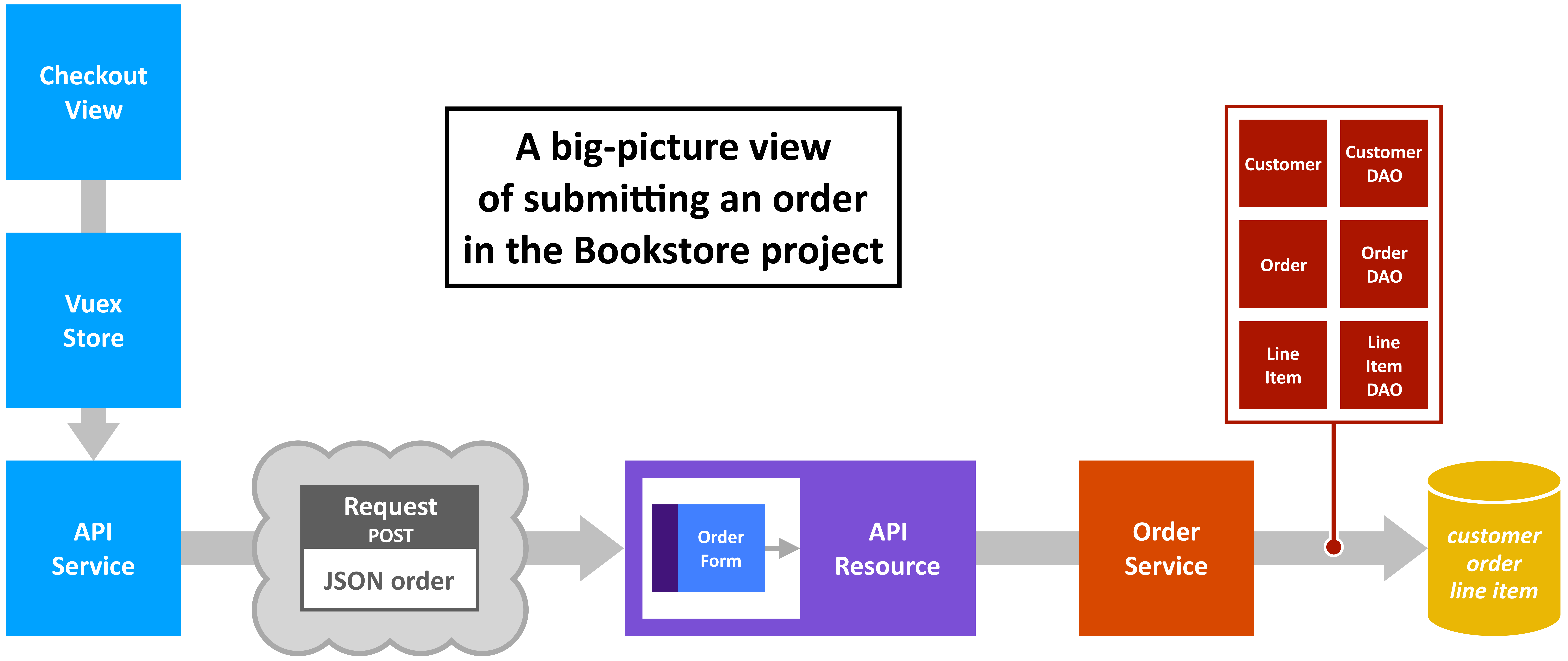
Submit-Order Promise Chain

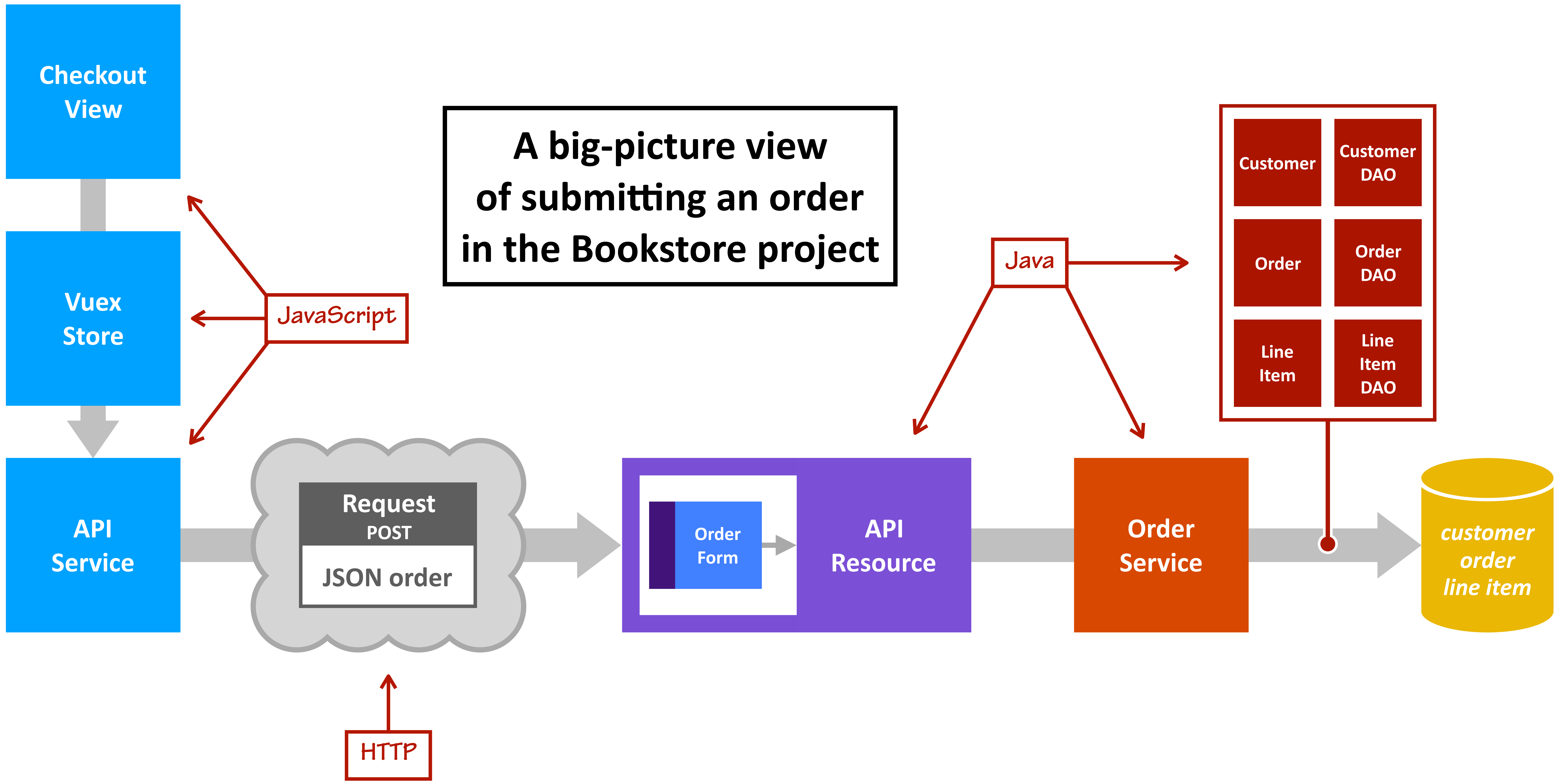
(in the CS5244 Bookstore project)

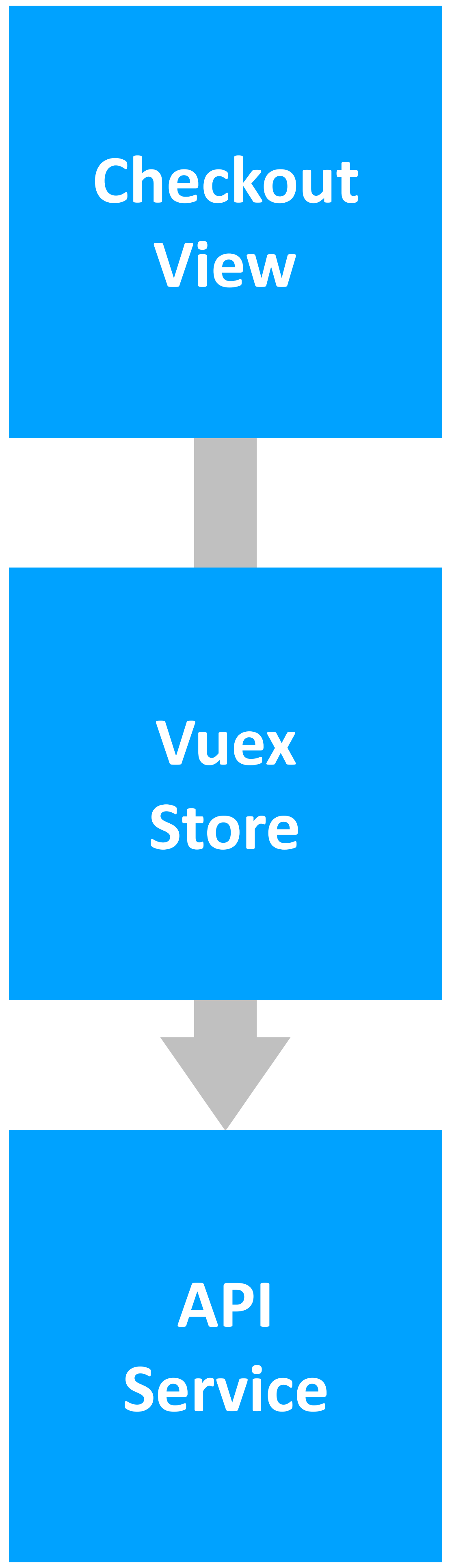
by Dr. K

A big-picture view of submitting an order in the Bookstore project



A big-picture view of submitting an order in the Bookstore project





```
submitOrder() {
  ...
  this.$store
    .dispatch("placeOrder", customerForm)
    .then(() => {
      this.checkoutStatus = "OK";
      this.$router.push({ name: "confirmation" })
    })
    .catch(reason => {
      this.checkoutStatus = "SERVER_ERROR";
    });
```

```
placeOrder(context, customerForm) {
  return ApiService.placeOrder({
    cart: context.state.cart,
    customerForm: customerForm
  })
  .then(orderDetails => {
    context.commit("CLEAR_CART");
    context.commit("SET_ORDER_DETAILS", orderDetails);
  });
```

```
placeOrder(order) {
  ...
  return fetch(url, options)
  .then(response => {
    if (response.ok) {
      return response.json();
    }
    throw Error("Network response not OK");
  });
```

dispatch: invoking an action in the Vuex store

customerForm is a JSON object of key-value pairs based on from on checkout page

```
submitOrder() {
  ...
  this.$store
    .dispatch("placeOrder", customerForm)
    .then(() => {
      this.checkoutStatus = "OK";
      this.$router.push({ name: "confirmation" })
    })
    .catch(reason => {
      this.checkoutStatus = "SERVER_ERROR";
    });
```

if Promise from placeOrder action is fulfilled, set checkout status to OK and go to confirmation page

if Promise from placeOrder action is rejected, set status to SERVER_ERROR

```
placeOrder(context, customerForm) {
  return ApiService.placeOrder({
    cart: context.state.cart,
    customerForm: customerForm
  })
  .then(orderDetails => {
    context.commit("CLEAR_CART");
    context.commit("SET_ORDER_DETAILS", orderDetails);
  });
```

```
placeOrder(order) {
  ...
  return fetch(url, options)
  .then(response => {
    if (response.ok) {
      return response.json();
    }
    throw Error("Network response not OK");
  });
```

```
submitOrder() {
  ...
  this.$store
    .dispatch("placeOrder", customerForm)
    .then(() => {
      this.checkoutStatus = "OK";
      this.$router.push({ name: "confirmation" })
    })
    .catch(reason => {
      this.checkoutStatus = "SERVER_ERROR";
    });
```

return a Promise from placeOrder in API service

order is cart and customer form

```
placeOrder(context, customerForm) {
  return ApiService.placeOrder({
    cart: context.state.cart,
    customerForm: customerForm
  })
  .then(orderDetails => {
    context.commit("CLEAR_CART");
    context.commit("SET_ORDER_DETAILS", orderDetails);
  });
```

the Promise from placeOrder service is fulfilled only if the order details are returned; therefore, store the details in Vuex state and clear the cart

```
placeOrder(order) {
  ...
  return fetch(url, options)
  .then(response => {
    if (response.ok) {
      return response.json();
    }
    throw Error("Network response not OK");
  });
```

if Promise from placeOrder service is rejected, do nothing; Error will propagate up to calling object

```
submitOrder() {
  ...
  this.$store
    .dispatch("placeOrder", customerForm)
    .then(() => {
      this.checkoutStatus = "OK";
      this.$router.push({ name: "confirmation" })
    })
    .catch(reason => {
      this.checkoutStatus = "SERVER_ERROR";
    });
```

```
placeOrder(context, customerForm) {
  return ApiService.placeOrder({
    cart: context.state.cart,
    customerForm: customerForm
  })
  .then(orderDetails => {
    context.commit("CLEAR_CART");
    context.commit("SET_ORDER_DETAILS", orderDetails);
  });
```

NOTE: if there is no second parameter present, fetch defaults to a GET request

```
const options = {
  method: "POST",
  body: JSON.stringify(order),
  headers: {
    "Content-Type": "application/json"
  }
};
```

fetch the resource at the URL. options object holds 3 pieces of info:
(1) POST directive
(2) body (JSON order as text)
(3) header denoting JSON content

```
placeOrder(order) {
  ...
  return fetch(url, options)
  .then(response => {
    if (response.ok) {
      return response.json();
    }
    throw Error("Network response not OK");
  });
```

if Promise returned by fetch is fulfilled, check the response

if response is OK (HTTP status code is in 200's) parse response into JSON object

if response is not OK, construct a JS Error and throw it

if Promise is rejected, a network error occurred, and the error is propagated up the chain

```
submitOrder() {
  ...
  this.$store
    .dispatch("placeOrder", customerForm)
    .then(() => {
      this.checkoutStatus = "OK";
      this.$router.push({ name: "confirmation" })
    })
    .catch(reason => {
      this.checkoutStatus = "SERVER_ERROR";
    });
```

```
placeOrder(context, customerForm) {
  return ApiService.placeOrder({
    cart: context.state.cart,
    customerForm: customerForm
  })
  .then(orderDetails => {
    context.commit("CLEAR_CART");
    context.commit("SET_ORDER_DETAILS", orderDetails);
  });
```

```
placeOrder(order) {
  ...
  return fetch(url, options)
  .then(response => {
    if (response.ok) {
      return response.json();
    }
    throw Error("Network response not OK");
  });
```

The JS code in the three separate files can be viewed as one long promise chain for a fetch action. A promise chain executes multiple asynchronous operations back-to-back; "then" functions return new promises; rejected promises are caught with the final "catch" function

```
fetch(url, options)
  .then(response => {
    if (response.ok) {
      return response.json();
    }
    throw Error("Network response not OK");
  })
  .then(orderDetails => {
    context.commit("CLEAR_CART");
    context.commit("SET_ORDER_DETAILS", orderDetails);
  })
  .then(() => {
    this.checkoutStatus = "OK";
    this.$router.push({ name: "confirmation" })
  })
  .catch(reason => {
    this.checkoutStatus = "SERVER_ERROR";
  })
```