МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**WEB-ПРОГРАММИРОВАНИЕ**

ДОМАШНЯЯ КОНТРОЛЬНАЯ РАБОТА №1

Выполнил Учащийся группы ПО-455 М.Г.Верхов

Шифр 15

2023

**26 Опишите способы доступа к элементам форм, особенности проверки корректности данных, вводимых пользователем и обработки данных форм средствами JavaScript.**

Перед отправкой данных на сервер важно убедиться, что все обязательные поля формы заполнены данными в корректном формате. Это называется валидацией на стороне клиента и помогает убедиться, что данные, введённые в каждый элемент формы, соответствуют требованиям.

Валидация на стороне клиента — это первичная проверка введённых данных, которая существенно улучшает удобство взаимодействия с интерфейсом; обнаружение некорректных данных на стороне клиента позволяет пользователю немедленно их исправить. Если же проверка происходит только на сервере, процесс заполнения может быть более трудоёмким, так как требует повторения одних и тех же действий отправки данных на сервер для получения обратного ответа с сообщением о том, что нужно исправить.

Однако, не следует рассматривать валидацию на стороне клиента как достаточную меру безопасности! Любые данные, отправляемые через форму, необходимо дополнительно проверять на безопасность и на стороне сервера, поскольку валидацию на стороне клиента достаточно просто обойти и она может не остановить злоумышленников.

Зайдите на любой популярный сайт, имеющий форму регистрации. Вы заметите, что при вводе данных в неправильном формате, пользователя сразу уведомляют о наличии проблемы. Вы получите примерно такое сообщение:

* "Обязательное поле" (Вы не можете оставить поле пустым).
* "Пожалуйста, введите номер телефона в формате xxx-xxxx" (Чтобы данные считались корректными, их необходимо указать в определённом формате).
* "Пожалуйста, введите корректный email-адрес" (вы ввели данные в неправильном формате).
* "Длина пароля должна быть от 8 до 30 символов и включать одну заглавную букву, один символ, и одну цифру." (Требования к формату данных достаточно конкретные).

Это называется валидацией формы. По мере ввода, браузер и/или сервер проверяют данные, чтобы определить, соответствуют ли они требуемому формату. Валидация, выполняемая в браузере, называется валидацией на стороне клиента, а выполняемая на сервере — валидацией на стороне сервера. В этом разделе мы сосредоточимся на валидации, выполняемой на стороне клиента.

Если формат корректен, приложение позволяет отправить данные на сервер и (обычно) сохранить в базу данных; в противном случае выводится сообщение с описанием того, что нужно исправить, позволяя ввести данные снова.

Мы хотим максимально упростить заполнение веб-форм. Тогда почему мы настаиваем валидации данных? На это есть три основные причины:

Мы хотим получать правильные данные в правильном формате. Наши приложения не будут работать должным образом, если данные от пользователей хранятся в неправильном формате, некорректны сами по себе или вовсе пропущены.

* Мы хотим защитить данные пользователей. Принуждение пользователей вводить надёжные пароли облегчает защиту их аккаунтов.
* Мы хотим защитить себя. Существует множество способов, позволяющих злоумышленникам с помощью незащищённых форм навредить приложению.

Существует два типа валидации на стороне клиента, с которыми вы столкнётесь в Интернете:

* Встроенная валидация форм использует функционал валидации HTML5, который мы неоднократно обсуждали в этом модуле. HTML5-валидация обычно не требует большого количества JavaScript-кода и демонстрирует лучшую производительность, но не настолько настраиваема, как валидация с помощью JavaScript.
* JavaScript-валидация кодируется с помощью JavaScript. Она полностью настраиваема, но требует программирования всей логики (или использования библиотеки).

Если нужно управлять внешним видом встроенных сообщений об ошибке или работать с устаревшими браузерами, которые не поддерживают встроенную валидацию форм HTML, вам следует использовать JavaScript.

Большинство браузеров поддерживают Constraint Validation API, который состоит из набора свойств и методов, доступных на DOM-интерфейсах следующих элементов форм:

* HTMLButtonElement (представляет элемент <button>)
* HTMLFieldSetElement (представляет элемент <fieldset>)
* HTMLInputElement (представляет элемент <input>)
* HTMLOutputElement (представляет элемент <output>)
* HTMLSelectElement (представляет элемент <select>)
* HTMLTextAreaElement (представляет элемент <textarea>)
* Для перечисленных выше элементов Constraint Validation API делает доступными следующие свойства.
* validationMessage: Возвращает локализованное сообщение, описывающее ограничения валидации (если таковые имеются), которым не удовлетворяет определённый элемент. Если элемент не участвует в валидации (willValidate установлено в false) или значение элемента удовлетворяет установленным ограничениям (является валидным), будет возвращена пустая строка.
* validity: Возвращает объект ValidityState, который содержит несколько свойств, описывающих состояние валидности элемента. Подробное описание всех свойств доступности можно найти на странице справочника ValidityState; ниже приведён список наиболее используемых:
* patternMismatch: Возвращает true, если значение не соответствует шаблону, указанному в атрибуте pattern, и false если соответствует. Если true, элемент соответствует CSS-псевдоклассу :invalid.
* tooLong: Возвращает true, если значение длиннее максимальной длины, указанной в атрибуте maxlength, и false если оно короче или равно ей. Если true, элемент соответствует CSS-псевдоклассу :invalid.
* tooShort: Возвращает true, если значение короче минимальной длины, указанной в атрибуте minlength, и false если оно длинее или равно ей. Если true, элемент соответствует CSS-псевдоклассу :invalid.
* rangeOverflow: Возвращает true, если значение больше указанного в атрибуте max максимума, и false если меньше или равно ему. Если true, элемент соответствует CSS-псевдоклассам :invalid и :out-of-range
* rangeUnderflow: Возвращает true, если значение меньше указанного в атрибуте min, и false если больше или равно ему. Если true, элемент соответствует CSS-псевдоклассу :invalid и :out-of-range.
* typeMismatch: Возвращает true, если значение не соответствует требуемому синтаксису (когда для type задано значение email или url), и false если синтаксис корректный. Если true, элемент соответствует CSS-псевдоклассу :invalid.
* valid: Возвращает true, если элемент соответствует всем ограничениям валидации — следовательно, считается валидным, и false если не соответствует какому-то ограничению. Если true, элемент соответствует CSS-псевдоклассу :valid; иначе :invalid.
* valueMissing: Возвращает true, если у элемента есть атрибут required, но не введено значенение, иначе возвращает false. Если true, элемент соответствует CSS-псевдоклассу :invalid.
* willValidate: Возвращает true, если элемент будет участвовать в валидации при отправке формы; иначе возвращает false.

Также для перечисленных выше элементов Constraint Validation API делает доступными следующие методы.

checkValidity(): Возвращает true, если значение элемента проходит валидацию, иначе возвращает false. Если элемент не валиден, данный метод также запускает на нём событие invalid.

setCustomValidity(message): Позволяет добавить в элемент кастомное сообщение об ошибке; при этом элемент будет считаться не валидным и отобразится указанная ошибка. Это позволяет использовать JavaScript-код, чтобы представить ошибку валидации иначе, чем это предусмотрено стандартными средствами валидации HTML5. При сообщении об ошибке данное кастомное сообщение показывается пользователю.

**56 Решение задач на языке JavaScript при помощи циклов. Дан массив a = [15, 97, 2, 158, 26, 268, 88, 105, 43]. При помощи цикла while найти сумму четных элементов данного массива.**

<!-- index.html -->

<!DOCTYPE html>

<html lang='ru'>

<head>

<meta charset='utf-8'>

<title>Добро пожаловать!</title>

</head>

<body>

<script>

function test(){

let array = [15, 97, 2, 158, 26, 268, 88, 105, 43];

let result = 0;

for (let i = 1; i < array.length; i+=2) {

result += array[i];

}

console.log(result);

alert(result);

}

</script>

<input type=button value='Вычислить' onclick = test()><br>

<br>

<br>

let array = [15, 97, 2, 158, 26, 268, 88, 105, 43];<br>

let result = 0;<br>

for (let i = 1; i < array.length; i+=2) { <br>

result += array[i];<br>

}<br>

console.log(result);<br>

alert(result);<br>

</body>

</html>

**80 Решение задач обработки массивов на языке JavaScript.**

**Решите задачу. В массиве из 10 целых чисел найти наименьший элемент и поменять его местами с последним элементом.**

<!-- index.html -->

<!DOCTYPE html>

<html lang='ru'>

<head>

<meta charset='utf-8'>

<title>Добро пожаловать!</title>

</head>

<body>

<script>

function test(){

let array = [15, 97, 2, 158, 26, 268, 88, 105, 43, 33];

let emin = array[0];

let pmin = 0;

console.log(array);

for (let i = 1; i < array.length; i++) {

if(array[i] < emin){

emin = array[i];

pmin = i;

}

}

let t = array[9];

array[9] = array[pmin];

array[pmin] = t;

console.log(emin);

console.log(array);

alert(emin);

}

</script>

<input type=button value='Выполнить' onclick = test()><br>

<br>

<br>

let array = [15, 97, 2, 158, 26, 268, 88, 105, 43, 33];<br>

let result = 0;<br>

for (let i = 1; i < array.length; i+=2) { <br>

result += array[i];<br>

}<br>

console.log(result);<br>

alert(result);<br>

</body>

</html>

**96 Решение задач на использование языка PHP для взаимодействия с базой данных MySQL**

В базе данных содержится информация о журналах: название, изображение обложки, год выпуска, номер, издательство, число страниц, цена.

Запросы:

а) вывести информацию о журналах с заданным названием;

б) вывести информацию о журналах, число страниц в которых лежит в заданном диапазоне.

[CREATE](http://localhost/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://localhost/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `magaz`.`jour` (`name` VARCHAR(100) [NOT](http://localhost/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL , `photo` VARCHAR(500) NULL , `god` INT [NOT](http://localhost/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL , `nomer` INT [NOT](http://localhost/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL , `izd` VARCHAR(300) [NOT](http://localhost/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL , `pages` INT [NOT](http://localhost/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL , `cena` FLOAT [NOT](http://localhost/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ) ENGINE = InnoDB;

SELECT \* FROM `jour` WHERE 1

"TRUNCATE TABLE `magaz`.`jour`"

**Список использованных источников**

1 Минник, К. JavaScript / К. Минник, Е. Холланд. - М. : Москва, 2017. - 320 с.

2 Маккоу, А Веб-приложения на JavaScript / А. Маккоу. - СПб. : Питер, 2014. - 285 с.

3 Макфарланд, Д. JavaScript и jQuery. Исчерпывающее руководство / Д. Макфарланд. - СПб.: Питер, 2016. - 880 с.

4 Жадаев, А.Г. PHP для начинающих / А.Г. Жадаев – СПб.: Питер, 2014. – 288 с.

5 Котеров Д.В. PHP 7 в подлиннике / Д.В. Котеров – СПб.: Питер, 2016. – 1073 с.

6 Никольский А.П. JavaScript на примерах / А.П. Никольский – СПб.: Питер, 2018. – 272 c.