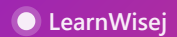


AI Coding Exercise

Build this module's VisualOperationsStudio sample with ChatGPT or Claude from a ready-made prompt that points the model at this module's specification, then review, run and extend what it gives you.



Writing the code is the easy half now.

Getting working code for "Wisej.Web.Canvas and the HTML5 Canvas 2D Model" out of a modern assistant takes one good prompt. Judging it, correcting it and extending it takes everything this module covered. That is the split this exercise is built around — you supply the specification and the review, the assistant supplies the typing.

What you are asking for

Build a Canvas playground page in VisualOperationsStudio that ports a set of browser Canvas 2D examples to the documented Wisej.Web.Canvas API. Drive everything from a DrawPlayground method called by the Canvas Redraw event so the scene returns after a resize, and start it with ClearRect over the full surface. Render a stroked path, filled rectangles, a placed caption, a linear gradient and a radial gradient, then wrap each transformed section in Save() and Restore() and demonstrate Translate, Scale, Rotate and SetTransform. Add a clipped region, an alpha section, a dashed line and a shadow, port one browser example that rotates by $\text{Math.PI} / 4$ into the degree value Wisej expects, and keep LiveUpdate off for the main render while showing it in one small progressive example.

The module asks for these deliverables, and the prompt below asks the assistant for exactly them:

- ✓ Canvas playground whose entire scene is drawn by a DrawPlayground method called from Redraw
- ✓ Paths, rectangles, a placed caption, a linear gradient and a radial gradient using documented Wisej.Web.Canvas members only
- ✓ Save() and Restore() around every Translate, Scale, Rotate and SetTransform section
- ✓ Clipping, alpha, a dashed line and a shadow, plus one radians-to-degrees port written out in a comment
- ✓ A LiveUpdate demonstration kept separate from the main render, with a note on when it is appropriate

Why it is shaped this way

Everything VisualOperationsStudio has drawn so far was produced on the server: the Module 2 gauge and the Module 3 Health and Trend cells are rendered with System.Drawing and reach the user as images. The topology view the application needs next is dozens of small nodes, edges and labels that a user will eventually pan, zoom and select, which is exactly the work a browser canvas does well.

These are the terms the assistant has to get right. If what comes back uses them loosely, it has not understood the module:

Term	What it means here
Wisej.Web.Canvas	A Wisej control exposing the browser's immediate-mode 2D surface to C#; its calls are drawing commands the browser executes, not System.Drawing.Graphics operations.
Redraw event	The hook the surface raises when it needs its content again; because Wisej.NET does not retain the commands, the handler has to reconstruct the whole scene.
PascalCase correspondence	Members map by case — fillRect() to FillRect(), strokeStyle to StrokeStyle — while CSS values become .NET types such as Color, Rectangle, Font and Image.
Degree-based angles	Wisej.NET documents angle arguments in degrees for simplicity where the browser uses radians, which changes every ported Arc() and Rotate() call.
Save() / Restore()	Push and pop the whole rendering state — styles, transform, clip, alpha, line and text settings — so a local change cannot leak into the next shape.
TextFont, TextAlign, TextBaseline	The Wisej.NET stand-in for the browser's font, textAlign and textBaseline; the font is a real System.Drawing.Font.
LiveUpdate	False by default, so commands travel with the request's normal UI update; true pushes each call at once, which suits a progressive reveal and ruins an animation loop.

Build it — pick your assistant

Both buttons open a new tab with the **same ready-made prompt** already filled in. The prompt points the model at this module's specification — a single Markdown file holding the lesson, the lab brief and the deliverables — so it works from the real material instead of guessing what Module 4 is about. You can open [module-04.md](#) yourself to see exactly what it will read.



Use ChatGPT

Opens [chatgpt.com](#) with this module's prompt loaded. Works with GPT-5 or Codex; any plan that can browse the web will fetch the spec.



Use Claude

Opens [claude.ai](#) with the same prompt loaded. Claude Code users can paste it into a terminal in an empty folder instead.



Send it as-is and you get a project, not a snippet

The prompt ends by asking for the whole solution as a single [.zip](#) with build-and-run steps. Download it, open the solution in Visual Studio, press F5 and VisualOperationsStudio comes up on [localhost](#). You need the .NET SDK and the [Wisej-4](#) NuGet package; everything else the prompt asks the assistant to write itself.

The prompt, in full — read it, then edit it if you want a different sample

I am working through Module 4 – "Wisej.Web.Canvas and the HTML5 Canvas 2D Model" – of the LearnWisej Drawing & Painting in Wisej.NET course, and I want to build the module sample with you.

Start by reading the module specification. It is a single Markdown file holding the full lesson, the lab brief and the deliverables:

<https://learnwisej.com/assets/courses/drawing-and-painting/specs/module-04.md>

Then build a complete Wisej.NET 4 project I can open in Visual Studio:

1. Project "VisualOperationsStudio", C#, net8.0 or later, NuGet package Wisej-4 4.1.x, Kestrel host calling `app.UseWisej()`.
2. Build what the specification's lab asks for: Build a Canvas playground page in VisualOperationsStudio that ports a set of browser Canvas 2D examples to the documented Wisej.Web.Canvas API. Drive everything from a `DrawPlayground` method called by the Canvas Redraw event so the scene returns after a resize, and start it with `ClearRect` over the full surface. Render a stroked path, filled rectangles, a placed caption, a linear gradient and a radial gradient, then wrap each transformed section in `Save()` and `Restore()` and demonstrate `Translate`, `Scale`, `Rotate` and `SetTransform`. Add a clipped region, an alpha section, a dashed line and a shadow, port one browser example that rotates by $\text{Math.PI} / 4$ into the degree value Wisej expects, and keep `LiveUpdate` off for the main render while showing it in one small progressive example.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - Canvas playground whose entire scene is drawn by a `DrawPlayground` method called from `Redraw`
 - Paths, rectangles, a placed caption, a linear gradient and a radial gradient using documented Wisej.Web.Canvas members only
 - `Save()` and `Restore()` around every `Translate`, `Scale`, `Rotate` and `SetTransform` section
 - Clipping, alpha, a dashed line and a shadow, plus one radians-to-degrees port written out in a comment
 - A `LiveUpdate` demonstration kept separate from the main render, with a note on when it is appropriate
4. Follow the specification wherever it names a Wisej.NET class, property or event – use exactly that one. Where it leaves a choice open, tell me which API you picked and why.
5. Keep the server authoritative: state and validation live in .NET, and the browser renders and reports. Never move a decision into JavaScript because it is easier there.
6. Make the result demonstrable: I should be able to run it and see the behaviour this module is about, including what happens when something goes wrong.

Before you write any code, restate in your own words what you are going to build and where each piece will live, so I can correct the design while it is still cheap.

When the code is done, package the whole solution as a single .zip I can download, and give me the exact steps to build and run it.

Finally, answer these questions from the specification, using your own implementation as the evidence:

- Your playground looks right until a colleague resizes the browser, after which only the last shape you drew is visible. What does that tell you about where the rest of the scene was coming from?
- You need the pixel width of a caption so a badge can be sized to fit it. The Wisej.NET Canvas API does not list a text-measurement member – describe two ways to get the layout you

want anyway, and what each one costs.

– A reviewer asks why the clip in your state section is wrapped in `Save()` and `Restore()` when the shapes after it are drawn somewhere else entirely. Answer them.

Ask me about anything that is ambiguous before you start.

Then review what came back

Code that compiles is not yet code that is right. The interesting questions are design questions: is this the Wisej.NET control the job actually calls for, does it follow the lifecycle the framework gives you, is the authoritative state on the server rather than parked in the browser because that was convenient? Read the result as a reviewer, not as a customer.

This module gives you the checklist. Answer each one against the code you just received:

- ✓ Your playground looks right until a colleague resizes the browser, after which only the last shape you drew is visible. What does that tell you about where the rest of the scene was coming from?
- ✓ You need the pixel width of a caption so a badge can be sized to fit it. The Wisej.NET Canvas API does not list a text-measurement member — describe two ways to get the layout you want anyway, and what each one costs.
- ✓ A reviewer asks why the clip in your state section is wrapped in `Save()` and `Restore()` when the shapes after it are drawn somewhere else entirely. Answer them.

Anything you had to correct is worth writing down. A short note of the conventions and decisions you settled on, pasted into the first message of your next session, is worth more than any single clever prompt.



No reference sample for this course yet — yours is the first

The [LearnWisej samples repository](#) carries worked builds for most courses, but not yet for this one. That means there is nothing to compare against and no shortcut: the specification and your own review are the whole safety net. Keep what you build — and if it turns out well, we would like to see it.

Three prompts to take it further

Point your assistant at the project you just built and try these. Each one is a review-and-extend exercise, not a rewrite.

- ✓ **Swap a piece without moving the seams.** *"Pick the part of this VisualOperationsStudio project that is most obviously a stand-in for something real — a fake data source, an in-memory store, a simplified service — and replace it with a production-shaped implementation. The contract either side of it must not change: same types, same calls, same behaviour under failure. Tell me what the real implementation forced you to handle that the stand-in did not."*
- ✓ **Make it survive a bad day.** *"Review this VisualOperationsStudio project for the failure modes it does not handle yet — bad or missing input, a dependency that is slow or unavailable, work arriving after the user navigated away, and two actions racing each other. For each one, tell me first whether it is a real risk in Wisej.NET 4 and why, and only then write the fix. Where you decide something needs no handling, say so and explain the reasoning rather than adding defensive code."*

- ✓ **Add the second case.** *"Extend this VisualOperationsStudio project with a second case that follows the same pattern as the first — another entity, screen or control of the same shape, wired the same way. Factor out what is genuinely shared into a base class or service, but only where it earns its place; two similar things do not always want an abstraction. Show me the diff and justify each thing you extracted."*



Carry it into the lab

The hands-on lab that closes this module asks for the same deliverables you just had the assistant produce. Use what you have here as the shape of your answer and the evidence behind it — not as the answer itself. The lab is graded on your reasoning, and that part has to be yours.