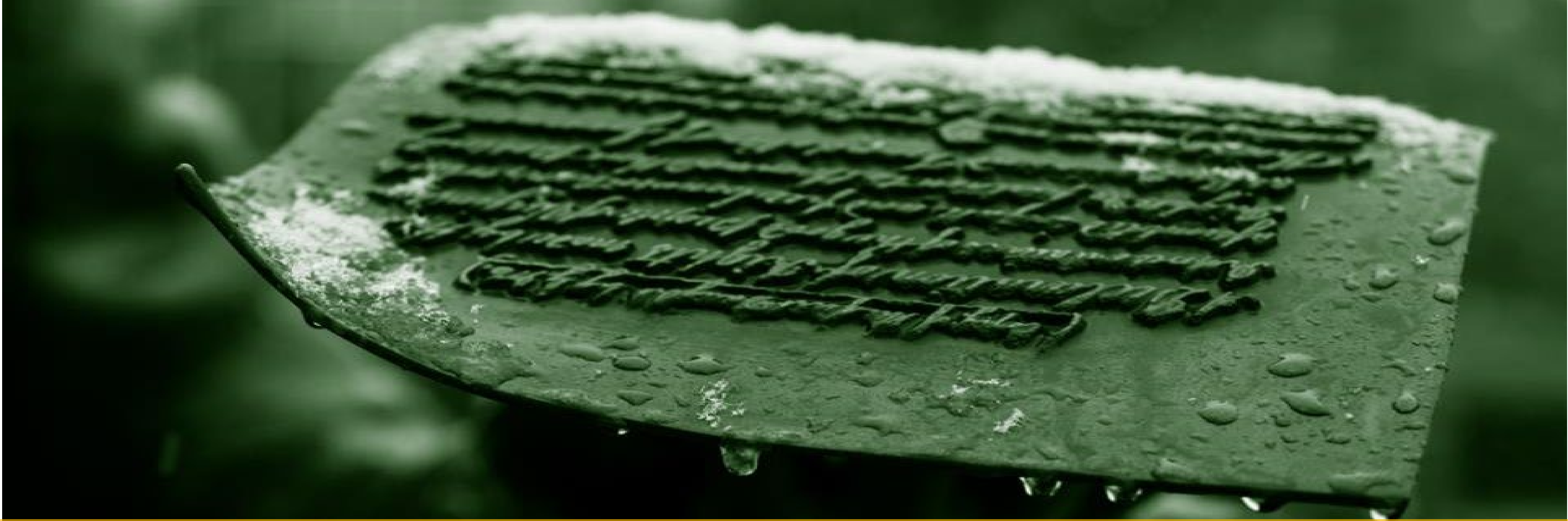


MODULE 4 - HTTP SERVER FUNDAMENTALS

IT 207 – IT Programming

GEORGE MASON UNIVERSITY

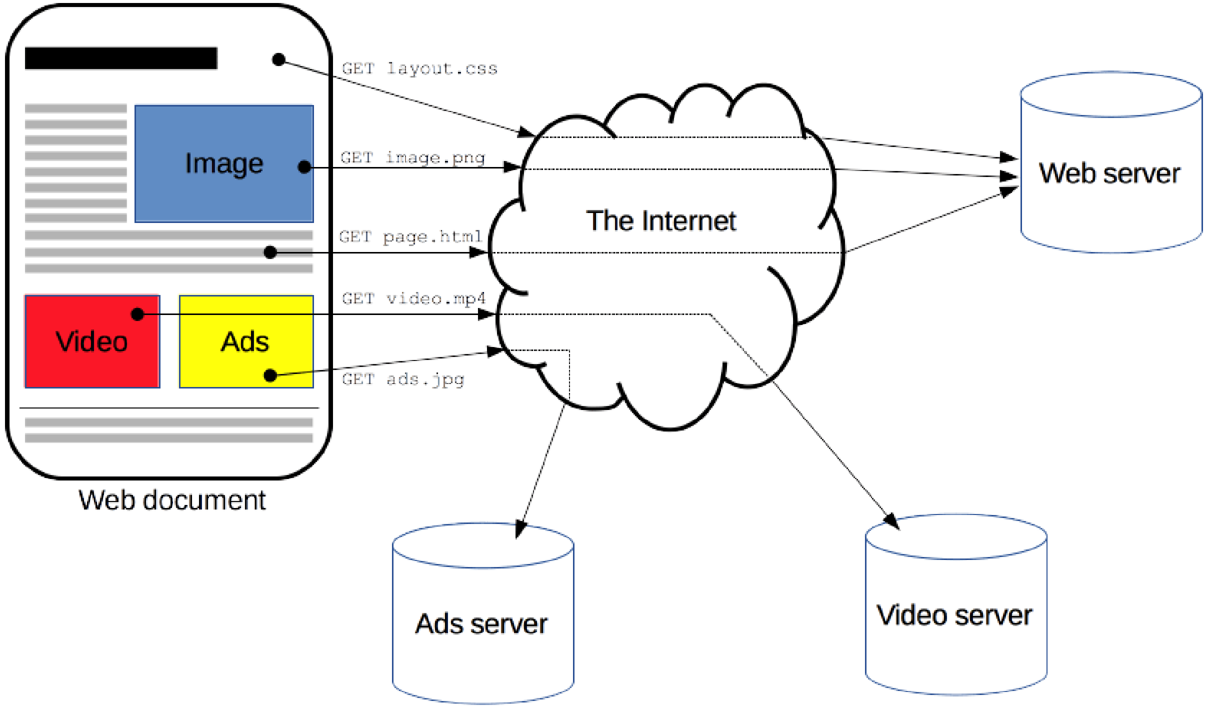
Lecture Outline

- HTTP protocol methods, headers, and status codes
- CRUD Operations
- REST Architecture
- Building a RESTful web service in Nodejs

HTTP
PROTOCOL

HTTP OVERVIEW

- HTTP is a protocol for fetching resources on the web
  - Resource includes HTML documents, images, videos, scripts, etc..
- HTTP is a Server-Client protocol
  - Client (e.g.Web browser) send requests for resources
  - Server construct a response from different resources and sends it to the client



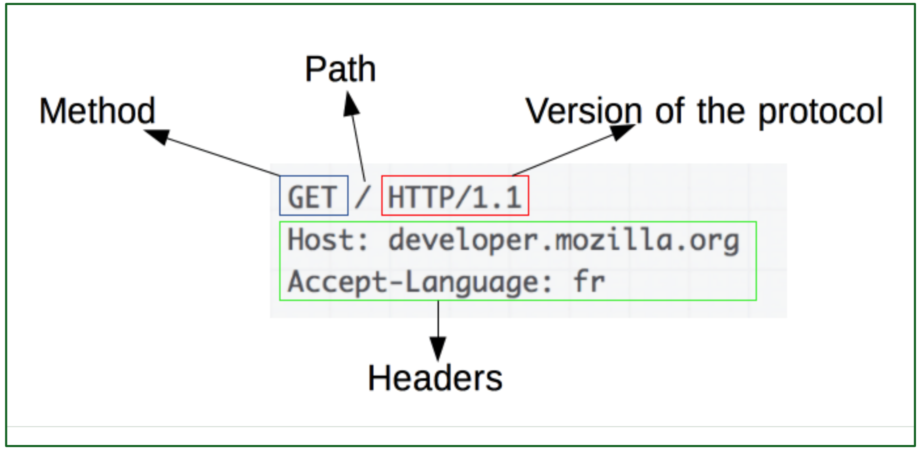
https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#requests

HTTP OVERVIEW

- **HTTP is simple**
  - Designed to be human readable
- **HTTP is stateless**
  - State is not stored
- **HTTP is extensible**
  - new functionalities can be introduced.

HTTP Requests

- Requests are messages sent by HTTP clients
- **Requests are composed of:**
  - Request method: a verb indicating the desired action to be performed for a given resource.
  - The path of the resource to fetch
  - The version of the HTTP protocol.
  - Optional headers that convey additional information for the servers.
  - A body required for some methods



```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

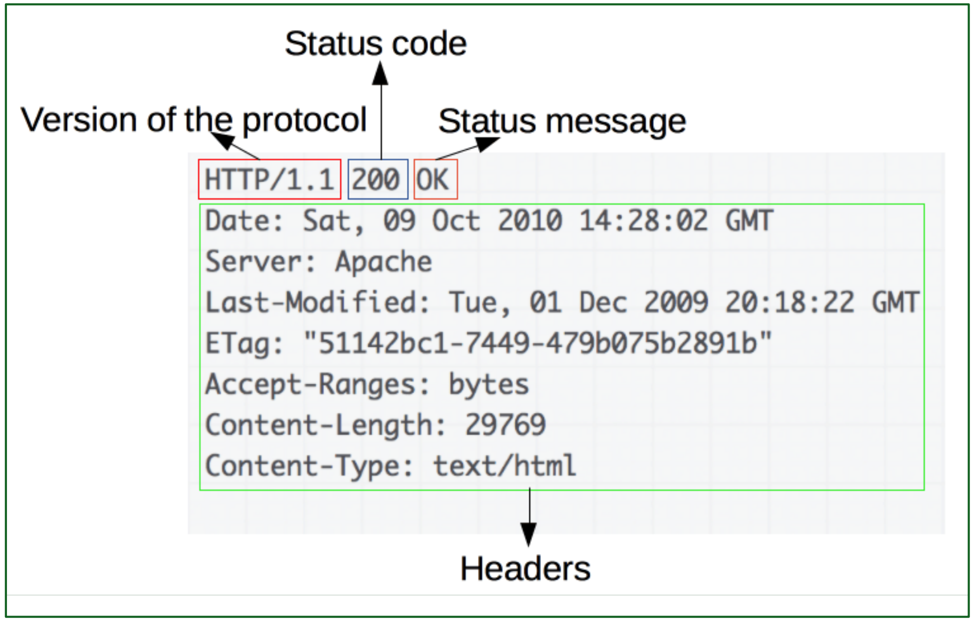
HTTP Methods

- HTTP defines a number of methods.

- **The most commonly used are:**
  - **GET:** only retrieves a representation of the specified resource.
  - **POST:** submits an entity to the specified resource, often causing a change in state or side effects on the server.
  - **DELETE:** deletes the specified resource.
  - **PUT:** replaces all current representations of the target resource with the request payload.

HTTP Responses

- Responses are messages sent by HTTP servers
- **Responses are composed of:**
  - The version of the HTTP protocol used.
  - A status code, indicating if the request was successful or not, and why.
  - A status message describing the status code..
  - Optional headers that convey additional information for the servers like those for requests.
  - Optionally, a body containing the fetched resource



```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html
```

HTTP Status Codes

- HTTP response status codes indicate whether a specific HTTP request has been successfully completed.

- **Responses are grouped in five classes:**
  1. Informational responses (100 – 199)
  2. Successful responses (200 – 299)
  3. Redirection messages (300 – 399)
  4. Client error responses (400 – 499)
  5. Server error responses (500 – 599)

The status codes listed are defined by RFC 9110

CRUD OPERATIONS

CRUD Overview

- CRUD is an acronym of the four basic operations a software application should be able to perform
  - Create
  - Read
  - Update
  - Delete
- **CRUD apps consist of 3 parts:**
  - An API (or server): contains the code and logic of the app
  - A database: stores the app information and data
  - A user interface (UI): helps users interact with the app

CRUD AND HTTP

- Each letter in the CRUD acronym has a corresponding HTTP request method

CRUD OPERATION	HTTP REQUEST METHOD
Create	POST
Read	GET
Update	PUT
Delete	DELETE

REST
ARCHITECTURE

REST Overview

- REST is an acronym for REpresentational State Transfer
- REST is an architectural style for software design of web applications
- REST provides standards for computer systems on the web, making it easier for systems to communicate with each other

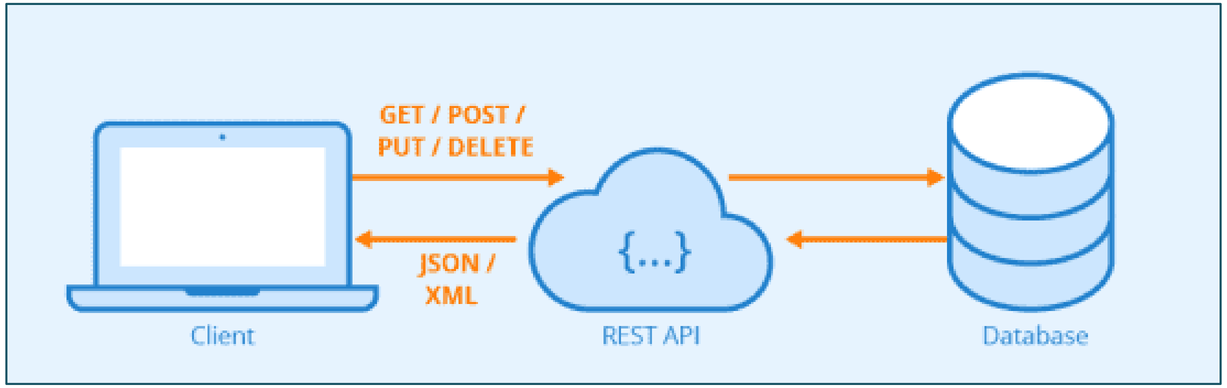
- **An architectural style is**
  - a collection of principles that shape or govern the design of applications.
  - a coarse-grained pattern that provides an abstract framework for a family of systems.
  - improves partitioning and promotes design reuse by providing solutions to frequently recurring problems

RESTful Systems

- **A RESTful System is a REST compliant system**
  - The design of REST systems is characterized by being stateless and by applying the principle of separation of concerns.
- **In Web based systems**
  - The client and the server can be implemented independently.
  - The code on the client side can be changed at any time without affecting the operation of the server, and vice versa.
  - User interface is separated from the app logic and data storage.
  - Communication is done using well-defined messaging formats

REST API

- An API (application program interface) is a set of rules that enables different programs to communicate with one another.
- A REST API is a type of API that follows the principles of Representational State Transfer (REST) architecture.
  - It provides a standard way for web applications to communicate with each other over the internet.



HTTP + CRUD + REST API = RESTFUL WEB SERVICE

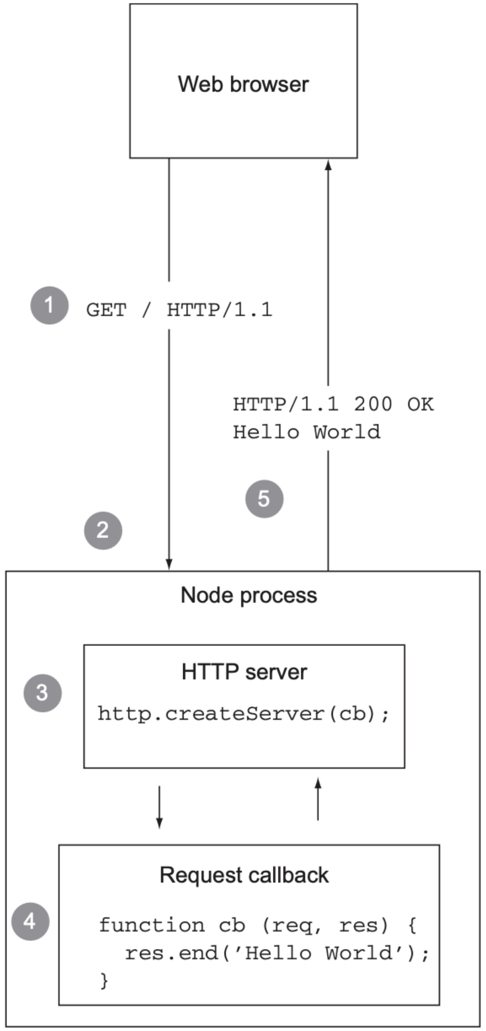
- Web services based on REST Architecture are known as RESTful web services.
  - A web service is a collection of open protocols and standards used for exchanging data between applications or systems.
- RESTful web services uses HTTP methods to implement the concept of REST architecture.
- RESTFul web services are resources and can be identified by their URIs.
- Example

URL	HTTP Method	body	Result
/UserService/users	GET	empty	Show list of all the users.
/UserService/addUser	POST	JSON String	Add details of new user.
/UserService/getUser/:id	GET	empty	Show details of a user.

BUILDING A RESTFUL WEB SERVICE IN NODEJS

HTTP Module in NodeJS

- Node provides HTTP server and client interfaces through the http module
  - Node servers are long-running processes that serve many requests throughout their lifetimes.
  - For every HTTP request received by the server, the request callback function will be invoked with new req and res objects.
  - Node will not automatically write any response back to the client.
    - It is the programmer’s responsibility to end the response



```
http.createServer(cb);
```

```
function cb (req, res) {
  res.end('Hello World');
}
```

1 An HTTP client, like a web browser, initiates an HTTP request.

2 Node accepts the connection, and incoming request data is given to the HTTP server.

3 The HTTP server parses up to the end of the HTTP headers and then hands control over to the request callback.

4 The request callback performs application logic, in this case responding immediately with the text “Hello World.”

5 The request is sent back through the HTTP server, which formats a proper HTTP response for the client.

Building a to-do List Web Service

- Create a to-do list web service that implements the CRUD operations as follows
  - POST: Add items to the to-do list
  - GET: Display a listing of the current items, or display the details of a specific item
  - DELETE: Remove items from the to-do list
  - PUT: Modify existing items
- The web service will be accessed through the cURL command

Refer to Nodejs in Action – 2014 section 4.2