**Build your own server in a week**

A web server is a computer system that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web. The term can refer to the entire system, or specifically to the software that accepts and supervises the HTTP requests.

1. Get to know the basic of what web server is and what are the minimal things required to build a web server (i.e. web sockets, HTTP etc).

2. Since WebSockets makes it possible to open an interactive communication session between the user's browser and a server.

Read about TCP sockets and make an ‘echo server’. This will ‘serve’ as a foundation.

3. Once the ‘echo server’ has been made, move on to know about the message syntax of HTTP requests, HTTP responses.

Source: RFC 7230 --- <https://tools.ietf.org/html/rfc7230>

4. Start with the ‘message format’ section of the RFC and once you get the format, resume your coding by writing a request parser.

* which will take the request break the request header, and response
* take the header and split it into key, value pairs making an object (or a dictionary in Python)
* similarly, depending upon the content-type of the request, parse the request body (e.g. If the content type is json parse it into an object, if the type is ‘mutipart’ parse it into an object etc.)

5. Once your request object is ready we can move forward to handle that request in order to get the response.

Here, you will have to choose about how you want to handle concurrency.

* asynchronously
* using muti-threading

6. If you choose to handle concurrency asynchronously, then after parsing your request make all the subsequent handlers in such a away that the next handler could be called using a next() function.

e.g. list\_of\_all\_handlers = [static\_handler, route\_handler, .......]

The first time when we call next() -- static\_handler is executed.

Secondly, when next() is called within the static\_handler – route handler is executed.

7. Start with the handlers to get the response.

It would be great help if you see how a backend framework works...this will save you a lot of time.

1. static\_file\_handler: Decide on a file structure that you want the application developed on your server to have.

Write the handler to server the files of the folder, you have decided will store the static files(such as: .html, .js, images....)

1. route\_handler: First, make a function that will add all routes in to an object, then the route\_handler will match the path that comes in the request with the path of the dictionary and execute the function of the coresponding path, that will give you the content for your body.
2. err\_404\_handler: If your file is not served with either of the above handlers then it has to through a 404 error.

8. In the these handlers you have to implement HTTP methods (i.e. GET, POST, PUT, DELETE etc.)

Refer RFC: <https://tools.ietf.org/html/rfc7231>

9. While you are making your handlers you will be keeping various response headers in a dictionary, which later you have to convert into a byte stream to send it to the client.