A Backbone Model

- Every backbone application should begin with a model definition
- A model is where data resides. You should break your problem domain into individual elements and describe each as a model
- A model is created by *extending* backbone’s base Model class
- The constructor object of the model is the *initialize* function
- Models must have default *attributes* available

Model attributes

- Represent the data the the model holds, and they are in the form of key-value pairs.
- Retrieved by using the *.get* model function
- Set by using the *.set* model function
- Deleted using the *unset* model function
- Can be passed to the model during creation
- Listed in JSON format using the *.attributes* model property
- You can check for the existence of an attribute using the *.has* model function

Models Change Event

- The most important event is the *change* event
- Normally added in the *initialize* constructor function
- Configured using the *.on* model function
- *.on* function takes “*change*” as the first argument or “*change:attributeName*”
- Does not fire when the *silent* parameter is used with the *set* model function

What has changed?

- You can figure this out by using:
  - *.hasChanged(attributeName)* to determine whether or not a specific attribute has changed.
  - *.changed* to get a list of all changed attributes in a JSON object (use JSON.stringify to display)
  - *.previous(attributeName)* method returns the previous value of the the passed attribute
  - Will not get triggered if *.set* or *.unset* had the *true* parameter set

Model Validation

- Used to determine whether model data is in correct state or not (for example an incorrect date)
- Used by specifying a *validate* method in model creation
- Triggered on the following conditions:
  - When *.save* function is invoked
  - During every *set* operation if *{validate:true}* parameter is provided
- To provide feedback, you add a listener for "invalid" event in in the initialize function
- To check if the model is valid or not you use the *isValid* attribute

Node.js

- A simple JavaScript based web server that works on all platforms
- Can be downloaded freely from www.nodejs.org
- Uses *npm* to install additional components
- Can be used to test and implement a RESFful API

What is a RESTful API?

- **RE:** Representational **S:** State **T:** Transfer
- A standard way of working with data commonly used in single page applications
- Not obligatory to use but is considered a best practice
- Uses the following methods to work with data:

Resource	Verb	Use
http://localhost/books	GET	List all books
http://localhost/books	POST	Create a new book
http://localhost/books/1	GET	Retreives information for book with ID 1
http://localhost/books/1	PUT	Updates information for book with ID 1
http://localhost/books/1	DELETE	Deletes book with ID 1

Model Identifiers

- Models have three attributes for identifying themselves while working with the server: *id*, *cid*, and *idattribute*
  - **id**: Often will map to DB primary key
  - **cid**: built-in, provides temp id until a real *id* is set to the model
  - **idattribute**: used if the *id* provided by the backend server is different from the *id* of the model. It provides the necessary mapping

Saving Models

- *urlRoot* attribute is used to instruct the model to connect to the backend server
- The *save* method:
  - Invoke the *validate* model method (if specified)
  - Is invoked to make the model persist to the backend server.
  - Will make a POST request if the model *id* is not set and will make a PUT request if it is
  - Can be called without any parameters or can take specific model attributes to be saved
  - Can be called with an *success* and *error* handlers to report the outcome of the server request
  - If the model has an *id* set, save will make a PUT request to update the model, otherwise it will make a POST request to add a new one

Retrieving Models

- The *fetch* model function retrieves the model from the server
- It accepts *success* and *error* callbacks
- If there is a change between the model on the client and the one stored on the server, a *change* event will be triggered

Deleting Models

- The *destroy* model function deletes the model from the server
- It accepts *success* and *failure* callback functions
- It's useful to add *wait:true* object parameter to the function to ensure that the server has deleted the model before removing it from the client side