Webserver vs OS

How are webservers similar to an OS?

How are they different?

What lessons can be learned from web functionality?
What is a Webserver?

Serves files (index.html)
javascript files, css, etc.

Receives and serves requested data
From local or remote sources

Processes requests
Computations, ‘likes’, data storage, etc.
Comparison to OS

Many things need to happen at the same time

  Multiple Users, Multiple Requests

Work queues

  How does user / req get something done?

I/O behaviors

  Access to disk and other processes
Implementations

New process on request

    Process can do what compared to threads?

Work pool / work queue

    Pool of threads or process are always available

Event-Driven vs Polling

    What’s more efficient?
Server vs OS

Why are server tasks not handled by OS?

How do they work together?

Hardware abstraction, security, process safety
Spawn Processes

Server

mod_rewrite
mod_auth_*
mod_mimce
mod_alias
mod_usertrack
mod_cgi
mod_php

CGI applications
PHP-based application

Apache core

Apache conf.
System binaries
Application conf.
PHP conf.
System files
Application data

To/from client

From/to Database
char response[] = "HTTP/1.1 200 OK\n" 
"Content-Type: text/html; charset=UTF-8\n" 
"<!DOCTYPE html> <html> <head> <title>Bye-bye baby bye-bye</title> 
" <style>body { background-color: #111; } 
" h1 { font-size: 4cm; text-align: center; color: black; 
" text-shadow: 0 0 2mm red } </style> </head> 
" </body> <html> 
" <h1>Goodbye, world!</h1> </body> </html> \n"

```c
int main()
{
    int one = 1, client_fd;
    struct sockaddr_in svr_addr, cli_addr;
    socklen_t sin_len = sizeof(cli_addr);
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0)
        err(1, "can't open socket");
    setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &one, sizeof(int));
    int port = 8080;
    svr_addr.sin_family = AF_INET;
    svr_addr.sin_addr.s_addr = INADDR_ANY;
    svr_addr.sin_port = htons(port);
    if (bind(sock, (struct sockaddr *)&svr_addr, &sin_len) == -1)
        close(sock);
        err(1, "Can't bind");
    listen(sock, 5);
    while (1) {
        client_fd = accept(sock, (struct sockaddr *)&cli_addr, &sin_len);
        printf("got connection\n");
        if (client_fd == -1) {
            perror("Can't accept");
            continue;
        }
        write(client_fd, response, sizeof(response) - 1);
        close(client_fd);
    }
}
```

```javascript
Event Driven (node.js)

var http = require('http');
var SERVER_PORT = 8124;

// Creating the HTTP Server
var server = http.createServer(function(request, response){
    // Called each time a request is made.
    response.writeHead(200, {'Content-Type': 'text/plain'});
    response.end('Hello World
');
});

// Starting the server
server.listen(SERVER_PORT);

console.log('Server running on port : ' + SERVER_PORT);
```
**Work Queues**

- Event-driven
- Asynchronous
- Non-blocking
Parallelism

How to do more at the same time
Multiple cores, multiple threads
Buffered I/O, Batch processing
Nagle’s Algorithm
Cluster programming, OpenMP
Asynchronous vs Synchronous

Maximize efficiency by minimizing waiting (for resources, other processes)
Requires different programming model

Deferred processing
Coroutines
Phone call vs text
Message passing
Other examples?