Описание:

DriveAPI – продукт для реализации возможности использования Goole Drive в качестве основного файлого хранилища для веб-ресурса. В своей работе DriveAPI использует PHP-библиотеку, раеализующую Google API для получения токена для исполнения запросов к хранилищу. Способы встраивания библиотеки в ваш проект и требования к ней есть официальной документации от Google. Но я на всякий случай все же выложил соответствующие php-файлы на случай если у кого-то возникнут затруднения.

И так, для корректной работы модуля вам понадобится сгенерированный в консоли разработчика файл стандарта PKCS#12 (можно сгенерировать при создании сервис-аккаунта), имя файла будет похоже на your-account-c5e5588daee0.p12. Файл этот предназначен для организации аутентифицированной связи между вашим сервером и Google Drive. Собственно, вся механика такова: ваш сервер по запросу от клиента при помощи PHP Google API с ключем P12 на борту делает запрос к Google на манер: «Разреши мне записать данные в твое хранилище, вот мое удостоверение (ключ P12)» и если Google признал вас как собственника вашего виртулаьного диска в хранилище Google Drive, то он отвечает: “Ок, бро, это можно. Вот твой пароль. Присылай ео со всеми своими файлами и их будет без проблем пропускать». На самом деле все немного сложнее, но если опустить техподробности, то все будет выглядеть примерно так. Далее сервер отдает полученный токен клиенту – JS-части библиотеки и эта часть уже занимается непосредственно отправкой файла.

Одна из основных фишек модуля состоит в том что при подключении его к сайту можно загружать файлы в хранилище Google без авторизации в Google-аккаунте. Сайт берет на себя ответственность за аутентификацию (если это требуется) пользователей, загружающих файлы и выступает посредеником между пользователем Google Drive'ом, позволяя загружать файлы от своего имени, файл при этом на сервер хостящий сайт никаким образом не попадает – загрузка производится напрямую на сервера Google.

Одной из проблем связанных с повседневной загрузкой файлов в Google Drive через веб является проблема загрузки больших файлов в хранилище. Дело в том, что Google принимает в виде файлов (или частей файлов) только данные, закодированные в Base64. Стандартные примеры кода, для загрузки файлов в Google Drive через JavaScript включают себя обязательную конвертацию загружаетмого файла, для этого нужно сначала считать с диска в память браузера весь файл, а потом его еще и переконвертировать в Base64. Штука в том, что на больших файлах для (IE это 40+ Мб) браузеры валятся от переполения собственной памяти. Однако, как оказалось никто не мешает считывать файл в память кусками, затем эти куски конвертировать и отправлять последоватлеьно на сервер. Таким образом, в один момент времени в памяти браузера перелопачивается только один кусок файла. В сочетании с простым механизмом докачки из REST API Google Drive, получает достаточно удобный механизм закачки больших файлов и как бонус индикацию процесса загрузки, если нужно.

Алгоритм следующий: читаем из нужного файла CHUNK\_SIZE байт данных, конвертируем их и отправляем Google в синхронном режиме. В этом есть небольшой недостаток: мы не можем отпавлять куски файла асинхронно в несколько потоков, Google жалуется, что куски приходят в неправильном порядке. Далее повторяем операции до вплоть до конца загружаемого файла.

Файлы:

Серверная часть:

**config.php** – файл конфигурации серверной части модуля.

*Директивы:*

**AUTOLOAD\_PATH** – путь к файлу autoload.php в библиотеке Google API, который отвечает за подгрузку ее необходимых компонентов;

**SERVICE\_ACCOUNT\_NAME** – имя аккаунта пользователя Google API (можно посмотреть в консоли разработчика).

**KEY\_FILE\_LOCATION** – путь к файлу ключа PKCS#12

**getToken.php** – файл с кодом, запрашивающим токен у Google. В простейшем случае, клиенту достаточно просто запросить это файл на сервере и в теле ответа он получит токен в текстовом виде.

Клиентская часть:

**config.js** – конфигурация клиентской части модуля, представляет собой объект.

*Директивы:*

**FOLDERS** – разделы для файлов в хранилище. Предполаегается что в хранилище могут быть созданы директории для хранения разничных видов файлов. Впоследствие, можно выбрать конкретную директорию для загрузки конкретного файла, идентифицировав ее по идентификатору заданному в этом свойстве. Так же файлы можно писать и в корень диска, если не задавать родительскую папку.

**DRIVE\_FILE\_INFO** – строка, задающая то какие свойства файла (файлов), нам нужно получить как ответ на тот или иной запрос к серверу Google.

**BOUNDARY** – используется при multipart-загрузке файлов. Представляет собой строку-разделитель для партифионирования частей тела запроса, при отправкев нем дополнительных параметров и заголовков.

**CHUNK\_SIZE** – уже обозначенный выше параметр, указывает количестве байт файла, которые нужно считывать и отправлять на сервер итеративно, при resumable-загрузке файлов. Если райзмер файла в байтах меньше этого параметра, то используется mutlipart-загрузка.

**SERVER\_URL** – путь к php-файлу, возвращающему Google-токен.

**drive\_api.js**

*Методы:*

**DriveAPI** – конструктор класса.

**function** *DriveAPI* (label, folders, cfg)

*Параметры:*

**label** – тот самый идентификатор директории, в которую предполагается писать. Один объект DriveAPI по умолчанию пишет в директорию, указанную при создании объекта.

**folders** = собственно, тот самый список директорий для записи, представляет собой ассоциативный массив, а в терминах JS объект, где ключами выступают идентификаторы разделов внутри модуля, а значениями – идентификаторы оных в Google Drive вида **0B-e8MNz22zZvfk9wS0o1c2JMZHIyV2xDeHFwLWQ2anJHQTB5YzNlVDg3YjJKX1ZHTVlEMVk**. Если задан, то перепишет элементы из конфига, которые будут иметь такие же ключи, все новое допишет в конец.

**cfg** – объект конфигурации, перепишет элементы считанного из config.js при совпадении ключей.

**checkConfig** – проверяет наличие и верный формат всех необходимых для корректной работы модуля директив конфигурации.

DriveAPI.**prototype**.checkConfig = **function** ()

*Параметры:* нет

**initFolders** – устанавливает массив директорий для записи.

DriveAPI.**prototype**.initFolders = **function** (folders)

*Параметры:*

**folders** – массив идентификаторов папок.

**setActiveFolder** – устанавливает активную директорию, в которую будут писаться файлы по умолчанию. Строго говоря для активная директория актуальна и для метода getFiles, если при его вызове не задавать второй параметр, но такое использование getFiles лично мне не видится правильным.

DriveAPI.**prototype**.setActiveFolder = **function**(label)

*Параметры:*

**label** – идетификатор директории.

**addFolder** - добавляет дополнительную директория для записи и чтения

DriveAPI.**prototype**.addFolder = **function** (label, folder\_id)

*Параметры:*

**label** - идентификатор директории в объекте

**folder\_id** - идетификатор директории в хранилище

**deleteFolder** - удаляет элемент списка доступных директорий.

DriveAPI.**prototype**.deleteFolder = **function** (label)

*Параметры:*

**label** - идентификатор удаляемой директории

**getFile** - получить информацию о загруженном файле

DriveAPI.**prototype**.getFile = **function** (callback, file\_id)

*Параметры:*

**callback** - функция, выполняющаяся при успешном получении информации от хранилища;  
**file\_id** - идетификатор файла в хранилище.

**getFiles** - Получить информацию обо всех файлах из директории.

DriveAPI.**prototype**.getFiles = **function** (callback, folder\_id)

*Параметры:*

**callback** - функция, выполняющаяся при успешном получении информации о файлах;  
**folder\_id** - идентификатор директории.

**uploadFile** - загрузка файла в хранилище multipart-способом.

DriveAPI.**prototype**.uploadFile = **function** (file, callback, folder\_id)

*Параметры:*

**file** - файл из input[type=file];  
**callback** - функция-обработчик успешной загрузки;  
**folder\_id** - идентификатор родительской директории для файла.

**IEBinary** – формирует байтовую последовательность из файла или части файла (актуально для IE)

DriveAPI.**prototype**.IEBinary = **function** (buffer)

*Параметры:*

**buffer** - содержимое файла

**getChunkRange** - формирует заголовок для задания интервала байт, части файла, которая будет отправляться на сервер.

DriveAPI.**prototype**.getChunkRange = **function** (total\_size, last\_size)

*Параметры:*

**total\_size** - общий размер файла в байтах;  
**last\_size** - начальное значение для интервала.

**uploadResumable** - загрузка файла на сервер с использование докачки.

DriveAPI.**prototype**.uploadResumable = **function** (file, callback, folder\_id)

*Параметры:*

**file** - файл из input[type=file];  
**callback** - функция-обработчик успешной загрузки файла;  
**folder\_id** - идентификатор родительской директории для файла.

**readChunk** - читает часть файла.

DriveAPI.**prototype**.readChunk = **function** (reader, file, shank, last\_size)

*Параметры:*

**reader** - объект FileReader();  
**file** – файл;**shank** - размер считываемого куска;  
**last\_size** - с какого места читать.

**updateFile** - меняет уже загруженный в хранилище файл на новый, загруженный multipart-способом.

DriveAPI.**prototype**.updateFile = **function** (file, file\_id, callback)

*Параметры:*

**file** - файл из input[type=file];  
**file\_id** - идентификатор меняемого файла;  
**callback** - функция-обработчик успешного изменения файла.

**updateResumable** - меняет уже загруженный в хранилище файл на новый, загруженный resumable-способом.

DriveAPI.**prototype**.updateResumable = **function** (file, file\_id, callback)

*Параметры:*

**file** - файл из input[type=file];  
**file\_id** - идентификатор меняемого файла;  
**callback** - функция-обработчик успешного изменения файла.

**createFolder** - создает директорию в хранилище и возвращает ее идентификатор.

DriveAPI.**prototype**.createFolder = **function** (callback, parent\_id)

*Параметры:*

**callback** - обработчик успешного создания директории;  
**parent\_id** - родительская директория для создаваемой.

**delete** - удаляет файл или директорию из хранилища

DriveAPI.**prototype**.delete = **function** (id, callback)

*Параметры:*

**id** - идентификатор обекта для удаления;  
**callback** - обработчик успешного удаления.

**getInternetExplorerVersion** - возвращает версию браузера, если это IE

DriveAPI.**prototype**.getInternetExplorerVersion = **function** ()

*Параметры:* нет.

**setToken** - запрашивает у сервера токен для доступа к хранилищу и сохраняет его в объекте.

DriveAPI.**prototype**.setToken = **function** ()

*Параметры:* нет

**deleteToken** - Удаляет токен для доступа к хранилищу

DriveAPI.**prototype**.deleteToken = **function** ()

*Параметры:* нет.