OpenReplay

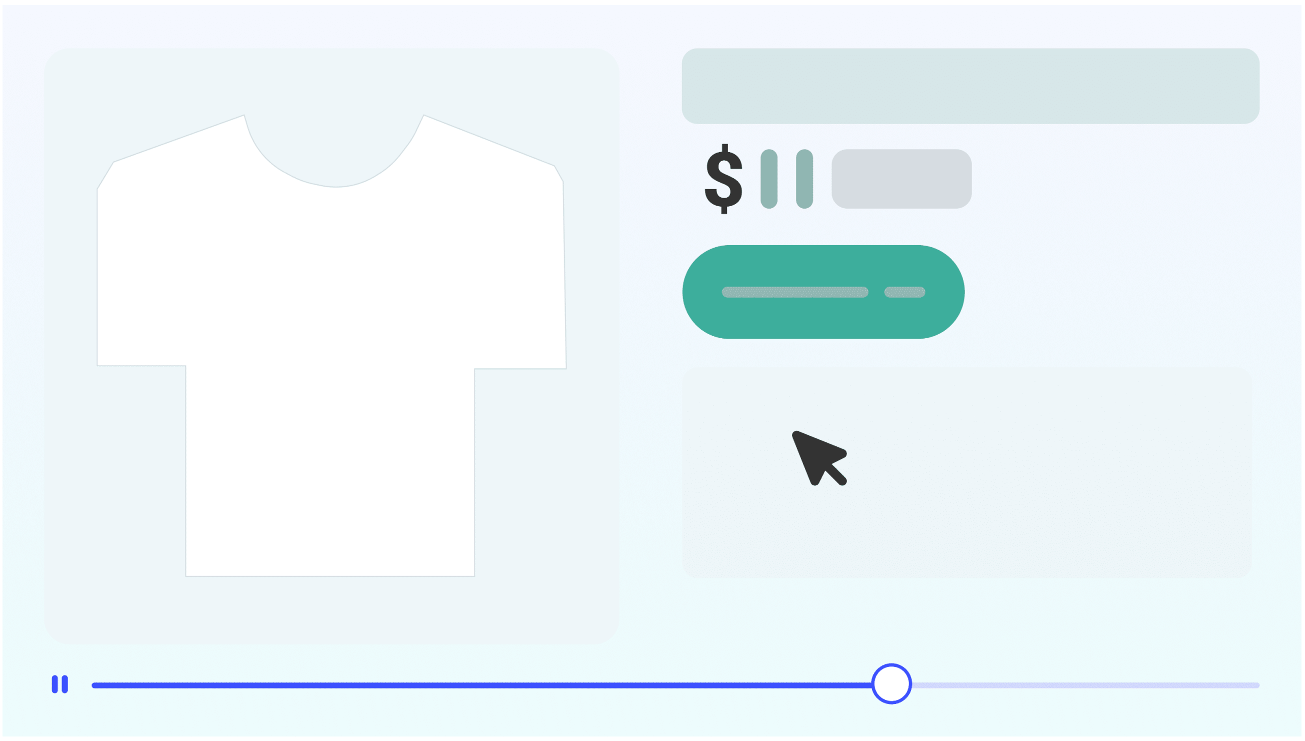
E-BOOK

The complete Session Replay Guide

This comprehensive guide tells how large corporations utilize session replays as a final stage of analysis, leading to improved business outcomes and enhanced customer experiences for their audience.

Table of contents

What is Session Replay?



Imagine sitting next to your users while they're browsing your website. That would be awesome, wouldn't it? You'd get to experience first-hand what kind of issues they run into, and you'd see the parts of your application they don't get and feel frustrated about. Imagine now being able to move them aside when those problems happen and open up the DevTools on their browser to make notes about what went wrong.

That would be incredible, but this is nothing more than a dream because you can't do that with all your users. Granted, you can have focus groups or ask some favors here and there to get first-hand feedback, but those results would only represent a tiny fraction of your users.

What can you do instead?

Session replay allows you to watch how all of your users navigate your web application and record everything you care about so that you have all the context and details you can possibly need if anything goes wrong.

What alternatives are there to session replay?

Several alternatives out there can give you some of the benefits session replay offers. One could group them into two categories: monitoring tools and product analytics tools.

Monitoring tools

This category is filled with solutions targeting different technical aspects of your application. For example, application performance tools like New Relic or Dynatrace are great for tracking slow code in your backend or poorly performing transactions with your database. However, that only shows you one side of the story: the backend, but you still need the front-end state.

You can, of course, go all in and use a logging tool like Elastic or Splunk and capture every piece of log you can get your hands on. The downside would be that you'll be drowning in a sea of data, and making sense of issues may not be easy. Instead, you could focus on errors only and rely on an error-tracking tool like Sentry. Still, you'll fall short while trying to reproduce bugs as you'll never get hold of the entire context, like the actions from the users, the network activity, any debug logs left by the developers and even the app state at a given point in time.

You can also add a Real User Monitoring tool, which would let you track other aspects, such as application performance, loading times (including web vital metrics), failing requests, and so on. But you'd still be missing the context that comes from watching a session replay. The perfect complement for RUM is session replay, because it provides information that can only be obtained visually:

- Did our site design confuse your client into using the wrong feature?
- Was the user entering too much data in that field and that’s why the app crashed?
- Was the error message visible to the user long enough for them to read it?

Those are questions you can't answer with RUM tools and that's one of the main benefits of using session replay. These monitoring tools are great, but they only tell you one side of the story because of their limited view. With session replay, you can see exactly what happened and what went wrong as if you were there, and this visual inspection gives you extra insights into how to solve any present issues.

Product Analytics

This category is filled with solutions targeting different technical aspects of your application. For example, application performance tools like New Relic or Dynatrace are great for tracking slow code in your backend or poorly performing transactions with your database. However, that only shows you one side of the story: the backend, but you still need the front-end state.

How do they compare with Session replay tools?

None of these tools captures the whole picture; if you use them individually, you're missing out on a lot of extra context.

However, adding session replay into the mix can help you bridge the gap between raw numbers and insights.

You won't have to guess why the drop percentage of your main funnel is so significant, or why the new feature you released last week has not been used by your users yet.

With session replay, you'll see precisely what your users are doing and how your site reacts to each of their interactions along the way.

Should I add session replay to my stack?

If you're wondering whether or not to go for session replay if you already have one of the above tools, the answer is most certainly "yes".

Session replay complements whatever observability or analytics stack you may already have, and removes existing blind spots in your digital experience by turning metrics into insights.

It’s like looking over your users’ shoulders when something wrong happens.

The real questions should be*: How can you not have it?*

Problems solved by Session Replay

Session replay tools can cater to many different areas within the same company, making picking the right alternative even harder.

- Want to check out how users are interacting with the new design update? Session replay has got you covered!
- Trying to figure out why the app keeps crashing? Session replay can give you the answer to that question.
- Curious about how users are reacting to the new features? You guessed it, session replay can help you out!

Let's look at some of the most common use cases where session replay tools can shine like no other.

Reproduce and fix issues

Issue reproduction is one of the most powerful use cases for session replay as it helps developers get to the root cause of a wide variety of problems.

Developers cannot rely on users to create bug reports when they face a problem in their app. And they're not to blame; after all, users have no clue — and are not supposed to — about your technical stack, or how to raise a well-documented bug. This means you end up with reports that look like this:

"The application stops working after a while."

What's the alternative at this point? How can you get the information you need without making the user uncomfortable?

Begging for screenshots, debuging logs or a even asking for a screen recording are all options. And yes, you can ask your power users for favors and even jump on a zoom call to save yourself some trouble, but you cannot do it all the time, and certainly not with all users.

Why waste time going back and forth with users when you can watch the recorded session instead? With session replay, you can easily fast forward and rewind to the exact spot where the error, frustration, or crash occurred. Plus, with access to developer tools inside the player, you'll have all the information you need right at your fingertips, like errors, network requests, and performance metrics. This means you can spend less time trying to reproduce the problem and more time fixing it.

What used to take hours now takes minutes.

Track your app's performance

Keeping track of your website's performance is something some session replay tools do and do very well. When recording what users are doing on your website, session replay can also track and measure resource utilization, loading times, and other web vitals.

With metrics like CPU/memory consumption, speed index, image load times or page response distribution, any performance hiccup or regression caused by an update in your code becomes trivial to find. You can even set alerts on those metrics to roll back code changes if things go wrong after a new release.

Understand and alleviate user frustrations

Session replay tools can surface insights about features or sections of your site that frustrate users. In other words, when your app doesn't behave the way users expect it to, they get annoyed or, worse, stop using it.

Buttons that don't work, users rage-clicking and other similar frictions can be quickly found by session replay tools. It's as if you were there, looking over your user's shoulder. With these valuable insights, you can dig further and inspect those replays to get to the root cause and fix your user experience in record time.

Engage with your customers when they need support

A handful of session replay tools allow you to engage your customers — in *real-time* — while they're having problems.

For example, imagine one of your users is having trouble doing the checkout on your ecommerce site and they reach out to your support chat for help. With a live session replay, you can jump into the user's active session and troubleshoot the problem in real-time. This means you can offer a solution in minutes, instead of having them wait for hours or even days for the issue to be resolved. It's like having a virtual helper right there with the user, getting them back on track in no time.

This is a game changer for support teams. It brings down their mean time to resolution (also known as MTTR), helps reduce user frustration and avoid attrition. A happy user is someone who converts into a paying customer sooner or later.

Improve your product

Take everything mentioned so far, and you'll have a powerful tool to help you unlock insights about your product.

With a session replay tool, it’s no longer just about the numbers, you can feel the user’s frustrations and see exactly what’s causing them.

With this new insight you can create a product plan and deliver exactly what your users need.

Some session replay tools even offer analytics capabilities to aggregate data and keep an eye on what matters the most to the business, ranging from increasing conversions to improving your application’s digital experience.

Who is Session Replay Useful For?

Session replay tools are so versatile that they provide value to many different teams within organizations.

In fact, this is also the reason why there are so many different session replay alternatives in the market. They each focus on a different team and on their individual interests. There is hardly any tool that encompasses all of them together.

So make sure to have your primary user and their priorities in mind when picking your next session replay tool. Otherwise they might end up missing critical features.

Developers

Developers will benefit from session replay tools that track issues and their context.

Part of the day-to-day of developers includes debuging problems. Sometimes that process requires them to contact users and ask for details. Either that or somehow find a way to reproduce the problem themselves using the bug report provided.

What would your dev team think if instead of a bug report, they received the exact reproduction steps? And what if on top of that, you threw in a recording of the user's actions and the response from your system?

They could now work on fixing the problem as if they had the user sitting next to them, walking them through what they did and what happened afterwards.

The ideal session replay tool for developers should record the following:

- The content of the DevTool's console. Effectively recording every JavaScript error and any debug or info messages left by the developers. This also includes more app-specific information like state mutations, web vital metrics (like loading time, time to first byte, etc) and even some performance metrics (like CPU and memory utilization).
- The content of the Network tab. Where devs can see the different interactions with outside resources like API request and the associated payloads.

A tool like this has the potential to reduce debugging time significantly and improve the developer's experience while fixing problems.

Support Agents

The main goal of any support agent is to help users as fast as possible. Time To Resolution is a metric they care about very much. However, they can't help them if they don't have the right tools for the job.

The ability to contact a user while they're having problems and help them live, sounds too good to be true. However, some session replay tools have this capability. Instead of waiting to receive the user's complaints, support agents can watch active sessions and help users right there, on the spot.

They can jump on a video call with their users, co-browse the website with them and even add annotations to their screens to how exactly what they need to do.

These tools can also help bridge the gap between the support team and other teams inside big organizations. Other teams like product or development could interact with support speaking the same language (through session replays).

This makes session replay a must-have for any support agent. It gives them some very interesting abilities:

- A chance to help their users solve problems on the spot.
- A chance to give them a very personalized onboarding experience.

If you're considering adding session replay to your support stack, the answer should always be "yes".

Product Managers

With dashboards and funnels, product managers can understand what are the most impactful features of an app or where (*and why*) are customers leaking from their conversion funnels.

This is why session replay is perfect for them; the data coming from these tools powers those visualizations and in turn, the visualizations help PMs feel closer to their users. They're able to iterate a lot faster because they're now dealing with qualitative data (instead of cold numbers). Think of it like this: they can literally *see* their user's pain points and how they browse their product. That alone represents a goldmine of insights to product managers.

We can think of session replay tools as the perfect complement to analytic tools when it comes to Product Managers. By combining the two, they get the aggregated view of the product's current state, and they can also see the details of each interaction.

Individually each tool adds some value, but only when combined is that they truly shine.

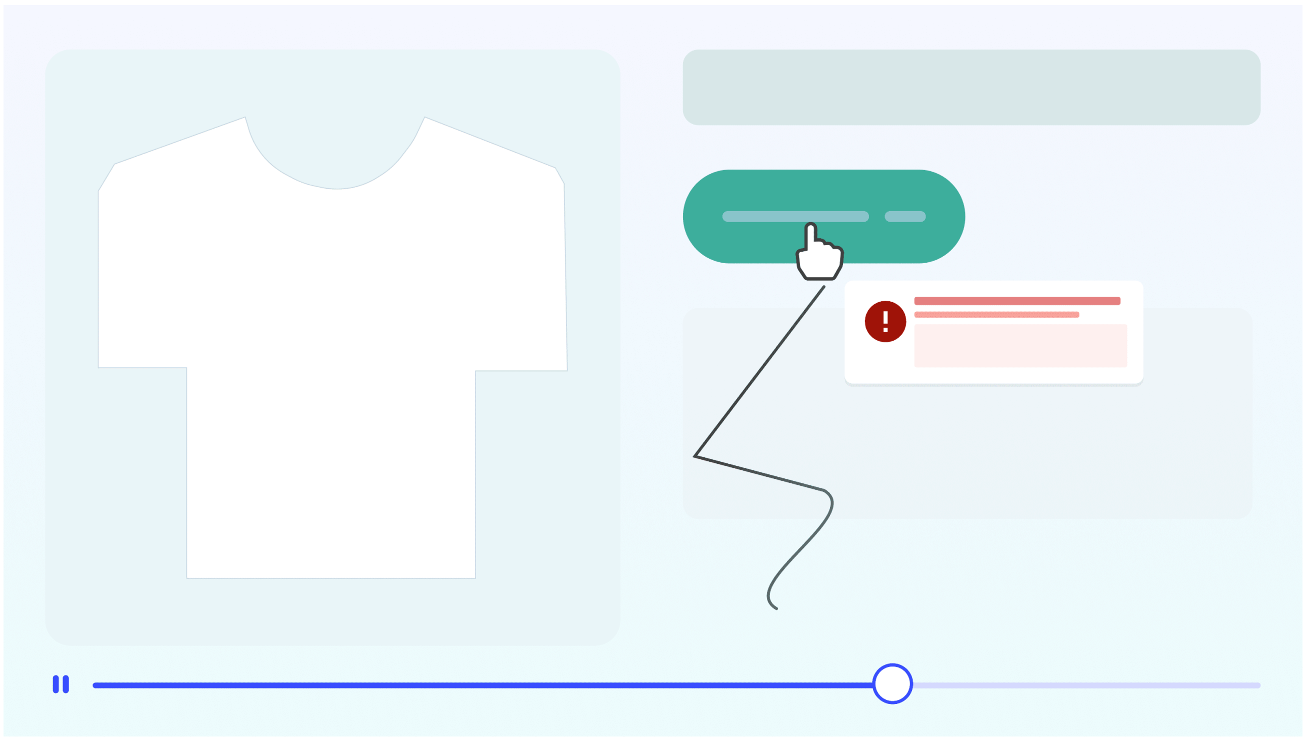
Designers and UX researchers

Designers get a lot of benefits from looking at how users interact with their designs, but conducting focus trials takes time and effort. And on top of that, results are not always trustworthy since user behavior can sometimes be affected by the fact that *they know* they're being observed; this is known as the Hawthorne Effect.

The data provided by session replays (like click-maps, mouse movement, etc) can give designers the chance to watch their users and iterate over different design choices to understand how their work look in production. After all, different devices, resolutions and even browser versions will affect the final look, which in turn will affect the end user's experience.

All of this done without the added costs or limited trust provided by in-person user tests.

How Does Session Replay Work?



Keeping tabs on everything that happens with your website while your users browse through it is not easy. There are a lot of moving parts, the user's actions, the code of your website, the browser being used, and more.

But the thing is, when you see a replay, everything clicks. You see why going through the trouble of creating such a product makes sense.

In this chapter we'll try to cover some of the aspects of what happens behind the curtain when you add the tracker code to your website. We might get a little technical in some parts, but don't worry, it'll all make sense.

What gets captured inside session replays?

Some of the data captured by the tracker will depend on the main focus area of the tool you're using. Remember, session replay is useful to many different areas within the enterprise, thus you have to know what you want to do with it before picking one.

That said, there are some standards that can benefit everyone, so let's take a look at those data points.

User behavior: we’re talking clicks, mouse movements, scrolling habits and more. Anything the user is doing, the tracker should capture it. This way you can see exactly how your app is being used. Are users scrolling over important messages? Or perhaps are they clicking on elements that are not clickable? This data is also post-processed to give you even more insights (keep reading).

- DOM mutation:
- Dev-Tools
- Web-vitals
- Frustrations
- Custom events
- Metadata: user identifications and more

For example, a session replay solution for developers will track a lot of information about JavaScript errors, such as:

- The stack trace
- Source code maps
- Information about requests sent during the session and more

There are, of course, many points in common between session replay tools, that information includes:

- **Mouse events:** this, of course, includes mouse clicks and the elements related to them (i.e., clicking on a button), but it also includes mouse movement. Yes, you'll be able to see how your user's mouse pointer moves during the session. So you can check how the user navigates through your site with the highest fidelity.
- **DOM mutations:** the tracker will record every change in your page's Document Object Model (the DOM). This includes changes to attributes such as classes, but it also means creating and destroying elements (like

a dialog box or a menu item). This behavior translates to replays showing exactly how a dynamic web application behaves.

- **Extra information:** some trackers will allow for the addition of plugins to their tracking process. Plugins enable you to capture extra information, such as changes in state data, and they also allow you to add custom processing to understand usage patterns, such as detecting rage-clicking. The possibilities here are endless as long as the tool allows this extensibility.

Some session replay solutions will add to that list the content of the **browser's DevTools**. Data points such as JavaScript errors, info messages, and request status codes will also be part of the recording of these tools.

How hard is it to add session replay to your site?

A good session replay tool will reproduce every single step and action the user takes.

To do that, the tracked website needs to add a **tracking code snippet**, like what Google Analytics needs. This code will then watch everything that happens within the website. Was there a click? It'll track it. Did the site show a modal window? It'll be there on the replay.

What happens to the recorded data?

Once the recording starts, the information goes to a centralized platform. There the data is processed and stored long-term. Other than saving event information and the HTML of the page, some session replay tools also save assets such as CSS stylesheets and images.

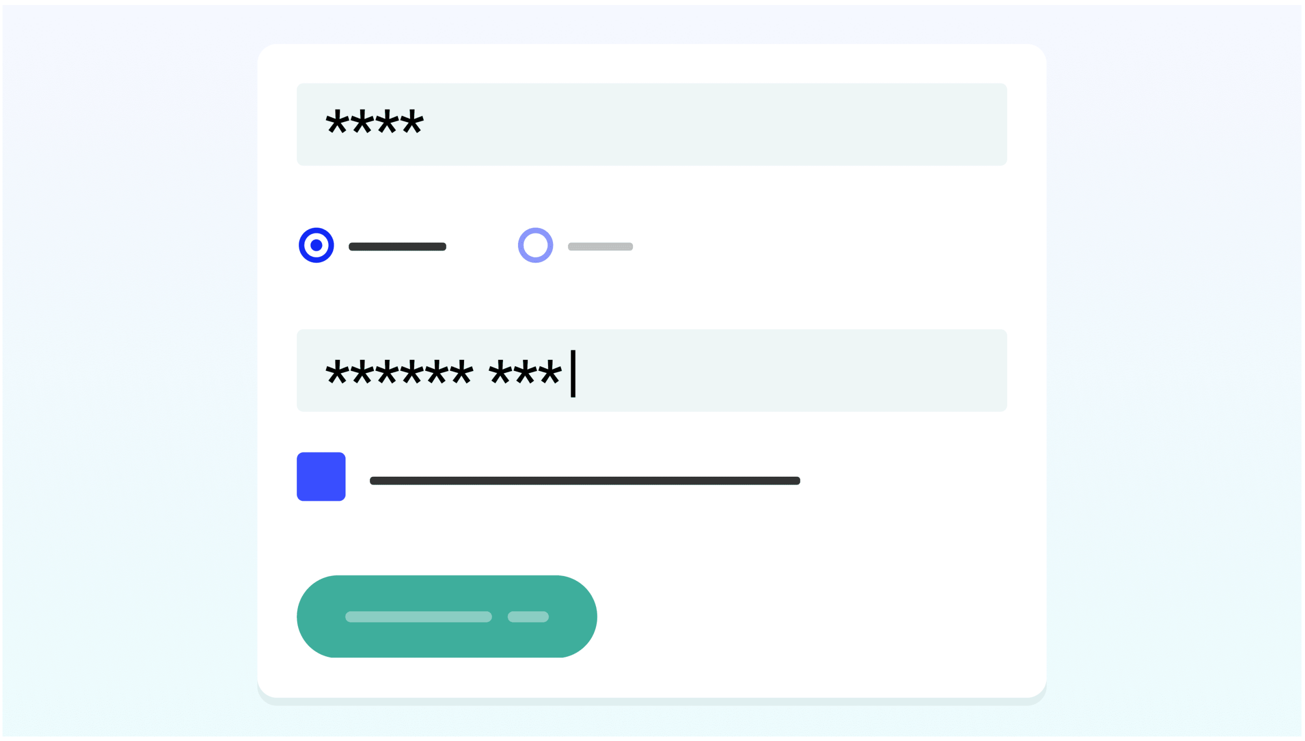
Why are assets also important for a session replay?

Imagine recording a session for your new site, and then two months later, removing the images from your homepage and re-doing the styles. If the assets weren't cached, the old HTML inside the recording would try to reference those images and classes, and it wouldn't find them.

The website in the replay would look broken. But that wouldn't be the reality, would it? Asset caching fixes that and ensures the fidelity of the replay, no matter how much time goes by.

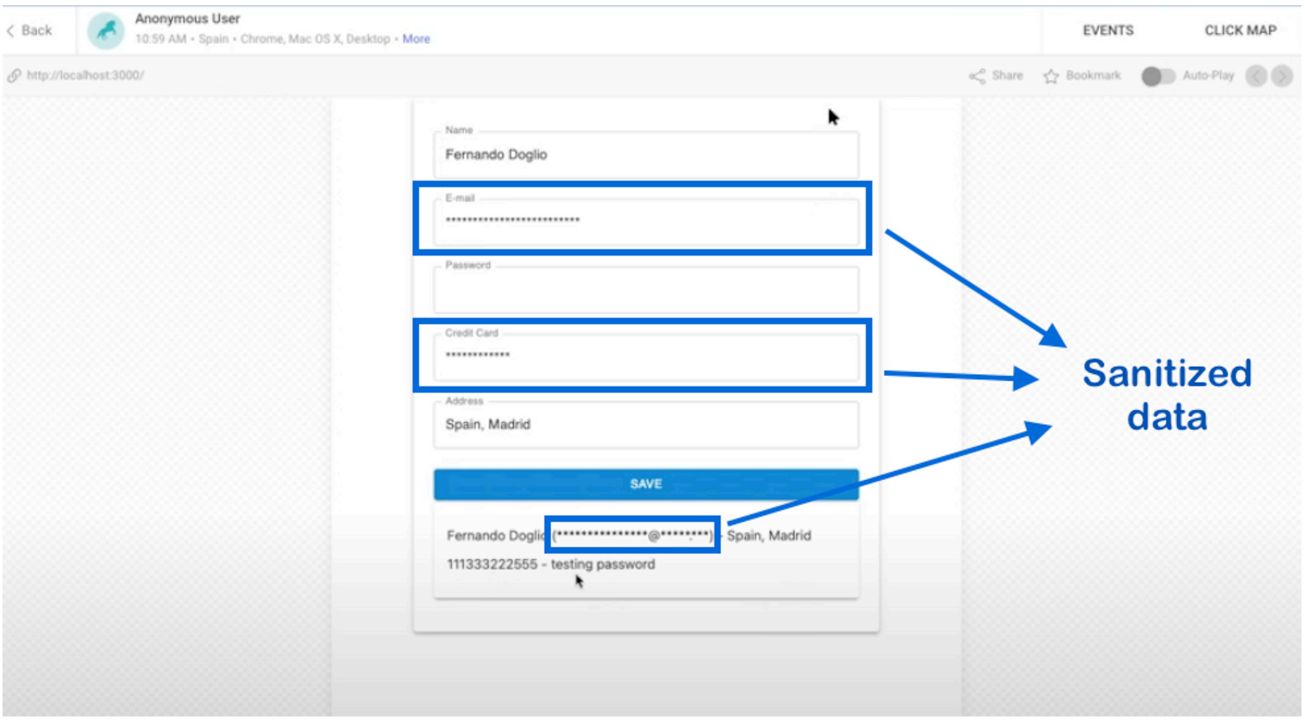
All major session replay tools will capture all you need to get a basic reproduction of your user's experience. What sets them apart from each other are the extra recording capabilities.

Importance of data privacy in Session Replay



Most session replay tools allow developers to control exactly what gets recorded and what gets ignored. Ignored information is not sent to the server; instead, it's either sanitized or not tracked.

Sanitized data gets replaced by asterisks showing that there is content without actually showing the content. Ignored data means the user's private information doesn't leave the browser as part of the session recording data.



Name: Fernando Doglio

E-mail: *************************

Password:

Credit Card: ************

Address: Spain, Madrid

Fernando Doglio (***************@*********) - Spain, Madrid
111333222555 - testing password

Data sanitization keeps vital information out of the session replay platform while, at the same time, continuing to add value.

Data Ownership

Answer the following question: *Who owns the data generated from recording **your users' actions** on **your website** when the tracking platform is **hosted in 3rd party-owned servers?***

Thinking about that compliance nightmare can give you a headache.

That's why some session replay solutions allow you to self-host the entire platform on your own servers.

That means you own the data, which in turn means you'll have an easier time complying with your internal security policies.

With such a solution, the answer is trivial: **you own the data.**

On the other hand, if you're using a 3rd-party-controlled solution, you might run into problems with your security compliance department. Chances are, they won't be too happy when you try to push your tracked app into production.

Compliance

If you have user-generated data, there are always internal and external security compliances to follow.

Internally imposed compliances can range from security restrictions around the infrastructure all the way to the software you can install.

When it comes to external compliances, the three most important ones are:

- GDPR You must follow this one if you're collecting data from EU citizens. Remember that GDPR allows users to remove their data from your database. It might not be obvious, but that also covers session replay data.

- HIPPA This one is specific for healthcare-related data. If you're collecting this information, you're likely already complying with it.

- CCPA This one is similar to GDPR but for citizens of California.

If you're not part of the healthcare industry, and your app is not localized for a specific country, chances are you're having to comply with GDPR and CCPA.

But how can you be sure that you comply with all the above regulations when adding a session replay solution to your application?

How do session replay tools solve this problem?

Session replay tools solve data privacy concerns through data sanitization. Developers can tell the tracker which data to track and which to ignore.

For example, let's say you're working on your app's profile page. The development team working on it could mark the password, address, and phone number fields as "private" for the tracker.

With that, anytime the user is looking at their profile or even updating its content, the tracker would sanitize that information. Resulting in a secured replay that still shows what the user is doing without revealing personal information.

Because of this, it's always important to review and understand how session replay tools can deal with and follow the imposed regulations.

The two most important points to consider about data privacy and session replay are:

1. Make sure the tool you've chosen allows for extensive data sanitization.
2. Make sure the selected tool provides the required functionality to remove user-generated data.

Session Replay and Pricing

Regarding the pricing model of session replay tools, there are mainly two trends, and the main difference between them is how they scale over time for growing companies.

Volume-based pricing

This pricing model presents different "gates" based on the number of sessions recorded. The higher the amount of the gate, the higher the price.

This is a good pricing model for small websites that don't get much traffic because the smaller gates tend to be quite affordable, and companies can get a lot of value for the lower price.

That said, once the website starts to grow and the traffic increases, so will the session replay fee. A way to mitigate this is to throttle the amount sessions being recorded.

Some trackers will let you configure the percentage of sessions to record to keep the expenses under control.

The downside to this technique is that you might miss critical errors from specific sessions that got ignored due to the throttling.

A much better option is to go with a value-based pricing model when available.

Value-based pricing

This model works based on the value added to the company through the session replays. The way to measure that value is through the number of users that need access to the platform.

A company that is only trying and looking to make a small initial investment in this technology can spend the minimum fixed amount of money to accommodate a few users regardless of whether they have 10 or a 1M visits monthly.

Once the value is appreciated, the model can grow slowly by adding new users, regardless of the application's progress.

This model is great for big companies looking to start using session replay or for those looking to reduce their investment in this technology without losing any of its benefits.

The downside is that the pricing model can be too expensive for smaller companies and startups, even at its lower tier.

That is when the free tier comes into play. Usually, these models will also have a limited free tier where the company can try the product for a limited amount of time to understand its value.

If, on top of that, the tool also offers a self-hosted version that can be completely installed and administered by the client, then the costs are significantly reduced (or at least replaced by infrastructure costs, which tend to be smaller).

What to Look in a Session Replay Tool

Because of the versatility of session replay and how it can provide value to so many different areas of the company, picking the right session replay alternative for your particular business case can be a daunting task.

In this section, we'll list all major points to take into account when picking one where your development team is your primary focus.

Implementation

Or in other words, getting the answer to these two questions:

- How easy is it to add the tracker to your site?
- How little is your site affected by this change?

Ideally, a session replay tracker should not require more than a few lines of code to get started, no matter your tech stack.

Once the basic setup is ready, adding more information and interwinding your custom business logic into the tracking capabilities of the tool is an entirely different process.

An ideal tracker should be easy to set up with minimum code, but at the same time, it should provide you with the tools to customize everything you need (i.e., capture custom events, sanitize the data you want, add extra information to the captured session, etc.).

All of this while keeping the performance impact down to (or as close as possible to) zero.

Powerful DevTools

If you want this tool to be beneficial to your front-end developers, then you have to present them with an environment that is very similar to what they already know and use daily.

Every major browser provides some variation of DevTools, and they follow a similar standard to split and showcase the information inside them.

An ideal session replay tool should do the same. It should be able to copy the pattern and provide the same (or very similar) user experience that developers are accustomed to. This is mainly because one of the primary use cases for these tools is the time reduction for identifying and finding bugs. And in that scenario, a familiar UX is crucial to a quick and successful adoption of the tool.

Self-hosting

The session replay alternatives that provide a self-hosting option will let you install their entire platform (the back-end that processes the replay data) into your servers.

This gives you two major advantages:

1. You have **full control over the data** generated by your users. Since the tracker will send the replay data directly to your servers, there is no room for 3rd party companies to use that data for their own interest.

2. **Complying with internal security policies** is much easier this way. When you own the data and the servers where that data is stored, you can safely configure and secure them as much as needed.

In the end, self-hosting is the solution most companies go with because of the control they gain over the data.

Make sure to look for this option when going through a list of session replay alternatives if security and data ownership are important concerns for your company.

Extensibility

Session replay trackers work under the premise that the DOM works in the same way for all applications and under all browsers. However, your particular application, the one you're tracking, does have custom logic that does not have to follow known standards.

This is why many trackers will fall short when you start looking to build complex tracking reports.

Consider the scenario where you develop a brand new app using your own state management library. Yet, you'd like to give developers using the replay tool, the ability to see how the state changes througout the user session. There is no way for a platform built by someone else to understand how your library works.

However, an extensible session replay solution can help in that regard, by giving developers the tools to extend its functionality themselves. That way they can add the support they need without having to wait for official releases that might never come.

An ideal session replay solution gives you this option, either by providing an open-source core that you can extend or through a plugin-based architecture, where you can add your code and logic following the platform's standards.

Either option makes that session replay solution future-proof because it can grow with the industry standards and at the pace each user needs them to.

If extensibility and customization are two factors that matter to you, then consider looking for open-source session replay solutions when you're evaluating alternatives.

Search capabilities

While it might sound like a trivial feature, searching is key to ensuring you find the replay you want to see when you need to see it.

Think about how much traffic your website is getting. If you're capturing 100% of it, you'll have the same number of sessions daily. How will you find the one that shows the problems your application is having? Or how will you review the session of a user who sent you an email complaining about a feature not working?

The search capabilities of the ideal session replay tool need to be powerful and versatile. They need to let you filter by aspects that matter to you.

And on top of that, if you have a way to identify users or categories of users (like free plan vs. premium plan), you should also be able to filter by custom fields, giving the tool the flexibility you need. The point of this feature is to let you find the right replay as soon as possible, so you can spend most of your time reviewing it, instead of looking for it.

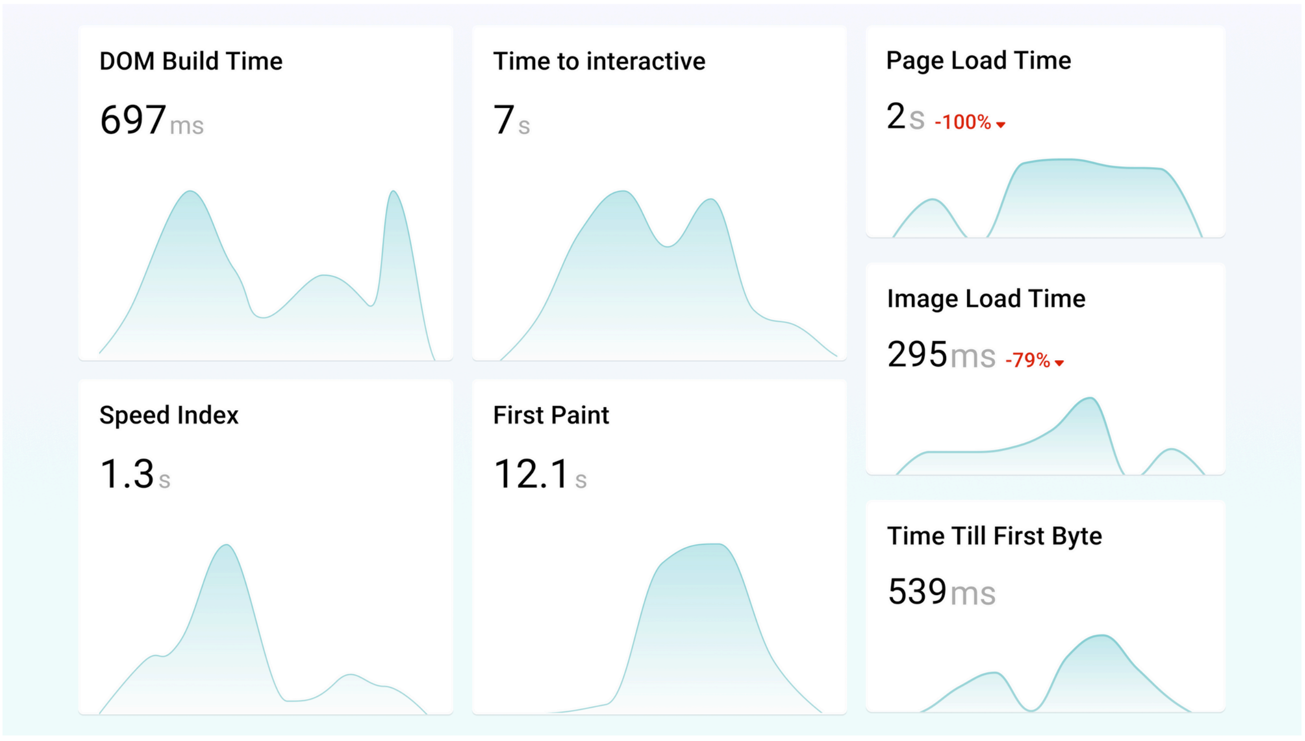
Analytics

With the number of data points the tracker can record, it should be expected that a good session replay solution presents a set of analytics to show users the status of their website.

The number of metrics to display and the focus of these metrics is completely up to the solution, but an ideal session replay tool would provide the flexibility that comes with the power of having users build their dashboards with the set of metrics they want.

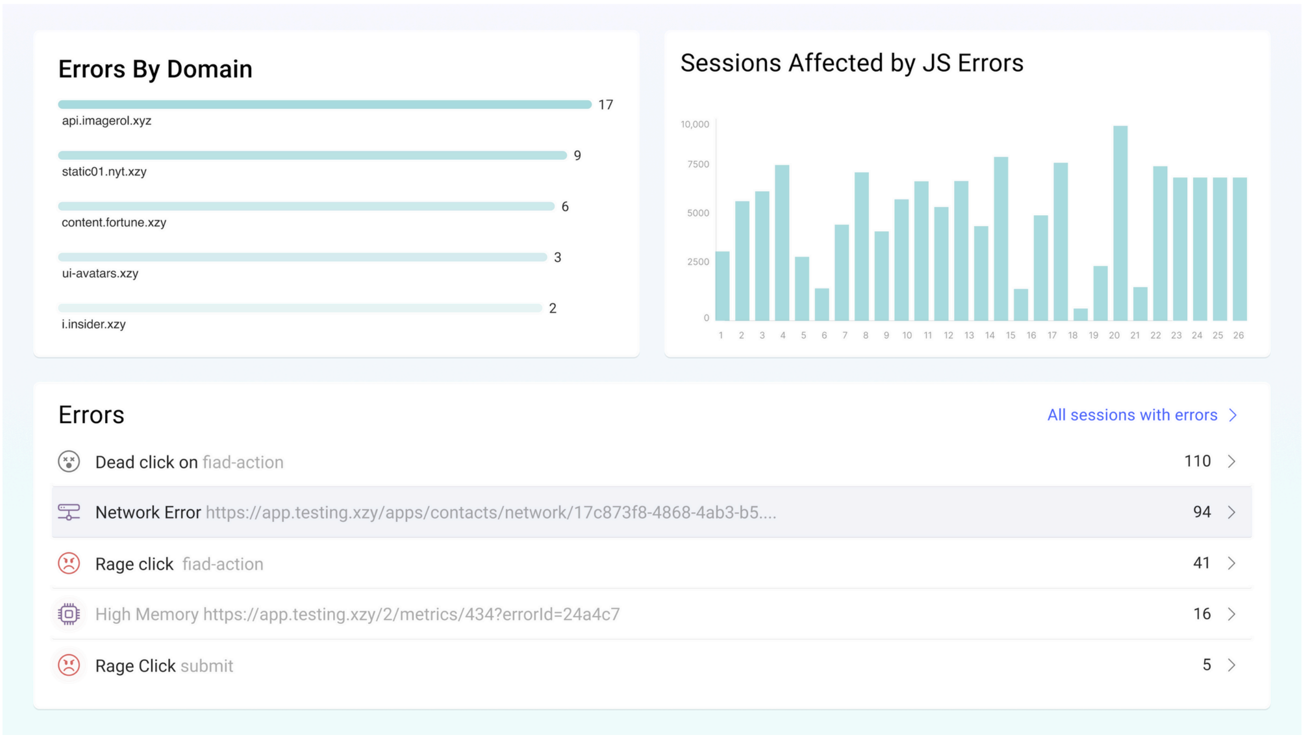
That way, a single solution can show different angles of your application/ website by focusing on different metrics.

For example, the following dashboard shows the performance of the homepage of a website, including loading times, slowest resources and some other web vital metrics:



While the following dashboard, for the same application, shows the health of the checkout page. The list of JavaScript errors, together with the type of errors by category and the list of failed requests.

(Picture in next page)



Errors | All sessions with errors >

Dead click on fiad-action	110
Network Error https://app.testing.xzy/apps/contacts/network/17c873f8-4868-4ab3-b5....	94
Rage click fiad-action	41
High Memory https://app.testing.xzy/2/metrics/434?errorId=24a4c7	16
Rage Click submit	5

This, in turn, allows different team members to focus on their tasks and on the most useful insights for their role.

Pricing scalability

As we already covered, there are two major pricing models for session replay solutions, and the important aspect to analyze when choosing one is how scalable it is for your particular use case.

Are you planning on testing an application that will grow to a hundred thousand sessions per month (if not more)? Then pick a value-based model.

Are you planning on testing a small website that will not surpass the tens of thousands of monthly sessions? Then, in that case you'll find some volume-based solutions out there that are cheaper than value-based alternatives. Or look for an open-source solution to self-host and reduce costs without compromising on the product.

Integrations

Just like with plugins and the ability to extend the code and main functionalities of a session replay tool to fit your particular needs, the capacity to integrate with existing tools is also a trademark of a powerful session replay solution.

The reason for this is that session replay tools usually show you only one side of the story; no matter how powerful and feature-rich they might be, they're limited to the front-end of your application.

But what happens if problems in the back-end trigger the errors? Integrating with other tools that monitor your application's infrastructure is a very good place to start.

Another interesting and useful integration for these tools is ticket tracking systems. If you can create a ticket from a session replay showcasing a problem with your application, then you can improve the workflow of your QA people significantly.

Your imagination is the limit here, every use case can benefit from different integrations. Just make sure you have options available when picking your replay tool.

Breadth of features

While it is essential to ensure the solution you pick has all the features you need, it's also good to keep other roles and departments in mind that might benefit from the secondary features a session replay solution might have.

As we've already stated, these tools can provide insight into many different areas. While they usually have one focus, if the tool also provides complementary features that others might benefit from, then it's a great idea to take advantage of them.

Community

Finally, when deciding which session replay is the best for you, take a look at the community behind your choices to understand how other users feel about the product.

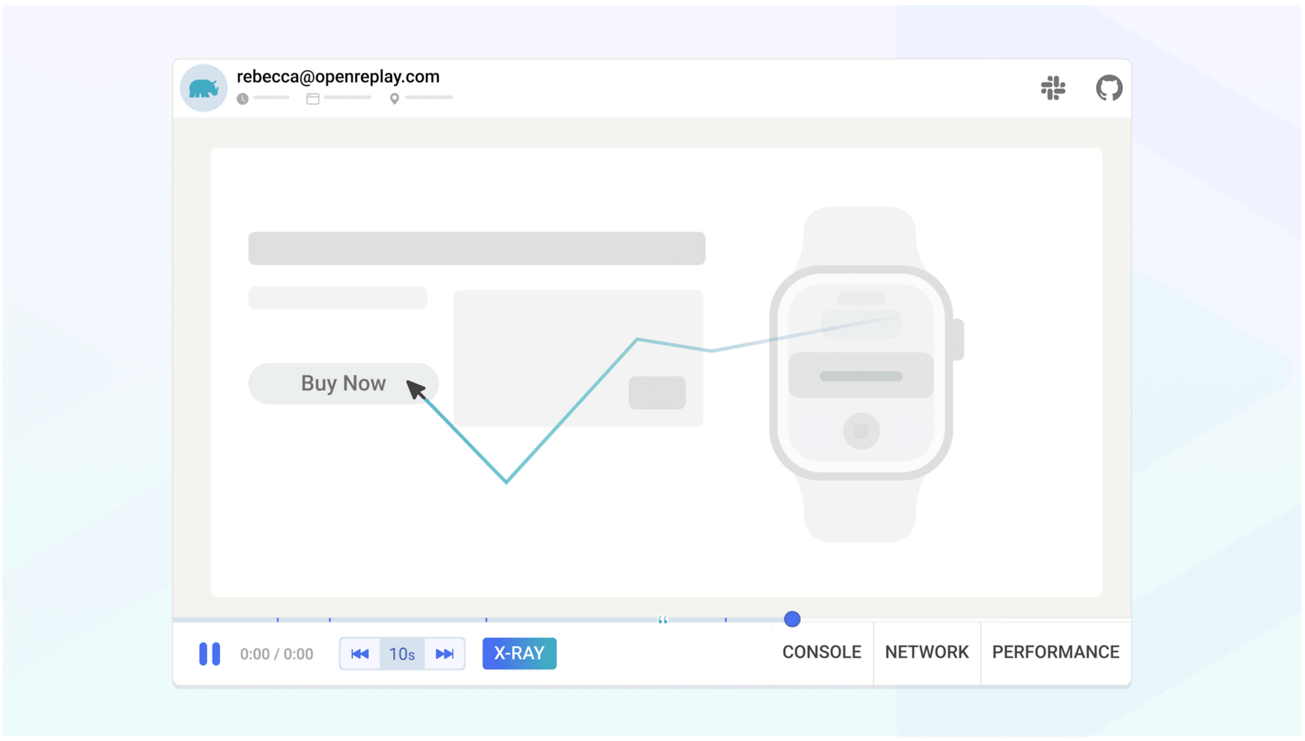
An active community where developers are engaged and potentially even contributing their time and effort into improving it is the sign of a successful product with a great and responsive team behind it.

The way in which you can assess the size of a community from the outside is to look at places like:

- Does the product/company have a public communication channel with its users? Either through Github Discussions, Slack, Discord or some other solution.
- If the solution is open-source, does it have an active list of public issues? This indicates that people are using it and care about it enough to want it to improve.
- Is the company active and engaged publicly with developer communities? This can be anything from Twitter to specific Subreddits. Anything where the development community resides, these companies should be there to understand what users expect from them and their products.

OpenReplay

Open-source Session Replay you can self-host



Session replay solutions have been historically focused on areas such as UX, Product development and even Marketing, but rarely can you see a solution focused on the development team's needs.

OpenReplay is one of the first of its kind because it is mainly aimed at front-end developers looking to reduce the time they spend debugging their applications. It also offers all the key characteristics mentioned in this guide, including an open-source self-hosted version of itself.

OpenReplay is open-source

The community edition of OpenReplay is open-source, meaning all major features are completely extensible and have been tested and improved by our growing community of developers worldwide.

Not only that, but it also means teams can add missing features, custom plugins, and heuristics to target custom business needs without having to wait for OpenReplay's timetable.

Are you using an in-house state management solution? Not a problem, you can add your plugin for that and track the state changes natively.

Do you need to capture a particular behavior of your users inside your app? Easy, add your custom heuristics to our back end and you'll start getting those metrics immediately.

The extensibility of an open-source solution makes OpenReplay future-proof and compatible with whatever use case you can think of.

OpenReplay is self-hosted

Yes, if data privacy and data ownership are two significant concerns for your company, then you can go with the self-hosted option. Our Community Edition and Enterprise Edition can be self-hosted under your own infrastructure.

This adds an extra layer of security on top of the already existing one provided by OpenReplay to ensure the data generated by your users is safe and complies with internal and external regulations.

OpenReplay has all the tools your developers need

The DevTools integration that comes with OpenReplay out of the box includes all the must-have items any front-end developer would need:

- A detailed JavaScript console with all the information captured on the live console while the session was recorded. This includes JavaScript errors and exceptions but also info messages and warnings. They can easily be filtered by type, so if you have a verbose console, you can quickly find what you need.
- All the network requests from the Network tab. This tab is fully captured, and if enabled, you can also get details about response and request data to understand what information your front-end is asking for and getting from the back end.

- Custom plugin integration. The DevTool expands to include custom plugin data to make sure developers still work under a familiar UX, even when dealing with custom metrics.

Additionally, the DevTools integrate something we like to call X-Ray, which allows you to quickly spot up-to three different problems directly on timeline, for you to quickly jump ahead and find exactly what you're looking for.

What else do you need?

If you've read through this guide, OpenReplay ticks all the boxes for the ideal session replay tool for software teams.

Other than being open-source, extensible, and self-hosted, OpenReplay also:

- **Has a growing community behind it.** Both on our GitHub repository, you can find active discussions with our users and our Slack community, where people come to get support directly from the developers.

- **It is great for support operators.** The live session support that comes with OpenReplay (called Assist) allows the operator to understand what problems users are having while using the application. It also allows them to jump into a video call with their users and take remote control over their mouse pointer to show them exactly how to solve their problems. Live sessions are also recorded like any other session, which makes it a perfect fit for onboarding training sessions where new support operators can review the actions of their colleagues as if they were there when they took place.

- **Integrates with industry-leading solutions in different fields.** As part of the integrations provided by OpenReplay, you can interact and exchange information with some of the industry-leading solutions from different areas. For integration with back-end monitoring tools, you have options like: Sentry, BugSnag, CloudWatch, NewRelic, ElasticSearch (if you’re using Kibana), DataDog, and more. As for other types of integrations, you can create issues in GitHub or Jira directly from within a session replay. You can also share notifications from the platform directly into your Slack channel of preference. And because OpenReplay is open-source, if you need extra integrations, you can add them yourself and take the platform wherever you need it to.

- **OpenReplay has a value-based pricing model.** The pricing model behind OpenReplay focuses on the value the tool adds to your business rather than blindingly charging you for the amount of traffic you receive. Of course, the open-source, self-hosted community version is completely free, and you can use it without limitations. Keep in mind that some enterprise-specific features are not included in the community version.

Get started with OpenReplay now

If you want to get started with OpenReplay but don't know how, check out some of these links:

- Check out our tutorials to understand how to get started with your current tech stack.
- Join our Slack community and ask our devs directly!
- Check our GitHub repo and don't forget to star us!