

Not sure what the difference is ??

#### ERROR ?? Event

The source event that triggered the webhook

{

"name": "TestData1",

"data": "{\"temp\":852 }",

"ttl": 60,

"published\_at": "2018-10-06T01:08:19.064Z",

"coreid": "22002a000647343339373536"

}

void loop() {

tempvalue = analogRead(tempsensor);

String data = String::format( "{\"temp\":%d }", tempvalue );

data = String::format( "{\"lat\":%f , \"lng\":%f}", lat, lng);

Particle.publish("TestData1", data, PRIVATE);

float lat = 39.7391536;

float lng = -104.984703;

data = String::format( "{\"lat\":%f, \"lng\":%f}", lat, lng);

Particle.publish("elevation", data, PRIVATE);

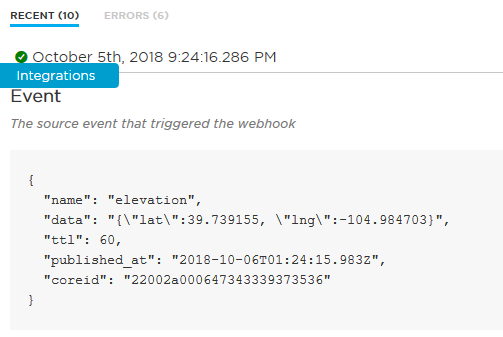
// Wait 60 seconds

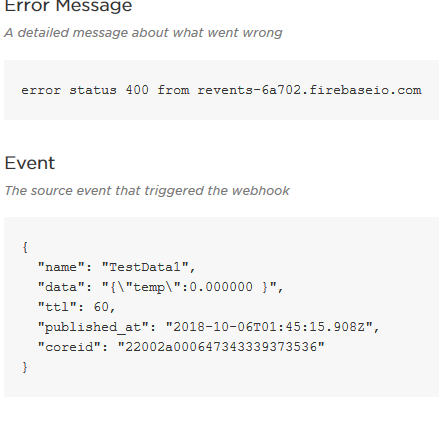
delay(60000);

}









void setup() {

// First, declare all of our pins. This lets our device know which ones will be used for outputting voltage, and which ones will read incoming voltage.

pinMode(led,OUTPUT); // Our LED pin is output (lighting up the LED)

pinMode(photoresistor,INPUT); // Our photoresistor pin is input (reading the photoresistor)

pinMode(power,OUTPUT); // The pin powering the photoresistor is output (sending out consistent power)

// Next, write the power of the photoresistor to be the maximum possible, so that we can use this for power.

digitalWrite(power,HIGH);

// We are going to declare a Particle.variable() here so that we can access the value of the photoresistor from the cloud.

Particle.variable("analogvalue", &analogvalue, INT);

// This is saying that when we ask the cloud for "analogvalue", this will reference the variable analogvalue in this app, which is an integer variable.

// We are also going to declare a Particle.function so that we can turn the LED on and off from the cloud.

Particle.function("led",ledToggle);

// This is saying that when we ask the cloud for the function "led", it will employ the function ledToggle() from this app.

}

float lat = 39.7391536;

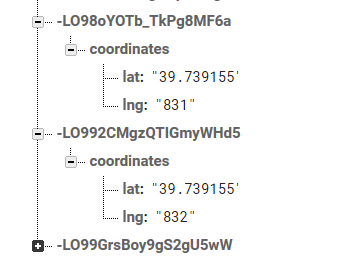
float lng = -104.984703;

data = String::format( "{\"lat\":%f, \"lng\":%d}", lat, tempvalue );

Particle.publish("elevation", data, PRIVATE);

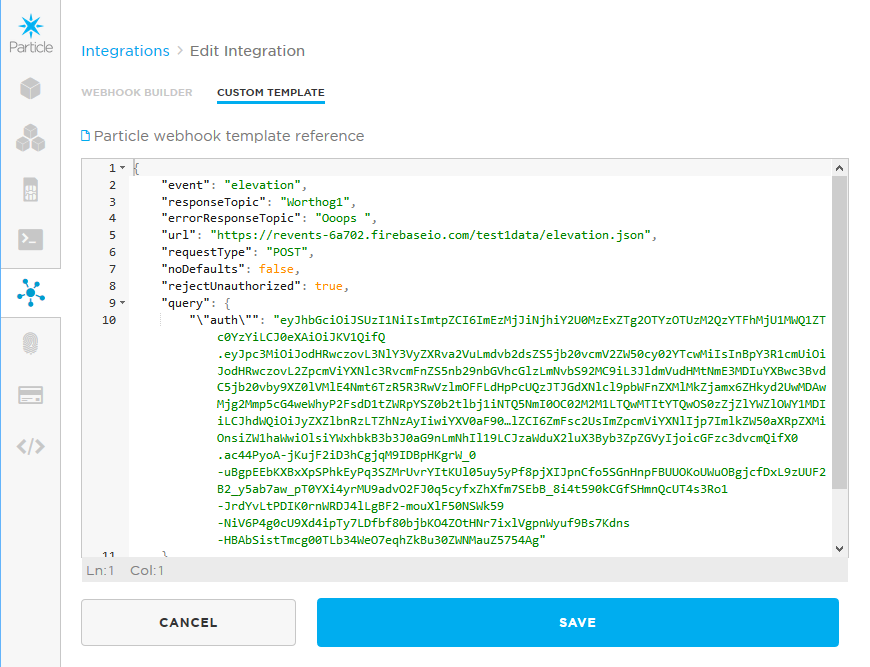
// Wait 60 seconds

delay(60000);

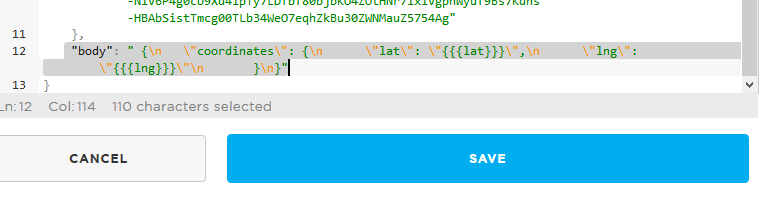


Not sure why this works ???

Check out the custom template



"body": " {\n \"coordinates\": {\n \"lat\": \"{{{lat}}}\",\n \"lng\": \"{{{lng}}}\"\n }\n}"



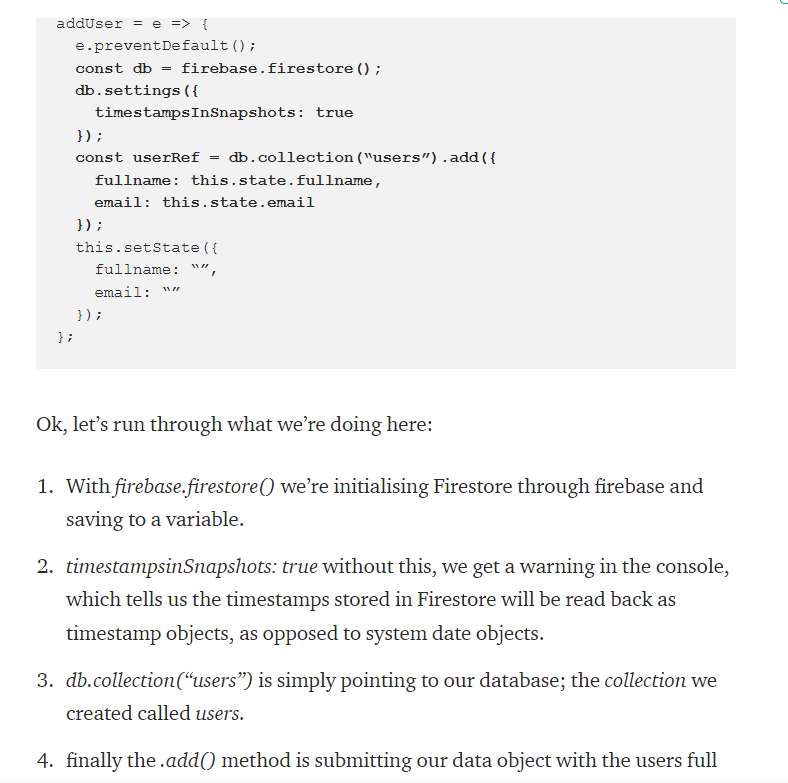
"body": " {\n \"coordinates\": {\n \"lat\": \"{{{lat}}}\",\n \"lng\": \"{{{lng}}}\"\n }\n}"

"body": "{\n \"event\": \"{{{PARTICLE\_EVENT\_NAME}}}\",\n \"data\": \"{{{PARTICLE\_EVENT\_VALUE}}}\",\n \"coreid\": \"{{{PARTICLE\_DEVICE\_ID}}}\",\n \"published\_at\": \"{{{PARTICLE\_PUBLISHED\_AT}}}\",

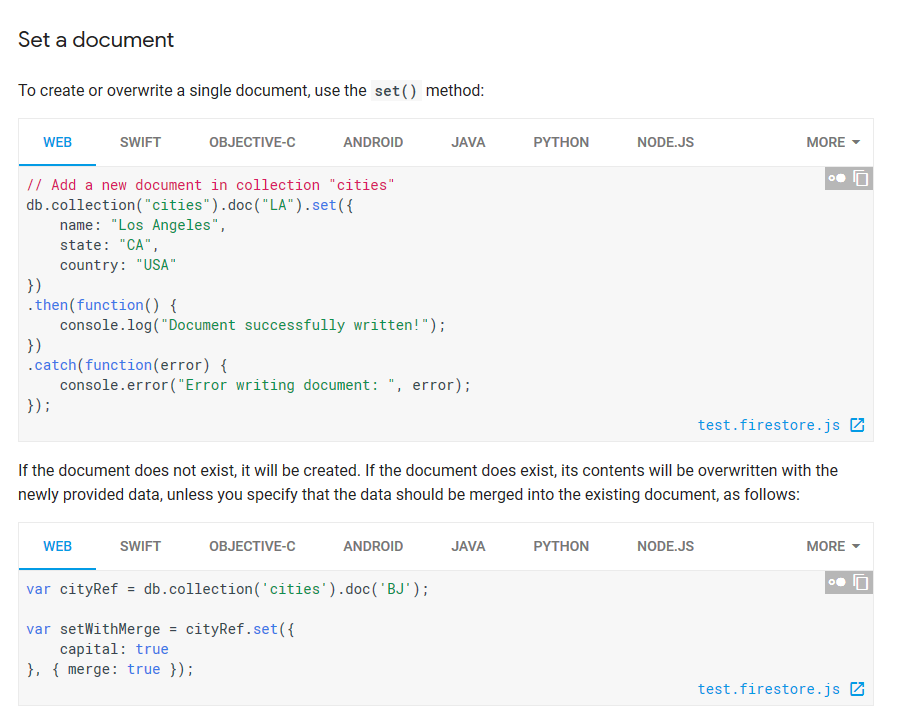
\n \"temperature\" : \ "{{{temp}}} \ "\n

}"

\n \"lng\" : \ "{{{lng}}} \ "\n



<https://firebase.google.com/docs/firestore/manage-data/add-data>



// Add a new document in collection "cities"  
db.collection("cities").doc("LA").set({  
    name: "Los Angeles",  
    state: "CA",  
    country: "USA"  
})  
.then(function() {  
    console.log("Document successfully written!");  
})  
.catch(function(error) {  
    console.error("Error writing document: ", error);  
});



// Add a new document with a generated id.  
db.collection("cities").add({  
    name: "Tokyo",  
    country: "Japan"  
})  
.then(function(docRef) {  
    console.log("Document written with ID: ", docRef.id);  
})  
.catch(function(error) {  
    console.error("Error adding document: ", error);  
});

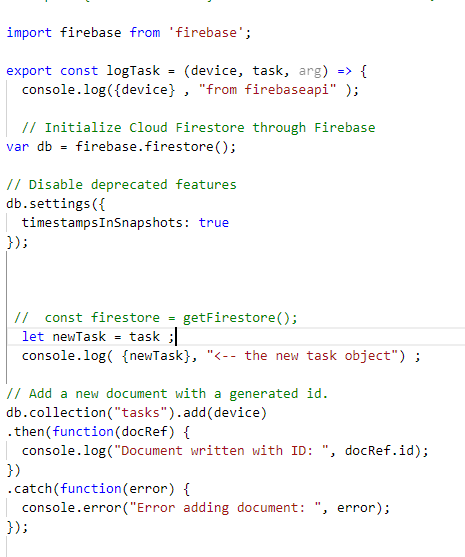
**Use the Cloud Firestore REST API**

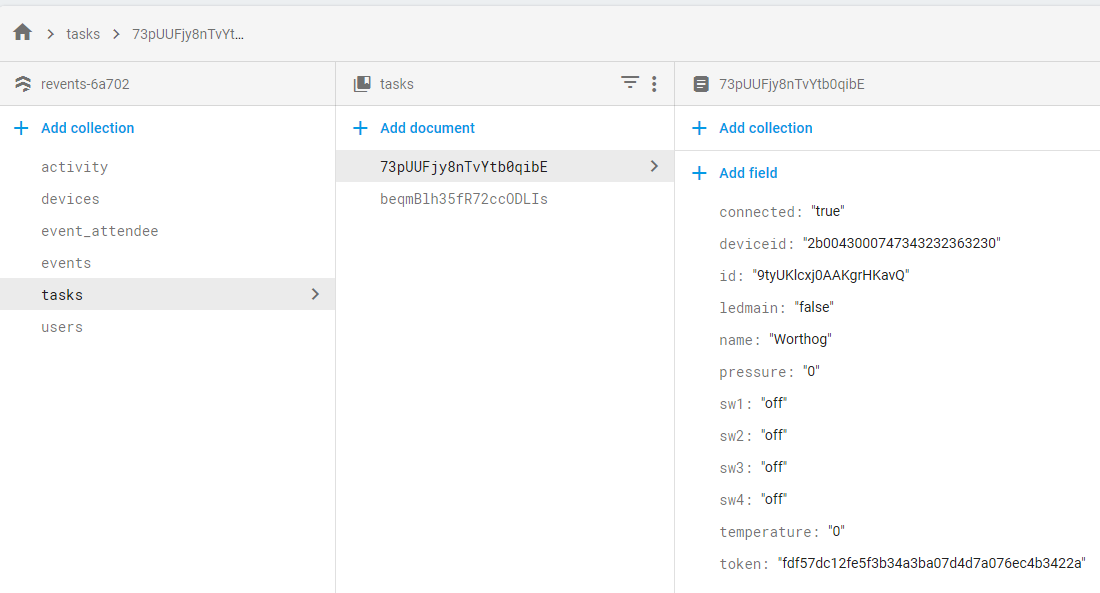
While the easiest way to use Cloud Firestore is to use one of the native client libraries, there are some situations when it is useful to call the REST API directly.

The REST API can be helpful for the following use cases:

* Accessing Cloud Firestore from a resource-constrained environment, such as an internet of things (IoT) device, where running a complete client library is not possible.
* Automating database administration or retrieving detailed database metadata.

If you are using a [gRPC-supported language](https://grpc.io/about/#osp), consider using the [RPC API](https://firebase.google.com/docs/firestore/reference/rpc/) rather than the REST API.

The following code “WORKS”, added the Device object to the Tasks db in Firestore   




<https://firebase.google.com/docs/firestore/query-data/get-data>

Get all documents in a collection

In addition, you can retrieve all documents in a collection by omitting the where() filter entirely:

WebSwiftObjective-CAndroidJavaPythonNode.js

More

db.collection("cities").get().then(function(querySnapshot) {

querySnapshot.forEach(function(doc) {

// doc.data() is never undefined for query doc snapshots

console.log(doc.id, " => ", doc.data());

});

});

Simple queries

The following query returns all cities with state CA:

WebSwiftObjective-CAndroidJavaPythonNode.js

More

// Create a reference to the cities collection

var citiesRef = db.collection("cities");

// Create a query against the collection.

var query = citiesRef.where("state", "==", "CA");

test.firestore.js

The following query returns all the capital cities:

WebSwiftObjective-CAndroidJavaPythonNode.js

More

var citiesRef = db.collection("cities");

var query = citiesRef.where("capital", "==", true);

test.firestore.js

The where() method takes three parameters: a field to filter on, a comparison operation, and a value. The comparison can be <, <=, ==, >, >=, or array\_contains. For iOS, Android, and Java, the comparison operator is explicitly named in the method.

Some example filters:

WebSwiftObjective-CAndroidJavaPythonNode.js

More

citiesRef.where("state", "==", "CA")

citiesRef.where("population", "<", 100000)

citiesRef.where("name", ">=", "San Francisco")

test.firestore.js

Array membership

You can use the array\_contains operator to filter based on array values. For example:

WebSwiftObjective-CAndroidJavaPythonNode.js

More

citiesRef.where("regions", "array-contains", "west\_coast")

test.firestore.js

This query returns every city document where the regions field is an array that contains west\_coast. If the array has multiple instances of the value you query on, the document is included in the results only once.

See if we can map the table to the tasks in the firestore state…

>>> Old version :

<Table.Body>

{\_.map(data, ({ deviceid, ledmain, name }) => (

<Table.Row key={name}>

<Table.Cell>{name}</Table.Cell>

<Table.Cell>{deviceid}</Table.Cell>

<Table.Cell>{ledmain}</Table.Cell>

</Table.Row>

))}

</Table.Body>

const todosList = !isLoaded(todos)

? 'Loading'

: isEmpty(todos)

? 'Todo list is empty'

: Object.keys(todos).map(

(key, id) => (

<TodoItem key={key} id={id} todo={todos[key]}/>

)

)

return (

</Table.Header>

<Table.Body>

{\_.map(tasks, ({ id, deviceid, ledmain, name }) => (

<Table.Row key={id}>

<Table.Cell>{name}</Table.Cell>

<Table.Cell>{deviceid}</Table.Cell>

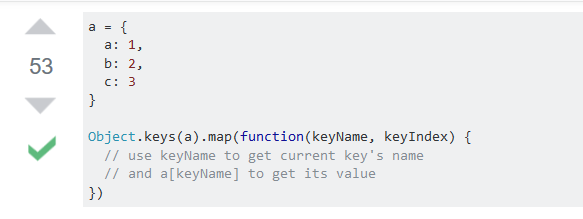
<Table.Cell>{ledmain}</Table.Cell>

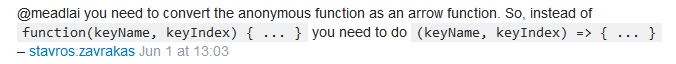
</Table.Row>

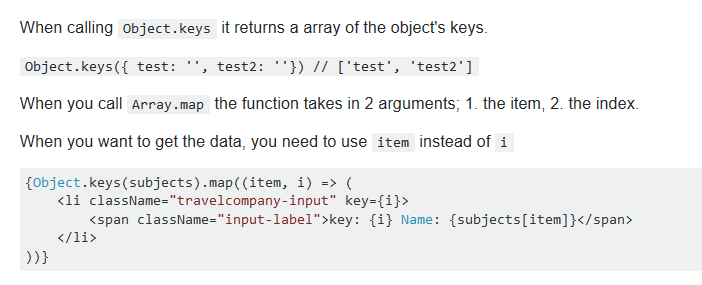
))}

</Table.Body>

</Table>







Map over the keys of the object using Object.keys():

{Object.keys(yourObject).map(function(key) {

return <div>Key: {key}, Value: {yourObject[key]}</div>;

})}

6 down vote accepted

Also, you can't use map for objects like this. You should write

var customer = Object.keys(Customers).map(function(s){ return Customers[s].name });

const arr = Object.keys(bands).map((key) => {

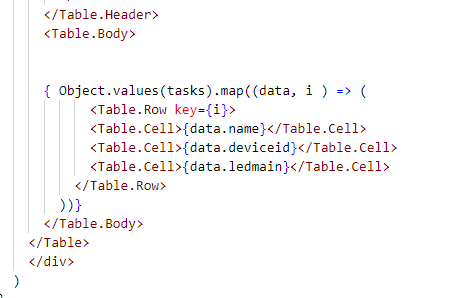
bands[key].\_id = key;

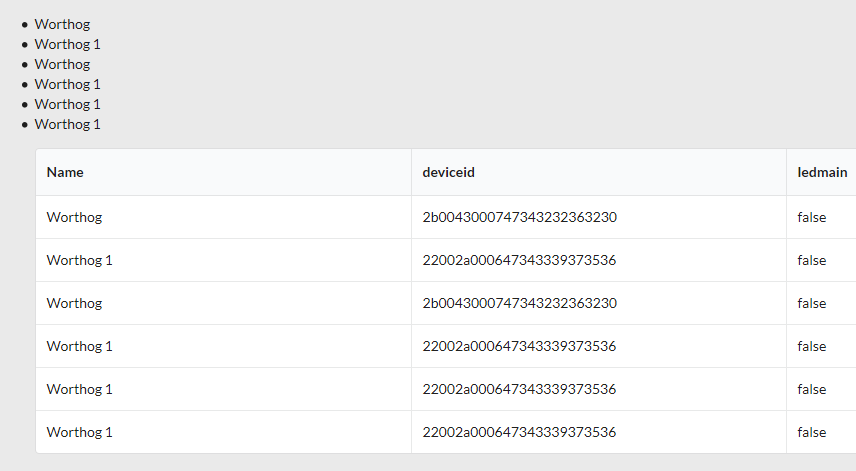
return bands.[key];

});

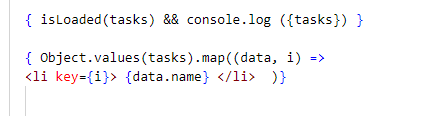
Finally got the map function to work with Firebase and allow us to use the Key value as the key for the react component : see stats.jsx

This version works





Note the debug code:



import { firestoreConnect, isLoaded, isEmpty } from 'react-redux-firebase';

render() {

const { tasks } = this.props;

const { column, data, direction } = this.state

if (!isLoaded(tasks) || isEmpty(tasks)) return <LoadingComponent inverted={true} />;