



# Getting Started with Angular 1 Projects

This guide prepares you to work on Angular 1 projects. Follow these steps every time for a consistent and reliable workflow.

## Install NodeJS

Open [this link](#) and follow the instructions to learn more about NodeJS and NPM. This step is only necessary one time. If you already have NodeJS installed, skip this step.

## Clone the Starter Project

To speed up the process of setup, Bloc has created a starter project on GitHub. This repo includes common application files, configuration files, and a NodeJS server file (`server.js`). These files must be downloaded locally and reconfigured to the specifics of your application. Starting in the directory where you want your project to live, follow these steps:

1. Clone the Bloc starter project repo and give it a new name. Change `project-name` to the application name that you want.

```
$ git clone -b version-0.0.3-without-grunt  
https://github.com/Bloc/bloc-frontend-project-  
starter.git project-name
```

2. Change into the app directory: `$ cd project-name`
3. Run `$ npm install` to download and install the needed dependencies
4. Start the node server: `npm start`
5. Open a web browser and go to `http://localhost:3000`

Remember: Press CTRL + C to stop the server.

## Reset Git and Push To Github

This project was downloaded from Bloc's Github repository. Cloning it also clones the git history. All we want from this repo are the files and structure not the history. To start fresh, we will delete the current git history and start over.

1. Remove the existing Git local repository and history: `$ rm -rf .git`
2. Reinitialize Git: `$ git init`
3. Re-stage your existing files: `$ git add .`
4. Create an initial commit of your existing files: `$ git commit -m 'initial commit'`
5. Create a new GitHub repository at [GitHub](#)
6. Add that remote to your project: `$ git remote add origin <link to your github repository>`
7. Push your initial commit to your new repository: `$ git push origin master`

## Add the Project to GitHub

1. Create a new GitHub repository at <http://www.github.com>
2. **DO NOT** add a README to your repo on GitHub
3. Add that remote to your project: `$ git remote add origin <link to your github repository>`
4. Now push your code to GitHub `$ git push -u origin master`

# Add Angular to the Project

For simplicity's sake we are going to use a reference to include Angular into our application. If you are building a capstone or other large project, consult with your mentor on the best methods for including Angular in your project.

In your editor, open `app/index.html` and add the Angular reference to the body of the document. Make sure that it is referenced above the `app.js` reference. The `app.js` reference will be dependent on the Angular library and must come after (below) the Angular reference.

Optional: Add a comment title before and an empty line between your references to outside libraries and references to internal files. This may seem like a simple formatting detail, but this extra effort will make it easier for your eyes to quickly scan the list of references.

app/index.html

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title>Bloc Base Project</title>
  <link rel="stylesheet" href="/styles/main.css">
</head>
<body>
+   <!-- Angular -->
+   <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/

+   <!-- Application Files -->
      <script src="/scripts/app.js"></script>
</body>
</html>
```

## Include AngularUI Router

**UI-Router** is more flexible and features behaviors not found in the Angular default router. With UI-Router, an application can be in different **states** that determine what to display when a user navigates to a specific route. In the file `app/index.html`, add the AngularUI Router reference to the body of the page.

app/index.html

```
<body>
  <!-- Angular -->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/
+
+ <!-- AngularUI Router -->
+ <script src="https://cdnjs.cloudflare.com/ajax/libs/
router/0.4.2/angular-ui-router.min.js"></script>

  <!-- Application Files -->
  <script src="/scripts/app.js"></script>
</body>
```

UI-Router will take care of replacing the contents of `<ui-view></ui-view>` with a template when a user navigates to the proper route. Each template can be unique, while the shared code is kept in the global file. Since UI-Router uses JavaScript to switch the views, the browser won't load a new HTML document when a user navigates to a new route. Add the `<ui-view></ui-view>` directive element to your index file.

app/index.html

```

</body>
+   <ui-view></ui-view>

    <!-- Angular -->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/

    <!-- AngularUI Router -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/
router/0.4.2/angular-ui-router.min.js"></script>

    <!-- Application Files -->
    <script src="/scripts/app.js"></script>
</body>

```

## Include AngularFire

**Optional:** Make sure that you are going to use Firebase before adding AngularFire to your project. If you are unsure, check with your mentor. If you are not, skip ahead to "Bootstrapping Angular"

**Firebase** is freemium storage solution for adding a web application backend that saves and syncs data with minimal effort. It has Angular modules that make including it similar to using built-in Angular services and modules. There's no need to deeply understand the architecture of a web application backend with Firebase now - the models created with Firebase are analogous to Angular's models.

AngularFire is the officially supported AngularJS binding for Firebase. The combination of Angular and Firebase provides a three-way data binding between your HTML, your JavaScript, and the Firebase database.

- Create an Account

The first thing we need to do is sign up for a free Firebase account. A brand new Firebase project will automatically be created for you which you will use in conjunction with AngularFire to authenticate users and store and sync data.

- Add Script Dependencies

In order to use AngularFire and Firebase in a project, include the following script tags at the bottom of the body of your index.html file:

app/index.html

```
<body>
  <ui-view></ui-view>

  <!-- Angular -->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/

  <!-- AngularUI Router -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/
router/0.4.2/angular-ui-router.min.js"></script>

+   <!-- FireBase -->
+   <script src="https://www.gstatic.com/firebasejs/3.9.
+   <!-- AngularFire -->
+   <script
src="https://cdn.firebase.com/libs/angularfire/2.3.0/angular

  <!-- Application Files -->
  <script src="/scripts/app.js"></script>
</body>
```

- Initialize the Firebase SDK
  - Go to <https://console.firebase.google.com/>
  - Create a new project
  - Name the project and choose your country/region
  - From the Overview page, click the button to 'Add Firebase to your web app'.
  - That will popup a message with some pre-filled JS code that looks like the below code snippet. Copy/paste that snippet into bottom of the body tag in your index.html file:

app/index.html

```

<body>
  <ui-view></ui-view>

  <!-- Angular -->
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/

  <!-- AngularUI Router -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/
router/0.4.2/angular-ui-router.min.js"></script>

  <!-- FireBase -->
  <script src="https://www.gstatic.com/firebasejs/3.9.
  <!-- AngularFire -->
  <script
src="https://cdn.firebase.com/libs/angularfire/2.3.0/angulara
+   <script>
+     // Initialize Firebase
+     var config = {
+       apiKey: <your unique api key>,
+       authDomain: <your unique auth url>,
+       databaseURL: <your unique db url>,
+       storageBucket: <your unique bucket url>,
+       messagingSenderId: <your unique messagingSenderId>
+     };
+     firebase.initializeApp(config);
+   </script>

  <!-- Application Files -->
  <script src="/scripts/app.js"></script>
</body>

```

## Set Firebase read/write to true

Firebase defaults to only allowing users that have logged into your application to read or write to the database. We don't really need that level of security right now. So lets set Firebase to allow anyone to read/write.

1. Go to <https://console.firebase.google.com>

2. Choose the bloc-chat application
3. Click on 'Database' link in the sidebar
4. Click the 'Rules' tab from the navigation tabs
5. Set the read and write rules to true:

```
console.firebase.google.com/project//database/rules
```

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Additional information about connecting to Firebase via Angular can be found at the [AngularFire Quick Start guide](#) or at the [Firebase documentation](#).

## Bootstrapping Angular

Add the `ngApp` directive to the `<html>` tag to initialize the Angular application in the index. Remember to use a camelCase format for the application name. We used "projectName" here as a demonstration. You should replace that with your project's name.

```
app/index.html
```

```
<!DOCTYPE html>
+ <html ng-app="projectName">
  <head lang="en">
    ...
```

In `app.js`, use an **IIFE** to build the structure for your JavaScript code. Invoke `angular` and connect it to your application's name with the `module` method.

```
app/scripts/app.js
```



```
+ (function() {  
+     angular  
+         .module('projectName', [])  
+     })();
```

Include `ui.router` and `firebase` as dependencies.

app/scripts/app.js

```
    (function() {  
        angular  
+        .module('projectName', ['ui.router',  
'firebase'])  
    })();
```

Add a configuration function and pass it to the angular `config` method and **inject** the `$stateProvider` and `$locationProvider`. Use the providers to define your first state which will hold the main view for the application.

app/scripts/app.js

```

    (function() {
+       function config($locationProvider, $stateProvider)
    {
+         $locationProvider
+         .html5Mode({
+           enabled: true,
+           requireBase: false
+         });
+
+         $stateProvider
+         .state('home', {
+           url: '/',
+           controller: 'HomeCtrl as home',
+           templateUrl: '/templates/home.html'
+         });
+       }
+
      angular
        module('projectName', ['ui.router',
+          'firebase'])
+        .config(config);
    })();

```

- [ui-routers's GitHub Page](#) has examples using `ui-router` in other contexts, along with an [API Reference](#), [resources](#), and a [proprietary guide](#) for additional exposure.

Several lines of code were added in the last few steps, with little explanation. If you want to refresh your understanding of what this routing code does, refer back to the BlocJams Angular project checkpoints for detailed explanations.

## Create a Controller

Create a Home controller. You will have to create the `controllers` directory.

```
app/scripts/controllers/HomeCtrl.js
```

```
(function() {  
    function HomeCtrl() {  
    }  
  
    angular  
        .module('blocChat')  
        .controller('HomeCtrl', [HomeCtrl]);  
})();
```

## Create a Template

The last file we need to add is a template for the "home" state. Create this template `app/templates/home.html`. The templates directory may not exist. If it doesn't, then create the directory first. Add some html to the template so you can test visually whether or not it is working.

`app/templates/home.html`

```
+ <h1>Home Page</h1>
```

That is all the markup you need for the template to start. You just want enough to allow for manual testing. Angular will replace the `<ui-view></ui-view>` directive element in the `index.html` file with the html you put in the template, creating a full valid html file.

## Test your App

1. Start the node server: `$ npm start`
2. Open a web browser and go to `http://localhost:3000`

You should see the words "Home Page" in large letters in your browser. Check the Chrome inspector for any errors in the JavaScript console. If the text shows and there are no errors, you have succeeded. If you don't see those words or there are errors, go back through the steps and make sure you completed each step. If you after checking, you believe all the steps have been accurately completed yet the app still does not work, then it's time to get some

help. Ask your mentor or post a question on Slack.

## Finish with Git

You have made several more changes to your project since your last commit. Add and commit the remaining changes before you start adding any new features. You are still on the `master` branch which is fine for the initial setup. However, once you start adding features, make sure you're always creating a feature branch and adding any changes to it.

1. Stage all the files in your project `$ git add .`
2. Commit all the files with a message `$ git commit -m "Finished basic setup"`
3. Push your work to GitHub `$ git push origin master`

You are now ready to begin creating features. Check with your mentor and clarify what git workflow they would prefer you to be using. Good luck and happy hacking!

[Submission](#)[Next→](#)

# 1 Introduction

## Overview and Purpose

In this project you'll use Firebase to build an application that sends and receives messages in real time.

## Objectives

After this project, you should be able to:

- Register Firebase as a module in an Angular application.
- Inject the `$firebaseArray` service into a controller.
- Understand and use the Firebase JavaScript and AngularFire APIs – methods such as `child()` and `$add()`.
- Query a Firebase array.
- Use UI Bootstrap to create a modal.
- Use cookies to store information in the user's web browser.

## Use Case

Bloc Chat uses Firebase and AngularJS to create a real-time chat application.



## User Stories

User Story	Difficulty Rating
As a user, I want to see a list of <b>available chat rooms</b>	3
As a user, I want to <b>create</b> chat rooms	3
As a user, I want to see a <b>list of messages</b> in each chat room	3
As a user, I want to set my <b>username</b> to display in chat rooms	2
As a user, I want to <b>send messages</b> associated with my username in a chat room	2

Before you begin working on user stories, complete this project's **Getting Started guide**. Later user stories often rely on the completion of the former, therefore, work on them in the order prescribed.

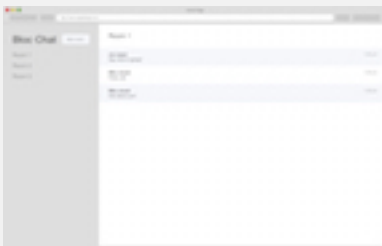
## Wireframes

These wireframes are meant to suggest a design, not dictate one.

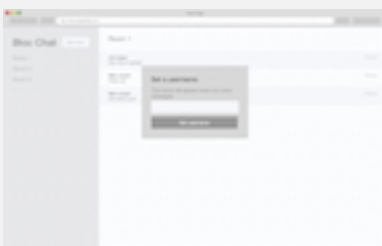
Wireframe	Description
	List of chat rooms.
	New chat room button.



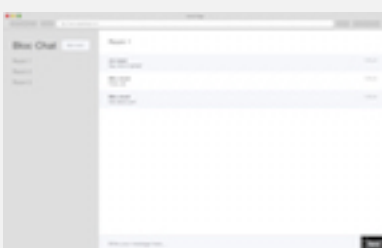
New chat room modal.



Chat room with messages.



Set username modal.



Chat room with the ability to send a new message.

How would you rate this checkpoint and assignment?



1. Introduction

 **Assignment**

 **Discussion**

 **Submission**




## 2 List Chat Rooms

As a user, I want to see a list of **available chat rooms**

**Difficulty Rating:** 3

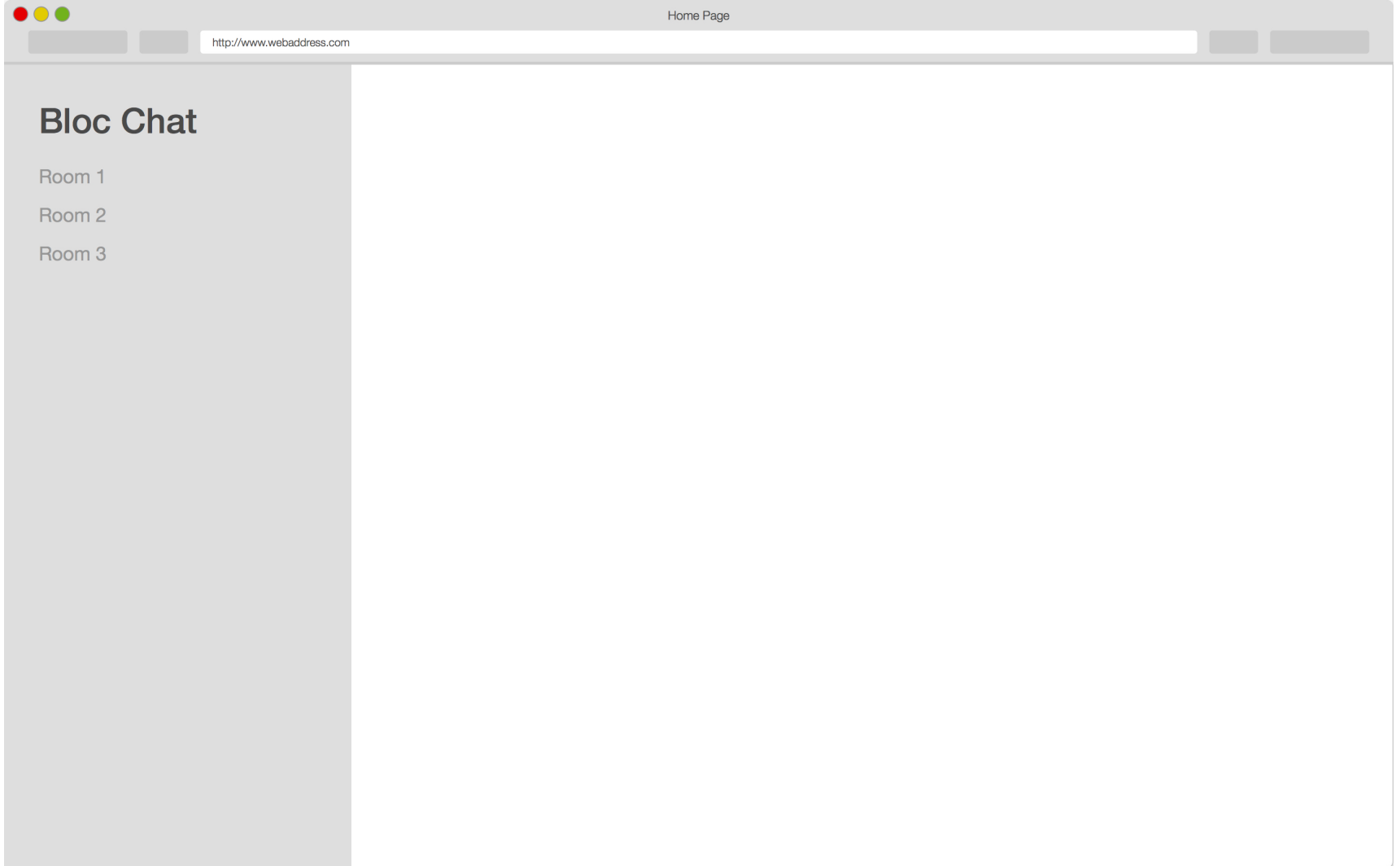
### Create fake 'rooms' in Firebase

1. Log into **firebase**, click the 'Database' link on the sidebar
2. Add a 'rooms' key to the db. Add rooms as the key, but instead of value, click the  to add sub objects under rooms
3. Add a few generic rooms to the database. Put something like name: 'room1', value: 'room1' in as root elements of the db. It should look something like this:

```
bloc-chat-1482f
|__rooms
  |__1:"room1"
  |__2:"room2"
  |__3:"room3"
```

### Create a Factory





How can I query a list of Rooms from Firebase?\*

Create a `Room` factory in Angular that defines all Room-related API queries. Create a reference to your Firebase database inside, and inject the `$firebaseArray` service provided by AngularFire:

```
(function() {  
  function Room($firebaseArray) {  
    var ref = firebase.database().ref();  
  }  
  
  angular  
    .module('blocChat')  
    .factory('Room', ['$firebaseArray', Room]);  
})();
```

The styles we use for Angular differ from the styles used in the Firebase documentation. Follow the style guide we've proposed, as they are more recently accepted by the Angular development community as best practices.

Use Firebase's `child()` **method** (called on an instance of its API object) to either query

an existing set of data or reference one you intend to populate with data in the future. Use the `$firebaseArray` **service** to ensure the data is returned as an array:

```
(function() {  
  function Room($firebaseArray) {  
    var Room = {};  
    var ref = firebase.database().ref().child("rooms");  
    var rooms = $firebaseArray(ref);  
  
    Room.all = rooms;  
  
    return Room;  
  }  
  
  angular  
    .module('blocChat')  
    .factory('Room', ['$firebaseArray', Room]);  
})();
```

Remember to add a html `<script>` tag with a src link to `Room.js` from your `index.html` file!

## Create a Controller

How can I display my queried Rooms in the view?\*




Create a controller and associate it with the home template in a `$state`. Inject the Room service so that you can assign the array of objects retrieved by the `all` method to a `$scope` variable. Display the rooms in the template using `ng-repeat`.

## Test Your Code

- Launch Bloc Chat, verify that empty chat rooms appear.

How would you rate this checkpoint and assignment?



2. List Chat Rooms		
 Assignment	 Discussion	 Submission



# 3 Create Chat Rooms

As a user, I want to create chat rooms

**Difficulty Rating:** 3

## Modify the Factory

How do I create Room objects with AngularFire?\*

You can call **AngularFire's** `$add()` on any array created or retrieved with the `$firebaseArray` service. In this case, the array is the data stored in the `rooms` variable in the `Room` service. Use the AngularFire method `$add()` inside a `Room` factory method `add`. This will give the application the ability to add rooms to the firebase database. You don't want AngularFire specifics leaking into the rest of our application. To avoid this leaking, we will create an abstract method in the service. This may seem like unnecessary duplication of another `add` method, but this purposeful abstraction will create a nice barrier between AngularFire and the application's controllers. This is the main purpose for creating this service and a foundation for decoupled code.

```

(function() {
  function Room($firebaseArray) {
    var Room = {};
    var ref = firebase.database().ref().child("rooms");
    var rooms = $firebaseArray(ref);

    Room.all = rooms;

    Room.add = function(room) {
      //Use the firebase method $add here
    }

    return Room;
  }

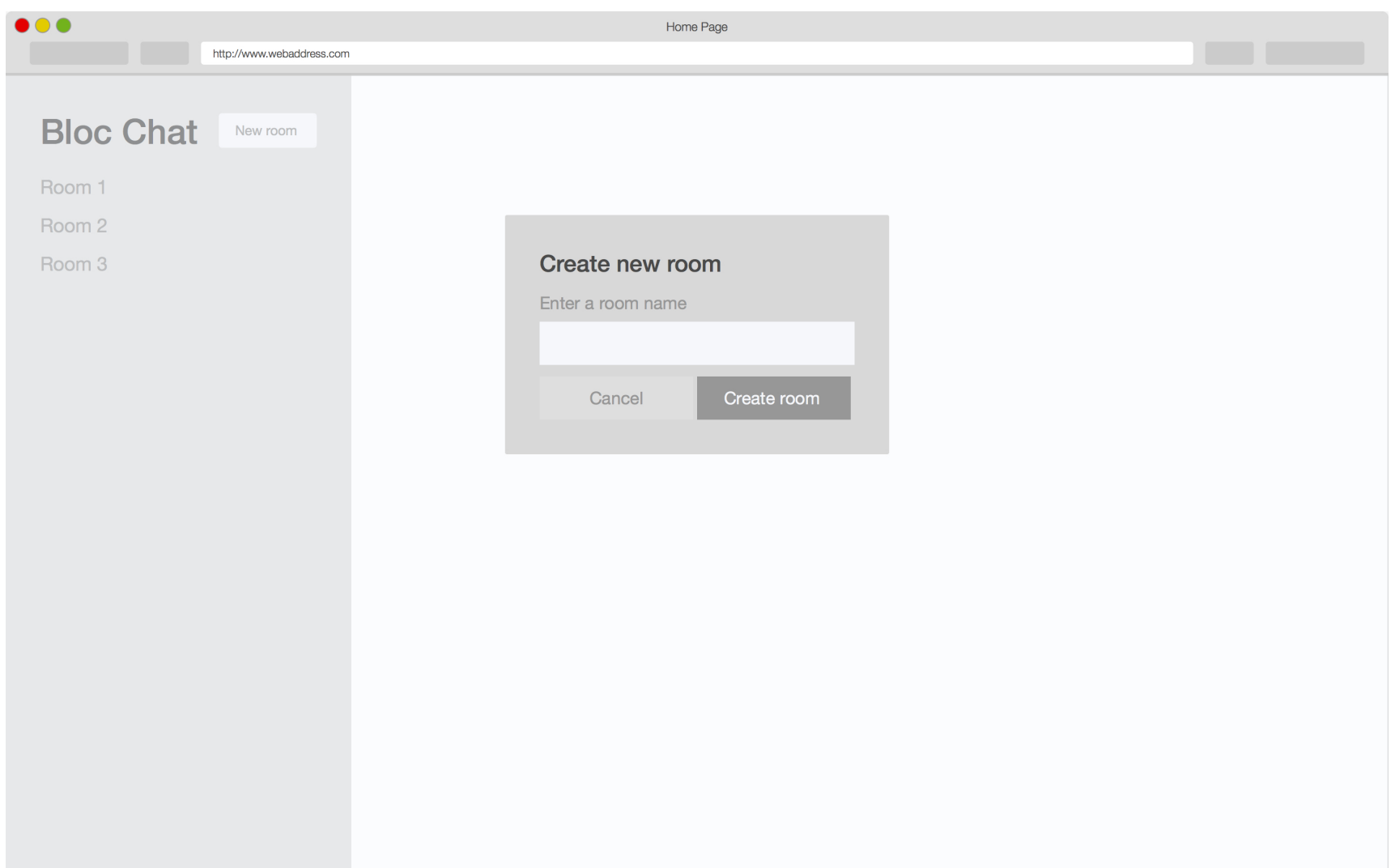
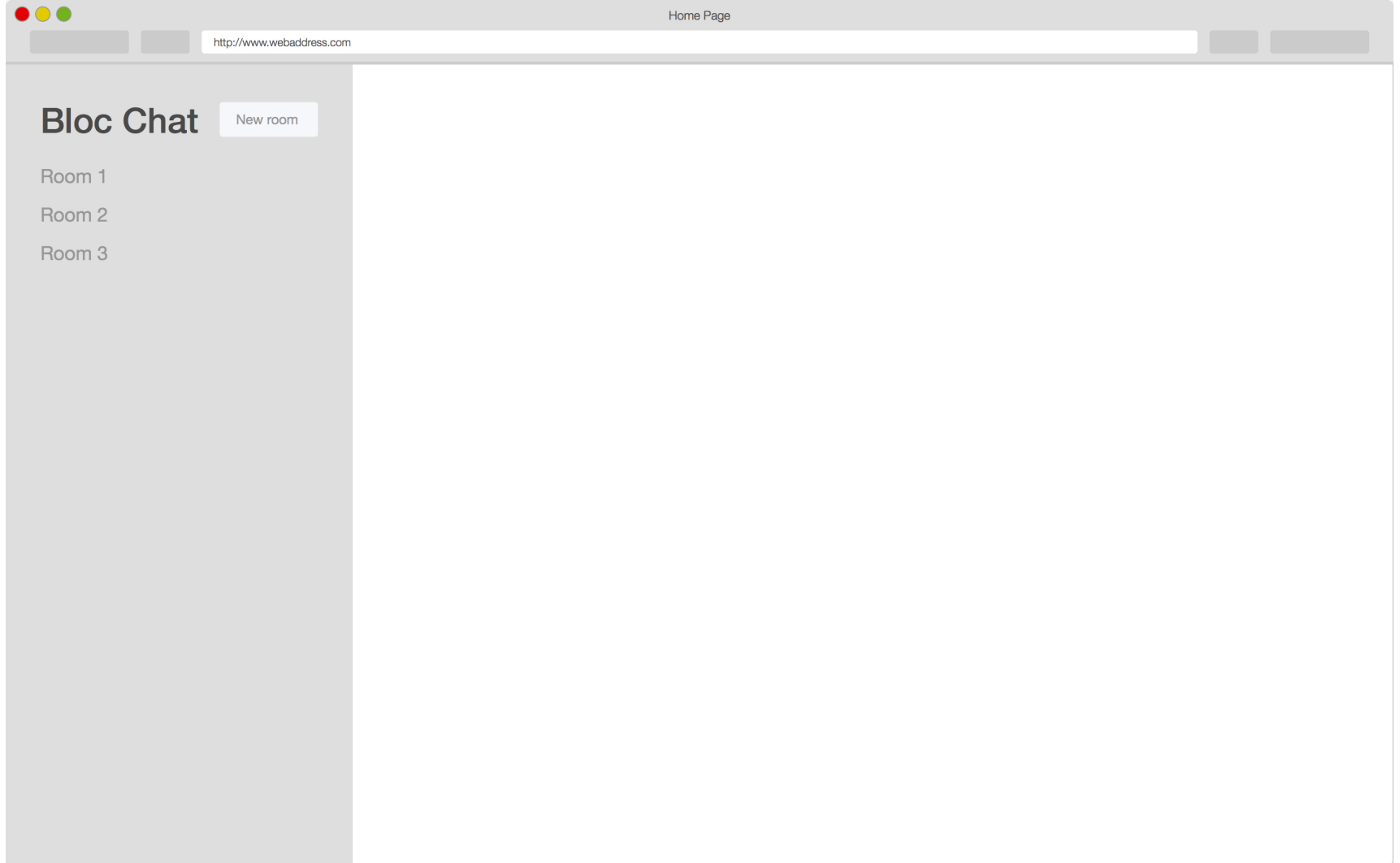
  angular
    .module('blocChat')
    .factory('Room', ['$firebaseArray', Room]);
})();

```

The `Room` service method `add` should take a `room` object as an argument. This room object will need to be created outside of this service. Where could you create the room object? What are the pros and cons of where you create that object? Do some research and experimentation on this topic. Discuss your findings and questions on Slack and with your mentor.

## Include Bootstrap

How can I initiate room creation in the app's interface?\*



There needs to be a form that you can use to submit the new room's data using `ngClick` or `ngSubmit`. Presenting a modal is an unobtrusive way to trigger a form on the interface. Use **UI Bootstrap's** `$uibModal` **service** to define a method for toggling a modal on the frontend. To fully integrate a UI Bootstrap modal:

1. Include the **UI Bootstrap library** via a `<script>` tag on `index.html`. We will be using version 2.5.0.

~/bloc/bloc-chat/app/index.html

```
...  
+  
+ <!-- AngularUI Bootstrap -->  
+ <script src="https://cdnjs.cloudflare.com/ajax/libs/angular-ui-bootstrap/2.5.0/  
...
```

2. Inject the module into your Angular app's dependency array
3. Create a separate controller for the modal
4. Inject the proper dependencies for using the modal (see the **UI Bootstrap** documentation)
5. Add methods to open, close and submit data to Firebase from the modal

When you've finished, you should see your array of rooms update in real time.

## Test Your Code

- Launch Bloc Chat, verify that your array of rooms updates in real time as soon as you create one.

### 3. Create Chat Rooms

 **Assignment**

 Discussion

 Submission

[< Prev](#)[Submission](#)[Next >](#)

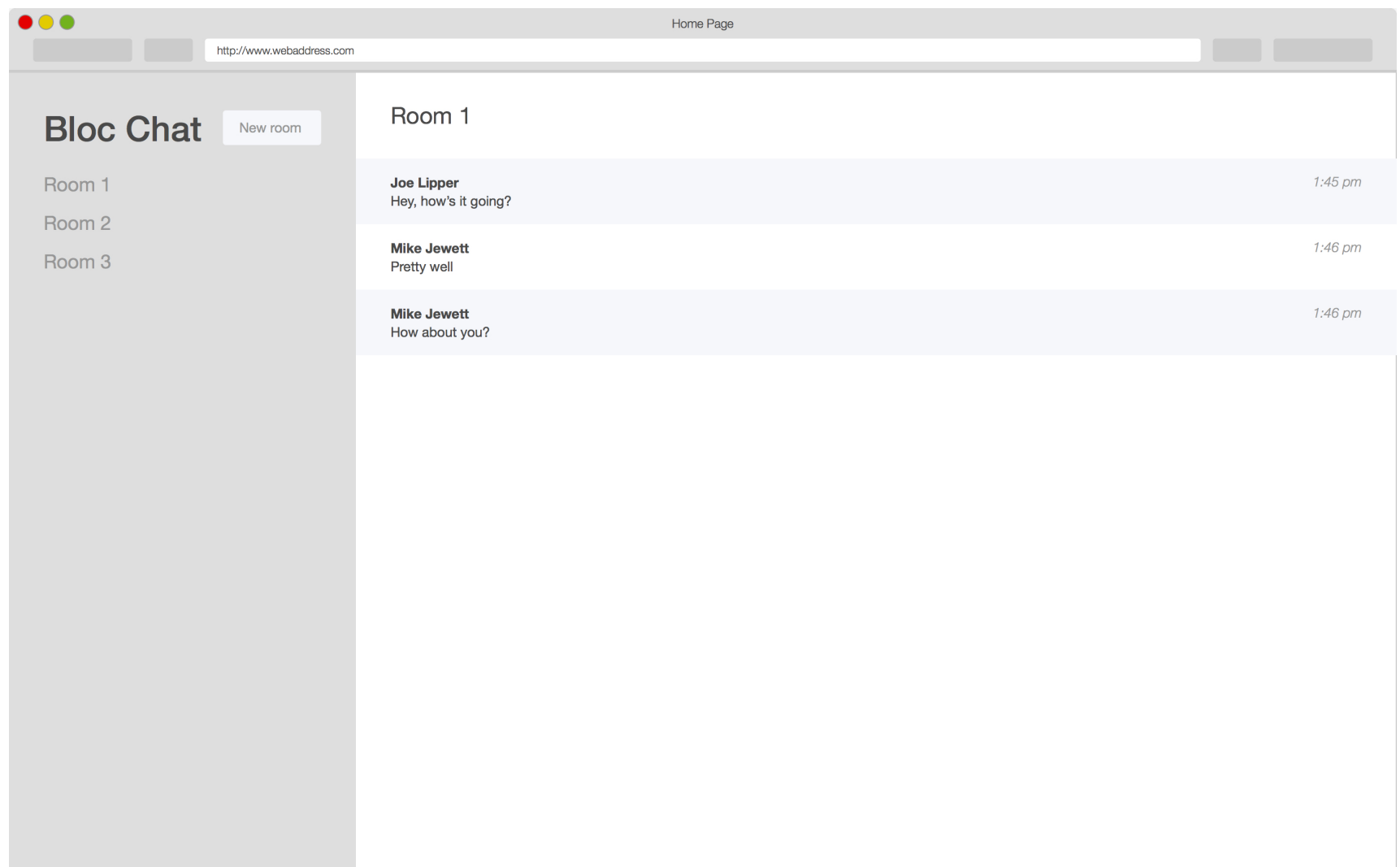
# 4 List Messages

As a user, I want to see a **list of messages** in each chat room

**Difficulty Rating:** 3

## Create a Container

How can I display an individual room's content?\*



Chat rooms tend to be designed so that only one room's messages show at a time. Create and style a container for holding a list of messages to the right of the list of available chat rooms. The active room should be stored in a `$scope` object in the main controller, so that



the title of the active room changes every time you visit a different room. The active room should be triggered by clicking on the name of the room in the sidebar.

## Associate Messages With a Room

How can I associate messages with a room so that only an active room's messages show when I've selected it?\*

Associating objects with other related objects, like rooms with messages, requires using a reference to the parent data (in this case, the room) in the child data. When creating a message object, each message object in your Firebase database should have four properties:

```
{
  username: "<USERNAME HERE>",
  content: "<CONTENT OF THE MESSAGE HERE>",
  sentAt: "<TIME MESSAGE WAS SENT HERE>",
  roomId: "<ROOM UID HERE>"
}
```

The last property, `roomId`, references the room where the message was sent. The ID is generated every time an object saves to Firebase, and can be viewed on the Firebase web interface.

If you haven't already, create a few rooms using the method you programmed in the last story so that you can use the IDs generated from those rooms to complete this story. Firebase UUIDs (or unique identifiers) are strings of randomly generated characters like `-Jf1GqAZWtS94xlfZA4a`.

**Create a few messages manually** on the Firebase dashboard with the above data structure, and associate them all with one of your rooms so you can test querying messages with the rooms.

For help with structuring data, read Firebase's **guide on the subject**.

## Query Messages with a Factory

Create a `Message` factory in Angular that defines all Message-related API queries. Create a reference to your Firebase database inside, and inject the `$firebaseArray` service provided by AngularFire:

```

(function() {
  function Message($firebaseArray) {
    var Message = {};
    var ref = firebase.database().ref().child("messages");

    return Message;
  }

  angular
    .module('blocChat')
    .factory('Message', ['$firebaseArray', Message]);
})();

```

How can I query messages for an active room?\*

Using the `child()` method on the `$firebaseArray` service again, query `messages` instead of `rooms` this time. To get the messages for a given room, you need to chain the `child()` method with Firebase's `orderByChild()`<sup>1</sup> method, targeting the `roomId` child.

Recall that `roomId` is a nested property of each message object. A nested property in Firebase is equivalent to a child, hence its compatibility with the `orderByChild()` method.

Messages depend on the ID of a room, you will need to pass an argument into the `getByRoomId` method that contains the `roomId` associated with a rooms message. With the `roomId`, use Firebase's `equalTo()` method to find all messages whose `roomId` property is equal to the `roomId` in the argument:

```

(function() {
  function Message($firebaseArray) {
    var Message = {};
    var ref = firebase.database().ref().child("messages");
    var messages = $firebaseArray(ref);

    Message.getByRoomId = function(roomId) {
      // Filter the messages by their room ID.
    };

    return Message;
  };
}

angular
  .module('blocChat')
  .factory('Message', ['$firebaseArray', Message]);
})();

```

Read how `orderByChild()` and `equalTo()` are used [in this StackOverflow question](#) to get an idea of how to chain them to query the proper messages by a `roomId`.

## Test Your Code

- Launch Bloc Chat.
  - Verify that messages appear when selecting a conversation.
  - Verify that switching chat rooms replaces the messages with those associated with the new chat room.

## Footnotes

1. The event listener (`.on('<event name>')`) chained to the `orderByChild()` method is not required because messages will always be ordered by `roomId` in this application.

How would you rate this checkpoint and assignment?



4. List Messages

 Assignment

 Discussion

 Submission

[< Prev](#)[Submission](#)[Next >](#)

# 5 Set Username

As a user, I want to set my **username** to display in chat rooms

**Difficulty Rating:** 2

## Use Cookies

How can I efficiently store a username?\*

A username is a string identifying a user. A common way to store a string in your browser is to use cookies. Angular has an external module for including the services and methods associated with cookies. To integrate the module:

1. Include the Angular cookies module via a `<script>` tag in `index.html`:  
`https://ajax.googleapis.com/ajax/libs/angularjs/X.Y.Z/angular-cookies.js`, where X.Y.Z is the AngularJS version you are running.
2. Inject the `ngCookies` **module** into your Angular app's dependency array.

How can I require each user to enter a username when they visit Bloc Chat for the first time?\*

Angular modules have a `.run()` method that runs code when the app instance is created. Use a `.run()` block to make sure that a username is set at the time the app is initialized. You will need to inject the `$cookies` **service** into the **run block's** dependencies to check for the presence of the cookie holding the username:

```

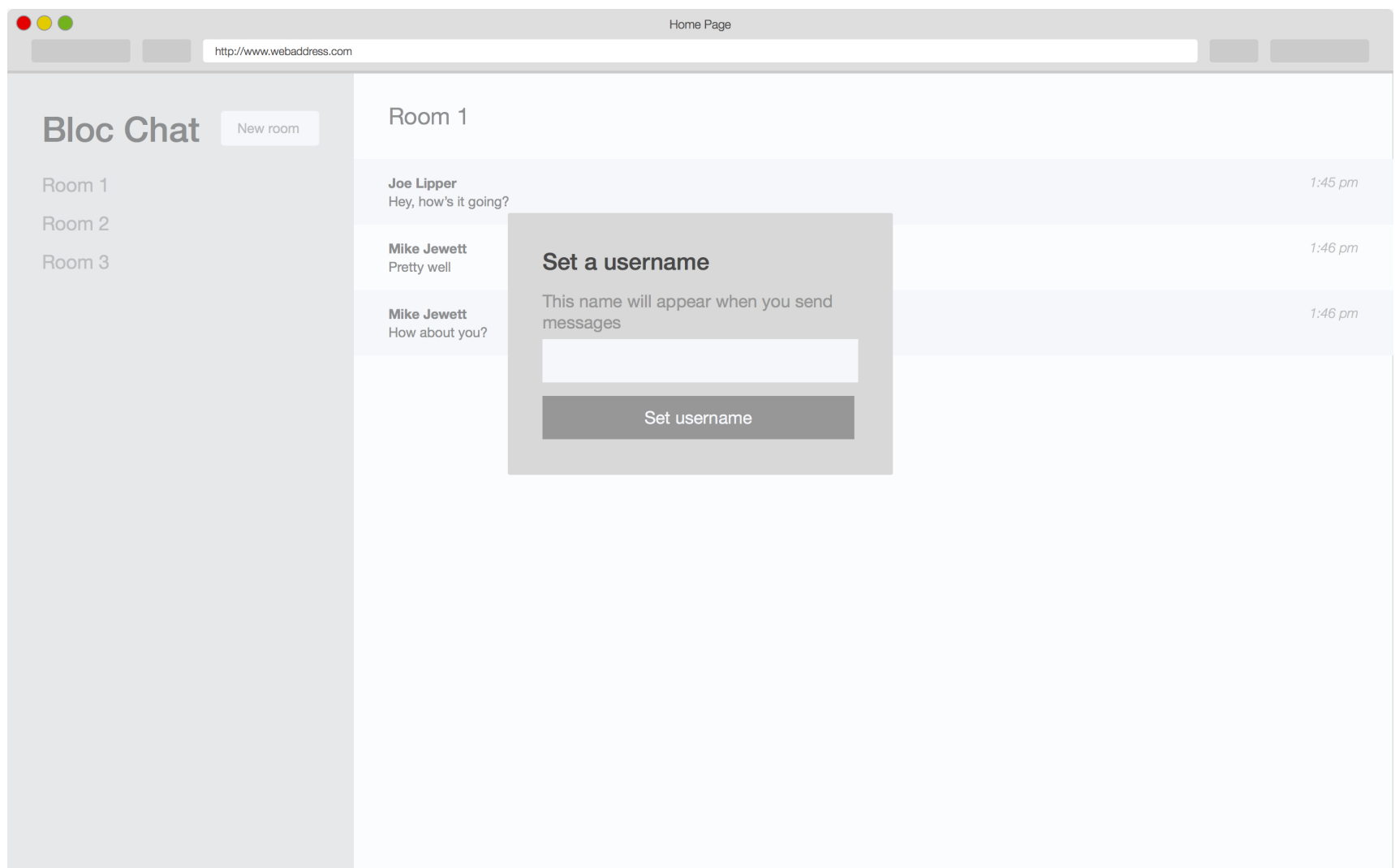
(function() {
  function BlocChatCookies($cookies) {
    var currentUser = $cookies.get('blocChatCurrentUser');
    if (!currentUser || currentUser === '') {
      // Do something to allow users to set their username
    }
  }

  angular
    .module('blocChat')
    .run(['$cookies', BlocChatCookies]);
})();

```

## Prompt the User

How can a user enter a username?\*



If the app detects that a username isn't present, there needs to be a way to enter one. Inside the conditional that checks for the presence of a username, trigger another **UI Bootstrap modal** that requires a user to enter one. Do not provide a “cancel” option this time, so the user cannot access the chat until their username has been set. To create a fully functional modal:

1. Inject the `$uibModal` service in the `.run()` block.
2. Call `$uibModal.open()` and pass in a configuration object.
3. Create a template and a controller for the modal.

```
(function() {  
  function BlocChatCookies($cookies, $uibModal) {  
    var currentUser = $cookies.get('blocChatCurrentUser');  
    if (!currentUser || currentUser === '') {  
      $uibModal.open({  
        // Modal configuration object properties  
      })  
    }  
  }  
}  
  
angular  
  .module('blocChat')  
  .run(['$cookies', '$uibModal', BlocChatCookies]);  
})();
```

## Test Your Code

- Launch Bloc Chat.
  - Verify that a modal prompts you for a username.
  - Verify that you cannot dismiss the modal.
  - Verify that submitting an empty username (or whitespace) does not succeed.
  - Verify that providing a username grants access to Bloc Chat.
  - Verify that the username is saved to the appropriate cookie.

How would you rate this checkpoint and assignment?



### 5. Set Username

 Assignment

 Discussion

 Submission

[< Prev](#)

Submission

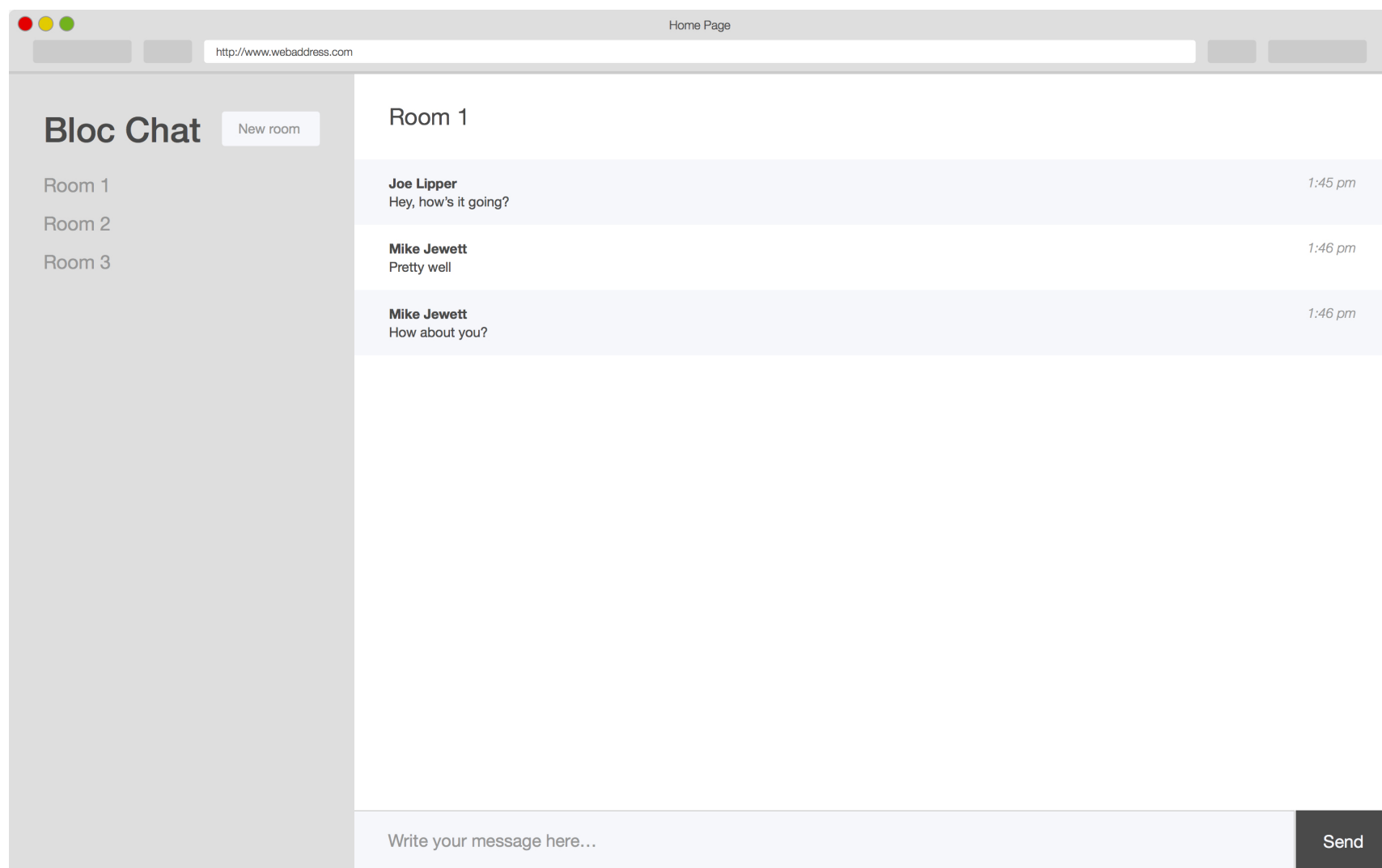
[Next >](#)

# 6 Send Messages

As a user, I want to **send messages** associated with my username in a chat room

**Difficulty Rating:** 2

## Message Factory



How can I send messages?\*

Add a method to your `Message` factory called `send`, that takes a message object as an argument and submits it to your Firebase server:



```

(function() {
  function Message($firebaseArray) {
    var Message = {};
    var ref = firebase.database().ref().child("messages");
    var messages = $firebaseArray(ref);

    Message.getByRoomId = function(roomId) {
      // .. logic for filtering messages
    };

    Message.send = function(newMessage) {
      // Send method logic
    };

    return Message;
  }

  angular
    .module('blocChat')
    .factory('Message', ['$firebaseArray', Message]);
})();

```

Create a controller method that is invoked via `ngClick` or `ngSubmit` on the frontend.

## Associate Messages with Usernames

How can I make sure that the messages that a user sends are associated with their username?<sup>\*</sup>

In the message object detailed earlier, there was a `username` property that held a string referring to the user crafting the message. Populate that property with the current user's username by injecting the `$cookies` service and referencing the current user object on it.

## Test Your Code

- Launch Bloc Chat, open a chat room.
  - Verify that messages are submitted to the active chat room.
  - Verify that your username is associated with each message you create.
  - Verify that new messages are associated with no chat rooms other than the active.

How would you rate this checkpoint and assignment?



6. Send Messages

 Assignment

 Discussion

 Submission