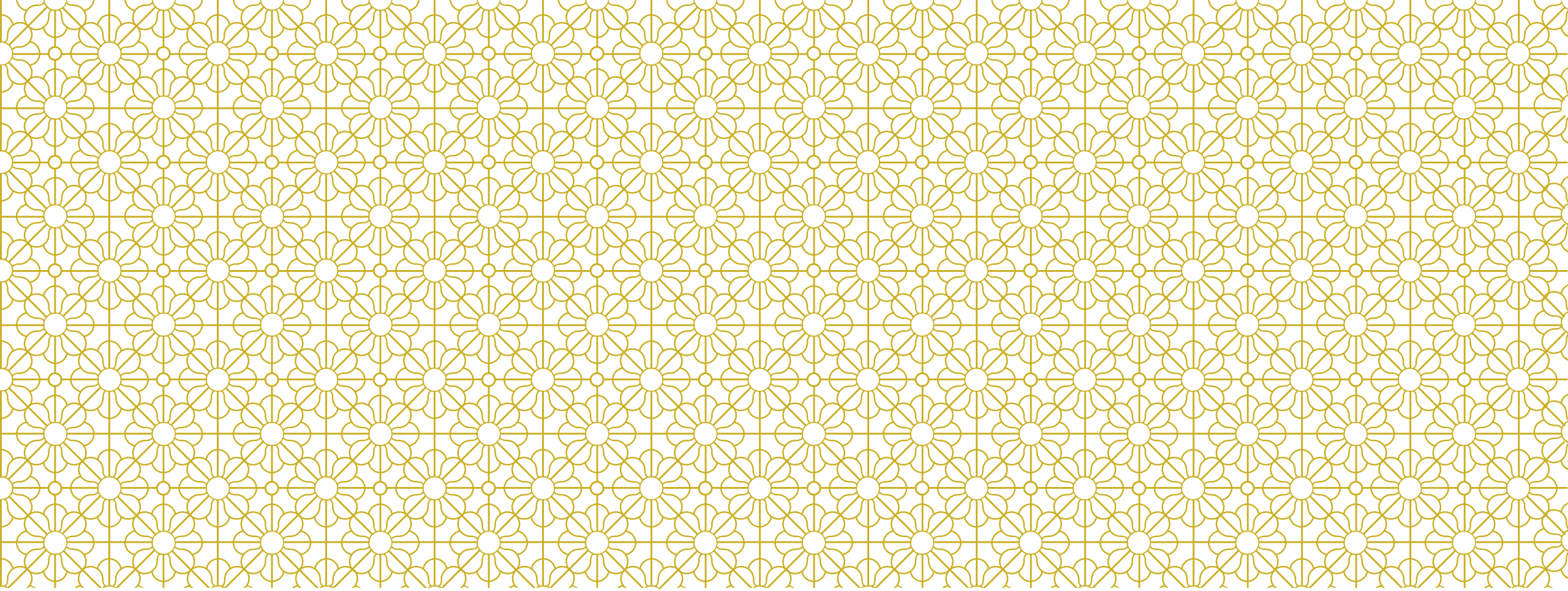


INTRODUCTION TO REPRESENTATIONAL STATE TRANSFER

Andrew Sheehan
Metropolitan College

WHAT IS A RESTFUL API?

Architecture. How things Communicate with each other.

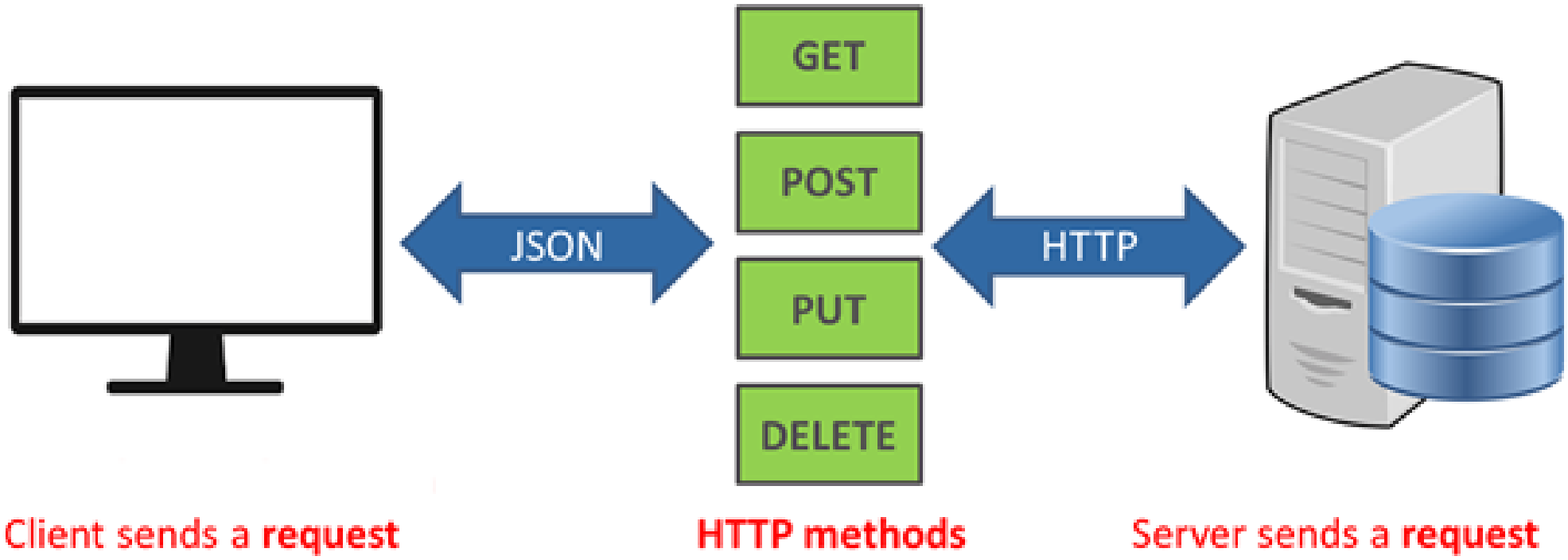
Created by Roy Fielding
(Ph.d @UC Urvine)

WHAT IS A RESTFUL API?

REST is stateless.

After backend responds, it forgets about you.

HIGH LEVEL OVERVIEW



CLIENTS
RESTFUL TENANTS

Frontend handles the rendering and interactions

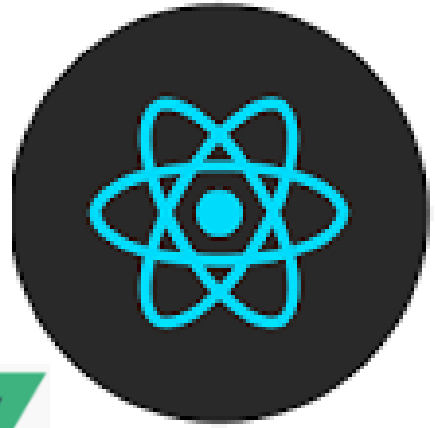
The back end interacts with the data.

CLIENTS
RESTFUL

Clients: Tesla, Alexa, Samsung Gear … fetching or updating information.

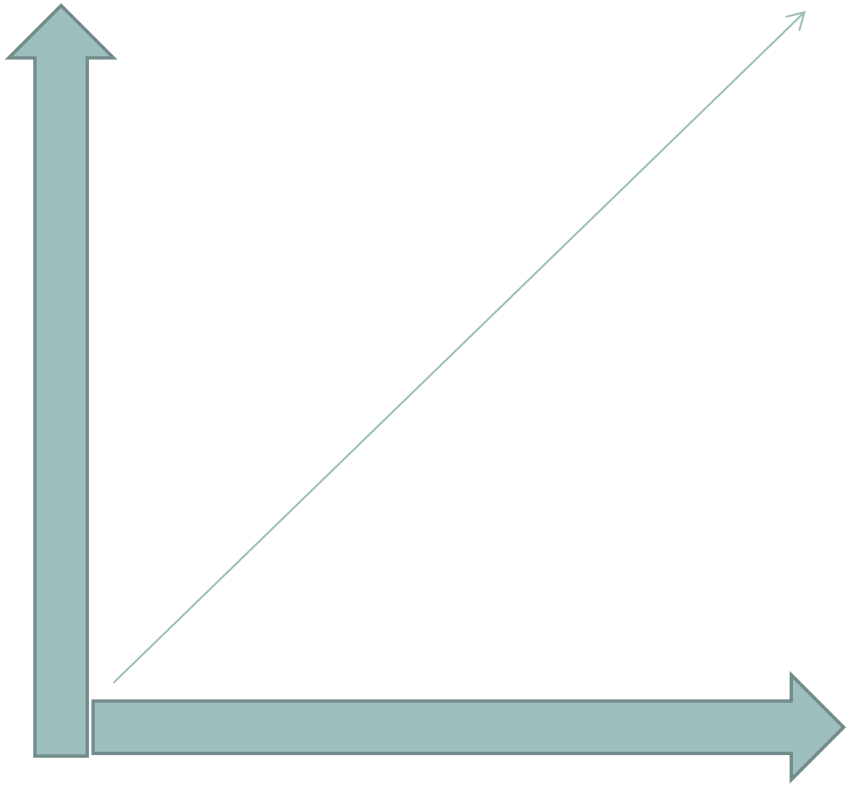
Tesla dashboard

RESTFUL
STATELESS



Typical clients (ReactJS, Angular, SPA's, PHP, VueJS – applications with state management.

RESTFUL API'S ARE ENDPOINTS



Endpoints are only an URL.

Your middleware & application services need to be designed to scale.

Think about it now. Plan.

CLIENTS
METHODS

It does not matter what language the server code is written with.

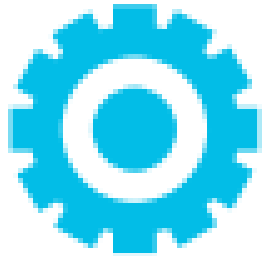
CLIENTS
METHODS

Common HTTP **Request Methods**

GET	Retrieves data from a remote server. It can be a single resource or a list of resources.
POST	Creates a new resource on the remote server. *
PUT	Updates the data on the remote server.
DELETE	Deletes data from the remote server.

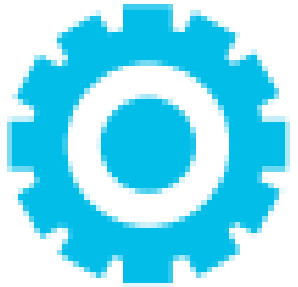
SERVER RESPONDING (NODE, PHP, JAVA, ETC..)

SET YOUR HEADERS



```
header("Access-Control-Allow-Origin: * ");
header("Access-Control-Allow-Methods: POST, GET, PUT");
header("Content-Type:  application/json");
```

EXAMPLE
REQUEST WITH FETCH (JAVASCRIPT)



```
const req_config = new Request(“ http://localhost:3000/someURL” , {
    method:       “ GET ” ,
    mode:         “ cors ”,
    redirect:     “ follow ” ,
    credentials:  “ include ”  ,
    headers: new Headers({ "Content-Type": “application/json" })
});

fetch(req_config).then(response => { return response.json();})
    .then(data => {    // do something with the data…     });
```

TYPICAL RESPONSE
JSON

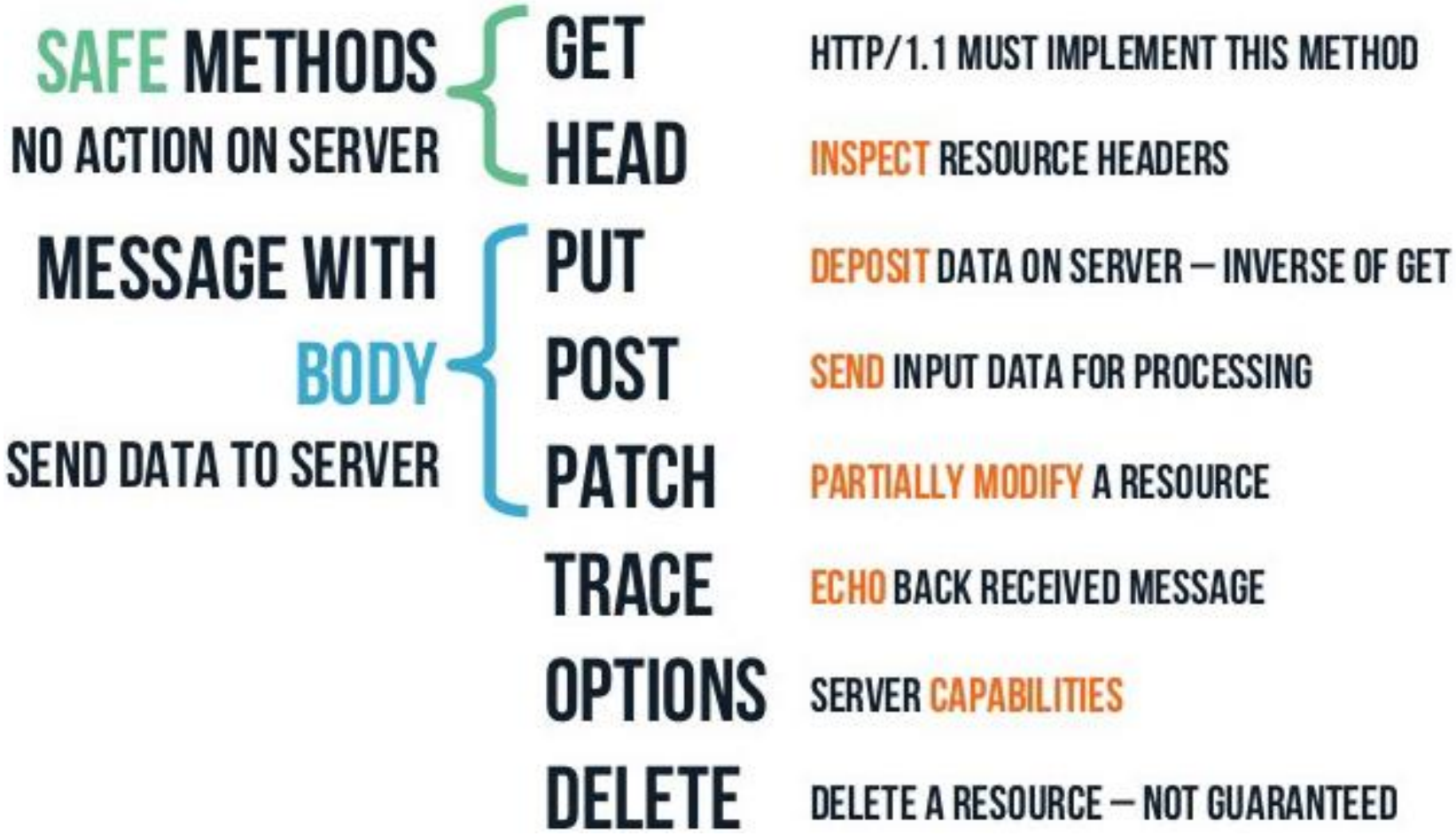
XML is more difficult to parse.

JSON is not. Ready to use.

RESTFUL
URL EXAMPLES

POST	/api/note	Post a new note
GET	/api/note/1	Fetch note. ID: 1
PUT	/api/note/1	Update note. ID: 1
DELETE	/api/note/1	Delete note. ID: 1

REST API
HTTP METHODS



REST API
HTTP METHODS

RESTful APIs have various methods to indicate the type of operation we are going to perform with this API —

- **GET**—To get a resource or collection of resources.
- **POST**—To create a resource or collection of resources.
- **PUT/PATCH**—To update the existing resource or collection of resources.
- **DELETE**—To delete the existing resource or the collection of resources.

REST APIS
AMAZON'S MANDATES (BEZOS)

1. All teams will henceforth expose their data and functionality through service interfaces.

2. Teams must communicate with each other through these interfaces.

3. There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.

4. It doesn't matter what technology they use. HTTP, Corba, Pubsub, custom protocols—doesn't matter. Bezos doesn't care.

5. All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.

6. **Anyone who doesn't do this will be fired.**

REST APIS
URL STRUCTURES

Use Nouns - not verbs

/products

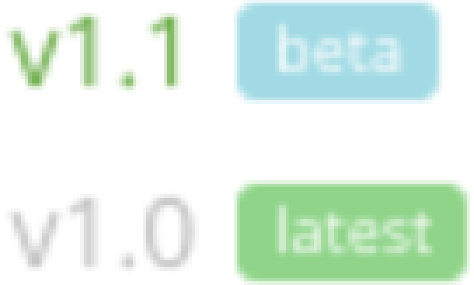
/getAllProducts

REST APIS
USE THE APPROPRIATE **HTTP CODES**

Examples

- **200 OK**—This is most commonly used HTTP code to show that the operation performed is successful.
- **201 CREATED**—This can be used when you use POST method to create a new resource.
- **400 BAD REQUEST**—This can be used when client side input validation fails.

REST APIS
VERSIONS



Indicate the API version in your Request Headers. Not in the URL

Example:

X-Endpoint-Version: v1.22.1

REST APIS
SET YOUR HEADERS

v1.1 beta

v1.0 latest

```
1  var myHeaders = new Headers();
2  myHeaders.append('Content-Type', 'image/jpeg');
3
4  var myInit = {
5    method: 'GET',
6    headers: myHeaders,
7    mode: 'cors',
8    cache: 'default'
9  };
10
11 var myRequest = new Request('flowers.jpg', myInit);
12
13 myContentType = myRequest.headers.get('Content-Type'); // returns 'image/jpeg'
```

DIFFERENCES
REST VS SOAP

Simple Object Access Protocol

SOAP provides the following advantages when compared to REST:

- Language, platform, and transport independent (REST requires use of HTTP)
- Works well in distributed enterprise environments (REST assumes direct point-to-point communication)
- Standardized
- Provides significant pre-build extensibility in the form of the WS* standards
- Built-in error handling
- Automation when used with certain language products

REST is easier to use for the most part and is more flexible. It has the following advantages when compared to SOAP:

- Uses easy to understand standards like swagger and OpenAPI Specification 3.0
- Smaller learning curve
- Efficient (SOAP uses XML for all messages, REST mostly uses smaller message formats like JSON)
- Fast (no extensive processing required)
- Closer to other Web technologies in design philosophy

<EXAMPLE>

SOAP Simple Object Access Protocol

The response from the service:

```
HTTP/1.1 200 OK
Date: Fri, 22 Nov 2013 21:09:44 GMT
Server: Apache/2.0.52 (Red Hat)
SOAPServer: SOAP::Lite/Perl/0.52
Content-Length: 566
Connection: close
Content-Type: text/xml; charset=utf-8
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
   <namesp1:easter_dateResponse
xmlns:namesp1="http://www.stgregorioschurchdc.org/Calendar">
<s-gensym3 xsi:type="xsd:string">2014/04/20</s-gensym3>
</namesp1:easter_dateResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<EXAMPLE>

SOAP Simple Object Access Protocol

A request from the client:

```
POST http://www.stgregorioschurchdc.org/cgi/websvccal.cgi HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: "http://www.stgregorioschurchdc.org/Calendar#easter_date"
Content-Length: 479
Host: www.stgregorioschurchdc.org
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
<?xml version="1.0"?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:cal="http://www.stgregorioschurchdc.org/Calendar">
<soapenv:Header/>
<soapenv:Body>
   <cal:easter_date soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <year xsi:type="xsd:short">2014</year>
</cal:easter_date>
</soapenv:Body>
</soapenv:Envelope>
```