Download API of Android Extension

Abstract

This specification defines an API that provides information and a series of operations

about the download tasks of the hosting device.

1. Introduction

*This section is non-normative.*

The Download API specification defines a means for web developers or android app developers to programmatically operate the download tasks of the hosting device, including creating new downloads, controlling ongoing downloads, retrieving downloads and getting a list of download tasks. Without those operations we have designed for the download tasks, an application developer must design the web application with operations connected to the OS of the hosting device. That means the developers must have great knowledge of norms for development concerning a specific OS, such as Android or iOS. This requirement will obstruct the developers who must spend a lot of time designing an APP. Using the download API we will design below, developers are able to operate the download tasks more efficiently.

1. Conformance

As well as sections marked as *non-normative,* all authoring guidelines, diagrams, examples, and notes in this specification are *non-normative*. Everything else in this specification is *normative*.

he key words must, must not, required, should, should not, recommended, may, and optional in this specification are to be interpreted as described in [RFC2119].

This specification defines conformance criteria that apply to a single product: the ***user agent*** that calls the services that it contains.

1. Terminology

The following concepts, terms and interfaces are defined in [HTML5]:

* EventHandler
* event handler and event types

[***Promise***](http://people.mozilla.org/~jorendorff/es6-draft.html#sec-promise-objects)objects are defined in [[ECMASCRIPT](http://www.w3.org/TR/battery-status/#bib-ECMASCRIPT)].

1. The DownloadItem interface

This interface defines asingle download task with necessary attributes and operations.

|  |
| --- |
| WebIDL: |
|  |

The ***user agent*** have no permission to create a new *DownloadItem* object. A new *DownloadItem* object can be only created by a *method* of the *DownloadManager* object named *createDownload.*

**Enumeration type:**

* enum DownloadState: defines all the states which will occur in the process of download
* enum DownloadError: defines all the errors which will occur in the process of download. The meaning of every error will be described in the following.

**Attributes:**

* readonly int id [required] : the download item identifier.
* readonly String sourceUrl [required]: the URL of the download source
* readonly String targetFile[required]: the target file or the absolute path for putting the downloaded data
* readonly Date startTime[required]: the time when the download task starts.
* readonly DownloadState state[required]: the current state of the download task.
* readonly DownlaodError error [nullable]: the current error of the download task.
* readonly long totalBytes[required]: the total bytes of the download file.
* readonly String MIMEType[required]: the mime type of the download file

**Event Handlers:**

The following are the *event handlers* (and their corresponding *event handler event types*) that must be supported as attributes by the DownloadItem object:

|  |  |
| --- | --- |
| Event Handlers | event handler event types |
| onPause | pause |
| onResume | resume |
| onCancel | cancel |
| onProgress | progress |

**Methods:**

1. **start:** starts the download task for the first time, or restarts a download task that failed or has been canceled.



**Privilege level:** public

**Parameters:** none

**Exceptions:**

* Exception with error type *FileAlreadyExistError*, if the requested destination file already exists.
* Exception with error type *DeviceNotFoundError*, if bi exterbak storage was found
* Exception with error type *HttpDataError,* if an error receiving data occurred at the HTTP level.
* Exception with error type *InsufficientSpaceError,* if there was insufficient storage space
* Exception with error type *UnhandledHttpError,* if a HTTP code was received that download manager can’t handle
* Exception with error type *UnknownError,* if the method cannot be completed because of an unknown error.

1. **pause:** pauses the download when the download task is running.



**Privilege level:** public

**Parameters:** none

**Exceptions:**

* Exception with error type *HttpDataError,* if an error receiving data occurred at the HTTP level.
* Exception with error type *UnhandledHttpError,* if a HTTP code was received that download manager can’t handle
* Exception with error type *UnknownError,* if the method cannot be completed because of an unknown error.

1. **resume:** resumes the download task when it has been paused.



**Privilege level:** public

**Parameters:** none

**Exceptions:**

* Exception with error type *HttpDataError,* if an error receiving data occurred at the HTTP level.
* Exception with error type *UnhandledHttpError,* if a HTTP code was received that download manager can’t handle
* Exception with error type *UnknownError,* if the method cannot be completed because of an unknown error.
* Exception with error type *CanNotResumeError,* if some possibly transient error occurred but we can’t resume the download.

1. **cancel:** cancels the download task



**Privilege level:** public

**Parameters:** none

**Exceptions:**

* Exception with error type *HttpDataError,* if an error receiving data occurred at the HTTP level.
* Exception with error type *UnhandledHttpError,* if a HTTP code was received that download manager can’t handle
* Exception with error type *UnknownError,* if the method cannot be completed because of an unknown error.

1. The DownloadManager interface

This interface provides a single entry point to manage the download tasks, including starting new downloads, controlling ongoing downloads, removing downloads and getting a list of download tasks.

|  |
| --- |
| WebIDL: |
|  |

Since this interface have no attribute, when the ***user agent*** is to create a new *DownloadManager* object, it can instantiate a new *DownloadManager* object without set any attribute’s value.

**Methods:**

1. **createDownload:** creates a new DownloadItem object.

For each browsing context, there is a promise, which is initially set to null. It is a Promise object which holds a DownloadItem.



**Privilege level:** public

**Parameters:**

* sourceUrl: URL of the download source. It must be String.
* targetFile: the target file or the absolute path for putting the downloaded data. It must be String.

**Exceptions:**

* Exception with error type *InvalidValuesError*, if any input parameter does not contain a valid value.
* Exception with error type *TypeMismatchError*, if any input parameter is not compatible with the expected type for that parameter.
* Exception with error type *UnknownError,* if the method cannot be completed because of an unknown error.

1. **getAll:** gets an array of *DownloadItem* objects that are currently in the system



**Privilege level:** public

**Parameters:**  none

**Return value:**

DownloadItem[] : all DownloadItem objects.

**Exceptions:**

* Exception with error type *HttpDataError*, if an error receiving or processing data occurred at the HTTP level
* Exception with error type UnknownError, if the method cannot be completed because of an unknown error.

1. **getDownloadItemById:** gets a *DownloadItem* object of the same *id* as the parameter



**Privilege level:** public

**Parameters:**

* id: The ID of the download item to retrieve. It must be unsigned long.

**Return value:**

DownloadItem: a DownloadItem object with the specified ID.

**Exceptions:**

* Exception with error type *NotFoundError*, if this download task identifier cannot be found in the system of the hosting device.
* Exception with error type *TypeMismatchError*, if any input parameter is not compatible with the expected type for that parameter.
* Exception with error type *InvalidValuesError,* if any input parameter does not contain a valid value.
* Exception with error type *UnknownError,* if the method cannot be completed because of an unknown error.

1. **getDownloadItemsByState:** gets an array of *DownloadItem* objects of the same *state* as the parameter.



**Privilege level:** public

**Parameters:**

* state: The state of the download items to retrieve. It must be one of values within the enum type DownloadState .

Every value as follows:

“running” , “paused” , “canceled” , “failed”, “successful”.

**Return value:**

DownloadItem[]: all DownloadItem objects with the specified state.

**Exceptions:**

* Exception with error type *NotFoundError*, if this download task identifier cannot be found in the system of the hosting device.
* Exception with error type *TypeMismatchError*, if any input parameter is not compatible with the expected type for that parameter.
* Exception with error type *InvalidValuesError,* if any input parameter does not contain a valid value.
* Exception with error type *UnknownError,* if the method cannot be completed because of an unknown error.

1. **removeDownloads:** removes some *DownloadItem* objects whose identifier is within the parameter.



**Privilege level:** public

**Parameters:**

* idArray: a list of IDs of the download items to remove. It must be an Array of unsigned long type.

**Exceptions:**

* Exception with error type *NotFoundError*, if any download task identifier cannot be found in the system of the hosting device.
* Exception with error type *TypeMismatchError*, if any input parameter is not compatible with the expected type for that parameter.
* Exception with error type *InvalidValuesError,* if any input parameter does not contain a valid value.
* Exception with error type *UnknownError,* if the method cannot be completed because of an unknown error.

1. **removeAll:** removes all *DownloadItem* objects which are currently in the system.



**Privilege level:** public

**Parameters:**  none

**Exceptions:**

* Exception with error type *UnknownError,* if the method cannot be completed because of an unknown error.

1. Examples

This section is *non-normative*.

|  |
| --- |
| A simple download example: |
|  |

1. Normative references

***[ECMASCRIPT]***

ECMA-262 ECMAScript Language Specification, Edition 6. Draft.

URL: http://people.mozilla.org/~jorendorff/es6-draft.html

***[HTML5]***

Robin Berjon; Steve Faulkner; Travis Leithead; Erika Doyle Navara; Edward O'Connor; Silvia Pfeiffer. HTML5. 17 June 2014. W3C Last Call Working Draft.

URL: http://www.w3.org/TR/html5/

***[RFC2119]***

S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. March 1997. Best Current Practice.

URL: http://www.ietf.org/rfc/rfc2119.txt

***[WEBIDL]***

Cameron McCormack. Web IDL. 19 April 2012. W3C Candidate Recommendation.

URL: http://www.w3.org/TR/WebIDL/