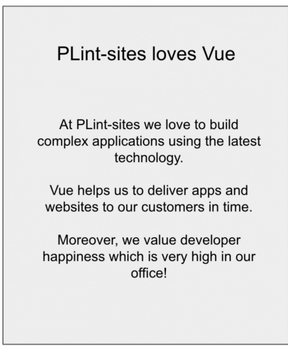
Creating a product configurator with Vue

Part 1: concept, sketch and component definition

Pim Hooghiemstra Follow
May 1 · 3 min read

In this blog series I'll discuss the process of developing a functionality we see a lot these days: a configurator for a custom product. Things that come to mind are mugs with custom text or personalized t-shirts. I'll build a product configurator for a fictive webshop that sells postcards. An example of a custom postcard is shown below:



In the configurator of our fictive webshop, a customer should be able to choose and configure the various options and features for the postcard. For example, the shape of the card, the paper quality and the amount.

To implement such a product configurator we'll use Vue. In this series we won't bother ourselves with validation or processing of the configured products, we'll focus on the frontend only.

In this first part, I like to focus on the initial phase where I create a sketch of the product configurator and design the component structure for Vue. You could of course fire up your IDE of choice and start programming directly. Although I feel the same urge, the end result will be better if we think about the structure of the project first.

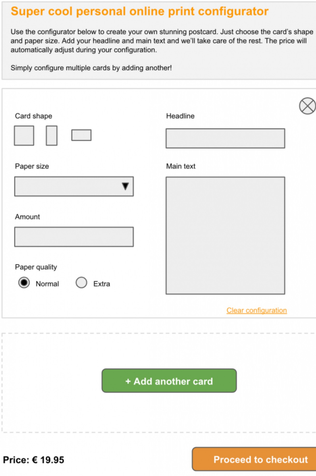
Concept

So let's write down the requirements of the configurator. It should be able to:

- configure multiple postcards at once
  - add a product
  - remove a product
- per postcard we want to:
  - choose the shape (square, rectangular portrait and landscape)
  - choose the papersize
  - choose the amount
  - choose the paper quality (normal, extra)
  - fill in the heading text
  - fill in the body text
  - be able to reset configuration
- during configuration, the price should update automatically on every change

Sketch

Based on the specifications defined above, I created the following sketch for the postcard configurator.



Components

Now it is time to think in Vue's language, that is, components. Although Vue is not very opinionated how to set up your components, I like the approach of a single 'smart' component that passes all required information to its children as props. The child components pass events up when something happens.

So let's call the smart component PostCardConfigurator. It consists of a ProductList, a PriceContainer and a ProceedToCheckoutButton.

The ProductList will list a set of SingleProduct components and holds an AddProduct component. With these in place we can sketch the structure of the app

```
<PostCardConfigurator>
    <ProductList>
        <SingleProduct />
        <SingleProduct />
        ...
        <AddProduct />
    </ProductList>

    <PriceContainer />
    <ProceedToCheckoutButton />
</PostCardConfigurator>
```

printwork-configurator-components.vue hosted with ♥ by GitHub — view raw

A note on the SingleProduct component

As you can see in the sketch, the SingleProduct component holds a lot of inputs. When developing this component I'll definitely compose it using smaller components, such as a CardShape component, a PaperQuality component and so on. The complexity here lies in managing the state of the configurator. We'll use Vuex for this and discuss this in detail in the next post.

Conclusion and next steps

In this post we discussed the first phase of developing a custom product configurator for a fictive webshop. We started with the specifications, made a sketch and defined Vue components.

In the upcoming posts I'll discuss the Vuex setup (state, modules, actions and mutations), the individual components and finally how to connect the dots to make it work. Stay tuned!

. . .

Originally published at https://www.blog.plint-sites.nl on May 1, 2019.

JavaScript Vuejs Vuex

7 claps

Pim Hooghiemstra Follow
Love to build Laravel + Vue applications! Founder of PLint-sites. https://plint-sites.nl

PLint-sites Follow
Dutch based webdevelopment company, focussing on modern webapplications using Laravel and Vue. Mainly writes about stuff to achieve cool things!