Firebase

Authentication Security and Rules

Resources

https://www.firebase.com/docs/security/guide/securing-data.html

https://www.firebase.com/docs/security/bolt/quickstart.html

Understanding Security

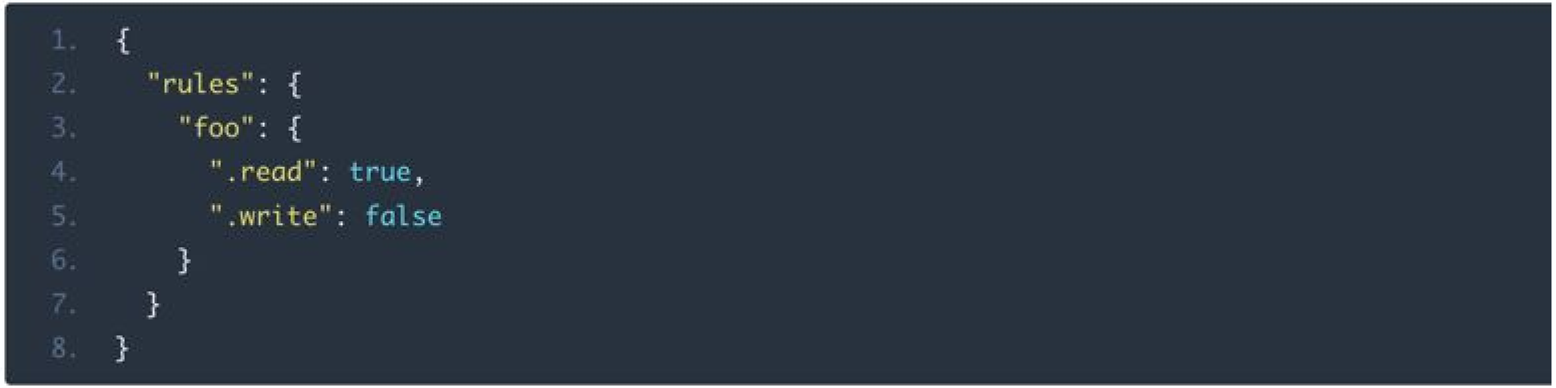
Firebase provides a full set of tools for managing the security of your app. These tools make it easy to authenticate your users, enforce user permissions, and validate inputs.

Firebase has a declarative language for specifying rules that live on the Firebase servers and determine the security of your app. You can edit them by selecting a Firebase app in your Account Dashboard and viewing the **Security & Rules** tab.

Security & Rules

These Security and Firebase Rules allow you to control access to each part of your database. Rules applied to a node in your database cascade to all of its children.

This example allows anyone to read /foo/ and its children but they can't write to it

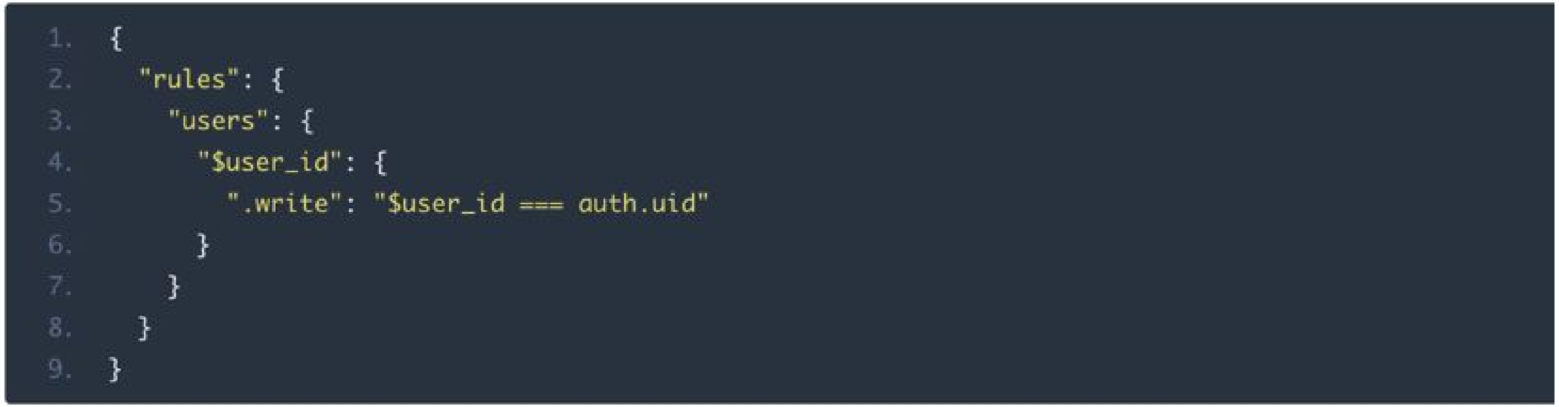


```
{
  "rules": {
    "foo": {
      ".read": true,
      ".write": false
    }
  }
}
```

Security & Rules

The Security and Firebase Rules include a number of built-in variables and functions. You can use these variables and functions to build expressive rules.

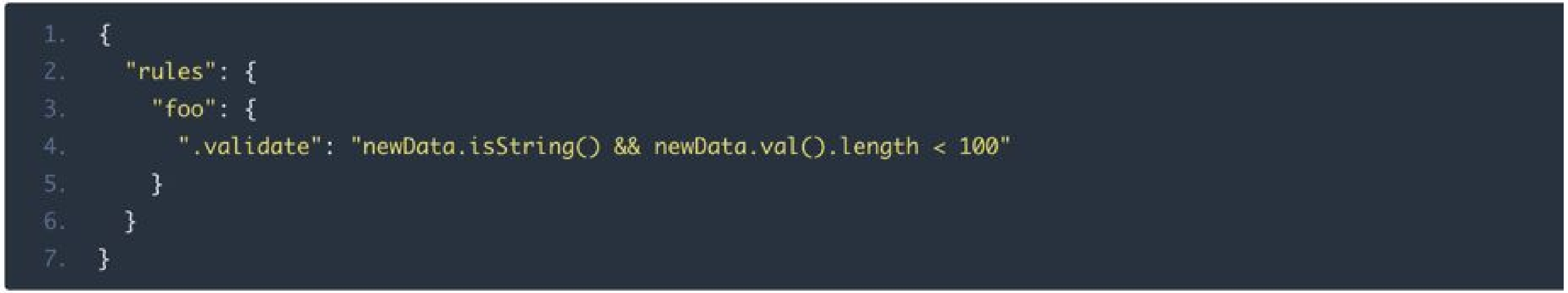
For instance: This rule grants a user write access on `/users/<auth.uid>/` to the user whose unique ID matches the dynamic path, `$user_id`.



```
{
  "rules": {
    "users": {
      "$user_id": {
        ".write": "$user_id === auth.uid"
      }
    }
  }
}
```

Security & Rules

Every Firebase application includes a schema-less database. This makes it easy to change things as you develop, but once your app is ready to distribute, it's important for data to stay consistent. The rules language includes a `.validate` rule. Use it to specify declarative validation rules just like `.read` and `.write` rules. The only difference is that validation rules do not cascade.



```
{
  "rules": {
    "foo": {
      ".validate": "newData.isString() && newData.val().length < 100"
    }
  }
}
```

In-class Exercise

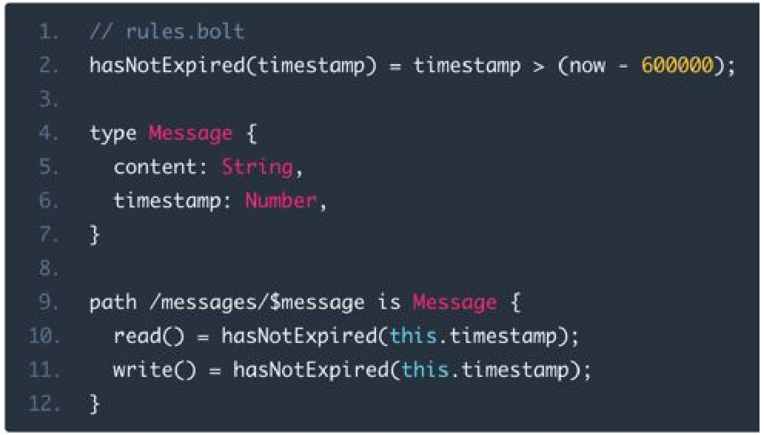
Add a .read, a .write, and a .validate rule to either your chat project, your to-do list app, or any other Firebase project you may have.

The End

The Bolt Compiler

The Bolt Compiler is a Type-based modeling and authorization language that compiles to JSON based security rules.

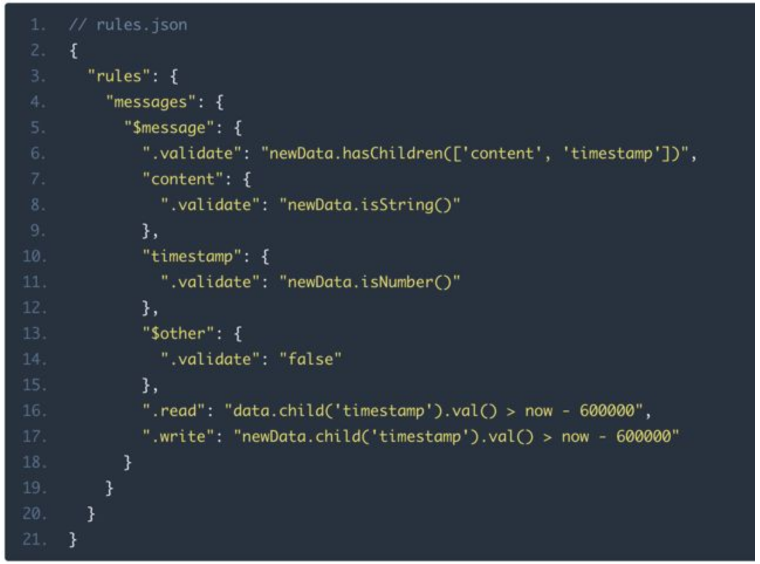
Turns this, into that --------------------->



```
// rules.bolt
hasNotExpired(timestamp) = timestamp > (now - 600000);

type Message {
  content: String,
  timestamp: Number,
}

path /messages/$message is Message {
  read() = hasNotExpired(this.timestamp);
  write() = hasNotExpired(this.timestamp);
}
```



```
// rules.json
{
  "rules": {
    "messages": {
      "$message": {
        ".validate": "newData.hasChildren(['content', 'timestamp'])",
        "content": {
          ".validate": "newData.isString()"
        },
        "timestamp": {
          ".validate": "newData.isNumber()"
        },
        "$other": {
          ".validate": "false"
        },
        ".read": "data.child('timestamp').val() > now - 600000",
        ".write": "newData.child('timestamp').val() > now - 600000"
      }
    }
  }
}
```