Avoiding SEO pitfalls in modern JavaScript frameworks.

Will Kennard
SEO Consultant

@willkennard
linkedin.com/in/willkennard

Better understand JS technology so you can talk to developers confidently & build better websites.

Today's mission.

Why should we care?

Google is good at JS.

Google is good at JS. LLMs not so much.

"

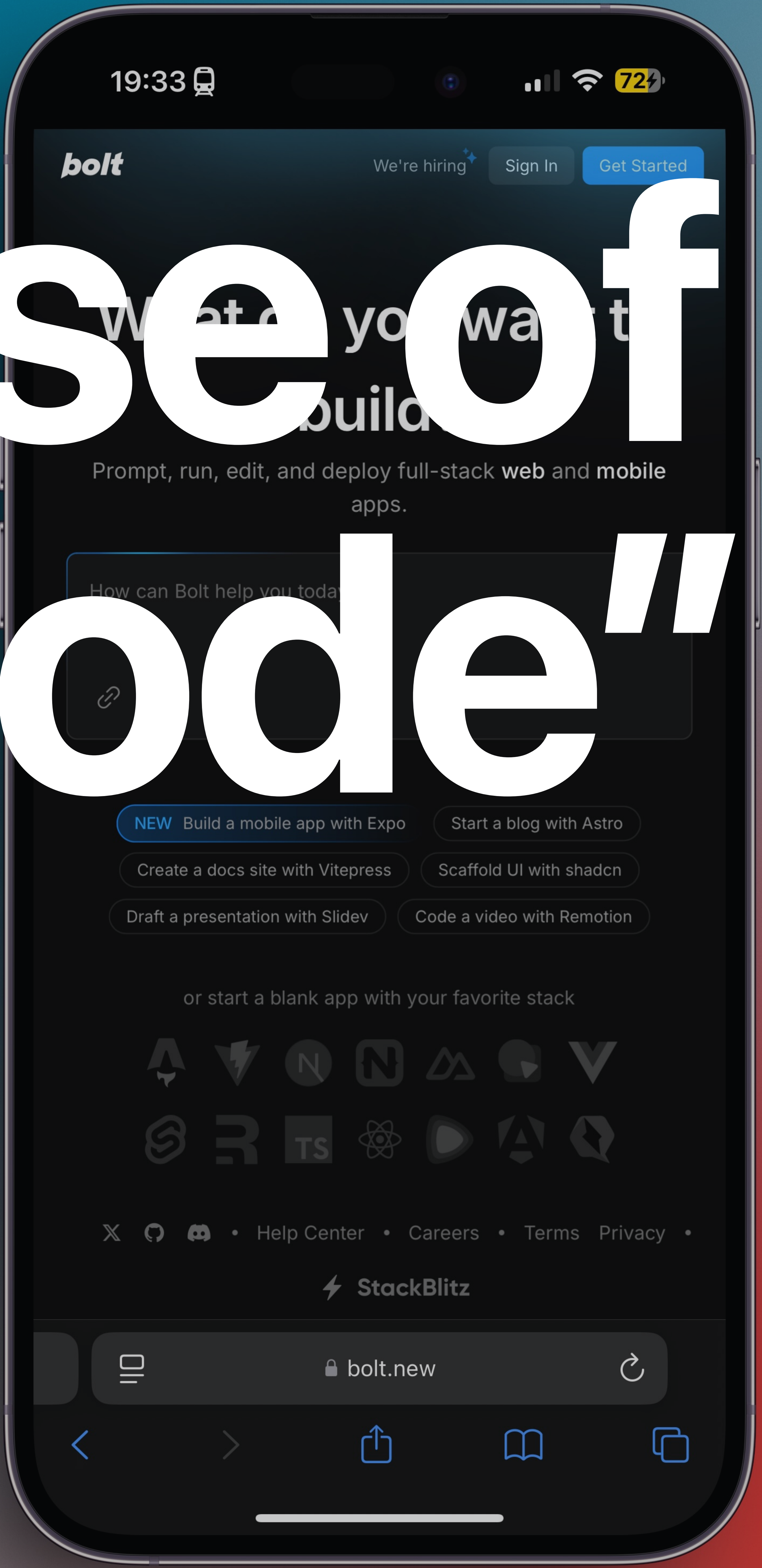
The results consistently show that none of the major AI crawlers currently render JavaScript.

Vercel

willkennard.com

It’s pretty easy to get an app up and running now.

The rise of AI "No Code" tools


19:33
bolt
We're hiring
Sign In
Get Started
Prompt, run, edit, and deploy full-stack web and mobile apps.
How can Bolt help you today
NEW Build a mobile app with Expo
Start a blog with Astro
Create a docs site with Vitepress
Scaffold UI with shadcn
Draft a presentation with Slidev
Code a video with Remotion
or start a blank app with your favorite stack
Help Center
Careers
Terms
Privacy
StackBlitz
bolt.new

willkennard.com

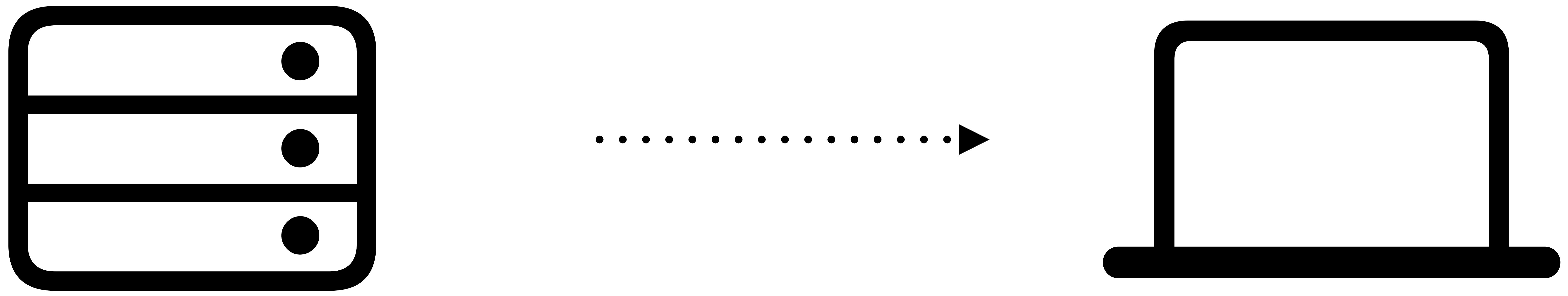
These tools favour JS frameworks and they are sure to produce a lot of issues.

SEOs are well placed to build *and* fix the web.

willkennard.com

The Web & JavaScript.

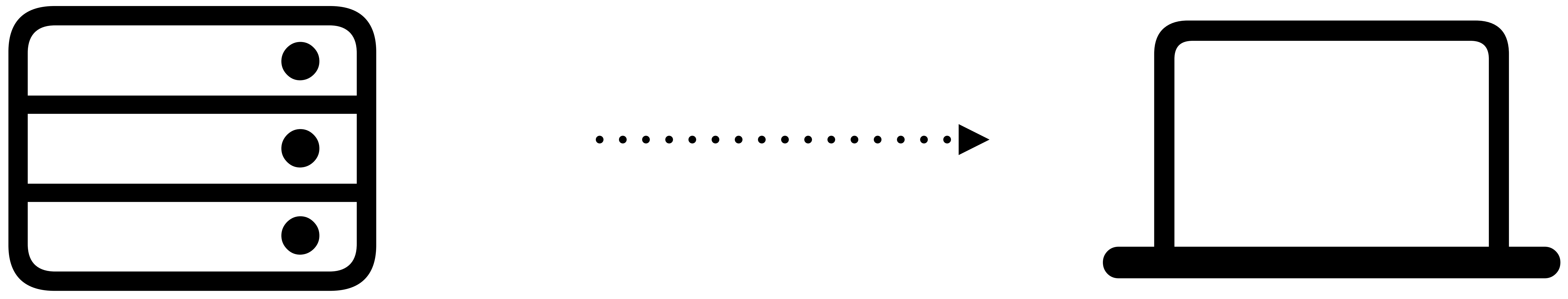
Why is JavaScript "bad" for SEO?



Routing

Data Fetching

Displaying



Routing

Data Fetching

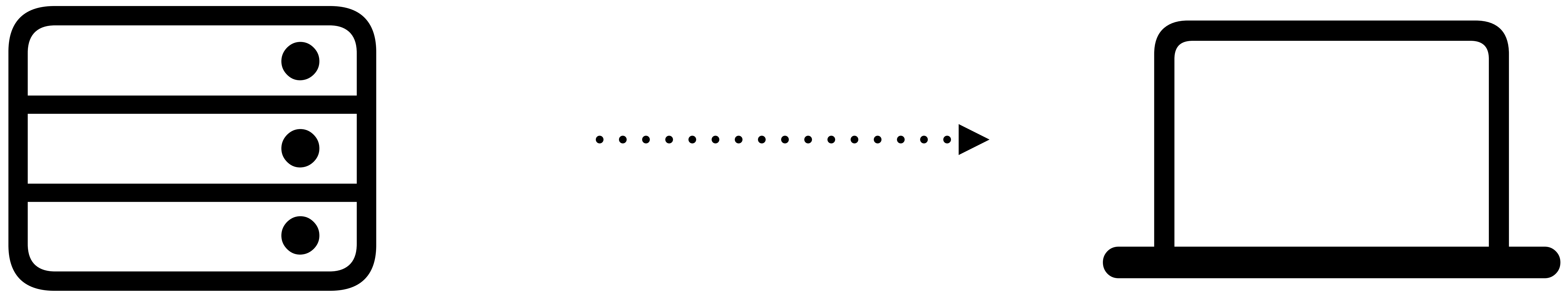
Rendering

Displaying

willkennard.com

“App like” asynchronous behaviour became a big focus.

willkennard.com



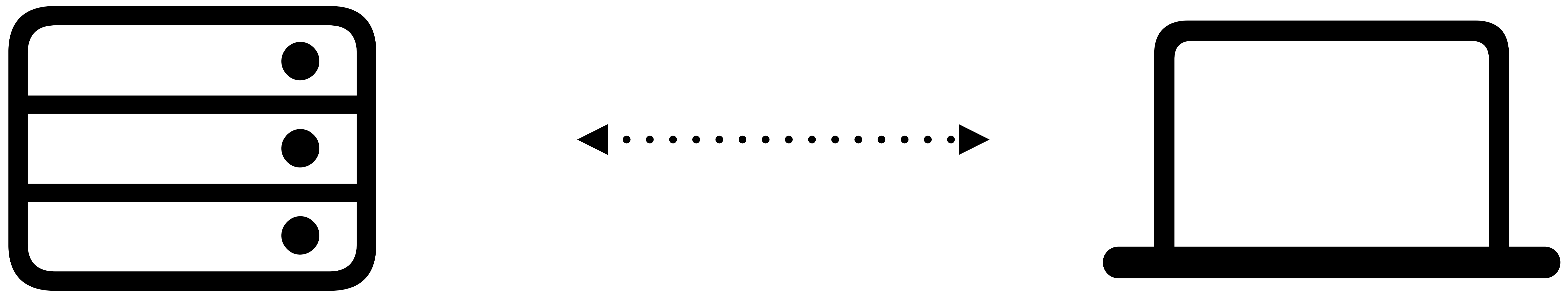
Routing

Data Fetching

Rendering

Displaying

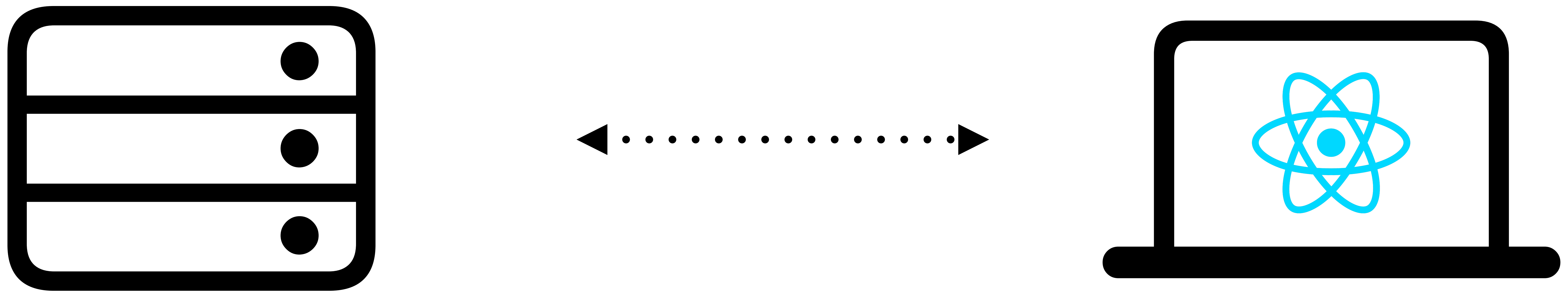
willkennard.com



- Routing
- Data Fetching
- Rendering

- Displaying
- Client Routing
- Async Fetching
- Rendering

willkennard.com



- Routing
- Data Fetching
- Rendering

- Displaying
- Client Routing
- Async Fetching
- Rendering

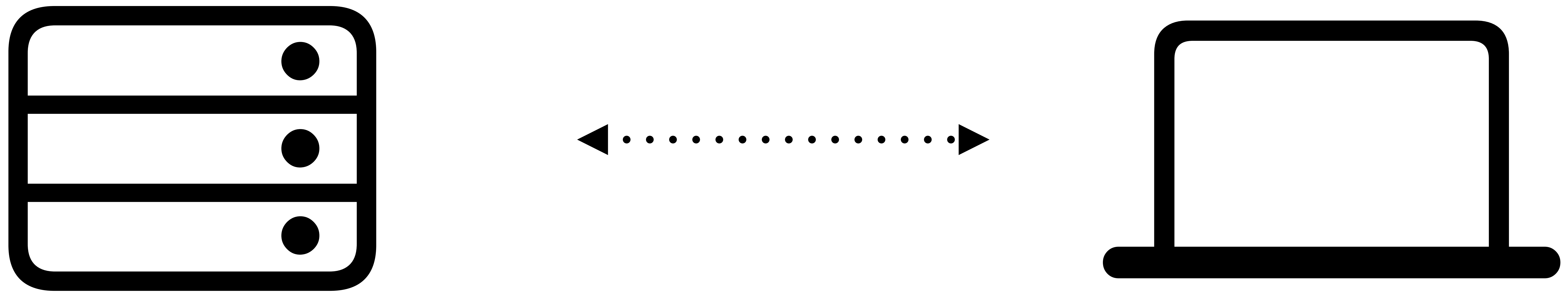
willkennard.com

JS is powerful & not inherently bad for the web.

SEOs are primarily focused on <u>crawling</u> & <u>rendering</u>.

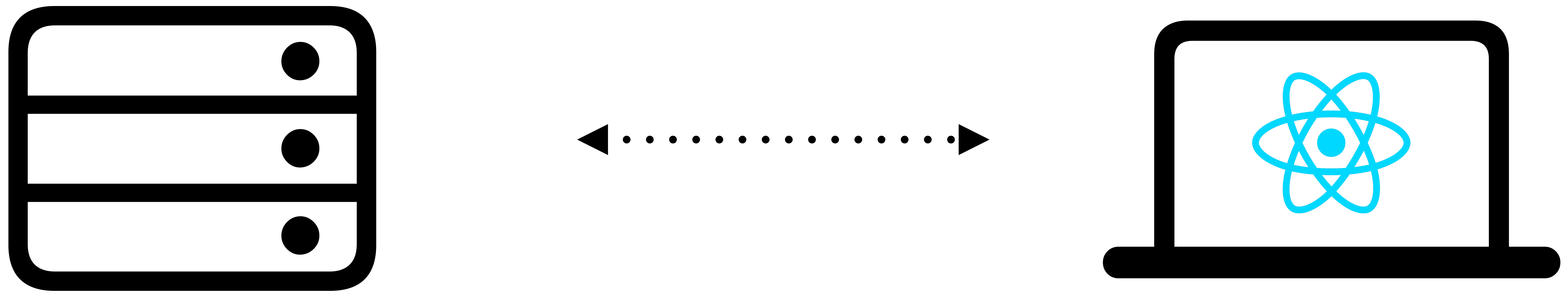
Crawling == routing issues
Rendering == client side issues

Single page apps.



- Routing
- Data Fetching
- Rendering

- Displaying
- Client Routing
- Async Fetching
- Rendering



Initial Routing

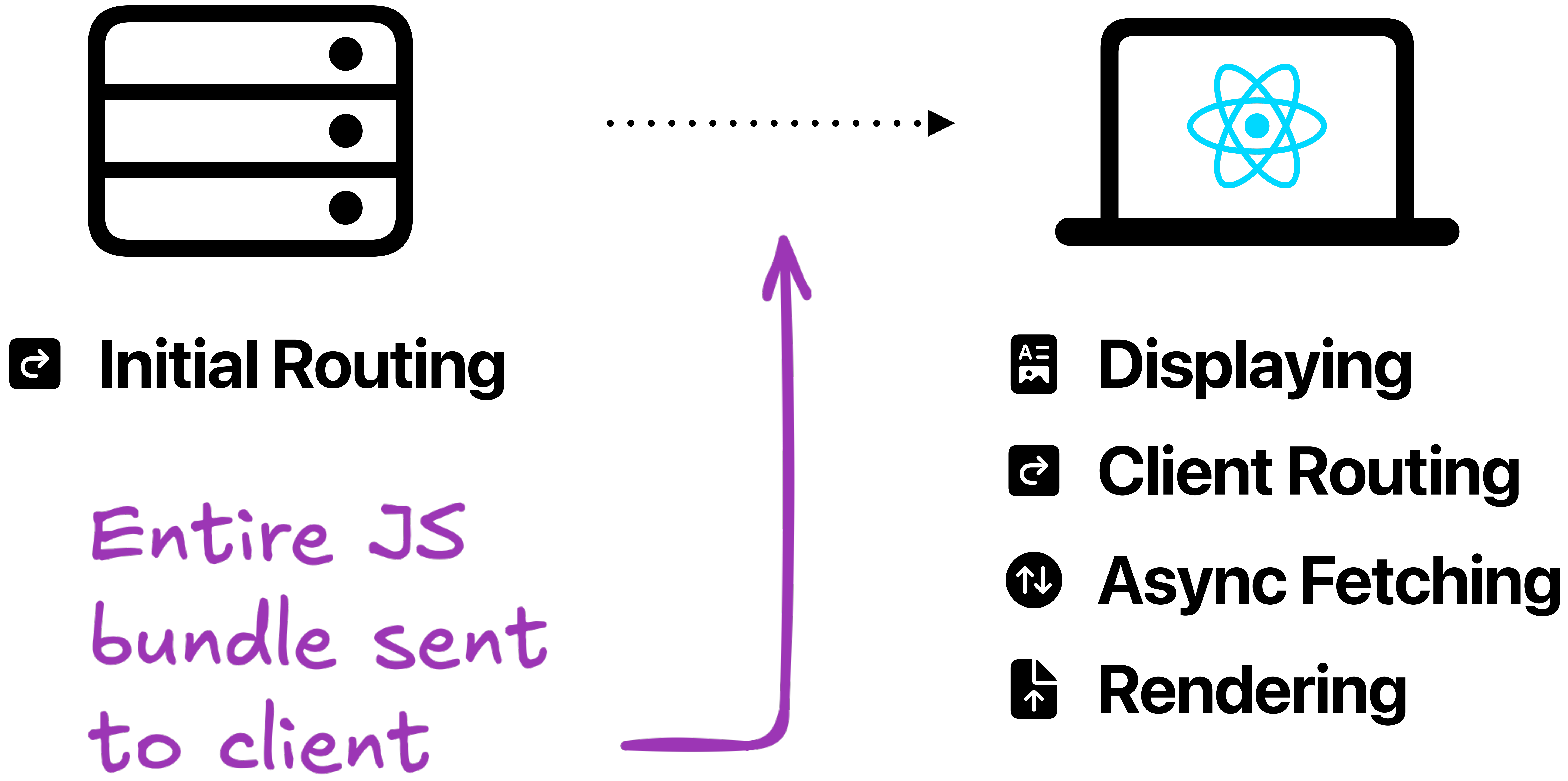
Displaying

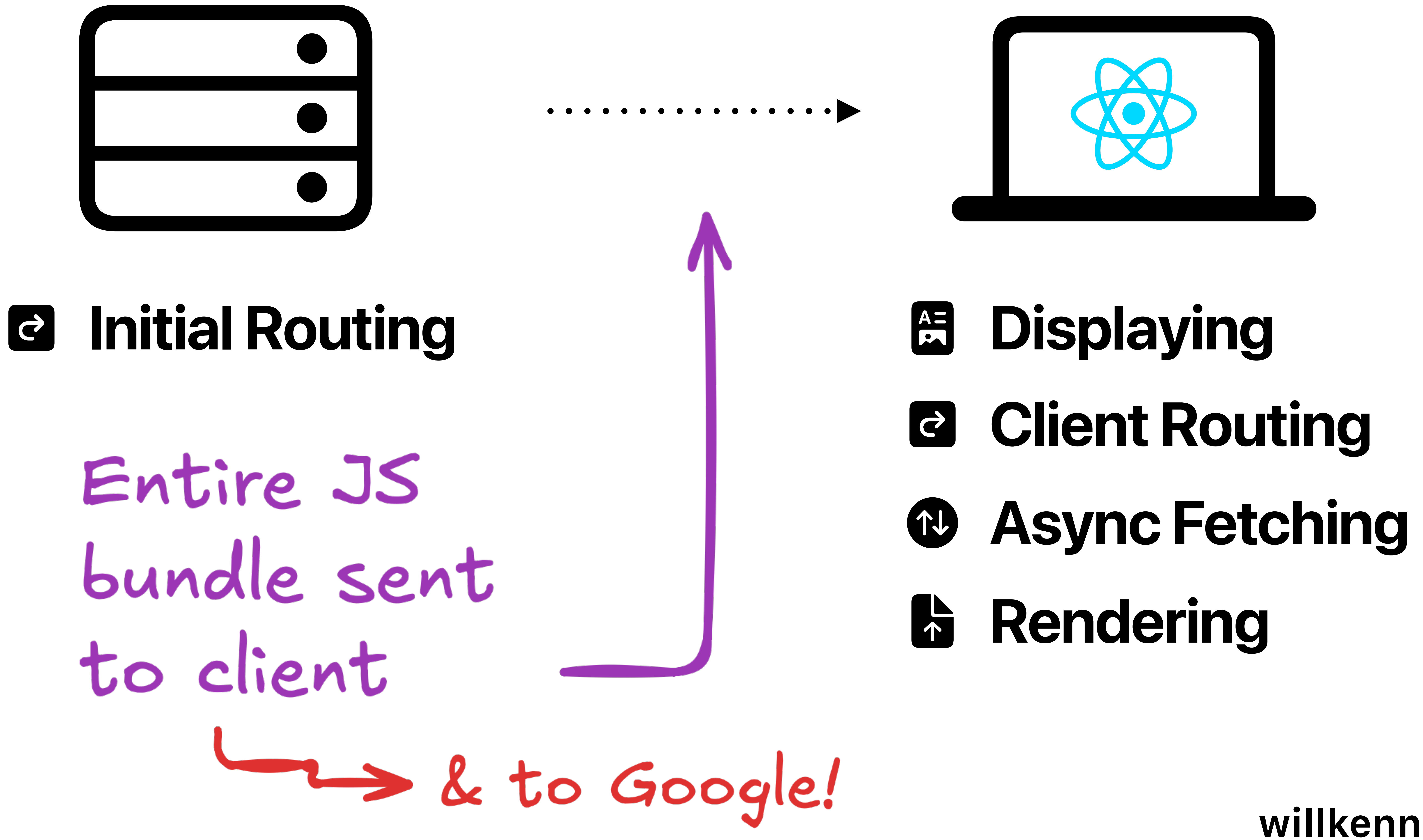
Client Routing

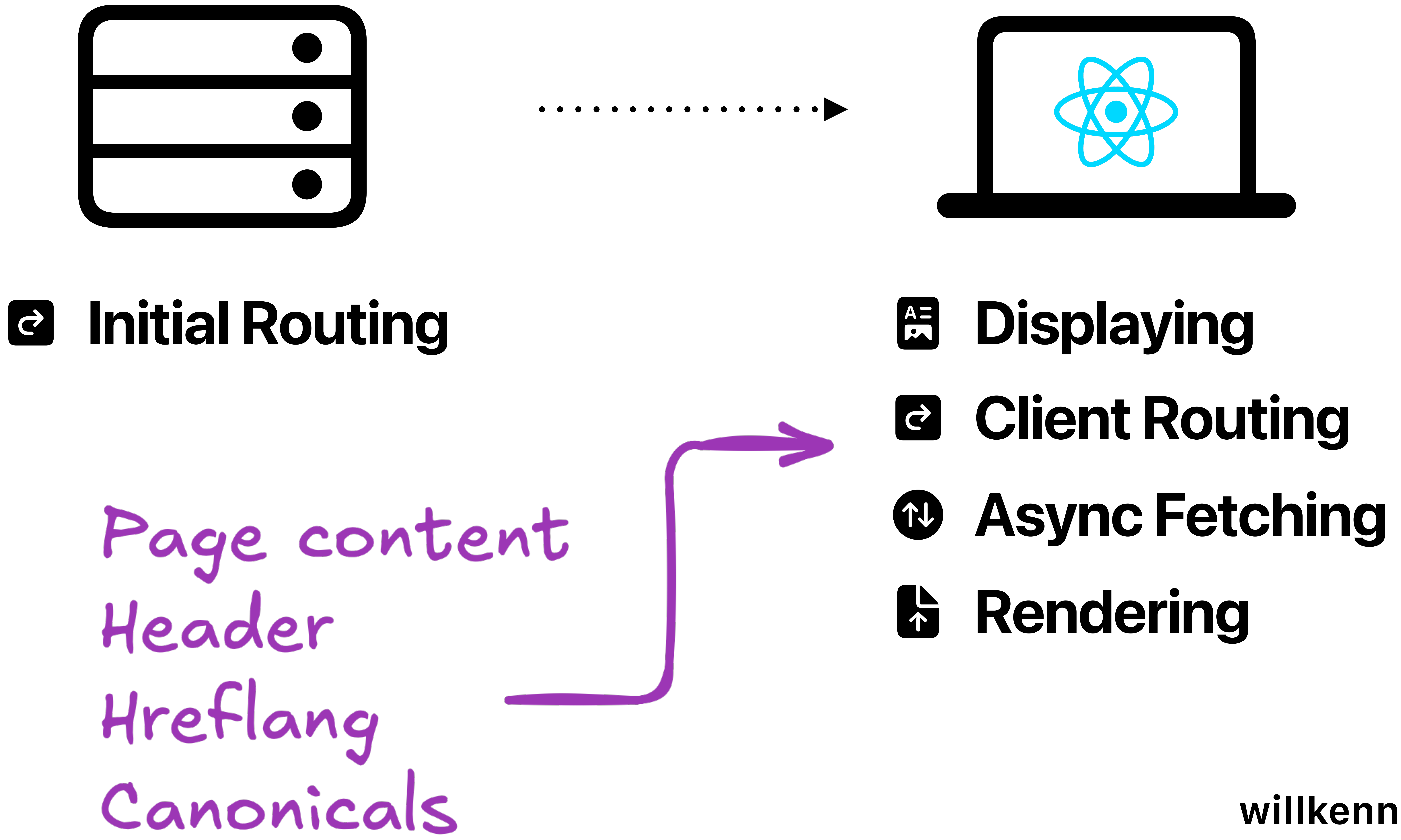
Async Fetching

Rendering

willkennard.com


Initial Routing
Entire JS bundle sent to client
Displaying
Client Routing
Async Fetching
Rendering


Initial Routing
Displaying
Client Routing
Async Fetching
Rendering
Entire JS bundle sent to client
& to Google!



willkennard.com

Crawl sites using Screaming Frog without JS enabled.

willkennard.com

If the crawl isn't crawling properly or lacks metadata, you have SPA issues.

Devs don't usually build bad websites on purpose. (probably)

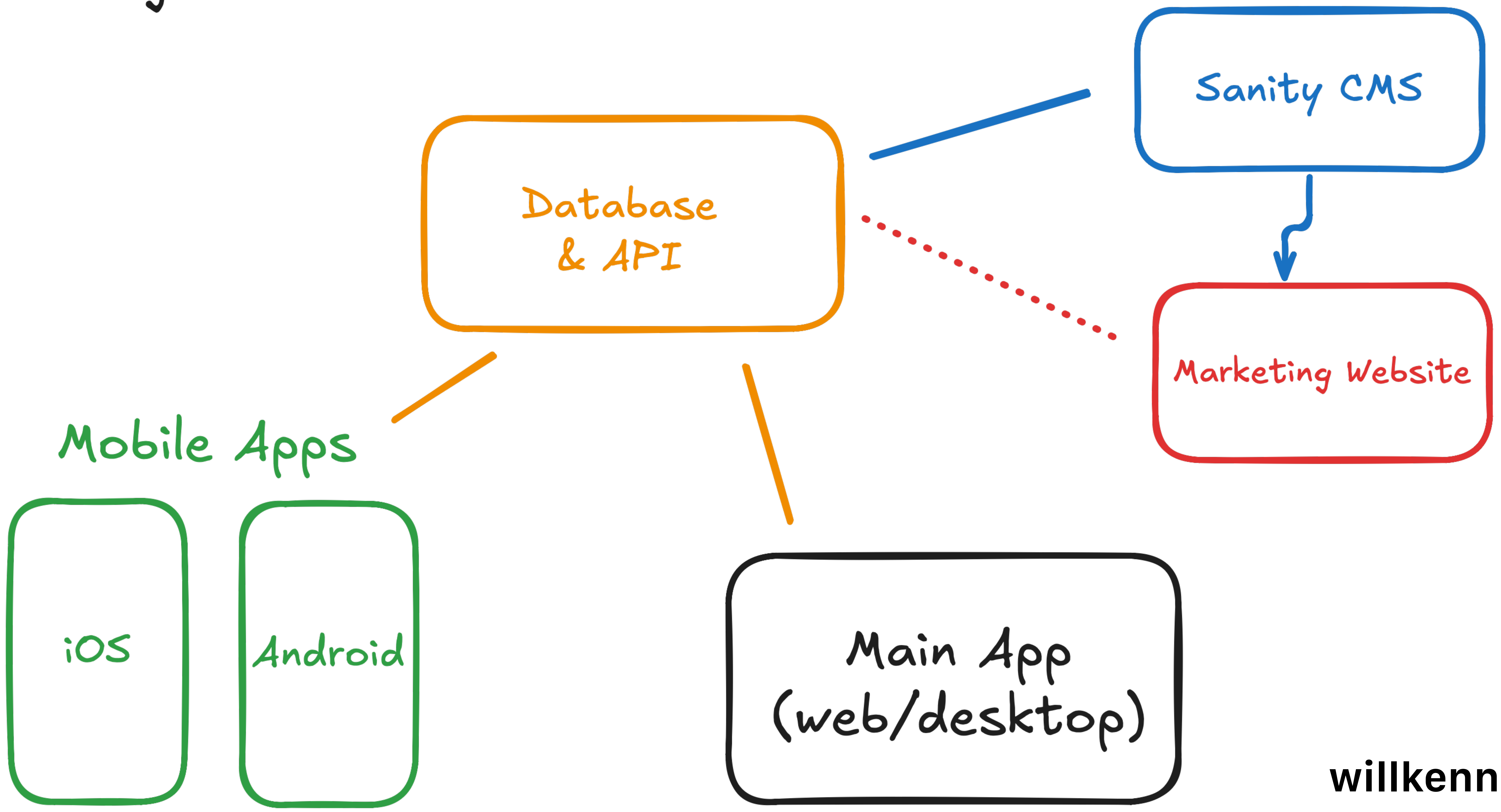
willkennard.com

JS frameworks make it convenient to work with one common language.

programming

willkennard.com

Multiple Customer Facing Platforms



willkennard.com

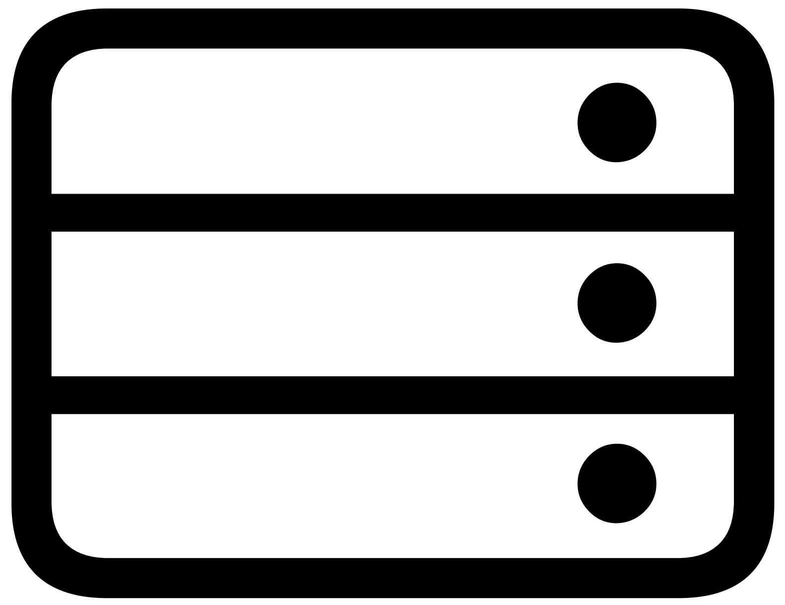
Work with devs to help them understand the importance of on-page content and links (including hreflang).

Most frameworks have SSR capability.

Next.js is now server first by default.

willkennard.com

Lots of frameworks still create client side apps by default.



- Next.js
- Remix
- Astro

- React
- Vue.js
- Angular
- SolidJS

willkennard.com

Try to get developers to think SSR first.

Client side functionality should need to be client side.

```
<div id="app">
```

```
<SiteHeader/>
  <TourDetails/>
  <VirtualTour/>
<SiteFooter/>
```

willkennard.com

```
<SiteHeader/>
 <TourDetails/>
 <VirtualTour/>
<SiteFooter/>
```

If basic page content & links are rendered client side, you should question why.

i18n
Internationalisation
&
l10n
Localisation

Crawling == routing issues

Rendering == client side issues

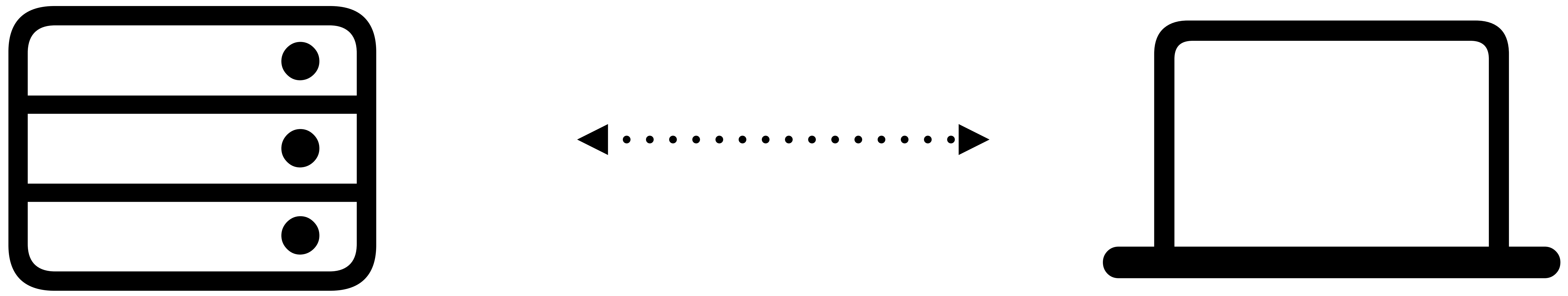
i18n ==
routing issues
l10n ==
client side issues

<u>i18n</u> ==
routing issues
<u>l10n</u> ==
client side issues (kind of)

willkennard.com

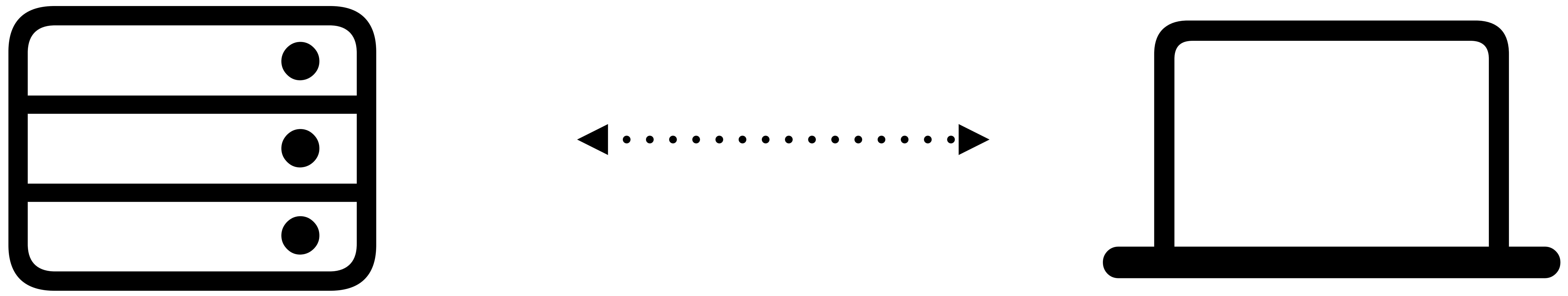
Dealing with <u>location routing</u>, content <u>rendering</u> & <u>localising</u> that content.

Let’s look at some framework specific examples.



- Routing
- Data Fetching
- Rendering

- Displaying
- Client Routing
- Async Fetching
- Rendering



1: Development Layer

2: Build Layer

3: Render Layer

4: Search Layer

5: User Layer

willkennard.com

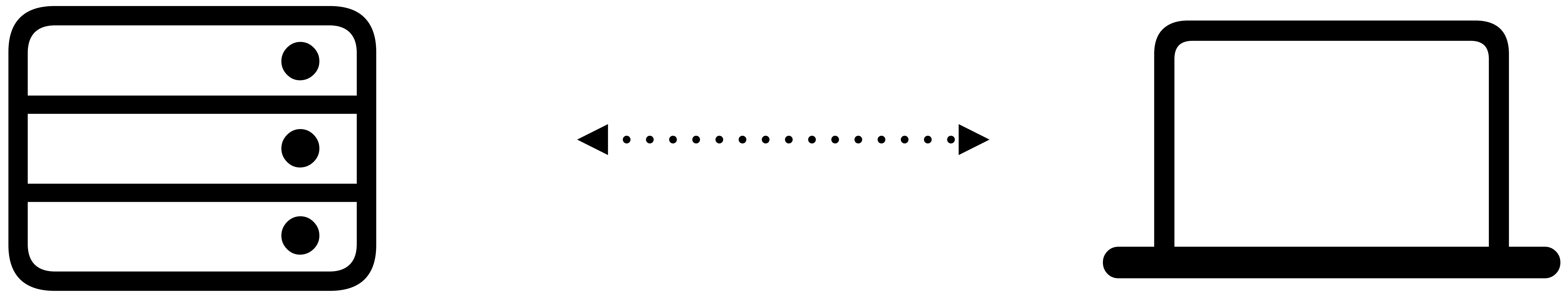
Next.js recommend routing by browser language.

Set locales and defaults in the Middleware file.

```
// In the Next.js project middleware.ts

let locales = ['en-GB', 'es-ES', 'fr-FR', 'de-DE']
let defaultLocale = 'en-GB'
```

Next 307 redirects the user to the lang route.

```
// In Next.js 'routes' are basically folders

app/
  [lang]/
    page.tsx
    layout.tsx
```



1: Development Layer

2: Build Layer

3: Render Layer

4: Search Layer

5: User Layer

willkennard.com

Nuxt & Angular also use the same approach.

Nuxt Angular

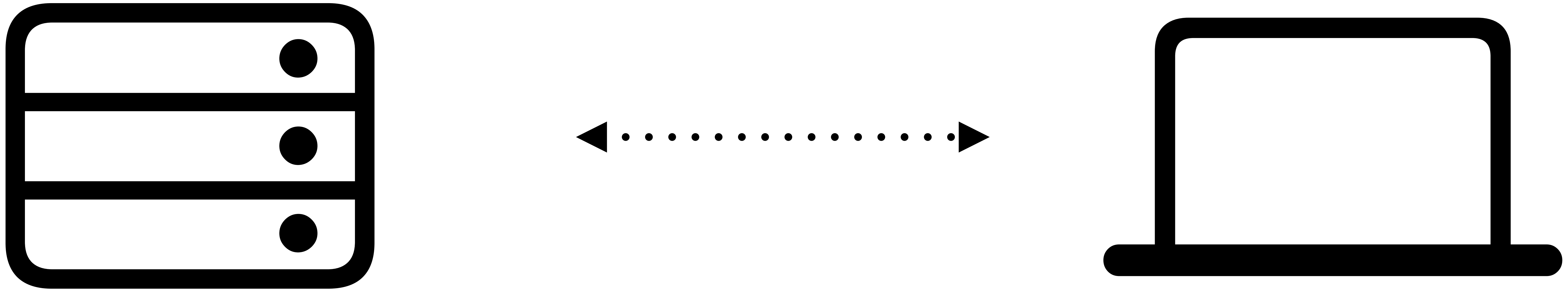
Next does i18n on a <u>subfolder</u> level.

Next does i18n on a <u>subfolder</u> level.

And so should you :)

willkennard.com

Vue & React soft redirect users client side.



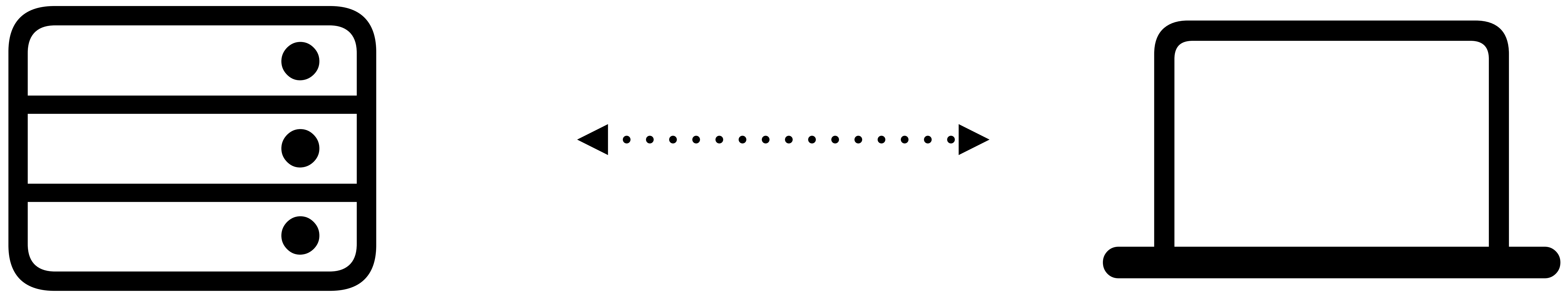
1: Development Layer

2: Build Layer

3: Render Layer

4: Search Layer

5: User Layer



1: Development Layer

2: Build Layer

3: Render Layer

4: Search Layer

5: User Layer

This means the JS bundle has to execute client side to enable i18n routing.

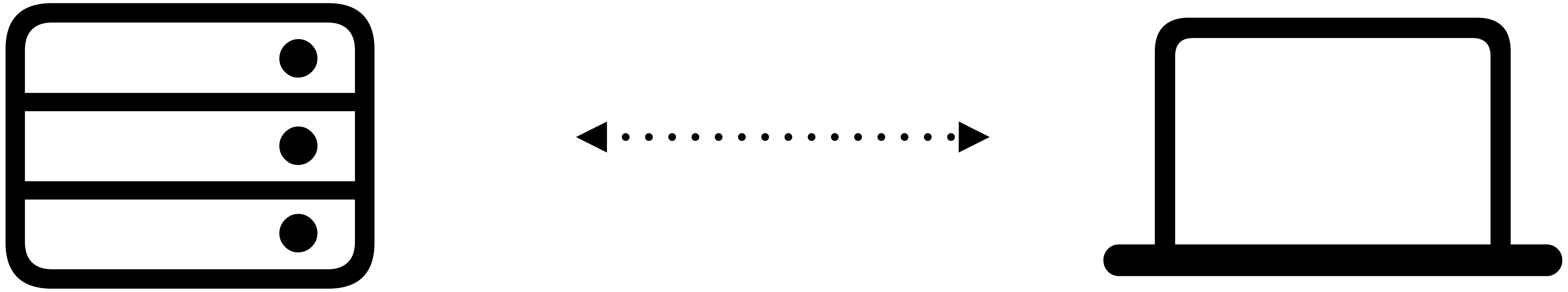
Which means crawlers that don’t render JS won’t see that.

Generally devs will be using a better solution. One to keep an eye on.

The app can then dynamically pull data that is internationalised.

Next 307 redirects the user to the lang route.

```
// In Next.js 'routes' are basically folders

app/
  [lang]/
    page.tsx
    layout.tsx
```



1: Development Layer

2: Build Layer

3: Render Layer

4: Search Layer

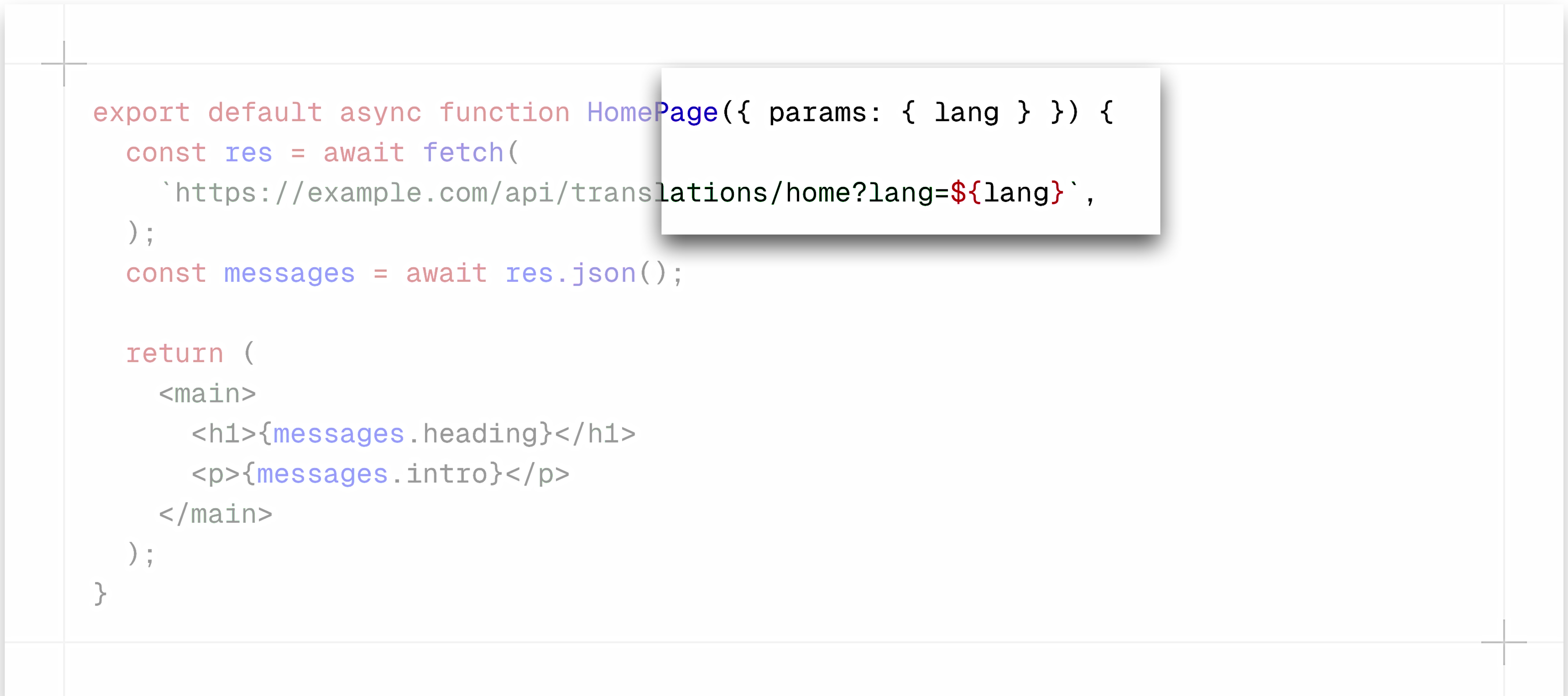
5: User Layer

willkennard.com

Then the whole app is passed lang props.

```
export default async function HomePage({ params: { lang } }) {
  const res = await fetch(
    `https://example.com/api/translations/home?lang=${lang}`,
  );
  const messages = await res.json();

  return (
    <main>
      <h1>{messages.heading}</h1>
      <p>{messages.intro}</p>
    </main>
  );
}
```

Meaning you can query lang specific data.

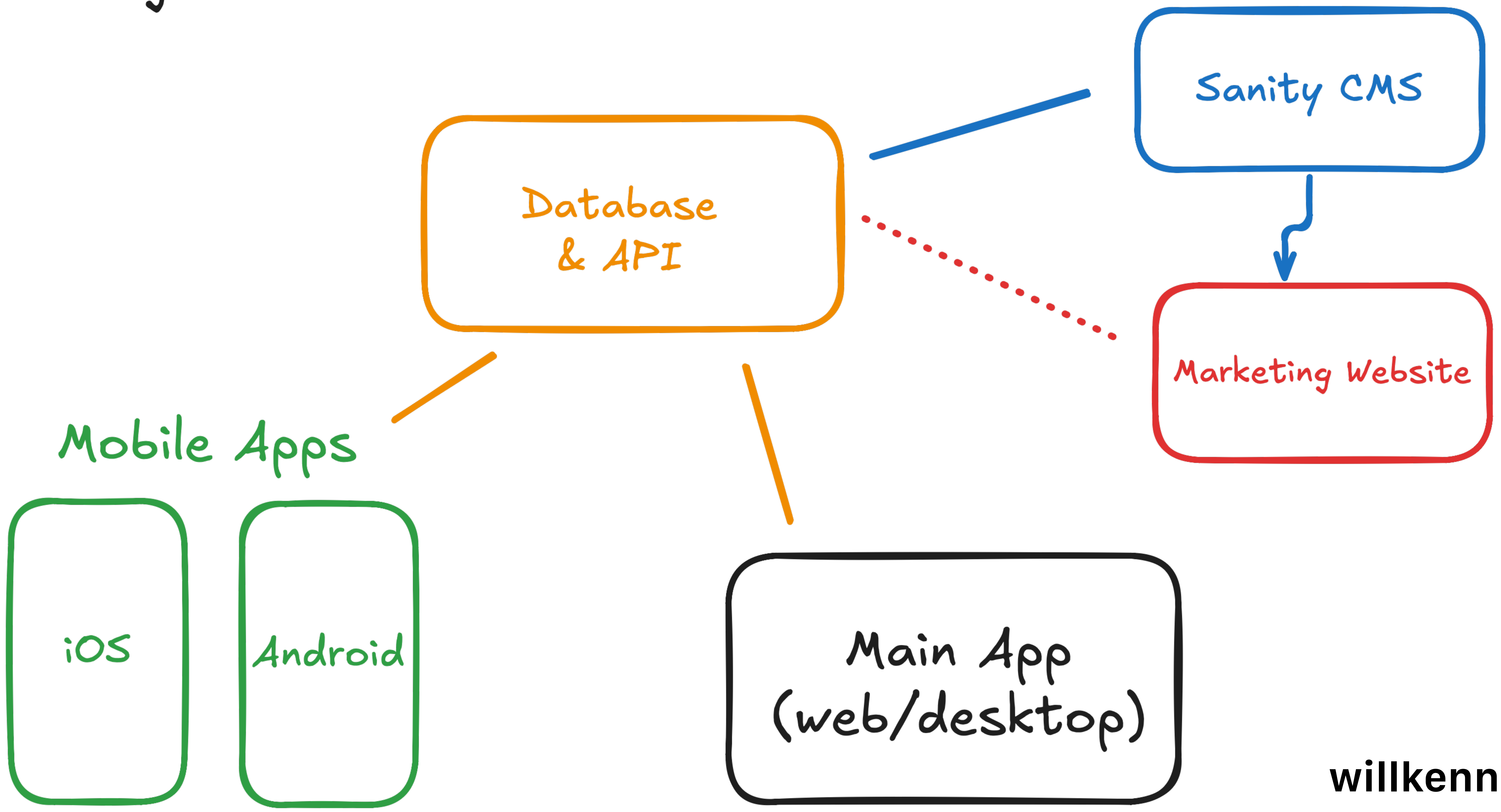


```
export default async function HomePage({ params: { lang } }) {
  const res = await fetch(
    `https://example.com/api/translations/home?lang=${lang}`,
  );
  const messages = await res.json();

  return (
    <main>
      <h1>{messages.heading}</h1>
      <p>{messages.intro}</p>
    </main>
  );
}
```

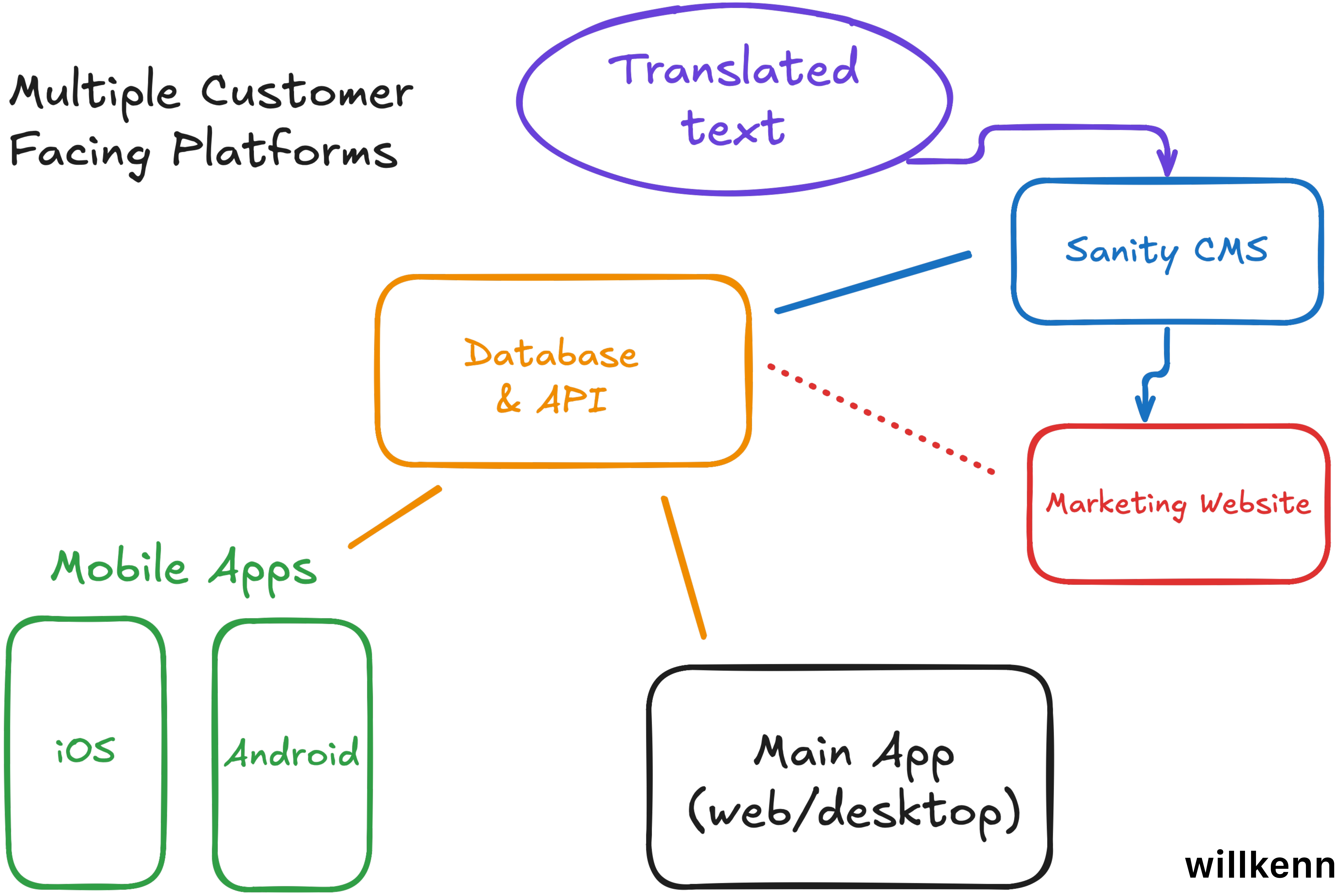
International ~~apps~~ websites are complex!

You would of course use a CMS.

Multiple Customer Facing Platforms



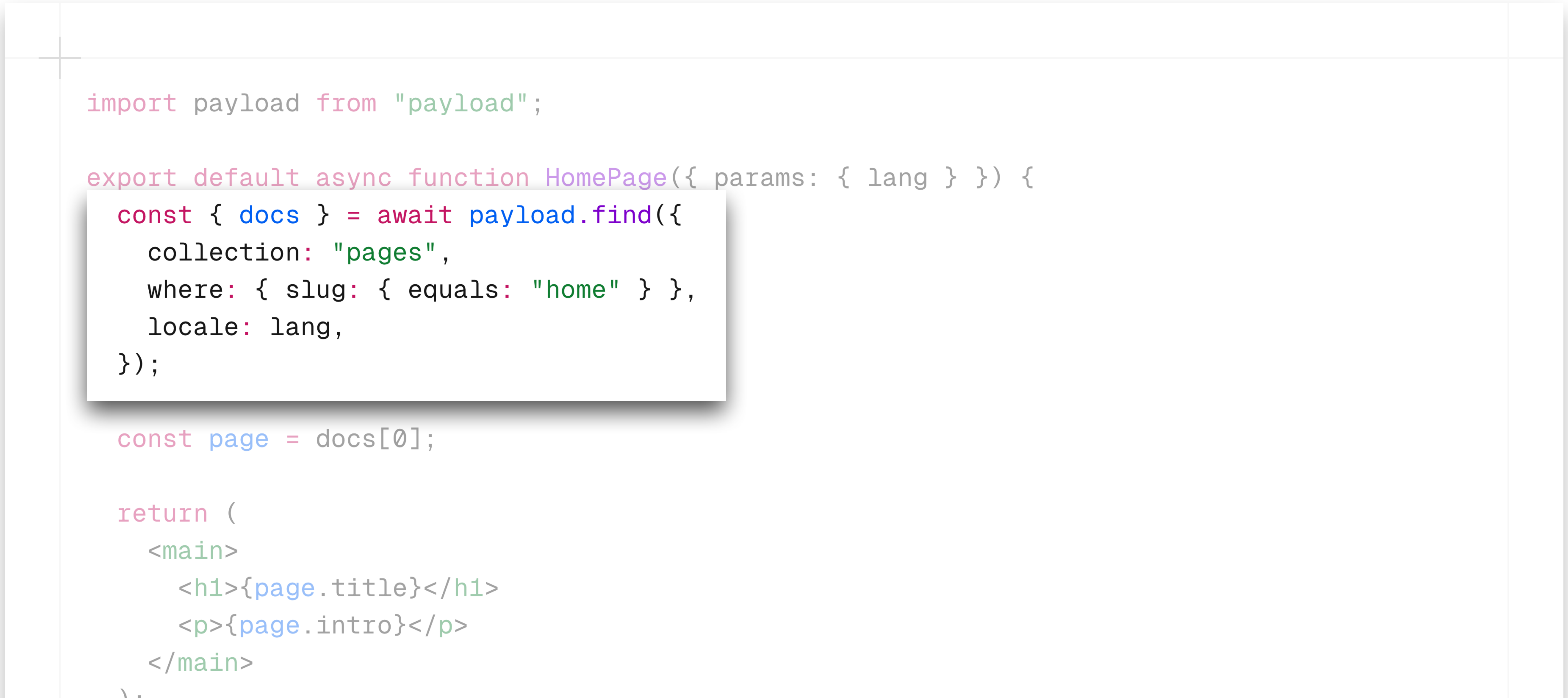
willkennard.com


Multiple Customer Facing Platforms
Translated text
Sanity CMS
Database & API
Marketing Website
Mobile Apps
iOS
Android
Main App (web/desktop)

Which means the app is then querying the CMS on a lang directory level.

```
import payload from "payload";

export default async function HomePage({ params: { lang } }) {
  const { docs } = await payload.find({
    collection: "pages",
    where: { slug: { equals: "home" } },
    locale: lang,
  });

  const page = docs[0];

  return (
    <main>
      <h1>{page.title}</h1>
      <p>{page.intro}</p>
    </main>
  );
```

The app doesn’t hardcode copy, it asks the CMS for it.

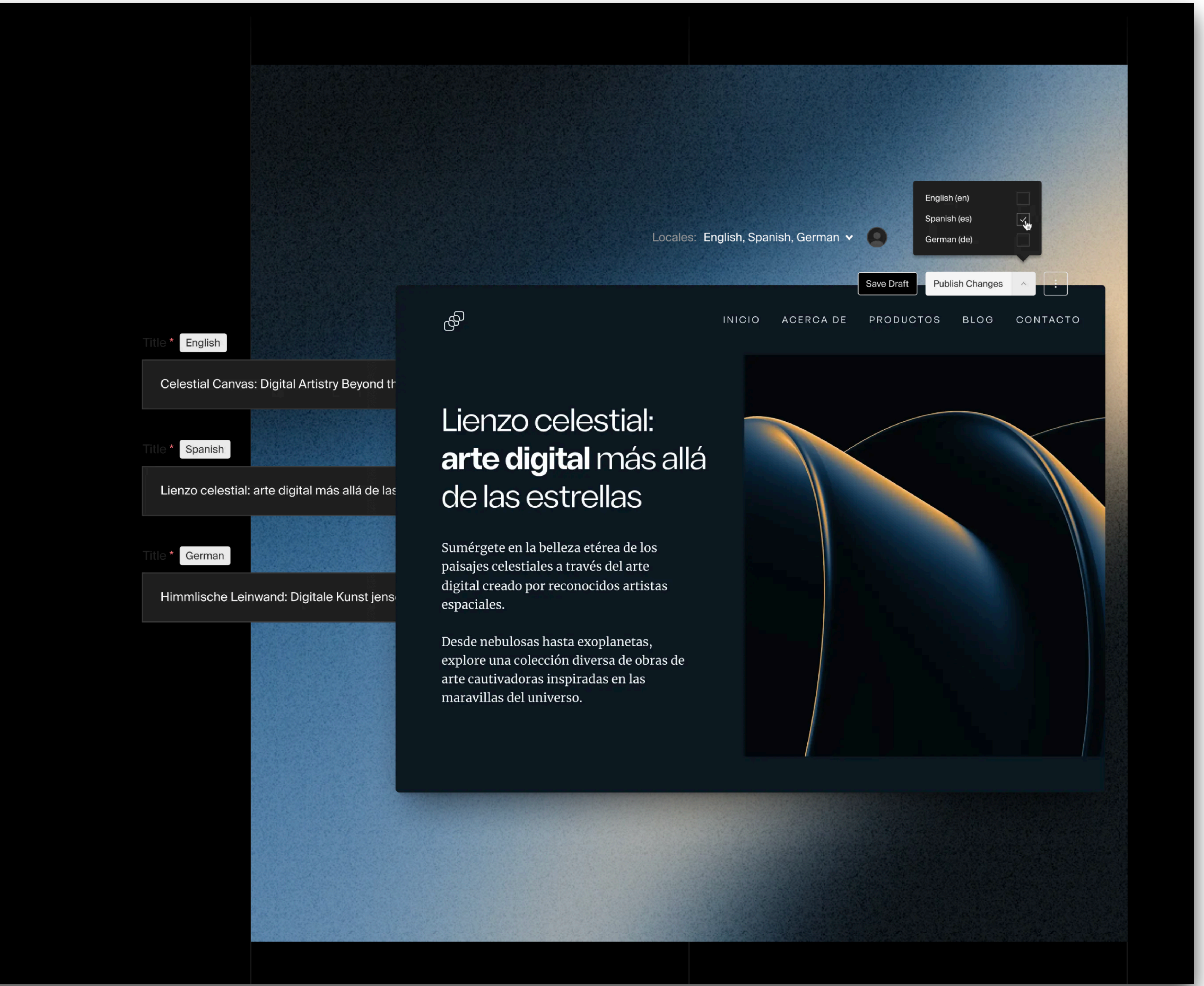


```
import payload from "payload";

export default async function HomePage({ params: { lang } }) {
  const { docs } = await payload.find({
    collection: "pages",
    where: { slug: { equals: "home" } },
    locale: lang,
  });

  const page = docs[0];

  return (
    <main>
      <h1>{page.title}</h1>
      <p>{page.intro}</p>
    </main>
  );
```

Localisation can be enabled on a field level in most modern CMS's.

willkennard.com

Once the app is configured, the content is entered by your CMS users.



Locales: English, Spanish, German

Title * English

Celestial Canvas: Digital Artistry Beyond th

Title * Spanish

Lienzo celestial: arte digital más allá de las

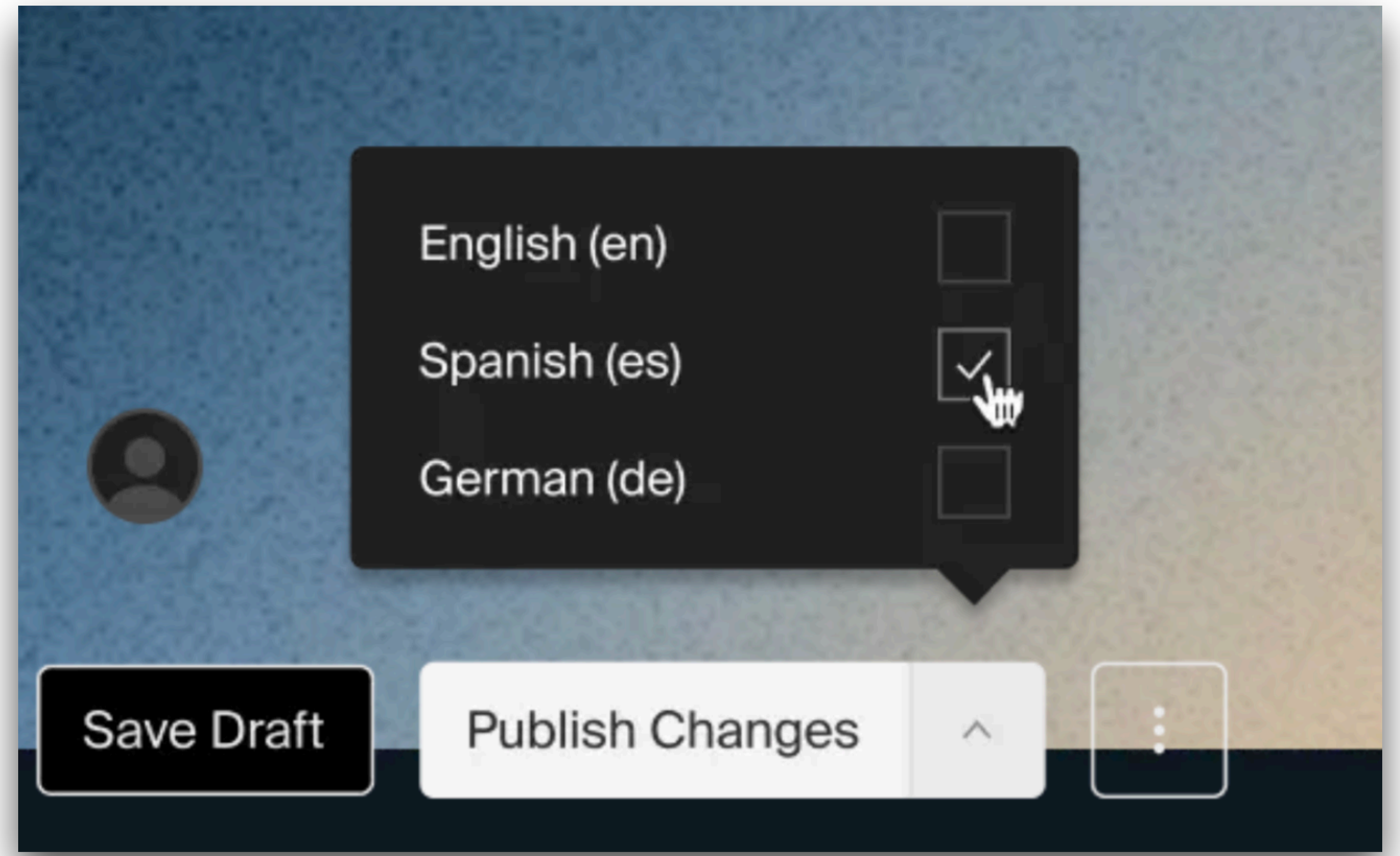
Title * German

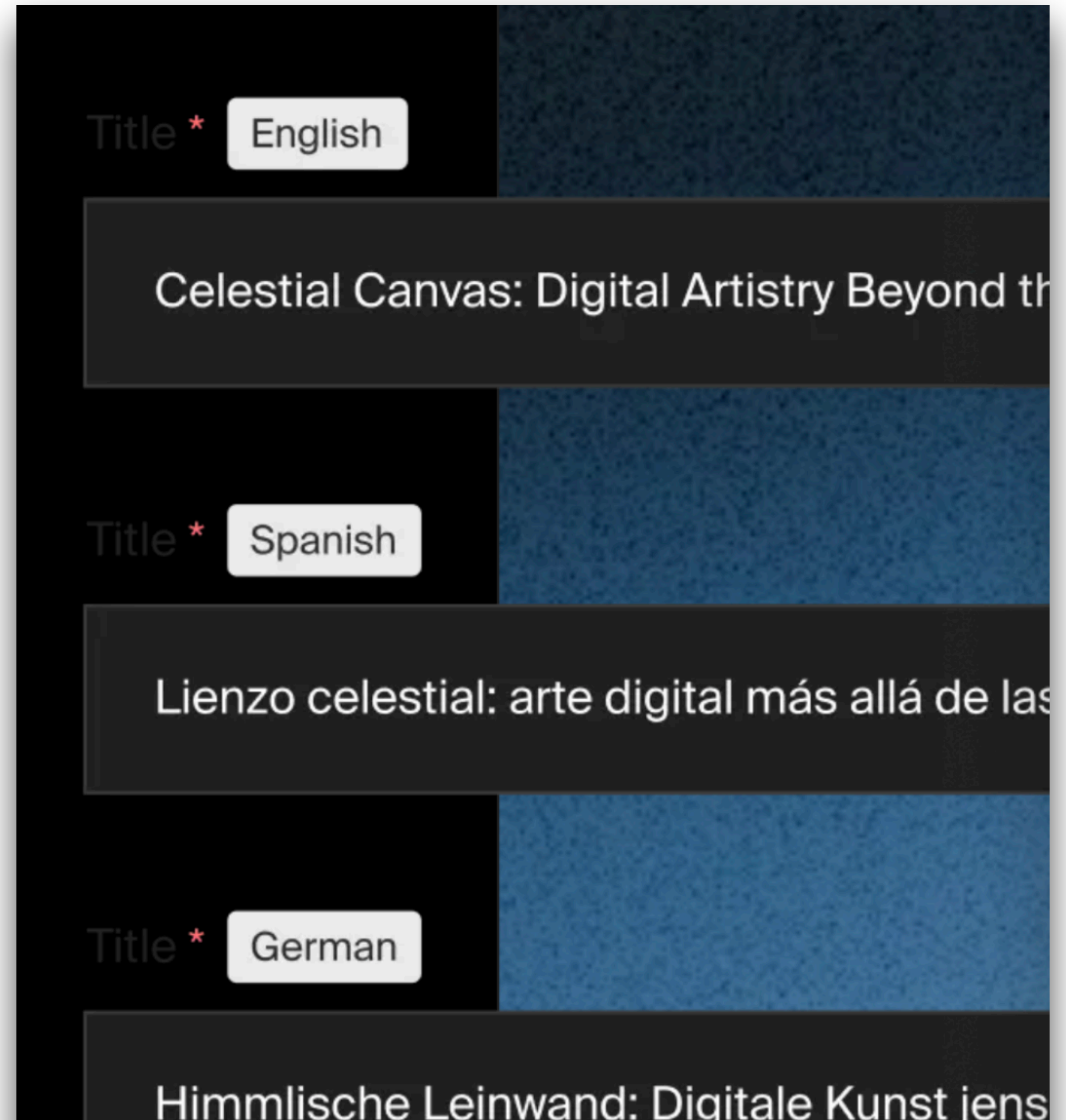
Himmlische Leinwand: Digitale Kunst jens

Lienzo celestial: **arte digital** más allá de las estrellas

Sumérgete en la belleza etérea de los paisajes celestiales a través del arte digital creado por reconocidos artistas espaciales.

Desde nebulosas hasta exoplanetas, explore una colección diversa de obras de arte cautivadoras inspiradas en las maravillas del universo.





Title * English

Celestial Canvas: Digital Artistry Beyond th

Title * Spanish

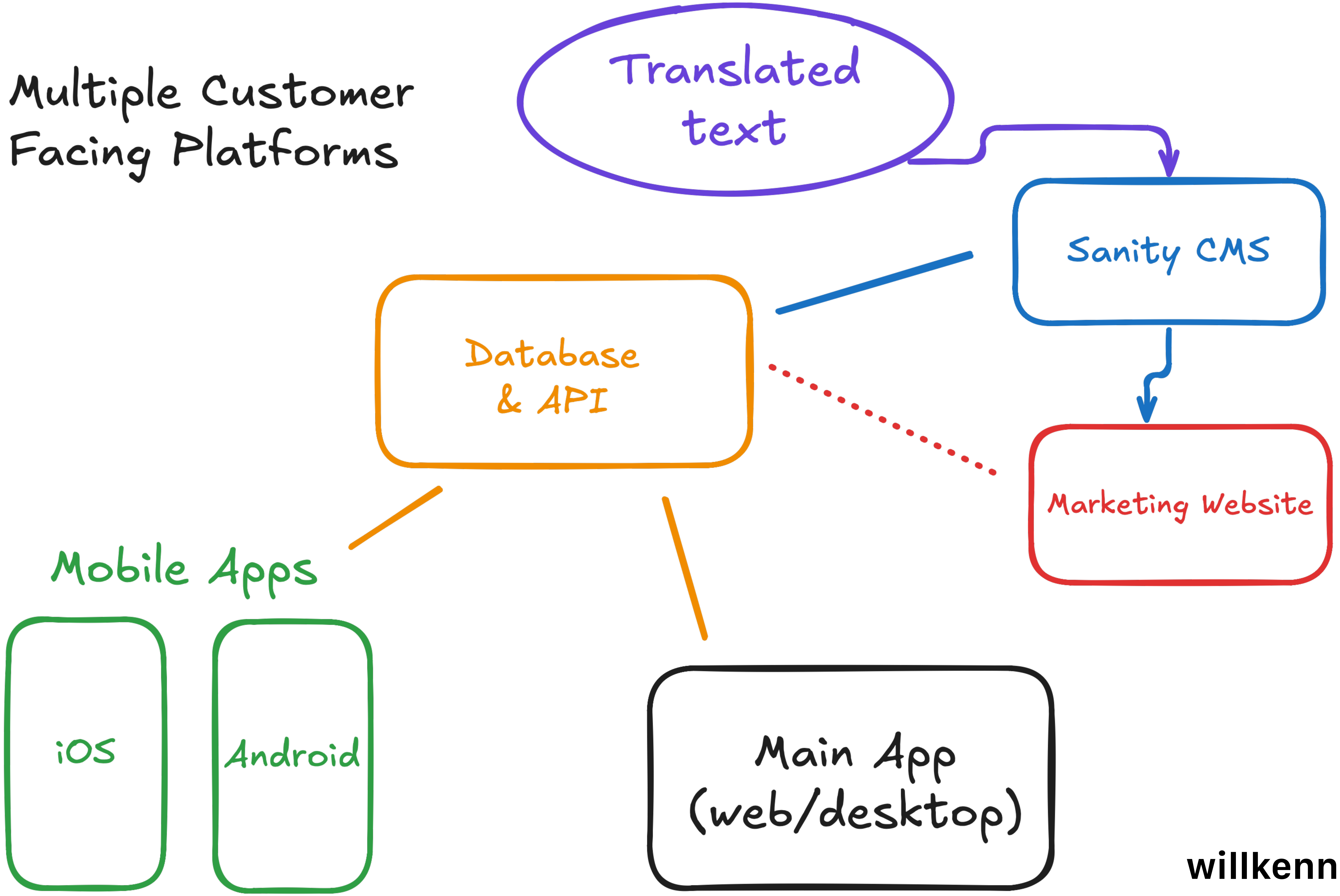
Lienzo celestial: arte digital más allá de las

Title * German

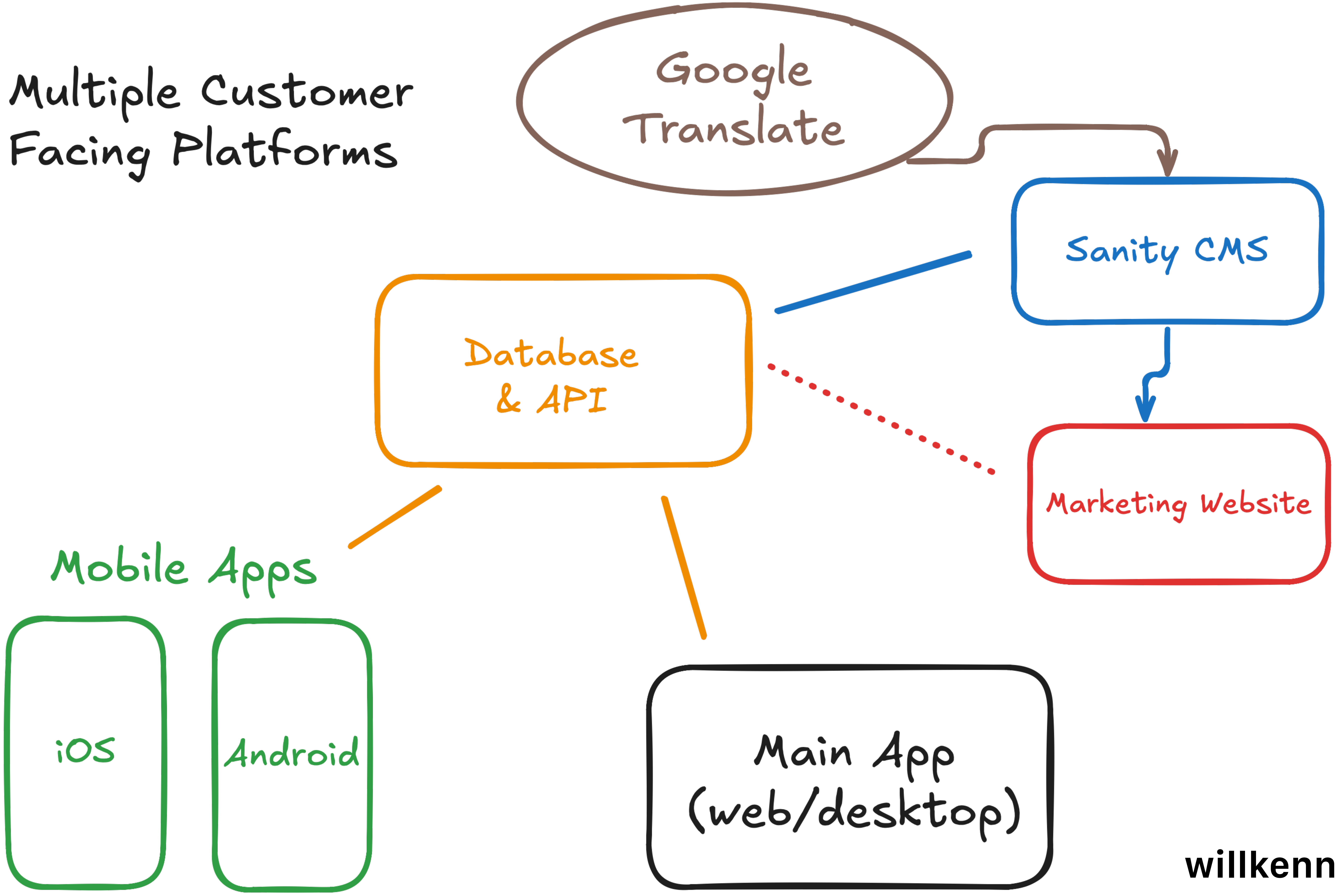
Himmlische Leinwand: Digitale Kunst jens

Now it is left to human error :)

Or you can leave it to computer error.

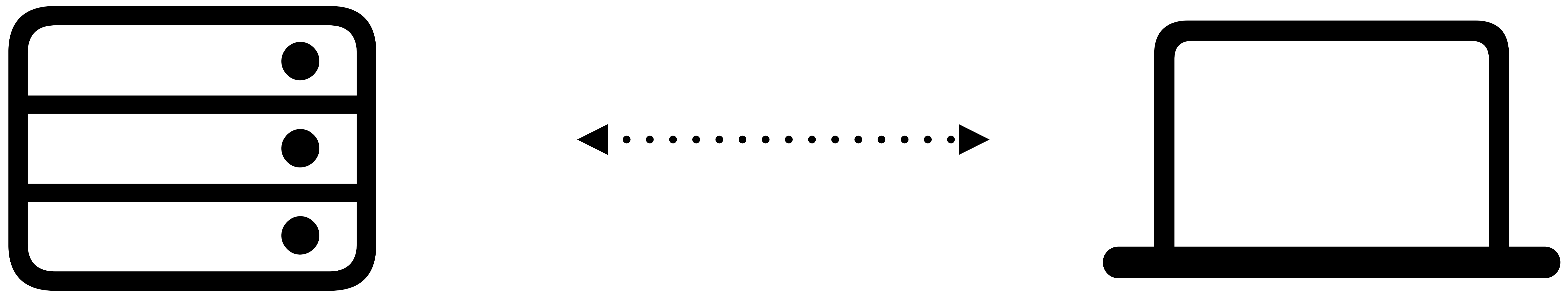


willkennard.com



willkennard.com

If you take away one thing from today…



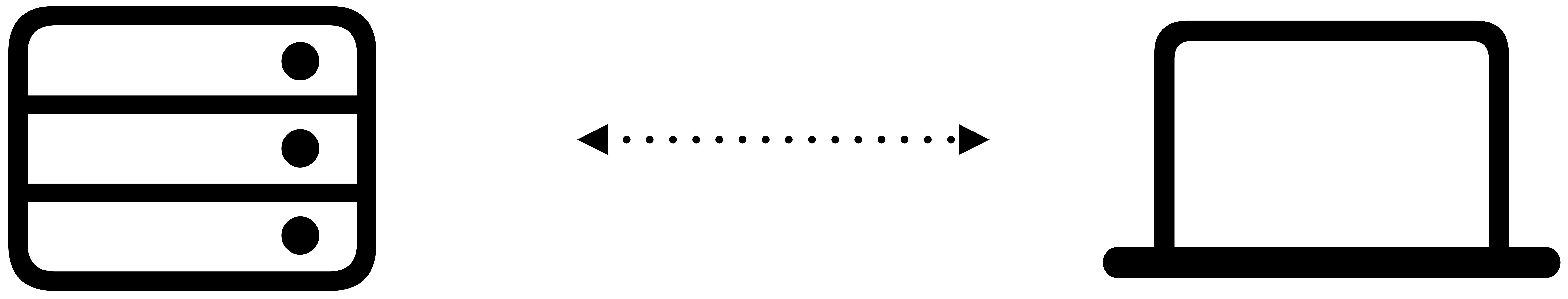
1: Development Layer

2: Build Layer

3: Render Layer

4: Search Layer

5: User Layer



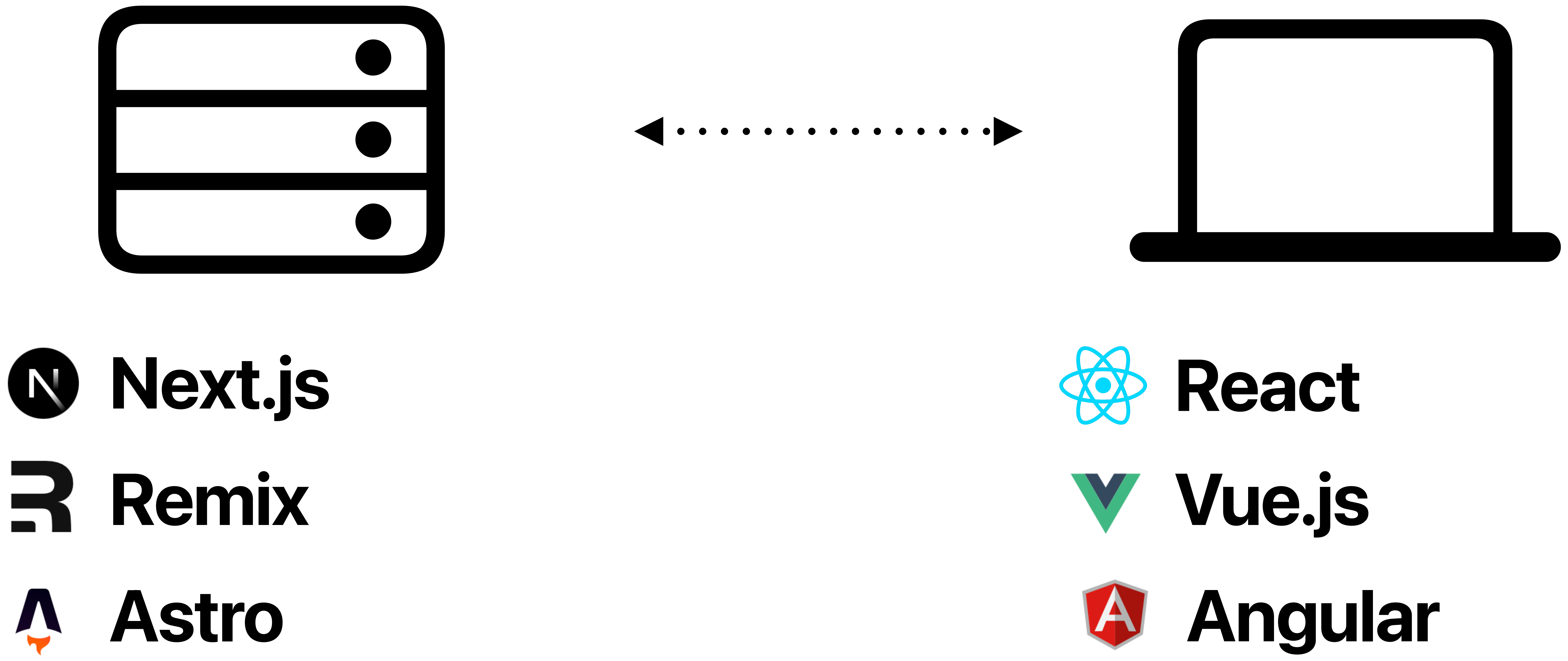
1: Development Layer

2: Build Layer

3: Render Layer

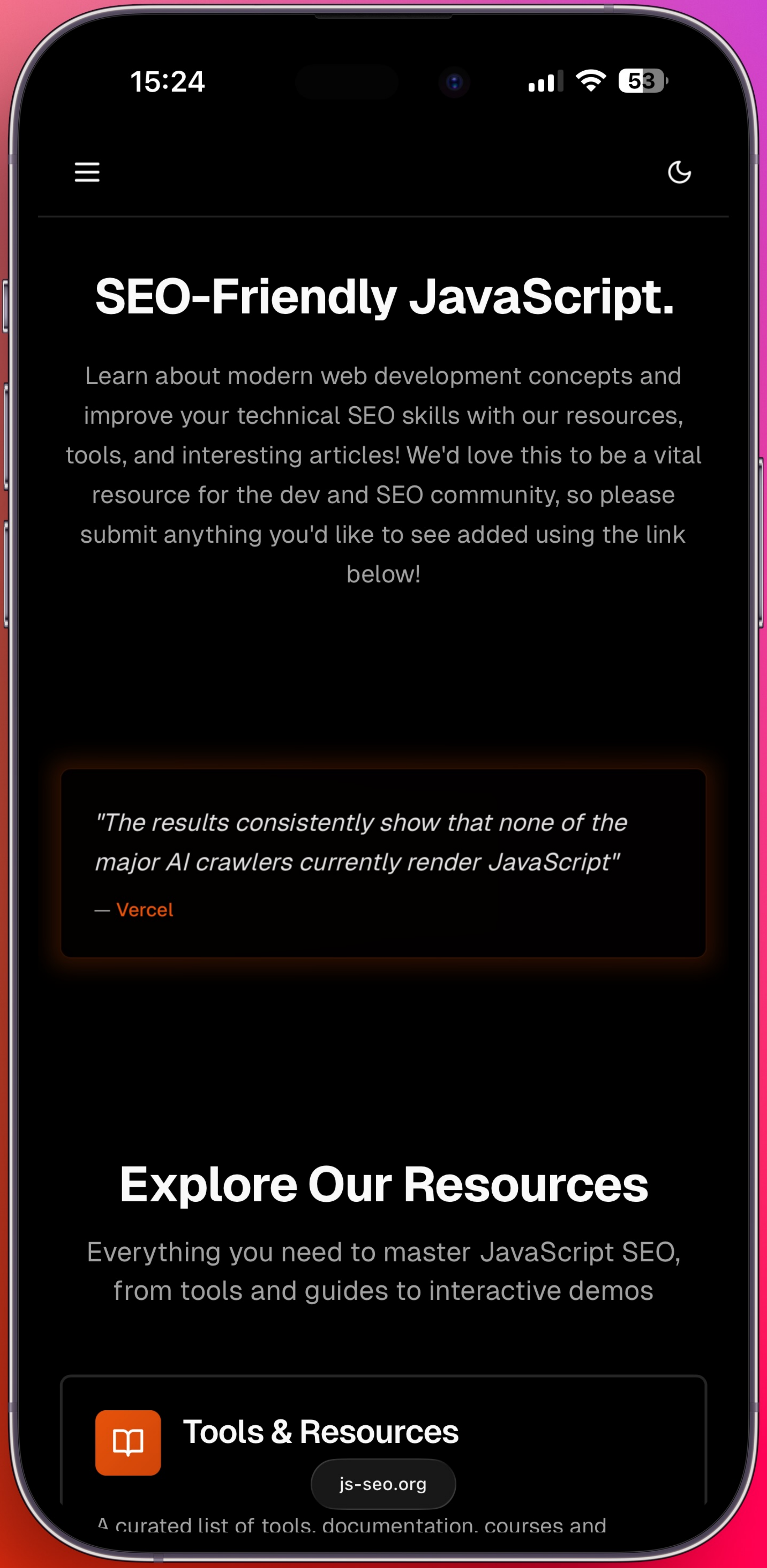
4: Search Layer

5: User Layer



willkennard.com

Submit your JS resources or case studies!


15:24

SEO-Friendly JavaScript.

Learn about modern web development concepts and improve your technical SEO skills with our resources, tools, and interesting articles! We'd love this to be a vital resource for the dev and SEO community, so please submit anything you'd like to see added using the link below!

"The results consistently show that none of the major AI crawlers currently render JavaScript"

— Vercel

Explore Our Resources

Everything you need to master JavaScript SEO, from tools and guides to interactive demos

Tools & Resources

js-seo.org

A curated list of tools, documentation, courses and

js-seo.org

willkennard.com

Will Kennard
Founder, Consultant

@willkennard
linkedin.com/in/willkennard