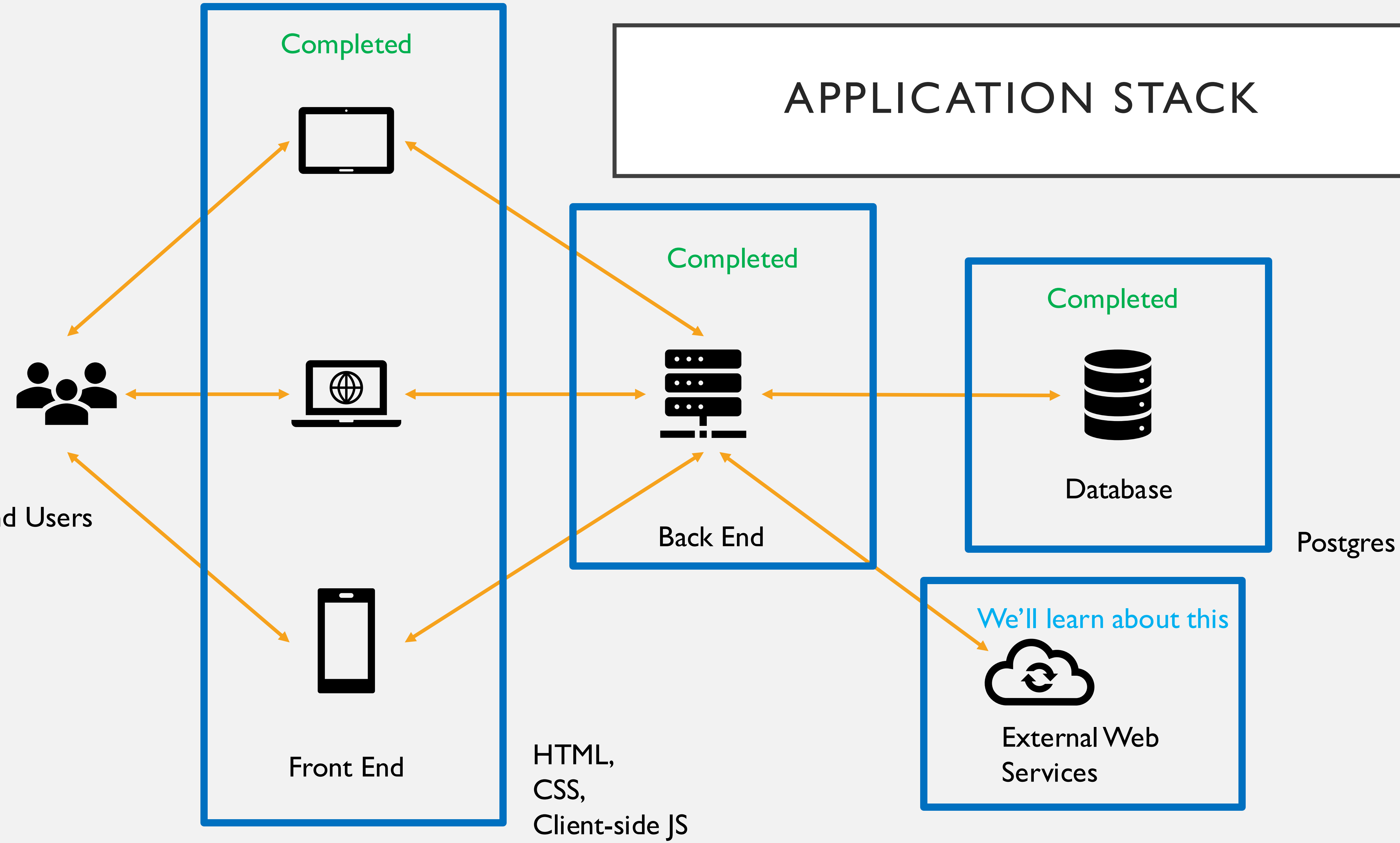
WEB SERVICES

Mar 3, 2024

AGENDA

- Internet Protocols
  - Identifying resources on the web
  - HTTP
- Passing data over HTTP
  - XML, JSON
- AJAX
- WEB SERVICES
- REST
- SOAP

APPLICATION STACK

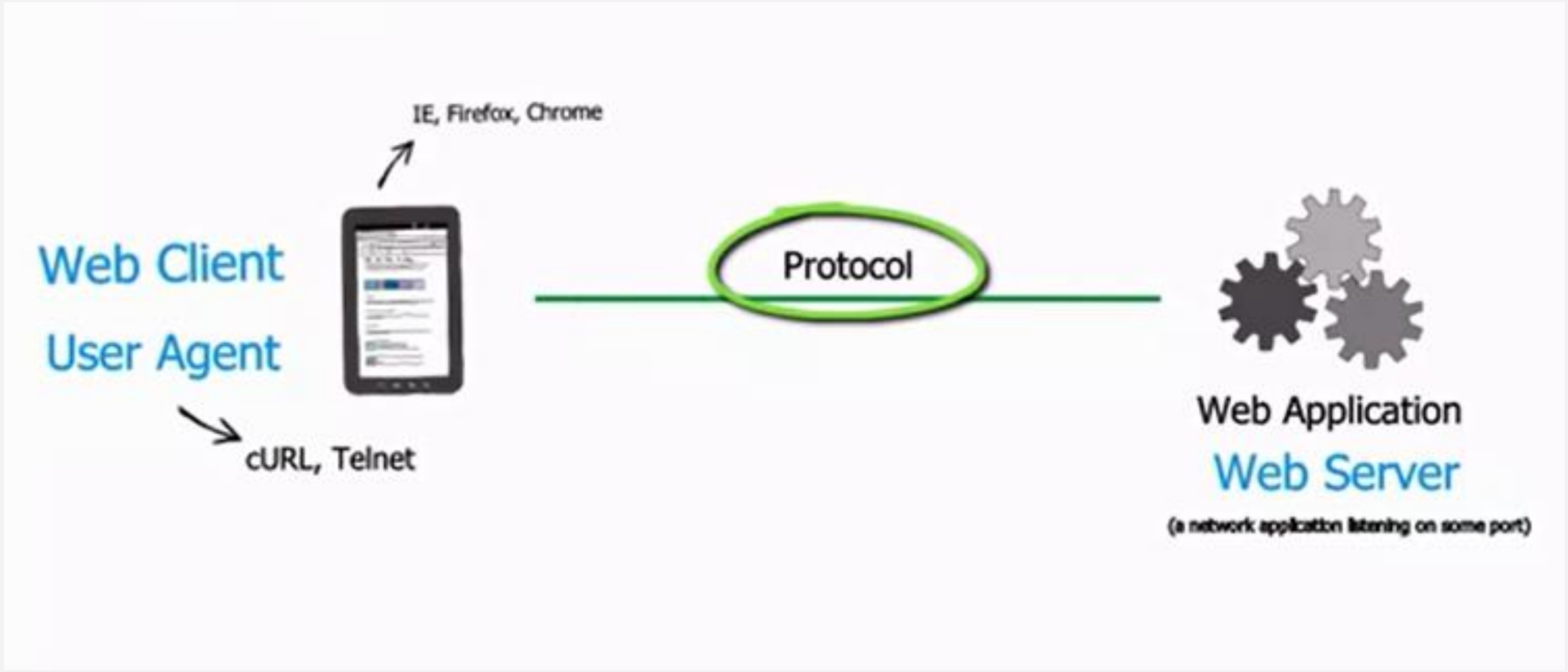


INTERNET PROTOCOLS

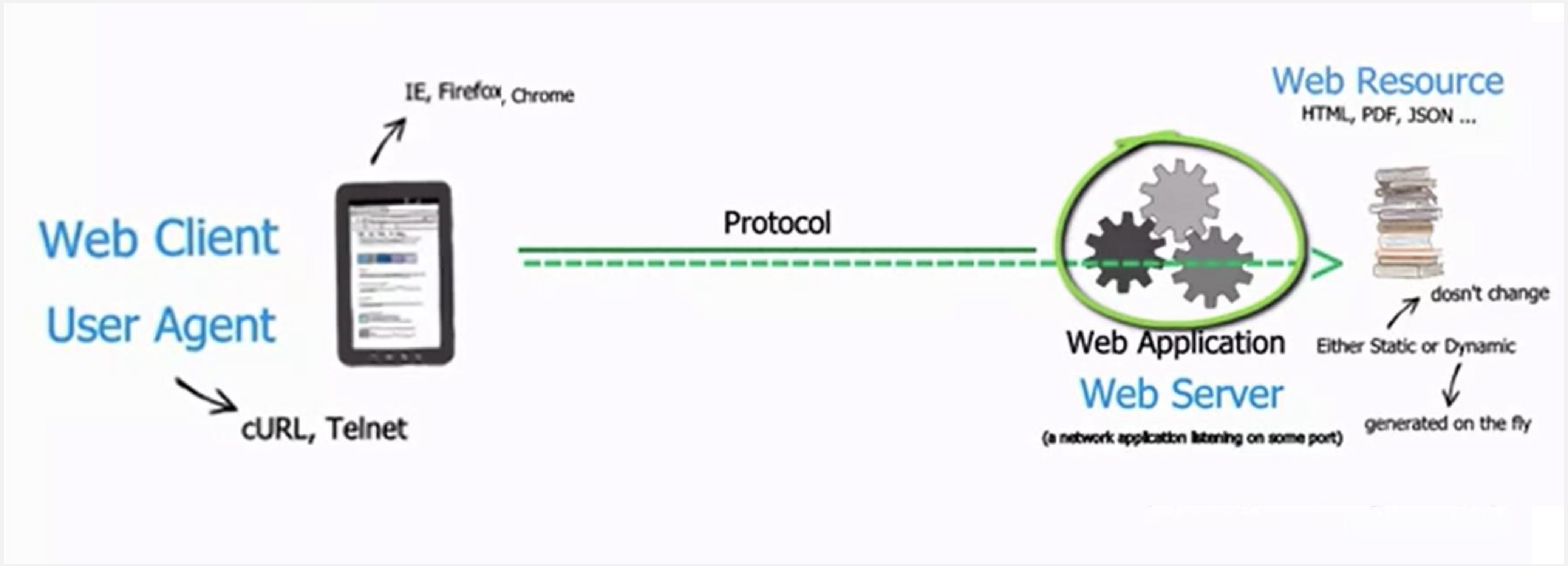
- What happens when you type a URL into a browser and press <ENTER>?
- What happens when you click on a hyperlink in a web page?
- How do we pass messages and requests from one layer to another?

PROTOCOLS !

PROTOCOLS


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

PROTOCOLS

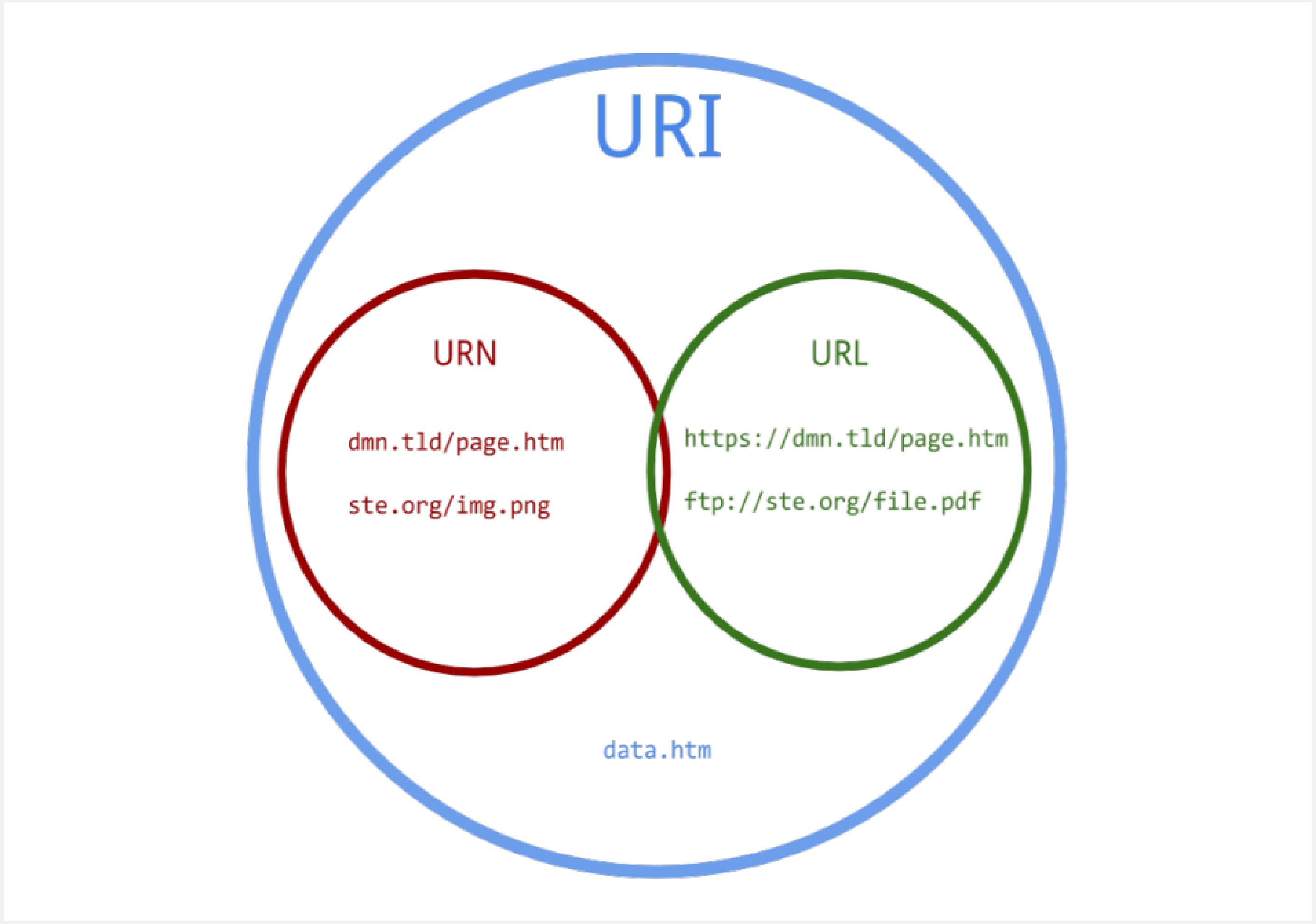


Each web resource is identified by a URI

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



HTTP – HYPERTEXT TRANSFER PROTOCOL

- HTTP – a request/response protocol
- It is **STATELESS**
  - A stateless protocol does not **require** the HTTP server to retain information or status about each user for the duration of multiple requests.
- 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
  - Requests using GET should only retrieve data and should have no other effect.
- **POST** - Submits a resource to the URI
  - The POST method requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI.
- **PUT** - Stores a resource under the URI
  - The PUT method requests that the enclosed entity be stored under the supplied URI.
- **DELETE** - Deletes the URI
  - The DELETE method deletes the specified resource.

COMMON HTTP RETURN CODES

200 : OK	Standard response for successful HTTP requests.
302 : Redirect	Tells the client to temporarily look at (browse to) another URL.
400 : Bad Request	The server cannot or will not process the request due to an apparent client error
401 : Unauthorized	The server cannot or will not process the request because the client is not authenticated
403 : Forbidden	The request contained valid data and was understood by the server, but the server is refusing action because client does not have authorization
404 : Not Found	The requested resource could not be found but may be available in the future.
500 : Server Error	A generic error message, given when an unexpected condition was encountered, and no more specific message is suitable.

PASSING DATA TO/FROM THE WEB SERVER

XML - eXtensible Markup Language

JSON - JavaScript Object Notation

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

XML

```
<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>
        <year>2003</year>
        <price>49.99</price>
    </book>
</bookstore>
```

JSON

"Java Script Object Notation"

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

```
"bookstore": {
             "book": [
                   {"title": "Everyday Italian",
                   “author": "Giada De Laurentiis",
                   "year": "2005",
                   "price": "30.00"},
                   {"title": "Harry Potter",
                   "author": "J K. Rowling",
                   "year": "2005",
                   "price": "29.99"},
                   {"title": "XQuery Kick Start",
                   "author": ["James McGovern",
                         "Per Bothner",
                         "Kurt Cagle",
                         "James Linn"],
                   "year": "2003",
                   "price": "49.99"}
                   ]
}
```

JSON VS. XML

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 VS. XML

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 HTTP Request

JSON is Unlike XML Because

- JSON doesn't use end tags
- JSON is shorter
- JSON is quicker to read and write
- JSON can use array

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

OK. So, what is AJAX?

- AJAX is a technique for creating better, faster, and more interactive web applications.
- AJAX uses:
  - **XHTML** for content
  - **CSS** for presentation
  - **DOM** and **JavaScript** for dynamic content display. (JavaScript runs on the client-end)

WEB SERVICES - HISTORY

Early Web (CGI) 1989

- hypertext / hyperlinks
- page by page

Web 2.0 (AJAX) 2004

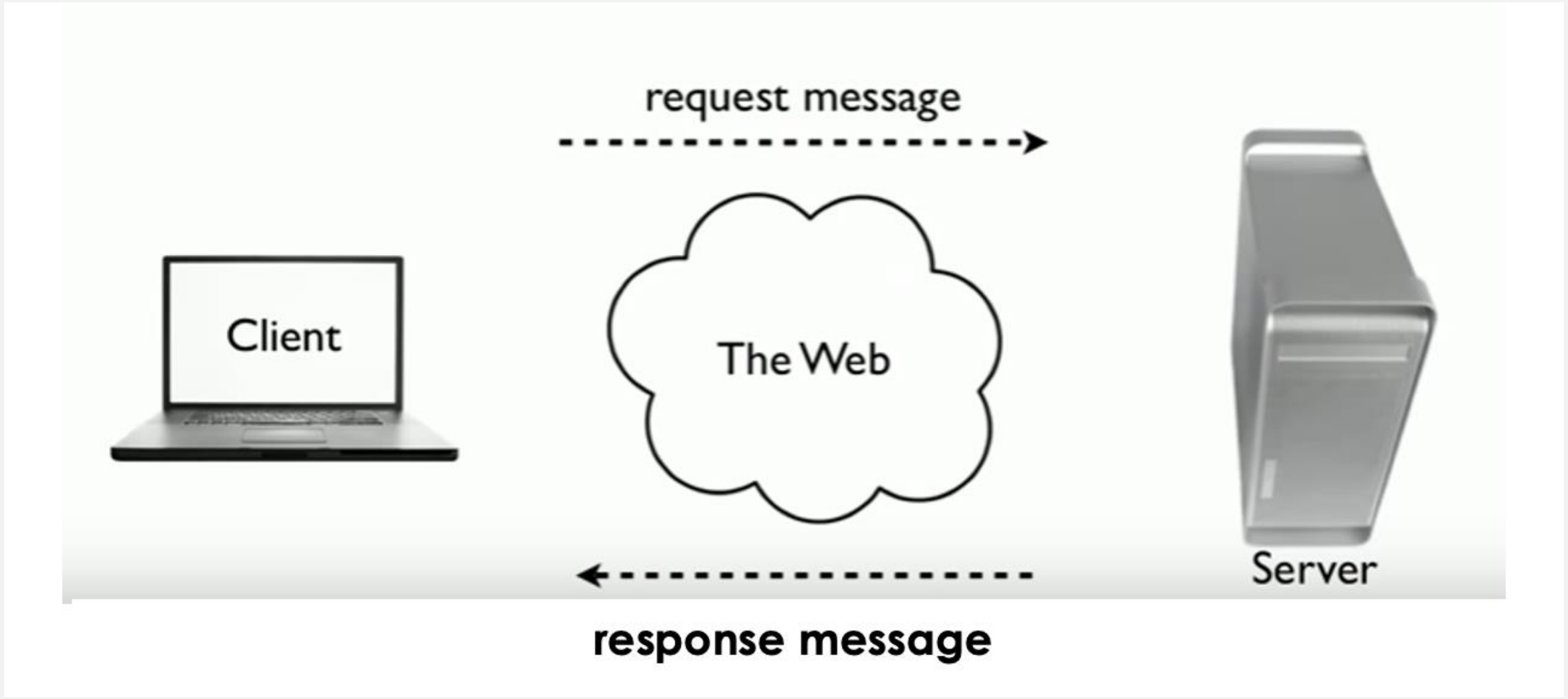
- web page stays in place
- parts of the web page are updated

How are web 2.0 requests handled between client and server?

Web Services !

WEB SERVICES

- A framework for a conversation between computers over the web



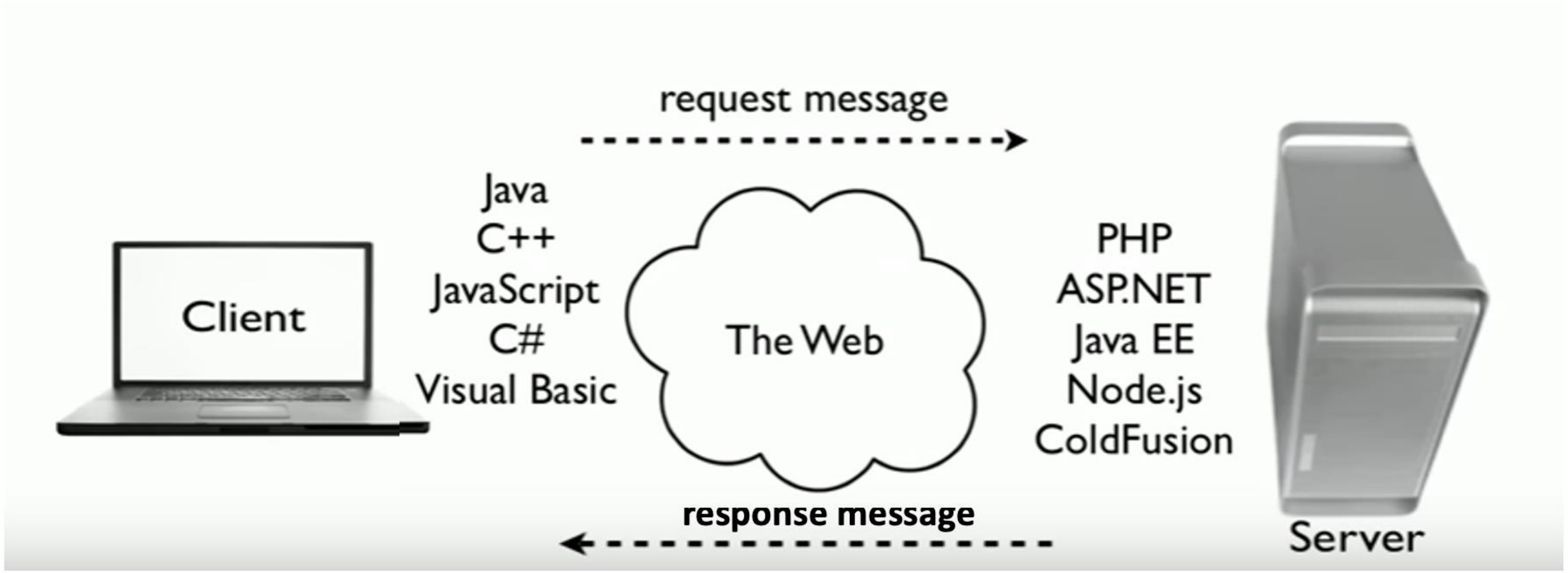
WEB SERVICES

- 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

WEB SERVICES

- The web service hides its complexity behind the API



REpresentative

State

Transfer

REST IS AN *ARCHITECTURAL STYLE*

Modern Architectural Style:

Colonial Architectural Style:

REST

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

REST

- 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.
  - 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) are supported, when HTTP is used.

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

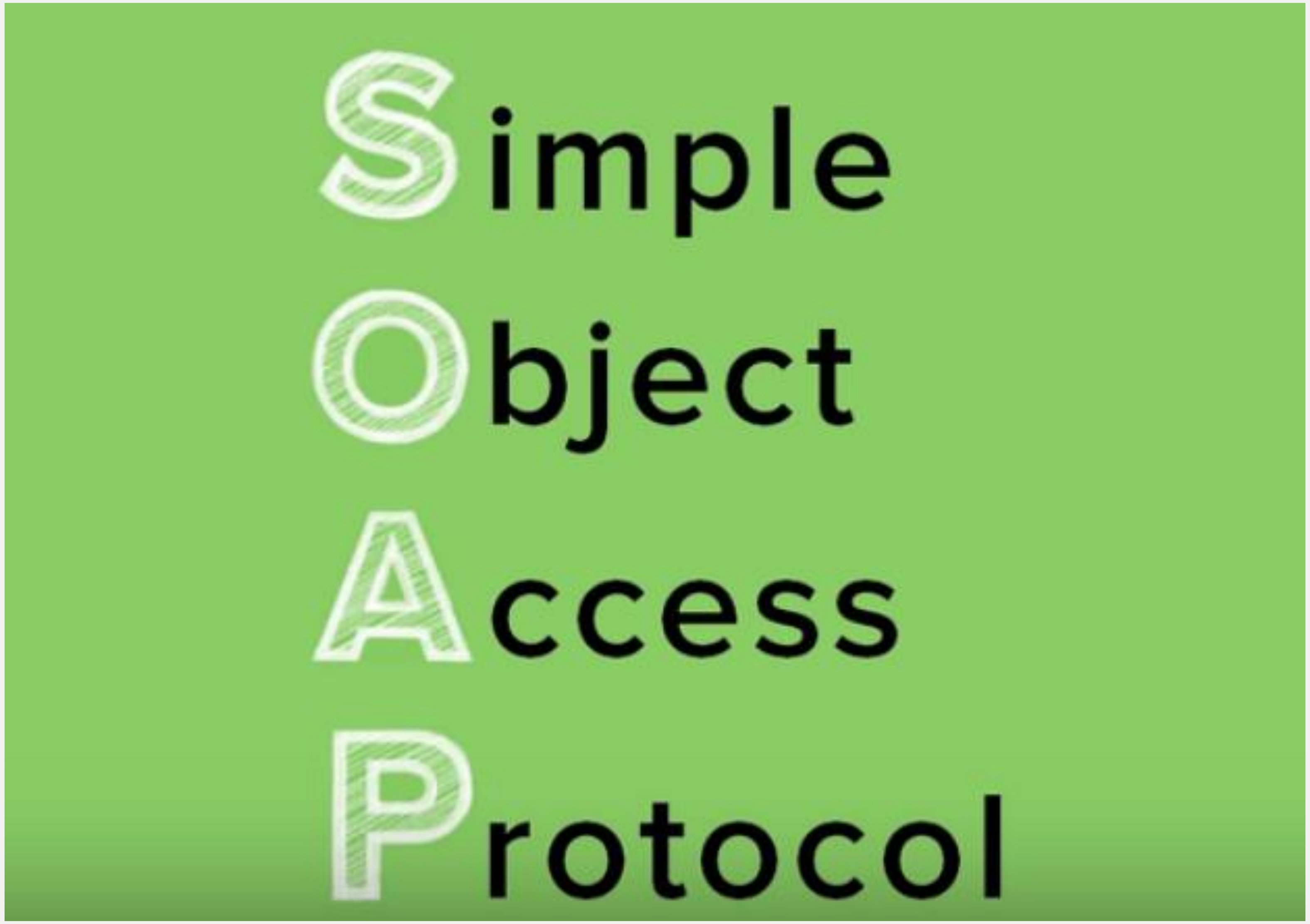
(through caching & compression)

Scalability

(gateways distribute traffic, caching, statelessness allows different intermediaries)

User Perceived Performance

(code on demand, client validation, caching)



Simple

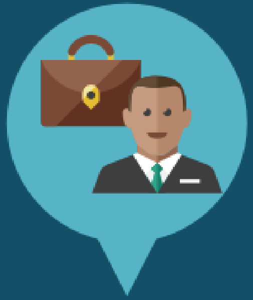
Object

Access

Protocol

REST

REST (Representational State Transfer) was Created in 2000 by Roy Fielding in UC, Irvine. Developed in an academic environment, this protocol embraces the philosophy of the open Web.

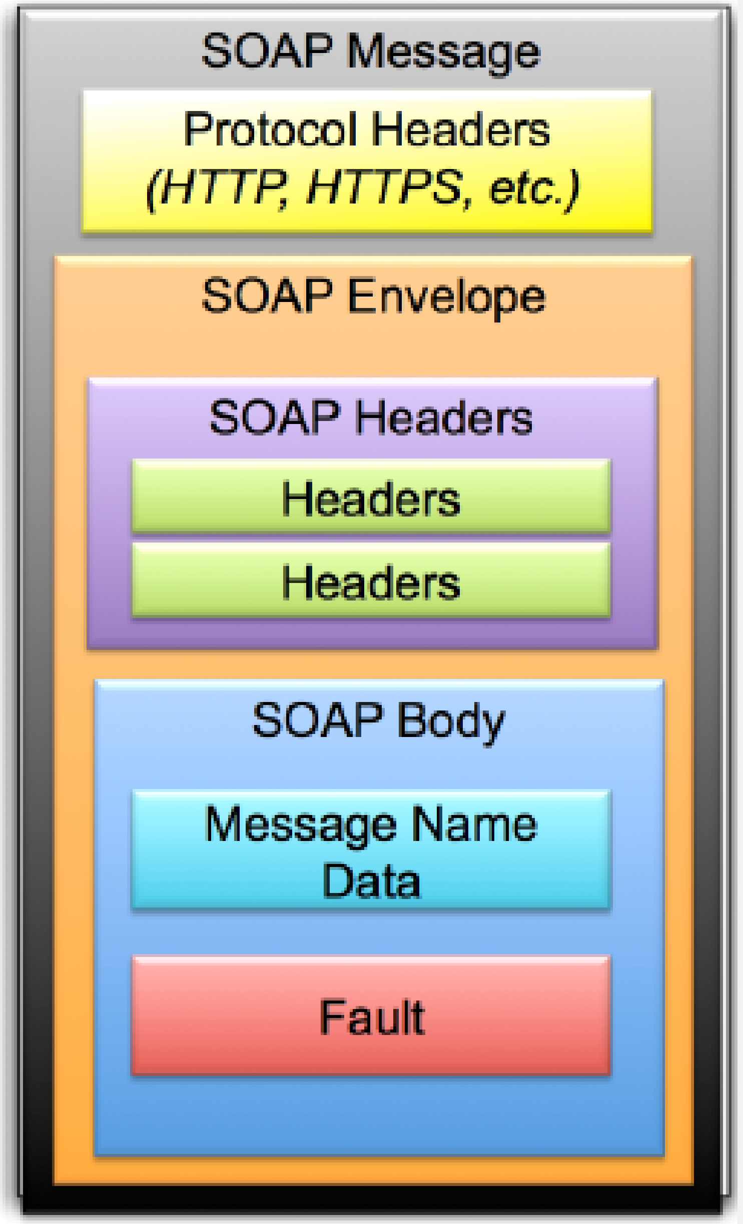


SOAP

SOAP (Simple Object Access Protocol), was created in 1998 by Dave Winer et al in collaboration with Microsoft. Developed by a large software company, this protocol addresses the goal of addressing the needs of the enterprise market.

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



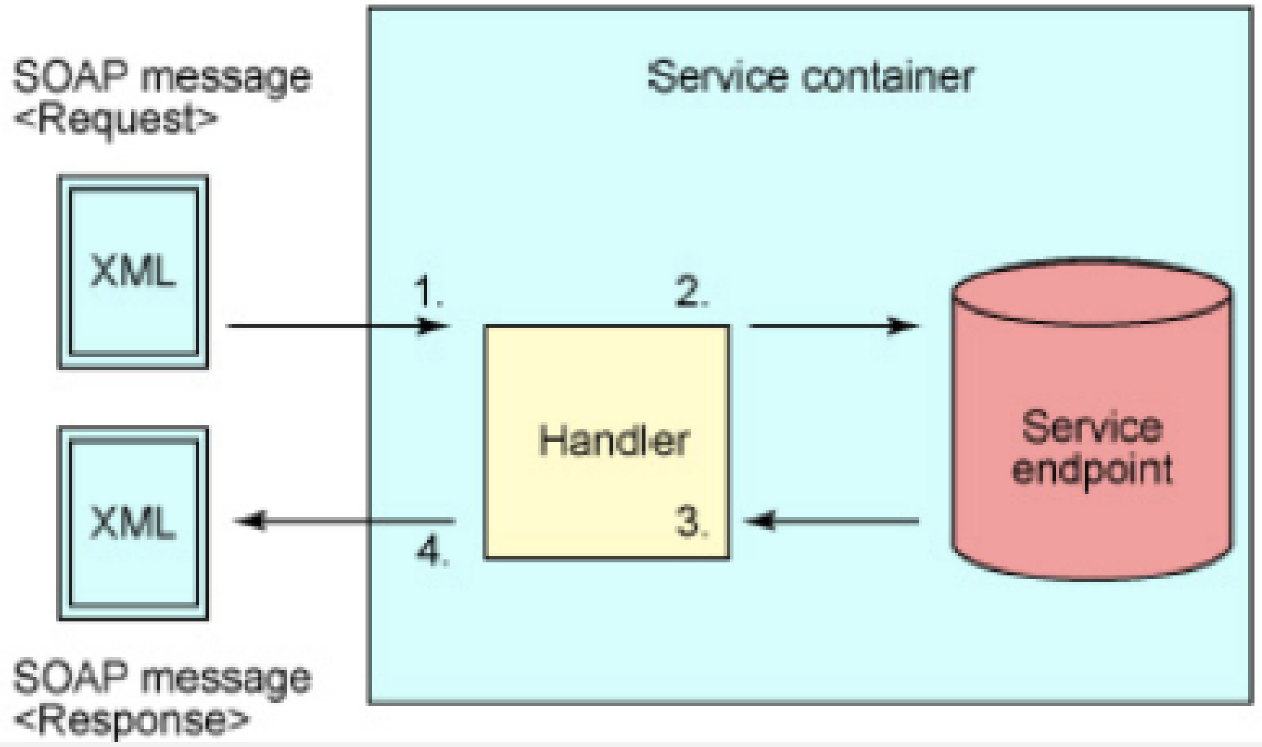
SOAP

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

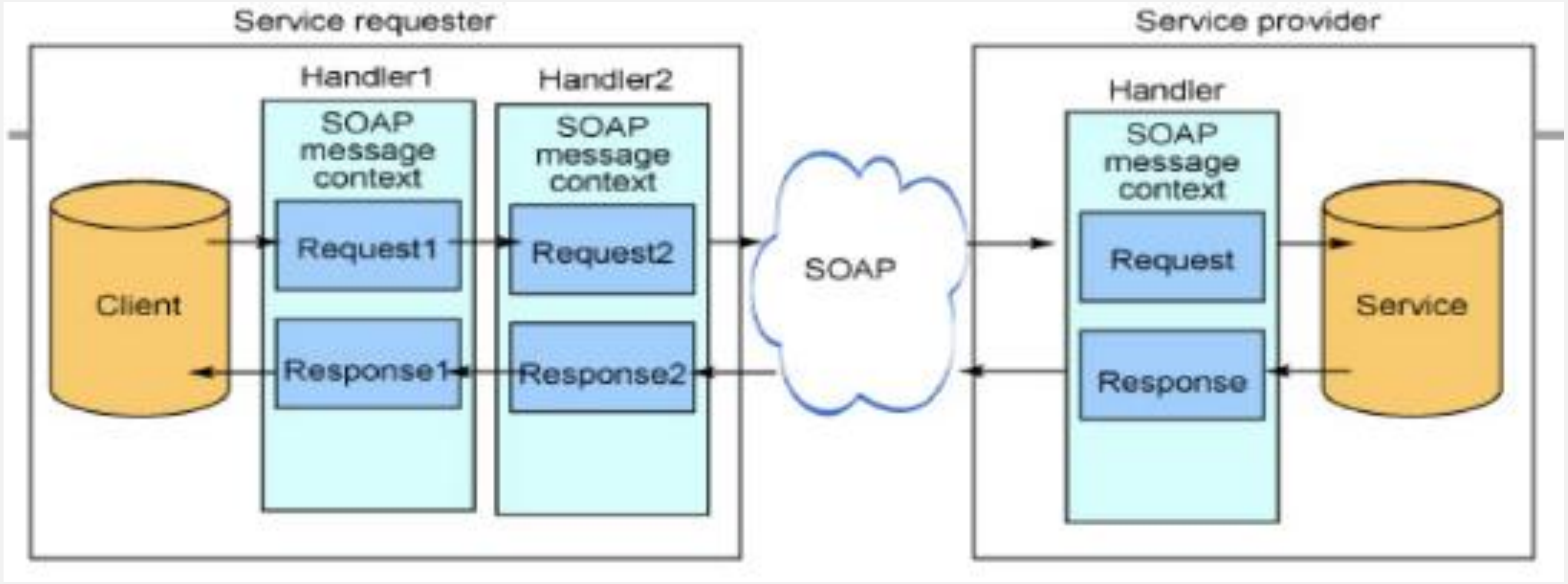
SOAP builds an XML based protocol on top of HTTP or some other protocol according to the rules described in the WSDL for that Web Service.

SOAP Handlers

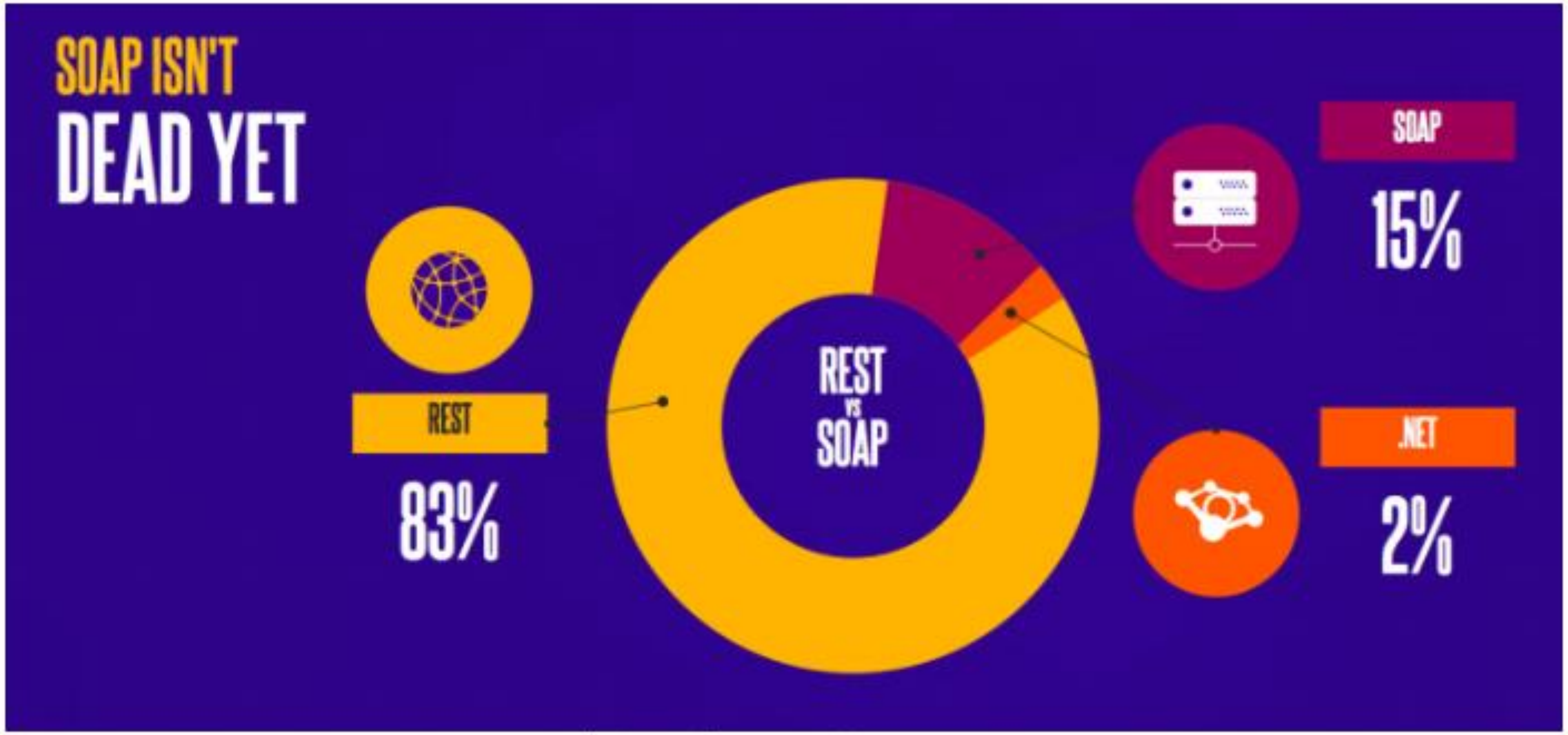
- **Handlers** are pluggable classes that can be associated with a Web service or Web service client to provide pre-processing or post-processing of XML messages.
  - Ex: logging XML traffic through a Web service
  - Ex: measure performance by accessing the SOAP header to insert initial and finish times between two checkpoints



SOAP


Service requester
Service provider
Handler1
Handler2
Handler
SOAP message context
SOAP message context
SOAP message context
Client
Request1
Request2
Response1
Response2
SOAP
Request
Response
Service

According to this year's report, one area that continues to come up year after year is the balance of power between SOAP and REST. Although REST dominates the scene, there is still a decent percentage SOAP APIs out there that can't be ignored – at least 15% based on Cloud Elements' experience.



The State of API Integration report 2017