Web Application Development

Restful API (Backend Development 1)

Instructor: Thanh Binh Nguyen

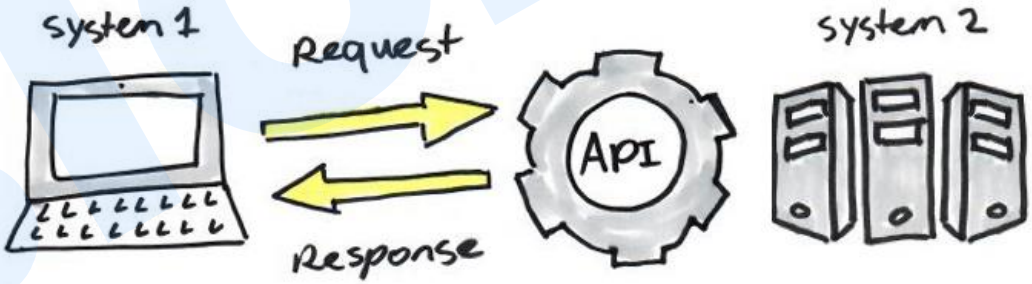
February 1st, 2020

S3Lab
Smart Software System Laboratory

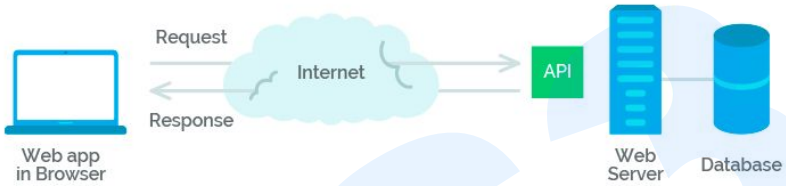

“A successful website does three things:
It attracts the right kinds of visitors.
Guides them to the main services or product you offer.
Collect Contact details for future ongoing relation.”
– Mohamed Saad

Application Programming Interface

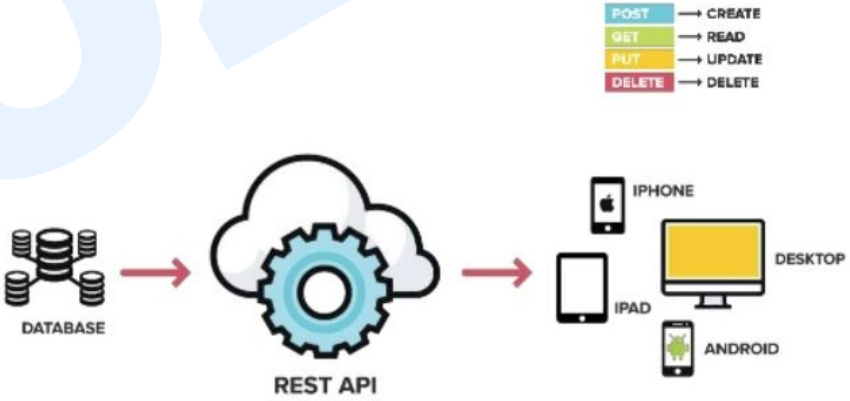
- A set of **rules** and **mechanisms** by which one application or component **interacts** with the others.
- An API allows specifically **exposed methods** of an application to be **accessed** and **manipulated outside** of the program itself.
- The API can return data that need for your application in a convenient format
  - JSON
  - XML



Restful API

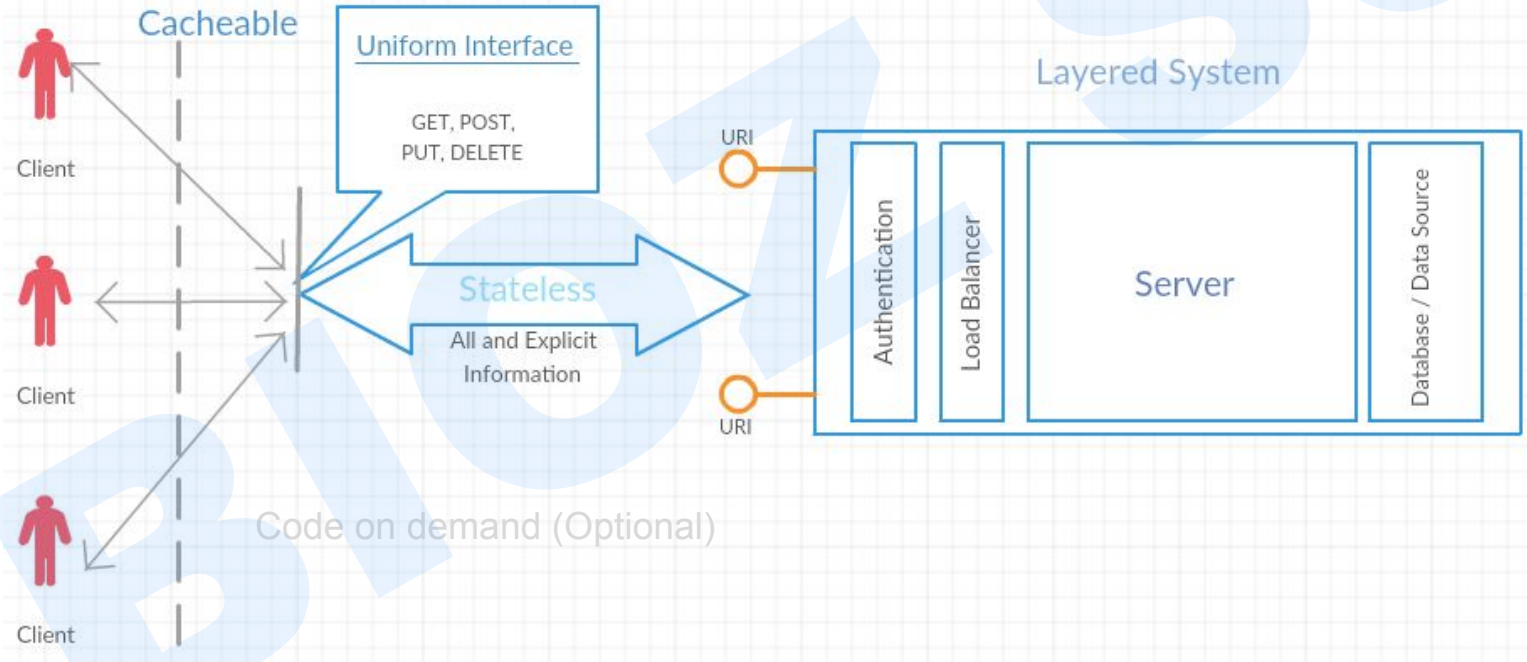


- **Web APIs**: use web protocols such as HTTP, HTTPs, JSON, XML, etc. For example, a web API can be used to obtain data from a resource (such as U.S. postal service zip codes) without having to actually visit the application itself (checking usps.com).
- **REST**: is a short for **R**epresentational **S**tate **T**ransfer.
  - An architectural style for distributed hypermedia systems.
  - A set of rules and conventions for the creation of an API.
  - Was first presented by Roy Fielding in 2000 in his famous dissertation.



Restful API

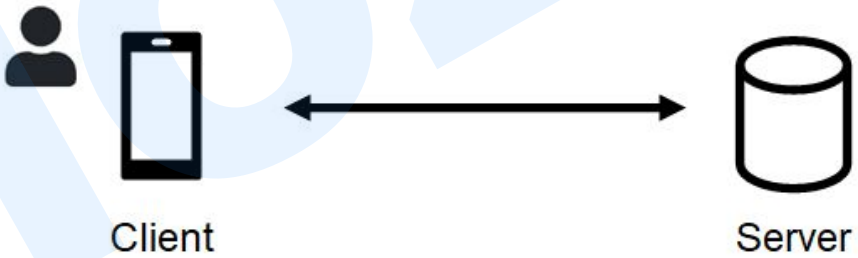
6 Constraints



Restful API

6 Constraints - 1. Client-Server architecture

- Separate the systems responsible for storing and processing the data (**the server**) from the systems responsible for collecting, requesting, consuming, and presenting the data to a user (**the client**).
- This separation should be so distinct that the client and server systems can be **improved** and **updated independently** each other.

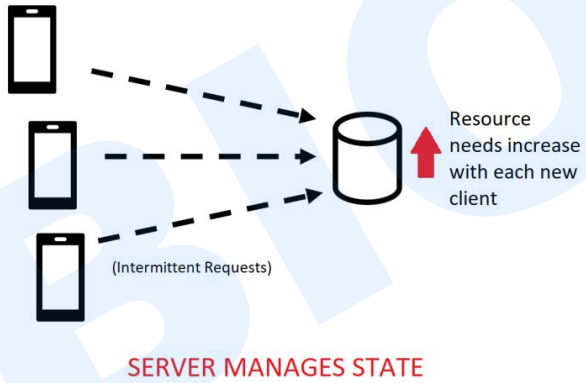


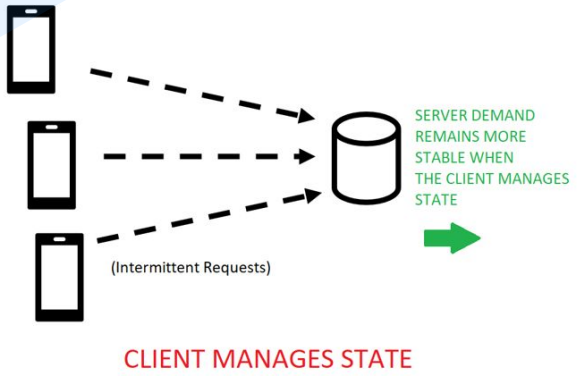
Restful API

6 Constraints - 2. Statelessness

- As far as the server is concerned, all client requests are treated equally. There's no special, server-side memory of past client activity. The responsibility of managing state (for example, logged in or not) is on the client. This constraint is what makes the **REST**ful approach so scalable.

the state gets transferred with each request





Restful API

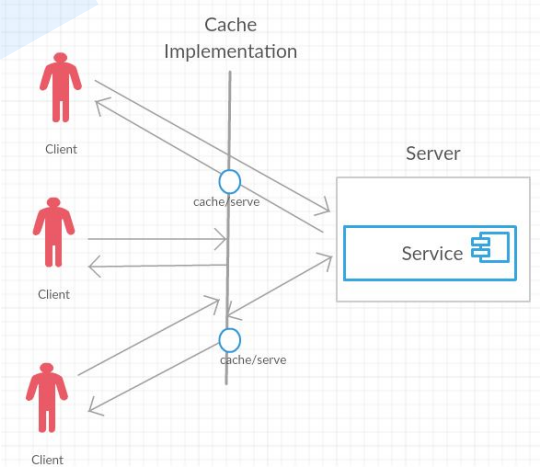
6 Constraints - 3. Cacheability

- **Clients** and **servers** should be able to cache resource data that changes infrequently.
- **Example**: there are 52 states and other jurisdictions in the U.S.A. That's not likely to change soon. So, it is inefficient to build a system that queries a database of states each and every time you need that data. Clients should be able to cache that infrequently updated date and web servers should be able to control the duration of that cache.

Browser caches

Proxy caches

Gateway caches
(reverse-proxy)



Restful API

6 Constraints - 3. Cacheability

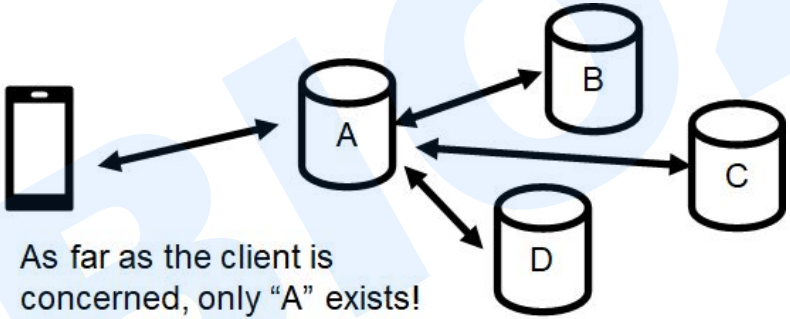
Several ways that we can control the cache behavior

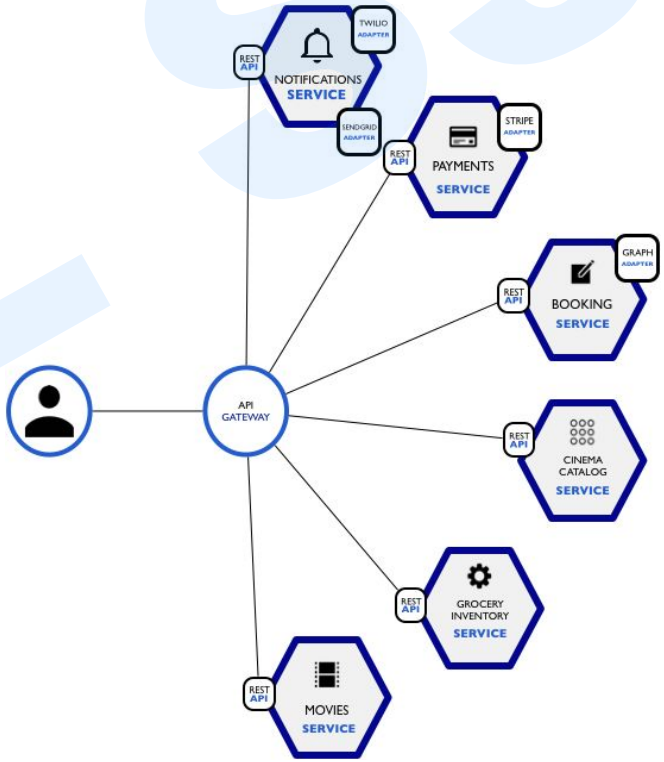
Headers	Description	Samples
Expires	Header attribute to represent date/time after which the response is considered stale	Expires: Fri, 12 Jan 2018 18:00:09 GMT
Cache-control	A header that defines various directives (for both requests and responses) that are followed by caching mechanisms	`Max age=4500` , cache-extension
E-Tag	Unique identifier for server resource states	ETag: `uqv2309u324klm`
Last-modified	Response header helps to identify the time the response was generated	Last-modified: Fri, 12 Jan 2018 18:00:09 GMT

Restful API

6 Constraints - 4. Layered System

- A client cannot tell whether it is connected directly to an end server, or to an intermediary along the way.
- Intermediary servers can also improve system scalability

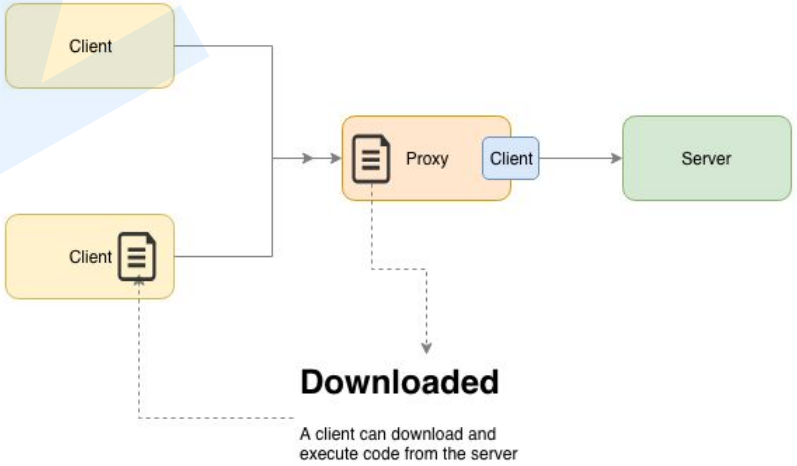




Restful API

6 Constraints - 5. Code on demand (Optional)

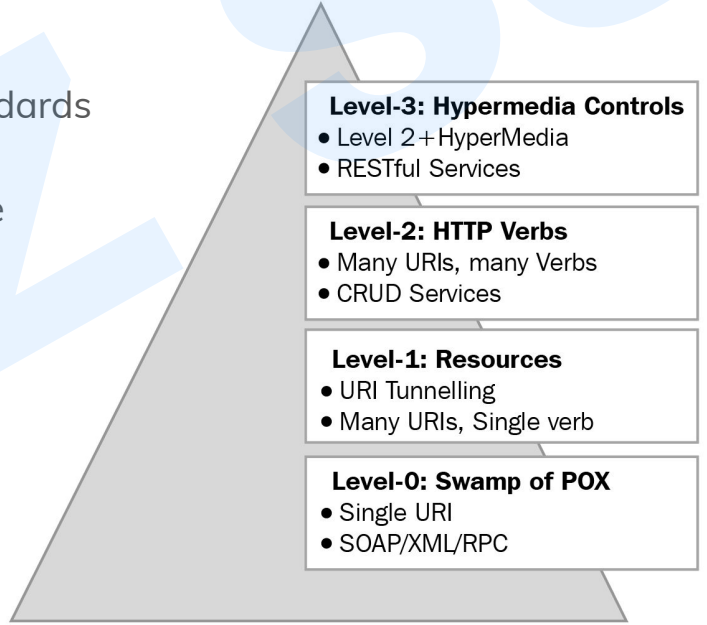
- Servers can temporarily extend or customize the functionality of a client by transferring executable code (scripts / applets).
- Allows server to decide how some things will be done.
- This is constraint is optional.



Restful API

6 Constraints - 6. Uniform Interface

- Identification of resources (URL or IRI)
- Manipulation of resources through HTTP standards
- Self descriptive messages
- Hypermedia as the engine of application state (A.K.A. HATEOAS)



Restful API

Conventions - Resource Naming

- Using nouns: `/users/{id}` instead of `/getUser`
- Pluralized resources: `/users` (typical resource) or `/users/{id}/address` (singleton resource)
- Forward slashes for hierarchy: `/users/{id}/address` clearly falls under the `/users/{id}` resource which falls under the `/users` collection
- Punctuation for lists: `/users/{id1},{id2}` to access multiple user resources
- Lowercase letters and dashes (or snake_case, camelCase): `/users/{id}/pending-orders` instead of `/users/{id}/Pending_Orders`
- Query parameters where necessary: `/users?location=USA` to find all users living in the United States
- Standard American English:
  - `/airplanes` instead of `/aeroplanes`
  - `/users/{id}/card-number` instead of `/users/{id}/pan`
  - `/users/{id}/phone-number` instead of `/users/{id}/tel-no`
  - `/users/{id}/pending-orders` instead of `/users/{id}/pending-orders.xml`

Restful API

Conventions - Filter, Search, paginate and Sort

- Paginate: `GET https://.../users?page=2&per-page=20`
- Custom Search by keyword: `GET https://.../users?q=text`
- Sort: `GET https://.../users?sort=-username,+id`
- Filter: `GET https://.../users?filter=[{key='type',operator='=',value='admin'}]`

Restful API

Conventions - Let HTTP method define action

Resource	GET (Read)	POST (Create)	PUT (Update)	DELETE (Delete)
/users	Returns a list of users	Creates a new user	Bulk update of users	Delete all users
/users/123	Returns a specific User	Method not allowed (405)	Updates a specific user	Deletes a specific user

Restful API

Conventions - Errors

- HTTP status codes: `2xx for successful, 3xx and 4xx for client error, 5xx for server error.`
  - `200 - OK -> GET, PUT, PATCH or DELETE`
  - `201 - Created -> POST`
  - `204 - No Content -> no content response`
  - `304 - Not Modified`
  - `400 - Bad Request`
  - `401 - Unauthorized`
  - `403 - Forbidden`
  - `404 - Not found`
  - `405 - Method Not Allowed`
  - `410 - Gone`
  - `415 - Unsupported Media Type`
  - `422 - Unprocessable Entity`
  - `429 - Too many requests`

Restful API

Conventions - Errors

- Custom Error Messages:

```
{
  "code" : 1234,
  "message" : "Something bad happened :(",
  "description" : "More details about the error here"
}
```

```
{
  "code" : 1024,
  "message" : "Validation Failed",
  "errors" : [
    {
      "code" : 5432,
      "field" : "first_name",
      "message" : "First name cannot have fancy characters"
    },
    {
      "code" : 5622,
      "field" : "password",
      "message" : "Password cannot be blank"
    }
  ]
}
```

Restful API

Conventions - others

- KISS
- Versons:
  - `https://api.domain.com/v1/`
  - `https://v1.api.domain.com/`
  - ```
    GET https:///orders/1325 HTTP/1.1
    Accept: application/json
    Version: 1
    ```
  - ```
    GET https:///orders/1325 HTTP/1.1
    Accept: application/json; version=1
    ```
  - `http://localhost:9090/investors?version=1`
- Always Use HTTPs & Use Password Hash
- Never Expose sensitive information on URLS
- Input Parameter Validation
- Consider adding Timestamp in Request

Restful API

Conventions - examples of request URI

- Basic of CRUD:
  - `POST https://abc.com/v1/customers`
  - `GET https://abc.com/v1/customers`
  - `GET https://abc.com/v1/customers/445839`
  - `PUT https://abc.com/v1/customers/445839`
  - `DELETE https://abc.com/v1/customers/445839`
- Expose the URI:
  - `POST https://abc.com/v1/orders` or
  - `POST https://abc.com/v1/customers/445839/orders`
  - `GET https://abc.com/v1/customers/445839/orders`
  - `GET https://abc.com/v1/customers/445839/orders/7384/items`

Restful API

Conventions - examples of response JSON

- With HATEOAS:
  -
    {
    "departmentId": 10,
    "departmentName": "Administration",
    "locationId": 1700,
    "links": [
    {
    "href": "10/employees",
    "rel": "employees",
    "type" : "GET"
    }
    ]
    }

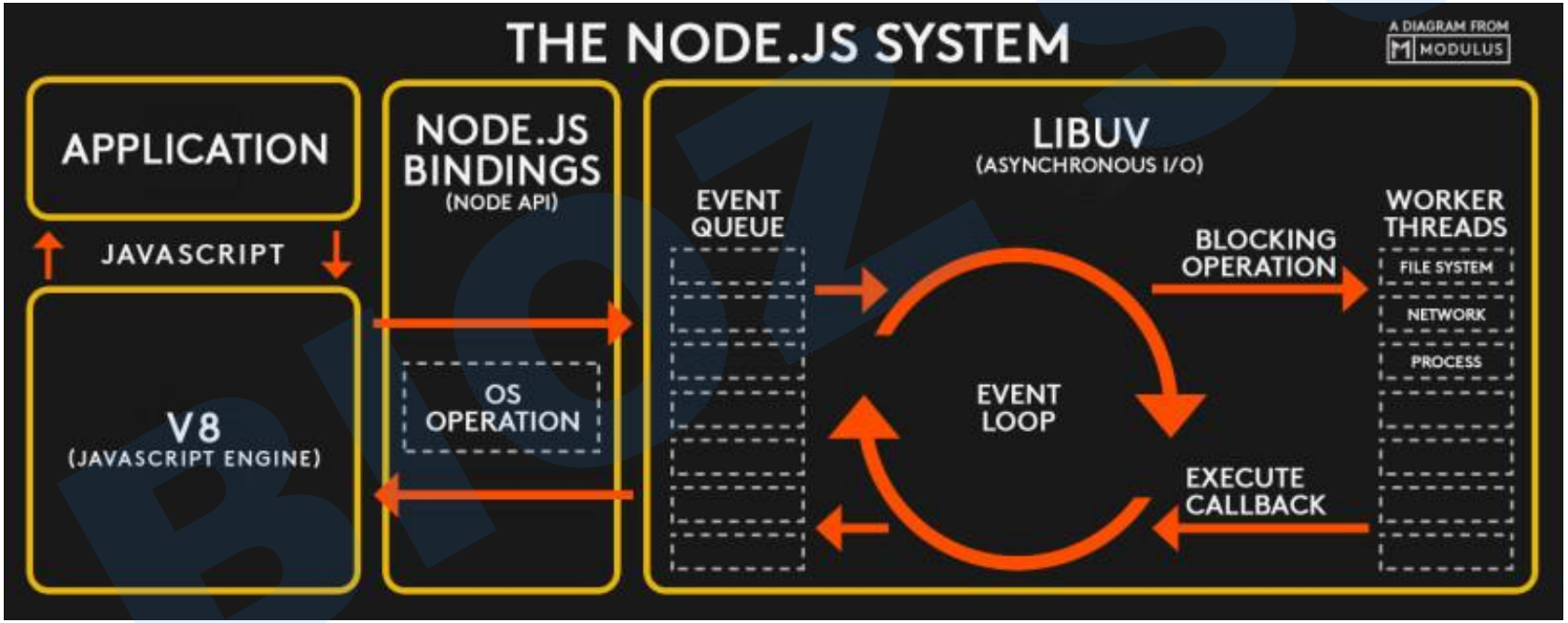
Restful API

Node.Js - Overview

- Event-Driven, Asynchronous IO, Server-Side Javascript library in C
- Open Source
- Single Threaded but Highly Scalable
- No Buffering
- Available on Windows, Unix Systems
- As a service on Azure, Heroku

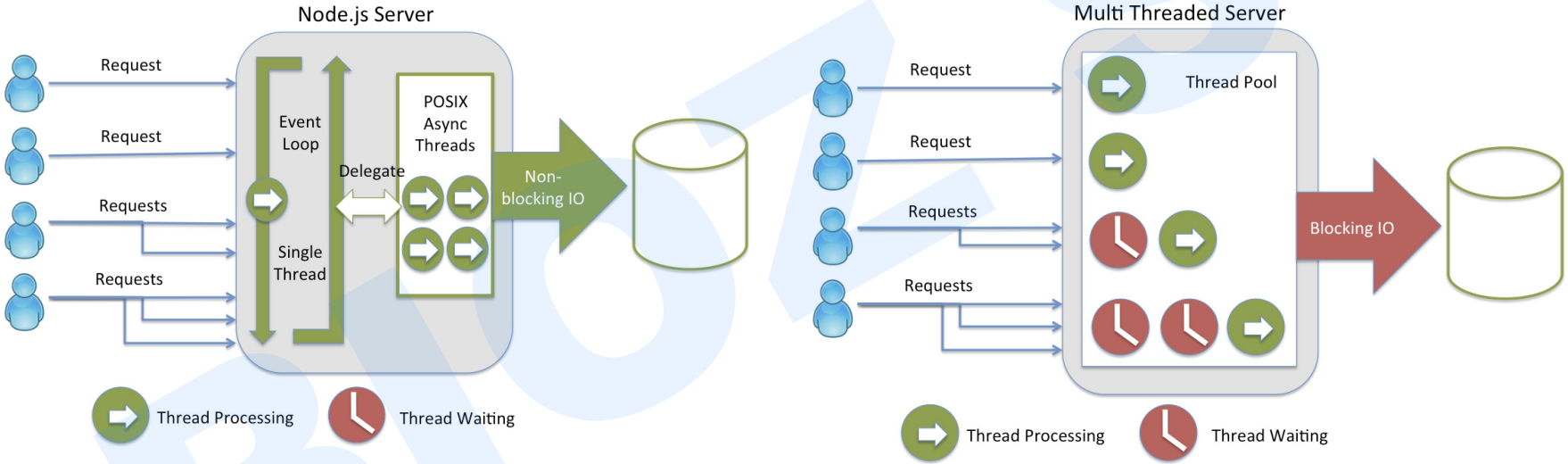
Restful API

Node.Js - Architecture



Restful API

Node.Js - Event Loop



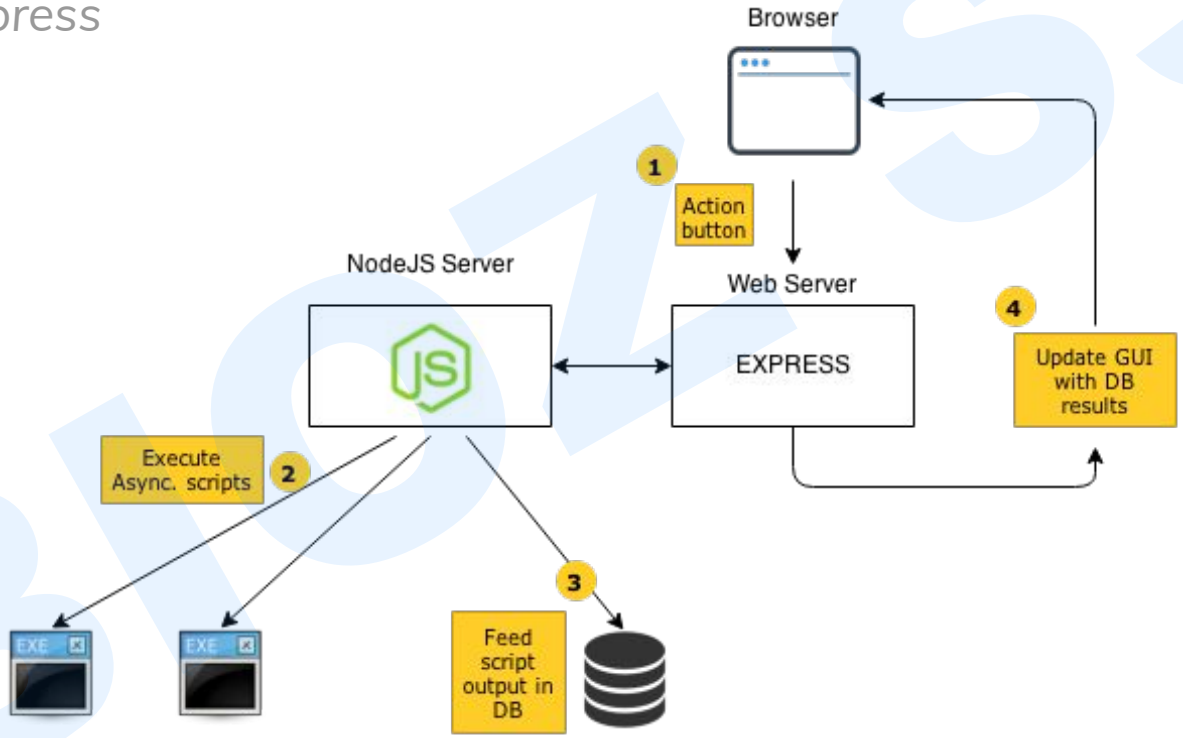
Restful API

Node.Js -> Hello World

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

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

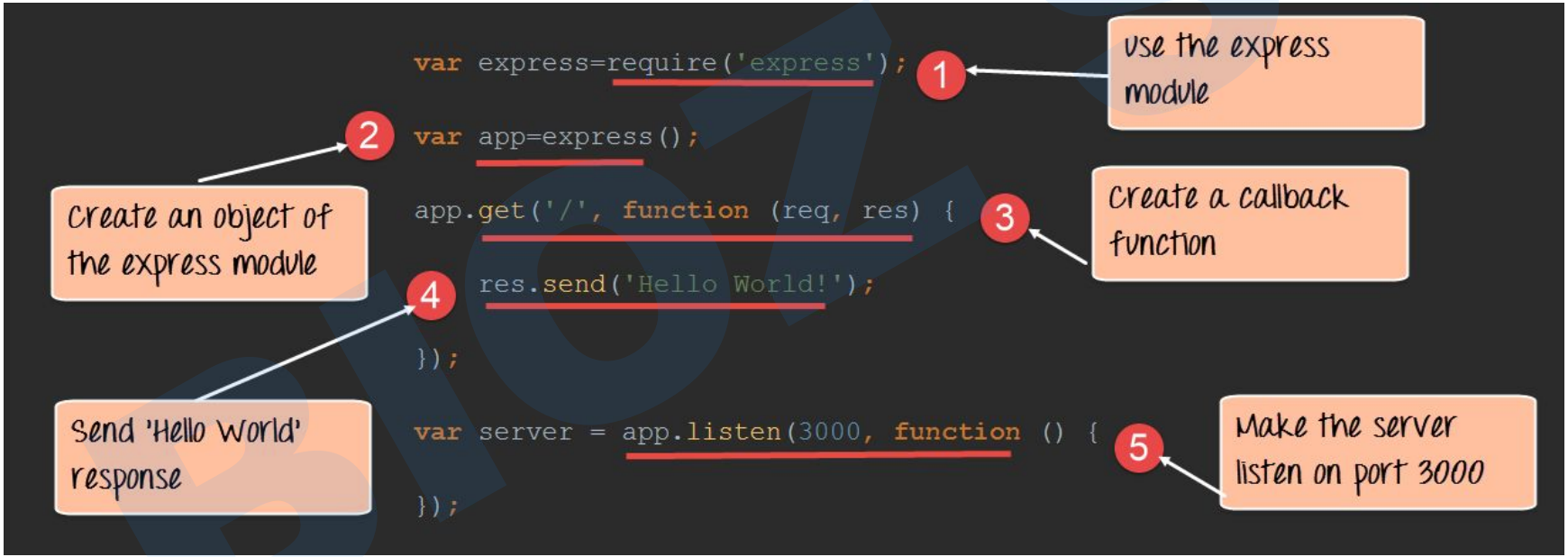
Restful API

Node.Js + eXpress



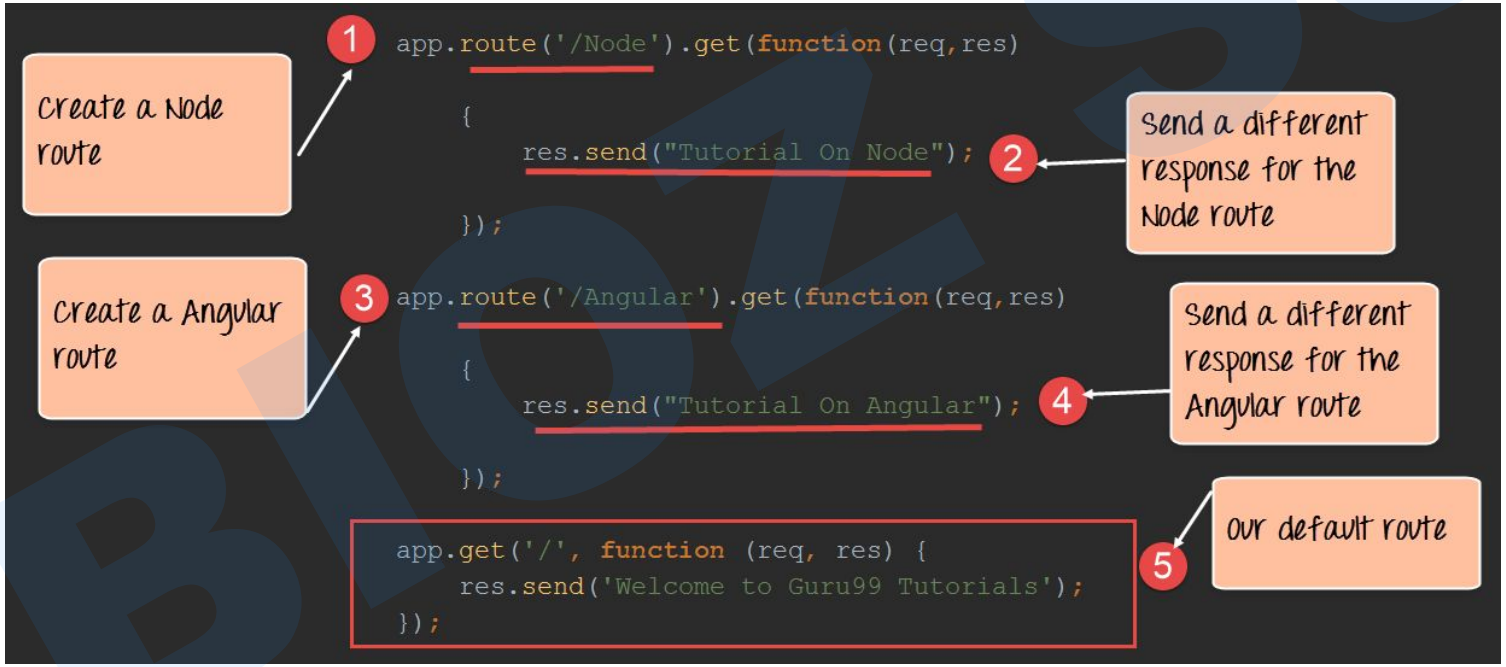
Restful API

Node.Js + eXpress



Restful API

Node.Js + eXpress



Restful API

Node.Js + eXpress -> Hello World

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

Restful API

Node.Js -> some common modules

- **Async.js**: provide structure for working with asynchronous JS.
- **PM2**: Process Management
- **Socket.IO**: support websocket
- **Passport**: authentication module
- **Nodemailer**: sending email
- **Mongoose**: MongoDB object modeling tool
- **Sharp**: image processing module
- **Multer**: a middleware for handling multipart/form-data for uploading file
- **Validator**: String validators and sanitizers
- **Bcryptjs**: Work with the password

Restful API

Node.Js -> some common modules

- **cron**: manage the cron task scheduler.
- **extract-zip**: unzip
- **fluent-ffmpeg**: A fluent API to FFMPEG
- **jsonwebtoken**: JSON Web Token implementation
- **body-parser**: body parsing middleware
- **fs-extra**: file system methods support
- **moment**: Parse, validate, manipulate, and display dates
- **sequelize**: a multi dialect ORM
- **Apidoc**: Restful web API documentation generator
- ... https://www.npmjs.com/

Restful API

Node.Js -> Demo and Explain a Template Project

- Project for MongoDB

  https://sectic@bitbucket.org/sectic_s3lab/s3lab_nodejs_mongodb_tpl.git

- Project for SQL DB

  https://sectic@bitbucket.org/sectic_s3lab/s3lab_nodejs_mysql_tpl.git

Q & A

Cảm ơn đã theo dõi

Hy vọng cùng nhau đi đến thành công.