MIRACLE
SOFTWARE SYSTEMS

Creating your first Hybrid Application with Ionic in 60 Minutes

Miracle Summer of Code Virtual Labs Series

Venkatesh Voona
Lead Researcher – Mobile/DevOps
Miracle Software Systems, Inc.

May 11th, 2016

Miracle's SoC Series : Creating your first Hybrid Application using Ionic in 60 Minutes

Overview

In this lab we will be using Cloud9(Web IDE) to create a Hybrid Mobile App using Ionic, an Angular-based Mobile Framework. The app will have a simple form that calls a REST API and shows the results.

Prerequisites

You will need the following to complete this lab successfully,

- Active email ID for registering with Cloud9 (or) a GitHub ID
- Up to date browser to access Cloud9

Note : You will not be required to download (or) install anything on your laptops for this lab, but this lab can be recreated on your laptop instead of using C9 with the exact same steps.

Technology Involved

The following technologies will be covered in this lab,

- Node.js and NPM
- Cloud9(Web IDE)
- Cordova(Above v5.5)
- HTML/CSS/JavaScript
- REST APIs
- Ionic Framework and Angular JS

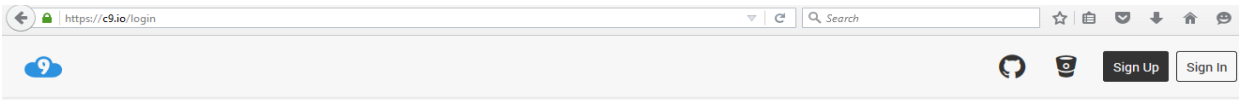
**Basic knowledge on web/mobile applications and using the command line will be beneficial but not entirely required*

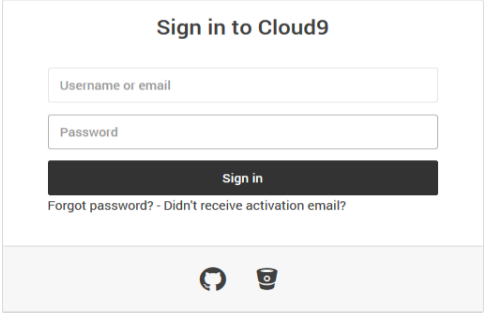
Lab Steps

So, let us get started with the lab!

#1 | Access Cloud9

The first step would be to access Cloud9 and either register (or) login to your account. Login at **https://c9.io/login** (or) register at **https://c9.io/signup**.



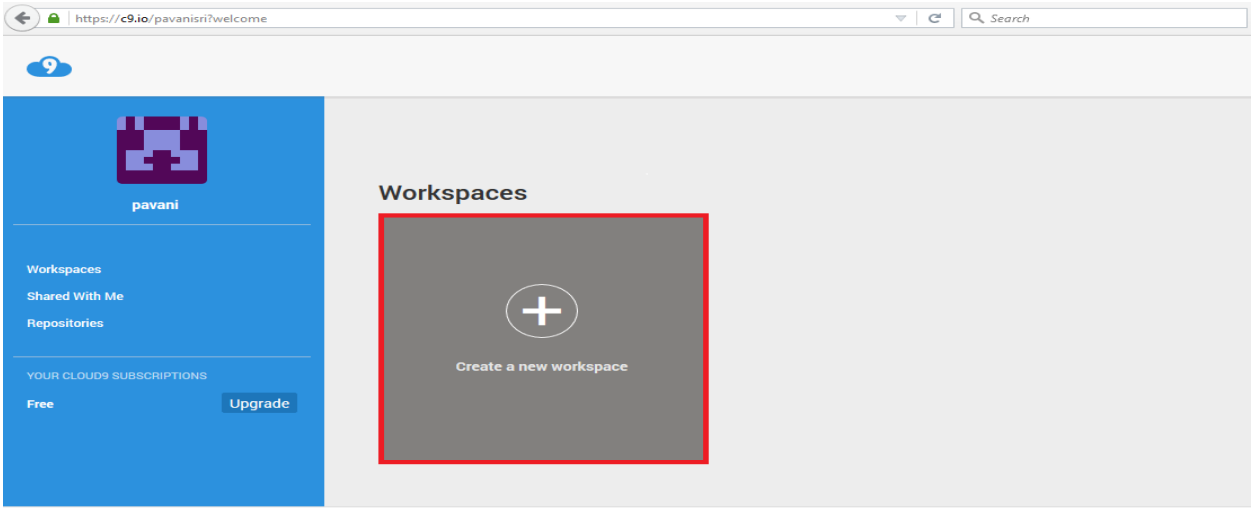


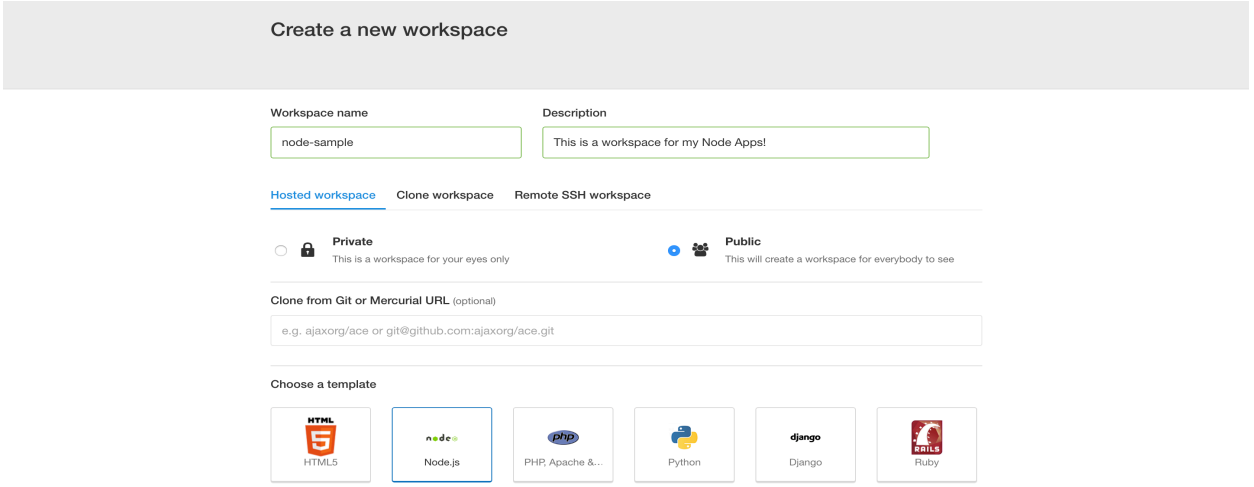
#2 | Create a new workspace

Create a new workspace in Cloud9 by clicking on the new workspace button(As Shown Below). When prompted fill in the following details,

- **Workspace Name** – Define what your workspace will contain
- **Description** – Describe the contents of your workspace
- Under Hosted Workspace, select the **public** option
- **Template** – Choose the Node.js template to get Node/NPM pre-installed

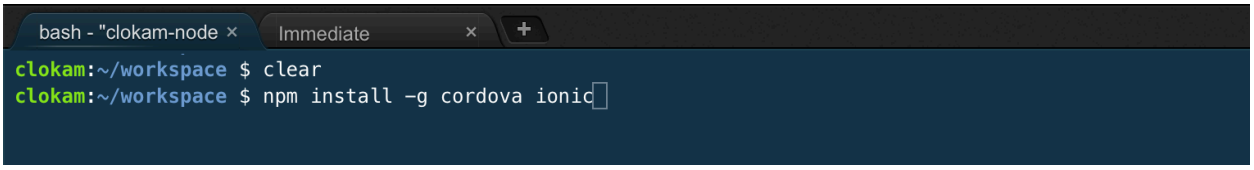
Click on **"Create Workspace"** to create your new workspace and initialize the terminal for you with Node.js and NPM pre-installed. You will be automatically taken to your new workspace.





#3 | Install Ionic using the C9 Terminal (2-3 Minutes)

Open the C9 terminal and Install the Ionic Framework by using the command **npm install -g cordova ionic.** To install Ionic, it will take 2-3 minutes.



#4 | Create a new blank Ionic Application

Create an Ionic project using a blank template to start off with the Lab. When prompted you can choose 'no' for creating an Ionic Account for this lab.

ionic start <application-name> blank

```
bash - "clokam-node ×   Immediate ×
clokam:~/workspace $ ionic start consumerdemo blank
```

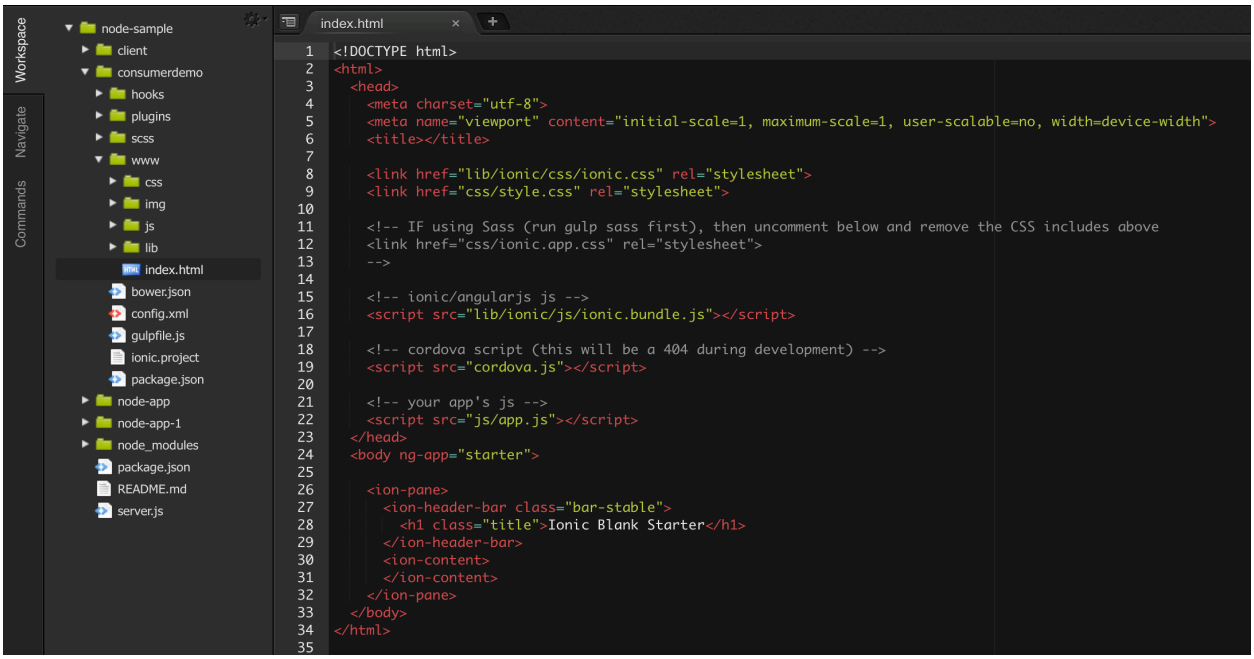
#5 | Get the REST API from the Cheat Sheet

Getting the below URL from the cheat sheet (or) copy from below, (The REST API returns the Geolocation data of a city based on a city name that is passed to it)

https://maps.googleapis.com/maps/api/geocode/json?address="+$scope.cityname+"

#6 | Edit the application to consume the API Endpoint

In the left panel of IDE open **/www/index.html** page,



Copy the following code into the **index.html** page. The page contains a header, two form fields and submit button.

index.html

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="initial-scale=1, maximum-
scale=1, user-scalable=no, width=device-width">
  <title>API Consumer Demo App</title>

  <link href="lib/ionic/css/ionic.css" rel="stylesheet">
  <link href="css/style.css" rel="stylesheet">
  <!-- ionic/angularjs js -->
  <script src="lib/ionic/js/ionic.bundle.js"></script>
  <!-- cordova script (this will be a 404 during development) --
>
  <script src="cordova.js"></script>

  <!-- your app's js -->
  <script src="js/app.js"></script>
</head>

<body ng-app="starter" ng-controller="BodyCtrl">
  <ion-pane>
    <ion-header-bar class="bar-calm">
      <h1 class="title"><strong>I am an API
Consumer</strong></h1>
    </ion-header-bar>
    <ion-content class="had-header has-footer">
      <div class="list">
        <label class="item item-input">
          <input type="text" id="cname" placeholder="City Name"
ng-model="city.cname">
        </label>
```

```
        <label class="item item-input">
          <input type="text" id="address" placeholder="Full
Address" value="{{address}}" disabled="true">
        </label>
        <button class="button button-full button-dark" ng-
click="getLocation(city)">
          Get Geolocation Details
        </button>
      </div>
    </ion-content>
    <ion-footer-bar class="bar-calm">
      <h1 class="title">This is a footer</h1>
    </ion-footer-bar>
  </ion-pane>
</body>

</html>
```

Code Explanation

```
<ion-header-bar class="bar-calm">
      <h1 class="title"><strong>I am an API
Consumer</strong></h1>
    </ion-header-bar>

<ion-footer-bar class="bar-calm">
      <h1 class="title">This is a footer</h1>
    </ion-footer-bar>
```

The **ion-header-bar** and **ion-footer-bar** allow us to define the header and the footer for the page. The contents of the page are defined in the **ion-content** section.

```
<input type="text" id="cname" placeholder="City Name" ng-
model="city.cname">
```

The input field for the city name is bound to the model with the **ng-model** directive.

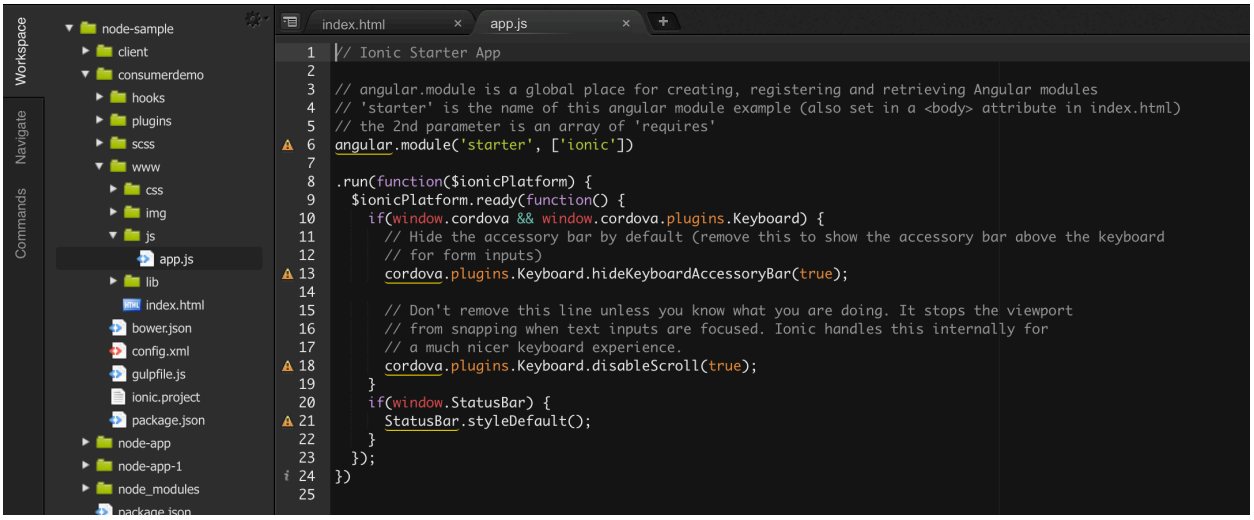
```
<input type="text" id="address" placeholder="Full Address" value="{{address}}" disabled="true">
```

The address field is also bound to the model with the value **{{address}}** being placed into the field through the controller.

```
<button class="button button-full button-dark" ng-click="getLocation(city)">
```

The **ng-click** directive binds the buttons function to the controller which is implement in the applications JavaScript(**app.js**) files.

Next, edit the **app.js** file to implement the functionality for consuming the API and binding the POST response to the display page.



Place the following code, which implements the controller for your web page, in your **/js/app.js** file.

app.js

```
angular.module('starter', ['ionic'])

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {
    if(window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);

      cordova.plugins.Keyboard.disableScroll(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }
  });
})

.controller('BodyCtrl',function($scope,$http){
    $scope.getLocation = function(city){
      $scope.cityname = city.cname;

$http.get("https://maps.googleapis.com/maps/api/geocode/json?add
ress="+$scope.cityname+"").then(function(response){
          $scope.address =
response.data.results[0].formatted_address;
      });
    }
})
```

Code Explanation

Within the **.controller** we can use the **$scope** to define our functions which bind data to/from the model and view.

```
$scope.getLocation = function(city){
      $scope.cityname = city.cname;
```

```
$http.get("https://maps.googleapis.com/maps/api/geocode/json?add
ress="+$scope.cityname+"").then(function(response){
          $scope.address =
response.data.results[0].formatted_address;
      });
    }
```

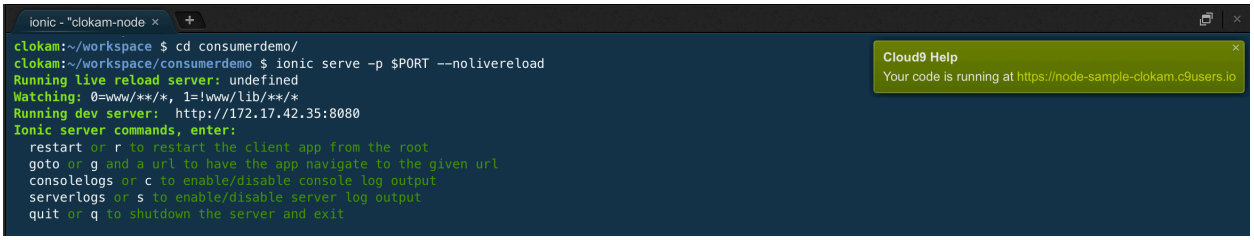
We use the **$http.get** to make the GET call to the REST end-point, and then use the callback function to bind the response(Only the parsed address) to the address filed in the view.

#7 | Use Ionic Serve to Simulate your App

To run the application, use the following command after going into the application directory using the **cd <app-name>** command.

ionic serve –p $PORT --nolivereload

When you run the command it will prompt you for the address that you want to use. Choose option #1 which is the numbered IP Address.



After you run the serve command you will be shown the URL for your app, and on clicking it you will be able to see your app simulated.

[Scroll Below to see final demo screen shots]

Enter the city name of your choice in the form and click on the **"Get Geolocation Details"** button to call the API. You will then see the API being called and the response being placed in the **"Full Address"** text field.