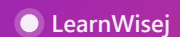


Exercice de code avec l'IA

Construisez l'exemple AdaptiveOps de ce module avec ChatGPT ou Claude à partir d'un prompt prêt à l'emploi qui oriente le modèle vers la spécification du module, puis relisez, exécutez et étendez ce qu'il vous renvoie — y compris en le comparant à notre propre solution de référence sur GitHub.



L'assistant, ce sont les mains rapides. L'architecte, c'est toujours vous.

Un assistant peut produire l'exemple de ce module en quelques minutes. Ce qu'il ne peut pas faire, c'est décider si le résultat est la bonne conception pour votre application — cette décision vous revient. C'est vous qui apportez le jugement : briefez-le correctement, puis vérifiez ce qu'il vous renvoie à la lumière de ce que le Module 2 vient de vous apprendre.

Ce que vous demandez

Créez le thème AdaptiveOps à partir d'un thème de base intégré dans le Theme Builder. Définissez une palette de couleurs nommées avec brandPrimary, brandAccent, surface, surfaceAlt, textMain, textMuted, danger, warning, success et focusFrame ainsi que les polices default, heading et mono, puis restylez les appearances du bouton, du panneau, de l'onglet, de l'en-tête de grille, de l'éditeur invalide et de l'infobulle à l'aide de ces tokens. Ajoutez une appearance action-button qui hérite de button et ne redéfinit que ses états default, hovered et pressed, vérifiez chaque modification dans l'aperçu du Theme Builder et chargez le thème dans Adaptive Operations Console.

Le module demande ces livrables, et le prompt ci-dessous demande exactement ceux-là à l'assistant :

- ✓ AdaptiveOps.theme créé à partir d'un thème de base et chargé par l'application
- ✓ Tokens de couleurs nommés et tokens de polices default, heading et mono définis une seule fois et réutilisés
- ✓ Appearances du bouton, du panneau, de l'onglet, de l'en-tête de grille, de l'éditeur invalide et de l'infobulle restylées à partir des tokens
- ✓ Appearance action-button héritant de button avec les états default, hovered et pressed redéfinis
- ✓ Captures de l'aperçu du Theme Builder et note sur les choix d'ordre des états et d'héritage

Pourquoi il est conçu ainsi

Le Module 1 a donné à Adaptive Operations Console un shell ancré, mais le thème de base porte encore chaque couleur et chaque police. Le raccourci consiste à appliquer la marque contrôle par contrôle avec BackColor, Font, des chaînes CssStyle ou du CSS visant le DOM interne des widgets. Chacune de ces approches disperse une valeur hexadécimale en des dizaines d'endroits, combat le moteur de thème qui rend déjà chaque état, et casse à la mise à jour suivante. Un thème est un unique fichier JSON qui porte

l'identité des contrôles standard; le Theme Builder est l'outil qui permet de le produire et de le maintenir, et ce module s'en sert pour construire le thème AdaptiveOps.

Voici les termes que l'assistant doit maîtriser. Si ce qu'il vous renvoie les emploie de façon approximative, il n'a pas compris le module :

Terme	Ce qu'il signifie ici
thème de base	Le thème intégré que le Theme Builder copie pour créer le vôtre, si bien que des centaines d'apparences, composants, états, métriques, images et polices existent avant que vous ne changiez la moindre valeur.
tokens nommés de couleurs et de polices	Des entrées des sections colors et fonts (brandPrimary, danger, heading) que les apparences référencent par leur nom, pour qu'un changement de marque se fasse en une seule modification.
appearance et AppearanceKey	Une appearance est le bloc du thème pour un type de widget; l' AppearanceKey d'un contrôle choisit le bloc qui s'applique, et c'est ainsi qu'existe une variante sémantique comme un bouton d'action.
héritage des apparences	Une entrée inherit permet à une appearance spécialisée de réutiliser une appearance de base et de ne redéfinir que quelques états, si bien qu' action-button reste un vrai bouton pour tout le reste.
ordre des états	Les états comme default, hovered, pressed, focused, disabled et invalid s'appliquent dans l'ordre de la liste; en déplacer un change les styles qui l'emportent, l'ordre est donc un comportement.
composant enfant	Une partie interne d'un widget, comme la barre de titre d'un panneau, un en-tête de colonne de grille ou le bouton d'un éditeur, avec son propre chemin d'apparence et ses propres états.
styles et propriétés	Les styles sont des valeurs de type CSS que le thème transforme en classes CSS dynamiques et qu'il possède; les propriétés sont des propriétés du widget qui peuvent devenir du style inline ou un comportement, et que l'application peut redéfinir.

Construisez-le — choisissez votre assistant

Les deux boutons ouvrent un nouvel onglet avec le **même prompt prêt à l'emploi** déjà rempli. Le prompt oriente le modèle vers la spécification de ce module — un unique fichier Markdown qui contient la leçon, l'énoncé du lab et les livrables — pour qu'il travaille sur le vrai contenu au lieu de deviner de quoi parle le Module 2. Vous pouvez ouvrir vous-même [module-02.md](#) pour voir exactement ce qu'il lira.



Utiliser ChatGPT

Ouvre [chatgpt.com](#) avec le prompt de ce module déjà chargé. Fonctionne avec GPT-5 ou Codex ; toute offre capable de



Utiliser Claude

Ouvre [claude.ai](#) avec le même prompt déjà chargé. Si vous utilisez Claude Code,

naviguer sur le web récupérera la spécification.

vous pouvez plutôt le coller dans un terminal ouvert sur un dossier vide.



Envoyez-le tel quel et vous obtenez un projet, pas un extrait

Le prompt se termine en demandant toute la solution sous forme d'un unique `.zip` , avec les étapes pour la compiler et l'exécuter. Téléchargez-le, ouvrez la solution dans Visual Studio, appuyez sur F5 et AdaptiveOps s'ouvre sur `localhost` . Il vous faut le SDK .NET et le package NuGet `Wisej-4` ; tout le reste, le prompt demande à l'assistant de l'écrire lui-même.

Je suis le Module 2 - « Theme Builder et structure interne du JSON de thème » - du cours LearnWisej « Thèmes et mises en page responsives », et je veux construire avec toi l'exemple de ce module. Réponds-moi en français.

Commence par lire la spécification du module. C'est un unique fichier Markdown (en anglais) qui contient la leçon complète, l'énoncé du lab et les livrables :

<https://learnwisej.com/assets/courses/theming-and-responsive-layouts/specs/module-02.md>

Construis ensuite un projet Wisej.NET 4 complet que je puisse ouvrir dans Visual Studio :

1. Projet « AdaptiveOps », C#, net8.0 ou ultérieur, package NuGet Wisej-4 4.1.x, hôte Kestrel qui appelle `app.UseWisej()`.
2. Construis ce que demande le lab de la spécification : Créez le thème AdaptiveOps à partir d'un thème de base intégré dans le Theme Builder. Définissez une palette de couleurs nommées avec `brandPrimary`, `brandAccent`, `surface`, `surfaceAlt`, `textMain`, `textMuted`, `danger`, `warning`, `success` et `focusFrame` ainsi que les polices `default`, `heading` et `mono`, puis restylez les apparences du bouton, du panneau, de l'onglet, de l'en-tête de grille, de l'éditeur invalide et de l'infobulle à l'aide de ces tokens. Ajoutez une apparence `action-button` qui hérite de `button` et ne redéfinit que ses états `default`, `hovered` et `pressed`, vérifiez chaque modification dans l'aperçu du Theme Builder et chargez le thème dans Adaptive Operations Console.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - AdaptiveOps.theme créé à partir d'un thème de base et chargé par l'application
 - Tokens de couleurs nommés et tokens de polices `default`, `heading` et `mono` définis une seule fois et réutilisés
 - Apparences du bouton, du panneau, de l'onglet, de l'en-tête de grille, de l'éditeur invalide et de l'infobulle restylées à partir des tokens
 - Apparence `action-button` héritant de `button` avec les états `default`, `hovered` et `pressed` redéfinis
 - Captures de l'aperçu du Theme Builder et note sur les choix d'ordre des états et d'héritage
4. Suis la spécification partout où elle nomme une classe, une propriété ou un événement Wisej.NET - utilise exactement celui-là. Là où elle laisse le choix, dis-moi quelle API tu as choisie et pourquoi.
5. Garde l'autorité côté serveur : l'état et la validation vivent dans .NET, le navigateur affiche et remonte les informations. Ne déplace jamais une décision dans JavaScript parce que c'est plus simple là-bas.
6. Rends le résultat démontrable : je dois pouvoir l'exécuter et voir le comportement dont traite ce module, y compris ce qui se passe quand quelque chose tourne mal.

Avant d'écrire la moindre ligne de code, reformule avec tes propres mots ce que tu vas construire et où vivra chaque élément, pour que je puisse corriger la conception tant que cela ne coûte pas cher.

Quand le code est terminé, rassemble toute la solution dans un unique fichier .zip téléchargeable et donne-moi les étapes exactes pour la compiler et l'exécuter.

Enfin, réponds à ces questions de la spécification en t'appuyant sur ta propre implémentation :

- Le marketing change demain `brandPrimary` pour une nouvelle valeur hexadécimale. Quels fichiers et quelles lignes changent dans votre solution, et quelle aurait été la réponse si vous aviez défini `BackColor` sur chaque bouton?

- Vous avez déplacé l'état pressed au-dessus de hovered dans l'apparence action-button. Que voit l'utilisateur tant qu'il maintient le bouton de la souris enfoncé, et pourquoi est-ce l'ordre plutôt que la couleur qui en décide?
- Un TextBox de l'éditeur de détails reste gris lorsqu'il est invalide, alors que vous avez stylé l'état invalid. Quelles sont les trois choses à vérifier, et dans quel ordre, avant de modifier à nouveau le thème?

Pose-moi des questions sur tout ce qui est ambigu avant de commencer.

Puis relisez ce qui vous revient

Un code qui compile n'est pas encore un code juste. Les questions intéressantes sont des questions de conception : est-ce bien le contrôle Wisej.NET que la tâche exige, suit-il le cycle de vie que le framework vous offre, l'état de référence est-il sur le serveur ou a-t-il été laissé dans le navigateur parce que c'était pratique ? Lisez le résultat en relecteur, pas en client.

Ce module vous donne la checklist. Répondez à chaque point en vous appuyant sur le code que vous venez de recevoir :

- ✓ Le marketing change demain brandPrimary pour une nouvelle valeur hexadécimale. Quels fichiers et quelles lignes changent dans votre solution, et quelle aurait été la réponse si vous aviez défini BackColor sur chaque bouton?
- ✓ Vous avez déplacé l'état pressed au-dessus de hovered dans l'apparence action-button. Que voit l'utilisateur tant qu'il maintient le bouton de la souris enfoncé, et pourquoi est-ce l'ordre plutôt que la couleur qui en décide?
- ✓ Un TextBox de l'éditeur de détails reste gris lorsqu'il est invalide, alors que vous avez stylé l'état invalid. Quelles sont les trois choses à vérifier, et dans quel ordre, avant de modifier à nouveau le thème?

Tout ce que vous avez dû corriger mérite d'être noté. Une courte note des conventions et des décisions retenues, collée dans le premier message de votre prochaine session, vaut plus que n'importe quel prompt astucieux.



Nous l'avons déjà construit — comparez votre version à la nôtre

L'implémentation de référence du Module 2 se trouve dans le dépôt public des exemples. Construisez d'abord la vôtre, puis ouvrez la nôtre : ce sont les différences qui sont intéressantes.



[Ouvrir l'exemple du Module 2 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Theming and Responsive Layouts Course/Module 2`.



Notre exemple est un point de départ, pas un produit fini

Il a été construit de la même manière que vous construisez le vôtre — par prompt, puis corrigé — et il s'arrête volontairement là où s'arrête le module. Attendez-vous à des éléments provisoires là où irait du code de production, à une gestion des erreurs minimale et à l'absence de tests. Voyez-y une invitation : utilisez ce que vous avez appris dans ce module, avec l'assistant, pour aller plus loin. Améliorer du code que vous n'avez pas écrit, c'est précisément la compétence que ce cours enseigne.

Trois prompts pour aller plus loin

Orientez votre assistant vers le projet que vous venez de construire — ou vers le nôtre et essayez ceux-ci. Chacun est un exercice de relecture et d'extension, pas une réécriture.

- ✓ **Prouvez que ça fonctionne vraiment.** *"Passe en revue ce projet livrable par livrable par rapport à la spécification du module sur <https://learnwisej.com/assets/courses/theming-and-responsive-layouts/specs/module-02.md> et montre-moi la preuve que chacun est réellement terminé — le chemin dans le code, où cliquer et ce que je dois voir. Là où un livrable n'est que partiellement satisfait, dis-le clairement au lieu de le défendre. Ensuite, ajoute ce qui manque."*
- ✓ **Relisez-le comme une pull request.** *"Fais-moi parcourir ce code AdaptiveOps comme si tu relisais la pull request d'un collègue et que tu ne l'avais pas écrit. Où vit l'état, que se passe-t-il de bout en bout lors d'une action de l'utilisateur, que renverrais-tu pour correction, et quelles parties ne serais-tu pas à l'aise de montrer à un client ? Sois précis et cite les lignes dont tu parles — ne me résume pas la conception."*
- ✓ **Remplacez une pièce sans déplacer les jointures.** *"Choisis la partie de ce projet AdaptiveOps qui remplace le plus manifestement quelque chose de réel — une source de données factice, un stockage en mémoire, un service simplifié — et remplace-la par une implémentation digne de la production. Le contrat de part et d'autre ne doit pas changer : mêmes types, mêmes appels, même comportement en cas d'échec. Dis-moi ce que l'implémentation réelle t'a obligé à gérer et que la version provisoire ne gérait pas."*



Emportez-le dans le lab

Le lab pratique qui clôt ce module demande les mêmes livrables que ceux que vous venez de faire produire à l'assistant. Servez-vous de ce que vous avez ici comme forme de votre réponse et comme preuve à l'appui — pas comme la réponse elle-même. Le lab est noté sur votre raisonnement, et cette partie doit venir de vous.