Model Collections

- Deal with groups of models instead of a single model
- Used to fetch sets of models from backend server
- Can be used to listen for events across all the models at the same time

Collection Initialization

- The collection must use an already created model (a Book for example)
- Has an *initialize* function like models. Used to set up event listeners and other startup behavior
- When instantiating a collection, an array of models can be passed on to be contained in the collection
- The number of models in a given collection can be determined by the *length* property or the *size*() function

Adding models to a collection

- New models can be added using the *.add* or *.push* methods. Both accept a single model or an array of models.
- *.push* and *.add* add the model to the end of the collection
- The *unshift* method is used to add the model to the beginning of the collection
- If the model exists already, the *add* method will ignore it unless the *{merge:true}* parameter is passed

Removing models from collection

- *.remove* method is used to remove a single model or an array of models from a collection
- It triggers the *remove event*. The event handler function provides an *options* object to find the index of the removed model
- *.pop* method removes the last model in the collection and returns it
- *.shift* method removes the first model in the collection and returns it
- *.reset* method:
  - Replaces the models inside a collection in one go
  - Raises one reset event rather than multiple add and remove events
  - Can be used to empty the collection if called without arguments

Updating the Models in Collection

- *.set* method is used to make "smart updates" to the collection. It accepts an array of models and an *options* parameter object
- It follows the following rules:
  - The new model will be added if not already in the collection unless *{add:false}* is specified
  - If the new model is in the collection, it will be merged with the existing one unless *{merge:false}* is passed
  - A model in the collection that is not in the array will be removed from the collection unless the *{remove:false}* is provided

Retrieving Models from Collection

- The .*get* collection method retrieves the model using it's *id* or *cid*
- The .*at* collection method retrieves the model using it's index in the collection
- The .*at* can be used with JavaScript *for* loop to retrieve all the models in a collection
- .*forEach* method (provided by *Underscore*) can also be used to retrieve models from the collection

Other Useful Methods

- *.pluck* to get a list of only one attribute in the collection
- *.where* is used to search the collection. It accepts key-value pairs as search criteria.
- *.findWhere* is like .where but it returns only the first match instead of an array of matches
- *.groupBy* groups models based on an a common attribute

Working with Back End Server

- Specify the REST url in the url section while creating the collection
- *.fetch* retrieves the collection from the server. It provides *success* and *error* callback functions
- Collections cannot be saved in batches to the server. Instead, each model is added to the collection and then saved manually using the *save* model method
- *.create* collection method is used to add a model to the collection and save it to the server in one operation
- Collections cannot be deleted in batch. Instead, each model is deleted from the server using the *destroy* model method