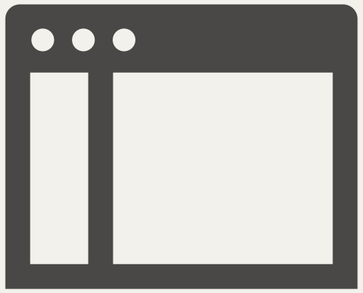
NO. 1

Intro

Forget Everything You Know About MVC

Ember is not Ruby on Rails. It's not a server. It follows MVC as used on desktop applications.



Major Objects in Ember

Router	Matches URLs to code objects
Controller	Handles actions and passes model data to views
Model	Stores, retrieves, and saves data
View	Handles clicks and taps
Template	Describes HTML output

The API has changed and may change again

Go through this video with the included code download from PeepCode. Then try the latest distribution of Ember. We plan to keep the code in sync but there may be minor changes.

Explicit/Implicit

Important parts of Ember will never appear by name in your application code. If Ember needs an object, it can often generate it for you, in memory.

It leans heavily on *Convention Over Configuration*. Learn its naming scheme. Understand its general flow of events. This will make it less frustrating and help you avoid writing unnecessary code.

Errors

Use `ember.js` in development for helpful error messages, not the minified production version.

Follow Ember's naming configurations. Start by using the defaults.

If something is very difficult to do, you're probably doing it wrong.

NO. 2

Starter Code

GOALS FOR STARTER CODE

- ✓ Understand basic files and directories
- ✓ Know where JavaScript and HTML are stored in the app
- ✓ Run the application
- ✓ Review the completed application

NO. 3

Basic App

GOALS FOR BASIC APP

- ✓ **Build the simplest possible Ember.js application**
- ✓ **Understand the** `outlet`
- ✓ **Render an** `index` **template**

```
<h1>
  {{ title }}
</h1>
```

Ember uses Handlebars for all templating

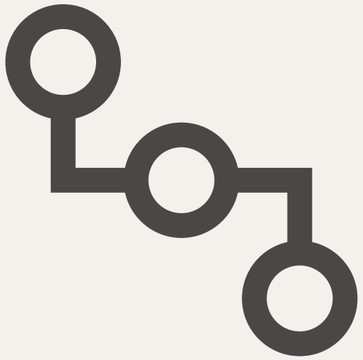
Variables in curly braces are populated with data and rendered as HTML.

NO. 4

Router

GOALS FOR ROUTER

- ✓ Display a list of tables in the restaurant
- ✓ Make a route
- ✓ Implement a handler
- ✓ Implement a controller
- ✓ Use a model
- ✓ Implement a template



Router

Runs code when URLs are visited.

Also loads data and assigns it to a controller.

EXAMPLE

When `/tables` is visited, Ember calls the `App.TablesRoute` object, which finds the list of tables in the restaurant and assigns it to the `App.TablesController`.

Naming Conventions

ROUTER	this.resource(‘tables’) App.TablesRoute
CONTROLLER	App.TablesController
MODEL	App.Table
VIEW	App.TablesView
TEMPLATE	tables

Controller

Delivers model data to views and templates.

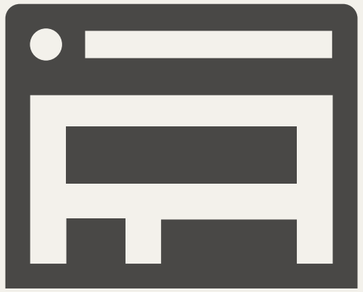
Array and Object controllers manage a `model` property. They act as a proxy to the model's attributes and methods, sending them along if asked.

NO. 5

Markup

GOALS FOR MARKUP

- ✓ Add markup for display
- ✓ Reorganize templates
- ✓ Use a partial template



Partial Template

The simplest way to render a template.

```
{{ partial “tableMenu” }}
```

Reuses the existing controller and all other context (including the controller’s model data).

Follows the Rails convention of prepending an underscore to the name of the template (on file or an inline template).

EXAMPLE

A `_tableMenu` rendered from a controller’s main view.

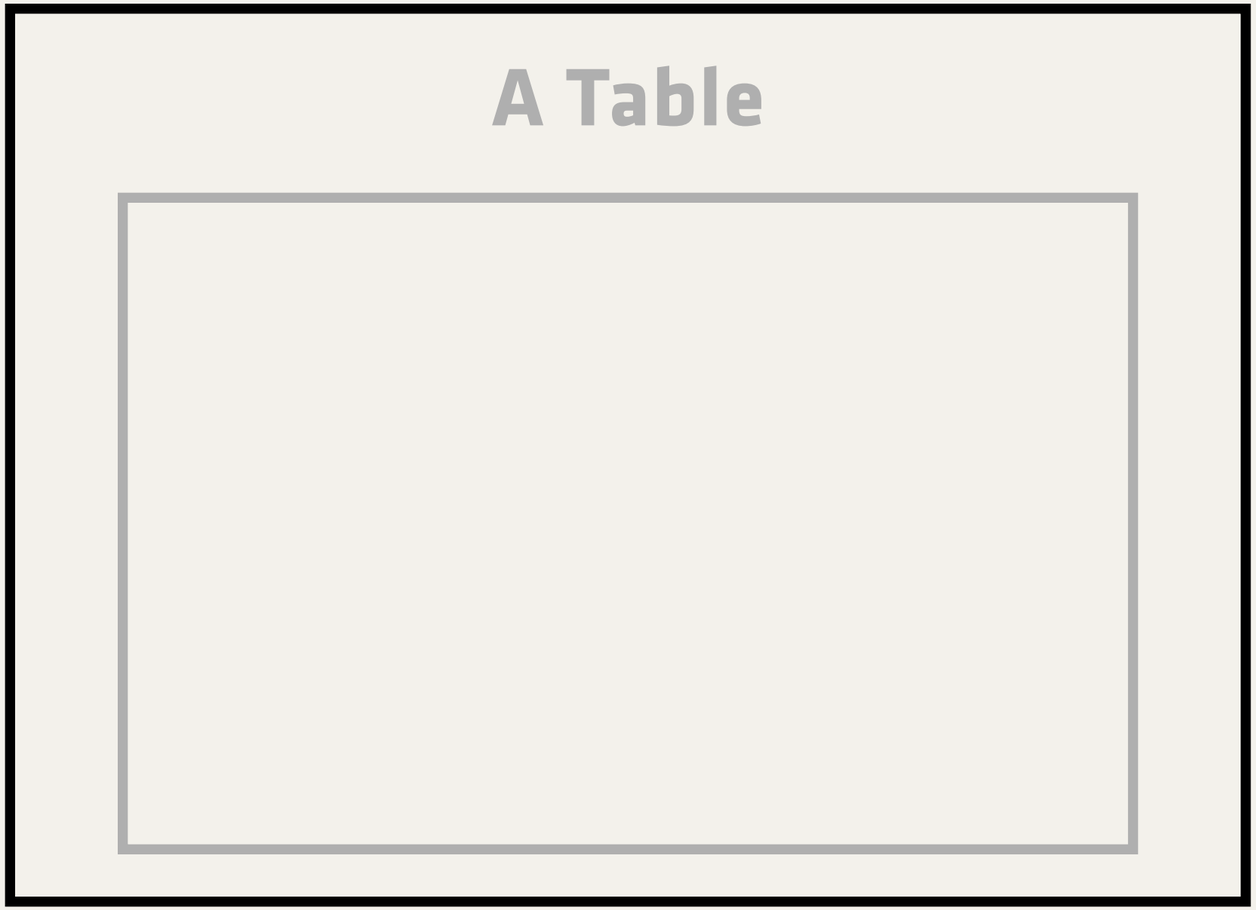
NO. 6

Table Resource

GOALS FOR TABLE RESOURCE

- ✓ Configure a nested table resource URL
- ✓ Link to details about a single table


Tables


A Table

"If your user interface is nested, your routes should be nested"

– Yehuda Katz

NO. 7

Table Controller & Data

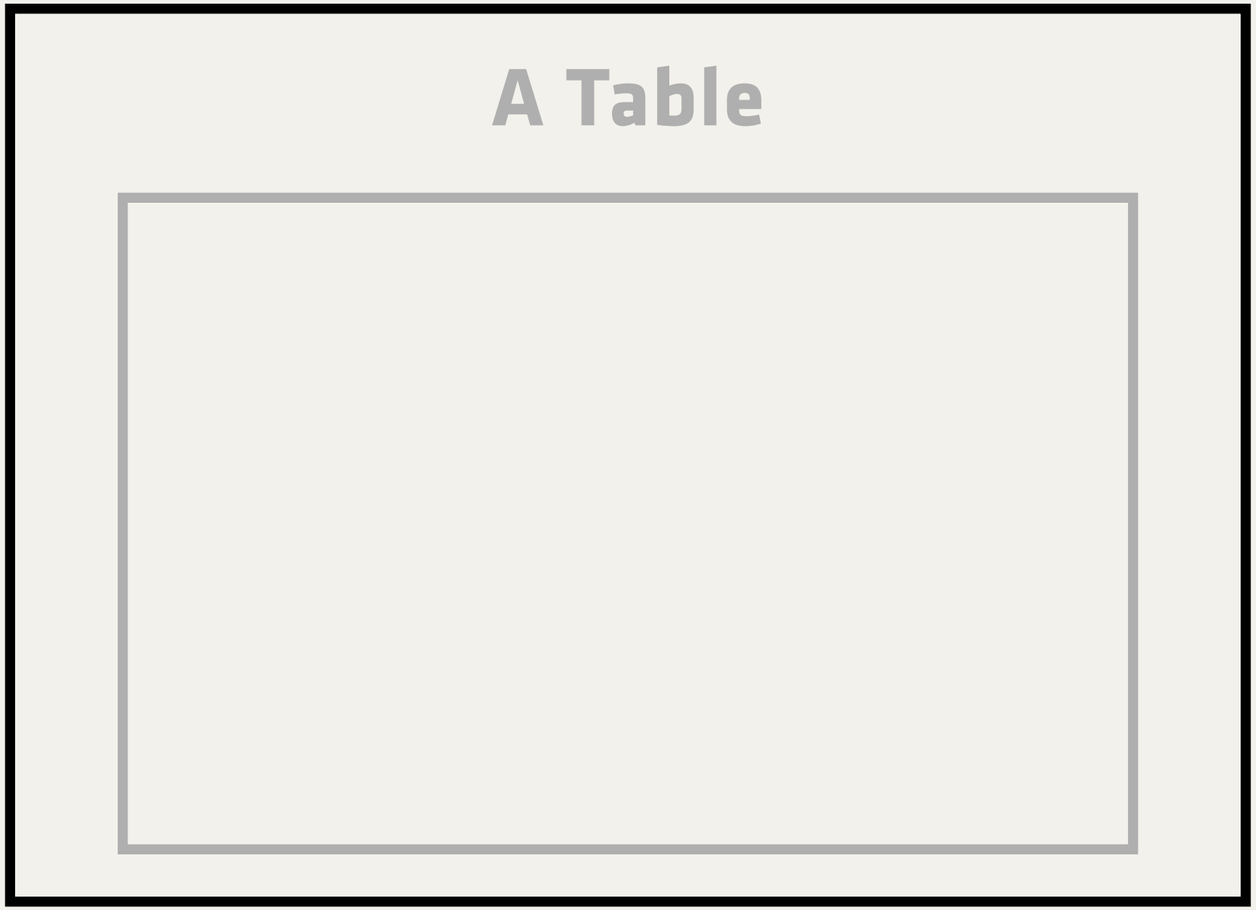
GOALS FOR TABLE CONTROLLER & DATA

- ✓ Display an indicator for the currently selected table
- ✓ Understand master/detail
- ✓ Use a nested Route object
- ✓ Use an ObjectController
- ✓ Use a nested outlet

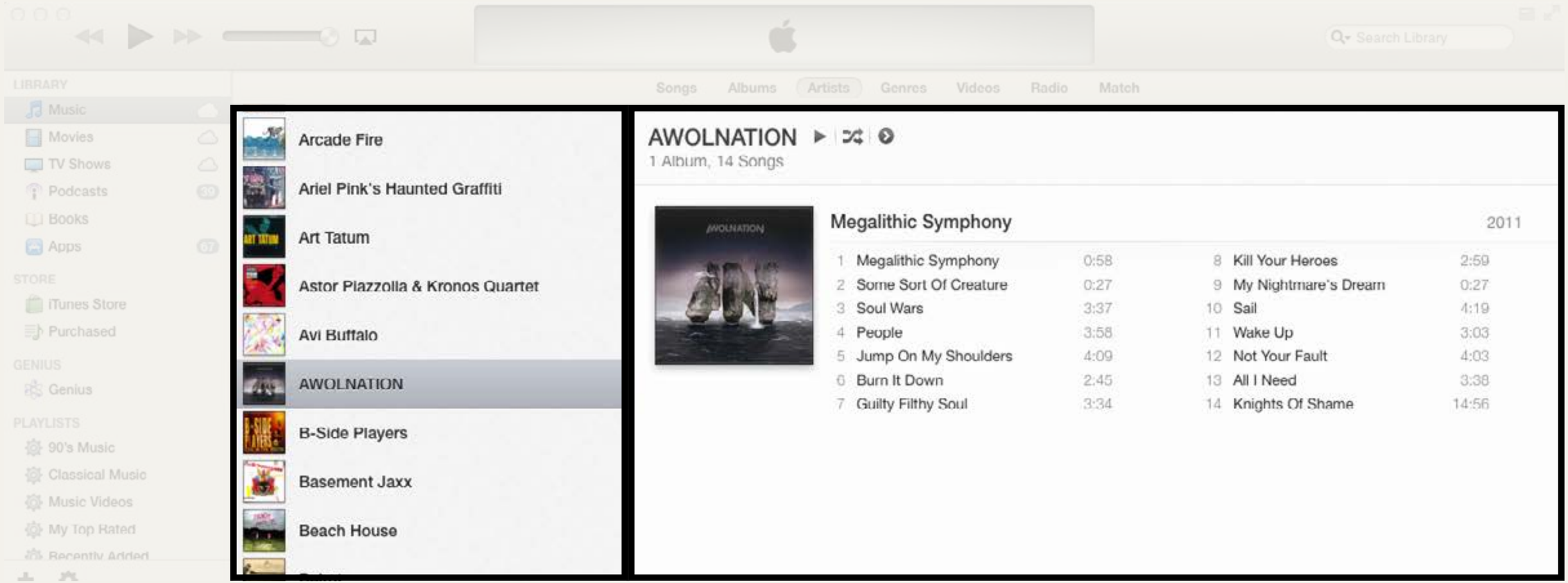
Naming Conventions

ROUTER	this.resource(‘table’) App.TableRoute
CONTROLLER	App.TableController
MODEL	App.Table
VIEW	App.TableView
TEMPLATE	table

Master



Detail



Master **Detail**

“If your user interface is nested, your routes should be nested”

– Yehuda Katz

Ember.Controller

Proxy to its own properties. If a template asks for a property, it will return the value from itself.

EXAMPLE

A template looking for `{{ horsepower }}` would receive `myController.horsepower`.

Ember.ObjectController

Proxies first to its own properties, then to its `model` property.

EXAMPLE

A template asking for `{{ name }}` will get `myController.name` (if it exists) or `myController.model.name`.

Ember.ArrayController

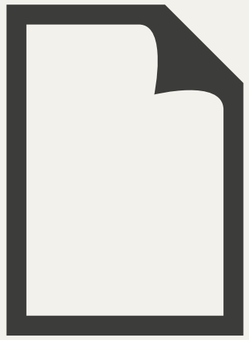
Proxies to itself, then to its `model` property.

Assumes that its `model` property is a list of many objects.

Also provides iteration around its `model` with `each` and has a `length` property.

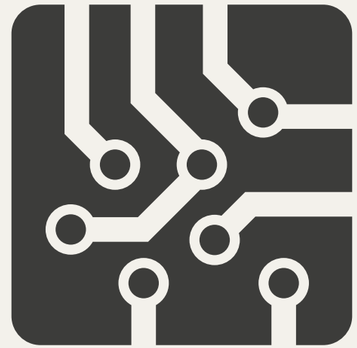
EXAMPLE

A template asking for `{{#each controller}}` will get each of the objects in `myController.model`.



Passive Code Generation

Lines of code are generated from templates and *written to disk*. It happens once, then it's up to you to edit it.



Active Code Generation

Objects are built *in memory* for you. You'll never see the code, just the results in your application. You can implement the objects in code if you need to override the built-in behavior.

All controllers are instantiated at boot

Ember creates one of each of your controllers when your application starts.

IMPLICATION

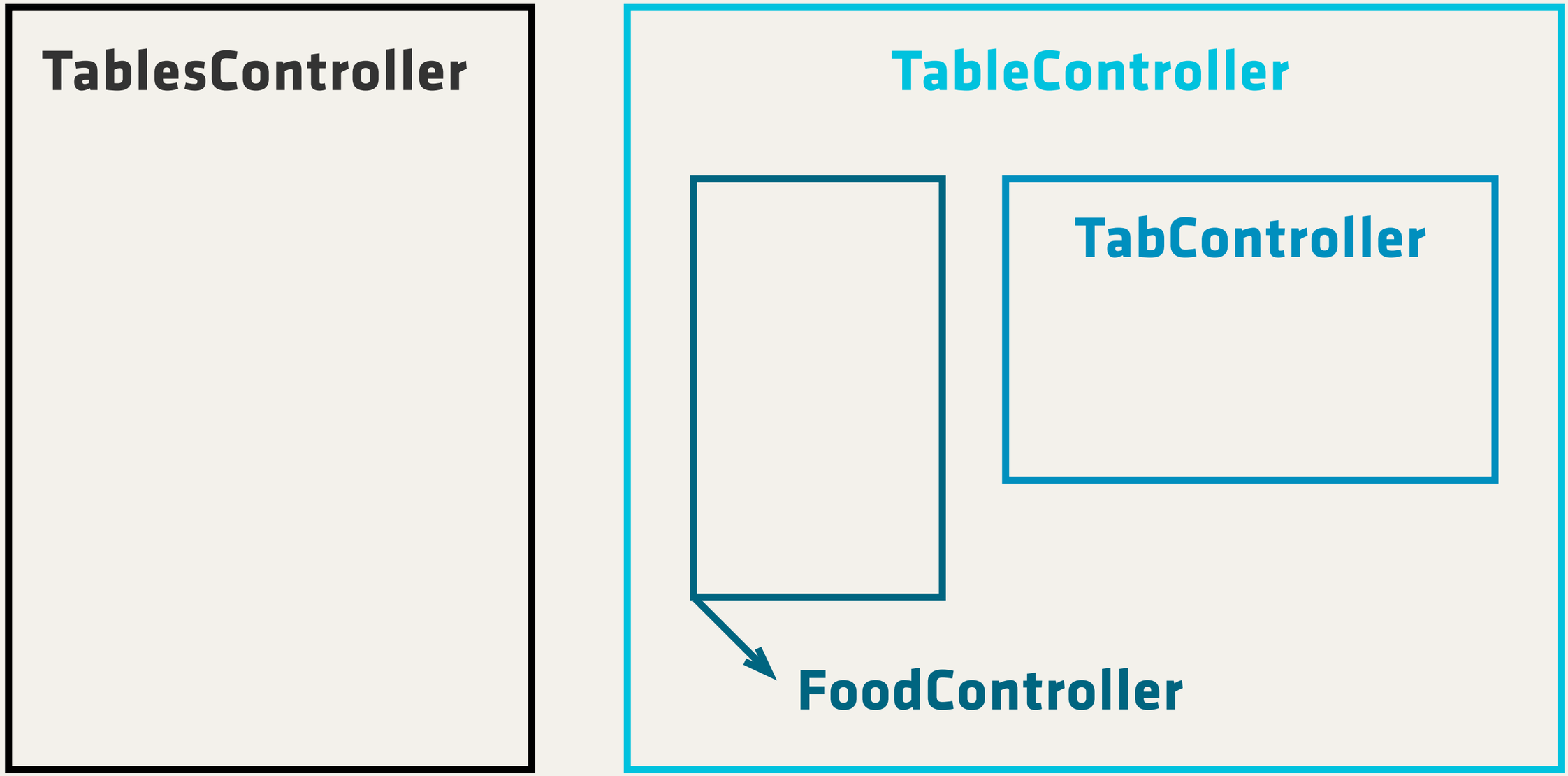
You can assign data to any controller's `model` property at any time.

Controllers are long lived

Controller instances stay around as long as your application is running in a user's browser.

IMPLICATION

A single controller instance will manage many different `model` objects over the course of its lifetime.


TablesController
TableController
TabController
FoodController

NO. 8

Model Fixture Data

GOALS FOR MODEL FIXTURE DATA

- ✓ Understand data models
- ✓ Make relationships between models
- ✓ Use fixture data
- ✓ Use pre-baked model and fixture data

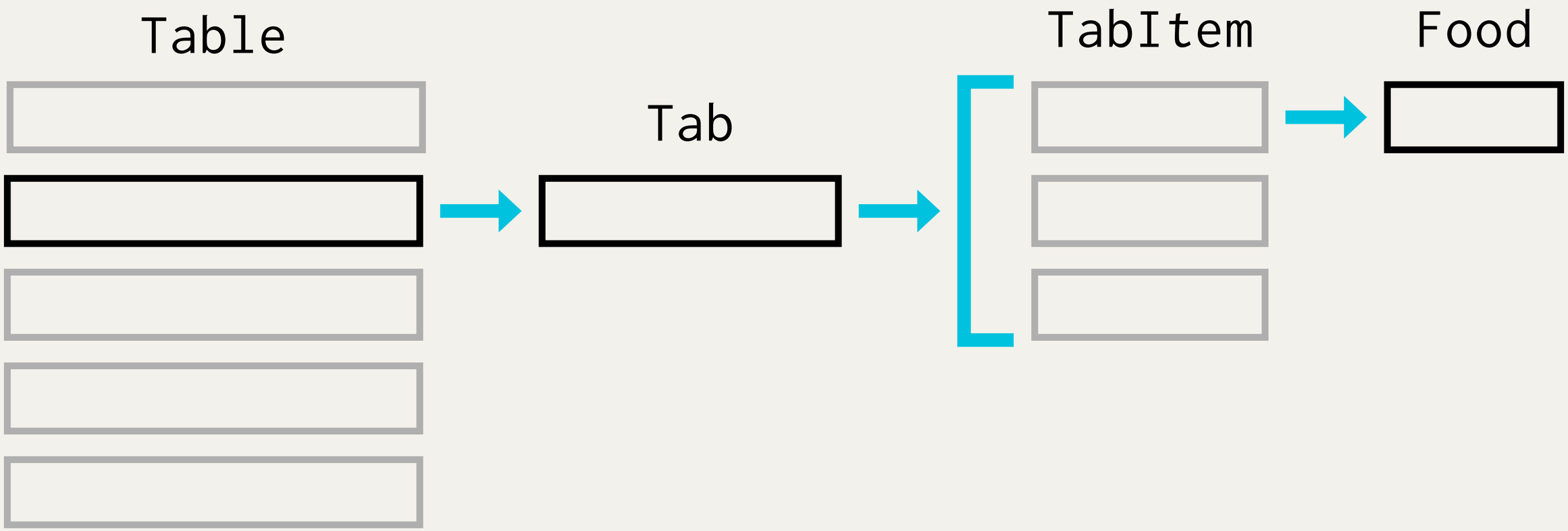


Table
Tab
TabItem
Food

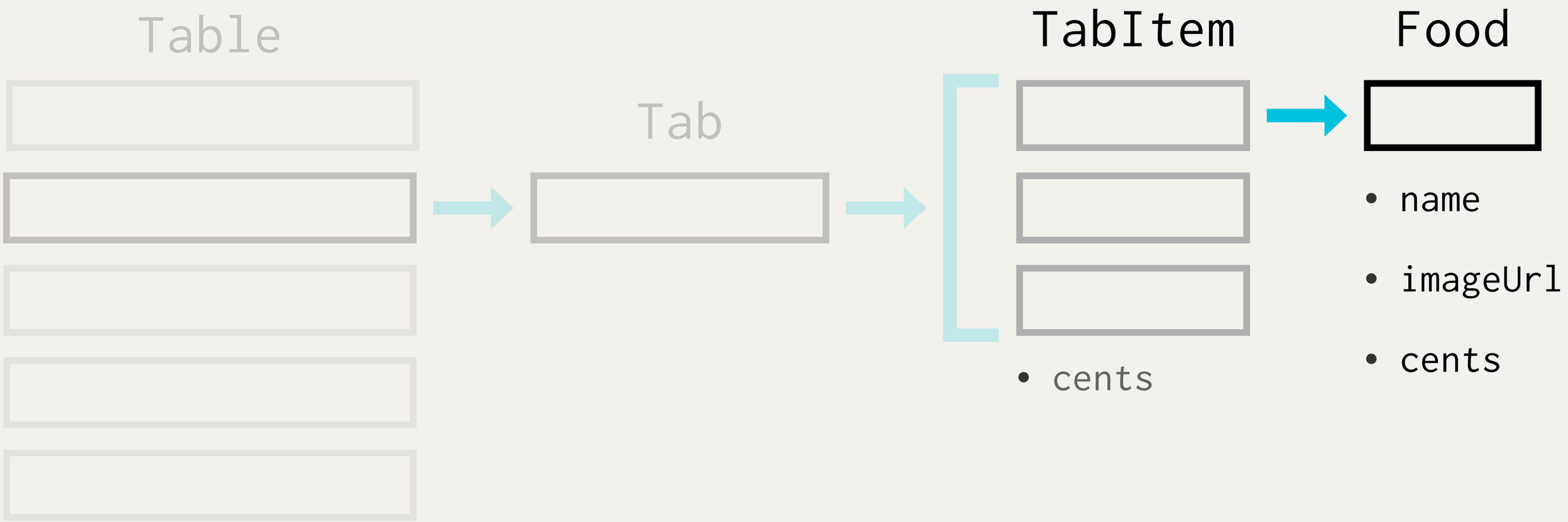


Table
Tab
TabItem
• cents
Food
• name
• imageUrl
• cents


TabItem
Food
• cents
• name
• imageUrl
• cents

NO. 9

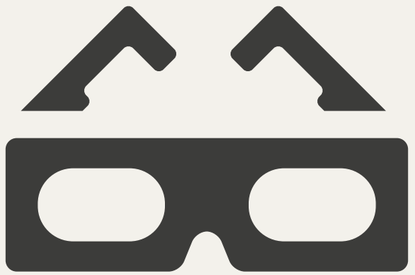
FoodController

GOALS FOR FOODCONTROLLER

- ✓ Display a list of food items on the menu
- ✓ Use a non-route based controller
- ✓ Load model data
- ✓ Use a custom controller in another template
- ✓ Use `{{ render }}`
- ✓ Bind a view to attributes

{{ partial }}

The simplest way to render. Turns template text into HTML using the current controller, data, and all other context.



{{ render }}

Uses a specified controller to render a matching template. The template uses the named controller's model data and context, not the controller from where `render` was called.

NO. 10

Debugging

GOALS FOR DEBUGGING

- ✓ **Use Ember.js non-minified build**
- ✓ **Refresh**
- ✓ `App.Router.router.recognizer.names`
- ✓ `debugger;`
- ✓ `{{ controller }}`
- ✓ **Console**

7 Tips for Debugging

Ember.js Developer Build

Refresh the browser

App.Router.router.recognizer.names

`debugger;`

`{{ controller }}` in views

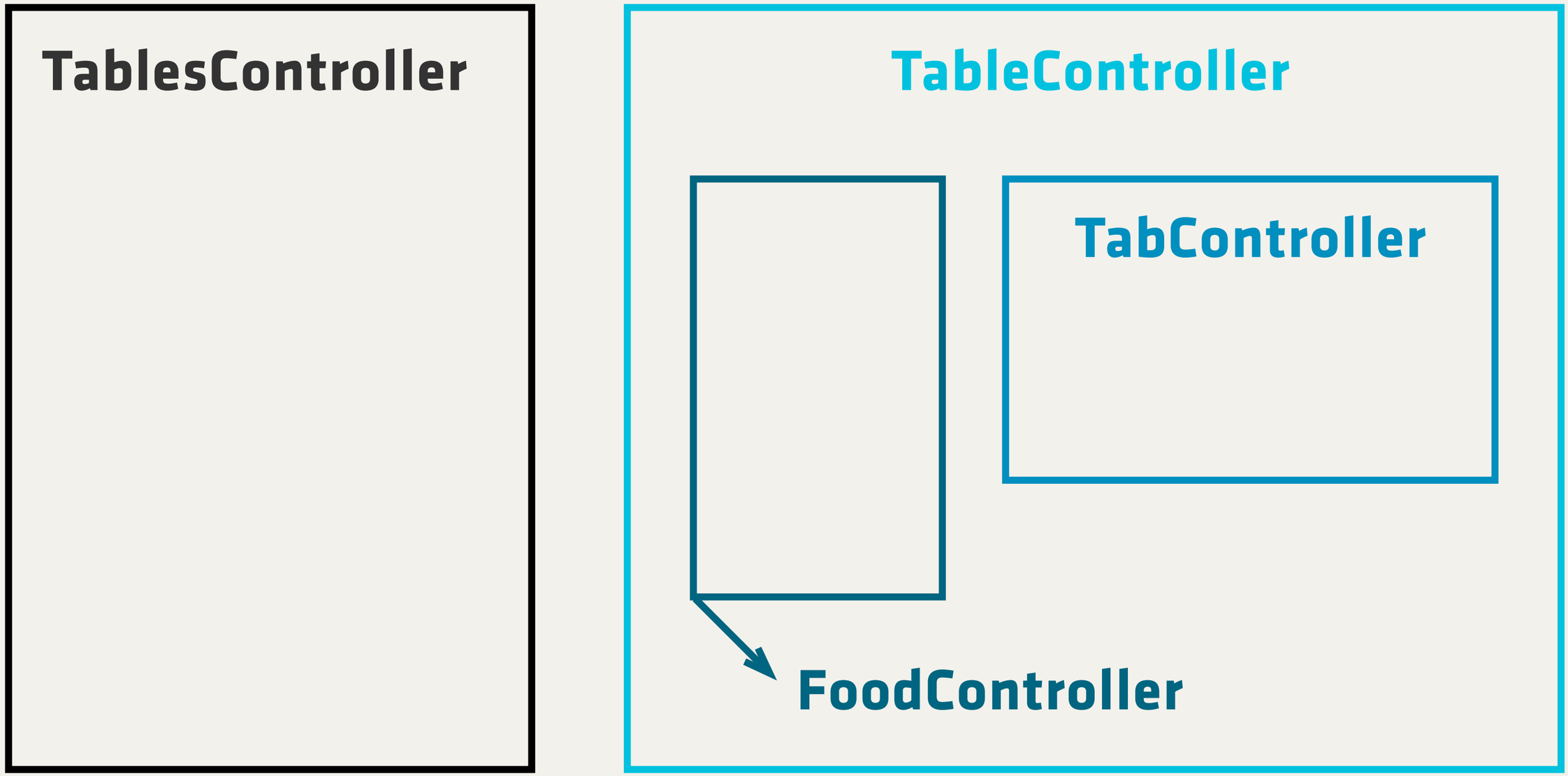
Use a model in the Console

NO. 11

TabController

GOALS FOR TABCONTROLLER

- ✓ Implement the TabController
- ✓ Display the items ordered by the people at a table
- ✓ Render a controller with specific model data


TablesController
TableController
TabController
FoodController

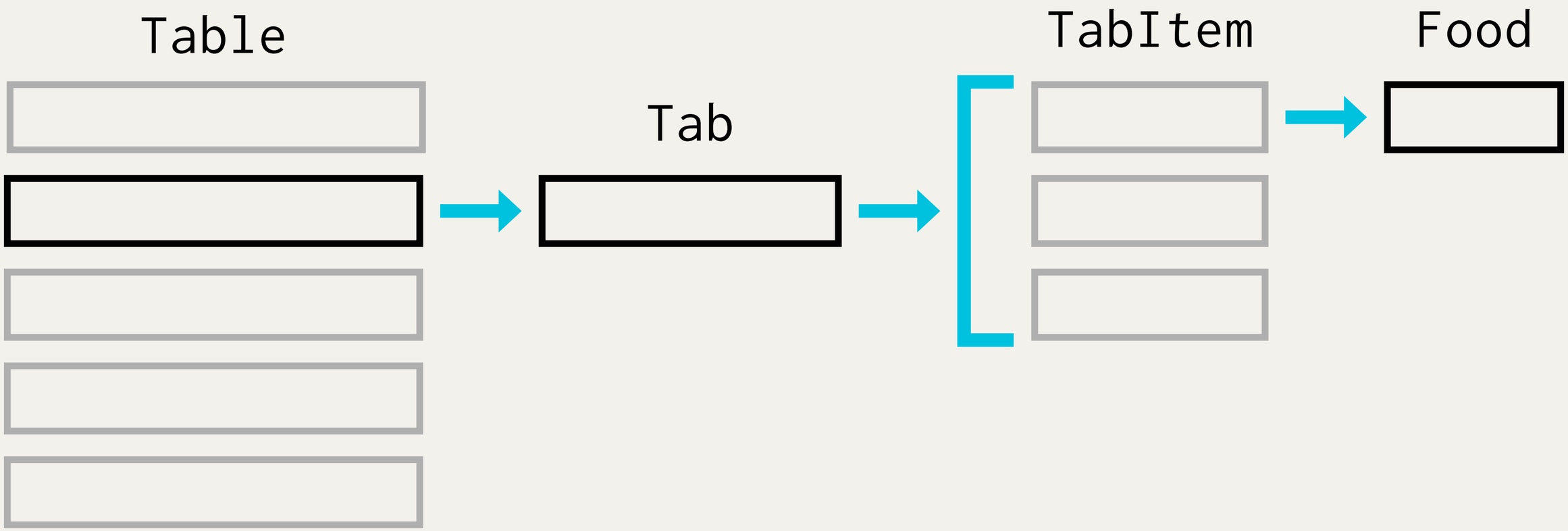


Table
Tab
TabItem
Food

NO. 12

Tab Cents Property

GOALS FOR TAB CENTS PROPERTY

- ✓ Write a computed property on a model
- ✓ Understand property descriptors
- ✓ Use model relationships
- ✓ Calculate a total with `reduce`

```
  cents: Ember.computed ->

    # Calculation

  .property('tabItems.@each.cents')
```

Computed Properties in CoffeeScript

CoffeeScript's indentation makes it difficult to tack a method on the end of another method. Use `Ember.computed` to setup a function that works.

NO. 13

Money Helper

GOALS FOR MONEY HELPER

- ✓ Write a Handlebars helper
- ✓ Format cents as dollars and cents

Formatting Currency

`1,550` → `“15.50”`

NO. 14

Action

GOALS FOR ACTION

- ✓ Tie a clickable link to a method in a controller
- ✓ Write an action
- ✓ Understand where actions can be located in controllers or route objects

"If you find that something is difficult to do [in Ember] or requires significant configuration away from the defaults, it should be a signal to you that you might be going in the wrong direction."

NO. 15

Wrap Tips

GOALS FOR WRAP TIPS

- ✓ Understand generated objects
- ✓ Redirect from the IndexRoute
- ✓ Use setupController with arguments
- ✓ Sort records