



Rapid API

## | Getting started

**Time:** 15-20 mins

**Prior knowledge:** none

**Author:** Iddo Gino, CEO & Founder



## 1. Intro

In this tutorial, we you will go through the basics of creating web services, adding endpoints and pouring functionality into them with the RapidAPI system. If you are already familiar with the concept of Web Services, Endpoints and REST, you may head over the chapter 3. Otherwise, chapter 2 has a quick & dirty intro on the matter.

## 2. Web services

The RapidAPI system allows you to quickly & easily create an online web-service. A web service is basically a collection of functions, run on a cloud server which you can access from your mobile app / website. Each function is called an endpoint and has an address. For example, if you are implementing a basic user management system, you would have a sign-up end-point, to which you will send all the new user's data. The endpoint will perform some basic checks (for example - making sure that the email address is valid), and then it'll save the new user to the database.

### < POST >

used when sending new information to the server. (e.g. create a new user / post a new picture).

### < GET >

used for receiving information from the server. (e.g. get a user's birthday / get recent posts by friends).

### < PUT >

used for updating and modifying information on the server. (e.g. update a users phone number / update events invitees list).

### < DELETE >

used for when deleting information from the server. (e.g. remove a picture / remove a user).

It's up to the developer to decide the type of each endpoint, but its best practice to try and have the type related to the functionality.

To identify endpoints and access them, each endpoint has an address. All addresses on the system have this basic format:

<http://projectName.imrapid.com/path>

Notice that projectName is the name you have chosen for your project where all your endpoints are contained, and path is the path to your endpoint. Each endpoint has a path that starts with a / and can contain words, numbers and more



slashes. For example, the path to a signup endpoint may look like:  
<http://myAwesomeProject.imrapid.com/signup>.

Do notice that several endpoints can have the same path, as long as they are of different types (POST/GET/DELETE/PUT).

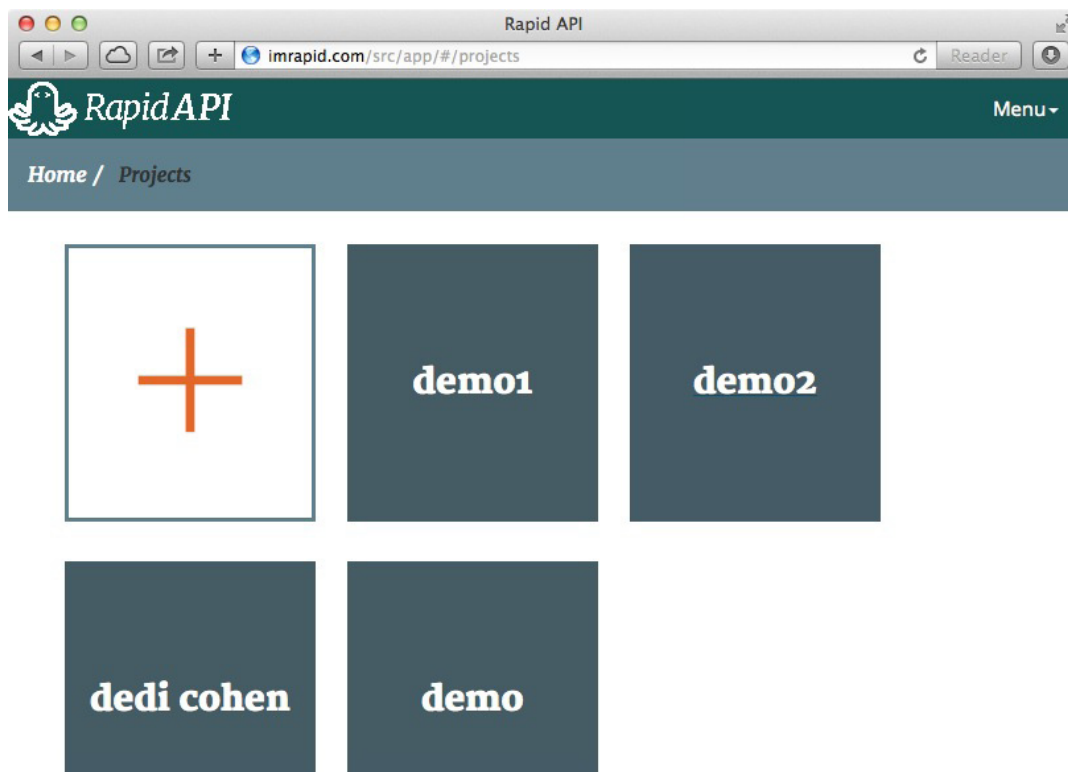
As a matter of fact, when you write <http://google.com> in your browser, you make a GET request to that address.

From your mobile app / website, you will access these endpoints (with HTTP requests) and get messages back. Also, endpoints may do things like send text messages, push notifications, emails and more. Throughout this tutorial, we will see how to create a project, create endpoints in it, and add functionality into them.

### 3. Creating a new project

First things first, you need to create an account on RapidAPI. Creating an account on RapidAPI is free, quick and painless. Just head over to <http://www.imrapid.com/src/app/>, fill in your details and that's it. You can now login from the same address and start using RapidAPI.

Once you are logged in, you will see the projects page (pictured below).



In this page, you can see all your projects, and create new ones. You can go ahead and create a new project by clicking the ‘+’ button and giving it a name. Notice names must be unique and can only contain characters, the ‘-’ sign and numbers.

Now that you have your project ready and set up, you can go ahead and add some endpoints into it.

## 4. Creating your first block based endpoint

To create your first block based endpoint, click on the project in which you want to create it, enter the ‘endpoints’ tab, and click on the button saying ‘+Click to add a new endpoint’. You’ll now see a form with a dropdown for choosing the endpoint type (for now leave it as ‘GET’), the path (must start with a ‘/’), and the way you want to create your endpoint (for now choose blocks, we’ll go over the other types later).

Rapid API

Menu

Home / Projects / demo / Endpoints

Overview

Endpoints

Resources

Databases

Give some details about this endpoint

GET

Enter URL

blocks

Cancel

Save

URL

GET

/foo

js

Updated:

URL

GET

/page

static

Updated:

URL

GET

/send

blocks

Updated:

/mine?

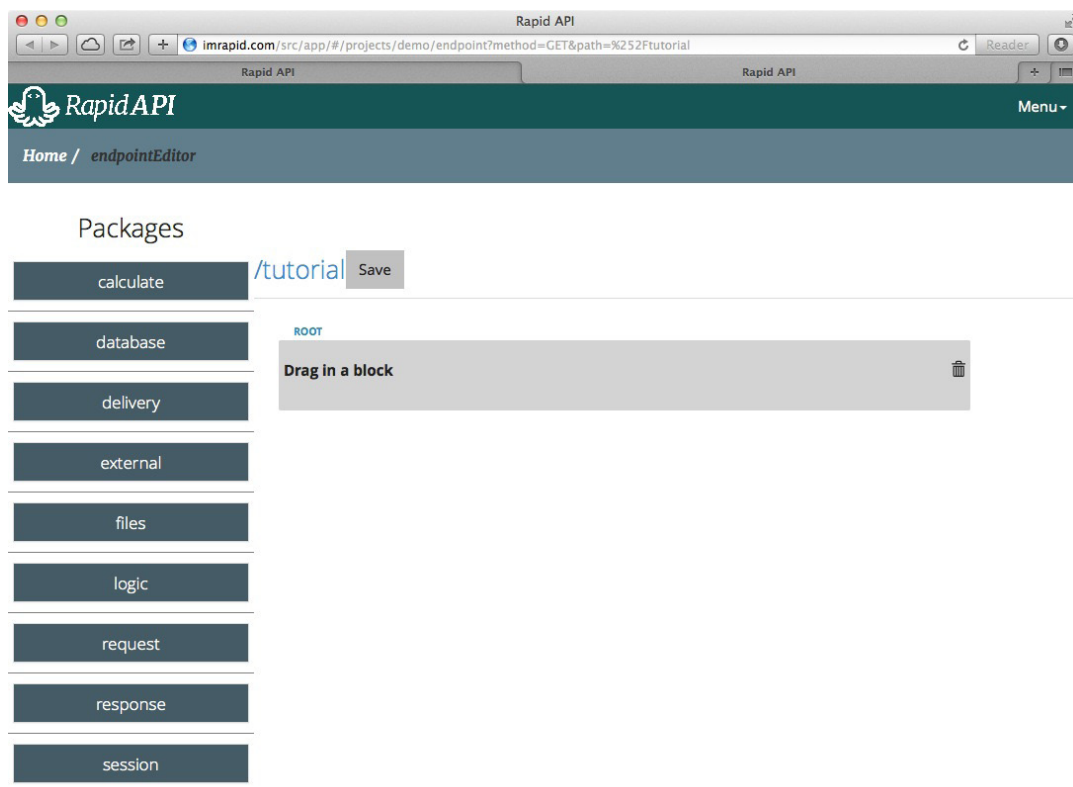
blocks

Updated:

Once you have created your endpoint and clicked ‘save’, you should see the new endpoint in the list (if you don’t, refresh the page). Click on it to edit the endpoint – you’ll now enter the endpoint editor.

### < Blocks >

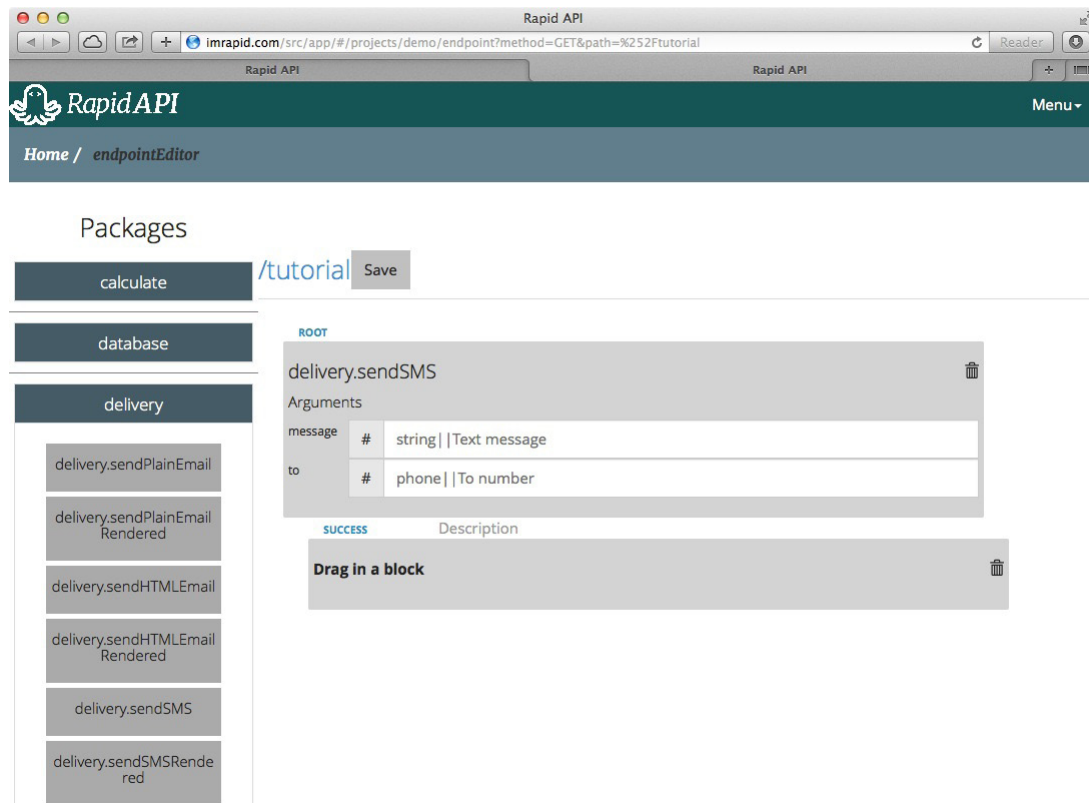
Block based endpoints are made out of a sequence of actions. Actions can be things like ‘query database’, ‘send email’, ‘perform multiplication’ or ‘send message back to user’. Each action is represented by a block. On the right hand side of the editor, you will see the package menu.



Each **package** contains several blocks that have similar functionality. For example, the delivery package has blocks for doing things like sending emails, text messages and push notifications.

You can go ahead and drag a block in. For example, open the delivery package and drag in the ‘delivery.sendSMS’ block.



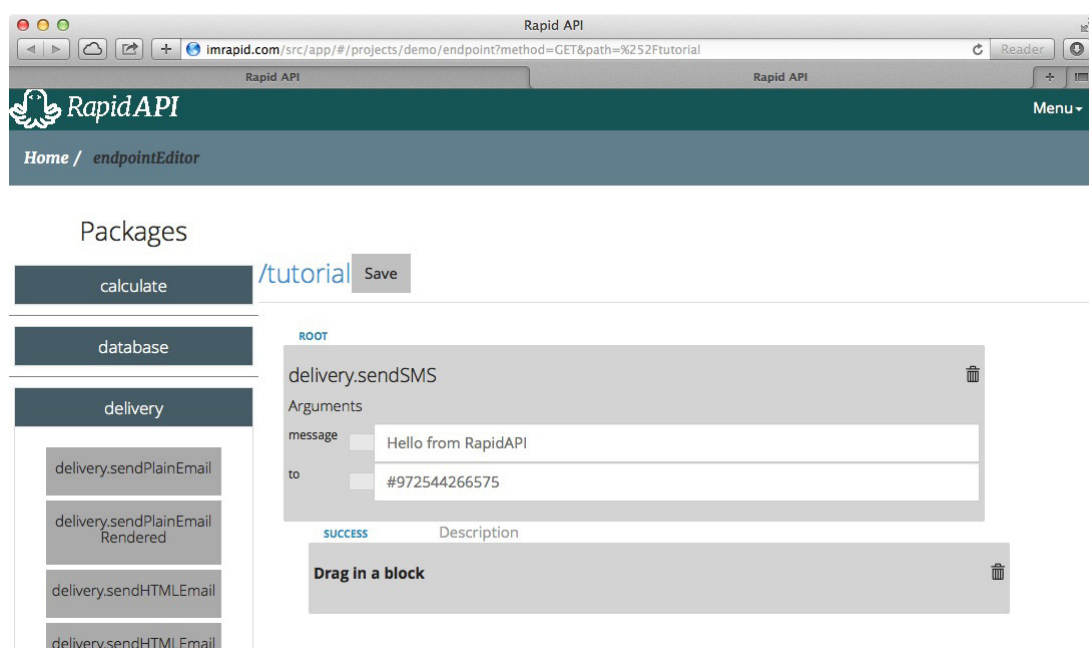


The screenshot shows the Rapid API interface. On the left, under the 'Packages' section, the 'delivery' package is selected, showing a list of endpoints including 'delivery.sendSMS'. The main area displays the 'delivery.sendSMS' block with two arguments: 'message' (string | Text message) and 'to' (phone | To number). A 'SUCCESS' description block is also visible below the arguments.

This block sends a text message to a given phone number. You can see that the block receives arguments. Arguments are pieces of data you pass to the block so it can function as you desire. In this example, the block receives 2 arguments:

- message – the text message to be sent to the number
- to – the phone number to which you wish to send the message (in the format of ‘+[country code][phone number]’).

You can give it any desired arguments, for example:



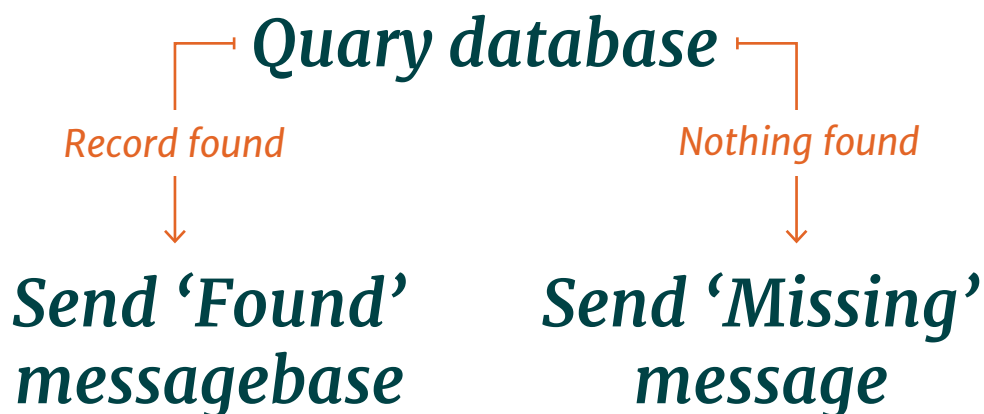
This screenshot shows the same Rapid API interface as the previous one, but with example values entered into the arguments. The 'message' argument is filled with 'Hello from RapidAPI' and the 'to' argument is filled with '#972544266575'. The 'SUCCESS' description block remains below.



Now, when a request is made to this endpoint, it'll send the message to the phone number. That's it. If you have made it a GET endpoint, you can put your phone number in, save it, and then put the end-point's URL in your browser. If all goes as planned, you should receive the text message.

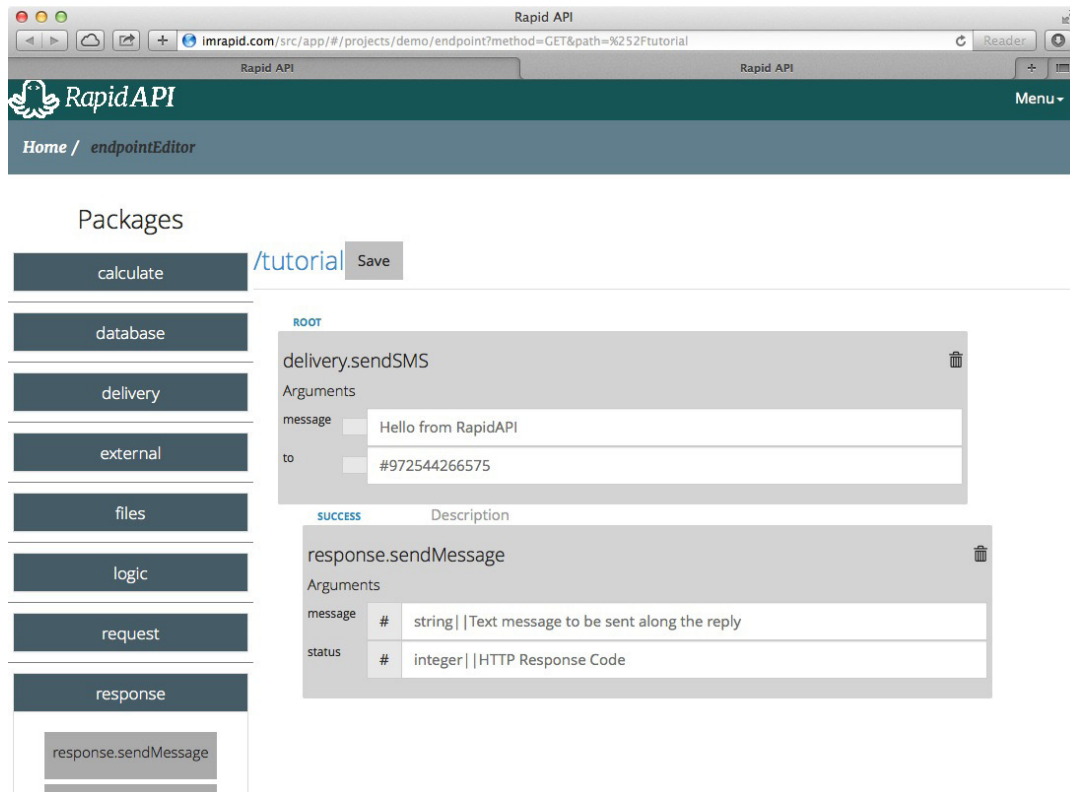
So that's pretty exciting, but still not enough to create your \$1,000,000,000 startup. Most real endpoints will have more than one blocks, and will proceed based on the outcomes of blocks - that's how a sequence is created. In our example, you may want to do something after the SMS is sent (maybe log it in a database), or if you query a database, you may want to do something if a record is found, and something else if nothing is found. To do just that, we use outcomes. Each block has certain outcomes, where you can place other blocks.

For example, imagine you want to query a database, if a record is found, send a "Found" message back to the user, and if nothing is found, send a "Missing" message. This will function similarly to the following diagram:



To accomplish that, you will see that under every block you have, there are little titles with the possible outcomes for this block. The sendSMS block for example has only one outcome - 'SUCCESS'. Under that outcome, you can drag in another block, which will be executed after the text message is sent. If you want to send a message back to the client (so it'll appear in the browser window after the request is finished), drag a 'response.sendMessage' from the 'response' package under the 'sendSMS' block. It should look like that:





angular) but are not so good for other front-end technologies.

In the response message, you can enter some arguments as well:

**1. status:** the HTTP request status code. This is an integer representing the state of the request (a convention used by developer). Common codes are:

- 200 – All good, request performed
- 400 – Error in request
- 401 – Authentication error
- 500 – Server error

**2. message:** the message to be sent back (something like “Success” or “User created”).

### < The context >

Some blocks may also produce data, that may be used later down the sequence by other blocks (for example, the calculate.add block produces the sum, and we may want to use it later down the execution sequence). To do so, blocks save those pieces of data in the **context**. The context is a basic key-value store, that lives for the execution of the endpoint, and to which you can store data and use it later.

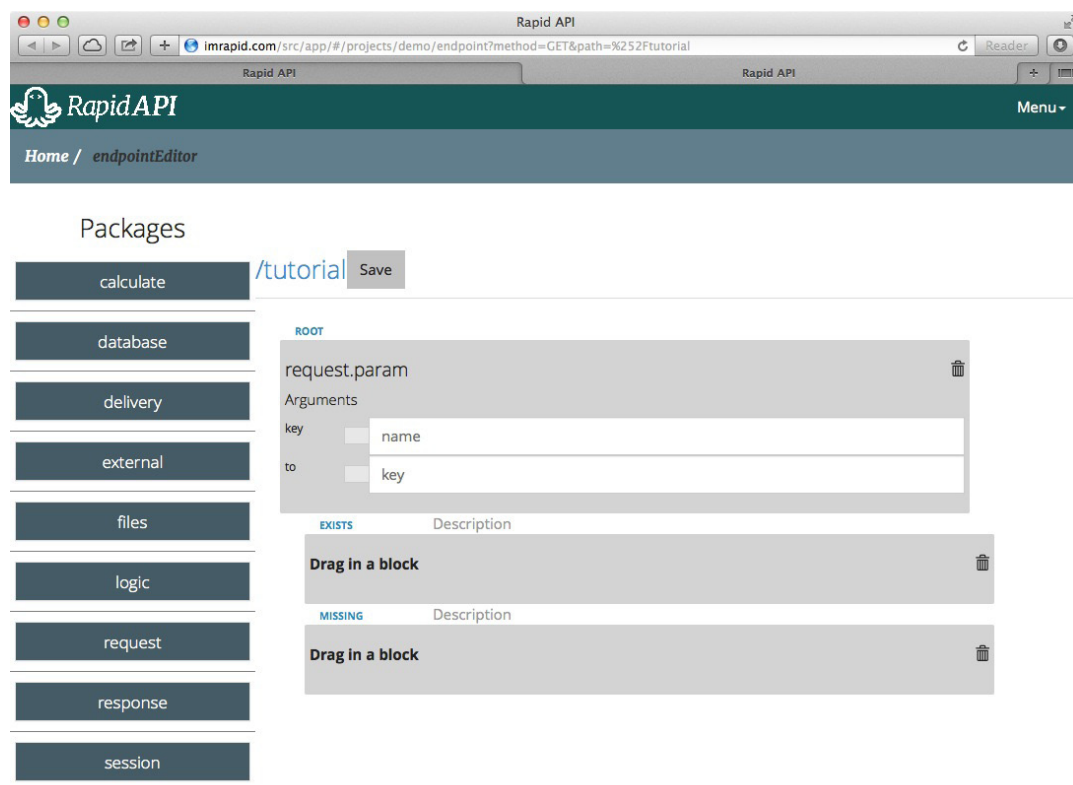




For example, if you look at the request.param block, which fetches a parameter passed along in the request (in the URL or the body), it receives 2 arguments: key – the key of the parameter you want to get (if it is a url, so the key will be http://...url...?key=value).

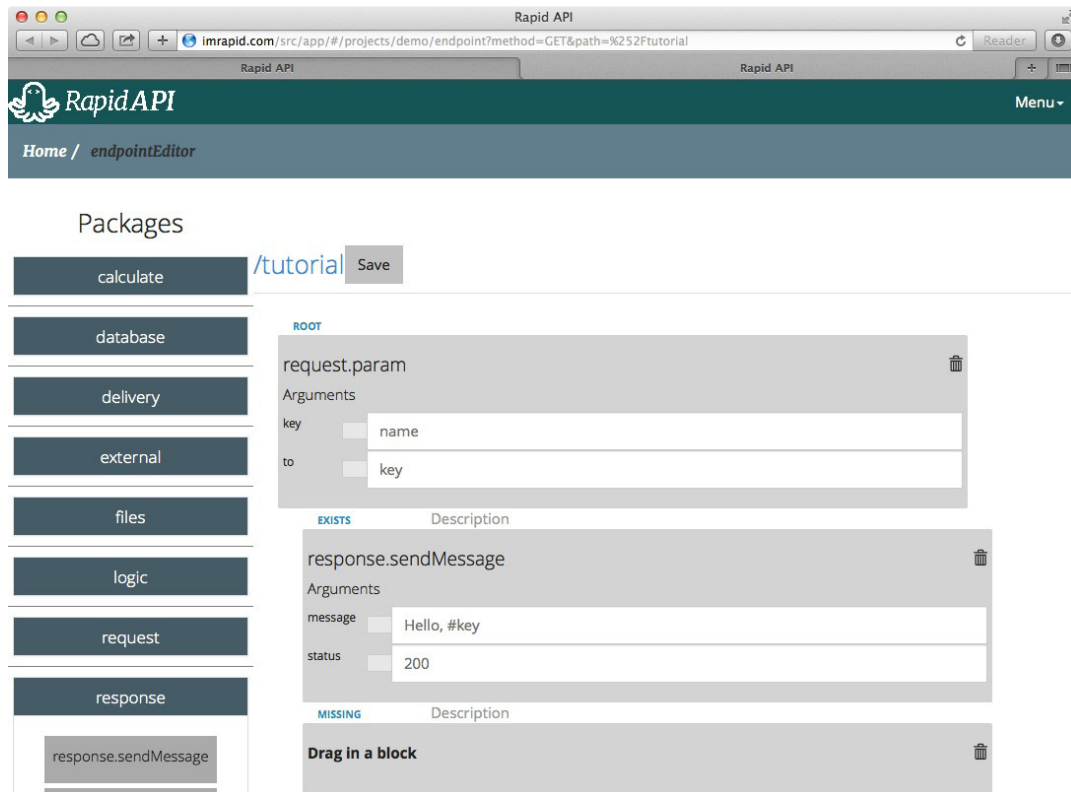
to – the name of that data in the context – any name that doesn't contain white spaces

Let's say you are trying to get a name passed to the request, and store it in the context under "name". The request block for that would look like that:



Now, let's say you want to respond to the request with a message with "Hello, " and the name that was passed (e.g. if you pass "Cole", you will get back "Hello, Cole". To do that, you must drag in a response.sendMessage block in the success outcome of the req.param block, and in it refer to the value in the context. You refer to keys in context with "#key" – where key is the key you assigned to the value. In our example, we will get:





In the context, there are many pre-installed constants you can use anytime. Here are some of them:

- #date** - the current date in DD/MM/YYYY format
- #time** - the time in HH:MM format
- #dayInWeek** - the current day in week (e.g. Wednesday)
- #month** - the month (e.g. December)
- #PI** - pi (3.14...)
- #e** - euler's constant (2.7...)

Moreover, you can access many of the basic request properties and parameters (such as information passed in the URL or body of the request) thru built in variables in the context:

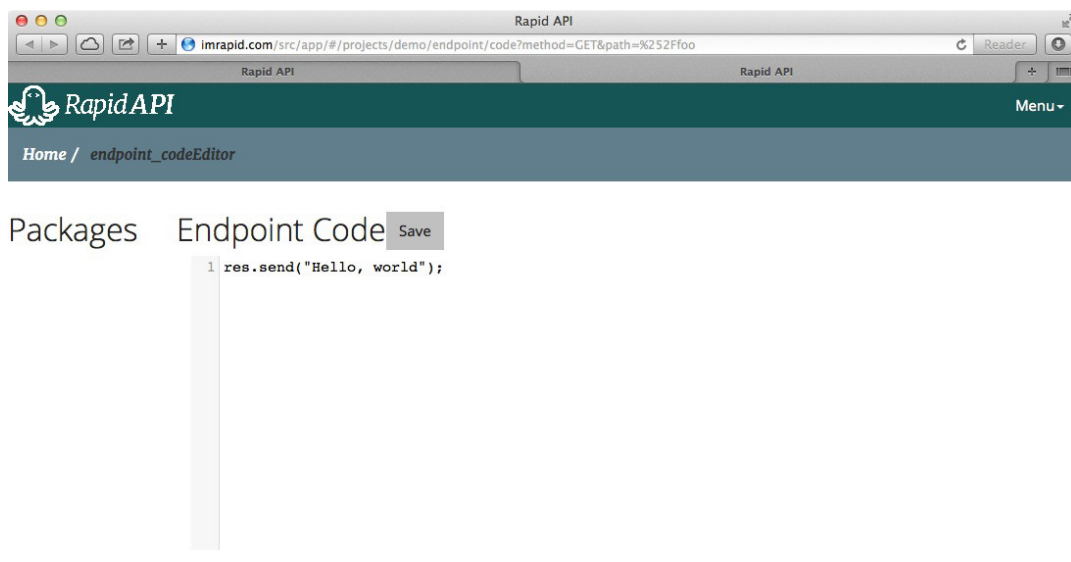
- #query** - used to access parameters passed in the request URL (after the '?' sign). For example, if the request was made with the following parameter in the URL: <http://project.imrapid.com/path?num=100>, you can use `#query.num` to fetch it.
- #body** - used to access parameters passed on the request body.
- #cookies** - used to access the cookies sent with the request.
- #session** - used to access the session parameter associated with the request.



## 5. Code endpoints

This section will discuss the ability to create endpoints by programming them. If you have no programming experience, you may move on.

Although block endpoints allow you to do almost anything imaginable, you may encounter situations where you have a really really complex endpoint and would rather code it yourself. To do so, you can create a new endpoint, and choose 'code' instead of 'blocks' as the endpoint type. You can now click on the endpoint you have created to open up the code endpoint editor, which will allow you to edit the code.



You can use any valid JavaScript, alongside with any of the methods and data objects found in the left menu, that let you perform tasks like sending emails, querying databases and responding to the client.

Note that every endpoint is being run on a virtual machine with a limited time of 10 seconds. If your endpoint exceeds this time limit, it'll be stopped and an error message will be sent back to the client.

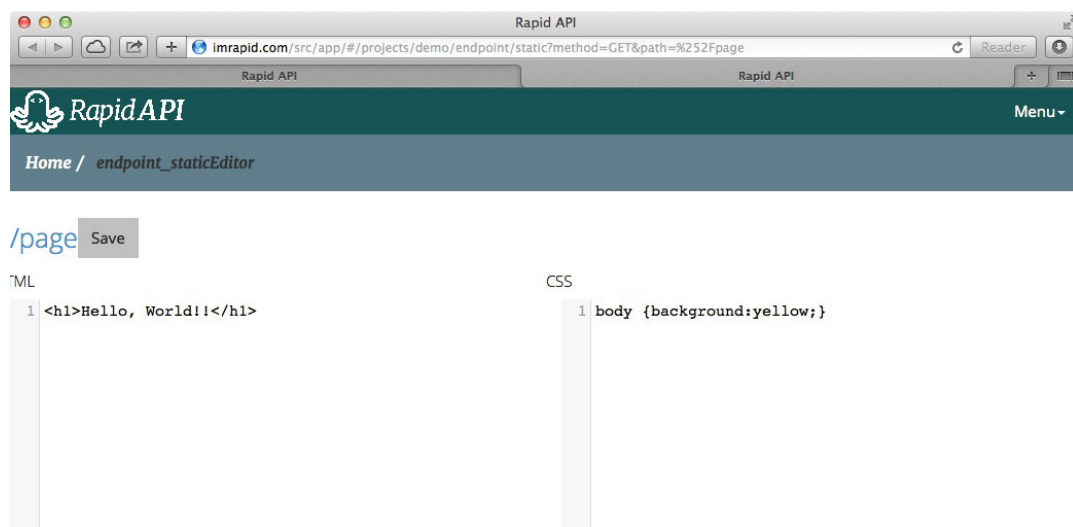


## 6. Static endpoints

Another type of endpoints you can create are static endpoints. The idea behind static endpoints is to serve static web pages on certain paths. As a reminder, when you put `www.google.com` in your browser, you actually make a GET request to that address, and receive a static page in return. You can do the same thing by creating an endpoint of type 'static'.

Once you click on the static endpoint, you will see 3 text editors: HTML (for the page structure), CSS (for styling) and JavaScript (for function). You can just write code into those, and once you'll access the endpoint, you will receive this code as a static page. You can see an example below:

We put this code into an endpoint:



And when we access it from a browser, we will see this:



Beautiful, right?





## 7. Connecting to a database

One of the main jobs of a web service is to keep and manage data. This is usually done in a database. RapidAPI currently supports 3 types of databases: MongoDB, Redis and PostgreSQL.

As for hosting the database, you have 2 options. The first one is to host it yourself on your own server (or use a database you already have, and that you may already have some data in). To do that, head over to the databases tab and click '+ connect to database'. You will then have to fill in the following details:

**name** - this is the name you use to refer to the database from blocks in the project.

**type** - type of database (MongoDB / Redis / PostgreSQL).

**clusterURL** - the url of the database.

**dbID** - the ID of the database.

**user** - a username to access the database.

**password** - a password to access the database.

Another option you have is to host the database with us, and let us take care of management, backups, scaling on our state of the art servers. To do that, just click '+ create database' and give it a name (used to access it from blocks) and choose it's type (again, MongoDB / Redis / PostgreSQL).

