

# JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

# HELLO!

1. Pull changes from the `svodnik/JS-SF-13-resources` repo to your computer
2. Open the `15-crud-firebase` folder in your code editor

---

**JAVASCRIPT DEVELOPMENT**

---

# **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

---

## INTRO TO CRUD AND FIREBASE

---

# WEEKLY OVERVIEW

### WEEK 9

Closures & this / CRUD & Firebase

### WEEK 10

Deploying your app / Final project lab

### WEEK 11

React / Final Project Presentations

## **EXIT TICKET QUESTIONS**

1. I had this note but left it incomplete. how does it finish? 'why use 'this instead of variable name' => if you're creating an object instead of a ?????'
2. Should I be trying to build everything in modules? When is it ok to have stuff in global scope?
3. I still need practice predicting what 'this' will be

# CONTEXT RULES

situation	what <code>this</code> maps to
<b>method invocation</b>	the object that owns the method
<b>constructor function</b>	the newly created object
<b>event handler</b>	the element that the event was fired from
<b>function invocation</b>	the global object (window)
<b>function invocation (strict mode)</b>	undefined
<b>arrow function</b>	the context of the caller



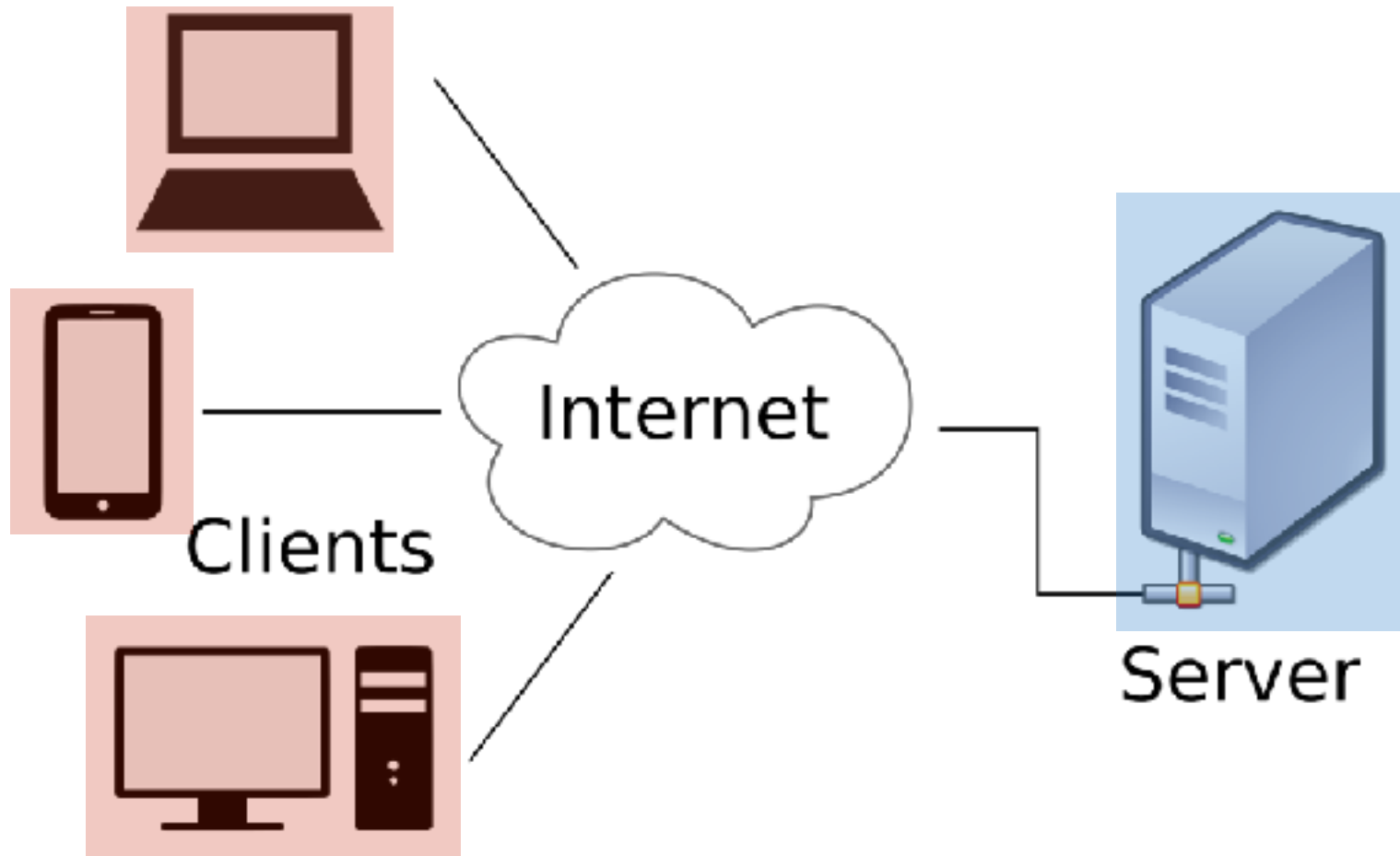
# CRUD

**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

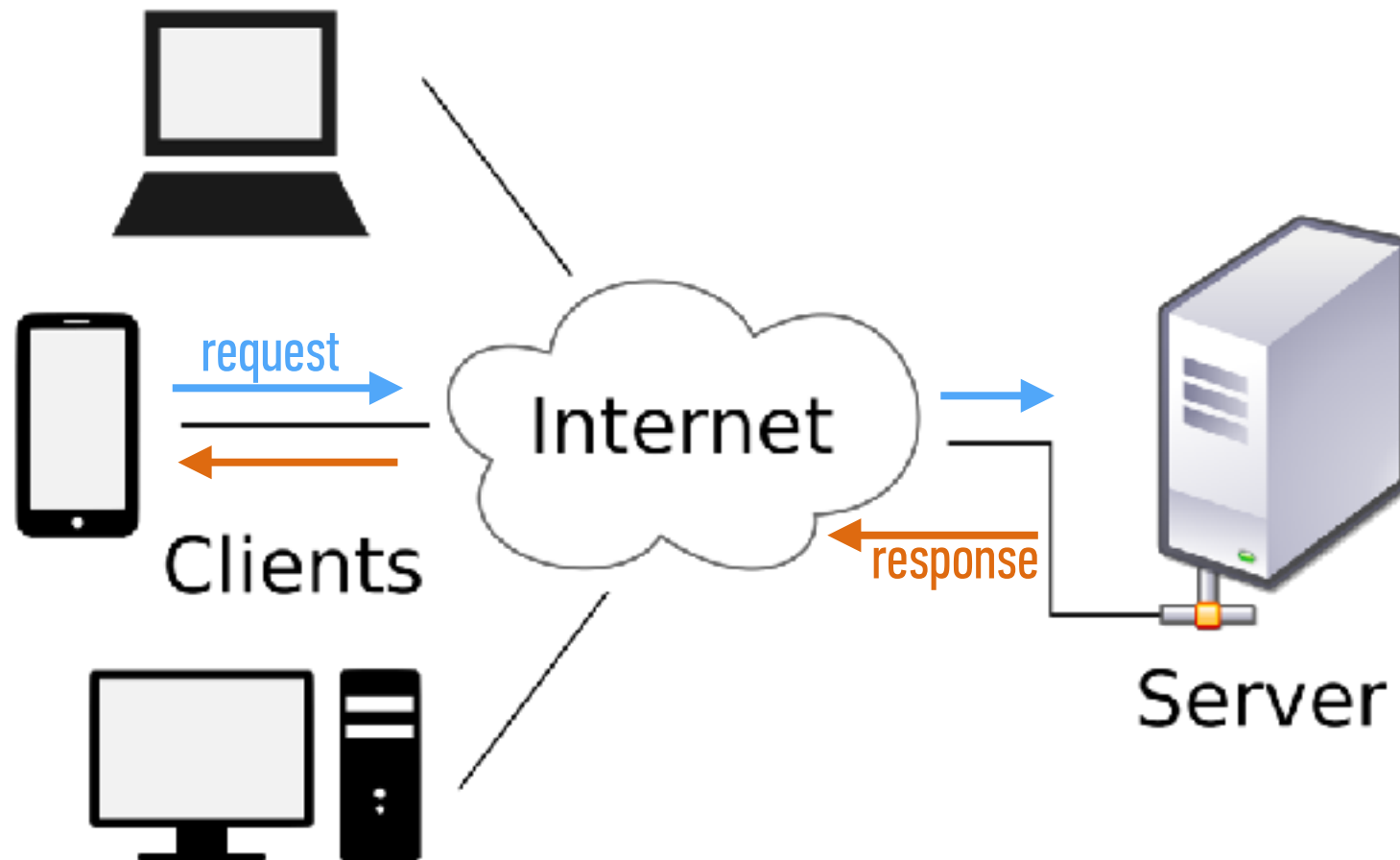
### TIMING

---

*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



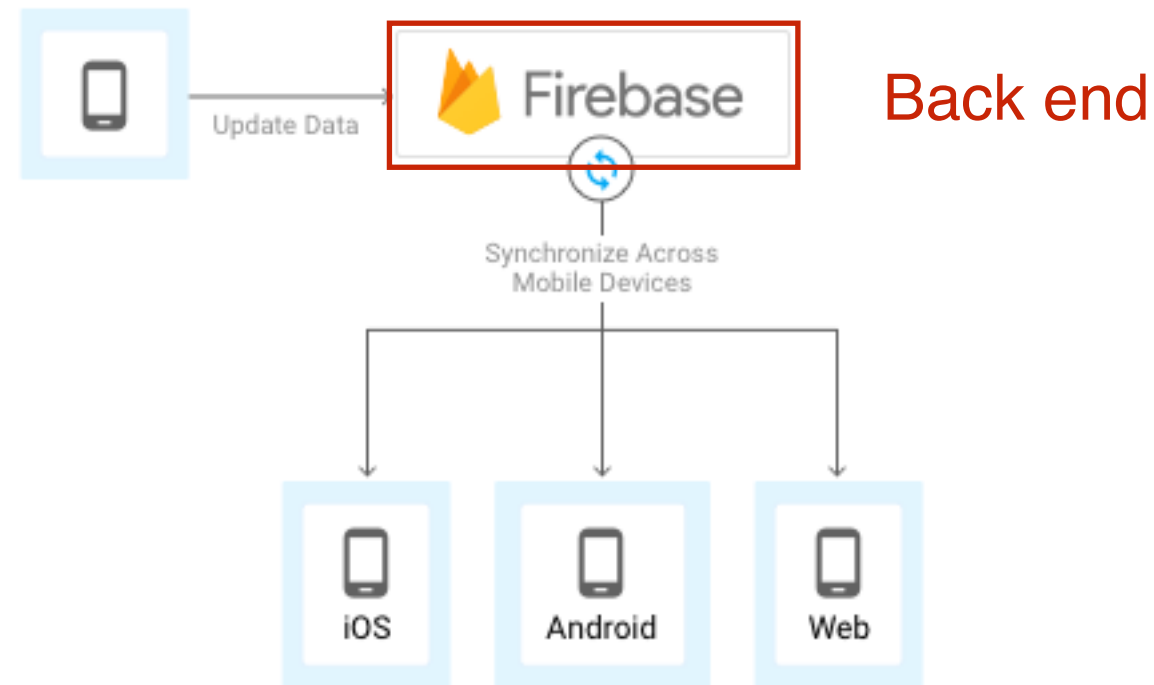
**Stores HTML/CSS/JS code**

- Accepts HTTP requests
- Generates HTTP responses

**Stores database**









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

# FIREBASE



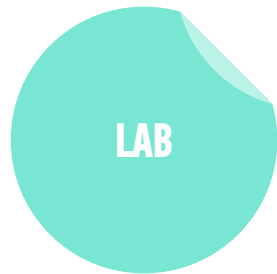


# ALTERNATIVE “SERVERLESS” SERVICES

 <b>Google Firebase</b> <b>Relevant Capabilities</b> <ul style="list-style-type: none"><li>Auth</li><li>Realtime Database</li><li>Media Storage</li><li>Cloud Functions</li></ul> <b>Quick Overview</b> <p>Google Firebase is very powerful while being very easy to use. For example, you can run cloud functions, but you don't even need to for most data storage and retrieval or auth. It might be expensive to scale on though.</p>	 <b>Google Cloud Platform</b> <b>Relevant Capabilities</b> <ul style="list-style-type: none"><li>Everything</li></ul> <b>Quick Overview</b> <p>More of a major infrastructure provider in vein of Amazon Web Services than a toolkit for building out an app like Firebase is.</p>	 <b>Amazon Web Services</b> <b>Relevant Capabilities</b> <ul style="list-style-type: none"><li>Everything</li></ul> <b>Quick Overview</b> <p>Lambda, API Gateway, S3, and Cognito (auth) are probably the most relevant things to front-end developers. <a href="#">AWSync</a> is a hit like Firebase. There are frameworks that help you deploy to Lambda, like <a href="#">Craffie</a> and <a href="#">Functional Fleet</a>.</p>	 <b>Microsoft Azure</b> <b>Relevant Capabilities</b> <ul style="list-style-type: none"><li>Everything</li></ul> <b>Quick Overview</b> <p>A major infrastructure provider with solutions for about just everything, and generally considered the cheapest. For working with cloud functions, there is an online editor, but it also allows Git-hub sync and <a href="#">integrates directly with VS Code</a>. Data storage is through Cosmos DB.</p>	 <b>StdLib</b> <b>Relevant Capabilities</b> <ul style="list-style-type: none"><li>Cloud Functions</li></ul> <b>Quick Overview</b> <p>StdLib is based on Function as a Service ("server-less") architecture, popularized by AWS Lambda. You can use StdLib to build modular, scalable APIs for yourself and other developers in minutes without having to manage servers, gateways, domains, write documentation, or build SDKs. They also offer <a href="#">Codeboxz</a>, and online code editor for working with the APIs.</p>	 <b>Webtask</b> <b>Relevant Capabilities</b> <ul style="list-style-type: none"><li>Cloud Functions</li><li>Basic JSON data store</li></ul> <b>Quick Overview</b> <p>An in-browser editor for creating and testing cloud functions. Seems like the nicest experience for this particular job. It's kinda of an elaborate demonstration of <a href="#">Auth0 Extend</a>, which is essentially a way to take Webtask and put it in your own app.</p>	 <b>IBM Cloud Functions</b> <b>Relevant Capabilities</b> <ul style="list-style-type: none"><li>Cloud Functions</li></ul> <b>Quick Overview</b> <p>Based on <a href="#">Apache OpenWhisk</a>.</p>	 <b>Backendless</b> <b>Relevant Capabilities</b> <ul style="list-style-type: none"><li>Realtime Database</li><li>Auth</li></ul> <b>Quick Overview</b> <p>All-in-one kind of service similar to Firebase, including the realtime database. Has a PHP version you can host yourself if you're, ya know, into running servers.</p>
--	---	--	--	--	--	---	--

# LAB — PLAN A CRUD APP

---



## 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	Firestore method
Create	POST	push()
Read	GET	ref()
Update	PATCH	update()
	PUT	set()
Delete	DELETE	remove()

# LAB — IMPLEMENT CREATE FUNCTIONALITY

---



## KEY OBJECTIVE

---

- Build the Create functionality of a full-stack app

## TYPE OF EXERCISE

---

- Solo or in pairs

## TIMING

---

*20 min*

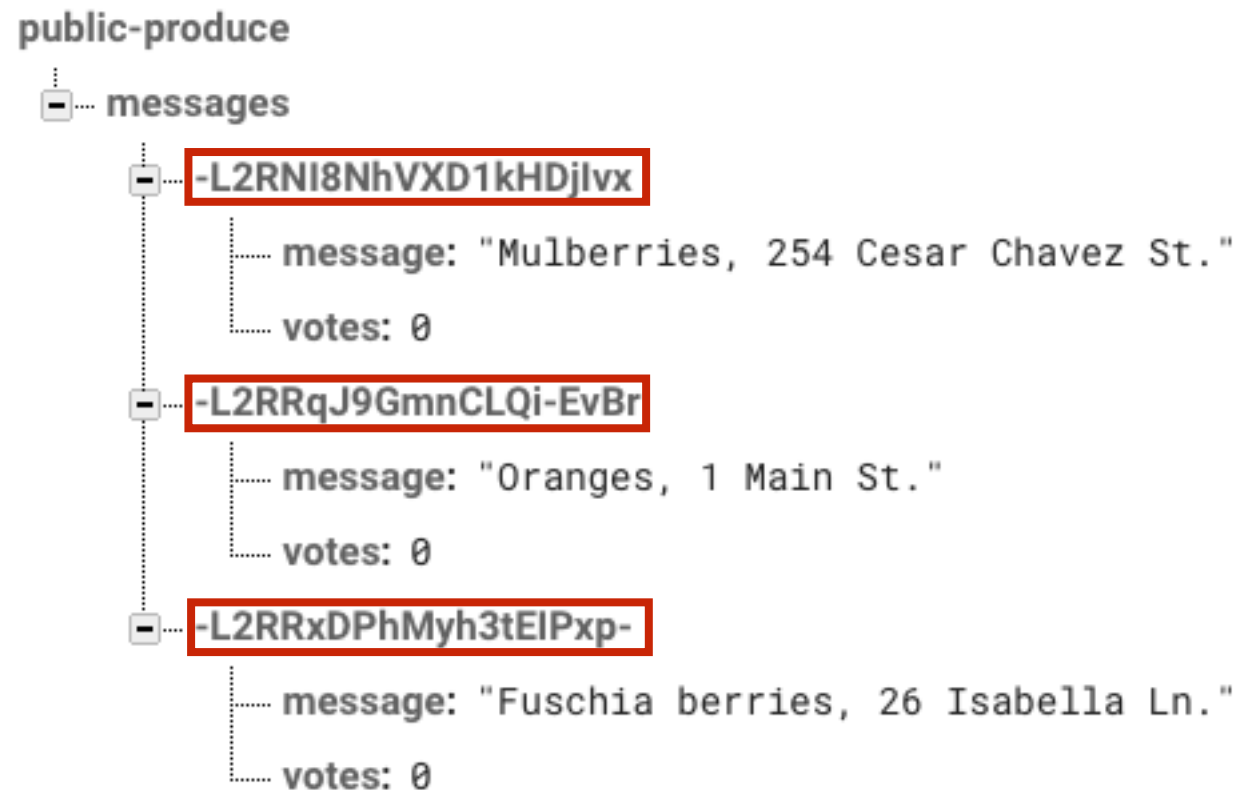
1. Create a form
2. Get user input
3. Create a section in your database for your data
4. Save your data to the database
5. Change security rules to allow access without authentication
6. View your data in the Firebase dashboard

# ASSOCIATING DOM ELEMENTS WITH DATABASE RECORDS

# 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



# HTML data ATTRIBUTE

Allows us to associate metadata with DOM elements

Attribute name is  
data- plus any string



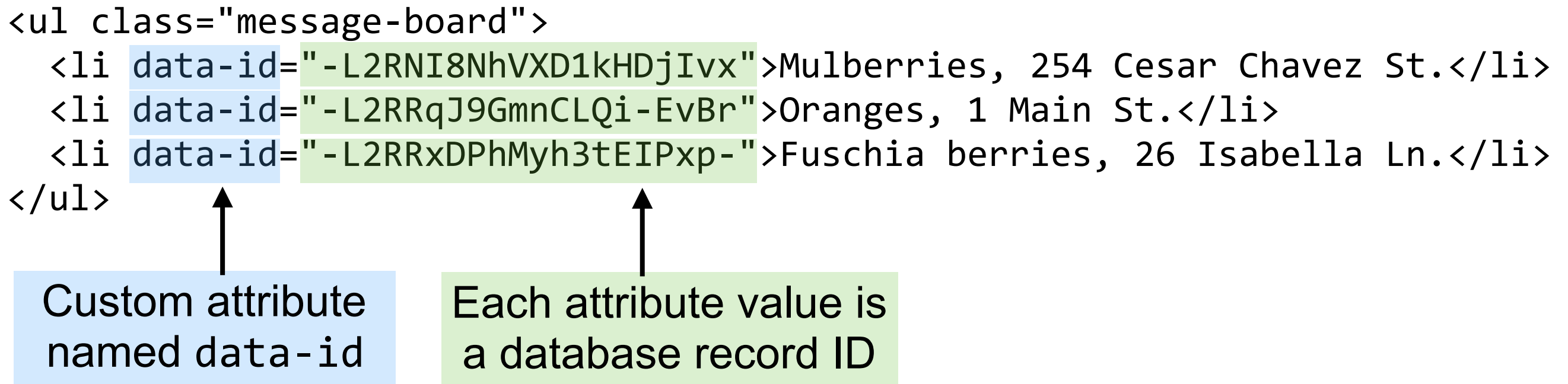
```
<article  
  id="electriccars"  
  data-columns="3"  
  data-index-number="12314"  
  data-parent="cars">  
  ...  
</article>
```



# 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>
```

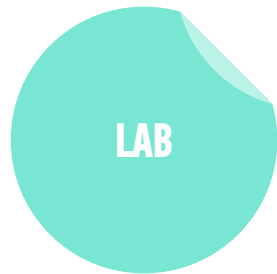
Custom attribute  
named data-id



Each attribute value is  
a database record ID

# LAB — IMPLEMENT READ FUNCTIONALITY

---



## 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

---



## 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

---



## 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