

Hex Then and Now

Wojtek Mach

Hex Then and Now (For Me)

Wojtek Mach



Dashbit

Elixir Development Subscription



hex

Hex.pm

The screenshot shows the homepage of the Hex package manager website. The URL 'hex.pm' is visible in the browser's address bar. The page features a dark header with the 'hex' logo and navigation links for 'Packages', 'Pricing', 'Docs', and 'Log in'. The main heading 'The package manager for the Erlang ecosystem' is displayed prominently. Below it is a search bar with the placeholder 'Find packages'. Two sections are shown: 'Using with Elixir' (with a purple background and a flame icon) and 'Using with Erlang' (with a red background and a hex icon). Both sections provide instructions on how to integrate Hex into their respective environments.

hex.pm

Packages Pricing Docs Log in

The package manager for the Erlang ecosystem

Find packages

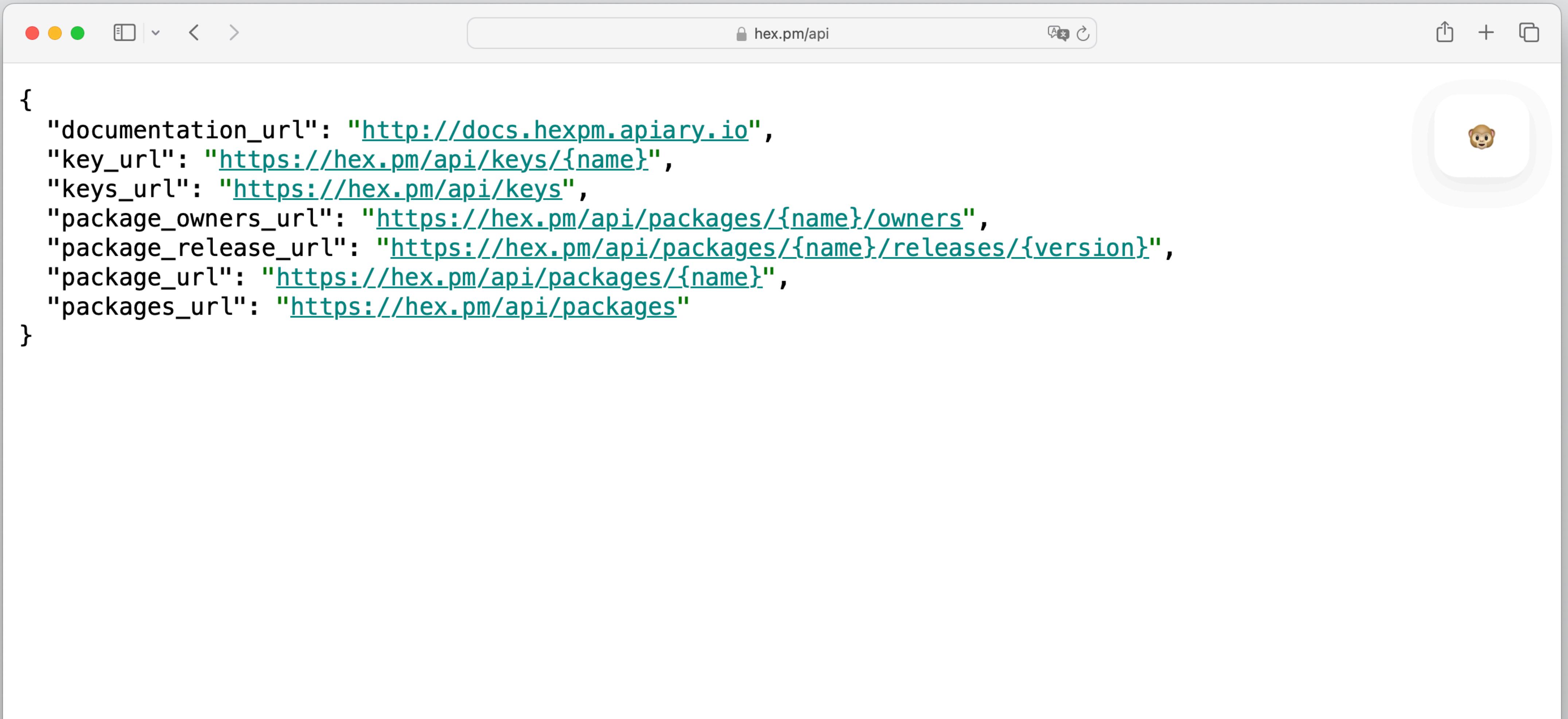
Using with Elixir

Specify your Mix dependencies as two-item tuples like `{:plug, "~> 1.1.0"}` in your dependency list, Elixir will ask if you want to install Hex if you haven't already.

Using with Erlang

Download `rebar3`, put it in your `PATH` and give it executable permissions. Now you can specify Hex dependencies in your `rebar.config` like `{deps, [hackney]}`.

Hex.pm/api



A screenshot of a web browser window displaying the Hex.pm API documentation. The URL in the address bar is `hex.pm/api`. The page content is a JSON object with the following structure:

```
{  
  "documentation_url": "http://docs.hexpm.apiary.io",  
  "key_url": "https://hex.pm/api/keys/{name}",  
  "keys_url": "https://hex.pm/api/keys",  
  "package_owners_url": "https://hex.pm/api/packages/{name}/owners",  
  "package_release_url": "https://hex.pm/api/packages/{name}/releases/{version}",  
  "package_url": "https://hex.pm/api/packages/{name}",  
  "packages_url": "https://hex.pm/api/packages"  
}
```

The JSON object contains several URLs related to the Hex.pm API, such as documentation, key management, package owners, releases, and packages. The URLs are primarily in red, indicating they are links.

Hex Specifications

The screenshot shows a GitHub repository page for `hexpm/specifications`. The page has a light gray background with a white header bar. The header includes standard browser controls (red, yellow, green buttons, back, forward, search), a URL bar with the address, and a right side with a user icon, a plus sign, and a refresh icon.

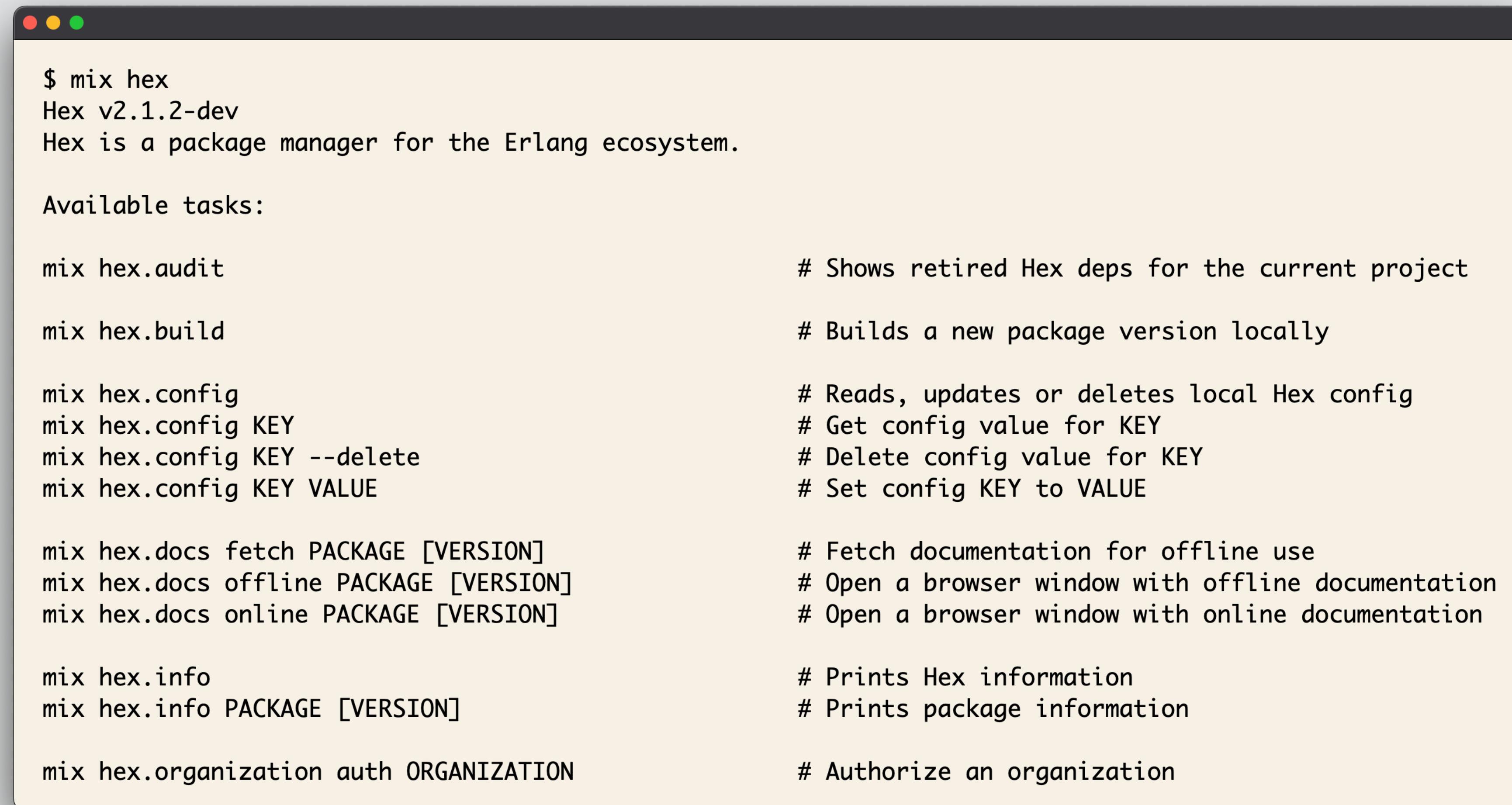
The main content area has a header "README" with a book icon. Below it is a large section titled "Specifications" with a bold font. A horizontal line follows this section. To the right of the "Specifications" section is a "Languages" section with a green progress bar and a single entry: "API Blueprint 100.0%".

Under the "Specifications" section, there is a bulleted list of links:

- [Endpoints](#)
- [HTTP API](#)
- [ETS registry format \(Deprecated\)](#)
- [New registry format](#)
- [Tarball format](#)
- [Package metadata](#)
- [purl \(Package URL\) format](#)

Below the "Specifications" section is another section titled "Repositories and mirrors" with a bold font. A horizontal line follows this section. At the bottom of the page, a note states: "To host a repository only the [Repository endpoint](#) has to be implemented."

Hex for Mix



```
$ mix hex
Hex v2.1.2-dev
Hex is a package manager for the Erlang ecosystem.

Available tasks:

mix hex.audit                                     # Shows retired Hex deps for the current project

mix hex.build                                      # Builds a new package version locally

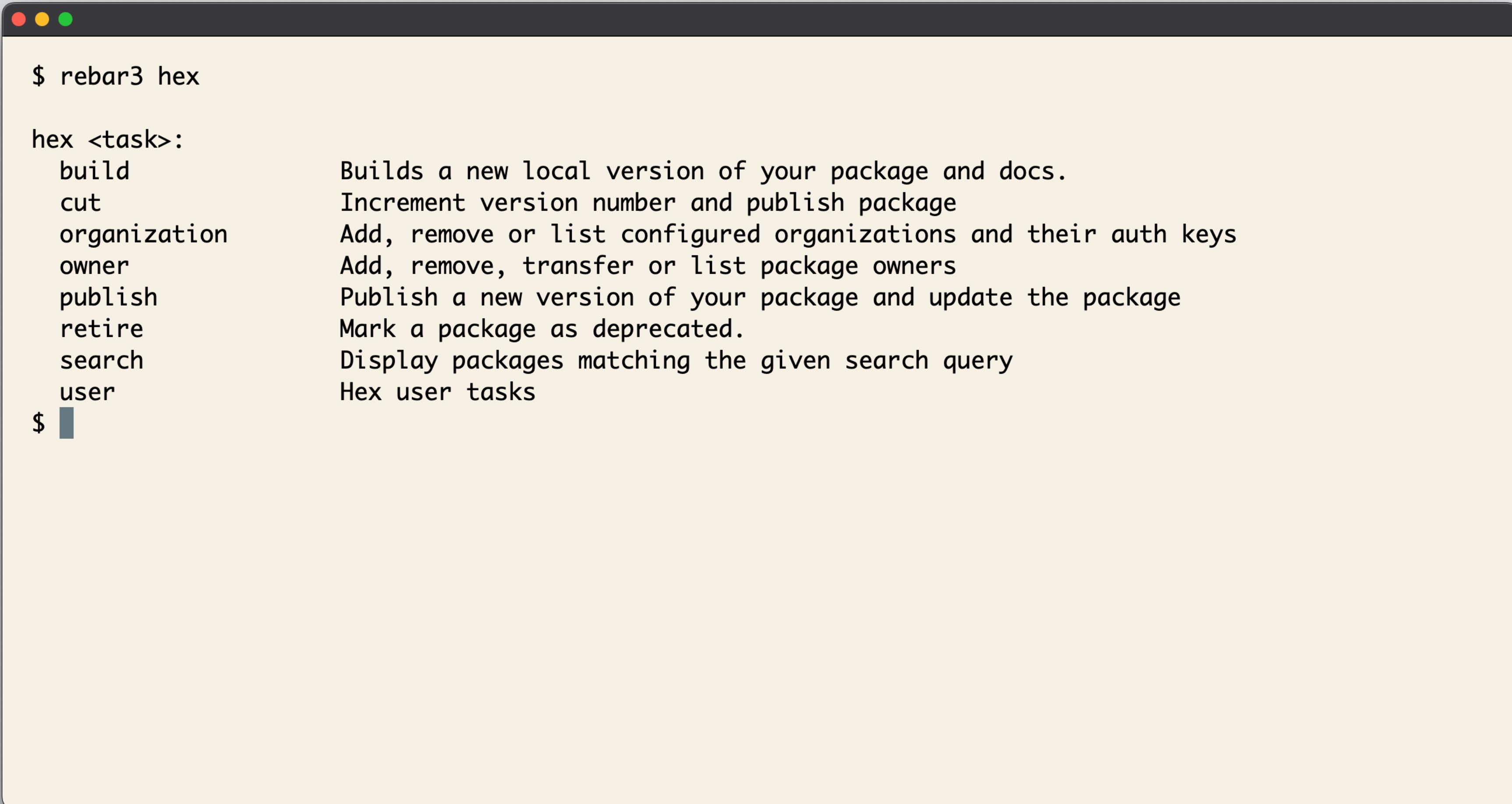
mix hex.config                                     # Reads, updates or deletes local Hex config
mix hex.config KEY                                # Get config value for KEY
mix hex.config KEY --delete                         # Delete config value for KEY
mix hex.config KEY VALUE                           # Set config KEY to VALUE

mix hex.docs fetch PACKAGE [VERSION]              # Fetch documentation for offline use
mix hex.docs offline PACKAGE [VERSION]            # Open a browser window with offline documentation
mix hex.docs online PACKAGE [VERSION]             # Open a browser window with online documentation

mix hex.info                                       # Prints Hex information
mix hex.info PACKAGE [VERSION]                    # Prints package information

mix hex.organization auth ORGANIZATION          # Authorize an organization
```

Hex for Rebar3



```
$ rebar3 hex

hex <task>:
  build           Builds a new local version of your package and docs.
  cut             Increment version number and publish package
  organization    Add, remove or list configured organizations and their auth keys
  owner           Add, remove, transfer or list package owners
  publish         Publish a new version of your package and update the package
  retire          Mark a package as deprecated.
  search          Display packages matching the given search query
  user            Hex user tasks
$
```

Hex Core

The screenshot shows a web browser window displaying the Hex Core documentation for the `hex_repo` module. The URL in the address bar is `hexdocs.pm/hex_core/hex_repo.html#get_names/1`. The left sidebar lists various modules under `hex_core v0.10.3`, with the `MODULES` tab selected. The main content area shows the `get_names(Config)` function, which gets names resource from the repository. It includes examples of how to call the function with a configuration and a snippet of Elixir code demonstrating its use.

hex_core
v0.10.3

PAGES MODULES

- hex_api
- hex_api_auth
- hex_api_key
- hex_api_organization
- hex_api_organization_member
- hex_api_package
- hex_api_package_owner
- hex_api_release
- hex_api_user
- hex_core
- hex_http
- hex_http_https
- hex_licenses
- hex_registry

get_names(Config)

Gets names resource from the repository.

Examples:

```
> hex_repo:get_names(hex_core:default_config()).  
{ok,{200, ...,  
     #{packages => [  
         #{name => <<"package1">>}],  
         #{name => <<"package2">>}},  
     ... ]}}}
```

get_package(Config, Name)

Gets package resource from the repository.

Hex Repo

Hex Repo

Hexdocs

Hex Repo

Hexdocs

Hex Builds

Hex Repo

Hexdocs

Hex Builds

Hex Diff

Hex Repo

Hexdocs

Hex Builds

Hex Diff

Hex Preview

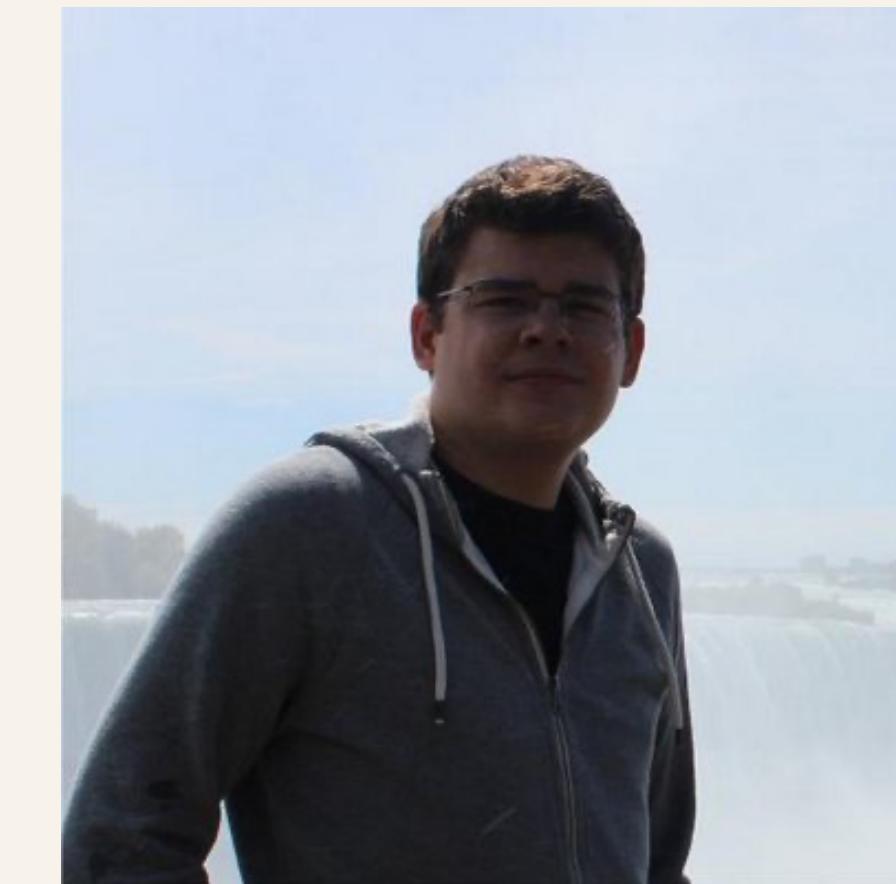
Hex Core



Eric Meadows-Jönsson



Todd Resudek



Wojtek Mach

github.com/hexpm/hex/commit/eeb76400ec97b631e95aa641db348fe

Initial commit · hexpm/hex@eeb7640

hexpm / hex

Type / to search

Code Issues Pull requests Discussions Actions Projects Wiki Security Insights

Initial commit

ericmj committed on Dec 23, 2013

main · v2.1.1 · v0.0.1 0 parents commit eeb7640

Filter files...

6 files changed +41 -0 lines changed

.gitignore

... @@ -0,0 +1,4 @@

1 + /_build
2 + /deps
3 + erl_crash.dump
4 + *.ez

+4

README.md

... @@ -0,0 +1,3 @@

+3

lib

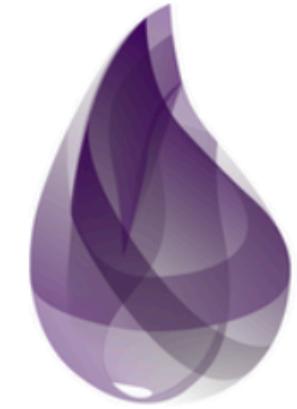
explex.ex

mix.exs

test

explex_test.exs

test_helper.exs



elixir

HOME INSTALL LEARNING DOCS GUIDES CASES BLOG

Elixir v0.13.0 released, hex.pm and ElixirConf announced

April 21, 2014 · by José Valim · in [Releases](#)

Hello folks!

Elixir v0.13.0 has been released. It contains changes that will effectively shape how developers will write Elixir code from now on, making it an important milestone towards v1.0! On this post we are going to cover some of those changes, the road to Elixir v1.0, as well as the announcement of [hex.pm](#).

Before we go into the changes, let's briefly talk about ElixirConf!

ElixirConf

News: [Elixir v1.17 released](#)

Search...

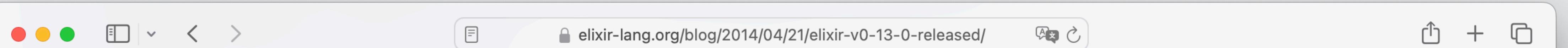


BLOG CATEGORIES

- [Announcements](#)
- [Elixir in Production](#)
- [Internals](#)
- [Releases](#)

IMPORTANT LINKS

- [Development & Team](#)



Maps

Maps are key-value data structures:

```
iex> map = %{"hello" => :world}  
%{"hello" => :world}  
iex> map["hello"]  
:world  
iex> map[:other]  
nil
```

Maps do not have an explicit ordering and keys and values can be any term.

Maps can be pattern matched on:

```
iex> %{ "hello" => world} = map  
%{"hello" => :world}  
iex> world  
:world  
iex> %{ } = map  
%{"hello" => :world}
```



elixir

HOME INSTALL LEARNING DOCS GUIDES CASES BLOG

Elixir v1.0 released

September 18, 2014 · by José Valim · in [Releases](#)

We are glad to announce Elixir v1.0 is finally out. It has been 8005 commits [by 189 contributors](#), including the initial commit on [January 9th, 2011!](#)

What's Elixir?

Elixir is a dynamic, functional language designed for building scalable and maintainable applications.

Elixir leverages the Erlang VM, known for running low-latency, distributed and fault-tolerant systems, while also being successfully used in web development and the embedded software domain.

News: [Elixir v1.17 released](#)

Search...



BLOG CATEGORIES

- [Announcements](#)
- [Elixir in Production](#)
- [Internals](#)
- [Releases](#)

IMPORTANT LINKS

- [Development & Team](#)

A screenshot of a GitHub pull request page for the hexpm/hexpm repository. The page shows a pull request titled "Fix postgres setup instruction #58" that has been merged. The pull request details include the merge commit message from ericmj, the date (Oct 11, 2014), and the number of commits (1). A comment from wojtekmach is visible, describing an error encountered during migrations. The right side of the page displays review and assignment settings, showing no reviews or assignees.

Fix postgres setup instruction #58

Merged ericmj merged 1 commit into `hexpm:master` from `wojtekmach:fix_postgres_setup` on Oct 11, 2014

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 -0

wojtekmach commented on Oct 11, 2014

When running migrations for the first time I saw this error:

```
* running UP _build/dev/lib/hex_web/priv/migrations/20140323232653_add_package.epgsql
* running UP _build/dev/lib/hex_web/priv/migrations/20140510114425_add_install_pg_trgm.epgsql
* running UP _build/dev/lib/hex_web/priv/migrations/20140511133315_add_optional_to_release.epgsql
* running UP _build/dev/lib/hex_web/priv/migrations/20140518153329_add_checksum_to_release.epgsql
* running UP _build/dev/lib/hex_web/priv/migrations/20140527204944_change_packages_in_hex_web.epgsql
** (Postgrex.Error) ERROR (42501): permission denied to create extension "pg_trgm"
(ecto) lib/ecto/adapters/postgres/worker.ex:20: Ecto.Adapters.Postgres.Worker.query/1
(ecto) lib/ecto/adapters/postgres.ex:337: Ecto.Adapters.Postgres.use_worker/3
(elixir) lib/enum.ex:537: Enum."-each/2-lists^foreach/1-0-/2
(elixir) lib/enum.ex:537: Enum.each/2
(ecto) lib/ecto/adapters/postgres.ex:473: anonymous fn/3 in Ecto.Adapters.Postgres.migrate_up/3
(ecto) lib/ecto/adapters/postgres.ex:306: Ecto.Adapters.Postgres.transaction/3
(ecto) lib/ecto/adapters/postgres.ex:472: Ecto.Adapters.Postgres.migrate_up/3
(elixir) lib/elixir.ex:537: Enum."-each/2-lists^foreach/1-0-/2
(elixir) lib/elixir.ex:537: Enum.each/2
```

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

A screenshot of a GitHub pull request page for a project named "hexpm / hexpm". The URL in the address bar is github.com/hexpm/hexpm/pull/58. The page shows a pull request titled "Fix postgres setup instruction #58" which has been merged. The merge commit was pushed by ericmj on Oct 11, 2014, from the branch `wojtekmach:fix_postgres_setup` into the `hexpm:master` branch. The pull request has 1 commit, 0 checks, and 1 file changed. A comment from user **wojtekmach** on Oct 11, 2014, describes an error encountered during migration. The right sidebar displays various metadata for the pull request, including Reviewers, Assignees, Labels, Projects, and Milestones.

Fix postgres setup instruction #58

Merged ericmj merged 1 commit into `hexpm:master` from `wojtekmach:fix_postgres_setup` on Oct 11, 2014

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 -0

wojtekmach commented on Oct 11, 2014

Member

When running migrations for the first time I saw this error:

```
* running UP _build/dev/lib/hex_web/priv/migrations/20140323232653_add_package
* running UP _build/dev/lib/hex_web/priv/migrations/20140510114425_add_installs_.esl
* running UP _build/dev/lib/hex_web/priv/migrations/20140511133315_add_optional_to_r
* running UP _build/dev/lib/hex_web/priv/migrations/20140518153329_add_checksum_to_r
* running UP _build/dev/lib/hex_web/priv/migrations/20140527204944_change_packages_i
** (Postgrex.Error) ERROR (42501): permission denied to create extension "pg_trgm"
(ecto) lib/ecto/adapters/postgres/worker.ex:20: Ecto.Adapters.Postgres.Worker.qu
(ecto) lib/ecto/adapters/postgres.ex:337: Ecto.Adapters.Postgres.use_worker/3
(elixir) lib/enum.ex:537: Enum."-each/2-lists^foreach/1-0-/2
(elixir) lib/enum.ex:537: Enum.each/2
(ecto) lib/ecto/adapters/postgres.ex:473: anonymous fn/3 in Ecto.Adapters.Postgr
(ecto) lib/ecto/adapters/postgres.ex:306: Ecto.Adapters.Postgres.transaction/3
(ecto) lib/ecto/adapters/postgres.ex:472: Ecto.Adapters.Postgres.migrate_up/3
```

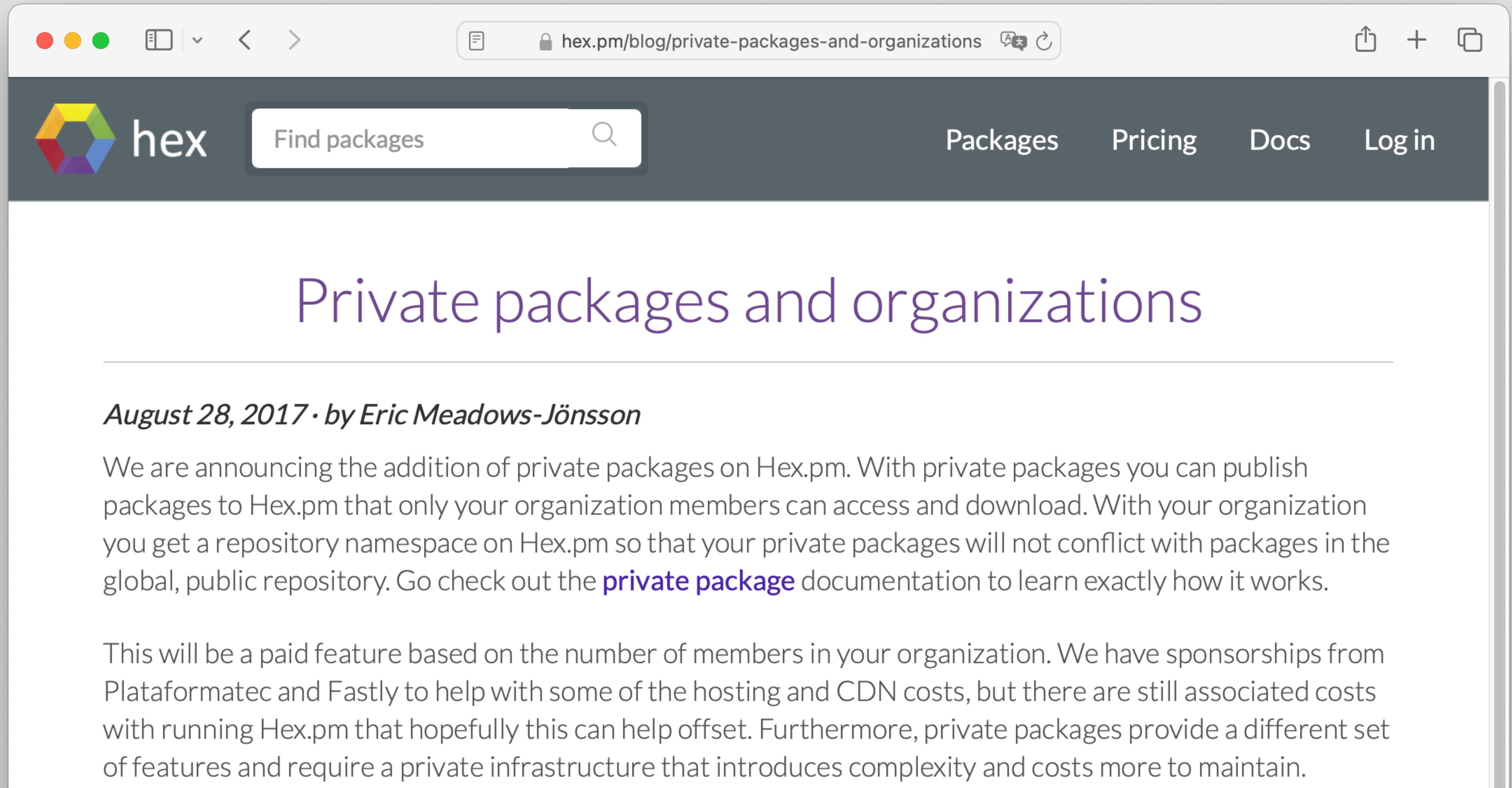
Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestones



The screenshot shows a web browser window with the URL hex.pm/blog/private-packages-and-organizations. The page header includes the Hex logo, a search bar with the placeholder "Find packages", and navigation links for "Packages", "Pricing", "Docs", and "Log in". The main content features a large purple heading "Private packages and organizations" followed by a timestamp "August 28, 2017 · by Eric Meadows-Jönsson". The text explains the addition of private packages, noting that they are organization-specific and conflict-free. It also mentions that this is a paid feature based on organization size, with sponsorships from Plataformatec and Fastly.

Private packages and organizations

August 28, 2017 · by Eric Meadows-Jönsson

We are announcing the addition of private packages on Hex.pm. With private packages you can publish packages to Hex.pm that only your organization members can access and download. With your organization you get a repository namespace on Hex.pm so that your private packages will not conflict with packages in the global, public repository. Go check out the [private package](#) documentation to learn exactly how it works.

This will be a paid feature based on the number of members in your organization. We have sponsorships from Plataformatec and Fastly to help with some of the hosting and CDN costs, but there are still associated costs with running Hex.pm that hopefully this can help offset. Furthermore, private packages provide a different set of features and require a private infrastructure that introduces complexity and costs more to maintain.

Pushing public packages will of course stay free and if you run an open source project that needs private packages you can contact us to get free access. The revenue from private packages will help us increase the

A screenshot of a web browser window displaying a blog post from the Hex website. The browser's top bar shows standard OS X controls (red, yellow, green buttons, minimize, maximize, close) and a tab for 'hex.pm/blog/announcing-hex-core'. The main content area has a dark grey header with the Hex logo (a colorful hexagon icon), a search bar containing 'Find packages', and navigation links for 'Packages', 'Pricing', 'Docs', and 'Log in'. The main title 'Announcing hex_core' is displayed in a large, purple, sans-serif font. Below the title, the date 'August 8, 2018 · by Wojtek Mach' is shown in a smaller, italicized font. The post begins with a paragraph about releasing the first version of hex_core, followed by a section explaining what Hex is and its various components.

Announcing hex_core

August 8, 2018 · by Wojtek Mach

Today we are releasing the first version of hex_core, an Erlang library to interact with Hex.pm and other servers implementing Hex specifications.

Before talking about hex_core, let's ask a simple question: What is Hex? The short answer is, it's a package manager for the Erlang ecosystem. The long answer is that by Hex we may mean a few different things:

1. A set of specifications of building clients and servers that can interact with each other:
<https://github.com/hexpm/specifications>
2. A server for hosting packages like the official server located at <https://hex.pm>
3. Clients for interacting with servers, e.g. **Hex** for Elixir and **rebar3_hex** for Erlang projects

The goal of hex_core is to be the reference implementation of Hex specifications used by Hex clients and

A screenshot of a web browser window. The address bar shows the URL hex.pm/blog/hex-v020-released. The page content is from the hex blog, featuring the hex logo, a search bar, and navigation links for Packages, Pricing, Docs, and Log in.

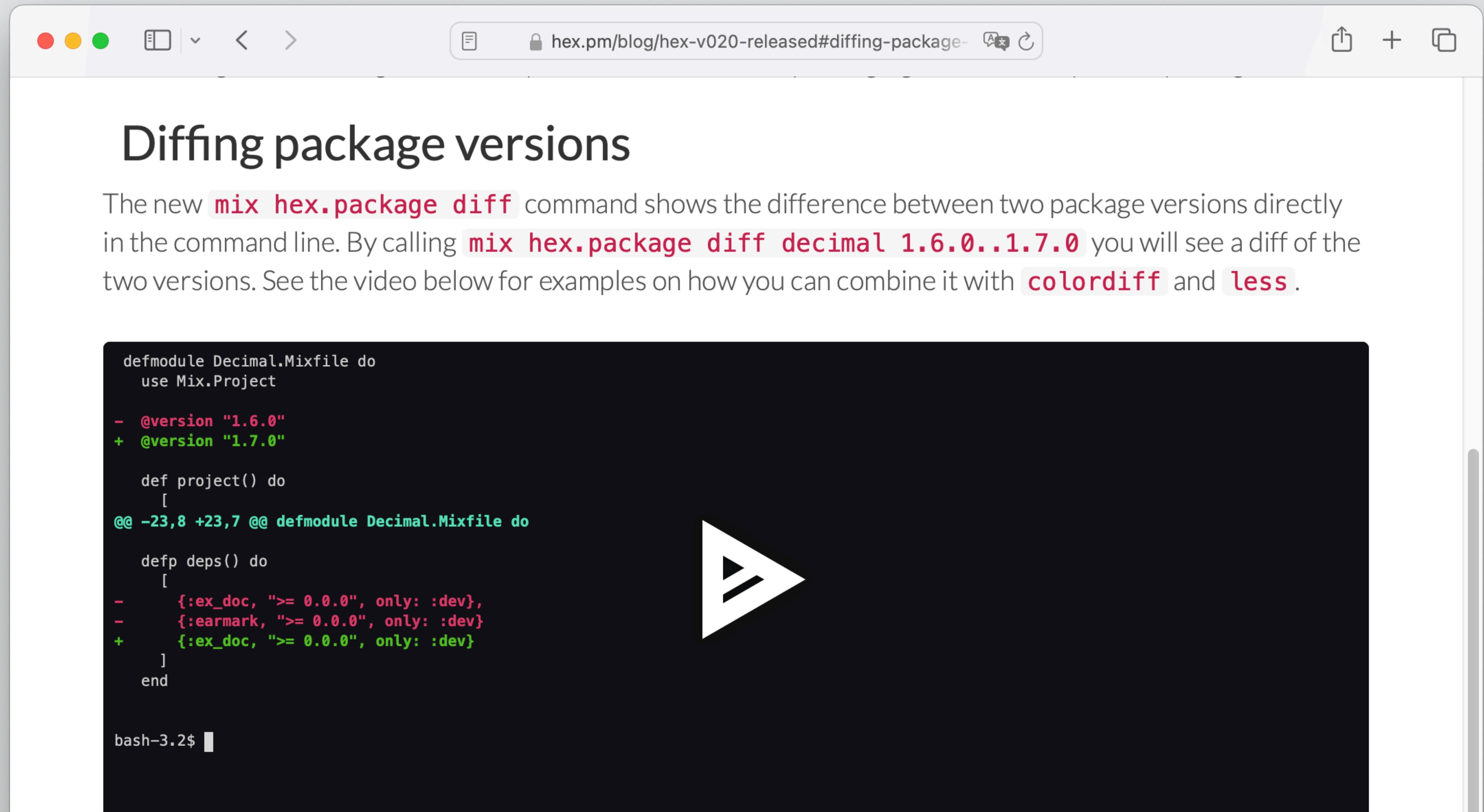
Hex v0.20 released

10 June, 2019 · by Eric Meadows-Jönsson

Organization managed public packages

A bit more than year ago we released private packages with organizations. With it you can add members to your organization and publish private packages only accessible by the members of the organization. The private packages are published to a separate repository (hosted by hex.pm) making sure they are separate from public packages.

Now we are adding the ability to also manage public packages through your organizations. This means that a company such as [Plataformatec](#) can use Hex organizations to manage open source Hex packages just like private packages can be managed by organizations. It can also be useful for open source projects that manage multiple packages with multiple maintainers.



The screenshot shows a web browser window with the URL hex.pm/blog/hex-v020-released#diffing-package. The main content is a heading "Diffing package versions" followed by a paragraph explaining the `mix hex.package diff` command. Below the text is a terminal session showing a diff of a Mixfile between version 1.6.0 and 1.7.0.

```
defmodule Decimal.Mixfile do
  use Mix.Project

  - @version "1.6.0"
  + @version "1.7.0"

  def project() do
    [
@@ -23,8 +23,7 @@
  defp deps() do
    [
      - {:ex_doc, ">= 0.0.0", only: :dev},
      - {:earmark, ">= 0.0.0", only: :dev}
      + {:ex_doc, ">= 0.0.0", only: :dev}
    ]
  end

bash-3.2$
```



```
$ mix hex.package fetch decimal
```



```
$ mix hex.package fetch decimal
decimal v2.1.1 downloaded to /Users/wojtek/decimal-2.1.1.tar
$
```



```
$ mix hex.package fetch decimal --unpack
```



```
$ mix hex.package fetch decimal --unpack  
decimal v2.1.1 extracted to /Users/wojtek/decimal-2.1.1  
$
```



```
$ mix hex.package diff decimal 2.1.0..2.1.1
```

```
@@ -1,5 +1,13 @@
# CHANGELOG

+## v2.1.1 (2023-04-26)
+
+Decimal v2.1 requires Elixir v1.8+.
+
+### Bug fixes
+
+* Fix `Decimal.compare/2` when comparing against `0`
+
## v2.1.0 (2023-04-26)

Decimal v2.1 requires Elixir v1.8+.

diff --git a/var/folders(mb/mv73n10n1kjd8gdymvf_4j0w0000gn/T/decimal-2.1.0-CA950D90/hex_metadata.config b/var/folders(mb/mv73n10n1kjd8gdymvf_4j0w0000gn/T/decimal-2.1.1-16F7B1D1/hex_metadata.config
index e34c1e3..f83241f 100644
--- a/var/folders(mb/mv73n10n1kjd8gdymvf_4j0w0000gn/T/decimal-2.1.0-CA950D90/hex_meta
:
```

A screenshot of a GitHub issue page for a repository named "hexpm / hexpm". The URL in the address bar is github.com/hexpm/hexpm/issues/848. The page shows the following navigation tabs: Code, Issues (12), Pull requests (4), Discussions, Actions, Projects (1), Wiki, and a three-dot menu. The "Issues" tab is currently selected.

Web based package diffs #848

Closed joladev opened this issue on Sep 8, 2019 · 6 comments

joladev commented on Sep 8, 2019 · edited

Hi! I wanted to bring this issue up for discussion. I'll try to describe the importance of package diffs, existing solutions, some possible architectures and some possible solutions that might fit hex.pm. By web-based diffs I mean highlighted outputs from `git diff` in a browser, with shareable links.

We first started seeing npm packages get hijacked, but recently RubyGems has been having issues (`rest_client`, `strong_password`, many others). By hijacking I mean the type of attack where someone gets access to the credentials of the author of a package and uploads malicious versions. Some examples of scenarios:

Assignees: No one—[assign yourself](#)

Labels: `Note:Discussion`

Projects: None yet

The screenshot shows a Mac OS X desktop environment with a browser window open. The browser's address bar displays the URL hex.pm/blog/announcing-hex-diff. The page content is from the Hex website, featuring a dark header with a colorful hexagonal logo, a search bar, and navigation links for Packages, Pricing, Docs, and Log in. The main content area has a light background and features a large purple title "Announcing Hex Diff". Below the title, a date "20 January, 2020 · by Johanna Larsson" is shown, followed by a paragraph about the new service and credits to the Hex team members. A large section below is titled "What does it do?" with a descriptive paragraph.

Announcing Hex Diff

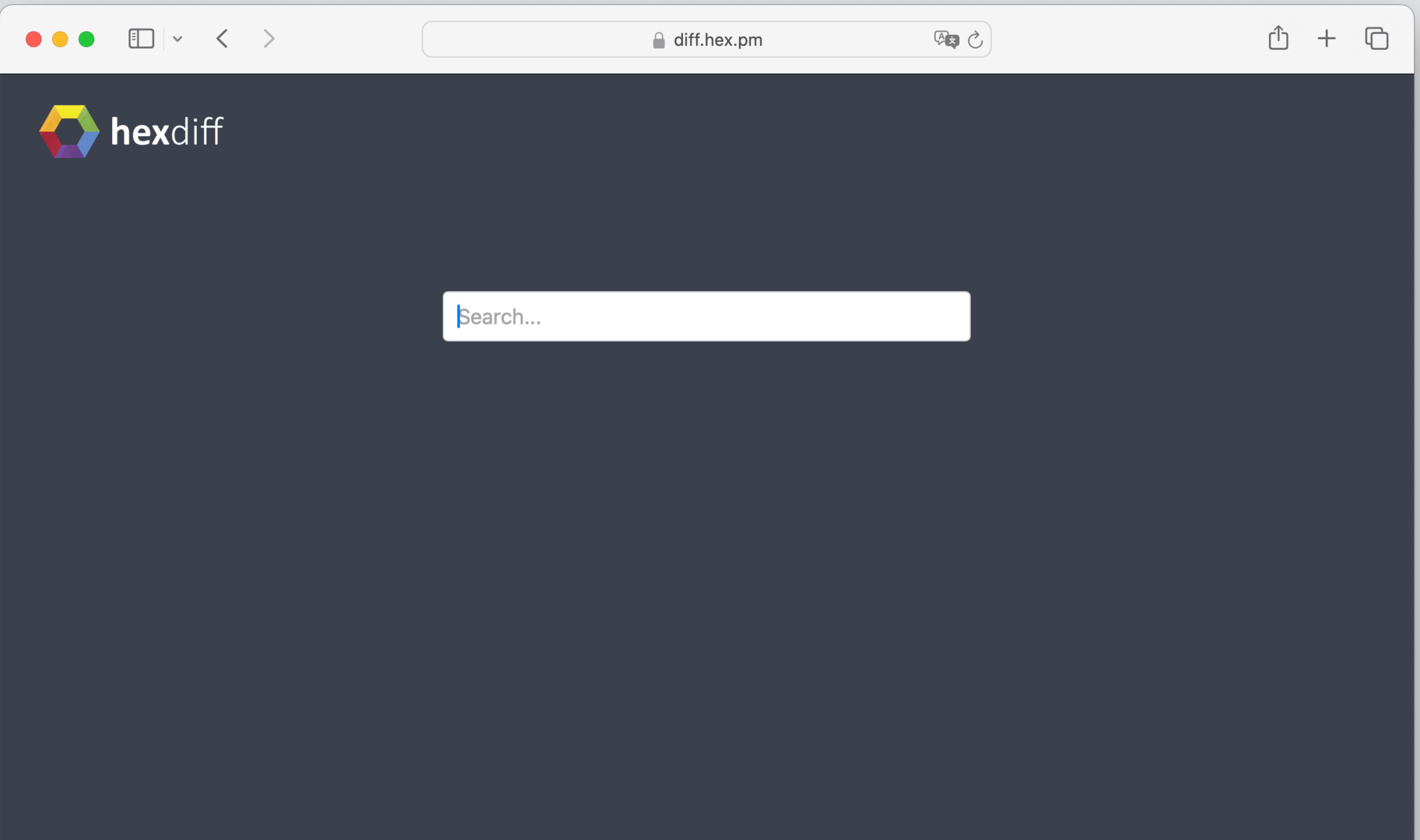
20 January, 2020 · by Johanna Larsson

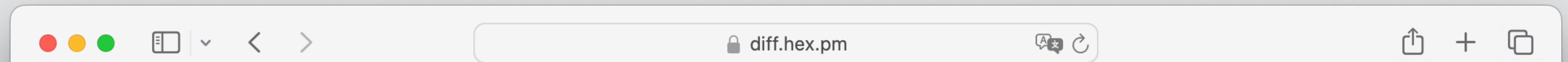
I'm incredibly excited to announce the new web-based Hex package differ: [diff.hex.pm](#), maintained by the Hex team! This is the result of the issue on the [hex.pm Github repo](#) and the discussion it started.

I'm super grateful to the Hex team, [Eric Meadows-Jönsson](#), [Wojtek Mach](#), and [Todd Resudek](#), for all their support and help in turning this idea into a live service.

What does it do?

In short, you input any Hex package name and a version range, and it will generate a highlighted git diff for you, right there in your browser. Not only that, but you can also share the link to the diff, and even highlight a specific row. [Please take a moment to try it out!](#)





hexdiff

telemetry

Did you mean: [telemetry_async](#) [telemetry_child_init](#)
[telemetry_decorator](#)

From

1.2.1

To

1.3.0

DIFF

The screenshot shows a web browser window with a dark theme. The address bar displays the URL `diff.hex.pm/diff/telemetry/1.2.1..1.3.0`. The page content is titled "telemetry" and "1.2.1 → 1.3.0". A "hexdiff" logo is visible in the top left corner. The main content area shows a "CHANGED" file named "CHANGELOG.md". The file content is as follows:

```
@@ -5,6 +5,18 @@ All notable changes to this project will be documented in this file.  
5 5     The format is based on [Keep a Changelog](https://keepachangelog.com/en/1.0.0/),  
6 6     and this project adheres to [Semantic Versioning](https://semver.org/spec/v2.0.0.html).  
7 7  
8 + ## [1.3.0](https://github.com/elixir-telemetry/telemetry/tree/v1.3.0)  
9 +  
10 + ### Added  
11 +  
12 + - Ability to return extra measurements from `telemetry:span/3`.  
13 +  
14 + ### Changed  
15 +  
16 + - Rewrite docs from edoc to OTP 27 `--moduledoc`/`-doc`.  
17 +  
18 + Internal macros `?DOC` and `?MODULEDOC` are used. They are no-ops prior to OTP 27.  
19 +
```

A screenshot of a web browser window displaying the Hex Preview homepage. The address bar shows the URL hex.pm/blog/introducing-hex-preview. The page features a dark header with the Hex logo (a colorful hexagon icon) and the word "hex". A search bar contains the placeholder "Find packages" with a magnifying glass icon. To the right of the search bar are links for "Packages", "Pricing", "Docs", and "Log in". The main content area has a light background and displays the title "Introducing Hex Preview" in large, purple, sans-serif font.

25 January, 2021 · by Todd Resudek

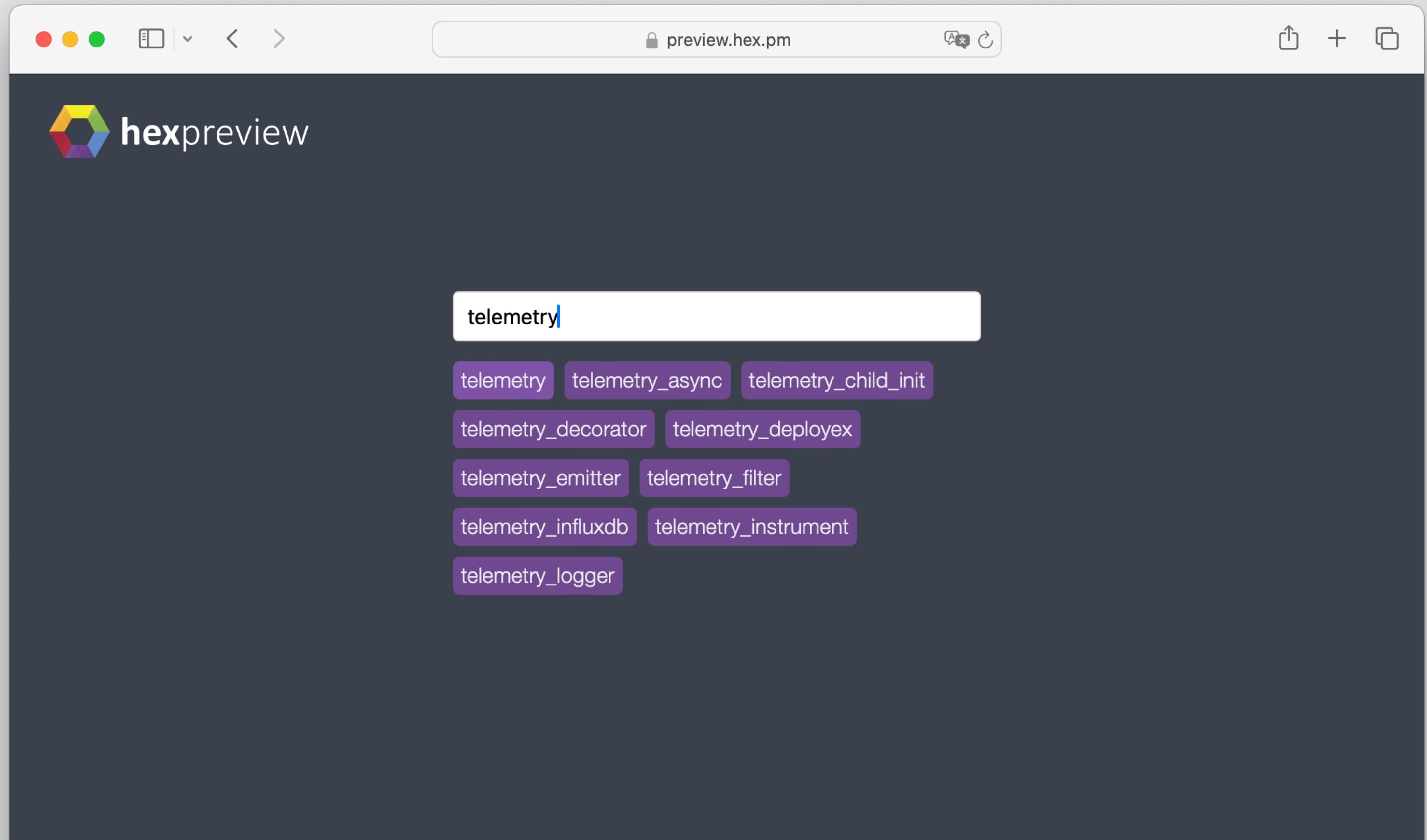
After months of development, I am proud to announce the launch of [preview.hex.pm](#)!

It is important to note that this was built on top of all the work that was put into [diff.hex.pm](#). Preview relies on much of the code contributed to that project (thanks, in no small part, to [Johanna Larsson](#).)

What does it do?

Hex Preview is an online tool for viewing the source files of a Hex package. By searching for the package name, and then selecting a version number, you will see all of the files contained in the release.

Can't I already do this on Github?



The screenshot shows a web-based interface for previewing a Hex package. The title bar indicates the URL is `preview.hex.pm`. The main content area features a dark background with several purple rounded rectangular buttons containing white text. At the top left is a logo consisting of overlapping colored squares (yellow, green, blue, red) followed by the text "hexpreview". Below this, a large button contains the word "telemetry". Underneath are several smaller buttons: "telemetry", "telemetry_async", "telemetry_child_init", "telemetry_decorator", "telemetry_deployex", "telemetry_emitter", "telemetry_filter", "telemetry_influxdb", "telemetry_instrument", and "telemetry_logger". Further down, a section titled "Version" contains the text "1.3.0". At the bottom is a large purple button with the word "PREVIEW" in white capital letters.

hexpreview

telemetry

telemetry telemetry_async telemetry_child_init
telemetry_decorator telemetry_deployex
telemetry_emitter telemetry_filter
telemetry_influxdb telemetry_instrument
telemetry_logger

Version

1.3.0

PREVIEW



hexpreview

Select File:

rebar.config

rebar.config

```
1 {erl_opts, [debug_info]}.
2 {deps, []}.
3
4 {profiles, [
5     {test, [{erl_opts, [nowarn_export_all]},
6             %% create junit xml for circleci
7             {ct_opts, [{ct_hooks, [cth_surefire]}]}],
8     {cover_enabled, true},
9     {cover_opts, [verbose]},
10    %% convert to data codecov understands
11    {plugins, [covertool]},
12    {covertool, [{coverdata_files, ["ct.coverdata"]}]}
13 ]}
```

hex.pm/packages/decimal

Find packages

Packages Pricing Docs Log in

decimal 2.1.1

Arbitrary precision decimal arithmetic.

Links

[Online documentation](#)

[GitHub](#)

Downloads

The chart displays the number of monthly downloads for the decimal package over the past 30 days. The y-axis represents the number of downloads, ranging from 0 to 56,000. The x-axis represents time, with labels for 0, 14,000, 28,000, 42,000, and 56,000. The data shows significant fluctuations, with peaks around 55,000 and troughs near 10,000. A label "Last 30 days, all versions" is positioned above the chart area.

License

Apache-2.0

Config

mix.exs

```
{:decimal, "~> 2.1"}
```

rebar.config

```
{decimal, "2.1.1"}
```

erlang.mk

```
dep_decimal = hex 2.1.1
```

Checksum

```
53cfe5f497ed0e7771ae
```

Build Tools

hex.pm/packages/decimal

Downloads

Last 30 days, all versions

Time Period	Downloads
this version	16 015 664
yesterday	57 243
last 7 days	293 855
all time	129 598 185

Versions (29)

- 2.1.1** Apr 26, 2023 [View](#) [Depend](#) [Info](#)
- 2.1.0** Apr 26, 2023 [View](#) [Depend](#) [Info](#)
- 2.0.0** Sep 08, 2020 [View](#) [Depend](#) [Info](#)
- 2.0.0-rc.0** Jan 21, 2020 [View](#) [Depend](#) [Info](#)
- 1.9.0** Sep 08, 2020 [View](#) [Depend](#) [Info](#)

[Show All Versions](#)

Dependencies (0)

rebar.config

```
{decimal, "2.1.1"} Edit
```

erlang.mk

```
dep_decimal = hex 2.1.1 Edit
```

Checksum

```
53cfe5f497ed0e7771ae... Edit
```

Build Tools

mix

Owners

-  ericmj
-  wojtekmach

Publisher

-  ericmj

Dependents (210)

- jason, ecto, postgrex, poison,
- absinthe, logger_json,
- open_api_spex,
- ex_json_schema, number,
- elixir_schemer, and many more...

hex.pm/packages/decimal

Downloads

56000
42000
28000
14000
0

Last 30 days, all versions

16 015 664
this version

57 243
yesterday

293 855
last 7 days

129 598 185
all time

Versions (29)

2.1.1 Apr 26, 2023 [View](#) [Depends](#) [Commits](#)

2.1.0 Apr 26, 2023 [View](#) [Depends](#) [Commits](#)

2.0.0 Sep 08, 2020 [View](#) [Depends](#) [Commits](#)

2.0.0-rc.0 Jan 21, 2020 [View](#) [Depends](#) [Commits](#)

1.9.0 Sep 08, 2020 [View](#) [Depends](#) [Commits](#)

Show All Versions

Dependencies (0)

rebar.config

```
{decimal, "2.1.1"} Edit
```

erlang.mk

```
dep_decimal = hex 2.1.1 Edit
```

Checksum

```
53cfe5f497ed0e7771ae... Edit
```

Build Tools

mix

Owners

ericmj

wojtekmach

Publisher

ericmj

Dependents (210)

jason, ecto, postgresql, poison,
absinthe, logger_json,
open_api_spex,
ex_json_schema, number,
elixir_schemify, elixir_schemify

hex.pm/packages/decimal

rebar.config

```
{decimal, "2.1.1"}
```

erlang.mk

```
dep_decimal = hex 2.1.1
```

Checksum

```
53cfe5f497ed0e7771ae
```

Build Tools

mix

Owners

ericmj

wojtekmach

Publisher

ericmj

Dependents (210)

jason, ecto, postgresql, poison,
absinthe, logger_json,
open_api_spex,
ex_json_schema, number,

The chart displays the number of downloads for the 'decimal' package over the last 30 days. The y-axis represents the number of downloads, ranging from 0 to 56,000. The x-axis represents time. The data shows significant fluctuations, with major peaks around 56,000 and troughs around 10,000. A legend indicates the data represents 'Last 30 days, all versions'.

Period	Downloads
this version	16 015 664
yesterday	57 243
last 7 days	293 855
all time	129 598 185

Versions (29)

2.1.1 Apr 26, 2023

2.1.0 Apr 26, 2023

2.0.0 Sep 08, 2020

2.0.0-rc.0 Jan 21, 2020

1.9.0 Sep 08, 2020

Show All Versions

Dependencies (0)

Hexdocs

hex.pm/packages/decimal

rebar.config

```
{decimal, "2.1.1"}
```

erlang.mk

```
dep_decimal = hex 2.1.1
```

Checksum

```
53cfe5f497ed0e7771ae
```

Build Tools

mix

Owners

ericmj

wojtekmach

Publisher

ericmj

Dependents (210)

jason, ecto, postgrex, poison,
absinthe, logger_json,
open_api_spex,
ex_json_schema, number,

Downloads

56000
42000
28000
14000
0

56000
42000
28000
14000
0

16 015 664
this version

57 243
yesterday

293 855
last 7 days

129 598 185
all time

Last 30 days, all versions

Versions (29)

2.1.1 Apr 26, 2023

2.1.0 Apr 26, 2023

2.0.0 Sep 08, 2020

2.0.0-rc.0 Jan 21, 2020

1.9.0 Sep 08, 2020

Hex Diff

Show All Versions

Dependencies (0)

hex.pm/packages/decimal

Downloads

56000
42000
28000
14000
0

16 015 664 this version
57 243 yesterday
293 855 last 7 days
129 598 185 all time

Versions (29)

2.1.1 Apr 26, 2023 [View](#) [Depends](#) [Hex Preview](#)

2.1.0 Apr 26, 2023 [View](#) [Depends](#) [Hex Preview](#)

2.0.0 Sep 08, 2020 [View](#) [Depends](#) [Hex Preview](#)

2.0.0-rc.0 Jan 21, 2020 [View](#) [Depends](#) [Hex Preview](#)

1.9.0 Sep 08, 2020 [View](#) [Depends](#) [Hex Preview](#)

Show All Versions

Dependencies (0)

rebar.config

```
{decimal, "2.1.1"} Edit
```

erlang.mk

```
dep_decimal = hex 2.1.1 Edit
```

Checksum

```
53cfe5f497ed0e7771ae... Edit
```

Build Tools

mix

Owners

ericmj
 wojtekmach

Publisher

ericmj

Dependents (210)

jason, ecto, postgresql, poison,
absinthe, logger_json,
open_api_spex,
ex_json_schema, number,
elixir_schemify, elixir_schemify

A screenshot of a web browser window displaying the Hex v0.21 release blog post. The browser has a dark mode theme. The address bar shows the URL: hex.pm/blog/hex-v021-released#diff-amp-depend. The page header includes the Hex logo, a search bar with the placeholder "Find packages", and navigation links for "Packages", "Pricing", "Docs", and "Log in".

Hex v0.21 released

15 January, 2021 · by Wojtek Mach

Hex v0.21 adds registry self-hosting, diff & dependencies improvements, `mix hex.sponsor` task, and more!

Self-hosting

Hex ships with a new `mix hex.registry` task to easily build a local Hex registry. See the new [self-hosting guide](#) for more information.

Diff & dependencies

The new release also brings many improvements to better understand and manage dependencies in your Mix projects.

The screenshot shows a web browser window with the URL `hex.pm/docs/self-hosting`. The title of the page is "Building the registry". The content discusses the introduction of the `mix hex.registry build` task in Hex v0.21, which provides an easy way to build a local registry. It lists three requirements for the task: the name of the registry, a directory to hold public files, and a private key used to sign the registry. It then provides a terminal command to create an "acme" registry, generate a private key, and build the registry.

Building the registry

Hex v0.21 introduced the `mix hex.registry build` task which provides an easy way to build a local registry.

`mix hex.registry build` needs three things:

- the name of the registry
- a directory to hold public files
- a private key used to sign the registry.

Let's create an "acme" registry, generate a random private key, a `public` directory, and finally let's build the registry:

```
$ mkdir acme
$ cd acme
$ openssl genrsa -out private_key.pem
$ mkdir public
$ mix hex.registry build public --name=acme --private-key=private_key.pem
* creating public/public_key
* creating public/tarballs
* creating public/names
* creating public/versions
```



```
$ mix hex.outdated
```

```
$ mix hex.outdated
```

<u>Dependency</u>	<u>Current</u>	<u>Latest</u>	<u>Status</u>
aws_signature	0.3.2	0.3.2	Up-to-date
bandit	1.5.7	1.5.7	Up-to-date
brotli	0.3.2	0.3.2	Up-to-date
bypass	2.1.0	2.1.0	Up-to-date
ex_doc	0.34.2	0.34.2	Up-to-date
ezstd	1.1.0	1.1.0	Up-to-date
finch	0.17.0	0.19.0	Update possible
jason	1.3.0	1.4.4	Update possible
mime	2.0.6	2.0.6	Up-to-date
nimble_csv	1.2.0	1.2.0	Up-to-date
plug	1.16.1	1.16.1	Up-to-date

Run `mix hex.outdated APP` to see requirements for a specific dependency.

To view the diffs in each available update, visit:

<https://hex.pm/l/3LFLr>

\$ █



```
$ mix hex.outdated
```

<u>Dependency</u>	<u>Current</u>	<u>Latest</u>	<u>Status</u>
aws_signature	0.3.2	0.3.2	Up-to-date
bandit	1.5.7	1.5.7	Up-to-date
brotli	0.3.2	0.3.2	Up-to-date
bypass	2.1.0	2.1.0	Up-to-date
ex_doc	0.34.2	0.34.2	Up-to-date
ezstd	1.1.0	1.1.0	Up-to-date
finch	0.17.0	0.19.0	Update possible
jason	1.3.0	1.4.4	Update possible
mime	2.0.6	2.0.6	Up-to-date
nimble_csv	1.2.0	1.2.0	Up-to-date
plug	1.16.1	1.16.1	Up-to-date

Run `mix hex.outdated APP` to see requirements for a specific dependency.

To view the diffs in each available update, visit:

<https://hex.pm/l/3LFLr>

```
$
```

diff.hex.pm/diffs?diffs[]={finch:0.17.0:0.19.0&diffs[]={jason}}

Package Name	From	To	
finch	0.17.0	0.19.0	VIEW DIFF
jason	1.3.0	1.4.4	VIEW DIFF

The screenshot shows a web browser window with the following details:

- Address Bar:** diff.hex.pm/diff/finch/0.17.0..0.19.0
- Page Title:** hexdiff
- Section Headers:** finch, 0.17.0 → 0.19.0
- File Type:** CHANGED CHANLOG.md
- Content:** A diff view of the CHANLOG.md file comparing version 0.17.0 and 0.19.0. The changes are color-coded: red for deleted text, green for added text, and grey for unchanged text.

```
@@ -1,5 +1,34 @@
1 1 # Changelog
2 2
3 + ## v0.19.0 (2024-09-04)
4 +
5 + ### Enhancements
6 +
7 + - Update @mint_tls_opts in pool_manager.ex #266
8 + - Document there is no backpressure on HTTP2 #283
9 + - Fix test: compare file size instead of map #284
10 + - Finch.request/3: Use improper list and avoid Enum.reverse #286
11 + - Require Mint 1.6 #287
12 + - Remove castore dependency #274
13 + - Fix typos and improve language in docs and comments #285
14 + - fix logo size in README #275
15 +
```

A screenshot of a web browser window displaying a blog post from the Hex website. The browser has standard OS X-style controls at the top. The address bar shows the URL: hex.pm/blog/hex-v10-released-and-the-future-of-. The main content area features the Hex logo (a colorful hexagon icon) and the word "hex". A search bar with the placeholder "Find packages" and a magnifying glass icon is positioned above the main heading. To the right of the search bar are navigation links for "Packages", "Pricing", "Docs", and "Log in".

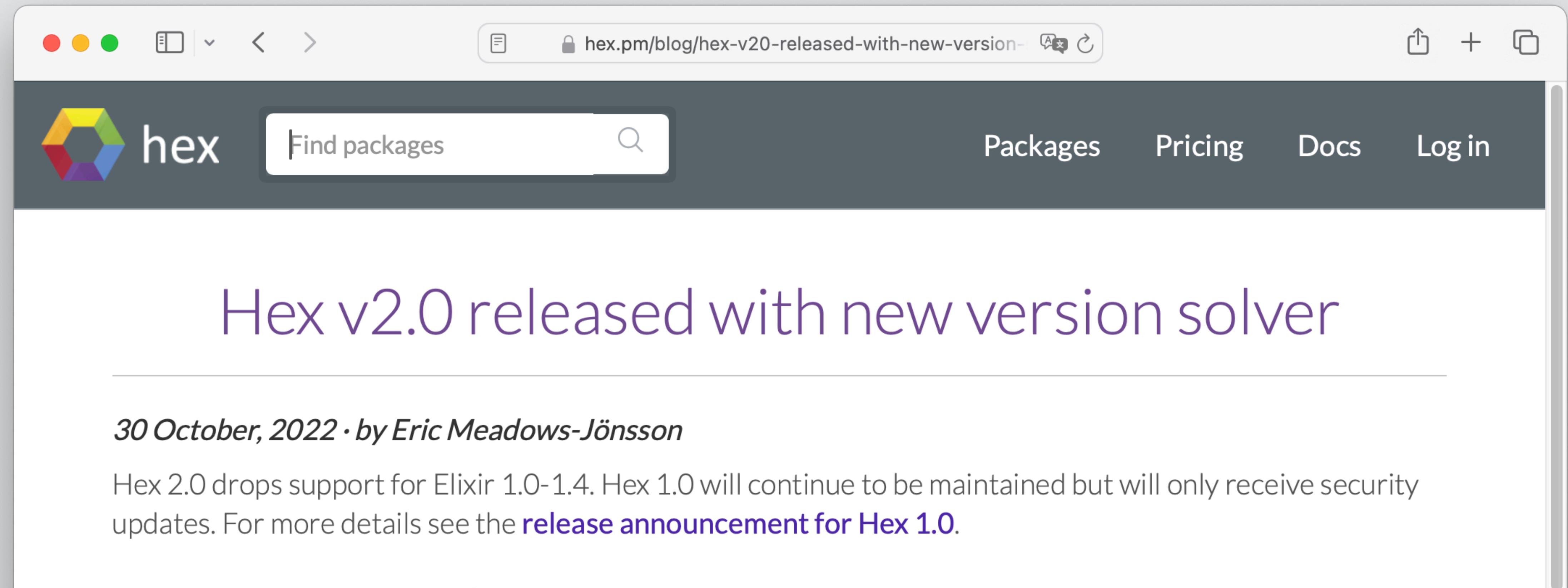
Hex v1.0 released and the future of Hex

13 December, 2021 · by Eric Meadows-Jönsson

We just released Hex 1.0.0 with no major changes compared to the last release 0.21.0 and we will soon release Hex 2.0.0, again with no major changes. Let's talk about why we are doing this and what Hex being version 1.0.0 or 2.0.0 means.

Ensuring backwards compatibility

Backwards compatibility in a package manager is very important. Hex has so far maintained backwards compatibility with Elixir 1.0.0 with every new release of Hex. The reason for this is because Hex needs to be kept up to date to continue working, otherwise things like certificates or changing APIs means things will break. If Hex stopped working on a given Elixir version it would effectively mean that the Elixir version becomes unusable for any real development. So far we have kept a single development track for Hex, meaning new features and security fixes are applied to the same branch.



A screenshot of a web browser window displaying the Hex v2.0 release announcement. The browser has a dark mode interface with a light gray header bar. The address bar shows the URL hex.pm/blog/hex-v20-released-with-new-version-solver. The main content area features a large purple title "Hex v2.0 released with new version solver". Below the title is a date and author "30 October, 2022 · by Eric Meadows-Jönsson". A paragraph explains the support drop for Elixir 1.0-1.4 and links to the Hex 1.0 release announcement. A section titled "New version solver" discusses the introduction of the hex_solver. The footer contains standard navigation links like Packages, Pricing, Docs, and Log in.

Hex v2.0 released with new version solver

30 October, 2022 · by Eric Meadows-Jönsson

Hex 2.0 drops support for Elixir 1.0-1.4. Hex 1.0 will continue to be maintained but will only receive security updates. For more details see the [release announcement for Hex 1.0](#).

New version solver

The biggest change in 2.0 is the introduction of our new version solver [hex_solver](#). A version solver for a package manager needs to be able to find the latest versions of a set of compatible packages that do not violate the user specified version requirements of each package, it needs to finish in a reasonable amount of time, and if version solving failed it needs to display a clear message that explains why it failed.

The current solver algorithm has been around since work began on Hex in early 2014. Many improvements have been made to it since its introduction such as performance improvements and better error messages but

Builds

A screenshot of a GitHub repository page for "hexpm/bob".

The page title is "hexpm/bob".

The README file content includes:

- Bob the builder**
- Bob performs automated tasks for the Elixir and Hex projects.
- Elixir builds**
- Elixir builds are compiled on each git push to <https://github.com/elixir-lang/elixir> for any branch. After the build completes it will be available at <https://builds.hex.pm/builds/elixir/{REF}.zip> where {REF} is the git ref for that push. Examples of URLs are:
 - <https://builds.hex.pm/builds/elixir/main.zip>
 - <https://builds.hex.pm/builds/elixir/v1.12.3.zip>
- These Elixir builds will be compiled against the oldest supported OTP version to ensure maximum compatibility for all users. We also build Elixir for every officially supported OTP version, if possible always use an Elixir compiled against the latest OTP version to get all available features in Elixir.

On the right side of the page, there is a "Languages" section showing a horizontal bar chart with the following data:

Language	Percentage
Elixir	66.9%
Dockerfile	16.4%
Shell	16.4%
HTML	0.3%

There are also links for "+ 30 contributors" and a "Languages" section.

github.com/hexpm/bob?tab=readme-ov-file#elixir-

README

Elixir docs

On git pushes documentation is built and pushed to
<https://hexdocs.pm/{APPLICATION}/{VERSION}> where {APPLICATION} is an application in the Elixir standard distribution and {VERSION} is the Elixir version, examples are:

- <https://hexdocs.pm/elixir/>
- <https://hexdocs.pm/elixir/main>
- <https://hexdocs.pm/mix/1.12.3>

Documentation tarballs are also uploaded to
<https://repo.hex.pm/docs/{APPLICATION}-{VERSION}.tar.gz>, examples are:

- <https://repo.hex.pm/docs/elixir-main.tar.gz>
- <https://repo.hex.pm/docs/mix-1.12.3.tar.gz>

Erlang builds

The screenshot shows a GitHub README page for the 'erlang' repository. The page title is 'README'. The main content section is titled 'Erlang builds'. It explains that Erlang builds on Ubuntu LTS versions are built periodically by a bot named Bob. Bob checks for new tagged releases every 15 minutes and builds any new versions it discovers. The 'master' and 'maint*' branches are built once a day. After builds complete, they are available at a URL: [https://builds.hex.pm/builds/otp/\\${ARCH}/\\${OS_VER}/\\${REF}.tar.gz](https://builds.hex.pm/builds/otp/${ARCH}/${OS_VER}/${REF}.tar.gz), where \${ARCH} is the CPU architecture, \${OS_VER} is the OS name and version, and \${REF} is the name of the git tag or branch. Supported architectures listed are amd64 and arm64. Supported OS versions listed are ubuntu-20.04.

README

Erlang builds

Erlang builds compiled on Ubuntu LTS versions are built periodically. Bob checks for new tagged releases every 15 minutes and builds any new versions it discovers. The "master" and "maint*" branches are built once a day.

After the builds complete they will be available at

[https://builds.hex.pm/builds/otp/\\${ARCH}/\\${OS_VER}/\\${REF}.tar.gz](https://builds.hex.pm/builds/otp/${ARCH}/${OS_VER}/${REF}.tar.gz)

where \${ARCH} is the CPU architecture, \${OS_VER} is the OS name and version, and \${REF} is the name of the git tag or branch.

Supported architectures:

- amd64
- arm64

Supported OS versions:

- ubuntu-20.04

github.com/hexpm/bob?tab=readme-ov-file#docke

README

Docker images

Docker images for Bob's Elixir and Erlang builds are built periodically. Bob checks for new Elixir and Erlang releases every 15 minutes and builds images for any new versions it discovers. The list of operating system distributions we base the images on can be found here:

https://github.com/hexpm/bob/blob/main/lib/bob/job/docker_checker.ex#L5.

Tagged images are never changed, that means `hexpm/erlang@22.0-alpine-3.11.2` will always target the tag `OTP-22.0` and won't update when `OTP-22.0.1` is released.

Erlang builds are found at <https://hub.docker.com/r/hexpm/erlang>, they use the versioning scheme `${OTP_VER}-${OS_NAME}-${OS_VER}` for tags. Builds for all releases since OTP 17 are provided.

Elixir builds are found at <https://hub.docker.com/r/hexpm/elixir>, they use the versioning scheme `${ELIXIR_VER}-erlang-${OTP_VER}-${OS_NAME}-${OS_VER}`. Builds for all major releases since Elixir 1.0.0 are provided. Images are built for all pairs of compatible Elixir and OTP versions.



```
$ docker run -it erlang
```



```
$ docker run -it erlang  
$ docker run -it erlang:27.1
```



```
$ docker run -it erlang  
$ docker run -it erlang:27.1  
$ docker run -it erlang:27.1.1
```



```
$ docker run -it erlang
$ docker run -it erlang:27.1
$ docker run -it erlang:27.1.1
$ docker run -it elixir:1.17.3-otp-27
```

```
$ docker run -it erlang
$ docker run -it erlang:27.1
$ docker run -it erlang:27.1.1
$ docker run -it elixir:1.17.3-otp-27
$ docker run -it hexpm/erlang:27.1.1-ubuntu-noble-20241009
```

```
$ docker run -it erlang
$ docker run -it erlang:27.1
$ docker run -it erlang:27.1.1
$ docker run -it elixir:1.17.3-otp-27
$ docker run -it hexpm/erlang:27.1.1-ubuntu-noble-20241009
$ docker run -it hexpm/elixir:1.17.3-erlang-27.1.1-ubuntu-noble-20241009
```

```
$ docker run -it erlang
$ docker run -it erlang:27.1
$ docker run -it erlang:27.1.1
$ docker run -it elixir:1.17.3-otp-27
$ docker run -it hexpm/erlang:27.1.1-ubuntu-noble-20241009
$ docker run -it hexpm/elixir:1.17.3-erlang-27.1.1-ubuntu-noble-20241009
$ curl https://hub.docker.com/v2/repositories/hexpm/erlang/tags | jq .count
```

```
$ docker run -it erlang
$ docker run -it erlang:27.1
$ docker run -it erlang:27.1.1
$ docker run -it elixir:1.17.3-otp-27
$ docker run -it hexpm/erlang:27.1.1-ubuntu-noble-20241009
$ docker run -it hexpm/elixir:1.17.3-erlang-27.1.1-ubuntu-noble-20241009
$ curl https://hub.docker.com/v2/repositories/hexpm/erlang/tags | jq .count
8386
```

```
$ docker run -it erlang
$ docker run -it erlang:27.1
$ docker run -it erlang:27.1.1
$ docker run -it elixir:1.17.3-otp-27
$ docker run -it hexpm/erlang:27.1.1-ubuntu-noble-20241009
$ docker run -it hexpm/elixir:1.17.3-erlang-27.1.1-ubuntu-noble-20241009
$ curl https://hub.docker.com/v2/repositories/hexpm/erlang/tags | jq .count
8386
$ curl https://hub.docker.com/v2/repositories/hexpm/elixir/tags | jq .count
```

```
$ docker run -it erlang
$ docker run -it erlang:27.1
$ docker run -it erlang:27.1.1
$ docker run -it elixir:1.17.3-otp-27
$ docker run -it hexpm/erlang:27.1.1-ubuntu-noble-20241009
$ docker run -it hexpm/elixir:1.17.3-erlang-27.1.1-ubuntu-noble-20241009
$ curl https://hub.docker.com/v2/repositories/hexpm/erlang/tags | jq .count
8386
$ curl https://hub.docker.com/v2/repositories/hexpm/elixir/tags | jq .count
148030
```

github.com/erlef/build-and-packaging-wg/issues/80

OTP macOS builds #80

Open wojtekmach opened this issue on Aug 12 · 20 comments

wojtekmach commented on Aug 12 · edited Member ...

I'd like to propose setting up a central place where macOS binary builds can be downloaded from.

The main use cases are:

- [asdf](#) (and similar tools) - Currently, installing OTP using this tools requires building from scratch. Which requires additional steps and there have been problems and caveats over the years:
<https://github.com/asdf-vm/asdf-erlang?tab=readme-ov-file#osx>

Assignees
No one—assign yourself

Labels
Agenda Item

Projects
None yet

github.com/erlef/otp_builds/tree/wm-initial

OTP Builds

This is a collection of community-maintained [Erlang/OTP](#) binary builds.

Supported operating systems:

- macos

Supported architectures:

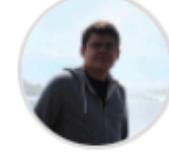
- amd64
- arm64

The goal of these builds is to be as self-contained as possible. OpenSSL (used by `crypto` app) and wxWidgets (used by `wx` app and its dependants, `observer`, `debugger`, and `et`) are statically linked.

List of Builds

+ 11 releases

Contributors 2

 **wojtekmach** Wojtek Mach

 **github-actions[bot]**

A screenshot of a GitHub release page for OTP-27.1.1. The URL in the address bar is github.com/erlef/otp_builds/releases/tag/OTP-27.1.1. The page shows the following details:

erlef / otp_builds

Code | **Issues 16** | **Pull requests 6** | **Discussions** | **Actions** | **Security** | **Insights** | **...**

Releases / OTP-27.1.1

OTP-27.1.1 Latest

 **github-actions** released this 4 days ago |  **OTP-27.1.1** |  **b5893a3**

Automated build for <https://github.com/erlang/otp/releases/tag/OTP-27.1.1>.

▼ Assets 4

	OTP-27.1.1-macos-amd64.tar.gz	57.5 MB	4 days ago
	OTP-27.1.1-macos-arm64.tar.gz	57 MB	4 days ago

A screenshot of a GitHub release page for the repository `erlef / otp_builds`. The URL in the address bar is `github.com/erlef/otp_builds/releases/tag/maint-latest`.

The page shows the `maint-latest` release, which was released by `github-actions` 6 minutes ago. The commit hash is `b5893a3`. The release notes mention an automated build for `erlang/otp@ 19ede2c`.

The `Assets` section lists four files:

Asset	Size	Created
<code>maint-macos-amd64.tar.gz</code>	57.5 MB	6 minutes ago
<code>maint-macos-arm64.tar.gz</code>	57 MB	6 minutes ago
<code>maint-linux-x86_64.tar.gz</code>	57 MB	6 minutes ago
<code>maint-windows-x86_64.zip</code>	57 MB	6 minutes ago

A screenshot of a GitHub release page for the repository `erlef / otp_builds`. The URL in the address bar is `github.com/erlef/otp_builds/releases/tag/master-latest`.

The page shows the `master-latest` release, which was created by `github-actions` 7 minutes ago. The commit hash is `b5893a3`. The release notes mention an "Automated build for [erlang/otp@ 5439fba](#)".

The `Assets` section lists four files:

Asset	Size	Created At
<code>master-macos-amd64.tar.gz</code>	57.5 MB	7 minutes ago
<code>master-macos-arm64.tar.gz</code>	57 MB	7 minutes ago
(2 more assets)		

A screenshot of a web browser window showing the GitHub release page for Elixir's main branch. The URL in the address bar is `github.com/elixir-lang/elixir/releases/tag/main-latest`. The page title is "elixir-lang / elixir". The navigation bar includes links for Code (selected), Issues (21), Pull requests (8), Actions, Wiki, Security, and Insights.

The main content area shows a release titled "main-latest" created by "github-actions" 3 weeks ago. The commit hash is "3ca7aae". A note below states "Automated release for latest main."

The "Assets" section lists two files: "Docs.zip" (6.04 MB, 8 hours ago) and "Docs.zip.sha1sum" (51 Bytes, 8 hours ago).

Asset	Size	Created
Docs.zip	6.04 MB	8 hours ago
Docs.zip.sha1sum	51 Bytes	8 hours ago

Docs

The screenshot shows a web browser displaying the Elixir documentation at hexdocs.pm/elixir/Kernel.html. The page title is "Elixir v1.17.3". The left sidebar has tabs for "PAGES" and "MODULES", with "MODULES" selected. Under "MODULES", there is a section for "Kernel" which includes "Sections" (with links to "The standard library", "Guards", "Truthy and falsy values", "Structural comparison", and "Inlining"), "Summary", "Guards", "Functions", and "Kernel.SpecialForms". Below "Kernel" is a "DATA TYPES" section with links to "Atom", "Base", "Bitwise", "Date", "DateTime", "Duration", "Exception", "Float", "Function", "Integer", "Module", "NaiveDateTime", and "Record". The main content area starts with a note about importing, followed by a section titled "The standard library" which describes the Kernel module and its capabilities. It then transitions to a section titled "Built-in types" which lists the following modules: Atom, Base, Bitwise, Date, DateTime, Duration, Exception, Float, Function, Integer, Module, NaiveDateTime, and Record. A bulleted list details the built-in data types handled by these modules.

See [import/2](#) for more information on importing.

Elixir also has special forms that are always imported and cannot be skipped. These are described in [Kernel.SpecialForms](#).

The standard library

[Kernel](#) provides the basic capabilities the Elixir standard library is built on top of. It is recommended to explore the standard library for advanced functionality. Here are the main groups of modules in the standard library (this list is not a complete reference, see the documentation sidebar for all entries).

Built-in types

The following modules handle Elixir built-in data types:

- [Atom](#) - literal constants with a name (`true`, `false`, and `nil` are atoms)
- [Float](#) - numbers with floating point precision
- [Function](#) - a reference to code chunk, created with the `fn/1` special form
- [Integer](#) - whole numbers (not fractions)
- [List](#) - collections of a variable number of elements (linked lists)
- [Map](#) - collections of key-value pairs
- [Process](#) - light-weight threads of execution

The screenshot shows a web browser displaying the Elixir documentation at hexdocs.pm/elixir/Kernel.html. The page title is "Kernel". The left sidebar lists sections under "Elixir v1.17.3" including "PAGES" and "MODULES". Under "MODULES", "Kernel" is selected, showing sections like "Sections", "Summary", "Guards", "Functions", and "Kernel.SpecialForms". Below "DATA TYPES" are listed "Atom", "Base", "Bitwise", "Date", "DateTime", "Duration", "Exception", "Float", "Function", "Integer", "Module", "NaiveDateTime", and "Record". The main content area starts with a note about importing functions from other modules. It then discusses the "The standard library" and "Built-in types". A list of built-in types is provided, each with a brief description.

See [import/2](#) for more information on importing.

import(module, opts) Elixir v1.17.3

Imports functions and macros from other modules.

The standard library

`Kernel` provides the basic capabilities the Elixir standard library is built on top of. It is recommended to explore the standard library for advanced functionality. Here are the main groups of modules in the standard library (this list is not a complete reference, see the documentation sidebar for all entries).

Built-in types

The following modules handle Elixir built-in data types:

- `Atom` - literal constants with a name (`true`, `false`, and `nil` are atoms)
- `Float` - numbers with floating point precision
- `Function` - a reference to code chunk, created with the `fn/1` special form
- `Integer` - whole numbers (not fractions)
- `List` - collections of a variable number of elements (linked lists)
- `Map` - collections of key-value pairs
- `Process` - light-weight threads of execution

```
iex> h import
```

```
iex> h import  
  
defmacro import(module, opts)  
  
Imports functions and macros from other modules.  
  
import/2 allows one to easily access functions or macros from other modules without using the qualified name.  
  
## Examples  
  
If you are using several functions from a given module, you can import those functions and reference them as local functions, for example:  
  
iex> import List  
iex> flatten([1, [2], 3])  
[1, 2, 3]  
  
## Selector
```

Docs in the Shell

Radek Szymczyszyn · [Follow](#)

3 min read · Feb 9, 2016

5

TL;DR: To access module documentation in the Erlang shell use [*docsh*](#).

Hi! I'm Radek Szymczyszyn, an [Erlang](#) dev. This is my first post here. You can also read it on [Erlang Solutions' blog](#).

For quite some time now I've been envying a few of [Elixir's](#) features: protocols, a consistent standard library, the pipe operator, macros and online documentation (as in “when connected to a system”). Personally, I don't like Elixir's syntax, but I “love” it when people use this same argument as an excuse not to use Erlang. Joking apart, some of these features could be present in Erlang, but they are not *yet*.

www.erlang.org/eeps/eep-0048.html

 ERLANG

Author: José Valim, Eric Bailey , Radek Szymczyszyn

Status: Final/24.0 Implemented in OTP version 24.0

Type: Standards Track

Created: 04-Jan-2018

EEP 48: Documentation storage and format <#>

Abstract <#>

This EEP proposes an official API documentation storage to be used by BEAM languages. By standardizing how API documentation is stored, it will be possible to write tools that work across languages.

Rationale <#>

Currently, different programming languages and libraries running on BEAM devise their own schemas for storing and accessing documentation.

For example, Elixir and LFE provide a `h` helper in their shell that can print the documentation of any module:

A screenshot of a GitHub pull request page for Erlang/OTP issue #2545. The page shows the pull request details, commit history, and a comment from the author.

The URL in the browser bar is github.com/erlang/otp/pull/2545.

The pull request title is "Introduce EEP-48 and help functions in shell #2545".

The status of the pull request is "Merged".

The author of the pull request is garazdawi, who merged 27 commits into `erlang:master` from `garazdawi:lukas/kernel/code-chunk-lookup` on Feb 24, 2020.

The pull request has 33 conversations, 27 commits, 0 checks, and 177 files changed.

A comment from garazdawi on Feb 19, 2020, states:

This PR implements generation of [EEP-48](#) style documentation and also functions in the shell to display the documentation.

This is a work in progress, but it should be functional enough for most usecases. At the moment it is not possible to create EEP-48 style documentation for anything but Erlang/OTP modules.

Generation

The documentation is built when all other Erlang/OTP documentation is built. The format for the documentation is an erlang term format using three tuples describing an HTML document. Example:

```
[{p, [], [<<"Returns an integer or float that is the arithmetical absolute value of ">>]}
```

github.com/erlang/otp/pull/2545

Merged Introduce EEP-48 and help functions in shell #2545
garazdawi merged 27 commits into erlang:master from garazdawi:lukas/kernel/code-chun... on Feb ...

Rendering

Six new functions have been introduced in the `shell_default` which means that they will be available in all shells that do not implement their of `shell_default`. The functions are `h(Module)`, `h(Module, Function)`, `h(Module, Function, Arity)`, `ht(Module)`, `ht(Module, Function)` and `ht(Module, Function, Arity)`.

The `h/1` function displays the documentation for a module. Example:

```
> h(erlang).  
  
erlang  
  
By convention, most Built-In Functions (BIFs) are included in this module. Some of the BIFs are viewed more or less as part of the Erlang programming language and are auto-imported. Thus, it is not necessary to specify the module name. For example, the calls atom_to_list(erlang) and erlang:atom_to_list(erlang) are identical.
```

The `h/2` and `h/3` functions show the function documentation. Example:

```
> h(erlang,abs,1).  
  
-spec abs(Float) -> float() when Float :: float();  
          (Int) -> non_neg_integer() when Int :: integer().  
  
Returns an integer or float that is the arithmetical absolute value of Float or Int, for example:  
  
> abs(-3.33).  
3.33
```

www.erlang.org/eeps/eep-0059.html

 ERLANG

Author: José Valim

Status: Final/27.0 Implemented in OTP version 27.0

Type: Standards Track

Created: 02-Jun-2021

EEP 59: Module attributes for documentation <#>

Abstract <#>

This EEP proposes a structured documentation API for Erlang where the documentation is handled as part of the language parser and included directly in the compiled .beam files, as a replacement for EDoc style comments. Python, Elixir, and Clojure are examples of languages that follow this approach of treating documentation as data rather than code comments.

Rationale <#>

The main limitation in EDoc today is that the documentation is kept as code comments. This requires an explicit tool to parse said code comments, which complicates access to the docs by IDEs, from the shell etc. There have been recent improvements in this area by making

This EEP proposes two new attributes: `-doc` and `-moduledoc`. They could be used as follows:

```
-module(base64).
-moduledoc "
Convenience functions for encoding and decoding from
".

-doc "
Encodes the given binary to base64.
".
-spec encode(binary()) -> binary().
encode(Binary) ->
    % ....

-doc "
Decodes the given binary from base64.
".
-spec decode(binary()) -> {ok, binary()} | error.
decode(Binary) ->
    % ....
```

The new `-moduledoc` attribute can be listed anywhere and it will contain the documentation for the given module. The `-doc` attribute must be listed anywhere before a function, type attribute or callback attribute and it will contain the documentation for the following function, type or callback. For instance, the example below:

This EEP proposes two new attributes: `-doc` and `-moduledoc`. They could be used as follows:

```
-module(base64).
-moduledoc "
Convenience functions for encoding and decoding from base64.

".
-doc "
Encodes the given binary to base64.
".
-spec encode(binary()) -> binary().
encode(Binary) ->
    % ...

-doc "
Decodes the given binary from base64.
".
-spec decode(binary()) -> {ok, binary()} | error().
decode(Binary) ->
    % ..."

The new -moduledoc attribute can be listed anywhere and it will contain the documentation for the given module. The -doc attribute must be listed anywhere before a function, type attribute or callback attribute and it will contain the documentation for the following function, type or callback. For instance, the example below:
```

A screenshot of a GitHub pull request page for EEP-59 - Documentation Attributes (#7936) on the erlang/otp repository. The page shows the pull request details, commit history, and a comment from kikofernandez.

The pull request has been merged. The commit message indicates it merged 1 commit from kikofernandez:eep48_to_markdown into erlang:master on Jan 3.

Key statistics for the pull request:

- Conversation: 10
- Commits: 1
- Checks: 18
- Files changed: 79

A comment from kikofernandez on Dec 6, 2023, discusses the implementation of documentation attributes:

Implementation of EEP-59 - Documentation Attributes

- Documentation attributes are added to the binary beam file, following format of [EEP-48](#), via `+beam_docs` compiler flag
- Warnings related to documentation attributes are dealt with in the `beam_docs.erl` instead of adding them to `erl_lint.erl`

Example 1

```
-module(warn_missing_doc).  
  
-export([test/0, test/1, test/2]).  
-export_type([test/0, test/1]).
```

github.com/erlang/otp/pull/8026

erlang / otp

Code Issues Pull requests Actions Projects Wiki

<> Code ▾ Jump to bottom

Change documentation to use ExDoc and EEP-59 style documentation #8026

Merged

garazdawi merged 30 commits into erlang:master from garazdawi:lukas/otp/convert-docs-to-markdown on Jan 31

Conversation 15 Commits 30 Checks 51 Files changed 3,258

garazdawi commented on Jan 16

This PR changes the way that Erlang/OTP is documented from using erl_docgen to ExDoc.

This is a huge PR that uses a tool to automatically convert all our documentation from XML to Markdown, trying to keep as close as possible to the original docs.

There are bound to be many small mistakes in the final docs, as the conversion is not perfect, but it is a step into what we think is an easier to maintain and evolve way of writing documentation.

The tool that was used to convert erl_docgen XML to Markdown can also be used to convert edoc XHTML to Markdown. This tool will be made available in a later PR for anyone who wants to migrate their edoc code bases to use Markdown and ExDoc

github.com/erlang/otp/pull/8026

Change documentation to use ExDoc and EEP-59 style documentation #8026

Merged

garazdawi merged 30 commits into erlang:master from garazdawi:lukas/otp/convert-docs-to-markdown on Jan 31

Conversation 15 Commits 30 Checks 51 Files changed 3,258

garazdawi comm +325,239 -487,186

This PR changes the way we generate our documentation. It uses a tool to automatically convert all our documentation from XML to Markdown, trying to keep as close as possible to the original docs.

There are bound to be many small mistakes in the final docs, as the conversion is not perfect, but it is a step into what we think is an easier to maintain and evolve way of writing documentation.

The tool that was used to convert erl_docgen XML to Markdown can also be used to convert edoc XHTML to Markdown. This tool will be made available in a later PR for anyone who wants to migrate their edoc code bases to use Markdown and ExDoc.

The screenshot shows a web browser window displaying the Erlang stdlib API Reference. The URL in the address bar is www.erlang.org/doc/apps/stdlib/api-referen. The page title is "stdlib v6.0.1". On the left, there is a sidebar with navigation links: "PAGES", "MODULES" (selected), "API Reference" (highlighted in red), "Modules", "STDLIB Application", "STDLIB Release Notes", "USER'S GUIDES", "Introduction", "The Erlang I/O Protocol", "Using Unicode in Erlang", "Uniform Resource Identifiers", "REFERENCES", and "assert.hrl". A search bar at the top right contains the placeholder "Press / to search". The main content area features a large heading "API Reference" and a section titled "Modules". Below this, several modules are listed with their descriptions:

- argparse**
Command line arguments parser.
- array**
Functional, extendible arrays.
- base64**
Provides base64 encode and decode, see [RFC 2045](#).
- beam_lib**
This module provides an interface to files created by the BEAM Compiler ("BEAM files").
- binary**
Library for handling binary data.
- c**
Command line interface module.
- calendar**
Local and universal time, day of the week, date and time conversions.
- dets**
A disk-based term storage.
- dict**
A Kev-value dictionary.

The screenshot shows a web browser window displaying the Erlang stdlib API Reference. The URL in the address bar is www.erlang.org/doc/apps/stdlib/api-referen. The page title is "stdlib v6.0.1". On the left, there's a sidebar with links for "PAGES", "MODULES", "API Reference", "USER'S GUIDES", and "REFERENCES". The "API Reference" section is currently active. A search bar at the top right contains the query "str". Below it, a modal window titled "Autocompletion results for \"str\" " lists several entries:

- str/2** FUNCTION
string
- string** MODULE
- strip/1** FUNCTION
beam_lib
- strip/2** FUNCTION
beam_lib
- strip/1** FUNCTION
string

Below the search results, the letter "c" is highlighted, followed by the text "Command line interface module." and the module "calendar".

c
Command line interface module.

calendar
Local and universal time, day of the week, date and time conversions.

dets
A disk-based term storage.

dict
A Kev-value dictionary.

Mint



HOME INSTALL LEARNING DOCS GUIDES CASES BLOG

Mint, a new HTTP client for Elixir

February 25, 2019 · by Eric Meadows-Jönsson · in [Announcements](#)

[Mint](#) is a new low-level HTTP client that aims to provide a small and functional core that others can build on top. Mint is connection based: each connection is a single struct with an associated socket belonging to the process that started the connection. Since no extra processes are started for the connection, you can choose the process architecture that better fits your application.

To validate this we built out the library with a common API supporting both HTTP/1 and HTTP/2 with automatic version negotiation. In addition, Mint comes with a [CA certificate store](#) to do safe by default HTTPS connections.

News: [Elixir v1.17 released](#)

Search...



BLOG CATEGORIES

- [Announcements](#)
- [Elixir in Production](#)
- [Internals](#)
- [Releases](#)

IMPORTANT LINKS

Mint, a new HTTP client for Elixir

February 25, 2019 · by Eric Meadows-Jönsson · in [Announcements](#)

[Mint](#) is a new low-level HTTP client that aims to provide a small and functional core that others can build on top. Mint is connection based: each connection is a single struct with an associated socket belonging to the process that started the connection. Since no extra processes are started for the connection, you can choose the process architecture that better fits your application.

To validate this we built out the library with a common API supporting both HTTP/1 and HTTP/2 with automatic version negotiation. In addition, Mint comes with a [CA certificate store](#) to do safe by default HTTPS connections.

News: [Elixir v1.17 released](#)

Search...



BLOG CATEGORIES

- [Announcements](#)
- [Elixir in Production](#)
- [Internals](#)
- [Releases](#)

IMPORTANT LINKS

Let's look at an example of sending a request with Mint:

```
iex(1)> {:ok, conn} = Mint.HTTP.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = Mint.HTTP.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = Mint.HTTP.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}
]
```

As we can see all calls to `Mint.HTTP` functions return an updated `conn` which holds the state for the connection. It is important to carry on the `conn` to the next function call or the state will be corrupted.

- Hex.pm package manager
- @elixirlang on Twitter
- #elixir on irc.libera.chat
- Elixir Forum
- Elixir on Slack
- Elixir on Discord
- IDE/Editor support
- Meetups around the world
- Jobs and hiring (community wiki)
- Events and resources (community wiki)

join the



ERLANG ECOSYSTEM FOUNDATION



Let's look at an example of sending a request with Mint:

```
iex(1)> {:ok, conn} = Mint.HTTP.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = Mint.HTTP.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = Mint.HTTP.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}
]
```

As we can see all calls to `Mint.HTTP` functions return an updated `conn` which holds the state for the connection. It is important to carry on the `conn` to the next function call or the state will be corrupted.

- Hex.pm package manager
- @elixirlang on Twitter
- #elixir on irc.libera.chat
- Elixir Forum
- Elixir on Slack
- Elixir on Discord
- IDE/Editor support
- Meetups around the world
- Jobs and hiring (community wiki)
- Events and resources (community wiki)

join the



ERLANG ECOSYSTEM FOUNDATION



Let's look at an example of sending a request with Mint:

```
iex(1)> {:ok, conn} = Mint.HTTP.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = Mint.HTTP.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = Mint.HTTP.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}
]
```

As we can see all calls to `Mint.HTTP` functions return an updated `conn` which holds the state for the connection. It is important to carry on the `conn` to the next function call or the state will be corrupted.

- Hex.pm package manager
- @elixirlang on Twitter
- #elixir on irc.libera.chat
- Elixir Forum
- Elixir on Slack
- Elixir on Discord
- IDE/Editor support
- Meetups around the world
- Jobs and hiring (community wiki)
- Events and resources (community wiki)

join the



ERLANG ECOSYSTEM FOUNDATION



Let's look at an example of sending a request with Mint:

```
iex(1)> {:ok, conn} = Mint.HTTP.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = Mint.HTTP.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = Mint.HTTP.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}
]
```

As we can see all calls to `Mint.HTTP` functions return an updated `conn` which holds the state for the connection. It is important to carry on the `conn` to the next function call or the state will be corrupted.

- Hex.pm package manager
- @elixirlang on Twitter
- #elixir on irc.libera.chat
- Elixir Forum
- Elixir on Slack
- Elixir on Discord
- IDE/Editor support
- Meetups around the world
- Jobs and hiring (community wiki)
- Events and resources (community wiki)

join the



ERLANG ECOSYSTEM FOUNDATION



Let's look at an example of sending a request with Mint:

```
iex(1)> {:ok, conn} = Mint.HTTP.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = Mint.HTTP.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = Mint.HTTP.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}
]
```

As we can see all calls to `Mint.HTTP` functions return an updated `conn` which holds the state for the connection. It is important to carry on the `conn` to the next function call or the state will be corrupted.

- Hex.pm package manager
- @elixirlang on Twitter
- #elixir on irc.libera.chat
- Elixir Forum
- Elixir on Slack
- Elixir on Discord
- IDE/Editor support
- Meetups around the world
- Jobs and hiring (community wiki)
- Events and resources (community wiki)

join the



ERLANG ECOSYSTEM FOUNDATION



Let's look at an example of sending a request with Mint:

```
iex(1)> {:ok, conn} = Mint.HTTP.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = Mint.HTTP.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = Mint.HTTP.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}
]
```

- Hex.pm package manager
- @elixirlang on Twitter
- #elixir on irc.libera.chat
- Elixir Forum
- Elixir on Slack
- Elixir on Discord
- IDE/Editor support
- Meetups around the world
- Jobs and hiring (community wiki)
- Events and resources (community wiki)



ERLNG ECOSYSTEM FOUNDATION

As we can see all calls to `Mint.HTTP` functions return an updated `conn` which holds the state for the connection. It is important to carry on the `conn` to the next function call or the state will be corrupted.



Let's look at an example of sending a request with Mint:

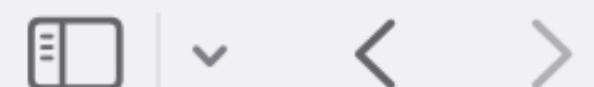
```
iex(1)> {:ok, conn} = Mint.HTTP.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = Mint.HTTP.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = Mint.HTTP.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}
]
```

- Hex.pm package manager
- @elixirlang on Twitter
- #elixir on irc.libera.chat
- Elixir Forum
- Elixir on Slack
- Elixir on Discord
- IDE/Editor support
- Meetups around the world
- Jobs and hiring (community wiki)
- Events and resources (community wiki)



ERLNG ECOSYSTEM FOUNDATION

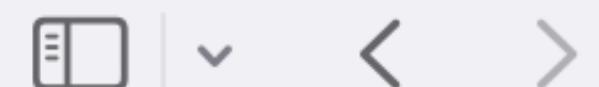
As we can see all calls to `Mint.HTTP` functions return an updated `conn` which holds the state for the connection. It is important to carry on the `conn` to the next function call or the state will be corrupted.



elixir-lang.org/blog/2019/02/25/mint-a-



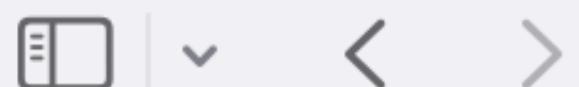
```
iex(1)> {:ok, conn} = Mint.HTTP.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = Mint.HTTP.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = Mint.HTTP.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}]
```



elixir-lang.org/blog/2019/02/25/mint-a-



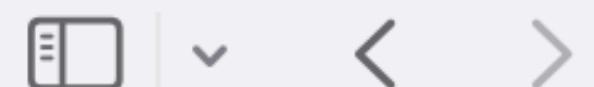
```
iex(1)> {:ok, conn} = Mint.HTTP.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = Mint.HTTP.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = Mint.HTTP.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}
]
```



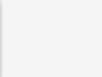
elixir-lang.org/blog/2019/02/25/mint



```
iex(1)> {:ok, conn} = GenHTTP.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = GenHTTP.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = GenHTTP.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}]
```



elixir-lang.org/blog/2019/02/25/mint-a-



```
iex(1)> {:ok, conn} = :gen_http.connect(:http, "httpbin.org", 80)
iex(2)> {:ok, conn, request_ref} = :gen_http.request(conn, "GET", "/", [], "")
iex(3)> receive do
...>   message ->
...>     IO.inspect(message, label: :message)
...>     {:ok, conn, responses} = :gen_http.stream(conn, message)
...>     IO.inspect(responses, label: :responses)
...>   end
message: {:tcp, #Port<0.8>, "HTTP/1.1 200 OK\r\n" <> ...}
responses: [
{:status, #Reference<...>, 200},
{:headers, #Reference<...>, [{"connection", "keep-alive"}, ...]},
{:data, #Reference<...>, "<!DOCTYPE html>" <> ...},
{:done, #Reference<...>}
]
```



Building a new MySQL adapter for Ecto, Part I: Hello World

• Wojtek Mach November 14th, 2018

• [ecto](#), [elixir](#), [myxql](#)

Writing a complete driver involves quite a bit of work. To name just a few things, we need to support: all protocol messages and data types, authentication schemes, connection options (TCP/SSL/UNIX domain socket), transactions and more. Rather than going through all of these in detail, I plan to distill this knowledge into 4 parts, each with a quick overview of a given area:

- [Part I: Hello World](#) (you're here!)
- [Part II: Encoding/Decoding](#)
- [Part III: DBConnection Integration](#)
- [Part IV: Ecto Integration](#)

Search with DuckDuckGo...

Subscribe to Dashbit Updates

Add your e-mail below to receive e-mails with Dashbit Updates on free ebooks, technical articles, and more. By doing so, you agree with our [Privacy Policy](#).

EMAIL

I'm not a robot



reCAPTCHA
[Privacy](#) - [Terms](#)

SUBSCRIBE

```
defmodule MyXQL.Connection do
  use DBConnection

  @impl true
  def connect(options) do
    case :gen_tcp.connect(...) do
      {:ok, socket} -> # ...
      {:error, reason} -> # ...
    end
  end

  @impl true
  def disconnect(_reason, state) do
    :gen_tcp.close(state.socket)
  end

  # ...
end
```

```
defmodule MyXQL.Connection do
  use DBConnection

  @impl true
  def connect(options) do
    case :gen_tcp.connect(...) do
      {:ok, socket} -> # ...
      {:error, reason} -> # ...
    end
  end

  @impl true
  def disconnect(_reason, state) do
    :gen_tcp.close(state.socket)
  end

  # ...
end
```

```
defmodule MyXQL.Connection do
  use DBConnection

  @impl true
  def connect(options) do
    case MyXQL.Client.connect(...) do
      {:ok, socket} -> # ...
      {:error, reason} -> # ...
    end
  end

  @impl true
  def disconnect(_reason, state) do
    MyXQL.Client.close(state.socket)
  end

  # ...
end
```

```
defmodule Postgrex.Connection do
  use DBConnection

  @impl true
  def connect(options) do
    case Postgrex.Client.connect(...) do
      {:ok, socket} -> # ...
      {:error, reason} -> # ...
    end
  end

  @impl true
  def disconnect(_reason, state) do
    Postgrex.Client.close(state.socket)
  end

  # ...
end
```

```
defmodule Postgrex.Connection do
  use DBConnection

  @impl true
  def connect(options) do
    case :gen_postgres.connect(..) do
      {:ok, socket} -> # ...
      {:error, reason} -> # ...
    end
  end
```

```
@impl true
def disconnect(_reason, state) do
  :gen_postgres.close(state.socket)
end
```

```
# ...
end
```

supabase.com/blog/supervisor-1-million

72.5K Dashboard

Supervisor: Scaling Postgres to 1 Million Connections

11 Aug 2023 • 14 minute read

Egor Romanov
Engineering

Chase Granberry
Engineering

Stanislav Muzhyk
Engineering



launch-week supervisor postgres

On this page

- What is Supervisor?
- Benchmarking 1 million connections
- Setup
- Establishing a baseline
- Supervisor's scaling capabilities
- Scaling to 1,000,000 connections
- Supervisor's impact on query duration
- Supervisor on Supabase Platform
- Getting started
- Conclusion

Share this article

X in Y

One of the most widely-discussed shortcomings of Postgres is its connection system. Every Postgres connection has a reasonably high memory footprint, and determining the maximum number of connections your database can handle is a bit of an art.

A common solution is connection pooling. Supabase currently offers pgbouncer which is single-threaded, making it difficult to scale.

We've seen some novel ways to scale pgbouncer, but we have a few

supabase.com/blog/supervisor-1-million

Product Developers Enterprise Pricing Docs Blog 72.5K Dashboard

Supervisor is a scalable, cloud-native Postgres connection pooler. It has been developed with multi-tenancy in mind, handling millions of connections without significant overhead or latency. Supervisor is built in Elixir, in partnership with [José Valim](#) (the creator of Elixir) and the [Dashbit](#) team.

The diagram illustrates the architecture of the Supervisor connection pooler. At the bottom, four client boxes (CLIENT 1, CLIENT 2, CLIENT 3, CLIENT 4) connect to a central LOAD BALANCER. The LOAD BALANCER then routes requests to an ELIXIR CLUSTER. The ELIXIR CLUSTER consists of two NODES. NODE 1 contains POOL 1 and POOL 2, while NODE 2 contains POOL 3. An INTERNAL DATABASE is also part of the cluster. Arrows show the flow of connections from clients through the load balancer to the elixir cluster, and finally to the tenants' databases (DATABASE 1, DATABASE 2, DATABASE 3).

Rustler

https://github.com/rusterlium/rustler

README **Apache-2.0 license** **License**

Rustler

[Documentation](#) | [Getting Started](#) | [Example](#)

 CI passing  hex v0.34.0  crates.io v0.34.0  last commit today

Rustler is a library for writing Erlang NIFs in safe Rust code. That means there should be no ways to crash the BEAM (Erlang VM). The library provides facilities for generating the boilerplate for interacting with the BEAM, handles encoding and decoding of Erlang terms, and catches rust panics before they unwind into C.

The library provides functionality for both Erlang and Elixir, however Elixir is favored as of now.

Features

Safety : The code you write in a Rust NIF should never be able to crash the BEAM.

Interop : Decoding and encoding rust values into Erlang terms is as easy as a function : call.

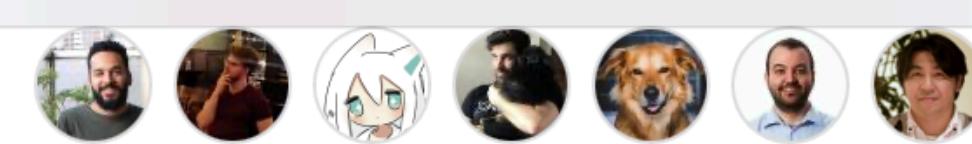
Type composition : Making a Rust struct encodable and decodable to Erlang or Elixir can be done : with a single attribute.

Resource objects : Enables you to safely pass a reference to a Rust struct into Erlang code. The : struct will be automatically dropped when it's no longer referenced.

Getting started

The easiest way of getting started is the [rustler Elixir library](#).

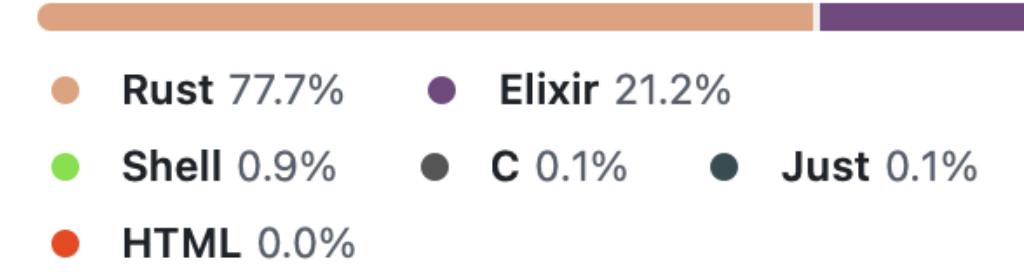
- Add the [rustler Elixir library](#) as a dependency of your project.
- Run `mix rustler new` to generate a new NIF in your project. Follow the instructions

 + 62 contributors

Deployments 1

 [github-pages](#) 8 years ago

Languages



Language	Percentage
Rust	77.7%
Elixir	21.2%
Shell	0.9%
C	0.1%
Just	0.1%
HTML	0.0%

The screenshot shows a web browser window with the URL <https://github.com/rusterlium/rustler>. The page content is as follows:

README **Apache-2.0 license** **License**

What it looks like

This is the code for a minimal NIF that adds two numbers and returns the result.

```
#[rustler::nif]
fn add(a: i64, b: i64) -> i64 {
    a + b
}

rustler::init!("Elixir.Math");
```

Minimal Supported Rust Version (MSRV)

Rustler currently has a minimal supported Rust version (MSRV) of 1.70. This is the configured version in `.clippy.toml`.

Supported OTP and Elixir Versions

Rustler aims to support the newest three major OTP versions as well as newest three minor Elixir versions.

Supported NIF version

The minimal supported NIF version for a library should be defined via Cargo features. The default is currently 2.15 (Erlang/OTP 22). To use features from NIF version 2.16 (Erlang/OTP 24) or 2.17 (Erlang/OTP 26), the respective feature flag has to be enabled on the dependency:

```
[dependencies]
rustler = { version = "...", features = ["nif_version_2_16"] }
```

https://github.com/rusterlium/rustler

What it looks like

This is the code for a minimal NIF that adds two numbers and returns the result.

```
#[rustler::nif]
fn add(a: i64, b: i64) -> i64 {
    a + b
}

rustler::init!("Elixir.Math");
```

Minimal Supported Rust Version (MSRV)

Rustler currently has a minimal supported Rust version (MSRV) of 1.70. This is the configured version in `.clippy.toml`.

Supported OTP and Elixir Versions

Rustler aims to support the newest three major OTP versions as well as newest three minor Elixir versions.

Supported NIF version

The minimal supported NIF version for a library should be defined via Cargo features. The default is currently 2.15 (Erlang/OTP 22). To use features from NIF version 2.16 (Erlang/OTP 24) or 2.17 (Erlang/OTP 26), the respective feature flag has to be enabled on the dependency:

```
[dependencies]
rustler = { version = "...", features = ["nif_version_2_16"] }
```

github.com/elixir-explorer/explorer

README MIT license

Explorer

Elixir CI passing hex.pm docs hex v0.9.2

Explorer brings series (one-dimensional) and dataframes (two-dimensional) for fast data exploration to Elixir.

Features and design

Explorer high-level features are:

elixir-explorer / explorer

Type / to search

Code Issues 16 Pull requests 2 Actions Security Insights

Releases / v0.9.2

v0.9.2

Latest

Compare ▾

philss released this Aug 27 · 21 commits to main since this release

v0.9.2 · 1175ff9 ✓

Added

- Add a new `:keep` option to the `mutate_with/3` function and `mutate/3` macro. This option allows users to control which columns are retained in the output dataframe after a mutation operation. You can use `:all` (the default) or `:none`.

Fixed

- Fix handling of "LazySeries" with remote dataframes.
- Fix typespecs of `Explorer.Series.cast/2` by adding a `dtype_alias()` type.
- Stop converting `io_dtypes()` to maps in order to preserve names ordering.

Build with attestations: <https://github.com/elixir-explorer/explorer/actions/runs/10579468339>

Contributors



philss and brendon9x

▼ Assets 15

 explorer-v0.9.2-nif-2.15-x86_64-pc-windows-gnu--legacy_cpu.dll.tar.gz	21.1 MB	Aug 27
 explorer-v0.9.2-nif-2.15-x86_64-pc-windows-gnu.dll.tar.gz	21.2 MB	Aug 27
 explorer-v0.9.2-nif-2.15-x86_64-pc-windows-msvc--legacy_cpu.dll.tar.gz	21.4 MB	Aug 27
 explorer-v0.9.2-nif-2.15-x86_64-pc-windows-msvc.dll.tar.gz	21.5 MB	Aug 27
 libexplorer-v0.9.2-nif-2.15-aarch64-apple-darwin.so.tar.gz	18.8 MB	Aug 27
 libexplorer-v0.9.2-nif-2.15-aarch64-unknown-linux-gnu.so.tar.gz	20.7 MB	Aug 27
 libexplorer-v0.9.2-nif-2.15-aarch64-unknown-linux-musl.so.tar.gz	20.2 MB	Aug 27
 libexplorer-v0.9.2-nif-2.15-x86_64-apple-darwin.so.tar.gz	21 MB	Aug 27
 libexplorer-v0.9.2-nif-2.15-x86_64-unknown-freebsd--legacy_cpu.so.tar.gz	21.7 MB	Aug 27
 libexplorer-v0.9.2-nif-2.15-x86_64-unknown-freebsd.so.tar.gz	21.8 MB	Aug 27
 libexplorer-v0.9.2-nif-2.15-x86_64-unknown-linux_gnu--legacy_cpu.so.tar.gz	21.7 MB	Aug 27

dashbit.co/blog/rustler-precompiled

Introducing Rustler Precompiled

Philip Sampaio March 15th, 2022

rustler, rust, precompilation, explorer, nerves

Rust is becoming widely used for systems programming. Because of its safety guarantees and performance, it is a good language for writing NIFs to the BEAM ecosystem, and it's fairly easy to write them using the awesome [Rustler project](#).



Rustler was created a few years ago by [Hans Elias J.](#), and it's a project that aims to be a

Get in touch

Search with DuckDuckGo...

Subscribe to Dashbit Updates

Add your e-mail below to receive e-mails with Dashbit Updates on free ebooks, technical articles, and more. By doing so, you agree with our [Privacy Policy](#).

EMAIL

I'm not a robot

reCAPTCHA

Privacy - Terms

SUBSCRIBE

Follow us

Recent Posts

Remix's concurrent submissions are fundamentally flawed (without causal ordering)

José Valim - September 12th, 2024

zigler
▼ v0.13.2

PAGES MODULES

:zigler

Zig

Summary

Functions

Zig.Formatter

EXCEPTIONS

Zig.CompileError

Zig.Type.ParseError

ZIG CODE

beam

hexdocs.pm/zigler/Zig.html

A copy icon link icon refresh icon

Zig

Inline NIF support for [Zig](#)

Motivation

Zig is a general-purpose programming language designed for robustness, optimality, and maintainability.

The programming philosophy of Zig matches up nicely with the programming philosophy of the BEAM VM and in particular its emphasis on simplicity and structure should very appealing to the practitioners of Elixir.

The following features make Zig extremely amenable to inline language support in a BEAM language:

- **simplicity.** Zig's syntax is definable in a simple YACC document and Zig takes a stance against making its featureset more complex (though it may evolve somewhat en route to 1.0)
- **Composability.** Zig is unopinionated about how to go about memory allocations. Its allocator interface is very easily able to be backed by the BEAM's, which

The screenshot shows a web browser window displaying the documentation for the `zigler` module at version `v0.13.2`. The URL in the address bar is `hexdocs.pm/zigler/Zig.html`.

The left sidebar contains a navigation tree:

- PAGES**
- MODULES**
- `:zigler`
- `Zig` (selected)

 - Summary
 - Functions

- `Zig.Formatter`
- EXCEPTIONS**
- `Zig.CompileError`
- `Zig.Type.ParseError`
- ZIG CODE**
- `beam`

The main content area has a title **Example** and contains the following Zig code:

```
defmodule BasicModule do
    use Zig, otp_app: :zigler

    ~Z"""
    pub fn add_one(number: i64) i64 {
        return number + 1;
    }
    """
end

test "basic module with nif" do
    assert 48 = BasicModule.add_one(47)
end
```

Below the code, there is a callout box with the heading **i otp_app setting**:

You should replace `:zigler` in the following example with the name of your own app. If no such app exists (e.g. you are using livebook or are in the terminal or escript), you can use `:zigler` as a fallback.

Mix.install

```
$ iex
```

```
iex(1)> 
```

```
iex(1)> Mix.install[:req]
```

```
iex(1)> Mix.install[:req]
Resolving Hex dependencies...
Resolution completed in 0.345s
New:
  finch 0.19.0
  hpax 1.0.0
  jason 1.4.4
  mime 2.0.6
  mint 1.6.2
  nimble_options 1.1.1
  nimble_pool 1.1.0
  req 0.5.6
  telemetry 1.3.0
* Getting req (Hex package)
* Getting finch (Hex package)
* Getting jason (Hex package)
* Getting mime (Hex package)
* Getting mint (Hex package)
* Getting nimble_options (Hex package)
* Getting nimble_pool (Hex package)
```

```
Compiling 10 files (.ex)
Generated jason app
==> hpax
Compiling 4 files (.ex)
Generated hpax app
==> mint
Compiling 1 file (.erl)
Compiling 20 files (.ex)
Generated mint app
==> nimble_pool
Compiling 2 files (.ex)
Generated nimble_pool app
==> finch
Compiling 14 files (.ex)
Generated finch app
==> req
Compiling 17 files (.ex)
Generated req app
:ok
iex(2)> █
```

```
iex(2)> 
```

```
iex(2)> Req.get!("
```

```
iex(2)> Req.get!("https://api.github.com/repos/wojtekmach/req")
```

```
iex(2)> Req.get!("https://api.github.com/repos/wojtekmach/req").body
```

```
iex(2)> Req.get!("https://api.github.com/repos/wojtekmach/req").body["description"]
```

```
iex(2)> Req.get!("https://api.github.com/repos/wojtekmach/req").body["description"]
"Req is a batteries-included HTTP client for Elixir."
iex(3)> █
```

```
$ erl
```

```
1> 'Elixir.Mix':install([req]).
```

```
1> 'Elixir.Mix':install([req]).  
ok  
2> █
```

```
1> 'Elixir.Mix':install([req]).  
ok  
2> {ok, Resp} = 'Elixir.Req':get(~"https://api.github.com/repos/wojtekmach/req"),  
.. #{body := #{~"description" := Description}} = Resp,  
.. Description.
```

```
1> 'Elixir.Mix':install([req]).  
ok  
2> {ok, Resp} = 'Elixir.Req':get(~"https://api.github.com/repos/wojtekmach/req"),  
#{body := #{~"description" := Description}} = Resp,  
Description.  
<<"Req is a batteries-included HTTP client for Elixir.">>  
3> █
```

```
1> 'Elixir.Mix':install([req]).  
ok  
2> {ok, Resp} = 'Elixir.Req':get(~"https://api.github.com/repos/wojtekmach/req"),  
#{body := #{~"description" := Description}} = Resp,  
Description.  
<<"Req is a batteries-included HTTP client for Elixir.">>  
3> h('Elixir.Req').■
```

3>

```
h('Elixir.Req').
```

Elixir.Req

The high-level API.

Req is composed of:

- Req - the high-level API (you're here!)
- Req.Request - the low-level API and the request struct
- Req.Steps - the collection of built-in steps
- Req.Test - the testing conveniences

The high-level API is what most users of Req will use most of the time.

Examples

more (y/n)? (y) █

```
$ cat .erlang
```

```
$ cat .erlang
Root = filename:dirname(filename:dirname(os:find_executable("elixir"))),
code:add_path(filename:join([Root, "lib", "elixir", "ebin"])),
code:add_path(filename:join([Root, "lib", "mix", "ebin"])),
code:add_path(filename:join([Root, "lib", "logger", "ebin"])).
$ █
```

```
$ cat .erlang
Root = filename:dirname(filename:dirname(os:find_executable("elixir"))),
code:add_path(filename:join([Root, "lib", "elixir", "ebin"])),
code:add_path(filename:join([Root, "lib", "mix", "ebin"])),
code:add_path(filename:join([Root, "lib", "logger", "ebin"])).
$ █
```

```
$ cat .erlang
Root = filename:dirname(filename:dirname(os:find_executable("elixir"))),
code:add_path(filename:join([Root, "lib", "elixir", "ebin"])),
code:add_path(filename:join([Root, "lib", "mix", "ebin"])),
code:add_path(filename:join([Root, "lib", "logger", "ebin"])).
$ █
```

```
$ cat .erlang
Root = filename:dirname(filename:dirname(os:find_executable("elixir"))),
code:add_path(filename:join([Root, "lib", "elixir", "ebin"])),
code:add_path(filename:join([Root, "lib", "mix", "ebin"])),
code:add_path(filename:join([Root, "lib", "logger", "ebin"])).
$
```

```
1> 'Elixir.Mix':install([telemetry]).
```

```
1> 'Elixir.Mix':install([telemetry]).  
Resolving Hex dependencies...  
Resolution completed in 0.009s  
New:  
  telemetry 1.3.0  
* Getting telemetry (Hex package)  
===> Analyzing applications...  
===> Compiling telemetry  
ok  
2> █
```

```
2> h(telemetry).  
|
```

```
2> h(telemetry).  
  
telemetry  
  
telemetry allows you to invoke certain functions whenever a particular event is emitted.  
  
For more information see the documentation for attach/4,  
attach_many/4 and execute/2.  
ok  
3> █
```



```
telemetry$ cat src/telemetry.erl
```

```
telemetry$ cat src/telemetry.erl
-module(telemetry).

-export([attach/4,
         attach_many/4,
         detach/1,
         list_handlers/1,
         execute/2,
         execute/3,
         span/3]).

-export([report_cb/1]).


-include("telemetry.hrl").


?MODULEDOC"""
`telemetry` allows you to invoke certain functions whenever a
particular event is emitted.

For more information see the documentation for `attach/4`, `attach_many/4`
```

```
telemetry$ cat src/telemetry.erl
-module(telemetry).

-export([attach/4,
         attach_many/4,
         detach/1,
         list_handlers/1,
         execute/2,
         execute/3,
         span/3]).

-export([report_cb/1]).


-include("telemetry.hrl").


?MODULEDOC("")
`telemetry` allows you to invoke certain functions whenever a
particular event is emitted.

For more information see the documentation for `attach/4`, `attach_many/4`
```



```
telemetry$ cat src/telemetry.hrl
```

```
telemetry$ cat src/telemetry.hrl
%% (...)

-if(?OTP_RELEASE >= 27).
-define(MODULEDOC(Str), -moduledoc(Str)).
-define(DOC(Str), -doc(Str)).
-else.
-define(MODULEDOC(Str), -compile()).
-define(DOC(Str), -compile()).
-endif.
telemetry$
```

```
telemetry$ cat src/telemetry.hrl
%% (...)

-if(?OTP_RELEASE >= 27).
-define(MODULEDOC(Str), -moduledoc(Str)).
-define(DOC(Str), -doc(Str)).
-else.
-define(MODULEDOC(Str), -compile()).
-define(DOC(Str), -compile()).
-endif.
telemetry$
```

```
telemetry$ cat src/telemetry.hrl
%% (...)

-if(?OTP_RELEASE >= 27).
-define(MODULEDOC(Str), -moduledoc(Str)).
-define(DOC(Str), -doc(Str)).
-else.
-define(MODULEDOC(Str), -compile([])).
-define(DOC(Str), -compile([])).
-endif.
telemetry$
```

github.com/wojtekmach/mix_install_examples

mix_install_examples Public

Pin Unwatch 26 Fork 55 Star 717

main Go to file t + Code

zkayser Add example Pacer workflow (#44) 824f7e2 · last month 111 Commits

.formatter.exs	Update .formatter.exs	3 years ago
.gitignore	Add c_libcurl.exs	2 years ago
README.md	Update README.md	2 years ago
absinthe.exs	Absinthe example (#39)	4 months ago
ash.exs	Upgrade to Ash 3.0! Add profile_b...	5 months ago
ash_json_api.exs	Add ash_json_api.exs (#41)	3 months ago
bandit.exs	Update bandit.exs (#34)	8 months ago
beam_file.exs	Add beam_file.exs	last month
benchee.exs	Add benchee.exs	3 years ago
brod.exs	Brod example (#15)	2 years ago
brotli.exs	Update brotli.exs	2 years ago
bypass.exs	Update bypass.exs	3 years ago
	Add ... (#46)	last month

About

A collection of simple Elixir scripts that are using Mix.install/2.

Readme Activity 717 stars 26 watching 55 forks

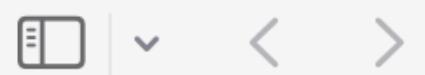
Contributors 25

Elixir 100.0%

Suggested workflows

Based on your tech stack

Livebook



localhost:58622/sessions/7wjp7f66ouzmzugoxtbsntq3pkftyd4s3k2zhir



Untitled notebook

Using My Hub ▾ workspace



Notebook dependencies and setup

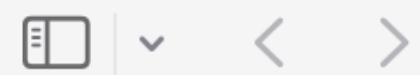
Evaluated

Section

Evaluate ▾



1 + 2



localhost:58622/sessions/7wjp7f66ouzmzugoxtbsntq3pkftyd4s3k2zhir



Untitled notebook

Using My Hub ▾ workspace



Notebook dependencies and setup

Evaluated

Section

▶ Reevaluate ▾



1 1 + 2

Evaluated

3

The screenshot shows a web-based interface for an Elixir IDE. The top bar includes standard window controls (red, yellow, green buttons) and a title bar with the URL `localhost:58622/sessions/7wjp7f66ouzmzugoxtbsntq3pkftyd4s3k2zhir`. The main area has a dark theme.

Untitled

Using

+value

[View on Hexdocs](#)

```
@spec +integer() :: integer()
@spec +float() :: float()
```

Arithmetic positive unary operator.
Allowed in guard tests. Inlined by the compiler.

Examples

```
iex> +1
1
1 1 + 2
3
```

Evaluated

Evaluated

Reeval

Sections

⋮

Star

Settings

Search

Link

Up

Down

Delete



< >

localhost:58622/sessions/7wjp7f66ouzmzugoxbsntq3pkftyd4s3k2zhir



Untitled notebook

Using My Hub ▾ workspace



Notebook dependencies and setup

Evaluated

Section

▶ Reevaluate ▾



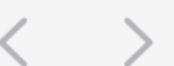
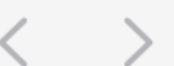
1 1 + 2 + Integer.

Evaluated*

3

f digits/1
f digits/2
f extended_gcd/2
f floor_div/2
f gcd/2
f is_even/1

Returns the greatest common divisor of the two given integers.
`Integer.gcd(integer1, integer2)`



Untitled notebook

Using My Hub ▾ workspace



Notebook dependencies and setup

Evaluated

Section

▶ Reevaluate ▾



1 1 + 2

Evaluated

3

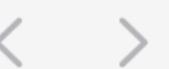
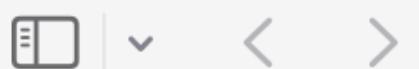
+ Elixir ▾ + Block + Smart



Elixir



Erlang



Untitled notebook

Using My Hub ▾ workspace



Notebook dependencies and setup

Evaluated

Section

1 + 2

Evaluated

3

+ Elixir ▾

+ Block

+ Smart

Chart

Data transform

Database connection

FLAME runner cell

Map

Neural Network task

Remote execution



Automate **code & data** workflows with interactive notebooks

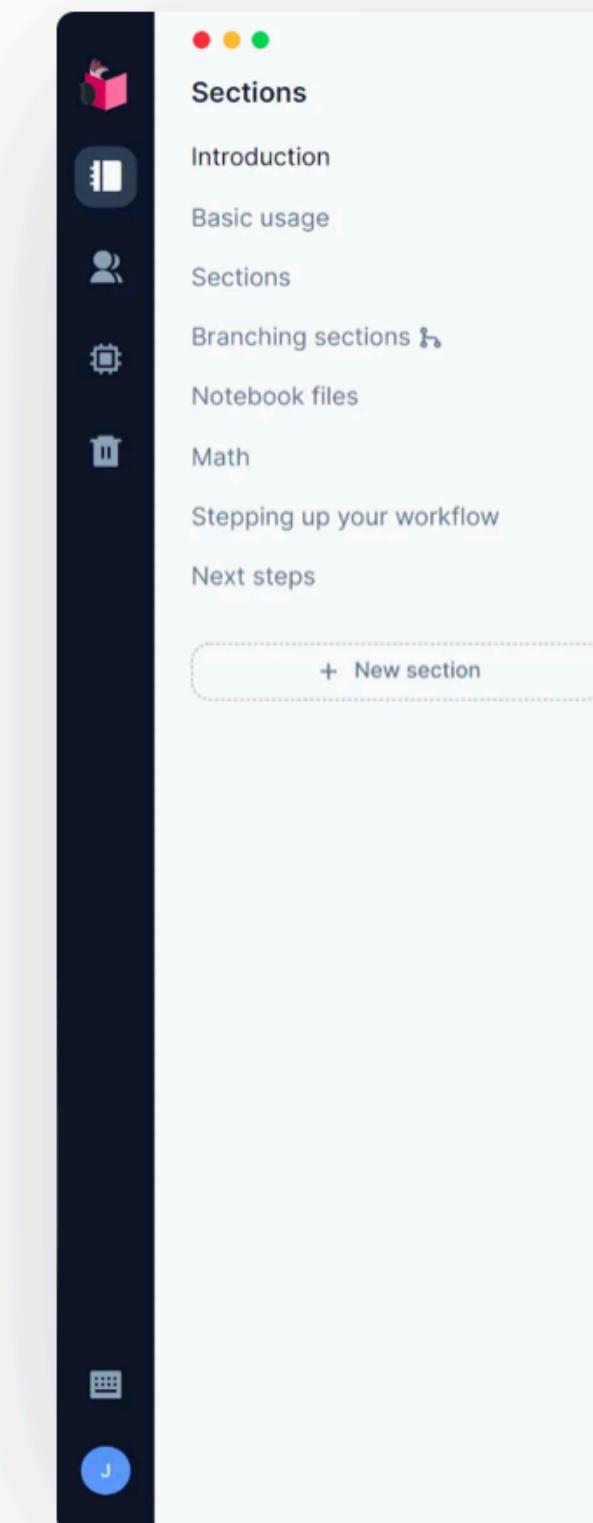
Get rid of scripts, manual steps, and outdated docs. Use Elixir and Livebook to share knowledge, deploy apps, visualize data, run machine learning models, debug systems, and more!



Install Livebook
Locally or on the cloud



Star us on GitHub
and join our community



Welcome to Livebook

Introduction

We are happy you decided to give Livebook a try, hopefully it empowers you to build great stuff! 🎉

Livebook is a tool for crafting **interactive** and **collaborative** code notebooks. It is primarily meant as a tool for rapid prototyping - think of it as an IEx session combined with your editor. You can also use it for authoring shareable articles that people can easily run and play around with. Package authors can write notebooks as interactive tutorials and include them in their repository, so that users can easily download and run them locally.

Basic usage

Each notebook consists of a number of cells, which serve as primary building blocks. There are **Markdown** cells (like this one) that allow you to describe your work and **Elixir** cells to run your code!

To insert a new cell move your cursor between cells and click one of the revealed buttons. 🌟

```
1 # This is an Elixir cell - as the name suggests that's where the code goes.
2 # To evaluate this cell, you can either press the "Evaluate" button above
3 Amy use 'Ctrl + Enter' (or Cmd + Enter on a Mac)! Jake
4 message = "hey, grab yourself a cup of ☕"
5
```

"hey, grab yourself a cup of ☕"

Subsequent cells have access to the bindings you've defined:

```
1 String.replace(message, "☕", "🍺")
```

"hey, grab yourself a cup of 🍺"

Install Livebook in just a minute

Livebook is open source, free, and ready to run anywhere.

Livebook

Home Teams Integrations What's new?

Create badge

Install Livebook in just a minute

Livebook is open source, free, and ready to run anywhere.

Run on your machine with Livebook Desktop

Mac (Universal) Windows

Run in the cloud on select platforms

Run on Hugging Face

To run on Linux, Docker, embedded devices, or Elixir's Mix, [check our README](#).

Livebook

Home Teams Integrations What's new? Create badge

Do more with Livebook

Notebooks mix prose, code, and rich output. Livebook supercharges them with awesome features:

Daily max temperatures (C) in Seattle, WA

Month

Day

Interactive insights

Visualize your data using charts, tables, maps, and diagrams. Then add inputs and controls to make it all interactive.

Explore and automate

Use Smart cells to perform high-level tasks and write notebooks faster than ever! Join our growing community to learn and master new workflows, from database queries to Machine Learning models.

CONNECT TO PostgreSQL ASSIGN TO conn

Hostname: localhost Port: 5432 Database: shop

User: postgres Password: *****

QUERY conn (PostgreSQL) ASSIGN TO result

```
1 select * from orders join users on users.id = user_id
```

[Home](#) [Teams](#) [Integrations](#) [What's new?](#) [Create badge](#)

Integrations

Livebook comes with built-in integrations with Elixir, Hugging Face, multiple data sources, data visualization libraries, and more!

[Browse Integrations](#)

Elixir
LANGUAGE



Hugging Face
MACHINE LEARNING



Slack
MESSAGING



PostgreSQL
DATABASE



MySQL
DATABASE



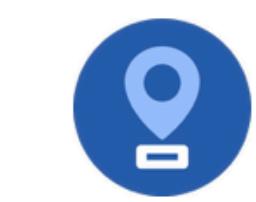
Google BigQuery
DATA WAREHOUSE



Amazon Athena
DATA WAREHOUSE



VegaLite
VISUALIZATION

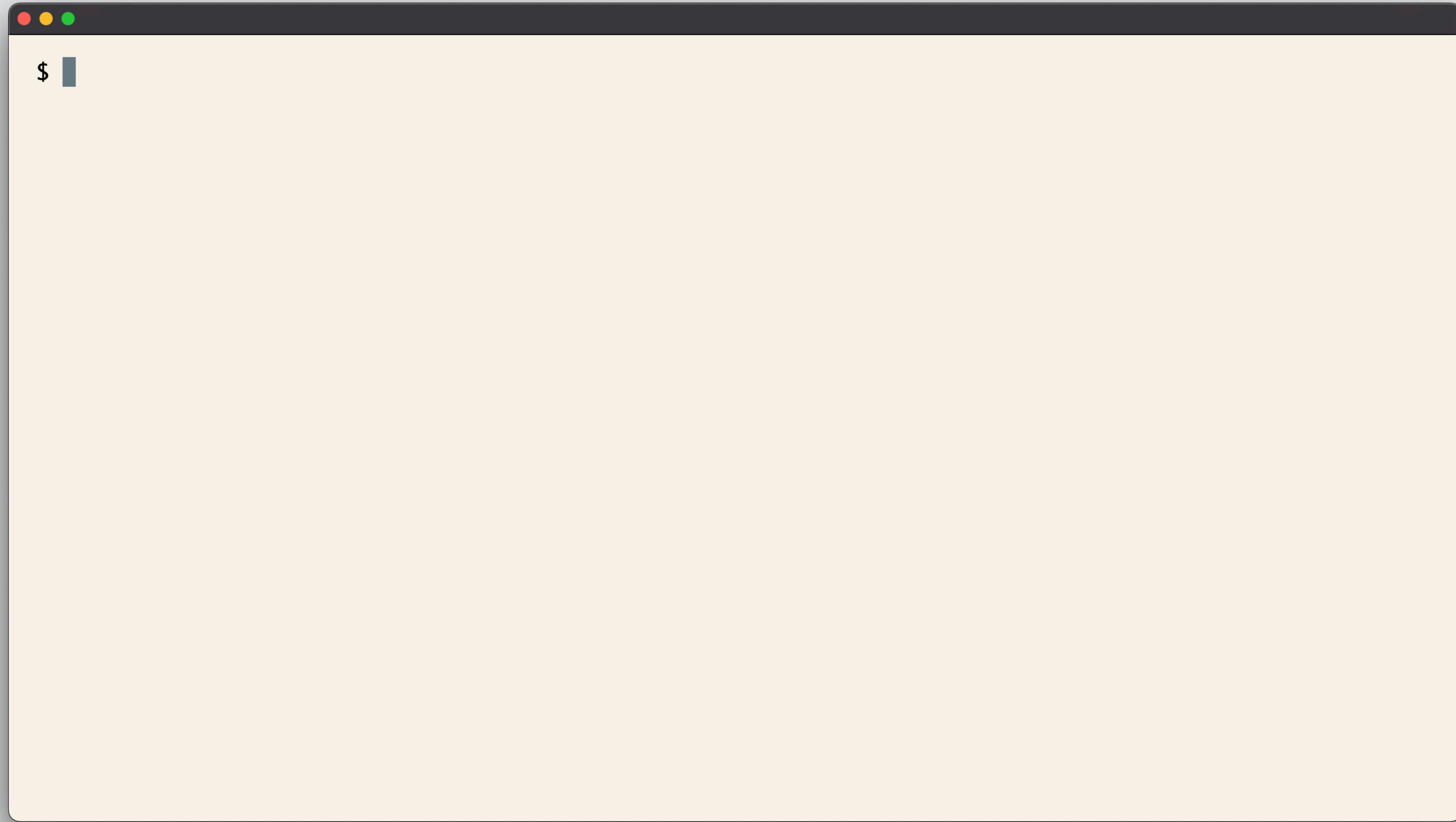


MapLibre
VISUALIZATION



See All

Elixir Install





```
$ curl -fsSL0 https://elixir-install.org/install.sh
```



```
$ curl -fsSL0 https://elixir-install.org/install.sh  
$
```



```
$ curl -fsSL0 https://elixir-install.org/install.sh  
$ sh install.sh elixir@1.17.3 otp@27.1.1
```

```
$ curl -fsSL0 https://elixir-install.org/install.sh
$ sh install.sh elixir@1.17.3 otp@27.1.1
downloading https://github.com/erlef/otp_builds/releases/download/OTP-27.1.1/OTP-27.1.1-macos-arm64.tar.gz
unpacking OTP-27.1.1-macos-arm64.tar.gz to /Users/wojtek/.elixir-install/installs/otp/27.1.1...
checking OTP... 27 ok
downloading https://github.com/elixir-lang/elixir/releases/download/v1.17.3/elixir-otp-27.zip
unpacking elixir-otp-27.zip to /Users/wojtek/.elixir-install/installs/elixir/1.17.3-otp-27...
checking Elixir... 1.17.3 ok
```

Add this to your shell:

```
export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
export PATH=$HOME/.elixir-install/installs/elixir/1.17.3-otp-27/bin:$PATH
```

\$

```
$ curl -fsSL0 https://elixir-install.org/install.sh
$ sh install.sh elixir@1.17.3 otp@27.1.1
downloading https://github.com/erlef/otp_builds/releases/download/OTP-27.1.1/OTP-27.1.1-macos-arm64.tar.gz
unpacking OTP-27.1.1-macos-arm64.tar.gz to /Users/wojtek/.elixir-install/installs/otp/27.1.1...
checking OTP... 27 ok
downloading https://github.com/elixir-lang/elixir/releases/download/v1.17.3/elixir-otp-27.zip
unpacking elixir-otp-27.zip to /Users/wojtek/.elixir-install/installs/elixir/1.17.3-otp-27...
checking Elixir... 1.17.3 ok
```

Add this to your shell:

```
export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
export PATH=$HOME/.elixir-install/installs/elixir/1.17.3-otp-27/bin:$PATH

$ export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
$ █
```

```
$ curl -fsSL0 https://elixir-install.org/install.sh
$ sh install.sh elixir@1.17.3 otp@27.1.1
downloading https://github.com/erlef/otp_builds/releases/download/OTP-27.1.1/OTP-27.1.1-macos-arm64.tar.gz
unpacking OTP-27.1.1-macos-arm64.tar.gz to /Users/wojtek/.elixir-install/installs/otp/27.1.1...
checking OTP... 27 ok
downloading https://github.com/elixir-lang/elixir/releases/download/v1.17.3/elixir-otp-27.zip
unpacking elixir-otp-27.zip to /Users/wojtek/.elixir-install/installs/elixir/1.17.3-otp-27...
checking Elixir... 1.17.3 ok
```

Add this to your shell:

```
export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
export PATH=$HOME/.elixir-install/installs/elixir/1.17.3-otp-27/bin:$PATH

$ export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
$ export PATH=$HOME/.elixir-install/installs/elixir/1.17.3-otp-27/bin:$PATH
$ █
```

```
$ curl -fsSL0 https://elixir-install.org/install.sh
$ sh install.sh elixir@1.17.3 otp@27.1.1
downloading https://github.com/erlef/otp_builds/releases/download/OTP-27.1.1/OTP-27.1.1-macos-arm64.tar.gz
unpacking OTP-27.1.1-macos-arm64.tar.gz to /Users/wojtek/.elixir-install/installs/otp/27.1.1...
checking OTP... 27 ok
downloading https://github.com/elixir-lang/elixir/releases/download/v1.17.3/elixir-otp-27.zip
unpacking elixir-otp-27.zip to /Users/wojtek/.elixir-install/installs/elixir/1.17.3-otp-27...
checking Elixir... 1.17.3 ok
```

Add this to your shell:

```
export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
export PATH=$HOME/.elixir-install/installs/elixir/1.17.3-otp-27/bin:$PATH

$ export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
$ export PATH=$HOME/.elixir-install/installs/elixir/1.17.3-otp-27/bin:$PATH
$ iex
```

```
$ curl -fsSL0 https://elixir-install.org/install.sh
$ sh install.sh elixir@1.17.3 otp@27.1.1
downloading https://github.com/erlef/otp_builds/releases/download/OTP-27.1.1/OTP-27.1.1-macos-arm64.tar.gz
unpacking OTP-27.1.1-macos-arm64.tar.gz to /Users/wojtek/.elixir-install/installs/otp/27.1.1...
checking OTP... 27 ok
downloading https://github.com/elixir-lang/elixir/releases/download/v1.17.3/elixir-otp-27.zip
unpacking elixir-otp-27.zip to /Users/wojtek/.elixir-install/installs/elixir/1.17.3-otp-27...
checking Elixir... 1.17.3 ok
```

Add this to your shell:

```
export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
export PATH=$HOME/.elixir-install/installs/elixir/1.17.3-otp-27/bin:$PATH

$ export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
$ export PATH=$HOME/.elixir-install/installs/elixir/1.17.3-otp-27/bin:$PATH
$ iex
Erlang/OTP 27 [erts-15.1.1] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:1] [jit]
```

```
Interactive Elixir (1.17.3) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> █
```

Σ Windows PowerShell

X + | v

PS C:\Users\wojtek> curl.exe -fsSLO https://elixir-install.org/install.bat

Σ Windows PowerShell

```
PS C:\Users\wojtek> curl.exe -fsSLO https://elixir-install.org/install.bat
PS C:\Users\wojtek> .\install.bat elixir@1.17.3 otp@27.1.1
downloading https://github.com/erlang/otp/releases/download/OTP-27.1.1/otp_win64_27.1.1.zip...
checking OTP...27 ok
downloading https://github.com/elixir-lang/elixir/releases/download/v1.17.3/elixir-otp-27.zip...
checking Elixir... 1.17.3 ok
```

Add this to your powershell:

```
$env:PATH = "$env:USERPROFILE\.elixir-install\installs\otp\27.1.1\bin;$env:PATH"
$env:PATH = "$env:USERPROFILE\.elixir-install\installs\elixir\1.17.3-otp-27\bin;$env:PATH"
```

Or cmd.exe:

```
set PATH=%USERPROFILE%\elixir-install\installs\otp\27.1.1\bin;%PATH%
set PATH=%USERPROFILE%\elixir-install\installs\elixir\1.17.3-otp-27\bin;%PATH%

PS C:\Users\wojtek> $env:PATH = "$env:USERPROFILE\.elixir-install\installs\otp\27.1.1\bin;$env:PATH"
PS C:\Users\wojtek> $env:PATH = "$env:USERPROFILE\.elixir-install\installs\elixir\1.17.3-otp-27\bin;$env:PATH"
PS C:\Users\wojtek> ie
      iex.bat
Erlang/OTP 27 [erts-15.1.1] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:1] [jit:ns]

Interactive Elixir (1.17.3) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> |
```



parallels@ubuntu-linux-2404: ~



```
$ curl -fsSL0 https://elixir-install.org/install.sh
```

```
parallels@ubuntu-linux-2404:~
```

```
$ curl -fsSLO https://elixir-install.org/install.sh
$ sh install.sh elixir@1.17.3 otp@27.1.1
downloading https://builds.hex.pm/builds/otp/arm64/ubuntu-22.04/OTP-27.1.1.tar.gz
unpacking OTP-27.1.1.tar.gz to /home/parallels/.elixir-install/installs/otp/27.1.1...
checking OTP... 27 ok
downloading https://github.com/elixir-lang/elixir/releases/download/v1.17.3/elixir-otp-27.zip
unpacking elixir-otp-27.zip to /home/parallels/.elixir-install/installs/elixir/1.17.3-otp-27...
checking Elixir... 1.17.3 ok

Add this to your shell:

export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
export PATH=$HOME/.elixir-install/installs/elixir/1.17.3-otp-27/bin:$PATH

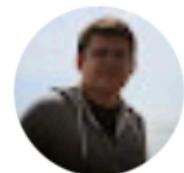
$ export PATH=$HOME/.elixir-install/installs/otp/27.1.1/bin:$PATH
$ export PATH=$HOME/.elixir-install/installs/elixir/1.17.3-otp-27/bin:$PATH
$ iex
Erlang/OTP 27 [erts-15.1.1] [source] [64-bit] [smp:2:2] [ds:2:2:10] [async-threads:1] [jit]

Interactive Elixir (1.17.3) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> █
```



```
$ sh install.sh elixir@main otp@master
```

Proposal: elixir-lang.org/install.sh



Wojtek Mach

do elixir-lang-core

13 wrz 2024, 14:06:26



Hey everyone,

We already have multiple ways of installing Elixir (<https://elixir-lang.org/install.html>) but I believe we can still do better. Elixir is cross-platform but Erlang/OTP is not. We need to get OTP for **our OS/architecture**, ideally prebuilt or otherwise we need to compile it from source. I've listed some challenges with existing installation methods here: <https://github.com/erlef/build-and-packaging-wg/issues/80>.

Since a few releases ago, Elixir project maintains installer for Windows, e.g. <<https://github.com/elixir-lang/exe/releases/download/v1.17.2/exe.exe>>, but that still requires OTP. The installer tries to be helpful, finds whether OTP is already installed and the version matches and otherwise show a link to download it. This is a GUI installer that fortunately can be running headless, `.\exe /S /D=C:\elixir`, but, again, we need to first install OTP.

I believe we can significantly improve Elixir getting started experience by having a "one click install" but for terminals, download a single script that installs Elixir and OTP for their system.

I've created a proof-of-concept called Elixir Install (<https://elixir-install.org>) and we can use it in bash for macOS/Ubuntu/Windows:

```
$ curl -fsS https://elixir-install.org/install.sh
$ sh install.sh eli...@1.17.2 o...@27.0.1
$ export PATH=$HOME/.elixir-install/install/otp/27.0.1/bin:$PATH
$ export PATH=$HOME/.elixir-install/install/elixir/1.17.2-otp-27/bin:$PATH
$ iex
```

and Powershell on Windows:

```
> curl.exe -fsS https://elixir-install.org/install.bat
> .\install.bat eli...@1.17.2 o...@27.0.1
> $env:PATH = "$env:USERPROFILE\.elixir-install\install\otp\27.0.1\bin;$env:PATH"
> $env:PATH = "$env:USERPROFILE\.elixir-install\install\elixir\1.17.2-otp-27\bin;$env:PATH"
```

A screenshot of a GitHub repository page for "beamup" by tsloughter. The page includes a navigation bar, a sidebar with "README" and "Apache-2.0 license" links, and a main content area featuring the title "BEAMUp", a "Release passing" badge, and a cartoon illustration of a pink alien in a spacesuit. The right side of the page shows deployment statistics ("+ 20 deployments") and a language distribution chart (99.0% Rust, 1.0% Other).

README Apache-2.0 license

BEAMUp

Release passing



A tool for installing languages (support for Gleam, Erlang and Elixir) that run on the [Erlang VM](#) (BEAM) and related components -- component support to come in the future.

Install

An install script is provided for both Linux/Mac:

```
curl --proto '=https' --tlsv1.2 -LsSf https://github.com/tsloughter/beamup/releases/c
```

And Windows Powershell:

Top 10 packages
(recent downloads)

The screenshot shows a web browser window displaying the hex.pm/packages page. The interface has a light gray header with navigation icons (back, forward, search, etc.) and a lock icon indicating a secure connection. The main content area lists five Elixir packages with their versions, descriptions, and recent download counts:

- jason 1.4.4**
A blazing fast JSON parser and generator in pure Elixir.
Recent downloads: 5 307 033
- telemetry 1.3.0**
Dynamic dispatching library for metrics and instrumentations
Recent downloads: 4 496 713
- mime 2.0.6**
A MIME type module for Elixir
Recent downloads: 4 311 464
- ssl_verify_fun 1.1.7**
SSL verification library
Recent downloads: 4 302 321
- ranch 2.1.0**
Socket acceptor pool for TCP protocols.
Recent downloads: 4 297 689

hex.pm/packages

jason 1.4.4

A blazing fast JSON parser and generator in pure Elixir.

5 307 033
recent downloads

telemetry 1.3.0

Dynamic dispatching library for metrics and instrumentations

4 496 713
recent downloads

mime 2.0.6

A MIME type module for Elixir

4 311 464
recent downloads

ssl_verify_fun 1.1.7

SSL verification library

4 302 321
recent downloads

ranch 2.1.0

Socket acceptor pool for TCP protocols.

4 297 689
recent downloads

hex.pm/packages

jason 1.4.4

A blazing fast JSON parser and generator in pure Elixir.

5 307 033 recent downloads

telemetry 1.3.0

Dynamic dispatching library for metrics and instrumentations

4 496 713 recent downloads

mime 2.0.6

A MIME type module for Elixir

4 311 464 recent downloads

ssl_verify_fun 1.1.7

SSL verification library

4 302 321 recent downloads

ranch 2.1.0

Socket acceptor pool for TCP protocols.

4 297 689 recent downloads

The screenshot shows a web browser window displaying the hex.pm/packages page. The interface has a light gray header with standard browser controls (back, forward, search, etc.) and a URL bar. Below the header is a navigation bar with icons for search, refresh, and other functions. The main content area displays a list of packages, each in its own card:

- parse_trans** 3.4.2 - Parse transform library. Recent downloads: 4 279 652.
- certifi** 2.13.0 - CA bundle adapted from Mozilla by <https://certifi.io>. Recent downloads: 4 224 758.
- mimerl** 1.3.0 - Library to handle mimetypes. Recent downloads: 4 215 603.
- unicode_util_compat** 0.7.0 - unicode_util compatibility library for Erlang < 20. Recent downloads: 4 163 492.
- idna** 6.1.1 - A pure Erlang IDNA implementation. Recent downloads: 4 157 159.

hex.pm/packages

parse_trans 3.4.2

Parse transform library

4 279 652 recent downloads

 certifi 2.13.0

CA bundle adapted from Mozilla by https://certifi.io

4 224 758 recent downloads

mimerl 1.3.0

Library to handle mimetypes

4 215 603 recent downloads

unicode_util_compat 0.7.0

unicode_util compatibility library for Erlang < 20

4 163 492 recent downloads

idna 6.1.1

A pure Erlang IDNA implementation

4 157 159 recent downloads

hex.pm/packages

parse_trans 3.4.2

Parse transform library

4 279 652 recent downloads

 certifi 2.13.0

CA bundle adapted from Mozilla by https://certifi.io

4 224 758 recent downloads

mimerl 1.3.0

Library to handle mimetypes

4 215 603 recent downloads

 unicode_util_compat 0.7.0

unicode_util compatibility library for Erlang < 20

4 163 492 recent downloads

idna 6.1.1

A pure Erlang IDNA implementation

4 157 159 recent downloads

**“Most people overestimate what they can do in one year
and underestimate what they can do in ten years.”**

–Bill Gates

Thank you!