How Stellar's dApp Tooling Optimizes for Joy

me

me

- Chad O

me

- Chad O
  - (chadoh)

me

- Chad O
  - (chadoh)
- Cofounder & CEO, Aha Labs

me

- Chad O
  - (chadoh)
- Cofounder & CEO, Aha Labs
- prev

me

- Chad O
  - (chadoh)
- Cofounder & CEO, Aha Labs
- prev
  - Ruby (on Rails)

me

- Chad O
  - (chadoh)
- Cofounder & CEO, Aha Labs
- prev
  - Ruby (on Rails)
  - NEAR

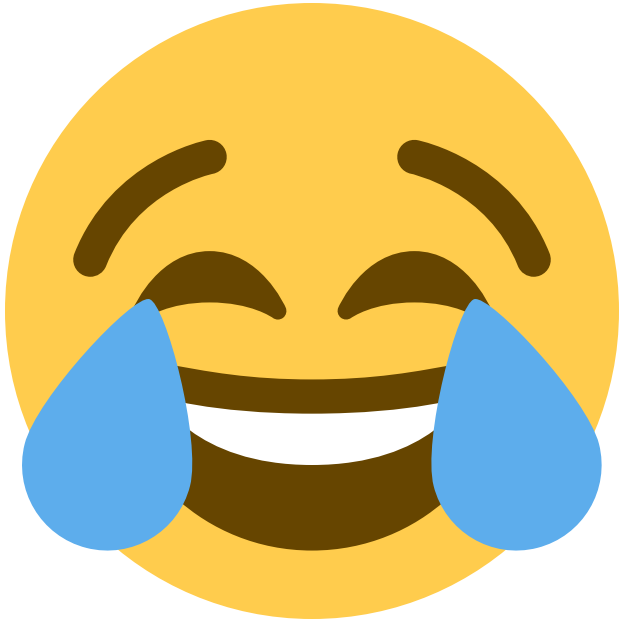
How Stellar's dApp Tooling Optimizes for Joy

How Stellar's (dApp) Tooling Optimizes for Joy

How Stellar('s (dApp) Tooling) Optimizes for Joy


JOY

:joy:



joy

noun

- Intense and especially ecstatic or exultant happiness, or an instance of such feeling.

joy

noun

- Intense and especially ecstatic or exultant happiness, or an instance of such feeling.
- 

joy

noun

- the emotion evoked by well-being, success, or good fortune

joy

noun

- the emotion evoked by well-being, success, or good fortune or by the prospect of possessing what one desires

joy

noun

- the emotion evoked by well-being, success, or good fortune or by the prospect of possessing what one desires, *delight*

joy

noun

- the emotion evoked by well-being, success, or good fortune or by the prospect of possessing what one desires, *delight*
- a source or cause of delight

joy

noun

- the emotion evoked by well-being, success, or good fortune or by the prospect of possessing what one desires, *delight*
- a source or cause of delight
- 

stellar

stellar

1. real value, real utility

stellar

1. real value, real utility
2. familiar developer workflows

stellar

1. real value, real utility
2. familiar developer workflows
3. built-in programmability

stellar

1. real value, real utility
2. familiar developer workflows
3. built-in programmability
    - command line example

stellar

1. real value, real utility
2. familiar developer workflows
3. built-in programmability
    - command line example
    - app example

stellar

1. real value, real utility
2. familiar developer workflows
3. built-in programmability
    - command line example
    - app example
4. you

1. real value, real utility

blockchains

what are they good for?

as with data,
so with value

prev 30 years: info

prev 30 years: info
next 30 years: value

prev 30 years: info
next 30 years: value

real-world utility

real-world utility

- MoneyGram

real-world utility

- MoneyGram
- UNHCR

real-world utility

- MoneyGram
- UNHCR
- Decaf, Beans, Blend, ...

real-world utility

- MoneyGram
- UNHCR
- Decaf, Beans, Blend, ...
- almost a decade of real value

~~1. real value, real utility~~

2. familiar developer workflows

stellar network container

`stellar network container`

- Horizon

`stellar network container`

- Horizon
- RPC

`stellar network container`

- Horizon
- RPC
- Friendbot

stellar network container

`stellar network container`

- no Ganache or whatever

`stellar network container`

- no Ganache or whatever
- local development

`stellar network container`

- no Ganache or whatever
- local development
- controlled environment for tests

`stellar network container`

- no Ganache or whatever
- local development
- controlled environment for tests
- snapshotting

`stellar network container`

- no Ganache or whatever
- local development
- controlled environment for tests
- snapshotting
- in the future: UI

~~2. familiar developer workflows~~

3. built-in programmability

ABIs

Application Binary Interfaces

ABIs published to...

ABIs published to… ???

Etherscan API

bake it in

contract spec

really slick developer tools

example

example

```
stellar contract init my-project
--with-example increment
```

```rust
/// Increment increments an internal counter, and returns the value.
pub fn increment(env: Env) -> u32 {
    ...
}
```

```rust
/// Increment increments an internal counter, and returns the value.
pub fn increment(env: Env) -> u32 {
    ...
}
```

```
stellar contract build
```

```rust
/// Increment increments an internal counter, and returns the value.
pub fn increment(env: Env) -> u32 {
    ...
}
```

```
stellar contract deploy --wasm .../increment.wasm
--alias counter
```

```rust
/// Increment increments an internal counter, and returns the value.
pub fn increment(env: Env) -> u32 {
    ...
}
```

```
stellar contract invoke --id counter -- --help
```

```rust
/// Increment increments an internal counter, and returns the value.
pub fn increment(env: Env) -> u32 {
    ...
}
```

```
stellar contract invoke --id counter -- --help
```

```
Commands:
  increment  Increment increments an internal counter, and returns the value.
  help       Print this message or the help of the given subcommand(s)
```

```rust
/// Increment increments an internal counter, and returns the value.
pub fn increment(env: Env) -> u32 {
    ...
}
```

```
stellar contract invoke --id counter -- --help
```

```
Commands:
  increment  Increment increments an internal counter, and returns the value.
  help       Print this message or the help of the given subcommand(s)
```

```
stellar contract invoke --id counter -- increment
```

```rust
/// Increment increments an internal counter, and returns the value.
pub fn increment(env: Env) -> u32 {
    ...
}
```

```rust
/// Increment counter, return new value
pub fn increment(env: Env) -> u32 {
    ...
}
```

```rust
/// Increment counter by `by`, return new value
pub fn increment(env: Env, by: u32) -> u32 {
    ...
}
```

```rust
/// Increment counter by `by`, return new value
pub fn increment(env: Env, by: u32) -> u32 { ... }
/// Get current value of counter
pub fn get(env: Env) -> u32 { ... }
```

```rust
/// Increment counter by `by`, return new value
pub fn increment(env: Env, by: u32) -> u32 { ... }
/// Get current value of counter
pub fn get(env: Env) -> u32 { ... }
```

```
stellar contract build
```

```rust
/// Increment counter by `by`, return new value
pub fn increment(env: Env, by: u32) -> u32 { ... }
/// Get current value of counter
pub fn get(env: Env) -> u32 { ... }
```

```
stellar contract deploy --wasm .../increment.wasm
```

```
--alias counter
```

```rust
/// Increment counter by `by`, return new value
pub fn increment(env: Env, by: u32) -> u32 { ... }
/// Get current value of counter
pub fn get(env: Env) -> u32 { ... }
```

```
stellar contract invoke --id counter -- --help
```

```rust
/// Increment counter by `by`, return new value
pub fn increment(env: Env, by: u32) -> u32 { ... }
/// Get current value of counter
pub fn get(env: Env) -> u32 { ... }
```

```
stellar contract invoke --id counter -- --help
```

```
Commands:
  increment  Increment counter by `by`, return new value
  get        Get current value of counter
  help       Print this message or the help of the given subcommand(s)
```

```rust
/// Increment counter by `by`, return new value
pub fn increment(env: Env, by: u32) -> u32 { ... }
```

```
stellar contract invoke --id counter -- --help
```

```
stellar contract invoke --id counter -- increment
```

```
--help
```

```rust
/// Increment counter by `by`, return new value
pub fn increment(env: Env, by: u32) -> u32 { ... }
```

```
stellar contract invoke --id counter -- increment
```

```
--help
```

```
Increment counter by `by`, return new value

Usage: increment [OPTIONS]

Options:
      --by <u32>
          Example:
            --by 1
```

```rust
/// Increment counter by `by`, return new value
pub fn increment(env: Env, by: u32) -> u32 { ... }
```

```
stellar contract invoke --id counter -- increment
```

```
--help
```

```
stellar contract invoke --id counter -- increment
```

```
--by 5
```

```
alias counter="stellar contract invoke --id counter --"
```

```
alias counter="stellar contract invoke --id counter --"
```

```
counter --help
```

```
alias counter="stellar contract invoke --id counter --"
```

```
counter --help
```

```
counter increment --help
```

```
alias counter="stellar contract invoke --id counter --"
```

```
counter --help
```

```
counter increment --help
```

```
counter increment --by 5
```

javascript

javascript

```
import { Client } from '@stellar/stellar-sdk/counter'
```

javascript

```
import { Client } from '@stellar/stellar-sdk/counter'

const counter = Client.from({ contractId: 'C…' })
```

javascript

```
import { Client } from '@stellar/stellar-sdk/counter'

const counter = Client.from({ contractId: 'C…' })

counter.increment({ by: 5 })
```

typescript

typescript

```
stellar contract bindings typescript --id counter
--output-dir packages/counter
```

typescript

```
stellar contract bindings typescript --id counter
--output-dir packages/counter
```

package.json

```
  "workspaces": [
    "packages/*"
  ],
```

typescript

```
stellar contract bindings typescript --id counter
--output-dir packages/counter
```

```
import { Client, networks } from 'counter'
```

typescript

```
stellar contract bindings typescript --id counter
--output-dir packages/counter
```

```
import { Client, networks } from 'counter'

const counter = new Client({ ...networks.testnet })
```

fromJSON
get
increment
options
spec
txFromJSON
txFromXDR

Construct and simulate a increment transaction. Returns an `AssembledTransaction` object which will have a `result` field containing the result of the simulation. If this transaction changes contract state, you will need to call `signAndSend()` on the returned object.
Increment counter by `by`, return new value

```
await counter.
```

```
(property) Client.increment: ({ by }: {
    by: number;
}, options?: {
    fee?: number;
    timeoutInSeconds?: number;
    simulate?: boolean;
} | undefined) ⇒ Promise<AssembledTransaction<number>>
```

Construct and simulate a increment transaction. Returns an `AssembledTransaction` object which will have a `result` field containing the result of the simulation. If this transaction changes contract state, you will need to call `signAndSend()` on the returned object. Increment counter by `by`, return new value

```
await counter.increment({ by: 5 });
```

~~3. built-in programmability~~

4. you

don't just build apps

don't just build apps
build tooling!

the foundations you need

the foundations you need

1. technological

the foundations you need

1. technological

- Rust

the foundations you need

1. technological

- Rust

- Typescript

the foundations you need

1. technological
2. financial

the foundations you need

1. technological
2. financial
    - Stellar Community Fund dev tooling track

the foundations you need

1. technological
2. financial
3. community

the foundations you need

1. technological
2. financial
3. community
    - Okashi

the foundations you need

1. technological
2. financial
3. community
    - Okashi
    - CommuniDAO

the foundations you need

1. technological
2. financial
3. community
    - Okashi
    - CommuniDAO
    - Loam

the foundations you need

1. technological
2. financial
3. community
    - Okashi
    - CommuniDAO
    - Loam
        - (come see me again!)

got ideas?
build on Stellar!

questions?

1. real value, real utility
2. familiar developer workflows
3. built-in programmability
    - command line example
    - app example
4. you