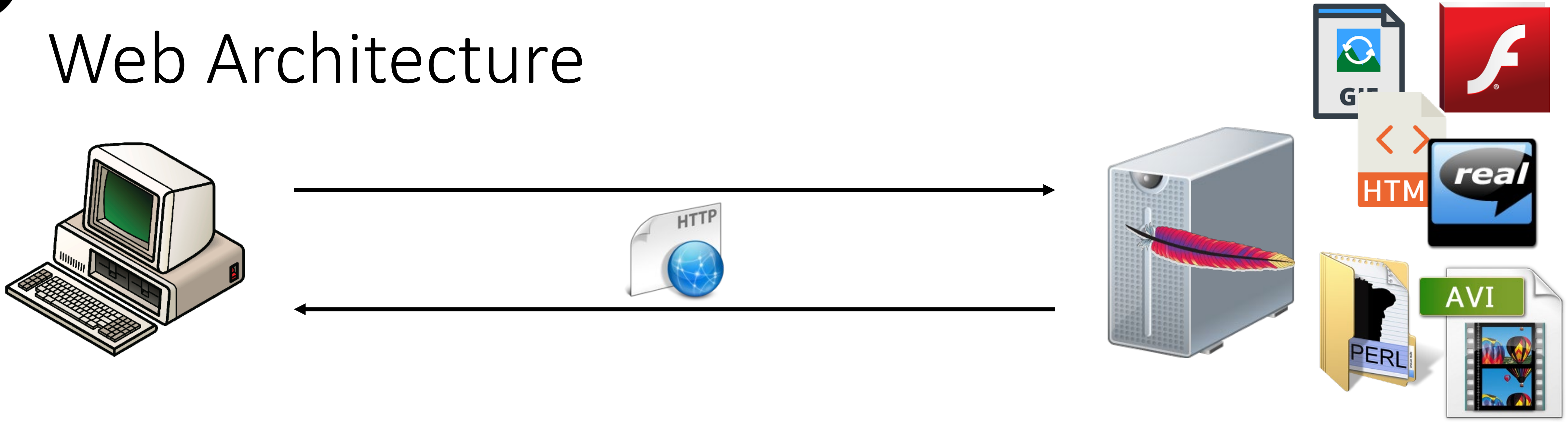
RESTful Web Services

Web Application

- NCSA (National Centre for Supercomputing Applications) create a way for programs to be executed inside a web server
  - The application could return non static dynamically generated hypertext documents!
  - Content no longer limited by what is uploaded to the web server
- This environment for executing applications inside a web server is call CGI (Common Gateway Interface)
- Early CGIs were Perl and Bash shell scripts

Web Architecture



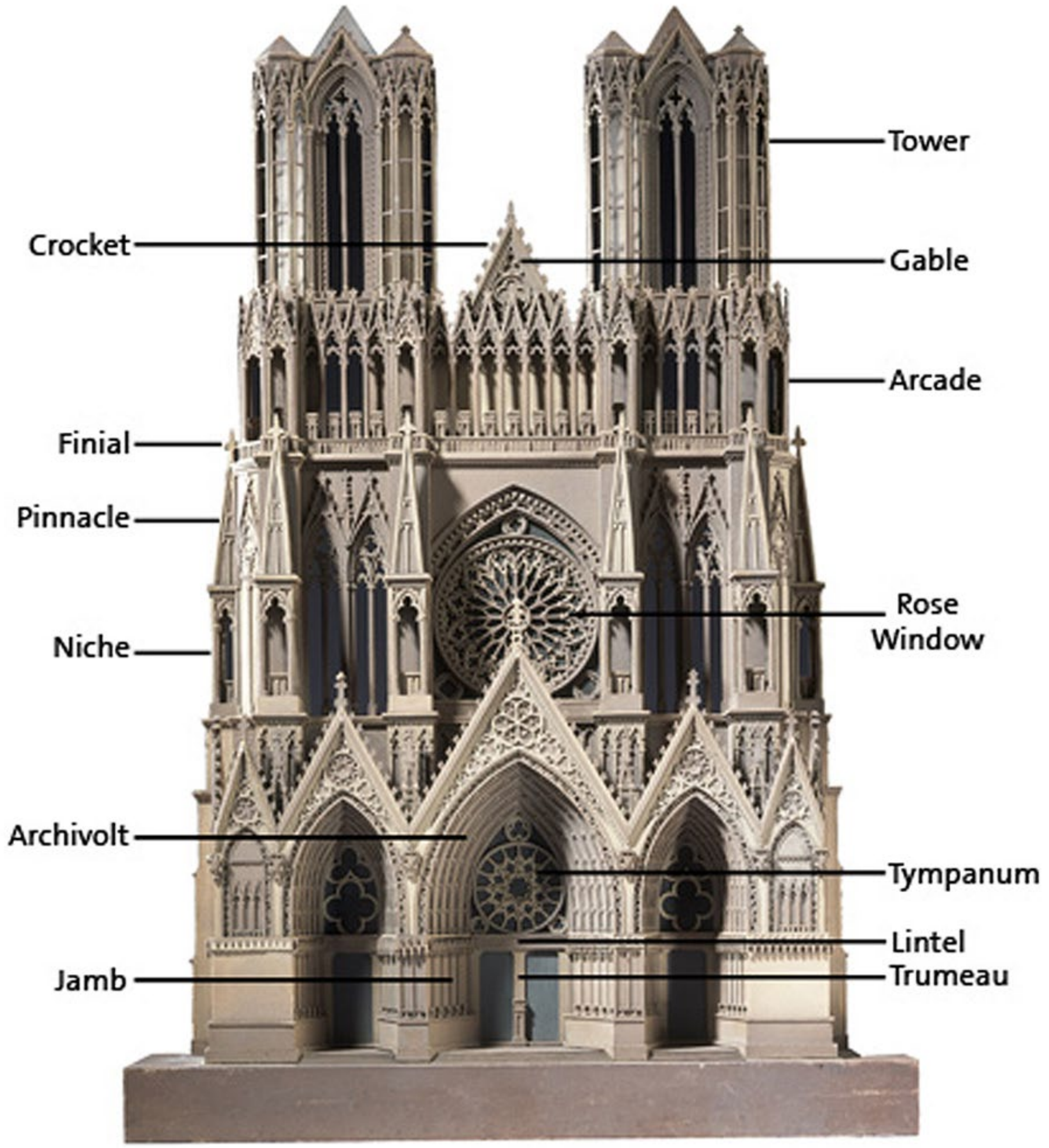
- Web has a very simple architecture compared to CORBA, DCOM and other distributed architectures
- Dual purpose – serve files and application services
- Web servers are free
- Very easy to develop applications, no specialized tools were require
- HTTP is very easy to understand and debug
- All it requires is an architectural blue print which is to come in 2000

RESTful Web Services

- HTTP based application services allows a way for different computer systems to interoperate
  - The requestor does not have to be a browser
- Roy Fielding formalizes RESTful Web Service in his 2000 PhD thesis
  - Architectural Styles and the Design of Network-based Software Architectures
  - https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
- Coined the term **REST**
  - **RE**presentational **S**tate **T**ransfer

RESTful Web Services

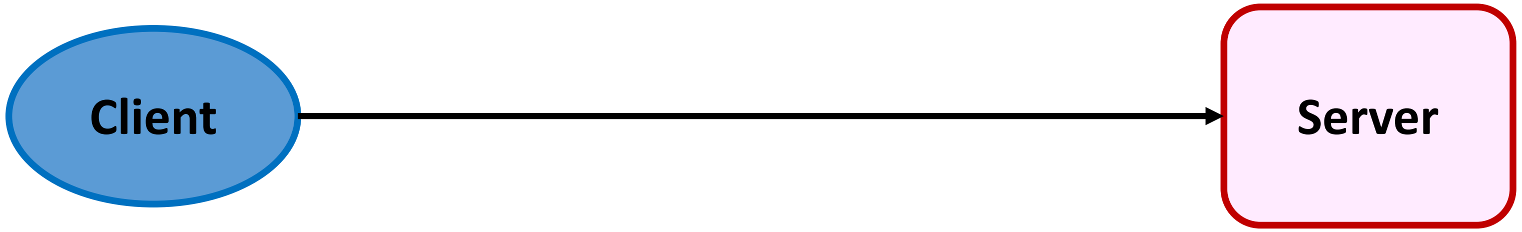
- Architectural style for distributed system
  - Building distributed systems based on a set of principles
- Architectural style consist of a set of constrains applied to elements within the architecture
- Jazz music
  - Music elements: key, notes, timing, phrasing, etc
  - Architectural constrains: syncopation, improvisation, etc.



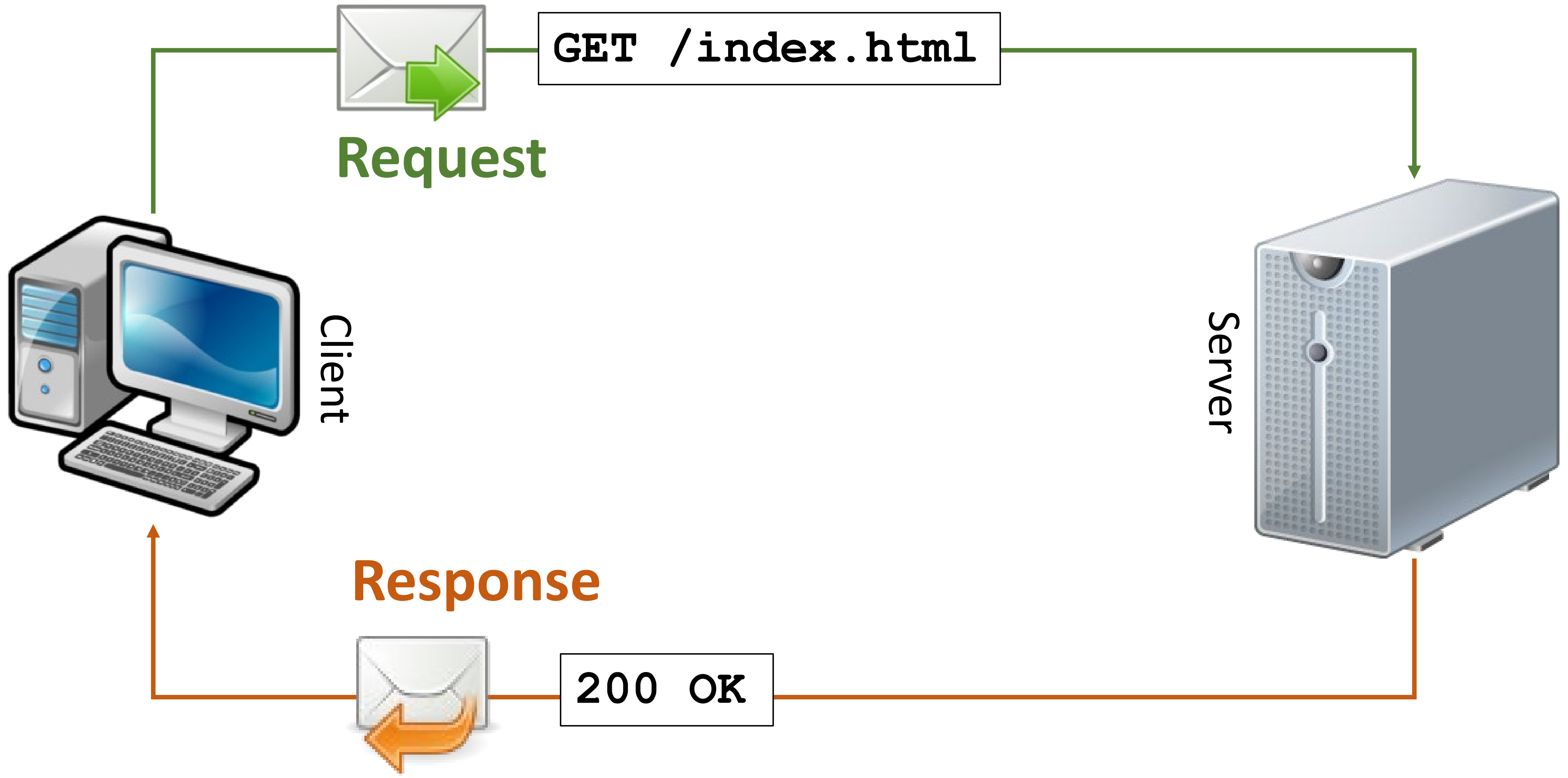
http://www.vam.ac.uk/__data/assets/image/0004/185062/2006bb3218_rheims_cathedral_model_annotated.jpg

REST Constraint 1 - Client Server

- Allows for separation of concerns
- Separation of client specific concerns with server specific concerns
  - Eg UI, affordance, persistence, etc.

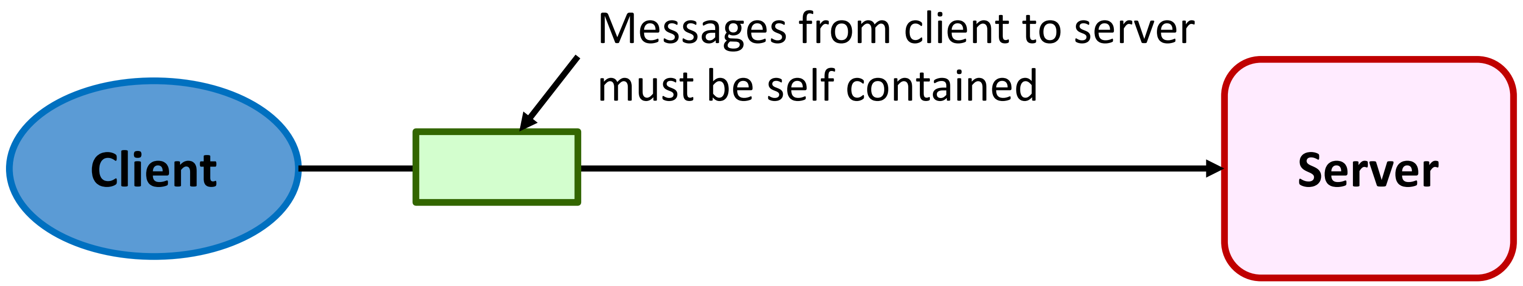


The Web is Client-Server

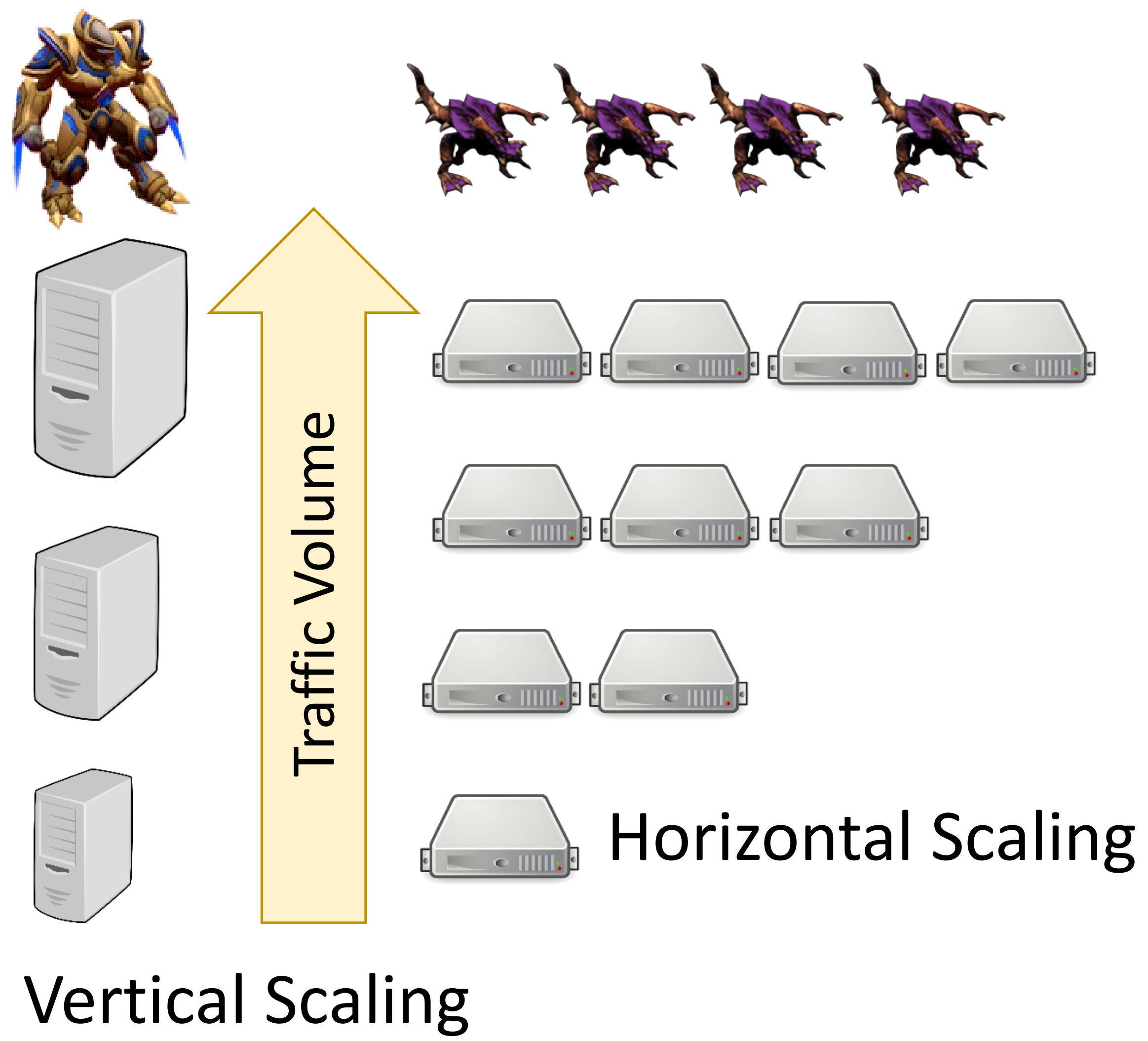


REST Constraint 2 - Stateless

- Communication between client and server must be stateless
  - Viz. Stateless server
- Each request from client to server must contain all information required to understand and to process that request



Statelessness for Scaling



- Scaling is the capability of the system to handle more workload by provisioning more resources
- Two types of scaling
  - Horizontal scaling - scales by provision more Pods
    - Applications must be stateless allowing the ingress controller to route the request to any Pod
  - Vertical scaling - scaling by giving the application more resources
    - Application must be able to utilize the extra resources eg. more vCPUs or memory

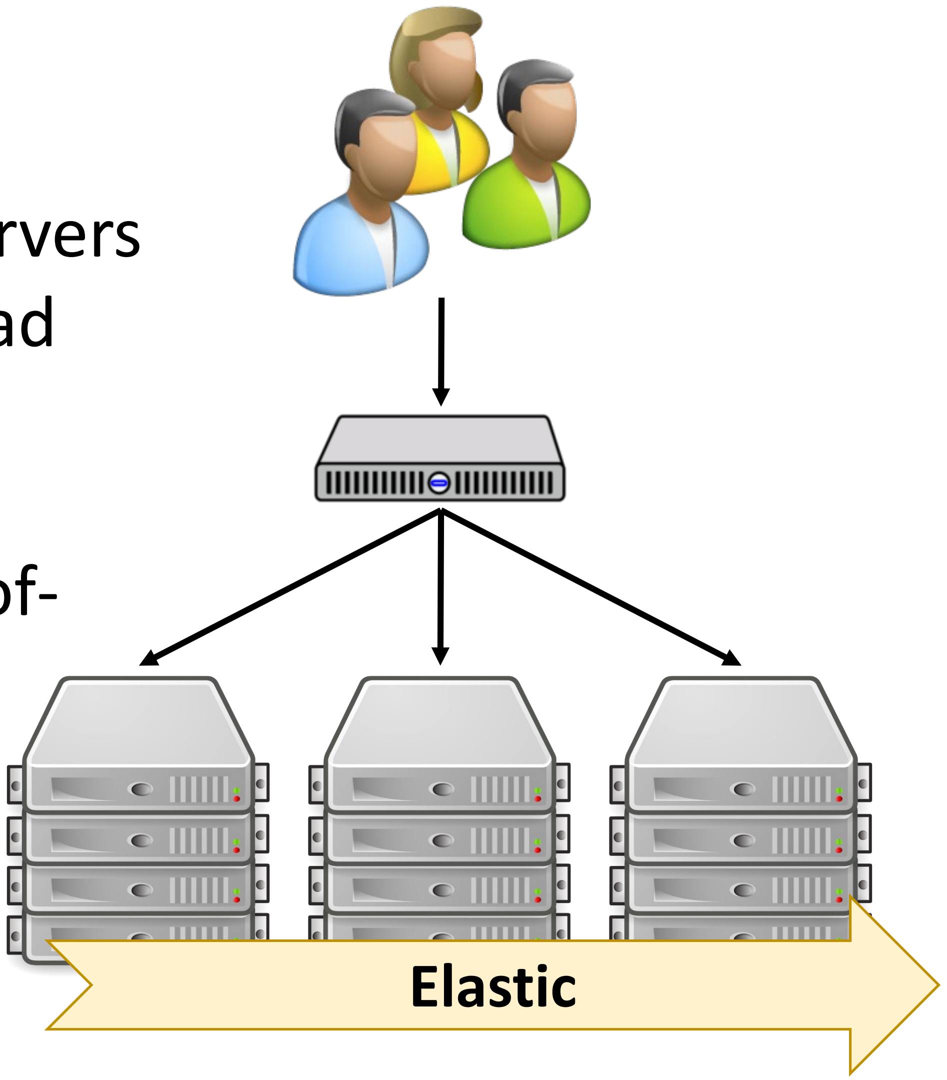
Why is Statelessness Important?

- Predicable - know how much work needs to be done by looking at the request
- Reliable - can easily recover from error because all information required to process a request is contained within that request
  - Can be processed by the next available server if the current one fails
- Scalable - not having to store states allows servers to be allocated when workload is high and scale down when workload is low

The Web is Stateless

Easily increase the number of servers (scale out) to handle the workload
Makes the environment elastic

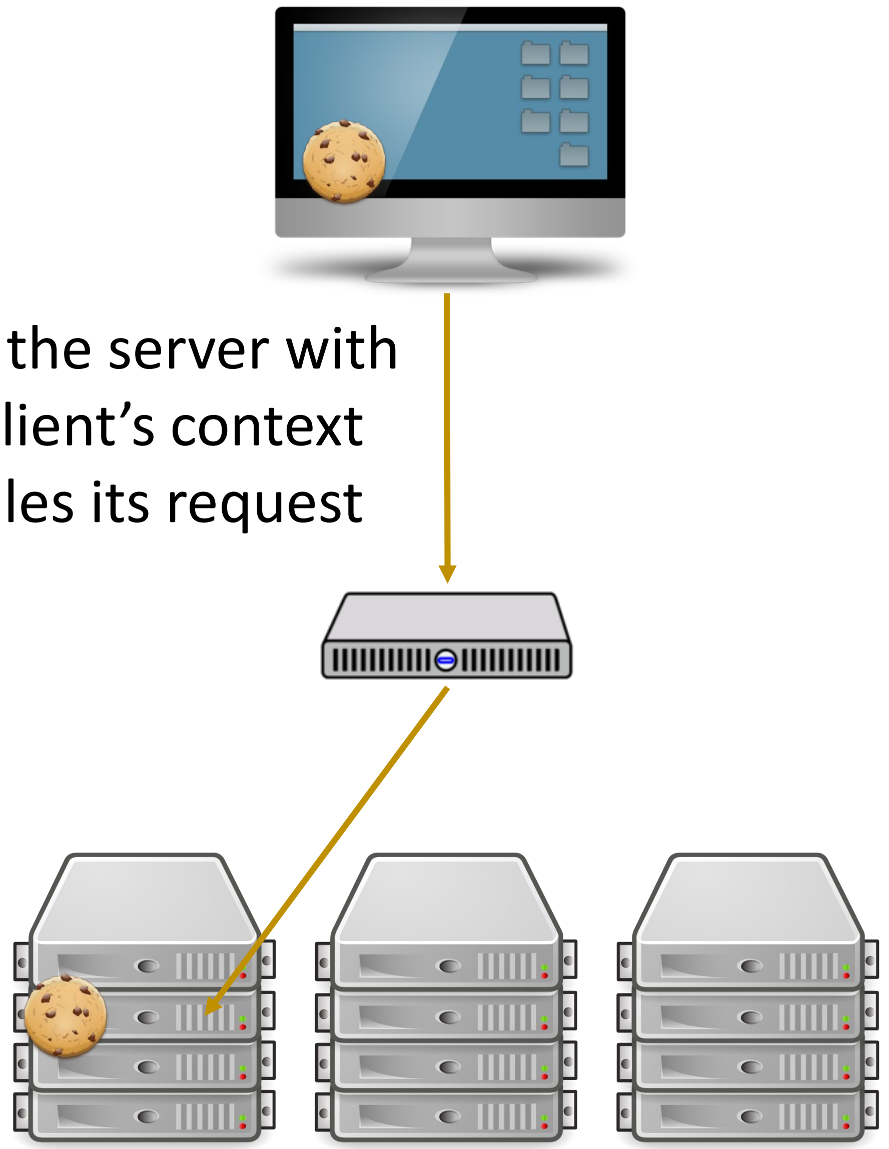
Scaling could be based on time-of-day, resource utilization or by anticipation



Stateful

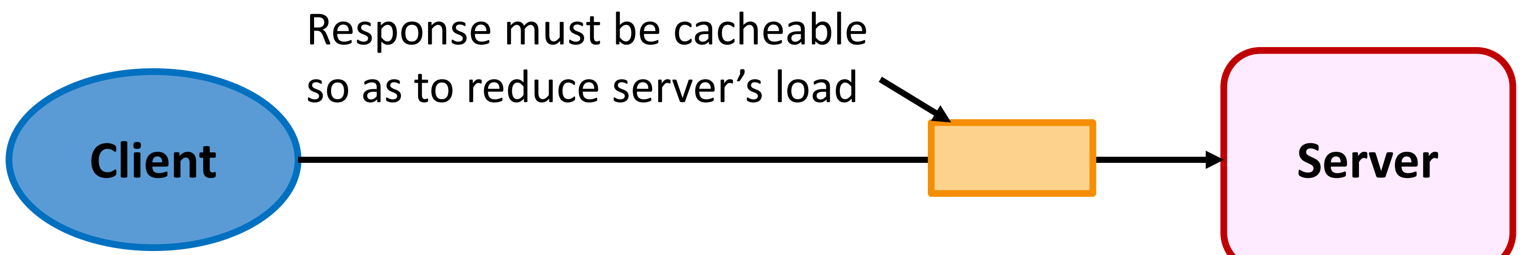
- Some process require state to work, viz. require to remember the context
  - Context - what has transpired between the application and the client
  - Eg. shopping session
- To uniquely identify a client and make sure that only the server that has the context of that client process its request
  - Eg. using cookies to associate with a server side session

Only the server with the client's context handles its request



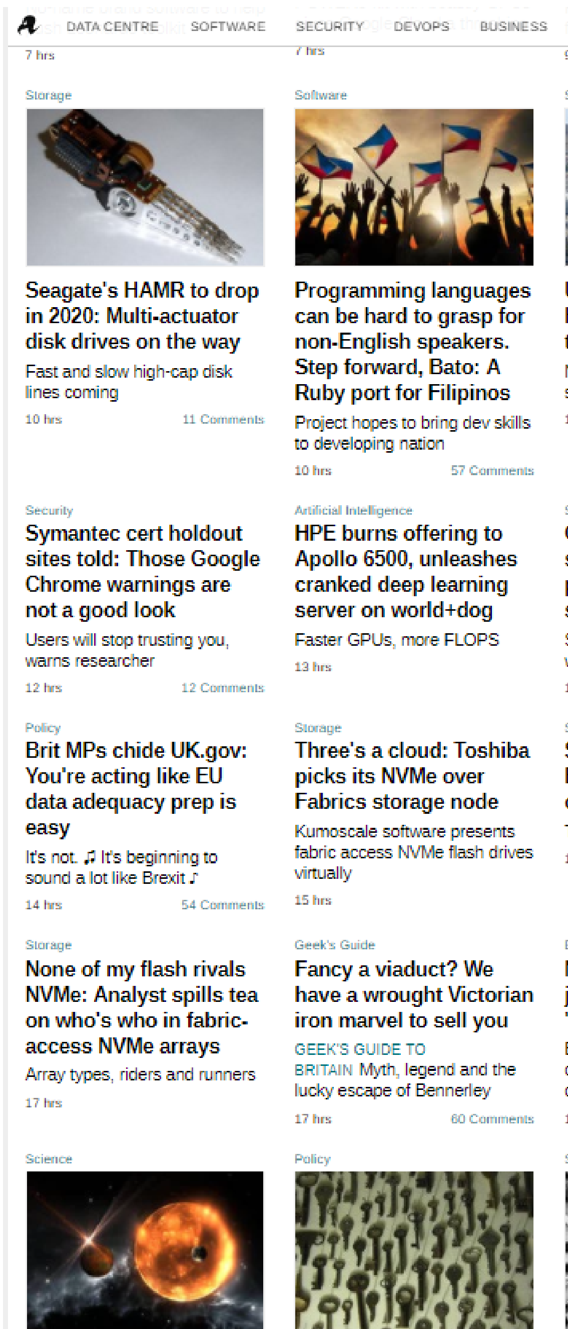
REST Constraint 3 - Cache

- Data from a response may be implicitly or explicitly labelled as cacheable or non cacheable
  - A cache has the right to return a cache data for an equivalent request
  - Avoid full trip back to the server



Cache

- Caching improves response time
  - Response can be returned by web caches (on the Internet)
  - Response can be returned by your browser (on your computer)
- What to cache?
  - Static resources - resources that are not updated frequently
    - Eg. images, pdf, audio, etc
  - Hard/expensive to create resources
    - Eg. news paper front page
  - Often used resources
- How to cache?
  - By time - the response will be stale in 30 minutes
  - By content - need to retrieve if the content changes


DATA CENTRE SOFTWARE SECURITY DEVOPS BUSINESS

7 hrs

7 hrs

Storage

Seagate's HAMR to drop in 2020: Multi-actuator disk drives on the way

Fast and slow high-cap disk lines coming

10 hrs 11 Comments

Software

Programming languages can be hard to grasp for non-English speakers. Step forward, Bato: A Ruby port for Filipinos

Project hopes to bring dev skills to developing nation

10 hrs 57 Comments

Security

Symantec cert holdout sites told: Those Google Chrome warnings are not a good look

Users will stop trusting you, warns researcher

12 hrs 12 Comments

Artificial Intelligence

HPE burns offering to Apollo 6500, unleashes cranked deep learning server on world+dog

Faster GPUs, more FLOPS

13 hrs

Policy

Brit MPs chide UK.gov: You're acting like EU data adequacy prep is easy

It's not. ♫ It's beginning to sound a lot like Brexit ♪

14 hrs 54 Comments

Storage

Three's a cloud: Toshiba picks its NVMe over Fabrics storage node

Kumoscale software presents fabric access NVMe flash drives virtually

15 hrs

Storage

None of my flash rivals NVMe: Analyst spills tea on who's who in fabric-access NVMe arrays

Array types, riders and runners

17 hrs

Geek's Guide

Fancy a viaduct? We have a wrought Victorian iron marvel to sell you

GEEK'S GUIDE TO BRITAIN Myth, legend and the lucky escape of Bennerley

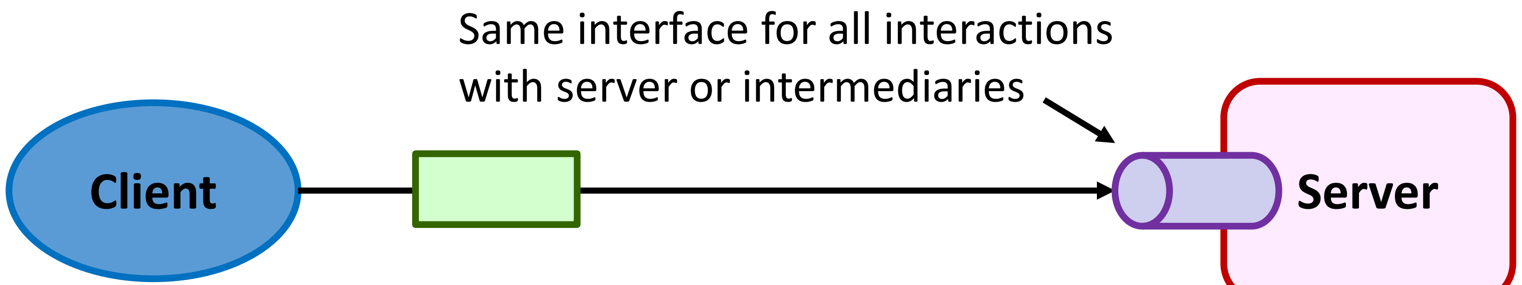
17 hrs 60 Comments

Science

Policy

REST Constraint 4 - Uniform Interface

- Provide a general interface for client to invoke the server on all layers of the system


Same interface for all interactions with server or intermediaries
Client
Server

- A uniform interface has the following sub-constraints
  - Identify resources thru identifiers
  - Manipulation of resources through its representation
  - Self describing message
  - Hypermedia as the engine of application states

Uniform Interface - Resources Identification

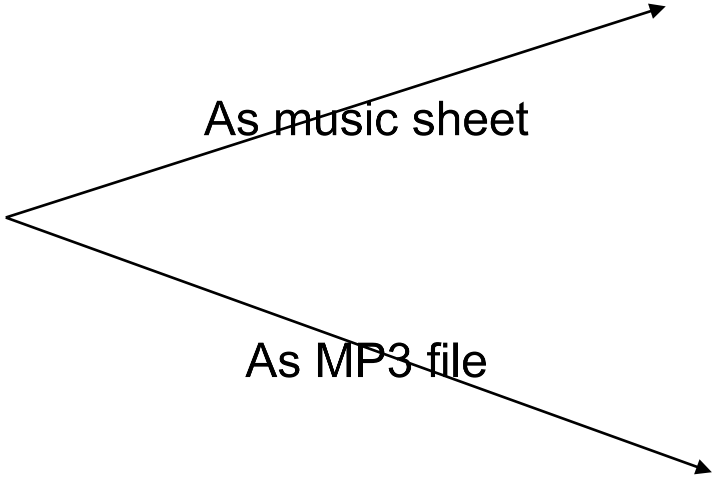
Every resource can be uniquely identified

$$\texttt{GET}\ \overbrace{\texttt{/doc/test.html}}^{\text{A resource name within the server}}$$

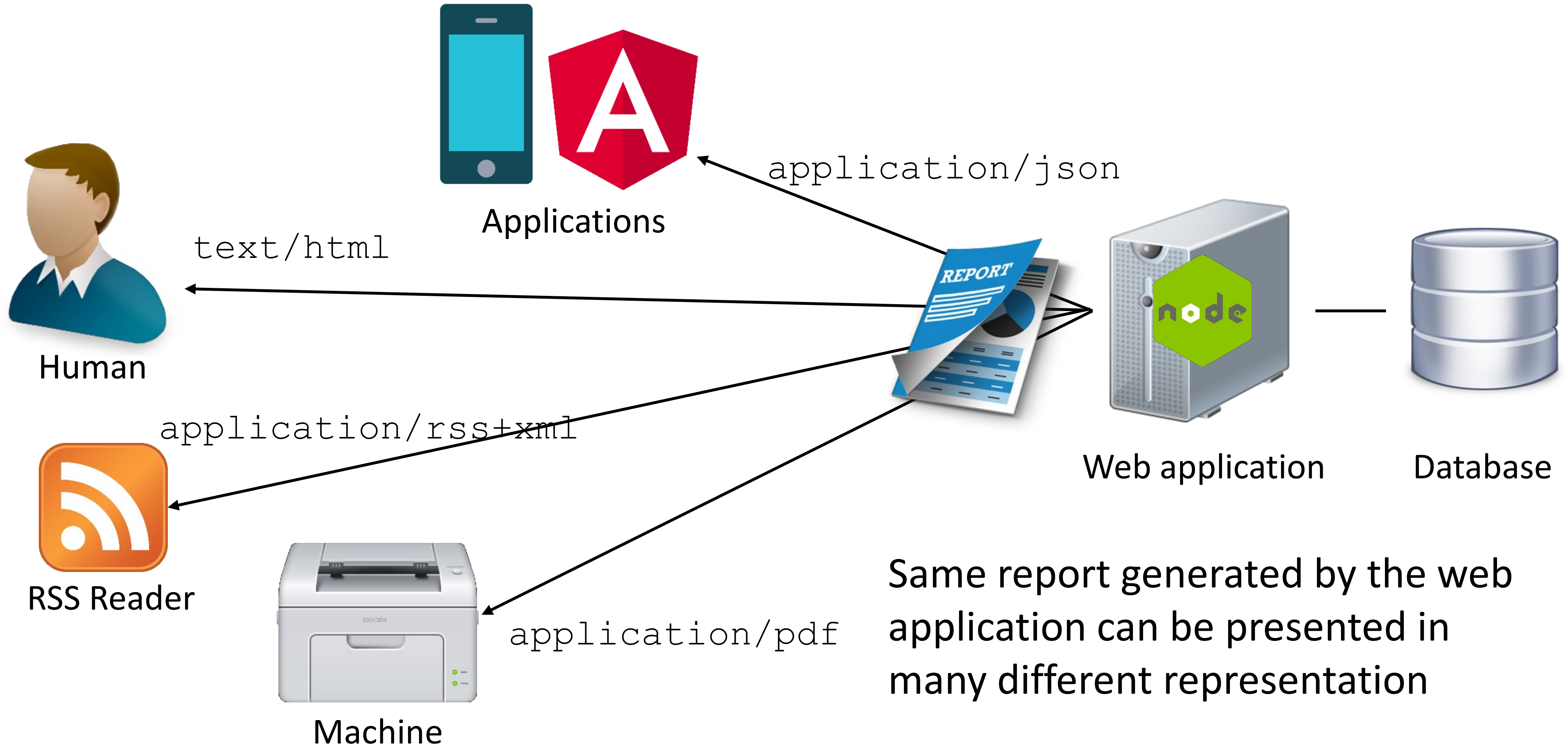
Uniform Interface – Manipulate Resources thru its Representation

Representation

Canon in D



Uniform Interface – Manipulate Resources thru its Representation

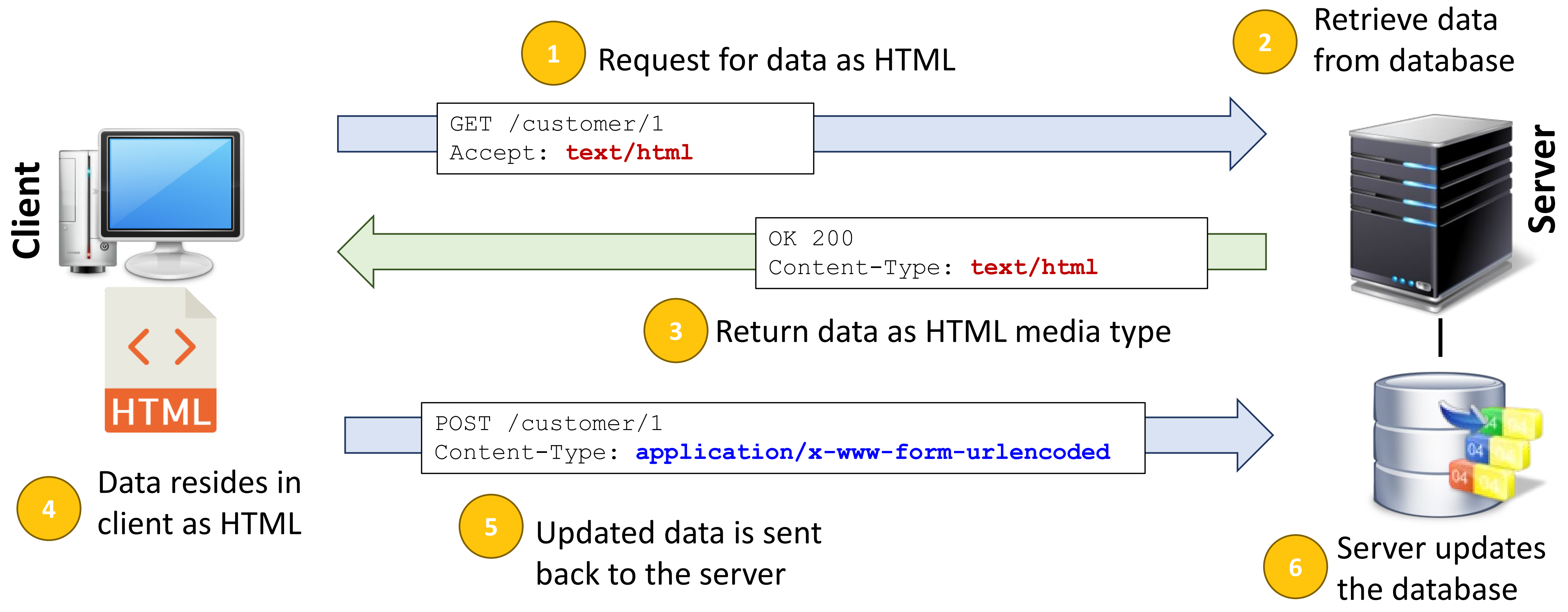


Same report generated by the web application can be presented in many different representation

Uniform Interface – Manipulate Resources thru its Representation

- A single resource can be represented by many different types of media
- Content negotiation - a client can potentially present a list of media types to the server
  - Indicating its choice and preference for a particular resource

Uniform Interface – Manipulate Resources thru its Representation



1 Request for data as HTML
2 Retrieve data from database
Client
Server
3 Return data as HTML media type
HTML
4 Data resides in client as HTML
5 Updated data is sent back to the server
6 Server updates the database

```
GET /customer/1
Accept: text/html
```

```
OK 200
Content-Type: text/html
```

```
POST /customer/1
Content-Type: application/x-www-form-urlencoded
```

Uniform Interface - Self Describing Messages

- Every message passed from a client to a server must be self contained
  - Like a postcard or letter
- Everything to understand and decode the message is in the message
- Supports stateless servers
- The message semantics is also visible to intermediaries (if any, layer system) to understand and cache it

```
GET /search?limit=50&q=hello
Accept: application/json,text/csv=0.3,*/*;q=0.1
```

```
200 OK
Content-Type: text/csv
```

Uniform Interface - HATEOAS

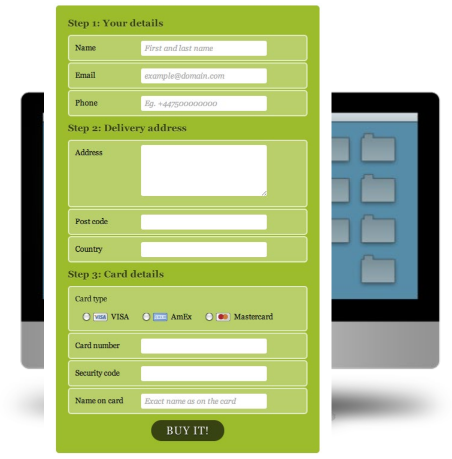
- HATEOAS - Hypermedia as the engine of application states
- The hypermedia/text provides hyperlinks
  - Some link navigate to other hypermedia

```
<a href=http://....>
```

  - Some links provide ways to change the state of the hypermedia

```
<form method="POST" action="/update">
  <input type="text" name="first_name">
  ...
  <button type="submit">Update</button>
</form>
```

Uniform Interface - HATEOAS



Step 1: Your details

Name	*First and last name*
Email	*example@domain.com*
Phone	*Eg. +447500000000*

Step 2: Delivery address

Address	
Post code	
Country	

Step 3: Card details

Card type

○ VISA ○ AmEx ○ Mastercard

Card number	
Security code	
Name on card	*Exact name as on the card*

BUY IT!

1 Wish to update a particular resource

2 Retrieve the resource

`GET /resource/1`

3 Requested resource is returned as a HTML form. The state of the resource is transferred to the client

HTML

4 Form is the representation of the resource's state. Also contains hyperlinks to operate/change the resource

5 Modified state of the resource is send back to the server

`POST /resource/1`

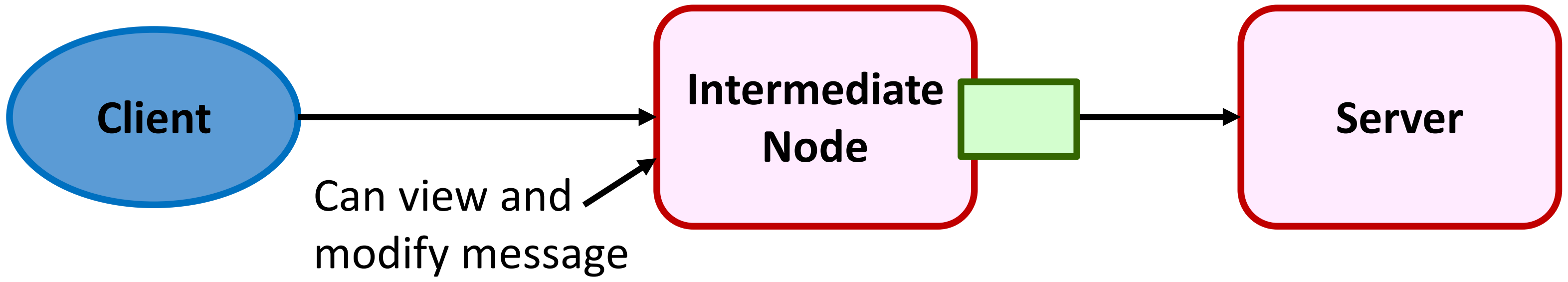
UPDATE

6 State of the resource is updated by the web application

7 State changes because of `POST`

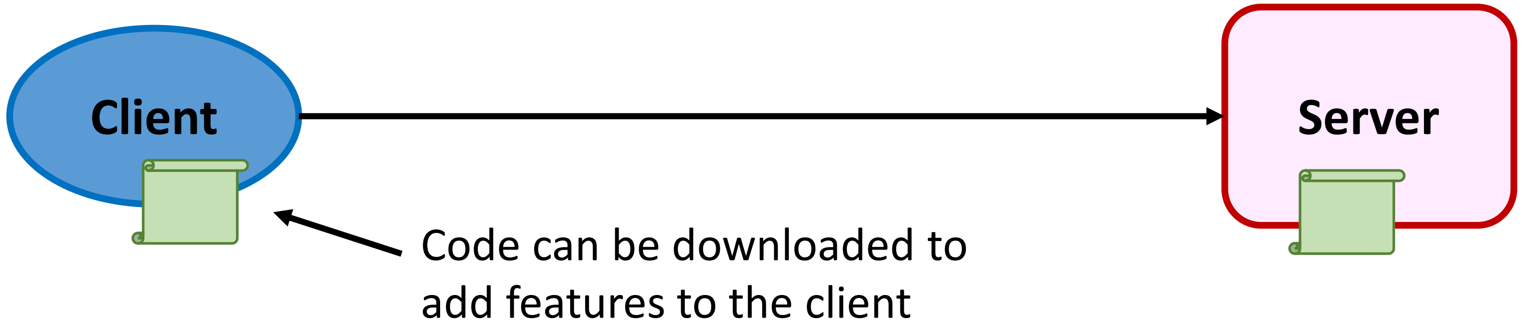
REST Constraint 5 - Layered System

- Application from one layer cannot “see” beyond the intermediate layer
- Promotes decoupling, reduce the complexity of the overall system
  - Wrap legacy application eg. main frame application by presenting modern and simpler ways to access it
- Intermediate layers can modify data

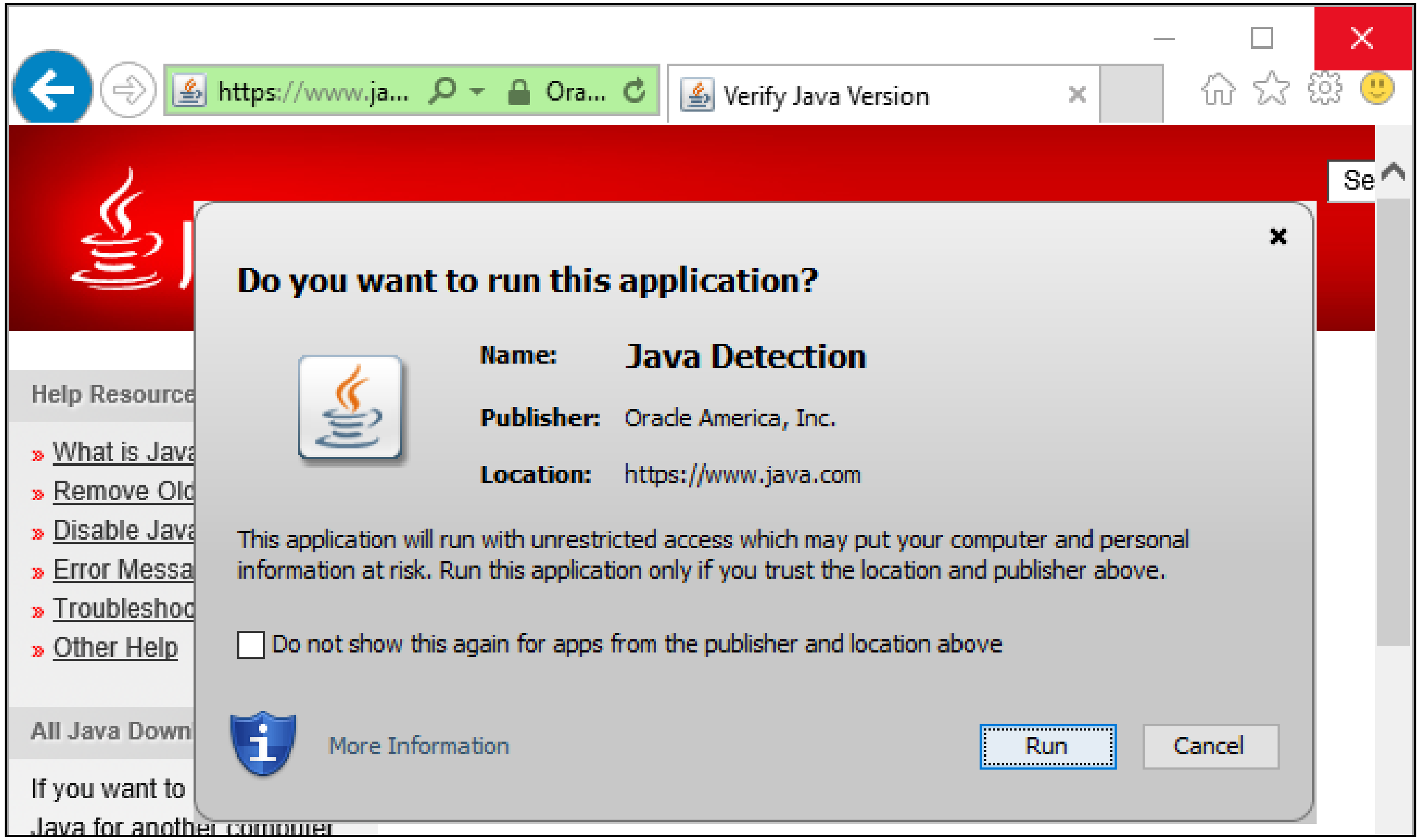


REST Constraint 6 - Code on Demand

- Client functionality can be extended by download code from the server
    - Progressively adds functionality to client



Download Code !



Help Resource

» What is Java
» Remove Old
» Disable Java
» Error Messa
» Troubleshoo
» Other Help

All Java Down

If you want to
Java for another computer

Do you want to run this application?

Name: **Java Detection**

Publisher: Oracle America, Inc.

Location: https://www.java.com

This application will run with unrestricted access which may put your computer and personal information at risk. Run this application only if you trust the location and publisher above.

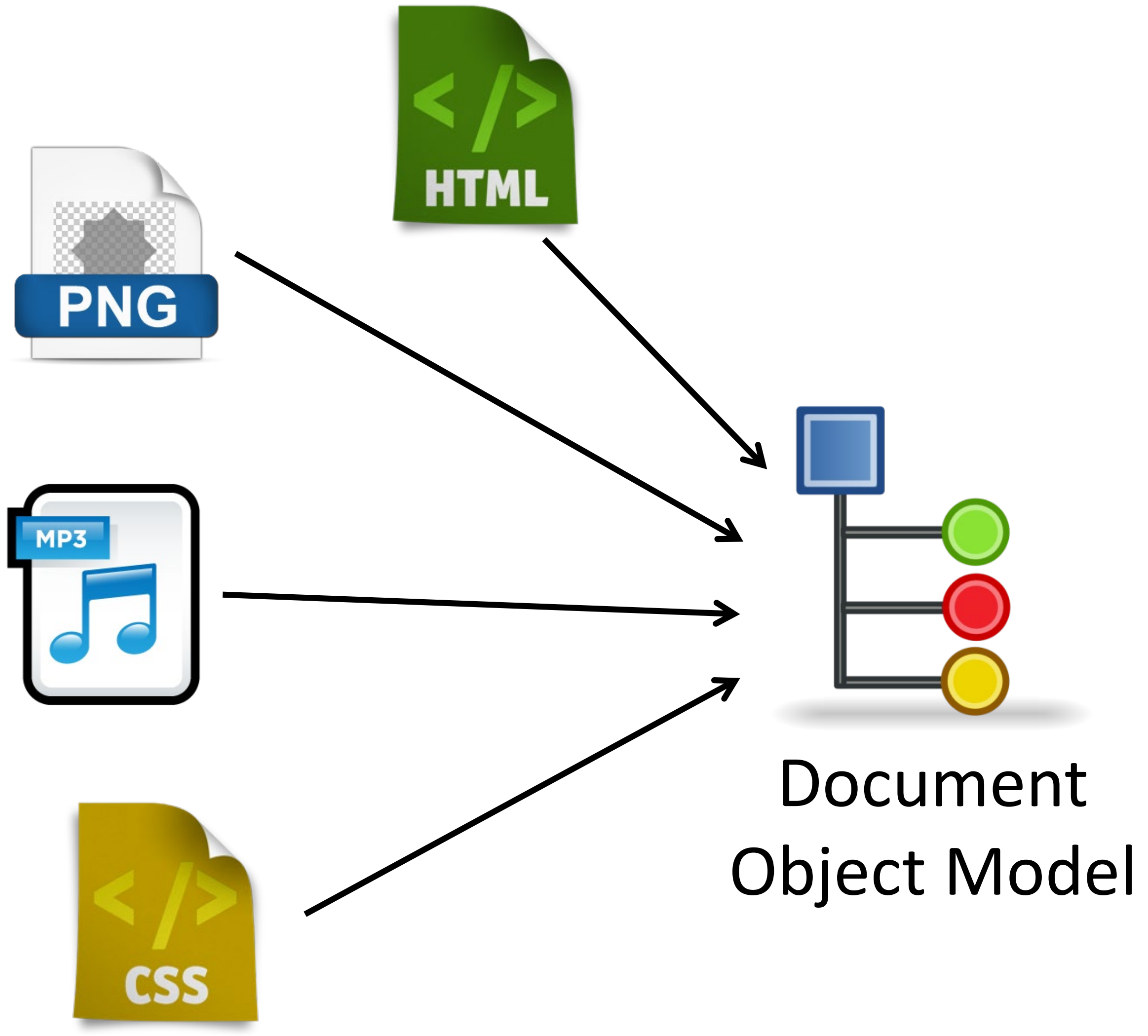
☐ Do not show this again for apps from the publisher and location above

HTML Document

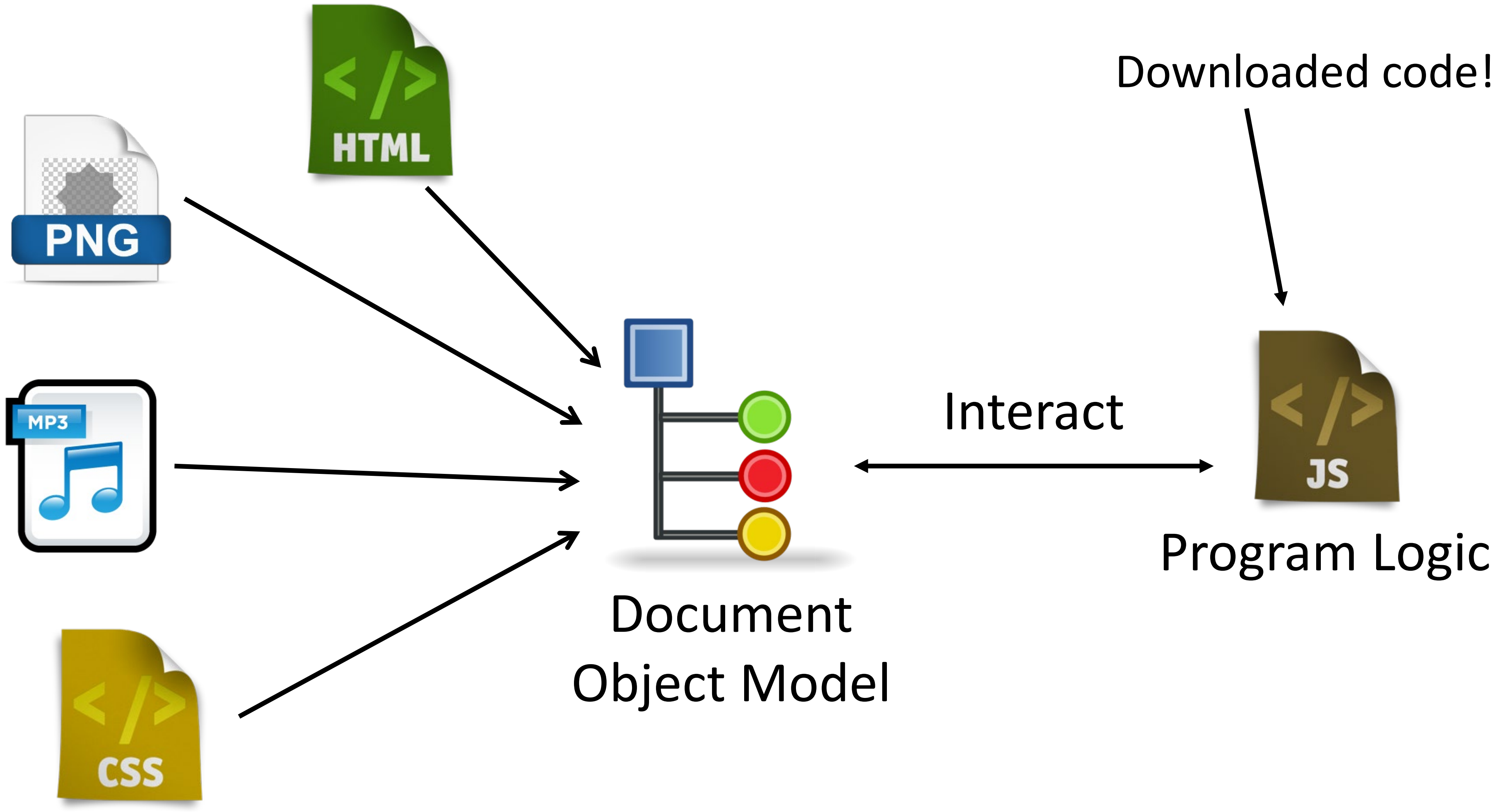
```
<html>
  <head>
    <link rel="stylesheet" href="styles.css">
    <script src="script.js">
  </head>
  <body>
    <h1>hello, world</h1>
    <img src="https://goo.gl/b5a2M7">
    ...
  <body>
</html>
```

Downloading code to your browser

HTML5 Application

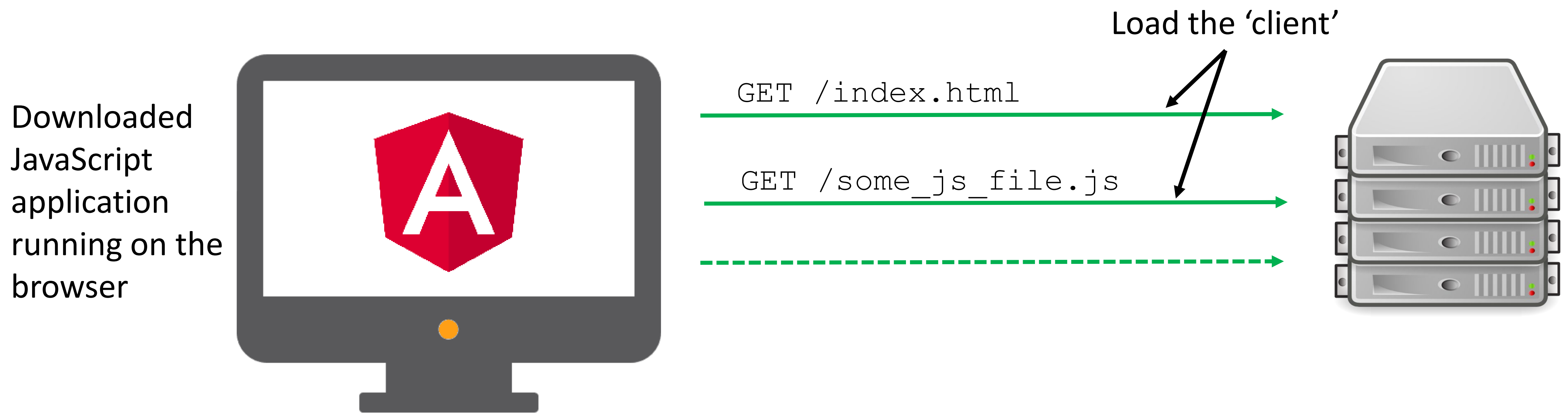


HTML5 Application



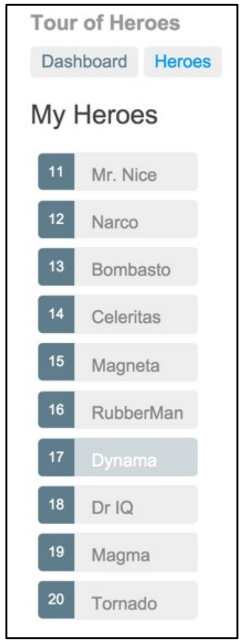
Code on Demand

- The website that provides that API also supply the client to use the API
  - Some API are public some are internal



Code on Demand and HATEOAS

`GET /api/heros`
loaded the master list



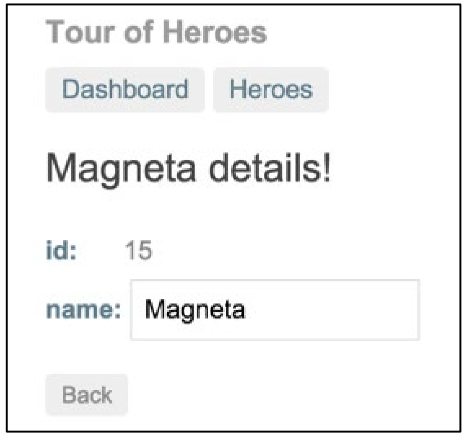
Tour of Heroes

Dashboard | Heroes

My Heroes

11	Mr. Nice
12	Narco
13	Bombasto
14	Celeritas
15	Magneta
16	RubberMan
17	Dynama
18	Dr IQ
19	Magma
20	Tornado

Selecting a hero on the master's list resulted in an AJAX call to retrieve the details



Tour of Heroes

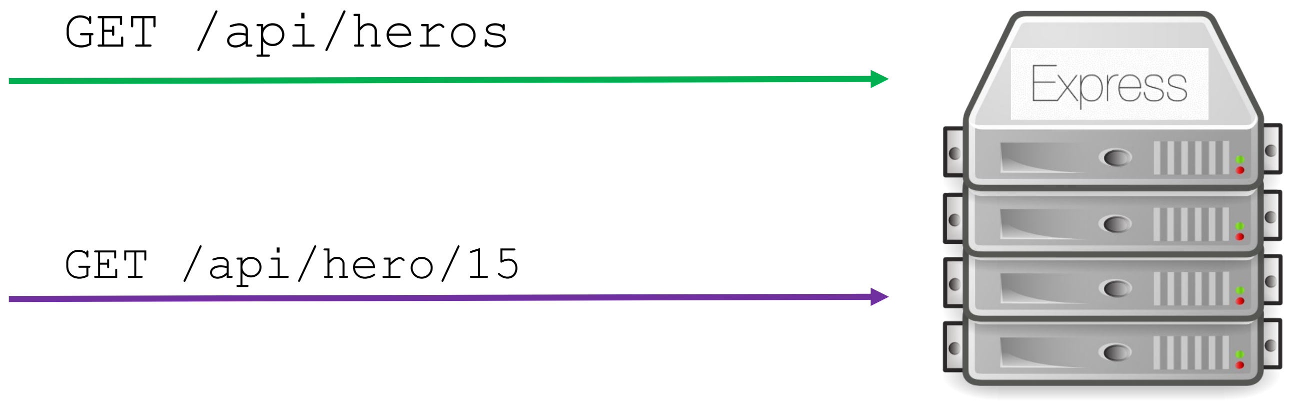
Dashboard | Heroes

Magneta details!

id: 15

name: Magneta

Back



Selection of a hero causes a change in application state