



# VueJS + Vuex vs. Vanilla JS

Yut and Micky

Growth Session #24 - May 16-17 2019

# Agenda

- What is a Braive questionnaire?
- How is the questionnaire implemented in vanilla JavaScript?
- What are the pain points when we implement it with vanilla JavaScript?
- How we solve the pain points with VueJS + Vuex?
- Conclusion

# What is a Braive questionnaire?

[About](#) ▾[Programs](#)[Articles](#) ▾[FAQs](#)[Contact](#)[Emergency Line](#)

## Yes, More K10!

[Continue](#)

Question 2 of 10

About how often did you feel nervous?

[BACK](#)[Next Question →](#)

In the PAST WEEK, how often have you been bothered by the above statement?

- ☒ a None of the time
- ☐ b A little of the time
- ☐ c Some of the time
- ☐ d Most of the time
- ☐ e All of the time

## Mental Health Check

4%

[BACK](#)

I found it hard to wind down

In the PAST WEEK, how often have you been bothered by the above statement?

- ☐ a Never
- ☐ b Sometimes
- ☐ c Often
- ☐ d Almost Always

Question 5 of 10


About how often did you feel restless or fidgety?

[BACK](#)

In the PAST WEEK, how often have you been bothered by the above statement?

- ☐ a None of the time
- ☐ b A little of the time
- ☐ c Some of the time
- ☐ d Most of the time
- ☐ e All of the time

# How is the questionnaire implemented in vanilla JavaScript?

 **braive**  
building healthy minds

About ▾ProgramsArticles ▾FAQsContactEmergency Line

Yes, More K10!

Continue

Question 2 of 10

BACKNext Question →

About how often did you feel nervous?

In the PAST WEEK, how often have you been bothered by the above statement?

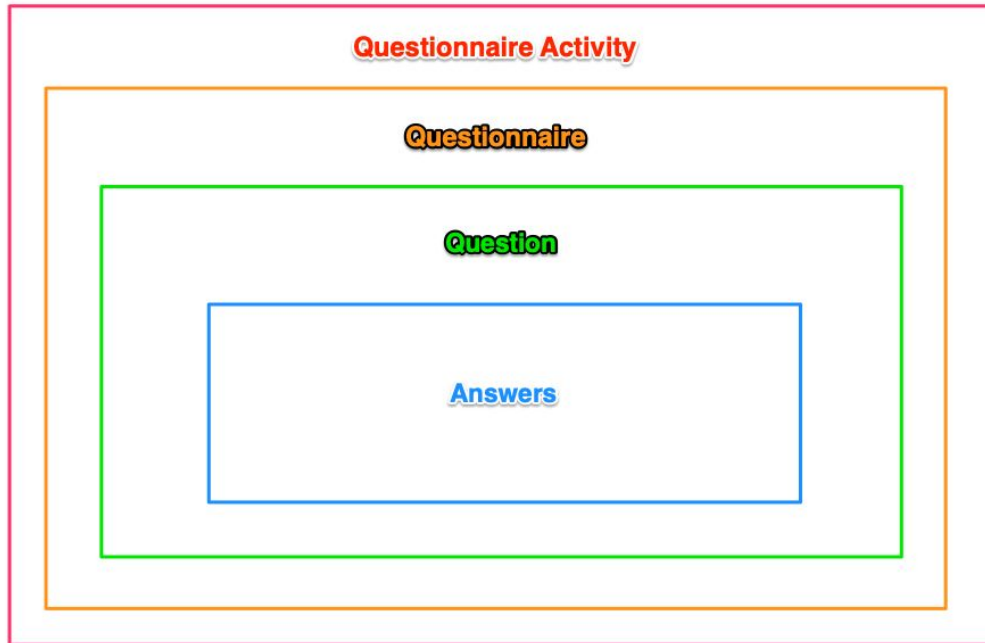
aNone of the time

bA little of the time

cSome of the time

dMost of the time

eAll of the time



# What is the pain when we implement with vanilla JavaScript?

- Hard to maintain and extend functionality
- When component get bigger it is harder to manage the state

Files changed (47)

```
+19 -0 A app/assets/javascripts/adapters/responseSet.js
+46 -16 M app/assets/javascripts/components/QuestionnaireActivity/index.js
+9 -8 M app/assets/javascripts/helpers/questionnaire/index.js
+54 -0 A app/assets/javascripts/helpers/questionnaire/kSurvey/question.js
+146 -0 A app/assets/javascripts/helpers/questionnaire/kSurvey/questionnaire.js
+16 -0 A app/assets/javascripts/helpers/questionnaire/kSurvey/survey.js
+1 -1 M app/assets/javascripts/helpers/questionnaire/mhc/mhc_questionnaire.js
+9 -5 M app/assets/javascripts/helpers/questionnaire/question.js
+2 -0 M app/assets/javascripts/helpers/questionnaire/response_set.js
+16 -0 M app/controllers/activities/activities_controller.rb
```

```
class KSurveyQuestion extends Question {
  constructor(questionData,
    previousQuestionCallback, nextQuestionCallback,
    submitResponseCallback, submitQuestionnaireCallback,
    questionnaireCompletedCallback = null,
    lastQuestionSubmitCallback = null) {
    super(questionData,
      previousQuestionCallback, nextQuestionCallback,
      submitResponseCallback, submitQuestionnaireCallback,
      questionnaireCompletedCallback);
    // Callbacks
    this.lastQuestionSubmitCallback = lastQuestionSubmitCallback;
  }

  // Private
  _initNextQuestionButton() {
    if (this._hasResponses()) {
      if (this.isLastQuestion()) {
        !this.isCompleted && this.lastQuestionSubmitCallback();
      } else {
        this._enableNextQuestionButton();
      }
    } else {
      this._disableNextQuestionButton();
    }
  }
}
```

```
class Question {
  constructor(questionData,
    previousQuestionCallback, nextQuestionCallback,
    submitResponseCallback, submitQuestionnaireCallback,
    questionnaireCompletedCallback = null) {
    Object.assign(this, questionData);
    this.correctAnswerIds = this._getCorrectAnswerIds();
    this.pickType = PICK_TYPE;
    this.selectors = SELECTORS;
    this.answered = [];

    // Elements
    this.$questionProgress = $(Config.questionInfo.progress);
    this.$questionName = $(Config.questionInfo.name);
    this.$questionHint = $(Config.questionInfo.hint);
    this.$form = $(Config.form);
    this.$formGroup = $(Config.formGroup);
    this.$formLegend = $(Config.formLegend);
    this.$previousQuestionButton = $(Config.previousQuestionButton);
    this.$nextQuestionButton = $(Config.nextQuestionButton);

    // Callbacks
    this.previousQuestionCallback = previousQuestionCallback;
    this.nextQuestionCallback = nextQuestionCallback;
    this.submitResponseCallback = submitResponseCallback;
    this.submitQuestionnaireCallback = submitQuestionnaireCallback;
    this.questionnaireCompletedCallback = questionnaireCompletedCallback;
  }

  // Methods
  _getCorrectAnswerIds() {
    return this.correctAnswerIds;
  }
}
```

```
class MHCQuestion extends Question {
  render() {
    super.render();
    this.$formLegend.text(this.sectionTitle);
    if (this._hasResponses()) {
      this._enableNextQuestionButton();
    }
  }

  getUserResponses() {
    let checkedInputs = document.body.querySelectorAll(this.selectors.CHECKED_INPUT);

    return checkedInputs.map(input => parseInt(input.value));
  }

  isLastQuestion() {
    return this.isLastQuestionOfSection();
  }
}
```

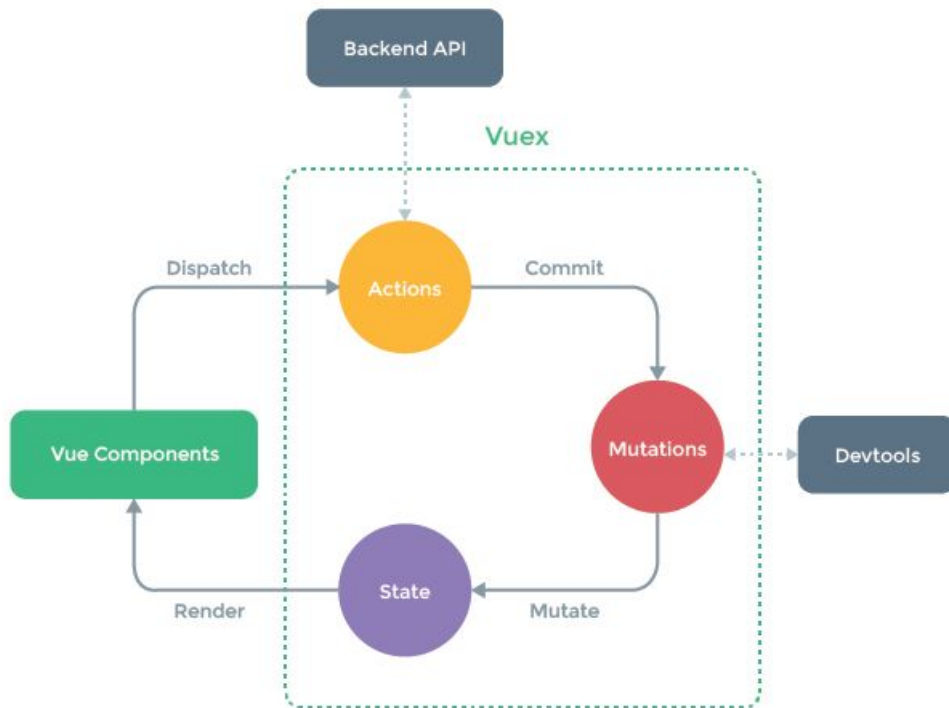


## How to solve the pain points with VueJS + Vuex?

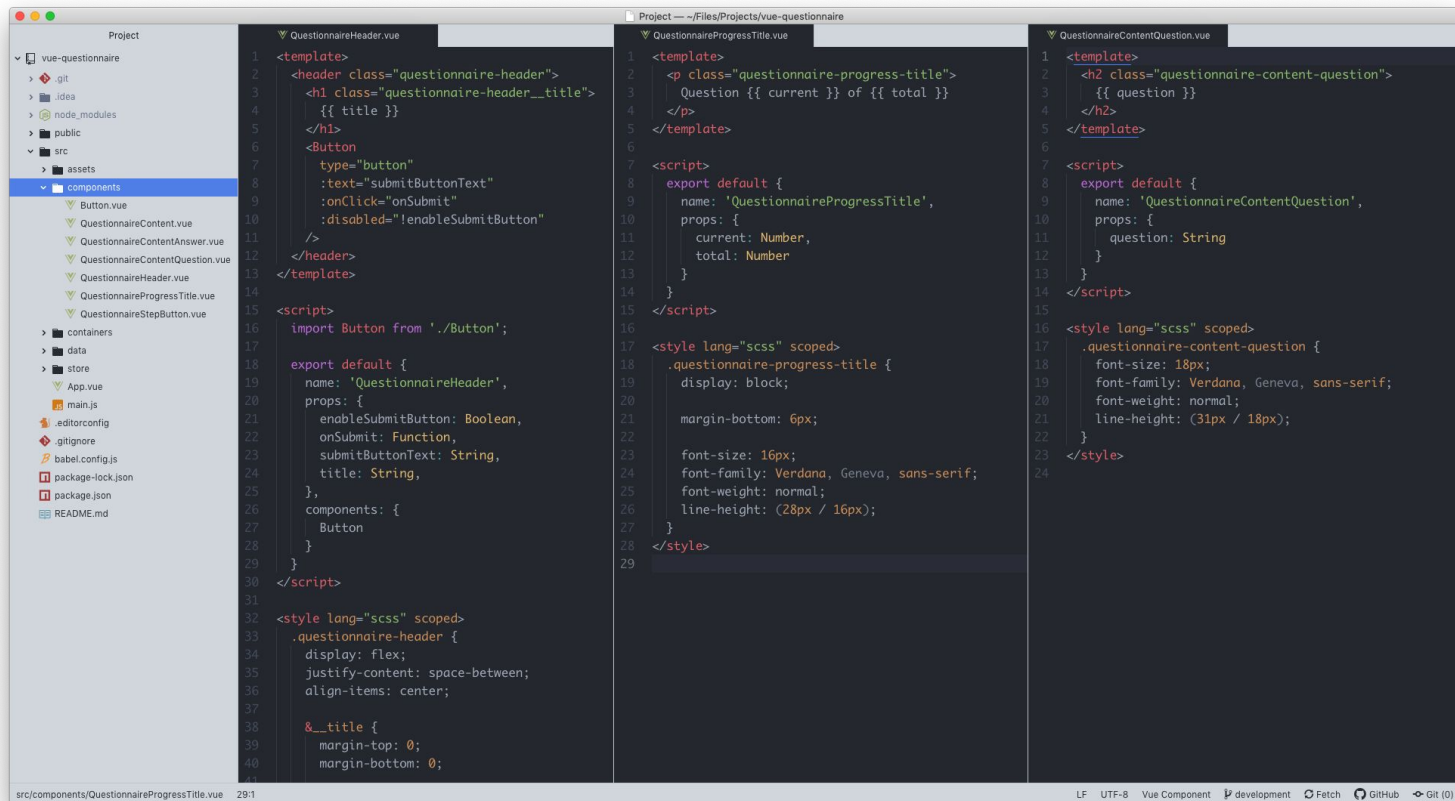
Refactoring the questionnaire into Vue components:

- Divide all questionnaire UI to small and stateless components.
- Create only one component to orchestrate all stateless component.
- Create a Vuex store to store the data.
- Connect the container component to the Vuex store.
- Perform all user interactions via the container component.
- Mutate the state of the data in the store and send the updates to the View via the container component.

# The Vuex's workflow



# Divide all questionnaire UI to small and stateless components





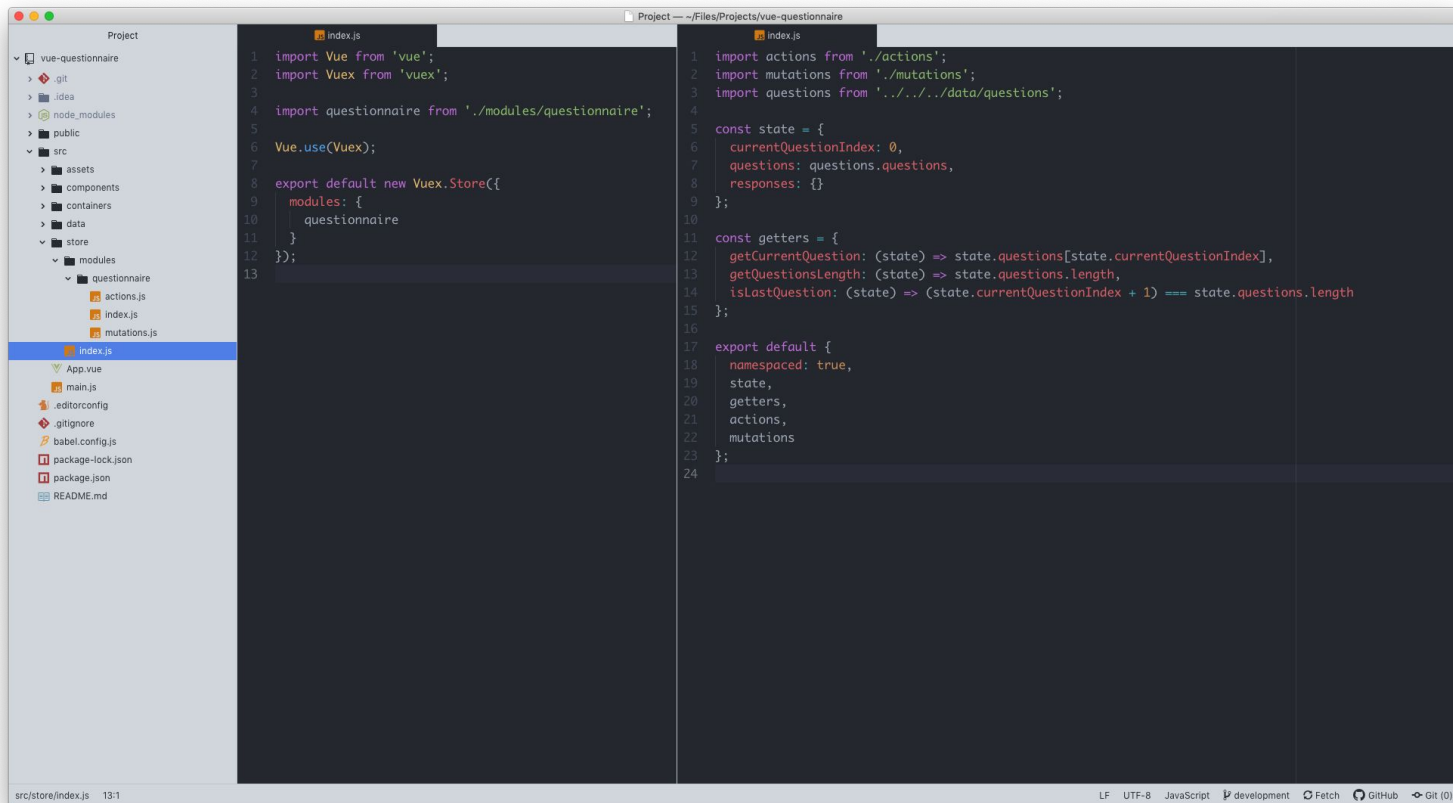
# Create only one component to orchestrate all stateless component

```
Project
└─ vue-questionnaire
  ├─ .git
  ├─ .idea
  ├─ node_modules
  ├─ public
  └─ src
    ├─ assets
    ├─ components
    └─ containers
      └─ Questionnaire.vue
        ├─ data
        └─ store
          └─ App.vue
            └─ main.js
              └─ .editorconfig
                └─ .gitignore
                  └─ babel.config.js
                    └─ package-lock.json
                      └─ package.json
                        └─ README.md

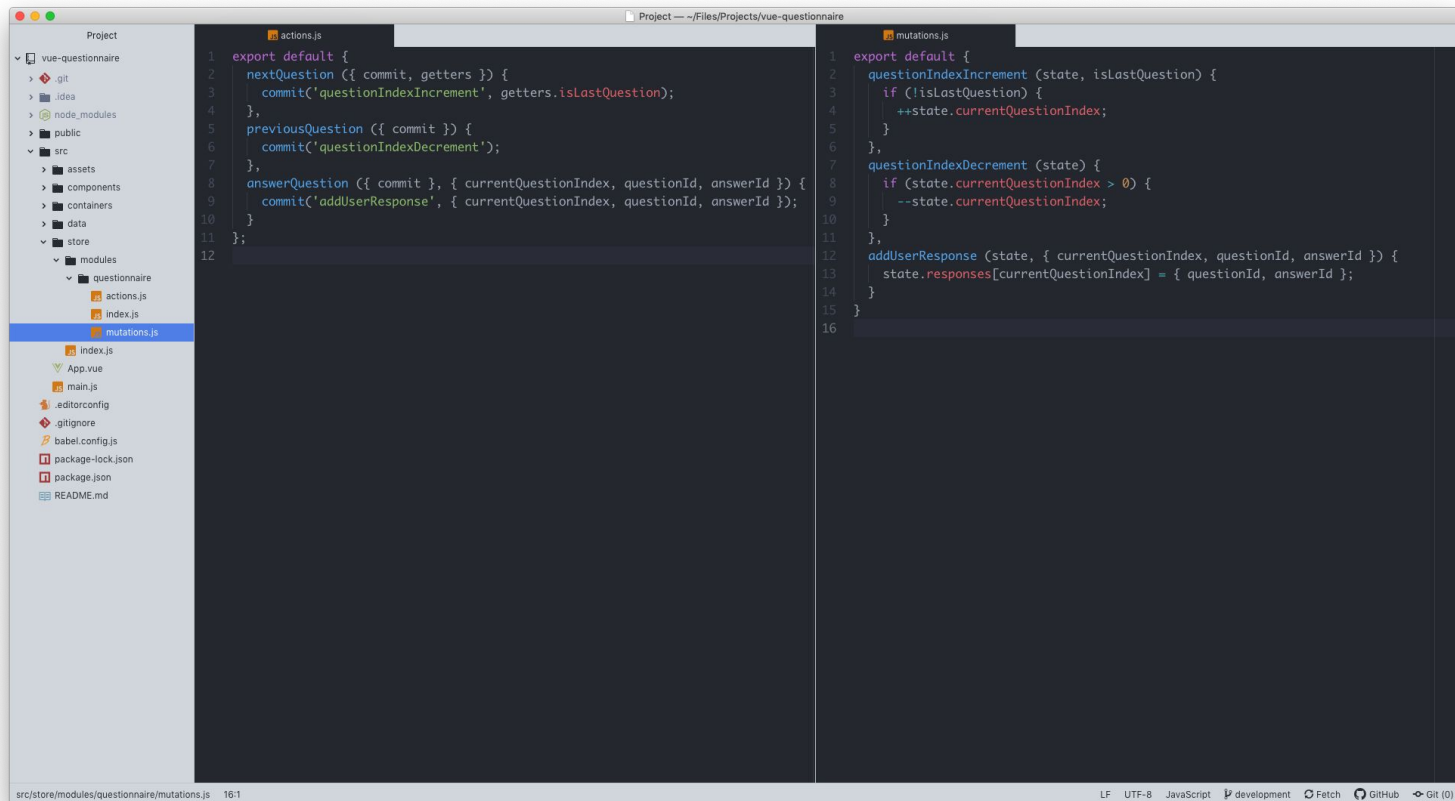
Questionnaire.vue
1 <template>
2 <section class="questionnaire">
3   <QuestionnaireHeader
4     title="K10"
5     submitButtonText="Continue"
6     :enableSubmitButton="true"
7     :onSubmit="submitHandler"
8   />
9
10  <QuestionnaireContent>
11    <template v-slot:progressTitle>
12      <QuestionnaireProgressTitle
13        :current="currentQuestionIndex + 1"
14        :total="totalQuestion"
15      />
16    </template>
17
18    <template v-slot:question>
19      <QuestionnaireContentQuestion :question="question.text" />
20    </template>
21
22    <template v-slot:questionStepButton>
23      <QuestionnaireStepButton
24        :isShowPrevious="isShowPrevious()"
25        :isShowNext="isShowNext()"
26        :onPreviousQuestion="previousQuestionHandler"
27        :onNextQuestion="nextQuestionHandler"
28      />
29    </template>
30
31    <template v-slot:answers>
32      <QuestionnaireContentAnswer
33        :answers="question.answers"
34        :onAnswer="answerHandler"
35        :currentAnswer="responses[currentQuestionIndex]"
36      />
37    </template>
38  </QuestionnaireContent>
39 </section>
40 </template>

Questionnaire.vue
52 export default {
53   name: 'Questionnaire',
54   computed: {
55     ...mapState('questionnaire', {
56       currentQuestionIndex: 'currentQuestionIndex',
57       responses: 'responses'
58     }),
59     ...mapGetters('questionnaire', {
60       question: 'getCurrentQuestion',
61       totalQuestion: 'getQuestionsLength',
62       isLastQuestion: 'isLastQuestion',
63     })
64   },
65   components: {
66     QuestionnaireContent,
67     QuestionnaireContentAnswer,
68     QuestionnaireContentQuestion,
69     QuestionnaireHeader,
70     QuestionnaireProgressTitle,
71     QuestionnaireStepButton
72   },
73   methods: {
74     nextQuestionHandler() {
75       this.$store.dispatch('questionnaire/nextQuestion');
76     },
77     previousQuestionHandler() {
78       this.$store.dispatch('questionnaire/previousQuestion');
79     },
80     submitHandler() {
81       window.console.log('DISPATCH: ', this.responses);
82     },
83     answerHandler(answerId) {
84       this.$store.dispatch('questionnaire/answerQuestion', {
85         currentQuestionIndex: this.currentQuestionIndex,
86         questionId: this.question.id,
87         answerId
88       });
89       this.$store.dispatch('questionnaire/nextQuestion');
90     },
91     isShowNext() {
92       return this.currentQuestionIndex < this.totalQuestion - 1;
93     }
94   }
95 }
```

# Create a Vuex store to store the data.



# Connect the container component to the Vuex store.



## Connect the container component to the Vuex store.



```
import Vue from 'vue';
import App from './App.vue';

import store from './store';

Vue.config.productionTip = false;

new Vue({
  el: '#app',
  store,
  render: h => h(App),
});
```

# Perform all user interactions via the container component.

```
<template>
  <section class="questionnaire">
    <QuestionnaireHeader
      title="K10"
      submitButtonText="Continue"
      :enableSubmitButton="true"
      :onSubmit="submitHandler"
    />

    <QuestionnaireContent>
      <template v-slot:progressTitle>
        <QuestionnaireProgressTitle
          :current="currentQuestionIndex + 1"
          :total="totalQuestion"
        />
      </template>

      <template v-slot:question>
        <QuestionnaireContentQuestion :question="question.text" />
      </template>

      <template v-slot:questionStepButton>
        <QuestionnaireStepButton
          :isShowPrevious="isShowPrevious()"
          :isShowNext="isShowNext()"
          :onPreviousQuestion="previousQuestionHandler"
          :onNextQuestion="nextQuestionHandler"
        />
      </template>

      <template v-slot:answers>
        <QuestionnaireContentAnswer
          :answers="question.answers"
          :onAnswer="answerHandler"
          :currentAnswer="responses[currentQuestionIndex]"
        />
      </template>
    </QuestionnaireContent>
  </section>
</template>
```

# Mutate the state of the data in the store

```
import { mapState, mapGetters } from 'vuex';

// import components

export default {
  name: 'Questionnaire',
  computed: {
    ...mapState('questionnaire', {
      currentQuestionIndex: 'currentQuestionIndex',
      responses: 'responses'
    }),
    ...mapGetters('questionnaire', {
      question: 'getCurrentQuestion',
      totalQuestion: 'getQuestionsLength',
      isLastQuestion: 'isLastQuestion',
    })
  },
  methods: {
    nextQuestionHandler() {
      this.$store.dispatch('questionnaire/nextQuestion');
    },
    previousQuestionHandler() {
      this.$store.dispatch('questionnaire/previousQuestion');
    },
    submitHandler() {
      window.console.log('DISPATCH: ', this.responses);
    },
    answerHandler(answerId) {
      this.$store.dispatch('questionnaire/answerQuestion', {
        currentQuestionIndex: this.currentQuestionIndex,
        questionId: this.question.id,
        answerId
      });

      this.$store.dispatch('questionnaire/nextQuestion');
    },
    isShowNext() {
      return this.responses[this.currentQuestionIndex] && !this.isLastQuestion;
    },
    isShowPrevious() {
      return this.currentQuestionIndex !== 0;
    }
  }
}
```

## Conclusion

- It separates the logic for manage the data from the view layer.
- It forces you to think in the terms of component-based architecture.
- Easy to test, extend and read.



# Demo Time




## Next Steps

- Unit testing.
- Integrate with Braive Backend:
  - Fetch the questions.
  - Submit user response.
- Implement different types of questionnaires:
  - Multiple answers.
  - Questionnaire with multiple set of question.

**JAVASCRIPT**

**JAVASCRIPT EVERYWHERE**

makeameme.org



Build with vanilla JS,  
Reimplement with Vue.js +  
Vuex when needed

# Thanks!

## Contact Nimble

[nimblehq.co](https://nimblehq.co)

[hello@nimblehq.co](mailto:hello@nimblehq.co)

## Bangkok

399 Interchange 21 Sukhumvit Road, Unit  
#2402-03, Klong Toei, Wattana, Bangkok  
10110, Thailand

## Singapore

28C Stanley St, Singapore 068737

## Hong Kong

20th Floor, Central Tower  
28 Queen's Road, Central, Hong Kong

