Firebase Application Building

The Coding Bootcamp

Class Objectives

- Provide a "real-world" application development scenario that utilizes HTML, CSS, Databases, and Data Manipulation
- Introduce the concept of creating lists and children using Firebase
  - New Methods!
- Introduce the MomentJS library for date-time manipulation
- Complete Timers and APIs Checkpoint.

Firebase Review

```
database = firebase.database()

.set({})

.ref()

.on("value", function(){})
```

Firebase

Firebase Review

`database = firebase.database()` references the database

`.set({})` saves the data

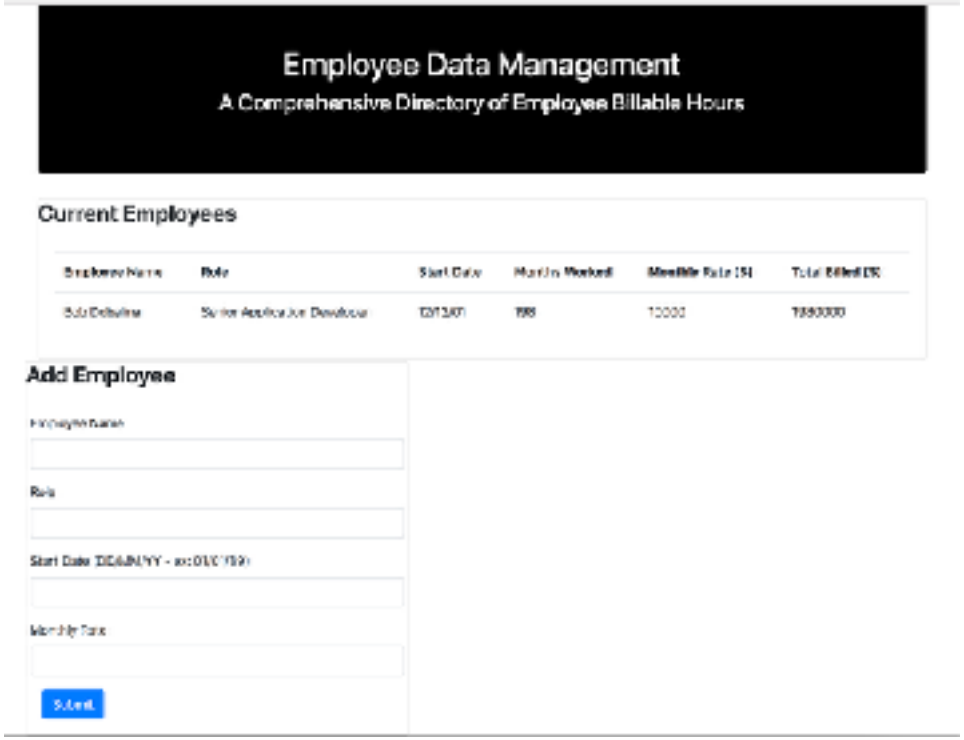
`.ref()` specifies where the data will be saved

`.on("value", function(){})` which creates an "on-change" event so that the moment the page first loads or the moment the database changes, the impact is reflected immediately.

Database vs. Local Storage

- "Which fields MUST BE in the database?"

- "Which fields can we avoid sending to the database in favor of calculating locally?



Main Application Design Phase (20 Mins)

For the next 20 minutes, focus all your efforts on creating the application layout for your site.

This phase involves both:

- **Creating the overall HTML/CSS/Bootstrap Layout**

- **Creating the initial `.on("click")` event that will dynamically trigger new HTML rows to be generated.**

This phase DOES NOT involve sending or receiving data to Firebase.

If you finish early:

- Continue refining the design! Take things to the next level. Make this application portfolio-grade!

- Begin reading about `push({})` and `.on("child_added")` in the Firebase documentation.

set vs. push!

```
112     // Code for "Setting values in the database"
113     database.ref().set({
114         name: name,
115         email: email,
116         age: age,
117         comment: comment
118     })
```

.set({ }) is limited to replaceing data

```
113     // Code for handling the push
114     database.ref().push({
115         name: name,
116         email: email,
117         age: age,
118         comment: comment
119     })
```

Changing to .push({ }) allows us to save multiple records

Main Application - Push Phase (25 mins)

- **Instructions**
  - Setup project/firebase database
  - Using your newfound knowledge of the `.push({})` method, create the code necessary to push employee data into the database upon clicking the `submit` button on your webpage.
  - NOTE: **Don't worry about getting the data to display in the HTML just yet. Just focus on getting data pushed to the database.**
  - If you finish early, begin reading about `.on("child_added")` in the Firebase documentation and/or the MomentJS library.

Timestamp!

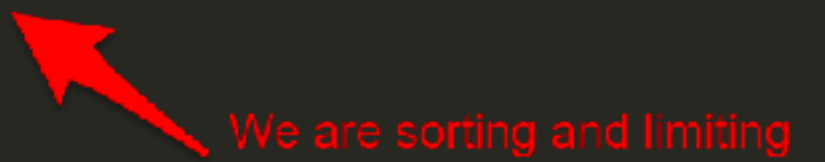
Special property called:
`firebase.database.ServerValue.TIMESTAMP`
This will add a timestamp in unix format in our database.

```
129     // Code for the push
130     dataRef.ref().push({
131         name: name,
132         email: email,
133         age: age,
134         comment: comment,
135         dateAdded: firebase.database.ServerValue.TIMESTAMP
136     })
```

Sorting and Limiting

`.on("child_added", function(){})`

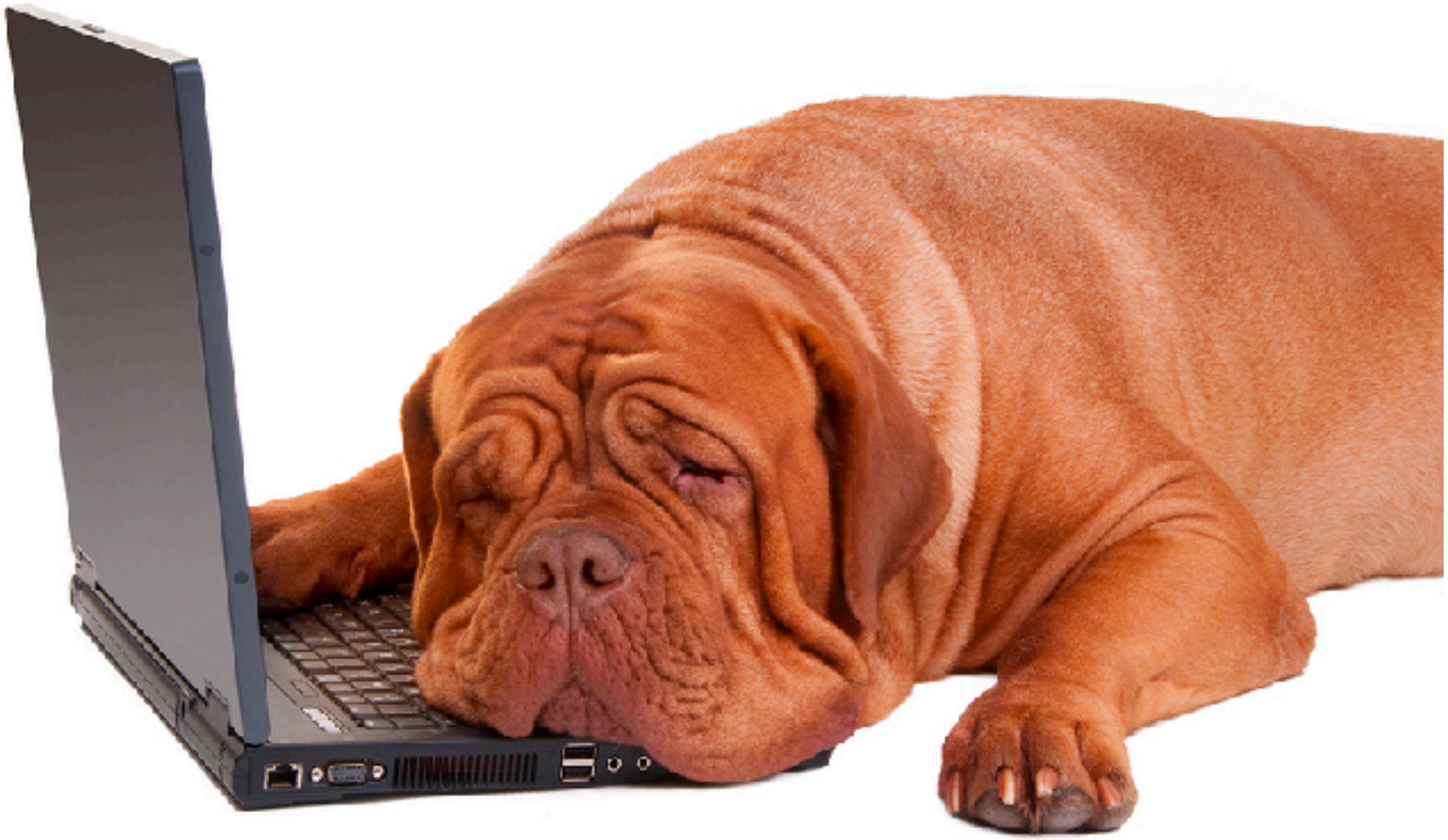
```
161  dataRef.ref().orderByChild("dateAdded").limitToLast(1).on("child_added", function(snapshot){
162      // Change the HTML to reflect
163      $("#namedisplay").html(snapshot.val().name);
164      $("#emaildisplay").html(snapshot.val().email);
165      $("#agedisplay").html(snapshot.val().age);
166      $("#commentdisplay").html(snapshot.val().comment);
167  })
```



Main Application - Child_Added Phase (25 mins) (High)

- **Instructions**
  - Using your newfound knowledge of the .on("child_added") method, begin to retrieve your employee data from the database and populating the records into your table.
  - Note: Don't worry about calculating Months Worked or the Total Billed just yet. Just focus on retrieving the data that is already in the database.
  - If you finish easily, continue refining the aesthetics of your website, consider incorporating "update" or "delete" employee buttons, or begin reading up on the MomentJS library.

Lunch Break! (35mins)



Students Do: MomentJS Activity (25 mins)

- **Instructions**
  - Complete each of the activities listed in the comments.
  - Note: You don't need to go in order.
  - Note: **Don't let the simple example fool you. Working with a new library can be tough. Be prepared to get frustrated. Stick with it!**

Main Application - Datetime Manipulation

- Utilizing your newfound datetime knowledge to calculate the number of months worked and subsequently the total amount billed.
- Don’t forget to add the momentJS library!

Everyone Do: Traintime Prediction (Math) (10 mins) (High)

- With your group write out the steps you would use "mathematically" to determine the answer to the following situation:

- Assuming:

  a. The first train of the day comes in at 3:00 AM.
  b. The train runs every 17 minutes
  c. The current time is 7:12 PM.
  d. There have been no delays and will be no delays.

- Question:

  a. How many minutes away is the next train?

Traintime Prediction

- Using the comments in the code as a guide, determine the mathematical formula for calculating train times.

- Then explain to one another how the code works (line by line) and how it could be used in relationship to the homework assignment.

Next two weeks will be Project Weeks

- Must uses at least two APIs
- Must use AJAX to pull data
- Must utilize at least one new library or technology that we haven't discussed
- Must have a polished frontend / UI
- Must meet good quality coding standards (indentation, scoping, naming)
- Must NOT use alerts, confirms, or prompts (look into modals!)
- Must have some sort of repeating element (table, columns, etc)
- Must use Bootstrap or Alternative CSS Framework
- Must be Deployed (GitHub Pages or Firebase)
- Must have User Input Validation

Timers and APIs Checkpoint

Timers and APIs Checkpoint.

https://www.switchboard.tech/

Questions