Building a Thriving Developer Community

Sam Flamini
DevX Lead, Superfluid
Let’s Paint a Scenario...
And You Just Sponsored a Hackathon...

You’re building a really cool protocol
You’re building a really cool protocol

And You Just Sponsored a Hackathon...

You’ve paid $10k in travel expenses and sponsorship fees
And You Just Sponsored a Hackathon...

You’ve paid $10k in travel expenses and sponsorship fees

You tell everyone how excited you are to see what hackers will build
You’re building a really cool protocol

And You Just Sponsored a Hackathon…

You’ve paid $10k in travel expenses and sponsorship fees

You tell everyone how excited you are to see what hackers will build

And you tell your team to be ready: the event will be intense & devs will have lots of questions
Then...
Then...

Nothing...
The Hackathon Flop

You get 3 submissions

- 2 of them are Figma files that don’t actually use your tech

And your docs get ripped to shreds by frustrated developers
It’s not enough to build awesome tools
It’s not enough to build awesome tools

You need people to know about them, and to make using them a great experience
Who Am I?

- Lead DevX at Superfluid
- Reformed SaaS salesperson & self-taught developer
- Host of Devs Do Something podcast

At Superfluid:

- 300+ projects built on Superfluid in the last ~12 months
- Teams building on Superfluid have raised 8 figures in total venture + grant funding
I have learned a *lot* in the devrel trenches
And I’m Still Learning...

I have learned a *lot* in the devrel trenches.
I have learned a *lot* in the devrel trenches

And I’m Still Learning...

I know the feeling of sponsoring a hackathon, and getting just a handful of submissions

I know the feeling of having our docs torn to shreds by frustrated developers
Superfluid is a cool protocol

0.2290650 USD

0.0000715 ETHx so far
0.00000039 ETHx per second
3201.52 USD per month
Superfluid is a cool protocol

Which makes my job easier…
But we’ve had to learn a *lot*
But we’ve had to learn a *lot*

Today, I’ll share some of those lessons
Why Build a Developer Community?
If Your Tech Matters to People, You Win
If Your Tech Matters to People, You Win

What Does Success Look Like?
What Does Success Look Like?
What Does Success Look Like?

1) An ecosystem of *new* and successful applications on top of your tech
What Does Success Look Like?

1) An ecosystem of *new* and successful applications on top of your tech

2) Integrations between your tech and *existing* products
Web2 Example: Twilio

Enable new applications:
- Uber, Lyft, AirBnB

Integrate with existing products:
- Literally anything that needs to send SMS notifications
Web3 Example: Ethereum

Enable new applications:
- Uniswap, Maker, ENS

Integrate with existing products:
- Global settlement layer + non financial use cases
What Is Developer Relations?
The Dev & Rel of DevRel
The Dev & Rel of DevRel

Developer
Libraries, APIs, Docs, Tooling, Smart Contracts

- Looks like engineering
- You must PM your developer products
The Dev & Rel of DevRel

**Developer**
Libraries, APIs, Docs, Tooling, Smart Contracts

- Looks like engineering
- You must PM your developer products

**Relations**
Tutorials, videos, podcasts, workshops, meetups

- Looks like marketing
- But CANNOT feel like marketing
The Dev & Rel of DevRel

Developer
Libraries, APIs, Docs, Tooling, Smart Contracts

- Looks like engineering
- You must PM your developer products

Relations
Tutorials, videos, podcasts, workshops, meetups

- Looks like marketing
- But CANNOT feel like marketing

Great Developer Experience Teams Have Both!
The Dev In DevRel
You Need to PM Your Developer Products

What does this mean?

1) Your docs are your most valuable product

2) Understand your developer personas

3) Be mindful of your abstractions
“We view all user input as error”

Elon Musk on Tesla Autopilot
For your docs
For your docs

View all developer questions as error
Your Docs Are Your Most Valuable DevX Product

No matter how good your docs are, you will still get developer questions

- Obviously

But it’s a useful mindset to view developer confusion as a **failure of your documentation**

Every minute spent making your docs better is worth an hour of answering individual questions
How do you maximize the utility of your docs & developer tooling?
How do you maximize the utility of your docs & developer content?

Start by defining your developer personas
Case Study: Superfluid Developer Personas
Case Study: Superfluid Developer Personas

Segment By Skill Level

- Beginner
- Intermediate
- Gigabrain
Case Study: Superfluid Developer Personas

<table>
<thead>
<tr>
<th>Segment By Skill Level</th>
<th>Segment By Role/Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beginner</td>
<td>Hackathon/Indy Dev</td>
</tr>
<tr>
<td>Intermediate</td>
<td>Dev at potential</td>
</tr>
<tr>
<td>Gigabrain</td>
<td>integration partner</td>
</tr>
<tr>
<td></td>
<td>Future</td>
</tr>
<tr>
<td></td>
<td>founder/entrepreneur</td>
</tr>
</tbody>
</table>
For Beginners/Hackathon Devs

Docs
- Quickstart page

Tooling
- JS SDK
- Simple solidity libraries

Tutorials/Examples
- Beginner front end examples
- Zero to Hero video series + examples in Github
For Intermediate Devs

Docs
- Reference Docs
- Advanced guides sorted by topic

Tooling
- Typings for the SDK for 1st class Typescript support
- Developer console

Tutorials/Examples
- Full fledged example applications
For Entrepreneurial Devs

Docs

- A library of ideas we’d like to see built

Tooling

- Superfluid Reactor Program

Tutorials/Examples

- ‘Primitives’ which are useful as building blocks for larger applications
Other Considerations

Be **Mindful of Your Abstractions**

- Too *little* abstractions in your APIs/Libraries = you lose beginners
- Too *many* abstractions = you frustrate intermediate/advanced devs, and stifle innovation
Other Considerations

Be Mindful of Your Abstractions

- Too little abstractions in your APIs/Libraries = you lose beginners
- Too many abstractions = you frustrate intermediate/advanced devs, and stifle innovation

Naming Matters

- We haven’t done very well with this (:
- Think about this for your smart contracts!
Most Developers Aren’t Good at Explaining Their Product to Newcomers

Why?
They’re blind to assumed context

What does this mean?

You need to go back to zero and often explain things from the ground up.

Engineers are deep in the weeds, it’s not easy to be aware of how much context they have.
Developer Relations
You Can’t Market to Developers
You Can’t Market to Developers

You need to pique curiosity instead
You Can’t Market to Developers

You need to pique curiosity instead

1) What doesn’t work?

2) What does work?

3) Be mindful of assumed context
What doesn’t work when marketing developers?
What Doesn’t Work

Traditional ‘demos’ or sales/marketing collateral or language

Developers can see it from a mile away
What Doesn’t Work

Anything that feels like shilling

“I can’t define shilling, but *I know it when I see it*”
What Doesn’t Work

Convincing people to use your tech for use cases that don’t make sense

Engineers can sense the dishonesty, or they’ll discover it later

Sometimes the most persuasive thing is being honest about what your tech isn’t good for
What **does** work when marketing to developers?
What Does Work

Focus on education

Create useful technical content, even if it’s just adjacent to your technology!
Good Content: Example 1
What Does Work

For your protocol/technology: **Create interesting examples & proof of concepts**

Stay up to date with popular dev tools and other technologies

- Create tutorials & examples of how your product can be *integrated* with them
- It helps you stay in the conversation
What Does Work

Create **superstars** in your developer community

- Make them feel *special*

Promote the projects that your community builds
Spending Money on Developer Acquisition
Our industry spends a *lot* of money on developer acquisition.

What is it spent on?

1) Hackathons
2) Bounties
3) DevRel Team Members
How Do You Get the Most Out of Your DevX Spend?

Doing Hackathons, Bounties, and DevRel Teams the Right Way
You cannot throw money at your community and hope for a good outcome.

You get out what you put in.
Hackathon Selection Matters

You’re looking for high quality co sponsors, and organizations that put out good content of their own.

Ask yourself: what kind of community will this hackathon attract?
  ● And is this community who I want building on our protocol?

ETHGlobal & Devfolio are examples of great orgs.
Running the Hackathon

Prepare some ideas to hack on before the event

- Ideally, these ideas also integrate well with other sponsors
- Hackathon devs love to “prize stack”
- We have a page at hack.superfluid.finance

Be as friendly as possible to every dev you meet
Don’t use bounties to complete tasks
Don’t use bounties to complete tasks

Use them to get gigabrain looking at your repo
Bounties

We have mixed results when using bounties for task completion.

However, they are great for two things:

- Getting smart devs to learn more about your product or protocol
- Crowdsourcing tutorials & educational content
DevRel Failure Modes
Common Failure Modes

1) Trying to throw money at the problem
2) Not seeding your community with ideas & examples
3) Assuming too much context in your docs, tutorials, & talks
4) Doing too much low leverage work
5) Not being willing to do any low leverage work
Remember What Success Looks Like
What Does Success Look Like?

A thriving community of developers who:

1) Tell their friends, bosses, and teammates about your product

2) Help other devs within your community

3) Build integrations between your product and their own

4) Launch businesses and projects of their own on your tech
Get in touch 👋

I’m still learning too!

@sflamini5

Sam F | Superfluid#0902
Thank You!