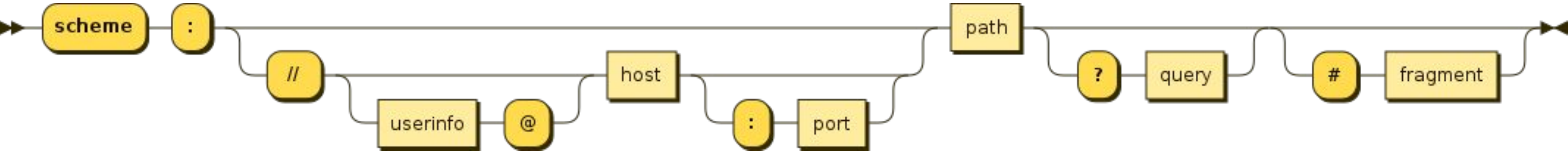
CS 576 – Systems Security
Introduction to the WWW

Georgios (George) Portokalidis

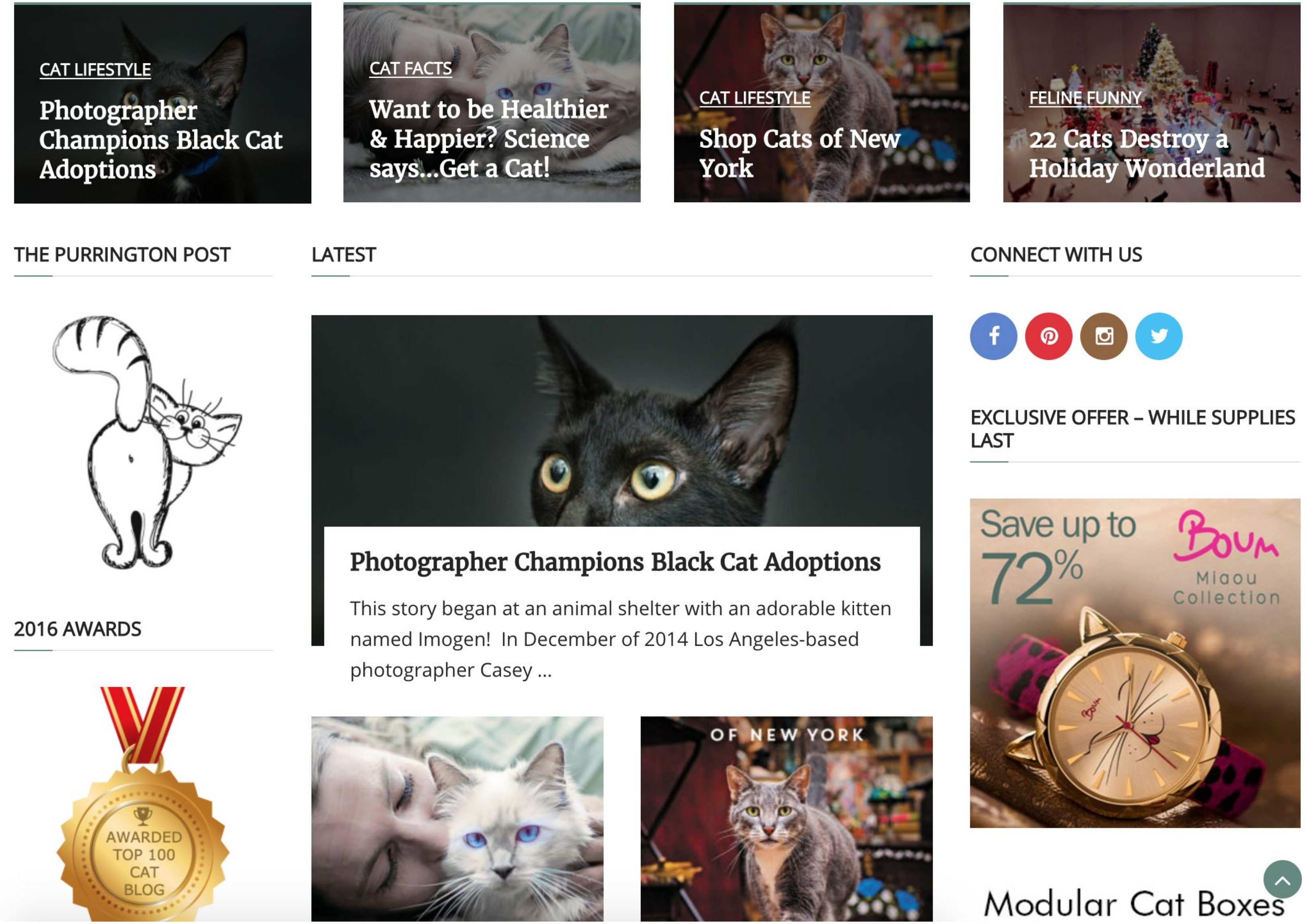
The World Wide Web (WWW)

- Commonly known as the Web, is an information system where documents and other web resources are identified by Uniform Resource Locators (URLs) –Wikipedia
- URL Scheme



- Most common schemes include http and https
  - Example: http://www.example.com:8080/questions/3456/my-document?q=10

Mostly Cat Photos and Videos



CAT LIFESTYLE

Photographer Champions Black Cat Adoptions

CAT FACTS

Want to be Healthier & Happier? Science says...Get a Cat!

CAT LIFESTYLE

Shop Cats of New York

FELINE FUNNY

22 Cats Destroy a Holiday Wonderland

THE PURRINGTON POST

2016 AWARDS

LATEST

Photographer Champions Black Cat Adoptions

This story began at an animal shelter with an adorable kitten named Imogen! In December of 2014 Los Angeles-based photographer Casey ...

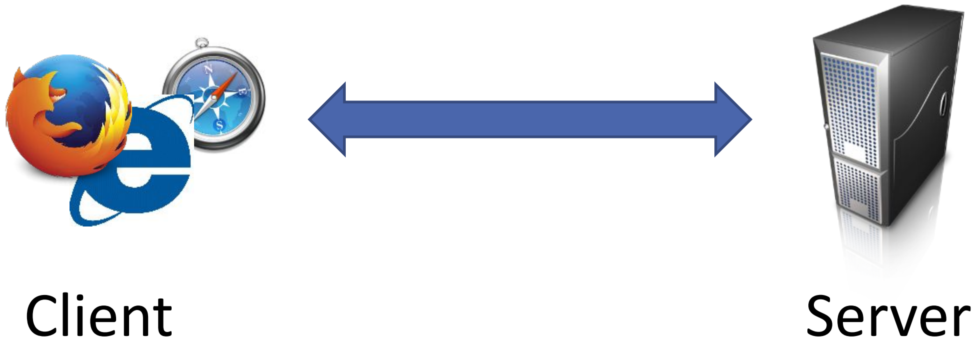
CONNECT WITH US

EXCLUSIVE OFFER – WHILE SUPPLIES LAST

Modular Cat Boxes

Web Applications

- A web application (or web app) is application software that runs on a web server … Web applications are accessed by the user through a web browser with an active network connection. – Wikipedia



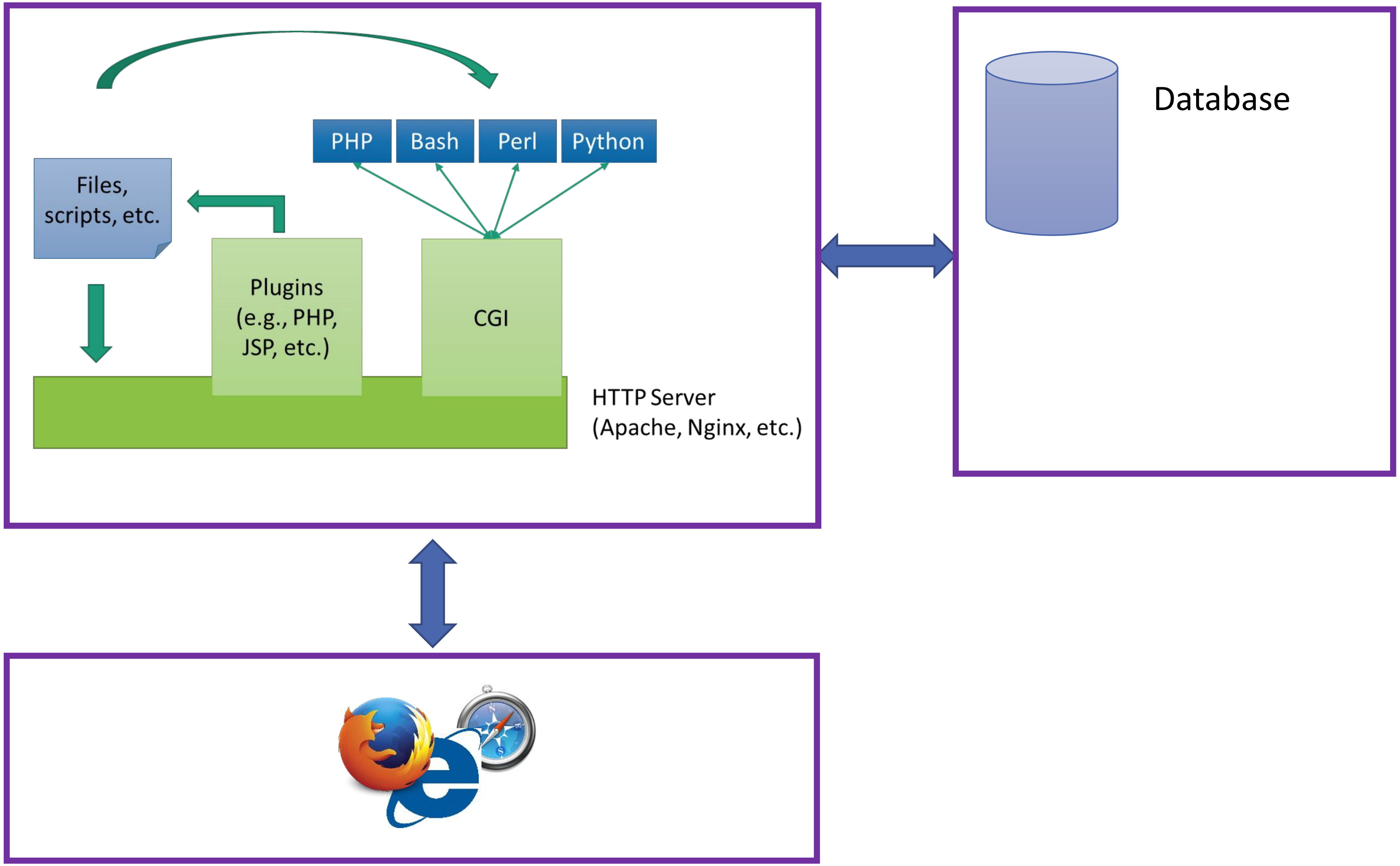
Client

Server

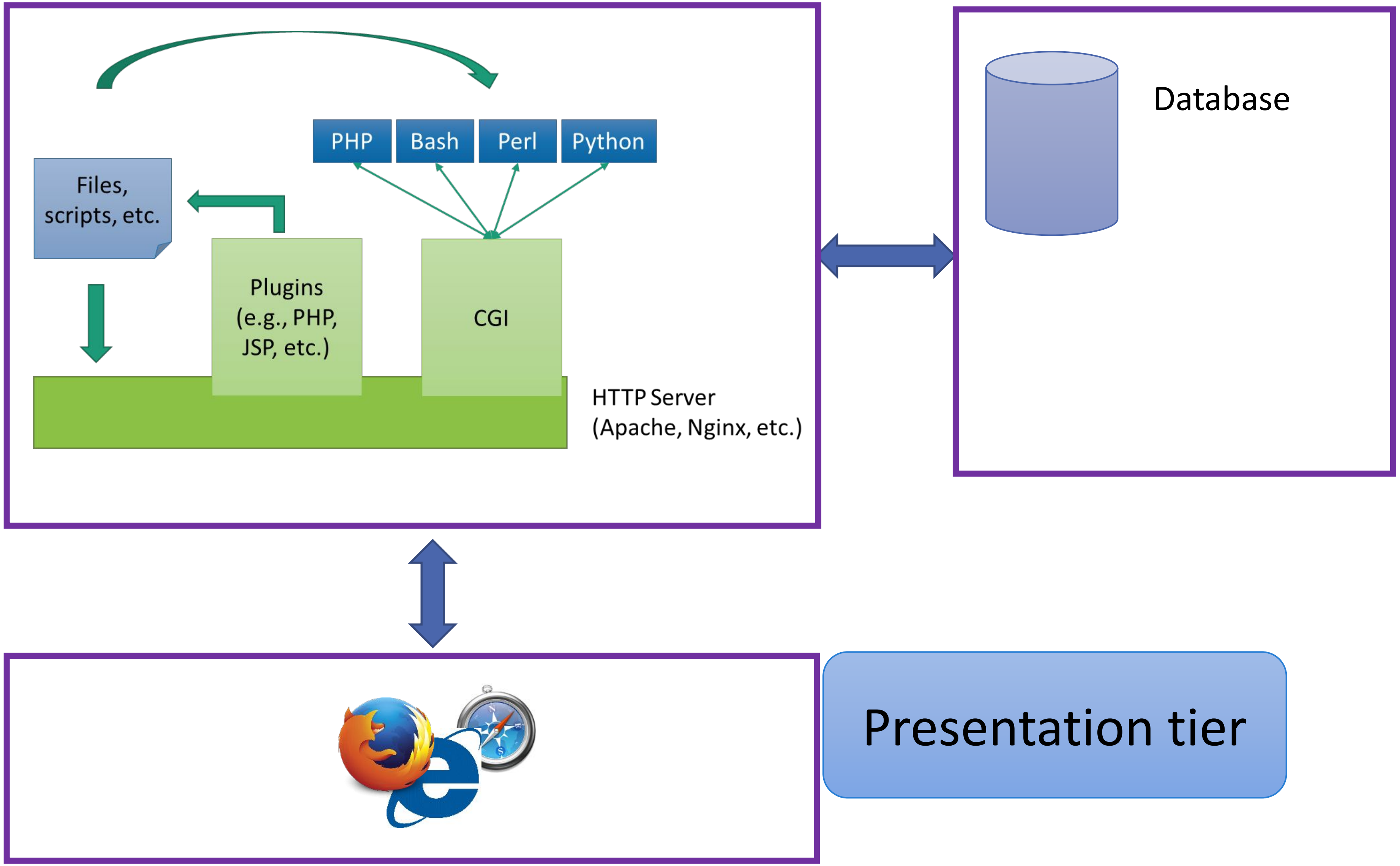
In practice, **a lot** more complicated

The Web is a Multitier Architecture

The Web as a 3-tier Architecture


PHP
Bash
Perl
Python
Files, scripts, etc.
Plugins (e.g., PHP, JSP, etc.)
CGI
HTTP Server (Apache, Nginx, etc.)
Database

The Web as a 3-tier Architecture



PHP

Bash

Perl

Python

Files, scripts, etc.

Plugins (e.g., PHP, JSP, etc.)

CGI

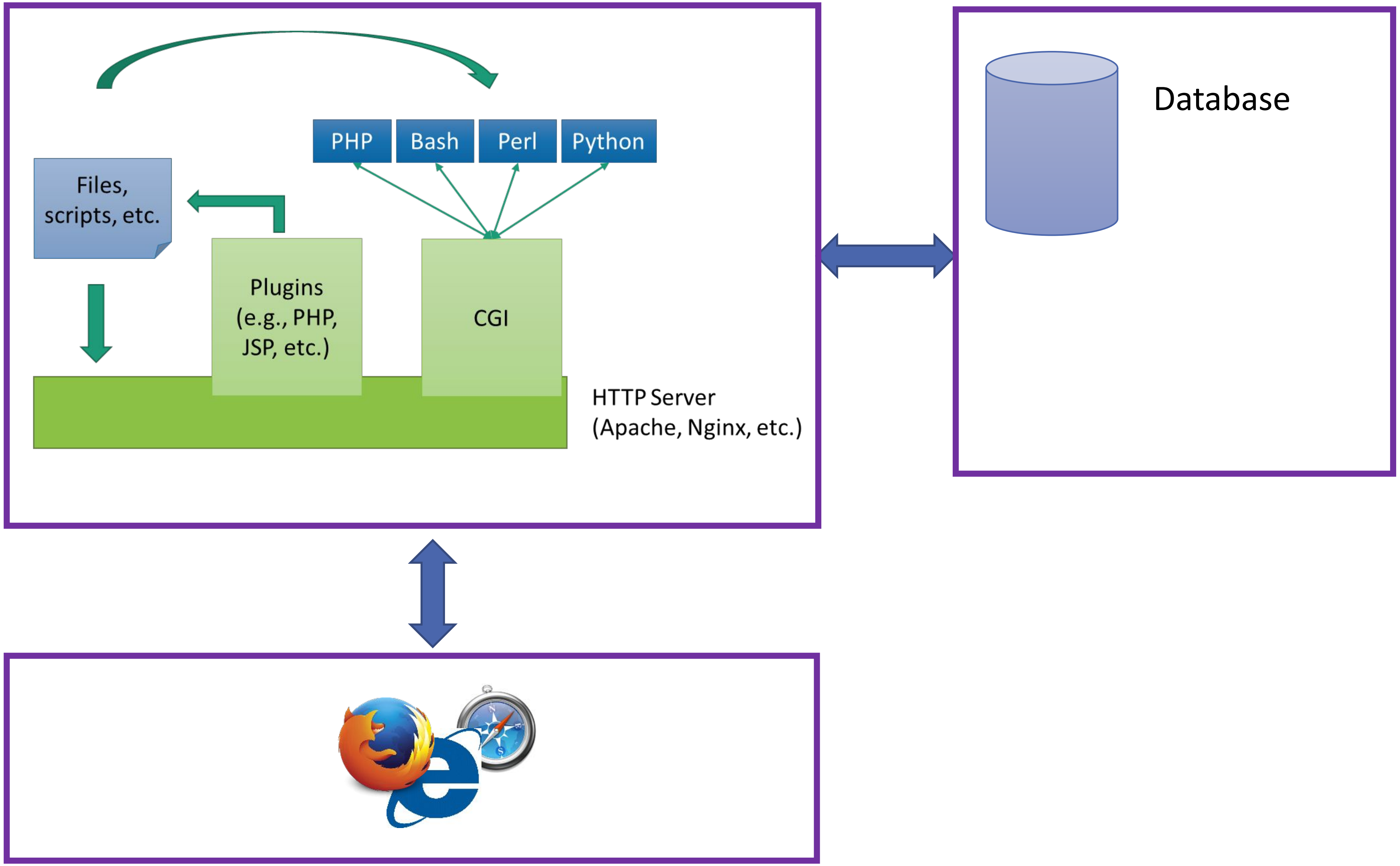
HTTP Server (Apache, Nginx, etc.)

Database

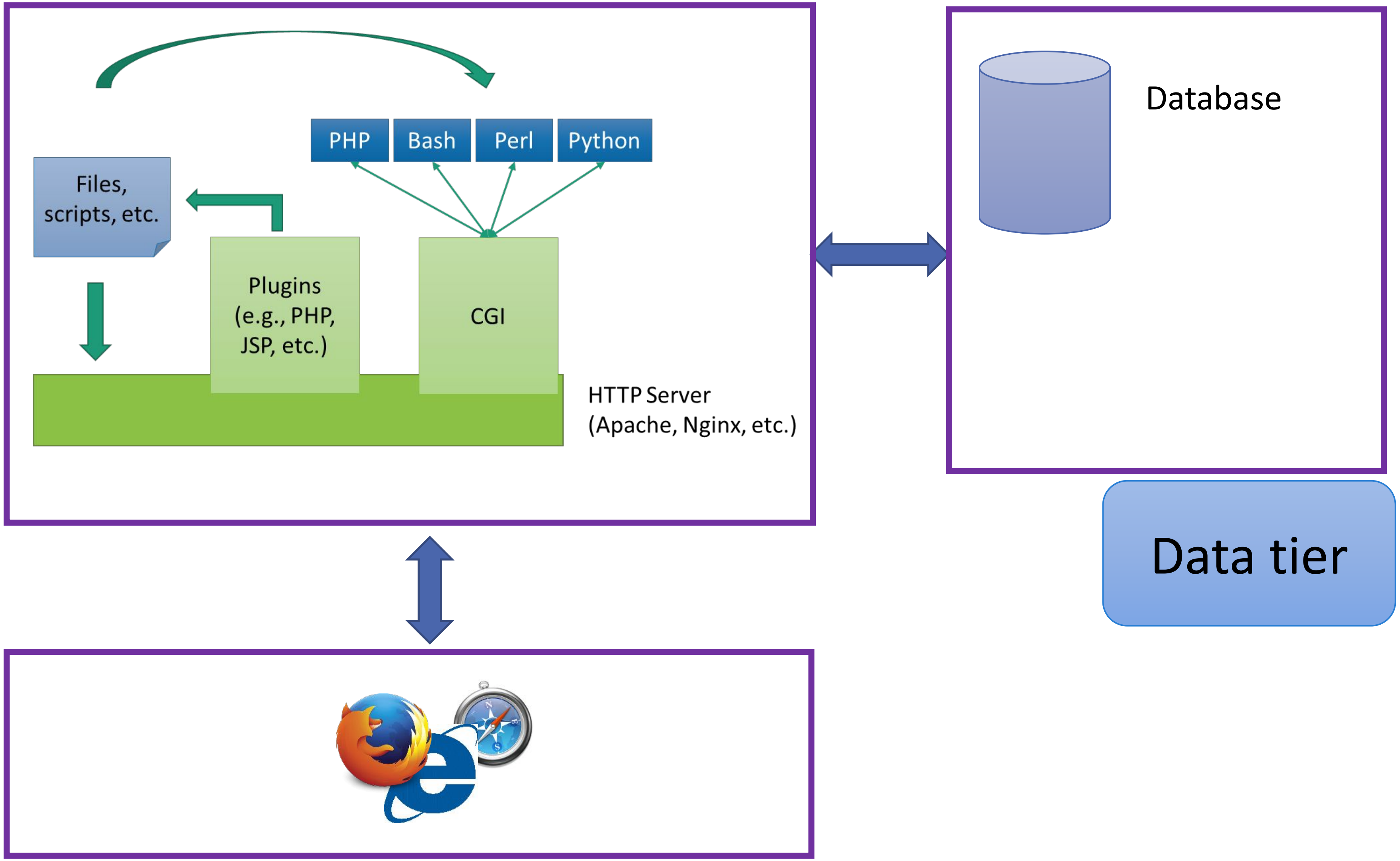
Presentation tier

The Web as a 3-tier Architecture

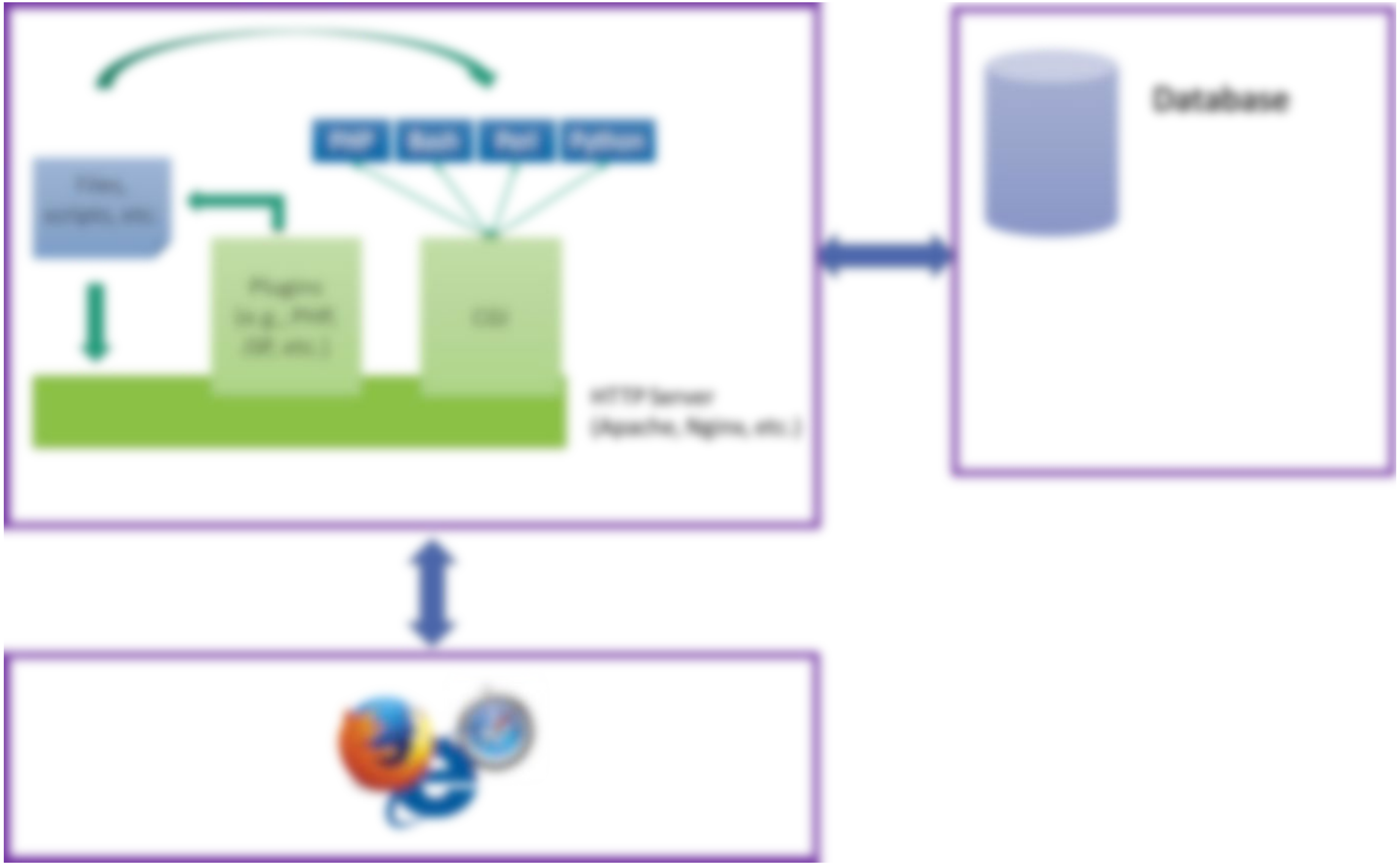
Logic tier


PHP
Bash
Perl
Python
Files, scripts, etc.
Plugins (e.g., PHP, JSP, etc.)
CGI
HTTP Server (Apache, Nginx, etc.)
Database

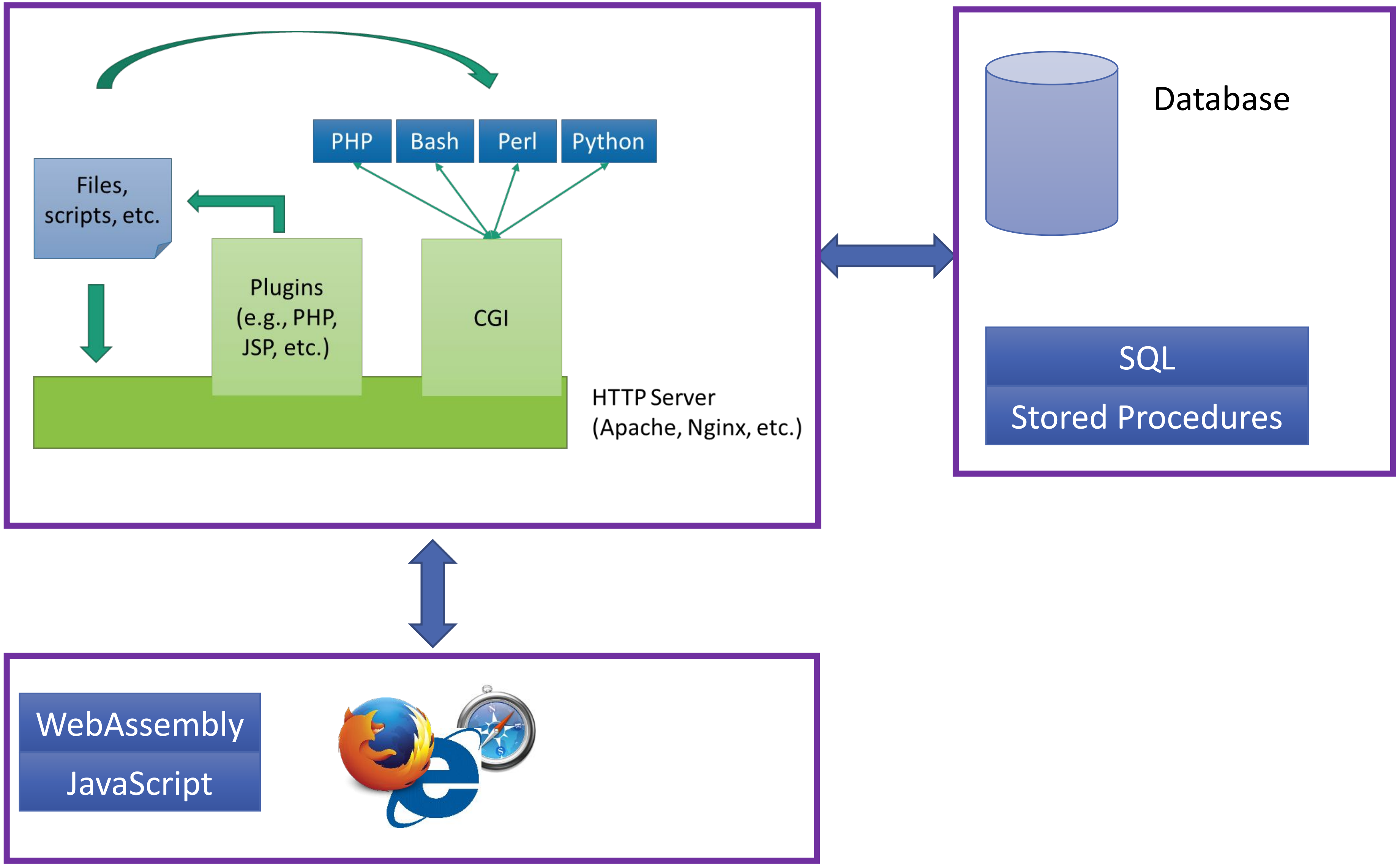
The Web as a 3-tier Architecture


PHP
Bash
Perl
Python
Files, scripts, etc.
Plugins (e.g., PHP, JSP, etc.)
CGI
HTTP Server (Apache, Nginx, etc.)
Database
Data tier

Blurry Application Boundary


PHP
Bash
Perl
Python
Files, scripts, etc.
Plugins (e.g., PHP, ISP, etc.)
CGI
HTTP Server (Apache, Nginx, etc.)
Database

Application Software Can Execute in Either Tier



PHP

Bash

Perl

Python

Files, scripts, etc.

Plugins (e.g., PHP, JSP, etc.)

CGI

HTTP Server (Apache, Nginx, etc.)

Database

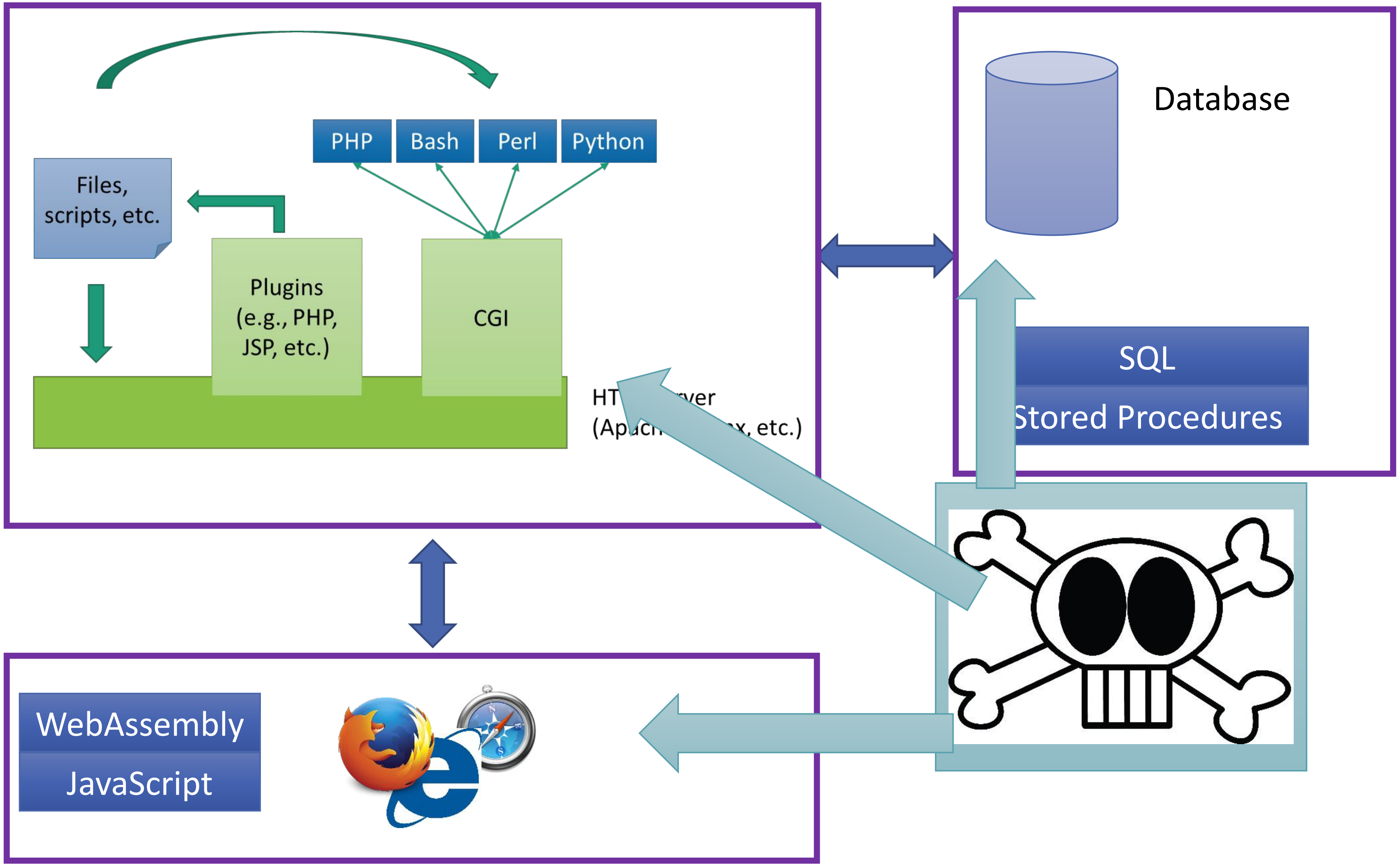
SQL

Stored Procedures

WebAssembly

JavaScript

All Tiers Can Be Vulnerable



Hyper Text Transfer Protocol (HTTP)

HTTP Basics

- **Stateless** protocol used to send and receive data
  - Text-based □Human readable
- Used by many applications
  - Simplicity
  - Most firewalls & intrusion prevention systems allow HTTP
- HTTP transactions follow the same general format
  - 3-part client request / server response

1. request or response line
2. header section
3. entity body

HTTP Request

- Request line
- `<METHOD>  /path/to/resource?query_string  HTTP/1.1`

GET Parameter

Request with a Header Section

- The header contains name value pairs

The Body of the Response

- The browser gets the response and starts consuming it
    - Drawing on the screen according to HTML code
    - Fetching additional resources
    - Executing code (JS, etc.)

- The content received can be classified as

- Static
    - Content that is stable and determined by the path of the URL

- Dynamic
    - Content that is changes based on user input and server state

Request with a Body Section

- In this example the body is used to send parameters

POST
Parameter

HTTP Response

- Response line
- `HTTP/1.1 <STATUS CODE> <STATUS MESSAGE>`

HTTP Response

- Here the body is used to send the requested data compressed

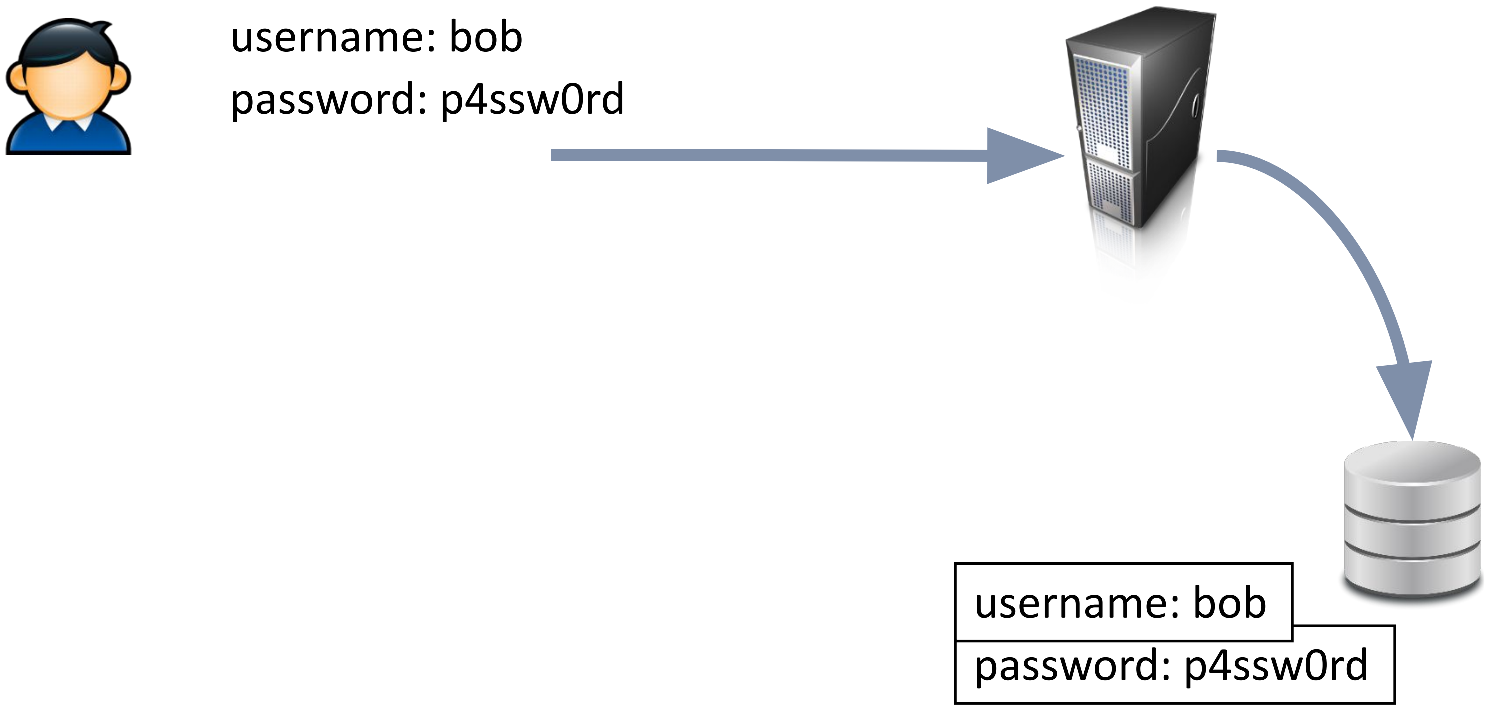
Authentication on the Web

- Passwords are the most commonly used means of authenticated

- Process
  - User provides name/login and password
  - System compares password with the one stored for that specified login

- The user ID:
  - Determines that the user is authorized to access the system
  - Determines the user’s privileges
  - Is used in discretionary access control

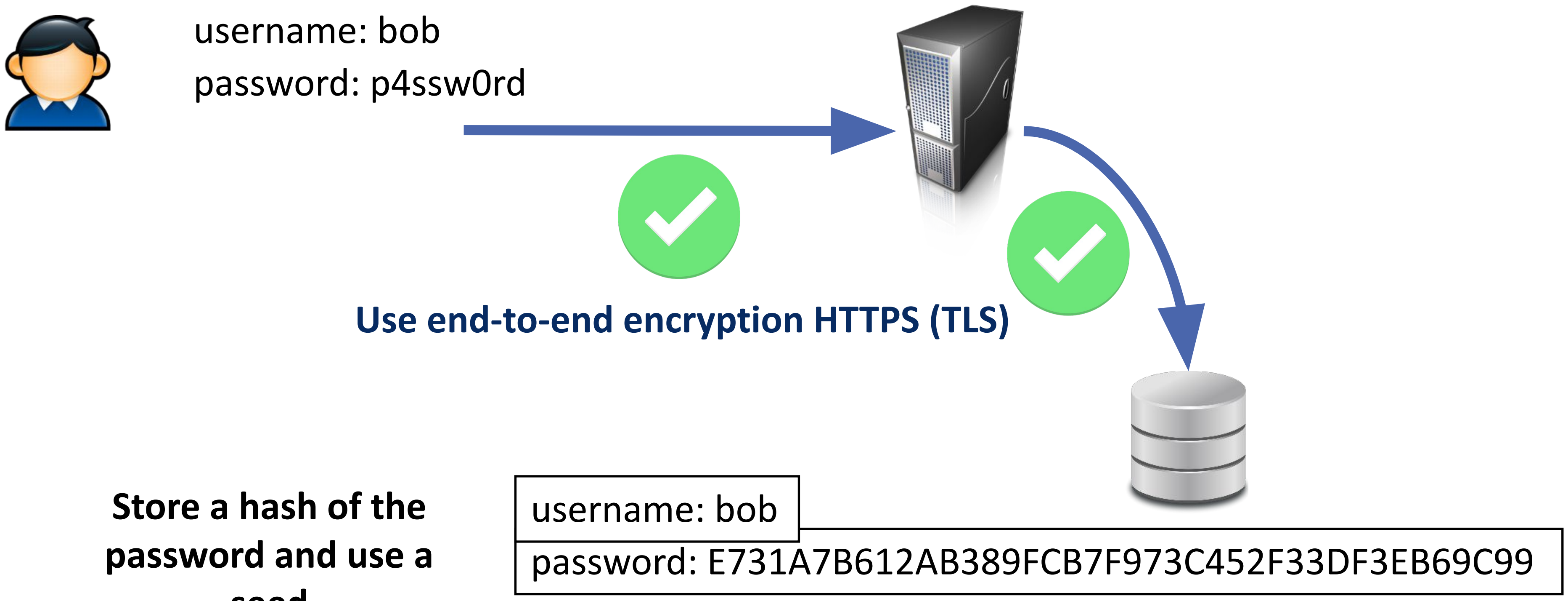
Authentication with Passwords



username: bob
password: p4ssw0rd

username: bob
password: p4ssw0rd

Good Practices



username: bob
password: p4ssw0rd

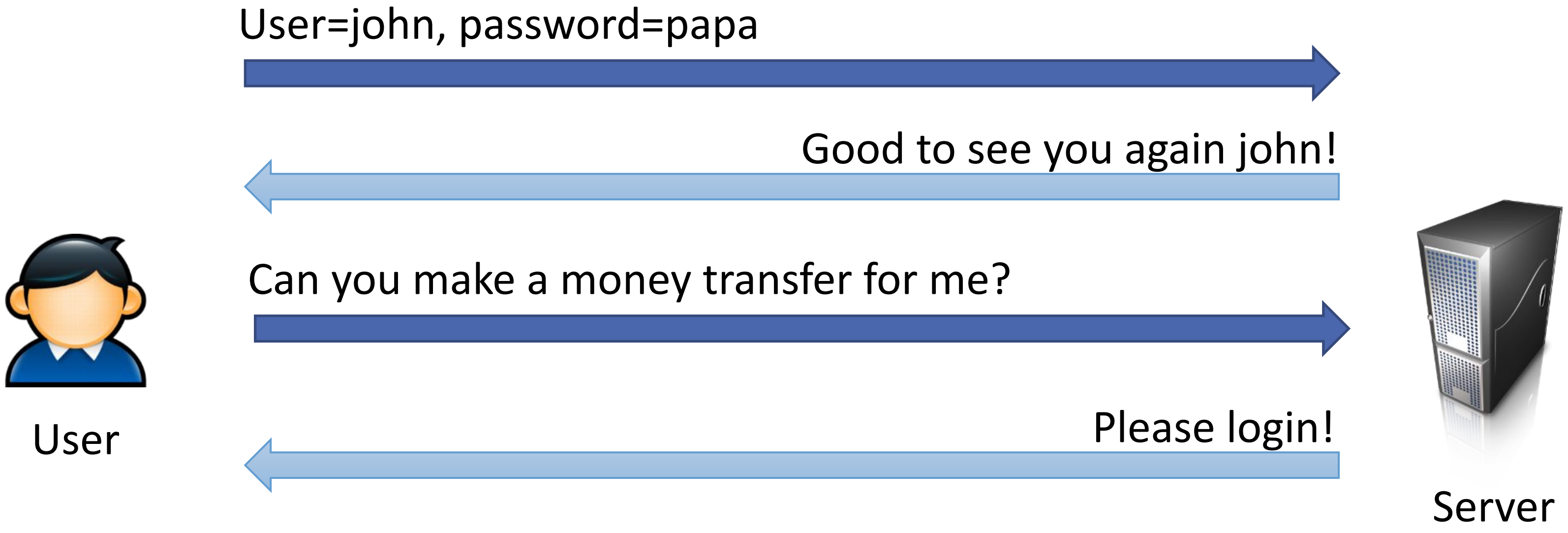
Use end-to-end encryption HTTPS (TLS)

Store a hash of the password and use a seed

username: bob
password: E731A7B612AB389FCB7F973C452F33DF3EB69C99

HTTP is a Stateless Protocol

HTTP is a Stateless Protocol



User=john, password=papa

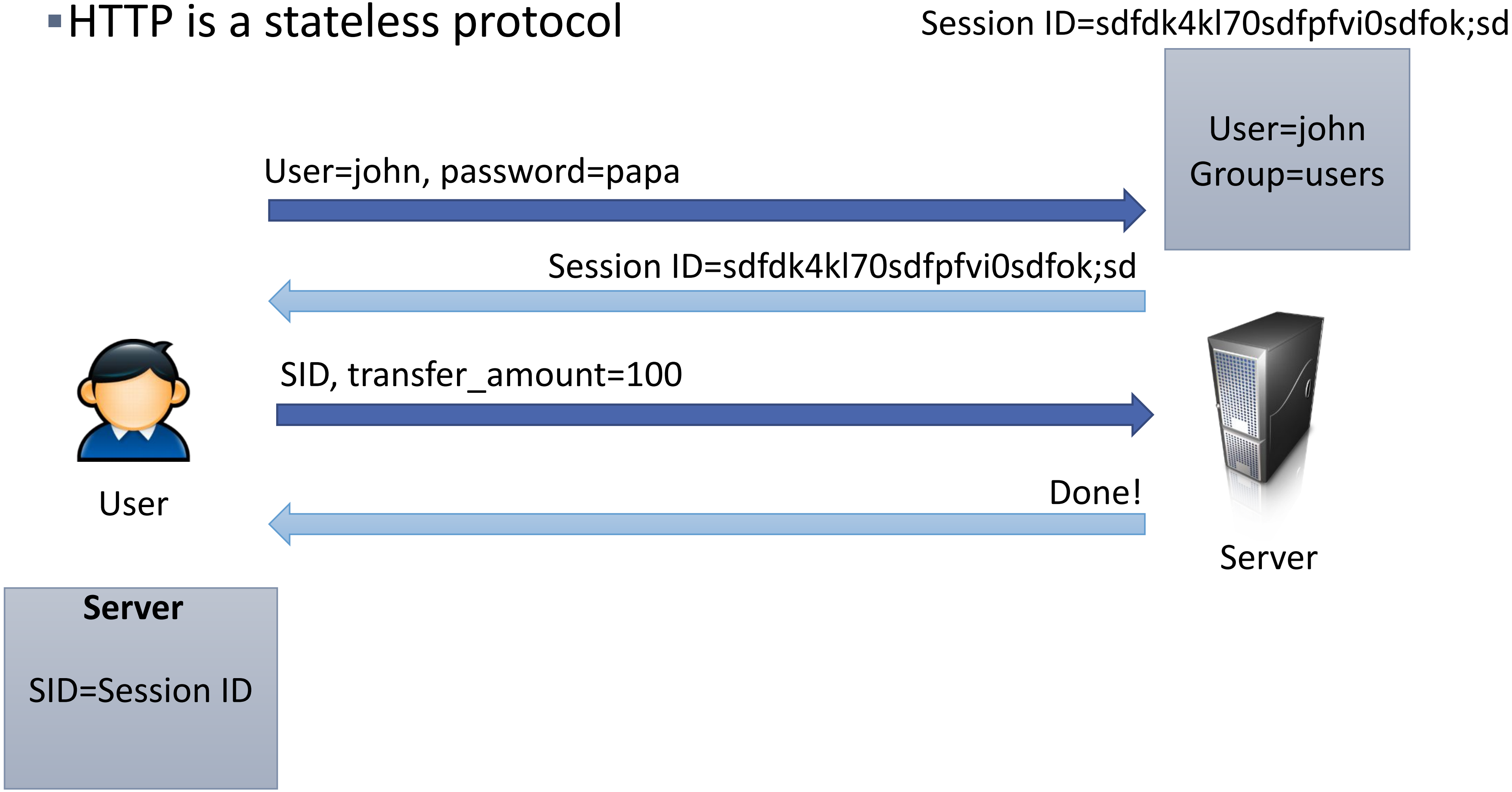
Good to see you again john!

Can you make a money transfer for me?

Please login!

HTTP Session Management

- HTTP is a stateless protocol

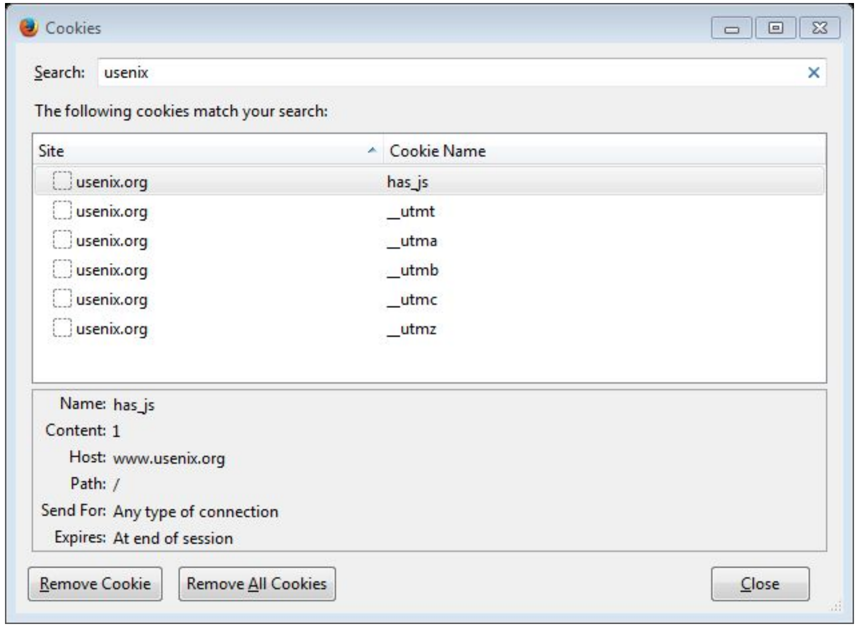


Implementing Session IDs

- Encoding it into the URL as GET parameter
  - Exposed! Visible
    - Even when using encrypted connections
      - Stored in logs, history, visible in browser location bar
- Hidden form field submitted in POST requests
  - Lost when browser tab is closed
- Cookies
  - Preferable
  - Survives when browser tab is closed
  - Can be rejected by clients

Cookies

- Token that is set by server, stored on client
- Key-value pairs (“name=value“)
- Access control based on server domain



Search: usenix

The following cookies match your search:

Site	Cookie Name
□ usenix.org	has_js
□ usenix.org	__utmt
□ usenix.org	__utma
□ usenix.org	__utmb
□ usenix.org	__utmc
□ usenix.org	__utmz

Name: has_js
Content: 1
Host: www.usenix.org
Path: /
Send For: Any type of connection
Expires: At end of session

What Are Cookies Used For?

- Authentication
  - The cookie proves to the website that the client previously authenticated correctly
- Personalization
  - Helps the website recognize the user from a previous visit
- Tracking
  - Follow the user from site to site; learn his/her browsing behavior, preferences, and so on

Cookie Variations

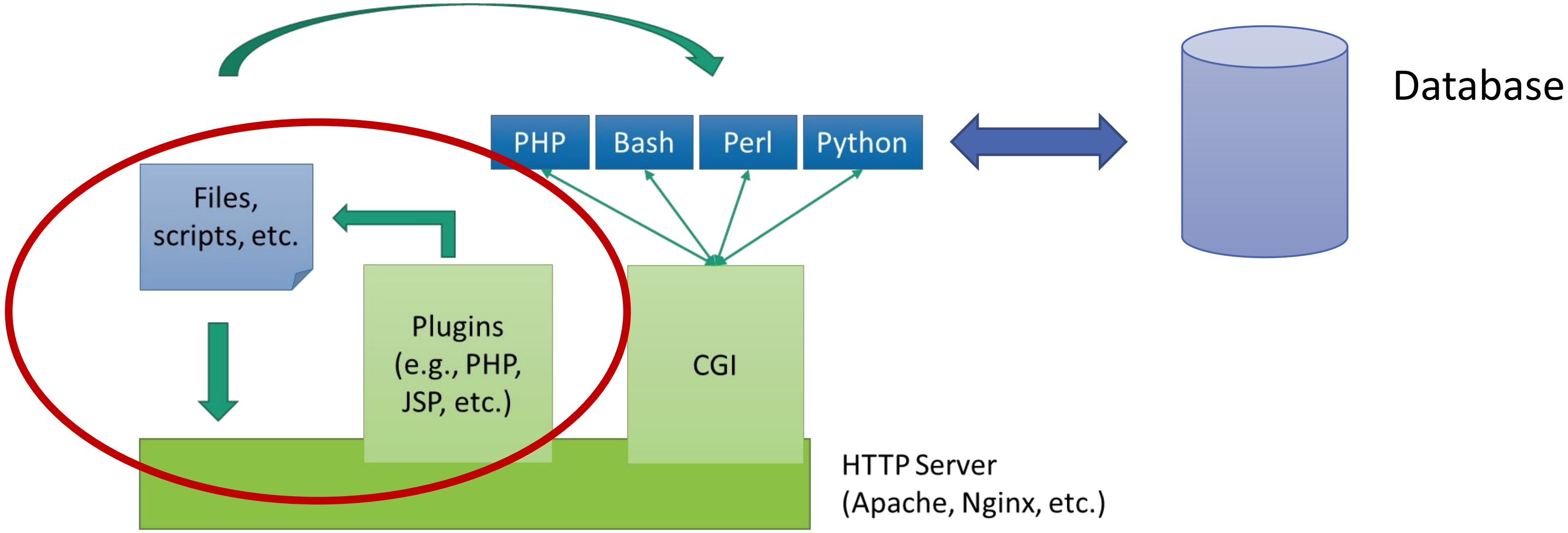
- Non-persistent cookies
  - Only stored in memory during browser session

- Secure cookies
  - Only transmitted over encrypted (SSL) connections
  - Only encrypting the cookie is vulnerable to replay attacks

- Cookies that include the IP address
  - Example: hash(IP) + nonce
  - Makes cookie stealing harder
  - Breaks session if IP address of client changes during that session

Passing Data to Web Applications

Passing Data to Web Applications

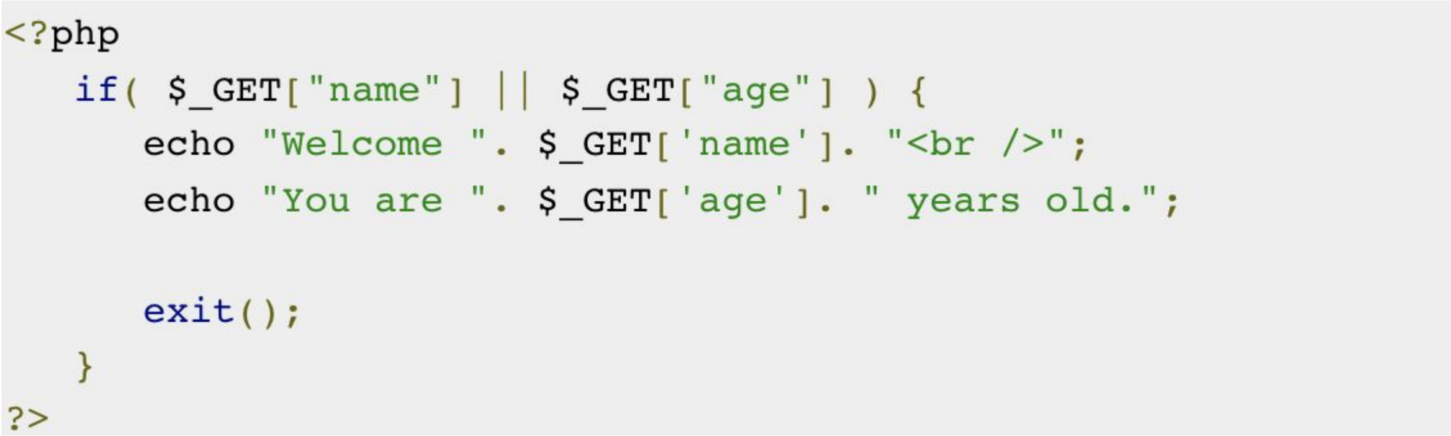

PHP
Bash
Perl
Python
Database
Files, scripts, etc.
Plugins (e.g., PHP, JSP, etc.)
CGI
HTTP Server (Apache, Nginx, etc.)

- JSP, PHP, Python (Web Server Gateway Interface), Ruby on Rails, etc.

PHP Example: Reading GET Variables

- Variables passed in GET requests are made available to apps using the special global array $_GET

```
GET /index.html&name=no_one&age=120&… HTTP/1.0
```

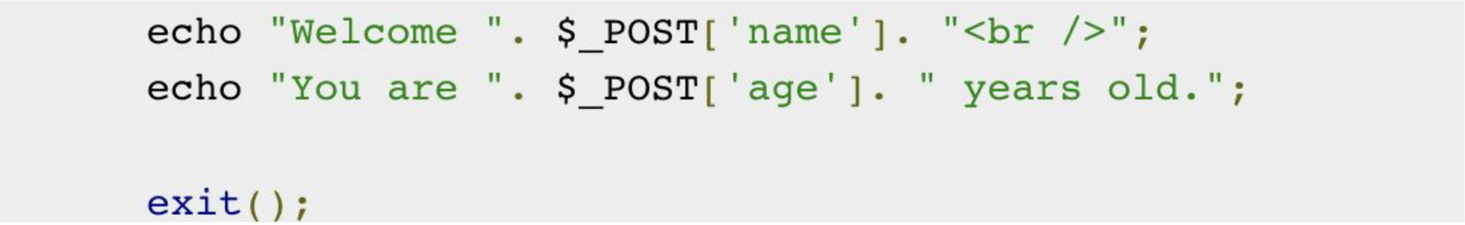


```
<?php
   if( $_GET["name"] || $_GET["age"] ) {
      echo "Welcome ". $_GET['name']. "<br />";
      echo "You are ". $_GET['age']. " years old.";

      exit();
   }
?>
```

PHP Example: Reading POST Variables

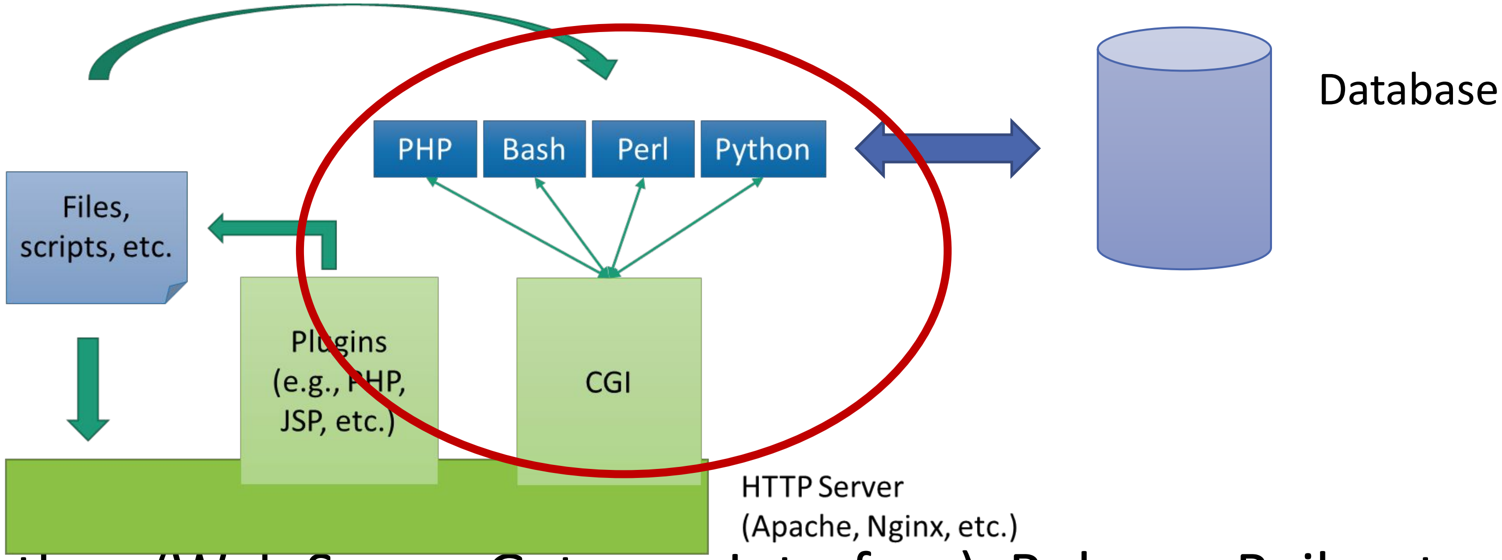
- Variables passed in POST requests are made available to apps using the special global array $_POST

```
POST /index.html HTTP/1.0

name=no_one&age=120&…
```



```
echo "Welcome ". $_POST['name']. "<br />";
echo "You are ". $_POST['age']. " years old.";

exit();
```

Passing Data to Web Applications


PHP
Bash
Perl
Python
Database
Files, scripts, etc.
Plugins (e.g., PHP, JSP, etc.)
CGI
HTTP Server (Apache, Nginx, etc.)

- JSP, PHP, Python (Web Server Gateway Interface), Ruby on Rails, etc.
- Common Gateway Interface (CGI)
  - Executes a **any** program to handle HTTP requests and generate dynamic content
    - Body of request is given as standard input
    - Header data and other CGI-specific data are passed as environment variables
    - Standard output produced by program is returned as the body of the response

CGI Example: Bash

```
GET /index.html&var1=val1&var2=val2&…
HTTP/1.0
X-HEADER: X-VALUE
```

CGI defines shell environment variables

```
REQUEST_METHOD=GET
QUERY_STRING=var1=val1&var2=val2&…
X-HEADER=X-VALUE
```

Application accesses them

```
#!/bin/bash
if [ "$REQUEST_METHOD" = "GET" ]; then
         # read value of "var1”
        Var1=$(echo "$QUERY_STRING" | sed -n 's/^.*var1=\([^&]*\).*$/\1/p’)
        # read value of "var1”
        Var2=$(echo "$QUERY_STRING" | sed -n 's/^.*var2=\([^&]*\).*$/\1/p')
```

CGI Example: Python

- Using a helper package to access user data

```
#!/usr/bin/env python2

import cgi
import cgitb
cgitb.enable()

input_data = cgi.FieldStorage()

print 'Content-Type:text/html' # HTML is following
print                          # Leave a blank line
print '<h1>Addition Results</h1>'
try:
  num1 = int(input_data["num1"].value)
  num2 = int(input_data["num2"].value)
except:
  print '<p>Sorry, we cannot turn your inputs into numbers (integers).</p>'
  return 1
print '<p>{0} + {1} = {2}</p>'.format(num1, num2, num1 + num2)
```

```
<!DOCTYPE html>
<html>
 <body>
  <form action="add.cgi" method="POST">
   Enter two numbers to add:<br />
   First Number: <input type="text" name="num1" /><br />
   Second Number: <input type="text" name="num2" /><br />
   <input type="submit" value="Add" />
  </form>
 </body>
</html>
```

Appendix

Other HTTP methods

- HEAD
  - Works like GET but the server does not send the body of a response (it only sends the appropriate headers)
- TRACE
  - Designed for diagnostic purposes. Returns in its response body the exact request it received.
- OPTIONS
  - Returns the available methods for a specific resource.
- PUT
  - Allows the upload of a file in certain location. This should be disabled by default.

Popular Request Headers

- All request headers are meant to communicate some information to the server
- User-Agent

  Family and version of browser, as well as the underlying environment
- Accept
  - Kind of content the client is willing to accept
- Accept-encoding
  - What type of encoding the client supports (e.g. gzip)
- Host
  - The target website of this request
- Cookie
  - Send the server all cookies the browser has for this specific website
- Referer
  - Specifies the URL from which the current request originated
  - Note the misspelling. This is intentional.

Popular Response Headers

- All response headers are meant to communicate some information to the client (browser)
- Cache-control:
  - Passing caching directives to the client (e.g. no-cache)
- Expires:
  - How long the content is valid (and may be cached for)
- Server
  - Provides information about the identity of the server
- Set-Cookie
  - Sets cookies for this website