* [Tutorial](https://docs.angularjs.org/tutorial)

PhoneCat Tutorial App

A great way to get introduced to AngularJS is to work through this tutorial, which walks you through the construction of an AngularJS web app. The app you will build is a catalog that displays a list of Android devices, lets you filter the list to see only devices that interest you, and then view details for any device.

demo
application running in the browser

Follow the tutorial to see how Angular makes browsers smarter — without the use of native extensions or plug-ins:

* See examples of how to use client-side data binding to build dynamic views of data that change immediately in response to user actions.
* See how Angular keeps your views in sync with your data without the need for DOM manipulation.
* Learn a better, easier way to test your web apps, with Karma and Protractor.
* Learn how to use dependency injection and services to make common web tasks, such as getting data into your app, easier.

When you finish the tutorial you will be able to:

* Create a dynamic application that works in all modern browsers.
* Use data binding to wire up your data model to your views.
* Create and run unit tests, with Karma.
* Create and run end to end tests, with Protractor.
* Move application logic out of the template and into Controllers.
* Get data from a server using Angular services.
* Apply animations to your application, using ngAnimate.
* Identify resources for learning more about AngularJS.

The tutorial guides you through the entire process of building a simple application, including writing and running unit and end-to-end tests. Experiments at the end of each step provide suggestions for you to learn more about AngularJS and the application you are building.

You can go through the whole tutorial in a couple of hours or you may want to spend a pleasant day really digging into it. If you're looking for a shorter introduction to AngularJS, check out the [Getting Started](https://docs.angularjs.org/misc/started)document.

Get Started

The rest of this page explains how you can set up your local machine for development. If you just want to read the tutorial then you can just go straight to the first step: [Step 0 - Bootstrapping](https://docs.angularjs.org/tutorial/step_00).

Working with the code

You can follow along with this tutorial and hack on the code in the comfort of your own computer. In this way you can get hands-on practice of really writing AngularJS code and also on using the recommended testing tools.

The tutorial relies on the use of the [Git](http://git-scm.com/) versioning system for source code management. You don't need to know anything about Git to follow the tutorial other than how to install and run a few git commands.

Install Git

You can download and install Git from <http://git-scm.com/download>. Once installed you should have access to the git command line tool. The main commands that you will need to use are:

* git clone ... : clone a remote repository onto your local machine
* git checkout ... : check out a particular branch or a tagged version of the code to hack on

Download angular-phonecat

Clone the [angular-phonecat repository](https://github.com/angular/angular-phonecat) located at GitHub by running the following command:

git clone --depth=14 https://github.com/angular/angular-phonecat.git

This command creates the angular-phonecat directory in your current directory.

The --depth=14 option just tells Git to pull down only the last 14 commits. This makes the download much smaller and faster.

Change your current directory to angular-phonecat.

cd angular-phonecat

The tutorial instructions, from now on, assume you are running all commands from the angular-phonecatdirectory.

Install Node.js

If you want to run the preconfigured local web-server and the test tools then you will also need [Node.js v0.10.27+](http://nodejs.org/).

You can download a Node.js installer for your operating system from <http://nodejs.org/download/>.

Check the version of Node.js that you have installed by running the following command:

node --version

In Debian based distributions, there is a name clash with another utility called node. The suggested solution is to also install the nodejs-legacy apt package, which renames node to nodejs.

apt-get install nodejs-legacy npm

nodejs --version

npm --version

If you need to run different versions of node.js in your local environment, consider installing [Node Version Manager (nvm)](https://github.com/creationix/nvm).

Once you have Node.js installed on your machine you can download the tool dependencies by running:

npm install

This command will download the following tools, into the node\_modules directory:

* [Bower](http://bower.io/) - client-side code package manager
* [Http-Server](https://github.com/nodeapps/http-server) - simple local static web server
* [Karma](https://github.com/karma-runner/karma) - unit test runner
* [Protractor](https://github.com/angular/protractor) - end to end (E2E) test runner

Running npm install will also automatically use bower to download the Angular framework into theapp/bower\_components directory.

Note the angular-phonecat project is setup to install and run these utilities via npm scripts. This means that you do not have to have any of these utilities installed globally on your system to follow the tutorial. See **Installing Helper Tools** below for more information.

The project is preconfigured with a number of npm helper scripts to make it easy to run the common tasks that you will need while developing:

* npm start : start a local development web-server
* npm test : start the Karma unit test runner
* npm run protractor : run the Protractor end to end (E2E) tests
* npm run update-webdriver : install the drivers needed by Protractor

Install Helper Tools (optional)

The Bower, Http-Server, Karma and Protractor modules are also executables, which can be installed globally and run directly from a terminal/command prompt. You don't need to do this to follow the tutorial, but if you decide you do want to run them directly, you can install these modules globally using,sudo npm install -g ....

For instance, to install the Bower command line executable you would do:

sudo npm install -g bower

*(Omit the sudo if running on Windows)*

Then you can run the bower tool directly, such as:

bower install

Running Development Web Server

While Angular applications are purely client-side code, and it is possible to open them in a web browser directly from the file system, it is better to serve them from a HTTP web server. In particular, for security reasons, most modern browsers will not allow JavaScript to make server requests if the page is loaded directly from the file system.

The angular-phonecat project is configured with a simple static web server for hosting the application during development. Start the web server by running:

npm start

This will create a local webserver that is listening to port 8000 on your local machine. You can now browse to the application at:

http://localhost:8000/app/index.html

To serve the web app on a different ip address or port, edit the "start" script within package.json. You can -a to set the address and -p to set the port.

Running Unit Tests

We use unit tests to ensure that the JavaScript code in our application is operating correctly. Unit tests focus on testing small isolated parts of the application. The unit tests are kept in the test/unit directory.

The angular-phonecat project is configured to use [Karma](https://github.com/karma-runner/karma) to run the unit tests for the application. Start Karma by running:

npm test

This will start the Karma unit test runner. Karma will read the configuration file at test/karma.conf.js. This configuration file tells Karma to:

* open up a Chrome browser and connect it to Karma
* execute all the unit tests in this browser
* report the results of these tests in the terminal/command line window
* watch all the project's JavaScript files and re-run the tests whenever any of these change

It is good to leave this running all the time, in the background, as it will give you immediate feedback about whether your changes pass the unit tests while you are working on the code.

Running End to End Tests

We use End to End tests to ensure that the application as a whole operates as expected. End to End tests are designed to test the whole client side application, in particular that the views are displaying and behaving correctly. It does this by simulating real user interaction with the real application running in the browser.

The End to End tests are kept in the test/e2e directory.

The angular-phonecat project is configured to use [Protractor](https://github.com/angular/protractor) to run the End to End tests for the application. Protractor relies upon a set of drivers to allow it to interact with the browser. You can install these drivers by running:

npm run update-webdriver

*(You should only need to do this once.)*

Since Protractor works by interacting with a running application, we need to start our web server:

npm start

Then in a separate terminal/command line window, we can run the Protractor test scripts against the application by running:

npm run protractor

Protractor will read the configuration file at test/protractor-conf.js. This configuration tells Protractor to:

* open up a Chrome browser and connect it to the application
* execute all the End to End tests in this browser
* report the results of these tests in the terminal/command line window
* close down the browser and exit

It is good to run the end to end tests whenever you make changes to the HTML views or want to check that the application as a whole is executing correctly. It is very common to run End to End tests before pushing a new commit of changes to a remote repository.