ДИСЦИПЛИНА	Фронтенд и бэкенд разработка
ИНСТИТУТ	ИПТИП
КАФЕДРА	Индустриального программирования
ВИД УЧЕБНОГО МАТЕРИАЛА	Методические указания к практическим занятиям
ПРЕПОДАВАТЕЛЬ	Астафьев Рустам Уралович
СЕМЕСТР	1 семестр, 2025/2026 уч. год

Ссылка на материал:
https://github.com/astafiev-rustam/frontend-and-backend-development/tree/practice-1-25

Практическое занятие 25: Формы React: валидация, сообщения об ошибках и элементы доступности

В рамках данного занятия будут рассмотрены возможности работы с формами в React и обеспечением доступности. Материалы занятия соответствуют представлениям о формировании ввода и доступности страниц из предыдущих практик и лекций.

Теоретическая часть

Пример 1. Форма с валидацией в реальном времени

Проблема: Нужно создать форму добавления технологии с валидацией полей в реальном времени и понятными сообщениями об ошибках.

Подход к решению: Используем состояние для хранения ошибок, проверяем валидность при каждом изменении и блокируем отправку при ошибках.

Исходный код в файле `TechnologyForm.jsx`:

```
import { useState, useEffect } from 'react';

function TechnologyForm({ onSave, onCancel, initialData = {} }) {
  // Состояние формы
  const [formData, setFormData] = useState({
    title: initialData.title || '',
    description: initialData.description || '',
    category: initialData.category || 'frontend',
    difficulty: initialData.difficulty || 'beginner',
    deadline: initialData.deadline || '',
    resources: initialData.resources || ['']
  });

  // Состояние ошибок
  const [errors, setErrors] = useState({});
```

```
  const [isFormValid, setIsFormValid] = useState(false);

  // Валидация формы
  const validateForm = () => {
    const newErrors = {};

    // Валидация названия
    if (!formData.title.trim()) {
      newErrors.title = 'Название технологии обязательно';
    } else if (formData.title.trim().length < 2) {
      newErrors.title = 'Название должно содержать минимум 2 символа';
    } else if (formData.title.trim().length > 50) {
      newErrors.title = 'Название не должно превышать 50 символов';
    }

    // Валидация описания
    if (!formData.description.trim()) {
      newErrors.description = 'Описание технологии обязательно';
    } else if (formData.description.trim().length < 10) {
      newErrors.description = 'Описание должно содержать минимум 10 символов';
    }

    // Валидация дедлайна
    if (formData.deadline) {
      const deadlineDate = new Date(formData.deadline);
      const today = new Date();
      today.setHours(0, 0, 0, 0);

      if (deadlineDate < today) {
        newErrors.deadline = 'Дедлайн не может быть в прошлом';
      }
    }

    // Валидация ресурсов
    formData.resources.forEach((resource, index) => {
      if (resource && !isValidUrl(resource)) {
        newErrors[`resource_${index}`] = 'Введите корректный URL';
      }
    });

    setErrors(newErrors);
    setIsFormValid(Object.keys(newErrors).length === 0);
  };

  // Проверка URL
  const isValidUrl = (string) => {
    try {
      new URL(string);
      return true;
    } catch (_) {
      return false;
    }
  };
```

```
  // Валидация при каждом изменении формы
  useEffect(() => {
    validateForm();
  }, [formData]);

  // Обработчик изменения полей
  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({
      ...prev,
      [name]: value
    }));
  };

  // Обработчик изменения ресурсов
  const handleResourceChange = (index, value) => {
    const newResources = [...formData.resources];
    newResources[index] = value;
    setFormData(prev => ({
      ...prev,
      resources: newResources
    }));
  };

  // Добавление нового поля ресурса
  const addResourceField = () => {
    setFormData(prev => ({
      ...prev,
      resources: [...prev.resources, '']
    }));
  };

  // Удаление поля ресурса
  const removeResourceField = (index) => {
    if (formData.resources.length > 1) {
      const newResources = formData.resources.filter((_, i) => i !== index);
      setFormData(prev => ({
        ...prev,
        resources: newResources
      }));
    }
  };

  // Обработчик отправки формы
  const handleSubmit = (e) => {
    e.preventDefault();

    if (isFormValid) {
      // Очищаем пустые ресурсы перед сохранением
      const cleanedData = {
        ...formData,
        resources: formData.resources.filter(resource => resource.trim() !== '')
      };
```

```
      onSave(cleanedData);
    }
  };

  return (
<form onSubmit={handleSubmit} className="technology-form" noValidate>
      <h2>{initialData.title ? 'Редактирование технологии' : 'Добавление новой
технологии'}</h2>

      {/* Поле названия */}
      <div className="form-group">
        <label htmlFor="title" className="required">
          Название технологии
        </label>
        <input
          id="title"
          name="title"
          type="text"
          value={formData.title}
          onChange={handleChange}
          className={errors.title ? 'error' : ''}
          placeholder="Например: React, Node.js, TypeScript"
          aria-describedby={errors.title ? 'title-error' : undefined}
          required
        />
        {errors.title && (
          <span id="title-error" className="error-message" role="alert">
            {errors.title}
          </span>
        )}
      </div>

      {/* Поле описания */}
      <div className="form-group">
        <label htmlFor="description" className="required">
          Описание
        </label>
        <textarea
          id="description"
          name="description"
          value={formData.description}
          onChange={handleChange}
          rows="4"
          className={errors.description ? 'error' : ''}
          placeholder="Опишите, что это за технология и зачем её изучать..."
          aria-describedby={errors.description ? 'description-error' : undefined}
          required
        />
        {errors.description && (
          <span id="description-error" className="error-message" role="alert">
            {errors.description}
          </span>
        )}
      </div>
```

```
        {/* Выбор категории */}
        <div className="form-group">
          <label htmlFor="category">Категория</label>
          <select
            id="category"
            name="category"
            value={formData.category}
            onChange={handleChange}
          >
            <option value="frontend">Frontend</option>
            <option value="backend">Backend</option>
            <option value="mobile">Mobile</option>
            <option value="devops">DevOps</option>
            <option value="database">Базы данных</option>
            <option value="tools">Инструменты</option>
          </select>
        </div>

        {/* Выбор сложности */}
        <div className="form-group">
          <label htmlFor="difficulty">Уровень сложности</label>
          <select
            id="difficulty"
            name="difficulty"
            value={formData.difficulty}
            onChange={handleChange}
          >
            <option value="beginner">Начинающий</option>
            <option value="intermediate">Средний</option>
            <option value="advanced">Продвинутый</option>
          </select>
        </div>

        {/* Поле дедлайна */}
        <div className="form-group">
          <label htmlFor="deadline">
            Планируемая дата освоения
          </label>
          <input
            id="deadline"
            name="deadline"
            type="date"
            value={formData.deadline}
            onChange={handleChange}
            className={errors.deadline ? 'error' : ''}
            aria-describedby={errors.deadline ? 'deadline-error' : undefined}
          />
          {errors.deadline && (
            <span id="deadline-error" className="error-message" role="alert">
              {errors.deadline}
            </span>
          )}
        </div>
```

```
      {/* Поля ресурсов */}
      <div className="form-group">
        <label>Ресурсы для изучения</label>
        {formData.resources.map((resource, index) => (
          <div key={index} className="resource-field">
            <input
              type="url"
              value={resource}
              onChange={(e) => handleResourceChange(index, e.target.value)}
              placeholder="https://example.com"
              className={errors[`resource_${index}`] ? 'error' : ''}
              aria-describedby={errors[`resource_${index}`] ? `resource-${index}-
error` : undefined}
            />
            {formData.resources.length > 1 && (
              <button
                type="button"
                onClick={() => removeResourceField(index)}
                className="remove-resource"
                aria-label="Удалить ресурс"
              >
                ×
              </button>
            )}
            {errors[`resource_${index}`] && (
              <span
                id={`resource-${index}-error`}
                className="error-message"
                role="alert"
              >
                {errors[`resource_${index}`]}
              </span>
            )}
          </div>
        ))}
        <button
          type="button"
          onClick={addResourceField}
          className="add-resource"
        >
          + Добавить ещё ресурс
        </button>
      </div>

      {/* Кнопки формы */}
      <div className="form-actions">
        <button
          type="submit"
          disabled={!isFormValid}
          className="btn-primary"
        >
          {initialData.title ? 'Обновить технологию' : 'Добавить технологию'}
        </button>
```

```
        <button
          type="button"
          onClick={onCancel}
          className="btn-secondary"
        >
          Отмена
        </button>
      </div>

      {/* Информация о валидности формы */}
      {!isFormValid && (
        <div className="form-validation-info" role="status">
          ⚠ Заполните все обязательные поля корректно
        </div>
      )}
    </form>
  );
}

export default TechnologyForm;
```

Также организуем компонент `TechnologyManager.jsx` для внешнего контроля элемента формы:

```
// В компоненте, где используется TechnologyForm (например, App.js или
TechnologyManager.js)
import { useState } from 'react';
import TechnologyForm from './TechnologyForm';

function TechnologyManager() {
  const [technologies, setTechnologies] = useState([]);
  const [showForm, setShowForm] = useState(false);
  const [editingTech, setEditingTech] = useState(null);

  // Обработчик сохранения технологии
  const handleSaveTechnology = (techData) => {
    if (editingTech) {
      // Редактирование существующей технологии
      setTechnologies(prev =>
        prev.map(tech =>
          tech.id === editingTech.id
            ? { ...tech, ...techData, updatedAt: new Date().toISOString() }
            : tech
        )
      );
    } else {
      // Добавление новой технологии
      const newTechnology = {
        id: Date.now(), // В реальном приложении ID генерируется на сервере
        ...techData,
        status: 'not-started',
        createdAt: new Date().toISOString(),
```

```
        notes: '',
        progress: 0
      };
      setTechnologies(prev => [...prev, newTechnology]);
    }

    // Закрываем форму после сохранения
    setShowForm(false);
    setEditingTech(null);
  };

  // Обработчик редактирования
  const handleEdit = (technology) => {
    setEditingTech(technology);
    setShowForm(true);
  };

  // Обработчик отмены
  const handleCancel = () => {
    setShowForm(false);
    setEditingTech(null);
  };

  return (
    <div className="technology-manager">
      <div className="manager-header">
        <h2>Управление технологиями</h2>
        <button
          onClick={() => setShowForm(true)}
          className="btn-primary"
        >
          + Добавить технологию
        </button>
      </div>

      {/* Список технологий */}
      <div className="technologies-list">
        {technologies.map(tech => (
          <div key={tech.id} className="technology-item">
            <h3>{tech.title}</h3>
            <p>{tech.description}</p>
            <div className="tech-actions">
              <button onClick={() => handleEdit(tech)}>
                Редактировать
              </button>
            </div>
          </div>
        ))}
      </div>

      {/* Форма добавления/редактирования */}
      {showForm && (
        <div className="form-modal">
          <div className="modal-content">
```

```
            <TechnologyForm
              onSave={handleSaveTechnology}
              onCancel={handleCancel}
              initialData={editingTech || {}}
            />
          </div>
        </div>
      )}
    </div>
  );
}

export default TechnologyManager;
```

Добавим компонент менеджера в App.js и запустим приложение.

Пример 2. Форма с доступностью (Accessibility)

Проблема: Нужно сделать форму доступной для пользователей с ограниченными возможностями.

Подход к решению: Добавляем ARIA-атрибуты, правильную семантику и управление фокусом.

Исходный код в файле `WorkingAccessibleForm.jsx`:

```
import { useState, useRef, useEffect } from 'react';

function WorkingAccessibleForm() {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [message, setMessage] = useState('');
  const [errors, setErrors] = useState({});

  const statusRef = useRef(null);
  const nameRef = useRef(null);

  // Валидация при изменении полей
  useEffect(() => {
    const newErrors = {};

    if (name && name.length < 2) {
      newErrors.name = 'Имя должно быть не короче 2 символов';
    }

    if (email && !email.includes('@')) {
      newErrors.email = 'Email должен содержать @';
    }

    if (message && message.length < 5) {
      newErrors.message = 'Сообщение должно быть не короче 5 символов';
    }

    setErrors(newErrors);
```

```
  }, [name, email, message]);

  const handleSubmit = (e) => {
    e.preventDefault();

    const newErrors = {};

    if (!name) newErrors.name = 'Введите имя';
    if (!email) newErrors.email = 'Введите email';
    if (!message) newErrors.message = 'Введите сообщение';

    if (Object.keys(newErrors).length > 0) {
      setErrors(newErrors);
      if (statusRef.current) {
        statusRef.current.textContent = 'Заполните все обязательные поля';
      }
      // Фокусируемся на первом поле с ошибкой
      if (nameRef.current) {
        nameRef.current.focus();
      }
      return;
    }

    // Если есть другие ошибки валидации
    if (Object.keys(errors).length > 0) {
      if (statusRef.current) {
        statusRef.current.textContent = 'Исправьте ошибки в форме';
      }
      return;
    }

    // Успешная отправка
    if (statusRef.current) {
      statusRef.current.textContent = 'Форма успешно отправлена!';
    }
    console.log('Отправлены данные:', { name, email, message });
  };

  return (
    <div style={{ maxWidth: '500px', margin: '20px', padding: '20px', border: '1px
solid #ccc' }}>
      <h1>Контактная форма</h1>

      {/* Область для скринридера */}
      <div
        ref={statusRef}
        aria-live="assertive"
        style={{
          position: 'absolute',
          left: '-10000px',
          width: '1px',
          height: '1px',
          overflow: 'hidden'
        }}
```

```
      />

      <form onSubmit={handleSubmit} noValidate>
        {/* Поле имени */}
        <div style={{ marginBottom: '20px' }}>
          <label htmlFor="name" style={{ display: 'block', marginBottom: '5px' }}>
            Ваше имя *
          </label>
          <input
            ref={nameRef}
            id="name"
            type="text"
            value={name}
            onChange={(e) => setName(e.target.value)}
            aria-required="true"
            aria-invalid={!!errors.name}
            aria-describedby={errors.name ? "name-error" : undefined}
            style={{
              width: '100%',
              padding: '8px',
              border: errors.name ? '2px solid red' : '1px solid #ccc',
              fontSize: '16px'
            }}
          />
          {errors.name && (
            <div id="name-error" style={{ color: 'red', fontSize: '14px',
marginTop: '5px' }}>
              {errors.name}
            </div>
          )}
        </div>

        {/* Поле email */}
        <div style={{ marginBottom: '20px' }}>
          <label htmlFor="email" style={{ display: 'block', marginBottom: '5px'
}}>
            Email адрес *
          </label>
          <input
            id="email"
            type="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            aria-required="true"
            aria-invalid={!!errors.email}
            aria-describedby={errors.email ? "email-error" : undefined}
            style={{
              width: '100%',
              padding: '8px',
              border: errors.email ? '2px solid red' : '1px solid #ccc',
              fontSize: '16px'
            }}
          />
          {errors.email && (
```

```
            <div id="email-error" style={{ color: 'red', fontSize: '14px',
marginTop: '5px' }}>
              {errors.email}
            </div>
          )}
        </div>

        {/* Поле сообщения */}
        <div style={{ marginBottom: '20px' }}>
          <label htmlFor="message" style={{ display: 'block', marginBottom: '5px'
}}>
            Ваше сообщение *
          </label>
          <textarea
            id="message"
            value={message}
            onChange={(e) => setMessage(e.target.value)}
            rows="4"
            aria-required="true"
            aria-invalid={!!errors.message}
            aria-describedby={errors.message ? "message-error" : undefined}
            style={{
              width: '100%',
              padding: '8px',
              border: errors.message ? '2px solid red' : '1px solid #ccc',
              fontSize: '16px',
              fontFamily: 'inherit'
            }}
          />
          {errors.message && (
            <div id="message-error" style={{ color: 'red', fontSize: '14px',
marginTop: '5px' }}>
              {errors.message}
            </div>
          )}
        </div>

        <button
          type="submit"
          style={{
            padding: '12px 24px',
            backgroundColor: '#007bff',
            color: 'white',
            border: 'none',
            borderRadius: '4px',
            fontSize: '16px',
            cursor: 'pointer'
          }}
        >
          Отправить сообщение
        </button>
      </form>

      {/* Визуальный статус */}
```

```
        <div
          aria-live="polite"
          style={{
            marginTop: '15px',
            padding: '10px',
            border: '1px solid #ddd',
            backgroundColor: '#f9f9f9',
            display: 'none' // Скрыт визуально, но доступен для скринридеров
          }}
        >
          Статус формы будет отображаться здесь для скринридеров
        </div>
      </div>
    );
}

export default WorkingAccessibleForm;
```

Пример 3. Экспорт и импорт данных в приложениях

Рассмотрим пример по загрузке/выгрузке данных приложения в компоненте `DataImportExport.jsx`:

```
import { useState, useEffect } from 'react';

function DataImportExport() {
  const [technologies, setTechnologies] = useState([]);
  const [status, setStatus] = useState('');

  // Загрузка данных из localStorage при старте
  useEffect(() => {
    const savedData = localStorage.getItem('techTrackerData');
    if (savedData) {
      try {
        const parsedData = JSON.parse(savedData);
        setTechnologies(parsedData);
        setStatus(`Загружено ${parsedData.length} технологий из памяти`);
      } catch (error) {
        setStatus('Ошибка загрузки данных из памяти');
      }
    }
  }, []);

  // Автосохранение при изменении technologies
  useEffect(() => {
    if (technologies.length > 0) {
      localStorage.setItem('techTrackerData', JSON.stringify(technologies));
    }
  }, [technologies]);

  // Экспорт данных в JSON файл
  const handleExport = () => {
```

```
      const exportData = {
        version: '1.0',
        exportedAt: new Date().toISOString(),
        technologies: technologies,
        stats: {
          total: technologies.length,
          completed: technologies.filter(t => t.status === 'completed').length,
          inProgress: technologies.filter(t => t.status === 'in-progress').length
        }
      };

      const dataStr = JSON.stringify(exportData, null, 2);
      const blob = new Blob([dataStr], { type: 'application/json' });
      const url = URL.createObjectURL(blob);

      const link = document.createElement('a');
      link.href = url;
      link.download = `tech-tracker-${new Date().toISOString().split('T')[0]}.json`;
      document.body.appendChild(link);
      link.click();
      document.body.removeChild(link);
      URL.revokeObjectURL(url);

      setStatus(`Экспортировано ${technologies.length} технологий`);
    };

    // Импорт данных из JSON файла
    const handleImport = (event) => {
      const file = event.target.files[0];
      if (!file) return;

      const reader = new FileReader();

      reader.onload = (e) => {
        try {
          const fileContent = e.target.result;
          const importedData = JSON.parse(fileContent);

          // Проверяем структуру файла
          if (!importedData.technologies ||
!Array.isArray(importedData.technologies)) {
            throw new Error('Неверный формат файла');
          }

          // Валидация каждой технологии
          const validTechnologies = importedData.technologies.filter(tech =>
            tech && tech.id && tech.title && tech.description
          );

          if (validTechnologies.length === 0) {
            throw new Error('В файле нет валидных технологий');
          }

          // Добавляем импортированные технологии
```

```
        setTechnologies(prev => {
          const newTech = validTechnologies.filter(newTech =>
            !prev.some(existingTech => existingTech.id === newTech.id)
          );
          return [...prev, ...newTech];
        });

        setStatus(`Импортировано ${validTechnologies.length} технологий`);

      } catch (error) {
        setStatus(`Ошибка импорта: ${error.message}`);
      }
    };

    reader.onerror = () => {
      setStatus('Ошибка чтения файла');
    };

    reader.readAsText(file);

    // Сбрасываем input чтобы можно было выбрать тот же файл снова
    event.target.value = '';
  };

  // Добавление тестовой технологии
  const addSampleTechnology = () => {
    const newTech = {
      id: Date.now(),
      title: `Технология ${technologies.length + 1}`,
      description: 'Описание технологии для демонстрации',
      status: 'not-started',
      category: 'frontend',
      createdAt: new Date().toISOString()
    };

    setTechnologies(prev => [...prev, newTech]);
    setStatus('Добавлена тестовая технология');
  };

  // Очистка всех данных
  const clearAllData = () => {
    setTechnologies([]);
    localStorage.removeItem('techTrackerData');
    setStatus('Все данные очищены');
  };

  // Изменение статуса технологии
  const toggleStatus = (techId) => {
    setTechnologies(prev =>
      prev.map(tech => {
        if (tech.id === techId) {
          const statuses = ['not-started', 'in-progress', 'completed'];
          const currentIndex = statuses.indexOf(tech.status);
          const nextStatus = statuses[(currentIndex + 1) % statuses.length];
```

```
          return { ...tech, status: nextStatus };
        }
        return tech;
      })
    );
  };

  return (
    <div style={{
      maxWidth: '800px',
      margin: '20px auto',
      padding: '20px',
      fontFamily: 'Arial, sans-serif'
    }}>
      <h1>Импорт/Экспорт данных</h1>

      {/* Статус */}
      {status && (
        <div style={{
          padding: '10px',
          margin: '10px 0',
          backgroundColor: status.includes('Ошибка') ? '#ffebee' : '#e8f5e8',
          border: `1px solid ${status.includes('Ошибка') ? '#f44336' :
'#4caf50'}`,
          borderRadius: '4px'
        }}>
          {status}
        </div>
      )}

      {/* Управление данными */}
      <div style={{
        display: 'flex',
        gap: '10px',
        flexWrap: 'wrap',
        marginBottom: '20px'
      }}>
        <button
          onClick={addSampleTechnology}
          style={{
            padding: '10px 15px',
            backgroundColor: '#2196f3',
            color: 'white',
            border: 'none',
            borderRadius: '4px',
            cursor: 'pointer'
          }}
        >
          + Добавить тестовую технологию
        </button>

        <button
          onClick={handleExport}
```

```
            disabled={technologies.length === 0}
            style={{
              padding: '10px 15px',
              backgroundColor: technologies.length === 0 ? '#ccc' : '#4caf50',
              color: 'white',
              border: 'none',
              borderRadius: '4px',
              cursor: technologies.length === 0 ? 'not-allowed' : 'pointer'
            }}
          >
            📥 Экспорт в JSON ({technologies.length})
          </button>

          <label style={{
            padding: '10px 15px',
            backgroundColor: '#ff9800',
            color: 'white',
            border: 'none',
            borderRadius: '4px',
            cursor: 'pointer',
            display: 'inline-block'
          }}>
            📤 Импорт из JSON
            <input
              type="file"
              accept=".json"
              onChange={handleImport}
              style={{ display: 'none' }}
            />
          </label>

          <button
            onClick={clearAllData}
            disabled={technologies.length === 0}
            style={{
              padding: '10px 15px',
              backgroundColor: technologies.length === 0 ? '#ccc' : '#f44336',
              color: 'white',
              border: 'none',
              borderRadius: '4px',
              cursor: technologies.length === 0 ? 'not-allowed' : 'pointer'
            }}
          >
            🗑️ Очистить все
          </button>
        </div>

        {/* Список технологий */}
        <div>
          <h2>Технологии ({technologies.length})</h2>

          {technologies.length === 0 ? (
            <p style={{ color: '#666', fontStyle: 'italic' }}>
              Технологий пока нет. Добавьте первую или импортируйте данные.
```

```
          </p>
        ) : (
          <div style={{ display: 'flex', flexDirection: 'column', gap: '10px' }}>
            {technologies.map(tech => (
              <div
                key={tech.id}
                style={{
                  padding: '15px',
                  border: '1px solid #ddd',
                  borderRadius: '4px',
                  backgroundColor: '#f9f9f9'
                }}
              >
                <div style={{
                  display: 'flex',
                  justifyContent: 'space-between',
                  alignItems: 'flex-start'
                }}>
                  <div>
                    <h3 style={{ margin: '0 0 5px 0' }}>{tech.title}</h3>
                    <p style={{ margin: '0 0 10px 0', color: '#666' }}>
                      {tech.description}
                    </p>
                    <div style={{ fontSize: '14px', color: '#888' }}>
                      Категория: {tech.category} • ID: {tech.id}
                    </div>
                  </div>

                  <button
                    onClick={() => toggleStatus(tech.id)}
                    style={{
                      padding: '5px 10px',
                      border: 'none',
                      borderRadius: '4px',
                      cursor: 'pointer',
                      backgroundColor:
                        tech.status === 'completed' ? '#4caf50' :
                        tech.status === 'in-progress' ? '#ff9800' : '#f44336',
                      color: 'white',
                      fontSize: '12px'
                    }}
                  >
                    {tech.status === 'completed' ? '☑ Завершено' :
                     tech.status === 'in-progress' ? '🔄 В процессе' : '⏳ Не
начато'}
                  </button>
                </div>
              </div>
            ))}
          </div>
        )}
      </div>

      {/* Статистика */}
```

```
      {technologies.length > 0 && (
        <div style={{
          marginTop: '20px',
          padding: '15px',
          backgroundColor: '#e3f2fd',
          borderRadius: '4px'
        }}>
          <h3>Статистика:</h3>
          <div style={{ display: 'flex', gap: '20px', flexWrap: 'wrap' }}>
            <div>Всего: <strong>{technologies.length}</strong></div>
            <div>Завершено: <strong>{technologies.filter(t => t.status === 'completed').length}</strong></div>
            <div>В процессе: <strong>{technologies.filter(t => t.status === 'in-progress').length}</strong></div>
            <div>Не начато: <strong>{technologies.filter(t => t.status === 'not-started').length}</strong></div>
          </div>
        </div>
      )}
    </div>
  );
}

export default DataImportExport;
```

Протестируйте приложение. Пример для загрузки сразу трёх технологий и импорта:

```
{
  "version": "1.0",
  "exportedAt": "2025-11-23T15:33:40.532Z",
  "technologies": [
    {
      "id": 1763912015622,
      "title": "Технология 1",
      "description": "Описание технологии для демонстрации",
      "status": "not-started",
      "category": "frontend",
      "createdAt": "2025-11-23T15:33:35.622Z"
    },
    {
      "id": 1763912015623,
      "title": "Технология 2",
      "description": "Описание второй технологии",
      "status": "in-progress",
      "category": "backend",
      "createdAt": "2025-11-23T15:33:36.123Z"
    },
    {
      "id": 1763912015624,
      "title": "Технология 3",
      "description": "Описание третьей технологии",
      "status": "completed",
```

```
      "category": "fullstack",
      "createdAt": "2025-11-23T15:33:36.624Z"
    }
  ],
  "stats": {
    "total": 1,
    "completed": 0,
    "inProgress": 0
  }
}
```

Практическая часть

Реализация функциональности экспорта данных

Шаг 1: Создайте компонент для экспорта данных

```
// components/DataExporter.js
import { useState } from 'react';

function DataExporter({ technologies }) {
  const [exportFormat, setExportFormat] = useState('json');
  const [includeUserData, setIncludeUserData] = useState(true);

  // Функция для экспорта данных
  const exportData = () => {
    const exportData = {
      version: '1.0',
      exportedAt: new Date().toISOString(),
      technologies: includeUserData
        ? technologies.map(tech => ({
            ...tech,
            userNotes: tech.notes || '',
            userStatus: tech.status || 'not-started',
            userDeadline: tech.deadline || ''
          }))
        : technologies.map(({ notes, status, deadline, ...tech }) => tech) //
Исключаем пользовательские данные
    };

    let dataStr, fileType, fileName;

    if (exportFormat === 'json') {
      dataStr = JSON.stringify(exportData, null, 2);
      fileType = 'application/json';
      fileName = `technology-roadmap-${new Date().toISOString().split('T')
[0]}.json`;
    }

    // Создаем и скачиваем файл
    const blob = new Blob([dataStr], { type: fileType });
```

```
    const url = URL.createObjectURL(blob);
    const link = document.createElement('a');
    link.href = url;
    link.download = fileName;
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);
    URL.revokeObjectURL(url);
  };

  // Валидация перед экспортом
  const canExport = technologies.length > 0;

  return (
    <div className="data-exporter">
      <h3>Экспорт данных</h3>

      <div className="export-options">
        <div className="form-group">
          <label htmlFor="export-format">Формат экспорта</label>
          <select
            id="export-format"
            value={exportFormat}
            onChange={(e) => setExportFormat(e.target.value)}
          >
            <option value="json">JSON</option>
            <option value="csv" disabled>CSV (скоро)</option>
          </select>
        </div>

        <div className="form-group checkbox-group">
          <label>
            <input
              type="checkbox"
              checked={includeUserData}
              onChange={(e) => setIncludeUserData(e.target.checked)}
            />
            Включить мои заметки и прогресс
          </label>
          <span className="help-text">
            При включении будут экспортированы ваши личные заметки и статусы
изучения
          </span>
        </div>
      </div>

      {!canExport && (
        <div className="export-warning" role="alert">
          ⚠ Нет данных для экспорта. Добавьте технологии в трекер.
        </div>
      )}

      <button
        onClick={exportData}
```

```
          disabled={!canExport}
          className="btn-primary"
          aria-describedby={canExport ? 'export-help' : 'export-warning'}
        >
          📥 Экспортировать данные
        </button>

        <div id="export-help" className="help-text">
          Данные будут сохранены в выбранном формате на вашем устройстве
        </div>
      </div>
    );
}

export default DataExporter;
```

Шаг 2: Добавьте обработку ошибок импорта

```
// components/DataImporter.js
import { useState } from 'react';

function DataImporter({ onImport }) {
  const [importError, setImportError] = useState('');
  const [isDragging, setIsDragging] = useState(false);

  // Валидация импортируемых данных
  const validateImportData = (data) => {
    if (!data.technologies || !Array.isArray(data.technologies)) {
      throw new Error('Неверный формат файла: отсутствует массив technologies');
    }

    data.technologies.forEach((tech, index) => {
      if (!tech.title || !tech.description) {
        throw new Error(`Технология #${index + 1}: отсутствует название или
описание`);
      }

      if (tech.title.length > 50) {
        throw new Error(`Технология "${tech.title}": название слишком длинное`);
      }
    });

    return true;
  };

  // Обработка загруженного файла
  const handleFileUpload = (file) => {
    setImportError('');

    const reader = new FileReader();

    reader.onload = (e) => {
```

```
      try {
        const fileContent = e.target.result;
        const importedData = JSON.parse(fileContent);

        validateImportData(importedData);
        onImport(importedData.technologies);

      } catch (error) {
        setImportError(`Ошибка импорта: ${error.message}`);
      }
    };

    reader.onerror = () => {
      setImportError('Ошибка чтения файла');
    };

    reader.readAsText(file);
  };

  // Обработчик выбора файла
  const handleFileSelect = (e) => {
    const file = e.target.files[0];
    if (file) {
      if (file.type === 'application/json') {
        handleFileUpload(file);
      } else {
        setImportError('Поддерживаются только JSON файлы');
      }
    }
  };

  // Обработчики drag & drop
  const handleDragOver = (e) => {
    e.preventDefault();
    setIsDragging(true);
  };

  const handleDragLeave = (e) => {
    e.preventDefault();
    setIsDragging(false);
  };

  const handleDrop = (e) => {
    e.preventDefault();
    setIsDragging(false);

    const file = e.dataTransfer.files[0];
    if (file) {
      handleFileUpload(file);
    }
  };

  return (
    <div className="data-importer">
```

```
        <h3>Импорт дорожной карты</h3>

        <div
          className={`drop-zone ${isDragging ? 'dragging' : ''} ${importError ?
'error' : ''}`}
          onDragOver={handleDragOver}
          onDragLeave={handleDragLeave}
          onDrop={handleDrop}
        >
          <div className="drop-zone-content">
            <p>📁 Перетащите JSON файл сюда или</p>
            <input
              type="file"
              accept=".json,application/json"
              onChange={handleFileSelect}
              id="file-input"
              className="file-input"
            />
            <label htmlFor="file-input" className="btn-secondary">
              Выберите файл
            </label>
          </div>
        </div>

        {importError && (
          <div className="import-error" role="alert">
            ❌ {importError}
          </div>
        )}

        <div className="import-help">
          <h4>Требования к файлу:</h4>
          <ul>
            <li>Формат: JSON</li>
            <li>Обязательные поля: title, description</li>
            <li>Максимальная длина названия: 50 символов</li>
          </ul>
        </div>
      </div>
    );
}

export default DataImporter;
```

Самостоятельная работа

Задание 1: Создайте форму для установки сроков изучения с валидацией

Задание 2: Добавьте компонент для массового редактирования статусов технологий

Что проверить перед завершением:

- Валидация форм работает в реальном времени

- Сообщения об ошибках понятны и доступны
- Экспорт данных создает корректный JSON файл
- Импорт данных обрабатывает ошибки формата
- Формы доступны для пользователей с ограниченными возможностями