**url:** [**https://www.digitalocean.com/community/tutorials/vuejs-implementing-infinite-scroll**](https://www.digitalocean.com/community/tutorials/vuejs-implementing-infinite-scroll)

**How To Implement an Infinite Scroll with Vue.js**

Published on April 3, 2018 · Updated on March 24, 2021

* [Vue.js](https://www.digitalocean.com/community/tags/vue-js)



By [Dave Berning](https://www.digitalocean.com/community/users/daveberning)

Developer and Author



**Introduction**

Infinite scrolling is a feature on websites and applications where a user scrolls down and reaches the bottom of the current page of content and then the next page of content is loaded and displayed. This effect replaces clicking through pagination navigation. In situations with mobile devices and touchscreens, infinite scrolling may present a better user experience.

This feature is especially useful when you need to load large amounts of data or images when the user needs them rather than at all once. Social media outlets like Twitter, Facebook, and Instagram have popularized this over the years.

In this tutorial, you will build an example Vue.js application that uses infinite scrolling for fetching and displaying data from the [Random User API](https://randomuser.me/).

**Prerequisites**

To complete this tutorial, you will need:

* Node.js installed locally, which you can do by following [How to Install Node.js and Create a Local Development Environment](https://www.digitalocean.com/community/tutorial_series/how-to-install-node-js-and-create-a-local-development-environment).
* Some familiarity with [setting up a Vue.js project](https://vuejs.org/v2/guide/installation.html) and [using Vue.js components](https://vuejs.org/v2/guide/#Composing-with-Components) may be beneficial.

This tutorial was verified with Node v15.3.0, npm v6.14.9, vue v2.6.11, and axios v0.21.0. This tutorial was edited to reflect changes in migrating from earlier versions of @vue/cli.

**Step 1 — Setting Up the Project**

For the purpose of this tutorial, you will build from a default Vue project generated with @vue/cli.

1. npx @vue/cli create vue-infinite-scrolling-example --default

Copy

This will configure a new Vue project with default configurations: Vue 2, babel, eslint.

Navigate to the newly created project directory:

1. cd vue-infinite-scrolling-example

Copy

Next, install axios:

1. npm install axios@0.21.0

Copy

At this point, you should have a new Vue project with Axios support.

**Step 2 — Getting Initial User Data**

There are various npm packages for an infinite scroll that you can use for your Vue app. But some of these may be overkill for your needs due to features you do not require or a large number of dependencies.

For the purposes of this tutorial, you will build a JavaScript function that fetches a new set of data when scrolled to the bottom of the browser window. This will not require additional plugins or packages.

First, open App.vue in your code editor.

Next, replace the code in the template with a display that loops over an array of users to display a picture, first name, last name, date of birth, city, and state:

src/App.vue

<template>

<div id="app">

<h1>Random User</h1>

<div

class="user"

v-for="user in users"

:key="user.first"

>

<div class="user-avatar">

<img :src="user.picture.large" />

</div>

<div class="user-details">

<h2 class="user-name">

{{ user.name.first }}

{{ user.name.last }}

</h2>

<ul>

<li><strong>Birthday:</strong> {{ formatDate(user.dob.date) }}</li>

<li><strong>Location:</strong> {{ user.location.city }}, {{ user.location.state }}</li>

</ul>

</div>

</div>

</div>

</template>

Then, replace the the code in the <style> with CSS rules for arranging the display of each user:

src/App.vue

<style>

.user {

display: flex;

background: #ccc;

border-radius: 1em;

margin: 1em auto;

}

.user-avatar {

padding: 1em;

}

.user-avatar img {

display: block;

width: 100%;

min-width: 64px;

height: auto;

border-radius: 50%;

}

.user-details {

padding: 1em;

}

.user-name {

margin: 0;

padding: 0;

font-size: 2rem;

font-weight: 900;

}

</style>

Next, replace the code in the <script> with code that makes an initial request of five users to the API and is called during the beforeMount lifecycle:

src/App.vue

<script>

import axios from "axios";

export default {

data() {

return {

users: [],

};

},

methods: {

getInitialUsers() {

axios.get(`https://randomuser.me/api/?results=5`).then((response) => {

this.users = response.data.results;

});

},

},

beforeMount() {

this.getInitialUsers();

},

};

</script>

The date of birth (DOB) is provided as a string in [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601) format. In order to make this more human-readable, you can convert the string into a date string using Date.prototype.toDateString():

src/App.vue

<script>

// ...

export default {

// ...

methods: {

formatDate(dateString) {

let convertedDate = new Date(dateString);

return convertedDate.toDateString();

}

// ...

},

// ...

};

</script>

This initial request will display five users when a user opens the application.

**Note:** Previously, this tutorial performed multiple requests to the Random User API to initially load more than the single user result. This section has been rewritten to use the new results parameter that is provided by the API.

In your terminal window, compile and serve the application:

1. npm run serve

Copy

After opening the application in your web browser, there will be five random users displayed.

**Step 3 — Implementing the Infinite Scroll Logic**

The infinite scroll logic will require detecting when the user has reached the bottom of the window. This can be accomplished with the following three properties:

* document.documentElement.offsetHeight: the amount of pixels for the entire height of the document element.
* document.documentElement.scrollTop: the current amount of pixels positioned from the top of the document element.
* window.innerHeight: the number of pixels for the height of the screen.

When document.documentElement.scrollTop plus window.innerHeight are equal to document.documentElement.offsetHeight, it can be assumed that the user has reached the bottom of the window.

window.onscroll = () => {

let bottomOfWindow = document.documentElement.scrollTop + window.innerHeight === document.documentElement.offsetHeight;

if (bottomOfWindow) {

// ...

}

};

Copy

Here, window.onscroll listens for the scroll event and will perform the check when the event is detected.

**Note:** When binding events, especially to scroll events, it is good practice to *debounce* the events. Debouncing is when you only run a function once a specified amount of time has passed since it was last called.

However, for the needs of this tutorial, a debouncer will not be applied.

Inside of the if condition, let’s add a GET service method with Axios to fetch another random user from the Random User API:

axios.get(`https://randomuser.me/api/`).then(response => {

this.users.push(response.data.results[0]);

});

Copy

Now, revisit App.vue in your text editor and add your new code.

In your component’s methods, you will need to create a new function called, getNextUser() and have that loaded in the mounted() lifecycle method.

src/App.vue

<script>

import axios from "axios";

export default {

data() {

return {

users: [],

};

},

methods: {

formatDate(dateString) {

let convertedDate = new Date(dateString);

return convertedDate.toDateString();

},

getInitialUsers() {

axios.get(`https://randomuser.me/api/?results=5`).then((response) => {

this.users = response.data.results;

});

},

getNextUser() {

window.onscroll = () => {

let bottomOfWindow = document.documentElement.scrollTop + window.innerHeight === document.documentElement.offsetHeight;

if (bottomOfWindow) {

axios.get(`https://randomuser.me/api/`).then(response => {

this.users.push(response.data.results[0]);

});

}

}

}

},

beforeMount() {

this.getInitialUsers();

},

mounted() {

this.getNextUser();

}

}

</script>

Now, recompile and serve your application.

After opening the application in your web browser and scrolling to the bottom of the page, a new user is added to the page.

With each scroll to the bottom of the page, you fetch new data with Axios then push that data to an array.

**Conclusion**

In this tutorial, you built an implementation of an infinite scroll in a Vue.js application. It relied upon beforeMount and mounted lifecycle hooks to initialize and prefetch the requests to an API.

To lazy load images, push an image source to a data array, iterate through it in your template and bind your <img :src=""> to the array.

If you’d like to learn more about Vue.js, check out [our Vue.js topic page](https://www.digitalocean.com/community/tags/vue-js) for exercises and programming projects.