GENERAL ASSEMBLY

CORE DATA

Zeke Abuhoff

Lead iOS Instructor, General Assembly

LEARNING OBJECTIVES

+ Explain the utility of a relational database
+ Create a data model for core data
+ Save data via core data
+ Load data via core data

DATABASES

Firebase stores JSON.

How, in JSON, would we describe a dog?

Include the dog's name, age, breed and friends (other dogs).

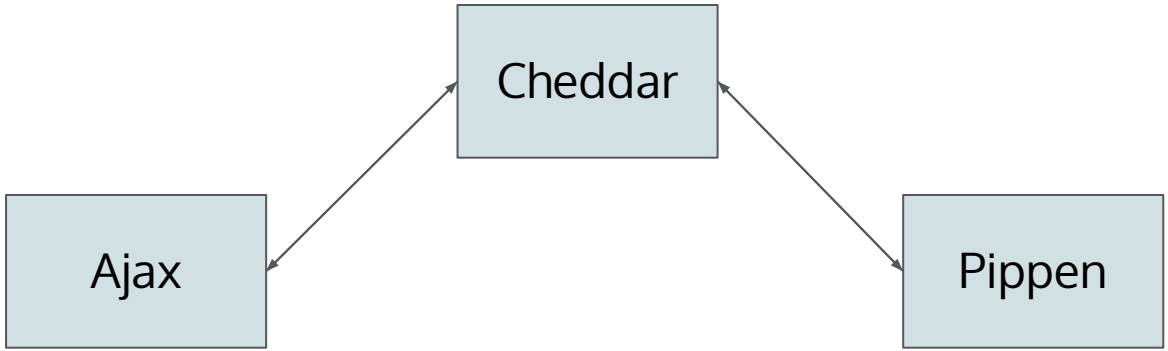
DATABASES

JSON is good at recording copies of information, like a struct.
But it doesn't model references, like a class does.

So, while recording a dog's name is easy enough on Firebase (add the key "name" with the value "Zero"), recording the dog's friends is trickier.

We need a database that can describe the relationships between pieces of data.

RELATIONAL DATABASES



A **relational database** is a database that stores not only simple types (like strings and integers) but also relationships between objects.

CORE DATA

Core Data is a first-party iOS framework that persists data on the device in a relational database.

You can add Core Data functionality to an app, but to save yourself time, it helps to start a project with the “Core Data” box checked.

DATA MODEL

In the file with extension ".datamodeld", we can configure our data model.

We'll create entities that have attributes and relationships.

Entities
These are types of objects.

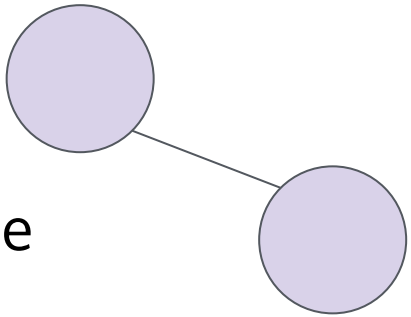
Attributes
These are basic properties of entities.

Relationships
These are connections between entities.

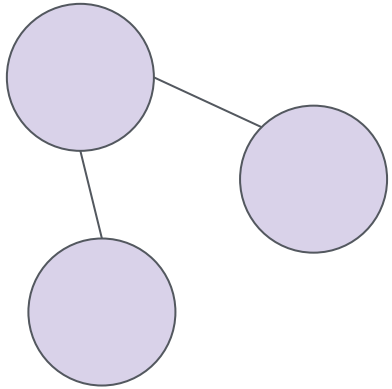
DATA MODEL

Relationships can be one-to-one or one-to-many. Also, each relationship should have an inverse that describes the relationship from the other entity's perspective.

one-to-one



one-to-many



DATA MODEL

Once you've configured your data model, select "Create NSManagedObject Subclass" from the "Editor" menu.

Select all your entities.

This action will create new files for your data model's classes.

Now your code can refer to entities as types!

CONTEXT

Core Data stages data in something called an NSManagedObjectContext.

When you start a project with Core Data enabled, Xcode automatically sets up a context in your app delegate (it's the viewContext property of the persistent container).

```
lazy var persistentContainer: NSPersistentContainer = { }

let context = persistentContainer.viewContext
```

SAVING

By creating an entity description, you can insert data into a context and save it.

```
func saveDog(name: String) {
     let context = appDelegate.persistentContainer.viewContext
    let entity = NSEntityDescription.entity(forEntityName: "Dog", in: context)!
     let dog = Dog(entity: entity, insertInto: context)
     dog.name = name
     do {
            try context.save()
        } catch {
            return
        }
}
```

LOADING

You can use your context and NSManagedObject subclass to retrieve data.

```
func loadDog(name: String) -> Dog? {
     let context = appDelegate.persistentContainer.viewContext
     let fetchRequest: NSFetchRequest<Driver> = Dog.fetchRequest()
     fetchRequest.predicate = NSPredicate(format: "name = %@", name)

     do {
            let results = try context.fetch(fetchRequest)
            return results[0]
        } catch {
            return nil
        }
}
```

CORE DATA

Practice:

1) Create a new project with Core Data enabled.
2) Design a data model for companies. A company can have many employees. A company can have many products. An owner can have many companies. All entities have names.
3) Create a function to save each entity.
4) Create a function to load each entity.