**Приложение Ж. Листинг Java Script**

|  |
| --- |
| document.addEventListener('DOMContentLoaded', () => {    const promotionTitles = document.querySelectorAll('.promotion-item h2'); // Или '.promotion-item-content h2'    promotionTitles.forEach(title => {      const containerWidth = title.offsetWidth;      const textWidth = title.scrollWidth;      if (textWidth > containerWidth) {        // Если текст не помещается, уменьшаем размер шрифта        let currentFontSize = parseFloat(window.getComputedStyle(title, null).getPropertyValue('font-size'));        const step = 0.1; // Шаг уменьшения размера шрифта        while (title.scrollWidth > title.offsetWidth && currentFontSize > 1) { // Уменьшаем до 1rem (или другого минимального значения)          currentFontSize -= step;          title.style.fontSize = `${currentFontSize}rem`;        }      }    });  }); |

Листинг 1 – Адаптивный шрифт

|  |
| --- |
| document.addEventListener('DOMContentLoaded', function() {      const addressesButton = document.querySelector('.addresses button');      const addressesPopup = document.getElementById('addresses-popup');      const closeButton = document.querySelector('#addresses-popup .close-button');      const popupContent = document.querySelector('#addresses-popup .popup-content');      if (addressesButton && addressesPopup && popupContent && closeButton) {          addressesButton.addEventListener('click', function() {              addressesPopup.style.display = 'flex';              setTimeout(() => addressesPopup.classList.add('show'), 0);          });          closeButton.addEventListener('click', function() {              addressesPopup.classList.remove('show');              setTimeout(() => addressesPopup.style.display = 'none', 300);          });          window.addEventListener('click', function(event) {              if (event.target === addressesPopup) {                  addressesPopup.classList.remove('show');                  setTimeout(() => addressesPopup.style.display = 'none', 300);              }          });      }  }); |

Листинг 2 – Выезжающее меню

|  |
| --- |
| class BannerSlider {      constructor() {          this.currentSlide = 0;          this.slides = [              {                  image: 'media/SpringProm.png',                  alt: 'Весенние акции'              },              {                  image: 'media/StirkaProm.png',                  alt: 'Реклама Стиралки'              },              {                  image: 'media/CreditProm.png',                  alt: 'Реклама кредита'              },              {                  image: 'media/TefalProm.png',                  alt: 'Реклама Tefal'              },              {                  image: 'media/LgBanner.png',                  alt: 'Реклама ЛГ'              }          ];          this.isAnimating = false;          this.init();      }      init() {          this.container = document.querySelector('.banner-image');          this.prevBtn = document.querySelector('.banner-nav.prev');          this.nextBtn = document.querySelector('.banner-nav.next');          this.indicators = document.querySelector('.slide-indicators');          this.images = this.slides.map((slide, index) => {              const img = document.createElement('img');              img.src = slide.image;              img.alt = slide.alt;              img.className = index === 0 ? 'active' : 'next';              this.container.appendChild(img);              return img;          });          this.createIndicators();          this.setupEventListeners();          this.updateSlidePositions();      }      createIndicators() {          this.indicators.innerHTML = '';          this.slides.forEach((\_, index) => {              const dot = document.createElement('div');              dot.classList.add('slide-indicator');              if (index === 0) dot.classList.add('active');              dot.addEventListener('click', () => this.goToSlide(index));              this.indicators.appendChild(dot);          });      }      setupEventListeners() {          this.prevBtn.addEventListener('click', () => this.prev());          this.nextBtn.addEventListener('click', () => this.next());      }      updateIndicators() {          const dots = this.indicators.children;          for (let i = 0; i < dots.length; i++) {              dots[i].classList.toggle('active', i === this.currentSlide);          }      }      updateSlidePositions() {          this.images.forEach((img, index) => {              if (index === this.currentSlide) {                  img.className = 'active';              } else {                  img.className = 'hidden';              }          });          // Only show next and prev slides when not animating          const nextIndex = (this.currentSlide + 1) % this.slides.length;          const prevIndex = (this.currentSlide - 1 + this.slides.length) % this.slides.length;            if (!this.isAnimating) {              this.images[nextIndex].className = 'next';              this.images[prevIndex].className = 'prev';          }      }      async slide(direction) {          if (this.isAnimating) return;          this.isAnimating = true;          const nextIndex = direction === 'next'              ? (this.currentSlide + 1) % this.slides.length              : (this.currentSlide - 1 + this.slides.length) % this.slides.length;          // Reset all slides first          this.images.forEach(img => {              img.className = 'hidden';          });          const currentSlide = this.images[this.currentSlide];          const nextSlide = this.images[nextIndex];          // Set initial positions          currentSlide.className = 'active';          nextSlide.className = direction === 'next' ? 'next' : 'prev';          // Force browser reflow          void nextSlide.offsetWidth;          // Trigger animation          requestAnimationFrame(() => {              currentSlide.className = direction === 'next' ? 'slide-left' : 'slide-right';              nextSlide.className = 'active';              setTimeout(() => {                  this.currentSlide = nextIndex;                  this.updateIndicators();                  this.updateSlidePositions();                  this.isAnimating = false;              }, 600);          });      }      async goToSlide(index) {          if (this.isAnimating || index === this.currentSlide) return;          const direction = index > this.currentSlide ? 'next' : 'prev';          await this.slide(direction);      }      prev() {          this.slide('prev');      }      next() {          this.slide('next');      }  }  // Инициализация слайдера  document.addEventListener('DOMContentLoaded', () => {      new BannerSlider();  }); |

Листинг 3 – Баннер-слайдер

|  |
| --- |
| document.addEventListener('DOMContentLoaded', function() {    const orderFormPopup = document.getElementById('order-form-popup');    const orderForm = document.getElementById('order-form');    const orderSuccessDiv = document.getElementById('order-success');    const orderNumberSpan = document.getElementById('order-number');    const orderFormCloseButton = document.querySelector('.order-form-close');    const sortBySelect = document.getElementById('sort-by');    // Элементы каталога    const catalogGrid = document.querySelector('.catalog-page .catalog-grid');    const categoryLinks = document.querySelectorAll('.left-menu ul li a');    const catalogTitle = document.querySelector('.catalog-page h1');    const paginationContainer = document.createElement('div');    paginationContainer.classList.add('pagination');    if (catalogGrid && catalogGrid.parentNode) {      catalogGrid.parentNode.insertBefore(paginationContainer, catalogGrid.nextSibling);    }    let allItems = [];    const itemsPerPage = 15;    let currentPage = 1;    let currentCategory = 'televisors'; // По умолчанию    const getPrice = (item) => {      const priceText = item.querySelector('price').textContent.trim();      const numericPrice = priceText.replace(/[^\d.]/g, '');      return parseFloat(numericPrice);    };    function displayItems(items, page, sortBy = 'default') {      if (!catalogGrid) return;      const itemsToSort = [...items];      switch (sortBy) {        case 'price-asc':          itemsToSort.sort((a, b) => getPrice(a) - getPrice(b));          break;        case 'price-desc':          itemsToSort.sort((a, b) => getPrice(b) - getPrice(a));          break;        case 'default':          break;      }      catalogGrid.innerHTML = '';      const startIndex = (page - 1) \* itemsPerPage;      const endIndex = startIndex + itemsPerPage;      const itemsToDisplay = itemsToSort.slice(startIndex, endIndex);      itemsToDisplay.forEach(item => {        const imageSrc = item.querySelector('image').textContent;        const altText = item.querySelector('alt').textContent;        const titleText = item.querySelector('title').textContent;        const priceText = item.querySelector('price').textContent;        const id = item.querySelector('id')?.textContent || titleText.replace(/\s+/g, '-').toLowerCase();        const productCard = document.createElement('div');        productCard.classList.add('product-card');        productCard.innerHTML = `          <img src="${imageSrc}" alt="${altText}">          <h3>${titleText}</h3>          <p class="price">${priceText}</p>          <button class="buy-button" data-product-id="${id}" data-product-name="${titleText}">Купить</button>        `;        catalogGrid.appendChild(productCard);      });      attachBuyButtonListeners();    }    function renderPagination(totalItems) {      if (!paginationContainer) return;      paginationContainer.innerHTML = '';      const totalPages = Math.ceil(totalItems / itemsPerPage);      for (let i = 1; i <= totalPages; i++) {        const pageButton = document.createElement('button');        pageButton.textContent = i;        if (i === currentPage) {          pageButton.classList.add('active');        }        pageButton.addEventListener('click', function() {          currentPage = i;          displayItems(allItems, currentPage, sortBySelect ? sortBySelect.value : 'default');          renderPagination(allItems.length);        });        paginationContainer.appendChild(pageButton);      }    }    function loadCatalog(xmlFile, categoryTitle) {      if (!catalogTitle) return;      currentCategory = xmlFile.replace('.xml', '');      catalogTitle.textContent = categoryTitle;      fetch('xml/' + xmlFile)        .then(response => response.text())        .then(xmlString => {          const parser = new DOMParser();          const xmlDoc = parser.parseFromString(xmlString, 'application/xml');          allItems = Array.from(xmlDoc.querySelectorAll('item'));          currentPage = 1;          displayItems(allItems, currentPage, sortBySelect ? sortBySelect.value : 'default');          renderPagination(allItems.length);        })        .catch(error => {          console.error('Ошибка загрузки XML:', error);          if (catalogGrid) catalogGrid.innerHTML = '<p>Ошибка загрузки каталога.</p>';          if (paginationContainer) paginationContainer.innerHTML = '';        });    }    function attachBuyButtonListeners() {      const buyButtons = document.querySelectorAll('.buy-button');      buyButtons.forEach(button => {        button.addEventListener('click', openOrderForm);      });      console.log(`Найдено и обработано ${buyButtons.length} кнопок "Купить".`);    }    function openOrderForm(event) {      if (!orderFormPopup) {        console.error('Элемент формы заказа (#order-form-popup) не найден.');        return;      }      const productId = event.target.dataset.productId;      const productName = event.target.dataset.productName;      console.log(`Открываю форму заказа для товара: ${productName} (ID: ${productId})`);      orderFormPopup.style.display = 'flex';      if (orderSuccessDiv) {        orderSuccessDiv.style.display = 'none';      }      if (orderForm) {        orderForm.reset();      }    }    function closeOrderForm() {      if (orderFormPopup) {        orderFormPopup.style.display = 'none';      }    }    function submitOrderForm(event) {      event.preventDefault();      if (!orderForm) {        console.error('Элемент формы (#order-form) не найден.');        return;      }      const formData = new FormData(orderForm);      const orderDetails = {};      formData.forEach((value, key) => {        orderDetails[key] = value;      });      orderDetails.type = 'order';      console.log('Данные заказа для отправки:', orderDetails);      fetch('http://localhost:3000/submit-application', {        method: 'POST',        headers: {          'Content-Type': 'application/json',        },        body: JSON.stringify(orderDetails),      })      .then(response => {        if (!response.ok) {          if (response.status === 409) {            return response.json().then(errorData => {              throw new Error(errorData.error || 'Заявка с такими данными уже была отправлена.');            });          }          throw new Error(`HTTP error! status: ${response.status}`);        }        return response.json();      })      .then(data => {        console.log('Ответ от сервера:', data);        if (orderNumberSpan) {          orderNumberSpan.textContent = data.id;        }        if (orderFormPopup) {          orderFormPopup.style.display = 'none';        }        if (orderSuccessDiv) {          orderSuccessDiv.style.display = 'block';        }        orderForm.reset();        setTimeout(() => {          if (orderSuccessDiv) {            orderSuccessDiv.style.display = 'none';          }        }, 3000);      })      .catch(error => {        console.error('Ошибка при отправке заказа:', error);        alert(error.message);      });    }    // Обработчики событий для формы заказа    if (orderFormCloseButton) {      orderFormCloseButton.addEventListener('click', closeOrderForm);    }    window.addEventListener('click', function(event) {      if (event.target === orderFormPopup) {        closeOrderForm();      }    });    if (orderForm) {      orderForm.addEventListener('submit', submitOrderForm);    }    // Обработчик событий для сортировки    if (sortBySelect) {      sortBySelect.addEventListener('change', function() {        const selectedSort = this.value;        currentPage = 1;        displayItems(allItems, currentPage, selectedSort);        renderPagination(allItems.length);      });    }    // Загрузка каталога по умолчанию    loadCatalog('televisors.xml', 'Телевизоры');    categoryLinks.forEach((link, index) => {      let category = '';      switch (index) {        case 0: category = 'televisors'; break;        case 1: category = 'fridges'; break;        case 2: category = 'stirka'; break;        case 3: category = 'pilesos'; break;        case 4: category = 'microwave'; break;        case 5: category = 'dishwasher'; break;        case 6: category = 'duhovki'; break;        case 7: category = 'plates'; break;        case 8: category = 'dishes'; break;        case 9: category = 'gadjets'; break;        default: category = 'televisors'; break;      }      link.addEventListener('click', function(event) {        event.preventDefault();        let xmlFile = '';        let title = '';        switch (category) {          case 'pilesos': xmlFile = 'pilesos.xml'; title = 'Пылесосы'; break;          case 'televisors': xmlFile = 'televisors.xml'; title = 'Телевизоры'; break;          case 'fridges': xmlFile = 'fridges.xml'; title = 'Холодильники'; break;          case 'stirka': xmlFile = 'stirka.xml'; title = 'Стиральные машины'; break;          case 'microwave': xmlFile = 'microwaves.xml'; title = 'Микроволновки'; break;          case 'dishwasher': xmlFile = 'dishwashers.xml'; title = 'Посудомойки'; break;          case 'duhovki': xmlFile = 'duhovki.xml'; title = 'Духовки'; break;          case 'plates': xmlFile = 'plates.xml'; title = 'Кухонные плиты'; break;          case 'dishes': xmlFile = 'dishes.xml'; title = 'Кухонная утварь'; break;          case 'gadjets': xmlFile = 'gadjets.xml'; title = 'Гаджеты'; break;          default: xmlFile = 'televisors.xml'; title = 'Телевизоры'; break;        }        loadCatalog(xmlFile, title);      });    });  }); |

Листинг 4 – Скрипт для каталога и заказов

|  |
| --- |
| document.addEventListener('DOMContentLoaded', function() {      const menuIcon = document.querySelector('.mobile-logo');      const mobileMenuContainer = document.querySelector('.mobile-menu-container');      const mobileMenuWrapper = document.querySelector('.mobile-menu-wrapper');      const mobileMenu = document.querySelector('.mobile-menu');      const closeMenuButton = document.querySelector('.close-menu-button');      const catalogLink = document.querySelector('.catalog-link');      const categoriesMenu = document.querySelector('.mobile-menu-categories'); // Получаем элемент      const backButton = document.querySelector('.back-button');      // Проверяем, был ли найден элемент categoriesMenu      if (categoriesMenu) {          categoriesMenu.classList.remove('show');      }      if (menuIcon) {          menuIcon.addEventListener('click', function() {              mobileMenuContainer.style.display = 'block';              mobileMenu.classList.add('show');              if (categoriesMenu) {                  categoriesMenu.classList.remove('show');              }          });      }      if (closeMenuButton) {          closeMenuButton.addEventListener('click', function() {              mobileMenu.classList.remove('show');              if (categoriesMenu) {                  categoriesMenu.classList.remove('show');              }              setTimeout(function() {                  mobileMenuContainer.style.display = 'none';              }, 300);          });      }      if (catalogLink) {          catalogLink.addEventListener('click', function(e) {              e.preventDefault();              if (categoriesMenu) {                  categoriesMenu.classList.add('show');              }          });      }      if (backButton) {          backButton.addEventListener('click', function(e) {              e.preventDefault();              if (categoriesMenu) {                  categoriesMenu.classList.remove('show');              }          });      }      if (mobileMenuContainer) {          mobileMenuContainer.addEventListener('click', function(e) {              if (e.target === mobileMenuContainer) {                  mobileMenu.classList.remove('show');                  if (categoriesMenu) {                      categoriesMenu.classList.remove('show');                  }                  setTimeout(function() {                      mobileMenuContainer.style.display = 'none';                  }, 300);              }          });      }  }); |

Листинг 5 – Бургер-меню

|  |
| --- |
| document.addEventListener('DOMContentLoaded', function() {      const openFormButton = document.querySelector('#jobs-btn button');      const jobsPopup = document.querySelector('#jobs-popup');      const closeFormButton = document.querySelector('.close-form-button');      const jobForm = document.querySelector('#job-form');      const jobSuccess = document.querySelector('#job-success');        if (openFormButton && jobsPopup && closeFormButton && jobForm) {        openFormButton.addEventListener('click', function() {          jobsPopup.style.display = 'flex';        });          closeFormButton.addEventListener('click', function() {          jobsPopup.style.display = 'none';        });          window.addEventListener('click', function(event) {          if (event.target === jobsPopup) {            jobsPopup.style.display = 'none';          }        });          jobForm.addEventListener('submit', function(event) {          event.preventDefault();            const formData = new FormData(jobForm);          const data = {};          formData.forEach((value, key) => {            data[key] = value;          });            fetch('http://localhost:3000/submit-application', {            method: 'POST',            headers: {              'Content-Type': 'application/json',            },            body: JSON.stringify(data),          })          .then(response => {            if (!response.ok) {              throw new Error(`HTTP error! status: ${response.status}`);            }            return response.text(); // Или response.json(), если сервер возвращает JSON          })          .then(responseData => {            console.log('Ответ от сервера:', responseData);            jobsPopup.style.display = 'none';            if (jobSuccess) {              jobSuccess.style.display = 'block';            }            jobForm.reset();          })          .catch(error => {            console.error('Ошибка при отправке данных на сервер:', error);          });        });      } else {        console.error('Одна или несколько необходимых кнопок или элементов формы не найдены в DOM.');      }    }); |

Листинг 6 – Форма вакансий

|  |
| --- |
| const http = require('http');  const fs = require('fs').promises;  const path = require('path');  const hostname = 'localhost';  const port = 3000;  const dataFilePath = path.join(\_\_dirname, '../', 'data', 'form-data.json');  function generateId() {    return Math.floor(Math.random() \* 100000000).toString().padStart(8, '0');  }  const server = http.createServer(async (req, res) => {    res.setHeader('Access-Control-Allow-Origin', '\*');    res.setHeader('Access-Control-Allow-Methods', 'POST, OPTIONS');    res.setHeader('Access-Control-Allow-Headers', 'Content-Type');    if (req.method === 'OPTIONS') {      res.writeHead(200);      res.end();      return;    }    if (req.method === 'POST' && req.url === '/submit-application') {      let body = '';      req.on('data', chunk => {        body += chunk.toString();      });      req.on('end', async () => {        try {          const applicationData = JSON.parse(body);          const applicationId = generateId();          const applicationType = applicationData.type || 'vacancy';          const formDataWithIdAndType = {            id: applicationId,            type: applicationType,            ...applicationData          };          const existingData = await fs.readFile(dataFilePath, 'utf8').catch(() => '[]');          const allFormData = JSON.parse(existingData);          // Проверка на дублирование по типу, имени, телефону и email          const isDuplicate = allFormData.some(item =>            item.type === formDataWithIdAndType.type &&            item.name === formDataWithIdAndType.name &&            item.phone === formDataWithIdAndType.phone &&            item.email === formDataWithIdAndType.email          );          if (!isDuplicate) {            allFormData.push(formDataWithIdAndType);            await fs.writeFile(dataFilePath, JSON.stringify(allFormData, null, 2));            res.writeHead(200, { 'Content-Type': 'application/json' });            res.end(JSON.stringify({ message: 'Заявка успешно сохранена!', id: applicationId }));          } else {            res.writeHead(409, { 'Content-Type': 'application/json' });            res.end(JSON.stringify({ error: `Заявка (${applicationType}) с таким именем, телефоном и email уже существует.` }));          }        } catch (error) {          console.error('Ошибка при обработке данных:', error);          res.writeHead(500, { 'Content-Type': 'application/json' });          res.end(JSON.stringify({ error: 'Не удалось сохранить заявку.' }));        }      });    } else {      res.writeHead(404, { 'Content-Type': 'text/plain' });      res.end('Not Found');    }  });  server.listen(port, hostname, () => {    console.log(`Сервер запущен на http://${hostname}:${port}/`);  }); |

Листинг 7 – Сервер для данных