firebase-arduino Documentation

Release 1.0

firebase-arduino

Nov 17, 2017

Contents

FirebaseArduino is a library to simplify connecting to the Firebase database from arduino clients.

It is a full abstraction of Firebase's REST API exposed through C++ calls in a wiring friendly way. All Json parsing is handled by the library and you may deal in pure C/Arduino types.

class `FirebaseArduino`

Main class for Arduino clients to interact with Firebase.

This implementation is designed to follow Arduino best practices and favor simplicity over all else. For more complicated usecases and more control see the Firebase class in Firebase.h.

Public Functions

void **`begin`**(**const** String &*host*, **const** String &*auth* = "")

Must be called first.

This initialize the client with the given firebase host and credentials.

Parameters

- `host`: Your firebase db host, usually X.firebaseio.com.
- `auth`: Optional credentials for the db, a secret or token.

String **`pushInt`**(**const** String &*path*, int *value*)

Appends the integer value to the node at path.

Equivalent to the REST API's POST. You should check *success()* after calling.

Return The unique key of the new child node.

Parameters

- `path`: The path of the parent node.
- `value`: Integer value that you wish to append to the node.

String **`pushFloat`**(**const** String &*path*, float *value*)

Appends the float value to the node at path.

Equivalent to the REST API's POST. You should check *success()* after calling.

Return The unique key of the new child node.

Parameters

- `path`: The path of the parent node.
- `value`: Float value that you wish to append to the node.

String **`pushBool`**(**const** String &*path*, bool *value*)

Appends the bool value to the node at path.

Equivalent to the REST API's POST. You should check *success()* after calling.

Return The unique key of the new child node.

Parameters

- `path`: The path of the parent node.
- `value`: Bool value that you wish to append to the node.

String **pushString**(**const** String &*path*, **const** String &*value*)
Appends the String value to the node at path.

Equivalent to the REST API's POST. You should check *success()* after calling.

Return The unique key of the new child node.

Parameters

- `path`: The path of the parent node.
- `value`: String value that you wish to append to the node.

String **push**(**const** String &*path*, **const** JsonVariant &*value*)
Appends the JSON data to the node at path.

Equivalent to the REST API's POST. You should check *success()* after calling.

Return The unique key of the new child node.

Parameters

- `path`: The path of the parent node.
- `value`: JSON data that you wish to append to the node.

void **setInt**(**const** String &*path*, int *value*)
Writes the integer value to the node located at path equivalent to the REST API's PUT.

You should check *success()* after calling.

Parameters

- `path`: The path inside of your db to the node you wish to update.
- `value`: Integer value that you wish to write.

void **setFloat**(**const** String &*path*, float *value*)
Writes the float value to the node located at path equivalent to the REST API's PUT.

You should check *success()* after calling.

Parameters

- `path`: The path inside of your db to the node you wish to update.
- `value`: Float value that you wish to write.

void **setBool**(**const** String &*path*, bool *value*)
Writes the bool value to the node located at path equivalent to the REST API's PUT.

You should check *success()* after calling.

Parameters

- `path`: The path inside of your db to the node you wish to update.
- `value`: Bool value that you wish to write.

void **setString**(**const** String &*path*, **const** String &*value*)
Writes the String value to the node located at path equivalent to the REST API's PUT.

You should check *success()* after calling.

Parameters

- `path`: The path inside of your db to the node you wish to update.

- `value`: String value that you wish to write.

void **`set`**(**const** String &*path*, **const** JsonVariant &*value*)
Writes the JSON data to the node located at path.

Equivalent to the REST API's PUT. You should check *success()* after calling.

Parameters

- `path`: The path inside of your db to the node you wish to update.
- `value`: JSON data that you wish to write.

int **`getInt`**(**const** String &*path*)
Gets the integer value located at path.

You should check *success()* after calling.

Return The integer value located at that path. Will only be populated if *success()* is true.

Parameters

- `path`: The path to the node you wish to retrieve.

float **`getFloat`**(**const** String &*path*)
Gets the float value located at path.

You should check *success()* after calling.

Return The float value located at that path. Will only be populated if *success()* is true.

Parameters

- `path`: The path to the node you wish to retrieve.

String **`getString`**(**const** String &*path*)
Gets the string value located at path.

You should check *success()* after calling.

Return The string value located at that path. Will only be populated if *success()* is true.

Parameters

- `path`: The path to the node you wish to retrieve.

bool **`getBool`**(**const** String &*path*)
Gets the boolean value located at path.

You should check *success()* after calling.

Return The boolean value located at that path. Will only be populated if *success()* is true.

Parameters

- `path`: The path to the node you wish to retrieve.

FirebaseObject **`get`**(**const** String &*path*)
Gets the json object value located at path.

You should check *success()* after calling.

Return a *FirebaseObject* value located at that path. Will only be populated if *success()* is true.

Parameters

- `path`: The path to the node you wish to retrieve.

void **remove**(**const** String &*path*)
Remove the node, and possibly entire tree, located at path.

You should check *success()* after calling.

Parameters

- `path`: The path to the node you wish to remove, including all of its children.

void **stream**(**const** String &*path*)
Starts streaming any changes made to the node located at path, including any of its children.

You should check *success()* after calling. This changes the state of this object. Once this is called you may start monitoring *available()* and calling *readEvent()* to get new events.

Parameters

- `path`: The path inside of your db to the node you wish to monitor.

bool **available**()
Checks if there are new events available.

This is only meaningful once *stream()* has been called.

Return If a new event is ready.

FirebaseObject **readEvent**()
Reads the next event in a stream.

This is only meaningful once *stream()* has been called.

Return *FirebaseObject* will have ["type"] that describes the event type, ["path"] that describes the effected path and ["data"] that was updated.

bool **success**()

Return Whether the last command was successful.

bool **failed**()

Return Whether the last command failed.

const String &**error**()

Return Error message from last command if *failed()* is true.

class **FirebaseObject**
Represents value stored in firebase, may be a singular value (leaf node) or a tree structure.

Public Functions

FirebaseObject(**const** char **data*)
Construct from json.

Parameters

- `data`: JSON formatted string.

bool **getBool**(**const** String &*path* = "") **const**
Return the value as a boolean.

Return result as a bool.

Parameters

- `optional`: path in the JSON object.

int **`getInt`**(**const** String &*path* = "") **const**
Return the value as an int.

Return result as an integer.

Parameters

- `optional`: path in the JSON object.

float **`getFloat`**(**const** String &*path* = "") **const**
Return the value as a float.

Return result as a float.

Parameters

- `optional`: path in the JSON object.

String **`getString`**(**const** String &*path* = "") **const**
Return the value as a String.

Return result as a String.

Parameters

- `optional`: path in the JSON object.

JsonVariant **`getJsonVariant`**(**const** String &*path* = "") **const**
Return the value as a JsonVariant.

Return result as a JsonVariant.

Parameters

- `optional`: path in the JSON object.

bool **`success`**() **const**

Return Whether there was an error decoding or accessing the JSON object.

bool **`failed`**() **const**

Return Whether there was an error decoding or accessing the JSON object.

const String &**`error`**() **const**

Return Error message if *failed()* is true.

Index

F