

INTRODUCTORY GUIDE TO

EMBERJS GUIDE

BROUGHT TO YOU BY TUTORIALSLINK.COM

Author
Yamini Sharma

Co Author
Nitin Pandit

EmberJs Guide

Table of Content

1. Overview 4
2. Introduction 5-6
    2.1 What is ember.js?
    2.2 History of ember.js.
    2.3 Importance of ember.js.
    2.4 Features of ember.js.
3. Installation of ember.js 7-9
4. Fundamental concepts of ember.js 10-11
5. Router 12-17
    5.1 Introduction
    5.2 Defining Route
    5.3 Specify the route's model
    5.4 Rendering template
    5.5 Redirecting
    5.6 Preventing and Retrying Transition
6. Components 18-22
    6.1 Templates and Declaring components
    6.2 Component lifecycle
    6.3 Passing properties of a component
    6.4 Wrap content into a component
    6.4 Event Handling
7. Models 23-27
    7.1 Define a model
    7.2 Record finding
    7.3 Record creating and deleting
    7.4 Relationship between Models
    7.5 Pushing the Records
8. Ember.js Object Modelling 28-30
    8.1 Class and Instance
    8.2 Reopening class and Instance
    8.3 Computed properties
9. Application of Ember.js 31-32
    9.1 Dependencies Injection
    9.2 Initializers
    9.3 Service

9.4 Run Loop
10. Configure of Ember.js 33-34
10.1 Configure CLI and app
10.2 Disabling the extension of prototype and specifying URL type
10.3 Embedded application and flag features
11. Ember Inspector 35

Overview

Ember.js is an open-source framework of client-side JavaScript. It used for developing web applications. It uses the MVC model. In this tutorial, you will get an in-depth knowledge of the working of Ember.js.

Who can learn Ember.js?

Anyone can learn Ember.js. But if you are a software developer it is easy for you to understand and grab the concepts of the framework. You will get enough knowledge of all the basic concepts of Ember.js.

What is the basic prerequisite required to learn the Ember.js?

To understand the concepts efficiently, you must be aware of the HTML, CSS, JavaScript, and any other text. It will help you to understand the components in a clear way.

Chapter-1

EMBER.JS

What is Ember.js?

It is an open-source and free framework of JavaScript client-side. It is used in developing a web application of client-side JavaScript. It serves a complete solution of managing the data along with the application flow. It permits us to develop a client-side JavaScript Application.

History of Ember.js

Initially, it was named as a SproutCore MVC framework. Yehuda Katz is the founder. In December 2011, Yehuda Katz released the first version of Ember.js. It follows the MVVM (Model-view ViewModel) pattern. The most recent version is 3.20.0 that is released in July 29,2020.

Importance of Ember.js

It has its importance. Try to understand it with the following points:

- It is a free and open-source framework underneath the MIT license.
- It provides the new syntax with the help of the HTMLBars template engine.
- It also focuses on increasing the rendering speed by using the Glimmer rendering engine.
- It also comes up with command-line utility which helps in faster development of an application.
- It also updates the property of the automatically, if one property changes as it establishing a link between the two properties.

Features of Ember.js

Ember.js has some noteworthy features:

- Reusable and maintainable application of JavaScript can be easily developed by Ember.js
- It uses HTML and CSS for the fundamental development of the application.
- URL's are managed by one of its core components known as routes.
- It also provides a debugging tool.
- It uses the kind of template that updates the information automatically whenever the content of an application gets changed.

Why is preferable for creating a web application?

It is used in many well-known websites such as LinkedIn, Vine, Apple Music, Twitch and so on. It is very simple to create a desktop and mobile application with the help of Ember.js

Chapter -2

EMBER.JS Installation

You can easily install the ember.js in your system. The project creation and management are done by Ember Command-line Interface (CLI). CLI deals with the different kinds of components of management. It also comes up with generators that provide the routers, models, etc.
Following prerequisite dependencies are required to install the Ember CLI:

- Git: It is used to track the changes in the file. Ember is using Git to manages and control the dependencies.
  You can install the Git on your Operating system with the help of the following links.
  - For Windows: https://git-scm.com/download/win.
  - For Linux: http://git-scm.com/download/linux.
  - For Mac OS: https://git-scm.com/download/mac

- Node.js and npm package: Node.js is used for developing the server-side and networking applications. Node.js is written in JavaScript. It is used to create an application faster and also in a scalable manner. It uses on JavaScript server. NPM stands for the Node Package manager. It is used for performing various functions on dependencies such as installing, sharing, and managing. Ember CLI uses the run time of the Node.js and npm is used to get the dependencies.

- Browser: It is a pursuit for governing the sub-components like HTML, CSS, image files, and those which can be installed by using npm.

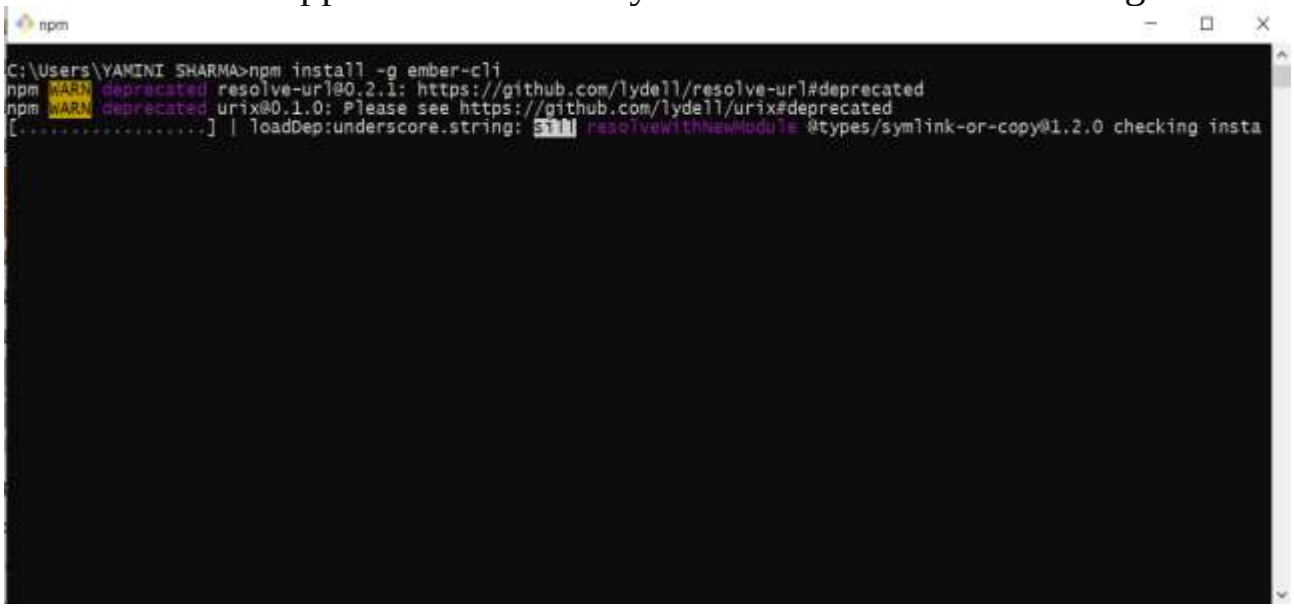
- Watchman: As the name suggests, it is used to keep the eye on the files and directories. In case of any changes, it executes the actions. It is an optional dependency.

Steps to install the Ember CLI

Ember CLI unites the patterns of ember into the development process of application and keeps the eye fruitfully on the productivity of the developer.

Once you have installed the git in your system, run the following commands to install the Ember CLI.

- Install the ember by using npm. Type the given command below in the npm.
  npm install -g ember-cli
- The installation appears the same as you can see in the below image:



- Now, you can install the beta version by using the following command.
  npm install -g ember-cli@2.10

- You can check whether ember is installed in your system or not by enter the command given below:
  ember -v
  Once it is installed, it will appear like this:

```
ember-cli: 2.10.1
node: 12.18.0
os: win32 x64
```

How to check whether the ember.js is running on a machine?

Steps for check that your ember.js is working fine on your machine:

[1] Install a npm package.
[2] Test the ember version installed in your machine.
[3] You can create an application anywhere in your machine such as desktop, c: drive, or etc.
[4] Type the following command to create an application

```
C:\Users\YAMINI SHARMA\Desktop>ember new first-app
```

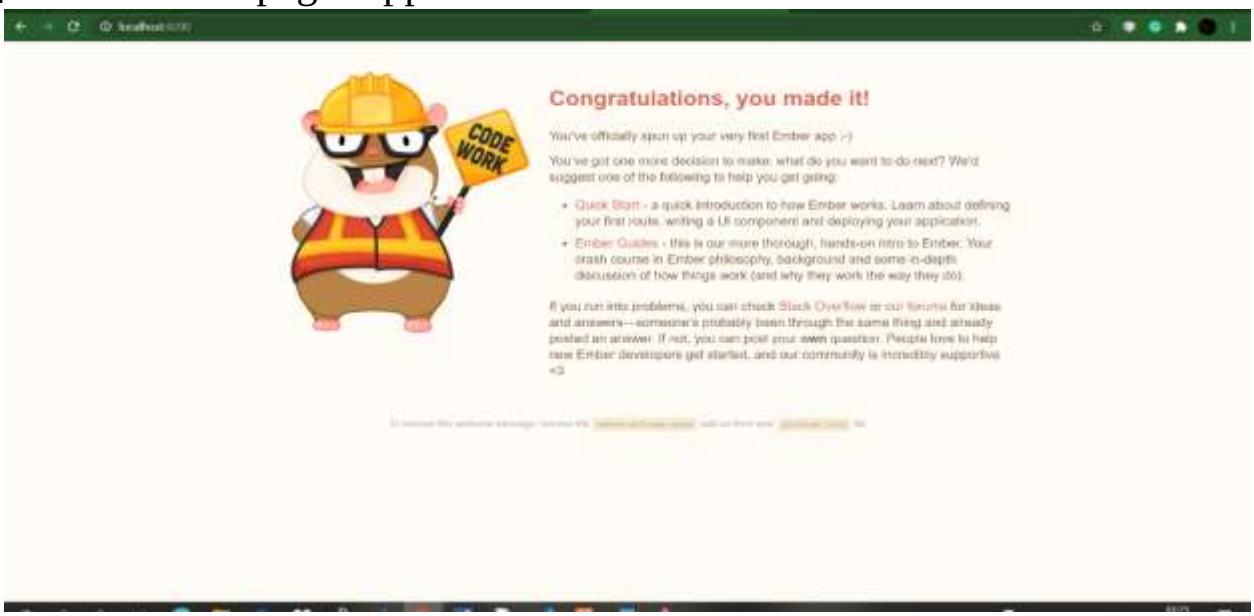
This command will create an application and install all the packages with it.

[5] Open your application in the any of the editor in which you are going to edit the application.

[6] Write the following command into the terminal

```
YAMINI_SHARMA@LAPTOP-DGRDDUT9 MINGW64 ~/Desktop/ember/first-app (master)
$ ember server
```

[7] The first pages appear like this.



Chapter -3

EMBER.JS Fundamental Concepts

It has four main concepts:

- Router
- Template
- Models
- Components

Router: The main function of the router to map the URL to the router handler. Enter the URL in the address bar to load the application. The link will appear to the user. The user will click on the link within the application. The router will now match the route with the existing URL that are pre-owned for using the data loading, visualizing the templates, and also setting up the state of the application.

Router handler performs these measures:

- It gives a template.
- It expounds on the model. The model is approachable to the template.
- The router will redirect to the new page if the user doesn't have permission to access a specified part of the application.

Template: It is the strongest UI for the end-user. It is using the syntax of Handlebars templates. It uses the HTML to build the front-end application. The template is providing the user interface look means how the application appears to the user. It supports regular expression syntax. It also dynamically updates the expressions.

Model: It is just a class that extends the functionality of the data of ember. Ember Data is nothing but a library. This library is close coupled with Ember.js. It is used to handle the data that is stored in the database. The model is rendered by the router handler. It manipulates the data that is stored in the database which carries on the information on the webserver.

Component: It has two parts that control user behavior of the user interface:

- A template which is written in JavaScript.

- The behavior of the components is provided by the sources file which is written in JavaScript.

Chapter-4

Router

Introduction

EmberJs has a router as its fundamental feature. It converts the URL into the number of templates and also illustrates the state of an application. The router is used to map the URL with the router Handler. It matches the entered URL with the already pre-owned routes that are used for loading the data, to set up the sates of an application and also displaying the templates.
Router handler performs these measures:

- It gives a template.
- It expounds on the model. The model is approachable to the template.
- The router will redirect to the new page if the user doesn't have permission to access a specified part of the application.

The table contains the list of routers along with their description used in Ember.js.

Router	Description
Defining Routers	It matches the current URL with those routes who are loading the data, set the state of an application and displaying template.
Specifying a Router's Model	The template is required to display the data from the model to declare the router's model.
Rendering the template	The external templated for the screen is done by the router.
Redirecting	It is the redirection mechanism of the URL. If the entered URL is not found, this mechanism redirects the user into another page.
Preventing and retrying transitions	During the route transition, the two methods are used transition.abort() to abort the transition and transition.retry() to retry the transition.
Loading/error Substates	It provides the information of a loading route. It also provides information of errors, if occurs during loading the route.
Asynchronous Routing	It is used for handling complex logic in an application.

3.1 Defining Routes

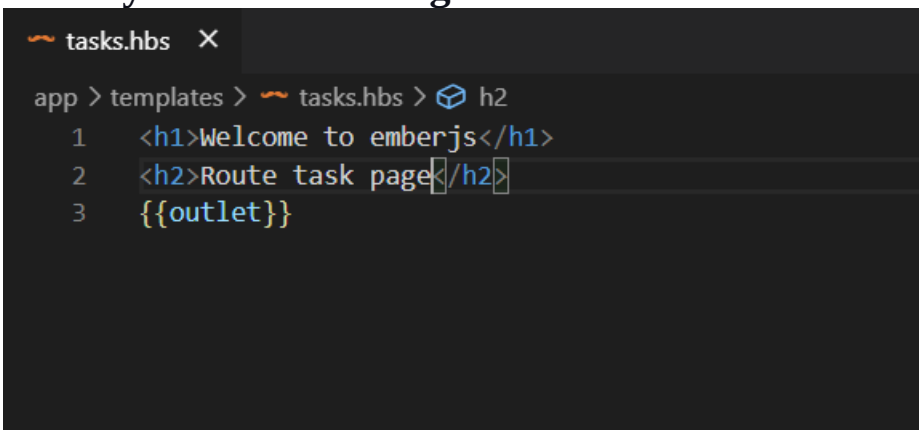
You need to type the following command:

ember -g route route-name

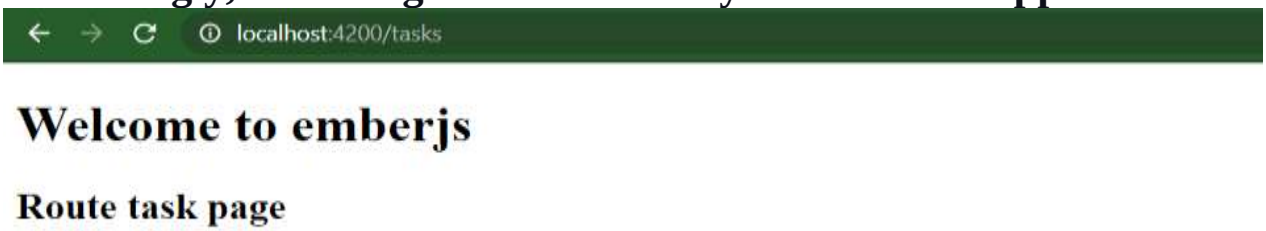
It will appear like this while creating a route.

```
C:\Users\YAMINI SHARMA\first-app>ember g route tasks
installing route
  create app\routes\tasks.js
  create app\templates\tasks.hbs
updating router
  add route tasks
installing route-test
  create tests\unit\routes\tasks-test.js
```

You can change make changes in route according to your design. The file in which you made a change looks like



Accordingly, the changes will reflect in your server. It appears:



3.2 Specifying the Route's Model

<LinkTo> component is used to switch between the routes and the links on the template page.

Example:
<LinkTo @route=”picture.view” @models={{p}}>{{p.title}}></LinkTo>

In the above code, @route will take the route of the page and @model will take the object of the portion of the template you want to display.

Alternatively, you can also pass the id of the object you want to display instead of passing the full object of the portion. Accordingly, the CSS associated with the page will be active on the template.

Query parameter

With the help of {{hash}}helper, you can also set a @query parameter in <LinkTo> component.

For example

<LinkTo @route= ”picture” @query={{hash direction = “acs”}}>View< LinkTo>

3.3 Rendering template

Route handlers will have the same as the route of the template. From the model hook, you can fetch the data with an HTTP request or can pass it on the template. When a function run during setup is commonly referred to as Hooks.

```
Router.map(function() {
  this.route('task');
});
```

The hooks will run the /app/routes/task.js automatically when the user visits the task route.

routes > JS tasks.js > TasksRoute > model

```
import Route from '@ember/routing/route';

export default class TasksRoute extends Route {
    model()
    {
        return 'Welcome';
    }
}
```

Some points need to be considered about model hook:

- Returning the data from the model, it by default becomes available in the .hbs file as models and in the controller as this.model.
- A hook can return can type of data such as date, string, and more.
- Mostly, it returns the JavaScript Promise and route will wait for the promise to be solved before template furnishing.

3.4 Redirecting

Sometimes the user is directed to another page instead of requesting the page. For example, a user wants to open the shopping cart but didn't log in. So, the redirecting mechanism will redirect the user to the login page. The redirecting mechanism has its own importance. It will keep your private information secure and prevent from unauthorized use.

The two methods are used transitionTo() and replaceWith(). Both works in a similar way, the only difference between the two is replaceWith() redirects to any page which is initially embedded for redirection and transitionTo() create a new route of the function.

Before the Model is known

```
Router.map(function() {
  this.route('task');
});
```

```
pp > routes > JS tasks.js > TasksRoute > beforeModel
1   import Route from '@ember/routing/route';
2
3   export default class TasksRoute extends Route {
4       beforeModel(/*Transition*/){
5           this.transitionTo('Task');
6       }
7   }
8
```

transitionTo() will redirect the user where it left. For example, the user is checking the hidden task it might happen the user will redirect to the login page. The user will get back to the same age once it is log in.

After the Model is known

```
Router.map(function() {
  this.route('task');
});
```

```
routes > JS tasks.js > ...
import Route from '@ember/routing/route';

export default class TasksRoute extends Route {
    // eslint-disable-next-line no-unused-vars
    afterModel(model, transition){
        if(model.get('status-line')===1)
        {
            this.transitionTo('task', model.get("Status") );
        }
    }
}
```

Once the model is known, it will redirect the transition to page written in the model hook.

3.5 Preventing and Retrying Transition

Preventing transition through willTransition()

The preventing transition works similarly to LinkTo. It will redirect the page user is calling. It will check the transition is going to happen or not. If the route is calling the following transition the browser will direct it to the new page. If the transition is not occurring. It will abort it by using the transition.abort() function.

Retrying transition

transition.retry() function is used for the retrying the transition.

You can try these functions with model, beforeModel, and afterModel. On the successful completion of the transition, the user will traverse to the new page but if transition fails due to login interruption you can retry it else the transition will abort with the transition.abort() function.

Chapter - 5

COMPONENTS

It provides the absolute encapsulation UI widgets and it uses the W3C web components specialization. It has templates, shadow DOM and custom elements as its major specialization. The component is declared in the data-template-name.

The table consists of the events of action along with their description:

Action Event	Description
Declaring a component.	It is very easy to define a component and it carries a dash in their name.
Lifecycle of component	It carries few methods which will execute to specific code at times in the life cycle of the component.
Passing properties of a component	The component is not allowed to access the property directly in the scope of the template. Always, declare the property in the component declaration.
Wrap the content in the component	With the help of the template, you can wrap the content in the component.
Formatting the component's element	With the help of Ember.Component, you can customize the component's element.
Handling Events	Events like double-click, hovering, keypress, etc are handled by the event handler.
Triggering changes	Components can activate the changes and communicate with events by using the actions.

4.1 Templates and Declaring Component

Command to create a component:

```
C:\Users\YAMINI SHARMA\first-app>ember generate component task-model
```

After declaring a component, you can create a specified template for your website. A static template can be applicable to the pages.

What is Template?

A template is nothing a specified format that applies to all the pages. If you made any change in one of the pages of the template then these changes will automatically get reflected all the pages of the template. It provides you control on all the pages.

The table carries the type of templates along with their description.

Templates	Description
HandleBars Basics	It is a library that allows us to create a rich application that also includes the static HTML and dynamic content.
Built-in Helpers	It does nothing but just provide the extra functionality into the template. It evaluates the raw values of the model. Setup the components in the proper format for the user.
Conditionals	These conditional statement helps to control the flow of the program. It has two types of a conditional statements.
Displaying the list of items	#each helper is used to displaying the list of items of the array.
Exhibiting the key in an object	#each-in helper is used for exhibiting the keys in the object.
Links	It is utilizing to create the link to a route with the help of {{link-to}} component.
Writing Helpers	It used for adding the features into the template. Converts the raw values from models and for users converts components into the proper format.
Input Helpers	{{input}} and {{textarea}} are used for common from controls.
Actions	Through this you can make the HTML tag clickable with the help of {{action}}
Development Helpers	By using the HandlerBars and Ember, the development becomes easier.

You can create a sample component like this:

app > components > task-model.hbs > article.task > h2

```
1  <article class="task">
2      <h1>Hello</h1>
3      <h2>Welcome to tutorial Links</h2>
4  </article>
```

4.2 Component Lifecycle

Component is the most useful and powerful part. It gives the control to DOM section for manipulation, rendering and listening for responding browser events.

The steps are followed in the lifecycle of component:

- Rendered: On initial component rendering the following function you can perform:
  - init()
  - didReceiveAttrs()
  - willRender()
  - didInsertElement()
  - didRender()

- Re-Render: You can also re-render the component with the help of these functions:
  - didUpdateAttrs()
  - didReceiveAttrs()
  - willUpdate()
  - willRender()
  - didUpdate()
  - didRender()

- Destroy: The components are destroyed by using the following functions:
  - willDestoryElement()
  - willClearRender()
  - didDestoryElement()

4.3 Passing Properties of a Component

We need to pass the data into the component, if we want to use it.
For example, you have task component which is used to display the task.

If you include the component of task:

app > templates > tasks.hbs

```
{{#each model as |task|}}
{{task-model}}
{{/each}}
```

HTML needs to be render:

```
<article class="task">
    <h1>{{Title}}</h1>
    <h2>{{Description}}</h2>
</article>
```

To include the property into the component, you need to describe in such manner:

app > templates > tasks.hbs

```
{{#each model as |task|}}
{{task-model title=task.title description=task.description}}
{{/each}}
```

4.4 Wrap the content into the component

Wrapping the content means you can use the content of the template into the specified component.

For example, you have created a component like this:

```
<article class="task">
    <h1>{{Title}}</h1>
    <h2>{{Description}}</h2>
</article>
```

Now, you want to display the content of the task.hbs into this. You can do that also.

4.5 Event Handling

With the help of event handler, you can handle the event by using double-clicking, hovering and key presses. Event are registered using the Application.customEvents.

There are few types of built-in events:

- Touch Event: You can control it with the help of touchStart, touchMove, touchEnd and touchCancel.
- Mouse Event: mouseDown, mouseUp, contextMenu, click, mouseMove, mouseEnter and mouseLeave are used for controlling the mouse events.
- Keyboard Event: keyDown, keyUp and keypress are used for controlling the keyborad events.

Chapter -6

MODELS

To merge the functionality of the Ember data into the class model. The page's content should be represented by the model, whenever the user refreshes the page. Models are linked with every route in the ember.js. It is used to improve performance. It manipulates the data stored in the server and allows it to work easily with the APIs.

There are fundamental concepts:

- Store
- Models
- Records
- Caching
- Adapter.

STORE: It is a central repository. It stores the cache of all the available records in the applications. The stored data of an application is shared by the components and route. To share the data with the entire project, DS.store is created automatically.

this.store().find() is used to fetch the store.

MODELS: It specifies the relationship with the other objects. The model is used to represent the content of the page.

RECORDS: A record is the object of a model. It carries the information loaded from the server to identify the record by its model type and ID.

You can find the record by using the following statement.

this.get('store').findRecord('name',1);

ADAPTER: It is an instance through which requested records from Ember are translated into the appropriate calls to specific server backend.

CACHING: The records are retrieves from the server automatically. When you load the instance of the object, it will revert the same object instance. It helps in improving the performance and UI of an application.

The table consists of the detail of the model along with the description:

Model	Description
Define a model	It is just a class that merges the functionality of the ember-data.
Record finding	Ember datastore is used to get the records back.
Creation and deletion of records	The creation and deletion of the model are done on the instance.
Relationships	Through this, you can specify the relationship between the different models.
Metadata handling	Metadata is used for storing a large amount of data instead of using records.
Records pushing in the store	You don't need to request the record of an application to storing the cache in the store.
Formatting Adapter	It specifies how data is stored in the backend.

5.1 Define a model

Command used for creating a model

```
C:\Users\YAMINI SHARMA\first-app>ember generate model task-model
```

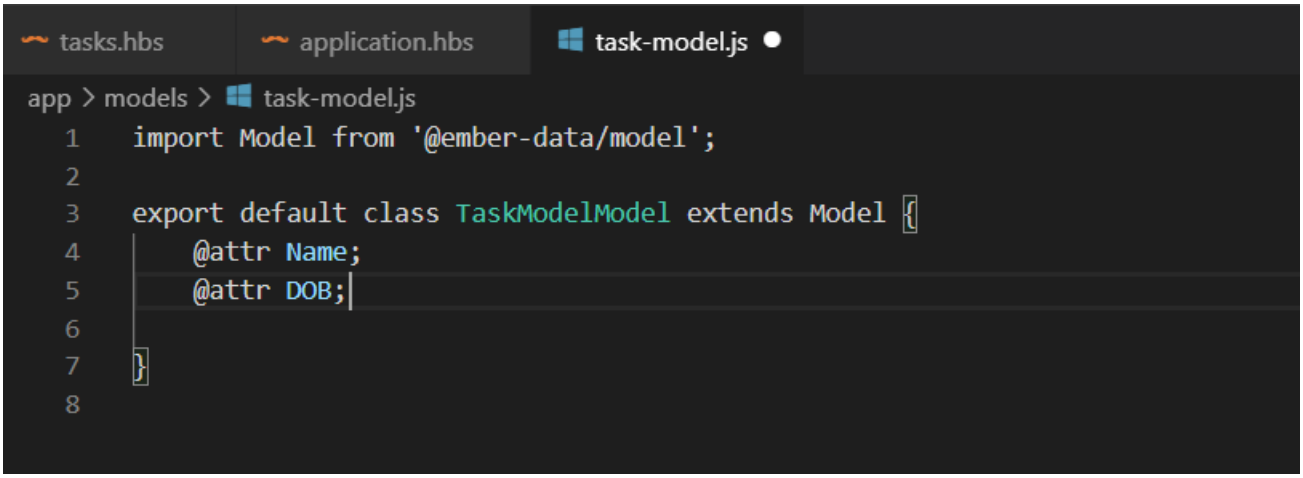
It extends the model class and generate the file like this:

tasks.hbs | application.hbs | task-model.js

app > models > task-model.js > ...

```
1  import Model from '@ember-data/model';
2
3  export default class TaskModelModel extends Model {
4
5  }
6
```

You can define an attribute like this in your model file



5.2 Record Finding

Retrieving a Single Record

There are two methods through which you can get the record back:

- store.findRecord(): It is used to retrieve the single record along with the type and id. You need to make a network request in this way.
- store.peekRecord(): In this way of retrieving the record you don't need to make a network request. It the requesting record is present in the store it will show the record.

Retrieving Multiple Record

There are two methods through which you can get the record back:

- store.findAll(): It is used to retrieve the single record along with the type and id. You need to make a network request in this way.
- store.peekAll(): In this way of retrieving the record you don't need to make a network request. It the requesting record is present in the store it will show the record.

5.3 Record creating and deleting

Record creation

createRecord() function is used for creating the record in the store.

For example

store.createRecord('task',{

title: 'Create a Record',

description: 'Creating a new record and store it'});

Record deletion

It is as simple as creating a record. deleteRecord() method is used for deleting the record. In order the save the changes call Save() method. Apart from this, you can also opt for the alternative option in action to delete the record for the specified time. destroyRecord() is used for destroying the record for the particular time.

Record Updation

Updating a record is also simple. Apply the finding record command and use set inside the function. It will automatically update the record which you found.

5.4 Relationship Between Models

There are two types of relationship can be established between two models:

- One-to-one: Two models are connected in one to one way relationship. It uses DS.belongsTo.

```
import DS from 'ember-data';

export default DS.model.extend({
// eslint-disable-next-line ember/use-ember-data-rfc-395-imports
post:DS.belongsTo('post')
});
```

```
import DS from 'ember-data';

export default DS.model.extend({
// eslint-disable-next-line ember/use-ember-data-rfc-395-imports
task:DS.belongsTo('task-model')
});
```

  In this way, you can establish a one-to-one.

- One-to-many: DS.hasMany is used to establish a many relationship with the another model. So, use hasMany in the model in which can you want to create many relationships.
- Many-to-Many: DS.hasMany is used in the both the model commands.

5.5 Pushing the records

Pushing the records in the cache is best practice. Ember.js stores the records either in the cache or in the server. Requesting the records from adapter, network will search it from the server, and provide the data.

Store's push() method is used to push the record into the store.

For example

app > models > JS post.js > default

```
import post from '@ember/optional-features';

export default post.Model.extend({
    title: post.attr(),
    Description: post.attr()
});
```

```
Router.map(function() {
  model(){
    this.store.push({
      data: [{
        id: 1,
        type: 'blog',
        attributes: {
          title: 'Ember.js',
          description: 'What is Ember.Js'
        },
        relationships: {}
      }, {
        id: 2,
        type: '',
        attributes: {
          title: 'Applications of Ember.js',
          description: 'Where it is used in real-life scenario'
        },
        relationships: {}
      }]
    });
```

With push() method, you can push the records into the store.

Chapter-7

EMBER.JS OBJECT MODELLING

Object modeling is analysis and designing techniques of object-oriented. Ember.object provides all the objects in the Ember.js. By using the class system, various features are supported by the Ember.Object likewise mixins and constructor method. To observe the changes in the array, Ember.Enumerable interface is used by the ember to extend the array prototype of JavaScript. It extends the String model, formatting and localization methods are used.

The following table contains the list of different type of object model along with their description:

Object Model	Description
Class and Instances	Class is a prototype that creates variables and methods which are common for all the objects. Instances are interrelated to the objects of the class. With the help of Ember.Object's extend() method, the new ember class can be created.
Reopening classes and instance	You can update the implementation of the class without even redefine it.
Computed Properties	It helps in declaring the functions as properties. Computed properties are automatically called by Ember.js whenever it is needed and combines the properties either one or more into one variable.
Computed Properties and Aggregate Data	All the items of array are accessed by the computer properties to calculate its value.
Observers	The observer does nothing just observe the computed property. It regularly updates the text of the computed property.
Bindings	It is one of the interesting features of the ember.js. It ties a link between the two properties if any of the properties are changed, the other property is automatically changing.

7.1 Class and Instances

7.1.1 Define class

extend() method is used to define a class which is imported from EmberObject.

```
Import EmberObject from '@ember/object';
const Name = EmberObject.extend({
say(thing){
alert(thing);
}});
```

say() is defined name class.

7.1.2 Overriding parent class method

You can override the methods of the parent class in the subclass but still you can access the methods of the parent class with the help of the _super() method. Sometime you need to pass an argument to the _super() method before or after overriding.

7.1.3 Define Instances

create() methods is used to define the class instance once the class is defined. Any property, methods and computer properties defined inside a class is visible in the instance.

```
Import EmberObject from '@ember/object';
const Name = EmberObject.extend({
say(thing){
alert(thing);
}});
Let name=Name.create();

name.say('TutorialLinks');
```

7.2 Reopening classes and instances

It is not essential to define a property at one time. You can reopen the class with the help of the reopen() method and define the properties.

```
Name.reopen({

isName:true
```

```
});
```

```
Name.create().get('isName');
```

With the help of reopen method() you can override the existing method and also can call this._super(). If you want to add static methods and properties then you need to open a class with the help of reopenClass() method.

7.3 Computed Properties

Computed properties allow you to declare function as properties. It is automatically called when you call the properties of the of the class. It is super fine as it takes two or more properties and combine them into one property.

```
import EmberObject, {computed} from '@ember/object';
Name = EmberObject.extend({
        f_name: null,
        l_name: null,
        Fullname: computed('f_name', 'l_name', function() {
                return ${this.f_name} ${this.l_name};
})
});
let n=Name.create({
        f_name: 'TutorialLinks',
        l_name: '.com'
});
```

Chapter-8

Concerns of Application

Ember.Application class is used to declare and configures the object which is used for developing the application. The manages the state of the application Ember.ApplicationInstance class is used.

The table consists of a more detailed description of models.

Model Ways	Description
Dependency Injection	It supplies the dependencies from one object to another. It is used by the ember application to declare the object and also dependencies classes between them.
Initializers	As it boots, it is used to configure an application.
Services	It is an object of an Ember which is available in the different parts of the application.
Run Loop	A place where we keep the code of the application.

8.1 Dependencies Injection

Dependencies Injection is used in application to design and instantized class objects and dependences between them. In the implementation of dependency injection both application and application instance are giving out their role. Application is set out as a “registry” for dependency declaration whereas application instance is set out as a “owner” for objects are the initialized from registered factories.

8.2 Initializers

It serves an opportunity for you to configure your application as its boot.

There are two types of Initializer:

- Application Initializer: It runs as an application boot. The serves the fundamental goal to assure the dependency injection configuration in your application.
- Application Instance Initializer: It is running as application instance is loaded. Through this you can configure the initial stage of your application along with injection dependencies that are local to the application instance.

8.3 Service

Command to create a service:

```
C:\Users\YAMINI SHARMA\first-app>ember generate service add-task
```

It will generate the following file:

add-task.js

```
app > services > add-task.js > ...
1   import Service from '@ember/service';
2
3   export default class AddTaskService extends Service {
4   }
5
```

8.4 Run Loop

It is used to organize the work in a way that is the utter most effective and efficient.

The order in which the queues are processed is as follow:

- actions.
- routerTransitions
- render
- afterRender
- destroy

This is the priority order that is followed by the run Loop.

The working of these queue is as follow:

- actions: It is the general work queue. It contains the schedule tasks i.e to be performed.
- routerTransition: It contains the jobs in the transition router.
- render: It contains the rendering jobs and updating the DOM.
- afterRender: It contains those that needs to be executed once all the above rendering task is completed.
- destroy: It contains those jobs that are scheduled for destroy.

Chapter-9

Configure Ember.js

The ways to configure the Ember.js for managing the environment of an application:

Ways	Description
Configure Ember CLI and app	To manage the application's environment, you can configure the Ember CLI and app.
Disabling the extension of prototype and specifying URL type	You can disable the extension of the prototype by setting the EXTEND-PROTOTYPES flag to false and the ember route option is used for specifying the URL type.
Embed application and feature flags	Applications can be embedded into the existing page by just changing the root element. With the project's configuration, feature flags can be enabled.

9.1 Configure Ember CLI and App

Ember CLI already have default environment configure file with the name config/environment. For each environment you can define an ENV object.

There are three important factors that ENV contains:

- EmberENV can be used to declare ember feature flags.
- APP is providing to pass it to options to the application instance.
- Environment is containing the current providing environment whether is development, test and production.

You can also configure the ember CLI. This is done only when you add the .ember-cli to your application root. For permanent configuration, edit your .ember-cli as follow:

```
{
    "port" : 8080
}
```

9.2 Disabling the extension of prototype and specifying URL type

There are four ways to manage the application in ember.js:

- history: it uses HTML5 API of history.
- hash: it uses the URL of anchor-based.
- auto: it uses the history if it is supported by the user's browser otherwise it will return back to the hash.
- none: it never updates the URL.

By default, ember-cli configured in auto. You can change the configuration in ENV file in locationType.

9.3 Embed application and feature flags

This is a new added feature. The flag codes are enabled only based on your project configuration. @ember/canary-features controlled the build generated by the flag status.

There are three types of flag in features:

- true: If the feature is present and enabled**,** the flag code is always enabled in the generated build.
- null: If the feature is present and disabled, the flag code is enabled only during the runtime.
- false: If the feature is entirely disabled, the flag code is not available in the generated build.

Chapter-10

Ember-inspector

It is used to debug the application. It is just a browser add-on. To use the ember-inspector, you need to install the inspector separately.

Tables consists the following topics cover in the inspector along with their description:

Ember inspector way	Description
Install an inspector	To debug the application, install the inspector.
Object inspector	It interacts with the ember objects.
View tree	It gives you a current state of the application.
Inspecting Routes, Data Tab, and library info.	A list of application routes is visible to you that are defined by the inspector whereas the data tab is used to display the list of models used in the application.
Object inspection and evaluate performance	An Object instance is inspected by the container. With the help of render performance option, you can calculate the render performance time.
Promises for debugging	The promises of the inspector are based on their states.

You can install the ember inspector in Google Chrome, Firefox, other browser and also in mobile phone.

- You can install it as a developer tool and add the extension on browser.
- Once the installation is completed, click on the rightmost tab i.e ember.
- All access to the file URL to use the URL file//: protocol.
- Visit chrome//:extension and click on the option. After that make sure “Display the tomster” checked box is checked.