firebase

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.

- **Firebase Features**
- **Real-time Database − Firebase 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 Firebase servers.**

- **npm install firebase --save**
- **bower install 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}});
```

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});
```

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.

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);
```

Transactional data is used when you need to return some data from the database then make some calculation with it and store it back.
Let us say we have one player inside our player list.
We want to retrieve property, add one year of age and return it back to Firebase.
The amandaRef is retrieving the age from the collection and then we can use the transaction method. We will get the current age, add one year and update the collection.

```
var ref = new Firebase('https://tutorialsfirebase.firebaseio.com');var
amandaAgeRef = ref.child("players").child("-KGb1Ls-
gEErWbAMMnZC").child('age');

amandaAgeRef.transaction(function(currentAge) {return currentAge + 1;});
```

Read data

We can use the on() method to retrieve data. This method is taking the event type as "value" and then retrieves the snapshot of the data. When we add val() method to the snapshot, we will get the JavaScript representation of the data.

Example

Let us consider the following example.

```
var ref = firebase.database().ref();ref.on("value", function(snapshot) {
   console.log(snapshot.val());}, function (error) {
   console.log("Error: " + error.code);});
```

Queries

- **order by child**
- **order by value**
- **order by key**

filters

- **limitToFirst method returns the specified number of items beginning from the first one.**
- **limitToLast method returns a specified number of items beginning from the last one.**
- **startAt(), endAt() and the equalTo()**

Authentication

- email
- facebook
- twitter
- anonymous
- google