|  |  |
| --- | --- |
| Министерство науки и высшего образования Российской Федерации  Федеральное государственное бюджетное образовательное  учреждение высшего образования  «Вятский государственный университет»  Институт математики и информационных систем  Факультет автоматики и вычислительной техники  Кафедра систем автоматизации управления | |
| Изучение методов работы компонентов PWA приложений  Отчет по практической работе № 4  по дисциплине  «Инфокоммуникационные системы и сети» | |
|  | Выполнили  студенты гр. ИТб-3301-01-00:  Прозоров С. К.  Акатов Д. В.  Проверил:  Земцов М. А. |
|  |
|  |
|  |
|  |
| Киров 2020 | |

### 1. Service Worker

В рамках задания была изучена предложенная документация по компоненту Service Worker.

Ответы на вопроосы:

1.1 Какие технологии лежат в основе Service Worker?

* JavaScript;
* архитектура на промисах (Promise);
* обязательность HTTPS.

1.2 Как работает Service Worker?

Сперва веб-страница регистрирует service worker, используя .js-файл, содержащий код сервис-воркера. Service worker устанавливается, активируется и начинает в фоновом режиме прослушивать события "fetch" и "message" и при необходимости их изменять или отменять совсем. Также, при обработке события install в коде сервис-воркера происходит открытие кэша и кэширование необходимых материалов. Этот функционал в дальнейшем позволит веб-приложению работать оффлайн.

1.3 Возможно ли проксирование запросов с разных удаленных серверов с разными IP адресами? Почему и как можно реализовать?

Нет, Service Worker работает только с тем адресом, с которого загружен его код. То есть можно перехватывать только запросы к своему домену. Притом, данная функция доступна только для не кросдоменных запросов (CORS). Для проксирования запросов с чужих серверов в 2016 году Google предлагал введение события "foreignfetch", но на данный момент эту идею полностью закрыли. Возможно, реализовать такое можно, используя Cross-Origin worker'a.

1.4 Сохраняется ли проксированный контент? Как настроить сохранение запросов в ServiceWoker?  
Данные в кеше сохраняются до тех пор, пока разработчик сам не решит их удалить. Для кэширования следует использовать встроенный API хранилища Service Worker — cache — глобальный для всех сервис-воркеров, который позволяет нам хранить результаты запросов.

1.5 Какие есть ограничения в работе Service Woker?

* работа только на localhost либо по протоколу HTTPS;
* поддерживаемость далеко не всеми браузерами (только самыми популярными и крупными);
* невозможность проксировать CORS-запросы.

### 2. IndexedDB

В рамках данного задания была изучена предложенная документация по IndexedDB.

Ответы на вопроосы:

2.1 Каким типом хранилища является IndexedDB?

Это хранилище ключей/значений (база данных noSQL).

2.2 Какая последовательность работы с хранилищем?

Работа с базой данных начинается с запроса на открытие. Затем необходимо создать хранилище объектов в базе данных, над которой будут выполняться операции. Далее запустить транзакцию и задать некоторый запрос, например, добавление или извлечение данных. Дождаться завершения операции, "слушая" событие DOM, на которое должен быть установлен обработчик, и сделать что-то с результатами.

2.3 Какой жизненный цикл хранилища?

Открытие БД -> Создание хранилища -> Некоторая работа -> Удаление хранилища -> Удаление БД

2.4 Будет ли доступна информация из хранилища при закрытии веб-приложения? Если нет – какой тип хранилища надо использовать?

Да, IndexedDB предоставляет возможность постоянно хранить данные внутри пользовательского браузера. Это позволяет создавать web-приложения с богатыми возможностями в независимости от доступа к сети. Такие приложения могут работать как online, так и offline.

2.5 Будет ли доступно хранилище двум одновременно открытым экземплярам веб-приложения? Почему?

IndexedDB - транзакционная БД. Все, что происходит внутри - происходит в контексте транзакции. Кроме того, концепция IndexedDB гарантирует атомарность транзакций. К тому же данное хранилище - асинхронное. Всё это позволяет поддерживать доступность хранилища двум и более одновременно открытым экземплярам приложения.

### 3. LocalStorage, SessionStorage

В рамках данного задания была изучена предложенная документация по LocalStorage и SessionStorage.

Также, было выполнено задание: создать поле textarea, значение которого будет автоматически сохраняться при каждом его изменении. Когда пользователь закроет страницу и потом откроет её заново он должен увидеть последнее введённое значение. Исходный код выполненного задания размещен в ветке [ToolsPWApplication](https://github.com/Xander11rgn/ISN_2020_2/tree/ToolsPWApplication/localStorage) данного репозитория.

Ответы на вопроосы:

3.1 Каким типом хранилища является LocalStorage и SessionStorage?  
Это тоже хранилище ключей/значений.

3.2 Какая последовательность работы с хранилищем?  
Сперва необходимо с помощью метода setItem(key, value) назначить пару/ключ значение. Получить записанные данные по ключу можно с помощью метода getItem(key). Если нужно удалить данные с ключом key - используется метод removeItem(key). Для полной очистки хранилища существует метод clear().

3.3 Какой жизненный цикл хранилища?

Данные, которые записаны в sessionStorage, сохраняются после обновления страницы, а данные, записанные в localStorage - даже после перезапуска браузера и ОС.

3.4 Будет ли доступна информация из хранилища при закрытии веб-приложения? Если нет – какой тип хранилища надо использовать?

Да, но только данные, записанные в localStorage, поскольку они сохраняются даже после перезапуска браузера и даже ОС. Данные, записанные в sessionStorage, не сохранятся после закрытия приложения. Более того, хранилище sessionStorage даже для разных вкладок одного и того же приложения разное.

3.5 Будет ли доступно хранилище двум одновременно открытым экземплярам веб-приложения? Почему?

Хранилище localStorage привязано к источнику (домен/протокол/порт). Это значит, что разные протоколы или поддомены определяют разные объекты хранилища, и они не могут получить доступ к данным друг друга. Поэтому, два одновременно открытых экземпляра приложения будут иметь доступ к хранилищу, ведь они будут на одних и тех же доменах/протоколах/портах.

### 4. Push Notification

В рамках задания была изучена предложенная документация по созданию push-уведомлений. Также, было выполнено задание: дополнить разработанное ранее vue-приложение в практическом задании push-уведомлениями согласно следующих требований к ним:

* случайное время уведомления;
* случайный текст на уведомлении;
* уведомления являются периодичными;
* уведомления хранятся в локальном хранилище.

Исходный код выполненного задания размещен в ветке [ToolsPWApplication](https://github.com/Xander11rgn/ISN_2020_2/tree/ToolsPWApplication/push) данного репозитория.

Демонстрация push-уведомления (рисунок 1).

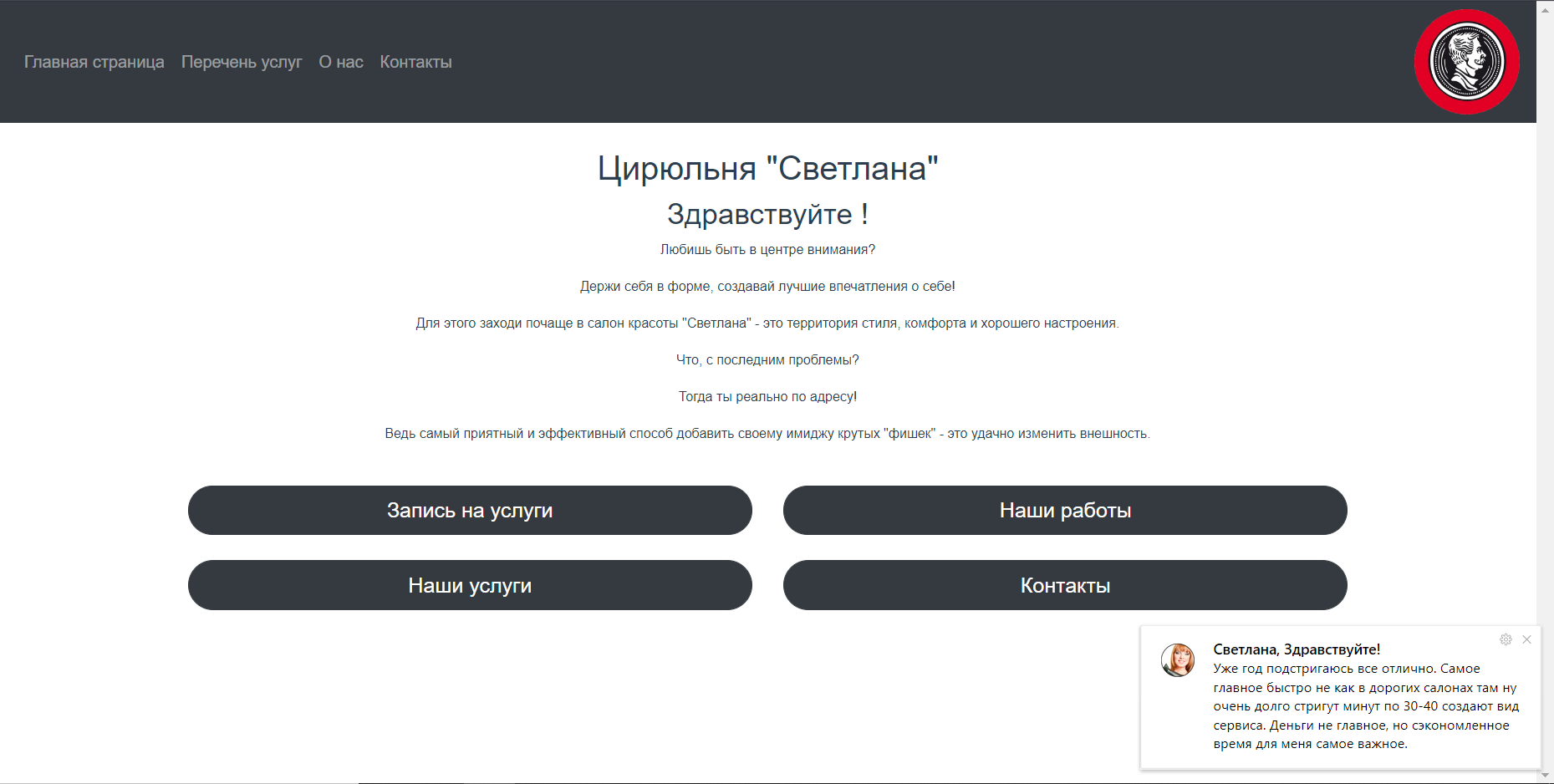


Рисунок 1 – Демонстрация push-уведомления

Демонстрация хранения уведомлений в localStorage (рисунок 2).

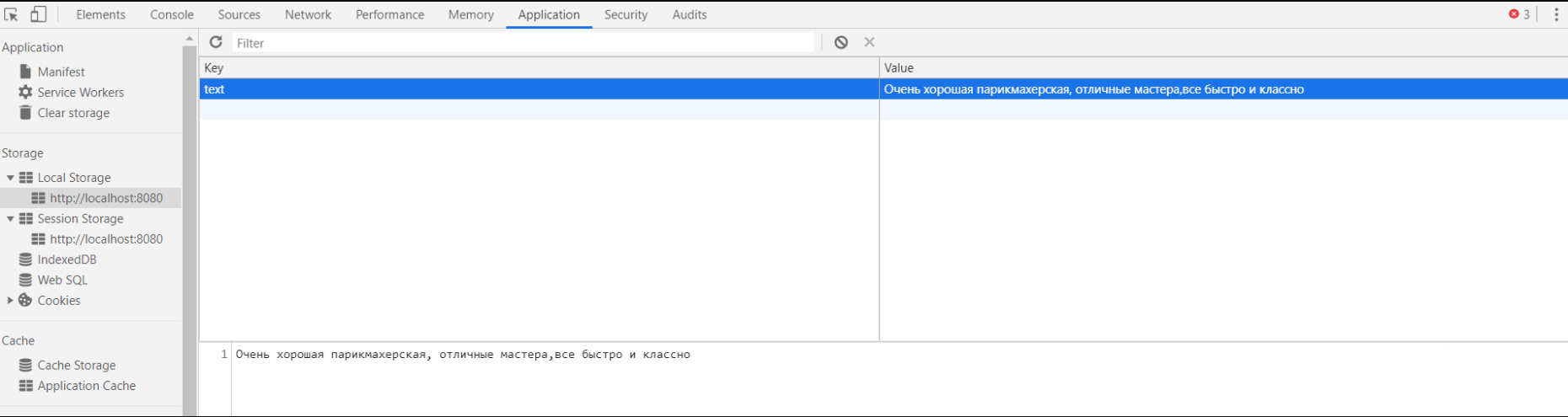


Рисунок 2 – Демонстрация хранения уведомлений в localStorage

### 5. Выводы:

В ходе данной лабораторной работы были получены знания и навыки работы с такими компонентами PWA, как Service Worker, IndexedDB, localStorage и sessionStorage, а также рассмотрен функционал push-уведомлений. Все эти компоненты в совокупности могут значительно улучшить производительность и юзабилити веб-приложения, обеспечить работу, не зависящую от наличия соединения с интернетом, ускорить процесс загрузки страниц и т.п. В качестве практических заданий были реализованы два проекта: работа с localStorage и работа с push-уведомлениями. При работе с уведомлениями необходимо было хранить уведомления в локальном хранилище. В качестве него был выбран localStorage, как наиболее подходящий вариант, поскольку sessionStorage живет лишь в пределах одной вкладки.