**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Кафедра безопасности информационных систем**

**ОТЧЁТ**

по практическому заданию № 4

Предмет  
**«Технологии Front-end разработки веб-приложений»**

Выполнил: студент группы ИБ-01вп,

Яковлев Е.Ю.

САНКТ-ПЕТЕРБУРГ  
2024

Содержание

[Задание 3](#__RefHeading___1)

[Ход работы часть 1 4](#__RefHeading___2)

[Код часть 1 4](#__RefHeading___3)

[HTML 4](#__RefHeading___4)

[JavaScript 5](#__RefHeading___5)

[CSS 7](#__RefHeading___6)

[Пояснение к коду 8](#__RefHeading___7)

[Ход работы часть 2 11](#__RefHeading___8)

[Код часть 2 11](#__RefHeading___9)

[Пояснение к коду часть 2 12](#__RefHeading___10)

[Вывод 13](#__RefHeading___11)

# Задание

Используя HTML, CSS и язык программирования JavaScript создать тестовую html-страницу для расчета стоимости заказа, сделанного покупателем в интернет-магазине.

Этапы работы:

1. Создать поля для ввода данных в соответствии с образцом, приведенным на рисунке 1.

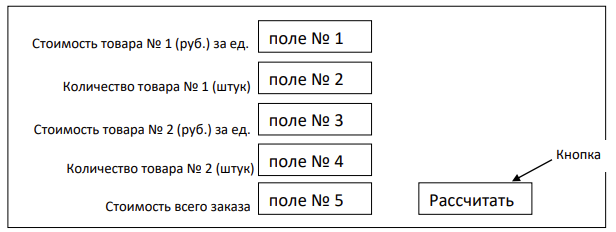


Рисунок 1 - пример формы

2. Написать программу, выполняющую следующие задачи:

* Проверку введенных данных на наличие только (целых или дробных) чисел в поля 1 и 3;
* Проверку введенных данных на наличие только целых чисел в поля 2 и 4;
* Запрет (для пользователей сайта) ввода данных в поле № 5;
* Проведение расчета стоимости всего заказа с выводом результата в поле № 5;
* При некорректно введенных данных выводить диагностику на экран;

В задаче использовать HTML, CSS и язык программирования JavaScript.

1. Создать на одной из страниц вашего сайта поле для ввода текста (например: текста с отзывом о сайте). Параметры создаваемого поля: 60 символов в строке, количество строк 17 (и кнопку для сохранения или обработки данных).

3. Организовать вывод введенного текста в другом блоке с помощью тега <div>.

4. При выводе в блоке <div> первую букву текста сделать в 2 раза больше остальных букв и цвет у первой буквы сделать красным.

5. Вставить в созданное поле следующий текст: Теги не фильтруются <a href="http://sut.ru" >Hacked by Vasya!”)</a> сайт уязвим для атак XSS.

6. Организовать проверку вводимого в поле текста на отсутствие в нем тегов.

7. Автоматически удалять введенные теги в тексте. 8. Выводить диагностику на экран о наличии тегов в тексте.

# Ход работы часть 1

1. Создаем HTML структуру сайта и подключаем сторонние файлы CSS и JavaScript.
2. Создаем JavaScript, в нем получаем данные из формы, проверяем введенные данные, рассчитываем результат и возвращаем в форму.
3. Создаем CSS файл и прописываем в нем селекторы для настройки стилей.

# Код часть 1

# HTML

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<link rel="stylesheet" href="style.CSS">

<title>Laba4</title>

</head>

<body>

<form action="#">

<fieldset>

<legend>Расчет стоимости заказа</legend>

<div>

<label for="field1">Стоимость товара № 1 (руб.) за ед.</label>

<input type="text" id="field1" name="field1">

</div>

<div>

<label for="field2">Количество товара № 1 (штук)</label>

<input type="text" id="field2" name="field2">

</div>

<div>

<label for="field3">Стоимость товара № 2 (руб.) за ед.</label>

<input type="text" id="field3" name="field3">

</div>

<div>

<label for="field4">Количество товара № 2 (штук)</label>

<input type="text" id="field4" name="field4">

</div>

<div>

<label for="field5">Стоимость всего заказа</label>

<input type="text" id="field5" name="field5" readonly>

</div>

<div>

<button type="button" id="calculateButton">Рассчитать</button>

</div>

</fieldset>

</form>

<script src="script.js"></script>

</body>

</html>

# JavaScript

// Получаем элементы формы

const field1 = document.getElementById('field1');

const field2 = document.getElementById('field2');

const field3 = document.getElementById('field3');

const field4 = document.getElementById('field4');

const field5 = document.getElementById('field5');

const calculateButton = document.getElementById('calculateButton');

// Функция проверки ввода чисел

function isNumeric(n) {

return !isNaN(parseFloat(n)) && isFinite(n);

}

// Функция проверки ввода целых чисел

function isInteger(n) {

return Number.isInteger(parseFloat(n));

}

// Функция расчета стоимости заказа

function calculateCost() {

// Проверяем введенные данные

if (!isNumeric(field1.value)) {

alert('В поле 1 должны быть введены только числа');

return;

}

if (!isInteger(field2.value)) {

alert('В поле 2 должны быть введены только целые числа');

return;

}

if (!isNumeric(field3.value)) {

alert('В поле 3 должны быть введены только числа');

return;

}

if (!isInteger(field4.value)) {

alert('В поле 4 должны быть введены только целые числа');

return;

}

// Расчитываем стоимость заказа

const cost = parseFloat(field1.value) \* parseInt(field2.value) + parseFloat(field3.value) \* parseInt(field4.value);

// Выводим результат в поле 5

field5.value = cost.toFixed(2);

}

// Добавляем обработчик события на кнопку расчета стоимости

calculateButton.addEventListener('click', calculateCost);

# CSS

body {

display: flex;

align-items: center;

justify-content: center;

height: 100vh;

width: 100vw;

font-family: cursive;

}

input {

width: 100%;

padding: 5px;

outline: none;

}

label {

line-height: 1.9rem;

}

button {

background-color: #025326;

color: white;

font-size: 16px;

padding: 10px 20px;

margin-top: 10px;

border: none;

border-radius: 105px;

cursor: pointer;

}

fieldset {

padding: 20px 40px;

}

# Пояснение к коду

<!DOCTYPE html>

Это объявление типа документа (DOCTYPE) и является первой строкой в любом HTML-документе. Оно указывает браузеру, какую версию HTML использует страница, чтобы браузер мог правильно ее отобразить. В этом случае используется HTML5.

<html> Это корневой элемент HTML-документа. Все остальные элементы должны быть помещены внутри этого тега. <head> Этот тег содержит метаинформацию о документе, которая не отображается на странице, но используется браузером и поисковыми системами. Это может включать в себя заголовок страницы, внешние файлы CSS и JavaScript, кодировку символов и другую информацию. <meta charset="utf-8"> Этот тег устанавливает кодировку символов для документа. В этом случае используется кодировка UTF-8, которая поддерживает большинство языков мира. <link rel="stylesheet" href="style.CSS"> Этот тег устанавливает внешнее связь с файлом CSS для стилизации страницы. Атрибут rel указывает тип связи (в этом случае стилевой лист), а атрибут href указывает путь к файлу. <title>Laba4</title> Этот тег устанавливает заголовок страницы, который отображается в закладке браузера и в результатах поиска. <body> Этот тег содержит все видимое содержимое страницы, включая текст, изображения, формы и другие элементы. <form action="#"> Этот тег создает форму, которую пользователь может заполнить и отправить на сервер. Атрибут action указывает URL-адрес, по которому будет отправлена форма. В этом случае форма не отправляется на сервер, а используется только для расчета стоимости заказа на клиентской стороне. <fieldset> Этот тег используется для группировки связанных элементов формы, таких как радиокнопки или чекбоксы. В этом случае он используется для группировки всех элементов формы. <legend>Расчет стоимости заказа</legend> Этот тег устанавливает заголовок для группы элементов формы, заключенных в тег fieldset. <div> Этот тег является блочным элементом и используется для группировки других элементов в блоки.

<label for="field1">Стоимость товара № 1 (руб.) за ед.</label>

Этот тег создает метку для элемента формы, которая отображается рядом с ним. Атрибут for связывает метку с элементом формы по его атрибуту id.

<input type="text" id="field1" name="field1"> Этот тег создает элемент формы, в который пользователь может ввести данные. Атрибут type указывает тип элемента (в этом случае текстовое поле), атрибут id устанавливает уникальный идентификатор элемента, а атрибут name устанавливает имя элемента, которое используется при отправке формы на сервер.

<button type="button" id="calculateButton">Рассчитать</button>

Этот тег создает кнопку, которая может быть использована для выполнения определенного действия. Атрибут type указывает тип кнопки (в этом случае просто кнопка), атрибут id устанавливает уникальный идентификатор кнопки, а текст внутри тега отображается на кнопке.

<script src="script.js"></script>

Этот тег устанавливает внешнее связь с файлом JavaScript для добавления функциональности странице. Атрибут src указывает путь к файлу.

body {

display: flex;

align-items: center;

justify-content: center;

height: 100vh;

width: 100vw;

font-family: cursive;

}

Этот блок CSS-кода устанавливает стили для тега body. Свойство display устанавливает тип отображения (в этом случае гибкий контейнер), свойства align-items и justify-content выравнивают содержимое контейнера по вертикали и горизонтали соответственно, свойство height устанавливает высоту контейнера, свойство width устанавливает ширину контейнера, а свойство font-family устанавливает шрифт текста.

const field1 = document.getElementById('field1');

Этот код JavaScript получает элемент HTML-документа по его атрибуту id и сохраняет его в переменной field1.

function isNumeric(n) {

return !isNaN(parseFloat(n)) && isFinite(n);

}

vh - это единица измерения высоты, которая равняется 1% высоты области просмотра (viewport height). Единица измерения vw (viewport width) означает, что 1vw равняется 1% ширины области просмотра.

line-height: 1.9rem; - это CSS-свойство, которое устанавливает высоту строки текста. Значение 1.9rem означает, что высота строки будет равняться 1.9 размерам шрифта элемента.

Единица измерения rem (root em) основана на размере шрифта корневого элемента (html или body). Если размер шрифта корневого элемента равен 16 пикселям, то 1rem будет равняться 16 пикселям. Если размер шрифта корневого элемента изменится, то значение rem также изменится пропорционально.

Этот код JavaScript определяет функцию isNumeric, которая проверяет, является ли переданное значение числом. Функция parseFloat преобразует строку в число с плавающей точкой, а функция isNaN проверяет, является ли число NaN (Not a Number). Функция isFinite проверяет, является ли число конечным (не бесконечным и не NaN).

calculateButton.addEventListener('click', calculateCost);

Этот код JavaScript добавляет обработчик события на кнопку расчета стоимости, который вызывает функцию calculateCost при каждом нажатии на кнопку. Функция calculateCost рассчитывает стоимость заказа на основе введенных данных и выводит результат в поле 5.

# Ход работы часть 2

1. Создать HTML и JavaScript, таблица стилей буду прописываться в теге style.
2. Написать структуру HTML страницы в соответствии с заданием.
3. Написать скрипт в файле JvaScript, который получает вводимый текст обрабатывает его и выводит в тег div.

# Код часть 2

HTML

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Задача</title>

<style>

#output {

font-size: 1em;

}

</style>

</head>

<body>

<textarea id="textarea" cols="60" rows="17">Теги не фильтруются <a href="http://sut.ru" >Hacked by Vasya!”)</a> сайт уязвим для атак XSS.</textarea>

<button id="saveButton">Сохранить</button>

<div id="output"></div>

<script src="script.js"></script>

</body>

</html>

JavaScript

document.getElementById('saveButton').addEventListener('click', function() {

let text = document.getElementById('textarea').value;

// удаляем теги из текста

let cleanedText = text.replace(/<[^>]\*>?/gm, '');

// выводим диагностику о наличии тегов в тексте

if (text !== cleanedText) {

alert('В тексте обнаружены теги!');

}

// выводим отформатированный текст в блоке <div>

let firstLetter = cleanedText.charAt(0).toUpperCase();

let restText = cleanedText.slice(1);

let outputText = '<span style="font-size: 2em; color: red;">' + firstLetter + '</span>' + restText;

document.getElementById('output').innerHTML = outputText;

});

# Пояснение к коду часть 2

Этот код создает HTML-страницу с полем для ввода текста, кнопкой сохранения и блоком для вывода отформатированного текста. При нажатии на кнопку сохранения теги будут автоматически удалены из текста, выведена диагностика о наличии тегов в тексте, а первая буква текста станет в два раза больше остальных букв и красного цвета.

1. `<!DOCTYPE html>` - это обязательная декларация типа документа, которая должна быть первой строкой в любом HTML5-документе.

2. `<html>` - это корневой элемент HTML-документа.

3. `<head>` - это элемент, содержащий метаинформацию о документе, такую как кодировка, заголовок страницы и внешние файлы стилей и скриптов.

4. `<meta charset="utf-8">` - это элемент, устанавливающий кодировку документа в UTF-8.

5. `<title>Задача</title>` - это элемент, устанавливающий заголовок страницы.

6. `<style>` - это элемент, содержащий внутренние стили для документа. В этом случае устанавливается размер шрифта для блока `<div>` с идентификатором `output`.

7. `<body>` - это элемент, содержащий видимую часть документа.

8. `<textarea id="textarea" cols="60" rows="17">Теги не фильтруются <a href="http://sut.ru" >Hacked by Vasya!”)</a> сайт уязвим для атак XSS.</textarea>` - это элемент, создающий поле для ввода текста. Атрибуты `cols` и `rows` устанавливают количество столбцов и строк в поле ввода. Внутри элемента находится текст, содержащий теги.

9. `<button id="saveButton">Сохранить</button>` - это элемент, создающий кнопку сохранения.

10. `<div id="output"></div>` - это элемент, создающий блок для вывода отформатированного текста.

11. `<script src="script.js"></script>` - это элемент, подключающий внешний файл JavaScript.

12. `document.getElementById('saveButton').addEventListener('click', function() {...});` - это код JavaScript, добавляющий обработчик события нажатия на кнопку сохранения. При нажатии на кнопку выполняется анонимная функция.

13. `let text = document.getElementById('textarea').value;` - это строка кода создает новую переменную `text`, которая содержит значение текста из поля ввода.

14. `let cleanedText = text.replace(/<[^>]\*>?/gm, '');` - это строка кода удаляет все теги из текста с помощью регулярного выражения.

15. `if (text !== cleanedText) { alert('В тексте обнаружены теги!'); }` - это условный оператор, выводящий диагностику о наличии тегов в тексте.

16. `let firstLetter = cleanedText.charAt(0).toUpperCase();` - это строка кода создает новую переменную `firstLetter`, которая содержит первую букву очищенного текста в верхнем регистре.

17. `let restText = cleanedText.slice(1);` - эта строка кода создает новую переменную `restText`, которая содержит очищенный текст без первой буквы.

18. `let outputText = '<span style="font-size: 2em; color: red;">' + firstLetter + '</span>' + restText;` - эта строка кода создает новую переменную `outputText`, которая содержит отформатированный текст. Первая буква текста заключена в тег `<span>` со стилями `font-size: 2em; color: red;`, которые делают ее в два раза больше остальных букв и красного цвета. Затем первая буква конкатенируется с остальным текстом с помощью оператора `+`.

19. `document.getElementById('output').innerHTML = outputText;` - эта строка кода выводит отформатированный текст в блок `<div>` с идентификатором `output`.

В итоге, этот код создает HTML-страницу с полем для ввода текста, кнопкой сохранения и блоком для вывода отформатированного текста. При нажатии на кнопку сохранения теги будут автоматически удалены из текста, выведена диагностика о наличии тегов в тексте, а первая буква текста станет в два раза больше остальных букв и красного цвета.

# Вывод

В ходе выполнения практической работы был получен опыт работы с HTML тегами form, input, label. Так же изучены основы синтаксиса языка программирования JavaScript и написан скрипт, который получает данные из HTML формы обрабатывает и возражает результат в форму. Во втором задании была сделана форма для редактирования текста с помощью JavaScript, которая решает проблему  HTML инъекции при вводе данных в поле ввода.