Web Development

COMP 431 / COMP 531

Web Servers

Scott E Pollack, PhD

March 8, 2016

Part II – Back End Development

- Homework Assignment 5 (Front-End App)
  - **Due THURSDAY 3/10**

Homework Assignment 6
(Draft Back-End)
Due Thursday 3/24

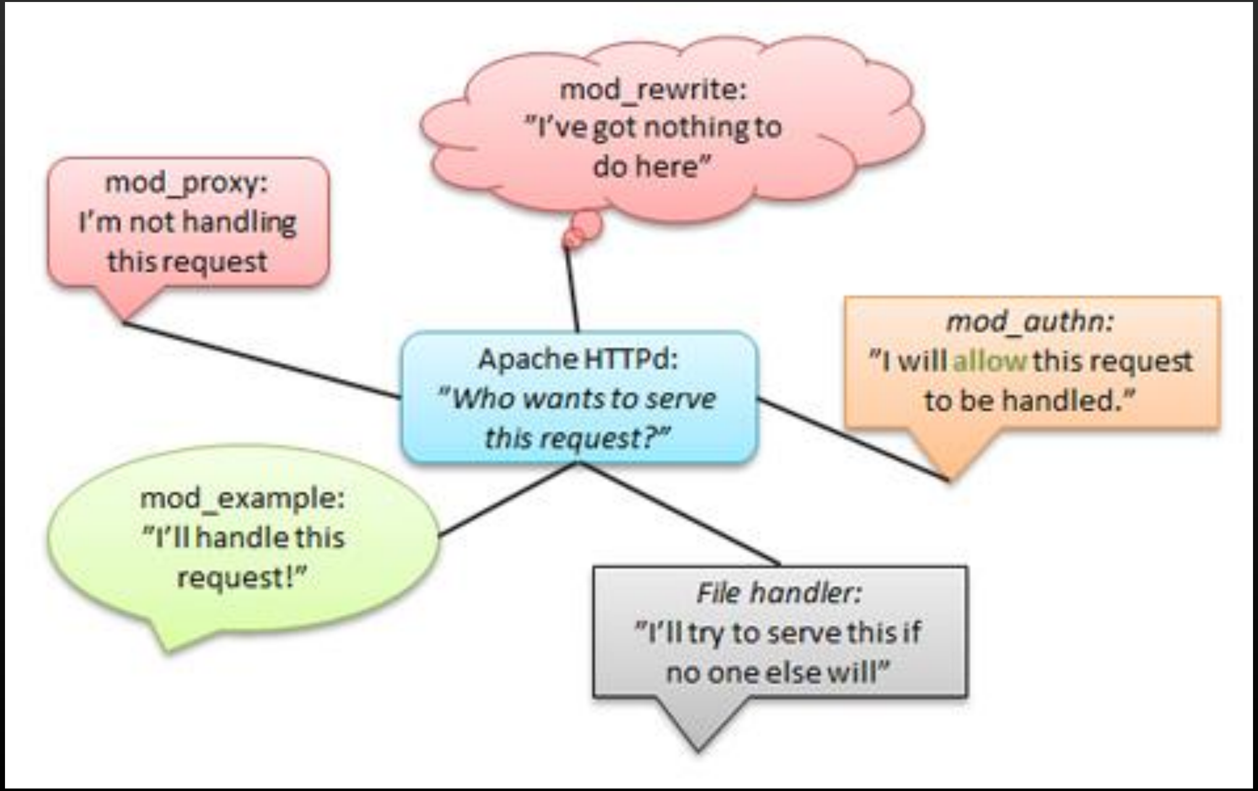
PART II
Web Servers
Backend
Architecture
Unit Testing
Web Hosting
Databases

In the beginning…

- 1990 – There was CERN’s httpd
- Then came NCSA HTTPd and the introduction of the Common Gateway Interface (CGI)
- 1995 – Apache hit the scene
- 1996 – MS IIS introduced
- 2002 – Nginx introduced
- 2009 – Apache powered over 100 million websites
- June 2013 – Apache serves ~54% of all websites

Apache HTTP Server

- Written in C, runs as a daemon (httpd)
- Typically used for static serving, CGI, or as a proxy
- Based on modules

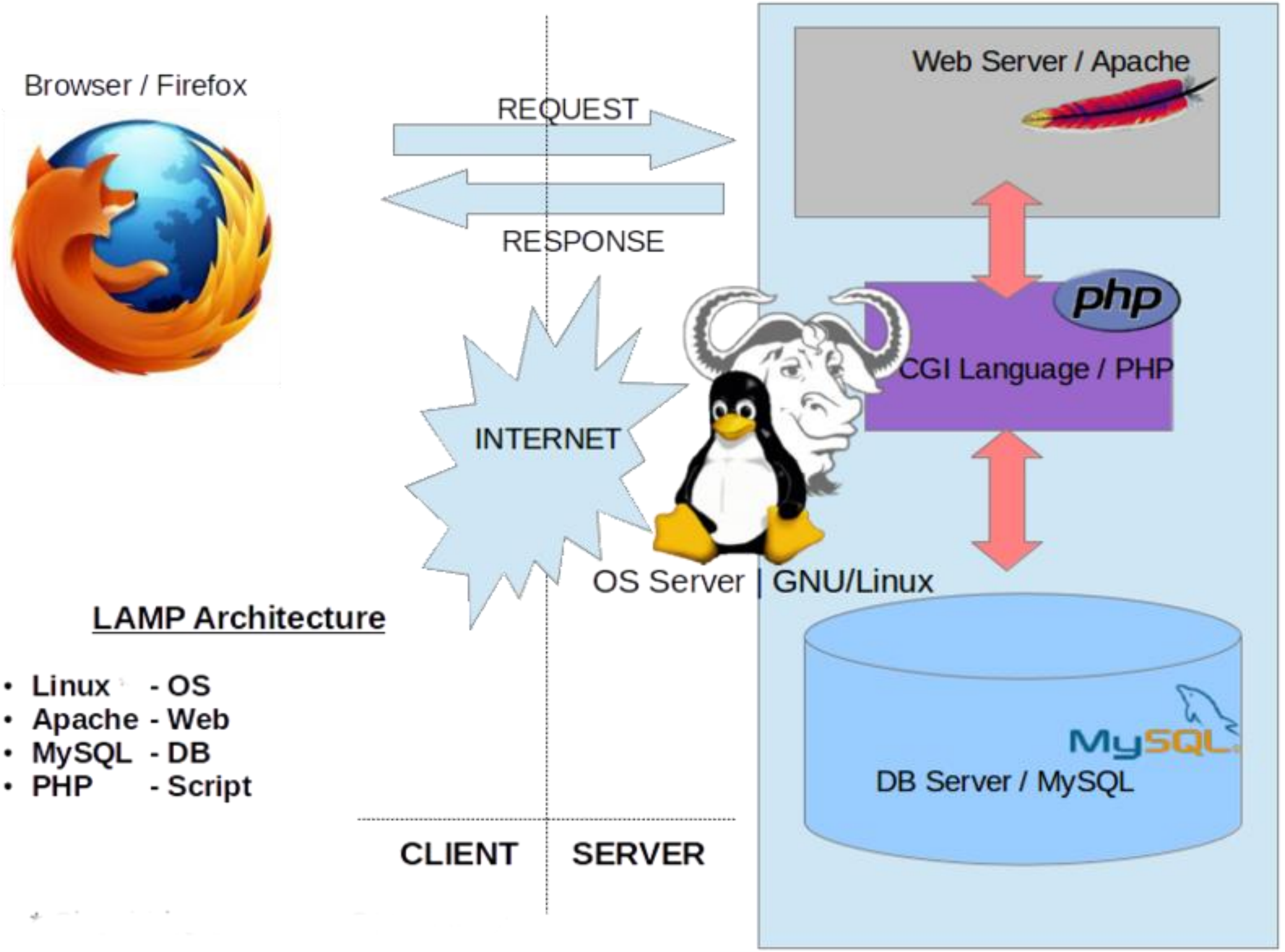

mod_rewrite: "I've got nothing to do here"
mod_proxy: I'm not handling this request
Apache HTTPd: "Who wants to serve this request?"
mod_authn: "I will allow this request to be handled."
mod_example: "I'll handle this request!"
File handler: "I'll try to serve this if no one else will"

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>
. . .
```

Apache Bundles

- WAMP, LAMP, MAMP, XAMPP

Platform	Windows | Linux | Mac
Web Server	Apache
Database	MySQL
Server Side Scripting	PHP/Perl/Python

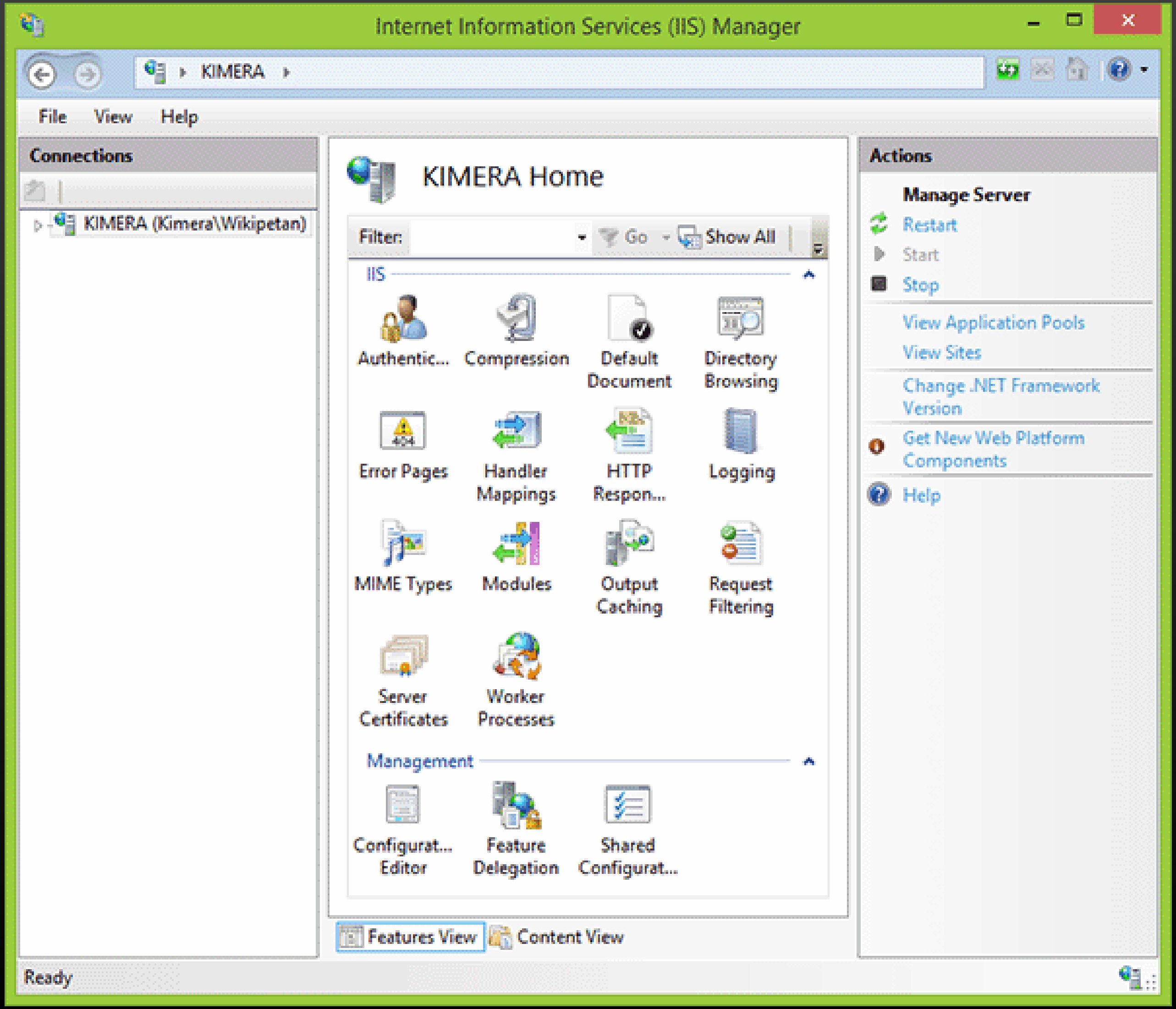


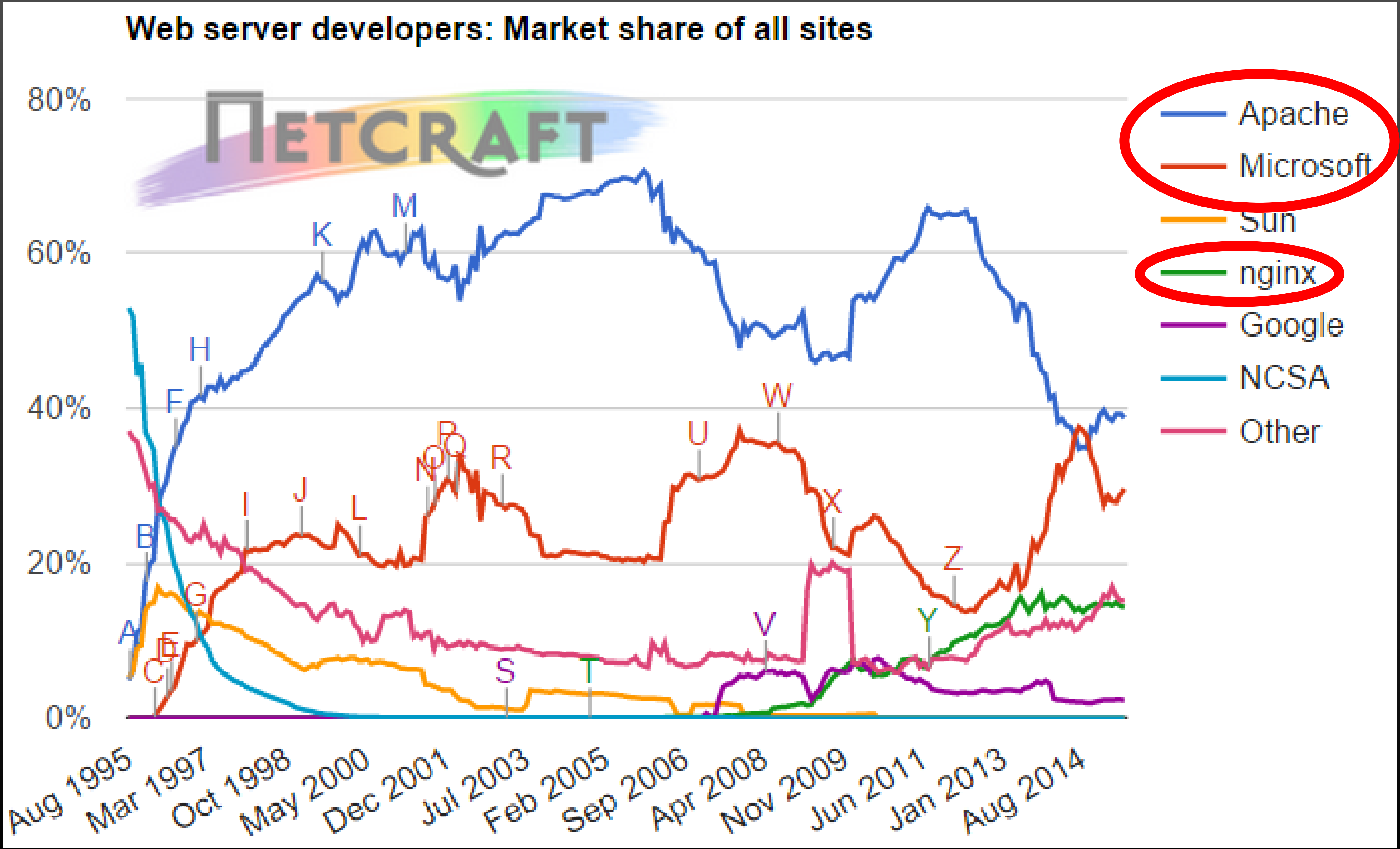
LAMP Architecture

- Linux - OS
- Apache - Web
- MySQL - DB
- PHP - Script

Internet Information Services

- Integrates well with MS products
- Comparable to Apache in many ways
- … what more needs to be said?

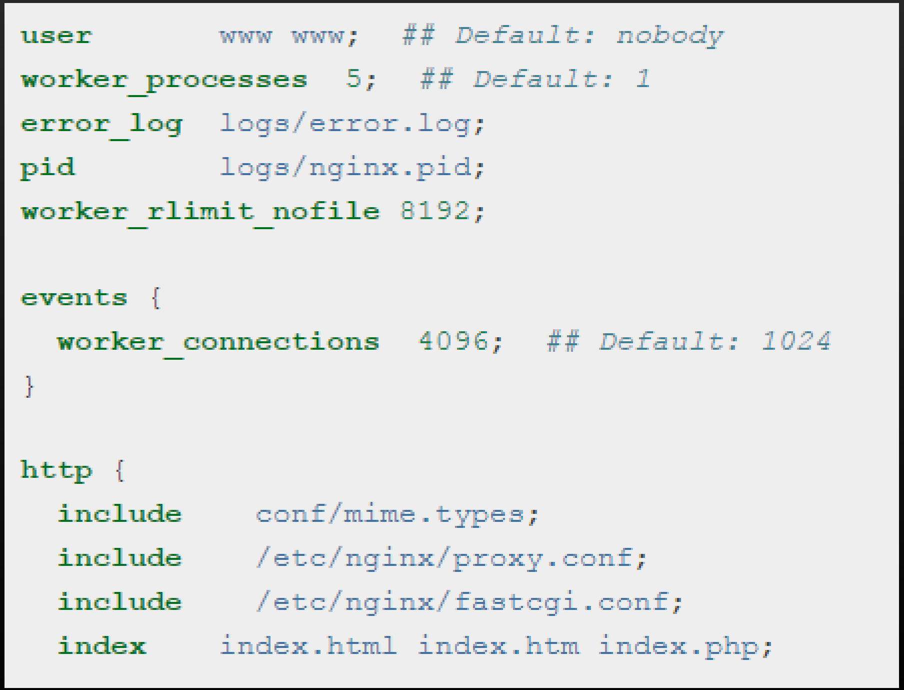




Web server developers: Market share of all sites
80%
60%
40%
20%
0%
NETCRAFT
Apache
Microsoft
Sun
nginx
Google
NCSA
Other
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
Aug 1995
Mar 1997
Oct 1998
May 2000
Dec 2001
Jul 2003
Feb 2005
Sep 2006
Apr 2008
Nov 2009
Jun 2011
Jan 2013
Aug 2014

Nginx

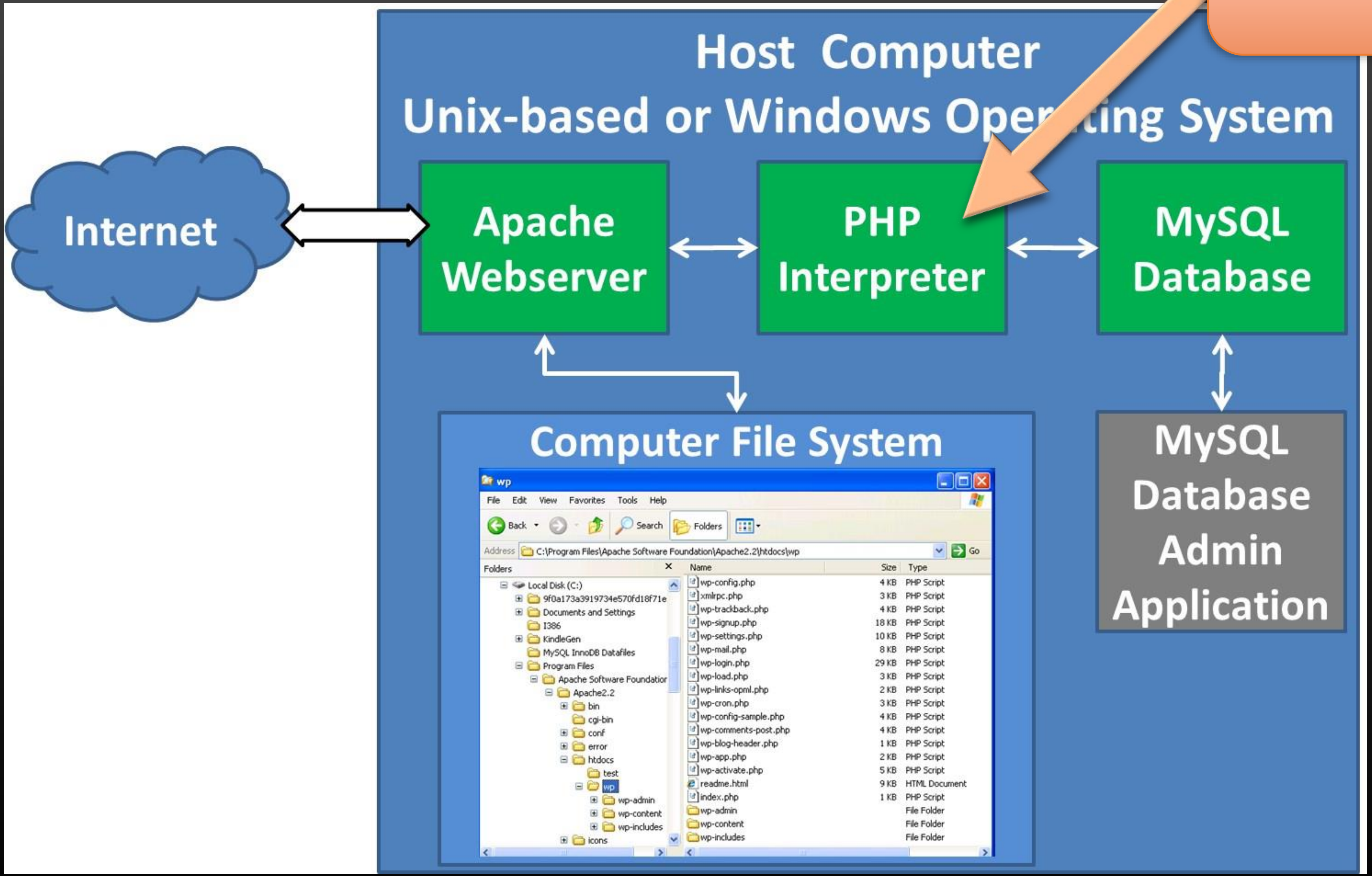
- The web server racecar: fast and low memory usage
- Also typically used for static serving, CGI, or as a proxy
- Designed to solve the problem of massive concurrent connections
  - Apache can begin to choke
- Designed as a proxy in mind
  - Setting up Apache as a proxy requires a mod
- Nginx not as extensible (limited modules)
  - So use Apache back proxy!



```
user       www www;  ## Default: nobody
worker_processes  5;  ## Default: 1
error_log  logs/error.log;
pid        logs/nginx.pid;
worker_rlimit_nofile 8192;

events {
  worker_connections  4096;  ## Default: 1024
}

http {
  include    conf/mime.types;
  include    /etc/nginx/proxy.conf;
  include    /etc/nginx/fastcgi.conf;
  index    index.html index.htm index.php;
```

Server-Side Scripting

Apache has a mod to execute this



Name	Size	Type
wp-config.php	4 KB	PHP Script
xmlrpc.php	3 KB	PHP Script
wp-trackback.php	4 KB	PHP Script
wp-signup.php	18 KB	PHP Script
wp-settings.php	10 KB	PHP Script
wp-mail.php	8 KB	PHP Script
wp-login.php	29 KB	PHP Script
wp-load.php	3 KB	PHP Script
wp-links-opml.php	2 KB	PHP Script
wp-cron.php	3 KB	PHP Script
wp-config-sample.php	4 KB	PHP Script
wp-comments-post.php	4 KB	PHP Script
wp-blog-header.php	1 KB	PHP Script
wp-app.php	2 KB	PHP Script
wp-activate.php	5 KB	PHP Script
readme.html	9 KB	HTML Document
index.php	1 KB	PHP Script
wp-admin		File Folder
wp-content		File Folder
wp-includes		File Folder

http://troubleshootingwp.com/wp-content/uploads/2013/01/ApacheMySQLwithAdmin.jpg

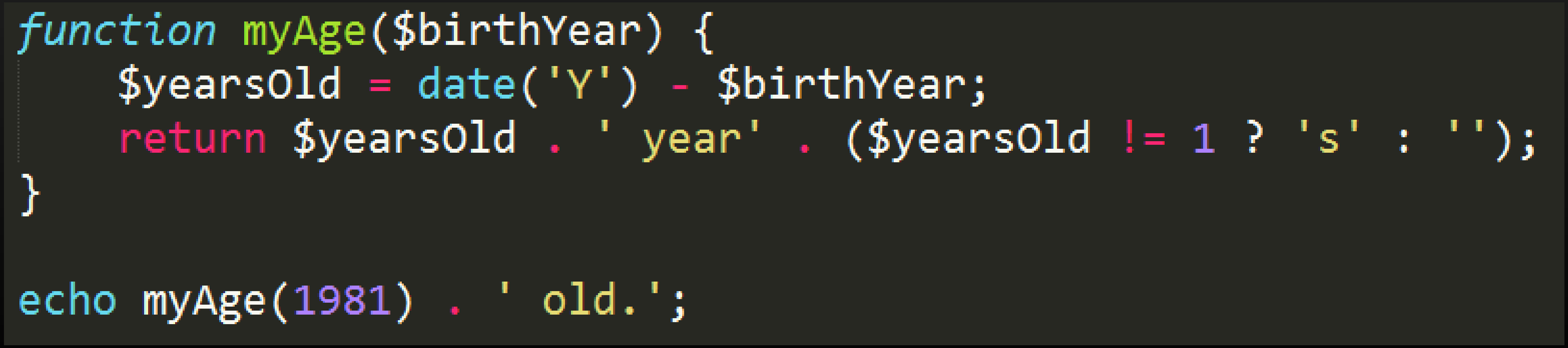
Common Gateway Interface (CGI)

- "gateway" from web server to a file on disk that is executed
- The output from the executed script is the response to the request
- Typically place scripts in /cgi-bin
  - Sometimes with extension .cgi *(regardless of content)*
- Can write cgi in Perl, bash, VB, C, FORTRAN, AppleScript, …
  - For non-scripts you must compile, typically from /cgi-src to /cgi-bin
- Each execution spins up a new process and executes the script

- FastCGI – pool of managed processes. Much faster!

PHP Hypertext Preprocessor

- In 1994 Rasmus Lerdorf wrote CGI and extended it to include web forms capability which he called Personal Home Page / Forms Interpreter
- PHP grew organically out of control…
- “Similar to Perl in syntax, it is simpler, more limited, and less consistent”



```php
function myAge($birthYear) {
    $yearsOld = date('Y') - $birthYear;
    return $yearsOld . ' year' . ($yearsOld != 1 ? 's' : '');
}

echo myAge(1981) . ' old.';
```

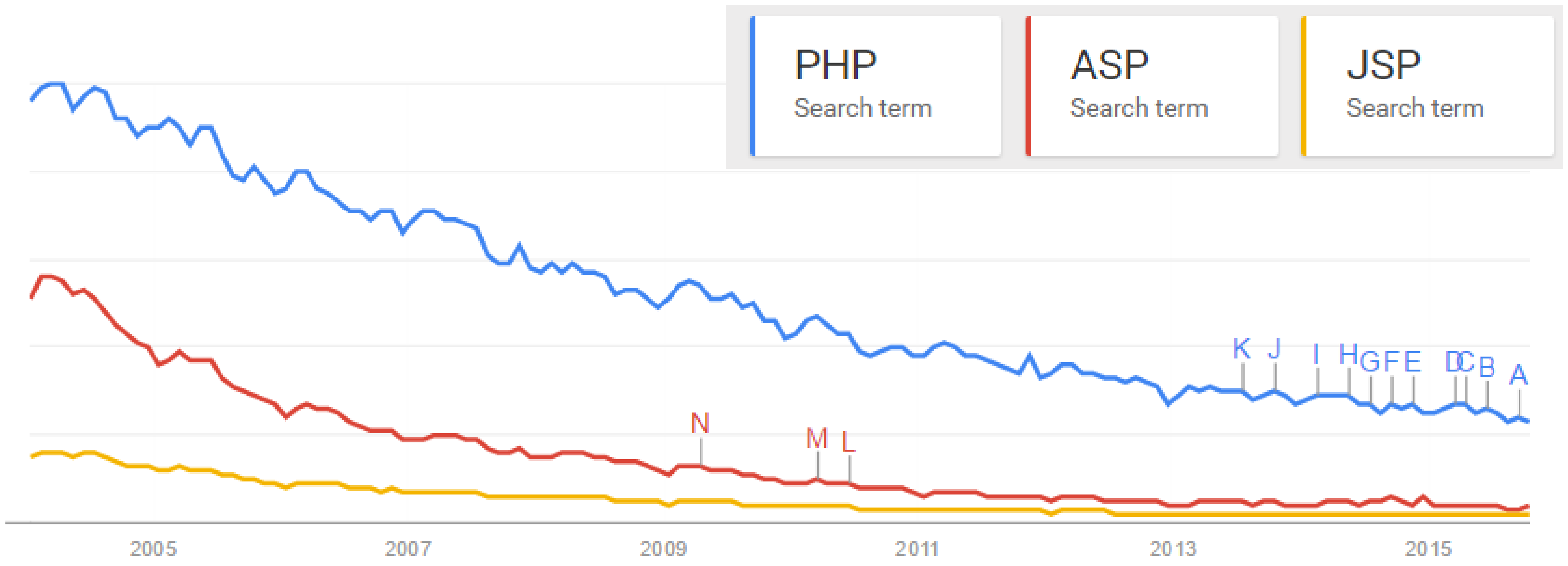
OOP in PHP

```php
class Person
{
    public $firstName;
    public $lastName;

    public function __construct($firstName, $lastName = '') {
        $this->firstName = $firstName;
        $this->lastName  = $lastName;
    }

    public function greet() {
        return 'Hello, my name is ' . $this->firstName .
               (($this->lastName != '') ? (' ' . $this->lastName) : '') . '.';
    }

    public static function staticGreet($firstName, $lastName) {
        return 'Hello, my name is ' . $firstName . ' ' . $lastName . '.';
    }
}

$he    = new Person('John', 'Smith');
$she   = new Person('Sally', 'Davis');
$other = new Person('iAmine');

echo $he->greet(); // prints "Hello, my name is John Smith."
echo '<br />';
```

Symfony

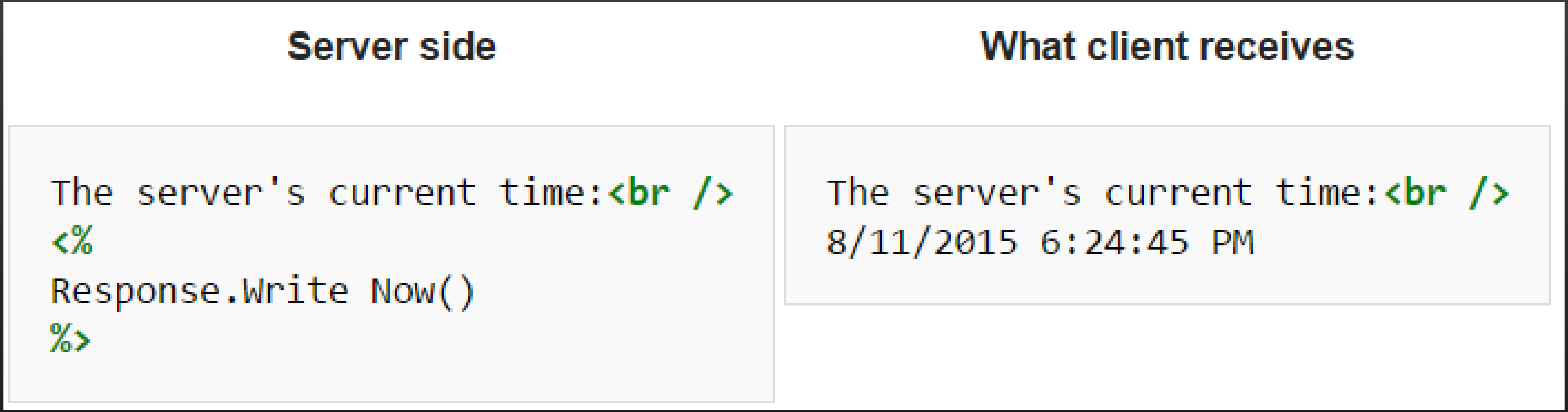
Laravel

Trends


PHP
Search term
ASP
Search term
JSP
Search term
K
J
I
H
G
F
E
D
C
B
A
N
M
L
2005
2007
2009
2011
2013
2015

Active Server Pages

index.asp



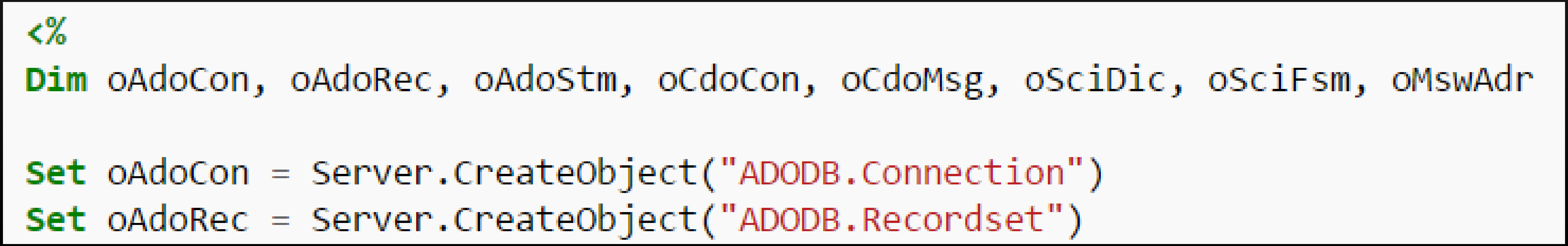
Server side

```
The server's current time:<br />
<%
Response.Write Now()
%>
```

What client receives

```
The server's current time:<br />
8/11/2015 6:24:45 PM
```

ASP written in VBScript (theoretically JScript or PerlScript possible)



```
<%
Dim oAdoCon, oAdoRec, oAdoStm, oCdoCon, oCdoMsg, oSciDic, oSciFsm, oMswAdr

Set oAdoCon = Server.CreateObject("ADODB.Connection")
Set oAdoRec = Server.CreateObject("ADODB.Recordset")
```

Java Servlets

- A servlet is a Java object that receives requests and provides responses

- Servlets live within a container
  which is typically also the web server

- Each Servlet is a singleton

- Tomcat and Jetty are each a Java based web server and servlet container

~~Jakarta~~ Apache Tomcat

Eclipse Jetty

jetty://

Java Server Pages

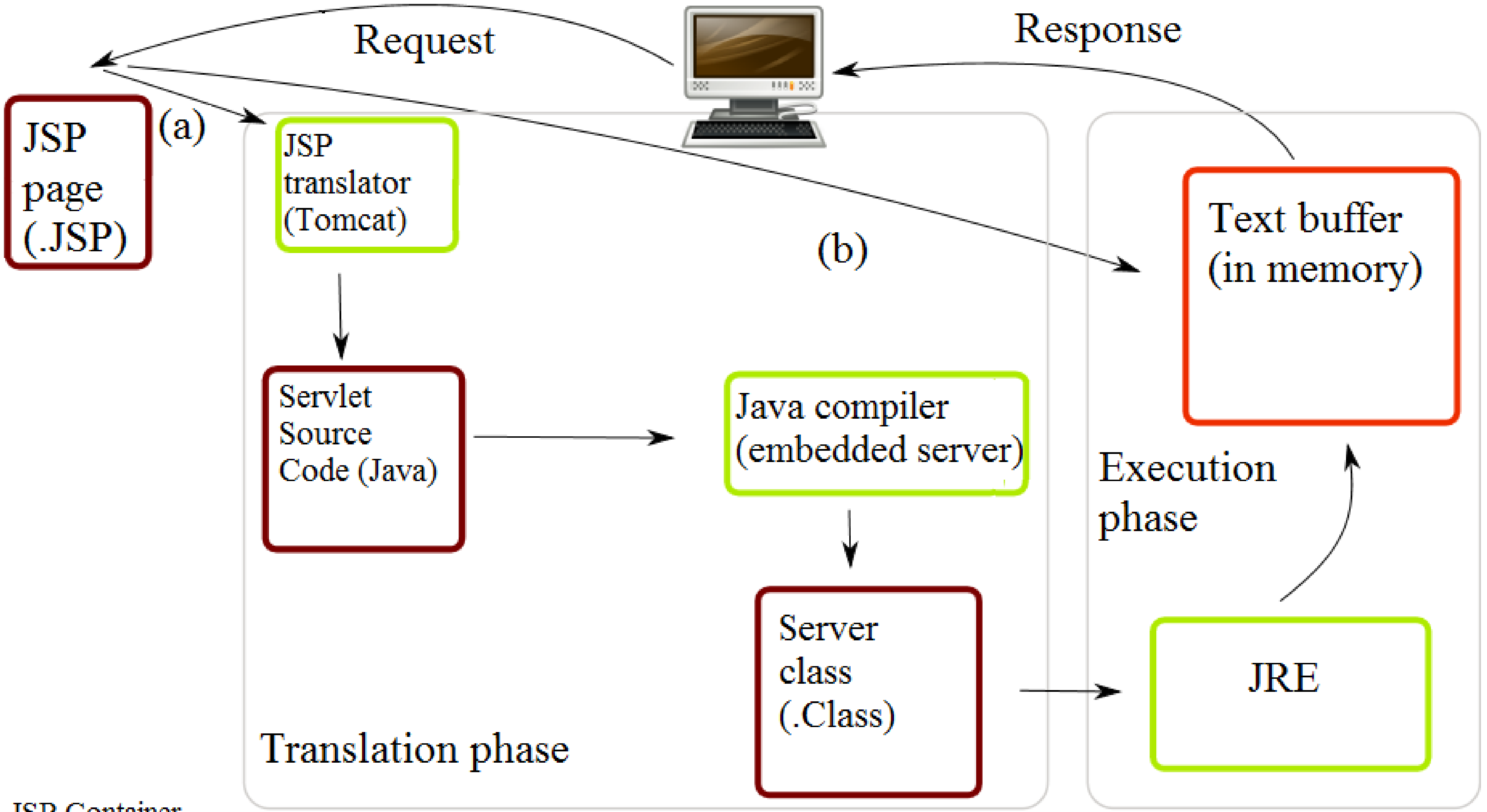
index.jsp

- A JSP is an abstraction of a Java servlet
- JSP (text file) is translated into a servlet at runtime
    - Each is cached and reused
- JSP can therefore be dynamically updated
- Follows MVC paradigm
    - M = JavaBean
    - V = imbedded HTML
    - C = servlet

```
<p>Counting to three:</p>
<% for (int i=1; i<4; i++) { %>
    <p>This number is <%= i %>.</p>
<% } %>
<p>OK.</p>
```

Groovy Server Pages

index.gsp



JSP Container
(a) Translation occurs at this point, if JSP has been changed or is new.
(b) If not, translation is skipped.

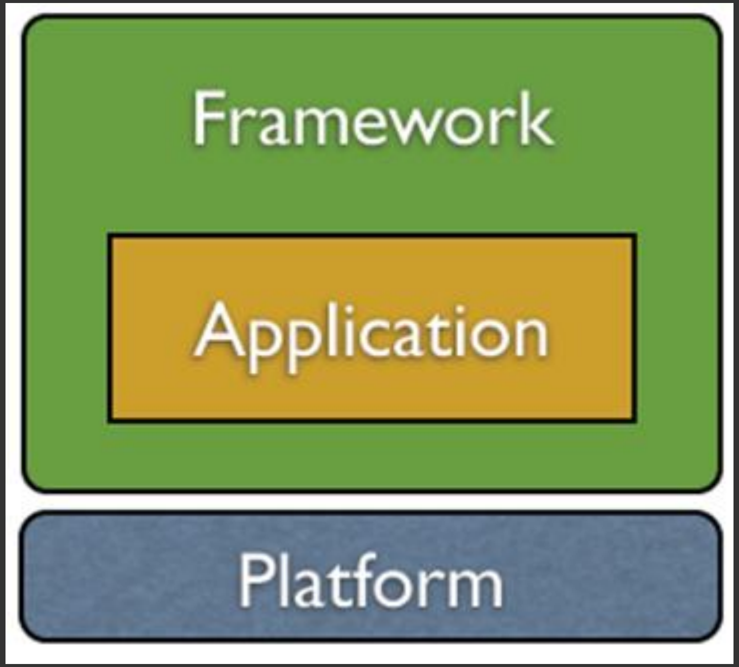
can’t cover everything…

- C#.NET (VB.NET if you must) → ASP.NET
- Java Spring
- Google Web Toolkit
- Ruby on Rails .or. Groovy Grails
- Ruby Sinatra .or. Scalatra
- Play Framework for Java & Scala
- Eclipse RAP (Remote App Platform .or. Rich Ajax Platform)
- …

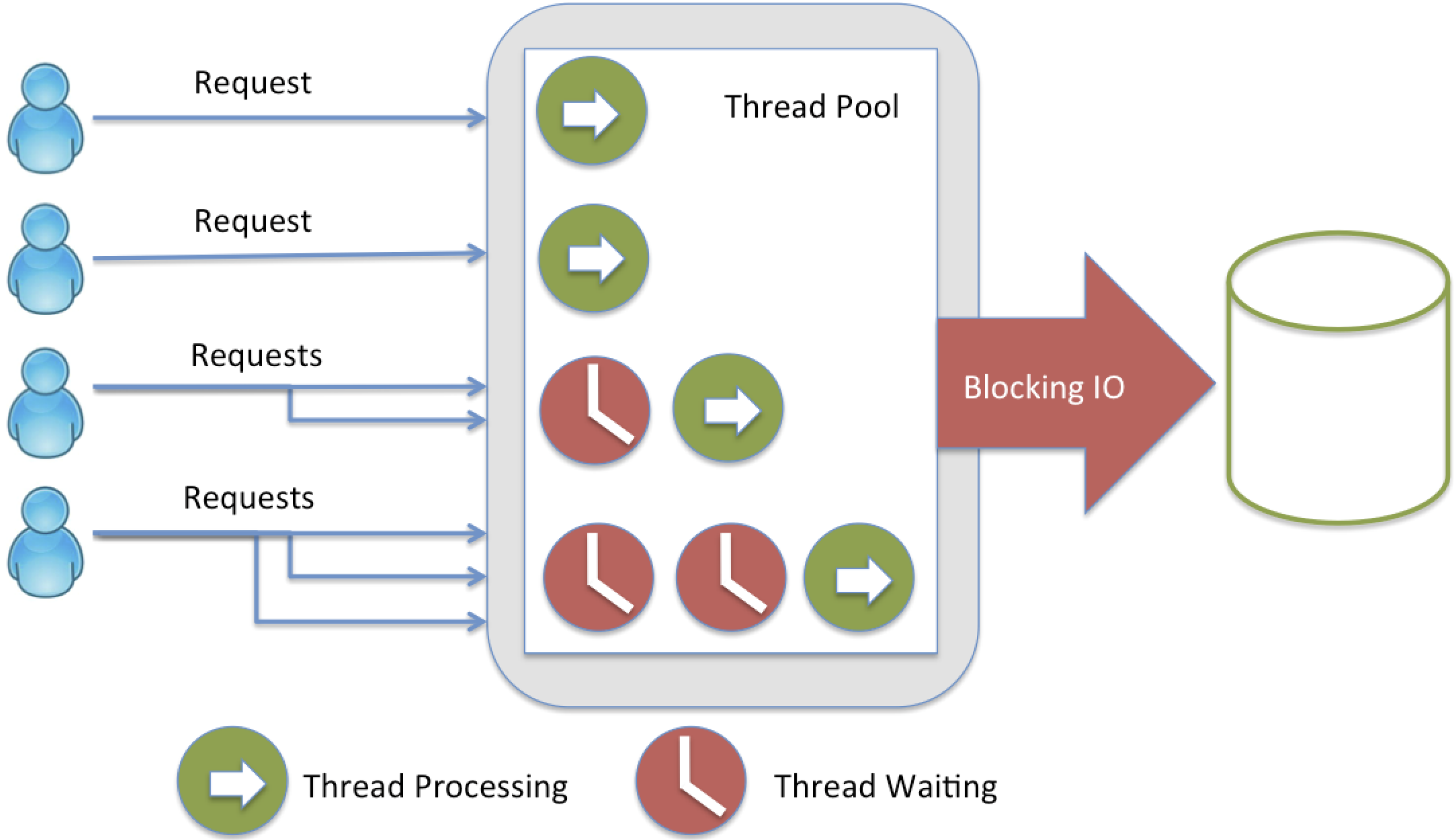
Node JS

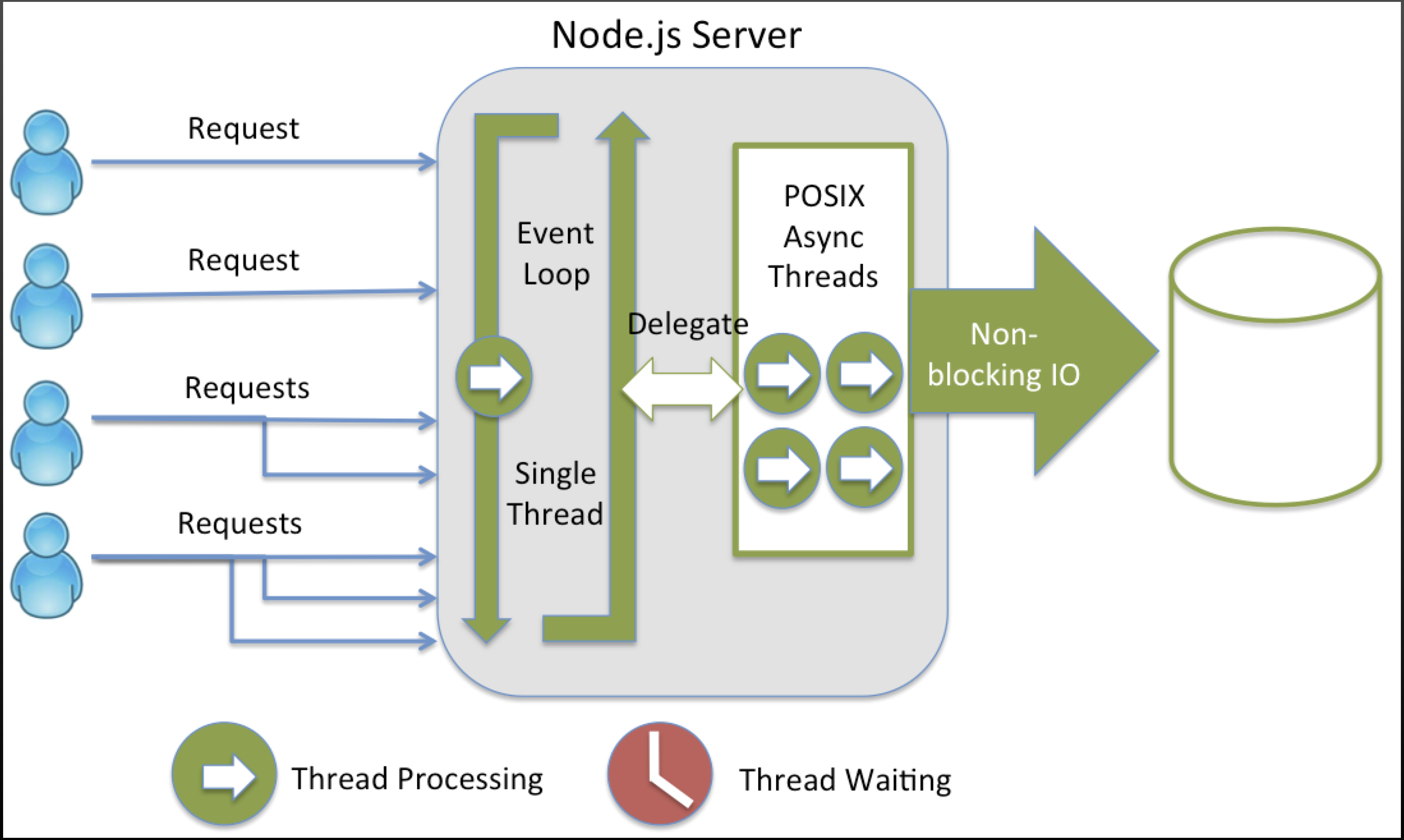
- 2009 – Invented by Ryan Dahl at Joyent *(virtualization+cloud computing)*
- 2011 – npm created by Isaac Schlueter
- 2014 – Timohy Fontaine is new lead
- June 2015 – Node.js Foundation

- Operating system agnostic
- Built on Google's V8 JavaScript engine
- asynchronous, event driven, single thread
- Non-blocking and Event driven I/O
- Data Intensive Real-Time (DIRT)
- Node is a **platform** (not a framework)

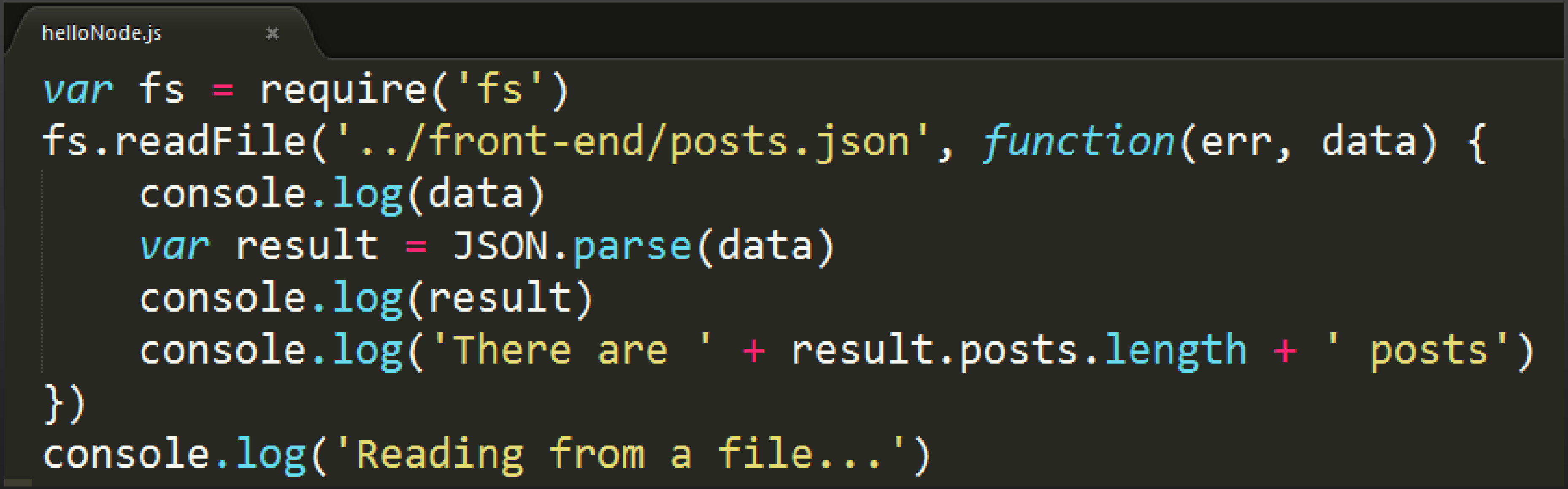


Multi Threaded Server

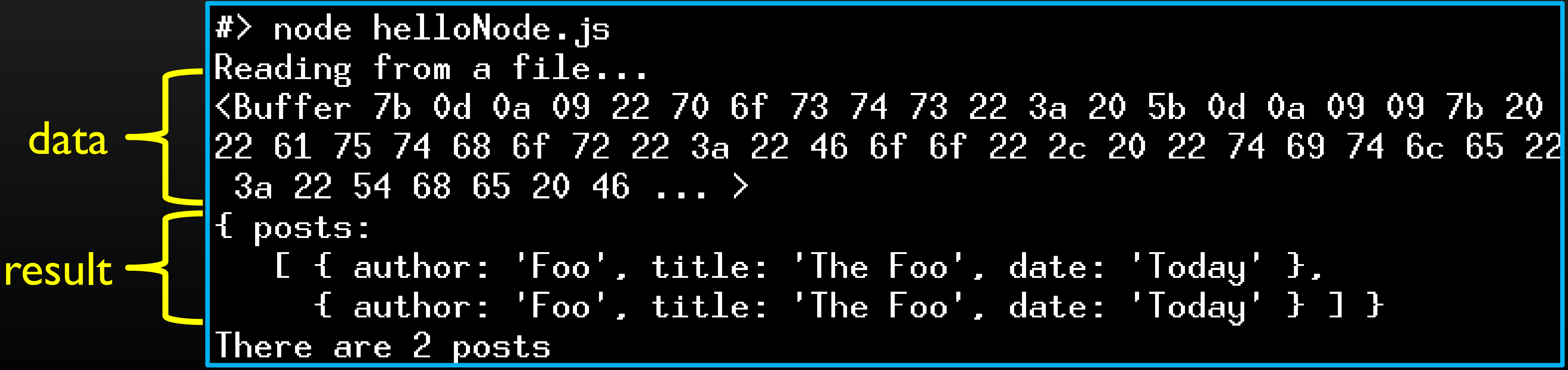



Node.js Server
Request
Request
Requests
Requests
Event Loop
Single Thread
Delegate
POSIX Async Threads
Non-blocking IO
Thread Processing
Thread Waiting

Asynchronous I/O



```
var fs = require('fs')
fs.readFile('../front-end/posts.json', function(err, data) {
    console.log(data)
    var result = JSON.parse(data)
    console.log(result)
    console.log('There are ' + result.posts.length + ' posts')
})
console.log('Reading from a file...')
```



```
#> node helloNode.js
Reading from a file...
<Buffer 7b 0d 0a 09 22 70 6f 73 74 73 22 3a 20 5b 0d 0a 09 09 7b 20
22 61 75 74 68 6f 72 22 3a 22 46 6f 6f 22 2c 20 22 74 69 74 6c 65 22
 3a 22 54 68 65 20 46 ... >
{ posts:
   [ { author: 'Foo', title: 'The Foo', date: 'Today' },
     { author: 'Foo', title: 'The Foo', date: 'Today' } ] }
There are 2 posts
```

Simple Server

```
var port = 3000
var http = require('http')
http.createServer(function(req, res) {
    console.log('Request called')
    res.writeHead(200, { 'Content-Type': 'text/plain' })
    res.end('Hello World\n')
}).listen(port)
console.log('Server listening on port ' + port)
```

```
#> node helloNode.js
Server listening on port 3000
Request called
```

```
#> curl http://localhost:3000
Hello World
```

In-Class Exercise: Simple Node Server

- We use **vanilla** NodeJS to make a simple server
  - Download http://www.clear.rice.edu/comp431/sample/simple.js
- Start the server: `node simple.js`
- Make a request: `curl -i http://127.0.0.1:3333/`
- Add the following logic

Request	Response
`GET /`	`{ hello: ‘world’ }`
`GET /posts`	`{ posts: [ { id:1 author: ‘Scott’, body:’A post’ }, … add two more posts ] }`
`POST /login` *with payload* `{ username: …, password: … }`	`{ username: <username>, result: ‘success’ }`
`PUT /logout`	`‘OK’`

***Submit* simple.js *to* COMP431-S16:inclass-15**