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.**

**Исходный код страницы.**

<!-- 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;

let i = 1;

while ( i < array.length) {

result += array[i];

i = i + 2;

}

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>

let i = 1;<br>

while ( i < array.length) { <br>

result += array[i];<br>

i = i + 2; <br>

}<br>

console.log(result);<br>

alert(result);<br>

</body>

</html>

**Разъяснение исходного кода.**

<!DOCTYPE html> - Указание типа документа, HTML

<html lang='ru'> - Указание браузеру о языке страницы

<head> - заголовочная часть

<meta charset='utf-8'> - кодировка текста

<title>Добро пожаловать!</title> - заголовок страницы, будет выведено в названии вкладки.

</head>

Далее следует основная часть страницы

<body> - начало тела страницы

<script> - начало скрипта

function test(){ // создание фнкции для кнопки

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

let result = 0;// в эту переменную накапливаем сумму

let i = 1// начинаем со второго элемента

while ( i < array.length) {// пока не достигли конца массива

result += array[i] // накапливаем сумму

i = i + 2; // шаг =2, только четные позиции

}

console.log(result);// выводим результат в консоль (клавиша F12)

alert(result); // выводим результат в диалоговое окно

}

</script> - конец скриптовой части страницы

Далее следует часть страницы которая отображается на экране

<input type=button value='Вычислить' onclick = test()><br> Кнопка при нажатии на которую вызывается наша функция

<br>

<br> далее следует код функции

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

let result = 0;<br>

let i = 1;<br>

while ( i < array.length) { <br>

result += array[i];<br>

i = i + 2; <br>

}<br>

<br>

console.log(result);<br>

alert(result);<br>

</body> - завершение основного тела страницы

</html> - завершение документа, размеченного по стандартам HTML.

Вид страницы после открытия представлен на рисунке1.

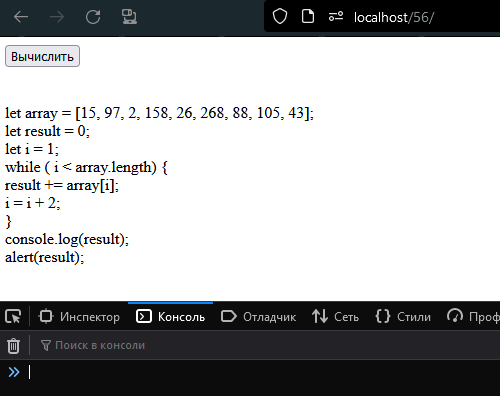


Рисунок 1. Вид страницы после открытия.

Вид страницы после нажатия кнопки представлен на рисунке 2.

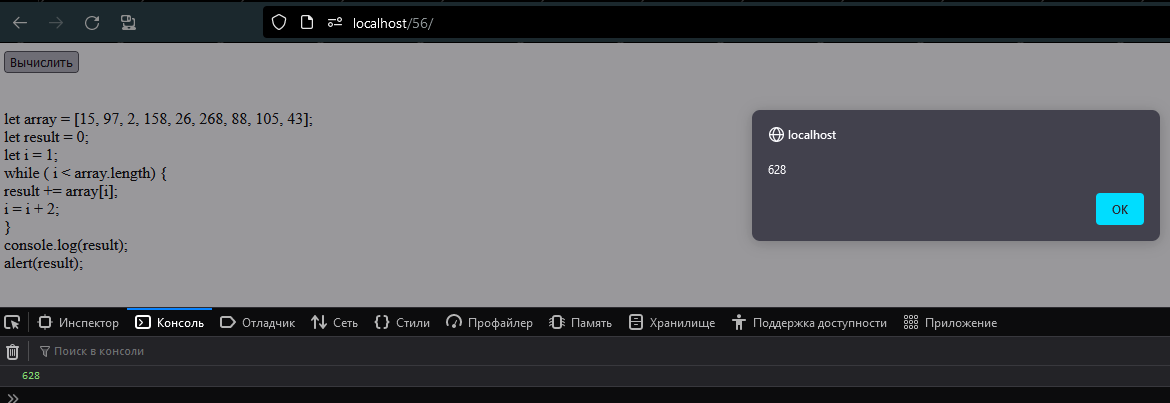


Рисунок 2. Вид страницы после нажатия кнопки.

**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.**

**Исходный код страницы.**

<!-- 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>

**Разъяснение исходного кода.**

<!DOCTYPE html> - Указание типа документа, HTML

<html lang='ru'> - Указание браузеру о языке страницы

<head> - заголовочная часть

<meta charset='utf-8'> - кодировка текста

<title>Добро пожаловать!</title> - заголовок страницы, будет выведено в названии вкладки.

</head>

Далее следует основная часть страницы

<body> - начало тела страницы

<script> - начало скрипта

function test(){ задание функции обработки массива

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

let emin = array[0]; // в качестве минимального для начала везьмем первый элемент массива

let pmin = 0; // и его позицию

console.log(array); // выведем в консоль изначальный массив

for (let i = 1; i < array.length; i++) { // по всем элементам массива кроме первого

if(array[i] < emin){ // сравним с минимальным

emin = array[i]; // и если текущий меньше минимального

pmin = i; // то запомним минимальный и его позицию

}

}

let t = array[9];// сохраним во временную переменную последний элемент

array[9] = array[pmin]; // а на его место запишем минимальный

array[pmin] = t; // на место минимального запишем последний

console.log(emin);// выведем в консоль минимальный элемент

console.log(array); // и новый массив, будет красиво выглядеть

alert(emin);// и сообщением на экран выведем минимальный элемент

}

</script> - завершение части скрипта

Далее следует часть, которая выводится на экран

<input type=button value='Выполнить' onclick = test()><br> кнопка для вызова нашего скрипта на исполнение

<br>

<br> далее следует текст функции

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> - завершение основного тела страницы

</html> - завершение документа, размеченного по стандартам HTML.

Вид страницы после открытия представлен на рисунке 3.



Рисунок 3. Вид страницы после открытия.

Вид страницы после нажатия на кнопку представлен на рисунке 4,

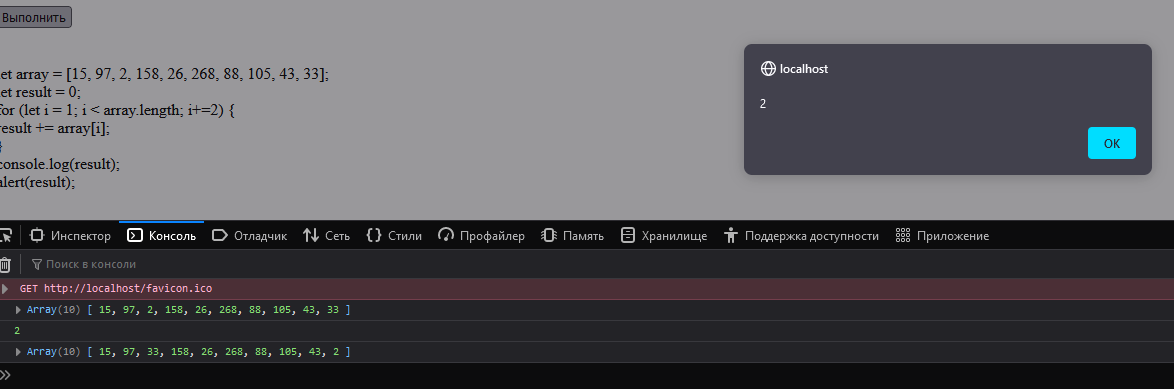


Рисунок 4. Вид страницы после нажатия на кнопку

**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.

**Исходный код страницы.**

<!-- index.php -->

<!-- далее идут заголовки файла для корректного отображения в браузере-->

<!DOCTYPE html>

<html lang='ru'>

<head>

<meta charset='utf-8'>

<title>Добро пожаловать!</title>

</head>

<body>

<!-- начинается код-->

<?php

// тут мы переносим переменные параметры из глобального массива

// в локальный, сделано для безопасности на стороне сервера

// ограничение по передаче переменных

isset($\_GET['act'])?$act = $\_GET['act']:$act=0;

isset($\_POST['nm'])?$nm = $\_POST['nm']:$nm="";

isset($\_POST['s1'])?$s1 = $\_POST['s1']:$s1="0";

isset($\_POST['s2'])?$s2 = $\_POST['s2']:$s2="0";

// соединяемся с базой данных

$link = mysqli\_connect("localhost", "root", "");

if ($link == false){

print("Ошибка: Невозможно подключиться к MySQL " . mysqli\_connect\_error()."<br>");

}

else {

echo "Соединение с базой данных установлено успешно<br>";

}

// если скрипт запускается впервые а базы не существует то создадим её

$sql = "CREATE DATABASE IF NOT EXISTS `magaz` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci;";

$result = mysqli\_query($link, $sql);

// и выберем для работы

$sql = "USE `magaz`;";

if (!isset($act)) $act = 0;

$result = mysqli\_query($link, $sql);

// организуем выбор режимов и этапов работы

switch ($act)

{

case 0:// часть которая выпоняется если нет параметров, основная

echo"<br><form action='index.php?act=1' method=POST><input type=submit value='Создать и заполнить базу'></form><br>";

break; // выводим кнопку для создания и заполнения тоаблицы

case 1: // в этой части мы создаем таблицу, если её не существует

echo"<br> База данных журналов создана и заполнена. Содержимое: <br><br>";

$sql = "CREATE TABLE IF NOT EXISTS`magaz`.`jour` (`name` VARCHAR(100) NOT NULL , `photo` VARCHAR(500) NULL , `god` INT NOT NULL , `nomer` INT NOT NULL , `izd` VARCHAR(300) NOT NULL , `pages` INT NOT NULL , `cena` FLOAT NOT NULL ) ENGINE = InnoDB;";

$result = mysqli\_query($link, $sql);

// а если существует то очищаем

$sql = "TRUNCATE TABLE `magaz`.`jour`";

$result = mysqli\_query($link, $sql);

// и вставляем 10 записей с журналами

$sql = "INSERT INTO `jour` (`name`, `photo`, `god`, `nomer`, `izd`, `pages`, `cena`) VALUES ('Охота и рыбалка', '1.jpg', '1995', '1', 'Издательство 1', '10', '23.12');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `jour` (`name`, `photo`, `god`, `nomer`, `izd`, `pages`, `cena`) VALUES ('Крючки и спицы', '2.jpg', '1996', '2', 'Издательство 2', '20', '13.1');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `jour` (`name`, `photo`, `god`, `nomer`, `izd`, `pages`, `cena`) VALUES ('Кухня', '3.jpg', '1997', '3', 'Издательство 3', '30', '203.11');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `jour` (`name`, `photo`, `god`, `nomer`, `izd`, `pages`, `cena`) VALUES ('Ремонт', '4.jpg', '1998', '4', 'Издательство 4', '40', '23.00');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `jour` (`name`, `photo`, `god`, `nomer`, `izd`, `pages`, `cena`) VALUES ('Кройка и шитьё', '5.jpg', '1999', '5', 'Издательство 5', '50', '150.12');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `jour` (`name`, `photo`, `god`, `nomer`, `izd`, `pages`, `cena`) VALUES ('Борщ и щи', '6.jpg', '2000', '6', 'Издательство 6', '60', '77.12');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `jour` (`name`, `photo`, `god`, `nomer`, `izd`, `pages`, `cena`) VALUES ('Спид-Инфо', '7.jpg', '2001', '7', 'Издательство 7', '70', '86.12');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `jour` (`name`, `photo`, `god`, `nomer`, `izd`, `pages`, `cena`) VALUES ('Банк и финансы', '8.jpg', '2002', '8', 'Издательство 8', '80', '69.12');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `jour` (`name`, `photo`, `god`, `nomer`, `izd`, `pages`, `cena`) VALUES ('Телевизор', '9.jpg', '2003', '9', 'Издательство 9', '90', '5.12');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `jour` (`name`, `photo`, `god`, `nomer`, `izd`, `pages`, `cena`) VALUES ('Мода', '10.jpg', '2004', '10', 'Издательство 10', '100', '36.12');";

$result = mysqli\_query($link, $sql);

// выводим жданные в таблицу

echo"<table border=1>";// начало таблицы

$sql = 'SELECT \* FROM jour';// выбираем все записи

$result = mysqli\_query($link, $sql);

// выводим заголовки таблицы

echo"<tr><th>Название</th><th>Фото</th><th>Год</th><th>Номер</th><th>Издание</th><th>Страниц</th><th>Цена</th></tr>";

while ($row = mysqli\_fetch\_array($result)) {// и по каждой записи из выборки

echo"<tr><td>" . $row['name'] . "</td><td><img src='" . $row['photo'] . "' width=100></td><td>". $row['god'] . "</td><td>". $row['nomer'] . "</td>";

echo "<td>" . $row['izd'] . "</td><td>". $row['pages'] . "</td><td>". $row['cena'] . "</td></tr>";

} // формируем строки таблицы

echo"</table>";// закрываем таблицу

// далее выводим кнопки с выбором варианта действий

echo"<br><form action='index.php?act=0' method=POST><input type=submit value='Первая страница'></form>";

echo"<br><form action='index.php?act=2' method=POST><input type=submit value='Задание А'></form>";

echo"<br><form action='index.php?act=3' method=POST><input type=submit value='Задание Б'></form>";

break;

case 2: // Выбрано действие поиска журнала по названию, создаем форму

echo"<br><form action='index.php?act=21' method=POST>

<br>Введите название журнала:<input type=text name='nm'><br>

<input type=submit value='Вывести'></form>"; // в ней запрашиваем название

// далее выводим кнопки с выбором варианта действий

echo"<br><form action='index.php?act=0' method=POST><input type=submit value='Первая страница'></form>";

echo"<br><form action='index.php?act=2' method=POST><input type=submit value='Задание А'></form>";

echo"<br><form action='index.php?act=3' method=POST><input type=submit value='Задание Б'></form>";

break;

case 21: // выбираем журнал по названию, nm - параметр с названием, формируем запрос

$sql = "SELECT \* FROM jour where name='".$nm."'";

$result = mysqli\_query($link, $sql);

// выводим данные в таблицу

echo"<table border=1>";

echo"<tr><th>Название</th><th>Фото</th><th>Год</th><th>Номер</th><th>Издание</th><th>Страниц</th><th>Цена</th></tr>";

while ($row = mysqli\_fetch\_array($result)) {

echo"<tr><td>" . $row['name'] . "</td><td><img src='" . $row['photo'] . "' width=100></td><td>". $row['god'] . "</td><td>". $row['nomer'] . "</td>";

echo "<td>" . $row['izd'] . "</td><td>". $row['pages'] . "</td><td>". $row['cena'] . "</td></tr>";

}

echo"</table>";

// далее выводим кнопки с выбором варианта действий

echo"<br><form action='index.php?act=0' method=POST><input type=submit value='Первая страница'></form>";

echo"<br><form action='index.php?act=2' method=POST><input type=submit value='Задание А'></form>";

echo"<br><form action='index.php?act=3' method=POST><input type=submit value='Задание Б'></form>";

break;

case 3: // выводим форму для выбора диапазона страниц

echo"<br><form action='index.php?act=31' method=POST>

<br>Введите страницы от:<input type=text name='s1'> и до:<input type=text name='s2'><br>

<input type=submit value='Вывести'></form>";

// далее выводим кнопки с выбором варианта действий

echo"<br><form action='index.php?act=0' method=POST><input type=submit value='Первая страница'></form>";

echo"<br><form action='index.php?act=2' method=POST><input type=submit value='Задание А'></form>";

echo"<br><form action='index.php?act=3' method=POST><input type=submit value='Задание Б'></form>";

break;

case 31: // формируем запрос с параметрами s1 и s2

$sql = "SELECT \* FROM jour where pages between ".$s1." and ".$s2;

$result = mysqli\_query($link, $sql);

// выводим результат таблицей

echo"<table border=1>";

echo"<tr><th>Название</th><th>Фото</th><th>Год</th><th>Номер</th><th>Издание</th><th>Страниц</th><th>Цена</th></tr>";

while ($row = mysqli\_fetch\_array($result)) {

echo"<tr><td>" . $row['name'] . "</td><td><img src='" . $row['photo'] . "' width=100></td><td>". $row['god'] . "</td><td>". $row['nomer'] . "</td>";

echo "<td>" . $row['izd'] . "</td><td>". $row['pages'] . "</td><td>". $row['cena'] . "</td></tr>";

}

echo"</table>";

// далее выводим кнопки с выбором варианта действий

echo"<br><form action='index.php?act=0' method=POST><input type=submit value='Первая страница'></form>";

echo"<br><form action='index.php?act=2' method=POST><input type=submit value='Задание А'></form>";

echo"<br><form action='index.php?act=3' method=POST><input type=submit value='Задание Б'></form>";

break;

}

?>

</body></html>

**Разъяснение хода выполнения задания.**

Конструктивно программа состоит из одного файла которому передаются параметры для выбора необходимой операции. Если параметров не передано, то открывается стартовая страница, на которой можно создать и заполнить базу данных при её отсутствии. После создания и заполнения базы становятся доступными кнопки выбора заданий. Каждое задание состоит из 2х частей: задание необходимых параметров и выполнение задания с выводом результата. Результат выполнения выводится в динамически генерируемой таблице. В нижней части страницы располагаются навигационные кнопки.

Стартовая страница представлена на рисунке 5.

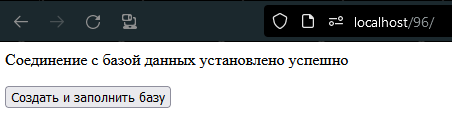


Рисунок 5. Стартовая страница.

Результат создания и заполнения базы данных. Представлен на рисунке 6.



Рисунок 6. Результат создания и заполнения базы данных.

Окно заполнения параметров задания А представлено на рисунке 7.

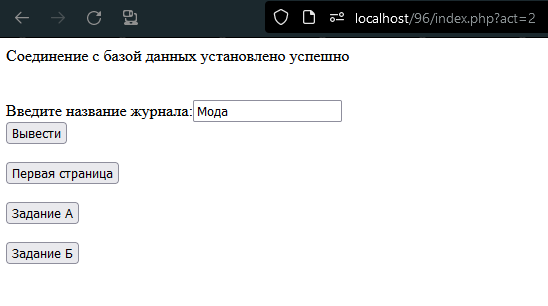


Рисунок 7. Окно заполнения параметров задания А.

Результат выполнения задания А представлен на рисунке 8.

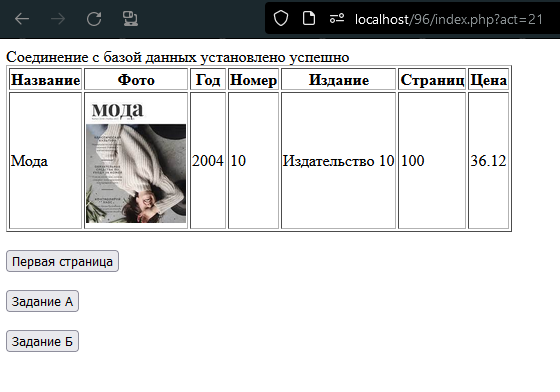


Рисунок 8. Результат выполнения задания А

.

Окно заполнения параметров задания Б представлено на рисунке 9.

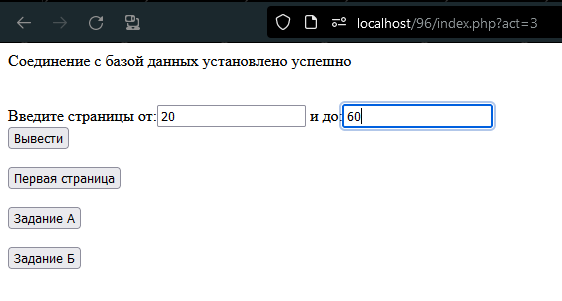


Рисунок 9. Окно заполнения параметров задания Б.

На рисунке 10 представлен результат выполнения задания Б.

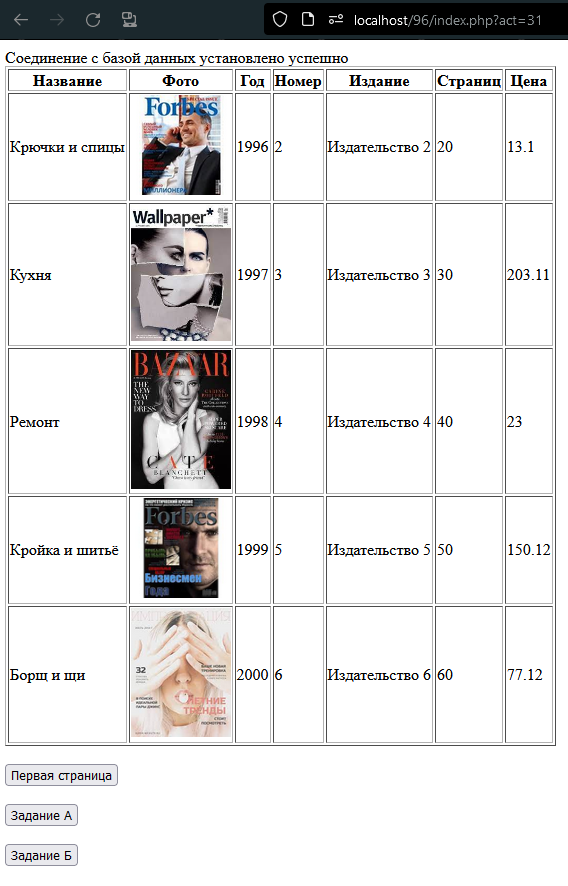


Рисунок 10. Результат выполнения задания Б.

**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.