Guides

How to build and deploy a PWA with Next.js

Damilola Ezekiel Published on July 18, 2022

Progressive Web Apps (PWAs) are websites that are built with web technologies and behave like apps. By utilizing the latest technologies, PWAs combine the benefits of web and mobile apps. Many of the features of PWAs make them similar to native applications in terms of user experience.

Advantages of Progressive Web Apps

- Availability in the offline mode
- Enhanced performance
- It's quick and easy to install

- Fast loading
- PWAs are more secure when you compare them to normal website apps because they have to run with HTTPS.

In this article, we will be building a Progressive Web App with Next.js and then deploying it to a server.

Prerequisites

To follow along with this article, here are the things you'll need:

- Knowledge of React.js and Next.js.
- Knowledge of Git and Github.
- Have node installed on your system.

Getting Started

To start a new Next.js project, it is recommended to use create-next-app because it automatically sets up everything. You'll need the run the command below

```
//using npm
npx create-next-app demo-app

//using yarn
yarn create next-app
```

The above command creates a new Next.js project named demo-app in your computer.

Next, you'll move into the project folder by running this command:

```
cd demo-app
```

To start your Next.js app in the Development server, run this command:

```
// using yarn
yarn dev
```

Now that our Next.js is up and running, we can start building. We will be building a random advice generator app using the Advice Slip JSON API.

Index.js

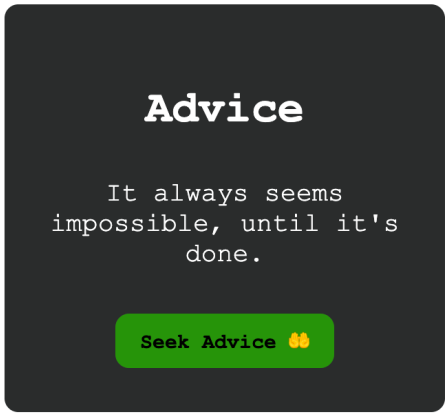
This is where all the logic of our app happens. This file contains three functions, the first one is the function that calls the API, the second one is the default function for the page and the last one is a function that helps to get data from the API whenever we call it. Then we return a JSX element that displays the data from the API.

```
import { useState } from "react";
import styles from "../styles/Home.module.css";


//fetching the data from API
async function fetchData() {
  const response = await fetch("https://api.adviceslip.com/adv
  const data = await response.json();

  return { data };
}

export default function Home(props) {
  const[data, setData] = useState(props.data);

  // function to reload the API after every call

  async function refresh() {
    const refreshedProps = await fetchData();
    setData(refreshedProps.data);
  }

  return (
    <div className={styles.App}>
      <h2 className={styles.headerText}> Advice </h2>
```

```
      </button>
</div>
  );
}


Home.getInitialProps = fetchData;
```

We will also add a little bit of CSS for basic styling. The css code needs to be added to the home.modules.css file, otherwise it won't work.

```
.App {
  font-family: 'Courier New', Courier, monospace;
  text-align: center;
  width: 20rem;
  height: auto;
  padding: 2rem;
  background-color: #2a2c2c;
  border-radius: 10px;
  margin: 10rem auto;
}
.headerText{
  font-size: 2rem;
  color: #fff;
}
.paragraph{
  margin: 2rem 0;
  color: #fff;
  font-size: 1.2rem;
}
.button {
  background-color: #259409;
  padding: 0.5rem 1rem;
  border-radius: 10px;
  border: 2px solid #259409;
  font-size: 15px;
  font-weight: 600;
  font-family: 'Courier New', Courier, monospace;
}
```



Converting the Nextjs App to PWA

To turn our Next.js app into a progressive web app, we need to follow the steps below.

- Install the Progressive Web App dependency
- Create or generate a manifest file
- Create a document.js file
- Configure the next.config.js file

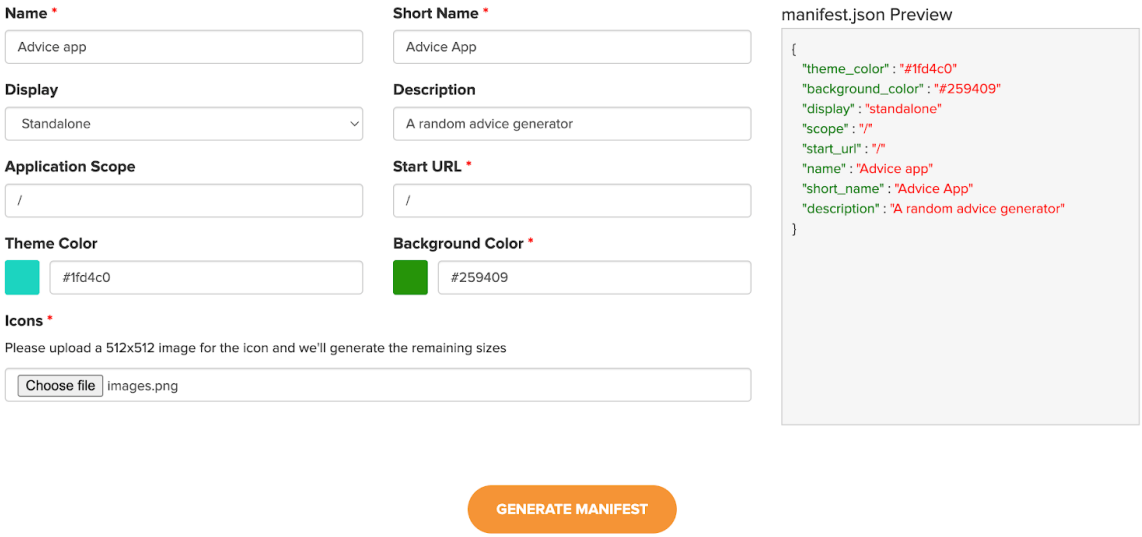
Installing the PWA dependency

The first step is to install the PWA dependency and you can check out the README for the package here:

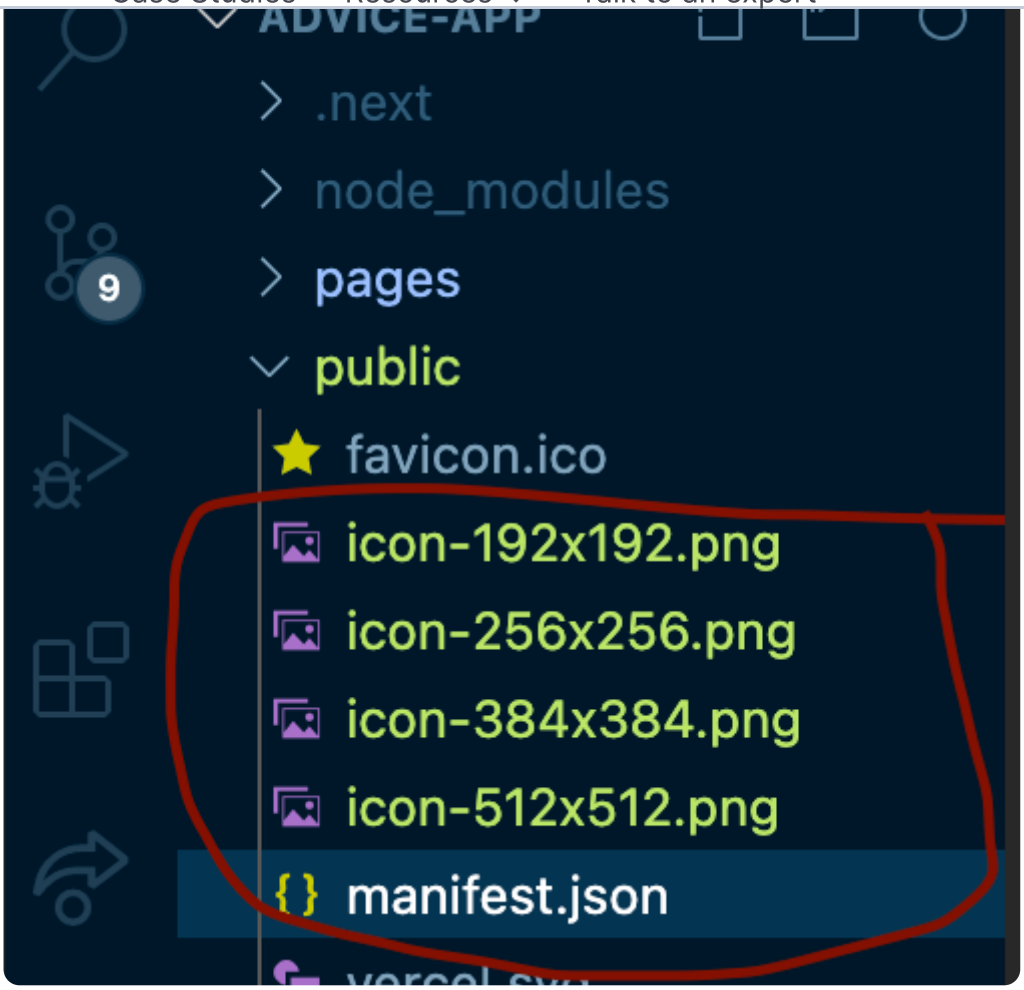
```
//using npm
npm i next-pwa

//using yarn
yarn add next-pwa
```

The next step is to create a manifest.json file, which tells the browser how the Progressive Web App should behave when installed on the user's desktop or

We'll be using the simicart PWA manifest generator and you'll need to enter the details of your app to generate the manifest file. The interface should look this:



When you click on the generate manifest button, It downloads the manifest file and you will add the contents of the file to the public folder of your app. You'll also need to rename the manifest.webmanifest file to manifest.json



Your manifest.json file should look similar to this:

```
{
    "theme_color": "#1fd4c0",
    "background_color": "#259409",
    "display": "standalone",
    "scope": "/",
    "start_url": "/",
    "name": "Advice app",
    "short_name": "Advice App",
    "description": "A random advice generator",
    "icons": [
        {
            "src": "/icon-192x192.png",
            "sizes": "192x192",
            "type": "image/png"
        },
        {
```

```
            "type": "image/png"
        },
        {
            "src": "/icon-384x384.png",
            "sizes": "384x384",
            "type": "image/png"
        },
        {
            "src": "/icon-512x512.png",
            "sizes": "512x512",
            "type": "image/png"
        }
    ]
}
```

Creating a _document.js file

In the pages folder, you should create a _document.js file to override the default document in Next.js because pages in Next.js skips the definition of the surrounding document's markup. For example, you never include <html>, <body>, etc.

_document.js forms the overall structure of the HTML. If you need need to modify the html or body tags,you have to do it in the _document.js file.

```
import { Html, Head, Main, NextScript } from "next/document";
export default function Document() {
  return (
    <Html>
      <Head>
        <link rel="manifest" href="/manifest.json" />
        <link rel="apple-touch-icon" href="/icon.png"></link>
        <meta name="theme-color" content="#fff" />
      </Head>
      <body>
        <Main />
        <NextScript />
      </body>
    </Html>
```

Configure the next.config.js file

Lastly, we will need to configure the PWA dependency by adding this piece of code to our `next.config.js` file. You can check out the PWA plugin for more configuration options.

```
module.exports = {
        reactStrictMode: true,
};
const withPWA = require("next-pwa");
module.exports = withPWA({
        pwa: {
                dest: "public",
                register: true,
        disable: process.env.NODE_ENV ===      'development',
                skipWaiting: true,
        },
});
```

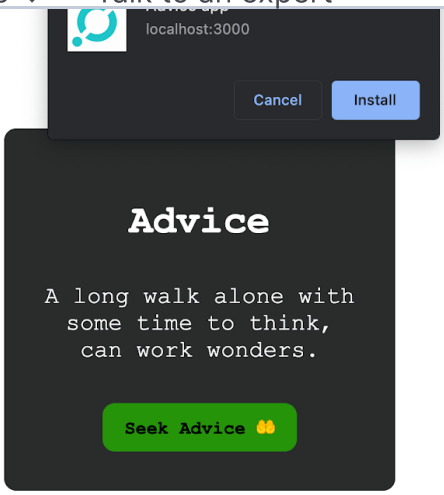
As you can see in the code above, we disable PWA in development environments since the Cache API/Cache Storage provides an extra layer of caching that can complicate debugging.

After configuring the next.config.js file, we'll need to run the command below in the terminal. This command builds the application for production usage.

```
npm run build
```

Next we run `npm run start` to start our Next.js app in production server.

The PWA is now ready and you should see an icon appear in the search bar. By clicking on that icon you will be prompted to install the app



After creating the PWA, you'll notice that some files like service workers and workbox have been added to the public folder. These files are not really important and do not need to be pushed to production. All you need to do is to delete those files and add them to .gitignore

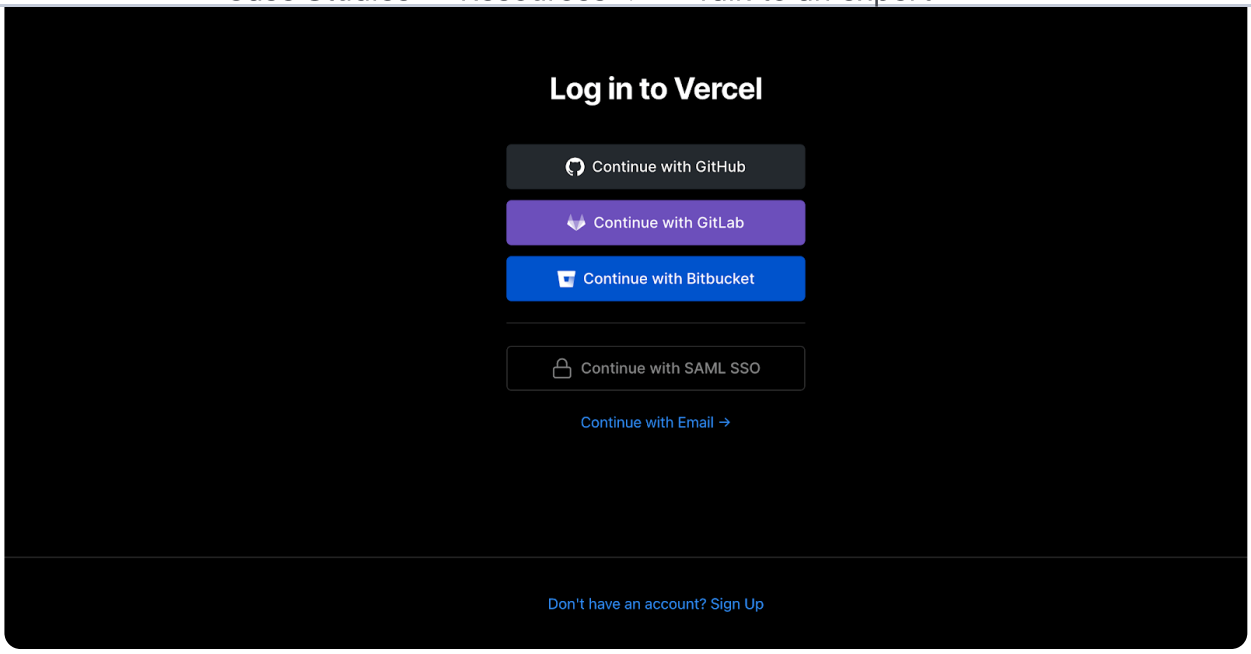
```
# PWA files
**/public/sw.js
**/public/workbox-*.js
**/public/worker-*.js
**/public/sw.js.map
**/public/workbox-*.js.map
**/public/worker-*.js.map
```

Deploying our PWA

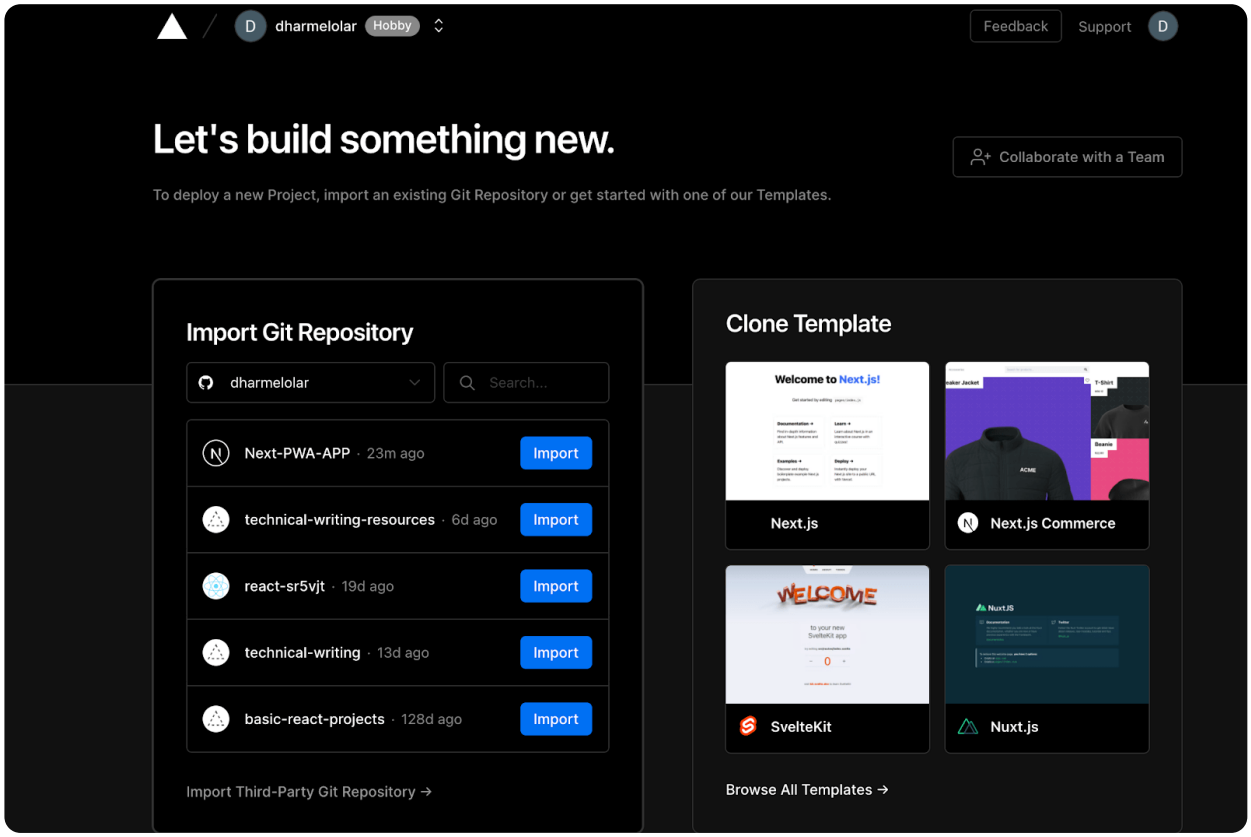
Vercel and Netlify are the most popular hosting and serverless services for Next.js Apps. We will be deploying our app to Vercel, a platform developed by the creators of Next.js.

To deploy your app to Vercel, you'll need to follow these steps:

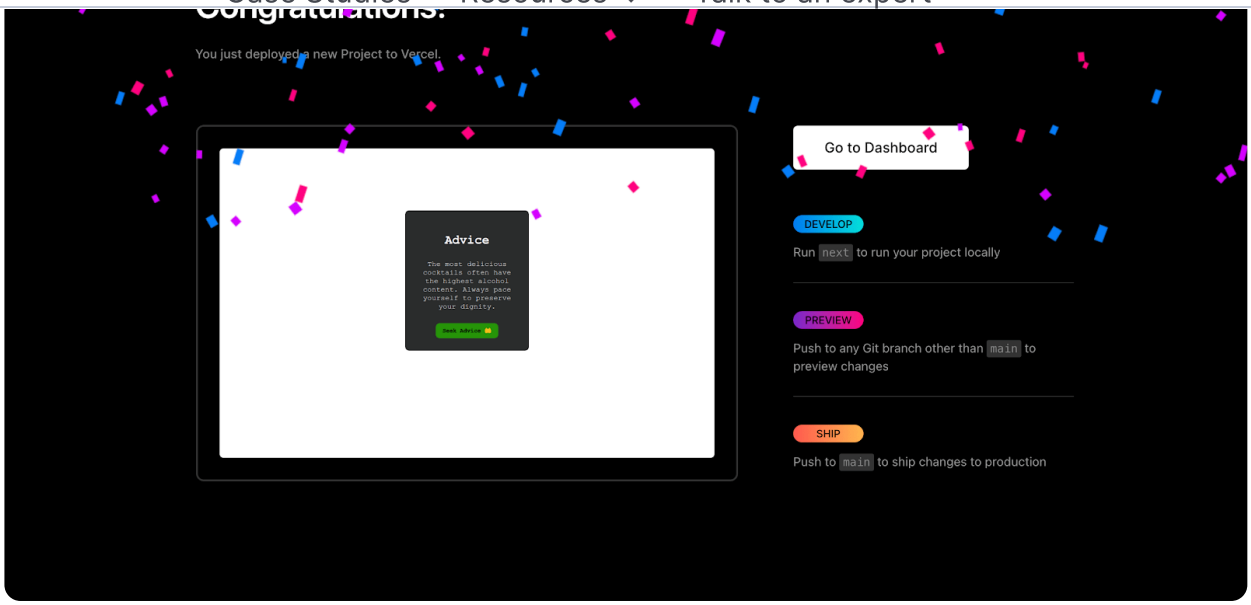
- Push your code to Github or any code hosting platform
- Sign up/ sign in to your account on the Vercel website using the options listed there.



- After signing in, you'll be redirected to your dashboard and you can click on the new project button to import a `new project` from Github.



- Next, you'll follow the steps as instructed to configure your project.
- Finally, our app has been deployed and you'll be provided with a link to access the app.



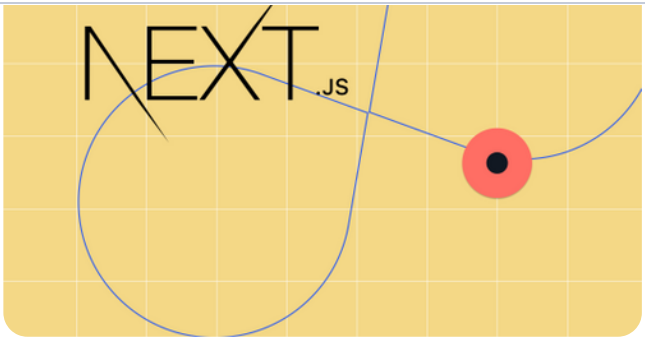
Check out these links for the live project and Github repo.

Wrapping up

Progressive web apps are gaining popularity because of the difficulties faced by users with their native mobile apps. This article is just a basic introduction to PWAs as there are lots of other PWA features that can be implemented in a project to give users a better experience. For further readings, MDN has a lot of resources on Progressive Web Apps.

Liked the article? Spread the word

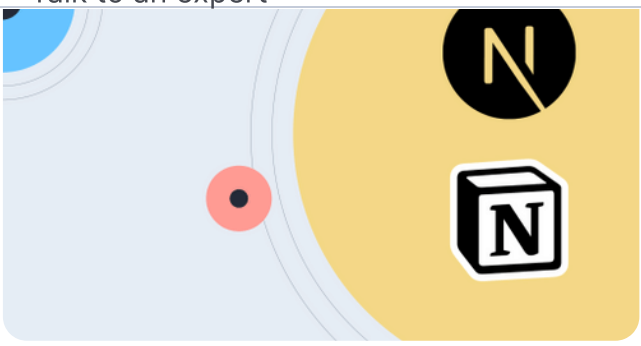
Continue reading



Guides

A guide to continuous integration in Next.js

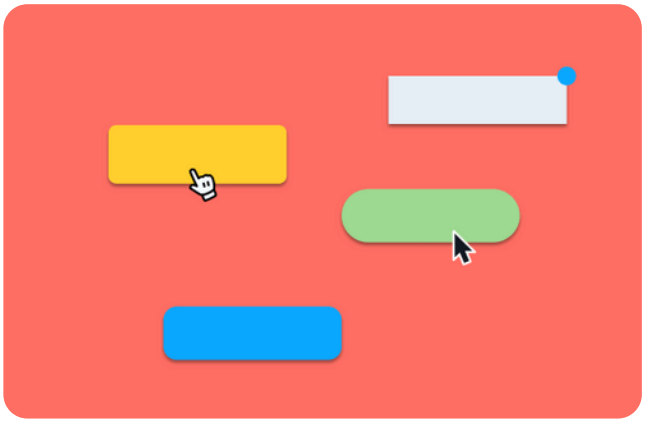
In this blog, we'll cover how to set up a testing framework for continuous feature integration into your Next.js applications without breaking them.



Guides

Build a Chrome Extension in Next.js and Notion API

Chrome extensions are a great way to customize your browsing experience. In this tutorial, you'll learn to build a Chrome extension using Next.js.



Guides

Creating hover effects with Tailwind CSS

In this Tailwind CSS tutorial we'll learn about hover classes and how to apply over effects to elements.

Browse all articles →

Put your knowledge to practice

Try Bird on your next bug - you'll love it

Sign up for Bird