МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа №8**

по дисциплине: «Веб-технологии»

на тему: «Объекты клиентских приложений. Обработка событий JavaScript.»

|  |  |
| --- | --- |
| Выполнил | Студент группы 22-ИТ-1  Короткевич П.А. |
|  |  |
| Проверил | Макарычева В.А |

Полоцк 2024

**Цель:** ознакомиться с объектами клиентских приложений, изучить принцип работы обработки событий JavaScript, научиться применять полученные знания на практике.

**Задание 1**

Создайте галерею изображений, в которой основное изображение изменяется при клике на уменьшенный вариант. Используйте делегирование.

Листинг 1

body style="text-align: center;">

<h1>Задание 1</h1>

<div style="border: 1px solid #000; padding: 10px; width: 700px; margin: 0 auto;">

<div class="gallery">

<div class="main-image">

<img src="imag/Bird1.jpg" style="width: 900px; height: 300px;" alt="Main Image">

</div>

<div class="thumbnails">

<img src="imag/BIR1.gif" alt="Bird">

<img src="imag/Bird.jpg" alt="Bird">

<img src="imag/BIR2.gif" alt="Bird">

<img src="imag/Bird3.jpg" alt="Bird">

</div>

</div>

<script>

document.addEventListener("DOMContentLoaded", function() {

const thumbnailsContainer = document.querySelector(".thumbnails");

const mainImage = document.querySelector(".main-image img");

thumbnailsContainer.addEventListener("click", function(event) {

if (event.target.tagName === "IMG") {

const imageUrl = event.target.src;

mainImage.src = imageUrl;

}

});

});

</script>

Этот HTML и JavaScript код создает галерею изображений, где основное изображение меняется при клике на миниатюры. В HTML разметке задается структура галереи: есть заголовок "Задание 1", контейнер для галереи с классом "gallery", внутри которого располагается основное изображение в блоке с классом "main-image" и миниатюры изображений в блоке с классом "thumbnails". В JavaScript части выбирается контейнер с миниатюрами изображений и основное изображение. Затем добавляется обработчик события "click" на контейнере миниатюр. При клике на миниатюру извлекается URL изображения и присваивается свойству "src" основного изображения, таким образом, изменяя основное изображение на выбранное миниатюрное изображение.

На следующем рисунке представлена реализация:



Рисунок 1 – Реализация первого задания

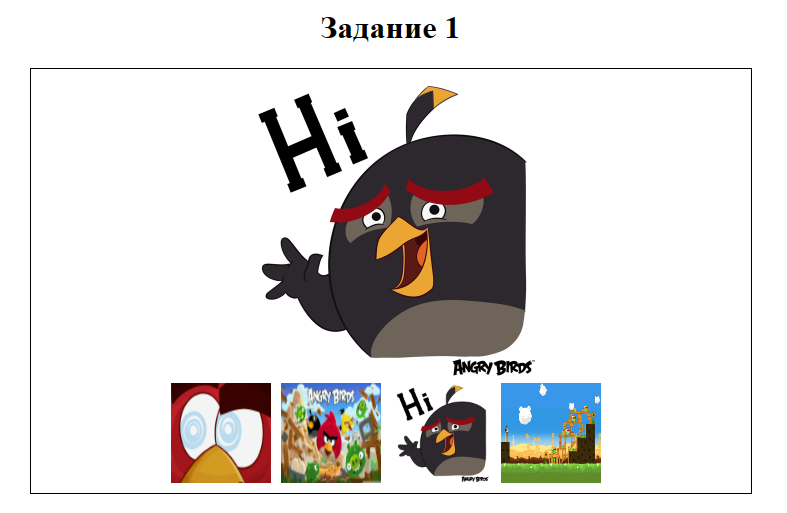


Рисунок 2 – Смена главной картинки на миниатюрную

**Задание 2**

Создайте список, в котором элементы могут быть выделены, как в файловых менеджерах.

При клике на элемент списка выделяется только этот элемент (добавляется класс .selected), отменяется выделение остальных элементов.

Если клик сделан вместе с Ctrl (Cmd для Mac), то выделение переключается на элементе, но остальные элементы при этом не изменяются.

! В этом задании все элементы списка содержат только текст. Без вложенных тегов.

! Предотвратите стандартное для браузера выделение текста при кликах.с

Листинг 2

<h1>Задание 2</h1>

<div style="border: 1px solid #000; padding: 10px; width: 700px; margin: 0 auto;">

<ul id="selectableList">

<li>Выбрать 1</li>

<li>Выбрать 2</li>

<li>Выбрать 3</li>

<li>Выбрать 4</li>

<li>Выбрать 5</li>

</ul>

<button onclick="resetSelection()">Сбросить выделение</button>

<script>

const list = document.getElementById("selectableList");

function resetSelection() {

const selectedItems = list.querySelectorAll(".selected");

selectedItems.forEach(item => {

item.classList.remove("selected");

item.style.backgroundColor = "";

});

}

list.addEventListener("click", function(event) {

const target = event.target;

if (target.tagName === "LI") {

const isShiftPressed = event.shiftKey;

const isCmdPressed = event.metaKey;

const isSelectionToggled = isShiftPressed || isCmdPressed;

if (!isSelectionToggled) {

resetSelection();

}

target.classList.toggle("selected");

if (target.classList.contains("selected")) {

target.style.backgroundColor = "green";

} else {

target.style.backgroundColor = "";

}

}

});

list.addEventListener("mousedown", function(event) {

event.preventDefault();

});

</script>

</div>

Этот HTML и JavaScript код создает интерактивный список элементов, где каждый элемент списка можно выделять при клике на него.

В HTML разметке есть заголовок "Задание 2" и контейнер <div>, который содержит список <ul> с идентификатором "selectableList". Каждый элемент списка <li> содержит текст "Item N", где N - порядковый номер элемента списка. Также есть кнопка "Сбросить выделение", при клике на которую сбрасывается выделение всех элементов списка.

В JavaScript части обрабатываются следующие события: при клике на элемент списка добавляется или удаляется класс "selected", что позволяет выделять или снимать выделение. Если при клике удерживается клавиша Shift или Cmd (для Mac), то выделение элементов сохраняется. При клике на кнопку "Сбросить выделение" вызывается функция resetSelection(), которая снимает выделение со всех элементов списка. Также добавлен обработчик события "mousedown", который предотвращает стандартное выделение текста при клике на элементы списка.

На следующем рисунке представлена реализация:

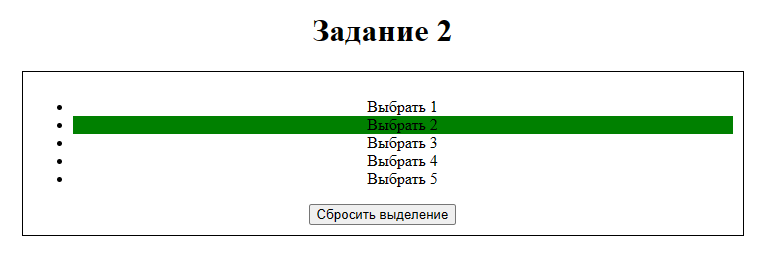


Рисунок 3 – Выбор одного элемента

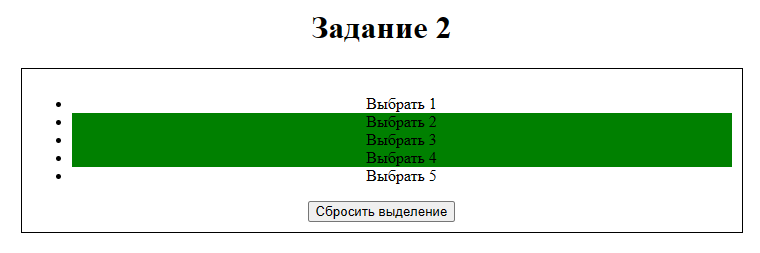


Рисунок 4 – Выбор нескольких элементов

**Задание 3**

Создайте функцию runOnKeys(func, code1, code2, ... code\_n), которая запускает func при одновременном нажатии клавиш с кодами code1, code2, …, code\_n.

Например, код ниже выведет alert при одновременном нажатии клавиш “Q” и “W” (в любом регистре, в любой раскладке)

Листинг 3

<h1>Задание 3</h1>

<script>

let isTaskRunning = true;

function runOnKeys(func, stopKey, ...codes) {

let pressedKeys = new Set();

function checkKeys() {

for (const code of codes) {

if (!pressedKeys.has(code)) {

Продолжение листинга 3

return false;

}

}

return true;

}

const keyDownHandler = function(event) {

pressedKeys.add(event.code);

if (checkKeys()) {

func();

}

if (event.code === stopKey) {

stopTask();

}

};

const keyUpHandler = function(event) {

pressedKeys.delete(event.code);

};

document.addEventListener("keydown", keyDownHandler);

document.addEventListener("keyup", keyUpHandler);

function stopTask() {

document.removeEventListener("keydown", keyDownHandler);

document.removeEventListener("keyup", keyUpHandler);

isTaskRunning = false;

}

}

// Пример использования функции runOnKeys

runOnKeys(

() => alert("Функция выполнена при нажатии клавиш"),

"KeyS",

"KeyA",

"KeyB",

"KeyC"

);

// Функция для остановки выполнения задания по нажатию кнопки

function stopTask() {

if (isTaskRunning) {

alert("Задание 3 остановлено");

isTaskRunning = false;

}

}

</script>

В данном коде реализована функция runOnKeys, которая позволяет выполнять определенную функцию при одновременном нажатии определенных клавиш. При вызове функции runOnKeys, в качестве аргументов передается функция func, которая будет выполнена при нажатии клавиш, а также коды клавиш и одна особая клавиша, после нажатия которой выполнение функции прекратится.

Внутри функции runOnKeys создается множество pressedKeys, в котором хранятся коды нажатых клавиш. Затем определяются обработчики событий keyDownHandler и keyUpHandler, которые добавляют и удаляют коды клавиш из множества pressedKeys при нажатии и отпускании клавиш соответственно.

Также в keyDownHandler проверяется, были ли нажаты все клавиши из списка codes с помощью функции checkKeys(). Если все клавиши нажаты, вызывается функция func(). Если нажата клавиша stopKey, вызывается функция stopTask(), которая останавливает выполнение задания.

После этого, с помощью методов addEventListener добавляются обработчики событий "keydown" и "keyup" для документа.

Также в коде определена функция stopTask(), которая выводит сообщение остановки задания и устанавливает флаг isTaskRunning в false, если задание еще не было остановлено.

На следующем рисунке представлена реализация:

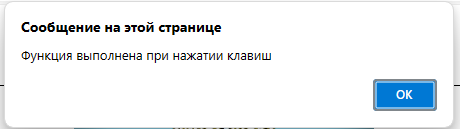


Рисунок 5 – Нажатие на A, B, C

**Задание 4**

Создайте интерфейс, позволяющий ввести сумму банковского вклада и процент, а затем рассчитать, какая это будет сумма через заданный промежуток времени.

Любое изменение введённых данных должно быть обработано немедленно.

Листинг 4

<h1>Задание 4</h1>

<div style="border: 1px solid #000; padding: 10px; width: 700px; margin: 0 auto;">

<div>

<table style="margin: 0 auto;">

<tr>

<td><label for="deposit">Сумма вклада:</label></td>

<td><input type="number" id="deposit" min="0" step="0.01" value="0"></td>

</tr>

<tr>

<td><label for="interestRate">Процент:</label></td>

<td><input type="number" id="interestRate" min="0" max="100" step="0.01" value="0"></td>

</tr>

<tr>

<td><label for="years">Количество лет:</label></td>

<td><input type="number" id="years" min="0" step="1" value="0"></td>

</tr>

<tr>

<td colspan="2"><button onclick="calculateDeposit()">Рассчитать</button></td>

</tr>

<tr>

<td colspan="2"><div id="depositResult"></div></td>

</tr>

</table>

</div>

<script>

function calculateDeposit() {

const depositAmount = parseFloat(document.getElementById('deposit').value);

const interestRate = parseFloat(document.getElementById('interestRate').value) / 100;

Продолжение листинга 4

const years = parseInt(document.getElementById('years').value);

const resultElement = document.getElementById('depositResult');

const initialAmountElement = document.getElementById('initialAmount');

const finalAmountElement = document.getElementById('finalAmount');

if (isNaN(depositAmount) || isNaN(interestRate) || isNaN(years)) {

resultElement.textContent = 'Пожалуйста, введите корректные значения.';

return;

}

const result = Math.round(depositAmount \* (1 + interestRate \* years));

resultElement.textContent = `Через ${years} лет сумма вклада будет: ${result} рублей`;

initialAmountElement.textContent = depositAmount;

finalAmountElement.textContent = result;

}

</script>

В этом коде создается форма для расчета вклада. Внутри блока <div> находится таблица, содержащая поля для ввода данных о сумме вклада (deposit), процентной ставке (interestRate) и сроке вклада в годах (years). Кнопка "Рассчитать" вызывает функцию calculateDeposit() при клике на нее.

Функция calculateDeposit() извлекает значения из полей ввода формы, преобразует их в числа и выполняет расчет по формуле для сложного процента. Результат расчета выводится в элемент <div> с идентификатором depositResult. Если хотя бы одно из введенных значений не является числом, выводится сообщение об ошибке в этом элементе.

Кроме того, в функции calculateDeposit() также устанавливаются значения начальной (initialAmountElement) и конечной (finalAmountElement) сумм вклада для последующего отображения на странице.

На следующем рисунке представлена реализация:

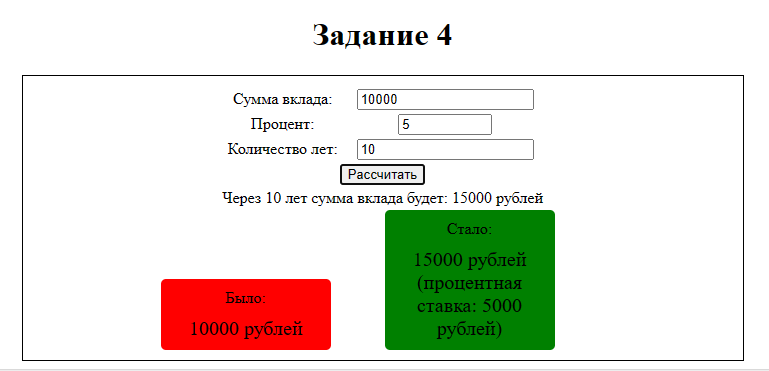


Рисунок 6 – Реализация четвертого задания

**Вывод:** Изучение объектов клиентских приложений и обработки событий JavaScript является ключевым для разработчиков веб-приложений. Объекты, такие как document, window, form, location, event, предоставляют доступ к различным аспектам веб-страницы и интерактивным возможностям. Понимание работы событий и их обработки позволяет создавать динамические и отзывчивые пользовательские интерфейсы.