
Exercise. Todo (part 3) – Making HTTP calls from client to the back-end

This exercise demonstrates, how to call previously made backend. Make sure, that your backend is working and NodeJS server is running before starting to make this exercise.

Key learnings for this part are:

- Making HTTP calls from the front-end using JavaScript
- Get data from back-end
- Post data to back-end
- Error handling when making HTTP calls

Create a separate function for rendering a task. This (same) function will be used, when a new task is added, or tasks are retrieved from backend. Test out that front-end is still working after making following modification.

```
const renderTask = (task) => {  
  const li = document.createElement('li')  
  li.setAttribute('class', 'list-group-item')  
  li.innerHTML = task  
  list.append(li)  
}  
  
input.addEventListener('keypress', (event) => {  
  if (event.key === 'Enter') {  
    event.preventDefault()  
    const task = input.value.trim()  
    if (task !== '') {  
      renderTask(task)  
      input.value = ''  
    }  
  }  
})
```

Add constant variable that holds url for the backend. By default, input field is disabled.

```
const BACKEND_ROOT_URL = 'http://localhost:3001'

const list = document.querySelector('ul')
const input = document.querySelector('input')

input.disabled = true
```

Define following function that fetches data from the backend by making HTTP call. JSON is received as response. JSON (array) is looped through and each JSON object holding task is rendered to the UI. User can add new tasks after data is retrieved.

```
const li = document.createElement('li')
li.setAttribute('class', 'list-group-item')
li.innerHTML = task
list.append(li)
}

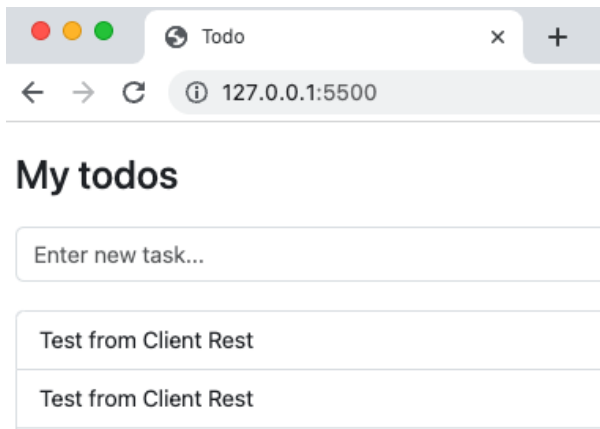
const getTasks = async () => {
  try {
    const response = await fetch(BACKEND_ROOT_URL)
    const json = await response.json()
    json.forEach(task => {
      renderTask(task.description)
    })
    input.disabled = false
  } catch (error) {
    alert("Error retrieving tasks " + error.message)
  }
}

const saveTask = async (task) => {
```

Call getTasks function as a last line in JavaScript.

```
49     saveTask(task).then((json)=> {
50       renderTask(task)
51       input.value = ''
52     })
53   }
54 }
55 })
56
57 getTasks()
```

Test out that website works. You should be able to see data retrieved from database.



Define function for saving task. A fetch call using post HTTP method is implemented. Provide additional parameters for fetch (method, headers and body including data in JSON format). Call is made to /new endpoint on the backend. This function return a [promise](#) (return is without await), which needs to handle by the caller.

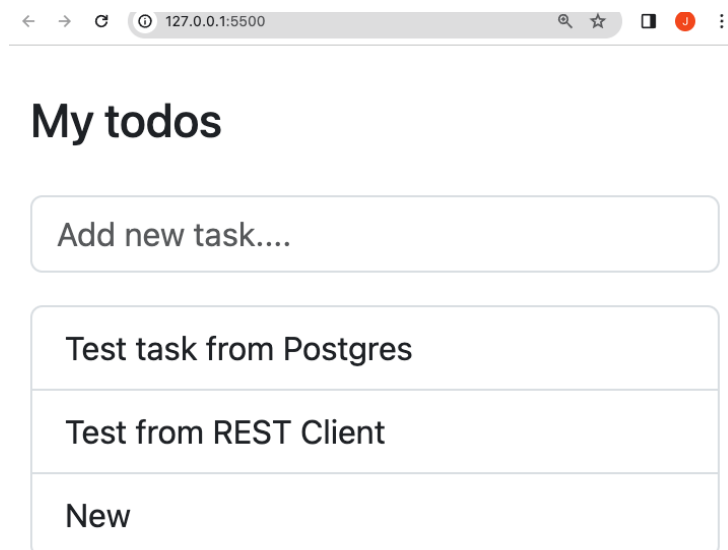
```
24     alert("Error retrieving tasks " + error.message)
25   }
26 }
27
28 const saveTask = async (task) => {
29   try {
30     const json = JSON.stringify({description: task})
31     const response = await fetch(BACKEND_ROOT_URL + '/new',{
32       method: 'post',
33       headers: {
34         'Content-Type':'application/json'
35       },
36       body: json
37     })
38     return response.json()
39   } catch (error) {
40     alert("Error saving task " + error.message)
41   }
42 }
43
44 input.addEventListener('keypress',(event) => {
45   if (event.key === 'Enter') {
```

Modify code that adds new task. Since function saveTask returns a promise, .then syntax is used to handle to return value. It means, that asynchronous response is handled after execution is finished. In this case new task is rendered ("printed") into UI

and input field is emptied after backend is finished with saving data into database and returns a response.

```
43
44   input.addEventListener('keypress', (event) => {
45     if (event.key === 'Enter') {
46       event.preventDefault()
47       const task = input.value.trim()
48       if (task !== '') {
49         saveTask(task).then((json) => {
50           renderTask(task)
51           input.value = ''
52         })
53       }
54     }
55   })
56
```

Test out that you can add new tasks and data is saved into database.



← → ↻ 127.0.0.1:5500 🔍 ☆ 📱 🔴 ⋮

My todos

Add new task....

Test task from Postgres

Test from REST Client

New