Министерство науки и высшего образования Российской Федерации
Севастопольский государственный университет
Кафедра ИС

Отчет
по лабораторной работе №2
«Исследование возможностей программирования на стороне клиента. Основы языка JavaScript»
по дисциплине
«ВЕБ-ТЕХНОЛОГИИ»

Выполнил студент группы ИС/б-17-2-о
Горбенко К. Н.
Проверил
Дрозин А.Ю.

Севастополь
2019

1 ЦЕЛЬ РАБОТЫ

Изучить основы языка JavaScript и объектной модели браузера, приобрести практические навыки проверки HTML-форм с использованием JavaScript.

2 ЗАДАНИЕ НА РАБОТУ

1. Модифицировать страницу «Фотоальбом» (использовать HTMLстраницы, разработанные при выполнении предыдущей лабораторной работы), реализовав вывод таблицы, содержащей фото, с использованием операторов циклов. Значения имен файлов фото и подписей к фото предварительно разместить в массивах fotos и titles.

2. Модифицировать страницу «Мои интересы», реализовав вывод списков с использованием JavaScript-функции с переменным числом аргументов.

3. Добавить на страницах «Контакт» и «Тест по дисциплине «. . . »» функции проверки заполненности форм. В случае если какое-либо из полей формы осталось незаполненным при нажатии на кнопку отправить, вывести сообщение об ошибке и установить фокус на незаполненный элемент.

4. Добавить на странице «Контакт» текстовое поле «Телефон». И для полей «Фамилия Имя Отчество» и «Телефон» добавить функции специфической проверки значений. В случае если какое-либо из полей формы заполнено не верно, при нажатии на кнопку отправить, вывести сообщение об ошибке и установить фокус на неверно заполненный элемент. Формат правильных значений полей:

- Фамилия Имя Отчество – введено три слова, разделенные одним пробелом.
- Телефон – строка может состоять только из цифр; начинаться только с последовательности «+7» или «+3»; не содержит пробелов; количество цифр в строке от 9 до 11.

5. Добавить на странице «Тест по дисциплине «. . . »» функции специфической проверки значений полей в соответствии с вариантом задания. В случае не выполнения условия сформировать сообщение об ошибке и установить фокус на неверно заполненный элемент ввода.

6. Необходимо выполнить проверку разработанных JavaScript файлов с использованием сервиса jshint.

Вариант № 3: 3 вопрос, количество слов в ответе не более 30.

3 ХОД РАБОТЫ

3.1 Страница «Фотоальбом»

Для выведения фотографий средствами языка JavaScript была написана следующая программа:

```
window.onload = () => {
    let photoWrapper = document.getElementsByClassName('photo-wrapper').item(0)
        ;
    photos.forEach(photo => {
        let img = document.createElement('img');
        img.setAttribute('src', photo);
        img.setAttribute('alt', 'photoalbum photo');
        photoWrapper.appendChild(img);
    });
};
const photos = [
    'picachu.jpg',
    'images.jpeg',
    'lion-king.jpg',
    'road.jpeg',
    'car.png',
    'bridge.jpg',
    'green.jpeg',
    'statue.jpg',
    'cock.jpg',
    'city.jpg',
    'fire.jpg',
    'rose.jpeg',
    'photo.jpg',
    'castrle.jpg',
    'shimpanze.jpg',
].map(image => image = `../images/${image}`);
```

В данной программе пути к фото хранятся в массиве `photos`. При загрузке страницы происходит вызов функции, которая в цикле добавляет фото в документ.

3.2 Страница «Мои интересы»

На данной странице функция `loadInterests` с переменным числом аргументов используется для выведения списков:

```
window.onload = () => {
    let links = document.getElementsByClassName('contents');
    const interestsKeys = Object.keys(interests);
```

```
4     loadInterests(interestsKeys[0], interestsKeys[1], interestsKeys[2],
          interestsKeys[3]);
5 };
6
7 const loadInterests = (...interestsIds) => {
8     const interestsList = document.getElementById('interests');
9     interestsIds.forEach(interestId => {
10        const content = interests[interestId];
11        let interestWrapper = document.createElement('div');
12        interestWrapper.setAttribute('id', interestId);
13        interestWrapper.appendChild(createElement('h2', content.header));
14        interestWrapper.appendChild(createElement('p', content.values[0]));
15        interestWrapper.appendChild(createElement('p', content.values[1]));
16        interestsList.appendChild(interestWrapper);
17    });
18 };
19
20 const createElement = (tagName, content) => {
21    let element = document.createElement(tagName);
22    element.innerHTML = content;
23    return element;
24 };
25
26 const interestContent = `Lorem ipsum dolor sit amet, consecteturadipiscing elit
      . Sed gravida
27    dolor et ultricies placerat. Pellentesque in lectus at augue rutrum
28    volutpat vel sed tellus. Donec feugiat ac nibh et pharetra. Curabitur
29    fringilla lacinia nisl tempus mollis. Nam semper augue et ante ornare
30    semper. Etiam nunc velit, pharetra eget fermentum tincidunt, lacinia
31    vitae sapien. Etiam quis elit tincidunt, consequat urna eget, vulputate
32    libero. Donec dignissim ornare porttitor. Donec id augue turpis. Morbi
          massa
33    sem, dapibus at semper in, mollis a sem. Ut eu lacus ut leo aliquam laoreet
          .
34    Integer sed ex vel erat dictum finibus sit amet id mi. Duis lorem neque,
35    faucibus ut laoreet eu, sodales in orci. Nam mattis mauris sit amet lectus
36    condimentum, in maximus massa ultrices. Fusce pharetra accumsan dui sed
          bibendum.`;
37
38 const interests = {
39    'Hobbies': {
40        header: 'My hobbies',
41        values: [
42            'My hobbies are supposed to be here',
43            interestContent
44        ]
45    },
46    'Books': {
```

```
47          header: 'My books',
48          values: [
49              'My books are supposed to be here',
50              interestContent
51          ]
52      },
53      'Music': {
54          header: 'My music',
55          values: [
56              'My music is supposed to be here',
57              interestContent
58          ]
59      },
60      'Films': {
61          header: 'My films',
62          values: [
63              'My films are supposed to be here',
64              interestContent
65          ]
66      }
67  };
```

Списки хранятся в объекте `interests`, выводятся с помощью функции с одним rest-параметром, в которую передаются элементы массива.

3.3 Страницы «Контакт» и «Тест»

На данных страницах добавлено поле «Номер телефона». На все поля добавлена валидация с выводом сообщений и установкой фокуса на неправильно введенных полях. Тексты программ представлены далее. Текст файла `contact.js`:

```
1  class FieldFilledValidator {
2      constructor() {
3          this.errorMessage = 'This field should be filled';
4          this.validate = (value) => {
5              return value !== '';
6          };
7      }
8  }
9  class PhoneNumberValidator {
10     constructor() {
11         this.errorMessage = 'Entered number is in incorrect format';
12         this.validate = (value) => {
13             const pattern = /^[+][7|3]\d{8,10}$/g;
14             return pattern.test(value);
15         };
16     }
```

```
17 }
18 class NameValidator {
19     constructor() {
20         this.errorMessage = 'Enter three space-separated words';
21         this.validate = (value) => {
22             const pattern = /[A-Za-z]+ [A-Za-z]+ [A-Za-z]+/;
23             return pattern.test(value);
24         };
25     }
26 }
27 class FormComponent {
28     constructor(componentId, validators) {
29         this.validate = () => {
30             this.validators.forEach(validator => {
31                 const message = validator.errorMessage;
32                 if (validator.validate(this.value)) {
33                     this.errorMessages.splice(this.errorMessages.indexOf(
                           message), 1);
34                 }
35                 else {
36                     if (!this.errorMessages.includes(message)) {
37                         this.errorMessages.push(message);
38                     }
39                 }
40             });
41         };
42         this.componentId = componentId;
43         this.validators = validators;
44         this.errorMessages = [];
45     }
46     ;
47 }
48 const fields = [
49     new FormComponent('full-name-input', [
50         new NameValidator(),
51     ]),
52     new FormComponent('email-input', [
53         new FieldFilledValidator(),
54     ]),
55     new FormComponent('phone-input', [
56         new PhoneNumberValidator(),
57     ]),
58     new FormComponent('message-input', [
59         new FieldFilledValidator(),
60     ]),
61 ];
62 window.onload = () => {
63     const form = document.forms.item(0);
```

```
64      form.addEventListener('submit', validateForm);
65      fields.forEach(field => {
66          const fieldElement = document.getElementById(field.componentId);
67          fieldElement.addEventListener('change', () => validateField(field));
68      });
69  };
70  const validateField = (field) => {
71      const errorsMessages = document.getElementById(`${field.componentId}-errors
            `);
72      if (errorsMessages !== null) {
73          errorsMessages.remove();
74      }
75      field.value = document.getElementById(field.componentId).value;
76      field.validate();
77      if (field.errorMessages.length > 0) {
78          showMessages(field);
79      }
80  };
81  const validateForm = () => {
82      let firstError;
83      fields.forEach(field => {
84          validateField(field);
85          if (!firstError && field.errorMessages.length > 0) {
86              firstError = field;
87          }
88      });
89      if (firstError) {
90          document.getElementById(firstError.componentId).focus();
91      }
92      if (fields.some(field => field.errorMessages.length > 0)) {
93          event.preventDefault();
94      }
95  };
96  const showMessages = (field) => {
97      const targetElement = document.getElementById(field.componentId);
98      const parentElement = targetElement.parentElement;
99      let contentWrapper = document.createElement('div');
100     contentWrapper.setAttribute('id', `${field.componentId}-errors`);
101     contentWrapper.setAttribute('class', 'error-messages');
102     field.errorMessages.forEach(message => {
103         let messageListItem = document.createElement('li');
104         messageListItem.innerHTML = message;
105         contentWrapper.appendChild(messageListItem);
106     });
107     parentElement.insertBefore(contentWrapper, targetElement);
108 };
```

Текст файла `test.js`:

```
class FieldFilledValidator {
    constructor() {
        this.errorMessage = 'This field should be filled';
        this.validate = (value) => {
            return value !== '';
        };
    }
}
class PhoneNumberValidator {
    constructor() {
        this.errorMessage = 'Entered number is in incorrect format';
        this.validate = (value) => {
            const pattern = /^[+][7|3]\d{8,10}$/g;
            return pattern.test(value);
        };
    }
}
class NameValidator {
    constructor() {
        this.errorMessage = 'Enter three space-separated words';
        this.validate = (value) => {
            const pattern = /[A-Za-z]+ [A-Za-z]+ [A-Za-z]+/;
            return pattern.test(value);
        };
    }
}
class FormComponent {
    constructor(componentId, validators) {
        this.validate = () => {
            this.validators.forEach(validator => {
                const message = validator.errorMessage;
                if (validator.validate(this.value)) {
                    this.errorMessages.splice(this.errorMessages.indexOf(
                        message), 1);
                }
                else {
                    if (!this.errorMessages.includes(message)) {
                        this.errorMessages.push(message);
                    }
                }
            });
        };
        this.componentId = componentId;
        this.validators = validators;
        this.errorMessages = [];
    }
    ;
}
```

```
48 const fields = [
49     new FormComponent('full-name-input', [
50         new NameValidator(),
51     ]),
52     new FormComponent('email-input', [
53         new FieldFilledValidator(),
54     ]),
55     new FormComponent('phone-input', [
56         new PhoneNumberValidator(),
57     ]),
58     new FormComponent('message-input', [
59         new FieldFilledValidator(),
60     ]),
61 ];
62 window.onload = () => {
63     const form = document.forms.item(0);
64     form.addEventListener('submit', validateForm);
65     fields.forEach(field => {
66         const fieldElement = document.getElementById(field.componentId);
67         fieldElement.addEventListener('change', () => validateField(field));
68     });
69 };
70 const validateField = (field) => {
71     const errorsMessages = document.getElementById(`${field.componentId}-errors
        `);
72     if (errorsMessages !== null) {
73         errorsMessages.remove();
74     }
75     field.value = document.getElementById(field.componentId).value;
76     field.validate();
77     if (field.errorMessages.length > 0) {
78         showMessages(field);
79     }
80 };
81 const validateForm = () => {
82     let firstError;
83     fields.forEach(field => {
84         validateField(field);
85         if (!firstError && field.errorMessages.length > 0) {
86             firstError = field;
87         }
88     });
89     if (firstError) {
90         document.getElementById(firstError.componentId).focus();
91     }
92     if (fields.some(field => field.errorMessages.length > 0)) {
93         event.preventDefault();
94     }
```

```
95 };
96 const showMessages = (field) => {
97     const targetElement = document.getElementById(field.componentId);
98     const parentElement = targetElement.parentElement;
99     let contentWrapper = document.createElement('div');
100    contentWrapper.setAttribute('id', `${field.componentId}-errors`);
101    contentWrapper.setAttribute('class', 'error-messages');
102    field.errorMessages.forEach(message => {
103        let messageListItem = document.createElement('li');
104        messageListItem.innerHTML = message;
105        contentWrapper.appendChild(messageListItem);
106    });
107    parentElement.insertBefore(contentWrapper, targetElement);
108 };
```

На рисунках 1 - 2 изображены сообщения при нажатии на кнопку «Submit» при пустой форме.

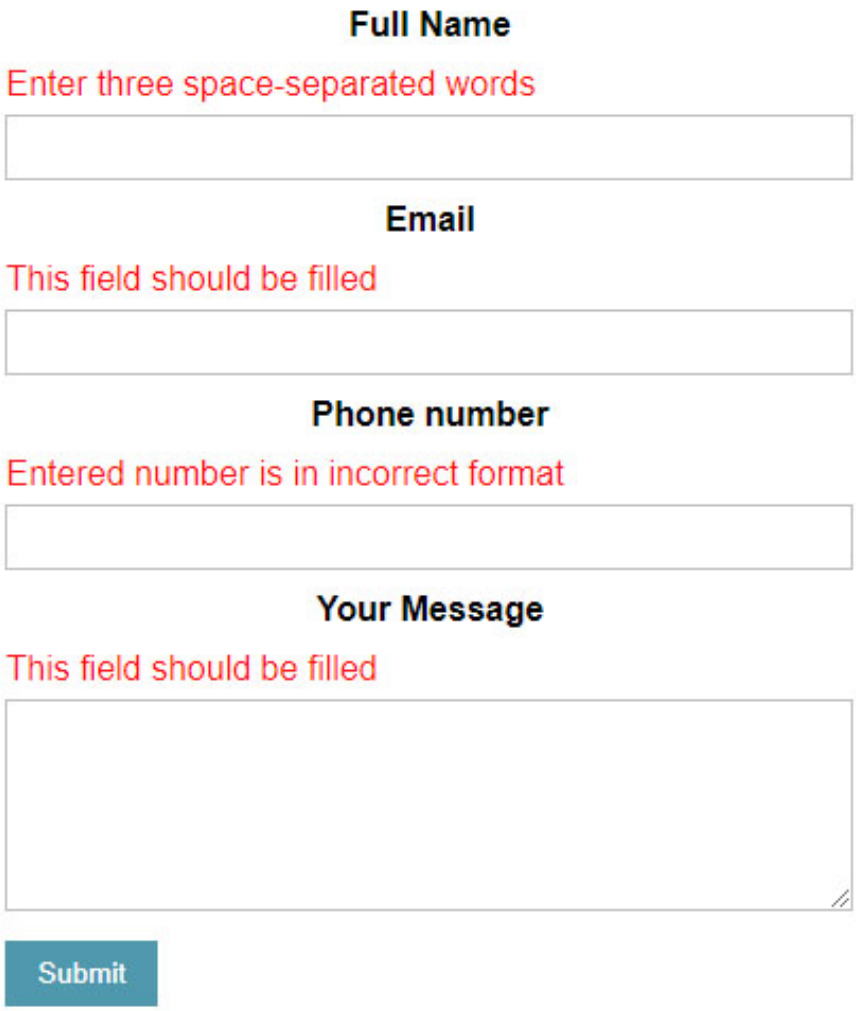

Рисунок 1 – Тексты сообщений при ошибке валидации на странице «Контакт»

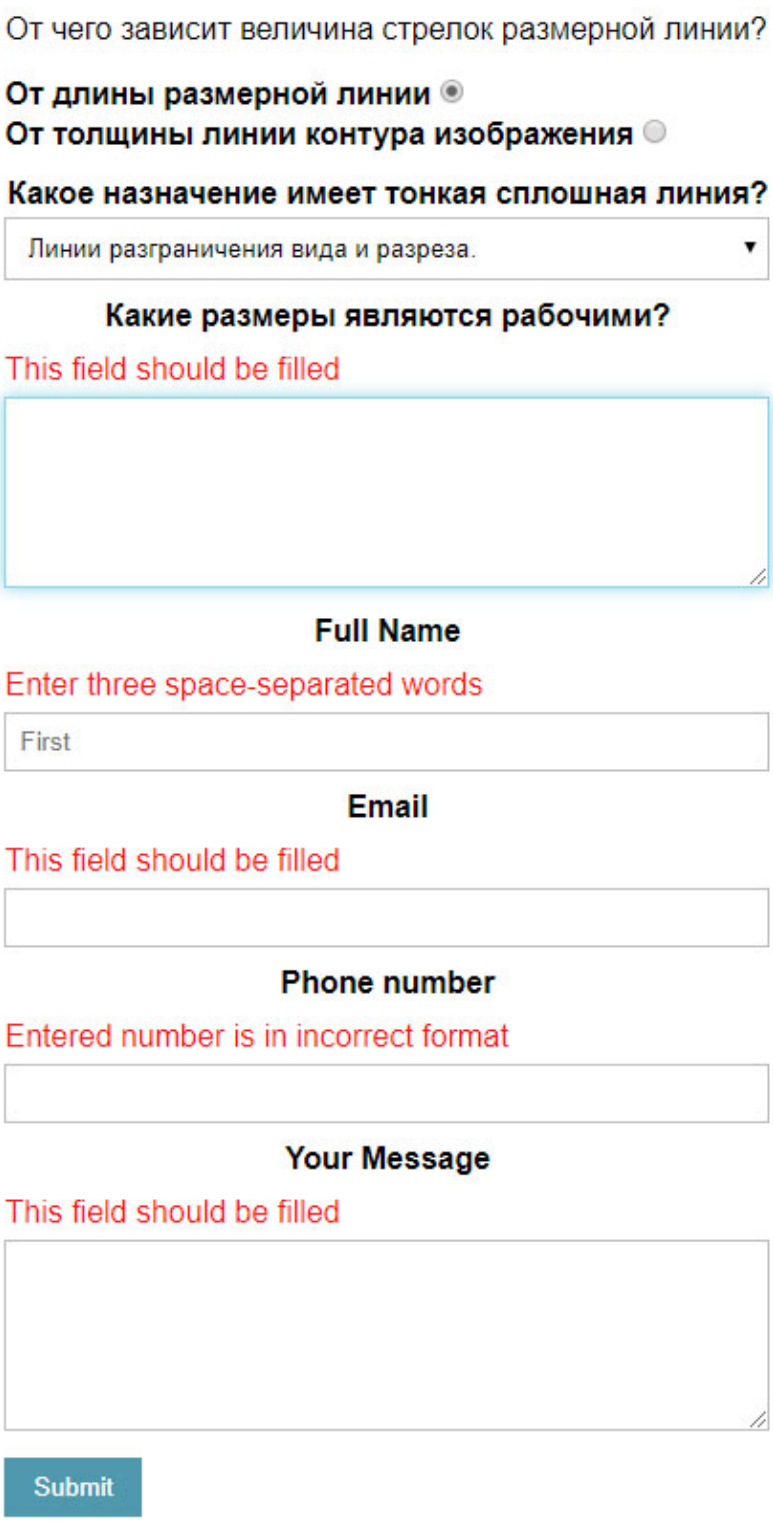

Рисунок 2 – Тексты сообщений при ошибке валидации на странице «Тест»

В данных программаю создаются массивы элементов формы, в которых к элементам привязываются классы валидации, содержащие саму функцию валидации и сообщение об ошибке валидации. При изменении любого из полей, на нем

срабатывает функция валидации.

В формы добавлены поля «Номер телефона». Поле «Имя» валидируется как три слова, разделенные пробелом. Текстовое поле ответа на третий вопрос теста требует менее 30 слов.

ВЫВОДЫ

В ходе лабораторной работы были изучены способы получения и вставки элементов HTML-документа. Для работы с HTML используется механизм DOM, в котором каждый элемент документа представляет из себя объект языка JavaScript.

Валидация проводилась с помощью функций, которые реагировали на событие «change». Привязка к полям документа осуществлялась по их id.