Starting soon!

IT Friday #008

Weekly live stream for Ukrainian IT professionals worldwide · Every Friday 19:00 London / 21:00 Kyiv

Today

Content Security Policy (CSP) · Oleksandr Blazheiko — ~40–60 min

React+Vite vs Next.js — a neutral guide + decision tree — ~20 min

Q&A — ~10 min

Today

Content Security Policy (CSP) · Oleksandr Blazheiko — ~40–60 min

React+Vite vs Next.js — a neutral guide + decision tree — ~20 min

Q&A — ~10 min

Today

Content Security Policy (CSP) · Oleksandr Blazheiko — ~40–60 min

React+Vite vs Next.js — a neutral guide + decision tree — ~20 min

Q&A — ~10 min

Today

Content Security Policy (CSP) · Oleksandr Blazheiko — ~40–60 min

React+Vite vs Next.js — a neutral guide + decision tree — ~20 min

Q&A — ~10 min

React+Vite vs Next.js

npm create vite — or npx create-next-app? A checklist, not a verdict.

What each one is

Vite — build tool + dev server. Fast HMR. No opinion on routing or rendering.

Next.js — full framework. Routing, rendering strategies, API routes, optimizations — out of the box.

Same first command, very different amount of decisions made for you

What each one is

Vite — build tool + dev server. Fast HMR. No opinion on routing or rendering.

Next.js — full framework. Routing, rendering strategies, API routes, optimizations — out of the box.

Same first command, very different amount of decisions made for you

What each one is

Vite — build tool + dev server. Fast HMR. No opinion on routing or rendering.

Next.js — full framework. Routing, rendering strategies, API routes, optimizations — out of the box.

Same first command, very different amount of decisions made for you

What each one is

Vite — build tool + dev server. Fast HMR. No opinion on routing or rendering.

Next.js — full framework. Routing, rendering strategies, API routes, optimizations — out of the box.

Same first command, very different amount of decisions made for you

First look after scaffolding

npm create vite@latest — vs — npx create-next-app@latest

📸 Vite — minimal: `src/`, `index.html`, `vite.config.ts`

📸 Next.js — already opinionated: `app/`, `layout.tsx`, `next.config.ts`

First look after scaffolding

npm create vite@latest — vs — npx create-next-app@latest

💥 Vite — minimal: `src/`, `index.html`, `vite.config.ts`

💥 Next.js — already opinionated: `app/`, `layout.tsx`, `next.config.ts`

First look after scaffolding

npm create vite@latest — vs — npx create-next-app@latest

💣 Vite — minimal: `src/`, `index.html`, `vite.config.ts`

💣 Next.js — already opinionated: `app/`, `layout.tsx`, `next.config.ts`

Rendering

Vite defaults to CSR — empty HTML shell, JS assembles everything in the browser

Next.js ships SSR / SSG / ISR out of the box — page can render on the server or be pre-built

Why it matters: SEO, first contentful paint, performance on low-end devices

Rendering

Vite defaults to CSR — empty HTML shell, JS assembles everything in the browser

Next.js ships SSR / SSG / ISR out of the box — page can render on the server or be pre-built

Why it matters: SEO, first contentful paint, performance on low-end devices

Rendering

Vite defaults to CSR — empty HTML shell, JS assembles everything in the browser

Next.js ships SSR / SSG / ISR out of the box — page can render on the server or be pre-built

Why it matters: SEO, first contentful paint, performance on low-end devices

Rendering

Vite defaults to CSR — empty HTML shell, JS assembles everything in the browser

Next.js ships SSR / SSG / ISR out of the box — page can render on the server or be pre-built

Why it matters: SEO, first contentful paint, performance on low-end devices

Same page, two models

Vite / CSR

```
function Page() {
  const [data, setData] = useState(null);
  useEffect(() => {
    fetch('/api/posts').then(r => r.json()).then(setData);
  }, []);
  if (!data) return <Spinner />;
  return <PostList posts={data} />;
}
```

Next.js / Server Component

```
async function Page() {
  const data = await fetch(
    'https://api.example.com/posts'
  ).then(r => r.json());
  return <PostList posts={data} />;
}
```

Routing

Vite + React Router

```
const router = createBrowserRouter([
  { path: '/', element: <Home /> },
  { path: '/posts/:id', element: <Post /> },
]);
```

Next.js — file-based

```
app/
  page.tsx            → /
  posts/
    [id]/
      page.tsx        → /posts/:id
```

Data fetching

Vite — you choose: TanStack Query, SWR, or plain `fetch`. Flexibility, but the responsibility too.

Next.js — `fetch` directly in a Server Component, caching and `revalidate` built in

More magic out of the box → less manual control. And vice versa.

Data fetching

Vite — you choose: TanStack Query, SWR, or plain `fetch`. Flexibility, but the responsibility too.

Next.js — `fetch` directly in a Server Component, caching and `revalidate` built in

More magic out of the box → less manual control. And vice versa.

Data fetching

Vite — you choose: TanStack Query, SWR, or plain `fetch`. Flexibility, but the responsibility too.

Next.js — `fetch` directly in a Server Component, caching and `revalidate` built in

More magic out of the box → less manual control. And vice versa.

Data fetching

Vite — you choose: TanStack Query, SWR, or plain `fetch`. Flexibility, but the responsibility too.

Next.js — `fetch` directly in a Server Component, caching and `revalidate` built in

More magic out of the box → less manual control. And vice versa.

Dev experience & build

Vite dev server — near-instant start and HMR, even on mid-sized projects

Next.js with Turbopack caught up on raw speed

But the mental model is heavier: server vs client components, request caching

Dev experience & build

Vite dev server — near-instant start and HMR, even on mid-sized projects

Next.js with Turbopack caught up on raw speed

But the mental model is heavier: server vs client components, request caching

Dev experience & build

Vite dev server — near-instant start and HMR, even on mid-sized projects

Next.js with Turbopack caught up on raw speed

But the mental model is heavier: server vs client components, request caching

Dev experience & build

Vite dev server — near-instant start and HMR, even on mid-sized projects

Next.js with Turbopack caught up on raw speed

But the mental model is heavier: server vs client components, request caching

Deploy

Vite — static build (`dist/`), any host: GitHub Pages, Netlify, S3 + CloudFront

Next.js — best on Vercel out of the box; self-host needs a Node server for SSR/ISR

`output: 'export'` gives you a static Next.js build — but you lose SSR/ISR

Deploy

Vite — static build (`dist/`), any host: GitHub Pages, Netlify, S3 + CloudFront

Next.js — best on Vercel out of the box; self-host needs a Node server for SSR/ISR

`output: 'export'` gives you a static Next.js build — but you lose SSR/ISR

Deploy

Vite — static build (`dist/`), any host: GitHub Pages, Netlify, S3 + CloudFront

Next.js — best on Vercel out of the box; self-host needs a Node server for SSR/ISR

`output: 'export'` gives you a static Next.js build — but you lose SSR/ISR

Deploy

Vite — static build (`dist/`), any host: GitHub Pages, Netlify, S3 + CloudFront

Next.js — best on Vercel out of the box; self-host needs a Node server for SSR/ISR

`output: 'export'` gives you a static Next.js build — but you lose SSR/ISR

Ecosystem & flexibility

Vite — framework-agnostic: same tool for React, Vue, Svelte, vanilla JS

Next.js — React only, but routing + rendering + data fetching are pre-wired together

Ecosystem & flexibility

Vite — framework-agnostic: same tool for React, Vue, Svelte, vanilla JS

Next.js — React only, but routing + rendering + data fetching are pre-wired together

Ecosystem & flexibility

Vite — framework-agnostic: same tool for React, Vue, Svelte, vanilla JS

Next.js — React only, but routing + rendering + data fetching are pre-wired together

Decision tree

Not "which is better" — answer honestly for your project

Is SEO / first contentful paint critical?

No → **Vite** is enough

Is the team OK with a heavier rendering model (server/client components, caching)?

No → **Vite**, add SSR later if needed · Yes → **Next.js**

Decision tree

Not "which is better" — answer honestly for your project

Is SEO / first contentful paint critical?

No → **Vite** is enough

Is the team OK with a heavier rendering model (server/client components, caching)?

No → **Vite**, add SSR later if needed · Yes → **Next.js**

Decision tree

Not "which is better" — answer honestly for your project

Is SEO / first contentful paint critical?

No → **Vite** is enough

Is the team OK with a heavier rendering model (server/client components, caching)?

No → **Vite**, add SSR later if needed · Yes → **Next.js**

Decision tree — cont.

Internal tool / dashboard / prototype?

Yes → **Vite** — SEO and SSR are rarely worth it here

Deploy must be off Vercel, no Node server?

Yes → **Vite**, or Next.js with `output: 'export'`

Decision tree — cont.

Internal tool / dashboard / prototype?

Yes → **Vite** — SEO and SSR are rarely worth it here

Deploy must be off Vercel, no Node server?

Yes → **Vite**, or Next.js with `output: 'export'`

Decision tree — cont.

Internal tool / dashboard / prototype?

Yes → **Vite** — SEO and SSR are rarely worth it here

Deploy must be off Vercel, no Node server?

Yes → **Vite**, or Next.js with `output: 'export'`

What do you usually reach for — Vite or Next.js? Why? 👇

Not better or worse — different jobs

Vite — when you want to assemble the stack yourself and stay in control

Next.js — when you want a framework that already decided routing, rendering, and data fetching

The decision tree is a checklist you can run on any new project tomorrow

Not better or worse — different jobs

Vite — when you want to assemble the stack yourself and stay in control

Next.js — when you want a framework that already decided routing, rendering, and data fetching

The decision tree is a checklist you can run on any new project tomorrow

Not better or worse — different jobs

Vite — when you want to assemble the stack yourself and stay in control

Next.js — when you want a framework that already decided routing, rendering, and data fetching

The decision tree is a checklist you can run on any new project tomorrow

Not better or worse — different jobs

Vite — when you want to assemble the stack yourself and stay in control

Next.js — when you want a framework that already decided routing, rendering, and data fetching

The decision tree is a checklist you can run on any new project tomorrow

Q&A

CSP · React+Vite vs Next.js — ask away

Every Friday · 19:00 London / 21:00 Kyiv

itfriday.community