curl://up

Deprecating legacy, introducing news

Items

Why remove things

How to remove things

What things we remove(d)

"experimental" features

Why remove things

Old

Bad

Unused / untested

Not in line with current thinking

Keep the complexity level down

Keep the footprint down

How to remove things

docs/DEPRECATE.md

Must not break API or ABI promises

Disabled-by default (no warning)

Disabled in code (next-release warning)

Removed from code (>= 6 months from previous step)

Removed things

We removed all axTLS code in 7.63.0

- Basically abandoned project

- Doesn't support modern TLS features

- Doesn't allow external bug reports

- Virtually no discussions

Removing HTTP Pipelining

Not to be confused with persistent connections.

We will remove all Pipelining code in 7.65.0

- Rarely used

- Buggy

- Sparse test coverage

- The popular browsers already dropped it long ago

- "Done proper" with HTTP/2

Removing global DNS cache

We will remove global DNS cache code in 7.65.0

- Rarely used

- Global without locking isn't thread-safe

- Recommended in docs to avoid since 2004!

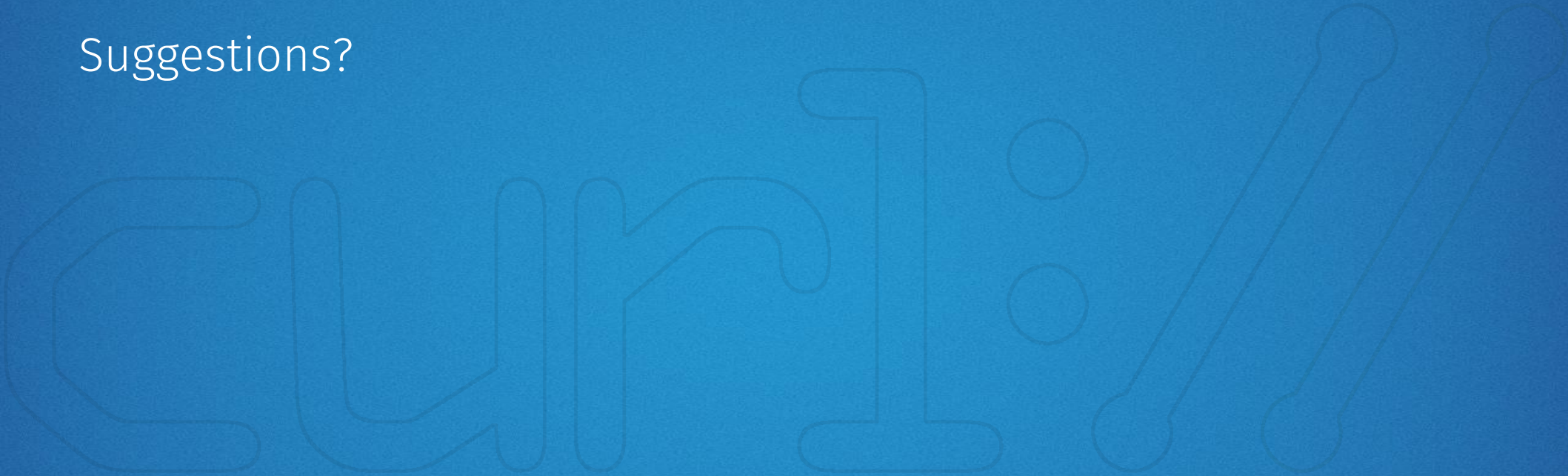- Better/proper cache sharing with share interface

Make HTTP/0.9 opt-in

HTTP/0.9 gets disabled by default in September 2019

- Rarely used protocol

- Surprising to users

- Security risk?

- Possibly an ABI break but one we can motivate

More cleanups?

Suggestions?

Experimental code

Add early code for testing and "polishing" without immediately making promises of API and ABI.

- Because very few people use non-releases

- Disabled by default in the build

- Documented as "experimental"

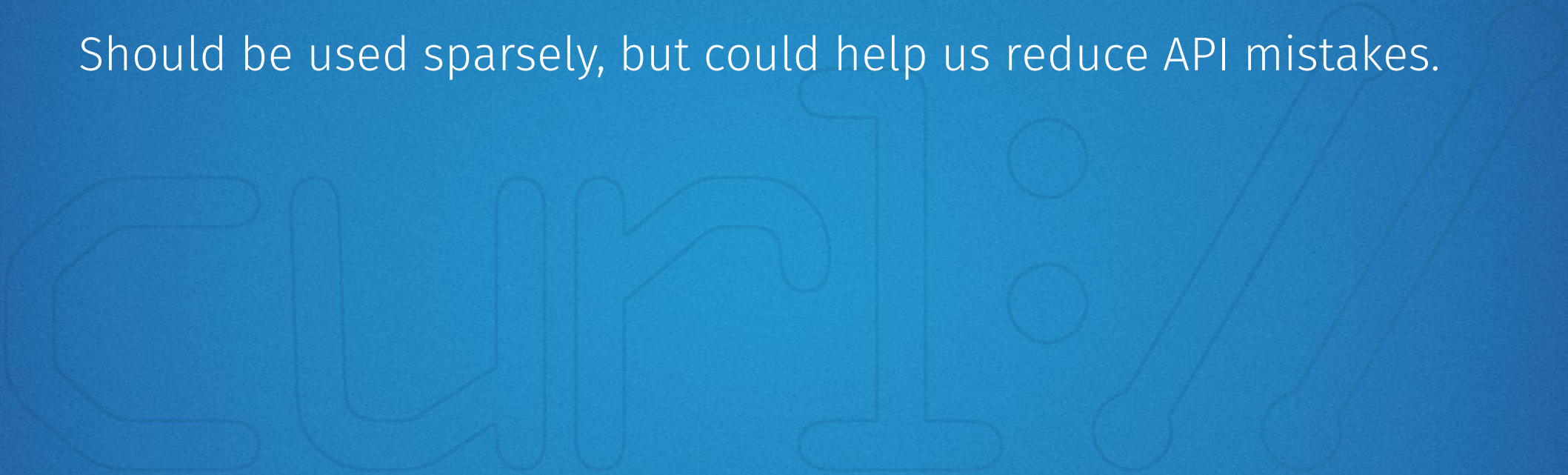- Should only remain tagged like this for a limited period of time

Experimental code: alt-svc

Alt-svc is a way for servers to inform clients that the same "origin" also exists elsewhere (port, host, protocol) for NNN seconds.

- The header was introduced for HTTP/2

- Necessary for bootstrapping HTTP/3 by spec

- Still fairly unused in the wild

- Risks remaining virtually unused in curl until HTTP/3 tests starts

Experimental code: more?

Should be used sparsely, but could help us reduce API mistakes.