The FormAssembly Guide to Forms

Download the PDF of this article.

In this Article

Related Articles

Introduction

FormAssembly is a platform for collecting data and moving it where it needs to go. It starts with forms, but it doesn't end there.

If you're new to FormAssembly, it's normal to feel unsure about where to begin. You might be familiar with forms in general, or you may have inherited an existing setup and are trying to understand how everything fits together. This resource is meant to help orient you, but it won't turn you into an expert overnight.

The goal of this document is to help you get started by highlighting the most common and foundational features of FormAssembly. It is not an exhaustive list of everything the platform can do, and it is not meant to cover every possible configuration or use case. Instead, it focuses on giving you a clear starting point and a sense of what's possible as you begin building.

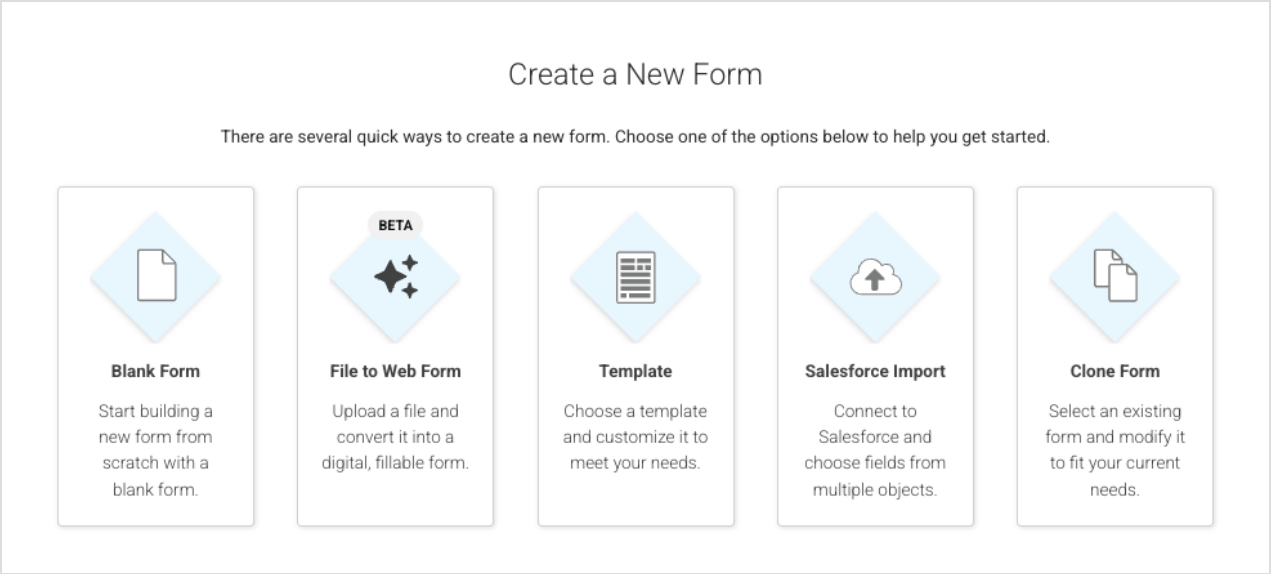
You don't need to understand every feature to be successful. Many people start with a simple form and gradually add more structure or automation as their needs grow. You may discover additional features that support more advanced scenarios.

As you read through these sections, you'll see how forms are built, how data can be shaped and protected, and how that data can move to other systems after submission. By the end, you should have a clearer sense of what you can build, what you might want to try next, and where to go when you're ready to learn more.

Create Your First Form

New Forms in FormAssembly's Classic Platform

Creating a form begins when you click New Form. This is the first real decision point in FormAssembly, and it's also one of the easiest to overthink. The options you see here are not different types of forms. They are simply different ways to arrive at the same place.



Every option leads to the same form builder, and everything can be changed later.

Starting with a blank form gives you a clean slate. This option is often the most straightforward when you're learning, because you add each field intentionally and can see how the form takes shape piece by piece. If you're unsure where to begin, this is a safe and flexible choice.

Templates offer a different kind of starting point. They include prebuilt fields and structures designed around common use cases. A template can be helpful if there is one that closely resembles what you need. You can remove, rename, or rearrange anything within a template. It is a shortcut, not a commitment.

You may also see options to create a form by importing from Salesforce or converting an existing file, such as a PDF or spreadsheet. These options are useful when your form already exists elsewhere, and you want to avoid recreating it manually. Using these options results in a fully editable form, just like any other option from this screen.

Cloning an existing form creates a copy of something that you already have in FormAssembly. This is especially useful if you've inherited forms from someone else or want to experiment without affecting the original version.

What matters most at this stage is not which option you choose, but that you choose one. Forms in FormAssembly are designed to be revisited, adjusted, and improved.

Once the form is created, the real work begins in the Form Builder.

You can learn more in this help document.

The Form Builder

Turning an Empty Form Into Something Intentional

Once your form is created, you're taken into the Form Builder. This is where you decide what information you're collecting and how the form will feel to someone filling it out. It's also where many people realize that they have more control than they expected.

At its core, the Form Builder is about arranging questions in a way that makes sense. You add fields, you organize them, and you adjust things as your understanding of the use case improves. You don't need a finished plan before you start. In fact, most forms improve because they're built iteratively.

Starting With the Basics

One of the first things you'll notice is the form title. This title helps you identify the form later, and you can choose whether or not it should be visible to respondents. Some forms benefit from a visible title that sets context, while others work better when the page flows directly into the questions. Either choice is easy to change.

Adding questions happens through Add Content > Question. Each field you add represents a single piece of information you want to collect. As you build, you can add fields, remove them, rename them, and reorder them. Making changes to your form is a normal part of the process! Don't be afraid to just get started.

Try this out here!

Organizing for Clarity

As forms grow, organization starts to matter more than the number of questions. Sections, groups, and pages exist to help structure longer forms so they're easier to understand and complete.

Sections and groups help visually organize related questions, while pages let you break a form into steps. You don't need to use all of these tools, and many effective forms use only one or two. The goal isn't complexity, it's clarity.

You can learn more in this help document.

Reusing What You Build

Over time, you may notice that you're creating the same fields or sets of questions across multiple forms. Predefined Content allows you to save those pieces so they can be reused later. This becomes especially valuable as your library of forms grows or when multiple people are building forms in the same account.

You don't need to plan for reuse on day one. It's there for when repetition starts to appear.

You can learn more in this help document.

Previewing and Iterating

Previewing your form lets you see it from the respondent's perspective. This is one of the easiest ways to catch confusing wording, awkward flow, or missing context. Many builders preview frequently and make small

adjustments as they go.

We encourage you to save your work often, especially as you move, update, and remove parts from your form. (You never know when life might throw you a power outage!) The form builder is designed to support experimentation and refinement, but make sure you're keeping your work saved as you go!

You can learn more in this help document.

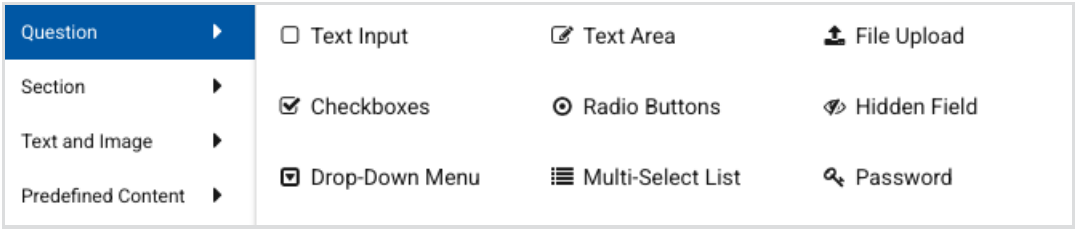
Multiple Choice Fields

Designing Answers Instead of Interpreting Them

Any time a respondent types a free-text answer, you're asking them to interpret the question, and you're taking on the work of interpreting their response later. Multiple choice fields flip that dynamic. You define the possible answers up front, which leads to cleaner, more consistent data from the start.

Choosing How People Respond

FormAssembly offers several types of multiple choice fields, including dropdowns, radio buttons, and checkboxes. While they look different, they all serve the same purpose: guiding respondents toward structured answers.



Dropdowns work well when a list is long or when you want to conserve space. Radio buttons are useful when it's important for all options to be visible at once. Checkboxes are ideal when more than one selection is allowed. What matters most is that you're deciding how answers should be given, rather than leaving everything open-ended.

You can learn more in this help document.

Shaping the Choices Themselves

Choices within a multiple choice field are fully editable. You can add new options, rename existing ones, reorder them, or remove them entirely. This makes it easy to adjust language, clarify meaning, or prioritize the most common selections.

Multiple choice fields are also where forms often begin to feel more responsive. Using features like menu dependencies, the options shown in one field can change based on a previous answer. This keeps forms focused and prevents respondents from seeing choices that don't apply to them.

When Choices Come From Your Data

When your choices already exist somewhere else, you don't need to recreate them by hand. Dynamic Picklists allow a multiple choice field to be populated directly from Salesforce data.

Instead of managing static lists that can become outdated, your form can reflect what's actually in Salesforce at the moment someone fills it out. That means options stay accurate without ongoing maintenance, even as records change, new values are added, or old ones are retired.

Dynamic Picklists can represent standard Salesforce picklist values, but they can also represent entire Salesforce records. To a respondent, this still looks like a familiar dropdown or list of options. Behind the scenes, however, each choice is tied to real data. Selecting an option isn't just choosing a label; it's selecting a record.

This is where forms start to feel less like questionnaires and more like live entry points into your systems. Dynamic Picklists make it possible to guide respondents using existing data, reduce errors, and ensure that what gets submitted lines up cleanly with how your Salesforce org is structured.

Once you start using them, it becomes much easier to imagine forms that stay in sync with your data instead of constantly chasing it.

You can learn more in this help document.

Why Multiple Choice Fields Matter

By the time you finish working with multiple choice fields, you're no longer just deciding how questions look. You're deciding how reliable, usable, and connected your data will be after submission.

Multiple choice fields help reduce ambiguity, prevent cleanup work later, and make it easier for your forms to scale as your needs grow.

This is one of the places where small design decisions can have an outsized impact.

Repeatables

Collecting More Without Collecting More Forms

Repeatables solve a very common problem: needing the same kind of information more than once. Instead of asking someone to submit multiple forms or squeezing everything into a single long list of fields, repeatables let respondents add additional entries as needed within one form submission.

Fields, sections, and even entire pages can be made repeatable. To the respondent, this usually appears as an option to "add another" entry. Behind the scenes, each repeated set of fields is captured cleanly and consistently, making the data easier to work with after submission.

This approach is especially useful when you don't know in advance how many entries someone will need to provide. For example, a person might need to list multiple contacts, multiple line items, or multiple experiences. A repeatable allows the form to adapt to the respondent instead of forcing them into a fixed structure.

You can learn more in this help document.

Control Without Complexity

Repeatables don't have to be unlimited. You can choose to limit how many times something can be repeated, or allow respondents to add as many entries as they need. This gives you control without requiring additional logic or multiple versions of the same form.

Because repeatables are built into the form itself, they reduce the need for workarounds like duplicate fields or follow-up forms. The result is a cleaner experience for respondents and more structured data for you.

Why Repeatables Matter

Repeatables help keep forms shorter, clearer, and more flexible. Instead of designing for the most extreme case, you can design a form that works well for everyone.

By using repeatables, you're not just collecting more data. You're allowing your form to scale naturally based on the person filling it out.

Validation Rules, Default Values, and Hint Text

Helping People Get It Right the First Time

Even the best-designed form can fall apart if respondents are unsure of what to enter. Validation rules, default values, and hint text exist to quietly guide people as they fill out your form, so you get usable data without adding friction or frustration.

These tools don't change what your form collects. They change how clearly your form communicates expectations.

Validation Rules: Guardrails, Not Roadblocks

Validation rules make sure that responses follow a specific format before the form can be submitted. This might mean confirming that an email address looks like an email address, that a date is entered correctly, or that a number falls within an expected range.

When used thoughtfully, validation rules prevent small mistakes from becoming bigger problems later. Instead of fixing data after the fact, you're catching issues at the moment they happen, while the respondent is still there to correct them.

Validation rules work best when they feel predictable and reasonable. They should reinforce what the question is asking, not surprise the respondent with unexpected restrictions.

You can learn more in this help document.

Default Values: Starting From a Helpful Place

Default values allow a field to be pre-filled when the form loads. Sometimes this saves time, and sometimes it simply provides a useful example of what belongs in the field.

A default value can suggest the type of response you're looking for, reflect a common choice, or carry over known information that doesn't need to be retyped. Respondents can usually change or remove default values, so these are meant to assist, not decide for them.

Used well, default values reduce effort and make forms feel faster and more intentional.

You can learn more in this help document.

Hint Text: Setting Expectations Up Front

Hint text appears near a field to explain what kind of information should be entered. It's one of the simplest ways to reduce confusion, especially when a field name alone might be open to interpretation.

Good hint text answers the question the respondent is already thinking: "What does this mean?" or "What format should I use?" Even a short clarification can prevent incorrect entries and unnecessary back-and-forth later.

Hint text works quietly, but its impact can add up across a form.

You can learn more in this help document.

Why These Tools Matter Together

Validation rules, default values, and hint text are most effective when they work as a set. Hint text explains expectations, default values offer a starting point, and validation rules confirm the final result.

Together, they shift effort away from cleaning up data later and toward collecting it correctly from the start. The form becomes clearer, more forgiving, and easier to complete.

At this point, you're not just designing questions. You're designing understanding.

Conditional Rules

When Forms Respond in Real-Time

Conditional rules allow your form to react to what a respondent enters. Instead of showing everyone every question, you can show or hide parts of the form based on earlier answers. This keeps forms shorter, more relevant, and easier to complete.

At a basic level, conditional rules help you avoid asking unnecessary questions. If someone selects an option that makes a follow-up question irrelevant, that follow-up never needs to appear. The form adapts in real time, and the respondent only sees what applies to them.

You can learn more in this help document.

What Can Be Conditional

Conditional rules can be applied to many parts of a form, not just individual fields. You can show or hide fields, sections, groups, pages, and even the submit button. This gives you flexibility to control both the flow and the structure of the form without creating multiple versions.

Conditions are triggered by responses to other fields, such as text inputs, dates, or multiple choice selections. This makes conditional rules a natural extension of the work you've already done designing clear questions and structured choices.

Simple Logic, Big Impact

Conditional rules support logical operators like AND and OR, which allow you to combine multiple conditions into a single rule. These operators help you describe when something should happen in a way that more closely matches real-world decision making.

An AND condition means that all specified conditions must be true for the rule to apply. This is useful when something should only happen if several criteria are met at the same time. For example, a follow-up question might appear only if a respondent selects a specific option and answers yes to a related question.

An OR condition means that any one of the specified conditions can be true. This is helpful when the same outcome should occur for multiple possible answers. For example, a section might appear if a respondent selects option A or option B, without requiring separate rules for each case.

As forms grow, conditional rules can also be layered or nested. This allows you to group conditions together so that one set of logic is evaluated before another. Nesting makes it possible to express more nuanced scenarios, such as showing a section only when a specific combination of answers is given.

When logic becomes harder to read at a glance, that's usually a sign to pause and re-evaluate the rule from the respondent's perspective. Asking "What am I trying to show, and when?" often clarifies whether a rule needs to be simplified or broken into smaller pieces.

Why Conditional Rules Matter

Conditional rules often fade into the background once they're in place. What respondents notice instead is that the form feels shorter, clearer, and more relevant to them.

Logic makes this possible by allowing the form to respond to different combinations of answers without asking extra questions or branching into separate paths. A single form can support many situations while still feeling straightforward.

This flexibility is what allows forms to grow in capability without becoming harder to use.

Calculated Fields

Using Calculations in Your Form

Calculated fields allow your form to create new values based on information that's already been entered. Instead of asking respondents to do math or repeat information, the form can derive those values automatically.

At their simplest, calculated fields can handle straightforward tasks like adding numbers together, calculating totals, or combining values from multiple fields. This is useful anytime the answer can be determined from information the respondent has already provided.

To make this possible, fields can be labeled as variables. A variable is simply a way to reference a field elsewhere in the form. Once a field has a variable name, it can be used inside a calculated field to pull in its value.

You can learn more in this help document.

From Simple Calculations to Smarter Logic

Calculated fields don't need to be complex to be useful. Many forms rely on them for small conveniences, like showing a total before submission or calculating a score based on responses. These small touches can reduce errors and improve clarity for both respondents and the people working with the data later.

When needed, calculated fields also support a single line of JavaScript. This allows for more custom logic without turning the form into a full development project. Importantly, this is optional. You can get meaningful value from calculated fields without writing any code at all.

Form Security Options

Protecting Data Without Overcomplicating the Form

As forms begin to collect real information, questions of access, visibility, and sensitivity become more important. Not every form handles the same type of data, but every form benefits from clear decisions about how information should be protected once it's collected.

FormAssembly includes security options that allow you to manage these concerns without changing how a form looks or feels to the respondent. These options focus on how data is handled behind the scenes, rather than adding friction to the submission experience.

Some security settings control who can see or change information after it's submitted. Others determine how sensitive data is stored and whether it should be masked or restricted. Together, they support responsible data handling while keeping the form itself straightforward to complete.

These tools are designed to be applied deliberately. You can start with a simple form and add protections as requirements evolve, rather than needing to anticipate every scenario up front.

Controlling Visibility and Overwrites

Access Control settings let you control how individual fields behave after submission. Depending on your needs, a field can be visible, invisible, or set so it cannot be overwritten. This is especially useful when a form is used to review or update existing information, and you want to protect certain values from being changed.

Access Control helps ensure that data stays accurate as it moves through different steps or hands.

You can learn more in this help document.

Handling Sensitive Information

Some data requires extra care. FormAssembly allows sensitive data to be marked so it is handled appropriately. All customers are required to mark credit card fields as sensitive, and customers on certain plans can mark additional data types as sensitive as well.

This ensures that sensitive information is protected throughout the submission process and not exposed where it shouldn't be.

Preventing Spam and Restricting Access

Publicly available forms can sometimes attract unwanted submissions. Adding reCAPTCHA helps prevent automated spam without adding friction for real respondents.

For forms that should only be accessible to specific people, some plans support respondent authentication. This allows you to hide a form behind a login, ensuring that only authorized users can access and submit it.

You can learn more in this help document.

Security Matters

Security options help you collect data responsibly while maintaining a smooth experience for respondents. They allow you to balance accessibility with protection and ensure that your forms can be used confidently in real-world scenarios.

You don't need to use every security option at once. Many forms start simple and add protections as requirements evolve.

After Submission Options

After Submission Options

Submitting a form is a natural transition point. At that moment, the information has been collected, structured, and validated, and you can decide what should happen with it next.

After-submission options give you control over how that moment is handled. You can acknowledge the respondent, notify others, generate records or documents, or simply let the submission stand on its own. Nothing here is required, and many forms work perfectly well without any additional steps.

These options exist to support different workflows and expectations. Sometimes a submission needs confirmation. Sometimes it needs to trigger awareness or create a record. Other times, it just needs to be stored. FormAssembly allows you to choose what makes sense for your use case without forcing a particular pattern.

Communicating With Respondents and Teams

If your form collects an email address, you can automatically send a message when the form is submitted. This might be a confirmation for the respondent, a receipt, or a simple acknowledgment that their information was received.

Emails can also be sent to internal recipients. This is useful when submissions should trigger awareness or action from a team without requiring someone to constantly monitor responses.

These messages help close the loop and make the submission feel intentional rather than invisible.

You can learn more in this help document.

Designing the End of the Form

By default, forms display a Thank You page after submission. This page can be customized using rich text or HTML, which allows you to explain next steps, share additional resources, or simply confirm that everything worked as expected.

If a Thank You page isn't the right fit, you can redirect respondents to another webpage instead. Redirects are useful when the form is part of a larger flow, such as sending someone back to a portal, a confirmation page, or a follow-up experience.

You can learn more in this help document.

Capturing Signatures and Generating Documents

Forms can also be used to collect a single electronic signature from a respondent. This is helpful when an agreement, acknowledgment, or approval is required as part of the submission.

You can learn more in this help document.

In addition, form-level PDFs can be generated automatically when a form is submitted. These PDFs can capture the submitted data in a structured format and be sent via email. This is often used for records, confirmations, or internal documentation.

You can learn more in this help document.

Why After-Submission Options Matter

After a form is submitted, there is a natural pause. The respondent has done their part, and the system now holds information that may need to be acknowledged, recorded, or acted on. After-submission options exist to shape how that moment is handled.

For respondents, these options provide clarity. A confirmation email, a customized Thank You page, or a redirect helps set expectations about what happens next. Even when no follow-up action is required, clear acknowledgment reassures people that their submission was received and understood.

For teams and systems, after-submission options help establish continuity. Notifications can surface new submissions without requiring constant monitoring. Generated PDFs can create a record of what was submitted at a specific point in time. Electronic signatures can formalize agreements or approvals as part of the same interaction, rather than as a separate step.

These options also allow forms to fit more naturally into existing workflows. Some submissions need to trigger action immediately. Others simply need to be stored, documented, or reviewed later. By choosing how submissions are handled, you align the form with how work already happens, instead of forcing a new process around it.

Over time, after-submission behavior becomes part of how people experience your forms. Submissions feel complete. Expectations are clearer. Follow-up feels intentional rather than improvised. The form becomes part of a complete submission experience, rather than ending at collection.

Working with Responses

Seeing and Managing What Your Form Collects

Once your form is live, submissions are collected on the Responses page for that form. This is where you can review what's been submitted, search for specific entries, and understand how your data is shaping up over time.

Responses aren't static. They're meant to be reviewed, adjusted, and reused as needed.

You can learn more in this help document.

Finding and Reviewing Submissions

Responses can be searched, which makes it easier to locate specific entries without scrolling through everything manually. You can also choose which fields appear in the response report by adjusting the visible columns. This allows you to focus on the information that matters most for your use case.

This flexibility is especially helpful as forms grow or when they collect more data than you need to see at once.

Editing and Reopening Responses

In some situations, a response may need to be corrected or completed. Responses can be reopened and sent

back to the respondent for editing, which avoids the need for resubmission or duplicate entries.

As the form owner, you can also edit responses directly. This gives you a way to make small corrections or updates while keeping all related data tied to a single submission.

Exporting Data for Other Uses

Response data can be exported in several formats, including CSV, HTML, XML, and PDF. Exports make it easy to share data, analyze it elsewhere, or archive submissions for record-keeping.

Exporting is often a bridge between form data and other systems or workflows, especially when integrations aren't part of the initial setup.

Connectors and Integrations

Placing Form Data Where It Belongs

Connectors and integrations describe how form data can be used beyond the form itself. They define where information goes, when it moves, and how it becomes part of a broader system or workflow.

Not every form needs an integration, and nothing in this section is required to get value from FormAssembly. Instead, this section is about building an understanding of what connectors make possible, so you can recognize when a form needs to connect to something else and why.

As you explore connectors, it can be helpful to think less about configuration and more about intent. Questions like "Where should this data live?" or "What needs to happen after a submission?" often clarify whether an integration is needed and which kind of connector fits best.

When Connectors Run

Connectors don't all run at the same moment. Where a connector sits on the connector timeline determines when it runs and what information is available at that point.

A helpful way to think about the timeline is as a series of moments in the life of a form submission. Each moment offers slightly different options.

Some connectors run when a form opens. These are typically used to prepare data before a respondent starts filling anything out. At this point, there is no submission yet, so connectors here are usually focused on looking things up or pre-filling information.

Other connectors run when a form is saved. This moment happens when the respondent saves the form, but before the submission is finalized. Connectors placed here can work with the data as it's being saved and allow data collection to occur even before the respondent submits the form.

Connectors can also run as a form is submitted or after submission is complete. These positions are commonly used when the goal is to take action with the final data, such as creating or updating records, sending information to another system, or triggering follow-up processes.

The key idea is that timeline placement isn't about right or wrong. It's about intent. Asking "What should happen before the form is finished?" leads to a different placement than asking "What should happen once the submission is complete?"

Understanding this distinction early makes connectors easier to reason about and reduces the guesswork when something doesn't behave as expected.

You can learn more in this help document.

Why This Matters More Than It Seems

Timeline placement affects both behavior and troubleshooting. If a connector runs earlier than expected, it may not have access to all the data you assume it does. If it runs later, it may not be able to influence the submission itself.

PRO TIP: When something doesn't work, the timeline is often the first thing to check. Knowing when a connector runs gives you a clearer starting point for understanding why it behaved the way it did.

Connecting to Other Systems

FormAssembly includes connectors for a variety of systems. Payment connectors allow you to collect payments from respondents while ensuring that credit card details are masked and never stored after submission.

Other integrations allow form data to be sent to external systems or services so it can be acted on immediately, without manual intervention.

Understanding the Salesforce Connector

The Salesforce Connector is one of the most commonly used integrations with FormAssembly. It authenticates using a Salesforce login, and that login determines what the connector can see and do. In other words, the authenticated user acts as the "window" through which the connector views Salesforce.

If the authenticated user can access an object or record in Salesforce, the connector can access it too, as long as it's available through the Salesforce API. If the user cannot access it, the connector cannot either. This model helps keep data access aligned with existing Salesforce permissions.

This perspective also helps explain certain connector behaviors. For example, Salesforce duplicate rules may prevent a record from being created and return an error instead. In these cases, the connector respects Salesforce's rules for protecting data quality rather than failing unexpectedly.

You can learn more in this help document.

Why Integrations Matter

Connectors allow forms to participate in real-world workflows. Instead of collecting data and manually moving it

elsewhere, your form can create, update, or look up records and pass information along automatically to where you need that data to go.

You don't need to use every integration to benefit from this capability. Even understanding that forms can act as entry points into other systems can shape how you design them.

At this stage, forms stop being isolated tools and start becoming part of a connected ecosystem.

Publishing

Making Your Form Available to Respondents

Once your form is ready, it's time to make it available for your audience to fill out. Publishing controls where and how respondents access your form.

By default, every form has a FormAssembly-hosted link. This link can be shared directly and is often the fastest way to make a form live. It's especially useful for testing, internal forms, or when you need a quick way to start collecting responses.

You can learn more in this help document.

Embedding Forms Into Other Experiences

Forms don't have to live on their own page. They can be embedded into websites using HTML, JavaScript Quick Publish, or an iframe. Embedding allows your form to feel like a natural part of an existing site or portal rather than a separate destination.

Different embedding methods offer different levels of flexibility, but they all serve the same purpose: placing the form where your audience already is. You can start with the simplest option and move to more advanced approaches later if needed.

Platform-Specific Publishing Options

Forms can also be published to platforms like WordPress, which can simplify sharing forms on content-driven sites. For more specialized use cases, advanced publishing options such as the REST API are available.

These options aren't required to get started, but they exist to support more complex distribution needs as they arise.

Why Publishing Matters

Publishing is the bridge between building and collecting. A well-designed form only creates value once people can use it.

By understanding the different ways forms can be shared, you can choose the option that fits your situation now and know where to look if your needs change later.

Where to Go Next

At this point, you've seen how forms are built, how they can guide respondents, how data can be protected and shaped, and how submissions can trigger action or move into other systems.

You don't need to use every feature covered here to be successful. Many effective forms rely on only a handful of these ideas. As you build and iterate, you'll naturally discover which features are most helpful for your use cases.

When you're ready to go deeper, our help documentation is there to support more detailed questions and specific configurations. Certification is also available as a future step once you feel comfortable with the basics and want to validate or expand your knowledge.

Most importantly, you now have enough understanding to start building, testing, and improving with confidence!