

WEB SERVICES

MODULE 3, DAY 18



PULSE SURVEY

ROUTING REVIEW

VIEWS 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 QuestionCancel




Photo by Alexey Savchenko on Unplash

What? We didn't have enough questions already?

- Sally the Squirrel

COMPONENTS

Question

Answer

Difficulty

2

Add QuestionCancel

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-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>
```

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

```
1 <!-- By default Vue adds the router-link-active class to routes beginning with the current page's path -->
2 <nav>
3   <router-link :to="{name: 'Home'}"          active-class="active" exact>Home</router-link>
4   <router-link :to="{name: 'Questions'}"      active-class="active" exact>Questions</router-link>
5   <router-link :to="{name: 'AskQuestion'}"    active-class="active">Ask a Question</router-link>
6   <router-link :to="{name: 'Services'}"       active-class="active">Services</router-link>
7   <router-link :to="{name: 'About'}"          active-class="active">About</router-link>
8   <router-link :to="{name: 'Contact'}"        active-class="active">Contact</router-link>
9 </nav>
```


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 created*

beforeMount()
mounted() → *Instance is being mounted*

beforeUpdate()
updated() → *Instance is being updated*

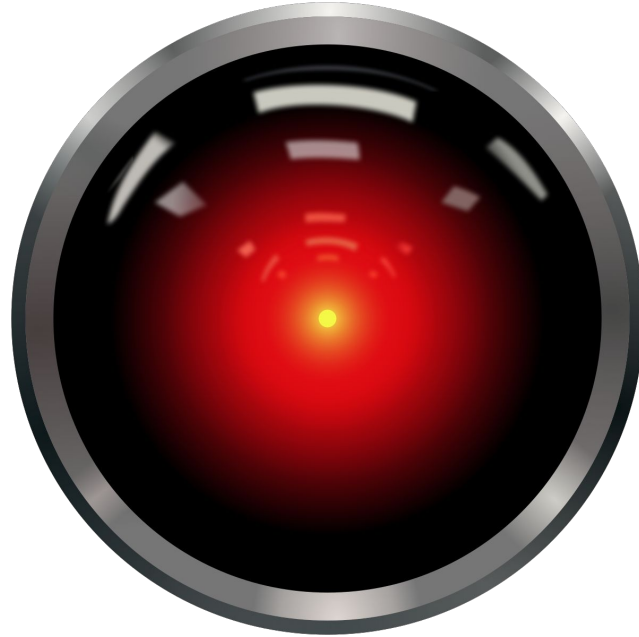
beforeDestroy()
destroyed() → *Instance is being destroyed*

created()

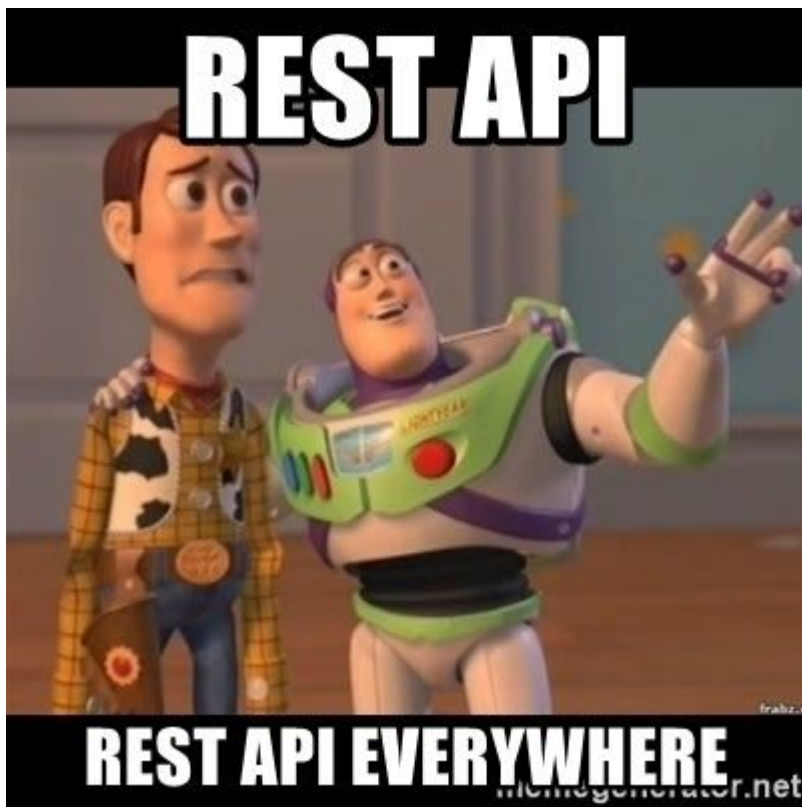
```
1 // This fires after the component is created but before it renders.
2 // You can access any data, props, and $route info you need to here.
3 // This is also a good place to kick off requests for data your component will eventually need
4 created() {
5     const id = this.$route.params.id; // Grabs 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 }
```

ROUTING QUIZ

DON'T FORGET



TO RECORD



AGENDA

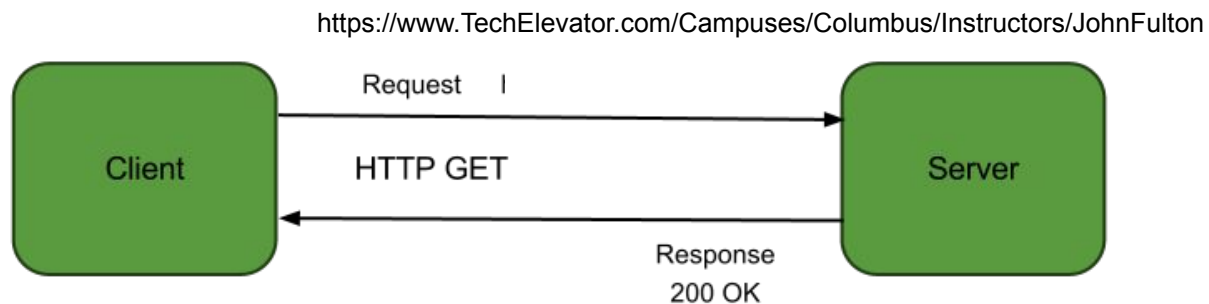
- REST Refresher
- Synchronous vs Asynchronous
- Web Service Requests
- Promises
- JavaScript Services
- A Teaser on CORS

REST

REST

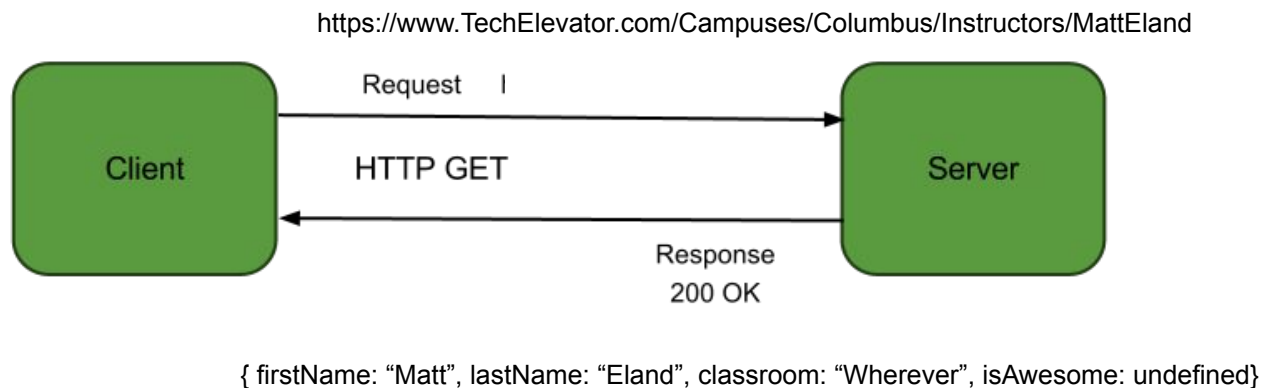


REST

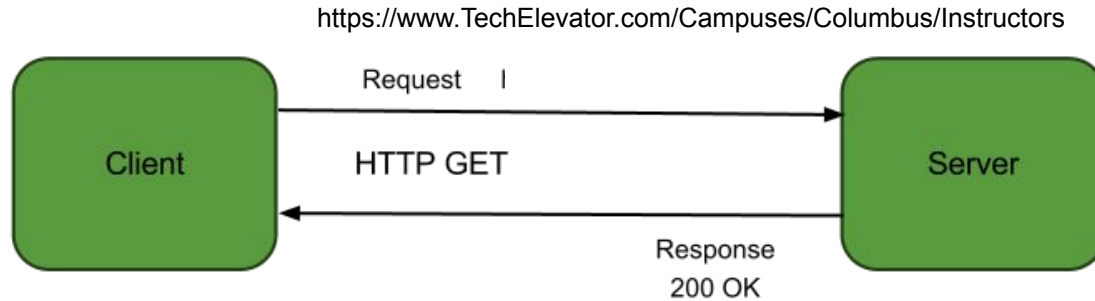


`{ firstName: "John", lastName: "Fulton", classroom: ".NET", isAwesome: true }`

REST



REST



```
[  
  { firstName: "John", lastName: "Fulton", classroom: ".NET", isAwesome: true },  
  { firstName: "Matt", lastName: "Eland", classroom: "Wherever", isAwesome: undefined },  
  // ... more instructors here  
]
```

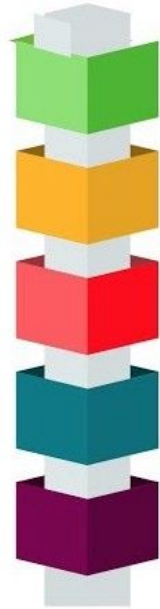
VERBS & RESOURCES

GET	/pet/{petId}	Find pet by ID
PUT	/pet	Update an existing pet
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image

STATUS CODES



STATUS CODES



1XX
INFORMATIONAL

2XX
SUCCESS

“Fine. Done.”

3XX
REDIRECTION

4XX
CLIENT ERROR

“You messed up.”

5XX
SERVER ERROR

“I messed up”

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

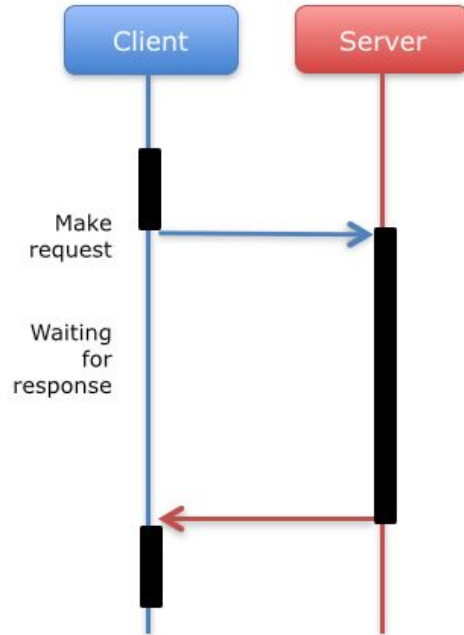
418 I'M A TEAPOT



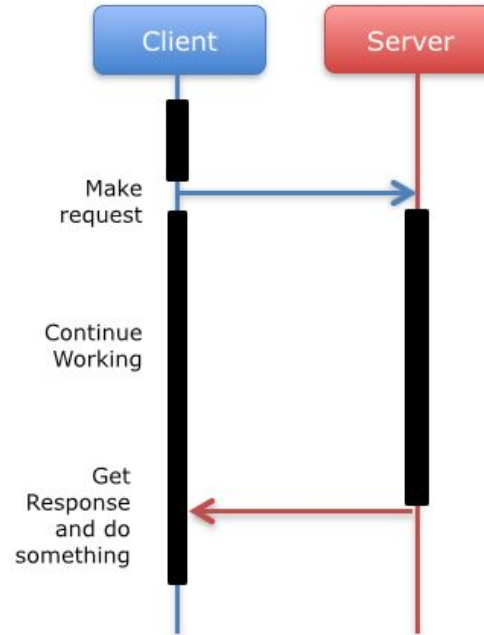
SYNCHRONOUS & ASYNCHRONOUS

SYNC VS ASYNC

Synchronous



Asynchronous

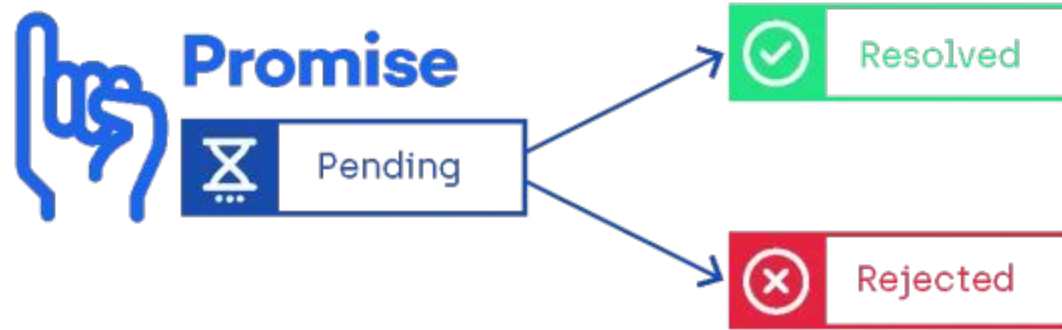


AXIOS

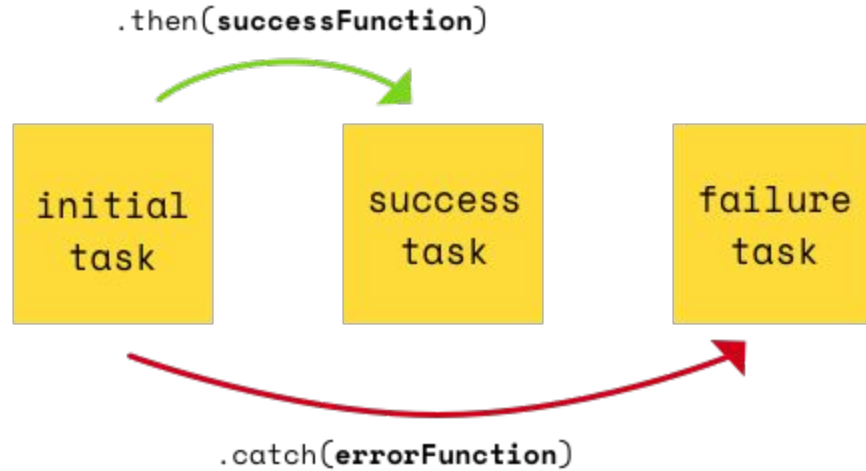
AXIOS GET

```
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?



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 }
```

```
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 });
```

GETTING QUESTIONS

```
8 [Route("[controller]")]
9 [ApiController]
10 public class QuestionsController : ControllerBase
11 {
12     private readonly IQuestionRepository _repository;
13
14     public QuestionsController(IQuestionRepository repository)
15     {
16         _repository = repository;
17     }
18
19     [HttpGet]
20     public ActionResult<IEnumerable<QuestionEntry>> Get()...
24
25     [HttpGet("{id}")]
26     public ActionResult<QuestionEntry> GetById(int id)...
35
36     [HttpPut("{id}")]
37     public ActionResult<IEnumerable<QuestionEntry>> Update(int id, QuestionEntry newEntry)...
54
55     [HttpDelete("{id}")]
56     public ActionResult<IEnumerable<QuestionEntry>> Delete(int id)...
65
66     [HttpPost]
67     public ActionResult<QuestionEntry> Add(QuestionEntry entry)...
73
74 }
```

PROMISE ERROR HANDLING

```
1 getAllQuestions()  
2   .then(response => {  
3     // Data is loaded from the contents of the response body  
4     // It's typically going to be a JavaScript object or an array of objects  
5     const questions = response.data;  
6     this.$store.commit('QUESTIONS_LOADED', questions);  
7   })  
8   .catch(error => {  
9     console.error('An error occurred trying to load questions', error);  
10  })  
11  .finally(() => console.log('Finally!'));
```

WHY PROMISES?

```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```



SERVICES

```
1 import axios from 'axios';
2
3 // Create our Axios instance used to communicate with the server
4 const http = axios.create({
5   baseURL: 'https://some.url.net'
6 });
7
8 export default { // This object is what other files will import via the import keyword
9
10   getAllItems() {
11     return http.get('/items'); // This is added to the end of baseURL specified above
12   },
13
14   getItem(id) {
15     return http.get(`/items/${id}`);
16   },
17
18   update(myItem) {
19     return http.put(`/items/${myItem.id}`, myItem);
20   },
21
22   create(myItem) {
23     return http.post('/items', myItem);
24   },
25
26   delete(myItem) {
27     return http.delete(`/items/${myItem.id}`);
28   }
29
30 };
```

AXIOS VERBS

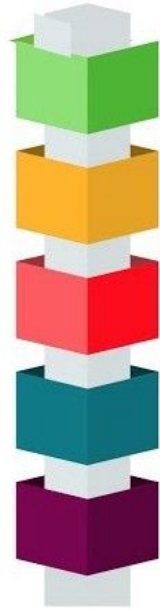
```
getItem(id) {  
  return http.get(`/items/${id}`);  
},  
  
update(myItem) {  
  return http.put(`/items/${myItem.id}`, myItem);  
},  
  
create(myItem) {  
  return http.post('/items', myItem);  
},  
  
delete(myItem) {  
  return http.delete(`/items/${myItem.id}`);  
}
```

Review

VERBS & RESOURCES

GET	/pet/{petId}	Find pet by ID
PUT	/pet	Update an existing pet
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image

STATUS CODES



1XX
INFORMATIONAL

2XX
SUCCESS

“Fine. Done.”

3XX
REDIRECTION

4XX
CLIENT ERROR

“You messed up.”

5XX
SERVER ERROR

“I messed up”

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

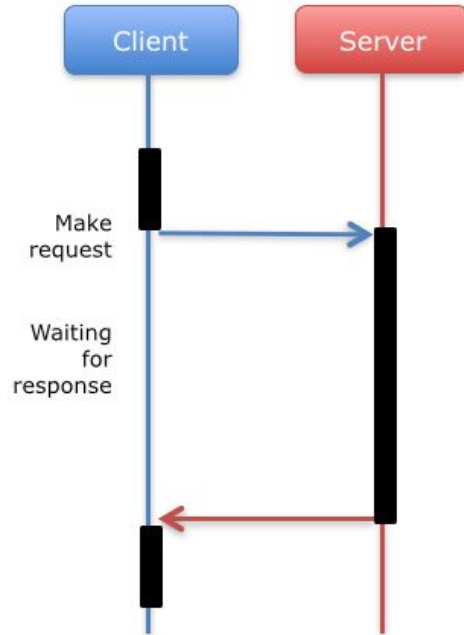
502 Bad Gateway

503 Service Unavailable

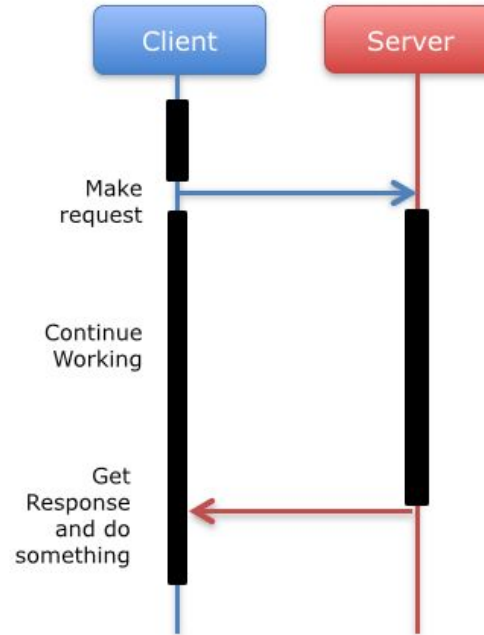
504 Gateway Timeout

SYNC VS ASYNC

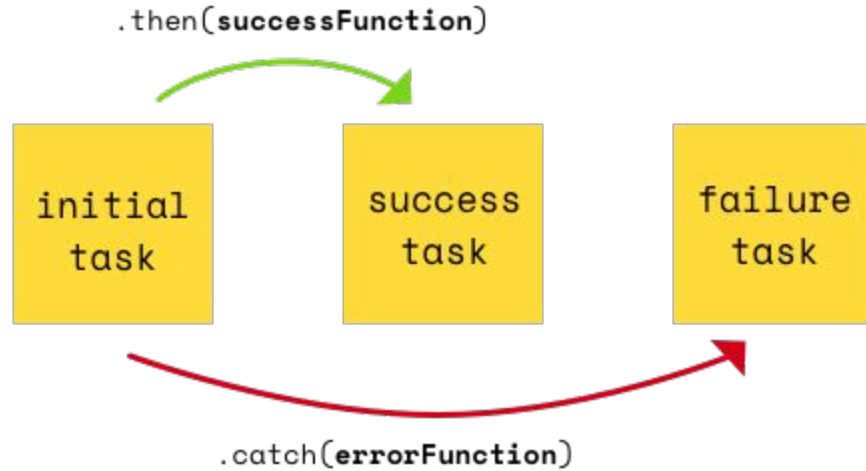
Synchronous



Asynchronous



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 }
```

```
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 });
```

PROMISE ERROR HANDLING

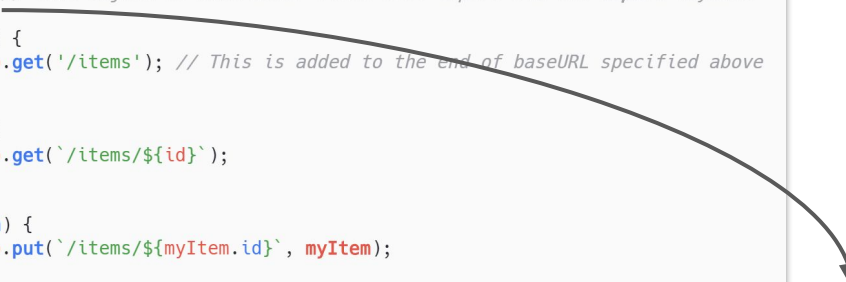
```
1 getAllQuestions()  
2   .then(response => {  
3     // Data is loaded from the contents of the response body  
4     // It's typically going to be a JavaScript object or an array of objects  
5     const questions = response.data;  
6     this.$store.commit('QUESTIONS_LOADED', questions);  
7   })  
8   .catch(error => {  
9     console.error('An error occurred trying to load questions', error);  
10  })  
11  .finally(() => console.log('Finally!'));
```

AXIOS VERBS

```
getItem(id) {  
  return http.get(`/items/${id}`);  
},  
  
update(myItem) {  
  return http.put(`/items/${myItem.id}`, myItem);  
},  
  
create(myItem) {  
  return http.post('/items', myItem);  
},  
  
delete(myItem) {  
  return http.delete(`/items/${myItem.id}`);  
}
```


SERVICES

```
1 import axios from 'axios';
2
3 // Create our Axios instance used to communicate with the server
4 const http = axios.create({
5   baseURL: 'https://some.url.net'
6 });
7
8 export default { // This object is what other files will import via the import keyword
9
10   getAllItems() {
11     return http.get('/items'); // This is added to the end of baseURL specified above
12   },
13
14   getItem(id) {
15     return http.get(`/items/${id}`);
16   },
17
18   update(myItem) {
19     return http.put(`/items/${myItem.id}`, myItem);
20   },
21
22   create(myItem) {
23     return http.post('/items', myItem);
24   },
25
26   delete(myItem) {
27     return http.delete(`/items/${myItem.id}`);
28   }
29
30 };
```



```
1 import QuestionService from './services/QuestionService.js';
2
3 QuestionService.getAllItems()
4   .then(response => console.log(response));
```

ADDITIONAL RESOURCES

- [Axios Documentation](#)
- [Student Book GET](#)
- [Student Book POST / PUT / DELETE](#)

**HAPPY
CODING!**

