# Лабораторная работа № 8 Обработка и валидация форм с использованием JavaScript

**Цель работы:** научиться обрабатывать значения, введенные пользователем в элементы форм, применять валидацию формы на стороне клиента.

**Краткие теоретические сведения:**

Один из способов взаимодействия с пользователями представляют html-формы. Например, если нам надо получить от пользователя некоторую информацию, мы можем определить на веб-странице форму, которая будет содержать текстовые поля для ввода информации и кнопку для отправки. После ввода данных пользователем мы можем обработать введенную информацию.

В JavaScript форма представлена объектом **HtmlFormElement**. И после создания формы мы можем к ней обратиться различными способами.

Первый способ заключается в прямом обращении по имени формы:

<form name="search">

</form>

<script>

let searchForm = document.search;

</script>

Второй способ состоит в обращении к коллекции форм документа и поиске в ней нужной формы:

let searchForm;

for (var i = 0; i < document.forms.length; i++) {

if(document.forms[i].name==="search")

searchForm = document.forms[i];

}

Форма имеет ряд свойств, из которых наиболее важными являются свойство name, а также свойство **elements**, которое содержит коллекцию элементов формы.

Среди методов формы надо отметить метод **submit()**, который отправляет данные формы на сервер, и метод **reset()**, который очищает поля формы:

**Элементы форм**

Форма может содержать различные элементы ввода html: input, textarea, button, select и т.д. Но все они имеют ряд общих свойств и методов.

Также, как и форма, элементы форм имеют свойство name, с помощью которого можно получить значение атрибута name.

Для получения и установки значения элементов **input** и **textarea** используется свойство input.value (строка) или input.checked (булево значение) для чекбоксов, textarea.value:

input.value = "Новое значение";

textarea.value = "Новый текст";

input.checked = true; // для чекбоксов и переключателей

Элемент **select** имеет 3 важных свойства:

* select.options – коллекция из подэлементов option;
* select.value – значение выбранного в данный момент option;
* select.selectedIndex – номер выбранного option.

Они дают три разных способа установить значение в select:

* найти соответствующий элемент option и установить в option.selected значение true;
* установить в select.value значение нужного option;
* установить в select.selectedIndex номер нужного option.

В отличие от большинства других элементов управления, select позволяет нам выбрать несколько вариантов одновременно, если у него стоит атрибут multiple. Эту возможность используют редко, но в этом случае для работы со значениями необходимо использовать первый способ, то есть ставить или удалять свойство selected у подэлементов option.

**Валидация формы**

HTML5 представил новую концепцию валидации HTML, названную проверкой ограничений, которая основана на:

* атрибутах элементов ввода HTML;
* псевдо-селекторах CSS;
* свойствах и методах DOM.

Для валидации формы используются следующие атрибуты элементов ввода HTML:

* **disabled**: указывает, что элемент ввода должен быть отключен;
* **max**: определяет максимальное значение элемента ввода;
* **min**: определяет минимальное значение элемента ввода;
* **pattern**: определяет шаблон значения элемента ввода;
* **required**: указывает, что для поля ввода требуется элемент;
* **type**: определяет тип элемента ввода.

Проверка ограничений CSS использует следующие псевдо-селекторы:

* **:disabled** – выбирает элементы ввода с указанным атрибутом «disabled»;
* **:invalid** – выбирает элементы ввода с недопустимыми значениями;
* **:optional** – выбирает элементы ввода без указания атрибута «required»;
* **:required** – выбирает элементы ввода с указанным атрибутом «required»
* **:valid** – выбирает элементы ввода с допустимыми значениями

Методы DOM включают функции:

* **checkValidity ()**: возвращает true, если входной элемент содержит допустимые данные;
* **setCustomValidity ()**: устанавливает свойство **validationMessage** элемента ввода.

Свойства DOM:

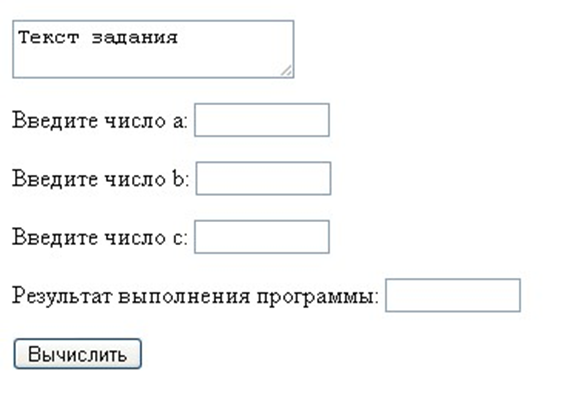
* **validity**: содержит логические свойства, связанные с достоверностью входного элемента;
* **validationMessage**: содержит сообщение, которое браузер отобразит, когда допустимость ложна;
* **willValidate**: указывает, будет ли проверен входной элемент.

В свою очередь свойство validity является объектом со следующими свойствами:

* **customError**: устанавливается в true, если настроено пользовательское сообщение о достоверности;
* **patternMismatch**: устанавливается в true, если значение элемента не соответствует его атрибуту pattern;
* **rangeOverflow**: устанавливается в true, если значение элемента больше, чем его атрибут max;
* **rangeUnderflow**: устанавливается в true, если значение элемента меньше его атрибута min;
* **stepMismatch**: устанавливается в true, если значение элемента недопустимо для его атрибута шага;
* **tooLong**: устанавливается в true, если значение элемента превышает его атрибут maxLength;
* **typeMismatch**: устанавливается в true, если значение элемента недопустимо для его атрибута типа;
* **valueMissing**: устанавливается в true, если элемент (с обязательным атрибутом) не имеет значения;
* **valid**: устанавливается в true, если значение элемента допустимо.

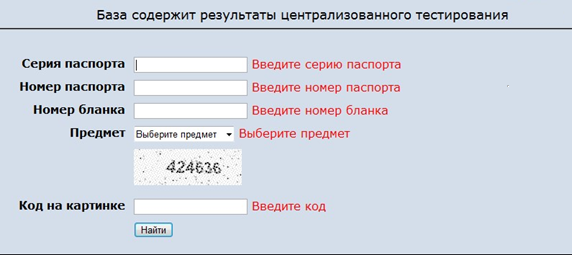
**Практическая часть:**

**Задание 1.** Необходимо создать веб-страницу с формой.

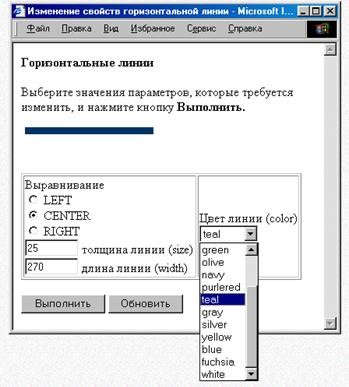


Написать скрипт, который находит среднее арифметическое введенных значений и выводит результат в поле формы при нажатии на кнопку «Вычислить». В программе предусмотреть обработку исключительной ситуации, когда пользователь не ввел значений (хотя бы одного) или введенные значения не являются числами, с выводом сообщения в модальном окне. Модальное окно реализовать самостоятельно, не использую стандартные диалоговые окна объекта window.

**Задание 2.** Необходимо реализовать форму с валидацией. Валидация осуществляется без использования атрибутов элементов ввода HTML. В поле паспорт могут вводится только две латинские буквы, в поле номер паспорта – только 7 цифр, номер бланка ответа – 7 цифр, предмет предполагает выбор языка (белорусский или русский), математика, физика. Код на картинке – 6 цифр. При не заполнении какого-либо поля должна выводится соответствующая подсказка.



**Задание 3.** Создайте файл с HTML-формой. Разработайте код JavaScript, в котором анализируются введенные пользователем данные, позволяющие изменить размер, положение на странице и цвет горизонтальной линии:



Вместо линии можно использовать блок и свойства CSS – width, height, backgroundColor. Для выравнивания блока используйте свойство margin.

**Задание 4.** В соответсвии со своим вариантом (задание 1 лабораторной работы №6), создать форму для ввода свойств объекта. Сформировать массив объектов, организовав цикл, в котором вызывается форма для ввода как модальное окно. Введенный макссив должен быть отображен на странице в виде таблицы.

**Требования по оформлению отчета:**

Отчет должен включать следующие пункты:

1. Титульный лист.
2. Цель работы.
3. Задание.
4. Листинг программного кода.
5. Результаты выполнения сценариев.
6. Выводы.

**Контрольны вопросы для защиты:**

1. Какие механизмы используются при валидации форм?
2. Какие атрибуты элементов ввода используются при валидации форм?
3. Какие псевдоселекторы CSS используются при проверке ограничений?
4. Как получить доступ к конкретной форме на веб-странице?
5. Как получить доступ к элементам формы?
6. Как получить значение, введенное пользователем в элемент ввода?
7. Как проверить состояние переключателей (radio и checkbox)?
8. Как получить значение элемента select?
9. Какие методы DOM используются при валидации форм?
10. Какие свойства DOM используются при валидации форм?