



I just saw my wife trip and fall while carrying a laundry basket full of ironed clothes. I watched it all unfold!



My son told me he didn't understand cloning. I told him, 'That makes two of us.'

Consuming API's in VUE

Objectives

- Review typical HTTP request
- GET request
- 2xx Status Code indicates "success"
- Make an HTTP GET request using Postman and inspect the result
- Review JSON
- Use Axios
- Build a service object
- Use the Vue lifecycle hook `created()`
- Synchronous vs. asynchronous code
- What is a promise and how does it work
- Async coding techniques

But first...

A small review!!



View vs. Components

VIEWS

Ask a Question

At Sally's we realize we can't have every question ever asked, but we sure can try! If you find something we missed, use this form to add a new entry so others can learn from your experience.

Questions submitted will be reviewed in a timely manner. Or not. We don't know. I'm mostly just interested in acorns. But still submit them!

Add Question

Question

Answer

Difficulty

2

Add Question

Cancel




Photo by Alexey Savchenko on Unplash

What? We didn't have enough questions already?

- Sally the Squirrel

COMPONENTS

Question

Answer

Difficulty

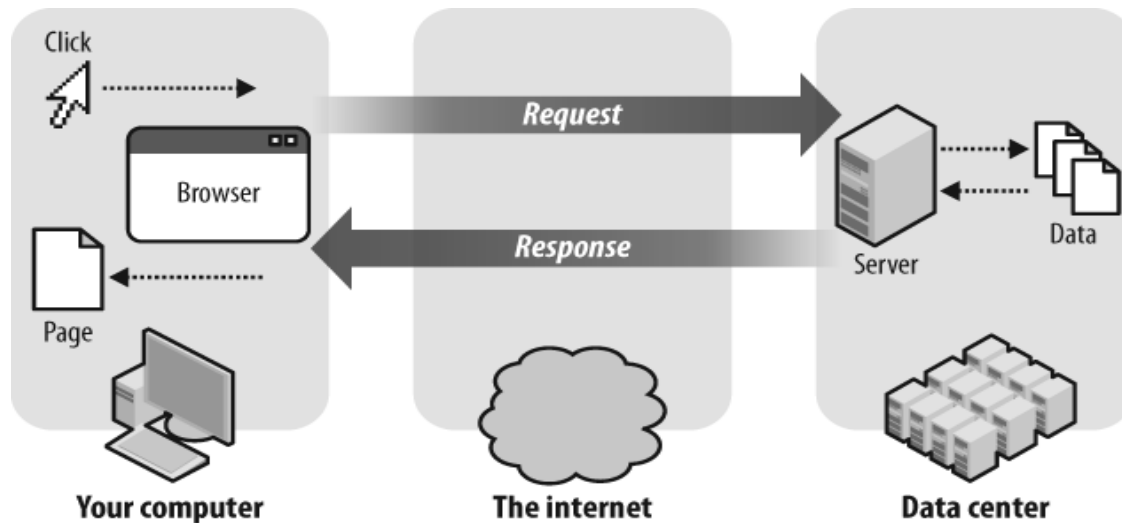
2

Add Question

Cancel

What is Routing?

- Routing allows users to be redirected to a certain component via a URL.
- Remember MVC Spring RequestMappings? Similar idea.



Defining Routes

```
1 import Home from '../views/Home.vue'
2 import About from '../views/About.vue'
3 import NotFound from '../views/NotFound.vue'
4
5 const routes = [
6   {
7     path: '/',           // Required
8     name: 'Home',        // Recommended, but not required
9     component: Home      // Required
10  },
11  {
12    path: '/About',
13    name: 'About',
14    component: About
15  },
16  {
17    path: '*',
18    name: 'NotFound',
19    component: NotFound
20  }
21 ];
```

Router-link

```
1 <small>
2   * - this form is a joke intended to demonstrate different input types.
3   Do not submit confidential information to untrusted sources.
4   See <router-link v-bind:to="{name: 'About'}">site disclaimer</router-link>
5   for more info.
6 </small>
```


Router-link styling

```
<div id="nav">
  <router-link :to="{name: 'users'}">Users</router-link> |
  <router-link :to="{ name: 'currencies' }">Currencies</router-link>
</div>
<router-view/>
</div>
</template>

<style>
|

#nav a {
  font-weight: bold;
  color: #1262b1;
}
#nav a.router-link-active {
  color: pink;
}
```

USERS | CURRENCIES

Users

ID	Name	Email
1	Leanne Graham	
2	Ervin Howell	
3	Clementine Bauch	

Router-view

```
1 <template>
2   <div id="app">
3     <AppHeader />
4     <router-view /> <!-- The currently active view will be presented here -->
5     <app-footer />
6   </div>
7 </template>
```

Dynamic Routes

```
1 const routes = [  
2   {  
3     path: '/Questions',  
4     name: 'Questions',  
5     component: Questions  
6   },  
7   {  
8     path: '/Questions/:id',  
9     name: 'QuestionDetails',  
10    component: QuestionDetails  
11  },  
12  {  
13    path: '/Questions/:id/Edit',  
14    name: 'EditQuestion',  
15    component: QuestionEdit  
16  },  
17  // others omitted...  
18 ];
```

Router-link Params

```
1 <section>
2   <!-- Params below can be accessed from the destination page via this.$route.params.parameterName -->
3   <router-link v-bind:to="{name: 'EditQuestion', params: {id: question.id}}">
4     Edit this Question
5   </router-link>
6 </section>
```

\$Router.Push

```
1 saveQuestion() {  
2   this.$store.commit('QUESTION_UPDATED', this.question);  
3   this.$router.push({name: 'QuestionDetails', params: {questionId: this.question.id}});  
4 }
```

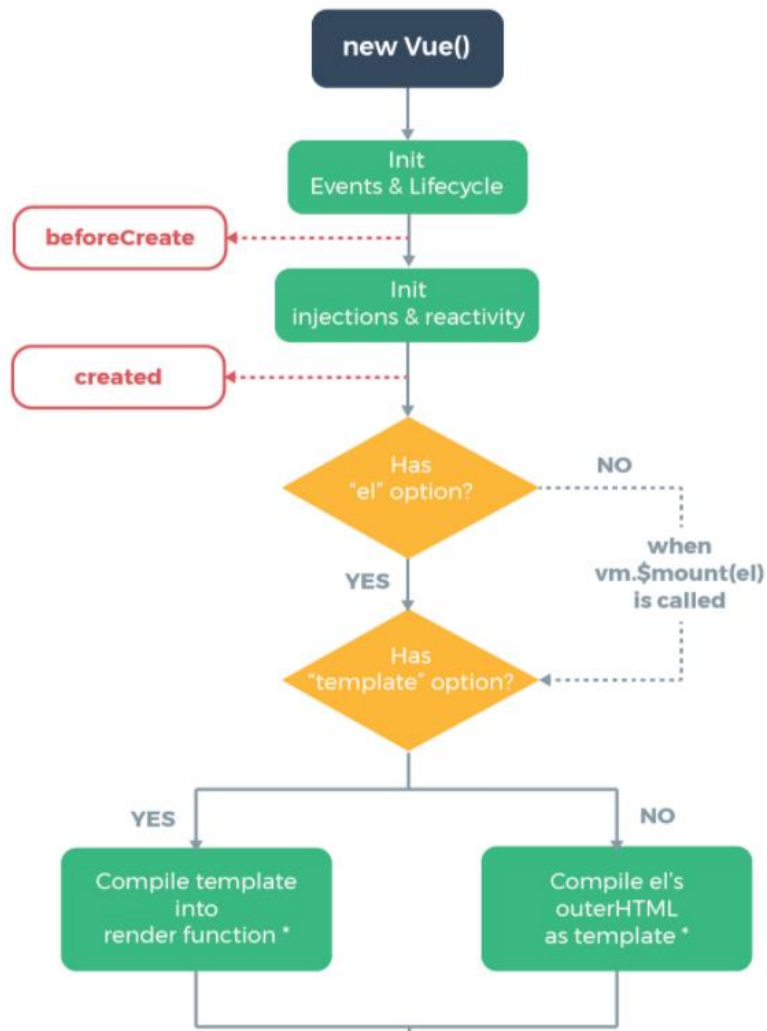
\$Route.Params

```
1 created() {  
2   const id = this.$route.params.id; // Grabs the route parameter named id, if it was present  
3   this.question = this.$store.state.questions.find(q => q.id === id);  
4  
5   if (!this.question) {  
6     this.$router.push({name: 'NotFound'});  
7   }  
8 }
```

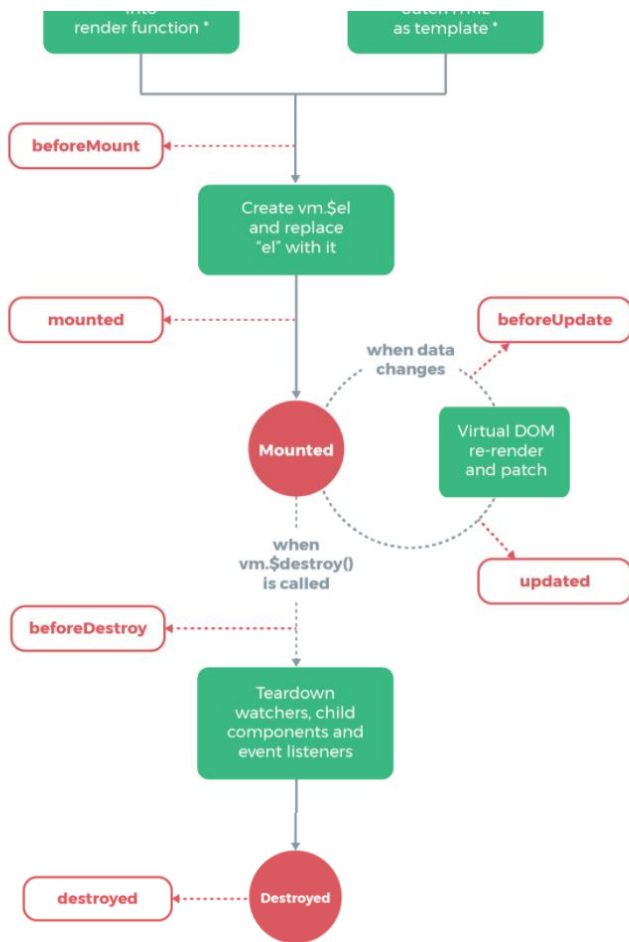
Lifecycle Events

beforeCreate() created()	→	Instance is being <i>created</i>
beforeMount() mounted()	→	Instance is being <i>mounted</i>
beforeUpdate() updated()	→	Instance is being <i>updated</i>
beforeDestroy() destroyed()	→	Instance is being <i>destroyed</i>

Lifecycle Events



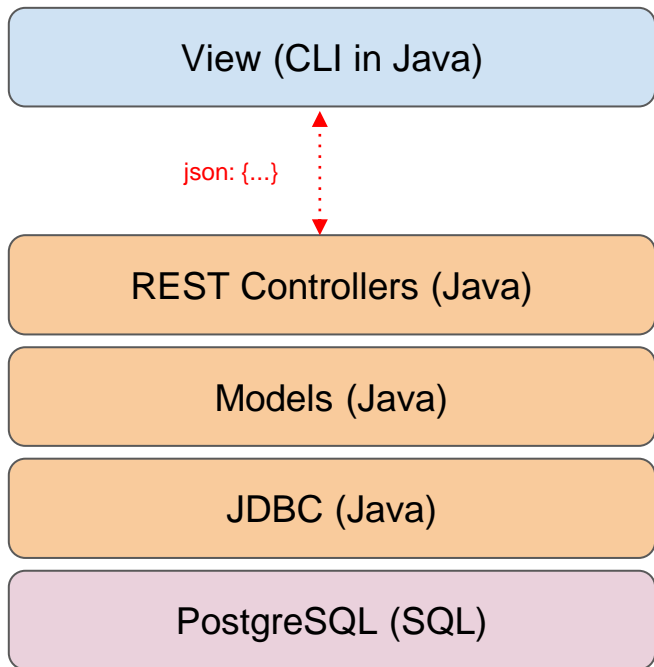
Lifecycle Events



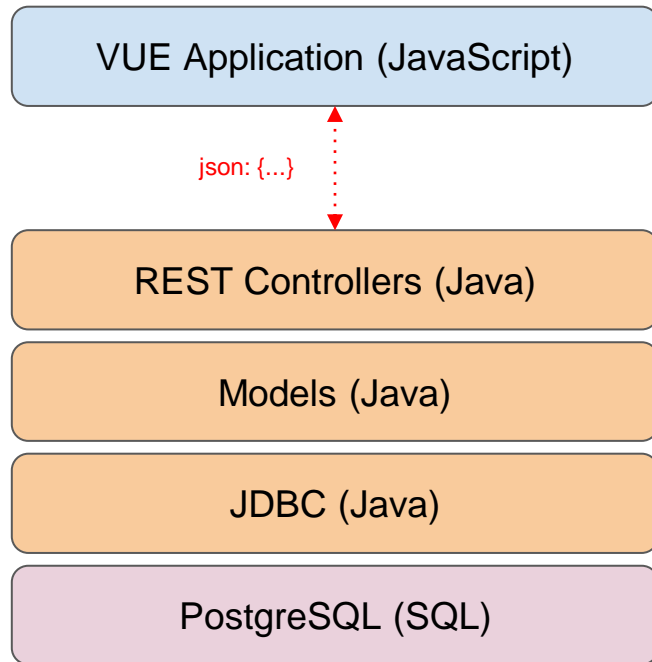
created()

```
1 // This fires after the component is created but before it renders.
2 // You can access any data, props, and $route info you need to here.
3 // This is also a good place to kick off requests for data your component will eventually need
4 created() {
5     const id = this.$route.params.id; // Grabs the route parameter named id, if it was present
6     this.question = this.$store.state.questions.find(q => q.id === id);
7
8     if (!this.question) {
9         this.$router.push({name: 'NotFound'});
10    }
11 }
```

Module 2 vs Module 3 (Comparing Stacks)

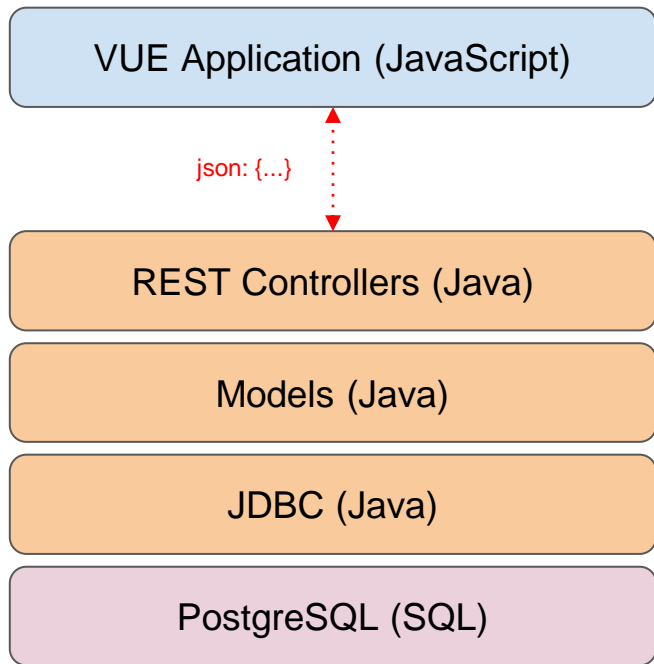


Module 2



Module 3

Consuming API's with VUE



We will use all the techniques we've learned so far in VUE to construct an application capable of consuming a REST API.

While we can use fetch syntax from Vanilla JS, we will be learning Axios for its ease.

Module 3

Requests to a REST Endpoint

Recall that a REST controller can be configured to handle various types of requests. Let's review them:

- **GET**: Ideally suited to retrieve all the records from a REST endpoint.
- **GET (with path variable)**: We can configure path variables (i.e. `doggo/1`) to retrieve a single record of data.
- **POST**: Ideally suited for inserting new data into the data source.
- **PUT**: Ideally suited for updating an existing record within a data source.
- **DELETE**: Ideally suited for removing an existing record from the data source.

HTTP STATUS CODES

2xx Success

200 Success / OK

3xx Redirection

301 Permanent Redirect

302 Temporary Redirect

304 Not Modified

4xx Client Error

401 Unauthorized Error

403 Forbidden

404 Not Found

405 Method Not Allowed

5xx Server Error

501 Not Implemented

502 Bad Gateway

503 Service Unavailable

504 Gateway Timeout

418 I'M A TEAPOT



Let's code!

*When you help someone fix
their code but you can't fix
your own*



Synchronous vs Asynchronous



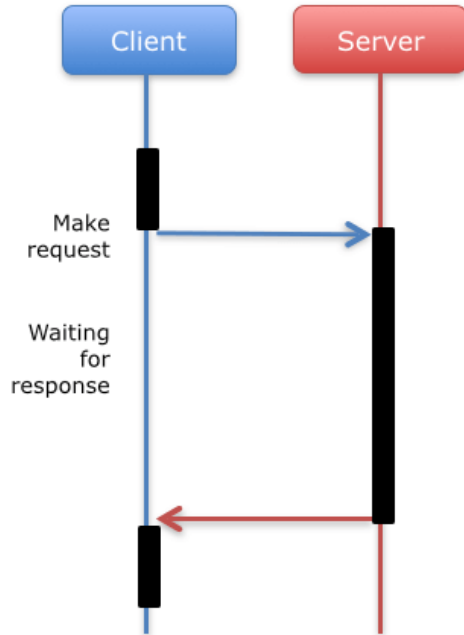
©2015 by CWOOD



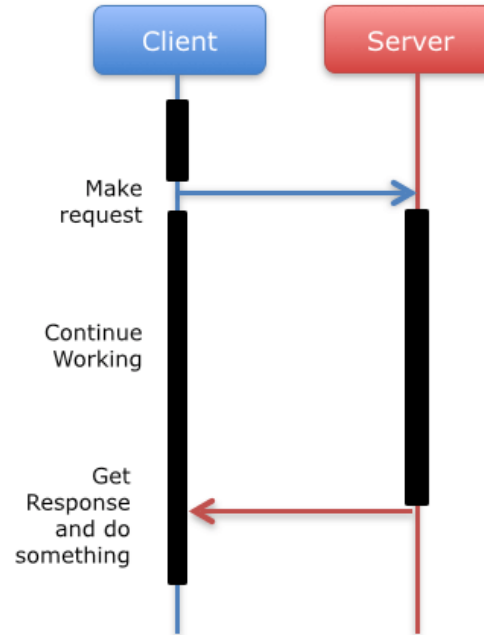
smallblueyonder.com

Sync vs Async

Synchronous



Asynchronous



AXIOS

```
npm install axios --save
```

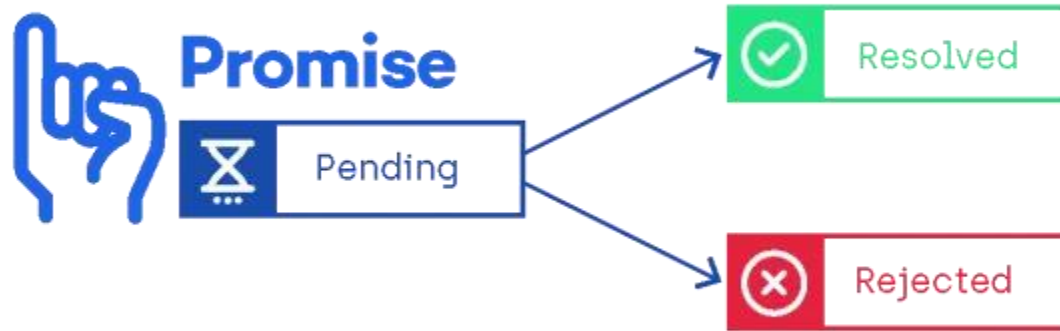
Axios Get

```
1 /**
2  * Gets all items on the server
3  * @returns {Promise} a promise that will complete with a list of items
4  */
5 getAllItems() {
6     // Create our Axios instance used to communicate with the server
7     const http = axios.create({
8         baseURL: 'https://some.website.com'
9     });
10
11     return http.get('/items'); // This is added to the end of baseURL specified above
12 }
```

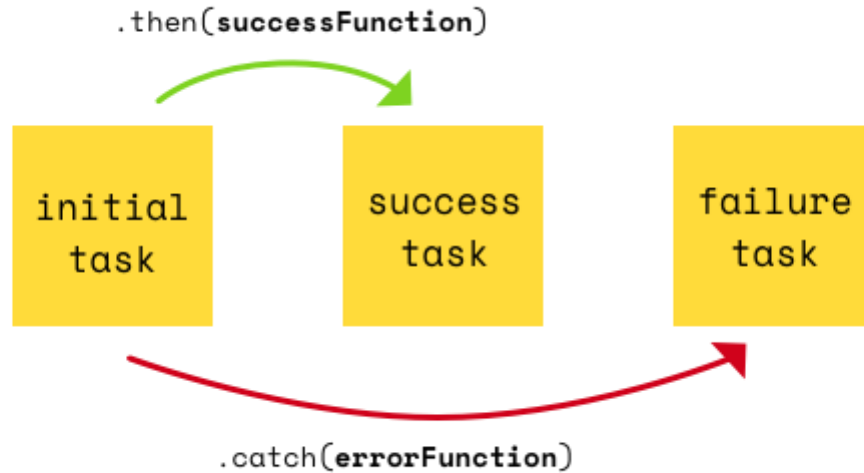
What Is A Promise?



What Is A Promise?



Using A Promise



Axios Get

```
1 /**
2  * Gets all items on the server
3  * @returns {Promise} a promise that will complete with a list of items
4  */
5 getAllItems() {
6   // Create our Axios instance used to communicate with the server
7   const http = axios.create({
8     baseURL: 'https://some.website.com'
9   });
10
11   return http.get('/items'); // This is added to the end of baseURL specified above
12 }
```

Axios Get

```
1 /**
2  * Gets all items on the server
3  * @returns {Promise} a promise that will complete with a list of items
4  */
5 getAllItems() {
6   // Create our Axios instance used to communicate with the server
7   const http = axios.create({
8     baseURL: 'https://some.website.com'
9   });
10
11   return http.get('/items'); // This is added to the end of baseURL specified above
12 }
```

```
1 getAllItems().then(response => {
2   // response.data is loaded from the contents of the response body
3   // It's typically going to be a JavaScript object or an array of objects
4   const items = response.data;
5   this.$store.commit('ITEMS_LOADED', items);
6 });
```

Objectives

- Review typical HTTP request
- GET request
- 2xx Status Code indicates "success"
- Make an HTTP GET request using Postman and inspect the result
- Review JSON
- Use Axios
- Build a service object
- Use the Vue lifecycle hook `created()`
- Synchronous vs. asynchronous code
- What is a promise and how does it work
- Async coding techniques

Let's code!

When you try to look at
the code you wrote a month ago

