Firebase (Version-9)

Firebase is also known as backend as a service , which means they provide backend services like day to day authentication, file storage etc.. and we can plug those services directly into our frontend applications or websites to make use of them and firebase takes care of all these services.

Firebase takes care of all the service side logic so that we developer need not to focus much. So it is an alternative for setting our backend infrastructure using tools like node js server and mongo db insted firebase handles all of that for us and we can focus on our frontend application.

“Version 8 of firebase uses objected-oriented approach where we call methods on firebase objects.”

“Version 9 uses tree-shaking approach where any unused function can be removed out from final bundled js file to do that we use module bundlers.”

→How to set-up WebPack?

1. Create a folder and open it in a text editor such as vscode. create another folder “src” and create a index.js file inside it which will be our javascript entry file. So this is the file where we will be writing all of our js code.
2. Create a dist folder where final bundle js code will live. Inside this folder create index.html file which will be the webpage that we serve to the browser .
3. Make sure nodejs is installed on your device.
4. open terminal inside code-editor and type `“npm i -y”` this will create a new package.json file which will keep track of all of our dependencies we need to install.
5. in terminal type `“npm i webpack webpack-cli -D”`
6. Create a “webpack.config” file inside the root of your project directory. Inside this file we will configure what we want webpack to do(load into our source index.js file and any other imports and then bundle all of that code together into a single bundle file).

```
//webpack.config

const path = require('path')

module.exports = {
    mode: 'development',     //specify mode
    entry: './src/index.js',  //entry file(where we want webpack to intially look for our j
    output: {
        path: path.resolve(__dirname,'dist'), //path we want output file to put into
        filename: 'bundle.js'  //file name for output file.
    },
    watch: true    //everytime we make any change the new code is rebundled up into bundle.
}
```

7. To run webpack make a custom script inside “package.json”

```
"scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
```

```
    "build": "webpack"  //when we run the script it will run the webpack command that will a
  },
```

8. To run the script type `npm run build` webpack will take our code and bundle it into output file inside dist folder. it will also be watching our index.js file for changes.
9. Link Your index.js to index.html using `<script src="bundle.js"></script>`

→Getting started with firebase.

1. go to firebase site and signup.
2. click on go to console where all of your firebase projects are listed.

→ In order to take advantage of tree-shaking in firebase 9 to remove any unused code we need a module bundler like webpack (if we are not using any CLI tools like react react app) a module bundler bundles different pieces of imported source code together into a final js bundle file.

3. Click on Add Project . Give a name to it . Click on create project (You will be directed to project dashboard)
4. Choose desired type of application such as webapp that you want to create.
5. Click on continue to console .(You will be directed to the frontend application of your project)
6. click on menu icon where "1 app is written" and click on setting icon. Scroll down and grab the code of config object. This config object contains information about our firebase project and we need to use it on our frontend so that our website gets connnect to firebase .
7. Paste config object inside index.js .

```
const firebaseConfig = {
  apiKey: "AIzaSyBHaReZ1w6Ai11i2s9EUQHBEOzINQEIjj8",
  authDomain: "fir-9-dojo-b6e7b.firebaseapp.com",
  projectId: "fir-9-dojo-b6e7b",
  storageBucket: "fir-9-dojo-b6e7b.appspot.com",
  messagingSenderId: "697877099",
  appId: "1:697877099:web:39fc6a1ea14e14dd87a72e"
};
```

8. Open terminal and run `"npm i firebase"`
9. Connect to the firebase backend.

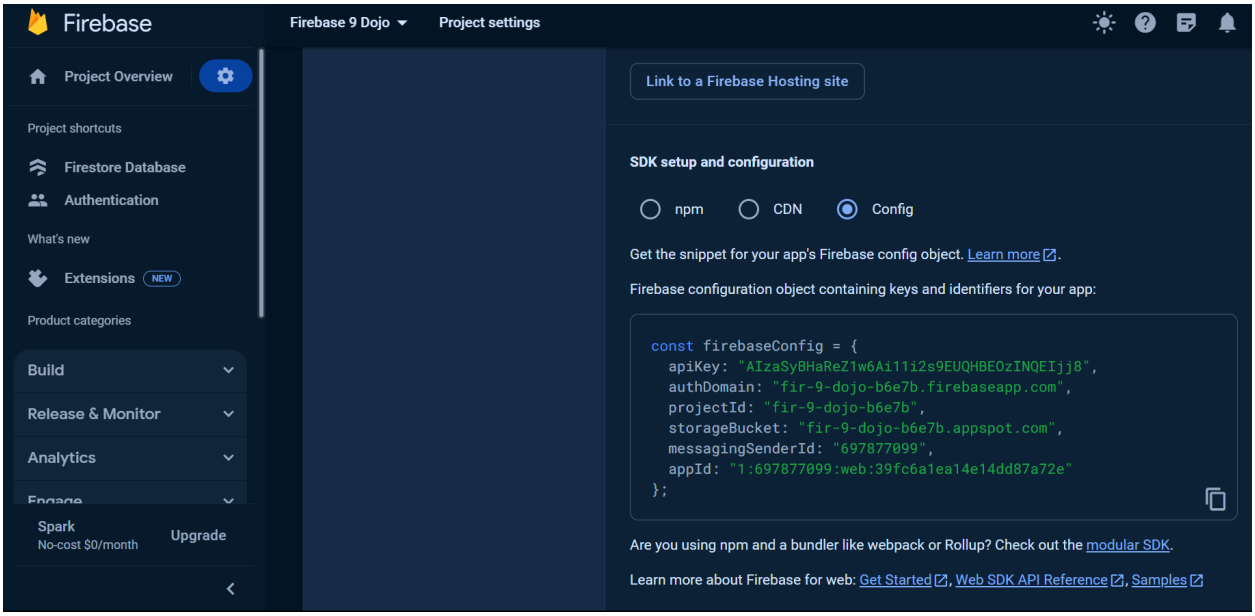
```
import {initializeApp} from 'firebase.app'

const firebaseConfig = {
  apiKey: "AIzaSyBHaReZ1w6Ai11i2s9EUQHBEOzINQEIjj8",
  authDomain: "fir-9-dojo-b6e7b.firebaseapp.com",
  projectId: "fir-9-dojo-b6e7b",
  storageBucket: "fir-9-dojo-b6e7b.appspot.com",
  messagingSenderId: "697877099",
  appId: "1:697877099:web:39fc6a1ea14e14dd87a72e"
};

firebase.initializeApp(firebaseConfig)
```

→Setting up Firebase database(Firestore) and connecting it to Frontend.

1. Go to your app inside firebase. Click on “Firestore Database” . Click create database button select test mode and press next. Choose firestore location(euro).



1. The database is split into collections(Collection of datatype like book) and documents.
2. Create on start collection and create a collection (set confugrations for your collection).
3. Create documents for prticular collection with properties(fields) and values.
4. Connect the database with your frontend as:-

```
import {initializeApp} from 'firebase.app'
import {getFirestore,collection,getDocs} from 'firebase/firestore'

const firebaseConfig = {
  apiKey: "AIzaSyBHaReZ1w6Ai11i2s9EUQHBEOzINQEIjj8",
  authDomain: "fir-9-dojo-b6e7b.firebaseapp.com",
  projectId: "fir-9-dojo-b6e7b",
  storageBucket: "fir-9-dojo-b6e7b.appspot.com",
  messagingSenderId: "697877099",
  appId: "1:697877099:web:39fc6a1ea14e14dd87a72e"
};

//init firebase app
initializeApp(firebaseConfig)
```

```
//init services
const db = getFirestore()

//collection reference
const colRef = collection(db,'books')

//get collection data
getDocs(colRef)
  .then((snapshot)=>{
    console.log(snapshot.docs)   //will return data(document) inform of arrays
    let books =[]
    snapshot.docs.forEach((doc)=>{
       books.push({...doc.data() , doc.id})
   })
    console.log(books)
  })
   .catch(err =>{
      console.log(err.message)
   })
```

→Adding and Deleting Documents.

```
<h1>Getting started with firebase 9</h1>

    <h2>Firebase Firestore</h2>

    <form class="add">
        <label for="title">Title:</label>
        <input type="text" name ="title" required>
        <label for="author">Author:</label>
        <input type="text" name="author" required>

        <button>Add a new book</button>
    </form>
  <br>
    <form class="delete">
       <label for="id">Document id:</label>
       <input type="text" name="id" required>
       <button>delete a book</button>
    </form>
```

```
import {addDoc,deleteDoc,doc(reference to doc)} from 'firebase/firestore'

//Adding documents
const addBookForm  = document.querySelector('.add')
```

```
addBookForm.addEventListener('submit',(e)=>{
    e.preventDefault()


    addDoc(colRef,{
        title: addBookForm.title.value,
        author: addBookForm.author.value,
    })
    .then(()=>{
        addBookForm.reset()
    })
})

//deleting documents
const deleteBookForm = document.querySelector('.delete')
deleteBookForm.addEventListener('submit',(e)=>{
    e.preventDefault();

    const docRef = doc(db,'books',deleteBookForm.id.value)

    deleteDoc(docRef)
       .then(()=>{
         deleteBookForm.reset()
       })
})
```

→Real-Time Collection Data

“So that data gets updated instantly , no need to referesh the page to see the logs data.”

onSnapshot

import {getFirestore,collection,getDocs} from 'firebase/firestoe'

```
import {onSnapshot} from 'firebase/firestore'

onSnapshot(colRef, (snapshot)=>{
    let books=[]
    snapshot.docs.forEach((doc)=>{
    books.push({...doc.data(), id: doc.id})
    })
    console.log(books)
})
```

→Filtering out Firebase queries.

```
const q = query(colRef,where("author","==","patrik rothfuss"))
onSnapshot(q, (snapshot)=>{
    let books=[]
    snapshot.docs.forEach((doc)=>{
```

```
    books.push({...doc.data(), id: doc.id})
    })
    console.log(books)
})
```

→Ordering data and timestamps.

“Go to the firebase and delete all the collections.

```
import {orderBy, serverTimestamp} from 'firebase/firestore'
//Adding documents
const addBookForm  = document.querySelector('.add')
addBookForm.addEventListener('submit',(e)=>{
    e.preventDefault()


    addDoc(colRef,{
        title: addBookForm.title.value,
        author: addBookForm.author.value,
        cretedAt: serverTimestamp()
    })
    .then(()=>{
        addBookForm.reset()
    })
})

const q = query(colRef,orderBy("createdAt"))
onSnapshot(q, (snapshot)=>{
    let books=[]
    snapshot.docs.forEach((doc)=>{
    books.push({...doc.data(), id: doc.id})
    })
    console.log(books)
})
```

→Fetching a Single Document.

```
import {getDoc} from 'firebase/firestore'

const docRef = doc(db,"books",'Q3y12py3X3sWFmJSytbq')
getDoc(docRef)
 .then((doc)=>{
    console.log(doc.data(), doc.id)
 })
```

→Updating Documents.

```
   <form class="update">
        <label for="id">Document id:</label>
        <input type="text" name="id" required>
        <button>update a book</button>

     </form>
```

```
import {updateDoc} from 'firebase/firestore'
const updateForm = document.querySelector('.update')
updateForm.addEventListener('submit',(e)=>{
    e.preventDefault()

    const docRef = doc(db,"books",updateForm.id.value)

    updateDoc(docRef,{
        title: 'updated title'
    })
    .then(()=>{
        updateForm.reset()
    })
})
```

→Setting Firebase Auth.

“Firebase auth uses json web token to authnticate with users who signup, sign in ,login, logout on the application.”

1. Go to authentication tab and click get started.
2. Click on email and password and enable the switch and click on save.

```
import {getAuth} from 'firebase/auth'

const auth = getAuth()
```

→Signing New Users Up

```
   <h2>Firebase auth</h2>

       <form class ="signup">
                <label for="email">email:</label>
                <input type="email" name="email">
                <label for="password">password:</label>
                <input type="password" name="password">
                <button>signup</button>
       </form>
```

```
import {createUserWithEmailAndPassword} from 'firebase/auth'

const signupForm =  document.querySelector('.signup')
signupForm.addEventListener('submit',(e)=>{
    e.preventDefault()
    const email = signupForm.email.value
    const password = signupForm.password.value

    createUserWithEmailAndPassword(auth,email,password)
       .then((cred)=>{
            // console.log('user created:  ',cred.user)
            signupForm.reset()
       })
       .catch((err)=>{
         console.log(err.message);
       })
})
```

→Login and LogOut

```
 <form class="login">

          <label for="email">email:</label>
          <input type="email" name="email">
          <label for="password">password:</label>
          <input type="password" name="password">
          <button>login</button>

       </form>

       <button class="logout">Logout</button>
```

```
import {
    signOut, signInWithEmailAndPassword,
} from 'firebase/auth

const logoutButton = document.querySelector('.logout')
logoutButton.addEventListener('click',()=>{
      signOut(auth)
       .then(()=>{
        // console.log('the user signed out')
      })
      .catch((err)=>{
        console.log(err.message)
      })

})
```

```
const loginForm = document.querySelector('.login')
loginForm.addEventListener('submit', (e)=>{
    e.preventDefault()
    const email =  loginForm.email.value
    const password = loginForm.password.value


    signInWithEmailAndPassword(auth,email,password)
      .then((cred)=>{
        // console.log('user logged in',cred.user)
      })
      .catch((err)=>{
        console.log(err.message)
      })
})
```

→Subscribing to Auth Changes

```
import {
    getAuth,
    createUserWithEmailAndPassword,
    signOut, signInWithEmailAndPassword,
    onAuthStateChanged
} from 'firebase/auth'

//subscribing to auth changes
const unsubAuth = onAuthStateChanged(auth,(user)=>{
  console.log("user status changed: ", user)
})
```

→Unsubscribing to Auth Changes

```
      <h2>Unsubscribing</h2>
      <button class="unsub">unsubscribe from db/auth changes</button>
```

```
const unsubButton = document.querySelector('.unsub')
unsubButton.addEventListener('click',()=>{

  console.log("unsubscribing")
  unsubCol()
  unsubDoc()
  unsubAuth()
})
```