# Практическая работа №9 «Парсинг веб-сайтов с использованием JavaScript»

**Цель работы:** Научиться извлекать данные с веб-страниц с помощью JavaScript в консоли браузера, используя методы DOM API (querySelectorAll, forEach и другие).

## Задание 1: Основы работы с консолью браузера

Парсинг веб-сайтов — это процесс автоматического извлечения данных с веб-страниц. В этой работе мы будем использовать встроенные инструменты разработчика браузера (DevTools) и JavaScript для получения информации со страниц.

Как открыть консоль:

* Chrome/Edge: F12 или Ctrl+Shift+I (Windows) / Cmd+Option+I (Mac)
* Firefox: F12 или Ctrl+Shift+K (Windows) / Cmd+Option+K (Mac)
* Safari: Cmd+Option+C (Mac, предварительно включить меню «Разработка»)

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Содержимое, созданное искусственным интеллектом, может быть неверным.

Основные методы для работы с DOM:

* document.querySelector(selector) — находит первый элемент, соответствующий селектору
* document.querySelectorAll(selector) — находит все элементы, соответствующие селектору (возвращает NodeList)
* element.textContent — получает текстовое содержимое элемента
* element.innerHTML — получает HTML-содержимое элемента
* element.getAttribute(name) — получает значение атрибута элемента
* element.href — получает URL ссылки

Метод forEach для перебора элементов:  
После получения списка элементов через querySelectorAll, мы можем перебрать их с помощью метода forEach:

*Пример:*

// Найти все заголовки на странице  
const headings = document.querySelectorAll('h1, h2, h3');  
headings.forEach((heading, index) => {  
console.log(`Заголовок ${index + 1}: ${heading.textContent}`);  
});

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Задание 2: Парсинг главной страницы новостей

**Задача:** Извлечь данные со страницы https://www.spbstu.ru/media/news/

**Шаг 1: Откройте страницу в браузере**

Перейдите по адресу: https://www.spbstu.ru/media/news/

Изображение выглядит как текст, снимок экрана, Веб-сайт, веб-страница

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Шаг 2: Откройте консоль разработчика**

Нажмите F12 и перейдите на вкладку «Console».

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Шаг 3: Найдите заголовки новостей**

Используйте инструмент «Inspect Element» (правый клик → «Просмотреть код») или нажмите Ctrl+Shift+C, чтобы определить структуру HTML.

*Пример кода для извлечения заголовков:*

// Найти все заголовки новостей  
const newsTitles = document.querySelectorAll('селектор\_заголовков');  
const titlesArray = [];  
newsTitles.forEach((title) => {  
titlesArray.push(title.textContent.trim());  
});  
console.log('Заголовки новостей:', titlesArray);

**Задача:** Определите правильный селектор для заголовков новостей на странице и выполните код выше, заменив 'селектор\_заголовков' на реальный селектор.

Изображение выглядит как текст, программное обеспечение, веб-страница, Веб-сайт

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Шаг 4: Извлечь изображения новостей**

Найдите все изображения, связанные с новостями, и получите их URL-адреса.

*Пример кода:*

// Найти все изображения новостей  
const newsImages = document.querySelectorAll('селектор\_изображений');  
const imagesArray = [];  
newsImages.forEach((img) => {  
const imageUrl = img.src || img.getAttribute('src');  
const imageAlt = img.alt || 'Нет описания';  
imagesArray.push({

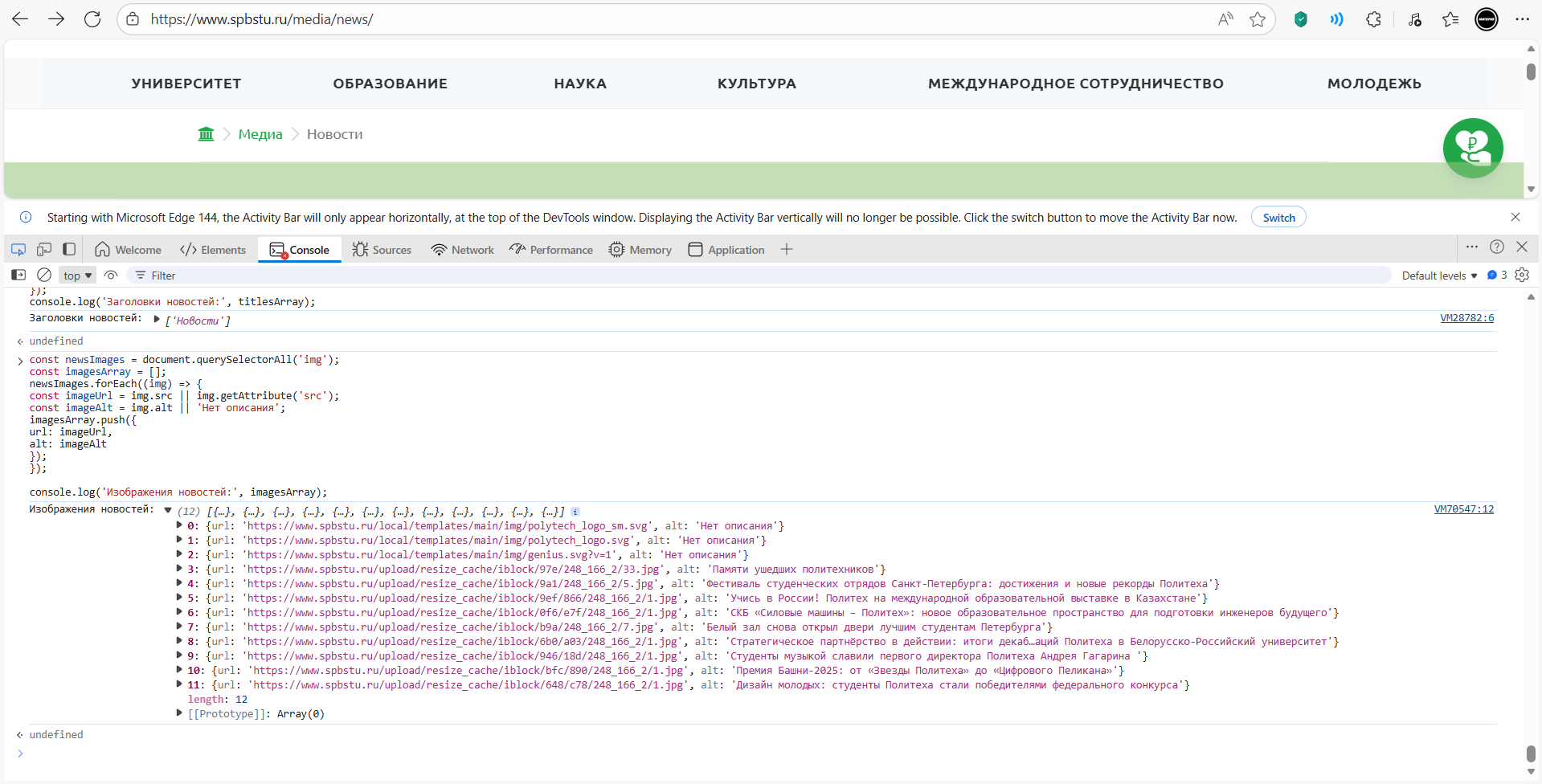
url: imageUrl,

alt: imageAlt

});

});

console.log('Изображения новостей:', imagesArray);

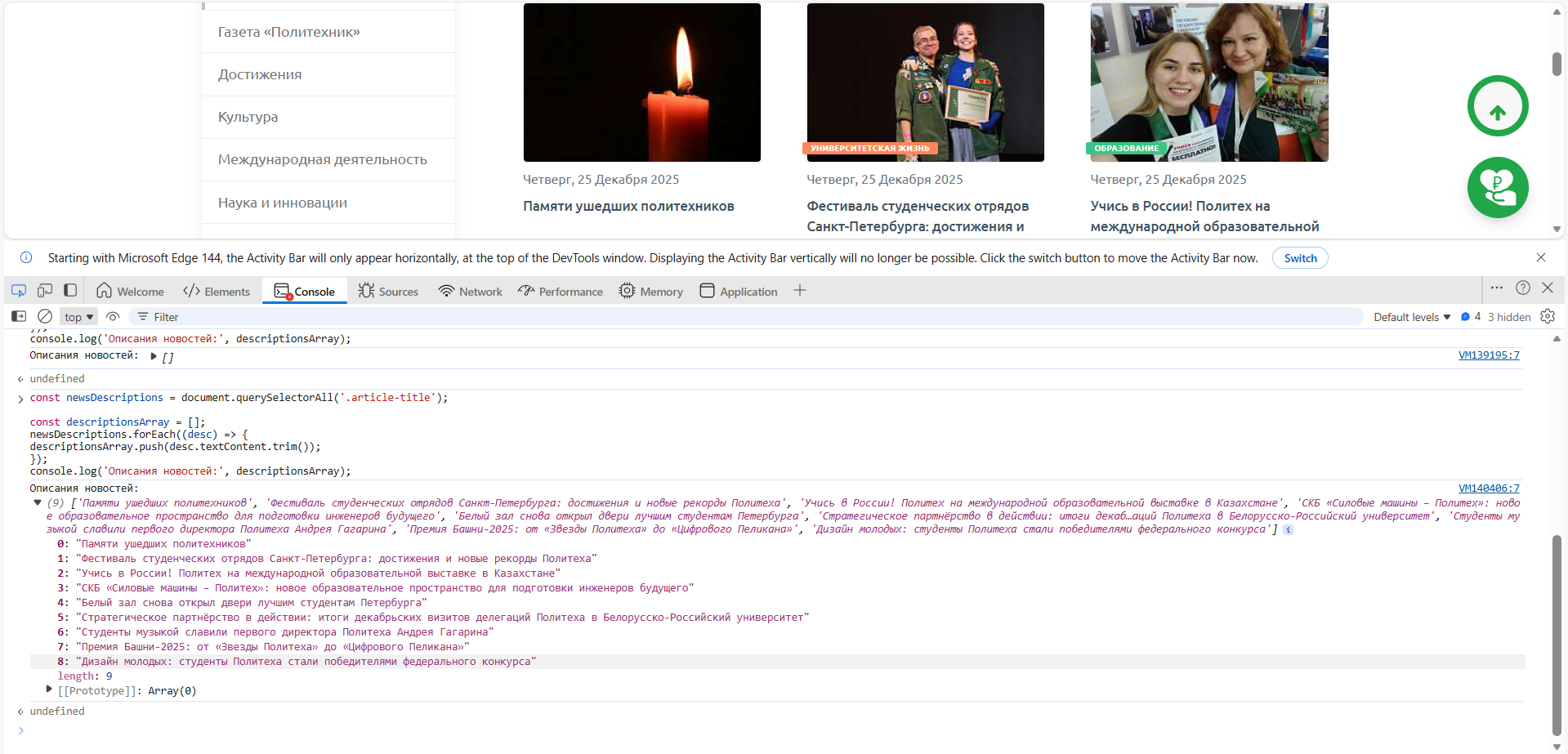


**Шаг 5: Извлечь расширенные заголовки (аннотации/описания)**

Многие новостные сайты имеют краткие описания под заголовками. Найдите эти элементы и извлеките их текст.

*Пример кода:*

// Найти описания новостей  
const newsDescriptions = document.querySelectorAll('селектор\_описаний');  
const descriptionsArray = [];  
newsDescriptions.forEach((desc) => {  
descriptionsArray.push(desc.textContent.trim());  
});  
console.log('Описания новостей:', descriptionsArray);



**Шаг 6: Собрать все данные в один объект**

Объедините все извлеченные данные в структурированный формат.

*Пример кода:*

// Собрать все данные о новостях  
const newsItems = document.querySelectorAll('селектор\_контейнера\_новости');  
const allNews = [];

newsItems.forEach((item, index) => {  
const title = item.querySelector('селектор\_заголовка')?.textContent.trim() || 'Нет заголовка';  
const image = item.querySelector('селектор\_изображения')?.src || 'Нет изображения';  
const description = item.querySelector('селектор\_описания')?.textContent.trim() || 'Нет описания';  
const link = item.querySelector('a')?.href || 'Нет ссылки';

allNews.push({

id: index + 1,

title: title,

image: image,

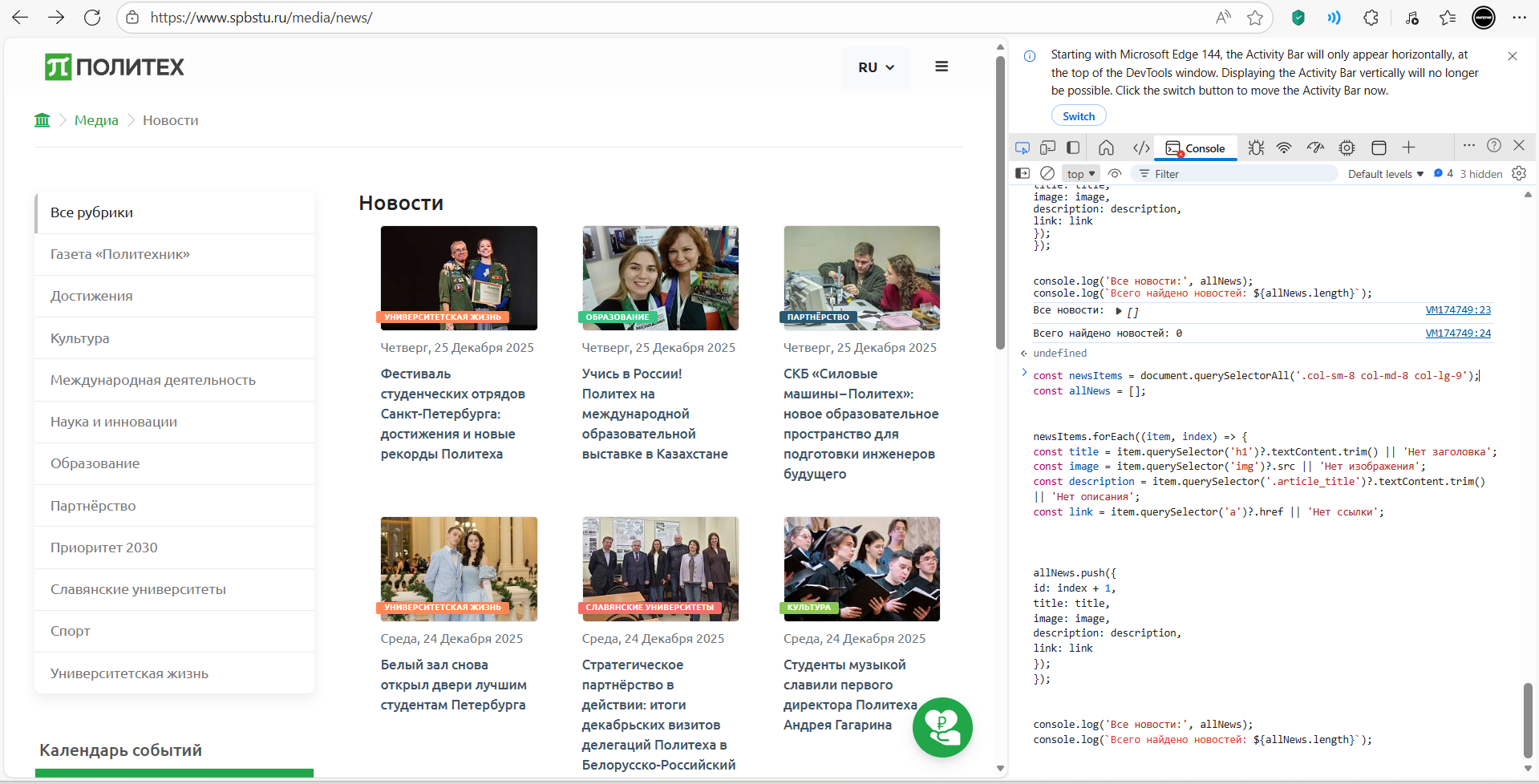
description: description,

link: link

});

});

console.log('Все новости:', allNews);  
console.log(`Всего найдено новостей: ${allNews.length}`);



Честно говоря, не до конца понимаю, что не так, я опробовал несколько вариантов селекторов и классов, которые были найдены мной при исследовании нашего сайта, но я так и не понял причину.

## Задание 3: Парсинг отдельной страницы новости

**Задача:** Перейти на страницу отдельной новости и извлечь детальную информацию.

**Шаг 1: Получить ссылки на новости**

Сначала извлеките все ссылки на отдельные новости с главной страницы:

// Получить все ссылки на новости  
const newsLinks = [];  
document.querySelectorAll('селектор\_ссылок\_на\_новости').forEach((link) => {  
const href = link.href || link.getAttribute('href');  
if (href && href.includes('/media/news/')) {  
newsLinks.push(href);  
}  
});  
console.log('Ссылки на новости:', newsLinks);

Изображение выглядит как текст, Человеческое лицо, снимок экрана, одежда

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Шаг 2: Перейти на страницу новости**

Выберите одну из ссылок и перейдите на страницу новости (можно открыть в новой вкладке).

Изображение выглядит как текст, снимок экрана, человек, Веб-сайт

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Шаг 3: Найти контент новости**

Определите, в каких HTML-тегах находится основной контент статьи (обычно это <article>, <div class="content">, <p> и т.д.).

*Пример кода для извлечения контента:*

// Найти основной контент новости  
const contentElements = document.querySelectorAll('селектор\_контента');  
const articleContent = [];

contentElements.forEach((element) => {  
const tagName = element.tagName.toLowerCase();  
const text = element.textContent.trim();

if (text.length > 0) {

articleContent.push({

tag: tagName,

text: text,

html: element.innerHTML

});

}

});

console.log('Контент новости по тегам:', articleContent);

**Шаг 4: Извлечь время публикации**

Найдите элемент, содержащий дату и время публикации новости.

*Пример кода:*

// Найти дату и время публикации  
const dateElement = document.querySelector('селектор\_даты');  
const publicationDate = dateElement ? dateElement.textContent.trim() : 'Дата не найдена';  
console.log('Дата публикации:', publicationDate);

// Альтернативный вариант: найти все элементы с датами  
const allDates = [];  
document.querySelectorAll('селектор\_даты').forEach((dateEl) => {  
allDates.push({

text: dateEl.textContent.trim(),

datetime: dateEl.getAttribute('datetime') || 'Нет атрибута datetime'

});

});

console.log('Все даты на странице:', allDates);

Изображение выглядит как текст, снимок экрана, Веб-сайт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Та же самая проблема, что и в прошлом задании. По сей видимости, не до конца понимаю, что из них верное.

**Шаг 5: Собрать полную информацию о новости**

Создайте объект, содержащий всю информацию о новости.

*Пример кода:*

// Собрать полную информацию о новости  
const fullNewsInfo = {  
title: document.querySelector('селектор\_заголовка\_новости')?.textContent.trim() || 'Нет заголовка',  
date: document.querySelector('селектор\_даты')?.textContent.trim() || 'Нет даты',

content: [],

images: [],

author: document.querySelector('селектор\_автора')?.textContent.trim() || 'Автор не указан'  
};

// Извлечь весь текстовый контент  
document.querySelectorAll('селектор\_контента p, селектор\_контента div').forEach((p) => {  
const text = p.textContent.trim();

if (text.length > 50) { // Только параграфы с достаточным количеством текста

fullNewsInfo.content.push({

tag: p.tagName.toLowerCase(),

text: text

});

}

});

// Извлечь все изображения из статьи  
document.querySelectorAll('селектор\_контента img').forEach((img) => {  
fullNewsInfo.images.push({

src: img.src,

alt: img.alt || 'Нет описания'

});

});

console.log('Полная информация о новости:', fullNewsInfo);

## Задание 4: Автоматизация парсинга нескольких новостей

**Задача:** Создать функцию, которая автоматически извлекает данные с нескольких страниц новостей.

**Шаг 1: Создать функцию для парсинга одной новости**

Напишите функцию, которая принимает URL новости и возвращает объект с данными (примечание: для работы с несколькими страницами потребуется более сложный подход, но мы можем создать функцию для текущей страницы):

function parseNewsPage() {  
const newsData = {

url: window.location.href,

title: document.querySelector('селектор\_заголовка')?.textContent.trim() || '',  
date: document.querySelector('селектор\_даты')?.textContent.trim() || '',

content: [],

images: []

};

// Извлечь контент  
document.querySelectorAll('селектор\_контента p').forEach((p) => {  
const text = p.textContent.trim();  
if (text.length > 0) {  
newsData.content.push(text);  
}  
});

// Извлечь изображения  
document.querySelectorAll('селектор\_контента img').forEach((img) => {  
newsData.images.push({

src: img.src,

alt: img.alt || ''

});

});

return newsData;

}

// Вызвать функцию  
const news = parseNewsPage();  
console.log('Данные новости:', news);

**Шаг 2: Экспорт данных в JSON**

Преобразуйте извлеченные данные в формат JSON для дальнейшего использования:

// Преобразовать данные в JSON  
const jsonData = JSON.stringify(news, null, 2);  
console.log('JSON данные:', jsonData);

// Скопировать в буфер обмена (работает не во всех браузерах)  
// navigator.clipboard.writeText(jsonData);

**Шаг 3: Сохранение данных**

Выведите данные в консоль в удобном формате для копирования:

// Красивый вывод данных  
console.table(news); // Табличный формат (если структура подходит)  
console.log(JSON.stringify(news, null, 2)); // JSON с отступами

## Задание 5: Дополнительные задачи

**Задача 1: Найти все ссылки на внешние ресурсы**

Извлеките все внешние ссылки (ссылки, ведущие на другие домены) со страницы новости.

// Найти все внешние ссылки  
const externalLinks = [];  
document.querySelectorAll('a').forEach((link) => {  
const href = link.href;  
if (href && !href.includes('spbstu.ru') && href.startsWith('http')) {  
externalLinks.push({

text: link.textContent.trim(),

url: href

});

}

});

console.log('Внешние ссылки:', externalLinks);

**Задача 2: Подсчитать количество слов в статье**

Подсчитайте общее количество слов в тексте новости.

// Подсчитать слова в статье  
const allText = document.querySelector('селектор\_контента')?.textContent || '';  
const words = allText.split(/\s+/).filter(word => word.length > 0);  
console.log(`Количество слов в статье: ${words.length}`);

**Задача 3: Найти все цитаты**

Найдите все элементы <blockquote> или элементы с классом, указывающим на цитату.

// Найти все цитаты  
const quotes = [];  
document.querySelectorAll('blockquote, .quote, [class\*="quote"]').forEach((quote) => {  
quotes.push(quote.textContent.trim());  
});  
console.log('Цитаты:', quotes);

Варианты заданий:

## Вариант 7: Парсинг страницы с вакансиями

Сайт: Раздел с вакансиями

Задачи:

* Извлечь названия вакансий
* Найти требования к кандидатам
* Извлечь информацию о зарплате (если указана)
* Найти контактную информацию для отклика

## Критерии оценки практической работы:

Техническая правильность (40%):

* Корректное использование querySelectorAll и forEach
* Правильное определение селекторов для элементов страницы
* Корректное извлечение данных (текст, атрибуты, ссылки)
* Отсутствие ошибок в консоли при выполнении кода

Полнота выполнения (30%):

* Выполнены все задания из варианта
* Извлечены все требуемые данные (заголовки, изображения, описания, контент, даты)
* Данные собраны в структурированный формат (объекты, массивы)
* Выполнены дополнительные задачи (если указаны в варианте)

Качество кода (20%):

* Код читаемый и понятный
* Использованы комментарии для объяснения логики
* Данные структурированы логично
* Применены современные методы JavaScript (стрелочные функции, деструктуризация, если уместно)

Оформление отчета (10%):

* Скриншоты работы кода в консоли
* Примеры извлеченных данных
* Описание найденных селекторов
* Выводы о структуре HTML страницы

## Содержание отчета:

1. Титульный лист с указанием номера работы, темы, ФИО студента, группы
2. Цель работы
3. Описание выбранного сайта

* URL сайта
* Краткое описание структуры страницы
* Скриншот главной страницы

1. Задание 1: Парсинг главной страницы

* Скриншот консоли с кодом для извлечения заголовков
* Список найденных заголовков (первые 5-10)
* Скриншот консоли с кодом для извлечения изображений
* Список URL изображений (первые 5)
* Скриншот консоли с кодом для извлечения описаний
* Примеры описаний

1. Задание 2: Парсинг отдельной страницы

* URL выбранной новости/страницы
* Скриншот консоли с кодом для извлечения контента
* Описание структуры HTML (какие теги содержат контент)
* Примеры извлеченного контента (первые 3-5 параграфов)
* Скриншот консоли с кодом для извлечения даты
* Найденная дата публикации

1. Задание 3: Структурирование данных

* Код функции для сбора всех данных
* Скриншот консоли с результатом выполнения
* Пример JSON-структуры данных (минимум 2-3 новости/элемента)

1. Дополнительные задания (если выполнялись)

* Код для выполнения дополнительных задач
* Скриншоты результатов

1. Выводы

* Какие методы DOM API были использованы
* Какие селекторы оказались наиболее эффективными
* С какими трудностями столкнулись при парсинге
* Как можно улучшить код для более эффективного парсинга

1. Приложение

* Полный код всех скриптов, использованных в работе
* Примеры извлеченных данных в текстовом формате

## Примечания:

* Все скриншоты должны быть четкими и читаемыми
* Код в скриншотах должен быть виден полностью
* При копировании кода из консоли убедитесь, что он корректно отформатирован
* Если селекторы не работают, объясните, как вы их определяли и какие использовали альтернативы

## Ссылки для изучения:

* MDN Web Docs: Document.querySelectorAll() - https://developer.mozilla.org/ru/docs/Web/API/Document/querySelectorAll
* MDN Web Docs: NodeList.forEach() - https://developer.mozilla.org/ru/docs/Web/API/NodeList/forEach
* JavaScript.info: Работа с DOM - https://learn.javascript.ru/dom-nodes
* Chrome DevTools: Работа с консолью - https://developer.chrome.com/docs/devtools/console/