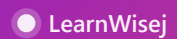


KI-Coding-Übung

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.



Instruieren, lesen, dann auf die Probe stellen.

Diese Übung hat drei Teile, und nur der erste dreht sich um den Prompt. Sie instruieren einen Assistenten mit der Modulspezifikation und erhalten eine AdaptiveOps-Projektmappe zurück; Sie lesen, was er geschrieben hat, und entscheiden, ob es wirklich tut, was das Modul verlangt; dann treiben Sie es weiter, als das Modul verlangt hat. Im zweiten und dritten Teil steckt das eigentliche Lernen.

Worum Sie bitten

Fügen Sie der Adaptive Operations Console ein bereichsbezogenes Stylesheet AdaptiveOps.css mit Klassen für Kennzahlenkarten und kompakte Badges hinzu, wenden Sie diese über CssClass an und verwenden Sie CssStyle nur für einen einzigen berechneten Wert, etwa eine Fortschrittsbreite. Weisen Sie dem primären Befehl den AppearanceKey action-button zu, fügen Sie den Zeilen des Ticket-Grids oder einem Status-Panel einen eigenen Zustand stale hinzu und implementieren Sie in der Toolbar einen Hell/Dunkel-Umschalter, der das Theme nur für die aktuelle Session wechselt, ohne das gemeinsame Theme zu verändern.

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

- ✓ Bereichsbezogene AdaptiveOps.css mit den Klassen metric-card und compact-badge, angewendet über CssClass
- ✓ CssStyle für genau einen berechneten Einzelwert verwendet, mit dokumentierter Begründung
- ✓ Primärer Befehl mit dem AppearanceKey action-button
- ✓ Eigener Zustand stale über States hinzugefügt und im Theme gestaltet
- ✓ Hell/Dunkel-Umschalter nur für die Session, der das gemeinsame Theme unangetastet lässt

Warum es so aufgebaut ist

Das AdaptiveOps-Theme aus Modul 2 bestimmt jetzt die Identität jedes Standard-Controls in der Adaptive Operations Console, und die nächsten Anforderungen sind alle klein: Eine Kennzahlenkarte braucht eine abgerundete, erhöhte Fläche, ein SLA-Badge soll kompakt sein, ein Fortschrittsbalken braucht eine aus Live-Daten berechnete Breite, der Befehl Assign soll wie die primäre Aktion aussehen, und die Nachtschicht wünscht sich eine dunkle Variante. Für jede Anforderung gibt es eine verlockende Abkürzung – hier ein CssStyle -String, dort ein BackColor, das gemeinsame Theme direkt bearbeitet –, und jede Abkürzung

verschiebt eine visuelle Entscheidung stillschweigend auf eine Ebene, die nicht dafür zuständig ist. Sechs Monate später kann niemand mehr sagen, was das Theme rendert, was das CSS überschreibt und was der Code flickt. Dieses Modul ordnet jede dieser fünf Anforderungen dem Mechanismus zu, der dafür zuständig ist.

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
CssClass	Ein oder mehrere Klassennamen, die im Browser dem Hauptelement des Widgets hinzugefügt werden: der wiederverwendbare Anknüpfungspunkt, den ein bereichsbezogenes Anwendungs-Stylesheet ansprechen kann, ohne das Theme anzufassen.
CssStyle	Ein CSS-String, der direkt als style-Attribut des Widget-Hauptelements gerendert wird; richtig für einen einzelnen berechneten Wert, falsch als allgemeines Styling-Werkzeug, weil Inline-Styles schwer zu untersuchen, zu themen und an einen Designer zu übergeben sind.
bereichsbezogenes Anwendungs-Stylesheet	Eine CSS-Datei, die neben dem Theme geladen wird und deren Selektoren bei eigenen Klassennamen wie <code>metric-card</code> beginnen und nie in das interne DOM eines Standard-Widgets von Wisej greifen.
Legacy-StyleSheet-Extender	Eine ältere Komponente, die geeigneten Controls CSS-Klassen hinzufügt und Regeln in ihrer Eigenschaft <code>StyleSheetSource</code> hält; eine Migrationshilfe, in neuem Code überflüssig, weil heute jedes Control <code>CssClass</code> und <code>CssStyle</code> besitzt.
AppearanceKey	Der Name einer Appearance im aktiven Theme, mit der ein Control gerendert wird; so drückt C# die Absicht aus (<code>action-button</code>), und das Theme bestimmt Farben, Schriften und Metriken jedes Zustands.
eigene Zustände	Namen in der <code>States</code> -Collection eines Controls, die das Theme wie <code>hovered</code> oder <code>focused</code> gestalten kann – für vorübergehende UI-Bedingungen wie <code>stale</code> , <code>warning</code> oder <code>dirty</code> , die ein Designer sehen und neu gestalten können soll.
Start-Theme und <code>Application.Theme</code>	Das anwendungsweite Theme wird vor dem Start der Anwendung in <code>Web.config</code> oder <code>Default.json</code> gewählt; mit <code>Application.Theme</code> kann eine Session zur Laufzeit mit einem Theme-Objekt arbeiten, ohne zu ändern, was andere Sessions sehen.

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 3 geht. Sie können [module-03.md](#) selbst öffnen, um genau zu sehen, was es lesen wird.



ChatGPT verwenden

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



Claude verwenden

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 3 – „CSS, CssClass, CssStyle, Zustände und Theme-Wechsel zur Laufzeit“ – 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-03.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: Fügen Sie der Adaptive Operations Console ein bereichsbezogenes Stylesheet `AdaptiveOps.css` mit Klassen für Kennzahlenkarten und kompakte Badges hinzu, wenden Sie diese über `CssClass` an und verwenden Sie `CssStyle` nur für einen einzigen berechneten Wert, etwa eine Fortschrittsbreite. Weisen Sie dem primären Befehl den `AppearanceKey` `action-button` zu, fügen Sie den Zeilen des Ticket-Grids oder einem Status-Panel einen eigenen Zustand `stale` hinzu und implementieren Sie in der Toolbar einen Hell/Dunkel-Umschalter, der das Theme nur für die aktuelle Session wechselt, ohne das gemeinsame Theme zu verändern.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Bereichsbezogene `AdaptiveOps.css` mit den Klassen `metric-card` und `compact-badge`, angewendet über `CssClass`
 - `CssStyle` für genau einen berechneten Einzelwert verwendet, mit dokumentierter Begründung
 - Primärer Befehl mit dem `AppearanceKey` `action-button`
 - Eigener Zustand `stale` über `States` hinzugefügt und im Theme gestaltet
 - Hell/Dunkel-Umschalter nur für die Session, der das gemeinsame Theme unangetastet lässt
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:

- Ein Designer wünscht einen kleineren Eckenradius für jede Kennzahlenkarte. Welche Datei ändert sich, und an wie vielen Stellen hätten Sie ändern müssen, wenn der Radius mit `CssStyle` gesetzt worden wäre?
- Warum braucht der Button `Assign` keinen Code für `hover` oder `pressed`, und was genau würden Sie verlieren, wenn Sie seinen `AppearanceKey` durch zur Palette passende Werte für `BackColor` und `ForeColor` ersetzen?
- Zwei Personen sind angemeldet, und eine davon wechselt in den Dunkelmodus. Beschreiben

Sie, was jede sieht, und zeigen Sie auf die Zeile Ihres Codes, die das garantiert.

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:

- ✓ Ein Designer wünscht einen kleineren Eckenradius für jede Kennzahlenkarte. Welche Datei ändert sich, und an wie vielen Stellen hätten Sie ändern müssen, wenn der Radius mit `CssStyle` gesetzt worden wäre?
- ✓ Warum braucht der Button Assign keinen Code für `hover` oder `pressed`, und was genau würden Sie verlieren, wenn Sie seinen `AppearanceKey` durch zur Palette passende Werte für `BackColor` und `ForeColor` ersetzen?
- ✓ Zwei Personen sind angemeldet, und eine davon wechselt in den Dunkelmodus. Beschreiben Sie, was jede sieht, und zeigen Sie auf die Zeile Ihres Codes, die das garantiert.

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 3 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 3 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 3](#).



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.

- ✓ **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."*
- ✓ **Machen Sie es fit für einen schlechten Tag.** *"Untersuche dieses AdaptiveOps-Projekt auf Fehlerszenarien, die es noch nicht behandelt – fehlerhafte oder fehlende Eingaben, eine Abhängigkeit, die langsam oder nicht erreichbar ist, Arbeit, die eintrifft, nachdem der Benutzer die Seite verlassen hat, und zwei Aktionen, die sich gegenseitig in die Quere kommen. Sag mir für jedes zuerst, ob es in Wisej.NET 4 ein echtes Risiko ist und warum, und schreib erst dann die Korrektur. Wo du entscheidest, dass nichts behandelt werden muss, sag das und begründe es, statt defensiven Code hinzuzufügen."*



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.