|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 – ИУ6-32Б**

**Отчет**

**по лабораторной работе № 1**

**Название:** Основы JavaScript

**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-32Б |  |  | Э. Кулиев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В.Д. Шульман |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

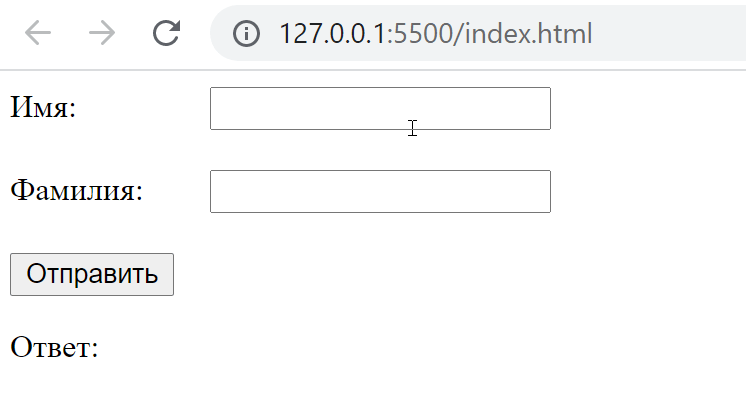
Москва, 2024

**Цель работы:** научиться работать с Javascript

**Задание 1**

**Что должно получиться:**пользователь открывает страницу, вводит имя и фамилию, нажимает "Отправить". Внизу появляется текст с приветствием. Затем можно изменить фамилию или имя, вновь нажать "Отправить" - приветствие обновится.

В данном случае не нужно использовать форму <form>.



**Критерии оценивания:**

* вы можете применить к странице любые стили или не применять их вовсе;
* не обязательно, чтобы элементы были расположены как в примере;
* **ОБЯЗАТЕЛЬНО**: в input вводятся имя и фамилия, после нажатия на кнопку Отправить появляется приветственное предложение в формате: Здравствуйте, Имя Фамилия!
* засчитывайте варианты решения, если приветствие появляется при введении одного из полей или при отсутствии введенных значений. Здорово, если у вас получится сделать проверку на ввод значений, и только в этом случае показывать приветствие.

В ответ на задание приложите ссылку на любое облачное хранилище с размещенной там полной папкой вашего проекта или ссылку на CodePen. Не забывайте разрешить доступ, иначе задание будет оценено в ноль баллов.

**Код:**

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Приветствие</title>

    <style>

        body {

            font-family: Arial, sans-serif;

        }

        .container {

            max-width: 300px;

            margin: 0 auto;

            padding: 20px;

            border: 1px solid #ccc;

            border-radius: 5px;

            text-align: center;

        }

        input {

            width: 100%;

            padding: 10px;

            margin: 10px 0;

            box-sizing: border-box;

        }

        #greeting {

            margin-top: 20px;

            font-size: 18px;

            color: #333;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Форма приветствия</h1>

        <input type="text" id="firstName" placeholder="Имя">

        <input type="text" id="lastName" placeholder="Фамилия">

        <button onclick="showGreeting()">Отправить</button>

        <div id="greeting"></div>

    </div>

    <script>

        function showGreeting() {

            const firstName = document.getElementById('firstName').value;

            const lastName = document.getElementById('lastName').value;

            if (firstName && lastName) {

                document.getElementById('greeting').innerText = `Здравствуйте, ${firstName} ${lastName}!`;

            } else {

                document.getElementById('greeting').innerText = '';

            }

        }

    </script>

</body>

</html>

**Результат программы:**

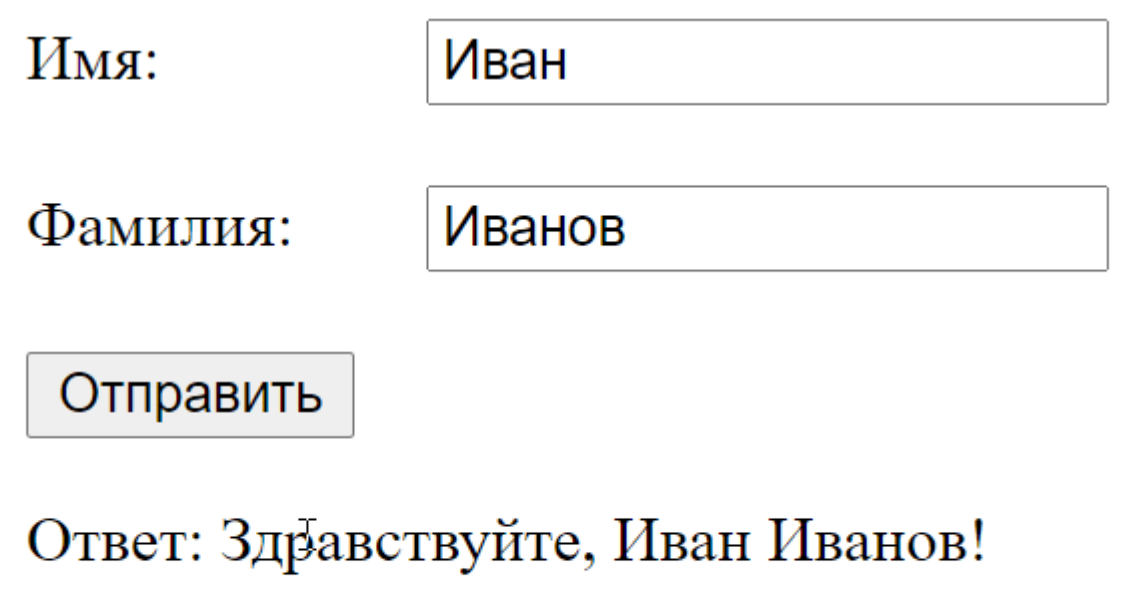
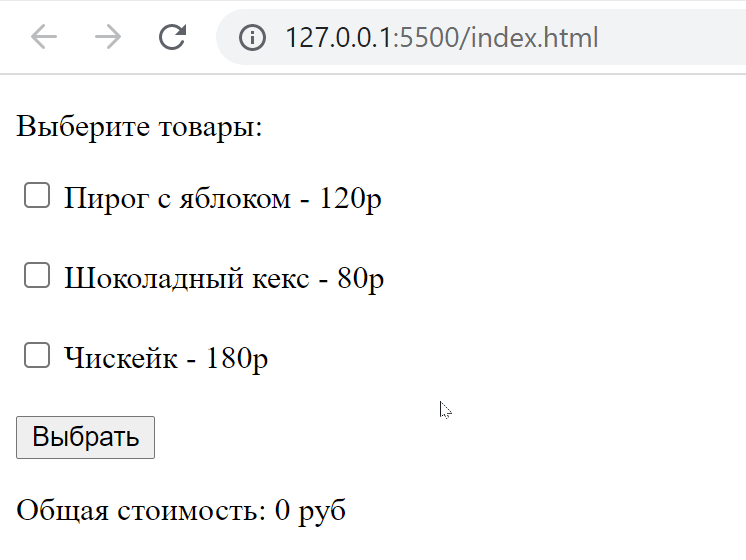
****

Рис.1

**Задание 2**

**Что должно получиться:**пользователь открывает страницу и выбирает товары, которые хочет приобрести. После этого нажимает «Выбрать» и в строке «Общая стоимость» появляется стоимость покупки. Если пользователь меняет выбранные товары, после нажатия на кнопку «Выбрать» происходит перерасчет «Общей стоимости».



**Критерии оценивания:**

* вы можете применить к странице любые стили;
* не обязательно, чтобы элементы были расположены как в примере;
* **ОБЯЗАТЕЛЬНО**: наличие элементов checkbox, по нажатию на кнопку Выбрать рассчитывается и выводится Общая стоимость выбранных продуктов. Для основы можно взять код из предыдущего шага. У элементов **input**с типом **checkbox**в данном случае в **value**указана стоимость (<input type="checkbox" id="Choice3" name="food" value="180">).

В ответ на задание приложите ссылку на любое облачное хранилище с размещенной там полной папкой вашего проекта или ссылку на CodePen. Не забывайте разрешить доступ, иначе задание будет оценено в ноль баллов.

**Код:**

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Расчет стоимости товаров</title>

<style>

body {

font-family: Arial, sans-serif;

}

.container {

max-width: 300px;

margin: 0 auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 5px;

text-align: center;

}

.products {

text-align: left;

}

.result {

margin-top: 20px;

font-size: 18px;

color: #333;

}

</style>

</head>

<body>

<div class="container">

<h1>Выбор товаров</h1>

<form class="products">

<label><input type="checkbox" name="product" value="100"> Товар 1 (100 рублей)</label><br>

<label><input type="checkbox" name="product" value="200"> Товар 2 (200 рублей)</label><br>

<label><input type="checkbox" name="product" value="300"> Товар 3 (300 рублей)</label><br>

</form>

<button onclick="calculateTotal()">Выбрать</button>

<div class="result" id="totalCost">Общая стоимость: 0 рублей</div>

</div>

<script>

function calculateTotal() {

const checkboxes = document.querySelectorAll('input[name="product"]:checked');

let total = 0;

checkboxes.forEach(checkbox => {

total += parseInt(checkbox.value);

});

document.getElementById('totalCost').innerText = `Общая стоимость: ${total} рублей`;

}

</script>

</body>

</html>

**Результат программы:**

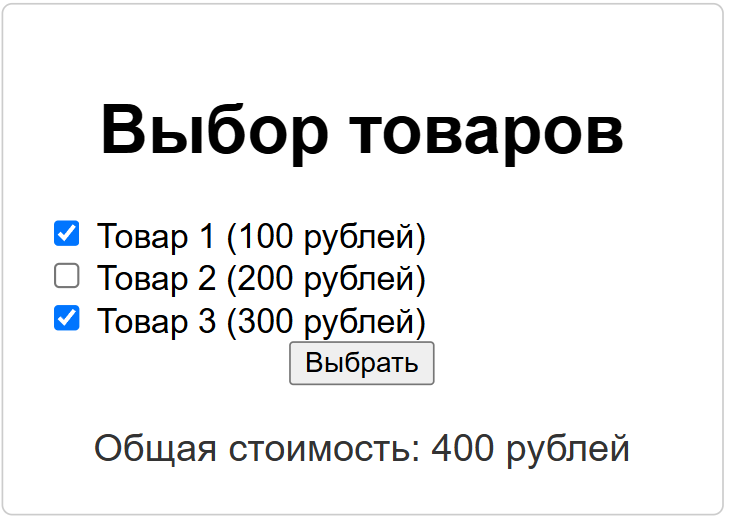
****

Рис.2

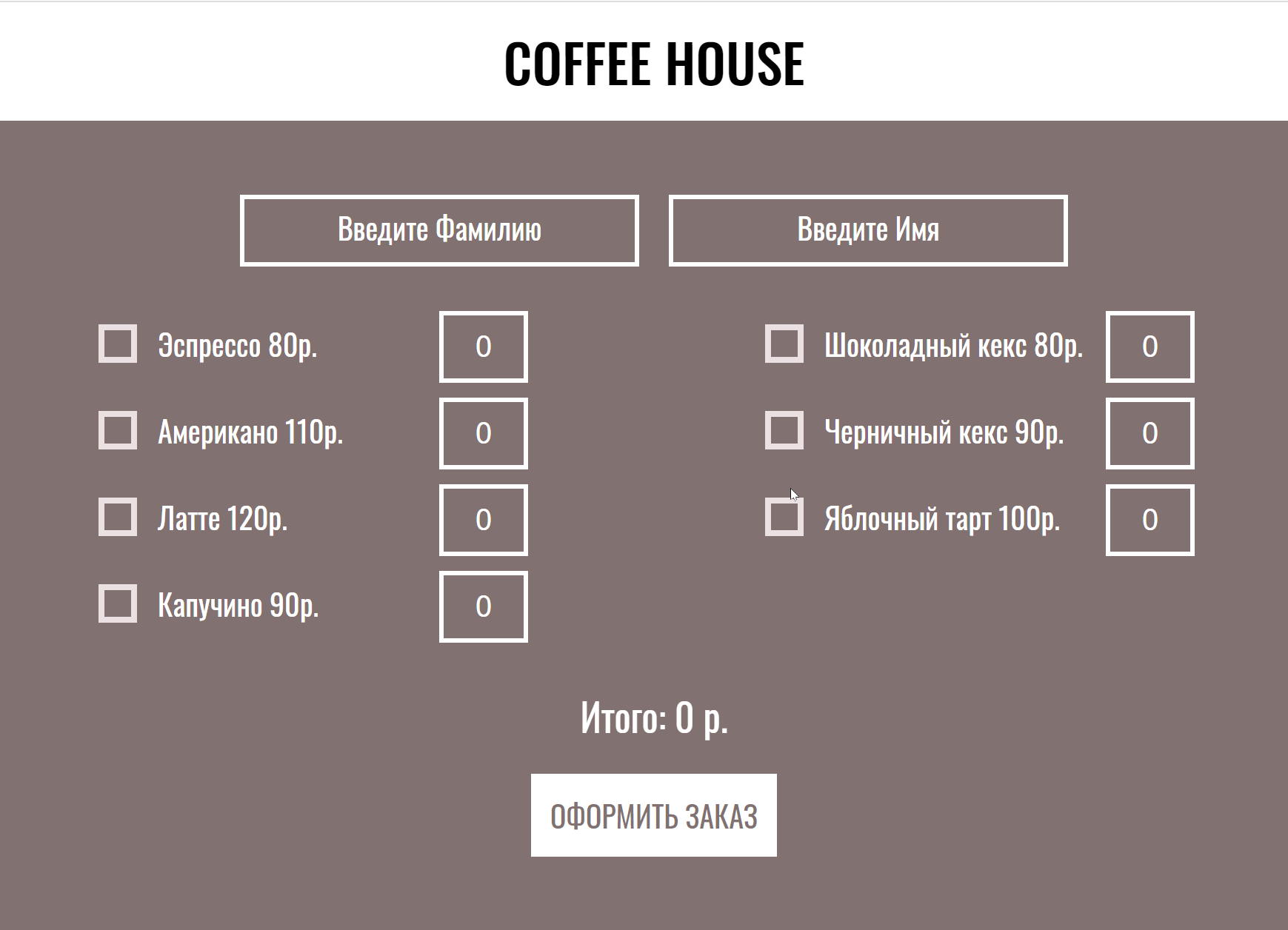
**Задание 3**

Вам нужно с помощью JavaScript сделать страницу "живой". Перед вами страница оформления заказа на сайте кофейни. Пользователь отмечает товар, который хочет заказать и указывает кол-во. По умолчанию кол-во равно 0.

Внизу страницы есть строка Итого, которая меняет свое значение в процессе действий пользователя.

По нажатию на кнопку "Оформить заказ" появляется модальное окно, где указаны заказчик (Фамилия Имя) и Итого (общая сумма покупки).

Пример работы:



**Что нужно реализовать:**

* Поля ввода для Фамилии и Имени: по кнопке "Оформить заказ" в модальном окне alert выводится строка *Заказчик: Фамилия Имя* (те, что были введены пользователем, если ничего не введено - пустая строка).
* Чекбоксы с наименованиями товаров. Пользователь нажимает на чекбокс и программа должна понимать, что данный товар выбран, а значит его нужно учитывать в общей сумме, либо наоборот - чекбокс снят, значит данный товар должен исчезнуть из суммы покупки.
* Рядом с каждым товаром находится поле ввода для указания количества товара. По умолчанию значение равно 0. В общей сумме покупки должно учитываться число, которое вводит пользователь в данное поле. Однако, если пользователь ввел число, а чекбокс не выбрал - данное кол-во не должно учитываться.
* Внизу над кнопкой "Оформить заказ" пользователю всегда пересчитывается итоговая сумма покупки исходя из действий пользователя.

**Код:**

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Оформление заказа</title>

<style>

body {

font-family: Arial, sans-serif;

max-width: 400px;

margin: 0 auto;

padding: 20px;

}

.container {

border: 1px solid #ccc;

padding: 20px;

border-radius: 5px;

}

.field {

margin-bottom: 10px;

}

.result {

margin-top: 20px;

font-size: 18px;

color: #333;

}

.result span {

font-weight: bold;

}

.buttons {

text-align: center;

}

</style>

</head>

<body>

<div class="container">

\*\*Оформление заказа\*\*

<div class="field">

<label for="lastName">Фамилия:</label>

<input type="text" id="lastName">

</div>

<div class="field">

<label for="firstName">Имя:</label>

<input type="text" id="firstName">

</div>

\*\*Выберите товары и укажите количество:\*\*

<div class="field">

<label><input type="checkbox" name="product" value="100" data-id="p1"> Кофе (100 рублей)</label>

<input type="number" id="p1" value="0" min="0">

</div>

<div class="field">

<label><input type="checkbox" name="product" value="150" data-id="p2"> Чай (150 рублей)</label>

<input type="number" id="p2" value="0" min="0">

</div>

<div class="field">

<label><input type="checkbox" name="product" value="200" data-id="p3"> Круассан (200 рублей)</label>

<input type="number" id="p3" value="0" min="0">

</div>

\*\*Итого: <span id="totalCost">0</span> рублей\*\*

<div class="buttons">

<button onclick="submitOrder()">Оформить заказ</button>

</div>

</div>

<script>

document.querySelectorAll('input[type="checkbox"]').forEach(checkbox => {

checkbox.addEventListener('change', updateTotal);

});

document.querySelectorAll('input[type="number"]').forEach(numberInput => {

numberInput.addEventListener('input', updateTotal);

});

function updateTotal() {

let total = 0;

document.querySelectorAll('input[type="checkbox"]:checked').forEach(checkbox => {

const productId = checkbox.getAttribute('data-id');

const quantityInput = document.getElementById(productId);

const quantity = parseInt(quantityInput.value) || 0;

if (quantity > 0) {

total += parseInt(checkbox.value) \* quantity;

}

});

document.getElementById('totalCost').innerText = total;

}

function submitOrder() {

const lastName = document.getElementById('lastName').value || "";

const firstName = document.getElementById('firstName').value || "";

const total = document.getElementById('totalCost').innerText || "0";

alert(`Заказчик: ${lastName} ${firstName}\nИтого: ${total} рублей`);

}

</script>

</body>

</html>

**Результат выполнения программы:**

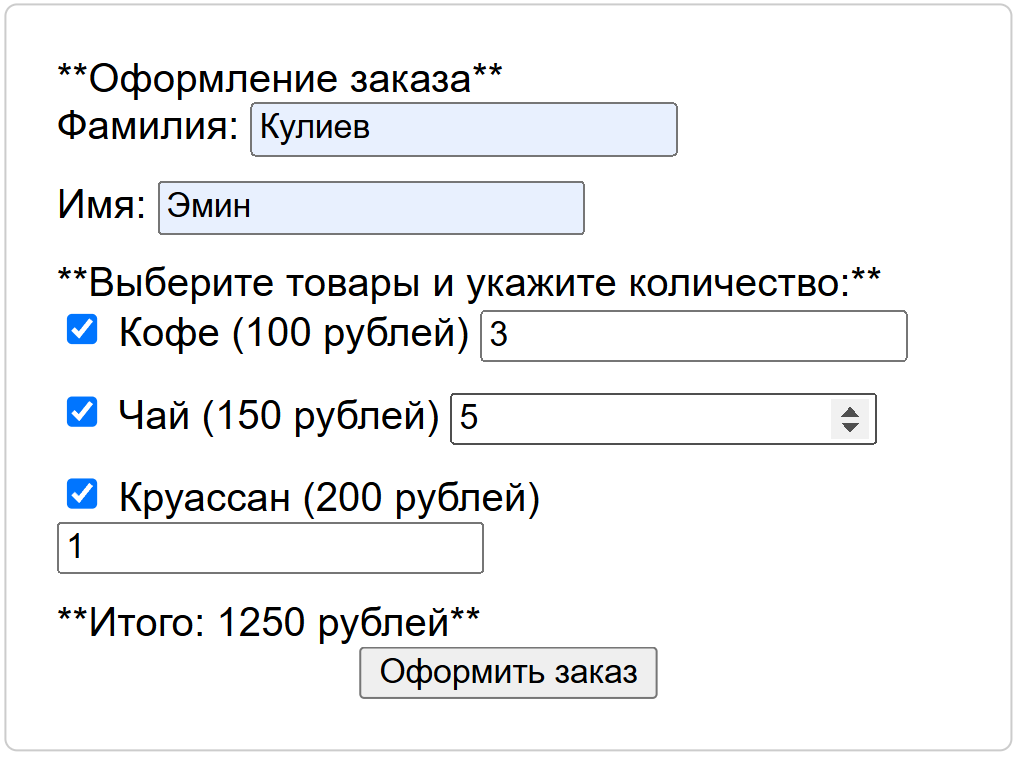
****

Рис. 3

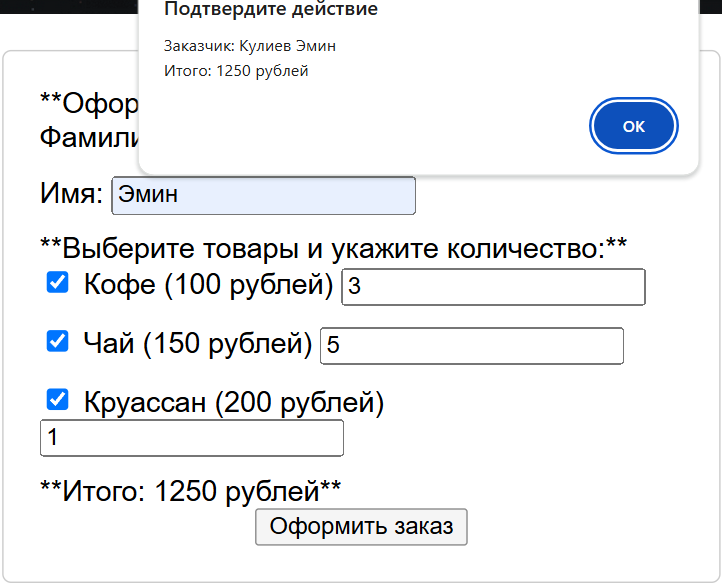


Рис. 4

Заключение

Я научился работать с JavaScript. Как доставать конкретные элементы страницы, добавлять слушатели событий и изменять и считывать значения элементов.

Контрольные вопросы

1. В чём разница между let и var? -

var объявляет переменную в глобальной области видимости или в области видимости функции, в которой она объявлена. Это означает, что переменная будет доступна во всей функции или даже за пределами функции, если она объявлена вне функции.

let, с другой стороны, объявляет переменную в блоковой области видимости, то есть она доступна только внутри блока, в котором она объявлена. Это означает, что переменная не будет доступна вне блока, даже если блок находится внутри функции.

2. Какие существуют способы подключения js-кода к html-документу?

Существует два основных способа подключить JavaScript-код к HTML-документу:

1. Встраивание кода непосредственно в HTML:\*\* Это делается с помощью тега `<script>`. JavaScript-код пишется непосредственно между открывающим и закрывающим тегами `<script>` и `<\/script>`. Этот метод подходит для небольших фрагментов кода, которые должны быть выполнены на одной странице.

2. Подключение внешнего файла JavaScript: Это делается с помощью атрибута `src` в теге `<script>`. В атрибуте `src` указывается путь к внешнему файлу `.js`, содержащему JavaScript-код. Этот метод предпочтительнее для больших проектов, так как позволяет организовать код более эффективно и упрощает его обслуживание. Код из внешнего файла выполняется после загрузки страницы.

\*\*3. В чём отличие операторов `===` и `==`?

Операторы `===` (строгое равенство) и `==` (равенство) используются для сравнения значений в JavaScript, но имеют ключевое различие:

`==` (равенство): Этот оператор выполняет нестрогое сравнение. Он пытается преобразовать значения к одному типу перед сравнением. Например, `1 == "1"` вернет `true`, потому что строка "1" неявно преобразуется в число 1 перед сравнением. Это может привести к неожиданным результатам, поэтому его использование часто не рекомендуется.

`===` (строгое равенство):\*\* Этот оператор выполняет \*строгое\* сравнение. Он сравнивает значения без преобразования типов. Если типы значений различны, то результат сравнения всегда будет `false`. Например, `1 === "1"` вернет `false`, потому что типы данных разные (число и строка). Для надежного и предсказуемого сравнения всегда предпочтительнее использовать строгий оператор `===`.

\*\*4. Как можно отладить js-код с помощью браузера?

Отладка JavaScript-кода в браузере обычно осуществляется через встроенные инструменты разработчика:

1. \*\*Консоль (Console): В консоли отображаются сообщения, выводимые с помощью функций `console.log()`, `console.error()`, `console.warn()` и других. Это помогает отслеживать выполнение кода и выявлять ошибки. Консоль также отображает ошибки и предупреждения, возникающие во время выполнения скрипта.

2. \*\*Точки останова (Breakpoints): Точки останова позволяют приостановить выполнение кода в определенном месте. Это позволяет пошагово проанализировать состояние переменных и выполнить код построчно, чтобы найти причину ошибки. Для установки точки останова обычно нужно кликнуть в левой части строки кода в отладчике.

3. \*\*Отладчик (Debugger): Встроенный отладчик позволяет пошагово проходить по коду, просматривать значения переменных, вызывать функции и т.д. Он предоставляет мощные инструменты для детального анализа выполнения кода.

5. Для чего нужна директива `use strict`?

Директива `'use strict'` включает строгий режим в JavaScript. Включение строгого режима накладывает дополнительные ограничения на код, что приводит к:

Улучшению безопасности: Предотвращает случайное создание глобальных переменных. В строгом режиме, если переменная не объявлена с помощью `var`, `let` или `const`, то вы получите ошибку.

Улучшению производительности: В некоторых случаях строгий режим может незначительно повысить производительность, так как компилятор может оптимизировать код лучше.

Более предсказуемому поведению: Избегает неявных преобразований типов и других неопределённых поведений, которые могут привести к неожиданным результатам.

Ошибки на этапе компиляции: Многие ошибки, которые в обычном режиме проявились бы только во время выполнения, будут обнаружены на этапе компиляции, что облегчает отладку.

Ограничения строгого режима:

Запрещается использовать необъявленные переменные.

Запрещается удалять переменные и функции с помощью оператора `delete`.

Запрещается использовать зарезервированные слова в качестве имён переменных.

\* Запрещается использовать инструкцию `with`.

В целом, использование `'use strict'` рекомендуется для повышения качества и надёжности JavaScript-кода.