VueJS

Class 13 - Dynamic Image Gallery

Getting started

- Open the start file, which has the CSS already created and the Vue.js library ready for use:

-
  ```
  <div id="app">
          <h1 class="text-center">Submitted images</h1>
          <hr/>
          <div id="container">
              <ul>
                  <li v-for="(res, index) in myResult"><img
  class="listImg" v-bind:src="'img/' + res.Thumb">
                      <p>{{ res.Thumb }} - {{ res.Label }}</p>
                  </li>
              </ul>
          </div>
  </div>
  ```

- We are going to display a list of images on the page so we will loop through the list with **v-for**.

Displaying a list

- The image is placed in the img tag by binding the src, it then looks inside the img folder and adds the resulting thumbnail in there.

- `<img class="listImg" v-bind:src="'img/' + res.Thumb">`

- Then we display the name of the image and it's description below it:

- `<p>{{ res.Thumb }} - {{ res.Label }}</p>`

Add the Vue code

- Before the closing body add script tags and then add this inside:

-
  ```
  new Vue({
              el: '#app',
              data: {
                  bigImg: '',
                  cssClass: 'hide',
                  myResult: []
              },
              created: function() {
                  this.lookUpData();
              },
  });
  ```

- The data is holding variables that we will use later, but the myResult is going to be the list of images, but it’s empty! We’ll load them from an outside JSON file.

Created

- We added some new code in something called created - this means as soon as Vue is running, run the lookUpData function:

```
• created: function() {
          this.lookUpData();
  },
```

Method

- Let’s add the lookUpData function by creating our methods directly after the close of created },:

-
  ```
  methods: {
        lookUpData() {
              var app = this;

  axios.get('records.json').then(function(response) {
                    app.myResult = response.data.records;
              }).catch(function(error) {
                    app.myResult.Thumb = "error";
             })
        }
  }
  ```

Method

- We are using a small library called axios that loads outside data using something called XHR, add it to the head:
- `<script src="https://unpkg.com/axios@0.12.0/dist/axios.min.js"></script>`
- so we load ‘**records.json**’ and put it in the variable called ‘**myResult**’
- Save and try this and you should see your images appear

Adding Interaction

- What we need to do now is make a way for us to be able to click the image and then see a larger image displayed over the top.

- Add the following after container div to display the larger image over the top:

-
  ```
  <div id="panel" v-bind:class="cssClass">
       <div id="holder">
            <img class="listImg"
  v-bind:src="bigImg">
       </div>
  </div>
  ```

Two bindings

- We will make the panel appear by changing it’s class, at the moment the cssClass is set to ‘hide’
- **`v-bind:class="cssClass"`**
- The bigger image is controlled here:
- **`v-bind:src="bigImg"`**
- Now we need a way to click the thumbnail and make this happen - an event

Using it to bind CSS

- This is the first event you've created in Vue find the thumbnail image list and add the code in red:

- `<img class="listImg" v-bind:src="'img/' + res.Thumb" v-on:click="showImg(index)">`

- This is calling a special function in Vue known as a method. The function is called '**showImg**' and we pass in the current **index** of our list.

- This way we know which big image to add in to the panel.

Add the method

- In the Vue code add this function inside the methods, adding a comma , to the end of lookUpData:

-
  ```
  showImg(id) {
          var app = this;
          app.bigImg = 'img/' +
  app.myResult[id].Content;
          app.cssClass = 'show';
  }
  ```

- We pass in the index of the image so it uses that to get the big image. The show css class is added instead of hide. Save and test it.

Closing the image

- It works great except there is no way to close the image on the screen

- Let's fix that, add the code in red, to add more events:

-
```
<div id="panel" v-bind:class="cssClass" v-on:click="removeImg">
      <div id="holder">
           <img class="listImg" v-bind:src="bigImg" v-on:click="removeImg">
      </div>
</div>
```

- This adds a click event to both the panel and the image, as we don't know where the user will click.

Final method

- Go to your methods and add a comma , after the show image function, and add this:

-
  ```
  removeImg() {
          var app = this;
          app.cssClass = 'hide';
  }
  ```

- When the user clicks the panel or image the cssClass is set back to ‘**hide**’ and removes it from the page. OK save and test.

Going further

- Today’s lesson added CSS dynamically to bring elements onto the screen. Take a look at the CSS classes so that you can see exactly what is going on.

- We learned about loading the JSON file, so take a look at how that is structured, the names associated with the data

- We added events for the first time, make sure you fully understand the interactivity of events, ask if you are not sure!