JAVASCRIPT

The Firebase Realtime Database

Firebase

The Firebase Realtime Database - облачная база данных.

Данные хранятся в JSON и синхронизируются в реальном времени с каждым подключенным клиентом. Все клиенты совместно используют один экземпляр базы данных Realtime и автоматически получают обновления с новейшими данными.

```
// Set the configuration for your app
// TODO: Replace with your project's config object
var config = {
  apiKey: "apiKey",
  authDomain: "projectId.firebaseapp.com",
  databaseURL: "https://databaseName.firebaseio.com",
  storageBucket: "bucket.appspot.com"
};
firebase.initializeApp(config);

// Get a reference to the database service
var database = firebase.database();
```

Database Rules

database.rules.json

DEFAULT

```
// These rules require authentication
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Database Rules

database.rules.json

DEFAULT

```
// These rules require authentication
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

PUBLIC

```
// These rules give anyone, even people who are not users of your app,
// read and write access to your database
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Database Rules

database.rules.json

DEFAULT

```
// These rules require authentication
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

USER

```
// These rules grant access to a node matching the authenticated
// user's ID from the Firebase auth token
{
  "rules": {
    "users": {
      "$uid": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

PUBLIC

```
// These rules give anyone, even people who are not users of your app,
// read and write access to your database
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Database Rules

database.rules.json

DEFAULT

```
// These rules require authentication
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

USER

```
// These rules grant access to a node matching the authenticated
// user's ID from the Firebase auth token
{
  "rules": {
    "users": {
      "$uid": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

PUBLIC

```
// These rules give anyone, even people who are not users of your app,
// read and write access to your database
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

PRIVATE

```
// These rules don't allow anyone read or write access to your database
{
  "rules": {
    ".read": false,
    ".write": false
  }
}
```

Database Structure

```
{
  // Chats contains only meta info about each conversation
  // stored under the chats's unique ID
  "chats": {
    "one": {
      "title": "Historical Tech Pioneers",
      "lastMessage": "ghopper: Relay malfunction found. Cause: moth.",
      "timestamp": 1459361875666
    },
    "two": { ... },
    "three": { ... }
  },

  // Conversation members are easily accessible
  // and stored by chat conversation ID
  "members": {
    // we'll talk about indices like this below
    "one": {
      "ghopper": true,
      "alovelace": true,
      "eclarke": true
    },
    "two": { ... },
    "three": { ... }
  },
```

```
  // Messages are separate from data we may want to iterate quickly
  // but still easily paginated and queried, and organized by chat
  // conversation ID
  "messages": {
    "one": {
      "m1": {
        "name": "eclarke",
        "message": "The relay seems to be malfunctioning.",
        "timestamp": 1459361875337
      },
      "m2": { ... },
      "m3": { ... }
    },
    "two": { ... },
    "three": { ... }
  }
}
```

Database Reference

```
var ref = firebase.database().ref('users');

ref = firebase.database().ref('users/' + userId);


// child method
var child = ref.child('test');

// parent method
var parent = ref.getParent()
ref.parent
```

Database writing

set()

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

update()

```
// 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;

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

Listen for value events

on()

```
var postsRef = firebase.database().ref('posts');
postsRef.on('value', function(snapshot) {
  console.log(snapshot.val());
});
```

once()

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

Delete data

remove()

```
firebase.database().ref().child('posts').remove()
```

set “null” as value

Lists of Data

push()

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

playersRef.push({
    name: "John",
    number: 1,
    age: 30
});

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

'child_added' EVENT

```
playersRef.on('child_added', function(data, prevChildKey) {
    var newPlayer = data.val();
    console.log('name: ' + newPlayer.name);
    console.log('age: ' + newPlayer.age);
    console.log('number: ' + newPlayer.number);
    console.log('Previous Player: ' + prevChildKey);
});
```

'child_removed' EVENT

```
playersRef.on('child_removed', function(data) {
    var deletedPlayer = data.val();
    console.log(deletedPlayer.name + ' has been deleted');
});
```

'child_changed' EVENT

```
playersRef.on('child_changed', function(data) {
    var player = data.val();
    console.log('The updated player name is ' + player.name);
});
```

JAVASCRIPT

The Firebase Realtime Database