САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет по лабораторной работе

взаимодействие с внешним API

Выполнил:
В. Д. Елистратов

Группа: K33392

Проверил:
Добряков Д. И.

Санкт-Петербург
2023

1. Задача

Варианты остаются прежними. Теперь Вам нужно привязать то, что Вы делали в ЛР1 к внешнему API

2. Ход работы

2.1. Скрипт регистрации, логина

Два класса и их методы отвечают за процессы регистрации и логина соответственно.

```
class LogInC{
    constructor (){
        this.data = {};
    }
    async worker(event){
        let fields = Array.from(event.target.querySelectorAll('input'));
        for(const field of fields){
            this.data[field.name] = field.value;
        }

        await this.req();
    }

    async req(){
        const response = await fetch('http://localhost:3000/login', {
            method: "POST",
            credentials: 'include',
            body: JSON.stringify(this.data),
            headers: {
                'Accept': 'application/json',
                'Content-Type': 'application/json',
            }
        })

        const responseJson = await response.json();

        const {accessToken, user} = responseJson;
        if(accessToken){
            localStorage.accessToken = accessToken;
            localStorage.user = JSON.stringify(user);
        }
    }

}
```

Класс **LogInC**
Метод **worker** принимает данные из модального окна логина и переносит их в переменную data.
Метод **req** выполняет запрос на запись данных к JSON-server

```
class RegistrationC{
    constructor(){
        this.data = {};
    }
    async worker(event) {
        let fields = Array.from(event.target.querySelectorAll('input'));
        for(const field of fields){
            this.data[field.name] = field.value;
        }
        this.data['capCount'] = 0;
        this.data['lastUpdate'] = (new Date(new Date())).getTime();
        //console.log(this.data);

        await this.req()
    }

    async req(){
        const response = await fetch('http://localhost:3000/users', {
            method: "POST",
            body: JSON.stringify(this.data),
            headers: {
                'Content-Type': 'application/json'
            }
        })
    }

}
```

Класс **RegistrationC**.
Метод **worker** принимает данные из модального окна регистрации и переносит их в переменную data.
Метод **req** выполняет запрос на запись данных к JSON-server

```
loginC = new LogInC()
registrationC = new RegistrationC()

async function login(event){
    event.preventDefault();
    await loginC.worker(event);
    location.reload();
}

async function registration(event){
    event.preventDefault();
    await registrationC.worker(event);
    location.reload();
}
```

Создание экземпляров классов и вызов методов. Функции реагируют на события в HTML коде.

2.2. Скрипт для создания новых и вывода на сайт информации об открытых капсулах

```
class CapsuleC{
    constructor (event){
        this.data = {};
    }
    async encapsulate(event){
        const form = event.target.elements
        for(const field of form){
            if(field.name == "access"){
                if(field.checked){
                    this.data[field.name] = 'FFA';
                }
                else{
                    this.data[field.name] = 'my';
                }
                continue;
            }
            this.data[field.name] = field.value;

        }
        this.data['userId'] = JSON.parse(localStorage.getItem('user'))['id'];
        this.data['userName'] = JSON.parse(localStorage.getItem('user'))['
           userName'];
        const response = await fetch('http://localhost:3000/664/capsules', {
            method: "POST",
            body: JSON.stringify(this.data),
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${localStorage.accessToken}`
            }
        })

        const uData = JSON.parse(localStorage.user)
        uData['capCount'] += 1;
        localStorage.user = JSON.stringify(uData)

        this.reloadUser();
    }

    async reloadUser(){
        const data = JSON.parse(localStorage.user);
        const id = data['id'];
        const url = `http://localhost:3000/users/${id}`
        const response = await fetch(url, {
            method: "PUT",
            body: JSON.stringify(data),
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${localStorage.accessToken}`
            }
        })
    }


```

```
53      async getCapsule(id){
54          const searchParams = new URLSearchParams();
55          searchParams.set('id', id)
56
57          const url = "http://localhost:3000/660/capsules?" +
                searchParams.toString();
58
59          const response = await fetch(url, {
60              method: "GET",
61              headers: {
62                  'Content-Type': 'application/json',
63                  'Authorization': `Bearer ${localStorage.accessToken}`
64              }
65          })
66
67          const capsule = await response.json();
68
69          return capsule[0];
70      }
71
72      async decapsulate(event){
73          let fields = Array.from(event.target.querySelectorAll('button'));
74          const capsule = await this.getCapsule(fields[0].value);
75
76          var myModal = new bootstrap.Modal(document.getElementById('
                openCapsuleFormModalId'))
77          const modalBody = document.querySelector('#modalCapsuleOpenedBody');
78          const modalTitle = document.querySelector('#modalCapsuleOpenedTitle')
                ;
79
80          modalTitle.innerHTML = `${capsule['name']}`;
81
82          modalBody.innerHTML = `
83              <div class="mb-4">
84                  <p class="fw-bold">Описание:</p>
85                  <div>${capsule['description']}</div>
86              </div>
87
88              <hr class="content-dividing-line rounded mt-3"/>
89                  <div class="form-outline mb-10">
90                  <p class="fw-bold mb-5">Содержимое капсулы</p>
91              </div>
92
93              <div class="mb-4">
94                  <p class="fw-bold">Послание:</p>
95                  <div>${JSON.stringify(capsule['text'])}</div>
96              </div>
97
98              <div class="mb-4">
99                <p class="fw-bold">Прикрепленные файлы:</p>
100               <div src="">${capsule['files']}</div>
101             </div>
102         `;
103
104         myModal.show()
105     }
106 }
```

Класс **CapsuleC**
Метод **encapsulate** принимает данные из модального окна создания капсулы и переносит их в переменную data. После чего отправляет созданную капсулу на JSON-server. После выполняется увеличение кол-ва созданных пользователем капсул на 1.
Метод **reloadUser** перезаписывает данные о пользователе
Метод **decapsulate** считывает id открытой капсулы из HTML кода и отправляет запрос на получение капсулы методу **getCapsule**. Далее он записывает полученные данные в модальное окно и вызывает его.
Метод **getCapsule** получает данные о капсуле с JSON-server по заданному ID.

```
const capsule = new CapsuleC();

async function encapsulate(event){
    event.preventDefault();
    await capsule.encapsulate(event);
    location.reload();
}

async function decapsulate(event){
    event.preventDefault();
    capsule.decapsulate(event);
}
```

Создание экземпляра класса и вызов методов. Функции реагируют на события в HTML коде.

2.3. Скрипт для выгрузки данных на страницу

```
class HeaderC{
    constructor(){
        this.defaultHeader = `
            <div class="col-xl-1 col-md-2 col-sm-3">
                <button type="button" class="btn btn-lg btn-my-main"
                    data-bs-toggle="modal" data-bs-target="#loginFormModal">
                    Вход</button>
            </div>
            <div class="col-xl-2 col-md-3 col-sm-4">
                <button type="button" class="btn btn-lg btn-my-main"
                    data-bs-toggle="modal" data-bs-target="#regFormModal">
                    Регистрация</button>
            </div>
        `;

        this.loggedHeader = `
            <div class="col-xl-1 col-md-2 col-sm-3 me-5">
                <button type="button" class="btn btn-lg btn-my-main"
                    data-bs-toggle="modal" data-bs-target="#
                    CreateCapsuleFormModalId">Создать</button>
            </div>
            <div class="col-xl-2 col-md-3 col-sm-4">
                <p class="text-end mb-0 mt-0"><a href="/html/profile.html"
                    class="btn btn-my-card btn-lg">Профиль</a></p>
            </div>
        `;

```

```
21          this.capsuleAddButton = `
22              <div class="row ms-1 me-1">
23                  <button type="button" class="btn btn-lg btn-my-main"
                        data-bs-toggle="modal" data-bs-target="#
                        CreateCapsuleFormModalId">+</button>
24              </div>
25          `;
26      }
27
28      async worker(){
29          let logged = localStorage.accessToken;
30          let header_L_R_P_B = document.querySelector('#pageHeaderButtons');
31          //let headerCapsuleAddButton = document.querySelector('#
                capsuleAddButton');
32          if(logged){
33              //headerCapsuleAddButton.innerHTML = this.capsuleAddButton;
34              header_L_R_P_B.innerHTML = this.loggedHeader;
35          }
36          else{
37              header_L_R_P_B.innerHTML = this.defaultHeader;
38              //headerCapsuleAddButton.innerHTML = this.capsuleAddButton;
39          }
40      }
41  }
```

Класс **HeaderC**

Метод **worker**, в зависимоти от того, выполнил пользователь вход в систему или нет, выводит на страницу один из двух варинтов "шапки"

```
class SelectorC{
    constructor(){
        const selOption = sessionStorage.selOption;
        this.defaultSelectingOptions = `
            <div class="col-4 justify-c form-check">
                <input class="form-check-input me-1" type="radio" name="
                    OwnerSelect" value="All" id="RadioBtnAll" onclick="
                    selChange(event)" ${selOption === 'All' ? 'checked' : ''}
                    />
                <label class="form-check-label" for="RadioBtnAll" > Все </
                    label>
            </div>
            <div class="col-4 justify-c form-check">
                <input class="form-check-input me-1" type="radio" name="
                    OwnerSelect" value="opened" id="RadioBtnOpened" onclick="
                    selChange(event)" ${selOption === 'opened' ? 'checked' :
                    ''}/>
                <label class="form-check-label" for="RadioBtnOpened" >
                    Открытые </label>
            </div>
            <div class="col-4 justify-c form-check">
                <input class="form-check-input me-1" type="radio" name="
                    OwnerSelect" value="closed" id="RadioBtnClosed" onclick="
                    selChange(event)" ${selOption === 'closed' ? 'checked' :
                    ''}/>
                <label class="form-check-label" for="RadioBtnClosed" >
                    Закрытые </label>
            </div>
        `;

        this.loggedSelectingOptions = `
            <div class="col-xl-2 col-md-4 col-sm-4 justify-c form-check">
                <input class="form-check-input me-1" type="radio" name="
                    OwnerSelect" value="All" id="RadioBtnAll" checked
                    onclick="selChange(event)" ${selOption === 'All' ? '
                    checked' : ''}/>
                <label class="form-check-label" for="RadioBtnAll" > Все </
                    label>
            </div>
            <div class="col-xl-2 col-md-4 col-sm-4 justify-c form-check">
                <input class="form-check-input me-1" type="radio" name="
                    OwnerSelect" value="opened" id="RadioBtnOpened" onclick="
                    selChange(event)" ${selOption === 'opened' ? 'checked' :
                    ''}/>
                <label class="form-check-label" for="RadioBtnOpened" >
                    Открытые </label>
            </div>
            <div class="col-xl-2 col-md-4 col-sm-4 justify-c form-check">
                <input class="form-check-input me-1" type="radio" name="
                    OwnerSelect" value="closed" id="RadioBtnClosed" onclick="
                    selChange(event)" ${selOption === 'closed' ? 'checked' :
                    ''}/>
                <label class="form-check-label" for="RadioBtnClosed" >
                    Закрытые </label>
            </div>
            <div class="col-xl-3 col-md-4 col-sm-6 justify-c form-check">
                <input class="form-check-input me-1" type="radio" name="
                    OwnerSelect" value="My" id="RadioBtnMy" onclick="
                    selChange(event)" ${selOption === 'My' ? 'checked' : ''}/
```

```
                    >
34                  <label class="form-check-label" for="RadioBtnMy" >
                        Созданныемной  </label>
35              </div>
36              <div class="col-xl-3 col-md-4 col-sm-6 justify-c form-check">
37                  <input class="form-check-input me-1" type="radio" name="
                        OwnerSelect" value="notMy" id="RadioBtnNM" onclick="
                        selChange(event)" ${selOption === 'notMy' ? 'checked' : '
                        '}/>
38                  <label class="form-check-label" for="RadioBtnNM" >
                        Созданныенемной   </label>
39              </div>
40          `;
41      }
42
43      async selector(){
44          let logged = localStorage.accessToken;
45          let selectorPlace = document.querySelector('#capsuleSelector');
46          if(logged){
47              selectorPlace.innerHTML = this.loggedSelectingOptions;
48          }
49          else{
50              selectorPlace.innerHTML = this.defaultSelectingOptions;
51          }
52      }
53      async selChange(event){
54          sessionStorage.selOption = event.target.value;
55          //console.log(sessionStorage.selOption);
56      }
57  }
```

Класс **SelectorC**
Метод **selector**, в зависимоти от того, выполнил пользователь вход в систему или нет, выводит на страницу один из двух варинтов HTML кода выборки капсул
Метод **selChange** заменяет данные о нужном типе капсул хранящиеся в sessionStorage

```
class BodyCapsuleC{

    constructor(){

    }

    async exportCapsules(){
        let capsulesPlace = document.querySelector('#capsules');

        const response = await fetch('http://localhost:3000/664/capsules', {
            method: "GET"})

        const capsules = await response.json();

        //console.log(capsules)

        capsulesPlace.innerHTML = ``;

        for (const capsule of capsules){
            //console.log(capsule);
            if(this.checkAccessToCapsule(capsule) && this.checkSelOptions(
                capsule)){
                capsulesPlace.innerHTML += this.getCapsule(capsule)
            }
        }

    }
    checkAccessToCapsule({userId, access}){
        //console.log();
        const lStorageId = (JSON.parse(localStorage.user))['id'];
        if( (access === 'my' && userId === lStorageId) || (access === 'FFA'))
            {
            return true;
        }
        return false;
    }
    checkSelOptions({userId, openDate}){
        const lStorageId = (JSON.parse(localStorage.user))['id'];
        const selOption = sessionStorage.selOption;
        if((selOption === 'My' && lStorageId === userId) || (selOption === '
            notMy' && lStorageId != userId) ){
            return true;
        }
        if(selOption === 'All'){
            return true;
        }
        const status = (new Date(openDate).getTime() > (new Date(new Date().
            toISOString().slice(0, 16))).getTime())
        //console.log((selOption === 'opened' && !status), (selOption === '
            closed' && status));
        if( (selOption === 'opened' && !status) || (selOption === 'closed' &&
             status)){
            return true;
        }
        return false;
    }
```

```
50      getCapsule({id, userId, name, description, openDate, userName}){
51          let status = (new Date(openDate).getTime() > (new Date(new Date().
                toISOString().slice(0, 16))).getTime())
52          //console.log(sessionStorage.selOption, sessionStorage.selOption ===
                'opened');
53          if(status){
54              return (`
55              <div class="col-xl-3 col-md-4 col-sm-6 mb-3" data-ownerUser-id="$
                    {userId}">
56                  <div class="card capsule-param">
57                      <div class="card-header">
58                          <img src="../img/pattern.jpg" alt="" class="
                                card-img-top">
59                      </div>
60                      <div class="card-body">
61                          <h5 class="card-title">${name}</h5>
62                          <p class="card-text mb-2">${description}</p>
63                      </div>
64                      <div class="card-footer justify-s-b">
65                          <p>${openDate}</p>
66                          <p>@${userName}</p>
67                      </div>
68                  </div>
69              </div>
70              `);
71          }
72          if(!status){
73              return (`
74                  <div class="col-xl-3 col-md-4 col-sm-6 mb-3">
75                      <div class="card capsule-param">
76                          <div class="card-header">
77                              <img src="../img/pattern.jpg" alt="..." class="
                                    card-img-top">
78                          </div>
79                          <div class="card-body">
80                              <h5 class="card-title">${name}</h5>
81                              <p class="card-text mb-2">${description}</p>
82                              <form onsubmit="decapsulate(event)" class="
                                    justify-f-e">
83                                  <button type="submit" class="btn btn-my-card
                                        btn-block btn-sm" value="${id}">Открыть</
                                        button>
84                              </form>
85                          </div>
86                          <div class="card-footer justify-s-b">
87                              <p>${openDate}</p>
88                              <p>@${userName}</p>
89                          </div>
90                      </div>
91                  </div>
92              `);
93              //<p class="text-end mb-0 mt-0"><a href="#" class="btn
                    btn-my-card btn-sm" data-bs-toggle="modal" data-bs-target="#
                    openCapsuleFormModalId">Открыть</a></p>
94          }
95          return ``;
96      }
97  }
```

Класс **BodyCapsuleC**
Метод **exportCapsules** запрашивает данные о всех капсулах с JSON-server и выводит их на страницу, предварительно капсулы проверяются двумя методами: **checkSelOptions** и **checkAccessToCapsule**, на соответствие параметрам выборки и наличия доступа у текущего пользователя
Метод **getCapsule** обертывает данные капсулы в один из двух вариантов HTML кода, в зависимости от того, открылась капсула или нет

```
class ProfileC{

    constructor(){

    }

    loadMyProfileInfo(flag){
        if(flag === 0){
            const user = JSON.parse(localStorage.user);
            let profilePlace = document.querySelector('#profileCard');
            profilePlace.innerHTML = this.getProfile(user);
        }
        if(flag === 1){
            const user = JSON.parse(localStorage.user);
            let profilePlace = document.querySelector('#profileCard');
            profilePlace.innerHTML = this.getProfileForm(user);
        }
        if(flag === 3){

        }
    }

    getProfile({email='', firstName='', userName='', capCount=0, lastUpdate=0
        }){
        return (`
        <div class="row">
            <div class="col-xl-2 col-md-4 col-sm-4">
                <img src="../img/profile.jpg" class="img-fluid rounded-start"
                    alt="...">
            </div>
            <div class="col-xl-10 col-md-8 col-sm-8">
                <div class="card-body">
                    <h5 class="card-title me-2 mt-2 text-end">${userName}</h5
                        >
                    <div class="card-text text-start">
                    <p class="fw-bold">Имя: <span class="fw-normal" > ${
                        firstName} </span></p>
                    <p class="fw-bold">Email: <span class="fw-normal" > ${
                        email} </span></p>
                    <p class="fw-bold">Колво- созданныхкапсул : <span class=
                        "fw-normal" > ${capCount} </span></p>
                    </div>
                </div>
                <div class="card-footer">
                    <div class="justify-s-b">
                        <p class="card-text text-muted">Последнее
                            обновление: <span class="fw-normal" > ${new Date(
                            lastUpdate).toISOString().slice(0, 16)} </span>
                            </p>
                        <button type="button" class="btn btn-sm btn-my-main"
                            onclick="loadProfilePage(event, 1)">Изменить
                            данные</button>
                    </div>
                </div>
            </div>
        </div>
        `);
```

```
47      }
48
49      getProfileForm({id, email, firstName, userName, capCount}){
50          return (`
51          <div class="row">
52            <div class="col-xl-2 col-md-4 col-sm-4">
53                <img src="../img/profile.jpg" class="img-fluid rounded-start"
                      alt="...">
54            </div>
55            <div class="col-xl-10 col-md-8 col-sm-8">
56                  <form onsubmit="changeProfileInfo(event)" data-userId="${id}"
                        data-useremail="${email}">
57                      <div class="card-body">
58                          <div class="form-outline mb-4">
59                              <label class="form-label" for="LogInFormPassword"
                                    >Имя пользователя:</label>
60                              <input type="text" id="LogInFormPassword" class="
                                    form-control" name="userName" value="${
                                    userName}"/>
61                          </div>
62                          <div class="form-outline mb-4">
63                              <label class="form-label" for="LogInFormPassword"
                                    >Имя:</label>
64                              <input type="text" id="LogInFormPassword" class="
                                    form-control" name="firstName" value="${
                                    firstName}"/>
65                          </div>
66                              <p class="fw-bold">Email: <span class="fw-normal"
                                     > ${email} </span></p>
67                              <p class="fw-bold">Колво- созданныхкапсул :
                                    <span class="fw-normal" > ${capCount} </span>
                                    </p>
68                          </div>
69                          <button type="submut" class="btn btn-sm btn-my-main
                                mb-3">Сохранить изменения</button>
70                      </div>
71                  </form>
72            </div>
73          </div>
74          `);
75      }
76
77      async changeProfileInfo(event){
78          let fields = Array.from(event.target.querySelectorAll('input'));
79          const id = event.target.dataset.userid;
80          //console.log(event.target, id)
81          const data = JSON.parse(localStorage.user);
82          for(const field of fields){
83              data[field.name] = field.value;
84          }
85          data['lastUpdate'] = (new Date(new Date())).getTime();
86          data['password']=localStorage.accessToken;
87
88          localStorage.user = JSON.stringify(data)
89
90          await this.reloadUser();
91      }
92
```

```
93      async reloadUser(){
94          const data = JSON.parse(localStorage.user);
95          const id = data['id'];
96          const url = `http://localhost:3000/users/${id}`
97          console.log(url);
98          const response = await fetch(url, {
99              method: "PUT",
100             body: JSON.stringify(data),
101             headers: {
102                 'Content-Type': 'application/json',
103                 'Authorization': `Bearer ${localStorage.accessToken}`
104             }
105         })
106     }
107 }
```

Класс **ProfileC**
Метод **loadMyProfileInfo** в зависимости от параметра, с которым он был вызван осуществляет загрузку профиля пользователя определенным образом.
Метод **getProfile** обертывает данные о пользователе в HTML код
Метод **getProfileForm** обертывает данные о пользователе в HTML код с форсмой, предполагающей частичное изменение этих данных
Метод **changeProfileInfo** принимает данные о пользователе из формы, после их изменения, и записывает их в localStorage, после чего вызывает метод **reloadUser** для отправки обновленных данных на сервер

```
1  mainHeader = new HeaderC();
2  mainCapsulePlace = new BodyCapsuleC();
3  mainSelector = new SelectorC();
4  profileC = new ProfileC();
5
6  async function loadMainPage(){
7      mainHeader.worker();
8      mainSelector.selector();
9      mainCapsulePlace.exportCapsules();
10 }
11
12 function loadProfilePage(event, flag){
13     profileC.loadMyProfileInfo(flag);
14 }
15
16 async function changeProfileInfo(event){
17     event.preventDefault();
18     await profileC.changeProfileInfo(event);
19     //location.reload();
20 }
21
22 async function selChange(event){
23     //event.preventDefault();
24     await mainSelector.selChange(event);
25     mainCapsulePlace.exportCapsules();
26 }
```

Создание экземпляров классов и вызов методов. Функции реагируют на события в HTML коде.

3. Вывод

В ходе выполнения лабораторной работы, я релизовал скрипты на языке JS обеспечивающие работоспособность сайта. Я не использовал внешнее API, т.к. для моей задачи такого не обнаружил.