ADDING SPINNER WHEN SUBMITTING THE FORM

In the last video we've seen how to easily add AJAX spinner to a certain page by adding `spinner` element in the component's template. Now in this guide I'm going to show you how to refactor the `Spinner` component so that it can be used not only in a container element but also in small component such as button.

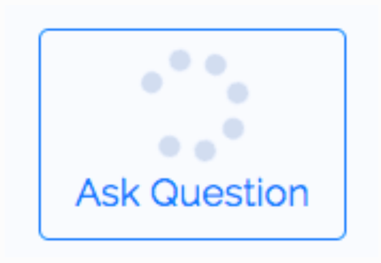
So when we hit the submit button whether in question or answer form we actually send a request to our server and waiting for the response. So I think it would be better to show the spinner as soon as we hit the button.

ADDING THE SPINNER ON THE SUBMIT BUTTON

So let's open the `QuestionForm.vue` then add the spinner before the button's text.

```
<button type="submit" class="btn btn-outline-primary btn-lg">
  <spinner></spinner>
  {{ buttonText }}
</button>
```

If you save the change and see in the browser. The spinner is appear but it make the button look ugly.



To fix this issue let's provide two sizes for the `spinner`: a normal size which we have so far and a small size which can be used in the button.

MODIFYING THE SPINNER COMPONENT

1. Adjust the spinner size

Let's open our `Spinner` component. And inside `i` tag let's get rid of the `fa-3x` class and add a class binding to `sizeClass` .

```
<div class="spinner">
  <i class="fa fa-spinner fa-spin" :class="sizeClass"></i>
</div>
```

Let's add script tag then define `props` property and put inside it another property called `small` . This property is going to be boolean and it should be `false` by default. By doing this way we keep the existing spinner's size in normal size. If we need to have a small spinner we need to specify the `small` attribute explicitly.

```
<script>
export default {
    props: {
        small: {
            type: Boolean,
            default: false
        }
    }
}
</script>
```

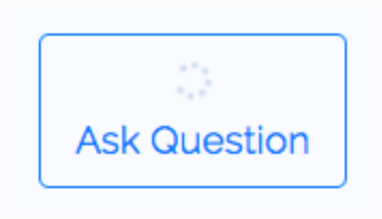
We can then add `computed` property and define the `sizeClass` method. In this method we simply check the `small` property. If it's true we'll return nothing otherwise we return the `fa-3x` class.

```
export default {
    // ...

    computed: {
        sizeClass () {
            return this.small ? '' : 'fa-3x'
        }
    }
}
```

Back to `QuestionForm` component and specify the `small` attribute in the `spinner` .

```
<spinner :small="true"></spinner>
```

Now if you save this change and see in your browser it will look better.



2. Control the spinner visibility

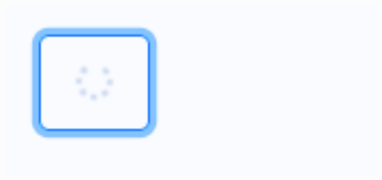
Now we can control the spinner visibility by making use the `$root.loading` on the `spinner` as we did before. We can then wrap the `buttonText` in a `span` and add `v-else` directive on it.

```
<button type="submit" class="btn btn-outline-primary btn-lg">
  <spinner :small="true" v-if="$root.loading"></spinner>
  <span v-else>{{ buttonText }}</span>
</button>
```

Before we test this let's open the `Api/QuestionsController` then add php `sleep` function before returning the json response.

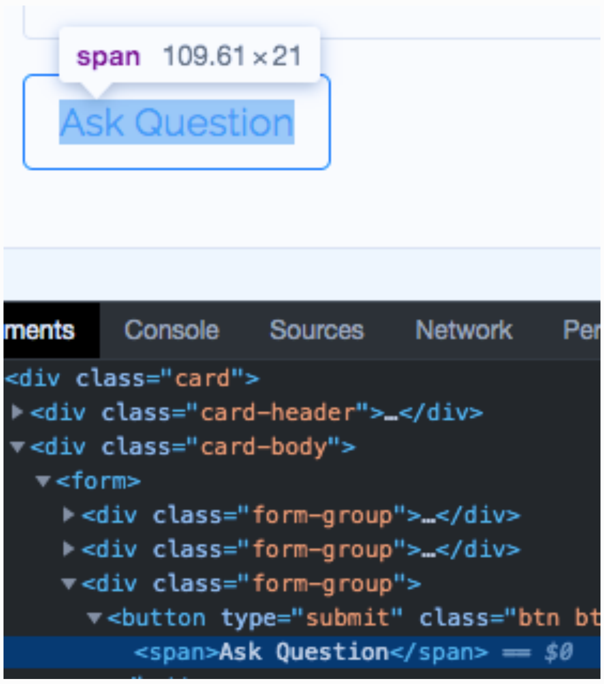
```
public function store(AskQuestionRequest $request)
{
  // ...

  if (env('APP_ENV') == 'local') sleep(2);

  return response()->json([...]);
}
```

Now if you enter title and body in the form then hit the **Ask Question** button you'll see the spinner. But the problem is the button's width is shrinking.



3. Fix the width shrinking

There could be many different ways to fix this issue. But the simple way is to set the css `min-width` to the spinner. Firstly we can inspect the span inside the button and see its width.



```
<div class="card">
 ▶<div class="card-header">…</div>
 ▼<div class="card-body">
   ▼<form>
     ▶<div class="form-group">…</div>
     ▶<div class="form-group">…</div>
     ▼<div class="form-group">
       ▼<button type="submit" class="btn bt
           <span>Ask Question</span> == $0
```

Here I can see that the width is 109.61 pixels. So back to our `Spinner` component then add `minWidth` property inside `props`. Its type is going to be `Number` and you could also set the default value to 109.61.

```
props: {
  // ...
  minWidth: {
    type: Number,
    default: 109.61
  }
},
```

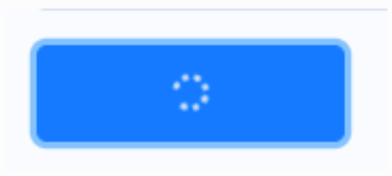
Then in `computed` property we can define another method let' say `getMinWidth`. This method will simply return an object contains css `min-width` and set it to `minWidth`.

```
computed: {
  // ...
  getMinWidth () {
    return {
      minWidth: this.minWidth + 'px'
    }
  }
}
```

Last, in the template you can perform style binding to the spinner container element.

```
<div class="spinner" :style="getMinWidth">...</div>
```

Now if you try to enter new question then hit the **Ask Question** button you'll see the spinner appear and the button's widht no longer shrinking.



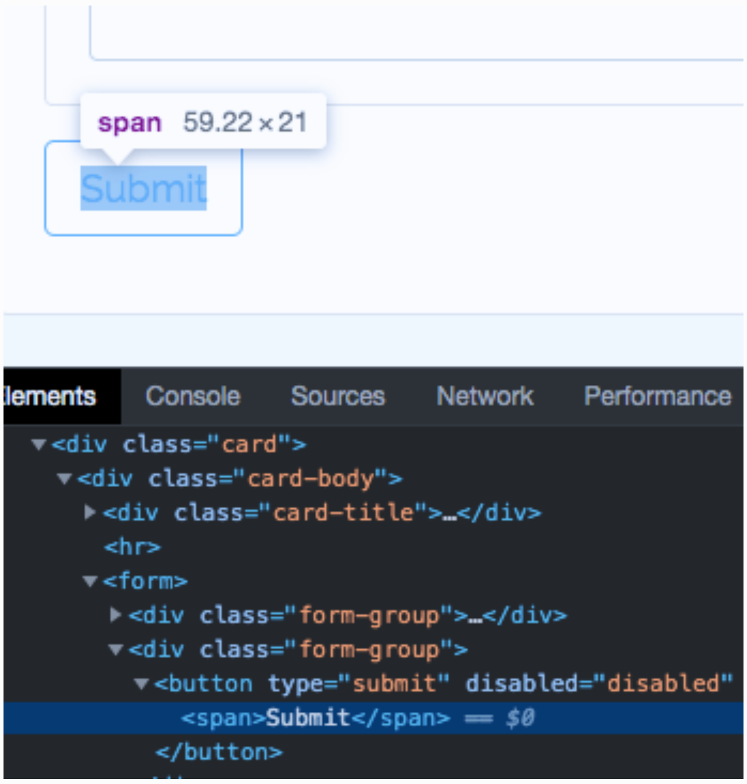
ADDING AJAX SPINNER ON NEWANSWER COMPONENT

If you want to add the AJAX spinner on the submit button in the `NewAnswer` component you follow the similar steps.

Wrap the button's text into a `span` then add `spinner` component before it.

```
<button ...>
  <spinner :small="true" v-if="$root.loading"></spinner>
  <span v-else>Submit</span>
</button>
```

Go to your browser and inspect the `span` on the submit button.



```
▼<div class="card">
  ▼<div class="card-body">
    ▶<div class="card-title">…</div>
     <hr>
    ▼<form>
      ▶<div class="form-group">…</div>
      ▼<div class="form-group">
        ▼<button type="submit" disabled="disabled"
            <span>Submit</span> == $0
          </button>
```

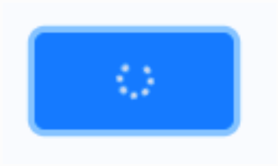
Here I can see that the width is 59.22. So in `spinner` component call we can bind the `min-width` to `59.22`.

```
<spinner :small="true" :min-width="59.22" v-if="$root.loading">
</spinner>
```

Last you can open `Api/AnswesController` then add php `sleep` function in the `store` method.

```
public function store(Question $question, Request $request)
{
  // ...

  if (env('APP_ENV') == 'local') sleep(2);

  return response()->json([...]);
}
```

If you try to enter new answer you'll see the spinner appear once you hit the submit button.



SUMMARY

So now that we have our Single Page Application fully functional with transition effect when switching between pages. We also have AJAX spinner which appear once we hit a certain endpoint. These will make our application look amazing and more professional.

So congratulation if you reach until this point so far. Don't forget to commit all changes to your git repository.

```
git add .
git commit -m "Add transition effect & AJAX spinner"
git push origin lesson-64
```