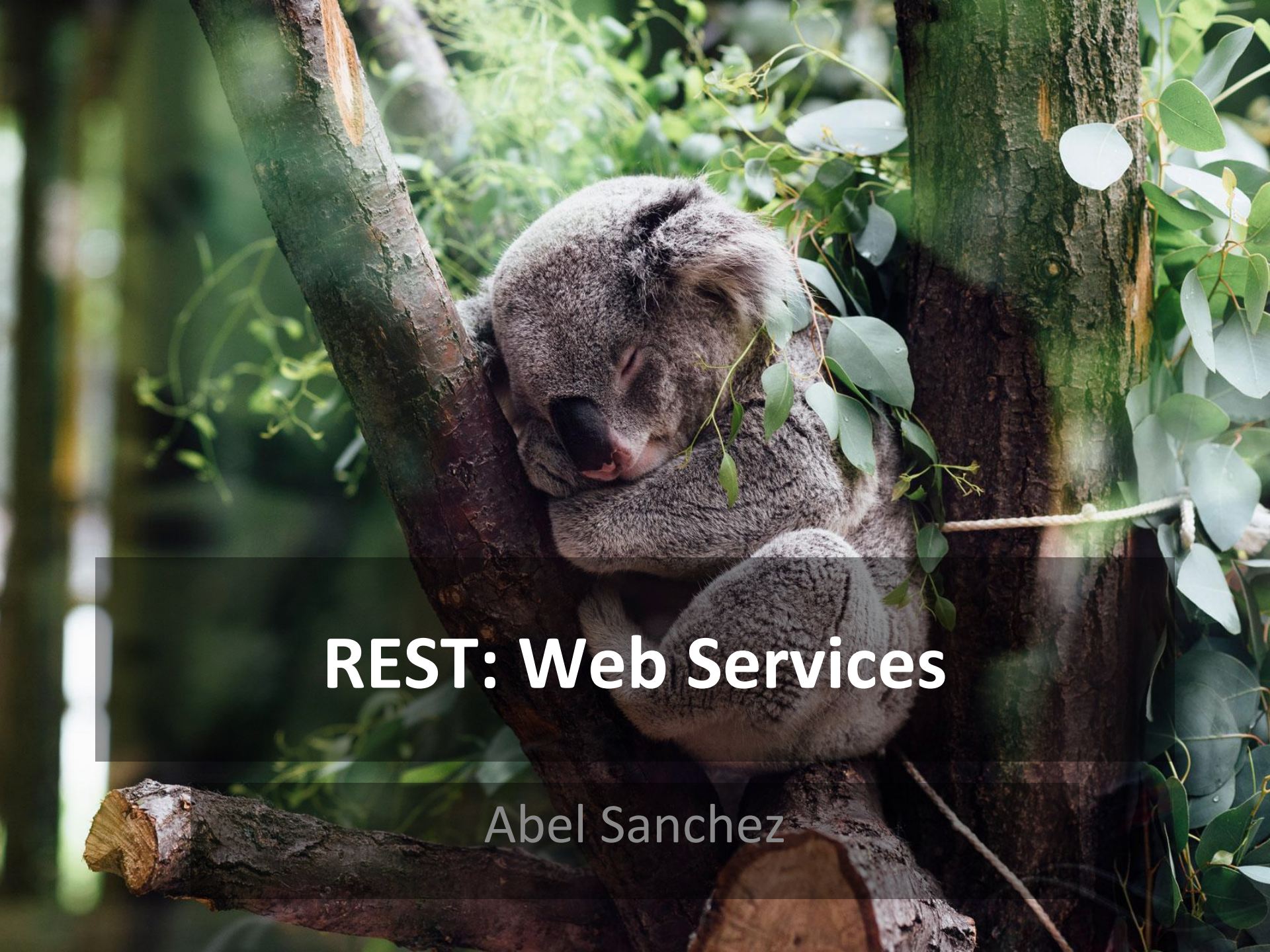
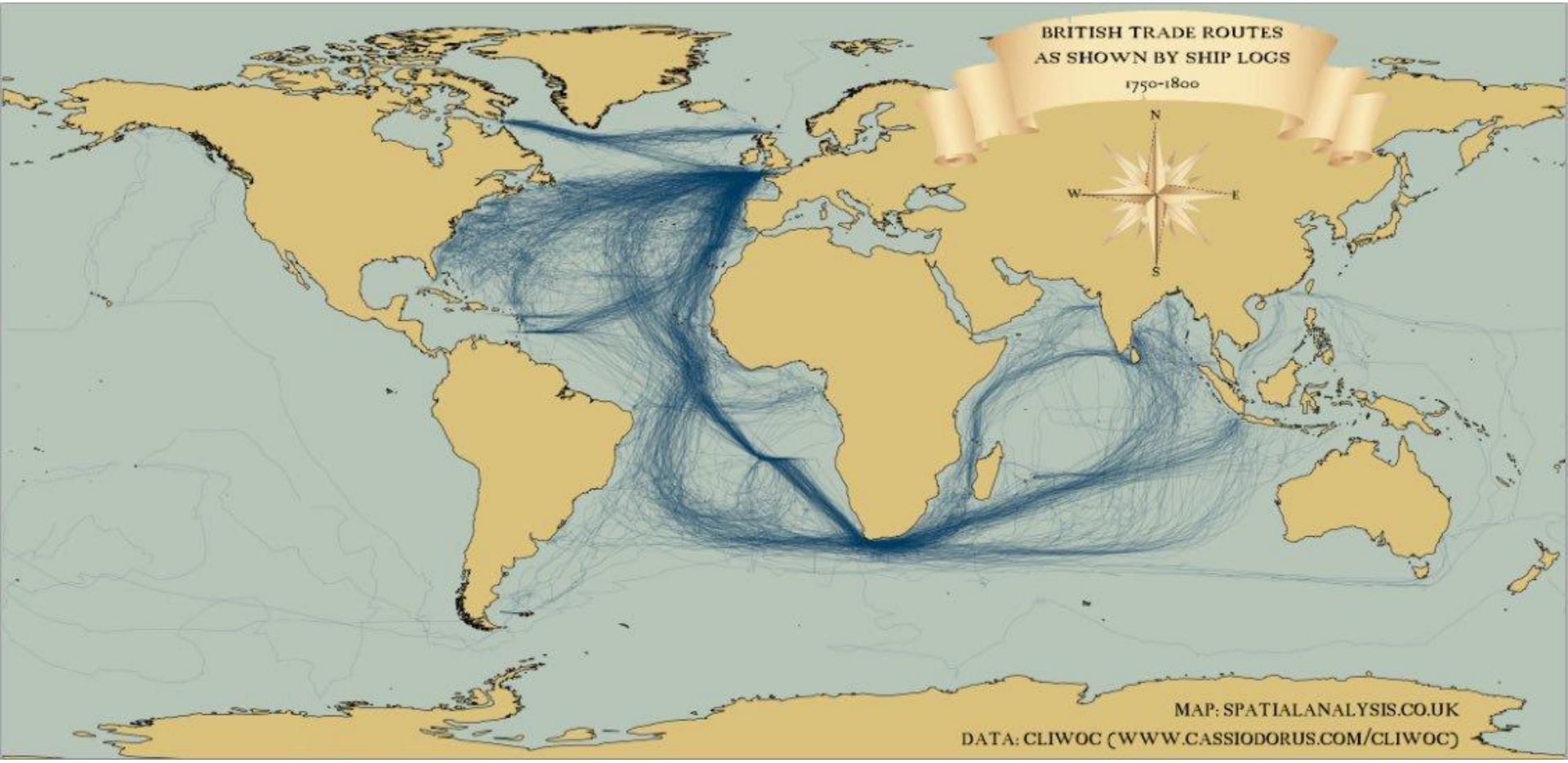


REST: Web Services

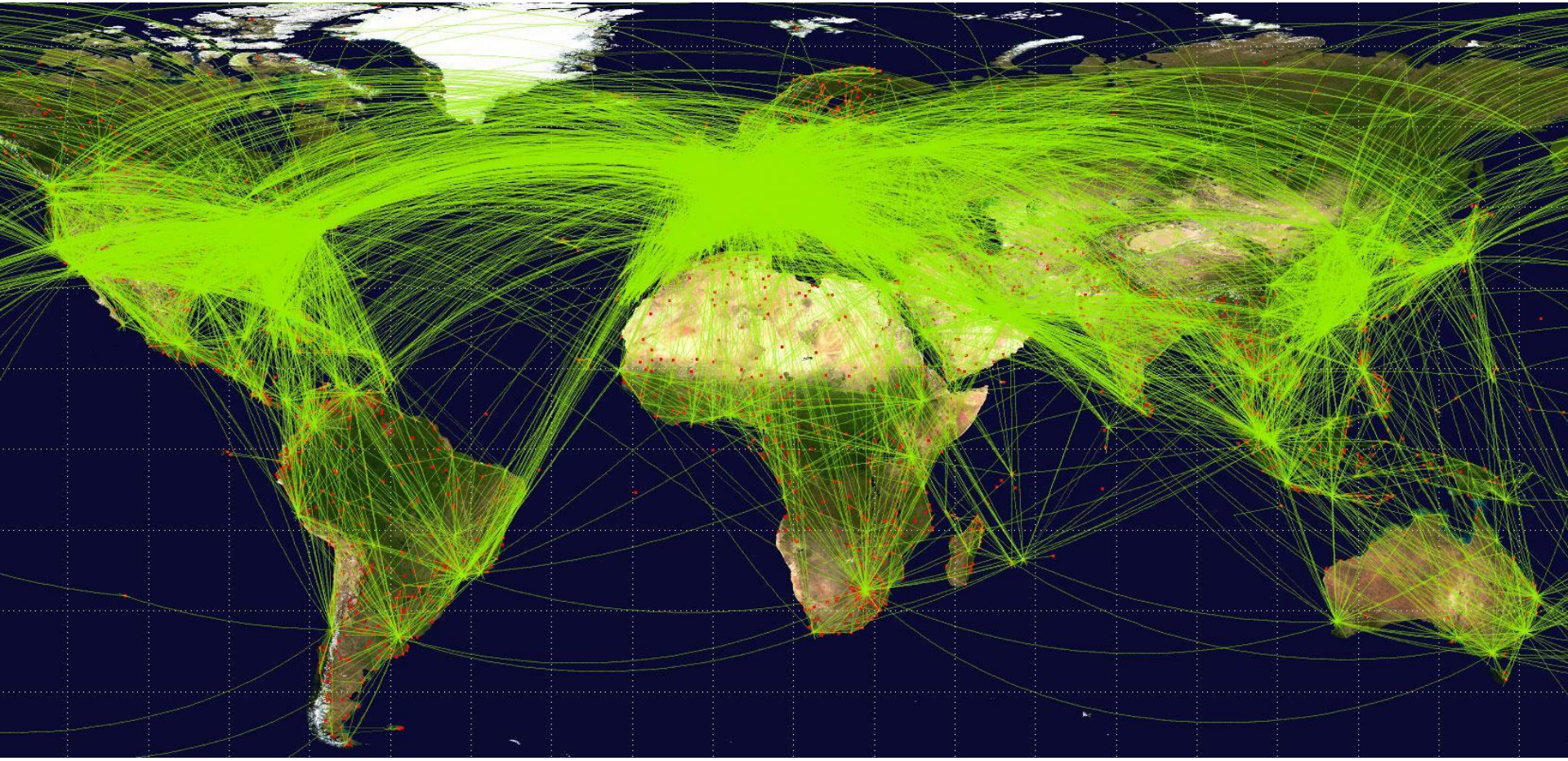
Abel Sanchez

Work Requires Connections



Shipping Routes

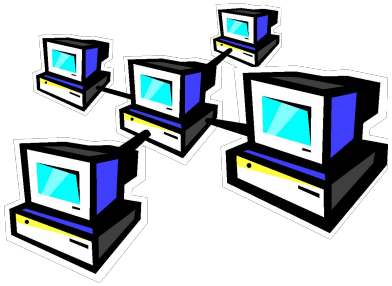
Work Requires Connections



Flight Routes

Connecting

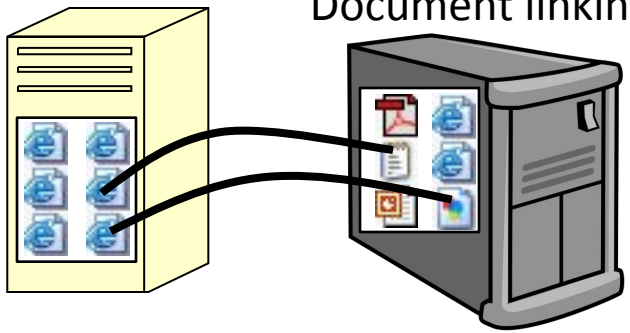
1980s “Internetworking Protocols”



Connecting Computers
The Internet

1995, Web 1.0

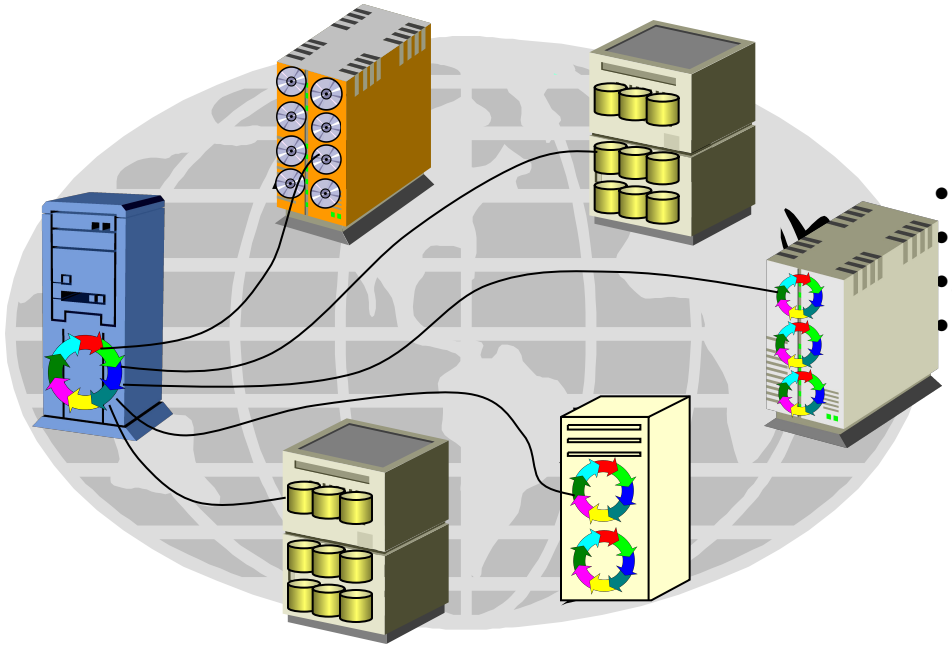
Document linking



1960s Licklider

Man-Computer Symbiosis
The Computer as a Communication Device

2005, Web 2.0

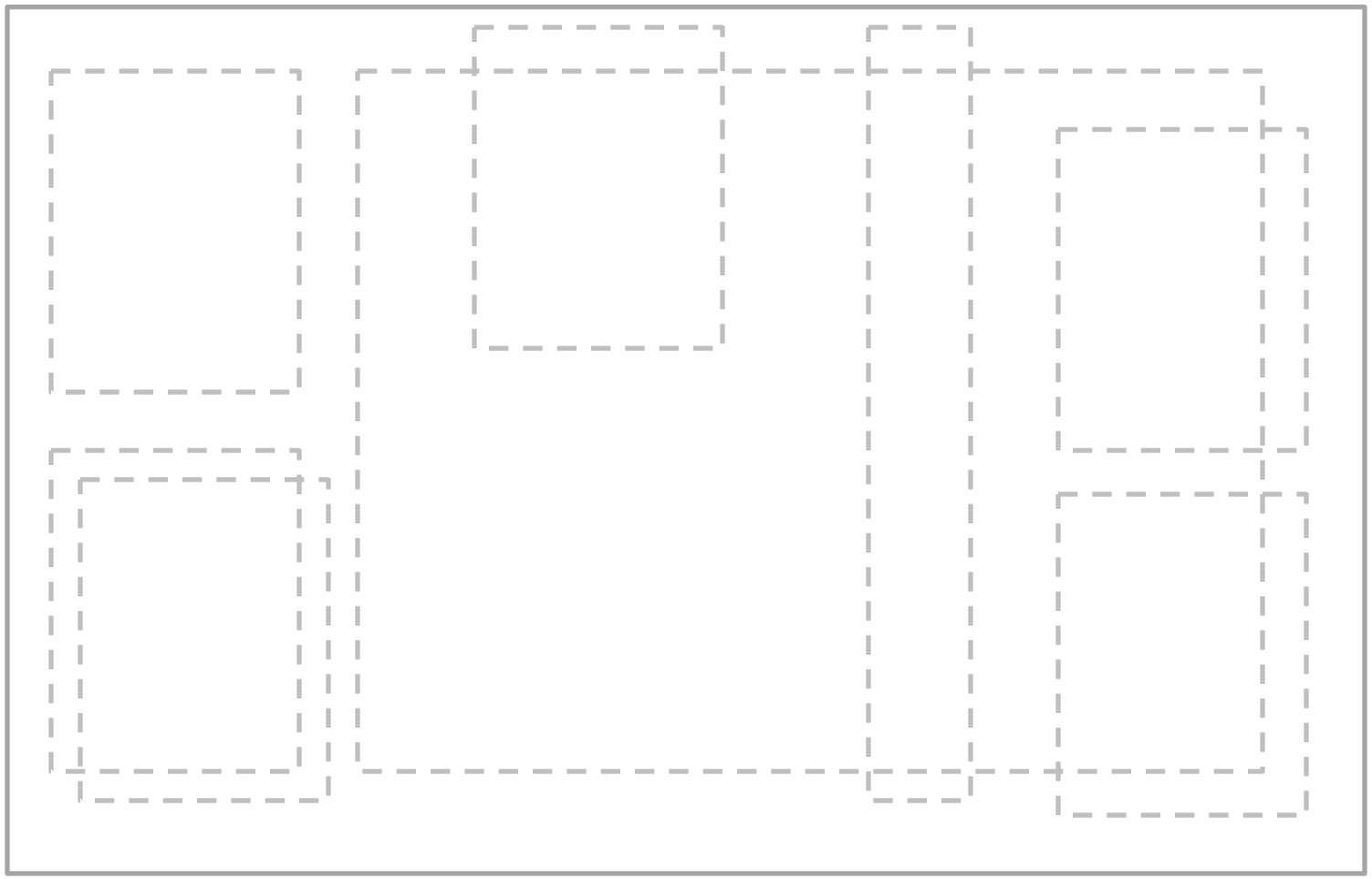


Connecting software/data:

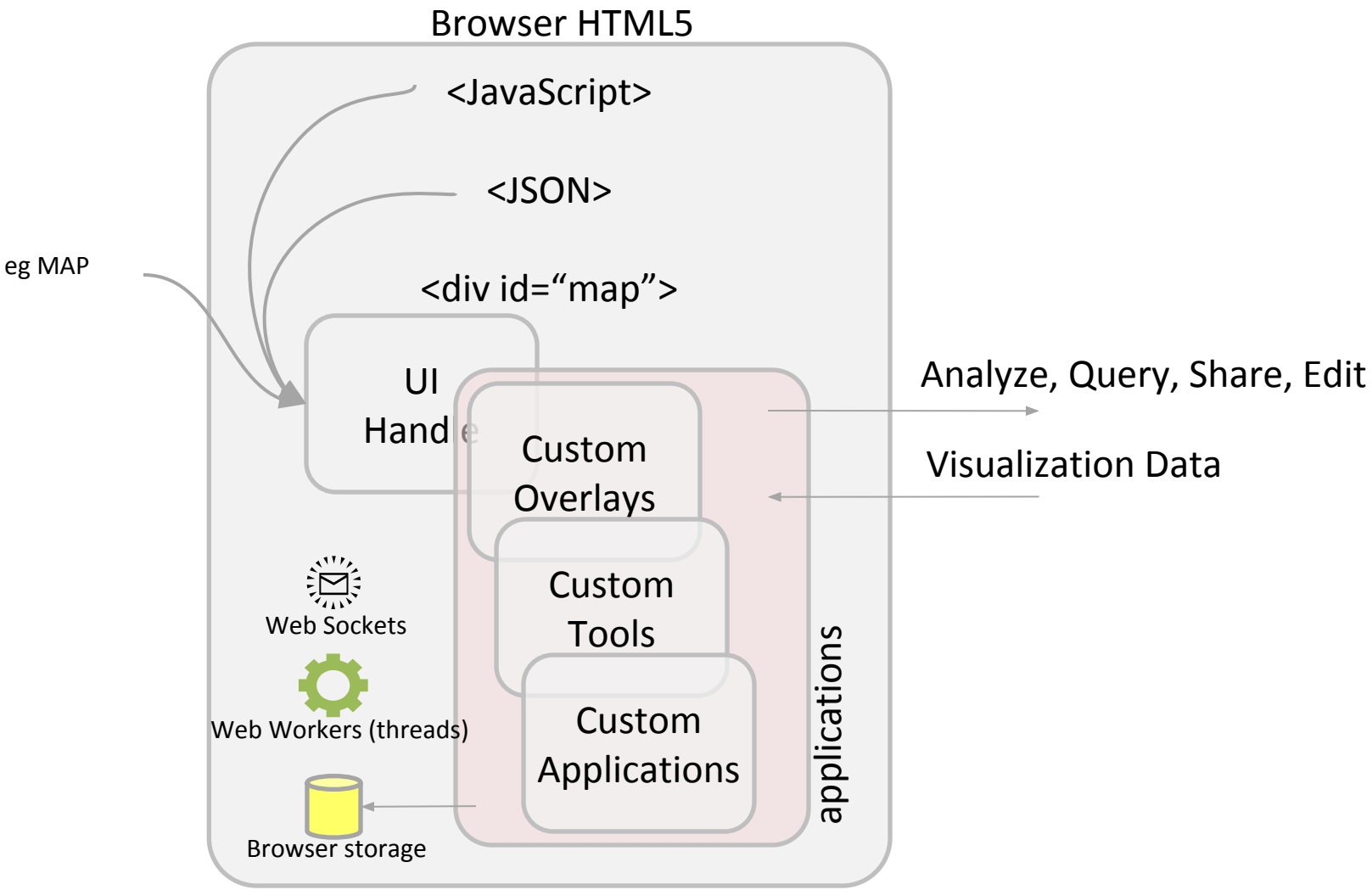
- Databases
- Computational resources
- Simulation and visualization tools
- Number crunching power of computers

MODERN APPLICATION LANDSCAPE

Browser UI Components

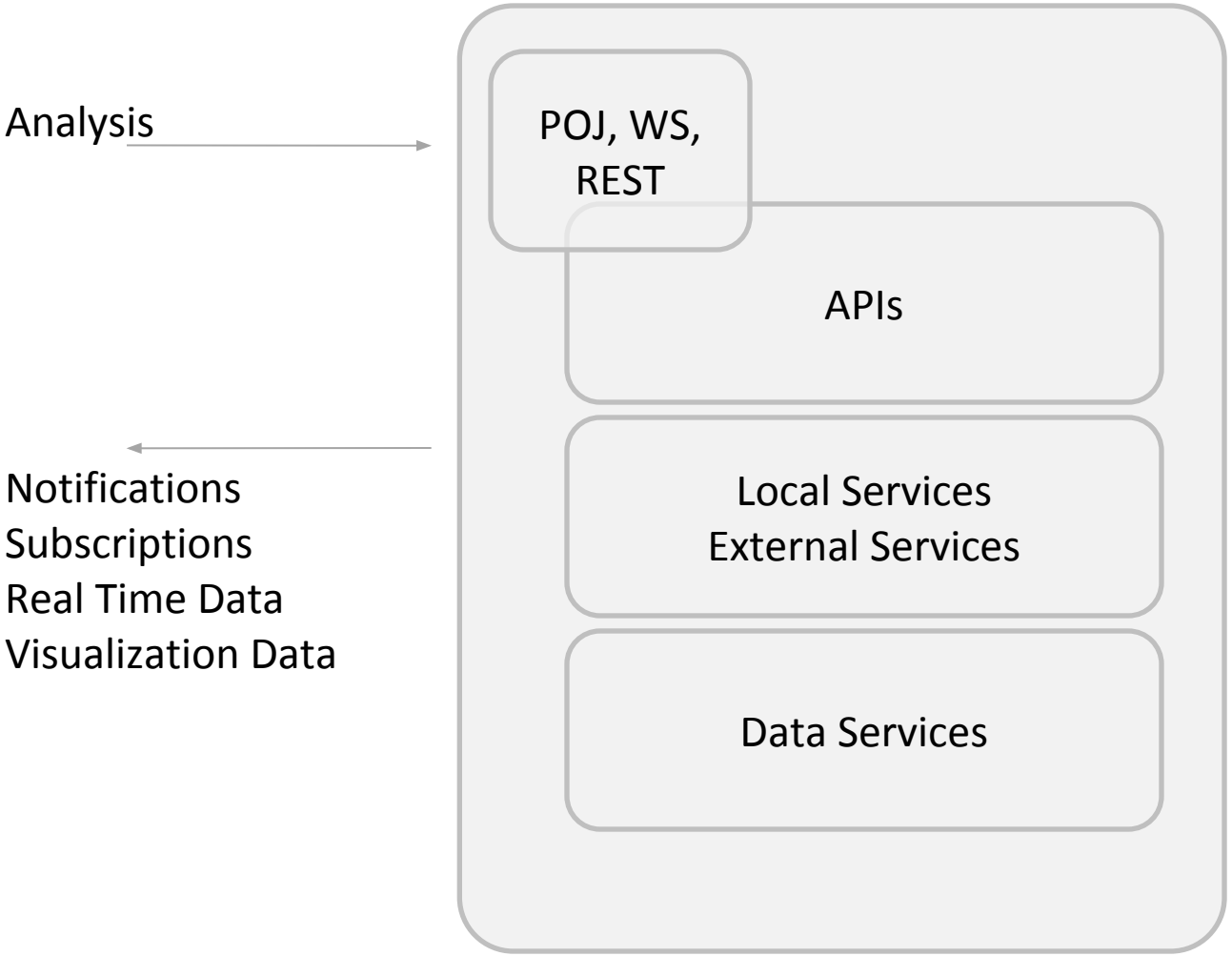


Browser

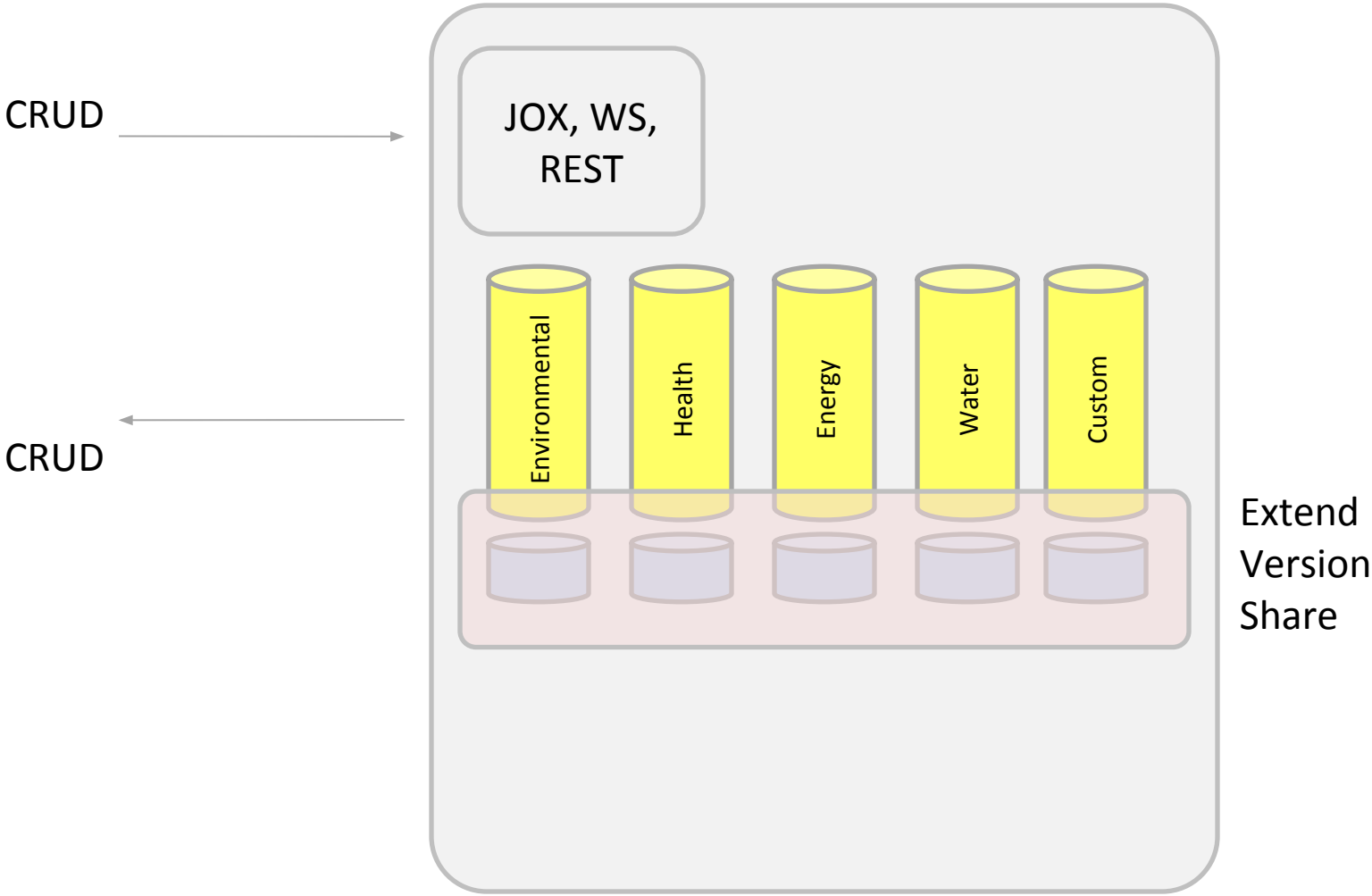

Browser HTML5
<JavaScript>
<JSON>
eg MAP

UI
Handle
Custom
Overlays
Custom
Tools
Custom
Applications
applications
Analyze, Query, Share, Edit
Visualization Data
Web Sockets
Web Workers (threads)
Browser storage

Server


Analysis
POJ, WS,
REST
APIs
Notifications
Subscriptions
Real Time Data
Visualization Data
Local Services
External Services
Data Services

Storage


CRUD
CRUD
JOX, WS,
REST
Environmental
Health
Energy
Water
Custom
Extend
Version
Share

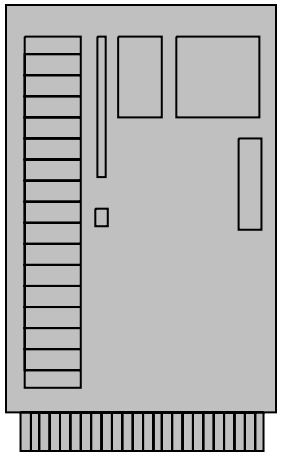
LETS THINK ABOUT DATA

Public Facades?

- Web services and web sites are merely public facades to your internal databases.

Web Sites

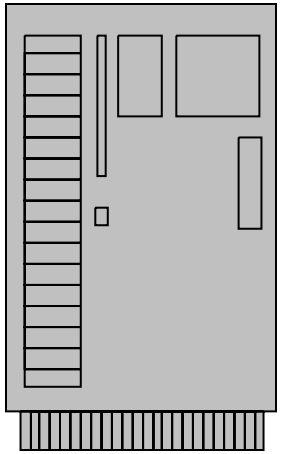
- Web sites expose your database rendered as HTML



HTML
(data + layout)

Web Services

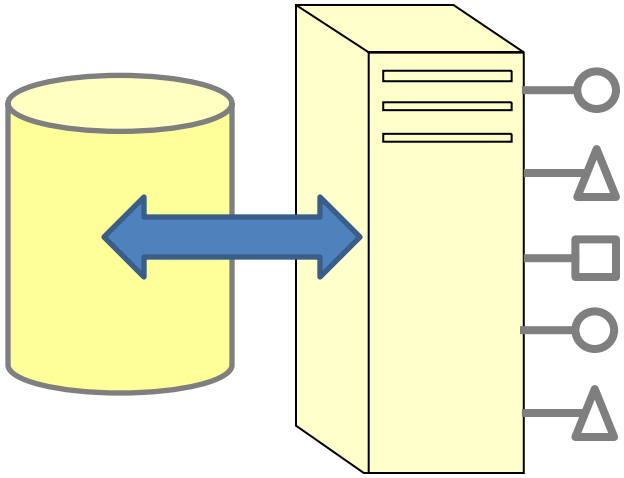
- Web services expose your database in some data format, defined on a case-by-case basis.



Your Data Format
Your Schema

Web Services

- You need to anticipate what bits of information should be exposed and how to expose them



How much do you expose?

What methods do you create?

What data formats do you offer?

What schema do you use?

Do you have the time and can you anticipate every possible use?

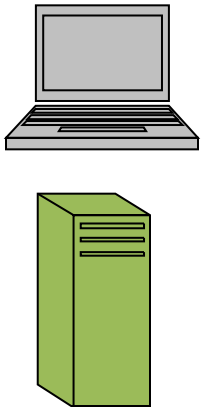
Would be Great to

- Read
- Query
- Edit
- Version
- Share (with friends)
- Share (with programs)
- Discover

ARCHITECTURES

Simple Object Access Protocol (SOAP)

- SOAP

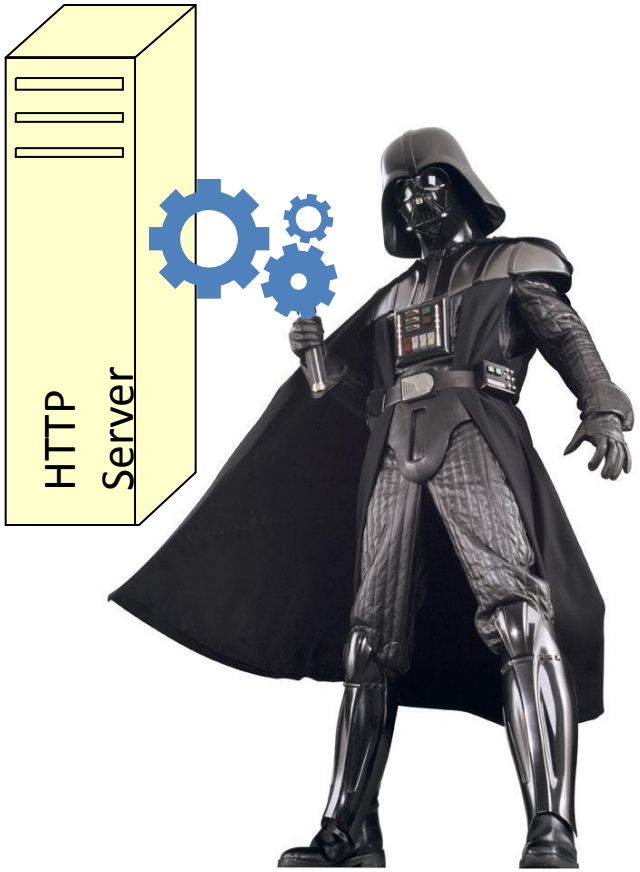


Post Standard SOAP

Response Standard SOAP

```
POST /BareBonesServer/MathService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 319
SOAPAction: "http://LocalHost/BareBonesServer/Add"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <Add xmlns="http://LocalHost/BareBonesServer/">
            <a>100</a>
            <b>300</b>
        </Add>
    </soap:Body>
</soap:Envelope>
```

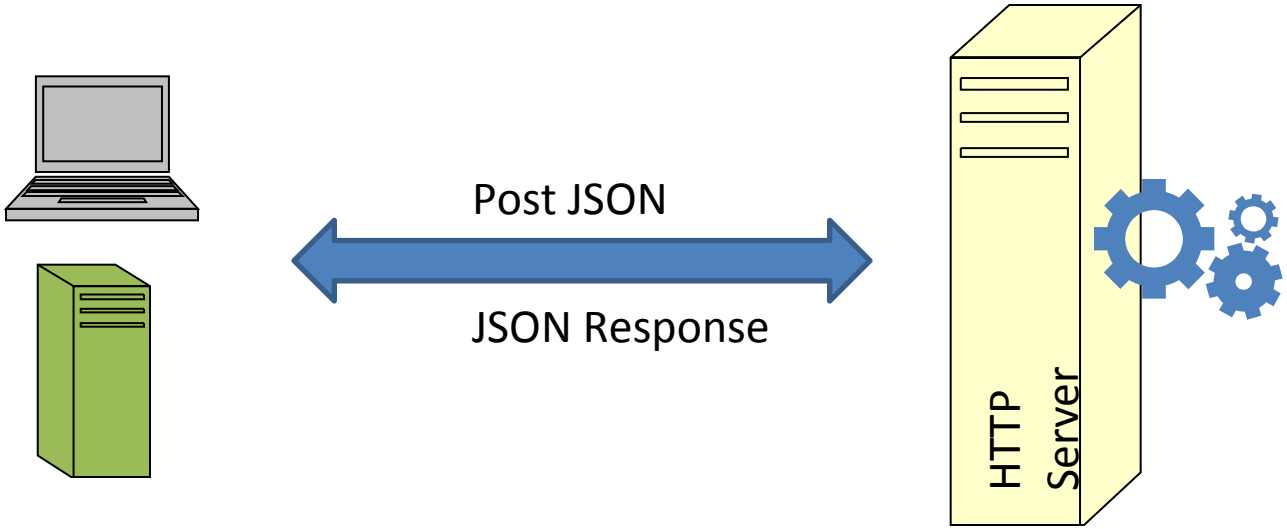


Forget the Standard

- Just use HTTP
- Roll your own

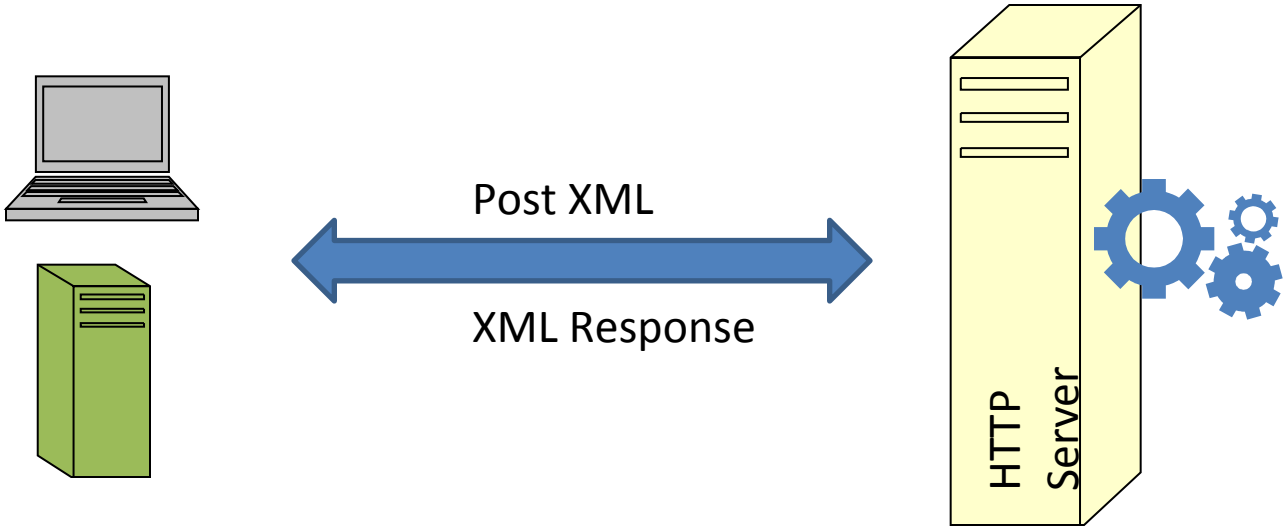
POJ

- Plain Old JSON



POX

- Plain Old XML



REST

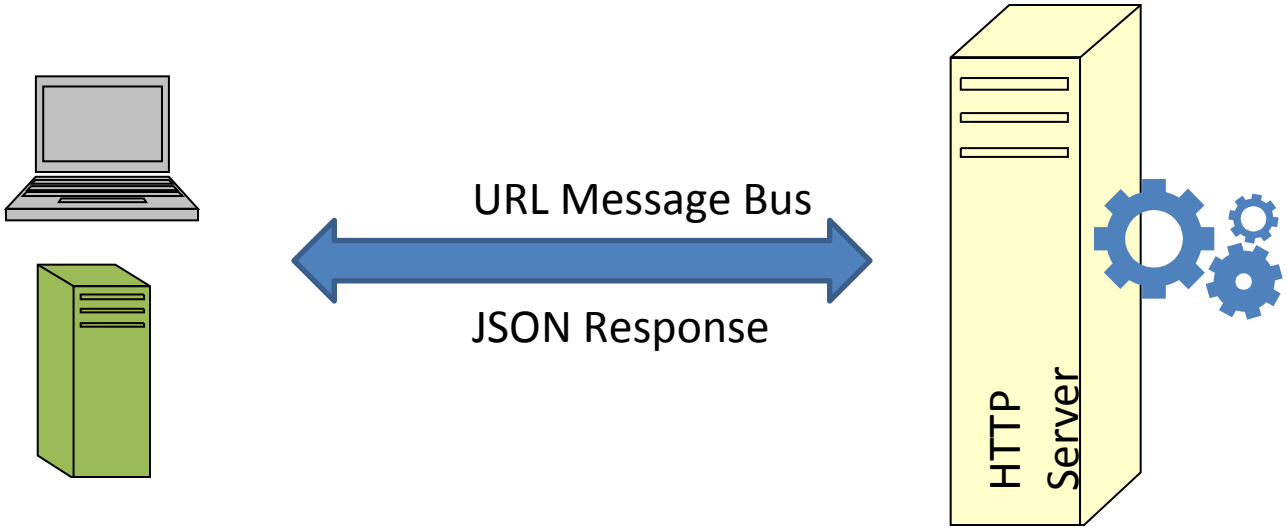
- Representational State Transfer (Roy Fielding)
- An architectural style (no a protocol)
- Web centric
- Simple (a different way of thinking)

REST Principles

- Everything is a **Resource**
- Resources have Names
- Resources are addressable via **URI**s
- Resources are **self descriptive**
  - content types (“application/xml”)
- Resources are **stateless**
- Resources are manipulated via **verbs**

REST

- URL in, JSON out



HTTP Methods

- To create a resource, use POST
- To retrieve a resource, use GET
- To change the state/update, use PUT
- To delete a resource, use DELETE

Directory structure-like URIs

- http://myservice.edu/university/course/{number}
  - http://myservice.edu/mit/1/100

- http://myservice.edu/university/{number}/{semester}/{year}