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

Отчет
по лабораторной работе №4
«Исследование возможнойстей библиотеки JQuery»
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 возможность программирования на клиентской стороне с использованием библиотеки jQuery. Приобрести практические навыки использования библиотеки jQuery для обработки форм, модификации содержимого HTML-страницы, создания эффектов анимации.

2 ЗАДАНИЕ НА РАБОТУ

1. Модифицировать JavaScript, разработанные при выполнении предыдущих лабораторных работ, используя везде, где возможно библиотеку jQuery для выполнения поставленных задач.

2. Модифицировать страницу «Фотоальбом», добавив возможность просмотра увеличенных версий изображений. При щелчке по стрелкам должна происходить анимированная смена изображений.

3. Реализовать возможность отображения всплывающего блока Popover, который показывается рядом с элементом, если на него наводится указатель мыши и который исчезает через M секунд после того, как указатель мыши покидает элемент.

4. Использовать разработанный код для отображения подсказок уточняющих формат ввода пользователю в формах.

5. Реализовать возможность отображения всплывающего модального окна, которое показывается в центре экрана. При этом задний фон вокруг модального окна должен размываться.

3 ХОД РАБОТЫ

3.1 Использование JQuery для работы с DOM

Используем JQuery где возможно для работы с DOM. Выведение истории:

```
1 const globalHistory = getGlobalHistory();
2 for (let key in globalHistory) {
3     $(‘.global-history .${key}-page‘).append(globalHistory[key]);
4 }
5 const sessionHistory = getSessionHistory();
6 for (let key in sessionHistory) {
7     $(‘.session-history .${key}-page‘).append(sessionHistory[key]);
8 }
```

В данном случае используются функции append для вставки контента в элемент td таблицы.

Валидация форм:

```
const validateForm = (event, fields) => {
    let firstError : FormComponent;
    fields.forEach(field => {
        validateField(field);
        if (!firstError && field.errorMessages.length > 0) {
            firstError = field;
        }
    });
    if (firstError) {
        $(`#${firstError.componentId}`).trigger('focus');
    }
    if (fields.some(field => field.errorMessages.length > 0)) {
        event.preventDefault();
    }
};

const validateField = (field: FormComponent) => {
    const presentErrorMessages = $(`#${field.componentId}-errors`);
    if (presentErrorMessages.length > 0) {
        presentErrorMessages.remove();
        $(`#${field.componentId}`).removeClass('validation-failed');
    }
    field.validate();
    if (field.errorMessages.length > 0) {
        showMessages(field);
    }
};

const showMessages = (field: FormComponent) => {
    const targetElement = $(`#${field.componentId}`).addClass('validation-
        failed');
    let contentWrapper = $('<div></div>').attr('id', `${field.componentId}-
        errors`).addClass('error-messages');
    field.errorMessages.forEach(message => contentWrapper.append($(`<li>${
        message}</li>`)));
    contentWrapper.insertBefore(targetElement);
};
```

JQuery использовался для всевозможный манипуляций DOM: отображения/сокрытия элементов, выборки объектов из DOM, добавления и удаления классов и атрибутов.

3.2 Реализация просмотра фотографий

Реализуем просмотр фотографий:

```
1 import * as $ from 'jquery';
2
3 let body = $('body');
4 let lightbox = $('#lightbox');
5 let lightboxImage = $('#lightbox img');
6 let lightboxPrevArrow = $('.lightbox-arrow-left');
7 let lightboxNextArrow = $('.lightbox-arrow-right');
8
9 let currentPhoto;
10 let photos;
11
12 export default (lightboxPhotosWrapper: Element, photosPaths: string[]) => {
13     photos = photosPaths;
14     lightboxPhotosWrapper.addEventListener('click', (event) => expandLightbox(
           event));
15
16     window.onclick = (event) => {
17         if (event.target === document.getElementById('lightbox')) {
18             collapseLightbox();
19         }
20     }
21 }
22
23 const expandLightbox = (event) => {
24     currentPhoto = event.target.getAttribute('src');
25     lightbox.toggle();
26     lightboxImage.attr('src', currentPhoto);
27     body.addClass('no-scroll');
28     lightboxPrevArrow.on('click', prevButtonClickHandler);
29     lightboxNextArrow.on('click', nextButtonClickHandler);
30 };
31
32 const collapseLightbox = () => {
33     lightbox.toggle();
34     body.removeClass('no-scroll');
35 };
36
37 const prevButtonClickHandler = () => {
38     const currentIndex = photos.indexOf(currentPhoto);
39     if (currentIndex === 0 || currentIndex === -1) return;
40     currentPhoto = photos[currentIndex - 1];
41     lightboxImage.attr('src', currentPhoto);
42 };
43
44 const nextButtonClickHandler = () => {
```

```
45      const currentIndex = photos.indexOf(currentPhoto);
46      if (currentIndex === photos.length - 1 || currentIndex === -1) return;
47      currentPhoto = photos[currentIndex + 1];
48      lightboxImage.attr('src', currentPhoto);
49  };
```

В данном файле реализуется включение отображения окна просмотра фотографий. На кнопки назад/далее навешиваются обработчики событий с проверкой выхода за границы массива.

Получившийся элемент имеет следующий вид (рисунок **??**):

Рисунок 1 – Вид реализации переключения фотографий

3.3 Реализация всплывающего окна

Для реализации всплывающего окна была использована сторонняя библиотека `tooltipster`. Ее использование:

```
1  import 'tooltipster';
2
3  export const setFieldsForValidation = (fields: FormComponent[], form:
       HTMLFormElement) => {
4       $(form).on('submit', (event) => validateForm(event, fields));
5       fields.forEach(formComponent => {
6           const field =  $(`#${formComponent.componentId}`);
7           field.on('blur', () => validateField(formComponent));
8           field.tooltipster({ content: formComponent.correctValueDescription,
                theme: 'tooltipster-light', side: 'right' });
9       });
10 };
```

Реализованный компонент имеет следующий вид (рисунок **??**):

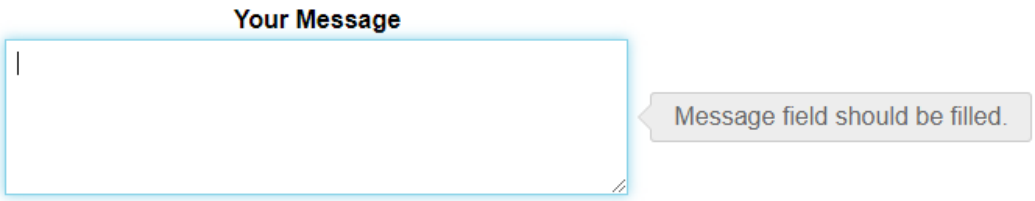

Рисунок 2 – Вид реализации всплывающего окна

4 РЕАЛИЗАЦИЯ МОДАЛЬНОГО ОКНА

Реализуем модальное окно:

```
1 import * as $ from 'jquery';
2
3 export interface IModalOptions {
4     message?: string;
5     confirmButtonText?: string;
6     cancelButtonText?: string;
7     confirmAction?: () => void;
8     cancelAction?: () => void;
9 }
10
11 const body = $('body');
12 const getModal = () => $('.modal');
13 const message = 'Do you really want to do this?';
14 const confirmButtonText = 'Yes';
15 const cancelButtonText = 'No';
16
17 export default (options: IModalOptions) => {
18     $('.modal-message').text(options.message || message);
19     $('.modal-confirm-button').text(options.confirmButtonText ||
          confirmButtonText).off('click').on('click', getAction(options.
          confirmAction));
20     $('.modal-cancel-button').text(options.cancelButtonText || cancelButtonText
          ).off('click').on('click', getAction(options.cancelAction));
21     expandModal();
22 }
23
24 const expandModal = () => {
25     body.addClass('no-scroll');
26     getModal().toggle();
27 };
28
29 const collapseModal = () => {
30     body.removeClass('no-scroll');
31     getModal().toggle();
32 };
```

```
33
34 const getAction = (action: () => void) => {
35     if (action) {
36         return () => { collapseModal(); action(); }
37     }
38     return () => collapseModal();
39 };
```

Интерфейс модуля модального окна включает объект с настройками: `callback`-функции, вызываемые при нажатии на подтверждение или отмену, тексты кнопок, сообщение.

Элемент имеет следующий вид (рисунок **??**):

Рисунок 3 – Вид реализации модального окна

ВЫВОДЫ

В ходе лабораторной работы была исследована библиотека JQuery. Лна была использована для упрощения работы с DOM.

Преимущества JQuery следующие: упрощенная работа с выборкой элементов из DOM, богатая библиотека для работы с обектами JQuery, легко трансформируемыми в объекты DOM, возможность последовательного вызова функций.