Dioxus

Fullstack, crossplatform, typesafe, and blazingly fast.
Dioxus: web, desktop, and mobile in one codebase

- Inspired by React and SolidJS, based on signals, HTML and CSS
- Up-and-running in seconds with integrated hotreloading and bundling
- Executing natively across six platforms with a single codebase.

```rust
use dioxus::prelude::*;

fn main() {
    dioxus::launch(app);
}

fn app() -> Element {
    let mut count = use_signal(|| 0);

    h1 { "High-Five counter: {count}" }
    button { onclick: move |_| count += 1, "Up high!" }
    button { onclick: move |_| count -= 1, "Down low!" }
}
```
Larger ecosystem
More projects result in more libraries, docs, code, and examples.

Growing Dioxus

Improve developer productivity
Ship things like hotreloading, autoformatting, bundling, signals

Ship projects faster
Better tooling means more projects get shipped!
Strategy for increasing developer productivity:

1. Decrease quantity of boilerplate code
2. Improve reliability
3. Shorten iteration cycles
“Computer, make me an app”
A perfect Dioxus Copilot

1. Help build and improve UIs with minimal friction

2. Automatically perform quality assurance by running tests and giving feedback

3. Improve existing code in-place, fixing bugs and iterating on things like accessibility, layout, styling, and content

Automatic refactors, accessibility, styling, new pages, forms, backend, database queries, 3rd party libraries, responsiveness, and more!
A perfect Dioxus Copilot
Hard requirement: local-first

- LLMs are increasingly capable on lower-end hardware. Models like Phi-3 can run on a phone!

- Local-first LLMs save us from running any infrastructure - we want to improve Dioxus, not run servers!

- Local-first increases our “hackability” - meaning the OSS community can improve our work without much cost overhead

- We can roll out Dioxus AI via our CLI tools without requiring internet access or sending anyone’s code to our servers

**Phi-3-mini-128k-instruct**
Required mem: 1.8 GB @ 4bit

**Llama 3 8b**
Required mem: 5.7 GB @ 4bit
“Can a robot take a blank canvas and turn it into a masterpiece?”

“Can you?”
So much potential in automatic UI generation!

- Generate your team’s own component library
- Iterate on styles, content, and layout
- Generate mocks to present your app with fake user data
- Automating boring, tedious UI code like forms and login flows

See: Vercel’s v0.dev
Can plain Llama 70B generate Dioxus?

Nope, not really.

Llama has never seen Dioxus code before and gets confused when asked to generate it. Llama is still confused even when we give it our docs!

But it has seen a lot of HTML!
Can Llama 70B generate HTML?

Here is the response:

1. What should the UI look like?
   The UI should display a profile picture and a header image, responsive and accessible on various devices.

2. What are the individual components that make up the UI?
   * ProfilePictureComponent: Displays a profile picture.
   * HeaderImageComponent: Displays a header image.
   * UserProfileHeader: Takes children, contains a profile picture and header image.

3. What does the HTML look like?

```html
```UserProfileHeader
  ProfilePictureComponent />
</UserProfileHeader>
```

4. What is the HTML for each component?

```html
```
```UserProfileHeader`:

```html
```
```ProfilePictureComponent`:

```html
```
```HeaderImageComponent`:

```html
```
```UserProfileHeader`:

```html
```

Welcome to Kalosm 0.2
Kalosm is an open-source Rust framework for local machine learning
Get Started
Can Llama 70B generate HTML?

Yes!

- Doesn't always follow the output format
- Generating components that are never used, used incorrectly, or contain duplicates
- Embed javascript into JSX

But not very well...

- Generating noisy SVGs
- A correct output less than 20% of the time, and usable less than 10%
Fine-tuning to the rescue!

- Base Llama might not be too smart for our use case, but we can make it smarter!
- LoRA (low rank adaptation) makes it possible to refine even the largest of LLMs.
- We can retrain Llama using a large dataset of optimized prompts.
- We somehow need to synthesize a large, high quality dataset of prompts!
Fine Tuning using a Synthetic Dataset

1. Generate prompts
2. Filter bad responses
3. Run response optimizer
4. Save optimized prompt
5. Adjust weights
Technique: Filter out subpar responses

- Programmatically filter out low-quality responses from the baseline model
- Ensure the component can be parsed and that it'll generate an output
- Perform integrity and quality checks
- Lightly rewrite components for readability

- Can we parse the output?
- Will the component compile?
- Does it answer all the questions we ask about the UI?
- Are the components actually used?
- Are there any problematic elements?
Technique: Optimize responses with normalization

Multiple functionally identical code snippets

A single normalized HTML output
Technique: Response compression for performance

Old: 247 tokens

New: 161 tokens

35% performance boost!
UI generation accuracy jumped from 10% to 20%!

~10000 prompts generated
~8000 followed our format
~1500 passed quality filters
Inference
Inference with Kalosm

- Quantized models that work with text, audio and image data
- Constrained generation to precisely control model outputs
- Context extraction: Webpages, PDFs, RSS, or even text from live transcription
- Integrations: Headless browser, surreal DB, arroy vector database

```rust
let llm = Llama::new().await.unwrap();

// Control what the LLM can generate next with contraints
let validator = RegexParser::new(r"(\d, ){4}\d").unwrap();
// Stream the output with async rust
let stream = llm.stream_structured_text("Five prime numbers: 2, ", validator).await.unwrap();
stream.to_std_out().await.unwrap();
```

What is Kalosm?

- Fast and scalable machine learning
- Easy integration with Rust ecosystem
- Open-source and community-driven
Why Kalosm: Structured Generation

How do we *guarantee* that the model outputs code that *will* compile?

- Small models are especially prone to hallucination and “runaway” over large sequences

- Programmatically influence the output of the model depending on the state of the streaming HTML parser - Rust is really good at this! 🦀
COMPONENTS:

Constrained Generation

are 25% 0%
/

The 50% 0%
A 15% 0%
-

Check Constraints

100%
Constrained generation is powerful!

Improves model format accuracy from **20%** to **90%!**

10 times smaller model performs 5 times better

About 2s-10s per component in the response

Constrained generation is powerful!
<Generative UI Demo>
“Engineering, status report!”

“Your app could be improved, sir!”
Kalosm Agent Tools

Agents combine:

- Instructions about the task
- Constraints to follow a consistent format
- Context from the environment to ground the model’s responses in reality

[Diagram showing the process of LLM action, Tool response, and LLM response]

- search google → the top article → based on that article...
Automated Quality Assurance with Agents

- A very small language model (phi 3) can understand HTML
- Dioxus knows what elements are interactive
- Take actions on your site and flag unexpected behavior
< | Kalosm Agent Demo | >
Automatically improve code in-place

```javascript
1 fn ProgressBar() -> Element {
  2  rsx! {
  3    div {
  4      id: "percent-loaded",
  5      role: "progressbar",
  6      aria_valuenow: "{progress}"
  7      aria_valuemini: "0",
  8      aria_valuemax: "100",
  9      "{progress}"}
 10  }
 11 }
```

Improve accessibility for screen readers

```javascript
1 fn ProgressBar() -> Element {
  2  rsx! {
  3    div { class: "bg-gray-200 text-gray-700",
  4      "Loading... ",
  5  }
  6 }
```

Add darkmode and media queries
Suggested refactors

Understand the codebase and dynamically suggest VSCode “magic hints”

Perform complex refactors like:

- Hoisting state into parent components
- Adding keys and other optimizations
- Moving global state to context
- Extracting elements into components
- Organizing project files
What’s next?

- Dioxus CLI + VSCode extension integration
- Interactive UI generation
- Automated UX testing
- Fine-tuning Phi-3 and Llama 70b
- Generic implementation for other frameworks
- Shipping to production!

https://github.com/dioxuslabs/dioxus-ai
Thank you