Web Services

Agenda

- Web Services Overview
- Protocols
  - http
  - xml
  - json
  - api
  - REST
  - SOAP
- **Some code examples**

Protocols

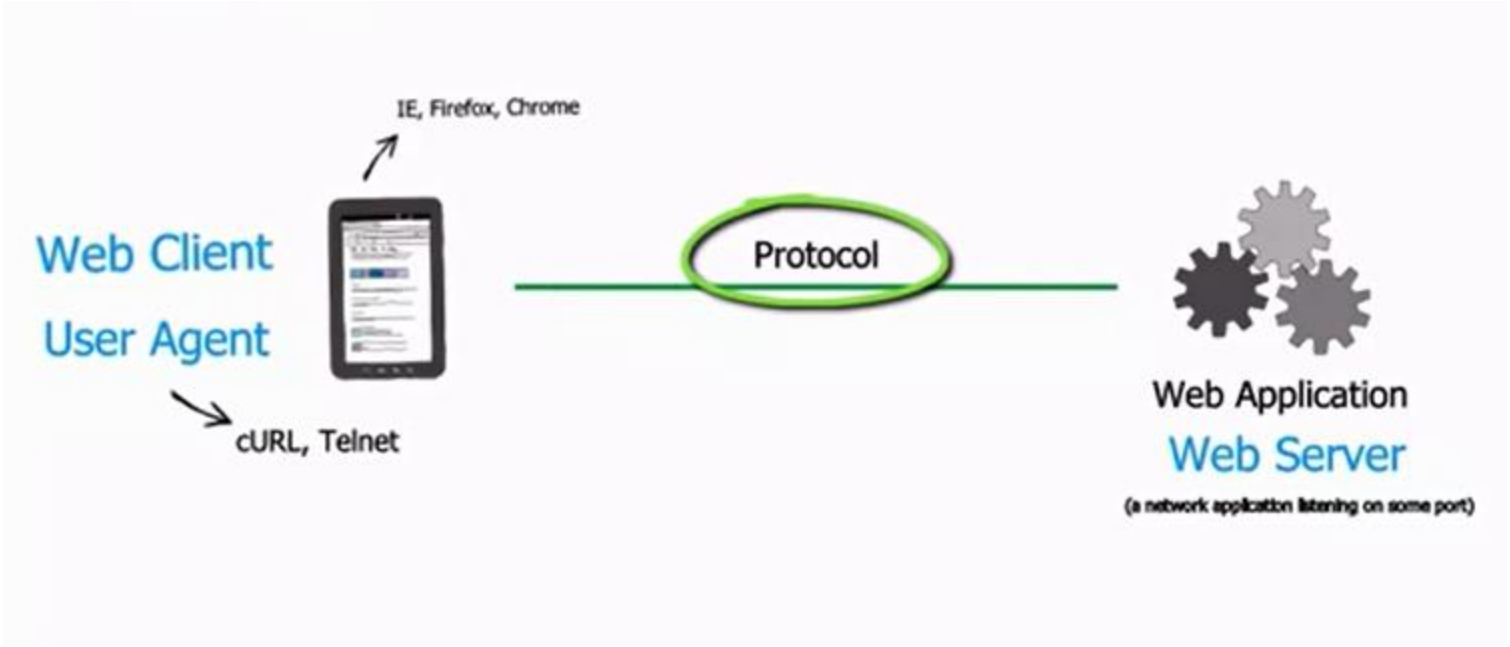
- How do we pass messages and requests from one layer to another?

  PROTOCOLS !

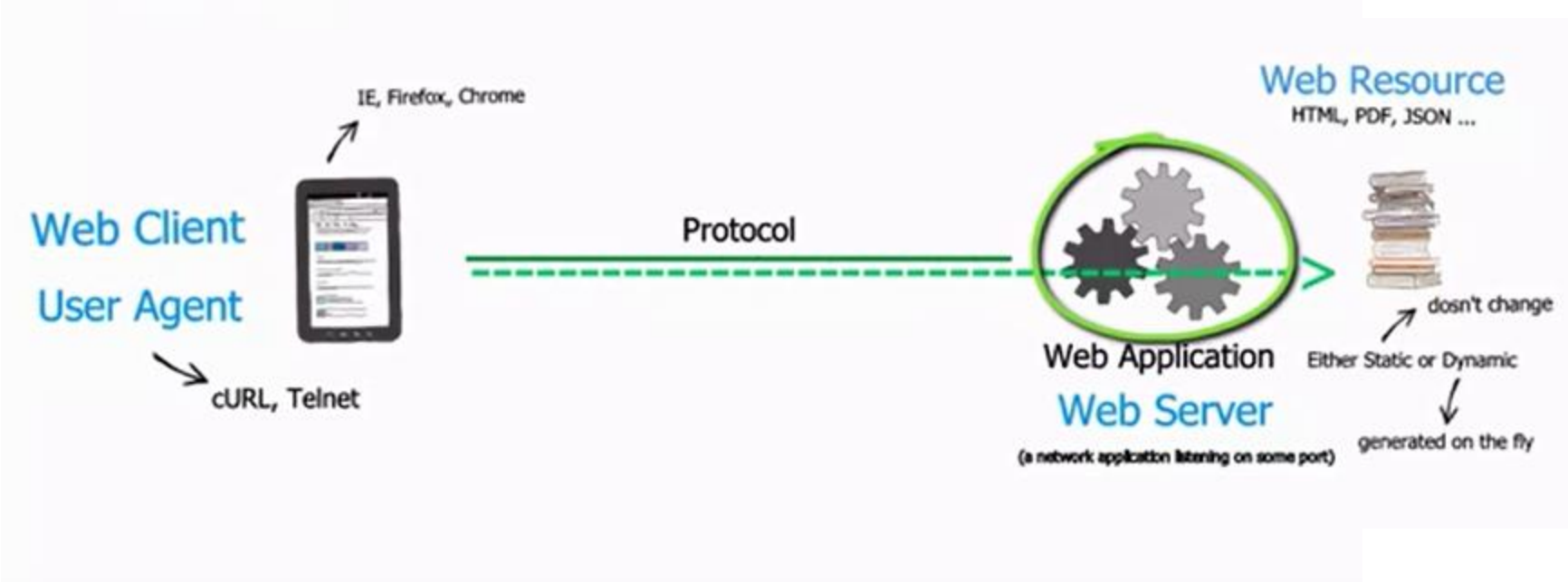
Protocols

- Internet Protocols
- What happens when you type a URL into a browser and press <ENTER>?
- What happens when you click on a hyperlink in a web page?

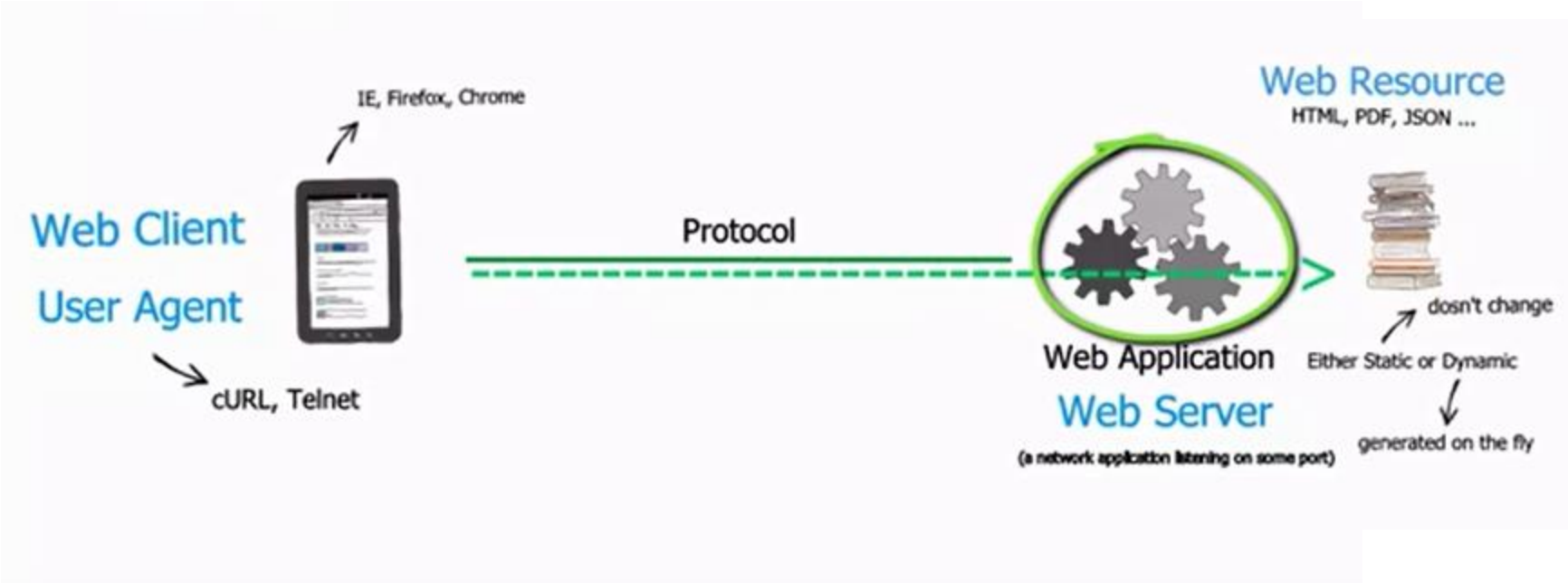
Protocols


IE, Firefox, Chrome
Web Client
User Agent
cURL, Telnet
Protocol
Web Application
Web Server
(a network application listening on some port)

Protocols


IE, Firefox, Chrome
Web Client
User Agent
cURL, Telnet
Protocol
Web Resource
HTML, PDF, JSON ...
dosn't change
Either Static or Dynamic
generated on the fly
Web Application
Web Server
(a network application listening on some port)

Protocols



Each web resource is identified by a URI

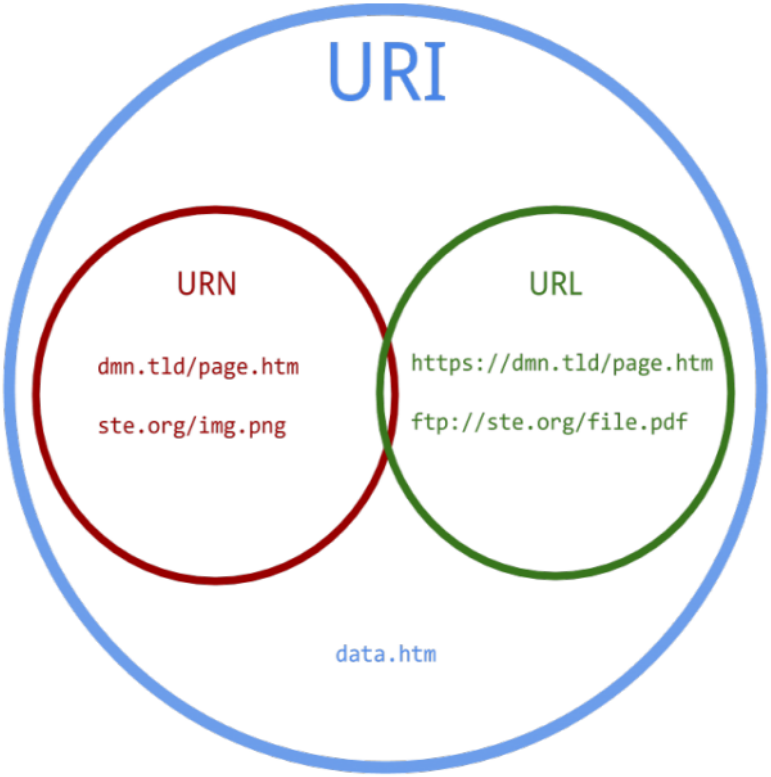
Protocols

- The URI
  - http://www.colorado.edu
  - URI, URL, URN ?

URL = locator
(where/how to find it)

URN = name
(what is its name)

URI = either one



Protocols

- **HTTP** – a request/response protocol
  - It is **STATELESS**
  - The client submits a request, HTTP responds with the requested resource and a return code
    - Resources may be static or dynamic
    - Resources may redirect, include other resources, etc.

- **HTTP Methods**
  - GET Retrieves the URI
  - POST Submits a resource to the URI
    Like submitting a FORM to be processed by a script
  - PUT Stores a resource under the URI
  - DELETE Deletes the URI

Protocols

- **Common HTTP return codes**
- 200 : OK
- 302 : Redirect
- 400 : Bad Request
- 401 : Unauthorized
- 403 : Forbidden
- 404 : Not Found
- 500 : Server Error

Protocols

- **Passing data to/from the web server**
- XML Extensible Markup Language
- JSON Java Script Object Notation

XML

XML Extensible Markup Language

- “Tag” based, like HTML
- Tags are user-defined
- XML is human readable AND machine readable
- Tags describe the data (XML tags do NOT display the data like HTML tags do)
- You can use a programming language like javascript or php or python to read, parse, modify and write XML documents sent/received to/from a Web Service
- The XML document structure is defined by a DOM – Document Object Model

```
<bookstore>
   <book category="cooking">
      <title lang="en">Everyday Italian</title>
      <author>Giada De Laurentiis</author>
      <year>2005</year>
      <price>30.00</price> </book>
   <book category="children">
      <title lang="en">Harry Potter</title>
      <author>J K. Rowling</author>
      <year>2005</year>
      <price>29.99</price> </book>
   <book category="web">
      <title lang="en">XQuery Kick Start</title>
      <author>James McGovern</author>
      <author>Per Bothner</author>
      <author>Kurt Cagle</author>
      <author>James Linn</author>
      <author>Vaidyanathan Nagarajan</author>
      <year>2003</year>
      <price>49.99</price> </book>
   <book category="web" cover="paperback">
      <title lang="en">Learning XML</title>
      <author>Erik T. Ray</author>
      <year>2003</year>
      <price>39.95</price> </book>
</bookstore>
```

JSON

- Represents data in key:value pair format.
- Many think JSON is easier to use than XML
- More compact than XML
- Like XML, JSON is easy for both humans & computers to understand

JSON:

```
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
] }
```

XML:

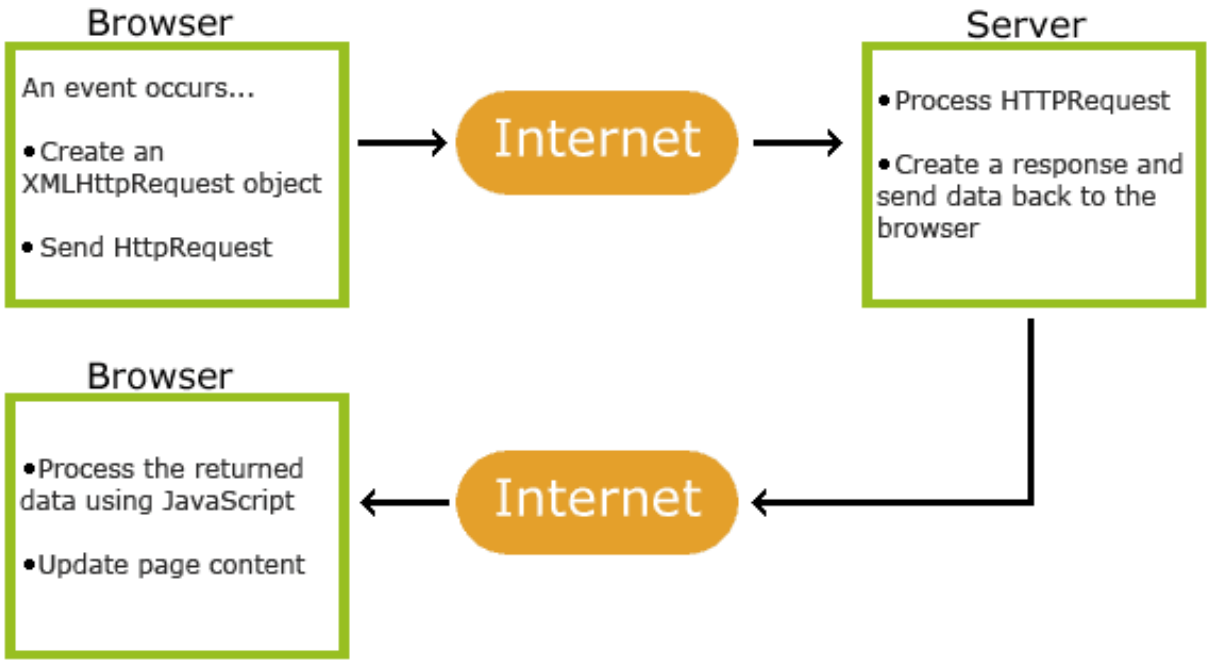
```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

JSON is Like XML Because

- Both JSON and XML are "self describing" (human readable)
- Both JSON and XML are hierarchical (values nested within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an HttpRequest

JSON is Unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays



1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

JSON

For AJAX applications, JSON is faster and easier than XML:

Using XML

- Fetch an XML document
- Use the XML DOM to loop through the document
- Extract values and store in variables

Using JSON

- Fetch a JSON string
- JSON.Parse the JSON string

AJAX = Asynchronous JavaScript And XML

“AJAX is a developer's dream, because you can:

- **Update a web page without reloading the page**
- **Request data from a server - after the page has loaded**
- **Receive data from a server - after the page has loaded**
- **Send data to a server - in the background”**

(quote from w3schools.com)

AJAX is a technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display. (JavaScript runs on the CLIENT)

Example AJAX versus CGI (Common Gateway Interface)

https://www.tutorialspoint.com/ajax/ajax_examples.htm

Web Services

Early Web (CGI) 1989

- hypertext / hyperlinks
- page by page

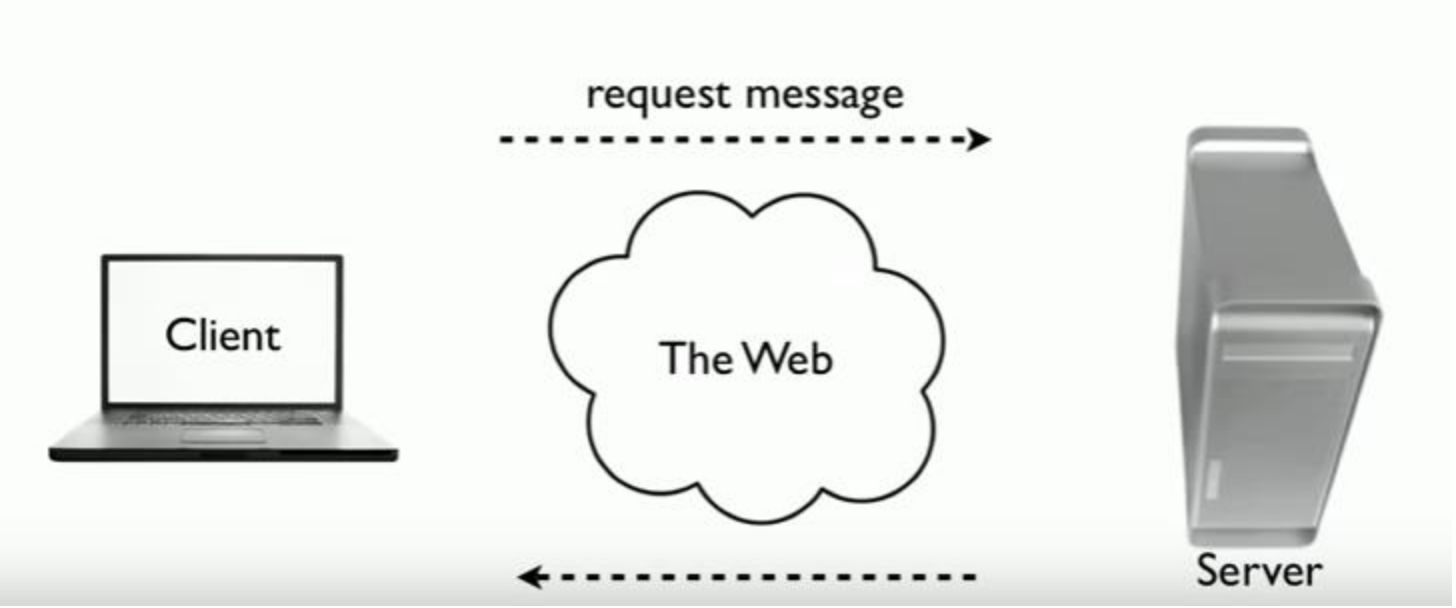
Web 2.0 (AJAX) 2004

- web page stays in place
- parts of the web page are updated

How are web 2.0 requests handled between client and server?

Web Services !

A framework for a conversation between computers over the web



Web Services

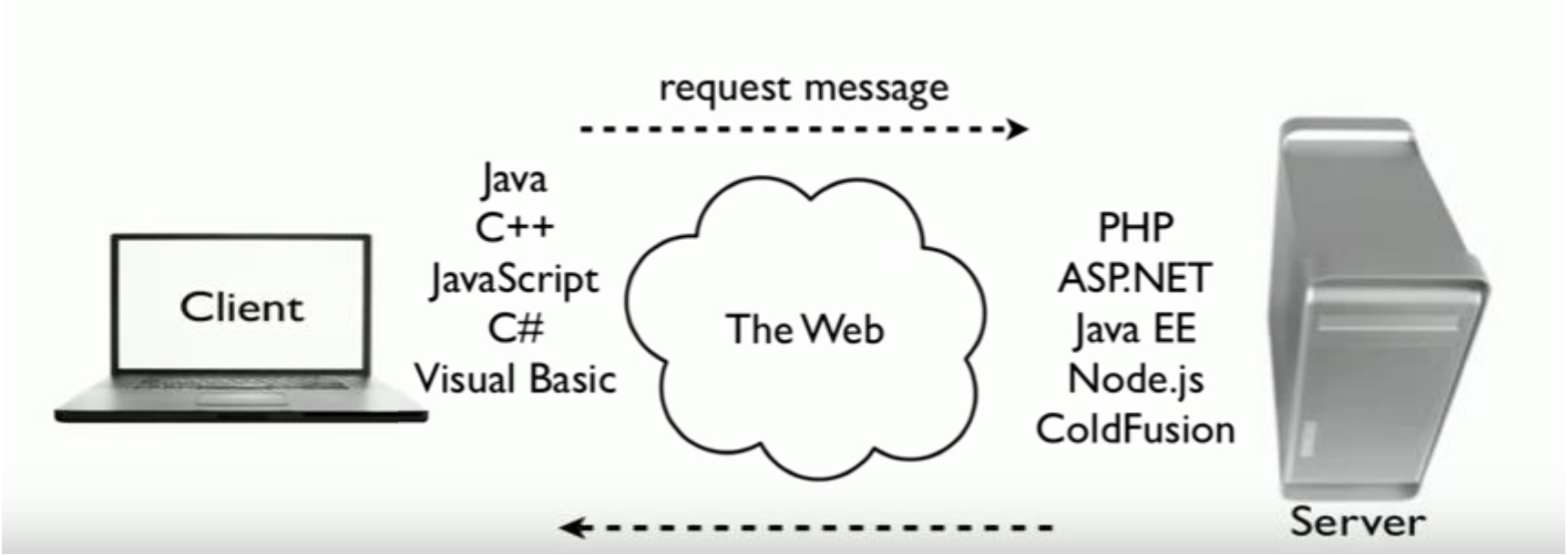
If you want to use a web service, you must use an API (application programming interface)

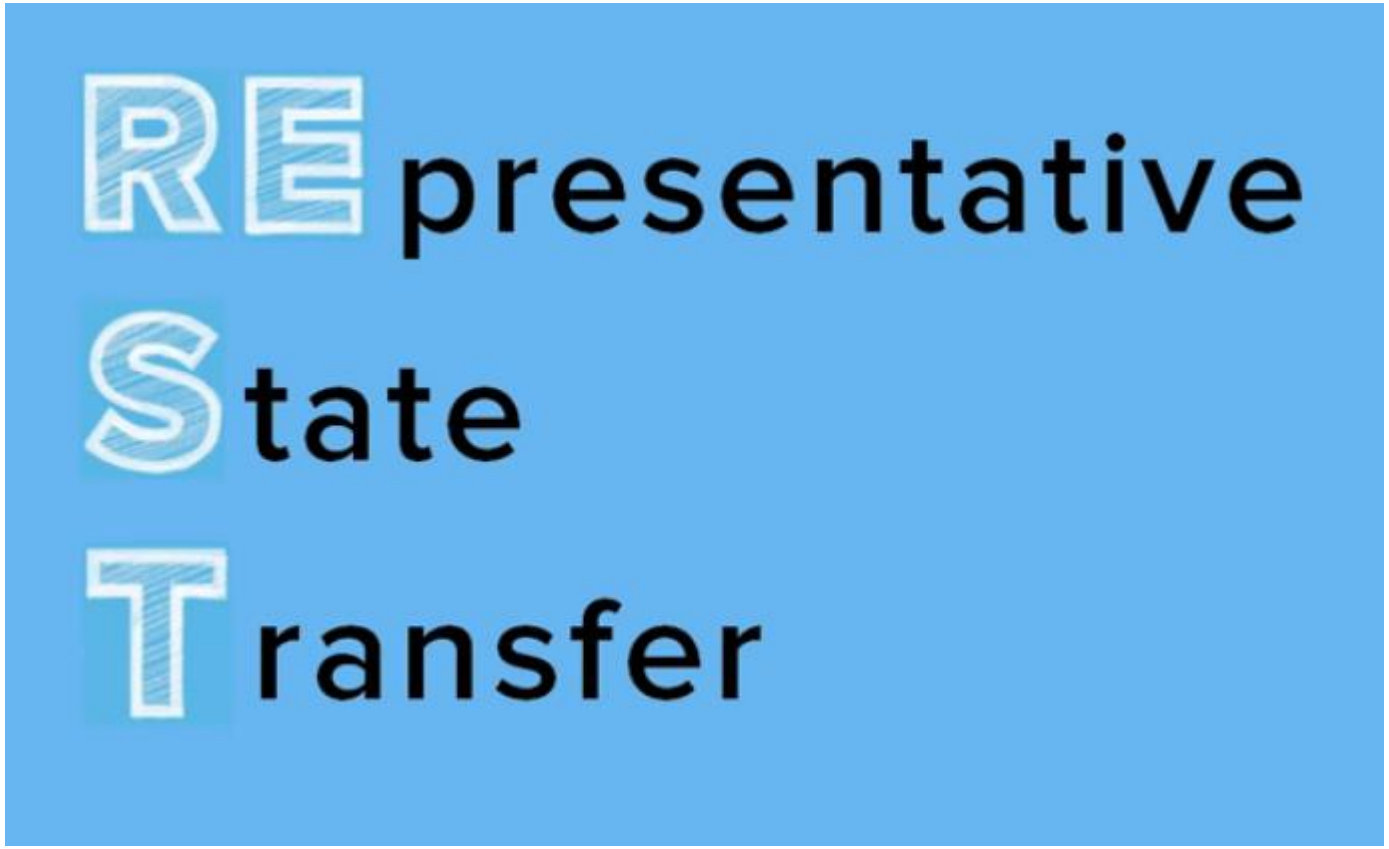
Defines everything you need to know to talk to a web service:

1. **Message format: SOAP, XML, JSON, etc.**
2. **Request syntax: URI, Parameters & Data types**
3. **Actions on the server: named methods, HTTP verbs**
4. **Security: authentication (username & password)**
5. **Response format: SOAP, XML, JSON, etc.**

The web service hides its complexity behind the API

The web service hides its complexity behind the API




REpresentative
State
Transfer

REST
is an *architectural style*

Modern Architectural Style:

Colonial Architectural Style:

The "architectural style" is an abstract concept
it defines the characteristics and features you would find in a house built according to that style

It is NOT the same as the house itself.

REST is an abstract concept that defines the characteristics and features you would find in a web service request built according to the REST style

REST is not really a protocol – it is a set of standards used to define Web Services

Web Services

- Everything in REST is considered as a resource.
  - Every resource is identified by an URI.
  - Uses uniform interfaces. Resources are handled using http POST, GET, PUT, DELETE operations
  - Stateless. Every request is an independent request. Each request from client to server must contain all the information necessary to understand the request.

Web Services

- RESTFul web services are based on HTTP methods
- a RESTFul web service typically defines the base URI for the services, the format/rules of the API, and the set of operations (POST, GET, PUT, DELETE) which are supported.

Characteristics of a request/response following the REST style

Resources follow the rules

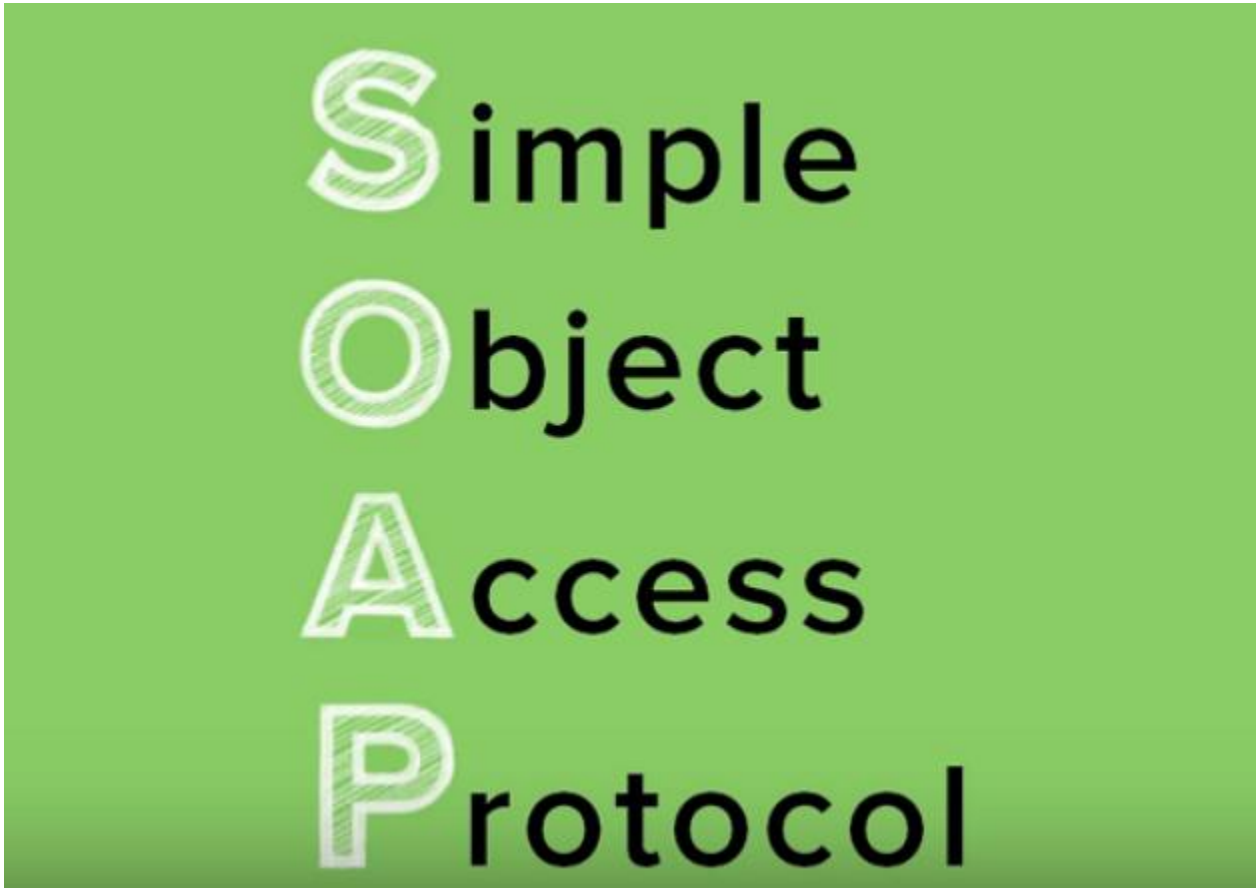
- **URI (**identifies the resource being requested**)**
- **Uniform Interface Methods (**GET, PUT, POST, etc.**)**
- **Uniform Interface Representation (**XML, JSON, HTML**)**

Protocols offer features

- **Client-Server (**like HTTP**)**
- **Stateless (**each request is independent**)**
- **Layered (**may pass through intermediaries**)**
- **Cacheable (**intermediaries may cache for performance**)**

Advantages of a request/response following the REST protocol

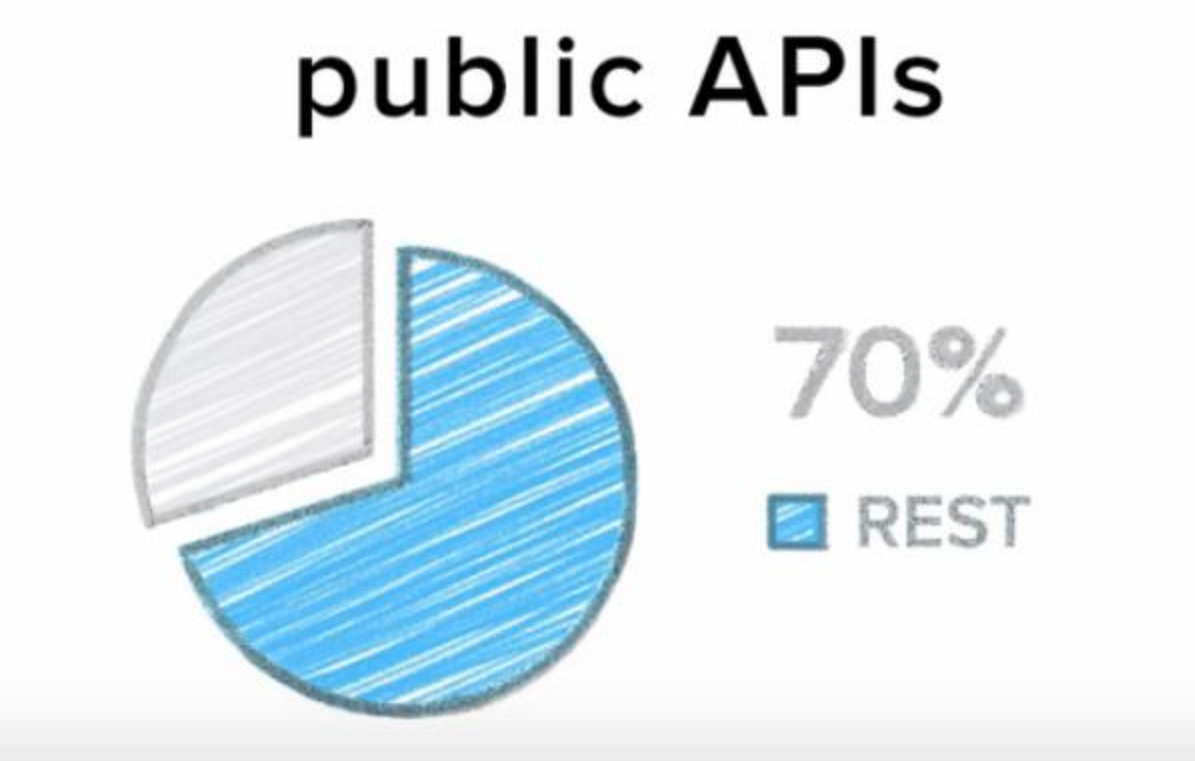
- **Efficiency**
  (through caching & compression)
- **Scalability**
  (gateways distribute traffic, caching, statelessness allows different intermediaries)
- **User Perceived Performance**
  (code on demand, client validation, caching)
- **Simplicity**


Simple
Object
Access
Protocol

SOAP -- Simple Object Access Protocol

REST	SOAP
Representational State Transfer	Simple Object Access Protocol
Architecture Style	An actual protocol
Uses simple HTTP	Uses SOAP envelope, then HTTP (or FTP, or other) to transfer the data
Uses many different data formats like JSON, XML, YAML*	Supports only XML format
Performance & Scalability & Caching	Slower performance. Scalability is limited and complex. Caching is not possible.
Widely and frequently used	Used where REST is not possible

*YAML: YAML Ain't Markup Language
What It Is: YAML is a human friendly data serialization standard for all programming languages.


public APIs
70%
REST

Web Services

- WSDL (Web Service Description Language) is an XML document that defines “contract” between client and service and is static by its nature.

- SOAP builds an XML based protocol on top of HTTP or some other protocol according to the rules described in the WSDL for that Web Service.

SOAP

- A SOAP message is an XML document containing the following elements:
  - An **Envelope** element that identifies the XML document as a SOAP message
  - A **Header** element that contains header information
  - A **Body** element that contains call and response information
  - A **Fault** element containing errors and status information

Web Services

http://www.w3schools.com/graphics/google_maps_basic.asp

Demo of an API for using a web service

```
center:new google.maps.LatLng(40.0150,-105.2705),
zoom:10,
```

http://apigee.com/providers/

A list of published web services and APIs

https://www.youtube.com/watch?v=7YcW25PHnAA

– This video shows a clear example of how we can use REST framework API for using a web service