# Notes

## Realtime Database

See the Realtime Database docs for web.

### Set a ref

```
function writeUserData(userId, name, email, imageUrl) {
  firebase
    .database()
    .ref('users/' + userId)
    .set({
      username: name,
      email: email,
      profile_picture: imageUrl,
    });
}
```

### Value events

Value events fire with the entire data payload for any and all changes

> Listen to ongoing events

```
var starCountRef = firebase.database().ref('posts/' + postId + '/starCount');
starCountRef.on('value', function(snapshot) {
  updateStarCount(postElement, snapshot.val());
});
```

> Listen to a single event and stop listening

```
var userId = firebase.auth().currentUser.uid;
return firebase
  .database()
  .ref('/users/' + userId)
  .once('value')
  .then(function(snapshot) {
    var username = (snapshot.val() && snapshot.val().username) || 'Anonymous';
    // ...
  });
```

## Multi-path updates

```
function writeNewPost(uid, username, picture, title, body) {
  // A post entry.
  var postData = {
    author: username,
    uid: uid,
    body: body,
    title: title,
    starCount: 0,
    authorPic: picture,
  };

  // Get a key for a new Post.
  var newPostKey = firebase
    .database()
    .ref()
    .child('posts')
    .push().key;

  // Write the new post's data simultaneously in the posts list and the user's
post list.
  var updates = {};
  updates['/posts/' + newPostKey] = postData;
  updates['/user-posts/' + uid + '/' + newPostKey] = postData;

  return firebase
    .database()
    .ref()
    .update(updates);
}
```

## Delete data

```
function deleteUser(userId) {
  return firebase
    .database()
    .ref('/users/' + userId)
    .remove();
}
```

## Detach listener

```
var starCountRef = firebase.database().ref('posts/' + postId + '/starCount');
var listener = starCountRef.on('value', function(snapshot) {
  updateStarCount(postElement, snapshot.val());
});

function detachListener() {
  starCountRef.off('value', listener);
}
```

## Transactions

```
function toggleStar(postRef, uid) {
  postRef.transaction(function(post) {
    if (post) {
      if (post.stars && post.stars[uid]) {
        post.starCount--;
        post.stars[uid] = null;
      } else {
        post.starCount++;
        if (!post.stars) {
          post.stars = {};
        }
        post.stars[uid] = true;
      }
    }
    return post;
  });
}
```

## Child events

- **child_added**: fires once for every existing result and then again for every new result; does not fire for changes or removals, only new records
- **child_changed**: fires when the underlying object or value is changed in any way
- **child_removed**: fires when the entire record is removed

```
var commentsRef = firebase.database().ref('post-comments/' + postId);
commentsRef.on('child_added', function(data) {
  addCommentElement(postElement, data.key, data.val().text, data.val().author);
});

commentsRef.on('child_changed', function(data) {
  setCommentValues(postElement, data.key, data.val().text, data.val().author);
});

commentsRef.on('child_removed', function(data) {
  deleteComment(postElement, data.key);
});
```

## Sort data

- **orderByChild('childName')**: Orders by a child attribute
- **orderByKey()**: Orders by record keys
- **orderByValue()**: Orders by record values; only relevant when values are strings or numbers and not nested objects

```
var topUserPostsRef = firebase
  .database()
  .ref('user-posts/' + myUserId)
  .orderByChild('starCount');

var mostViewedPosts = firebase
  .database()
  .ref('posts')
  .orderByChild('metrics/views');
```

## Filter data

> Assumes that data is ordered by key unless otherwise specified

- **limitToFirst(count)**: Sets the maximum number of items to return from the beginning of the ordered list of results.
- **limitToLast(count)**: Sets the maximum number of items to return from the end of the ordered list of results.
- **startAt(value)**: Return items greater than or equal to the specified key or value, depending on the order-by method chosen.
- **endAt(value)**: Return items less than or equal to the specified key or value, depending on the order-by method chosen.
- **equalTo(value)**: Return items equal to the specified key or value, depending on the order-by method chosen.

```
var first100Days = firebase
  .database()
  .ref('days/2018')
  .orderByChild('dayOfYear')
  .limitToFirst(100);

var first10DaysOfFebruary = firebase
  .database()
  .ref('days/2018')
  .orderByChild('dayOfYear')
  .limitToFirst(10)
  .startAt(32);

var last10DaysOfJanuary = firebase
  .database()
  .ref('days/2018')
  .orderByChild('dayOfYear')
  .limitToLast(10)
  .endAt(31);

var first10DaysOfJanuary = firebase
  .database()
  .ref('days/2018')
  .orderByChild('dayOfYear')
  .limitToFirst(100) // Limit is never hit
  .endAt(10); // endAt stops the query before it hits the limit
```

## Authenticate Node.js

> Full admin privileges

```
var admin = require('firebase-admin');

// Fetch the service account key JSON file contents
var serviceAccount = require('path/to/serviceAccountKey.json');

// Initialize the app with a service account, granting admin privileges
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: 'https://databaseName.firebaseio.com',
});

// As an admin, the app has access to read and write all data, regardless of
Security Rules
var db = admin.database();
var ref = db.ref('restricted_access/secret_document');
ref.once('value', function(snapshot) {
  console.log(snapshot.val());
});
```

## Initialize Node.js with limited privileges

> Set auth token variables to limit access

```
// Initialize the app with a custom auth variable, limiting the server's access
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: 'https://databaseName.firebaseio.com',
  databaseAuthVariableOverride: {
    uid: 'my-service-worker',
  },
});
```

> Act as an un-authenticated user

```
// Initialize the app with a custom auth variable, limiting the server's access
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: 'https://databaseName.firebaseio.com',
  databaseAuthVariableOverride: null,
});
```