

APPENDIX J: Claude Artifacts: What They Are and How to Build One

If you clicked any of the interactive links embedded in this book — the [CORRECT Method Trainer](#) in Chapter 4, the [GenAI Policy Drafting Tool](#) in Chapter 5, the [GenAI Use Disclosure Form](#) in Chapter 8 — you have already used a Claude artifact. All three are running, interactive tools that were built inside Claude’s chat interface, require no technical background to create, and can be shared with a single link. This appendix explains what they are and how to build your own.

What an Artifact Is

When you ask Claude something in the standard chat interface, the response appears in the conversation as scrollable text. An artifact is different. Claude generates it in a separate panel to the right of the chat window, and it renders there as a live, functional object rather than raw text. You can interact with it directly, fill in fields, click buttons, and move between sections, all without leaving the browser.

The underlying mechanism is that Claude writes code (typically HTML, CSS, and JavaScript) and the artifact panel renders that code as a working web page or interactive tool. You never see the code unless you want to. From your perspective, you describe what you want, Claude produces it, and you interact with the result.

Claude can generate several kinds of artifacts. **HTML web applications** are the most flexible type and the most useful for legal educators. **Markdown documents** are structured text files with headers, tables, and formatted lists, useful for templates and reference guides. **Code artifacts** are working scripts in any language Claude supports. **SVG graphics** are scalable diagrams and illustrations. **React components** are a more technical option used in modern web development. For the purposes of this appendix, the focus is on HTML artifacts, because they run directly in a browser, require no technical knowledge to use, and can be shared with anyone via a link.

How Sharing Works

Once an artifact is generated, Claude provides a Share option in the artifact panel. Clicking it produces a public link in the format *claude.ai/public/artifacts/[unique identifier]*. Anyone with that link can open and interact with the artifact in a browser. A Claude account is not required to use the tool, though it is required if the artifact itself calls Claude for AI-generated feedback (as the CORRECT Method Trainer does). Those are the links embedded throughout this book. They were created exactly this way: describe the tool, iterate until it works, click Share, copy the link. The process from first prompt to shareable link typically takes between fifteen minutes and an hour, depending on the complexity of what you’re building and how much refinement it needs.

How to Build an Artifact

The single most important thing to understand is that you do not need to know how to code. Claude writes the code. What you need is a clear description of what you want the tool to do. The clearer and more specific your description, the less iteration you'll need.

Step 1: Open a new Claude conversation

Go to claude.ai and start a new chat. You'll need a Claude account; a free account works, though usage limits mean you may hit a cap if you're doing a lot of iteration in a single session. A Pro subscription (\$20/month at time of writing) removes that friction.

Step 2: Enable artifacts

In most Claude interfaces, artifacts are enabled by default and generate automatically when Claude produces something that fits the format. If you're not seeing the side panel, check your settings: click your profile icon at the bottom left of the screen, select *Settings*, then *Capabilities*, and confirm that **Artifacts** and **AI-powered artifacts** is toggled on.

Step 3: Describe the tool

Tell Claude what you want to build, who will use it, and what it should do. Specificity matters here more than anywhere else in the process. A vague prompt produces a vague artifact; a precise one produces something you can actually use.

Compare these two prompts:

Weak: Make me an interactive tool for students.

Stronger: Create an interactive HTML tool that helps first-year law students practice applying the IRAC framework to a torts fact pattern. Present a fact pattern involving a slip-and-fall. Below it, display four labeled text fields: Issue, Rule, Application, and Conclusion. Include a 'Check My Work' button. When clicked, display a model analysis for each element, with a short explanation of what to look for. Use a clean, professional layout with a white background and a dark blue header.

The second prompt produces something you could actually embed in your course materials. The first produces something generic that you'll spend the next forty minutes fixing.

See below for the [IRAC Practice Tool](#) created from this prompt. I also included the iterative changes I made to the tool.

Step 4: Include any content Claude needs

If the tool should contain specific material, paste it directly into your prompt. Your course policies, a particular fact pattern, a rubric, a list of jurisdictions, a set of exam questions: Claude will incorporate whatever you provide. The GenAI Policy Drafting Tool was built this way, seeded with a set of policy options and structured to take user inputs and generate a customized draft.

Step 5: Review and iterate

Your first artifact will probably be close but not perfect. That is normal, and it is not a sign that something went wrong. Iteration is the process. Look at what Claude produced and tell it specifically what to change.

Useful revision prompts include:

- The fact pattern is too simple. Make it a comparative negligence scenario.
- Add a progress indicator at the top so students can see how far they are.
- The model answer is showing immediately. It should only appear after the student clicks the button.
- Change the color scheme to match the university's colors. The primary color is #003366 (or "beige")
- Add an export button that lets students copy their responses to the clipboard.

Claude retains the context of your artifact across the conversation, so you can describe changes without re-explaining the whole tool each time. If the revision breaks something that was working before, just say so: "that introduced a bug, the Check My Work button stopped working." Claude will fix it.

Step 6: Share the artifact

Once the artifact does what you want:

- Click the **Share** icon in the artifact panel (the box-with-arrow icon at the upper right of the panel).
- Select **Publish** or **Share publicly**.
- Copy the link Claude generates.
- Paste it into your syllabus, Canvas course shell, assignment instructions, or wherever students will access it.

The link is permanent as long as the artifact remains published. You can unpublish it at any time by returning to the artifact and toggling the sharing setting off.

A Full Example Prompt

[IRAC Practice Tool for 1Ls](#)

IRAC Practice Tool

Choose a subject and difficulty, then analyze the fact pattern using IRAC.

1L SKILLS

SUBJECT: Property ▼

DIFFICULTY: Basic Intermediate Advanced

[↻ New Fact Pattern](#)

FACT PATTERN

Marguerite Holloway has lived in a rural farmhouse in Dunmore County for thirty-two years. When she purchased the property in 1992, her deed described the boundary as running "along the creek bed to the old oak tree, then north to the stone wall." Over the past decade, the creek has gradually shifted fifteen feet eastward due to natural erosion, and the oak tree fell and was removed by a neighbor in 2019. Marguerite has continuously maintained, mowed, and used the disputed strip of land between the original creek bed and its current location, treating it as her own without objection from anyone. Her neighbor, Desmond Pruitt, recently purchased the adjacent parcel and, relying on a survey that follows the creek's current course, has begun erecting a fence that cuts through what Marguerite considers her garden. Desmond argues that because the deed references the creek as a monument, the boundary must follow the creek wherever it moves. Marguerite seeks a declaratory judgment quieting title to the disputed strip in her favor and a permanent injunction requiring Desmond to remove the fence.

Each field requires a minimum of 15 characters. 0 of 4 fields complete

I Issue
State the precise legal question the court must resolve.

What is the legal question raised by these facts?

A prompt at this level of specificity produces a working, deployable tool in a single generation. Claude will generate the fact patterns. If you want a specific fact pattern, paste it into the prompt in place of the parenthetical instruction.

I started with the prompt above, and suggested the following tweaks:

- Signal that a minimum of 15 characters in each field is required before feedback can be given.
- The fact patterns are taking too long to generate. How can you fix this?
 - Answer from Claude:
 - **Switched to Haiku.** Haiku is Anthropic's fastest model. It's built for speed over depth. For generating a 4-sentence torts fact pattern and a structured IRAC analysis, it's more than adequate and noticeably quicker than Sonnet.

- **Split into two calls.** The fact pattern and the model answer now fetch separately. The fact pattern is a short, focused call and should appear in a couple of seconds. The moment it lands, the model answer starts fetching in the background while the student reads and starts writing. By the time all four fields are filled, the analysis is almost always ready. If it's still loading when the student finishes, a small spinner appears next to the status note: "Preparing model analysis almost ready", so they know it's coming rather than wondering if something broke.
- After the first deployment, I asked Professor Nancy Rapoport to take a look. She suggested the following question be added: What facts, if proven, might change your analysis and conclusion? I asked Claude to revise. It took a few iterations to ensure "Check My Work" worked.

Claude fixed the issues outlined above, and created a new Artifact for publication and sharing. Note that on occasion, you have to ask Claude to create a new URL and Artifact to be published. To unpublish old Artifacts (heads up, it's tedious):

1. Open the artifact (click on it in the conversation to expand it in the right panel).
2. Click the **share icon** (the arrow-out-of-box icon in the artifact toolbar).
3. Toggle off the "Publish" or "Share publicly" option — this revokes the public link immediately.

Note that I also developed a CustomGPT for this purpose called "[UNTD CoL: Practicing IRAC](#)."

Which is better - Claude Artifact or a CustomGPT? It's a matter of opinion, really. Take a look at both and consider which your students would prefer. Alternatively, if some of your students have subscriptions to ChatGPT, then give them the CustomGPT. If others have Claude subscriptions, give them an Artifact.

Whose tokens are used?

When you share an artifact, the token usage it charged against the user, not against you as the creator.

- When you're building/testing: Usage counts against your plan
- When others use your shared artifact: Usage counts against their plan, not yours

So sharing is completely free for you (just as it is with CustomGPTs) - whether 10 people or 10,000 people use your artifact, you pay nothing for their usage. Each user needs their own Claude account and their usage counts against their own subscription limits.

For Team and Enterprise plans, when you share artifacts within your organization, team members can use them without additional costs to anyone.

What Artifacts Cannot Do

A few constraints are worth knowing before you build something that depends on capabilities the format doesn't have.

Artifacts cannot connect to external systems.

A Claude artifact cannot log into Westlaw, access your Canvas gradebook, send emails, or retrieve information from the internet. It is self-contained. Everything the tool does, it does with whatever Claude built into it.

Artifacts do not store data between sessions.

If a student fills out a form in an artifact and closes the browser, nothing is saved. There is no backend, no database, and no record of who used the tool or what they entered. If you want students to submit their work, build in an "Export" or "Copy to Clipboard" button so they can paste their responses somewhere else, such as a Canvas submission box.

Artifacts are public once shared.

The share link is accessible to anyone who has it. Do not build a tool that displays student names, grades, or other sensitive information if you plan to share it publicly.

Artifacts occasionally need to be rebuilt.

Because Claude generates artifacts from code it writes on the fly, a tool that worked in January may behave differently if Claude's underlying model updates. Keep your original prompt saved somewhere. Regenerating an artifact from a good prompt takes about thirty seconds. My practice is to do a quick weekly check to make sure my artifacts are still working.

Ideas for Legal Educators

The tools in this book were not the product of any specialized technical knowledge. All it took was a little curiosity and a willingness to fail. After all, what could happen? They were built through iteration: describe what you want, review the result, refine the description, repeat. If you have a clear sense of what a good learning tool should do, you have everything you need to build one. A few starting points worth considering:

An interactive self-assessment rubric. Students rate their own work against your rubric criteria before submitting. Seeing the rubric as a functional checklist, rather than a static document, changes how students engage with it.

A fact-pattern generator. Seed the artifact with a set of legal issues and ask Claude to build a tool that generates random fact patterns from that pool on demand. Useful for practice, for study groups, and for students who want to drill on a particular area before an exam. See the [IRAC Practice Tool for 1Ls](#).

A memo-review checklist. A tool that walks a student through the conventions of objective memo writing, field by field, prompting them to confirm that each element is present before they submit. The checklist becomes the first line of quality control, before the work ever reaches you.

A course-policy generator. This is exactly what the [GenAI Policy Drafting Tool](#) in Chapter 5 does. Describe a set of policy options, build a form that lets users specify their preferences, and produce a draft policy tailored to their selections. You could build the same structure for other parts of your syllabus.

A disclosure form. The [GenAI Use Disclosure Form](#) in Chapter 8 generates a formatted PDF students can complete and submit. A similar artifact could work for any form-based assignment requirement where you want students to produce a structured document rather than a freehand narrative.

None of these require a developer, a technology budget, or a request to your institution's IT department. They require a clear description of what you want the tool to do and enough patience to iterate until it does.