

Deleting product

In this lesson we're going to see how to remove a product in our server by implementing this endpoint in our Vue.js application.

Modify remove method in app.js like so:

```
remove (product) {  
  if (confirm("Are you sure?")) {  
    axios.delete('/products/' + product.id)  
      .then(res => {  
        let index = this.products.findIndex(item => item.id === product.id);  
  
        this.products.splice(index, 1);  
      })  
  }  
}
```

If you save the change and try to remove a product, you'll see that the product is completely removed. You can confirm that by refreshing the page.

Adding delete effect

Instead of removing the product item in the UI immediately once it deleted in the database, we can add an effect to the table row with a certain color for a moment of time before it completely removed from the UI.

Probably the common way is to use transition or transition group. But in this case I'll show you a different approach.

Because we're using bootstrap, we can apply table-danger class in the table row like so.

Est autem magni eum.

Et a ducimus ad ullam accusamus ea sint.

Elements Console Sources Network Performance Memory Application Security Lighthouse

```
> <tr>_</tr>
> <tr>_</tr>
▼ <tr class="table-danger"> == $0
  <td>Est autem magni eum.</td>
  <td>Inventore est sunt.</td>
  <td>1</td>
  > <td>_</td>
  </tr>
```

What we are going to do is to bind that class for the deleted product.

In data property, let's define a new property let's say `removedProductId` which is `null` initially.

```
// ...
errors: {},

removedProductId: null,
```

In the template bind the `table-danger` to the `tr` if `removedProductId` is equals to `product.id`.

```
<tr
  v-for="product of productsPaginated"
  :class="{ 'table-danger': removedProductId == product.id}"
>
```

Modify the remove method like so:





```
remove (product) {
  if (confirm("Are you sure?")) {
    axios.delete('/products/' + product.id)
      .then(res => {
        // store the product.id in removedProductId
        this.removedProductId = product.id

        // delay the execution for 1 second
        // then set the removedProductId back to null to detach
        // the table-danger class from <tr>
        // after that remove the tr from UI
        new Promise(resolve => setTimeout(resolve, 1000))
          .then(() => {
            this.removedProductId = null
            let index = this.products.findIndex(item => item.id ===
product.id);

            this.products.splice(index, 1)
```

```
    }
    });
  });
}
```

Save the change, and try to remove a product. Once confirmed you'll see the deleted product row is in red.

Product Name▲	Category	Price	Action
Distinctio nisi provident porro eos numquam magni pariatur.	Et laborum aperiam.	12	 
Dolore esse officia dolorem consequatur totam sit.	Et laborum aperiam.	8	 

After one second it will then completely removed from the UI.

Product Name▲	Category	Price	Action
Dolore esse officia dolorem consequatur totam sit.	Et laborum aperiam.	8	 
Est autem magni eum.	Inventore est sunt.	1	 