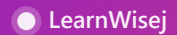


KI-Coding-Übung

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.



Dieses Beispiel tippen Sie nicht ab. Sie steuern es.

Kaum jemand schreibt Wisej.NET-Code heute noch in einer leeren Datei: Ein Assistent liefert den ersten Entwurf, und der Entwickler entscheidet, ob er stimmt. Das funktioniert aber nur, wenn Sie den Stoff beherrschen – und genau den hat Ihnen der Rest dieses Moduls vermittelt. Hier geben Sie dieses Wissen an ChatGPT oder Claude weiter, erhalten das Beispielprojekt des Moduls als lauffähige Visual-Studio-Projektmappe zurück und prüfen es so, wie Sie den Pull-Request eines Kollegen prüfen würden.

Worum Sie bitten

Legen Sie die Solution Adaptive Operations Console und ihre erste Desktop-Shell an: eine Haupt-Page mit einem Toolbar-Panel oben, einem Navigations-Panel links, einem füllenden Arbeitsbereich, einem Details-Panel rechts und einem Status-Panel unten, aufgebaut mit Dock und verschachtelten Panels. Wählen Sie ein integriertes Basis-Theme und dokumentieren Sie die Wahl, schreiben Sie eine VisualInventory.md, die die Controls mit eigenen Appearances, die benannten Farb- und Schrift-Tokens, die Layout-Bereiche sowie die Breakpoints für Desktop, Tablet und Smartphone auflistet, und fügen Sie ein Status-Label hinzu, das die aktuelle Browserbreite meldet, damit spätere Module die Profile überprüfen können.

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

- ✓ Solution Adaptive Operations Console mit gedockter Desktop-Shell aus Toolbar-, Navigations-, Arbeitsbereichs-, Details- und Status-Panel
- ✓ Basis-Theme ausgewählt und konfiguriert, Begründung in den Lab-Notizen festgehalten
- ✓ VisualInventory.md mit den Controls, die Appearances benötigen, den Theme-Tokens, den Layout-Bereichen und den Breakpoints
- ✓ Status-Label, das die aktuelle Browserbreite meldet
- ✓ Desktop-Screenshot und eine Notiz, welche Ebene für welche visuelle Entscheidung zuständig ist

Warum es so aufgebaut ist

Ein Team, das Wisej.NET wie ein CSS-first-Framework behandelt, landet bei BackColor auf zweihundert Controls, bei Selektoren, die beim nächsten Upgrade brechen, und bei Resize-Handlern, die Bounds von Hand neu berechnen; wird dann eine Smartphone-Version verlangt, bleibt nur noch eine Neuentwicklung. Dieses Modul klärt, woher Aussehen und Layout eines Controls wirklich kommen, benennt die drei

Anpassungsebenen und ihre Zuständigkeiten und beginnt die Shell der Adaptive Operations Console, die jedes spätere Modul erweitert.

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
Server-Control und Client-Widget	Sie schreiben .NET-Controls auf dem Server; der Browser zeigt JavaScript-Widgets an, die WISEJ.NET synchron hält. Eine visuelle Entscheidung kann also auf beiden Seiten dieser Grenze liegen.
Theme	Eine einzige JSON-Datei mit Farben, Bildern, Schriften, Appearances, Zuständen, Styles, Eigenschaften und Einstellungen, die jedem Standard-Control seine Identität gibt; der Vertrag zwischen beiden Welten.
Appearance und AppearanceKey	Jedes Control stellt einen Appearance-Schlüssel bereit; das aktive Theme enthält eine Appearance für diesen Schlüssel, und ein semantischer Schlüssel wie action-button wählt eine Variante aus.
Zustand	Ein benannter Interaktionszustand innerhalb einer Appearance (default, hovered, pressed, focused, invalid, selected), dessen Styles und Eigenschaften gelten, solange er aktiv ist.
CssClass und CssStyle	Direktes Browser-Styling: eine Klasse aus einem bereichsbezogenen Anwendungs-Stylesheet oder ein roher Style-String am Widget-Element für einen seltenen Einzelwert.
Layout-Engine und Dock	Position und Größe ergeben sich aus der Container-Hierarchie, nicht aus dem Theme; Docking stapelt Bereiche an den Rändern des Parents und füllt den verbleibenden Platz.
visuelle Bestandsaufnahme	Die schriftliche Liste der Controls, die Appearances benötigen, der Farb- und Schrift-Tokens, der Layout-Bereiche und der Breakpoints, auf deren Grundlage Theme, Layout und Profile geplant werden.

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 1 geht. Sie können [module-01.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



Claude verwenden

Öffnet [claude.ai](#) mit demselben Prompt. Wenn Sie Claude Code nutzen, können Sie

Codex; jeder Tarif mit Websuche ruft die Spezifikation ab.

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 1 – „Das visuelle System von Wisej.NET“ – 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-01.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: Legen Sie die Solution Adaptive Operations Console und ihre erste Desktop-Shell an: eine Haupt-Page mit einem Toolbar-Panel oben, einem Navigations-Panel links, einem füllenden Arbeitsbereich, einem Details-Panel rechts und einem Status-Panel unten, aufgebaut mit Dock und verschachtelten Panels. Wählen Sie ein integriertes Basis-Theme und dokumentieren Sie die Wahl, schreiben Sie eine `VisualInventory.md`, die die Controls mit eigenen Appearances, die benannten Farb- und Schrift-Tokens, die Layout-Bereiche sowie die Breakpoints für Desktop, Tablet und Smartphone auflistet, und fügen Sie ein Status-Label hinzu, das die aktuelle Browserbreite meldet, damit spätere Module die Profile überprüfen können.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Solution Adaptive Operations Console mit gedockter Desktop-Shell aus Toolbar-, Navigations-, Arbeitsbereichs-, Details- und Status-Panel
 - Basis-Theme ausgewählt und konfiguriert, Begründung in den Lab-Notizen festgehalten
 - `VisualInventory.md` mit den Controls, die Appearances benötigen, den Theme-Tokens, den Layout-Bereichen und den Breakpoints
 - Status-Label, das die aktuelle Browserbreite meldet
 - Desktop-Screenshot und eine Notiz, welche Ebene für welche visuelle Entscheidung zuständig ist
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:

- Der primäre Befehl der Toolbar braucht auf jedem Bildschirm die Markenfarbe. Welche der drei Ebenen ist für diese Entscheidung zuständig, und was ginge schief, wenn Sie stattdessen `BackColor` am Button setzen?
- Ihr Breiten-Label zeigt 1024 px, und das Details-Panel ist noch sichtbar. Welcher Mechanismus aus einem späteren Modul entscheidet, ob es das sein sollte, und warum ist eine

CSS-Media-Query in Wisej.NET die falsche erste Antwort?

– Wenn das Basis-Theme morgen gegen ein anderes integriertes Theme getauscht würde: Welche Zeilen Ihrer VisualInventory.md würden sich ändern und welche nicht, und was sagt Ihnen das darüber, wo die Entscheidungen der Bestandsaufnahme angesiedelt sind?

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:

- ✓ Der primäre Befehl der Toolbar braucht auf jedem Bildschirm die Markenfarbe. Welche der drei Ebenen ist für diese Entscheidung zuständig, und was ginge schief, wenn Sie stattdessen BackColor am Button setzen?
- ✓ Ihr Breiten-Label zeigt 1024 px, und das Details-Panel ist noch sichtbar. Welcher Mechanismus aus einem späteren Modul entscheidet, ob es das sein sollte, und warum ist eine CSS-Media-Query in Wisej.NET die falsche erste Antwort?
- ✓ Wenn das Basis-Theme morgen gegen ein anderes integriertes Theme getauscht würde: Welche Zeilen Ihrer VisualInventory.md würden sich ändern und welche nicht, und was sagt Ihnen das darüber, wo die Entscheidungen der Bestandsaufnahme angesiedelt sind?

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 1 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 1 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 1](#).



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.

- ✓ **Ergänzen Sie den zweiten Fall.** *"Erweitere dieses AdaptiveOps-Projekt um einen zweiten Fall, der demselben Muster folgt wie der erste – eine weitere Entität, Maske oder ein Control derselben Form, auf dieselbe Weise verdrahtet. Lagere wirklich Gemeinsames in eine Basisklasse oder einen Service aus, aber nur dort, wo es sich lohnt; zwei ähnliche Dinge brauchen nicht immer eine Abstraktion. Zeig mir den Diff und begründe jede Auslagerung."*
- ✓ **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-01.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."*



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.