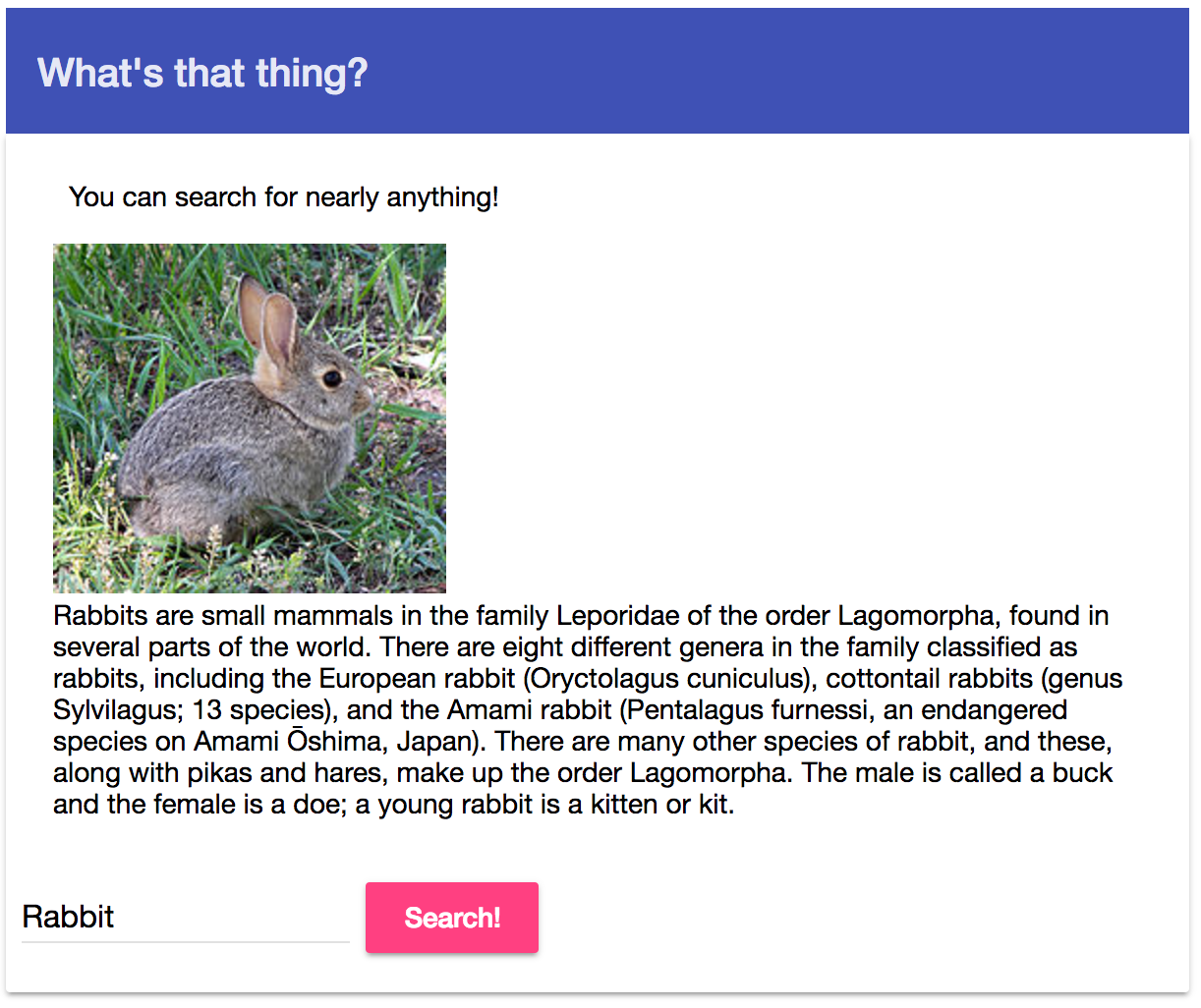
**What’s that thing?**

Workshop Angular

# Introduction

In this workshop, you will learn to create a simple Angular application, called “*What’s that thing?*”. The app allows a user to search for a thing, after which it will display a short description and picture about that thing. The result will look something like this:



In this workshop, you will familiarize yourself with core Angular concepts: modules, components, services, dependency injection and working with the Angular CLI.

# Prerequisites

For you to work on the application, as with many front-end applications, certain tooling is required to be installed on your laptop. Let’s get to that right away.

## NodeJS

Install *NodeJS* by downloading it from [www.nodejs.org](http://www.nodejs.org). NodeJS is a JavaScript run-time, enabling you to run JavaScript programs natively on your laptop. This works much like the Java JRE for running Java Applications, or the .NET CLR for running Windows-based .NET applications. The Angular CLI is such a NodeJS application.

## Visual Studio Code

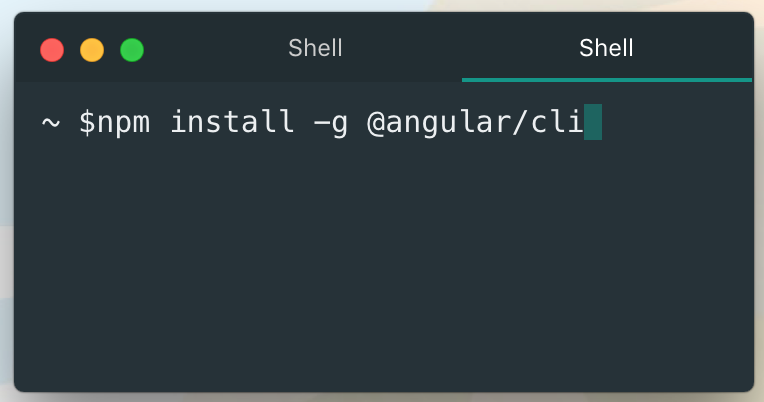
Next, you need a nice editor for working on your code. You can bring any editor you like. It is recommended to use one with TypeScript support and an integrated terminal (more on this later). A very good and free IDE is Visual Studio Code. It is available for all major platforms and can be downloaded from <https://code.visualstudio.com/>.

## Angular CLI

Finally, we need to install the Angular CLI. As mentioned, this is a NodeJS application. You install these by running a terminal command (*Powershell* on Windows and *Terminal* on a Mac):

npm install –g @angular/cli

This command tells the *Node Package Manager* to look for a package called *@angular/cli*, which is the official package containing the Angular CLI. It will be installed globally, meaning we can directly access it from the command line.



# Getting ready!

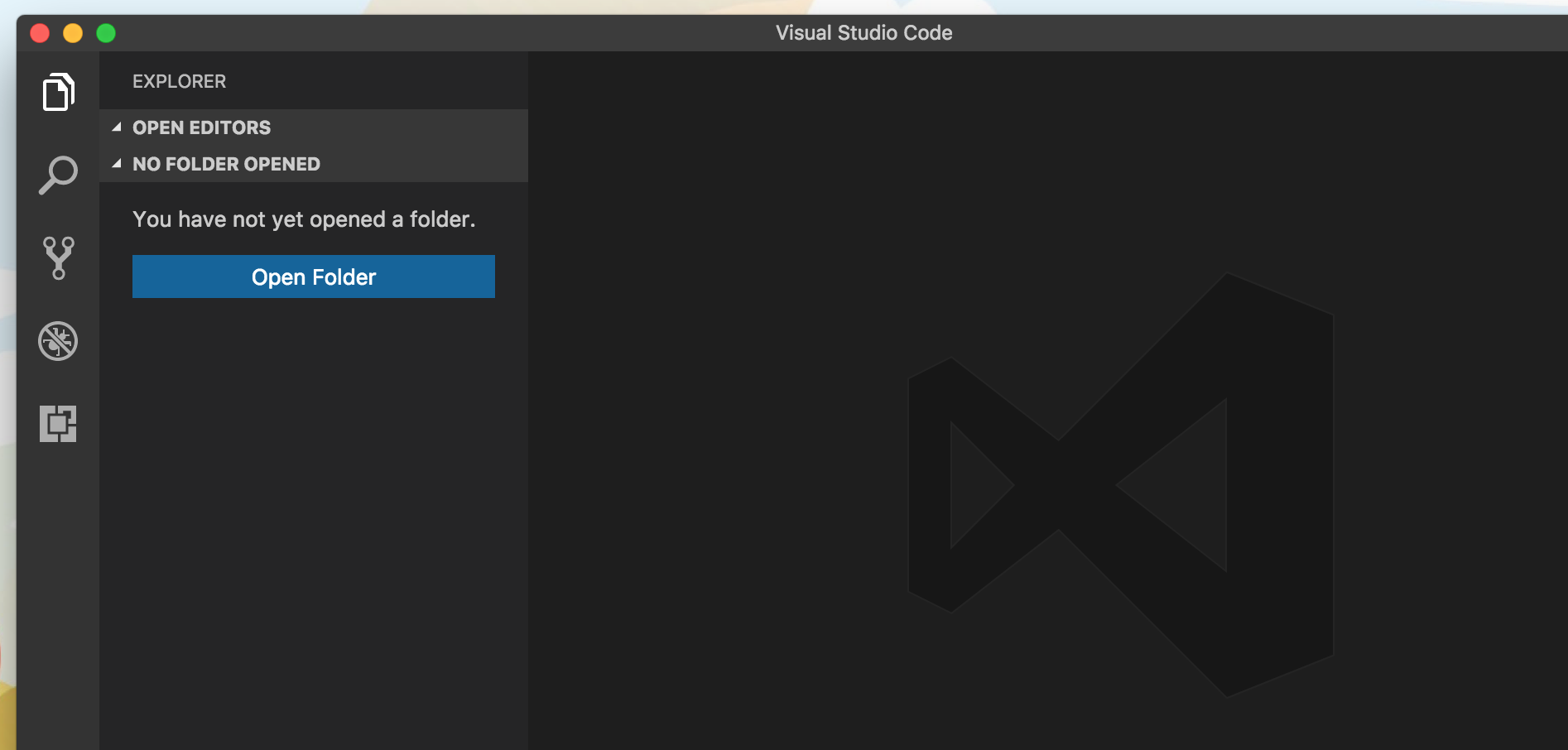
Without further ado, let’s initialize our app. Go back to your terminal, navigate to a suitable location where you’d like to place the source code files and run the following command:

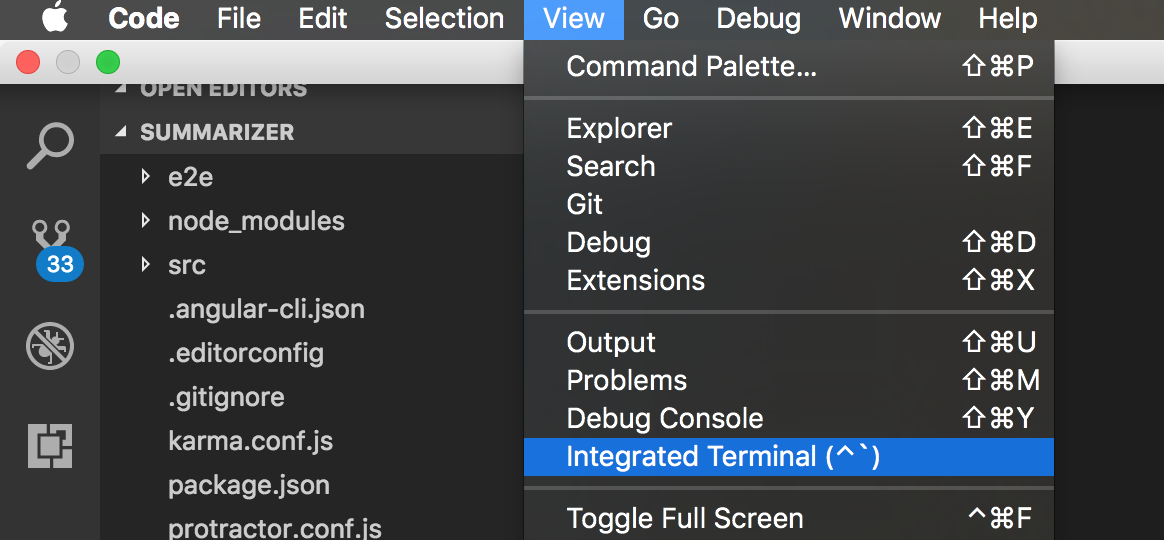
ng new whats-that-thing

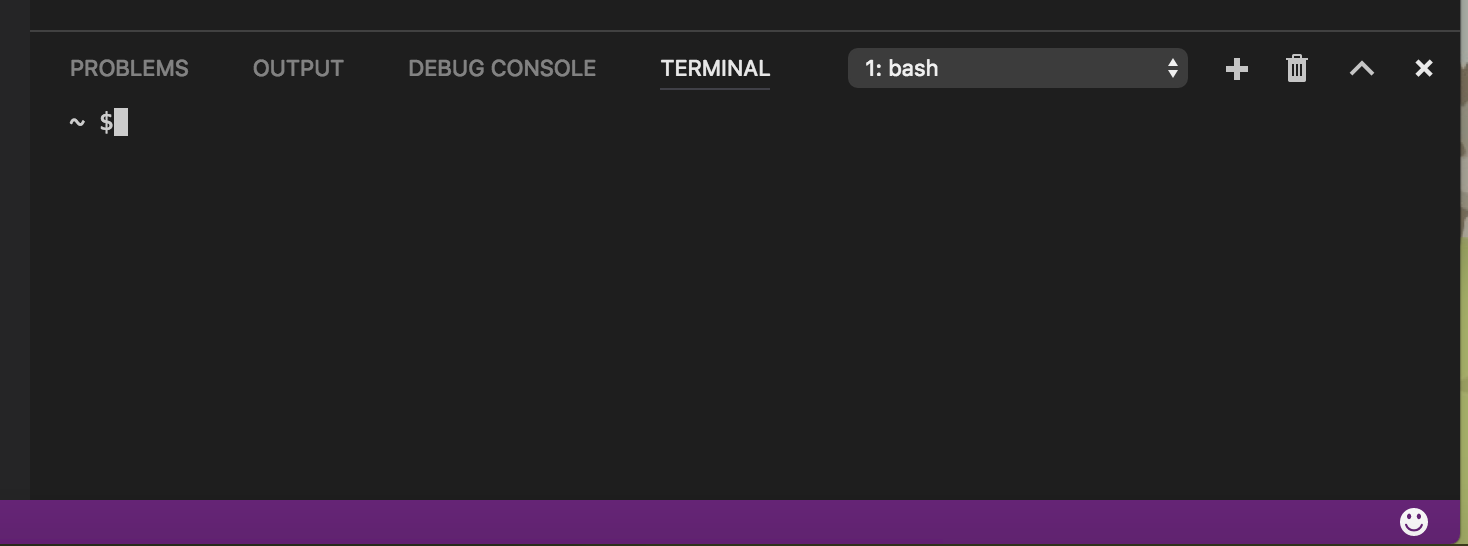
This command tells the Angular CLI to initialize a new project, called *whats-that-thing*. It will generate several configuration files and start downloading the dependencies, such as the Angular framework itself. Give it a moment to complete.

After it completed, open Visual Studio Code and open the newly created ‘whats-that-thing’ folder, in which you initialized the app, by clicking the *open folder* button.

Also open the integrated terminal (*View 🡪 Integrated terminal*). We will use it for running further commands in the project.







# Material Design

Material Design is a design specification by Google for building beautiful modern user interfaces. It comes with a set of high quality prebuilt Angular components which you will use to build the user interface. You can check them out at <https://material.angular.io/>. During this workshop, you will use the *Button*, *Card* and *Input* components.

These components are grouped together in a set of modules and bundled in their own separate node package. In the integrated terminal, run the following command:

npm install --save @angular/material @angular/animations

This command installs the *material* and *animations* package files into the project.

Next, you need to declare that your own Angular app uses these modules. Find the file *src/app/app.module.ts*. This file describes the main Angular module: the entry point of your application. Take a few seconds to familiarize yourself with this file.

At the head of every TypeScript file, *imports* may be defined. An import is enabling you to use code defined in another file or package. They are used to cleanly separate functionality and enable reuse. Extend it by importing the required Material modules and the Animations package as follows:

import { MdButtonModule, MdCardModule, MdInputModule } from '@angular/material';

import { BrowserAnimationsModule} from '@angular/platform-browser/animations';

You can now use these imported definitions in your own module definition. Extend the *imports* array of the Angular Module as follows:

imports: [

BrowserModule,

FormsModule,

HttpModule,

**BrowserAnimationsModule,**

**MdButtonModule,**

**MdCardModule,**

**MdInputModule**

],

It is now possible to use the contents of these modules throughout the application, which you will do shortly.

The final step is to load a preconfigured color set, to make it all look nice. Find the *src/app/index.html* file and add the following to the *<head>* section:

<link href="../node\_modules/@angular/material/prebuilt-themes/indigo-pink.css" rel="stylesheet">

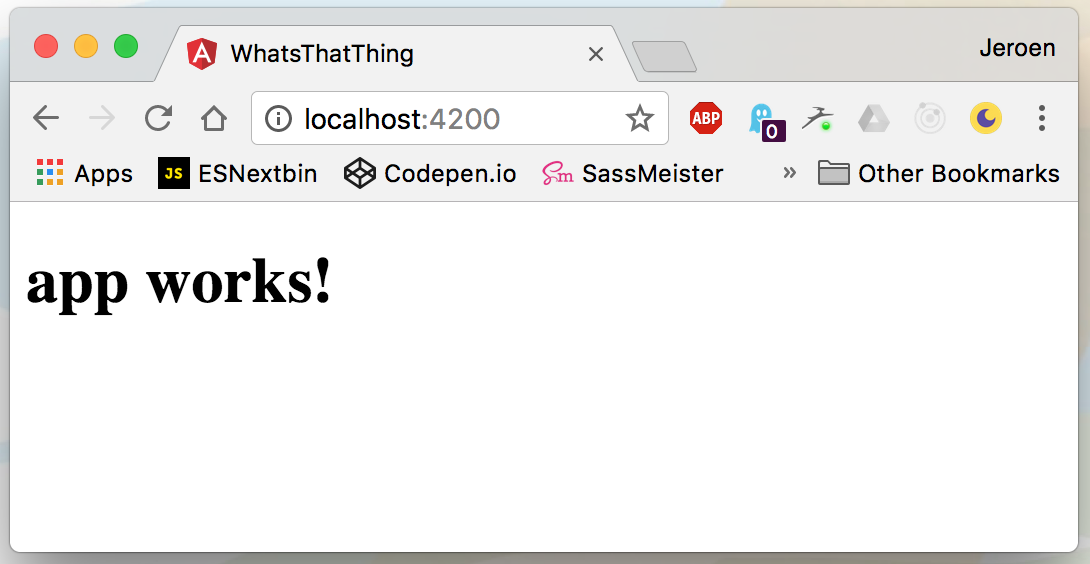
# Let’s run it!

That’s enough initial setup without any gratifying results for now. Let’s try and run the application. In the integrated terminal, run the following command:

ng serve

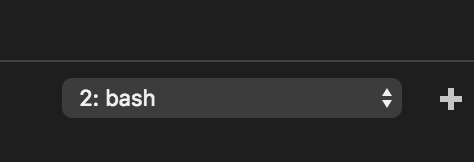
This command will compile and optimize all our source files and set up a local development server hosting the app. The first time round, it may take a while to complete. When it is done, open a web browser and navigate to <http://localhost:4200/>. If all went right, you should see your Angular application, stating that the app works. Congratulations!

The development server is configured in such a way that when you make a change to your code, it automatically recompiles and reloads the web page. This makes development super-fast. Try it out by finding the file *src/app/app.component.ts* and updating the title to “What’s that thing?”. Save the file and notice that the browser nearly immediately reloads, reflecting the changes you just made.



# Create your first component

Back in Visual Studio Code, open a second terminal window, so that you can leave the development server running. You can do this by clicking the **+** button in the Terminal tab.



Run the following command to create a new component called *summary*:

ng generate component summary

This will create the source files that encompass a component, as well as update our Module definition, so that we can use it straight away. Have a look at the generated files in the *src/app/components/summary* folder.

The *summary.component.ts* file holds the component class and metadata. It tells you that the *selector* for this component is *app-summary*. Let’s try and add the summary component to our app component by adding it to its HTML template. Find the file *src/app/app.component.html* and update it as follows:

<h1>

{{title}}

</h1>

**<app-summary></app-summary>**

Go back to your web browser and check that the new component is loaded.

# Style the summary component with Material Design

## The user interface

Earlier in this workshop, you have added Material Design modules to your app. It is time to put those to good use. You will use the Card, Input and Button component to build the user interface for the summary component. Have you checked out their documentation?

Update the *src/app/components/summary.component.html* file as follows:

<md-card>

<md-card-header>

<md-card-title>Find a summary</md-card-title>

</md-card-header>

<md-card-content>

<p>

{{ summary }}

</p>

</md-card-content>

<md-card-actions>

<md-input-container>

<input mdInput #searchbox>

</md-input-container>

<button md-raised-button color='accent'>Search!</button>

</md-card-actions>

</md-card>

## Interpolation bindings

Notice the *{{ summary }}* interpolation binding? This means that Angular will look for a property called *summary* on the component class and populate the view with its value. If the value changes, the view is automatically synchronized by Angular.

For now, let’s simply initialize it with a static value. Update the *src/app/components/summary.component.ts* file as follows.

export class SummaryComponent implements OnInit {

**private summary = 'Search for a summary';**

constructor() { }

ngOnInit() {

}

}

# Create the data service

## Generating the service

Whilst starting to look pretty good, your app doesn’t actually do very much. Let’s change that now. The goal of the app is to provide the user with descriptive information about a thing they search for. A service is a good spot for putting such reusable application logic. A service is a TypeScript class which can be *injected* into other services or components. Let’s create one using the Angular CLI, by typing the following command in the integrated terminal:

ng generate service summary

To be able to use the service throughout your application, update the *src/app/app.module.ts*, this time extending the *providers* array (which contains the services this module provides internally):

import { SummaryService } from './summary.service';

// ...

providers: [

SummaryService

],

## Injecting the Angular HTTP service

You are going to create a function called *getSummary* whose job it is to return a summary text based on a search term. One approach would be to grab a dictionary and start typing. Another is to use the Wikipedia API to save yourself quite the effort.

To use the Wikipedia API, you need to be able to do HTTP requests. Luckily, the Angular framework provides a service that does just that, aptly named *Http*. To use the *Http* service, you need to *inject* it in your *Summary* service. Services are injected by adding them as arguments to the *constructor* method. This instructs Angular to create an instance of the *Http* service and provide it to the *Summary* service upon creation.

Open the file *src/app/summary.service.ts* and update it as follows:

import { Injectable } from '@angular/core';

**import { Http } from '@angular/http';**

@Injectable()

export class SummaryService {

constructor(**private http: Http**) { }

}

## Creating the *getSummary* method

The Wikipedia API can be queried for data at the following URL:

[https://en.wikipedia.org/w/api.php?origin=\*&format=json&action=query](https://en.wikipedia.org/w/api.php?origin=*&format=json&action=query)

You will append further arguments to that URL, instructing it to return a page extract (the introduction section) and to return plain text instead of HTML. A complete example URL is:

[https://en.wikipedia.org/w/api.php?origin=\*&format=json&action=query&prop=extracts&exintro=&explaintext=&titles=Rabbit](https://en.wikipedia.org/w/api.php?format=json&action=query&prop=extracts&exintro=&explaintext=&titles=Rabbit)

You can open this URL in a browser. The Wikipedia API should return a JSON object (which is a JavaScript-friendly format for structuring data), containing a short description about rabbits.

The *getSummary* method should do three things:

1. It should use the *Http* service to call the above URL, substituting the word “Rabbit” for a term entered by the user (hence, a *function argument)*;
2. It should parse the response to a JavaScript object using the *.json()* method, to enable programming the remaining application flow;
3. Because the API returns all (or no) pages that might relate to the searched term, it should extract the first page to simplify things a bit.

A complete implementation is given below. Add it to the *summary.service.ts* file.

private url = 'https://en.wikipedia.org/w/api.php?origin=\*&format=json&action=query';

getSummary(term) {

return this.http.get(`${this.url}&exintro=&explaintext=&prop=extracts&titles=${term}`)

.map(result => result.json())

.map(this.getFirst);

}

// private helper function to extract the first Wikipedia page

private getFirst(response: any) {

const pageId = Object.keys(response.query.pages)[0];

return response.query.pages[pageId];

}

# Putting it all together

To use the *summary* service in the *summary* component, inject it by updating the constructor method in the component class.

Update *src/app/components/summary.component.ts* as follows:

import { Component, OnInit } from '@angular/core';

**import { SummaryService } from '../summary.service';**

@Component({

selector: 'app-summary',

templateUrl: './summary.component.html',

styleUrls: ['./summary.component.css']

})

export class SummaryComponent implements OnInit {

private summary = 'Search for a summary';

constructor(**private summaryService: SummaryService**) { }

ngOnInit() {

}

}

We can now create a component method which updates the *summary* property using the *summary* service. Add the following code to the component class.

getSummary(searchText) {

this.summaryService.getSummary(searchText).subscribe(result => {

this.summary = result.extract;

});

}

The final piece of the puzzle is calling the method with the entered search text when the user clicks the Search button. Open the file *src/app/components/summary.component.html* and update the button as follows:

<button md-raised-button color='accent'

**(click)="getSummary(searchbox.value)">**Search!</button>

This tells the view to listen for *click* event and call the *getSummary* method with the value of the *searchbox* (the **referenced** input field) when that happens.

Return to your web browser and verify that everything works. Have fun searching for marvelous things!

# Additional exercises

If you are excited and feel like expanding the application further, here are some suggestions.

## Use a good-looking header

The application header could do with a bit of styling. Go to <https://material.angular.io> and search for the toolbar component. Try incorporating it in the application instead of the current header.

## Add an image (advanced)

The Wikipedia API can also be used to retrieve an image for a search term. The URL looks a lot like the one used for retrieving the summary. Here is an example URL:

[https://en.wikipedia.org/w/api.php?origin=\*&format=json&action=query&titles=Rabbit**&prop=pageimages&pithumbsize=200**](https://en.wikipedia.org/w/api.php?origin=*&format=json&action=query&titles=Rabbit&prop=pageimages&pithumbsize=200)

Open the above URL in your web browser and try to analyze the Wikipedia API response. Compare it to the API response for retrieving a text extract used earlier. You will need to pull out the image URL instead of the extract text this time.

Extend the *summary* service with a method to get an image for a search term and use it in the *summary* component. To get started, update the component class with the following default property:

private imgUrl = '';

Update the HTML template by adding the following:

<img [src]="imgUrl" />

# Thank you

I hope you have enjoyed taking your first steps with Angular. If you have any feedback or suggestions for me, please contact me at [jeroen@frontmen.nl](mailto:jeroen@frontmen.nl). I’d love to hear from you!