|  |  |
| --- | --- |
| **Prerequisites:** | You need to first know how the Internet works, and what a Web server is. |
| **Objective:** | You will learn how to set up a local testing server. |

Local files versus remote files

You know if you are running the example from a local file because the web address will have file:// at the start of it, followed by the path to the file on your local hard drive. In contrast, if you view one of our examples hosted on GitHub (or an example on some other remote server), the web address will have http:// or https:// at the start of it, to show that the file has been received via HTTP.

The problem with testing local files

Some examples won't run if you open them as local files. This can be due to a variety of reasons, the most likely being:

**They feature asynchronous requests**. Some browsers (including Chrome) will not run async requests (see [Fetching data from the server](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data)) if you just run the example from a local file. This is because of security restrictions (for more on web security, read [Website security](https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Website_security)).

**They feature a server-side language**. Server-side languages (such as PHP or Python) require a special server to interpret the code and deliver the results.

Running a simple local HTTP server[**Edit**](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/set_up_a_local_testing_server$edit#Running_a_simple_local_HTTP_server)

To get around the problem of async requests, we need to test such examples by running them through a local web server. One of the easiest ways to do this for our purposes is to use Python's SimpleHTTPServer module.

To do this:

Install Python. If you are using Linux or Mac OS X, it should be available on your system already. If you are a Windows user, you can get an installer from the Python homepage and follow the instructions to install it:

Go to [python.org](https://www.python.org/)

Under the Download section, click the link for Python "3.xxx".

At the bottom of the page, choose the *Windows x86 executable installer* and download it.

When it has downloaded, run it.

On the first installer page, make sure you check the "Add Python 3.xxx to PATH" checkbox.

Click *Install*, then click *Close* when the installation has finished.

Open your command prompt (Windows)/terminal (OS X/Linux). To check Python is installed, enter the following command:

python -V

This should return a version number. If this is OK, navigate to the directory that your example is inside, using the cd command.

# include the directory name to enter it, for examplecd Desktop# use two dots to jump up one directory level if you need tocd ..

Enter the command to start up the server in that directory:

# On Mac and Linuxpython -m http.server# On Windowspython -m http.server

By default, this will run the contents of the directory on a local web server, on port 8000. You can go to this server by going to the URL localhost:8000 in your web browser. Here you'll see the contents of the directory listed — click the HTML file you want to run.

**Note**: If you already have something running on port 8000, you can choose another port by running the server command followed by an alternative port number, e.g. python -m SimpleHTTPServer 7800. You can then access your content at localhost:7800.

Running server-side languages locally[**Edit**](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/set_up_a_local_testing_server$edit#Running_server-side_languages_locally)

Python's SimpleHTTPServer module is useful, but it doesn't know how to run code written in languages such as PHP or Python. To handle that you'll need something more — exactly what you'll need depends on the server-side language you are trying to run. Here are a few examples:

To run Python server-side code, you'll need to use a Python web framework. You can find out how to use the Django framework by reading [Django Web Framework (Python)](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django). [Flask](http://flask.pocoo.org/) is also a good (slightly less heavyweight) alternative to Django. To run this you'll need to [install Python/PIP](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/development_environment#Installing_Python_3), then install Flask using pip3 install flask. At this point you should be able to run the Python Flask examples using for example python3 python-example.py, then navigating to localhost:5000 in your browser.

To run Node.js (JavaScript) server-side code, you'll need to use raw node or a framework built on top of it. Express is a good choice — see [Express Web Framework (Node.js/JavaScript)](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs).

To run PHP server-side code, you'll need a server setup that can interpret PHP. Good options for local PHP testing are [MAMP](https://www.mamp.info/en/downloads/) (Mac and Windows) , [AMPPS](http://ampps.com/download) (Mac, Windows, Linux) and [LAMP](https://www.linux.com/learn/easy-lamp-server-installation) (Linux, Apache, MySQL, and PHP/Python/Perl). These are complete packages that create local setups to allow you to run the Apache server, PHP, and MySQL databases.