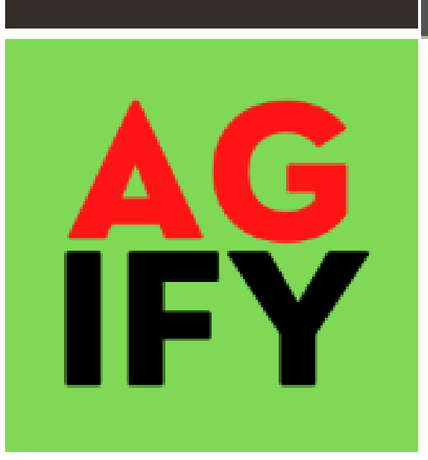
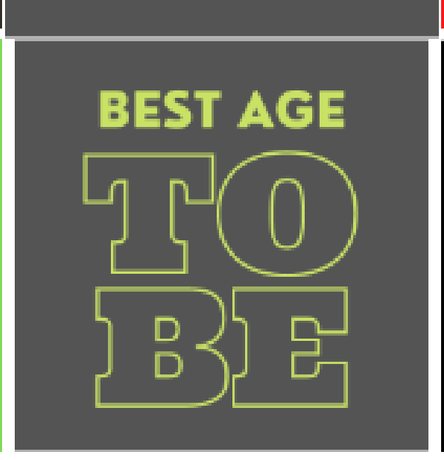
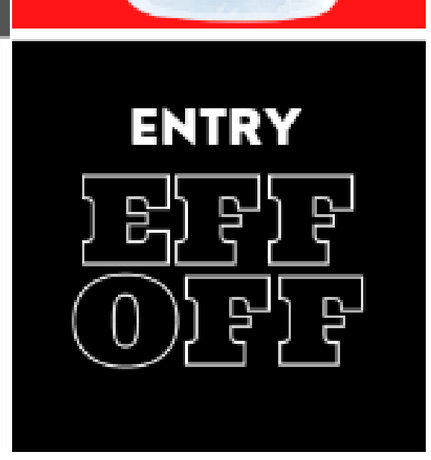
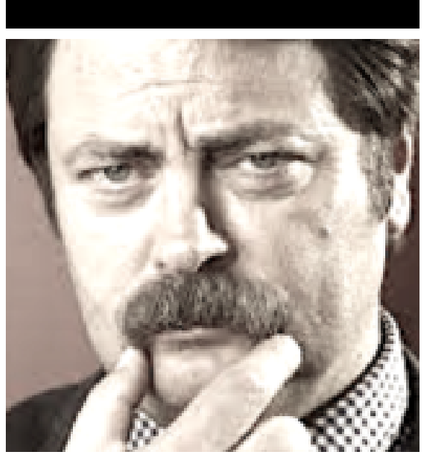
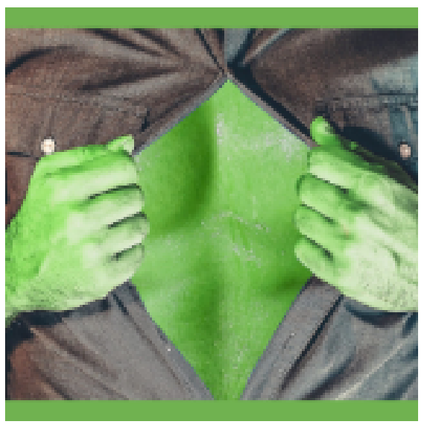
CREATIVITY USEEFFECT MENTORING DEBUGGING
LOCAL STORAGE RECURSION SYMMETRY BLUR EFX
EMAIL API SASS USESTATE REACT SELECT TRELLO
GITHUB CACHING MATERIAL UI CELEBRATIONS
REGEX CSS SUMMARY DOCUMENTATION AXIOS





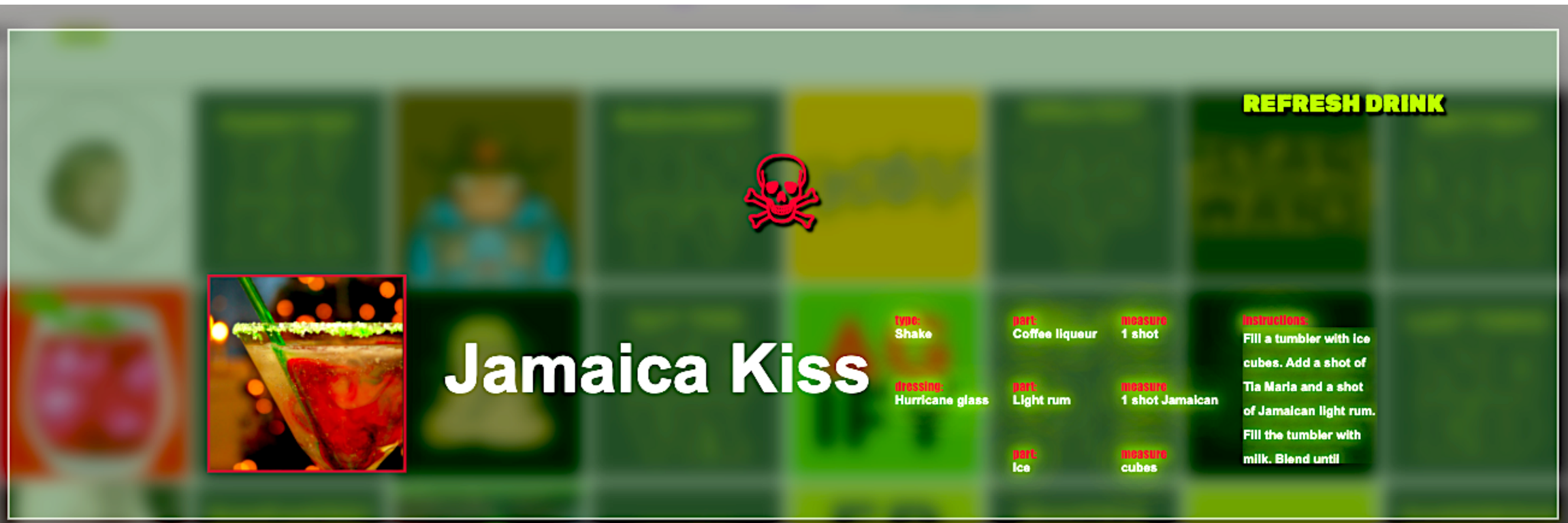






GROWTH GRAPHIC DESIGN BACKLOG MARKDOWN
LOADING INDICATORS FONTAWESOME PATIENCE
MODE OPTS PAIR PROGRAMMING MODALS TOGGLE
VISIT COUNT ERROR HANDLING APIS DOWNLOADS
HTML CURIOSITY FETCH LONG DAYS DFD TERNARIES

EPICAPIS Events



SUMMARY

Designed to be a unique, vibrant user experience, **Epic Apis** is a collection of quirky apis - ten - paired with semantic HTML in a single-page React App. In addition to each HTML element that complements a corresponding api, there are an additional four - 12 accept user input. The remaining two offer downloadable resources. **Epic Apis** is styled with css, sass, and material ui, uses Font Awesome, and features a streamlined user interface on the main page and in each modal. Take your time, make selections, click submit, then email part of your experience through the baked in SendGrid Email API. **Epic Apis** is to be indulged.

CODING EPIC APIS

Epic Apis reflects my story of continual learning. Stepping aside from advanced concepts, and re-entertaining foundational ones. Using them as they are. Expanding on them. Auditioning how I manage them in and of themselves - versus in cahoots with other possibilities. Then, naturally, reaching for the upper shelves again.

Essentially, though, center of my efforts remains tireless pursuit to do right by my code. Asking, Is it industry standard? How can it be cleaner? What makes the most sense for this scenario? I learned about myself I really do care how my code performs. And if it is regarded as worthy in the company of peers. In its most basic form, Epic Apis is a centerpiece re: my ability to take a simple concept - create an api, style it, repeat - and scale it through creative vision to a unique user experience. Epic Apis showcases my graphic design skills, passion for communication, and propensity for pushing boundaries into what else I am or will be competent to code.

Epic Apis started as a project I stuck on a shelf for a long while. Before dusting it off. Taking for granted the time, energy, emotion I would eventually invest in it. Epic Apis taught me patience - including with myself, the value of sufficient rest, and humility in times of toggling from *I can do this* (on my own) to needing/seeking senior-level guidance. It taught me, retaught me, no project is small. No matter how much I tell myself I'll only do the basics, I'll only crank out the MVP, I will put my best into what I do. Continue for a breakdown of the epic apis summary.

EA Events

SUMMARY
OBJECTIVES

my objectives in creating Epic Apis, were simple:

- wire apis for output only;
- wire apis for output per user input;
- upskill agility re: event listeners/handlers;
- revisit semantic html;

Specifically creating forms, sections, single- v multi- selection input, using little to no tags.

- implement pure css;
- create and maintain clean, clear hierarchal file system/structure;
- and, bake in clear sight to my design prowess.

LINKS TO LEARN MORE

Readme: Epic Apis Production 101
Github / Frontend: Where & How The Code Magic Happens

Readme: SendGrid Email API 101
Github / Backend: Getting The Code - Experience - In Your Inbox

Trello: Project Management - Vision to Implementation

SUMMARY
OUTCOMES

objectives are easy to write. outcomes reveal themselves. and as such require more of us. my objectives were met. and added are the following:

- writing functions in react js;
- adding loading indicators;
- re-calling apis for updated output;
- working with ternary operators;
- constructing autocomplete / react select from api calls;
- implementing alternate page mode (aka dark mode);
- implementing css advanced styles (star wars crawl; starry sky; text effects);
- learning sass (variables, mixins, consistent implementation);
- using external styles libraries (mui, font awesome, react select);
- leveraging localStorage for custom user experience;
- caching;
- refactoring;
- pair programming;
- and, creating technology-focused videography (project feature).

SUMMARY
TECH STACK

tech stacks stay the same for a project. react, html, and css in this project. not so. tech stacks can and may change, expand:

- react / js
- html
- css
- sass
- font awesome
- mui
- react select

SUMMARY
ENHANCEMENTS

this list could likely go on. one never knows what a code future will bring:

- optimize media queries for standard screen sizes;
- add user specified content for inclusion in email;
- include database of previous users by email, selections, and possibly name used on site;
- option for user to pick autocomplete versus user input search for apis with search;
- option for user to search by creator or character or comic in marvel api (not just by creator);
- option for user to search by starship or character or film in star wars api (not just by starship);
- option for user to customize foaas api output;
- option for user to embed custom api call in foaas api;
- option for user to search cocktail api (not just get random output);
- input option to all html/semantic fillers so users not limited to preset options;
- implement higher grade styles throughout the project;
- option or mandatory inclusion of preferred mode(s) and visit count in user-specific email;
- integration of blurb/paragraph into insight of meaning of name and/or how names are aged by agify api;
- option for user to re-arrange api/semantic input elements on the site;
- display unique visitor count on main page;
- incorporate user input in apis to content user emails (ie. pokemon searched; starship searched; favorite wow though this would require implementing a wow search feature to the own wilson api).