Nuxt for Drupal Developers

Modern Frontend Without the JavaScript Jungle

Drupal Dev Days 2025-04-15 - Wolfgang Ziegler // fago

drunomics

About me

Wolfgang Ziegler // fago
Managing Partner, CTO of drunomics

Contributions
- Drupal 8 Entity API & Typed Data API
- Contrib: Rules, Entity API, Field Collection
- Lupus Decoupled Drupal lead

@wolfz.bsky.social
drupal.org/u/fago

drunomics

Quick poll

➔ Who would consider themselves as frontend-dev?

➔ Who has experience working with React?

drunomics

Why Nuxt

- ➔ **Vue Framework** - Build on top of Vue's simplicity. Easy!
- ➔ **Great DX** - Auto-imports, file-based routing, zero-config
- ➔ **Performance focused** - SSR, static sites, code splitting
- ➔ **Production ready** - Enterprise grade, battle-tested
- ➔ **Versatile** - Works for content sites, apps, e-commerce
- ➔ **Active ecosystem** - Thriving community, regular updates

drunomics

A new project…

➔ Command for getting started

```
npm create nuxt@latest
```

➔ Start development

```
npm run dev
```

drunomics

```
nuxt-project/
├── assets/          # Unprocessed assets (will be processed)
├── components/      # Vue components (auto-imported)
├── composables/     # Reusable Vue composition functions
├── layouts/         # Page layouts (default.vue, etc.)
├── pages/           # Application routes/views
├── public/          # Static assets (copied as-is)
├── server/          # Server routes and API endpoints
├── app.vue          # Main application component
└── nuxt.config.ts   # Nuxt configuration file
```

drunomics

Vue.js 1x1

- ➔ Vue.js is web-standard oriented!
- ➔ Vue Single File Components: `Component.vue`
  - HTML <template>
  - CSS
  - JavaScript
- ➔ Docs: https://vuejs.org/docs

drunomics

Vue.js Component Model

➔ Use components via Markup

```
<MyComponent prop="value" :prop2="1+2">
</MyComponent>
```

➔ Pass data down via props, upwards via emitting events

drunomics

Vue.js Component Example

```
<script setup>
const props = defineProps({
  title: { type: String, default: 'Default Title' },
  subtitle: String
})
</script>
<template>
  <div class="card">
    <h2>{{ props.title }}</h2>
    <p>{{ props.subtitle }}</p>
  </div>
</template>
```

drunomics

Vue Template Syntax

```
<template>
  <!-- Text interpolation -->
  <h1>Hello, {{ name }}</h1>

  <!-- Attribute binding -->
  <img :src="imageUrl" :alt="imageDesc">

  <!-- Event handling -->
  <button @click="count++">Clicked {{ count }} times</button>
</template>
```

drunomics

Vue Template Syntax (2/2)

```
<!-- Conditionals -->
 <p v-if="count > 10">You clicked a lot!</p>
 <p v-else>Just started clicking</p>

 <!-- List rendering -->
 <ul>
   <li v-for="item in items" :key="item.id">
       {{ item.name }}
   </li>
 </ul>
```

drunomics

Vue.js Reactivity Model

- ➔ Reactivity handled automatically
- ➔ Vue wraps objects with refs()
- ➔ Vue tracks dependencies using those refs()
- ➔ Vue re-renders when a dependency changes
- ➔ Vue uses a virtual DOM to re-render DOM parts

drunomics

Vue.js Reactivity Example

```
<script setup>
const count = ref(0)
</script>

<template>
  <div>
    <p>Count: {{ count }}</p>
    <button @click="count++">Increment</button>
  </div>
</template>
```

drunomics

Entering Nuxt

Working with Vue components

drunomics

Nuxt Auto-Imports

```
<script setup>
~~import TheHeader from ../components/TheHeader.vue~~
~~import Button from ../components/Button.vue~~
const { width } = useWindowSize()
</script>

<template>
  <TheHeader />
  <Button @click="count++">Count: {{ count }}</Button>
</template>
```

drunomics

CSS handling

- ➔ Easily define global CSS in Nuxt
- ➔ Add per-component CSS in Vue files
- ➔ Nuxt takes care of processing
- ➔ Option to inline styles!
- ➔ Easily add PostCSS, Tailwind, SCSS ...

drunomics

Scoped component styles

```
<template>
  <div class=”my-wrapper”>
    <Button @click="count++">Count: {{ count }}</Button>
  </div>
</template>

<style lang=”postcss” scoped>
.my-wrapper {
  display: inline-block;
}
</style>
```

drunomics

Tailwind CSS

➔ Simply add nuxt tailwind module

```
npx nuxi add tailwind
```

➔ Start using it

```
<template>
  <div class=”inline-block”>
    <Button @click="count++">Count: {{ count }}</Button>
  </div>
</template>
```

drunomics

Client-side Navigation

- ➔ Avoid full-page loads for navigation → better UX!
- ➔ Built-into Nuxt, with vue-router & full fallback to regular links
- ➔ Simply use **<nuxt-link>** component instead of **<a href>**

```
<template>
  <div class="search-link">
    <nuxt-link to="/search">Search</nuxt-link>
  </div>
</template>
```

drunomics

Nuxt Image module

- ➔ Automatic format conversion
- ➔ Generates required image sizes with varying providers
- ➔ Responsive breakpoints

```
<nuxt-img
    src="/img/hero.jpg"
    sizes="sm:100vw md:50vw lg:400px"
    format="webp"
    loading="lazy"
    placeholder
/>
```

drunomics

SVG Icons

- ➔ Nuxt icon module: **`npx nuxi add icon`**
- ➔ Easy usage of OSS Icones icon sets -> icones.js.org
- ➔ Easy to add custom icons via Vue components

```
<Icon
  name="uil:github"
  color="black"
  size="24px"
/>
```

drunomics

Webfonts - done properly.

- ➔ Nuxt fonts module: **`npx nuxi add nuxt-fonts`**
- ➔ Supports many fonts providers (Google, Adobe, Fontshare, ...)
- ➔ Local / Static hosting support
- ➔ Automatic font metric optimisation to avoid layout shifts during load

```
<style>
  div {
    font-family: 'Inter', sans-serif;
  }
</style>
```

drunomics

Nuxt Scripts

- ➔ Nuxt scripts module: **`npx nuxi add scripts`**
- ➔ Supports many 3rd-party scripts providers (Analytics, Maps, Video,...)

  see scripts.nuxt.com/scripts
- ➔ Loads after render, no blocking or delaying of Nuxt
- ➔ Provides facade components and integrates with user-consent
- ➔ Optionally bundles and hosts scripts on your server
- ➔ Rather new feature, still in “beta”

drunomics

Demo time

Launch yourself:

nuxt.com -> Examples -> Features ->Auto Imports

drunomics

So.. where are we?

- ➔ Everything is there, or in a nuxt module
- ➔ DX is great!
- ➔ Write HTML, CSS as usual

  - no JavaScript knowledge required!

drunomics

... but

aren't JavaScript frontends

bad

for performane

?

drunomics

Rendering modes

for JavaScript frontends

drunomics

Nuxt rendering modes (1/2)

- ➔ Client-side rendering (CSR)
  - ◆ No Nuxt-server needed
  - ◆ Generates simple, static **`index.html`**
  - ◆ Runs Nuxt in the browser
- ➔ Server side rendering (SSR)
  - ◆ Pre-renders HTML on the server
  - ◆ Solves SEO and initial load performance

drunomics

Nuxt rendering modes (2/2)

- ➔ Static Site Generation (SSG)
  - ◆ Pre-builds all pages at build time
  - ◆ Perfect for (smaller) content sites
- ➔ Hybrid rendering
  - ◆ Best of both worlds
  - ◆ Mix and match by route

drunomics

Understanding server-rendering

- ➔ Pages are pre-rendered on the server
- ➔ Page is transferred and rendered in the browser
- ➔ JavaScript loads
- ➔ Nuxt-app is loaded in the browser
  - ◆ The app is *hydrated*
- ➔ Nuxt-app runs in the browser

drunomics

What is hydration?

➔ Rendering the page is “replayed” in the browser

- Nuxt ships data-inputs as “payload” to the client
- App is re-run with payload
- Vue components are instanted and “mounted”
- App is active and running - “fully hydrated”

drunomics

Nuxt Features

Good to know what’s there

drunomics

Bundling magic

- ➔ Bundles JS, CSS efficiently
- ➔ Builds upon vite (or webpack)
- ➔ Automatic code splitting by routes or lazy-loaded components
- ➔ No configuration needed!

drunomics

Nuxt Concepts (1/2)

- ➔ Components: Vue
- ➔ Composables: Re-usable Vue helpers
- ➔ Pages: “Controller” component + auto-registered route
- ➔ Layouts: Switch “site-templates” with Header, Footer, …
- ➔ Server routes
- ➔ Nuxt plugins: Run code when Vue app starts

drunomics

Nuxt Concepts (2/2)

- ➔ Middleware: Run before navigating to a route
- ➔ Server middleware: Runs pre-request server-side
- ➔ Nuxt hooks: Hook into nuxt lifecycle
- ➔ Nuxt modules: Hooks, components, plugins, ...

drunomics

Extensible via Modules

- ➔ Ecosystem of OSS modules: nuxt.com/modules
- ➔ Official modules by Nuxt team,
  e.g. @nuxtjs/i18n, @nuxt/ui
- ➔ Many useful community modules, e.g.
  nuxt-jsonapi, nuxtjs-drupal-ce

Nuxt Layers

drunomics

- ➔ Re-usable nuxt-apps across projects
- ➔ Extend parent layers and add overrides
- ➔ Allows for re-usable themes or just splitting large projects

```
export default defineNuxtConfig({
  extends: [
    '../base-layer',          // local layer
    '@my-themes/blue',        // npm package
    'github:org/layer#main' // GitHub source
]})
```

Nitro Server Engine

- ➔ Powers Nuxt's server capabilities
- ➔ Server routes and API endpoints
- ➔ Serverless/Edge compatible
- ➔ Platform-agnostic deployment
- ➔ Part of UnJS ecosystem: Collection of JavaScript utilities
- ➔ Created by the Nuxt team
- ➔ Becoming popular, used by Astro, Solid-Start, ...

drunomics

Nuxt + Drupal

or how to actually use the thing

drunomics

Custom data fetching

```
<script setup>
const { data: articles } = await useFetch('/api/articles')
</script>

<template>
  <article v-for="article in articles" :key="article.id">
    <h2>{{ article.title }}</h2>
    <p>{{ article.summary }}</p>
  </article>
</template>
```

drunomics

Lupus Decoupled Drupal

- ➔ Complete, fully integrated solution with Nuxt frontend
- ➔ Renders Drupal pages into high-level frontend components (Custom Elements), served as a JSON response
- ➔ Keeps Drupal in control of paths, metadata, caching,..
- ➔ Low-code: Configure Drupal API output - no PHP required!
- ➔ Supports Views, Layout builder, Webform, Multi-Lingual, ...

More at lupus-decoupled.org !

drunomics

Deployment options

- Server:
  - Node.js runtime
  - Container deployments (Docker)
- Serverless
  - Vercel, Netlify, Cloudflare, Nuxthub
  - AWS Lambda, Azure Functions
- Static
  - Your webserver (Apache, Nginx, ...)
  - Netlify, Vercel, Cloudflare Pages

druonomics

Deployment presets

- ➔ Lots of ready to go presets
- ➔ Check out nuxt.com/deploy

```
// nuxt.config.ts
export default defineNuxtConfig({
  nitro: {
    preset: 'vercel' // or: netlify, cloudflare, node, etc.
  }
})
```

drunomics

Performance optimization

➔ Lazy-load heavy components

```
<LazyMyHeavyComponent v-if="showHeavy" />
```

➔ Delay hydration of components

```
<LazyHeavyComponent hydrate-on-visible />

<LazyMobileMenu hydrate-on-media-query="(max-width: 768px)" />
```

drunomics

Debugging

- ➔ Nuxt Dev Tools!
- ➔ Built-in solution

```
export default defineNuxtConfig({
  devtools: { enabled: true }
})
```

- ➔ Comes with Component inspector, API visualization, Payload inspection, Performance analysis, ...

drunomics

Testing: Nuxt-Test-Utils

- ➔ Unit testing with Vitest → Test components!
- ➔ End-to-end testing with Playwright
- ➔ Provides various handy helpers!

Read more at nuxt.com/docs → Testing

Example Test

drunomics

```
// tests/components/SomeComponents.nuxt.spec.ts
import { mountSuspended } from '@nuxt/test-utils/runtime'
import App from '~/app.vue'

// tests/App.nuxt.spec.ts
it('can also mount an app', async () => {
    const component = await mountSuspended(App, { route: '/test' })
    expect(component.html()).toMatchInlineSnapshot(`
      "<div>This is an auto-imported component</div>
      <div> I am a tested component </div>
    `)
})
```

drunomics

True Open Source

- ➔ Vue and Nuxt are true open source
- ➔ Community-driven development
- ➔ Accessibility and progressive enhancement

Resources

- ➔ Nuxt.com
  - ◆ Docs, Modules, Examples, ...
  - ◆ API reference: nuxt.com/docs/api
- ➔ Code & Releases: github.com/nuxt/nuxt
- ➔ https://vuejs.org
- ➔ https://learn.nuxt.com
- ➔ https://masteringnuxt.com

Thank you!