МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики»

**Факультет программной инженерии и компьютерной техники**

**Направление (специальность) —  09.04.02 Информационные системы и технологии**

**Специализация —  Веб-технологии**

**Дисциплина — Проектирование и анализ языков веб-решений**

**Курсовой проект (работа)**

**ТЕМА: Анализ кэширования данных в различных локациях браузера.**

ВЫПОЛНИЛ

Студент группы P41622 \_\_\_\_\_\_\_\_\_\_\_\_ Гарифуллина Г.Р.

№  группы подпись, дата ФИО

ПРОВЕРИЛ   к.педагог.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_ Государев И.Б.

ученая степень, должность подпись, дата ФИО

**САНКТ-ПЕТЕРБУРГ  
2019 г.**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_1fob9te) **3**

[ГЛАВА 1. КЭШИРОВАНИЕ В БРАУЗЕРЕ](#_4d34og8) **5**

[1.1 Принцип работы кэша в браузере](#_v89e3i6x679p) 5

[1.2 Особенности кэша в браузере](#_433er9lz5ckr) 9

[ГЛАВА 2. ОСНОВНЫЕ ЛОКАЦИИ КЭШИРОВАНИЯ В БРАУЗЕРЕ](#_3gv29coh3mr2) **11**

[2.1 База данных IndexedDB](#_71hu4lhofbeu) 11

[2.2 Cache API](#_8g2yl8grzpdq) 12

[2.3 Web Storage](#_e4xognk7e2e7) 12

[2.4 FileSystem API](#_s19w0p1qwmp5) 13

[ГЛАВА 3. ОСНОВНЫЕ КРИТЕРИИ СРАВНЕНИЯ ТЕХНОЛОГИЙ КЭШИРОВАНИЯ В БРАУЗЕРЕ](#_yq2m9zrdgzn1) **14**

[ЗАКЛЮЧЕНИЕ](#_knm5qefxqaae) **17**

[ПРИЛОЖЕНИЕ 1](#_vem1swuj141h) **18**

[ПРИЛОЖЕНИЕ 2](#_fexntjjde5dq) **19**

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ](#_3j2qqm3) **20**

# **ВВЕДЕНИЕ**

**Актуальность темы**.

Постепенно увеличивается число веб-приложений, содержащих в себе сложную бизнес-логику. При этом время отклика очень важно для положительного впечатления пользователей[1].Чтобы уменьшить время ожидания посетителей сайтов, появляется необходимость в оптимальной обработке запросов к веб-приложению.

Одна из основных техник оптимизации запросов, которая используется и в серверной, и в браузерной части веб-приложения, – это кэширование.

Кэширование – это процесс сохранения результата долгих вычислений или операций для дальнейшего использования. Кэширование позволяет ускорить получение данных, вычисление сложных операций и уменьшить нагрузку на источник.

Одними из преимуществ кэширования для веб-приложений являются:

1. улучшенная производительность сайта,
2. более быстрое время загрузки,
3. сокращение времени отклика веб-приложения,
4. уменьшение сетевого трафика.

Различные локации отвечают за кэширование различных ресурсов и служат для различных целей. Например, кэширование на сервере используется для улучшения производительности и масштабируемости сайта, в то время как кэширование на стороне клиента – для ускорения времени загрузки страницы и уменьшения сетевого трафика[2].

Как правило, чем ближе кэш находится к месту использования, тем большую выгоду он дает в случае попадания[3]. Следовательно, кэширование в браузере дает наибольший выигрыш для пользователя и способствует хорошему пользовательскому опыту взаимодействия с сайтом.

**Степень теоретической разработанности темы.**

Технологии кэширования браузера на данный момент бурно развиваются, поэтому есть множество ресурсов, описывающих данные технологии.

К сожалению, многие работы направлены на объяснение новых технологий и рассматривают их с точки зрения использования и разработки[4, 5, 6]. При этом источников литературы сравнивающих различные методы кэширования клиентской части веб-приложения не так много[7, 8]. В свою очередь каждая технология имеет свои ограничения, которые необходимо знать в процессе разработки.

В связи с вышеизложенным, разработчику при выборе метода кэширования необходимо ознакомиться со всеми существующими технологиями, что является энергозатратным процессом.

Данное исследование направлено на выявление основных ограничений при разработке с использованием кэширования в браузере и определение основных метрик при сравнении основных подходов.

**Цель исследования:** провести анализ основных локаций, используемых при кэшировании в браузере, с выделением отличительных признаков и определением основных критериев, по которым можно сравнивать существующие подходы.

**Задачи:**

1. Анализ литературы и веб-ресурсов по теме кэширования в браузере.
2. Выделение особенностей и ограничений кэширования, используемого на клиентской части веб-приложения.
3. Исследование основных подходов и локаций, используемых для кэширования в браузере.
4. Выделение основных критериев для сравнения.

# **ГЛАВА 1. КЭШИРОВАНИЕ В БРАУЗЕРЕ**

## 1.1 Принцип работы кэша в браузере

Кэширование в браузере определяет Стандарт Хранения (Storage Standard )[9], описывающий общую систему хранения данных, которую могут использовать веб-сайты. В данный стандарт входят:

1. база данных IndexedDB,
2. данные Cache API,
3. регистрация Service Worker,
4. память Web Storage API ,
5. состояние истории просмотра страниц пользователя,
6. кэши приложения (Application cache),
7. уведомления,
8. другие виды данных специфичные для сайта.

Данный стандарт делит общее пространство памяти браузера на неделимые единицы (unit): каждому ***источнику(origin)***[[1]](#footnote-0) выделяется фиксированный блок памяти, который только он может использовать. По сути, каждый веб-сайт или веб-приложение имеет свой собственный блок хранения, в который помещаются его данные. На рис.1 показан пример кэша с тремя источниками данных.

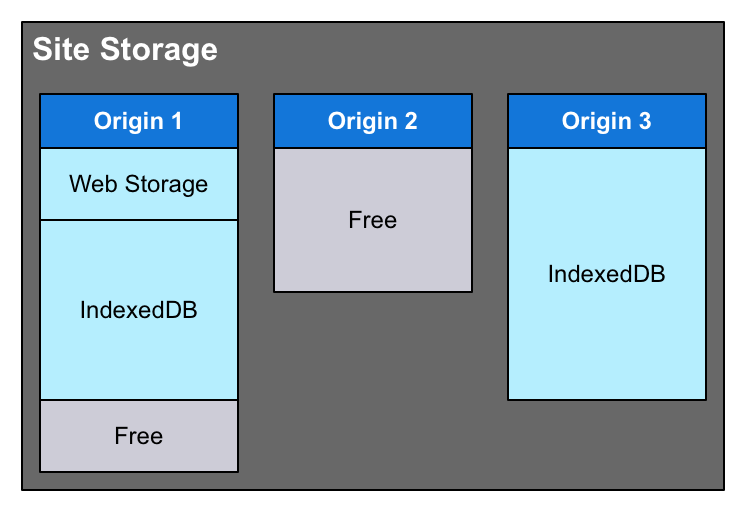


Рисунок 1. Кэш, содержащий три блока памяти.

Как видно на рис.1. каждый блок памяти имеет фиксированный размер и в нем могут храниться различные типы данных. Если блок содержит свободное пространство, то оно распределяется между всеми исходными хранилищами: IndexedDB, Cache API, localStorage, Web Storage, Файловая система и т.д.

В браузере есть два основных вида памяти:

1. Постоянная – данные с особой значимостью, которые предназначены для длительного хранения. Браузер не может их удалить без разрешения пользователя.
2. Временная – данные, которые не имеют особой значимости. Браузер будет пытаться хранить данные, содержащиеся в блоке, до тех пор пока не закончится память и не возникает необходимости очистить блок, чтобы уменьшить нагрузку на хранилище.

Все данные по умолчанию считаются временными.

Чтобы гарантировать корректную работу, браузер использует диспетчер квот, который отслеживает используемое дисковое пространство и следит за тем, чтобы оно не вышло за допустимые границы по памяти.

Каждый пользовательский агент имеет свои квоты (максимальные пределы хранения) по памяти, заданные спецификацией. В общем их можно поделить на три вида[10]:

* глобальный лимит – максимальный размер памяти, который может использовать пользовательский агент. Многие браузеры имеют динамический размер, зависящий от свободного пространства на компьютере. Глобальный лимит рассчитывается как 50% свободного дискового пространства.
* лимит на группу – размер памяти, выделенный на группу источников. Одна группа объединяет источники, принадлежащие домену eTLD + 1[[2]](#footnote-1). Обычно лимит на группу определяется как 20% от глобального лимита, но он имеет минимум 10 МБ и максимум 2 ГБ.
* лимит на источник. Данный предел варьируется в зависимости от устройства и оставшегося свободного пространства. В табл. 1 показана зависимость максимального лимита на источник от пользовательского агента.

Таблица 1. Зависимость максимальных квот на источник от браузера[11].

|  |  |
| --- | --- |
| ***Браузер*** | ***Лимит*** |
| Chrome | <6% от свободной памяти |
| Firefox | <10% от свободной памяти |
| Safari | <50MB |
| IE10 | <250MB |
| Edge | зависит от тома диска |

Если глобальный лимит памяти или лимит на группу превышен, то начинается процесс ***вытеснения источника***: в соответствии с политикой вытеснения (см. табл. 2): по очереди удаляются все данные источника до тех пор, пока не освободиться достаточное место в памяти. В процессе удаления не применяется частичное удаление данных источника, так как это может вызвать проблемы несогласованности данных.

Таблица 2. Политики вытеснения источников в зависимости от браузера[11].

|  |  |
| --- | --- |
| ***Браузер*** | ***Политика вытеснения*** |
| Chrome | LRU[[3]](#footnote-2), если у Chrome заканчилось память |
| Firefox | LRU , если диск компьютера переполнен |
| Safari | Нет вытеснения |
| Edge | Нет вытеснения |

## 1.2 Особенности кэша в браузере

В результате анализа кэширования в браузере можно выделить признаки, свойственные только данной области.

1. Нельзя гарантировать, что данные кэша не были модифицированы или удалены пользователем.

В отличие от кэширования на серверной части, кэш браузера не имеет ограничений на неавторизованный доступ: кэш может быть доступен расширениям браузера и через инструменты разработчика. В результате, данные могут быть модифицированы или удалены пользователем. Следовательно, необходимо разрабатывать веб-приложения, которые могут справиться с такими ситуациями без ошибок.

1. Не все методы, инструменты поддерживаются всеми пользовательскими агентами.

Данная проблема обусловлена тем, что различные браузеры имеют различные спецификации и не всегда придерживаются общих стандартов. В результате надо учитывать поддержку используемых технологий.

1. Операция записи в кэш может провалиться.

Ошибки записи в кэш могут происходить из-за множества причин и не во всех случаях они подвластны разработчику. Например, в режиме приватного просмотра операция записи в IndexedDB недоступна, либо возможна ситуация, что в устройстве почти не осталось места на диске, в итоге браузер запрещает хранить какие-либо данные.

1. Размер кэша определяется браузером и состоянием системы.

Разработчик не может увеличить размер кэша.

1. Веб-приложение не может знать сколько конкретно осталось свободной памяти в доступе.

Не так давно был разработан Storage API[9][[4]](#footnote-3), который позволяет веб-приложениям узнавать объем свободной памяти до записи. Следует отметить, что данный метод выдает примерную оценку свободной памяти.

1. Все данные по умолчанию считаются временными.

Следовательно эти данные могут быть удалены. Данная ситуация опасна тем, что пользователь может ввести важные данные в момент, когда веб-приложение находится в оффлайн режиме, в результате нет гарантии, что данные не удалятся. Чтобы решить данную проблему, был разработан Persistent Storage, входящий в Storage API, который позволяет приложениям специфицировать данные как постоянные.

1. Пользователь может отключить кэширование в браузере.

В результате веб-приложение будет загружаться каждый раз заново, исключая возможности оптимизации.

# **ГЛАВА 2. ОСНОВНЫЕ ЛОКАЦИИ КЭШИРОВАНИЯ В БРАУЗЕРЕ**

На данный момент инструменты кэширования в браузере бурно развиваются, в связи с этим некоторые подходы устаревают и заменяются новыми. В то же время есть технологии, которые на данный момент находятся в экспериментальном состоянии и имеют хороший потенциал использоваться в будущем.

В ходе данного обзора интерес представляют технологии, которые используются повсеместно, и новые разработки, которые находятся на стадии развития. В результате были выделены следующие хранилища данных в браузере:

1. IndexedDB,
2. Cache API,
3. Web Storage,
4. FileSystem API.

## 2.1 База данных IndexedDB

IndexedDB[4] - это низкоуровневый API для хранения на стороне клиента значительных объемов структурированных данных, в том числе файлов и больших двоичных объектов. Этот API использует индексы для обеспечения высокопроизводительного поиска данных.

IndexedDB API является объектно-ориентированной базой данных, основанной на транзакциях. IndexedDB API является мощным, но может показаться слишком сложным для простых случаев. Данная технология входит в Web Workers.

## 2.2 Cache API

Cache API[14] является экспериментальной технологией и входит в пакет Service Worker. Интерфейс Cache обеспечивает механизм кэширования запросов и ответов.

Данная технология в первую очередь обеспечивает полностью управляемый процесс кэширования ресурсов, поэтому многие низкоуровневые операции доступны разработчику. Благодаря этому функционалу, Cache API позволяет реализовать сложные алгоритмы кэширования, например, stale-while-revalidate[15].

В свою очередь высокая управляемость процесса кэширования требует большой ответственности, например система не будет сама обновлять кэш, за это отвечает разработчик.

Следует отметить, что Cache API не поддерживает заголовки кэширования HTTP.

## 2.3 Web Storage

В отличие от IndexedDB, Web Storage[16] используется для хранения меньшего объема данных и служит для предоставления отдельного пространства памяти для каждого источника. Есть два основных механизма работы:

1. SessionStorage хранит данные на время сеанса страницы (до закрытия браузера).
2. LocalStorage хранит данные даже после закрытия браузера.

Данный метод поддерживается всеми браузерами. Однако начиная с Firefox 45, при аварийном завершении работы браузера объем данных, сохраняемых на источник, ограничен 10 МБ. Это было сделано, чтобы избежать проблем с памятью, вызванных чрезмерным использованием веб-хранилища.

## 2.4 FileSystem API

Данная технология на данный момент не входит в стандарт.

FileSystem API[17, 18] имитирует локальную файловую систему, в которой веб-приложения могут совершать различные операции с файлами. FileSystem API – это решение файловой системы для Интернета. Разработчики могут хранить большие объекты в изолированной части файловой системы пользователя и напрямую ссылаться на них через URL. Хотя Chrome и Opera являются единственными браузерами, которые в настоящее время реализуют этот функционал, ее стандартизация продолжается.

# 

# **ГЛАВА 3. ОСНОВНЫЕ КРИТЕРИИ СРАВНЕНИЯ ТЕХНОЛОГИЙ КЭШИРОВАНИЯ В БРАУЗЕРЕ**

В ходе проведения анализа существующих веб-хранилищ и их спецификаций работы были выделены основные критерии, которые определяют эти технологии и могут использоваться как метрики при выборе той или иной технологии.

В основном метрики можно поделить на три большие группы, которые оценивают данные технологии с различных сторон:

1. влияния на систему, выполнения основных задач кэширования;
2. пользователя;
3. разработчика.

В первую группу можно выделить метрики, которые определяют цели кэширования и возможности данной технологии. Данные критерии рассматривают хранение каких данных поддерживает система, какие ограничения по памяти система определяет, скорость работы транзакций и используемые политики при кэшировании.

Вторая группа направлена для выявления особенностей поведения данных технологий с точки зрения пользователя, а именно поведение в различных браузерах и функции, обеспечивающие безопасность пользователя.

Третья группа рассматривает аспекты важные в процессе разработки и включает используемые технологии и удобство разработки.

Полный список представлен ниже:

1. Хранение данных:
   1. Поддержка различных типов данных.
   2. Поддержка хранения файлов.
   3. Структурированность данных.
2. Память:
   1. Минимальный размер памяти, необходимый для корректной работы хранилища.
   2. Минимальный размер единицы записи.
   3. Максимальный размер объекта, записываемого в кэш.
   4. Возможность использования битовых операций.
   5. Поддержка общей разделяемой памяти для двух связанных приложений.
3. Запросы и скорость работы:
   1. Скорость записи.
   2. Скорость чтения.
   3. Эффективность записи большого числа данных по сравнению с кусочным.
   4. Блокирование DOM.
   5. Синхронность или асинхронность операций.
4. Политика кэширования:
   1. Политика замещения.
   2. Время хранения данных.
   3. Гарантия консистентности данных.
5. Критерии зависящие от пользовательских агентов:
   1. Поддержка в браузерах.
   2. Максимальная выделяемая память в различных браузерах.
6. Безопасность:
   1. Возможность шифрования данных.
   2. Поведение в режиме инкогнито.
7. Удобство использования:
   1. Настраиваемость, низко уровневость функционала.
   2. Сложность изучения.
   3. Сложность реализации.
   4. Возможность отладки при возникающих исключениях, например при неудачной операции записи в кэш. Степень подробности сообщений системы.
8. Технологии:
   1. Технологии, на которых основано хранилище.
   2. Поддержка Service Worker.

В будущем планируется сделать исследование классифицирующее технологии кэширования по этим пунктам, основанные на реальных экспериментах.

# 

# **ЗАКЛЮЧЕНИЕ**

В ходе данной работы был проведен анализ процесса кэширования клиентской части веб-приложения и на основе этого были выявлены основные особенности данного вида кэширования:

* отсутствие гарантии, что данные кэша не были модифицированы пользователем;
* различные технологии поддерживаются различными пользовательскими агентами;
* операция записи в кэш может быть неудачной;
* размер кэша нельзя настроить, так как зависит от состояния внешней системы;
* нет возможности узнать точную оценку свободной памяти;
* все данные по умолчанию считаются временными.

Были рассмотрены различные хранилища: IndexedDB, Cache API, Web Storage и FileSystem API, и определены их отличительные особенности.

Впоследствии, были определены основные критерии для сравнения технологий кэширования в браузере, которые в общем можно поделить на три группы, оценивающие эти методы в точки зрения различных ролей: с точки зрения разработчика, пользователя и веб-приложения.

В результате данные критерии будут использоваться при дальнейшем исследовании для классификации и сравнении существующих технологий кэширования в браузере, которые позволят составить методологию выбора хранилищ данных в браузере.

# 

# **ПРИЛОЖЕНИЕ 1**

***Источник (origin) или первичные данные*** определяется протоколом, хостом (доменом) и портом. Два источника считаются одинаковыми, если у них совокупность этих данных одинакова.  
Например, <https://google.com>, <https://google.com/edu> имеют одинаковый источник.

***Пользовательский агент*** – это компьютерная программа, представляющая человека, например, в контексте веба это браузер.

***Группа источников*** объединяет источники, принадлежащие домену eTLD + 1. Например,

|  |  |  |
| --- | --- | --- |
| mozilla.org | group1 | origin1 |
| www.mozilla.org | group1 | origin2 |
| joe.blogs.mozilla.org | group1 | origin3 |
| firefox.com | firefox.com | origin4 |

[mozilla.org](http://mozilla.org), [www.mozilla.org](http://www.mozilla.org), [joe.blogs.mozilla.org](http://joe.blogs.mozilla.org) будут принадлежать одной группе, в отличие от [firefox.com](http://firefox.com).

# **ПРИЛОЖЕНИЕ 2**

Как видно по рис. 2 Internet Explorer, Edge Mobile и Safari на IOS не поддерживают Storage API.

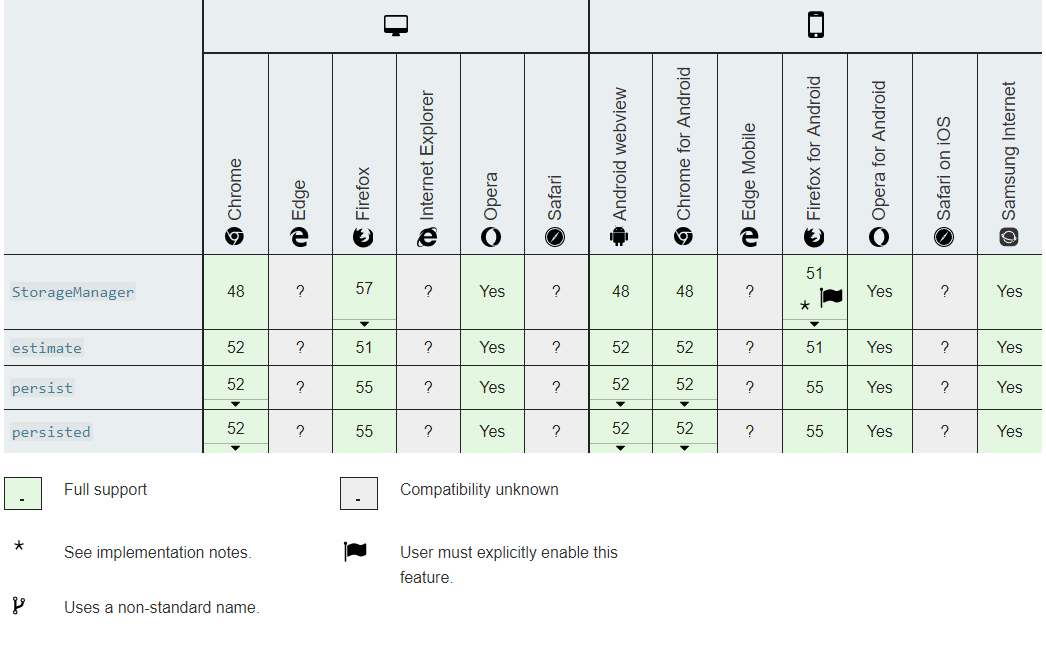


Рисунок 2. Поддержка Storage API.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Смирнов Д.Ю. "Комплексный подход к оптимизации производительности при разработке современных веб-приложений"
2. Mohan, C. "Caching Technologies for Web Applications." VLDB. Vol. 1. 2001.
3. Mertz J. Nunes I. Understanding application-level caching in web applications: a comprehensive introduction and survey of state-of-the-art approaches. // ACM Computing Surveys (CSUR). – 2018. – V. 50. – N. 6. – № 98. – P. 1-34.
4. IndexedDB API [Электронный ресурс] / Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API> (дата обращения: 20.05.2019)
5. Using the Cache API [Электронный ресурс] / Режим доступа: <https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/cache-api> (дата обращения: 20.05.2019)
6. Best Practices for Using IndexedDB [Электронный ресурс] / Режим доступа: <https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/indexeddb-best-practices> (дата обращения: 20.05.2019)
7. Working with quota on mobile browsers [Электронный ресурс] / Режим доступа: <https://www.html5rocks.com/en/tutorials/offline/quota-research/> (дата обращения: 20.05.2019)
8. How JavaScript works: Storage engines + how to choose the proper storage API [Электронный ресурс] / Режим доступа: <https://blog.sessionstack.com/how-javascript-works-storage-engines-how-to-choose-the-proper-storage-api-da50879ef576> (дата обращения: 20.05.2019)
9. Storage API [Электронный ресурс] / Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/Storage_API> (дата обращения: 20.05.2019)
10. Browser storage limits and eviction criteria [Электронный ресурс] / Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Browser_storage_limits_and_eviction_criteria> (дата обращения: 20.05.2019)
11. Offline Storage for Progressive Web Apps[Электронный ресурс] / Режим доступа: <https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/offline-for-pwa> (дата обращения: 20.05.2019)
12. Storage, Living Standard — Last Updated 18 February 2019 [Электронный ресурс] / Режим доступа: <https://storage.spec.whatwg.org/> (дата обращения: 20.05.2019)
13. Persistent Storage [Электронный ресурс] / Режим доступа: <https://developers.google.com/web/updates/2016/06/persistent-storage> (дата обращения: 20.05.2019)
14. Cache [Электронный ресурс] / Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/Cache> (дата обращения: 20.05.2019)
15. Two HTTP Caching Extensions [Электронный ресурс] / Режим доступа: <https://www.mnot.net/blog/2007/12/12/stale> (дата обращения: 20.05.2019)
16. Web Storage API [Электронный ресурс] / Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API> (дата обращения: 20.05.2019)
17. File and Directory Entries API [Электронный ресурс] / Режим доступа: <https://wicg.github.io/entries-api/> (дата обращения: 20.05.2019)
18. File and Directory Entries API [Электронный ресурс] / Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/API/File_and_Directory_Entries_API> (дата обращения: 20.05.2019)

1. См. определение в Приложении 1. [↑](#footnote-ref-0)
2. Более подробное описание можно найти в Приложении 1. [↑](#footnote-ref-1)
3. ***LRU (Least Recently Used, Замещение наименее позднего элемента)*** – это алгоритм замещения, который хранит информацию о последнем доступе к элементу и в случае переполнения кэша удаляет элемент с наиболее ранним последним доступом. [↑](#footnote-ref-2)
4. Storage API не поддерживается всеми браузерами см. Приложение 2 [↑](#footnote-ref-3)