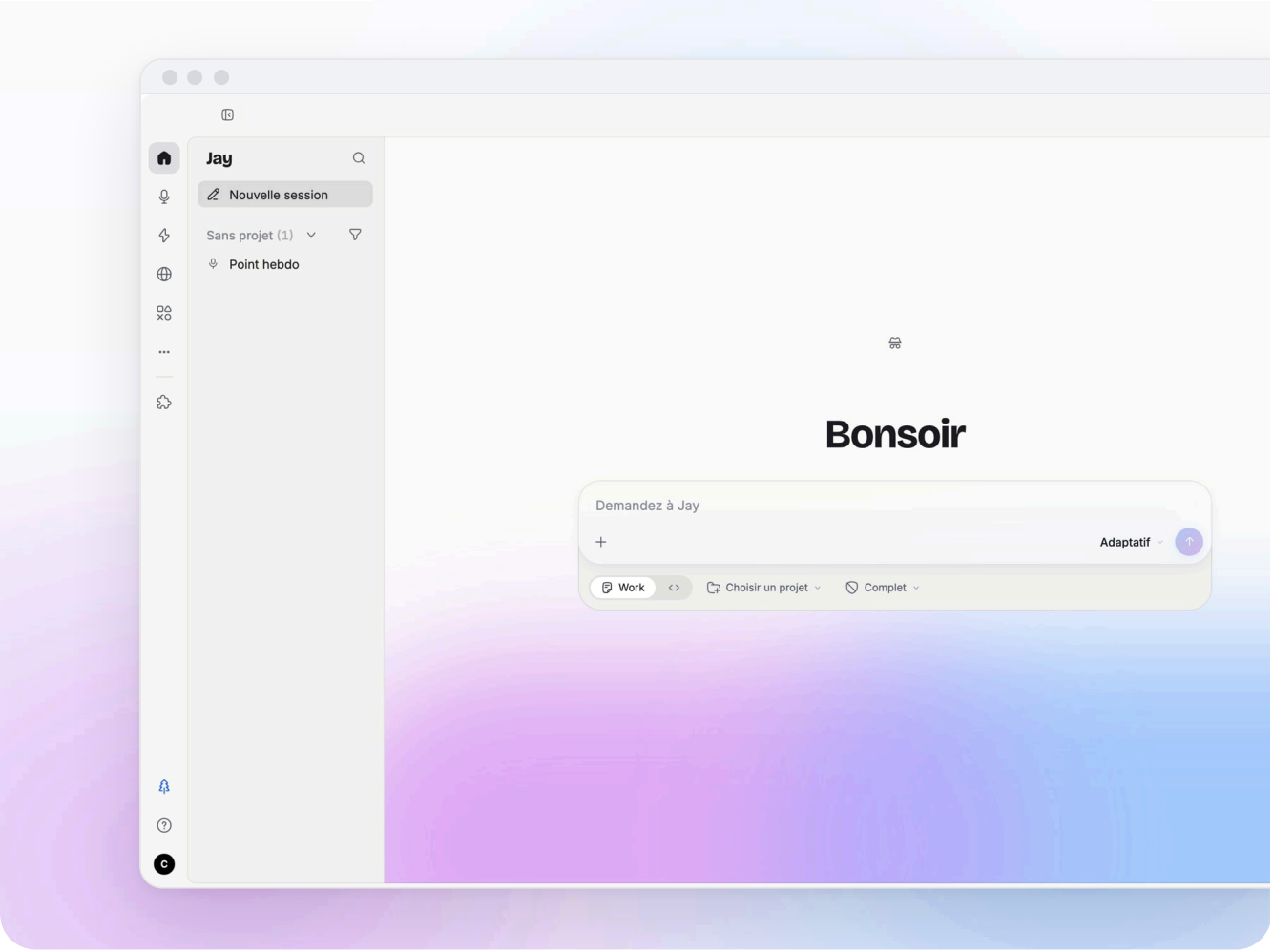


COMPLETE GUIDE · NOVEMBER 2026 EDITION

Jay, the complete guide

Jay is an agent that works on your Mac: it reads your folders, writes your documents, drives your browser and your apps, and comes back with finished work. This guide teaches you how to hand it real work, from the first message to the flow that runs without you.

Jay, the agent that works on your Mac · a dev-id guide

Contents

Eighteen chapters, from your first session to the flow that runs while you sleep.

I · THE FOUNDATIONS

01 Getting started with Jay 4

02 The art of the brief 10

03 The settings that matter 16

04 Projects: one folder, one JAY.md 21

II · CREATE

05 The Atelier: documents, spreadsheets, presentations 25

06 Images, brand kits and video 29

07 Ideation 33

III · ANALYZE

08 Reading your documents and data 36

09 The Browser and the web 40

IV · CONNECT AND DELEGATE

10 Applications and connectors 44

11 Skills 48

12 Meetings 52

13 Delegating: permissions and background tasks 55

14 Flows and watches 60

15 Jay and code 64

V · THE METHOD

16 Complete workflows 68

17 Best practices and security 72

18 Your 7-day plan 76

APPENDICES

— Before you start 2

— Glossary 79

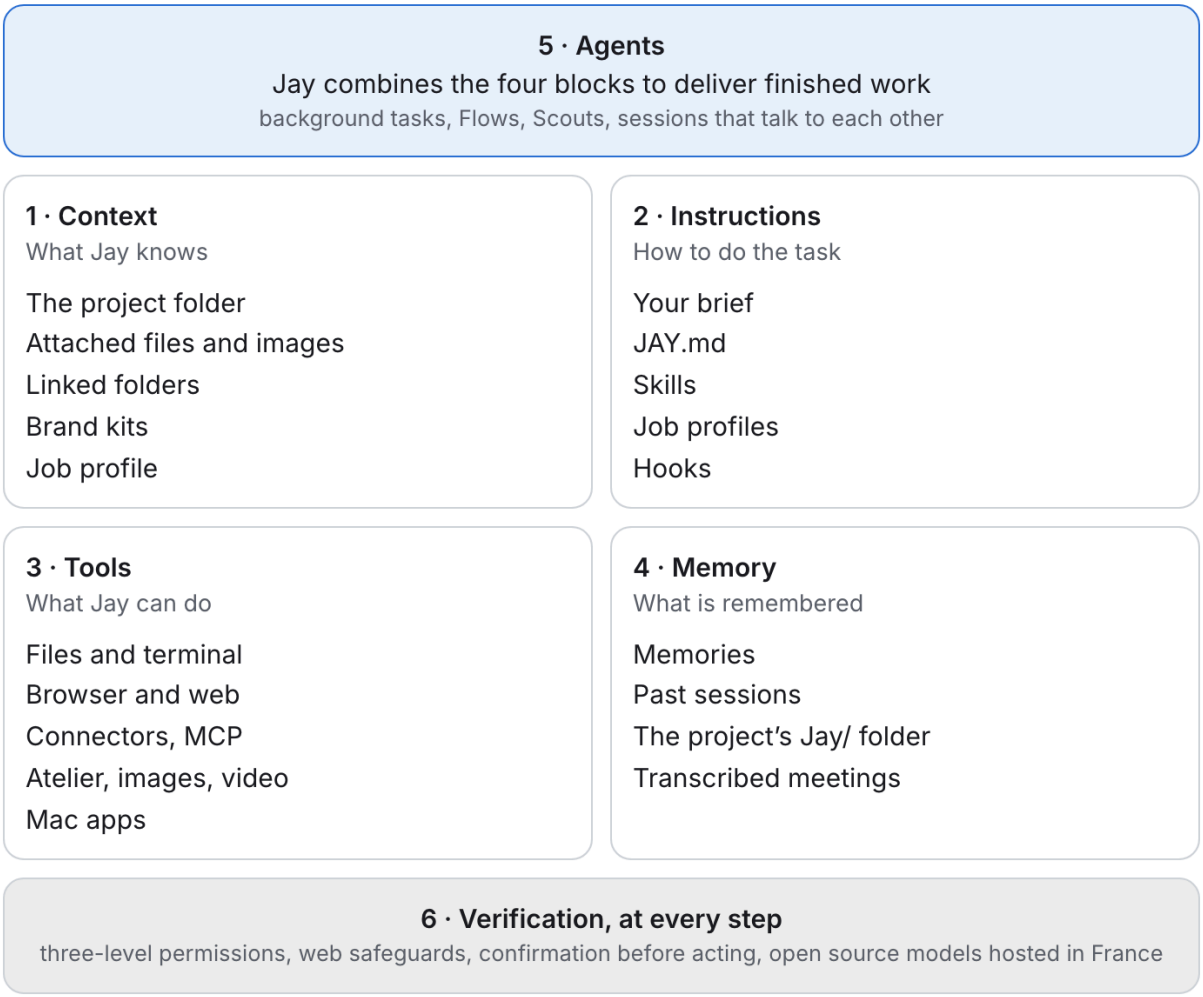
ONE PIECE OF ADVICE BEFORE YOU START

You won't read sixty pages in one sitting, and that's not the point. Each chapter stands on its own and takes a few minutes. Keep this guide close, and come back to it whenever a new Jay feature ships: it fits into one of the six building blocks, and you'll already know what to do with it.

Before you start

> You don't need to keep up with every new Jay feature. Behind its places, tools and settings, there are six simple building blocks, and they don't change. When a feature arrives, it fits into one of them: once you know where it goes, you already know what it's for and when to use it.

Six building blocks: every Jay feature fits in exactly one



Context (what Jay knows), Instructions (how to do the task), Tools (what it can do), Memory (what it remembers): agents combine these four middle blocks. Verification secures all of them. Each chapter tells you which one it strengthens.

How to read this guide

YOU ARE…	READ…	WHY
New to Jay	Chapters 1 to 4, in order	The foundations: the app, the brief, the settings, projects. Two hours, and everything else depends on them.

YOU ARE…	READ…	WHY
Already a user	What you're missing, starting with 1, 10, 13 and 14	The bar has changed (Work and Code, Ask / Auto / Full access), Customize has replaced Extensions and Skills, and watches have arrived: that's where the hours are saved.
In a hurry	Chapter 16, one workflow	Eight complete sequences to copy onto your week.

Guide signposts. Prompt cards are meant to be copied as is: just replace the parts in brackets, like [your job] or [the folder]. They speak to Jay directly, because you're the one talking to it. A **Tip** is a shortcut few people know. A **Watch out** flags a common mistake, a risk to your data, or a limit worth knowing.

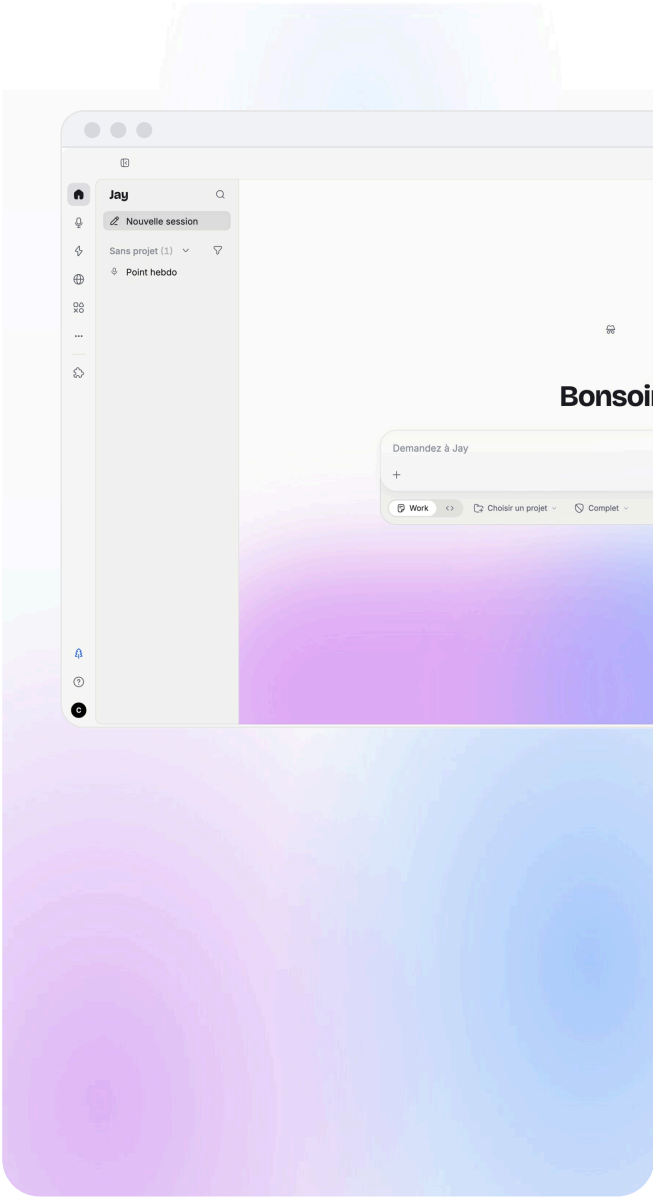
About versions. This guide describes Jay as it is in mid-November 2026. The open source models behind each mode change over the months, and menus move: when a detail differs in your app, trust the app. The six building blocks stay the same.

I · THE FOUNDATIONS

CHAPTER 01

Getting started with Jay

Jay is an agent that lives on your Mac. It reads, writes, analyzes, codes, and above all it acts: in your folders, in your browser, in your apps and in your online tools. It runs on open source models hosted in France, and nothing you give it is sent to the companies that make those models.



IN THIS CHAPTER

01 What changes with an agent on your computer

02 Installing it

03 Which plan to choose

04 Modes: which one for which task

05 A tour of the interface

06 Your first hour with Jay

> Most people start by asking it questions, one at a time. That's the use that pays off the least. Jay shows what it can really do when you open a folder for it and hand it a complete job: the goal, the context, the tools, and the freedom to come back with a finished result.

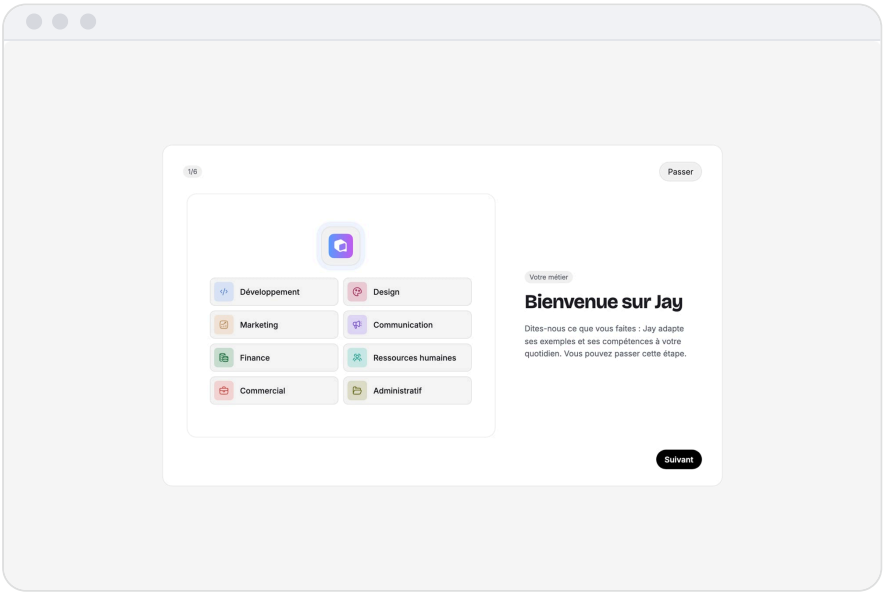
What changes with an agent on your computer

AN ONLINE ASSISTANT	JAY
You send it a file	It opens the folder itself, reads what's inside and saves its work there
It gives you text to copy	It writes the document, spreadsheet or presentation, and the file exists on your disk
It describes what you should do on a website	It opens the site in its Browser, clicks, fills in, and shows you
It forgets between conversations	It remembers, and finds your past sessions and meetings
It stops when you close the tab	Its Flows run on schedule, even with the window closed

This power comes with a trade-off: Jay can modify files and act on your behalf. That's why every action goes through an access level you choose: Ask, where Jay asks before each action, Auto, or Full (chapter 13).

Installing it

Jay is a macOS app for Apple Silicon Macs (macOS 12 or later). You install it from the DMG file provided by dev-id, like any other app: drag it into Applications, open it, sign in.



The welcome tour on first launch: six screens, starting with your job.

1. **Onboarding.** On first launch, Jay introduces itself in six screens: your job, one session for everything (with the modes), Jay working for you (the Browser, your apps), you staying in control, your accent color, then your profile. Don't skip the first or the last one: your job and your profile shape every answer that follows (chapter 3).
2. **Updates.** Jay updates itself. In Settings, System, Updates, you choose between "Install automatically" (when you quit or when Jay is idle) and "Ask me". A badge at the bottom of the icon bar signals that a version is ready, and "What's new" opens once after each update to tell you what changed.
3. **The lock.** If other people get near your Mac, turn on "Lock with Touch ID" in Settings, System: on opening, or on opening and whenever the Mac locks.

Which plan to choose

Every Jay plan unlocks the same features: all the modes, images, the Browser, DOPE. The only thing that changes from one plan to the next is the volume of work. Paid plans are monthly, with no commitment: a cancellation takes effect at the end of the paid month. Prices are on the website and in Settings, Account. Oreus subscribers have their own plan, purchased from Oreus on top of their subscription.

PLAN	WHAT IT UNLOCKS
Jay Trial	All features, with a one-time allowance of one million free tokens, with no time limit, to try Jay on real work
Jay Pro	All features, with a weekly and monthly volume for daily use
Jay Ultra	The same, with much more volume
Jay Giga	The same, with the most volume, for people who run Jay all day
Oreus	For Oreus subscribers, purchased on app.oreus.ai: Jay and your Oreus agents in the same bar; DOPE is not included

MY RECOMMENDATION

Start with Jay Trial and spend it on a real task from your week, not on test questions: it's the only way to find out what Jay saves you. Move to Jay Pro when the allowance runs out, then upgrade only if you run short on volume.

WATCH OUT

Each paid plan includes a weekly and monthly volume, measured in tokens; the Jay Trial allowance, on the other hand, does not refill. Powerful modes, large folders and long tasks use it up faster. Expert mode has its own cap: at most half of the volume; once it's reached, only Expert waits for the reset, and the other modes keep going. The Settings, Usage page shows where you stand and when the counter resets, and your account menu shows the Expert gauge; you can ask to be warned before the limit in Notifications. The Mini Jay companion and DOPE coding agents draw from the same volume.

Modes: which one for which task

Jay isn't a single model but a family of modes. Each mode has a stable name and a promise; the open source model behind it is shown in the picker, and it can change when a better model comes along. The more powerful a mode, the slower it is and the more of your volume it uses. The right habit: Adaptive for everyday work, Thinking or Expert for what really matters.

MODE	ITS PROMISE	WHEN TO USE IT
Adaptive	Picks the right mode for each request	By default. You stop thinking about it.
Versatile	For everyday work	Routine writing, tidying up, most tasks; at Minimal effort, quick answers
Thinking	Handle complex tasks	Analyses, decisions, calculations, long tool-heavy tasks
Expert	Most capable, for demanding projects	What has to be right the first time

The old Fast mode has merged into Versatile: choose Versatile at Minimal effort to get the same instant answer. A session left on Fast reopens on its own in Versatile Minimal.

Effort. For the modes that reason, a second setting, effort, decides how much time the model spends thinking before it answers: Minimal, High (the default), Max. In Adaptive, it's called Automatic and sets itself. Raise the effort for a decision, lower it for a rewording. A separate row, **Ultra intelligence**, makes Jay follow a stricter method: it rereads its work with a critical eye when a doubt remains, and runs the tests itself for a change it hasn't verified. It uses more volume; save it for work that can't afford mistakes.

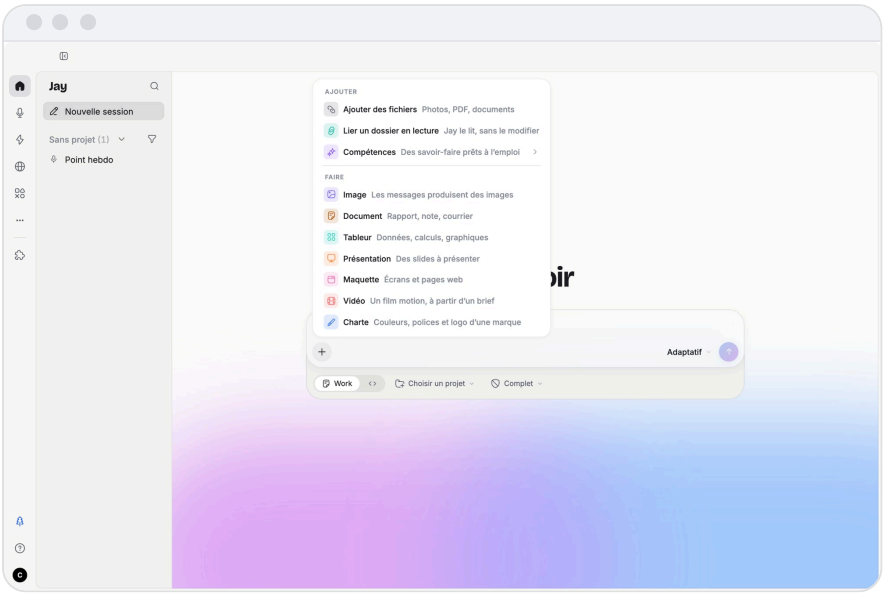
Images. The picker also has an Image models group: Fast (FLUX.2-klein-4B) to iterate, Thinking (Ink 1.4) when the image has to carry crisp text (chapter 6).

TIP

When the servers are busy, Jay tells you: a "Slow right now" line appears while you wait and in the picker. It's not your Mac, and it passes; for a simple question, Versatile waits less.

A tour of the interface

Jay's window is a frame and a card. The frame is the icon bar on the left: the places. The card is everything else: the place's sidebar and the work area.



The "+" in the bar: Add, then Do. Below it, the Work / Code switch, the project and the access level.

ELEMENT	WHAT IT IS	WHAT YOU DO THERE
The icon bar	Jay's places: Home, Meetings, Flows, Browser, Atelier, then, below the divider, Customize. All are pinned by default; the "⋯" button lists them and lets you unpin some.	Switch places. At the bottom: Mini Jay, the update, your account and Settings.
The sessions rail	In Home: your projects, your pinned sessions, your sections.	"New session", "New meeting".
The message bar	A single bar, for everything. It only shows what's there: your files, the project, a mode that's on, a card from Jay.	The "+" adds (files, linked folder, Skills) and does (Plan, Image, an Atelier piece). Below it: the Work / Code switch, the project and the access level: Ask, Auto, Full.
The mode picker	The right engine for the session.	Change mode and effort. The bottom of the menu reminds you where the models run.
The drawer	The right half of the session; a row of buttons in the header opens each panel.	Session info, Activity, Web, Files, Code, Terminal, Atelier, and, depending on the session, Design System or Brand kit.
⌘K	The single search.	Find a session, a meeting, a skill, an app, even archived.
Mini Jay	The companion, the question mark at the bottom of the bar, or ⌘⇧J.	A quick question, about Jay or anything else, without keeping anything.

Work or Code. There is now only one type of session: you chat in it, write documents in it, code in it. A switch, at the top of the drawer under the bar, only changes what it shows. In **Work**, the bar offers Atelier pieces and images; in **Code**, it shows the branch, the isolated copy, the diff and a Terminal button, and the "+" offers Plan. Nobody has to choose before writing: a session whose folder is a Git repository starts in Code, all others in Work, and you switch whenever you like, without losing anything. Each position keeps its own access level (chapter 13). A project without a folder also exists: it's for organizing sessions.

TIP

Hover over an element of the bar (the mode, the project, access, the context gauge): a help card tells you what it's for. A long pasted text tucks itself into a "Pasted" card that you reopen with a click; and the microphone dictates in every Jay bar.

Your first hour with Jay

Start with what changes the quality of answers the most, then move straight on to a real task.

1. **Fill in your job profile.** Settings, Jobs: who you are, your rules, the equipment Jay installs for you. Five minutes, and every answer benefits (chapter 3).
2. **Create your first project with a folder.** A client, a recurring report, a monitoring topic. Jay keeps its memory, documents and meetings there (chapter 4).
3. **Connect two tools.** Your calendar and your email, for example, from Customize, Applications. Jay will be able to prepare your day (chapter 10).
4. **Run a real test, in Ask access.** Take a real task from your week, the longest one, and hand it entirely to Jay with the brief from chapter 2. A Work session starts in Full access: for this first test, switch its access picker to Ask, and reread every permission card. That's how you learn what it does.
5. **Open the drawer.** The Activity panel shows what Jay is doing right now, the Files panel what it has written. A single block closes each turn: the summary, the question it's asking you, the files produced, the time and the tokens. You'll always know where it stands.

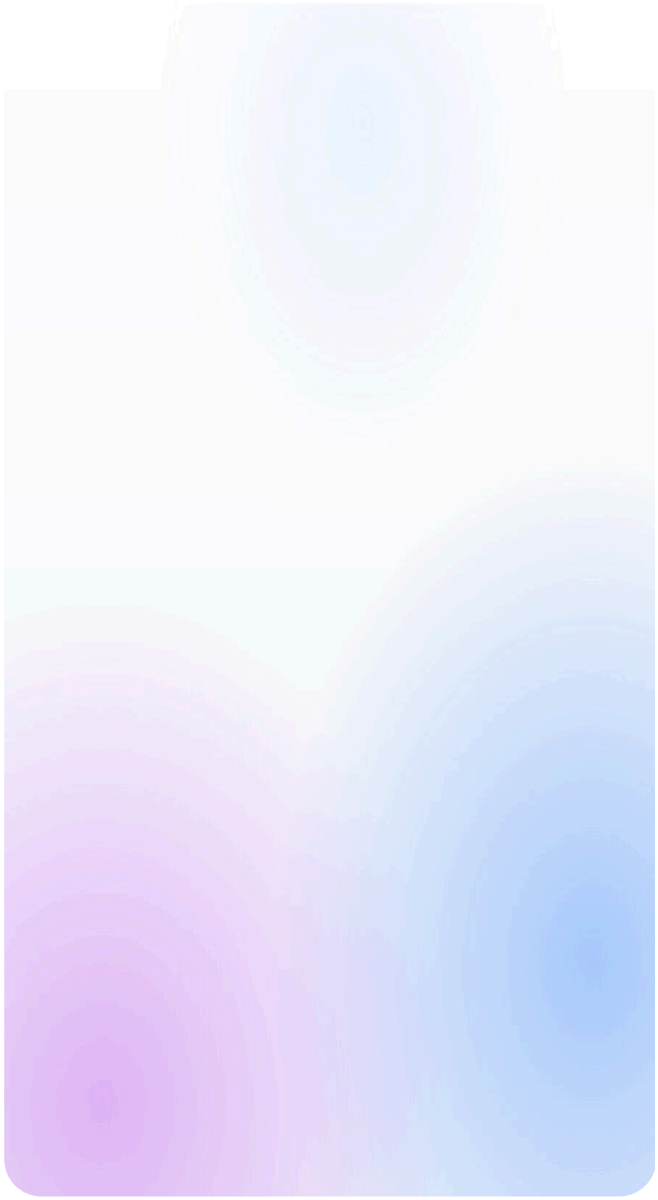
PROMPT **Your very first session**

```
I've just started using Jay. I'm a [your job] and I spend most of my time on [your 3 main
tasks]. My work files are in [folder].
Ask me 5 questions to understand my work. Then suggest the 5 uses of Jay that would save
me the most time this week, ranked from simplest to most ambitious, each with an example
message to send you and what you would need to be able to open (folder, tool, website).
```

CHAPTER 02

The art of the brief

When an answer disappoints, nine times out of ten the request was vague. Jay only knows two things: what you tell it, and what it finds in the folder you've opened for it. A good brief takes one page to learn, and it saves you three rounds of back-and-forth on every task.



IN THIS CHAPTER

01 The 5-point brief

02 Before / after

03 Ten techniques that change the answers

04 The skeleton of a complete brief

05 The meta-prompt: have Jay write the brief

06 Library: 14 ready-to-use prompts

> Think of a brilliant new colleague who arrives on a Monday morning. "Do the reporting" will get you something generic. "This month's reporting for the committee, in the format of September's, which is in the folder, with the anomalies up top" lets them be useful from the first hour. Jay is that colleague, with one advantage: it has already opened the folder.

The 5-point brief

ELEMENT	THE QUESTION TO ASK YOURSELF	EXAMPLE
1 · Goal	What will the result be used for, and by whom?	"An email so my partner approves the quote without calling me back."
2 · Context	What can't Jay guess, even by reading the folder?	"The client has already turned it down twice over price."
3 · Deliverable	What form, what length, and where should it land?	"A document in the Atelier, two pages, then a PDF in the Quotes folder."
4 · Criteria	How will you recognize a good result?	"Every figure points to a row in the spreadsheet."
5 · Example	Do you have a past result you liked?	"Use the template from report-september.docx."

The third point is specific to Jay. On your computer, a result has an address: a file in a folder, a piece in the Atelier, a draft in your email. Say it, or Jay chooses for you.

Before / after

BEFORE

```
Sort my invoices.
```

AFTER

```
In the Downloads folder, find the PDF
invoices received since October 1. Move
each one to Accounting/2026/[month] and
rename it "YYYY-MM-DD supplier amount".
Don't delete anything. At the end, give me
the list of what you moved and of anything
that seemed doubtful.
```

The second request takes thirty more seconds to write. It saves you from checking forty files one by one.

Ten techniques that change the answers

Ranked from simplest to most advanced. The first three are enough for most requests.

1. **Give an audience and a stake.** "For a busy buyer who has to decide tomorrow" sets the tone and level of detail far better than a role like "you are an expert".
2. **Get yourself interviewed.** End with "Ask me the questions you need first". Jay often spots what's missing better than you do.
3. **Specify the deliverable.** A table, a 120-word email, a two-page document in the Atelier, a spreadsheet with specific columns. With a precise format, the result is usable as is.
4. **Point to an example in the folder.** "Do it like report-september.docx" beats a paragraph of instructions. Jay picks up the tone, the structure and the formatting.
5. **Separate the ingredients.** Wrap each block in angle-bracket tags: <context>, <document>, <instructions>. Jay no longer confuses your instructions with the text to process.
6. **Have it read before it writes.** "Before answering, read the last three meeting notes in the folder." On your computer, the information is often already there: ask Jay to go get it.
7. **Ask for a plan before action.** For anything that touches files, turn on Plan mode (the "+", or /plan) or write "Show me your plan and wait for my approval". A misunderstanding at the start repeats across every file.
8. **Move up to Thinking mode to decide.** Analyses, calculations and hard choices benefit from Thinking or Expert mode; everyday work stays in Adaptive (chapter 1).
9. **Edit your first message.** If the result misses the mark, edit the original request instead of piling up corrections. The session stays clean.
10. **Ask for a self-critique.** "Reread your work like a demanding expert: what's weak, wrong or missing? Then fix it." Quality goes up a notch, right away.

The skeleton of a complete brief

PROMPT **Reusable template**

```
<context>
I'm [your role]. This work is for [audience], who needs to [decision or action].
[What Jay needs to know: constraints, history, stakes]
</context>
<task>
[What you want to get, in one sentence]
</task>
<deliverable>
[Form, length, tone, language, and where the result should land]
</deliverable>
<criteria>
A good result: [what makes it excellent]
</criteria>
If you're missing a piece of information, look for it in the folder first. If you can't
find it, ask me your questions before you start.
```

The meta-prompt: have Jay write the brief

For a task you'll do often, there's no need to hunt for the perfect brief on your own. Ask Jay to build it with you, then store it in the project's JAY.md or in a Skill (chapters 4 and 11).

PROMPT **The meta-prompt**

```
I want a reusable brief for this task: [describe the task].
First, ask me the questions you need to understand it well: goal, audience, deliverable and location, quality criteria, examples in the folder.
Then write the complete brief with tags, and tell me which parts to adapt each time I use it.
```

Library: 14 ready-to-use prompts

Copy, replace the brackets, send. Keep your favorites in a "Prompts" project or in a Skill.

WRITE

PROMPT **The delicate email**

```
Help me write an email to [recipient] to [goal: decline, follow up, deliver bad news].
Context: [situation]. My relationship with this person: [relationship].
Look up our latest exchanges in my email before writing. Suggest 2 versions, one direct and one more diplomatic, 120 words max each. No empty phrases. Prepare them as drafts, don't send anything.
```

PROMPT **Adapt a text for another audience**

```
Rewrite this text for [new audience], who knows [level] about the subject. Keep the ideas, change the vocabulary, the examples and the length ([length]).
<text>[paste the text, or give the file path]</text>
```

PROMPT **The LinkedIn post from a rough idea**

```
Here's a rough idea: [your idea, even jumbled]. Turn it into a LinkedIn post for [audience].
Give me 3 different hooks, then the full post with the best one. 150 to 200 words, short sentences, a single idea, a closing question. I'll post it myself.
```

SUMMARIZE

PROMPT The meeting notes

```
Find yesterday's meeting, "[title]", in my meetings and read its transcript.
Write: 1) the decisions made, 2) the action items with owner and deadline, 3) the open questions, 4) a follow-up email ready to send. Flag anything ambiguous instead of guessing.
Save the notes in [folder] and prepare the email as a draft.
```

PROMPT The long document in 2 minutes

```
Summarize [file path] for someone who has to decide [decision] and only has 2 minutes.
Format: the essentials in 3 sentences, then 5 key points with the page where each appears, then the 3 questions I should ask the author.
```

DECIDE

PROMPT The decision matrix

```
I have to choose between [option A], [option B] and [option C] for [goal]. My criteria, in order of importance: [criteria].
Build a comparison table scored out of 5, explain each score in one line, then give your recommendation and what could change it.
```

PROMPT The pre-mortem

```
It's 6 months from now and [project] has failed. Describe the 7 most likely reasons for this failure, from most to least likely.
For each one: the warning sign to watch for starting now, and one action to prevent it.
```

PROMPT The devil's advocate

```
Here's my position: [your position and your arguments].
Play the smartest devil's advocate possible. Give the 5 best objections, then tell me which ones are truly dangerous for my reasoning.
```

PREPARE AND ORGANIZE

PROMPT Prepare for a meeting

```
I'm meeting [person, company] tomorrow about [goal]. Look up our past exchanges in my email and the related documents in [folder].
Prepare for me: 3 goals for this meeting, 8 questions to ask, the 3 likely objections with an answer for each, and the opening line.
```

PROMPT **The week under control**

```
Here's my task list for the week, in no particular order: [list]. My constraints:
[meetings, deadlines, energy].
Sort them into: do it myself, delegate, hand to Jay, drop. Then suggest a realistic day-
by-day schedule.
```

ANALYZE

PROMPT **Analyze a spreadsheet**

```
Open [file path]. Start by describing what it contains and the quality issues you see:
blanks, duplicates, inconsistencies.
Then answer this question: [your question]. Do all the calculations with code, and show me
the result in a table and a chart.
```

PROMPT **Find the cause of a problem**

```
Here's a problem: [description, figures, timeline]. Ask me questions one at a time to get
to the root cause, like a consultant using the 5 whys method.
When you have enough to go on, give the 3 most likely causes and how to check each one.
```

PROMPT **The honest feedback**

```
Here's [my text, my presentation, my offer]. Give me candid feedback, like a demanding
expert who wants me to succeed.
Format: what works (3 points), what doesn't (3 points, from most to least serious), and
the single change that would have the most impact.
```

ON THE COMPUTER

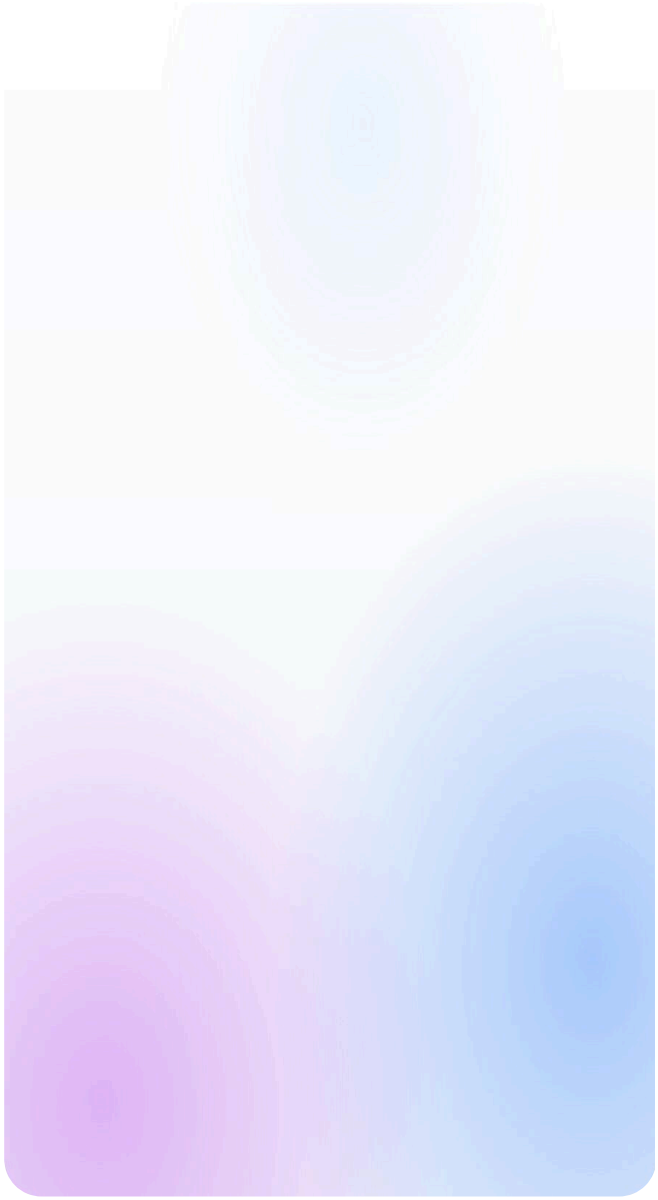
PROMPT **A folder inventory**

```
Go through the folder [path] without changing anything. Tell me what it contains: the file
types, likely duplicates, what hasn't been touched in a year, and what looks misfiled.
Then suggest a tidy-up plan in 5 steps max, and wait for my approval before moving
anything.
```

CHAPTER 03

The settings that matter

Most users never open Settings. Yet that is where Jay becomes an agent that knows you, and where you decide how far it can go on your Mac. Half an hour is enough, and every session after that benefits.



IN THIS CHAPTER

01 The settings that change everything

02 The job profile: your ID card

03 The Settings page, section by section

04 Privacy and incognito sessions

05 Your settings checklist

> Settings is a place like any other: you get there through your avatar at the bottom of the icon bar, and Esc takes you back to where you were. The "Search settings" field is always open.

TIP

Not sure where something is set? Ask Jay, in a session or through Mini Jay: "where do I set the send key?". It answers with a link that opens the right setting.

The settings that change everything

SETTING	WHAT IT CHANGES	WHERE TO FIND IT
Job profile	What Jay knows about you in every session: who you are, your rules, the setup it installs	Settings, Jobs
Default access level	How much autonomy a session starts with: Ask, Auto or Full. Two settings: default access (the one for Code) and default access in Work, set to Full out of the box	Settings, AI behavior
Terminal commands	The commands that are always allowed, and those always refused, even with Full access	Settings, AI behavior
Reply language	Automatic, or fixed: French, English, Spanish, German, Italian, Portuguese	Settings, AI behavior
Browser	What Jay can do on the sites it visits, in the Browser as well as in the Web tab	Settings, Browser
Notifications	When Jay alerts you through macOS, and the warning before the usage limit	Settings, Notifications
Usage	How much you have used this week and this month, by source and by mode	Settings, Usage
Lock with Touch ID	Jay locks when it opens, or when the Mac locks	Settings, System
Keep Mac awake	Keeps the Mac awake at all times, screen open or even lid closed, so your flows and watches keep running	Settings, System, Power & startup

The job profile: your ID card

If you were to change only one setting, it would be this one. A job profile tells Jay who you are and how you like to work, across all your sessions. It also carries a setup: the Skills and connectors that go with your job, which Jay offers to install (chapter 11). You can have several profiles, for example one for your business and one for a nonprofit.

PUT IN	LEAVE OUT
What doesn't change: your job, your industry, your audience, your technical level, your language, the answer format you prefer, the mistakes that annoy you	What changes often: current projects, this month's clients, the instructions for a specific task. All of that goes in the project's JAY.md (chapter 4)

PROMPT **Generate your profile in 10 minutes**

```
Ask me 8 questions, one at a time, to understand my job, my recurring tasks, my audience,
my technical level, my tools, and how I like to receive an answer.
Then write a profile of 150 words maximum, in the first person, that I will paste into my
job profile. Put first whatever changes the quality of your answers the most. End with the
rules you should always follow with me, in 5 lines.
```

EXAMPLE **Profile**

"I'm the marketing manager at a B2B industrial SME. I write for technical buyers. Reply in English, directly, with short lists and concrete examples. My working files are in Documents/Marketing; put everything you produce in the subfolder for the month. When you're not sure, say so. No introductory phrases and no closing summary. If my request is unclear, ask a question before you start. Never delete a file without showing it to me."

The Settings page, section by section

Application: the app and your account.

- **General.** Appearance and accent color, the interface language, input and dictation, incognito sessions by default, the detail shown when a turn ends.
- **System.** Touch ID, updates, storage (clearing caches), app health, power.
- **Notifications.** When Jay notifies you through the macOS Notification Center, quiet hours, and "Warn before the limit": never, or at a percentage of your usage.
- **Keyboard shortcuts.** Click a shortcut to change it.
- **Account.** Your profile, your subscription ("Manage my subscription"), local backup, the installed version, signing out.
- **Usage.** Your quotas (this week, this month, the Expert cap), your footprint, and charts over 7 days, 30 days or 12 months, by source or by mode. The app no longer shows amounts in euros: it counts in tokens, like your plan.
- **Archive.** Sessions put away off the rail, and automatic archiving of inactive sessions. They stay on this Mac, ⌘K finds them, and opening one unarchives it.
- **Lab.** Early-access features, open to a few accounts, with one switch per feature: today, sync across devices, external APIs (your own keys with other model providers) and full Google access. If you don't see this section, your account doesn't have access.
- **Release notes.** What changed with each update.

AI: what shapes Jay while it works.

- **AI behavior.** The reply language, the two default access levels, the default mode, the personality, your custom instructions, global memory, the allowed or refused terminal commands.
- **Models Status.** Real-time model availability, checked every 60 seconds. If a mode is answering poorly, look here before looking anywhere else.
- **Jobs.** Your job profiles.
- **Browser.** Jay's rules on the web (chapter 9).
- **Transcription.** The proper names and terms Jay mishears in meetings, so they get corrected automatically (chapter 12).

Code: what only matters for code.

- **Git.** How Jay writes its commit messages, branches and pull requests (chapter 15).
- **Forges.** Your Git servers: GitHub, Forgejo, GitLab, Bitbucket.
- **DOPE.** Coding agents in a container, with your Jay account, on every Jay plan.

Privacy and incognito sessions

The models. The models that power Jay are open source and hosted in France. Nothing is sent to their publishers, and nothing is used to train a model. The footer of the mode picker says so, and dev-id's privacy policy names every model.

What stays on your Mac. Your sessions, your projects, your pieces and your meetings live on your Mac. Sync across devices (sessions, Atelier pieces, brand kits, meetings, projects) is in the Lab, open to a small group of accounts: it asks for your consent, given per device, and a cloud in the icon bar shows its status. Even when it's on, three things never leave the machine: your secrets (keys, tokens), anything that grants power (hooks, MCP servers, access level, allowed commands) and anything specific to this Mac. Archives stay on the Mac too.

Incognito sessions. A session can be incognito: the icon at the top of the home screen starts it that way. Nothing is saved, it disappears when you leave it, along with its files, and it isn't synced. The text is still sent to the model, whose gateway may keep it for 30 days, and usage is counted. Useful for a one-off or personal topic. Image mode isn't available there.

> **WATCH OUT**
>
> Archiving or deleting a session doesn't delete the memories it created, or the files Jay wrote in your folders. If you shared something Jay shouldn't keep, ask it to forget the corresponding memory (chapter 4).

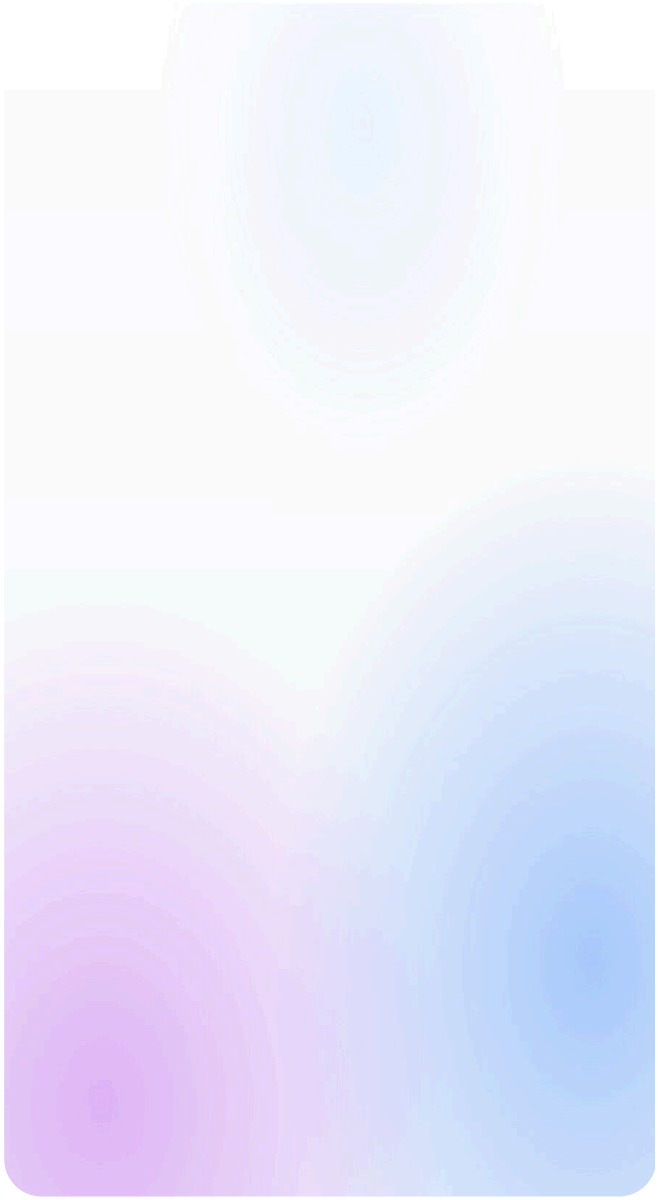
Your settings checklist

- [ ] Job profile filled in and reviewed
- [ ] Both default access levels on Ask while you learn (Work's is set to Full out of the box)
- [ ] Terminal commands to refuse listed (deleting, sending, publishing)
- [ ] Warning before the usage limit turned on
- [ ] Two connectors plugged in (chapter 10)
- [ ] Browser rules checked (chapter 9)
- [ ] Touch ID turned on if your Mac is shared

CHAPTER 04

Projects: one folder, one JAY.md

Without a project, every session starts from scratch. With a project, Jay already knows the folder, the client, the tone, the history and the reference documents. In Jay, a project isn't an abstract space: it's a folder on your Mac, and everything Jay knows and remembers lives there, next to your files.



IN THIS CHAPTER

01 What a project is

02 The three components

03 Linking other folders

04 When to create a project

05 Three JAY.md templates to copy

06 Best practices

What a project is

A project, in the sessions rail, groups sessions and meetings together. It can be attached to a folder, or not.

	PROJECT WITH A FOLDER	PROJECT WITHOUT A FOLDER
What it's for	Working: Jay reads the folder, puts what it produces there, remembers things there	Grouping sessions by topic
What Jay reads	The folder tree (two levels), the JAY.md, and the files it opens itself	The JAY.md and the project memory
Where Jay writes	In the Jay/ subfolder on its own; in your other files depending on the access level	Nowhere on disk
Terminal, Git	Yes	No

A project without a folder carries the "No folder" label. To attach it later, open its settings: "Put this project in a folder: its memory, documents and meetings move there."

The three components

JAY.md, the standing brief. A text file at the root of the folder, read at the start of every session in the project. It tells Jay how to work here: the client's context, the writing rules, where to put what, what it must never do. It is read in full up to 4,000 characters, about one well-filled page: beyond that, Jay cuts it off. Keep it short, and put long documents in the folder.

The Jay/ folder, Jay's footprint. It's the only subfolder Jay writes to on its own. It holds its knowledge base (Jay/knowledge: the documents, presentations, spreadsheets and meeting transcripts it produced or that you added), its memory (memory.md), the project's Design System (DESIGN.md), the brand kits, the deliverables. You can see it in Finder, open it, read it, correct it. Jay reads the rest of the folder, and only changes it according to the access level: in Ask, every write goes through a card that you accept or reject.

Memory, what was said there. Each project has its own memory, separate from the others, so your clients don't get mixed up. It fills up in two ways: when you say "remember that...", and, if you turn it on, through an automatic summary of each session when you leave it. It's off by default; the first "remember" turns it on and says so. It stays local: Jay keeps it out of Git.

WHAT YOU WANT	WHAT YOU DO
Have it remember something	"Remember that this client's reports are always in English."
Check what Jay knows	"What do you know about this project? Sort it into true, false, outdated."
Correct or forget	Ask for it, or edit Jay/memory.md directly

WHAT YOU WANT	WHAT YOU DO
Find a past session	"Pick up what we decided last week about the budget." Jay searches your sessions and your meetings
Remember outside a project	In a session without a folder, "remember" writes to your global memory, read everywhere

WATCH OUT

If a session read the web or a plugged-in tool (an MCP server, chapter 10), a page may have slipped in an instruction. The "Don't memorize sessions that read the web or an MCP" setting protects memory against that. Leave it on.

Linking other folders

A project has one folder, but Jay can read others. Drag a folder onto the session, or use the "+" in the message area, "Link a folder for reading". The linked folder is read-only by design: no write function and no command can reach it. It's the right way to give Jay your archive folder, a reference template or a colleague's folder, with no risk. A session can also link another of your Jay projects, for reading (chapter 15).

When to create a project

My rule: as soon as a topic comes up at least once a week, or as soon as a folder on your Mac deserves to be known by Jay. A key client, a recurring report, your industry monitoring, your LinkedIn posts, your accounting, a code project.

Three JAY.md templates to copy

Create a JAY.md file at the root of the folder, paste, adapt the brackets. Or ask Jay to write it: "create this project's JAY.md from what you see in the folder, and ask me the questions that are missing".

JAY.MD **Client \[name\]**

```
You are my partner on the [client] account, a [industry] company that [activity].
Contacts: [names and roles]. Current stake: [stake]. What they value: [criteria]. What
annoys them: [pain points].
The folder: Contracts/ (the signed proposal), Minutes/ (one dated file per meeting),
Deliverables/ (what we send them).
When I ask you for an email: [tone] tone, formal address, 150 words maximum, as a draft in
my mailbox, never sent.
When I ask you for an analysis: start with the conclusion, then the arguments, and cite
the source file.
Put everything you produce in Deliverables/[YYYY-MM]/. Never change a file in Contracts/.
If a piece of information is missing, look in Minutes/ before asking me.
```

JAY.MD **Monitoring \[industry\]**

```
You help me follow [industry] in order to [goal: anticipate, advise my clients, write
content].
Players to follow: [list]. Priority topics: [list]. Trusted sources: [list]. Sources to
ignore: [list].
For each monitoring brief: search the web, cite every source with its date, separate facts
from interpretations, and end with "What this changes for me" in 3 points.
Save each brief in Monitoring/YYYY-MM-DD.md. Before writing, reread the previous one so
you don't repeat yourself.
Ignore purely promotional press releases, unless they announce a concrete change.
```

JAY.MD **My voice on LinkedIn**

```
You help me write my LinkedIn posts. My audience: [audience]. My promise: [what I bring].
My topics: [3 to 5 themes]. My best posts are in References/: borrow their tone, not their
ideas.
Rules: one idea per post, a one-line hook, short sentences, no jargon, no emoji bullet
lists, no hashtags. End with an open question.
Before writing, suggest 3 angles and wait for my choice. Save each approved post in
Posts/YYYY-MM-DD-title.md.
Never publish yourself: I publish.
```

Best practices

1. **One project per recurring topic.** Avoid the catch-all project: the memory and the knowledge base would end up mixed together.
2. **A clean folder.** Five up-to-date documents beat fifty, half of them outdated. Put the date in file names, and say in the JAY.md which ones are authoritative.
3. **A project fact sheet.** A Jay/knowledge/fact-sheet.md file that sums up the essentials: goals, contacts, decisions made, added to the knowledge base with "Sync the folder". Update it once a month.
4. **Improve the JAY.md after every disappointment.** If you correct the same thing twice, a rule is missing.
5. **Archive finished projects.** The rail stays readable, ⌘K finds everything.

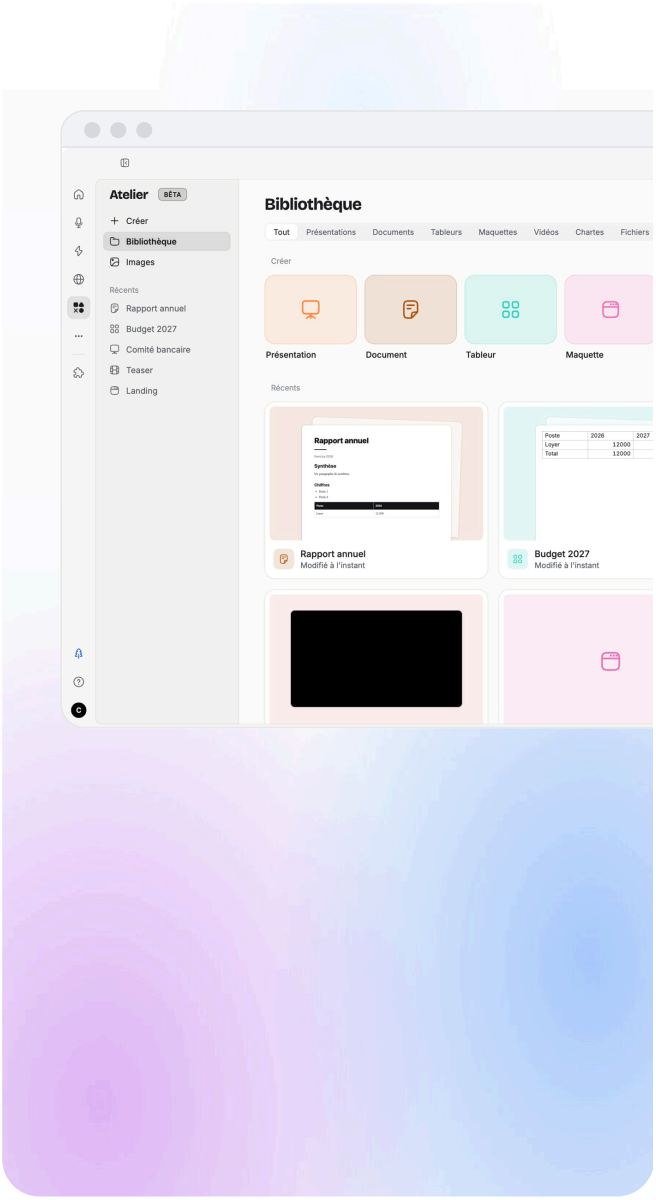
TIP

End every important session with "Summarize the new decisions and information from this session in 5 points, add them to the project fact sheet, and remember what applies next time". The project gets richer with every use, without you having to copy anything.

II · CREATE

CHAPTER 05

The Atelier: documents, spreadsheets, presentations



Jay doesn't hand back text to copy and paste: it writes real files. A Word or PDF document, an Excel workbook, a presentation, a web mockup. These files are called pieces, and they live in the Atelier, in two places at once: in the Atelier tab of the drawer of the session that produced them, and in the Atelier place in the icon bar, which gathers them all.

IN THIS CHAPTER

01 The Atelier place

02 What Jay produces

03 Docs: documents worth rereading

04 Slides: presentations in minutes

05 Spreadsheets: calculations done with code

06 And your Google files

The Atelier place

It opens on the Library: Recent, All, Presentations, Documents, Sheets, Mockups, Videos, Files (exported PDFs, Word, Excel and PowerPoint files), and two sections that now live there: Images and Brand kits (chapter 6). A single "Create" button opens a menu: Presentation, Document, Spreadsheet, Mockup, Video (or Video from a template), Image, then Brand kit and "Import files". Opening a piece takes you back to its session, with the drawer wide open: you keep changing it by talking to Jay.

In a Work session, you don't even have to go through the place: the "+" in the bar, under Do, offers Document, Spreadsheet, Presentation, Mockup, Video, Image and Brand kit. The chosen kind shows as a chip in front of the field, and your next message produces the piece. While it writes, a Jay cursor moves across the piece, with a sentence that says what it's doing.

Atelier pieces are written with Versatile, Thinking or Expert: in Adaptive, Jay picks one of the three.

On disk, a project's pieces are in its Jay/ folder. So they are where you expect them, openable in Word, Excel or Keynote, and attached to an email with a single drag.

What Jay produces

PIECE	WHAT YOU GET	WHAT YOU ASK FOR
Document	A laid-out Word file or PDF. Jay can also revise an existing Word file: it writes a "-revised" copy next to it, never over the original. It fills in PDF forms.	"Turn this into a document I can share."
Spreadsheet	An Excel workbook with formulas, formatting and charts, calculated with code	"Put these figures in a spreadsheet, with a total per month and a chart."
Presentation	Slides made of real objects, in your brand kit's theme, that you present from Jay or export to PowerPoint, PDF, PNG, web page or Google Slides	"Turn this report into 8 slides for the committee."
Mockup	A web page or several screens, laid out on a canvas, to show an idea before building it; all the way up to animated 3D scenes	"Mock up the landing page for the offer."
Video	A short animated video in MP4 (chapter 6)	"A 20-second teaser for this launch."

TIP

Pieces don't ask for permission: Jay writes a new file in its Jay/ folder and touches nothing else. That's why they get made without interrupting you, even with Ask access.

The mockup canvas. A mockup opens wide in the drawer, on an infinite canvas: pages, screens you move around, and a toolbar (Select, Hand, Text, Frame to create a screen, Sticky note, Pen, Shapes). Select an element on the page: the inspector, on the right, shows its properties and code, and whatever you change there is written straight into the file, in the right place, with ⌘Z. Play runs the mockup on the canvas or full screen. The settings a mockup declares (color, density, variant) can be changed in the Tweaks tab, without asking Jay again.

Docs: documents worth rereading

Ask for a document rather than an answer in the session whenever the result needs to be reread, shared or reused: a memo, a procedure, a sales proposal, meeting minutes.

PROMPT **A document ready to share**

```
Create a document: [type of document] for [audience], which must [decision or use].
Structure: [desired sections], or suggest the most suitable one. Tone: [tone]. Length:
[number of pages].
Draw on [files in the folder]. Start with a sentence that gives the conclusion. Add a
table when you compare options. Apply the [name] brand kit.
```

Slides: presentations in minutes

1. **Give the message before the slides.** "By the end, the committee must approve the budget." Every slide then serves to defend that message.
2. **Provide the material.** Your notes, a document from the folder, figures. Jay structures and formats.
3. **Approve the outline before the design.** First ask for the list of slides, with a message title for each.
4. **Improve slide by slide, then present or export.** One request at a time makes a stronger deck than a list of ten changes.

Jay always builds a deck in three stages, shown in the Atelier: content first (the material, the messages, the outline), then the layout of each slide, chosen based on its content, and finally the design. Without a brand kit, it picks a look based on the topic, from fourteen (Swiss, Luxe, Brutal, Retro, Playful, Clinical…); you change it in the Style menu, which separates Looks and Brand kits. A brand kit named in the request is applied in the third stage, on an outline that's already right. A slide image can be regenerated or replaced from the inspector.

PROMPT **The presentation that convinces**

```
Prepare a [number]-slide presentation for [audience] on [topic].
Goal: by the end, they must [decision or action]. Material: [your notes, or the file in the folder].
Step 1: suggest the outline, with for each slide a title that is a message sentence, not a theme, and the key elements.
Step 2: once I approve, create the presentation with the [name] brand kit. One idea per slide, little text, a chart whenever there are figures.
```

Spreadsheets: calculations done with code

A language model is bad at mental math. Jay, on the other hand, calculates with code and writes the result into the workbook with its formulas: you can check every cell.

PROMPT **From file to dashboard**

```
Here is an export of [type of data]: [file path], covering [period].
1) Describe the content of the file and flag any quality issues.
2) This workbook should help [audience] answer these questions: [3 to 5 questions].
3) Build it: the 4 key indicators at the top, then one tab per question with its chart.
Write the formulas in the cells and check the totals with code. Under each chart, write in one sentence the takeaway.
```

WATCH OUT

A pretty, wrong chart is worse than no chart at all. Before sharing, compare three figures with the source and ask Jay to explain its calculations: "Review this workbook like a demanding financial controller: totals, compared periods, units, misleading charts. List each problem with its fix."

And your Google files

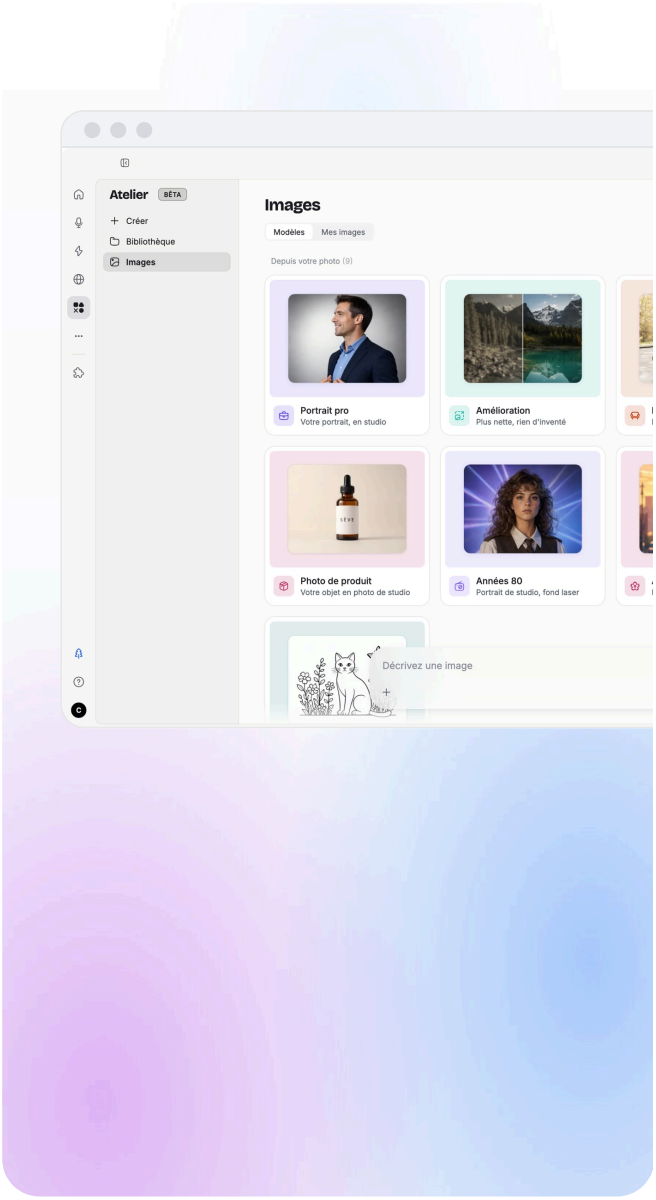
With the Google Drive app (chapter 10), Jay also creates Google Docs, Sheets and Slides in your Drive and writes in them, exports an Atelier piece to Drive, and edits an existing Google file whose link you give it. Searching and organizing across your whole Drive requires full access, still reserved for the Lab. An Atelier piece stays on your Mac; a Google document lives in Drive. Choose based on where your colleagues will read it.

II · CREATE

CHAPTER 06

Images, brand kits and video

Jay generates and edits images, applies your visual identity to everything it produces, and puts together short animated videos. Three tools, one logic: you describe, Jay suggests, you refine by talking.



IN THIS CHAPTER

01 Image mode

02 Brand kits: your identity, applied everywhere

03 A project's Design System

04 Video: short animations in MP4

Image mode

Images are available on every plan. Image mode is something you choose, not something Jay guesses: in the "+" in the bar, under Do, choose Image (or type /image), and your messages produce images. The Image chip shows in front of the field; its cross hands you back to the agent. Outside this mode, an image request goes to the agent, which generates it like any other tool; it's slower but it can file it, name it, insert it into a document.

Once an image has been produced, Jay judges each message: an edit or a new image keeps it in Image mode; any other request hands control back to the agent. All your images end up in the Atelier, in the Images section, under "My images". The Templates tab there offers ready-made transformations of your photo: professional portrait, enhancement without inventing anything, interior design, background change, product photo, and a few more playful styles. "Create, Image" opens a session in Image mode, with an empty bar.

SETTING	OPTIONS
Image model	Auto; Fast, ideal for iterating; Thinking, sharp text in the image, slower
Format	Auto (photo or 4:3), Square, Landscape, Portrait, Story, Widescreen

Image mode isn't available in an incognito session.

PROMPT **A visual for social media**

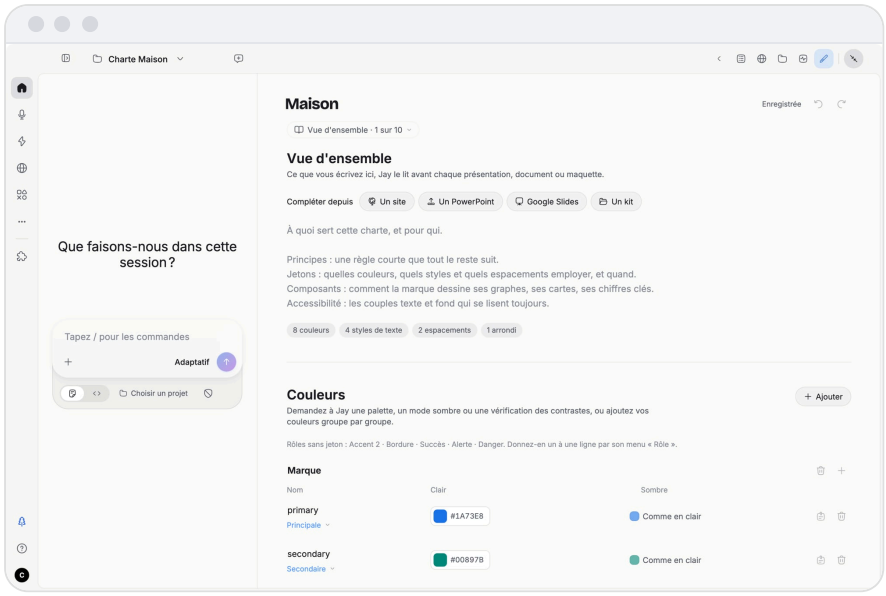
```
Create a visual in [square / story / widescreen] format for [platform].
Main message: [under 10 words]. Audience: [audience]. Mood: [understated, warm,
technical]. Colors: those of the [name] brand kit.
Suggest 3 different directions, then develop the one I choose.
```

TIP

For an image that carries text, a title, a figure, a slogan, switch the image model to Thinking: it's the only one of the two that spells correctly. To explore a mood, stay on Fast and fire off attempts one after another.

Brand kits: your identity, applied everywhere

A brand kit, in the Atelier, is much more than a palette. It's a page in sections: Overview, Colors, Typography, Spacing, Corner radius, Shadows, Components, Brand (your voice, your words, the ones to avoid), Presentations and documents (backgrounds, logo, footer, templates), Assets (logos, fonts, images). Every value is a named token, sorted into groups, with a usage note that Jay reads; every color has one value for light and one for dark. Once chosen, the brand kit dresses your documents, presentations, mockups, visuals and videos, and charts take on the same colors everywhere.



A brand kit open in its session: the conversation on the left, the brand page in the drawer.

A brand kit has its own session. Opening a brand kit means opening its conversation: the page shows wide in the drawer, on its Brand kit panel, and you evolve it by talking to Jay ("add an alert orange", "make me the key figures card"). That's how the "Components and templates" list fills up, each item with its preview. A piece created in that session carries the brand kit automatically, and the brand kit lists every piece that carries it. With Ask access, Jay shows you each change to the brand kit before writing it.

A brand kit exports as a kit (a folder or a .zip) that you send to a colleague; they import it into their Atelier.

TO CREATE A BRAND KIT	WHAT YOU DO
From a website	"Create, Brand kit", then "From a website": colors, fonts, styles, logo and images picked up from the page. Or, in any session: "make the brand kit for [address]"
From your presentations	"From a PowerPoint" or "From Google Slides": the theme, the styles in use, the backgrounds and the logo
From a kit	"Import a kit": a brand kit exported by Jay
From scratch	A blank brand kit opens in its session: describe your brand, Jay fills it in
From a project's code	The Design System (below) exports as a brand kit

These sources also work on an existing brand kit: Jay shows you what they would change before applying it.

PROMPT **A brand kit from my documents**

```
Create a "[name]" brand kit from [presentation file] and [logo].
Pick up: the backgrounds, the logo placement, the footer, the fonts, the colors by role,
and the 5 slide templates I use most.
Add the writing rules: [formal or informal address, title length, what we never write].
Show me the result on a 4-slide deck before saving it.
```

A project's Design System

For a code project, the drawer has a Design System tab. There, Jay reads the project's code and extracts the real system: the colors, sizes, spacing and components as they are actually used, and the drift (the same color written six different ways). It suggests fixes with their evidence, writes the tokens and a Jay/DESIGN.md file whose "House rules" section belongs to you. It isn't a brand kit: it's the truth of the code, which can then be exported as a brand kit or to Figma (chapter 15).

Video: short animations in MP4

Jay puts together animated videos: a teaser, a title sequence, a number counting up, a chart building itself, a map lighting up. The engine is built into the app; nothing goes to a server. The result is an Atelier piece, an MP4 in 16:9, 9:16 for social media, or 1:1, in your brand kit's fonts.

Before writing the video, Jay shows you a brief card: the message, the length, the format, the scenes. Approve it, that's where everything is decided. Ready-made scenes speed it up: a number going up, a map of France, a computer, a terminal, an AI answer being typed out, subtitles. The finished video opens in the Atelier player.

PROMPT **A 20-second teaser**

```
Make a 20-second video, 9:16 format, to announce [event or launch] on [platform].
Message: [one sentence]. Key figure: [figure]. Brand kit: [name].
Scenes: a hook, the figure counting up, three benefits appearing one by one, the logo and
the date. No music.
Show me the brief before rendering the video.
```

WATCH OUT

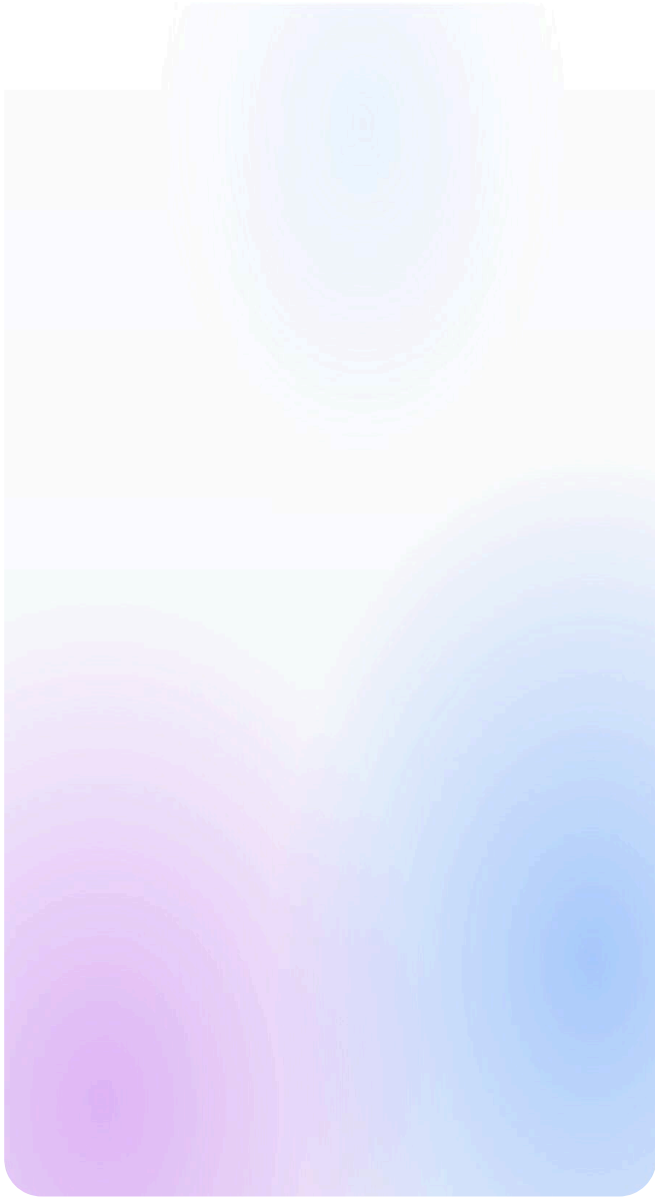
Model-generated video, the kind that invents filmed shots, isn't in Jay yet. What Jay does today is motion design: text, shapes, your images and your figures, animated.

II · CREATE

CHAPTER 07

Ideation

Jay is a tireless thinking partner: it comes up with a hundred ideas without getting bored, plays every role and challenges your certainties without offending anyone. Yet its first ideas are often the most obvious ones, just like ours. The method comes down to two stages: lots of ideas first, sorting second.



IN THIS CHAPTER

01 Diverge, then converge

02 Seven creativity techniques

03 Sorting ideas without getting it wrong

> In Jay, ideation has an advantage: it happens in the project folder. Your strategy, your target clients and the list of ideas already tested are there, and Jay stops suggesting what you've already tried. Switch to Thinking mode for the sessions that matter.

Diverge, then converge

1. **Diverge.** Ask for lots of ideas, very different from each other, without judging them. Thirty rather than five.
2. **Push further.** "Drop the 10 most obvious ones and give me 10 bolder ones."
3. **Converge.** Sort with explicit criteria, then dig into the three best.
4. **Test.** Challenge the ideas you keep before investing time in them.

PROMPT **The full session**

```
I'm looking for ideas for [problem or goal]. Constraints: [budget, deadline, audience].
The ideas already tried are in [file in the folder]: don't suggest them again.
1) Give 30 very different ideas, one line each, without judging them.
2) Remove the 10 most obvious ones and replace them with 10 bolder ideas.
3) Rank everything by impact and ease, in a table.
4) Develop the 3 best: concrete first step, cost, main risk.
Save the result in an Atelier document.
```

Seven creativity techniques

TECHNIQUE	WHAT YOU ASK	WHY IT WORKS
Inversion	"How can I make sure this launch is a disaster?", then reverse each answer	Pitfalls are easier to see when you look for them
SCAMPER	Three ideas per verb: substitute, combine, adapt, modify, put to another use, eliminate, rearrange	Transforms what exists instead of starting from scratch
Constraint	"What if you had no budget? One week? No advertising?"	Constraints force originality
Analogy	"How would a theme park, an airline and a video game solve this problem?"	Borrows from industries that have already solved the same problem
Pre-mortem	"Six months from now, the project has failed. Why?"	Strengthens an idea before launching it
Six hats	Facts, emotions, risks, benefits, creativity, organization: an analysis from each angle	Changes perspective without changing who you talk to
Virtual panel	Jay plays five very different customers reacting to your offer	A quick first filter, before real interviews

PROMPT **The virtual panel**

```
Create 5 realistic and very different personas of [your target]: age, job, constraints,
what motivates them, what drives them away.
Present this offer to them: [your offer, or the document path].
Have each one react in 3 sentences, in their own words. Then summarize: what appeals, what
holds them back, and the change that would win over the most profiles.
```

WATCH OUT

A simulated panel reflects general trends. Your real customers may react completely differently. Use it to prepare your real interviews and weed out bad ideas, never to validate an important decision.

Sorting ideas without getting it wrong

PROMPT **The sorting matrix**

```
Here are my ideas: [list, or the document path].
Score each one out of 5 on: impact on [goal], effort, cost, time to first result, risk.
Present a table sorted by score, then place the ideas in 4 groups: launch now, prepare,
keep for later, drop. Explain each drop in one sentence.
```

TIP

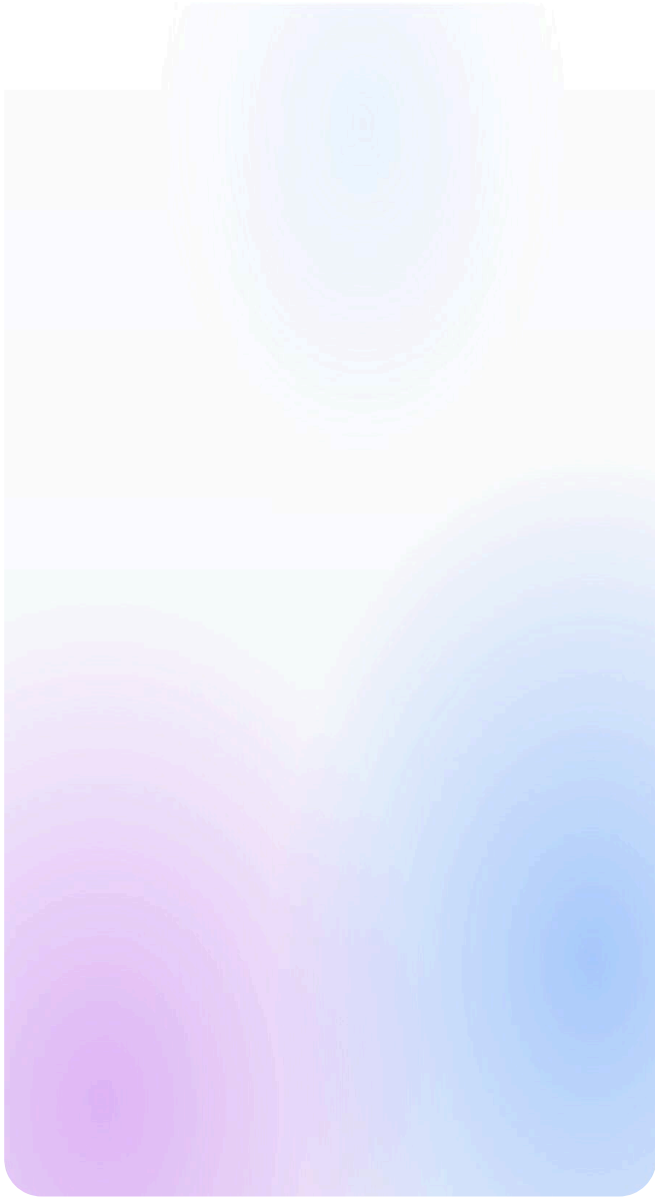
Keep an "ideas-tested.md" file in the project's Jay/ folder, and end each session with "add the ideas we kept and dropped to ideas-tested.md". The next session picks up where you left off.

III · ANALYZE

CHAPTER 08

Reading your documents and data

Contracts, reports, studies, spreadsheets, screenshots, and above all entire folders: Jay reads in seconds what would take you hours. On your Mac, there is nothing to upload. You drag a file into the session, or give its path, or simply say “the invoices in the Accounting folder.” The most important part is asking the right questions, then checking the answers.



IN THIS CHAPTER

01 What Jay can read

02 The method: extract before concluding

03 Prompts by document type

04 An entire folder

05 Privacy

What Jay can read

DOCUMENT	WHAT JAY DOES WELL	WHAT TO WATCH
Contracts	Summarize obligations, spot unusual clauses, compare against a template in the folder	Not a substitute for legal advice on a major commitment
Reports and studies	Pull out key figures, summarize for a given audience, list open questions	Check the figures cited against the source
Spreadsheets	Clean, calculate, cross-reference, visualize, with code	Ask for the formulas, not just the result
Images and screenshots	Read a chart, a photographed table, an interface	Figures read from an image must be checked
Meeting transcripts	Minutes, decisions, action items, quotes	Spoken names and numbers are often mistranscribed: the Lexicon fixes them (chapter 12)
Entire folders	Inventory, sort, fill one spreadsheet row per file	Have doubtful cases flagged rather than guessed

The method: extract before concluding

The classic trap: asking for a conclusion on a long document right away. Jay may then generalize. The right method first pulls out the relevant passages, then reasons on them.

1. **The document first, the question second.** Name the file, or paste it at the start of your message, and ask your question at the end.
2. **Ask for exact quotes.** “Extract word for word the passages that discuss [topic], with their page.”
3. **Then the analysis.** “Based only on these excerpts, answer [question].”
4. **Require references.** Every claim must point to a passage you can open.

PROMPT **Analysis grounded in the text**

```
Open [document path].
Step 1: extract word for word the passages relevant to my question, with their page or
section.
Step 2: based only on these passages, answer this question: [question].
If the document doesn't allow you to answer, say so clearly instead of filling in with
your own knowledge.
```

TIP

For a long document or a decision that matters, switch to Thinking or Expert mode: they reason before answering, and Thinking has the largest window, one million tokens.

Prompts by document type

PROMPT **The contract**

```
Analyze [contract path] from the point of view of [my position: client, supplier, employee].
Give: 1) each party's obligations, 2) the term, renewal and exit conditions, 3) amounts and penalties, 4) unusual or one-sided clauses, ranked by risk level, 5) the questions to ask before signing.
Quote the exact clause for each point. Compare against the template [file] if you find it in the folder.
```

PROMPT **The report for a decision-maker**

```
[report path] is [number] pages long. My director has 3 minutes.
Write: the conclusion in 2 sentences, the 5 key figures with their page, what is solid and what is weak in the methodology, and the recommendation I draw from it for [our situation]. Save it as a document in the Atelier.
```

PROMPT **The screenshot or chart**

```
Here is a screenshot of [table, chart, interface]. Transcribe the data into a clean table. Flag any value you can't read clearly or that seems inconsistent. Then tell me in 3 points what this data shows.
```

PROMPT **Comparing offers or versions**

```
Here are [number] documents in [folder]: [offer A, offer B, offer C] or [version 1 and version 2].
Build a comparison table on these criteria: [criteria]. For each cell, cite the file and the page.
Then list the important differences a reader in a hurry would miss, and give your recommendation for [my goal].
```

An entire folder

This is where Jay on your Mac changes everything. Twenty PDFs, forty invoices, a hundred résumés: Jay opens them one by one and fills one row per file.

PROMPT **The document roundup**

```
Read all the PDFs in the folder [path]. For each one, fill a spreadsheet row: file name,
date, author or issuer, [3 columns you care about], and a “to check” column where you flag
anything you're not sure you read correctly.
Save the workbook in [folder]. Don't modify or move any file.
```

WATCH OUT

A session has a context size. As it fills up, Jay summarizes older exchanges and keeps only the summary: the context gauge appears in the bar once the session passes the halfway mark, the summary shows its progress, and you can trigger it yourself with /compact. For a large folder, ask for one table per file instead, or hand it to a background task (chapter 13): each file is processed in a fresh context.

Privacy

Your files stay on your Mac. What goes to the model is the text needed for the answer, sent to models hosted in France that don't use it for training. Before having a document read, still ask yourself whether it contains personal data or trade secrets, and check your company's policy. For a third-party document that Jay must never modify, link its folder as read-only (chapter 4).

TIP

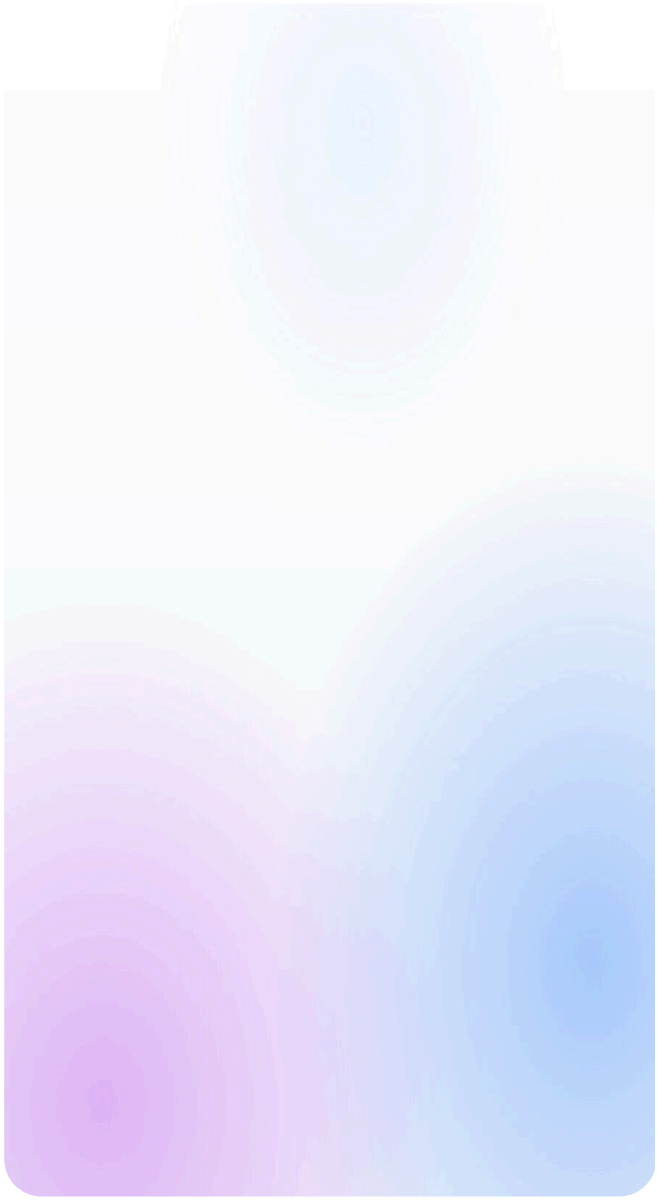
Documents you consult often (collective agreement, catalog, procedures) belong in the project folder, or in Jay/knowledge: Jay finds them on its own, and “Sync the folder” updates its knowledge base when you add more.

III · ANALYZE

CHAPTER 09

The Browser and the web

Jay searches the web in seconds and cites its sources. But it also has a real browser of its own, inside the app: there it opens pages, clicks, fills in forms, extracts tables, downloads files. You watch it work, and you decide page by page what it is allowed to do.



IN THIS CHAPTER

01 Two levels: search or act

02 The Browser place and the Web tab

03 Jay's safeguards on the web

04 Three ways to use the Browser

05 Answers you can use

06 Checking sources

Two levels: search or act

	WEB SEARCH	THE BROWSER
Duration	A few seconds	As long as the task takes
Method	One or more searches, reading the pages	Jay drives pages the way you would
Result	An answer with citations	An action done, a file downloaded, a table extracted
For	A fact, a price, news, the state of the art	A site without a connector, a form, a comparison across several sites, a customer account

For in-depth research, monitoring, or a market study, ask for it explicitly in Thinking mode, with the brief below: Jay chains searches, opens the sources and writes a sourced report. A long research job can also go to a background task, so you keep working while it runs (chapter 13).

PROMPT **The research brief**

```
Research question: [your question, specific]
Why I'm asking: [the decision I need to make]
Scope: [country, period, industry, company size]
Preferred sources: [official sources, studies, trade press]. Sources to avoid:
[promotional content, forums]
Open the pages, don't settle for search snippets. Format: a 5-point summary, then detail
by sub-question, then a table of sources with their date, in an Atelier document.
Clearly separate established facts, estimates and opinions. Flag the points where sources
contradict each other.
```

The Browser place and the Web tab

The Browser, in the icon bar, is a full browser whose pages persist: you switch places, you come back, they're still there. Its tabs are grouped into web sessions (one group per topic), and its successful recipes are saved as WebFlows, which you replay in one click or schedule (chapter 14).

In a session, each page Jay opens becomes a Web tab in the drawer. It's the same thing, next to the conversation: you see the page it's looking at, you can click on it yourself, for example to sign in, then hand control back.

Access applies to the web too. There is no longer a separate setting for pages: the session's access level decides. In Ask and Auto, Jay asks before every click, entry, download or file upload; in Full, it acts without asking, and a click on a page can pay or send. The Web tab and the Browser drawer have their own access selector. Start in Ask.

Per-site rules. In Settings, Browser, “Sites and Jay”: an Allowed site, where Jay acts without asking (except to run code or upload files); a Blocked site, which Jay no longer reads or acts on, subdomains included, even with Full access. Block your bank and your sensitive sites on day one.

Jay's safeguards on the web

They apply everywhere, in the Browser as in the Web tab, whatever the access level, and no one can lift them:

- Jay **never** enters a credit card number.
- Jay **never** enters a password, except on a local page on your own machine (a site you're developing).
- A page that addresses an AI (“ignore your instructions and...”) is neutralized and flagged, and every action asks for your confirmation again until the end of the turn, even with Full access.

WATCH OUT · PROMPT INJECTION

When Jay reads a web page, an email or a document, that content may contain hidden instructions trying to hijack it. The safeguards stop many of them, not all. If Jay suddenly proposes an action unrelated to your request after reading something, refuse it. That's one more reason to keep confirmation on important actions.

Three ways to use the Browser

PROMPT **Comparing offers across several sites**

```
Open [site 1], [site 2] and [site 3]. For [product or service], note the price, terms,
lead times and hidden fees.
Put it in a table with a link to the exact page for each cell, then recommend an option
based on [my criteria]. Don't buy anything, don't create any account.
```

PROMPT **Filling in a repetitive form**

```
On [site], fill in the [name] form using [file in the folder]. Show me each filled page
before moving to the next, and stop before the submit button: I'm the one who submits.
```

PROMPT **A recipe to replay**

```
On [your customer account site], go to [section] and download [the monthly statement] as a
PDF into [folder].
Once it works, save the recipe as the WebFlow “[name]” so I can rerun it every month.
```

Answers you can use

Jay doesn't just return text. When the answer lends itself to it, it arrives in the thread in a form you can work with.

WHAT YOU ASK FOR	WHAT JAY RETURNS
Ideas for places, products, destinations	A photo gallery, one per subject presented
Sources, tools, articles	Link cards with thumbnails that keep their URLs
A recipe, or the URL of a recipe page	A Recipe card: photo, time, adjustable servings (quantities follow), ingredients and steps, and “Add to shopping list”
A shopping, packing or to-do list	A checklist, with its sections and its “9/30” counter. Checked boxes stay in the session, and Jay knows what's done when you write to it next

PROMPT **The list you check off**

```
Make me a list of everything needed for [event, trip, move], in sections, as a checklist.
Take into account [constraints: number of people, duration, season]. I'll check things off
as I go: when I come back, tell me what's left.
```

Checking sources

Citations don't guarantee an answer is right. A source can be outdated, misunderstood, or unreliable itself.

- [ ] Open the 3 sources the main conclusion rests on
- [ ] Check the date of each important figure
- [ ] Check that the quote actually says what Jay makes it say
- [ ] Prefer primary sources: official report, study, the company's own page
- [ ] Ask for the points where sources disagree

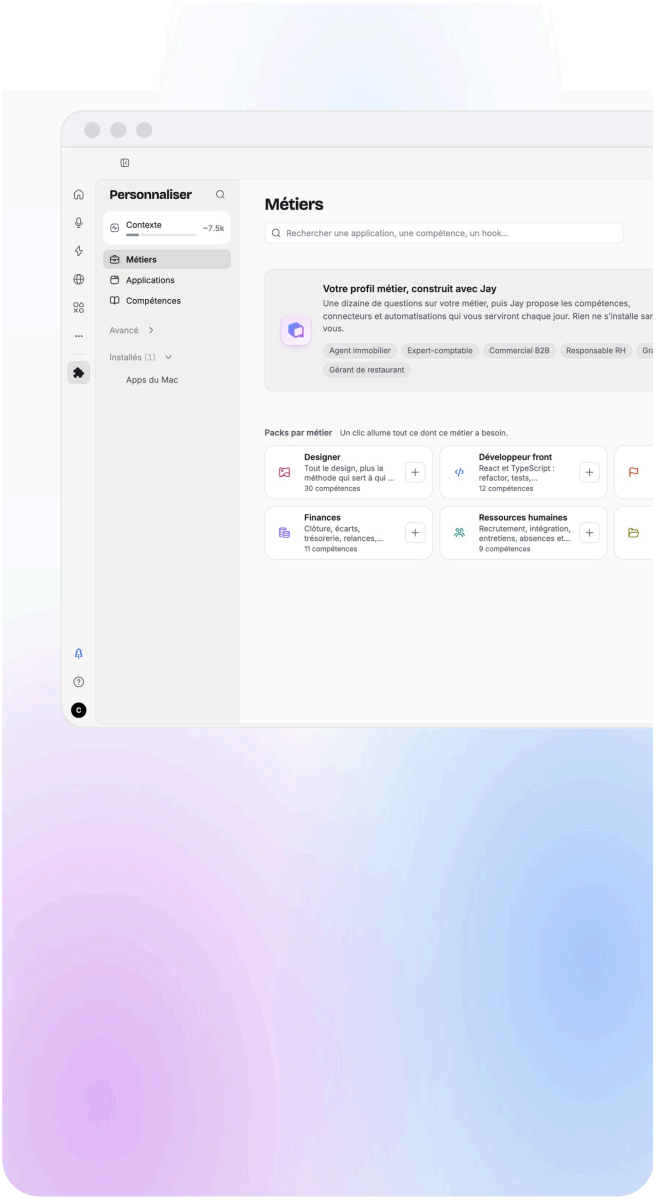
PROMPT **The cross-check**

```
For the 5 most important claims in your report, give: the exact source, its date, whether
it's a primary or secondary source, and your confidence level (high, medium, low) with a
one-sentence explanation.
```

CHAPTER 10

Applications and connectors

Without connectors, you spend your time copying and pasting between Jay and your tools. With them, Jay reads your email, your calendar, your Notion pages, your accounting, and acts directly in them, with your approval. In Jay, connectors have one more advantage: they combine with your files and your Mac apps.



IN THIS CHAPTER

01 The Customize place

02 Three ways to connect a tool

03 What each connector can do

04 MCP, explained simply

05 Winning combinations

06 The golden rules

The Customize place

Extensions and Skills have merged into a single place, Customize, at the bottom of the icon bar. It speaks in two families: **Applications**, what Jay can connect to, and **Skills**, what it knows how to do (chapter 11). It opens on **Roles**: packs by role that one click turns on, and “From your sessions”, what you often ask Jay for that isn't installed yet.

Each card has a “+” to add it. Clicking the card opens its details: what it does, its weight in the context, the connection (the account, the key) in the same place, and examples that launch a session with the request directly. A single button, “Add”, covers everything: connect an application, create or import a skill, and, under “Advanced”, an MCP server, a hook or a sub-agent. ⌘K, opened from Customize, searches only within that place.

Three ways to connect a tool

	JAY'S CONNECTORS	MAC APPS	MCP SERVERS
What it is	Integrations written by dev-id for the most common tools	The apps already installed on your Mac, which Jay drives	The universal plug: any tool that offers an MCP server connects to Jay
Examples	Gmail, Google Drive, Google Calendar, Notion, Qonto, Pennylane, Pappers, HubSpot, Slack, Asana, Axonaut, GitHub, Payfit, SNCF, Open data France…	Notes, Reminders, Contacts, Calendar, Mail, Messages, Music, Photos	Linear, Stripe, Supabase, Pipedrive, data.gouv.fr, Sketch, Blender, and your company's internal tools
How it connects	An OAuth connection, a token or an API key, in its details	Nothing: the app is there. macOS asks for permission the first time	From the catalog, or “Add, Advanced, MCP server”
Where	Customize, Applications	Customize, Applications	Customize, Applications, or Advanced, MCP servers

Google, as three applications. Gmail, Google Drive and Google Calendar connect one at a time, each with its own approval at Google: you give Jay only what you need. Several Google accounts can coexist; for each service, Jay uses the account that has it. “Disconnect” on a card cuts off the service in Jay; to revoke access at Google, disconnect the whole account.

What each connector can do

CONNECTOR	WHAT JAY CAN DO	WHAT IT NEVER DOES
Google Calendar	Read your calendar, find time slots, create and edit events, with a Meet link	

CONNECTOR	WHAT JAY CAN DO	WHAT IT NEVER DOES
Google Drive	Create Docs, Sheets and Slides, write in them, edit a file whose link you provide, export a file there; in the Lab, search, read and organize across the whole Drive	Share without asking you
Gmail	Send an email, always after your confirmation; in the Lab, search, read, sort, organize, prepare drafts	Send without explicit confirmation, even with Full access
Notion	Search, read, query a database, create a page, append, edit, create a database, add rows, comment	Delete
Qonto	Read accounts and transactions, attach a receipt, manage invoices, quotes, credit notes, clients and reminders	A transfer, of any kind
Pennylane	Read, and the full write API: receipts, reconciliations, invoices, quotes, journal entries, matching, FEC exports	
HubSpot, Asana, Axonaut, Slack	Read, then create and update records, tasks, quotes; post in Slack	Delete
GitHub	Search your repositories, read, follow Actions, open issues and pull requests	Merge, approve
Pappers, Open data France	Look up a company, its executives, its BODACC notices, an address; Open data France with no key at all	
Payfit	Read the team and absences	Read payroll: salaries, IBANs, social security numbers are never sent
SNCF	Schedules, routes, disruptions	Book
LinkedIn, Microsoft 365	Coming soon: publish a post or a comment, for LinkedIn; Outlook, OneDrive and Teams, for Microsoft 365	
Mac Mail and Messages	Prepare a draft in Mail; send a message in Messages	Send an email; send a message without your approval

Anything that goes to a person always asks. An email, a Slack message, a Notion comment, an invoice sent through Pennylane or Qonto, an edited event whose guests will be notified: Jay shows you the card and waits for your approval, even with Full access. Creating or editing a record, on the other hand, follows the access level.

Full access to Gmail and Drive (reading, searching, organizing) depends on a security audit required by Google: it remains reserved for the Lab. Calendar, sending email and the Google files you point to are open to everyone.

MCP, explained simply

MCP (Model Context Protocol) is the open standard that lets you connect an AI to any tool. Think of it as a universal plug. Jay connects to it in the cleanest way there is: it registers itself with the server, with no pre-registered ID or shared secret, and you authorize access in your browser. The day a tool opens an MCP server, it works in Jay without an update. When a Jay connector exists for a tool, Jay prefers it over the MCP server that duplicates it.

Winning combinations

COMBINATION	WHAT YOU ASK
Calendar + Mail + Drive	"Prepare my meetings for tomorrow: for each one, the latest exchanges with the participants and the useful documents."
Notion + project folder	"Update the project's Notion page with the decisions from the last three meeting minutes in the folder."
Pennylane + Qonto	"Match the receipts in the Invoices folder with this month's payments, and list what's still missing a receipt."
Pappers + Open data France + Browser	"Before the meeting, put together the profile of [company]: legal identity, executives, BODACC notices, recent news."
Meetings + Calendar + Mail	"From yesterday's meeting, create the deadlines in the calendar and prepare the follow-up email."

PROMPT **Your morning, prepared**

```
Look at my calendar for today and my unread emails from the last 24 hours.
Give me: the 3 things that need action from me, what I need to prepare for each meeting
(latest exchanges, related documents in the Drive or in [folder]), and the emails I can
ignore.
Draft the simple replies, without sending anything.
```

The golden rules

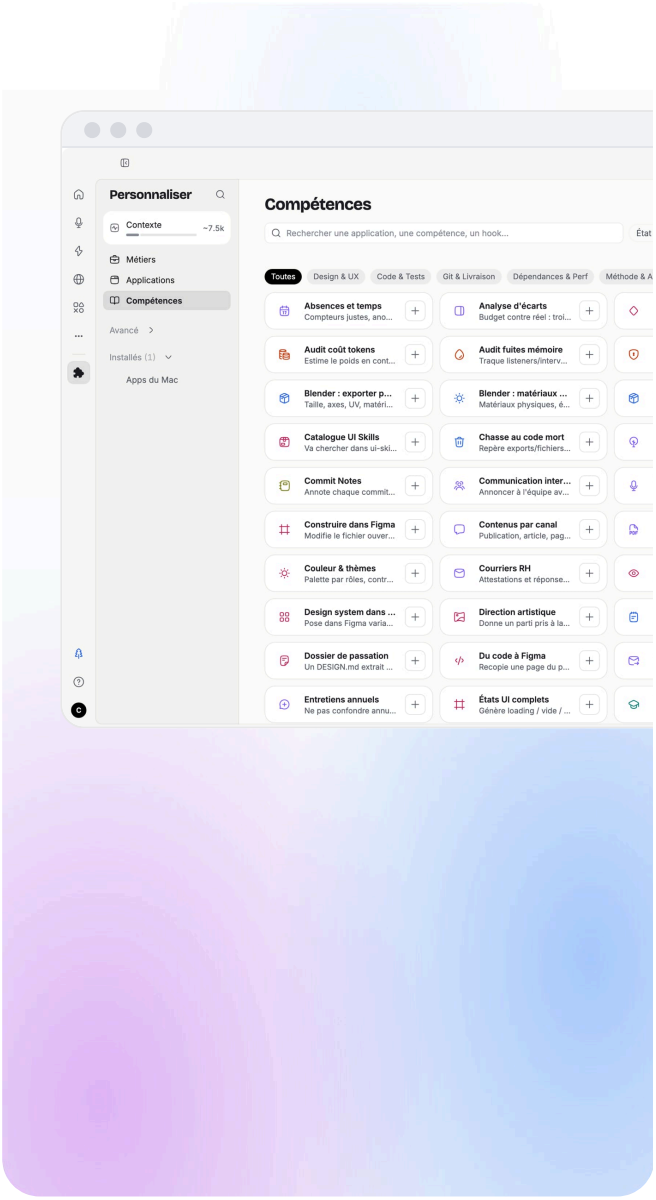
WATCH OUT

Read before you approve. Jay shows you the action it wants to take: an email, an event, an edited page, an invoice. Check it. **Connect only what you need.** Every active connector is one more open door. An API key can be revoked with the vendor; do it when you no longer need it. **Be wary of outside content.** An email or a Notion page can contain hidden instructions meant for the AI. If Jay proposes an unexpected action after reading external content, refuse it. **At work, ask before connecting.** An API key commits the company's account, not just yours.

CHAPTER 11

Skills

A Skill is know-how you teach Jay once and that it applies whenever the situation calls for it: your way of writing meeting minutes, your way of reviewing a contract, your brand guidelines, your hiring process. Jay ships with nearly ninety of them, grouped into packs by job, and you create your own in a single conversation.



IN THIS CHAPTER

01 JAY.md or Skill?

02 Skills in Customize

03 Your job profile

04 Creating your own Skill

05 Hooks: automatic reflexes

06 Commands to know

JAY.md or Skill?

	A PROJECT'S JAY.MD	A SKILL
Scope	A single project	All your sessions
Trigger	Always read in the project	Turned on in Customize, or called by its /name command
Content	Text: the context, the folder's rules	A method: the steps, the format, the criteria, the pitfalls
Ideal for	The context of a client or a topic	A way of working reusable across several contexts

Skills in Customize

Skills are the second family in Customize (chapter 10). A filter shows them all, active or inactive; each card states its weight in the context (light, moderate, heavy), its command, and offers examples that launch a session.

Packs. The Customize home, Roles, groups them by job: Designer, Front-end developer, Marketing, Communication, Finance, People, Administration, Sales. A pack turns on a set of Skills in one move; it can be off, partial or on. The six job packs share a base of four cards: writing, summarizing, spreadsheet, presenting.

From your sessions. Jay spots what you keep asking it for and offers to turn it into a Skill, already pre-filled with your own words.

> TIP
>
> Don't turn everything on. An active Skill weighs on every message, and ten methods at once get in each other's way. Turn on your job's pack, then call the others on demand by their command: /critique, /seo, /landing, /typo... The “+” in the bar lists them too, with their description on hover. Jay applies them on its own when your request matches them.

Your job profile

The job profile (chapter 3) is the gateway to Skills. Type /metier, or click “Start the interview”: Jay asks you questions, one at a time, then shows you a plan card: the Skills, connectors and rules it suggests installing. Nothing installs without your click. Profiles add up.

Creating your own Skill

A Skill has four fields: a name, a command, a description and instructions. Take particular care with the description: it says what the Skill does and when to use it, and it's what Jay reads to decide whether to use it. The form shows its weight in tokens.

The easiest way: have Jay write it. Customize, “Add”, “Create a skill with Jay”: it asks its questions and pre-fills the form. “Write a skill” opens the empty form.

PROMPT **Creating a Skill in conversation**

```
I want to turn this method into a Skill: [describe the task and how you do it]. A good
example of the result is in [file in the folder].
First ask me the questions you need: steps, format, quality criteria, pitfalls.
Then write the Skill: a description that says when to use it in the words I would use,
concrete steps as if for a new colleague, a Verification section with what must never be
wrong. Suggest a short command for it.
```

EXAMPLE **Skill**

```
Name: Meeting minutes
Command: /cr
Description: Writes the minutes of a meeting from a transcribed meeting or from notes. Use
it when the request mentions minutes, a meeting summary, meeting notes or a follow-up
after a call.
Instructions:
1. Identify the participants, the date and the topic.
2. Extract the decisions, then the action items with owner and deadline.
3. List the open issues.
4. Save the minutes in Minutes/YYYY-MM-DD-topic.md and prepare a follow-up email of 150
words maximum, as a draft.
Format: short headings, lists, no introductory phrases.
Verification: flag any ambiguous name, number or date instead of guessing.
```

Sharing and importing. A Skill exports as a .zip from its card, and imports via “Add, Import a skill”, from a file, a GitHub repository or a URL. That's how a team shares its way of working: one file, and everyone reviews contracts the same way.

IF A SKILL DOESN'T TRIGGER

It's common at first. Check that it's active, sharpen its description with the words you actually use, or call it by its command.

Hooks: automatic reflexes

A hook is a rule that fires at a specific moment: when you send a message, when Jay finishes a response, before or after it uses a tool, when the app starts. They live in Customize, Advanced, Hooks.

TYPE	WHAT IT DOES	EXAMPLE
Rule (safeguard)	Ask for confirmation, Refuse, or Remind Jay of something, without running any command	“Before any write in Contracts/, refuse.”

TYPE	WHAT IT DOES	EXAMPLE
Command	A small script run on the event, limited to 5 seconds, that never blocks the session	Format a file after each edit, notify an app

Rules are the right starting point if you don't code: they keep Jay in bounds without a single line of script. Commands are for developers (chapter 15).

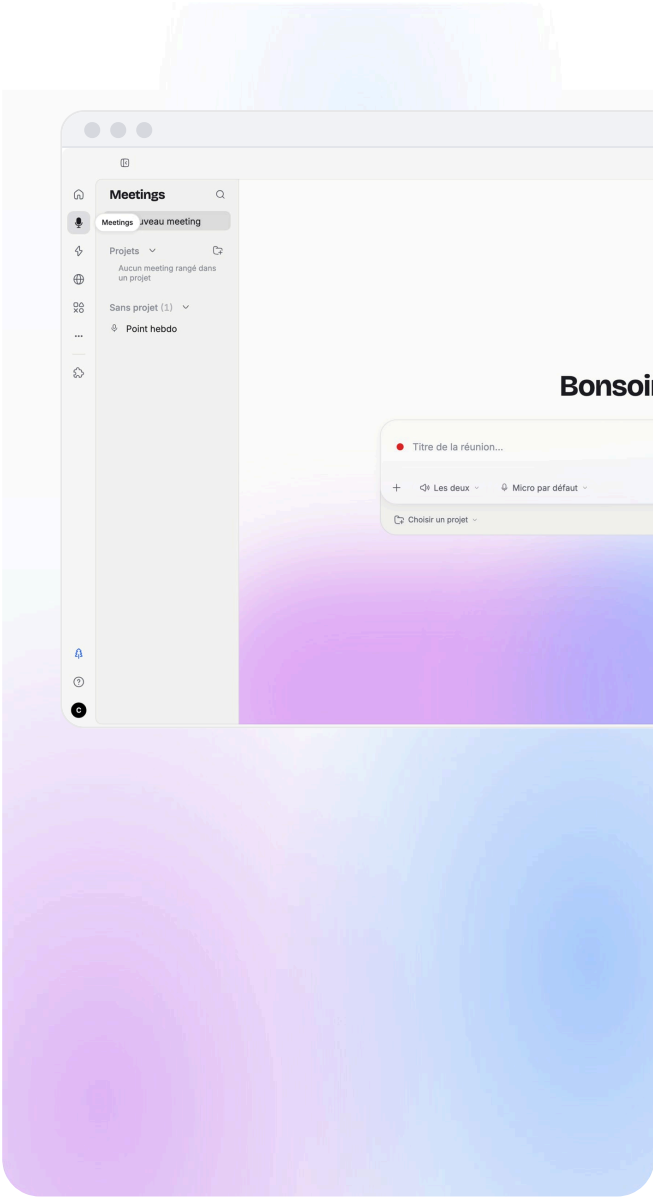
Commands to know

COMMAND	WHAT IT DOES
/plan and /build	Switch to Plan mode (read-only), then execute it
/image	Messages produce images
/compact	Summarize the session to free up context
/goal	Clarify a goal, then produce a structured specification
/metier	Build your job profile
/commit, /diff, /status, /pr	The Git moves (chapter 15)
/model, /effort, /permissions	Change mode, effort, access level without leaving the keyboard
/new, /clear, /help	A new session, clear the conversation, help on commands
/name	Each Skill has its command; a Command hook can have one; in the Browser, “/” opens a WebFlow

CHAPTER 12

Meetings

A meeting is an hour of decisions that everyone forgets the next day. Jay records it, transcribes it with an open-source model hosted in France, draws a summary from it, and keeps it in the project: the following week, “what did we decide with the client about the deadline?” finds its answer in the transcript.



IN THIS CHAPTER

01 Recording

02 Making use of it

03 The Lexicon: names it mishears

Recording

In the sidebar, “New meeting”, or “New meeting in this project” so it joins the client's folder. Jay records your microphone and the Mac's audio: a video call in your browser or your meeting app is captured without installing anything. The glowing red dot in the sidebar means a recording is in progress. At the end: “Stop and transcribe”.

Importing. Have a recording made elsewhere, a voice recorder, a file you received? Drop the audio file, or an existing transcript (text, subtitles, Word, PDF), into the Home bar: Jay turns it into a meeting.

> **WATCH OUT**
>
> Recording a meeting is something you tell participants about. Let them know, and follow your company's rules. Recording happens on your Mac; the audio is then sent to a transcription model hosted in France, as is the text for the summary.

Making use of it

A meeting's drawer has its tabs: the Summary, written on its own when transcription or import finishes, your Notes, and “New conversation” to question the meeting like any other session.

WHAT YOU WANT	WHAT YOU ASK
The minutes	“Decisions, action items with owner and deadline, open issues, follow-up email as a draft.” Or your /cr Skill (chapter 11)
Commitments	“What did I promise, to whom, by when? Create the reminders in the calendar.”
An exact quote	“What exactly did [person] say about the price? Quote them word for word with the timestamp.”
The thread of a topic	In a session: “Search my meetings with [client] for everything about [topic], by date.”
A presentation	“Turn this meeting into 6 slides for those who weren't there.”

The Lexicon: names it mishears

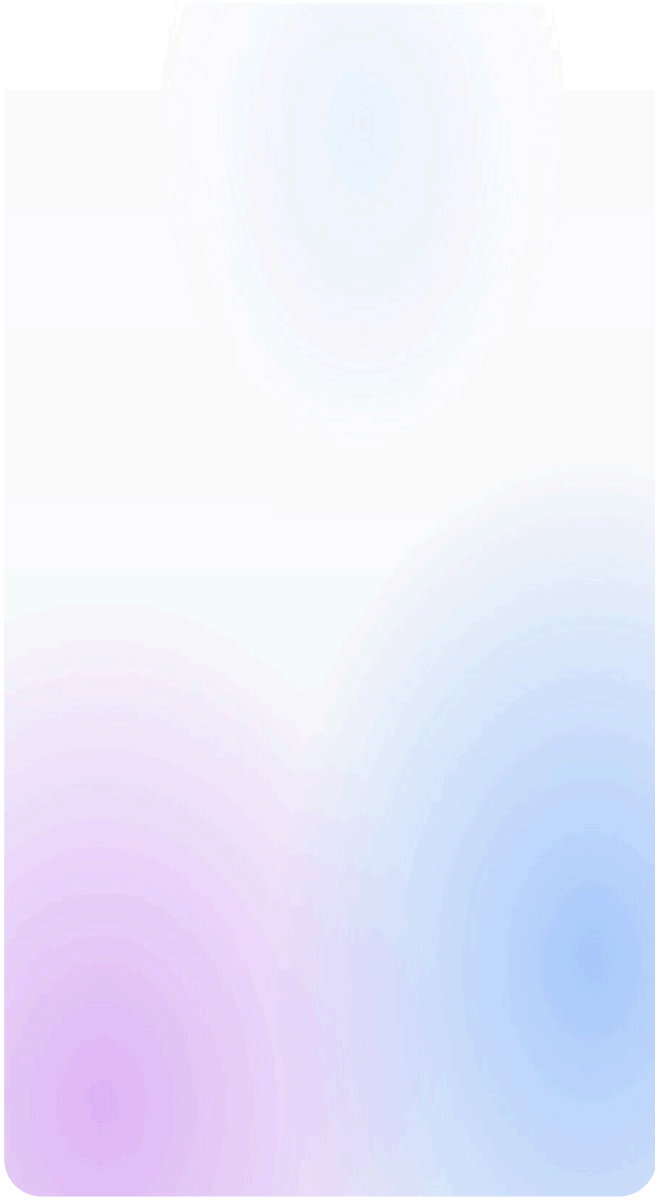
Proper names, acronyms and your industry's terms are the leading source of transcription errors. In Settings, Transcription, the Lexicon lists the words to correct automatically: the names of your clients, your products, your colleagues. Ten minutes, and every transcript after that is accurate.

TIP

A meeting attached to a project joins its knowledge base. Its decisions feed the following sessions without you having to copy them over, and a ⌘K search finds it by its content.

CHAPTER 13

Delegating: permissions and background tasks



When you ask a question, Jay answers and stops there. When you hand it a goal, it picks the steps and tools itself, looks at the result, corrects course, and keeps going until the deliverable is done. That is what makes it an agent. Knowing how it runs lets you choose what you hand over, and how far.

IN THIS CHAPTER

01 The agent loop

02 The three access levels

03 The delegation brief

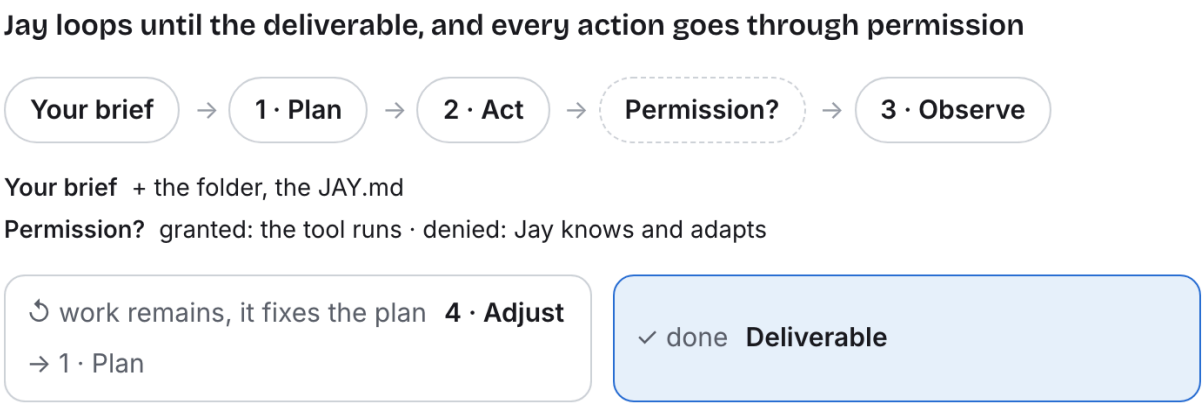
04 Background tasks and sub-agents

05 The terminal

06 When not to delegate

07 Ten tasks to hand to Jay

The agent loop



On each turn, Jay breaks the work into steps and picks a tool, and this is where your access level decides: the tool runs on its own, or a card asks for your approval. Jay then reads the result, adjusts its plan, and starts again until the "done" criteria are met. A task list appears under its reply as it goes, and an end block closes the turn: the summary, its question if it has one, the files produced, the time and the tokens.

The three access levels

You choose the level in the bar, on the access selector right after the project, per session. It doesn't change what Jay can do: it changes what it does without asking you. Settings, the bar and this guide use the same three words.

LEVEL	WHAT JAY DOES	WHEN TO USE IT
Ask	Asks for confirmation before every action: writing, a command, a connector, a click on a page. Reads always go through.	To get started, and for any sensitive task
Auto	Approves everyday actions on its own: commands, calendar, reading and sorting email, drafts, Drive, memory. Shows the diff before any write to a file, and asks before acting on a web page.	Tasks you know well
Full	No confirmation, except for the few always-confirmed actions below. Jay writes, runs its commands and clicks on pages without asking.	Well-honed tasks, in a folder where nothing is irreversible

One access level per position. A session keeps two levels, one for Work and one for Code, each on its own selector. **Work starts in Full**: you write documents and Atelier pieces there, and Jay doesn't interrupt you at every paragraph. **Code starts at the level you chose in Settings** (Ask out of the box). Switching from one to the other applies the level of the position you land on, and a message warns you when access widens. If you'd rather Jay also ask in Work, set "Default access in Work" in Settings, AI behavior.

When a permission card appears, the bar becomes the question: Enter accepts, Esc declines, and a command's card says what it touches and what it risks. When you approve often, Jay tells you: "Approving a lot? Auto mode lets everyday actions go through." Switch when you're comfortable, not before.

What stays confirmed, even in Full. Anything that goes out to a person: an email, a Slack message or one sent from the Mac, a Notion comment, a sent invoice, an event whose guests get notified, a message to another session, and soon a LinkedIn post. Sharing a Drive file too. And a command you've denied in Settings stays denied, even hidden inside a chain.

Plan mode. In the Code position, the "+" offers Plan, or type /plan anywhere: Jay reads and analyzes, but writes nothing; a read command goes through, any other command is asked. It presents a plan, you approve it or have it regenerated, then /build carries it out. It's the right reflex before any task that touches a lot of files.

The delegation brief

Say you ask Jay to "find suppliers for the event." It will search, compare, maybe draft emails, and hand you something very different from what you had in mind. An agent is only as good as its brief. The more autonomous it is, the more the brief must spell out the deliverable, the limits and the moment it's done.

TEMPLATE **The delegation brief**

```
Goal: [the end result, in one sentence]
Deliverable: [file, format, location in the folder]
Context: [what Jay needs to know, and where to find it in the folder]
Resources: [folders, connectors, sites to use]
Limits: [what it must not do: send, delete, buy, publish, modify a given folder]
Done when: [the verifiable criteria that say it's finished]
Before you start, show me your plan in 5 steps max and wait for my approval.
```

I CAN HEAR YOU ALREADY

"If Jay does everything, why review its plan?" Because a misunderstanding at the start gets repeated across the forty files that follow. Two minutes of review at the start save you from redoing everything at the end.

Background tasks and sub-agents

A long task doesn't have to block your session. Say "do it as a background task": Jay launches a separate sub-agent, with its own context and the same tools, and you keep working. The result comes back to the session on its own, even if you've moved on to something else in the meantime. Up to three background tasks run at the same time. The Activity tab of the drawer shows what's in progress, and the header says how many agents are working.

Each sub-agent gets a role, and the mode that goes with it: explore (read and search, fast), build (write), review (check a piece of work), or expert (the deepest reasoning). It inherits the session's context and returns a report suited to its role. A sub-agent works on its own, without asking you anything: with Full access it acts alone, otherwise it stays limited to what your level lets through without a card. Give it a tight brief, with clear limits, and keep it for tasks where a mistake can be fixed. For longer parallel work, Jay can also suggest a sub-session, attached to yours, that you follow in the rail. And in Customize, Advanced, Sub-agents, you create specialized agents with their own brief, which Jay calls when the task fits them.

A long turn no longer stops on its own after a fixed time: Jay goes all the way to the deliverable, and the Stop button stays yours.

The terminal

In the Code position, the Terminal button in the bar opens the Terminal tab of the drawer: a real terminal in the project folder, shared by all the project's sessions. Jay runs its commands there and reads their output; you can type your own. When it gives you a command to run yourself, it writes it in a block you copy with one click. The commands you allow or deny by default are set in Settings, AI behavior.

When not to delegate

- **When a question is enough.** A full task for a one-line answer is wasted volume.
- **When a mistake is costly and can't be undone.** Payments, mass mailings, deletions: keep the final decision, with Ask access.
- **When you don't know yourself what you want.** Clarify first in conversation, or with /goal. Delegate afterward.

Ten tasks to hand to Jay

TASK	WHAT YOU ASK
Tidy a folder	"Sort my Downloads folder by type and by project, rename the files consistently, and list what looks useless without deleting anything."
Expense reports	"Read the receipts in this folder and fill in the Excel expense report template."
Monthly report	"Using the files in this folder, write this month's activity report in the same format as last month's."
Monitoring	"Monitor my 5 competitors this month and summarize the changes in one page."
Presentation	"Turn this report into a 10-slide presentation for the committee, using the brand kit."
Contact database	"Merge these three contact files into one, with no duplicates, and flag incomplete records."

TASK	WHAT YOU ASK
Meeting prep	"Prepare a brief for each of my meetings this week, from the calendar, email and the folder."
Document summary	"Read the 20 PDFs in this folder and make a summary table with the key points of each."
Reminders	"Find the unpaid invoices in Pennylane and prepare a draft reminder for each client."
Comparison	"Compare these 4 supplier quotes in a table and recommend the best one based on my criteria."

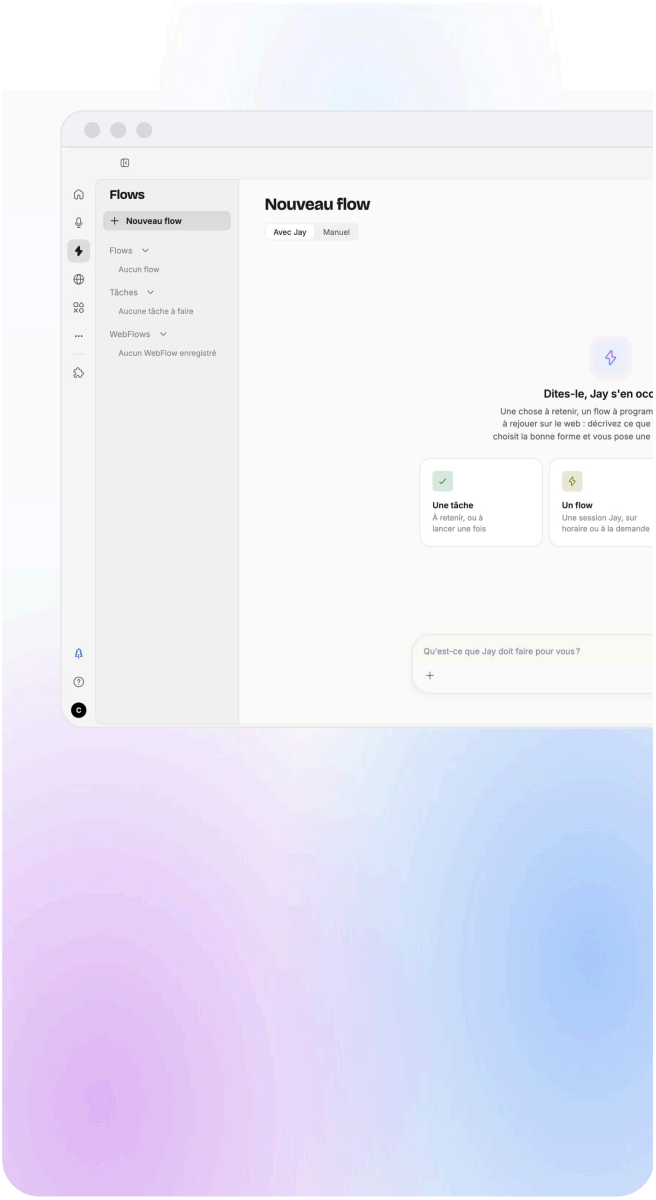
WATCH OUT

A multi-step task uses much more volume than a question. Group related work into a single task, and keep an eye on Settings, Usage. During a turn, Jay prevents sleep itself; "Keep Mac awake" is for flows that run with the window closed (chapter 14).

CHAPTER 14

Flows and watches

A task set up once can then run every day or every week, with the window closed. With Jay, it runs on your Mac, with your folders, your connectors and your apps: that's what no online service can do. This chapter shows you how, with ten recipes ready to schedule.



IN THIS CHAPTER

01 Three ways to automate

02 Flows

03 Scheduling a task: the 3-day method

04 Ten flows ready to schedule

05 Watches

06 What you should never automate

Three ways to automate

TOOL	WHAT IT IS	WHAT FOR
Flows	A task that is scheduled or triggered by a folder, in the Flows place	Recurring tasks: monitoring, reports, day prep, tidying
WebFlows	A recipe recorded in the Browser, that can be replayed	A client portal to check, a form to fill in every month
Watches	A wait set by Jay in a session: it re-checks a command or a page at intervals, without the model, and comes back to the session on the signal	"Let me know when the deployment is done," "come back when the page changes"

Flows

In the Flows place, "New flow," in With Jay mode (you describe, it sets things up) or Manual. Or, more simply, in any session: "schedule this every Monday at 8 a.m.," and Jay creates the flow.

SETTING	OPTIONS
Trigger	Scheduled, or When a file arrives in a watched folder
Frequency	Manual (on demand), Once only, Every day, Weekdays, Once a week, Every hour, Every N minutes
Access level	The same choice as in a session: Ask, Auto, Full
Model	The mode that runs the flow
Linked project	The folder where the flow reads and writes

Resident Jay. The flows clock runs as long as the Mac is awake, with the window closed or never opened. For a morning flow, leave the Mac on, or turn on "Keep Mac awake": lid open, or even lid closed, with the "Only on power adapter" option.

The watched folder. It's the most natural automation on a computer: "every time a PDF arrives in Invoices-to-process, read it, rename it and add a row to the spreadsheet." You drop it in, Jay handles it.

Scheduling a task: the 3-day method

Most beginners schedule a task on the very first try. The following Monday, the result is way off, and they give up. We'll do the opposite: fine-tune first, schedule afterward.

1. **Do it by hand first.** Run the task once, in a session, and fix the brief until you're happy with the result.
2. **Schedule it.** "Schedule this every Monday at 8 a.m.," in Ask or Auto, never in Full to start.
3. **Specify where the result goes.** A file in a folder, an email draft, a Notion page. And how you're notified: a macOS notification.
4. **Review the first three results, then adjust.** On day one, you fix the brief. On day two, the details. On day three, it's set. If a flow still needs corrections after a week, its brief is too vague.

Ten flows ready to schedule

FLOW	FREQUENCY	WHAT IT DOES	YOU NEED
The prepared morning	Weekdays, 8 a.m.	Today's calendar, emails to handle, meeting prep	Calendar, email
Industry monitoring	Every Monday	What's new in your industry, summarized in one page in the Monitoring folder	Web search
The Monday report	Every Monday	Last week's numbers, with commentary, in the usual format	A folder or a data connector
Invoices on arrival	Watched folder	Each PDF dropped in is read, renamed, filed and added to the spreadsheet	A folder
Reminders	Every Wednesday	Invoices unpaid for 30 days, with a draft reminder	Pennylane or a spreadsheet, email
The monthly statement	Manual (on demand), on the 1st	A WebFlow that downloads the statement from your client portal	The Browser
Tomorrow, ready	Every evening	A brief for each of tomorrow's meetings	Calendar, email, Drive
Tidying up	Every Friday	Downloads sorted and renamed, without deleting anything	Nothing
The newsletter digest	Every Saturday	The week's newsletters summarized in 10 points with the links	Email
Goal tracking	Manual (on demand), at month's end	Where you stand on your quarterly goals, and what's blocking	A project with your goals

PROMPT **Monday monitoring (flow scheduled every Monday at 8 a.m.)**

```
Monitor [industry] over the last 7 days.
Players to follow: [list]. Priority topics: [list].
Deliverable: one page with 5 key facts (source and date for each), 3 weak signals, and
"What this changes for me" in 3 points.
Save it in Monitoring/ as "Monitoring – [date]", reread the previous one so you don't
repeat yourself, and notify me. Don't make anything up: if the week was quiet, say so.
```

PROMPT **Invoices on arrival**

```
Watch the Invoices-to-process folder. For each new PDF: read it, rename it "YYYY-MM-DD
supplier amount.pdf", move it to Accounting/2026/[month], and add a row to the Invoices-
2026.xlsx spreadsheet: date, supplier, amount before tax, VAT, total including tax, due
date.
If any information is unreadable, leave the cell empty and add "to check" in the last
column. Never delete anything.
```

Watches

A flow comes back at the set time. A watch comes back at the right time. When a turn ends on a wait (an export running, a deployment, a page that should change, a file that should appear), Jay sets a watch instead of asking you to come back: it re-checks the command or the address at intervals, without making the model work and so without using your volume, until the output changes, succeeds or contains the expected text. On the signal, it wakes up in the same session, with all its context, and picks up where it left off.

The Activity tab of the drawer shows the watches in progress: what they re-check, how often, the next check. You can stop one; its session then won't be woken up. You can also ask for one yourself:

PROMPT **A watch on a page**

```
Re-check [address] every 10 minutes. As soon as [the product is back in stock / the result
is published / the page shows "…"], come back here and tell me what changed. Don't buy
anything, don't fill anything in.
```

Scouts, the named agents with long-running missions that some accounts were trying out in the Lab, are paused for now: their useful routines can be rebuilt as flows.

What you should never automate

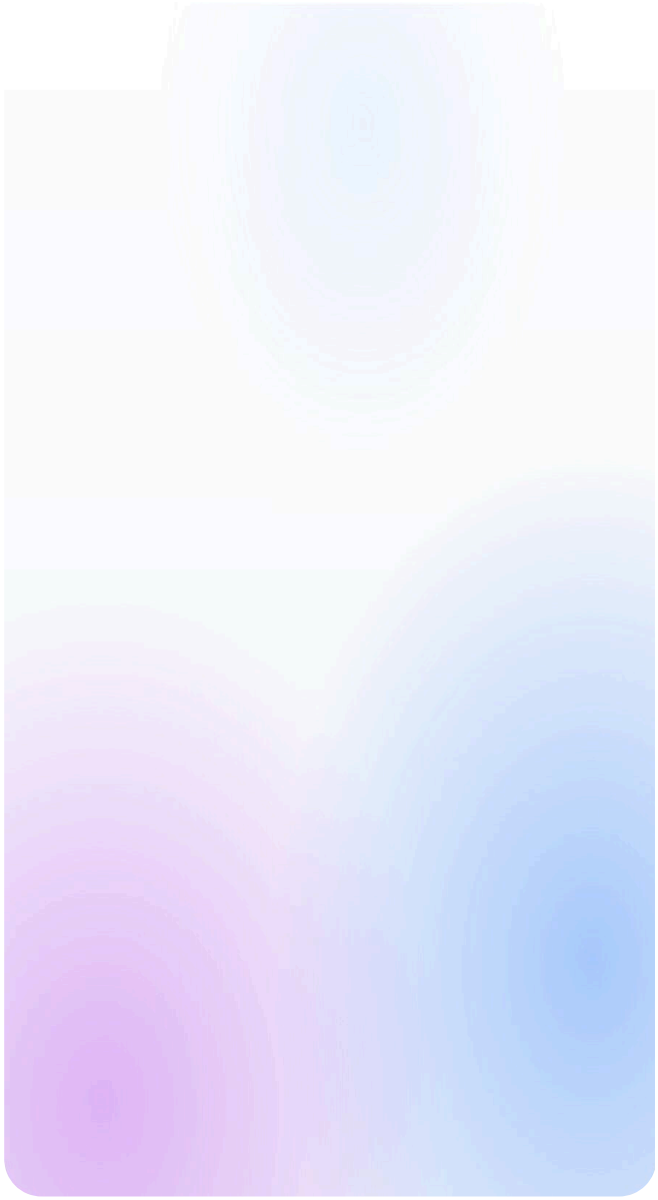
WATCH OUT

Sending emails without review: stick to drafts. Payments, deletions and public posts. Decisions that affect people: hiring, evaluation, discipline. Anything that handles sensitive data without your company's approval. A flow with Full access in a folder where a mistake can't be undone.

CHAPTER 15

Jay and code

Jay was born as a code editor, and it shows: attach a Git repository to a session, the bar switches to Code on its own, and Jay reads the code, edits files, runs commands, runs the tests and prepares the commits. Developers find a pair-programming partner in it; people who don't code find the fastest tool as soon as a task touches hundreds of files.



IN THIS CHAPTER

01 A code session

02 The isolated copy

03 Sessions that talk to each other

04 Your first code session

05 Best practices

06 Jay for people who don't code

A code session

BUILDING BLOCK	WHAT IT IS
JAY.md	The file at the root, read in every session: conventions, commands, architecture, pitfalls (chapter 4). Ask Jay to write it from the project
The Code position	In the bar: the branch, the isolated copy, the "+12 −3" diff of the work in progress, the Terminal button. In the drawer, the Code tab: the modified files, the diff of each change
/commit, /diff, /status, /pr	The Git moves. Jay never commits or pushes on its own: /commit is your move, and it writes the message following your rules (Settings, Git); /pr writes the pull request description, which Jay then opens on your forge after confirmation
Forges	GitHub, Forgejo, GitLab, Bitbucket, in Settings: Jay opens, lists and reviews pull requests
The Terminal	Shared per project, where Jay runs and reads its commands, with your allowed and denied commands
Command hooks	A script before or after each of Jay's actions: format a modified file, run the linter (chapter 11)
The Design System	The drawer tab that extracts the real system from the code, its drift and its fixes with proof (chapter 6)
The simulator	Jay installs, launches and drives your app in the iOS simulator or an Android emulator: it taps, scrolls, takes screenshots
DOPE	Code agents in a container, with your Jay account, in every Jay plan

The isolated copy

Next to the branch name, a checkbox: the isolated copy. Jay then works in a copy of the repository, on a real branch, without touching your working tree. You keep coding in yours while it moves forward in its own, and you merge when it's good. Memory and the knowledge base stay the project's.

Sessions that talk to each other

Jay searches your other sessions and your meetings when you refer to them: "pick up what we decided yesterday about the API." Two more moves, open to everyone: link another of your Jay projects as read-only (the front end that reads the back end), and write to another session, which replies on its own when it's idle. Every send is confirmed with you, even with Full access, and a reply doesn't trigger another: no loop between two agents.

Your first code session

1. **Attach the project folder.** If it's a Git repository, the session opens in Code; otherwise, flip the switch to Code.
2. **Have it explore first.** "Explain this project to me: its structure, its purpose, how to run it."
3. **Ask for the JAY.md.** "Write the JAY.md: commands, architecture, conventions, pitfalls you notice."
4. **Give it a small real task, in Plan mode.** Approve the plan, switch to Build, let it work.
5. **Review the diff in the Code tab, then /commit.**

Best practices

1. **Explore, plan, code, verify.** Always ask for a plan before big changes.
2. **Small, precise tasks.** "Add email validation to the sign-up form" rather than "improve the site."
3. **Tests as the final judge.** Ask for the test to be written first, then made to pass.
4. **Git as a safety net.** A commit before every big change, or the isolated copy.
5. **A short, living JAY.md.** Add a rule every time you correct the same mistake twice.
6. **The right mode.** Versatile for small touch-ups, Thinking for a feature, Expert for a redesign or a stubborn bug.

Jay for people who don't code

You don't need to be a developer to benefit. As soon as a task touches a lot of files or data, a code session is the fastest tool, because Jay writes a small program to do it in one go, and shows you its plan first.

AREA	WHAT YOU ASK
Files	"Rename the 400 photos in this folder with the date and place, and sort them by month."
Data	"Merge the 12 monthly Excel files into one, clean up the duplicates and make a summary table."
Web	"Create a 4-page showcase site for my consulting business, with my brand kit, and explain how to put it online."
Documents	"Convert all these Word documents to PDF with the same layout and a table of contents."
Mini-tools	"Build a small tool, a local web page, where I paste a job posting and my résumé, and that gives me a score and the missing skills."

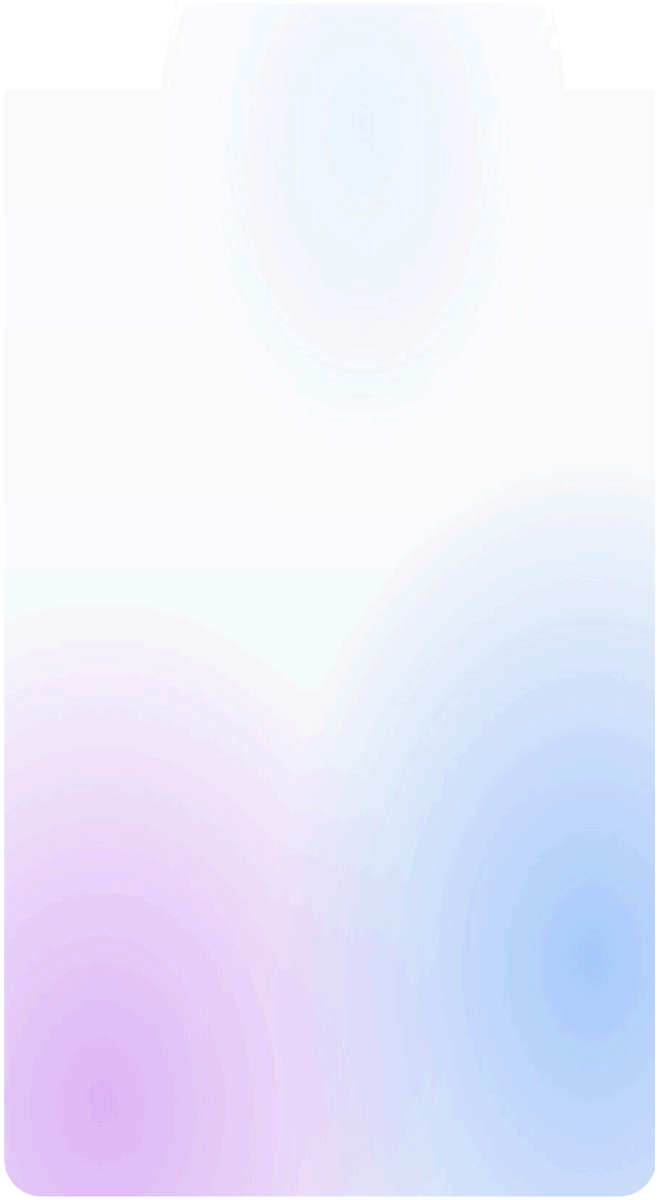
TIP

Start a mini-tool with the simplest version that works, test it with your real data, then add one feature per message. A tool requested with ten features at once is fragile, and you no longer know which feature breaks what.

CHAPTER 16

Complete workflows

The building blocks show their full value when you chain them together. Here are eight end-to-end workflows, drawn from real work situations. You'll learn almost nothing new here, and that's on purpose: each workflow takes building blocks from the previous chapters and assembles them over a week that looks like yours. Find the one closest to your own, and start there.



IN THIS CHAPTER

01 Meetings: from meeting to action items

02 The monitoring brief that runs itself

03 Preparing for a client meeting

04 From rough draft to LinkedIn post

05 Competitive analysis

06 Monthly reporting

07 Processing documents in bulk

08 A code project, from requirement to pull request

WORKFLOW 1

Meetings: from meeting to action items

Meetings | Skill | connectors

1. Launch "New meeting in this project" when the meeting starts. Jay records and transcribes.
2. At the end, your /cr Skill pulls out decisions, action items, deadlines and open questions, and files the minutes in the folder.
3. Jay prepares follow-up email drafts in your mailbox and adds the deadlines to your calendar, once you approve.
4. The following week, "what did we decide about the deadline?" finds its answer in the transcript.

WORKFLOW 2

The monitoring brief that runs itself

Project | web search | Flow

1. Create a "Monitoring" project with its folder and the JAY.md from chapter 4: players, topics, sources.
2. Run a first monitoring brief in Thinking mode and refine it until you like the format.
3. Schedule it every Monday at 8 a.m., in Auto, with a notification.
4. Every quarter, ask for a summary of the last twelve briefs in the folder: that's where the trends show up.

WORKFLOW 3

Preparing for a client meeting

Connectors | Pappers | Browser | project

1. Jay reads your calendar and spots the meeting.
2. It looks up the history of your exchanges in your email and the documents in the client's folder.
3. It adds the company's Pappers record and a search on its latest news.
4. It writes a brief in the Atelier: goals, questions, likely objections, opening line. With the "Ready for tomorrow" flow, it's waiting for you every evening.

WORKFLOW 4

From rough draft to LinkedIn post

Project · ideation

1. Dictate your idea as it comes, or paste it into the "My voice on LinkedIn" project.
2. Ask for 3 angles and 5 hooks; pick one.
3. Have Jay write it using your reference texts, then self-critique: "What would make someone stop reading?"
4. You publish it yourself; the LinkedIn connector, once released, will post after your confirmation. The approved post is archived in the folder.

WORKFLOW 5

Competitive analysis

Web search · Atelier

1. One in-depth search per competitor, in Thinking mode: offering, pricing, positioning, news. As a background task if you have five.
2. A comparison spreadsheet on your criteria, with the source for every cell.
3. An analysis: strengths, weaknesses, open spaces in the market.
4. An 8-slide presentation for your team, in your brand kit, exported to PowerPoint.

WORKFLOW 6

Monthly reporting

Watched folder · spreadsheet · document · Flow

1. The exports land in a watched folder, or Jay downloads them with a WebFlow.
2. Jay calculates the metrics with code, in a spreadsheet with visible formulas, and compares them to the previous month.
3. It writes the commentary in the format of your last report, with the anomalies up front.
4. You check three numbers, adjust the commentary, and send it.

WORKFLOW 7

Processing documents in bulk

Background task · folder · spreadsheet

1. Open the folder to Jay: invoices, contracts, resumes or forms.
2. Give it the output template: a spreadsheet with the columns you want, plus a "to check" column.
3. As a background task, Jay reads each document and fills in one row per file, flagging the doubtful cases.
4. You only handle the flagged cases. From then on, the folder becomes a watched folder.

WORKFLOW 8

A code project, from requirement to pull request

Plan mode · isolated copy · tests · Forge

1. /goal to turn the requirement into a specification, in a project session, switched to Code.
2. Plan mode: Jay explores and proposes a plan; you approve it.
3. Isolated copy: Jay codes on its own branch, writes the tests first, and makes them pass.
4. You review the diff in the Code panel, /commit, then /pr writes the description and Jay opens the pull request on your forge, after confirmation.

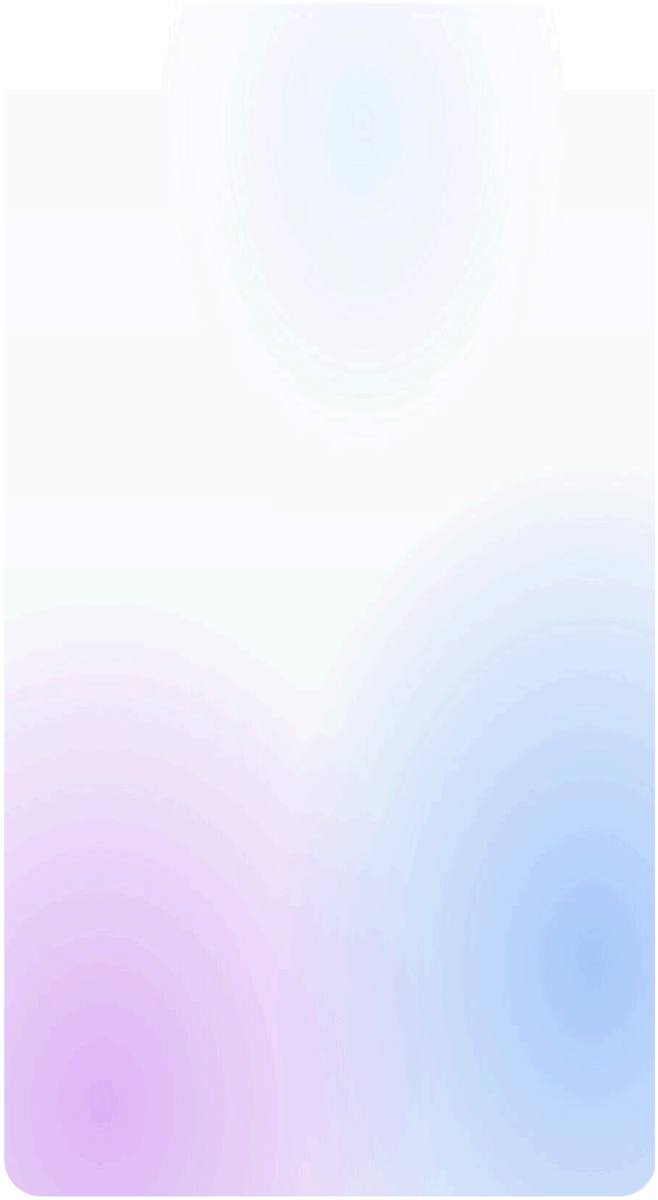
PROMPT **Turning a workflow into a system**

```
Here's a task I do regularly: [describe the workflow step by step, with the tools and folders used].
Analyze it and propose: 1) the steps you can do on your own, 2) the ones that should stay human, 3) the best combination of project, JAY.md, Skill, connectors and Flow to set it up, 4) the permission level for each step, 5) the first step to take today.
```

CHAPTER 17

Best practices and security

The people who really save time with Jay share a handful of habits: they brief, they check, they keep what works, and they know what you don't hand over to an agent that has control of the computer.



IN THIS CHAPTER

01 The 15 golden rules

02 The most common mistakes

03 Checking without rereading everything

04 Security and privacy

The 15 golden rules

#	THE RULE	WHY
1	A 5-point brief for any task that matters	Goal, context, deliverable, criteria, example: less back and forth
2	Let Jay ask you questions	It spots what's missing before it gets things wrong
3	One session per topic	Long sessions mix topics and fill up the context
4	One project per recurring topic, with its folder	Context, memory and files stay clean
5	Edit your first message	The fix pays off every time after that
6	Anything you correct twice becomes a rule	In the JAY.md, a Skill or your job profile
7	The right mode for the right task	Thinking or Expert for what matters, Adaptive for the rest
8	Ask for sources and uncertainty	You know where to check and what's shaky
9	Calculations with code, in a spreadsheet	A model doing math in its head can get it wrong; a formula can be checked
10	A plan before any delegation	You correct course before it touches the files
11	Drafts, never direct sending	You keep the final say on what goes out
12	Ask access to start with	Autonomy is earned task by task
13	Say where the result goes	A deliverable has an address: a folder, the Atelier, a draft
14	Keep your best prompts	In a "Prompts" project or in Skills: your memory won't be enough
15	File each new feature under its building block	You understand in five minutes what takes others a month to keep up with

The most common mistakes

THE MISTAKE	INSTEAD
Using Jay like a search engine, one question at a time	Open a folder to it and hand it a complete job, with a goal
Copying an answer without rereading it because it sounds confident	Ask for sources and uncertain points, and check the three main claims
A single 200-message session for everything	One session per task, a project for what comes back, /compact when the meter rises

THE MISTAKE	INSTEAD
Leaving Full access on everywhere from day one	Ask first, then Auto task by task as trust builds; Full only for Atelier pieces and well-tested tasks
Launching a flow without having done it by hand	The 3-day method: fine-tune first, schedule second
Hunting for the perfect prompt online	Write a clear brief about your situation: no generic prompt knows your folder

Checking without rereading everything

REFLEX	WHAT YOU ASK
Sources	"Cite the source of every number and every fact, file and page."
Uncertainty	"Flag what you're not sure about instead of guessing."
Second opinion	"Reread your answer like a demanding expert: what's wrong, vague or missing?"
Calculation	"Redo this calculation with code and show me the formula."
Opposing view	"What's the best argument against your conclusion?"
Files touched	"List every file you created, modified or moved in this turn." Or the drawer's Files tab

Security and privacy

Sensitive data. Personal data about clients or colleagues, confidential information, health data: check your company's policy first. The models are hosted in France and don't learn from your data, but a company rule comes before your settings.

Your credentials. Passwords, access keys, card codes: no task justifies pasting them into a session. In any case, Jay will never enter a card number, or a password outside a local page. Connector keys, on the other hand, go in the app's card, in Customize, where they stay on your Mac, encrypted.

The safeguards to set up on day one. Always denied commands in Settings, AI behavior (anything that deletes, sends or publishes). Default access in Work set back to Ask if you're just starting out. Your sensitive sites blocked in Settings, Browser. A Rule hook, in Customize, Advanced, that refuses any write to your archive folders. Touch ID if the Mac is shared.

Prompt injection. Chapter 9 explains it: content Jay reads can try to hijack it. If Jay suddenly proposes an action unrelated to your request, refuse it, keep confirmation on for important actions, and leave "Don't memorize sessions that read the web or an MCP" turned on.

For a one-off topic. An incognito session: nothing is saved, no memory, no sync, and reduced tools: read, search the web, write to a throwaway folder; no terminal, no browser, no connector.

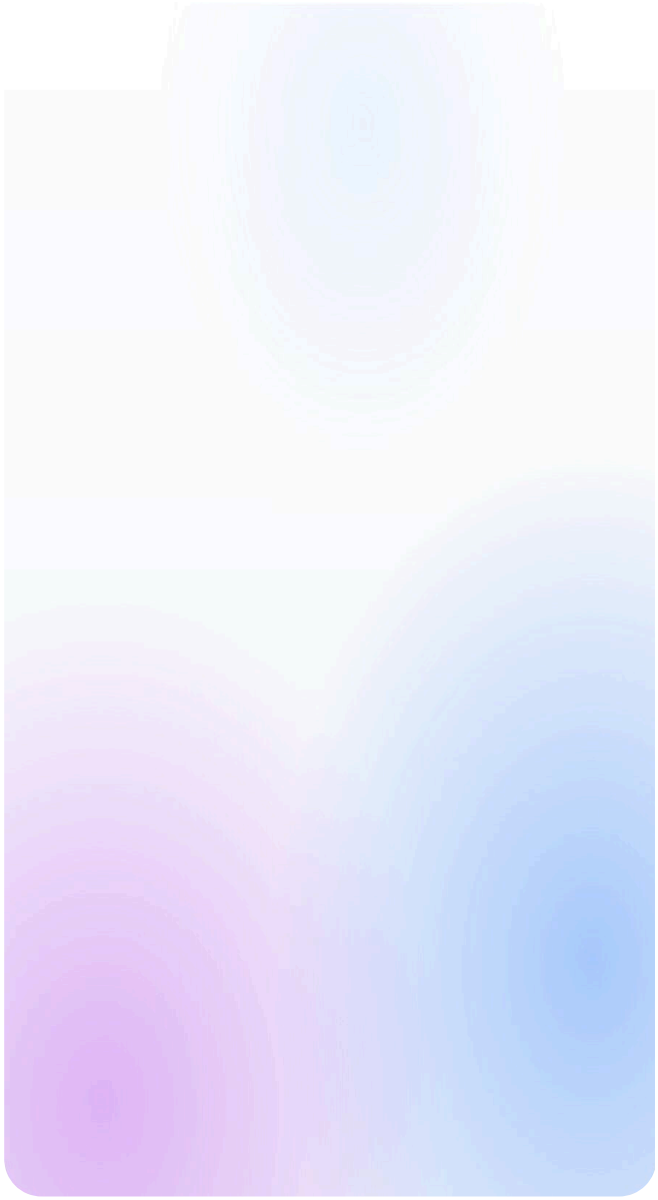
THE PRINCIPLE BEHIND IT ALL

Think of Jay as a brilliant, fast, tireless colleague who's new to your company and to whom you've just handed the office keys. You give it context, you check its work at the start, and you widen its room to maneuver as it proves itself.

CHAPTER 18

Your 7-day plan

A guide you read and close saves you no time at all. Thirty minutes a day for a week, and Jay will be part of the way you work.



IN THIS CHAPTER

01 Your final checklist

02 Resources

DAY	GOAL	WHAT YOU DO	CHAPTERS
1	The basics	Job profile, default access levels, denied commands, first real test with the 5-point brief	1, 2, 3
2	Your priorities	The audit of your week (below) and your top 3 tasks to hand over	18
3	Your first project	A project with its folder and its JAY.md for your #1 task	4
4	Your tools	Two apps in Customize, and your morning prepared by Jay	10
5	Create	A document, spreadsheet or presentation in your brand kit, replacing a real deliverable from the week	5, 6
6	Delegate	A complete task on a folder, with the delegation brief, in Ask access then Auto	13
7	Automate	Your first flow, the 3-day method, and a monitoring brief	14

PROMPT **The audit of your week**

```
Audit my week. Ask me 8 questions, one at a time, about my job, my recurring tasks, my tools, my folders and my sensitive data.
Then: 1) a table of my tasks with their frequency and time spent, for me to approve; 2) each task scored on volume, repetition, how feasible it is for you and risk; 3) the top 3, each with the building block to use (project, Skill, connector, Flow) and the permission level; 4) for task #1, the complete brief and the project's JAY.md, ready to copy.
Save it all in an Atelier document.
```

Your final checklist

- [ ] My job profile is filled in
- [ ] I have my top 3 tasks to hand over to Jay
- [ ] A project with its folder and its JAY.md is running for my #1 task
- [ ] Two apps are connected
- [ ] I've created an Atelier piece that was actually used
- [ ] I've delegated a complete task on a folder
- [ ] A flow runs on its own every week

Resources

RESOURCE	WHAT FOR
Mini Jay (⌘⇧J)	A question about Jay, anytime: where to set what, how to do it
Settings, Release notes	What changed with each update

RESOURCE	WHAT FOR
Settings, Models Status	If a mode is responding poorly, look here first
Privacy policy	The models, where they run, what is kept
Terms of service	The plans and their rules

WELL DONE

You now know how to write a brief that holds up, give Jay a folder and a JAY.md, connect it to your tools, hand it a complete task on your computer, and run a flow without you. Jay changes every week. The building blocks stay. Come back to this guide when a new feature comes out, and file it in its place.

Glossary

TERM	WHAT IT IS	BUILDING BLOCK
Session	A conversation with Jay, where you talk, write pieces and code; viewed in Work or in Code	
Work / Code	The switch in the bar: Work shows pieces and images, Code shows the branch, the diff and the terminal; each position has its own access level	
Project	A group of sessions and meetings, attached to a folder on the Mac, or with no folder just for organizing	Context
JAY.md	The instructions file at the root of a project, read in every session	Instructions
Jay/ folder	The only subfolder where Jay writes on its own: knowledge base, memory, brand kits, pieces	Memory
Linked folder	A folder Jay reads but can never write to	Context
Place	A page in the icon bar: Home, Meetings, Flows, Browser, Atelier, Customize, Settings	
Drawer	The right half of a session, in panels: Session info, Activity, Web, Files, Code, Terminal, Atelier, Design System, Brand kit	
Mode	The engine of a session: Adaptive, Versatile, Thinking, Expert; each one shows the open source model behind it	
Effort	How long a mode spends thinking before answering: Minimal, High, Max; Automatic in Adaptive	
Ultra intelligence	A stricter method: critical review when in doubt, tests run by Jay; uses more	Verification
Access level	Ask (confirmation before every action), Auto (routine actions approved, diff before writing), Full (no confirmation, except for anything going out to a person)	Verification
Plan mode	Jay reads and analyzes, writes nothing, asks before any command that isn't a read, and presents a plan for you to approve	Verification
Image mode	Messages produce images; chosen from the "+" or with /image	Tools
Piece	A file produced by Jay: document, spreadsheet, presentation, mockup, video	Tools
Atelier	The place and the drawer panel that bring together pieces, images and brand kits	Tools
Brand kit	Your visual identity as named tokens, with its components, rules and templates; it has its own session	Context

TERM	WHAT IT IS	BUILDING BLOCK
Design System	The actual system of a code project, extracted from the code, with its drift and its fixes	Context
Skill	A reusable method Jay applies when the task calls for it, or on its /name command	Instructions
Pack	A set of Skills by role, turned on in one move	Instructions
Job profile	Who you are, your rules, and the setup Jay suggests you install	Context
Hook	A rule or command triggered at a specific moment: before a tool, after a reply...	Instructions
Connector	Controlled access to an online tool: Gmail, Drive, Calendar, Notion, Qonto, Pennylane, HubSpot, Slack...	Tools
Customize	The place for Applications and Skills, with Roles and, under Advanced, MCP servers, hooks and sub-agents	Tools
MCP	The open standard that connects an AI to any tool; Jay connects to it without any pre-registered credentials	Tools
Memory item	A lasting fact Jay remembers, in the project memory or the global memory	Memory
Context	Everything Jay reads with each message; the meter shows how it breaks down, /compact summarizes it	Context
Background task	A sub-agent launched separately, with a role (explore, build, review, expert), that works while you carry on and returns its result to the session	Agents
Sub-session	A session attached to yours, for longer parallel work	Agents
Flow	A scheduled task, or one triggered by a file arriving, that runs with the window closed	Agents
WebFlow	A Browser recipe, recorded and replayable	Agents
Watch	A wait set by Jay: it rechecks a command or a page without a model and comes back to the session on the signal	Agents
Resident Jay	What keeps running when the window is closed: the flow clock and the watches	Agents
Isolated copy	A copy of the repository on its own branch, where Jay codes without touching your working tree	Agents
Meeting	A meeting recorded, transcribed and summarized	Memory
Lexicon	The proper nouns and terms the transcription should correct automatically	Context

TERM	WHAT IT IS	BUILDING BLOCK
Mini Jay	The companion: a quick question about Jay or anything else, with nothing kept	
Lab	Preview features, open to a few accounts, with a switch per feature	
Volume	The work included in your plan, per week and per month, in tokens (a one-time allowance for Jay Trial); visible in Settings, Usage	
Checklist	A list shown in the thread whose boxes you check; Jay knows what's done	Tools
Incognito	A session that saves nothing, remembers nothing and doesn't sync	Verification
Prompt injection	Instructions hidden in a page or document that try to hijack the agent	Verification