storyblok

Nuxt 3

CheatSheet

by Alex Jover & Manuel Schröder

Learn more

Read the 5 minute tutorial on how to add Storyblok to Nuxt:
https://www.storyblok.com/tp/add-a-headless-CMS-to-nuxt-3-in-5-minutes

Build a full-blown, multilingual website by following the Ultimate Tutorial Series:
https://www.storyblok.com/tp/storyblok-nuxt-ultimate-tutorial

1. Setup

Start by creating a Nuxt app and installing the Storyblok Nuxt SDK:

```
npx nuxi init <project-name>

npm install @storyblok/nuxt
```

nuxt.config.js

2. Configuring the Storyblok Module

Add the preview token of your Storyblok space in the `nuxt.config.js` file:

```
modules: [
  ['@storyblok/nuxt', { accessToken: '<your-access-token-
here>' }],
],
```

3. Creating Storyblok Components

Create your components in a `storyblok` folder – the Storyblok Nuxt module finds them automatically. To enable live editing, pass the `v-editable` property to each component. Example:

```
<template>
  <div v-editable="blok">
    {{ blok.headline }}
  </div>
</template>

<script setup>
defineProps({ blok: Object })
</script>
```

storyblok/Teaser.vue

4. Getting Content from the API and Listening to the Visual Editor live changes

To fetch the content of the Home story, you can use the auto-imported `useStoryblok`. It will automatically fetch the story content and listen for the Visual Editor live changes.

Optionally, you can use the `language` parameter to fetch the story in a particular language:

```
<script setup>
const story = await useStoryblok('home', {
  version: 'draft',
  language: 'es',
  fallback_lang: 'default',
})
</script>

<template>
  <StoryblokComponent v-if="story" :blok="story.content" />
</template>
```

pages/index.vue

To learn more about `useStoryblok`, check out our docs on Github: https://github.com/storyblok/storyblok-nuxt

To learn more about localization and translation, check out our article on internationalization: https://www.storyblok.com/docs/guide/in-depth/internationalization

5. Rendering Pages dynamically

Render pages dynamically by using the following code in `pages/[...slug].vue`:

```
<script setup>
const { slug } = useRoute().params

const story = await useStoryblok(slug ? slug : 'home', {
  version: 'draft',
})
</script>

<template>
  <StoryblokComponent v-if="story" :blok="story.content" />
</template>
```

pages/[...slug].vue

6. Rendering Nested Components dynamically

Use the provided `<StoryblokComponent />` to render nested components dynamically:

```
<template>
  <div v-editable="blok">
    <StoryblokComponent
      v-for="blok in blok.columns"
      :key="blok._uid"
      :blok="blok"
    />
  </div>
</template>

<script setup>
defineProps({ blok: Object })
</script>
```

storyblok/Grid.vue