

Реализация интерактивной временной шкалы событий с использованием современных веб-технологий

Екимова Яна Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается методика создания интерактивной временной шкалы с применением технологий HTML5, CSS3 и JavaScript. Разработанное решение позволяет визуализировать хронологические данные с поддержкой динамического добавления событий, валидации ввода и адаптивного отображения. Материал предназначен для преподавателей и студентов, заинтересованных в применении современных технологий в обучении и стремящихся сделать процесс образовательного взаимодействия более эффективным и увлекательным.

Ключевые слова: временная шкала, HTML5, CSS, JavaScript, веб-визуализация, адаптивный дизайн.

Implementation of an interactive timeline of events using modern web technologies

Ekimova Yana Sergeevna

Sholom Aleichem Priamurskiy State University

Student

Abstract

The article discusses a technique for creating an interactive timeline using HTML5, CSS3, and JavaScript technologies. The developed solution allows you to visualize chronological data with support for dynamic event addition, input validation, and adaptive display. The material is intended for teachers and students interested in using modern technologies in teaching and striving to make the process of educational interaction more effective and exciting.

Keywords: timeline, HTML5, CSS, JavaScript, web visualization, responsive design.

1 Введение

1.1 Актуальность

Современные образовательные и научные проекты активно используют цифровые технологии для повышения эффективности подачи информации, улучшения её восприятия и вовлеченности аудитории. Одним из перспективных инструментов в этой сфере являются интерактивные временные шкалы, которые позволяют наглядно представить

хронологические данные, упрощая анализ исторических, социальных и политических процессов.

В контексте обучения и исследовательской работы визуализация временных данных играет ключевую роль, помогая структурировать информацию, выявлять закономерности и устанавливать причинно-следственные связи. Использование технологий HTML5, CSS3 и JavaScript открывает широкие возможности для создания динамических, адаптивных и удобных в использовании веб-приложений, способных работать на различных устройствах.

1.2 Обзор исследований

В статье рассматривается организация подготовки педагогических работников к использованию технологий HTML5 для разработки электронных образовательных ресурсов в формате веб-приложений. Отмечена актуальность использования мультимедийных интерактивных электронных образовательных ресурсов в современной школе и обоснована эффективность разработки подобных электронных образовательных ресурсов именно в формате офлайновых или онлайновых веб-приложений. [1]. Показала процесс создания и использования мультимедийных интерактивных плакатов по информатике К.А. Цилинская. Представила опыт разработки мультимедийных интерактивных плакатов средствами языка HTML5 и Интернет-сервисов [2]. Я. А. Королюк, Н. Ю. Куликова показали опыт разработки и использования интерактивных образовательных ресурсов на основе html5 для онлайн-обучения школьников информатике [3]. В работе рассмотрели возможности языка разметки гипертекста HTML5 для создания макета электронного учебника Д.М. Размологов, А.С. Панина, П.В. Захаров. Привели разработанный макет, а также часть таблицы стилей для дизайна макета [4]. И.С. Шегаев, А.Г. Полозов разработали новейший политологический глоссарий: основные категории, понятия, термины [5].

1.3 Цель исследования

Целью исследования является создание интерактивной временной шкалы событий с использованием языка HTML5.

2 Материалы и методы

Для создания приложения используется язык программирования HTML5. В качестве среды разработки используется текстовый редактор Notepad++, оптимизированным для языков программирования.

3 Результаты

Открываем Notepad++ и создаём файл index.html, для этого в меню выбираем файл → новый. Далее нажимаем сочетание клавиш "Ctrl+S" и в поле "Имя файла" вводим название файла "index" с расширением .html, сохраняем данный файл на рабочем столе.

Создадим основную структуру HTML-документа (рис.1), для создания веб-страницы. Документ начинается со стандартной разметки (DOCTYPE, html, head, body), где в секции <head> указаны метаданные (кодировка UTF-8, viewport) и заголовок страницы. В <body> размещены: заголовок первого уровня, контейнер временной шкалы с центральной осью (timeline-line) и областью событий (timeline-events), кнопка добавления элементов (+) и скрытое модальное окно с формой (eventForm) для ввода даты, названия и описания. Код семантически организован: использует контейнеры для визуальной шкалы (timeline-container, timeline-line), кнопку с иконкой плюса (+), модальное окно (modal) для взаимодействия с пользователем и форму (eventForm), которая в дальнейшем будет содержать поля для ввода даты, названия и описания событий. Структура готова для подключения CSS-стилей и JavaScript-логики управления событиями, валидации данных и сохранения в localStorage.

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Интерактивная временная шкала</title>
  <!-- Стили и скрипты -->
</head>
<body>
  <h1>Интерактивная временная шкала</h1>

  <!-- Контейнер временной шкалы -->
  <div class="timeline-container">
    <div class="timeline-line"></div>
    <div id="timeline-events"></div>
  </div>

  <!-- Кнопка добавления -->
  <button class="add-event-btn" id="addEventBtn">+ Добавить событие</button>

  <!-- Модальное окно -->
  <div id="eventModal" class="modal">
    <div class="modal-content">
      <span class="close-btn">&times;</span>
      <h2>Добавить новое событие</h2>
      <form class="event-form" id="eventForm">
        <!-- Поля формы -->
      </form>
    </div>
  </div>

  <!-- JavaScript -->
  <script>/* Логика приложения */</script>
</body>
```

</html>

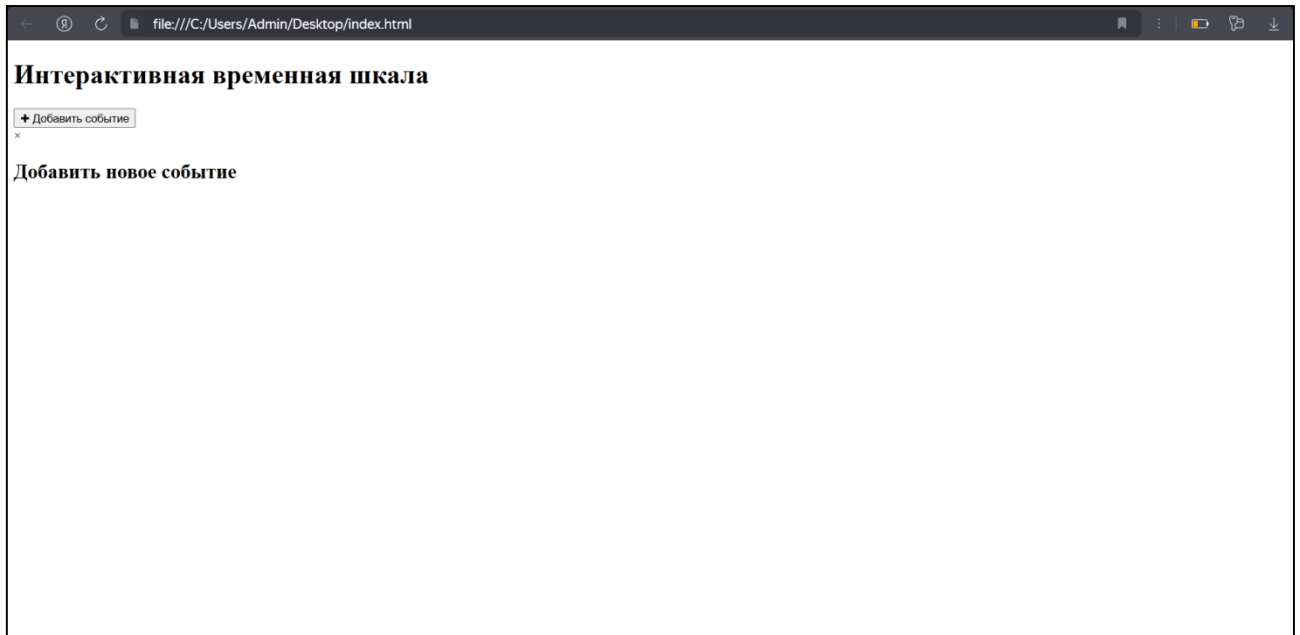


Рисунок 1. Основная структура HTML-документа

Далее добавим блок CSS-стилей, который используется для оформления веб-страницы (рис.2). CSS стили создают адаптивную временную шкалу с чередующимися карточками событий (позиционированными слева/справа от центральной синей линии), интерактивными элементами (кнопки добавления/редактирования с анимацией масштаба и трансформации), модальным окном на затемнённом фоне с анимацией открытия, валидацией форм, а также мобильной адаптацией, где карточки занимают 85% ширины, а временная линия смещается к краю, сохраняя визуальную иерархию через тени, цвета и плавные переходы.

Данный код пропишем на следующей строчке после тега `<title>Интерактивная временная шкала</title>` в секции `<head>` HTML-документа.

```
<style>
  body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    line-height: 1.4;
    color: #333;
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
    background-color: #f8fafc;
  }

  h1 {
    text-align: center;
    color: #2c3e50;
    margin-bottom: 30px;
  }
```

```
        font-size: 2.2em;
        letter-spacing: -0.5px;
    }

    .timeline-container {
        position: relative;
        max-width: 1000px;
        margin: 40px auto;
    }

    .timeline-line {
        position: absolute;
        width: 4px;
        background-color: #3498db;
        top: 0;
        bottom: 0;
        left: 50%;
        transform: translateX(-50%);
        z-index: 0;
        border-radius: 2px;
    }

    .timeline-event {
        padding: 15px 20px;
        position: relative;
        width: 45%;
        background-color: white;
        border-radius: 8px;
        box-shadow: 0 3px 10px rgba(0, 0, 0, 0.1);
        margin-bottom: 20px;
        transition: all 0.3s ease;
        opacity: 0;
        transform: translateY(10px);
    }

    .timeline-event:nth-child(odd) {
        left: 0;
    }

    .timeline-event:nth-child(even) {
        left: 55%;
    }

    .timeline-event::after {
        content: ";";
        position: absolute;
        width: 16px;
        height: 16px;
        background-color: #3498db;
        border-radius: 50%;
        top: 20px;
        z-index: 1;
```

```
        border: 3px solid white;
    }

    .timeline-event:nth-child(odd)::after {
        right: -10px;
    }

    .timeline-event:nth-child(even)::after {
        left: -10px;
    }

    .event-date {
        font-weight: 700;
        color: #3498db;
        margin-bottom: 8px;
        font-size: 0.95em;
        letter-spacing: 0.5px;
    }

    .event-title {
        font-size: 1.15em;
        margin-bottom: 8px;
        color: #2c3e50;
        font-weight: 600;
    }

    .event-description {
        color: #666;
        margin-bottom: 8px;
        font-size: 0.95em;
        line-height: 1.5;
    }

    .event-actions {
        display: flex;
        justify-content: flex-end;
        gap: 10px;
        margin-top: 12px;
    }

    .edit-btn, .delete-btn {
        background: none;
        border: none;
        cursor: pointer;
        padding: 5px;
        transition: transform 0.2s;
    }

    .edit-btn:hover, .delete-btn:hover {
        transform: scale(1.1);
    }
```

```
.timeline-event:hover {
  transform: translateY(-3px);
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.15);
}

.add-event-btn {
  display: block;
  margin: 30px auto;
  padding: 12px 30px;
  font-size: 1.1em;
  background-color: #3498db;
  color: white;
  border: none;
  border-radius: 25px;
  cursor: pointer;
  transition: all 0.3s ease;
  box-shadow: 0 4px 6px rgba(52, 152, 219, 0.2);
}

.add-event-btn:hover {
  background-color: #2980b9;
  transform: translateY(-2px);
  box-shadow: 0 6px 12px rgba(52, 152, 219, 0.3);
}

.modal {
  display: none;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0,0,0,0.5);
  justify-content: center;
  align-items: center;
  z-index: 1000;
}

.modal-content {
  background-color: #fff;
  padding: 25px;
  border-radius: 12px;
  position: relative;
  width: 90%;
  max-width: 500px;
  animation: modalOpen 0.3s ease;
}

@keyframes modalOpen {
  from { transform: scale(0.95); opacity: 0; }
  to { transform: scale(1); opacity: 1; }
}
```

```
.close-btn {
  position: absolute;
  top: 15px;
  right: 20px;
  font-size: 28px;
  cursor: pointer;
  color: #666;
  transition: color 0.2s;
}

.close-btn:hover {
  color: #333;
}

.event-form {
  display: flex;
  flex-direction: column;
  gap: 15px;
}

.event-form input,
.event-form textarea {
  padding: 12px;
  border: 2px solid #eee;
  border-radius: 8px;
  font-size: 1em;
  transition: border-color 0.3s;
}

.event-form input:focus,
.event-form textarea:focus {
  outline: none;
  border-color: #3498db;
}

.submit-btn {
  padding: 12px;
  background-color: #3498db;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-size: 1em;
  transition: background-color 0.3s;
}

.submit-btn:hover {
  background-color: #2980b9;
}

.error-message {
```

```
        color: #e74c3c;
        font-size: 0.85em;
        display: none;
        margin: -10px 0 5px;
    }

    .invalid {
        border-color: #e74c3c !important;
    }

    @media screen and (max-width: 768px) {
        .timeline-event {
            width: 85%;
            margin-bottom: 25px;
        }

        .timeline-event:nth-child(even) {
            left: 15%;
        }

        .timeline-line {
            left: 30px;
        }

        .timeline-event::after {
            width: 14px;
            height: 14px;
            top: 18px;
        }

        .add-event-btn {
            width: 100%;
            max-width: 300px;
            padding: 15px;
        }
    }
</style>
```

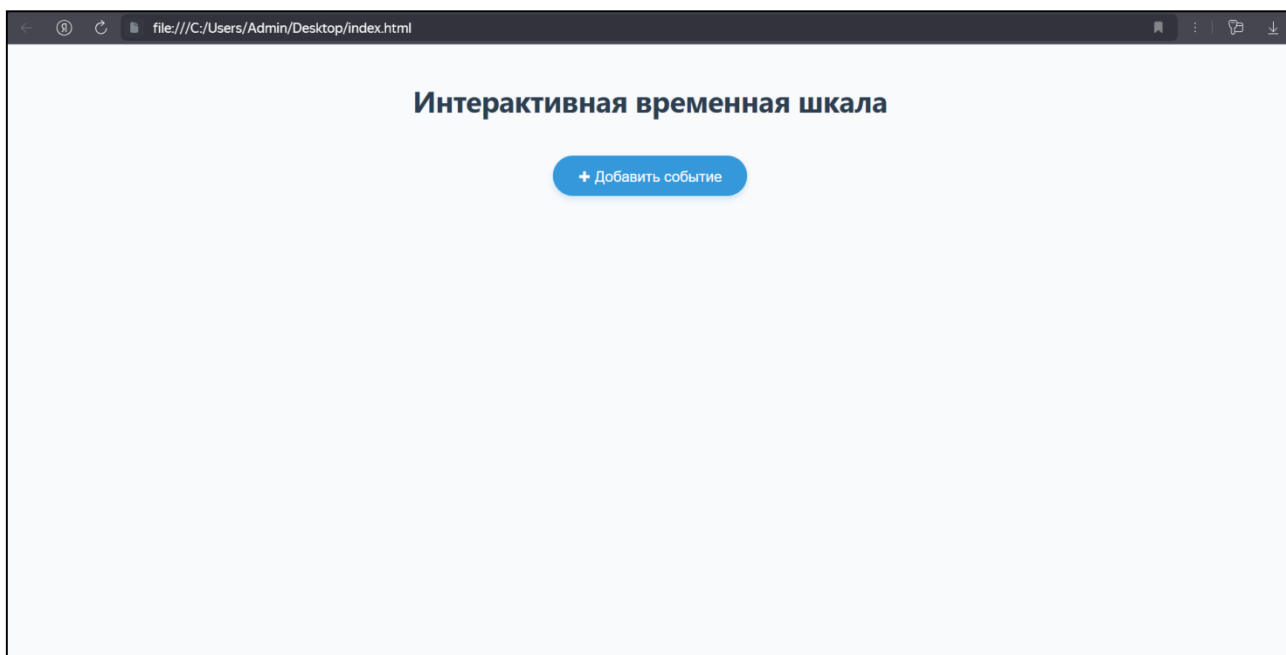


Рисунок 2. Добавление CSS-стилей

Приступим к созданию HTML-формы для добавления событий с полями ввода даты (с валидацией формата ДД.ММ.ГГГГ через регулярное выражение), названию и описанию события, включая блок для вывода ошибок валидации (dateError) и кнопку отправки данных, где все поля помечены как обязательные (required).

Вставляем данный код после открывающего тега `<form>`:

```
<input type="text" id="eventDate" placeholder="Дата (ДД.ММ.ГГГГ)" required
      pattern="^(0[1-9]|[12][0-9]|3[01])\.(0[1-9]|1[0-2])\.\d{4}$">
  <div id="dateError" class="error-message"></div>
  <input type="text" id="eventTitle" placeholder="Название события" required>
  <textarea id="eventDescription" placeholder="Описание события" rows="4"
required></textarea>
  <button type="submit" class="submit-btn">Сохранить событие</button>
```

Далее напишем структуру данных и инициализацию: Данный код начинает работу с загрузки массива событий events из localStorage (или инициализации примерами при первом запуске). Объявляются DOM-элементы: контейнер временной шкалы, кнопка добавления, модальное окно и форма. Переменная editingIndex отслеживает индекс редактируемого события.

Вставьте данный код после закрывающего тега `</div>` и после открывающего тега `<script>`:

```
let events = JSON.parse(localStorage.getItem('timelineEvents')) || [
  {
    date: "20.07.1969",
    title: "Первый человек на Луне",
```

```

        description: "Нил Армстронг становится первым человеком, ступившим на
поверхность Луны."
      },
      {
        date: "09.11.1989",
        title: "Падение Берлинской стены",
        description: "Символическое окончание Холодной войны и начало
объединения Германии."
      },
      {
        date: "06.08.1991",
        title: "Создание Всемирной паутины",
        description: "Тим Бернерс-Ли публикует первый веб-сайт, положив начало
WWW."
      }
    ];

```

Теперь сделаем рендеринг событий: Функция `displayEvents()` динамически создает HTML-элементы для каждого события, используя шаблон с датой, заголовком, описанием и кнопками действий. Для анимации используется `setTimeout` с задержкой, зависящей от индекса события, создавая эффект последовательного появления. Кнопки удаления и редактирования привязываются к обработчикам: удаление через `splice()` с подтверждением, редактирование – заполнение формы существующими данными.

Вставьте данный код в продолжение тега `<script>`:

```

const timelineEvents = document.getElementById('timeline-events');
const addEventBtn = document.getElementById('addEventBtn');
const eventModal = document.getElementById('eventModal');
const closeBtn = document.querySelector('.close-btn');
const eventForm = document.getElementById('eventForm');
let editingIndex = null;
function displayEvents() {
  timelineEvents.innerHTML = "";

  events.forEach((event, index) => {
    const eventElement = document.createElement('div');
    eventElement.className = 'timeline-event';
    eventElement.innerHTML = `
      <div class="event-date">${event.date}</div>
      <div class="event-title">${event.title}</div>
      <div class="event-description">${event.description}</div>
      <div class="event-actions">
        <button class="edit-btn">✎ </button>
        <button class="delete-btn">🗑 </button>
      </div>
    `;

    eventElement.querySelector('.delete-btn').addEventListener('click', () => {
      if (confirm('Удалить это событие?')) {

```

```

        events.splice(index, 1);
        saveAndUpdate();
    }
});

eventElement.querySelector('.edit-btn').addEventListener('click', () => {
    editingIndex = index;
    document.getElementById('eventDate').value = event.date;
    document.getElementById('eventTitle').value = event.title;
    document.getElementById('eventDescription').value = event.description;
    eventModal.querySelector('h2').textContent = 'Редактировать событие';
    eventModal.style.display = 'flex';
});

setTimeout(() => {
    eventElement.style.opacity = '1';
    eventElement.style.transform = 'translateY(0)';
}, index * 100);
timelineEvents.appendChild(eventElement);
});
}

```

Далее займемся валидацией и логикой формы: Функция `validateDate()` проверяет дату в два этапа: регулярное выражение для формата ДД.ММ.ГГГГ и валидация реальности даты через объект `Date`. При отправке формы (`submit`) данные либо обновляют существующее событие (если `editingIndex` установлен), либо добавляются как новые. Ошибки валидации отображаются в блоке `dateError`.

Вставьте данный код в продолжение тега `<script>`:

```

function validateDate(dateString) {
    const pattern = /^(0[1-9]|[12][0-9]|3[01])\.(0[1-9]|1[0-2])\.d{4}$/;
    if (!pattern.test(dateString)) return false;
    const [day, month, year] = dateString.split('.').map(Number);
    const date = new Date(year, month - 1, day);
    return date.getDate() === day &&
        date.getMonth() === month - 1 &&
        date.getFullYear() === year;
}

```

Работа с состоянием: Функция `saveAndUpdate()` сохраняет массив `events` в `localStorage` и перерисовывает интерфейс. Модальное окно управляется через `display: flex/none`, сброс формы выполняется в `closeModal()`. Обработчики закрытия модалки включают клик по крестику, клик вне контента и успешный сабмит формы.

Вставьте данный код в продолжение тега `<script>`:

```

function saveAndUpdate() {
    localStorage.setItem('timelineEvents', JSON.stringify(events));
    displayEvents();
}

```

```

    }
    function closeModal() {
        eventModal.style.display = 'none';
        editingIndex = null;
        eventForm.reset();
        document.getElementById('dateError').style.display = 'none';
        document.getElementById('eventDate').classList.remove('invalid');
    }

    addEventBtn.addEventListener('click', () => {
        eventModal.querySelector('h2').textContent = 'Добавить новое событие';
        eventModal.style.display = 'flex';
    });
    closeBtn.addEventListener('click', closeModal);
    window.addEventListener('click', (e) => e.target === eventModal && closeModal());
    eventForm.addEventListener('submit', (e) => {
        e.preventDefault();
        const dateInput = document.getElementById('eventDate');
        const dateError = document.getElementById('dateError');
        if (!validateDate(dateInput.value)) {
            dateInput.classList.add('invalid');
            dateError.textContent = dateInput.value ?
                'Некорректная дата (например, 30.02 не существует)' :
                'Пожалуйста, введите дату в формате ДД.ММ.ГГГГ';
            dateError.style.display = 'block';
            return;
        }
        const newEvent = {
            date: dateInput.value,
            title: document.getElementById('eventTitle').value.trim(),
            description: document.getElementById('eventDescription').value.trim()
        };

        if (editingIndex !== null) {
            events[editingIndex] = newEvent;
        } else {
            events.push(newEvent);
        }
        closeModal();
        saveAndUpdate();
    });

```

Теперь займемся динамическими эффектами: При скролле (`window.addEventListener('scroll')`) проверяется позиция карточек событий относительно видимой области окна. Элементы плавно проявляются (`opacity: 1`), когда их верхняя граница достигает 80% высоты экрана, создавая эффект ленивой загрузки. Анимация модального окна реализована через CSS-ключ `modalOpen` с трансформацией масштаба.

Вставьте данный код в продолжение тега `<script>`:

```

document.addEventListener('DOMContentLoaded', displayEvents);
window.addEventListener('scroll', () => {

```

```

document.querySelectorAll('.timeline-event').forEach(event => {
  const rect = event.getBoundingClientRect();
  event.style.opacity = rect.top < window.innerHeight * 0.8 ? '1' : '0';
});
});
});

```

```

296 <script>
297 let events = JSON.parse(localStorage.getItem('timelineEvents')) || [
298   {
299     date: "20.07.1969",
300     title: "Первый человек на Луне",
301     description: "Нил Армстронг становится первым человеком, ступившим на поверхность Луны."
302   },
303   {
304     date: "09.11.1989",
305     title: "Падение Берлинской стены",
306     description: "Символическое окончание Холодной войны и начало объединения Германии."
307   },
308   {
309     date: "06.08.1991",
310     title: "Создание Всемирной паутины",
311     description: "Тим Бернерс-Ли публикует первый веб-сайт, положив начало WWW."
312   }
313 ];
314 const timelineEvents = document.getElementById('timeline-events');
315 const addEventBtn = document.getElementById('addEventBtn');
316 const eventModal = document.getElementById('eventModal');
317 const closeBtn = document.querySelector('.close-btn');
318 const eventForm = document.getElementById('eventForm');
319 let editingIndex = null;
320
321 function displayEvents() {
322   timelineEvents.innerHTML = '';
323
324   events.forEach((event, index) => {
325     const eventElement = document.createElement('div');
326     eventElement.className = 'timeline-event';
327     eventElement.innerHTML = `
328       <div class="event-date">${event.date}</div>
329       <div class="event-title">${event.title}</div>
330       <div class="event-description">${event.description}</div>
331       <div class="event-actions">
332         <button class="edit-btn">✎</button>
333         <button class="delete-btn">🗑️</button>
334       </div>
335     `;
336
337     eventElement.querySelector('.delete-btn').addEventListener('click', () => {
338       if (confirm('Удалить это событие?')) {
339         events.splice(index, 1);
340         saveAndUpdate();
341       }
342     });
343
344     eventElement.querySelector('.edit-btn').addEventListener('click', () => {
345       editingIndex = index;
346       document.getElementById('eventDate').value = event.date;
347       document.getElementById('eventTitle').value = event.title;
348       document.getElementById('eventDescription').value = event.description;

```

Рисунок 3. Код раздела <script>

```

344 eventElement.querySelector('.edit-btn').addEventListener('click', () => {
345   editingIndex = index;
346   document.getElementById('eventDate').value = event.date;
347   document.getElementById('eventTitle').value = event.title;
348   document.getElementById('eventDescription').value = event.description;
349   eventModal.querySelector('h2').textContent = 'Редактировать событие';
350   eventModal.style.display = 'flex';
351 });
352
353 setTimeout(() => {
354   eventElement.style.opacity = '1';
355   eventElement.style.transform = 'translateY(0)';
356 }, index * 100);
357
358 timelineEvents.appendChild(eventElement);
359 });
360
361 function validateDate(dateString) {
362   const pattern = /^(0[1-9]|[12][0-9]|3[01])\.(0[1-9]|1[0-2])\.d{4}$/;
363   if (!pattern.test(dateString)) return false;
364
365   const [day, month, year] = dateString.split('.').map(Number);
366   const date = new Date(year, month - 1, day);
367   return date.getDate() === day &&
368     date.getMonth() === month - 1 &&
369     date.getFullYear() === year;
370 }
371
372 function saveAndUpdate() {
373   localStorage.setItem('timelineEvents', JSON.stringify(events));
374   displayEvents();
375 }
376
377 function closeModal() {
378   eventModal.style.display = 'none';
379   editingIndex = null;
380   eventForm.reset();
381   document.getElementById('dateError').style.display = 'none';
382   document.getElementById('eventDate').classList.remove('invalid');
383 }
384
385 addEventBtn.addEventListener('click', () => {
386   eventModal.querySelector('h2').textContent = 'Добавить новое событие';
387   eventModal.style.display = 'flex';
388 });
389
390 closeBtn.addEventListener('click', closeModal);
391 window.addEventListener('click', (e) => e.target === eventModal && closeModal());
392
393 eventForm.addEventListener('submit', (e) => {
394   e.preventDefault();
395   const dateInput = document.getElementById('eventDate');
396   const dateError = document.getElementById('dateError');

```

Рисунок 4. Код раздела <script>

```

395   const dateInput = document.getElementById('eventDate');
396   const dateError = document.getElementById('dateError');
397
398   if (!validateDate(dateInput.value)) {
399     dateInput.classList.add('invalid');
400     dateError.textContent = dateInput.value ?
401       'Некорректная дата (например, 30.02 не существует)' :
402       'Пожалуйста, введите дату в формате ДД.ММ.ГГГГ';
403     dateError.style.display = 'block';
404     return;
405   }
406
407   const newEvent = {
408     date: dateInput.value,
409     title: document.getElementById('eventTitle').value.trim(),
410     description: document.getElementById('eventDescription').value.trim()
411   };
412
413   if (editingIndex !== null) {
414     events[editingIndex] = newEvent;
415   } else {
416     events.push(newEvent);
417   }
418
419   closeModal();
420   saveAndUpdate();
421 }
422
423 document.addEventListener('DOMContentLoaded', displayEvents);
424 window.addEventListener('scroll', () => {
425   document.querySelectorAll('.timeline-event').forEach(event => {
426     const rect = event.getBoundingClientRect();
427     event.style.opacity = rect.top < window.innerHeight * 0.8 ? '1' : '0';
428   });
429 });
430 </script>
431 </body>
432 </html>
433

```

Рисунок 5. Код раздела <script>

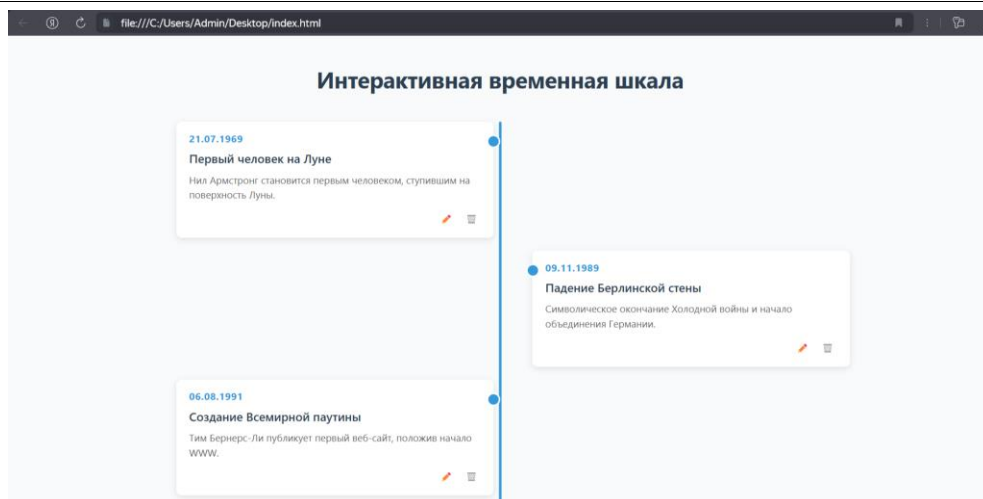


Рисунок 6. Главная страница

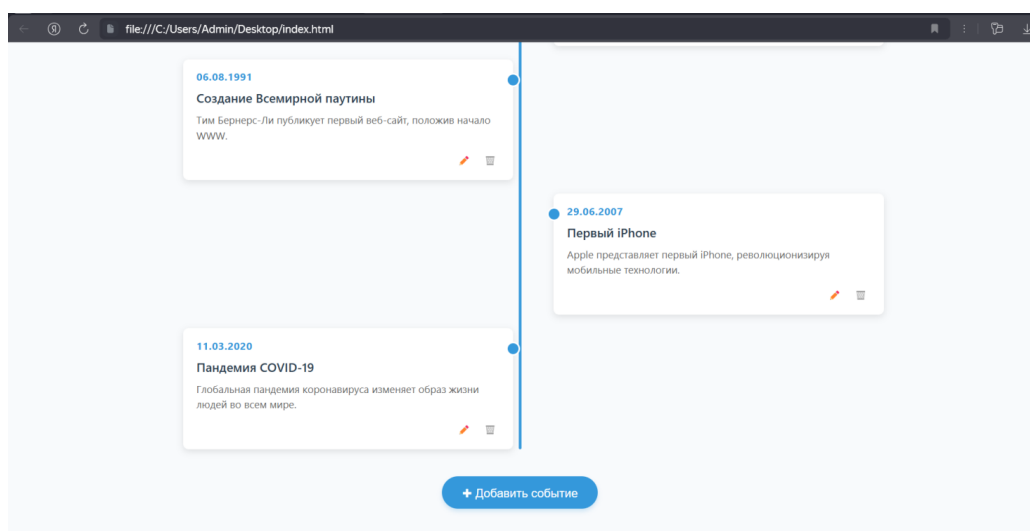


Рисунок 7. Главная страница

Нажимаем на кнопку «Добавить событие», заполняем необходимые данные и сохраняем новое событие:

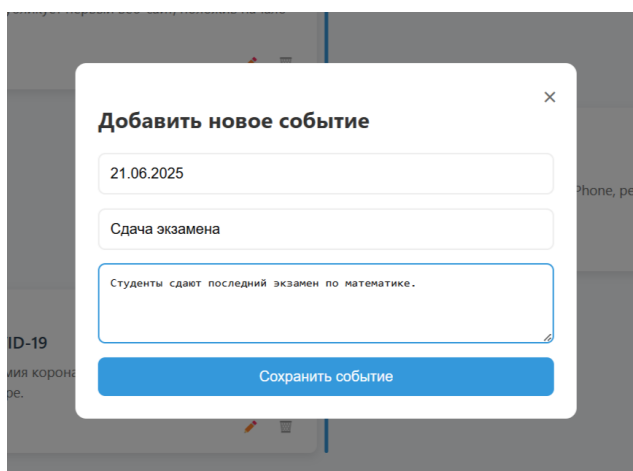
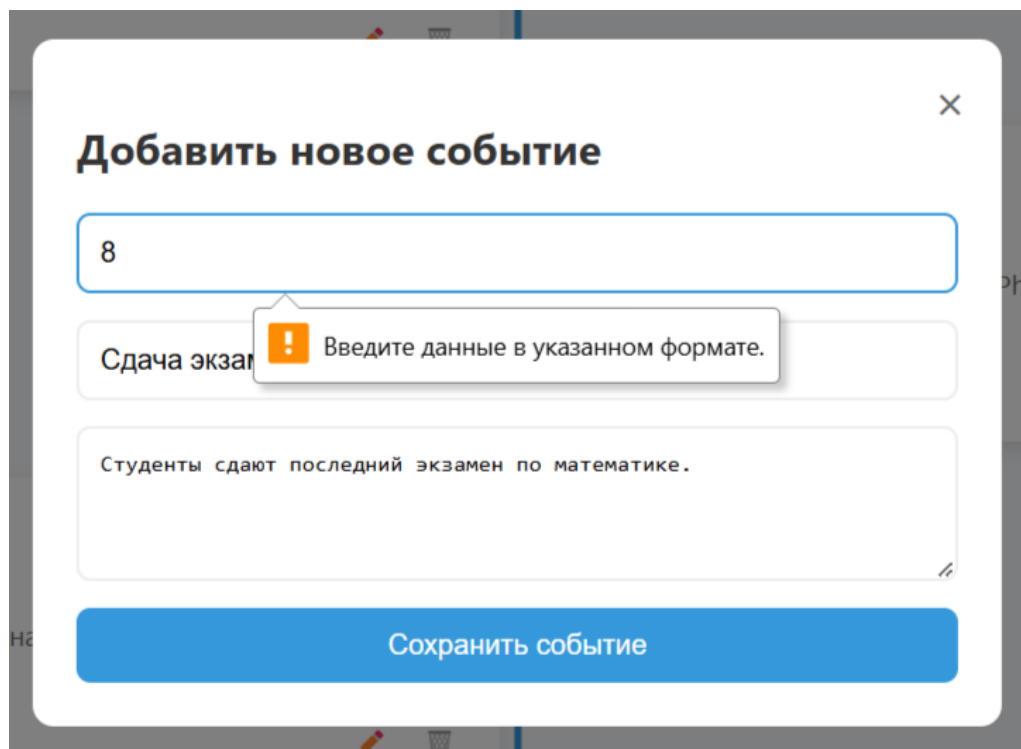


Рисунок 8. Добавление нового события

Если дата введена в неверном формате, система отображает сообщение об ошибке с требованием повторить ввод в указанном формате (например: ДД.ММ.ГГГГ).



The screenshot shows a modal window titled "Добавить новое событие" (Add new event). It contains a date input field with the value "8". Below the input field, a tooltip with an orange exclamation mark icon displays the message "Введите данные в указанном формате." (Enter data in the specified format.). The form also includes a text area with the placeholder text "Сдача экзамена" (Exam taking) and a description field containing the text "Студенты сдают последний экзамен по математике." (Students take the final exam in mathematics.). At the bottom of the form is a blue button labeled "Сохранить событие" (Save event).

Рисунок 9. Добавление нового события

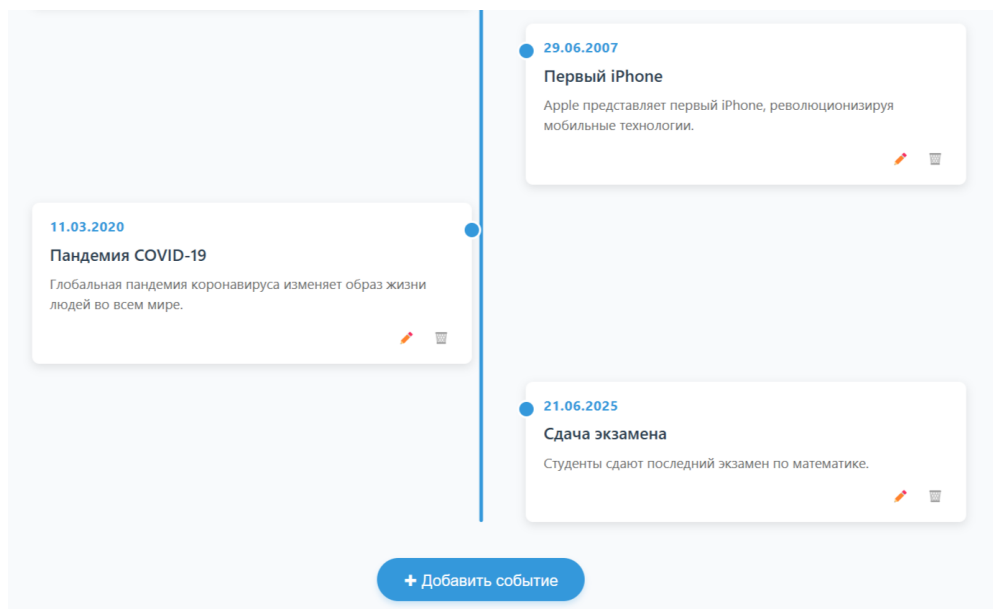


Рисунок 10. Созданное событие «Сдача экзамена»

Если необходимо изменить описание, то нужно нажать на кнопку «» (рис.11).

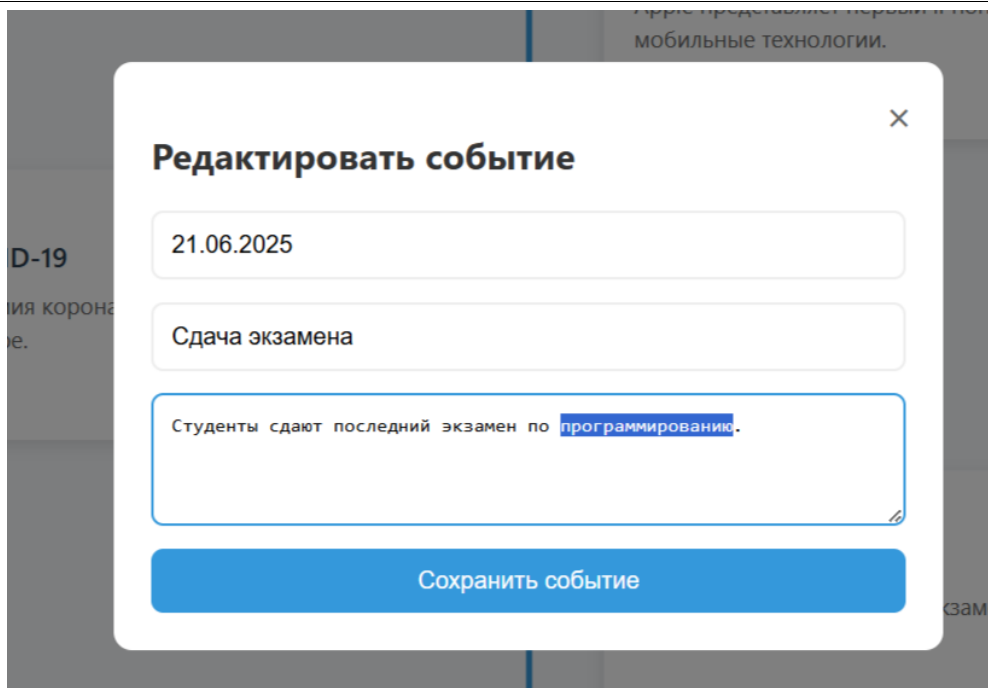


Рисунок 11. Редактирование события

Чтобы удалить событие, нажмите кнопку «» — система выведет диалоговое окно с подтверждением «Удалить это событие?», где необходимо выбрать «Да» (для удаления) или «Нет» (для отмены действия).

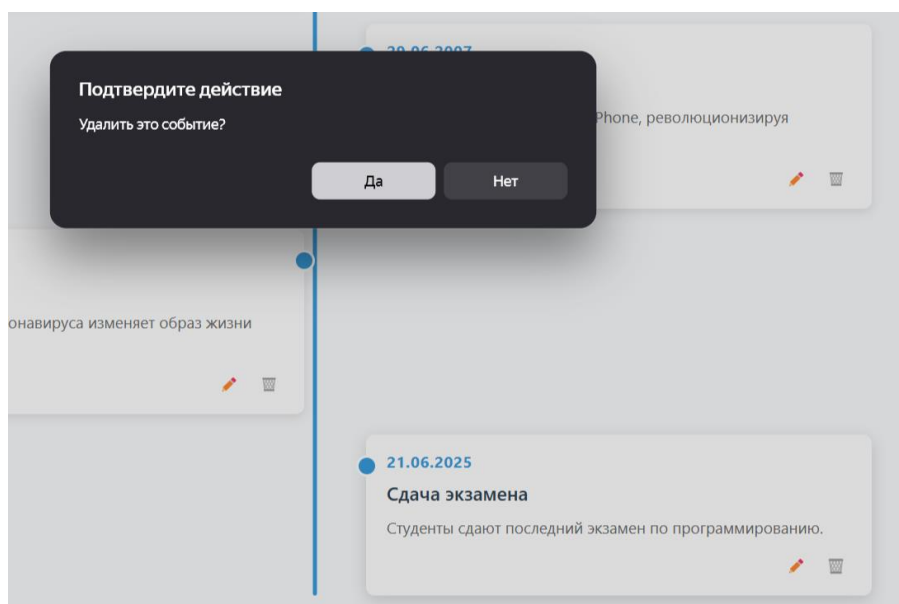


Рисунок 12. Удаление события

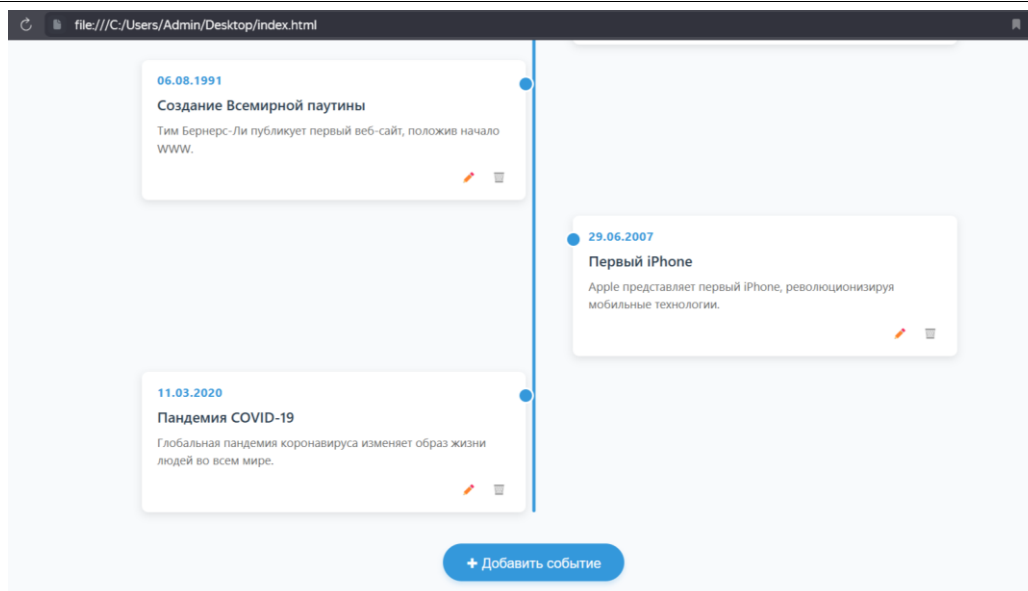


Рисунок 13. Готовая временная шкала

4 Выводы

В данной работе подробно рассмотрено создание интерактивной временной шкалы с использованием языка HTML5, которая поддерживает динамическое добавление событий, валидацию ввода и адаптивное отображение контента. Предложенное решение ориентировано на преподавателей и студентов, заинтересованных в применении современных веб-технологий для повышения эффективности образовательного процесса и превращения обучения в более увлекательный и интерактивный опыт.

Библиографический список

1. Бакулевская С. С. Подготовка педагогических работников к использованию технологий html5 для разработки электронных образовательных ресурсов //Информатика и образование. 2019. №. 5. С. 32-40.
2. Цилинская К.А. Создание и использование мультимедийных интерактивных плакатов по информатике //Будущее науки-2019. 2019. С. 340-344.
3. Королюк Я. А., Куликова Н. Ю. Опыт разработки и использования интерактивных образовательных ресурсов на основе html5 для онлайн-обучения школьников информатике //XXV Всероссийская студенческая научно-практическая конференция Нижневартковского государственного университета. 2023. С. 48-55.
4. Размологов Д. М., Панина А. С., Захаров П. В. Использование html5 для разработки электронного учебника //Современные тенденции развития науки и технологий. 2016. №. 4-4. С. 92-94.
5. Шегаев И. С., Полозов А. Г. Новейший политологический глоссарий: основные категории, понятия, термины. Общество с ограниченной ответственностью Издательство Перо, 2015. С. 242-242.

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Интерактивная временная шкала</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      line-height: 1.4;
      color: #333;
      max-width: 1200px;
      margin: 0 auto;
      padding: 20px;
      background-color: #f8fafc;
    }

    h1 {
      text-align: center;
      color: #2c3e50;
      margin-bottom: 30px;
      font-size: 2.2em;
      letter-spacing: -0.5px;
    }

    .timeline-container {
      position: relative;
      max-width: 1000px;
      margin: 40px auto;
    }

    .timeline-line {
      position: absolute;
      width: 4px;
      background-color: #3498db;
      top: 0;
      bottom: 0;
      left: 50%;
      transform: translateX(-50%);
      z-index: 0;
      border-radius: 2px;
    }

    .timeline-event {
      padding: 15px 20px;
      position: relative;
      width: 45%;
      background-color: white;
      border-radius: 8px;
```

```
    box-shadow: 0 3px 10px rgba(0, 0, 0, 0.1);
    margin-bottom: 20px;
    transition: all 0.3s ease;
    opacity: 0;
    transform: translateY(10px);
}
```

```
.timeline-event:nth-child(odd) {
    left: 0;
}
```

```
.timeline-event:nth-child(even) {
    left: 55%;
}
```

```
.timeline-event::after {
    content: "";
    position: absolute;
    width: 16px;
    height: 16px;
    background-color: #3498db;
    border-radius: 50%;
    top: 20px;
    z-index: 1;
    border: 3px solid white;
}
```

```
.timeline-event:nth-child(odd)::after {
    right: -10px;
}
```

```
.timeline-event:nth-child(even)::after {
    left: -10px;
}
```

```
.event-date {
    font-weight: 700;
    color: #3498db;
    margin-bottom: 8px;
    font-size: 0.95em;
    letter-spacing: 0.5px;
}
```

```
.event-title {
    font-size: 1.15em;
    margin-bottom: 8px;
    color: #2c3e50;
    font-weight: 600;
}
```

```
.event-description {
    color: #666;
}
```

```
margin-bottom: 8px;
font-size: 0.95em;
line-height: 1.5;
}

.event-actions {
display: flex;
justify-content: flex-end;
gap: 10px;
margin-top: 12px;
}

.edit-btn, .delete-btn {
background: none;
border: none;
cursor: pointer;
padding: 5px;
transition: transform 0.2s;
}

.edit-btn:hover, .delete-btn:hover {
transform: scale(1.1);
}

.timeline-event:hover {
transform: translateY(-3px);
box-shadow: 0 5px 15px rgba(0, 0, 0, 0.15);
}

.add-event-btn {
display: block;
margin: 30px auto;
padding: 12px 30px;
font-size: 1.1em;
background-color: #3498db;
color: white;
border: none;
border-radius: 25px;
cursor: pointer;
transition: all 0.3s ease;
box-shadow: 0 4px 6px rgba(52, 152, 219, 0.2);
}

.add-event-btn:hover {
background-color: #2980b9;
transform: translateY(-2px);
box-shadow: 0 6px 12px rgba(52, 152, 219, 0.3);
}

.modal {
display: none;
position: fixed;
```

```
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0,0,0,0.5);
    justify-content: center;
    align-items: center;
    z-index: 1000;
  }

.modal-content {
  background-color: #fff;
  padding: 25px;
  border-radius: 12px;
  position: relative;
  width: 90%;
  max-width: 500px;
  animation: modalOpen 0.3s ease;
}

@keyframes modalOpen {
  from { transform: scale(0.95); opacity: 0; }
  to { transform: scale(1); opacity: 1; }
}

.close-btn {
  position: absolute;
  top: 15px;
  right: 20px;
  font-size: 28px;
  cursor: pointer;
  color: #666;
  transition: color 0.2s;
}

.close-btn:hover {
  color: #333;
}

.event-form {
  display: flex;
  flex-direction: column;
  gap: 15px;
}

.event-form input,
.event-form textarea {
  padding: 12px;
  border: 2px solid #eee;
  border-radius: 8px;
  font-size: 1em;
  transition: border-color 0.3s;
```

```
}

.event-form input:focus,
.event-form textarea:focus {
  outline: none;
  border-color: #3498db;
}

.submit-btn {
  padding: 12px;
  background-color: #3498db;
  color: white;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-size: 1em;
  transition: background-color 0.3s;
}

.submit-btn:hover {
  background-color: #2980b9;
}

.error-message {
  color: #e74c3c;
  font-size: 0.85em;
  display: none;
  margin: -10px 0 5px;
}

.invalid {
  border-color: #e74c3c !important;
}

@media screen and (max-width: 768px) {
  .timeline-event {
    width: 85%;
    margin-bottom: 25px;
  }

  .timeline-event:nth-child(even) {
    left: 15%;
  }

  .timeline-line {
    left: 30px;
  }

  .timeline-event::after {
    width: 14px;
    height: 14px;
    top: 18px;
  }
}
```

```

    }

    .add-event-btn {
        width: 100%;
        max-width: 300px;
        padding: 15px;
    }
}
</style>
</head>
<body>
    <h1>Интерактивная временная шкала</h1>

    <div class="timeline-container">
        <div class="timeline-line"></div>
        <div id="timeline-events"></div>
    </div>

    <!-- Кнопка добавления -->
    <button class="add-event-btn" id="addEventBtn">+ Добавить событие</button>

    <!-- Модальное окно -->
    <div id="eventModal" class="modal">
        <div class="modal-content">
            <span class="close-btn">&times;</span>
            <h2>Добавить новое событие</h2>
            <form class="event-form" id="eventForm">
                <input type="text" id="eventDate" placeholder="Дата (ДД.ММ.ГГГГ)"
required
                pattern="^(0[1-9]|[12][0-9]|3[01])\.(0[1-9]|1[0-2])\.\d{4}$">
                <div id="dateError" class="error-message"></div>
                <input type="text" id="eventTitle" placeholder="Название события"
required>
                <textarea id="eventDescription" placeholder="Описание события" rows="4"
required></textarea>
                <button type="submit" class="submit-btn">Сохранить событие</button>
            </form>
        </div>
    </div>

    <!-- JavaScript -->
    <script>
        let events = JSON.parse(localStorage.getItem('timelineEvents')) || [
            {
                date: "20.07.1969",
                title: "Первый человек на Луне",
                description: "Нил Армстронг становится первым человеком, ступившим на
поверхность Луны."
            },
            {
                date: "09.11.1989",
                title: "Падение Берлинской стены",

```

```

        description: "Символическое окончание Холодной войны и начало
объединения Германии."
    },
    {
        date: "06.08.1991",
        title: "Создание Всемирной паутины",
        description: "Тим Бернерс-Ли публикует первый веб-сайт, положив начало
WWW."
    }
];

const timelineEvents = document.getElementById('timeline-events');
const addEventBtn = document.getElementById('addEventBtn');
const eventModal = document.getElementById('eventModal');
const closeBtn = document.querySelector('.close-btn');
const eventForm = document.getElementById('eventForm');
let editingIndex = null;

function displayEvents() {
    timelineEvents.innerHTML = "";

    events.forEach((event, index) => {
        const eventElement = document.createElement('div');
        eventElement.className = 'timeline-event';
        eventElement.innerHTML = `
            <div class="event-date">${event.date}</div>
            <div class="event-title">${event.title}</div>
            <div class="event-description">${event.description}</div>
            <div class="event-actions">
                <button class="edit-btn">✎</button>
                <button class="delete-btn">🗑️</button>
            </div>
        `;

        eventElement.querySelector('.delete-btn').addEventListener('click', () => {
            if (confirm("Удалить это событие?")) {
                events.splice(index, 1);
                saveAndUpdate();
            }
        });

        eventElement.querySelector('.edit-btn').addEventListener('click', () => {
            editingIndex = index;
            document.getElementById('eventDate').value = event.date;
            document.getElementById('eventTitle').value = event.title;
            document.getElementById('eventDescription').value = event.description;
            eventModal.querySelector('h2').textContent = 'Редактировать событие';
            eventModal.style.display = 'flex';
        });

        setTimeout(() => {
            eventElement.style.opacity = '1';
            eventElement.style.transform = 'translateY(0)';
        }, 100);
    });
}

```

```
    }, index * 100);

    timelineEvents.appendChild(eventElement);
  });
}

function validateDate(dateString) {
  const pattern = /^(0[1-9]|[12][0-9]|3[01])\.(0[1-9]|1[0-2])\.\d{4}$/;
  if (!pattern.test(dateString)) return false;

  const [day, month, year] = dateString.split('.').map(Number);
  const date = new Date(year, month - 1, day);
  return date.getDate() === day &&
    date.getMonth() === month - 1 &&
    date.getFullYear() === year;
}

function saveAndUpdate() {
  localStorage.setItem('timelineEvents', JSON.stringify(events));
  displayEvents();
}

function closeModal() {
  eventModal.style.display = 'none';
  editingIndex = null;
  eventForm.reset();
  document.getElementById('dateError').style.display = 'none';
  document.getElementById('eventDate').classList.remove('invalid');
}

addEventBtn.addEventListener('click', () => {
  eventModal.querySelector('h2').textContent = 'Добавить новое событие';
  eventModal.style.display = 'flex';
});

closeBtn.addEventListener('click', closeModal);
window.addEventListener('click', (e) => e.target === eventModal &&
closeModal());

eventForm.addEventListener('submit', (e) => {
  e.preventDefault();
  const dateInput = document.getElementById('eventDate');
  const dateError = document.getElementById('dateError');

  if (!validateDate(dateInput.value)) {
    dateInput.classList.add('invalid');
    dateError.textContent = dateInput.value ?
      'Некорректная дата (например, 30.02 не существует)' :
      'Пожалуйста, введите дату в формате ДД.ММ.ГГГГ';
    dateError.style.display = 'block';
    return;
  }
}
```

```
const newEvent = {
  date: dateInput.value,
  title: document.getElementById('eventTitle').value.trim(),
  description: document.getElementById('eventDescription').value.trim()
};

if (editingIndex !== null) {
  events[editingIndex] = newEvent;
} else {
  events.push(newEvent);
}

closeModal();
saveAndUpdate();
});

document.addEventListener('DOMContentLoaded', displayEvents);
window.addEventListener('scroll', () => {
  document.querySelectorAll('.timeline-event').forEach(event => {
    const rect = event.getBoundingClientRect();
    event.style.opacity = rect.top < window.innerHeight * 0.8 ? '1' : '0';
  });
});
</script>
</body>
</html>
```