More on current JS

Our coverage this term

- Modern JS in ordinary code: let and block scoping, fetch and promises, arrow functions
- Use of simple objects
- But no object classes: i.e., no way to stamp out a bunch of objects of a certain structure
- And no way to encapsulate code and data
- These both exist in JS, but not as easy to master as in Java or PHP (and not as closely related to each other either) and the weirdness shows up to mystify you.

How to study further?

- Could read Eloquent Javascript, free online
- Could enroll in free or paid JS courses online
- Eventually, probably want to use a JS framework
- https://rubygarage.org/blog/best-javascript-frameworks-for-front-end
- Satisfaction, etc. https://2019.stateofjs.com/front-end-frameworks/
- Vue is supposed to be the easiest to master, can be coded without new classes written by the programmer, just use theirs
- pizzaVue project does our student UI using Vue and pizza2_server

Idea of JS frameworks

- The software engineering challenge in writing JS/browser apps is the huge global variable, the DOM
- Way too easy to get mysterious bugs because some code fiddles with it unexpectedly somewhere.
- How can we bring order to this chaos?
- Idea: the DOM is a tree, so can be broken up into subtrees.
- Idea: with encapsulated objects, can create an object for a DOM subtree that controls the access to that DOM subtree
- Other code has to use the object's API to have any effect there.

Idea of JS frameworks

- This can be done recursively—an object/subtree can be broken up ito subtrees controlled by objects.
- Events can percolate up the tree of objects to a point where enough is known (in the object’s properties) to handle them properly.
- Commands can be sent down the tree.
- You can understand a part of the app without worrying about the rest of the app, except via stated methods
- Note this does not apply to JQuery (it’s not a framework)—a JQuery app can be a horrible mess. JS developers shun old JQuery apps.
- With Vue, the view can be specified by HTML with added directives like v-if. You can read the enhanced HTML and see what you’re getting, unlike React, where you have to build JSX for the object’s HTML programmatically.

Look at pizzaVue

- Only two sizable files with code:
- Index.html: HTML with enhancements
- main.js: the Javascript
- The libraries are brought in from the CDN (content delivery network), just as the JQuery library is supplied in many programs
- The styling is via Bootstrap CSS, also brought in from the CDNs.
- Intro to Vue

Look at pizzaVue: index.html

- index.html has <b-form> <b-table> etc. This is Bootstrap v4 styling. I'm using BootstrapVue, Vue with Bootstrap. It's easy to go from plain HTML to Bootstrap professional-looking styling.
- See v-if, v-model, etc. This is Vue itself accepting input and controlling what's displayed, etc.

pizzaVue’s main.js: sets up and uses objects with methods

- const Welcome = object for welcome page control/DOM
- const OrderPizza = object for order-pizza page control/DOM
- const API = helper object for fetches
- const routes =   config array for router (ref’ing Welcome and OrderPizza)
- const router = new VueRouter({   the router object for tabs control/DOM

```
    routes // short for `routes: routes`
})
```

- const app = new Vue({

```
    router                    // and the router knows about Welcome and OrderPizza
}).$mount('#app')   // ref’s <div id=“app> in index.html: where app shows up
```

JS code in main.js

- Never uses document.whatever calls or other direct references to the DOM
- i.e., leaves DOM manipulation to Vue, replaced with Vue control structures to use from the JS
- Still uses fetch, with help from axios, but could be direct use.
- So fills toppings and sizes arrays using fetch from the server (pizza2_server)
- How do these get displayed?

Displaying toppings, Vue case

```
<h2>Available Toppings</h2>
<ul>
  <li v-for="topping in toppings" :key="topping.topping" class="horizontal">
     {{ topping.topping }}
  </li>
</ul>
```

v-for: looping construct of Vue, here used for toppings array

{{ topping.topping }}: what to display for the element of the array, using a “template”

pizzaVue: Acknowledge Delivery of Pizzas

- This needs to capture user input—how is that done?
- How is this area turned on and off, since it’s not always displayed on the page?

```
<b-button variant="primary" v-if="showAckDeliveryButton"
   @click="this.ackClick">Acknowledge Delivery of Baked Pizzas
</b-button>
```

v-if: ref’s program variable to see whether to display this area

@click: a <button> is clickable, and this specifies what to call when the user clicks this button. It’s a method of the object in control of this part of the DOM, i.e. Welcome.

We still use ordinary JS programming…

- When the user clicks the acknowledge pizzas button, Vue arranges a call to Welcome’s ackClick method. That JS code:
  - Refreshes the orders array by using fetch
  - Loops through the orders array, finding orders for the current user
  - for each, uses fetch to update the order status
- Size Summary
  - With BootstrapVue, 257 lines of JS, 128 lines of HTML
  - Without: 378 lines of JS, 75 lines of HTML
  - So no huge change in size, but better styling, better basis for larger project