NoSQL

SWE 432, Fall 2017
Design and Implementation of Software for the Web
Today

• What is NoSQL?
• How can you create a NoSQL app with Firebase?
NoSQL

- non SQL, non-relational, "not only" SQL databases
- Emphasizes simplicity & scalability over support for relational queries
- Important characteristics
  - Schema-less: each row in dataset can have different fields (just like JSON!)
  - Non-relational: no structure linking tables together or queries to "join" tables
  - (Often) weaker consistency: after a field is updated, all clients eventually see the update but may see older data in the meantime
- Advantages: greater scalability, faster, simplicity, easier integration with code
- Several types. We'll look only at key-value.
Key-Value NoSQL

https://www.thoughtworks.com/insights/blog/nosql-databases-overview
Firebase Realtime Database

• Example of a NoSQL datastore
• Google web service
  • [https://firebase.google.com/docs/database/](https://firebase.google.com/docs/database/)
• Realtime database
  • Data stored to remote web service
  • Data synchronized to clients in real time
• Simple API
  • Offers library wrapping HTTP requests & responses
  • Handles synchronization of data
• Can also be used on frontend to build web apps with persistence without backend
Setting up Firebase

- Detailed instructions to create project, get API key
  - https://firebase.google.com/docs/web/setup

```html
<script src="https://www.gstatic.com/firebasejs/3.4.0/firebase.js"></script>
<script>
// Initialize Firebase
// TODO: Replace with your project's customized code snippet
var config = {
    apiKey: "<API_KEY>",
    authDomain: "<PROJECT_ID>.firebaseapp.com",
    databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
    storageBucket: "<BUCKET>.appspot.com",
};
firebase.initializeApp(config);
</script>
```
Setting up Firebase Realtime Database

• Go to https://console.firebase.google.com/, create a new project
• Install firebase module

```javascript
const firebase = require("firebase");

// Initialize Firebase
// TODO: Replace with your project's customized code snippet
const config = {
    apiKey: "<API_KEY>",
    authDomain: "<PROJECT_ID>.firebaseapp.com",
    databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
    storageBucket: "<BUCKET>.appspot.com",
};
firebase.initializeApp(config);
```

• Include Firebase in your web app
Get Config Object for your Project

- Go to Authentication, Click "Web Setup" in the upper right corner

---

Add Firebase to your web app

Copy and paste the snippet below at the bottom of your HTML, before other script tags.

```html
<script src="https://www.gstatic.com/firebasejs/4.4.0/firebase.js"></script>

// Initialize Firebase
var config = {
  apiKey: "AIzaSyDX--YJecTWhdO8t18WxqzNgnp39m06M",
  authDomain: "swe432-f17.firebaseapp.com",
  databaseURL: "https://swe432-f17.firebaseio.com",
  projectId: "swe432-f17",
  storageBucket: "swe432-f17.appspot.com",
  messagingSenderId: "806966638516"
};
firebase.initializeApp(config);
</script>
```

Check these resources to learn more about Firebase for web apps:

- Get Started with Firebase for Web Apps
- Firebase Web SDK API Reference
- Firebase Web Samples
Permissions

• By default, Firebase *requires* authentication
  • All unauthenticated requests will be refused
  • Do not want anyone with your URL to steal, destroy your production data
  • Will look at authentication in later lecture
  • For development, ok to allow anonymous access
Firebase Console

- See data values, updated in realtime
- Can edit data values

https://console.firebase.google.com
Simple test program

• After successfully completing previous steps, should be able to replace config and run this script. Can test by viewing data on console.

```javascript
const firebase = require("firebase");

// Initialize Firebase
const config = {
    ... [copied from your config object on Firebase]
};
firebase.initializeApp(config);

let database = firebase.database();
database.ref('users/' + 'Josh').set(
    { username: 'Josh', email: 'josh@gmail.com' }
);
```
Activity: Set up Firebase Realtime Database
Firebase data model: JSON

- JSON format data
- Hierarchic tree of key/value pairs
- Can view as one big object
- Or describe path to descendent and view descendent as object
Structuring data

• How do you organize lots of data?

• One idea: just have a hierarchic structure
  • Level 1: most general elements
  • Level 2: elements that belong to Level 1 elements
  • Level n: elements that belong to Level n-1 elements

• What are the pros and cons of this approach?
Structuring data

• Should be considering what types of records clients will be requesting.
• Do not want to force client to download data that do not need.
• Better to think of structure as lists of data that clients will retrieve.
• Can duplicate deeply nested structures with separate indexes.

```json
// 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."
  },
  "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": { ... }
},
```
Activity: Design a data model

• Imagine you are designing a data model for a restaurant review site.
  • Each user has a profile, may submit reviews (linked to the user) for a restaurant, and may upvote, downvote, and comment on reviews (linked to the reviewer and restaurant).

• How would you structure this data?
Storing Data: Set

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

“On the active firebase database”
Must be initialized first (coming soon….)

“Get the users/[userID] node”
Arbitrary nodes in the tree can be addressed by their path.

“Set value”
Sets the value to specified JSON.
Storing Data: Push

```javascript
var key = firebase.database().ref().child('posts').push(
  { author: username, uid: uid, body: body, title: title });
```

- What about storing collections?
  - Use push to create key automatically
  - All data MUST have a key so it can be uniquely referenced
    - Arrays given index keys
    - Should never have multiple clients synchronizing an array
      - Local indexes could get of sync with remote keys
      - Instead, use JSON object with number as key
Storing Data: Delete

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

Removes the ‘posts’ subtree.

- Can delete a subtree by setting value to null or by calling remove on ref
  - If you want to store null, first need to convert value to something else (e.g., 0, '')
Listening to data changes

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

“When values changes, invoke function”

Specify a subtree by creating a reference to a path. Listen to one or more events by using on(eventName, eventHandlerFunction(snapshot))

• Read data by *listening* to changes to specific subtrees
• Events will be generated for initial values and then for each subsequent update
Data Update Events

• Types of events
  • value: entire contents of a path
  • child_added
  • child_changed
  • child_removed

• Can listen to events on any part of subtree
  • Could have subtrees that correspond to different collections of data
  • Should always listen to lowest subtree of interest to minimize extraneous communication

• Can read data exactly one time (and not get updates) using once
Ordering data

• Data is by, default, ordered by key in ascending order
  • e.g., numeric index keys are ordered from 0…n
  • e.g., alphanumeric keys are ordered in alphanumeric order

• Can get only first (or last) n elements
  • e.g., get n most recent news items

```javascript
var recentPostsRef = firebase.database().ref('posts').limitToLast(100);
recentPostsRef.once('value', function(snapshot) {
    displayPost(snapshot.val());
});
```
Activity: Persist restaurant data in Firebase

- Assume you have some endpoints to gather submitted data from users for previous scenario.
- Using data submitted through these endpoints, store to Firebase.
- Offer endpoints for retrieving this data back from Firebase.
Reading for next time

• Intro to CORS: https://www.moesif.com/blog/technical/cors/Authoritative-Guide-to-CORS-Cross-Origin-Resource-Sharing-for-REST-APIs/