Step 0 - Ember Starter kit

* Starting off with Ember starter kit with tweaks
  + Twitter.css
  + showdown.js
  + moment.js

Step 1 - Create an App

* App = Ember.Application.create()
  + Namespace
  + Event Listener
  + Router -> routing
* Refresh the page and see nothing
* Add the application template(Snippet: **step1**)
  + Handlebars
  + Describes the UI
  + Backed by a model
  + Binding
  + Plain HTML wrapped in a script tag with a special type
* Refresh the page and see that Ember has rendered this template to the page

Step 2 - The About route

* Create a router (Snippet: router)
  + The router translates a URL into a series of nested templates, each backed by a model. As the templates or models being shown to the user change, Ember automatically keeps the URL in the browser's address bar up-to-date.
* Create about template (Snippet: **about\_template**)
* Highlight convention of template id and route name
* Refresh page and nothing happens.
* Add #/about to url and still nothing happens
* Add {{outlet}} to main application template
  + Outlets are placeholders for template rendering
* Refresh the page and see the about content
* Add {{#linkTo 'about' }} helper
* Click the link
  + renders content
  + Updates the url
  + Adds the active class (bold)

Step 3 - Posts (Getting data from the server)

* 3a - Ember data
  + Define a store (Snippet: **store**)
    - The store is the repository that holds loaded models, and is responsible for retrieving models that have not yet been loaded.
    - Using the FixtureAdapter store for now
  + Define a model (Snippet: **post\_model**)
    - A model is an object that stores persistent state.
  + Fixture data (Snippet: **post\_fixture**)
    - Call out the convention
* 3b - Displaying static content
  + Define Posts resource
    - Add to Router: this.resource('posts');
  + Define Posts template (Snippet: **posts\_template**)
  + Change posts link in navbar to use {{linkTo 'posts'}}
  + Refresh the page and click posts and you should see static content
* 3c - Display the fixture data
  + Use the handlebars each helper to iterate over the posts
  + wrap <tr><td> with {{#each model}}....{{/each}}
  + Replace Title and Author with {{title}} and {{author}}
  + Refresh the page, click Posts
    - nothing is displayed
    - We haven't told Ember what model to use for the template
* 3d - Create a Route
  + A route is an object that tells the template which model it should display.
  + App.PostsRoute (Snippet: **posts\_route**)

Step 4 - Display post detail

* When clicking post link, display contents of the post on the right.
* 4a - Create a nested route (works with the URL)
  + Add function() argument to 'posts' resource definition
  + function() { this.resource('post', { path: ':post\_id' }) }
  + It includes a dynamic segment
* 4b - Define the post template
  + Snippet: **post\_template**
  + Add linkTo helper to posts template
    - {{linkTo 'post' this}}...{{/linkTo}}
  + Refresh..hover over link and see the url...click and nothing happens. Why not?
  + Add {{outlet}} inside "span9" of posts template
  + Refresh the page
  + Click the link
  + Look at the url

Step 5 - Handlebar helpers

* Fix the ugly date with moment.js
* 5a - Create the date helper (Snippet: **date\_helper**)
  + Add it to the template everywhere that we are using
  + Refresh that page and voila!
  + Bound helper = Update the date in dev tools
    - App.Post.find(1).set('publishedAt', new Date())
* 5b - Format the contents of the posts as Markdown (Snippet: **markdown\_helper**)
  + Ember templates escape html by default so we have to use SafeString
  + Add it to intro and extended
  + Refresh the page

Step 6 - CRUD

* 6a - Add edit mode content to the post template
  + Snippet: **post\_edit\_template**
  + #if helper
  + action helper - triggers events (controller -> route)
* 6b – PostController (Snippet: **post\_controller**)
  + A controller is an object that stores application state. A template can optionally have a controller in addition to a model, and can retrieve properties from both.
  + Named functions to handle actions
  + Notice the "set" functions - this is for binding
* 6c - Editing UI
  + Create a partial (Snippet: **posts\_edit\_partial**)
  + View
    - Need advanced user event handling
    - Re-usable component
  + Use partial helper to include content {{partial 'post/edit'}}
  + Show that edit works

Step 7 - Loose ends

* 7a - Nothing is displayed when Posts clicked
  + Posts index (Snippet: **posts\_index\_template**)
* 7b - On app initial load, show posts
  + Explicitly create IndexRoute that redirects to posts
  + Snippet: **index\_route**

Step 8 - Get the data from a server

* Delete the fixture data
* Change store to use DS.RESTAdapter
* adapter: DS.RESTAdapter.extend({
  + url: 'http://localhost:3000'
* })
* Serve the app

Step 9 - Implement save

* In doneEditing:
* this.get('store').commit();

Tooling

- Ember Chrome Plugin

- Ember tools

- About

- node-based tool

- file organization conventions

- scaffolding (rudimentary)

- template precompilation for performance

- single file application build for convenience

- generators for faster application development

- commonjs (node-style) modules

- Demo

- ember create ember-tools-client

- ember generate --scaffold person name:string age:number

- ember build

- open index.html

- visit #/people

- Yeoman

- About

- a collection of tools and best practices working in harmony to make developing for the web even better.

- Node

- Yo - Scaffolds

- Grunt - Task-based build tool

- Bower - dependency management for the web

- Ember Generator

- Demo

- mkdir yeoman-client

- cd yeoman-client

- yo ember

- grunt server

- grunt test