GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-9-resources` repo to your computer
2. Open the `15-crud-firebase > starter-code` folder in your code editor

INTRO TO CRUD AND FIREBASE

LEARNING OBJECTIVES

At the end of this class, you will be able to

- Explain what CRUD is.
- Explain the HTTP methods associated with CRUD.
- Implement Firebase in an application.
- Build a full-stack app with CRUD functionality.

AGENDA

- CRUD
- Firebase intro and setup
- Create
- Read
- Update
- Delete

WEEKLY OVERVIEW

WEEK 9	CRUD & Firebase / Deploying your app
WEEK 10	React / Final project lab
WEEK 11	Final project presentations

HOMEWORK REVIEW

ACTIVITY

EXERCISE

KEY OBJECTIVE

- Review Feedr project and show off your work

TYPE OF EXERCISE

- Groups of 3-4

TIMING

10 min

1. Open Feedr sites on laptops and display them proudly!
2. Give feedback to your peers: "I like" and "I wish/wonder”
3. Share a challenge you ran into in your project and discuss how other group members may have worked with it.
4. Did you incorporate template literals in your project? Show your group how you did it!

EXIT TICKET QUESTIONS

1. Still a little confused about the scope of closures

BUILDING BLOCKS OF CLOSURES

1. **nested functions**
2. **scope**

   inner function has access to outer function's variables

3. **`return` statements**

   outer function returns reference to inner function

CLOSURES

- A **closure** is an inner function that has access to the outer (enclosing) function's variables.

```
function getTemp() {
  let temp = 75;
  let tempAccess = function() {
    console.log(temp);
  }
  return tempAccess;
}
```

the `tempAccess()` function is a closure

outer function `getTemp()` returns a reference to the inner function `tempAccess()`

CLOSURES

- A **closure** is an inner function that has access to the outer (enclosing) function's variables.
- You create a closure by nesting a function inside another function.

ACTIVITY

EXERCISE

KEY OBJECTIVE

- Review closures

TYPE OF EXERCISE

- Groups of 2-3

TIMING

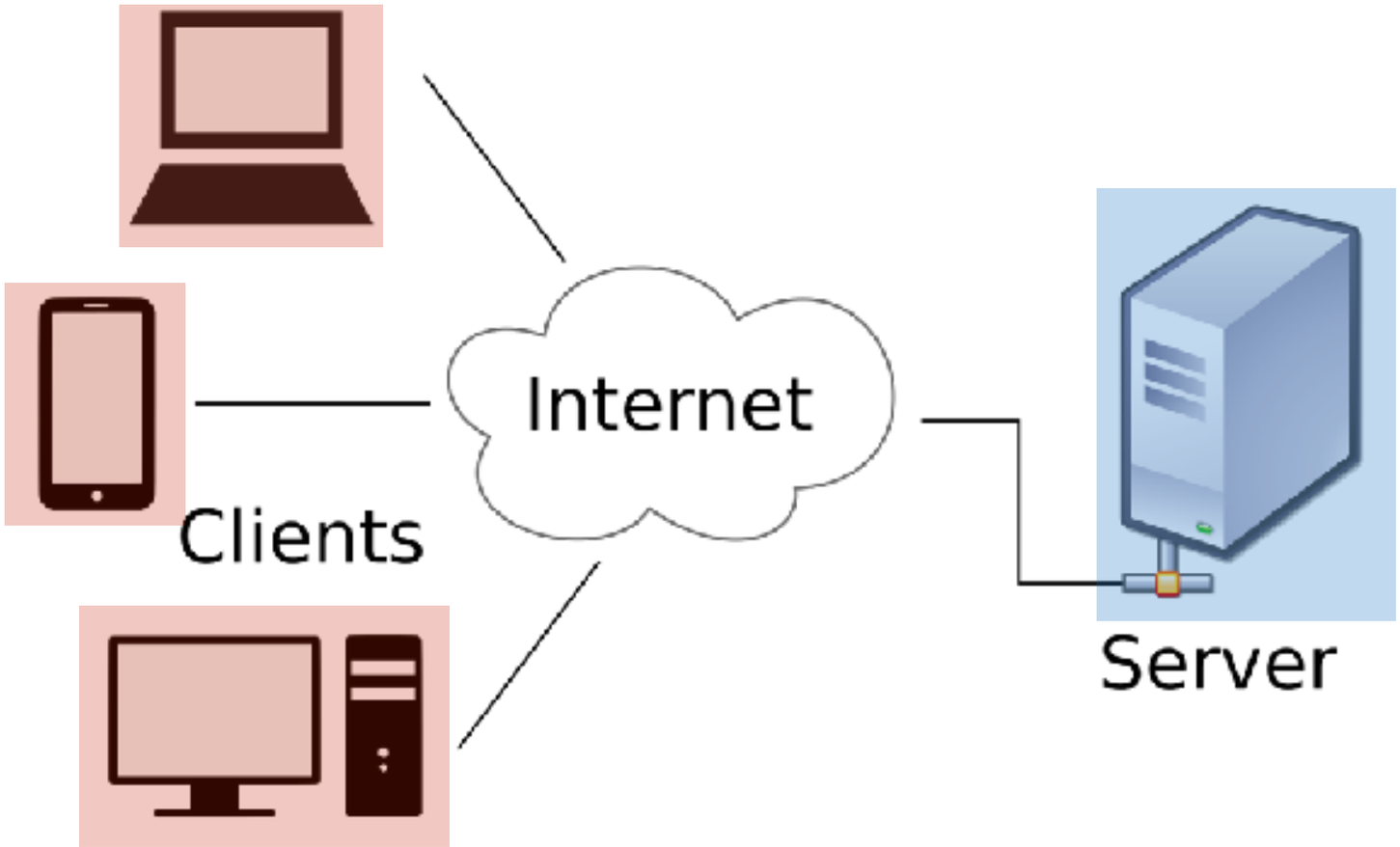
5 min

1. In the `closures-example` folder, open the `app.js` file. This app uses a closure to return and create functions that calculate local sales tax based on a specified sales tax rate.
2. With your group members, identify the 3 building blocks of closures in the code, and explain how the 3 building blocks work together to create this closure.
3. BONUS: Add a statement to the app.js file that uses the `createTaxCalculator` function to create a new function called `calcSacramentoTax` with a tax rate of 0.0825. Add a second statement to call this new function to calculate sales tax on a purchase of 100 dollars. Save your work and check the result in the browser console.

What are some apps that allow you to create, read, update, and delete data?

Back-end review

Front end
- HTML
- CSS
- JS



Back end
- JS
- Python
- Ruby
- PHP
- …

CRUD

- Create
- Read
- Update
- Delete

CRUD and HTTP

CRUD action	HTTP verb
Create	POST
Read	GET
Update	PATCH/PUT
Delete	DELETE

EXERCISE — API METHODS

EXERCISE

KEY OBJECTIVE

- Identify API methods that let you implement CRUD functionality using a popular web service

TYPE OF EXERCISE

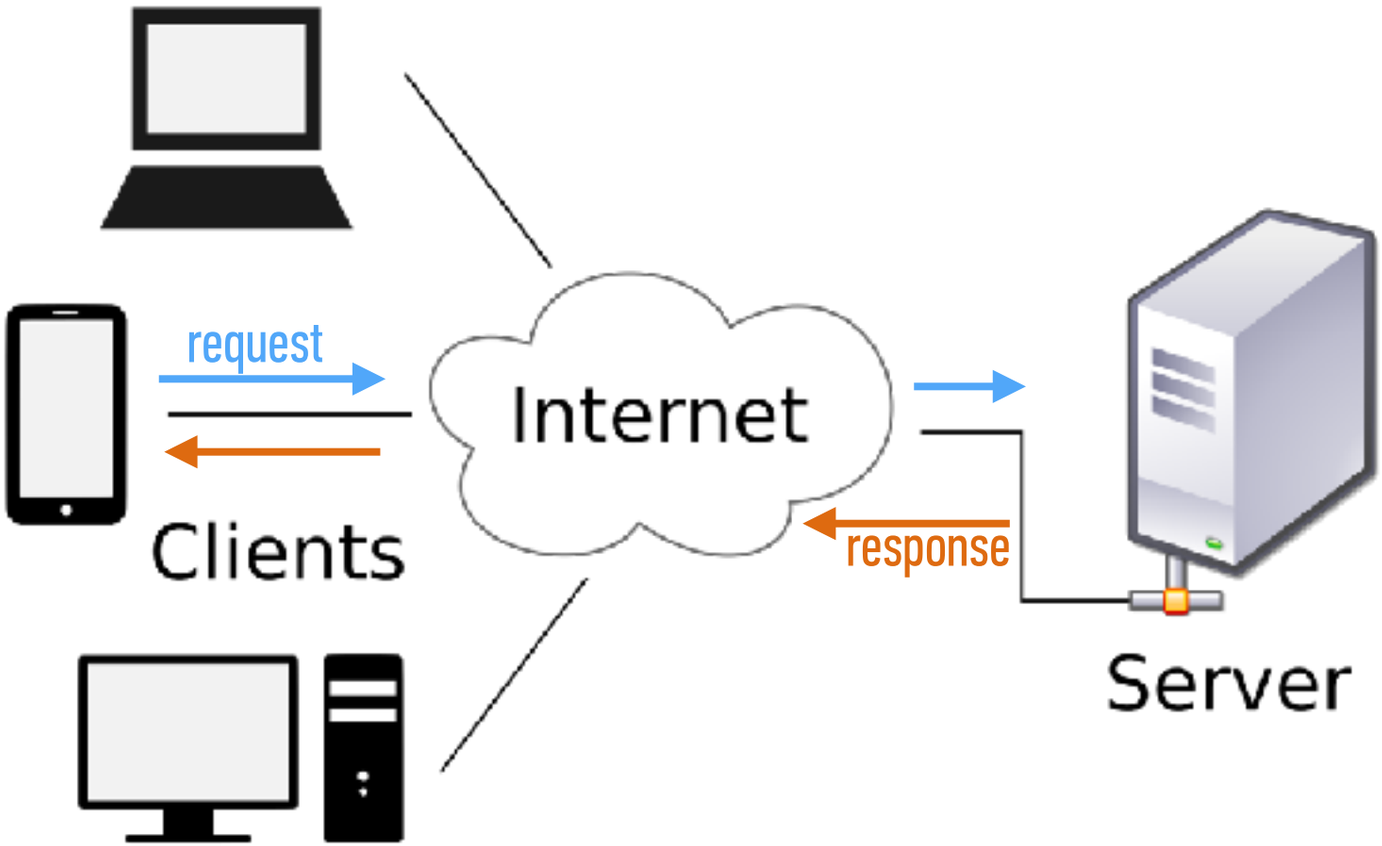
- Groups of 3

TIMING

5 min

1. Research your assigned API to see what HTTP methods a developer must use to perform at least one instance of create, read, update and delete. (If your API doesn't fully support CRUD, note any limitations.)
2. Further, define what exactly is being created, read, updated or deleted. For example, for Facebook what HTTP method on what endpoint must you ping in order to create a post in a feed?

THE CLIENT-SERVER MODEL WITH CRUD



Stores HTML/CSS/JS code

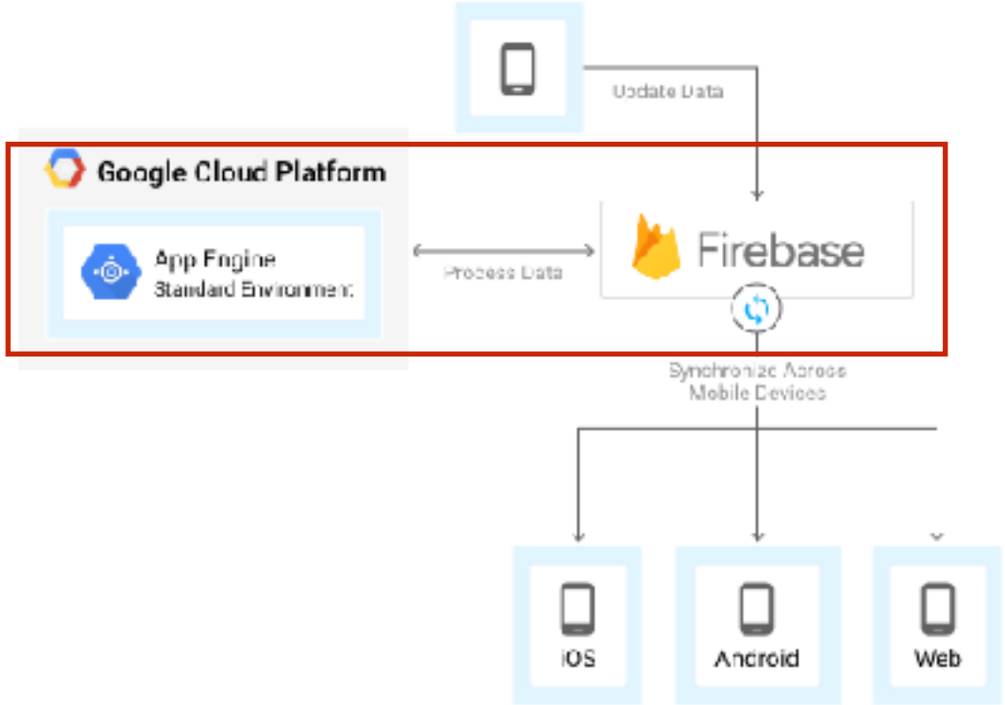
- Accepts HTTP requests
- Generates HTTP responses

Stores database

- Provides create access
- Provides read access
- Provides update access
- Provides delete access

FIREBASE

Back end



LAB — PLAN A CRUD APP

LAB

KEY OBJECTIVE

- Plan a full-stack app with full CRUD functionality

TYPE OF EXERCISE

- Solo or in pairs

TIMING

10 min

1. Come up with an idea for an app that implements CRUD. You'll build your app this week in class (this is not your final project). Your app must be able to Create, Read, Update and Delete data.
2. Build out your HTML, CSS, and JS files.
3. Add code generated from your Firebase project to your HTML and JS files.

CRUD and HTTP

CRUD action	HTTP verb	Firebase method
Create	`POST`	`push()`
Read	`GET`	`ref()`
Update	`PATCH`	`update()`
	`PUT`	`set()`
Delete	`DELETE`	`remove()`

LAB — IMPLEMENT CREATE FUNCTIONALITY

LAB

KEY OBJECTIVE

- Build the Create functionality of a full-stack app

TYPE OF EXERCISE

- Solo or in pairs

TIMING

20 min

1. Create a form
2. Get user input
3. Create a section in your database for your data
4. Save your data to the database
5. Change security rules to allow access without authentication
6. View your data in the Firebase dashboard

HOW TO ASSOCIATE LIST ITEMS WITH DATABASE ENTRIES?

```
<ul class="message-board">
  <li>Mulberries, 254 Cesar Chavez St.</li>
  <li>Oranges, 1 Main St.</li>
  <li>Fuschia berries, 26 Isabella Ln.</li>
</ul>
```

EACH RECORD SAVED IN FIREBASE HAS A UNIQUE ID

```
public-produce
  messages
    -L2RNI8NhVXD1kHDjlvx
      message: "Mulberries, 254 Cesar Chavez St."
      votes: 0
    -L2RRqJ9GmnCLQi-EvBr
      message: "Oranges, 1 Main St."
      votes: 0
    -L2RRxDPhMyh3tElPxp-
      message: "Fuschia berries, 26 Isabella Ln."
      votes: 0
```

HTML `data` ATTRIBUTE

Allows us to associate metadata with DOM elements

Attribute name is `data-` plus any string

```
<article
  id="electriccars"
  data-columns="3"
  data-index-number="12314"
  data-parent="cars">
...
</article>
```

DOM WITH CUSTOM `data-id` ATTRIBUTES

```
<ul class="message-board">
  <li data-id="-L2RNI8NhVXD1kHDjIvx">Mulberries, 254 Cesar Chavez St.</li>
  <li data-id="-L2RRqJ9GmnCLQi-EvBr">Oranges, 1 Main St.</li>
  <li data-id="-L2RRxDPhMyh3tEIPxp-">Fuschia berries, 26 Isabella Ln.</li>
</ul>
```

Custom attribute named `data-id`

Each attribute value is a database record ID

LAB — IMPLEMENT READ FUNCTIONALITY

LAB

KEY OBJECTIVE

- Build the Read functionality of a full-stack app

TYPE OF EXERCISE

- Solo or in pairs

TIMING

20 min

1. Examine the API documentation at https://firebase.google.com/docs/reference/js/firebase.database.Reference
2. Listen for changes (use `.ref()` and `.on()`)
   - https://firebase.google.com/docs/reference/js/firebase.database.Reference#ref
   - https://firebase.google.com/docs/reference/js/firebase.database.Reference#on
3. Add returned data to your front end using DOM manipulation

LAB — IMPLEMENT UPDATE FUNCTIONALITY

LAB

KEY OBJECTIVE

- Build the Update functionality of a full-stack app

TYPE OF EXERCISE

- Solo or in pairs

TIMING

20 min

1. Examine the API documentation at
   - https://firebase.google.com/docs/reference/js/firebase.database.Reference#update
   - https://firebase.google.com/docs/reference/js/firebase.database.Reference#set
2. Create a function to make updates to the database
3. Add calls to your new function when data is changed in your app

LAB — IMPLEMENT DELETE FUNCTIONALITY

LAB

KEY OBJECTIVE

- Build the Delete functionality of a full-stack app

TYPE OF EXERCISE

- Solo or in pairs

TIMING

10 min

1. Examine the API documentation at https://firebase.google.com/docs/reference/js/firebase.database.Reference#remove
2. Create a function to delete records from the database
3. Add calls to your new function when data is deleted in your app

Exit Tickets!

(Class #15)

LEARNING OBJECTIVES - REVIEW

- Explain what CRUD is.
- Explain the HTTP methods associated with CRUD.
- Implement Firebase in an application.
- Build a full-stack app with CRUD functionality.

NEXT CLASS PREVIEW

Deploying your app

- Understand what hosting is.
- Identify a program's needs in terms of host providers.
- Ensure backward compatibility by using Babel to transpile code.
- Deploy to a web host.

Q&A