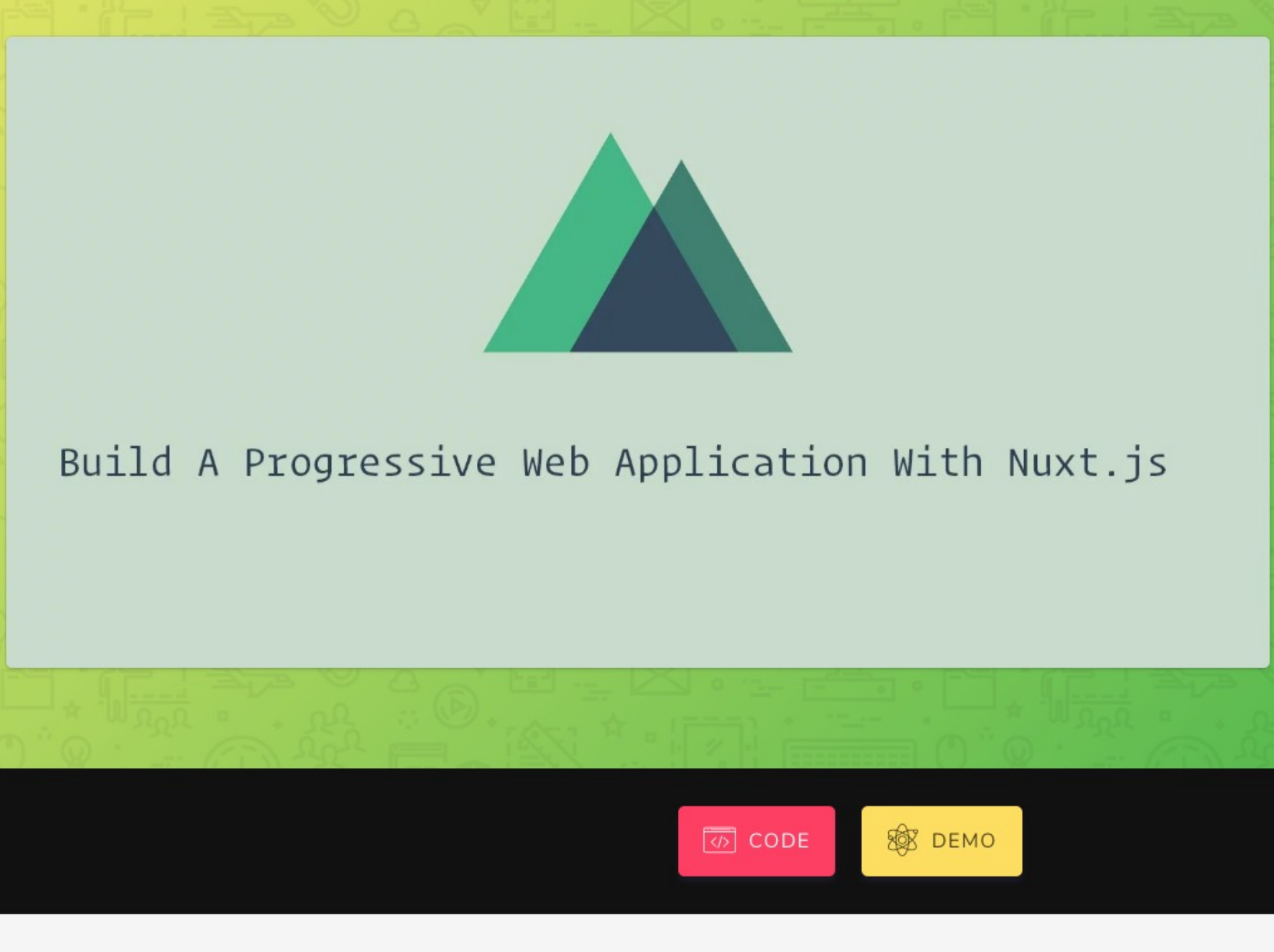
Build A Progressive Web Application With Nuxt.js




Chimezie Enyinnaya
Follow @ammezie 576 followers

June 06
#vue
2 COMMENTS
3,041 VIEWS

CODE DEMO

Ever since its announcement by Google, the adoption of progressive web apps has skyrocketed as many traditional web apps have been and are being converted to progressive web apps. In this tutorial, I'll be showing you how to build a progressive web app with Nuxt.js. For the purpose of demonstration, we'll be building a news app.

This tutorial assumes a basic knowledge of progressive web app.

Nuxt.js is a framework for building server-side rendered Vue.js applications.

Table of Contents

Getting started

We'll start by creating a new Nuxt.js app. For this, we'll make use of the Vue CLI, so you need to first install the Vue CLI in case you don't have it installed already:

```bash
npm install -g vue-cli
```

Then we can create a Nuxt.js app:

```bash
vue init nuxt/starter pwa-news
```

Next, we need to install the dependencies:

```bash
cd pwa-news
npm install
```

We can now launch our app:

```bash
npm run dev
```

The app should be running on http://localhost:3000.

With our app up and running, let's now install the necessary Nuxt.js modules that we'll be needing for our news app:

```bash
npm install @nuxtjs/axios @nuxtjs/bulma @nuxtjs/dotenv
```

Let's quickly go over each of the module:

> Related Course: Build an Online Shop with Vue

- **@nuxtjs/axios:** Secure and easy Axios integration with Nuxt.js.
- **@nuxtjs/bulma:** This module leverages the lovely Bulma CSS framework.
- **@nuxtjs/dotenv:** This module loads your `.env` file into your context options.

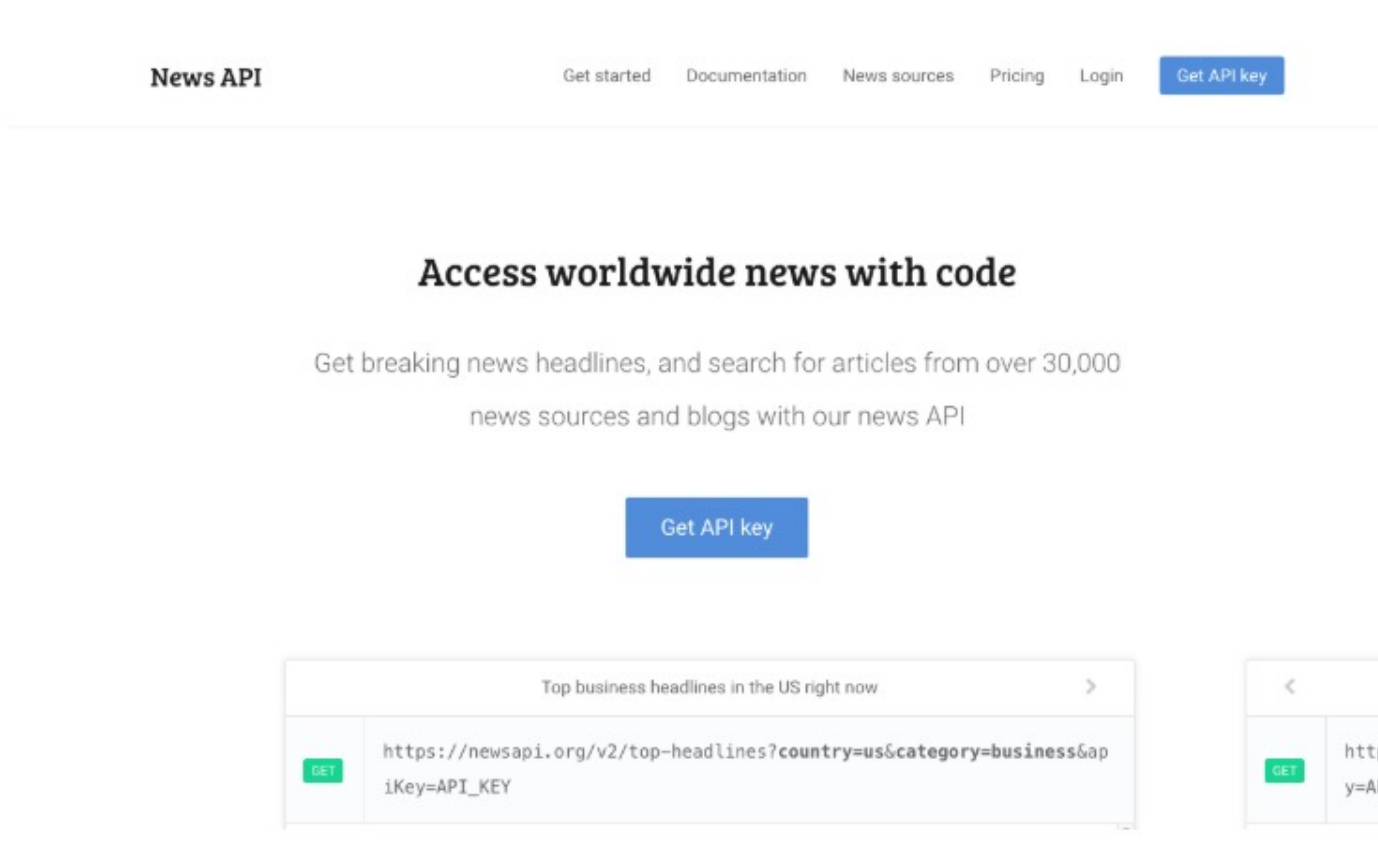
Next, let's make Nuxt.js use these modules. We'll do that by adding them in the `modules` section of the `nuxt.config.js` file:

```javascript
// nuxt.config.js

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

Getting our API key

Our news app will be built on News API. So we need to get our API key.



Access worldwide news with code

Get breaking news headlines, and search for articles from over 30,000 news sources and blogs with our news API

https://newsapi.org/v2/top-headlines?country=us&category=business&apiKey=API_KEY

Click on the **Get API key** button then follow along the registration to get your API key.

Building the news app

With our API key in place, let's start building our news app. First, let's update the `layouts/default.vue` file as below:

```html
// layouts/default.vue

<template>
    <div>
        <section class="hero has-text-centered is-primary">
            <div class="hero-body">
                <div class="container">
                    <h1 class="title">PWA News</h1>
                    <h2 class="subtitle">All the headlines making the wavy!</h2>
                </div>
            </div>
        </section>
        <nuxt/>
    </div>
</template>

<style>
    html {
        font-family: 'Source Sans Pro', -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'He
    }
</style>
```

We are simply making use of the Bulma classes.

Next, let's update the `pages/index.vue` file as well:

```javascript
// pages/index.vue

<template>
    <section class="section">
        <div class="container">
            <div class="columns is-multiline">
                <div
                    class="column is-one-quarter"
                    v-for="(article, index) in articles"
                    :key="index">
                        <a :href="article.url" target="_blank">
                            <div class="card">
                                <div class="card-image">
                                    <figure class="image is-3by2">
                                        <img :src="article.urlToImage" :alt="article.title">
                                    </figure>
                                </div>
                                <div class="card-content">
                                    <div class="content">{{ article.title }}</div>
                                </div>
                            </div>
                        </a>
                </div>
            </div>
        </div>
    </section>
</template>

<script>
    export default {
        async asyncData({ app }) {
            const { articles } = await app.$axios.$get(
                `https://newsapi.org/v2/top-headlines?sources=cnn&apiKey=${
                    process.env.API_KEY
                }`
            );

            return { articles };
        },
    };
</script>
```

In the `template` section, we loop through the news headlines and display each headline in a card (using Bulma classes) with a link to view the news directly on the source. On the `script` section, because we'll be fetching the news headlines and rendering them on the server-side, so we make use of the `asyncData` method. Then making use of the Nuxt.js axios module installed earlier, we make a `GET` request to the New API endpoint to fetch news headlines, passing along the source we want to fetch from and our API key. Lastly, we return a `articles` object containing the fetched news headlines. With this, we can access the `articles` object as we would access any other component `data`.

You will notice we are getting our API key from an environment variable, which we are yet to create. Let's do that now. Create a new `.env` file directly in the project root directory:

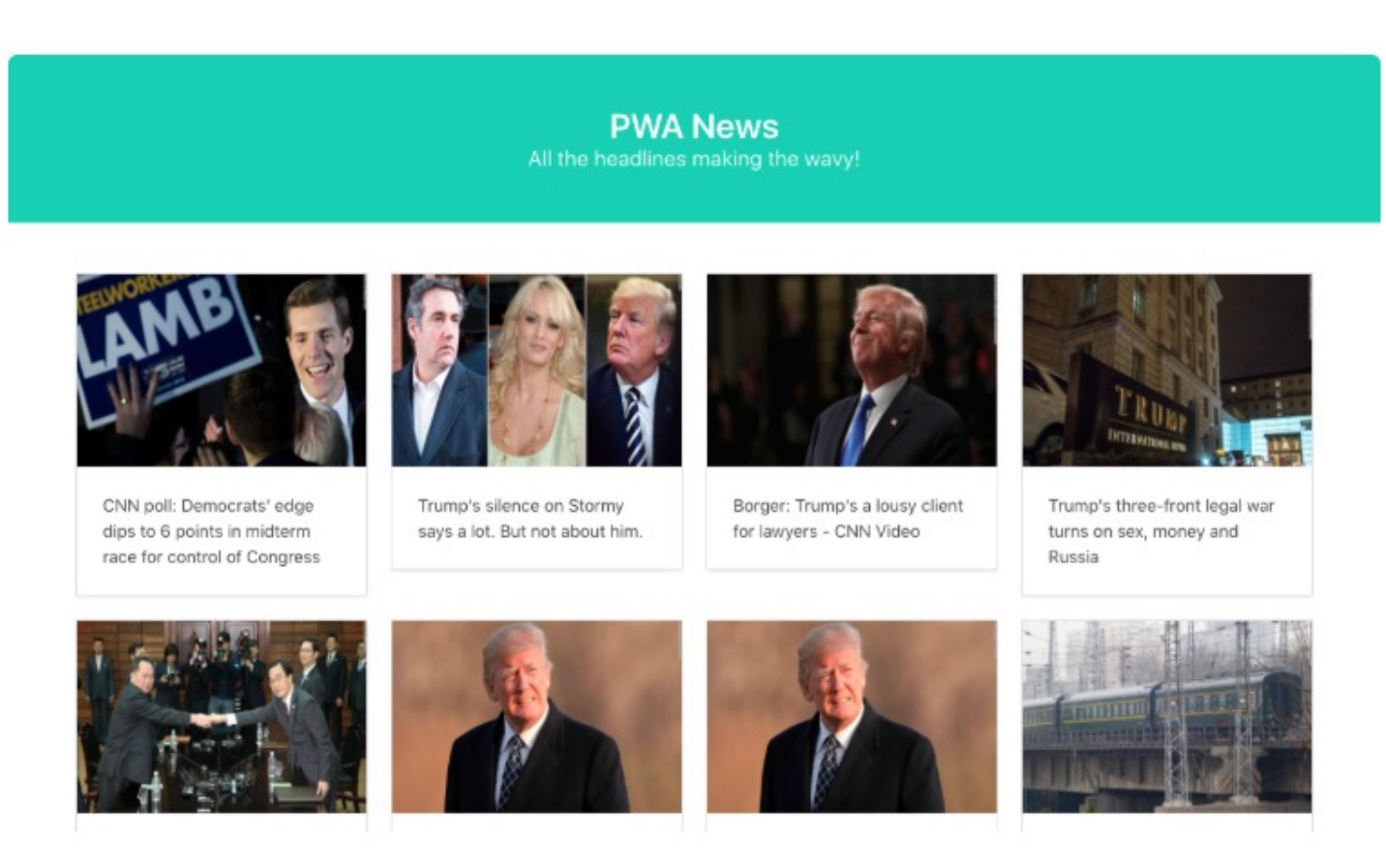
```bash
touch .env
```

Then add the code below into it:

```txt
// .env

API_KEY=YOUR_API_KEY
```

Now, if we test our app, we should get something similar to the image below:



Converting our app to a progressive web app

So far our news is feature complete. But our goal for this tutorial is to build a progressive web app. So, let's add the progressive web app awesomeness to our news app. To do this, we make of a Nuxt.js module called Nuxt PWA.

> Using Nuxt PWA you can supercharge your current or next Nuxt project with a heavily tested, updated and stable PWA solution and zero-config!

Nuxt PWA module is a collection of smaller modules that are designed to magically work out of the box together. These modules includes:

- **Workbox** - Registers a service worker for offline caching.
- **Manifest** - Automatically generate `manifest.json` file.
- **Meta** - Automatically adds SEO friendly meta data with manifest integration.
- **Icon** - Automatically generates app icons with different sizes.
- **Onesignal** - Free background push notifications using onesignal.

For the purpose of this tutorial, we'll be making use of only the first 4 modules, as we won't be covering push notifications.

The lovely thing about Nuxt PWA is that it works straight out of the box with zero configuration. So let's install and set it up:

```bash
npm install @nuxtjs/pwa
```

Next, we add the module to the `nuxt.config.js` file:

```javascript
// nuxt.config.js

modules: [
    ...,
    '@nuxtjs/pwa'
]
```

Lastly, we need an icon for our news app. Though this is optional, it will give our app that native feel once our app is added to the home screen of our user's devices. For this tutorial, we'll use the icon from the Nuxt.js HackerNews clone. So download and place it in the `static` directory.

> Tips: It is recommended that the icon be a square png and >= `512x512px` and named `icon.png`.

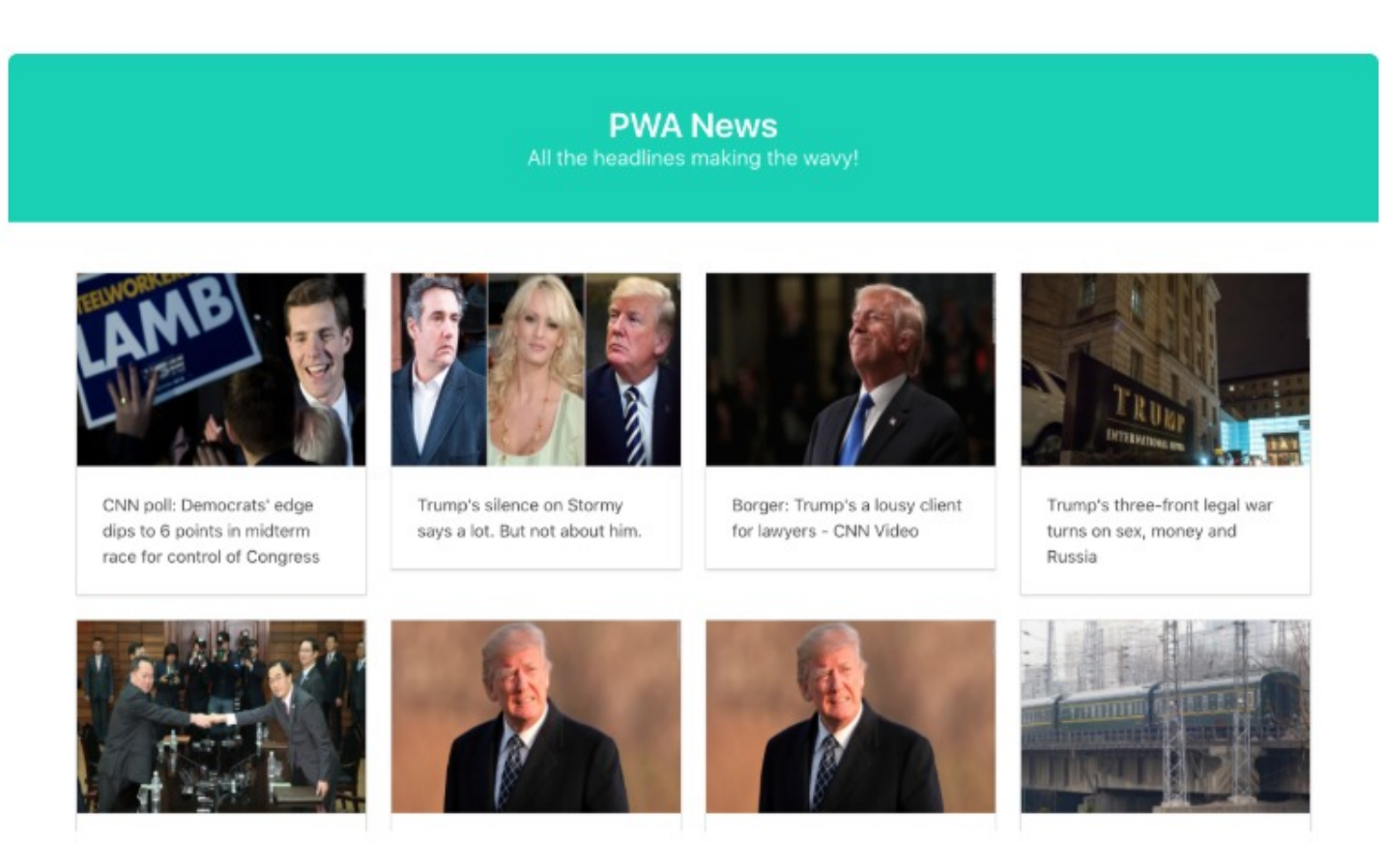
That's all we need to do to make use of the Nuxt PWA module. Because by default, workbox module is only enabled on `production` builds, we need to build our news app for production:

```bash
npm run build
```

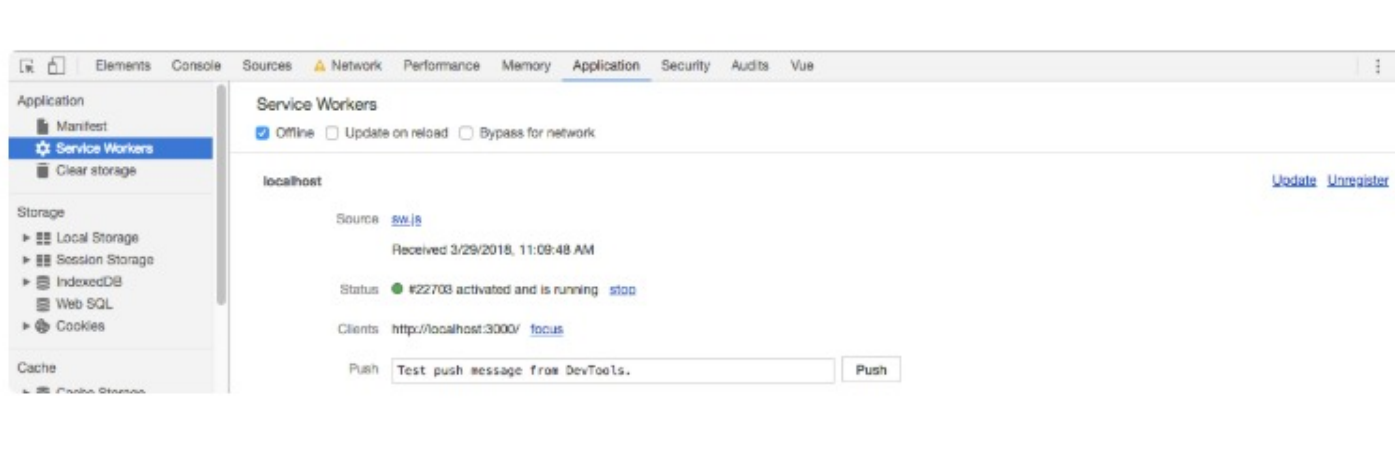
Once the the build is done, we can start our app with:

```bash
npm start
```

Then we can view the app at http://localhost:3000.



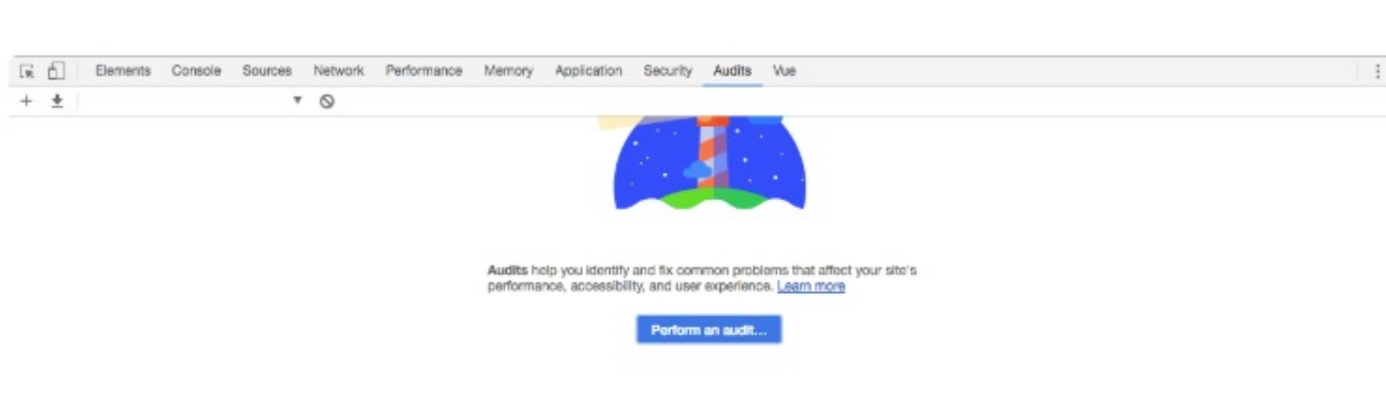
We get pretty much the same thing from earlier. To see indeed that our app is now a progressive web app, try setting our app to **offline** (which simulates no internet connection) under the **Application** tab on Chrome devtools and refresh the page.



localhost
Source sw.js
Received 3/29/2018, 11:09:48 AM
Status #22703 activated and is running stop
Clients http://localhost:3000/ focus
Push Test push message from DevTools.

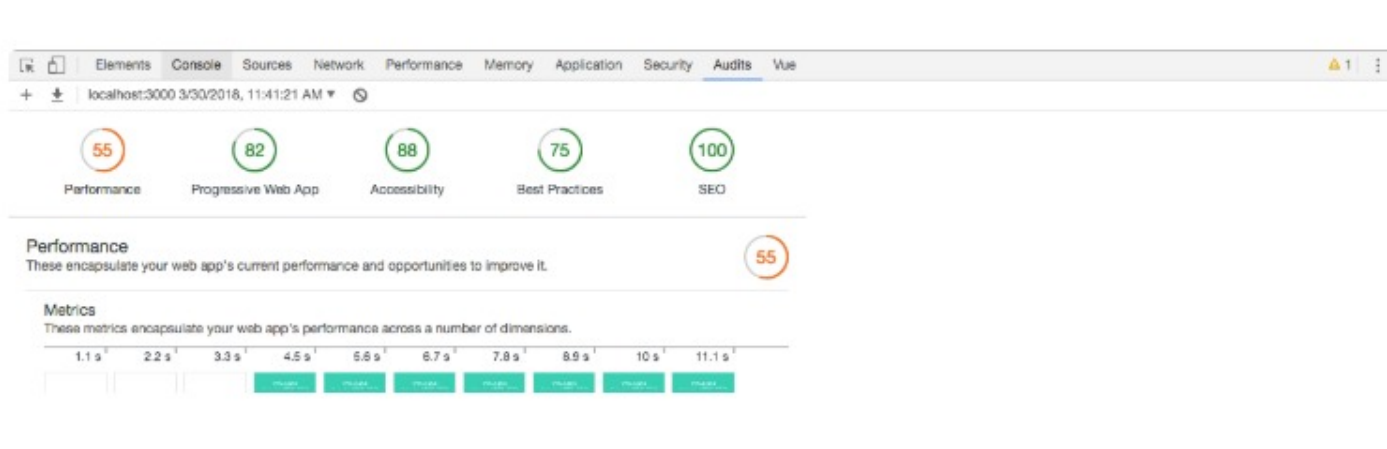
The app should still be running fine as our app is been cached for offline view.

Also, we can use the Lighthouse extension to test if our app meets the standards for a progressive web app. Under the **Audits** tab (you might have to download the Lighthouse extension if you can't find this tab) of Chrome devtools, click on the **perform an audit...** button and click on **Run aduit**:



Audits help you identify and fix common problems that affect your site's performance, accessibility, and user experience. Learn more

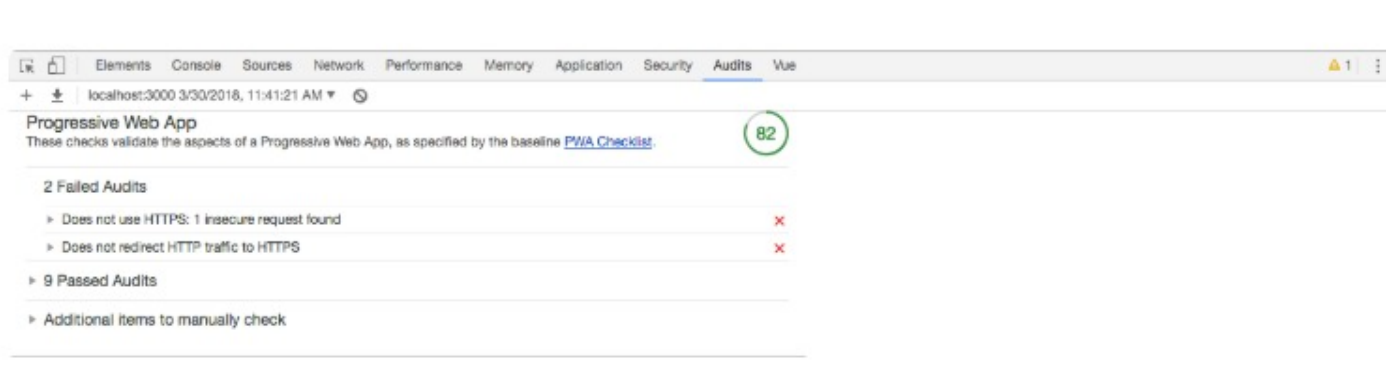
This will start performing some checks on our app. Once it's complete, we should get a screen as below:



Performance 55
These encapsulate your web app's current performance and opportunities to improve it.

Metrics
These metrics encapsulate your web app's performance across a number of dimensions.

We are more focused on the Progressive Web App section. As you can see, our app got an 82/100, which is a great score. If we narrow the result down, you'll see that our app only failed 2 audits, which are understandable since our app is still running on localhost.



Progressive Web App
These checks validate the aspects of a Progressive Web App, as specified by the baseline PWA Checklist.

2 Failed Audits
- Does not use HTTPS: 1 insecure request found ✕
- Does not redirect HTTP traffic to HTTPS ✕

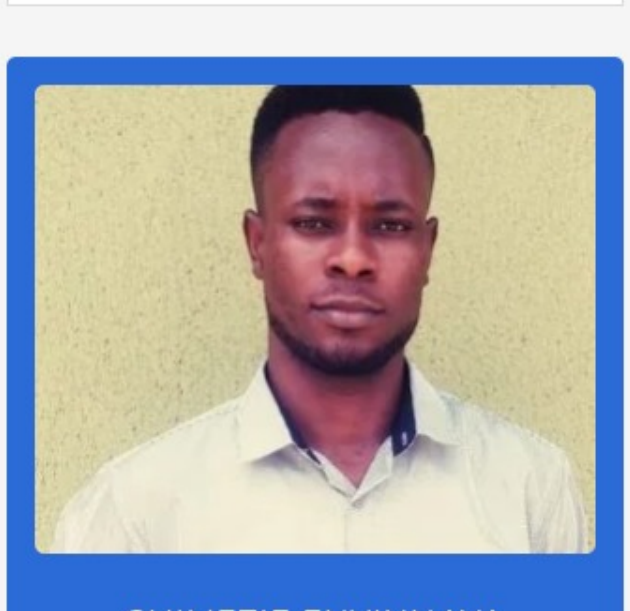
9 Passed Audits

Additional items to manually check

Conclusion

So in this tutorial, we looked at how we can build a progressive web app with Nuxt.js. With this tutorial, you should be able to convert an existing Nuxt.js application into a progressive web app. We also looked at how we can test a progressive web app using the Lighthouse extension. I hope you found this tutorial helpful.

CODE DEMO


CHIMEZIE ENYINNAYA
Software Developer [PHP Laravel JavaScript NodeJS AdonisJS VueJS] | movie lover | run
http://openlaravel.com
Follow @ammezie 576 followers
VIEW MY 19 POSTS

Code Demo 1 2


Chimezie Enyinnaya
19 posts
Software Developer [PHP Laravel JavaScript NodeJS AdonisJS VueJS] | movie lover | run
http://openlaravel.com