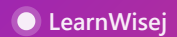


KI-Coding-Übung

Erstellen Sie das Beispielprojekt AdaptiveOps dieses Moduls mit ChatGPT oder Claude aus einem fertigen Prompt, der das Modell auf die Spezifikation des Moduls verweist – und prüfen, starten und erweitern Sie dann, was Sie erhalten, auch im Vergleich mit unserem eigenen Referenz-Build auf GitHub.



Der Assistent liefert die schnellen Hände. Der Architekt bleiben Sie.

Ein Assistent erstellt das Beispielprojekt dieses Moduls in wenigen Minuten. Was er nicht kann, ist zu entscheiden, ob das Ergebnis der richtige Entwurf für Ihre Anwendung ist – diese Entscheidung bleibt bei Ihnen. Das Urteilsvermögen bringen Sie mit: Instruieren Sie ihn sorgfältig und prüfen Sie dann, was zurückkommt, anhand dessen, was Ihnen Modul 2 gerade vermittelt hat.

Worum Sie bitten

Erstellen Sie im Theme Builder das AdaptiveOps-Theme aus einem integrierten Basis-Theme. Definieren Sie eine benannte Farbpalette mit brandPrimary, brandAccent, surface, surfaceAlt, textMain, textMuted, danger, warning, success und focusFrame sowie die Schriften default, heading und mono, und gestalten Sie anschließend die Appearances für Button, Panel, Tab, Grid-Header, ungültigen Editor und Tooltip mit diesen Tokens neu. Fügen Sie eine action-button-Appearance hinzu, die von button erbt und nur ihre Zustände default, hovered und pressed überschreibt, prüfen Sie jede Änderung in der Vorschau des Theme Builders und laden Sie das Theme in der Adaptive Operations Console.

Das Modul verlangt diese Ergebnisse, und der Prompt unten fordert genau diese vom Assistenten an:

- ✓ AdaptiveOps.theme aus einem Basis-Theme erstellt und von der Anwendung geladen
- ✓ Benannte Farb-Tokens sowie die Schrift-Tokens default, heading und mono einmal definiert und wiederverwendet
- ✓ Appearances für Button, Panel, Tab, Grid-Header, ungültigen Editor und Tooltip anhand der Tokens neu gestaltet
- ✓ action-button-Appearance, die von button erbt und die Zustände default, hovered und pressed überschreibt
- ✓ Screenshots der Theme-BUILDER-Vorschau und eine Notiz zu den Entscheidungen bei Zustandsreihenfolge und Vererbung

Warum es so aufgebaut ist

Modul 1 hat der Adaptive Operations Console eine gedockte Shell gegeben, doch Farben und Schriften bestimmt noch vollständig das Basis-Theme. Die Abkürzung wäre, sie Control für Control mit BackColor, Font, CssStyle -Strings oder CSS gegen das interne DOM der Widgets zu branden. Jeder dieser Wege

verstreut denselben Hex-Wert über Dutzende Stellen, kämpft gegen die Theme-Engine, die bereits jeden Zustand rendert, und bricht beim nächsten Upgrade. Ein Theme ist eine einzige JSON-Datei, die die Identität der Standard-Controls bestimmt; mit dem Theme Builder erstellen und pflegen Sie es, und in diesem Modul bauen Sie damit das AdaptiveOps-Theme.

Diese Begriffe muss der Assistent richtig verwenden. Geht die Antwort nachlässig mit ihnen um, hat er das Modul nicht verstanden:

Begriff	Was er hier bedeutet
Basis-Theme	Das integrierte Theme, das der Theme Builder kopiert, um Ihres zu erstellen; so existieren Hunderte Appearances, Komponenten, Zustände, Metriken, Bilder und Schriften, bevor Sie einen einzigen Wert ändern.
benannte Farb- und Schrift-Tokens	Einträge in den Abschnitten colors und fonts (brandPrimary, danger, heading), auf die Appearances über den Namen verweisen, sodass eine Markenänderung eine einzige Bearbeitung ist.
Appearance und AppearanceKey	Eine Appearance ist der Theme-Block für eine Art von Widget; der AppearanceKey eines Controls wählt aus, welcher Block gilt – so entsteht eine semantische Variante wie ein Aktions-Button.
Vererbung von Appearances	Ein inherit -Eintrag erlaubt einer spezialisierten Appearance, eine Basis-Appearance wiederzuverwenden und nur wenige Zustände zu überschreiben, sodass action-button überall sonst ein echter Button bleibt.
Reihenfolge der Zustände	Zustände wie default, hovered, pressed, focused, disabled und invalid werden in Listenreihenfolge angewendet; verschiebt man einen, ändert sich, welche Styles gewinnen. Die Reihenfolge ist also Verhalten.
Child-Komponente	Ein innerer Teil eines Widgets, etwa die Titelleiste eines Panels, ein Spaltenkopf im Grid oder ein Editor-Button, mit eigenem Appearance-Pfad und eigenen Zuständen.
Styles und Eigenschaften	Styles sind CSS-ähnliche Werte, die das Theme in dynamische CSS-Klassen umwandelt und selbst verwaltet; Eigenschaften sind Widget-Eigenschaften, die zu Inline-Styles oder Verhalten werden können und von der Anwendung überschrieben werden dürfen.

Selbst bauen – wählen Sie Ihren Assistenten

Beide Buttons öffnen einen neuen Tab, in dem **derselbe fertige Prompt** bereits eingetragen ist. Der Prompt verweist das Modell auf die Spezifikation dieses Moduls – eine einzelne Markdown-Datei mit der Lektion, der Lab-Aufgabe und den geforderten Ergebnissen –, sodass es mit dem echten Material arbeitet, statt zu raten, worum es in Modul 2 geht. Sie können [module-02.md](#) selbst öffnen, um genau zu sehen, was es lesen wird.



ChatGPT verwenden



Claude verwenden

Öffnet chatgpt.com mit dem Prompt dieses Moduls. Funktioniert mit GPT-5 oder Codex; jeder Tarif mit Websuche ruft die Spezifikation ab.

Öffnet claude.ai mit demselben Prompt. Wenn Sie Claude Code nutzen, können Sie ihn stattdessen in einem leeren Ordner in ein Terminal einfügen.



Unverändert abgeschickt, erhalten Sie ein Projekt, kein Code-Fragment

Der Prompt verlangt am Ende die gesamte Projektmappe als einzelne [.zip](#) -Datei mit Anleitung zum Kompilieren und Starten. Laden Sie sie herunter, öffnen Sie die Projektmappe in Visual Studio, drücken Sie F5, und AdaptiveOps startet auf [localhost](#) . Sie benötigen das .NET SDK und das NuGet-Paket [Wisej-4](#) ; alles andere soll der Assistent laut Prompt selbst schreiben.

Ich arbeite gerade Modul 2 – „Theme Builder und der interne Aufbau von Theme-JSON“ – des LearnWisej-Kurses „Theming und responsive Layouts“ durch und möchte das Beispielprojekt des Moduls gemeinsam mit dir erstellen. Bitte antworte auf Deutsch.

Lies zuerst die Modulspezifikation. Sie ist eine einzelne Markdown-Datei (auf Englisch) mit der vollständigen Lektion, der Lab-Aufgabe und den geforderten Ergebnissen:

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

Erstelle dann ein vollständiges Wisej.NET-4-Projekt, das ich in Visual Studio öffnen kann:

1. Projekt „AdaptiveOps“, C#, net8.0 oder neuer, NuGet-Paket Wisej-4 4.1.x, Kestrel-Host, der `app.UseWisej()` aufruft.
2. Setze um, was das Lab der Spezifikation verlangt: Erstellen Sie im Theme Builder das AdaptiveOps-Theme aus einem integrierten Basis-Theme. Definieren Sie eine benannte Farbpalette mit `brandPrimary`, `brandAccent`, `surface`, `surfaceAlt`, `textMain`, `textMuted`, `danger`, `warning`, `success` und `focusFrame` sowie die Schriften `default`, `heading` und `mono`, und gestalten Sie anschließend die Appearances für Button, Panel, Tab, Grid-Header, ungültigen Editor und Tooltip mit diesen Tokens neu. Fügen Sie eine action-button-Appearance hinzu, die von button erbt und nur ihre Zustände `default`, `hovered` und `pressed` überschreibt, prüfen Sie jede Änderung in der Vorschau des Theme Builders und laden Sie das Theme in der Adaptive Operations Console.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - AdaptiveOps.theme aus einem Basis-Theme erstellt und von der Anwendung geladen
 - Benannte Farb-Tokens sowie die Schrift-Tokens `default`, `heading` und `mono` einmal definiert und wiederverwendet
 - Appearances für Button, Panel, Tab, Grid-Header, ungültigen Editor und Tooltip anhand der Tokens neu gestaltet
 - action-button-Appearance, die von button erbt und die Zustände `default`, `hovered` und `pressed` überschreibt
 - Screenshots der Theme-Builder-Vorschau und eine Notiz zu den Entscheidungen bei Zustandsreihenfolge und Vererbung
4. Halte dich an die Spezifikation, wo immer sie eine Wisej.NET-Klasse, -Eigenschaft oder ein -Event nennt – verwende genau diese. Wo sie eine Wahl offenlässt, sag mir, welche API du gewählt hast und warum.
5. Der Server bleibt maßgeblich: Zustand und Validierung leben in .NET, der Browser stellt dar und meldet zurück. Verlagere nie eine Entscheidung nach JavaScript, nur weil es dort einfacher ist.
6. Mach das Ergebnis vorführbar: Ich muss es starten und das Verhalten sehen können, um das es in diesem Modul geht – auch, was passiert, wenn etwas schiefgeht.

Bevor du Code schreibst, fasse in eigenen Worten zusammen, was du bauen wirst und wo jedes Teil liegen wird, damit ich den Entwurf korrigieren kann, solange das noch wenig kostet.

Wenn der Code fertig ist, packe die gesamte Projektmappe in eine einzelne .zip-Datei zum Herunterladen und nenne mir die genauen Schritte, um sie zu kompilieren und auszuführen.

Beantworte zum Schluss diese Fragen aus der Spezifikation und nutze dabei deine eigene Implementierung als Beleg:

- Das Marketing ändert `brandPrimary` morgen auf einen neuen Hex-Wert. Welche Dateien und welche Zeilen ändern sich in Ihrer Solution, und wie hätte die Antwort gelautet, wenn Sie stattdessen `BackColor` an jedem Button gesetzt hätten?
- Sie haben den Zustand `pressed` in der action-button-Appearance über `hovered` verschoben. Was

sieht ein Benutzer, während er die Maustaste gedrückt hält, und warum entscheidet darüber die Reihenfolge und nicht die Farbe?

– Eine TextBox im Details-Editor bleibt grau, wenn sie ungültig ist, obwohl Sie den Zustand invalid gestaltet haben. Welche drei Dinge prüfen Sie in welcher Reihenfolge, bevor Sie das Theme erneut bearbeiten?

Frag mich nach allem, was unklar ist, bevor du anfängst.

Prüfen Sie dann, was zurückkommt

Code, der kompiliert, ist noch lange kein richtiger Code. Die spannenden Fragen sind Designfragen: Ist das wirklich das Wisej.NET-Control, das die Aufgabe verlangt? Folgt der Code dem Lebenszyklus, den das Framework vorgibt? Liegt der maßgebliche Zustand auf dem Server – oder wurde er im Browser abgelegt, weil das bequemer war? Lesen Sie das Ergebnis als Reviewer, nicht als Kunde.

Die Checkliste liefert Ihnen dieses Modul. Beantworten Sie jeden Punkt anhand des Codes, den Sie gerade erhalten haben:

- ✓ Das Marketing ändert brandPrimary morgen auf einen neuen Hex-Wert. Welche Dateien und welche Zeilen ändern sich in Ihrer Solution, und wie hätte die Antwort gelautet, wenn Sie stattdessen BackColor an jedem Button gesetzt hätten?
- ✓ Sie haben den Zustand pressed in der action-button-Appearance über hovered verschoben. Was sieht ein Benutzer, während er die Maustaste gedrückt hält, und warum entscheidet darüber die Reihenfolge und nicht die Farbe?
- ✓ Eine TextBox im Details-Editor bleibt grau, wenn sie ungültig ist, obwohl Sie den Zustand invalid gestaltet haben. Welche drei Dinge prüfen Sie in welcher Reihenfolge, bevor Sie das Theme erneut bearbeiten?

Alles, was Sie korrigieren mussten, sollten Sie festhalten. Eine kurze Notiz zu den Konventionen und Entscheidungen, auf die Sie sich festgelegt haben – eingefügt in die erste Nachricht Ihrer nächsten Sitzung –, ist mehr wert als jeder noch so clevere Prompt.



Dieses Beispiel haben wir schon gebaut – vergleichen Sie Ihres mit unserem

Die Referenzimplementierung zu Modul 2 liegt im öffentlichen Beispiel-Repository. Bauen Sie zuerst Ihre eigene Version und öffnen Sie dann unsere: Die Unterschiede sind der interessante Teil.



[Beispiel zu Modul 2 auf GitHub öffnen](#)

Oder klonen Sie alle Beispiele mit `git clone https://github.com/iceteagroup/LearnWisej-Samples` und öffnen Sie [Theming and Responsive Layouts Course/Module 2](#).



Unser Beispiel ist ein Ausgangspunkt, kein fertiges Produkt

Es ist genauso entstanden wie Ihres – per Prompt und anschließend korrigiert – und endet bewusst dort, wo das Modul endet. Rechnen Sie mit Platzhaltern, wo Produktivcode hingehört, mit knapper Fehlerbehandlung und ohne Tests. Verstehen Sie das als Einladung: Nutzen Sie, was Sie in diesem Modul gelernt haben, und dazu den Assistenten, um es weiterzuentwickeln. Code zu verbessern, den Sie nicht selbst geschrieben haben, ist die Fähigkeit, die dieser Kurs eigentlich vermittelt.

Drei Prompts für den nächsten Schritt

Richten Sie Ihren Assistenten auf das Projekt, das Sie gerade gebaut haben – oder auf unseres –, und probieren Sie diese Prompts aus. Jeder ist eine Übung zum Prüfen und Erweitern, kein Neuschreiben.

- ✓ **Beweisen Sie, dass es wirklich funktioniert.** *"Geh dieses Projekt Ergebnis für Ergebnis anhand der Modulspezifikation unter <https://learnwisej.com/assets/courses/theming-and-responsive-layouts/specs/module-02.md> durch und zeig mir den Beleg, dass jedes wirklich erledigt ist – den Codepfad, worauf ich klicken muss und was ich sehen sollte. Wo ein Ergebnis nur teilweise erfüllt ist, sag das klar, statt es schönzureden. Ergänze dann, was fehlt."*
- ✓ **Prüfen Sie es wie einen Pull-Request.** *"Führ mich durch diesen AdaptiveOps-Code, als würdest du den Pull-Request eines Kollegen prüfen und hättest ihn nicht selbst geschrieben. Wo lebt der Zustand, was passiert bei einer einzelnen Benutzeraktion von Anfang bis Ende, was würdest du zur Überarbeitung zurückgeben, und welche Teile würdest du nicht guten Gewissens einem Kunden vorlegen? Sei konkret und zitiere die Zeilen, die du meinst – fasse mir nicht den Entwurf zusammen."*
- ✓ **Tauschen Sie ein Teil aus, ohne die Schnittstellen zu verschieben.** *"Wähle den Teil dieses AdaptiveOps-Projekts, der am offensichtlichsten nur ein Platzhalter für etwas Echtes ist – eine Fake-Datenquelle, ein In-Memory-Speicher, ein vereinfachter Service – und ersetze ihn durch eine praxisnahe Implementierung. Der Vertrag auf beiden Seiten darf sich nicht ändern: dieselben Typen, dieselben Aufrufe, dasselbe Verhalten im Fehlerfall. Sag mir, was du durch die echte Implementierung behandeln musstest, was der Platzhalter nicht behandelt hat."*



Nehmen Sie es mit ins Lab

Das praktische Lab am Ende dieses Moduls verlangt dieselben Ergebnisse, die Sie gerade vom Assistenten haben erstellen lassen. Nutzen Sie, was Sie hier haben, als Gerüst für Ihre Antwort und als Beleg dafür – nicht als die Antwort selbst. Bewertet wird im Lab Ihre Argumentation, und dieser Teil muss von Ihnen stammen.