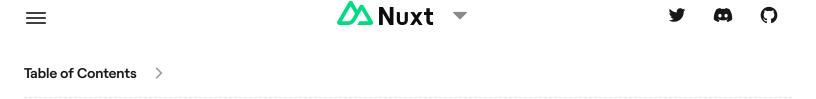
Register 7



Views

Nuxt provides several component layers to implement the user interface of your application.

app.vue



By default, Nuxt will treat this file as the entrypoint and render its content for every route of the application.

```
<template>
  <div>
     <h1>Welcome to the homepage</h1>
     </div>
  </template>
```

If you are familiar with Vue, you might wonder where main.js is (the file that normally creates a Vue app). Nuxt does this behind the scene.

Components



Most components are reusable pieces of the user interface, like buttons and menus. In Nuxt, you can create these components in the components/ directory, and they will be automatically available across your application without having to explicitly import them.

Pages



Pages represent views for each specific route pattern. Every file in the pages/directory represents a different route displaying its content.

To use pages, create pages/index.vue file and add <NuxtPage /> component to the app.vue (or remove app.vue for default entry). You can now create more pages and their corresponding routes by adding new files in the pages/ directory.

You will learn more about pages in the Routing section

Layouts



Layouts are wrappers around pages that contain a common User Interface for several pages, such as a header and footer display. Layouts are Vue files using <slot /> components to display the **page** content.

The layouts/default.vue file will be used by default. Custom layouts can be set as part of your page metadata.

If you only have a single layout in your application, we recommend using app.vue with the NuxtPage /> component instead.

layouts/default.vue pages/index.vue pages/about.vue

```
<template>
  <div>
        <AppHeader />
        <slot />
        <AppFooter />
        </div>
        </template>
```

If you want to create more layouts and learn how to use them in your pages, find more information in the Layouts section.

Advanced: Extending the HTML template

If you only need to modify the head, you can refer to the SEO and meta section.

You can have full control over the HTML template by adding a Nitro plugin that registers a hook. The callback function of the render:html hook allows you to mutate the HTML before it is sent to the client.

```
export default defineNitroPlugin((nitroApp) => {
    nitroApp.hooks.hook('render:html', (html, { event }) => {
        // This will be an object representation of the html template.
        console.log(html)
        html.head.push(`<meta name="description" content="My custom description" />`)
    })
    // You can also intercept the response here.
    nitroApp.hooks.hook('render:response', (response, { event }) => { console.log(response) })
})
```

△

© 2016-2023 Nuxt - MIT License Enterprise

Design Kit

Nuxt Studio

NuxtLabs

INGACEG



