



Firestore

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

 <https://www.facebook.com/tutorialspointindia>

 <https://twitter.com/tutorialspoint>

About the Tutorial

Firebase is a backend platform for building Web, Android and IOS applications. It offers real time database, different APIs, multiple authentication types and hosting platform.

This is an introductory tutorial, which covers the basics of the Firebase platform and explains how to deal with its various components and sub-components.

Audience

This tutorial is directed towards developers in need for a simple, user-friendly backend platform. After you finish this tutorial, you will be familiar with the Firebase Web Platform. You can also use this as a reference in your future development.

This tutorial is intended to make you comfortable in getting started with the Firebase backend platform and its various functions.

Prerequisites

You will need some JavaScript knowledge to be able to follow this tutorial. Knowledge about some backend platform is not necessary, but it could help you to understand the various Firebase concepts.

Copyright and Disclaimer

© Copyright 2017 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial.....	i
Audience	i
Prerequisites	i
Copyright and Disclaimer	i
Table of Contents	ii
1. FIREBASE – OVERVIEW	1
2. FIREBASE – ENVIRONMENT SETUP	3
3. FIREBASE – DATA.....	6
4. FIREBASE – ARRAYS	8
5. FIREBASE – WRITE DATA.....	11
Set.....	11
Update	12
6. FIREBASE – WRITE LIST DATA	14
The Push Method	14
The Key Method.....	15
7. FIREBASE – WRITE TRANSACTIONAL DATA	17
8. FIREBASE – READ DATA	19
9. FIREBASE – EVENT TYPES.....	23
10. FIREBASE – DETACHING CALLBACKS.....	25

11. FIREBASE – QUERIES.....26

 Order by Child26

 Order by Key27

 Order by Value27

12. FIREBASE – FILTERING DATA.....30

 Limit to First and Last30

 Other Filters31

13. FIREBASE – BEST PRACTICES33

14. FIREBASE – EMAIL AUTHENTICATION34

 Create user.....34

 Sign In35

 Signout.....35

15. FIREBASE – GOOGLE AUTHENTICATION37

16. FIREBASE – FACEBOOK AUTHENTICATION.....40

17. FIREBASE – TWITTER AUTHENTICATION44

18. FIREBASE – GITHUB AUTHENTICATION.....47

19. FIREBASE – ANONYMOUS AUTHENTICATION50

20. FIREBASE – OFFLINE CAPABILITIES.....52

21. FIREBASE – SECURITY54

 Read and Write54

22. FIREBASE – DEPLOYING56

1. FIREBASE – OVERVIEW

As per official Firebase documentation –

Firestore can power your app's backend, including data storage, user authentication, static hosting, and more. Focus on creating extraordinary user experiences. We will take care of the rest. Build cross-platform native mobile and web apps with our Android, iOS, and JavaScript SDKs. You can also connect Firestore to your existing backend using our server-side libraries or our REST API.

Firestore Features

- **Real-time Database** – Firestore supports JSON data and all users connected to it receive live updates after every change.
- **Authentication** – We can use anonymous, password or different social authentications.
- **Hosting** – The applications can be deployed over secured connection to Firestore servers.

Firestore Advantages

- It is simple and user friendly. No need for complicated configuration.
- The data is real-time, which means that every change will automatically update connected clients.
- Firestore offers simple control dashboard.
- There are a number of useful services to choose.

Firestore Limitations

- Firestore free plan is limited to 50 Connections and 100 MB of storage.

In the next chapter, we will discuss the environment setup of Firestore.

2. FIREBASE – ENVIRONMENT SETUP

In this chapter, we will show you how to add Firebase to the existing application. We will need **NodeJS**. Check the link from the following table, if you do not have it already.

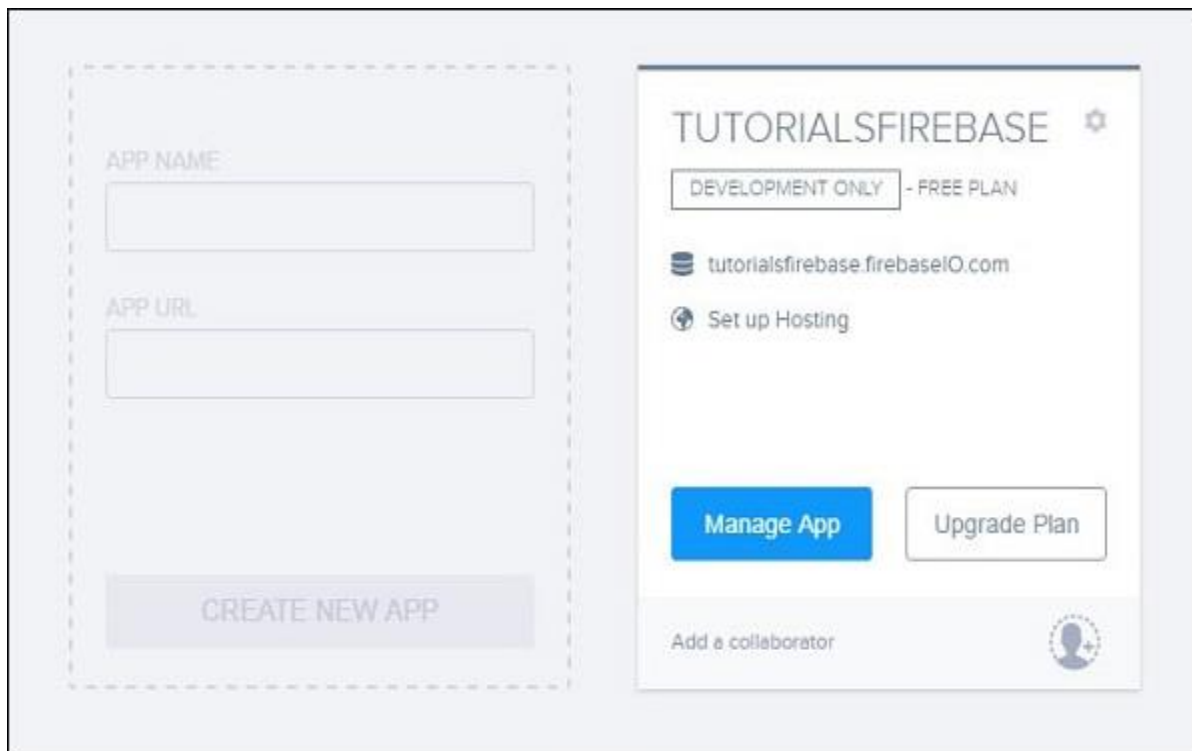
S No.	Software & Description
1	NodeJS and NPM NodeJS is the platform needed for Firebase development. Checkout our NodeJS Environment Setup .

Step 1 – Create a Firebase Account

You can create a Firebase account [here](#).

Step 2 – Create Firebase App

You can create new app from the dashboard page. The following image shows the app we created. We can click the **Manage App** button to enter the app.



Step 3a – Create a basic HTML/JS App

You just need to create a folder where your app will be placed. Inside that folder, we will need **index.html** and **index.js** files. We will add Firebase to the header of our app.

index.html

```
<html>
  <head>
    <script src =
"https://cdn.firebase.com/js/client/2.4.2/firebase.js"></script>
    <script type = "text/javascript" src = "index.js"></script>
  </head>

  <body>

  </body>

</html>
```

Step 3b - Use NPM or Bower

If you want to use your existing app, you can use Firebase NPM or Bowers packages. Run one of the following command from your apps root folder.

```
npm install firebase --save
```

```
bower install firebase
```

3. FIREBASE – DATA

The Firebase data is representing JSON objects. If you open your app from Firebase dashboard, you can add data manually by clicking on the + sign.

We will create a simple data structure. You can check the image below.



In the previous chapter, we connected Firebase to our app. Now, we can log Firebase to the console.

```
console.log(firebase)
```



```

▼ Object {SDK_VERSION: "3.0.5", INTERNAL: Object} index.js:1
  ► INTERNAL: Object
  ► Promise: function Promise()
    SDK_VERSION: "3.0.5"
  ► User: function (a,b,c)
  ► app: function a(a)
    apps: (...)
  ► get apps: function ()
  ► auth: function (c)
  ► database: function (c)
  ► initializeApp: function (a,c)
  ► storage: function (c)
  ► __proto__: Object
  > |

```

We can create a reference to our player's collection.

```

var ref = firebase.database().ref('players');

console.log(ref);

```

We can see the following result in the console.

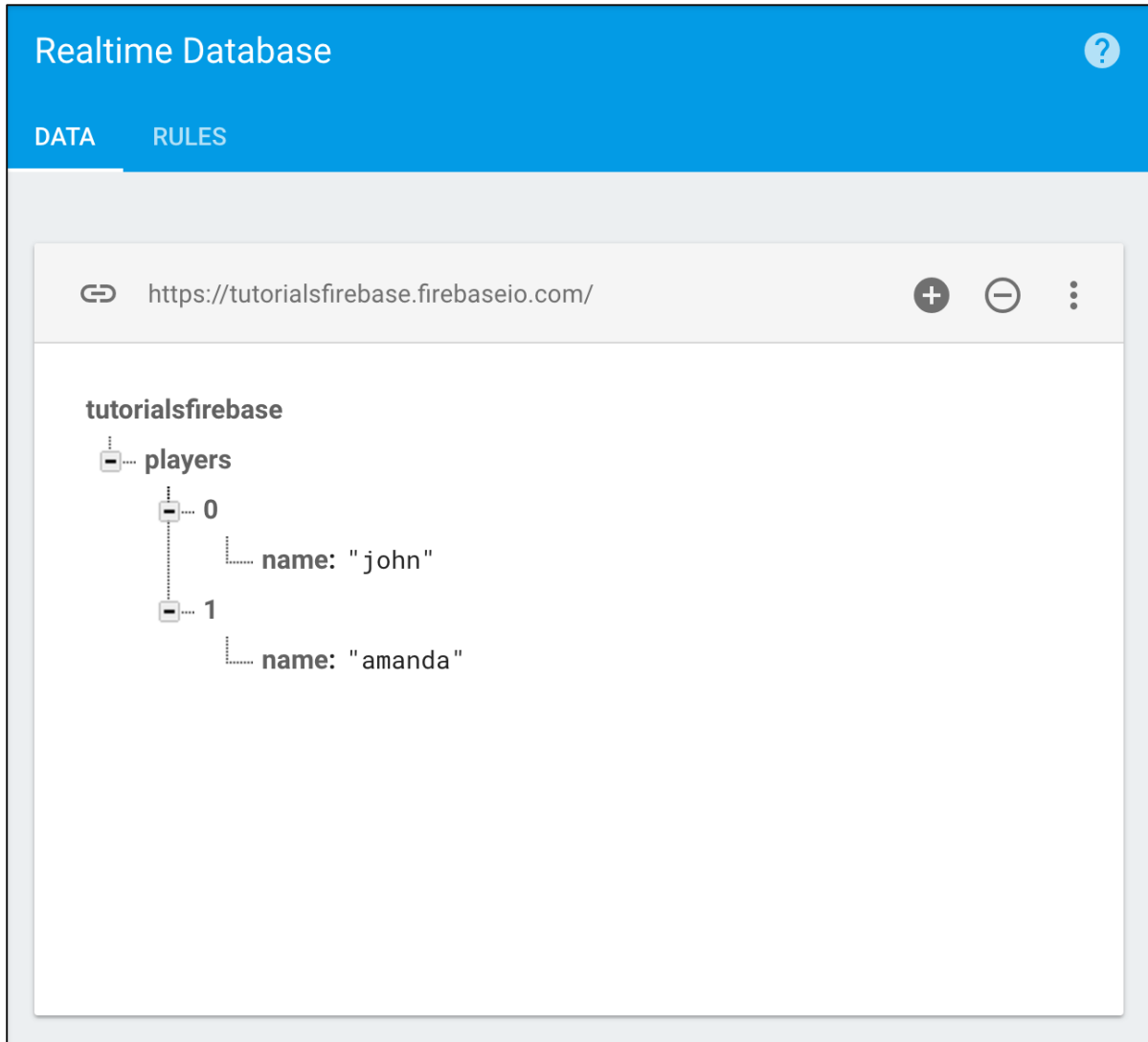
```

▼ U {w: pf, path: L, n: Ae, Oc: false, then: undefined...} index.js:3
  0c: false
  catch: undefined
  database: (...)
  key: (...)
  ► n: Ae
  parent: (...)
  ▼ path: L
    Z: 0
    ▼ o: Array[1]
      0: "players"
      length: 1
      ► __proto__: Array[0]
    ► __proto__: Object
  ref: (...)
  root: (...)
  then: undefined
  ► w: pf
  ► __proto__: X
  > |

```

4. FIREBASE – ARRAYS

This chapter will explain the Firebase representation of arrays. We will use the same data from the previous chapter.



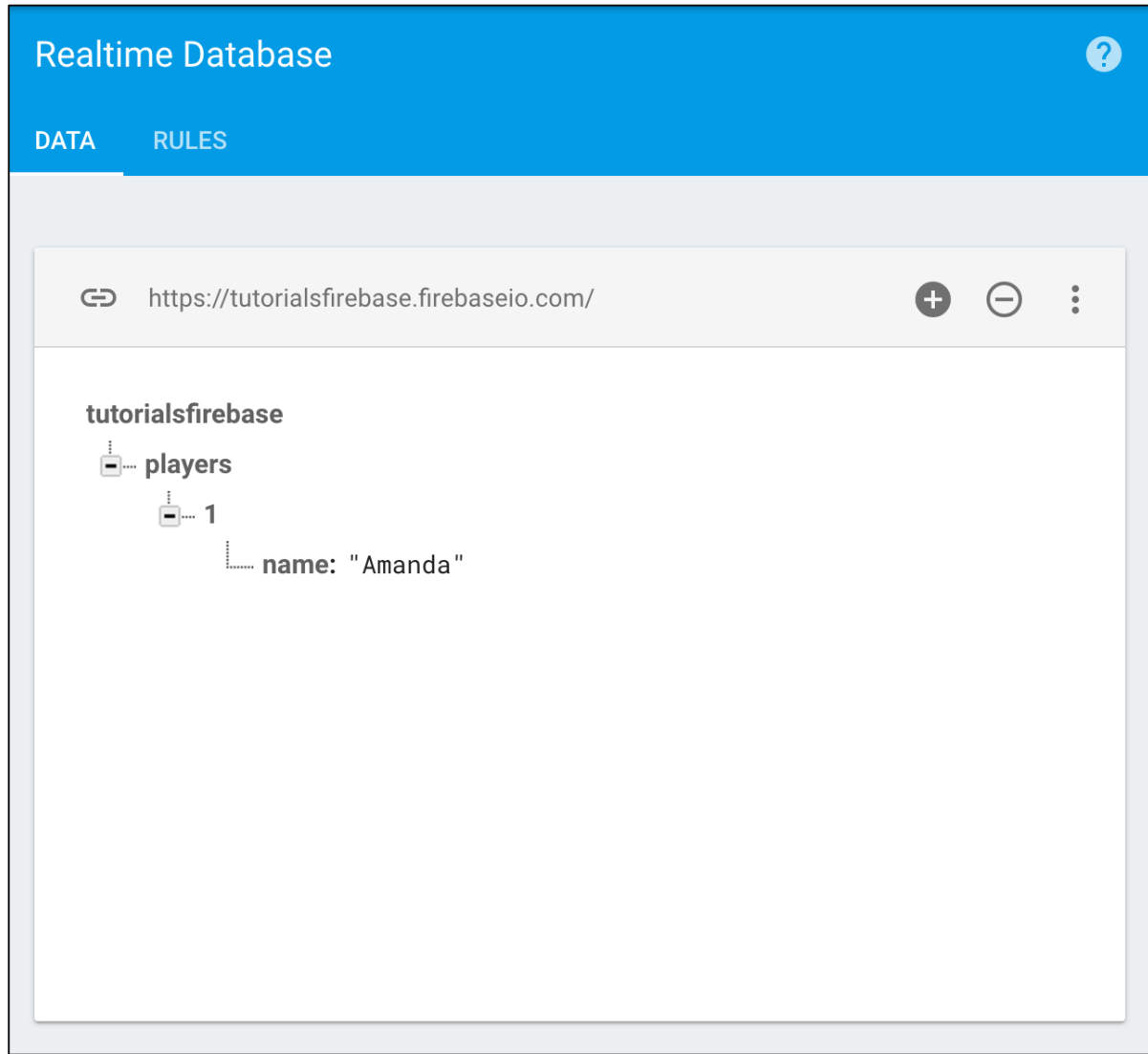
We could create this data by sending the following JSON tree to the player's collection.

```
['john', 'amanda']
```

This is because Firebase does not support Arrays directly, but it creates a list of objects with integers as key names.

The reason for not using arrays is because Firestore acts as a real time database and if a couple of users were to manipulate arrays at the same time, the result could be problematic since array indexes are constantly changing.

The way Firestore handles it, the keys (indexes) will always stay the same. We could delete **john** and **amanda** would still have the key (index) 1.



5. FIREBASE – WRITE DATA

In this chapter, we will show you how to save your data to Firebase.

Set

The **set** method will write or replace data on a specified path. Let us create a reference to the player's collection and set two players.

```
var playersRef = firebase.database().ref("players/");

playersRef.set({
  John: {
    number: 1,
    age: 30
  },

  Amanda: {
    number: 2,
    age: 20
  }
});
```

We will see the following result.



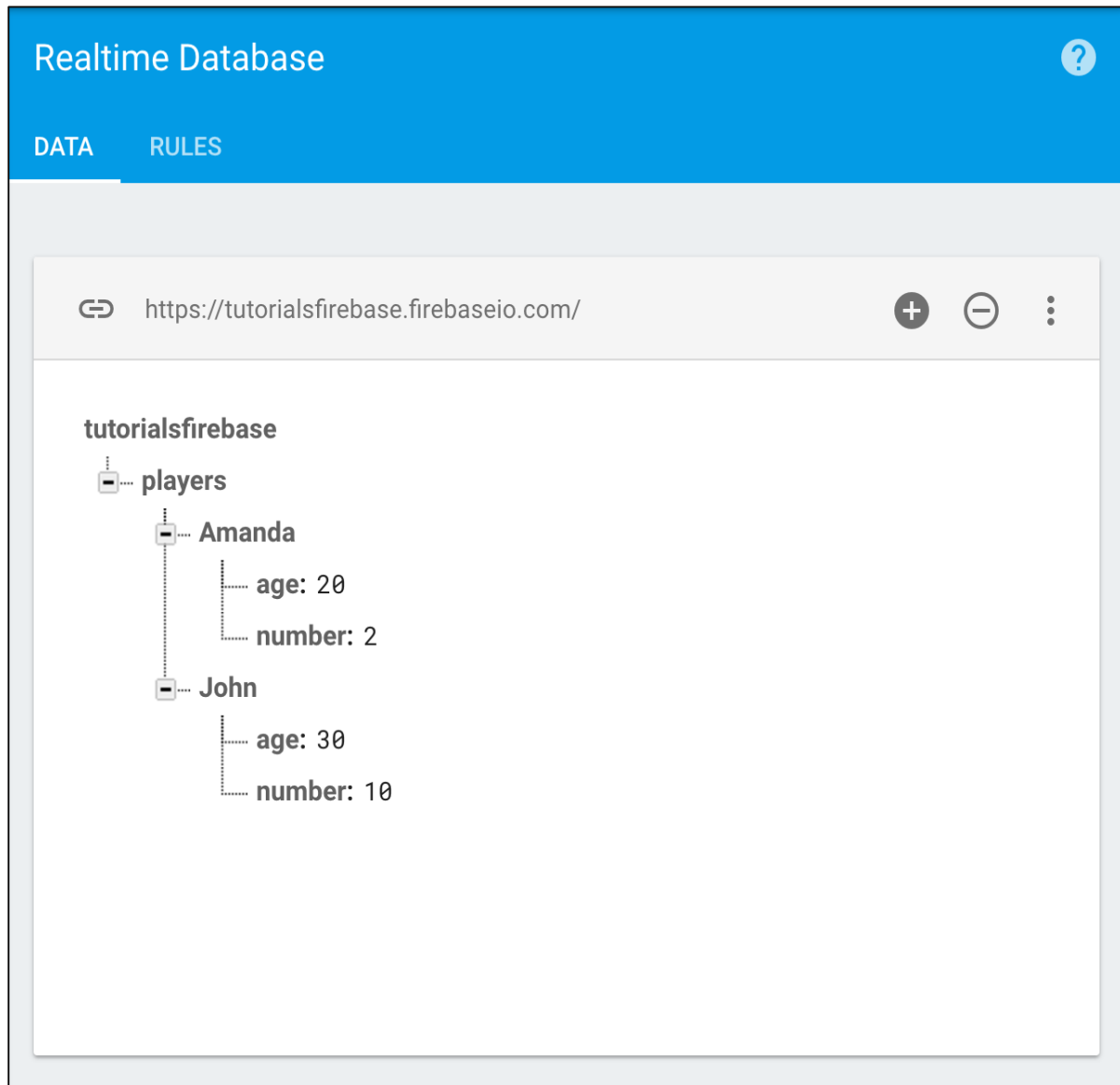
Update

We can update the Firebase data in a similar fashion. Notice how we are using the **players/john** path.

```
var johnRef = firebase.database().ref("players/John");

johnRef.update({
  "number": 10
});
```

When we refresh our app, we can see that the Firestore data is updating.



6. FIREBASE – WRITE LIST DATA

In our last chapter, we showed you how to write data in Firebase. Sometimes you need to have a unique identifier for your data. When you want to create unique identifiers for your data, you need to use the push method instead of the set method.

The Push Method

The **push()** method will create a unique id when the data is pushed. If we want to create our players from the previous chapters with a unique id, we could use the code snippet given below.

```
var ref = new Firebase('https://tutorialsfirebase.firebaseio.com');

var playersRef = ref.child("players");
playersRef.push({
  name: "John",
  number: 1,
  age: 30
});

playersRef.push({
  name: "Amanda",
  number: 2,
  age: 20
});
```

Now our data will look differently. The name will just be a name/value pair like the rest of the properties.

Realtime Database

DATA RULES

https://tutorialsfirebase.firebaseio.com/

tutorialsfirebase

- players
 - KMZ17JP4LKxISS_i1qn
 - age: 30
 - name: "John"
 - number: 1
 - KMZ17JTecKa_SsfhCiP
 - age: 20
 - name: "Amanda"
 - number: 2

The Key Method

We can get any key from Firebase by using the **key()** method. For example, if we want to get our collection name, we could use the following snippet.

```
var ref = new Firebase('https://tutorialsfirebase.firebaseio.com');  
  
var playersRef = ref.child("players");  
  
var playersKey = playersRef.key();  
console.log(playersKey);
```


End of ebook preview
If you liked what you saw...
Buy it from our store @ <https://store.tutorialspoint.com>