Nuxt.js Cheat Sheet

A comprehensive cheat sheet covering essential Nuxt.js concepts, configurations, components, and deployment strategies for building universal Vue.js applications.

Nuxt.js Fundamentals

Project Setup

Create a new Nuxt.js project:

```
npx create-nuxt-app <project-name>
```

Follow the prompts to configure your project (e.g., UI framework, modules).

Navigate to the project directory:

```
cd <project-name>
```

Run the development server:

```
npm run dev
# or
yarn dev
```

Directory Structure

`pages/`	Contains the application's routes/views. Each `.vue` file becomes a route based on its filename.
`layouts/`	Defines the application's layout. `default.vue` is the default layout.
`components/`	Reusable Vue components.
`store/`	Vuex store files (state management).
`nuxt.config.js`	Main configuration file for Nuxt.js.
`static/`	Files served directly (e.g., `favicon.ico`, `robots.txt`).

Key Concepts

Automatic Routing: Nuxt.js automatically generates routes based on the files in the `pages/` directory.

Server-Side Rendering (SSR): Nuxt.js renders Vue components on the server before sending them to the client, improving SEO and initial load time.

Single Page Application (SPA): Nuxt.js can also be configured as a SPA application, creating only client side rendered apps.

Vuex Integration: Seamless integration with Vuex for state management.

Middleware: Custom functions that run before rendering a page. Useful for authentication, etc.

Modules: Extend Nuxt.js core functionality with pre-built modules (e.g., `axios`, `vuetify`).

Routing & Pages

Basic Routing

Create `.vue` files in the `pages/` directory to define routes.

`pages/index.vue` becomes the root route `/`. `pages/about.vue` becomes the `/about` route.

Use the `<nuxt-link>` component for client-side navigation:

```
<nuxt-link to="/about">About Us</nuxt-link>
```

Dynamic Routes

Create dynamic routes using a leading underscore in the filename.

`pages/posts/_id.vue` will match routes like `/posts/1`, `/posts/2`, etc.

Access the dynamic route parameter using `this.$route.params.id` in your component.

```
<template>
  <div>Post ID: {{ $route.params.id }}</div>
</template>
```

Programmatic Navigation

Use `this.$router.push()` to navigate programmatically.

```
this.$router.push('/about')
```

Navigate with a named route:

```
this.$router.push({ name: 'about' })
```

Navigate with parameters:

```
this.$router.push({ name: 'posts-id', params: { id: 123 } })
```

Middleware in Pages

Define middleware in your page component:

```
export default {
  middleware: 'auth',
  // ...
}
```

Create middleware files in the `middleware/` directory (e.g., `middleware/auth.js`):

```
export default function ({ redirect }) {
  // If the user is not authenticated
  if (!isAuthenticated()) {
    return redirect('/login')
  }
}
```

Layouts & Components

Layouts

Create layouts in the `layouts/` directory.

`layouts/default.vue` is the default layout applied to all pages.

Use `<nuxt />` to render the page content within a layout.

```
<template>
  <div>
    <header>My App Header</header>
    <nuxt />
    <footer>My App Footer</footer>
  </div>
</template>
```

Define a custom layout for a page:

```
export default {
  layout: 'custom'
}
```

Components

Create reusable components in the `components/` directory.

Import and use components in your pages and layouts.

```
<template>
  <div>
    <MyComponent />
  </div>
</template>

<script>
import MyComponent from
'~/components/MyComponent.vue'

export default {
  components: {
    MyComponent
  }
}
</script>
```

Head Management

Manage the document `<head>` using the `head` property in your pages and layouts.

```
export default {
  head () {
    return {
      title: 'My Page Title',
      meta: [
        { hid: 'description', name:
'description', content: 'Page description' }
      ]
    }
  }
}
```

Data Fetching & API

Async Data

Use the `asyncData` method to fetch data before rendering a page (server-side).

```
export default {
  async asyncData ({ $axios, params }) {
    const { data } = await
$axios.$get(`/posts/${params.id}`)
    return { post: data }
  }
}
```

`asyncData` is only available in pages.

Fetch Hook

Use the `fetch` hook to fetch data client-side or server-side.

```
export default {
  data() {
    return {
      posts: []
    }
  },
  async fetch() {
    this.posts = await
this.$axios.$get('/posts')
  }
}
```

Nuxt Axios Module

The `@nuxtjs/axios` module simplifies making HTTP requests.

Install:

```
npm install @nuxtjs/axios
# or
yarn add @nuxtjs/axios
```

Add it to `modules` section in `nuxt.config.js`:

```
modules: [
  '@nuxtjs/axios'
],

axios: {
  baseURL: 'https://api.example.com'
}
```

Use `$axios` to make requests:

```
async mounted () {
  const { data } = await
this.$axios.get('/users')
  this.users = data
}
```