Standby API Specification

February 2014

Editor: Dariel Marlow

Draft for W3C Proposed Recommendation

Abstract

This specification defines an API that provides scripted access to device standby behavior.

Table of Contents

Conformance Requirements

All diagrams, examples, and notes in this specification are non-normative, as are all sections explicitly marked non-normative. Everything else in this specification is normative.

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in the normative parts of this document are to be interpreted as described in RFC2119. For readability, these words do not appear in all uppercase letters in this specification. [RFC2119]

Requirements phrased in the imperative as part of algorithms (such as "strip any leading space characters" or "return false and abort these steps") are to be interpreted with the meaning of the key word ("must", "should", "may", etc.) used in introducing the algorithm.

Conformance requirements phrased as algorithms or specific steps may be implemented in any manner, so long as the end result is equivalent. (In particular, the algorithms defined in this specification are intended to be easy to follow, and not intended to be performant.)

User agents may impose implementation-specific limits on otherwise unconstrained inputs, e.g. to prevent denial of service attacks, to guard against running out of memory, or to work around platform-specific limitations.

The IDL blocks in this specification are conforming IDL fragments as defined by the WebIDL specification. [WEBIDL]

Introduction

This section is non-normative

This specification introduces a mechanism for handling device standby behavior control.

It is designed for scenarios where control of device standby behavior is desired to be altered. For example, an application may request that a device does not enter standby mode due to a prolonged period of user inactivity (i.e. no input from user via touch or peripherals).

The following code extract illustrates how to control the device standby behavior:

```
// Disables the device from entering standby and thus
// prevents screen from turning off.
var standbyErr = powerManagement.device.disableStandby();

// Gets the state of the device standby behavior.
var standbyEnabled = powerManagement.device.isStandbyEnabled();

// Gives control back to the device to allow it to enter
// standby if it chooses (i.e. no other application
// has also requested to disable device standby).
powerManagement.device.enableStandby();
```

Security and Privacy Considerations

This specification defines control of standby behavior. Thereby, in most cases, having an impact in the overall operable duration of a device when using an internal power source. Therefore, a conforming implementation of this specification should, at the very least, inform the user, and preferably, request permission, when the device standby behavior is altered programmatically.

API Description

PowerManagement Interface

The PowerManagement object is used by scripts to programmatically control the device standby behavior. The control is performed by a user-agent specific algorithm.

```
[NoInterfaceObject]
interface IPowerManagement {
      readonly attribute Device device;
}
```

Objects implementing the PowerManagement interface (e.g. the window.powerManagement object) must also implement the IPowerManagent interface. An instance of IPowerManagment would then be obtainable by using binding-specific casting methods on an instance of PowerManagement.

Device Interface

```
[NoInterfaceObject]
interface Device {
      StandbyError disableStandby();
      bool isStandbyEnabled();
      void enableStandby();
}
```

When the `disableStandby()` method is called, it should prompt the user asking for permission to allow the device standby to be altered. The user-agent is responsible for altering the device properties to allow the device to refrain from entering standby. The `isStandbyEnabled()` method serves to check the state of standby behavior. The `enableStandby()` method allows an application to restore standby behavior; thereby, allowing the device to enter standby should there be no other requests to disable standby.

StandbyError Interface

```
[NoInterfaceObject]
interface StandbyError {
      const unsigned short PERMISSION_DENIED = 1;
      readonly attribute unsigned short code;
      readonly attribute DOMString message;
}
```

The code attribute must return the appropriate code from the following list:

- PERMISSION_DENIED (numeric value 1) - Disabling standby process failed because the document does not have permission to use the PowerManagement API.

The message attribute must return an error message describing the details of the error occurred. This attribute is primarily intended for debugging and developers should not use it directly in their application user interface. It is represented as a DOMString.

Use-Cases and Requirements

Use-Cases

Long running websites

A user visiting a website that contains an embedded HTML5 video that does not play in full screen on the device or a long running visual animation. This video, or animation, may be longer than the device's current standby timeout value. The web application may adjust the device standby timeout to provide a better experience.

Games that don’t require direct touch or peripheral input

A user may engage in a web driven game that uses a sensor input on a device. For example, a jump game driven by an accelerometer where the user must tilt the device to play the game. Again, the duration of gameplay may exceed that of the device standby timeout value.

Navigation

A user may wish to utilize a web application containing map features while operating a vehicle. The device may enter standby due to prolonged periods of inactivity.

Requirements

- The standby API must provide a way for an application to prevent a device from entering standby from a lack of user initiated input.
- The standby API must provide a way for an application to return control back to the device should it no longer require the device from entering standby.
- The standby API must be agnostic to the underlying hardware platforms that differ in power management routines.
- The user-agent must revert standby control to the device upon the user navigating away from the web page that requested standby to be disabled.

References

[DOMString]

The DOMString Type, Arnaud Le Hors, Philippe Le Hégaret, Gavin Nicol, Lauren Wood, Mike Champion, Steve Byrne, Editors. World Wide Web Consortium, 7 April 2004. See http://www.w3.org/TR/DOM-Level-3-Core/core.html#DOMString

[RFC2119]

Key words for use in RFCs to Indicate Requirement Levels, Scott Bradner. Internet Engineering Task Force, March 1997. See http://www.ietf.org/rfc/rfc2119.txt

[WEBIDL]

web IDL, Cameron McCormack, Editor. World Wide Web Consortium, 19 April 2012. See http://www.w3.org/TR/2012/CR-WebIDL-20120419/