Michael Adam Berry

Software Developer

Contact

Address
648 Manhattan Ave
Brookyn, NY 11222

Email
me@adamberry.dev

Phone
270.314.7807

website
adamberry.dev

Skills & Tools

JavaScript, HTML, CSS, SCSS, React (Class Components or Functional Components with Hooks), Animation, Svelte, SVG Filters and Animation, D3, Node, Express, Gatsby, GraphQL, Next, Sapper, Firebase, Observable Notebook, GreenSock, Three.JS, Adobe Illustrator, Git, Bash, Vim, VSCode, Web APIs, Python, SQL, NoSQL,

Other Interests

Being a therapy human to my rescue dogs, living that podcast life, having an inconsistant yoga practice, speculative fiction, live music (have seen more Grateful Dead cover bands than I care to admit), Avid distance runner (2:46 Marathon PR, 3rd place Brooklyn Marathon in 2015).

Open Source Development *– New York | 20017 - Present*

Eleftable - 2019 / (ongoing through Democratic primary) TLDR : Interactive Polling App

•Simplified the management of a complex UI design with Svelte– chosen for obvious benefits of a component based modular framework and the added enhancement of a compoilation process on build to reliably ship small, performant bundles to production.

•Employed a Firebase backend by configuring a cloud hosted, realtime Firestore NoSQL database and offloading user auth concerns by way of built-in Firebase authentication tools.

•Carefully weighed speed-of-development needs versus concerns of Firebase's proprietary ecosystem and data persistence before deciding that due to the time sensitive/short life cycle nature of this project that Firebase's services were the ideal solution.

•Limited database writes and reads by shifting workload to client and designing a front end architecture to handle UI complexities and state management.

•Utilized Svelte's built-in transition and easing tools, as well as classic CSS keyframe animations to create an engaging game-like UI.

Trumpcare - 2020 - TLDR : News App - Interactive Timeline

•Created SPA to render animated transitions based on scrolling events. Finite, static data allowed text and links to exist in client application state rather than on a server but images were hosted on cloudinary CDN for increased speed and reliability.

•The application combines effective "scrolly-telling" techniques by using JavaScript and web APIs to detect DOM element positions on during a scroll event, and triggering animated transitions of elements upon entry into veiwport.

•Employed tools like Adobe Illustrator to apply image filters and constructed wireframe of beginning and end state of animations to increase development speed.

Timepeace - 2019 - TLDR : E-Commerce JAM Stack Template for watch vendor

•React, Gatsby, and Shopify for JAM Stack solutions to classic e-commerce headaches while creating faster sites in a more stream-lined dev process.

•Through Gatsby's static site generation, SEO is enhanced and sites retain interactivity with React rehydration to give an app-like experience.

•Backend-less as core data (inventory, Sku variation, fulfillment ect.) is handled through vendor Shopify store account, but easlity retrievable with Gatsby's rich plugin ecosystem and graphQL API.

Grateful Data - 2019 - TLDR : Data Viz Project - Visualizing Setlist Data

•Created a command line node app to build a customized dataset and mine data from setlist.fm. The app parsed, filtered and formatted response data before writing JSON files which were uploaded to a CDN. These files are imported into the application at runtime to serve as primary data source for all charts.

•A variety of d3 scale and interpolation methods are used to build visualizations which include circle packs, area charts, and circular bar charts to present findings

•Stylistic elements (including SVG animations that are triggered on scroll events, circular gradients to make SVG <circle> elements appear as three dimensional orbs) are applied to direct attention and heighten a user experience.

Find My Reps - 2018 - TLDR : Search Application for Elected Officials by Address

•Leverages React and Next.js to compose form, input, and auto search components that store and validate user input state.

•Used Next's built-in router API to allow user data to persist as a query string and handled asynchronous requests with React's lifecycle methods allowing for the UI components to be populated with response data.