

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЕТ
по лабораторной работе № 4
по дисциплине «Web программирование»
Тема: «Интернационализация Web-
приложений»

Студент гр. 3311 Баймухамедов Р. Р. _____

Преподаватель Калмыков М.А. _____

Санкт-Петербург

2025

Цель работы

Знакомство со способами отображения данных на различных языках при использовании файлов ресурсов.

Выполнение лабораторной работы

В данной лабораторной работе будем работать с файлами script.js и server.js. Немного затрагивая остальные файлы.

index.html

```
<!doctype html>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title id="app_title">Movies</title>

<div style="display:flex; gap:8px; align-items:center; margin:8px 0;">
  <span>Lang:</span>
  <button type="button" data-lang="en">EN</button>
  <button type="button" data-lang="ru">RU</button>
</div>

<a id="app_title" href="#">About cinema</a>

<form id="search-form">
  <input id="q" placeholder="" autocomplete="off" />
  <button id="search_btn">Search</button>
</form>

<p id="hint"></p>
<div id="results" style="margin-top:16px;"></div>

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

Добавили кнопки выбора языка

movies.json

```
[
  {
    "id": 1,
    "title": "Матрица",
    "year": 1999,
    "genres": ["Боевик", "Фантастика"],
    "director": "Лана Вачовски, Лилли Вачовски",
    "rating": 8.7
  },
  {
    "id": 2,
    "title": "Начало",
    "year": 2010,
```

```

    "genres": ["Боевик", "Фантастика", "Триллер"],
    "director": "Кристофер Нолан",
    "rating": 8.8
  },
  {
    "id": 3,
    "title": "Унесённые призраками",
    "year": 2001,
    "genres": ["Анимация", "Фэнтези", "Семейный"],
    "director": "Хаяо Миядзаки",
    "rating": 8.6
  },
  {
    "id": 4,
    "title": "Интерстеллар",
    "year": 2014,
    "genres": ["Приключения", "Драма", "Фантастика"],
    "director": "Кристофер Нолан",
    "rating": 8.6
  },
  {
    "id": 5,
    "title": "Паразиты",
    "year": 2019,
    "genres": ["Драма", "Триллер"],
    "director": "Пон Джун-хо",
    "rating": 8.5
  },
  {
    "id": 6,
    "title": "Крёстный отец",
    "year": 1972,
    "genres": ["Криминал", "Драма"],
    "director": "Фрэнсис Форд Коппола",
    "rating": 9.2
  }
]

```

Добавили json файл с данными фильмов на русском

Теперь перейдем к основным изменениям

script.js

```

// src/script.js
console.log("Movies UI: search-by-title + translate-by-ids", new
Date().toLocaleString());

function $(sel) { return document.querySelector(sel); }
function api(path) { return fetch(path).then(r => r.json()); }

// --- URL helpers ---
function getParam(name) {

```

```

    return new URLSearchParams(location.search).get(name) || "";
}
function setParams(obj, opts) {
    var replace = opts && opts.replace === true;
    var url = new URL(location.href);
    Object.keys(obj).forEach(function (k) {
        var v = obj[k];
        if (v === null || v === undefined || v === "") url.searchParams.delete(k);
        else url.searchParams.set(k, v);
    });
    if (replace) history.replaceState(null, "", url);
    else history.pushState(null, "", url);
}

// --- Lang + i18n ---
function detectLang() {
    return getParam("lang")
        || localStorage.getItem("lang")
        || (navigator.language || "en").slice(0, 2)
        || "en";
}
var lang = (["ru", "en"].indexOf(detectLang()) >= 0) ? detectLang() : "en";
var dict = null;
var lastIds = []; // запоминаем id текущей выдачи

async function loadStrings() {
    try {
        dict = await fetch(`/i18n/${lang}.json`).then(r => r.json());
    } catch {
        dict = (lang === "ru")
            ? { app_title: "Около Кино", search_placeholder: "Введите название фильма...",
search_button: "Искать", no_results: "Ничего не найдено" }
            : { app_title: "About movies", search_placeholder: "Enter movie title...",
search_button: "Search", no_results: "No results" };
    }

    document.title = dict.app_title;

    const map = [
        ["#app_title", "app_title"],
        ["#h1_title", "app_title"],
        ["#search_btn", "search_button"]
    ];
    for (const [sel, key] of map) {
        const el = document.querySelector(sel);
        if (el && dict[key] !== undefined) el.textContent = dict[key];
    }

    const q = document.querySelector("#q");
    if (q && dict.search_placeholder) q.placeholder = dict.search_placeholder;
}

// --- Render ---
function render(items) {

```

```

var box = $("#results");
if (!Array.isArray(items) || items.length === 0) {
    box.innerHTML = "<p>" + (dict ? (dict.no_results || "No results") : "No
results") + "</p>";
    lastIds = [];
    return;
}
var html = items.map(function(m){
    return (
        '<div style="border:1px solid #ccc; padding:12px; border-radius:10px;
margin:10px 0;">' +
        '<h3 style="margin:0 0 6px;">' + m.title + ' <small>(' + m.year +
')</small></h3>' +
        '<div><b>Genres:</b> ' + ((m.genres||[]).join(", ")) + '</div>' +
        '<div><b>Director:</b> ' + (m.director || "-") + '</div>' +
        '<div><b>Rating:</b> ' + (m.rating != null ? m.rating : "-") + '</div>' +
        '</div>'
    );
}).join("");
box.innerHTML = html;
lastIds = items.map(function(m){ return m.id; });
}

// --- выбрать title на новой локали для адресной строки ---
function chooseSearchTitle(items) {
    if (!Array.isArray(items) || items.length === 0) return "";
    return items[0].title || "";
}

// --- API runners ---
function runSearchByTitle(title) {
    if (!title) {
        document.querySelector("#results").innerHTML = "";
        lastIds = [];
        return Promise.resolve();
    }
    return api(`/api/movies?title=${encodeURIComponent(title)}&lang=${lang}`)
        .then(data => { render(data.items); });
}

function isFiniteNumber(n){ return typeof n === "number" && isFinite(n); }

function runFetchByIds(ids) {
    if (!ids || ids.length === 0) return Promise.resolve({ items: [] });
    var qs = "/api/movies?ids=" + ids.join(",") + "&lang=" + lang;
    return api(qs).then(function(data){
        render(data.items);
        return data; // чтобы взять локализованный title
    });
}

// --- Init ---
document.addEventListener("DOMContentLoaded", function() {
    if (getParam("lang") !== lang) setParams({ lang: lang }, { replace: true });

```

```

localStorage.setItem("lang", lang);

loadStrings().then(function(){
  var initialTitle = getParam("title");
  var idsParam = getParam("ids");
  var qEl = $("#q");
  if (qEl) qEl.value = initialTitle;

  if (idsParam) {
    var arr = idsParam.split(",").map(function(s){ return parseInt(s,10);
}).filter(isFiniteNumber);
    runFetchByIds(arr);
  } else if (initialTitle) {
    runSearchByTitle(initialTitle);
  } else {
    var hint = $("#hint");
    if (hint) hint.textContent = dict.prompt_type_title || "Type a movie title";
  }

  // Переключение языка (кнопки с data-lang="en"/"ru")
  document.querySelectorAll("[data-lang]").forEach(btn => {
    btn.addEventListener("click", async () => {
      const newLang = btn.getAttribute("data-lang");
      if (newLang === lang) return;
      lang = newLang;
      localStorage.setItem("lang", lang);

      // 1) Сразу обновляем UI-тексты
      await loadStrings();

      if (lastIds.length > 0) {
        // 2) Тихо получаем те же фильмы в новой локале
        const data = await runFetchByIds(lastIds);
        // 3) Выбираем title для адресной строки
        const chosenTitle = chooseSearchTitle(data.items);
        // 4) Обновляем URL на ?lang=...&title=..., ids чистим
        setParams({ lang, title: chosenTitle, ids: "" });
        // 5) Повторяем поиск по title (для консистентности URL и выдачи)
        await runSearchByTitle(chosenTitle);
        const q = document.querySelector("#q");
        if (q) q.value = chosenTitle;
      } else {
        // Нет результатов – просто сменили язык, UI уже обновлён
        setParams({ lang, ids: "" });
        const qv = (document.querySelector("#q")?.value || "").trim();
        if (qv) await runSearchByTitle(qv);
      }
    });
  });

  // Поиск по форме
  var form = $("#search-form");
  if (form) {

```

```

        form.addEventListener("submit", function(e){
            e.preventDefault();
            var qvEl = $("#q");
            var title = qvEl ? (qvEl.value || "").trim() : "";
            setParams({ title: title, lang: lang, ids: "" });
            runSearchByTitle(title);
        });
    }
    });
});

// Навигация назад/вперёд
window.addEventListener("popstate", function() {
    var t = getParam("title");
    var l = getParam("lang");
    var idsParam = getParam("ids");
    if (l && l !== lang) {
        lang = l;
        loadStrings().then(function(){
            var qEl = $("#q"); if (qEl) qEl.value = t;
            if (idsParam) {
                var arr = idsParam.split(",").map(function(s){ return parseInt(s,10);
            }).filter(isFiniteNumber);
                runFetchByIds(arr);
            } else {
                runSearchByTitle(t);
            }
        });
    } else {
        var qEl2 = $("#q"); if (qEl2) qEl2.value = t;
        if (idsParam) {
            var arr2 = idsParam.split(",").map(function(s){ return parseInt(s,10);
        }).filter(isFiniteNumber);
            runFetchByIds(arr2);
        } else {
            runSearchByTitle(t);
        }
    }
});
});

```

Добавим функции обработки языка и прогрузка корректной страницы

server.js

```

// app/server.js
const fs = require("fs");
const path = require("path");
const express = require("express");
const app = express();

const PORT = parseInt(process.env.PORT || "3000", 10);
const API_PREFIX = process.env.API_PREFIX || "/api";
const DEFAULT_PAGE_SIZE = parseInt(process.env.DEFAULT_PAGE_SIZE || "50", 10);

```

```

const DEFAULT_LANG = String(process.env.DEFAULT_LANG || "en").toLowerCase();

const SUPPORTED = new Set(["en", "ru"]);
const cache = new Map(); // lang -> movies[]

function normLang(x) {
  const s = String(x || "").toLowerCase().slice(0, 2);
  return SUPPORTED.has(s) ? s : DEFAULT_LANG;
}

function pickLang(req) {
  if (req.query.lang) return normLang(req.query.lang);
  const cookie = (req.headers.cookie || "").match(/(?:^|;|s*)lang=([a-z\-\+]+)/i)?.[1];
  if (cookie) return normLang(cookie);
  return DEFAULT_LANG;
}

function loadMovies(lang) {
  if (cache.has(lang)) return cache.get(lang);
  const file = path.join(__dirname, `movies.${lang}.json`);
  let data = [];
  try {
    data = JSON.parse(fs.readFileSync(file, "utf8"));
  } catch (e) {
    console.error(`[FATAL] Cannot read ${file}:`, e.message);
    cache.set(lang, null);
    return null;
  }
  cache.set(lang, data);
  return data;
}

function parseIdsParam(ids) {
  const seen = new Set();
  const out = [];
  String(ids).split(",").forEach(s => {
    const n = parseInt(s, 10);
    if (Number.isFinite(n) && !seen.has(n)) {
      seen.add(n);
      out.push(n);
    }
  });
  return out;
}

// Единый эндпоинт в двух режимах:
// 1) ?ids=1,2,5&lang=ru -> вернуть фильмы по id в заданной локали
// 2) ?title=матрица&lang=ru -> поиск по названию в заданной локали
app.get(`${API_PREFIX}/movies`, (req, res) => {
  const lang = pickLang(req);
  const dataset = loadMovies(lang);
  if (!dataset) {
    return res.status(500).json({

```



```

        error: lang === "ru" ? "Сервер: файл фильмов не найден" : "Server: movies
file missing",
        lang
    });
}

const { ids, title } = req.query;

// Режим 1: выбор по ids (для "перевода" текущей выдачи)
if (ids) {
    const idList = parseIdsParam(ids);
    const items = idList.map(id => dataset.find(m => m.id ===
id)).filter(Boolean);
    res.setHeader("Set-Cookie", `lang=${lang}; Path=/; Max-Age=31536000`);
    return res.json({ meta: { lang, mode: "ids", count: items.length, ids: idList
}, items });
}

// Режим 2: поиск по title
if (!title || !String(title).trim()) {
    const msg = lang === "ru" ? 'Требуется параметр "title"' : 'Query param
"title" is required';
    return res.status(400).json({ error: msg, lang });
}
const q = String(title).toLowerCase();
const items = dataset
    .filter(m => (m.title || "").toLowerCase().includes(q))
    .slice(0, DEFAULT_PAGE_SIZE);

res.setHeader("Set-Cookie", `lang=${lang}; Path=/; Max-Age=31536000`);
res.json({ meta: { lang, mode: "title", count: items.length }, items });
});

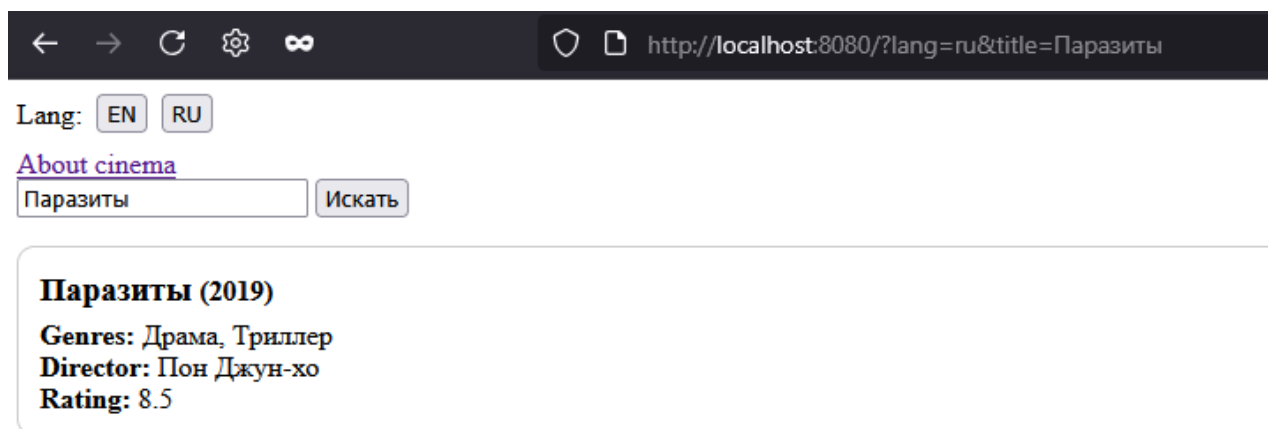
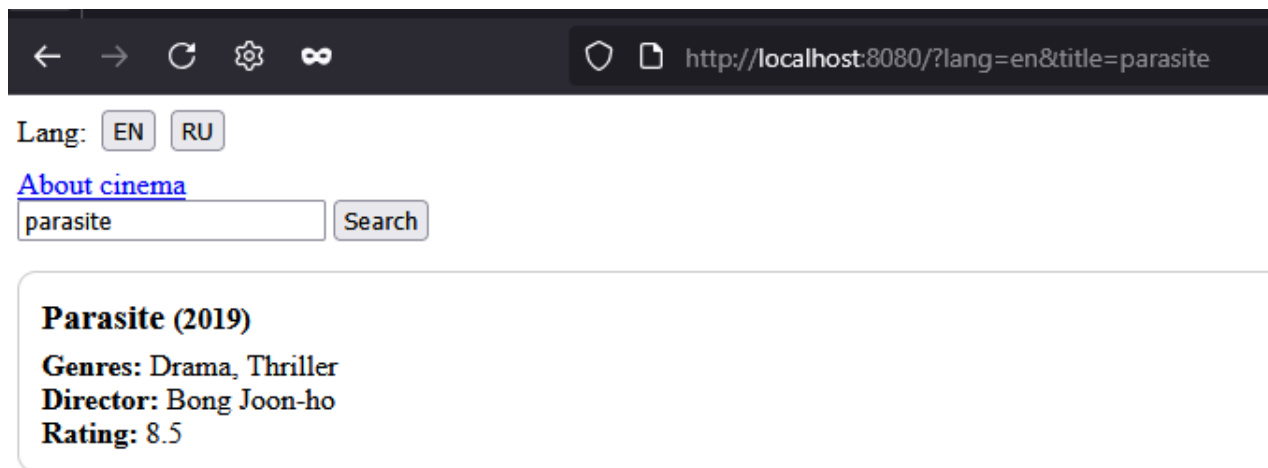
// 404 только для /api
app.use(API_PREFIX, (req, res) => res.status(404).json({ error: "Not found" }));

app.listen(PORT, () => {
    console.log(`Movies API :${PORT} prefix=${API_PREFIX}
defaultLang=${DEFAULT_LANG}`);
});

```

Здесь описана логика выбора языка

Пример работоспособности программы



Заключение

В ходе выполнения лабораторной работы мы ознакомились со способами отображения данных на различных языках при использовании файлов ресурсов