Telerik Academy

NodeJS Web Server

Serving HTTP requests

End-to-end JavaScript Applications

Telerik Software Academy

http://academy.telerik.com

Telerik Academy

Table of Contents

Telerik Academy

The HTTP Protocol

How HTTP Works?

HTTP

- Hyper Text Transfer Protocol (HTTP)
  - Client-server protocol for transferring Web resources (HTML files, images, styles, etc.)
- Important properties of HTTP
  - Request-response model
  - Text-based format
  - Relies on unique resource URLs
  - Provides resource metadata (e.g. encoding)
  - Stateless (cookies can overcome this)

HTTP: Request-Response Protocol

- Client program
  - Running on end host
  - E.g. Web browser
  - Requests a resource
- Server program
  - Running at the server
  - E.g. Web server
  - Provides resources

Telerik Academy

Example: Hyper Text Transfer Protocol

- HTTP request:

```
GET /academy/about.aspx HTTP/1.1
Host: www.telerik.com
User-Agent: Mozilla/5.0
<CRLF>
```

The empty line denotes the end of the request header

- HTTP response:

```
HTTP/1.1 200 OK
Date: Mon, 5 Jul 2010 13:09:03 GMT
Server: Microsoft-HTTPAPI/2.0
Last-Modified: Mon, 12 Jul 2010 15:33:23 GMT
Content-Length: 54
<CRLF>
<html><title>Hello</title>
Welcome to our site</html>
```

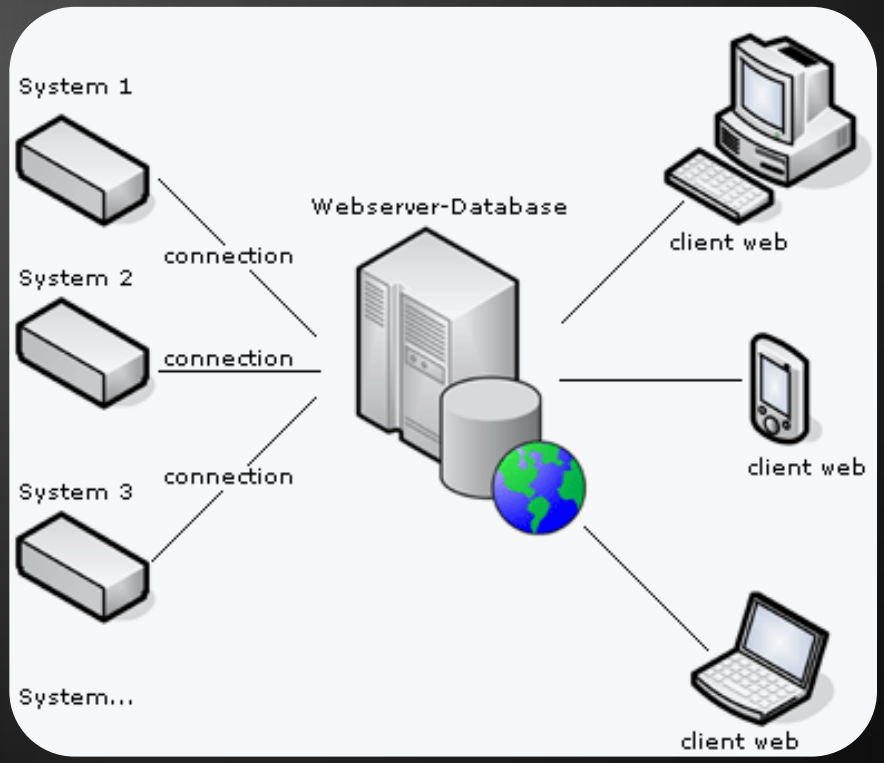
The empty line denotes the end of the response header

HTTP Response Codes

- HTTP response code classes
  - 1xx: informational (e.g., "100 Continue")
  - 2xx: success (e.g., "200 OK")
  - 3xx: redirection (e.g., "304 Not Modified", "302 Found")
  - 4xx: client error (e.g., "404 Not Found")
  - 5xx: server error (e.g., "503 Service Unavailable")
- "302 Found" is used for redirecting the Web browser to another URL

Telerik Academy

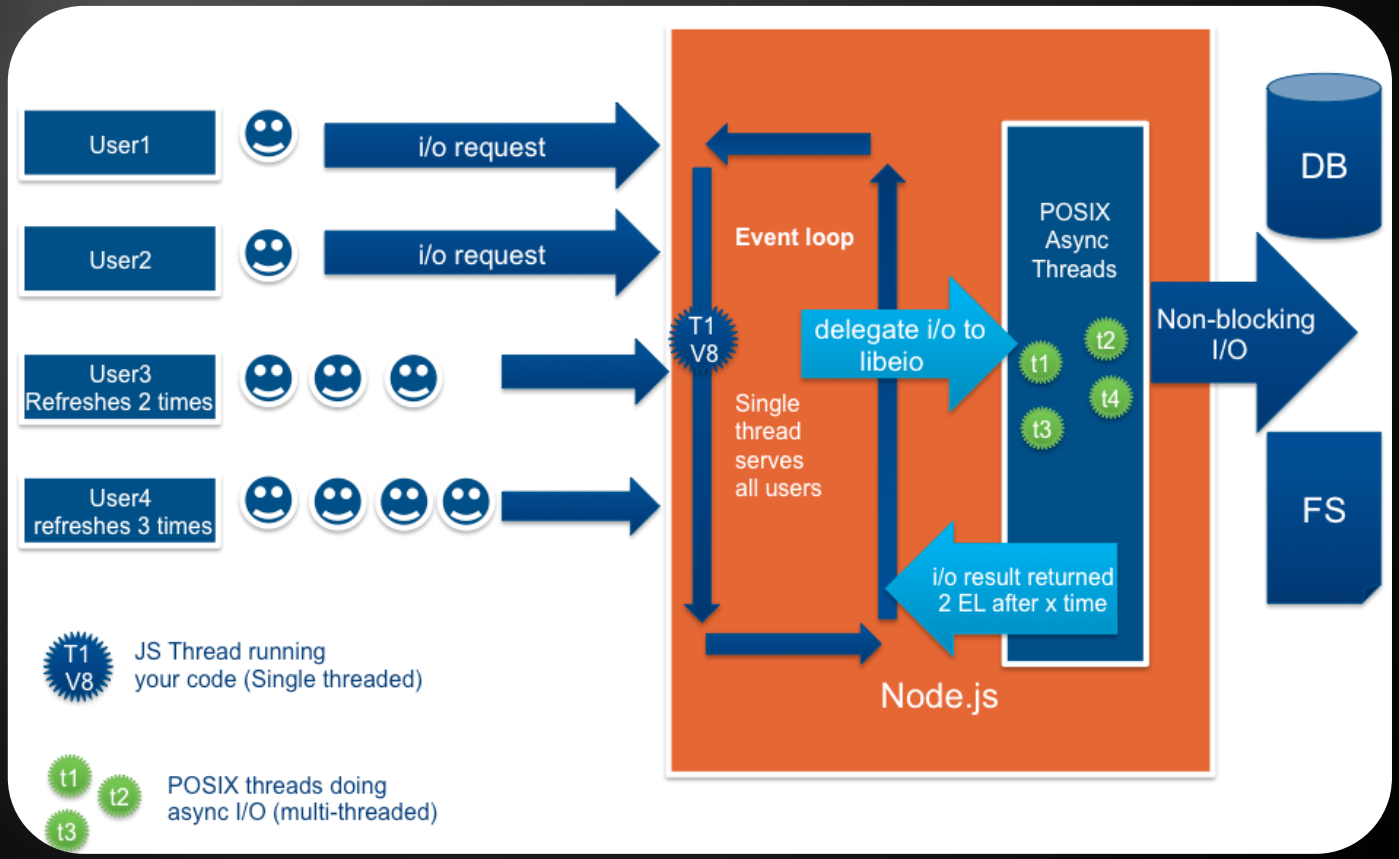
What is a Web Server



What Do the Web Servers Do?

- All physical servers have hardware
- The hardware is controlled by the operating system
- Web servers are software products that use the operating system to handle web requests
  - Web servers serve Web content
- These requests are redirected to other software products (ASP.NET, PHP, etc.), depending on the web server settings

NodeJS Web Server



Telerik Academy

NodeJS Web Server

- Require the 'http' module
  - Create server function (createServer)
  - Request/Response wrapper objects
  - Listen function (specifies port and IP (host))
  - Headers are objects (keys are lowercased)

```
{ 'content-length': '123',
  'content-type': 'text/plain',
  'connection': 'keep-alive',
  'accept': '*/*' }
```

Telerik Academy

NodeJS Web Server

- Basic server implementation

```
var http = require('http');

http.createServer(function(req, res) {
    res.writeHead(200, {
        'Content-Type': 'text/plain'
    }); //return success header

    res.write('My server is running! ^_^'); //response
    res.end(); //finish processing current request
}).listen(1234);
```

- http://nodejs.org/api/http.html

Telerik Academy

NodeJS Basic Web Server

Live Demo

Telerik Academy

The Request Wrapper

Telerik Academy

The Request Wrapper

- The Request wrapper
  - http.IncommingMessage class [docs]
  - Implements the Readable Stream interface
- Properties
  - httpVersion – '1.1' or '1.0'
  - headers – object for request headers
  - method – 'GET', 'POST', etc
  - url – the URL of the request

Telerik Academy

The Request Wrapper

Live Demo

Telerik Academy

The Response Wrapper

Telerik Academy

The Response Wrapper

- The Response wrapper
  - http.ServerResponse class [docs]
  - Implements the Writable Stream interface
- Methods
  - writeHead(statusCode, [headers])

```
var body = 'hello world';
response.writeHead(200, {
  'Content-Length': body.length, // not always valid
  'Content-Type': 'text/plain',
  'Set-Cookie': ['type=ninja', 'language=javascript']
});
```

Telerik Academy

The Response Wrapper

- Methods
  - write(chunk, [encoding])

```
response.writeHead('Hello world!'); // default encoding: utf8
```

  - end()
- Always call the methods in the following way
  - writeHead
  - write
  - end

Telerik Academy

The Response Wrapper

Live Demo

Telerik Academy

Route Requests

Route Requests

- URL Parsing modules
  - 'url' and 'querystring'
  - both have parse method

```
var url = require('url');
console.log(url.parse('/status?name=ryan', true));

// logs
// {
//    href: '/status?name=ryan',
//    search: '?name=ryan',
//    query: { name: 'ryan' },
//    pathname: '/status'
// }
```

Telerik Academy

Route Requests

Live Demo

Telerik Academy

NodeJS as Client

NodeJS As Client

- http.request and http.get
  - both have (options, callback) signature

```
var req = http.request(options, function(res) {
  console.log('STATUS: ' + res.statusCode);
  console.log('HEADERS: ' + JSON.stringify(res.headers));
  res.setEncoding('utf8');
  res.on('data', function (chunk) {
    console.log('BODY: ' + chunk);
  });
});

http.get("http://www.google.com/index.html", function(res) {
  console.log("Got response: " + res.statusCode);
})
```

Resources

- http://nodejs.org/ - NodeJS official web site
- http://nodejs.org/api/ - API documentation
- http://blog.nodejitsu.com/npm-cheatsheet - NPM documentation
- https://npmjs.org/ - NPM official web site
- https://github.com/felixge/node-style-guide - NodeJS style guide

Telerik Academy

NodeJS Web Server

http://academy.telerik.com

Exercises

1. **Create file upload web site with NodeJS. You should have the option to upload a file and be given an unique URL for its download. Use GUID.**
   - You are not allowed to use ExpressJS