul exemplaire. Sauvegardez-les régulièrement.*

- **Fichiers du Kit, remplacés en bloc:** les quatre stylesheets et leurs Reference, les artefacts exécutables kit_*.py et kit_*.js, Kit - *Registry.js*, et les documents du Kit lui-même. La procédure de remplacement est décrite dans le *Guide de mise à jour*.
- **Fichiers du projet, jamais touchés:** votre *Registry*, votre *Prompt*, vos documents, vos *modules de données*, vos artefacts exécutables — *Structure commune* §12 — et les stylesheets que ce document vous apprend à écrire.

Note: *Un fichier du Kit modifié sur place perd sa modification à la mise à jour suivante, sans avertissement: le remplacement est un écrasement. C'est la raison pour laquelle l'extension passe par l'ajout.*

3 Remplacer les couleurs

Les couleurs des stylesheets du Kit sont des défauts, et trois niveaux se superposent. Kit-X - Colors.js porte celles qui vous suivent d'un projet à l'autre. [Préfixe] - Colors.js porte celles qui ne valent que pour un projet, et prime sur les précédentes. Chaque fichier ne déclare que les clés qu'il change; les autres gardent la valeur du niveau au-dessus.

Le fichier du projet est piloté par le drapeau requires.colors de son *Registry*: posez-le à true, sans quoi le générateur ne le chargera pas. Celui de Kit-X n'est piloté par aucun drapeau — il est présent ou il ne l'est pas. Chaque fichier exporte colorEntries, objet à deux tables: docx pour les documents, html pour les pages web. Les séparer permet de régler le contraste sur papier imprimé indépendamment de celui de l'écran. Les noms de clés et leurs défauts sont tabulés dans les Reference des deux stylesheets, et le contrat du fichier à *Structure commune* §6.5.

Le Kit ne contrôle ni le contraste ni l'harmonie: une couleur illisible produit un document valide et pénible à lire. Une valeur qui n'est pas une couleur arrête en revanche la *génération*, avec le nom de la clé fautive.

```
// Kit-X - Colors (2026-08-23 - 17h00).js
const colorEntries = {
  docx: { HEADING_COLOR: '8B0000' },
  html: { HEADING_COLOR: '#8B0000', CODE_BG: '#FAFAFA' },
};
module.exports = { colorEntries };
```

Le générateur charge les deux niveaux et les applique dans l'ordre. Le *stylesheet* n'importe ni le *Registry* ni ces fichiers: le générateur fait le pont, comme pour la langue et le glossaire.

```
// setColors fusionne : deux appels successifs cascadent.
if (kitX)     style.setColors(kitX.docx);
if (couleurs) style.setColors(couleurs.docx);
```

4 Écrire un stylesheet de projet

Un *stylesheet* de projet exporte des fonctions qui produisent des blocs, exactement comme ceux du Kit. Il complète le *stylesheet* General plutôt que de le remplacer: votre générateur importe les deux et appelle l'un ou l'autre selon le bloc à produire.

4.1 Nommage et couplet

Le nom suit la convention des stylesheets du Kit, avec le préfixe qui correspond à la portée du fichier: "Kit-X - *Stylesheet* - [Sujet] - Code.js" pour un *stylesheet* qui vous suit d'un projet à l'autre, "[Préfixe] - *Stylesheet* - [Sujet] - Code.js" pour un *stylesheet* propre à un seul projet. Kit-X convient à la plupart des cas: une présentation utile une fois l'est généralement ailleurs.

Le .docx de Reference porte le même préfixe, le même sujet et le même *horodatage* que son .js. Les deux se livrent ensemble — la règle du *couplet*, *Prompt* §6, s'applique à vos stylesheets comme à ceux du Kit.

La Reference documente ce que le code seul ne dit pas: les valeurs de formatage, les contrats de retour, les enchaînements obligatoires. Elle se régénère à chaque évolution du .js, jamais reconstruite depuis le code seul.

4.2 Ce que votre stylesheet importe

Les constantes de géométrie du Kit — indentations, largeurs de table, polices, couleurs — appartiennent au *stylesheet* General et s'y lisent. Les recopier

chez vous crée une divergence qui ne se voit qu'au premier changement de géométrie.

```
const style = require('./Kit - Stylesheet - General - Code.js');

// La geometrie vient de General, jamais d'une copie locale.
const { INDENT_L1, INDENT_L2, TABLE_WIDTH_L1, FONT, BODY_TEXT_COLOR } = style;
```

Note: *Le stylesheet YAML du Kit fait exception et n'importe rien: il doit rester utilisable seul. Ses valeurs de géométrie sont donc recopiées et documentées comme telles. Cette dépendance manuelle a laissé une divergence de niveau 3 dormir plusieurs mois — l'importation est le choix par défaut.*

4.3 Ce que votre stylesheet exporte

Chaque fonction rend soit un Paragraph, soit un tableau d'éléments, comme les fonctions du Kit. Le contrat de retour se documente dans votre Reference: ce que la fonction rend, ce qui doit la précéder, ce qui doit la suivre.

Deux règles du Kit s'appliquent à vos fonctions. Le nommage est explicite, sans abréviation ni sigle à décoder — *Prompt* §12. Et toute chaîne affichée passe par une table de *localisation*, même si votre projet ne parle aujourd'hui qu'une langue — *Prompt* §4.1.

5 Déclarer votre stylesheet

Un *stylesheet* Kit-X vous suit d'un projet à l'autre; sa version appartient donc à Kit-X - *Registry.js*, votre registre à vous, et non au *Registry* d'un projet en particulier. Créez ce fichier quand vous écrivez votre premier *stylesheet*.

```
// Kit-X - Registry.js
module.exports = {
  stylesheets: {
    planches: { version: '1.00', ts: '2026-08-23 - 14h00' },
  },
};
```

Vos générateurs y lisent la version active plutôt que de la porter en dur. Trois registres coexistent alors, chacun avec son domaine: celui du Kit pour les stylesheets du Kit, celui de Kit-X pour les vôtres, celui du projet pour ses documents et ses horodatages.

6 Écrire un générateur qui combine les deux

Un générateur importe le *stylesheet* du Kit et le vôtre, puis appelle chacun selon le bloc à produire. L'ordre des appels détermine l'ordre du document, comme pour un générateur ordinaire.

```
const style    = require('./Kit - Stylesheet - General - Code.js');
const planches = require('./Kit-X - Stylesheet - Planches - Code.js');
const registry = require('./Monprojet - Registry.js');

style.setLanguage(registry.project.docLanguage);
planches.setLanguage(registry.project.docLanguage);
```

```
const corps = [
  style.h1('1', 'Montage'),
  style.paragraph('Le montage se fait en trois passes.'),
  planches.schema(image, 'Vue eclatee'),   // votre fonction a vous
  style.releaseParagraph(),
];
```

Le niveau de section est tenu par le *stylesheet* General, posé par h1, h2 et h3 et lu par les autres fonctions. Un *stylesheet* de projet qui a besoin du niveau courant le lit par style.currentLevel() et ne l'écrit jamais: trois fonctions seulement le posent, et en ajouter une quatrième rouvre l'erreur de concordance que ce dispositif ferme.

7 Ce que la validation couvre

Les artefacts de contrôle du Kit s'appliquent à vos documents sans adaptation. Ils lisent le fichier produit, pas le code qui l'a produit.

- **kit_validate_docx.js:** vérifie la structure du .docx produit — niveaux, enchaînements, *empreinte* de version. Il attend l'*empreinte* du *stylesheet* General, que vos documents portent puisque c'est lui qui les construit.
- **kit_check_setters.py:** seule exception à la phrase ci-dessus: il lit le générateur, pas le document. Il vérifie que les *setters* requis par les drapeaux du *Registry* sont bien appelés en tête — un *setter* omis désactive silencieusement le rendu correspondant. Il s'applique donc au générateur que ce document vous apprend à écrire.
- **kit_check_markup.py:** compare le marquage entre un binaire source et le document régénéré. Utile dès que vous reprenez un document existant.
- **kit_check_ossature.py:** compare le squelette de deux variantes de langue.
- **kit_extract_map.py et kit_check_fidelite.py:** l'extraction reconnaît les blocs du Kit. Un bloc produit par votre *stylesheet* lui est inconnu et ressortira dégradé. Le contrôle de fidélité vous le dira — c'est sa fonction. Deux réponses possibles: étendre l'extraction dans votre propre copie de travail, ou remonter le besoin au Kit.

Note: *L'extraction fait partie du Kit et suit ses mises à jour. Une extension locale sera donc écrasée. Un type de bloc appelé à durer se remonte au Kit plutôt qu'il ne se maintient à côté — Structure commune §13 décrit la voie.*

8 Ce qui casse quand le Kit évolue

Le Kit n'assure aucune compatibilité, ni ascendante ni descendante. Une évolution peut renommer une fonction, changer une signature ou retirer un helper. Votre *stylesheet*, qui appelle celui du Kit, en subit les conséquences.

- **Un nom de fonction change:** vos appels échouent bruyamment au premier lancement. C'est le cas le plus confortable — l'erreur est immédiate et localisée.

- **Une valeur de géométrie change:** vos blocs se décalent sans que rien n'échoue, si vous avez recopié la valeur au lieu de l'importer. C'est la raison de la règle du §4.2.
- **Un contrat d'enchaînement change:** un *successeur* devenu obligatoire produit un document mal formé que le *validateur* signale. Lisez la Reference du *stylesheet* après chaque mise à jour.

La session d'absorption d'une mise à jour, décrite dans le *Guide de mise à jour*, est le moment où ces écarts se traitent. Vos stylesheets y sont relus au même titre que vos générateurs.

9 Créer des modèles hors Kit

Un projet dont la matière l'exige — une recette, une fiche de terrain, un format de page étranger à la documentation — peut créer ses propres modèles, hors des formes du Kit. Le droit se déclare au *Registry* par allowNonKitTemplates, à false par défaut. C'est une autorisation écrite, non un verrou: aucun contrôle ne l'applique, et sans elle un projet ne crée pas de *modèle hors Kit*.

Le droit porte sur les formes, jamais sur les règles. Le projet garde les mêmes réflexes: fichier source d'abord, *Registry* qui fait foi, *horodatage* frais, numéro de version qui monte dès qu'un fichier change, générateur réécrit à neuf, *couplets* accordés, chaîne de validation avant toute livraison, aucune perte de contenu, zéro formatage de mémoire, décision écrite dans les documents en même temps qu'elle s'applique au code. Ce qui demanderait de changer une règle commune remonte au Kit — *Structure commune* §13 — plutôt que de se corriger sur place.

Ce que le Kit cesse de garantir: il ne répond plus du rendu produit par un *modèle hors Kit*, ni de sa tenue lors d'une évolution de *feuille de style*. Il continue d'exiger les réflexes, la chaîne de validation et l'identité du *sous-site*.

10 Quand remonter le besoin au Kit

Une extension locale convient à un besoin propre à votre domaine. Un besoin que d'autres projets rencontreraient a sa place dans le Kit, où il sera maintenu et contrôlé.

- Le besoin se formule sans nommer votre domaine: il est général.
- Vous avez écrit deux fois la même fonction pour deux documents différents.
- Votre *stylesheet* recopie une valeur de géométrie du Kit parce qu'aucune fonction ne l'expose.
- L'extraction ne reconnaît pas votre bloc et vous maintenez une copie locale de l'outil.

La voie de remontée est décrite dans *Structure commune* §13. Un besoin remonté et accepté devient une fonction du Kit, et votre extension locale disparaît au profit de celle-ci.