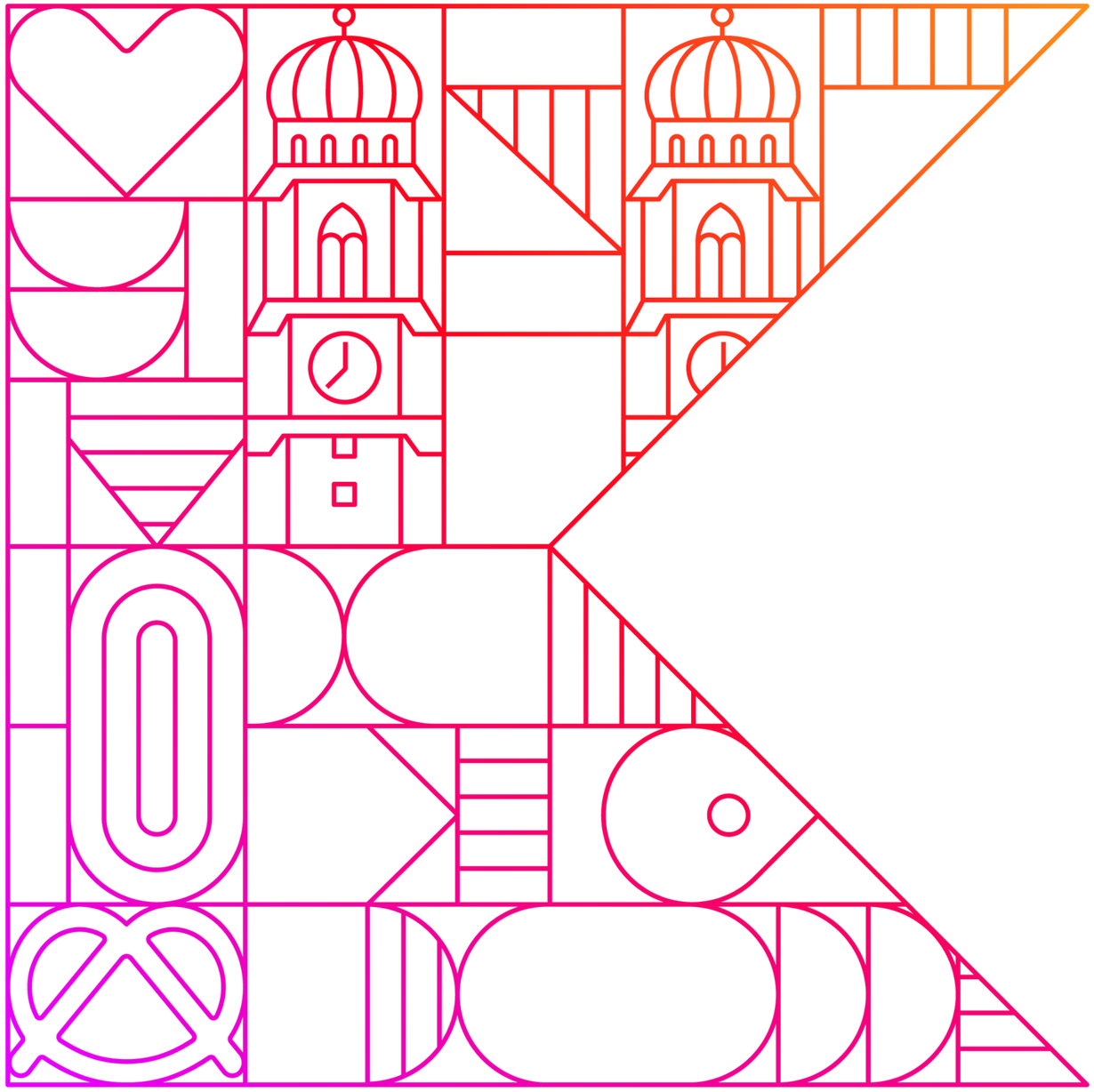
KotlinConf 2026

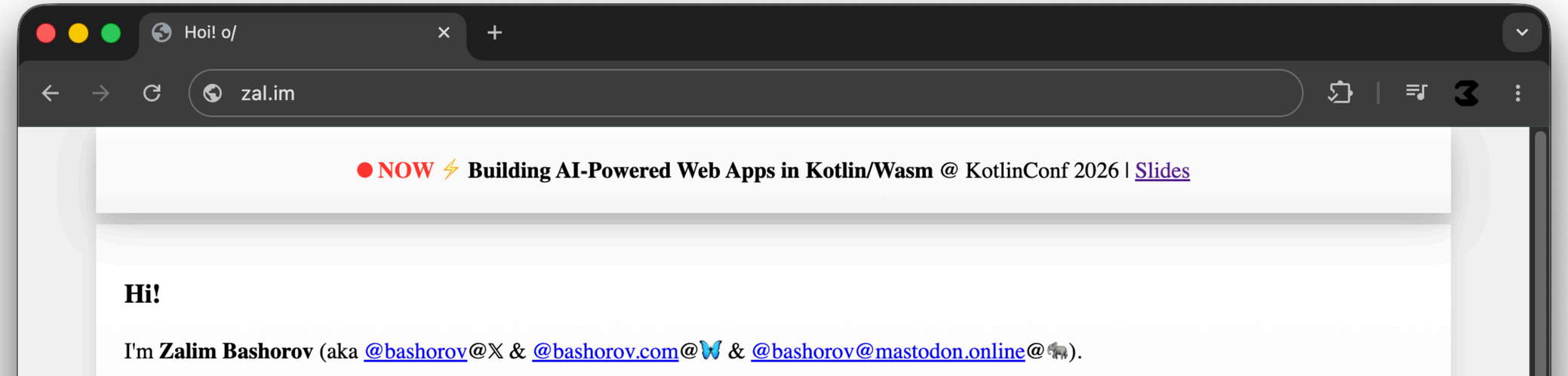
Building AI-Powered Web Apps in Kotlin/Wasm

Zalim Bashorov | @bashorov.com



⚡Slides 👇

bashorov.com | zal.im



Web Apps with Kotlin/Wasm

Why?

- Easy Distribution
- Easy to Run
- Near-Native performance **with WebAssembly**
- Kotlin & Compose Multiplatform advantages **with Kotlin/Wasm**

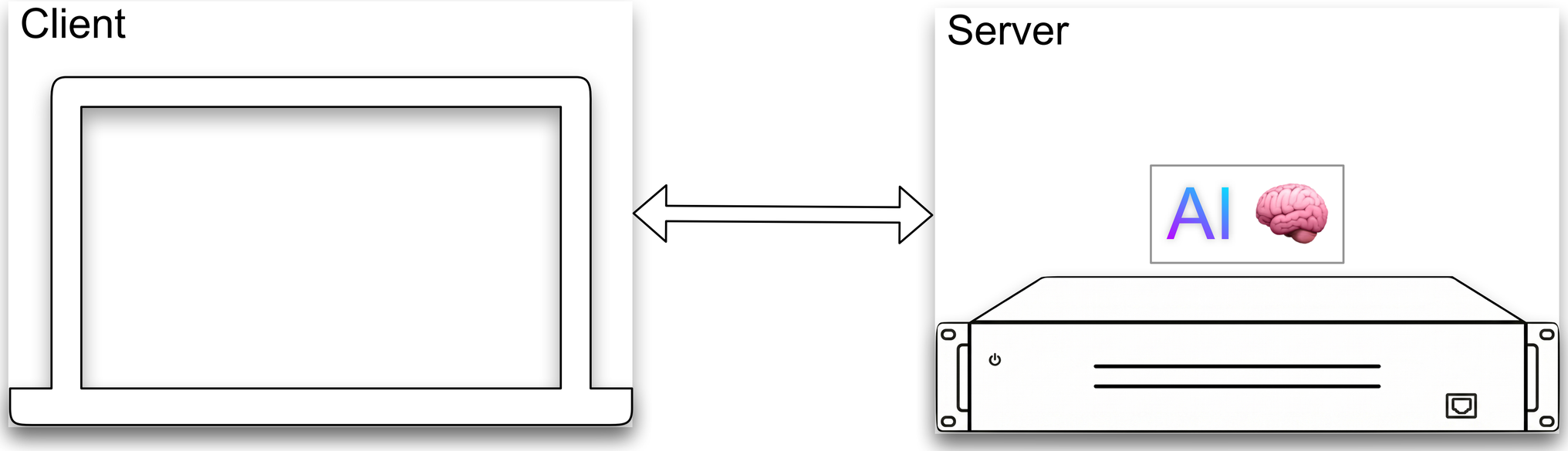
https://kotl.in/wasm
https://whatpwacando.today/

Web AI


AI

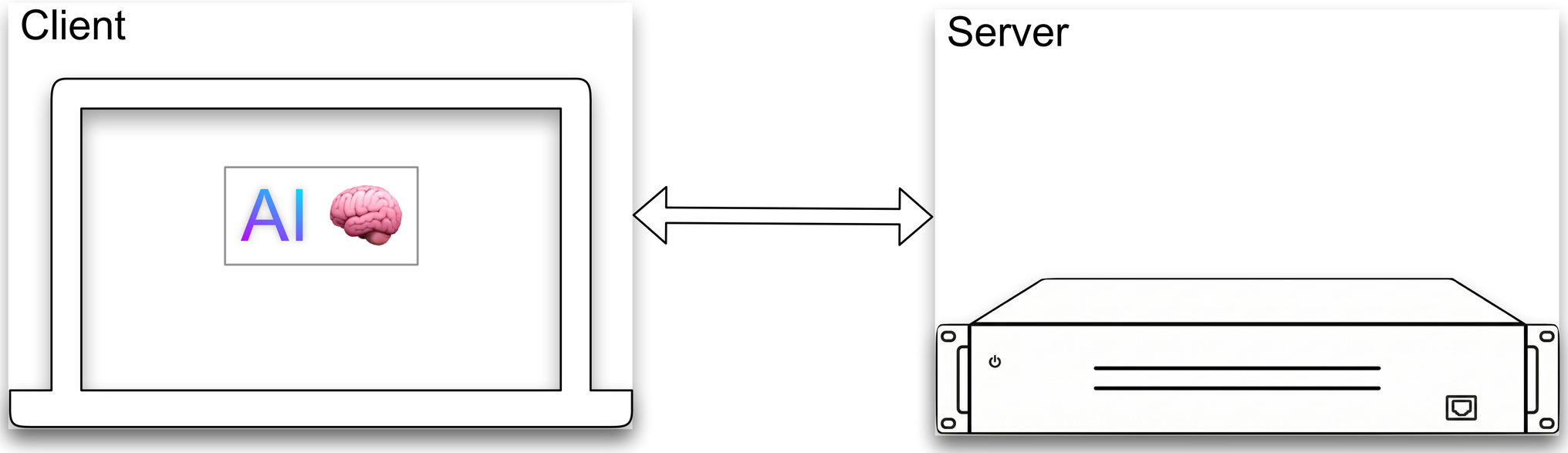
Cloud AI

(Server-side / Hosted / Remote AI)



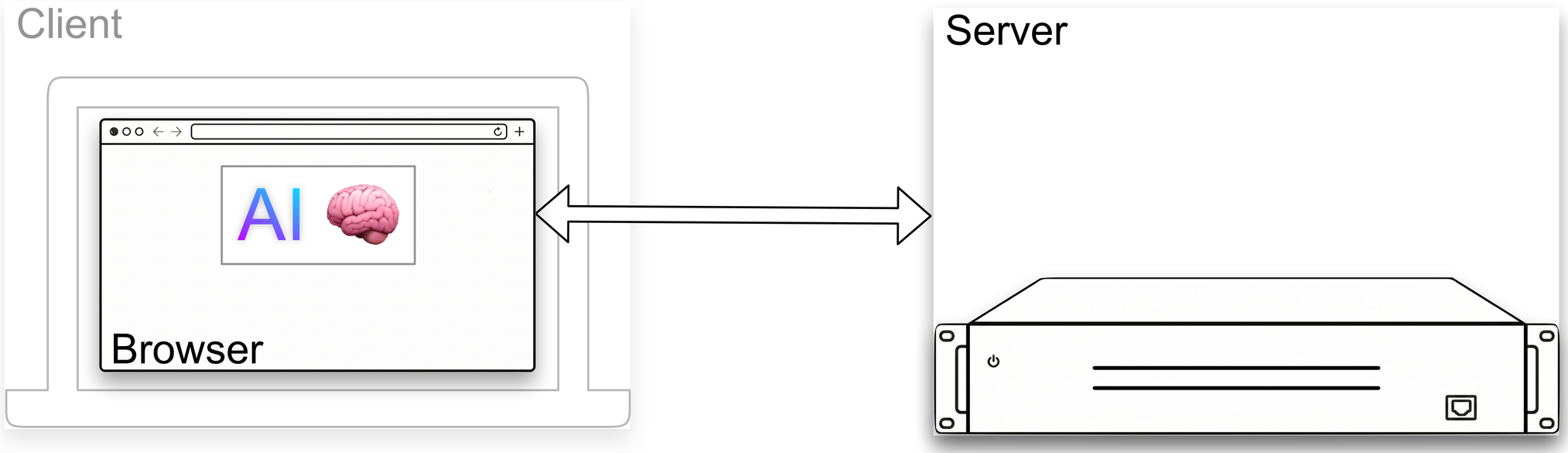
On-Device AI

(Client-side / Local AI)



Web AI

(In Browser / Built-in / Client-side AI)



https://web.dev/learn/ai/client-side
https://developer.chrome.com/docs/ai/built-in
https://web.dev/learn/ai/platform
https://learn.microsoft.com/en-us/microsoft-edge/web-platform/prompt-api
https://chrome.dev/web-ai-demos/
https://microsoftedge.github.io/Demos/built-in-ai/playgrounds/prompt-api/

Built-in AI in Browsers

Benefits:

- Privacy
- Low latency
- Offline
- Zero cost

Limitations:

- Hardware requirements
- Limited capabilities
- Evolving APIs & availability

https://developer.chrome.com/docs/ai/get-started
https://learn.microsoft.com/en-us/microsoft-edge/web-platform/prompt-api

Built-in AI in Browsers: APIs

Available*

- Translator API
- Language Detector API
- Summarizer API

Origin trial

- Proofreader API
- Prompt API

Developer trial

- Writer API
- Rewriter API

https://developer.chrome.com/docs/ai/built-in-apis
https://learn.microsoft.com/en-us/microsoft-edge/web-platform/prompt-api
https://chrome.dev/web-ai-demos/
https://microsoftedge.github.io/Demos/built-in-ai/playgrounds/prompt-api/

Translator API usage

```
val translator = Translator.create(translatorOptions("en", "de")).await()
```

```
const translator = await Translator.create({ sourceLanguage: ‘en’, targetLanguage: ‘de’ });
```

JS

https://developer.chrome.com/docs/ai/translator-api
https://learn.microsoft.com/en-us/microsoft-edge/web-platform/translator-api

Translator API availability

```
if (isTranslatorAvailable()) {
    // The Translator API is supported.
}
```


```
if ('Translator' in self) {
    // The Translator API is supported.
}
```

https://developer.chrome.com/docs/ai/translator-api
https://learn.microsoft.com/en-us/microsoft-edge/web-platform/translator-api

Translator API availability

```
if (isTranslatorAvailable()) {
    // The Translator API is supported.
}

val translatorAvailability =
    Translator.availability(translatorOptions("en", "de")).await()
```

```
if ('Translator' in self) {
    // The Translator API is supported.
}

const translatorAvailability =
    await Translator.availability({ sourceLanguage: ‘en’, targetLanguage: ‘de’ });
```

https://developer.chrome.com/docs/ai/translator-api
https://learn.microsoft.com/en-us/microsoft-edge/web-platform/translator-api

Yes, but…

Interop with JavaScript

- **`external`** modifier
- @Js... annotations
- `js(…)` function
- Libraries: kotlinx-browser, kotlin-wrappers, ...

https://kotlinlang.org/docs/wasm-js-interop.html

DEMO 🚀

...

Custom On-device AI in Browser

Solutions

- WebLLM
- Transformers.js
- ONNX Runtime Web
- LiteRT.js

https://web.dev/learn/ai/client-side#custom_deployments_with_libraries
https://webllm.mlc.ai/
https://huggingface.co/docs/transformers.js/
https://onnxruntime.ai/docs/tutorials/web/
https://ai.google.dev/edge/litert/web

Custom On-device AI in Browser

Pros:

- Works across browsers
- More control over models
- No vendor lock-in

Cons:

- Large bundle size
- Slower startup (model download)
- Harder to cache efficiently
- More setup & tuning

Key Takeaways

- Local AI in Browsers is already practical
  (Privacy, Low Latency, Offline)
- Always have a fallback strategy
- Kotlin and WebAssembly are great for Web AI
- Try it!

https://kotl.in/wasm
https://web.dev/learn/ai/client-side

Key Takeaways

- Local AI in Browsers is already practical
  (Privacy, Low Latency, Offline)
- Always have a fallback strategy
- Kotlin and WebAssembly are great for Web AI
- Try it!

https://kotl.in/wasm
https://web.dev/learn/ai/client-side

@bashorov.com

Thank You, and Don’t Forget to Vote